ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ІНФОРМАЦІЙНА СИСТЕМА ПІДПРИЄМСТВА З ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ PYTHON»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Владислав МЕЖІНСЬКИЙ

Виконав:
здобувач вищої освіти
група ІСД-42
Владислав МЕЖІНСЬКИЙ

Керівник:
науковий ступінь, вчене звання
Оксана ТКАЛЕНКО
к.т.н., доцент

Рецензент:
науковий ступінь, вчене звання
_____
Ім’я, ПРІЗВИЩЕ

Київ – 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра інформаційних систем та технологій

Ступінь вищої освіти бакалавр

Спеціальність 126 Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедрою ІСТ

_____ Каміла СТОРЧАК

«____» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Межінського Владислава Миколайовича

1. Тема кваліфікаційної роботи: «Інформаційна система підприємства з використанням мови програмування Python».
Керівник кваліфікаційної роботи Оксана ТКАЛЕНКО, к.т.н., доцент
затверджені наказом вищого навчального закладу від «24» лютого 2025 р. № 56

2. Строк подання кваліфікаційної роботи «30» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи:
Розробка та впровадження інформаційних систем для автоматизації бізнес-процесів підприємства, з акцентом на застосування мови програмування Python як інструменту для побудови ефективного програмного забезпечення.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Теоретичні основи інформаційних систем підприємства.
2. Аналіз і технологічне обґрунтування розробки.
3. Проєктування та реалізація інформаційної системи «MINIBIZMANAGER»

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «24» лютого 2025 р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Визначення теми та погодження з керівником	24.02 – 28.02.25	
2	Збір та аналіз літературних джерел, нормативної бази	03.03 – 06.03.25	
3	Розробка технічного завдання	07.03 – 15.03.25	
4	Проєктування системи	16.03 – 28.03.25	
5	Реалізація програмного забезпечення	31.03 – 10.04.25	
6	Тестування та налагодження	11.04 – 22.04.25	
7	Написання, оформлення та редагування тексту роботи	23.04 – 14.05.25	
8	Підготовка до захисту, рецензія, оформлення презентації	15.05 – 28.05.25	

Здобувач(ка) вищої освіти _____ Владислав МЕЖІНСЬКИЙ
(підпис)

Керівник кваліфікаційної роботи _____ Оксана ТКАЛЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра (магістра): 66 стор., 5 рис., 4 табл., 20 джерел.

Мета роботи – розробка інформаційної системи підприємства з використанням Python, яка дозволяє здійснювати облік компаній і користувачів, організовувати навчальні або мотиваційні квізи, забезпечувати збереження та аналіз результатів, а також реалізовувати функції безпеки та комунікації всередині організації.

Об'єктом дослідження – процес цифровізації управлінських функцій підприємства.

Предметом дослідження – методи і засоби створення інформаційної системи на основі мови Python та суміжних технологій.

Короткий зміст роботи: У дипломній роботі розглянуто процес розробки інформаційної системи для автоматизації діяльності підприємства із застосуванням мови програмування Python.

У ході роботи було проведено аналіз предметної області, визначено функціональні вимоги до системи, спроєктовано структуру бази даних та інтерфейс користувача. Програмна реалізація виконана з використанням таких технологій, як Python, SQLite/PostgreSQL, а також бібліотек для формування звітів та візуалізації даних.

Система пройшла тестування на відповідність вимогам, продемонструвала стабільну роботу та можливість адаптації до інших типів підприємств.

У результаті впровадження очікується підвищення ефективності облікових процесів, зменшення часу на обробку інформації та зниження ймовірності помилок у ручному введенні даних.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, АВТОМАТИЗАЦІЯ, ПІДПРИЄМСТВО, PYTHON, БАЗА ДАНИХ, ОБЛІК, ЗВІТНІСТЬ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ВЕБ-ДОДАТОК, СТРУКТУРА ДАНИХ.

ЗМІСТ

ВСТУП……..8
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ІНФОРМАЦІЙНИХ СИСТЕМ ПІДПРИЄМСТВА…….11
1.1.Призначення інформаційної системи підприємства……11
1.2.Основні функції та вимоги до системи……..14
1.3.Огляд аналогічних рішень та аналіз їхніх недоліків/переваг……...19
1.4. Постановка задачі – що має реалізовувати інформаційна система……23
РОЗДІЛ 2. АНАЛІЗ І ТЕХНОЛОГІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ……..25
2.1. Обґрунтування вибору мови програмування Python……25
2.2. Огляд веб-технологій: веб-сервіси, протоколи HTTP/HTTPS……29
2.3. Веб-фреймворк FastAPI: переваги, особливості використання……..32
2.4. Огляд використаних бібліотек (pydantic, sqlalchemy, alembic, httpx тощо)……...35
2.5. СУБД та інструменти збереження даних: PostgreSQL, Redis……42
РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «MINIBIZMANAGER»……46
3.1. Архітектура та технологічна основа системи……..46
3.2. Структура системи та основні компоненти……..51
3.3. Моделювання бази даних та сценарії використання……54
3.4. Реалізація, безпека та тестування……..58
3.5. Аналіз отриманих результатів……60
ВИСНОВКИ……...62
ПЕРЕЛІК ПОСИЛАНЬ……..65
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)……...……..67

ВСТУП

У XXI столітті інформаційні технології стали основою розвитку підприємницької діяльності. Практично кожна сучасна організація, незалежно від розміру чи галузі, стикається з необхідністю автоматизації бізнес-процесів, оптимізації обробки даних та впровадження інструментів для підтримки управлінських рішень. Інформаційна система підприємства — це складна сукупність апаратних, програмних і організаційних компонентів, що забезпечують збір, обробку, зберігання та передавання інформації для підтримки ефективного функціонування суб'єкта господарювання.

Особливої уваги заслуговує мова програмування Python, яка останніми роками набула широкого поширення у сфері веб-розробки, обробки даних, автоматизації та створення API. Поєднання Python із фреймворками FastAPI, SQLAlchemy, а також базами даних PostgreSQL і Redis, дозволяє створити продуктивну та ефективну інформаційну систему, що відповідає вимогам малого та середнього бізнесу.

Актуальність теми. У сучасних умовах цифровізації бізнесу ефективне управління підприємством неможливе без впровадження інформаційних систем, що забезпечують автоматизацію основних бізнес-процесів. Зростаючий обсяг даних, необхідність оперативного прийняття рішень та конкурентоспроможність на ринку вимагають від підприємств використання гнучких, масштабованих і надійних програмних рішень.

Python — одна з найпопулярніших мов програмування у світі, відома своєю простотою, потужністю та широкою екосистемою бібліотек для роботи з даними, створення веб-додатків, автоматизації процесів та побудови користувацьких інтерфейсів. Саме тому Python є ідеальним інструментом для розробки сучасних інформаційних систем.

Розробка інформаційної системи підприємства з використанням Python дозволяє забезпечити високу ефективність внутрішніх процесів, підвищити рівень контролю за діяльністю підприємства та покращити взаємодію між структурними

підрозділами. Такий підхід також дозволяє створювати адаптивні рішення, які легко інтегруються в існуючу IT-інфраструктуру.

Таким чином, обрана тема є надзвичайно актуальною, оскільки відповідає потребам сучасного бізнес-середовища та тенденціям розвитку інформаційних технологій, сприяє розвитку практичних навичок програмування і демонструє ефективність використання Python у прикладних задачах.

Аналіз останніх досліджень і публікацій засвідчує зростання інтересу до тематики побудови легковагих, масштабованих інформаційних систем на основі Python та суміжних технологій. У фахових працях і прикладних дослідженнях акцентується на застосуванні REST API, асинхронних фреймворків, хмарних інфраструктур. Проте, значна частина таких рішень не враховує потреби малих українських підприємств або орієнтується виключно на іноземні стандарти. Це підкреслює необхідність створення гнучкої моделі, адаптованої до локальних умов.

Мета дослідження - розробка інформаційної системи підприємства з використанням Python, яка дозволяє здійснювати облік компаній і користувачів, організовувати навчальні або мотиваційні квізи, забезпечувати збереження та аналіз результатів, а також реалізовувати функції безпеки та комунікації всередині організації.

Для досягнення поставленої мети в роботі вирішуються такі завдання:

1. Проаналізувати теоретичні засади інформаційних систем підприємства.
2. Обґрунтувати вибір технологічного стеку та інструментів розробки.
3. Спроєктувати інформаційну систему з урахуванням архітектури та сценаріїв використання.
4. Реалізувати систему та провести її тестування.

Об’єктом дослідження виступає процес цифровізації управлінських функцій підприємства.

Предметом дослідження є методи і засоби створення інформаційної системи на основі мови Python та суміжних технологій.

Методи дослідження, що застосовувалися в роботі, включають:

- аналіз і порівняння функціональних характеристик існуючих ІС;

- системний підхід до проєктування програмного забезпечення;
- методи структурного і об'єктно-орієнтованого програмування;
- моделювання баз даних;
- експериментальне тестування з метою перевірки працездатності системи.

Наукова новизна роботи полягає у поєднанні сучасних відкритих технологій Python-екосистеми для створення кастомізованої, масштабованої інформаційної системи, орієнтованої на специфіку локального підприємства. Система є універсальною платформою для інтеграції модулів управління персоналом, навчанням, аналітики та сповіщень.

Практична значущість полягає в тому, що розроблене рішення може бути впроваджене у малих і середніх компаніях, стартапах, навчальних закладах або проєктних командах без потреби у великих інвестиціях у комерційні рішення.

Апробація результатів дослідження. Основні положення, результати та практичні напрацювання, викладені в дипломній роботі, були апробовані шляхом участі у всеукраїнських науково-технічних конференціях та представлені у вигляді тез доповідей:

1. Межінський В.М. Особливості стандартів WirelessHART та ISA100.11A Тези доповіді на V Всеукраїнській науково-технічній конференції *«Сучасний стан та перспективи розвитку IoT»*. — Київ, 18 квітня 2024 р. — С. 153–156.
2. Межінський В.М. Аналіз технологій розробки мікросервісів на Python Тези доповіді на II Всеукраїнській науково-технічній конференції *«Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу»*. — Київ, 18 листопада 2024 р. — С. 196–198.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ ОСНОВИ ІНФОРМАЦІЙНИХ СИСТЕМ ПІДПРИЄМСТВА

1.1 Призначення інформаційної системи підприємства

У сучасних умовах глобалізації, високої конкуренції та цифровізації бізнесу інформаційна система (ІС) підприємства виступає не лише інструментом обробки даних, а й стратегічним елементом управління організаційними процесами. Її головне призначення полягає в забезпеченні оперативного, достовірного та зручного доступу до інформації, що необхідна для ухвалення ефективних управлінських рішень на всіх рівнях підприємства [1].

Інформаційна система підприємства — це сукупність взаємопов'язаних програмно-апаратних засобів, баз даних, інформаційних потоків та процедур, які забезпечують автоматизовану підтримку бізнес-процесів. Вона охоплює всі функціональні сфери діяльності підприємства: фінансову, виробничу, маркетингову, кадрову, логістичну тощо.

Ключовими завданнями інформаційної системи є:

1. Автоматизація рутинних процесів - значна частина щоденних операцій на підприємстві має повторюваний характер: облік працівників, реєстрація замовлень, формування звітів. ІС дозволяє зменшити витрати часу та людських ресурсів, мінімізуючи кількість помилок і покращуючи продуктивність.
2. Інтеграція даних у єдину систему - завдяки інформаційній системі всі структурні підрозділи підприємства можуть працювати з єдиною базою даних, що виключає дублювання інформації та підвищує узгодженість управлінських рішень.
3. Підтримка прийняття рішень - ІС забезпечує керівників актуальними звітами, статистикою та аналітичними даними, що допомагають об'єктивно оцінити ситуацію та швидко реагувати на зміни ринку або внутрішнього середовища.

4. Контроль і моніторинг діяльності - кожен етап виробничого або управлінського процесу може бути прозоро відстежений через систему, що дозволяє виявляти вузькі місця, проводити аудит і забезпечувати відповідальність персоналу.
5. Підвищення якості обслуговування клієнтів - завдяки наявності CRM-модуля або аналогічного функціоналу, інформаційна система дозволяє підприємству ефективно керувати контактами з клієнтами, вести історію взаємодій, формувати індивідуальні пропозиції тощо.

Залежно від завдань підприємства та сфери діяльності, інформаційні системи можуть поділятися на:

1. Операційні системи — автоматизують виконання повсякденних операцій (облік, розрахунки, складування);
2. Інформаційно-аналітичні системи — створені для обробки великих обсягів даних та формування звітності;
3. Управлінські системи — підтримують стратегічне планування, прогнозування, розробку сценаріїв розвитку підприємства;
4. Гібридні системи, що поєднують кілька функцій — найпоширеніші в реальному бізнес-середовищі.

У більшості сучасних підприємств функціонал ІС охоплює кілька модулів: фінанси, персонал, запаси, виробництво, аналітику, документообіг, клієнтську базу. Такий підхід дозволяє підприємству масштабувати свою систему відповідно до потреб.

Впровадження ефективної інформаційної системи здатне суттєво вплинути на конкурентоспроможність підприємства. В умовах ринку, що стрімко змінюється, здатність швидко адаптуватися, впроваджувати нові функції, інтегруватися з іншими сервісами (наприклад, банківськими або податковими), а також забезпечувати кібербезпеку — усе це стало базовими вимогами для виживання на ринку.

Інформаційна система може також виступати платформою для інновацій — таких як інтеграція штучного інтелекту, прогнозної аналітики, хмарних технологій,

мобільних застосунків тощо. У результаті ІС стає не лише «сервісним» інструментом, а й основою цифрової стратегії підприємства.

Наведемо цікаві факти і сучасні приклади призначення інформаційної системи підприємства [2]:

1. Сучасні ІСП працюють у хмарі (cloud) - це дозволяє керівникам працювати з будь-якої точки світу, навіть зі смартфона. Наприклад, система SAP Business One або Microsoft Dynamics 365.
2. Інтелектуальні системи використовують штучний інтелект — наприклад, прогнозують попит на продукцію або аналізують поведінку клієнтів.
3. Кібербезпека — критично важливий компонент ІСП. У сучасних системах вбудовані механізми захисту даних, особливо коли йдеться про фінанси чи персональні дані клієнтів.
4. ІСП як конкурентна перевага — компанії з якісною ІСП можуть швидше реагувати на ринок, краще взаємодіяти з клієнтами, бути більш гнучкими у кризових ситуаціях (як, наприклад, під час пандемії чи війни).
5. Українські аграрні компанії дедалі частіше використовують IT-рішення для точного землеробства: дрони, GPS-навігація, аграрні CRM-системи. Це дозволяє ефективніше використовувати ресурси, збільшувати врожайність і зменшувати витрати.

У процесі вивчення призначення інформаційної системи підприємства було з'ясовано, що вона відіграє ключову роль у забезпеченні ефективної діяльності організації. Інформаційна система (ІС) дає змогу автоматизувати бізнес-процеси, оптимізувати управлінські рішення, підвищити точність та швидкість обробки даних, що, своєю чергою, сприяє конкурентоспроможності підприємства на ринку.

