

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-системи управління шкільним
розкладом для викладачів, учнів та адміністратора»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

(підпис) Богдана КОВРИГА

Виконав: здобувачка вищої освіти групи ПД-41

Богдана КОВРИГА

Керівник: _____
Ірина ЗАМРІЙ
д-р техн. наук, проф.

Рецензент: _____

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ

« ____ » _____ 2026 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Ковризі Богдані Олександрівні

1. Тема кваліфікаційної роботи: «Розробка web-системи управління шкільним розкладом для викладачів, учнів та адміністратора»

керівник кваліфікаційної роботи д-р техн. наук, професор Ірина ЗАМРІЙ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості щодо організації навчального процесу та формування шкільного розкладу, аналіз існуючих вебсистем управління розкладом, методи проєктування та розробки web-застосунків, технічна документація фреймворків і бібліотек для розробки клієнтської та серверної частини застосунку, технології керування базами даних, засоби забезпечення автентифікації та авторизації користувачів.

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

1. Аналіз предметної області та існуючих рішень.

2. Аналіз засобів реалізації та структури web-системи управління шкільним розкладом.

3. Програмна реалізація web-системи управління шкільним розкладом.
4. Демонстрація результатів та тестування.
5. Перелік графічного матеріалу: *презентація*
 1. Аналіз аналогів.
 2. Вимоги до застосунку.
 3. Програмні засоби реалізації.
 4. Діаграма архітектури.
 5. Діаграма класів.
 6. Модель «сутність-зв'язок».
 7. Діаграма послідовності для учня класу/викладача.
 8. Діаграма послідовності для адміністратора.
 9. Діаграма варіантів використання.
 10. Діаграма діяльності.
 11. Екранні форми роботи застосунку.
 12. Апробація результатів дослідження
6. Дата видачі завдання «23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	23.02-28.02.2026	
2	Аналіз та дослідження існуючих аналогів	01.03-07.03.2026	
3	Формування вимог до вебсистеми управління шкільним розкладом	08.03-14.03.2026	
4	Проектування архітектури системи та бази даних	15.03-21.03.2026	
5	Розробка серверної частини вебсистеми	22.03-04.04.2026	
6	Розробка клієнтської частини, інтерфейсу користувача, системи авторизації та перевірки конфліктів розкладу	05.04-18.04.2026	
7	Тестування вебсистеми та виправлення помилок	19.04-30.04.2026	
8	Розробка демонстраційних матеріалів	01.05-10.05.2026	
9	Попередній захист роботи	11.05-22.05.2026	

Здобувачка вищої освіти

_____ (підпис)

Богдана КОВРИГА

Керівник
кваліфікаційної роботи

_____ (підпис)

Ірина ЗАМРІЙ

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 73 стор., 4 табл., 26 рис., 20 джерел.

Мета роботи – автоматизація процесу створення та управління шкільним розкладом із забезпеченням зручного доступу до інформації для різних категорій користувачів.

Об'єкт дослідження – процес формування та управління шкільним розкладом у навчальному закладі.

Предмет дослідження – методи та програмні засоби автоматизації управління розкладом на основі веб-технологій.

Короткий зміст роботи: У кваліфікаційній роботі розглянуто процес розробки web-системи управління шкільним розкладом для викладачів, учнів та адміністратора. Проведено аналіз предметної області та існуючих програмних рішень для організації навчального розкладу. Визначено функціональні та нефункціональні вимоги до системи, спроектовано архітектуру застосунку та структуру бази даних. У роботі реалізовано вебсистему, яка забезпечує можливість перегляду, створення, редагування та видалення записів розкладу, а також підтримує розмежування прав доступу для різних категорій користувачів. Реалізовано механізми автентифікації та авторизації користувачів, перевірку конфліктів у розкладі та обробку помилок введення даних. Проведено функціональне тестування системи та перевірку роботи основних програмних модулів.

Сферою використання застосунку є автоматизація процесу управління шкільним розкладом у закладах освіти.

КЛЮЧОВІ СЛОВА: ВЕБСИСТЕМА, SPRING BOOT, SPRING SECURITY, THYMELEAF, MYSQL, АВТЕНТИФІКАЦІЯ, БАЗА ДАНИХ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	
ВСТУП.....	10
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ.....	12
1.1 Опис предметної області та постановка задачі.....	12
1.2 Аналіз задачі та потреб користувачів.....	13
1.3 Аналіз існуючих підходів до вирішення задачі.....	14
1.4 Огляд існуючих програмних продуктів для управління розкладом.....	16
1.4.1 Microsoft Excel і Google Sheets.....	16
1.4.2 Система WebUntis.....	18
1.4.3 Система aSc Timetables.....	19
1.4.4 Система Google Classroom.....	21
1.4.5 Система TimetableMaster.....	22
1.4.6 Порівняльний аналіз.....	24
1.5 Визначення вимог до застосунку.....	26
2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ ТА СТРУКТУРИ ВЕБСИСТЕМИ УПРАВЛІННЯ ШКІЛЬНИМ РОЗКЛАДОМ.....	28
2.1 Середовище розробки та мова програмування.....	28
2.2 Фреймворки та серверна основа.....	29
2.3 Аутентифікація та авторизація.....	30
2.4 Робота з базами даних.....	31
2.5 Робота з моделями даних.....	32
2.6 Контейнеризація і розгортання.....	33
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБСИСТЕМИ УПРАВЛІННЯ ШКІЛЬНИМ РОЗКЛАДОМ.....	34
3.1 Загальна архітектура системи.....	34

3.2	Сценарії взаємодії користувачів із системою.....	36
3.2.1	Отримання розкладу (учень, викладач).....	38
3.2.2	Зміна розкладу (адміністратор).....	39
3.3	Проектування структури даних.....	40
3.4	Реалізація серверної частини.....	49
3.4.1	Реалізація доступу до даних.....	49
3.4.2	Реалізація бізнес-логіки.....	50
3.4.3	Реалізація контролерів.....	50
3.5	Реалізація системи безпеки.....	51
3.6	Реалізація інтерфейсу користувача.....	52
3.7	Перевірка даних і обробка помилок.....	53
4	ДЕМОНСТРАЦІЯ РОБОТИ СИСТЕМИ ТА ТЕСТУВАННЯ.....	54
4.1	Демонстрація роботи системи.....	54
4.1.1	Сторінка авторизації.....	55
4.1.2	Головна сторінка системи.....	55
4.1.3	Перегляд розкладу для класів.....	57
4.1.4	Перегляд розкладу викладачів.....	58
4.1.5	Адміністративна панель.....	60
4.1.6	Додавання нового запису розкладу.....	61
4.1.7	Перевірка конфліктів у розкладі.....	62
4.1.8	Редагування та видалення записів.....	63
4.2	Тестування авторизації та безпеки.....	66
4.3	Тестування функцій керування розкладом.....	68
4.4	Тестування перевірки даних та обробки помилок.....	69
	ВИСНОВКИ.....	71
	ПЕРЕЛІК ПОСИЛАНЬ.....	73
	ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ.....	75
	ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ.....	86

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

CRUD – Create, Read, Update, Delete

CSS – Cascading Style Sheets

DBMS – Database Management System

ER – Entity Relationship

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

JPA – Java Persistence API

JSON – JavaScript Object Notation

MVC – Model-View-Controller

ORM – Object-Relational Mapping

REST – Representational State Transfer

SQL – Structured Query Language

UI – User Interface

URL – Uniform Resource Locator

UX – User Experience

ВСТУП

Актуальність: у сучасних закладах освіти процес формування шкільного розкладу залишається складною та трудомісткою задачею, яка часто виконується вручну або з використанням загальних інструментів, таких як електронні таблиці. Це призводить до виникнення помилок, конфліктів у розкладі, неефективного використання ресурсів та ускладнює внесення змін. В умовах цифровізації освіти зростає потреба у впровадженні спеціалізованих інформаційних систем, які дозволяють автоматизувати процес управління розкладом, забезпечити його узгодженість, зручний доступ для користувачів та підвищити ефективність організації навчального процесу. Тема є актуальною з огляду на необхідність підвищення якості освітнього процесу та оптимізації роботи навчальних закладів.

Об'єктом дослідження є процес формування та управління шкільним розкладом у навчальному закладі.

Предметом дослідження є методи та програмні засоби автоматизації управління розкладом на основі веб-технологій.

Метою роботи є автоматизація процесу створення та управління шкільним розкладом із забезпеченням зручного доступу до інформації для різних категорій користувачів.

Методи дослідження: на першому етапі було проведено аналіз предметної області, зокрема особливостей процесу складання розкладу та основних проблем, що виникають при його формуванні. Також було досліджено існуючі підходи та програмні рішення для управління розкладом з метою визначення їх переваг і недоліків та формування вимог до власної системи.

На наступному етапі було обрано технологічний стек для реалізації застосунку, що забезпечує ефективну розробку та подальшу підтримку системи. Для створення серверної частини використано мову програмування Java та фреймворк Spring Boot, який спрощує розробку завдяки автоматичній конфігурації

та вбудованому серверу. Для зберігання даних обрано реляційну СКБД MySQL. Керування залежностями та збіркою проєкту здійснюється за допомогою Apache Maven, а для розгортання середовища використано Docker Compose [10]. Архітектура системи базується на шаблоні Model-View-Controller, що передбачає розподіл на модель, контролер і представлення. Для реалізації інтерфейсу використано шаблонізатор Thymeleaf, який дозволяє створювати динамічні HTML-сторінки з прив'язкою даних із серверної частини та підтримує умовне відображення елементів залежно від ролі користувача.

Наукова новизна роботи полягає у покращенні процесу створення та управління шкільним розкладом через розробку вебсистеми, яка поєднує простоту використання з достатнім рівнем функціональності, забезпечує розмежування доступу на основі ролей користувачів, а також реалізує механізми перевірки узгодженості даних і запобігання конфліктам у розкладі.

Практична значущість результатів полягає у можливості використання розробленого веб-застосунку в навчальних закладах для автоматизації процесу формування розкладу, зменшення кількості помилок, спрощення внесення змін та забезпечення швидкого доступу до актуальної інформації для учнів, викладачів та адміністрації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Опис предметної області та постановка задачі

На сьогоднішній день у розвитку інформаційних технологій все більшого значення набуває автоматизація процесів управління в різних сферах діяльності, зокрема в освітній галузі. Складання та оновлення шкільного розкладу занять є одним із ключових елементів організації навчального процесу в закладах середньої освіти.

Розклад занять встановлює порядок проведення занять, розподілу навчального навантаження між іншими викладачами, використання навчальних аудиторій, а також тривалість навчальних занять. Від того, наскільки він якісно і раціонально побудований, залежить і ефективність навчання, і навантаження на учнів та викладачів, навчального закладу в цілому.

Принцип складання розкладу - це складна комбінаторна задача, яка характеризується величезним числом параметрів і обмежень [11, 12, 16, 17, 20]. Основні фактори, які слід врахувати при складанні розкладу:

- кількість навчальних класів;
- перелік навчальних дисциплін;
- кадровий склад викладачів та їх кваліфікація;
- доступність аудиторій;
- часові обмеження (уроки, перерви, зміни);
- індивідуальні побажання або обмеження викладачів.

Однією з найбільш поширених проблем є виникнення конфліктів у розкладі [11, 16, 20]. Наприклад, один викладач не може одночасно проводити заняття у двох різних класах, а один клас не може бути присутнім на кількох уроках одночасно. Окрім цього, існують обмеження за добовим навантаженням, рівномірного міжпредметного розподілу складних предметів, а також

адміністративні рішення з організації навчального процесу.

У багатьох школах процес формування розкладу досі здійснюється вручну або з використанням простих інструментів, таких як електронні таблиці [14, 15].

Такий підхід має низку суттєвих недоліків:

- висока трудомісткість процесу;
- значна ймовірність виникнення помилок;
- складність внесення змін;
- відсутність централізованого доступу до актуальної інформації;
- обмежені можливості масштабування.

Треба розробити такі інформаційні системи, які б уможливили автоматизацію процесу керування розкладом, мінімізацію кількості помилок і забезпечення зручного доступу до інформації для різних груп користувачів [14, 15, 20].

Отже, завдання даної бакалаврської роботи розробити веб-застосунок, який би дозволяв створювати, редагувати, зберігати та переглядати розклад для школи з урахуванням ролей користувачів (адміністратор, вчитель, учень) і підтримував цілісність і узгодженість даних.

1.2 Аналіз задачі та потреб користувачів

Розробка інформаційної системи управління шкільним розкладом передбачає виконання етапу аналізу вимог на основі особливостей предметної області і потреб кінцевих користувачів. Однозначне розуміння функціоналу системи, а також які умови її використання повинні бути забезпечені, є необхідними умовами для розвитку відповідного програмного продукту. Формування вимог до системи базується на аналізі ролей користувачів, їх взаємодії з системою та підсумків, які очікують отримати від використання системи.

У досліджуваній предметній області, як правило, можливо виділити три ролі користувачів, з відмінними цілями і рівнем доступу, так само як і різний тип взаємодії з системою. Адміністратор є центральною фігурою в процесі розробки і

підтримки актуального розкладу. Саме цей користувач піклується про побудову каркаса розкладу, його наповнення і за потреби редагування. Адміністратор може повноцінно працювати з інформацією від початку до кінця, тобто як редагувати, так і видаляти записи. Важливо підкреслити, що він мав бути адміністратором менших залежностей, таких як класи, викладачі і предмети. Крім того, адміністратор несе відповідальність за цілісність даних, а це означає, що в розкладі не повинно бути логічних протиріч, наприклад, збіжності занять, або неправильного використання ресурсів.

Ще однією важливою категорією користувачів є вчителі, для яких система є інструментом отримання інформації про навчальний процес. Взаємодія цієї групи з системою має переважно інформаційний характер і не впливає на структуру розкладу. Основна вимога до викладача - мати швидкий доступ до свого розкладу, що дозволяє ефективно планувати свій робочий час та організовувати навчальний процес. Важливе значення має своєчасна актуалізація інформації, оскільки зміни в розкладі повинні оперативно відображатися в системі. Також важливим є забезпечення доступу до системи незалежно від місцезнаходження користувача, що реалізується за допомогою веб-технологій.

Третьою категорією користувачів є учні, які отримують кінцеву інформацію про розклад занять. Для цієї категорії користувачів система повинна надавати максимально зручний і швидкий доступ до потрібної інформації без зайвих ускладнень. Учні хочуть мати можливість переглядати розклад свого класу у зручному вигляді, щоб орієнтуватися в навчальному процесі. Важливою вимогою є й актуальність інформації, оскільки своєчасне отримання змін у розкладі допоможе уникнути непорозумінь і сприятиме більшій організованості навчального процесу. Що стосується мобільних пристроїв, то можливість заходити в систему з мобільних пристроїв також впливає на зручність використання [14, 15].

1.3 Аналіз існуючих підходів до вирішення задачі

У сучасний період розвитку інформаційних технологій в освіті існує кілька

підходів до складання шкільного розкладу занять, які відрізняються за рівнем автоматизації, складністю та ефективністю. Кожен із них має низку переваг, а також значні обмеження, що впливають на доцільність їхнього застосування в конкретних умовах навчального закладу.

Більш традиційним методом складання розкладу занять є ручний метод [11], при якому для створення розкладу не використовується жодне спеціалізоване програмне забезпечення. У цьому випадку вся відповідальність за створення розкладу покладається на адміністратора. Такий підхід вважається простим, оскільки його реалізація не передбачає використання спеціальних пристроїв чи знань з ІТ. Крім того, він дає змогу повністю контролювати процес формування розкладу, що дозволяє швидко реагувати на зміни у його форматі. Однак такий підхід до ручного складання розкладу має низьку ефективність, оскільки займає багато часу та зусиль. Велика кількість можливих помилок, пов'язаних із людським фактором, а також складність внесення змін до необхідного розкладу істотно ускладнює його практичне застосування. Відсутність будь-яких елементів автоматизації ускладнює застосування цього методу для сучасних навчальних закладів, особливо для тих, що мають велику кількість класів або викладачів.

Сучасним варіантом є використання електронних таблиць (табличний редактор типу Excel), де інформацію можна частково структурувати і це полегшує процес створення розкладу [14]. У цьому випадку користувач формує таблицю, в якій відображаються дні тижня, часові проміжки, а також дізнається про те, які заняття у нього на кожен день, що дозволяє наочно представити дані. Великим плюсом даного на підходу є те, що він доступний і достатньо простий у використанні, адже такі інструменти існують у великій кількості і не потрібно ніяких спеціальних знань на базовому рівні. У той же час електронні таблиці дають можливість оперативно змінювати дані, редагувати структуру інформації і виконувати нескладні розрахунки за допомогою вбудованих формул. Але у цього способу є і значні обмеження. Зокрема, відсутність автоматизованої перевірки конфліктів у розкладі, що істотно підвищує ймовірність помилок. При збільшенні об'єму інформації, керування таблицями ускладнюється, і стає менш зручним, що

погано впливає на продуктивність роботи.

Додатково слід відзначити і неабиякі складності для реалізації організації спільного доступу з боку користувачів, і відсутність єдиної структури зберігання даних, яка б уніфікувала обробку та аналіз даних.

Найкращим варіантом є застосування спеціалізованих програмних засобів, орієнтованих безпосередньо на автоматизацію процесу розробки навчального розкладу [12, 16, 17, 20]. Такі програми дають змогу враховувати значну кількість обмежень і зв'язків, які накладаються на процес планування, і значно мінімізують обсяг рутинної роботи. Вони підтримують формування розкладу на основі заданих параметрів, надають можливість диференціювати права доступу між різними категоріями користувачів, а також забезпечують централізоване збереження даних. Це враховує підвищення точності, узгодженості і своєчасності інформації, а також доступність для неї. Насправді у таких систем теж є недоліки. Зокрема, впровадження цих систем може бути пов'язане з додатковими фінансовими витратами. Крім того, налаштування та використання таких програм може спричинити значні труднощі, особливо для непідготовленого користувача. Недостатня гнучкість під конкретні запити окремої установи при її адаптації та залежність від розробника програмного забезпечення також можуть бути негативними факторами під час її використання.

Отже, аналіз існуючих методів формування шкільного розкладу показує, що кожен з них має як переваги, так і недоліки. Це пояснює необхідність розробки сучасної програмної реалізації рішення, яка з одного боку, за своїм призначенням відповідатиме вимогам зручності в користуванні, комплексного спостереження та контролю, а з іншого боку достатнього ступеня гнучкості для адаптації вимогам конкретної освітньої установи.

1.4 Огляд існуючих програмних продуктів для управління розкладом

1.4.1 Microsoft Excel і Google Sheets

Електронні таблиці, такі як Microsoft Excel і Google Sheets, показані на

рисунку 1.1, є популярними інструментами для створення розкладу занять. Такий метод має формування структурної таблиці з днями тижня, часовим проміжком і заняттями, що відповідає за візуалізацію даних і стандартизацію інформації.

Основним плюсом при роботі з електронними таблицями є те, що на них не вимагаються ніякі витрати і вони досить прості у використанні. Вони не вимагають ніякого спеціального програмного забезпечення або особливих налаштувань і підходять для широкої аудиторії. Електронні таблиці забезпечують легкість і швидкість зміни даних, дають змогу змінювати структуру таблиць, виконувати прості обчислення, скористатись вбудованими засобами для дуже простих результатів. Через це такі інструменти часто застосовуються у невеликих установах або на першому рівні планування.

Але разом з тим виникає і ряд недоліків для використання електронних таблиць. Зокрема, відсутність автоматичної перевірки конфліктів розкладу може призводити до виникнення помилок під час складання розкладу занять [11, 16]. При зростанні розміру даних працювати з таблицями стає не так легко, що ускладнює їхню використання. Слід також відзначити, що є певні обмеження в можливості фрагментувати права доступу, захист інформації не є надійним, і немає централізованого менеджменту над даними.

З огляду на це електронні таблиці доцільно розглядати передусім як допоміжний інструмент, оскільки він не забезпечує потрібного рівня автоматизації чи функціональності для ефективного управління навчальним планом у контексті сучасного навчального процесу.

Training Schedule							start:▼ 08/31/2026
	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
WEEK 1	☐ Pending August 31	☐ Pending September 1	☒ Finished September 2	☒ Finished September 3 Introduction to low-fidelity wireframes and interactive prototyping using modern tools.	☐ Pending September 4	☒ Finished September 5 Understanding component-based design and how to build scalable UI libraries.	☒ Finished September 6 Explore user-centered problem solving and how to apply empathy in design processes.
WEEK 2	☒ Finished September 7 Learn how to conduct user interviews, surveys, and usability tests to inform design decisions.	☒ Finished September 8 Understanding color psychology, accessibility, and creating harmonious palettes for interfaces.	☒ Finished September 9 Designing interfaces that adapt beautifully across mobile, tablet, and desktop screens.	☐ Pending September 10	☐ Pending September 11 Real-time teamwork in Figma: components, auto-layout, and version control.	☐ Pending September 12 Tips on curating projects, writing case studies, and presenting your design thinking.	☐ Pending September 13 Real-time teamwork in Figma: components, auto-layout, and version control.
WEEK 3	☐ Pending September 14	☐ Pending September 15	☐ Pending September 16	☐ Pending September 17 Understand how to create cohesive branding across typography, color, and imagery.	☐ Pending September 18 Explore emerging trends, how to use them tastefully, and where to find creative sparks.	☐ Pending September 19 Visualize the end-to-end user experience and identify friction points in your product flow.	☐ Pending September 20
WEEK 4	☐ Pending September 21	☐ Pending September 22	☐ Pending September 23 Reflect: What did you learn? What sparked interest? What's your focus next week?	☐ Pending September 24	☐ Pending September 25	☐ Pending September 26	☐ Pending September 27
WEEK 5	☐ Pending September 28	☐ Pending September 29	☐ Pending September 30 Discover platforms like Webflow and Framer to bring your designs to life	☐ Pending October 1	☐ Pending October 2	☐ Pending October 3	☐ Pending October 4

Рис. 1.1 Інтерфейс роботи з розкладом у Microsoft Excel

1.4.2 Система WebUntis

Система WebUntis зображена на рисунку 1.2 входить у перелік найбільш популярних програмних продуктів для планування та управління навчальним часом закладів освіти. Це дає змогу різним категоріям користувачів, зокрема школярам, вчителям та адміністраторам, отримувати доступ до розкладу через вебінтерфейс, що сприяє оперативному інформуванню про стан освітнього процесу.

Однією з важливих переваг даної системи є підтримка автоматизованого складання розкладу з урахуванням заданих обмежень, що значно полегшує процес планування. Крім того, реалізовано механізм розпізнавання рівня доступу, що дозволяє активувати різний функціонал для користувачів з різними ролями. Оскільки застосовуються вебтехнології, система доступна з будь-якого пристрою, що підключений до Інтернету, що підвищує її зручність використання. Крім того, потрібно відзначити можливість інтегрування з іншими освітніми послугами, що

дозволяє збагатити функції платформи. Також система дає змогу переглядати розклад у реальному часі, тож можна слідкувати за змінами у ньому.

Водночас, WebUntis має і свої мінуси. Зокрема, користувачам без досвіду роботи з подібними програмними продуктами може бути складно налаштувати систему. Ціна використання цієї системи може бути висока для навчальних закладів. Часто інтерфейс системи сприймається як перевантажений через велику кількість доступних функціональних можливостей, що ускладнює адаптацію нових користувачів. Як зазначається, існують обмеження гнучкості в системі, що обмежує можливість доопрацювання системи під конкретні вимоги закладу освіти.

