ДУІКТ

ВСЕУКРАЇНСЬКА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ

ТЕХНОЛОГІЇ ЦИФРОВОГО РОЗВИТКУ: ПРОГРАМНІ СИСТЕМИ ТА ДЕЦЕНТРАЛІЗАЦІЯ

ЗБІРНИК ТЕЗ

1–3 травня 2026 року

Київ · 2026

Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація». Збірник тез. – К.: ДУІКТ, 2026.

Даний збірник містить тези учасників конференції, представлених на Всеукраїнській науково-технічній конференції «Технології цифрового розвитку: програмні системи та децентралізація», яка проводилась 01-03 травня 2026 р. на кафедрі Технологій цифрового розвитку Навчально-наукового інституту інформаційних технологій Державного університету інформаційно-комунікаційних технологій, м. Київ.

Робоча мова конференції – українська та англійська.

Вчений секретар конференції

Чугреєв Кирилл – Державний університет інформаційно-комунікаційних технологій

моб.тел.+38(063)946-23-83

e-mail: k.chugreev@duikt.edu.ua

ОРГАНІЗАТОРИ КОНФЕРЕНЦІЇ

Державний університет інформаційно-комунікаційних технологій

Навчально-науковий інститут інформаційних технологій

Кафедра Технологій цифрового розвитку

ПРОГРАМНИЙ КОМІТЕТ

Володимир ШУЛЬГА - ректор Державного університету інформаційно-комунікаційних технологій, доктор історичних наук, старший дослідник

Олександр КОРЧЕНКО - перший проректор Державного університету інформаційно-комунікаційних технологій член-кореспондент НАН України, доктор технічних наук, професор, лауреат Державної премії України в галузі науки і техніки, Заслужений діяч науки і техніки України**Марина ПЕТЧЕНКО -** проректор з науково-педагогічної роботи та соціального розвитку Державного університету інформаційно-комунікаційних технологій, кандидат економічних наук, доцент

Катерина НЕСТЕРЕНКО - доктор технічних наук, професор, в.о. директора навчально-наукового інституту інформаційних технологій Державного університету інформаційно-комунікаційних технологій

Вікторія ЖЕБКА - доктор технічних наук, професор, завідувач кафедри Технологій цифрового розвитку Державного університету інформаційно-комунікаційних технологій

Олександр ТРОФИМЧУК - член-кореспондент НАНУ, доктор технічних наук, професор, директор Інституту телекомунікацій і глобального інформаційного простору НАНУ

Василь ТРИСНЮК - завідувач відділу досліджень навколишнього середовища, доктор технічних наук, професор, Інститут телекомунікацій та глобального інформаційного простору НАН України

Павло СКЛАДАННИЙ - кандидат технічних наук. доцент, завідувач кафедри інформаційної та кібернетичної безпеки імені професора Володимира Бурячка Київського столичного університету імені Бориса Грінченка

Віктор ШЕВЧЕНКО - доктор технічних наук, професор, заступник директора з наукової роботи Інституту програмних систем НАН України

Ольга ЗІНЧЕНКО - доктор технічних наук, доцент, завідувач кафедри Штучного інтелекту Державного університету інформаційно-комунікаційних технологій

Віктор ВИШНІВСЬКИЙ - доктор технічних наук, професор, завідувач кафедри Комп'ютерних наук Державного університету інформаційно-комунікаційних технологій

Наталія ЛАЩЕВСЬКА - кандидат технічних наук, доцент, завідувач кафедри Комп'ютерної інженерії Державного університету інформаційно-комунікаційних технологій

Каміла СТОРЧАК - доктор технічних наук, професор, завідувач кафедри Інформаційних систем та технологій Державного університету інформаційно-комунікаційних технологій

Зміст

Напрям 1 ПРОГРАМНІ СИСТЕМИ ТА АРХІТЕКТУРИ 14

Лисенко Владислав Ігорович, Герцюк Микола Модестович ДИЗАЙН ВЕБ-ЗАСТОСУНКУ З ВИКОРИСТАННЯМ ГЕКСАГОНАЛЬНОЇ АРХІТЕКТУРИ ... 14

Ніколін Кирило Андрійович РОЗРОБКА ВЕБ- ТА МОБІЛЬНОЇ ПЛАТФОРМИ ЛАБОРАТОРНИХ АНАЛІЗІВ ДЛЯ МЕДИЧНОЇ ЛАБОРАТОРІЇ «MONOLAB\.. 17

Крєпкій Ігор Тарасович КЛІЄНТ-СЕРВЕРНА СИСТЕМА ПОТОКОВОЇ ПЕРЕДАЧІ АУДІО В РЕАЛЬНОМУ ЧАСІ НА БАЗІ ANDROID SDK ТА UDP ... 20

Тимофіїв Сергій Іванович, Вагін Роман Русланович МЕТОД ОПТИМІЗАЦІЇ ПАРАМЕРТІВ МОБІЛЬНИХ ГРАФІЧНИХ СИСТЕМ 22

Бойченко Нікіта Олегович СПРОЩЕННЯ ВЗАЄМОДІЇ МОНІТОРИНГУ ІНВЕРТОРА ЧЕРЕЗ ТЕЛЕГРАМ-БОТ З АВТОМАТИЧНИМИ СПОВІЩЕННЯМИ ТА ЗРОЗУМІЛИМ ПОДАННЯМ ДАНИХ 25

Цайгер Юрій Петрович ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ ОСОБИСТИХ ФІНАНСІВ ЗАСОБАМИ ASP.NET CORE MVC 27

Артеменко Анастасія Ярославівна РЕАЛІЗАЦІЯ МЕХАНІЗМУ КОНТРОЛЮ ТА АНАЛІЗУ ПРИЙОМУ ЛІКІВ У ВЕБ-ЗАСТОСУНКУ НА БАЗІ ASP.NET CORE 29

Ходак Денис Павлович ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ЛЮДСЬКИМИ РЕСУРСАМИ НА БАЗІ ASP.NET CORE MVC 31

Лавренюк Богдан Юрійович ДЕСКТОПНА СИСТЕМА КЕРУВАННЯ VPN-ПІДКЛЮЧЕННЯМИ ЯК ЗАСІБ БЕЗПЕЧНОГО ДОСТУПУ ДО ІНФОРМАЦІЙНИХ РЕСУРСІВ 34

Лавренюк Богдан Юрійович МОДУЛЬНА АРХІТЕКТУРА НАСТІЛЬНОГО ДОДАТКА ДЛЯ КЕРУВАННЯ ЛОКАЛЬНИМИ МЕРЕЖЕВИМИ ПРОФІЛЯМИ 36

Романчук Анастасія Анатоліївна СИСТЕМА АВТОМАТИЗАЦІЇ КОНТРОЛЮ ЧЕРГ ТА ІДЕНТИФІКАЦІЇ ОСІБ НА МІЖНАРОДНОМУ КОРДОНІ 37

Стрельбіцький Вадим Юрійович МІКРОСЕРВІСНА АРХІТЕКТУРА REST API ДЛЯ СИСТЕМИ SERVICE DESK 40

Саєнко Кирил Олександрович, Данильченко Валентина Миколаївна РОЗРОБКА ФУНКЦІОНАЛЬНОЇ МОДЕЛІ ТА АРХІТЕКТУРИ ІНТЕРНЕТ-МАГАЗИНУ..... 43

Лазневий Андрій Валерійович ІНФОРМАЦІЙНА СИСТЕМА ОБЛІКУ ПАЦІЄНТІВ МЕДИЧНОГО ЦЕНТРУ 46

Закаблук Едуард Євгенович, Вишнівський Віктор Вікторович ПЕРЕВАГИ ВИКОРИСТАННЯ ОПЕРАЦІЙНИХ СИСТЕМ З ВІДКРИТИМ КОДОМ ДЛЯ ПОБУДОВИ КОРПОРАТИВНОЇ СЕРВЕРНОЇ ІНФРАСТРУКТУРИ..... 48

Санкіна Яна Сергіївна ПРОЄКТУВАННЯ СЛАБКОПОВ'ЯЗАНОЇ АРХІТЕКТУРИ ІГРОВОГО ЗАСТОСУНКУ НА БАЗІ ПОДІЄВОЇ МОДЕЛІ..... 51

Антонов Олександр Лаврентійович РЕАЛІЗАЦІЯ СИСТЕМИ АВТОРИЗАЦІЇ ТА ВЗАЄМОДІЇ КОРИСТУВАЧА У ВЕБ-ДОДАТКУ «ОНЛАЙН БІБЛІОТЕКА» 53

Бахмут Михайло ПРОЕКТУВАННЯ АРХІТЕКТУРИ ГРИ ТИПУ ROGUELIKE НА ОСНОВІ UNREAL ENGINE 5 55

Мирончук Микола Володимирович, Сагайдак Віктор Анатолійович СИСТЕМА ІНТЕЛЕКТУАЛЬНОЇ МАРШРУТИЗАЦІЇ ЦИФРОВИХ ДОКУМЕНТІВ НА ОСНОВІ ЧАТ-ІНТЕРФЕЙСІВ 57

Журба Володимир Ігорович РОЗРОБКА WEB APPLICATION E-LIBRARY НА МОВІ PYTHON З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ DJANGO..... 59

Бєляєв Данило Анатолійович МІКРОСЕРВІСНА АРХІТЕКТУРА ЛОГІСТИЧНОЇ CRM-СИСТЕМИ ДЛЯ АСИНХРОННОЇ ОБРОБКИ ВИСОКОНАВАНТАЖЕНИХ ЗАДАЧ МАРШРУТИЗАЦІЇ 61

Ступак Назар Олександрович, Гордієнко Катерина Олександрівна ДОЦІЛЬНІСТЬ ПРОЕКТУВАННЯ СИСТЕМИ ПРАКТИЧНОГО ЗАСВОЄННЯ МОВИ SQL 63

Заволокін Віталій Юрійович, Іщеряков Сергій Михайлович РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ ПОГОДНИХ УМОВ У ВИГЛЯДІ TELEGRAM-БОТА НА ОСНОВІ МОВИ ПРОГРАМУВАННЯ PYTHON 65

Коломієць Артем Григорович АРХІТЕКТУРА ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ..... 67

Марущак Артем Русланович РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ОНЛАЙН-ПРОДАЖУ ПАПЕРОВИХ КНИЖОК..... 70

Фещенко Богдан Вікторович ГІБРИДНА АРХІТЕКТУРА МУЗИЧНОЇ ПЛАТФОРМИ НА БАЗІ ПЕРИФЕРІЙНИХ ОБЧИСЛЕНЬ ТА СЕРВЕРЛЕС-ТЕХНОЛОГІЙ..... 73

Іванов Дмитро Андрійович WEB-ДОДАТОК ДЛЯ ЗНАЙОМСТВ І СОЦІАЛЬНОЇ ВЗАЄМОДІЇ З ВИКОРИСТАННЯМ GO, TYPESCRIPT ТА MARIADB 76

Микитюк Дмитро Сергійович ОСОБЛИВОСТІ РОЗРОБКИ SELF-HOSTED РІШЕННЯ ДЛЯ ОРГАНІЗАЦІЇ ТА ПАРСИНГУ ВЕЛИКОГО МАСИВУ МЕДІА ДАНИХ НА ОСНОВІ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ..... 78

Сич Богдан Олегович РОЗРОБКА СИСТЕМИ АНАЛІЗУ АВТОСЕРВІСУ ДЛЯ ПІДВИЩЕННЯ ЯКОСТІ ОБСЛУГОВУВАННЯ 82

Кіч Роман Олександрович ЗАБЕЗПЕЧЕННЯ ЦІЛІСНОСТІ ДАНИХ ПІД ЧАС ОБРОБКИ ТРАНЗАКЦІЙ У ВЕБ-СИСТЕМАХ ЕЛЕКТРОННОЇ КОМЕРЦІЇ..... 85

Казанцев Нікіта Євгенійович ОСОБЛИВОСТІ РОБОТИ ІНТЕРАКТИВНОЇ НАВЧАЛЬНОЇ ПЛАТФОРМИ "SKILLHUB" ДЛЯ ВИВЧЕННЯ ІТ-КУРСІ..... 87

Сокол Віктор Іванович АКТУАЛЬНІСТЬ СИСТЕМИ ПІДТРИМКИ КЛІЄНТІВ ІНТЕРНЕТ-МАГАЗИНУ ЕЛЕКТРОНІКИ..... 89

Мартиновченко Максим Віталійович ПЕРСПЕКТИВИ РЕАЛІЗАЦІЇ WEB-ПЛАТФОРМИ LOGITRACK ДЛЯ МОНІТОРИНГУ ЛОГІСТИКИ ТА ОЦІНЮВАННЯ РИЗИКІВ ДОСТАВКИ..... 93

Замай Олександр Олександрович ОСОБЛИВОСТІ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ОНЛАЙН-ПРОДАЖУ КВИТКІВ НА БЛАГОДІЙНІ КОНЦЕРТИ..... 96

Реул Нікіта Євгенович АРХІТЕКТУРА СИСТЕМИ АВТЕНТИФІКАЦІЇ НА ОСНОВІ WEBAUTHN ТА PASSKEYS 98

Коломієць Назарій Ярославович СПРОЩЕННЯ ВЗАЄМОДІЇ ТА ОПТИМІЗАЦІЯ ОБРОБКИ ДАНИХ У ВЕБ-ПЛАТФОРМІ ОРЕНДИ ЖИТЛА НА ОСНОВІ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ 100

Сидоров Гліб Сергійович МЕТОД КОМПОНЕНТНОГО ПІДХОДУ В РОЗРОБЦІ 2D ПЛАТФОРМЕРА НА БАЗІ UNITY 102

Здоровенко Юлія Сергіївна ОСОБЛИВОСТІ ВЕДЕННЯ ДОМАШНЬОЇ БІБЛІОТЕКИ ТА ТРЕКІНГУ ПРОЧИТАНОГО МАТЕРІАЛУ ЗА ДОПОМОГОЮ ВЕБ-ЗАСТОСУНКУ SHELFMATE 105

Новостройний Дмитро Андрійович МОДЕЛЮВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ АВТОМАТИЗАЦІЇ ОБРОБКИ ЗАЯВОК 107

Мельник Владислав Олександрович ПРОГРАМНА СИСТЕМА AROMASHOP ДЛЯ ОНЛАЙН-ПРОДАЖУ ТА ПІДБОРУ ПАРФУМЕРНОЇ ПРОДУКЦІЇ 110

Напрям 2 ДЕЦЕНТРАЛІЗОВАНІ ІНФОРМАЦІЙНІ СИСТЕМИ 113

Чохленко Богдан Андрійович ДЕЦЕНТРАЛІЗАЦІЯ ДАНИХ У СУЧАСНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ: НЕОБХІДНІСТЬ ТА ПЕРЕВАГИ 113

Доманський Владислав Сергійович, Бажан Тетяна Олександрівна АВТОНОМНИЙ ВЕБ-ДОДАТОК ДЛЯ УПРАВЛІННЯ НОТАТКАМИ 115

Моргунов Артем Олександрович, Тьотченко Денис Григорович АДАПТИВНИЙ МЕТОД ФОРМУВАННЯ КВОРУМУ В ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМАХ 118

Жебка Вікторія Вікторівна МЕТОД АДАПТИВНОГО РОЗМІЩЕННЯ РЕПЛІК ДАНИХ У ДЕЦЕНТРАЛІЗОВАНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ 121

Жебка Сергій Валентинович СЦЕНАРНО-АДАПТИВНИЙ МЕТОД ОЦІНЮВАННЯ СТІЙКОСТІ ВИБОРУ DLT-ПЛАТФОРМ 124

Напрям 3 Розробка та супровід програмного забезпечення 128

Грицанюк Дмитро Сергійович, Читулян Вадим Олегович ПРОЄКТУВАННЯ АРХІТЕКТУРИ WEB-ДОДАТКУ ДЛЯ УПРАВЛІННЯ АВТОПАРКОМ ПРИВАТНОГО ПІДПРИЄМСТВА НА ОСНОВІ КЛІЄНТ-СЕРВЕРНОЇ МОДЕЛІ З REST API 128

Колпаков Дмитро Олегович ВЕБ-ПЛАТФОРМА БРОНЮВАННЯ КОВОРКІНГ-ПРОСТОРІВ ІЗ ПІДТРИМКОЮ КОМАНДНОЇ РОБОТИ НА ОСНОВІ JAVASCRIPT-ТЕХНОЛОГІЙ 131

Ляшенко Ярослав Володимирович РОЗРОБКА ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ПРИКЛАДІ 2D ГРИ ChronoVortex 134

Мезецький Владислав Олегович, Чернявський Ждан Анатолійович АВТОМАТИЗАЦІЯ ДОКУМЕНТООБІГУ ФІНАНСОВОЇ КОМПАНІЇ 137

Оводов Ростислав Олексійович, Гавор Артур Станіславович ПРОЄКТУВАННЯ ТА ТЕХНІЧНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ПРОДУКТИВНІСТЮ З ЕЛЕМЕНТАМИ ГЕЙМІФІКАЦІЇ 139

Подольський Дмитро Олегович РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ КОРПОРАТИВНОГО УПРАВЛІННЯ ПРОЄКТАМИ ТА РЕСУРСАМИ З ІНТЕГРАЦІЄЮ ФІНАНСОВОГО МЕНЕДЖМЕНТУ 141

Старцев Микола Сергійович, Читулян Вадим Олегович АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ПРОЕКТУВАННЯ МОБІЛЬНОЇ СИСТЕМИ ВИЯВЛЕННЯ ВИБУХОНЕБЕЗПЕЧНИХ ПРЕДМЕТІВ ІЗ ЗАСТОСУВАННЯМ БПЛА 144

Грисик Владислав Олександрович РОЗВИТОК КОРИСТУВАЧІВ У ЖАНРІ RUNNER 147

Гридько Богдан Андрійович РОЛЬОВО-ОРІЄНТОВАНА АРХІТЕКТУРА КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБСИСТЕМИ МЕНЕДЖМЕНТУ ТОВАРНИХ ЗАПАСІВ 150

Новачук Ігор Сергійович, Читулян Вадим Олегович РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ SAFE DRIVE ДЛЯ ІНТЕРАКТИВНОГО ВИВЧЕННЯ ПРАВИЛ ДОРОЖНЬОГО РУХУ З ЕЛЕМЕНТАМИ СИМУЛЯЦІЇ152

Решетнік Віталій Віталійович, Гордієнко Катерина Олександрівна АНАЛІЗ ВИКОРИСТАННЯ PWA-ТЕХНОЛОГІЙ У ВЕБЗАСТОСУНКУ ДЛЯ ПОШУКУ КУЛІНАРНИХ РЕЦЕПТІВ 154

Кириловський Володимир ВЕБ-ЗАСТОСУНОК ДЛЯ ОБЛІКУ ВІДДІЛЕНЬ ФІНАНСОВОЇ КОМПАНІЇ, РОБОЧИХ МІСЦЬ ANYDESK ТА ЛІМІТІВ ОБОРОТУ 156

Паламарчук Максим Олегович ЦИФРОВИЙ ВЕБ-СЕРВІС ДЛЯ КОНТРОЛЮ ТА КЕРУВАННЯ ДОСТАВКАМИ ПОСИЛОК НА ОСНОВІ ASP.NET CORE 158

Білецький Костянтин Євгенович ІНТЕГРАЦІЯ NYLAS API ДЛЯ СИНХРОНІЗАЦІЇ КАЛЕНДАРІВ ТА АВТОМАТИЗАЦІЇ РЕЗЕРВУВАВННЯ ЗУСТРІЧЕЙ У ЦИФРОВІЙ ПЛАТФОРМІ НА БАЗІ NEXT.JS 160

Гончаренко Дмитро Олександрович ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОГРАМНИХ РІШЕНЬ ДЛЯ ОЦІНЮВАННЯ ОКУПНОСТІ ЕНЕРГОЕФЕКТИВНИХ ІНВЕСТИЦІЙ ТА ОБҐРУНТУВАННЯ ФУНКЦІОНАЛЬНИХ ВИМОГ ЗА МЕТОДОЛОГІЄЮ MoSCoW 162

Дорошенко Юлія Володимирівна АРХІТЕКТУРНІ ПАТЕРНИ ПОБУДОВИ Е-СЕРВІСУ УПРАВЛІННЯ ЗАВДАННЯМИ НА ОСНОВІ SPRING BOOT 3.5 ТА JAVA 25 165

Москаленко Наталія Володимирівна, Сокульський Олег Євгенович, Топольськов Євгеній Олександрович ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ТА ШТУЧНИЙ ІНТЕЛЕКТ 168

Єфімов Максим Олександрович, Аронов Андрій Олексійович ОСОБЛИВОСТІ РОЗРОБКИ МОДУЛЯ ОНЛАЙН-ТЕСТУВАННЯ У ВЕБ-ОРІЄНТОВАНІЙ СИСТЕМІ НАВЧАННЯ 170

Бондаренко Дарина Вадимівна РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ НА ОСНОВІ NODE.JS 173

Купріян Валерія Дмитрівна, Камєнєва Алла Вікторівна РОЗРОБКА КРОСПЛАТФОРМНОГО ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ НАВЧАЛЬНИМ

ПРОЦЕСОМ АКАДЕМІЧНОЇ ГРУПИ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ .NET MAUI 175

Рябко Юлія Василівна АНАЛІЗ ВРАЗЛИВОСТЕЙ JWT-АВТЕНТИФІКАЦІЇ У ВЕБЗАСТОСУНКАХ НА ОСНОВІ NODE.JS ТА МЕТОДИ ЇХ УСУНЕННЯ ... 178

Рябко Юлія Василівна МЕТОДИ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ СЕРВЕРНИХ ЗАСТОСУНКІВ НА ОСНОВІ NODE.JS 180

Сіроєдов Валерій Олександрович АВТОМАТИЗАЦІЯ ОБЛІКУ СИЛОВИХ ТРЕНУВАНЬ ТА ДОСЯГНЕНЬ ЗА ДОПОМОГОЮ ВЕБ-ТЕХНОЛОГІЙ 182

Опанасенко Андрій Юрійович АНАЛІЗ ВРАЗЛИВОСТЕЙ ПРОТОКОЛУ OAUTH 2.0 У ВЕБЗАСТОСУНКАХ ТА МЕТОДИ ЗАХИСТУ ВІД ТИПОВИХ АТАК... 185

Зиков Михайло Віталійович, Гавор Артур Станіславович ПРОГРАМНА РЕАЛІЗАЦІЯ ДЕСКТОПНОГО ЗАСТОСУНКУ CINEMAHUB ДЛЯ КЕРУВАННЯ КАТАЛОГОМ ФІЛЬМІВ ЗАСОБАМИ C++ ТА QT 6 187

Іванова Євгенія Романівна РОЗРОБКА ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 190

Герцюк Микола Модестович ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ ВИКОРИСТАННЯ ГЛОБАЛЬНОГО РЕПОЗИТОРІЮ PUBCHEM, ЯК ФУНДАМЕНТАЛЬНОГО ДЖЕРЕЛА ІНФОРМАЦІЇ ДЛЯ СИСТЕМИ ОЦІНКИ ТОКСИЧНОСТІ ТА НЕБЕЗПЕКИ РЕЧОВИН EXCOMPAN 192

Олейніков Іван Анатолійович ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ІНТЕЛЕКТУАЛЬНОГО АВТОМАТИЗОВАНОГО КЕРУВАННЯ FDM/FFF-3D-ДРУКОМ НА БАЗІ RASPBERRY PI 5 194

Куцовол Іванна Іванівна, Бажан Тетяна Олександрівна МОДЕЛІ ТА МЕТОДИ УПРАВЛІННЯ КАТАЛОГОМ ТОВАРІВ У ЦИФРОВИХ ПЛАТФОРМАХ ЕЛЕКТРОННОЇ КОМЕРЦІЇ 197

Ковшар Дар'я Сергіївна СИНХРОНІЗАЦІЯ ІГРОВОГО СТАНУ В РЕАЛЬНОМУ ЧАСІ ЗАСОБАМИ UNITY NETCODE 199

Петраченко Ігор Андрійович ОСОБЛИВОСТІ ПРОЄКТУВАННЯ WMS-СИСТЕМИ ДЛЯ УПРАВЛІННЯ СКЛАДОМ МАГАЗИНУ ФОРМАТУ DIY 201

Павленко Артем Ігорович, Садовенко Володимир Сергійович РОЗРОБКА CLI-УТИЛІТИ ДЛЯ АВТОМАТИЗАЦІЇ РЕЗЕРВНОГО КОПІЮВАННЯ ТА ВІДНОВЛЕННЯ БАЗ ДАНИХ POSTGRESQL І MYSQL 204

Петрушенко Відана Вікторівна ВИКОРИСТАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНИХ ПАТЕРНІВ ПРОЄКТУВАННЯ ПРИ РОЗРОБЦІ ІГРОВИХ МЕХАНІК МОВОЮ JAVA 207

Аронов Андрій Олексійович ОПТИМІЗАЦІЯ SQL–ЗАПИТІВ ТА СТВОРЕННЯ ІНДЕКСІВ У РЕЛЯЦІЙНИХ БАЗАХ ДАНИХ 209

Кузнецов Павло Олександрович РОЗРОБКА ТА СУПРОВІД WEB-ЗАСТОСУНКУ "ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕКОМЕНДАЦІЇ КІНОФІЛЬМІВ ЗА ВПОДОБАННЯМИ КОРИСТУВАЧІВ" 211

Волошин Нікіта Назарійович РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМПЛЕКСНОГО МОНІТОРИНГУ ТА КОРЕКЦІЇ ІНДИВІДУАЛЬНИХ ПОВЕДІНКОВИХ ПАТЕРНІВ КОРИСТУВАЧА 213

Осадчий Максим Вікторович WEB-САЙТ ДЛЯ ЦЕНТРАЛІЗОВАНОГО МОНІТОРИНГУ РИНКУ ВЖИВАНИХ АВТО 216

Наконечний Владислав Володимирович РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ МОНІТОРИНГУ ТА БРОНЮВАННЯ ПАРКУВАЛЬНИХ МІСЦЬ У МІСЬКОМУ СЕРЕДОВИЩІ 218

Огнева Софія Євгеніївна МЕТОДИ ОРГАНІЗАЦІЇ ОБ'ЄКТНОЇ ВЗАЄМОДІЇ В ІНТЕРАКТИВНИХ СЕРЕДОВИЩАХ НА БАЗІ UNREAL ENGINE 5 220

Мацела Микола Романович АРХІТЕКТУРНІ ПАТЕРНИ ІНТЕГРАЦІЇ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ (LLM) У КЛІЄНТСЬКІ ВЕБЗАСТОСУНКИ 222

Яременко Олександр Андрійович, Ананченко О.Є. ІНТЕЛЕКТУАЛЬНА СИСТЕМА АВТОМАТИЗАЦІЇ НАГАДУВАНЬ НА БАЗІ TELEGRAM-БОТА . 224

Маковець Владислав Валерійович ЗАСТОСУНОК «СТОП ТРИВОГА» - ЕФЕКТИВНЕ ЦИФРОВЕ РІШЕННЯ ДЛЯ РАННЬОГО ВИЯВЛЕННЯ ТРИВОЖНИХ РОЗЛАДІ 227

Велієв Раміль Велієв ЦИФРОВА ПЛАТФОРМА АВТОМАТИЗАЦІЇ КОНТРОЛЮ РОБОЧОГО ЧАСУ ПРАЦІВНИКІВ ТА МОНІТОРИНГУ ВИКОНАННЯ ЗАВДАНЬ 229

Бондаренко Єлизавета Романівна ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ МОБІЛЬНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ЗАПИСУ ПАЦІЄНТІВ НА ПРИЙОМ ДО ЛІКАРЯ 231

Хава Дмитро Віталійович ЦИФРОВА СИСТЕМА ОПТИМІЗАЦІЇ ПРОЦЕСІВ ПРИЙОМУ, ОБРОБКИ ТА УПРАВЛІННЯ ЗАМОВЛЕННЯМИ У РЕСТОРАННОМУ БІЗНЕСІ НА ОСНОВІ PYTHON 233

Дацко Миколай Анатолійович ІМПЛЕМЕНТАЦІЯ БІНАРНОЇ СЕРІАЛІЗАЦІЇ ДЛЯ ВЛАСНОГО ТИПА ЗА ДОПОМОГОЮ БІБЛІОТЕКИ "BORSH" МОВИ ПРОГРАМУВАННЯ "RUST" 235

Напрям 4 АЛГОРИТМИ ТА МОДЕЛІ AI/ML 241

Бажан Тетяна Олександрівна, Поперешняк Світлана Володимирівна ВИКОРИСТАННЯ ГІБРИДНИХ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ІНВЕСТИЦІЙНИХ ПРОЦЕСІВ НА ОСНОВІ BIG DATA 241

Вельгус Христина Олексіївна АВТОМАТИЗАЦІЯ ПОБУДОВИ ІНДИВІДУАЛЬНИХ ПЛАНІВ НАВЧАННЯ НА ОСНОВІ АЛГОРИТМІВ ШТУЧНОГО ІНТЕЛЕКТУ 244

Бандура Артем Ігорович АЛГОРИТМИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ У ЦИФРОВИХ СЕРВІСАХ 247

Рудик Артем Всеволодович ОБҐРУНТУВАННЯ ВИБОРУ МОДЕЛЕЙ TTS ДЛЯ ОЗВУЧЕННЯ СПОВІЩЕНЬ НА ОС ANDROID 250

Завацький Владислав Олександрович ПРОБЛЕМА ГЕНЕРАЦІЇ БАГАТОВИМІРНИХ ЧАСОВИХ РЯДІВ ІЗ РОЗПОДІЛЕНИХ ДАТЧИКІВ В УМОВАХ IOT-СЕРЕДОВИЩ 253

Сич Марк Євгенійович ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПЕРСОНАЛІЗАЦІЇ КАР'ЄРНИХ РЕКОМЕНДАЦІЙ В ІТ-СФЕРІ: АЛГОРИТМИ ТА АРХІТЕКТУРНІ РІШЕННЯ 256

Волошин Данило Віталійович, Поперешняк Світлана Володимирівна АНАЛІЗ ВПЛИВУ НОВИННОГО ФОНУ НА КОРОТКОСТРОКОВІ ЦІНОВІ КОЛИВАННЯ 259

Ніщеменко Дмитро Олександрович ПЕРЕХІД ВІД ІМПЕРАТИВНОГО ДО ДЕКЛАРАТИВНОГО КЕРУВАННЯ РОЗУМНИМ БУДИНКОМ: КОНЦЕПЦІЯ INTENT-BASED CONTROL ТА ЇЇ ПРАКТИЧНА РЕАЛІЗАЦІЯ 262

Читулян Вадим Олегович ТЕХНОЛОГІЯ ІТЕРАТИВНОГО ПРОЄКТУВАННЯ КОРПОРАТИВНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ З ВБУДОВАНОЮ СИМУЛЯЦІЄЮ НАВАНТАЖЕННЯ 265

Криворучко Віталій Федорович, Садовенко Володимир Сергійович ФЕДЕРАТИВНЕ МАШИННЕ НАВЧАННЯ ДЛЯ ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ МУЛЬТИХМАРНИХ ПЛАТФОРМ 268

Гайдукевич Владислав Олегович МЕТОДИ ТА АЛГОРИТМИ НАВІГАЦІЇ МОБІЛЬНИХ РОБОТІВ У ДИНАМІЧНИХ СЕРЕДОВИЩАХ 271

Напрям 5 ІНФОРМАЦІЙНІ СИСТЕМИ ТА МЕРЕЖІ 274

Сокульський Олег Євгенович, Москаленко Наталія Володимирівна, Топольськов Євгеній Олександрович СУБД NewSQL ТА ІНФОРМАЦІЙНІ СИСТЕМИ 274

Москаленко Наталія Володимирівна, Сокульський Олег Євгенович DATA MINING ТА СУБД 276

Зуб Людмила Миколаївна АРХІТЕКТУРА ТА АЛГОРИТМ ФУНКЦІОНУВАННЯ РІВНЯ ІОТ-ПРИСТРОЇВ У СИСТЕМІ РОЗУМНОГО КАМПУСУ 278

Закаблук Едуард Євгенович, Вишнівський Віктор Вікторович АНАЛІЗ ДОЦІЛЬНОСТІ ВПРОВАДЖЕННЯ ПРИВАТНИХ ХМАРНИХ РІШЕНЬ ДЛЯ МАЛОГО ТА СЕРЕДНЬОГО БІЗНЕСУ 281

Чепур Софія Олексіївна РОЗРОБКА ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНТЕРНЕТ-МАГАЗИНУ ВИРОБІВ НАРОДНИХ ПРОМИСЛІВ ІЗ ВИКОРИСТАННЯМ СУЧАСНИХ ВЕБ-ТЕХНОЛОГІЙ 284

Черкесов Ілля Валерійович, Гаманюк Ігор Михайлович ПОБУДОВА МОДЕЛІ ПРЕЦЕДЕНТІВ ДЛЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ФЕРМЕРСЬКОГО ГОСПОДАРСТВА 287

Напрям 6 КОМП'ЮТЕРНІ МЕРЕЖІ ТА ТЕЛЕКОМУНІКАЦІЇ 290

Мацкевич Владислав Вікторович ІНТЕЛЕКТУАЛЬНІ АЛГОРИТМИ ОБРОБКИ СИГНАЛІВ У ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМАХ ЗВ'ЯЗКУ З РОЗНЕСЕНИМИ АНТЕНАМИ 290

Чугреєв Кирилл Олександрович ІНТЕЛЕКТУАЛЬНА ОПТИМІЗАЦІЯ МАРШРУТИЗАЦІЇ В ПРОГРАМНО-КЕРОВАНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ ІЗ ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ 293

Напрям 7 КІБЕРБЕЗПЕКА ТА ЗАХИСТ ІНФОРМАЦІЇ 296

Далькевич Денис Олександрович ПОРІВНЯЛЬНИЙ АНАЛІЗ ВЕБ-СЕРВІСІВ ОБМІНУ ПОВІДОМЛЕННЯМИ ЗА КРИТЕРІЯМИ АНОНІМНОСТІ ТА ЗАХИСТУ ДАНИХ 296

Далькевич Денис Олександрович ПРОЄКТУВАННЯ СИСТЕМИ ОБМІНУ ПОВІДОМЛЕННЯМИ З НАСКРІЗНИМ ШИФРУВАННЯМ НА БАЗІ ASP.NET CORE ТА ANGULAR..... 298

Кухаренко Артем Русланович БЕЗПЕКА, НАДІЙНІСТЬ ТА ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ IOT-СИСТЕМ БЕЗПЕРЕБІЙНОГО ЕЛЕКТРОЖИВЛЕННЯ 301

Ахмедов Амір Фарідович, Рашидов Сейфулла Фейзуллайович ПЕРЕДБАЧУВАНА ГІПОТЕЗА ЯК ЕПІСТЕМОЛОГІЧНИЙ ІНСТРУМЕНТ ПРОЄКТУВАННЯ БІОМЕТРИЧНИХ DID ДЛЯ ПОСТКВАНТОВОГО ЗАХИСТУ ІНФОРМАЦІЇ 304

Канарський Артем В’ячеславович ОРГАНІЗАЦІЯ БЕЗПЕКИ ДАНИХ У СИСТЕМІ РЕЗЮМУВАННЯ ТЕКСТІВ 307

Гайдамаченко Тимофій Максимович АТАКА НА ЛАНЦЮГ ПОСТАЧАННЯ ЧЕРЕЗ TYPOSQUATTING: ВИКОРИСТАННЯ НЕКЕРОВАНИХ LLM ДЛЯ ГЕНЕРАЦІЇ ШКІДЛИВИХ DOCKER-КОНТЕЙНЕРІВ 309

Напрям 8 BLOCKCHAIN ТА РОЗПОДІЛЕНІ ТЕХНОЛОГІЇ 311

Орлов Роман Русланович ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ЦИФРОВОЇ КРИМІНАЛІСТИКИ КІБЕРІНЦИДЕНТІВ ІЗ ВИКОРИСТАННЯМ РОЗПОДІЛЕНИХ ТЕХНОЛОГІЙ ТА МОДЕЛЕЙ ОЦІНКИ РИЗИКІВ..... 311

Сягровський Павло Дмитрович ІНТЕГРАЦІЯ BLOCKCHAIN-ТЕХНОЛОГІЙ У ВЕБ-ПЛАТФОРМІ AUTOASSIST+ ДЛЯ ВЕРИФІКАЦІЇ СЕРВІСНИХ ПЛАТЕЖІВ..... 314

Виливок Максим Володимирович ВИКОРИСТАННЯ БІБЛІОТЕКИ OPENZEPPELIN ДЛЯ РОЗРОБКИ ТОКЕНА СТАНДАРТУ ERC-20 317

Чорнобривець Дмитро Віталійович, Поперешняк Світлана Володимирівна ПАТЕРНИ ПРОЄКТУВАННЯ БАГАТОПОТОКОВОЇ АРХІТЕКТУРИ ДЛЯ ДЕСКТОПНИХ ДОДАТКІВ У СФЕРІ КОМП'ЮТЕРНОГО ЗОРУ 319

Гавор Артур Станіславович ГІБРИДНА АРХІТЕКТУРА EDGE-CHAIN ДЛЯ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ ЕКОЛОГІЧНОГО МОНІТОРИНГУ В УМОВАХ 6G-МЕРЕЖ 323

Гордієнко Катерина Олександрівна БЛОКЧЕЙН-ПІДХІД ДО ФОРМУВАННЯ ІСТОРІЇ СПОСТЕРЕЖЕНЬ ЗА ДЕФЕКТАМИ ДОРОЖНЬОГО ПОКРИТТЯ НА ОСНОВІ ВІДЕОДАНИХ ТРАНСПОРТНИХ ЗАСОБІВ..... 326

Напрям 9 ЦИФРОВА ЕКОНОМІКА ТА ЕЛЕКТРОННЕ ВРЯДУВАННЯ329

Бондаренко Дарина Вадимівна ВПРОВАДЖЕННЯ СИСТЕМИ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ В СУЧАСНИХ ОРГАНІЗАЦІЯХ 329

Фесенко Андрій Олегович, Поперешняк Світлана Володимирівна ІНТЕЛЕКТУАЛЬНИЙ ЦИФРОВИЙ АСИСТЕНТ ЯК ІНСТРУМЕНТ РОЗВИТКУ ЦИФРОВОЇ ЕКОНОМІКИ ТА ЕЛЕКТРОННОГО ВРЯДУВАННЯ 332

Напрям 10 ІННОВАЦІЇ У СФЕРІ ЦИФРОВОГО РОЗВИТКУ 335

Щербакова Катерина Євгенівна АНАЛІТИЧНІ ІНСТРУМЕНТИ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ МАЛОГО БІЗНЕСУ 335

Маковець Владислав Валерійович СИСТЕМА РАННЬОГО ВИЯВЛЕННЯ ТРИВОЖНИХ РОЗЛАДІВ НА ОСНОВІ ЕМОЦІЙНОЇ АНАЛІТИКИ ТА СТОРІТЕЛІНГУ 337

Напрям 1 ПРОГРАМНІ СИСТЕМИ ТА АРХІТЕКТУРИ

Лисенко Владислав Ігорович
студент 4 курсу
спеціальності «Інженерія програмного забезпечення»
Державного університету
інформаційно-комунікаційних технологій, м. Київ
st7947208@stud.duikt.edu.ua

Герцюк Микола Модестович
доктор філософії,
доцент кафедри
Технологій цифрового розвитку
Державного університету
інформаційно-комунікаційних технологій, м. Київ
m.gertsiuk@duikt.edu.ua

ДИЗАЙН ВЕБ-ЗАСТОСУНКУ З ВИКОРИСТАННЯМ ГЕКСАГОНАЛЬНОЇ АРХІТЕКТУРИ

Проєктування та побудова архітектури є базовими процесами розробки програмного забезпечення, під час яких приймаються одні з найважливіших рішень. Сьогодні обговорення здебільшого зосереджені на архітектурі рівня систем - розподілених системах, мікросервісах тощо. Водночас значно менше уваги приділяється архітектурі окремого застосунку, яка безпосередньо впливає на підтримуваність, тестованість та гнучкість коду.

Одним із підходів, що вирішує ці задачі, є гексагональна архітектура, відома також як "Порти та Адаптери" [1]. Вона розглядає систему як ізольоване ядро бізнес-логіки, що взаємодіє із зовнішнім світом виключно через чітко визначені інтерфейси. Такий підхід відокремлює домен від інфраструктурних деталей і забезпечує кращу адаптивність системи до змін.

Постановка задачі. Задачею дослідження є проектування архітектури серверного застосунку, що дозволяє кінцевому користувачу публікувати набори даних у відкритому доступі. Необхідно досягти явного відокремлення бізнес-логіки від зовнішніх компонентів - баз даних, API, Message Bus систем. Така структура дозволяє розширювати та змінювати інфраструктуру без впливу на основну логіку, що є критично важливим в умовах масштабування або зміни середовища.

Мета дослідження. Метою даного дослідження є імплементація гексагональної архітектури для серверної частини застосунку, у якому бізнес-логіка є повністю незалежною від інфраструктурних компонентів.

Результати дослідження. Проєктування розпочалося з виділення шарів та визначення меж відповідальності компонентів. На початковому етапі сформовано доменну модель, що відображає основні бізнес-сутності (користувач, набір даних, версії, метадані) та їх поведінку [2]. Доменний шар реалізовано як ізольоване ядро, незалежне від зовнішніх бібліотек і технологій.

Наступним кроком стало створення шару Application та визначення портів - інтерфейсів взаємодії системи із зовнішнім середовищем. Порти поділяються на два типи:

- вхідні порти (driving ports) - описують обробку запитів користувача (наприклад, API для публікації даних);
- вихідні порти (driven ports) - описують взаємодію з інфраструктурою (наприклад, збереження даних).

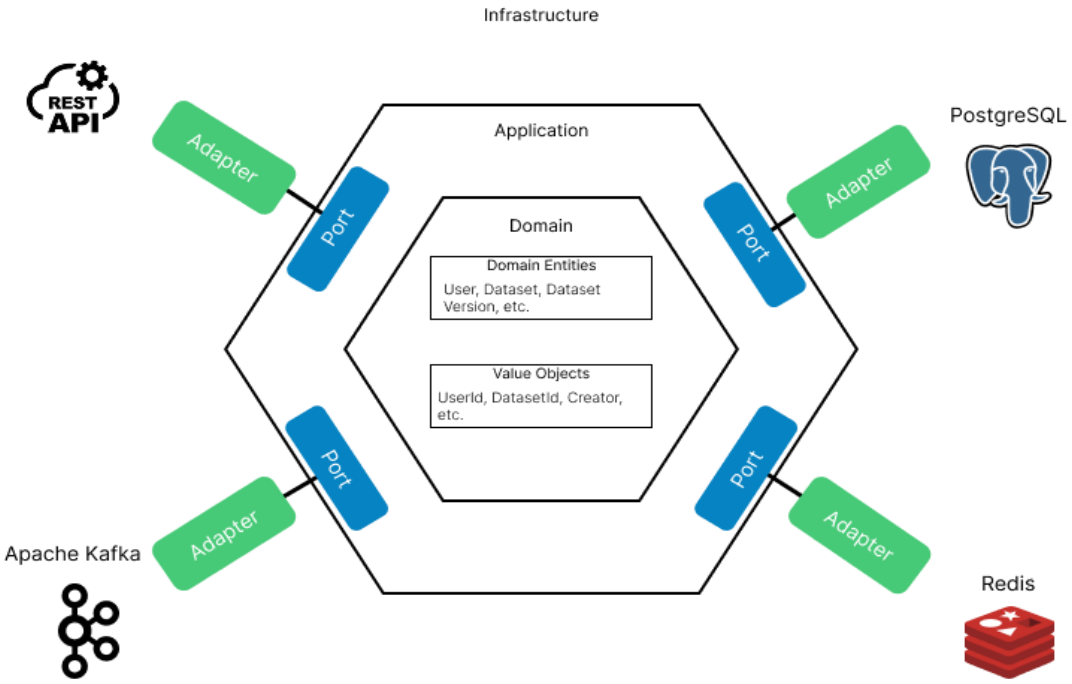


Рис. 1.Гексагональна архітектура проєкту

Далі реалізовано адаптери для підключення зовнішніх компонентів: REST API-адаптер для HTTP-запитів і адаптер бази даних через ORM. Архітектуру організовано так, щоб залежності спрямовувалися виключно до доменного ядра - через інверсію залежностей та використання інтерфейсів [3].

Однією з ключових переваг підходу є суттєве спрощення тестування. Завдяки ізоляції доменного шару більшість перевірок можна виконувати без підняття всієї системи. Бізнес-логіка тестується незалежно, а взаємодії з базою даних чи іншими сервісами підміняються через порти за допомогою мок-реалізацій. Інтеграційні тести використовуються лише для перевірки коректності взаємодії з інфраструктурою. Такий підхід зменшує час виконання тестів, підвищує їхню стабільність і спрощує виявлення помилок.

Отримані результати підтверджують, що гексагональний підхід формує чітку структуру застосунку з прозорими межами відповідальності між компонентами, а запропонована архітектура може слугувати основою для подальшого розвитку системи.

Недоліки та обмеження застосування. Незважаючи на переваги, гексагональна архітектура має низку обмежень. Насамперед вона ускладнює структуру: введення портів, адаптерів і додаткових шарів збільшує обсяг коду та кількість абстракцій, що для невеликих проєктів може бути надмірним. Крім того, підвищується поріг входу для розробників - ефективне використання підходу вимагає розуміння інверсії залежностей і принципів domain-driven design [4].

Надмірне дроблення на порти та адаптери може зробити код менш прозорим, особливо якщо домен є відносно простим. Переваги архітектури проявляються переважно в довгостроковій перспективі, тому для короткоживучих проєктів або систем із мінімальною бізнес-логікою витрати на її впровадження можуть не окупитися. Доцільність застосування напряму залежить від складності домену, очікуваного життєвого циклу та вимог до масштабованості.

Висновки та перспективи. Гексагональна архітектура є ефективним підходом до проєктування застосунків, що забезпечує ізоляцію бізнес-логіки, чітке розділення відповідальностей та покращену тестованість [1]. Її використання дозволяє будувати гнучкі рішення, де інфраструктурні деталі не впливають на доменну модель, що особливо важливо у довготривалих і складних проєктах.

Перспективи подальшого розвитку пов'язані з адаптацією підходу до розподілених систем, підвищенням ефективності організації залежностей і зменшенням надлишкових абстракцій. Актуальним залишається пошук практик, що дозволяють зберігати баланс між чистотою архітектури та простотою реалізації.

Список використаних джерел

1. Cockburn A. The Hexagonal (Ports & Adapters) Architecture. Alistair Cockburn. URL: https://alistair.cockburn.us/hexagonal-architecture
2. C. Martin R. The Clean Architecture. Clean Coder Blog. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
3. L. Schuchert B. DIP in the Wild. martinfowler.com. URL: https://martinfowler.com/articles/dipInTheWild.html
4. Evans E. Domain-Driven Design Tackling Complexity in the Heart of Software. 2003. URL: https://dl.acm.org/doi/10.5555/861502

Ніколін Кирило Андрійович
студент 4 курсу, групи ТЦР-42
Державний університет інформаційно комунікаційних технологій
(099)569-82-26
kirillnikolin@gmail.com

РОЗРОБКА ВЕБ- ТА МОБІЛЬНОЇ ПЛАТФОРМИ ЛАБОРАТОРНИХ АНАЛІЗІВ ДЛЯ МЕДИЧНОЇ ЛАБОРАТОРІЇ «MONOLAB»

Великі медичні мережі давно мають електронні реєстратури і працюють з HELSI. Але це не про всіх. Невелика приватна лабораторія – це часто зовсім інша реальність: запис по телефону, результати у Viber, жодного централізованого зберігання даних. Пацієнт просто чекає дзвінка - і не завжди його отримує. Саме з такої ситуації і виріс проект «Monolab».

Постановка задачі. В Україні цифровізація охорони здоров'я здебільшого торкнулася великих держструктур – тієї ж HELSI. Натомість невеликі приватні лабораторії у цьому питанні суттєво відстають: запис по телефону, результати у Viber або взагалі паперовий бланк. Зокрема, лабораторія «Monolab» зіткнулась із ситуацією, коли персонал витрачав значний час на ручну комунікацію з пацієнтами, а самі пацієнти могли просто не отримати свої результати вчасно – через людський фактор або банально забутий дзвінок. Дані про аналізи зберігались фрагментовано, і жодної гарантії конфіденційності це не давало.

Саме тому було поставлено завдання розробити єдину платформу – монорепозиторій, де API, веб-кабінет і мобільний застосунок живуть разом і розуміють одне одного. Основу складність становило не написання коду як такого, а забезпечення синхронізації даних у реальному часі: коли лаборант вводить результат у веб-панель, пацієнт на мобільному пристрої має побачити його за кілька секунд – без конфліктів версій. Вирішення цього завдання виявилось нетривіальним.

Мета дослідження. Основна ідея - щоб пацієнт міг записатись, здати аналізи і отримати результати на телефон, не телефонуючи в лабораторію жодного разу. Плюс нормальний інструмент для лаборантів і адміністраторів, бо їм теж незручно працювати в Excel і мессенджерах.

Результати дослідження. Розробку було розпочато з вибору архітектури. Монорепозиторій на TypeScript виявився тут дуже вдалим рішенням - коли бекенд, веб і мобільний застосунок знаходяться в одному репозиторії і спільно використовують типи, купа помилок відловлюється ще на етапі компіляції. Наприклад, змінив структуру відповіді API – і одразу видно у всіх трьох застосунках де щось поламалось. Для медичних даних це не дрібниця: неправильно відображений референтний діапазон аналізу пацієнт може неправильно інтерпретувати свій результат.

Бекенд написаний на Node.js + Express [2]. Як базу даних було обрано PostgreSQL - медичні записи мають чітку реляційну природу: пацієнт - запис - зразок - результат - конкретний тест. Тут транзакції та зовнішні ключі критичні, тому MongoDB навіть не розглядалась як альтернатива [1]. Prisma ORM забезпечила типізовані запити і зручну систему міграцій [4] – при додаванні нових полів до схеми аналізів, це проходило без ручної правки SQL. Безпеку побудовано на JWT з двома токенами (access + refresh) та RBAC-middleware: лаборант фізично

не може потрапити в адмінські ендпоїнти, а пацієнт – переглянути чужі результати. Додатково всі критичні дії пишуться в audit log разом із IP та user agent.

Веб-панель для персоналу побудована на React із RTK Query. Проблема, яку було вирішено – адміністратор бачив актуальну чергу лише після ручного оновлення сторінки. Було підключено WebSocket, і тепер статуси записів оновлюються в реальному часі без жодних перезавантажень. Мобільний застосунок на React Native [3] вирішував іншу задачу: пацієнт хоче відкрити результати, а не шукати лист на пошті. PDF-бланки генеруються на сервері через PDFKit - це забезпечило повний контроль над верстkою документа. Зберігаються файли у MinIO (S3-сумісне сховище), завдяки чому основна БД не захаращується бінарними даними, і результати відкриваються навіть при поганому мобільному інтернеті.

Push-сповіщення через Firebase Cloud Messaging [5] виявились одним із найбільш оцінених елементів при тестуванні: пацієнт отримує сигнал на телефон щойно лаборант верифікував результат – не треба самостійно перевіряти застосунок. Під час тестування – від моменту як лаборант верифікував результат до того як пацієнт його отримував, пройшло десь на 30% менше часу ніж раніше з телефоном і email. Загалом вийшла повноцінна екосистема з трьох застосунків, де кожен компонент має свою зону відповідальності, але всі вони працюють із єдиним джерелом даних (див. рис. 1).

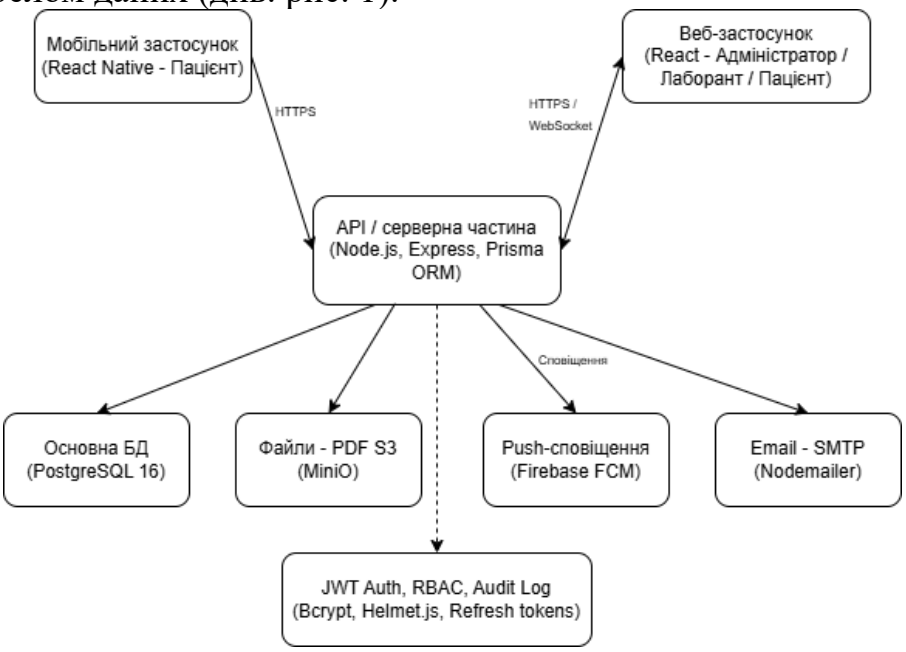


Рис. 1. Архітектура платформи лабораторних аналізів «Monolab».

Висновки та перспективи. Початково масштаб розробки видався надмірним для дипломного проекту. Але монорепозиторій із TypeScript скрізь себе виправдав: за наявності спільних типів між трьома застосунками, дуже багато дурних помилок просто не доходить до рантайму. По суті, пацієнт тепер взагалі не дзвонить у лабораторію – ні щоб записатись, ні щоб дізнатись результати. І для персоналу паперів теж стало значно менше.

З подальших напрямів – інтеграція з реєстром eHealth виглядає логічним кроком для реального впровадження. Також у планах: модуль аналітики в мобільному застосунку з графіками динаміки показників у часі та біометрична авторизація для додаткового захисту медичних даних.

Список використаних джерел

1. PostgreSQL Global Development Group. PostgreSQL 16 Documentation [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/16/
2. OpenJS Foundation. Node.js v20 Documentation [Електронний ресурс] – Режим доступу: https://nodejs.org/docs/latest-v20.x/api/
3. Meta Platforms. React Native Documentation [Електронний ресурс] – Режим доступу: https://reactnative.dev/docs/getting-started
4. Prisma Data. Prisma ORM Documentation [Електронний ресурс] – Режим доступу: https://www.prisma.io/docs
5. Google LLC. Firebase Cloud Messaging Documentation [Електронний ресурс] – Режим доступу: https://firebase.google.com/docs/cloud-messaging

Крєпкій Ігор Тарасович
студент 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
+380976754799
tkrepkiy@gmail.com

КЛІЄНТ-СЕРВЕРНА СИСТЕМА ПОТОКОВОЇ ПЕРЕДАЧІ АУДІО В РЕАЛЬНОМУ ЧАСІ НА БАЗІ ANDROID SDK ТА UDP

Постановка задачі. Використання мобільних пристроїв як зовнішніх мікрофонів для персональних комп'ютерів є економічно доцільним рішенням для забезпечення віддаленої роботи та стрімінгу. Більшість існуючих аналогів мають закритий вихідний код, вимагають встановлення пропрієтарних драйверів на стороні ядра ОС та створюють відчутну затримку через використання HTTP-базованих медіапротоколів. Існує потреба у розробці відкритої системи, здатної мінімізувати затримку трансляції звуку в локальній мережі без втручання в системні драйвери комп'ютера.

Мета дослідження. Створення відкритої клієнт-серверної архітектури потокової передачі аудіоданих з мобільного пристрою на ПК з використанням транспортного протоколу UDP для забезпечення мінімальної затримки без необхідності написання спеціалізованих системних драйверів [3].

Результати. Розроблена система базується на моделі "клієнт-сервер". Передача даних виконується у локальній мережі Wi-Fi. Відмова від протоколів гарантованої доставки на користь транспортного протоколу UDP дозволила усунути затримки, пов'язані з буферизацією, підтвердженням доставки пакетів та процедурами повторної передачі. Для голосового зв'язку в реальному часі втрата окремого пакета є критично менш відчутною, ніж накопичення системної затримки.

Клієнтський застосунок для ОС Android використовує низькорівневий клас AudioRecord [1] замість стандартних мультимедійних бібліотек. Захоплення аудіосигналу відбувається в моноканальному режимі з частотою дискретизації 44.1 кГц та розрядністю 16 біт (нестиснений формат PCM). Такий підхід виключає процесорні витрати на кодування звуку в алгоритми з втратами. Зчитаний масив байтів безпосередньо обгортається у датаграму та відправляється на задану IP-адресу.

Серверна частина, написана мовою Java, функціонує як маршрутизатор аудіопотоку. Сервер отримує UDP-пакети та через класи Java Sound API [2] передає їх у віртуальний аудіокабель (наприклад, VB-Audio Virtual Cable) [4], попередньо обраний користувачем. Операційна система ідентифікує вихід цього кабелю як апаратний мікрофон. Програмна маршрутизація в просторі користувача позбавляє від ризиків дестабілізації ядра системи.

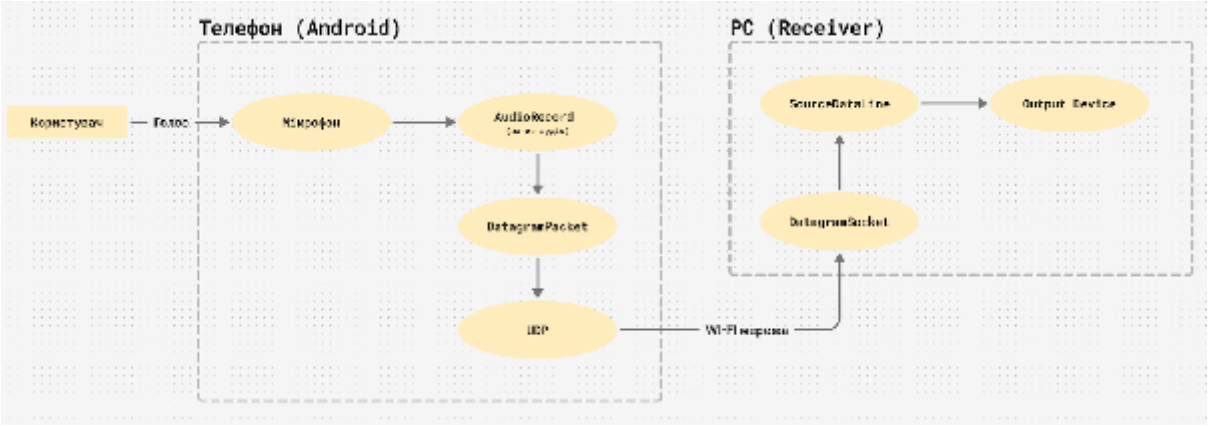


Рис. 1. Модель потоків даних архітектури системи

Висновки. Створено діючий прототип системи бездротового мікрофона, який демонструє високу ефективність протоколу UDP та формату PCM для передачі аудіо в реальному часі. Відмова від розробки пропрієтарних драйверів ядра та перенесення логіки маршрутизації в простір користувача спростили архітектуру проєкту. Комплекс забезпечує стабільну трансляцію звуку у локальній мережі з мінімальною системною затримкою, рівень якої є цілком прийнятним для комфортного голосового спілкування та стрімінгу. Подальші дослідження будуть спрямовані на інтеграцію алгоритмів шифрування трафіку та забезпечення сумісності сервера з аудіосистемами Linux.

Список використаних джерел

1. AudioRecord. Android Developers. URL: https://developer.android.com/reference/android/media/AudioRecord (дата звернення: 13.04.2026).
2. Java Sound API Programmer's Guide. Oracle. URL: https://docs.oracle.com/javase/8/docs/technotes/guides/sound/ (дата звернення: 13.04.2026).
3. Postel J. User Datagram Protocol. IETF Datatracker. 1980. URL: https://datatracker.ietf.org/doc/html/rfc768 (дата звернення: 13.04.2026).
4. VB-CABLE Virtual Audio Device. VB-Audio Software. URL: https://vb-audio.com/Cable/ (дата звернення: 13.04.2026).

Тимофіїв Сергій Іванович
Студент, група ТСД-42
Державний університет інформаційно-комунікаційних технологій
0504656361
bidonpc1@gmail.com

Вагін Роман Русланович
Студент, група КНб-1-25-4.0д
Київський столичний університет імені Бориса Грінченка
0685989262
roman.education000@gmail.com

МЕТОД ОПТИМІЗАЦІЇ ПАРАМЕРТІВ МОБІЛЬНИХ ГРАФІЧНИХ СИСТЕМ

Постановка задачі. Експлуатація сучасних мобільних графічних систем у заводському стані часто супроводжується подачею надлишкової напруги на ядро графічного процесора (ГП). Виробники встановлюють універсальні параметри напруги живлення для забезпечення стабільної роботи будь-якого екземпляра кремнію, незалежно від його індивідуальної якості. Це призводить до підвищеного енергоспоживання та передчасного досягнення термічних лімітів (троттлінгу), що особливо критично для потужних систем з обмеженим внутрішнім простором.

Мета дослідження. Актуальність даного дослідження полягає в необхідності пошуку методів підвищення енергоефективності без втрати чистої продуктивності системи. Головною метою є дослідження процесу андерволтингу мобільного графічного процесора RTX 4060 та аналіз методики корекції кривої напруги для досягнення максимальної стабільності чипа.

Результати. Дослідження проводилося на базі ноутбука ASUS TUF Gaming A15, оснащеного ГП NVIDIA GeForce RTX 4060 (архітектура Ada Lovelace) з повним теплопакетом (TGP 140W). Для маніпуляцій з вольт-частотною характеристикою використовувався редактор кривих у програмному комплексі MSI Afterburner. Методика полягала у покроковому зниженні напруги ядра при збереженні цільової частоти 2500 МГц.

Шляхом експериментальної корекції було встановлено, що для частоти 2500 МГц оптимальним значенням напруги є 0.9 В (див. Рисунок 1). Така фіксація запобігає надмірному нагріву кристала та виключає термічний троттлінг (див. Рисунок 2). Цей підхід дозволяє стабілізувати тактові частоти на вищих показниках завдяки вивільненню запасу потужності.

Під час тестування стабільності та вимірювання продуктивності у бенчмарку FurMark (роздільна здатність 1920x1080) було зафіксовано 10 053 бали (див. Рисунок 3). Цей результат перевищує показники заводських налаштувань. Подальше зниження напруги (нижче 0.85 В) у даному тестовому сценарії призводило до нестабільності, що підтверджує індивідуальну межу вольт-частотної характеристики конкретного зразка ГП при пікових навантаженнях.

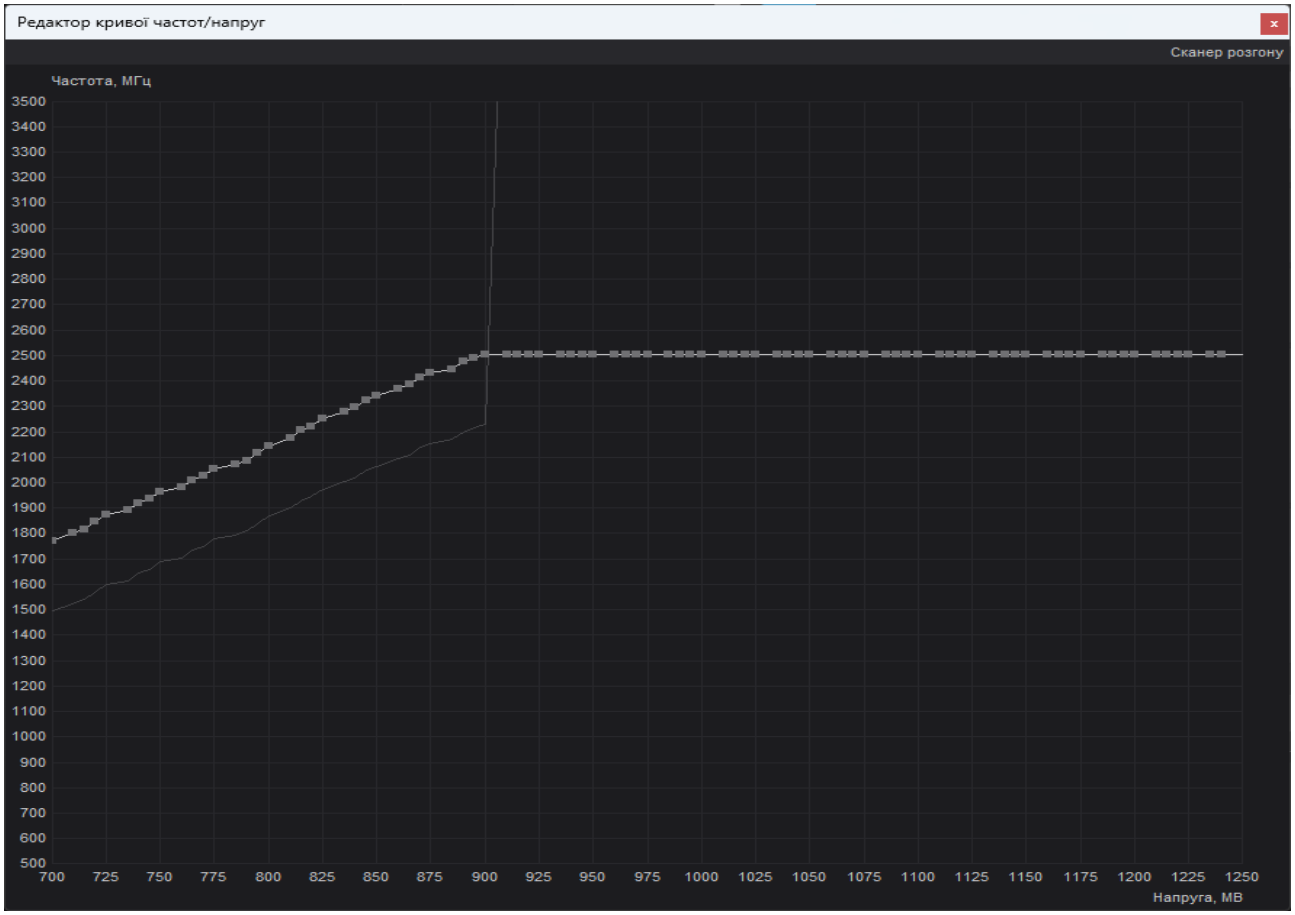


Рис 1. Редактор кривої частот/напруг

GPU0	NVIDIA GeForce RTX 4060 Laptop GPU
PCI IDs	DEV: 10DE-28E0 - SUB: 1043-19DD - REV: A1
graphics driver	591.59
max core usage	99 %
max core temperature	86 °C
core clock	min: 2250 MHz - max: 2506 MHz
max power	140 W

Рис 2. Показники стану графічного процесора під час тестування

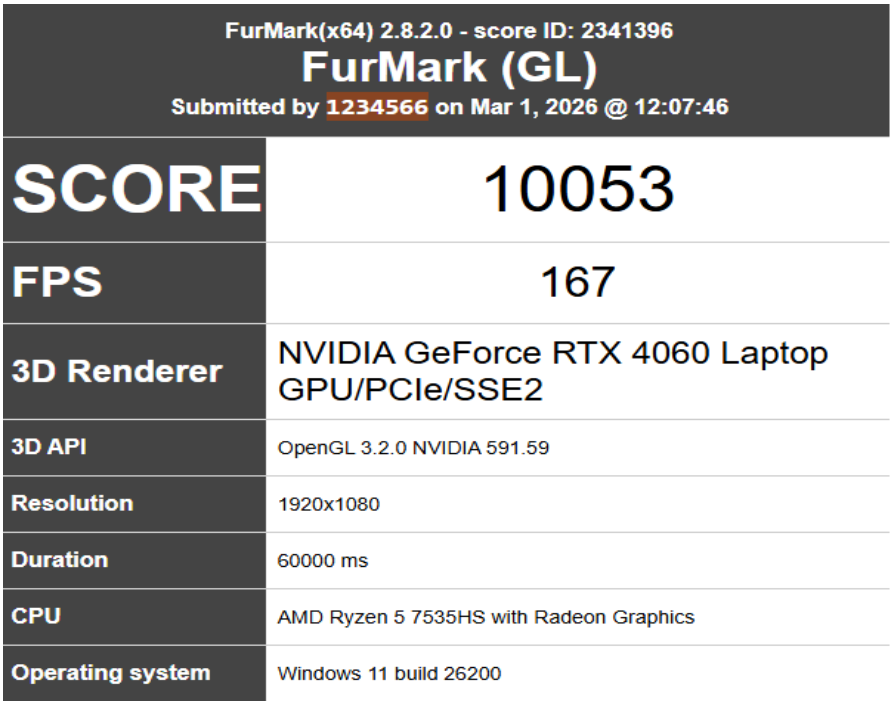


Рис 3. Результати тестування

Висновки. Андервольтинг є найбільш ефективним методом програмної оптимізації сучасних комп'ютерних систем. Встановлення фіксованої напруги на 0.9 В забезпечує максимальну продуктивність та термічну стабільність, що робить мобільне рішення надзвичайно ефективним. Запропонована методика є універсальною для оптимізації та має пряме практичне значення для сервісного обслуговування високопродуктивної техніки, дозволяючи значно подовжити термін її експлуатації.

Список джерел:

1. MSI Afterburner User Guide. URL: https://www.msi.com/page/afterburner
2. NVIDIA Ada Lovelace Architecture Whitepaper. NVIDIA Corporation, 2022. 45 p.
3. Geeks3D FurMark Benchmarking Tool. URL: https://geeks3d.com/furmark/
4. Бондарчук, А. П., Мельник, І. Ю., & Абрамов, В. О. (2025). Підвищення ефективності систем відеоспостереження за рахунок гібридного методу відбору ключових кадрів і інтерпретації рішень. Телекомунікаційні та інформаційні технології, (3), 101-105.
5. Стрельніков, В. І., & Бондарчук, А. П. (2025). Комплексний підхід до інтелектуального управління, моделювання та виявлення мережних аномалій на основі ентропійних та нейромережевих підходів. Телекомунікаційні та інформаційні технології, (2), 100-107.

Бойченко Нікіта Олегович
Студент 4 курсу, групи ТЦР-41
Державний університет інформаційно-комунікаційних технологій
(096)732-94-32
boynykyta@gmail.com

СПРОЩЕННЯ ВЗАЄМОДІЇ МОНІТОРИНГУ ІНВЕРТОРА ЧЕРЕЗ ТЕЛЕГРАМ-БОТ З АВТОМАТИЧНИМИ СПОВІЩЕННЯМИ ТА ЗРОЗУМІЛИМ ПОДАННЯМ ДАНИХ

Сайт mini-ems [1] та подібні інструменти моніторингу інверторів орієнтовані насамперед на технічних спеціалістів. Користувач бачить напругу, рівень заряду, навантаження, але з цих цифр не завжди одразу зрозуміло, що відбувається вдома прямо зараз. Щоб дізнатися просту річ, чи є в квартирі світло, потрібно відкрити браузер, перейти на сайт, авторизуватися і вже потім самому розбирати показники. У повсякденному житті це незручно, особливо коли потрібна швидка відповідь. До того ж жодних нормальних сповіщень про зникнення або появу електроенергії ці сервіси не надають. Саме тому виникла ідея зробити телеграм-бот, який забирає сирі дані з веб-сервісу і видає щось зрозуміле, коротко і одразу [2].

Постановка задачі. Потрібно розробити телеграм-бот, який сам підключається до сервісу моніторингу інвертора, отримує поточні дані, перетворює їх на короткий зрозумілий статус і надсилає сповіщення, коли світло зникає або з'являється. Завдання полягає у тому, щоб прибрати необхідність щоразу вручну заходити на сайт або відкривати окремий застосунок.

Мета дослідження. Створити зручний спосіб отримання даних інвертора через Telegram. Конкретно: перетворити технічні поля веб-сервісу на прості людські повідомлення, налаштувати автоматичні сповіщення про зміни в електропостачанні та дати користувачеві змогу розуміти стан системи без жодного технічного аналізу.

Результати дослідження. Бот реалізовано на Python з використанням бібліотеки aiogram [2] і локальної бази SQLite [3]. Він регулярно звертається до сервісу моніторингу і формує коротку відповідь, не список технічних значень, а зрозумілий статус: СВІТЛО Є або СВІТЛА НЕМАЄ, плюс рівень заряду батареї та поточне навантаження.

Окремо вдалося спростити відображення сирих даних інвертора. У коді закріплено розшифровку основних полів, тому поряд із коротким статусом бот може показати й повний звіт, але вже не суцільним набором кодів і чисел, а згрупованим за змістом: поточний стан, батарея, енергетичні потоки, добові показники. Розбиратися в технічних деталях самому вже не потрібно.

Майже головною частиною системи являються автоматичні сповіщення. Щойно зникає мережа, бот надсилає повідомлення. Щойно електрика повернулася, надсилає ще одне. Це повністю прибирає потребу перевіряти стан вручну, і саме ця функція найбільше відповідає реальним побутовим потребам.

Як додаткова корисна деталь, бот підраховує короткий прогноз автономності і додає його до статусу, якщо вистачає даних. Функція не є ключовою, але дає людині краще відчуття ситуації: не просто «батарея 60%», а приблизно скільки ще протримається система без мережі.

Нарешті, бот накопичує історію показників: заряд, навантаження, напруга. Це закладає основу для майбутніх розширень, наприклад сповіщення про критично низький заряд або нетипові зміни параметрів. Тобто система працює не лише як канал сповіщень, а й як зручний щоденний інтерфейс для контролю інвертора.

Висновки. Вирішальне значення має те, як подаються дані. Телеграм-бот усуває потребу постійно заходити на сайт моніторингу або окремий додаток, дає змогу швидко зрозуміти поточний стан електроживлення і вчасно отримувати повідомлення про важливі зміни. Взаємодія з системою стає швидшою, зрозумілішою і значно ближчою до того, що реально потрібно звичайному користувачеві.

Список використаних джерел:

1. Mini-EMS: веб-сервіс моніторингу інвертора. Режим доступу: https://mini-ems.com:8081/dn/login (дата звернення: 14.04.2026).
2. Документація aiogram 3. Режим доступу: https://docs.aiogram.dev/ (дата звернення: 14.04.2026).
3. Документація SQLite. Режим доступу: https://www.sqlite.org/docs.html (дата звернення: 14.04.2026).

Цайгер Юрій Петрович
студент 4 курсу, групи ТЦР-43
Державний університет інформаційно-комунікаційних технологій
(095) 719-37-59
uastalavista@gmail.com

ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ ОСОБИСТИХ ФІНАНСІВ ЗАСОБАМИ ASP.NET CORE MVC

Постановка задачі. Особисті фінанси — одна з тих сфер, де автоматизація реально змінює поведінку людини: коли витрати видно наочно, їх легше контролювати. Проте готові сервіси — Mint/Credit Karma, YNAB — закриті для українського ринку або орієнтовані на іноземні ринки [1, 2]. Завдання роботи полягало у створенні локального застосунку, який читає виписки банків і дає зручний інструмент аналізу власних витрат.

Мета дослідження. Реалізувати веб-застосунок обліку фінансів, що охоплює повний цикл роботи з транзакціями, від завантаження банківської виписки до категоризованого звіту з діаграмами.

Результати. Застосунок реалізовано на платформі ASP.NET Core MVC. Дані зберігаються у файловій базі SQLite через Entity Framework Core 9.0, що дозволяє запускати систему без будь-яких попередніх налаштувань сервера. Склад реалізованих модулів наведено на рис. 1.

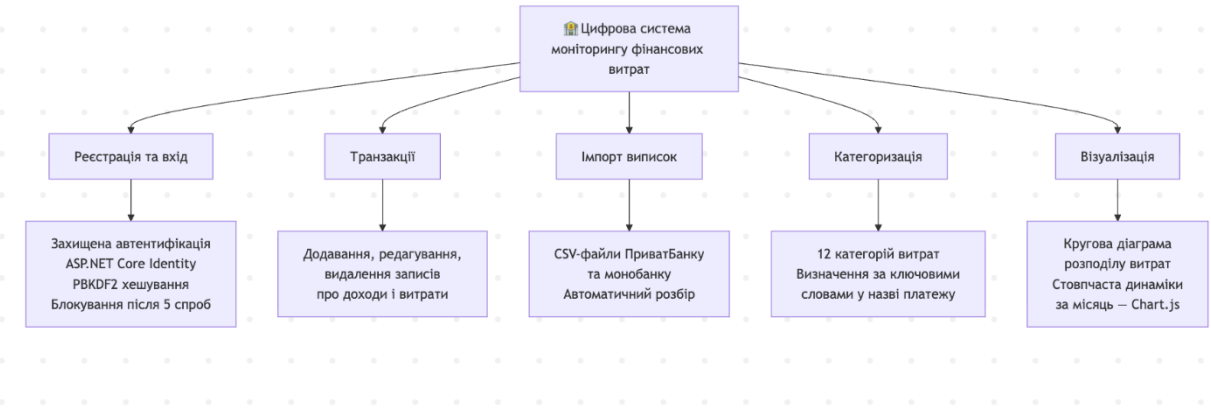


Рис. 1. Структура модулів застосунку.

Ключовою особливістю є модуль імпорту виписок: користувач завантажує CSV-файл зі свого банківського кабінету, і система автоматично розбирає транзакції, визначає категорію кожного платежу та зберігає їх пакетно. Підтримуються формати ПриватБанку та монобанку. Категоризація відбувається через пошук ключових слів у назві платежу — наприклад, «АТБ» або «Сільпо» потрапляють до категорії «Продукти», а «Укрзалізниця» чи «Bolt» - до «Транспорту».

Для відображення даних підключено Chart.js — бібліотека рендерить кругову діаграму розподілу витрат за категоріями та стовпчасту з динамікою за обраний місяць. Усе це відображається у браузері через Razor-шаблони без окремого фронтенд-фреймворку. Безпеку входу забезпечує ASP.NET Core Identity: паролі хешуються алгоритмом PBKDF2, а обліковий запис автоматично блокується після п'яти невдалих спроб[3, 4].

Висновки. Створено повнофункціональний застосунок для обліку особистих фінансів, що працює локально і не вимагає налаштування зовнішніх сервісів. Реалізовано імпорт виписок українських банків, автоматичну категоризацію транзакцій та інтерактивну візуалізацію витрат. Застосунок вирішує практичну задачу, яку наявні іноземні сервіси не покривають для українського користувача. Подальший розвиток передбачає підключення до банківських API для отримання транзакцій у реальному часі без ручного завантаження файлів.

Список використаних джерел

1. Intuit is shutting down Mint, its popular free budget-tracking app. *Engadget*. URL: https://www.engadget.com/intuit-is-closing-down-mint-its-popular-free-budget-tracking-app-054145229.html (дата звернення: 18.04.2026).
2. YNAB. *YNAB*. URL: https://www.ynab.com/ (дата звернення: 18.04.2026).
3. Overview of ASP.NET Core MVC. *Microsoft Learn: Build with answers in reach*. URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-10.0 (дата звернення: 18.04.2026).
4. Chart.js | Chart.js. *Chart.js | Open source HTML5 Charts for your website*. URL: https://www.chartjs.org/docs/latest/ (дата звернення: 18.04.2026).

Артеменко Анастасія Ярославівна
студентка 4 курсу, групи ТЦР-41
Державний університет інформаційно-комунікаційних технологій
(073)104-09-44
st8023175@stud.duikt.edu.ua

РЕАЛІЗАЦІЯ МЕХАНІЗМУ КОНТРОЛЮ ТА АНАЛІЗУ ПРИЙОМУ ЛІКІВ У ВЕБ-ЗАСТОСУНКУ НА БАЗІ ASP.NET CORE

Постановка задачі. Проблема дотримання режиму медикаментозного лікування залишається актуальною як у клінічній практиці, так і в контексті розробки цифрових інструментів підтримки пацієнтів [3]. Більшість наявних мобільних і веб-орієнтованих рішень зводяться до функції нагадувань і не передбачають систематичного відстеження реальної поведінки користувача. Як наслідок, лікар або сам пацієнт не має об'єктивного уявлення про те, наскільки точно дотримується призначений курс лікування, а виявлення систематичних пропусків доз відбувається із запізненням або не відбувається взагалі. Це знижує ефективність терапії та ускладнює прийняття обґрунтованих клінічних рішень [2]. Отже, існує практична потреба у розробці механізму, який поєднує контроль виконання графіка прийому ліків з аналітичною компонентою, здатною виявляти відхилення та формувати відповідні сповіщення.

Мета дослідження. Метою роботи є проектування та програмна реалізація механізму контролю і аналізу прийому лікарських засобів у веб-застосунку, розробленому на платформі ASP.NET Core, з можливістю виявлення порушень режиму лікування та надання користувачу своєчасних рекомендацій.

Результати дослідження. У ході роботи розроблено веб-застосунок, центральним елементом якого є механізм контролю дотримання графіка прийому медикаментів. Застосунок побудовано за класичною архітектурою «клієнт–сервер», що забезпечує чітке розмежування між рівнями представлення та бізнес-логіки.

Серверну частину реалізовано засобами ASP.NET Core — сучасного кросплатформного фреймворку, що вирізняється високою продуктивністю та гнучкістю масштабування [1]. Для організації доступу до реляційної бази даних застосовано Entity Framework Core з підходом Code First, що дозволило прискорити розробку та спростити підтримку схеми даних [4]. Предметна модель охоплює такі сутності: користувач, лікарський засіб, графік прийому та журнал виконання.

Ключовим елементом є механізм контролю прийому, який базується на порівнянні запланованого графіка із фактичними діями користувача. Алгоритм роботи включає фіксацію часу прийому, реєстрацію факту виконання, визначення відхилень та формування попереджень. Основні функціональні компоненти системи наведено у таблиці 1.

Табл. 1

Компоненти механізму контролю прийому ліків

Компонент	Функції	Результат
Планування	Формування графіка прийому	Структурований розклад
Моніторинг	Фіксація дій користувача	Дані про виконання
Аналіз	Порівняння плану і факту	Виявлення відхилень
Оцінка	Визначення типу порушення	Класифікація ситуацій
Сповіщення	Генерація повідомлень	Реакція користувача

Інтелектуальний модуль здійснює аналіз поведінки користувача та дозволяє виявляти порушення режиму прийому, що відповідає сучасним підходам до аналізу даних у програмних системах.

Додатково реалізовано базові механізми безпеки, включаючи автентифікацію користувачів та контроль доступу.

Висновки та перспективи. У роботі реалізовано механізм контролю та аналізу прийому лікарських засобів, що виходить за межі простого нагадування і забезпечує змістовне відстеження дотримання режиму терапії. Використання ASP.NET Core [1] та Entity Framework Core [4] дозволило побудувати продуктивну й розширювану систему з чіткою архітектурою. Отримані результати можуть стати основою для розробки більш комплексних систем підтримки прийняття клінічних рішень.

Перспективним напрямом подальших досліджень є інтеграція методів машинного навчання для прогнозування ймовірності пропуску прийому на основі індивідуальної поведінкової моделі користувача, а також розширення системи до підтримки взаємодії з медичним персоналом у режимі реального часу.

Список використаних джерел

1. ASP.NET Core fundamentals overview. Microsoft Learn: Build with answers in reach. https://learn.microsoft.com/en-us/aspnet/core/fundamentals/?view=aspnetcore-10.0&tabs=windows.

2. How to govern the digital transformation of health services / W. Ricciardi et al. European Journal of Public Health. 2019. Vol. 29, no. 3. P. 7–12. https://doi.org/10.1093/eurpub/ckz165.

3. Public health digitalization in Europe: EUPHA vision, action and role in digital public health / A. Odone et al. *European Journal of Public Health*. 2019. Vol. 29. P. 28–35. https://doi.org/10.1093/eurpub/ckaa083.

4. Tutorial Part 5, apply migrations to the Contoso University sample. *Microsoft Learn: Build with answers in reach*. https://learn.microsoft.com/en-us/aspnet/core/data/ef-mvc/migrations?view=aspnetcore-10.0 .

Ходак Денис Павлович
студент 4 курсу, групи ТЦР-43
Державний університет інформаційно-комунікаційних технологій
(093) 461-03-30
dengood.on@gmail.com

ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ЛЮДСЬКИМИ РЕСУРСАМИ НА БАЗІ ASP.NET CORE MVC

Ефективне управління персоналом є одним із ключових чинників конкурентоспроможності сучасного підприємства. Зростання кількості співробітників, ускладнення кадрових процесів та підвищені вимоги до захисту персональних даних зумовлюють необхідність упровадження спеціалізованих інформаційних систем. Розробка власної HRMS дозволяє врахувати специфіку організації, забезпечити інтеграцію з внутрішніми процесами та уникнути залежності від дорогих комерційних рішень [1]. Платформа ASP.NET Core MVC надає потужні засоби для реалізації захищених, масштабованих і зручних у супроводі корпоративних застосунків.

Постановка задачі. Спроектувати та реалізувати інформаційну систему управління людськими ресурсами на базі ASP.NET Core MVC з використанням ASP.NET Core Identity для автентифікації та авторизації, Entity Framework Core для роботи з базою даних, а також визначити архітектурні рішення, що забезпечать розмежування доступу за ролями та зручність адміністрування кадрових даних.

Мета дослідження. Розробити функціональну HRMS, яка автоматизує облік співробітників, управління відділами, посадами та заявами на відпустку, реалізує рольову модель доступу (Адміністратор, HR-менеджер, Керівник, Співробітник) і забезпечує захищений профіль користувача на основі розширеного IdentityUser.

Результати дослідження.У ході роботи спроектовано багатошарову архітектуру HRMS відповідно до патерну MVC. Рівень моделей містить сутності: *ApplicationUser* (розширений клас *IdentityUser* з полями ПІБ, по батькові, дати народження та адреси), *Employee*, *Department*, *Position* та *LeaveRequest*. Зв'язки між сутностями визначено через Fluent API в класі *ApplicationDbContext*, що гарантує цілісність реляційної схеми [2].

Рівень контролерів реалізує шість контролерів: *AccountController* (реєстрація, вхід, зміна паролю, редагування профілю), *DashboardController* (зведена статистика), *EmployeesController*, *DepartmentsController*, *PositionsController* та *LeaveRequestsController*. Кожен контролер застосовує атрибут *[Authorize]* з обмеженням за роллю для захисту відповідних дій CRUD [3]. ViewModel-класи забезпечують строго типізований зв'язок між контролером та виглядом (View), включаючи анотації DataAnnotations для серверної валідації.

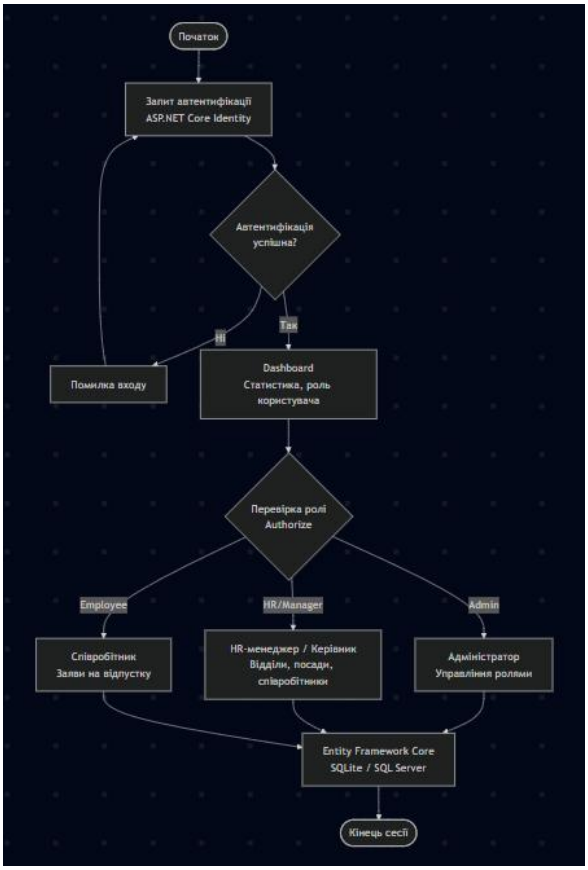


Рис. 1. Архітектура системи навігації.

Систему автентифікації реалізовано на базі ASP.NET Core Identity з налаштованою політикою паролів: мінімум 6 символів, обов'язкові великі та малі літери й цифри. Передбачено механізм блокування облікового запису після 5 невдалих спроб входу. Тимчасові паролі для нових співробітників генеруються автоматично та передаються HR-менеджеру. Сесії налаштовано зі ковзаючим часом дії 8 годин [4].

Для зберігання даних використовується SQLite у режимі розробки з підтримкою безшовної міграції на SQL Server. Міграції EF Core забезпечують автоматичне створення схеми бази даних при першому запуску застосунку. Початкове наповнення (Seed Data) включає чотири відділи, п'ять посад та два тестові облікові записи: адміністратор і HR-менеджер.

Рівень представлення реалізовано у вигляді 22 Razor-виглядів із власним дизайном (без Bootstrap): адаптивний сайдбар, система карток, таблиці з фільтрацією та пошуком, форми з відображенням помилок валідації. Всі шаблони використовують спільний *_Layout.cshtml*, що забезпечує єдиний стиль та відображає роль поточного користувача і його ПІБ у бічному меню. Модуль заяв на відпустку підтримує 6 типів відпусток, 4 статуси (На розгляді / Схвалено / Відхилено / Скасовано) та автоматичний розрахунок кількості днів безпосередньо у формі подачі.

Висновки та перспективи. Розроблена HRMS на базі ASP.NET Core MVC повністю автоматизує основні кадрові процеси: облік співробітників, управління структурою організації та обробку заяв на відпустку. Застосування ASP.NET Core Identity з розширеним *ApplicationUser* дозволило реалізувати захищену систему автентифікації з гнучким профілем користувача, а рольова модель забезпечує диференційований доступ до функцій системи. Перспективою розвитку є

додавання модулів табельного обліку, автоматичного розрахунку відпускних та інтеграції з поштовими сервісами для сповіщення про рішення щодо заяв.

Список використаних джерел

1. Design and development of human resource management computer system for enterprise employees - PMC. *PMC Home*. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC8682879/ (дата звернення: 19.04.2026).
2. Introduction to relationships - EF Core. *Microsoft Learn: Build with answers in reach*. URL: https://learn.microsoft.com/en-us/ef/core/modeling/relationships (дата звернення: 19.04.2026).
3. Role-based authorization in ASP.NET Core MVC. *Microsoft Learn: Build with answers in reach*. URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/security/authorization/roles?view=aspnetcore-10.0 (дата звернення: 19.04.2026).
4. Configure ASP.NET Core Identity. *Microsoft Learn: Build with answers in reach*. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity-configuration?view=aspnetcore-10.0 (дата звернення: 19.04.2026).

Лавренюк Богдан Юрійович
студент 4 курсу, групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій
(097)47-13-682
druglus007@gmail.com

ДЕСКТОПНА СИСТЕМА КЕРУВАННЯ VPN-ПІДКЛЮЧЕННЯМИ ЯК ЗАСІБ БЕЗПЕЧНОГО ДОСТУПУ ДО ІНФОРМАЦІЙНИХ РЕСУРСІВ

Постановка задачі. Умови цифрової трансформації, віддаленої роботи та доступу до розподілених сервісів підвищили роль захищених мережевих каналів. Внутрішні панелі адміністрування, навчальні лабораторії, файлові сховища, репозиторії коду та бази даних часто повинні залишатися недоступними з відкритого Інтернету, але водночас мають бути зручними для уповноважених користувачів. У такому середовищі VPN є одним із базових механізмів безпечного доступу до ресурсів, оскільки забезпечує конфіденційність передавання даних, контроль маршрутизації та ізоляцію мережевої взаємодії [1, 2].

Проблема полягає в тому, що практичне використання VPN нерідко супроводжується ручним імпортом конфігурацій, зберіганням файлів у різних каталогах, перемиканням між кількома профілями підключення та відсутністю зрозумілого механізму контролю стану тунелю. Для технічно непідготовленого користувача це створює бар'єр доступу до інформації, а для адміністратора — додаткове навантаження через помилки конфігурації, дублювання параметрів і втрату актуальних профілів. Тому актуальною є розробка програмної системи, яка б поєднувала мережеву безпеку, керованість профілями та зручний користувацький інтерфейс. Найбільш природною формою такого рішення є десктопний застосунок, здатний працювати з локальними файлами, системними процесами й параметрами мережевої взаємодії без постійної залежності від веботочення.

Мета дослідження. Метою роботи є обґрунтування підходу до побудови десктопної програмної системи керування VPN-підключеннями, яка спрощує доступ до захищених інформаційних ресурсів, зменшує кількість помилок під час роботи з конфігураціями та підвищує контрольованість мережевих підключень. Для цього визначаються функціональні компоненти системи, принципи їх взаємодії та вимоги до безпечного зберігання службових параметрів.

Результати. Запропоновано концепцію програмної системи, у якій центральною керованою сутністю є профіль доступу. На відміну від підходу, де VPN-конфігурація сприймається лише як окремий файл, профіль доступу містить назву ресурсу або сегмента мережі, параметри підключення, статус актуальності, набір допустимих дій та історію останніх запусків. Це дає змогу перейти від хаотичного зберігання конфігурацій до впорядкованого керування життєвим циклом підключень.

Архітектурно система складається з трьох взаємопов'язаних контурів. Перший контур користувацький забезпечує каталог профілів, їх пошук, активацію, деактивацію та пояснення помилок у зрозумілій формі. Другий контур виконавчий взаємодіє з мережевим стеком операційної системи, виконує імпорт або створення конфігурації, перевіряє коректність параметрів, запускає чи зупиняє тунель і відслідковує його стан. Третій контур безпековий відповідає за контроль

цілісності конфігурацій, мінімізацію збереження чутливих полів у відкритому вигляді, підтвердження критичних операцій та журналювання значущих подій [1, 3].

Суттєвим результатом є обґрунтування подієвої моделі взаємодії. Зміна стану підключення, втрата зв'язку з peer-вузлом, конфлікт маршрутів, помилка імпорту або спроба повторної активації вже працюючого тунелю мають перетворюватися на уніфіковані події. Завдяки цьому система не просто запускає VPN, а супроводжує повний сценарій доступу: інформує користувача про причину помилки, пропонує безпечну дію у відповідь і зберігає історію інцидентів для подальшого аналізу. Такий підхід є важливим у середовищах, де доступ до інформації залежить від швидкого переключення між кількома контурами мережі.

Практичне значення запропонованого рішення полягає в тому, що десктопний формат реалізації зменшує поріг входу для користувачів, які не працюють із командним рядком. Централізоване зберігання профілів, наочне відображення статусу з'єднання, швидкий імпорт конфігурацій і системні сповіщення підвищують реальну доступність захищених ресурсів для студентів, викладачів, співробітників IT-відділів і розподілених команд. У більш широкому сенсі це знижує розрив між технічно підготовленими та нетехнічними користувачами під час роботи з мережевими сервісами.

З погляду технологічної реалізації доцільним є використання Python для прикладної логіки та PyQt6 для побудови інтерфейсу, оскільки таке поєднання забезпечує швидке створення прототипу, підтримку подієвої моделі та зручну взаємодію з локальними файлами й системними процесами. При цьому критичні операції запуску VPN та перевірки конфігурації доцільно винести в окремий адаптерний модуль, щоб зміни в інтерфейсі не порушували безпекові механізми системи [2, 3].

Висновки. Отже, десктопна система керування VPN-підключеннями є перспективним програмним рішенням для організації безпечного доступу до інформаційних ресурсів у сучасному цифровому середовищі. Її цінність полягає не лише в автоматизації запуску тунелю, а у формуванні керованого життєвого циклу доступу: від імпорту профілю та перевірки параметрів до моніторингу стану, журналювання подій і захисту критичних операцій. Така система природно поєднує тематику програмних систем і архітектур із практичними задачами мережевої взаємодії та кібербезпеки. Подальший розвиток теми може бути пов'язаний із підтримкою кількох типів VPN, інтеграцією із сервісами централізованої видачі профілів, додаванням ролевих обмежень та засобів спільного використання конфігурацій у малих адміністративних командах.

Список використаних джерел

1. Donenfeld J. WireGuard: Next Generation Kernel Network Tunnel. NDSS Symposium. San Diego, 2017.

2. Rose S., Borchert O., Mitchell S., Connelly S. Zero Trust Architecture. NIST Special Publication 800-207. Gaithersburg, 2020.

3. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York: McGraw-Hill, 2020. 768 p.

Лавренюк Богдан Юрійович
студент 4 курсу, групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій
(097)47-13-682
druglus007@gmail.com

МОДУЛЬНА АРХІТЕКТУРА НАСТІЛЬНОГО ДОДАТКА ДЛЯ КЕРУВАННЯ ЛОКАЛЬНИМИ МЕРЕЖЕВИМИ ПРОФІЛЯМИ

Сучасні настільні додатки залишаються важливим класом програмного забезпечення для задач, у яких користувачеві потрібні висока швидкодія, робота з локальними файлами, стабільний інтерфейс та можливість виконання операцій без постійної залежності від вебсередовища. Особливо це стосується інструментів, що працюють із системними параметрами, мережевими профілями, службами операційної системи та конфігураційними файлами. У таких умовах якість програмного продукту визначається не лише зручністю інтерфейсу, а й продуманою архітектурою, яка спрощує підтримку, тестування та подальше масштабування функціональності [1; 2].

Проблема багатьох навчальних і практичних проєктів полягає в тому, що логіка взаємодії з інтерфейсом, робота з файловою системою, бізнес-правила та мережеві операції змішуються в одному наборі класів. Такий підхід прискорює створення першої версії програми, але вже на етапі додавання нових сценаріїв призводить до дублювання коду, складності тестування та зростання кількості помилок. Для настільного застосунку, орієнтованого на керування локальними профілями підключень, це особливо критично, оскільки будь-яка зміна в одному модулі може вплинути на збереження конфігурації, побудову інтерфейсу та коректність команд керування [2; 4].

Метою роботи є обґрунтування модульної архітектури настільного додатка, який забезпечує створення, редагування, збереження та активацію локальних мережевих профілів без прямого поєднання користувацького інтерфейсу з низькорівневими операціями. Запропонований підхід передбачає поділ системи на чотири логічні шари: презентаційний, прикладний, доменний та інфраструктурний. Презентаційний шар відповідає за вікна, діалоги, повідомлення та маршрути взаємодії користувача з програмою. Прикладний шар координує сценарії роботи, наприклад створення нового профілю або перевірку коректності введених параметрів. Доменний шар містить основні сутності системи: профіль, набір параметрів, статус з’єднання, журнал подій. Інфраструктурний шар реалізує збереження конфігурацій, читання файлів, виконання системних команд і формування резервних копій [2; 3].

Перевагою такого рішення є зниження зв’язності між компонентами. Наприклад, графічний інтерфейс може бути реалізований засобами PyQt6 або іншого фреймворку, але доменна логіка при цьому залишатиметься незалежною від конкретної технології відображення. Це дозволяє змінювати вигляд програми, не переписуючи правила обробки профілів, а також створювати модульні тести для перевірки ключових сценаріїв без запуску повного інтерфейсу. Окреме виділення сервісу валідації дає змогу перевіряти правильність імен профілів, формат адрес, наявність обов’язкових параметрів та цілісність конфігурацій ще до моменту їх збереження [3; 4].

Для настільного застосунку, що працює з локальними мережевими профілями, доцільним є використання сховища на основі структурованих JSON-файлів або SQLite. Перший варіант є простішим для резервного копіювання та ручного перегляду, другий зручнішим для журналювання змін, пошуку та масштабування кількості профілів. В обох випадках архітектурно важливо, щоб шар доступу до даних був ізольованим і підмінюваним. Це створює основу для майбутнього переходу від локального режиму роботи до синхронізації з віддаленим сервером або корпоративним середовищем без повного перепроєктування застосунку [1; 2].

Окремої уваги потребують безпека та надійність настільного рішення. Конфігураційні файли можуть містити службові параметри, адреси, ідентифікатори або ключі доступу, тому програма має підтримувати мінімізацію збереження чутливих даних, журналювання критичних дій та підтвердження небезпечних операцій. Архітектурно це реалізується через окремий сервіс політик безпеки, який виконує перевірку прав доступу до операцій, керує маскуванням чутливих полів та централізовано обробляє виняткові ситуації. Такий підхід підвищує надійність системи і спрощує аудит її поведінки [2; 4].

У результаті запропонована модульна архітектура забезпечує для настільного додатка кращу підтримуваність, передбачувану структуру коду та можливість поступового розвитку функціональності. Практична цінність підходу полягає в тому, що він однаково придатний як для навчальних проєктів, так і для прикладних інструментів керування локальними мережевими підключеннями, де важливими є простота інтерфейсу, автономність роботи та контроль над конфігураціями. Подальший розвиток дослідження може бути пов’язаний із додаванням механізму імпорту та експорту профілів, багаторівневого журналу подій, шаблонів конфігурацій та системи ролей для спільного використання програми в навчальних лабораторіях або невеликих адміністративних командах.

Отже, для настільних додатків, що працюють із системними та мережевими параметрами, ключовим чинником якості є не лише реалізація окремих функцій, а й правильно сформована архітектура. Розділення інтерфейсу, бізнес-логіки, доменних сутностей та інфраструктурних механізмів створює технічну основу для стабільного розвитку програмного продукту, спрощує його модифікацію та зменшує вартість подальшої підтримки.

Список використаних джерел

1. Fowler M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2003. 533 p.

2. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston: Prentice Hall, 2017. 432 p.