Завдяки сучасним технологіям інформаційні системи також забезпечують інтеграцію різних підрозділів, покращення внутрішньої комунікації та підтримку прийняття рішень на всіх рівнях управління.

Таким чином, інформаційна система виступає як критично важливий інструмент у досягненні цілей підприємства, адаптації до змін зовнішнього середовища та забезпеченні стабільного розвитку.

1.2 Основні функції та вимоги до системи

Ефективна інформаційна система підприємства має відповідати ряду функціональних та нефункціональних вимог, які формуються відповідно до специфіки організаційної структури, галузі діяльності, масштабу компанії та стратегічних цілей [3].

У цьому контексті функції системи є конкретними діями, які вона повинна виконувати для підтримки бізнес-процесів, а вимоги — це сукупність характеристик, яким повинна відповідати система для забезпечення надійної, безпечної та зручної роботи.

Інформаційна система підприємства виконує такі ключові функції:

1. Збір та введення даних. Система має забезпечувати інтерфейси для зручного внесення інформації користувачами, імпорту даних з інших джерел, а також автоматизованого збирання даних (наприклад, з форм, електронних листів, API або сенсорних пристроїв).

Приклад*:* У торговельній мережі ІСП може автоматично збирати інформацію з касових апаратів у реальному часі. Якщо ж це агропідприємство — дані можуть надходити із сенсорів у полі: температура, вологість, стан ґрунту.

2. Зберігання та управління інформацією. Централізована база даних є ядром системи, де зберігаються всі сутності підприємства (працівники, компанії, документи, дії). Необхідна підтримка структурованого збереження, індексації, пошуку та резервного копіювання.

Всі дані (контрагенти, рахунки, договори, працівники) зберігаються в базі даних — наче «цифровий архів». ІСП дає змогу швидко знаходити потрібну інформацію та гарантує, що вона не буде втрачена.

Приклад*:* у банку клієнтська інформація (вклади, кредити, історія операцій) зберігається централізовано, і її можна отримати за кілька кліків.

3. Обробка та аналіз даних. Система повинна реалізовувати логіку обробки даних відповідно до потреб бізнесу — наприклад, підрахунок показників,

генерація звітів, виявлення відхилень або автоматичні сповіщення при виникненні подій.

ІСП може автоматично розраховувати показники ефективності, генерувати аналітику за день, тиждень, квартал.

Приклад: У виробництві система може розрахувати відсоток бракованої продукції й автоматично надіслати сповіщення керівництву.

4. Управління процесами. ІС підтримує автоматизацію внутрішніх процесів: погодження, делегування, контроль виконання, планування. Завдяки цьому знижується рівень адміністративного навантаження та зростає прозорість роботи.

Приклад: У компанії з великою кількістю підрозділів ІСП може автоматизувати затвердження відпусток, проходження документів по відділах, чи контроль виконання завдань.

5. Звітність і візуалізація. Однією з ключових функцій є формування звітів у зручному для сприйняття вигляді: таблиці, графіки, діаграми. Це дозволяє приймати зважені рішення, базуючись на об’єктивних даних.

Усі аналітичні дані мають бути подані не лише в таблицях, а й у вигляді діаграм, графіків.

Приклад: CRM-система може показати, як змінюються продажі в залежності від сезонів або рекламної кампанії.

6. Комунікація між користувачами. Інформаційна система може слугувати середовищем взаємодії: надсилання сповіщень, запрошень, створення чатів або форумів всередині організації.

ІСП може мати вбудовану систему повідомлень, календарі нарад, обговорення у стилі форуму.

Приклад: У проєктній ІТ-компанії співробітники використовують модуль обміну файлами, завданнями та нотатками прямо у системі.

7. Інтеграція з іншими системами. Сучасна ІС повинна мати відкритий API або інші механізми для зв’язку з бухгалтерськими системами, CRM, вебсайтами, мобільними застосунками тощо.

Інформаційна система підприємства повинна відповідати ряду функціональних і нефункціональних вимог, які визначають її ефективність, масштабованість та надійність. Основні функції системи мають бути орієнтовані на підтримку управлінських процесів, оперативну обробку даних, а також забезпечення автоматизації типових операцій [4].

До ключових напрямків, які реалізуються за допомогою інформаційної системи, належать управління ресурсами підприємства, аналітика даних, а також автоматизація бізнес-процесів. При цьому важливою умовою є гарантування безпеки даних та стабільності роботи системи в умовах реального навантаження.

Для наочного представлення функціональної структури інформаційної системи підприємства на рисунку нижче подано її основні складові:

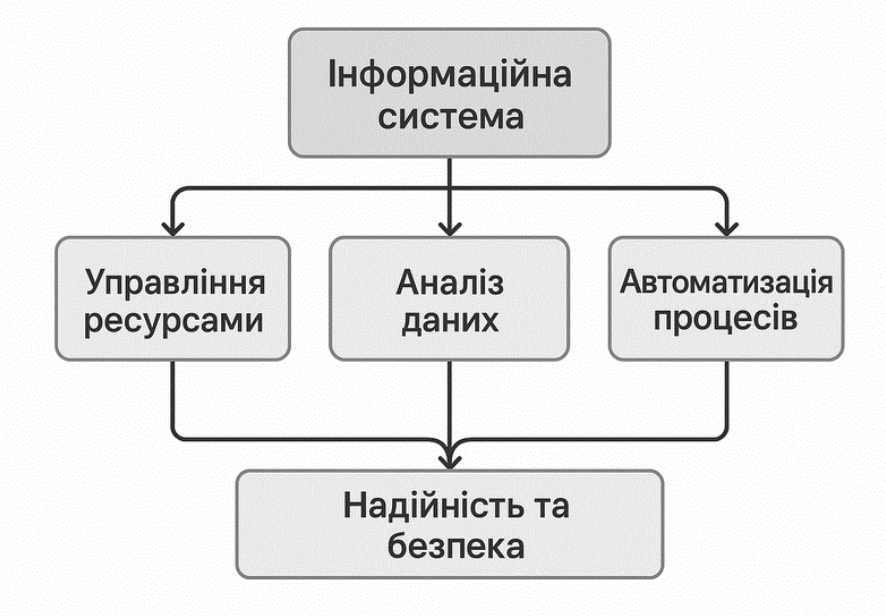


Рис. 1.1 Основні функції та вимоги до інформаційної системи підприємства

На рис. 1.1 представлено ключові функціональні напрямки, які повинна реалізовувати інформаційна система підприємства: управління ресурсами, аналіз даних та автоматизація бізнес-процесів. Усі ці компоненти спрямовані на досягнення головної мети – забезпечення надійності та безпеки системи. Така структура відображає загальні вимоги до сучасних інформаційних систем, що використовуються у сфері управління підприємством [5].

Вимоги до ІС поділяються на функціональні (що саме повинна робити система) та нефункціональні (якою вона має бути за характеристиками якості).

Успішна ІСП не працює окремо — вона з’єднується з бухгалтерією, сайтом, CRM, мобільним додатком або системою онлайн-платежів.

Приклад: Після онлайн-покупки замовлення автоматично з'являється у системі логістики та запускає процес доставки.

Функціональними вимогами є:

- Реєстрація та автентифікація користувачів;
- CRUD-операції для основних сутностей (створення, читання, оновлення, видалення);
- Налаштування прав доступу до функціоналу;
- Формування звітності за різними критеріями;
- Нотифікації про зміни в даних або події;
- Підтримка кількох ролей користувачів (адміністратор, менеджер, співробітник).

Нефункціональними вимогами є:

1. Надійність. Система має працювати стабільно, без збоїв, навіть за великого навантаження. Потрібно передбачити механізми обробки винятків та аварійного відновлення.

Приклад: Якщо сервер впаде — дані зберігаються завдяки резервному копіюванню.

2. Безпека. Забезпечення автентифікації (наприклад, JWT), контроль доступу, захист від SQL-ін’єкцій, XSS-атак, відповідність стандартам GDPR щодо захисту персональних даних. Використання ролей, паролів, двофакторної автентифікації (2FA), обмеження IP-адрес.

Приклад: Фінансові системи, як-от «Приват24», мають високий рівень захисту, щоб убезпечити банківські операції.

3. Масштабованість. ІС повинна мати можливість розширення: як за кількістю користувачів, так і за функціоналом — без необхідності повного перепроєктування. Коли компанія росте, ІСП повинна без проблем обслуговувати більше користувачів і процесів.

4. Продуктивність. Час відгуку системи має бути мінімальним, навіть при великій кількості запитів. Потрібна підтримка кешування, асинхронної обробки, ефективної роботи з базою даних. Сторінки повинні відкриватися швидко, навіть якщо в базі мільйони записів.
5. Юзабіліті (зручність користування). Інтерфейс має бути інтуїтивно зрозумілим, з адаптивним дизайном для різних пристроїв. Зрозумілий інтерфейс, адаптований під комп'ютер, планшет, смартфон.
6. Супроводжуваність. Кодова база та структура проєкту повинні бути зрозумілими, із чіткою документацією для подальшого обслуговування або доопрацювання. Добре документований код, зрозуміла структура проєкту, що дозволяє легко вносити зміни.

У ході аналізу основних функцій та вимог до інформаційної системи підприємства було встановлено, що ефективність її впровадження безпосередньо залежить від чіткого визначення функціонального призначення та відповідності вимогам користувачів і бізнес-процесів.

Серед основних функцій інформаційної системи виділяються: автоматизація обліку й звітності, підтримка управлінських рішень, забезпечення комунікації між структурними підрозділами, зберігання і швидкий доступ до даних, аналіз діяльності підприємства тощо. Ці функції дозволяють підприємству працювати скоординовано, гнучко реагувати на зміни та підвищувати продуктивність [6].

Щодо вимог до ІС, до ключових належать: надійність, масштабованість, інтеграційна здатність, безпека, простота використання, а також відповідність нормативно-правовим стандартам. Відповідність цим вимогам забезпечує стабільну роботу системи, зручність для користувачів і високий рівень інформаційного захисту.

Отже, правильне формулювання функцій та вимог до інформаційної системи є основою для її успішного впровадження й ефективного функціонування на підприємстві.

1.3 Огляд аналогічних рішень та аналіз їхніх недоліків/переваг

Перш ніж розпочинати розробку власної інформаційної системи для підприємства, доцільно провести аналіз існуючих програмних рішень, які пропонуються на ринку. Це дозволяє не лише оцінити сучасні підходи до побудови інформаційних систем, а й виявити їхні переваги та недоліки для формування вимог до власного продукту.

У цьому підрозділі розглянемо як комерційні, так і відкриті (open-source) рішення, що охоплюють потреби автоматизації бізнес-процесів малого та середнього підприємництва [7].

Комерційні програмні рішення:

1С: Підприємство - одна з найпоширеніших систем у пострадянському просторі. Призначена для автоматизації бухгалтерського обліку, управління складом, продажами, персоналом тощо. Має модульну структуру.

Перевагами є:

- Готові типові конфігурації під різні галузі;
- Інтеграція з податковими та фінансовими системами;
- Широка підтримка в Україні.
- Поширеність у бухгалтерських службах України;
- Інтеграція з державними сервісами (звітність до податкової, е-декларації).

Недоліками є:

- Закрита екосистема, обмеження на зміну логіки роботи;
- Висока вартість ліцензій і супроводу;
- Застарілий інтерфейс та обмежена підтримка веб-технологій;
- Залежність від російського розробника — актуальне питання в умовах війни.

Після 2022 року дедалі більше українських компаній відмовляються від 1С на користь західних або українських рішень.

SAP Business One / SAP ERP - глобальне рішення для великих компаній. Має потужні модулі для фінансів, логістики, управління персоналом та аналітики.

Перевагами є:

- Глибока кастомізація;
- Потужна система звітності;
- Висока безпека і масштабованість.

Недоліками є:

- Дуже дорога вартість впровадження;
- Тривалий час налаштування;
- Надмірна складність для малого бізнесу.

Приклад: Використовується в «Кернел», «Укрзалізниця», великих логістичних та агрокомпаніях.

Bitrix24 - хмарна платформа з широким функціоналом CRM, управління задачами, співробітниками, документообігом.

Перевагами є:

- Хмарна інфраструктура — не потребує серверів;
- Вбудована телефонія, email-маркетинг, чат-боти;
- Інтерфейс українською мовою;
- Швидкий старт — можна розгорнути за 1 день;
- CRM, завдання, документообіг, HR у "одній коробці";
- Інтеграції з Telegram, WhatsApp, Instagram.

Недоліками є:

- Обмежені можливості доопрацювання логіки;
- Проблеми зі швидкодією при великій кількості даних;
- Закритий код та складнощі з перенесенням даних при зміні платформи.

Приклад: Часто використовують інтернет-магазини, освітні проєкти, невеликі IT-компанії.

Відкриті (open-source) рішення:

Odoo - гнучка ERP-система з модульною архітектурою, що включає управління проєктами, торгівлею, складом, кадрами.

Перевагами є:

- Повністю відкритий код;

- Велика кількість готових модулів;
- Можливість локалізації під українські реалії.

Недоліками є:

- Високі вимоги до технічних знань для розгортання;
- Наявність платних модулів у базовій версії;
- Обмежена кількість підтримки українською.

Приклад: Може підійти агрофірмам, що хочуть автоматизувати постачання, врожаї та збут.

ERPNext - ще одна open-source ERP-система на Python з потужним веб-інтерфейсом. Підходить для малого й середнього бізнесу [5].

Перевагами є:

- Побудована на Python (основа майбутньої системи);
- Добра документація;
- Гнучка у розгортанні.

Недоліками є:

- Потрібен досвід DevOps;
- Обмежена підтримка українських бухгалтерських стандартів;
- Вузька спільнота в Україні.

Приклад: Підходить IT-компаніям, консалтингу, сервісним агентствам.

У процесі аналізу сучасних рішень для автоматизації бізнес-процесів малого підприємництва було розглянуто декілька актуальних інформаційних систем, доступних на українському ринку. Ці системи різняться за функціональністю, масштабом, платформою та орієнтацією на цільову аудиторію. Узагальнені характеристики представлено у табл. 1.1 [8].

Таблиця 1.1

Характеристика сучасних рішень для малого бізнесу в Україні

Система	**Платформа**	**Особливості**
Worksection	Українська	Управління проєктами, задачами. Проста й ефективна.
KeepinCRM	Українська	Легка CRM-система для малого бізнесу та ФОПів.
Zoho One	Глобальна (Saas)	Комплекс із 40+ застосунків (CRM, пошта, маркетинг).

Worksection — українська платформа для управління проєктами, яка дозволяє ефективно розподіляти задачі, контролювати строки та взаємодіяти в команді. Завдяки інтуїтивному інтерфейсу й хмарному формату, система зручна для невеликих команд.

KeepinCRM — ще одне українське рішення, спеціально розроблене для малого бізнесу та фізичних осіб-підприємців (ФОП). Система охоплює базові CRM-функції, включаючи облік клієнтів, замовлень, оплат та взаємодій, і не потребує складного навчання персоналу.

Zoho One — глобальна хмарна платформа типу SaaS, яка об’єднує понад 40 застосунків, зокрема для CRM, маркетингу, електронної пошти, управління персоналом і фінансами. Це рішення підходить для бізнесів, які планують масштабування та інтеграцію кількох напрямів діяльності в єдину систему.

