МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

«Інформаційна система онлайн-менеджменту освітніх процесів Digital University UA: модуль "Розклад занять"»

Виконав студент групи ІПЗ-22-1
ІГНАТЮК Юрій Юрійович

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
САВІЦЬКИЙ Роман Святославович

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»
Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«27» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ІГНАТЮК Юрій Юрійович**
Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**
Тема роботи: **«Інформаційна система онлайн-менеджменту освітніх процесів Digital University UA: модуль "Розклад занять"»**,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**
Вихідні дані для роботи: **розробка системи управління освітніми процесами в університеті, що покращить роботу його адміністрації та надасть функціонал для автоматизації та цифровізації рутинних завдань.**
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	27.02.2026р.	27.02.2026р.
2	Вакалюк Т.А.	24.04.2026р.	24.04.2026р.
3	Вакалюк Т.А.	02.05.2026р.	02.05.2026р.

РЕФЕРАТ

Кваліфікаційна робота складається з програмного комплексу для інформаційної системи онлайн-менеджменту освітніх процесів Digital University UA, зокрема модуля «Розклад занять», та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить 107 сторінок, 39 ілюстрацій та 2 таблиці.

Головною метою роботи є створення модуля інформаційної системи, який дозволяє цифровізувати процеси формування, імпортування, редагування, контролю та відображення розкладу навчальних занять у закладах вищої освіти. Розроблений модуль забезпечує зручну взаємодію з розкладом для різних категорій користувачів, зокрема студентів, викладачів та адміністративного персоналу.

У ході виконання роботи було визначено основні завдання розробки модуля, проведено аналіз аналогів програмного продукту, обґрунтовано вибір архітектури системи, інструментальних засобів та системи керування базами даних. Також було спроєктовано структуру бази даних, реалізовано імпорт розкладу з FET/XML-файлу, підтримку версій розкладу, редагування, копіювання та перенесення занять, перевірку конфліктів, роботу з посиланнями й нотатками, персоналізоване відображення розкладу студента, кешування даних за допомогою Redis та експорт розкладу у форматі Excel.

КЛЮЧОВІ СЛОВА: BACKEND, ІНФОРМАЦІЙНА СИСТЕМА, РОЗКЛАД, PHP, YII2, MYSQL, REDIS, FET/XML, ІМПОРТ, КЕШУВАННЯ.

ABSTRACT

The qualification work consists of a software package for the online educational process management information system Digital University UA, in particular the “Schedule” module, and an explanatory note. The explanatory note contains 107 pages, 39 illustrations, and 2 tables.

The main goal of the work is to create an information system module that digitalizes the processes of generating, importing, editing, controlling, and displaying the class schedule in higher education institutions. The developed module provides convenient interaction with the schedule for different categories of users, including students, teachers, and administrative staff.

During the work, the main tasks for developing the module were defined, analogues of the software product were analyzed, and the choice of system architecture, development tools, and database management system was justified. The database structure was designed, and the functionality for importing schedules from FET/XML files, managing schedule versions, editing, copying and moving classes, detecting conflicts, managing links and notes, displaying personalized student schedules, caching data with Redis, and exporting schedules to Excel was implemented.

KEYWORDS: BACKEND, INFORMATION SYSTEM, SCHEDULE, PHP, YII2, MYSQL, REDIS, FET/XML, IMPORT, CACHING.

ЗМІСТ

РЕФЕРАТ **3**

ABSTRACT **4**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 7

ВСТУП 8

РОЗДІЛ 1. АНАЛІЗ ВИМОГ КОРИСТУВАЧА ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ 10

1.1. Постановка задачі 10

1.2. Аналіз аналогів програмного продукту 12

1.3. Вибір архітектури системи 17

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення 24

Висновки до першого розділу 29

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ОНЛАЙН-МЕНЕДЖМЕНТУ ОСВІТНІХ ПРОЦЕСІВ 32

2.1. Визначення варіантів використання та структури системи 32

2.2. Обґрунтування вибору системи управління базами даних 40

2.3. Розробка бази даних системи 45

2.4. Реалізація функціоналу системи управління навчальним процесом 51

Висновки до другого розділу 66

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ОНЛАЙН-МЕНЕДЖМЕНТУ ОСВІТНІХ ПРОЦЕСІВ 70

3.1. Порядок встановлення та налаштування параметрів додатку 70

3.2. Структура інтерфейсу додатку 75

3.3. Тестування роботи розробленого додатку 93

Висновки до третього розділу 99

ВИСНОВКИ 102

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 106

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ЗВО – заклад вищої освіти

ПЗ – програмне забезпечення

ІС – інформаційна система

СУБД – система управління базою даних

CRUD – Create, Read, Update, Delete

MVC – Model, View, Controller

API – Application Programming Interface

ORM – Object-Relational Mapping

HTTP – HyperText Transfer Protocol

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JS – JavaScript

PHP – Hypertext Preprocessor

Yii2 – PHP-фреймворк для розробки вебзастосунків

MySQL – реляційна система керування базами даних

Redis – сховище даних у пам’яті, що використовується для кешування

Docker – платформа контейнеризації програмного забезпечення

XML – Extensible Markup Language

FET – програмне забезпечення для автоматичного формування розкладу

FET/XML – XML-файл, сформований програмним забезпеченням FET

UML – Unified Modeling Language

ER – Entity-Relationship

Excel – табличний формат для експорту даних

TXT – текстовий формат файлу

ВСТУП

З розвитком цифрових технологій заклади вищої освіти все активніше впроваджують інформаційні системи для автоматизації освітніх процесів. Одним із найважливіших елементів організації навчальної діяльності є розклад занять, який повинен враховувати значну кількість параметрів, зокрема навчальні плани, аудиторний фонд, навантаження викладачів, академічні групи, підгрупи та вибіркові дисципліни. Використання застарілих або частково автоматизованих підходів до формування та супроводу розкладу ускладнює його адміністрування, збільшує ризик виникнення конфліктів та потребує значних часових витрат на внесення змін. Крім того, сучасні умови організації освітнього процесу вимагають оперативного доступу до актуальної інформації про заняття для студентів, викладачів та адміністрації. У зв'язку з цим актуальним є розроблення програмних рішень, які забезпечують централізоване управління розкладом та його інтеграцію з іншими компонентами інформаційної системи закладу освіти.

Метою дослідження є розробка модуля «Розклад занять» інформаційної системи онлайн-менеджменту освітніх процесів Digital University UA, який забезпечує автоматизацію процесів формування, редагування, контролю та відображення розкладу занять для різних категорій користувачів.

Досягнення поставленої мети передбачає виконання низки важливих завдань, серед яких аналіз сучасних підходів до організації розкладу в закладах вищої освіти, дослідження існуючих програмних рішень у даній сфері, проєктування структури даних та архітектури модуля, реалізація механізмів імпорту й обробки розкладу, розробка інструментів його редагування та адміністрування, впровадження системи контролю конфліктів між групами, викладачами та аудиторіями, а також забезпечення відображення персоналізованого розкладу в особистих кабінетах користувачів.

Об’єктом дослідження є процеси формування, управління та супроводу розкладу навчальних занять у закладах вищої освіти.

Предметом дослідження виступають технології, методи та програмні засоби розробки інформаційних систем, призначених для автоматизації

створення, редагування, контролю та відображення розкладу навчальних занять, а також механізми їх інтеграції в єдине цифрове освітнє середовище.

Практичне значення роботи полягає у створенні програмного модуля, який може бути використаний у межах інформаційної системи Digital University UA для централізованого управління розкладом навчальних занять. Розроблений функціонал дозволяє спростити процес імпорту, редагування, перевірки та публікації розкладу, зменшити кількість ручних операцій і підвищити точність даних. Використання модуля забезпечує студентам і викладачам швидкий доступ до актуальної інформації про заняття, а адміністративним працівникам надає зручні інструменти для контролю розкладу, виявлення конфліктів, роботи з версіями та експорту необхідних даних.

РОЗДІЛ 1. АНАЛІЗ ВИМОГ КОРИСТУВАЧА ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ

1.1. Постановка задачі

Одним із ключових елементів організації освітнього процесу у закладах вищої освіти є формування та супровід розкладу навчальних занять[1;2]. Від якості його побудови залежить ефективність використання аудиторного фонду, рівномірність навантаження викладачів, своєчасне проведення занять та зручність доступу учасників освітнього процесу до актуальної інформації. Незважаючи на активний розвиток цифрових технологій, у багатьох закладах освіти процеси управління розкладом залишаються частково автоматизованими або потребують значної кількості ручних операцій, що збільшує ризик виникнення помилок та ускладнює оперативне внесення змін.

Планування розкладу є складною задачею, яка потребує врахування великої кількості взаємопов'язаних параметрів, зокрема академічних груп, підгруп, викладачів, аудиторій, навчальних дисциплін, семестрів та особливостей навчальних планів[20]. Відсутність централізованого механізму управління розкладом призводить до появи конфліктних ситуацій, коли одна група, викладач або аудиторія можуть бути задіяні у кількох заняттях одночасно. Крім того, складність адміністрування значно зростає при необхідності підтримки вибіркових дисциплін, індивідуальних освітніх траєкторій студентів та декількох версій розкладу протягом навчального року.

Для вирішення зазначених проблем доцільним є впровадження централізованої інформаційної системи, яка забезпечить автоматизацію процесів формування, редагування, контролю та відображення розкладу навчальних занять. Таке рішення дозволить зменшити кількість ручних операцій, підвищити точність даних, спростити адміністрування навчального процесу та забезпечити швидкий доступ до актуального розкладу для всіх категорій користувачів.

Саме тому в межах інформаційної системи «Цифровий університет» було прийнято рішення про розробку модуля «Розклад занять». Даний модуль

призначений для централізованого управління розкладом навчальних занять, його імпортування з зовнішніх джерел, контролю конфліктів, внесення змін та подальшого відображення інформації в особистих кабінетах студентів і працівників університету. Особливістю модуля є підтримка різних версій розкладу, навчальних років, семестрів, підгруп та вибіркових дисциплін, що дозволяє враховувати специфіку організації сучасного освітнього процесу.

Основним завданням модуля є забезпечення повного циклу роботи з розкладом навчальних занять. Система повинна надавати можливість створювати та редагувати розклад, змінювати параметри окремих занять, працювати з посиланнями й нотатками, прикріпленими до пар, а також підтримувати різні версії розкладу.

Окреме значення має можливість затвердження та публікації актуальної версії, оскільки саме вона надалі відображається для студентів, викладачів та інших користувачів системи.Важливою частиною роботи модуля є імпортування та первинне формування розкладу. Система повинна підтримувати завантаження даних із FET/XML-файлів, сформованих стороннім програмним забезпеченням, створювати нові версії розкладу на основі імпортованих даних та оновлювати інформацію про заняття, викладачів, академічні групи й аудиторії. Такий підхід дозволяє використовувати зовнішні засоби для початкового формування розкладу, а подальше редагування, перевірку та публікацію виконувати вже в межах Digital University UA.

Окремим завданням системи є контроль коректності розкладу. Модуль повинен виявляти конфлікти між академічними групами, перевіряти накладання у навантаженні викладачів та контролювати одночасне використання аудиторій. Крім того, система має забезпечувати перегляд усіх пар через сторінку адміністративного контролю з можливістю фільтрації за основними параметрами та подальшого експорту відфільтрованих даних. Це дозволяє перевіряти структуру розкладу перед його публікацією та своєчасно виявляти помилки.

Не менш важливим є завдання відображення розкладу для різних категорій користувачів. Система повинна забезпечувати перегляд розкладу за академічними групами, викладачами та аудиторіями, а також формувати персоналізований розклад студента в особистому кабінеті. У розкладі має відображатися не лише основна інформація про заняття, а й додаткові дані, зокрема посилання на онлайн-зустрічі, електронні ресурси або службові нотатки.

Модуль також повинен підтримувати роботу з вибірковими дисциплінами. Для цього передбачається формування окремого розкладу вибіркових компонентів та автоматичне включення до персонального розкладу студента лише тих занять, що відповідають його індивідуальному вибору. Це дозволяє враховувати індивідуальні освітні траєкторії здобувачів освіти та уникати відображення зайвих дисциплін у їхньому розкладі.Додатковим завданням системи є експорт даних у форматі Excel. Така можливість потрібна для підготовки звітних матеріалів, перевірки інформації та подальшого використання даних структурними підрозділами університету. У межах модуля експорт пов’язаний передусім зі сторінкою адміністративного контролю, де користувач може сформувати потрібну вибірку за допомогою фільтрів і зберегти її у табличному форматі.

Реалізація зазначеного функціоналу дозволить підвищити ефективність управління навчальним процесом, спростити роботу працівників, відповідальних за формування розкладу, а також забезпечити студентів і викладачів актуальною та достовірною інформацією про навчальні заняття.

1.2. Аналіз аналогів програмного продукту

У процесі проєктування модуля «Розклад занять» було проведено аналіз існуючих програмних рішень, які частково вирішують задачі цифровізації освітнього процесу та автоматизації формування розкладу занять. Варто зазначити, що повного аналога розроблюваного модуля у відкритому доступі не було виявлено, оскільки більшість наявних систем або орієнтовані на загальне управління навчанням, або спеціалізуються лише на побудові розкладу без

глибокої інтеграції з особистими кабінетами студентів, вибірковими дисциплінами, ролями користувачів та внутрішніми даними закладу освіти.

У багатьох закладах вищої освіти розклад занять досі формується та поширюється за допомогою табличних редакторів, зокрема **Microsoft Excel** або **Google Sheets**. Такий підхід є простим у використанні та не потребує складної технічної інфраструктури. Відповідальні працівники можуть створювати таблиці з розкладом, надавати до них доступ іншим користувачам або експортувати їх у формат PDF для подальшого поширення серед студентів і викладачів.