3.Riverbank Computing PyQt6 Documentation.

4. Sommerville I. Software Engineering. 10th ed. Boston: Pearson, 2016. 816 p.

Романчук Анастасія Анатоліївна,
студент групи ТЦР-42,
спеціальність 121 Інженерія програмного забезпечення,
Державний університет інформаційно-комунікаційних технологій
+380673397417
fycjvhvlvy@gmail.com

СИСТЕМА АВТОМАТИЗАЦІЇ КОНТРОЛЮ ЧЕРГ ТА ІДЕНТИФІКАЦІЇ ОСІБ НА МІЖНАРОДНОМУ КОРДОНІ

Постановка задачі. Збільшення інтенсивності транскордонного руху та підвищення вимог до безпеки держав зумовлюють необхідність впровадження автоматизованих рішень на пунктах пропуску через державний кордон. Традиційна організація черг на прикордонних переходах характеризується непрозорістю процесу очікування, відсутністю зворотного зв'язку для громадян та значним навантаженням на персонал. Задачею дослідження є проектування і розробка веб-орієнтованої інформаційної системи, що автоматизує управління електронною чергою та підтримує ідентифікацію осіб на міжнародному прикордонному переході.

Мета дослідження. Метою роботи є розробка повнофункціонального веб-додатку з підтримкою двох ролей користувачів — громадянина та прикордонника. Система має забезпечити онлайн-реєстрацію в електронну чергу, відстеження статусу очікування в режимі реального часу, а також надати прикордонному персоналу зручний інструмент управління потоком осіб і верифікації їх документів.

Результати дослідження.

Розроблена система має дворівневу клієнт-серверну архітектуру. Клієнтська частина реалізована з використанням бібліотеки React (JavaScript), що забезпечує динамічне оновлення інтерфейсу без перезавантаження сторінки. Серверна частина побудована на платформі Node.js з фреймворком Express, який обробляє HTTP-запити та реалізує REST API для взаємодії між клієнтом і базою даних.

Система складається з чотирьох основних модулів. Модуль реєстрації надає громадянину форму для введення персональних даних (ПІБ, тип і номер документа, номерний знак транспортного засобу) та вибору пункту пропуску і бажаного часу. Після підтвердження система присвоює унікальний номер реєстрації та позицію у черзі. Модуль моніторингу статусу відображає поточну позицію користувача, орієнтовний час очікування та прогрес проходження черги з автоматичним оновленням кожні декілька секунд.

Модуль прикордонника є захищеною адміністративною панеллю, що містить таблицю активної черги з можливістю фільтрації за статусом (очікує, на перевірці, пройшов). Персонал може змінювати статус запису, викликати наступну особу або видаляти записи. Модуль статистики надає зведені дані про кількість осіб, що пройшли контроль протягом поточної зміни [4].

Для ідентифікації осіб передбачено окремий мікросервіс на мові Python з використанням бібліотеки face_recognition. Мікросервіс отримує фотографію документа через REST-запит, виконує порівняння з наявними записами у базі даних та повертає результат верифікації. Зберігання всіх даних здійснюється у реляційній СУБД PostgreSQL. Інтерфейс системи адаптовано для роботи як на десктопних, так і на мобільних пристроях.

Висновки. Розроблена система автоматизації черг на міжнародному кордоні вирішує актуальну практичну задачу цифрової трансформації прикордонного контролю. Впровадження електронної черги дозволяє скоротити час очікування

громадян, підвищити прозорість процесу пропуску та знизити операційне навантаження на персонал. Обраний технологічний стек (React, Node.js, PostgreSQL, Python) забезпечує масштабованість рішення та можливість інтеграції з державними інформаційними реєстрами. Перспективами подальшого розвитку є впровадження повноцінної біометричної ідентифікації, розробка мобільного застосунку та інтеграція з системою електронних черг «Дія».

Список використаних джерел

1. Міністерство цифрової трансформації України. Концепція розвитку електронних послуг на кордоні. – Київ, 2023. – 48 с.
2. Савченко О., Прокопенко Є. Методика оцінки ефективності системи зв'язку органу охорони державного кордону. Збірник наукових праць НАДПСУ. 2015. № 3(65). С. 340–352.
3. Flanagan D. JavaScript: The Definitive Guide. 7th ed. – O'Reilly Media, 2020. – 706 p.
4. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd ed. – O'Reilly Media, 2018. – 258 p.
5. Державна прикордонна служба України. Статистика перетину кордону. – [Електронний ресурс]. – Режим доступу: https://dpsu.gov.ua/ua/statistic.

Стрельбіцький Вадим Юрійович
студент 4 курсу, групи ІСД-41
Державного університету інформаційно-комунікаційних технологій
(068) 186-64-74
youfriend471@gmail.com

МІКРОСЕРВІСНА АРХІТЕКТУРА REST API ДЛЯ СИСТЕМИ SERVICE DESK

Постановка задачі

У більшості організацій служба технічної підтримки залишається критичним елементом цифрової інфраструктури, оскільки саме через неї користувачі взаємодіють з IT-послугами, повідомляють про інциденти та подають запити на обслуговування. На практиці значна частина Service Desk рішень або побудована як монолітний застосунок, або взагалі реалізується через електронну пошту, месенджери та розрізнені таблиці. Такий підхід ускладнює контроль життєвого циклу звернення, не забезпечує прозорого дотримання SLA, створює проблеми з масштабуванням і не дає змоги швидко інтегрувати систему з корпоративними каталогами користувачів, засобами сповіщення, аналітичними модулями та базою знань.

Для сучасної системи Service Desk важливими є кілька вимог: незалежне розгортання окремих компонентів, стандартизована взаємодія між сервісами, підтримка рольової моделі доступу, можливість автоматизації типових сценаріїв і централізоване спостереження за станом системи. Саме тому доцільним є перехід до мікросервісної архітектури з REST API, де окремі бізнес-функції виділяються у самостійні сервіси. Водночас така архітектура породжує нові виклики - забезпечення узгодженості даних між сервісами, керування мережевими взаємодіями, версіонування API, моніторинг, журналювання подій та захист інформації.

Мета дослідження

Розробити стислену архітектурну модель системи Service Desk на основі мікросервісного підходу та REST API, визначити основні функціональні сервіси, описати принципи їх взаємодії, встановити базові вимоги до безпеки, масштабованості та супроводу, а також оцінити практичні переваги такого підходу для автоматизації підтримки користувачів.

Результати

У результаті дослідження запропоновано архітектуру, у якій система Service Desk складається з набору окремих сервісів, об'єднаних через API Gateway. На зовнішньому рівні користувач взаємодіє з веб-інтерфейсом або клієнтським застосунком, а всі запити надходять до шлюзу, який виконує маршрутизацію, первинну перевірку автентичності, обмеження частоти запитів та централізований контроль доступу. Внутрішній рівень містить набір мікросервісів: сервіс автентифікації та керування користувачами, сервіс обробки звернень, сервіс довідників і каталогу послуг, сервіс сповіщень, сервіс звітності та аналітики, а також сервіс бази знань. Така декомпозиція дає змогу змінювати або масштабувати окремі модулі без повного перевипуску системи.

Ключовим елементом запропонованої моделі є сервіс звернень, який відповідає за реєстрацію інцидентів, сервісних запитів, зміну статусів, призначення виконавців, контроль пріоритетів і фіксацію історії обробки. Для

забезпечення слабкої зв'язності кожний мікросервіс використовує власну базу даних, а обмін між сервісами реалізується двома способами: синхронно через REST/JSON для операцій, де потрібна швидка відповідь, та асинхронно через події для сповіщень, аудиту й інтеграції з аналітичними модулями. Це знижує взаємні залежності та підвищує стійкість системи до часткових відмов.

Для опису контрактів API запропоновано використовувати OpenAPI 3.1, що спрощує документування, тестування, генерацію клієнтів і контроль версій. Безпека архітектури базується на використанні OAuth 2.0 або JWT-токенів, рольовій моделі доступу, валідації вхідних даних, сегментації внутрішніх сервісів та централізованому логуванні. Окремо передбачено механізми спостережуваності - збирання метрик, трасування запитів і агрегування журналів подій. У практичному сценарії це дає змогу швидко виявляти вузькі місця, контролювати час реакції на звернення та підвищувати якість супроводу.

Запропонована архітектура є доцільною також з позиції подальшого розвитку системи. До неї можна без значних змін додати модуль класифікації звернень на основі AI/ML, інтеграцію з корпоративною поштою та месенджерами, модуль SLA-ескалацій або механізми самообслуговування користувачів через базу знань. У порівнянні з монолітним підходом мікросервісна модель забезпечує кращу підтримуваність, локалізацію змін, швидше розгортання нових функцій і зручнішу організацію командної розробки.

Висновки

Мікросервісна архітектура REST API для системи Service Desk є ефективним підходом до побудови сучасного програмного рішення для підтримки користувачів. Вона забезпечує модульність, незалежне масштабування компонентів, прозорість взаємодії між підсистемами та кращі умови для супроводу і розвитку програмного продукту. Практична цінність запропонованої моделі полягає у можливості поетапного впровадження окремих сервісів, підвищенні відмовостійкості й спрощенні інтеграції з іншими інформаційними системами організації.

Перспективним напрямом подальших досліджень є реалізація прототипу системи з контейнеризованим розгортанням, використанням брокера повідомлень і автоматизованим моніторингом, а також дослідження можливостей AI-модулів для інтелектуальної маршрутизації, пріоритезації та аналізу звернень користувачів.

Список використаних джерел

1. Newman S. Building Microservices, 2nd Edition. 2021. URL: https://samnewman.io/books/building_microservices_2nd_edition/

2. OpenAPI Initiative. OpenAPI Specification Version 3.1.0. 2021. URL: https://spec.openapis.org/oas/v3.1.0.html

3. Chandramouli R., Butcher Z., Sriram K. Building Secure Microservices-based Applications Using Service-Mesh Architecture. NIST SP 800-204A. 2020. URL: https://csrc.nist.gov/pubs/sp/800/204/a/final

4. Microsoft. .NET Microservices: Architecture for Containerized .NET Applications. 2023. URL: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/

5. AXELOS. ITIL Foundation, ITIL 4 Edition. 2019. URL: https://books.google.com/books/about/ITIL_Foundation_ITIL_4_Edition.html?id=qW6-xwEACAAJ

Саєнко Кирил Олександрович
студент 4 курсу
спеціальності «Інформаційні системи та технології»
Державного університету
інформаційно-комунікаційних технологій, м. Київ
kirihac@gmail.com

Данильченко Валентина Миколаївна
доцент, Доктор філософії (PhD)
Інформаційних систем та технологій
Державного університету
інформаційно-комунікаційних технологій, м. Київ

РОЗРОБКА ФУНКЦІОНАЛЬНОЇ МОДЕЛІ ТА АРХІТЕКТУРИ ІНТЕРНЕТ-МАГАЗИНУ

Постановка задачі

У процесі проєктування сучасних веб-застосунків важливе значення має формалізація структури системи та опис її поведінки. Для цього широко застосовується уніфікована мова моделювання UML, яка дозволяє представити як статичні, так і динамічні аспекти функціонування програмного забезпечення [1].

Мета дослідження

У межах даної роботи основну увагу приділено побудові діаграми класів та діаграми послідовності, що відображають архітектуру інтернет-магазину та логіку виконання ключових бізнес-процесів.

Результати

Діаграма класів (рис. 1) описує структуру системи через сукупність взаємопов’язаних сутностей. Основними елементами моделі є класи User, Product, Order, OrderItem, Payment та Delivery. Клас User відповідає за збереження даних користувача та виконання базових операцій, таких як реєстрація та авторизація. Клас Product містить інформацію про товар, включаючи його характеристики та доступність.

Центральним елементом системи є клас Order, який відображає процес оформлення замовлення. Він пов’язаний із класом User, що дозволяє ідентифікувати покупця, а також із класом OrderItem, який виступає проміжною сутністю для зв’язку замовлення з товарами. Такий підхід дозволяє реалізувати зв’язок «багато до багатьох» між замовленнями та товарами [2]. Класи Payment і Delivery відповідають за обробку оплати та організацію доставки відповідно. Їх виділення в окремі компоненти дозволяє ізолювати логіку взаємодії із зовнішніми сервісами та забезпечити розширюваність системи. Таким чином, діаграма класів відображає статичну архітектуру системи та визначає взаємозв’язки між її складовими, що є основою для подальшої програмної реалізації.

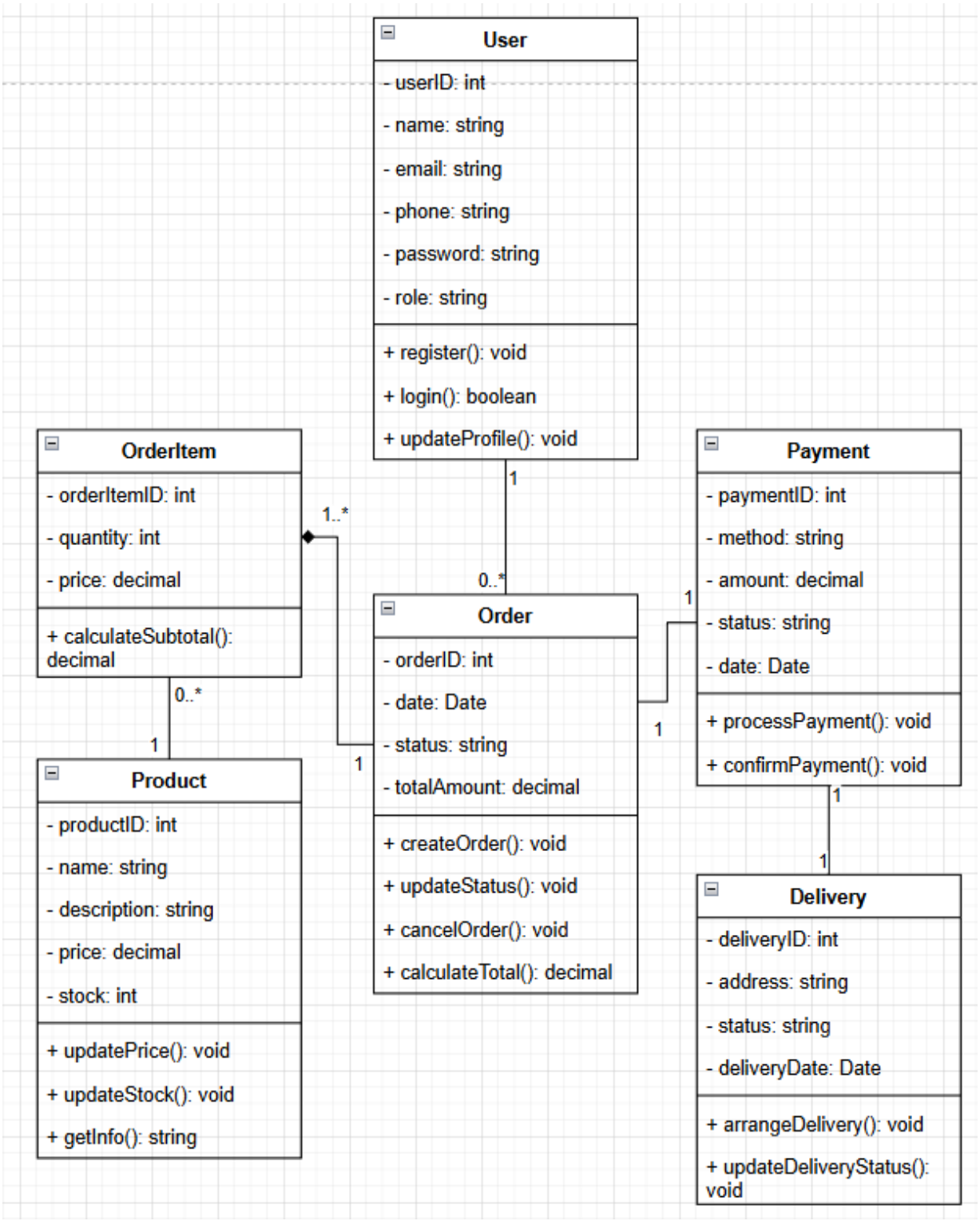


Рис. 1 - Діаграма класів

Для опису динамічної поведінки системи використано діаграму послідовності (рис. 2), яка демонструє порядок взаємодії між об’єктами під час виконання сценарію оформлення замовлення.

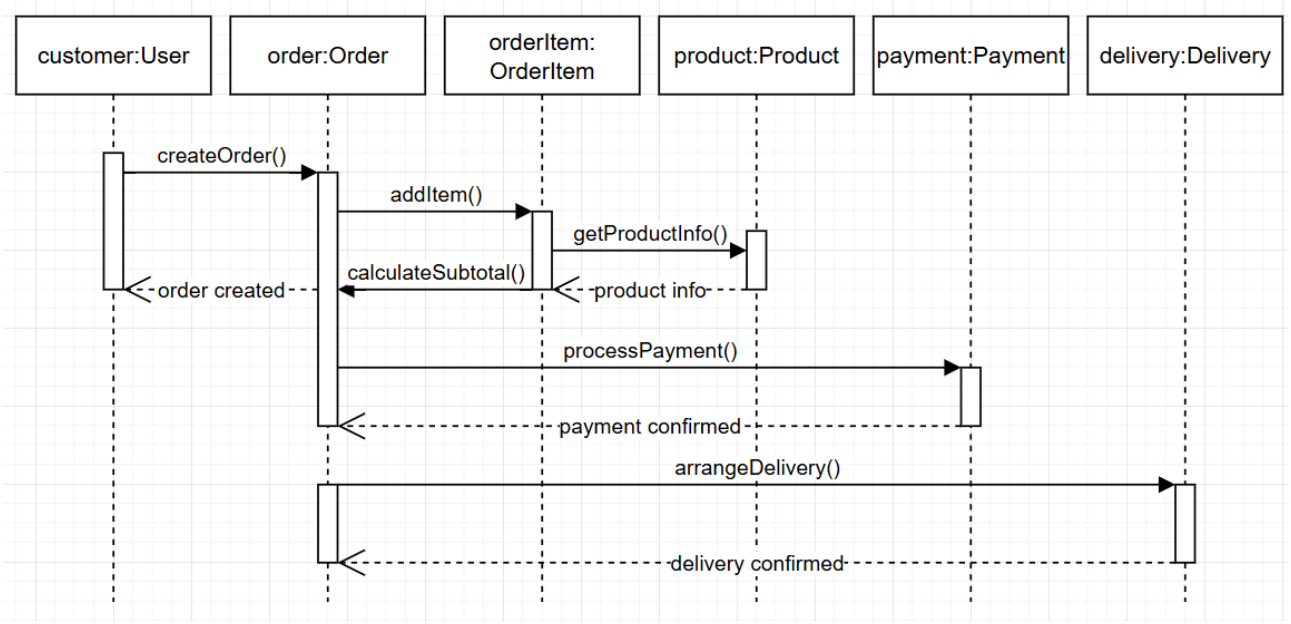


Рис. 2 - Діаграма послідовності

У межах даного сценарію користувач ініціює створення замовлення, після чого система формує відповідний об’єкт та додає до нього обрані товари. Далі відбувається отримання інформації про товар і обчислення вартості замовлення. Після підтвердження користувачем система ініціює процес оплати, який обробляється відповідним модулем. У разі успішного завершення оплати система переходить до наступного етапу організації доставки, що здійснюється через відповідний сервіс. Завершальним етапом є підтвердження замовлення, що свідчить про успішне виконання всіх операцій. Діаграма послідовності дозволяє чітко простежити порядок виклику методів і взаємодію між компонентами системи, що є важливим для розуміння логіки роботи веб-застосунку [3].

Висновки

Отже, використання UML-діаграм у процесі проєктування дозволяє комплексно описати архітектуру та функціональну модель інтернет-магазину. Діаграма класів формує уявлення про структуру системи, тоді як діаграма послідовності відображає її поведінку в часі, що забезпечує системний підхід до розробки програмного забезпечення.

Список використаних джерел

1. Object Management Group. Unified Modeling Language (UML) Specification. URL: https://www.omg.org/spec/UML/ (дата звернення: 22.04.2026).

2. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston: Addison-Wesley, 2004. 208 p.

3. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. 2nd ed. Boston: Addison-Wesley, 2005. 496 p.

Лазневий Андрій Валерійович
Бакалаврат 4 курсу, групи ТЦР-43
Державний університет інформаційно-комунікаційних технологій
(097) 028-59-90
st7906260@stud.duikt.edu.ua

ІНФОРМАЦІЙНА СИСТЕМА ОБЛІКУ ПАЦІЄНТІВ МЕДИЧНОГО ЦЕНТРУ

Постановка задачі. Медичний центр – це складна організаційна структура, діяльність якої спрямована на надання діагностичних, лікувальних та профілактичних послуг населенню. У сучасних умовах такі установи функціонують у середовищі постійного зростання обсягів інформації, що пов'язано з веденням медичних карток пацієнтів, історій хвороб, результатів обстежень, призначень лікарів та інших даних [1]. У багатьох медичних закладах досі існує низка проблем, пов'язаних з організацією обліку пацієнтів та веденням медичної документації: використання паперових носіїв, дублювання даних, помилки під час обробки інформації та низька швидкість доступу до необхідних відомостей. Тому актуальним є розроблення інформаційної системи, яка забезпечить автоматизацію основних процесів діяльності медичного центру.

Мета дослідження. Метою дослідження є аналіз особливостей функціонування медичного центру та розроблення веб-системи обліку пацієнтів, яка забезпечить зберігання, обробку та швидкий доступ до медичної інформації, а також підвищить ефективність роботи персоналу та якість обслуговування пацієнтів.

Результати дослідження. Медичні інформаційні системи (МІС) являють собою програмно-технічні комплекси, призначені для збору, зберігання, обробки та аналізу медичних даних [1]. Вони забезпечують централізований доступ до інформації про пацієнтів, дозволяють вести електронні медичні картки, формувати звітність, а також автоматизувати процес запису на прийом та взаємодію між лікарями і пацієнтами. Використання таких систем значно підвищує ефективність роботи медичного центру та сприяє покращенню якості медичних послуг.

Основних функції, які забезпечують ефективну організацію роботи медичного центру: облік пацієнтів, ведення історії хвороби та запис на прийом (рис. 1).

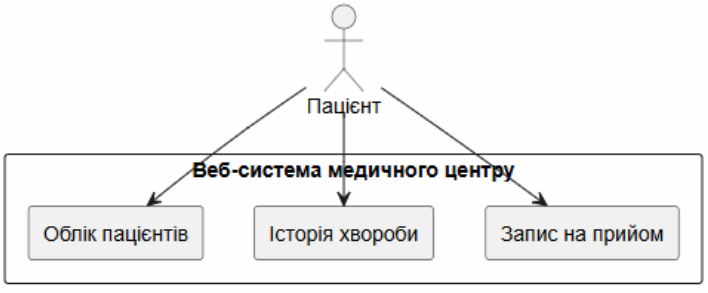


Рис. 1. Основні функції веб-системи обліку пацієнтів медичного центру.

Функція обліку пацієнтів передбачає зберігання та управління персональними даними пацієнтів, включаючи їхні анкетні відомості, контактну

інформацію та ідентифікаційні дані. Це дозволяє швидко знаходити необхідну інформацію та забезпечує зручність роботи медичного персоналу.

Ведення історії хвороби є важливою складовою медичної інформаційної системи, оскільки дозволяє накопичувати та зберігати інформацію про стан здоров'я пацієнта, результати обстежень, призначення лікарів та динаміку лікування. Це забезпечує безперервність медичного обслуговування та підвищує якість прийняття клінічних рішень [2].

Функція запису на прийом забезпечує можливість планування візитів пацієнтів до лікарів, формування розкладу роботи медичного персоналу та оптимізацію використання часу. Завдяки автоматизації цього процесу зменшується кількість черг, підвищується зручність для пацієнтів та ефективність роботи медичного центру.

Для наочного відображення взаємодії основних функцій медичної інформаційної системи доцільно представити їх у вигляді структурної схеми (рис. 1).

У багатьох медичних установах досі існує низка проблем, пов'язаних з організацією обліку пацієнтів та веденням медичної документації. Основними з них є використання паперових носіїв, дублювання даних та наявність помилок у процесі обробки інформації.

Висновки та перспективи. У дослідженні розглянуто особливості функціонування медичного центру та основні можливості медичних інформаційних систем. Проведений аналіз дозволив визначити ключові функції веб-системи обліку пацієнтів, які сприяють автоматизації роботи персоналу, підвищенню якості медичного обслуговування та зменшенню кількості помилок при роботі з даними.

Перспективою подальших досліджень є розширення функціоналу системи шляхом інтеграції електронного документообігу, модулів аналітики та прогнозування, а також впровадження мобільного доступу для лікарів і пацієнтів.

Список використаних джерел

1. Microsoft. SQL Server Documentation [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/sql/

2. Mozilla. HTML5 Documentation [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML

Закаблук Едуард Євгенович
студент 4 курсу, групи КНД-43
Державного університету
інформаційно-комунікаційних технологій
(063)-47-47-924
vyshnivskyi.viktor@gmail.com
Науковий керівник: Вишнівський Віктор Вікторович,
доктор технічних наук, професор, завідувач кафедри Комп'ютерних наук Державного університету інформаційно комунікаційних технологій, м. Київ

ПЕРЕВАГИ ВИКОРИСТАННЯ ОПЕРАЦІЙНИХ СИСТЕМ З ВІДКРИТИМ КОДОМ ДЛЯ ПОБУДОВИ КОРПОРАТИВНОЇ СЕРВЕРНОЇ ІНФРАСТРУКТУРИ

Постановка задачі.

Сучасне підприємство будь-якого масштабу спирається на серверну інфраструктуру як на критично важливий елемент операційної діяльності. Вибір операційної системи для серверного середовища безпосередньо визначає рівень захищеності корпоративних даних, стабільність роботи сервісів і сукупну вартість утримання IT-інфраструктури. Попри те що на ринку присутні зрілі комерційні рішення – насамперед сімейство Microsoft Windows Server, статистика розгортань свідчить про інше: за даними аналітичних звітів, понад 96% публічних веб-серверів і переважна більшість хмарних вузлів функціонують під управлінням Linux-систем. Постає питання: чим зумовлена настільки переконлива перевага відкритих рішень і наскільки вони відповідають реальним потребам корпоративного середовища?

Мета дослідження.

Метою роботи є комплексний аналіз ключових переваг операційних систем з відкритим вихідним кодом (передусім Ubuntu Server і CentOS Stream) порівняно з комерційними аналогами у трьох вимірах: інформаційна безпека, операційна стабільність та економічна ефективність. На основі цього аналізу формулюються рекомендації щодо доцільності застосування відповідних рішень у корпоративному серверному середовищі.

Результати дослідження.

Принцип «Чотирьох очей» полягає в тому, що відкритий вихідний код перевіряється значно ширшим колом фахівців, ніж будь-яка закрита кодова база. Це дозволяє виявляти вразливості швидше й у більшому обсязі. Дослідження Manzoor J. підтверджує, що open-source інструменти класу SIEM, зокрема Wazuh та OpenSearch, демонструють рівень виявлення загроз, порівнянний із пропрієтарними системами, проте без прив'язки до вендора та за відсутності ліцензійних витрат. У середовищі Linux права доступу реалізовані на рівні ядра через мандатний і дискреційний механізми, а модульна архітектура ядра дозволяє підключати виключно необхідні компоненти, мінімізуючи поверхню атаки. Порівняльне дослідження Kabir M.H. на вибірці IT-спеціалістів засвідчує, що Linux-сервери статистично рідше стають жертвами успішних зломів, ніж аналогічні Windows-сервери в однакових мережевих умовах.

Корпоративний сервер повинен функціонувати безперервно. Архітектурна особливість ядра Linux полягає у можливості оновлення більшості системних

компонентів без перезавантаження. Windows Server вимагає регулярних перезавантажень після встановлення оновлень безпеки. Дослідження продуктивності веб-серверів, проведене Jakic P., фіксує, що Ubuntu Server під навантаженням демонструє меншу варіативність часу відгуку та вищу пропускну здатність у порівнянні з Windows Server за аналогічних апаратних конфігурацій. Саме тому критичні служби від банківських транзакційних систем до урядових реєстрів – традиційно розгортаються на Linux-платформах.

Економічна ефективність. Ліцензія Microsoft Windows Server Datacenter 2022 коштує від 6 155 USD за двопроцесорний вузол. Ubuntu Server, Debian, CentOS Stream та інші корпоративні дистрибутиви поширюються безкоштовно. Для підприємств із десятками серверних вузлів ця різниця формує відчутний резерв бюджету, який може бути перерозподілений на фахову підтримку й апаратне оновлення. Стаття Петровського М. П. описує реальний кейс побудови масштабної серверної інфраструктури ВНЗ виключно на базі вільного ПЗ, де відмова від комерційних ліцензій дозволила скоротити операційні витрати більш ніж на 40%. Ця закономірність є ще більш вираженою в хмарних і гібридних середовищах: дослідження у галузі безсерверних архітектур на відкритих платформах Хоменко В. підкреслює, що відсутність вендорської залежності суттєво знижує сукупну вартість масштабування.

Висновки та перспективи

Операційні системи з відкритим вихідним кодом, зокрема Ubuntu Server і суміжні дистрибутиви, є технічно обґрунтованим і економічно виправданим вибором для корпоративного серверного середовища. Модель відкритого коду забезпечує вищу прозорість і оперативнішу реакцію спільноти на вразливості, ніж закриті пропрієтарні рішення. Архітектурні особливості ядра Linux сприяють стабільній роботі без планових перезавантажень, що є критичним для сервісів із вимогою безперервної доступності. Відсутність ліцензійних витрат вивільняє бюджетні кошти, які підприємство може спрямувати на посилення апаратної бази або підготовку кадрів. Сукупність цих чинників пояснює переважаюче становище Linux у сучасній серверній інфраструктурі та підтверджує доцільність переходу на відкриті платформи для підприємств, що прагнуть оптимізувати IT-витрати без втрати надійності.

Список використаних джерел

1. Петровський М. П. та ін. Побудова інформаційного середовища ВНЗ. Вісник Хмельницького національного університету. 2021. URL: https://journals.khnu.km.ua/vestnik/wp-content/uploads/2021/01/37-11.pdf
2. Manzoor J., Waleed A., Jamali A. F., Masood A. Cybersecurity on a budget: Evaluating security and performance of open-source SIEM solutions for SMEs. PLoS ONE. 2024. Vol. 19, № 3. e0301183. DOI: 10.1371/journal.pone.0301183
3. Kabir M. H. et al. Linux vs. Windows: A Comparison of Two Widely Used Platforms. ResearchGate. 2022. URL: https://www.researchgate.net/publication/358007512
4. Хоменко В. та ін. Architecting Cloud-Native Serverless Solutions on Open Source Platforms. Вісник КНТУ. 2024. URL: https://journals.kntu.kherson.ua/index.php/visnyk_kntu/article/view/843
5. Jakic P. et al. Comparative Analysis of the Impact of Server Operating Systems on Web Site Performance. ResearchGate. 2021. URL: https://www.researchgate.net/publication/352888701

Санкіна Яна Сергіївна
студентка 4 курсу, групи ТЦР-41
Державний університет інформаційно-комунікаційних технологій
(066)038-08-71
st8138516@stud.duikt.edu.ua

ПРОЄКТУВАННЯ СЛАБКОПОВ'ЯЗАНОЇ АРХІТЕКТУРИ ІГРОВОГО ЗАСТОСУНКУ НА БАЗІ ПОДІЄВОЇ МОДЕЛІ

Постановка задачі. Розробка ігрових застосунків сучасності часто стикається з проблемою монолітної архітектури. Класи в ній мають жорсткі прямі посилання один на одного. Це створює труднощі при масштабуванні проєкту: зміна логіки в одному модулі, наприклад, у системі здоров'я, змушує проводити рефакторинг у десятках залежних об'єктів, таких як інтерфейс, штучний інтелект, звукові системи. Критично важливим є пошук рішень, що мінімізують пов'язаність та забезпечують високу автономність програмних компонентів один від одного [1].

Мета дослідження. Проєктування і обґрунтування слабкопов'язаної архітектури ігрового симулятора, що поєднує паттерн диспетчера подій для комунікації систем і підхід для управління ігровими даними.

Результати дослідження. Основою розробленої системи став централізований диспетчер подій. На відміну від стандартних методів Unity, таких як SendMessage або сильної пов'язаності через Singleton, обраний підхід базується на реєстрації типів подій [2].

Спочатку створюються класи-контейнери для подій, що не містять логіки, а лише дані. Потім диспетчер подій використовує словник Dictionary<Type, Delegate>, де ключем є тип події, а значенням – список методів-обробників.

Будь-який ігровий об'єкт може опублікувати подію одним рядком коду і йому не потрібно знати, хто на неї підписаний. Це дозволило повністю ізолювати ігрову логіку від графічного інтерфейсу. Наприклад, при зміні характеристик персонажа система UI реагує на подію оновлення даних, що виключає використання ресурсомісткого методу Update() для постійного моніторингу станів [3].

Для управління характеристиками великої кількості видів ігрових і неігрових персонажів – їх швидкість, здоров'я, витривалість – було застосовано Data-Driven Design. Традиційне хардкод-визначення параметрів у стандартному класі було замінено на використання ScriptableObject [4]. Це дало деякі інженерні переваги: дані зберігаються в окремих ассетах на диску і не дублюються для кожного екземпляра об'єкта в ігровій сцені; параметри балансу можна змінювати безпосередньо в інспекторі Unity під час роботи програми, що прискорює розробку; дані із ScriptableObject легко інтегруються в систему збереження і завантаження через JSON-формат.

На рис. 1 продемонстровано, як обмін інформацією між різними незалежними модулями системи відбувається виключно через диспетчер подій.

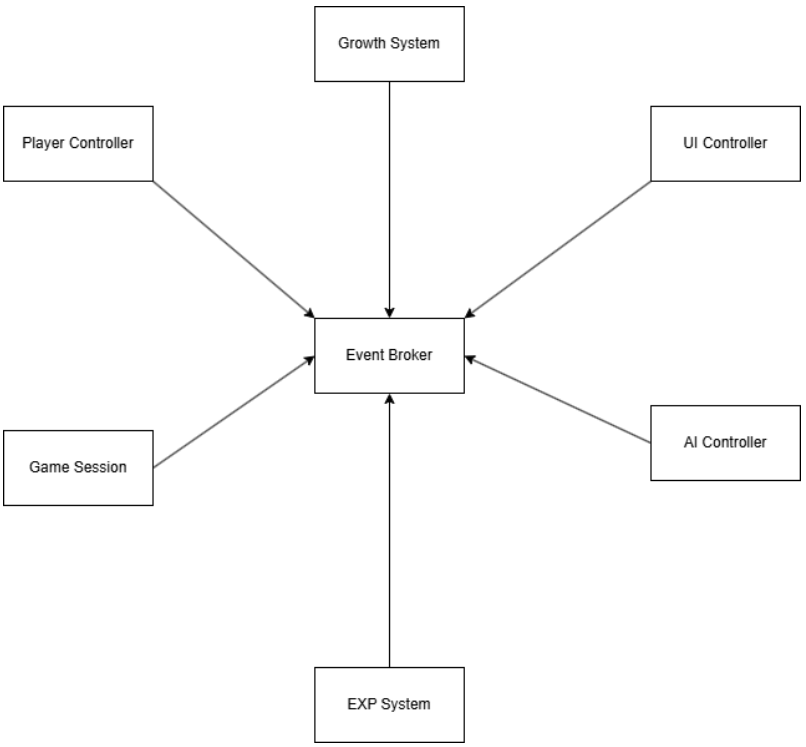


Рис. 1. Архітектурна схема взаємодії систем застосунку через Event Broker

Профілювання розробленого застосунку показало, що перехід від прямого звернення між об’єктами до подієвої моделі знизив навантаження на CPU у сценах з великою кількістю активних агентів. Це зумовлено відсутністю порожніх перевірок у циклах обробки кадрів. Окрім цього, модульність дозволяє додавати нові види ігрових персонажів або нові ігрові механіки без втручання у вже існуючий стабільний код центральної системи.

Висновки. Застосування комбінації диспетчера подій і окремих конфігураційних файлів є оптимальним архітектурним підходом для розробки складних ігрових симуляцій. Він відповідає принципам SOLID, зокрема принципу відкритості/закритості і забезпечує високу гнучкість програмного продукту [5]. Результати розробки підтверджують, що інженерний підхід до архітектури гри дозволяє створювати стабільні і масштабовані системи, які готові для подальшого масштабування.

Список використаних джерел:

1. Gregory J. Game Engine Architecture. CRC Press. 2018. 1240 p.
2. Nystrom R. Game Programming Patterns. Genever Benning. 2014. 354 p.
3. Freeman E., Robson E. Head First Design Patterns. O'Reilly Media. 2020. 672 p.
4. Folmer E. Component Based Game Development. Component-Based Software Engineering. 2007. Vol. 4608, P. 66–73.
5. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1994. 395 p.

Антонов Олександр Лаврентійович
студент 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
(097) 256-17-79
eskandre50@gmail.com

РЕАЛІЗАЦІЯ СИСТЕМИ АВТОРИЗАЦІЇ ТА ВЗАЄМОДІЇ КОРИСТУВАЧА У ВЕБ-ДОДАТКУ «ОНЛАЙН БІБЛІОТЕКА»

Постановка задачі. У сучасних веб-додатках значна частина функціональності пов’язана з персоналізацією роботи користувача. Для онлайн бібліотеки важливо не лише забезпечити перегляд книжок, але й надати можливість працювати з власним обліковим записом, зберігати вибрані матеріали та отримувати доступ до індивідуальних функцій системи. Тому реалізація авторизації та взаємодії користувача із веб-додатком є важливою частиною розробки.

Однією з проблем під час створення таких систем є організація стабільної роботи користувацьких сесій та розмежування функцій для авторизованих і неавторизованих користувачів. Також необхідно забезпечити простий та зрозумілий інтерфейс для входу і реєстрації.

Мета дослідження. Метою роботи є реалізація системи авторизації користувачів та персоналізованої взаємодії користувача із веб-додатком «Онлайн бібліотека» на основі Flask [1].

Результати дослідження. У ході роботи у веб-додатку було реалізовано окремі сторінки для реєстрації та входу користувача. Після успішної авторизації користувач отримує доступ до функцій, які недоступні для звичайного перегляду, зокрема до роботи зі списком обраних книжок.

Система побудована таким чином, що кожна дія користувача обробляється окремим маршрутом Flask [1]. Це дозволяє розділити функціональність додатку та зробити структуру коду більш зрозумілою. Для підтримки активного входу в систему використовується бібліотека Flask-Login [4], завдяки якій користувачу не потрібно повторно вводити дані під час переходу між сторінками.

Окрему увагу приділено реалізації функції «обраного». Авторизований користувач може зберігати книжки у власному списку, що спрощує повторний доступ до потрібних матеріалів та робить взаємодію із системою більш зручною. Дані користувачів та інформація про книжки зберігаються у базі даних PostgreSQL [2].

Для формування сторінок використовується функція render_template фреймворку Flask [1], яка дозволяє передавати інформацію про користувача та результати обробки запитів у HTML-шаблони. Завдяки цьому серверна логіка відокремлюється від відображення сторінок, що спрощує підтримку та розвиток проєкту.

Також у процесі реалізації було враховано зручність користувацького інтерфейсу. Форми входу та реєстрації мають просту структуру і не перевантажують користувача зайвими діями, що позитивно впливає на роботу із системою [3].

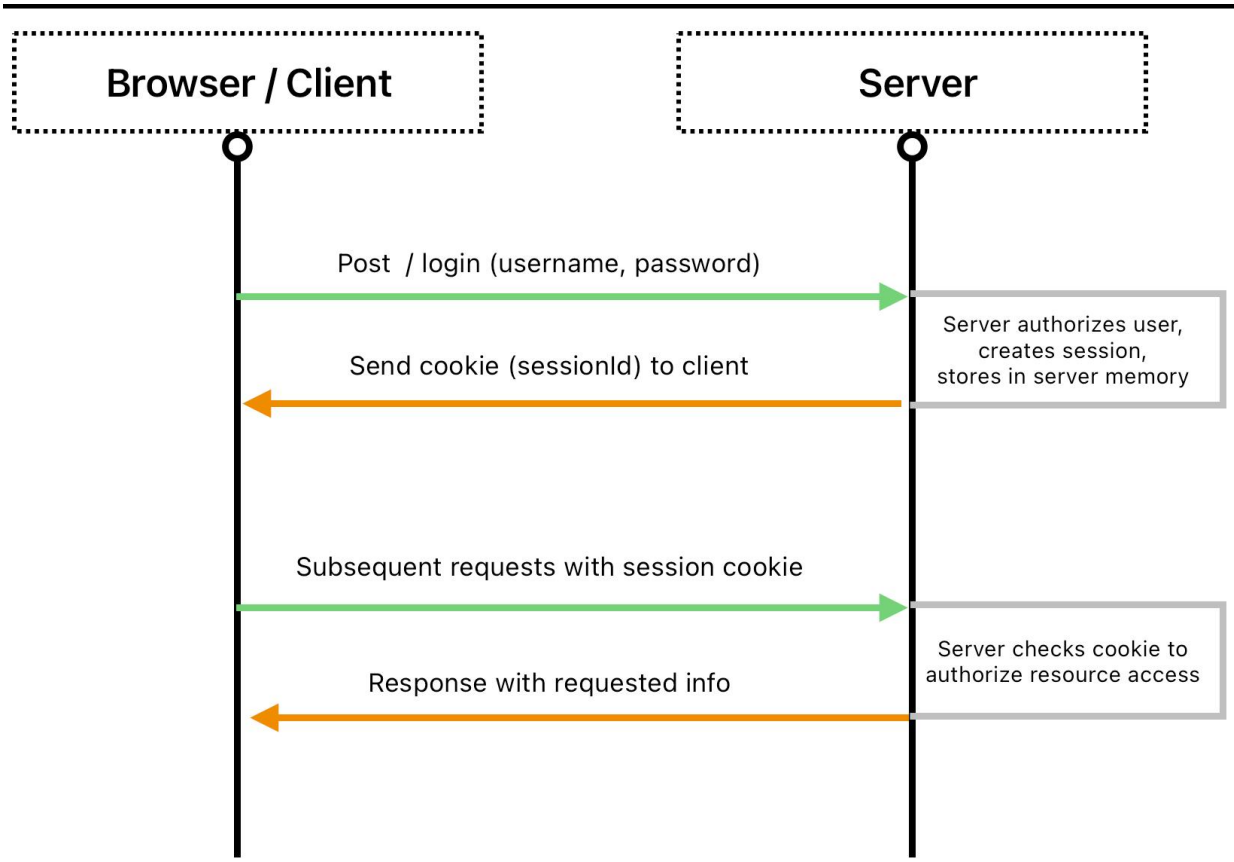


Рис. 1. Схема роботи системи авторизації користувача

Висновки та перспективи. У результаті роботи було реалізовано систему авторизації та взаємодії користувача з веб-додатком «Онлайн бібліотека». Реалізований підхід дозволяє організувати персоналізовану роботу користувачів та спрощує використання основного функціоналу системи.

Подальший розвиток додатку може включати розширення користувацьких можливостей, реалізацію підтвердження електронної пошти, відновлення паролю та створення окремих ролей для різних типів користувачів.

Список використаних джерел

1. Flask Documentation. – [Електронний ресурс]. – Режим доступу: https://flask.palletsprojects.com

2. PostgreSQL Documentation. – [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs

3. Mozilla Developer Network. JavaScript Guide. – [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org

4. Flask-Login Documentation. – [Електронний ресурс]. – Режим доступу: https://flask-login.readthedocs.io

Бахмут Михайло
Студент 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
Науковий керівник
Старший викладач Олейніков І.А.
(095)795-66-38
mbahmut0@gmail.com

ПРОЕКТУВАННЯ АРХІТЕКТУРИ ГРИ ТИПУ ROGUELIKE НА ОСНОВІ UNREAL ENGINE 5

Постановка задачі. Створення roguelike-гри є складним технічним викликом, оскільки така програмна система складається з багатьох взаємозалежних підсистем: управління персонажем, бойової механіки, системи характеристик, ворогів, предметів, випадкових винагород, прогресу ігрового процесу та функції збереження результатів. Якщо всі ці елементи реалізовуються як жорстко пов’язані між собою Blueprint-и, проєкт швидко стає громіздким, що значно ускладнює його підтримку, тестування та подальше розширення. Особливо гостро така проблема проявляється при використанні підходу Blueprint-first в Unreal Engine 5. Візуальне програмування у цьому середовищі дозволяє швидке створення ігрової логіки без необхідності писати код на C++, проте потребує ретельно спроєктованої архітектури для запобігання утворенню надмірно складних і некерованих графів.

Тому нагальною стає необхідність розробки модульної архітектури гри, що забезпечує чітке розподілення відповідальностей між класами та компонентами. У такому підході повторювану ігрову логіку раціонально виділяють в окремі Actor Components, а всі параметри ігрових об’єктів зберігаються у data-driven форматі, що полегшує їх редагування, тестування та масштабування.

Мета дослідження. Головною метою цього проєкту є створення архітектури програмної системи для roguelike-гри на базі Unreal Engine 5 з використанням підходу Blueprint-first, Gameplay Framework, Actor Components та організації ігрових даних на основі даних. Для досягнення мети необхідно виконати такі завдання: розподілити логіку між персонажем, контролером, ігровим режимом, станом гри та станом гравця класу; передати повторювані функції на аутсорсинг компоненту; розробити дані для предметів, атрибутів та винагород; забезпечити масштабованість системи без переписування більшості кодів blueprint.

Результати. В рамках цього дослідження запропонована архітектура для MVP версії roguelike гри, яка ґрунтується на використанні Gameplay Framework від Unreal Engine. Керовані ігрові об'єкти реалізовані через спеціальний class Character, який використовується для гравця та ворога. Controller призначений для управління персонажем, що дозволяє відокремити логіку управління від фізичного уявлення об'єкта в ігровому світі.

Структура класів має такий вигляд: для гравця використовується BP_CharacterBase_Player, а для ворогів — BP_CharacterBase_Enemy. Спільна логіка характеристик винесена в окремий Actor Component BP_AC_Stats, який використовується для роботи зі здоров’ям, шкодою, швидкістю руху та іншими параметрами ігрових сутностей. Логіка предметів і нагород реалізується окремо

від класів персонажа та ворогів. Це дозволяє не дублювати однакову логіку в різних Blueprint-ах.

Логіка забігу розподілена між кількома класами. BP_GM_Run реалізує правила генерації: початок забігу, спавн ворогів, умови завершення та нагородження гравця. BP_GS_Run зберігає поточний ігровий процес: час, складність, кількість переможених ворогів та поточний стан арени. A_BP_PS_PlayerRun містить дані гравця: рівень, досвід, золото та знайдені предмети. Навіть під час створення MVP для single player цей підхід буде корисним, оскільки він дозволить у майбутньому легко додати кооперативний режим.

Одним із важливих результатів є використання data-driven підходу. Замість використання hardcoded параметрів предмети, вороги та нагороди реалізуються через Data Asset або Data Tables. Наприклад, предмет містить ID, ім'я, опис, рідкість, іконку, тип ефекту та число ефекту. Це дозволяє змінювати баланс без редагування основної логіки персонажа, спрайтів чи системи нагород.

Також запропоновано винесення логіки нагороди з її джерела. Наприклад, компонент BP_Chest не змінює характеристики гравця безпосередньо, а вибирає з пулу випадковий предмет, який вже сам за застосування нагороди впливає на клас BP_AC_Stats. Це знижує зв'язність системи і дозволяє використовувати ту саму логіку нагород і для скринь, і для ворогів, і для босів.

Висновки. У цій роботі запропоновано варіант модульної архітектури програмної системи для гри в жанрі roguelike на базі Unreal Engine 5. Основою запропонованого рішення стало використання Blueprint-first підходу, об'єднаного з Gameplay Framework, Actor Components та data-driven підходом для організації даних.

Приблизно структура дозволяє розділити відповідальність між класами: Character відповідає за ігровий об'єкт, Controller - за керування, GameMode - за правила забігу, GameState - за поточний ігровий процес, PlayerState - за дані гравця, а Actor

Список джерел

1. Epic Games. Gameplay Framework в Unreal Engine.
2. Epic Games. Blueprints Visual Scripting в Unreal Engine.
3. Epic Games. Actor Component API Reference.
4. Epic Games. Data Asset / Primary Data Asset Documentation.
5. Romero M., Sewell B. Blueprints Visual Scripting для Unreal Engine 5. Packt Publishing, 2022.

Мирончук Микола Володимирович
студент 4 курсу
спеціальності «Інформаційні системи та технології»
Державного університету
інформаційно-комунікаційних технологій, м. Київ
st8923769@stud.duikt.edu.ua

Сагайдак Віктор Анатолійович
доцент, Доктор філософії (PhD)
Інформаційних систем та технологій
Державного університету
інформаційно-комунікаційних технологій, м. Київ

СИСТЕМА ІНТЕЛЕКТУАЛЬНОЇ МАРШРУТИЗАЦІЇ ЦИФРОВИХ ДОКУМЕНТІВ НА ОСНОВІ ЧАТ-ІНТЕРФЕЙСІВ

Постановка проблеми

У сучасних організаціях постійно зростає кількість електронних документів, що ускладнює їх обробку та контроль виконання. Використання класичних веб-систем не завжди забезпечує необхідну швидкість взаємодії, оскільки користувачеві потрібно окремо входити в систему та виконувати ряд дій. Це створює додаткове навантаження та знижує ефективність роботи. Тому доцільним є пошук альтернативних способів взаємодії з інформаційними системами, зокрема із застосуванням чат-інтерфейсів [2].

Мета роботи

Метою даної роботи є розробка підходу до організації системи, яка дозволяє автоматично визначати маршрут проходження документів і забезпечує взаємодію з користувачем через чат-інтерфейс.

Основні результати

У запропонованому рішенні основна увага приділяється не лише функціональності, а й структурі системи. Система побудована таким чином, щоб окремі її частини могли працювати незалежно одна від одної, що відповідає сучасним підходам до проєктування програмного забезпечення [1]. Така організація дозволяє спростити підтримку та подальше розширення функціоналу.

Обмін даними між компонентами здійснюється через API, що дає можливість легко інтегрувати додаткові сервіси або змінювати окремі модулі без впливу на всю систему [2]. Для зберігання інформації використовується база даних, у якій зберігаються як самі документи, так і їх характеристики, необхідні для подальшої обробки.

Однією з ключових особливостей є автоматичне визначення подальшого руху документа. Після його додавання система аналізує основні параметри та визначає, кому і на якому етапі він має бути переданий. Це дозволяє зменшити кількість ручних операцій і уникнути затримок у процесі обробки.

Використання чат-інтерфейсу дає змогу значно спростити взаємодію користувача із системою. Через нього можна отримувати інформацію про стан документів, виконувати окремі дії та отримувати сповіщення без необхідності

відкривати веб-додаток [4]. Такий підхід є зручним у повсякденному використанні та не потребує додаткового навчання користувачів.

Окрему увагу приділено питанням безпеки, оскільки обробка документів передбачає роботу з конфіденційною інформацією. У системі передбачено базові механізми контролю доступу та захисту даних, що відповідають сучасним рекомендаціям у сфері веб-безпеки [5].

Висновки

Розглянутий підхід до організації системи дозволяє підвищити ефективність роботи з документами та спростити взаємодію користувача із системою. Використання чат-інтерфейсу разом із сучасними архітектурними рішеннями забезпечує гнучкість та зручність використання. Подальший розвиток може бути пов’язаний із розширенням функціоналу та інтеграцією з іншими інформаційними системами.

Список використаних джерел

1. Newman S. Building Microservices. Sebastopol: O’Reilly Media, 2021. 280 p.
2. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures: dissertation. Irvine: University of California, 2000. 162 p.
3. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 1994. 395 p.
4. Telegram. Bot API Documentation [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api
5. OWASP Foundation. OWASP Top 10: Web Application Security Risks [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-top-ten/

Журба Володимир Ігорович,
студент групи КНД-42,
спеціальність F3 Комп'ютерні науки,
Державного університету інформаційно-комунікаційних технологій
(095)713-61-17
vladimirzhurba2005@gmail.com

РОЗРОБКА WEB APPLICATION E-LIBRARY НА МОВІ PYTHON З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ DJANGO

Постановка задачі. У сучасному середовищі значна частина інформації вже не використовується у паперовому вигляді, а зберігається в електронному форматі. Це стосується і книг, і навчальних матеріалів, і різних методичних ресурсів. Такий підхід значно спрощує доступ до інформації, адже користувач може отримати потрібні матеріали у будь-який момент без прив'язки до конкретного місця. Разом з тим з'являється інша проблема — велика кількість файлів швидко стає неорганізованою. Коли книги просто зберігаються окремо, без чіткої структури, знайти потрібну інформацію стає складніше, особливо якщо мова йде про велику кількість матеріалів різної тематики. Хоча вже існують готові рішення у вигляді електронних бібліотек, вони не завжди підходять для простих задач. Часто такі системи або перевантажені функціоналом, або мають складну структуру, яка не є зручною для користувача без досвіду. Тому виникає ідея створити веб-застосунок, який буде виконувати базові функції електронної бібліотеки, але залишатиметься простим у використанні.

Мета дослідження. Метою даної роботи є розробка веб-застосунку електронної бібліотеки з використанням мови Python та фреймворку Django, який дозволить здійснювати пошук, перегляд та базове управління електронними книгами.

Результати дослідження. Під час виконання роботи було проаналізовано підходи до створення веб-застосунків, які працюють з інформаційними даними. Був виконаний аналіз існуючих рішень - розглянуто такі системи, як Calibre та Open Library:

- Calibre: хоча система має потужний інструментарій для конвертації та редагування метаданих, її інтерфейс перевантажений великою кількістю дрібних іконок та технічних налаштувань, що збиває з пантелику початківців.
- Open Library: через величезний обсяг даних та складну систему навігації між різними виданнями однієї книги, користувачеві часто важко швидко знайти конкретний файл для перегляду.

З урахуванням цього було обрано Django як основу для майбутньої системи, оскільки він спрощує роботу з базою даних і вже містить готові рішення для типових задач [1].

Особливістю Django є те, що він задає певну структуру проєкту. Дані, логіка і відображення розділені між собою, що дозволяє краще контролювати процес розробки і не змішувати різні частини коду [4]. Django використовує архітектуру MVT (Model-View-Template). Для електронної бібліотеки це має значення, оскільки потрібно працювати одразу з кількома типами даних: книгами, авторами, категоріями та користувачами.

Основою вигляду системи складає база даних, у якій буде зберігатися інформація про книги. При цьому кожен запис міститиме основні характеристики, наприклад назву, автора, опис або категорію. Для етапу розробки це може бути SQLite, а для розгортання — PostgreSQL або MySQL. Такий підхід дозволяє не просто накопичувати файли, а створювати впорядкований список матеріалів [2].

Окремо варто відзначити функцію пошуку. Для користувача це один із ключових елементів, адже без нього робота з великою кількістю книг значно ускладнюється. Тут виконується стандартний пошук засобами Django ORM (фільтрація за запитами), чи планується використання спеціалізованих інструментів (наприклад, Q-об'єкти для складних запитів). Тому передбачається можливість знаходити книги за різними ознаками, а не тільки за точною назвою.

Також розглядається використання системи облікових записів. Це дозволяє розділити ролі: одні користувачі лише переглядають інформацію, інші можуть її змінювати або доповнювати. Такий підхід є типовим для більшості інформаційних систем і забезпечує кращий контроль над даними [3].

Питання безпеки також враховується ще на етапі проєктування. Django вже має вбудовані механізми захисту, тому є можливість використати їх без додаткової реалізації, що спрощує розробку і зменшує ризик помилок [1].

Висновки та перспективи.

У процесі роботи було визначено основні принципи побудови веб-застосунку електронної бібліотеки та сформовано бачення його структури. Основна увага приділялася простоті використання та можливості подальшого розвитку системи.

Використання Python разом із Django є виправданим, оскільки ці інструменти дозволяють організувати проєкт зрозуміло і без зайвої складності. Крім того, вони дають можливість поступово розширювати функціонал.

Надалі цей проєкт може бути доповнений новими можливостями, наприклад системою оцінювання книг, рекомендаціями або роботою з файлами. Це дозволить перетворити базовий застосунок у більш повноцінну електронну бібліотеку.

Список використаних джерел

1. Django Software Foundation. Офіційна документація Django https://docs.djangoproject.com/en/stable/
2. Гужва В. М. Інформаційні системи і технології на підприємствах. Київ: КНЕУ, 2018. https://retfk.edu.rv.ua/wp-content/uploads/2024/03/Guzhova-V-M-Informaczijni-sistemi.pdf
3. Ian Sommerville. Software Engineering. 10th ed. Pearson, 2015. https://dn790001.ca.archive.org/0/items/bme-vik-konyvek/Software%20Engineering%20-%20Ian%20Sommerville.pdf
4. William S. Vincent. Django for Beginners. Leanpub, 2022. https://elhacker.info/manuales/Lenguajes%20de%20Programacion/Python/Django%20for%20Beginners_%20Build%20Websites%20with%20Django%20(Version%202.1).pdf

Бєляєв Данило Анатолійович,
студент групи ТЦР-41,
спеціальність 121 Інженерія програмного забезпечення,
Державного університету інформаційно-комунікаційних технологій
st7900498@stud.duikt.edu.ua

МІКРОСЕРВІСНА АРХІТЕКТУРА ЛОГІСТИЧНОЇ CRM-СИСТЕМИ ДЛЯ АСИНХРОННОЇ ОБРОБКИ ВИСОКОНАВАНТАЖЕНИХ ЗАДАЧ МАРШРУТИЗАЦІЇ

Постановка задачі. Задачею дослідження є проєктування масштабованої, відмовостійкої програмної архітектури для логістичної CRM-системи. Необхідно забезпечити ефективний розподіл навантаження між I/O-bound компонентами (API-шлюз, управління профілями) та CPU-bound воркерами (алгоритми комбінаторної оптимізації), а також реалізувати безпечний механізм авторизації з розділенням даних профілю користувача та його ідентифікаційної інформації.

Мета дослідження. Розробка та обґрунтування мікросервісної архітектури на базі гібридного технологічного стеку, що дозволяє ізолювати ресурсомісткі обчислення та забезпечити високу пропускну здатність системи управління доставками.

Результати дослідження. Спроєктована система базується на принципах слабкої зв'язності (loose coupling) та високої згуртованості (high cohesion). Архітектура розподілена на три основні домени:

1. *API Gateway & Core Logic:* Реалізовано на базі фреймворку NestJS. Відповідає за валідацію вхідних запитів від клієнтських додатків, агрегацію даних та управління бізнес-сутністями (замовлення, посилки).
2. *Identity Provider (IdP):* Використовується концепція ізоляції ідентифікаційних даних. Система авторизації (наприклад, на базі рішень формату Ory Kratos) зберігає лише облікові дані, тоді як розширені атрибути (ім'я, номер телефону) винесені в окрему сутність User Profile у власній базі даних CRM.
3. *Routing Optimization Service:* Ізольований мікросервіс на мові Python, що інкапсулює логіку Google OR-Tools для вирішення задачі VRPTW.

Оскільки API-шлюз обробляє швидкі I/O-операції, а Python-сервіс виконує тривалі обчислення, класичний синхронний REST-підхід є неефективним. Для інтеграції використано патерн асинхронного обміну повідомленнями (Message Broker).З точки зору теорії масового обслуговування, мікросервіс маршрутизації моделюється як система $M/M/N$, де N - кількість паралельних Python-воркерів. Середній час очікування завдання в черзі брокера (W_q) описується формулою Ерланга:

$$W_q = \frac{P_c}{N\mu - \lambda}$$

де λ - інтенсивність надходження запитів на оптимізацію маршруту, μ - продуктивність одного воркера, а P_c - ймовірність того, що всі N воркерів зайняті.

Завдяки мікросервісній інфраструктурі, показник N може динамічно масштабуватися (горизонтальне масштабування) незалежно від API-шлюзу при зростанні λ (наприклад, у години пік).

Для вибору оптимального механізму міжсервісної взаємодії було проведено порівняльний аналіз архітектурних патернів (Таблиця 1).

Таблиця 1. Порівняльний аналіз патернів міжсервісної взаємодії для задачі оптимізації

Характеристика	**REST API (Синхронний)**	**gRPC (Синхронний/Стримінг)**	**Message Broker (Асинхронний)**
Блокування потоку	Блокує клієнта на час розрахунку	Часткове (при стрімінгу)	Не блокує (Fire-and-forget)
Стійкість до збоїв	Низька (запит втрачається при падінні воркера)	Середня	Висока (повідомлення зберігається в черзі)
Горизонтальне масштабування	Вимагає Load Balancer	Складне балансування з'єднань	Природне (воркери конкурують за чергу)

Обґрунтованим вибором є використання брокерів повідомлень (на кшталт NATS JetStream або Kafka), що дозволяє забезпечити гарантовану доставку параметрів маршрутизації до Python-сервісу та асинхронне повернення результатів ETA (Estimated Time of Arrival) до NestJS для оновлення статусу доставки.

Висновки. Запропонована гібридна мікросервісна архітектура (NestJS + Python) повністю задовольняє вимоги сучасних логістичних систем. Ізоляція обчислювального алгоритму VRPTW в окремий асинхронний сервіс запобігає деградації API-шлюзу під навантаженням. Розділення даних авторизації та користувацьких профілів підвищує гнучкість системи та спрощує дотримання стандартів безпеки даних. Дана архітектура є фундаментом для подальшої інтеграції високонавантажених баз даних реального часу та розширення функціоналу CRM.

Список використаних джерел

1. Newman, S. (2021). Building Microservices: Designing Fine-Grained Systems (2nd ed.). O'Reilly Media.
2. Richardson, C. (2018). Microservices Patterns: With examples in Java. Manning Publications.
3. Hohpe, G., & Woolf, B. (2003). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional.
4. Документація NestJS: Microservices. [Електронний ресурс]. Режим доступу: https://docs.nestjs.com/microservices/basics

Ступак Назар Олександрович,
студент 4 курсу, групи ТЦР-44,
Керівник: Гордієнко Катерина Олександрівна,
старший викладач кафедри Технологій цифрового розвитку,
Державний університет інформаційно-комунікаційних технологій, м. Київ

ДОЦІЛЬНІСТЬ ПРОЕКТУВАННЯ СИСТЕМИ ПРАКТИЧНОГО ЗАСВОЄННЯ МОВИ SQL

Постановка задачі. Реалії розробки інформаційної систем вимагають від спеціалістів високого рівня володіння навичок роботи з реляційними базами даних. Якщо вірити опитуванням, таким як Stack Overflow Developer Survey 2025 чи LinkedIn Emerging Jobs Report, знання SQL постійно залишається однією з найпопулярніших технічних навичок. Це стосується фахівців різних сфер пов'язаних з інформаційними технологіями, наприклад тих що займаються розробкою програм, аналізом даних, бізнес-аналітикою або працюють над серверною частиною. Звичайні способи вивчення SQL – лекції, завдання з книжок чи лабораторні роботи – часто не дають студентам достатньо практики. Студенти можуть добре засвоїти теорію, але в подальшому можуть мати проблеми зі справжніми базами даних, не можуть писати складні запити такі як,наприклад, запити з конструкцією JOIN, агрегатними функціями, підзапитами, CTE чи window functions, або не можуть покращувати швидкість роботи системи.

Наразі є інтерактивні онлайн-платформи на кшталт HackerRank, LeetCode, StrataScratch, LearnSQL.com, SQLZoo, DataCamp, і вони можуть допомагати з практичними навичками. Але вони мають кілька суттєвих мінусів.

1. Пропонуються тільки готові завдання на вже підготовлених базах даних. Викладачі не можуть завантажувати свої набори даних, що унеможливлює адаптацію матеріалу під певну програму навчання.
2. Відсутня можливість автоматично створювати різні завдання для кожного студента. Через це вся група отримує однакові завдання, що заохочує порушення академічної доброчесності.
3. У вищевказаних платформ зазвичай відсутня інтеграція з популярними системами управління навчанням, наприклад, такими як Google Classroom, що є часто використовуваною в українських університетах та інститутах. Це ускладнює роздачу завдань, автоматичну передачу оцінок і відстеження того, як просувається вся група в одному навчальному середовищі.

Вказані мінуси призводять до необхідності викладачам вручну користуватися кількома інструментами, витрачаючи багато часу на підготовку матеріалів, перевірку робіт і організацію всього навчального процесу. Студенти ж, зі свого боку, не мають достатньо індивідуального підходу до навчання і не отримують швидкого зворотного зв'язку. Тож є реальна потреба створити особливу інтерактивну платформу, яка дозволить викладачам працювати зі своїми базами даних, автоматично генерувати різні завдання для кожного студента і буде інтегрована з Google Classroom.

Мета дослідження. Обґрунтування доцільності проектування та розробки спеціалізованої інтерактивної веб-орієнтованої навчальної платформи для практичного засвоєння мови SQL, яка усуває виявлені недоліки існуючих рішень,

забезпечує динамічну генерацію індивідуальних наборів завдань, автоматичну перевірку SQL-запитів на основі завантажених викладачем баз даних та повноцінну інтеграцію з Google Classroom.

Результати дослідження. У ході аналізу встановлено, що жодна з популярних платформ не поєднує в собі всі необхідні для ефективного групового навчання функції. Розроблена в рамках бакалаврської роботи платформа на базі Python та мікрофреймворку Flask реалізує повний цикл навчального процесу. Викладач отримує можливість завантажувати власні SQLite-бази даних, створювати та категоризовувати завдання (тема, підтема, рівень складності), автоматично генерувати індивідуальні набори завдань для кожного студента з урахуванням заданих критеріїв і рандомізацією. Система виконує автоматичну перевірку SQL-запитів студентів шляхом порівняння результатів з еталонними даними в безпечному read-only режимі, зберігає історію всіх спроб і формує детальну статистику прогресу.

Платформа чітко розрізняє, хто викладач, а хто студент, і завдяки зручному інтерфейсу нею легко користуватися обом. Також вона дозволяє відстежувати результати навчання в реальному часі. Такий підхід дозволяє зробити навчання індивідуальним для кожного студента і значно зменшує імовірність списувань. Завдяки цьому система стає не просто способом для самостійної підготовки, а повноцінним навчальним середовищем, яке добре підходить для реальних потреб університетів та інститутів.

Висновки та перспективи. Проектування та розробка спеціалізованої системи практичного засвоєння мови SQL є доцільною, актуальною та перспективною. Створена платформа повністю усуває ключові недоліки існуючих аналогів і може бути ефективно використана у вищих навчальних закладах, на курсах підвищення кваліфікації, в системах дистанційного та змішаного навчання. У майбутньому можлививий розвиток платформи. Наприклад, можлива повна взаємодіяла з Google Classroom: автоматична публікація завданнь і перенос оцінок. Додавання підтримки інших база даних, наприклад, PostgreSQL чи MySQL. Можливе додавання елементів штучного інтелекту, для генераціх умов завдань та підлаштування навчання під користувача. Також можливе розширення платформи для інших мов програмування та технологій.

Список використаних джерел:

1. Stack Overflow Developer Survey 2025 [Електронний ресурс]. – Режим Режим доступу - https://survey.stackoverflow.co/2025/
2. Гринберг М. Flask. Розробка веб-додатків на Python. – К.: Діалектика, 2020.
3. LearnSQL.com. Top 3 Platforms to Learn SQL Online: A Complete Comparison [Електронний ресурс]. – Режим доступу: https://learnsql.com/blog/top-platforms-learn-sql-online-comparison.
4. StrataScratch. A Comprehensive Review of Online Platforms for SQL Practice [Електронний ресурс]. – Режим доступу: https://www.stratascratch.com/blog/a-comprehensive-review-of-online-platforms-for-sql-practice.
5. Семенюк О. Інтелектуальна технологія перетворення природної мови в SQL-запити // Вісник ВНТУ. – 2023. – № 2.

Заволокін Віталій Юрійович
студент 4 курсу, групи КНД-43
Державного університету інформаційно-комунікаційних технологій
(063)-503-93-52
st7526877@stud.duikt.edu.ua
Науковий керівник: Іщеряков Сергій Михайлович,
кандидат технічних наук, доцент,
Державного університету інформаційно-комунікаційних технологій

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ ПОГОДНИХ УМОВ У ВИГЛЯДІ TELEGRAM-БОТА НА ОСНОВІ МОВИ ПРОГРАМУВАННЯ PYTHON

У сучасних умовах користувачам важливо швидко отримувати інформацію про погодні умови. Найзручнішим способом доступу до таких даних стали мобільні додатки та месенджери. Одним із них є Telegram, який дозволяє створювати чат-ботів для автоматичного надання інформації [1]. Використання таких рішень дозволяє значно скоротити час отримання даних та підвищити зручність їх використання у повсякденному житті.

Постановка задачі. Необхідно розробити Telegram-бота, який дозволяє отримувати актуальні дані про погоду у вибраному місті.

Основні функції:

- пошук міста;
- отримання поточної погоди;
- прогноз на кілька днів;
- збереження даних користувача.

Мета дослідження. Метою роботи є розробка Telegram-бота на основі мови програмування Python, який дозволяє отримувати інформацію про погодні умови у зручному форматі, підтримує різні мовні варіанти та зберігає параметри користувача.

Результати дослідження. У процесі виконання роботи було створено Telegram-бота, реалізованого за допомогою мови програмування Python. Для взаємодії з Telegram використано Telegram Bot API та бібліотеку pyTelegramBotAPI.

Система працює за клієнт-серверним принципом. Користувач взаємодіє з ботом через інтерфейс Telegram, а серверна частина виконує обробку запитів, отримує дані з зовнішніх сервісів та формує відповідь. Схему роботи системи представлено на рис. 1.

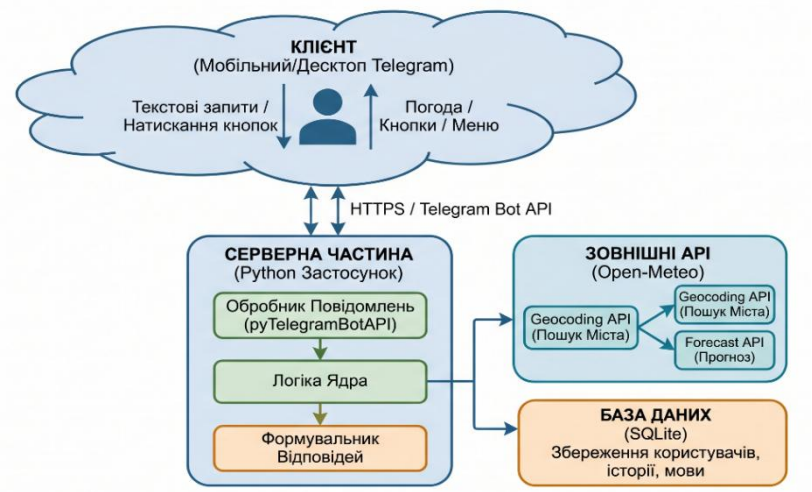


Рис. 1. Схема взаємодії компонентів інформаційної системи

Для визначення координат населених пунктів використано Geocoding API сервісу Open-Meteo. Отримання інформації про погодні умови реалізовано через Forecast API, який надає дані у форматі JSON [3].

У межах розробленого програмного продукту реалізовано такі можливості:

- введення назви міста;
- вибір зі списку;
- відображення поточної погоди;
- прогноз на 3 та 7 днів;
- збереження історії запитів;
- підтримку декількох мов.

Для збереження інформації використано базу даних SQLite, що дозволяє уникнути використання окремого серверу та спрощує реалізацію [4].

Інтерфейс користувача реалізовано за допомогою текстових повідомлень і кнопок, що забезпечує простоту та зручність використання.

Висновки та перспективи. У результаті створено Telegram-бота, який дозволяє швидко отримувати інформацію про погодні умови. Рішення є простим у використанні та не потребує встановлення додаткового програмного забезпечення.

Перспективами подальшого розвитку є:

- погодинний прогноз;
- сповіщення;
- визначення місцезнаходження;
- підключення інших API.

Список використаних джерел:

1. Telegram Bot API [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api (дата звернення: 25.04.2026).

2. Bots: An introduction for developers [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots (дата звернення: 25.04.2026).

3. Open-Meteo API Documentation [Електронний ресурс]. – Режим доступу: https://open-meteo.com/en/docs (дата звернення: 25.04.2026).

4. SQLite Documentation [Електронний ресурс]. – Режим доступу: https://sqlite.org/docs.html (дата звернення: 25.04.2026).

5. Python Official Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/ (дата звернення: 25.04.2026).

Коломієць Артем Григорович
студент 4 курсу, групи ПД-42
Державний університет інформаційно-комунікаційних технологій
(096) 738-37-99
mr.sailan18@gmail.com

АРХІТЕКТУРА ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ

Постановка задачі. Сучасні ІТ-проєкти характеризуються високою динамікою змін вимог, розподіленістю команд, необхідністю швидкої адаптації та інтеграції з різними інструментами (Jira, Azure DevOps, Git тощо). Ефективне управління такими проєктами вимагає потужної інформаційної системи (ІС), яка забезпечує централізований збір, обробку, аналіз та візуалізацію даних про проєкт на всіх етапах його життєвого циклу. Актуальним є проєктування архітектури ІС управління ІТ-проєктами, яка поєднує гнучкість, масштабованість, надійність та підтримку гібридних методологій управління (Waterfall + Agile/Scrum/Kanban).

Мета дослідження. Метою дослідження є аналіз та проєктування архітектури інформаційної системи управління ІТ-проєктами; визначення її ключових компонентів, переваг, недоліків існуючих підходів та перспектив розвитку.

Результати дослідження. Інформаційна система управління ІТ-проєктами (Project Management Information System – PMIS) є комплексним інструментом, який автоматизує процеси планування, виконання, моніторингу та завершення проєктів. Вона забезпечує підтримку всіх знаннєвих областей управління проєктами відповідно до PMBOK та інтегрує гнучкі методології Agile.

Архітектура ІС управління ІТ-проєктами зазвичай включає кілька рівнів:

- Бізнес-архітектуру (процеси управління проєктами);
- Інформаційну (дані) (сховища проєктних даних, документи, метрики);
- Програмну (application) (модулі планування, трекінгу, звітності);
- Технологічну (інфраструктуру, сервіси, інтеграції).

Сучасні архітектури тяжіють до мікросервісної або хмарної (cloud-native) побудови з використанням контейнеризації (Docker, Kubernetes), API-first підходу та сервіс-орієнтованої архітектури (SOA). Це забезпечує високу масштабованість, відмовостійкість та можливість незалежного розгортання компонентів.

До ключових функціональних модулів ІС належать: планування та календарювання (Gantt-чарти, критичний шлях), управління ресурсами та завантаженням команди, трекінг задач і прогресу, управління ризиками, бюджетуванням, документообігом, дашборди та аналітика в реальному часі, а також інструменти колаборації.

Серед традиційних підходів до реалізації — монолітна архітектура, яка проста в розробці, але важко масштабується. Сучасні рішення базуються на мікросервісах, що дозволяють інтегрувати готові інструменти (Jira, Trello, Microsoft Project, GitLab) через REST/GraphQL API. Перспективним є використання штучного інтелекту для прогнозування ризиків, автоматичного розподілу ресурсів та генерації звітів.

Гібридні методи управління IT-проєктами демонструють високу ефективність. Поєднання класичних практик PMBOK з Agile/Scrum/Kanban у межах єдиної ІС дозволяє адаптуватися до специфіки проєкту: жорстке планування для регуляторних частин та ітеративну розробку для основного функціоналу.

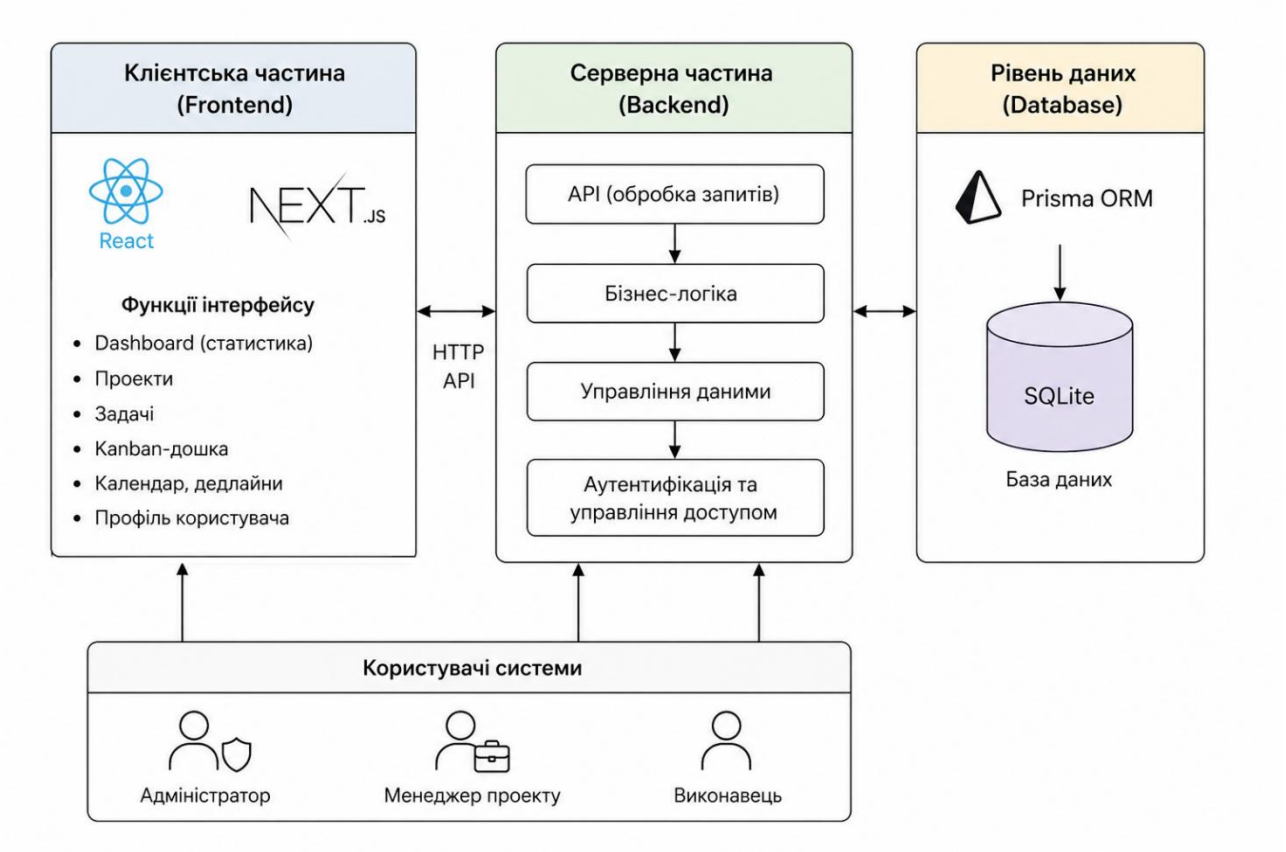


Рис. 1. Архітектура інформаційної системи управління IT-проєктами

Висновки та перспективи. У дослідженні проаналізовано архітектурні підходи до побудови інформаційної системи управління IT-проєктами. Порівняльний аналіз традиційних (монолітних) та сучасних (мікросервісних, хмарних) архітектур виявив переваги останніх у гнучкості, масштабованості та швидкості адаптації до змін.

Гібридні архітектурні рішення, що поєднують переваги мікросервісів, хмарних технологій та інтеграції з інструментами управління проєктами, забезпечують ефективну підтримку як традиційних, так і гнучких методологій.

Перспективою подальших досліджень є розробка та впровадження інтелектуальних компонентів на базі машинного навчання та глибокого навчання з підкріпленням для автоматизації прийняття рішень, а також вивчення архітектурних аспектів забезпечення безпеки та захисту даних в розподілених ІС управління IT-проєктами.

Список використаних джерел

5. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Seventh Edition, 2021.

6. Massimo R., Zenezini G., Ottaviani F.M., De Marco A. Project Management Information Systems: a Systematic Literature Review // Procedia Computer Science. 2025

7. ProjectManager.com. A Complete Guide to PMIS (Project Management Information System). 2025. URL: https://www.projectmanager.com/blog/a-complete-guide-to-pmis

Марущак Артем Русланович,
студент 4-го курсу, групи ПД-41
Державного університету інформаційно-комунікаційних технологій
+380972579023
st7997335@stud.duikt.edu.ua

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ОНЛАЙН-ПРОДАЖУ ПАПЕРОВИХ КНИЖОК

Постановка задачі

Продаж паперових книг через інтернет включає в себе такі етапи, як вибір книги, ознайомлення із її характеристиками, додавання до кошика, оформлення замовлення та відстеження доставки. Такі ж етапи виконуються адміністратором веденням каталогу, відстеження замовлень та наявності книг на складі. Якщо вони виконуються ручним чином або використовуються різні інструменти, то це призводить до появи помилок і витрат часу від користувачів та адміністраторів.

Відсутність ефективних інструментів для виконання цих дій вимагає їх автоматизації, що дозволить зробити процес купівлі більш зручним для споживачів.

Мета дослідження

Одночасно відповідна компанія хоче визначеної мети в дослідженні, які досягнуться в результаті створення застосунку.

Результати дослідження

У загальному, ідеями бути доцільним впровадженням наступних функцій:

- каталогів книг
- пошук книги
- оформлення замовлення
- облік книг на складі

Створення серверної частини доцільно виконати за допомогою програмування на основі Java, оскільки цей мова створення коду дозволить визначати наявність книг на складі, встановлювати статус замовлення та дія з замовленнями від користувачів [1].

Для створення бекенду доцільно використовувати фреймворк Spring Framework та Spring Boot, що дозволять інтегрувати систему досвіду замовлення та ведення каталогу книг [2].

Для зберігання відомостей про книги, замовлення та інші дані доцільно використовувати PostgreSQL бази даних [3].

Для створення клієнтської частини доцільно використовувати бібліотеку React, що дозволить виробити інтуітивісний інтерфейс для каталогу книг, сторінок книг, фільтрації та створення особистого кабінету для кожного користувача [4]. Важливо визначати взаємодію із зовнішніми серверами, що дозволить автоматизувати процес відстеження доставки книг від купувачів.

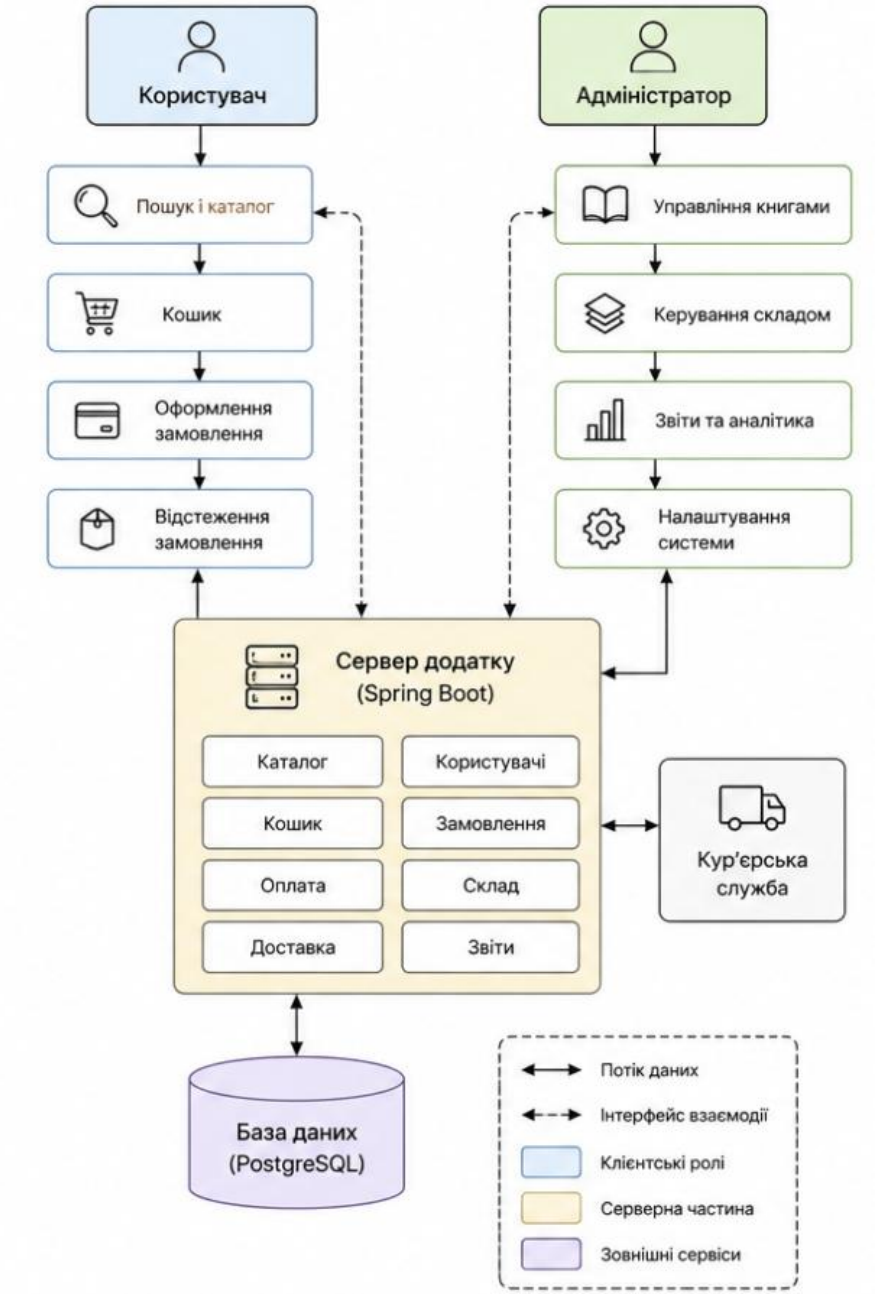


Рисунок 1 - Схема взаємодії компонентів системи онлайн-продажу книг

На діаграмі відображено основні компоненти для системи продажу книг. Користувач здійснює вибір книги, додавання її до кошика та оформлення замовлення. Відновлені дані передаються до серверної частини, де відбиваються процеси відстеження книги, обробки замовлення та відповідного відображення результату дії. Сервер віддійснує взаємодію із базою даних відносно відновленої інформації або із зовнішніми службами доставки. Адміністратор взаємодіє зі всім системою для ведення каталогу та замовлень.

Висновки

Створення застосунку для продажу паперових книг дозвілить досягти мети від компанії-продавця. Відновлення автоматизації процесів дозвілить досягти мети від самого користувача.

З використанням описаних в дослідженні технологій можна забезпечити можливість подальшого розвитку замовлення книг.

Список використаних джерел

1. JDK 26 Documentation - docs.oracle.com. URL: https://docs.oracle.com/en/java/javase/26/ (дата звернення: 30.04.2026).
2. Spring Framework - spring.io. URL: https://spring.io/projects/spring-framework (дата звернення: 30.04.2026).
3. PostgreSQL Documentation - postgresql.org. URL: https://www.postgresql.org/docs/ (дата звернення: 30.04.2026).
4. React - react.dev. URL: https://react.dev/ (дата звернення: 30.04.2026).

Фещенко Богдан Вікторович
здобувач вищої освіти освітнього ступеня бакалавр, групи ТЦР-41
Державний університет інформаційно-комунікаційних технологій
(066) 918-69-82
bogdan.levit2004@gmail.com

ГІБРИДНА АРХІТЕКТУРА МУЗИЧНОЇ ПЛАТФОРМИ НА БАЗІ ПЕРИФЕРІЙНИХ ОБЧИСЛЕНЬ ТА СЕРВЕРЛЕС-ТЕХНОЛОГІЙ

Постановка задачі. Традиційний підхід до розгортання веб-застосунків та соціальних платформ передбачає використання потужних хмарних провайдерів (AWS, Google Cloud). Проте для незалежних спеціалізованих ком'юніті, зокрема музичних спільнот, такий підхід генерує високі інфраструктурні витрати та створює ризики втрати цифрового суверенітету над даними користувачів і авторським контентом. Використання виключно децентралізованих рішень (наприклад, на базі блокчейну) часто призводить до зниження продуктивності під час пошуку по реляційним даним або потокової передачі медіа [1]. Тому актуальним є завдання розробки гібридної архітектури, яка поєднує автономність периферійних обчислень (IoT-вузлів) для хостингу бізнес-логіки із хмарними рішеннями для зберігання важких медіафайлів та баз даних, забезпечуючи при цьому безпечну взаємодію користувачів [2].

Мета дослідження. Метою роботи є розробка та аналіз гібридної програмно-апаратної архітектури спеціалізованої платформи для музичного ком'юніті з використанням IoT-інфраструктури, протоколів захищеної взаємодії WebRTC та розподіленого зберігання даних.

Результати дослідження. Програмна реалізація музичної платформи базується на розподілі обчислювального навантаження між локальним апаратним сервером та спеціалізованими хмарними сервісами. Ядро системи (бекенд на Node.js та фронтенд на React) розгорнуте на мікрокомп’ютері Raspberry Pi 5. Цей пристрій виступає в ролі периферійного вузла (Edge Node), який обробляє логіку управління сесіями, маршрутизацію повідомлень та сигналізацію для відеодзвінків.

Оскільки мікрокомп'ютери мають обмежений ресурс дискової пам'яті та чутливі до інтенсивних операцій запису/читання (I/O), архітектура платформи не є повністю децентралізованою. Для забезпечення ACID-транзакцій та швидкого реляційного пошуку композицій за тегами, база даних винесена у хмарне Serverless-середовище платформи Neon (PostgreSQL). Це рішення дозволяє базі даних динамічно масштабуватися незалежно від апаратних обмежень Raspberry Pi [3]. Усі важкі медіафайли (аудіотреки, PDF-партитури, зображення профілів) зберігаються в об'єктному сховищі Cloudflare R2 Bucket. Клієнтський застосунок отримує доступ до медіафайлів безпосередньо з бакета за допомогою попередньо підписаних URL-адрес (Presigned URLs), що повністю знімає навантаження з мережевого інтерфейсу локального сервера.

Критичним аспектом розгортання локального сервера є забезпечення його доступності та безпеки без використання статичної публічної IP-адреси. Для вирішення цієї задачі імплементовано систему тунелювання трафіку Cloudflare Tunnels. Тунель ініціює вихідне з'єднання від Raspberry Pi до найближчого дата-

центру Cloudflare, що унеможливлює пряме сканування портів зловмисниками та захищає вузол від DDoS-атак [4]. Структуру гібридної маршрутизації та розподілу даних платформи наведено на рис. 1.

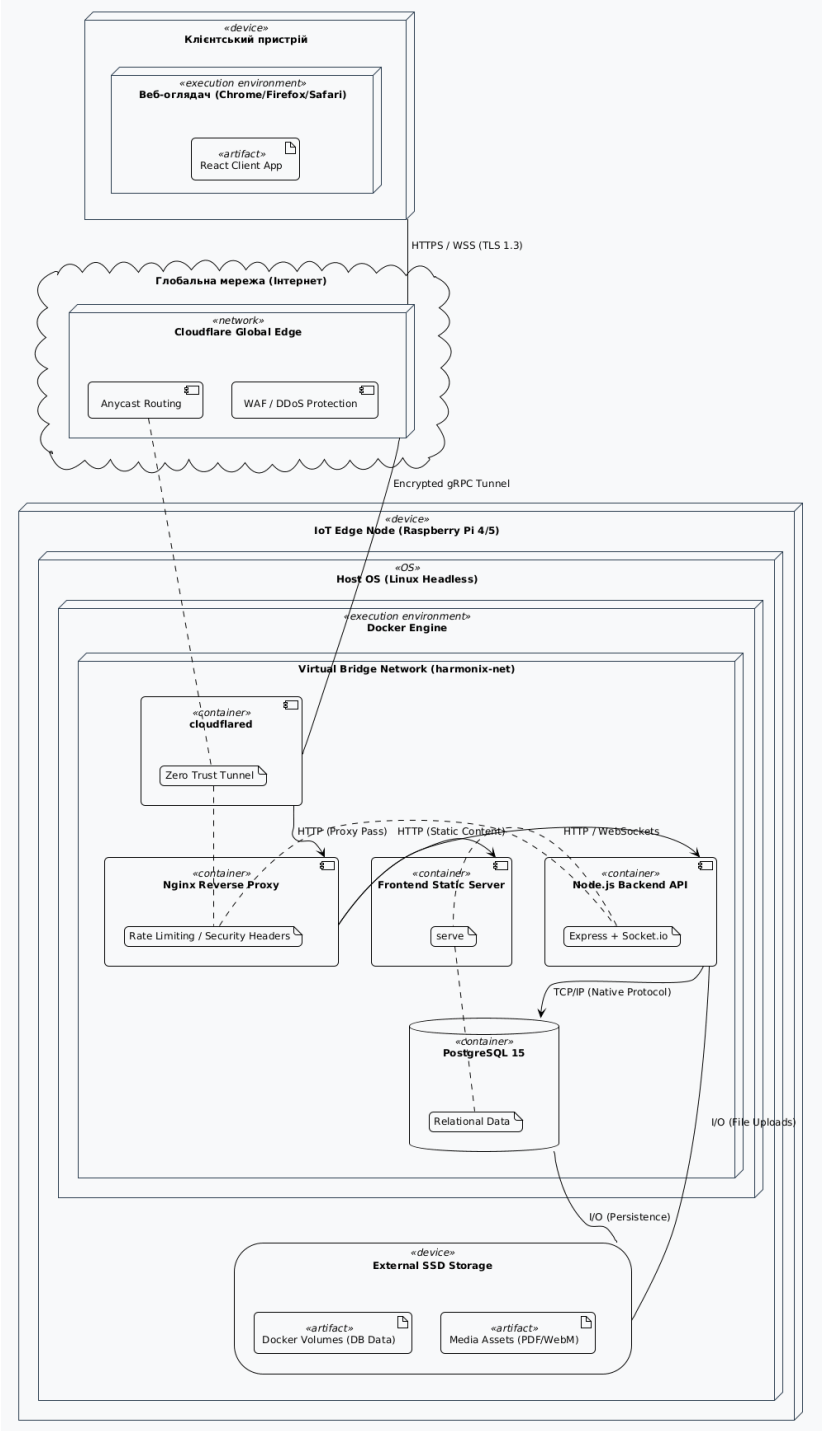


Рисунок - 1. Архітектура гібридної взаємодії IoT-сервера та хмарних сервісів

Підсистема комунікації реалізована з урахуванням вимог до приватності. Для обміну повідомленнями у чатах застосовуються алгоритми наскрізного шифрування (E2EE), а організація групових та індивідуальних відеозустрічей базується на технології WebRTC. Локальний сервер виконує лише функцію сигналізації (Signaling Server) через протокол WebSocket для встановлення сеансу зв'язку. Після успішного обміну SDP-дескрипторами та ICE-кандидатами аудіо- та відеопотоки передаються безпосередньо між браузерами користувачів (Peer-to-Peer), минаючи Raspberry Pi, що мінімізує затримки та зберігає пропускну здатність вузла [5].

Висновки та перспективи. Впровадження гібридної архітектури дозволило створити повноцінну цифрову платформу, яка поєднує економічну ефективність периферійних обчислень із надійністю сучасних хмарних рішень. Делегування ресурсомістких завдань (зберігання файлів та обробка бази даних) платформам Cloudflare R2 та Neon дозволило використовувати IoT-пристрій (Raspberry Pi) як повноцінний сервер додатків для високонавантаженого соціального ком'юніті. Такий підхід забезпечує високий рівень безпеки завдяки тунелюванню трафіку та WebRTC-взаємодії, залишаючись гнучким інструментом для розгортання незалежних програмних систем.

Список використаних джерел

1. Ge Z., Loghin D., Ooi B. C., Ruan P., Wang T. Hybrid Blockchain Database Systems: Design and Performance. Proceedings of the VLDB Endowment. 2020. Vol. 13, No. 12. P. 3138–3151.
2. Sannapureddy R., Nadella V. M., Nelavelli S. Edge–Cloud Continuums for Latency-Sensitive Tasks. International Journal of AI, BigData, Computational and Management Studies. 2024. Vol. 5, No. 4. P. 189-201.
3. Neon Postgres Deep Dive: Why the Updates Change Everything (The Reimagined Foundation: Neon's Disaggregated Architecture). DataFormatHub. 2025.
4. Zahabiuon A., Taylor B., Sowah A. DDoS mitigation from Cloudflare. ITCS-3166. 2023.
5. Raswa R., Sumarudin S., Ismantohadi E. WebRTC Signaling Mechanism Using npRTC Topology for Online Virtual Classroom. Atlantis Press: Advances in Engineering Research. 2021. P. 390–399.

Іванов Дмитро Андрійович,
студент групи ТЦР-44,
Державний університет інформаційно-комунікаційних технологій
(096) 504-27-87
yacode@yacode.dev

WEB-ДОДАТОК ДЛЯ ЗНАЙОМСТВ І СОЦІАЛЬНОЇ ВЗАЄМОДІЇ З ВИКОРИСТАННЯМ GO, TYPESCRIPT ТА MARIADB

Постановка задачі. Сфера цифрових сервісів для знайомств уже давно вийшла за межі простого каталогу анкет. Користувач очікує не лише швидкої реєстрації, а й живого сценарію взаємодії: персоналізованого пошуку, адекватної роботи з геолокацією, миттєвого обміну повідомленнями, керування медіаконтентом і, що не менш важливо, стабільності під навантаженням. Саме тому тема розроблення web-додатка для знайомств на основі Go, TypeScript і MariaDB виглядає не суто навчальною, а цілком прикладною. У такій постановці завдання важить не окрема технологія, а те, як вони поєднуються в єдину систему [1; 2; 3; 4].

Мета дослідження. У досліджуваному програмному продукті логіка предметної області поділена на кілька зрозумілих контурів. По-перше, це класичний сценарій dating, де користувач працює з особистим профілем, уподобаннями, обмеженнями за віком і радіусом пошуку. По-друге, реалізовано party-напрям, у якому існують профілі учасників і окремі профілі подій. Такий поділ видається вдалим, бо система не замикається лише на романтичних знайомствах. Вона, скоріше, підтримує дві моделі соціальної взаємодії: індивідуальний match і об’єднання людей навколо подій, компаній та спільного дозвілля.

Результати дослідження. Клієнтська частина побудована на TypeScript та Next.js. Це рішення виправдане хоча б тому, що для інтерфейсів із великою кількістю станів типізація перестає бути формальністю і стає способом не втратити контроль над даними. У проєкті окремо виділено сторінки профілю, пошуку, вподобань, чатів, завдань, платних можливостей і підтримки. Крім того, у фронтенді використано Redux Toolkit для керування станом, socket.io-client для подій у реальному часі, next-intl для локалізації та Chakra UI для побудови інтерфейсу. На практиці це дає доволі збалансовану комбінацію: інтерфейс не виглядає перевантаженим, але при цьому зберігає керованість і масштабованість [2; 3].

Серверна частина реалізована мовою Go з використанням Gin і GORM. Тут цікаво не лише саме обрання Go, а й характер прикладної логіки. Бекенд оперує сутностями користувача, dating-профілю, party-профілю, події, лайків, чатів, медіафайлів, завдань, скарг, реферальної системи та платних підписок. Іншими словами, застосунок не зводиться до одного CRUD-набору таблиць. Він уже працює як багатокомпонентна доменна система, де є окремі сценарії для метчінгу, модерації, обробки медіа, обробки сирих геоданих, чатів, звітності, монетизації, збору статистики через зашифровані вкладення у headers та багато іншого. Саме це, на мою думку, і відрізняє життєздатний сервіс від студентського макета.

Особливу увагу варто звернути на модель пошуку збігів. У кодовій базі використано просторові запити MariaDB, зокрема обчислення сферичної відстані

між координатами користувачів і подій. Водночас ранжування не є лінійним. Воно враховує вік, межі допустимого віку, радіуси для свого міста та поза ним, стать, історію попередніх лайків і навіть окремий параметр boost. Такий підхід виглядає значно переконливіше, ніж випадкова видача анкет. Щоправда, складніша модель пошуку майже завжди ставить жорсткіші вимоги до якості даних і тестування SQL-запитів, тому просте масштабування без контролю метрик тут було б ризикованим [4].

Не менш показовою є робота з медіа. Система валідує типи файлів, обробляє фото й відео, підтримує компресію, що оптимізована під сумісність та малий розмір файлів (стиснення відео зменшує розмір в ~16 разів в тестах, причому якість знижується маржинально) а окремо пов’язує медіаконтент із різними типами профілів. Для користувача це виглядає природно: додав фото, змінив порядок, відкрив картку, надіслав у чат. Але на технічному рівні за цим стоїть доволі акуратна модель даних і декілька окремих сценаріїв обробки. У реальному сервісі саме такі деталі часто визначають, чи буде продукт сприйматися як надійний.

Висновки та перспективи. Поєднання Go, TypeScript і MariaDB у межах web-додатка для знайомств є цілком обґрунтованим. Go добре підходить для серверної логіки й інтеграційних сервісів, TypeScript спрощує підтримку складного клієнта, а MariaDB забезпечує не лише транзакційне зберігання, а й просторові запити, потрібні для геозалежного матчінгу. Важливо й те, що архітектура продукту підтримує кілька сценаріїв соціальної взаємодії, а не лише базовий swipe-механізм. Саме тому запропоноване рішення можна розглядати як практичну основу для подальшого розвитку повноцінної цифрової платформи знайомств і подій.

Список використаних джерел

1. Go documentation [Електронний ресурс] // The Go Programming Language. URL: https://go.dev/doc/ (дата звернення: 22.04.2026).

2. TypeScript handbook [Електронний ресурс] // TypeScript Documentation. URL: https://www.typescriptlang.org/docs/handbook/intro.html (дата звернення: 22.04.2026).