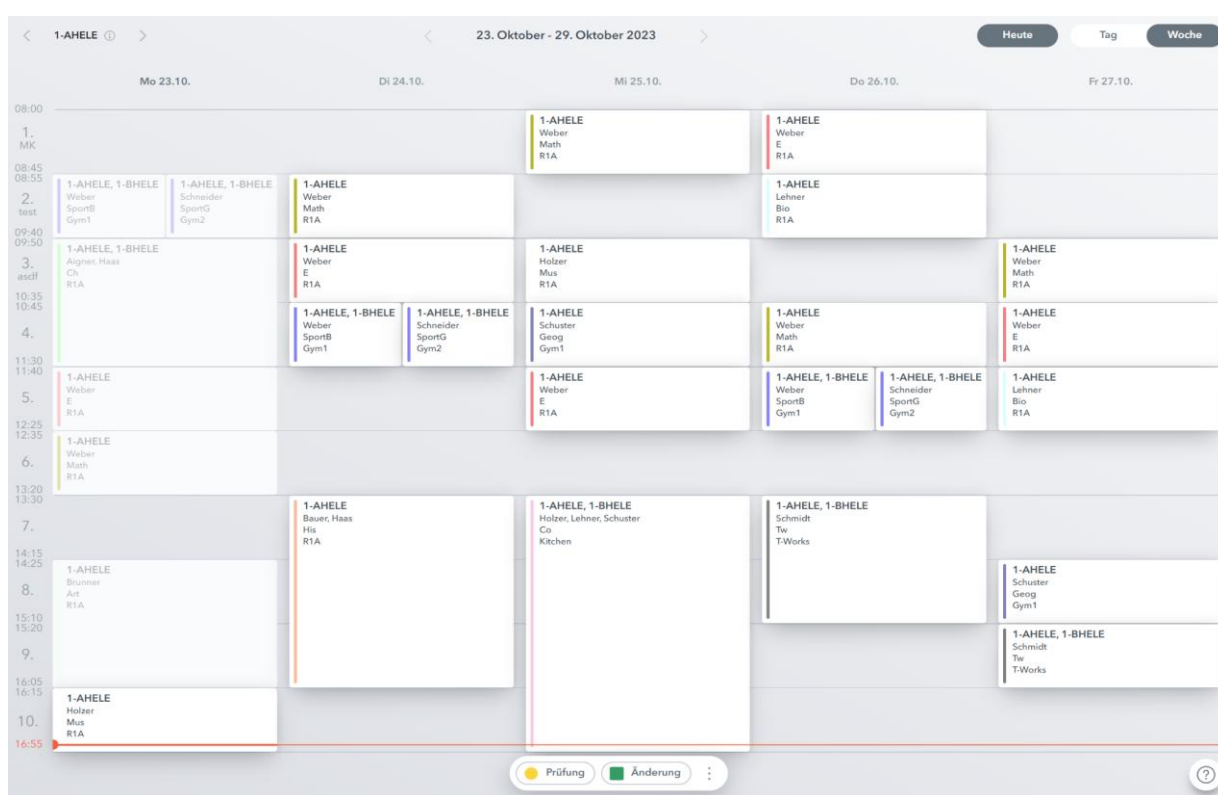


Рис. 1.2 Інтерфейс системи WebUntis

1.4.3 Система aSc Timetables

aSc Timetables, показаний на рисунку 1.3, є продуктом, що орієнтований на те, щоб максимально полегшити рішення задачі організації часу і складання розкладу. Програмне забезпечення спеціалізоване на шкільних й університетських розкладах, також дещо полегшує процес планування занять із застосуванням алгоритмічних засобів створення розкладу.

До основних переваг системи можна віднести можливість розрахунку розкладу з урахуванням різноманітних обмежень, таких як: зайнятість викладачів, встановлені часові інтервали тощо. Таким чином, платформа забезпечує високий рівень автоматизації навіть у великих за масштабом (з кількістю учнів) навчальних закладах. Ще однією перевагою є підтримка експорту готового розкладу у різні формати, що сприяє легкості подальшого використання, збереженню та поширенню даних.

Водночас використання aSc Timetables має певні обмеження. Зокрема, для нових користувачів система може бути складною у засвоєнні через специфічні особливості інтерфейсу та відносно високий поріг входження. Обмежені можливості веб-доступу ускладнюють роботу з системою поза локальним середовищем, а також часто виникає необхідність встановлення додаткового програмного забезпечення. Крім того, інтеграція з іншими інформаційними системами може бути ускладненою, що знижує ефективність використання системи у складі комплексних ІТ-рішень навчального закладу.

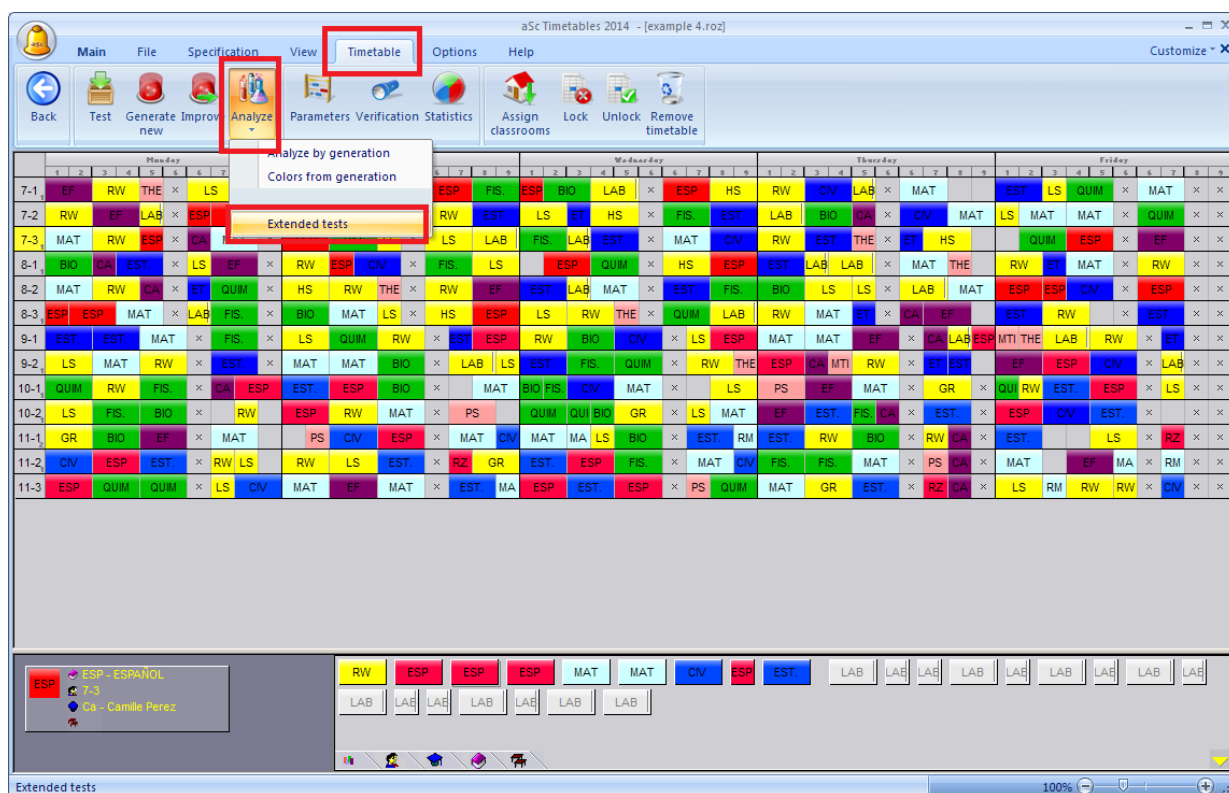


Рис. 1.3 Інтерфейс програми aSc Timetables

1.4.4 Система Google Classroom

Система Google Classroom, зображена на рисунку 1.4, не є спеціалізованим інструментом для створення навчального розкладу, однак широко використовується для організації освітнього процесу. Вона надає можливість створювати навчальні курси, розповсюджувати матеріали та забезпечувати взаємодію між викладачами і учнями.

До переваг платформи належить простота використання та інтуїтивно зрозумілий інтерфейс, що дозволяє швидко освоїти роботу з нею навіть користувачам без спеціальної підготовки. Важливим аспектом є інтеграція з іншими сервісами екосистеми Google, такими як документи, календар та хмарне сховище, що забезпечує зручне управління навчальними матеріалами. Доступ до системи можливий із будь-якого пристрою через веб-браузер, що підвищує її мобільність і гнучкість використання. Крім того, платформа є безкоштовною, що робить її доступною для широкого кола навчальних закладів.

Разом із тим, Google Classroom має обмеження у контексті управління розкладом. Зокрема, система не призначена для автоматичного формування розкладу та не підтримує складні механізми планування. Також вона не надає достатніх інструментів для детального управління часовими інтервалами, що знижує її ефективність при вирішенні задач, пов'язаних із організацією розкладу занять.



Рис. 1.4 Інтерфейс платформи Google Classroom

1.4.5 Система TimetableMaster

TimetableMaster, показаний на рисунку 1.5, є інноваційним програмним комплексом, який орієнтований на потреби навчальних закладів у сфері автоматизації процесу управління навчальним розкладом. До його функцій належить можливість створення розкладу занять, його редагування та перегляду, і все це реалізовано з урахуванням потреб різних категорій користувачів - адміністрації, викладачів та учнів. У цій ролі система стає інструментом централізованого управління даними, що сприяє покращенню якості інформації та/або полегшенню доступу до даних.

Відмінною рисою програмного продукту є те, що він використовує автоматизацію під час основних операцій формування розкладу. Це дозволить не тільки значно зменшити кількість ручної роботи, але й істотно зменшити потенційні помилки та підвищити ефективність планування. Для розробки інтерфейсу системи були визначені потреби користувачів, що дозволяє охарактеризувати простоту освоєння даної системи та її використання в роботі. Важливою особливістю є також підтримка ролей користувачів [13, 18, 19], що дає змогу обмежувати доступ до функціоналу залежно від прав та обов'язків

користувачів кожної категорії. Це сприяє підвищенню безпеки системи та запобіганню несанкціонованих змін.

Не менш важливим є веборієнтований дизайн, який забезпечує доступ до системи з будь-яких пристроїв без встановлення додаткового програмного забезпечення. Таким чином, користувачам будуть доступні найсвіжіші дані розкладу в будь-який час і в будь-якому місці за умови наявності інтернет-з'єднання. Таке рішення забезпечує більшу гнучкість системи та допомагає адаптуватися до змін у навчальному процесі.

Разом з тим система має свої обмеження, які слід брати до уваги при використанні. Зокрема, безкоштовна версія платформи має обмежені можливості, через що може знадобитися перехід на платну підписку для повного доступу до всіх інструментів. Це може спричинити додаткові фінансові витрати для освітніх закладів, особливо при обмеженому бюджеті. Крім того, існування потреби в постійному доступі до мережі інтернет може ускладнити використання системи в зоні нестабільного чи відсутнього мережевого сигналу, що є вкрай актуальним для певних територій чи навчальних закладів з малою технічною базою.

Також варто зауважити, що існуючі можливості кастомізації системи є доволі обмеженими. Це ускладнює або навіть унеможлиблює адаптацію програмного забезпечення до потреб конкретного навчального закладу. У деяких випадках установі може доводитися підлаштовувати існуючі процеси під можливості системи, а не навпаки, що зменшує її універсальність.

Крім вказаних аспектів, ця система інтегрується також і з іншим програмним забезпеченням для освіти, які використовуються у середовищі навчання. Недостатня підтримка інтерфейсів інтеграції ускладнює обмін інформацією з іншими платформами, що негативно позначається на можливості ефективно використовувати систему спільно з іншими цифровими технологіями.

Таким чином, TimetableMaster є досить пристосованим до організації та керування навчальним розкладом користувача інструментом, який поєднує між собою автоматизацію основних процедур та зручність використання. У той же час існуючі обмеження, зокрема в доступності повного функціоналу і залежності від

зовнішніх умов, викликають нагальну потребу в аналізі доцільності застосування його в конкретних умовах. Це додає, що можна виготовити і власні програмні рішення, які більш точно відповідатимуть потребам конкретного навчального закладу.

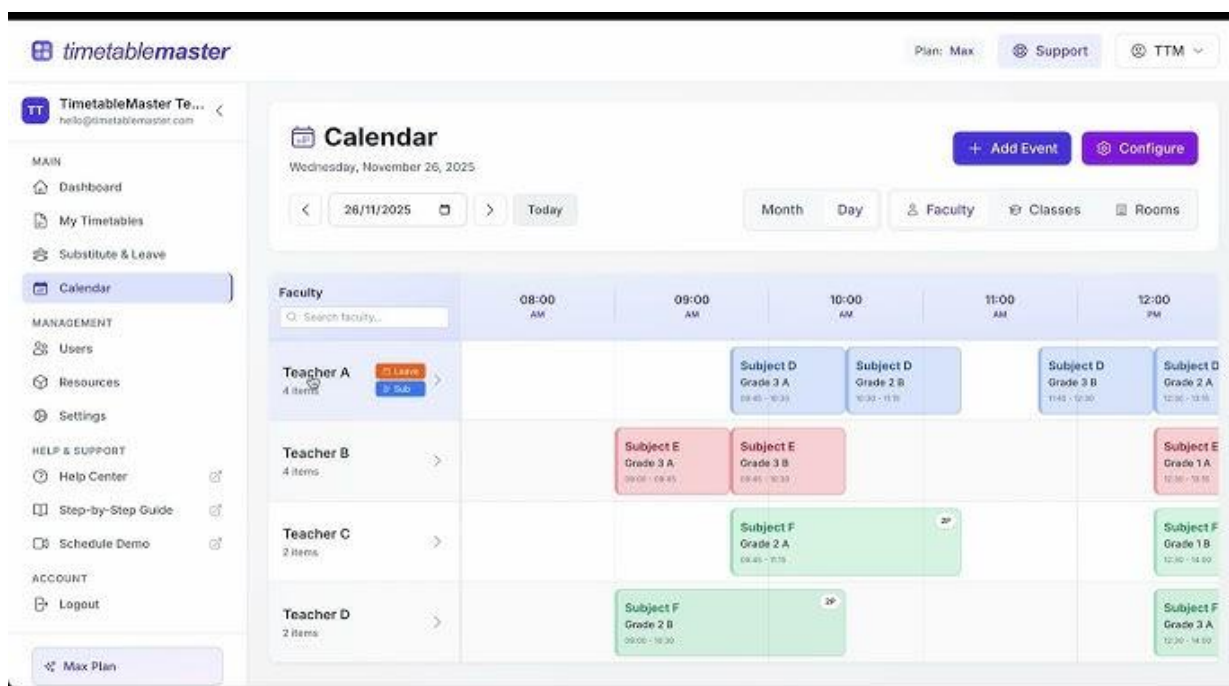


Рис. 1.5 Інтерфейс системи TimetableMaster

1.4.6 Порівняльний аналіз

Загальний аналіз вказує на наявність спільних обмежень серед розглянутих рішень. Більшість програм або не забезпечує достатнього рівня автоматизації процесу формування розкладу, або, навпаки, є надто складною у використанні та налаштуванні для кінцевих користувачів. Частина з них має обмежені можливості гнучкої адаптації під специфічні потреби конкретного навчального закладу, що ускладнює їх практичне застосування. Також спостерігається недостатній рівень інтеграції з іншими інформаційними системами та відсутність зручних механізмів розширення функціоналу. Лише окремі рішення поєднують автоматизацію, зручність використання та веб-доступ, проте навіть вони можуть мати обмеження у кастомізації або вимагати додаткових ресурсів для впровадження і підтримки.

Результати аналізу аналогів наведено у таблиці 1.1.

Таблиця 1.1

Порівняння існуючих програм аналогів

Показник	WebU ntis	aSc Timetables	Google Classroom	Timetable Master	Schedify (розроблюва на система)
Веб-доступ	+	-	+	+	+
Ролі користувачів	+	+	+	+	+
Автоматизація розкладу	+	+	-	-	+
Управління розкладом	+	+	-	+	+
Редагування розкладу в реальному часі	-	+	-	-	+
Система авторизації та автентифікації	+	-	+	-	+

Виявилось, що класичні рішення мають свої слабкі та сильні сторони, які загалом впливають на їхню застосовність в освітніх закладах. З одного боку, такі інструменти дають змогу частково або повністю автоматизувати процес планування, а також забезпечують збереження та відображення інформації, завдяки чому покращується організованість навчального процесу. З іншого - існуючі

рішення не завжди відповідають реальним потребам користувачів або містять обмеження, які перешкоджають їх застосуванню на практиці.

Зокрема, універсальні рішення, такі як електронні таблиці, хоч і доступні та прості у використанні, не сприяють достатній автоматизації процесу складання розкладу. Вони не містять механізмів перевірки конфліктів, не підтримують складну логіку планування та передбачають багато ручної роботи. Це, у свою чергу, призводить до збільшення кількості помилок і ускладнює внесення змін, особливо при роботі з великими масивами даних.

Спеціалізовані програми, призначені для управління розкладом, мають набагато ширший функціонал, зокрема включають в себе автоматичне генерування розкладу. Але ці рішення часто відзначаються складністю у налаштуванні та використанні, що може передбачати необхідність навчання користувачів або залучення технічних фахівців. Таким чином, результати проведеного аналізу вказують на наявність дисбалансу між функціональними можливостями наявних рішень і реальними потребами користувачів. Це зумовлює необхідність створення власної вебсистеми, яка б поєднувала простоту використання, гнучкість у налаштуванні та достатній рівень функціональності.

1.5 Визначення вимог до застосунку

На основі проведеного аналізу предметної області, існуючих підходів та програмних рішень, а також з урахуванням потреб основних категорій користувачів, було сформовано перелік вимог до розроблюваного застосунку. Ці вимоги визначають ключові функціональні можливості системи.

Функціональні вимоги:

1. Система повинна забезпечувати реєстрацію та авторизацію користувачів із розмежуванням прав доступу відповідно до ролей (учень, викладач, адміністратор).
2. Користувачі повинні мати можливість переглядати розклад занять за різними параметрами (клас, викладач) у зручному табличному вигляді.

3. Адміністратор повинен мати можливість створювати, редагувати та видаляти розклад, а також здійснювати перевірку конфліктів (перетин викладачів або занять).

4. Система повинна забезпечувати управління даними, зокрема класами, предметами, викладачами.

5. Система повинна забезпечувати CRUD-операції для класів, викладачів, предметів.

6. Дані повинні зберігатися у базі даних.

Нефункціональні вимоги:

1. Час відгуку системи під час виконання основних операцій не повинен перевищувати 2 секунд за стандартного навантаження.

2. Система повинна забезпечувати одночасну роботу не менше 50 користувачів без суттєвого зниження продуктивності.

3. У разі виникнення помилок система повинна відображати повідомлення про помилку без втрати введених користувачем даних.

4. Паролі користувачів повинні зберігатися у зашифрованому вигляді з використанням сучасних алгоритмів хешування.

5. Доступ до функціональних можливостей системи повинен обмежуватися відповідно до ролей користувачів.

6. Система повинна забезпечувати захист від несанкціонованого доступу шляхом використання механізмів автентифікації та авторизації.

7. Вебзастосунок повинен коректно працювати у сучасних веббраузерах, зокрема Google Chrome, Mozilla Firefox та Microsoft Edge.

Особливу увагу слід приділити зручності використання системи, оскільки кінцевими користувачами можуть бути особи без технічної підготовки.

2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ ТА СТРУКТУРИ ВЕБСИСТЕМИ УПРАВЛІННЯ ШКІЛЬНИМ РОЗКЛАДОМ

У даному розділі розглядаються основні засоби реалізації вебсистеми управління шкільним розкладом, обґрунтовується вибір технологій. Особливу увагу приділено архітектурним принципам, які забезпечують ефективну взаємодію між компонентами системи, а також дозволяють досягти необхідного рівня продуктивності, надійності та зручності використання.

2.1 Середовище розробки та мова програмування

Вибрати, як писати код, дуже важливо, бо це впливає на зручність і швидкість створення програми, а також на те, як її далі можна покращувати і підтримувати. Для цієї системи було обрано мову Java [1, 8]. Це одна з найпоширеніших мов у світі для створення сайтів і різних застосунків. Вона добре підходить, бо надійна, дозволяє писати зрозумілий код, а також підтримує роботу з об'єктами. Ще одна важлива особливість у Java — це те, що можна запускати програми чи то на Windows, чи то на Linux, чи на інших платформах без змін у коді. Так стається завдяки віртуальній машині Java, яка дає змогу просто брати програму і вмикати її на різних системах. Через все це Java часто вибирають, коли треба створити надійну, зрозумілу та зручну систему для багатьох людей.

Java має багато різних бібліотек і рамок [8, 9]. Це робить простішою роботу для тих, хто пише код і створює програму. З цим можна легше робити такі справи, як обробляти запит, працювати з базою даних, додавати захист, або робити так, щоб частини системи могли між собою говорити. Якщо використовувати строгий тип і чітку структуру, то програмісти менше роблять помилок, а програми стають кращими і більш надійними. Це дуже важливо тоді, коли створюють складну інформаційну систему.

Під час створення вебсистеми робота була у середовищі розробки IntelliJ

IDEA [6]. Це середовище швидке і зручне, там легко писати код. Головні функції середовища — підказки для коду, перевірка коду, відлагодження, робота з системами збереження версій [6]. Усе це допомагає працювати краще, зменшує кількість помилок, дає змогу швидко виправляти і змінювати програми. Також це місце для роботи з інструментами, які самі збирають проєкти. Так легше керувати всім, що входить до програми, і всіма її налаштуваннями. Це спрощує роботу і робить процес розробки більш зрозумілим і простим. Такий підхід дуже важливий, коли треба робити складні вебсистеми, що мають багато частин і функцій.

Java разом з програмою IntelliJ IDEA дає добрі умови для роботи. Створювати програми стає простіше. За допомогою них можна швидко писати код, легко бачити помилки та їх виправляти. Система працює без збоїв. Пізніше, коли потрібно зробити нові зміни чи додати щось нове, це не викликає труднощів. Усе стає зручним та швидким для розробника. Також це допомагає вирішувати задачі й підтримувати програму на довший час.

2.2 Фреймворки та серверна основа

Для створення серверної частини застосунку був вибраний новий підхід, який часто зараз використовують для розробки вебсайтів. Для цього було взято такі інструменти, які дають змогу робити роботу простіше і швидше, а продукт — кращим. В основі всієї роботи стоїть Spring Boot [1, 2, 7]. Це фреймворк, яка допомагає швидко збудувати сайт, адже там вже є основні настройки і свій вебсервер. Тому розробник може більше уваги дати важливим речам — головній роботі сайту. Налаштування не займають багато часу.

Ще одна важлива річ — це метод, коли керування передається самому коду, а не використовується напряму. Тобто, коли самі частини системи можуть «просити» інші частини, що їм треба (цю ідею називають інверсія керування чи скорочено IoC) [8, 9]. Для цього є спеціальний спосіб, який дозволяє вставляти одні частини в інші без жорсткого зв'язку. Тоді система стає більш гнучкою — легше щось змінити і перевірити окрему частину роботи. Завдяки цьому кожен модуль

системи виконує чітко визначену функцію, що позитивно впливає на підтримуваність і масштабованість проєкту.

Для роботи з HTTP-запитами й для створення вигляду для користувача, використовується модуль Spring MVC [2, 5]. Він працює за схемою Model-View-Controller [9]. При такому підході контролер приймає запит, бере дані, обробляє їх і передає далі на рівень вигляду. Це дає змогу чітко розрізнити, за що яка частина системи відповідає. Так стає простіше розбиратися з кодом, легше щось змінити чи додати нове. Проєкт виходить добре поділений і йому легше рости.

Для того, щоб зробити сторінки для користувачів, було використано Thymeleaf [5]. Це такий інструмент, який дає змогу з HTML створювати сторінки, котрі змінюються під час роботи. Він добре працює разом із сервером, тому контролер може одразу передати дані у шаблон сторінки, і це все так і з'явиться на сторінці, бо є спеціальні слова для цього [5, 9]. Це робить збір інтерфейсу легшим і швидшим для розробника. Особливістю цього інструменту є тісна інтеграція з серверною частиною, що дає змогу передавати дані безпосередньо з контролерів у шаблони та відображати їх за допомогою спеціальних виразів.

Серверна основа цієї системи створена так, щоб вона могла добре і довго працювати. Вона добре справляється з тим, щоб приймати і обробляти багато запитів від різних людей одночасно. Якщо треба, можна легко зробити більше функцій, не змінюючи те, що вже працює.

У цій роботі використано сучасні фреймворки, які дають системі високу швидкість, надійність і зручність для всіх користувачів, які будуть працювати із цією системою. Усе це разом допомагає зробити єдину і просту серверну архітектуру, що відповідає сучасним вимогам до такого роду програм і робить роботу всіх частин системи швидкою і зручною.

2.3 Аутентифікація та авторизація

Безпека – дуже важлива частина створення будь-якого сайту чи застосунку, особливо коли він працює з даними людей і потрібен різний рівень доступу. У цій

роботі для захисту даних використано Spring Security [2, 13, 19]. Цей фреймворк має багато гнучких та простих інструментів, які допомагають робити доступ безпечним для всіх користувачів і легко ним керувати.

У системі перед входом треба перевірити дані користувача. Це дає змогу точно сказати, хто зайшов у систему. Коли дані вірні, користувач може користуватися функціями згідно зі своєю роллю. Авторизація показує, до чого саме має доступ кожен користувач, та які дії може виконувати у системі [13, 18, 19]. У застосунку є розподіл користувачів на групи. Основні групи — це адміністратори, викладачі та учні. Для кожної ролі є список того, що дозволено робити. Такий підхід не дає людям, які не мають права, бачити або змінювати важливі частини системи.