Перевагою такого способу є його доступність, оскільки більшість користувачів уже мають базові навички роботи з електронними таблицями. Крім того, хмарні редактори дозволяють кільком працівникам одночасно переглядати або редагувати документ, що частково спрощує процес оновлення інформації.

Проте такий підхід має суттєві недоліки. Табличні редактори не забезпечують повноцінної перевірки конфліктів між групами, викладачами та аудиторіями. У разі зміни однієї пари відповідальна особа повинна вручну перевіряти, чи не виникають накладання в інших частинах розкладу. Крім того, таблиці не дозволяють автоматично формувати персоналізований розклад студента з урахуванням вибіркових дисциплін. Також виникають труднощі з контролем версій, правами доступу, історією змін та інтеграцією з іншими модулями інформаційної системи університету.

Ще більш спрощеним підходом є поширення розкладу у вигляді друкованих таблиць або статичних файлів. У такому випадку розклад може розміщуватися на інформаційних стендах, надсилатися через електронну пошту або публікуватися у вигляді зображення чи PDF-файлу. Такий спосіб є простим, але не відповідає сучасним вимогам до оперативності та зручності доступу до інформації.

Основним недоліком такого підходу є складність оновлення інформації. У разі зміни аудиторії, викладача або часу проведення заняття необхідно повторно формувати та поширювати нову версію документа. Це підвищує ризик того, що частина користувачів продовжить користуватися застарілою інформацією. Крім

того, статичний розклад не підтримує пошук, фільтрацію, персоналізацію та автоматичну перевірку помилок.

Як аналог загальної інформаційної системи для підтримки освітнього процесу можна розглянути **Moodle**. Moodle є платформою з відкритим кодом, яка використовується для організації дистанційного навчання, створення електронних курсів, проведення тестування, оцінювання та комунікації між викладачами й студентами[3]. Система має широку екосистему модулів і може бути адаптована під потреби конкретного навчального закладу.

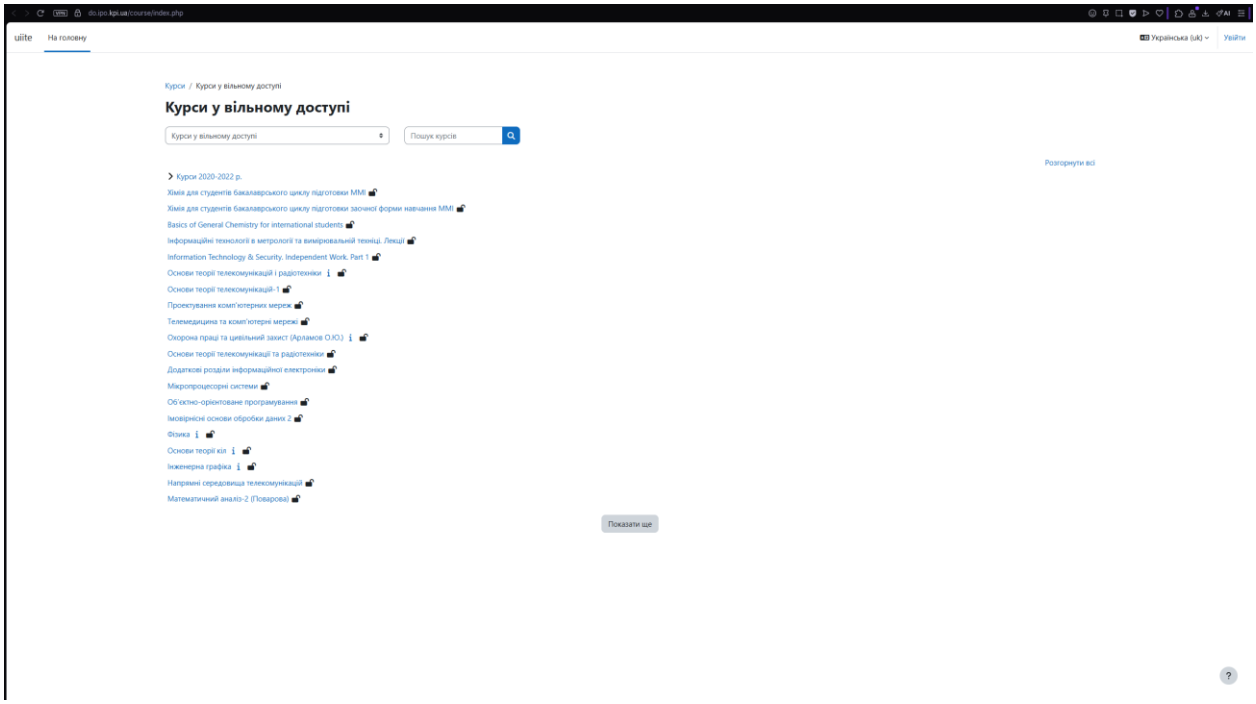

Рис. 1.1 – Платформа Moodle, приклад інституту післядипломної освіти КПІ ім. Ігоря Сікорського

Серед переваг Moodle можна виділити відкритий код, гнучкість налаштування, підтримку великої кількості плагінів та активну спільноту користувачів. Платформа добре підходить для організації електронного навчання, розміщення навчальних матеріалів, проведення тестування та збору результатів навчальної діяльності.

Водночас Moodle не є спеціалізованою системою для повноцінного управління університетським розкладом. Її основне призначення полягає в організації навчальних курсів та дистанційної взаємодії, а не в автоматизованому

формуванні розкладу з урахуванням аудиторій, викладачів, підгруп, конфліктів і версій. Для реалізації таких можливостей зазвичай потрібні додаткові плагіни або окремі інтеграції, що ускладнює впровадження та підтримку системи.

Окремим класом аналогів є спеціалізовані системи для побудови розкладу. Одним із таких рішень є **UniTime** — система для університетського планування занять, яка підтримує створення навчальних та екзаменаційних розкладів, керування змінами, роботу з аудиторіями та мінімізацію конфліктів між заняттями[4;5].

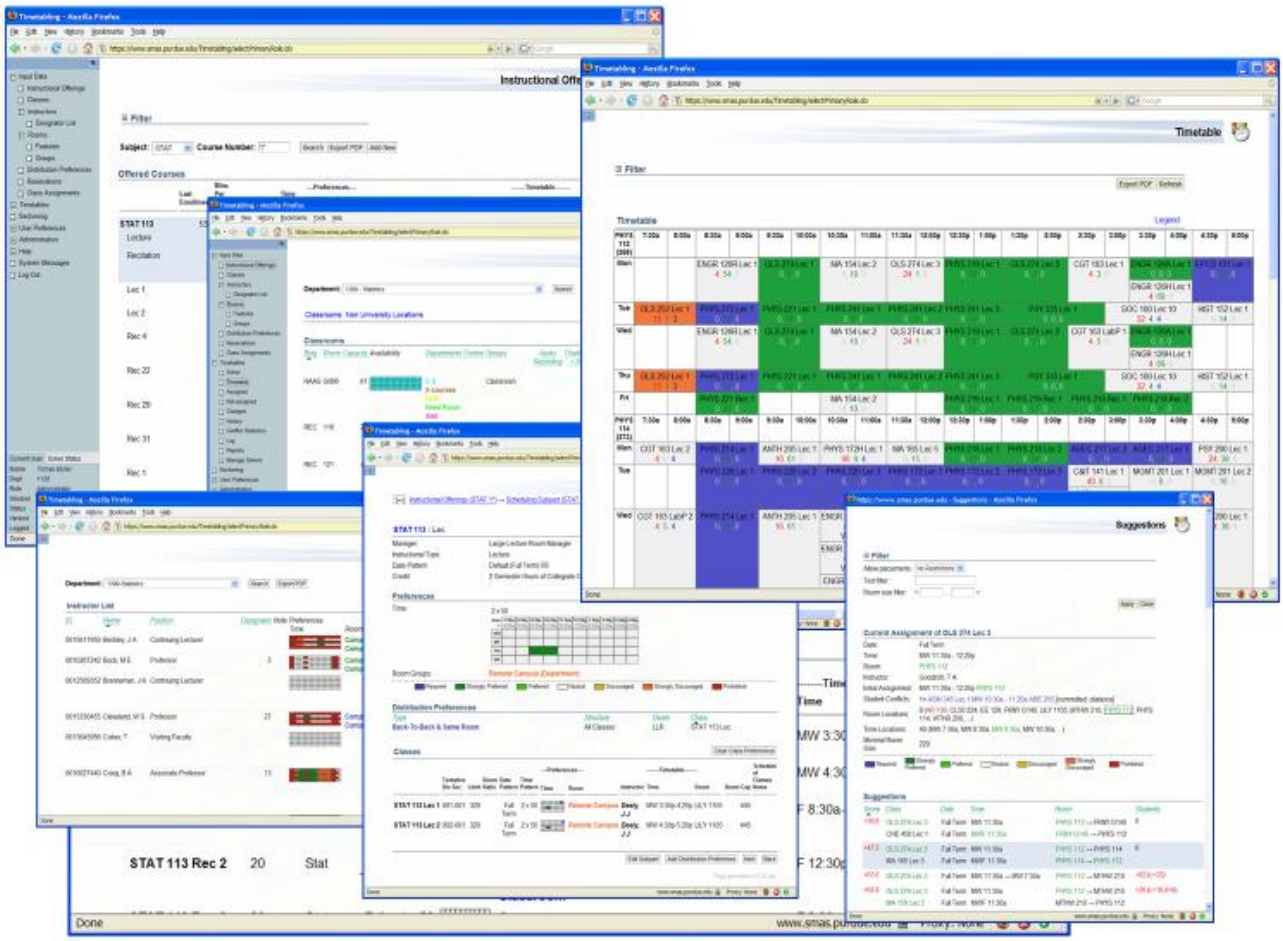

Рис. 1.2 – Система UniTime

Перевагою UniTime є орієнтація саме на університетське середовище. Система підтримує складні сценарії планування, може використовуватися різними підрозділами навчального закладу та дозволяє координувати процес формування розкладу між кількома відповідальними особами. Також важливою

перевагою є підтримка роботи з індивідуальними заняттями студентів і намагання мінімізувати конфлікти під час планування.

Недоліком UniTime у контексті даної роботи є те, що система є окремим програмним продуктом, який потребує окремого впровадження, налаштування та інтеграції з наявною інформаційною інфраструктурою університету. Для використання в межах Digital University UA таке рішення вимагало б додаткової синхронізації з існуючими даними про студентів, викладачів, групи, дисципліни та вибіркові компоненти. Крім того, використання готової системи не дозволяє повністю врахувати внутрішню логіку платформи, зокрема механізм персоналізованого розкладу студента на основі вибіркових дисциплін.

Також серед спеціалізованих аналогів можна розглянути **aSc Timetables** — програмний продукт для створення розкладів занять, який орієнтований переважно на школи та інші освітні установи[7]. Система надає інструменти для автоматизованого складання розкладу, роботи з викладачами, класами, аудиторіями та різними часовими обмеженнями.

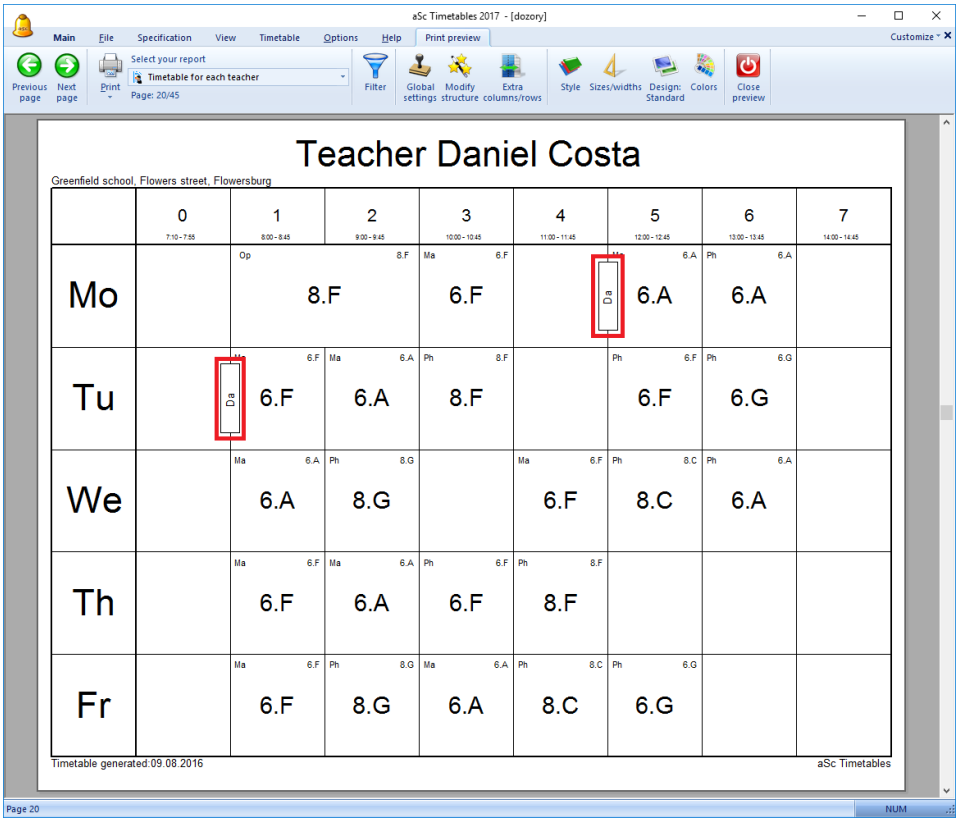

Рис. 1.3 – Система aSc Timetables

Перевагою aSc Timetables є зручний інтерфейс та наявність інструментів автоматичного формування розкладу. Система може бути корисною для навчальних закладів, яким потрібне відносно швидке створення розкладу без розробки власного програмного забезпечення.