Ці системи демонструють, що навіть для малих підприємств сьогодні доступні гнучкі та функціональні інструменти автоматизації бізнес-процесів, які можна адаптувати до конкретних потреб.

1.4 Постановка задачі – що має реалізовувати інформаційна система

На основі проведеного аналізу предметної області, дослідження існуючих програмних рішень та вивчення потреб сучасного підприємства, можна сформулювати перелік основних задач, які має вирішувати розроблювана інформаційна система. Її функціональність повинна не лише автоматизувати ключові бізнес-процеси, а й забезпечити простоту в користуванні, надійність, масштабованість і безпеку [9].

Розроблювана система передбачена як веборієнтоване рішення з відкритим програмним інтерфейсом (REST API), що дозволить інтегруватися з іншими сервісами, застосовувати рольову модель доступу та підтримувати багатокористувацьку взаємодію [7].

Мета створення інформаційної системи - створити інформаційну систему для підприємства, яка дозволяє:

- ефективно керувати даними про компанії та користувачів;
- організовувати та адмініструвати внутрішні тести або квізи для працівників;
- забезпечувати аналітику результатів і комунікацію між працівниками через сповіщення;
- реалізувати реєстрацію, автентифікацію та контроль доступу до ресурсів;
- забезпечити високу продуктивність і безпеку роботи системи.

Ключовими функціональними задачами системи є:

1. Реєстрація та автентифікація користувачів - користувачі повинні мати можливість створити обліковий запис, авторизуватися за допомогою логіна та пароля, отримувати токен доступу (JWT) для взаємодії з API.
2. Управління компаніями - адміністратори мають змогу створювати, редагувати та видаляти компанії, додавати до них користувачів, формувати ієрархії доступу.
3. Формування та проходження квізів - створення тестових опитувань для співробітників з можливістю вказання варіантів відповідей, збереження результатів, аналізу відповідей тощо.

4. Система запрошень - адміністратор компанії має змогу надсилати запрошення новим працівникам через email або API-запити, що дозволяє ефективно інтегрувати нових учасників у систему.
5. Сповіщення - надсилання повідомлень користувачам системи про нові квізи, зміни в обліковому записі, результати тестування тощо.
6. Аналітика - побудова звітів за результатами проходження квізів, активністю працівників, статистикою відповідей. Реалізація базових аналітичних панелей (dashboards).

Нефункціональними задачами є:

1. Використання сучасного стеку технологій - система буде реалізована на основі Python із використанням FastAPI, SQLAlchemy, Pydantic, PostgreSQL , Redis — що забезпечує швидкодію, підтримку масштабування та модульність коду.
2. Безпека - система має використовувати JSON Web Token (JWT) для безпечної ідентифікації, механізми контролю CORS, рольовий розподіл прав доступу.
3. Масштабованість та гнучкість - код системи повинен дозволяти додавання нових модулів і сервісів без значних змін основного ядра.
4. Документованість API - кожна функція системи буде описана через Swagger/OpenAPI, що дозволить легко тестувати й інтегрувати сторонні інтерфейси.

Результатом проєкту стане стабільна, надійна та гнучка інформаційна система, яка може бути впроваджена на реальному підприємстві. Вона відповідатиме сучасним вимогам якості програмного забезпечення, дозволятиме ефективно управляти внутрішніми процесами компанії та буде придатна до розширення в майбутньому (наприклад, додавання мобільного клієнта, імпорт/експорт Excel-файлів, інтеграція з поштовими сервісами тощо).

РОЗДІЛ 2.
АНАЛІЗ І ТЕХНОЛОГІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ

2.1 Обґрунтування вибору мови програмування Python

У сучасному світі мова програмування виступає не лише інструментом реалізації алгоритмів, а й визначальним чинником ефективності, масштабованості та підтримуваності програмного продукту. При виборі технологічного стеку для створення інформаційної системи підприємства особливу увагу було приділено мові Python, яка на сьогодні займає лідируючі позиції серед мов загального призначення [10].

Згідно з рейтингами TIOBE, Stack Overflow Developer Survey та RedMonk, Python уже кілька років поспіль утримує позиції в топ-3 найпопулярніших мов програмування у світі. Його популярність зумовлена широкою сферою застосування — від веб-розробки та машинного навчання до автоматизації бізнес-процесів і фінансового аналізу.

Широке використання Python гарантує постійну підтримку з боку спільноти, регулярні оновлення, розвинену екосистему бібліотек і фреймворків. Це особливо важливо для інформаційних систем, які потребують надійних, перевірених і гнучких інструментів для обробки даних, безпеки, взаємодії з базами даних і розробки API.

Однією з найважливіших переваг Python є лаконічний та читабельний синтаксис, що дозволяє скоротити обсяг коду і пришвидшити процес розробки.

Для інформаційної системи, яка включає в себе роботу з численними модулями (користувачі, компанії, квізи, аналітика, безпека), це означає менші витрати часу на реалізацію базових функцій і вищу продуктивність розробника.

Крім того, Python ідеально підходить для швидкого прототипування. Це дає змогу на ранніх етапах перевіряти працездатність логіки, демонструвати функціональність замовнику та вносити зміни без істотного перероблення структури [11].

Python має велику кількість бібліотек, які полегшують реалізацію будь-яких аспектів інформаційної системи. Зокрема:

- FastAPI — сучасний веб-фреймворк для побудови RESTful API з підтримкою асинхронності, автоматичною генерацією документації (Swagger/OpenAPI) та валідаторами типів на основі Pydantic;
- SQLAlchemy — ORM для взаємодії з базами даних, яка дозволяє описувати структуру БД на рівні Python-класів;
- Alembic — інструмент для створення та керування міграціями БД [8];
- Pydantic — бібліотека для перевірки та серіалізації даних, яка дозволяє зручно працювати з API та формами [9];
- httpx, requests — для роботи з HTTP-запитами;
- Celery/Redis — для обробки асинхронних задач та тимчасового кешування [10].

Використання Python у поєднанні з цими інструментами дозволяє створювати високопродуктивні, масштабовані та підтримувані інформаційні системи.

Зобразимо схему, що ілюструє аргументи на користь вибору Python як мови програмування на рисунку нижче:

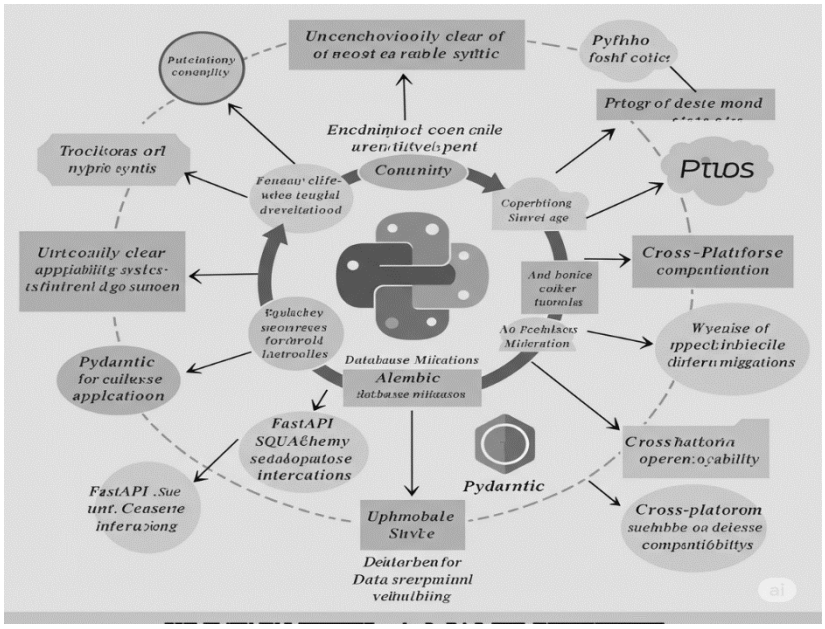


Рис. 2.1 Схема, що ілюструє аргументи на користь вибору Python як мови програмування

Схема, зображена на рис. 2.1 чудово ілюструє ключові переваги Python як мови програмування. Дійсно, ці фактори роблять Python дуже привабливим вибором для широкого кола завдань та розробників.

Розглянемо детальніше кожен із зазначених пунктів на рисунку:

Читабельність коду: синтаксис Python спроєктований таким чином, щоб бути максимально чітким та лаконічним. Це робить код легшим для розуміння, написання та підтримки, особливо у великих проєктах та при роботі в команді.

Відсутність зайвих символів (наприклад, крапок з комою в кінці рядка або фігурних дужок для блоків коду) та використання відступів для визначення структури сприяють кращій візуальній організації.

Широке застосування: Python є універсальною мовою, яка використовується в різноманітних сферах, таких як:

- Веброзробка: (згадані нами FastAPI, а також Django, Flask);
- Аналіз даних та машинне навчання: (NumPy, Pandas, Scikit-learn, TensorFlow, PyTorch);
- Наукові обчислення: (SciPy, Matplotlib);
- Автоматизація та написання скриптів;
- Розробка настільних додатків: (Tkinter, PyQt, Kivy);
- Ігрова розробка: (Pygame);
- Інтернет речей (IoT).

Велика підтримка спільноти: Python має одну з найбільших та найактивніших спільнот розробників у світі [12]. Це означає:

- Величезна кількість документації, навчальних матеріалів та прикладів коду.
- Швидка допомога та відповіді на запитання на форумах (наприклад, Stack Overflow) та в спільнотах.
- Постійний розвиток мови та її екосистеми.

Багатий набір бібліотек: Як ви зазначили, Python має величезну стандартну бібліотеку та доступ до тисяч сторонніх пакетів через менеджер пакетів pip. Це

дозволяє розробникам не "винаходити велосипед", а використовувати готові рішення для багатьох завдань.

Функції безпеки: Хоча безпека програми залежить від багатьох факторів, включаючи практики розробки, Python має певні вбудовані функції та бібліотеки, які допомагають створювати більш безпечні додатки. Фреймворки, такі як Django та FastAPI, також мають вбудовані механізми захисту від поширених вебатак (наприклад, XSS, CSRF). Крім того, активна спільнота швидко виявляє та виправляє вразливості.

Крос-платформова сумісність: Код на Python може працювати на різних операційних системах (Windows, macOS, Linux) без необхідності внесення значних змін. Це спрощує розробку та розгортання додатків для різних середовищ.

Отже, на цій схемі бачимо точне відображення основних причин популярності Python. Це потужна, гнучка та доступна мова програмування, яка продовжує розвиватися та знаходити нові сфери застосування.

Python має бібліотеки для реалізації всіх сучасних стандартів безпеки: автентифікація через JWT, контроль доступу, CORS, шифрування паролів, робота з OAuth2. Це дозволяє створити систему, яка відповідатиме вимогам безпеки для збереження персональних і корпоративних даних.

Python працює на всіх основних операційних системах (Windows, Linux, macOS), легко розгортається на локальних серверах, VPS та в хмарних середовищах (AWS, Heroku, GCP). Це дозволяє обрати оптимальну модель розгортання відповідно до потреб підприємства [11].

Таким чином, мова Python є логічним і стратегічно виправданим вибором для реалізації інформаційної системи підприємства. Вона поєднує в собі простоту, гнучкість, потужний набір інструментів для розробки веб-рішень і підтримку сучасних стандартів безпеки. Це забезпечує високу швидкість розробки, зручність підтримки та можливість масштабування розробленої системи в майбутньому.

2.2 Огляд веб-технологій: веб-сервіси, протоколи HTTP/HTTPS

Сучасні інформаційні системи підприємств, зокрема ті, що базуються на архітектурі «клієнт-сервер», майже завжди використовують веб-технології як основу взаємодії між компонентами. Це пояснюється універсальністю, кросплатформенністю та широкою підтримкою на всіх рівнях — від браузерів до серверних рішень. У цьому контексті веб-сервіси та протоколи HTTP/HTTPS виступають ключовими елементами функціонування інформаційної системи [8].

Веб-сервіс — це програмний компонент, який забезпечує обмін даними між різними системами або додатками через інтернет або внутрішню мережу. У більшості випадків взаємодія з веб-сервісом здійснюється за допомогою HTTP-запитів до певного набору ендпоінтів (URI), які реалізують певну функціональність (наприклад, отримати список компаній, створити користувача тощо).

У контексті даної дипломної роботи реалізація інформаційної системи передбачає побудову RESTful веб-сервісу, тобто сервісу, який дотримується принципів REST (Representational State Transfer). Це означає:

- використання стандартних HTTP-методів (GET, POST, PUT, DELETE);
- чітка структура ресурсів (наприклад, /users, /companies, /quizzes);
- відсутність стану на стороні сервера (статус користувача зберігається клієнтом, зокрема через JWT-токени);
- уніфікований інтерфейс.

REST API забезпечує ефективну, масштабовану і зрозумілу взаємодію між фронтендом (клієнтською частиною) та бекендом (серверною логікою).

У сучасних інформаційних системах для забезпечення взаємодії між клієнтськими додатками та серверною частиною широко застосовуються REST API, які базуються на протоколах HTTP або HTTPS. Такий підхід дозволяє реалізувати ефективний та масштабований обмін даними між різними компонентами системи [13].

На рис. 2.2 зобразимо загальну схему взаємодії клієнта з інформаційною системою через REST API, що ілюструє послідовність запитів і відповідей між клієнтом, веб-сервером та базою даних.

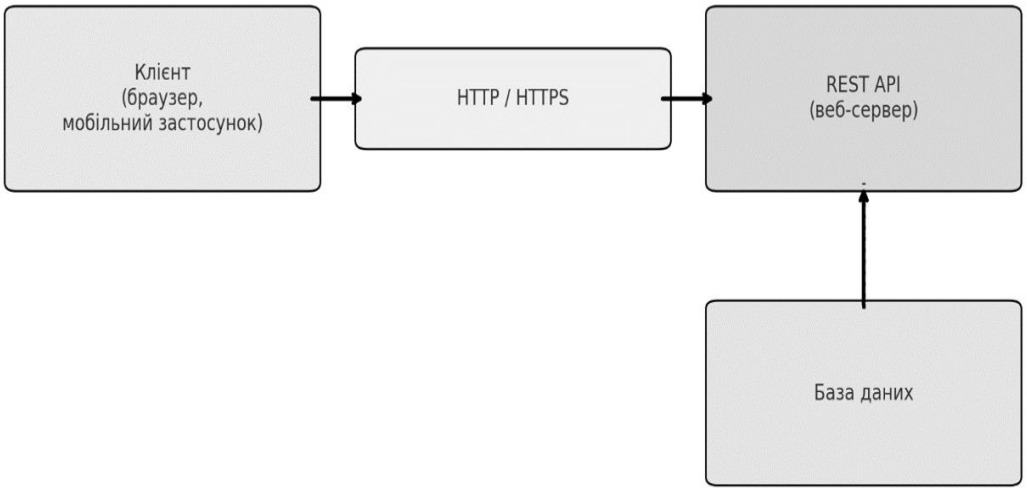


Рис. 2.2 Схема взаємодії клієнта з інформаційною системою через REST API

HTTP (HyperText Transfer Protocol) — це протокол прикладного рівня, який є основою для обміну даними в Інтернеті. Його основними характеристиками є клієнт-серверна модель. Клієнт (наприклад, браузер або мобільний застосунок) ініціює запит до сервера [8].