Ще важливо добре захищати дані користувачів. Паролі не зберігають у простому вигляді. Вони шифруються за допомогою сучасних способів хешування [13, 18, 19]. Це допомагає зберегти безпеку й не дозволити зловмисникам дізнатись справжні паролі, якщо вони отримають доступ до даних. Також система контролює доступ до окремих URL-ресурсів і виконує перевірку прав користувача перед виконанням кожної операції.

Завдяки використанню зазначених механізмів досягається належний рівень захисту даних, забезпечується контроль доступу та підвищується загальна надійність функціонування застосунку.

2.4 Робота з базами даних

Зберігання й обробка даних у цій вебсистемі зроблені за допомогою реляційної бази даних MySQL [3]. Така база дає змогу краще тримати інформацію, зв'язувати між собою різні частини та слідкувати, щоб дані були цілісними й не губилися. Вибір саме цієї СКБД допомагає швидко працювати з великими об'ємами даних, легко робити запити, навіть якщо вони складні, а також бути впевненими, що дані залишаться у безпеці під час роботи всього застосунку.

Щоб сервер міг спілкуватися з базою, було використано Spring Data JPA [2,

8, 9]. Завдяки цій технології, можна просто зберігати дані у таблицях, а у програмі ці дані можна бачити і використовувати як звичайні об'єкти. Окрім того, для роботи часто застосовуються різні анотації, що ще більше пришвидшують та спрощують розробку програми.

Однією з основних переваг такого підходу є те, що базові дії з даними, як додати, знайти, змінити чи прибрати, можна виконати без потреби писати багато складних SQL-запитів. Більша частина стандартних дій робиться автоматично через репозиторії – це скорочує код і зменшує шанс, що виникне помилка. Також, якщо потрібно, можна застосувати свої запити для більш важких завдань з обробки даних.

Такий спосіб роботи з базою даних приносить більше свободи та дозволяє зручно слідкувати за системою. Він дає змогу легко змінювати дані, пристосовувати їх до нових правил і дає хороше спілкування між різними частинами програми. Це допомагає швидко додавати нові можливості та робить підтримку системи простішою.

2.5 Робота з моделями даних

Під час створення цієї вебсистеми використано простий спосіб роботи та показу даних. Усі головні речі в системі, як класи, вчителі, уроки, а також розклад, зроблені у програмі як окремі частини — класи на Java. Кожен клас створює свою річ, її головні риси і як вона має зв'язок з іншими частинами системи [8, 9].

Щоб усі дані між програмою та базою даних працювали добре разом, застосовано просту схему для їх з'єднання. Для цього в програмі були використані JPA-мітки [2, 8, 9]. З такими мітками можна без проблем зв'язати та змінити те, як працюють класи і як зберігаються дані. Крім того, за допомогою анотацій визначаються різні типи зв'язків між сутностями, наприклад «один до одного», «один до багатьох» або «багато до багатьох», що забезпечує коректне відображення реальних взаємозв'язків у системі [8]. Застосування такого підходу сприяє підвищенню рівня абстракції під час роботи з даними, оскільки розробник оперує

об'єктами, а не безпосередньо записами у таблицях. Це забезпечує цілісність даних і узгодженість між різними рівнями застосунку, оскільки всі зміни проходять через єдину модель.

Використання об'єктно-орієнтованого моделювання даних у поєднанні з механізмами об'єктно-реляційного відображення дозволяє створити гнучку, масштабовану та зручну у супроводі структуру даних, яка відповідає вимогам сучасних інформаційних систем.

2.6 Контейнеризація і розгортання

Щоб організувати розгортання вебсистеми, було обрано спосіб контейнеризації [4, 10]. Такий спосіб допомагає зробити однакове середовище для роботи програми і полегшує її запуск на різних платформах. У цьому проекті використано Docker [4, 10]. Docker дозволяє скласти всю програму, разом з потрібними їй бібліотеками, залежностями і налаштуваннями, у контейнер. Завдяки цьому, програма працює окремо і не залежить від інших програм чи налаштувань на комп'ютері. Це допомагає не мати проблем через різні конфігурації на різних пристроях. Для керування кількома контейнерами використано інструмент Docker Compose, який дозволяє описати конфігурацію всієї системи в одному файлі. Зокрема, у межах застосунку окремо визначено контейнер для серверної частини та контейнер для бази даних. Це дає змогу одночасно запускати всі необхідні компоненти системи однією командою. Використання контейнеризації також забезпечує відтворюваність середовища, тобто можливість запуску застосунку з однаковими параметрами незалежно від платформи чи оточення [10].

Таким чином, застосування технологій Docker та Docker Compose дозволяє підвищити стабільність роботи системи, спростити процес її розгортання та забезпечити узгодженість середовища виконання на всіх етапах життєвого циклу застосунку [4].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБСИСТЕМИ УПРАВЛІННЯ ШКІЛЬНИМ РОЗКЛАДОМ

3.1 Загальна архітектура системи

Під час створення плану для вебзастосунку з керування шкільним розкладом було зроблено головну схему роботи зображену на рисунку 3.1. У цій схемі видно, як все працює разом, з чого складається сайт, як частини між собою співпрацюють і що роблять з даними. Головна ідея цього кроку — зробити такий план, щоб людям було просто користуватися сайтом, щоб він працював добре і довго, і в майбутньому його можна було легко змінювати чи додавати нові можливості.

Система, яка була розроблена, працює за схемою клієнт-сервер. Це значить, що робота розділена на дві великі частини — та, що бачить і з нею працює користувач (клієнтська частина), і та, що зберігає дані та все рахує (серверна частина). Такий спосіб є дуже звичний для багатьох нових сайтів. Він дає змогу швидко і добре працювати з інформацією та спілкуватися з клієнтами, які користуються сайтом.

Клієнтська частина допомагає бачити інтерфейс та працювати з ним. Тут користувач може дивитись розклад, обирає що потрібно, пише свої дані або починає дії. На цьому рівні з системою йде спілкування: людина шукає, тисне на кнопки, надсилає запити. Потім клієнт відправляє ці запити на сервер.

Важливо, щоб на клієнтському рівні все було ясно і просто. Це потрібно, бо систему можуть використовувати різні люди, з різним досвідом у техніці. Простий вигляд та зрозумілі кнопки роблять роботу легшою для всіх.

Серверна частина отримує ці запити і обробляє їх. Вона перевіряє чи дані правильні, робить все головне у програмі і з'єднується з базою даних. На сервері часто треба додати, змінити або видалити дані. Також сервер стежить, щоб усе було правильно, не було помилок, і дані не зникали. Він піклується, щоб все працювало добре, а інформація була ціла і не втрачалася.

Якщо застосунок побудований на клієнт-сервері, то це дає змогу зробити всю структуру простішою. У цій моделі кожна частина знає свою роботу й робить тільки те, що треба. Так легко міняти або покращувати будь-який шматок системи, бо все працює окремо й не треба перебудовувати цілу систему.

У межах архітектурного проектування також було визначено ролі користувачів, які взаємодіють із системою. Було виділено три головні типи користувачів: адміністратор, учитель і учень. Завдяки такій схемі легко слідкувати за тим, хто має доступ до певної частини системи, і тримати все під контролем.

Адміністратор має найбільше прав у системі. Він керує всім, що є в ній. Він може створювати новий розклад, змінювати вже існуючий чи видаляти його. Також адміністратор керує класами, вчителями та уроками. Вчителі можуть бачити свій розклад, що допоможе їм краще планувати роботу. Учні можуть переглядати розклад свого класу та швидко дізнатись, які в них сьогодні уроки та коли вони починаються.

Уся система зроблена так, щоб кожен знав свої задачі і міг зручно працювати. Тут важливо, щоб було просто і зрозуміло користуватись. Також система допомагає усім працювати разом: і учням, і вчителям, і адміністратору. Вона добре зберігає всю потрібну інформацію й може стати ще кращою в майбутньому.

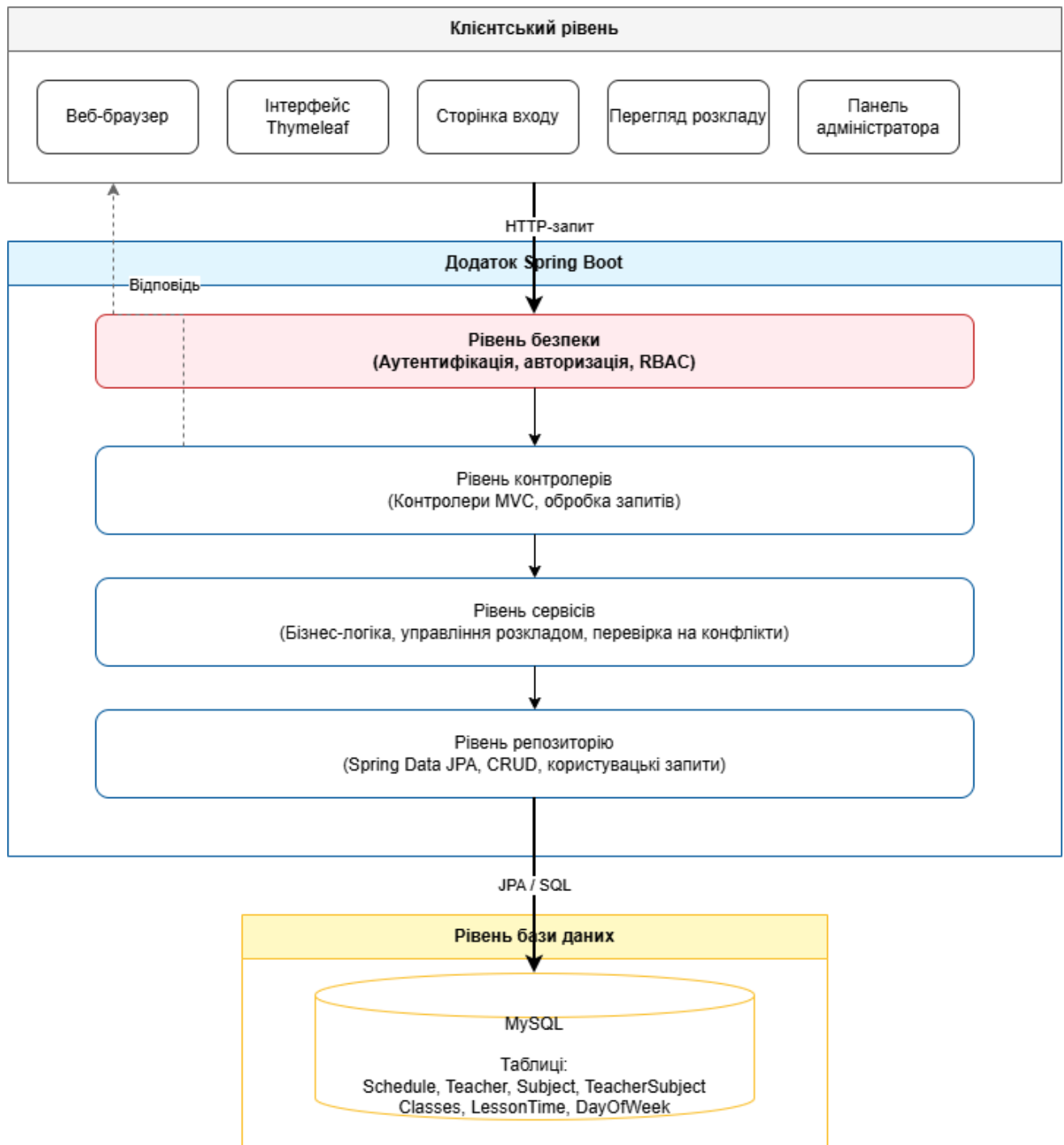


Рис. 3.1 Діаграма архітектури

3.2 Сценарії взаємодії користувачів із системою

Для кращого розуміння принципів роботи розробленої вебсистеми було розглянуто основні сценарії взаємодії користувачів із системою. Аналіз таких

сценаріїв дозволяє детально простежити послідовність дій користувача під час роботи з вебзастосунком, а також визначити, яким чином система реагує на різні запити та обробляє введені дані. Це дає можливість оцінити правильність реалізації функціональних можливостей системи та перевірити відповідність роботи вебзастосунку поставленим вимогам.

Опис сценаріїв використання також дозволяє продемонструвати логіку взаємодії між користувачем і системою, зокрема механізми перевірки доступу, обробки даних та відображення інформації. Завдяки цьому можна визначити, чи забезпечує система коректне виконання необхідних функцій, чи правильно реалізовано розподіл прав доступу та чи стабільно працюють механізми авторизації й керування розкладом.

У розробленій системі важливе місце відведено сценаріям перегляду розкладу для користувачів із базовими правами доступу, зокрема учнів і викладачів. Для таких користувачів система забезпечує швидкий доступ до актуального розкладу занять відповідно до їхніх потреб. Учні можуть переглядати власний навчальний розклад, а викладачі — інформацію про заплановані заняття та навчальні групи. Усі дані відображаються відповідно до ролі користувача після успішної авторизації в системі.

Також одним із ключових сценаріїв є редагування розкладу, доступ до якого надається виключно адміністратору системи. Адміністратор має можливість створювати нові записи, редагувати наявні заняття, змінювати час проведення уроків, аудиторії або викладачів, а також видаляти непотрібні записи. Під час виконання цих дій система автоматично перевіряє коректність введених даних та виявляє можливі конфлікти у розкладі. Такий підхід забезпечує цілісність інформації, запобігає виникненню помилок і гарантує актуальність даних для всіх користувачів вебсистеми.

3.2.1 Отримання розкладу (учень, викладач)

Переглядати розклад – це майже найосновніше, що тут можна зробити, адже саме ця можливість найбільше потрібна звичайним користувачам. Усе починається з того, що людина входить на потрібну сторінку і проходить вхід до системи. Після цього відкривається потрібний інтерфейс, і користувач бачить все, що йому треба.

Після відкриття сторінки клієнтська частина формує HTTP-запит до серверної частини системи [5]. Це зображено на рисунку 3.2. У цьому запиті передається інформація про користувача, зокрема його роль. Потім користувач може обрати розклад «Для викладача» або «Для учня» залежно від ролі і обрати необхідний клас, що дозволить серверу визначити, який саме розклад необхідно надати.

Отримавши запит, серверна частина виконує його обробку, що включає визначення параметрів вибірки даних та формування відповідного запиту до бази даних [3]. У базі даних зберігається структурована інформація про навчальний процес, зокрема дані про предмети, викладачів, класи, дні тижня та часові інтервали проведення занять. На основі цих даних формується узагальнена модель розкладу, яка відповідає запиту користувача.

Після отримання необхідної інформації з бази даних сервер здійснює її обробку та структурування, щоб підготувати дані до відображення. Наприклад, створюється проста таблиця. У цій таблиці кожен запис містить інформацію про предмет, клас, викладача, час. Уся ця підготовлена інформація далі повертається назад клієнту у вигляді простої відповіді на запит.

Далі клієнтська частина показує отримані дані у формі таблиці, яка легко читається. Це дає змогу будь-якому користувачу швидко знайти потрібну інформацію без зайвого часу. Так зберігання даних стає простим та однаковим для всіх, бо всі зміни йдуть просто через сервер. Це означає, що як тільки щось змінюється, всі користувачі одразу бачать це, і не треба тримати багато копій даних на різних пристроях.

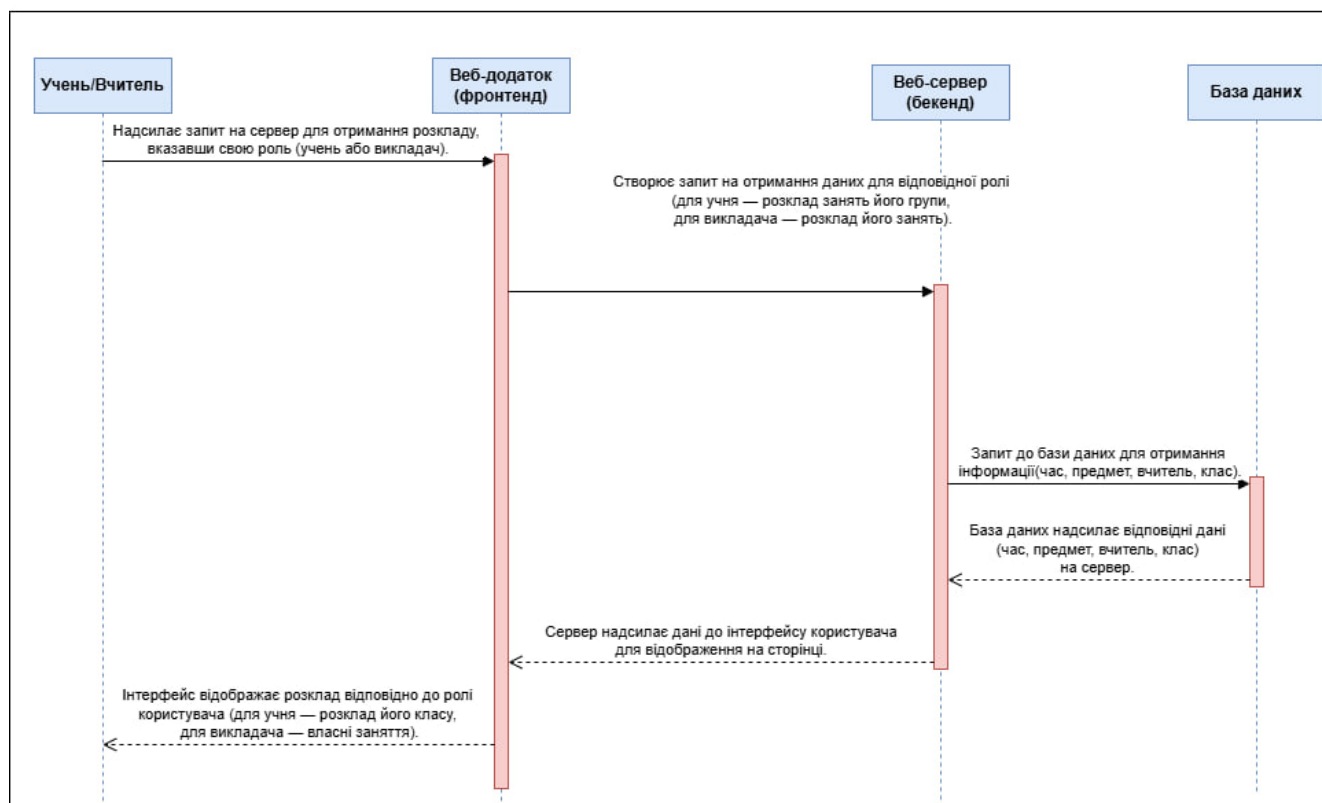


Рис. 3.2 Діаграма послідовності для викладача та учня

3.2.2 Зміна розкладу (адміністратор)

Сценарій зміни розкладу, який вказано на рисунку 3.3, дуже важливий для адміністратора, оскільки саме він відповідає за формування та актуалізацію навчального процесу. У цьому сценарії адміністратор має змогу створювати новий розклад, змінювати вже наявний або видаляти старий.

Усе починається з того, що адміністратор заходить у потрібну частину системи «Для адміністратора». Там він бачить вікно для роботи з розкладом. У цій секції він може змінити головну інформацію про заняття. Це такі речі, як клас, що вчитиметься, предмет, ім'я вчителя, день тижня і час, коли буде урок.

Після внесення змін адміністратор підтверджує виконання операції, внаслідок чого клієнтська частина формує запит до серверу та передає введені дані. Серверна частина отримує цей запит і виконує його обробку, що включає перевірку коректності введених даних [2, 5]. На цьому етапі система може здійснювати

додаткові перевірки, наприклад, на наявність конфліктів у розкладі, таких як накладання занять для одного викладача або класу [2].

Після того як сервер пройде перевірку, він бачить, що треба зробити далі. Це може бути новий запис, зміна вже наявного або просто видалення. Далі сервер зберігає зміни.

Якщо все проходить добре, сервер готує відповідь. У цій відповіді є нові дані. Цю відповідь сервер надсилає клієнту. Інтерфейс програми зміниться після того, як прийде ця відповідь. Так на сторінці одразу видно, що сталося і що зробив адміністратор.

Такі дії дають змогу керувати розкладом в одному місці. Це зменшує ризик помилки та гарантує цілісність даних.

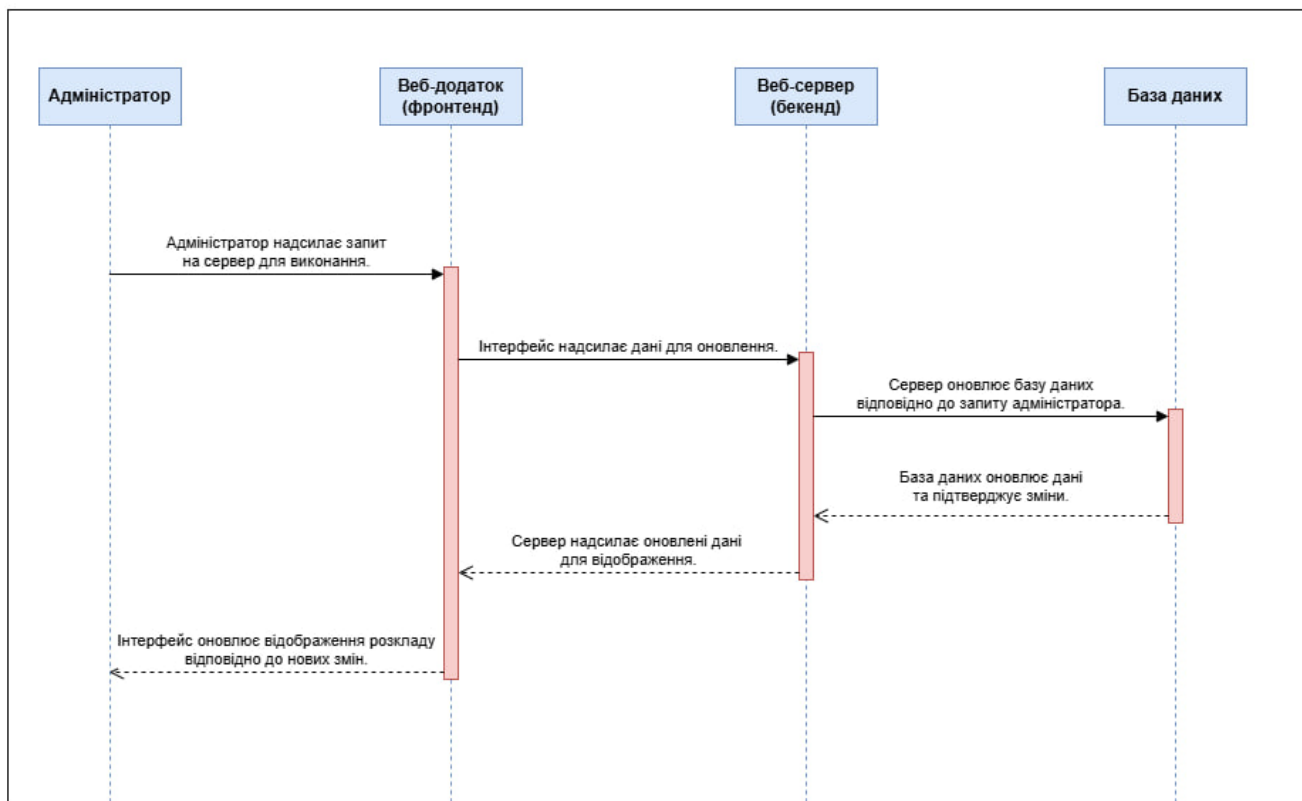


Рис. 3.3 Діаграма послідовності для адміністратора

3.3 Проектування структури даних

Одним з важливих кроків, коли створюється вебзастосунок, є план, як саме будуть зберігатися дані. Важливо не просто зберігати інформацію, а й робити це

так, щоб все працювало добре, швидко та можна було додати щось нове у майбутньому. У цій роботі всі дані зберігаються у реляційній базі даних [3]. Це дає змогу легко розкласти інформацію по таблицях, зробити зв'язки між даними та бути впевненим, що дані завжди правильні.

База даних має кілька таблиць, і всі вони пов'язані між собою. У кожній таблиці є свої окремі дані, що містять різні дані. Головна таблиця тут — таблиця Schedule. Саме вона містить розклад. Сюди потрапляють всі дані про заняття. Крім того, у ній є зв'язки з іншими таблицями, і все це працює через зовнішні ключі. Кожне заняття, яке є у розкладі, має посилання на потрібні дані в інших таблицях. Так інформація зберігається просто, зручно та зрозуміло.