Проте цей продукт також не вирішує повністю задачі, поставлені в межах даної кваліфікаційної роботи. Він не є частиною інформаційної системи університету, не забезпечує безпосередню інтеграцію з особистими кабінетами студентів і не враховує внутрішні механізми Digital University UA. Крім того, для використання такого рішення необхідно або вручну переносити дані в інші системи, або розробляти окремі механізми інтеграції.

У результаті аналізу аналогів можна зробити висновок, що наявні програмні рішення лише частково покривають потреби, пов’язані з управлінням розкладом занять. Табличні редактори та статичні файли є простими, але не забезпечують автоматизації, контролю конфліктів і персоналізації. Moodle є потужною платформою для електронного навчання, однак не спеціалізується на повноцінному управлінні розкладом. UniTime, FET та aSc Timetables мають інструменти для планування занять, проте потребують окремої інтеграції з університетською інформаційною системою[4;5;6;7].

Таким чином, розробка власного модуля «Розклад занять» у межах Digital University UA є доцільною, оскільки дозволяє врахувати особливості внутрішньої структури закладу освіти, забезпечити роботу з різними ролями користувачів, підтримати версійність розкладу, вибіркові дисципліни, перевірку конфліктів та персоналізоване відображення розкладу в особистому кабінеті студента.

1.3. Вибір архітектури системи

Під час розробки програмного забезпечення одним із найважливіших етапів є вибір архітектури системи. Від прийнятого архітектурного рішення залежить не лише спосіб організації програмного коду, а й подальша зручність супроводу, можливість масштабування, стабільність роботи, швидкість внесення

змін та якість інтеграції з іншими компонентами інформаційної системи. Для програмних продуктів, що використовуються у закладах вищої освіти, архітектура має особливе значення, оскільки такі системи працюють із великою кількістю взаємопов'язаних даних, підтримують різні категорії користувачів і повинні забезпечувати надійний доступ до актуальної інформації.

Модуль «Розклад занять» є частиною комплексної інформаційної системи Digital University UA, тому його архітектура не могла розглядатися окремо від загальної структури платформи. Розклад навчальних занять пов'язаний із багатьма сутностями освітнього процесу: академічними групами, викладачами, аудиторіями, дисциплінами, семестрами, навчальними роками, вибірковими компонентами та користувачами системи. Через це модуль повинен не лише забезпечувати власний функціонал, а й коректно взаємодіяти з іншими частинами програмного комплексу, використовуючи спільні дані та загальні механізми авторизації, доступу й адміністрування.

Перед вибором архітектурного підходу було проаналізовано кілька можливих варіантів побудови системи. Одним із найпоширеніших підходів є монолітна архітектура, за якої основний функціонал програмного продукту реалізується в межах єдиного застосунку. У такій архітектурі компоненти системи мають спільну кодову базу, використовують єдині механізми взаємодії з базою даних і розгортаються як частина одного програмного середовища. Перевагою монолітного підходу є простота реалізації, зручність розгортання, менша кількість зовнішніх залежностей та зрозуміла структура взаємодії між частинами системи.

Для модуля «Розклад занять» монолітний підхід є доцільним, оскільки розробка виконувалася в межах уже існуючої інформаційної системи. У цьому випадку створення окремого незалежного сервісу для розкладу вимагало б додаткової організації обміну даними, налаштування окремих програмних інтерфейсів та синхронізації з основною базою даних. Оскільки розклад активно використовує вже наявні довідники й сутності платформи, зокрема дані про студентів, працівників, академічні групи та дисципліни, інтеграція в основний

застосунок дозволяє уникнути дублювання інформації та спростити підтримку цілісності даних.

Іншим архітектурним підходом є мікросервісна архітектура, яка передбачає поділ системи на набір незалежних сервісів. Кожен сервіс відповідає за окрему функціональну область і взаємодіє з іншими сервісами через програмні інтерфейси. Такий підхід має переваги в масштабованих системах, де окремі компоненти можуть розгортатися, оновлюватися та масштабуватися незалежно один від одного. Водночас мікросервісна архітектура ускладнює супровід системи, оскільки потребує додаткових механізмів взаємодії, обробки помилок, моніторингу, логування та контролю узгодженості даних.

У межах розробки модуля «Розклад занять» використання повноцінної мікросервісної архітектури не було оптимальним рішенням. Основна причина полягає в тому, що модуль тісно пов’язаний із внутрішньою структурою Digital University UA та використовує спільну базу даних системи. Винесення його в окремий сервіс призвело б до ускладнення інтеграції, необхідності додаткової синхронізації даних і збільшення витрат на розробку та підтримку. Тому більш доцільним стало використання клієнт-серверної архітектури в межах існуючого вебзастосунку, доповненої внутрішнім поділом на логічні шари.

Обрана клієнт-серверна архітектура передбачає, що користувач взаємодіє із системою через веббраузер, а основна обробка запитів, перевірка прав доступу, робота з базою даних і формування результатів виконуються на серверній стороні. Такий підхід є зручним для закладу вищої освіти, оскільки користувачам не потрібно встановлювати додаткове програмне забезпечення на власні пристрої. Доступ до модуля може здійснюватися з різних комп’ютерів або мобільних пристроїв за умови наявності браузера та підключення до мережі.

Серверна частина системи реалізована з використанням фреймворку Yii2, який підтримує архітектурний шаблон MVC[8;9]. Даний шаблон передбачає розділення програмного продукту на три основні складові: модель, представлення та контролер. Модель відповідає за роботу з даними, описує основні сутності предметної області та забезпечує взаємодію з базою даних.

Представлення відповідає за формування інтерфейсу користувача, тобто за відображення інформації у зручному для сприйняття вигляді. Контролер приймає запити користувача, викликає відповідну бізнес-логіку, отримує дані з моделей або сервісів і передає результат у представлення.

Використання MVC є важливим для модуля «Розклад занять», оскільки він містить значну кількість сценаріїв роботи з даними. Наприклад, користувач може переглядати розклад за академічною групою, викладачем або аудиторією, адміністративний працівник може імпортувати розклад із FET/XML-файлу, редагувати заняття, працювати з версіями, перевіряти конфлікти або експортувати відфільтровані дані зі сторінки «Контроль». Завдяки поділу на моделі, представлення та контролери кожна частина системи виконує власну роль, що зменшує складність коду та полегшує його подальшу підтримку.

Разом із класичним підходом MVC у модулі використовується сервісний шар[18]. Його застосування є доцільним тому, що значна частина логіки розкладу є складнішою за звичайні операції створення, читання, оновлення та видалення даних. Наприклад, формування табличного представлення розкладу потребує групування занять за днями, тижнями та часовими парами. Спискове представлення, у свою чергу, має іншу логіку відображення й використовується в особистому кабінеті студента, на сторінці розкладу викладачів та на сторінці «Мій розклад». Якщо всю цю логіку розміщувати безпосередньо в контролерах, код став би перевантаженим і менш зручним для підтримки.

Сервісний шар дозволяє винести основну бізнес-логіку в окремі класи, які не залежать безпосередньо від елементів інтерфейсу. У межах модуля до таких сервісів належать компоненти, що відповідають за імпорт розкладу, обробку FET/XML-файлів, створення версій, формування різних представлень розкладу, перевірку конфліктів, роботу з кешем та експорт відфільтрованих даних. Такий підхід робить архітектуру більш гнучкою, оскільки зміни в одному сценарії роботи не потребують повного переписування інших частин модуля.

Важливою частиною архітектури є рівень зберігання даних. Для цього використовується реляційна система керування базами даних MySQL. Розклад

навчальних занять має чітку структуровану природу, оскільки кожна пара пов’язана з конкретним семестром, версією, дисципліною, типом заняття, групами, викладачами, аудиторіями, днем, навчальним тижнем і часовим проміжком. Реляційна модель добре підходить для таких даних, оскільки дозволяє встановлювати зв’язки між таблицями, підтримувати цілісність інформації та виконувати складні вибірки.

У базі даних модуль використовує окремі таблиці для збереження версій розкладу, навчальних тижнів, днів, часових пар, основних записів занять і зв’язків занять із групами, викладачами та аудиторіями. Така структура дозволяє уникати дублювання даних і забезпечує можливість гнучкої роботи з різними представленнями розкладу. Наприклад, одна й та сама пара може бути відображена в розкладі групи, викладача та аудиторії, оскільки ці представлення формуються на основі спільної базової структури даних.

Окремим елементом архітектури є підтримка версійності розкладу. У системі може існувати кілька варіантів розкладу для одного семестру, але для звичайних користувачів має відображатися лише актуальна активна версія. Це дозволяє адміністрації готувати зміни, перевіряти їх і лише після цього робити доступними для студентів та викладачів. З архітектурної точки зору це означає, що більшість запитів до розкладу повинні враховувати не тільки групу, викладача або аудиторію, а й конкретну версію розкладу та її статус.

Для підвищення продуктивності системи використовується Redis. Його застосування є важливим насамперед для персоналізованого розкладу студентів. Такий розклад формується не лише на основі академічної групи, а й з урахуванням вибіркових дисциплін, які обрав конкретний студент. Отже, для кожного користувача результат може відрізнятися, а формування розкладу потребує виконання додаткових запитів і обробки даних. Кешування дозволяє зберігати вже сформований результат і швидко віддавати його при повторному зверненні.

Використання Redis дозволяє зменшити навантаження на базу даних і пришвидшити роботу особистих кабінетів студентів. Це особливо важливо в

періоди активного користування системою, наприклад на початку семестру, коли велика кількість студентів і викладачів одночасно переглядає розклад. При зміні даних розкладу кеш має оновлюватися або очищуватися, щоб користувачі отримували актуальну інформацію. Таким чином, Redis виконує роль допоміжного інструмента для підвищення швидкодії, але не замінює основну базу даних.

Ще одним важливим архітектурним аспектом є контроль доступу. У системі існують різні категорії користувачів, серед яких студенти, викладачі, завідувачі кафедр, декани та адміністративні працівники. Кожна з цих категорій повинна мати доступ лише до тих дій і даних, які відповідають її ролі. Студент повинен бачити власний персоналізований розклад, викладач — власний розклад і доступні йому дії, а адміністрація — інструменти імпорту, редагування, контролю, роботи з версіями та експорту.

Окрім загального розмежування за ролями, для модуля важливо перевіряти доступ до конкретних об’єктів. Наприклад, викладач може мати право працювати зі своїм розкладом або з розкладом підлеглого працівника, але не повинен отримувати доступ до довільних даних, які не належать до його повноважень. Такий підхід підвищує безпеку системи та дозволяє точніше контролювати дії користувачів. У межах архітектури це реалізується через окремі механізми перевірки доступу, які застосовуються на рівні контролерів і сервісів.

Оскільки модуль є частиною веборієнтованої системи, важливу роль відіграє також рівень інтерфейсу користувача. Представлення повинні забезпечувати зручну роботу з різними сценаріями: переглядом розкладу в табличному форматі, переглядом у списковому форматі, редагуванням окремих занять через модальні вікна, роботою з версіями, переглядом помилок імпорту та використанням фільтрів на сторінці «Контроль». Для забезпечення інтерактивності використовується JavaScript, який дозволяє виконувати окремі дії без зайвого перезавантаження сторінок і підвищує зручність роботи користувача.

У межах обраної архітектури також передбачено контейнеризоване середовище розгортання. Використання Docker дозволяє уніфікувати програмне середовище, у якому працює застосунок, база даних, кеш-сховище та інші залежності. Це спрощує налаштування системи на різних серверах і зменшує ймовірність виникнення помилок, пов'язаних із відмінностями між локальним середовищем розробника та серверним середовищем. Для командної розробки та подальшого супроводу це є важливою перевагою, оскільки робить процес розгортання більш передбачуваним.

Загальна логіка роботи модуля в обраній архітектурі може бути описана таким чином. Користувач надсилає запит через вебінтерфейс, після чого контролер приймає цей запит і визначає необхідну дію. Якщо дія потребує доступу до даних, контролер звертається до відповідної моделі або сервісу. Сервіс виконує бізнес-логіку, наприклад формує розклад, перевіряє конфлікти, обробляє імпортований файл або готує дані для експорту. Після цього результат передається до представлення, яке формує сторінку або відповідь для користувача. Якщо дані часто використовуються, вони можуть бути отримані з кешу або збережені в ньому для подальших звернень.

Обрана архітектура дозволяє забезпечити баланс між складністю системи та зручністю її підтримки. З одного боку, модуль інтегрований у загальну платформу Digital University UA, що спрощує роботу зі спільними даними та користувачами. З іншого боку, використання MVC, сервісного шару, кешування та окремих механізмів контролю доступу дозволяє уникнути надмірної зв'язаності компонентів і забезпечує можливість подальшого розширення функціональності.

Таким чином, для реалізації модуля «Розклад занять» було обрано клієнт-серверну архітектуру в межах існуючого вебзастосунку Digital University UA з використанням шаблону MVC, сервісного шару, реляційної бази даних MySQL та кешування за допомогою Redis. Такий підхід забезпечує чітке розділення відповідальності між компонентами системи, підтримує інтеграцію з іншими модулями платформи, підвищує продуктивність під час роботи з

персоналізованими розкладами та створює основу для подальшого розвитку програмного продукту.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Під час розробки модуля «Розклад занять» інформаційної системи Digital University UA було необхідно обрати такі інструментальні засоби, які забезпечували б стабільну роботу програмного продукту, можливість його інтеграції з уже наявною платформою, зручність подальшого супроводу та відповідність функціональним вимогам. Оскільки модуль є частиною більшої системи онлайн-менеджменту освітніх процесів, вибір технологій не міг здійснюватися ізольовано від загальної архітектури проєкту. Необхідно було врахувати вже наявну серверну інфраструктуру, структуру бази даних, механізми авторизації користувачів, рольову модель доступу та особливості взаємодії з іншими складовими системи.