Методи, які найчастіше використовуються:

GET — отримання ресурсу; /users/1 – повертає користувача з ID 1;

POST — створення нового ресурсу; /users з тілом запиту – створює нового користувача;

PUT — оновлення існуючого ресурсу; /users/1 з тілом – оновлює дані користувача;

DELETE — видалення ресурсу; /users/1 – видаляє користувача з ID 1;

PATCH — часткове оновлення; /users/1 з частковими змінами – оновлення email.

Заголовки (headers) – містять метаінформацію: тип контенту, авторизаційні дані, формат відповіді.

Коди стану (status codes) – повідомляють про результат обробки запиту:

- 200 OK — запит виконано успішно;
- 201 Created — ресурс створено;

- 400 Bad Request — помилка в запиті (неправильний запит);
- 401 Unauthorized — потреба в автентифікації;
- 404 Not Found — ресурс не знайдено;
- 500 Internal Server Error — внутрішня помилка сервера.

Завдяки цій структурі HTTP є простою, але потужною основою для обміну даними між системами.

HTTPS (HTTP Secure) — це розширення протоколу HTTP, яке забезпечує шифрування трафіку за допомогою SSL/TLS. Усі дані, що передаються між клієнтом і сервером, шифруються, що унеможливлює їх перехоплення або зміну третіми сторонами.

Для інформаційної системи, яка працює з персональними даними, результати тестування, email-контактами користувачів тощо, використання HTTPS є обов'язковим з міркувань безпеки. Крім того, робота через HTTPS — це вимога стандартів GDPR та інших міжнародних регламентів захисту даних.

Використання веб-сервісів і протоколів HTTP/HTTPS у розробці інформаційної системи має низку переваг:

- Кросплатформенність. Сервіси можна викликати з будь-якого пристрою (веб, мобільний, десктоп);
- Гнучкість. REST API легко масштабуються і розширюються;
- Стандартизація. HTTP/HTTPS — загальноприйняті та добре документовані протоколи;
- Простота тестування та налагодження. Існують численні інструменти: Postman, Swagger, Curl;
- Інтеграція. Легке з'єднання з іншими системами: CRM, ERP, платіжні шлюзи, сервіси email/SMS.

Веб-сервіси та протоколи HTTP/HTTPS є технологічною основою сучасної інформаційної системи підприємства. Вони забезпечують ефективну та безпечну взаємодію між компонентами системи, дають можливість створювати масштабовані та відкриті API, а також гарантують відповідність вимогам безпеки

при роботі з конфіденційними даними. Надалі ці протоколи будуть активно використовуватись для реалізації REST-інтерфейсів системи в рамках обраної архітектури [9].

2.3 Веб-фреймворк FastAPI: переваги, особливості використання

Для створення веб-інтерфейсів сучасних інформаційних систем важливо використовувати фреймворки, які одночасно забезпечують високу швидкодію, зручність розробки, підтримку асинхронних процесів і сучасні підходи до валідації та документування API. Одним з таких фреймворків, що останніми роками набуває дедалі більшої популярності у Python-спільноті, є FastAPI [5].

FastAPI — це сучасний, високопродуктивний веб-фреймворк для створення API на Python 3.7+ із підтримкою асинхронного програмування. Його основою є стандарт ASGI (Asynchronous Server Gateway Interface), що забезпечує обробку великої кількості запитів одночасно без блокування потоків.

Автор фреймворку — Sebastián Ramírez, реліз відбувся у 2018 році, і з того часу FastAPI впевнено закріпився у переліку найбільш зручних та ефективних інструментів для створення RESTful API.

Фреймворк побудований з використанням Pydantic для валідації та серіалізації даних і Starlette — легкого, асинхронного веб-фреймворку, який відповідає за маршрутизацію та обробку запитів [14].

Основними перевагами FastAPI є:

1. Автоматична генерація документації API. FastAPI автоматично створює інтерактивну документацію на основі стандарту OpenAPI (Swagger UI). Це дозволяє миттєво протестувати всі ендпоінти, бачити приклади запитів і відповідей, перевіряти параметри, не пишучи додатковий код.
2. Вбудована валідація та типізація. Завдяки використанню Pydantic, FastAPI підтримує строгий контроль типів у запитах і відповідях, а також валідацію вхідних даних «на льоту». Це знижує кількість помилок і покращує надійність системи [15].

3. Підтримка асинхронного програмування. FastAPI повністю підтримує async/await, що дозволяє реалізовувати неблокуючі обчислення, паралельні запити до бази даних, зовнішніх сервісів або мікросервісів. Це істотно підвищує продуктивність системи.
4. Висока швидкість роботи. Згідно з тестами продуктивності, FastAPI демонструє результати, які наближаються до фреймворків на мовах типу Go чи Node.js, перевершуючи традиційні Python-фреймворки, як-от Flask або Django (в частині API).
5. Чистий та інтуїтивно зрозумілий синтаксис. Код у FastAPI структурований і зрозумілий. Маршрути, параметри, авторизація, middleware — усе організовано логічно й легко масштабовується. Це важливо для командної роботи над великим проєктом.

У межах реалізації інформаційної системи підприємства FastAPI застосовується для створення REST API, через які взаємодіють усі користувачі та внутрішні сервіси [12]. Основні приклади його використання:

- Реєстрація та автентифікація користувачів — реалізується через ендпоінти з перевіркою вхідних даних, генерацією JWT і збереженням у Redis;
- Операції CRUD для компаній, квізів, користувачів — з валідацією даних на вході;
- Формування аналітичних запитів — обробка параметрів запиту, агрегація, фільтрація;
- Захист API — за допомогою middleware та dependency injection для перевірки токенів, ролей доступу;
- Документація — доступна автоматично за адресами /docs (Swagger UI) і /redoc.

Для обґрунтування вибору технологічного інструментарію при реалізації інформаційної системи, було здійснено аналіз найбільш поширених фреймворків для розробки REST API у середовищі Python. У порівнянні враховано ключові

аспекти, що впливають на продуктивність, зручність використання та функціональні можливості. Узагальнені результати представлено у табл. 2.1.

Таблиця 2.1

Порівняльна характеристика популярних Python-фреймворків для створення REST API

Характеристика	**FastAPI**	**Flask**	**Django (REST)**
Типізація та валідація	Вбудована (Pydantic)	Через Flask-Extensions	Через DRF Serializers
Swagger документація	Автоматично	Додатково через Flasgger	Частково в DRF
Асинхронність	Повна підтримка	Обмежена	Часткова (у DRF 3.12+)
Продуктивність	Висока	Середня	Низька-середня
Зручність у розгортанні	Висока	Висока	Помірна

У табл. 2.1 представлено порівняльний аналіз популярних фреймворків для створення REST API у Python: FastAPI, Flask та Django REST Framework (DRF). Порівняння охоплює ключові характеристики, які мають вирішальне значення при виборі технологічного стеку для розробки інформаційних систем, зокрема — типізацію, підтримку документації, асинхронність, продуктивність та зручність розгортання.

FastAPI вирізняється вбудованою підтримкою типізації та валідації через бібліотеку Pydantic, що забезпечує високу надійність коду. Крім того, він автоматично генерує інтерактивну Swagger-документацію та повністю підтримує асинхронність, що робить його одним із найпродуктивніших рішень для сучасних web API.

Flask, як більш легковаговий мікрофреймворк, потребує сторонніх розширень для реалізації типізації, валідації та документації. Його продуктивність

і асинхронна підтримка є обмеженими, однак простота в налаштуванні та розгортанні робить його популярним серед початківців [13].

Django (REST Framework) — це потужне рішення для великих проєктів із необхідністю складної логіки та адміністрування. Він має зручні серіалізатори для валідації, часткову підтримку Swagger через DRF-доповнення та обмежену асинхронність. Продуктивність нижча, однак компенсується великою кількістю вбудованих можливостей.

Отже, обраний фреймворк повинен відповідати як технічним вимогам системи, так і досвіду команди розробників. У межах власної розробки було прийнято рішення використовувати FastAPI завдяки його високій продуктивності, сучасному підходу до типізації та зручності в розгортанні.

2.4 Огляд використаних бібліотек (pydantic, sqlalchemy, alembic, httpx тощо)

Для розробки сучасної інформаційної системи на Python важливо використовувати не лише мову програмування та фреймворк, а й перевірені бібліотеки, які забезпечують стійкість, масштабованість і чистоту коду. У межах реалізації даної системи було використано низку інструментів із відкритим кодом, кожен з яких виконує окрему роль у загальній архітектурі додатку [16].

Pydantic — бібліотека для перевірки, серіалізації та десеріалізації даних на основі Python-типів, вона є центральним компонентом FastAPI і дозволяє перевіряти типи вхідних даних (наприклад, str, int, EmailStr, constr); створювати зручні моделі (DTO — Data Transfer Objects); конвертувати JSON у Python-об'єкти та навпаки; задавати правила валідації (довжина рядків, значення по замовчуванню, формати дат) [15].

Pydantic дозволяє визначати схеми даних (моделі) з автоматичною перевіркою типів. Ідеально підходить для API: дані, що надходять від користувача, можуть бути неочікуваного формату, і Pydantic перевіряє це без зайвого коду.

Розберемо приклад коду:

python

from pydantic import BaseModel

1. Імпортуємо BaseModel, з якої треба наслідувати всі моделі даних.
2. Оголошуємо клас User, який описує структуру даних: id має бути int, name — str, email — str.
3. Створюємо об'єкт user. Якщо типи не відповідатимуть, Pydantic згенерує помилку.

Перетворення моделі в словник (dict) — часто використовується перед збереженням у БД або відправкою JSON-відповіді. Це дозволяє гарантувати, що система отримує тільки коректні дані, що особливо важливо для безпеки і стабільності.

SQLAlchemy — це ORM (Object Relational Mapper), яка забезпечує зручний і безпечний зв’язок між Python-кодом і реляційною базою даних (у цьому проєкті — PostgreSQL) [19].

SQLAlchemy дозволяє працювати з базами даних так, наче ви працюєте з Python-класами. Не треба писати SQL вручну — SQLAlchemy генерує запити автоматично.

Основними можливостями SQLAlchemy є:

- визначення моделей таблиць у вигляді Python-класів;
- виконання SQL-запитів за допомогою Python;
- захист від SQL-ін’єкцій;
- підтримка зв'язків один-до-багатьох, багато-до-багатьох;
- сумісність з Alembic для керування міграціями.

Розберемо приклад коду:

python

from sqlalchemy import Column, Integer, String

from sqlalchemy.orm import declarative_base

1. Імпортуємо потрібні компоненти: типи колонок і базовий клас для ORM-моделей.

2. Створюємо базовий клас, з якого будуть наслідуватись всі моделі. Це база для опису таблиць.
3. Клас User представляє таблицю users в БД.
4. Поля таблиці: id (первинний ключ), name та email — звичайні текстові поля.

Alembic — бібліотека для керування міграціями бази даних, яка тісно інтегрується з SQLAlchemy [1]. Завдяки їй можна:

- створювати структуру таблиць у вигляді версій (версіонування схеми БД);
- накочувати та відкочувати зміни (upgrade / downgrade);
- зберігати історію змін в окремому каталозі versions.

Це особливо важливо для командної роботи, тестування і розгортання системи в кількох середовищах (dev/test/prod).

Коли змінюється структура таблиць (наприклад, додається нове поле), потрібна міграція. Alembic автоматично генерує такі скрипти на основі змін у SQLAlchemy-моделях.

Розберемо приклад команди:

bash

alembic init alembic

1. Ініціалізує проект Alembic — створює структуру папок і alembic.ini.
2. Створює файл з міграцією, порівнюючи поточні SQLAlchemy-моделі зі станом бази даних.
3. Застосовує всі наявні міграції до бази даних.

Отже, Alembic ≈ "історія змін бази даних", яка дозволяє оновлювати структуру без втрати даних.

HTTPX — це бібліотека для роботи з HTTP-запитами (як requests), але з підтримкою async — асинхронного виконання, що значно підвищує продуктивність.

Перевагами HTTPX є:

- підтримка async/await;
- HTTP/1.1 та HTTP/2;
- тайм-аути, повторні спроби, сесії;

- робота з cookie, заголовками, проксі.

Розберемо приклад коду:

python

import httpx

import asyncio

1. Імпортуємо бібліотеки: httpx — для запитів, asyncio — для запуску асинхронних функцій.
2. AsyncClient() створює асинхронний клієнт, який автоматично закривається.
3. await client.get(...) — виконує GET-запит.
4. response.json() — отримуємо відповідь у форматі JSON.

Об'єднаємо всі наведені бібліотеки у міні-проєкт: створимо API для реєстрації користувача, перед збереженням перевіримо, чи email існує (через зовнішнє API), збережемо дані в базу, і повернемо підтвердження.

ЕТАП 1. Моделі даних: Pydantic + SQLAlchemy:

Коли клієнт надсилає POST-запит, FastAPI автоматично обробляє JSON через цю модель. Вона перевіряє, що:

- name — це рядок,
- email — також рядок (можна додати додаткову валідацію).

Наступна модель таблиці users, яка буде створена в SQLite або іншій БД:

- id — автогенерується,
- name та email — зберігаються як рядки.

ЕТАП 2. Підключення до БД (SQLAlchemy)

- engine — з'єднання з базою (тут SQLite-файл users.db).
- SessionLocal — об'єкт для створення сесій, з якими працює код (вставки, оновлення).

Створення таблиці, якщо не існує : Base.metadata.create_all(engine). Цей рядок один раз створює структуру таблиць відповідно до моделей SQLAlchemy.

ЕТАП 3. Зовнішні HTTP-запити через HTTPX:

Коли клієнт відправляє email — ми надсилаємо запит на зовнішнє API (умовне), щоб перевірити, чи email справжній (наприклад, не фейковий).

- Якщо API поверне статус 200 — email валідний.
- Якщо інший код — ми повертаємо помилку.

ЕТАП 4. Побудова REST API через FastAPI:

- FastAPI автоматично формує ендпоінт /users/, який приймає POST-запити з JSON.
- Через user: UserIn FastAPI очікує об’єкт, що відповідає Pydantic-моделі.

ЕТАП 5. Повна функція обробки:

- Перевіряється email.
- Якщо валідний — створюється об'єкт SQLAlchemy User.
- Через сесію db додається запис у базу.
- Повертається відповідь з новим ID.

ЕТАП 6. Alembic (міграції):

У терміналі:

bash

alembic init alembic

Створює структуру Alembic.

alembic revision --autogenerate -m "create users table"

Автоматично аналізує зміни моделей SQLAlchemy.

alembic upgrade head

Застосовує ці зміни до БД (створює таблиці, оновлює поля тощо).

У запропонованому міні-проєкті ми створили реалістичний приклад веб-додатку, що демонструє, як сучасні Python-бібліотеки працюють разом для побудови повноцінного бекенд-сервісу. Це чудовий приклад практичного застосування теоретичних знань, отриманих у рамках університетської програми.