До основних таблиць системи належать:

- Teacher, яка містить інформацію про викладачів, зокрема їх ідентифікатори, прізвища, імена та інші необхідні атрибути;
- Subject, яка зберігає перелік навчальних дисциплін;
- Classes, яка містить інформацію про навчальні класи;
- LessonTime, яка визначає часові інтервали проведення занять (номер уроку, час початку та завершення);
- DayOfWeek, яка містить перелік днів тижня, що використовується для структурування розкладу;
- TeacherSubject є проміжною таблицею, яка реалізує зв'язок між викладачами та предметами;
- Schedule, яка є основною таблицею, що об'єднує всі дані для формування повного запису розкладу.

Кожен рядок — це окреме заняття. У таблиці Schedule є ключі, що ведуть до інших таблиць: Teacher, Subject, Classes, LessonTime та DayOfWeek. Це дає змогу зібрати всі дані про заняття одним рядком, не множити інформацію. Один вчитель, предмет чи час зберігається в своїй таблиці раз. У розкладі є лише посилання.

Такий підхід має кілька важливих переваг. По-перше, він дозволяє уникнути надлишковості даних, що зменшує обсяг бази та спрощує її обслуговування. По-друге, забезпечується цілісність інформації, оскільки всі зв'язки між таблицями

контролюються за допомогою зовнішніх ключів. По-третє, це дає можливість формувати складні запити для отримання повної інформації про розклад, об'єднуючи дані з кількох таблиць.

Особливу увагу приділено реалізації зв'язків між сутностями. Зокрема, зв'язок між викладачами та предметами реалізовано за допомогою проміжної таблиці TeacherSubject, що відповідає типу «багато до багатьох». Це означає, що один викладач може викладати кілька предметів, а один предмет може викладатися кількома викладачами. Такий підхід дозволяє гнучко описувати навчальний процес і легко розширювати систему у разі зміни вимог.

Зв'язки між таблицями можна описати таким чином:

- один викладач (Teacher) може мати багато записів у таблиці Schedule;
- один клас (Classes) має багато занять у розкладі;
- один предмет (Subject) може входити до багатьох записів розкладу;
- кожен запис у Schedule відповідає одному дню (DayOfWeek) та одному часовому інтервалу (LessonTime);
- зв'язок між Teacher і Subject реалізовано через таблицю TeacherSubject.

Така структура даних є нормалізованою, що дозволяє мінімізувати дублювання інформації та підвищити ефективність роботи системи. Крім того, вона є достатньо гнучкою для подальшого розширення, наприклад, додавання нових сутностей або атрибутів без суттєвих змін у вже існуючій моделі.

Описана структура бази даних може бути наочно представлена у вигляді ER-діаграми, яка зображена на рисунку 3.4 та відображає всі сутності системи, їх атрибути та зв'язки між ними. Така діаграма дозволяє краще зрозуміти логіку побудови бази даних і використовується як важливий елемент документації системи.

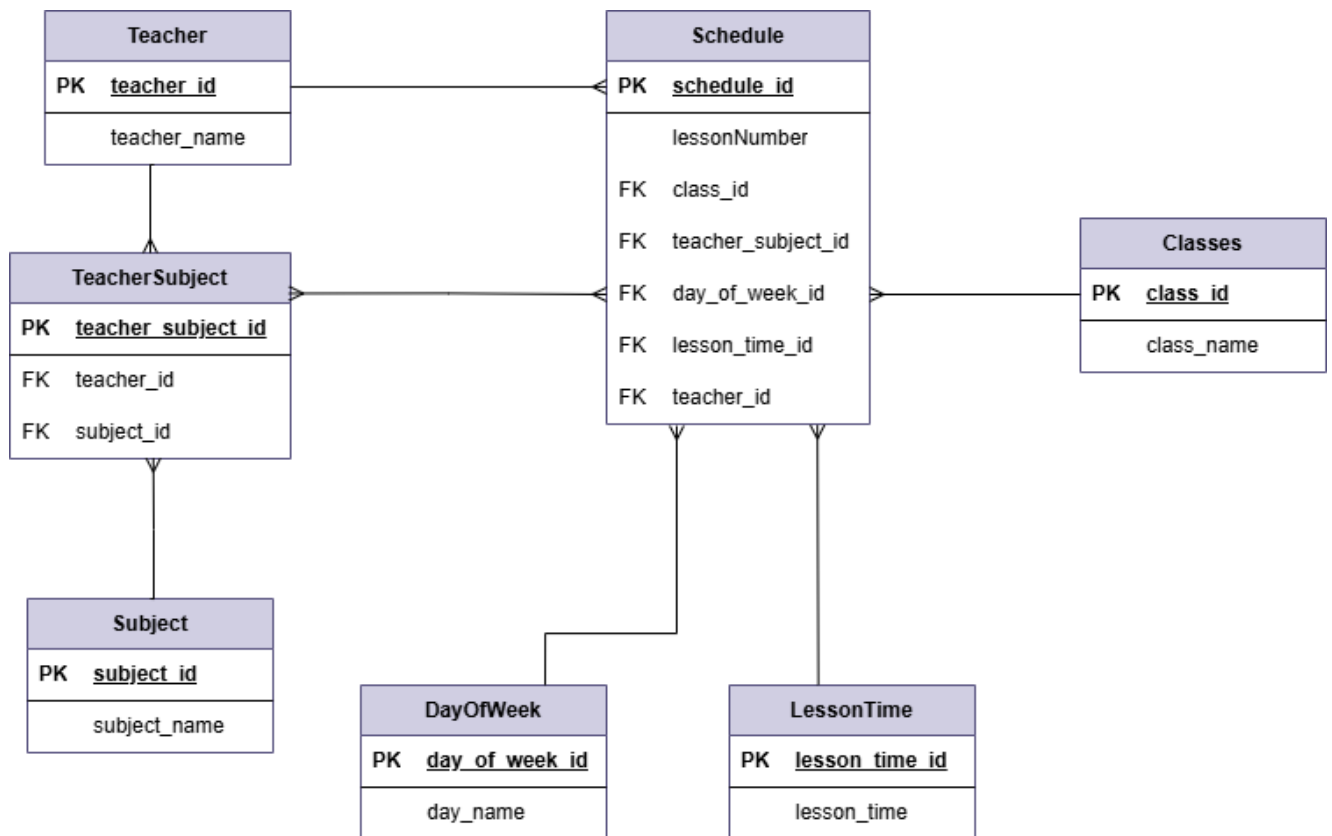


Рис. 3.4 Модель «сутність-зв'язок» бази даних вебсистеми управління шкільним розкладом

Модель даних є дуже важливою частиною будь-якої системи. Вона допомагає визначити, як виглядають і як зберігаються всі дані. У цій новій системі модель даних зроблена як набір класів, які дуже схожі на таблиці у базі даних.

Основною сутністю є користувач (User), який представляє обліковий запис у системі. Кожен користувач має унікальний ідентифікатор, логін та пароль. Для забезпечення розмежування прав доступу кожен користувач пов'язаний із роллю (Role), яка визначає його можливості у системі.

Сутність Role використовується для зберігання інформації про ролі користувачів, такі як адміністратор, викладач або учень. Це дозволяє реалізувати механізм авторизації на основі ролей.

Сутність Teacher представляє викладача та містить інформацію про нього. Вона пов'язана із сутністю User, що дозволяє забезпечити автентифікацію викладачів у системі.

Сутність *Classes* використовується для представлення навчальних класів. Вона містить інформацію про назву класу та використовується для групування учнів і формування розкладу.

Сутність *Subject* описує навчальні дисципліни. Вона використовується для визначення предметів, які викладаються у навчальному процесі.

Центральною сутністю системи є *Schedule*, яка представляє запис розкладу. Вона містить інформацію про клас, викладача, предмет, день тижня та номер уроку. Саме ця сутність забезпечує збереження та відображення розкладу занять.

Для того, щоб зробити зв'язки між сутностями, взято анотації JPA [2]. Також створені зв'язки типу «багато до одного», щоб показати, як розклад має зв'язок з іншими частинами, як класами, вчителями та предметами.

Такий спосіб створює модель даних, яку легко змінити чи додати нове. Він допомагає зберігати всю інформацію правильно та не втрачати дані. Також, такий підхід починає робити запити до бази даних простими і швидкими, тому що вся інформація вже добре зв'язана між собою.

На рисунку 3.5 представлена діаграма варіантів використання вебсистеми управління шкільним розкладом. Вона демонструє основні можливості системи та взаємодію користувачів із функціоналом застосунку. Діаграма дозволяє наочно показати, які дії можуть виконувати різні типи користувачів, а також як між собою пов'язані окремі функції системи.

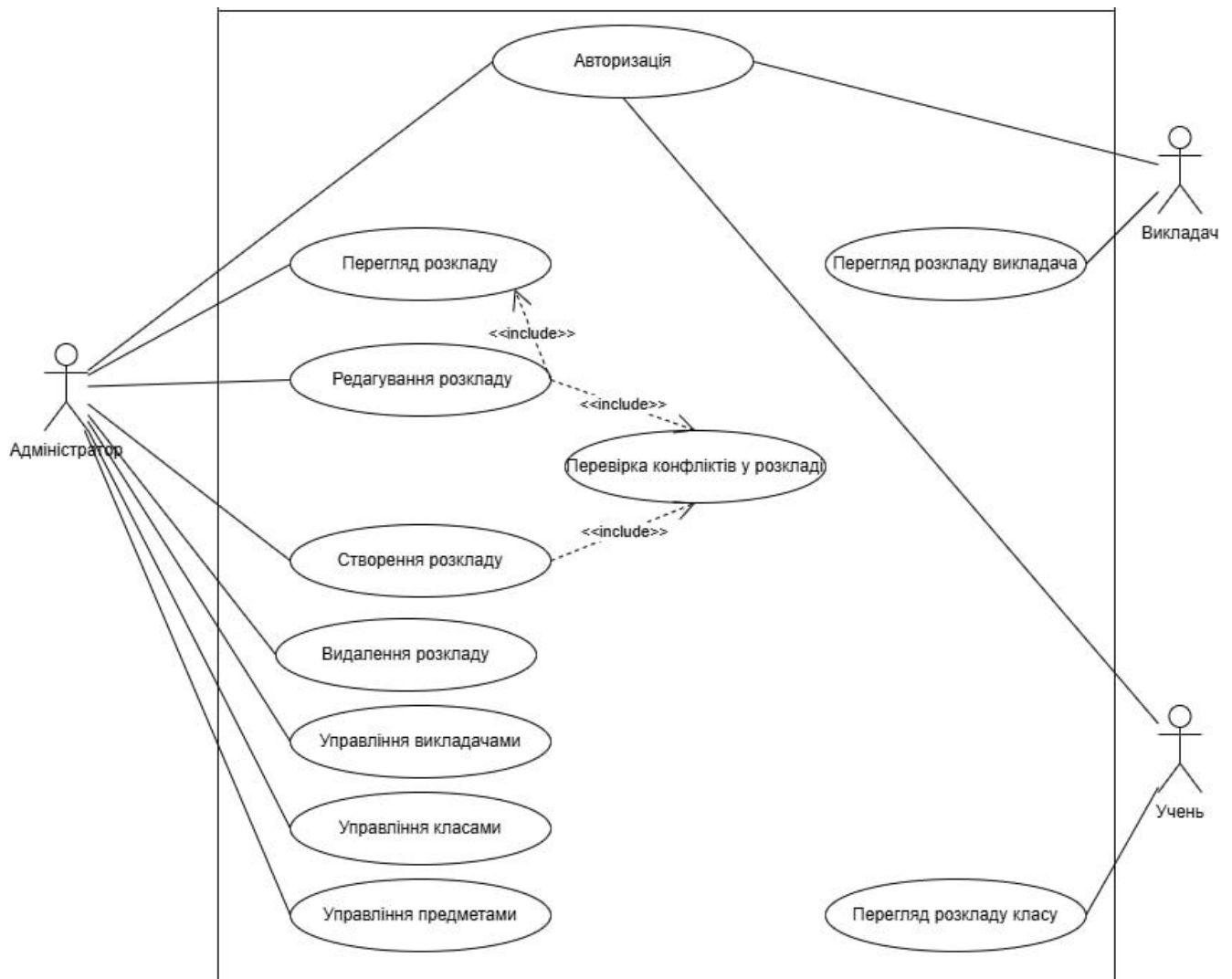


Рис. 3.5 Діаграма варіантів використання

У системі визначено три основні ролі користувачів: адміністратор, викладач та учень. Кожна роль має власний набір доступних дій відповідно до функціональних обов'язків і рівня доступу.

Адміністратор є основним користувачем із найбільшими правами доступу. Він входить у систему через авторизацію, бачить розклад, може створити новий розклад, змінювати його або видаляти те, що більше не треба. Окрім цього, адміністратор має можливість керувати викладачами, класами та навчальними предметами. Такий підхід забезпечує централізоване управління всією системою та дозволяє підтримувати актуальність даних.

Викладач теж може увійти в систему. Після цього він бачить тільки свій розклад. Він знає, коли і де будуть його заняття, і в якому класі це буде.

Учень у системі має свою роль і бачить те, що йому треба для навчання. Завдяки цьому учні можуть швидко отримувати інформацію про предмети, час проведення уроків та послідовність занять. Такий поділ допомагає кожному працювати просто й легко.

На діаграмі також відображено функцію перевірки конфліктів у розкладі. Вона використовується під час створення та редагування розкладу і позначена зв'язком <<include>>. Це означає, що під час виконання цих операцій система автоматично перевіряє, чи не виникає накладання занять для одного викладача або класу в однаковий час. Такий механізм дозволяє забезпечити цілісність даних та уникнути помилок при формуванні розкладу.

Так, діаграма варіантів використання показує головні можливості вебсистеми для роботи з шкільним розкладом. Тут видно, як користувачі працюють із системою, що саме вони можуть робити. Діаграма також дає зрозуміти, які права доступу має кожна роль, і як поділяється робота між усіма користувачами. Це допомагає краще бачити, як усе працює у системі та що може кожна людина.

На рисунку 3.6 представлено діаграму діяльності вебсистеми управління шкільним розкладом. Діаграма показує послідовність дій користувача, взаємодію із серверною частиною системи та роботу з базою даних під час виконання основних операцій застосунку.

На початку роботи користувач відкриває сторінку авторизації та вводить свої облікові дані. Після цього система виконує перевірку правильності введеної інформації. Якщо дані введені неправильно, користувач отримує повідомлення про помилку та може повторити спробу входу. У разі успішної авторизації система визначає роль користувача та надає доступ до відповідних функцій.

Для учня передбачено перегляд розкладу класу, а для викладача — отримання власного розкладу занять. У цих сценаріях система звертається до бази даних для отримання необхідної інформації та відображає її користувачу.

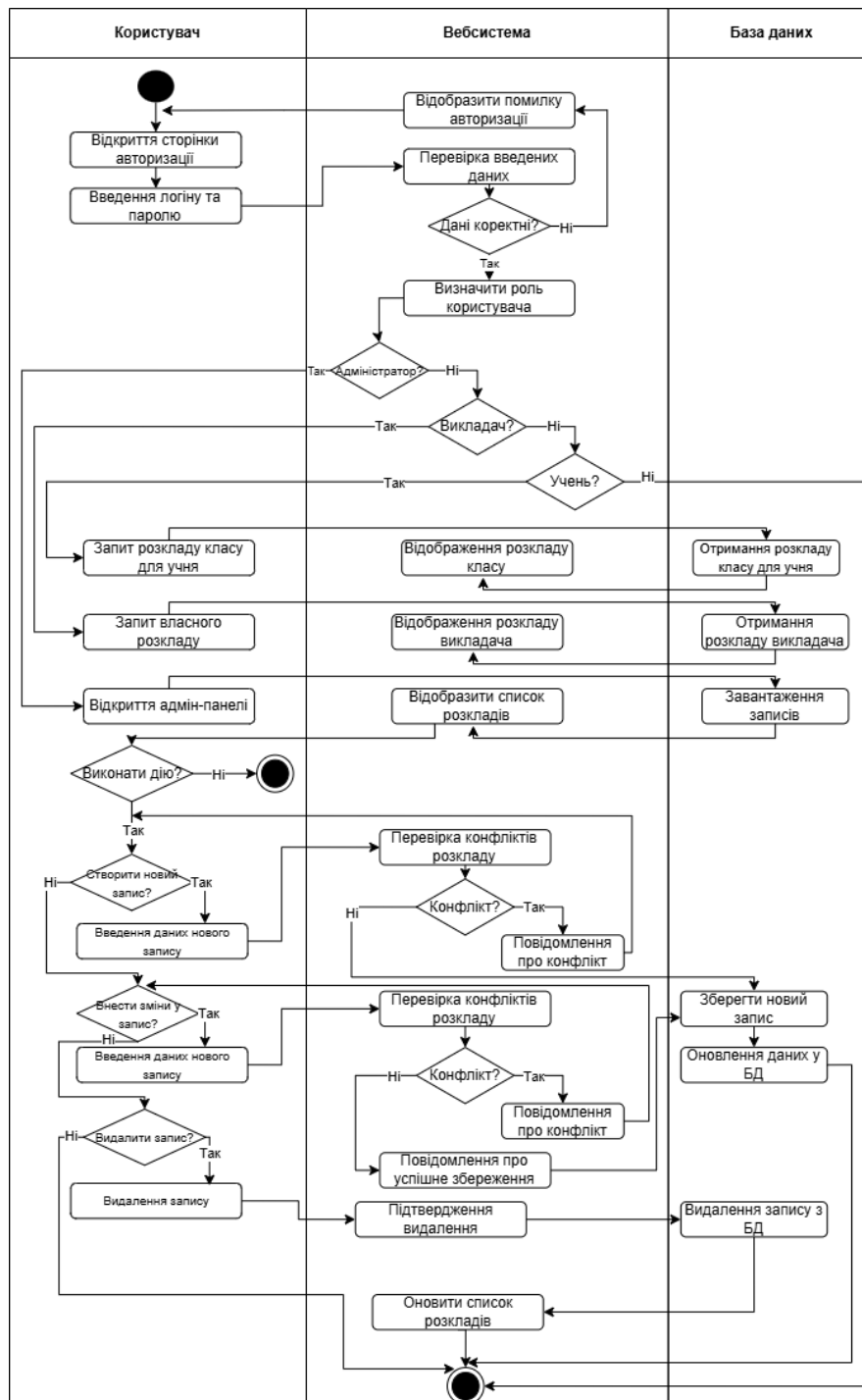


Рис. 3.6 Діаграма діяльності

Окрему частину діаграми займають дії адміністратора. Після відкриття адміністративної панелі адміністратор може переглядати список розкладів, додавати нові записи, змінювати наявні або видаляти їх. Перед збереженням змін система виконує перевірку можливих конфліктів розкладу. Якщо конфлікт

виявлено, користувач отримує відповідне повідомлення. У випадку відсутності помилок дані зберігаються у базі даних, після чого список розкладів оновлюється.

Подана діаграма дає змогу наочно показати логіку функціонування вебсистеми, взаємозв'язок між окремими компонентами та порядок виконання основних процесів у застосунку.

На рисунку 3.7 представлено діаграму класів серверної частини вебсистеми управління шкільним розкладом. На цій схемі показано основні сервіси, репозиторії та зв'язки між ними, які забезпечують роботу застосунку.

Центральне місце займає клас `ScheduleServiceImpl`, оскільки саме він відповідає за роботу з розкладом. Через нього виконується отримання списку занять, пошук розкладу для певного класу або викладача, а також збереження й видалення записів. Для цього сервіс взаємодіє з іншими компонентами системи, зокрема з сервісами викладачів, класів, предметів та днів тижня.

Кожен окремий модуль має власний сервіс і репозиторій. Наприклад, `TeacherServiceImpl` працює з інформацією про викладачів, `ClassesServiceImpl` — з даними класів, а `SubjectServiceImpl` — з предметами. Такий підхід дозволяє розділити логіку системи на окремі частини, щоб кожен компонент відповідав лише за свою область роботи.

Також на діаграмі можна побачити використання інтерфейсів сервісів і репозиторіїв. Це зроблено для того, щоб система була більш гнучкою та зручною для підтримки. Реалізація сервісу працює через відповідний інтерфейс, а доступ до бази даних виконується за допомогою репозиторіїв.

Окремо увага приділена зв'язкам між класами. Вони показують, які компоненти використовують інші сервіси або звертаються до репозиторіїв для отримання та збереження даних. Завдяки цьому структура застосунку виглядає організованою та зрозумілою, а додавання нових можливостей не потребує зміни всієї системи.

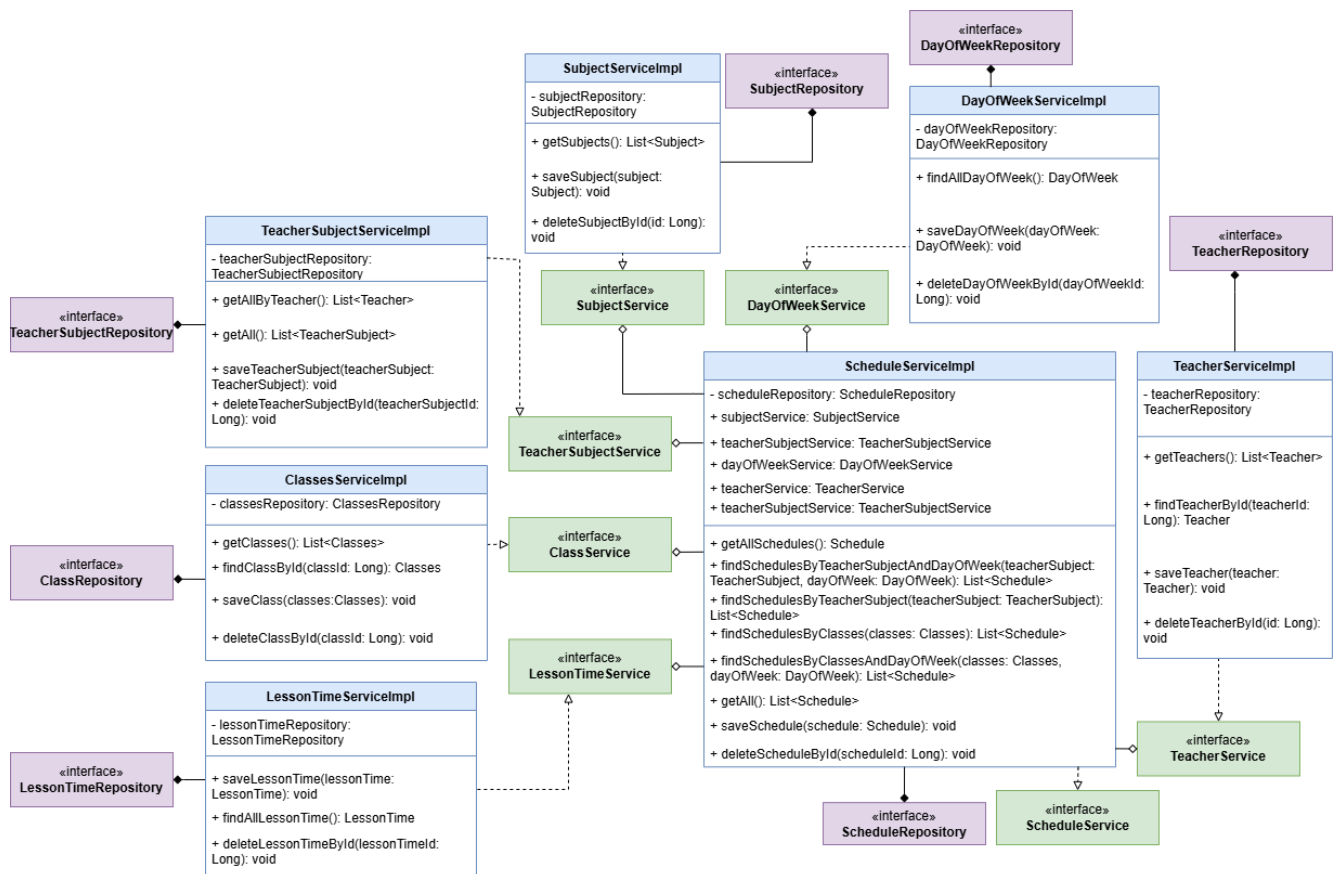


Рис. 3.7 Діаграма класів

Подана діаграма допомагає краще зрозуміти внутрішню структуру серверної частини застосунку та принцип взаємодії між основними компонентами вебсистеми.

3.4. Реалізація серверної частини

3.4.1 Реалізація доступу до даних

Для реалізації доступу до бази даних використовується підхід, заснований на застосуванні репозиторіїв. У межах даної системи використовується технологія Spring Data JPA [2], яка дозволяє значно спростити роботу з базою даних шляхом автоматичної генерації стандартних операцій.