Модуль «Розклад занять» повинен обробляти значну кількість взаємопов’язаних даних. До них належать відомості про навчальні роки, семестри, академічні групи, підгрупи, викладачів, аудиторії, дисципліни, типи занять, навчальні тижні, часові пари, версії розкладу, посилання, нотатки та обмеження. Крім того, система повинна забезпечувати різні сценарії роботи: імпорт FET/XML-файлів, створення та редагування занять, перегляд розкладу за групами, викладачами й аудиторіями, формування персоналізованого розкладу студента, перевірку конфліктів, роботу з версіями, кешування та експорт відфільтрованих даних. Саме тому обрані інструментальні засоби повинні були бути достатньо гнучкими для реалізації складної бізнес-логіки та достатньо надійними для використання в реальному освітньому середовищі.

Основною мовою програмування для серверної частини було обрано PHP[10;11]. Дана мова є поширеним інструментом для розробки веборієнтованих інформаційних систем і добре підходить для створення серверної логіки, роботи з базами даних та формування динамічних вебсторінок.

Важливою перевагою PHP є велика кількість готових бібліотек, зрозуміла структура розгортання, підтримка сучасних фреймворків і наявність широкої спільноти розробників. У контексті модуля «Розклад занять» PHP використовується для обробки запитів користувачів, реалізації логіки роботи з розкладом, взаємодії з базою даних, перевірки прав доступу та підготовки даних для відображення в інтерфейсі.

Вибір PHP також є доцільним з огляду на те, що інформаційна система Digital University UA уже побудована на технологіях, сумісних із цією мовою. Це дозволяє не створювати окреме середовище для нового модуля, а інтегрувати його в загальну структуру проєкту. Такий підхід спрощує підтримку системи, зменшує кількість технологічних залежностей та дозволяє використовувати спільні механізми авторизації, маршрутизації, доступу до даних і обробки помилок.

Для реалізації серверної частини використано фреймворк Yii2[8;9]. Він підтримує архітектурний шаблон MVC, що дозволяє чітко розділити логіку роботи системи, обробку запитів користувача та відображення даних у вебінтерфейсі. У межах модуля «Розклад занять» це має особливе значення, оскільки система містить велику кількість сценаріїв, які потребують структурованої організації коду. Наприклад, окремі контролери можуть відповідати за перегляд розкладу, роботу з версіями, імпорт файлів, сторінку адміністративного контролю або редагування занять, тоді як моделі описують відповідні сутності бази даних.

Однією з переваг Yii2 є підтримка механізму Active Record, який спрощує взаємодію з реляційною базою даних[8;9]. Завдяки цьому розробник може працювати з таблицями бази даних через об’єктні моделі, що підвищує зручність розробки та зменшує кількість повторюваного SQL-коду. Для модуля «Розклад занять» це важливо, оскільки структура бази даних містить значну кількість пов’язаних таблиць: основні записи пар, зв’язки занять із групами, викладачами й аудиторіями, таблиці версій, тижнів, годин, обмежень, нотаток і посилань.

Використання Active Record дозволяє зручніше описувати ці зв'язки та працювати з ними у коді.

Крім стандартного MVC-підходу, у модулі активно використовується сервісний шар. Це пов'язано з тим, що логіка роботи з розкладом є складнішою за звичайні операції створення, читання, оновлення та видалення записів. Наприклад, формування табличного представлення потребує групування даних за днями, тижнями та часовими парами, а спискове представлення має іншу логіку підготовки даних. Імпорт FET/XML-файлу також потребує окремої обробки, зіставлення даних із внутрішніми довідниками та створення нової версії розкладу. Тому винесення такої логіки в окремі сервіси дозволяє уникнути перевантаження контролерів і зробити код більш зрозумілим для супроводу.

Для зберігання основних даних було обрано систему керування базами даних MySQL[14;15]. Вона є реляційною СУБД, що добре підходить для роботи зі структурованими даними та зв'язками між сутностями. Розклад навчальних занять має саме таку природу, оскільки кожне заняття пов'язане з конкретною дисципліною, типом заняття, викладачами, академічними групами, аудиторіями, навчальними тижнями, днями, часовими парами та версією розкладу. Використання MySQL дозволяє підтримувати цілісність цих зв'язків, виконувати складні вибірки та забезпечувати надійне зберігання інформації.

MySQL також добре інтегрується з PHP та Yii2, що було важливим чинником під час вибору інструментальних засобів. У межах модуля база даних використовується не лише для зберігання готового розкладу, а й для підтримки його версійності, обмежень, додаткових даних до занять, зв'язків із вибірковими дисциплінами та формування персоналізованого розкладу студента. Завдяки реляційній структурі можна уникнути дублювання інформації та забезпечити узгодженість даних між різними представленнями розкладу.

Для підвищення продуктивності системи використовується Redis[26;27]. Його основним призначенням у межах модуля є кешування персоналізованих розкладів студентів. Такий розклад формується не лише на основі академічної групи, а й з урахуванням вибіркових дисциплін, які обрав конкретний студент.

Отже, для різних користувачів результат може відрізнятися, а його формування потребує додаткових запитів до бази даних і обробки інформації. Кешування дозволяє зберігати вже підготовлений результат і швидше відображати його при повторному відкритті сторінки.

Використання Redis є особливо доцільним у періоди підвищеного навантаження на систему. Наприклад, на початку семестру велика кількість студентів і викладачів може одночасно переглядати розклад. Якщо кожен такий запит буде щоразу повністю формуватися з бази даних, це може створювати зайве навантаження на сервер. Завдяки кешуванню частина запитів обробляється швидше, а база даних використовується ефективніше. При цьому Redis не замінює основне сховище даних, а виконує допоміжну роль для прискорення доступу до часто використовуваної інформації.

На клієнтській стороні використовується JavaScript[24]. Його застосування необхідне для забезпечення інтерактивності інтерфейсу та зручної взаємодії користувача із системою. У модулі «Розклад занять» JavaScript може використовуватися для роботи модальних вікон, асинхронного оновлення даних, відкриття форм створення або редагування занять, обробки дій користувача без повного перезавантаження сторінки та покращення загальної зручності роботи з інтерфейсом. Це особливо важливо для сторінок, де користувач активно взаємодіє з розкладом, фільтрами, формами редагування та списковими представленнями.

Для розгортання та супроводу програмного забезпечення використовується Docker[25]. Контейнеризація дозволяє створити уніфіковане середовище виконання для вебзастосунку, бази даних, Redis та інших необхідних компонентів. Завдяки цьому зменшується кількість проблем, пов’язаних із відмінностями між локальним середовищем розробника, тестовим сервером і середовищем експлуатації. Docker також спрощує запуск системи, оновлення залежностей і перенесення проєкту між серверами.

Окремо варто зазначити роль FET/XML-файлу в роботі модуля. FET/XML не є основною технологією розробки, однак використовується як формат вхідних

даних для первинного формування розкладу[6;33]. Файл, сформований стороннім програмним забезпеченням, імпортується до системи, після чого дані проходять обробку, зіставлення з внутрішніми довідниками та зберігаються у відповідних таблицях бази даних. Такий підхід дозволяє поєднати можливості зовнішнього інструмента для генерації розкладу з внутрішніми механізмами Digital University UA для його подальшого редагування, контролю, версійності та відображення користувачам.

Для стабільної роботи модуля необхідно забезпечити відповідне серверне та клієнтське середовище. Оскільки основна обробка даних виконується на серверній стороні, найбільші вимоги висуваються саме до серверного обладнання. Воно повинно забезпечувати роботу вебзастосунку, бази даних MySQL, кеш-сховища Redis, Docker-контейнерів, а також підтримувати можливість резервного копіювання та журналювання роботи системи. Основні вимоги до апаратного та програмного забезпечення наведено в таблиці 1.1.

Таблиця 1.1.

Вимоги до апаратного та програмного забезпечення для роботи модуля «Розклад занять»

Компонент	Вимоги
Процесор сервера	Не менше 4 ядер, тактова частота від 2.5 GHz, підтримка багатопоточності та архітектури x86_64.
Оперативна пам’ять сервера	Не менше 16 GB RAM. Для одночасної роботи вебзастосунку, MySQL, Redis та Docker-контейнерів рекомендовано 32 GB RAM.
Дискова підсистема	SSD-накопичувач обсягом не менше 100 GB. Додатково має бути передбачено простір для журналів роботи системи, тимчасових файлів і резервних копій бази даних.
Операційна система сервера	Linux-середовище, зокрема Ubuntu Server, Debian або аналогічний серверний дистрибутив.
Середовище розгортання	Підтримка Docker та Docker Compose для контейнеризованого запуску компонентів системи.
Мережеве підключення	Стабільний доступ до мережі Інтернет із пропускною здатністю не менше 100 Mbps.

продовж. таблиці 1.1.

Резервне копіювання та моніторинг	Підтримка регулярного резервного копіювання бази даних, журналювання подій і моніторингу стабільності роботи системи.

Наведені вимоги є достатніми для стабільної роботи модуля «Розклад занять» у складі інформаційної системи Digital University UA. Вони враховують потребу в одночасній роботі кількох серверних компонентів, зберіганні значного обсягу структурованих даних, кешуванні персоналізованих розкладів та забезпеченні доступу користувачів через вебінтерфейс. Завдяки тому, що основна обробка інформації відбувається на сервері, клієнтські пристрої не потребують складного налаштування або встановлення спеціалізованого програмного забезпечення.

Обраний набір технологій повністю відповідає вимогам до розроблюваного модуля та забезпечує необхідний рівень продуктивності, надійності та масштабованості інформаційної системи. Використання PHP і Yii2 дозволяє реалізувати серверну логіку в межах наявної архітектури платформи, MySQL забезпечує надійне зберігання структурованих даних, Redis підвищує швидкодію під час формування персоналізованих розкладів, JavaScript покращує інтерактивність інтерфейсу, а Docker спрощує розгортання та супровід системи. У сукупності ці засоби створюють технологічну основу для стабільної роботи модуля та його подальшого розвитку.

Висновки до першого розділу

У першому розділі було проведено аналіз поставленого завдання, що дозволило визначити основну мету кваліфікаційної роботи та сформувати перелік ключових вимог до розроблюваного модуля. Особливу увагу приділено проблемі організації розкладу занять у закладах вищої освіти, оскільки цей процес потребує врахування значної кількості взаємопов’язаних даних, зокрема академічних груп, підгруп, викладачів, аудиторій, навчальних дисциплін, семестрів та версій розкладу. Коректна постановка задачі стала основою для

подальшого проєктування програмного рішення, здатного автоматизувати процеси формування, редагування, контролю та відображення розкладу.

Проведений аналіз аналогічних програмних рішень показав, що повноцінного аналога розроблюваного модуля, який би одночасно забезпечував інтеграцію з інформаційною системою університету, підтримку вибіркових дисциплін, персоналізоване відображення розкладу та контроль конфліктів, фактично не існує. Наявні рішення здебільшого покривають лише окремі аспекти поставленої задачі. Табличні редактори та статичні файли є простими у використанні, але не забезпечують автоматичної перевірки помилок, розмежування прав доступу та інтеграції з іншими модулями системи. Спеціалізовані інструменти для формування розкладу можуть бути корисними на етапі генерації, однак потребують додаткової інтеграції з внутрішньою інфраструктурою закладу освіти.

На основі аналізу вимог було обґрунтовано вибір архітектури модуля. Оскільки розробка здійснюється в межах існуючої інформаційної системи Digital University UA, доцільним є використання клієнт-серверної архітектури з архітектурним шаблоном MVC. Такий підхід забезпечує розділення логіки роботи системи, обробки запитів користувача та відображення даних у вебінтерфейсі. Додаткове використання сервісного шару дозволяє винести основну бізнес-логіку в окремі компоненти, що спрощує супровід, тестування та подальший розвиток модуля.

Також у розділі було обґрунтовано вибір інструментальних засобів розробки. Для реалізації серверної частини використовується мова PHP та фреймворк Yii2, який забезпечує підтримку MVC-архітектури, роботу з базою даних і зручну організацію структури проєкту. Для зберігання даних обрано MySQL, а для підвищення швидкодії під час формування персоналізованих розкладів студентів використовується Redis. Застосування Docker дає змогу спростити розгортання системи та забезпечити стабільність середовища виконання.

Таким чином, у першому розділі було сформовано концептуальну основу для подальшої розробки модуля «Розклад занять». Було визначено основні проблеми предметної області, проаналізовано часткові аналоги програмного продукту, обґрунтовано вибір архітектури, інструментальних засобів та вимог до апаратного забезпечення. Отримані результати створюють підґрунтя для наступного етапу роботи, пов’язаного з проєктуванням структури даних, логіки взаємодії компонентів та реалізацією функціональних можливостей модуля.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ОНЛАЙН-МЕНЕДЖМЕНТУ ОСВІТНІХ ПРОЦЕСІВ

2.1. Визначення варіантів використання та структури системи

Для забезпечення ефективної роботи модуля «Розклад занять» у межах інформаційної системи онлайн-менеджменту освітніх процесів Digital University UA було сформульовано низку вимог до його розробки[20]. Вони охоплюють функціональні та нефункціональні характеристики, необхідні для стабільної роботи системи, зручної взаємодії користувачів із розкладом, підтримки різних рівнів доступу та інтеграції з іншими складовими цифрової платформи університету.