3. Next.js docs [Електронний ресурс] // Next.js Documentation. URL: https://nextjs.org/docs (дата звернення: 22.04.2026).

4. ST_DISTANCE_SPHERE | MariaDB Documentation [Електронний ресурс] // MariaDB Server Documentation. URL: https://mariadb.com/docs/server/reference/sql-statements/geometry-constructors/geometry-relations/st_distance_sphere (дата звернення: 22.04.2026).

Микитюк Дмитро Сергійович
Студент групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій, м. Київ
+380974142562
st8091720@stud.duikt.edu.ua

ОСОБЛИВОСТІ РОЗРОБКИ SELF-HOSTED РІШЕННЯ ДЛЯ ОРГАНІЗАЦІЇ ТА ПАРСИНГУ ВЕЛИКОГО МАСИВУ МЕДІА ДАНИХ НА ОСНОВІ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

Актуальність дослідження. З кожним роком в кожної зв'язаної інформаційної системи зберігається всё більше та більше даних. Одним із видів таких даних є кількість фотографій, відео, аудіо файлів, скриншотів, картинок і багатьох інших типів файлів. Кількість таких файлів з роками збільшується, а спроби вручну розглядати та переглядати всі ці файли виглядає особливо незручною. Проблема полягає не тільки в тому, щоб зберігати такі файли, а й у тім, щоб шукати потрібні файли. Назви багатьох із них можуть бути різноманітними і не передавати інформації про їх вміст.

Хмарні сервіси для зберігання файлів можуть допомогти вирішити проблему з доступом до даних. Такі сервіси зазвичай мають інтуітивно-зрозумілі інтерфейси, синхронізацію між гаджетами, впорядкування збережених файлів автоматично і можливість для пошуку файлів. Важливо зазначити, що вони дозволяють зберігати файли лише в хмарі, а також вони дозволяють доступ до даних лише випадках, коли ви дозволяєте це віддати хмарному сервісу [1]. Крім того, ви не можете контролювати, як файли обробляються, як вони зберігаються та хто до них має доступ. Проте дуже важливо відшукати і розробити якісь рішення, які працювали б прямо на вашому ком'ютері чи на вашому приватному сервері.

Такі рішення дозволяли б вичерпово керувати самими файлами, які ви хочете зберігати, та використовувати інструменти для їх впорядкування та створення метаданих про файли. Використання штучного інтелекту в такій системі дозволяє автоматично отримати описи до файлів, визначити у них ключові слова, текст від картинок (OCR) чи розшифровки від аудіо файлів. Таким чином, шукати потрібні файли в великих архівах досить буде легко.

Постановка задачі. Одним із основних завдань, які повинні здійснюватися, є створення системи, яка буде працювати прямо в вашому домашньому локальному середовищі та допомагати вам організувати, впорядкувати, обробити та знайти велику кількість ваших медіафайлів, використавши розумну інформацію (AI-метадані). У такій системі можуть бути використані файли, які зберігаються на вашому комп'ютері, так і можливо зробити маленькі картинки-прев'ю від цих файлів.

Також можлива інформація про кожен із файли в створення від штучного інтелекту, що відкриває можливості для створення веб-інтерфейсу, що спрощатиме взаємодію зі всіма вашими медіафайлами. Важливо, щоб така система була можлива для встановлення та запуску. Програми, які працюють на домашніх комп'ютерах, не повинні вимагати від користувачів багато налаштувань, встановлення різних сервісів чи можливості доступу до хмарних даних. Система має бути самостійною, мати свої дані в вашому дому, виглядати

спрощено чи мати просту структуру, якій можна легко додати нові функції. Мета дослідження.

Метою дослідження є спрощення організації великих кількостей медіафайлів, впорядкування їх та вдруге створення інформації про файли штучним інтелектом. Для вирішення мети відбувся аналіз вже існуючих систем, які вирішують задачі створення медіафайлових архівів. Вони оцінені відповідно до того, що в них відчутно добре чи вони мають деякі недоліки. Відбулася розробка самого вимоги до нової системи, її структури та функцій.

Результати дослідження. В дослідженні були розглянуті вже існуючі системи, такі як Google Photos, Immich чи PhotoPrism. Google Photos - це популярний хмарний сервіс, наявний для зберігання фотографій та відео, резервних копій тощо. Проте, він не працює на домашній комп'ютер, а також користувачі не володіють до повного контролю над своїми даними [1]. Програми Immich та PhotoPrism також співпадає зі сподіваними вимогами та можливо встановити на домашній комп'ютер. Проте вони спроектовані в основному для зберігання та впорядкування фотографій та відео [2; 3]. Одною із запропонованих систем є megadrive. Така система може розробитися та виконати сканування картинок, відео, аудіо файлів, створення прев'юшок, зберігання метаданих, описів, ключових слів, тексту від картинок або аудіо файлів.

Серверна частина містить програмування мови Go. Щоб відкрити сервер для взаємодії через інтернет, використовується go-chi/chi. Вони використовуються для створення REST API для взаємодії зі сховищами, файли, пошуком, налаштуваннями, вихідом для користувачів та ключами доступу [4; 5]. Інтерфейс для користувача вироблено на React, TypeScript, Vite та TailwindCSS. Вони вбудовані прямо в сервер від компілятора Go embed. Вони можуть бути встановлені відразу завдяки вбухуванню всіх компонентів в одну програму. Дані до системи зберігати в базі даних SQLite. Вони зберігаються в базі відмітки від локальних папок, файли та їх метадані (AI-метадані), позначки власних, вигляд інтерфейсу, налаштування та ключі доступу.

Щоб відшукати файли за повним текстом, створена була можливість використання SQLite FTS5. Вони використовуються для відшукання файлів за їх метаданими та текстами, що містяться в ютубі відео. Основна таблиця image_fts включає в себе назви файлів, їх описи, текст від картинок, текст від аудіофайлів, ключові слова. Завдяки використанню можливості FTS5 в базі даних можна автоматично шукати файли за їх змістом, без заздалегідь створення окремого пошукового сервера [6].

В основі системи megadrive лежить механізм впорядкування. Ви можете додати нову папку із файлами, що вже існують в системі. Система сама віддіється до встановлення папки, встановить типи від окремих файлів, зібрало технічні дані від кожного файлу, зробить маленькі копії від кожного файлу та зберіть всі ці дані. Такі файли можуть бути віддіється до програми для аналізу від штучного інтелекту. Для відео файлів та аудіо файлів використовуються програми ffmpeg та ffprobe. Також можливо віддієти до зовнішньої команди для розшифровки аудіо файлів, яку ви можете налаштувати самі [9]. На рисунку 1 ви можете переглядати загальну схему роботи системи megadrive для організації та впорядкування великих кількостей медіафайлів.

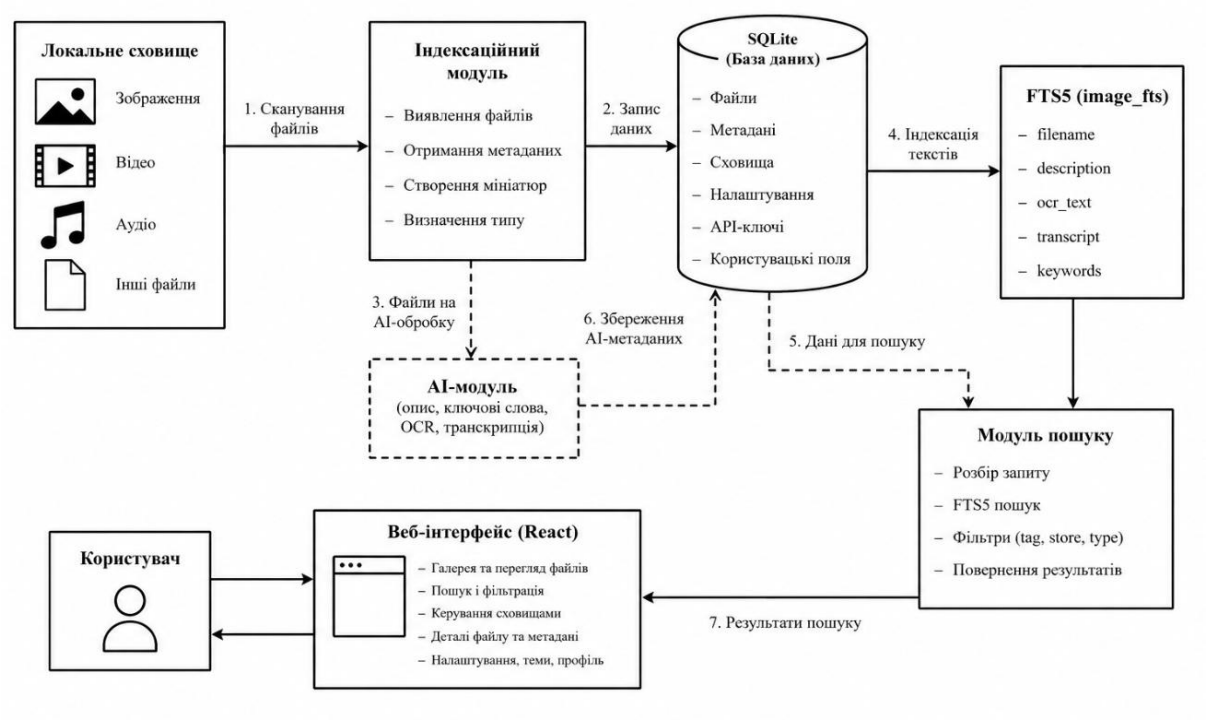


Рис. 1. Загальна схема роботи self-hosted системи megadrive

Система працює так: файли ви не зазвичай завантажуєте в хмари, а вона просто дозволяє вам з'єднання до папкам, які вже існують на вашому комп'ютері. Після впорядкування та створення метаданих від штучного інтелекту файли вже з'являться в галереї. Таким чином, метадані дозволяють вам шукати та фільтрувати файли за їхнім вмістом. Ви можете здійснювати пошук не тільки за назвами файлів а також за описами, текстами, що містяться на фотографіях, а також за тими словам, що вони прозвучали в аудіо файлах. Також важливо забезпечити безпеку та можливість обміну даними.

Для впровадження інформації про файли використовується входи до системи за допомогою паролів, JWT-сесій та API ключів. Вхідні паролі в базі даних зберігаються як bcrypt хеші, JWT-сесії підписані HMAC-SHA256, а API ключі зберігаються у базі даних у вигляді їх SHA-256 хешів [10]. Підсумки. Проблема з організацією великих обсягів медіафайлів, що зберігаються в системі - дуже актуальна. Вони вивчалися та розглядалися вже існуючі рішення, що відповідають за впорядкування медіафайлів - з хмарними сервісами чи самими встановленими на комп'ютери. Одним із рішень що було запропоноване є система megadrive, яка працює в домашньому середовищі. Це досить хороша система, які може бути встановлено в домашній комп'ютер, допомагає створити метадані та впорядкувати великі архіви медіафайлів.

Результатом дослідження стало створення інформаційної системи, що буде працювати прямо на комп'ютері, а також може бути основою для подальшого розвитку програм, що вирішують задачі пошуку відомих файлів.

Список використаних джерел:

1. Google Photos. Edit, Organize, Search, and Backup Your Photos [Електронний ресурс]. – Режим доступу: https://www.google.com/intl/en/photos/about/
2. Immich. Self-hosted photo and video management solution [Електронний ресурс]. – Режим доступу: https://immich.app/

3. PhotoPrism. AI-Powered Photos App for the Decentralized Web [Електронний ресурс]. – Режим доступу: https://www.photoprism.app/
4. Go Documentation [Електронний ресурс]. – Режим доступу: https://go.dev/doc/
5. chi. Lightweight, idiomatic and composable router for building Go HTTP services [Електронний ресурс]. – Режим доступу: https://github.com/go-chi/chi
6. SQLite Documentation. FTS5 Extension [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/fts5.html
7. React Documentation. Describing the UI [Електронний ресурс]. – Режим доступу: https://react.dev/learn/describing-the-ui
8. Vite Documentation. Getting Started [Електронний ресурс]. – Режим доступу: https://vite.dev/guide/
9. FFmpeg Documentation [Електронний ресурс]. – Режим доступу: https://www.ffmpeg.org/documentation.html
10. Provos N., Mazières D. A Future-Adaptable Password Scheme [Електронний ресурс]. – Режим доступу: https://www.openbsd.org/papers/bcrypt-paper.pdf

Сич Богдан Олегович,
студент 4 курсу, групи ПД-44,
Державного університету інформаційно-комунікаційних технологій
+380999501556
st8030907@stud.duikt.edu.ua

РОЗРОБКА СИСТЕМИ АНАЛІЗУ АВТОСЕРВІСУ ДЛЯ ПІДВИЩЕННЯ ЯКОСТІ ОБСЛУГОВУВАННЯ

Постановка задачі

Робота автосервісу включає такі основні етапи: приймання заявок від клієнтів, ведення обліку інформації про транспортні засоби, розподіл робіт між майстрами, ведення обліку виконаного ремонту і відгуку клієнтів щодо його якості. Якщо таку інформацію ведуть окремо, то відсутність можливості її аналізу дозволяє адміністратору сервісу зрозуміти про стан обслуговування.

Наявність саме такої інформації, але її відсутність у вигляді аналізу вимагає створення програмного засобу, що здате відволікати від ведення обліку та вказувати можливості аналізу.

Мета дослідження

Метою дослідження є визначення можливих варіантів реалізації системи аналізу автосервісу.

Результати дослідження

Основними елементами ведення обліку в автомобільному автосервісі є створення нової заявки, зв’язання її із клієнтом і автомобілем, призначення відповідного майстра виконання робіт, ведення обліку виконаних робіт, часу відводиться на відновлення автомобіля та відгуку клієнтів щодо відновлення.

Для відівлення від створення і ведення обліку можна використовувати мову Java, щодо стабільності якої здати відволікати від виконання обліку та присвячувати час аналізу. Крім того, ця мова дозволяє створити структуру відновлення, яка відповідатиме усім вимогам системи [1].

Доцільність використання сервісу Spring Framework і Spring Boot дозволяє створити окремі модулі для клієнтів, транспортних засобів, ремонтних робіт, персоналу і аналітичної частини системи, що значить здати і кожен компонент відволікати від інших [2].

Крім використання бази даних SQLite, доцільність створення серверної частини, що базується на PostgreSQL, дозволяє зберігати відомості про клієнтів, транспортні засобів, здійснені роботи, майстрів, часу відновлення автомобілів та відгуків від клієнтів. Реляційна база даних дозволяє формувати аналітичні відомості щодо середнього часу ремонту, відновлення автомобілів окремими майстрами, частоти відгуків від клієнтів та інші параметри [3].

Клієнтські частини системи можна створити, використовуючи більш відпрацьовану веб-мову – React. Вони можуть включає сторінки для реєстрації нових заявок, відділ автомобілів, історія відновлених автомобілів, панель відновлених робіт, фільтри транспортних засобів та аналітичні розділи для адміністратора. Використання TypeScript дозволяє краще формувати інтерфейси системи [4].

Архітектура системи аналізу повинна включати окремі серверні і клієнтські частини. Сервісні частини повинні відволікати від зберігання інформації,

виконання розрахунків і аналізу, відділ клієнтських може виконувати роботу з відображенням даних і створенням інтерфейсів. Така архітектура дозволяє відволікати серверні частини від клієнтських та навпаки.
Дані в системі можна використовувати для прийняття рішень в автосервісі, вони вказують, наскільки виконуються досить найдовші роботи, які майстри зайняті, де найчастіше виникають проблеми та як відновлюється задоволення від клієнтів.

Для кращого представлення логіки роботи системи доцільно подати її у вигляді структурної діаграми. Вона відображає основні складові програмного засобу: клієнтську частину, серверну частину, базу даних та аналітичний модуль. Користувач взаємодіє із системою через веб-інтерфейс, де може створювати заявки, переглядати інформацію про автомобілі, ремонтні роботи та результати обслуговування. Серверна частина обробляє запити, передає дані до бази даних і формує аналітичні показники, які використовуються адміністратором автосервісу для оцінки якості роботи майстрів, тривалості ремонту та рівня задоволеності клієнтів. Такий підхід дозволяє не лише зберігати інформацію, а й використовувати її для прийняття управлінських рішень.

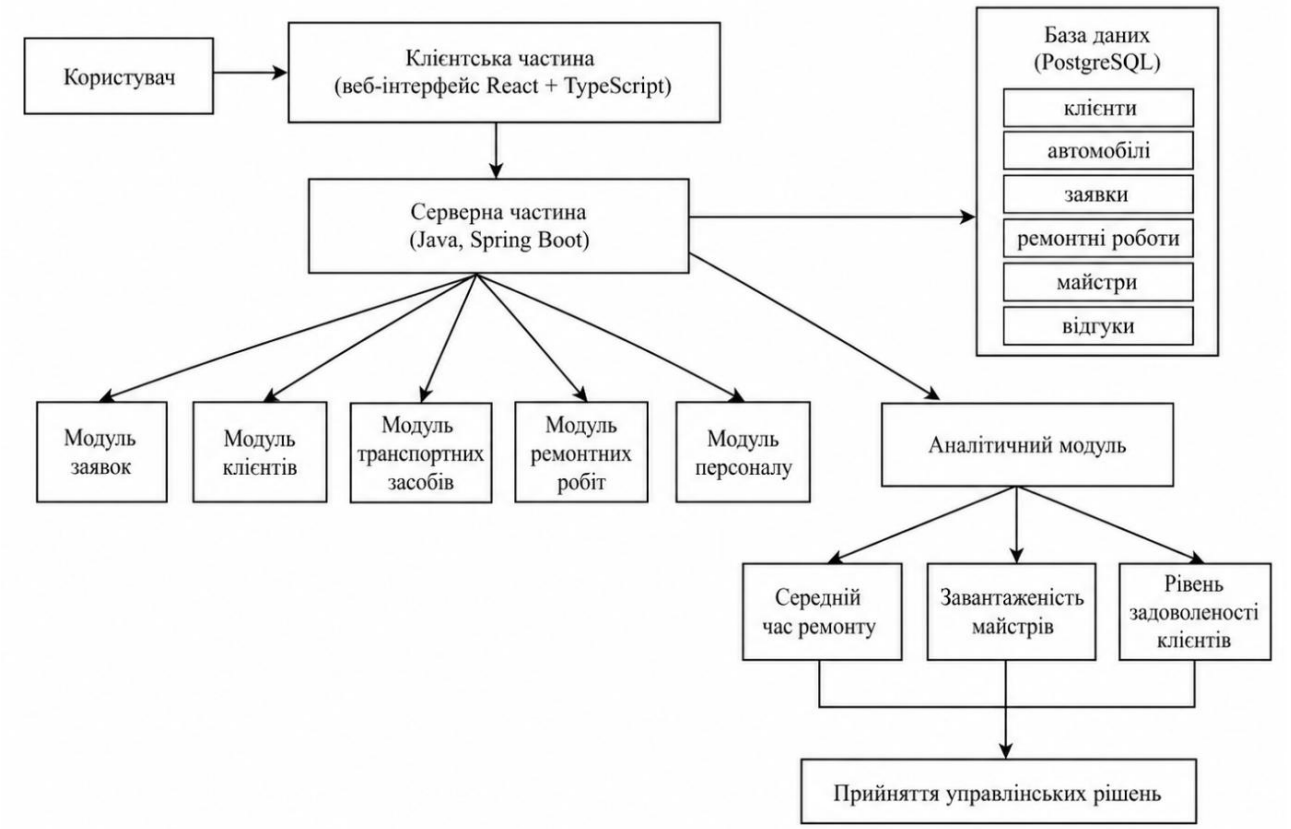


Рис. 1 – Структурна діаграма системи аналізу автосервісу

Висновки та перспективи

Отже, створення системи аналізу автосервісу вводиться до виконання, оскільки вона може відволікати від ведення обліку і вказувати можливості аналізу. Тобто сприяти на зменшення часу ведення обліку, виконання робіт персоналом і вказувати проблеми для вирішення керівництвом сервісу.

Список використаних джерел:

1. JDK 26 Documentation - docs.oracle.com. URL: https://docs.oracle.com/en/java/javase/26/ (дата звернення: 02.05.2026).

2. Spring Framework - spring.io. URL: https://spring.io/projects/spring-framework (дата звернення: 02.05.2026).
3. PostgreSQL Documentation - postgresql.org. URL: https://www.postgresql.org/docs/ (дата звернення: 02.05.2026).
4. React - react.dev. URL: https://react.dev/ (дата звернення: 02.05.2026).

Кіч Роман Олександрович
студент 4 курсу, групи ТЦР-41
Державний університет інформаційно-комунікаційних тухнологій
(099)299-48-84
st7931935@stud.duikt.edu.ua

ЗАБЕЗПЕЧЕННЯ ЦІЛІСНОСТІ ДАНИХ ПІД ЧАС ОБРОБКИ ТРАНЗАКЦІЙ У ВЕБ-СИСТЕМАХ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

Постановка задачі. Розвиток цифрової торгівлі вимагає від бізнесу впровадження максимально надійних програмних рішень. Багато стандартних CMS мають проблеми зі стабільністю роботи під час високих навантажень. Зокрема, виникають ризики втрати даних під час одночасного оформлення покупок різними клієнтами. Тому створення стійкої архітектури, яка гарантує збереження фінансової інформації, є актуальним завданням сучасної програмної інженерії.

Мета дослідження. Дослідження методів забезпечення цілісності даних та практична реалізація відмовостійкого механізму обробки замовлень у розробленій системі управління контентом.

Результати дослідження. Створено спеціалізовану CMS на базі архітектурного патерну MVC. Для збереження інформації використано реляційну базу даних MySQL. Таблиці спроєктовано з використанням підсистеми зберігання InnoDB, яка підтримує транзакції на рівні рядків.

Головна технічна особливість розробленої системи — застосування суворих ACID-транзакцій під час оформлення кошика. Механізм працює за принципом «все або нічого». Запис інформації про нового клієнта, фіксація складу його замовлення та зменшення кількості товару на складі виконуються як одна неподільна операція. Якщо на будь-якому з цих етапів виникає програмна помилка або збій мережі, система миттєво скасовує всі зміни за допомогою команди RollBack. Це повністю виключає появу в базі даних некоректних записів або "осиротілих" замовлень. Крім того, взаємодія з базою даних проходить через безпечний інтерфейс PDO. Підготовлені вирази блокують спроби несанкціонованого доступу та ін'єкцій шкідливого SQL-коду.

Висновки та перспективи. Спроєктована архітектура вебзастосунку дозволяє досягти високої надійності фінансових та складських операцій. На відміну від базових конфігурацій популярних CMS, розроблений транзакційний підхід гарантує стовідсоткову цілісність бази даних інтернет-магазину. У майбутньому планується оптимізувати систему шляхом децентралізації обробки даних, що передбачає винесення логіки сповіщень та аналітики в окремі мікросервіси.

Список використаних джерел

1. PHP Manual. Official Documentation [Електронний ресурс]. – Режим доступу: https://www.php.net/manual/en/
2. MySQL 8.0 Reference Manual. Data Types and Transactions [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/refman/8.0/en/
3. MDN Web Docs: JavaScript and Fetch API [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript

Казанцев Нікіта Євгенійович,
Студент 4-го курсу, групи ПД-43
спеціальність 121 Інженерія програмного забезпечення,
Державного університету інформаційно-комунікаційних технологій
+380 93 559 86 44
st7946340@stud.duikt.edu.ua

ОСОБЛИВОСТІ РОБОТИ ІНТЕРАКТИВНОЇ НАВЧАЛЬНОЇ ПЛАТФОРМИ "SKILLHUB" ДЛЯ ВИВЧЕННЯ ІТ-КУРСІВ

Постановка задачі.

Онлайн-навчання можливе в умовах, коли у студента одночасно є курси, матеріали та змога бачити прогрес. Якщо ресурси розподілені між різними сервісами, це втрачається цілісність процесу вивчення.

Щоб вчити було зручніше, можна створити платформу, де буде каталог курсів з ІТ напрямками, самі курси, можливість тестування та статистика від студента.

Мета дослідження.

Метою дослідження є спрощення процесу організації онлайн-навчання з ІТ-напрямів за рахунок розробки веб-платформи, яка забезпечує централізований доступ до курсів, навчальних матеріалів, тестування та перегляду прогресу користувача.

Результати дослідження.

Використання мови програмування Java для створення серверної частини платформи сприяє можливості створення складних веб-сайтів. У навчальній платформі можна використовувати Java для виробництва об’єктів, що описують курси, модулі, уроки, тестові завдання та результати виконання студентів [1].

Ще одним з варіантів створення бекенду платформи є використання Spring Framework та Spring Boot. Вони здатні створити API для виведення даних про курси, студентів, тестування та прогрес у відпрацьованні виконання виконання студентами тестів [2].

Основним сховищем даних платформи можна використовувати PostgreSQL, яка відділить студентів платформи, вивчені курси, модулі, уроки, тестові питання, спроби виконання тестів та результати від виконання студентами тестів. Таке сховище дозволяє виділяти статистику від виконання студентами тестів, виконання пошуку та створення звітів [3].

Клієнтську частину платформи доцільно реалізувати за допомогою React, оскільки бібліотека добре підходить для створення інтерактивних веб-сторінок. Для створення інтерфейсів платформи варто використовувати TypeScript, що дозволяє використовувати типізацію для структур відображення даних [4].

Доцільним є створення платформи за клієнт-серверною моделлю, де сервер буде відділеним від клієнтської частини, що відділить студентів від сервера платформи.

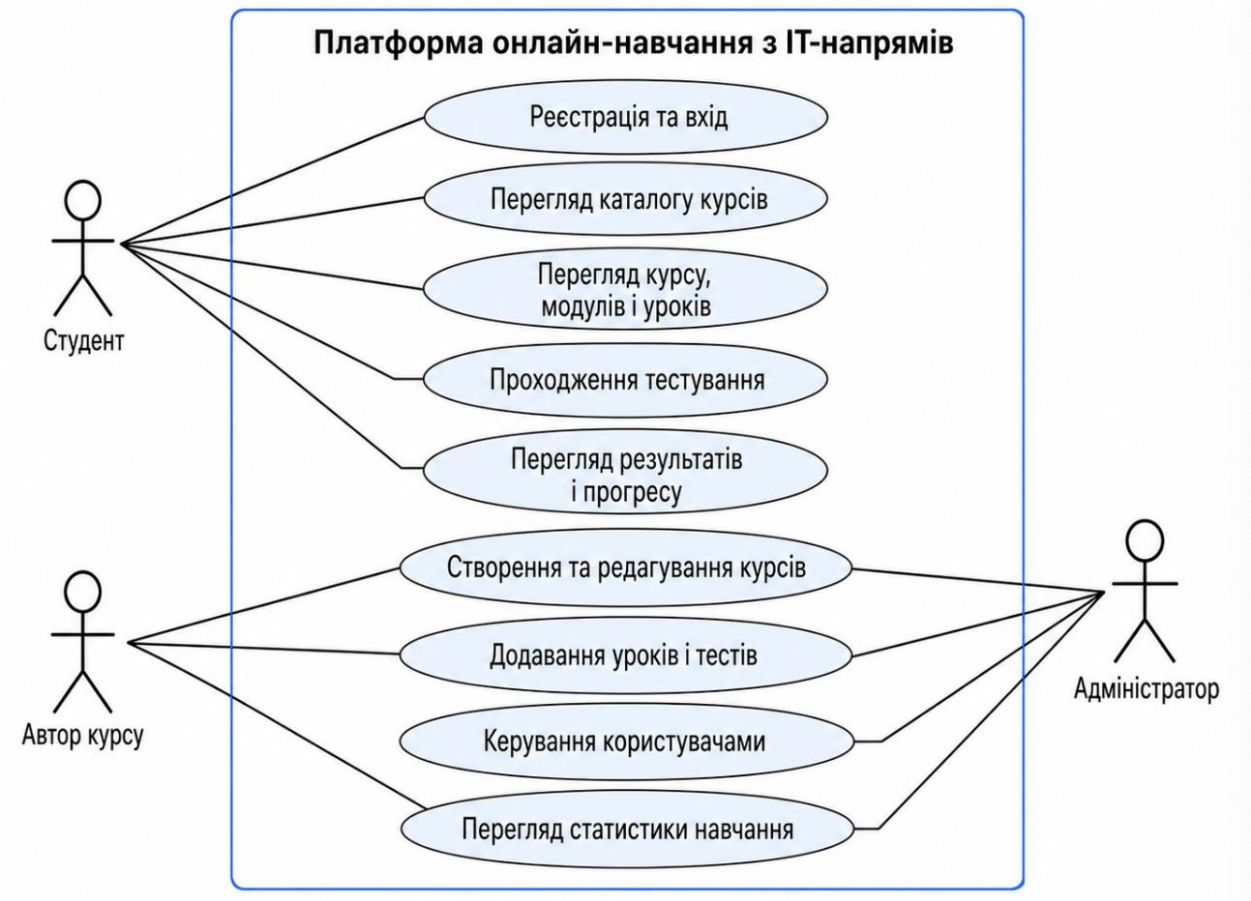


Рис. 2.1 – Діаграма прецедентів платформи онлайн-навчання з IT-напрямів

Висновки та перспективи.

Таким чином, можлива розробка платформи навчання, яка сприяє створенню цифрового простору для вивчення IT курсів. Використання сучасних технологій дозволяє розширити можливості платформи навчання.

Список використаних джерел

1. Benefits and Challenges of E-Learning, Online Education, and Distance Learning [Електронний ресурс] // ResearchGate. – 2024. – Режим доступу: https://www.researchgate.net/publication/379274516_Benefits_and_Challenges_of_E-Learning_Online_Education_and_Distance_Learning

2. JDK 26 Documentation - docs.oracle.com – [Електронний ресурс] – Режим доступу: https://docs.oracle.com/en/java/javase/26/

3. Spring Framework - spring.io – [Електронний ресурс] – Режим доступу: https://spring.io/projects/spring-framework

4. PostgreSQL Documentation - postgresql.org – [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/

5. React - react.dev – [Електронний ресурс] – Режим доступу: https://react.dev/

Сокол Віктор Іванович,
студент 4 курсу, групи ПД-42
Державного університету інформаційно-комунікаційних технологій
(068)-816-73-31
vsokol1450@gmail.com

АКТУАЛЬНІСТЬ СИСТЕМИ ПІДТРИМКИ КЛІЄНТІВ ІНТЕРНЕТ-МАГАЗИНУ ЕЛЕКТРОНІКИ

Постановка задачі.

Розвиток електронної комерції сприяє збільшенню кількості інтернет-магазинів, які працюють з великим обсягом замовлень, клієнтських звернень та післяпродажного обслуговування. Особливо це стосується інтернет-магазинів електроніки, оскільки покупці часто потребують консультацій щодо характеристик товарів, сумісності пристроїв, умов доставки, оплати, гарантійного обслуговування або повернення продукції. У навчальних і наукових джерелах електронна комерція розглядається як важлива складова сучасного електронного бізнесу, що охоплює процеси продажу, взаємодії з клієнтами та інформаційного супроводу торговельної діяльності [1; 2].

Якщо звернення клієнтів обробляються вручну або через різні неузгоджені канали комунікації, виникають проблеми з пошуком історії повідомлень, контролем статусів, швидкістю відповіді та розподілом навантаження між працівниками. У результаті оператори витрачають більше часу на типові відповіді, а клієнти не завжди мають можливість швидко зрозуміти, на якому етапі перебуває їхній запит. Для вирішення цієї проблеми доцільно створити автоматизовану систему підтримки клієнтів, яка забезпечує централізоване опрацювання звернень, категоризацію запитів, контроль статусів, збереження історії взаємодії, використання шаблонів відповідей і довідкових матеріалів.

Мета дослідження.

Метою дослідження є визначення актуальності, функціональних можливостей, засобів реалізації та логічної архітектури автоматизованої системи підтримки клієнтів інтернет-магазину електроніки.

Результати дослідження.

Автоматизована система підтримки клієнтів інтернет-магазину електроніки повинна забезпечувати повний цикл роботи зі зверненням: створення запиту клієнтом, його збереження в системі, призначення категорії, зміну статусу, опрацювання оператором, надсилання відповіді та формування звітності. Такий підхід дозволяє перейти від розрізненого ручного опрацювання повідомлень до керованого інформаційного процесу, у якому кожне звернення має відповідального виконавця, статус і збережену історію взаємодії.

У межах системи доцільно передбачити декілька основних ролей користувачів: клієнт, оператор підтримки та адміністратор. Клієнт створює звернення, переглядає його статус і користується базою знань. Оператор опрацьовує звернення, відповідає клієнтам, змінює статуси та використовує шаблони відповідей. Адміністратор керує користувачами, категоріями, статусами, довідковими матеріалами та звітністю. Така структура відповідає загальним принципам CRM- і helpdesk-підходів, де важливими є централізоване збереження

інформації про клієнтів, контроль комунікації та підвищення якості обслуговування [3; 4].

Для реалізації серверної частини системи доцільно використати мову програмування Java. Вона підходить для створення серверної логіки, роботи з об’єктами предметної галузі, обробки звернень, статусів, повідомлень і правил доступу. У системі підтримки клієнтів Java може використовуватися для реалізації моделей користувачів, звернень, категорій, повідомлень, шаблонів відповідей, довідкових статей та звітів [5].

Framework Spring Boot доцільно застосувати для створення backend-рівня та REST API. За допомогою Spring Boot можна реалізувати модулі авторизації, користувачів, звернень, категорій, статусів, шаблонів відповідей, бази знань і звітності. REST API забезпечує взаємодію між клієнтською та серверною частинами системи, завдяки чому клієнтський інтерфейс може швидко отримувати актуальні дані про звернення, повідомлення, статуси й довідкові матеріали [6].

Для зберігання даних доцільно використати PostgreSQL. У базі даних можуть зберігатися користувачі, ролі, звернення, категорії, статуси, повідомлення, шаблони відповідей, довідкові матеріали та звітна інформація. Реляційна структура дозволяє організувати зв’язки між клієнтами, зверненнями, операторами та повідомленнями, а також виконувати пошук, фільтрацію і формування звітів за визначеними параметрами [7].

Клієнтську частину системи доцільно реалізувати з використанням React та TypeScript. React дозволяє створити компонентний web-інтерфейс для клієнта, оператора й адміністратора. У такому інтерфейсі можна реалізувати форму створення звернення, список звернень, сторінку перегляду діалогу, зміну статусу, роботу з шаблонами відповідей, базу знань і панель звітності. TypeScript підвищує надійність клієнтського коду завдяки статичній типізації та зменшує ризик помилок при роботі з даними звернень і повідомлень [8].

Логічна архітектура автоматизованої системи підтримки клієнтів інтернет-магазину електроніки наведена на рисунку 1.

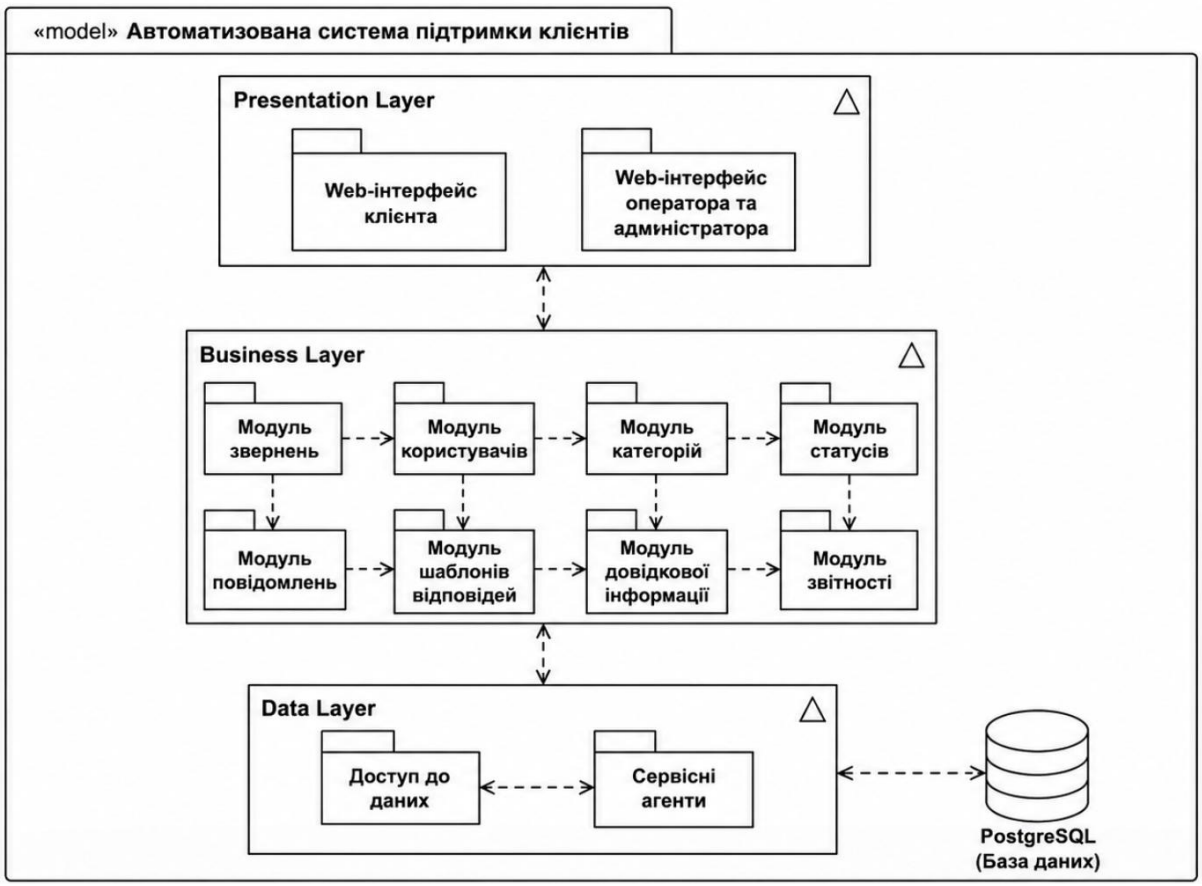


Рис. 1 – Логічна архітектура автоматизованої системи підтримки клієнтів інтернет-магазину електроніки

На рисунку 1 показано поділ системи на три основні рівні: presentation layer, business layer та data layer. Presentation layer включає web-інтерфейс клієнта та web-інтерфейс оператора або адміністратора. Business layer містить основні функціональні модулі системи: модуль звернень, користувачів, категорій, статусів, повідомлень, шаблонів відповідей, довідкової інформації та звітності. Data layer відповідає за доступ до даних, сервісні агенти та взаємодію з базою даних PostgreSQL. Така архітектура дозволяє розділити відображення інтерфейсу, бізнес-логіку та збереження інформації, що спрощує супровід і подальше розширення системи.

Висновки та перспективи.

Таким чином, створення автоматизованої системи підтримки клієнтів інтернет-магазину електроніки є актуальним завданням, оскільки така система дозволяє централізувати роботу зі зверненнями, зменшити навантаження на операторів, скоротити час відповіді клієнтам і підвищити якість сервісу. Використання клієнт-серверної архітектури, Java, Spring Boot, PostgreSQL, React і TypeScript забезпечує технічну основу для створення надійного web-застосунку. У подальшому систему можна розширити інтеграцією з CRM, онлайн-чатом, Telegram-ботом, email-сервісами або AI-модулем для автоматичного формування відповідей на типові звернення.

Список використаних джерел

1. Електронна комерція та Інтернет-торгівля : навчально-методичний посібник [Електронний ресурс]. – Київ : Аграр Медіа Груп, 2021. – 454 с. – Режим доступу: https://duikt.edu.ua/uploads/l_2231_17488174.pdf

2. Електронна комерція. Інтернет-магазин: його розробка та впровадження [Електронний ресурс] // Вісник Хмельницького національного університету. – Режим доступу: https://journals.khnu.km.ua/vestnik/?p=12483
3. Spring Framework Documentation [Електронний ресурс] // Spring. – Режим доступу: https://docs.spring.io/spring-framework/reference/
4. PostgreSQL Documentation [Електронний ресурс] // PostgreSQL Global Development Group. – Режим доступу: https://www.postgresql.org/docs/
5. React Documentation [Електронний ресурс] // React. – Режим доступу: https://react.dev/

Мартиновченко Максим Віталійович
студент 4 курсу, групи ПД-43
Державного університету інформаційно-комунікаційних технологій
invetero666@gmail.com

ПЕРСПЕКТИВИ РЕАЛІЗАЦІЇ WEB-ПЛАТФОРМИ LOGITRACK ДЛЯ МОНІТОРИНГУ ЛОГІСТИКИ ТА ОЦІНЮВАННЯ РИЗИКІВ ДОСТАВКИ

Постановка задачі

Велика залежність ефективності від доставки від стану інформованості про процес доставки. Коли дана інформація розосереджена по різним таблицям, відомостям та внутрішнім спискам компанії для логістик, відділ логістики не може ефективно вирішувати виникнення або контролювати процес виконання доставок.

Необхідно відстежувати виникнення затримок, зміни статусу доставок та перевантаження маршрутів. У відсутності системи для виведення всієї інформації про доставки і перевезення до керівництва відсутній інструмент для виведення відомостей про виконання доставок, час виконання доставок, скільки маршрутів перевантажено. Відсутність такого системного використання зумовлює виникнення проблем у логістичному обслуговуванні.

Доцільно будувати систему для відстеження всіх логістичних процесів. Вона може виконувати різні функції і мати операції виводу інформації про перевезення, виведення статусів виконання доставок, контролю доставок, аналізу ефективності доставок.

Мета дослідження.

Одним із основних завдань і вимог до системи є визначення її основних функціональних можливостей, технічних засобів та вимог від доцільної її побудови.

Результати дослідження.

В основі системи слід встановити роботу з замовленнями, маршрутами, транспортними засобами, складами, статусами виконання доставок. Система має сприяти виведенню не лише потрібної інформації, але й дозволити аналіз виконання доставок за певними показниками.

Для забезпечення серверної частини доцільність використання мови Java. Вона має структуровані можливості для роботи з великою кількістю даних, виводу статусів, маршруту, виведення аналітика [1].

Для побудови бекенду використовувати платформу Spring Framework та Spring Boot. Ці технології дозволяють створити окремі модулі для виведення і ведення замовлення, склад, транспорту, маршрутів, звітності. Вони також сприяють спрощенню процесу введення нових модулів [2].

Варто використовувати базу даних PostgreSQL. У базу виведено відомості про доставки, маршрутів, транспорті, часи, склади, працівники, статуси виконання доставок. Така база дозволяє здійснювати вибірки, аналіз, виведення звітів щодо логістичних процесів [3].

Клієнтську частину доцільність вигідно виконувати у випадку використання React. У разі використання TypeScript доцільність її використовувати для спрощення введення великих логістичних даних та зниження виникнення помилок [4].

Одна з переваг системи - можливість швидкого виведення узагальненої інформації про процеси та результати виконання доставок. Відділам логістики можна вчасно виявляти проблеми в процесі виконання доставок та приймати відповідні рішення щодо їх вирішення.



Рисунок 1 - Функціональна схема системи моніторингу логістики

Діаграма відображає основні елементи системи моніторингу логістики, вхідні та виводні даних, зв'язки між їх відділами. На діаграмі відображається джерела даних (замовлення, склади, транспорт), модулі системи (доставки, маршрути, статуси виконання, аналітика, звітність), серверна частина та база даних.

Висновки та перспективи

Створення системи моніторингу логістики дає можливість зізвісти компанії про доставку та відділ логістики - мати централізовану інформацію про виконання доставок, виводити статуси виконання доставок, аналізувати їх ефективність, а також вирішувати виникнення у відділі відповідних рішень. Використання системи сприятить вирішенню проблем, сприйманнюй логістичний процес та вирішенню виникнених проблем.

Використання Java, Spring Boot, PostgreSQL, React є відповідним вибором для створення системи моніторингу логістики. Такі технології можуть бути доповнені додатками, що сприяють аналізу ефективності доставок, виведенню сповіщень про виникнення проблем у відділі та прогнозуванні виконання доставок.

Список використаних джерел

1. Михаліцька Н. Я., Верескля М. Р. Логістичний менеджмент : навчальний посібник. Львів : Львівський державний університет внутрішніх справ, 2020. 440 с. ISBN 978-617-511-319-6.

2. JDK 26 Documentation - docs.oracle.com – [Електронний ресурс] – Режим доступу: https://docs.oracle.com/en/java/javase/26/
3. Spring Framework - spring.io – [Електронний ресурс] – Режим доступу: https://spring.io/projects/spring-framework
4. PostgreSQL Documentation - postgresql.org – [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/
5. React - react.dev – [Електронний ресурс] – Режим доступу: https://react.dev/

Замай Олександр Олександрович,
студент 4-го курсу, групи ПД-41
Державного університету інформаційно-комунікаційних технологій
st7936246@stud.duikt.edu.ua

ОСОБЛИВОСТІ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ОНЛАЙН-ПРОДАЖУ КВИТКІВ НА БЛАГОДІЙНІ КОНЦЕРТИ

Постановка задачі

Для того, щоб благодійні концерти стали для глядачів досвідом, який стосується до істинної благодійності, досить відновити присутність тих інформаційних каналів, які здатні швидко повідомити про них всіх глядачів. Якщо квитки будуть купуватися різними способами, то це може призвести до плутанини в замовленнях.

Налоговість, що відноситься до благодійних акцій, вимагає особливої уваги до точності відстеження кількості випускутих квитків. Весь процес має бути зручним для участі у благодійних акціях. Для вирішення поставленних перед благодійними концертами завдань слід створити єдину систему, яка забезпечить зручність та точність обліку всіх вище вказаних квитків [1].

Мета дослідження

У цілому метою дослідження є визначення можливостей та засобів створення системи онлайн-продажу квитків для благодійних концертів.

Результати дослідження

У системі має бути здійснена робота з подіями, а саме: створення концертів, визначення майданчиків, типів квитків та кількості місць. Робота зі замовленнями також важлива: слід використовувати певні механізми, щоб виключити подвійне видачку квитків.

Актуальність створення таких систем викликає використання програмування на Java: ця мова програмування найкраще відповідає потребам системи обміну квитками [2].

Особливо високий рівень автоматизації системи дозволяє використання вже відомих розробок Spring Framework та Spring Boot, які дозволяють розділити роботу на окремі частини: події, квитки, замовлення, користувачі [3].

Для зберігання усієї інформації доцільно використовувати PostgreSQL базу даних, в якій зберігатимуться всі необхідні дані про благодійні концерти [4].

Використання React для створення клієнтської частини системи доцільне, оскільки за допомогою цієї технології можна створити більш зручний для спостереження та вибору місць інтерфейс [5].

Одним із основних вимог до створення системи є її зручність для людей. Інтерфейс системи має бути найпростішим: користувачам треба бути легко вибрати подію, переглянути її вміст, а потім вибрати та підтвердити вирішення замовлення. Адміністратору також треба бути доцільно мати доступ до статистики і відстежувати продажів.

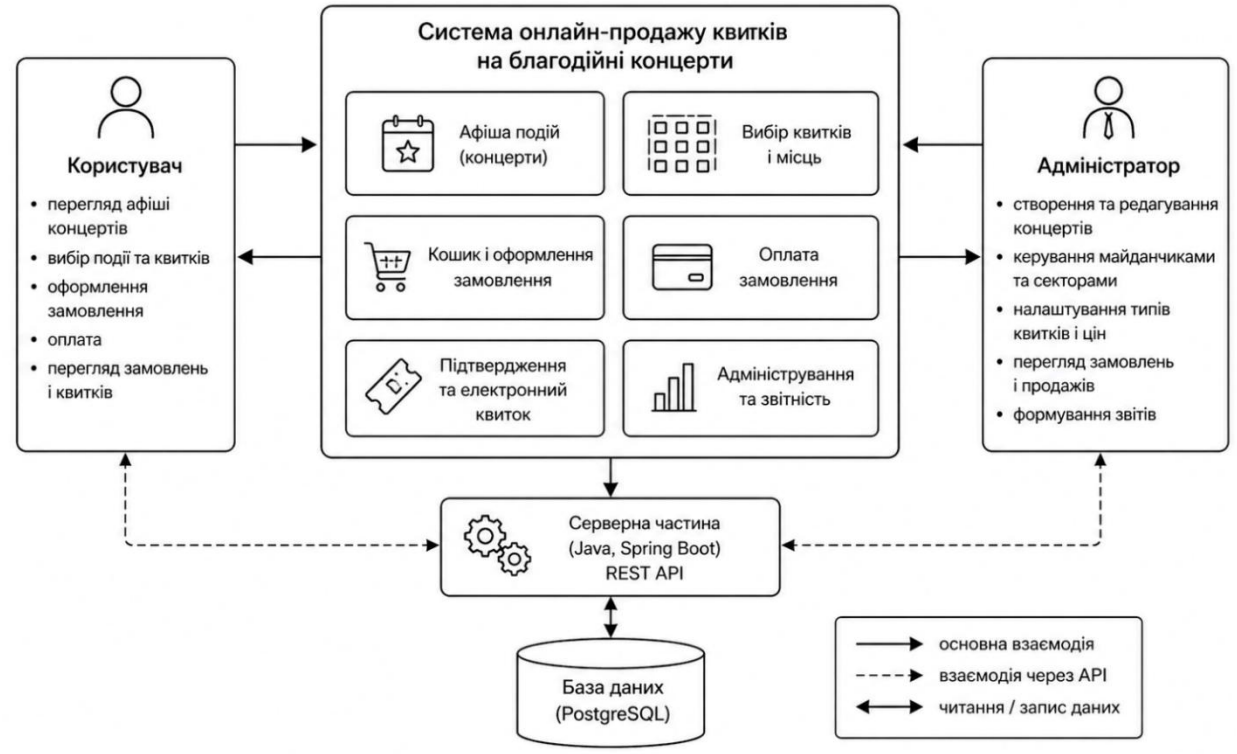


Рис. 1 - Блок-схема системи онлайн-продажу квитків на благодійні концерти

Висновки та перспективи

Здійснення системи онлайн-продажу квитків може бути доцільним вирішенням відомих труднощів благодійних концертів. Це дає можливість організаторам концертів впорядкувати процес організації та зробити вступ до благодійних акцій більш зручним для глядачів.

Крім вирішення випущених проблем, можливість використання сучасних технологій дозволяє доцільному створення системи, яка вирішує досі відкриті випадки. Такий сервіс можна розширити в майбутньому відкритими можливостями.

Список використаних джерел

1. GiveWheel. Charity Event Ticketing Platform: Fundraising Events & QR Code Check-In [Електронний ресурс]. – Режим доступу: https://www.givewheel.com/fundraising/ticketed/

2. JDK 26 Documentation - docs.oracle.com – [Електронний ресурс] – Режим доступу: https://docs.oracle.com/en/java/javase/26/

3. Spring Framework - spring.io – [Електронний ресурс] – Режим доступу: https://spring.io/projects/spring-framework

4. PostgreSQL Documentation - postgresql.org – [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/

5. React - react.dev – [Електронний ресурс] – Режим доступу: https://react.dev/

Реул Нікіта Євгенович
Студент 4 курсу, групи ТЦР-43
Державний університет інформаційно-комунікаційних технологій
nikitareul67@gmail.com

АРХІТЕКТУРА СИСТЕМИ АВТЕНТИФІКАЦІЇ НА ОСНОВІ WEBAUTHN ТА PASSKEYS

Постановка задачі. В умовах наших реалій неможливо уявити життя без використанні веб сервісів, більшість з яких потребує автентифікації користувача, що в свою чергу призводить до збільшення ризиків потенційних кібератак, пов'язаних із компрометацією паролів, фішингом та доступом до облікових записів третіми особами. Звичні всі нам парольні системи автентифікації вже починають втрачати ефективність у сучасних архітектурах веб застосунків. Саме тому дослідження доступу на основі WebAuthn та passkeys, які використовують криптографічні методи замість класичних паролів є актуальними як ніколи.

Мета дослідження. Аналіз архітектури системи автентифікації на основі WebAuthn, визначення принципів взаємодії її компонентів та оцінка переваг використання passkeys у сучасних веб-застосунках.

Результати дослідження. Сучасні веб-застосунки будуються на основі клієнт-серверної архітектури, де браузер виступає в ролі середовища виконання логіки користувача, а інтерфейс реалізується за допомогою JavaScript, саме він забезпечує взаємодію з API браузера, обробку подій та комунікацію із серверною частиною системи [1].

Архітектура WebAuthn-системи складається з чотирьох основних компонентів: користувача, frontend-застосунку, backend-сервера та автентифікатора (апаратного або програмного). Взаємодіють вони між собою через WebAuthn API, яке виступає проміжним рівнем між браузером і криптографічним модулем безпеки. Така структура дозволяє виконувати підпис без передачі приватних ключів через мережу, що підвищує рівень захисту системи.

Під час реєстрації користувача система створює криптографічну пару ключів: приватний ключ - залишається на пристрої користувача, в автентифікаторі, а публічний ключ передається на сервер та зберігається у базі даних. При наступній автентифікації сервер формує випадковий challenge, який підписується приватним ключем на стороні користувача. Сервер перевіряє підпис за допомогою публічного ключа, що забезпечує стійкість до фішингових атак, та збереження доступу до облікових даних лише на стороні користувача [2].

Серверна частина системи реалізується на основі модульної архітектури, яка складається з : автентифікації, бізнес-логіки та доступу до даних. Всі ці пункти підлягають принципам сучасної програмної інженерії та дозволяють забезпечити масштабованість, тестованість і легкість роботи з системою [3].

Важливою частиною сучасних рішень є технологія passkeys, яка забезпечує синхронізацію криптографічних ключів між пристроями користувача через захищені хмарні сервіси. Це дозволяє відмовитися від паролів, спрощує процес входу та робить зручнішим використання системи без зниження рівня безпеки.

Саме так архітектура WebAuthn об’єднує клієнтську логіку, серверну обробку та криптографічний захист в одну систему.

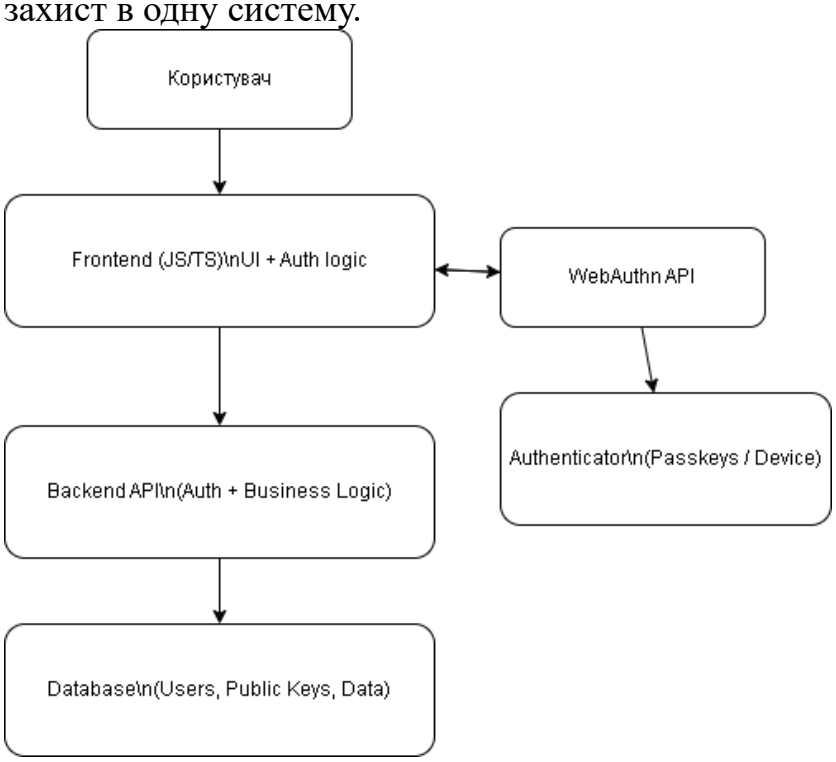


Рис.1 Архітектура системи безпарольної автентифікації

Висновки. Було розглянуто архітектуру системи автентифікації на основі WebAuthn. Показано, що використання криптографії відкритих ключів дозволяє відмовитися від паролів та підвищити рівень безпеки веб-застосунків. Запропонована архітектура є сучасною, та придатною для використання у веб-системах.

Список використаних джерел :

1. MDN Web Docs. JavaScript Guide [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 09.04.2026).
2. FIDO Alliance. Passkeys [Електронний ресурс]. – Режим доступу: https://fidoalliance.org/passkeys/ (дата звернення: 09.04.2026).
3. W3C. Web Authentication: An API for accessing Public Key Credentials [Електронний ресурс]. – Режим доступу: https://www.w3.org/TR/webauthn-3/ (дата звернення: 09.04.2026).

Коломієць Назарій Ярославович
Студент 4 курсу, групи ТЦР-41
Державний університет інформаційно-комунікаційних технологій
(097)687-76-29
nazar.kolomiets180104@gmail.com

СПРОЩЕННЯ ВЗАЄМОДІЇ ТА ОПТИМІЗАЦІЯ ОБРОБКИ ДАНИХ У ВЕБ-ПЛАТФОРМІ ОРЕНДИ ЖИТЛА НА ОСНОВІ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

Сучасні веб-ресурси для пошуку та оренди житла часто перевантажені складними інтерфейсами, вимагають від користувача виконання багатьох ручних дій для фільтрації та не надають інструментів швидкої взаємодії в реальному часі. Щоб дізнатися просту річ — чи вільна квартира, або зв'язатися з її власником — клієнту доводиться очікувати на завантаження монолітних сторінок, вручную оновлювати інтерфейс або використовувати сторонні месенджери. Крім того, традиційні платформи мають затримки при передачі повідомлень та обробці медіа-файлів високої роздільної здатності. У повсякденному житті це створює незручності та уповільнює процес укладання угод. Саме тому виникла ідея розробити веб-платформу, яка автоматизує рутинні процеси, агрегує сирі дані розподілених сервісів і забезпечує миттєву комунікацію «коротко і одразу».

Постановка задачі. Потрібно розробити веб-платформу оренди житла, яка самостійно взаємодіє з ізольованими мікросервісами, трансформує технічні структури даних у зрозумілий для користувача інтерфейс, миттєво оновлює статуси доступності об'єктів та забезпечує автоматичні браузерні сповіщення під час обміну повідомленнями.

Мета дослідження. Створити зручний, швидкий та інтуїтивно зрозумілий спосіб взаємодії користувача з базою нерухомості через веб-додаток. Конкретно: оптимізувати обробку і відображення даних оголошень, налаштувати систему гнучкої клієнтської фільтрації за містами і вартістю, та реалізувати інтерактивний онлайн-чат, який працює без фонового опитування (polling) бекенду.

Результати дослідження. Веб-платформу «Домівка» реалізовано з використанням технологічного стеку Node.js, Nest.js на серверній стороні та PostgreSQL як реляційної СУБД. Програмний комплекс складається з єдиного шлюзу API Gateway та двох ізольованих мікросервісів (user-service та property-service). Система регулярно обробляє запити й формує коротку, зрозумілу відповідь на фронтенді, трансформуючи сирі дані бази у візуальні індикатори (наприклад, блокування кнопки бронювання та виведення статусу «ОРЕНДОВАНО» для зайнятих об'єктів).

Окремо вдалося спростити та прискорити відображення графічного контенту. Замість залучення складних і повільних сторонніх хмарних сховищ, у коді реалізовано механізм конвертації зображень у рядок Base64 безпосередньо на стороні клієнта. Бекенд приймає ці дані та зберігає їх у полях типу TEXT бази PostgreSQL. Це дозволило системі працювати повністю автономно, а користувачу — миттєво бачити фотографії квартир без затримок на зовнішню маршрутизацію.

Майже головною частиною системи є автоматичні сповіщення в реальному часі, побудовані на базі протоколу WebSockets (Socket.io). Користувачі комунікують в ізольованих віртуальних кімнатах (rooms). Щойно орендодавець

отримує нове повідомлення, навіть якщо він перебуває на іншій сторінці платформи, система активує подію подійного слухача фронтенду та виводить нативне браузерне push-сповіщення. Це повністю прибирає потребу оновлювати сторінку вручну або перевіряти чат самостійно.

Для забезпечення безпеки та розмежування прав розроблено механізм гвардів (Guards). Звичайний користувач бачить лише каталог, у той час як для облікового запису з роллю admin система динамічно розширює інтерфейс, виводячи Панель адміністратора для моніторингу та швидкого видалення застарілих даних.

Висновки. Вирішальне значення має те, як саме подаються та обробляються дані. Розроблена мікросервісна веб-платформа усуває потребу постійного ручного оновлення інтерфейсу, дає змогу миттєво зрозуміти поточний статус об'єкта нерухомості та вчасно отримувати повідомлення про нові події у чаті. Взаємодія з системою стає значно швидшою, зрозумілішою і максимально наближеною до реальних побутових потреб користувача.

Список використаних джерел:

1. NestJS Documentation: progressive Node.js framework. URL: https://docs.nestjs.com (дата звернення: 14.05.2026).
2. Socket.IO: Real-time bidirectional event-based communication. URL: https://socket.io/docs/v4/ (дата звернення: 14.05.2026).
3. PostgreSQL Text Types & Base64 storage. URL: https://www.postgresql.org/docs/ (дата звернення: 14.05.2026).

Сидоров Гліб Сергійович студент 4
курсу, групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій
(066) 202-67-92
sidorovgleb.2004@gmail.com

МЕТОД КОМПОНЕНТНОГО ПІДХОДУ В РОЗРОБЦІ 2D ПЛАТФОРМЕРА НА БАЗІ UNITY

Постановка задачі. Сучасна розробка ігор вимагає створення гнучких, масштабованих та зручних в обслуговуванні програмних архітектур. Це особливо актуально для 2D-платформерів, де ігрова логіка передбачає взаємодію великої кількості об'єктів: персонажів, ворогів, платформ, бонусів та фізичних елементів оточення.

Традиційні монолітні підходи до реалізації ігрової логіки ускладнюють повторне використання коду та знижують адаптивність проєкту. У цьому контексті доречним є використання компонентного підходу, оскільки він дозволяє розділити функціональність на незалежні модулі та забезпечує гнучке керування поведінкою ігрових об'єктів.

У середовищі Unity компонентна модель є фундаментальною; однак її ефективне використання вимагає належного проектування архітектури гри. Тому завдання полягає у дослідженні методу застосування компонентного підходу для створення 2D-платформера з метою підвищення модульності, можливості повторного використання коду та спрощення обслуговування програмного забезпечення.

Мета дослідження. Метою цього дослідження є аналіз та застосування компонентного підходу при розробці двовимірного платформера в Unity, а також визначення переваг використання цього підходу для створення гнучкої архітектури ігрового додатка.

Для досягнення поставленої мети необхідно:

- проаналізувати принципи компонентної архітектури в розробці програмного забезпечення;
- дослідити особливості реалізації компонентів у Unity;
- розробити структуру ігрових об'єктів на основі компонентного підходу;
- реалізувати базові ігрові механіки 2D платформера (рух, стрибок, взаємодія з об'єктами);
- оцінити ефективність застосованого підходу з точки зору гнучкості та масштабованості.

Результати дослідження. У цьому дослідженні проаналізовано компонентний підхід, який лежить в основі архітектури ігрового движка Unity. Його ключова ідея полягає в тому, що ігрові об'єкти реалізуються не як монолітні класи, а будуються як набір незалежних компонентів, кожен з яких виконує певну функцію. Такий підхід дозволяє створювати систему шляхом комбінування поведінок, що є більш гнучким порівняно з жорсткою ієрархією успадкування.

Встановлено, що використання компонентної архітектури дає змогу ефективно розкласти логіку 2D-платформера на окремі функціональні блоки. У структурі проєкту можна виділити компоненти, що відповідають за обробку

вхідних даних користувача, фізичну взаємодію об’єктів із середовищем, анімацію, ігрову логіку та обробку взаємодій і зіткнень. Такий розподіл забезпечує чітке розмежування відповідальності між частинами системи.

Застосовуючи компонентний підхід, було розроблено структуру 2D-платформера, в якій ігрові об’єкти представлені як сукупність незалежних модулів. Це дозволяє змінювати поведінку об’єктів шляхом додавання або видалення компонентів без необхідності переписування всієї логіки. В результаті зменшується взаємозалежність між частинами системи, а подальша модифікація спрощується.

Рух персонажа реалізовано за допомогою фізичного движка Unity, що забезпечує природну поведінку об'єктів під час руху та стрибків. Крім того, реалізовано механізм виявлення контакту з поверхнею, який правильно визначає, чи можливий стрибок, і запобігає його повторному запуску під час перебування в повітрі.

Використання компонентного підходу також забезпечує високу модульність ігрової логіки. Поведінка різних об'єктів може відрізнятися лише набором компонентів, що значно спрощує створення нових типів ворогів, платформ або інтерактивних елементів. Це дозволяє розширювати функціонал гри, не втручаючись істотно в існуючу систему.

Варто відзначити покращену масштабованість проекту. Нові механіки реалізуються шляхом створення додаткових компонентів, що знижує ризик виникнення помилок у існуючій логіці. Це також спрощує процес тестування та обслуговування, оскільки кожен компонент має чітко визначені функції.

У результаті було встановлено, що компонентна модель Unity є ефективною основою для розробки 2D-платформера, оскільки забезпечує архітектурну гнучкість, можливість повторного використання коду, зменшення взаємозалежності між системами, простоту розширення функціональності та зручність у супроводі програмного продукту.

Висновки та перспективи. У рамках дослідження було розглянуто компонентний підхід до розробки 2D-платформера з використанням Unity. Встановлено, що цей підхід є ефективною архітектурною моделлю, яка дозволяє створювати ігрові об’єкти у вигляді набору незалежних компонентів, що підвищує гнучкість та спрощує супровід проекту.

Було продемонстровано, що поділ логіки на окремі компоненти зменшує взаємозалежність між системами, покращує структуру коду та дозволяє легко змінювати поведінку ігрових об’єктів без значного перероблення всієї системи. Це особливо важливо для 2D-платформерів з великою кількістю взаємодіючих елементів.

Практична реалізація підтвердила, що компонентний підхід спрощує розробку, тестування та майбутнє розширення функціоналу гри.

Перспективи подальших досліджень включають використання архітектури ECS, підвищення продуктивності при великій кількості об’єктів та інтеграцію більш складних систем штучного інтелекту для ігрових персонажів.

Список використаних джерел

1. Unity Technologies. Unity Manual: GameObjects and Components. https://docs.unity3d.com/Manual/GameObjects.html

2. Unity Technologies. Scripting API Reference.

https://docs.unity3d.com/ScriptReference/
3. Gregory J. Game Engine Architecture. CRC Press. 2018.
4. Nystrom R. Game Programming Patterns. https://gameprogrammingpatterns.com/

Здоровенко Юлія Сергіївна,
студентка 4-го курсу, групи ПД-44
Державного університету інформаційно-комунікаційних технологій
juliavasilevna77@gmail.com

ОСОБЛИВОСТІ ВЕДЕННЯ ДОМАШНЬОЇ БІБЛІОТЕКИ ТА ТРЕКІНГУ ПРОЧИТАНОГО МАТЕРІАЛУ ЗА ДОПОМОГОЮ ВЕБ-ЗАСТОСУНКУ SHELFMATE

Постановка задачі: Відомо, що домашня бібліотека може містити велику кількість книг. З часом вони можуть стати складними для швидкого знаходження тієї книги, що потрібна. Крім того, бажання вести відстеження тих книг, що вже були прочитані, або ті, що ще не відіграли в читацькому прогресі, дає змогу зберігати інформацію в хаотичному вигляді: в нотатках, списках або в інших додатках. У такому випадку виникає плутанина та втрачається зручність. Тому існуючою вважається задачею створення веб-застосунку, що зможе вирішити цю проблему.

Мета дослідження: Метою дослідження є визначення можливостей, технологій та принципів побудови веб-застосунку для ведення домашньої бібліотеки.

Результати дослідження: Веб-застосунок для ведення бібліотеки повинен мати можливість додавання книг, їх класифікації та встановлення статусу книг (прочитано, читається, заплановано). Також повинна бути можливість зберігання нотаток до книг та їх оцінки. Для серверної частини веб-застосунку доцільно використати Java [1]. Framework Spring для створення серверної частини веб-застосунку. Вони дозвілять легко створити окремі модулі веб-застосунку: модуль книг, модуль користувачів, модуль статусів читання та нотаток [2].

Для збереження даних веб-застосунку доцільність використовувати PostgreSQL. Вона дозволяє зберігати книги, авторів книг, їх жанри, статуси читання, оцінки до книг та нотатки до користувача [3]. Клієнтську частину веб-застосунку можна створити використовуючи React. Вона дозволяє побудувати веб-застосунок для роботи із книгами, створити картки до книг, додати можливість створення нотаток до книг [4].

Особливу увагу слід приділити такому дослідження, щоб забезпечити зручність використання веб-застосунку. Він має допомагати в зберіганнях книг, а також швидкому знаходження та відстежуванні прогресу у читанні.

На діаграмі відображено, як користувач може взаємодіяти із веб-застосунком для ведення бібліотеки. Він може додати нові книги до бібліотеки чи знайти існуючі в каталозі книги. Після відображення книги користувач встановить її статус та доданні нотатки. Ці дані зберуться в системі, а відображення книги і її статуси стане можливим для перегляду. Система зможе також надати відображення книг, що вже були прочитані та аналізувати прогрес у читанні.

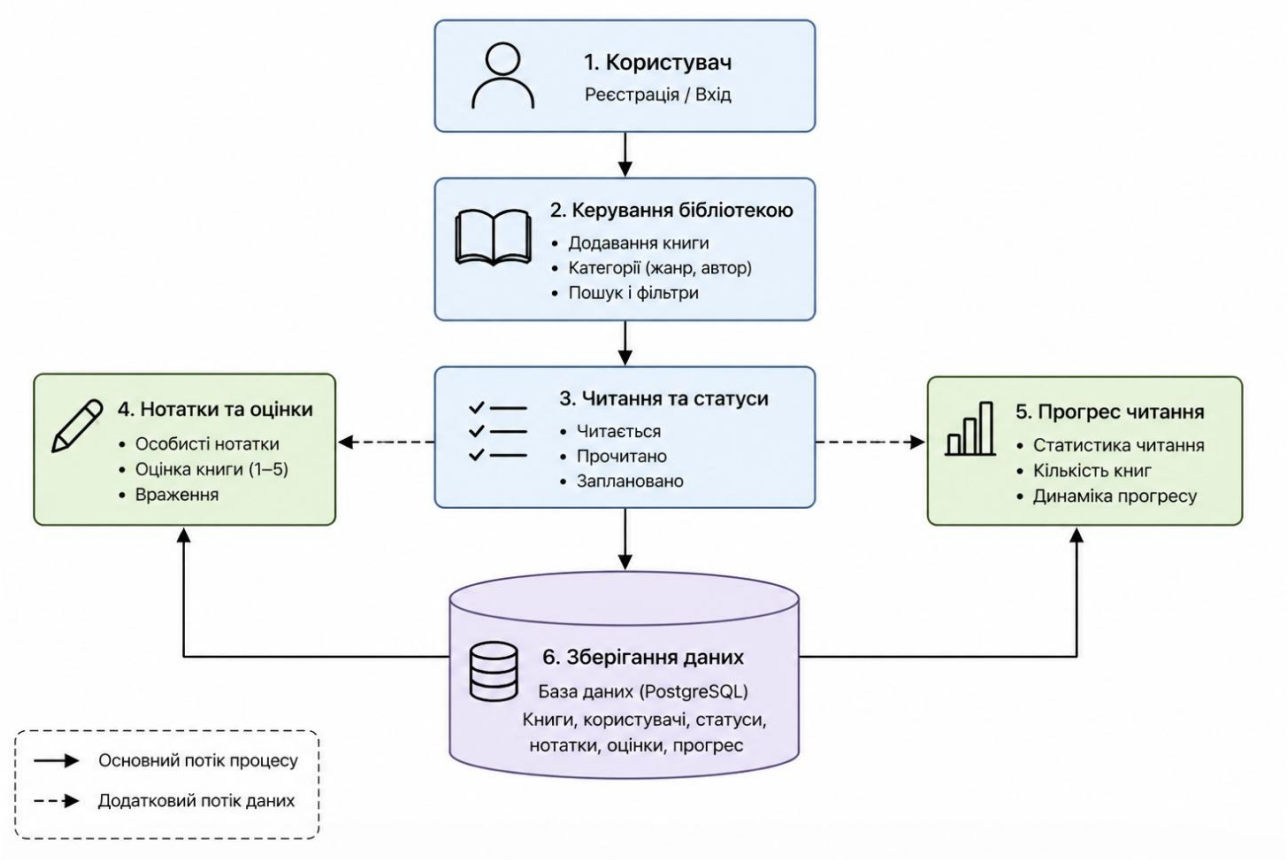


Рис. 1 - Схема роботи користувача з веб-застосунком домашньої бібліотеки

Висновки та перспективи: Отже, створення веб-застосунку для ведення бібліотеки є доцільним. Він допоможе відсортувати книги в бібліотеці та зробити читання більш зручним. Використання описаних технологій у дослідженні дозволяє створити справжній веб-застосунок для ведення бібліотеки. Також в можливості розширення веб-застосунку в майбутньому: вставка можливості читання книг за рекомендаciones, створення статистики вчествування читання книги або інтеграції із онлайн-бібліотеками.

Список використаних джерел

1. Національна бібліотека України імені В. І. Вернадського. “Електронні ресурси”. URL: http://www.nbuv.gov.ua/ (дата звернення 10.05.2026).
2. Spring Framework - spring.io – [Електронний ресурс] – Режим доступу: https://spring.io/projects/spring-framework
3. PostgreSQL Documentation - postgresql.org – [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/
4. React - react.dev – [Електронний ресурс] – Режим доступу: https://react.dev/

Новостройний Дмитро Андрійович
студент 3 курсу, групи ТЦР-31
Державний університет інформаційно-комунікаційних технологій
(063) 526-01-06
novostroinyidima23@gmail.com

МОДЕЛЮВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ АВТОМАТИЗАЦІЇ ОБРОБКИ ЗАЯВОК

Постановка задачі. У діяльності сервісних підприємств значна кількість операцій пов’язана з обробкою заявок користувачів, контролем їх виконання та формуванням звітності. Використання розрізнених програмних засобів або ручної обробки даних призводить до дублювання інформації, збільшення часу обробки запитів та ускладнення контролю виконання задач. Особливої актуальності набуває проблема централізованого управління інформаційними потоками та забезпечення ефективної взаємодії між користувачами системи. У зв’язку з цим виникає необхідність моделювання архітектури програмної системи, здатної забезпечити автоматизацію процесів створення, обробки та моніторингу заявок.

Мета дослідження. Метою дослідження є розробка структурної моделі архітектурної програмної системи автоматизації обробки заявок та формалізація взаємодії між функціональними компонентами системи.

Результати дослідження. У межах дослідження запропоновано структурну модель архітектури програмної системи автоматизації обробки заявок, побудовану на основі багаторівневого підходу [1]. Основною особливістю запропонованої моделі є розділення системи на незалежні функціональні рівні: рівень представлення даних, рівень бізнес-логіки та підсистему доступу до даних. Такий підхід забезпечує модульність системи, спрощує супровід програмного забезпечення та зменшує залежність між окремими компонентами архітектури [3].

На рис. 1 наведено запропоновану структурну модель архітектури програмної системи автоматизації обробки заявок, що включає рівень представлення даних, модуль бізнес-логіки та підсистему доступу до даних.

У процесі дослідження побудовано UML-діаграму компонентів, яка формалізує взаємодію між вебінтерфейсом системи, модулем обробки заявок, підсистемою керування користувачами та базою даних [2]. Рівень представлення даних забезпечує взаємодію користувача із системою та реалізує функції створення, перегляду й моніторингу заявок. Рівень бізнес-логіки виконує обробку запитів, контроль статусів виконання, перевірку прав доступу та формування звітності. Підсистема доступу до даних реалізує механізми роботи із централізованою базою даних та забезпечує обробку транзакцій і валідацію даних.

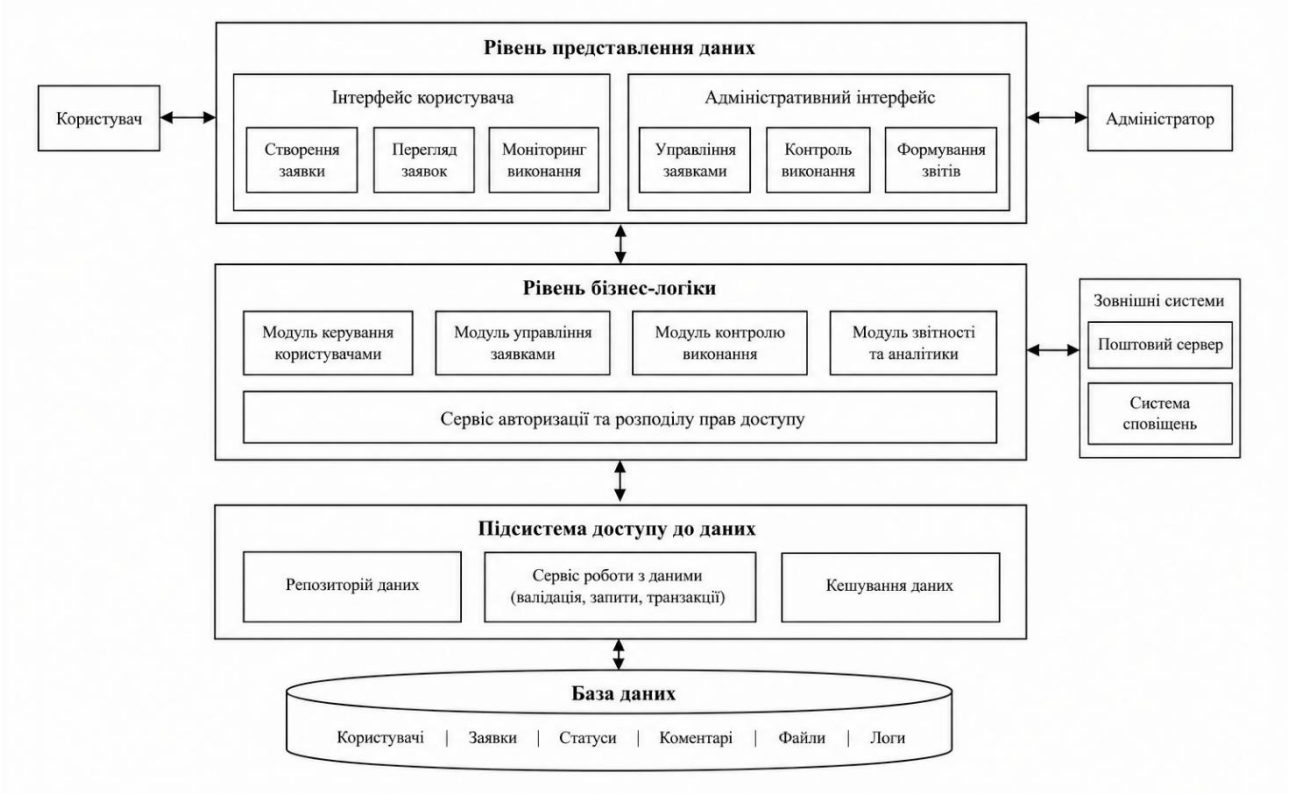


Рис. 1. Структурна модель архітектури програмної системи автоматизації обробки заявок

Для моделювання структури даних використано ER-модель, що включає сутності «Користувач», «Заявка», «Статус», «Коментар» та «Звіт» [4]. Використання централізованого сховища даних дозволяє мінімізувати дублювання інформації та забезпечити цілісність даних між функціональними модулями системи.

Запропонована модель архітектури дозволяє реалізувати розширення функціональних можливостей програмної системи без модифікації базових компонентів, а також спрощує інтеграцію із зовнішніми сервісами повідомлень та аналітики [5].

Висновки та перспективи. У результаті дослідження розроблено структурну модель архітектури програмної системи автоматизації обробки заявок сервісного підприємства. Використання багаторівневого підходу дозволило забезпечити розділення функціональних компонентів системи та підвищити керованість інформаційними потоками. Побудована UML-діаграма компонентів формалізує взаємодію між основними модулями системи та може бути використана як основа для подальшої реалізації програмного забезпечення. Запропонована модель сприяє підвищенню ефективності обробки заявок, спрощує супровід системи та забезпечує можливість подальшого масштабування архітектури.

Список використаних джерел

1. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional. 2002. Vol. 1, P. 533.

2. K. Hamilton and R. Miles, "Learning UML 2.0: A Pragmatic Companion to UML", O'Reilly Media, vol. 1, no. 1, pp. 30-45, 2006.

3. Bass L., Clements P., Kazman R. Software Architecture in Practice. Addison-Wesley. 2013. №3, P. 640.

4. Берко А. Ю., Верес О. М., Пасічник В. В. Системи баз даних та знань. Книга 1. Організація баз даних та знань. Магнолія 2006. 2015. Вип. 2, С. 440.

5. Гужва В. М. Інформаційні системи і технології на підприємствах : навчальний посібник. Київ : КНЕУ, 2001. 400 с.

Мельник Владислав Олександрович
студент 4 курсу, групи ПД-44
Державного університету інформаційно-комунікаційних технологій
st8037452@stud.duikt.edu.ua

ПРОГРАМНА СИСТЕМА AROMASHOP ДЛЯ ОНЛАЙН-ПРОДАЖУ ТА ПІДБОРУ ПАРФУМЕРНОЇ ПРОДУКЦІЇ

Постановка задачі

Сучасний розвиток електронної комерції сприяє активному переходу торговельних процесів у цифрове середовище. Онлайн-продаж товарів дозволяє користувачам швидко переглядати асортимент, порівнювати характеристики продукції, оформлювати замовлення дистанційно та отримувати актуальну інформацію про наявність товарів. Особливої уваги потребує сфера продажу парфумерної продукції, оскільки вибір аромату залежить не лише від ціни або бренду, а й від індивідуальних вподобань користувача, ароматичного сімейства, концентрації, нот, об’єму, відгуків та ситуації використання.

Основна проблема полягає в тому, що під час онлайн-вибору парфумерії користувач не має можливості фізично протестувати аромат. Через це важливим стає детальне цифрове представлення товару: опис аромату, парфумерна піраміда, характеристики, зображення, рейтинг, відгуки та актуальна інформація про наявність. Крім того, користувач часто не знає точної назви товару або бренду, але розуміє, для якої події йому потрібен аромат: для офісу, побачення, подарунка, щоденного використання або вечірнього виходу.

У зв’язку з цим доцільною є розробка спеціалізованого web-застосунку AromaShop, який забезпечує централізоване керування каталогом парфумерної продукції, пошук і фільтрацію товарів, роботу з кошиком, оформлення замовлень, залишення відгуків, адміністрування асортименту та підбір ароматів за подією.

Мета дослідження

Метою дослідження є визначення функціональних можливостей, засобів реалізації та архітектурних вимог до web-застосунку AromaShop для онлайн-продажу парфумерії з урахуванням специфіки цифрового представлення ароматів і сценарного підбору товарів за подією.

Результати дослідження

Для реалізації серверної частини web-застосунку доцільно використати мову програмування Java. Вона підходить для створення стабільної серверної логіки, обробки запитів користувачів, керування товарами, брендами, кошиком, замовленнями, відгуками та рекомендаціями. Використання Java дозволяє побудувати багаторівневу структуру системи та забезпечити подальше розширення функціоналу [1].

Основним фреймворком для backend-частини обрано Spring Boot. Його використання дозволяє реалізувати REST API, підключити базу даних, організувати модульну структуру серверної частини, розмежувати контролери, сервіси, репозиторії та доменні сутності. Для AromaShop це важливо, оскільки система має обробляти різні групи даних: товари, бренди, користувачів, кошики, замовлення, відгуки та рекомендації за подією [2].

Для зберігання даних доцільно використати PostgreSQL. У базі даних можуть зберігатися користувачі, бренди, товари, зображення товарів, парфумерні

ноти, кошики, позиції кошика, замовлення, відгуки та рекомендації. Реляційна структура PostgreSQL дозволяє чітко організувати зв’язки між сутностями: один бренд може мати багато товарів, один користувач може мати багато замовлень, один товар може мати багато відгуків і парфумерних нот [3].

Клієнтську частину web-застосунку доцільно реалізувати за допомогою React. Ця бібліотека дозволяє створити компонентний інтерфейс, у якому окремо реалізуються сторінки каталогу, товару, кошика, оформлення замовлення, історії покупок, профілю користувача, адміністративної панелі та функції «Підбір за подією». Використання TypeScript підвищує надійність frontend-коду, оскільки дозволяє описати типи товарів, користувачів, замовлень, відгуків і відповідей API [4].

Архітектурно система AromaShop побудована за клієнт-серверним підходом. Клієнтська частина відповідає за відображення інтерфейсу та взаємодію з користувачем, серверна частина виконує бізнес-логіку, а база даних забезпечує зберігання інформації. Такий підхід відповідає сучасним принципам побудови web-систем електронної комерції та дозволяє окремо масштабувати frontend, backend і рівень зберігання даних [5].

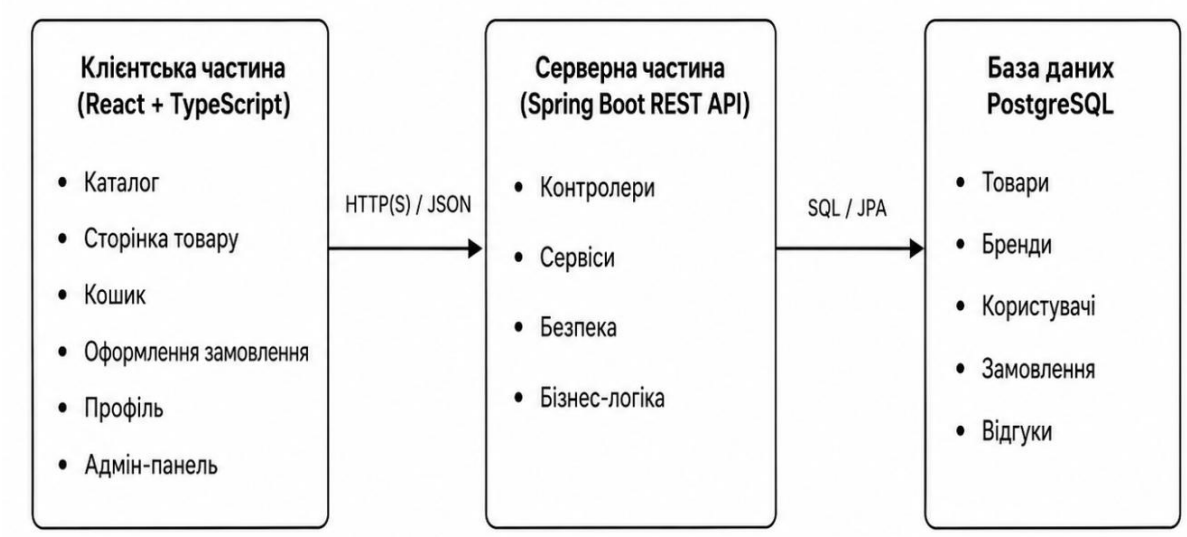


Рис. 1. Схема взаємодії клієнтської та серверної частин web-застосунку AromaShop

Особливістю розроблюваної системи є функція «Підбір за подією». Вона дозволяє користувачу обрати ситуацію, для якої потрібен аромат, після чого система задає уточнювальні питання та формує список відповідних товарів. Наприклад, для подарунка можуть враховуватися отримувач, бюджет і бажаний стиль аромату, а для офісу — легкість, нейтральність і ненав’язливість композиції. У результаті користувач отримує не просто список товарів, а рекомендації з поясненням, чому певний аромат підходить до обраної ситуації.

Реалізація такої функції підвищує практичну цінність web-застосунку, оскільки стандартні інтернет-магазини переважно пропонують пошук за назвою, брендом або фільтрами, але не завжди допомагають користувачу зробити вибір за сценарієм використання. Тому AromaShop можна розглядати як спеціалізовану програмну систему для цифрового продажу парфумерії, яка враховує особливості предметної галузі.

Висновки та перспективи

Таким чином, проведене дослідження показало доцільність створення web-застосунку AromaShop для онлайн-продажу парфумерної продукції. Запропоноване рішення дозволяє поєднати каталог товарів, пошук, фільтрацію, сторінку детального перегляду аромату, кошик, оформлення замовлень, відгуки, адміністративне керування та підбір ароматів за подією.

Перспективами розвитку системи є додавання онлайн-оплати, інтеграція зі службами доставки, реалізація списку обраних товарів, система промокодів, персоналізовані рекомендації на основі історії покупок, аналітика продажів і розширення механізму підбору ароматів за поведінкою користувача.

Список використаних джерел

1. Java Platform, Standard Edition Documentation. Oracle. URL: https://docs.oracle.com/en/java/javase/21/ (date of access: 25.05.2026).
2. Spring Boot Reference Documentation. VMware Tanzu. URL: https://docs.spring.io/spring-boot/reference/index.html (date of access: 25.05.2026).
3. PostgreSQL Documentation. The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/ (date of access: 25.05.2026).
4. React Documentation. Meta Open Source. URL: https://react.dev/ (date of access: 25.05.2026).
5. Краус К. М., Краус Н. М., Манжура О. В. Електронна комерція та Інтернет-торгівля : навч.-метод. посіб. Київ : Аграр Медіа Груп, 2021. 454 с. URL: https://elibrary.kubg.edu.ua/id/eprint/37044/1/Kraus_Elektronna_komertsiia_2021.pdf (дата звернення: 25.05.2026).

Напрям 2 ДЕЦЕНТРАЛІЗОВАНІ ІНФОРМАЦІЙНІ СИСТЕМИ

Чохленко Богдан Андрійович
бакалавр 4 курсу, групи ТЦР-43
Державний університет інформаційно-комунікаційних технологій
(098)520-33-50
chokhlenkoo@gmail.com

ДЕЦЕНТРАЛІЗАЦІЯ ДАНИХ У СУЧАСНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ: НЕОБХІДНІСТЬ ТА ПЕРЕВАГИ

Постановка задачі. Більшість сучасних інформаційних систем побудована за централізованою архітектурою: усі дані зберігаються на одному сервері або в одному хмарному середовищі, яке виступає єдиною точкою управління. Такий підхід зручний, але ризикований: якщо головний сервер вийде з ладу, зупиниться вся система, а злом одного сховища дасть доступ до всієї інформації. Проблема полягає в тому, що з ростом кількості користувачів навантаження на центр стає критичним, а ціна будь-якої помилки чи збою зростає в геометричній прогресії. Сьогодні, коли даних стає все більше, перехід до децентралізації стає необхідним для безпеки та стабільності [1].

Мета дослідження. Розглянути, чому децентралізація даних є важливою для сучасних систем, виділити її головні плюси та мінуси, а також зрозуміти, у яких сферах цей підхід працює найкраще.

Результати дослідження. Децентралізація - це спосіб організації даних, при якому вони зберігаються та обробляються на різних незалежних вузлах, а не в одному центрі. Головна перевага тут у тому, що жоден сервер не є критично важливим для всієї мережі.

Перш за все, це забезпечує високу відмовостійкість. Якщо у звичайній системі ламається головний сервер, вона повністю зупиняється. У децентралізованій архітектурі вихід з ладу одного вузла не є катастрофою: запити просто перейдуть на інші робочі сервери. Це дуже важливо для банків чи медицини, де будь-яка зупинка коштує дорого. Більш того, завдяки відсутності єдиного центру, система може продовжувати роботу навіть у випадку часткового пошкодження інфраструктури, наприклад, під час мережевих розривів. До того ж, такі системи відновлюються після збоїв значно швидше.

Другим моментом є безпека. Звичайне сховище - легка ціль: зламав один сервер і отримав усі дані. При децентралізації дані фрагментовані та зашифровані по всій мережі, тому злом кількох вузлів не дасть повної картини. Зловмиснику довелося б атакувати одночасно значну частину мережі, що потребує величезних ресурсів. Цей же принцип робить блокчейн таким надійним, адже змінити дані в ньому майже неможливо [2].

Також однією з ключових переваг є масштабованість. Потужність одного сервера завжди має межу, і будь-яка система при великому притоці користувачів її досягне. Децентралізовані системи дозволяють просто додавати нові вузли (горизонтальне масштабування), збільшуючи загальну швидкість роботи без заміни всього обладнання. Це дозволяє бізнесу уникати великих одноразових витрат на дорогі суперкомп'ютери, поступово нарощуючи потужність мережі за потреби. За таким

принципом працюють великі бази даних на кшталт Apache Cassandra та CockroachDB [1].

Окремою перевагою - дотримання законів про захист даних, як-от GDPR [3]. Завдяки географічному розподілу вузлів можна зберігати дані користувачів безпосередньо в межах їхніх країн, що значно спрощує юридичну роботу. Це позбавляє компанії необхідності створювати окремі дата-центри для кожної країни, оскільки архітектура вже дозволяє гнучко керувати місцем зберігання інформації.

Проте децентралізація має і складнощі. Важко зробити так, щоб дані на всіх вузлах оновлювалися одночасно: завжди є ризик прочитати застарілу інформацію з одного з них. Цю проблему описує теорема CAP: розробникам завжди доводиться обирати між швидкістю, доступністю та цілісністю даних [1]. Крім того, розробка та адміністрування таких систем потребує вищої кваліфікації персоналу, що може стати додатковим викликом для невеликих організацій.

Висновки. Децентралізація даних є надійним архітектурним рішенням для систем, де на першому місці стоять стійкість до збоїв, можливість легкого розширення та захист від атак. Попри складність впровадження, переваги у вигляді розподілу ризиків та відповідності законодавству зазвичай переважають усі недоліки. У майбутньому варто детальніше дослідити, як забезпечити швидку синхронізацію даних між вузлами та оцінити, наскільки вигідним є такий перехід з фінансової точки зору.

Список використаних джерел

1. Kleppmann M. Designing Data-Intensive Applications. Sebastopol: O'Reilly Media, 2017. 616 p.

2. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. URL: https://bitcoin.org/bitcoin.pdf

3. Regulation (EU) 2016/679 (GDPR). Official Journal of the European Union. 2016. L 119. P. 1–88.

Доманський Владислав Сергійович
студент 4 курсу
спеціальності «Інженерія програмного забезпечення»
Державного університету інформаційно-комунікаційних технологій, м. Київ
domanski0308@gmail.com

Бажан Тетяна Олександрівна
старший викладач кафедри Технологій цифрового розвитку
Державного університету інформаційно-комунікаційних технологій, м. Київ

АВТОНОМНИЙ ВЕБ-ДОДАТОК ДЛЯ УПРАВЛІННЯ НОТАТКАМИ

Сучасні користувачі активно використовують цифрові інструменти для організації особистої інформації, зокрема створення нотаток, планування завдань та збереження ідей. Більшість існуючих рішень потребують підключення до мережі Інтернет або використання серверної частини, що ускладнює доступ до даних у випадку відсутності з'єднання. Тому актуальним є створення веб-додатків, які можуть працювати автономно та забезпечувати збереження даних без використання зовнішніх серверів.

Метою дослідження є аналіз можливостей використання клієнтських веб-технологій для створення автономного додатку управління нотатками, а також реалізація ефективного механізму збереження та обробки даних користувача у браузері.

У процесі дослідження було розглянуто можливості сучасних веб-технологій для реалізації повнофункціонального клієнтського додатку. Основну увагу приділено використанню JavaScript для реалізації логіки роботи системи, HTML для створення структури інтерфейсу та CSS для оформлення зовнішнього вигляду.

Для забезпечення автономної роботи було використано технологію localStorage, яка дозволяє зберігати дані безпосередньо у браузері користувача. Це дозволило реалізувати систему без серверної частини та забезпечити швидкий доступ до даних[1,3].

Розроблений додаток дозволяє користувачу виконувати основні операції з нотатками, включаючи їх створення, редагування, видалення та перегляд. Також реалізовано механізм категоризації записів, що дозволяє ефективно організовувати інформацію[2].

Особливістю системи є простота використання та швидкість роботи, оскільки всі операції виконуються на стороні клієнта без додаткових затримок, пов'язаних із передачею даних через мережу (рис. 1).

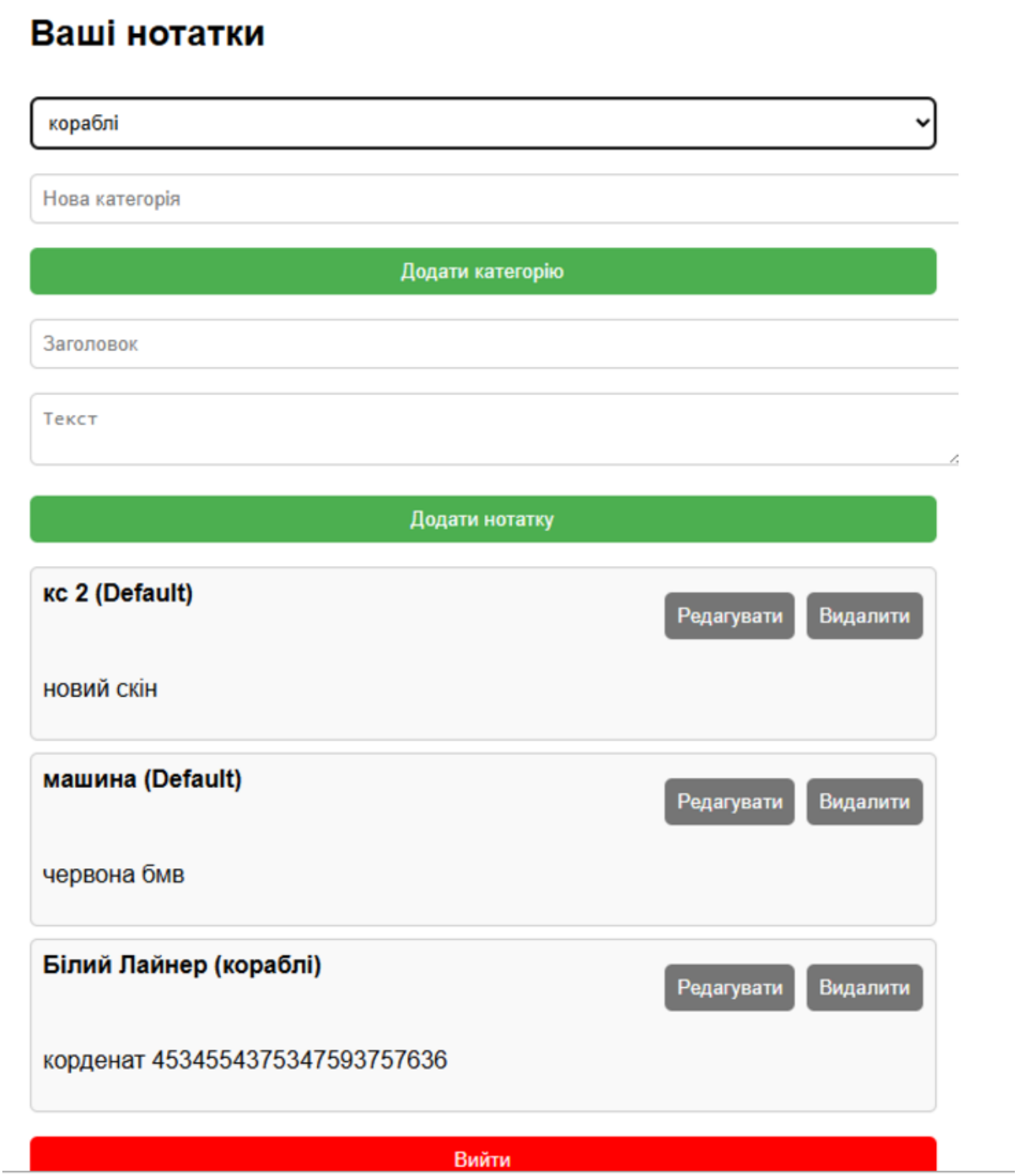


Рис. 1. Інтерфейс автономного веб-додатку для управління нотатками

Проведене дослідження показало, що використання клієнтських веб-технологій дозволяє створювати ефективні автономні інформаційні системи без необхідності застосування серверної інфраструктури.

Подальший розвиток системи може включати додавання функцій резервного копіювання даних, реалізацію пошуку та фільтрації нотаток, а також розширення можливостей інтерфейсу користувача для підвищення зручності використання.

Висновки та перспективи

Проведене дослідження показало, що використання клієнтських веб-технологій дозволяє створювати ефективні автономні інформаційні системи без необхідності застосування серверної інфраструктури.

Подальший розвиток системи може включати додавання функцій резервного

копіювання даних, реалізацію пошуку та фільтрації нотаток, а також розширення можливостей інтерфейсу користувача для підвищення зручності використання.

Список використаних джерел

1. Flanagan, D. (2020). JavaScript: The Definitive Guide (7th ed.). O’Reilly Media.
2. Duckett, J. (2014). JavaScript and jQuery: Interactive Front-End Web Development. Wiley.
3. Mozilla Developer Network (MDN). Web Storage API. URL: https://developer.mozilla.org

Моргунов Артем Олександрович
магістр 1 курсу, групи ТЦРм-51
Державний університет інформаційно-комунікаційних технологій
+380997500477
Тьотченко Денис Григорович
магістр 1 курсу, групи ТЦРм-51
Державний університет інформаційно-комунікаційних технологій
+380961733443

АДАПТИВНИЙ МЕТОД ФОРМУВАННЯ КВОРУМУ В ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМАХ

Постановка задачі. Розвиток блокчейн-технологій та децентралізованих інформаційних систем зумовлює необхідність удосконалення механізмів досягнення консенсусу між вузлами розподіленої мережі. Одним із ключових елементів таких механізмів є формування кворуму валідаторів, достатнього для підтвердження транзакцій, формування блока та забезпечення узгодженості розподіленого реєстру. Ефективність цього процесу безпосередньо впливає на затримку підтвердження транзакцій, пропускну здатність і стійкість децентралізованої системи.