Кожна сутність має відповідний репозиторій, який відповідає за виконання операцій над даними. Репозиторії реалізуються у вигляді інтерфейсів, що

наслідуються від базового інтерфейсу, який надає готові методи для виконання основних операцій.

До базових операцій належать створення, читання, оновлення та видалення записів. Вони реалізуються без необхідності написання додаткового коду, що значно скорочує час розробки та зменшує ймовірність виникнення помилок.

Використання репозиторіїв дозволяє ізолювати логіку доступу до даних від інших частин системи. Це забезпечує гнучкість у зміні реалізації роботи з базою даних, а також спрощує тестування та супровід застосунку.

3.4.2 Реалізація бізнес-логіки

Сервісний шар є центральним елементом системи, оскільки саме він відповідає за реалізацію бізнес-логіки застосунку. У цьому шарі виконуються всі операції, пов'язані з обробкою даних, перевіркою умов та координацією роботи інших компонентів.

Основною функцією сервісного шару є забезпечення коректності виконання операцій. Наприклад, перед створенням нового запису розкладу система перевіряє, чи не виникає конфліктів, пов'язаних із зайнятістю викладача або класу. Це дозволяє запобігти виникненню помилок та забезпечити узгодженість даних.

Сервісний шар також реалізує логіку створення, оновлення та видалення записів. Він взаємодіє з репозиторіями для отримання та збереження даних, а також забезпечує обробку виняткових ситуацій.

Однією з ключових функцій є перевірка конфліктів у розкладі. Вона передбачає аналіз існуючих записів і перевірку того, чи не збігаються часи занять для одного викладача або класу [2]. У разі виявлення конфлікту система блокує створення запису та повідомляє користувача про помилку.

Таким чином, сервісний шар забезпечує реалізацію всієї логіки роботи системи та виступає посередником між контролерами та репозиторіями.

3.4.3 Реалізація контролерів

Контролери є компонентами системи, які відповідають за обробку запитів від клієнта та формування відповідей. Вони виступають точкою входу до серверної частини застосунку та забезпечують взаємодію між користувачем і системою.

У системі реалізовано MVC-контролери [5]. Вони застосовуються для формування веб-сторінок із використанням шаблонізатора.

Кожен контролер містить набір методів, які відповідають за обробку різних типів HTTP-запитів [5]. Зокрема, GET-запити використовуються для отримання даних, POST — для створення нових записів, PUT — для оновлення, а DELETE — для видалення.

Контролери приймають дані від клієнта, передають їх у сервісний шар для обробки та повертають результат. При цьому вони не містять бізнес-логіки, що відповідає принципу розділення відповідальностей.

Особлива увага приділяється обробці помилок. У разі виникнення помилки контролер повертає відповідне повідомлення, що дозволяє користувачу зрозуміти причину проблеми та вжити необхідних заходів.

Таким чином, контролери забезпечують зручний та стандартизований спосіб взаємодії клієнта із серверною частиною застосунку, що є важливим для ефективної роботи всієї системи.

3.5 Реалізація системи безпеки

Забезпечення безпеки вебзастосунку є важливою частиною його реалізації, особливо з урахуванням того, що система працює з користувачами різних ролей та обробляє їхні дані. У даному проєкті механізми безпеки реалізовано з використанням фреймворку Spring Security [2], який дозволяє гнучко налаштовувати процеси входу в систему та розмежування доступу.

Автентифікація користувачів відбувається через стандартну форму входу, де користувач вводить свій логін і пароль. Після цього ці дані передаються на сервер, де проходять перевірку. Для підвищення рівня захисту паролі не зберігаються у відкритому вигляді — використовується хешування, що дозволяє уникнути витоку

конфіденційної інформації навіть у випадку доступу до бази даних. Якщо дані введені правильно, користувач отримує доступ до системи, якщо ні — система повідомляє про помилку.

Після входу в систему починає діяти механізм авторизації, який визначає, що саме може робити користувач. У застосунку використано підхід на основі ролей, де кожному користувачу призначається певна роль. Основними ролями є адміністратор, викладач та учень. Адміністратор має повний доступ до функціоналу системи, зокрема до редагування розкладу, тоді як викладачі та учні можуть лише переглядати відповідну інформацію.

Доступ до окремих частин системи обмежується на рівні конфігурації. Наприклад, адміністративна панель і всі операції, пов'язані зі зміною даних, доступні лише для користувачів із відповідними правами. Такий підхід дозволяє централізовано керувати правилами доступу та робить систему більш захищеною і зрозумілою в підтримці.

3.6 Реалізація інтерфейсу користувача

Клієнтська частина зроблена через Thymeleaf [2]. Це такий шаблонізатор, який дозволяє прямо на сервері формувати HTML-сторінки з даними. Тобто не просто статичні сторінки, а такі, що підставляють реальну інформацію з бази. За рахунок цього все досить зручно зв'язується: сервер обробив дані і одразу віддав готову сторінку.

У системі є кілька основних сторінок. Перша — це сторінка входу. Там відображається все стандартно: форма з логіном і паролем. Користувач вводить дані, вони перевіряються, і якщо такий користувач у базі даних є, то йому відкривається головна сторінка вебсистеми.

Далі основна сторінка — це перегляд розкладу. Розклад показується у вигляді таблиці: дні тижня, уроки, предмети, викладачі. Дані підтягуються із сервера і підставляються у шаблон.

Окремо є адмін-панель. Там більше функціоналу, а саме: можна додавати нові записи, редагувати існуючі, керувати викладачами, класами, предметами. Також є перевірки, щоб не було якихось конфліктів у розкладі.

Дані передаються з контролера у шаблон через модель. Тобто контролер підготував інформацію, передав її, а далі вона підставляється в HTML. Завдяки цьому сторінки виходять динамічними, але при цьому структура залишається зрозумілою. Це якраз те, що і закладається в MVC [2], [5].

3.7 Перевірка даних і обробка помилок

Коли йде робота з системою, дуже важливо, щоб дані, які дає користувач, були правильні. Також важливо, щоб сама система добре працювала, навіть коли є помилки. Тому у цьому застосунку є спосіб перевірити дані і прості дії для виправлення помилок.

Дані перевіряють на двох етапах. Перше — коли людина вводить інформацію у форму. Друге — на сервері, після того як дані вже надійшли. Наприклад, коли користувач робить чи змінює розклад, система дивиться, чи всі потрібні поля заповнені, і чи все вписане правильно. Якщо щось не так, користувач бачить підказку і може виправити. Це допомагає не пропустити помилку.

Окремо реалізована перевірка конфліктів у розкладі. Перед збереженням нового запису система перевіряє, чи не виникає накладання занять для одного викладача або класу в один і той самий час. Якщо такий конфлікт знаходиться, операція не виконується, а користувачу виводиться відповідне повідомлення.

Обробка помилок здійснюється на рівні контролерів і сервісного шару. У разі виникнення помилки система не завершує роботу аварійно, а повертає зрозуміле повідомлення, яке пояснює причину проблеми. Це дозволяє користувачу швидше зрозуміти, що пішло не так, і виправити свої дії.

Такий спосіб робить систему більш стійкою. Також це допомагає зменшити кількість помилок у базі даних і дає користувачеві змогу зручно працювати у застосунку.

4 ДЕМОНСТРАЦІЯ РОБОТИ СИСТЕМИ ТА ТЕСТУВАННЯ

4.1 Демонстрація роботи системи

Розроблена вебсистема управління шкільним розкладом призначена для автоматизації процесів створення, перегляду, редагування та адміністрування навчального розкладу. Використання системи дозволяє значно спростити роботу з розкладом занять, зменшити кількість помилок під час внесення змін та забезпечити швидкий доступ користувачів до актуальної інформації. Система реалізована у вигляді сучасного вебзастосунку, що забезпечує можливість роботи через браузер без необхідності встановлення додаткового програмного забезпечення.

Користувачі в цій системі поділені на групи за ролями. Адміністратор отримує більше прав: він може додати, змінити чи видалити записи в розкладі, керувати даними для навчання та стежити за всіма змінами. Учні та вчителі бачать свої уроки, можуть дізнатися про оновлення і завжди мають доступ до розкладу.

Розподіл ролей дає змогу обмежити права, щоб кожен міг робити тільки те, що дозволено його статусом. Усе це зменшує помилки та спрощує роботу.

Також система підтримує перевірку коректності введених даних та автоматичне виявлення конфліктів у розкладі, зокрема накладання занять для викладачів або навчальних класів. Це дозволяє уникнути помилок під час формування розкладу та забезпечує цілісність даних у системі. Після внесення змін інформація автоматично оновлюється та стає доступною для всіх користувачів, що забезпечує актуальність розкладу в реальному часі.

Розроблений вебзастосунок має зрозумілий інтерфейс користувача, що спрощує роботу з системою та забезпечує зручну навігацію між її функціями. У подальших підрозділах наведено основні екрани системи, приклади роботи функціональних можливостей та результати тестування реалізованого програмного забезпечення.

4.1.1 Сторінка авторизації

Після відкриття вебзастосунку користувач потрапляє на сторінку авторизації, яка зображена на рисунку 4.1. Для входу в систему необхідно ввести логін та пароль. Після успішної перевірки даних користувач отримує доступ до функціоналу відповідно до своєї ролі.

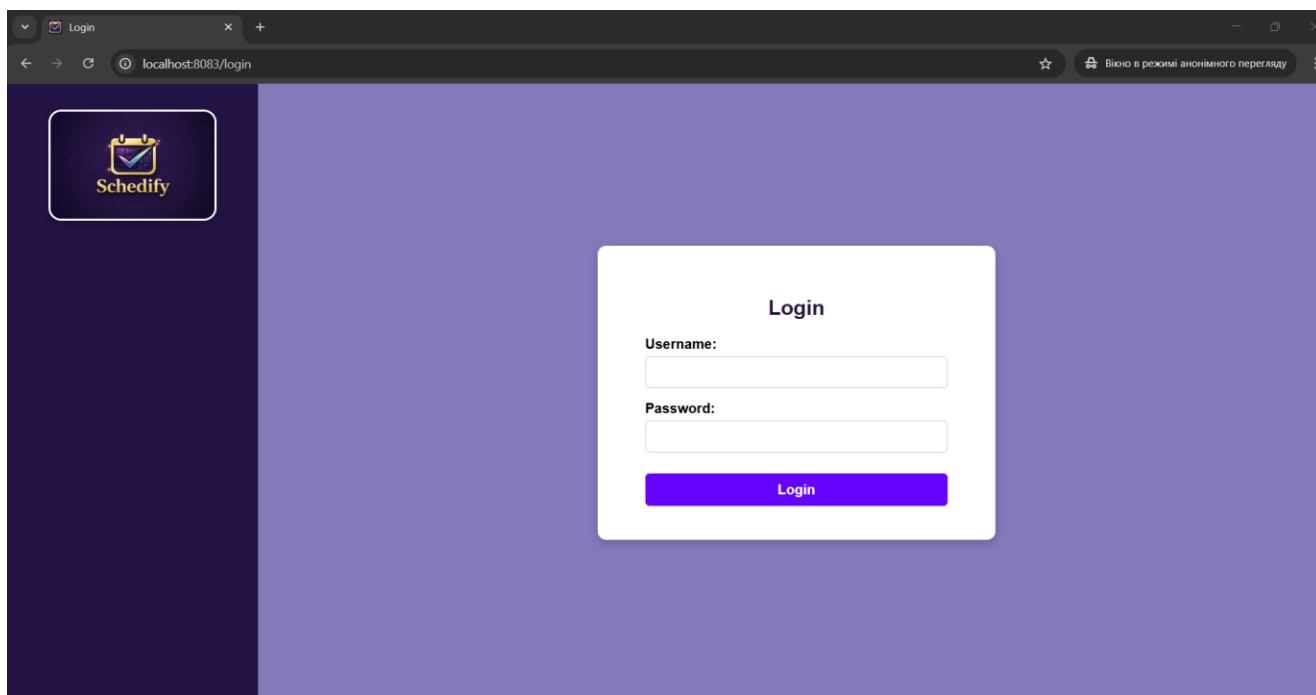


Рис. 4.1 Сторінка авторизації системи

4.1.2 Головна сторінка системи

Після успішної авторизації користувач потрапляє на головну сторінку вебсистеми, показану на рисунку 4.2. На цій сторінці розміщено основне меню навігації, яке дозволяє перейти до перегляду розкладу, сторінок роботи з класами, викладачами та окремими заняттями.

Інтерфейс системи побудований таким чином, щоб забезпечити швидкий доступ до основного функціоналу та зручну навігацію між розділами.

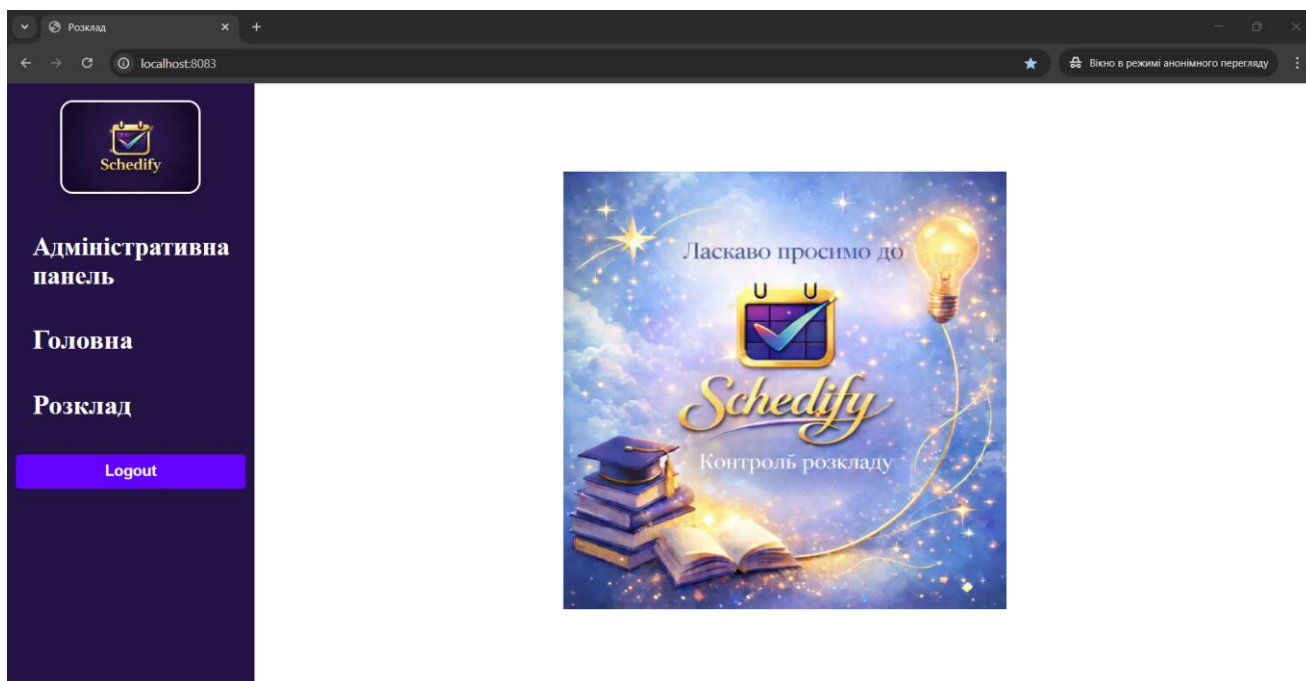


Рис. 4.2 Головна сторінка системи

На головній сторінці системи також реалізовано випадаюче меню «Розклад», показане на рисунку 4.3. Після натискання на нього користувачу відображаються дві основні секції: «Для учня» та «Для викладача».

Секція «Для учня» призначена для перегляду розкладу навчальних занять за класами. У цьому розділі користувач може обрати необхідний клас та переглянути інформацію про предмети, викладачів, день проведення заняття та час уроку. Це дозволяє швидко отримати актуальний розклад для конкретного класу.

Секція «Для викладача» використовується для перегляду індивідуального розкладу викладачів. Після вибору викладача система відображає всі заняття, які він проводить, із зазначенням класу, предмета та часу проведення уроку. Такий підхід спрощує контроль навантаження викладачів та забезпечує зручний доступ до інформації про їхні заняття.

Наявність випадаючого меню дозволяє організувати навігацію в системі більш зручною та забезпечує швидкий перехід між різними типами розкладу.

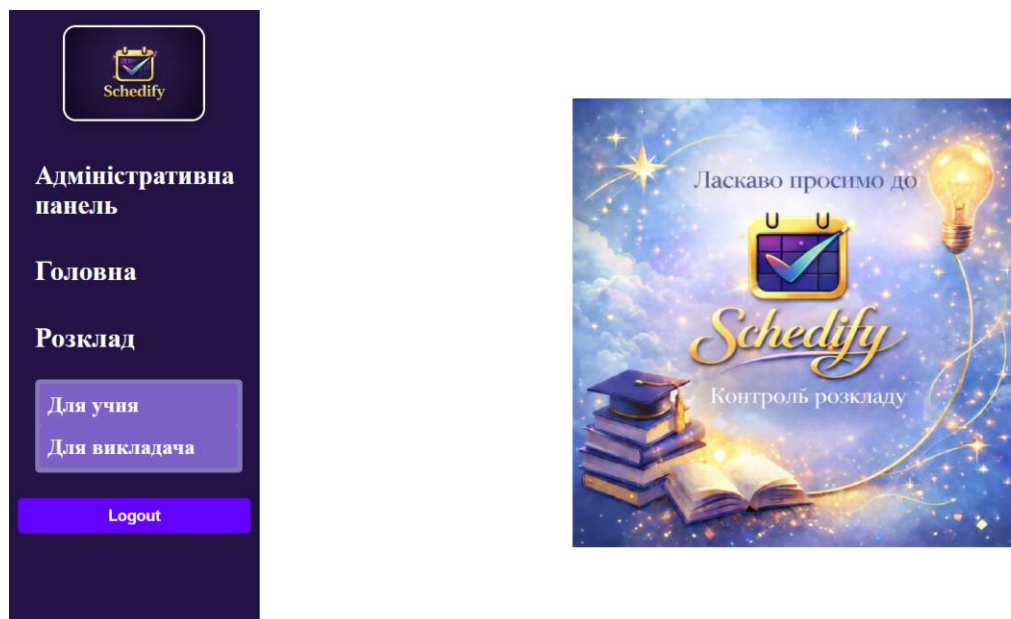


Рис. 4.3 Випадаюче меню «Розклад» на головній сторінці системи

4.1.3. Перегляд розкладу для класів

На сторінці розкладу для класів, яка зображена на рисунку 4.4, учень може обрати необхідний клас та переглянути актуальний розклад занять.

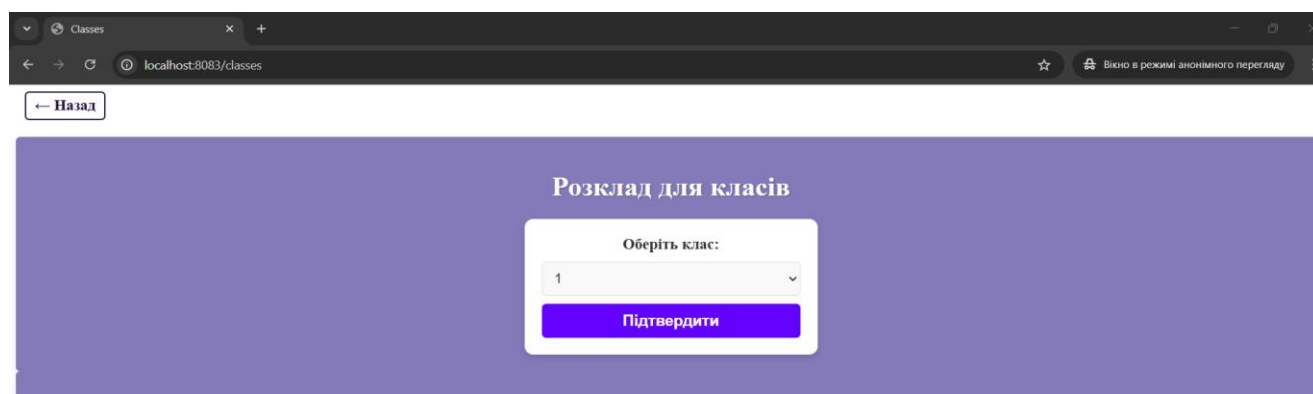


Рис. 4.4 Сторінка вибору класу для відображення розкладу

На рисунку 4.5 відображено список уроків із зазначенням предмета, викладача, дня тижня та часу проведення заняття, який система показує після вибору класу.

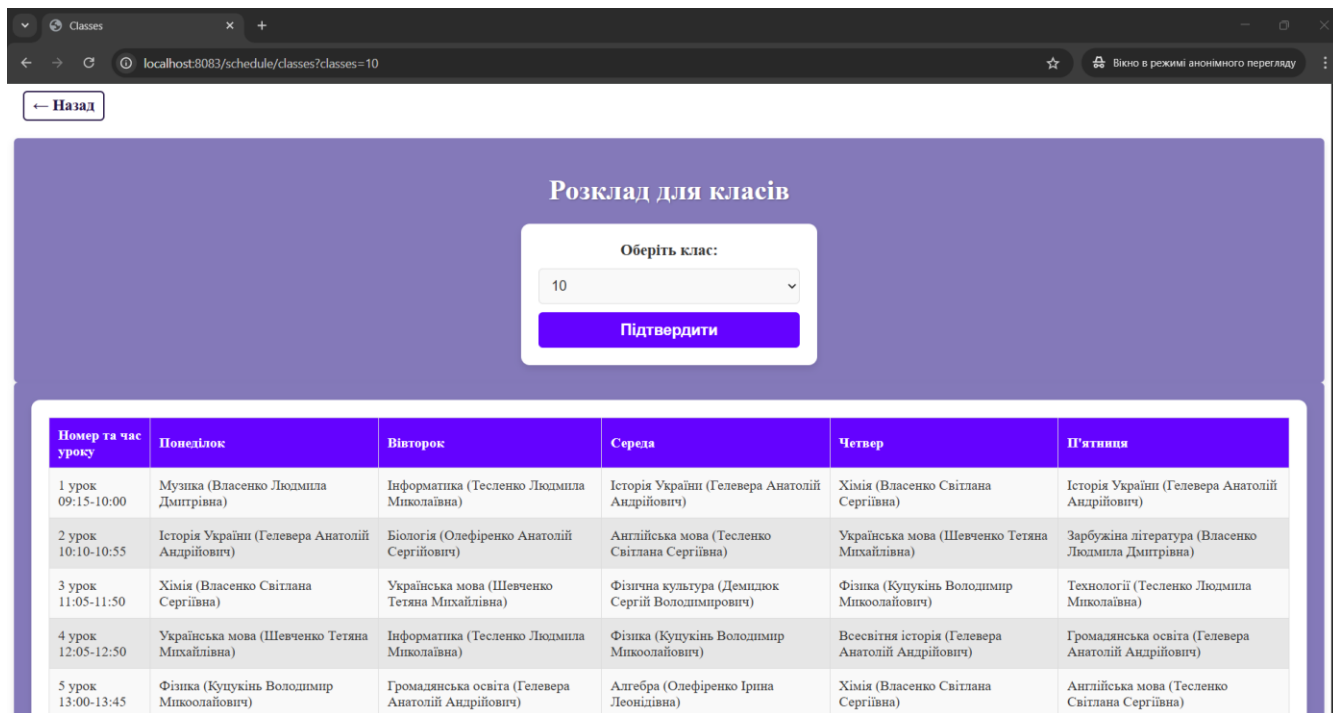


Рис. 4.5 Відображення розкладу класу

4.1.4 Перегляд розкладу викладачів

Для перегляду розкладу викладачів у системі реалізовано окрему сторінку, зображену на рисунку 4.6, на якій користувач може обрати необхідного викладача зі списку.

Після вибору викладача система формує та відображає його індивідуальний розклад занять. Це дозволяє швидко отримати інформацію про всі заняття, які проводить конкретний викладач.