До основних цілей розробки модуля можна віднести автоматизацію процесів формування, редагування, контролю та відображення розкладу навчальних занять у закладі вищої освіти. Модуль має забезпечити централізоване управління розкладом, зменшити кількість ручних операцій, підвищити актуальність даних та надати студентам, викладачам і адміністрації доступ до достовірної інформації про навчальні заняття. У межах роботи з розкладом важливо не лише сформувати таблицю занять, а й забезпечити її подальший супровід, перевірку, оновлення та коректне відображення для різних категорій користувачів.

Важливою особливістю модуля є підтримка різних способів перегляду розкладу. Система повинна дозволяти переглядати розклад за академічними групами, викладачами та аудиторіями. Крім того, необхідно враховувати підгрупи, вибіркові дисципліни, навчальні роки, семестри та версії розкладу. Це дає змогу адаптувати модуль до реальних умов організації освітнього процесу, де студент може мати не лише загальний розклад своєї академічної групи, а й окремі заняття за обраними вибірковими дисциплінами.

Модуль «Розклад занять» реалізується як складова частина веборієнтованої інформаційної системи Digital University UA. Користувачі взаємодіють із системою через вебінтерфейс за допомогою браузера, тому для роботи з модулем не потрібно встановлювати додаткове програмне забезпечення

на клієнтські пристрої. Такий підхід є зручним для закладу вищої освіти, оскільки забезпечує доступ до системи з різних пристроїв і дозволяє централізовано оновлювати функціональність без необхідності налаштування окремого програмного середовища для кожного користувача.

Серверна частина системи побудована на основі мови PHP та фреймворку Yii2. У межах розробки використовується архітектурний шаблон MVC, який передбачає розділення системи на моделі, представлення та контролери. Додатково в модулі використовується сервісний шар, що відповідає за формування табличного та спискового представлення розкладу, обробку імпортованих даних, перевірку конфліктів, експорт відфільтрованої інформації та кешування персоналізованих студентських розкладів. Така структура дозволяє розділити відповідальність між частинами системи та спростити подальший супровід програмного продукту.

Основні дані модуля зберігаються в реляційній базі даних MySQL. У ній розміщуються відомості про версії розкладу, заняття, академічні групи, викладачів, аудиторії, дисципліни, типи занять, навчальні тижні, часові проміжки, посилання, нотатки та обмеження. Для підвищення продуктивності під час відображення персоналізованого розкладу студентів використовується Redis, який дозволяє кешувати сформовані дані та зменшувати кількість повторних запитів до бази даних. Розгортання системи виконується в Docker-середовищі, що спрощує налаштування, тестування та підтримку програмного продукту.

Однією з основних потреб адміністративних користувачів є можливість централізованого управління розкладом занять. Система повинна надавати інструменти для перегляду розкладу за групами, викладачами та аудиторіями, а також забезпечувати створення, редагування, переміщення, копіювання та видалення окремих занять. Такий функціонал є необхідним для оперативного внесення змін у розклад, оскільки в реальних умовах навчального процесу часто виникає потреба змінити аудиторію, викладача, день проведення заняття або номер пари.

Важливою вимогою до модуля є підтримка роботи з версіями розкладу. Адміністрація повинна мати можливість створювати різні версії розкладу для відповідних семестрів, визначати актуальну версію, керувати її видимістю та доступністю для редагування. Це дозволяє готувати зміни без негайного оприлюднення та забезпечує контрольоване оновлення інформації для користувачів. Завдяки такому підходу студенти й викладачі бачать лише актуальний розклад, тоді як адміністративні працівники можуть окремо працювати з чернетками або майбутніми варіантами розкладу.

Система також повинна підтримувати імпорт розкладу із зовнішнього файлу, сформованого стороннім програмним забезпеченням для первинної генерації розкладу. Після імпорту дані мають зберігатися у внутрішній структурі Digital University UA, де вони можуть бути перевірені, змінені, доповнені та оприлюднені. Такий підхід дозволяє використовувати зовнішній інструмент для початкового формування розкладу, але подальше управління ним виконувати вже в межах єдиної інформаційної системи університету.

Окремою вимогою є наявність механізму контролю конфліктів[20]. Під час роботи з розкладом система повинна виявляти ситуації, коли одна академічна група, викладач або аудиторія одночасно задіяні в кількох заняттях. Також необхідно враховувати особливості підгруп, оскільки різні підгрупи однієї академічної групи можуть мати окремі заняття в один і той самий час, якщо це не суперечить правилам побудови розкладу. Контроль конфліктів дозволяє зменшити ризик помилок і підвищити якість підготовленого розкладу перед його публікацією.

Окремою адміністративною сторінкою модуля є сторінка «Контроль». Вона призначена для перегляду всіх пар розкладу з використанням фільтрів за семестром, версією розкладу, академічною групою, викладачем, аудиторією, дисципліною, днем, тижнем та номером пари. Основне призначення цієї сторінки полягає у швидкій перевірці коректності заповнення розкладу, пошуку потрібних занять, аналізі результатів імпорту та зручному експорті відфільтрованих даних. Важливо, що сторінка «Контроль» не використовується

для редагування посилань і нотаток до занять, оскільки її функціональність зосереджена саме на перевірці, фільтрації та експорті інформації.

Система повинна підтримувати додавання, редагування та відображення посилань і нотаток, прикріплених до окремих занять. Керування такими даними виконується безпосередньо в тих представленнях розкладу, де доступний списковий формат відображення, зокрема на сторінці розкладу викладача та на сторінці «Мій розклад». Це дозволяє працювати з додатковою інформацією до пари саме в контексті конкретного заняття. Посилання можуть використовуватися для переходу до онлайн-зустрічі, електронного курсу або іншого навчального ресурсу, а нотатки можуть містити службові уточнення щодо проведення заняття.

Крім того, адміністрація повинна мати можливість експортувати відфільтровані дані у форматі Excel зі сторінки «Контроль». Такий функціонал спрощує підготовку звітних матеріалів, перевірку інформації та використання розкладу структурними підрозділами університету. Завдяки фільтрам користувач може попередньо сформувати потрібну вибірку, наприклад за академічною групою, викладачем, аудиторією або конкретною версією розкладу, а потім зберегти її у табличному форматі.

Власники та технічні адміністратори системи очікують, що модуль буде інтегрований у загальну структуру Digital University UA та відповідатиме принципам безпечної й контрольованої роботи з даними. Система повинна підтримувати розмежування прав доступу відповідно до ролей користувачів, зокрема студентів, викладачів, завідувачів кафедр, деканів та адміністраторів. Такий підхід дозволяє обмежити доступ до критичних дій і забезпечити коректну роботу користувачів відповідно до їхніх повноважень.

Важливою вимогою з боку власників системи є можливість подальшого розвитку модуля без повної перебудови програмного продукту. Для цього функціональність має бути розділена на логічні компоненти, зокрема контролери, моделі, представлення та сервіси. Такий підхід дозволяє розширювати окремі частини модуля, не порушуючи роботу всієї платформи.

Також важливою є продуктивність системи, особливо при одночасному доступі великої кількості студентів до особистих кабінетів. Для цього передбачено використання кешування, яке дозволяє швидко відображати персоналізований розклад та зменшувати навантаження на базу даних.

Об’єктом комп’ютеризації є процес управління розкладом навчальних занять у закладі вищої освіти. Він охоплює створення розкладу, імпорт сформованих даних, редагування окремих занять, перевірку конфліктів, затвердження актуальної версії, експорт інформації та відображення розкладу для різних категорій користувачів. Основними користувачами модуля є студенти, викладачі, завідувачі кафедр, декани та адміністратори. Студенти використовують модуль переважно для перегляду персонального розкладу в особистому кабінеті. Викладачі отримують доступ до власного розкладу та, залежно від наданих прав, можуть редагувати окремі дані. Завідувачі кафедр і декани мають розширені можливості перегляду та керування розкладом підлеглих структурних підрозділів, а адміністратори відповідають за імпорт, налаштування, редагування, перевірку та публікацію розкладу.

До функціональних вимог модуля належить забезпечення авторизації користувачів і доступу до функціоналу відповідно до їхніх ролей. Для студентів має бути доступний перегляд персонального розкладу, який формується на основі загального розкладу академічної групи та вибіркових дисциплін, обраних конкретним студентом. Для адміністративних користувачів система повинна забезпечувати перегляд розкладу за академічними групами, викладачами та аудиторіями, а також створення нових занять, редагування існуючих, переміщення занять в інший часовий проміжок, копіювання та видалення пар.

Система повинна підтримувати імпорт розкладу із FET/XML-файлу, сформованого стороннім програмним забезпеченням для генерації розкладу. Імпортовані дані мають перетворюватися у внутрішню структуру системи та зберігатися у відповідних таблицях бази даних. Також система повинна виконувати перевірку імпортованого розкладу на відповідність попередньо встановленим обмеженням. До таких обмежень належать заборонені або

небажані часові проміжки для академічних груп, викладачів та аудиторій. Якщо під час імпорту FET/XML-файлу система виявляє, що певне заняття порушує встановлені правила, користувач повинен отримати повідомлення про помилку або попередження.

Крім відображення повідомлення про помилку в інтерфейсі, система повинна надавати можливість експортувати перелік знайдених помилок у текстовий файл формату .txt. Це дозволяє зберегти результати перевірки, передати їх відповідальним особам або використати для подальшого виправлення початкового FET/XML-файлу. Після усунення виявлених порушень файл може бути повторно імпортований у систему.

Необхідно також забезпечити роботу з версіями розкладу. Користувачі з відповідними правами повинні мати можливість створювати версії, визначати поточну версію, налаштовувати її видимість та можливість редагування. Це дозволяє відокремити робочі варіанти розкладу від опублікованої інформації. Окремою функціональною вимогою є контроль конфліктів, за якого система повинна виявляти накладання занять для академічних груп, викладачів та аудиторій, а для підгруп додатково враховувати, що заняття для різних підгруп можуть проводитися одночасно, якщо вони не перетинаються із заняттям для всієї групи.

Система повинна підтримувати два формати відображення розкладу: табличний та списковий. Табличне представлення є основним і використовується на більшості сторінок модуля, оскільки дозволяє зручно переглядати заняття за днями, тижнями та часовими парами. Спискове представлення є додатковим форматом і використовується в особистому кабінеті студента, на сторінці перегляду розкладу викладачів та на сторінці «Мій розклад». Саме в спискових представленнях доступна робота з посиланнями й нотатками до занять.

Модуль також повинен забезпечувати експорт відфільтрованих даних у форматі Excel зі сторінки «Контроль». Це необхідно для подальшої перевірки розкладу, аналізу навантаження та використання інформації структурними

підрозділами університету. Оскільки експорт виконується після застосування фільтрів, адміністративний користувач може отримати не весь розклад повністю, а лише ту частину даних, яка потрібна для конкретного завдання.

Нефункціональні вимоги до модуля стосуються зручності сприйняття, надійності, продуктивності та можливості подальшої експлуатації. Інтерфейс модуля повинен бути зрозумілим для користувачів із різним рівнем технічної підготовки. Основні дії, зокрема перегляд розкладу, вибір групи, викладача або аудиторії, повинні виконуватися без додаткового навчання. Форми редагування занять мають бути структурованими та містити зрозумілі поля для вибору дисципліни, типу заняття, групи, викладача, аудиторії, тижня, дня та номера пари.

Надійність системи забезпечується стабільною роботою під час внесення змін до розкладу. Операції створення, редагування, переміщення та копіювання занять мають виконуватися з перевіркою коректності даних. У разі помилки система повинна повідомляти користувача про причину невдалого виконання операції. Для зменшення ризику некоректного стану даних важливі операції мають виконуватися з використанням транзакцій, що дозволяє скасувати зміни у випадку виникнення помилки.

Продуктивність модуля має забезпечувати швидке відображення розкладу для студентів, викладачів та адміністрації. Оскільки персоналізований розклад студента може формуватися з урахуванням загального розкладу групи та вибіркових дисциплін, для оптимізації цього процесу використовується кешування. Це дозволяє зменшити кількість повторних запитів до бази даних і забезпечити комфортну роботу користувачів навіть за значної кількості одночасних звернень.

Можливість подальшої експлуатації та розвитку модуля забезпечується його структурною організацією. Використання MVC-архітектури, сервісного шару та реляційної бази даних дозволяє розділити відповідальність між компонентами системи. Це спрощує внесення змін, тестування окремих частин

функціоналу та розширення модуля новими можливостями без повної перебудови програмного продукту.

Після визначення основних характеристик майбутнього програмного забезпечення та виокремлення функцій, необхідних для кожної категорії користувачів, було створено діаграму варіантів використання модуля «Розклад занять» (рис. 2.1).