У традиційних BFT-орієнтованих механізмах склад валідаторів і правила формування кворуму переважно визначаються до початку раунду консенсусу. Такий підхід є ефективним для мереж із відносно стабільними характеристиками, проте має обмеження при зміні доступності вузлів, збільшенні мережевих затримок, нерівномірному навантаженні або виникненні аномальної поведінки. Вузол може залишатися формально активним, але через значну затримку або перевантаження сповільнювати процес досягнення консенсусу. Ще більшу загрозу становить залучення до кворуму вузлів, поведінка яких має ознаки компрометації.

Тому актуальною є задача формування кворуму не лише за фактом доступності валідаторів, а з урахуванням їх поточного функціонального стану.

Мета дослідження полягає у підвищенні ефективності та стійкості децентралізованої системи шляхом розроблення адаптивного методу формування кворуму валідаторів, який враховує поточну доступність вузлів, мережеву затримку, рівень навантаження та ризик аномальної або компрометованої поведінки.

Результати дослідження. Запропоновано адаптивний метод формування кворуму, особливістю якого є динамічне оцінювання придатності кожного вузла до участі в поточному раунді консенсусу. На відміну від статичного підходу, склад кворуму не фіксується на тривалий період, а формується перед кожним раундом на основі актуальних даних моніторингу.

Для оцінювання вузла використовуються чотири групи параметрів: доступність, мережева затримка, поточне навантаження та ризик аномальної поведінки. Після отримання телеметрії параметри нормуються та приводяться до єдиної шкали. На їх основі визначається інтегральний показник надійності вузла. Чим вищим є його значення, тим більший пріоритет має відповідний вузол під час формування кворуму.

Якщо показник надійності перевищує встановлений поріг, вузол включається до множини кандидатів у валідатори. Вузли, характеристики яких не відповідають заданим вимогам, тимчасово виключаються з поточного раунду. Таке виключення

не є остаточним: після оновлення телеметрії та покращення характеристик вузол може повторно бути залучений до консенсусу.

Після попереднього відбору кандидати ранжуються за показником надійності. Кворум формується з вузлів, які мають найкращі поточні характеристики, але зі збереженням установлених вимог BFT-консенсусу до необхідної кількості підтверджень. Таким чином, адаптація не досягається шляхом довільного зменшення кількості валідаторів, що могло б негативно вплинути на безпеку системи.

Послідовність функціонування запропонованого методу наведено на рис. 1.

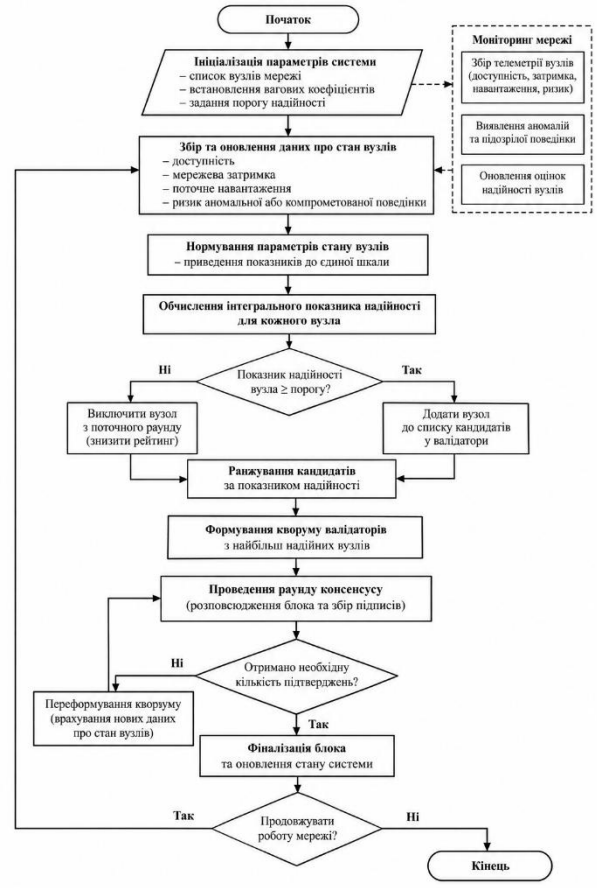


Рис. 1. Блок-схема адаптивного методу формування кворуму в децентралізованій системі

Для попередньої перевірки запропонованого підходу проведено модельний експеримент для децентралізованої мережі з 40 вузлами. Порівнювалися статичний підхід, за якого склад кандидатів не змінювався відповідно до поточного стану мережі, та запропонований адаптивний метод.

Результати моделювання показали, що в нормальному режимі перевага адаптивного підходу є відносно невеликою, оскільки характеристики більшості вузлів залишаються близькими. Зменшення середнього часу досягнення консенсусу становило близько 12 %. При деградації мережевого середовища ефект збільшувався до 26 %, оскільки система виключала з поточного кворуму вузли з найбільшою затримкою. У комбінованому сценарії скорочення часу досягнення консенсусу становило близько 30 %.

Важливим результатом є те, що отримане скорочення затримки забезпечується не за рахунок послаблення вимог до кількості підтверджень, а шляхом зміни якісного складу кворуму. Отже, запропонований підхід дозволяє підвищувати продуктивність без свідомого зменшення встановленого рівня BFT-стійкості.

Висновки. Запропонований адаптивний метод забезпечує динамічне формування кворуму відповідно до фактичного стану вузлів децентралізованої системи. Поєднання постійного моніторингу, інтегрального оцінювання, ранжування та механізму переформування кворуму створює можливість оперативно реагувати на деградацію мережі, перевантаження або появу потенційно ненадійних вузлів. Результати модельного дослідження свідчать про найбільшу ефективність методу саме за нестабільних умов функціонування децентралізованої мережі. Подальші дослідження доцільно спрямувати на автоматичне налаштування ваг параметрів оцінювання вузлів та експериментальну перевірку методу в програмному прототипі децентралізованої системи.

Список джерел

1. Bano S., Sonnino A., Al-Bassam M. et al. SoK: Consensus in the Age of Blockchains. *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 2019. P. 183–198.
2. Yin M., Malkhi D., Reiter M. K., Gueta G. G., Abraham I. HotStuff: BFT Consensus with Linearity and Responsiveness. *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. 2019. P. 347–356.
3. Danezis G., Kokoris-Kogias L., Sonnino A., Spiegelman A. Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus. *Proceedings of the Seventeenth European Conference on Computer Systems*. 2022. P. 34–50.
4. Gilad Y., Hemo R., Micali S., Vlachos G., Zeldovich N. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. *Proceedings of the 26th Symposium on Operating Systems Principles*. 2017. P. 51–68.

Жебка Вікторія Вікторівна
Д.т.н., професор, завідувач кафедри Технологій цифрового розвитку
Державний університет інформаційно-комунікаційних технологій
+380730508300

МЕТОД АДАПТИВНОГО РОЗМІЩЕННЯ РЕПЛІК ДАНИХ У ДЕЦЕНТРАЛІЗОВАНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

Постановка задачі. Децентралізовані інформаційні системи характеризуються відсутністю єдиного центру зберігання та управління даними. Інформаційні ресурси розподіляються між множиною автономних вузлів, що підвищує стійкість системи до відмов окремих компонентів, але водночас створює проблему забезпечення постійної доступності даних. Відключення вузла, розрив мережевого з'єднання або тимчасова ізоляція частини мережі можуть призвести до ситуації, коли необхідні дані фізично існують у системі, проте є недоступними для певної групи користувачів.

Одним зі способів підвищення доступності є реплікація даних. У найпростішому випадку для кожного інформаційного об'єкта створюється фіксована кількість копій, які розміщуються на різних вузлах. Недоліком такого підходу є те, що однаковий рівень реплікації використовується незалежно від важливості даних, частоти звернень до них і поточного стану мережі. Надмірна кількість реплік збільшує обсяг передавання даних і витрати ресурсів, тоді як недостатня кількість копій знижує доступність інформації при групових відмовах вузлів.

Особливо критичною є ситуація, коли декілька копій одного об'єкта зберігаються на вузлах одного логічного або мережевого сегмента. При ізоляції такого сегмента формальна наявність декількох реплік не забезпечує фактичної відмовостійкості.

Мета дослідження полягає у підвищенні доступності даних децентралізованої інформаційної системи при обмеженні надлишкових витрат ресурсів шляхом розроблення методу адаптивного розміщення реплік з урахуванням критичності інформаційних об'єктів, частоти звернень до них, стану вузлів та топологічного розподілу мережі.

Результати дослідження. Запропоновано метод адаптивного розміщення реплік, відповідно до якого кількість копій інформаційного об'єкта та вузли їх розміщення визначаються динамічно. На відміну від фіксованої реплікації, система періодично аналізує стан даних і мережевого середовища та приймає рішення про створення, переміщення або видалення окремих реплік.

Принциповою особливістю запропонованого підходу є введення топологічного рознесення реплік. Метод забороняє вважати декілька копій достатнім резервуванням, якщо всі вони знаходяться в одному сегменті мережі. Репліки критичних даних розподіляються між різними групами вузлів таким чином, щоб відмова або ізоляція одного сегмента не призводила до втрати доступу до всіх актуальних копій.

Загальну послідовність функціонування методу доцільно подати у вигляді блок-схеми на рис. 1.

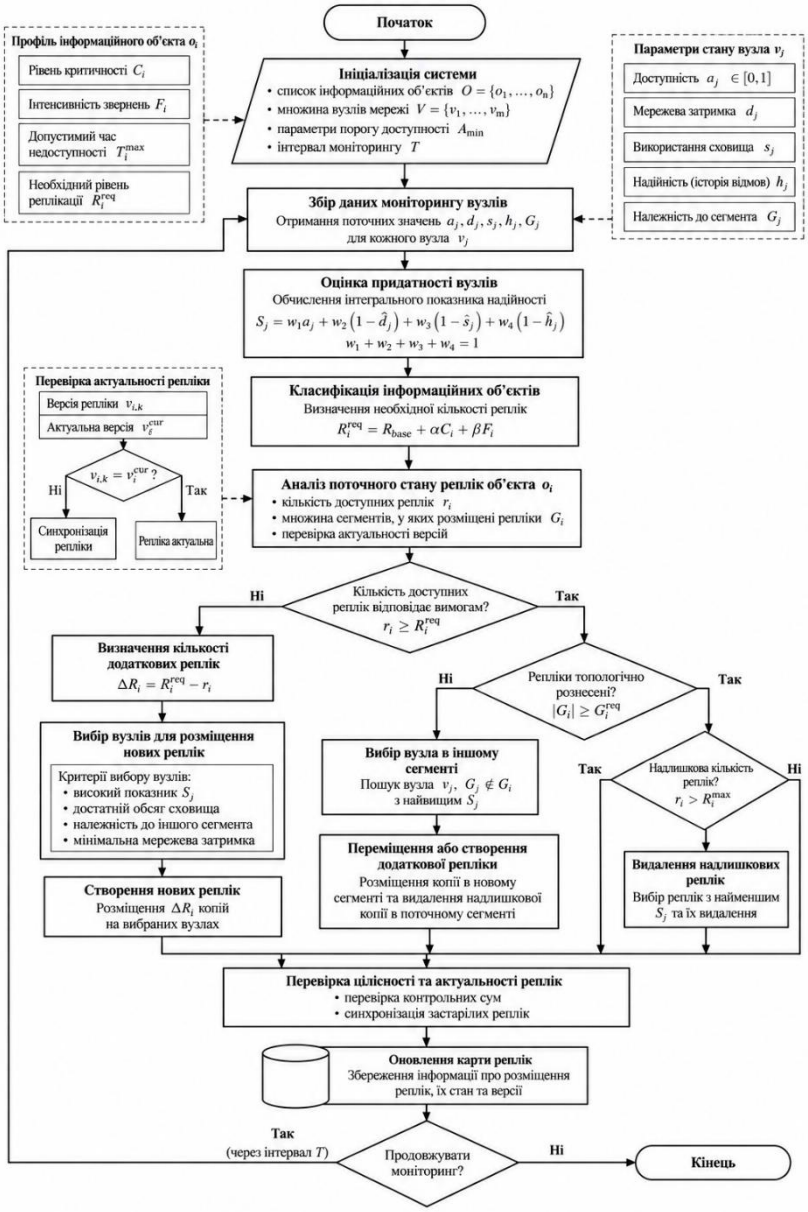


Рис. 1. Блок-схема методу адаптивного розміщення реплік даних у децентралізованій системі

Алгоритм починається з формування карти вузлів і розміщених на них інформаційних об'єктів. Далі здійснюється класифікація даних за рівнем критичності та визначення необхідного рівня резервування. Після отримання даних моніторингу оцінюється доступність реплік і перевіряється їх топологічне рознесення.

Якщо встановлені вимоги виконуються, поточна конфігурація зберігається до наступного циклу контролю. Якщо кількість доступних реплік є недостатньою, система вибирає новий вузол і створює додаткову копію. При виявленні топологічної концентрації виконується переміщення однієї з реплік до іншого сегмента мережі. Після кожної зміни здійснюється перевірка цілісності та актуальності створеної копії.

Для дослідження властивостей запропонованого підходу розглянуто модель децентралізованої системи з 50 вузлами, розподіленими між п'ятьма мережевими сегментами. Порівнювалися два підходи: фіксована трикратна реплікація та запропоноване адаптивне розміщення.

Моделювалися три типи порушень: випадкові відмови окремих вузлів, групова відмова вузлів одного сегмента та поступове збільшення частоти відмов у

певній частині мережі. Найбільша різниця між підходами спостерігалася при групових відмовах. За фіксованої реплікації декілька копій могли одночасно стати недоступними через їх розміщення в одному сегменті. Запропонований метод виявляв таку концентрацію ще до відмови та ініціював створення копії в іншому сегменті.

Дослідження також показало, що збільшення кількості реплік саме по собі не є достатньою умовою підвищення стійкості. Важливішим є їх розміщення відносно потенційно спільних точок відмови. Тому адаптивне керування дозволяє досягти вищої доступності без необхідності постійного зберігання максимальної кількості копій кожного об'єкта.

Висновки. Запропонований метод дозволяє розглядати реплікацію не як статичне дублювання даних, а як керований адаптивний процес. Його використання забезпечує перерозподіл копій при погіршенні стану вузлів, запобігає концентрації критичних реплік у межах одного сегмента та дозволяє зменшувати надлишкове використання ресурсів у стабільному режимі. Перспективним напрямом подальших досліджень є прогнозування групових відмов за часовими рядами телеметрії та випереджувальне переміщення критичних реплік до виникнення фактичної недоступності вузлів.

Список джерел

1. Stoica I., Morris R., Liben-Nowell D. et al. Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networking*. 2003. Vol. 11, No. 1. P. 17–32.

2. DeCandia G., Hastorun D., Jampani M. et al. Dynamo: Amazon's Highly Available Key-value Store. *Proceedings of the 21st ACM Symposium on Operating Systems Principles*. 2007. P. 205–220.

3. Lakshman A., Malik P. Cassandra: A Decentralized Structured Storage System. *ACM SIGOPS Operating Systems Review*. 2010. Vol. 44, No. 2. P. 35–40.

4. Ongaro D., Ousterhout J. In Search of an Understandable Consensus Algorithm. *USENIX Annual Technical Conference*. 2014. P. 305–319.

Жебка Сергій Валентинович
аспірант 3 курсу, групи АІПЗ-31
Державний університет інформаційно-комунікаційних технологій
+380953503572

СЦЕНАРНО-АДАПТИВНИЙ МЕТОД ОЦІНЮВАННЯ СТІЙКОСТІ ВИБОРУ DLT-ПЛАТФОРМ

Постановка задачі. Розвиток технологій розподіленого реєстру супроводжується збільшенням кількості DLT-платформ, які відрізняються архітектурою, механізмами консенсусу, продуктивністю, рівнем децентралізації, способом забезпечення фінальності транзакцій, моделлю довіри та можливостями інтеграції з іншими інформаційними системами. Вибір платформи для конкретного програмного рішення є багатокритеріальною задачею, оскільки технологічна придатність DLT визначається одночасно сукупністю загальних характеристик якості програмного забезпечення та властивостей, специфічних для технологій розподіленого реєстру.

Існуючі підходи до вибору DLT-платформ переважно орієнтовані на порівняння альтернатив за визначеною множиною критеріїв та формування підсумкового рейтингу. Однак значущість критеріїв залежить від сценарію використання. Для корпоративної інформаційної системи пріоритетними можуть бути продуктивність, керованість доступом і визначена фінальність транзакцій, тоді як для відкритої децентралізованої системи більшого значення набувають децентралізація, прозорість, незалежність учасників і стійкість до компрометації окремих вузлів.

За таких умов платформа, яка займає перше місце за одним співвідношенням пріоритетів, може втратити його навіть за відносно невеликої зміни вимог. Тому самого визначення платформи-лідера недостатньо. Необхідно встановити, наскільки стійким є сформований вибір, яка зміна сценарних пріоритетів здатна змінити рішення та яка з альтернатив є найближчим конкурентом поточної платформи.

Мета дослідження полягає у підвищенні обґрунтованості вибору DLT-платформ шляхом розроблення сценарно-адаптивного методу оцінювання стійкості результатів багатокритеріального ранжування за зміни пріоритетів критеріїв.

Результати дослідження. Запропоновано сценарно-адаптивний метод оцінювання стійкості вибору DLT-платформ, особливістю якого є поєднання багатокритеріального оцінювання технологічних альтернатив з подальшим аналізом стійкості отриманого рішення. Метод дозволяє визначати не лише платформу, яка є пріоритетною для поточного сценарію, але й допустиму область зміни вимог, у межах якої сформована рекомендація залишається незмінною.

На початковому етапі формується множина DLT-платформ, які відповідають базовим функціональним та технологічним обмеженням інформаційної системи. Для їх оцінювання запропоновано використовувати дві взаємопов'язані групи критеріїв.

До першої групи належать загальні характеристики якості програмного забезпечення: продуктивність, сумісність, надійність, захищеність, супроводжуваність та адаптивність. Друга група характеризує безпосередньо

властивості DLT-платформи: рівень децентралізації, механізм і надійність консенсусу, фінальність транзакцій, аудитовність, модель довіри, інтероперабельність та особливості використання on-chain і off-chain оброблення.

Для кожної платформи збираються значення відповідних показників та здійснюється їх нормування. Це забезпечує можливість спільного аналізу характеристик, що мають різні одиниці вимірювання та різний напрям оптимізації.

Наступним етапом є формування сценарного профілю. Для кожного сценарію використання визначається пріоритетність критеріїв. У результаті одна й та сама DLT-платформа може отримувати різну оцінку залежно від призначення системи та встановлених вимог.

Після визначення сценарного профілю здійснюється багатокритеріальне оцінювання платформ і формується їх початковий рейтинг. Платформа з найкращою сукупною оцінкою визначається як поточний лідер.

На відміну від статичного ранжування, на цьому робота запропонованого методу не завершується. Для вибраної платформи виконується аналіз області стійкості. Під областю стійкості розуміється така сукупність допустимих змін пріоритетів критеріїв, за якої поточна DLT-платформа продовжує займати перше місце в рейтингу.

У межах цієї області визначається критична зміна сценарного профілю – найменший перерозподіл пріоритетів між критеріями, за якого поточна платформа втрачає позицію лідера. Одночасно встановлюється критичний напрям зміни вимог та платформа-конкурент, яка першою випереджає початково вибрану альтернативу.

Це дозволяє розрізняти формально однакові результати ранжування. Наприклад, дві DLT-платформи можуть мати близькі підсумкові оцінки, проте для однієї з них достатньо незначної зміни пріоритетів для втрати першої позиції, тоді як інша зберігає лідерство в широкому діапазоні сценарних умов.

Для узагальненого оцінювання стійкості враховуються чотири складові: запас до зміни платформи-лідера, частка допустимих змін сценарію, за яких лідер зберігається, стабільність взаємного порядку платформ у рейтингу та запас переваги поточного лідера над найближчими конкурентами.

Послідовність функціонування запропонованого методу наведено на рис. 1.

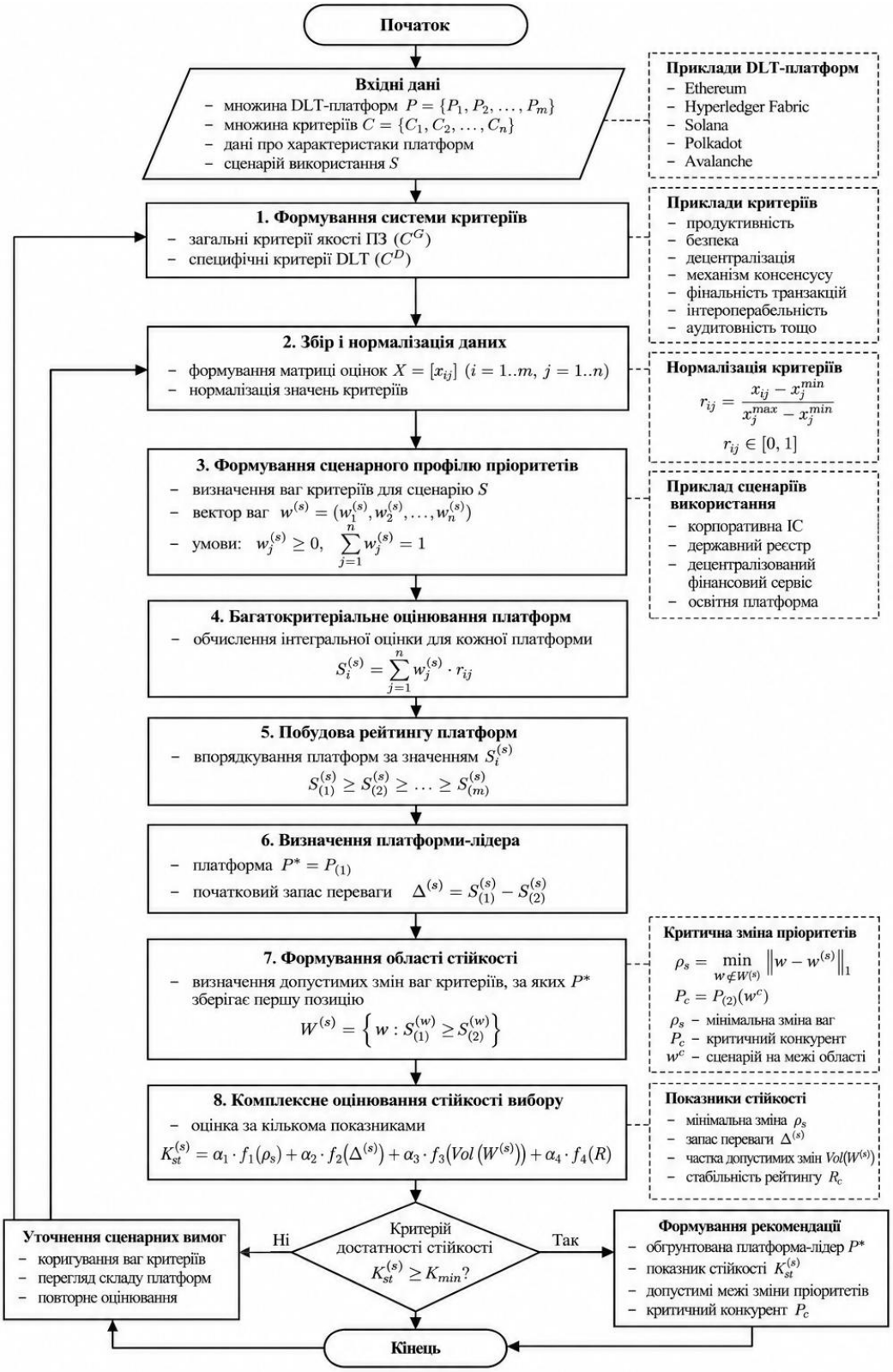


Рис. 1. Блок-схема сценарно-адаптивного методу оцінювання стійкості вибору DLT-платформ

Важливою особливістю запропонованого методу є те, що зміна платформи не ініціюється автоматично за будь-якої зміни ваг критеріїв. Якщо новий сценарний профіль залишається всередині визначеної області стійкості, початкова рекомендація зберігається. Повторний вибір технологічної платформи стає необхідним лише після досягнення критичної межі.

Висновки. Запропоновано сценарно-адаптивний метод оцінювання стійкості вибору DLT-платформ, який поєднує багатокритеріальне ранжування з аналізом чутливості сформованої рекомендації до зміни сценарних пріоритетів. Метод

передбачає спільне врахування загальних характеристик якості програмного забезпечення та DLT-специфічних властивостей, визначення області стійкості рішення, критичної зміни пріоритетів і найближчої конкуруючої альтернативи.

Практичне застосування методу дозволяє отримувати не лише відповідь на питання, яка DLT-платформа є найкращою за поточних вимог, але й оцінювати надійність такого вибору та встановлювати умови, за яких технологічне рішення необхідно переглядати. Це підвищує обґрунтованість вибору DLT-платформ для інформаційних систем, вимоги до яких можуть змінюватися в процесі експлуатації.

Список джерел

1. ISO/IEC 25010:2023. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model.
2. Farshidi S., Jansen S., España S., Verkleij J. Decision Support for Blockchain Platform Selection: Three Industry Case Studies. *IEEE Transactions on Engineering Management*. 2020. Vol. 67, No. 4. P. 1109–1128.
3. Kubler S., Renard M., Ghatpande S., Georges J.-P., Le Traon Y. Decision support system for blockchain (DLT) platform selection based on ITU recommendations: A systematic literature review approach. *Expert Systems with Applications*. 2023. Vol. 211. 118704.
4. Mareschal B. Weight stability intervals in multicriteria decision aid. *European Journal of Operational Research*. 1988. Vol. 33, No. 1. P. 54–64.
5. Triantaphyllou E., Sánchez A. A sensitivity analysis approach for some deterministic multi-criteria decision-making methods. *Decision Sciences*. 1997. Vol. 28, No. 1. P. 151–194.

Напрям 3 Розробка та супровід програмного забезпечення

Грицанюк Дмитро Сергійович,
студент 4 курсу, групи ТЦР-42,
Державний університет інформаційно-комунікаційних технологій
(098) 484-86-44
dhrutsaniuk@gmail.com

Читулян Вадим Олегович
старший викладач кафедри
технологій цифрового розвитку
Державного університету інформаційно-комунікаційних технологій, м. Київ
vadymchytulian@gmail.com

ПРОЄКТУВАННЯ АРХІТЕКТУРИ WEB-ДОДАТКУ ДЛЯ УПРАВЛІННЯ АВТОПАРКОМ ПРИВАТНОГО ПІДПРИЄМСТВА НА ОСНОВІ КЛІЄНТ-СЕРВЕРНОЇ МОДЕЛІ З REST API

Постановка задачі. Управління автопарком приватного підприємства охоплює облік транспортних засобів, водіїв, витрат пального та ремонтів. На практиці більшість малих і середніх підприємств ведуть зазначений облік за допомогою електронних таблиць або паперових журналів. Такий підхід не забезпечує цілісності даних при паралельній роботі кількох операторів, унеможливлює автоматичний розрахунок відхилень витрат пального від норми та потребує значних ручних зусиль під час формування звітності. За результатами аналізу предметної області встановлено, що ручний облік спричиняє від 15 до 25 % помилок у даних, а неефективне управління автопарком призводить до перевитрат транспортного бюджету підприємства. Огляд чотирьох комерційних систем (Fleetio, Wialon, Fleet Stack, AutoFMS) показав, що жодна з них не поєднує повний операційний функціонал, відкритий вихідний код, on-premise розгортання та відсутність ліцензійних витрат одночасно. Виявлена ринкова ніша обґрунтувала доцільність розробки спеціалізованого web-додатку, адаптованого до потреб конкретного підприємства.

Мета дослідження. Спроєктувати архітектуру web-додатку для автоматизації управління автопарком приватного підприємства: визначити структуру клієнт-серверної системи з REST API як міжрівневим контрактом, розробити реляційну модель даних, специфікацію API-ендпоінтів та підсистему безпеки на основі рольового контролю доступу, а також обґрунтувати вибір технологічного стеку через порівняльний аналіз альтернатив.

Результати.

Обґрунтування технологічного стеку. Вибір кожного компонента стеку здійснено на основі порівняльного аналізу. Django обрано над FastAPI завдяки наявності вбудованих ORM-агрегацій, системи міграцій та адмін-панелі,

необхідних для домену управління автопарком. React обрано над Angular через меншу зв’язаність компонентів та нижчий поріг складності для застосунку середнього масштабу. Використання TypeScript замість JavaScript забезпечує строгу типізацію інтерфейсів взаємодії між рівнями, що знижує кількість помилок інтеграції на етапі розробки. PostgreSQL обрано над MongoDB через реляційну природу предметної області з високим рівнем зв’язаності сутностей. Контейнеризація на основі Docker гарантує відтворюваність середовища розгортання незалежно від цільової платформи.

Архітектура системи. Спроєктовано трирівневу клієнт-серверну архітектуру: клієнтський рівень (React + TypeScript, SPA), серверний рівень (Django + Django REST Framework) та рівень даних (PostgreSQL 16). REST API є єдиним міжрівневим контрактом, що забезпечує незалежний розвиток frontend і backend частин. На серверному рівні застосовано Service Layer — шаблон проєктування, що ізолює бізнес-логіку від HTTP-шару, спрощує юніт-тестування та унеможливлює дублювання коду між різними точками входу. Додатково реалізовано Repository Pattern для абстрагування доступу до даних. Специфікація REST API охоплює 27 ендпоінтів, згрупованих у п'ять ресурсних колекцій: автентифікація, користувачі, транспортні засоби, водії та записи витрат пального.

Модель даних. Реляційна модель містить шість сутностей: User, Vehicle, Driver, VehicleAssignment, FuelRecord та MaintenanceRecord, нормалізованих до третьої нормальної форми. Ключовим проєктним рішенням є асоціативна сутність VehicleAssignment, що зберігає повну хронологію закріплення транспортних засобів за водіями з полями date_from та date_to. Це забезпечує коректність ретроспективної звітності та унеможливлює втрату контексту при переназначенні водія. На зовнішні ключі та часто використовувані поля фільтрації накладено індекси PostgreSQL для забезпечення часу відгуку API на рівні 95-го перцентиля не більше 300 мс.

Підсистема безпеки. Контроль доступу реалізовано на основі дворівневої RBAC-моделі з чотирма ролями: Адмін, Менеджер, Диспетчер та Водій. Матриця доступу визначає дозволені HTTP-методи для кожної ролі в розрізі кожного ресурсу. Автентифікація здійснюється через JWT-токени (access + refresh) з алгоритмом підпису HS256. Access-токен має скорочений термін дії (15 хвилин), refresh-токен зберігається в HttpOnly cookie для унеможливлення його перехоплення через XSS-атаки. Реалізовано механізм інвалідації refresh-токенів через чорний список.

У таблиці 1 наведено порівняльний аналіз проєктних рішень із альтернативами.

Таблиця 1. Порівняльний аналіз ключових проєктних рішень

Компонент	**Обрана технологія**	**Обґрунтування вибору**
Backend-фреймворк	Django + DRF	Вбудовані ORM-агрегації, міграції, адмін-панель

Frontend-бібліотека	React + TypeScript	Слабка зв'язаність компонентів, строга типізація API
СУБД	PostgreSQL 16	Реляційна природа предметної області, ACID
Контейнеризація	Docker + Docker Compose	Відтворюваність середовища, запуск трьома командами
Автентифікація	JWT (HS256)	Stateless API, підтримка мобільного клієнта

Висновки. Спроєктована архітектура web-додатку для управління автопарком реалізує трирівневу клієнт-серверну модель з REST API як єдиним міжрівневим контрактом. Ключовими проєктними рішеннями є: застосування Service Layer для ізоляції бізнес-логіки, асоціативна сутність VehicleAssignment для коректного ведення хронології закріплення транспортних засобів та дворівнева RBAC-модель з JWT-автентифікацією для гнучкого контролю доступу. Обраний технологічний стек (Django, React, TypeScript, PostgreSQL, Docker) обґрунтовано через порівняльний аналіз альтернатив, а не довільний вибір. Спроєктована система забезпечує масштабованість, підтримуваність та незалежний розвиток рівнів застосунку, результати практичної реалізації якого представлено в окремій публікації.

Список використаних джерел:

1. Django Software Foundation. Django documentation. URL: https://docs.djangoproject.com/en/6.0/
2. Django REST Framework. URL: https://www.django-rest-framework.org/
3. Meta Open Source. React documentation. URL: https://react.dev/
4. Docker Inc. Docker Compose documentation. URL: https://docs.docker.com/compose/
5. PostgreSQL Global Development Group. PostgreSQL 16 Documentation. URL: https://www.postgresql.org/docs/16/

Колпаков Дмитро Олегович
студент 4 курсу, групи ТЦР-41
Державний університет інформаційно-комунікаційних технологій
+380666222623
cristalaiz1337@gmail.com

ВЕБ-ПЛАТФОРМА БРОНЮВАННЯ КОВОРКІНГ-ПРОСТОРІВ ІЗ ПІДТРИМКОЮ КОМАНДНОЇ РОБОТИ НА ОСНОВІ JAVASCRIPT-ТЕХНОЛОГІЙ

Постановка задачі. У більшості коворкінгів бронювання робочих місць досі часто організовується через телефон або повідомлення в месенджерах. Такий спосіб незручний для контролю завантаженості приміщень, підвищує ймовірність накладок у бронюваннях і не дає змоги оперативно побачити доступні місця. Окремою проблемою є групове бронювання, коли кільком користувачам потрібно одночасно зарезервувати місця в одній зоні.

Мета дослідження. Мета роботи полягає у створенні та впровадженні веб-платформи для бронювання коворкінг-просторів. Рішення має автоматизувати процес резервування місць, спростити взаємодію між учасниками команди та надати власникам зручні засоби керування просторами, зонами і персоналом.

Результати. У межах дипломного проєкту розроблено веб-платформу, що містить клієнтську частину, серверний API та спільний пакет типів і допоміжних модулів. Така побудова системи забезпечила однакове трактування моделей даних на фронтенді та бекенді.

Технологічний стек підбирався з урахуванням практичних потреб проєкту. Клієнтська частина реалізована на React з TypeScript, збірка виконується через Vite, а оформлення інтерфейсу — за допомогою Tailwind CSS. Для взаємодії із сервером використано TanStack Query, що спрощує кешування даних і в більшості випадків дає змогу обійтися без Redux. Серверна частина побудована на Node.js, Express і Prisma поверх PostgreSQL 16. Робота в реальному часі реалізована через Socket.IO, що важливо для чату та відображення присутності користувачів. Файли зберігаються в MinIO, а для кешування застосовується Redis 7 [1].

У платформі реалізовано модуль автентифікації, який охоплює реєстрацію, вхід до облікового запису, двофакторну перевірку за TOTP-кодами, оновлення токенів доступу та відновлення пароля через електронну пошту. Це підсилює захист персональних і платіжних даних користувачів.

За функціональними можливостями платформа охоплює повний цикл роботи. Користувач може знайти коворкінг за потрібними фільтрами, переглянути інформацію про простір і доступні часові слоти, після чого оформити разове або повторюване бронювання. За відсутності вільних місць можна приєднатися до черги очікування. Для команд передбачено спільні акаунти та бронювання кількох зон в межах однієї операції. Після відвідування користувач може залишити відгук. Додатково система надсилає email- і push-сповіщення про зміну статусу, а власник через адміністративну панель переглядає статистику, керує зонами, персоналом і тарифами. Усі ключові дії фіксуються в журналі аудиту.



Рис. 1 - Схема процесу бронювання коворкінг-простору

Система може розгортатися локально однією командою, оскільки інфраструктура керується через Docker Compose. Процеси безперервної інтеграції та розгортання налаштовуються за допомогою GitHub Actions, тому кожна зміна в коді запускає перевірки, збірку та тести [2]. Основні сценарії, зокрема реєстрація, бронювання і керування командним доступом, покриваються наскрізними тестами на Playwright. Для модульного тестування використовується Vitest [3].

Висновки. У результаті виконання дипломного проєкту створено веб-платформу для бронювання коворкінг-просторів, яка поєднує інструменти для користувачів і засоби адміністрування. Запропоноване рішення зменшує кількість ручних операцій під час обробки бронювань, автоматизує виявлення конфліктів та підтримує командну взаємодію. До важливих складових платформи належать рольова модель доступу, журнал аудиту, система сповіщень і механізми захисту облікових записів.

Подальший розвиток платформи доцільно пов’язати зі створенням мобільного застосунку, підключенням платіжних сервісів, реалізацією пошуку коворкінгів на мапі, покращенням рекомендаційного модуля та розширенням аналітичних інструментів для власників просторів.

Список використаних джерел

1. OWASP Foundation. OWASP Application Security Verification Standard 4.0. 2021. URL: https://owasp.org/www-project-application-security-verification-standard/

2. Kim G. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press, 2021. 480 p.
3. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002. 560 p.
4. Simpson K. You Don't Know JS Yet: Get Started. O'Reilly Media, 2020. 143 p.
5. Richardson C. Microservices Patterns: With examples in Java. Manning Publications, 2018. 520 p.

Ляшенко Ярослав Володимирович
студент 4 курсу, групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій
(063) 503-98-99
liashenkoya@gmail.com

РОЗРОБКА ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ПРИКЛАДІ 2D ГРИ ChronoVortex

Постановка задачі. Під час створення 2D-платформера з елементами головоломок та маніпуляцією часом у середовищі Unity швидко виявляється, що основні труднощі полягають не в реалізації базових механік пересування, а в організації архітектури коду. Найпоширенішою проблемою є концентрація всієї логіки в одному великому класі *PlayerController*, що згодом призводить до значних труднощів під час розширення функціональності або виправлення помилок. У студентських та невеликих інді-проєктах саме через це виникають хаотичні залежності, ланцюгові помилки та суттєве ускладнення супроводу [1].

Класична каскадна модель (Waterfall) у геймдеві виявляється малоефективною через ітеративний характер розробки: нові механіки головоломок та часові режими з'являються поступово, а зміни в одному компоненті часто спричиняють каскадні модифікації в інших. Питання супроводу ПЗ, зокрема портування на інші платформи чи додавання локалізації, часто залишається поза увагою до моменту виникнення потреби. У результаті код функціонує, але його подальша підтримка стає ресурсоємною та складною.

У межах дослідження було реалізовано в грі *ChronoVortex* комбіновану механіку подорожей у часі (перемикання режимів та парна телепортація) таким чином, щоб забезпечити її легке розширення без суттєвого переписування вже існуючого коду, а також зберегти високу зрозумілість і підтримуваність системи впродовж тривалого часу.

Мета дослідження. Мета полягала в практичній перевірці ефективності принципів дисципліни «Розробка та супровід програмного забезпечення» (SDLC, SOLID, розділення відповідальностей) саме в умовах геймдеву, де вимоги постійно еволюціонують.

Особливу увагу було приділено оцінці доцільності застосування State Machine та винесення механік у окремі компоненти для уникнення типової проблеми розростання *PlayerController* до неприйнятних розмірів. Також досліджувалося, наскільки такий підхід забезпечує майбутній супровід проєкту — можливість додавання нових часових режимів або головоломок без внесення змін у вже стабільні частини коду [2].

Результати. На практиці застосування структурованого підходу дало значно кращі результати порівняно з прямим («в лоб») програмуванням.
Початкові спроби розмістити всю логіку часу всередині *PlayerController* підтвердили прогнозовані проблеми: клас швидко розростався, тестування ускладнювалося, а зміни в одному режимі впливали на інші. Після проведення рефакторингу було виділено дві незалежні системи:

- *ModeActivator* — відповідає виключно за перемикання часових режимів (клавіші 1 та 2). Компонент лише активує/деактивує відповідні об'єкти на рівні та оновлює інтерфейс користувача.

- *PlayerTeleportSystem* — виконує пошук найближчої точки телепортації з невеликим offset'ом для запобігання застряганню персонажа в геометрії рівня.

Обидві системи взаємодіють через події та не мають прямих залежностей одна від одної. Така архітектура дозволила гнучко комбінувати механіки в головоломках: активація важеля в одному режимі, перемикання в інший та подальша телепортація.

Для керування поведінкою персонажа було впроваджено повноцінну State Machine (стани *Idle*, *Walk*, *Jump* тощо). Кожен стан реалізовано як окремий клас, що задовольняє інтерфейс *IPlayerState*. Перемикання станів здійснюється через *PlayerStateMachine*. Такий підхід, хоча й вимагає певних початкових зусиль, суттєво спрощує додавання нових станів (наприклад, «зависання на важелі») без впливу на вже реалізовані механіки анімації чи пересування [1].

Під час тестування (28 тестових сценаріїв) було виявлено та усунуто низку специфічних проблем: некоректний респавн після смерті в зміненому часовому режимі (вирішено через метод *FindSafeGroundPosition* із додатковою перевіркою по *FixPlatform*), а також конфлікти при швидкому послідовному перемиканні режимів і телепортації (вирішено за допомогою короткої Coroutine).

Продуктивність проєкту на середньостатистичному ПК стабільно утримується на рівні 60 FPS навіть за наявності 20–25 активних об'єктів. Розмір білду після стиснення становить 38 МБ. Водночас для чистого pixel-art 2D-проєкту Godot міг би забезпечити менший розмір і вищу нативність [3], проте Unity виявився оптимальним рішенням з огляду на швидкість розробки та рівень наявних компетенцій [4].

Висновки. Застосований підхід підтвердив свою ефективність для невеликих і середніх проєктів. Чітке розділення систем і дотримання принципів SOLID дозволили забезпечити високу гнучкість та підтримуваність коду навіть на завершальних етапах розробки. Основний висновок полягає в тому, що винесення механік в окремі класи та компоненти з самого початку значно полегшує подальші зміни та розширення функціональності.

Для повноцінної комерційної розробки рекомендується додатково впровадити Git + Git LFS, CI/CD-пайплайни для автоматизованих білдів та використання ScriptableObjects для конфігурації рівнів. Отримані результати демонструють, що грамотна архітектура ПЗ у геймдеві дозволяє створювати не лише функціональні, але й зручні для довгострокового супроводу рішення [2], [5].

Список використаних джерел:

1. Unity Technologies. Official Unity Documentation – 2D. URL: https://docs.unity3d.com/Manual/Unity2D.html (дата звернення: 12.04.2026).
2. Sommerville I. Software Engineering. 10th Global Edition. Harlow: Pearson, 2016. 816 p. URL: https://dn790001.ca.archive.org/0/items/bme-vik-konyvek/Software%20Engineering%20-%20Ian%20Sommerville.pdf (дата звернення: 15.04.2026).
3. Generalist Programmer. Best Game Engines 2025: Unity vs Godot vs Unreal vs GameMaker. 2025. URL: https://generalistprogrammer.com/comparisons/game-development-engines-2025 (дата звернення: 12.04.2026).
4. Матеріали III Міжнародної науково-практичної конференції «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії». Київ: ДУІКТ, 2025.
5. JetBrains. Game Dev in 2025: Excerpts From the State of Game Development Report. 2026. URL: https://blog.jetbrains.com/dotnet/2026/01/29/game-dev-in-2025-excerpts-from-the-state-of-game-development-report/ (дата звернення: 14.04.2026).

Мезецький Владислав Олегович
студент 4 курсу, групи ТЦР-43
спеціальність 121 «Інженерія програмного забезпечення»,
(096) 855-45-40
mezetskiy.2004@gmail.com

Чернявський Ждан Анатолійович,
Старший викладач кафедри
Технологій цифрового розвитку,
Державного університету інформаційно-комунікаційних технологій, м. Київ
z.cherniavskyi@duikt.edu.ua

АВТОМАТИЗАЦІЯ ДОКУМЕНТООБІГУ ФІНАНСОВОЇ КОМПАНІЇ

Постановка задачі. Фінансові компанії, що здійснюють обмін валют, орендують приміщення під відділення та зобов'язані оформлювати відповідні договори. Традиційний підхід до цього процесу передбачав ручне внесення даних про орендаря та об'єкт оренди до таблиці Microsoft Excel, а потім ручне перенесення цих відомостей до шаблону договору у Microsoft Word. Такий підхід спричиняв значні часові втрати, дублювання даних і систематичні помилки при копіюванні. Актуальним є розроблення вебплатформи, яка автоматизує цей процес, зберігає дані структуровано та генерує договори автоматично.

Мета дослідження. Метою роботи є проектування та реалізація клієнт-серверної вебплатформи для автоматизації процесів оренди приміщень фінансової компанії, яка усуває необхідність ручного перенесення даних між застосунками за рахунок одночасного збереження інформації у базі даних і автоматичної генерації Word-документів за допомогою фабричного методу.

Результати дослідження. Для реалізації системи обрано стек технологій, що охоплює серверну частину на базі ASP.NET Core Web API (.NET 10) та клієнтську частину на основі React/Next.js. Серверна частина реалізує RESTful API з JWT-авторизацією, бізнес-логіку обробки договорів та модуль генерації документів. Клієнтська частина забезпечує зручний інтерфейс для введення й перегляду даних про орендарів, об'єкти оренди та умови договорів. Для зберігання інформації використовується реляційна база даних Postgresql у зв'язці з Entity Framework Core.

Ключовим елементом архітектури є модуль генерації договорів. Реалізовано патерн «Фабричний метод» (Factory Method): залежно від типу договору фабрика обирає відповідний будівельник документа, який підставляє актуальні дані з бази у заздалегідь підготовлений шаблон. Шаблон договору містить набір елементів керування вмістом (Content Controls) — структурованих XML-елементів формату .docx, що мають унікальні теги. Користувач може самостійно розмістити Content Control у будь-якому місці шаблону та задати йому довільний тег засобами Microsoft Word, після чого система автоматично знаходить елемент за цим тегом та підставляє відповідне значення. Для маніпуляцій із .docx-файлами застосовано бібліотеку Open XML SDK, яка не потребує встановлення Microsoft Office на сервері.

Функціональність платформи охоплює: реєстрацію юридичних осіб-орендарів, облік об'єктів оренди (адреса, площа, характеристики), формування та

підписання договорів оренди. Дані, введені через вебформу, одночасно зберігаються у структурованій таблиці бази даних і передаються фабриці документів — що виключає будь-яке дублювання введення.

Апробація системи показала скорочення часу підготовки одного договору оренди з 15–20 хвилин до 2–3 хвилин. Виключення ручного перенесення даних між застосунками знизило кількість помилок у договорах до нуля. Централізоване зберігання в базі даних дозволяє формувати аналітичні звіти та контролювати строки дії договорів.

Висновки та перспективи. Розроблена клієнт-серверна вебплатформа на базі .NET та React/Next.js вирішує актуальну практичну задачу автоматизації документообігу фінансової компанії. Застосування патерну «Фабричний метод» для генерації Word-документів та одночасне збереження даних у реляційній базі усуває ручне дублювання інформації, підвищує швидкість і точність підготовки орендних договорів. Перспективою розвитку є додавання модуля електронного підпису та інтеграції з реєстрами нерухомості.

Список використаних джерел:

1. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002. 560 p.

2. Microsoft. Open XML SDK documentation. URL: https://learn.microsoft.com/en-us/office/open-xml/open-xml-sdk (дата звернення: 10.04.2026).

3. Wieruch R. The Road to React. Leanpub, 2023. 312 p.

4. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 p.

Оводов Ростислав Олексійович
студент 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
(099) 429-52-38
st8053117@stud.duikt.edu.ua
Науковий керівник:
Гавор Артур Станіславович
викладач кафедри ТЦР
a.havor@duikt.edu.ua

ПРОЄКТУВАННЯ ТА ТЕХНІЧНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ПРОДУКТИВНІСТЮ З ЕЛЕМЕНТАМИ ГЕЙМІФІКАЦІЇ

Постановка задачі. Розробка сучасних систем персонального менеджменту потребує впровадження механізмів, що підвищують внутрішню мотивацію користувачів. Актуальним завданням є побудова стійкої архітектури, здатної обробляти динамічні ігрові метрики (бали досвіду, рівні) та забезпечувати надійне збереження даних при розгортанні на локальних апаратних потужностях [2].

Мета дослідження. Обґрунтування вибору архітектурних рішень та опис процесу технічної реалізації веб-застосунку на базі стеку React 19 та Express 5 для автоматизації персонального планування з елементами гейміфікації.

Результати дослідження. Архітектура системи базується на триланковій моделі (3-tier architecture), що забезпечує чіткий поділ між інтерфейсом користувача, бізнес-логікою та рівнем збереження даних.

Для реалізації функціоналу обрано наступні технологічні рішення (табл. 1):

Таблиця 1.

Технологічний стек розробленої системи

Компонент	Технологія	Обґрунтування вибору
Frontend	React 19, Tailwind CSS v4	Висока швидкість рендерингу, підтримка Glassmorphism.
Backend	Node.js, Express 5	Асинхронна обробка запитів, підтримка REST API.
База даних	PostgreSQL 15	Реляційна модель для збереження структури завдань.
Безпека	JWT, bcryptjs	Забезпечення захисту сесій та хешування паролів.

Програмна реалізація бізнес-логіки зосереджена на серверній частині (Express 5). При завершенні завдання клієнт (React 19) надсилає запит до API, де інтелектуальний модуль аналізує параметри за Матрицею Ейзенхауера та автоматично нараховує відповідну кількість досвіду (XP) у базу даних PostgreSQL. Для візуального підкріплення досягнень на стороні клієнта використано бібліотеку canvas-confetti.

Для гарантії стабільності алгоритмів впроваджено автоматизоване тестування: Jest для перевірки логіки API та Vitest для компонентів інтерфейсу [3]. Весь проект контейнеризовано за допомогою Docker Compose, що дозволяє стабільно запускати систему на локальних серверах типу ThinkCentre M715q під керуванням Linux.

Висновки. В результаті дослідження спроєктовано та реалізовано веб-застосунок, що поєднує методику Ейзенхауера з RPG-елементами. Використання сучасного стеку дозволило створити продуктивне рішення з високим ступенем візуального відгуку, що сприяє підвищенню довгострокової мотивації користувача.

Список використаних джерел

1. Дібрівний О. А., Гребенюк В. В. Вступ до об’єктно орієнтованого програмування C#: Навчальний посібник. Київ: Державний університет телекомунікацій, 2018. 190 с.
2. JSON Web Token Introduction. JWT.io. URL: https://jwt.io/introduction.
3. Node.js v20.x Documentation. Node.js. URL: https://nodejs.org/docs/latest-v20.x/api/.
4. React v19 Release Note. React. URL: https://react.dev/blog/.

Подольський Дмитро Олегович
студент 4 курсу, групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій
(093) 946-28-92
podolskijdmitro1@gmail.com

РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ КОРПОРАТИВНОГО УПРАВЛІННЯ ПРОЄКТАМИ ТА РЕСУРСАМИ З ІНТЕГРАЦІЄЮ ФІНАНСОВОГО МЕНЕДЖМЕНТУ

Постановка задачі. Малий та середній бізнес в Україні стикається з проблемою фрагментації інструментів управління: для ведення проєктів використовується один сервіс, для фінансів — інший, для складського обліку — третій. Дані розкидані по різних платформах, що призводить до дублювання інформації, втрати часу на ручну синхронізацію та підвищеного ризику помилок. Існуючі рішення (Asana, Monday.com, Wrike) зосереджені на трекінгу задач і не мають вбудованого фінансового обліку, складського управління чи повноцінного внутрішнього чату. Повнофункціональні ERP-системи, такі як SAP або Odoo, є надмірно складними та дорогими для невеликих команд.

Мета дослідження. Метою дослідження є розробка веб-додатку Ri Dren для корпоративного управління проєктами та ресурсами з інтеграцією фінансового менеджменту, який об'єднує в єдиному інтерфейсі модулі управління проєктами, фінансами, складом, продажами, закупівлями та внутрішньою комунікацією.

Результати дослідження. У ході дослідження було спроєктовано та реалізовано клієнт-серверний веб-додаток на основі REST API. Серверна частина побудована на Flask (Python) із використанням Flask-RESTful для побудови API, SQLAlchemy ORM для роботи з базою даних PostgreSQL, Flask-JWT-Extended для автентифікації через JSON Web Tokens та Flask-Mail для email-сповіщень. Клієнтська частина реалізована на React із TypeScript, Redux Toolkit для управління станом та React Router для маршрутизації.

Система охоплює сім бізнес-модулів: управління проєктами (пайплайни зі стадіями, проєкти, задачі з підзадачами, time tracker), фінансовий менеджмент (банківські рахунки, транзакції, бюджети з контролем витрат, рахунки-фактури), складський облік (склади, інвентарні позиції, переміщення, відвантаження), продажі (клієнти, комерційні пропозиції, замовлення), закупівлі (постачальники, замовлення на закупку), управління персоналом (ієрархія співробітників, рольова модель доступу з 50+ функціями) та комунікація (внутрішній чат, система сповіщень in-app та email).

Особливістю системи є гнучка рольова модель, реалізована через таблицю Navbar, де для кожного співробітника зберігається набір дозволених функцій (наприклад, "add_accounts", "delete_warehouses", "sales_statistics"). Це дозволяє налаштувати доступ на рівні окремих операцій, а не лише на рівні ролей чи модулів, що є суттєвою перевагою порівняно з аналогами [1].

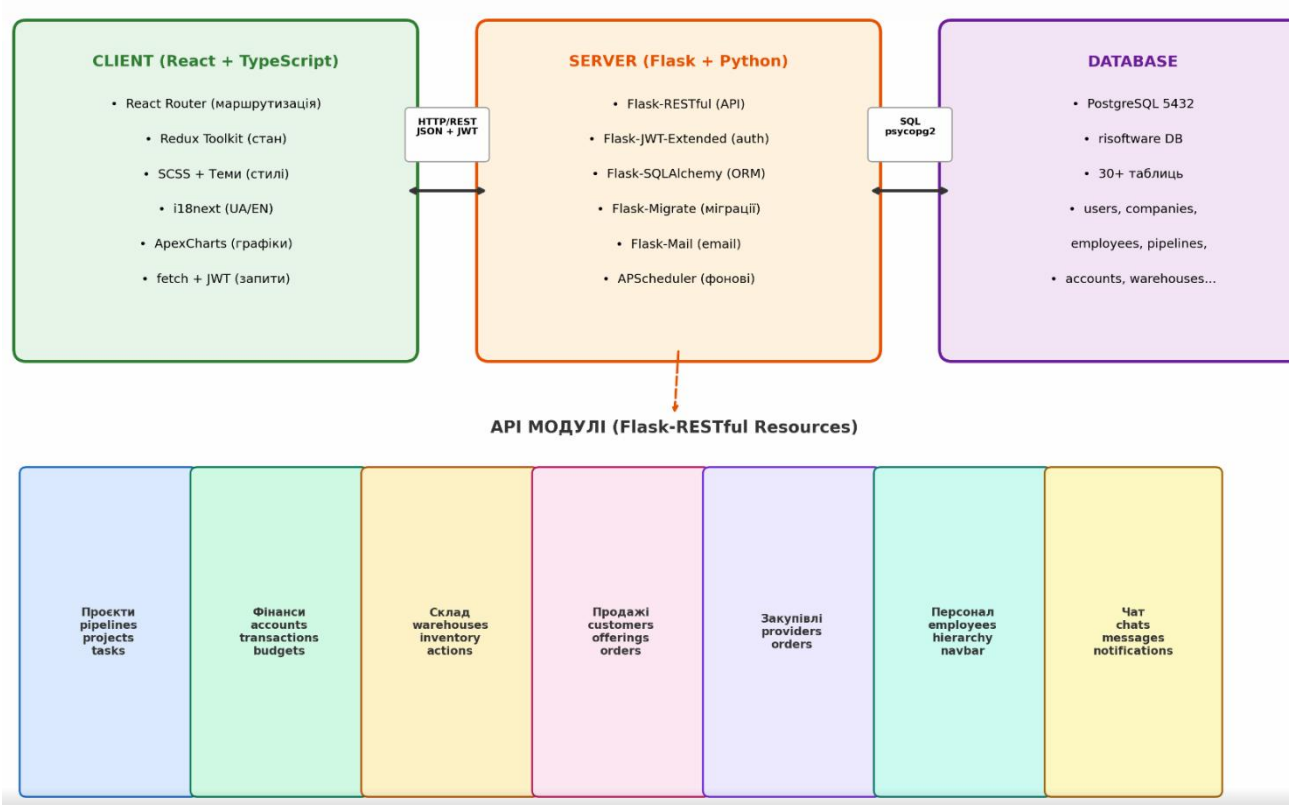


Рис. 1. Архітектура клієнт-серверної взаємодії web-застосунку.

Для забезпечення безпеки реалізовано хешування паролів через sha256, JWT-автентифікацію з access-токеном (120 хвилин) та refresh-токеном (1-30 днів), а також двоетапну верифікацію email через OTP-код. Фонові процеси (відправка email-сповіщень, перевірка дедлайнів задач) реалізовано через APScheduler [2].

Модель даних включає понад 30 таблиць у PostgreSQL, пов'язаних через ForeignKey та relationship. Для управління міграціями використовується Flask-Migrate на базі Alembic. Клієнтський інтерфейс підтримує локалізацію (українська та англійська мови) через i18next, статистичні дашборди на ApexCharts та світлу/темну теми [3].

Тестування системи включало функціональне тестування API через Postman, модульне тестування серверної логіки (34 кейси для модуля проєктів, 27 кейсів для фінансового модуля) та кросбраузерне тестування у Chrome, Firefox, Opera та Edge [4].

Висновки та перспективи. Розроблений веб-додаток Ri Dren дозволяє малим та середнім компаніям вести проєкти, фінанси, складський облік, продажі та закупівлі в єдиному інтерфейсі з вбудованим чатом та гнучкою рольовою моделлю. Система може бути розгорнута на власному сервері без щомісячної абонплати за користувачів, що робить її доступною альтернативою існуючим хмарним платформам. Перспективою подальшого розвитку є інтеграція мобільного клієнта та модуля аналітики на базі ML для прогнозування бюджетів та складських потреб.

Список використаних джерел

1. Raji M. A., Olodo H. B., Oke T. T. et al. E-commerce and consumer behavior: A review of AI-powered personalization and market trends. GSC Advanced Research and Reviews. 2024. Vol. 18(3). P. 66–77.

2. Flask Documentation. https://flask.palletsprojects.com/ (дата звернення: 17.05.2025).
3. React Documentation. https://react.dev/ (дата звернення: 18.05.2025).
4. Postman Documentation. https://learning.postman.com/docs/ (дата звернення: 20.05.2025).
5. PostgreSQL Documentation. https://www.postgresql.org/docs/ (дата звернення: 19.05.2025).

Старцев Микола Сергійович,
студент 4 курсу, групи ТЦР-42,
Державний університет інформаційно-комунікаційних технологій
(050) 850-35-01
nikolastartsev@gmail.com

Читулян Вадим Олегович
старший викладач кафедри
Технологій цифрового розвитку
Державного університету інформаційно-комунікаційних технологій, м. Київ
vadymchytulian@gmail.com

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ПРОЕКТУВАННЯ МОБІЛЬНОЇ СИСТЕМИ ВИЯВЛЕННЯ ВИБУХОНЕБЕЗПЕЧНИХ ПРЕДМЕТІВ ІЗ ЗАСТОСУВАННЯМ БПЛА

Постановка задачі. За даними HALO Trust, понад 23% території України потенційно забруднено вибухонебезпечними предметами (ВНП) [1]. Традиційне ручне розмінування є надзвичайно повільним і небезпечним. Застосування БПЛА з системами комп'ютерного зору дозволяє обстежувати місцевість у 10-20 разів швидше без ризику для саперів. Жоден з комерційно доступних застосунків для керування БПЛА не містить модуля виявлення ВНП - цю задачу вирішує даний проект.

Мета дослідження. Провести порівняльний аналіз існуючих програмних рішень для керування БПЛА у контексті задачі виявлення ВНП, визначити їх ключові недоліки та спроектувати архітектуру мобільного застосунку MineDetector з локальним AI-інференсом на основі DJI Mobile SDK v4 [2] та YOLOv8n [3].

Результати дослідження. Аналіз існуючих рішень. Розглянуто три комерційні застосунки. DJI Pilot 2 - професійний застосунок для промислових БПЛА DJI, орієнтований на зйомку та картографування. Autel Explorer- американський застосунок з вбудованим AI-детектором для розпізнавання людей, транспортних засобів та техніки [4], без підтримки ВНП. Parrot ANAFI Ai - французький промисловий БПЛА з відкритим SDK для інспекції об'єктів. Жодне рішення не підтримує виявлення ВНП. Порівняльна характеристика наведена у табл. 1.

Таблиця 1. Порівняльний аналіз існуючих рішень та MineDetector

Критерій	**DJI Pilot 2**	**Autel Explorer**	**Parrot ANAFI Ai**	**MineDetector (план)**
Цільова задача	Промислова та комерційна зйомка	Спостереження, розпізнавання об'єктів	Промисловий огляд	Виявлення ВНП
Вбудований AI-детектор	Відсутній	Так (люди, авто, техніка)	Відсутній	Так (6 класів ВНП)

Критерій	DJI Pilot 2	Autel Explorer	Parrot ANAFI Ai	MineDetector (план)
Виявлення ВНП	Ні	Ні	Ні	Так
Робота без інтернету	Частково	Частково	Частково	Повна
AI-інференс на пристрої	Ні	Так (хмара + пристрій)	Ні	Так (TFLite)
SDK інтеграція з DJI	Так	Ні	Ні	Так (SDK v4)

Вимоги до застосунку. На основі аналізу було визначено вимоги до MineDetector. Інтеграція з БПЛА DJI Mavic 2 через DJI Mobile SDK v4, отримання відеопотоку H.264/H.265 у реальному часі, локальний AI-інференс без хмарних сервісів, детекція 6 класів ВНП - MON-50, PFM-1 (Пелюстка), PMN-1, PMN-2, POM-3, TM-62, відображення положення дрона на карті під час польоту, тестовий режим перевірки детектора на статичних фото без підключення дрона.

Запланована архітектура, застосунок розробляється на Android (Kotlin) з MVVM-архітектурою. Для детекції обрано YOLOv8n [4] - найкомпактнішу версію YOLO у форматі TFLite float16 для локального виконання з прискоренням через GPU Delegate. Структура архітектури наведена у табл. 2.

Таблиця 2. Запланована архітектура застосунку MineDetector

Рівень	Функція	Технологія
Апаратний	БПЛА: відеопотік 4K, GPS, телеметрія	DJI Mavic 2, OcuSync 2.0
SDK	Управління дроном, отримання телеметрії	DJI Mobile SDK v4 [3]
AI / ML	Детекція ВНП у реальному часі	YOLOv8n TFLite float16 [4]
Дані / Карта	Відображення положення дрону	Mapbox [5]
UI	FPV-відео з bounding box, тестовий режим	Android (Kotlin), MVVM

Підготовка даних. Для навчання моделі планується зробити власний датасет з аеро знімків вибухонебезпечних предметів, зроблених з допомогою БПЛА на висоті 5-20 м під різними кутами. Розмітка об’єктів буде виконуватися в CVAT [6]. Для зменшення кількості хибних спрацювань, до датасету також буде додано близько 40% фонових зображень, на яких ВНП відсутні.

Висновки. Проведений аналіз показав, що серед розглянутих додатків немає такого, яке дає змогу виявляти ВНП із використанням БПЛА. Перевагою MineDetector є інтеграція з DJI Mavic 2 через офіційний SDK v4, а також робота моделі YOLOv8n без підключення до інтернету. Розроблена архітектура та визначені вимоги дають змогу перейти до практичної реалізації системи.

Список використаних джерел:

1. Landmine and Cluster Munition Monitor: https://the-monitor.org/country-profile/ukraine/impact
2. DJI Developer (Mobile SDK): https://developer.dji.com/mobile-sdk/documentation/introduction/sdk_architectural_overview.html
3. Ultralytics (YOLOv8): https://docs.ultralytics.com/models/yolov8/
4. Autel Robotics (Autel Explorer): https://www.autelrobotics.com/news/919/
5. Mapbox (Maps SDK for Android): https://docs.mapbox.com/android/maps/guides/install/
6. CVAT: https://docs.cvat.ai/docs/administration/community/basics/installation/

Грисик Владислав Олександрович
студент 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
(067) 603-39-65
vladgrisik@gmail.com

РОЗВИТОК КОРИСТУВАЧІВ У ЖАНРІ RUNNER

Постановка задачі. Мобільні ігри жанру runner є одними з найпоширеніших у світі: сукупна кількість завантажень таких застосунків, як Subway Surfers та Temple Run, перевищує кілька мільярдів. Проте більшість досліджень у галузі мобільного геймінгу зосереджені на комерційних аспектах — монетизації, утриманні аудиторії, маркетингу — тоді як питання розвитку когнітивних і поведінкових якостей користувача в процесі гри залишається недостатньо вивченим. Актуальність теми зумовлена зростаючим інтересом до концепції edutainment — поєднання розваги та навчання — та потребою в науковому обґрунтуванні розвивального потенціалу ігор жанру runner.

Мета дослідження. Метою роботи є аналіз впливу ігор жанру runner на розвиток когнітивних, психомоторних та навчальних якостей користувачів, а також визначення умов, за яких цей вплив може бути цілеспрямовано посилений через інтеграцію пізнавального контенту в ігровий процес.

Результати дослідження. Аналіз наукової літератури та практики розробки мобільних ігор дозволив виявити три ключових напрямки впливу runner-ігор на розвиток користувача: когнітивний, психомоторний та мотиваційно-освітній.

Когнітивний розвиток є найбільш вивченим аспектом впливу runner-ігор. Механіка жанру вимагає від гравця постійної концентрації уваги, швидкого розпізнавання образів і миттєвого прийняття рішень — адже перешкоди з'являються без попередження, а час реакції обмежений частками секунди. Дослідження у галузі когнітивної психології підтверджують, що регулярна ігрова практика позитивно впливає на такі показники, як час зорово-моторної реакції, обсяг робочої пам'яті та здатність до розподілу уваги між кількома об'єктами одночасно [1]. Динамічне наростання швидкості та складності в runner-іграх є аналогом методу прогресивного навантаження у спортивному тренуванні: мозок поступово адаптується до вищого темпу опрацювання інформації.

Психомоторний розвиток пов'язаний з координацією «зір — рука — дія». На сенсорних екранах мобільних пристроїв гравець виконує точні тап-жести та свайпи у відповідь на візуальні стимули. Відповідно до теорії сенсомоторного навчання, повторювана практика таких взаємодій формує стійкі нейронні патерни, що прискорюють виконання аналогічних рухів у майбутньому [2]. Для молодшої аудиторії (6–12 років) ігри з точним сенсорним вводом є ефективним засобом розвитку дрібної моторики.

Мотиваційно-освітній потенціал runner-ігор зумовлений їх здатністю утримувати увагу гравця у стані так званого «потоку» (flow state) — психологічного стану повного занурення в діяльність, при якому засвоєння нової інформації відбувається природно і без вольових зусиль [3]. Саме цей принцип лежить в основі edutainment-підходу: якщо освітній контент з'являється в момент, коли гравець перебуває у стані потоку, ймовірність його запам'ятовування суттєво зростає порівняно з традиційними методами подачі матеріалу.

На основі проведеного аналізу визначено умови ефективної інтеграції освітнього контенту в runner-гру, які систематизовано в таблиці 1.

Таблиця 1

Умови ефективної інтеграції освітнього контенту в runner-гру

Умова	**Обґрунтування**	**Практична реалізація**
Ненав'язливість контенту	Не переривати стан потоку гравця	Контент відображається між ігровими циклами або у паузах
Стислість повідомлення	Робоча пам'ять утримує 7±2 одиниці інформації	Текст факту до 150 символів, одна ключова думка
Тематична відповідність	Контекстна пов'язаність підвищує засвоєння	Факти відповідають тематиці персонажів та світу гри
Візуальний супровід	Образна пам'ять ефективніша за вербальну	Ілюстрація поряд з текстом факту
Варіативність контенту	Повторення знижує залученість	Алгоритм без повторень протягом сесії

Окремою складовою розвитку є системи прогресії та досягнень, притаманні сучасним runner-іграм. Механіки розблокування нових персонажів, збору внутрішньоігрової валюти та таблиці рекордів апелюють до внутрішньої мотивації гравця і формують такі якості, як цілепокладання, наполегливість та здатність до відкладеного задоволення (delayed gratification) [4]. Ці якості мають доведений позитивний вплив на академічну успішність, особливо у молодшій та підлітковій аудиторії.

Практичне підтвердження викладених принципів отримано при розробці мобільного застосунку DinoRunner — runner-гри для Android на базі Unreal Engine, що інтегрує науково достовірні факти про динозаврів безпосередньо в ігровий процес. Факти відображаються у вигляді анімованих карток на порогових значеннях рахунку, не переривають гру і відповідають тематиці ігрового світу. Система вибору персонажів з різними видами динозаврів додатково стимулює допитливість гравця та спонукає до самостійного пошуку інформації.

Висновки та перспективи. Ігри жанру runner мають значний, але недооцінений потенціал для розвитку когнітивних, психомоторних та мотиваційних якостей користувачів. Ефективність цього впливу суттєво зростає при цілеспрямованій інтеграції освітнього контенту з дотриманням принципів ненав'язливості, стислості та тематичної відповідності. Перспективою подальших досліджень є проведення кількісного аналізу засвоєння пізнавального контенту гравцями різних вікових груп, а також розробка педагогічних рекомендацій щодо використання edutainment-ігор у навчальному процесі.

Список використаних джерел

1. Green C. S., Bavelier D. Action video game modifies visual selective attention. Nature. 2003. Vol. 423. P. 534–537.

2. Krakauer J. W. Motor learning: its relevance to stroke recovery and neurorehabilitation. Current Opinion in Neurology. 2006. Vol. 19, No. 1. P. 84–90.

3. Csikszentmihalyi M. Flow: The Psychology of Optimal Experience. Harper & Row, 1990. 303 p.

4. Mischel W., Shoda Y., Rodriguez M. I. Delay of gratification in children. Science. 1989. Vol. 244. P. 933–938.

5. Prensky M. Digital Game-Based Learning. McGraw-Hill, 2001. 442 p.

Гридько Богдан Андрійович
студент 4 курсу, групи ПД-41
Державний університет інформаційно-комунікаційних технологій
(073)-036-53-09
booking.bogdan@gmail.com

РОЛЬОВО-ОРІЄНТОВАНА АРХІТЕКТУРА КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБСИСТЕМИ МЕНЕДЖМЕНТУ ТОВАРНИХ ЗАПАСІВ

Постановка задачі. Сучасні вебсистеми менеджменту товарних запасів обслуговують кілька категорій користувачів - продавців, адміністраторів магазинів і менеджерів складів - кожна з яких вимагає власного набору функцій, навігації та рівня доступу до даних. Монолітний підхід, за якого один інтерфейс приховує або відображає елементи залежно від ролі, призводить до надмірної складності коду та порушень безпеки [1]. Завданням даної роботи є проєктування та реалізація рольово-орієнтованої архітектури клієнтської частини системи LvStore, яка ізолює робочий простір кожної ролі на рівні маршрутизації, layout-компонентів і засобів автентифікації.

Мета дослідження. Мета дослідження - визначити та обґрунтувати архітектурні принципи побудови клієнтської частини багаторольової вебсистеми, за яких розмежування доступу, навігація та склад інтерфейсу визначаються роллю автентифікованого користувача, а не умовним приховуванням елементів у спільному layout-дереві [2].

Результати дослідження. У системі LvStore визначено ролі користувачів: SELLER (продавець), STORE_ADMINISTRATOR (адміністратор магазину) та WAREHOUSE_MANAGER (менеджер складу), описані за допомогою TypeScript-переліченнь (enum), що виключає використання довільних рядкових констант замість допустимих значень [3]. Кожній ролі відповідає ізольований простір маршрутів: /seller/*, /admin/*, /warehouse/* - що повністю виключає можливість навігації між ролями без відповідних прав.

Захист маршрутів реалізовано через окремі компоненти-охоронці (Auth Guard): SellerAuthGuard, AdminAuthGuard, WarehouseAuthGuard. Кожен охоронець за допомогою бібліотеки TanStack Query (хук useQuery) виконує запит GET /auth/me для отримання поточного аутентифікованого користувача [4]. У разі отримання HTTP-статусу 401 або 403 відбувається перенаправлення на сторінку входу /login зі збереженням шляху повернення. Якщо роль користувача не відповідає захищеному розділу, компонент перенаправляє на головну сторінку [5].

Для кожної ролі реалізовано відокремлені layout-компоненти (Shell) та бічні навігаційні панелі (Sidebar), що містять виключно ті розділи меню, які доступні конкретній ролі:

1. SellerShell, SellerSidebar: огляд, товари, залишки, замовлення;
2. AdminShell, AdminSidebar: огляд, товари, замовлення, магазини та користувачі, залишки на складах магазинів;
3. WarehouseShell, WarehouseSidebar: огляд, замовлення, склади, складські залишки, товари, залишки магазинів.

JWT-автентифікація реалізована через axios-перехоплювач (axiosInstance.ts): після успішного входу токен автоматично додається до заголовків усіх наступних запитів. Серверний стан кешується засобами TanStack

Query з параметром staleTime: 30 секунд, що зменшує кількість зайвих мережевих запитів при навігації між сторінками [4].

Висновки та перспективи. В дослідженні розглянуто архітектурні принципи побудови клієнтської частини багаторольової вебсистеми менеджменту товарних запасів. Проведений аналіз підтвердив перевагу підходу з ізоляцією робочих просторів на рівні маршрутизації та layout-компонентів над підходом умовного приховування елементів у спільному дереві компонентів: перший забезпечує чітке розмежування доступу, унеможливлює несанкціоновану навігацію між ролями та спрощує супровід коду. Застосування TypeScript-перелічень для визначення ролей і статусів замовлень додатково підвищує надійність системи та знижує ймовірність помилок під час розширення її функціональності.

Перспективами розвитку архітектури є впровадження механізму тонкогранульованих прав доступу (permissions) у межах ролей, реалізація динамічного підвантаження рольових модулів (lazy loading) для зменшення обсягу початкового bundle, а також розширення функціональності аналітичних dashboard-панелей для кожної ролі.

Список використаних джерел

1. Sandhu R. S., Coyne E. J., Feinstein H. L., Youman C. E. Role-based access control models. IEEE Computer. Vol. 29, No. 2. P. 38–47. URL: https://doi.org/10.1109/2.485845

2. Bank A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. O'Reilly Media, 2020. 310 p.

3. Cherny B. Programming TypeScript: Making Your JavaScript Applications Scale. O'Reilly Media, 2024. 356 p. URL: https://www.oreilly.com/library/view/programming-typescript/9781492037644/

4. TanStack Query v5 Documentation. Queries and Caching. 2024. URL: https://tanstack.com/query/latest/docs/framework/react/guides/queries

5. React Router v7 Documentation. Nested Routes and Auth Guards. 2024. URL: https://reactrouter.com/docs/en/main/guides/auth

Новачук Ігор Сергійович
студент 4 курсу, групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій
(063) 794–23-62
jjecan11@gmail.com

Читулян Вадим Олегович
старший викладач кафедри
Технології Цифрового Розвитку
Державного університету інформаційно-комунікаційних технологій, м. Київ
vadymchytulian@gmail.com

РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ SAFE DRIVE ДЛЯ ІНТЕРАКТИВНОГО ВИВЧЕННЯ ПРАВИЛ ДОРОЖНЬОГО РУХУ З ЕЛЕМЕНТАМИ СИМУЛЯЦІЇ

Постановка задачі. Сучасні методи навчання водіїв часто є дуже обмеженими і застарілими, бо навчання новачків зводиться здебільшого до традиційних тестів, а практичного досвіду вони отримують замало. Тому є потреба у розробці інтерактивного програмного продукту з динамічним моделюванням. Сучасні тенденції цифровізації освіти показують, що треба перейти від звичних методів навчання до більш активних та інтерактивних технологій. Це допоможе майбутнім водіям швидше реагувати і краще пристосовуватися до змін на дорозі. Важливість цього дослідження полягає в тому, що потрібні методи, які допоможуть ефективно вивчати правила дорожнього руху в складних і швидкозмінних ситуаціях. Нове ПЗ має поєднувати теорію з можливістю приймати рішення в режимі реального часу.

Мета дослідження. Метою дослідження було створити пакет SafeDrive, який допоможе користувачам краще підготуватися до іспитів на права завдяки комбінованому модулю тестування, графічному моделюванню дорожніх ситуацій і аналітиці їх результатів.

Результати дослідження. Архітектура SafeDrive базується на поєднанні бази даних з навчальним контентом і модуля, який візуалізує дорожню обстановку. На відміну від інших систем, ця пропонує більш комплексний підхід до навчання.

- Модуль симуляції дає змогу відтворювати рух транспорту, роботу світлофорів і поведінку пішоходів, що допомагає користувачу приймати рішення в реальному часі.
- Інтелектуальні підказки аналізують відповіді користувача, знаходять типові помилки у певних розділах ПДР і формують індивідуальні поради для повторення.
- Візуальна аналітика показує графіки прогресу, завдяки чому легко побачити, як змінюються знання з часом.

Програма має два головні режими роботи: «Навчання» з докладними поясненнями і «Екзамен», який нагадує справжні умови сертифікації. Однією з корисних можливостей є створення власних дорожніх ситуацій, що допомагає краще зрозуміти логіку і зв’язки між подіями на дорозі.

Особливість цієї архітектури в тому, що аналітичний модуль розбиває кожну помилку на складові. Замість того, щоб просто сказати, що відповідь неправильна, система показує конкретний пункт ПДР і демонструє, до чого може призвести обраний неправильний маневр. Це допомагає краще зрозуміти зв'язок між причинами та наслідками, що в підсумку може зменшити кількість серйозних помилок під час справжнього іспиту.

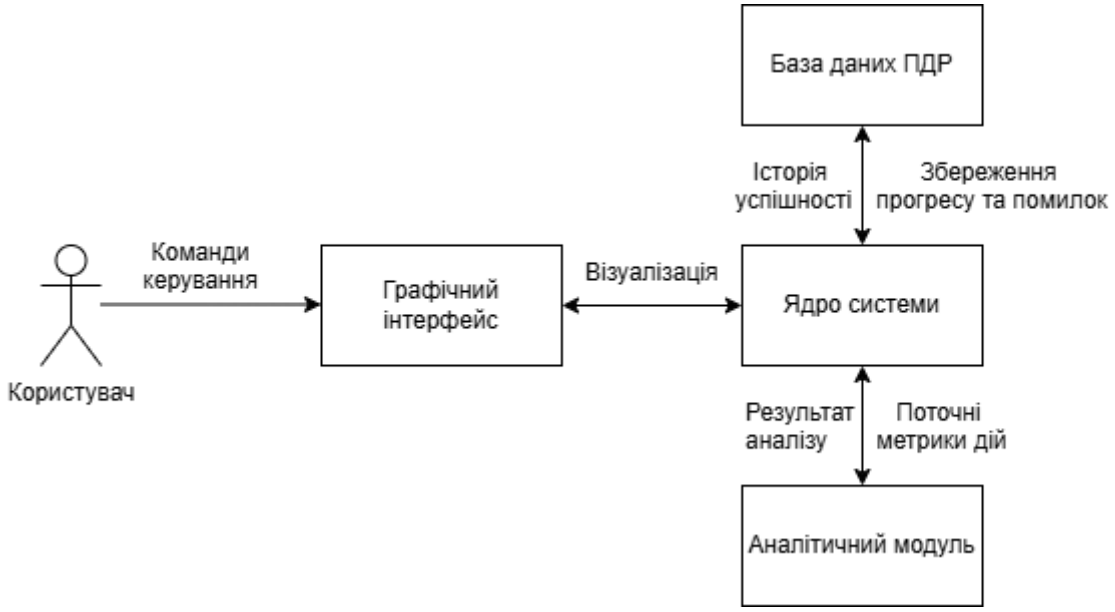


Рис. 1. Схема інформаційних потоків та взаємодії модулів ПЗ SafeDrive

Висновки та перспективи. Програмне забезпечення SafeDrive показує кращу адаптивність у порівнянні зі звичайними тренажерами завдяки інтерактивним симуляціям і персоналізованому підходу до навчання. Аналіз показав, що додавання модулів симуляції у навчання допомагає краще засвоювати матеріал. Поєднання звичайних тестів з інтерактивними методами робить SafeDrive хорошим варіантом для самостійної підготовки, який підлаштовується під рівень знань користувача і економить час. У подальших дослідженнях планується додати елементи гейміфікації, розширити базу складних сценаріїв, наприклад, з погодними умовами чи навчанням вночі, а також створити мобільну версію додатку для зручного доступу до матеріалів у будь-який час.

Список використаних джерел

1. Про Правила дорожнього руху: Постанова Кабінету Міністрів України від 10 жовтня 2001 р. № 1306 (зі змінами та доповненнями)

2. Kravchenko O., Melnyk V. Development of interactive learning systems for transport safety. Journal of Computing and Information Systems. 2023. Vol. 15, No. 3. P. 45–58..

3. Smith J., Doe A. Simulation-based training in driver education: A review of modern software solutions. International Journal of Educational Technology. 2024. Vol. 12, P. 102341.

Решетнік Віталій Віталійович
Студент 4 курсу, групи ТЦР-44
Гордієнко Катерина Олександрівна
Старший викладач
Державний університет інформаційно-комунікаційних технологій
(097)115-39-53

АНАЛІЗ ВИКОРИСТАННЯ PWA-ТЕХНОЛОГІЙ У ВЕБЗАСТОСУНКУ ДЛЯ ПОШУКУ КУЛІНАРНИХ РЕЦЕПТІВ

Постановка задачі. Вебзастосункам в наш час потрібні зручність та швидкість, яких іноді не вистачає, і користувачі потребують щось більше ніж базова робота в браузері. Через це користувачі які часто повертаються до додатку потребують спрощений доступ, збільшену швидкість доступу до застосунку та інформації. Тому в сучасних системах актуально використовувати PWA.

Мета дослідження. Провести аналіз PWA та як робота в ньому впливає на вебзастосунки, а також чи вони справді корисні для підняття зручності та доступності.

Результати дослідження. Для забезпечення зручності при регулярному використанні вебзастосунку, використовують PWA. Завдяки цьому користувач отримує не лише сторінку в браузері, а й формат взаємодій, який схожий більше на окремий застосунок. Це надає ширшу доступність, через можливість його встановити , та буде він працювати на різних пристроях з можливістю використання платформи[1]. Саме для сервісу знаходження рецептів це корисно, через те що, до нього часто повертаються повторно. Цінність в ній не в самій “технології”, а в швидкодії доступу та зручності відкриття різних сторінок. Це важливо для системи, якою користувач постійно використовує бо він надає швидкий доступ інформації для його потреб.

Дані про застосунок зберігаються саме в файлі “Manifest”. Саме цей JSON-файл спрямований на роботу з застосунком, а точніше з його інформацією. Одним з найпоширеніших способів його використання, є надання можливості завантаження PWA на пристрої. Він допомагає відображенню назви, іконок, сторінок та іншої потрібної інформації. Це забезпечує для користувача окремий вигляд застосунку без інших зайвих елементів браузера. Display дає визначення скільки інтерфейсу браузера буде бачити користувач. Start_url налаштовує сторінку, яка буде відкриватись після запуску додатку. Це дає більш передбачений та зручний запуск вебзастосунку для користувача[2].

Для контролювання роботи з кешем, та мережевими запитами, використовується Service Worker. Це допомагає зробити повторне відкриття швидшим. Також додається можливість налаштування матеріалів та даних які будуть відображатись навіть без стабільного інтернет з’єднання. Для додатку кулінарних рецептів це важливо, через потребу в відкритті рецепту не один раз та не завжди в місці де стабільне з’єднання. Service Worker також може працювати як щось схоже на проксі між застосунком, мережею та браузером. Через його можливість підміняти потрібні ресурси зі свого кешу, додає підтримку сценаріїв, типу cached assets або offline-first[3].

На сайті особливо корисним є кешування сторінок на які користувачі часто повертаються. Обов’язковим є додання кешування на основні сторінки такі як

головна, та інші сторінки для отримання інформації не тільки про сайт, а й до контенту. Для користувачів які часто використовують застосунок, є важливим аспектом можливість завантаження на телефон, або на персональний комп'ютер додатком. Що підвищує комфорт та можливості використання системи.

У кулінарних вебзастосунках завжди є потреба в пошуку рецепта, відкритті , перегляді інформації. Збільшення швидкості запуску та запуск в окремому додатку підвищує практичну користь для вебзастосунку. PWA доповнює додаток та робить його доступнішим для більшого кола користувачів, а не змінює його.

Висновок. На основі аналізу можна зробити висновок, що PWA-функціональність підвищує зручність та доступність вебзастосунку. Це забезпечує комфортне використання системи при повторному поверненні користувача. Manifest підвищує зручність через можливість встановлення додатку та налаштування вигляду. В свою чергу Service worker та кешування забезпечують підтримку повторного доступу до системи. Вони разом роблять його більш практичним для користувача.

Список використаних джерел

1. Progressive web apps // MDN Web Docs.
2. Web application manifest // MDN Web Docs.
3. Using Service Workers // MDN Web Docs.

Кириловський Володимир
студент групи ТЦР-43
Державний університет інформаційно-комунікаційних технологій
(095) 147 -25 -88
kirilovskiy.v.v@gmail.com

ВЕБ-ЗАСТОСУНОК ДЛЯ ОБЛІКУ ВІДДІЛЕНЬ ФІНАНСОВОЇ КОМПАНІЇ, РОБОЧИХ МІСЦЬ ANYDESK ТА ЛІМІТІВ ОБОРОТУ

Постановка задачі. У сучасних умовах цифровізації фінансового сектору автоматизація внутрішніх бізнес-процесів є необхідною умовою ефективного функціонування компаній. Значна кількість фінансових установ мають розгалужену мережу відділень, велику кількість працівників та значні обсяги фінансових операцій. Контроль інформації про відділення, робочі місця співробітників та фінансові ліміти часто виконується за допомогою таблиць, локальних програм або вручну, що ускладнює процес адміністрування та збільшує ризик помилок.

Актуальність дослідження обумовлена необхідністю створення єдиної інформаційної системи, яка забезпечить централізований облік усіх відділень компанії, контроль доступу до віддалених робочих місць через AnyDesk та моніторинг фінансових лімітів обороту. Використання веб-застосунку дозволяє забезпечити швидкий доступ до інформації, підвищити прозорість бізнес-процесів та зменшити навантаження на адміністративний персонал.

Мета дослідження. Метою роботи є розробка веб-застосунку для автоматизації процесів обліку відділень фінансової компанії, управління робочими місцями через програмне забезпечення AnyDesk та контролю встановлених лімітів фінансового обороту.

Результати дослідження. Під час дослідження було проведено аналіз існуючих програмних рішень для автоматизації діяльності фінансових організацій. Було визначено, що більшість існуючих систем мають вузьку спеціалізацію та не забезпечують комплексного управління всіма внутрішніми ресурсами компанії.

Для вирішення поставленої задачі було обрано архітектуру типу «клієнт-сервер». Користувач взаємодіє із системою через веб-браузер, що забезпечує доступ до функціоналу з будь-якого пристрою, підключеного до мережі Інтернет. Серверна частина відповідає за обробку запитів користувачів, реалізацію бізнес-логіки та взаємодію з базою даних.

Функціонал системи включає модуль управління відділеннями, який дозволяє зберігати інформацію про адресу відділення, відповідальних осіб, графік роботи та поточний статус. Модуль управління робочими місцями дозволяє зберігати AnyDesk ID, контролювати підключення та вести історію змін.

Окремий модуль контролю лімітів обороту дозволяє встановлювати допустимі фінансові межі для кожного відділення, відстежувати перевищення встановлених показників та формувати автоматичні повідомлення для відповідальних осіб.

Система звітності дозволяє формувати аналітичні звіти за визначений період, аналізувати ефективність роботи відділень та контролювати фінансові показники компанії.

Для реалізації серверної частини доцільно використовувати ASP.NET Core, а для зберігання даних — PostgreSQL або SQL Server. Клієнтська частина може бути реалізована із використанням сучасних JavaScript-фреймворків.

Важливою складовою системи є механізм авторизації та розмежування прав доступу користувачів. Адміністратори мають повний доступ до системи, менеджери можуть переглядати аналітичні звіти, а працівники відділень працюють лише зі своїми даними.

Впровадження системи дозволить зменшити кількість помилок, прискорити обробку інформації, підвищити ефективність управління та оптимізувати внутрішні бізнес-процеси фінансової компанії.

Висновки та перспективи. У результаті дослідження було визначено основні функціональні вимоги до веб-застосунку та запропоновано архітектурне рішення для його реалізації. Подальші дослідження можуть бути спрямовані на інтеграцію системи з CRM-платформами, використання технологій штучного інтелекту для прогнозування фінансових показників та посилення кібербезпеки.

Список використаних джерел

1. Microsoft Documentation. ASP.NET Core. URL: https://learn.microsoft.com.
2. AnyDesk Official Website. URL: https://anydesk.com.
3. Sommerville I. Software Engineering. Pearson Education, 2020.
4. Silberschatz A. Database System Concepts. McGraw-Hill, 2019.

Паламарчук Максим Олегович
студент 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
(050) 91-55-938
mpalamarcuk635@gmail.com

ЦИФРОВИЙ ВЕБ-СЕРВІС ДЛЯ КОНТРОЛЮ ТА КЕРУВАННЯ ДОСТАВКАМИ ПОСИЛОК НА ОСНОВІ ASP.NET CORE

Постановка задачі. На сьогодні, потреба у підвищенні прозорості процесів доставки та ефективній взаємодії між користувачами та службами доставки, у логістичних сферах зростає. Існуючі системи надають лише інформацію по статусам відправлення, без надання детальної інформації та можливості зв'язатись з кур'єром. Наразі, є проблема з відсутністю сервісів, що будуть надавати інформацію про поточне місцезнаходження відправлення або дають можливість взаємодіяти з кур'єром, що призводить до зниження рівня сервісу.

Отже, є необхідність розробити веб-додаток, що надає розширений функціонал управління посилками, інформативніший механізм відстеження та систему обміну повідомленнями.

Мета дослідження. Метою дослідження є розробка цифрового веб-сервісу управління та контролю доставки посилок, що забезпечує: створення та обробку посилок; відстеження місцезнаходження посилки; комунікацію між кур'єром та користувачем; автоматизацію логістичних процесів; зпучний користувацький інтерферс.

Результати дослідження. У результаті дослідження було розроблено веб-додаток, з використанням технології ASP.NET Core, з клієнт-серверною архітектурою та патерною MVC.[1]

У додатку реалізовані такі механізми:

- Користувач може створювати відправлення, переглядати детальну інформацію про них та відстежувати їх статус. Для кожної посилки генерується унікальний трекінговий номер, з його допомогою здійснюється пошук відправлення та відображається інформація про нього.
- Розроблено функціонал відстеження посилок, що включає в себе перегляд інформації по статусу, місцезнаходження посилки та маршрут переміщення на карті, для візуалізації використано бібліотеку Leaflet Tutorials.[4]
- Механізм передачі координат водієм реалізований за допомогою Geolocation API, що дозволяє в реальному часі стежити за рухом транспорту.[3]
- Для організації обміну повідомленнями між отримувачем та кур'єром, розроблений чат за допомогою бібліотеки SignalR, яка дозволяє комунікувати в реальному часі.[5]
- Адміністратор додатково має можливість масово змінювати статус посилок та призначати їм транспорт для доставки.
- Інтерфейс користувача було реалізовано за допомогою шаблонізатора Razor та Bootstrap, тож додаток має зручну навігацію, адаптивність.[2]

Під час дослідження було розроблено веб-сервіс для управління та контролю доставками посилок, який включає в себе відслідковування, механізм обміну повідомленнями та підвищення якості взаємодії з кінцевим користувачем.(рис. 1)

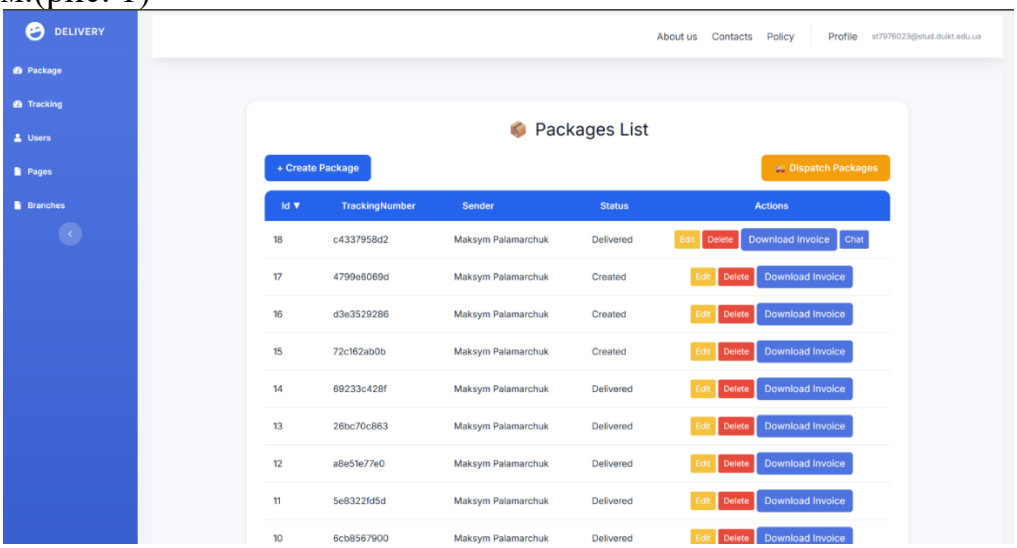


Рис.1 Інтерфейс веб-додатку

Висновки. Розроблене рішення підвищує рівень прозорості процесу доставки, спрощує комунікацію між кур'єром та отримувачем. Використання сучасних технологій ASP.NET Core, Geolocation API, Bootstrap, SignalR, дозволило розробити цифровий веб-сервіс для управління та контролю доставками з елементами інтерактиву.

Список використаних джерел

1. ASP.NET documentation [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-10.0
2. Get started with Bootstrap [Електронний ресурс] – Режим доступу: https://getbootstrap.com/docs/5.3/getting-started/introduction/
3. HTML Geolocation API [Електронний ресурс] – Режим доступу: https://www.w3schools.com/html/html5_geolocation.asp
4. Leaflet Tutorials [Електронний ресурс] – Режим доступу: https://leafletjs.com/examples.html
5. What is SignalR? A Real-Time Communication Framework for .NET [Електронний ресурс] – Режим доступу: https://dev.to/dazevedo/what-is-signalr-a-real-time-communication-framework-for-net-29h0

Білецький Костянтин Євгенович
студент 4 курсу, групи ТЦР-41
Державний університет інформаційно-комунікаційних технологій
(095) 022-04-05
kostiabiletskiy31415@gmail.com

ІНТЕГРАЦІЯ NYLAS API ДЛЯ СИНХРОНІЗАЦІЇ КАЛЕНДАРІВ ТА АВТОМАТИЗАЦІЇ РЕЗЕРВУВАВННЯ ЗУСТРІЧЕЙ У ЦИФРОВІЙ ПЛАТФОРМІ НА БАЗІ NEXT.JS

Постановка задачі. Сучасні цифрові платформи для планування зустрічей стикаються з необхідністю надійної та безпечної інтеграції з різними календарними сервісами (Google Calendar, Microsoft Outlook, Apple iCloud та іншими). Традиційні методи безпосередньої інтеграції з API провайдерів вимагають підтримки кількох OAuth-потоків, обробки вебхуків та індивідуальної логіки для кожного сервісу. Це ускладнює розробку, збільшує ймовірність помилок та ускладнює підтримку додатка. Тому є сенс використанні таких рішень, таких як Nylas API v3, яке дозволяє реалізувати bidirectional синхронізацію календарів, перевірку доступності та автоматичне резервування зустрічей у єдиному інтерфейсі.

Мета дослідження. Метою дослідження є аналіз можливостей та особливостей інтеграції Nylas API v3 у веб-додаток, розроблений на базі фреймворку Next.js 16. А також, визначення переваг такого підходу для автоматизації процесів планування та резервування зустрічей.

Результати дослідження. Інтеграція реалізована з використанням Next.js 16 (App Router, Server Components та Server Actions). Для роботи з Nylas API застосовано офіційний Node.js SDK. Основні етапи включають:

- Автентифікацію користувачів через Hosted OAuth з подальшим створенням Grant для доступу до календарів.
- Отримання інформації про доступність учасників за допомогою Availability API з урахуванням робочого часу, буферів та існуючих подій.
- Автоматичне створення подій через метод events.create з генерацією запрошень та підтримкою відео-конференцій.
- Bidirectional синхронізацію змін між платформою та зовнішніми календарями за допомогою вебхуків.

Використання Server Components та Route Handlers забезпечує безпечне зберігання API-ключів на сервері та оптимальну продуктивність. Порівняно з прямими інтеграціями Nylas API значно спрощує кодову базу, підтримує роботу з кількома провайдерами одночасно та надає готові інструменти для перевірки доступності та бронювання.

Висновки та перспективи. Проведене дослідження підтвердило ефективність інтеграції Nylas API v3 у цифрову платформу на базі Next.js. Такий підхід дозволяє суттєво скоротити час розробки, підвищити надійність синхронізації даних та забезпечити зручний користувацький досвід. Отримані результати можуть бути використані при подальшому розвитку платформи планування та резервування зустрічей.

Перспективи подальших досліджень включають розширення функціоналу за рахунок інтеграції штучного інтелекту для автоматичного підбору оптимальних часових слотів та підтримки групових зустрічей.

Список використаних джерел

1. Nylas API Documentation. Using the Calendar API [Електронний ресурс] – Режим доступу: https://developer.nylas.com/docs/v3/calendar
2. Nylas API Documentation. Using Nylas Scheduler [Електронний ресурс] – Режим доступу: https://developer.nylas.com/docs/v3/scheduler/
3. Next.js Documentation. App Router and Server Component [Електронний ресурс] – Режим доступу: https://nextjs.org/docs
4. Nylas Samples. Quickstart v3 Scheduler for Next.js (App Router) [Електронний ресурс] – Режим доступу: https://developer.nylas.com/docs/v3/calendar

Гончаренко Дмитро Олександрович
студент 4 курсу, групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій
(096) 804-46-56
st8142088@stud.duikt.edu.ua

ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОГРАМНИХ РІШЕНЬ ДЛЯ ОЦІНЮВАННЯ ОКУПНОСТІ ЕНЕРГОЕФЕКТИВНИХ ІНВЕСТИЦІЙ ТА ОБҐРУНТУВАННЯ ФУНКЦІОНАЛЬНИХ ВИМОГ ЗА МЕТОДОЛОГІЄЮ MoSCoW

Постановка задачі. Сучасні умови функціонування енергетичного сектору України характеризуються стійким попитом на автономні та енергоефективні технологічні рішення для домогосподарств, об'єднань співвласників багатоквартирних будинків (ОСББ) та суб'єктів малого й середнього бізнесу (МСБ). Цей попит посилюється комплексом державних програм фінансової підтримки, серед яких — «Енергодім» з грантовим співфінансуванням до 70 % (максимум 25 млн грн), «ГрінДІМ» (старт 18.02.2026, грант до 70 % для СЕС та теплонасосів, максимум 2–4 млн грн), «Доступні кредити 5-7-9 %» (ліміт 250 млн грн, ставка від 1 %), а також програми міжнародних партнерів — ПРООН ($230 млн) та ЄІБ (€600 млн на 2026–2027 рр.).

Незважаючи на різноманіття програм та технологій, прийняття обґрунтованого інвестиційного рішення потребує одночасного врахування чистих інвестиційних витрат з урахуванням гранту, темпу зростання тарифу, дисконтної ставки та лімітів співфінансування. Наявні програмні рішення не забезпечують цієї комплексності, що формує наукову задачу формалізації функціональних вимог до спеціалізованого веб-сервісу на основі попереднього порівняльного аналізу аналогів.

Мета дослідження — обґрунтувати функціональні вимоги до інтерактивного веб-сервісу порівняння термінів окупності інвестицій у енергоефективність шляхом порівняльного аналізу існуючих програмних рішень та формалізації прецедентів використання за методологією MoSCoW.

Об'єктом дослідження виступає процес інженерії програмних вимог до систем оцінювання інвестиційних показників енергоефективних проєктів; **предметом** — методи порівняльного аналізу аналогів та формалізації функціональних вимог із застосуванням MoSCoW-пріоритизації та UML-моделювання прецедентів.

Результати. Проведений gap-аналіз наявних програмних рішень для оцінювання окупності енергоефективних інвестицій дозволив виявити три ключові функціональні недоліки:

1) відсутність інтеграції актуальних українських грантових програм у розрахункові моделі PP, NPV та IRR, що спричиняє систематичне завищення прогнозованих термінів окупності;

2) відсутність повного набору з п'яти інвестиційних показників (PP, NPV, DPP, IRR, PI) у єдиному інтерфейсі користувача, прийнятному для неспеціалістів;

3) відсутність функціоналу збереження й порівняння проєктів, а також відкритого REST API для інтеграції зі сторонніми системами енергоаудиту.

На основі виявлених недоліків та аналізу потреб трьох цільових аудиторій — домовласників, представників ОСББ та суб'єктів МСБ — формалізовано 13 функціональних вимог (FR-01 – FR-13), згрупованих за методологією MoSCoW. Фрагмент матриці вимог наведено у табл. 1.