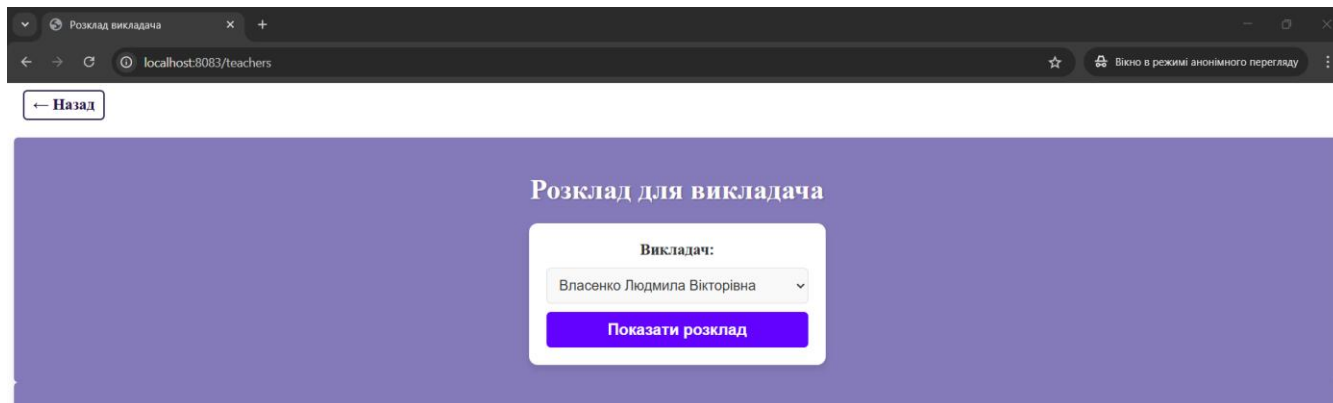


Рис. 4.6 Сторінка вибору викладача

На рисунку 4.7 показано сторінку перегляду розкладу викладача після його вибору у системі. У верхній частині сторінки розташована форма, де користувач може обрати потрібного викладача зі списку та відкрити його розклад. Це дає змогу швидко знаходити необхідну інформацію та переглядати розклад конкретного викладача без переходу між різними сторінками.

Нижче відображається таблиця з розкладом занять. У таблиці міститься інформація про номер уроку, час проведення занять, дні тижня, предмети та класи, у яких викладач проводить уроки. Дані впорядковані за днями тижня та часом уроків, тому користувач може легко переглядати зайнятість викладача протягом навчального тижня.

Такий формат подання інформації робить розклад більш зрозумілим та зручним для використання. Користувач може швидко знайти потрібний день або урок, перевірити наявність занять та переглянути навантаження викладача. Це спрощує роботу з розкладом і забезпечує швидкий доступ до актуальної інформації про навчальний процес.

Розклад для викладача

Викладач:

Куцукінь Володимир Миколайович

Показати розклад

Номер та час уроку	Понеділок	Вівторок	Середа	Четвер	П'ятниця
1 урок 09:15-10:00	Фізика (11 клас)		Фізика (11 клас)	Фізика (7 клас)	
2 урок 10:10-10:55	Фізика (9 клас)		Фізика (8 клас)		
3 урок 11:05-11:50	Фізика (7 клас)		Фізика (9 клас)	Фізика (10 клас)	
4 урок 12:05-12:50	Фізика (8 клас)		Фізика (10 клас)	Фізика (11 клас)	
5 урок 13:00-13:45	Фізика (10 клас)		Алгебра (11 клас)	Фізика (9 клас)	

Рис. 4.7 Відображення розкладу викладача

4.1.5. Адміністративна панель

Для користувачів із роллю адміністратора доступна адміністративна панель, яка показана на рисунку 4.8. Саме через цей розділ виконується основне керування системою та всіма даними, пов'язаними з навчальним розкладом. Адміністратор може створювати нові записи розкладу, редагувати вже наявні заняття та видаляти непотрібні записи. Це дає змогу швидко оновлювати інформацію та підтримувати актуальність розкладу для всіх користувачів системи.

На рисунку видно, що адміністративна панель поділена на окремі блоки для роботи з різними даними. Адміністратор має доступ до керування викладачами, предметами, класами, часом уроків, днями тижня та зв'язками між викладачами й предметами. Для кожного розділу передбачено окремі кнопки редагування, що спрощує роботу з системою та робить керування даними більш зручним.

Такий поділ функцій дозволяє впорядкувати роботу адміністратора та швидко знаходити потрібний розділ для внесення змін. Усі зміни, внесені через адміністративну панель, автоматично зберігаються в системі та відображаються користувачам під час перегляду розкладу.

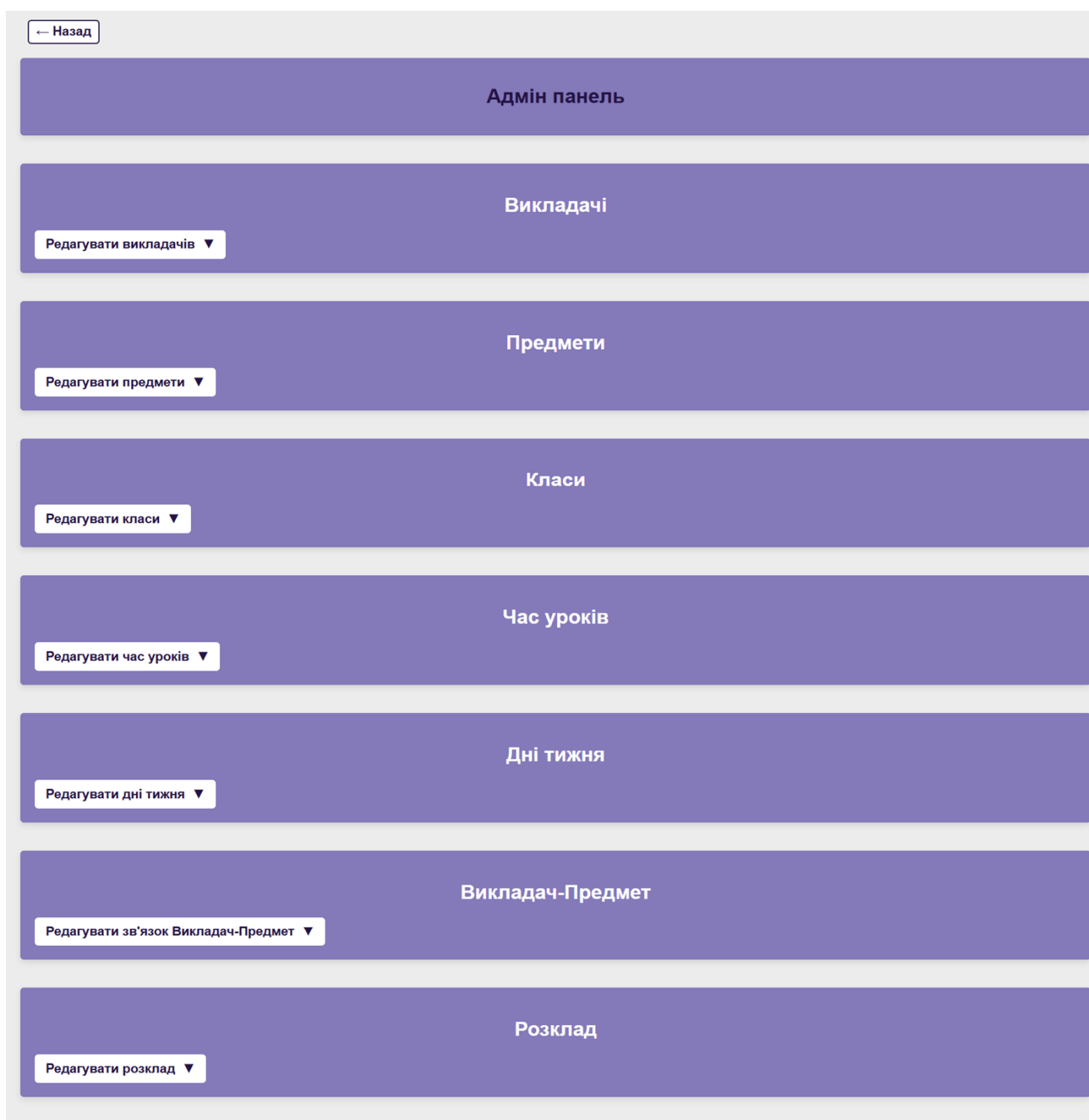


Рис. 4.8 Адміністративна панель системи

4.1.6 Додавання нового запису розкладу

Для створення нового запису адміністратор заповнює спеціальну форму, зображену на рисунку 4.9, у якій вказуються клас, предмет, викладач, день тижня та час проведення заняття.

Після підтвердження дані надсилаються на сервер для перевірки та збереження у базі даних.

Рис. 4.9 Форма створення запису розкладу

4.1.7 Перевірка конфліктів у розкладі

Система підтримує перевірку конфліктів у розкладі, щоб уникати накладання занять. Якщо адміністратор намагається додати урок для викладача або класу на час, який уже зайнятий, система автоматично показує повідомлення про помилку та не дозволяє зберегти такий запис. Це допомагає уникнути помилок у розкладі та підтримує правильність усіх даних.

На рисунку 4.10 показано приклад роботи цієї перевірки. У верхній частині сторінки відображається повідомлення «Викладач уже зайнятий у цей час!», яке з'явилося після спроби створити конфліктний запис. Нижче видно форму для редагування розкладу, де адміністратор може вибрати клас, викладача, день тижня та час уроку. Також на сторінці є таблиця з уже створеним розкладом, з яким система порівнює нові дані.

Такий механізм дозволяє швидко знаходити проблеми в розкладі, не допускати дублювання занять та спрощує роботу адміністратора під час редагування навчального процесу.

Викладач уже зайнятий у цей час!

Розклад

Редагувати розклад ▲

Показати/Приховати розклад ▲

Номер	Клас	Викладач - Предмет	День	Час уроку	Дії
1	6	Гелевера Анатолій Андрійович — Громадянська освіта	Понеділок	09:15-10:00	Редагувати Зберегти Видалити

Рис. 4.10 Повідомлення про конфлікт у розкладі

4.1.8 Редагування та видалення записів

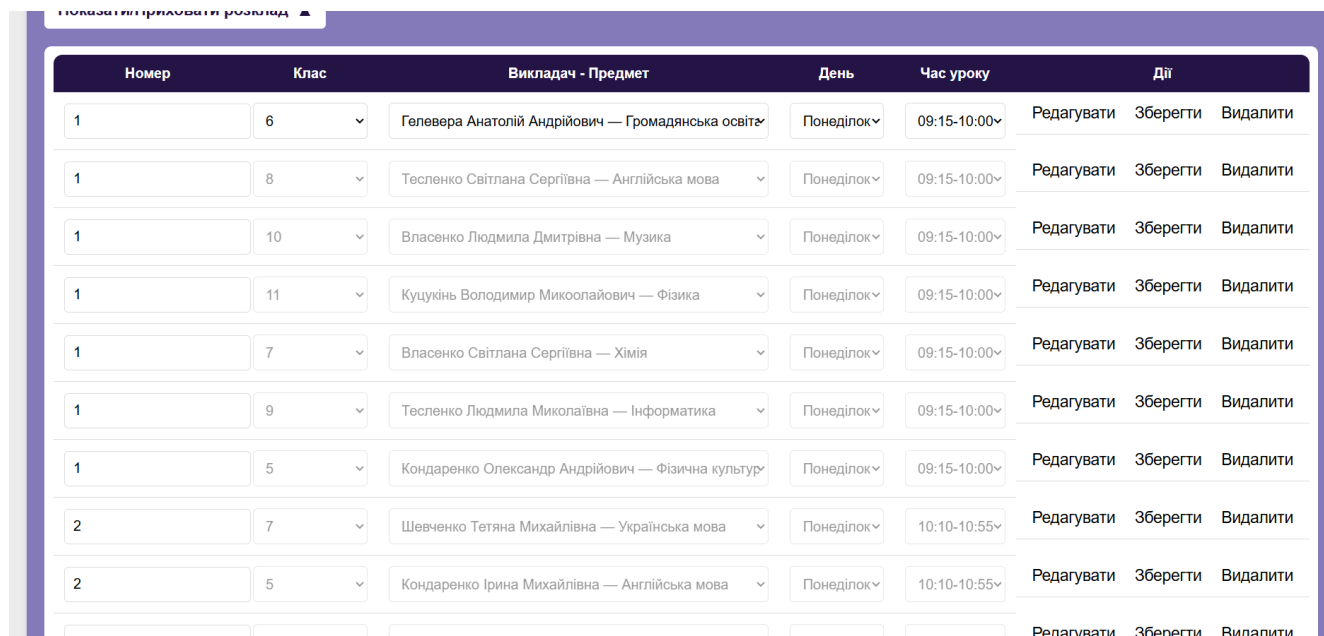
На рисунку 4.11 показано процес редагування та видалення записів розкладу адміністратором системи. Отримані результати підтверджують, що реалізований функціонал працює коректно та забезпечує зручне керування даними розкладу. Адміністратор може вносити зміни до існуючих записів, змінювати інформацію про заняття, час проведення або аудиторію, а також видаляти непотрібні записи без порушення роботи системи.

показати/приховати розклад ▲

Номер	Клас	Викладач - Предмет	День	Час уроку	Дії
1	6	Гелевера Анатолій Андрійович — Громадянська освіта	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	8	Тесленко Світлана Сергіївна — Англійська мова	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	10	Власенко Людмила Дмитрівна — Музика	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	11	Куцукінь Володимир Миколайович — Фізика	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	7	Власенко Світлана Сергіївна — Хімія	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	9	Тесленко Людмила Миколаївна — Інформатика	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	5	Кондаренко Олександр Андрійович — Фізична культура	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
2	7	Шевченко Тетяна Михайлівна — Українська мова	Понеділок	10:10-10:55	Редагувати Зберегти Видалити
2	5	Кондаренко Ірина Михайлівна — Англійська мова	Понеділок	10:10-10:55	Редагувати Зберегти Видалити
2	9	Куцукінь Володимир Миколайович — Фізика	Понеділок	10:10-10:55	Редагувати Зберегти Видалити

Рис. 4.11 Редагування запису розкладу

Після натискання кнопки «Редагувати» адміністратор може змінити необхідні параметри заняття, зокрема викладача або предмет. Приклад зміни викладача у розкладі наведено на рисунку 4.12.



Номер	Клас	Викладач - Предмет	День	Час уроку	Дії
1	6	Гелевера Анатолій Андрійович — Громадянська освіта	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	8	Тесленко Світлана Сергіївна — Англійська мова	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	10	Власенко Людмила Дмитрівна — Музика	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	11	Куцукін Володимир Миколайович — Фізика	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	7	Власенко Світлана Сергіївна — Хімія	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	9	Тесленко Людмила Миколаївна — Інформатика	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
1	5	Кондаренко Олександр Андрійович — Фізична культура	Понеділок	09:15-10:00	Редагувати Зберегти Видалити
2	7	Шевченко Тетяна Михайлівна — Українська мова	Понеділок	10:10-10:55	Редагувати Зберегти Видалити
2	5	Кондаренко Ірина Михайлівна — Англійська мова	Понеділок	10:10-10:55	Редагувати Зберегти Видалити
2	6	Куцукін Володимир Миколайович — Фізика	Понеділок	10:10-10:55	Редагувати Зберегти Видалити

Рис. 4.12 Редагування викладача у розкладі

На рисунку 4.13 показано результат успішного збереження зміненого викладача у розкладі. Після редагування запису та натискання кнопки «Зберегти» система оновлює інформацію, а новий викладач відображається у відповідному полі таблиці. Це підтверджує коректну роботу функції редагування розкладу та збереження змін у базі даних.

Після виконання змін інформація автоматично оновлюється в системі та одразу стає доступною для всіх користувачів. Це забезпечує актуальність розкладу та дозволяє уникнути відображення застарілих даних. Під час тестування було підтверджено, що механізми оновлення працюють стабільно, а зміни коректно зберігаються в базі даних.

Показати/Приховати розклад ▲

Номер	Клас	Викладач - Предмет	День	Час уроку	Дії		
1	6	Олефіренко Ірина Леонідівна — Алгебра	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	8	Тесленко Світлана Сергіївна — Англійська мова	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	10	Власенко Людмила Дмитрівна — Музика	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	11	Куцукін Володимир Миколайович — Фізика	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	7	Власенко Світлана Сергіївна — Хімія	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	9	Тесленко Людмила Миколаївна — Інформатика	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	5	Кондаренко Олександр Андрійович — Фізична культура	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
2	7	Шевченко Тетяна Михайлівна — Українська мова	Понеділок	10:10-10:55	Редагувати	Зберегти	Видалити
2	5	Кондаренко Ірина Михайлівна — Англійська мова	Понеділок	10:10-10:55	Редагувати	Зберегти	Видалити

Рис. 4.13 Збереження зміненого запису розкладу

Для перевірки правильності збереження змін було відкрито сторінку розкладу викладача, якому було призначено новий урок. На рисунку 4.14 видно, що після редагування розкладу нове заняття успішно з’явилося у таблиці розкладу викладача. Це підтверджує, що зміни були коректно збережені у системі та відображаються у відповідному розкладі.

Розклад для викладача

Викладач:

Олефіренко Ірина Леонідівна

Показати розклад

Номер та час уроку	Понеділок	Вівторок	Середа	Четвер	П'ятниця
1 урок 09:15-10:00	Алгебра (6 клас)	Математика (5 клас)	Математика (6 клас)	Математика (6 клас)	Геометрія (8 клас)
2 урок 10:10-10:55		Геометрія (8 клас)	Математика (5 клас)	Алгебра (9 клас)	Математика (6 клас)
3 урок 11:05-11:50	Математика (6 клас)	Математика (6 клас)	Геометрія (7 клас)	Математика (5 клас)	Математика (5 клас)
4 урок 12:05-12:50	Математика (5 клас)	Алгебра (7 клас)	Алгебра (8 клас)	Алгебра (7 клас)	Алгебра (7 клас)
5 урок 13:00-13:45	Алгебра (8 клас)	Геометрія (9 клас)	Алгебра (10 клас)	Геометрія (7 клас)	Геометрія (9 клас)

Рис. 4.14 Відображення нового уроку у розкладі викладача

Отже, під час демонстрації того, як все працює, можна побачити, що всі головні частини цієї системи для керування розкладом у школі працюють як слід. Ця система дозволяє дивитись розклад занять просто і швидко. Переглядати його можуть різні типи користувачів: учні, вчителі та адміністратор. Також система має захист — лише люди з правами доступу можуть увійти у систему під своїм ім'ям і паролем.

Якщо користувач є адміністратором, то він може легко і швидко змінювати все, що стосується уроків. Можна додавати нові уроки, змінювати їх або видаляти з розкладу. Також у системі легко дивитися дані про всі заняття, вчителів та класи.

Сторінки системи зроблені так, щоб будь-хто зміг легко знайти потрібне. Увесь інтерфейс дуже простий, його зручно і зрозуміло використовувати.

4.2 Тестування авторизації та безпеки

У таблиці 4.1 наведено процес тестування. Було перевірено коректність роботи механізмів автентифікації та авторизації користувачів. Основною метою тестування було підтвердити, що доступ до функцій системи надається відповідно до ролі користувача, а також перевірити правильність обробки помилок входу. Під час тестування також увагу приділено перевірці обмеження доступу до адміністративних функцій для користувачів без відповідних прав. Також було перевірено стабільність роботи системи після успішної авторизації та коректність перенаправлення користувачів на доступні їм сторінки. Отримані результати дали змогу підтвердити правильну роботу механізмів безпеки та захисту даних у вебсистемі.

Було протестовано:

- вхід у систему з правильними даними;
- вхід із неправильним логіном або паролем;
- обмеження доступу до адміністративної панелі;
- доступ до функцій редагування розкладу лише для адміністратора.

Таблиця 4.1

Тестування авторизації та безпеки

№	Назва тесту	Вхідні дані	Очікуваний результат	Фактичний результат
1	Успішний вхід адміністратора	Правильний логін і пароль admin	Відкриття головної сторінки та адміністративної панелі	Вхід виконано успішно
2	Успішний вхід учня	Правильний логін student1 і пароль studentpass1	Відкриття головної сторінки та сторінки розкладу для учня	Вхід виконано успішно
3	Успішний вхід викладача	Правильний логін teacher1 і пароль teacherpass1	Відкриття головної сторінки та сторінки розкладу для викладача	Вхід виконано успішно
4	Неправильний пароль	Коректний логін, неправильний пароль	Повідомлення про помилку входу	Повідомлення відображається
5	Порожні поля входу	Логін і пароль порожні	Повідомлення про необхідність заповнення	Валідація працює коректно
6	Спроба доступу USER до адмін-панелі	Користувач USER відкриває /admin	Доступ заборонено	Доступ заблоковано
7	Доступ ADMIN до редагування	ADMIN відкриває сторінку редагування	Сторінка доступна	Доступ надано

Тест показав, що система працює правильно. Коли користувачі входять у застосунок, то система знає, хто вони і дозволяє робити лише те, що їм можна.

Якщо у користувача є певна роль — вона бачить ті частини, що їй треба. Адміністратор може змінювати розклад і дивитись панель для цього. Учень та викладач бачать тільки сторінки розкладу.

Під час тесту ми перевіряли, як система реагує на помилки. Якщо пароль не співпадає з паролем у базі даних чи поля пусті, користувач одразу бачить повідомлення про необхідність це виправити.

З отриманих результатів видно, що все зроблено правильно. Механізми безпеки працюють коректно. Є чітке розмежування прав доступу, і система захищає свою роботу від тих, у кого не повинно бути доступу. Розроблена система авторизації дає змогу користувачам працювати безпечно. Усі вимоги до захисту даних і керування доступом виконано у повній мірі. Система відповідає всім запитам щодо безпеки і дотримується встановлених правил.

4.3 Тестування функцій керування розкладом

Під час тестування було перевірено коректність роботи функцій створення, редагування та видалення записів розкладу. Також перевірялася правильність збереження даних у базі даних. Це наведено у таблиці 4.2.

Тест показав, що система правильно виконує всі важливі дії для керування розкладом. Також було перевірено, чи добре працює вхід у систему, чи немає збоїв та чи все коректно відображається у розкладі для різних користувачів. Система стабільно працює під час виконання основних сценаріїв використання.

Таблиця 4.2

Тестування функцій керування розкладом

№	Назва тесту	Вхідні дані	Очікуваний результат	Фактичний результат
1	Додавання нового заняття	Коректні дані заняття	Новий запис додається до розкладу	Запис успішно створено

Продовження таблиці 4.2

Тестування функцій керування розкладом

2	Редагування заняття	Зміна часу заняття	Дані оновлюються	Дані оновлено
3	Видалення заняття	Видалення запису	Запис зникає з розкладу	Запис успішно видалено
4	Перегляд розкладу	Відкриття сторінки розкладу	Відображення всіх занять	Дані відображаються коректно
5	Збереження у БД	Створення нового запису	Дані зберігаються у базі	Дані успішно збережено

4.4 Тестування перевірки даних та обробки помилок

Для забезпечення стабільної роботи системи було проведено тестування механізмів перевірки введених даних та обробки помилок. Особливу увагу приділено перевірці конфліктів у розкладі та роботі валідації форм.

У ході тестування перевірялося:

- заповнення обов'язкових полів;
- обробка некоректних даних;
- перевірка конфліктів у розкладі;
- відображення повідомлень про помилки.

Процес та результати тестування наведено у таблиці 4.3.

Таблиця 4.3

Тестування перевірки даних та обробки помилок

№	Назва тесту	Вхідні дані	Очікуваний результат	Фактичний результат
1	Порожні поля форми	Незаповнені поля	Повідомлення про помилку	Повідомлення відображається
2	Конфлікт розкладу для викладача	Два заняття в один час	Помилка конфлікту	Конфлікт виявлено
3	Конфлікт розкладу для класу	Два уроки для одного класу	Помилка конфлікту	Конфлікт виявлено
4	Некоректні дані часу	Неправильний час уроку	Валідація блокує збереження	Помилка відображається
5	Повторне надсилення некоректної форми	Незаповнені дані	Запис не створюється	Дані не збережено

Результати тесту показали, що система добре перевіряє всі дані, які вводить користувач, та правильно надсилає помилки, якщо щось не так при роботі з розкладом. Усі поля, які мають бути заповнені, проходять перевірку. Якщо клієнт вводить не ті або не всі дані, вона бачить просте і чітке повідомлення про помилку. Таку помилку не можна проігнорувати, і запис зберегти теж не можна.

Коли проводилося тестування, чи знаходить система конфлікти, було видно, що вона правильно знаходить співпадіння уроків та занять для вчителів та класів. Якщо дві події стоять на один час, тоді програма не дає цей розклад зберегти. Вона відразу каже користувачу, що є проблема.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи розроблено вебсистему для управління шкільним розкладом, яка забезпечує автоматизацію процесу створення, редагування та перегляду навчального розкладу для адміністратора, викладачів і учнів.