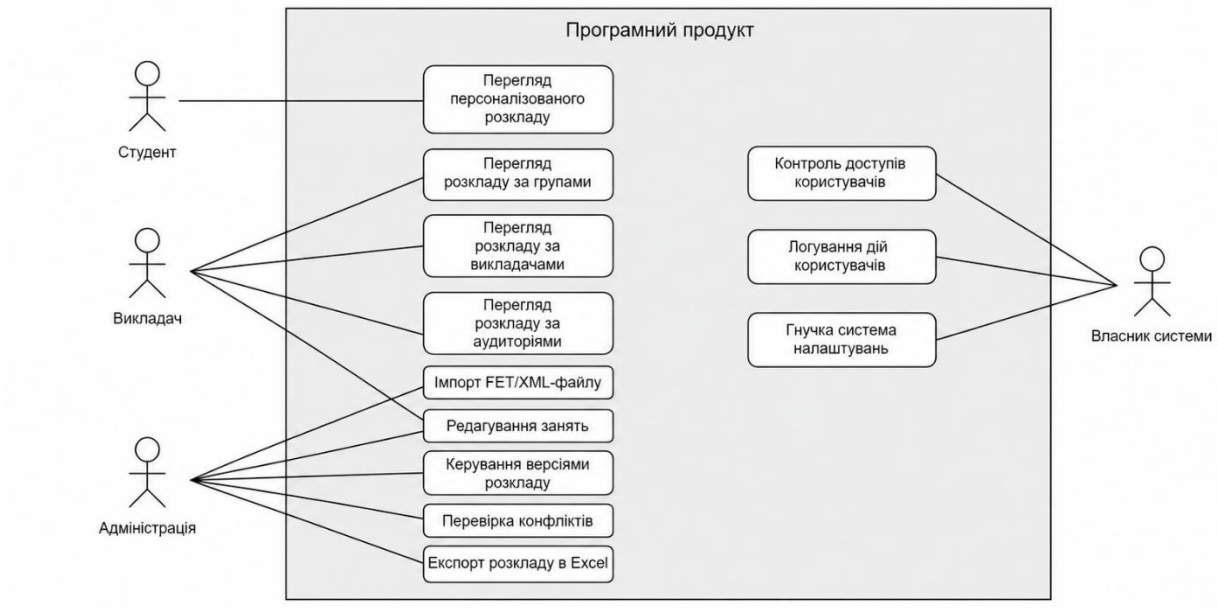

Рис. 2.1 – Діаграма варіантів використання модуля «Розклад занять»

На діаграмі варіантів використання відображено основні категорії користувачів, які взаємодіють із модулем «Розклад занять»: студента, викладача, завідувача кафедри, декана, адміністратора та власника системи. Для студента головним сценарієм роботи є перегляд персонального розкладу, який формується на основі загального розкладу академічної групи та вибіркових дисциплін, обраних конкретним здобувачем освіти. Викладач отримує доступ до власного розкладу через сторінку «Мій розклад» і, за наявності відповідних прав, може виконувати окремі дії з заняттями, посиланнями та нотатками.

Завідувач кафедри та декан мають розширені можливості перегляду й контролю розкладів підлеглих працівників або структурних підрозділів. Адміністратор відповідає за імпорт розкладу з FET/XML-файлу, керування версіями, створення та редагування занять, перевірку конфліктів, налаштування

обмежень, роботу зі сторінкою «Контроль» і експорт відфільтрованих даних. Власник або технічний адміністратор системи виконує функції, пов’язані з керуванням доступами, підтримкою стабільності роботи системи, налаштуванням середовища та контролем подальшого розвитку програмного продукту.

2.2. Обґрунтування вибору системи управління базами даних

У процесі реалізації модуля «Розклад занять» інформаційної системи Digital University UA важливим етапом стало обґрунтування вибору системи керування базами даних. Для такого програмного продукту база даних є не лише сховищем інформації, а й основою для забезпечення коректної роботи всієї бізнес-логіки. Саме в базі даних зберігаються відомості про навчальні роки, семестри, версії розкладу, академічні групи, викладачів, аудиторії, навчальні дисципліни, типи занять, тижні, дні, часові проміжки, вибіркові дисципліни, обмеження, посилання та нотатки до занять. Тому обрана система керування базами даних повинна бути надійною, продуктивною, зручною для інтеграції з серверною частиною застосунку та достатньо гнучкою для подальшого розширення функціональності.

Особливістю предметної області є те, що розклад навчальних занять має чітко структуровану реляційну природу. Кожне заняття не існує ізольовано, а пов’язане з великою кількістю інших сутностей. Наприклад, одна пара може належати до конкретної версії розкладу, мати визначений день, навчальний тиждень і номер пари, бути пов’язаною з певною дисципліною, типом заняття, викладачем, академічною групою та аудиторією. При цьому одне заняття може стосуватися кількох груп або підгруп, проводитися кількома викладачами чи використовувати більше ніж одну аудиторію. Така структура потребує збереження складних зв’язків між даними, що робить реляційну модель найбільш доцільною для реалізації модуля.

Для зберігання основної інформації в межах модуля було обрано MySQL. Ця система керування базами даних добре підходить для веборієнтованих

інформаційних систем, оскільки забезпечує надійне зберігання структурованих даних, підтримує зв'язки між таблицями, індексування, транзакції та виконання складних запитів. У контексті модуля «Розклад занять» такі можливості є особливо важливими, оскільки система повинна не лише відображати готовий розклад, а й підтримувати його створення, імпортування, редагування, перевірку конфліктів, роботу з версіями та формування персоналізованих представлень для різних користувачів.

Однією з основних причин вибору MySQL є її ефективна інтеграція з мовою PHP та фреймворком Yii2, які використовуються для реалізації серверної частини Digital University UA. Yii2 надає зручний механізм роботи з базою даних через Active Record, що дозволяє описувати таблиці у вигляді моделей, визначати зв'язки між ними та виконувати основні операції з даними без дублювання SQL-запитів у різних частинах програмного коду. Для модуля «Розклад занять» це є важливою перевагою, оскільки структура бази даних містить значну кількість пов'язаних сутностей, а робота з ними повинна залишатися зрозумілою та зручною для подальшого супроводу.

У межах модуля база даних використовується для зберігання як основних, так і допоміжних сутностей. Центральною сутністю є навчальне заняття, що зберігає основну інформацію про пару, її належність до певної версії розкладу, навчальний тиждень, день, часовий проміжок, дисципліну та тип заняття. При цьому зв'язки заняття з академічними групами, викладачами та аудиторіями винесені в окремі проміжні таблиці. Такий підхід дозволяє уникнути надмірного дублювання даних і підтримати гнучкі сценарії, коли одна пара може бути пов'язана з кількома учасниками освітнього процесу.

Окреме значення має підтримка версійності розкладу. У межах розробленого модуля версія розкладу відповідає конкретній редакції розкладу для певного семестру. Для цього використовується таблиця schedule_version, яка дозволяє створювати кілька версій розкладу, визначати активну версію, керувати її видимістю та можливістю редагування. Це важливо для реального процесу адміністрування, оскільки розклад може змінюватися до початку семестру або

впродовж навчального періоду. Адміністрація може підготувати нову версію, перевірити її, внести необхідні зміни й лише після цього зробити доступною для студентів і викладачів. Завдяки цьому користувачі бачать актуальну інформацію, а робота з чернетками не порушує стабільність опублікованого розкладу.

Окремо від версій розкладу в базі даних передбачено поняття варіанта розкладу. Варіант розкладу не є чернеткою або опублікованою редакцією розкладу, а описує шаблон організації навчального часу. Для цього використовуються таблиці schedule_variant, schedule_week, schedule_hour та schedule_variant_template. Вони дозволяють налаштовувати навчальні тижні, часові пари та логіку чергування тижнів у межах семестру. Наприклад, для одного закладу освіти може використовуватися стандартна двотижнева система, тоді як для іншого можуть бути налаштовані інші назви тижнів, інша кількість пар упродовж дня або інша логіка чергування навчальних тижнів. Таким чином, schedule_version відповідає за конкретні версії розкладу, а schedule_variant — за шаблон організації навчального часу.

Також база даних підтримує роботу з вибірковими дисциплінами. У системі передбачено окремі сутності для збереження вибіркових компонентів, потоків, зв’язків студентів із вибраними дисциплінами та занять, що належать до вибіркового блоку. Завдяки цьому персоналізований розклад студента формується не лише на основі академічної групи, а й з урахуванням індивідуального вибору дисциплін. Це дозволяє не відображати студенту всі можливі вибіркові заняття, а показувати лише ті, які безпосередньо стосуються його освітньої траєкторії.

MySQL також є доцільним вибором для реалізації механізмів контролю конфліктів. Оскільки інформація про заняття, групи, викладачів та аудиторії зберігається у взаємопов’язаних таблицях, система може перевіряти накладання за однаковими часовими параметрами. Наприклад, конфлікт може бути виявлений тоді, коли одна академічна група має кілька занять в один і той самий час, коли викладач призначений на різні пари одночасно або коли одна аудиторія використовується для кількох занять у межах одного часового проміжку.

Реляційна структура дозволяє виконувати такі перевірки за допомогою запитів до пов'язаних таблиць і враховувати додаткові параметри, зокрема версію розкладу, день, навчальний тиждень, номер пари та підгрупу.

Окрему роль у базі даних відіграють таблиці обмежень. Вони дозволяють визначати небажані або заборонені часові проміжки для академічних груп, викладачів та аудиторій. Такі обмеження можуть враховуватися під час перевірки імпортованого розкладу та допомагають виявляти ситуації, коли заняття призначено на час, який не відповідає встановленим правилам. Завдяки збереженню цих правил у базі даних адміністративний користувач може керувати ними без необхідності змінювати програмний код.

Крім основних даних розкладу, база даних забезпечує збереження додаткової інформації до занять. До такої інформації належать посилання та нотатки, які можуть бути прикріплені до окремої пари. Посилання можуть використовуватися для переходу до онлайн-зустрічі, електронного курсу або іншого навчального ресурсу, а нотатки можуть містити службові уточнення чи додаткові пояснення щодо проведення заняття. Зберігання таких даних в окремих пов'язаних таблицях дозволяє не перевантажувати основну таблицю занять і водночас забезпечити гнучку роботу з додатковою інформацією у спискових представленнях розкладу.

Важливою вимогою до бази даних є забезпечення цілісності інформації[16;17]. У модулі «Розклад занять» зміна одного запису може впливати на декілька пов'язаних сутностей. Наприклад, під час редагування пари може змінюватися її зв'язок із викладачем, аудиторією або академічною групою. У разі некоректного виконання таких операцій могли б виникати неузгодженості, коли основний запис заняття існує, але пов'язані дані відсутні або, навпаки, залишаються зайві записи у проміжних таблицях. Використання MySQL дає змогу застосовувати транзакційний підхід до таких операцій, що підвищує надійність роботи системи[14;28].

Реляційна структура також сприяє зручному формуванню різних представлень розкладу. Оскільки всі дані зберігаються в єдиній узгодженій

структурі, система може формувати розклад за академічною групою, викладачем або аудиторією без дублювання записів. У кожному випадку змінюється лише основний критерій вибірки, тоді як базова інформація про заняття залишається спільною. Це є важливою перевагою, оскільки одна зміна в розкладі автоматично відображається в усіх пов'язаних представленнях.

Для підвищення продуктивності системи разом із MySQL використовується Redis[26;27;28]. При цьому Redis не замінює основну базу даних, а виконує роль допоміжного інструмента для кешування. Основні структуровані дані зберігаються саме в MySQL, тоді як Redis використовується для тимчасового збереження сформованих персоналізованих розкладів студентів. Такий підхід дозволяє пришвидшити повторне відкриття розкладу в особистому кабінеті та зменшити кількість запитів до основної бази даних.

Необхідність використання кешування зумовлена тим, що персоналізований розклад студента може формуватися на основі кількох джерел даних. Система повинна враховувати загальний розклад академічної групи, вибрані студентом дисципліни, активну версію розкладу, тижні, дні та часові пари. Якщо кожне відкриття сторінки призводитиме до повторного виконання всіх запитів і формування розкладу з нуля, це може створювати зайве навантаження на базу даних. Використання Redis дозволяє зберігати вже підготовлений результат і повертати його користувачу швидше.

Розгортання бази даних виконується в Docker-середовищі. Це дозволяє забезпечити однакові умови роботи системи на різних етапах розробки, тестування та експлуатації. Контейнеризація спрощує налаштування середовища, підключення застосунку до бази даних і подальше супроводження програмного продукту. Крім того, використання Docker зменшує ризик виникнення проблем, пов'язаних із різними версіями програмного забезпечення на локальному комп'ютері розробника та сервері.

У межах етапу проєктування було визначено основні сутності модуля, їхні атрибути та взаємозв'язки. Логічна структура бази даних відображає зв'язки між версіями розкладу, заняттями, академічними групами, викладачами,

аудиторіями, часовими проміжками, обмеженнями, посиланнями, нотатками та вибірковими дисциплінами. Така структура забезпечує цілісність даних, зручність їх обробки та можливість подальшого розширення функціональності системи.

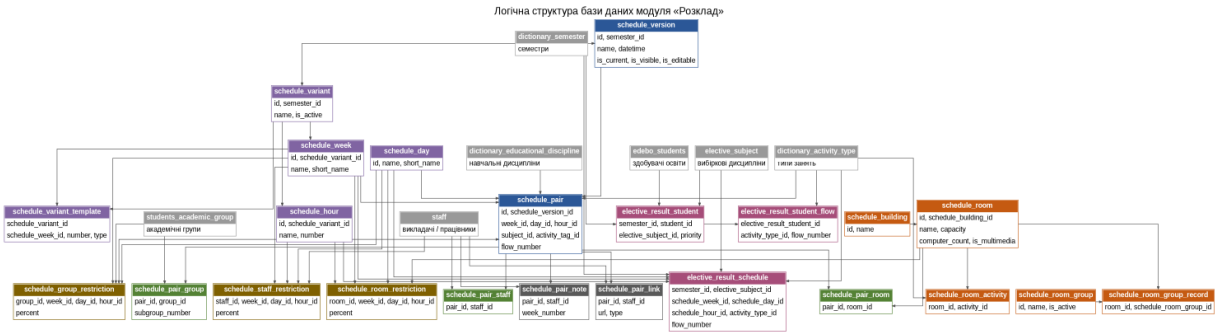

Рис. 2.2 – Логічна структура бази даних модуля «Розклад занять»

Таким чином, використання MySQL як основної системи керування базами даних є доцільним для розроблюваного модуля. Вона забезпечує надійне зберігання структурованої інформації, підтримку зв’язків між сутностями, можливість контролю цілісності даних, виконання складних запитів та ефективну інтеграцію з серверною частиною системи. У поєднанні з Redis для кешування та Docker для розгортання обрана технологічна основа відповідає вимогам до продуктивності, надійності, масштабованості й подальшого розвитку модуля «Розклад занять» у складі Digital University UA.