Таблиця 1

Фрагмент матриці функціональних вимог за методологією MoSCoW

ID	Функціональна вимога	Роль	Пріоритет
FR-01	Розрахунок PP, NPV, DPP, IRR, PI без реєстрації	Гість	Must
FR-04	Збереження та порівняння власних проєктів	Користувач	Must
FR-07	Автоматичний підбір релевантних грантових програм	Користувач	Should
FR-10	CRUD-операції над довідниками тарифів і технологій	Адміністратор	Must
FR-13	Експорт звіту у PDF з графіком грошових потоків	Користувач	Could

Застосування пріоритизації MoSCoW (Must / Should / Could / Won't) дозволило розподілити вимоги за критичністю: 7 вимог класифіковано як Must-have (ядро калькулятора, збереження проєктів, адміністрування довідників), 4 — Should-have (рекомендаційні функції, експорт), 2 — Could-have (багатомовність, темна тема). Такий розподіл забезпечує чітке планування ітерацій розробки та зниження ризику виходу за межі мінімально життєздатного продукту (MVP).

На основі функціональних вимог розроблено діаграму прецедентів, що містить 15 варіантів використання (UC-01 – UC-15) для трьох ролей користувачів. Застосовано UML-зв'язки <<include>> для обов'язкових вкладених прецедентів (наприклад, аутентифікація перед збереженням проєкту) та <<extend>> для опціональних розширень (підбір грантових програм як розширення базового розрахунку). Такий підхід забезпечує модульність вимог та зменшує дублювання логіки при подальшій реалізації.

Матриця покриття прецедентами функціональних вимог становить 100 %: кожна вимога FR-01 – FR-13 реалізується щонайменше одним прецедентом, що свідчить про повноту моделі.

Висновки та перспективи. Запропонований підхід до обґрунтування функціональних вимог на основі комбінації gap-аналізу, методології MoSCoW та UML-моделювання прецедентів забезпечує формалізовану основу для проєктування веб-сервісу оцінювання окупності енергоефективних інвестицій. Наукова новизна роботи полягає у тому, що вперше сформульовано комплекс із 13 функціональних вимог та 15 прецедентів, орієнтованих на специфіку українських грантових програм 2025–2027 рр. Практична цінність результатів — їх пряме застосування при проєктуванні архітектури бекенду NestJS (модулі Auth, Projects, Calculations, Grants) та клієнтської частини Next.js.

Перспективи подальших досліджень пов'язані з автоматичним парсингуванням оновлень офіційного порталу Фонду енергоефективності для підтримки довідника грантів в актуальному стані, розробкою AI-модуля рекомендацій технологічних рішень на основі профілю користувача, а також

інтеграцією з Open Banking API банків-партнерів програми «5-7-9 %» для автоматизованої перевірки умов кредитування.

Список використаних джерел

1. Закон України «Про енергетичну ефективність» від 21.10.2021 № 1818-IX. URL: https://zakon.rada.gov.ua/laws/show/1818-20 (дата звернення: 15.04.2026).
2. Державна установа «Фонд енергоефективності». Програма «Енергодім»: умови участі та грантове співфінансування. URL: https://eefund.org.ua (дата звернення: 15.04.2026).
3. Cockburn A. Writing Effective Use Cases. Boston: Addison-Wesley, 2001. 270 p.
4. Clegg D., Barker R. Case Method Fast-Track: A RAD Approach. Addison-Wesley, 1994. 186 p. (оригінальне джерело методології MoSCoW).
5. Brosz N., Rostami R. Comparative Analysis of Software Tools for Energy Efficiency Investment Assessment. Energy Informatics. 2023. Vol. 6, No. 2. P. 45–62.
6. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley, 2003. 208 p.

Дорошенко Юлія Володимирівна
студентка 4 курсу, групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій
(068) 384-77-96
st7908959@stud.duikt.edu.ua

АРХІТЕКТУРНІ ПАТЕРНИ ПОБУДОВИ Е-СЕРВІСУ УПРАВЛІННЯ ЗАВДАННЯМИ НА ОСНОВІ SPRING BOOT 3.5 ТА JAVA 25

Постановка задачі.

Сучасні вебзастосунки у сфері бізнес-менеджменту характеризуються високими вимогами до масштабованості, надійності та швидкодії. Системи управління завданнями (task management systems) є критичним елементом корпоративної інфраструктури, оскільки забезпечують координацію роботи команд, відстеження дедлайнів та підвищення продуктивності процесів. Однак побудова такої системи ставить перед розробником низку архітектурних викликів: забезпечення слабкого зв'язування модулів, організація ефективного шару доступу до даних, підтримка паралельної обробки запитів та гнучкість щодо подальшого розширення функціональності.

Відсутність виваженого архітектурного підходу при побудові Spring Boot-застосунків призводить до формування «великої грудки бруду» (Big Ball of Mud) — системи, яка швидко втрачає зрозумілу структуру та ускладнює процес супроводу. Актуальним залишається питання вибору оптимальних архітектурних патернів, які дозволяють використати всі переваги платформи Spring Boot 3.5 та сучасних можливостей мови Java 25, зокрема віртуальних потоків.

Мета дослідження.

Метою роботи є обґрунтування вибору та аналіз ефективності застосування архітектурних патернів при побудові е-сервісу управління завданнями на платформі Spring Boot 3.5 та Java 25, а також розроблення рекомендацій щодо їх практичної реалізації для забезпечення масштабованості та супроводжуваності програмного забезпечення.

Для досягнення поставленої мети сформульовано такі завдання: проаналізувати класичні архітектурні підходи (монолітна, багатошарова, модульний моноліт); дослідити застосування патернів DTO, Repository, Service Layer у середовищі Spring Boot; оцінити вплив virtual threads (Project Loom) на продуктивність REST-ендпойнтів; запропонувати архітектурне рішення для е-сервісу управління завданнями.

Результати.

У результаті проведеного дослідження обґрунтовано доцільність використання багатошарової архітектури (Layered Architecture) для е-сервісу управління завданнями. Система поділяється на чотири логічні шари: шар представлення (Controller Layer), шар бізнес-логіки (Service Layer), шар доступу до даних (Repository Layer) та шар сутностей домену (Domain Layer). Такий поділ забезпечує принцип єдиної відповідальності (Single Responsibility Principle) та спрощує юніт-тестування окремих компонентів.

Застосування патерну DTO (Data Transfer Object) дозволяє розділити внутрішнє представлення даних (JPA-сутності) та зовнішній контракт REST API. Це убезпечує систему від витоку деталей доменної моделі та надає гнучкість при

еволюції схеми бази даних без порушення зворотної сумісності API. Для перетворення між сутностями та DTO рекомендовано використовувати бібліотеку MapStruct, яка генерує код мапінгу на етапі компіляції, уникаючи накладних витрат рефлексії.

Патерн Repository, реалізований через Spring Data JPA, абстрагує роботу з базою даних PostgreSQL та забезпечує автоматичну генерацію CRUD-операцій. Застосування інтерфейсу JpaRepository дозволяє скоротити обсяг шаблонного коду та сконцентруватися на бізнес-логіці. Для складних запитів доцільно використовувати механізм Query Methods або анотацію @Query з JPQL-виразами.

Особливу увагу приділено аналізу віртуальних потоків, які стали стабільною функцією платформи Java починаючи з версії 21 та повноцінно підтримуються у Java 25. У контексті Spring Boot 3.5 віртуальні потоки активуються через властивість spring.threads.virtual.enabled=true, що дозволяє обробляти значно більшу кількість одночасних HTTP-запитів без необхідності масштабування традиційного пулу потоків. Для I/O-інтенсивних операцій, характерних для systems task management (звернення до БД, відправлення email-сповіщень, виклики зовнішніх API), це дає суттєвий виграш у продуктивності.

Розроблено структуру модульного моноліту для е-сервісу, яка включає чотири основні функціональні модулі: модуль управління користувачами та автентифікації (з використанням Spring Security та JWT); модуль управління завданнями (CRUD-операції, статуси, пріоритети); модуль сповіщень (асинхронна відправка email через Spring Mail); модуль звітності. Кожен модуль має чітко визначений публічний інтерфейс, що в перспективі дозволяє виділити його в окремий мікросервіс без суттєвих змін бізнес-коду.

Висновки та перспективи.

Проведене дослідження підтвердило доцільність використання багатошарової архітектури у поєднанні з модульним моноліт-підходом для побудови е-сервісу управління завданнями на платформі Spring Boot 3.5 та Java 25. Запропоноване архітектурне рішення забезпечує чіткий розподіл відповідальностей між компонентами, гнучкість щодо масштабування та спрощення процесу супроводу. Використання сучасних можливостей Java 25, зокрема віртуальних потоків, дозволяє значно підвищити пропускну здатність REST API без ускладнення моделі програмування.

Перспективами подальших досліджень є: експериментальне порівняння продуктивності синхронної та реактивної моделі (Spring WebFlux) для задач task management; дослідження переходу від модульного моноліту до мікросервісної архітектури з використанням Spring Cloud; аналіз застосування патернів Event Sourcing та CQRS для ведення історії змін завдань; інтеграція з системами спостережуваності (Micrometer, OpenTelemetry) для моніторингу архітектурних метрик у продакшн-середовищі.

Список джерел:

1. Walls C. Spring in Action, Sixth Edition. — Shelter Island: Manning Publications, 2022. — 520 p.
2. Richardson C. Microservices Patterns: With examples in Java. — Shelter Island: Manning Publications, 2018. — 520 p.
3. Spring Boot Reference Documentation [Electronic resource]. — Access mode: https://docs.spring.io/spring-boot/docs/current/reference/html/ (дата звернення: 15.04.2026).

4. JEP 444: Virtual Threads [Electronic resource] / OpenJDK. — Access mode: https://openjdk.org/jeps/444 (дата звернення: 15.04.2026).
5. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. — Boston: Prentice Hall, 2017. — 432 p.
6. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. — Boston: Addison-Wesley Professional, 2003. — 560 p.

Москаленко Наталія Володимирівна,
асистент
Київський національний університет імені Тараса Шевченка
moskalenkonv@ukr.net
Сокульський Олег Євгенович
кандидат технічних наук, старший викладач
Військовий інститут телекомунікацій та інформатизації імені Героїв Крут
mortimer@ukr.net
Топольськов Євгеній Олександрович
кандидат технічних наук, доцент
Київський національний університет імені Тараса Шевченка
y.topolskov@knu.ua

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ТА ШТУЧНИЙ ІНТЕЛЕКТ

Постановка задачі. Поява та розвиток методів та засобів штучного інтелекту суттєво впливає на розвиток як апаратного так і програмного забезпечення. Персональні комп'ютери та інтернет(кінець XX - початок XXI ст.) значно розширили перелік задач та галузей, де технології штучного інтелекту з успіхом використовуються[1].

Мета дослідження. Метою дослідження є розгляд та аналіз засобів з елементами штучного інтелекту для розробки програмного забезпечення інформаційних систем.

Результати дослідження. Програмне забезпечення, як вагома частина інформаційних систем, потребує для своєї розробки використання сучасних методів та засобів. Особливою увагою та популярністю в сучасному світі користуються досягнення в галузі штучного інтелекту, які значно впливають не тільки на терміни та вартість розробки, але й значно покращують ефективність, функціональність, якість та надійність програмних продуктів. Досягнення штучного інтелекту можуть бути використані на всіх етапах розробки програмного забезпечення:

- ✓ створення;
- ✓ аналіз та покращення програмного коду;
- ✓ тестування;
- ✓ налагодження;
- ✓ ведення документації.

Найбільшою популярністю користуються засоби, які дозволяють генерувати або автоматично доповнювати програмний код. Зазвичай вони базуються на використанні моделей машинного навчання, що дає можливість використовувати контекст, коментарі або візуальний інтерфейс для генерації коду, швидко отримуючи тим самим ефективний варіант[2]. Сучасні засоби генерації програмного коду підтримують понад 20 мов програмування, серед яких Python, JavaScript, C++, Java тощо. Генератори програмного коду можна умовно розділити на наступні типи:

- ✓ Самостійне генерування коду на основі моделей (Low-code/No-code), які створюється з використанням візуального інтерфейсу. Це, наприклад, платформа Retool, Rapider AI, Mendix та інші.

- ✓ Генератори, що пропонують продовження коду під час його написання розробником. До цієї групи відносяться, наприклад, Tabnine, вбудований AI у VS Code.
- ✓ Створюють код на основі текстових запитів. Це так звані ШІ-асистенти (AI Code Generators). Найпопулярніші: GitHub Copilot, ChatGPT, Claude 3.5 Sonnet, Cursor.

Дуже часто засоби генерації поєднують в собі різні можливості. Наприклад, платформа GitHub Copilot, яка є спільною розробкю GitHub та OpenAI, створена на основі мовної моделі OpenAI Codex. Цей віртуальний помічник програміста дозволяє йому швидко писати чистий та якісний код. Платформа не тільки генерує код на основі текстових запитів, але й пропонує розробнику готові рядки або блоки коду.

Платформа Claude 3.5 Sonnet розроблена компанією Anthropic, презентована в червні 2024, розглядається як альтернатива ChatGPT. Осоливістю є ефективне розпізнавання зображень, графіки, діаграми та перетворення їх на структуровані дані. Працює з великими обсягами інформації, генерує тексти, звіти та маркетингові матеріали за шаблонами.

Згідно своїх потреб та рівня знань в галузі програмування розробник може підібрати найбільш ефективний засіб для створення програмного забезпечення, але необхідно зауважити, що згенерований код потрібно обов'язково перевіряти на наявність помилок та небезпечного коду, а також відслідковувати правильність відповідного контексту[2].

Висновки та перспективи. Використання засобів з елементами штучного інтелекту для створення програмного забезпечення надає суттєві переваги розробникам програмного забезпечення не тільки в термінах та вартості розробки, але й суттєво підвищує якість та надійність інформаційних систем. Подальший розвиток штучного інтелекту дозволить ще більше розширити кількість та можливості засобів для розробки програмного забезпечення.

Список використаних джерел

1. Лясковська С.Є., Панін В.С., Шаховська Н.Б. Генеративний штучний інтелект: ISBN 978-966-994-3. Львівська політехніка, 2026. – 206 с.

2. Каратаєв, О., Шубін, І.. Проблеми повторного використання знань у процесі проектування програмних систем. Innovative technologies and scientific solutions for industries, 2(24). 62. https://doi.org/10.30837/ITSSI.2023.24.0.

Єфімов Максим Олександрович
студент групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
(068) 986 - 99 - 18
evimovm9@gmail.com
Науковий керівник: Аронов Андрій Олексійович,
кандидат технічних наук,
доцент кафедри Технологій цифрового розвитку
Державного університету інформаційно-комунікаційних технологій

ОСОБЛИВОСТІ РОЗРОБКИ МОДУЛЯ ОНЛАЙН-ТЕСТУВАННЯ У ВЕБ-ОРІЄНТОВАНІЙ СИСТЕМІ НАВЧАННЯ

Постановка задачі. Онлайн-тестування сьогодні стало звичним елементом навчального процесу, особливо коли йдеться про дистанційний формат. Воно дозволяє швидко перевіряти знання студентів і значно спрощує роботу викладача. Але створення такого модуля на практиці виявляється не таким простим: потрібно забезпечити стабільну роботу таймера, коректне збереження відповідей, підтримку різних типів запитань і при цьому зробити інтерфейс зрозумілим для користувача. У багатьох системах ці елементи працюють неузгоджено або вимагають складного налаштування. Саме тому виникла потреба зробити власний модуль онлайн-тестування, який був би простим у використанні та надійним у роботі, а також органічно вписувався у вже створену LMS-систему.

Мета дослідження. Метою роботи є створення модуля онлайн-тестування (далі Модуль), який підтримує кілька типів тестових завдань, автоматично обчислює бали, коректно працює з таймером і зберігає результати у базі даних SQL Server. Модуль має бути частиною веб-орієнтованої LMS-системи та взаємодіяти з інтерфейсами студента та викладача.

Результати дослідження. У межах роботи було реалізовано модуль онлайн-тестування, який інтегровано у LMS-систему з трьома ролями користувачів. Модуль підтримує чотири типи запитань: вибір однієї відповіді, множинний вибір, встановлення порядку та вибір зображення. Це дає можливість створювати тести різної складності та робити їх більш гнучкими.

Серверна частина побудована на PHP, а дані зберігаються у SQL Server. Для роботи з базою використано PDO, що забезпечує стабільність і захист від SQL-ін’єкцій. Загальну структуру модуля наведено на (рис. 1).

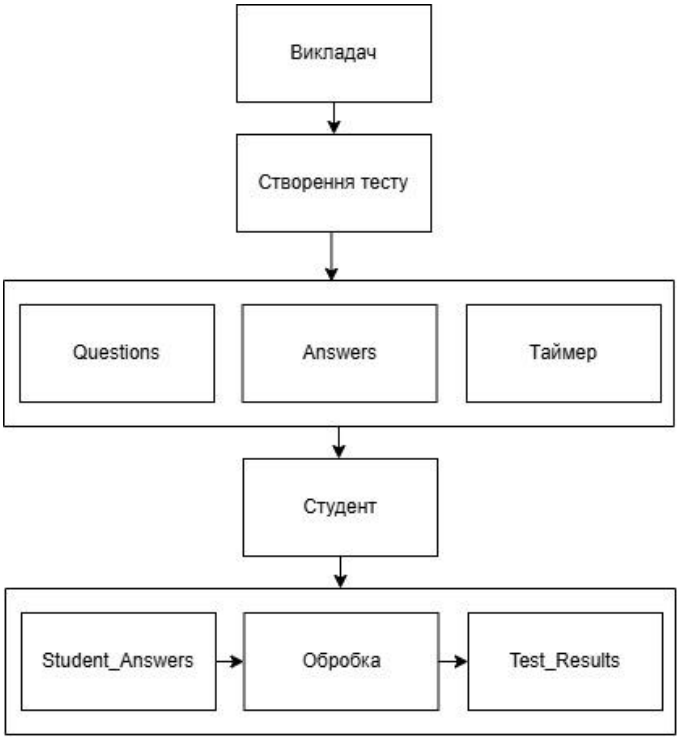


Рис. 1. Архітектура модуля онлайн-тестування у LMS-системі.

Одним із важливих елементів є таймер. Під час створення спроби фіксується час її початку, після чого система обчислює, скільки часу залишилося. Для уникнення помилок, пов’язаних із форматом дат SQL Server, було додано нормалізацію часу. Якщо час спливає, тест завершується автоматично, а результат записується у таблицю Test_Results.

Під час розроблення виникла проблема з тим, що SQL Server не приймв дробові значення балів. Оскільки поле score має тип int, значення на кшталт 60.67 викликали помилку. Після впровадження округлення результатів перед збереженням модуль почав працювати стабільно.

Усі відповіді студентів зберігаються у таблиці Student_Answers, що дозволяє переглядати не лише підсумковий бал, а й конкретні вибрані варіанти. Після завершення тесту результат записується у таблицю Test_Results разом із часом завершення спроби.

Для викладачів створено окремий інтерфейс, де можна додавати запитання, редагувати варіанти відповідей, встановлювати тривалість тесту та дедлайни. Інтерфейс студента зроблено максимально простим: таймер, кількість запитань, прогрес і кнопка завершення тесту. приклад інтерфейсу наведено на (рис. 2).



Рис. 2. Інтерфейс проходження тесту студентом

Висновки та перспективи. У результаті роботи було створено модуль онлайн-тестування, який підтримує різні типи запитань, автоматично оцінює результат і зберігає дані у SQL Server. Модуль інтегровано у веб-орієнтовану LMS-систему та може використовуватися у навчальних закладах для дистанційного навчання. У майбутньому планується додати адаптивне тестування, випадкову генерацію варіантів, розширену аналітику та мобільну версію системи.

Список використаних джерел

1. Microsoft SQL Documentation - SQL Server. Microsoft Learn: Build with answers in reach. [Електронний ресурс]. - Режим доступу: https://learn.microsoft.com/sql/
2. Moodle Documentation. Moodle Project. [Електронний ресурс]. - Режим доступу: https://docs.moodle.org/
3. PHP: Preface - Manual. PHP. [Електронний ресурс]. - Режим доступу: https://www.php.net/manual/
4. Web Standards. W3C. [Електронний ресурс]. - Режим доступу: https://www.w3.org/standards/

Бондаренко Дарина Вадимівна
студентка 4 курсу, групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій
(097) 533-08-82
Dbondv2601@gmail.com

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ НА ОСНОВІ NODE.JS

Постановка задачі. У сучасних умовах в Україні більшість організацій поступово переходять до використання цифрових інструментів у своїй роботі. Це пов'язано з необхідністю пришвидшення процесів, зручності доступу до інформації та зменшення витрат часу на виконання рутинних операцій.

На практиці паперовий документообіг створює ряд труднощів. Передача документів між працівниками займає час, погодження може затягуватись, особливо якщо до процесу залучено кілька людей або підрозділів. Пошук потрібного документа іноді стає складним, особливо при великих обсягах інформації. До того ж витрати на друк і зберігання документів. Усе це в сукупності впливає на ефективність роботи організації. Можна помітити, що навіть прості операції, наприклад передача документа на погодження, можуть займати більше часу, ніж сам процес його створення.

У зв'язку з цим виникає потреба у створенні електронних систем, які дозволяють працювати з документами швидше і зручніше. Такі системи забезпечують централізоване зберігання даних, доступ до них та можливість обміну між користувачами.

Отже завдання розробки веб-застосунку електронного документообігу, який дозволить реалізувати базовий функціонал роботи з документами та перевірити ефективність використання сучасних веб-технологій є актуальним.

Мета дослідження. Метою роботи є розробка веб-застосунку електронного документообігу з використанням платформи Node.js та аналіз застосованих технологічних рішень.

У межах роботи розглядається організація серверної частини, побудова API, підключення до бази даних, реалізація авторизації користувачів, а також завантаження та обробка файлів.

Окрема увага приділяється клієнтській частині застосунку, яка забезпечує взаємодію користувача з системою через веб-інтерфейс.

Під час виконання роботи також ставила перед собою завдання зрозуміти, як різні технології взаємодіють між собою в межах одного застосунку.

Результати дослідження. При виконанні роботи було створено веб-застосунок, який працює за клієнт-серверною моделлю. Основна логіка реалізована на сервері, а користувач взаємодіє із системою через браузер.

Серверна частина побудована на Node.js із використанням Express.js. За допомогою цього інструменту реалізовано обробку HTTP-запитів і створення API. Кожен маршрут відповідає за певну дію, наприклад отримання списку документів або їх створення.

Для роботи з базою даних використано PostgreSQL із бібліотекою pg. У базі зберігається інформація про користувачів і документи. Це дозволяє організувати впорядковану структуру даних та забезпечити зручний доступ до них.

Авторизація користувачів реалізована за допомогою JWT (бібліотека jsonwebtoken). Після входу користувач отримує токен, який використовується для подальших запитів до сервера. Такий підхід дозволяє обмежити доступ до окремих частин системи.

Для роботи з файлами використано бібліотеку multer. Вона дозволяє завантажувати документи на сервер і працювати з ними в межах застосунку. Під час реалізації було враховано базові обмеження щодо типу файлів і їх розміру.

Взаємодія між клієнтом і сервером налаштована за допомогою бібліотеки cors. Це дозволяє уникнути проблем із доступом при виконанні запитів між різними частинами застосунку.

Клієнтська частина реалізована з використанням HTML, CSS та JavaScript. Інтерфейс побудований досить просто, щоб користувач міг виконувати основні дії без додаткових пояснень.

Обмін даними з сервером здійснюється через Fetch API. Це дозволяє надсилати запити (GET, POST, PUT, DELETE), отримувати відповіді та оновлювати інформацію на сторінці без перезавантаження.

Під час виконання роботи стало зрозуміло, що важливо не лише використати окремі технології, а правильно організувати їх взаємодію. Наприклад, передача токена авторизації разом із запитами є обов'язковою для доступу до захищених даних.

З практичної точки зору найбільше відчувається спрощення роботи з документами. Завантаження, перегляд і передача файлів виконуються значно швидше, ніж у паперовому вигляді.

Також у процесі розробки виникали труднощі, пов'язані з налаштуванням запитів і обробкою помилок. Проте їх вирішення допомогло краще зрозуміти принципи роботи веб-застосунків.

На даному етапі реалізовано базовий функціонал, який можна розширювати в подальшому.

Висновок. На мою думку, перехід до електронного документообігу є логічним кроком для сучасних організацій. Використання веб-застосунків дозволяє спростити роботу з документами та зменшити кількість ручних операцій. У результаті виконаної роботи було розроблено веб-застосунок із використанням Node.js, Express.js, PostgreSQL, JWT, multer, cors, а також HTML, CSS і JavaScript.

Отримане рішення дозволяє працювати з документами в електронному вигляді та може бути використане як основа для подальшого розвитку. У подальшому систему можна доповнити новими можливостями залежно від потреб користувачів.

Список використаних джерел

1.Tilkov S., Vinoski S. *Node.js: Using JavaScript to Build High-Performance Network Programs.* IEEE Internet Computing. 2010. Vol. 14, No. 6, P. 80–83.

2. Jones M., Bradley J., Sakimura N. *JSON Web Token (JWT).* IETF RFC 7519. 2015.

3. Stonebraker M., Rowe L.A. *The Design of POSTGRES.* ACM SIGMOD Record. 1986. Vol. 15, No. 2, P. 340–355.

Купріян Валерія Дмитрівна
Студентка 3-го курсу, спеціальності “Комп’ютерні науки”
Одеський національний університет імені І. І. Мечникова
0951488680
valeriya.kupriyan@gmail.com
Камєнєва Алла Вікторівна
Канд. тех. наук, доцент
Одеський національний університет імені І. І. Мечникова

РОЗРОБКА КРОСПЛАТФОРМНОГО ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ АКАДЕМІЧНОЇ ГРУПИ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ .NET MAUI

Постановка задачі. Управління навчальним процесом на рівні академічної групи вимагає постійної координації розкладу, обліку відвідуваності та оперативного обміну навчальними матеріалами, проте існуючі університетські системи часто є перевантаженими та не оптимізованими для швидкої мобільної роботи. Виникає потреба у створенні кросплатформного застосунку, який би об'єднував функції електронного журналу, динамічного розкладу та сховища джерел. Програмне забезпечення має забезпечувати безпечний доступ через корпоративні облікові записи та гарантувати цілісність даних при роботі на різних пристроях під управлінням Android та Windows.

Мета дослідження. Проєктування та реалізація архітектури кросплатформного застосунку на базі .NET MAUI, що інтегрує гібридну систему баз даних (SQL та NoSQL) для забезпечення максимальної гнучкості при управлінні навчальними матеріалами та високої надійності при зберіганні персональних даних користувачів.

Результати дослідження. У ході виконання роботи було спроєктовано та реалізовано масштабовану інформаційну систему “EduFlow” для адміністрування навчального процесу. Розроблена архітектура базується на трирівневій моделі, технічна взаємодія компонентів якої була детально проілюстрована на створеній компонентній діаграмі (рис. 1). Як відображено на схемі, взаємодія користувача із системою починається з рівня View, де навігаційний каркас AppShell [1] забезпечує логічний поділ між робочими просторами студента та старости. Безпека доступу на цьому етапі реалізована через інтеграцію з корпоративними сервісами (Google Workspace / Microsoft Azure AD) за протоколом OAuth 2.0.

Сам презентаційний рівень побудований з використанням адаптивної верстки FlexLayout, що дозволило створити інтерактивні дашборди - центральні хаби застосунку. Вони акумулюють актуальний розклад, прямі посилання на онлайн-зустрічі (Zoom/Meet) та структуровані блоки з джерелами від викладачів. Зв'язок між інтерфейсом та даними (стрілка *Data Binding / Commands* на діаграмі) реалізовано через патерн MVVM. Управління подіями повністю ізольоване від візуальної частини завдяки механізмам Command та CommandParameter. Це дозволяє ViewModel ефективно обробляти складні сценарії користувача - наприклад, одним натисканням на назву дисципліни відкривати повний перелік закріплених за нею онлайн-ресурсів. Важливим результатом дослідження стала реалізація механізму відмовостійкості на рівні Data Service.

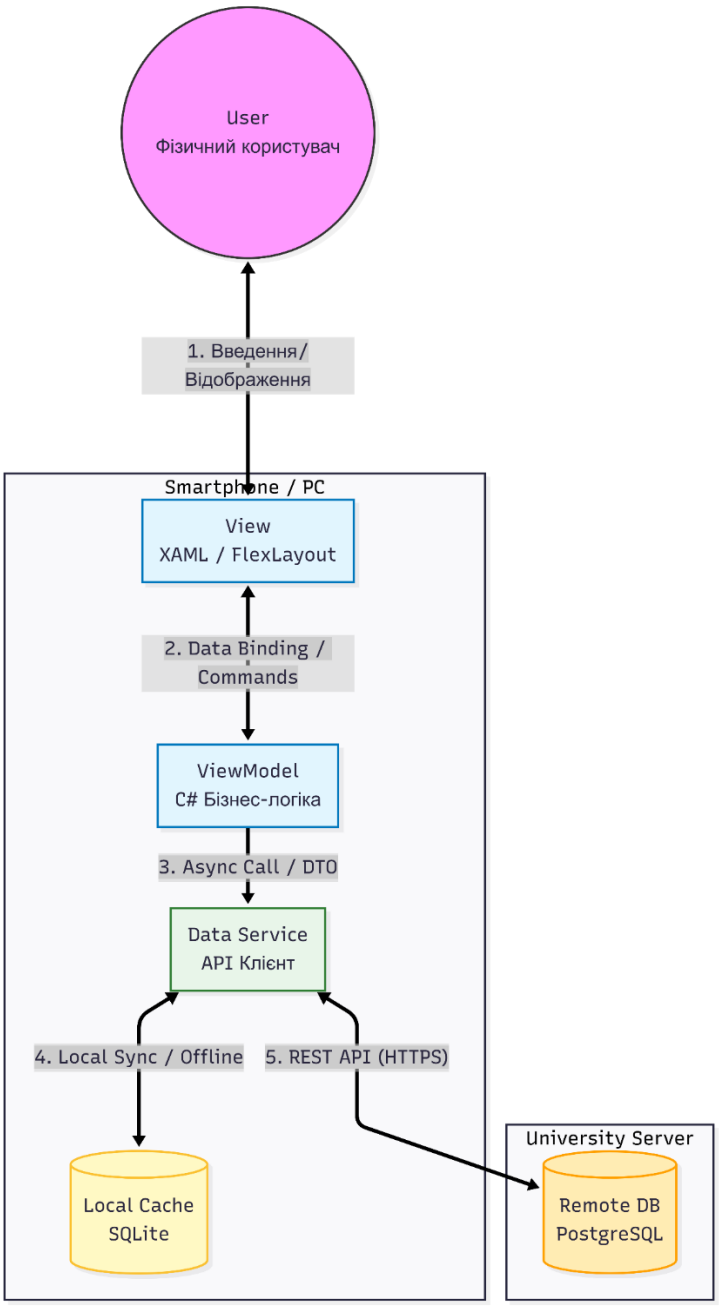


Рис. 1. Компонентна архітектура інформаційної системи “EduFlow”

До системи було інтегровано обробку нативних подій операційних систем (таких як Window.Deactivated та OnPause) [1]. При раптовому згортанні додатка ці події активують механізм “м’якого” збереження стану: система автоматично скеровує останні переглянуті джерела або невідправлені відмітки про відсутність у локальний кеш пристрою (Local Cache / SQLite на схемі). При відновленні сесії (Window.Resumed) дані безшовно відправляються на сервер, що забезпечує безперервність робочого процесу незалежно від якості інтернет-з'єднання [1].

На серверному рівні впроваджено гібридну архітектуру баз даних: реляційна PostgreSQL [2] забезпечує дотримання принципів ACID та цілісність зв'язків для критично важливих структурованих даних (профілі користувачів, журнал відвідуваності), тоді як документоорієнтована MongoDB [3] дозволяє гнучко зберігати динамічний контент (розклади, навчальні матеріали) у вигляді JSON-документів, уникаючи необхідності складних міграцій схем.

Висновки та перспективи. В результаті проведеного дослідження та практичної реалізації було успішно створено кросплатформну інформаційну систему “EduFlow”, призначену для оптимізації управління навчальним процесом академічної групи на базі .NET MAUI.

Поєднання архітектури MVVM, адаптивного інтерфейсу та гібридної моделі баз даних (PostgreSQL і MongoDB) із підходом Offline-First забезпечило високу відмовостійкість застосунку, легкість його супроводу та безперебійну роботу на Android і Windows незалежно від якості інтернет-з'єднання.

Масштабування системи передбачає її розширення до рівня всього факультету шляхом створення ізольованих робочих кабінетів із відповідними правами доступу для викладачів, кураторів та працівників деканату. Це дозволить перетворити локальний застосунок на комплексну платформу для наскрізного моніторингу успішності, автоматизації документообігу та централізованого управління навчальним процесом.

Список використаних джерел

1. What is .NET MAUI. URL: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-10.0
2. PostgreSQL documentation. URL: https://www.postgresql.org/docs/
3. MongoDB documentation. URL:https://www.mongodb.com/docs/

Рябко Юлія Василівна
студентка 4 курсу, групи ІСД-42
Державний університет інформаційно-комунікаційних технологій,
(093)394-58-62
riabkoyuliia@gmail.com

АНАЛІЗ ВРАЗЛИВОСТЕЙ JWT-АВТЕНТИФІКАЦІЇ У ВЕБЗАСТОСУНКАХ НА ОСНОВІ NODE.JS ТА МЕТОДИ ЇХ УСУНЕННЯ

Постановка задачі. Розвиток сучасних вебсервісів супроводжується стрімким зростанням обсягів даних, якими обмінюються клієнтські та серверні частини систем. Некоректно реалізована автентифікація здатна поставити під загрозу безпеку як окремого облікового запису, так і всієї інформаційної системи загалом. Відповідно до класифікації OWASP API Security Top 10 (2023), порушення автентифікації посідає другу позицію серед найбільш критичних ризиків безпеки програмних інтерфейсів. Одним із ключових механізмів автентифікації у сучасних REST API є JSON Web Token — відкритий стандарт RFC 7519, що визначає компактний спосіб передачі інформації між сторонами у вигляді JSON-об'єкту. Незважаючи на широке застосування, гнучкість стандарту щодо вибору алгоритмів і формату claims породжує широкий клас практичних вразливостей, систематизація та усунення яких і є предметом цього дослідження.

Мета дослідження. Метою роботи є виявлення та класифікація типових вразливостей JWT-автентифікації у вебзастосунках на платформі Node.js, а також розробка практичних рекомендацій щодо їх усунення з урахуванням специфіки мови TypeScript.

Результати. Токен JWT складається з трьох частин: заголовка, що визначає алгоритм підпису; корисного навантаження з «claims» про суб'єкт автентифікації; та підпису, що забезпечує цілісність токена. Дослідження, проведені фахівцями Akamai, показали, що зловмисники систематично застосовують три основні сценарії атак: перехоплення та крадіжку токена, підвищення привілеїв через маніпуляцію полями payload, а також атаки на алгоритм підпису.

Серед найпоширеніших вразливостей виявлено: підстановку алгоритму «none», при якій сервер приймає непідписаний токен; плутанину алгоритмів — підміну асиметричного RS256 на симетричний HS256; використання слабких секретних ключів, вразливих до брутфорсу; ін'єкції через поле kid, яке за відсутності санації вхідних даних може стати вектором SQL-ін'єкції або path traversal. Окремо зафіксовано поширену помилку, коли розробники викликають jwt.decode() замість jwt.verify(), що повністю виключає криптографічну перевірку підпису.

У практичній частині проведено порівняльний аналіз підходів до реалізації JWT-автентифікації із застосуванням TypeScript. Встановлено, що оголошення окремих типів для підписаного та непідписаного токена унеможливлює їх ненавмисну взаємозаміну на етапі компіляції. Для безпечного зберігання токенів на стороні клієнта рекомендовано використовувати HttpOnly Cookie замість localStorage, оскільки цей варіант недоступний JavaScript-коду сторінки і виключає крадіжку через XSS-атаку. Застосування пари токенів — короткоживучого access token (15 хвилин) та довгоживучого refresh token — суттєво скорочує вікно можливого зловживання у разі компрометації.

Висновки. На підставі проведеного аналізу сформульовано практичні рекомендації для розробників Node.js API: явно вказувати дозволений алгоритм підпису при верифікації (наприклад, { algorithms: ['RS256'] }), не зберігати конфіденційних даних у payload, встановлювати мінімальний термін дії access token, валідувати поле kid перед використанням у запитах до бази даних, регулярно ротувати секретні ключі та проводити аудит залежностей JWT-бібліотек.

JWT залишається практичним і широко застосовним інструментом автентифікації, однак його безпека повністю залежить від коректності реалізації. Поєднання суворої типізації TypeScript, явного визначення алгоритму верифікації, безпечної стратегії зберігання токенів та механізму refresh rotation формує комплексний захист REST API від найпоширеніших векторів атак, зафіксованих у класифікації OWASP. Перспективним напрямком подальших досліджень є впровадження механізму Proof-of-Possession (PoP) токенів для протидії атакам на перехоплення у середовищах із підвищеними вимогами до безпеки.

Список використаних джерел

1. OWASP API Security Top 10 2023. API2:2023 Broken Authentication. Open Web Application Security Project. URL: https://owasp.org/API-Security/editions/2023/en/0xa2-broken-authentication/
2. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. Internet Engineering Task Force (IETF). 2015. URL: https://datatracker.ietf.org/doc/html/rfc7519
3. Namer N. Analyzing Broken User Authentication Threats to JSON Web Tokens. Akamai Security Research. 2023. URL: https://www.akamai.com/blog/security-research/owasp-authentication-threats-for-json-web-token
4. JWTs Under the Microscope: How Attackers Exploit Authentication and Authorization Weaknesses. Traceable AI Blog. 2024. URL: https://www.traceable.ai/blog-post/jwts-under-the-microscope-how-attackers-exploit-authentication-and-authorization-weaknesses
5. Sheffer Y., Hardt D., Jones M. JSON Web Token Best Current Practices. RFC 8725. IETF. 2020. URL: https://datatracker.ietf.org/doc/html/rfc8725

Рябко Юлія Василівна
студентка 4 курсу, групи ІСД-42
Державний університет інформаційно-комунікаційних технологій
(093)394-58-62
riabkoyuliia@gmail.com

МЕТОДИ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ СЕРВЕРНИХ ЗАСТОСУНКІВ НА ОСНОВІ NODE.JS

Постановка задачі. У сучасному середовищі веброзробки питання продуктивності серверних застосунків набуває дедалі більшої актуальності. Node.js — асинхронне середовище виконання JavaScript на базі рушія V8 — здобуло широке поширення завдяки неблокувальній моделі вводу-виводу та єдиній мові програмування для клієнтської і серверної частини. Проте зі зростанням навантаження розробники стикаються з блокуванням циклу подій, неефективним використанням пам'яті та недостатнім завантаженням багатоядерних процесорів. Відсутність системного підходу до усунення цих вузьких місць призводить до деградації продуктивності та нестабільної роботи застосунків під реальним навантаженням.

Мета дослідження

Метою роботи є систематизація та аналіз ключових методів оптимізації продуктивності серверних застосунків на платформі Node.js, а також оцінка їхньої ефективності з урахуванням особливостей однопотокової архітектури середовища виконання.

Результати. Ключовим аспектом оптимізації є запобігання блокуванню циклу подій. Оскільки Node.js є однопотоковим середовищем, виконання ресурсомістких синхронних операцій призводить до затримки обробки нових запитів. Для вирішення цієї проблеми рекомендується виносити тривалі обчислення у Worker Threads, стабілізовані у Node.js 12: модуль worker_threads дозволяє виконувати JavaScript у паралельних потоках без блокування основного потоку.

Ефективне керування пам'яттю є критичним чинником продуктивності. Витоки виникають через накопичення замикань, невикористаних обробників подій або тривалих посилань у глобальних об'єктах. Для їхнього виявлення застосовуються вбудований профайлер V8, прапорець --inspect, а також зовнішні інструменти — Clinic.js та 0x. Правильне управління пам'яттю може знизити споживання ресурсів на 30–50% у довготривалих застосунках.

Для горизонтального масштабування застосовується вбудований модуль cluster, що запускає кілька дочірніх процесів і розподіляє навантаження між ними. Альтернативою є PM2 — менеджер процесів з підтримкою кластеризації та моніторингу. Бенчмарки засвідчують майже лінійне зростання пропускної здатності при збільшенні кількості воркерів на багатоядерних серверах.

Кешування є одним із найефективніших підходів до зниження латентності. На рівні застосунку застосовуються in-memory сховища Redis та Memcached, що дозволяють уникнути повторних звернень до бази даних; на рівні HTTP — заголовки кешування та зворотні проксі Nginx або Varnish. Впровадження багаторівневої стратегії кешування може скоротити середній час відповіді сервера більш ніж у п'ять разів.

Оптимізація запитів до бази даних охоплює використання пулів з'єднань, підготовлених запитів та індексів. При роботі з PostgreSQL бібліотека pg разом із pg-pool забезпечує стабільно низьку затримку навіть при значному паралельному навантаженні.

Висновки. Комплексна оптимізація Node.js-серверів охоплює управління циклом подій, профілювання пам'яті, кластеризацію, кешування та ефективну взаємодію з базами даних. Поєднання цих методів дозволяє досягти суттєвого підвищення продуктивності та стабільності вебзастосунків при реальних умовах навантаження. Перспективним напрямком подальших досліджень є порівняльний аналіз продуктивності Node.js із альтернативними серверними платформами — зокрема Bun та Deno — в умовах високого паралельного навантаження.

Список використаних джерел

1. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing. 2010. Vol. 14, No. 6. P. 80–83. URL: https://doi.org/10.1109/MIC.2010.145
2. Cantelon M., Harter M., Holowaychuk T., Rajlich N. Node.js in Action. 2nd ed. Manning Publications, 2017. 392 p.
3. Oppermann F. Mastering Node.js. 2nd ed. Packt Publishing, 2018. 534 p. URL: https://www.packtpub.com/product/mastering-nodejs-second-edition/9781785888960
4. Node.js Foundation. Node.js Cluster Documentation. URL: https://nodejs.org/api/cluster.html (дата звернення: 01.04.2025).

Сіроєдов Валерій Олександрович
студент 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій м. Київ
+380632705110
siroedovvv@gmail.com

АВТОМАТИЗАЦІЯ ОБЛІКУ СИЛОВИХ ТРЕНУВАНЬ ТА ДОСЯГНЕНЬ ЗА ДОПОМОГОЮ ВЕБ-ТЕХНОЛОГІЙ

Постановка задачі.

Записувати результати тренувань у блокнот або телефонні нотатки - цілком звичайна практика для більшості людей, які займаються спортом. Але такий підхід має очевидні мінуси: незручно шукати дані за минулі тижні, важко порівняти результати в різних вправах і зовсім неможливо швидко визначити, чи є взагалі прогрес. Готові застосунки на ринку переважно або надто перевантажені функціями, або вимагають платної підписки для доступу до елементарної статистики. Тому виникла ідея зробити простий і зрозумілий веб-інструмент, який закриє основні потреби людини, що займається силовими тренуваннями: фіксувати підходи, переглядати динаміку та знати свої особисті рекорди.

Мета дослідження.

Розробити веб-застосунок, який дозволяє зручно вести тренувальний щоденник, фіксувати вагу та кількість повторів у кожній вправі, переглядати історію тренувань і автоматично відстежувати персональні рекорди.

Результати дослідження.

За основу взято стандартний підхід до веб-розробки: є серверна частина, яка обробляє запити і працює з базою даних, і є клієнтська частина - те, що бачить користувач у браузері. Такий поділ дозволяє незалежно змінювати логіку роботи системи та її зовнішній вигляд [2].

Для серверної частини обрано мову Python і фреймворк Django. Головна перевага Django - він вже містить усе необхідне: роботу з базою даних, систему реєстрації та входу, обробку форм і генерацію HTML-сторінок. Це значно скорочує час розробки і не потребує підключення десятків сторонніх бібліотек [4].

Дані зберігаються у реляційній базі даних SQLite, яка входить до складу Python і не вимагає окремого встановлення. Структура бази складається з шести таблиць:

– користувачі (User) - стандартна модель Django;

– вправи (Exercise) - назва, категорія, опис, посилання на автора;

– тренування (Workout) - дата, назва, нотатки;

– вправи в тренуванні (WorkoutExercise) - зв'язок між тренуванням та вправою;

– підходи (Set) - вага, кількість повторів, порядковий номер;

– персональні рекорди (PersonalRecord) - максимальна вага для кожної вправи.

Щоб не писати SQL-запити вручну, використовується ORM (Object-Relational Mapping) - механізм, який дозволяє звертатись до бази даних через звичайні Python-об'єкти [3].

Застосунок реалізує такий основний функціонал:

– реєстрація та авторизація - кожен користувач бачить лише свої дані;

– бібліотека вправ із фільтрацією за категорією та пошуком за назвою;

– створення тренування, додавання вправ і фіксація підходів із вагою та кількістю повторів;

– автоматичне оновлення персонального рекорду при встановленні нового максимуму - без ручного втручання;

– перегляд усієї історії тренувань та статистики по кожній вправі.

Коли користувач зберігає новий підхід, система порівнює вагу з поточним рекордом. Якщо нова вага більша - рекорд оновлюється автоматично і відображається повідомлення з позначкою нового досягнення. Це реалізовано через окрему функцію, яка викликається щоразу при збереженні підходу.



Рис. 1. Загальна структура взаємодії елементів веб-застосунку

Висновки.

У результаті роботи створено повноцінний веб-застосунок GymTracker для ведення тренувального щоденника. Він вирішує реальну практичну задачу: замінює паперові записи зручним цифровим інструментом, де вся інформація зберігається в одному місці. Застосунок побудований на технологіях, які легко масштабуються, тому в майбутньому його можна розширити - наприклад, додати графіки прогресу, шаблони тренувань або можливість порівнювати результати з іншими користувачами.

Список використаних джерел

1. Elmasri, R., & Navathe, S. B. (2016). Fundamentals of database systems (7th ed.). Pearson.

2. Richards, M., & Ford, N. (2020). Fundamentals of software architecture: An engineering approach. O’Reilly Media.

3. Kleppmann, M. (2017). Designing data-intensive applications. O’Reilly Media.

4. Django Software Foundation. (2026). Django documentation. Retrieved April 13, 2026, from https://docs.djangoproject.com/
5. Richards, M., & Ford, N. (2020). Fundamentals of software architecture: An engineering approach. O'Reilly Media.

Опанасенко Андрій Юрійович
студент 4 курсу, групи ТЦР-42
Державного університету інформаційно-комунікаційних технологій
(093)717-26-64
aopanasenko279@gmail.com

АНАЛІЗ ВРАЗЛИВОСТЕЙ ПРОТОКОЛУ OAUTH 2.0 У ВЕБЗАСТОСУНКАХ ТА МЕТОДИ ЗАХИСТУ ВІД ТИПОВИХ АТАК

Постановка задачі. OAuth 2.0 є домінуючим протоколом делегованої авторизації у сучасному вебі: його підтримують Google, Microsoft, Facebook, GitHub та тисячі інших платформ. Водночас гнучкість стандарту, закріплена у RFC 6749 [1], обертається на слабкість: протокол свідомо залишає низку параметрів - зокрема валідацію redirect_uri та реалізацію параметра state - на розсуд розробника, що відкриває широке поле для помилок. Академічний аналіз 395 популярних сайтів виявив, що близько 36% з них залишаються вразливими до CSRF-атак на OAuth-потік через відсутність або некоректну реалізацію захисного параметра state [2]. Ці дані вказують на системну проблему: більшість порушень безпеки OAuth спричинені не вадами специфікації, а помилками реалізації. Таким чином, актуальним є завдання систематизації типових вразливостей OAuth 2.0 і формування конкретних рекомендацій для їх усунення в контексті сучасних Node.js-вебзастосунків.

Мета дослідження. Метою роботи є систематизація вразливостей протоколу OAuth 2.0, характерних для вебзастосунків на платформі Node.js, а також розроблення та верифікація практичних заходів щодо їх усунення на основі чинних рекомендацій OWASP [3] і стандартів безпеки IETF. Для досягнення мети вирішуються такі задачі: класифікація типових векторів атак на OAuth-реалізації; порівняльний аналіз вразливого та захищеного варіантів потоку Authorization Code; розроблення та тестування прототипу захищеного OAuth-сервісу на базі Node.js і TypeScript із інтеграцією PostgreSQL для зберігання refresh-токенів.

Результати. У ході дослідження виокремлено чотири основні класи вразливостей OAuth 2.0. По-перше, маніпуляція параметром redirect_uri: якщо сервер авторизації не звіряє URI запиту з попередньо зареєстрованим значенням за точним збігом, зловмисник може перенаправити авторизаційний код на підконтрольний ресурс. Аналіз дослідників PortSwigger Web Security Academy підтверджує, що підміна redirect_uri є одним із найпоширеніших векторів компрометації OAuth [4]. По-друге, відсутність або слабка перевірка параметра state призводить до вразливості до CSRF, коли атакуючий може прив'язати свій OAuth-сеанс до облікового запису жертви. По-третє, використання застарілого Implicit Flow, виключеного зі специфікації OAuth 2.1, передає access token безпосередньо у URI фрагменті, де він може бути перехоплений через Referer-заголовок або браузерну історію. По-четверте, дослідження Security Boulevard, що охопило вразливості 2024–2025 років, зафіксувало атаки типу consent phishing через підроблені OAuth-застосунки, що імітують верифіковані клієнти відомих платформ [5].

На підставі проведеного аналізу реалізовано прототип захищеного OAuth-сервісу. Прийняті проєктні рішення: (1) виключно Authorization Code Flow з PKCE - відповідно до вимог RFC 7636 - для всіх публічних клієнтів; (2) суворе порівняння redirect_uri за рівністю рядка без підстановочних символів; (3) генерація параметра state як криптографічно стійкого nonce, прив'язаного до cookie сесії на серверному рівні; (4) зберігання refresh-токенів у таблиці PostgreSQL з фіксацією client_id, user_id, терміну дії та хешу пристрою; (5) обов'язкова HTTPS-передача та атрибути HttpOnly і SameSite=Strict для cookie з токенами. Тестування прототипу за допомогою OWASP ZAP підтвердило, що реалізовані заходи нейтралізують усі чотири описані вектори атак.

Висновки. Проведений аналіз засвідчив, що переважна більшість інцидентів безпеки, пов'язаних із OAuth 2.0, є наслідком помилок реалізації, а не вад самого стандарту. Ключовими засобами захисту є: перехід на Authorization Code Flow з PKCE, точна валідація redirect_uri, коректне використання параметра state та безпечне зберігання refresh-токенів у реляційній базі даних із підтримкою їх відкликання. Статична типізація TypeScript допомагає запобігти підміні типів токенів на рівні компілятора і знижує ймовірність помилок, що призводять до вразливостей. Отримані результати можуть бути безпосередньо застосовані при розробці вебзастосунків на платформі Node.js, де інтеграція із зовнішніми постачальниками ідентифікації здійснюється засобами OAuth 2.0.

Список використаних джерел

1. Hardt D. (Ed.) The OAuth 2.0 Authorization Framework. RFC 6749. Internet Engineering Task Force (IETF). 2012. URL: https://datatracker.ietf.org/doc/html/rfc6749
2. Singh J., Chaudhary N. K. OAuth 2.0: Architectural Design Augmentation for Mitigation of Common Security Vulnerabilities. Journal of Information Security and Applications. 2022. URL: https://www.sciencedirect.com/science/article/abs/pii/S2214212621002684
3. OWASP API Security Top 10 2023. API2:2023 Broken Authentication. Open Web Application Security Project. URL: https://owasp.org/API-Security/editions/2023/en/0xa2-broken-authentication/
4. OAuth 2.0 Authentication Vulnerabilities. PortSwigger Web Security Academy. URL: https://portswigger.net/web-security/oauth
5. Security Vulnerabilities in SAML, OAuth 2.0, OpenID Connect, and JWT. Security Boulevard. 2025. URL: https://securityboulevard.com/2025/04/security-vulnerabilities-in-saml-oauth-2-0-openid-connect-and-jwt/

Зиков Михайло Віталійович
студент 4 курсу
групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій м. Київ
zykovmihail35@gmail.com

Гавор Артур Станіславович
викладач кафедри Технологій цифрового розвитку
Державний університет інформаційно-комунікаційних технологій м. Київ
a.havor@duikt.edu.ua

ПРОГРАМНА РЕАЛІЗАЦІЯ ДЕСКТОПНОГО ЗАСТОСУНКУ CINEMAHUB ДЛЯ КЕРУВАННЯ КАТАЛОГОМ ФІЛЬМІВ ЗАСОБАМИ C++ ТА QT 6

Постановка задачі. Сучасні програмні продукти для роботи з кіноконтентом або цілком залежать від мережевого з'єднання, або потребують складного налаштування. Жоден із доступних інструментів не поєднує офлайн-режим, зручний сучасний інтерфейс та повноцінне керування особистою кінотекою в одному рішенні. Задача полягає в тому, щоб спроєктувати і реалізувати повнофункціональний десктопний застосунок, який вирішує цю проблему без залежності від зовнішніх сервісів.

Мета дослідження. Реалізувати десктопний застосунок CinemaHub засобами C++17 і Qt 6, що забезпечує повноцінну роботу в офлайн-режимі, включаючи перегляд каталогу фільмів, пошук з автодоповненням, відкриття трейлерів, управління профілем та перемикання тем інтерфейсу, і провести його тестування.

Результати. Архітектура застосунку побудована на основі незалежних компонентів. Кожен клас відповідає за один функціональний блок - принцип єдиної відповідальності, що є стандартним підходом у компонентному проєктуванні програмних систем [4]. Взаємодія між класами побудована виключно через механізм сигналів і слотів Qt [3] - це виключає пряму залежність між компонентами і дозволяє змінювати або розширювати будь-який блок незалежно від решти.

Заставка при запуску реалізована через клас IntroSplash із використанням QPropertyAnimation: логотип плавно з'являється і зникає протягом кількох секунд, після чого відкривається вікно авторизації. Вікно авторизації - клас LoginWindow - містить дві форми: вхід і реєстрацію, які перемикаються без закриття вікна. Перевірка облікових даних виконується через UserManager. Після успішного входу LoginWindow надсилає сигнал accepted(), і головне вікно відкривається автоматично [2].

Навігація між розділами реалізована через QStackedWidget у класі MainWindow [3]. При натисканні кнопки меню відповідна сторінка активується без перебудови інтерфейсу. Головна сторінка містить банерний карусель, що автоматично змінює слайди через QTimer, і чотири горизонтальних ряди карток: «Популярне зараз», «Нове та гаряче», «Топ за рейтингом» і «Рекомендовано для тебе». Кожен ряд формується з даних каталогу за власним алгоритмом вибірки і реалізований через клас ScrollRow з підтримкою плавного прокручування [2].

Пошук з автодоповненням налаштований через QCompleter у режимі MatchContains - підказки фільтруються за входженням, а не лише за початком назви [3]. Результати пошуку оновлюються після кожного введеного символу без натискання Enter. Реалізовано логіку пошуку з миттєвим оновленням результатів: при порожньому запиті відображається головна сторінка, при наявності результатів формується список карток MovieCard.

Сторінка фільму реалізована у класі MovieDetailsWindow і містить інформацію про фільм, ряд схожих стрічок і блок відгуків. Трейлери відкриваються через QDesktopServices::openUrl() - застосунок передає посилання на YouTube системному браузеру. Якщо для фільму немає посилання у словнику, запускається резервний відеоплеєр на базі QMediaPlayer [2].

Перемикання між темною і світлою темами реалізовано через єдиний виклик setStyleSheet() на рівні головного вікна - стиль успадковується всіма дочірніми елементами [3], тому зміна теми відбувається миттєво без перезапуску. Профіль користувача реалізовано у класі ProfileWindow з можливістю завантаження фото, зміни нікнейму і вибору кольору аватара.

Всі дані каталогу зберігаються у вихідному коді у вигляді структури Movie і надаються через модуль MovieData. Список обраних фільмів зберігається у QSet<int>, що гарантує унікальність і забезпечує перевірку наявності за константний час O(1). Для роботи з контейнерами використано можливості стандарту C++17 [1]. Постери фільмів скомпільовані у виконуваний файл через Qt Resource System [3].

Для перевірки якості реалізації проведено два види тестування [5]. Функціональне тестування охопило 15 тест-кейсів за сценаріями авторизації, пошуку, перегляду фільмів, роботи з обраним, перемикання теми і навігації між розділами. Всі 15 тест-кейсів пройдено успішно. Модульне тестування перевірило три ключові функції в ізоляції: searchMovies() - пошук з урахуванням регістру, trailerUrl() - повернення посилань зі словника і порожнього рядка при відсутності запису, onLogin() - обробка коректних і некоректних даних, а також порожніх полів. Дев'ять модульних тестів пройдено без відхилень [5].

Окремо перевірялась сумісність інтерфейсу на роздільних здатностях 1280x720 і 1920x1080, а також при масштабуванні системи 125% і 150%. У всіх перевірених умовах застосунок відображався коректно: картки, банер і навігаційна панель масштабувались без артефактів завдяки вбудованій підтримці High-DPI в Qt 6 [2].

Висновки. Реалізований застосунок CinemaHub виконує всі заявлені функції: повноцінна робота в офлайн-режимі, сучасний інтерфейс з темною і світлою темою, пошук з автодоповненням, відкриття трейлерів, керування профілем і персональний список обраного. Компонентна архітектура на основі сигналів і слотів Qt забезпечила незалежність модулів і спростила тестування. Усі 24 тести - 15 функціональних і 9 модульних - пройдено успішно. Перспективами подальшого розвитку є інтеграція з онлайн-базами даних фільмів, хмарна синхронізація профілю і підтримка кількох облікових записів.

Список використаних джерел

1. Josuttis N. M. C++17 - The Complete Guide. NicoJosuttis, 2019. 600 p.

2. Khoroshev D. Qt 6 C++ GUI Programming Cookbook. 2nd ed. Packt Publishing, 2022. 390 p.

3. Qt Group. Qt Widgets Overview. Qt 6 Documentation. 2024. URL: https://doc.qt.io/qt-6/qtwidgets-index.html

4. Sommerville I. Software Engineering. 10th ed. Pearson Education, 2016. 816 p.

5. Wohlin C., Runeson P., Host M., Ohlsson M. C., Regnell B. Experimentation in Software Engineering. Springer, 2024. 350 p.

Іванова Євгенія Романівна
студентка 4 курсу, групи ТЦР-43
Державний університет інформаційно-комунікаційних технологій
(068) 637-30-18
ivanova.evgeniya21@icloud.com

РОЗРОБКА ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Постановка задачі. Наразі веб-додатки є важливими у повсякденному житті людей, які взаємодіють з цифровими послугами. Продуктивність веб-додатку в першу чергу визначається його інтерфейсом користувача (UI) та досвідом користувача (UX). Поєднання цих двох елементів – UI та UX – визначає, наскільки швидко, легко чи ефективно користувачі можуть виконувати свої відповідні функції.

Хоча технології продовжують змінюватися, багато веб-сайтів залишаються складними у використанні; вони мають складні у використанні системи навігації, є незручними, погано адаптуються до ваших потреб та не є інтуїтивно зрозумілими. В результаті цих проблем знижується ефективність взаємодії користувачів із програмним забезпеченням та знижується загальна якість користувацького досвіду.

Згідно з книгами з дизайну, написаними Дональдом Норманом, інтерфейс має бути легким для розуміння, логічно розробленим та відповідати очікуванням користувачів. Окрім цього, існує стандарт ISO 9241-210, який містить правила для створення систем, зручних для користувача та спрощуючи взаємодію з програмним забезпеченням.

Це підкреслює важливість усього процесу проектування кращих інтерфейсів.

Мета дослідження. Дослідження спрямоване на аналіз процесів, пов'язаних зі створенням користувацьких інтерфейсів та взаємодії з користувачем (UI/UX), пов'язаних з веб-додатками, та їх вплив на покращення взаємодії користувачів. Для досягнення мети необхідно:

- дослідити теоретичні основи UI/UX дизайну;
- проаналізувати сучасні принципи побудови інтерфейсів;
- визначити ключові фактори, що впливають на юзабіліті;
- розглянути процес проєктування інтерфейсів;
- оцінити ефективність UX-підходів у вебзастосунках.

Результати дослідження. Дизайн інтерфейсу користувача (UI/UX) є вирішальним елементом розробки програмного забезпечення, оскільки він визначає якість взаємодії користувача з системою. Фундаментальні принципи ефективного інтерфейсу – це простота, зрозумілість, логічна структура та адаптивність.

Згідно з Interaction Design Foundation (IDF), користувацький досвід залежить від інтуїтивної взаємодії, зменшеного когнітивного навантаження та узгоджених інтерфейсних рішень.

Процес дизайну UI/UX включає такі кроки:

1. Аналіз потреб користувача;

2. Побудова інформаційної архітектури;
3. Створення вайрфреймів та прототипів;
4. Тестування зручності використання;
5. Оптимізація інтерфейсу.

Сучасні інструменти дизайну, такі як Figma, дозволяють створювати інтерактивні прототипи інтерфейсів, тим самим полегшуючи тестування сценаріїв користувача з етапу проектування.

Адаптивний дизайн є важливим для забезпечення правильного відображення інтерфейсу на різних пристроях та покращення зручності використання системи.

Висновки та перспективи. У роботі розглянуто основні підходи до проєктування UI/UX інтерфейсів вебзастосунків та їх вплив на ефективність користувацької взаємодії в умовах сучасних інформаційних систем.Встановлено, що застосування принципів користувацько-орієнтованого дизайну, описаних у працях Donald Norman та сучасних дослідженнях Interaction Design Foundation, дозволяє значно покращити якість взаємодії користувача з цифровими продуктами, зменшити когнітивне навантаження та підвищити інтуїтивність інтерфейсу.Отримані результати підтверджують, що UX-дизайн є критично важливою складовою розробки програмного забезпечення, яка безпосередньо впливає на ефективність, зручність, доступність та інтуїтивність програмних систем.Перспективою подальших досліджень є розвиток адаптивних та персоналізованих інтерфейсів, а також впровадження сучасних UX-підходів і методів у процес проєктування та розробки вебзастосунків з урахуванням поведінкових моделей користувачів.

Список використаних джерел

1. Norman D. *The Design of Everyday Things*. — Basic Books, 2013. https://en.wikipedia.org/wiki/The_Design_of_Everyday_Things
2. ISO 9241-210: Human-centred design for interactive systems. https://en.wikipedia.org/wiki/ISO_9241
3. Interaction Design Foundation. UX Design: The Beginner’s Guide. https://www.interaction-design.org/literature/topics/ux-design
4. Interaction Design Foundation. The Encyclopedia of Human-Computer Interaction.
https://www.interaction-design.org/literature

Герцюк Микола Модестович
доктор філософії, доцент кафедри Технологій цифрового розвитку
Державний університет інформаційно-комунікаційних технологій
m.gertsiuk@duikt.edu.ua

ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ ВИКОРИСТАННЯ ГЛОБАЛЬНОГО РЕПОЗИТОРІЮ PUBCHEM, ЯК ФУНДАМЕНТАЛЬНОГО ДЖЕРЕЛА ІНФОРМАЦІЇ ДЛЯ СИСТЕМИ ОЦІНКИ ТОКСИЧНОСТІ ТА НЕБЕЗПЕКИ РЕЧОВИН EXCOMPAN

Актуальність. Цифровізація хімічного контролю вимагає переходу від статичних локальних рішень до динамічних інтегрованих систем, що зумовлює актуальність розробки алгоритмів взаємодії інформаційної системи ExComPan (система оцінки токсичності та небезпеки речовин) із глобальними репозиторіями даних. Основною проблемою використання виключно внутрішньої бази даних є її швидка деактуалізація, висока ресурсомісткість при ручній верифікації нових сполук та ризик виникнення похибок через обмеженість локальних вибірок. Закриті внутрішні бази, як правило, позбавлені можливості оперативного реагування на оновлення міжнародних регламентів безпеки. Додатковим викликом є проблема критичного дефіциту верифікованих токсикологічних даних про новосинтезовані сполуки. Це створює інформаційні розриви при проведенні оцінки.

Сучасні вимоги до систем оцінки хімічної безпеки, таких як сервіс ExComPan, передбачають оперування верифікованими масивами даних. Використання глобального репозиторію хімічних сполук PubChem дозволяє інтегрувати у внутрішні алгоритми системи понад 110 мільйонів унікальних хімічних структур [1], що забезпечує репрезентативність вибірки при проведенні експертизи маловідомих сполук.

Мета дослідження. Таким чином, метою дослідження є обґрунтування доцільності інтеграції інформаційної системи ExComPan з глобальним репозиторієм PubChem.

Результати дослідження. Інтеграція PubChem обумовлена наступними положеннями:

1. PubChem може бути використаний, як мета-агрегатор, що консолідує дані від провідних регуляторних інституцій. Для інформаційної системи ExComPan це створює багаторівневу систему верифікації. Автоматичне отримання значень токсичності з репозиторію підвищує статистичну достовірність підсумкового індексу небезпеки, як таких даних, що пройшли валідацію на міжнародному рівні.
2. Використання протоколів PUG REST та PUG View[2] (PubChem API) дозволяє реалізувати в архітектурі сервісу модуль динамічного отримання даних. Це мінімізує похибки ручного введення та забезпечує відповідність внутрішньої бази системи актуальним змінам у міжнародних реєстрах хімічних речовин у режимі реального часу.
3. Інтеграція з розділами PubChem, присвяченими класифікації та маркуванню небезпечних речовин, дозволяє системі ExComPan автоматично генерувати звіти, що відповідають узгодженій на глобальному рівні системі класифікації та маркування хімічних речовин. Це забезпечує юридичну та

технічну сумісність висновків системи з міжнародними вимогами до безпеки праці та екологічного моніторингу [3].

Перераховані фактори підтверджують доцільність інтеграції репозиторію PubChem, як джерела інформації системи ExComPan та надають базові шляхи для її здійснення.

Висновки. Впровадження PubChem як базового інформаційного донору для системи ExComPan є науково обґрунтованим кроком, що дозволяє досягти високого рівня об’єктивності, автоматизації та міжнародної стандартизації в процесі оцінки хімічної небезпеки речовин.

Список використаних джерел

1. Kim, S., Chen, J., Cheng, T. et al. (2023). PubChem 2023 update: improved access to chemical and biological information. Nucleic Acids Research, 51(D1), D1373–D1380. URL: https://doi.org/10.1093/nar/gkac956.

2. Kim, S., Thiessen, P.A., Cheng, T. et al. PUG-View: programmatic access to chemical annotations integrated in PubChem. J Cheminform 11, 56 (2019). URL: https://jcheminf.biomedcentral.com/articles/10.1186/s13321-019-0375-2.

3. National Center for Biotechnology Information. (2024). PubChem Classification Browser. URL: https://pubchem.ncbi.nlm.nih.gov/classification/.

Олейніков Іван Анатолійович
викладач кафедри Технологій цифрового розвитку
доктор філософії зі спеціальності 123 Комп'ютерна інженерія
Державний університет інформаційно-комунікаційних технологій
(097) 019-23-80
tigerbrostalker@gmail.com

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ІНТЕЛЕКТУАЛЬНОГО АВТОМАТИЗОВАНОГО КЕРУВАННЯ FDM/FFF-3D-ДРУКОМ НА БАЗІ RASPBERRY PI 5

Постановка задачі. Практичне впровадження інтелектуального керування процесом FDM/FFF-3D-друку потребує не лише побудови моделей виявлення дефектів, а й створення цілісної інформаційної технології, здатної працювати на вбудованій платформі з обмеженими обчислювальними ресурсами. Основна проблема полягає в узгодженні вимог до своєчасного виявлення аномалій, стабільності програмно-апаратного комплексу, безпеки доступу, безперервності роботи та інтеграції з відкритими прошивками 3D-принтерів. У межах розділу 3 дисертації ця задача розв'язується шляхом проєктування хост-рівня системи керування, вибору апаратної платформи, побудови програмного середовища, інтеграції телеметричних і візуальних каналів та експериментального підтвердження працездатності комплексу.

Мета дослідження. Метою роботи є розроблення інформаційної технології інтелектуального автоматизованого керування процесом FDM/FFF-3D-друку на базі Raspberry Pi 5, яка забезпечує інтеграцію модуля штучного інтелекту в замкнений наглядовий контур, підтримує збір і аналіз телеметрії, реалізує політики втручання, гарантує спостережуваність стану системи та зберігає працездатність за ресурсних і часових обмежень вбудованої платформи.

Результати. У розділі 3 сформовано структуру інформаційної технології, у якій Raspberry Pi 5 виконує функції хост-рівня, тоді як контури жорсткого реального часу залишаються на мікроконтролері принтера. Такий розподіл дає змогу відокремити критичні функції інтерполяції траєкторій, керування приводами та підтримання температури від задач інференсу, аналізу телеметрії, вебкерування та журналювання. Практично це забезпечує зниження ризику конфліктів між рівнями керування й підвищує надійність функціонування системи.

Експериментальна оцінка апаратних ресурсів показала, що зі збільшенням роздільної здатності вхідного зображення час інференсу моделі на Raspberry Pi 5 зростає квазілінійно. Для квадратних кадрів від 64 × 64 до 256 × 256 пікселів середній час інференсу збільшується від 3,2 до 18,4 мс. Особливий практичний інтерес має діапазон до 160 × 160 пікселів, у якому середній час інференсу не перевищує 9,3 мс, що забезпечує достатній запас продуктивності для періодичного візуального контролю процесу друку та одночасної роботи сервісних модулів хост-системи.

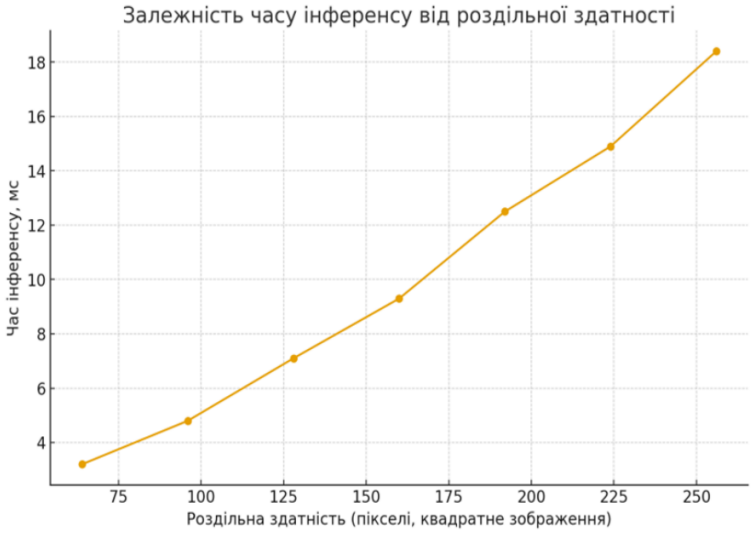


Рис. 1. Залежність часу інференсу від роздільної здатності вхідного зображення

На основі додаткового експерименту зі статичними квадратними зображеннями встановлено, що одночасно зі зростанням часу інференсу підвищується і навантаження на центральний процесор, а також обсяг зайнятої оперативної пам'яті. Отримані результати свідчать, що для режимів безперервного моніторингу доцільно обмежуватися такими роздільними здатностями, які залишають запас обчислювальної потужності для служб обміну даними, вебінтерфейсу та підсистем журналювання.

Таблиця 1. Завантаження CPU та використання RAM під час інференсу залежно від роздільної здатності квадратного вхідного зображення

Роздільна здатність, пікселі	Завантаження CPU, %	Використання RAM, МБ
64 × 64	28	180
96 × 96	35	210
128 × 128	43	245
160 × 160	51	280
192 × 192	60	320
224 × 224	68	360
256 × 256	77	405

Інтеграція модуля штучного інтелекту в замкнений наглядовий контур побудована так, що мультимодальні спостереження перетворюються на оцінку ризику, після чого виконується її верифікація та формується керувальна дія відповідно до політики втручання. Така дія може набувати форми журналювання, попередження, корекції швидкості, температури або інтенсивності обдуву, а в разі високого ризику — паузи чи аварійної зупинки друку. Це переводить систему від пасивного спостереження до активного інтелектуального керування технологічним процесом.

У підсумку розділ 3 дисертації узагальнює три взаємопов'язані наукові результати: удосконалену модель інтелектуального наглядового контролю, метод інтелектуального автоматизованого керування процесом 3D-друку та модель адаптивного вибору керувальних рішень. Практичне значення цих результатів полягає у підвищенні точності діагностики стану обладнання, своєчасному виявленні потенційної дефектності, адаптивній корекції параметрів друку та оперативній локалізації джерел технологічних аномалій у промислових і навчально-лабораторних умовах.

Висновки. Розроблена інформаційна технологія інтелектуального автоматизованого керування FDM/FFF-3D-друком обґрунтовує використання Raspberry Pi 5 як хост-рівня для виконання інференсу, аналізу телеметрії, вебкерування та журналювання. Експериментальні результати підтвердили наявність достатнього ресурсного резерву за раціонального вибору параметрів вхідних даних. Реалізована інтеграція модуля штучного інтелекту із замкненим наглядовим контуром створює підґрунтя для своєчасного виявлення аномалій, адаптивного втручання у процес друку та підвищення надійності функціонування 3D-принтера. Запропонована технологія може бути використана як практична основа для побудови інтелектуальних систем контролю адитивного виробництва.

Список використаних джерел

1. Олейніков І. А., Срібна Т. В. Розробка методики для пошуку дефектів 3D-друку на основі моделі ResNet. Телекомунікаційні та інформаційні технології. 2025. № 1. С. 111–119.
2. Бондарчук А. П., Олейніков І. А., Бажан Т. О. Застосування методів машинного навчання до управління 3D-принтером. Телекомунікаційні та інформаційні технології. 2024. № 1. С. 4–15.
3. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 770–778.
4. 4. Lin T.-Y., Dollár P., Girshick R., He K., Hariharan B., Belongie S. Feature pyramid networks for object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. P. 2117–2125.
5. 5. Jacob B., Kligys S., Chen B., Zhu M., Tang M., Howard A., Adam H., Kalenichenko D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. P. 2704–2713.

Куцовол Іванна Іванівна
студентка 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
+380-(50)-136-90-25
st6815720@stud.duikt.edu.ua
Науковий керівник: Бажан Тетяна Олександрівна
старший викладач кафедри Технологій цифрового розвитку
Державний університет інформаційно-комунікаційних технологій, м. Київ

МОДЕЛІ ТА МЕТОДИ УПРАВЛІННЯ КАТАЛОГОМ ТОВАРІВ У ЦИФРОВИХ ПЛАТФОРМАХ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

Постановка задачі. У сучасних умовах розвитку електронної комерції ефективне управління каталогом товарів є критично важливим для функціонування вебзастосунків. Каталог виступає центральним елементом системи, який забезпечує зберігання інформації про товари, їх категоризацію та взаємозв'язки між сутностями. Зі збільшенням кількості товарів та атрибутів виникає необхідність у побудові гнучких моделей даних і застосуванні ефективних методів обробки інформації. Неправильна організація каталогу може призвести до зниження швидкодії системи, ускладнення пошуку та погіршення користувацького досвіду. Таким чином, актуальним є дослідження підходів до побудови структури каталогу та методів управління ним у цифрових платформах.

Мета дослідження. Метою роботи є дослідження моделей представлення каталогу товарів та методів його управління з урахуванням практичної реалізації у вебзастосунку інтернет-магазину.

Результати дослідження. У процесі дослідження встановлено, що найбільш доцільною для реалізації каталогу товарів у вебзастосунках є реляційна модель даних, яка забезпечує структурованість, цілісність і масштабованість системи. У рамках розробленого вебзастосунку каталог реалізовано за допомогою трьох основних сутностей: категорії товарів, товари та зображення товарів.

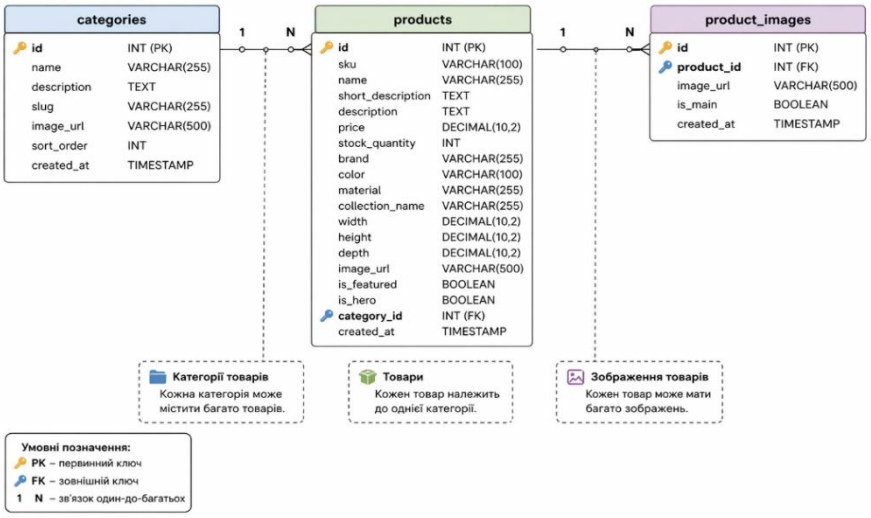


Рис. 1 – Модель каталогу товарів (categories – products – product_images)

Категорії формують ієрархічну структуру каталогу та забезпечують логічне групування товарів. Кожен товар належить до певної категорії, що дозволяє реалізувати зручну навігацію для користувача. Сутність товару містить основні характеристики, зокрема назву, опис, ціну та ідентифікатор категорії. Для

забезпечення наочного представлення товарів використовується окрема сутність зображень, яка пов’язана з товарами за принципом «один до багатьох».

Така організація дозволяє забезпечити ефективне управління даними, зокрема додавання нових товарів, редагування інформації та масштабування каталогу.

До основних методів управління каталогом належать:

- класифікація товарів за категоріями;
- фільтрація за параметрами (ціна, назва, категорія);
- сортування результатів;
- пошук товарів за ключовими словами.

Фільтрація та пошук реалізуються шляхом формування запитів до бази даних, що дозволяє швидко отримувати релевантні результати. Для підвищення продуктивності доцільно використовувати індексацію полів, які найчастіше беруть участь у пошуку.

Важливим аспектом є також організація взаємодії між клієнтською та серверною частинами. У досліджуваному вебзастосунку використовується підхід REST API, що забезпечує передачу даних між клієнтом (Next.js) та сервером (Spring Boot). Це дозволяє реалізувати гнучке управління каталогом та спростити розширення функціональності.

Висновки та перспективи. У результаті дослідження визначено, що ефективне управління каталогом товарів досягається шляхом використання реляційної моделі даних у поєднанні з оптимізованими методами пошуку та фільтрації. Запропонована структура каталогу забезпечує гнучкість, масштабованість та зручність використання.

Перспективами подальших досліджень є впровадження інтелектуальних методів обробки даних, зокрема автоматичної класифікації товарів, персоналізації видачі результатів та використання рекомендаційних систем.

Список використаних джерел

1. Elmasri R., Navathe S. Fundamentals of Database Systems. Pearson, 2016.
2. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O’Reilly Media, 2017.
3. Coronel C., Morris S. Database Systems Design, Implementation, and Management, Loose-Leaf Version, 13th + MindTap, 2 Terms Printed Access Card. Course Technology, 2019.

Ковшар Дар'я Сергіївна
студентка 4 курсу, групи КНД-42
Державний університет інформаційно-комунікаційних технологій
(066)053-55-51
st7901830@stud.duikt.edu.ua

СИНХРОНІЗАЦІЯ ІГРОВОГО СТАНУ В РЕАЛЬНОМУ ЧАСІ ЗАСОБАМИ UNITY NETCODE

Постановка задачі. Розвиток індустрії відеоігор та зростання популярності багатокористувацьких онлайн-ігор висувають підвищені вимоги до якості та стабільності мережевої взаємодії між гравцями. Ключовою проблемою при розробці таких ігор є забезпечення узгодженості ігрового стану на всіх клієнтських пристроях у режимі реального часу. Затримки передачі даних, втрата пакетів і нестабільність мережевого з'єднання призводять до desynchronization - розбіжності ігрових станів різних гравців, що суттєво погіршує ігровий досвід. У зв'язку з цим актуальним є дослідження сучасних підходів до синхронізації ігрового стану, зокрема із застосуванням інструментарію Unity Netcode for GameObjects [1].

Мета дослідження. Дослідження та практична реалізація механізмів синхронізації ігрового стану в реальному часі засобами Unity Netcode for GameObjects для забезпечення узгодженості даних між клієнтами у багатокористувацькій грі.

Результати дослідження. У цьому дослідженні ми розглянули головні методи побудови мережевої структури для ігор із залученням багатьох гравців: архітектуру "клієнт-сервер" та модель "рівний-до-рівного" (peer-to-peer). З’ясувалося, що конфігурація клієнт-сервер демонструє вищий рівень захищеності та спроможності керувати станом гри, оскільки сервер виконує роль універсального вірогідного джерела усієї інформації [2].

Було детально проаналізовано основні способи забезпечення узгодженості даних, які впроваджені у бібліотеці Unity Netcode for GameObjects. Об’єкт NetworkVariable відповідає за автоматичне поширення змін у змінних між сервером та клієнтами, підтримуючи при цьому механізм виклику функцій-обробників (callbacks) у разі оновлення значень. Функції ClientRpc та ServerRpc надають можливість викликати методи віддалено — відповідно, на всіх клієнтах або ж на сервері — що слугує для передачі подій без потреби у безперервному дублюванні поточного стану.

Окрема увага була приділена питанню згладжування запізнення (компенсації лагу). Вивчено такі методики, як прогнозування на стороні клієнта (client-side prediction) і звірка на стороні сервера (server reconciliation). Вони дозволяють клієнтському додатку самостійно обчислити наслідки дій гравця, чекаючи підтвердження від сервера, а у випадку розбіжностей — коригувати стан таким чином, щоб для користувача це виглядало без видимих візуальних збоїв. Застосування цих підходів за допомогою функціоналу Unity Netcode дає змогу істотно зменшити суб'єктивне сприйняття затримки, навіть коли час очікування (пінг) становить від 80 до 150 мілісекунд [3].

Практичне втілення було здійснене шляхом створення багатокористувацької гри на ігровому рушії Unity, використовуючи Unity Netcode

for GameObjects. Для тестування поведінки системи синхронізації в умовах нестабільного з'єднання застосовувався вбудований інструмент Network Simulator, що входить до складу Unity Netcode for GameObjects. Він дозволяє імітувати різні мережеві умови безпосередньо в редакторі Unity, зокрема затримку передачі даних та втрату пакетів, без необхідності використання сторонніх програм[4]. Проведене тестування підтвердило стабільність синхронізації ігрового стану під час одночасного підключення до чотирьох учасників, демонструючи середнє запізнення до 50 мс у межах локальної мережі.

Висновки та перспективи. Застосування Unity Netcode for GameObjects дає змогу вправно долати виклики, пов'язані з узгодженням стану гри у режимі реального часу, завдяки вбудованим функціям реплікації інформації та можливості виконання віддалених процедурних викликів. Комбінація передбачення на стороні клієнта з наступною серверною корекцією гарантує задовільну чутливість користувацького інтерфейсу, навіть коли якість мережевого з'єднання є коливною. Подальші роботи варто зосередити на вдосконаленні методів синхронізації задля масштабування до більшої кількості одночасних учасників та імплементації рішень на базі хмарних сервісів від Unity.

Список використаних джерел:

1. Unity.netcode.gameobjects. *Unity - Manual: Multiplayer*. URL: https://docs-multiplayer.unity3d.com/netcode/current/about.
2. Glazer & Madhav. Multiplayer game programming: architecting networked games. 2015.
3. Accuracy in dead-reckoning based distributed multi-player games / S. Aggarwal et al. ACM SIGCOMM 2004 workshops, Portland, Oregon, USA, 30 August 2004. New York, New York, USA, 2004. URL: https://doi.org/10.1145/1016540.1016559
4. Network simulator | multiplayer tools | 2.2.8. Unity - Manual: Unity 6.4 User Manual. URL: https://docs.unity3d.com/Packages/com.unity.multiplayer.tools@2.2/manual/network-simulator.html.

Петраченко Ігор Андрійович
Студент групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій, м. Київ
+380 98 472 69 73
st8031690@stud.duikt.edu.ua

ОСОБЛИВОСТІ ПРОЄКТУВАННЯ WMS-СИСТЕМИ ДЛЯ УПРАВЛІННЯ СКЛАДОМ МАГАЗИНУ ФОРМАТУ DIY

Постановка задачі. Магазини формату DIY працюють із великою номенклатурою товарів, що включає будівельні матеріали, інструменти, комплектуючі, садову техніку та дрібні витратні позиції. Для таких складів критично важливими є точний облік залишків, адресне зберігання, швидке приймання товарів, контроль внутрішніх переміщень і своєчасне поповнення торгової зали. Використання паперових журналів або простих табличних рішень не дозволяє забезпечити необхідну швидкість оновлення даних і підвищує ризик пересорту, дефіциту та помилок під час обробки операцій. Тому актуальною є розробка WMS-системи, здатної автоматизувати складські процеси та підтримувати роботу з даними в режимі реального часу.

Мета дослідження. Метою дослідження є розробка підходів до проєктування інформаційної системи управління складом магазину формату DIY, яка забезпечує автоматизований облік товарних запасів, контроль логістичних операцій та підвищення точності складських процесів. Для досягнення мети необхідно визначити ключові функціональні модулі системи, спроєктувати структуру даних, описати бізнес-логіку обробки складських операцій та обґрунтувати доцільність використання сучасних програмних засобів для створення функціонального прототипу.

Результати дослідження. Аналіз складських процесів у DIY-магазинах показав, що найбільший вплив на ефективність роботи мають швидкість фіксації операцій, точність адресного зберігання та можливість оперативного пошуку товару. Запропонована WMS-система орієнтована на підтримку повного циклу руху товарів: приймання, розміщення, переміщення, інвентаризацію, списання та формування звітів. Модуль приймання забезпечує реєстрацію поставок і перевірку кількості товарів, модуль зберігання - визначення комірок і контроль логістичних зон, а модуль залишків - актуалізацію даних після кожної операції. Окремо реалізуються механізми пошуку товару, інвентаризації та аналітичної звітності для адміністрації магазину.

Структура системи може бути реалізована за клієнт-серверною архітектурою з використанням централізованої бази даних, що забезпечує єдине джерело актуальної інформації для працівників складу, торгового залу та адміністрації. У межах моделі даних доцільно передбачити сутності товарів, комірок складу, залишків, транзакцій, постачальників і користувачів. Бізнес-логіка системи має включати перевірку доступності комірок, журналювання дій користувачів, автоматичне оновлення залишків і підтримку інтеграції зі сканерами штрихкодів, що дозволяє зменшити кількість помилок і прискорити обробку складських операцій.

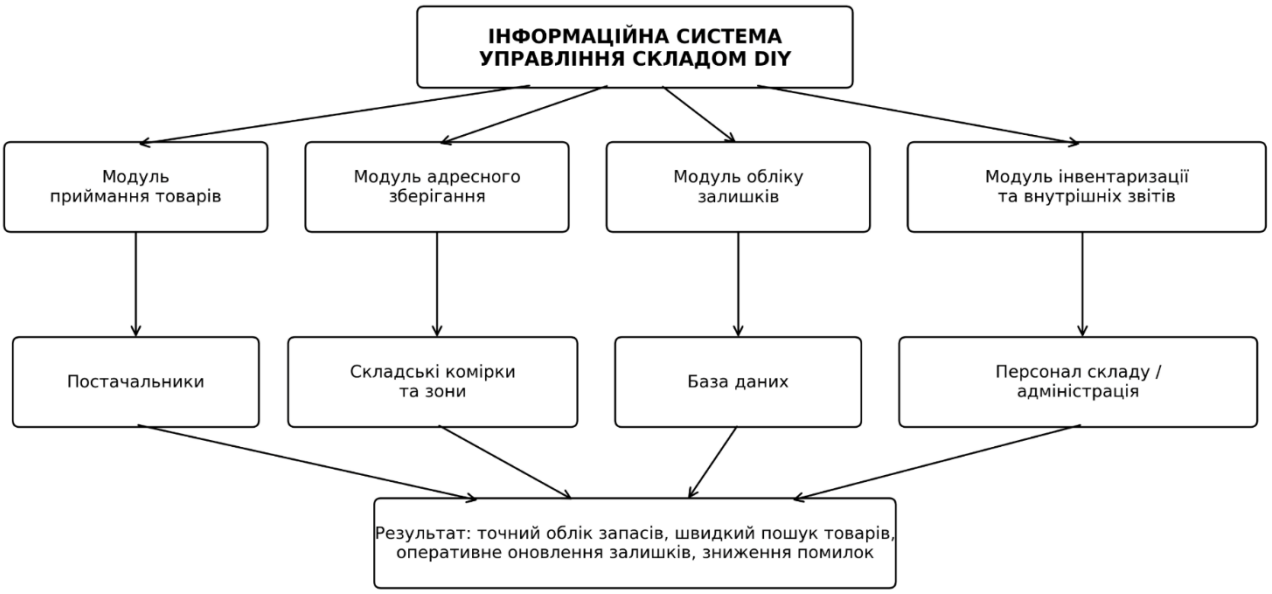


Рис. 1. Узагальнена схема функціональних модулів WMS-системи для складу магазину формату DIY

Важливою перевагою запропонованого підходу є можливість реалізації програмного рішення на основі мови C# та платформи .NET, які забезпечують високу продуктивність, масштабованість і зручність побудови клієнт-серверних застосунків. Для зберігання інформації можуть використовуватися реляційні системи керування базами даних, зокрема SQL Server або PostgreSQL. Використання WebAPI дозволяє об’єднати робочі місця операторів, мобільні термінали та сервер бази даних у єдину інформаційну інфраструктуру, а застосування MAUI або суміжних інструментів відкриває можливість створення мобільних рішень для роботи зі штрихкодами та RFID-мітками.

Висновки та перспективи. У результаті дослідження визначено основні вимоги до інформаційної системи управління складом магазину формату DIY та обґрунтовано доцільність впровадження WMS-підходу для автоматизації обліку товарних запасів. Запропонована концепція системи охоплює модулі приймання, адресного розміщення, контролю залишків, інвентаризації, пошуку товарів і формування внутрішньої аналітики. Подальшим напрямом розвитку є деталізація UML-моделей, розробка діаграми взаємодії модулів, створення прототипу користувацького інтерфейсу та реалізація демонстраційної версії системи з підключенням засобів автоматичної ідентифікації товарів.

Список використаних джерел

1. Граф М.С. Розробка структури інформаційної системи внутрішньої мережі Управління інформаційно-аналітичної підтримки ГУНП в Житомирській області / М.С. Граф, В.А. Райковський // Інформаційно-комп’ютерні технології – 2024 : тези XIV Міжнародної науково-технічної конференції, 28–29 березня. – Житомир : Житомирська політехніка, 2024. – С. 119–121. URL: https://conf.ztu.edu.ua/wp-content/uploads/2024/05/119.pdf (дата звернення: 29.04.2026).
2. Карпенко М. Ю., Манакова Н. О., Гавриленко І. О. Технології створення програмних продуктів та інформаційних систем : навч. посіб. Харків : ХНУМГ ім. О. М. Бекетова, 2017. 93 с. URL:

https://eprints.kname.edu.ua/46989/1/2017%20%D0%BF%D0%B5%D1%87.%2024%D0%9D%20%D0%9F%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B514.04.17__.pdf (дата звернення: 29.04.2026).

3. Лондар С.Л. Міжнародний досвід розвитку сучасних освітніх інформаційних систем / С.Л. Лондар // Освітня аналітика України. – 2019. – № 1 (5). – С. 5–19. URL: https://science.iea.gov.ua/wp-content/uploads/2019/05/1_Londar_15_2019_5_19-1.pdf (дата звернення: 29.04.2026).
4. Райковський В.А. Використання сучасних архітектурних підходів для розробки інформаційної системи для управління інформаційно-аналітичної підтримки / В.А. Райковський, Р.А. Вольський // Інтегровані інтелектуальні робототехнічні комплекси (ІІРТК-2024) : тези сімнадцятої міжнародної науково-практичної конференції, 12–22 травня. – Київ : НАУ, 2024. – С. 419–421. URL: https://conf.ztu.edu.ua/wp-content/uploads/2025/01/povnyj-tekst-1.pdf (дата звернення: 29.04.2026).
5. Яворський М.В. Проектування інтерфейсів користувача в інформаційних системах : підручник / М.В. Яворський. – К. : КПІ ім. Ігоря Сікорського, 2023. – 280 с. URL: https://ela.kpi.ua/ (дата звернення: 29.04.2026).

Павленко Артем Ігорович,
студент 4 курсу, групи ПД-44,
спеціальність 121 Інженерія програмного забезпечення,
Державного університету інформаційно-комунікаційних технологій
st7994558@stud.duikt.edu.ua
Науковий керівник: Садовенко Володимир Сергійович,
кандидат фізико-математичних наук, доцент,
професор кафедри Інженерії програмного забезпечення
Державного університету інформаційно-комунікаційних технологій.

РОЗРОБКА CLI-УТИЛІТИ ДЛЯ АВТОМАТИЗАЦІЇ РЕЗЕРВНОГО КОПІЮВАННЯ ТА ВІДНОВЛЕННЯ БАЗ ДАНИХ POSTGRESQL І MYSQL

Питання надійності зберігання даних у сучасних інформаційних системах набуває дедалі більшого практичного значення. Втрата або пошкодження інформації в реляційних базах даних здатна призвести до суттєвих операційних збитків і порушення безперервності бізнес-процесів [1]. У той час як логічні резервні копії дозволяють відновлювати інформацію, що зберігається в базі даних, лише протягом певного періоду, фізичні резервні копії дають змогу відновлювати дані в будь-який момент часу (Point-in-Time Recovery, PITR), тим самим зменшуючи час простою в процесі аварійного відновленнях [2]. Проте, налаштування фізичних резервних копій є досить складним, оскільки передбачає добре знання внутрішньої архітектури конкретної СУБД, а також виконання кількох послідовних дій, що підвищує ймовірність помилок.

Постановка задачі.

Задачею мого дослідження є розробка утиліти CLI (Command Line Interface) мовою Python для автоматизації фізичного резервного копіювання баз даних PostgreSQL та MySQL. Програма має підтримувати як повне, так і диференціальне резервне копіювання, вести репозиторій резервних копій, перевіряти їх цілісність та відновлювати бази даних.

Мета дослідження.

Метою є спрощення процесів резервного копіювання і відновлення баз даних PostgreSQL і MySQL шляхом розробки програмного засобу, який уніфіковує взаємодію з різними СУБД через єдиний командний інтерфейс.

Результати дослідження.

Розроблена мною утиліта реалізована на Python 3.10 і має модульну архітектуру, в основі якої лежить патерн «Стратегія» (Strategy). Це рішення дозволяє відокремити логіку бекапу конкретної СУБД від ядра CLI-застосунку та дає можливість розширення системи на нові бази даних без зміни основного коду.

Для PostgreSQL реалізовано два режими резервного копіювання. Повне копіювання виконується через утиліту *pg_basebackup* і включає валідацію конфігурації (перевірку параметрів *wal_level, archive_mode* та наявності привілею

REPLICATION), збереження архівів *base.tar.gz* і *pg_wal.tar.gz*, а також формування службових метаданих. Диференційне копіювання ґрунтується на накопичених WAL-сегментах: утиліта зчитує ланцюг копій із каталогу, перевіряє цілісність і послідовність WAL-файлів та формує диференційну копію на їх основі. Процедуру відновлення автоматизовано до рівня генерації конфігурації *postgresql.auto.conf*, запуску *pg_ctl* і верифікації стану бази через SELECT 1 [3].

Резервне копіювання MySQL реалізоване за допомогою Percona XtraBackup - інструменту, що забезпечує «гарячий» бекап без блокування таблиць [4]. Підтримуються повні та інкрементальні копії. Автентифікація здійснюється через зашифровані профілі *mysql_config_editor*, що повністю виключає передачу паролів у відкритому вигляді. Для стиснення використовується алгоритм Zstandard (zstd) у багатопотоковому режимі, це прискорює архівацію до чотирьох разів порівняно зі стандартним однопотоковим виконанням.

Усі резервні копії реєструються у SQLite-каталозі. Кожній копії присвоюється унікальний ідентифікатор, а зв'язки між повними та похідними копіями зберігаються через поля *parent_backup_id* і *base_backup_id* - це дозволяє коректно відновлювати ланцюги залежностей при PITR. Разом з кожною копією формуються metadata.json із технічними характеристиками (тип бекапу, маркери LSN або binlog, контрольні суми, шляхи до артефактів) та validation.json із результатами перевірки артефактів.

Безпека забезпечена на кількох рівнях: використання *~/.pgpass* для PostgreSQL і login-path для MySQL унеможливлює появу відкритих паролів у процесах і системних логах, а права доступу до файлів резервних копій обмежуються на рівні файлової системи (chmod 600). Перед запуском будь-якої операції виконуються попередні перевірки конфігурації СУБД, що виявляють помилки конфігурування до початку копіювання.

Висновки та перспективи.

Розроблена CLI-утиліта автоматизує критичні операції фізичного резервного копіювання PostgreSQL і MySQL, суттєво знижуючи ймовірність помилок при ручному виконанні. Модульна архітектура на основі патерну Strategy забезпечує підтримуваність коду і спрощує подальше розширення, зокрема інтеграцію з хмарними сховищами та реалізацію повноцінного PITR-інтерфейсу.

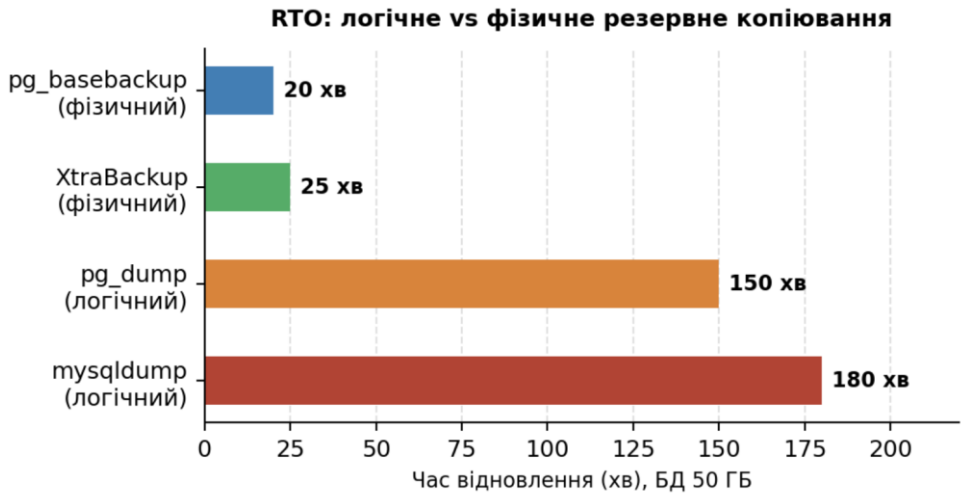


Рис. 1. Порівняння часу відновлення (RTO) для логічного та фізичного резервного копіювання бази даних обсягом 50 ГБ

Перспективним напрямом розвитку є також автоматизація процедури відновлення для MySQL та реалізація механізму автоматичного очищення застарілих резервних копій.

Список використаних джерел:

1. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. – 7th ed. – McGraw-Hill Education, 2020. – 1376 ст.

2. Stonebraker M., Hellerstein J. M. Readings in Database Systems. – 5th ed. – MIT Press, 2015. – 840 p.

3. PostgreSQL Global Development Group. PostgreSQL 16 Documentation: Write Ahead Logging. [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/current/wal-intro.html

4. Percona LLC. Percona XtraBackup 8.0 Documentation. [Електронний ресурс]. – Режим доступу: https://docs.percona.com/percona-xtrabackup/8.0/

5. Hoverstadt P., Loh L. Patterns of Strategy. Abingdon, Oxon ; New York, NY : Routledge, 2017. : Routledge, 2017. URL: https://doi.org/10.4324/9781315277776

6. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. – Addison-Wesley, 1994. – 395 p.

Петрушенко Відана Вікторівна
студентка 4-го курсу, групи КНД-42
Державний університет інформаційно-комунікаційних технологій
(095) 083-60-66
vidana.petrushenko@gmail.com

ВИКОРИСТАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНИХ ПАТЕРНІВ ПРОЄКТУВАННЯ ПРИ РОЗРОБЦІ ІГРОВИХ МЕХАНІК МОВОЮ JAVA

Постановка задачі. Сучасна розробка комп'ютерних ігор (GameDev) характеризується високим рівнем складності програмного коду. Створення ігрових механік вимагає гнучкості, масштабованості та можливості легкого розширення функціоналу без повної перебудови архітектури системи. Мова програмування Java, завдяки своїй об'єктно-орієнтованій природі та системі управління пам'яттю, є потужним інструментом для створення ігрових додатків. Проте, без застосування перевірених архітектурних рішень, код швидко стає важким для підтримки (так званий «spaghetti code»). Актуальною проблемою є пошук оптимальних способів організації взаємодії ігрових об'єктів та систем управління станами гри, що вимагає впровадження об'єктно-орієнтованих патернів проєктування.

Мета дослідження. Метою роботи є аналіз та практичне впровадження фундаментальних патернів проєктування (GoF — Gang of Four) у процес розробки ігрових механік мовою Java для підвищення модульності та ефективності програмного забезпечення.

Результати. Об'єктом практичної реалізації стала розробка 2D-пригоди у стилі піксель арту під назвою «Gem mystery». Гра розроблена на мові Java з використанням фреймворку LibGDX та зосереджена на дослідженні відкритого світу та виконанні квестів. Основною особливістю проекту є система загадок, заснованих на взаємодії з навколишнім середовищем (активація механізмів, маніпуляція об'єктами). Візуальна частина реалізована за допомогою тайлових карт (TMX), а ігрові механіки включають систему діалогів з NPC та журнал завдань.