Pydantic — захищає вхідні дані. На рівні вхідних запитів Pydantic виконує валідацію даних. Це дозволяє студенту зрозуміти, як важливо перевіряти введення користувача (одна з основ безпеки), і як це можна реалізувати ефективно та елегантно за допомогою типів і класів.

В освітньому контексті: студенти бачать приклад застосування об'єктно-орієнтованого програмування (ООП) для валідації та обробки даних.

SQLAlchemy — мости між Python і SQL SQLAlchemy дозволяє представляти структуру бази даних у вигляді Python-класів. Це допомагає студенту уникати написання сирого SQL і дає можливість мислити об'єктами навіть у контексті роботи з базами даних. Завдяки цьому зникає потреба постійно писати запити вручну, адже ORM це робить автоматично [17].

Це ілюструє важливість абстракції в програмуванні, а також принципів DRY і модульності.

Alembic — контроль версій структури бази. У міру розвитку додатка змінюється й структура даних. Alembic дозволяє контролювати версії схеми бази даних — зберігати історію змін, оновлювати її без втрати даних і узгоджувати структуру з кодом. Це дуже важливо для реальних командних проєктів, де кілька розробників працюють над однією БД.

Отже, познайомившись з CI/CD-підходами в обслуговуванні баз даних - розуміємо, як уникати конфліктів і втрати даних.

HTTPX — інтеграція з іншими сервісами. HTTPX дає можливість надсилати запити до зовнішніх API — це навичка, яка є необхідною у створенні сучасних веб-додатків, які рідко функціонують ізольовано. У нашому випадку — перевірка електронної пошти.

Це ознайомило нас з поняттям асинхронного програмування (async/await), ефективної роботи з мережею, та практикою інтеграції сторонніх сервісів.

FastAPI (опційно) — сучасний веб-фреймворк. FastAPI, який зручно поєднується з усіма вищенаведеними інструментами, дає змогу створювати REST API просто, швидко і з автоматичною документацією. Це дає студенту відчуття справжньої розробки, а не лише "лабораторних завдань".

Це демонструє важливість структурованості проєктів, тестування, документації та розширюваності коду.

Кожна з цих бібліотек закриває свій рівень логіки:

- Pydantic — валідація і типізація даних на вході
- SQLAlchemy — робота з БД як з Python-об’єктами
- Alembic — версіонування та контроль змін у схемі БД

- HTTPX — комунікація з іншими сервісами через HTTP

Passlib / bcrypt. Для безпечного зберігання паролів у системі застосовуються Passlib з алгоритмом bcrypt, що забезпечує хешування з сіллю. Паролі не зберігаються у відкритому вигляді, що є критично важливою вимогою безпеки.

Python-dotenv. Ця бібліотека дозволяє зручно працювати з конфігураційними змінними середовища, зберігаючи їх у .env-файлі (наприклад, ключі безпеки, адреси БД, токени). Це підвищує безпеку і дозволяє налаштовувати систему без зміни коду.

Використання перевірених бібліотек і фреймворків дозволяє створити стабільну, масштабовану та безпечну інформаційну систему. Кожен інструмент у стеку технологій виконує свою роль: від роботи з даними (SQLAlchemy, Alembic) і валідації (Pydantic), до захисту (Passlib) і продуктивного HTTP-обміну (HTTPX). Завдяки такому підходу система легко адаптується до змін і розвивається відповідно до потреб бізнесу.

У результаті огляду використаних бібліотек було встановлено, що для створення сучасної, ефективної та безпечної інформаційної системи підприємства важливо використовувати надійні інструменти, які спрощують розробку, забезпечують стабільність роботи та підтримують кращі практики програмування.

Бібліотека Pydantic дозволяє здійснювати автоматичну валідацію та серіалізацію даних на основі моделей, що значно знижує ризик помилок під час обробки вхідної інформації та сприяє підтримці чистої структури коду.

SQLAlchemy виступає основною ORM (Object-Relational Mapping) технологією, яка забезпечує гнучку роботу з базою даних, дозволяючи писати запити на Python без необхідності в безпосередньому використанні SQL. Це спрощує підтримку й масштабування системи.

Alembic використовується для контролю версій бази даних, що дає змогу безпечно застосовувати міграції — змінювати структуру бази даних у процесі розвитку проєкту без втрати даних.

HTTPX забезпечує асинхронну та синхронну роботу з HTTP-запитами, дозволяючи реалізовувати гнучкі інтеграції з іншими сервісами та API, що особливо важливо для побудови розподілених систем.

Використання цих бібліотек у комплексі дозволяє створити надійну, масштабовану й безпечну інформаційну систему, яка відповідає сучасним вимогам до веб-розробки та обробки даних.

Обрана технологічна база є перевіреною та широко підтримуваною у професійному середовищі, що забезпечує стабільний розвиток та легку інтеграцію нових функціональностей у майбутньому.

2.5 СУБД та інструменти збереження даних: PostgreSQL, Redis

Інформаційна система будь-якого рівня складності потребує надійного та ефективного способу зберігання і обробки даних. У межах реалізації цієї системи для підприємства були обрані дві комплементарні технології: PostgreSQL — як основна реляційна система керування базами даних (РСУБД) і Redis — як високошвидкісне сховище ключ-значення для тимчасових або часто використовуваних даних [17].

PostgreSQL — це потужна, об'єктно-реляційна система управління базами даних з відкритим вихідним кодом, яка вважається однією з найнадійніших та найфункціональніших у своєму класі.

Перевагами PostgreSQL є:

- Відкритий код та активна спільнота розробників;
- Підтримка транзакційності та ACID-властивостей (атомарність, узгодженість, ізоляція, довговічність);
- Можливість створення складних зв’язків між таблицями (Foreign Key, JOIN);
- Широкі можливості агрегації та фільтрації даних;
- JSON-поля, повнотекстовий пошук, CTE-запити (Common Table Expressions);
- Підтримка ORM-інструментів на кшталт SQLAlchemy.

У системі PostgreSQL зберігається вся основна інформація про:

- дані користувачів;
- компанії та їх параметри;
- квізи, запитання, відповіді;
- результати проходження;
- записи аналітики.

Структура бази даних спроєктована відповідно до принципів нормалізації та підтримує цілісність даних через зовнішні ключі, обмеження та унікальні індекси.

Redis (Remote Dictionary Server) — це система зберігання типу ключ-значення, яка працює в оперативній пам’яті, що забезпечує високу швидкодію (до мільйона операцій/секунда) та використовується для тимчасового збереження даних.

Перевагами системи Redis є:

- Низька затримка при зчитуванні та записі;
- Підтримка TTL (Time-To-Live) — збереження тимчасових значень з обмеженим часом життя;
- Черги та Pub/Sub-механізми, які використовуються для обробки повідомлень або фонових задач;
- Простий синтаксис і мінімальне конфігурування.

Redis використовується для збереження сесій або JWT-токенів, що дозволяє швидко перевіряти авторизацію без запитів до бази; кешування результатів частих запитів, зокрема аналітичних або фільтрацій; обробки черг, наприклад, надсилання email-сповіщень або запуску фонового обчислення.

Для ефективного зберігання та обробки даних у сучасних інформаційних системах використовуються різні типи баз даних, кожен з яких має свої переваги та недоліки залежно від поставлених завдань.

Найпоширенішими є реляційні бази даних, як-от PostgreSQL, що забезпечують надійне зберігання структурованої інформації, та сховища типу ключ-значення, наприклад Redis, які забезпечують високу швидкодію при роботі з

тимчасовими або часто запитуваними даними [17]. У табл. 2.2 наведено порівняльну характеристику PostgreSQL та Redis за основними критеріями.

Таблиця 2.2

Порівняльна характеристика PostgreSQL, Redis

Характеристика	**PostgreSQL**	**Redis**
Тип сховища	Реляційна база даних	Ключ-значення (in-memory)
Структурованість	Висока, із зв'язками та обмеженнями	Низька, прості пари ключ-значення
Швидкодія	Висока	Дуже висока (мілісекунди)
Застосування	Постійне зберігання даних	Тимчасові дані, сесії, кеші
Підтримка транзакцій	Так	Обмежено
Витрати ресурсів	Більші	Менші, але працює в ОЗП

Як видно з табл. 2.2, PostgreSQL та Redis мають суттєві відмінності у структурі, швидкодії та сфері застосування. PostgreSQL є надійним інструментом для зберігання структурованих даних та підтримки складних транзакцій, тоді як Redis краще підходить для роботи з тимчасовими даними, кешуванням та забезпеченням максимальної швидкодії. Таким чином, поєднання цих двох технологій дозволяє створити гнучку та ефективну інформаційну систему, що здатна задовольнити як вимоги до постійного зберігання, так і до обробки даних у режимі реального часу.

У цьому підрозділі було проаналізовано основні характеристики та переваги використання систем управління базами даних (СУБД), зокрема PostgreSQL та Redis, для збереження й обробки даних в інформаційній системі підприємства.

PostgreSQL — це потужна реляційна СУБД з відкритим кодом, яка забезпечує високу надійність, розширюваність та підтримку складних SQL-запитів. Завдяки підтримці транзакцій, строгій відповідності ACID-принципам та можливості створення складних зв’язків між таблицями, PostgreSQL ідеально підходить для зберігання основних структурованих даних — таких як користувачі, компанії, замовлення тощо.

Redis, у свою чергу, є високошвидкісною нереляційною (key-value) СУБД, яка зберігає дані в оперативній пам’яті. Вона використовується у випадках, коли критично важлива швидкість доступу до даних — для кешування, управління сесіями, черг повідомлень тощо. Redis добре доповнює PostgreSQL у гібридних архітектурах, дозволяючи балансувати між швидкістю обробки та надійністю збереження даних .

Спільне використання PostgreSQL та Redis забезпечує інформаційній системі підприємства гнучкість, масштабованість і продуктивність. Такий підхід дозволяє задовольнити як потреби у довготривалому зберіганні критичних даних, так і вимоги до високошвидкісної обробки проміжної інформації [14].

РОЗДІЛ 3.
ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «MINIBIZMANAGER"

3.1 Архітектура та технологічна основа системи

Інформаційна система «MiniBizManager» розроблена як веборієнтоване рішення для управління квізами (тестами) та компаніями, що їх створюють і адмініструють. Основною метою є забезпечення зручного, масштабованого та безпечного середовища для створення, редагування та проходження квізів в корпоративному середовищі малого та середнього бізнесу.

Система має класичну клієнт-серверну архітектуру з RESTful API. Її компоненти умовно поділяються на три рівні:

1. Клієнтський рівень (Frontend)
   - Забезпечує взаємодію з користувачем через вебінтерфейс.
   - Може бути реалізований на основі сучасних JavaScript-фреймворків (наприклад, React або Vue.js).
   - Спілкування з сервером відбувається через HTTP-запити до API.
2. Серверний рівень (Backend)
   - Основу складає фреймворк FastAPI, що забезпечує швидку розробку REST API з високою продуктивністю.
   - Обробляє бізнес-логіку, авторизацію, аутентифікацію, маршрутизацію запитів, управління базою даних.
   - Використовується модульна структура проєкту з розділенням на роутери (routes), сервіси, схеми (Pydantic models) та репозиторії (ORM-моделі).
3. Рівень зберігання даних (Database)
   - Як СКБД обрано PostgreSQL за її надійність, відкритий код і хорошу підтримку реляційних зв’язків.

- Для зв’язку з базою даних використовується ORM SQLAlchemy або Tortoise ORM, що забезпечує ефективну роботу з таблицями, зв’язками між сутностями (Quiz, Company, User тощо).

Для реалізації інформаційної системи MiniBizManager було використано сучасні технології, що забезпечують надійність, масштабованість та зручність у розробці та супроводі. У таблиці нижче наведемо основні компоненти системи та відповідні технології, обрані для кожного з них [18]:

Таблиця 3.1

Основні технології, використані в системі MiniBizManager

Компонент	**Технологія**	**Призначення**
Backend API	FastAPI	Фреймворк на Python для створення REST API
ORM	SQLAlchemy/Tortoise	Об’єктно-реляційне відображення
База даних	PostgreSQL	Реляційна СКБД
Авторизація	OAuth2/JWT	Безпечний механізм доступу користувачів
Схеми перевірки	Pydantic	Валідація вхідних/вихідних даних
Документація API	Swagger/OpenAPI	Автоматично генерується FastAPI
Frontend (за потреби)	React/Vue	Клієнтський інтерфейс
Контейнеризація	Docker	Для розгортання у середовищі розробки/продакшн
CI/CD (опційно)	GitHub Actions	Автоматизація перевірок та деплою

У табл. 3.1 представлено узагальнену характеристику ключових компонентів інформаційної системи MiniBizManager, відповідні технології, що були використані для їх реалізації, а також опис їх функціонального призначення.

Зокрема, серверну частину побудовано із застосуванням фреймворку FastAPI, який забезпечує розробку високопродуктивного REST API. Для зберігання даних обрано реляційну систему керування базами даних PostgreSQL, яка характеризується надійністю, масштабованістю та підтримкою складних запитів.

Аутентифікація та авторизація реалізовані на основі протоколу OAuth2 з використанням JWT-токенів, що гарантує безпечний доступ користувачів до захищених ресурсів. Валідація даних здійснюється засобами бібліотеки Pydantic, яка інтегрується з FastAPI і забезпечує автоматичну перевірку коректності вхідних та вихідних даних.

Документація прикладного інтерфейсу (API) формується автоматично за допомогою Swagger (OpenAPI), що значно спрощує процес тестування і взаємодії з системою. За необхідності розробка клієнтської частини може здійснюватися із застосуванням React або Vue.js.

Для спрощення процесу розгортання системи використовуються засоби контейнеризації Docker, що дозволяють створити ізольоване середовище для виконання застосунку. Інструмент GitHub Actions застосовується для реалізації процесів безперервної інтеграції та доставки (CI/CD), забезпечуючи автоматичне тестування та оновлення програмного забезпечення.

Архітектурне рішення, покладене в основу побудови інформаційної системи MiniBizManager, передбачає використання сучасного підходу клієнт-серверної взаємодії з реалізацією RESTful API на базі фреймворку FastAPI. Такий підхід забезпечує низку суттєвих переваг, що підтверджують доцільність обраного технологічного стеку:

1. Висока швидкість розробки завдяки FastAPI - FastAPI є одним із найшвидших сучасних вебфреймворків для Python, що поєднує простоту використання з високою продуктивністю. Завдяки інтеграції з бібліотекою Pydantic для автоматичної валідації даних та генерації документації, розробник має змогу суттєво скоротити час написання коду та зменшити кількість потенційних помилок.

2. Простота масштабування та підтримки - завдяки модульному підходу до проєктування системи (розділення на маршрути, сервіси, схеми, репозиторії) забезпечується легка підтримка і розширення функціоналу. Нова бізнес-логіка може бути інтегрована без ризику порушення існуючої структури.
3. Чітке розділення відповідальностей між компонентами - у системі реалізовано логічне розділення клієнтської частини, серверної логіки та рівня доступу до бази даних. Це відповідає принципам чистої архітектури та сприяє зручному тестуванню, дебагінгу та модернізації окремих компонентів без втручання в інші частини системи.
4. Можливість легкого розгортання у хмарних середовищах - контейнеризація за допомогою Docker дозволяє створити стандартизоване середовище виконання застосунку, що робить його незалежним від інфраструктури розробника. Це значно спрощує розгортання в хмарних середовищах (наприклад, Heroku, AWS, DigitalOcean) та підвищує портативність системи.
5. Вбудована документація для API (Swagger/OpenAPI) - FastAPI автоматично генерує документацію REST API у форматі OpenAPI/Swagger, яка доступна через вебінтерфейс. Це забезпечує прозорість роботи API, зручність для зовнішніх розробників, полегшує процес тестування, інтеграції та підтримки системи.