У процесі виконання роботи було отримано такі результати:

1. На основі аналізу предметної області управління шкільним розкладом та дослідження існуючих програмних рішень, зокрема WebUntis, aSc Timetables, Google Classroom і TimetableMaster, визначено основні переваги та недоліки наявних систем. У результаті аналізу сформовано функціональні та нефункціональні вимоги до вебсистеми управління шкільним розкладом.

2. У результаті дослідження сучасних підходів до розробки вебзастосунків обрано технологічний стек Java, Spring Boot, Spring Security, Thymeleaf, HTML, CSS та MySQL. Використання зазначених технологій дозволило забезпечити безпечну обробку даних, розподіл прав доступу та взаємодію між серверною частиною і користувацьким інтерфейсом.

3. На основі визначених вимог спроектовано архітектуру вебсистеми, структуру бази даних та основні модулі системи. Розроблена структура бази даних забезпечує збереження інформації про користувачів, класи, викладачів, предмети та розклад занять.

4. На основі сформованих вимог реалізовано вебзастосунок для управління шкільним розкладом із підтримкою ролей користувачів та розподілом прав доступу. Для адміністратора реалізовано можливість створення, редагування та видалення даних розкладу, а для викладачів і учнів — перегляд навчального розкладу.

5. У результаті розробки впроваджено механізми автентифікації та авторизації користувачів, перевірки конфліктів у розкладі, валідації введених

даних та обробки помилок. Реалізована перевірка конфліктів дозволяє уникати накладання занять для викладачів і класів в один і той самий час.

6. У процесі тестування вебсистеми виконано 17 тест-кейсів для перевірки механізмів авторизації та безпеки, функцій керування розкладом, перевірки даних і обробки помилок. Результати тестування підтвердили коректність роботи системи, стабільність обробки даних та відповідність вебзастосунок поставленим вимогам.

Отримані результати підтверджують практичну ефективність розробленої вебсистеми та можливість її використання для автоматизації процесу управління шкільним розкладом у навчальних закладах.

Робота пройшла апробацію. За результатами дослідження опубліковано такі тези доповідей:

1. Коврига Б.О., Замрій І.В. Розробка вебсистеми управління шкільним розкладом. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.471-478.

2. Коврига Б.О., Замрій І.В. Гарантування безпеки інформації у вебсистемі управління шкільних розкладом. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2026. С.237-240.

ПЕРЕЛІК ПОСИЛАНЬ

1. Java Documentation - Get Started. *Oracle Help Center*. URL: <https://docs.oracle.com/en/java/> (date of access: 23.04.2026).
2. Spring Boot Reference Documentation. *Spring*. URL: <https://docs.spring.io/spring-boot/docs/current/reference/html/> (date of access: 23.04.2026).
3. MySQL 8.0 Reference Manual. *MySQL*. URL: <https://dev.mysql.com/doc/refman/8.0/en/> (date of access: 23.04.2026).
4. Docker guides. *Docker*. URL: <https://docs.docker.com/guides/> (date of access: 23.04.2026).
5. Thymeleaf Documentation. *Thymeleaf*. URL: <https://www.thymeleaf.org/documentation.html> (date of access: 23.04.2026).
6. The Leading IDE for Professional Java and Kotlin Development. *JetBrains*. URL: <https://www.jetbrains.com/idea/> (date of access: 23.04.2026).
7. JUnit User Guide. *JUnit*. URL: <https://junit.org/junit5/docs/current/user-guide/> (date of access: 23.04.2026).
8. Spilca L. Spring Start Here: Learn What You Need and Learn It Well. Manning Publications Co. LLC, 2021. 336 p..
9. Heckler M. Spring Boot : Up and Running: Building Cloud Native Java and Kotlin Applications. O'Reilly Media, Incorporated, 2021. 300 p.
10. Poulton N. Docker Deep Dive: 2023 Edition. 2023. 307 p.
11. Аналіз методів автоматизованого формування навчального розкладу. *Репозитарій Вінницького Національного Технічного Університету*. URL: <https://ir.lib.vntu.edu.ua/handle/123456789/39697> (дата звернення: 23.04.2026).
12. Abdullah D.B., Abood I.N., Hamdi R.A. Real Time System Scheduling Approach : Survey. *AL-Rafidain Journal of Computer Sciences and Mathematics*. 2023. Vol. 17, no. 1. P. 43–51. URL: <https://doi.org/10.33899/csmj.2023.179512> (date of access: 23.04.2026).

13. Keil W., Bais T., Tijms A. *Definitive Guide to Security in Jakarta EE: Securing Java-Based Enterprise Applications with Jakarta Security, Authorization, Authentication and More*. Apress L. P., 2022.
14. Чілікіна Т. Розробка сайту для управління своїм розкладом та завданнями. *Information Technology and Society*. 2025. № 1 (16). С. 270–275. URL: <https://doi.org/10.32689/maup.it.2025.1.35> (дата звернення: 23.04.2026).
15. Самойленко О. Інтернет-технології в архітектурі програмного забезпечення систем управління навчальним процесом. *Наука і техніка сьогодні*. 2025. № 11(52). URL: [https://doi.org/10.52058/2786-6025-2025-11\(52\)-1377-1392](https://doi.org/10.52058/2786-6025-2025-11(52)-1377-1392) (дата звернення: 23.04.2026)
16. Федорченко Є.М. Генетичний метод розв'язання задачі складання розкладу навчальних занять. *Реєстрація, зберігання і обробка даних*. 2024. Т. 26, № 1. С. 9–23. URL: <https://doi.org/10.35681/1560-9189.2024.26.1.308332> (дата звернення: 23.04.2026).
17. Kutty Mammi H., Ying Ying L. Timetable Scheduling System using Genetic Algorithm for School of Computing (tsuGA). *International Journal of Innovative Computing*. 2021. Vol. 11, no. 2. P. 67–72. URL: <https://doi.org/10.11113/ijic.v11n2.342> (date of access: 23.04.2026).
18. Dedhia N. Secure Web Authentication Using JWT and Role-Based Access Control. *International journal of scientific research in engineering and management*. 2026. Vol. 10, no. 04. P. 1–9. URL: <https://doi.org/10.55041/ijsrem60274> (date of access: 23.04.2026).
19. Patil K. Authentication and Authorization in Web Applications. *Journal of Engineering and Applied Sciences Technology*. 2023. P. 1–2. URL: [https://doi.org/10.47363/jeast/2023\(5\)186](https://doi.org/10.47363/jeast/2023(5)186) (date of access: 23.04.2026).
20. Kosolap A. I., Dubovik T. M. Optimization of timetable at the university. *Radio Electronics, Computer Science, Control*. 2021. no. 3. P. 175–183. URL: <https://doi.org/10.15588/1607-3274-2021-3-15> (date of access: 23.04.2026).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Розробка web-системи управління шкільним розкладом для викладачів, учнів та адміністратора

Виконала студентка 4 курсу
групи ПД-41
Коврига Богдана Олександрівна
Керівник роботи
д.т.н., доцент, завідувач кафедри ІІЗ Замрій Ірина Вікторівна
Київ – 2026

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: автоматизація процесу створення та управління шкільним розкладом із забезпеченням зручного доступу до інформації для різних категорій користувачів.

Об'єкт дослідження: процес формування та управління шкільним розкладом у навчальному закладі.

Предмет дослідження: методи та програмні засоби автоматизації управління розкладом на основі веб-технологій.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати предметну область управління шкільним розкладом та існуючі програмні рішення.
2. Визначити недоліки існуючих засобів управління шкільним розкладом.
3. Визначити вимоги до вебсистеми управління шкільним розкладом для викладачів, учнів та адміністратора.
4. Спроектувати структуру вебсистеми та бази даних.
5. Реалізувати вебзастосунок для управління шкільним розкладом з використанням сучасних технологій.
6. Реалізувати механізми автентифікації, авторизації та перевірки конфліктів у розкладі.
7. Провести тестування вебсистеми та аналіз результатів її роботи.
8. Підготувати звітну документацію та демонстраційні матеріали.

3

АНАЛІЗ АНАЛОГІВ

Показник	WebUntis	aSc Timetables	Google Classroom	TimetableMaster	Schedify (розроблювана система)
Веб-доступ	+	-	+	+	+
Ролі користувачів	+	+	+	+	+
Автоматизація розкладу	+	+	-	-	+
Управління розкладом	+	+	-	+	+
Редагування розкладу в реальному часі	-	+	-	-	+
Система авторизації та автентифікації	+	-	+	-	+
Захист даних користувачів	+	+	+	+	+
Адаптивність для мобільних пристроїв	+	-	+	+	+
Автоматична перевірка конфліктів у розкладі	-	+	-	-	+

4

ВИМОГИ ДО ЗАСТОСУНКУ

ФУНКЦІОНАЛЬНІ ВИМОГИ

1. Система повинна забезпечувати реєстрацію та авторизацію користувачів із розмежуванням прав доступу відповідно до ролей (учень, викладач, адміністратор).
2. Користувачі повинні мати можливість переглядати розклад занять за різними параметрами (клас, викладач) у зручному табличному вигляді.
3. Адміністратор повинен мати можливість створювати, редагувати та видаляти розклад, а також здійснювати перевірку конфліктів (перетин викладачів або занять).
4. Система повинна забезпечувати управління даними, зокрема класами, предметами, викладачами.
5. Система повинна забезпечувати CRUD-операції для класів, викладачів, предметів.
6. Дані повинні зберігатися у базі даних.

НЕФУНКЦІОНАЛЬНІ ВИМОГИ

1. Час відгуку системи під час виконання основних операцій не повинен перевищувати 2 секунд за стандартного навантаження.
2. Система повинна забезпечувати одночасну роботу не менше 50 користувачів без суттєвого зниження продуктивності.
3. У разі виникнення помилок система повинна відображати повідомлення про помилку без втрати введених користувачем даних.
4. Паролі користувачів повинні зберігатися у зашифрованому вигляді з використанням сучасних алгоритмів хешування.
5. Доступ до функціональних можливостей системи повинен обмежуватися відповідно до ролей користувачів.
6. Система повинна забезпечувати захист від несанкціонованого доступу шляхом використання механізмів автентифікації та авторизації.
7. Вебзастосунок повинен коректно працювати у сучасних веббраузерах, зокрема Google Chrome, Mozilla Firefox та Microsoft Edge.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



Java

Spring Boot +
Spring MVC

Thymeleaf



HTML



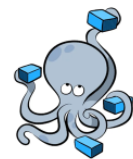
CSS



MySQL



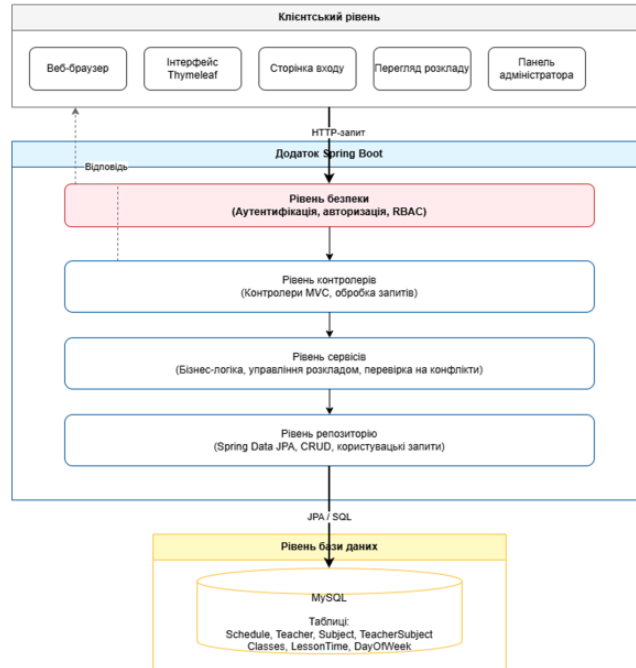
Apache Maven



Docker Compose

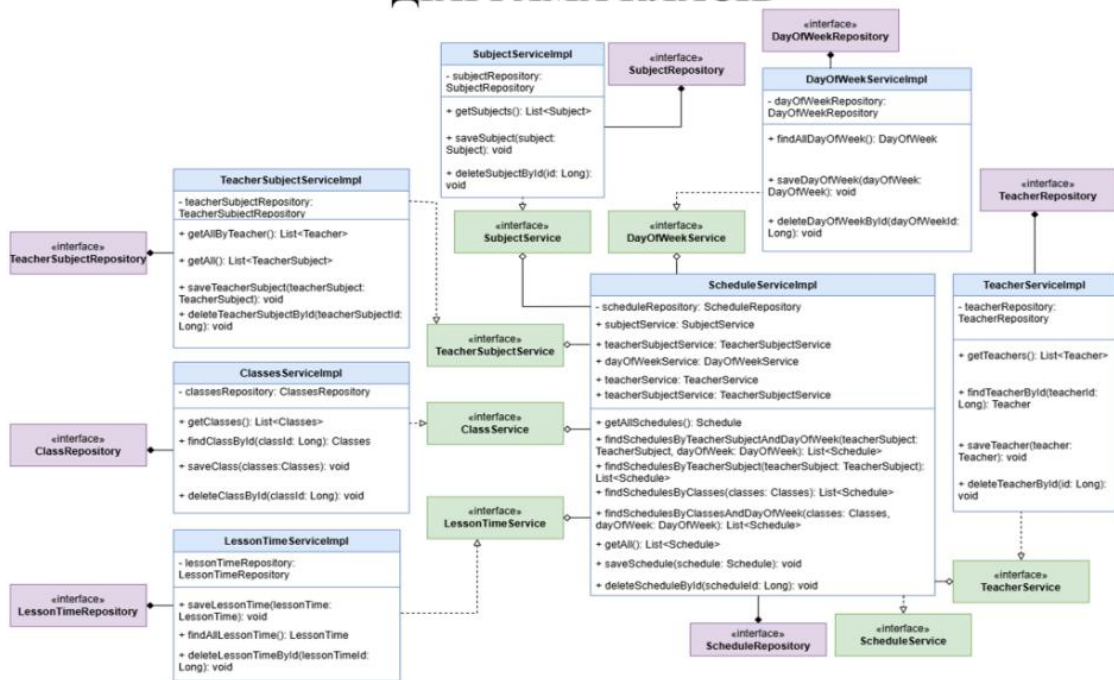
6

ДІАГРАМА АРХІТЕКТУРИ



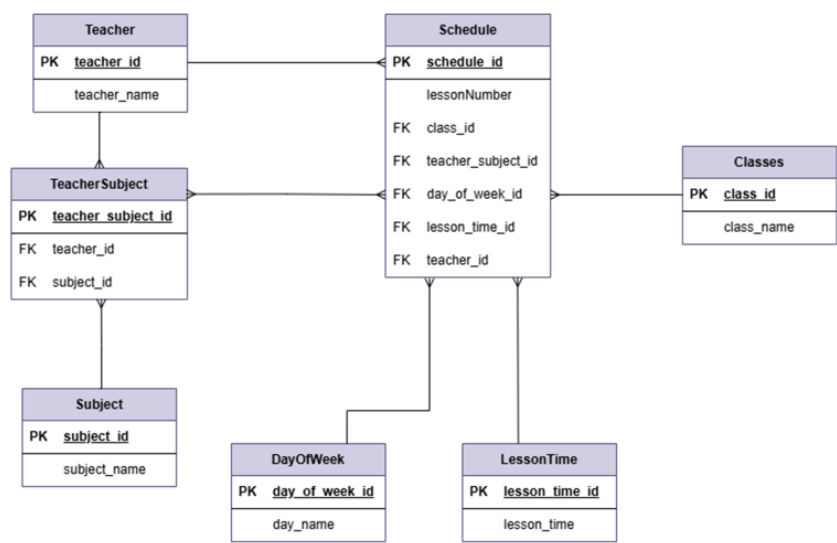
7

ДІАГРАМА КЛАСІВ



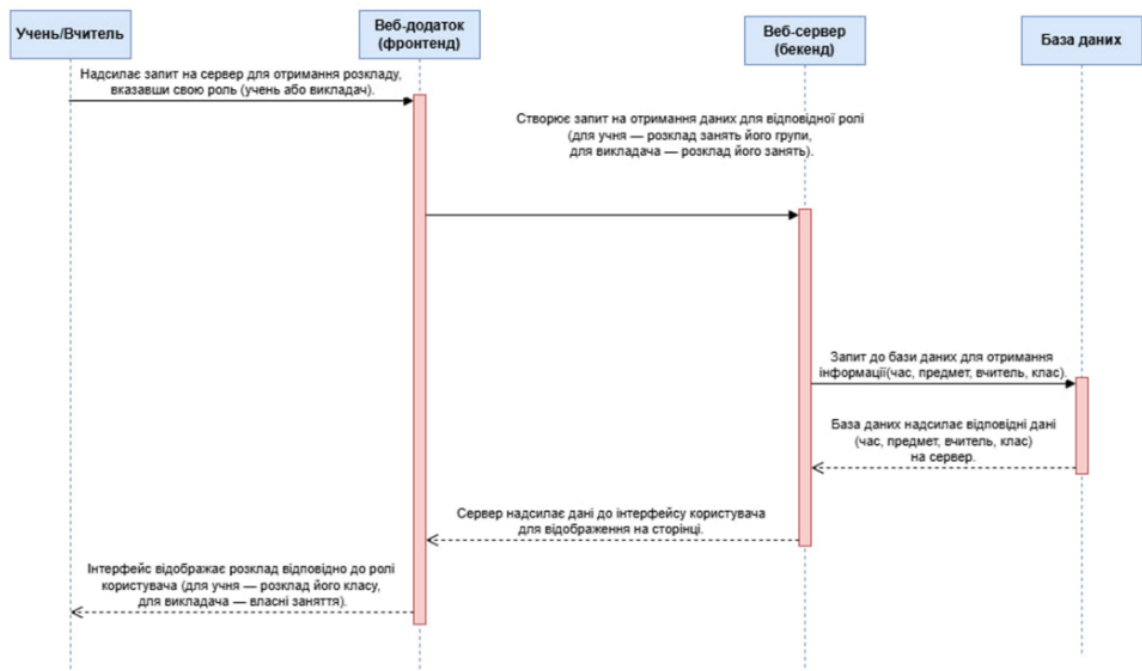
8

МОДЕЛЬ «СУТНІСТЬ-ЗВ'ЯЗОК»



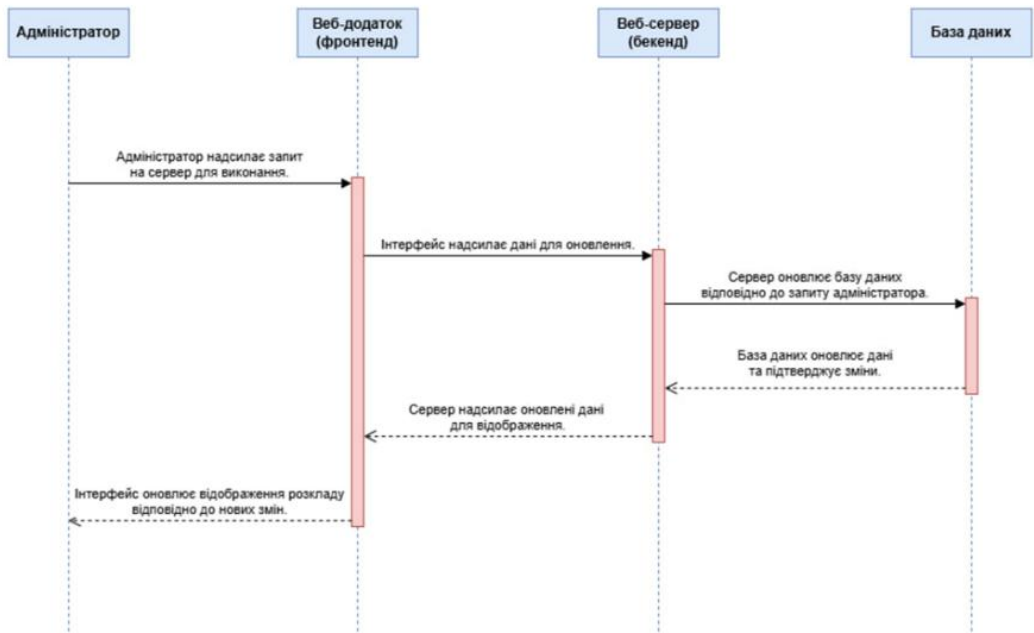
9

ДІАГРАМА ПОСЛІДОВНОСТІ ДЛЯ УЧНЯ КЛАСУ/ВИКЛАДАЧА

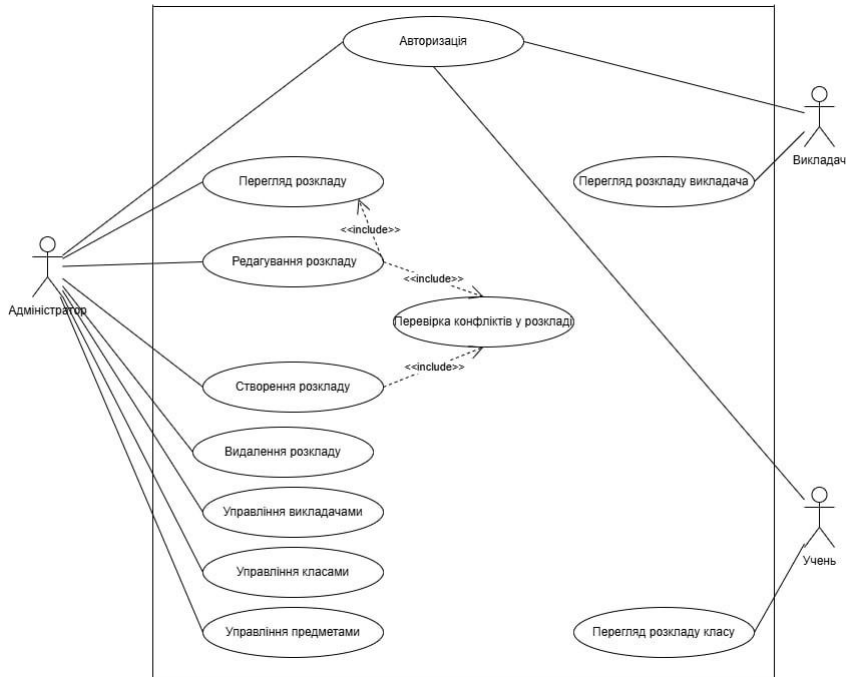


10

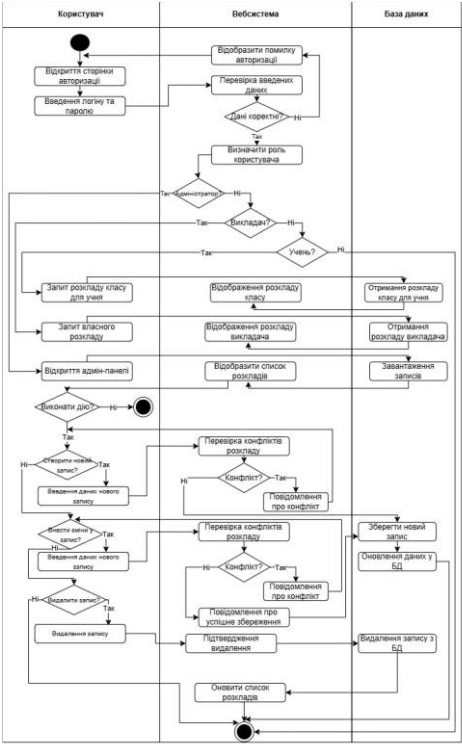
ДІАГРАМА ПОСЛІДОВНОСТІ ДЛЯ АДМІНІСТРАТОРА



ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

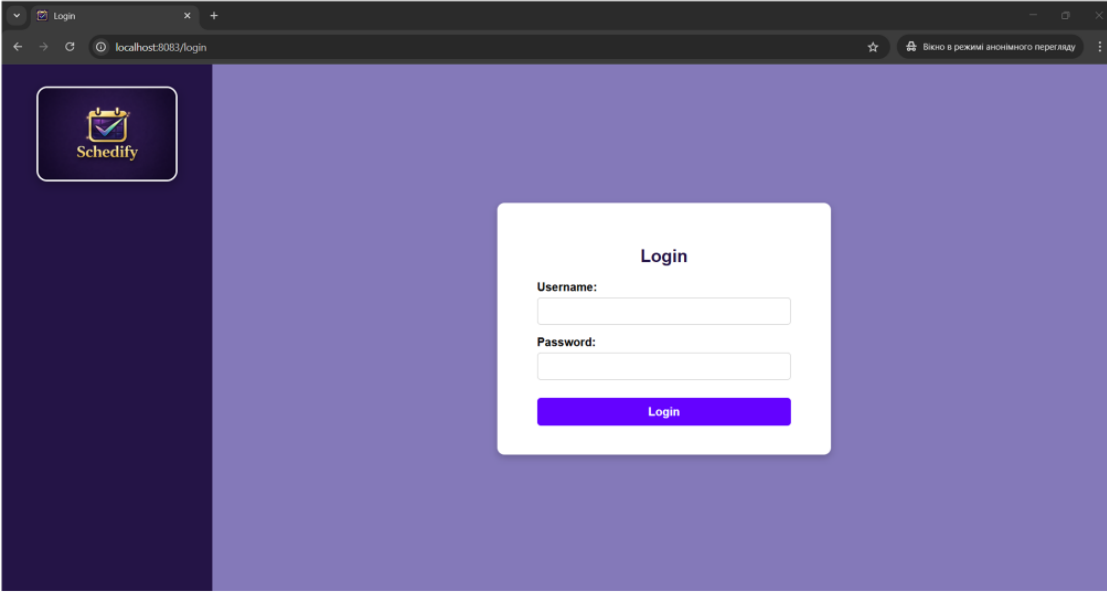


ДІАГРАМА ДІЯЛЬНОСТІ



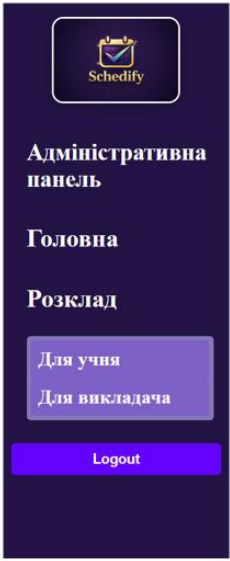
13

ЕКРАННІ ФОРМИ РОБОТИ ЗАСТОСУНКУ



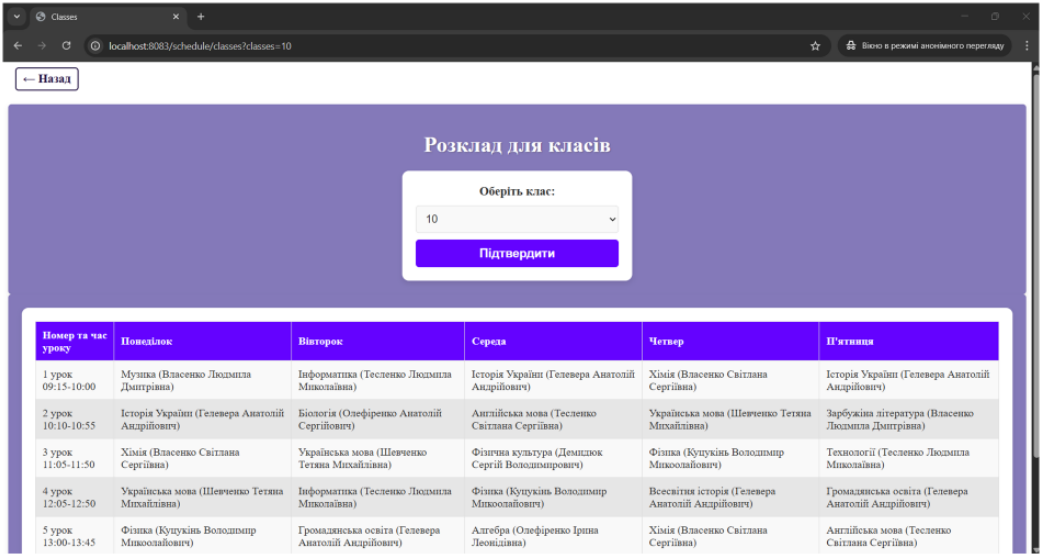
14

ЕКРАННІ ФОРМИ РОБОТИ ЗАСТОСУНКУ



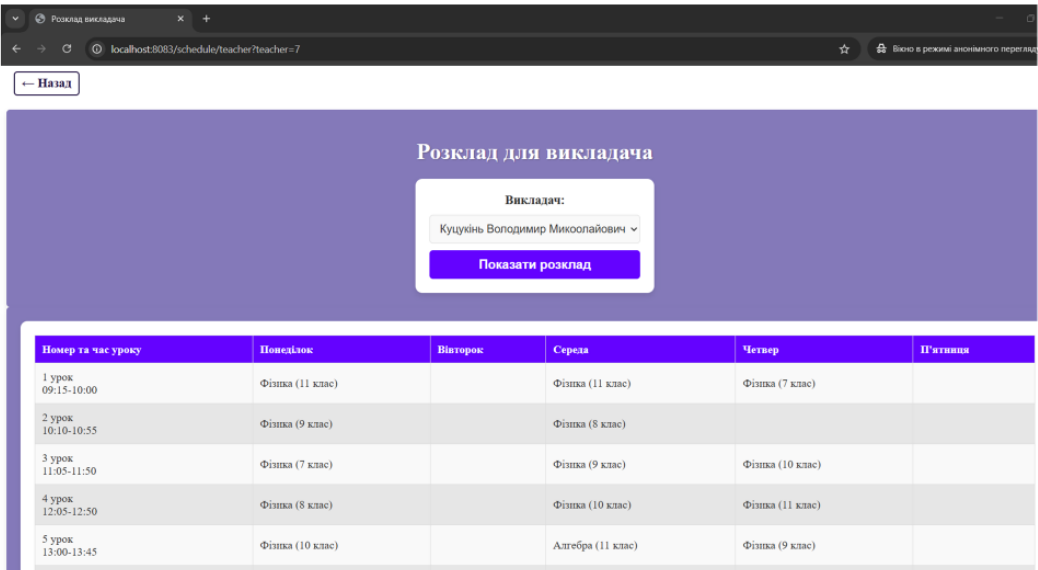
15

ЕКРАННІ ФОРМИ РОБОТИ ЗАСТОСУНКУ



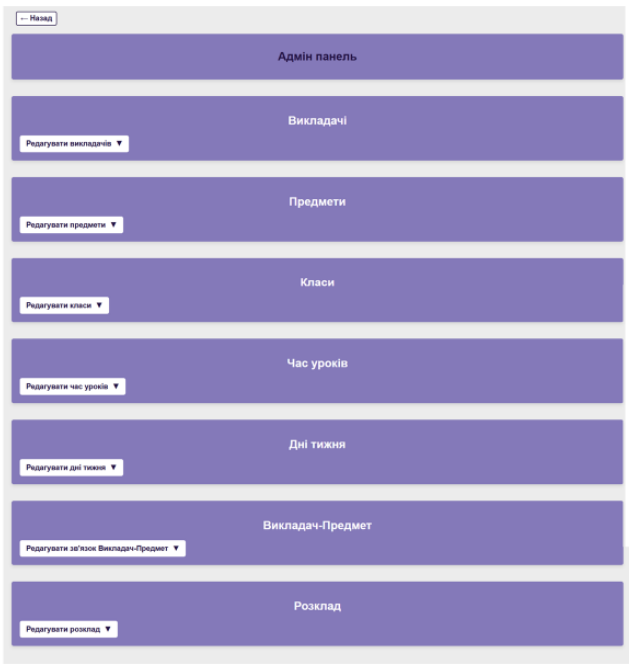
16

ЕКРАННІ ФОРМИ РОБОТИ ЗАСТОСУНКУ



17

ЕКРАННІ ФОРМИ РОБОТИ ЗАСТОСУНКУ



18

ЕКРАННІ ФОРМИ РОБОТИ ЗАСТОСУНКУ

показати/приховати розклад ▲

Номер	Клас	Викладач - Предмет	День	Час уроку	Дії		
1	6	Гелевера Анатолій Андрійович — Громадянська освіта	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	8	Тесленко Світлана Сергіївна — Англійська мова	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	10	Власенко Людмила Дмитрівна — Музика	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	11	Куцукінь Володимир Миколайович — Фізика	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	7	Власенко Світлана Сергіївна — Хімія	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	9	Тесленко Людмила Миколаївна — Інформатика	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
1	5	Кондаренко Олександр Андрійович — Фізична культура	Понеділок	09:15-10:00	Редагувати	Зберегти	Видалити
2	7	Шевченко Тетяна Михайлівна — Українська мова	Понеділок	10:10-10:55	Редагувати	Зберегти	Видалити
2	5	Кондаренко Ірина Михайлівна — Англійська мова	Понеділок	10:10-10:55	Редагувати	Зберегти	Видалити
2	9	Куцукінь Володимир Миколайович — Фізика	Понеділок	10:10-10:55	Редагувати	Зберегти	Видалити

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

- Коврига Б.О., Замрій І.В. Розробка вебсистеми управління шкільним розкладом. Матеріали Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.471-478.
- Коврига Б.О., Замрій І.В. Гарантування безпеки інформації у вебсистемі управління шкільних розкладом. Матеріали Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2026. С.237-240.

ВИСНОВКИ

1. На основі аналізу предметної області та існуючих програмних рішень, зокрема WebUntis, aSc Timetables, Google Classroom і TimetableMaster, визначено основні вимоги до вебсистеми управління шкільним розкладом.
2. У результаті дослідження сучасних підходів до розробки вебзастосунків обрано стек технологій Java, Spring Boot, Spring Security, Thymeleaf та MySQL, а також спроектовано архітектуру вебсистеми та структуру бази даних.
3. На основі сформованих вимог реалізовано вебзастосунок для управління шкільним розкладом із підтримкою ролей користувачів, механізмами автентифікації, авторизації та перевірки конфліктів у розкладі.
4. Під час тестування було виконано 17 тест-кейсів для перевірки механізмів авторизації та безпеки, функцій керування розкладом, валідації даних і обробки помилок. Результати тестування підтвердили коректність роботи вебсистеми та її відповідність поставленим вимогам.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```

1. Головний клас (вхідна точка)

SchoolScheduleApplication

package com.example;

import
org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SchoolScheduleApplication {
    public static void main(String[] args) {

SpringApplication.run(SchoolScheduleApplication.
class, args);
    }
}

2. Backend-частина (контролер, який
відповідає за розклад)
ScheduleController.java

package com.example.controller;

import com.example.model.*;
import com.example.service.*;
import
io.swagger.v3.oas.annotations.security.SecurityReq
uirement;
import
org.springframework.beans.factory.annotation.Auto
wired;
import
org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import
org.springframework.web.bind.annotation.GetMap
ping;
import
org.springframework.web.bind.annotation.Request
Mapping;
import com.example.model.DayOfWeek;

import java.util.*;

@Controller
@RequestMapping("/schedule")
public class ScheduleController {
    @Autowired
    private TeacherController
teacherController;
    @Autowired
    private ClassesController
classesController;
    @Autowired
    private TeacherService teacherService;
    @Autowired
    private TeacherSubjectService
teacherSubjectService;
    @Autowired
    private DayOfWeekService
dayOfWeekService;
    @Autowired
    private ClassService classService;
    @Autowired

```

```

private ScheduleService scheduleService;
@Autowired
private LessonTimeService lessonTimeService;

List<String> checkLessonNumber(List<Schedule> lessons) {
    List<String> result = new ArrayList<>(Arrays.asList(" ", "
", " ", " ", " ", " ", " ", " "));

    for (Schedule schedule : lessons) {
        String subjectName =
schedule.getTeacherSubject().getSubject().getName();
        String teacherName =
schedule.getTeacherSubject().getTeacher().getName();

        String subjectWithTeacher = subjectName + " (" +
teacherName + ")";

        result.set(schedule.getLessonNumber() - 1,
subjectWithTeacher);
    }

    return result;
}

@GetMapping("/teacher")
public String getScheduleByTeacher(Teacher teacher, Model
model) {
    Teacher teacherFromDB =
teacherService.getTeacherById(teacher.getId());
    model.addAttribute("selectedTeacherId", teacher.getId());

    List<TeacherSubject> teacherSubjectList =
teacherSubjectService.findAllByTeacher(teacherFromDB);
    List<DayOfWeek> dayOfWeekList =
dayOfWeekService.findAllDayOfWeek();

    List<String> scheduleMonday = new
ArrayList<>(Collections.nCopies(7, "")); // 7 уроків на день
    List<String> scheduleTuesday = new
ArrayList<>(Collections.nCopies(7, ""));
    List<String> scheduleWednesday = new
ArrayList<>(Collections.nCopies(7, ""));
    List<String> scheduleThursday = new
ArrayList<>(Collections.nCopies(7, ""));
    List<String> scheduleFriday = new
ArrayList<>(Collections.nCopies(7, ""));

    for (TeacherSubject teacherSubject : teacherSubjectList) {
        for (DayOfWeek dayOfWeek : dayOfWeekList) {
            List<Schedule> schedulesForDay =
scheduleService.findSchedulesByTeacherSubjectAndDayOfWeek(teacher
Subject, dayOfWeek);

            for (Schedule schedule : schedulesForDay) {
                int lessonIndex = schedule.getLessonNumber() - 1;
                String lessonName =
schedule.getTeacherSubject().getSubject().getName() +
" (" + schedule.getClasses().getName() + "
клас)";

                if (dayOfWeek.getName().equals("Понеділок")) {
                    scheduleMonday.set(lessonIndex, lessonName);
                } else if
(dayOfWeek.getName().equals("Вівторок")) {
                    scheduleTuesday.set(lessonIndex, lessonName);
                } else if (dayOfWeek.getName().equals("Середа"))
                {

```

```

scheduleWednesday.set(lessonIndex, lessonName);
        } else if
(dayOfWeek.getName().equals("Четвер")) {

scheduleThursday.set(lessonIndex, lessonName);
        } else if
(dayOfWeek.getName().equals("П'ятниця")) {

scheduleFriday.set(lessonIndex, lessonName);
        }
    }
}

List<Schedule> result = new
ArrayList<>();

for (TeacherSubject teacherSubject :
teacherSubjectList) {

result.addAll(scheduleService.findSchedulesByTeacherSubject(teacherSubject));
}

List<LessonTime>
lessonTimeFromDB =
lessonTimeService.findAll();

model.addAttribute("lessonTime",
lessonTimeFromDB);
model.addAttribute("schedule",
result);

model.addAttribute("scheduleMonday",
scheduleMonday);

model.addAttribute("scheduleTuesday",
scheduleTuesday);

model.addAttribute("scheduleWednesday",
scheduleWednesday);

model.addAttribute("scheduleThursday",
scheduleThursday);

model.addAttribute("scheduleFriday",
scheduleFriday);

teacherController.teacher(model);

return "teachers";
}

@GetMapping("/classes")
public String
getScheduleByClass(Class<Classes> classes, Model model)
{
    Classes classesFromDB =
classService.findClassById(classes.getId());

model.addAttribute("selectedClassesId",
classes.getId());

List<DayOfWeek> dayOfWeekList
= dayOfWeekService.findAllDayOfWeek();

List<String> scheduleMonday =
new ArrayList<>();

```

```

List<String> scheduleTuesday = new ArrayList<>();
List<String> scheduleWednesday = new ArrayList<>();
List<String> scheduleThursday = new ArrayList<>();
List<String> scheduleFriday = new ArrayList<>();

for (DayOfWeek dayOfWeek : dayOfWeekList) {
    if (dayOfWeek.getName().equals("Понеділок")) {

scheduleMonday.addAll(checkLessonNumber(scheduleService.findSchedulesByClassesAndDayOfWeek(classesFromDB, dayOfWeek)));
    }
    if (dayOfWeek.getName().equals("Вівторок")) {

scheduleTuesday.addAll(checkLessonNumber(scheduleService.findSchedulesByClassesAndDayOfWeek(classesFromDB, dayOfWeek)));
    }
    if (dayOfWeek.getName().equals("Среда")) {

scheduleWednesday.addAll(checkLessonNumber(scheduleService.findSchedulesByClassesAndDayOfWeek(classesFromDB, dayOfWeek)));
    }
    if (dayOfWeek.getName().equals("Четвер")) {

scheduleThursday.addAll(checkLessonNumber(scheduleService.findSchedulesByClassesAndDayOfWeek(classesFromDB, dayOfWeek)));
    }
    if (dayOfWeek.getName().equals("П'ятниця")) {

scheduleFriday.addAll(checkLessonNumber(scheduleService.findSchedulesByClassesAndDayOfWeek(classesFromDB, dayOfWeek)));
    }
}

List<Schedule> result =
scheduleService.findSchedulesByClasses(classesFromDB);

List<LessonTime> lessonTimeFromDB =
lessonTimeService.findAll();

model.addAttribute("lessonTime", lessonTimeFromDB);
model.addAttribute("schedule", result);
model.addAttribute("scheduleMonday", scheduleMonday);
model.addAttribute("scheduleTuesday", scheduleTuesday);
model.addAttribute("scheduleWednesday",
scheduleWednesday);
model.addAttribute("scheduleThursday",
scheduleThursday);
model.addAttribute("scheduleFriday", scheduleFriday);

classesController.getClasses(model);

return "classes";
}

2. Пакет Model (основна сутність застосунку)

Schedule.java

package com.example.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter
@Setter
@Table(name = "SCHEDULE")
public class Schedule {

```

```

@Id
@GeneratedValue(strategy =
GenerationType.IDENTITY)
private Long id;
@Column(name = "lesson_number")
private int lessonNumber;

@ManyToOne
@JoinColumn(name = "class_id")
private Classes classes;

@ManyToOne
@JoinColumn(name =
"teacher_subject_id")
private TeacherSubject teacherSubject;

@ManyToOne
@JoinColumn(name =
"day_of_week_id", referencedColumnName = "id")
private DayOfWeek dayOfWeek;

@ManyToOne
@JoinColumn(name =
"lesson_time_id")
private LessonTime lessonTime;

public Schedule() {

}

public Teacher getTeacher() {
return teacherSubject != null ?
teacherSubject.getTeacher() : null;
}

public Schedule(Long id, Classes
classes, TeacherSubject teacherSubject,
DayOfWeek dayOfWeek, LessonTime lessonTime)
{
this.id = id;
this.teacherSubject = teacherSubject;
this.classes = classes;
this.lessonTime = lessonTime;
this.dayOfWeek = dayOfWeek;
}
}

```

3. Пакет repository (головний клас, пов'язаний з розкладом).

ScheduleRepository.java

```

package com.example.repository;

import com.example.model.*;
import
org.springframework.data.jpa.repository.JpaRepository;
import
org.springframework.stereotype.Repository;
import java.util.List;

```

```

@Repository
public interface ScheduleRepository
extends JpaRepository<Schedule, Long> {

List<Schedule>
findSchedulesByTeacherSubject(TeacherSubject
teacherSubject);

```

```
List<Schedule> findSchedulesByClasses(Classes classes);
```

```

List<Schedule>
findSchedulesByTeacherSubjectAndDayOfWeek(TeacherSubject
teacherSubject, DayOfWeek dayOfWeek);

```

```

List<Schedule>
findSchedulesByClassesAndDayOfWeek(Classes classes, DayOfWeek
dayOfWeek);

```

```

}
4. Пакет service (сервіси, який відповідає за бізнес-логіку
розкладу).
ScheduleService.java

```

```
package com.example.service;
```

```

import com.example.model.*;
import java.util.List;

```

```

public interface ScheduleService {
List<Schedule> getAllSchedules();

```

```

List<Schedule>
findSchedulesByTeacherSubjectAndDayOfWeek(TeacherSubject
teacherSubject, DayOfWeek dayOfWeek);

```

```

List<Schedule>
findSchedulesByTeacherSubject(TeacherSubject teacherSubject);

```

```

List<Schedule> findSchedulesByClassesAndDayOfWeek
(Classes classes, DayOfWeek dayOfWeek);

```

```
List<Schedule> findSchedulesByClasses (Classes classes);
```

```
List<Schedule> getAll();
```

```
void saveSchedule(Schedule schedule);
```

```
void deleteScheduleById(Long id);
```

```
void updateSchedule(Long id, Schedule schedule);
```

```

}
5. Frontend частина (головна сторінка).
index.html

```

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>Розклад</title>
<link rel="icon" type="image/png"
href="/static/images/favicon.png">
</head>
<body>
<div class="sidebar">



<ul>
<li>
<a href="/adminPanel" style="font-size:
30px;">Адміністративна<br>панель</a>
</li>
<li><a href="#">Головна</a></li>

```

```

</li>
<a class="nav-link" href="#" data-
toggle="collapse" data-target="#w7-collapse2"
aria-expanded="false" aria-controls="w7-
collapse2">
  <i class="fas fa-fw fa-
calendar"></i><span>Розклад</span>
</a>
<div id="w7-collapse2"
class="collapse" aria-labelledby="headingPages"
data-parent="#accordionSidebar">
  <div class="bg-white py-2 collapse-
inner rounded">
    <a class="collapse-item"
href="/classes">Для учня</a>
    <a class="collapse-item"
href="/teachers">Для викладача</a>
  </div>
</div>
</li>
<form th:action="@{/logout}"
method="post" style="margin-top: 20px;">
  <input type="hidden"

th:name="{_csrf.parameterName}"
th:="{_csrf.token}" />

  <button type="submit" class="logout-
btn">Logout</button>
</form>
</ul>
</div>
<div class="content" style="display: flex;
justify-content: center; align-items: center; height:
100vh;">
  
</div>
<script>

document.addEventListener("DOMContentLoaded", function() {
  const navLink =
document.querySelector('.nav-link');
  const collapseMenu =
document.getElementById('w7-collapse2');

  if (navLink) {
    navLink.addEventListener('click',
function(event) {
      event.preventDefault();

collapseMenu.classList.toggle('show');
    });
  }
</script>
</body>
</html>
5. Конфігурація бази даних (прикладі
даних у головній таблиці розкладу)

```

```
data-school_schedule.sql
```

```

DELETE FROM SCHEDULE;
DELETE FROM CLASSES;
DELETE FROM TEACHER_SUBJECT;
DELETE FROM DAY_OF_WEEK;
DELETE FROM LESSON_TIME;

```

```

DELETE FROM SUBJECT;
DELETE FROM TEACHER;
INSERT INTO SCHEDULE (id, lesson_number, class_id,
teacher_subject_id, day_of_week_id, lesson_time_id) VALUES (1, 1, 6, 3,
1, 1);
schema-school_schedule.sql

```

```

CREATE TABLE IF NOT EXISTS CLASSES (
  id BIGINT AUTO_INCREMENT
PRIMARY KEY,
  name VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS DAY_OF_WEEK (
  id BIGINT AUTO_INCREMENT PRIMARY
KEY,
  name VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS LESSON_TIME (
  id BIGINT AUTO_INCREMENT PRIMARY
KEY,
  time VARCHAR(20) NOT NULL
);

CREATE TABLE IF NOT EXISTS SUBJECT (
  id BIGINT AUTO_INCREMENT
PRIMARY KEY,
  name VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS TEACHER (
  id BIGINT NOT NULL AUTO_INCREMENT
PRIMARY KEY,
  name VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS TEACHER_SUBJECT (
  id BIGINT AUTO_INCREMENT
PRIMARY KEY,
  teacher_id BIGINT,
  subject_id BIGINT,
  FOREIGN KEY (teacher_id)
REFERENCES TEACHER(id),
  FOREIGN KEY (subject_id) REFERENCES SUBJECT(id)
);

CREATE TABLE IF NOT EXISTS SCHEDULE (
  id BIGINT AUTO_INCREMENT
PRIMARY KEY,
  lesson_number INT NOT NULL,
  class_id BIGINT,
  day_of_week_id BIGINT,
  lesson_time_id BIGINT,
  teacher_subject_id BIGINT,
  FOREIGN KEY (teacher_subject_id)
REFERENCES TEACHER_SUBJECT(id),
  FOREIGN KEY (class_id) REFERENCES CLASSES(id),
  FOREIGN KEY (day_of_week_id) REFERENCES
DAY_OF_WEEK(id),
  FOREIGN KEY (lesson_time_id) REFERENCES
LESSON_TIME(id)
);

CREATE TABLE IF NOT EXISTS users (
  id BIGINT AUTO_INCREMENT
PRIMARY KEY,
  username VARCHAR(50) NOT NULL

```

```

UNIQUE,
    password VARCHAR(255) NOT
NULL,
    enabled BOOLEAN DEFAULT TRUE
);

```

```

CREATE TABLE IF NOT EXISTS roles
(
    id BIGINT
AUTO_INCREMENT PRIMARY KEY,
    name
VARCHAR(50) NOT NULL UNIQUE
);

```

```

CREATE TABLE IF NOT EXISTS
users_roles (
    user_id
BIGINT,
    role_id
BIGINT,
    PRIMARY
KEY (user_id, role_id),
    CONSTRAINT fk_user FOREIGN
KEY (user_id) REFERENCES users(id),
    CONSTRAINT fk_role FOREIGN
KEY (role_id) REFERENCES roles(id)
);

```