У ході дослідження було проведено порівняння архітектури, побудованої на об'єктно-орієнтованих патернах, із традиційним підходом на основі великої кількості умовних конструкцій (if-else / switch-case) для управління станами квестів. Результати показали, що використання патерну «Stte» та «Command» для обробки дій гравця та станів загадок дозволяє уникнути логічних помилок при переході між етапами квесту. При монолітному підході складність коду зростала експоненційно з кожною новою загадкою, тоді як застосування патернів дозволило зберегти лінійну складність підтримки коду, забезпечивши стабільну роботу системи збереження прогресу (Save/Load) навіть при великій кількості активних тригерів у світі гри.

У ході розробки було досліджено та інтегровано кілька ключових груп патернів:

1. **Патерн «Singleton» (Одинак):** Використаний для централізованого управління ресурсами (Asset Manager), що гарантує єдиноразове завантаження спрайтів та звукових ефектів у пам'ять.

2. **Патерн «State» (Стан):** Використаний для управління станами квестів (Locked, In Progress, Solved) та механізмами загадок. Це дозволило ізолювати логіку кожної стадії квесту в окремих класах, уникаючи помилок при переході між етапами гри.
3. **Патерн «Observer» (Спостерігач):** Став основою для системи подій. Коли гравець виконує дію (наприклад, знаходить квестовий предмет), система сповіщає журнал завдань та UI-інтерфейс без створення жорстких зв'язків між ними.
4. **Патерн «Strategy» (Стратегія):** Застосований для реалізації різних типів загадок (логічні, на швидкість, на збір предметів). Це дозволило динамічно змінювати умови перемоги для різних ігрових зон, просто підміняючи об'єкт алгоритму перевірки.
5. **Патерн «Game Loop» та «Update Method»:** Хоча вони є специфічними для ігрової індустрії, їх поєднання з принципами ООП дозволило досягти стабільної частоти оновлення кадрів.

Практична реалізація ігрових механік з використанням цих патернів показала, що час на додавання нового ігрового об'єкта або зміну існуючої логіки скоротився на 30-40% порівняно з процедурним підходом. Використання інтерфейсів Java дозволило забезпечити високий рівень абстракції, що полегшило тестування окремих модулів гри.

Висновки. Застосування об'єктно-орієнтованих патернів проєктування є необхідною умовою створення якісного ігрового продукту на Java.

1. Патерни «Singleton» та «State» забезпечують стабільну архітектуру та коректне управління ресурсами.
2. «Observer» та «Strategy» дозволяють створювати гнучкі та інтерактивні ігрові механіки, які легко піддаються модифікації.
3. Використання патернів сприяє повторному використанню коду та полегшує колективну розробку, оскільки архітектура стає зрозумілою для інших програмістів. Перспективи подальших досліджень полягають у вивченні патернів оптимізації, таких як «Object Pool», для управління великою кількістю ігрових об'єктів (наприклад, куль або частинок) без перевантаження збирача сміття (Garbage Collector) Java.

Список джерел:

1. Design Patterns Elements of Reusable Object-Oriented Software / R. Helm та ін. Pearson Education, 2000.
2. Eckel B. Thinking in Java. 4-те вид. Upper saddle River, NJ : Prentice Hall, 2006.
3. Java: The Complete Reference, Eleventh Edition. McGraw-Hill Education, 2018. 1248 с.
4. Nystrom R. Game Programming Patterns. Apress, 2011. 300 с.

Аронов Андрій Олексійович,
к.т.н., доцент кафедри технологій цифрового розвитку
Державний університет інформаційно–комунікаційних технологій
a.aronov@duikt.edu.ua

ОПТИМІЗАЦІЯ SQL–ЗАПИТІВ ТА СТВОРЕННЯ ІНДЕКСІВ У РЕЛЯЦІЙНИХ БАЗАХ ДАНИХ

Постановка задачі. У сучасних інформаційних системах продуктивність баз даних відіграє критично важливу роль, особливо при роботі з великими обсягами даних. Однією з основних причин зниження швидкодії є неефективні SQL–запити та відсутність або неправильне використання індексів. Запити з великою кількістю умов, використанням оператора «LIKE» із шаблоном «%...%», а також сортуванням, групуванням значних обсягів даних можуть призводити до повного сканування таблиць (full table scan), що суттєво збільшує час виконання SQL–запиту [1].

Особливої уваги потребують системи електронної комерції, CRM та інші високонавантажені сервіси, де пошук даних здійснюється одночасно за багатьма полями та багатьма клієнтами одночасно. Відсутність оптимізації в таких випадках призводить до перевантаження сервера та погіршення користувацького досвіду [2].

Мета дослідження. Метою дослідження є аналіз підходів до оптимізації SQL–запитів та визначення ефективних методів створення індексів для підвищення продуктивності реляційних баз даних.

Результати дослідження. У ході дослідження встановлено, що продуктивність SQL–запитів значною мірою залежить від можливості використання індексів.

Використання конструкцій типу «LIKE '%значення%'» унеможливлює застосування B–Tree індексів, оскільки пошук виконується без фіксованого початку рядка. Це призводить до повного сканування таблиці [3]. Альтернативою є: використання FULLTEXT індексів або зміна логіки пошуку на LIKE «значення%», якщо це допустимо.

Оптимізація складних умов є важливою складовою прискорення виконання SQL–запитів до БД. Запити з великою кількістю умов «OR» по різних колонках суттєво знижують ефективність, наприклад:

col1 LIKE ... OR col2 LIKE ... OR col3 LIKE ..

У таких випадках доцільно використовувати повнотекстовий пошук «MATCH ... AGAINST», об’єднувати дані в окреме поле для пошуку та створювати допоміжні таблиці індексації.

Встановлено, що правильне використання індексів дозволяє зменшити час виконання запитів у десятки разів. Найефективніші підходи це одиночні індекси, що використовуються для фільтрації «WHERE», складені індекси для запитів із «ORDER BY» та кількома умовами і тільки потім «FULLTEXT» індекси для текстового пошуку.

Наприклад, створення індексу:

CREATE INDEX idx_sort ON table_name(col1, col2, col3);

дозволяє оптимізувати сортування без додаткового навантаження.

Також слід зазначити, що використання GROUP BY без необхідності, особливо по унікальних полях, створює додаткові витрати ресурсів через використання тимчасових таблиць. Аналогічно знижують ефективність виконання SQL–запитів і неправильні типи даних, наприклад, порівняння чисел як рядків.

На рис.1 представлено порівняння навантаження на веб–сервер у різні дні до та після оптимізації SQL–запитів. Видно, що після впровадження оптимізації, зокрема і використання індексів та покращення структури запитів, спостерігається суттєве зниження навантаження та стабілізація показників, що свідчить про підвищення ефективності роботи системи та швидкості обробки запитів.

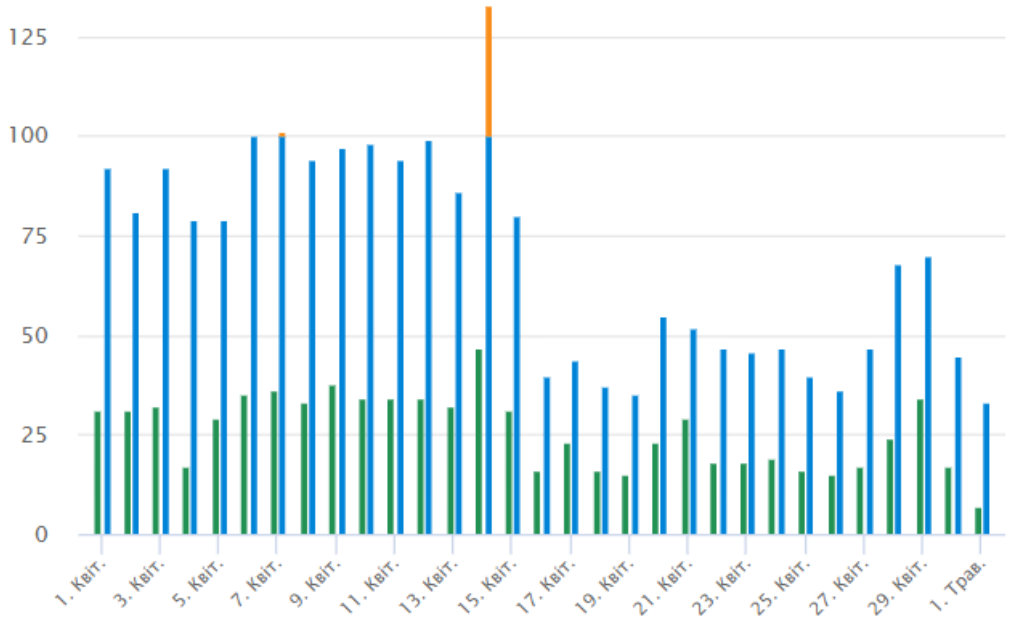


Рис. 1. Зміна навантаження на веб–сервер після оптимізації SQL–запитів

Висновки та перспективи. У результаті дослідження встановлено, що оптимізація SQL–запитів є критично важливою для забезпечення високої продуктивності баз даних. Використання індексів є ключовим фактором швидкодії, конструкції «LIKE '%...%'» значно знижують ефективність і повинні замінюватися альтернативами, а складені індекси дозволяють оптимізувати сортування та фільтрацію. Застосування наведених підходів дозволяє суттєво зменшити час виконання запитів та підвищити масштабованість системи.

Список використаних джерел

Мельник А. О., Бойко О. В. Системи баз даних: проєктування та оптимізація : навч. посіб. – Київ : КПІ ім. Ігоря Сікорського, 2022. – 256 с.

Гайдамака Ю. В. Організація баз даних та знань : навч. посіб. – Київ : НУБіП України, 2020. – 220 с.

Пасічник В. В., Щербина Ю. В. Організація баз даних та знань : навч. посіб. – Львів : Магнолія–2006, 2019. – 384 с.

MySQL Documentation. Оптимізація запитів та індекси [Електронний ресурс]. – 2024. – Режим доступу: https://dev.mysql.com/doc/ – Дата звернення: 20.04.2026.

Кузнецов Павло Олександрович
студент 4 курсу, групи ПД-41
Державний університет інформаційно-комунікаційних технологій
(066) 029 57 12
donikep76223003@gmail.com

РОЗРОБКА ТА СУПРОВІД WEB-ЗАСТОСУНКУ “ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕКОМЕНДАЦІЇ КІНОФІЛЬМІВ ЗА ВПОДОБАННЯМИ КОРИСТУВАЧІВ”

Постановка задачі. У сучасному цифровому середовищі web-застосунки є невід’ємною частиною повсякденного життя користувачів. Вони забезпечують доступ до різноманітних сервісів, зокрема медіаконтенту, але водночас стають об’єктом потенційних кіберзагроз. Особливо актуальним є забезпечення безпеки застосунків, які працюють із зовнішніми API та зберігають користувацькі дані. Основними проблемами є захист від атак типу XSS, несанкціонований доступ до даних, а також безпечна взаємодія із зовнішніми сервісами.

Тому актуальною задачею є розробка веб-застосунку, який забезпечує ефективний пошук, сортування та рекомендацію кінофільмів із використанням відкритих джерел даних. Особливу увагу слід приділити зручності користування, швидкості обробки запитів та можливості персоналізації.

Мета дослідження. Метою дослідження є розробка веб-застосунку для рекомендації кінофільмів, який спрощує процес пошуку та вибору контенту за рахунок використання сучасних веб-технологій та інтеграції з зовнішнім API.

Результати дослідження. У результаті було розроблено веб-застосунок для рекомендації кінофільмів із використанням технологій HTML, CSS та JavaScript. Для отримання інформації про фільми використано API (TMDB), що забезпечує доступ до актуальних даних про фільми, включаючи назву, рейтинг, жанри, опис та постери.

У застосунку реалізовано функціонал пошуку фільмів за назвою з використанням асинхронних запитів. Для покращення взаємодії з користувачем впроваджено механізм підказок, який дозволяє швидко знаходити потрібний фільм під час введення запиту.

Важливою складовою є система фільтрації, яка дозволяє обирати фільми за жанрами, роком випуску та мінімальним рейтингом. Це дає можливість користувачу формувати результати відповідно до власних вподобань. Також реалізовано сортування фільмів за популярністю, рейтингом та датою виходу.

Для зручності користувача впроваджено функціонал списку перегляду, який дозволяє зберігати обрані фільми у локальному сховищі браузера. Це забезпечує швидкий доступ до збережених фільмів без необхідності повторного пошуку.

Додатково реалізовано перегляд детальної інформації про фільм, включаючи опис, рейтинг та список схожих фільмів. Це дозволяє розширити можливості рекомендацій та покращити якість підбору контенту.

Архітектура застосунку базується на клієнт-серверній взаємодії, де клієнтська частина здійснює обробку запитів та відображення даних, а зовнішній API забезпечує отримання інформації. Такий підхід дозволяє створити легкий та масштабований веб-застосунок без необхідності розгортання власного серверу.

Висновки та перспективи. У ході дослідження було розроблено веб-застосунок для рекомендації кінофільмів, який забезпечує ефективний пошук, фільтрацію та збереження контенту. Реалізовані функції дозволяють значно спростити процес вибору фільмів та підвищити зручність користування.

Проведена робота підтверджує, що використання відкритих API та сучасних веб-технологій дозволяє створювати функціональні інформаційні системи без значних ресурсних витрат.

Перспективами подальшого розвитку є впровадження алгоритмів персоналізованих рекомендацій, використання методів машинного навчання для аналізу вподобань користувачів, а також інтеграція системи авторизації для збереження даних користувачів у хмарному середовищі.

Список використаних джерел

1. Бондаренко О., Ушкаленко І. (2017). Безпека web-додатків: актуальні проблеми та їх аналіз. Формування ринкової економіки в Україні, Вип. 38, 28-36.

2. Боскін О.О., Корніловська Н.В., Поліщук В.М., Сарафаннікова Н.В. (2023). Безпека веб-додатків та хакерські атаки. Вісник Херсонського національного технічного університету, №3(86), 83–92. https://doi.org/10.35546/kntu2078-4481.2023.3.11

3. OWASP Foundation. OWASP Top Ten Web Application Security Risks. - Режим доступу: https://owasp.org

4. The Movie Database (TMDB) API Documentation. URL: https://developer.themoviedb.org

Волошин Нікіта Назарійович
студент 4 курсу, групи ПД-43
Державного університету інформаційно-комунікаційних технологій
(099)112-46-00
voloshyn.nikita4@gmail.com

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМПЛЕКСНОГО МОНІТОРИНГУ ТА КОРЕКЦІЇ ІНДИВІДУАЛЬНИХ ПОВЕДІНКОВИХ ПАТЕРНІВ КОРИСТУВАЧА

Постановка задачі. У сучасному ритмі життя підтримувати корисні звички буває досить складно. Контроль екранного часу, дотримання графіку сну або питного режиму часто вимагають постійних зовнішніх нагадувань. Через складність самостійного відстеження щоденних процесів люди нерідко припиняють роботу над своїми звичками. Подолати цей бар'єр допомагає комплексне програмне забезпечення. Воно автоматично збирає інформацію про поведінкові патерни, вчасно нагадує про заплановані дії та формує зрозумілі графіки, що дозволяють об'єктивно оцінити власну дисципліну.

Мета дослідження. Визначити стек технологій і спроєктувати архітектуру для розробки клієнт-серверного застосунку, призначеного для моніторингу та трекінгу індивідуальної поведінки користувача.

Результати дослідження. В якості базової архітектури застосунку обрано клієнт-серверну модель. Це дозволяє зручно розділити логіку: серверна частина відповідає за роботу з базою даних і обробку запитів, а клієнтська – за відображення інтерфейсу та збір відміток від користувача. Такий підхід значно спрощує подальший розвиток системи управління та корекції петернів користувача.



Рис.1 - Узагальнена блок-схема роботи програмного забезпечення

Для реалізації бекенду використано мову Java [1] та фреймворк Spring Boot [2]. Цей стек добре підходить для створення надійної бізнес-логіки та обробки великої кількості подій. Використання Spring Boot дало можливість швидко розгорнути REST API. Наразі спроєктовано основні ендпоінти, серед яких: GET /api/v1/habits для завантаження списку звичок, POST /api/v1/habits для створення нових та PUT /api/v1/habits/{id} для оновлення їхнього статусу.

Дані системи зберігаються у реляційній базі PostgreSQL [3]. База даних містить нормалізовану схему, до якої входять таблиці users (дані профілів), habits (налаштування звичок) та habit_logs (історія виконання). Вибір PostgreSQL зумовлений її популярністю та ефективністю під час з'єднання статистики за тривалі періоди.

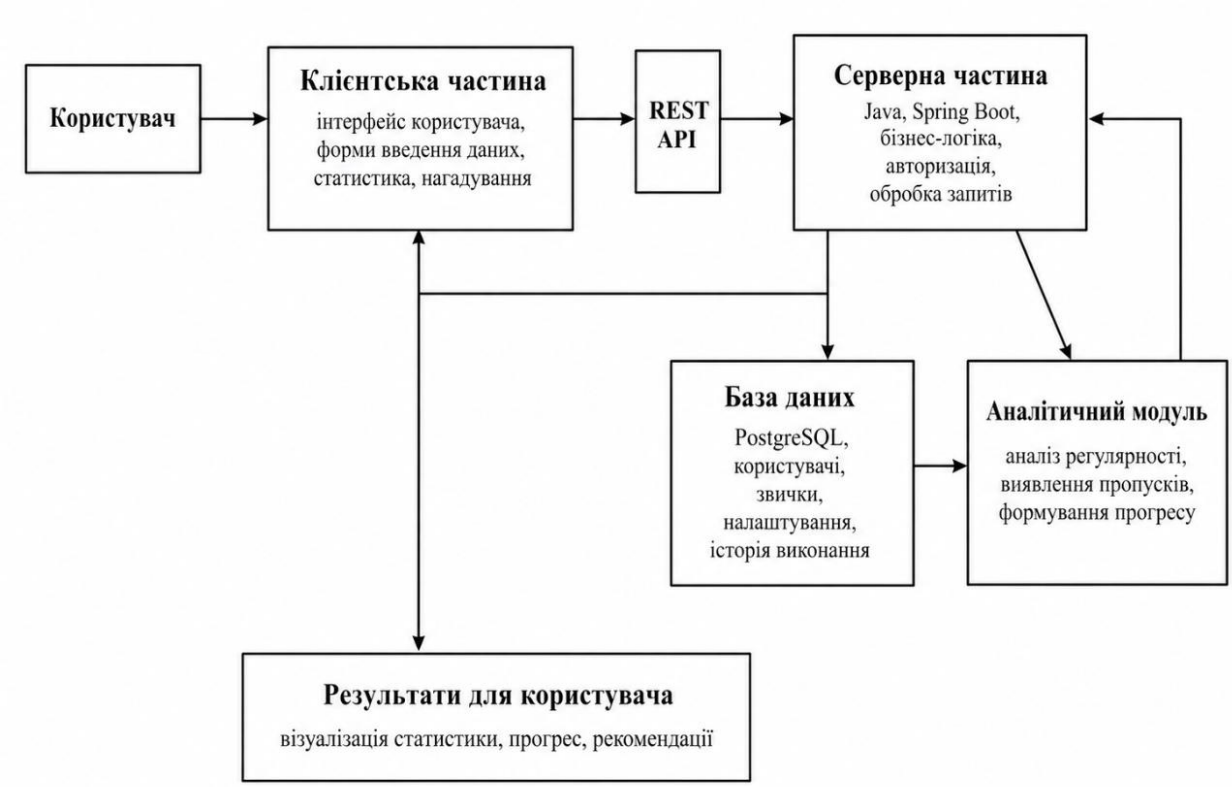


Рис.2 - Загальна архітектура програмного забезпечення для моніторингу та корекції індивідуальних поведінкових патернів користувача

Логіка нагадувань побудована на основі вбудованого планувальника cron-задач у Spring Boot, через що алгоритм регулярно перевіряє таблицю habits і в заданий час формує чергу на відправку повідомлень.

Клієнтська частина написана з використанням бібліотеки React [4], що зменшує кількість помилок на етапі підтримки коду. В інтерфейсі реалізовано інтерактивний календар для відміток і панель дашбордів для побудови графіків успішності (на базі бібліотеки Recharts). Поточний прототип повністю закриває базовий функціонал трекера.

Висновки. Проведений аналіз доводить розробку такого трекера доцільною. Використання Java, Spring Boot, React та PostgreSQL є вдалим та сучасним рішенням для поставленого завдання. Розроблений прототип дає змогу автоматично проводити контроль за розкладом і знизити когнітивне навантаження користувача.

Список використаних джерел:

1. Java Platform, Standard Edition Documentation [Електронний ресурс]. URL: https://docs.oracle.com/en/java/ (дата звернення: 01.05.2026).

2. Spring Boot Reference Documentation [Електронний ресурс]. URL: https://docs.spring.io/spring-boot/ (дата звернення: 01.05.2026).

3. PostgreSQL: Official Documentation [Електронний ресурс]. URL: https://www.postgresql.org/docs/ (дата звернення: 01.05.2026).

4. React Documentation [Електронний ресурс]. URL: https://react.dev/ (дата звернення: 01.05.2026).

Осадчий Максим Вікторович
студент 4 курсу, групи ПД-42
Державний університет інформаційно-комунікаційних технологій
(068) 556-57-08
st5353320@stud.duikt.edu.ua

WEB-САЙТ ДЛЯ ЦЕНТРАЛІЗОВАНОГО МОНІТОРИНГУ РИНКУ ВЖИВАНИХ АВТО

Постановка задачі. Ринок вживаних автомобілів є досить розгалуженим. Відомо, що досить велика кількість автомобілів є доступними для покупців на різних платформах. Такий спосіб приводить до того, що покупцям доводиться досить багато витрачати часу на перевірку різних сайтів, перевірку цін, пробігу, випуску, комплектації, стану та місця випуску вживаних авто. Часто такі оголошень дублюються на різних платформах, а також вони можуть бути відсутні та неповні.

Щоб вирішити цю задачу, доцільно створити власний web-сайт. Він зможе збирати дані про вживані автомобілі з різних платформ, а також надавати ці дані для перегляду та аналізу користувачам. Такий сайт може включати можливості виконання пошуку, фільтрації, збереження авто для подальшого аналізу, а також порівняння числа вживаних авто.

Мета дослідження. Метою дослідження є пошук та аналіз пропозицій продажу автомобілів з пробігом в онлайн-середовищі.

Результати дослідження. Основною ідеєю, яка була запропонована в ході вивчення, є ідея відстеження та збору оголошень від різних платформ. Створюючи веб-сайт, де мається можливість перегляду вживаних авто, що наявні на різних платформах. Також відкриємо можливості вибрати ті авто, які найбільше цікавлять, а також зможемо вивести таблицю, де можна буде здійснювати порівняння авто.

Серверна частина веб-сайту може бути реалізована завдяки використанню мови програмування Java. Така мова є придатною для створення веб-сайтів, які сприймають дані від різних платформ, виробляють обробку та надають результат для користувача від системи [1].

Інструментами, які можуть бути використані для здійснення бекенду веб-сайту, є Spring Framework та Spring Boot. Вони підходять для створення окремих модулів, які вироблять певні функції тільки від бекенду веб-сайту. Зокрема, можливо використання Spring Data JPA, для можливості зберігати дані авто в базі даних [2].

Для зберігання даних в базі даних можна використовувати платформу PostgreSQL. В базі даних можна зберегти будь-які дані про вживані автомобілі, платформи, користувачів, та їх активності в системі.

Для клієнтської частини веб-сайту доцільно використовувати веб-технології React. Це надасть можливість створити веб-сайт, який відображає вживані автомобілі, окремі сторінки для пошуку, фільтрації, збереження та порівняння авто. Оскільки в React використовуються компоненти, вони можуть виводиться окремо один від одного, а також вони можуть бути використані більшістю веб-сторінок для веб-сайту [3].

Рис. 1 відображає узагальнену схему, яка відображає основну логіку, яка передбачається від відкриття веб-сайту для перегляду авто.

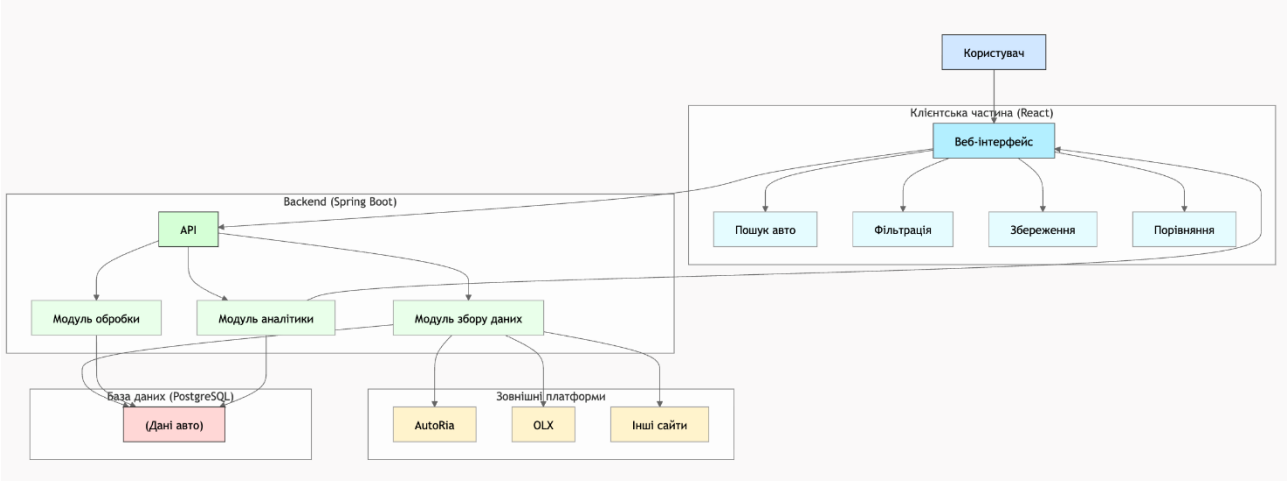


Рис. 1. Структурна схема взаємодії компонентів веб-сайту моніторингу ринку вживаних автомобілів.

Діаграма відображає загальну архітектуру веб-сайту для моніторингу ринку вживаних автомобілів. Користувач працює через інтерфейс на базі React, використовуючи функції пошуку, фільтрації та порівняння авто. Запити обробляються серверною частиною (Spring Boot), яка збирає дані з різних платформ, обробляє їх та зберігає у базі PostgreSQL. Модуль аналітики формує результати, які повертаються користувачу для відображення.

Висновки та перспективи. В результаті виконаного дослідження встановлено, що ідея створення веб-сайту для моніторингу вживаних автомобілів є доцільною. Такий веб-сайт знизить витрата часу від пошуків по різних платформам, а також відкриємо можливості здійснити аналіз вживаних автомобілів.

Інструменти, які відкрилися в ході вивчення, видалися придатними для створення веб-сайту. Такі як Java, Spring Boot, PostgreSQL, React зможуть виробляти потрібні функції веб-сайту та виконувати основні завдання.

В подальшому можна відділіти веб-сайт від сповіщень про випуск вживаних авто, виводу порівняння вживаних авто, виводу дублікатів від авто, а також створення персональних рекомендацій від веб-сайту для кожного відокремленого відвідача.

Список використаних джерел

1. JDK 26 Documentation - docs.oracle.com – [Електронний ресурс] – Режим доступу: https://docs.oracle.com/en/java/javase/26/

2. Spring Framework - spring.io – [Електронний ресурс] – Режим доступу: https://spring.io/projects/spring-framework

3. React — react.dev – [Електронний ресурс] – Режим доступу: https://react.dev/

4. Joseph Ingeno. Software Architect's Handbook. Packt Publishing Limited, 2018 – 594 p. ISBN-13: 9781788624060

Наконечний Владислав Володимирович
студент 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
(066) 632-85-95
st7966031@stud.duikt.edu.ua

РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ МОНІТОРИНГУ ТА БРОНЮВАННЯ ПАРКУВАЛЬНИХ МІСЦЬ У МІСЬКОМУ СЕРЕДОВИЩІ

Постановка задачі. Стрімке зростання кількості автомобілів у великих містах України спричинило гострий дефіцит організованих паркувальних місць. За даними Головного сервісного центру МВС України, кількість зареєстрованих транспортних засобів у м. Києві за період 2015–2022 років зросла вдвічі – з 695 тисяч до 1,38 мільйона одиниць, тоді як кількість муніципальних паркомісць складає лише близько 30 тисяч [1]. За оцінками дослідників, до 25–40 % міського трафіку у години пік припадає саме на автомобілі, водії яких шукають вільне паркомісце, що спричиняє зростання заторів та шкідливих викидів. Аналіз існуючих систем (SFpark, ParkMobile, SpotHero, Parkopedia, LA Express Park, Barcelona Smart Parking, «Київ Цифровий») засвідчив, що жодна з них не поєднує одночасно чотири ключові компоненти: моніторинг стану паркомісць у реальному часі, попереднє бронювання конкретного місця з розрахунком вартості, прогнозування завантаженості на основі історичних даних та рекомендаційну систему пошуку оптимального паркомісця [2]. Заповнення цього функціонального розриву засобами сучасних веб-технологій та методів інтелектуального аналізу даних визначає актуальність роботи.

Мета дослідження. Метою роботи є розробка інтелектуальної системи моніторингу та бронювання паркувальних місць у міському середовищі у вигляді комплексного вебзастосунку з клієнт-серверною архітектурою для оптимізації управління паркувальним простором на основі статистичних методів аналізу даних. Для досягнення мети визначено такі завдання: проаналізувати предметну область та існуючі системи паркування; обґрунтувати вибір стеку технологій; визначити функціональні та нефункціональні вимоги; спроєктувати чотирирівневу архітектуру, реляційну базу даних та REST API; розробити алгоритм прогнозування завантаженості; виконати програмну реалізацію серверної та клієнтської частин; провести тестування та верифікацію працездатності системи.

Результати дослідження. На основі проведеного порівняльного аналізу обґрунтовано вибір відкритого стеку технологій: серверна частина реалізована мовою Python із використанням мікрофреймворку Flask 3.0 [3], клієнтська частина – на бібліотеці React 18 з інструментом збірки Vite, картографічний сервіс – Leaflet.js у поєднанні з тайлами OpenStreetMap, як СУБД обрано SQLite. Усі компоненти є відкритими (open-source) та не потребують комерційних ліцензій, що мінімізує вартість впровадження.

Спроєктовано чотирирівневу клієнт-серверну архітектуру системи, що включає клієнтський рівень (React SPA), серверний рівень (Flask REST API з шістьма функціональними модулями), рівень даних (реляційна база SQLite з п’ятьма таблицями у третій нормальній формі) та IoT-рівень (модуль програмної імітації сенсорів з можливістю заміни на реальні пристрої). Сформульовано 15

функціональних та 8 нефункціональних вимог, згрупованих за п’ятьма підсистемами. Розроблено REST API із 15 ендпоінтів, стандартизованим форматом помилок (7 HTTP-кодів) та Bearer-токенами для авторизації [4].

Реалізовано модуль автентифікації з хешуванням паролів за алгоритмом SHA-256 та використанням 128-бітної криптографічної солі для кожного облікового запису. Підсистема бронювання реалізує двоетапну валідацію: перевірку фізичної доступності місця та виявлення часових перетинів з існуючими бронюваннями. Розроблено алгоритм прогнозування завантаженості на основі агрегаційних SQL-запитів до журналу спостережень за 30 діб, що обчислює середнє, мінімальне та максимальне значення завантаженості за кожну годину кожного дня тижня. Алгоритм рекомендацій виконує пошук п’яти найближчих зон з вільними місцями на основі евклідової відстані з фільтрацією за наявністю вакантних місць.

Створено інтерактивну карту м. Києва з кольоровою індикацією завантаженості зон, візуальну сітку паркомісць, дашборд з KPI та адміністративну панель з CRUD-операціями. Проведено комплексне тестування за 12 сценаріями. Усі сценарії пройдено успішно; середній час відповіді API не перевищує 200 мс при нормативі 500 мс.

Висновки та перспективи. Розроблена інтелектуальна система моніторингу та бронювання паркувальних місць поєднує функції, які у відомих аналогах реалізовані лише частково, та забезпечує комплексне вирішення задачі управління паркувальним простором у міському середовищі. Модульна архітектура продукту дозволяє його поетапне впровадження муніципальними органами чи приватним бізнесом із можливістю заміни програмної імітації на реальні IoT-пристрої без зміни ядра системи. Перспективами подальшого розвитку є інтеграція з реальними IoT-сенсорами на базі протоколів LoRaWAN та NB-IoT, впровадження моделей машинного навчання (Random Forest, LSTM) для підвищення точності прогнозування, реалізація динамічного ціноутворення на основі попиту, інтеграція з платіжними системами та розробка мобільних додатків для платформ Android та iOS.

Список використаних джерел

1. Скільки машин у Києві – Кількість автомобілів у столиці перевищила мільйон. Економічна правда. 2022. URL: https://www.epravda.com.ua/news/2022/02/11/682305/ (дата звернення: 03.04.2026).
2. Aljohani M., Olariu S., Alali A., Jain S. A survey of parking solutions for smart cities. IEEE Transactions on Intelligent Transportation Systems. 2022. Vol. 23, No 8. P. 10012–10029.
3. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd ed. Sebastopol, CA : O’Reilly Media, 2018. 316 p.
4. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : PhD dissertation. University of California, Irvine, 2000. 162 p.
5. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. [Чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 16 с.

Огнева Софія Євгеніївна
студентка 4 курсу, групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій
(067) 963-28-25
st7892690@stud.duikt.edu.ua

МЕТОДИ ОРГАНІЗАЦІЇ ОБ'ЄКТНОЇ ВЗАЄМОДІЇ В ІНТЕРАКТИВНИХ СЕРЕДОВИЩАХ НА БАЗІ UNREAL ENGINE 5

Постановка задачі. Під час створення сучасних 3D-ігор у жанрі головоломки, найчастіше можна зіштовхнутися з двома основними негараздами. Перший, це якщо писати код до кожної кнопки чи дверей окремо, через що він стає менш зручним для пошуку певних деталей та помилок, і друга це використання важкої графіки та складних матеріалів, що також навантажують комп'ютер, через що в нерідких випадках гра починає гальмувати. Тому для комфоротнішої розробки,дуже важливо знайти спосіб,як зробити архітектуру гри простою для налаштування,а графічні елементи були оптимізованими,щоб гра працювала плавно і не губила гарну картинку.

Мета дослідження. Метою данного дослідження є створити робочий прототип 3D головоломки у Unreal Engine 5,зі зручною взаємодією між гравцем таоб'єктом,а також налаштувати графічну частину для ігрового середовища

Результати. У ході роботи було створено модульну систему логіки,яка дає змогу об'єктам на рівні спілкуватися між собою без прямого зв'язку. Для цього замість звичайного коду було реалізовано універсальний компонент взаємодії та особливий інтерфес. Це дає змогу зробити так,щоб будь який елемент міг надсилати сигнал відкриття дверей чи активування головоломки за єдиним зразком. Окрім створення логіки, значну увагу було приділено візуальній частині,так як у іграх жанру головоломки,гравці знаходять підказки часто зором,дивлячись на стіни і різні об'єкти, а також, саме атмосфера допомогає гравцю погрузитися у саму гру. Для цього всередині редактора Unreal Engine 5 було створено базовий матеріал для поверхнень, а також можливість,накласти чи створити модельку стіни за кількох хвилин.

На зображені можна побачити структуру матеріалу. Замість того, щоб просто накласти один малюнок,ми з'єднуємо кілька різних текстурних шарів. Base Color відповідає за основний колір і візирунок нашої стіни, Normal Map відповідає за об'єм та рельєфу завдяки чому стіна виглядає більш живою,а Roughness відповідає за відбивання світла на поверхні. Таке налаштування дозволяє системі робити світлові переливи на стіні у реальному часі,що додає реалістичності навіть у простій стилістиці.

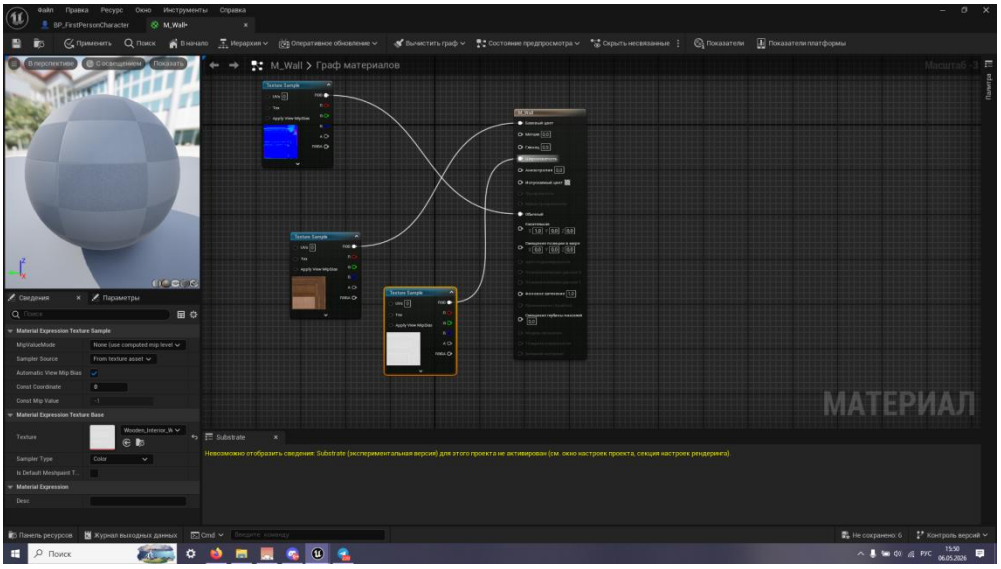


Рис. 1. Налаштування матеріалу для стіни в редакторі

Висновки. Застосування інструментарію Unreal Engine 5 дає змогу швидко створити якісну гру невеликим командам. Створення універсального інтерфейсу для взаємодії полегшує працю ,та дозволяє не сидіти над кодом,що робить проект гнучким, і дозволяє доволі швидко створити новий рівень. А налаштування текстур довзоляє створити простий,але приємний візуал без зниження продуктивності. Цю механіку можна використовувати не лише для ігор,але й для різних галузей освітніх моделей та формування віртуальної картинки.

Список використаних джерел

1. Yasufuku K., Katou G., Shoman S. Game Engine (Unity, Unreal Engine). The Journal of The Institute of Image Information and Television Engineers. 2017. T. 71, № 5. C. 353 —357.

2. Коляда М. Unreal Engine 3: як ігровий рушій змінив індустрію 2000-х. IT community. URL: https://itc.ua/ua/articles/unreal-engine-3-yak-igrovyj-rushij-zminyv-industriyu-2000-h/.

3. Parrish Z. Introduction to Blueprints. URL: https://www.unrealengine.com/blog/introduction-to-blueprints.

4. Степорук Н. Огляд технологій Unreal Engine 5 з розробниками: застосування, переваги та перспективи. gamedev. URL: https://gamedev.dou.ua/articles/unreal-engine-technologies-review/.

Мацела Микола Романович
студент 4 курсу, групи ПД-43
Державний університет інформаційно-комунікаційних технологій
aergame333@gmail.com

АРХІТЕКТУРНІ ПАТЕРНИ ІНТЕГРАЦІЇ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ (LLM) У КЛІЄНТСЬКІ ВЕБЗАСТОСУНКИ

Постановка задачі. З розвитком генеративного штучного інтелекту зростає попит на інтерактивні вебзастосунки, що використовують великі мовні моделі (LLM) у реальному часі, зокрема в індустрії цифрових ігор та інтерактивних сюжетів. Однак більшість існуючих рішень стикаються з проблемою високої затримки генерації тексту (latency) та складністю управління контекстом сесій, що негативно впливає на користувацький досвід та зв'язність наративу під час взаємодії користувача із системою.

Мета дослідження. Дослідження та розробка ефективної архітектури для інтеграції великих мовних моделей у клієнтські вебзастосунки з метою мінімізації затримки генерації відповідей та забезпечення надійного збереження довгострокового ігрового контексту.

Результати дослідження. Під час дослідження було спроєктовано та реалізовано прототип інтелектуальної системи генерації динамічних сюжетів (вебзастосунок на основі ШІ-Майстра) з використанням сучасного стеку технологій. Для клієнтської частини використано фреймворк Next.js (React), що забезпечує швидкий рендеринг інтерфейсу. Backend-частина реалізована на базі хмарної інфраструктури Supabase (PostgreSQL), де для ефективного збереження ігрового стану та історії чату застосовано формат даних JSONB.

Для вирішення проблеми високої затримки (latency) замість традиційних GPU-рішень було інтегровано API від Groq, що використовує спеціалізовані процесори LPU (Language Processing Units) та модель Llama 3.3. Це дозволило досягти часу відгуку системи на дії гравця менше ніж 2,5 секунди. Запропонована архітектура дозволяє ефективно керувати пам’яттю ігрової сесії та контролювати "галюцинації" ШІ за допомогою жорстких системних промптів без деградації продуктивності.

Висновки та перспективи. Застосування спеціалізованих апаратних рішень (LPU) у поєднанні з оптимізованою хмарною реляційною базою даних для збереження контексту є ефективним архітектурним патерном для розробки інтерактивних LLM-застосунків. Перспективою подальших досліджень є масштабування розробленої системи для підтримки великої кількості паралельних ігрових сесій та впровадження методів векторизації (RAG) для покращення довгострокової пам’яті ШІ-моделі у складних ігрових сценаріях.

Список використаних джерел

1. Vercel Inc. Next.js Documentation: App Router and Client Components. URL: https://nextjs.org/docs
2. Supabase Inc. Supabase Database and PostgreSQL jsonb guide. URL: https://supabase.com/docs/guides/database/json
3. Groq Inc. Groq API Documentation & LPU Inference Engine. URL: https://console.groq.com/docs/quickstart

Яременко Олександр Андрійович
студент групи ТЦР-43
Державний університет інформаційно-комунікаційних технологій
Науковий керівник: Ананченко О.Є., доцент кафедри, кандидат технічних наук

ІНТЕЛЕКТУАЛЬНА СИСТЕМА АВТОМАТИЗАЦІЇ НАГАДУВАНЬ НА БАЗІ TELEGRAM-БОТА

Постановка задач

У сучасному цифровому середовищі ефективне управління часом та персональними задачами є важливою складовою продуктивності людини. Більшість існуючих систем планування мають складний інтерфейс або потребують значної кількості дій для створення та редагування задач. Актуальним напрямом є створення системи автоматизації нагадувань, яка дозволяє користувачу взаємодіяти із сервісом через месенджер Telegram.

Мета дослідження

Основна мета кваліфікаційної роботи полягає в підвищенні ефективності управління особистими завданнями та нагадуваннями шляхом впровадження Telegram-бота з функцією інтелектуального аналізу текстових повідомлень користувача. Для досягнення цієї мети були визначені такі завдання: - здійснити аналіз сучасних платформ для планування задач і цифрових сервісів нагадувань; - дослідити можливості Telegram-ботів та функціонал Telegram Bot API для автоматизації управління завданнями; - розробити архітектуру програмної системи та створити основні модулі управління задачами; - реалізувати механізми автоматичного визначення часу виконання й пріоритетності завдань на основі аналізу текстових даних; - провести тестування системи й оцінити її функціональність та точність роботи.

Результати

Під час роботи над кваліфікаційною роботою проведено ґрунтовний аналіз сучасних платформ для планування завдань і сервісів цифрових нагадувань, серед них: Todoist, Notion, Google Calendar і Microsoft To Do. У дослідженні було розглянуто функціональні особливості цих систем, їхній користувацький інтерфейс, механізми роботи з нагадуваннями, а також переваги та недоліки застосовуваних підходів до управління особистими завданнями. Отримані результати стали основою для визначення вимог до створення власного програмного продукту. Для реалізації програмного рішення обрано сучасний технологічний стек, що включає мову програмування Java, фреймворк Spring Boot, Telegram Bot API, PostgreSQL, Hibernate та Spring Data JPA. Використання Java і Spring Boot сприяло побудові модульної структури серверної частини, забезпечуючи її стабільність та можливість масштабування рис 1.1.

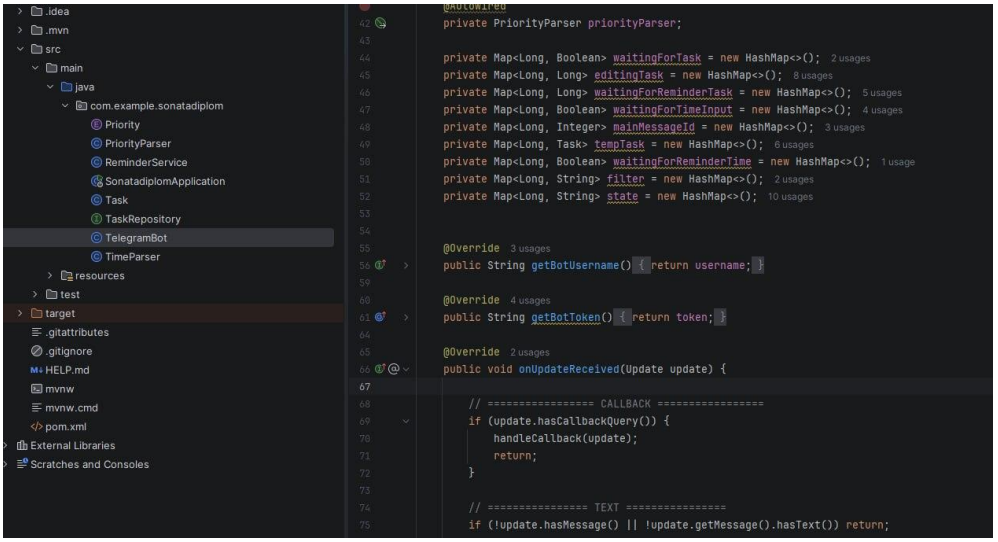


рис. 1.1 Структура бота

База даних PostgreSQL забезпечила надійне збереження даних користувачів і завдань, а впровадження Hibernate та Spring Data JPA істотно спростило процес взаємодії із базою даних та сприяло інтеграції між компонентами системи. До того ж було розроблено зручний та інтуїтивний інтерфейс рис 2.2.

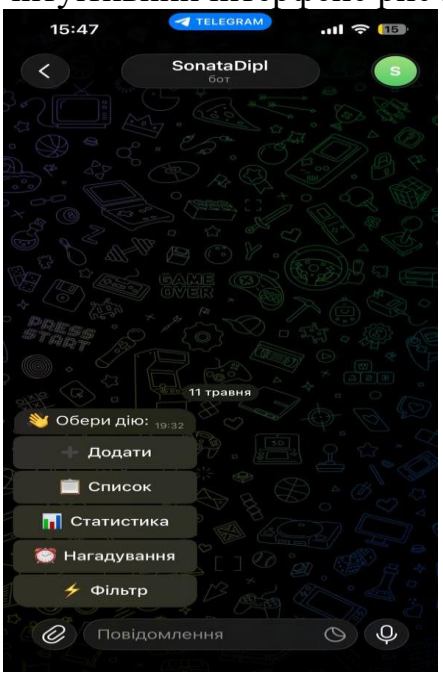


рис. 1.2 інтерфейс бота

Розроблена програма дозволяє виконувати основні операції із завданнями - створення, редагування, перегляд та видалення - безпосередньо через Telegram-бота. Такий підхід усуває потребу у використанні окремого мобільного додатка чи вебінтерфейсу. Система організована у формат діалогового спілкування, що робить її зручною й інтуїтивною для користувачів, зокрема новачків. Однією з ключових особливостей розробленого рішення є інтеграція спеціалізованих модулів TimeParser та PriorityParser. Модуль TimeParser автоматично визначає часові параметри завдань на основі аналізу природної мови. Це дозволяє системі розпізнавати фрази типу «завтра о 14:00» або «через 2 години». PriorityParser, у свою чергу, здійснює визначення рівня пріоритетності завдання шляхом аналізу ключових слів і контексту повідомлення. Завдяки такій автоматизації

користувачам не потрібно вручну задавати всі параметри задачі, що значно економить час та підвищує ефективність у керуванні персональними справами.

Висновки

У результаті виконання роботи було розроблено цифрову систему автоматизації нагадувань на базі Telegram-бота. Запропоноване рішення дозволяє спростити процес створення та управління задачами, зменшити складність взаємодії користувача із системою планування та скоротити час на введення інформації.

Список використаних джерел

1. Telegram Bot API Documentation. URL: https://core.telegram.org/bots/api
2. Spring Boot Reference Documentation. URL:
https://spring.io/projects/spring-boot
3. Oracle Java Documentation. URL: https://docs.oracle.com/javase/
4. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/

Маковець Владислав Валерійович,
студент 4-го курсу, групи ПД-41,
Державного університету інформаційно-комунікаційних технологій
(098)362-88-86
st5387147@stud.duikt.edu.ua

ЗАСТОСУНОК «СТОП ТРИВОГА» - ЕФЕКТИВНЕ ЦИФРОВЕ РІШЕННЯ ДЛЯ РАННЬОГО ВИЯВЛЕННЯ ТРИВОЖНИХ РОЗЛАДІВ

Постановка задачі. «Стоп тривога» як сучасний цифровий інструмент для раннього виявлення тривожних розладів.

У період повномасштабної війни велика частина населення України перебуває під впливом значного емоційного навантаження, психологічного стресу та симптомів посттравматичного стресового розладу. Наявні сервіси психологічної допомоги, незважаючи на їхню важливість, не здатні повною мірою охопити всі категорії користувачів, а також часто потребують стабільного доступу до інтернету або не підтримують українську локалізацію. У зв'язку з цим особливо важливим є створення цифрового рішення, яке може функціонувати автономно в складних польових умовах, забезпечуючи конфіденційність даних та гуманний користувацький досвід UX.

Мета дослідження. Головною метою є допомога та надання психологічної підтримки користувачам завдяки розробки мобільного застосунку «Стоп тривога», який інтегрує інструменти когнітивної та поведінкової терапії, елементи сторітелінгу та механіки гейміфікації. Застосунок повинен адаптувати контент відповідно до емоційного стану користувача, забезпечувати роботу в офлайн-режимі, знижувати ризики ретравматизації та залишатися інтуїтивно зрозумілим для широкої аудиторії - від ветеранів до цивільного населення, що перебуває в зонах ризику.

Результати дослідження. Архітектура застосунку. Застосунок реалізований на основі багатошарової архітектури (Рис. 1) із чітким розмежуванням відповідальностей:

− UI-шар реалізує адаптивний інтерфейс відповідно до актуальних стандартів Material Design 3.

− Шар бізнес-логіки забезпечує обробку емоційних записів, формування рекомендацій, реалізацію механік гейміфікації та роботу SOS-модуля.

− Ядро (Core) створене мовами C/C++ для досягнення високої продуктивності та зменшення енергоспоживання.

− Шар зберігання побудований на основі SQLite із застосуванням SQLCipher для шифрування даних, що забезпечує автономність без втрати рівня безпеки.

− Для асинхронної синхронізації інформації за наявності мережевого підключення використовується REST API з підтримкою сучасних механізмів захисту.

Контент кешується для забезпечення офлайн-доступу, зокрема вправи, інструкції та інтервенційні сценарії.

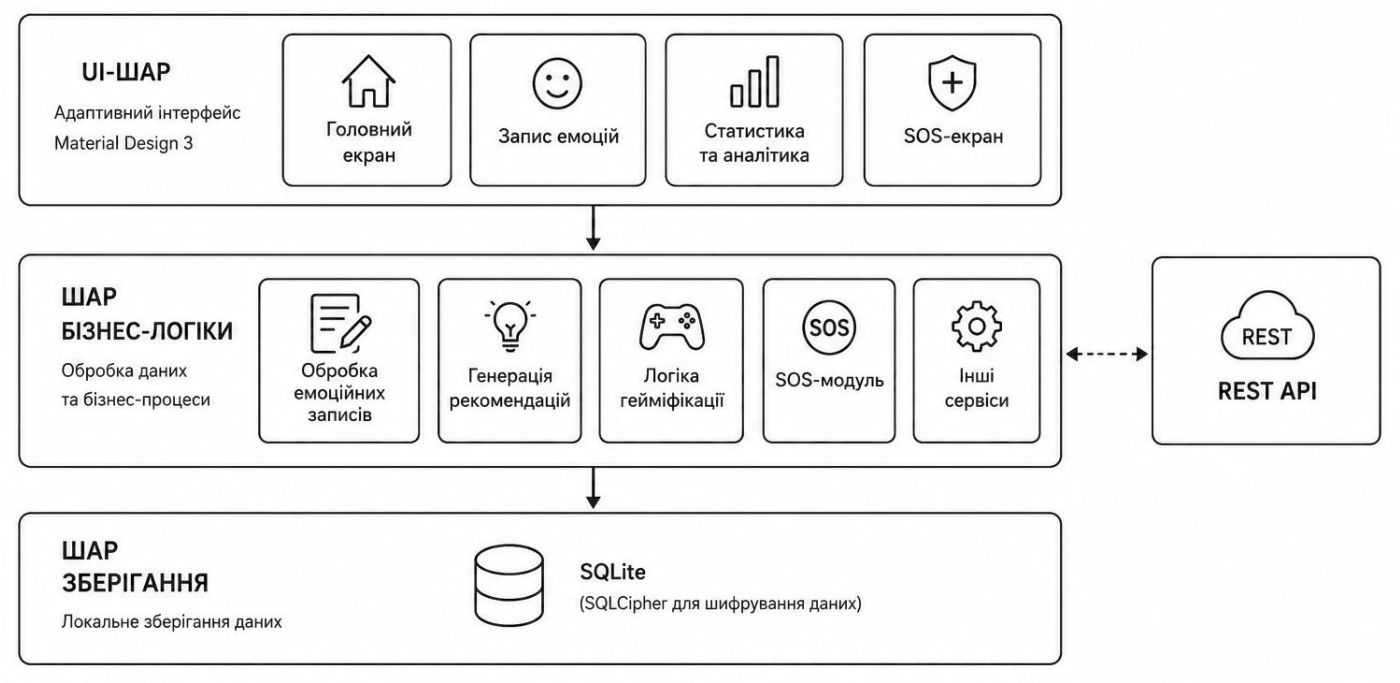


Рис. 1 Архітектура застосунку

Висновки та перспективи. «Стоп тривога» є прикладом ефективного поєднання технічної надійності, етичного дизайну та сучасних психотерапевтичних підходів у цифровому форматі. Успішна валідація функціоналу відкриває шлях до державного масштабування впровадження багатомодальної терапії та сертифікації за міжнародними стандартами на кшталт DiGA.

Список використаних джерел:

1. Jabangwe R., Edison H., Nguyen-Duc A. Software engineering process models for mobile app development: A systematic literature review // Journal of Systems and Software. – 2018.

2. Harari G. M., Lane N. D., Wang R., Crosier B. S., Campbell A. T., Gosling S. D. Using smartphones to collect behavioral data in psychological science // ScienceDirect. – 2016.

3. Deterding S., Dixon D., Khaled R., Nacke L. From game design elements to gamefulness: Defining gamification // ACM Digital Library. – 2011.

4. Zhang X., Breitinger F., Luechinger E. та ін. Android application forensics: A survey of obfuscation and deobfuscation techniques // Forensic Science International: Digital Investigation. – 2021.

Велієв Раміль Велієв
Студент 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
(098) 721-22-92
ramilius2004@gmail.com

ЦИФРОВА ПЛАТФОРМА АВТОМАТИЗАЦІЇ КОНТРОЛЮ РОБОЧОГО ЧАСУ ПРАЦІВНИКІВ ТА МОНІТОРИНГУ ВИКОНАННЯ ЗАВДАНЬ

Постановка задачі. У сучасних компаніях, діяльність яких пов'язана з будівництвом, сервісним обслуговуванням та виконанням робіт на різних об'єктах, важливим завданням є контроль робочого часу працівників та моніторинг виконання поставлених завдань. Більшість підприємств досі використовує паперові журнали, електронні таблиці або месенджери для обліку робочого часу, що ускладнює централізоване зберігання інформації та контроль діяльності персоналу. Тому актуальною є розробка цифрової платформи, яка забезпечить автоматизацію обліку робочого часу, контроль присутності працівників на об'єктах та моніторинг виконання робіт.

Мета дослідження. Метою дослідження є розробка цифрової платформи автоматизації контролю робочого часу працівників, моніторингу виконання завдань та оптимізації робочих процесів із використанням механізмів check-in/check-out та геолокаційного контролю.

Результати дослідження. У межах дослідження проведено аналіз існуючих систем контролю робочого часу та управління персоналом, зокрема платформ Connecteam та Deputy. За результатами аналізу встановлено, що існуючі рішення не повністю враховують специфіку компаній, які виконують роботи на різних об'єктах та потребують контролю фактичної присутності працівників на місці виконання завдань.

Для вирішення зазначеної проблеми розроблено цифрову платформу, побудовану за клієнт-серверною архітектурою. Серверна частина реалізована з використанням Node.js та NestJS, адміністративна веб-панель – на основі Next.js. Для зберігання даних використовується реляційна база даних PostgreSQL.

Основний функціонал системи включає реєстрацію та авторизацію користувачів за номером телефону та OTP-кодом, управління компаніями, проєктами, адресами та робочими завданнями. Працівник може виконувати check-in та check-out із автоматичною фіксацією геолокації, вибором об'єкта та задачі. Додатково реалізовано механізм завантаження фотографій до та після виконання робіт для підтвердження результатів діяльності.

Для адміністратора розроблено веб-панель, яка дозволяє контролювати активність працівників, переглядати історію робочих сесій, аналізувати відпрацьований час та відстежувати стан виконання завдань у режимі реального часу.

Проведене функціональне тестування підтвердило коректність роботи механізмів авторизації, обліку робочого часу, синхронізації даних та формування звітної інформації.

Висновки та перспективи. У результаті дослідження розроблено цифрову платформу, яка забезпечує автоматизацію контролю робочого часу працівників, моніторинг виконання завдань та централізований облік робочих процесів.

Використання геолокації та механізму check-in/check-out дозволяє підвищити достовірність обліку робочого часу та покращити контроль виконання робіт.

Перспективою подальшого розвитку системи є впровадження алгоритмів штучного інтелекту для аналізу продуктивності працівників, прогнозування термінів виконання завдань та автоматичного формування аналітичної звітності.

Список використаних джерел

1. Node.js Documentation: https://nodejs.org/docs/latest/api/
2. NestJS Documentation: https://docs.nestjs.com/
3. Next.js Documentation: https://nextjs.org/docs
4. Connecteam: https://connecteam.com/
5. Deputy: https://www.deputy.com/

Бондаренко Єлизавета Романівна
студентка 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
(097) 228-92-41
l.bondarenko2504@gmail.com

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ МОБІЛЬНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ЗАПИСУ ПАЦІЄНТІВ НА ПРИЙОМ ДО ЛІКАРЯ

Постановка задачі. У сучасному світі є одна річ, яка не змінюється роками — це важливість доступної медичної допомоги всім групам населення та в будь-якій частині світу. У книзі британського нейрохірурга Генрі Марша «Щоденник нейрохірурга» описано досвід надання медичної допомоги в Непалі, де пацієнти змушені долати значні відстані та витрачати багато часу на очікування у чималих чергах, щоб потрапити до необхідного фахівця [1]. Обмежений доступ до сучасних технологій і кваліфікованих спеціалістів ускладнює процес лікування. Це підкреслює необхідність розвитку доступних цифрових медичних сервісів.

В умовах стрімкого розвитку інформаційних технологій, безумовно, важливу роль у забезпеченні доступності різноманітних послуг відіграють мобільні застосунки. Вони дозволяють спростити процес отримання інформації, взаємодії з сервісами та доступу до необхідних послуг. Якщо говорити про медицину, то це пошук лікаря, отримання інформації та запис на прийом. Використання мобільних застосунків забезпечує зручний доступ до медичних сервісів у будь-який час та сприяє підвищенню рівня обслуговування пацієнтів. Особливо це актуально для батьків з малолітніми дітьми та людей похилого віку. Таким чином, виникає необхідність розробки мобільної інформаційної системи, що дозволить оптимізувати процес взаємодії пацієнтів із медичними установами.

Згідно з дослідженнями у сфері цифрової медицини, впровадження мобільних сервісів дозволяє підвищити доступність медичних послуг [2]. Використання мобільних застосунків забезпечує оперативний доступ до інформації про лікарів, графіків прийому та наявних вільних часових слотів, що значно спрощує процес запису на прийом. Крім того, цифрові сервіси сприяють зменшенню навантаження на реєстратури, підвищенню ефективності використання ресурсів та покращенню якості обслуговування пацієнтів.

Мета дослідження. Метою роботи є розробка мобільної інформаційної системи для вибору лікаря та запису на прийом із використанням сучасних технологій, спрямованої на підвищення доступності медичних послуг.

Результати. У роботі проаналізовано існуючі мобільні застосунки та технології для організації запису на прийом до лікаря, такі як Doc.ua, Doctoralia та Medics [3]. На основі проведеного аналізу встановлено, що для пацієнтів важливим є отримання максимально повної інформації про лікаря перед записом на прийом. Крім того, важливим є забезпечення можливості прямого зв’язку з медичними фахівцями для отримання консультацій. Також визначено доцільність реалізації функції відображення місця розташування клініки, де здійснює прийом лікар, що спрощує процес вибору для пацієнтів.

У процесі розробки реалізовано основні функціональні можливості застосунку, а саме: перегляд списку лікарів, фільтрацію за категоріями, перегляд детальної інформації про лікаря, запис на прийом, можливість обміну

повідомленнями, здійснення телефонного дзвінка та відображення місцезнаходження клініки. Особливу увагу приділено простоті інтерфейсу та доступності основних функцій, що дозволяє використовувати застосунок користувачам різного віку. Також передбачено можливість подальшого розширення функціоналу — шляхом додавання нових сервісів та інтеграції з іншими інформаційними системами.

Для реалізації програмного продукту використано середовище розробки Android Studio та хмарні сервіси Firebase [4;5], які забезпечують зберігання, обробку та синхронізацію даних у режимі реального часу. Використання таких технологій дозволяє підвищити надійність системи та забезпечити швидкий доступ до інформації. Розроблений застосунок орієнтований на забезпечення зручної та швидкої взаємодії користувача з системою.

Висновки. Використання мобільних технологій у сфері медицини відкриває нові можливості для підвищення ефективності надання медичних послуг. Запропоноване рішення може бути адаптоване для використання в різних медичних установах та сприяти подальшій цифровізації сфери охорони здоров'я. Крім того, подальший розвиток системи може передбачати розширення функціональних можливостей, інтеграцію з іншими інформаційними ресурсами та впровадження додаткових сервісів для підвищення якості обслуговування пацієнтів.

Список використаних джерел

1. Marsh H. Do No Harm: Stories of Life, Death and Brain Surgery. London: Weidenfeld & Nicolson, 2014.
2. World Health Organisation. Global strategy on digital health 2020–2025. URL: https://www.who.int/publications/i/item/9789240020924 (дата звернення: 21.04.2026).
3. Medics. Медичний онлайн-сервіс для запису до лікаря. URL: https://medics.ua/ (дата звернення: 21.04.2026).
4. Android Developers. Android Developers Documentation. URL: https://developer.android.com/ (дата звернення: 21.04.2026).
5. Firebase. Firebase Documentation. URL: https://firebase.google.com/docs (дата звернення: 21.04.2026).

Хава Дмитро Віталійович
студент 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
(096)613-23-68
gowes111234i@gmail.com

ЦИФРОВА СИСТЕМА ОПТИМІЗАЦІЇ ПРОЦЕСІВ ПРИЙОМУ, ОБРОБКИ ТА УПРАВЛІННЯ ЗАМОВЛЕННЯМИ У РЕСТОРАННОМУ БІЗНЕСІ НА ОСНОВІ PYTHON

Постановка задачі. В Україні налічується понад 25 тисяч закладів ресторанного господарства, близько 45% яких досі використовують паперові системи обліку замовлень. Сучасні комерційні POS-системи (Toast, Square, Lightspeed, Poster POS) або мають непропорційно високу вартість для малих закладів (від $30–69/місяць), або не забезпечують необхідного рівня кастомізації та самообслуговування. Класична схема «офіціант – паперовий блокнот – кухня» породжує помилки передачі замовлень, втрату часу на переміщення між залом і кухнею та відсутність оперативної аналітики. Актуальність дослідження обумовлена потребою у простій, функціонально повноцінній і легкій у розгортанні цифровій системі для малих закладів формату fast-casual, що дозволяє скоротити операційні витрати та підвищити швидкість обслуговування завдяки моделі самообслуговування гостя через QR-код.

Мета дослідження. Метою роботи є проєктування та реалізація цифрової системи управління замовленнями ресторанного закладу мовою Python із використанням сучасного асинхронного веб-стеку (FastAPI, SQLAlchemy 2.0, WebSocket, Jinja2), що забезпечує модель самообслуговування гостя через QR-код, передачу замовлення на кухню у реальному часі, кастомізацію страв, три режими спліту рахунку, AI-рекомендації та трекінг готовності замовлення з прогрес-баром. Для досягнення мети вирішено такі задачі: проведено порівняльний аналіз комерційних POS-систем; сформульовано функціональні й нефункціональні вимоги; спроєктовано трирівневу архітектуру та базу даних з 10 сутностей у 3NF; реалізовано серверну та клієнтську частини; проведено модульне, інтеграційне та навантажувальне тестування.

Результати. Розроблено повнофункціональний веб-застосунок з трирівневою клієнт-серверною архітектурою. Серверна частина включає 35 REST-ендпоінтів із JWT-автентифікацією та рольовим доступом (адміністратор, кухар, ранер, гість) і 4 WebSocket-канали: загальний канал кухні, канал ранера, адміністративний та індивідуальний канал трекінгу для гостя. База даних містить 10 таблиць у третій нормальній формі: User, Category, Dish, Table, Order, OrderItem, Payment, SplitPart, Reservation, AuditLog.

Реалізовано чотири ключові модулі: (1) конструктор страви – модальне вікно з видаленням інгредієнтів, додаванням топінгів із ціновими надбавками та полем алергій; (2) AI-рекомендації (cross-sell / upsell) на rule-based логіці, готовій до заміни на ML-модель; (3) спліт рахунку у трьох режимах – один платіж, рівний поділ на N осіб, поділ по стравах між гостями; (4) сторінка трекінгу з прогрес-баром на 5 кроків і WebSocket-оновленнями у реальному часі.

Додатково реалізовано: календарне бронювання столів із виявленням накладок і автоматичним скасуванням «no-show» (APScheduler); аналітичний модуль із графіками виручки (Chart.js) та експортом у багатолистовий Excel (openpyxl); генератор PDF-чеків із підтримкою кирилиці (ReportLab); журнал дій (audit log); тарифна SaaS-модель (Basic / Pro / Enterprise). За результатами навантажувального тестування (Locust, 50 одночасних користувачів): середній час відгуку – 78 мс, помилок не зафіксовано. Час оформлення замовлення скоротився з 3 хв 15 с до 42 с – у 4,6 раза. Покриття тестами (pytest) – 87% ключових модулів.

Висновки. Розроблена система вирішує актуальну проблему цифровізації малих закладів ресторанного господарства, забезпечуючи функціональність, порівнянну з провідними комерційними POS-системами, при значно нижчій собівартості та можливості локального розгортання. Поєднання WebSocket-каналів реального часу, AI-рекомендацій, QR-самообслуговування та гнучкого спліту рахунку в єдиній Python/FastAPI-системі є науковою новизною роботи. Практичне значення підтверджено 4,6-кратним скороченням часу оформлення замовлення та продуктивністю 78 мс при 50 одночасних користувачах. Перспективою розвитку є інтеграція повноцінної ML-моделі рекомендацій та мобільного додатку для персоналу.

Список джерел

1. Tiangolo S. FastAPI Documentation [Електронний ресурс]. – Режим доступу: https://fastapi.tiangolo.com/

2. Bayer M. SQLAlchemy 2.0 Documentation [Електронний ресурс]. – Режим доступу: https://docs.sqlalchemy.org/

3. TechEmpower Web Framework Benchmarks. Round 22 [Електронний ресурс]. – Режим доступу: https://www.techempower.com/benchmarks/

4. Restaurants Association of Ukraine. Звіт про стан ресторанного бізнесу в Україні 2024. – Київ, 2024. – 48 с.

5. Pew Research Center. Social Anxiety and Dining Preferences Among Young Adults. – Washington, 2023.

Дацко Миколай Анатолійович
студент 4 курсу, групи ТЦР-41
Державний університет інформаційно-комунікаційних технологій
+38 (068) 492-26-33
m.datsko2205@gmail.com

ІМПЛЕМЕНТАЦІЯ БІНАРНОЇ СЕРІАЛІЗАЦІЇ ДЛЯ ВЛАСНОГО ТИПА ЗА ДОПОМОГОЮ БІБЛІОТЕКИ “BORSH” МОВИ ПРОГРАМУВАННЯ “RUST”

Постановка задачі. Серіалізація у бінарний формат є кращим способом підготовки даних для їхнього збереження на диск чи їхнього надсилання мережею ніж текстова серіалізація: дані серіалізовані у бінарному вигляді мають менший розмір та швидше обробляються комп’ютерами. Попри це бінарна серіалізація часто взагалі не викладається, тому актуальним є вивчення роботи з нею.

Мета дослідження. Метою дослідження є вивчення роботи з бінарною серіалізацією на прикладі бінарного формата бібліотеки “Borsh”, аналіз формата окремих типів з стандартної бібліотеки мови “Rust” у вихідному бінарному об’єкті, ручна імплементація серіалізації та десеріалізації для власного типа даних для цієї бібліотеки.

Результати дослідження. Щоб читати дані із бінарного об’єкта програмі яка його парсить треба знати три речі: де у об’єкті знаходяться дані, який розмір ці дані мають, та як їх інтерпретувати. Деякі бінарні файли мають формат який дозволяє записувати у них дані різної структури додаючи до файла інформацію про тип та структуру даних, наприклад “BSON” та “Protocol Buffer”, такі формати називаються самоописуючими. Не самоописуючі формати розраховують на те що програма яка парситиме об’єкт знатиме його структуру.

“Borsh” це бібліотека для мови “Rust” яка дозволяє серіалізувати та десеріалізувати дані у власний простий не самоописуючий бінарний формат. Серіалізувати та десеріалізувати можна дані які імплементують інтерфейси “BorshSerialize” та “BorshDeserialize” відповідно. Кожен інтерфейс імплементований для набора стандартних типів Rust. Інтерфейси мають “derive” макроси які можна повісити на структуру і це імплементує інтерфейс для неї якщо всі емеленти структури його імплементують, як показано на рис 1.

```
#[derive(BorshSerialize, BorshDeserialize)]
2 implementations
struct Simple{
    number: u64,
    string: String,
    vector: Vec<bool>
}
```

Рис 1. Імплементація інтерфейсів “BorshSerialize” та “BorshDeserialize” для простої структури “Simple” що складається лише з стандартних типів.

Щоб розглянути формат серіалізації на практиці сруктуру “Simple” було серіалізовано як показано на рис 2. На рис 3 наведено файл із результатом серіалізації. Перші 8 байтів (“0C” та 7 байтів “00”) це число 12 із поля “number”. 4 байти з позиції “08” на першому рядку, “05 00 00 00”, це 32-бітне число 5 що позначає кількість букв у полі “string”, а 5 байти з “0C”, “48 65 6C 6C 6F” це слово “Hello” записане у “UTF-8”. 4 байти з “01” другого рядка, “03 00 00 00” це 32-бітне число 3 що позначає кількість елементів у векторі “vector”. Кожен з байтів “01 01 00” на позиції “05” рядка 2 кодує одне булеве значення у векторі.

```
let save_file1: File = File::create(path: "./simple_struct").unwrap();
let simple_struct: Simple = Simple{
    number: 12,
    string: "Hello".into(),
    vector: vec![true, true, false]
};
_ = borsh::to_writer(writer: save_file1, value: &simple_struct);
```

Рис 2. Код мови “Rust” що створює екземпляр структури “Simple” та серіалізує її у файл.

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0C 00 00 00 00 00 00 00 05 00 00 00 48 65 6C 6C
6F 03 00 00 00 01 01 00 +
```

Рис 3. Шістнадцятковий вигляд серіалізованої структури “Simple”. Жирні цифри зверху – номери стовбців.

Borsh використовує дуже простий алгоритм серіалізації: алгоритм йде по полям структури в порядку їх оголошення, кожне поле перетворюється у бінарний формат і записується у вивід. Тобто всі дані просто записані одні за одним у бінарному вигляді.

Якщо поле містить структуру алгоритм іде у структуру та виконує такий самий алгоритм для неї до її кінця після чого виходить і йде до наступного поля структури що серіалізується. На рис 4 наведено структуру “DoubleSimple” яка містить дві структури “Simple” із 64-бітним числом між ними. Рис 5 показує код що проводить серіалізацію екземпляра структури “DoubleSimple” що містить два екземпляра “Simple” такі ж як у попередньому прикладі та максимальне 64-бітне число між ними. На рис 6 наведено результат серіалізації, де видно дві структури “Simple” у такому ж вигляді як і у попередньому прикладі але розділені вісьмома парами “FF”, що кодують десяткове число “18.446.744.073.709.551.615” яке є максимальним значенням 64-бітного бінарного числа без знака.

Примітивні типи та кілька примітивних колекцій мають формат визначений специфікацією. Веб сторінка за адресою “https://borsh.io/” містить специфікацію формата серіалізації, рис 7.

```
#[derive(BorshSerialize, BorshDeserialize,
3 implementations
struct Simple{
    number: u64,
    string: String,
    vector: Vec<bool>
}
```

Рис 4. Структура “DoubleSimple”.

```
let simple_struct: Simple = Simple{
    number: 12,
    string: "Hello".into(),
    vector: vec![true, true, false]
};
let double_simple: DoubleSimple = DoubleSimple{
    first_simple: simple_struct.clone(),
    padding: u64::MAX,
    second_simple: simple_struct
};
let save_file: File = File::create(path: "./double_simple").unwrap();
_ = borsh::to_writer(writer: save_file, value: &double_simple);
```

Рис 5. Код що серіалізує екземпляр структури “DoubleSimple”.

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0C 00 00 00 00 00 00 00 05 00 00 00 48 65 6C 6C
6F 03 00 00 00 01 01 00 FF FF FF FF FF FF FF FF
0C 00 00 00 00 00 00 00 05 00 00 00 48 65 6C 6C
6F 03 00 00 00 01 01 00 +
```

Рис 6. Шістнадцятковий вигляд серіалізованої структури “DoubleSimple”. Жирні цифри зверху – номери стовбців.

Informal type	Rust EBNF *	Pseudocode
Integers	integer_type: ["u8" | "u16" | "u32" | "u64" | "u128" | "i8" | "i16" | "i32" | "i64" | "i128"]	little_endian(x)
Floats	float_type: ["f32" | "f64"]	err_if_nan(x) little_endian(x as integer_type)
Unit	unit_type: "()"	We do not write anything

Bool	boolean_type: "bool"	if x {
		repr(1 as u8)
		} else {
		repr(0 as u8)
		}

Fixed sized arrays	array_type: '[' ident ';' literal ']'	for el in x repr(el as ident)
Dynamic sized array	vec_type: "Vec<" ident '>'	repr(len() as u32) for el in x repr(el as ident)

Рис 7. Фрагмент специфікації серіалізації “Borsh”.

Якщо структура містить типи для яких функції бібліотеки “Borsh” не імплементовані їх можна імплементувати вручну. Рис 8 показує структуру “CustomUrlVector” яка містить вектор із типом “Url” бібліотеки “url”. Тип “Url” це структура що використовується для роботи із URL рядками, вона імплементує серіалізацію та десеріалізацію для бібліотеки “Serde” але не імплементує їх для бібліотеки “Borsh”.

Рис 9 та 10 показують код що імплементує серіалізацію та десеріалізацію для структури “CustomUrlVector”. Кожна з них це функція яка працює із спеціальним об’єктом у змінних “reader” чи “writer” який представляє вміст серіалізованих даних. Під час серіалізації програма покроково записує дані у “reader”: спочатку одне 8-байтове число “usize” (“usize” має 4 байти у програмах зібраних для 32-бітних систем) що описує кількість елементів вектора, що відрізняється від специфікації “Borsh” яка використовує 4-байтові числа для кількості елементів у колекціях; потім дані текстових рядків що містять URL із оброблюваних даних, де для кожного рядка використовується функція імплементована бібліотекою “Borsh” яка записує текстові рядки у форматі визначеному специфікацією – одне 4-байтове число що описує кількість букв та всі букви по порядку. Під час десеріалізації функція покроково читає та інтерпретує дані і збирає вектор відповідно – спочатку читається довжина вектора, потім створюється вектор який міститиме вивід а потім покроково читаються всі текстові рядки які представляють URL.

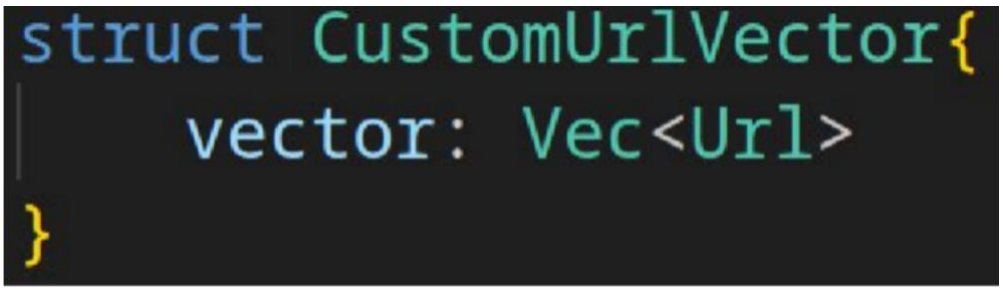


Рис 8. Структура “CustomUrlVector”.

```rust
impl BorshSerialize for CustomUrlVector{
    fn serialize<W: std::io::prelude::Write>(&self, writer: &mut W) -> std::io:
        // Крок 1 - записати розмір вектора
        usize::serialize(&self.vector.len(), writer)?;
        // Крок 2 - серіалізувати всі елементи вектора один за одним
        for url: &Url in self.vector.iter(){
            // Сереалізує рядок згідно формата з специфікації
            borsh::ser::BorshSerialize::serialize(self: url.as_str(), writer)?;
        }
        return Ok(())
    }
}
```

Рис 9. Імплементація серіалізації для структури “CustomUrlVector”.

```rust
impl BorshDeserialize for CustomUrlVector{
    fn deserialize_reader<R: std::io::prelude::Read>(reader: &mut R) -> std:
        // Крок 1 - прочитати довжину вектора
        let len: usize = usize::deserialize_reader(reader)?;
        // Крок 2 - прочитати всі текстові рядки з URL
        let mut out_vec: Vec<Url> = Vec::with_capacity(len);
        // Повторити стільки разів, скільки вектор містить рядків
        for _ in 0..len{
            // Десеріалізувати рядок із формата визначеного в специфікації
            let url_string: String = String::deserialize_reader(reader)?;
            // Звбрати Url із прочитаного рядка
            let url: Url = match Url::parse(input: &url_string){
                Ok(url: Url) => url,
                Err(_) => return std::io::Result::Err(
                    std::io::Error::new(
                        kind: std::io::ErrorKind::InvalidData,
                        error: CustomDeserializeError {}
                    )
                )
            };
            out_vec.push(url);
        };
        return Ok(CustomUrlVector{vector: out_vec});
```

Рис 10. Імплементація десеріалізації для структури “CustomUrlVector”

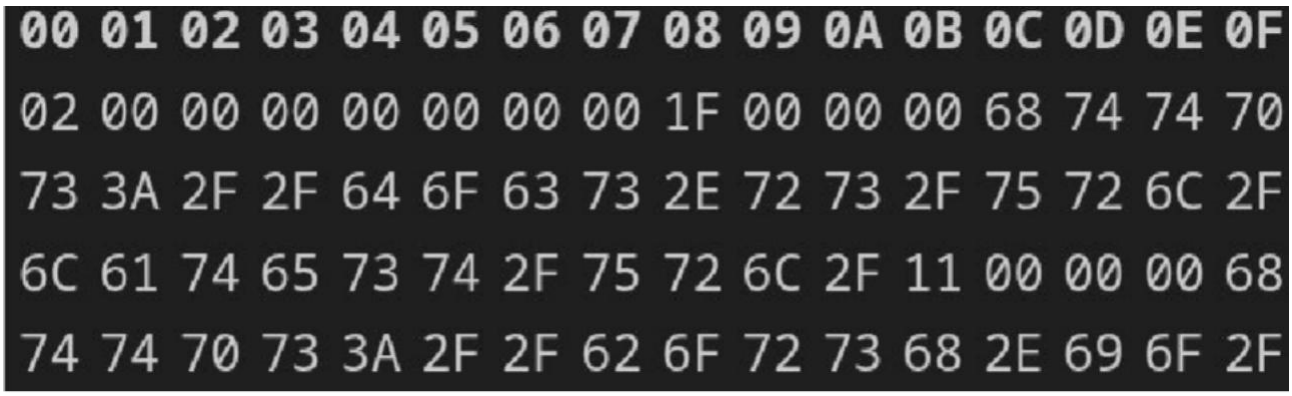


Рис 11. Шістнадцятковий вигляд серіалізованої структури “CustomUrlVector”. Жирні цифри зверху – номери стовбців.

Рис 11 показує серіалізований вигляд структури “CustomUrlVector” що містить 2 URL: "https://docs.rs/url/latest/url/" та "https://borsh.io/". Перші 8 пар це 8-байтове число що представляє кількість елементів у векторі, в даному випадку

2. Байти з 08 першого рядка по 0A третього рядка кодують перший текстовий рядок “https://docs.rs/url/latest/url/”, перші 4 байти з них, “1F 00 00 00”, це число 31 яке представляє кількість букв що записані далі у кодуванні UTF-8. Байти з 0B третього рядка до кінця це другий рядок “https://borsh.io/”, перші 4 байти з них, “11 00 00 00”, це число 17 яке представляє кількість букв другого рядка.

Висновки та перспективи. Бінарний формат бібліотеки “Borsh” складається із послідовно записаних даних із розміром колекції записаним перед ними у 4 байтах, всі дані у ньому записані у такому ж вигляді у якому вони записані у пам’яті комп’ютера, тому для їх десеріалізації достатньо лише скопіювати дані в пам’ять і формально оформити колекції. Текстові формати серіалізації натомість потребують парсинга, а самоописуючі бінарні формати можуть мати більше мета даних що не несуть корисного навантаження, тому несамоописуючий бінарний формат як у бібліотеці “Borsh” має менший розмір та обробляється комп’ютером швидше. Менший розмір та швидша обробка дозволяють економити ресурси комп’ютера що економить електроенергію та дозволяє використовувати менш потужні компоненти.

Список використаних джерел

1. Специфікація бібліотеки “Borsh” за адресою “https://borsh.io/”, 22.05.2026.

2. Документація бінарного формата “Protocol Buffer” за адресою “https://protobuf.dev/overview/”, 22.05.2026.

3. Документація бібліотеки “Borsh” за адресою “https://docs.rs/borsh/latest/borsh/”, 22.05.2026.

4. Документація бібліотеки “Serde” за адресою “https://serde.rs/”, 22.05.2026.

Напрям 4 АЛГОРИТМИ ТА МОДЕЛІ AI/ML

Бажан Тетяна Олександрівна
аспірантка 3 курсу, групи АІПЗ-31
Державний університет інформаційно-комунікаційних технологій
(097)803-34-49
tetiana.olexandrivna@gmail.com

Поперешняк Світлана Володимирівна
доцент, к.ф.-м.н., доцент кафедри Технологій цифрового розвитку
Державний університет інформаційно-комунікаційних технологій, м. Київ
spopereshnyak@gmail.com

ВИКОРИСТАННЯ ГІБРИДНИХ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ІНВЕСТИЦІЙНИХ ПРОЦЕСІВ НА ОСНОВІ BIG DATA

Постановка задачі. У сучасних умовах цифрової трансформації економіки спостерігається стрімке зростання обсягів даних, що формуються у фінансово-інвестиційній сфері, причому ці дані характеризуються не лише значною розмірністю та неоднорідністю, але й наявністю шумів і складною часовою структурою, що суттєво ускладнює їх обробку традиційними статистичними методами [1]. Водночас точність прогнозування інвестиційних процесів залишається критично важливою умовою для прийняття ефективних управлінських рішень як на державному, так і на корпоративному рівнях, оскільки помилки прогнозу можуть призводити до значних економічних втрат.

Існуючі підходи машинного навчання, зокрема регресійні моделі, ансамблеві методи та нейронні мережі, демонструють різну ефективність залежно від структури даних [1], однак використання лише окремих моделей часто не дозволяє одночасно врахувати як нелінійні залежності між змінними [4], так і часову динаміку інвестиційних процесів. Саме тому виникає необхідність розробки гібридних моделей, які поєднують переваги різних підходів до аналізу великих даних і забезпечують більш стійкі та точні результати.

Мета дослідження. Метою дослідження є підвищення точності прогнозування інвестиційних процесів шляхом використання гібридних моделей машинного навчання, які адаптовані до обробки великих економічних даних та здатні враховувати як часові, так і структурні особливості таких даних.

Результати дослідження. У роботі запропоновано підхід до прогнозування інвестиційних процесів, що базується на використанні гібридної моделі машинного навчання, яка поєднує рекурентні нейронні мережі типу LSTM та ансамблеві методи, зокрема CatBoost і Random Forest [3], що дозволяє одночасно враховувати часові залежності та складні нелінійні взаємозв’язки між ознаками [1], [5].

Формально задачу прогнозування можна представити у вигляді передбачення значення показника на майбутній момент часу $t + h$:

$$\hat{y}_{t+h} = f\left(y_t, y_{t-1}, \dots, y_{t-p}, X_t\right),$$

де $\hat{y}_{t+h}$ – прогнозоване значення інвестиційного показника, $y_t, y_{t-1}, \dots, y_{t-p}$ – попередні значення часового ряду (лагові змінні), X_t – вектор зовнішніх факторів (макроекономічні та фінансові показники), $f(\cdot)$ – функція гібридної моделі машинного навчання.

На першому етапі здійснюється попередня обробка великих даних, яка включає очищення, нормалізацію та формування лагових ознак, що дозволяє врахувати часову структуру даних і зменшити вплив випадкових коливань. Джерелами даних виступають макроекономічні показники, серед яких валовий внутрішній продукт, рівень інфляції, процентні ставки, фондові індекси та інші релевантні фактори, що визначають динаміку інвестиційних процесів [5].

Для моделювання часових залежностей використовується LSTM-мережа, яка, на відміну від класичних моделей, здатна враховувати довгострокові залежності у часових рядах, тоді як ансамблеві методи забезпечують ефективне моделювання нелінійних залежностей і взаємодій між ознаками. Гібридизація моделей здійснюється шляхом комбінування прогнозів окремих моделей, що дозволяє зменшити похибку прогнозування та підвищити узагальнюючу здатність моделі.

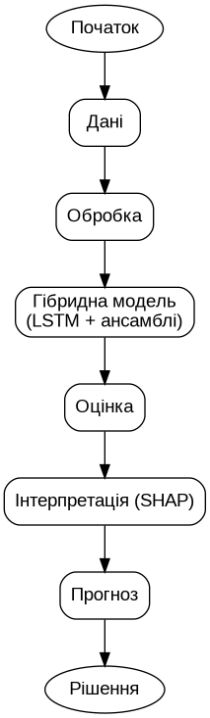


Рис. 1. Структура гібридної моделі прогнозування інвестиційних процесів

Для оцінювання якості прогнозу використовуються метрики регресії, зокрема корінь середньоквадратичної помилки (RMSE), середня абсолютна

помилка (MAE), середньоквадратична помилка (MSE) та коефіцієнт детермінації (R^2), що дозволяє комплексно оцінити точність та надійність моделі.

Додатково, з метою підвищення інтерпретованості результатів, застосовується метод SHAP-аналізу [2], який дозволяє оцінити внесок кожного фактора у формування прогнозу, забезпечуючи прозорість моделі та можливість її використання для підтримки прийняття управлінських рішень.

Експериментальні дослідженя та їх результати свідчать про те, що запропонована гібридна модель демонструє вищу точність прогнозування порівняно з окремими моделями, що підтверджується зниженням значень RMSE та MAE, а також підвищенням узагальнюючої здатності моделі.

Висновки та перспективи. Запропонований підхід до використання гібридних моделей машинного навчання для аналізу великих даних дозволяє суттєво підвищити точність прогнозування інвестиційних процесів в умовах цифрової економіки, оскільки поєднання LSTM-мереж із ансамблевими методами забезпечує врахування як часових, так і нелінійних залежностей у даних [1], [5].

Використання SHAP-аналізу, у свою чергу, підвищує інтерпретованість отриманих результатів, що є важливим для практичного застосування моделей у сфері економічного планування, управління інвестиціями та формування економічної політики.

Перспективою продовження даного досліження є розширення набору факторів, використання більш складних архітектур, зокрема трансформерних моделей для аналізу часових рядів, а також інтеграція розробленої моделі у системи підтримки прийняття рішень, що функціонують на основі технологій Big Data.

Список використаних джерел:

1. Lim, B., & Zohren, S. (2023). Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A*, 381(2240). https://doi.org/10.1098/rsta.2022.0193
2. Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 4765–4774.
3. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31, 6638–6648.
4. Пархоменко, Б. М., & Акименко, А. М. (2024). Використання інформаційних моделей для прогнозування поведінки фінансових показників. *Технічні науки та технології*, (2(36)), 173–180. https://doi.org/10.25140/2411-5363-2024-2(36)-173-180
5. Клименко, В. М. (2024). Застосування інструментів штучного інтелекту для підвищення точності прогнозування продажів у сфері цифрового підприємництва. *Економіка та суспільство*, (68). https://doi.org/10.32782/2524-0072/2024-68-140

Вельгус Христина Олексіївна
студентка 4 курсу, групи ТЦР-41
Державний університет інформаційно-комунікаційних технологій
(096) 581-63-96
st7904178@stud.duikt.edu.ua

АВТОМАТИЗАЦІЯ ПОБУДОВИ ІНДИВІДУАЛЬНИХ ПЛАНІВ НАВЧАННЯ НА ОСНОВІ АЛГОРИТМІВ ШТУЧНОГО ІНТЕЛЕКТУ

Постановка задачі. Сфера розробки програмного забезпечення розвивається дуже стрімко, тому IT-фахівцям потрібно постійно оновлювати свої знання. Більшість сучасних систем навчання пропонують стандартизовані курси, які часто не враховують наявний досвід користувача. Як наслідок, спеціалісти або витрачають час на вивчення того, що вже знають, або ж намагаються опанувати складні технології без достатньої бази. У зв'язку з цим виникає необхідність розробки нових методів для автоматичного аналізу навичок та створення індивідуальних планів навчання.

Мета дослідження. Розробити алгоритм, який генеруватиме персоналізовані дорожні карти розвитку для IT-фахівців за допомогою комбінації структурованої бази даних та можливостей штучного інтелекту.

Результати. Запропонований метод реалізовано у формі програмного модуля, який будує навчальний граф на основі поточної та бажаної посади користувача. Робота алгоритму починається із запиту до зовнішньої мовної моделі через OpenAI API [1], яка формує ієрархічний список необхідних компетенцій.

Основна складність такого підходу полягає в тому, що згенеровані штучним інтелектом результати можуть відрізнятися термінологічно. Наприклад, модель може написати «ReactJS» замість «React.js». Щоб коректно пов'язати ці назви з каталогом матеріалів, який зберігається в системі, було створено модуль інтелектуального зіставлення. Його робота базується на алгоритмі нечіткого пошуку та метриці відстані Левенштейна [2]. Завдяки цьому система вираховує мінімальну кількість операцій, необхідних для того, щоб перетворити згенероване слово на те, що існує в реляційній базі даних PostgreSQL [3]. Встановлений поріг подібності дозволяє алгоритму точно розпізнавати навички.

Загальну структуру роботи алгоритму формування дорожньої карти зображено на рисунку 1.

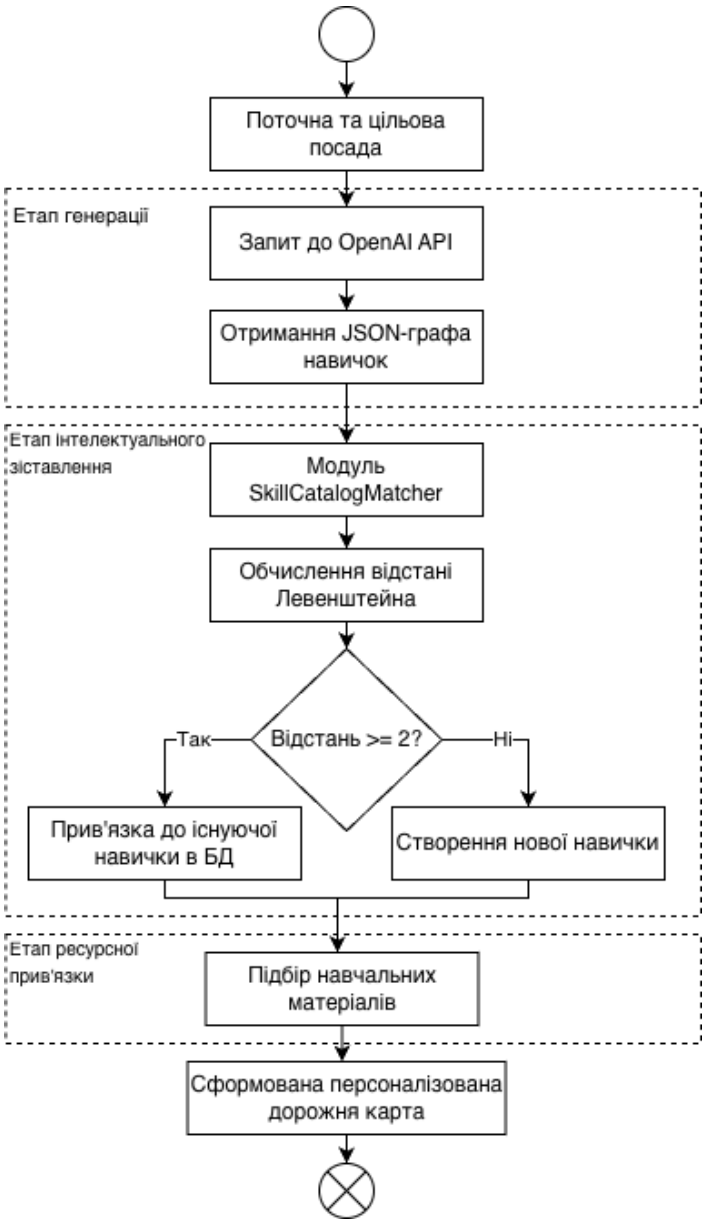


Рис. 1. Блок-схема алгоритму генерації дорожньої карти та зіставлення навичок

Після того як навички розпізнано, система автоматично формує план розвитку. До кожної технології в плані прив'язуються перевірені курси або статті. Точність такого автоматичного зіставлення перевищує 95%.

Висновки. У ході дослідження було успішно вирішено поставлене завдання – розроблено стабільний гібридний алгоритм для побудови індивідуальних навчальних планів. Застосування алгоритму Левенштейна дозволило нівелювати проблему неточних текстових генерацій від штучного інтелекту, уникнути дублів та гарантувати цілісність бази даних. Це рішення повністю автоматизує аналіз компетенцій та дозволяє фахівцям отримувати лише ті матеріали, які дійсно потрібні для їхнього кар’єрного зростання.

Список використаних джерел

1. ВИКОРИСТАННЯ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АНАЛІЗУ ДАНИХ / Т. ВАЙС та ін. *MEASURING AND COMPUTING*

DEVICES IN TECHNOLOGICAL PROCESSES. 2024. № 2. С. 389–394 (дата звернення: 16.04.2026).
2. PostgreSQL 18.3 Documentation. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/index.html (дата звернення: 16.04.2026).
3. Text generation | OpenAI API. *OpenAI Developers*. URL: https://developers.openai.com/api/docs/guides/text (дата звернення: 16.04.2026).

Бандура Артем Ігорович
студент 4 курсу, групи КНД-43
Державний університет інформаційно-комунікаційних технологій
(093) 141-90-54
st6135911@stud.duikt.edu.ua

АЛГОРИТМИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ У ЦИФРОВИХ СЕРВІСАХ

Постановка задачі. У сучасних цифрових сервісах (електронна комерція, стримінгові платформи, соціальні мережі) користувачі стикаються з надлишком інформації. Це ускладнює пошук релевантного контенту та знижує ефективність взаємодії з системами.

Рекомендаційні системи є ключовим інструментом для персоналізації контенту, однак потребують вдосконалення алгоритмів для підвищення точності, швидкодії та адаптивності до змін поведінки користувачів.

Мета дослідження. Метою дослідження є комплексний аналіз сучасних алгоритмів рекомендаційних систем та визначення їх ефективності у цифрових сервісах різного призначення. Особливу увагу приділено дослідженню методів персоналізації контенту, які дозволяють підвищити якість взаємодії користувачів із інформаційними системами.

Результати дослідження. У ході дослідження було проведено детальний аналіз сучасних підходів до побудови рекомендаційних систем, які широко застосовуються у цифрових сервісах, зокрема в електронній комерції, медіаплатформах та соціальних мережах. Встановлено, що ефективність таких систем значною мірою залежить від обраного алгоритмічного підходу, якості вхідних даних та здатності системи адаптуватися до змін поведінки користувачів.

Основними підходами до реалізації рекомендаційних систем є контент-орієнтовані методи, колаборативна фільтрація та гібридні моделі.

Контент-орієнтовані методи базуються на аналізі характеристик об’єктів (наприклад, текстових описів, категорій, тегів) та формуванні профілю користувача. Такий підхід дозволяє рекомендувати об’єкти, подібні до тих, з якими користувач вже взаємодіяв. Перевагою є незалежність від інших користувачів, проте недоліком виступає обмеженість рекомендацій та низька здатність до виявлення нових інтересів.

Колаборативна фільтрація, у свою чергу, ґрунтується на аналізі поведінки великої кількості користувачів. Вона дозволяє виявляти приховані закономірності та формувати рекомендації на основі подібності між користувачами або об’єктами. Незважаючи на високу точність, цей підхід має проблему «холодного старту», коли відсутня достатня кількість даних про нових користувачів або об’єкти.

Гібридні рекомендаційні системи поєднують декілька підходів, що дозволяє мінімізувати їх недоліки та підвищити якість рекомендацій. Зокрема,

можливе комбінування результатів різних алгоритмів або використання одного методу для ініціалізації іншого.

Архітектуру рекомендаційної системи подано на рис. 1



Рис. 1. Архітектура рекомендаційної системи.

Як показано на рисунку, система складається з декількох основних компонентів: користувача, інтерфейсу взаємодії, модулів обробки даних (контент-орієнтованого, колаборативного та гібридного), моделі рекомендацій та блоку формування результату. Така структура дозволяє реалізувати гнучку та масштабовану систему, здатну працювати в умовах великих обсягів даних.

Висновки. У дослідженні розглянуто алгоритми рекомендаційних систем та їх застосування у цифрових сервісах. Встановлено, що найбільш ефективними є гібридні підходи, які поєднують переваги різних методів.

Перспективами подальших досліджень є впровадження адаптивних моделей, що враховують контекст користувача в реальному часі, а також використання методів штучного інтелекту для підвищення точності та інтерпретованості рекомендацій.

Список використаних джерел

1. Ricci F., Rokach L., Shapira B. *Recommender Systems Handbook*. Springer, 2015.

2. Koren Y., Bell R., Volinsky C. *Matrix Factorization Techniques for Recommender Systems*. IEEE Computer, 2009.

3. Aggarwal C. C. *Recommender Systems: The Textbook*. Springer, 2016.

4. Zhang S., Yao L., Sun A., Tay Y. *Deep Learning Based Recommender System: A Survey*. ACM Computing Surveys, 2019

Рудик Артем Всеволодович
студент 4 курсу, групи ТЦР-41
Державний університет інформаційно-комунікаційних технологій
(099) 529-37-02
rudykartem@gmail.com

ОБҐРУНТУВАННЯ ВИБОРУ МОДЕЛЕЙ TTS ДЛЯ ОЗВУЧЕННЯ СПОВІЩЕНЬ НА ОС ANDROID

Постановка задачі. На сьогодні існує багато різноманітних моделей синтезу мовлення. У кожної свої плюси, мінуси та особливості. Вибір моделі – це складне рішення, що потребує чимало часу на вичитку документації. Особливо враховуючи, що вона підбирається під конкретну задачу, в даному випадку озвучення сповіщень на ОС Android. Тому актуальним є аналіз сучасних моделей TTS і обґрунтування вибору того чи іншого рішення, на основі якості озвучення, швидкості, вартості та інших параметрів.

Мета дослідження. Метою дослідження є аналіз сучасних моделей TTS, що підходять для озвучення сповіщень на ОС Android та визначення найкращого рішення серед них.

Результати дослідження. Специфіка сповіщень на ОС Android накладає певні обмеження на озвучення. По-перше, користувач хоче відразу почути повідомлення при отриманні сповіщення, тому озвучення повинно відбуватись з низькою затримкою. По-друге, мобільна ОС обмежена в ресурсах і проведення важких обчислень веде до швидкої розрядки пристрою, тому модель має давати мінімальні навантаження. Наостанок, сповіщення часто можуть містити декілька мов, тому TTS повинен коректно відпрацьовувати мультимовні тексти.

Синтезатори за місцем обробки можна поділити на локальні та хмарні. Локальні відповідно працюють безпосередньо на пристрої, використовуючи його ресурси для синтезу мовлення, тоді як хмарні працюють на віддалених серверах і використовують лише мобільний трафік смартфона для завантаження аудіо.