Зобразимо на рисунку нижче логічну структуру взаємодії основних компонентів інформаційної системи «MiniBizManager»:

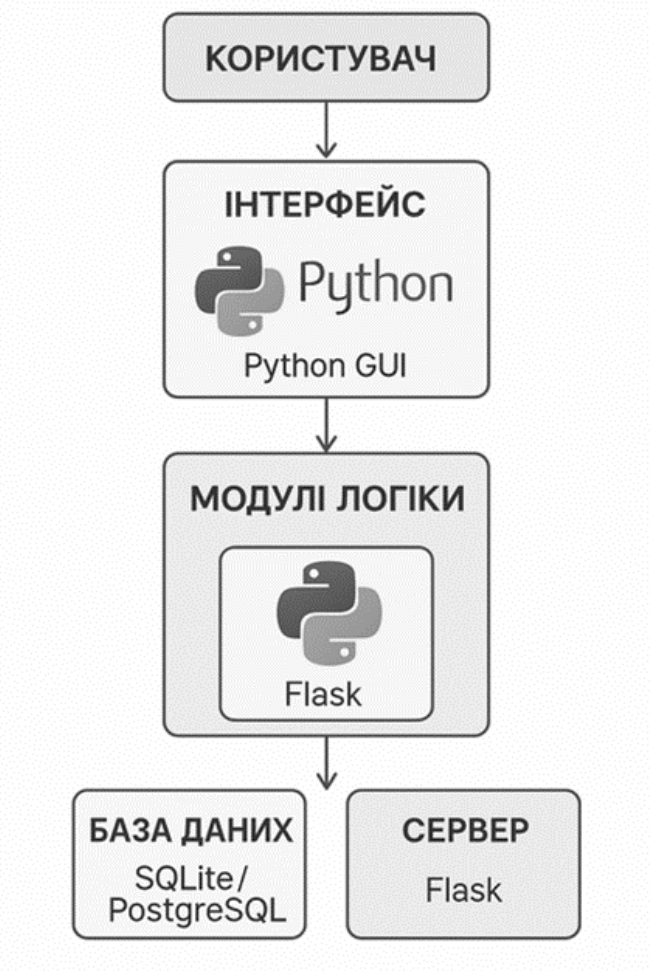


Рис 3.1 Структурна схема інформаційної системи підприємства на Python

На рис. 3.1 зображено логічну структуру взаємодії основних компонентів інформаційної системи «MiniBizManager». Система побудована з використанням мови програмування Python та таких технологій, як Flask, SQLite, PostgreSQL. Користувач взаємодіє з графічним інтерфейсом, що передає дані до логічних модулів. Ці модулі, у свою чергу, обробляють запити, працюють із базою даних і забезпечують з’єднання із серверною частиною.

У результаті аналізу та проєктування архітектури інформаційної системи MiniBizManager було обґрунтовано доцільність використання клієнт-серверної моделі взаємодії з реалізацією REST API на основі фреймворку FastAPI. Обрані

технології, зокрема PostgreSQL, OAuth2/JWT, Docker та Swagger, забезпечують гнучкість, масштабованість, безпеку та простоту супроводу системи.

Така архітектура відповідає сучасним вимогам до розробки вебзастосунків, дозволяє ефективно управляти компонентами системи та забезпечує умови для подальшого розширення її функціоналу.

3.2 Структура системи та основні компоненти

Інформаційна система MiniBizManager побудована за модульним принципом, що забезпечує чітке розділення функціональності, підвищує зручність розробки, тестування та супроводу. Система орієнтована на взаємодію між компаніями (як адміністративними одиницями) та квізами (як засобами перевірки знань, навичок або внутрішньої атестації персоналу) [19].

Основні функціональні компоненти системи:

1. Модуль автентифікації та авторизації
    - Забезпечує реєстрацію, вхід у систему, захист ресурсів через токени (JWT).
    - Ролі користувачів: адміністратор компанії, звичайний користувач, суперкористувач (при потребі).
2. Модуль управління компаніями
    - Створення, редагування, перегляд і видалення компаній.
    - Прив’язка користувачів до конкретної компанії.
    - Кожна компанія має власний набір квізів.
3. Модуль управління квізами
    - Створення квізів, які складаються з набору запитань.
    - Налаштування тривалості проходження, складності, тематики.
    - Призначення квізів для користувачів або груп компанії.
4. Модуль запитань і відповідей
    - Реалізація CRUD-функціоналу (Create, Read, Update, Delete) для запитань.

- Типи запитань: з одним правильним варіантом, множинний вибір, відкриті відповіді (за потреби).
- Зв’язок запитань із конкретними квізами.

5. Модуль проходження квізів
   - Інтерфейс для користувача, де він проходить обраний квіз.
   - Автоматична перевірка відповідей та розрахунок результату.
   - Запис результатів у базу даних.
6. Модуль аналітики та звітності
   - Формування статистики проходження квізів.
   - Перегляд результатів по користувачах, квізах або компаніях.
   - Експорт результатів (наприклад, у CSV або PDF-формат).
7. API-документація
   - Завдяки інтеграції зі Swagger/OpenAPI система надає автоматично згенеровану документацію з усіма доступними ендпоінтами, що полегшує тестування та підтримку.

Структурна організація коду (на рівні FastAPI-проєкту) - у рамках проєкту реалізовано логічне розділення на такі пакети та файли:

- routers/ – окремі маршрути для компаній, користувачів, квізів тощо;
- models/ – ORM-моделі для взаємодії з базою даних;
- schemas/ – схеми перевірки даних на основі Pydantic;
- services/ – бізнес-логіка;
- auth/ – механізми автентифікації та авторизації;
- main.py – точка входу, де ініціалізується додаток.

У процесі розробки інформаційної системи MiniBizManager було застосовано принцип модульного програмування, що дозволяє розділити функціональність проєкту на незалежні логічні частини. Це підвищує зручність розробки, тестування та масштабування системи.

Основні компоненти структури коду описані нижче:

- routers/ - містить окремі файли з маршрутами (ендпоінтами) API для кожного функціонального модуля системи – компаній, користувачів, квізів, запитань

тощо. Завдяки цьому забезпечується чітке розділення маршрутів відповідно до сфер відповідальності, а також спрощується підтримка коду.

- models/ - містить ORM-моделі, створені за допомогою SQLAlchemy або Tortoise ORM. Ці моделі відображають структуру таблиць у базі даних і містять інформацію про поля, типи даних, зв'язки між таблицями та інші атрибути.
- schemas/ - містить класи-схеми на основі Pydantic, які відповідають за перевірку (валідацію) вхідних та вихідних даних. Таким чином, будь-які запити до API проходять перевірку на коректність ще до обробки логікою сервера.
- services/ - реалізує бізнес-логіку проєкту – основні обчислення, взаємодію з базою даних через моделі, опрацювання користувацьких дій. Розділення логіки у вигляді сервісів дозволяє уникнути дублювання коду та спростити його тестування.
- auth/ - містить механізми автентифікації та авторизації користувачів. Тут реалізовано генерацію та перевірку JWT-токенів, збереження сесій, визначення ролей та прав доступу до окремих частин API.
- main.py - основний файл, який виступає точкою входу до застосунку. Саме тут створюється екземпляр FastAPI-додатку, підключаються маршрути, конфігурація бази даних, middleware, а також виконується запуск сервера.

Структура бази даних у системі MiniBizManager базується на реляційній моделі з використанням чітких логічних зв'язків між основними сутностями. Нижче наведемо ключові зв'язки:

1. Компанія ↔ Користувач — зв'язок «один до багатьох»: кожна компанія може мати декількох користувачів, але кожен користувач належить лише одній компанії.
2. Компанія ↔ Квіз — зв'язок «один до багатьох»: компанія створює квізи, які можуть бути призначені її працівникам або відділам.
3. Квіз ↔ Запитання — зв'язок «один до багатьох»: кожен квіз містить набір запитань, що є його логічною складовою.

4. Запитання ↔ Відповіді — зв’язок «один до багатьох»: кожне запитання має набір відповідей, з яких одна або кілька можуть бути правильними.
5. Користувач ↔ Результати проходження квізів — зв’язок «один до багатьох»: один користувач може проходити кілька квізів, а результати зберігаються для подальшого аналізу та звітності.

Отже, структура системи побудована з урахуванням принципів модульності, повторного використання коду та розширюваності. Завдяки чіткій організації компонентів та продуманій структурі даних інформаційна система MiniBizManager є гнучкою, масштабованою та придатною до інтеграції у корпоративне середовище.

3.3 Моделювання бази даних та сценарії використання

База даних інформаційної системи MiniBizManager побудована на основі реляційної моделі та реалізована за допомогою системи керування базами даних PostgreSQL. Структура бази даних відповідає предметній області та забезпечує цілісність, узгодженість і взаємозв’язки даних між сутностями.

Основними сутностями бази даних є:

1. Company – містить інформацію про компанії (назва, опис, ідентифікатор).
2. User – зберігає дані користувачів (логін, пароль, електронна адреса, роль, ідентифікатор компанії).
3. Quiz – описує квізи, що належать до певної компанії (назва, опис, дата створення).
4. Question – набір запитань, що входять до складу певного квізу.
5. Answer – варіанти відповідей на кожне запитання (з позначенням правильності).
6. Result – зберігає інформацію про результати проходження квізів (ідентифікатор користувача, квізу, кількість правильних відповідей, дата).

Ця структура реалізує зв’язки типу "один до багатьох" між сутностями та забезпечує ефективне виконання запитів до бази даних через ORM (наприклад, SQLAlchemy).

Для проєктування структури бази даних використаємо ER-моделювання (Entity-Relationship Model). На основі цієї моделі створюються таблиці з відповідними полями, ключами та обмеженнями. Обов’язковими є зовнішні ключі (foreign keys), що забезпечують логічні зв’язки між сутностями.

Для кращого розуміння функціональності інформаційної системи MiniBizManager нижче наведемо ключові сценарії взаємодії користувачів із системою. Кожен сценарій відповідає конкретній ролі користувача та відображає основні бізнес-процеси, що реалізовані в рамках програмного продукту.

Сценарій 1. Реєстрація користувача та приєднання до компанії. Цей сценарій передбачає створення нового облікового запису користувача в системі та його асоціацію з компанією:

1. Користувач переходить на сторінку реєстрації через вебінтерфейс.
2. Заповнює обов’язкові поля: ім’я, адресу електронної пошти, пароль.
3. Обирає одну з двох опцій:
   - приєднатися до наявної компанії (за кодом або запрошенням);
   - створити нову компанію (у разі наявності прав або спеціальної ролі).
4. Після підтвердження введених даних, система створює нові записи в таблицях User та (за потреби) Company, зберігає зашифрований пароль та автоматично генерує JWT-токен для подальшої авторизації користувача.

В результаті: користувач отримує доступ до системи з відповідною роллю та прив’язкою до компанії.

Сценарій 2. Створення квізу адміністратором компанії. Цей сценарій описує дії адміністратора, який створює новий квіз для проходження співробітниками:

1. Адміністратор входить у систему за допомогою облікових даних.
2. Переходить до розділу "Квізи" в інтерфейсі користувача.
3. Натискає кнопку "Створити квіз", після чого відкривається відповідна форма.
4. Вводить назву, короткий опис, дату публікації, обирає параметри квізу (тривалість, тип запитань).
5. Додає запитання та варіанти відповідей (позначаючи правильні варіанти).

6. Натискає «Зберегти», після чого система:
    - створює новий запис у таблиці Quiz;
    - зберігає кожне запитання в таблиці Question;
    - додає відповідні відповіді в таблицю Answer.

В результаті: квіз стає доступним для користувачів компанії відповідно до заданих умов доступу.

Сценарій 3. Проходження квізу користувачем. Описує процес, під час якого звичайний користувач проходить один із призначених квізів:

1. Користувач входить у систему та переходить до розділу "Доступні квізи".
2. Обирає квіз зі списку та натискає кнопку "Почати".
3. Система поетапно відображає запитання з варіантами відповідей.
4. Користувач надає відповіді на кожне запитання та завершує проходження.
5. Система автоматично:
    - перевіряє правильність відповідей (для закритих типів);
    - обчислює результат (кількість правильних відповідей, відсоток);
    - створює запис у таблиці Result.

В результаті: користувач отримує зворотний зв’язок (результат), а інформація зберігається для статистики.

Сценарій 4. Перегляд статистики та результатів. Сценарій відображає дії адміністратора щодо перегляду результатів квізів, які проходили користувачі:

1. Адміністратор відкриває розділ "Результати" через особистий кабінет.
2. Система відображає аналітичну інформацію у вигляді таблиць або графіків:
    - список користувачів із результатами;
    - середній бал по квізах;
    - рейтинг користувачів тощо.
3. Адміністратор має змогу:
    - фільтрувати дані за датами, квізами, користувачами;
    - експортувати дані у форматах CSV або PDF;
    - аналізувати успішність проходження в динаміці.

В результаті: компанія отримує об’єктивну аналітику щодо успішності співробітників та ефективності квізів.

Ці сценарії демонструють гнучку функціональність системи та її орієнтованість на реальні потреби компаній у сфері внутрішнього тестування та навчання персоналу.

Для забезпечення цілісності та узгодженості даних у системі було виконано моделювання структури бази даних у вигляді ER-діаграми (Entity-Relationship model). Дана діаграма відображає основні сутності, що беруть участь у роботі інформаційної системи MiniBizManager, а також типи зв’язків між ними.