2.3. Розробка бази даних системи

Головною особливістю бази даних модуля «Розклад занять» є необхідність не лише зберігати готовий розклад занять, а й забезпечувати коректне перенесення даних із зовнішнього джерела у внутрішню структуру інформаційної системи Digital University UA. Оскільки первинне формування одного з варіантів розкладу виконується за допомогою стороннього програмного забезпечення, у модулі передбачено механізм імпорту FET/XML-файлу, який містить інформацію про заняття, викладачів, групи, аудиторії, дні, години та інші параметри розкладу.

Розклад, отриманий із зовнішнього файлу, не може використовуватися системою без попередньої обробки, оскільки дані у файлі мають бути зіставлені

з внутрішніми сутностями бази даних. Зокрема, назви академічних груп, викладачів, аудиторій, дисциплін і типів занять повинні бути пов'язані з відповідними записами в таблицях системи. Це необхідно для того, щоб надалі розклад можна було редагувати, перевіряти на конфлікти, відображати у різних представленнях та використовувати для формування персоналізованого розкладу студента.

Діаграма послідовності імпортування розкладу з FET/XML-файлу зображена на рисунку 2.3.

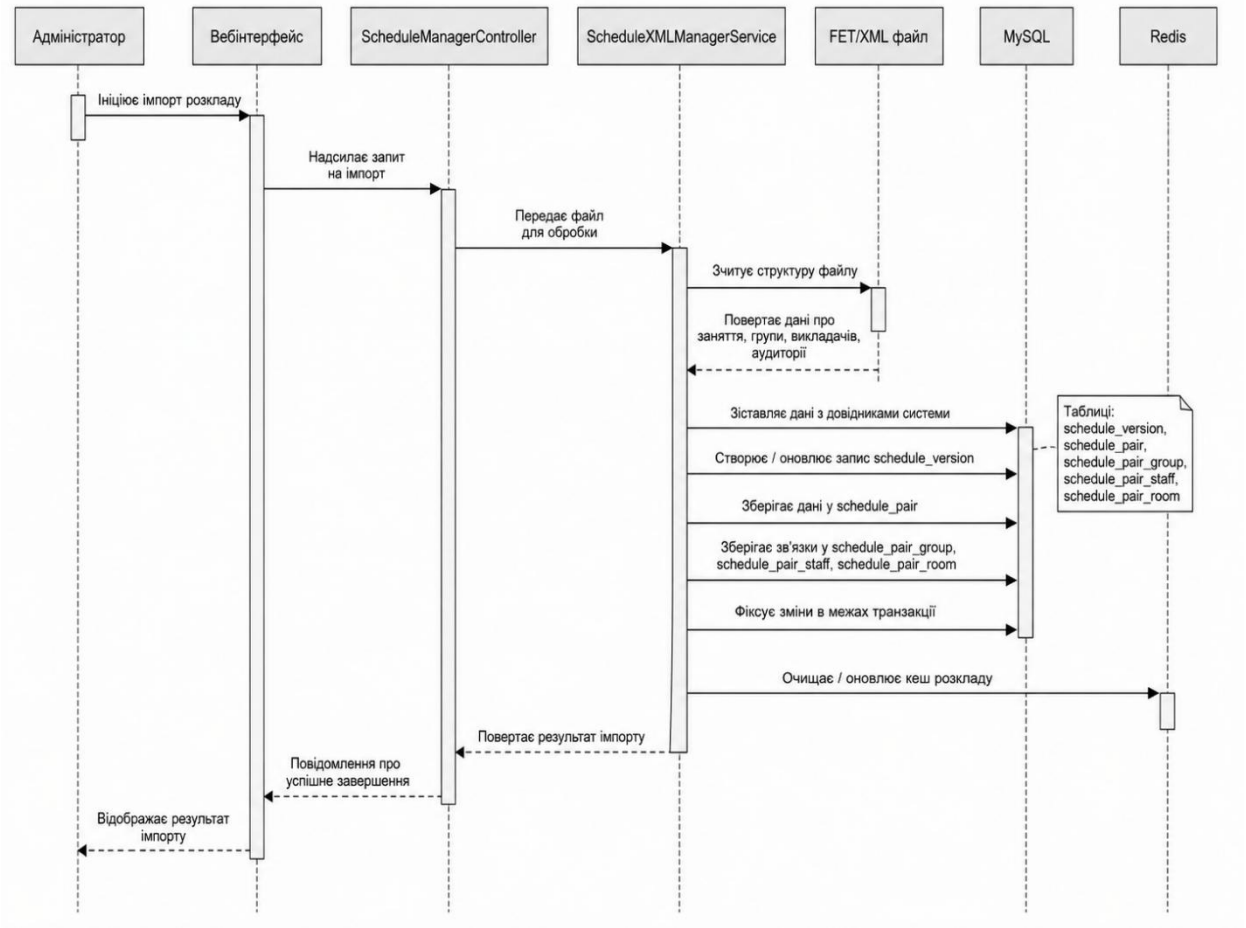

Рис. 2.3 – Діаграма послідовності імпортування розкладу з FET/XML-файлу

Процес імпорту розкладу починається з ініціації дії користувачем, який має відповідні права доступу до адміністративної частини системи. Користувач обирає файл або результат генерації розкладу, після чого запит передається до контролера, відповідального за управління імпортом. На цьому етапі система перевіряє наявність файлу, його формат та можливість подальшої обробки.

Після цього файл передається до сервісного шару, який відповідає за аналіз XML-структури та перетворення зовнішніх даних у формат, придатний для збереження в базі даних. Сервіс зчитує інформацію про заняття, дні, години, викладачів, групи та аудиторії, після чого формує внутрішні масиви даних для подальшої обробки. Такий підхід дозволяє відокремити логіку зчитування файлу від логіки збереження інформації в базу даних.

На наступному етапі виконується зіставлення отриманих із файлу значень із записами, які вже існують у базі даних системи. Наприклад, назва академічної групи зіставляється з відповідним записом у таблиці академічних груп, прізвище або ідентифікатор викладача — із записом у таблиці працівників, а назва аудиторії — з таблицею аудиторій. Аналогічно відбувається визначення навчальної дисципліни, типу заняття, дня тижня, навчального тижня та часу проведення пари.

Основна інформація про кожне заняття зберігається в таблиці schedule_pair. У ній фіксується належність заняття до певної версії розкладу, тиждень, день, час проведення, дисципліна, тип заняття та номер потоку. Оскільки одне заняття може бути пов’язане з кількома групами, викладачами або аудиторіями, ці зв’язки зберігаються не в основній таблиці заняття, а в окремих проміжних таблицях.

Зв’язок заняття з академічними групами зберігається в таблиці schedule_pair_group. Вона також містить інформацію про номер підгрупи, що дає змогу підтримувати окремі заняття для різних підгруп однієї академічної групи. Зв’язок із викладачами зберігається в таблиці schedule_pair_staff, а зв’язок з аудиторіями — у таблиці schedule_pair_room. Така структура дозволяє уникнути дублювання даних і забезпечує гнучкість при подальшому редагуванні розкладу.

Окремою особливістю структури бази даних є підтримка налаштовуваних варіантів організації навчального часу. У цьому контексті варіант розкладу не означає окрему версію або чернетку розкладу. Він визначає шаблон, за яким організовано навчальні тижні, часові пари та їх чергування в межах семестру.

Для цього використовуються таблиці schedule_variant, schedule_week, schedule_hour та schedule_variant_template.

Таблиця schedule_variant описує окремий варіант організації навчального часу. Таблиці schedule_week та schedule_hour дозволяють визначати набір навчальних тижнів і часових пар у межах відповідного варіанта. Завдяки цьому система не прив'язана до жорстко заданої структури навчального процесу, а може бути адаптована до потреб конкретного закладу освіти.

Таблиця schedule_variant_template використовується для опису шаблону тижневості семестру. Вона дозволяє зіставляти фактичні календарні тижні з логічними тижнями розкладу. На відміну від цього, таблиця schedule_version використовується для збереження конкретних версій розкладу, які можуть бути активними, видимими або доступними для редагування.

Для збереження додаткової інформації, пов'язаної з окремими заняттями, у базі даних передбачено таблиці schedule_pair_note та schedule_pair_link. Таблиця schedule_pair_note використовується для збереження текстових нотаток до пари, які можуть містити службові пояснення, уточнення щодо заняття або іншу додаткову інформацію. Таблиця schedule_pair_link призначена для збереження посилань, пов'язаних із заняттям, наприклад на онлайн-зустріч, електронний курс або навчальні матеріали. Обидві таблиці пов'язуються з основною таблицею schedule_pair, що дозволяє прикріплювати додаткові дані без ускладнення структури основного запису заняття.

Перед збереженням або під час подальшого редагування розкладу система може перевіряти коректність даних. Зокрема, важливим є виявлення ситуацій, коли одна аудиторія, викладач або академічна група використовуються одночасно в кількох заняттях. Для цього аналізуються записи з таблиць schedule_pair, schedule_pair_group, schedule_pair_staff та schedule_pair_room з урахуванням версії розкладу, тижня, дня та номера пари.

Окрему групу таблиць у структурі бази даних становлять таблиці обмежень розкладу. До них належать schedule_group_restriction, schedule_staff_restriction та schedule_room_restriction. Вони використовуються

для зберігання правил, які визначають можливість або небажаність проведення занять у певні часові проміжки для конкретних академічних груп, викладачів або аудиторій.

Таблиця schedule_group_restriction містить обмеження, пов’язані з академічними групами. Вона дозволяє вказати, що певна група не повинна мати заняття у визначений тиждень, день або номер пари. Такі дані можуть використовуватися під час перевірки розкладу, щоб уникнути призначення занять у непридатні для групи часові проміжки.

Таблиця schedule_staff_restriction призначена для збереження обмежень викладачів. Вона дозволяє враховувати ситуації, коли певний викладач не може проводити заняття у конкретний час. Це може бути пов’язано з особливостями навантаження, внутрішніми організаційними правилами або іншими обставинами, які потрібно враховувати під час формування розкладу.

Таблиця schedule_room_restriction використовується для збереження обмежень аудиторій. За її допомогою можна визначити часові проміжки, у які аудиторія не повинна використовуватися для проведення занять. Це може бути необхідно у випадках ремонту, технічного обслуговування, зайнятості аудиторії іншими заходами або інших організаційних причин.

Під час імпорту FET/XML-файлу система може зіставляти дані про заняття з наявними обмеженнями. Для цього аналізуються такі параметри, як навчальний тиждень, день, номер пари, академічна група, викладач та аудиторія. Якщо імпортований запис порушує обмеження, система фіксує відповідну помилку та відображає її адміністративному користувачу. Перелік таких помилок може бути експортований у текстовий файл, що спрощує подальше виправлення розкладу.

Таким чином, таблиці обмежень є важливою частиною бази даних модуля «Розклад занять», оскільки вони дозволяють не лише зберігати готовий розклад, а й контролювати його відповідність установленим правилам. Їх використання підвищує якість імпортованих даних, зменшує кількість ручних перевірок і допомагає виявляти проблеми ще до публікації розкладу для користувачів.

Під час роботи з розкладом система також може змінювати пов’язані з конкретним заняттям посилання та нотатки. Ці дані зберігаються в окремих таблицях і пов’язуються з основним записом пари. Їх редагування виконується через сторінки, де доступне спискове представлення розкладу, зокрема через розклад у вигляді списку, розклад викладача або сторінку «Мій розклад». Після внесення змін оновлена інформація відображається у відповідних представленнях розкладу.

У процесі імпорту також враховується версійність розкладу. Дані не просто додаються до загального набору занять, а прив’язуються до конкретної версії, що зберігається в таблиці schedule_version. Це дозволяє мати кілька версій розкладу для одного семестру, порівнювати їх, редагувати окремі версії та публікувати лише актуальну версію.

Після успішного збереження змін у базі даних система виконує інвалідацію кешованих записів, пов’язаних із відповідними академічними групами. Оскільки персоналізовані розклади студентів формуються на основі розкладу групи та вибіркових дисциплін, після зміни заняття застарілі кешовані дані мають бути видалені. Для цього використовується груповий тег кешу, який пов’язує академічну групу з кешованими розкладами студентських когорт. Під час наступного відкриття сторінки розклад формується повторно вже на основі актуальних даних із бази.

Розроблена структура бази даних дозволяє не лише імпортувати розклад, а й виконувати його подальше редагування в межах вебінтерфейсу. При зміні окремого заняття оновлюється запис у таблиці schedule_pair, а також за потреби змінюються записи у таблицях зв’язків із групами, викладачами та аудиторіями. Це забезпечує цілісність даних і дозволяє системі підтримувати актуальний стан розкладу після кожної операції.

Окрему роль у базі даних відіграють таблиці, пов’язані з вибірковими дисциплінами. Вони використовуються для формування персоналізованого розкладу студента. Загальний розклад академічної групи доповнюється заняттями з тих вибіркових дисциплін, які були обрані конкретним студентом.

Для цього система використовує інформацію про вибір студента, його потік і часові слоти занять за вибірковими дисциплінами.

Таким чином, розробка бази даних модуля «Розклад занять» передбачала створення структури, здатної підтримувати імпорт зовнішніх даних, збереження різних версій розкладу, зв'язок занять із групами, викладачами та аудиторіями, перевірку конфліктів, підтримку підгруп і вибіркових дисциплін. Запропонований підхід забезпечує цілісність даних, спрощує подальше редагування розкладу та створює основу для формування актуального і персоналізованого розкладу для різних категорій користувачів.