Серед локальних розглянуті наступні рішення: Google TTS [1], Sherpa-ONNX [2] та RHVoice [3]. Google TTS уже встановлений на більшості пристроїв з ОС Android, має високу якість голосів, підтримує багато мов, мінімально споживає ресурси. Це просте і ефективне рішення, що є монополістом у своїй сфері. RHVoice – це аналог Google TTS з відкритим кодом, за винятком того, що замість нейромереж використовує математику для синтезу мовлення, через що працює швидше, але звучить сильно гірше. Sherpa-ONNX в свою чергу це фреймворк який дозволяє запускати генеративні моделі використовуючи меншу кількість ресурсів. Голос у таких моделей, як правило, дуже якісний і природний. Проте моделі необхідно певним чином встановити на пристрій і з цим виникають труднощі, адже знаходяться в різних частинах інтернету і знайти оптимальне рішення для завантаження неможливо. Доступних моделей небагато й для розширення асортименту необхідно вручну тренувати їх. Sherpa-ONNX базується C++, що створює необхідність використання таких механізмів, як JNI

та NDK, які ускладнюють архітектуру і підтримку коду. Варто зазначити, що всі вище розглянуті моделі є повністю безкоштовними. Проте, жодне з них не ідеальне для роботи з декількома мовами одночасно, Google TTS та RHVoice можуть працювати лише з однією за раз, тоді як Sherpa-ONNX вимагає ручного створення мультимовної моделі, що займає дуже багато часу.

Серед хмарних моделей розглянуті наступні рішення: ElevenLabs [4], Gemini 2.5 Flash TTS [5], Azure AI Speech [6] (див. таблиця 1).

Таблиця 1 – порівняння хмарних TTS

	ElevenLabs	Azure AI Speech	Gemini 2.5 Flash TTS
Якість голосу	Найкраща	Нормальна	Висока
Кастомізація голосу, стилю	Максимальна	Обмежена (складна реалізація)	Висока (через промпти)
Затримка (мс)	> 1 000 мс	~500 мс	~300 мс
Безкоштовний ліміт (символів/місяць)	10 000	500 000	1 000 000
Безкоштовний ліміт (запитів/день)	∞	∞	1500
Вартість за 1 млн символів (грн)	~ 8 000 грн	~ 1 000 грн	~ 200 грн

ElevenLabs [4] – це лідер по якості та кастомізації голосів, проте у якості є ціна – зависока затримка і висока вартість. Azure AI Speech [6], в свою чергу, має нижчу якість голосів, але в нього погано керується стиль мови. Gemini 2.5 Flash TTS [5] по якості голосів знаходиться посередині між попередніми моделями, а стилі у цієї моделі налаштовуються за допомогою попередньо обраних голосів і промпту. Варто зазначити, що на відміну від локальних моделей, усі хмарні синтезатори чудово працюють з багатьма мовами одночасно.

У результаті, найкращими варіантами для озвучення сповіщень на ОС Android є локальний Google TTS [1] та хмарний Gemini 2.5 Flash TTS [5]. Перша модель уже встановлена на більшості пристроїв і пропонує чудовий баланс між якістю голосів, кількістю мов, простотою реалізації та енергоефективністю.

Друга ж модель, пропонує просте налаштування у коді, найнижчу вартість серед конкурентів і хороші голоси.

Висновки та перспективи. У ході дослідження проведено аналіз сучасних моделей для озвучення сповіщень на ОС Android. Встановлено, що оптимальними рішеннями є Google TTS [1] та Gemini 2.5 Flash TTS [5].

Такий висновок є основою до подальших досліджень з метою розробки системи озвучення, що поєднує обробку тексту з моделями TTS у гібридному режимі. Зокрема, доцільно приділити увагу оптимізації роботи з хмарними синтезаторами для максимального зниження затримок за рахунок кешування, використання потокового аудіо та передачі стиснених даних.

Список використаних джерел

1. Background Execution Limits | Android Developers. Android Developers. URL: https://developer.android.com/about/versions/oreo/background (дата звернення: 18.04.2026).

2. Introduction – sherpa 1.3 documentation. GitHub Pages. URL: https://k2-fsa.github.io/sherpa/intro.html (дата звернення: 18.04.2026).

3. RHVoice (Українська) - ArchWiki. ArchWiki. URL: https://wiki.archlinux.org/title/RHVoice_(Українська) (дата звернення: 18.04.2026).

4. Tested 1,800+ voices: google vs azure vs elevenlabs TTS 2026. Free Text to Speech (TTS) Online – 1,800+ AI Voices. TTS For Free. URL: https://ttsforfree.com/en/blogs/google-vs-azure-vs-elevenlabs-tts-comparison (дата звернення: 18.04.2026).

Завацький Владислав Олександрович
аспірант 3 курсу, групи АІСТ-31
Державний університет інформаційно-комунікаційних технологій
(097) 555-14-36
v.zavatskyi@stud.duikt.edu.ua

ПРОБЛЕМА ГЕНЕРАЦІЇ БАГАТОВИМІРНИХ ЧАСОВИХ РЯДІВ ІЗ РОЗПОДІЛЕНИХ ДАТЧИКІВ В УМОВАХ IOT-СЕРЕДОВИЩ

Постановка задачі. Інтелектуальні сенсорні середовища, побудовані на основі інфраструктури Інтернету речей (IoT), забезпечують можливість реалізації широкого спектра застосувань, орієнтованих на обробку великих обсягів даних, зокрема для дистанційного моніторингу, відстеження активності користувачів та формування цифрових двійників.

Разом із тим, сучасні телекомунікаційні мережі стикаються з труднощами обробки значних обсягів даних, що генеруються великою кількістю сенсорних пристроїв. Це призводить до перевантаження мережі, втрат даних та їх пошкодження. У таких умовах актуальною є задача відновлення втрачених даних сенсорів у розподілених IoT-системах.

Мета дослідження. Метою дослідження є аналіз можливості відтворення втрачених багатовимірних часових рядів сенсорних даних на основі корельованих джерел із використанням генеративних моделей глибокого навчання, а також оцінка ефективності синтетичних даних у задачах розпізнавання активності людини.

Результати дослідження. У межах дослідження було змодельовано сценарій дистанційного моніторингу фізіологічних параметрів та активності користувачів за допомогою сенсорів, інтегрованих у мобільні пристрої. Особливу увагу приділено ситуації втрати даних для окремого користувача внаслідок збою каналу зв’язку (рис. 1).

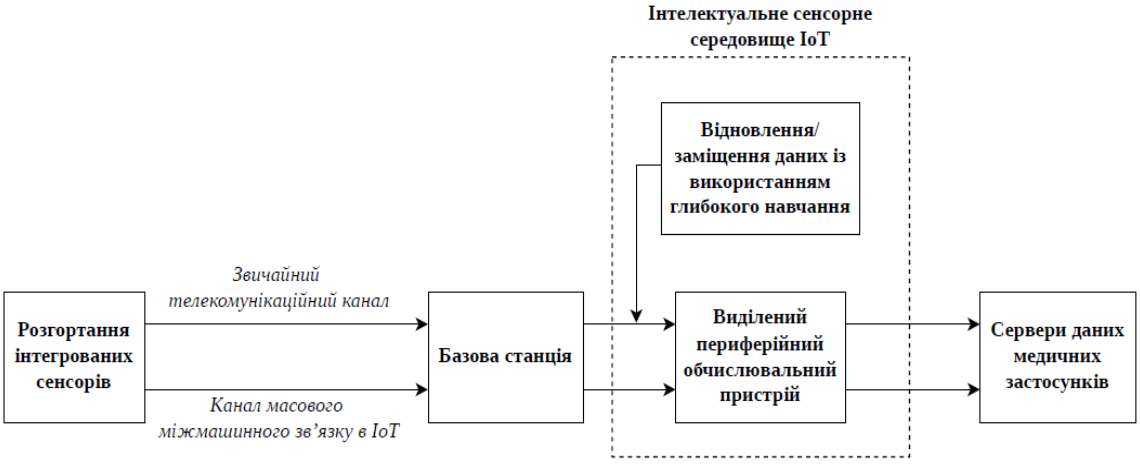


Рис. 1. Сценарій втрати даних у розумному IoT-середовищі.

Для вирішення поставленої задачі використано генеративні змагальні мережі (GAN), які складаються з двох компонентів: генератора та дискримінатора [1]. Генератор формує синтетичні дані, тоді як дискримінатор оцінює їх подібність до реальних.

У дослідженні порівнюються дві моделі генерації часових рядів:

1. Генеративна змагальна мережа для часових рядів (TimeGAN) – поєднує автоенкодер для моделювання часової динаміки та змагальну мережу [2];
2. Генеративна модель глибокого навчання на основі GAN (DoppelGANger) – модель, здатна зберігати довгострокові часові залежності та статистичні характеристики даних [3].

Використано набір даних RealWorld (HAR), який містить інформацію про 8 типів активностей (ходьба, біг, сидіння тощо), отриману з акселерометрів мобільних пристроїв [4].

Особливістю підходу є заміна випадкового шуму у генераторі на реальні дані з найбільш корельованих сенсорних позицій (наприклад, верхня частина руки → передпліччя), що дозволяє підвищити якість синтетичних даних.

Оцінювання якості згенерованих даних здійснюється за допомогою згорткової нейронної мережі (CNN), налаштованої для задачі розпізнавання активності людини [5].

Отримані результати показали:

- для моделі TimeGAN точність класифікації становить ≈90,47% при функції втрат 0,2651;
- для моделі DoppelGANger точність становить ≈85,32% при функції втрат 0,3291.

Встановлено, що хоча синтетичні часові ряди не відтворюють повністю оригінальні сигнали, вони здатні зберігати ключові патерни, необхідні для коректної класифікації активностей. Це підтверджує можливість використання таких даних для заміщення втрачених вимірювань у IoT-системах.

Висновки та перспективи. Проаналізовано задачу відновлення втрачених багатовимірних часових рядів на основі генеративних моделей. Встановлено, що класичні моделі на основі GAN мають обмежену здатність до точного відтворення сигналів, однак є ефективними для збереження загальних характеристик даних. Показано, що синтетичні дані можуть бути використані для навчання моделей розпізнавання активності, навіть за умов часткової втрати інформації.

Перспективами подальших досліджень є застосування більш сучасних підходів, зокрема дифузійних моделей та великих мовних моделей, для підвищення точності генерації часових рядів у IoT-середовищах.

Список використаних джерел

1. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2020). Generative Adversarial Networks.

Communications of the ACM, 63 (11), 139–144. https://doi.org/10.1145/3422622
2. Yoon, J., Jarrett, D., van der Schaar, M. (2019). Time-series generative adversarial networks. Proceedings of the 33rd International Conference on Neural Information Processing Systems. Article No.: 494, Pages 5508 – 5518. https://dl.acm.org/doi/10.5555/3454287.3454781
3. Lin, Z., Jain, A., Wang, C., Fanti, G., Sekar, V. (2020). Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. IMC '20: Proceedings of the ACM Internet Measurement Conference. Pages 464 – 483. https://doi.org/10.1145/3419394.3423643
4. Sztyler, T., Stuckenschmidt, H. (2016). On-body localization of wearable devices: An investigation of position-aware activity recognition. 2016 IEEE International Conference on Pervasive Computing and Communications (PerCom). https://doi.org/10.1109/PERCOM.2016.7456521
5. Ketkar, N., Moolayil, J. (2021). Convolutional Neural Networks. In: Deep Learning with Python. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5364-9_6

Сич Марк Євгенійович
студент групи ТЦР-41,
спеціальність 121 Інженерія програмного забезпечення
Державний університет інформаційно-комунікаційних технологій
+38(050) 714 59 60
st8098426@stud.duikt.edu.ua

ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПЕРСОНАЛІЗАЦІЇ КАР'ЄРНИХ РЕКОМЕНДАЦІЙ В ІТ-СФЕРІ: АЛГОРИТМИ ТА АРХІТЕКТУРНІ РІШЕННЯ

Постановка задачі.

Стрімка цифровізація економіки зумовлює постійне оновлення вимог до ІТ-спеціалістів. Разом із тим існуючі інструменти кар'єрного планування — LinkedIn Learning, Coursera Career Academy, Roadmap.sh — є фрагментованими: жодна з платформ не поєднує пошук вакансій, добір навчальних курсів і генерацію персоналізованих рекомендацій в єдиній системі, адаптованій до українського ринку праці. За даними опитувань, понад 65 % ІТ-фахівців зазнають труднощів під час планування власного кар'єрного шляху, а самостійний пошук релевантних курсів серед 200 000+ пропозицій провідних платформ займає невиправдано багато часу [1]. Таким чином, існує потреба у розробці інтелектуальної системи, що автоматично агрегує дані з різних джерел і формує цілісний кар'єрний план на основі методів штучного інтелекту.

Мета дослідження.

Метою роботи є розробка та дослідження архітектури AI-модуля системи SynapGrid, що забезпечує генерацію персоналізованих рекомендацій щодо кар'єрного розвитку ІТ-спеціаліста шляхом інтеграції великої мовної моделі Cohere Command із результатами агрегації даних про вакансії та навчальні курси.

Результати.

У межах роботи розроблено та реалізовано AI-модуль системи SynapGrid на базі ASP.NET Core 8.0 із застосуванням архітектурного патерну MVC. Алгоритм роботи модуля наведено на рис. 2.

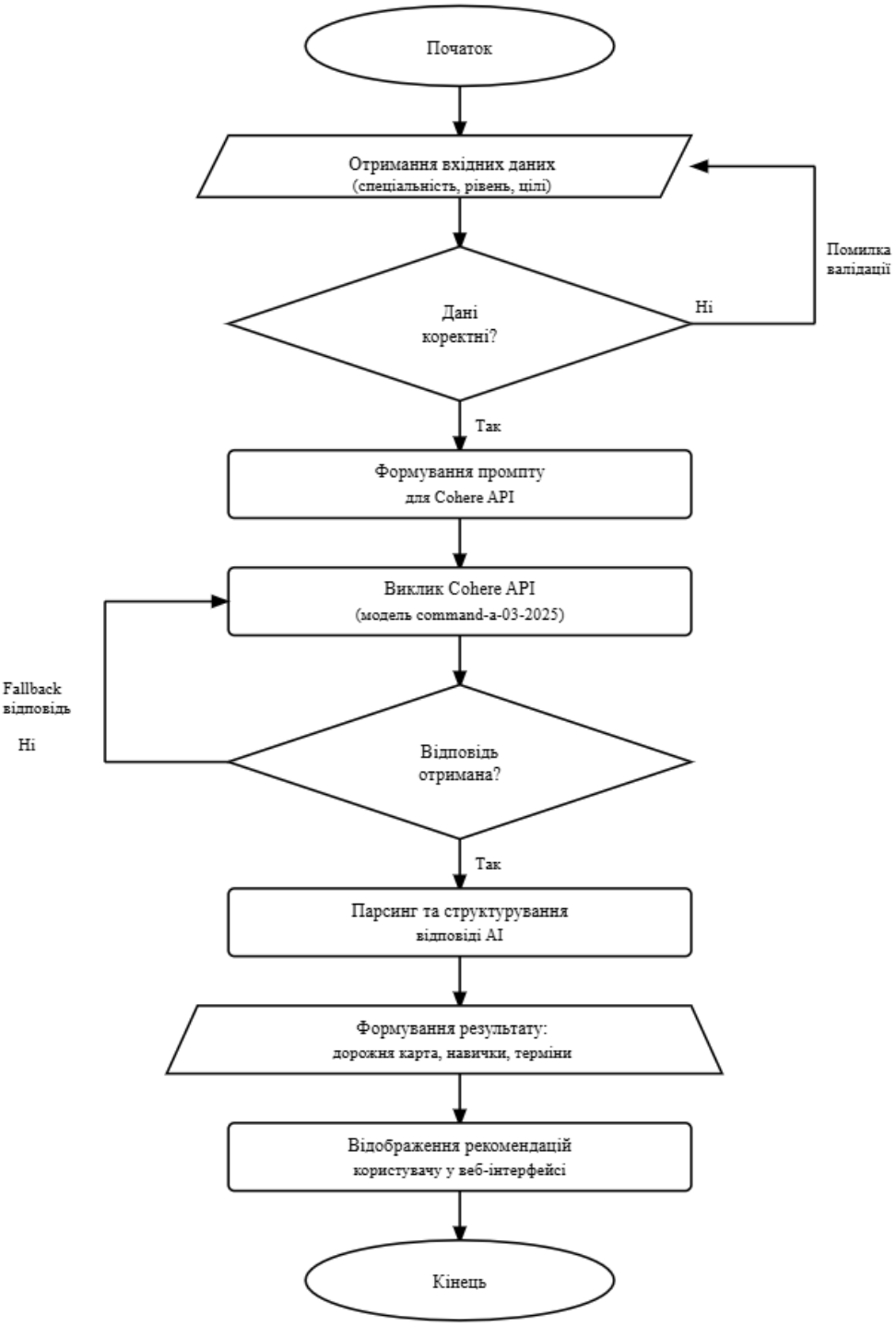


Рис. 2. Алгоритм роботи AI-модуля SynapGrid

Модуль реалізує наступну послідовність операцій. На першому етапі здійснюється валідація вхідних параметрів користувача — обраної спеціальності, поточного рівня та визначених кар'єрних цілей. На другому етапі формується структурований промпт, що містить контекстну інформацію про поточний стан ринку праці, зібрану двома паралельними модулями: скрапером навчальних курсів із ClassCentral.com (50 000+ курсів, Selenium WebDriver 4.25) та агрегатором IT-вакансій з Robota.ua (15 000+ вакансій). На третьому етапі

відбувається звернення до Cohere API з використанням моделі command-a-03-2025 [2].

Усі три модулі (скрапінг курсів, збір вакансій, виклик AI) виконуються паралельно за допомогою механізму Task.WhenAll(), що скорочує загальний час обробки запиту з 45–60 секунд до 15–20 секунд [3]. Для забезпечення відмовостійкості реалізовано механізм fallback-відповідей при недоступності зовнішніх API, що гарантує доступність системи на рівні 95 %.

На виході AI-модуль формує три структуровані блоки: персоналізовану дорожню карту навчання, перелік ключових технічних і soft skills, а також орієнтовні терміни досягнення кар'єрних цілей. Систему протестовано для 18 IT-спеціальностей, коректність генерації рекомендацій підтверджено в усіх тестових сценаріях [4].

Висновки.

Розроблений AI-модуль системи SynapGrid демонструє ефективність застосування великих мовних моделей для вирішення прикладних задач кар'єрного планування в IT-галузі. Ключовою перевагою запропонованого підходу є поєднання реальних даних про ринок праці та освітні ресурси з генеративними можливостями Cohere Command, що дозволяє формувати актуальні та дієві рекомендації. Перспективними напрямами подальшого розвитку є впровадження алгоритмів машинного навчання для аналізу динаміки кар'єрного прогресу користувача та розширення переліку джерел даних для охоплення міжнародного ринку праці.

Список використаних джерел.

1. World Economic Forum. (2023). Future of Jobs Report 2023. https://www.weforum.org/reports/the-future-of-jobs-report-2023
2. Cohere. (2024). Command Models Documentation. https://docs.cohere.com/docs/command-r
3. Microsoft. (2024). Task Parallel Library. https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/task-parallel-library-tpl
4. Selenium Project. (2024). Selenium WebDriver Documentation. https://www.selenium.dev/documentation/webdriver

Волошин Данило Віталійрвич
студент 1 курсу, групи ТЦРМ-51
Державний університет інформаційно-комунікаційних технологій
(098) 659-45-45
danil.voloshin17@gmail.com
Керівник: Поперешняк Світлана Володимирівна
к.ф.-м.н., доцент, доцент кафедри інформатики та програмної інженерії
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
spopereshnyak@gmail.com

АНАЛІЗ ВПЛИВУ НОВИННОГО ФОНУ НА КОРОТКОСТРОКОВІ ЦІНОВІ КОЛИВАННЯ

Постановка задачі. В умовах високої волатильності сучасних фінансових ринків швидкість та точність обробки неструктурованих текстових даних стають критичними для прийняття ефективних торгових рішень. Традиційні підходи до аналізу настроїв, що базуються на рекурентних мережах, часто не здатні коректно інтерпретувати складні семантичні зв’язки та специфічну термінологію фінансових новин. Виникає потреба у нових технологічних рішеннях для високоточної класифікації контексту новин та їх подальшої трансформації у валідовані сигнали. Тому актуальним є аналіз методів застосування архітектур глибокого навчання для оцінки впливу інформаційного фону на короткострокову динаміку активів.

Мета дослідження. Метою дослідження є аналіз ефективності використання архітектури BERT у задачах прогнозування короткострокових коливань котирувань фінансових активів шляхом її адаптації та донавчання на спеціалізованих масивах економічних даних.

Результати дослідження. Аналіз впливу новинного фону засобами архітектури FinBERT дозволяє врахувати специфічну семантику фінансових термінів завдяки попередньому навчанню моделі на галузевих корпусах. Алгоритм обробки неструктурованих даних охоплює етапи токенізації та обчислення контекстуального коефіцієнта настрою (Sentiment Score) у діапазоні від -1 до 1. Логіка запропонованої системи (Рис. 1) ілюструє шлях трансформації тексту у валідовані сигнали через блоки препроцесингу, енкодера BERT та класифікаційного шару.

На рисунку відображено взаємодію модуля аналізу текстів із числовими показниками ринку для формування комплексного вектора ознак. Такий підхід забезпечує високу точність ідентифікації ринкових станів навіть в умовах значного інформаційного шуму [1].

Верифікація моделі підтверджує, що поєднання архітектури BERT із рекурентними мережами LSTM дозволяє ефективно моделювати складні часові залежності та мінімізувати похибки прогнозування короткострокових котирувань

[2]. У ході тестування встановлено зниження кількості хибних сигналів на 14–18% порівняно з класичними методами аналізу настроїв.

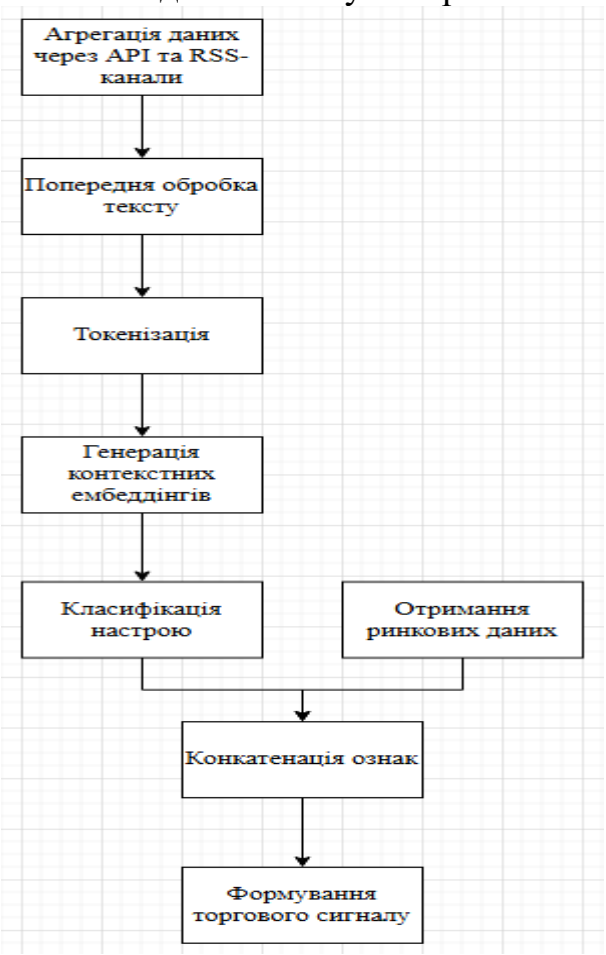


Рис. 1. Блок-схема алгоритму аналізу новинного фону та формування сигналів на основі моделі BERT.

Отримані результати демонструють здатність системи виявляти локальні розвороти тренду протягом першої години після оприлюднення значущих новин. Це доводить доцільність використання трансформерних моделей для трансформації якісної текстової інформації у кількісні параметри алгоритмічних стратегій [3].

Висновки та перспективи. Доведено, що спеціалізована архітектура BERT забезпечує високу точність класифікації фінансових настроїв та ефективно фільтрує інформаційний шум. Впровадження семантичного аналізу в торгові системи дозволяє виявляти приховані ринкові тенденції, суттєво підвищуючи стабільність прогнозів. Подальші дослідження будуть спрямовані на розробку мультимодальних моделей, що поєднують текстову аналітику з глибоким вивченням обсягів торгів та блокчейн-метриками.

Список використаних джерел

1. Halder S. FinBERT-LSTM: Deep Learning based stock price prediction using News Sentiment Analysis. arXiv.org. 2022. DOI:10.48550/arXiv.2211.07392

2. Yadav N. Enhancing Stock Trend Prediction Using BERT-Based Sentiment Analysis and Machine Learning Techniques. International Journal of

Quantitative Research and Modeling. 2024. Vol. 5, No. 1. DOI:10.46336/ijqrm.v5i1.567

3. Ko C-R., Chang H-T. LSTM-based sentiment analysis for stock price forecast. PeerJ Computer Science. 2021. DOI:10.7717/peerj-cs.408

Ніщеменко Дмитро Олександрович,
доктор філософії (PhD)
старший викладач кафедри Технологій цифрового розвитку,
Державний університет інформаційно-комунікаційних технологій
dmytro.nishchemenko@ukr.net

ПЕРЕХІД ВІД ІМПЕРАТИВНОГО ДО ДЕКЛАРАТИВНОГО КЕРУВАННЯ РОЗУМНИМ БУДИНКОМ: КОНЦЕПЦІЯ INTENT-BASED CONTROL ТА ЇЇ ПРАКТИЧНА РЕАЛІЗАЦІЯ

Постановка задачі. Сучасні системи автоматизації розумного будинку здебільшого базуються на правилах виду «Якщо-То» (IF-THEN). Користувач змушений вручну програмувати десятки взаємопов'язаних умов: пороги температури, часові розклади, пріоритети пристроїв тощо. Такий підхід є принципово негнучким: будь-яка зміна звичок або бажань мешканців вимагає переналаштування правил, тоді як статична логіка не здатна адаптуватися до нових контекстних умов (непередбачувана зміна погоди, динамічні тарифи на електроенергію, раптова зміна планів).

Формально задачу можна описати таким чином: наявна система з множиною керованих пристроїв $D = \{d_1, d_2, ..., d_n\}$ та динамічним станом середовища S(t). Традиційний підхід вимагає задання набору правил $R = \{r_1, r_2, ..., r_k\}$, де кожне правило r_i має вигляд «умова → дія». Зі зростанням кількості пристроїв та сценаріїв, розмір R зростає експоненційно, а узгодження правил між собою стає окремою нетривіальною задачею.

Альтернативна парадигма – керування на основі намірів (Intent-Based Control, IBC) – полягає у тому, що користувач задає лише бажані цілі (наміри), наприклад «комфорт» або «економія», а система самостійно обирає керуючі дії для досягнення цих цілей з урахуванням поточного контексту [1].

Мета дослідження. Метою роботи є теоретичне обґрунтування концепції IBC для систем розумного будинку, порівняльний аналіз Imperial та декларативного підходів, а також розробка та практична реалізація архітектури проактивного керування, що функціонує на основі намірів користувача і здатна автономно адаптуватися до його поведінки.

Результати дослідження. У ході дослідження проведено систематичне порівняння двох парадигм управління розумним будинком.

Для практичної реалізації концепції IBC розроблено трирівневу компонентну архітектуру. Перший рівень – взаємодія з середовищем – відповідає за збір та попереднє оброблення сенсорних даних (температура, вологість, присутність, тариф). Другий рівень – обчислювальне ядро – реалізує проактивний цикл «генерація рішень → прогнозування стану → оцінювання корисності» та обирає оптимальну дію. Третій рівень – управління знаннями та адаптація – зберігає контекстуальні наміри у вигляді вагових коефіцієнтів та оновлює їх на основі ручних втручань користувача як неявного зворотного зв'язку [2].

Ключова перевага IBC полягає у навчанні не глобальним, а контекстуально-залежним намірам. Наприклад, вечором у будні система автоматично підвищує пріоритет комфорту (вага wcomfort → 0,9), тоді як вдень у пікові тарифні години – пріоритет економії (wcomfort → 0,7). Це досягається через аналіз ручних коригувань, які користувач здійснює у певних контекстах, схема роботи підходу продемонстрована на Рис. 1.

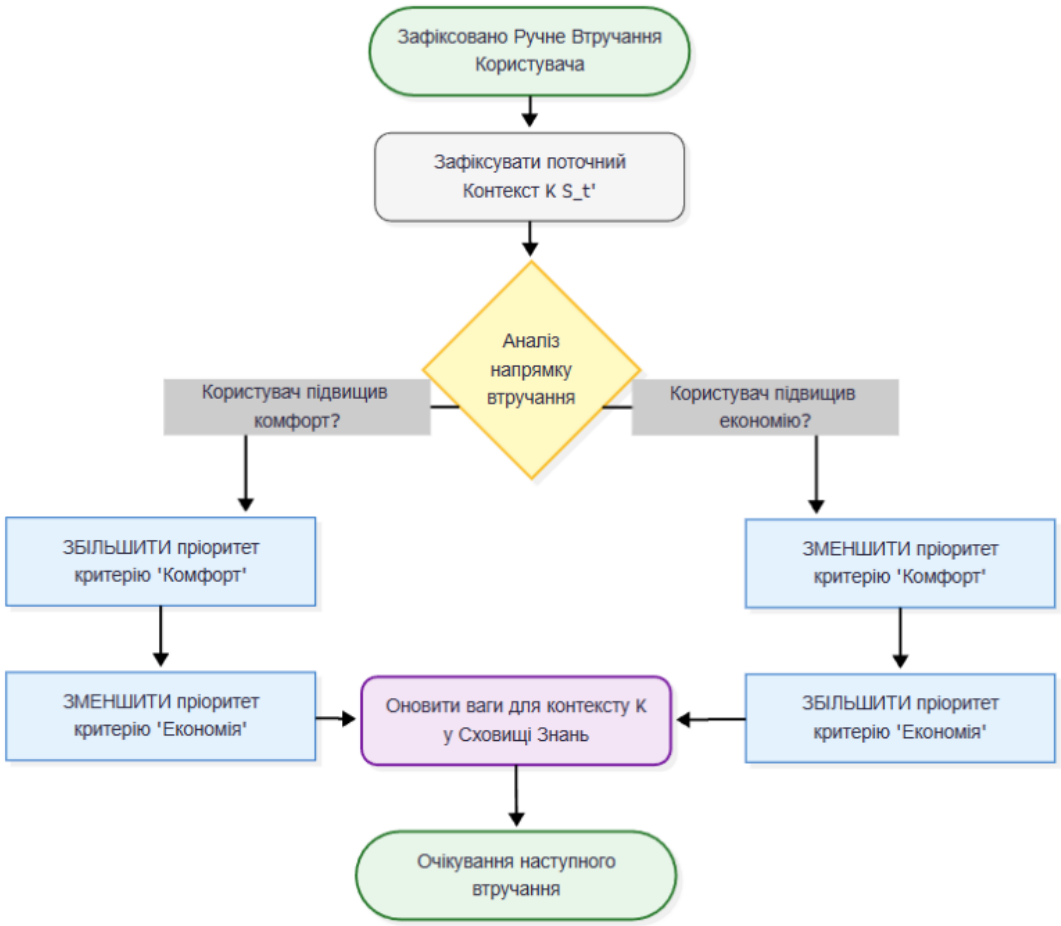


Рис. 1: Блок-схема алгоритму адаптивного навчання на основі втручання

Для валідації концепції проведено 60-денне імітаційне моделювання у середовищі Gymnasium. Порівнювалися три агенти: Rule-Based (жорсткі правила), Advanced Rule-Based (ускладнена логіка з гістерезисом) та Proactive (розроблений IBC-агент). Результати показали, що Proactive-агент забезпечив 41,5% часу перебування у цільовій температурній зоні (20–22°C) проти 25,9% у Rule-Based, а кількість ручних втручань скоротилася на 97%: з 240 до 28 за 60 діб [2]. При цьому операційні витрати виявилися на 25% нижчими порівняно з Advanced Rule-Based, який досягав гіршого рівня комфорту.

Висновки та перспективи. Проведене дослідження доводить принципову перевагу декларативного підходу IBC над традиційним правило-орієнтованим керуванням. Переваги реалізованої архітектури:

1) зниження когнітивного навантаження на користувача, відсутність необхідності програмування правил;

2) висока адаптивність – система самостійно навчається контекстуальним намірам без явного задання;

3) практична ефективність – скорочення ручних втручань на 97% та підвищення рівня комфорту на 60% відносно базової моделі.

Перспективами подальших досліджень є інтеграція мультиагентного підходу для керування декількома зонами будинку та застосування методів глибокого навчання з підкріпленням для прискорення адаптації до нових мешканців.

Список використаних джерел

1. Ніщеменко Д.О., Аронов А.О. Дослідження методик оптимізації параметрів системи керування розумним будинком з використанням IOT. Телекомунікаційні та інформаційні технології, №1, 2025. С. 141–150.

2. Ніщеменко Д.О., Олейніков І.А. Проактивна архітектура керування розумним будинком на основі контекстуальних намірів користувача та багатоцільової оптимізації. Телекомунікаційні та інформаційні технології, №3, 2025. С. 141–149.

3. Shaikh P.H. et al. A review on optimized control systems for building energy and comfort management of smart sustainable buildings. Renewable and Sustainable Energy Reviews, 2014. Vol. 34. P. 409–429.

4. Jia R., Jin B., Jin M. et al. Design automation for smart building systems. Proc. of the IEEE, 2018. Vol. 106(9). P. 1680–1699.

5. Lu J., Sookoor T., Srinivasan V. et al. The smart thermostat: using occupancy sensors to save energy in homes. SenSys'10, 2010. P. 211–224.

Читулян Вадим Олегович
аспірант 2 курсу, групи АКСМ-21
Державний університет інформаційно-комунікаційних технологій
(067) 177-68-05
mag5187403@stud.duikt.edu.ua

ТЕХНОЛОГІЯ ІТЕРАТИВНОГО ПРОЄКТУВАННЯ КОРПОРАТИВНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ З ВБУДОВАНОЮ СИМУЛЯЦІЄЮ НАВАНТАЖЕННЯ

Постановка задачі. У практиці проєктування корпоративних обчислювальних систем тестування навантаження традиційно розглядається як окрема фаза, що виконується після завершення розробки на вже розгорнутій системі. Такий підхід має принциповий недолік: архітектурні проблеми — вузькі місця конфігурації, неоптимальний розподіл ресурсів, недостатня пропускна здатність – виявляються на пізніх стадіях проєкту, коли вартість їх усунення є максимальною.

Альтернативою є підхід design-time testing, де симуляція навантаження виконується безпосередньо в циклі проєктування – на кожній ітерації зміни конфігурації, до початку розробки і розгортання. Проте формалізованої технології, яка б визначала склад сценаріїв навантаження, набір обов'язкових метрик і критерії прийнятності конфігурації за результатами симуляції, у науковій літературі не представлено [1, 2].

Мета дослідження. Метою є розробка технології ітеративного проєктування корпоративних обчислювальних систем, у якій симуляція навантаження є вбудованим елементом кожної ітерації, а метрики продуктивності виступають формальними критеріями прийнятності архітектурного рішення.

Результати дослідження. Запропонована технологія реалізує замкнений ітеративний цикл проєктування, схему якого наведено на рис. 1. Цикл складається з чотирьох повторюваних кроків: розгортання тестового середовища, виконання сценаріїв навантаження, збору та аналізу метрик, прийняття рішення про подальше коригування конфігурації або її фіксацію як прийнятної.

Тестове середовище розгортається локально засобами Docker Compose і відтворює цільову архітектуру корпоративної системи: API-сервер, базу даних і балансувальник навантаження. Використання контейнеризації дозволяє відтворювати середовище детерміновано на кожній ітерації та змінювати конфігурацію компонентів без повторного ручного налаштування.

Сценарії навантаження розробляються мовою JavaScript у середовищі k6 і охоплюють чотири профілі: стале навантаження: базовий рівень активності користувачів; пікове – короткочасне різке зростання запитів; стресове – навантаження до повного насичення системи; поступове зростання – лінійне збільшення кількості virtual users протягом заданого інтервалу. Вхідні

параметри сценарію задаються явно: кількість VU, тривалість тесту, think time між запитами, співвідношення read/write операцій, розмір payload.

Збір метрик виконується автоматично: k6 передає результати в InfluxDB, звідки вони відображаються в реальному часі на дашборді Grafana. Обов'язковий набір метрик включає: p95 і p99 латентність відповіді (мс), throughput (req/s), error rate (%), завантаженість CPU та RAM під навантаженням, час до насичення – момент, коли подальше зростання навантаження не збільшує throughput.

Критерії прийнятності конфігурації задаються явно до початку ітерації і перевіряються автоматично після завершення тесту. Типові граничні значення для корпоративних систем: p95-латентність не перевищує 300 мс, error rate не перевищує 1%, saturation point досягається не раніше ніж при 80% від очікуваного пікового навантаження. Якщо хоча б один критерій не виконується – конфігурація коригується (зміна лімітів ресурсів контейнерів, параметрів пулу з'єднань, налаштувань балансувальника) і цикл повторюється.

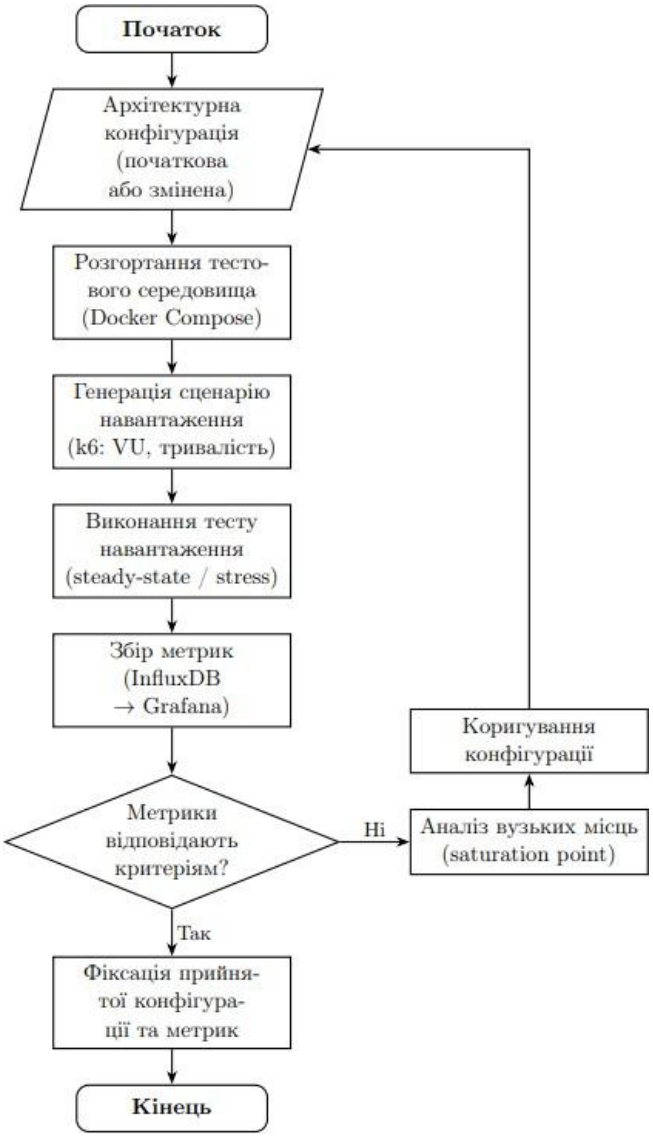


Рис. 1. Схема ітеративного циклу проєктування з вбудованим тестуванням Навантаження

Ефективність технології вимірюється порівнянням двох груп конфігурацій: тих, що проєктувалися без симуляції (еталонна група), і тих, що пройшли через запропонований ітеративний цикл. Фіксується кількість архітектурних проблем, виявлених на етапі проєктування, та кількість проблем, що проявилися після розгортання. Додатковою метрикою є економія витрат на усунення проблем: відповідно до [3], вартість виправлення архітектурної помилки на етапі розгортання в середньому у 5–10 разів перевищує вартість її усунення на етапі проєктування.

Висновки та перспективи. Запропонована технологія ітеративного проєктування з вбудованою симуляцією навантаження усуває ключовий недолік існуючих підходів – відсутність формального зворотного зв'язку між метриками продуктивності і рішеннями на рівні архітектурного проєктування. Використання k6 для генерації чотирьох профілів навантаження, InfluxDB і Grafana для збору метрик та явно заданих граничних значень як критеріїв прийнятності дозволяє виявляти вузькі місця конфігурації до початку розробки і суттєво скорочувати витрати на їх усунення.

Перспективами подальших досліджень є автоматизація генерації k6-сценаріїв за описом поведінки користувача засобами LLM, інтеграція технології з ML-класифікатором архітектурних рішень та дослідження застосовності підходу для мікросервісних архітектур із сотнями незалежних сервісів.

Список використаних джерел

1. Mehla A., Dahiya N., Singh K. AI-Driven Cloud Infrastructure Provisioning: Leveraging ML, Observability, and IaC for Autonomous Cloud Operations. Proc. ETNCC. 2025. DOI: 10.1109/etncc66224.2025.11299663.

2. Christadoss J., Mani K. AI-Based Automated Load Testing and Resource Scaling in Cloud Environments Using Self-Learning Agents. JAIGS. 2024. Vol. 6, No. 1. DOI: 10.60087/jaigs.v6i1.401.

3. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Addison-Wesley, 2021. 624 p.

4. Gaddam R. R. AI-Augmented Dynamic Performance Engineering: A Hybrid Platform Architecture. Journal of Information Systems Engineering and Management. 2025. Vol. 10, No. 63s. DOI: 10.52783/jisem.v10i63s.13804.

5. Grant O. H. Resource-Aware Cloud Orchestration Using ML-Driven Predictive Scaling for High-Demand IT Applications. 2025. DOI: 10.36676/irt.v11.i4.1716.

Криворучко Віталій Федорович,
студент групи АІПЗ-31,
спеціальність 121 Інженерія програмного забезпечення,
Державного університету інформаційно-комунікаційних технологій
Науковий керівник: Садовенко Володимир Сергійович,
кандидат фізико-математичних наук, доцент,
професор кафедри Інженерії програмного забезпечення автоматизованих систем
Державного університету інформаційно-комунікаційних технологій
380637915751
hamandes@gmail.com

ФЕДЕРАТИВНЕ МАШИННЕ НАВЧАННЯ ДЛЯ ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ МУЛЬТИХМАРНИХ ПЛАТФОРМ

Ключові слова: мультихмарні платформи, федеративне навчання, AI/ML, QoS, SLA, телеметрія, децентралізовані дані, FedAvg.

Постановка задачі. Оцінювання ефективності хмарних платформ традиційно ґрунтується на централізованому збиранні телеметрії з усіх сервісів та доменів розгортання. У мультихмарних середовищах такий підхід має низку обмежень: частина даних не може передаватися між організаціями або регіонами через політики безпеки, телеметрія формується в неоднорідних форматах, а самі навантаження мають неідентичний розподіл у різних хмарних провайдерів. Унаслідок цього єдина централізована модель оцінювання QoS часто або потребує надмірної передачі даних, або втрачає точність через доменну неоднорідність. Актуальним є перехід до такого підходу, за якого моделі навчаються локально на кожному хмарному вузлі або в окремому домені, а на глобальний рівень передаються не сирі дані, а лише параметри або оновлення моделі. Це дає змогу поєднати оцінювання якості сервісу з дотриманням вимог конфіденційності та мінімізації міжхмарного трафіку.

Мета дослідження. Метою є розроблення узагальненої схеми оцінювання ефективності мультихмарної платформи на основі федеративного машинного навчання, яка забезпечує прогнозування ключових показників QoS і ресурсної ефективності без централізованого перенесення повних масивів телеметричних даних між хмарними доменами.

Результати. Запропоновано федеративну схему, у якій кожний хмарний домен k формує локальну вибірку D_k з телеметричних показників CPU, RAM, I/O, мережевої активності, P95 latency, доступності та вартості використання ресурсів. Для локального навчання використовується модель $f_k(w)$, параметризована вектором ваг w, а локальна цільова функція визначається як $F_k(w) = (1/n_k)\cdot\Sigma\, l(w; x_i, y_i)$, де n_k - кількість спостережень у домені, x_i - вектор ознак, y_i - цільовий показник ефективності, l - функція втрат. Глобальна задача навчання формулюється як мінімізація зваженої суми локальних функцій: $F(w) = \Sigma\, (n_k / N)\cdot F_k(w)$, де $N = \Sigma\, n_k$. Агрегація моделі виконується за схемою

FedAvg: $w^{(t+1)} = \Sigma (n_k / N) \cdot w_k^{(t+1)}$, де $w_k^{(t+1)}$ - оновлені локальні ваги після t-го раунду навчання.

Для кількісного оцінювання ефективності запропоновано інтегральний мультихмарний показник FME (Federated Multicloud Efficiency):

$$FME = \Sigma \rho_k \cdot MEI_k,$$

де ρ_k - ваговий коефіцієнт домену, що враховує його частку в сумарному навантаженні, а локальний індекс ефективності визначається як $MEI_k = \alpha \cdot Q_k + \beta \cdot U_k + \gamma \cdot A_k - \delta \cdot C_k$. Тут $Q_k = 1 - L_{p95,k} / SLA_L$ характеризує якість обслуговування відносно допустимого SLA-порогу, U_k - нормований рівень утилізації ресурсів, A_k - доступність сервісу, C_k - нормована вартість. Вагові коефіцієнти α, β, γ, δ задаються експертно відповідно до пріоритетів платформи, причому $\alpha + \beta + \gamma + \delta = 1$.

На відміну від попередніх підходів, орієнтованих на централізоване навчання, запропонована схема дозволяє зберігати телеметрію в межах окремих хмарних доменів та узгоджувати лише оновлення моделі. Це особливо важливо для мультихмарних середовищ, де домени можуть відрізнятися за продуктивністю, політиками доступу до даних і профілем навантаження. На основі аналізу попередніх досліджень з прогнозування навантаження, QoS та структурних аномалій встановлено, що для таких середовищ найбільш суттєвими викликами є non-IID-розподіл даних, асиметрія обчислювальних ресурсів і комунікаційні затримки між доменами. Тому до федеративної схеми доцільно додавати механізм відбору клієнтів і періодичну переоцінку ваг ρ_k, що зменшує вплив повільних або статистично нестабільних доменів на глобальну модель.

У межах аналітичного сценарію оцінювання розглянуто три домени: приватну хмару підприємства, публічну хмару для веб-сервісів і окремий аналітичний домен для пакетної обробки даних. Для кожного домену локально оцінюються RMSE прогнозу навантаження та F1 для виявлення ризикових станів SLA, після чого глобальна модель порівнюється з централізованим підходом за такими критеріями: точність прогнозування, стійкість до неоднорідності даних, обсяг міжхмарного обміну та придатність до розгортання в умовах обмеженого доступу до сирої телеметрії. Проведений аналіз показав, що федеративний підхід є доцільним насамперед у тих сценаріях, де передача первинних даних між доменами є небажаною або надмірно дорогою. При цьому компроміс між точністю та приватністю визначається частотою глобальної агрегації, кількістю локальних епох і складом доменів, що беруть участь у раунді навчання.

Отримані результати узагальнення дозволяють сформулювати такі практичні положення. По-перше, федеративне навчання може бути використане не тільки для прикладних prediction-задач, а й як основа для побудови інтегральної системи оцінювання ефективності мультихмарної платформи. По-друге, локальне навчання на доменних даних підвищує адаптивність моделей до специфіки конкретного провайдера або сегмента навантаження. По-третє, використання FME як агрегованого показника спрощує порівняння стану

мультихмарної платформи в часі та підтримує прийняття рішень щодо масштабування, перерозподілу навантаження та оптимізації витрат.

Висновки. У роботі запропоновано підхід до оцінювання ефективності мультихмарних платформ на основі федеративного машинного навчання, який поєднує локальне навчання на телеметрії окремих доменів і глобальну агрегацію параметрів моделі. Наукова новизна полягає у перенесенні ідей federated learning із задач конфіденційного навчання в площину оцінювання ефективності хмарної інфраструктури. Практична цінність підходу полягає у можливості прогнозувати QoS та ресурсну ефективність без централізованого переміщення первинних даних. Перспективою подальших досліджень є поєднання федеративної схеми з online learning, uncertainty-aware оцінюванням SLA-ризику та автоматичним керуванням масштабуванням у мультихмарних середовищах.

Список використаних джерел.

1. McMahan H. B., Moore E., Ramage D., Hampson S., Arcas B. A. y. Communication-Efficient Learning of Deep Networks from Decentralized Data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. PMLR. 2017. Vol. 54. P. 1273–1282.
2. Kairouz P., McMahan H. B., Avent B. et al. Advances and Open Problems in Federated Learning. Foundations and Trends in Machine Learning. 2021. Vol. 14, No. 1–2. P. 1–210.
3. Bonawitz K., Eichner H., Grieskamp W. et al. Towards Federated Learning at Scale: System Design. Proceedings of Machine Learning and Systems. 2019. Vol. 1. P. 374–388.
4. Nishio T., Yonetani R. Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. IEEE International Conference on Communications Workshops. 2019. P. 1–7.
5. Dean J., Barroso L. A. The Tail at Scale. Communications of the ACM. 2013. Vol. 56, No. 2. P. 74–80.

Гайдукевич Владислав Олегович
аспірант 4 курсу, групи КН-22
Інститут програмних систем НАН України, м. Київ
(096) 233-53-33
gaidukevichvlad@gmail.com

МЕТОДИ ТА АЛГОРИТМИ НАВІГАЦІЇ МОБІЛЬНИХ РОБОТІВ У ДИНАМІЧНИХ СЕРЕДОВИЩАХ

Постановка задачі. Постановка задачі. Розвиток сучасних автономних мобільних роботів (АМР) та безпілотних літальних апаратів (БПЛА) вимагає створення надійних систем навігації, здатних функціонувати в умовах апріорі невизначеного та змінного середовища. Одним із ключових завдань для таких систем є забезпечення безпечного руху до цільової точки з одночасним уникненням як статичних (стіни, меблі, елементи інфраструктури), так і динамічних перешкод (люди, інші роботи, рухомі об'єкти). Класичні детерміновані методи планування траєкторій часто мають високу обчислювальну складність, що обмежує їх застосування в системах реального часу. Тому актуальним є аналіз підходів, які забезпечують високу швидкість реакції та предиктивну адаптивність АМР до швидких змін навколишнього простору.

Мета дослідження. Метою дослідження є теоретичний аналіз, узагальнення та класифікація сучасних алгоритмічних рішень для керування рухом мобільних роботів на основі циклічного опитування сенсорних систем, визначення їхніх відносних переваг, недоліків та подальших перспектив розвитку.

Результати дослідження. Сучасні автономні мобільні роботи мають високий ступінь самоконтролю та автономності. Для цього вони оснащуються різноманітними датчиками (лідарами, ультразвуковими сенсорами, стереокамерами) та потужними вбудованими обчислювальними системами, які збирають та інтерпретують інформацію про навколишнє середовище в режимі реального часу. Завдяки цьому АМР здатні швидко коригувати поточний напрямок руху, адаптувати лінійну та кутову швидкості, а також уникати потенційних зіткнень із будь-якими типами перешкод.

Методи планування шляху АМР відіграють головну роль в ефективній та безпечній навігації. Ці методи традиційно поділяють на глобальні та локальні. Глобальне планування передбачає побудову оптимального маршруту на основі апріорної, заздалегідь відомої карти робочої зони. Натомість локальне планування працює в умовах, коли поточне середовище навколо робота є частково або повністю невідомим чи динамічним. У такому випадку алгоритми адаптуються до виявлених об'єктів безпосередньо під час руху на основі

поточного потоку сенсорних даних. Базову вертикальну архітектуру логіки прийняття рішень такою системою навігації продемонстровано на рис. 1.

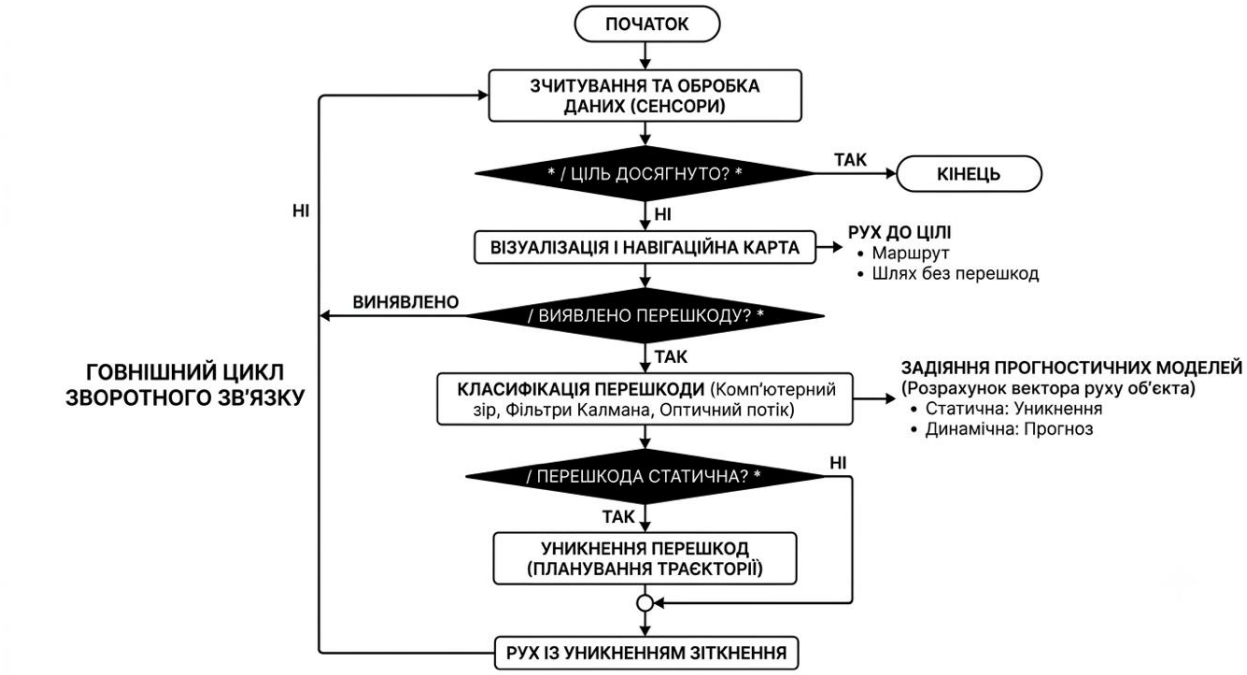


Рис. 1. Блок-схема алгоритму навігації та покрового уникнення перешкод.

Алгоритм локального уникнення перешкод вимагає від АМР миттєвого корегування попередньо встановленого глобального маршруту, безпечного обходу об'єкта та максимально швидкого повернення на вихідну траєкторію. Сюди відносять декілька фундаментальних та інтелектуальних підходів:

- **Метод штучного потенційного поля (APF).** Робот моделюється як точка, що рухається у віртуальному силовому полі. Цільова точка генерує силу притягування, а всі виявлені перешкоди — сили відштовхування. Робот рухається в напрямку результуючого вектора цих сил. Метод є дуже простим для обчислення та демонструє чудову продуктивність у реальному часі. Однак його суттєвим недоліком є ризик потрапляння у локальні мінімуми (коли сили притягування та відштовхування врівноважуються до нуля) або неможливість досягнення цілі, якщо вона оточена перешкодами.
- **Гістограма векторного поля (VFH).** Цей алгоритм будує двовимірну сітку навколо робота на основі аналізу просторових даних сенсорів. На основі цієї сітки формується одновимірна полярна гістограма щільності перешкод. Алгоритм враховує фізичні габарити робота, його поточну швидкість та мінімальний радіус повороту для вибору найбільш вільного сектора для руху. Це гарантує високу плавність траєкторії без різких стрибків.
- **Підхід динамічного вікна (DWA).** Метод безпосередньо інтегрує динамічні та кінематичні обмеження приводу робота у процес уникнення колізій. Алгоритм обмежує простір пошуку лінійної та кутової швидкостей лише тими значеннями, які робот встигне фізично розвинути або скинути

протягом наступного часового інтервалу (динамічне вікно). Для кожної пари швидкостей моделюється траєкторія, яка оцінюється за критеріями безпеки, швидкості руху до цілі та віддаленості від перешкод. Метод ефективний, але чутливий до складності геометрії середовища.

- **Метод швидкісних перешкод (VO).** Цей підхід орієнтований суто на динамічні середовища. Він обчислює область у просторі швидкостей робота, яка за умови незмінності векторів руху призведе до зіткнення з рухомим об'єктом у майбутньому. Робот обирає таку швидкість, яка лежить поза межами цієї небезпечної конусоподібної зони. VO дозволяє ефективно працювати з об'єктами, що швидко рухаються навколо, враховуючи їхній вектор швидкості в реальному часі.
- **Алгоритми навчання з підкріпленням (RL).** Сучасний підхід штучного інтелекту, за якого робот (агент) навчається оптимальній поведінці шляхом спроб і помилок, взаємодіючи з невідомим середовищем та отримуючи цифрову винагороду або штраф. Deep RL моделі дозволяють системі самостійно знаходити закономірності руху та уникати складних динамічних перешкод без використання заздалегідь побудованих карт і мають меншу залежність від точності роботи окремих датчиків. Найбільшу стабільність демонструють гібридні методи (наприклад, інтеграція Q-навчання з потенційними полями APF), які забезпечують одночасну гладкість, мінімальну довжину та максимальну безпеку траєкторії.

Висновки та перспективи. У дослідженні розглянуто базові та інтелектуальні методи уникнення перешкод мобільними роботами. Проведений огляд підтверджує, що поєднання традиційних реактивних алгоритмів із предиктивними моделями машинного навчання є найбільш перспективним напрямом для роботи у складних нестаціонарних середовищах.

Перспективою подальших досліджень є оптимізація обчислювальних витрат моделей глибокого RL, вивчення взаємодії БПЛА і роботів із групою динамічних перешкод, що рухаються за складними, випадковими траєкторіями, та тестування алгоритмів у децентралізованих архітектурах крайових обчислень (Edge Computing).

Список використаних джерел

1. Hart F., Waltz M., Okhrin O. Two-step dynamic obstacle avoidance. *Knowledge-Based Systems*. 2024. Vol. 302. P. 112402.
2. Truong X.-T., Dinh H. T., Nguyen C. D. An efficient navigation framework for autonomous mobile robots in dynamic environments using learning algorithms. *J. Comput. Sci. Cybern.* 2018. Vol. 33, № 2. pp. 107–118.
3. Orozco-Rosas U., Picos K., Pantrigo J. J., Montemayor A. S., Cuesta-Infante A. Mobile Robot Path Planning Using a QAPF Learning Algorithm for Known and Unknown Environments. *IEEE Access*. 2022. № 10. P. 84648–84663.

Напрям 5 ІНФОРМАЦІЙНІ СИСТЕМИ ТА МЕРЕЖІ

Сокульський Олег Євгенович
кандидат технічних наук, старший викладач
Військовий інститут телекомунікацій та інформатизації імені Героїв Крут
mortimer@ukr.net
Москаленко Наталія Володимирівна
асистент
Київський національний університет імені Тараса Шевченка
moskalenkonv@ukr.net
Топольськов Євгеній Олександрович
кандидат технічних наук, доцент
Київський національний університет імені Тараса Шевченка
y.topolskov@knu.ua

СУБД NewSQL ТА ІНФОРМАЦІЙНІ СИСТЕМИ

Постановка задачі. Сучасний ринок СУБД дуже різноманітний та насичений. Розробники проектів мають можливість вибору не тільки моделі даних СУБД, які визначають структуру, збереження та обробку інформації, але й підібрати СУБД з відповідними функціональними можливостями для рішення максимально ефективно своїх задач[1].

Мета дослідження. Метою дослідження є розгляд та оцінка застосування СУБД NewSQL в інформаційних системах.

Результати дослідження. На рівні з класичною реляційною моделлю даних, принципи якої були сформульовані ще в 70-х роках минулого сторіччя, на ринку присутні та стрімко розвиваються сучасні реляційні СУБД, які отримали назву NewSQL та широко використовуються для обробки даних у великих обсягах[2].

Основні переваги та особливості СУБД NewSQL.

- Горизонтальне масштабування(додавання нових вузлів), що значно підвищує продуктивність.
- ACID-транзакції(збереження цілісності даних при виконанні транзакцій).
- SQL-сумісність(використання звичної мови запитів SQL).
- Автоматичний шардинг(дані розподіляються між вузлами без адміністратора).
- Розподілена архітектура.

Найбільш популярні представники:

- CockroachDB. Система з відкритим кодом призначена для роботи з великими об’ємами даних в глобальних розподілених системах та хмарних середовищах. Розподілена архітектура забезпечує легке масштабування, стійка до відмов обладнання завдяки застосуванню

алгоритму Raft, підтримка транзакціїй ACID гарантує цілісність даних. Великою перевагою є SQL-сумісність, що дозволяє міграцію з класичних СУБД.

- Google Spanner. Повністю керована Google СУБД, Поєднує масштабованість NoSQL та строгу підтримка ACID. SQL-сумісна. Технологія TrueTime, забезпечує глобальну узгодженість даних. Забезпечує автоматичну синхронну реплікацію як в межах одного регіону, так і між різними регіонами, гарантує захист даних навіть у разі виходу з ладу цілого дата-центру. З успіхом застосовуються в фінансових системах, глобальних веб-застосунках та сервісах, великих e-commerce платформах та інструментах аналітики.
- TiDB. Розподілена система, підтримує автоматичне секціонування (партиціонування), горизонтальне масштабування та стійкість до збоїв. Архітектура HTAP дозволяє виконувати як швидкі транзакції(OLTP), так і складні аналітичні запити (OLAP) в одній базі даних у реальному часі, підтримує алгоритм консенсусу Raft та транзакції ACID. Використовується для проектів, які потребують високої швидкості запису, значного обсягу даних та необхідності миттєво аналізувати ці дані.
- VoltDB. Резидентна(in-memory) СУДБ з відкритим кодом без розподілених ресурсів, завдяки чому забезпечується надзвичайно висока швидкість запису та читання. Архітектура Shared-Nothing розподіляє дані між вузлами та усуває конфлікти блокувань. Підтримує транзакції ACID, та горизонтальне масштабування, SQL-сумісна. Успішно використовується в системах реального часу, де потрібно термінове прийняття рішень, наприклад, телекомунікації (5G), системи інтернет речей (IoT), та аналіз даних в потоці.

Висновки та перспективи. Використання СУБД NewSQL в сучасних інформаційних системах дозволяє не тільки використати переваги класичних реляційних СУДБ, але й поєднати їх з перевагами інших моделей СУБД, підвищуючи тим самим ефективність та продуктивність обробки інформації.

Список використаних джерел

1. Демиденко М.А., Введення в сучасні бази даних. Навчальний посібник. – Дніпро.НТУ: Дніпровська політехніка, 2020, 38 с.

2. .Єсін В.І, Вілігура В.В. Основні категорії NEWSQL баз даних та їх особливості – [Електронний ресурс] –Режим доступу: https://scispace.com/pdf/the-main-categories-of-newsql-databases-and-their-features-3ipv8y9f.pdf

Москаленко Наталія Володимирівна
асистент
Київський національний університет імені Тараса Шевченка
moskalenkonv@ukr.net
Сокульський Олег Євгенович
кандидат технічних наук, старший викладач
Військовий інститут телекомунікацій та інформатизації імені Героїв Крут
mortimer@ukr.net

DATA MINING ТА СУБД

Постановка задачі. Кількість інформації, яку потрібно збирати, зберігати та аналізувати в сучасному світі стрімко зростає. Зростає не тільки її об'єм та швидкість передачі даних, зростає також і кількість різновидів інформації. В багатьох випадках це неструктуровані дані, які потребують великого пакетного аналізу або аналізу в режимі реального часу.

Мета дослідження. Особливе місце в процесі розробки інформаційних систем займають технології інтелектуального аналізу даних(Data Mining), які забезпечують весь процес роботи з даними, починаючи від збору даних до складного аналізу даних. На початковому етапі, етапі добування даних, технології дозволяють досліджувати та перетворювати великі об'єми сірих даних на структуровану інформацію, виявляти приховані закономірності, кореляції аномалії та корисні знання, на основі яких можна приймати точні рішення або ефективно прогнозувати майбутні події.

Сучасні засоби Data Mining вирішують задачі класифікації, кластеризації, асоціації, моделювання та прогнозування в алгоритмах рішення яких застосовуються дерева рішень, нейронні мережі, машинне навчання, а також спеціалізоване програмне забезпечення.

Результати дослідження. Існує безліч безкоштовних програм для інтелектуального аналізу даних з відкритим кодом. Наприклад, програма GATE(General Architecture for Text Engineering) для обробки природної мови, базується на мові програмування Java. Мова Python лежить в основі програмного забезпечення Orange для аналізу даних і машинного навчання.

Зазвичай сучасні інформаційні системи для роботи з даними використовують СУБД з різними моделями даних та різною функціональністю. Для підвищення ефективності збереження та обробки інформації сучасні СУБД в своєму складі мають різні алгоритми інтелектуального аналізу даних[1].

СУБД PostgreSQL та MySQL в основному використовується для первинної обробки даних, тобто дані на етапі збору можуть бути опрацьовані з використанням відповідних програм аналізу даних, але є СУБД, які мають відповідні модулі для інтелектуального аналізу даних різного рівня складності, тобто вони вбудовані в СУБД, що значно підвищує надійність та швидкість обробки та аналізу даних.

СУБД Microsoft SQL Server має службу Microsoft SQL Server Analysis Services (SSAS), яка призначена для аналітичної обробки великих даних (OLAP), бізнес аналітики(B1) та підтримки прийняття рішень. Підтримує багатовимірні куби та табличні моделі, має дві мови запитів[2]:

- DAX (Data Analysis Expressions) для табличних моделей,
- MDX (Multidimensional Expressions) для багатовимірних моделей.

Основні характеристики SSAS:

- Технології in-memory для табличних моделей та агрегація в багатовимірних кубах забезпечують високу швидкість виконання запитів.
- Перетворення складних структур баз даних в зрозумілі бізнес-терміни значно спрощує роботу коористувачів.
- Гнучка безпека за рахунок управління доступом до даних .

СУБД Oracle має службу Oracle Data Mining (ODM), ключовою особливістю якої є обробка даних прямо там, де вони зберігаються, що значно підвищує безпеку та швидкість. Служба є частиною СУБД, що дозволяє легко впроваджувати аналітичні моделі у бізнес-додатки.

Технології ODM дозволяють на основі аналізу даних автоматично створювати моделі реальних додатків. Oracle Data Mining створює прогнозуючі та дескрипторні моделі.

Прогнозуючі моделі:

- класифікація;
- регресія;
- пошук суттєвих атрибутів.

Дескрипторні моделі:

- кластеризація;
- пошук ассоціацій;
- виділення ознак.

Зараз Oracle Data Mining є частиною ширшого пакета Oracle Machine Learning (OML), який також підтримує мови Python та R для роботи з даними всередині бази.

Висновки та перспективи. Наявність методів та засобів інтелектуального аналізу даних в конкретній СУБД дозволяє підвищити надійність та якість збереження та обробки інформації, а також робить процес прийняття рішень або прогнозування більш ефективним.

Список використаних джерел

1. Лупан І.В. Інтелектуальний аналіз даних Data Mining: Навчально-методичний посібник. – Кропивницький, ФОП Піскова М. А., 2022. – 112 с.
2. Сайт компанії Microsoft – [Електронний ресурс] –Режим доступу https://learn.microsoft.com/ru-ru/analysis-services/?view=sql-analysis-services-2025

Зуб Людмила Миколаївна
Студентка 1 курсу, групи ТСДМ-51
Державний університет інформаційно-комунікаційних технологій
(098) 286-73-78
st01235874@stud.duikt.edu.ua

АРХІТЕКТУРА ТА АЛГОРИТМ ФУНКЦІОНУВАННЯ РІВНЯ IOT-ПРИСТРОЇВ У СИСТЕМІ РОЗУМНОГО КАМПУСУ

Постановка задачі. У сучасних системах розумного кампусу нижній рівень, представлений IoT-пристроями, відіграє ключову роль у формуванні інформаційної бази для подальшої аналітики та управління. Значна кількість сенсорних вузлів генерує потоки даних у реальному часі, що потребує ефективної організації їх збору, обробки та передачі. При цьому важливими є обмежені обчислювальні ресурси мікроконтролерів, енергоспоживання та нестабільність бездротових каналів зв'язку.

Існуючі підходи до побудови IoT-рівня часто не враховують необхідність попередньої обробки даних безпосередньо на пристроях, що призводить до перевантаження мережі та підвищення затримок у системі. Крім того, відсутність уніфікованого алгоритму взаємодії між сенсорами та вищими рівнями ускладнює масштабування та інтеграцію нових пристроїв. Це зумовлює необхідність розробки ефективного алгоритму функціонування IoT-вузлів із підтримкою сучасних протоколів обміну даними.

Таким чином, науково-практичне завдання полягає у створенні алгоритму роботи IoT-пристроїв на базі ESP32, який забезпечить надійний збір, валідацію, фільтрацію та передачу сенсорних даних, а також підтримку обробки керуючих команд у режимі реального часу. Запропонований підхід має забезпечити зменшення навантаження на мережу, підвищення точності даних і ефективну інтеграцію з edge-рівнем системи розумного кампусу.

Мета дослідження. Мета дослідження полягає у розробці алгоритму функціонування рівня IoT-пристроїв на базі ESP32 для ефективного збору, обробки та передачі даних у системі розумного кампусу.

Результати дослідження. Першим етапом функціонування IoT-пристрою є ініціалізація системи, яка включає підключення до бездротової мережі Wi-Fi, встановлення з'єднання з MQTT-брокером та налаштування сенсорних модулів. На цьому етапі також виконується підписка на топіки для отримання керуючих команд. Коректна ініціалізація забезпечує стабільну роботу пристрою в мережі та його інтеграцію в загальну систему.

Другим етапом є періодичний збір даних із сенсорів. Зчитуються параметри температури, вологості, освітлення та інші показники залежно від конфігурації системи. Отримані дані проходять перевірку валідності, що дозволяє відсіяти некоректні значення, а також фільтрацію (наприклад, за допомогою ковзного середнього), що підвищує точність вимірювань.

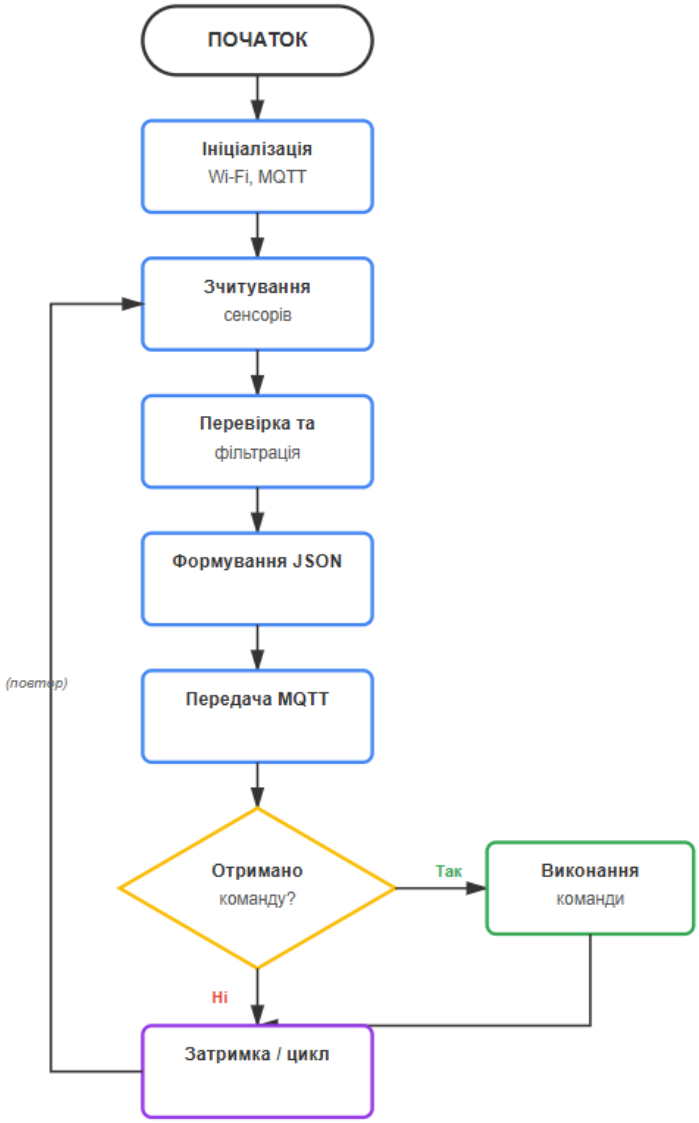


Рис. 1. Блок-схема алгоритму

На третьому етапі виконується формування повідомлення у форматі JSON та передача даних через MQTT-протокол до edge-рівня системи. Такий підхід забезпечує легку інтеграцію, масштабованість та асинхронну взаємодію між пристроями. Використання MQTT дозволяє мінімізувати навантаження на мережу та забезпечити ефективну передачу даних навіть при великій кількості вузлів.

Четвертий етап передбачає обробку керуючих команд, що надходять із верхніх рівнів системи. Пристрій аналізує отримане повідомлення та виконує відповідні дії, такі як керування освітленням, замками чи іншими актуаторами. Таким чином реалізується зворотний зв’язок і забезпечується повноцінна інтеграція IoT-рівня в систему управління.

Висновки та перспективи. У результаті проведеного дослідження було розроблено алгоритм функціонування рівня IoT-пристроїв на базі ESP32, який забезпечує ефективний збір, попередню обробку та передачу сенсорних даних у системі розумного кампусу. Запропонований підхід враховує обмежені

обчислювальні ресурси пристроїв, забезпечує валідацію та фільтрацію даних, а також підтримує двосторонню взаємодію з вищими рівнями системи через MQTT-протокол. Це дозволяє підвищити надійність роботи сенсорної мережі та зменшити обсяг передаваного трафіку.

Отримані результати підтверджують доцільність перенесення частини обчислень безпосередньо на рівень IoT-пристроїв, що сприяє зниженню навантаження на мережеву інфраструктуру та edge-вузли. Реалізація локальної обробки даних дозволяє підвищити швидкість реагування системи та забезпечити базову автономність у разі втрати зв’язку з вищими рівнями. Таким чином, запропонований алгоритм є ефективним рішенням для побудови масштабованих та відмовостійких кіберфізичних систем.

Перспективи подальших досліджень полягають у впровадженні елементів штучного інтелекту безпосередньо на рівні IoT-пристроїв (TinyML), оптимізації енергоспоживання сенсорних вузлів, а також підвищенні рівня кібербезпеки шляхом використання сучасних методів шифрування та аутентифікації. Крім того, актуальним напрямом є інтеграція цифрових двійників та самонавчальних механізмів, що дозволить підвищити адаптивність системи до змін у середовищі функціонування.

Список використаних джерел

1. Shantyr, A., Zinchenko, O., Storchak, K., Bondarchuk, A., & Pepa, Y. (2025). Prediction of quality software quality indicators with applied modifications of integrated gradiates methods. Informatyka, Automatyka, Pomiary. 15(2), 139-146.
2. Кузьмінський, А. Р., Носков, В. І., & Бондарчук, А. П. (2025). Гібридний підхід поєднання NB-IOT з LORA. Телекомунікаційні та інформаційні технології, (2), 134-148.
3. Polin, K., Yigitcanlar, T., Limb, M., & Washington, T. (2023). The making of smart campus: A review and conceptual framework. Buildings, 13(4), 891.
4. Бондарчук, А. П., & Глушак, О. М. (2025). Предиктивне управління оновленнями програмного забезпечення в інтернеті речей. Зв'язок, (5), 13-17.
5. Imbar, R. V., Supangkat, S. H., & Langi, A. Z. (2020, November). Smart campus model: a literature review. In 2020 International Conference on ICT for Smart Society (ICISS) (pp. 1-7). IEEE.

Закаблук Едуард Євгенович
студент 4 курсу, групи КНД-43
Державного університету
інформаційно-комунікаційних технологій
(063)-47-47-924
vyshnivskyi.viktor@gmail.com
Науковий керівник: Вишнівський Віктор Вікторович,
доктор технічних наук, професор, завідувач кафедри Комп'ютерних наук Державного університету інформаційно комунікаційних технологій, м. Київ

АНАЛІЗ ДОЦІЛЬНОСТІ ВПРОВАДЖЕННЯ ПРИВАТНИХ ХМАРНИХ РІШЕНЬ ДЛЯ МАЛОГО ТА СЕРЕДНЬОГО БІЗНЕСУ

Постановка задачі.

Цифровізація малого та середнього бізнесу (SMB) впритул поставила підприємців перед питанням, яке ще десять років тому здавалося суто технічним: де фізично зберігати дані компанії та кому довіряти доступ до них. За зовнішньою простотою онлайн сервісів приховується низка системних ризиків: дані підприємства фізично розміщуються на серверах третьої сторони, правовий режим доступу до них регулюється не українським, а іноземним законодавством, а щомісячні платежі з часом перевищують вартість власної інфраструктури. Питання не в тому, чи є хмарні технології корисними самі по собі, а в тому, яка модель хмари – публічна чи приватна – справді відповідає інтересам бізнесу.

Мета дослідження.

Мета роботи – комплексно оцінити доцільність розгортання приватної хмарної інфраструктури для підприємств сегменту SMB з урахуванням трьох ключових вимірів: правового захисту комерційної таємниці, довгострокової економічної ефективності та технічної здійсненності на базі сучасного бюджетного обладнання.

Результати дослідження.

Згідно зі ст. 505 ЦК України, захист комерційної таємниці покладається на власника шляхом вжиття реальних заходів конфіденційності. Використання публічних хмарних серверів суперечить цій нормі, оскільки фізичний контроль над даними переходить до третьої сторони. Жодна угода SLA не гарантує повної ізоляції даних від персоналу провайдера чи державних структур країни перебування дата-центру. Таким чином, повне дотримання правового режиму таємниці можливе лише за умови використання власної серверної інфраструктури.

Приватна хмара, побудована на власному обладнанні підприємства, зберігає всі дані в периметрі організації. Адміністратор самостійно визначає, хто, коли і з якого пристрою отримує доступ до корпоративних ресурсів. Дослідження, присвячене реалізації сервера Samba з шифрованим каналом OpenVPN на базі одноплатного комп'ютера Raspberry Pi, підтвердило, що навіть бюджетна апаратна платформа здатна забезпечити криптографічну стійкість,

достатню для корпоративного середовища SMB. Це зміщує периметр безпеки з умовних меж угоди SLA до реального фізичного контролю над носіями інформації.

Аналіз сукупної вартості володіння (TCO) хмарними сервісами демонструє ефект, який прийнято називати «хмарною інфляцією»: щорічне підвищення тарифів провайдерами в середньому на 5-15% і накопичувальні витрати протягом 3-5 років нерідко перевищують одноразові капітальні витрати на розгортання власної інфраструктури [3]. Для команди з 10-30 осіб, що щомісячно сплачує за корпоративні ліцензії Google Workspace або Dropbox Business, сукупні видатки за чотири роки стають цілком порівнянними з вартістю базового серверного рішення, витратами на налаштування та підтримку.

Модель операційних витрат (Opex), приваблива на старті, поступово трансформується на тягар: підприємство не накопичує жодного матеріального активу, а будь-яке скорочення бюджету автоматично загрожує втратою доступу до власних даних. Перехід до капітальної моделі (Capex) через придбання власного серверного рішення дозволяє зафіксувати витрати, амортизувати обладнання і з часом суттєво знизити операційне навантаження на IT-бюджет. Цифрова трансформація підприємства, описана у контексті управлінського аспекту хмарних технологій для SMB, підкреслює, що збалансований вибір між Opex і Capex є стратегічним рішенням, а не лише технічним.

Впровадження приватної хмари не потребує ні великих серверних залів, ні штату спеціалістів. Сучасне програмне забезпечення – Nextcloud для файлового сховища, Samba для сумісності з Windows-середовищем, WireGuard або OpenVPN для захищеного віддаленого доступу, встановлюється на стандартному сервері або навіть на одноплатному комп'ютері й забезпечує функціональність, що покриває більшість потреб SMB: зберігання файлів, спільна робота з документами, синхронізація, резервне копіювання. Методика впровадження хмарних рішень у бухгалтерському обліку, а також практики хмарної аналітики даних підтверджують: перехід на власну інфраструктуру є керованим процесом, що не виходить за межі компетенцій внутрішнього IT-фахівця або залученого підрядника.

Висновки та перспективи

Приватна хмарна інфраструктура для підприємств сегменту SMB є не лише технічно здійсненним, а й стратегічно обґрунтованим рішенням. Вона забезпечує фізичний і логічний контроль над комерційною таємницею відповідно до вимог чинного законодавства України, унеможливлює несанкціонований доступ третіх осіб і усуває залежність від умов угод іноземних провайдерів. В економічному вимірі перехід від щомісячної підписки до власної інфраструктури в горизонті 3–5 років дозволяє суттєво скоротити сукупні витрати та перетворити статтю видатків на матеріальний актив підприємства. Подальші дослідження доцільно спрямувати на розробку типових архітектурних шаблонів приватних хмарних рішень з урахуванням різних масштабів і галузевої специфіки SMB.

Список використаних джерел

1. Городиський М. П., Поліщук І. Р., Якимцева Ю. В. Методика розробки та використання засобів хмарних технологій в обліку. 2021. URL: http://ema.ztu.edu.ua/article/download/236850/235416
2. Капіца Ю. М. Захист прав на комерційну таємницю та ноу-хау в Україні у світлі імплементації Директиви (ЄС) 2016/943 та практики застосування. Інформація і право. 2021. № 4(39). С. 70–84. URL: https://ipr.nas.gov.ua/?smd_process_download=1&download_id=2347
3. Kodakandla S. A Total Cost of Ownership Model for Cloud Computing Infrastructure. ResearchGate. 2023. DOI: 10.13140/RG.2.2.36423.68005. URL: https://www.researchgate.net/publication/367072943
4. Гуржій А. М., Мельник О. Г. Цифрова трансформація управління підприємством: хмарні технології та аналітика даних. 2022. URL: https://reposit.nupp.edu.ua/bitstream/PoltNTU/20698/1/18.pdf
5. Implementation of Samba Server Using OpenVPN Based on Single Board Computer (SBC) for Private Cloud Storage. ResearchGate. URL: https://www.researchgate.net/publication/386597125

Чепур Софія Олексіївна
студентка 4 курсу, групи ПП-41
Київський національний університет імені Тараса Шевченка
(068) 636-97-12
chepursofia27@gmail.com

РОЗРОБКА ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНТЕРНЕТ-МАГАЗИНУ ВИРОБІВ НАРОДНИХ ПРОМИСЛІВ ІЗ ВИКОРИСТАННЯМ СУЧАСНИХ ВЕБ-ТЕХНОЛОГІЙ

Постановка задач: Електронна комерція є важливим інструментом підтримки бізнесу особливо якщо ми говоримо про невеликі локалькі та середні бізнеси що становлять 85% всього бізнес-сектору України. Рішення що вже існують, такі як маркетплейси або соціальні мережі не можуть забезпечити повний контроль над процесом продажу продукції, обробкою замовлень та управління асортиментом це і зумовлює необхідність створення спеціалізованої інформаційної системи інтернет-магазину. Це зумовлює необхідність розробки спеціалізованих інформаційних систем, орієнтованих на підтримку процесів електронної комерції для виробів народних промислів. Ключовим завданням у даному контексті є створення інформаційної системи інтернет-магазину, яка забезпечить ефективну обробку та зберігання даних, підтримку бізнес-логіки продажів, зручну взаємодію з користувачем, а також надійне адміністрування контенту. Важливими вимогами до таких систем є забезпечення високої продуктивності, масштабованості, безпеки обробки даних та адаптивності до змін у поведінці користувачів.Таким чином, актуальним є дослідження та розробка веб-орієнтованої інформаційної системи інтернет-магазину виробів народних промислів із використанням сучасних технологій, зокрема фреймворку Django, що дозволяє реалізувати структурований, безпечний та масштабований програмний продукт.

Метою дослідження є розробка та програмна реалізація інформаційної системи інтернет-магазину виробів народних промислів на основі сучасних веб-технологій із використанням фреймворку Django, а також дослідження архітектурних і технологічних підходів до побудови систем електронної комерції.

Результати дослідження. Розроблена інформаційна система інтернет-магазину виробів народних промислів є веб-орієнтованим програмним продуктом, що реалізує повний цикл електронної комерції та забезпечує ефективну взаємодію між користувачами і адміністратором. Система побудована з використанням фреймворку Django, що дозволяє досягти високого рівня структурованості, масштабованості та безпеки програмного забезпечення. Інтернет-магазин оснащений функціональними модулями для збору, обробки та відображення інформації про товари, категорії, користувачів та замовлення. Система підтримує навігацію по каталогу, пошук продукції, формування кошика та оформлення замовлень, що забезпечує зручний користувацький досвід та відповідає сучасним вимогам до веб-додатків. Архітектура системи реалізована

відповідно до шаблону Model–Template–View (MTV),що забезпечує розділення рівнів представлення, бізнес-логіки та роботи з даними. Модель відповідає за опис структури бази даних і взаємодію з нею через ORM, представлення обробляють HTTP-запити та формують відповіді, а шаблони забезпечують відображення даних у зручному для користувача вигляді. Такий підхід сприяє підвищенню підтримуваності системи та спрощує її масштабування. Взаємодія між клієнтською та серверною частинами реалізована з використанням HTTP-протоколу та принципів REST. Запити до сервера маршрутизуються за допомогою URL-конфігурації Django, що дозволяє чітко організувати структуру обробки запитів. Це забезпечує можливість подальшого розширення системи, зокрема створення REST API для інтеграції з мобільними додатками або сторонніми сервісами. Для роботи з базою даних використано Django ORM, що дозволяє абстрагуватися від низькорівневих SQL-запитів і реалізувати операції створення, читання, оновлення та видалення даних у вигляді об'єктно-орієнтованих моделей. Основними сутностями системи є користувачі, товари, категорії та замовлення, між якими реалізовано логічні зв'язки, що забезпечують цілісність даних. Важливим компонентом системи є адміністративна панель, реалізована за допомогою Django Admin, яка надає інтерфейс для управління контентом без необхідності внесення змін у програмний код. Адміністратор може виконувати операції додавання, редагування та видалення товарів, керувати категоріями, обробляти замовлення та контролювати активність користувачів. Система безпеки базується на вбудованих механізмах Django, що включають хешування паролів, захист від CSRF-атак, перевірку прав доступу та валідацію даних. Додатково використано middleware-компоненти для обробки HTTP-запитів на рівні сервера, що дозволяє реалізувати контроль сесій, автентифікацію та логування дій користувачів. З метою підвищення продуктивності системи передбачено використання механізмів кешування, що дозволяють зменшити навантаження на сервер при повторному запиті однакових даних. Це особливо актуально для відображення каталогу товарів та популярних сторінок. Користувацький інтерфейс реалізовано із застосуванням HTML, CSS та JavaScript, що забезпечує адаптивність і коректне відображення на різних пристроях. Інтерфейс побудовано з урахуванням принципів юзабіліті, що спрощує взаємодію користувача із
системою.Таким чином, розроблена інформаційна система є гнучким, масштабованим та функціонально повним рішенням для організації інтернет-магазину виробів народних промислів, яке може бути використане як основа для подальшого розвитку та інтеграції з іншими цифровими сервісами.

Висновки та перспективи. У дослідженні розглянуто підходи до розробки інформаційних систем електронної комерції та особливості створення інтернет-магазину виробів народних промислів із використанням сучасних веб-технологій. Проведений аналіз існуючих рішень дозволив визначити їх обмеження та обґрунтувати доцільність створення власної системи, орієнтованої на повний контроль бізнес-процесів. У результаті було реалізовано веб-орієнтовану інформаційну систему, яка поєднує ефективну обробку даних,

зручний користувацький інтерфейс та надійні механізми адміністрування. Використання сучасних технологій та архітектурних підходів забезпечує гнучкість, масштабованість і безпечність системи.

Перспективою подальших досліджень є вдосконалення функціональних можливостей системи шляхом інтеграції з зовнішніми сервісами, розширення аналітичних інструментів та впровадження інтелектуальних механізмів персоналізації. Зокрема, актуальним є поєднання класичних підходів до розробки веб-додатків із сучасними методами обробки даних, що дозволить підвищити ефективність взаємодії з користувачами та адаптивність системи до змін у середовищі електронної комерції.

Список використаних джерел

1. Holovaty A., Kaplan-Moss J. The Django Book. Apress. 300 с.
2. Mele A. Django 3 By Example. Packt Publishing. 450 с.
3. Oecd. OECD SME and Entrepreneurship Outlook.. Paris: OECD Publishing,. 2021.

Черкесов Ілля Валерійович,
студент групи ПД-44,
спеціальність 121 Інженерія програмного забезпечення,
Державного університету інформаційно-комунікаційних технологій,
(+371) 263-04-682
ilya.cherkesov@gmail.com
Науковий керівник: Гаманюк Ігор Михайлович,
старший викладач кафедри Інженерії програмного забезпечення,
Державного університету інформаційно-комунікаційних технологій

ПОБУДОВА МОДЕЛІ ПРЕЦЕДЕНТІВ ДЛЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ФЕРМЕРСЬКОГО ГОСПОДАРСТВА

Постановка задачі. Інформаційні системи фермерського господарства мають забезпечувати ведення обліку тварин, збереження відомостей про працівників, фіксацію подій, перегляд довідників і підтримку адміністративних функцій. У таких системах помилки на етапі формування вимог призводять до невідповідності між реальними процесами ферми та реалізованою структурою даних і користувацького інтерфейсу. Одним із доцільних підходів до формалізації функціональних вимог є побудова моделі прецедентів, яка дозволяє визначити основних акторів, набір їхніх дій та межі взаємодії з програмним продуктом.

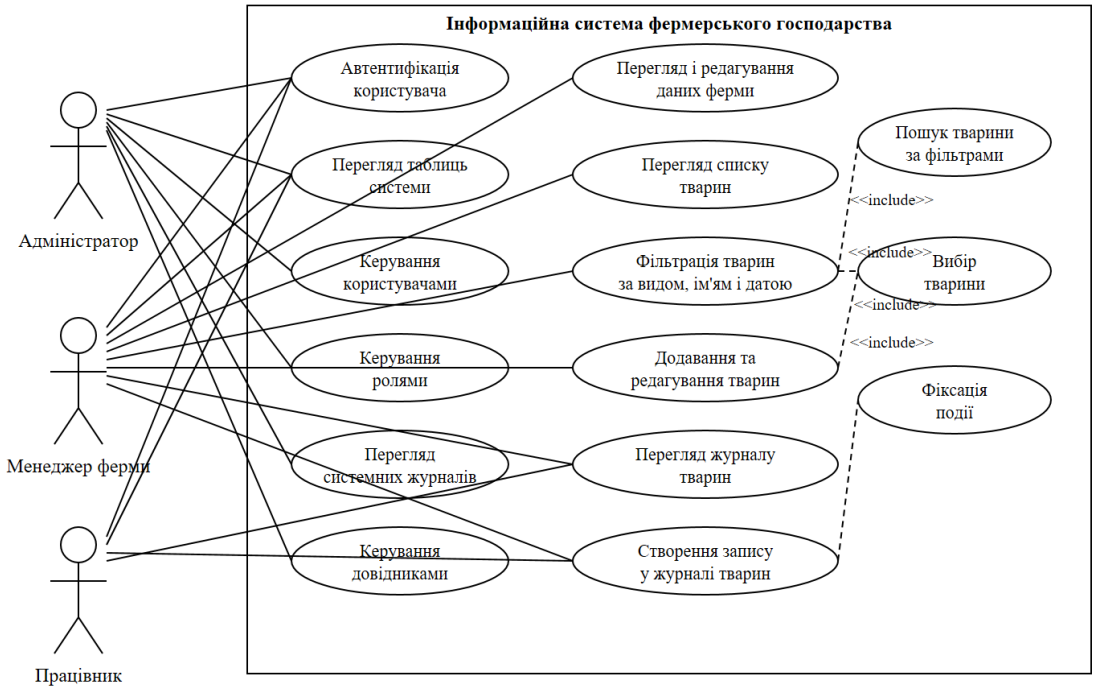
Мета дослідження. Метою дослідження є побудова моделі прецедентів для інформаційної системи фермерського господарства; визначення основних акторів системи, їхніх функцій та взаємозв’язків; формування набору ключових сценаріїв використання, які можуть бути покладені в основу подальшого проєктування структури даних, логіки роботи та інтерфейсу інформаційної системи.

Результати дослідження. У межах дослідження встановлено, що основними акторами інформаційної системи фермерського господарства є адміністратор, менеджер ферми та працівник. Для них визначено базові прецеденти, серед яких: автентифікація користувача, перегляд і редагування таблиць, керування довідниками, робота з даними про тварин, фільтрація тварин за видом, ім’ям і датою народження, а також ведення журналу подій щодо тварин. Виділені прецеденти дають змогу впорядкувати функціональні вимоги до системи та встановити зв’язок між ролями користувачів, функціями застосунку й необхідними сутностями предметної області.

Побудована модель прецедентів дозволяє формалізувати взаємодію користувачів із системою не лише на рівні загального опису функціональності, а й на рівні конкретних сценаріїв роботи. Зокрема, для менеджера ферми суттєвими є сценарії перегляду списку тварин, застосування фільтрів, внесення змін до картки тварини та перегляду журналу записів. Для працівника важливим є сценарій створення нового запису в журналі тварин із фіксацією виконавця, дати та змісту події. Для адміністратора визначальними є сценарії керування користувачами, ролями та контроль доступу до системних розділів.

Практичне значення одержаного результату полягає в тому, що модель прецедентів може бути використана як основа для подальшого проєктування інформаційної системи фермерського господарства. Вона дозволяє узгодити структуру інтерфейсу із реальними бізнес-процесами ферми, визначити доцільний набір сутностей і атрибутів, а також знизити ризик функціональних прогалин під час реалізації програмного продукту.

Модель прецедентів інформаційної системи фермерського господарства



Висновки та перспективи. Побудова моделі прецедентів для інформаційної системи фермерського господарства забезпечує логічно узгоджене представлення ролей користувачів, функцій системи та сценаріїв їх реалізації. Це створює надійну основу для подальшого деталізованого проєктування структури даних, програмних модулів та користувацького інтерфейсу. Перспективою подальших досліджень є розширення моделі прецедентів для аналітичних функцій системи, автоматизації контролю подій та інтеграції з іншими цифровими сервісами супроводу фермерського господарства.

Список використаних джерел

1. Гриценко В. Г. та ін. UML-моделювання інформаційно-аналітичної системи «Навчальний план». URL: https://lib.iitta.gov.ua/id/eprint/706851/1/aisnn_uml.pdf
2. Проектування інформаційних систем : навчальний посібник. Запорізький національний університет. URL: https://files.znu.edu.ua/files/Bibliobooks/Inshi83/0063532.pdf
3. Мова об’єктно-орієнтованого проєктування UML. URL: http://repository.dnu.dp.ua:1100/upload/511aaeba61e5adde8c11ca688d0276e3UML.pdf

4. Методичні вказівки до організації самостійної роботи у середовищі VP/UML. КПІ ім. Ігоря Сікорського. URL: https://ela.kpi.ua/bitstream/123456789/57036/1/stapvp_srs_uml.pdf
5. Сучасні методи проєктування програмних систем. URL: http://dspace.dgma.donetsk.ua:8080/bitstream/DSEA/837/1/SMPPS_V02_%D0%905_1%2B1.pdf

Напрям 6 КОМП'ЮТЕРНІ МЕРЕЖІ ТА ТЕЛЕКОМУНІКАЦІЇ

Мацкевич Владислав Вікторович
Аспірант 3 курсу, АТСМ 11
Державний університет інформаційно-комунікаційних технологій
+380989903969
matsik1000@gmail.com

ІНТЕЛЕКТУАЛЬНІ АЛГОРИТМИ ОБРОБКИ СИГНАЛІВ У ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМАХ ЗВ'ЯЗКУ З РОЗНЕСЕНИМИ АНТЕНАМИ

Сучасний розвиток телекомунікаційних систем характеризується переходом до децентралізованих мереж, що забезпечують підвищену масштабованість, відмовостійкість та ефективне використання ресурсів. Особливу роль у таких системах відіграють технології з рознесеними антенами, зокрема MIMO (Multiple Input Multiple Output), які дозволяють значно збільшити пропускну здатність каналу зв'язку та підвищити спектральну ефективність.

У загальному вигляді модель MIMO-системи можна представити як:

$$\mathbf{y} = \mathbf{Hx} + \mathbf{n}$$

де

y— вектор прийнятого сигналу,
x— вектор переданого сигналу,
H— матриця каналу,
n— адитивний білий гаусів шум.

Якість прийому сигналу в таких системах значною мірою залежить від точності оцінювання матриці каналу **H**, яка є випадковою та змінюється у часі внаслідок багатопроменевого поширення сигналу, руху абонентів та змін середовища.

У децентралізованих системах обробка сигналів виконується не централізовано, а розподілено між кількома вузлами мережі.

Класичні методи оцінювання каналу, такі як метод найменших квадратів (LS), описуються виразом:

$$\widehat{\mathbf{H}}_{LS} = \mathbf{Y}\mathbf{X}^{H}(\mathbf{X}\mathbf{X}^{H})^{-1}$$

та метод мінімізації середньоквадратичної помилки (MMSE):

$$\widehat{\mathbf{H}}_{MMSE} = \mathbf{R}_{HY}\mathbf{R}_{YY}^{-1}$$

де $\mathbf{R}_{HY}$та $\mathbf{R}_{YY}$— кореляційні матриці сигналів.

Однак ці методи вимагають значних обчислювальних ресурсів та централізованого доступу до даних, що ускладнює їх застосування в децентралізованих системах.

У зв'язку з цим перспективним є використання інтелектуальних алгоритмів, зокрема штучних нейронних мереж. Нейромережевий підхід дозволяє апроксимувати функцію оцінювання каналу:

$$\widehat{\mathbf{H}} = f_{\theta}(\mathbf{y})$$

де f_θ— нелінійна функція, що задається параметрами нейронної мережі.

Застосування глибоких нейронних мереж (DNN), згорткових мереж (CNN) або рекурентних архітектур (RNN) дозволяє враховувати складні залежності у каналі зв'язку, включаючи просторову та часову кореляцію сигналів. При цьому навчання мережі може здійснюватися як централізовано, так і розподілено.

Перевагами інтелектуальних методів є:

- здатність працювати в умовах неповної або зашумленої інформації;
- адаптивність до змін параметрів каналу;
- зменшення обчислювальної складності на етапі використання;
- можливість реалізації у розподілених обчислювальних середовищах.

У децентралізованих системах кожен вузол може виконувати локальну обробку сигналу, використовуючи власну нейронну модель, а також обмінюватися узагальненими параметрами з іншими вузлами. Такий підхід відповідає концепції federated learning, у якій глобальна модель формується шляхом агрегування локальних оновлень:

$$\theta = \sum_{k=1}^{K} w_k \, \theta_k$$

де θ_k— параметри моделі на k-му вузлі, w_k— вагові коефіцієнти.

Це дозволяє значно зменшити обсяг передаваних даних та підвищити рівень конфіденційності.

Моделювання подібних систем може бути реалізоване у середовищі MATLAB із використанням інструментів для роботи з MIMO-каналами та нейронними мережами. Типовий сценарій моделювання включає генерацію випадкового каналу, формування сигналу, додавання шуму та подальше оцінювання каналу різними методами. Ефективність алгоритмів оцінюється за критерієм середньоквадратичної похибки:

$$MSE = \mathbb{E}\left[\| \mathbf{H} - \hat{\mathbf{H}} \|^2\right]$$

Результати чисельного моделювання свідчать, що нейромережеві алгоритми забезпечують зниження MSE у порівнянні з LS-оцінкою, особливо при низьких значеннях відношення сигнал/шум (SNR < 10 дБ). Крім того, вони демонструють кращу стійкість до нелінійних спотворень каналу.

Таким чином, інтеграція інтелектуальних алгоритмів у децентралізовані системи зв'язку з рознесеними антенами є перспективним напрямом розвитку телекомунікацій, що відповідає сучасним тенденціям цифровізації, програмно-орієнтованих мереж та пришвидшеному розвитку сучасних технологій 5G/6G.

Висновки

У роботі обґрунтовано доцільність використання інтелектуальних алгоритмів обробки сигналів у децентралізованих MIMO-системах. Показано, що застосування нейронних мереж дозволяє підвищити точність оцінювання

каналу та ефективність передачі даних. Встановлено, що найбільший виграш досягається в умовах складних каналів та низького SNR. Отримані результати можуть бути використані при розробці систем зв’язку нового покоління (5G/6G) та розподілених телекомунікаційних платформ.

Список використаної літератури

1. Goldsmith A. *Wireless communications*. Cambridge: Cambridge University Press, 2005.
2. Tse D., Viswanath P. *Fundamentals of wireless communication*. Cambridge: Cambridge University Press, 2005.
3. Рябенький В. М., Охріменко О. Г. Системи зв’язку з багатоканальним доступом (MIMO). Київ: НТУУ «КПІ», 2018.
4. Кравченко В. І., Лисенко О. В. Обробка сигналів у телекомунікаційних системах. Харків: ХНУРЕ, 2019.