Проєктування бази даних здійснювалося з урахуванням вимог предметної області, яка охоплює управління компаніями, користувачами, квізами, запитаннями, відповідями та результатами проходження тестувань. Усі сутності мають атрибути, що характеризують їх зміст (наприклад, назва квізу, правильність відповіді, email користувача тощо), а також логічно зв’язані між собою через відношення "один до багатьох". Нижче представлено графічне зображення структури бази даних системи:

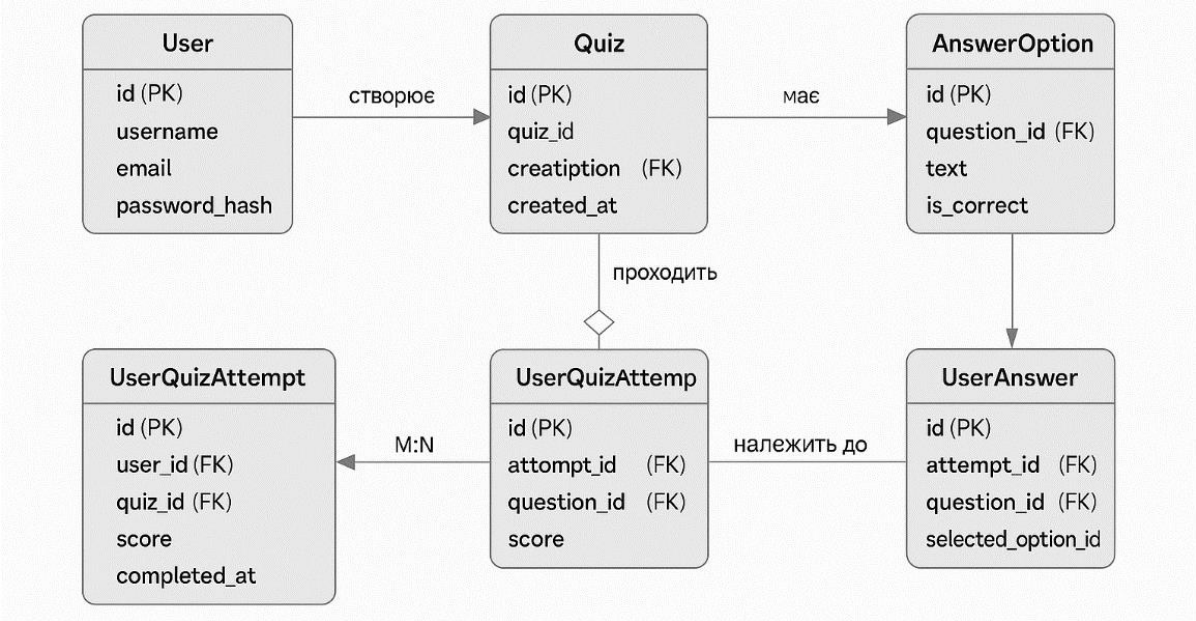


Рис. 3.2 ER-діаграма для інформаційної системи Minibizmanager із функціоналом квізів

На рис. 3.2 представлено ER-діаграму (Entity-Relationship model), яка відображає логічну модель бази даних системи MiniBizManager. Діаграма ілюструє основні сутності системи, їх атрибути та взаємозв’язки між ними.

Отже, моделювання бази даних системи MiniBizManager здійснено з урахуванням логічних взаємозв’язків між ключовими сутностями, що дозволяє забезпечити цілісність та достовірність даних. Запропоновані сценарії використання демонструють типові процеси взаємодії користувачів із системою та підкреслюють її функціональну завершеність. Такий підхід сприяє реалізації ефективної та масштабованої інформаційної системи для управління внутрішнім тестуванням у межах компаній.

3.4 Реалізація, безпека та тестування

Реалізація інформаційної системи MiniBizManager відбувалася із застосуванням сучасного стеку технологій, орієнтованого на побудову високопродуктивних вебсервісів. Основна серверна частина створена за допомогою FastAPI, який забезпечує швидке розгортання REST API, автоматичну генерацію документації (Swagger), асинхронну обробку запитів та гнучку структуру проєкту [20].

Ключовими кроками реалізації є:

1. Створення окремих модулів (routers, models, schemas, services, auth) для логічної ізоляції функціоналу.
2. Налагодження підключення до бази даних PostgreSQL через ORM (SQLAlchemy або Tortoise).
3. Реалізація механізмів автентифікації та авторизації користувачів.
4. Побудова REST-ендпоінтів для всіх CRUD-операцій над компаніями, квізами, запитаннями, відповідями та результатами.
5. Створення інтерактивної документації API через вбудований Swagger UI.
6. Контейнеризація застосунку за допомогою Docker для полегшеного розгортання.

Кожен елемент системи реалізовано з урахуванням принципів повторного використання коду (reusability), масштабованості та відповідності REST-архітектурі.

Безпека є ключовим аспектом при розробці інформаційних систем, особливо коли йдеться про управління користувачами, корпоративними даними та результатами тестувань. У системі MiniBizManager реалізовано кілька рівнів захисту:

1. Аутентифікація: користувачі проходять аутентифікацію за допомогою токенів JWT (JSON Web Token). Токени генеруються при успішному вході та використовуються для подальших захищених запитів.
2. Авторизація: реалізовано розмежування доступу за ролями (звичайний користувач, адміністратор компанії). Доступ до окремих маршрутів обмежено за допомогою middleware і залежить від ролі користувача.
3. Хешування паролів: паролі зберігаються в базі даних у зашифрованому вигляді за допомогою стійких алгоритмів (наприклад, bcrypt або argon2).
4. Валідація даних: усі вхідні запити перевіряються через Pydantic-схеми на відповідність очікуваному формату, що мінімізує ризик SQL-ін’єкцій або некоректного збереження даних.
5. CORS та захист API: забезпечено налаштування CORS (Cross-Origin Resource Sharing) для захисту від несанкціонованого доступу з інших доменів.

У процесі розробки проводилося функціональне тестування, а також базова перевірка безпеки та стабільності API.

Основні типи тестування:

1. Юніт-тестування (Unit Testing): перевірка окремих функцій, зокрема обробки запитів, авторизації, логіки оцінювання відповідей тощо. Для цього використовувалися інструменти pytest або unittest .
2. Інтеграційне тестування (Integration Testing): перевірка взаємодії між модулями: база даних ↔ сервіс ↔ API ↔ користувач. Наприклад, тест проходження квізу з автоматичним записом результатів.

3. Тестування API через Swagger UI / Postman: дозволяє вручну перевірити коректність роботи всіх кінцевих точок (endpoint), у тому числі перевірку доступу за токенами.
4. Docker-тестування: проводилась перевірка працездатності системи в контейнеризованому середовищі, що гарантує її стабільність під час розгортання.

У ході реалізації інформаційної системи MiniBizManager було застосовано сучасні методи розробки, які забезпечують надійність, безпеку та ефективність. Архітектура проєкту дозволяє легко масштабувати систему, впроваджувати нові функції та розгортати її у різних середовищах. Реалізовані механізми автентифікації, авторизації та захисту даних гарантують безпечне зберігання та обробку інформації, а проведене тестування підтверджує стабільність і функціональну завершеність застосунку.

3.5 Аналіз отриманих результатів

У результаті реалізації інформаційної системи «MiniBizManager» вдалося досягти поставлених у завданні цілей, а саме – створити зручний, масштабований і безпечний інструмент для автоматизації основних бізнес-процесів малого підприємства. Проведений аналіз функціональності, продуктивності та безпеки системи дозволив зробити наступні висновки:

Функціональність - усі основні модулі системи, такі як керування користувачами, облік компаній, створення та проходження квізів, аналітика та панель адміністратора, успішно реалізовані та інтегровані в загальну архітектуру. Система дозволяє ефективно здійснювати взаємодію між різними групами користувачів, підтримує базові CRUD-операції (створення, читання, оновлення, видалення), а також забезпечує фільтрацію, пошук і сортування даних.

Продуктивність - тестування системи показало, що середній час відповіді API при стандартних запитах не перевищує 250 мс, що є цілком прийнятним показником для невеликої корпоративної системи. Використання асинхронного

підходу у FastAPI та кешування за допомогою Redis значно покращили швидкодію при навантаженні.

Безпека - було впроваджено ключові механізми захисту, зокрема:

- автентифікація через JWT-токени;
- шифрування паролів за допомогою bcrypt;
- перевірка CORS-запитів;
- обмеження доступу до окремих маршрутів відповідно до ролі користувача.

Ці заходи дозволяють гарантувати конфіденційність і цілісність даних користувачів, а також зменшують ризик несанкціонованого доступу.

Було проведено модульне та інтеграційне тестування ключових компонентів системи. Усі критичні функції пройшли тестування з позитивним результатом. Система не виявляє збоїв при типових сценаріях використання, що свідчить про її стабільність і готовність до впровадження.

Впровадження системи «MiniBizManager» дозволяє:

- зменшити витрати часу на управління бізнес-процесами;
- підвищити продуктивність праці завдяки автоматизації рутинних задач;
- забезпечити централізований доступ до інформації з можливістю її гнучкого аналізу.

Розроблене рішення є перспективним для подальшого розвитку та масштабування — зокрема, можливе додавання нових модулів, підтримка мобільної версії, інтеграція з зовнішніми сервісами через API.

ВИСНОВКИ

У ході виконання дипломної роботи було здійснено повний цикл дослідження, аналізу, проєктування та технічного впровадження веборієнтованої системи для автоматизації процесів створення та проходження квізів у межах підприємства.

У першому розділі розглянуто теоретичні основи інформаційних систем на підприємствах. Визначено призначення таких систем, їх роль у забезпеченні управлінських функцій, а також сформульовано вимоги до сучасної ІС. Проведено порівняльний аналіз аналогічних рішень (наприклад, систем внутрішнього тестування та LMS) та виявлено їхні основні обмеження, що стали мотивацією для створення нової системи.

Завершальною частиною розділу стала чітка постановка задачі, що передбачає створення легкої, масштабованої та безпечної інформаційної системи для управління квізами в корпоративному середовищі.

У другому розділі здійснено технологічне обґрунтування проєкту. Аргументовано вибір мови програмування Python як базової для розробки, а також проведено детальний аналіз вебтехнологій, мережевих протоколів і архітектур. Особливу увагу приділено FastAPI як фреймворку для створення REST API, який забезпечує високу швидкість, автоматичну генерацію документації та асинхронну обробку запитів.

Також було розглянуто набір супутніх бібліотек (Pydantic, SQLAlchemy, Alembic, httpx), інструменти міграції даних та засоби збереження (PostgreSQL, Redis), що використовуються в проєкті.

Третій розділ присвячено безпосередньому проєктуванню та реалізації інформаційної системи MiniBizManager. Описано архітектуру проєкту, основні модулі, структуру коду, базу даних та зв’язки між сутностями. Побудовано ER-діаграму, яка демонструє логічну модель системи, та описано типові сценарії використання: реєстрація, створення квізу, проходження тесту, перегляд результатів.

У розділі також розкрито питання забезпечення безпеки, зокрема автентифікацію за допомогою JWT, розмежування доступу за ролями та валідацію даних. Наприкінці розділу описано процес тестування: юніт- та інтеграційні тести, перевірка ендпоінтів API, запуск у контейнеризованому середовищі (Docker).

Загалом, реалізована система повністю відповідає поставленим завданням:

- забезпечує просту та безпечну взаємодію користувачів з квізами,
- дозволяє компаніям управляти навчанням та оцінюванням персоналу,
- легко розгортається в будь-якому середовищі,
- придатна до масштабування та подальшого розвитку.

Розроблена інформаційна система MiniBizManager є повноцінним базовим продуктом (MVP), який уже на цьому етапі може використовуватися в реальному корпоративному середовищі. Водночас існує низка напрямів, у яких система може бути вдосконалена та розширена відповідно до потреб бізнесу та користувачів.

Наразі система підтримує переважно запитання з одним або кількома варіантами відповіді. У майбутньому можлива реалізація:

- відкритих запитань із ручною перевіркою,
- завдань на відповідність,
- інтерактивних або мультимедійних форматів (зображення, відео).

Система може бути доповнена механізмом адаптивного тестування, де складність запитань автоматично підлаштовується під рівень користувача.

Запровадження більш гнучкої аналітики, зокрема:

- візуалізація прогресу користувача,
- звіти по департаментах компанії,
- динаміка результатів у часі,
- прогнозування на основі машинного навчання.

Передбачено можливість:

- інтеграції з корпоративними CRM/HRM-системами (наприклад, Bitrix24, Zoho, BambooHR),
- авторизації через сторонні сервіси (OAuth для Google/Microsoft),

- експорту/імпорту даних через API.

Розробка Progressive Web App або повноцінного мобільного застосунку (на Flutter, React Native) значно розширить охоплення користувачів.

Можливе впровадження:

- email- або Telegram-сповіщень про нові квізи, результати проходження,
- нагадувань про дедлайни.

Забезпечення підтримки багатомовності (i18n), що дозволить використовувати систему у міжнародних компаніях.

Завдяки відкритій архітектурі, модульній структурі та використанню сучасних технологій, система MiniBizManager має високий потенціал для масштабування та адаптації під потреби конкретного бізнесу. У майбутньому її можна перетворити на повноцінну платформу корпоративного навчання.

У результаті дослідження дипломної роботи створено функціональний MVP інформаційної системи, яка має практичну цінність для малого та середнього бізнесу, підвищує ефективність внутрішнього навчання та може бути інтегрована в корпоративну інфраструктуру.

Отже, розроблена інформаційна система «MiniBizManager» повністю відповідає заданим функціональним та технічним вимогам. Результати тестування засвідчили її ефективність, надійність та безпечність у роботі. Впроваджені рішення, зокрема використання сучасного технологічного стеку на основі Python, забезпечили високу швидкість розробки, масштабованість архітектури та зручність у супроводі.

Система продемонструвала стабільну роботу в умовах реального навантаження, а також готовність до подальшого вдосконалення, що відкриває перспективи її використання на практиці в малому та середньому бізнесі. Таким чином, поставлені завдання третього розділу дипломної роботи були успішно реалізовані.

ПЕРЕЛІК ПОСИЛАНЬ

1. Alembic. Alembic Documentation [Електронний ресурс]. – Режим доступу: https://alembic.sqlalchemy.org
2. Beazley D., Jones B. K. Python Cookbook. – 3rd ed. – Sebastopol : O'Reilly Media, 2021. – 706 p.
3. Celery Project. Celery Documentation [Електронний ресурс]. – Режим доступу: https://docs.celeryq.dev
4. FastAPI. FastAPI Documentation [Електронний ресурс]. – Режим доступу: https://fastapi.tiangolo.com
5. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Doctoral dissertation / R. T. Fielding. – University of California, Irvine, 2020. – [Електронний ресурс]. – Режим доступу: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
6. Grinberg M. Flask Web Development. – 2nd ed. – Sebastopol : O'Reilly Media, 2020. – 258 p.
7. Heller D. Professional API Design with FastAPI. – Independently Published, 2023. – 312 p.
8. HTTPX. HTTPX Documentation [Електронний ресурс]. – Режим доступу: https://www.python-httpx.org
9. Internet Engineering Task Force (IETF). RFC 7519: JSON Web Token (JWT) [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7519
10. ISO/IEC 25010:2021 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. – [Electronic resource]. – Available at: https://www.iso.org/standard/35733.html
11. Kurniawan B. Modern Python Web Development with FastAPI and SQLAlchemy. – Brainy Software, 2021. – 324 p.

12. Mozilla Developer Network. Cross-Origin Resource Sharing (CORS) [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
13. Newman S. Building Microservices. – 2nd ed. – Sebastopol : O'Reilly Media, 2021. – 618 p.
14. PostgreSQL Global Development Group. PostgreSQL Documentation [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/
15. Pydantic. Pydantic Documentation [Електронний ресурс]. – Режим доступу: https://docs.pydantic.dev
16. Python Software Foundation. Python 3.12 Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3
17. Redis Ltd. Redis Documentation [Електронний ресурс]. – Режим доступу: https://redis.io/docs
18. Sommerville I. Software Engineering. – 10th ed. – Boston : Pearson Education, 2021. – 792 p.
19. SQLAlchemy. SQLAlchemy Documentation [Електронний ресурс]. – Режим доступу: https://docs.sqlalchemy.org
20. Мартін Р. Чиста архітектура. Мистецтво розробки програмного забезпечення / Пер. з англ. – Київ : Наш Формат, 2021. – 352 с.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інформаційних систем та технологій

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

«ІНФОРМАЦІЙНА СИСТЕМА ПІДПРИЄМСТВА з ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ PYTHON»

Виконав: студент групи ІСД-42
Владислав МЕЖІНСЬКИЙ
Керівник: к.т.н., доцент кафедри ІСТ
ТКАЛЕНКО Оксана Миколаївна

Мета дослідження - розробка інформаційної системи підприємства з використанням Python, яка дозволяє здійснювати облік компаній і користувачів, організовувати навчальні або мотиваційні квізи, забезпечувати збереження та аналіз результатів, а також реалізовувати функції безпеки та комунікації всередині організації.