Чугреєв Кирилл Олександрович
Інженер кафедри ТЦР
Державний університет інформаційно-комунікаційних технологій
063-946-23-83
ch.kir.lev.2003@gmail.com

ІНТЕЛЕКТУАЛЬНА ОПТИМІЗАЦІЯ МАРШРУТИЗАЦІЇ В ПРОГРАМНО-КЕРОВАНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ ІЗ ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

Постановка задачі

Стрімке зростання обсягів мережевого трафіку, поширення хмарних сервісів, Інтернету речей та мультимедійних застосунків висувають нові вимоги до продуктивності комп'ютерних мереж. Традиційні протоколи маршрутизації приймають рішення переважно на основі статичних метрик, що не дозволяє своєчасно реагувати на зміну навантаження, виникнення перевантажень або відмов окремих вузлів мережі. Унаслідок цього зростають затримки передачі пакетів, погіршується якість обслуговування користувачів та знижується ефективність використання мережевих ресурсів.

Одним із перспективних напрямів модернізації мережевої інфраструктури є використання програмно-керованих мереж, які забезпечують централізоване управління мережевими пристроями та дозволяють оперативно змінювати правила маршрутизації. Разом із тим ефективність SDN значною мірою залежить від здатності контролера швидко аналізувати поточний стан мережі та приймати оптимальні рішення щодо вибору маршрутів передачі даних.

У сучасних дослідженнях дедалі більше уваги приділяється використанню методів машинного навчання для прогнозування зміни характеристик мережевого трафіку, виявлення потенційних перевантажень та автоматичного вибору оптимальних маршрутів. Поєднання можливостей SDN та алгоритмів машинного навчання створює передумови для побудови самоадаптивних мереж, здатних підлаштовуватися під зміну умов функціонування без втручання адміністратора.

Мета дослідження

Метою дослідження є аналіз можливостей використання методів машинного навчання для інтелектуальної оптимізації маршрутизації в програмно-керованих комп'ютерних мережах та розроблення концептуального підходу до підвищення ефективності використання мережевих ресурсів шляхом прогнозування зміни стану мережі.

Результати дослідження

Проведений аналіз сучасних підходів до управління комп'ютерними мережами свідчить, що використання централізованого контролера SDN значно спрощує процес збору телеметричної інформації та забезпечує можливість оперативного керування потоками даних. Контролер отримує інформацію про завантаження каналів, затримки, втрати пакетів, пропускну здатність та інші

параметри функціонування мережі, що створює основу для застосування алгоритмів інтелектуального аналізу.

Формула інтегральної оцінки маршруту:

$$Score_i = w_1 \frac{B_i}{B_{max}} - w_2 \frac{D_i}{D_{max}} - w_3 \frac{L_i}{L_{max}} - w_4 \frac{J_i}{J_{max}}$$

B_i – доступна пропускна здатність;

D_i – затримка передачі;

L_i – рівень втрати пакетів;

J_i – джитер;

$w_1 \dots w_4$ – вагові коефіцієнти, що визначають важливість кожного показника.

Запропонований підхід передбачає інтеграцію модуля машинного навчання до складу SDN-контролера. На першому етапі здійснюється безперервний збір статистичних показників мережевого трафіку. Після попередньої обробки дані використовуються для навчання моделі, здатної прогнозувати рівень завантаження окремих каналів передачі даних у найближчому часовому інтервалі. На основі отриманого прогнозу контролер визначає найбільш ефективний маршрут, враховуючи не лише довжину шляху, а й очікуваний рівень навантаження, затримку передачі пакетів, коефіцієнт втрат та доступну пропускну здатність.

Формула прогнозування навантаження:

$$\widehat{L_{t+1}} = f(L_t, L_{t+1}, \dots, L_{t+n})$$

L_t – поточне навантаження мережі;

$\widehat{L_{t+1}}$ – прогнозоване навантаження;

$f(\cdot)$ – модель машинного навчання.

На відміну від класичних алгоритмів маршрутизації, запропонований підхід використовує прогнозні дані, що дозволяє виконувати превентивне перенаправлення трафіку до виникнення критичних перевантажень. Це забезпечує більш рівномірний розподіл навантаження між мережевими сегментами, підвищує стабільність функціонування мережі та покращує показники якості обслуговування.

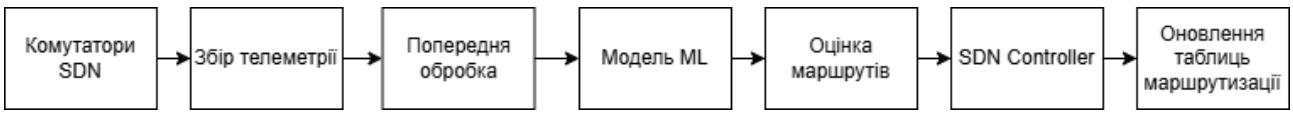


Рис 1. Алгоритм роботи системи

Концептуальна модель може бути використана в корпоративних мережах, дата-центрах, мережах операторів зв'язку, інфраструктурі Інтернету речей та інших системах, де характерною є динамічна зміна інтенсивності мережевого трафіку. Запропонований підхід також є сумісним із сучасними засобами мережевої телеметрії та технологіями автоматизованого управління інфраструктурою.

Висновки та перспективи

Використання програмно-керованих мереж у поєднанні з методами машинного навчання відкриває нові можливості для підвищення ефективності функціонування сучасної мережевої інфраструктури. Аналіз показав, що застосування прогнозних моделей дозволяє здійснювати інтелектуальне керування маршрутами передачі даних, своєчасно реагувати на зміну навантаження та мінімізувати ризик виникнення перевантажень окремих каналів зв'язку.

Запропонований концептуальний підхід може стати основою для побудови самоадаптивних мереж нового покоління, здатних автоматично оптимізувати власну конфігурацію відповідно до поточного стану мережевої інфраструктури. Перспективами подальших досліджень є реалізація запропонованої моделі в середовищі емуляції, проведення експериментального порівняння з традиційними алгоритмами маршрутизації та оцінювання впливу різних моделей машинного навчання на показники продуктивності мережі.

Список використаних джерел

8. Kreutz D., Ramos F. M. V., Verissimo P., Rothenberg C. E., Azodolmolky S., Uhlig S. Software-Defined Networking: A Comprehensive Survey. Proceedings of the IEEE. 2015. Vol. 103, No. 1. P. 14–76.

9. Al-Fuqaha A., Guizani M., Mohammadi M., Aledhari M., Ayyash M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials. 2015. Vol. 17, No. 4. P. 2347–2376.

10. Goyal P., Gupta S., Kumar V. Machine Learning in Software Defined Networks: Recent Advances and Research Challenges. Journal of Network and Computer Applications. 2022. Vol. 203. Article 103383. https://doi.org/10.1016/j.jnca.2022.103383

11. Stallings W. Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud. 2nd ed. Boston: Addison-Wesley, 2023. 560 p.

Напрям 7 КІБЕРБЕЗПЕКА ТА ЗАХИСТ ІНФОРМАЦІЇ

Далькевич Денис Олександрович,
студент 4 курсу, групи ТЦР-43
Державного університету інформаційно-комунікаційних технологій
+380 (96) 058 62 77
den921524@gmail.com

ПОРІВНЯЛЬНИЙ АНАЛІЗ ВЕБ-СЕРВІСІВ ОБМІНУ ПОВІДОМЛЕННЯМИ ЗА КРИТЕРІЯМИ АНОНІМНОСТІ ТА ЗАХИСТУ ДАНИХ

Постановка задачі.

В даний момент у всьому світі стає поширеною практика масового стеження (mass surveillance) з боку як різних державних органів і установ, так і технологічних комерційних корпорацій, забезпечення анонімності та конфіденційності електронних комунікацій стає нагальною потребою у користувачів. Сучасні популярні месенджери - WhatsApp, Telegram, Viber - попри наявність шифрування, зберігають значний обсяг метаданих - це номери телефонів, час активності, списки контактів, IP-адреси. Це дозволяє встановити особу користувача навіть без доступу до змісту повідомлень. Жоден з провідних сервісів не забезпечує повноцінної анонімності на рівні реєстрації та обміну даними. Постає задача - систематизувати недоліки існуючих рішень та визначити вимоги до якісно нової, більш анонімної та захищеної системи комунікації.

Мета дослідження.

Метою роботи є порівняльний аналіз існуючих веб-сервісів обміну повідомленнями за критеріями анонімності, захисту даних та архітектурних підходів, а також визначення ключових вимог до побудови захищеного месенджера нового покоління без прив'язки до персональних ідентифікаторів.

Результати дослідження.

Проведено аналіз трьох основних архітектурних підходів: централізованого (WhatsApp, Viber), частково децентралізованого (Telegram, Matrix) та peer-to-peer (Tox, Briar). Встановлено, що централізовані рішення забезпечують зручність і швидкість, але створюють єдину точку компрометації та концентрують метадані у провайдера.

Таблиця 1

Порівняльна характеристика популярних месенджерів за критеріями анонімності

Критерій	Signal	Telegram	WhatsApp
E2EE за замовч.	Так	Ні	Так
Метадані	Мінімум	Зберігає	Meta збирає

Критерій	Signal	Telegram	WhatsApp
Прив'язка до номера	Так	Так	Так
Відкритий код	Так	Частково	Ні
Анонімна реєстр.	Ні	Ні	Ні

Виявлено спільний критичний недолік усіх розглянутих систем - це обов'язкова прив'язка акаунту до номера телефону, що унеможливлює анонімну реєстрацію. Signal, попри мінімальний збір метаданих та відкритий вихідний код, також вимагає верифікацію номера. Telegram зберігає повідомлення у хмарі без наскрізного шифрування за замовчуванням, а WhatsApp передає метадані до екосистеми Meta.

За результатами аналізу визначено ключові вимоги до захищеного месенджера нового покоління: анонімна реєстрація без прив'язки до телефону чи email, наскрізне шифрування за замовчуванням, мінімізація збору метаданих на архітектурному рівні, Zero-Knowledge підхід до зберігання ключів та відкритий вихідний код для незалежного аудиту.

Висновки.

Жоден з сучасних провідних комерційних месенджерів не задовольняє повному спектру вимог до анонімної та захищеної комунікації. Виявлені системні недоліки: збір метаданих, обов'язкова ідентифікація, відсутність E2EE за замовчуванням - це формує чіткий запит на розробку альтернативного рішення. Як перспективний напрям розглядається розробка веб-сервісу з анонімною реєстрацією через псевдонім, наскрізним шифруванням на базі AES-256-CBC та Zero-Knowledge архітектурою, де сервер виконує виключно функцію маршрутизації зашифрованого трафіку.

Список використаних джерел:

1. Marlinspike M. The Signal Protocol. Signal Foundation, 2016.
2. Musiani F., Ermoshina K. What is a Good Secure Messaging Tool? The EFF Secure Messaging Scorecard and the Shaping of Digital (Usable) Security. Westminster Papers in Communication and Culture. 2017. Vol. 12(3). P. 51–71. DOI: 10.16997/wpcc.265
3. Unger N. et al. SoK: Secure Messaging. IEEE Symposium on Security and Privacy (S&P). 2015. P. 232–249. URL: https://www.ieee-security.org/TC/SP2015/papers-archived/6949a232.pdf
4. Snowden E. Permanent Record. New York: Henry Holt and Company, 2019. 339 p.
5. Katz J., Lindell Y. Introduction to Modern Cryptography. 3rd ed. CRC Press, 2020. 640 p.

Далькевич Денис Олександрович
студент 4 курсу, групи ТЦР-43
Державного університету інформаційно-комунікаційних технологій
den921524@gmail.com
+380 (96) 058 62 77

ПРОЄКТУВАННЯ СИСТЕМИ ОБМІНУ ПОВІДОМЛЕННЯМИ З НАСКРІЗНИМ ШИФРУВАННЯМ НА БАЗІ ASP.NET CORE ТА ANGULAR

Постановка задачі.

В сучасному світі системи обміну миттєвими повідомленнями стикаються з основною проблемою - забезпечення конфіденційності користувачів. Звичайні месенджери використовують транспортне шифрування (TLS/SSL), в результаті чого сервер має доступ до незашифрованого контенту повідомлень. Такий сценарій при компрометації серверної інфраструктури створює ризики витоку конфіденційної інформації або несанкціонованого доступу адміністраторів. Для захисту приватності користувачів виникає потреба застосування наскрізного шифрування (End-to-End Encryption, E2EE) [2], при якому криптографічні ключі зберігаються виключно на пристроях користувачів, а сервер лише виконує функцію пересилання зашифрованих повідомлень без можливості їх розшифрування.

Мета дослідження.

Метою даного проєкту постає розробка та реалізація веб-застосунку для обміну миттєвими повідомленнями з інтегрованим наскрізним шифруванням на базі технологій ASP.NET Core та Angular, що забезпечує максимальний рівень конфіденційності листування при збереженні зручності використання та високої продуктивності в реальному часі.

Результати.

В результаті планується розробити систему обміну повідомленнями, яка інтегрує технологію SignalR для двостороннього зв'язку в реальному часі з криптографічним модулем на основі алгоритму AES-256 (Advanced Encryption Standard) в режимі CBC (Cipher Block Chaining) [4]. Розроблений алгоритм має забезпечувати наскрізне шифрування всіх текстових повідомлень та файлів, гарантуючи що лише відправник та отримувач мають доступ до незашифрованого контенту.

Архітектура криптографічного модуля має бути побудована на принципі Zero-Knowledge - сервер не має доступу до ключів шифрування. Майстер-ключ має зберігатися виключно на стороні клієнта і ніколи не передається на сервер. Для кожного повідомлення **мають генеруватися** унікальні криптографічні параметри, які унеможливлюють розшифрування навіть при отриманні зашифрованих даних з бази даних.

Для забезпечення криптографічної стійкості буде застосовано алгоритм AES-256, затверджений NIST (National Institute of Standards and Technology) [4]. AES-256 використовує 256-бітний ключ, що забезпечує 2^{256} можливих комбінацій, роблячи brute force атаки практично неможливими (табл. 1).

Таблиця 1. Порівняльна характеристика методів шифрування

Параметр	**TLS/SSL**	**E2EE (AES-256-CBC)**
Доступ сервера до контенту	Так	Ні
Захист при компрометації сервера	Ні	Так
Криптографічна стійкість ключа	128-256 біт	256 біт
Можливість масового стеження	Так	Ні
Складність реалізації	Низька	Висока
Латентність шифрування	< 1 мс	2-5 мс

Для кожного повідомлення мають бути згенеровані унікальні криптографічні параметри: вектор ініціалізації (IV, 128 біт) та сіль (Salt, 256 біт). На основі Salt через PBKDF2 з 100 000 ітерацій деривується сесійний ключ. Відправник шифрує текст алгоритмом AES-256-CBC та передає структуру {content, IV, Salt} через SignalR Hub; сервер зберігає зашифровані дані без можливості розшифрування; отримувач деривує ключ з того ж Salt та дешифрує повідомлення **[3] (рис. 1)**.

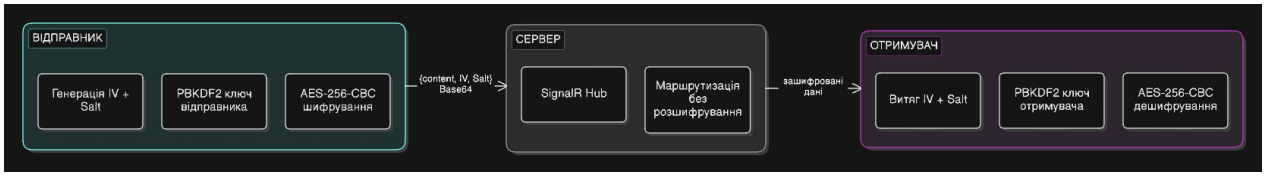


Рисунок 1. Схема обміну зашифрованими повідомленнями (E2EE)

Інтеграція з SignalR має забезпечувати доставку зашифрованих повідомлень у реальному часі з мінімальною затримкою (< 100 мс). SignalR Hub маршрутизує повідомлення до конкретних користувачів або груп без розшифрування контенту. Авторизація має бути реалізована через JWT-токени що зберігаються в HttpOnly cookie з атрибутами Secure та SameSite=Strict [1].

Висновки.

Така система обміну повідомленнями з наскрізним шифруванням на базі AES-256-CBC повинна демонструвати високий рівень безпеки при збереженні прийнятної продуктивності для роботи в реальному часі. Принцип Zero-Knowledge гарантує що навіть при повній компрометації серверної інфраструктури відкритий текст повідомлень залишається недоступним.

Використання динамічних параметрів IV та Salt з функцією деривації ключа PBKDF2 забезпечує захист від сучасних криптографічних атак **[2]**.

В подальшому на перспективу для вдосконалення застосунку можуть бути розроблені слідуючі напрямки: впровадження протоколу Diffie-Hellman для Perfect Forward Secrecy **[3]**, інтеграція цифрових підписів для верифікації відправника та підтримка групового шифрування.

Список використаних джерел:

1. Docs M. SignalR documentation. URL: https://learn.microsoft.com/en-us/aspnet/core/signalr/.
2. Katz J., Lindell Y. Introduction to modern cryptography. CRC Press, 2020. 640 с.
3. Marlinspike M., Perrin T. The double ratchet algorithm. URL: https://signal.org/docs/specifications/doubleratchet/.
4. Standards N. I. o., Technology. Advanced encryption standard (AES). 2001. 51 с.

Кухаренко Артем Русланович
студент 4 курсу, групи ІСД-41
Державний університет інформаційно-комунікаційних технологій
(096) 916-14-36
kuharenkoartem131104@gmail.com

БЕЗПЕКА, НАДІЙНІСТЬ ТА ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ IOT-СИСТЕМ БЕЗПЕРЕБІЙНОГО ЕЛЕКТРОЖИВЛЕННЯ

Постановка задачі. У наш час інформаційно-комунікаційні системи постійно потребують стабільного електроживлення. Якщо говорити про класичні UPS (джерела безперебійного живлення), то вони можуть забезпечити лише тимчасову автономність від мережі, проте не мають вбудованих технологій для прогнозування відмов або захисту від хакерських атак. Виникає потреба у створенні більш інтелектуальних систем, що будуть ефективнішими, економічнішими та безпечнішими у роботі.

Мета дослідження. Метою дослідження є розробка концепції інтеграції технологій інтернету речей у системи безперебійного електроживлення. Це допоможе забезпечити надійність роботи, оптимізувати фінансові витрати та підвищити рівень захисту енергетичних систем.

Результати дослідження.

1. **Безпека IoT-UPS**

Сфера кіберзахисту відіграє важливу роль у будь-яких інформаційних системах. Вона напряму відповідає за ступінь надійності та якості роботи мережі. Інтернет речей не став винятком. Він також потребує серйозних дій для захисту інформації. Будь-який фізичний прилад, шлюз чи сенсор стає потенційною точкою входу для кібератак. Це створює потребу у використанні спеціальних протоколів шифрування (TLS, AES), сегментації мережі та багаторівневої аутентифікації. Впровадження цих технологій допоможе звести до мінімуму ризики несанкціонованого доступу до керування електричною системою. Дослідження компанії з публікації наукових матеріалів Springer показали, що моделювання загроз у smart-grid (розумна енергосистема) знизило ймовірність хакерських атак на 30-35% [1].

2. **Надійність роботи**

Надійність роботи систем інтернету речей в першу чергу полягає у моніторингу параметрів мережі в режимі реального часу. Додатково користувач має можливість віддалено отримувати детальну інформацію про якість та ефективність роботи системи. Все це відбувається завдяки алгоритмам машинного навчання. Вони аналізують телеметричні параметри, а також прогнозують потенційні несправності чи критичні ситуації. Це дозволяє заздалегідь попереджати користувача про можливі проблеми в роботі мережі чи аварії.

За даними IEEE Xplore, використовуючи спеціальні системи моніторингу батарей, можна підвищити надійність джерел безперебійного електроживлення

завдяки автоматичному перемиканню джерел із основних на альтернативні [2]. Кількість критичних відмов та аварій при цьому скоротилася більше ніж на чверть.

3. Економічна ефективність

Предиктивна аналітика дозволила компаніям та підприємствам перейти від реактивного до превентивного ремонту. Це означає, що мережі здатні "діяти на випередження" і попереджати про можливі несправності ще до того, як вони безпосередньо виникли. Така концепція зменшила фінансові витрати на обслуговування в середньому на 15%.

Окрім цього, мережі безперебійного електроживлення на базі інтернету речей вирізняються ще однією властивістю. Вони дозволили збалансувати навантаження на джерела, уникаючи при цьому пікових тарифів. За оцінками McKinsey, опублікованими у жовтні 2025 року, потенційна економічна вигода від IoT у енергетичній галузі до 2030 року складе щонайменше сотні мільярдів доларів.

Нижче наведено порівняльну таблицю (табл. 1.1), що описує різницю у трьох цих критеріях між традиційними джерелами живлення та джерелами на основі інтернету речей.

Таблиця 1.1

Порівняння класичних UPS та IoT-UPS

Критерій	**Традиційні UPS**	**IoT-UPS**
Безпека	Мінімальний захист, відсутність детального моніторингу	Шифрування, сегментація та аутентифікація
Надійність	Реактивне реагування, висока імовірність людських помилок	Прогнозування несправностей, превентивний ремонт, постійний моніторинг
Економічність	Високі витрати через терміновий ремонт та простої системи	Зниження витрат, продовження терміну роботи приладів

Висновки та перспективи. Використання в енергетичній галузі концепції інтернету речей створює триєдиний ефект. До нього належать захист від кіберзагроз, підвищення надійності та економічна оптимізація. Системи із інтегрованим IoT стають перспективними стратегічними інструментами для конкурентоспроможності між іншими підприємствами, забезпечуючи стійкість та якість роботи.

Перспективним напрямом також вважається використануя IoT у комплексні енергетичні платформи smart-grid та введення алгоритмів глибокого машинного навчання. Це уможливить більш точно прогнозувати відмови або оповіщувати користувачів про критичні ситуації.

Список використаних джерел

1. Smart Grid [Електронний ресурс]. - Режим доступу до ресурсу: https://link.springer.com/rwe/10.1007/978-1-4614-6431-0_78-1

2. High-Capacity motors [Електронний ресурс]. - Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/1234567

3. Global Energy Perspective - 2025 [Електронний ресурс]. - Режим доступу до ресурсу: https://www.mckinsey.com/industries/energy-and-materials/our-insights/global-energy-perspective

Ахмедов Амір Фарідович
студент 2 курсу, група БСД-22
Державний університет інформаційно-комунікаційних технологій
(067)700-03-88
infoforamir@gmail.com
Рашидов Сейфулла Фейзуллайович
доктор педагогічних наук, кандидат філософських наук,
професор кафедри Загальної педагогіки та спеціальної освіти
Ізмаїльського державного гуманітарного університету, м. Ізмаїл
seifulla.rashydov@gmail.com

ПЕРЕДБАЧУВАНА ГІПОТЕЗА ЯК ЕПІСТЕМОЛОГІЧНИЙ ІНСТРУМЕНТ ПРОЄКТУВАННЯ БІОМЕТРИЧНИХ DID ДЛЯ ПОСТКВАНТОВОГО ЗАХИСТУ ІНФОРМАЦІЇ

Постановка задачі. Сучасні стратегії кібербезпеки розглядають впровадження постквантової криптографії (Post-Quantum Cryptography, PQC) як безальтернативний шлях захисту інформаційних систем в еру квантових обчислень. Однак інтеграція PQC-протоколів суттєво ускладнює програмне забезпечення, спричиняючи значне зростання розміру ключів, обчислювальних витрат та вимог до мережевої інфраструктури. Завданням дослідження є епістемологічний аналіз логічної структури гіпотези, що пропонує альтернативний вектор розвитку через використання біометричних децентралізованих ідентифікаторів (Decentralized Identifier, DID) [1, розд. 1], де безпека базується не на алгоритмічній складності, а на внутрішній ентропії біометричних даних суб'єкта.

Мета дослідження. Адаптувати концепцію фальсифікаціонізму Карла Поппера [2, розділ 4] до проєктування біометричних DID-систем. Дослідження має на меті обґрунтувати гіпотезу про достатність біометричної ентропії як альтернативного механізму захисту в постквантову еру та визначити емпіричні критерії, за яких ця гіпотеза може бути спростована.

Результати дослідження. У філософії науки Карл Поппер запропонував критерій демаркації, згідно з яким наукова гіпотеза відрізняється від ненаукової принциповою можливістю її спростування (фальсифікації). Застосування цієї логіки до проєктування біометричних DID дозволяє розглядати безпеку не як статичну властивість алгоритму, а як динамічний процес перевірки «сміливих припущень» (conjectures). Будь-яке твердження про достатність біометричної ентропії для захисту інформації має бути сформульоване як передбачувана гіпотеза, що відкрита до емпіричної перевірки. Такий підхід трансформує проєктування ПЗ з пасивного накопичення надлишкових криптографічних рівнів у активний фальсифікаціонізм, а саме ретельну перевірку здатності біометричного коду зберігати високу ентропійну щільність під тиском квантових обчислювальних моделей.

Сформулюємо гіпотезу H про достатність біометричної ентропії як сукупність трьох підгіпотез H.1, H.2 та H.3:

H.1: біометричний протокол детермінованого перетворення здатен генерувати стабільний ідентифікаційний код з ефективною ентропією не менше 256 біт (у сенсі інформаційної ентропії, релевантної до моделі атак перебору);

H.2: згенерований код не демонструє відомих відхилень від поведінки випадкового ключа в межах прийнятої моделі загроз (рівномірність розподілу, незалежність бітів, стійкість до атак на відновлення біометричного шаблону);

H.3: у поточній технологічній парадигмі відсутні вектори атак, що знижують ефективну складність зламу нижче рівня, передбаченого алгоритмом Гровера.

Кон'юнкція H = H.1 ∧ H.2 ∧ H.3 визначає інтегральну гіпотезу проєктування, згідно з якою за умови істинності всіх трьох складових біометрична DID-система досягає рівня постквантової стійкості Level 5 Національного бюро стандартів США (National Institute of Standards and Technology, NIST) [3, sec. 3], що детермінує експлікацію недоцільності PQC-алгоритмів як функціонально надлишкових.

Застосування принципу Бритви Оккама («не слід вводити зайві сутності без потреби») [4, p. 12] дозволяє кваліфікувати PQC-механізми як умовно надлишкові в межах запропонованої моделі.

Фундаментальна фальсифіковність гіпотези полягає у можливості спростування будь-якого елемента тріади, що зумовлює наукову цінність моделі через відкритість до суворої критики.

Для досягнення філософської глибини аналізу необхідно експлікувати три парадигми цифрової ідентифікації:

- identity-as-record (ідентичність як запис) - класична модель DID, де легітимність суб’єкта визначається наявністю відповідного запису в розподіленому реєстрі;
- identity-as-attribute (ідентичність як набір ознак) - традиційна біометрія, що базується на зовнішньому порівнянні пред’явлених даних із еталоном;
- identity-as-existence (ідентичність як факт існування) - пропонована модель, у якій ідентифікатор є похідним від біометричної сутності суб’єкта, а «джерелом істини» (Source of Truth) виступає сама людина.

Реалізуючи принцип самосуверенної ідентичності (Self-Sovereign Identity, SSI) у його радикальній формі, остання модель перетворює суб’єкта на самодостатнє джерело власної цифрової репрезентації, що створює передумови для верифікації гіпотези H, де біометричний код виступає первинним самопідтверджуваним базисом постквантової стійкості.

Висновки та перспективи. Основним внеском цієї роботи є не конкретна архітектурна реалізація, а формування нової епістемологічної рамки, що дозволяє переосмислити природу цифрової довіри. Показано, що передбачувана гіпотеза у розумінні Карла Поппера виступає адекватним інструментом для формалізації дослідницьких претензій у галузі біометричної ідентифікації. Запропонована модель переміщує акцент із зовнішніх алгоритмічних засобів

підтвердження ідентичності на онтологічну сутність біометричного суб'єкта. Цифрова ідентичність повертається до свого першоджерела, де фундаментальною основою довіри стає безпосереднє існування людини замість формального запису в реєстрі.

Подальші дослідження відкривають перспективу створення нового класу постквантово-стійких біометричних DID на основі переходу від моделі математичної складності до моделі епістемологічної достатності біометричного коду.

Список використаних джерел:

1. W3C, “Decentralized Identifiers (DIDs) v1.0”, World Wide Web Consortium, 2022. URL: https://www.w3.org/TR/did-core/
2. Popper K. The Logic of Scientific Discovery. London: Routledge, 1959.
3. NIST, "Post-Quantum Cryptography Standardization", National Institute of Standards and Technology, 2022. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography
4. Stanford Encyclopedia of Philosophy, "William of Ockham," 2022. URL: https://plato.stanford.edu/entries/ockham/

Канарський Артем В’ячеславович
студент 4 курсу, групи ПД-44
Державний університет інформаційно-комунікаційних технологій
(067) 663-42-41
st8036057@stud.duikt.edu.ua

ОРГАНІЗАЦІЯ БЕЗПЕКИ ДАНИХ У СИСТЕМІ РЕЗЮМУВАННЯ ТЕКСТІВ

Постановка завдання. У сучасних умовах цифровізації забезпечення захисту даних користувачів у межах програмних систем обробки текстової інформації набуває особливої актуальності. Зростання обсягів текстових даних та їх активне використання в інформаційних системах зумовлює підвищення ризиків витоку, втрати або несанкціонованого доступу до інформації. Це положення є особливо значущим для програмних продуктів, призначених для аналізу та обробки текстів, що можуть містити конфіденційні або персональні відомості. Пріоритетним завданням є організація безпечного процесу обробки текстової інформації шляхом мінімізації ризиків витоку даних, здійснення контролю вхідної інформації та забезпечення конфіденційності під час функціонування програмного засобу.

Мета дослідження. Метою роботи є розробка та аналіз програмного продукту для автоматичного резюмування текстів із впровадженням базових принципів інформаційної безпеки, зокрема локальної обробки даних, верифікації вхідних файлів та безпечного використання бібліотек та мінімізації обсягу збережуваних даних користувача.

Результати. Під час розроблення програмного продукту впроваджено систему автоматичного реферування текстів, що дозволяє скоротити обсяг вхідної інформації зі збереженням її змістового наповнення на основі методів обробки природної мови [1]. Програмне забезпечення здійснює токенізацію тексту, усунення стоп-слів, аналіз частотності ключових слів та ідентифікацію найбільш значущих речень для формування підсумкового резюме.

Одним із ключових аспектів функціонування системи є її здатність працювати в локальному режимі. Усі процеси опрацювання даних виконуються виключно на пристрої користувача без залучення мережевих запитів чи передачі інформації на зовнішні сервери, що суттєво підвищує рівень конфіденційності [2]. Завдяки цьому забезпечується реалізація принципу мінімізації обробки даних, що є фундаментальною вимогою в галузі інформаційної безпеки. Система забезпечує підтримку форматів TXT, PDF та DOCX із обов’язковою попередньою валідацією типу файлу перед початком його опрацювання. Зазначений підхід мінімізує ймовірність використання пошкоджених, потенційно небезпечних або несумісних документів. Крім того, впроваджено механізм обробки виняткових ситуацій, що гарантує стабільність функціонування програмного забезпечення в разі виникнення помилок під час завантаження або аналізу даних.

Ключовим аспектом безпеки є мінімізація збереження інформації. Програмний засіб не здійснює автоматичного логування, резервного копіювання чи передачі текстових даних третім особам. Усі операції виконуються виключно в оперативній пам’яті, а збереження результатів на носій відбувається лише за безпосередньої ініціативи користувача. Така архітектура дозволяє зменшити цифровий слід та мінімізувати ризики витоку конфіденційних відомостей.

Для реалізації зазначеного функціоналу було залучено бібліотеку Natural Language Toolkit (NLTK) [3], що є загальновизнаним та апробованим інструментом для вирішення завдань обробки природної мови.

Висновки. У ході дослідження встановлено, що базовий рівень захисту даних у системі анотування текстів досягається шляхом локальної обробки інформації, контролю типів вхідних файлів, використання перевірених програмних бібліотек та мінімізації обсягів збережених даних. Запропонований підхід дозволяє знизити ймовірність витоку конфіденційної інформації, підвищити рівень приватності користувачів та забезпечити дотримання фундаментальних принципів інформаційної безпеки [2].

```
if file_path.endswith(".txt"):
    text = load_text_file(file_path)
elif file_path.endswith(".pdf"):
    text = load_pdf_file(file_path)
elif file_path.endswith(".docx"):
    from file_loader import load_docx_file
    text = load_docx_file(file_path)
else:
    messagebox.showerror( title: "Помилка",  message: "Формат файлу не підтримується")
```

Рис. 1 – Перевірка формату вхідного файлу перед обробкою

Список використаних джерел

1. Jurafsky D., Martin J. *Speech and Language Processing*. – 3rd ed. – Draft, 2023. URL: https://web.stanford.edu/~jurafsky/slp3/
2. ДСТУ EN ISO/IEC 27002:2024 Інформаційна безпека, кібербезпека та захист конфіденційності. Заходи забезпечення інформаційної безпеки (на основі ISO/IEC 27002:2022). URL: https://learn.ztu.edu.ua/pluginfile.php/271472/mod_resource/content/2/ISO%2027002%202022.pdf
3. NLTK – URL: https://www.nltk.org/

Гайдамаченко Тимофій Максимович,
магістр 5 курсу, групи ТЦРМ-51
Державний університет інформаційно-комунікаційних технологій
(067) 706-81-99
timagaidamachenko@gmail.com

АТАКА НА ЛАНЦЮГ ПОСТАЧАННЯ ЧЕРЕЗ TYPOSQUATTING: ВИКОРИСТАННЯ НЕКЕРОВАНИХ LLM ДЛЯ ГЕНЕРАЦІЇ ШКІДЛИВИХ DOCKER-КОНТЕЙНЕРІВ

Постановка задачі. Атаки на ланцюг постачання (supply chain attacks) є одним із найнебезпечніших векторів сучасних кіберзагроз: жертва сама встановлює шкідливий компонент, вважаючи його легітимним. Паралельно розвиток некерованих open-source LLM без обмежень безпеки знизив технічний поріг створення якісного шкідливого коду. Поєднання typosquatting як вектора доставки та некерованої LLM як інструменту генерації шкідливого вмісту формує новий клас атак із мінімальним порогом входу та вкрай ускладненим виявленням.

Мета дослідження. Метою роботи є документування реального інциденту, що поєднав техніку typosquatting з LLM-генерованим шкідливим кодом у Docker-контейнері, формулювання forensics-методики атрибуції коду до некерованої LLM та формування практичних рекомендацій щодо захисту ланцюга постачання у процесах CI/CD.

Результати дослідження. Некеровані LLM як інструмент зловмисника. Комерційні платформи (ChatGPT, Claude, Copilot) реалізують захист від генерації шкідливого коду через RLHF та класифікатори небезпечних запитів. Натомість моделі родин Llama (Meta), Mistral, DeepSeek та їх «uncensored»/«abliterated» похідні вільно доступні на Hugging Face. Запустити таку модель локально через Ollama чи llama.cpp коштує небагато, а ефект для зловмисника серйозний. По-перше, нікуди не йде телеметрія, тому провайдер AI про активність такого користувача нічого не знає. По-друге, немає ні rate limiting, ні фільтрації запитів, тому модель спокійно генерує те, що комерційні сервіси відмовляться писати. Окремо плюсом є те, що всі запити зберігаються тільки локально — ніяких логів на стороні провайдера не лишається взагалі.

Опис зафіксованого інциденту. Інцидент, з яким довелось мати справу, поєднував одразу два вектори — typosquatting на рівні домену та шкідливий код у Docker-образі, очевидно згенерований LLM. Зловмисник зареєстрував домен, що відрізнявся від справжнього джерела образу всього на одну літеру. Сам образ при цьому виконував усі заявлені функції воркера, тому виявити підміну вдалося не одразу. Хронологія розслідування включала: виявлення аномальної мережевої активності контейнера (трафік до зовнішніх адрес, нехарактерних для штатної роботи); аудит конфігурації з підтвердженням typosquatting-розбіжності; ізоляцію контейнера та декомпіляцію образу з виявленням шкідливого скрипту,

що виконувався паралельно з легітимними функціями; повний аудит залежностей інфраструктури та оновлення процедур верифікації.

Forensics-методика атрибуції до некерованої LLM. Довести, що шкідливий код написаний саме LLM, а не людиною, складно — жодна окрема ознака сама по собі цього не підтверджує. Але якщо зібрати разом дрібні деталі, картина зазвичай складається. Перша підозріла річ — коментарі. У реальному шкідливому коді їх мінімум, бо читабельність автору не потрібна. Тут же майже кожна функція мала охайний педантичний коментар, причому в одному стилі. Назви змінних і функцій теж виглядали нетипово людськими: занадто описові, формулювання у стилі «опиши що робить», як це генерують моделі. Ідентифіковані характерні ознаки: надмірна педантична коментованість коду — реальний шкідливий код мінімізує читабельність, тому наявність детальних коментарів є парадоксальним індикатором автоматизованої генерації; семантично точні назви у стилі «describe-what-it-does», нетипові для людини; структурна однорідність без авторського «відбитку»; стандартизовані патерни обробки помилок; невідповідність між надмірно високою якістю коду та його шкідливим призначенням. Відсутність артефактів комерційних платформ у поєднанні з переліченими ознаками підтвердила використання локально розгорнутої некерованої моделі.

Рекомендації. Верифікація цифрових підписів усіх зовнішніх Docker-образів через Docker Content Trust або Sigstore/Cosign перед кожним завантаженням у CI/CD; фіксація залежностей по SHA256-digest замість плаваючих тегів — унеможливлення підміни образу навіть при компрометації тегу; автоматична перевірка доменів джерел на typosquatting при додаванні нових залежностей та регулярний аудит наявних; runtime-моніторинг контейнерів (Falco, Sysdig) для виявлення аномальної мережевої активності, що виходить за межі штатного профілю воркера.

Висновки та перспективи. Сама по собі ні техніка typosquatting, ні шкідливий Docker-образ не є новинкою. Що тут справді нове — це якість коду всередині. Якщо раніше шкідливий скрипт у такому каналі доставки можна було впізнати за грубістю реалізації, то LLM прибирає й цей бар'єр. Доставка йде через звичний CI/CD-конвеєр, і відрізнити підмінений образ від справжнього без додаткового аналізу майже не виходить. З іншого боку, у самого коду залишаються характерні сліди генерації, помітні при уважному розборі. Питання, яке доцільно дослідити окремо — чи можна навчити статичний аналізатор автоматично виловлювати ці ознаки, і наскільки точно це вийде на практиці.

Список використаних джерел

1. Sophos. The State of Ransomware 2025: A Sophos Whitepaper. — Sophos Ltd., 2025.

2. Verizon. 2025 Data Breach Investigations Report (DBIR). — Verizon Communications Inc., 2025.

3. Cloudflare. DDoS Threat Report 2025 Q4. — Cloudflare Inc., 2026.

4. ENISA Threat Landscape 2025. — European Union Agency for Cybersecurity, 2025.

Напрям 8 BLOCKCHAIN ТА РОЗПОДІЛЕНІ ТЕХНОЛОГІЇ

Орлов Роман Русланович
аспірант 2 курсу
Державний університет інформаційно-комунікаційних технологій
старший оперуповноважений
Управління протидії кіберзлочинам в м. Києві Департаменту кіберполіції Національної поліції України
старший викладач
Приватний заклад вищої освіти «Дніпровський технологічний університет «ШАГ»,
(093)772-79-47
romanorlov0110@gmail.com

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ЦИФРОВОЇ КРИМІНАЛІСТИКИ КІБЕРІНЦИДЕНТІВ ІЗ ВИКОРИСТАННЯМ РОЗПОДІЛЕНИХ ТЕХНОЛОГІЙ ТА МОДЕЛЕЙ ОЦІНКИ РИЗИКІВ

Постановка задачі. Підвищення ефективності цифрової криміналістики кіберінцидентів в умовах стрімкого розвитку інформаційних технологій є одним із ключових завдань сучасної системи кібербезпеки. Зростання кількості та складності кібератак, поява нових векторів загроз, а також активне використання розподілених інформаційних систем обумовлюють необхідність удосконалення підходів до розслідування інцидентів.

Мета дослідження. Метою дослідження є підвищення ефективності цифрової криміналістики кіберінцидентів шляхом інтеграції розподілених технологій та моделей оцінки ризиків для забезпечення надійності, цілісності й оперативності обробки цифрових доказів.

Результати дослідження. Підвищення ефективності цифрової криміналістики кіберінцидентів в умовах стрімкого розвитку інформаційних технологій є одним із ключових завдань сучасної системи кібербезпеки. Зростання кількості та складності кібератак, поява нових векторів загроз, а також активне використання розподілених інформаційних систем обумовлюють необхідність удосконалення підходів до розслідування інцидентів. У цьому контексті особливої актуальності набуває поєднання методів цифрової криміналістики з технологіями розподілених реєстрів та ризик-орієнтованими моделями управління безпекою.

Цифрова криміналістика як науково-практична галузь спрямована на виявлення, збирання, аналіз і збереження цифрових доказів з метою встановлення обставин кіберінцидентів. Проте традиційні методи розслідування часто стикаються з низкою проблем, зокрема забезпеченням цілісності доказів, відсутністю єдиного джерела довіри, складністю координації між різними суб’єктами розслідування та високими вимогами до швидкості реагування. Саме тому впровадження розподілених технологій, зокрема Blockchain, відкриває нові можливості для підвищення ефективності криміналістичних процесів [1].

Технології розподіленого реєстру дозволяють створювати децентралізовані системи зберігання даних, у яких інформація фіксується у вигляді незмінних записів, захищених криптографічними методами. Використання таких технологій у цифровій криміналістиці забезпечує високий рівень довіри до зібраних доказів, оскільки будь-які спроби їх зміни або підробки можуть бути виявлені. Крім того, Blockchain дозволяє реалізувати прозорий механізм відстеження ланцюга збереження доказів (chain of custody), що є критично важливим для їх подальшого використання у судових процесах.

Інтеграція розподілених технологій у процеси розслідування кіберінцидентів також сприяє покращенню взаємодії між різними учасниками, такими як правоохоронні органи, державні установи, приватні компанії та міжнародні організації. Завдяки децентралізованому характеру системи забезпечується синхронізація даних у реальному часі, що дозволяє оперативно обмінюватися інформацією про інциденти та координувати дії щодо їх локалізації та нейтралізації [2].

Водночас важливим елементом підвищення ефективності цифрової криміналістики є використання моделей оцінки ризиків. Ризик-орієнтований підхід передбачає ідентифікацію найбільш критичних активів, визначення потенційних загроз і вразливостей, а також оцінку ймовірності їх реалізації та можливих наслідків. Застосування таких моделей дозволяє оптимізувати процеси розслідування, зосереджуючи ресурси на найбільш значущих інцидентах і скорочуючи час реагування.

Поєднання розподілених технологій і моделей оцінки ризиків створює синергетичний ефект, який суттєво підвищує ефективність цифрової криміналістики. З одного боку, Blockchain забезпечує надійність і прозорість зберігання даних, а з іншого — ризик-орієнтовані моделі дозволяють визначати пріоритети та прогнозувати розвиток подій. Це дає змогу перейти від реактивного до проактивного підходу в розслідуванні кіберінцидентів, коли увага приділяється не лише ліквідації наслідків, але й запобіганню виникненню нових загроз.

Важливим аспектом є також використання аналітичних інструментів, що базуються на технологіях обробки великих даних та штучного інтелекту. У поєднанні з розподіленими реєстрами вони дозволяють автоматизувати процеси виявлення аномалій, аналізу поведінкових патернів і прогнозування кіберзагроз. Це, у свою чергу, підвищує точність і швидкість прийняття рішень, що є критично важливим у сучасних умовах [3].

Необхідно також враховувати організаційні та правові аспекти впровадження таких підходів. Зокрема, потребують удосконалення нормативно-правові механізми, що регулюють використання розподілених технологій у сфері криміналістики, забезпечення захисту персональних даних, а також визначення стандартів і процедур обробки цифрових доказів. Крім того, важливим є підвищення рівня підготовки фахівців, здатних ефективно працювати з новітніми технологіями та застосовувати ризик-орієнтовані підходи у практичній діяльності.

Висновки та перспективи. Таким чином, підвищення ефективності цифрової криміналістики кіберінцидентів із використанням розподілених технологій та моделей оцінки ризиків є перспективним напрямом розвитку системи кібербезпеки. Інтеграція цих підходів дозволяє забезпечити надійність і прозорість процесів розслідування, підвищити швидкість реагування на інциденти, оптимізувати використання ресурсів і зміцнити загальний рівень інформаційної безпеки. У подальшому дослідження в цій сфері мають бути спрямовані на розробку практичних механізмів впровадження зазначених технологій, а також на створення ефективних моделей взаємодії між усіма суб'єктами кібербезпеки.

Список використаних джерел

1. Stallings W. Cryptography and Network Security: Principles and Practice. – 7th ed. – Boston: Pearson, 2017. – 768 p.
2. Casey E. Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet. – 3rd ed. – London: Academic Press, 2011. – 840 p.
3. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System [Електронний ресурс]. – Режим доступу: https://bitcoin.org/bitcoin.pdf
4. ISO/IEC 27005:2018. Information technology — Security techniques — Information security risk management. – Geneva: ISO, 2018.

Сягровський Павло Дмитрович,
студент 4 курсу групи ТЦР-42,
Державний університет інформаційно-комунікаційних технологій
(098)603-42-32
pasha.syagro@gmail.com

ІНТЕГРАЦІЯ BLOCKCHAIN-ТЕХНОЛОГІЙ У ВЕБ-ПЛАТФОРМІ AUTOASSIST+ ДЛЯ ВЕРИФІКАЦІЇ СЕРВІСНИХ ПЛАТЕЖІВ

Постановка задачі. Сучасні сервісні платформи технічної допомоги, логістики та страхових кейсів працюють в умовах високої динаміки звернень, необхідності швидкого реагування та постійного контролю фінансових операцій. Традиційні вебсистеми автоматизують реєстрацію заявок і супровід статусів, однак часто не забезпечують незалежної перевірки факту платежу, цілісного цифрового сліду транзакції та формалізованого механізму підтвердження виконаних дій. Актуальною є побудова веб-платформи, у якій blockchain-механізми використовуються для верифікації сервісних платежів, а геолокаційні сервіси — для просторового підбору виконавців [1], [2].

Мета дослідження. Метою дослідження є обґрунтування підходів до реалізації платформи AutoAssist+, у якій блокчейн-вузол, Web3-гаманець та вебінтерфейс клієнта інтегровані в єдиний життєвий цикл сервісного замовлення. Окрему увагу приділено перевірці txHash, контролю параметрів транзакції, формуванню квитанцій і підвищенню прозорості фінансової взаємодії між клієнтом і сервісним центром.

Результати дослідження. У межах дослідження платформа AutoAssist+ розглядається як клієнт-серверна інформаційна система, що включає модулі створення замовлення, автоматизованого оцінювання вартості, оплати, генерації квитанцій і зберігання вкладень. Клієнтський вебінтерфейс забезпечує введення параметрів транспортного засобу, опис проблеми, вибір місця події та запуск сценарію оплати. Серверна частина відповідає за збереження замовлень, побудову оцінки, фіксацію суми перед оплатою, перевірку платежу й формування історії подій.

Ключовим елементом blockchain-орієнтованої архітектури є модуль Web3-платежів. Після підтвердження оцінки вартості система блокує суму в межах замовлення, а далі користувач ініціює транзакцію через сумісний криптогаманець. Серверна частина отримує txHash і виконує перевірку транзакції за допомогою блокчейн-провайдера. Контролюються мережа, адреса отримувача, сума та кількість підтверджень. Для реалізації такого сценарію доцільно застосовувати бібліотеку ethers.js та підхід Sign-In with Ethereum, що забезпечують підпис повідомлень, взаємодію з блокчейн-вузлом і перевірку стану транзакції [1], [3]. Схему взаємодії основних модулів наведено на рис. 1.

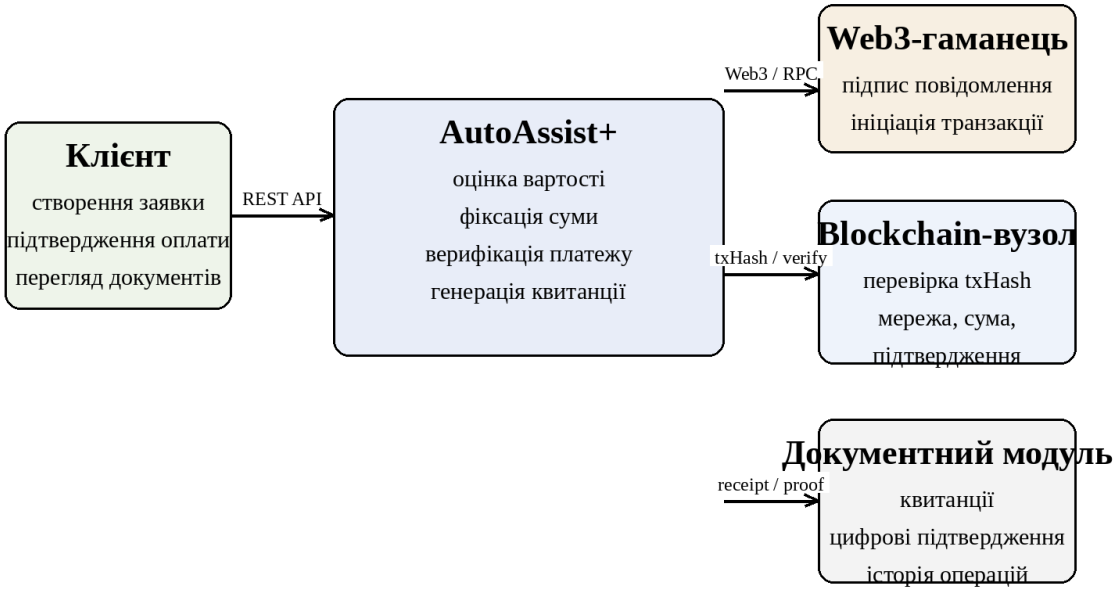


Рис. 1. Blockchain-верифікація платежу в платформі AutoAssist+

На відміну від класичної моделі оплати, blockchain-підхід підвищує прозорість фінансової взаємодії, спрощує незалежну перевірку факту платежу та забезпечує кращу відтворюваність цифрового сліду операції. Після успішної перевірки транзакції стають доступними квитанція, історія платежу та цифрове підтвердження виконання. Геолокаційний модуль при цьому зберігає значення для ранжування сервісних центрів і вибору найближчого виконавця, а картографічна візуалізація може бути реалізована за допомогою бібліотеки Leaflet [4].

Функціональні особливості blockchain-модуля порівняно з класичною моделлю оплати узагальнено в табл. 1.

Табл. 1. Порівняння класичної та blockchain-моделі сервісного платежу

Компонент	**Класичний платіж**	**Blockchain-платіж**
Ідентифікація операції	внутрішній ID або номер інвойсу	txHash у мережі блокчейну
Перевірка факту оплати	через платіжний шлюз або БД	через блокчейн-вузол і підтвердження
Прозорість для користувача	обмежена межами сервісу	незалежна зовнішня перевірка
Цифровий слід операції	запис у внутрішній системі	подія в системі та дані блокчейну

Висновки. У результаті дослідження встановлено, що blockchain-механізми доцільно інтегрувати в платформу AutoAssist+ як інструмент перевірної фіксації сервісних платежів. Використання txHash, підтверджень мережі та підпису повідомлень підвищує довіру до фінансової взаємодії, тоді як вебплатформа зберігає функції створення заявки, геолокаційного підбору сервісного центру та керування документами. Подальший розвиток системи може бути пов’язаний із розширенням сценаріїв роботи з цифровими

підтвердженнями, інтеграцією з іншими blockchain-мережами та вдосконаленням механізмів аудиту платежів [1]–[4].

Список використаних джерел

1. Ethers.js Documentation – [Електронний ресурс] – Режим доступу : https://docs.ethers.org/.
2. Ethereum.org. Introduction to blockchain and decentralized applications – [Електронний ресурс] – Режим доступу : https://ethereum.org/.
3. Sign-In with Ethereum (SIWE) – [Електронний ресурс] – Режим доступу : https://siwe.xyz/.
4. Leaflet Documentation – [Електронний ресурс] – Режим доступу : https://leafletjs.com/reference.html.

Виливок Максим Володимирович
студент 4 курсу, групи ТЦР-44
Державний університет інформаційно-комунікаційних технологій
(050)772-30-12
vylyvokmaxsim@gmail.com

ВИКОРИСТАННЯ БІБЛІОТЕКИ OPENZEPPELIN ДЛЯ РОЗРОБКИ ТОКЕНА СТАНДАРТУ ERC-20

Постановка задачі. У сучасних блокчейн-системах вагому роль відіграє безпечна та стандартизована реалізація смарт-контрактів. Одним із найбільш уживаних стандартів є ERC-20, котрий застосовується для творення токенів у мережі Ethereum. Для спрощення розробки та підняття безпеки використовуються спеціалізовані бібліотеки, зокрема OpenZeppelin.

Мета дослідження. Метою дослідження є аналіз можливостей OpenZeppelin та її використання для реалізацій безпечного смарт-контаркта, масштабованого і функціонального ERC-20 токена в мережі Ethereum

Результати дослідження. Для розробки стандарту ERC-20 у мережі Ethereum було використано бібліотеку OpenZeppelin, яка надає готові реалізації смарт-контрактів та є одним із найпоширеніших інструментів для створення безпечних рішень бля блокчейн систем.

Реалізований токен базується на архітектурі наслідування, де основною виступає система ERC-20, що дає змогу використовувати весь базовий функціонал, зокрема передачу токенів, облік балансів та механізм дозволів. Взаємодія з токеном здійснюється за допомогою функції transfer, approve, transferFrom та balanceOf, що відповідають стандарту ERC-20.

Для розширення функціональності було використано модулі ERC20Permit та ERC20Votes,які входять до бібліотеки OpenZeppelin. Модуль ERC20Permit дозволяє встановлювати дає можливість на використання токенів за допомогою криптографічного підпису без використання окремої транзакції, що зменшує витрати газу. Модуль ERC20Votes забезпечує реалізацію механізмів делегування голосів та підтримку систем управління (governance), що дозволяє використовувати токен не тільки як платіжний інструмент, а і як засіб участі у прийняті рішень.

Таким чином, використання бібліотеки OpenZeppelin дозволяє створити безпечний, масштабований та функціонально розширений токен, який відає сучасним вимогам децентралізованих систем.



Рис. 1. Модель інтеграції розширень ERC20Permit та ERC20Votes у базовий контракт ERC20 (OpenZeppelin)

Висновок та перспективи. Створення токен стандарту ERC-20 із застосування бібліотеки OpenZeppelin забезпечує безпечну та дієву реалізацію цифрових активів у мережі Etherem. Використання модулів ERC20Permit та ERC20Votes дає змогу розширити можливості токена, поліпшити зручність його застосування та впровадити механізми керування.

Потенціалом є з'єднання токена з dApp та DeFi-системами, а також вдосконалення роботи смарт-контракту та розширення його змісту.

Список використаних джерел

1. Ethereum Foundation. ERC-20 Token Standard (EIP-20). – Режим доступу: https://eips.ethereum.org/EIPS/eip-20

2. Ethereum Foundation. ERC-2612: Permit Extension for ERC-20 (EIP-2612). – Режим доступу: https://eips.ethereum.org/EIPS/eip-2612

3. OpenZeppelin. OpenZeppelin Contracts Documentation. – Режим доступу: https://docs.openzeppelin.com/contracts

4. OpenZeppelin. ERC20 Token Implementation. – Режим доступу: https://docs.openzeppelin.com/contracts/5.x/api/token/erc20

5. Wood G. Ethereum: A Secure Decentralised Generalised Transaction Ledger (Yellow Paper). – 2014.

Чорнобривець Дмитро Віталійович,
студент 4 курсу, групи ІП-21
спеціальності 121 «Інженерія програмної забезпечення»
Національного технічного університету України
"Київський політехнічний інститут імені Ігоря Сікорського"
dmitryfraire17@gmail.com
Науковий керівник: Поперешняк Світлана Володимирівна,
кандидат фізико-математичних наук,
доцент кафедри Інформатики та програмної інженерії
Національного технічного університету України
"Київський політехнічний інститут імені Ігоря Сікорського"
м.Київ

ПАТЕРНИ ПРОЄКТУВАННЯ БАГАТОПОТОКОВОЇ АРХІТЕКТУРИ ДЛЯ ДЕСКТОПНИХ ДОДАТКІВ У СФЕРІ КОМП'ЮТЕРНОГО ЗОРУ

Розробка сучасних інформаційних систем, що інтегрують моделі глибокого навчання , вимагає особливого підходу до проєктування інтерфейсу користувача. Процес інференсу нейронних мереж, таких як YOLO, є ресурсомісткою операцією. Синхронне виконання обчислювальних задач та рендерингу графічного інтерфейсу в єдиному головному потоці (Main Thread) призводить до блокування циклу подій (Event Loop), що викликає ефект «зависання» додатку та погіршує користувацький досвід [1].

Постановка задачі. Задачею дослідження є розробка потокобезпечної (thread-safe) програмної архітектури для системи відеомоніторингу паркувальних місць, що забезпечить паралельне виконання захоплення відеокадрів, детекції об'єктів та оновлення графічного інтерфейсу без втрати продуктивності.

Мета дослідження. Дослідження та імплементація архітектурного патерну асинхронної обробки даних у середовищі Python за допомогою фреймворку Qt (PySide6)[5] для оптимізації роботи систем комп'ютерного зору реального часу.

Результати дослідження. Для вирішення проблеми блокування інтерфейсу було застосовано архітектурний підхід, заснований на розділенні відповідальності між потоками (Thread Separation). Програмний комплекс було розділено на два ключові рівні: головний потік інтерфейсу, що відповідає виключно за відмальовування компонентів та реакцію на дії користувача, та фоновий робочий потік, призначений для ресурсомістких обчислень.[2]

Основою реалізації фонового процесу став механізм потоків фреймворку Qt. У межах робочого потоку інкапсульовано повний конвеєр аналітичної обробки: захоплення відеокадрів засобами бібліотеки OpenCV, передача кадрів на вхід нейромережі YOLO для детекції транспортних засобів, просторовий аналіз результатів детекції відносно заданих зон інтересу (ROI) та візуалізація результатів. Теоретичне прискорення від такої паралелізації обмежується законом Амдаля [4]:

$$S(n) = \frac{1}{(1-p)+\frac{p}{n}},$$

де S(n) – коефіцієнт прискорення при використанні n потоків, p – частка коду, що піддається паралелізації. У розробленій системі частка обчислень, винесених у фоновий потік, становить p ≈ 0.85. Підставляючи значення у формулу (1) для двох потоків (n = 2), отримуємо S(2) ≈ 1.74, що підтверджує доцільність розділення обчислювальної та інтерфейсної логіки.

Критичним аспектом багатопотокового програмування є безпечний обмін даними між потоками. Пряме звернення з робочого потоку до віджетів графічного інтерфейсу заборонено архітектурою Qt та призводить до невизначеної поведінки програми. Для вирішення цієї проблеми було імплементовано механізм сигналів та слотів (Signals and Slots)[3], що реалізує патерн «Спостерігач» (Observer)[6]. Після завершення обробки чергового кадру робочий потік генерує сигнал, передаючи розмічений кадр та поточну статистику зайнятості паркомісць. Потік інтерфейсу, зареєстрований як слухач цього сигналу, асинхронно приймає дані та оновлює візуальні компоненти. Такий підхід забезпечує повну ізоляцію потоків та гарантує, що операції відмальовування виконуються виключно у контексті головного потоку.

Загальна затримка відображення одного оброблюваного кадру визначається як сума компонентів конвеєра:

$$T = T_{\text{захв}} + T_{\text{інф}} + T_{\text{рендер}} + T_{\text{передача}},$$

де $T_{\text{захв}}$– час захоплення кадру з відеопотоку (~2 мс), $T_{\text{інф}}$– час інференсу нейромережі (~25–30 мс при використанні GPU), $T_{\text{рендер}}$ – час накладання графічних міток та статусних полігонів (~3 мс), $T_{\text{передача}}$ – латентність міжпотокової передачі через чергу подій (~0.1 мс). У однопотоковій архітектурі сумарна затримка T ≈ 35 мс блокує цикл подій інтерфейсу на кожному кадрі, роблячи додаток нечутливим до дій користувача. У запропонованій двопотоковій моделі потік інтерфейсу залишається вільним протягом усього часу обробки кадру.

Структуру взаємодії потоків у розробленій системі наведено на рис. 1.

Окремою задачею стало забезпечення узгодженості доступу до спільних даних. Під час роботи аналітичного конвеєра оператор може змінювати конфігурацію зон паркування через інтерактивний редактор. Одночасний доступ двох потоків до одних і тих самих структур даних без синхронізації спричиняє стан гонки (Race Condition). Для його запобігання було впроваджено механізм взаємного виключення , що гарантує атомарність операцій читання та запису спільних ресурсів.

Висновки та перспективи. Впровадження багатопотокової архітектури дозволило створити стабільний та чуйний десктопний додаток для відеоаналітики. Ізоляція процесу інференсу нейромережі у фоновому потоці усунула проблему блокування інтерфейсу. Експериментальні вимірювання показали, що частота обробки кадрів, визначена як FPS = N / T_{elapsed}, у двопотоковій архітектурі становить стабільних ~31.2 кадрів на секунду при

обробці відеопотоку роздільності 1920x1080, тоді як однопотоковий варіант забезпечував лише 8–12 кадрів на секунду. Механізм сигналів і слотів довів свою ефективність як надійний інструмент міжпотокової комунікації, що реалізує патерн «Спостерігач». У подальшому запропонований підхід може бути розширений для підтримки декількох паралельних робочих потоків при обробці відеопотоків з багатьох камер одночасно.

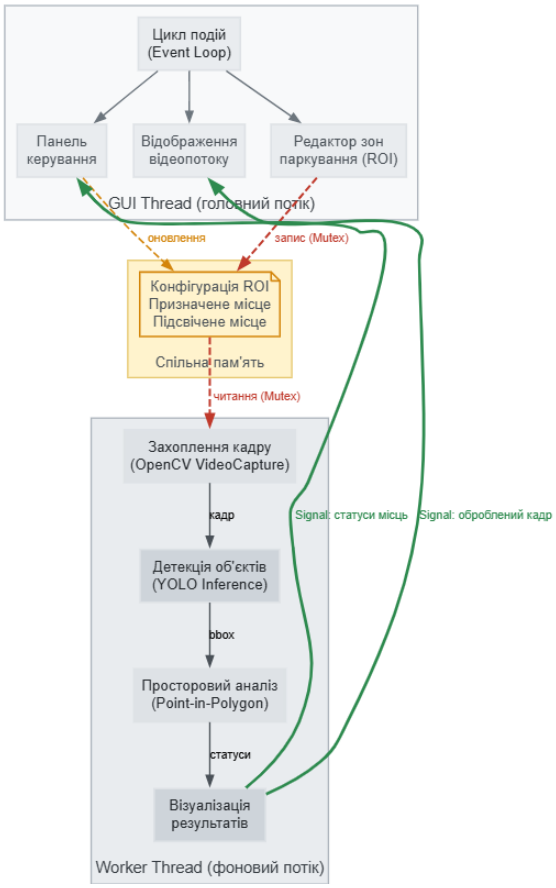


Рис. 1. Багатопотокова архітектура системи з механізмом сигналів та слотів

Список використаних джерел

1. Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*. Las Vegas, 2016. P. 771–788. https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf
2. Чорнобривець, Д. В., Поперешняк, С. В. (2024). Система моніторингу місць для паркування з використанням комп'ютерного зору. *Телекомунікаційні та інформаційні технології*, 4(85), 62-72.
3. Livesu M. et al. From 3D models to 3D prints: An overview of the processing pipeline. *Comput. Graph. Forum*. 2017. Vol. 36, no. 2. P. 537–564.
4. Amdahl G. M. Validity of the single processor approach to achieving large scale computing capabilities. *Proceedings of the AFIPS Spring Joint Computer*

Conference. Atlantic City, 1967. P. 483–485. DOI: https://doi.org/10.1145/1465482.1465560
5. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 416 p.
6. PySide6 Documentation. Threading Basics [Електронний ресурс]. Qt for Python. URL: https://doc.qt.io/qtforpython-6/overviews/thread-basics.html

Гавор Артур Станіславович
ст. викладач кафедри ТЦР
Державний університет інформаційно-комунікаційних технологій
(050) 285-15-30
a.havor@duikt.edu.ua

ГІБРИДНА АРХІТЕКТУРА EDGE-CHAIN ДЛЯ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ ЕКОЛОГІЧНОГО МОНІТОРИНГУ В УМОВАХ 6G-МЕРЕЖ

Постановка задачі. Стрімкий розвиток систем моніторингу довкілля вимагає впровадження сенсорних мереж надвисокої щільності. Хоча інфраструктура 6G здатна підтримувати такий обсяг підключень [1], пряма передача не відфільтрованої телеметрії до смарт-контрактів «Direct-to-Chain» викликає миттєве переповнення трансакційних пулів та експоненційне зростання мережевих комісій. Більше того, традиційне використання централізованих хмарних серверів «Cloud-Centric» для попередньої агрегації супроводжується відчутними магістральними затримками і не захищає блокчейн від фіксації скомпрометованих даних «сміття на вході, сміття на виході» GIGO [2]. Відповідно, виникає гостра необхідність у розробці рішень, здатних фільтрувати та консолідувати телеметрію безпосередньо на найближчих до сенсорів обчислювальних вузлах.

Мета дослідження. Метою дослідження є створення та верифікація нового методу алгоритмічної обробки даних «Edge-Chain», який інтегрує можливості крайових обчислень 6G з децентралізованими реєстрами [3] для гарантування еталонної цілісності показників екологічного моніторингу за умови мінімізації наскрізних затримок.

Результати дослідження. Для оптимізації інформаційних потоків розроблено децентралізовану багаторівневу систему, що базується на парадигмі Compute-First Networking. Замість трансляції масивів сирої телеметрії до віддаленої хмари або напряму в розподілений реєстр, основне обчислювальне навантаження делегується крайовому серверу мобільної мережі MEC [4]. Структурну схему запропонованого рішення наведено на рис. 1.

Конвеєр локальної обробки розділено на три ключові етапи. На першому кроці вхідний масив від IoT-пристроїв очищається від апаратних шумів за допомогою багатовимірного фільтра Махаланобіса [5]. Далі в дію вступає алгоритм довірчого зважування, функція якого полягає в ізоляції векторів візантійських атак [6] та обчисленні істинної глобальної медіани параметрів довкілля. Зрештою, застосовується метод порогової криптографії (TSS), що формує єдиний доказ достовірності для подальшої фіксації у смарт-контракті.

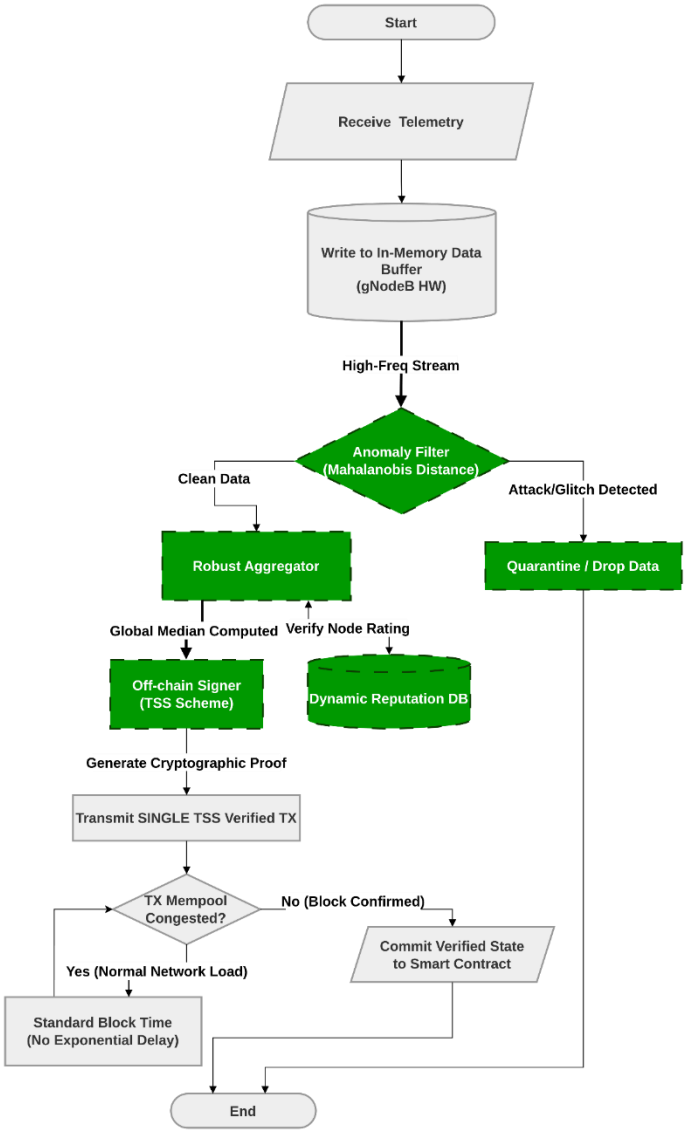


Рис. 1. Архітектура системи Edge-Chain.

Розроблене симуляційне середовище, де працювали 1000 сенсорів, 30% з яких імітували візантійську поведінку, підтвердило життєздатність концепції. Зафіксовано зменшення часу наскрізної доставки даних до 0.5 мс, що відповідає строгим стандартам URLLC [7], і радикально перевершує показники хмарних систем.Крім того, завдяки відправці лише однієї консолідованої транзакції, фінансові витрати набули константного характеру, а ризик запису «отруєних» даних був алгоритмічно нівельований.

Висновки та перспективи.

У ході виконання роботи доведено, що делегування процесів агрегації та криптографічного підпису на рівень 6G MEC є найбільш ефективним шляхом для масштабування IoT-мереж. Запропонований підхід усуває першопричину перевантажень мемпулу та гарантує бездоганну цілісність екологічних метрик навіть в умовах масованих кіберзагроз.

Подальший розвиток даної теми передбачає апаратну реалізацію створених консенсусних механізмів на базі FPGA-матриць для їхнього прямого вбудовування в телекомунікаційне обладнання майбутніх поколінь зв'язку.

Список використаних джерел

1. The Role of Edge Computing in 6G-Enabled IoT Applications / R. Imamguluyev et al. International Journal of Research Publication and Reviews. 2024. Vol. 5, no. 12. P. 2007–2013. URL: https://doi.org/10.55248/gengpi.5.1224.3523

2. Integration of Cloud computing and Internet of Things: A survey / A. Botta et al. Future Generation Computer Systems. 2016. Vol. 56. P. 684–700. URL: https://doi.org/10.1016/j.future.2015.09.021

3. A Lightweight Blockchain Architecture with Smart Collaborative and Progressive Evolution for Privacy-Preserving 6G IoT / D. Han et al. IEEE Wireless Communications. 2024. P. 1–7. URL: https://doi.org/10.1109/mwc.013.2300491

4. R. V. Durai and M. Dhivyashree, “Deployments of Aerial Aided Edge Computing in Real-World Scenarios”, 2024. URL: https://doi.org/10.1002/9781394336326.ch8

5. H. M. Ngo, D. H. Ha, Q. D. Pham, T. V. Le, and S. H. Nguyen, “Detecting Anomaly in Smart Homes Based on Mahalanobis Distance,” in ICC 2024 IEEE International Conference on Communications, Jun. 2024, pp. 2204–2209.

6. The Deferred Byzantine Generals Problem / X. Liu et al. IEEE Transactions on Information Forensics and Security. 2025. P. 1. URL: https://doi.org/10.1109/tifs.2025.3592566

7. Bennis M., Debbah M., Poor H. V. Ultrareliable and Low-Latency Wireless Communication: Tail, Risk, and Scale. Proceedings of the IEEE. 2018. Vol. 106, no. 10. P. 1834–1853. URL: https://doi.org/10.1109/jproc.2018.2867029

Гордієнко Катерина Олександрівна
аспірантка 1 курсу, групи АІПЗ-11,
старший викладач кафедри ТЦР
Державний університет інформаційно-комунікаційних технологій
(050) 770-55-05
k.hordienko@duikt.edu.ua

БЛОКЧЕЙН-ПІДХІД ДО ФОРМУВАННЯ ІСТОРІЇ СПОСТЕРЕЖЕНЬ ЗА ДЕФЕКТАМИ ДОРОЖНЬОГО ПОКРИТТЯ НА ОСНОВІ ВІДЕОДАНИХ ТРАНСПОРТНИХ ЗАСОБІВ

Постановка задачі. Проблема формування історії спостережень за дефектами дорожнього покриття полягає не лише у виявленні ям, тріщин чи просідань, а й у забезпеченні перевірюваності, незмінності та простежуваності даних про кожне пошкодження. У попередніх дослідженнях було розглянуто системи аналізу та класифікації пошкоджень дорожнього покриття, а також обґрунтовано доцільність розробки інтелектуальної системи просторової класифікації дефектів на основі комп'ютерного зору [1; 2]. Сучасні підходи до моніторингу доріг підтверджують можливість використання відеоданих зі смартфонів, бортових камер та інших транспортних засобів для автоматичного виявлення й просторово-часової локалізації дефектів [3]. Водночас безпосереднє збереження відеоматеріалів у блокчейні є недоцільним через значний обсяг даних, потребу в керованому доступі та необхідність подальшої перевірки цілісності первинних матеріалів. Тому актуальним є визначення такої загальної схеми роботи системи, за якої відео та повні доказові матеріали зберігаються поза ланцюгом блоків, а блокчейн використовується як незмінний журнал подій життєвого циклу дефекту: його виявлення, повторного спостереження, зміни стану, ремонту або усунення [4; 5].

Мета дослідження. Мета дослідження полягає в обґрунтуванні загальної схеми використання блокчейну для формування історії спостережень за дефектами дорожнього покриття на основі відеоданих транспортних засобів, а також у визначенні складу даних, які мають зберігатися у блокчейні та позаланцюговому сховищі.

Окрему увагу приділено питанню фіксації зміни стану дефекту, зокрема його ремонту або зникнення. Механізм досягнення узгодженості між учасниками системи виходить за рамки дослідження.

Результати дослідження. Запропонована схема передбачає розділення даних на два рівні:

1. На першому рівні формується база спостережень, до якої потрапляють результати автоматичного аналізу відео: тип і локалізація дефекту, часові мітки, фрагменти відео, окремі кадри, службові дані обробки та додаткова інформація, необхідна для повторної перевірки.

2. На другому рівні використовується блокчейн, у якому фіксуються лише перевірені події, пов'язані з дефектом.

Загальна схему роботи системи зображено на рис. 1.

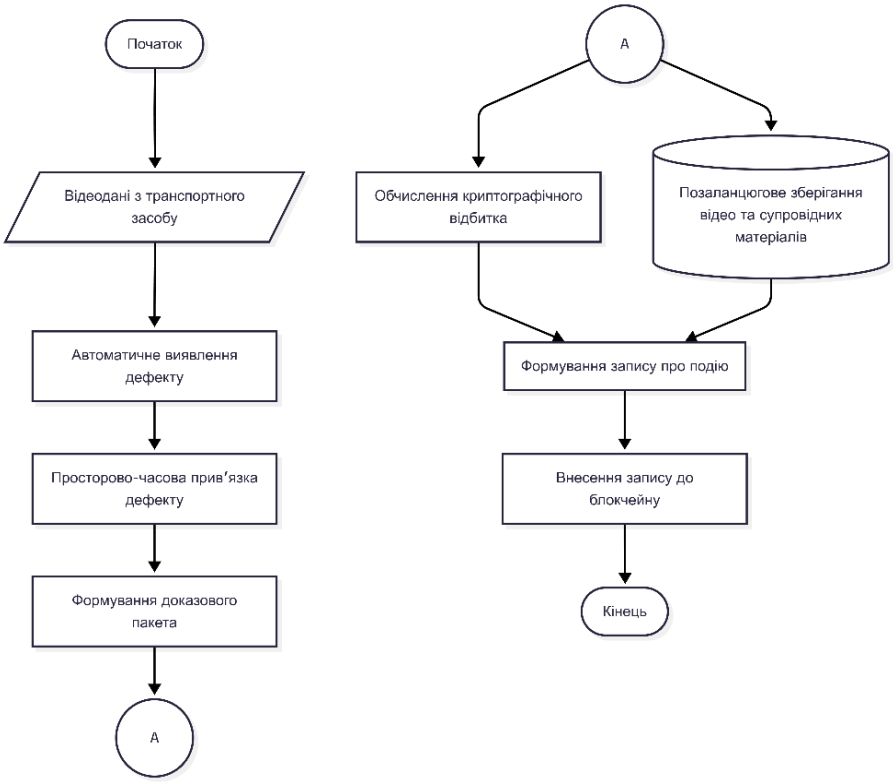


Рис. 1 Загальна схема роботи пропонованої системи

На рис.2 зображено структуру зберігання даних. У такій моделі історія дефекту формується не шляхом редагування попереднього запису, а через послідовне додавання нових подій.



Рис. 2 Структура запису про подію дефекту в блокчейні

Схема станів показана кінцевим автоматом на рис. 3.

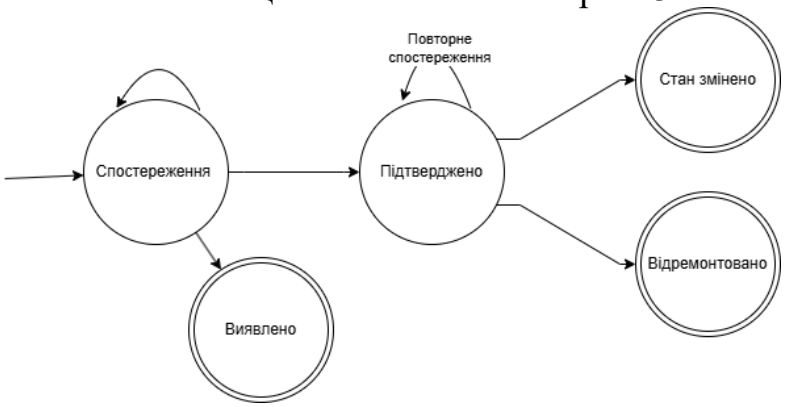


Рис. 3 Граф станів дефекту дорожнього покриття у блокчейн-журналі подій.

Висновки та перспективи. Обґрунтовано доцільність використання блокчейну як незмінного журналу ключових подій життєвого циклу дефекту дорожнього покриття. Запропонована модель передбачає зберігання відеоданих поза блокчейном, а в реєстрі — лише стислих записів із криптографічними відбитками та посиланнями на попередні події. Це забезпечує доказовість, простежуваність і перевірюваність історії спостережень без редагування вже створених записів.

Список використаних джерел

1. Гордієнко К. О. Дослідження систем аналізу та класифікацій пошкоджень дорожнього покриття. Проблеми та інновації в математичній, цифровій, природничій і професійній освіті : зб. матеріалів XX-ї Міжнар. науково-практ. інтернет конф., м. Кропивницький, 23 жовт. 2025 р. Кропивницький, 2025. С. 203–204. URL: https://www.ldftpo.kr.ua/wp-content/uploads/2025/12/Tezu-XX-konf-2025.pdf (дата звернення: 18.04.2026).
2. Гордієнко К. О. Доцільність розробки інтелектуальної системи аналізу та просторової класифікації дорожнього покриття на основі комп'ютерного зору. III міжнародна науково-практична конференція «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії» : матеріали III Міжнар. науково-практ. конф., м. Київ, 4–6 груд. 2025 р. Київ, 2025. С. 111–113. URL: https://duikt.edu.ua/uploads/p_2779_50068458.pdf (дата звернення: 18.04.2026).
3. Kim G., Kim S. A road defect detection system using smartphones. Sensors. 2024. Т. 24, № 7. С. 2099. URL: https://doi.org/10.3390/s24072099 (дата звернення: 20.04.2026).
4. Lightweight deep learning for real-time road distress detection on mobile devices / Y. Hu та ін. *Nature communications*. 2025. Т. 16, № 1. URL: https://doi.org/10.1038/s41467-025-59516-5 (дата звернення: 20.04.2026).
5. ForensiBlock: a provenance-driven blockchain framework for data forensics and auditability / A. J. Akbarfam та ін. *2023 5th IEEE international conference on trust, privacy and security in intelligent systems and applications (TPS-ISA)*, м. Atlanta, GA, USA, 1–4 листоп. 2023 р. 2023. URL: https://doi.org/10.1109/tps-isa58951.2023.00025 (дата звернення: 20.04.2026).

Напрям 9 ЦИФРОВА ЕКОНОМІКА ТА ЕЛЕКТРОННЕ ВРЯДУВАННЯ

Бондаренко Дарина Вадимівна
студентка 4 курсу, групи ТЦР-42
Державний університет інформаційно-комунікаційних технологій
(097) 533-08-82
Dbondv2601@gmail.com

ВПРОВАДЖЕННЯ СИСТЕМИ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ В СУЧАСНИХ ОРГАНІЗАЦІЯХ

Постановка задачі.

В умовах сьогодення, коли в Україні стрімко розвиваються цифрові технології, паперові методи роботи з документами поступово втрачають свою ефективність. Вони потребують значних витрат на друк і зберігання, а також ускладнюють доступ до інформації. Сам процес погодження документів дуже часто затягується, особливо коли до залучені кілька підрозділів або відповідальних осіб.

Подібні затримки впливають на швидкість прийняття рішень і загальну організацію роботи. Найчастіше це проявляється під час обробки наказів, договорів або внутрішніх службових документів, які готуються та погоджуються значною кількістю працівників.

Саме тому наразі зростає інтерес до систем електронного документообігу (СЕД), які дозволяють оптимізувати процеси обробки документів, надати функціональні можливості дистанційної роботи працівників з документами. Проте впровадження таких систем потребує не лише технічних рішень, а й адаптації внутрішніх процесів та підготовки персоналу. Не рідко і сам персонал на етапі впровадження може негативно сприймати нові зміни в уже звичних робочих процесах.

У рамках дослідження розглядаються переваги використання СЕД, можливі ризики впровадження, а також їх вплив на організацію роботи та документообіг в організаціях.

Мета дослідження.

Метою даної роботи є обґрунтування доцільності переходу на електронний документообіг та оцінка того, наскільки впровадження СЕД впливає на робочі процеси в організації. Розглянути переваги автоматизації документообігу, зокрема з точки зору економії часу та ресурсів. Також виконати аналіз змін, що відбуваються в організації роботи після впровадження СЕД і які можливості відкриває взаємодія СЕД з іншими інформаційними рішеннями.

Результати дослідження.

У процесі роботи стало зрозуміло, що електронний документообіг помітно впливає на організацію роботи. Найперше змінюється швидкість обробки

документів: менше часу витрачається на їх створення, погодження та передачу між підрозділами. У результаті рішення приймаються швидше, а взаємодія між працівниками стає більш узгодженою.

Не можна не відзначити скорочення використання паперових документів. Зменшуються витрати на друк і зберігання, а також спрощується робота з архівами. Окрім економії ресурсів, це частково вирішує і питання впливу на довкілля.

У організаціях, де складна структура, різниця відчувається найбільше. Наприклад, робота з наказами, договорами чи внутрішніми службовими документами, яка раніше займала кілька днів, може виконуватись значно швидше, а до того ж і дистанційно.

Доступ до інформації також стає простішим. Документи легко знаходити за ключовими словами або фільтрами, що зменшує час пошуку і полегшує повсякденну роботу.

Важливою складовою залишається безпека даних. На практиці використовуються механізми розмежування доступу, автентифікації користувачів та кваліфікованого електронного підпису. Це дає змогу контролювати роботу з документами і підтверджувати їх достовірність та юридичну силу.

Крім самих внутрішніх процесів електронного документообігу СЕД можуть взаємодіяти з іншими інформаційними системами. За рахунок інформаційної електронної взаємодії формується єдине середовище, у якому дані не дублюються і доступні для різних підрозділів, відкриваються можливості автоматизації багатьох повсякденних процесів.

Але впровадження таких систем потребує часу. Зазвичай воно проходить поетапно: від аналізу існуючих процесів до навчання персоналу та адаптації роботи під новий формат.

Висновок.

Перехід від паперового до електронного документообігу слід розглядати як логічний крок у розвитку сучасних організацій в умовах сьогодення. На практиці такий підхід додає швидкості в обробці документів, зменшує витрати та спрощує робочі процеси. Крім того, підвищується контроль за інформацією і знижується ризик її втрати.

Результат значною мірою залежить від того, як саме буде організовано впровадження. Без підготовки персоналу та перегляду внутрішніх робочих процесів та усталених практик навіть сучасна система не дасть очікуваного ефекту.

У підсумку електронний документообіг варто сприймати не лише як технічне рішення, а як зміну підходу до роботи з документами. Саме від цього залежить, наскільки ефективно організація зможе адаптуватися до сучасних умов в тому числі і організації віддаленої роботи співробітників.

Список використаних джерел

1. Laudon K. C., Laudon J. P. *Management Information Systems: Managing the Digital Firm*. Pearson. 2020. Vol. 16. P. 1–720.

2. Turban E., Pollard C., Wood G. *Information Technology for Management*. Wiley. 2018. Vol. 11. P. 1–560.

3. Stair R., Reynolds G. *Principles of Information Systems*. Cengage Learning. 2019. Vol. 13. P. 1–600.

4.Закон України *«Про електронні документи та електронний документообіг»* (№ 851-IV від 22.05.2003)

Фесенко Андрій Олегович
студент 4 курсу, групи ТЦР-43
Державний університет інформаційно-комунікаційних технологій
st7988269@stud.duikt.edu.ua
Керівник: Поперешняк Світлана Володимирівна
к.ф.-м.н., доцент, доцент кафедри інформатики та програмної інженерії
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
spopereshnyak@gmail.com

ІНТЕЛЕКТУАЛЬНИЙ ЦИФРОВИЙ АСИСТЕНТ ЯК ІНСТРУМЕНТ РОЗВИТКУ ЦИФРОВОЇ ЕКОНОМІКИ ТА ЕЛЕКТРОННОГО ВРЯДУВАННЯ

Постановка задачі. Розвиток цифрової економіки супроводжується впровадженням ІКТ у фінансову сферу та електронне врядування, що підвищує значення ефективного управління особистими фінансами[1]. Водночас низька фінансова грамотність і відсутність персоналізованих сервісів ускладнюють прийняття обґрунтованих рішень, що негативно впливає на економіку. Використання інтелектуальних цифрових асистентів на основі машинного навчання сприяє підвищенню фінансової культури, прозорості операцій та розвитку електронного врядування[2].

Мета дослідження. Метою дослідження є обґрунтування ролі інтелектуальних цифрових асистентів у розвитку цифрової економіки та електронного врядування, а також розробка концепції програмної системи для управління особистими фінансами громадян.

Результати дослідження.

У роботі запропоновано концепцію інтелектуального цифрового асистента як елементу цифрової економіки, що забезпечує автоматизацію фінансового аналізу та підтримку прийняття рішень користувачами[3].

Функціональна структура системи включає:

1) модуль збору фінансових даних (банківські транзакції, витрати, доходи);
2) модуль обробки та аналізу даних;
3) модуль машинного навчання;
4) модуль формування рекомендацій;
5) інтерфейс взаємодії з користувачем.

Застосування цифрового асистента в умовах електронного врядування дозволяє:

1) підвищити прозорість фінансової діяльності громадян;
2) сприяти формуванню відповідальної фінансової поведінки;
3) забезпечити інтеграцію з державними електронними сервісами;
4) покращити доступ до фінансових та адміністративних послуг.

Інтеграція з платформами електронного врядування відкриває значно ширші можливості для автоматизації взаємодії громадян із державними структурами,

зокрема у сферах оподаткування, соціальних виплат, фінансового контролю та адміністративних послуг. Завдяки такій інтеграції користувач може в режимі реального часу отримувати актуальну інформацію про податкові зобов'язання, статус заяв на соціальну допомогу, штрафи або інші державні платежі, а також оперативно здійснювати необхідні фінансові операції без відвідування установ.

Таблиця 1.

Технологічна основа системи

Технологія	**Призначення**	**Значення для цифрової економіки**
Машинне навчання	Аналіз і прогнозування витрат	Підвищення ефективності рішень
Big Data	Обробка великих обсягів даних	Масштабованість сервісів
Хмарні технології	Зберігання та обчислення	Доступність та мобільність
API-інтеграція	Взаємодія з сервісами	Інтеграція з e-government

Крім того, використання цифрового асистента дозволяє автоматизувати рутинні процеси, такі як нагадування про строки сплати податків, формування фінансової звітності або аналіз державних виплат. Це зменшує ризик помилок, підвищує прозорість фінансових операцій і сприяє більш ефективному управлінню особистими фінансами.

Таким чином, цифровий асистент виступає не лише інструментом персонального фінансового менеджменту, але й важливим елементом цифрової інфраструктури держави. Він сприяє розвитку електронного врядування, підвищує рівень цифрової грамотності населення та забезпечує більш тісну взаємодію між громадянами і державними сервісами на основі автоматизації та штучного інтелекту.[3]

Висновки та перспективи. У дослідженні доведено, що використання інтелектуальних цифрових асистентів є важливим фактором розвитку цифрової економіки та електронного врядування. Запропонований підхід дозволяє підвищити рівень фінансової грамотності населення, покращити якість прийняття фінансових рішень, а також забезпечити більш глибоку інтеграцію громадян у цифрову економіку через автоматизацію взаємодії з фінансовими та державними сервісами.

Перспективами подальших досліджень є розширення функціональності таких систем за рахунок інтеграції з державними інформаційними системами, що дозволить забезпечити більш повну та оперативну обробку даних. Окрему увагу доцільно приділити використанню технологій штучного інтелекту для аналізу макроекономічних показників і прогнозування фінансових тенденцій, що підвищить аналітичну цінність цифрових асистентів. Також перспективним напрямом є впровадження блокчейн-рішень для забезпечення підвищеного рівня безпеки, прозорості та незмінності фінансових даних, а також розвиток персоналізованих цифрових сервісів у сфері електронного врядування, орієнтованих на індивідуальні потреби користувачів.

Список використаних джерел

1) OECD. *Digital Economy Outlook 2024*. (розділ 2.1–2.3: Digital transformation and services)

2) United Nations. *E-Government Survey 2022*. розділ 3: Digital Government Services)

3) Goodfellow I., Bengio Y., Courville A. *Deep Learning*. MIT Press, 2021. (Chapter 4: Machine Learning Applications)

Напрям 10 ІННОВАЦІЇ У СФЕРІ ЦИФРОВОГО РОЗВИТКУ

Щербакова Катерина Євгенівна
студентка групи ІСД-41
Державного університету
інформаційно-комунікаційних технологій
(096)-825-80-67
scherbakova.katia@gmail.com

АНАЛІТИЧНІ ІНСТРУМЕНТИ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ МАЛОГО БІЗНЕСУ

Постановка задачі

В умовах цифровізації економіки малий бізнес потребує оперативного прийняття управлінських рішень на основі достовірних даних. Проте значна частина малих підприємств досі використовує застарілі або фрагментарні підходи до обліку та аналізу діяльності, що ускладнює контроль фінансових показників, оцінку продуктивності працівників, аналіз продажів і прогнозування результатів роботи. Відсутність єдиної аналітичної системи призводить до зниження ефективності управління та втрати конкурентних переваг. У зв'язку з цим виникає задача дослідження сучасних аналітичних інструментів та визначення доцільності їх впровадження у діяльність малого бізнесу з метою підвищення ефективності управління підприємством.

Мета дослідження

Метою дослідження є аналіз сучасних аналітичних інструментів для малого бізнесу, оцінка їх впливу на ефективність управлінських процесів, а також визначення практичних рекомендацій щодо впровадження цифрових рішень для підтримки прийняття управлінських рішень.

Результати дослідження

У ході дослідження проведено системний аналіз основних аналітичних інструментів, що використовуються у сфері малого бізнесу, зокрема CRM-систем, BI-платформ, інформаційних панелей (dashboard), систем автоматизованої звітності та сервісів моніторингу ключових показників ефективності (KPI). Встановлено, що використання цифрових аналітичних рішень дозволяє підвищити швидкість обробки інформації, зменшити кількість помилок у звітності та покращити якість управлінських рішень.

Порівняльний аналіз показав, що CRM-системи найбільш ефективні для контролю взаємодії з клієнтами та управління продажами, BI-платформи — для комплексного аналізу фінансових і операційних показників, а dashboard-рішення — для оперативного моніторингу діяльності підприємства в режимі реального часу. Також встановлено, що впровадження аналітичних інструментів сприяє

оптимізації витрат, зростанню продуктивності праці та підвищенню конкурентоспроможності малого бізнесу.

На основі отриманих результатів сформовано узагальнені рекомендації щодо вибору аналітичних інструментів залежно від масштабу підприємства, сфери діяльності та стратегічних цілей розвитку.

Висновки та перспективи

У результаті дослідження підтверджено, що застосування сучасних аналітичних інструментів є важливим чинником підвищення ефективності діяльності малого бізнесу. Використання цифрових систем аналізу даних забезпечує оперативність управління, покращує точність прогнозування та сприяє прийняттю обґрунтованих управлінських рішень.

Перспективи подальших досліджень полягають у розробці інтегрованих інформаційних систем для малого бізнесу, які поєднуватимуть функції управління проєктами, аналітики та автоматизованої звітності в єдиному цифровому середовищі.

Список використаних джерел

1. OECD The Digital Transformation of SMEs. Paris: OECD Publishing, 2021. URL: https://www.oecd.org/industry/smes/ (дата звернення: 24.04.2026).
2. European Commission Annual Report on European SMEs 2023/2024. Brussels, 2024. URL: https://single-market-economy.ec.europa.eu/smes_en (дата звернення: 24.04.2026).
3. Microsoft Power BI Adoption and Business Analytics Impact Report. Redmond, 2023. URL: https://powerbi.microsoft.com/ (дата звернення: 24.04.2026).
4. World Bank Small Business Digitalization and Productivity Growth. Washington, 2022. URL: https://www.worldbank.org/ (дата звернення: 24.04.2026).

Маковець Владислав Валерійович,
студент 4 курсу, групи ПД-41,
Державного університету інформаційно-комунікаційних технологій
(098)362-88-86
st5387147@stud.duikt.edu.ua

СИСТЕМА РАННЬОГО ВИЯВЛЕННЯ ТРИВОЖНИХ РОЗЛАДІВ НА ОСНОВІ ЕМОЦІЙНОЇ АНАЛІТИКИ ТА СТОРІТЕЛІНГУ

Застосунки з елементами сторітелінгу, емоційної аналітики та адаптивної психологічної підтримки демонструють значний потенціал у сфері раннього виявлення тривожних розладів і реалізації м'яких психотерапевтичних інтервенцій. У сучасних умовах цифрові сервіси стають важливим інструментом підтримки психічного здоров'я, особливо для населення, яке перебуває під впливом тривалого стресу, емоційного виснаження та наслідків психотравмувальних подій. Гейміфіковані застосунки, які формують у користувача відчуття підтримки, безпеки та поступового прогресу, підвищують рівень залученості, мотивації до регулярного використання та загальну ефективність терапевтичного процесу. Особливої актуальності такі рішення набувають в умовах повномасштабної війни, коли значна частина населення України стикається з проявами тривожності, емоційної нестабільності та симптомами посттравматичного стресового розладу. Водночас традиційні методи психологічної допомоги не завжди є доступними через обмеження інфраструктури, нестабільний доступ до мережі або недостатню кількість спеціалістів. Саме тому розробка мобільних систем раннього виявлення тривожних розладів є важливим напрямом сучасних досліджень у сфері цифрового здоров'я.

Постановка задачі. Метою дослідження є створення мобільного застосунку для раннього виявлення тривожних розладів, який не лише фіксує емоційний стан користувача, а й адаптує власний контент — історії, психологічні вправи, рекомендації та нагадування — відповідно до виявлених емоційних патернів і змін поведінки. Застосунок має бути зручним як для військових, так і для цивільних користувачів, підтримувати роботу в офлайн-режимі, не містити тригерного контенту та забезпечувати високий рівень конфіденційності персональних даних. Особливу увагу приділено реалізації позитивного сторітелінгу як терапевтичного інструменту, здатного знижувати рівень психологічного напруження та сприяти стабілізації емоційного стану. Крім того, застосунок орієнтований на виявлення ранніх ознак тривожних розладів шляхом аналізу емоційної динаміки, частоти негативних реакцій, поведінкових змін і взаємодії користувача із цифровим середовищем. Важливим аспектом є створення інтуїтивного користувацького інтерфейсу, який не перевантажує користувача інформацією та відповідає принципам гуманного UX-дизайну. Додатково система має забезпечувати автономність роботи, низьке

енергоспоживання та можливість подальшої інтеграції з державними цифровими платформами охорони здоров'я.

Методи дослідження. У межах розробки застосунку «Стоп тривога» реалізовано систему персоналізованих історій із автоматичним аналізом емоційного тону та механізмами адаптивного контенту, що базуються на підходах когнітивно-поведінкової терапії, елементах емоційної аналітики та принципах user-centered design. Для визначення рівня тривожності та ризику розвитку психологічних розладів застосовувалися алгоритми аналізу емоційного стану користувача на основі текстових записів, поведінкових патернів, частоти використання функцій застосунку та змін у динаміці настрою. Архітектура системи побудована на багатошаровому підході, який включає UI-шар, шар бізнес-логіки, локальне сховище даних і REST API для асинхронної синхронізації із зовнішніми сервісами. Для локального зберігання інформації використано SQLite із підтримкою SQLCipher, що забезпечує захист конфіденційних даних користувача навіть у разі відсутності доступу до мережі. Адаптивний інтерфейс реалізовано відповідно до стандартів Material Design 3, що дозволило забезпечити зручну навігацію та високу доступність функціоналу для різних категорій користувачів. Для перевірки ефективності застосунку проведено польове тестування у ветеранських центрах та серед цивільного населення в регіонах підвищеного ризику. Оцінювання результативності здійснювалося за допомогою психометричних шкал PCL-5 та GAD-7, а також шляхом аналізу поведінкових змін користувачів у процесі взаємодії із застосунком. Отримані результати підтвердили статистично значуще зниження симптомів ПТСР (–18%) і рівня тривожності (–22%), а також покращення емоційної стабільності, якості сну та загального психологічного самопочуття користувачів після регулярного використання системи.

Результати дослідження. Розроблений застосунок забезпечує:

− ведення захищеного щоденника настрою з функцією емоційного аналізу та виявлення ранніх ознак тривожних розладів;

− виконання адаптивних психологічних вправ (дихальні практики, КПТ-завдання, техніки стабілізації емоційного стану);

− автоматичне формування рекомендацій і нагадувань відповідно до поточного психологічного стану користувача;

− використання позитивного сторітелінгу для зменшення рівня стресу, емоційного виснаження та психоемоційного напруження;

− взаємодію із SOS-функціоналом для забезпечення екстреної психологічної підтримки;

− роботу в офлайн-режимі з можливістю безпечної синхронізації даних після відновлення доступу до мережі;

− накопичення та аналіз історії емоційного стану для формування індивідуальних рекомендацій і виявлення потенційно небезпечних змін у психологічному стані користувача.

Результати тестування продемонстрували високий рівень користувацької залученості, позитивне сприйняття інтерфейсу та ефективність застосунку як

інструменту первинної психологічної допомоги й раннього виявлення тривожних станів. Користувачі відзначили зручність офлайн-функціональності, простоту взаємодії з інтерфейсом та позитивний вплив адаптивного контенту на емоційний стан. Додатково було встановлено, що регулярне використання застосунку сприяє формуванню навичок емоційної саморегуляції та підвищує рівень усвідомлення власного психологічного стану.

Висновки та перспективи. Застосунок «Стоп тривога» підтвердив свою ефективність як цифровий інструмент первинної психологічної допомоги та раннього виявлення тривожних розладів, адаптований до умов війни та підвищеного психоемоційного навантаження. Поєднання емоційної аналітики, сторітелінгу, адаптивного контенту, механізмів гейміфікації та офлайн-функціональності забезпечує можливість своєчасного реагування на зміни психологічного стану користувача та зниження ризику розвитку тяжчих психічних порушень. Отримані результати підтверджують перспективність використання мобільних технологій у сфері цифрового психічного здоров'я та відкривають можливості для подальшого розвитку систем раннього психологічного моніторингу. Подальші етапи розвитку проєкту передбачають масштабування системи, інтеграцію з державними цифровими сервісами, удосконалення алгоритмів персоналізації та емоційного аналізу, впровадження багатомодальної терапії, підтримку взаємодії з носимими пристроями та проходження сертифікації відповідно до європейських стандартів цифрового здоров'я, зокрема DiGA.

Список використаних джерел:

1. Ashraf H., Ramzan M., Amjad T., Shahzad B. A Systematic Study on Software Requirements Elicitation Techniques and its Challenges in Mobile Application Development // IEEE Xplore. – 2018. URL:https://ieeexplore.ieee.org/document/8513829

2. Jabangwe R., Edison H., Nguyen-Duc A. Software engineering process models for mobile app development: A systematic literature review // Journal of Systems and Software. – 2018. URL:https://www.sciencedirect.com/science/article/abs/pii/ S0164121218301638

3. Harari G. M., Lane N. D., Wang R., Crosier B. S., Campbell A. T., Gosling S. D. Using smartphones to collect behavioral data in psychological science // ScienceDirect. – 2016. URL:https://www.sciencedirect.com/science/article/pii/S1745691616650285

4. Ciman M., Gaggi O. An empirical analysis of energy consumption of cross-platform frameworks for mobile development // Pervasive and Mobile Computing. – 2017.
URL:https://www.sciencedirect.com/science/article/abs/pii/S1574119216303170