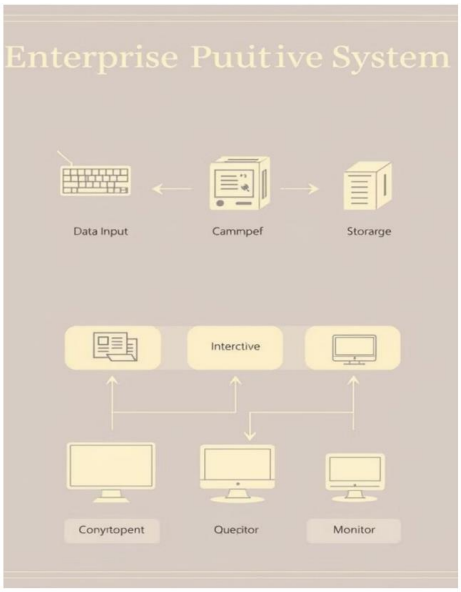
Об'єктом дослідження виступає процес цифровізації управлінських функцій підприємства.

Предметом дослідження є методи і засоби створення інформаційної системи на основі мови Python та суміжних технологій.

Для досягнення поставленої мети в роботі вирішуються такі завдання:

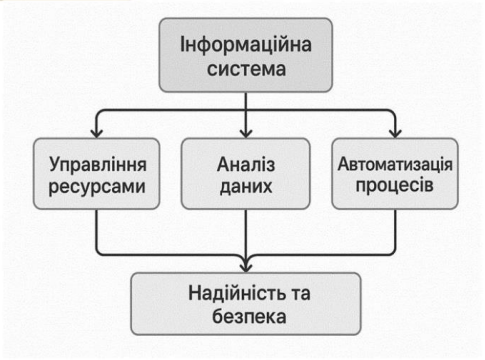
1. Проаналізувати теоретичні засади інформаційних систем підприємства.
2. Обґрунтувати вибір технологічного стеку та інструментів розробки.
3. Спроєктувати інформаційну систему з урахуванням архітектури та сценаріїв використання.
4. Реалізувати систему та провести її тестування.

Призначення та функції інформаційної системи підприємства



ПРИЗНАЧЕННЯ:

Автоматизація бізнес-процесів та підтримка прийняття рішень.



3

ОГЛЯД АНАЛОГІЧНИХ РІШЕНЬ: ПЕРЕВАГИ ТА НЕДОЛІКИ

ПЕРЕВАГИ:

- Інтеграція з іншими системами
- Автоматизація рутинних задач
- Покращений доступ до інформації

НЕДОЛІКИ:

- Висока вартість впровадження
- Складність підтримки
- Обмежена гнучкість

4

ПОСТАНОВКА ЗАДАЧІ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1. Основне завдання

Створити систему, що підтримує управління ресурсами підприємства.

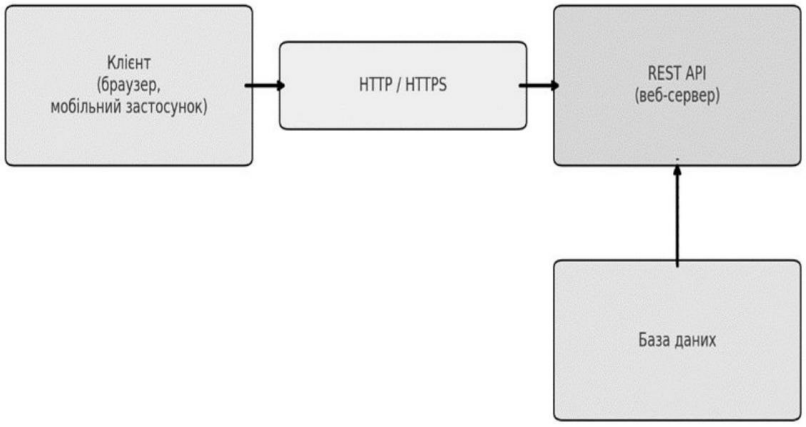
2. Ключові функції

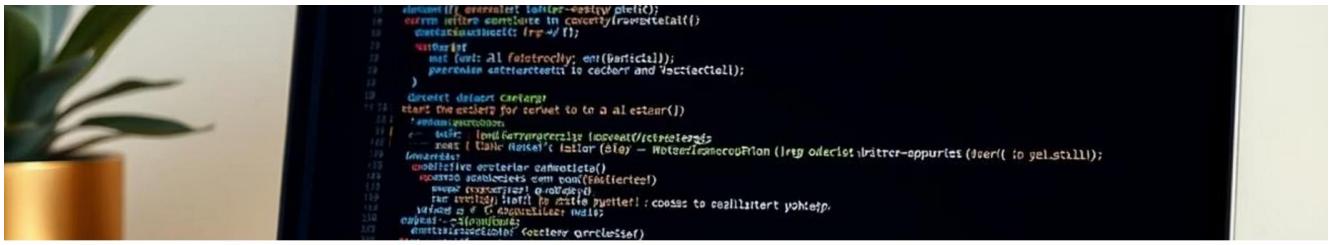
Обробка замовлень, управління персоналом, фінансовий контроль.

3. Ефективність

Забезпечення швидкого доступу до потрібної інформації.

Схема взаємодії клієнта з інформаційною системою через REST API





Вибір мови програмування Python

Гнучкість
Широке застосування в багатьох сферах IT.

Продуктивність
Висока швидкість розробки завдяки простоті.

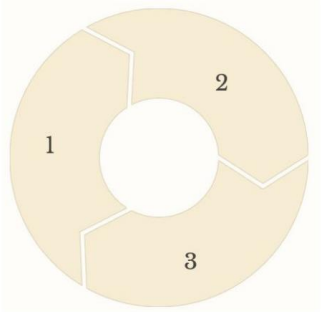
Велика спільнота

Багато бібліотек та фреймворків.

Огляд веб-технологій: сервіси та протоколи

HTTP/HTTPS

Протоколи передачі даних в мережі, забезпечують безпечне спілкування



Веб-сервіси

API для інтеграції різних систем і обміну інформацією.

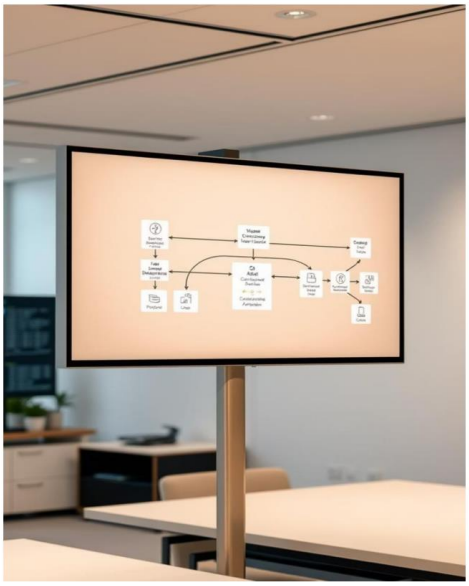
REST і SOAP

Популярні архітектурні стилі веб-сервісів.

Порівняльна характеристика PostgreSQL, Redis

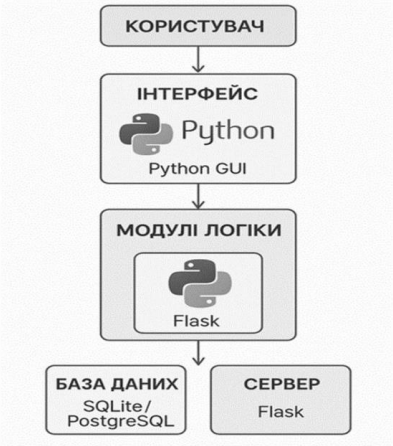
Характеристика	PostgreSQL	Redis
Тип сховища	Реляційна база даних	Ключ-значення (in-memory)
Структурованість	Висока, із зв'язками та обмеженнями	Низька, прості пари ключ-значення
Швидкодія	Висока	Дуже висока (мілісекунди)
Застосування	Постійне зберігання даних	Тимчасові дані, сесії, кеші
Підтримка транзакцій	Так	Обмежено
Витрати ресурсів	Більші	Менші, але працює в ОЗП

Основні технології, використані в системі MiniBizManager

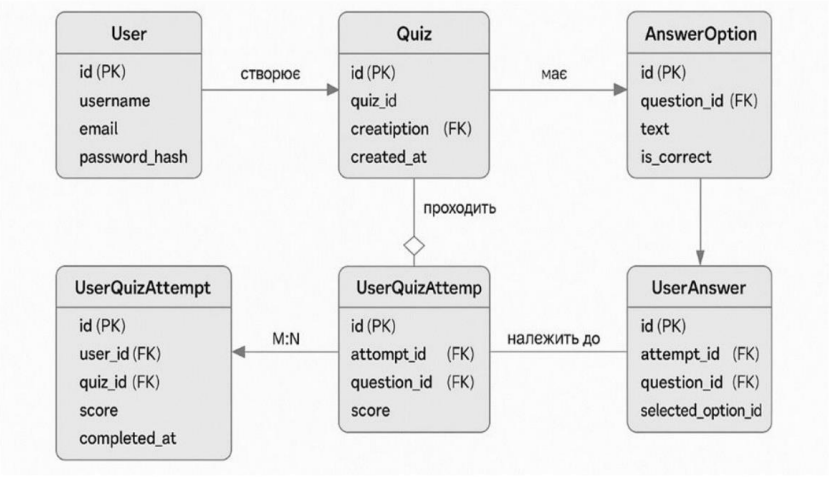


Компонент	Технологія	Призначення
Backend API	FastAPI	Фреймворк на Python для створення REST API
ORM	SQLAlchemy/Tortoise	Об’єктно-реляційне відображення
База даних	PostgreSQL	Реляційна СКБД
Авторизація	OAuth2/JWT	Безпечний механізм доступу користувачів
Схеми перевірки	Pydantic	Валідація вхідних/вихідних даних
Документація API	Swagger/OpenAPI	Автоматично генерується FastAPI
Frontend (за потреби)	React/Vue	Клієнтський інтерфейс
Контейнеризація	Docker	Для розгортання у середовищі розробки/продакшн
CI/CD (опційно)	GitHub Actions	Автоматизація перевірок та деплою

Структурна схема інформаційної системи підприємства на Python

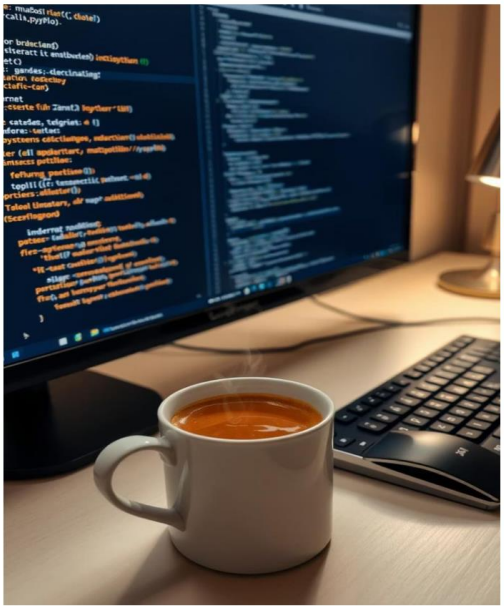


ER-діаграма для інформаційної системи Minibizmanager із функціоналом квізів

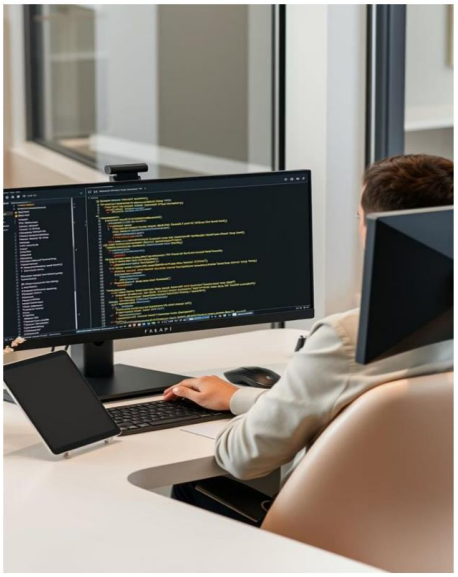


Бібліотеки у проєкті MINIBIZMANAGER

Бібліотека	Призначення
pydantic	Валідація та серіалізація даних
sqlalchemy	Об’ємна ORM для роботи з базами даних
alembic	Міграції бази даних
httpx	Асинхронні HTTP-запити



Архітектура та реалізація MINIBIZMANAGER



А Р Х І Т Е К Т У Р А:

Модульна побудова для легкого масштабування.

Б Е З П Е К А:

Сучасні методи шифрування та автентифікації.

Т Е С Т У В А Н Н Я:

Автоматизовані тести для надійності системи.

ВИСНОВКИ

У межах виконання дипломної роботи було розроблено інформаційну систему для підтримки бізнес-процесів малого підприємства з урахуванням сучасних вимог до функціональності, зручності використання та технічної надійності.

Проаналізовано роль ІС у сучасному бізнес-середовищі, сформульовано основні функції, яким повинна відповідати така система: автоматизація обліку, управління ресурсами, генерація звітності, інтеграція з користувачами. На основі аналізу існуючих аналогів (наприклад, 1С, Zoho, Odoo) виявлено їхні переваги та недоліки. Для створення бекенду використано FastAPI, що дозволяє швидко реалізовувати REST API з високою продуктивністю. В якості бази даних застосовано PostgreSQL для надійного збереження інформації, а Redis — для кешування та прискорення доступу до часто вживаних даних. Також активно застосовувалися сучасні бібліотеки — Pydantic для валідації даних, SQLAlchemy та Alembic для ORM і міграцій, HTTPX для асинхронної роботи з зовнішніми API.

Архітектура системи побудована за принципами клієнт-серверної взаємодії із чітким поділом логіки: бекенд на FastAPI, фронтенд (за потреби) може інтегруватися через REST API. Система має модульну структуру, що включає управління користувачами, клієнтами, товарами, послугами, звітами тощо. База даних спроєктована з урахуванням нормалізації, зв'язків між сутностями та можливості розширення. Реалізовано механізми реєстрації, автентифікації (JWT), ролей користувачів, що дозволяє гнучко керувати доступом. Здійснено тестування функціональності системи на основі реальних сценаріїв використання.
Загалом, у ході роботи було не лише реалізовано поставлену мету — створення власної ІС для малого підприємства — а й продемонстровано комплексний підхід до розв'язання інженерних, технологічних та аналітичних задач, що підтверджує нашу готовність до практичної діяльності в ІТ-сфері.