2.4. Реалізація функціоналу системи управління навчальним процесом

Розглянемо основні фрагменти реалізації модуля «Розклад занять» інформаційної системи Digital University UA. Оскільки модуль є частиною вже існуючої вебплатформи, його функціонал реалізовано відповідно до архітектурного підходу MVC, що використовується у фреймворку Yii2. Контролери відповідають за обробку запитів користувача, моделі — за взаємодію з базою даних, а сервісний шар містить основну бізнес-логіку, пов'язану з імпортом, формуванням, редагуванням, перевіркою та відображенням розкладу.

Одним із ключових процесів у модулі є імпорт розкладу з FET/XML-файлу. Такий файл формується стороннім програмним забезпеченням і містить первинний варіант розкладу, який надалі переноситься у внутрішню структуру Digital University UA. Для обробки XML-файлу використовується сервіс ScheduleXMLManagerService, який відповідає за зчитування файлу, перетворення його структури та передачу підготовлених даних у репозиторій для подальшого збереження.

Скорочений фрагмент методу імпорту має такий вигляд:

```
public static function prepareTeachersXMLData($filePath, $semesterId):
mixed
{
    $data = self::parseFile($filePath);
    $result                                                          =
ScheduleXMLManagerService::transformTeacherSchedule($data);
```

```
    $repository = new ScheduleRepository();

    return          $repository->saveMultipleTeachersSchedule($result,
$semesterId);
}
```

У наведеному фрагменті спочатку виконується зчитування XML-файлу за допомогою методу parseFile(). Після цього отримані дані перетворюються у внутрішній формат, зручний для подальшого збереження. Завершальним етапом є передача сформованого масиву до репозиторію, який виконує запис розкладу до бази даних.

Під час імпорту FET/XML-файлу система не лише зчитує та зберігає дані розкладу, а й виконує перевірку відповідності імпортованих занять встановленим обмеженням. Такі обмеження зберігаються в таблицях schedule_group_restriction, schedule_staff_restriction та schedule_room_restriction і дозволяють контролювати допустимість призначення занять для певних груп, викладачів або аудиторій у конкретні часові проміжки.

Якщо під час імпорту виявляється, що заняття порушує одне з таких обмежень, система формує повідомлення про помилку. Це повідомлення відображається в інтерфейсі сторінки імпорту, а також може бути збережене у вигляді текстового звіту. Такий підхід дозволяє адміністративному користувачу не лише побачити факт помилки, а й отримати детальну інформацію для подальшого виправлення початкового файлу розкладу.

У спрощеному вигляді логіку перевірки обмежень під час імпорту доцільно подати кількома невеликими фрагментами. Спочатку можна виділити окремий допоміжний метод, який перевіряє наявність обмеження для певної сутності розкладу. Це дозволяє не дублювати однакову логіку для академічних груп, викладачів та аудиторій.

```
private function hasRestriction(string $modelClass, array $params):
bool
{
    return $modelClass::find()
        ->where($params)
        ->exists();
}
```

У цьому фрагменті метод `hasRestriction()` приймає назву моделі обмеження та набір параметрів для пошуку. Завдяки цьому одна й та сама логіка може використовуватися для таблиць `schedule_group_restriction`, `schedule_staff_restriction` та `schedule_room_restriction`. Такий підхід робить код компактнішим і зменшує кількість повторюваних запитів у різних частинах методу імпорту.

Після цього для кожної імпортованої пари формується набір часових параметрів, за якими виконується перевірка. До таких параметрів належать навчальний тиждень, день та номер пари.

```
$timeParams = [
    'week_id' => $pair['week_id'],
    'day_id' => $pair['day_id'],
    'hour_id' => $pair['hour_id'],
];
```

Ці параметри є спільними для всіх типів обмежень, оскільки будь-яке обмеження в розкладі прив’язується до конкретного часового проміжку. Далі до них додається ідентифікатор відповідної сутності: академічної групи, викладача або аудиторії.

```
$groupRestricted = $this->hasRestriction(
    ScheduleGroupRestriction::class,
    array_merge($timeParams, ['group_id' => $pair['group_id']])
);

$staffRestricted = $this->hasRestriction(
    ScheduleStaffRestriction::class,
    array_merge($timeParams, ['staff_id' => $pair['staff_id']])
);

$roomRestricted = $this->hasRestriction(
    ScheduleRoomRestriction::class,
    array_merge($timeParams, ['room_id' => $pair['room_id']])
);
```

У цьому фрагменті система окремо перевіряє, чи існують обмеження для групи, викладача або аудиторії. Якщо хоча б одна з перевірок повертає позитивний результат, це означає, що імпортоване заняття порушує попередньо встановлені правила побудови розкладу.

```
if ($groupRestricted || $staffRestricted || $roomRestricted) {
    $errors[] = 'Виявлено порушення обмежень для заняття: '
        . $pair['subject_name'];
}
```

У результаті повідомлення про помилку додається до загального масиву помилок імпорту. У реальній реалізації таке повідомлення може містити не лише назву дисципліни, а й академічну групу, викладача, аудиторію, день, тиждень і номер пари. Це спрощує подальший аналіз проблеми та дозволяє швидше виправити початковий FET/XML-файл.

Після завершення перевірки система аналізує сформований перелік помилок. Якщо він не є порожнім, користувачу відображається повідомлення про проблеми імпорту. Додатково передбачено можливість експорту цих помилок у текстовий файл формату .txt, що дозволяє зберегти результати перевірки та використати їх для подальшого коригування FET/XML-файлу.

Скорочений приклад формування текстового файлу з помилками можна подати так:

```
if (!empty($errors)) {
    $content = implode(PHP_EOL, $errors);
    $fileName = 'schedule_import_errors.txt';

    Yii::$app->response->sendContentAsFile(
        $content,
        $fileName,
        ['mimeType' => 'text/plain']
    );
}
```

Завдяки такому механізму імпорт розкладу стає більш контрольованим. Адміністративний користувач може швидко визначити, які саме обмеження були порушені, експортувати перелік помилок і використати його для виправлення початкового файлу. Це зменшує ризик перенесення некоректного розкладу до бази даних і підвищує якість подальшої роботи з модулем.

Після імпорту дані розкладу зберігаються у відповідних таблицях бази даних. Основна інформація про пару потрапляє до таблиці schedule_pair, а зв’язки з академічними групами, викладачами та аудиторіями зберігаються в таблицях schedule_pair_group, schedule_pair_staff та schedule_pair_room. Такий підхід дозволяє уникнути дублювання даних і забезпечує можливість гнучкого редагування кожного заняття.

Окрім автоматичного імпорту, модуль підтримує ручне створення та редагування окремих занять. За цю частину відповідає контролер ScheduleBulkController, який обробляє запити на створення, оновлення, переміщення, копіювання та видалення пар. При створенні заняття система отримує з форми дані про тиждень, день, пару, групи, викладачів, аудиторії, дисципліну та тип заняття.

Скорочений приклад створення заняття доцільно подати не одним великим фрагментом, а кількома послідовними частинами. Першим етапом є підготовка відповіді контролера та отримання основних параметрів заняття з HTTP-запиту.

```
public function actionCreate(): array
{
    Yii::$app->response->format = Response::FORMAT_JSON;
    $request = Yii::$app->request;

    $weekId = $request->post('week_id');
    $dayId = $request->post('day_id');
    $hourId = $request->post('hour_id');

    $groupIds = $request->post('groups', []);
    $roomIds = $request->post('rooms', []);
    $staffIds = $request->post('staffs', []);
}
```

У цьому фрагменті контролер отримує тиждень, день та номер пари, а також списки академічних груп, аудиторій і викладачів, пов’язаних із заняттям. Такі дані надходять із форми створення пари, яка використовується в інтерфейсі модуля. Оскільки заняття може бути пов’язане з кількома групами, викладачами або аудиторіями, відповідні значення обробляються як масиви.

Наступним етапом є перевірка обов’язкових зв’язків і визначення версії розкладу, до якої буде додано заняття. Якщо версія не передана у запиті, система автоматично використовує поточну активну та видиму версію.

```
if ($error = $this->validateEmptyRelations($groupIds, $roomIds, $staffIds)) {
    return $error;
}

$scheduleVersionId = $request->post('schedule_version_id')
    ?? ScheduleVersion::find()
        ->where(['is_current' => true, 'is_visible' => true])
        ->select('id')
        ->scalar();
```

Ця перевірка потрібна для того, щоб не допустити створення заняття без ключових зв'язків. Наприклад, пара без викладача, групи або аудиторії не може вважатися повноцінним елементом розкладу. Автоматичне визначення активної версії дозволяє працювати з розкладом без необхідності щоразу вручну передавати її ідентифікатор.

Перед збереженням заняття система виконує перевірку конфліктів. У наведеному фрагменті показано приклад перевірки зайнятості аудиторій у вибраний часовий проміжок.

```
if ($collisionError = $this->checkRoomCollisions(
    $scheduleVersionId,
    $weekId,
    $dayId,
    $hourId,
    $roomIds
)) {
    return $collisionError;
}
```

Якщо вибрана аудиторія вже використовується в цей самий тиждень, день і номер пари, система повертає повідомлення про помилку та не дозволяє створити заняття. У повній реалізації аналогічна логіка може застосовуватися також до викладачів і академічних груп, що дає змогу запобігати накладанням у розкладі.

Після проходження попередніх перевірок створюється основний запис заняття. У таблицю schedule_pair зберігаються базові параметри пари: версія розкладу, навчальний тиждень, день, номер пари, тип заняття, потік та дисципліна.

```
$model = new SchedulePair();
$model->schedule_version_id = $scheduleVersionId;
$model->week_id = $weekId;
$model->day_id = $dayId;
$model->hour_id = $hourId;
$model->activity_tag_id = $request->post('activity_tag_id');
$model->flow_number = $request->post('flow_number', null);

$this->assignSubject(
    $model,
    $request->post('subject_id'),
    $request->post('subject_name')
);

$model->save();
```

Окремий метод assignSubject() використовується для визначення навчальної дисципліни. Це дозволяє винести логіку роботи з дисципліною за межі основного методу створення заняття та зробити код більш структурованим. Після заповнення основних полів модель зберігається в базі даних.

Після створення основного запису необхідно зберегти зв'язки заняття з академічними групами, аудиторіями та викладачами. Для цього використовуються проміжні таблиці, які дозволяють одному заняттю мати кілька пов'язаних сутностей.

```
$this->saveRelations(
    $model->id,
    $groupIds,
    SchedulePairGroup::class,
    'group_id'
);

$this->saveRelations(
    $model->id,
    $roomIds,
    SchedulePairRoom::class,
    'room_id'
);

$this->saveRelations(
    $model->id,
    $staffIds,
    SchedulePairStaff::class,
    'staff_id'
);
```

Такий підхід дозволяє не дублювати основну інформацію про заняття. Базовий запис пари зберігається в таблиці schedule_pair, а зв'язки з іншими сутностями — в окремих таблицях. Це робить структуру даних більш гнучкою та зручною для подальшого редагування.

Усі операції зі створення пари та збереження пов'язаних записів повинні виконуватися в межах транзакції. Це потрібно для того, щоб у разі помилки система не залишила в базі частково створене заняття.

```
$transaction = Yii::$app->db->beginTransaction();

try {
    // створення пари та збереження зв'язків

    $transaction->commit();
```

```
    $this->invalidateGroupsCache($this-
>extractPostedGroupIds($groupIds));

    return ['success' => true, 'message' => 'Пару створено успішно'];
} catch (Exception $e) {
    $transaction->rollBack();

    return ['success' => false, 'message' => $e->getMessage()];
}
```

Використання транзакції гарантує цілісність даних. Якщо основний запис заняття було створено, але під час збереження груп, викладачів або аудиторій виникла помилка, усі зміни буде скасовано. Після успішного створення заняття система очищає кеш відповідних академічних груп, щоб користувачі бачили актуальний розклад після внесених змін.

Окрім створення нових занять, модуль підтримує операції копіювання, редагування та перенесення вже існуючих пар. Це є важливою частиною роботи з розкладом, оскільки в реальних умовах навчального процесу часто виникає потреба змінити аудиторію, викладача, день або номер пари без повного повторного створення заняття. Редагування дозволяє змінити параметри конкретної пари, зокрема дисципліну, тип заняття, викладачів, групи, аудиторії, тиждень або час проведення. Перенесення пари використовується у випадках, коли заняття необхідно перемістити в інший часовий проміжок, зберігши основні зв’язки з групами, викладачами та аудиторіями. Копіювання дає змогу швидко створити нову пару на основі вже наявної, що зменшує кількість ручного введення та прискорює роботу відповідальних осіб.

У скороченому вигляді логіку зміни часової позиції заняття можна подати так:

```
$model = SchedulePair::findOne($pairId);

$model->week_id = $request->post('week_id');
$model->day_id = $request->post('day_id');
$model->hour_id = $request->post('hour_id');

if ($model->save()) {
    $this->invalidateGroupsCache($affectedGroupIds);

    return [
        'success' => true,
        'message' => 'Пару перенесено успішно'
    ];
```

```
}
```

Після виконання таких дій система оновлює відповідні записи в базі даних та очищає кеш для академічних груп, яких стосувалися зміни. Це дозволяє уникнути ситуації, коли після редагування або перенесення заняття користувачі продовжують бачити застарілий варіант розкладу.

Окрему роль у модулі відіграє механізм перевірки конфліктів. Під час перегляду або редагування розкладу система аналізує, чи не виникають накладання для академічної групи, викладача або аудиторії. На рівні даних така перевірка базується на порівнянні версії розкладу, тижня, дня, номера пари та відповідної сутності. Наприклад, для викладача конфлікт виникає тоді, коли в один і той самий час він призначений на кілька різних занять. Для аудиторії аналогічно перевіряється її зайнятість у конкретному часовому проміжку.

Під час реалізації модуля перевірка конфліктів виконується за допомогою спеціалізованих методів сервісного шару та контролерів, які аналізують наявні записи розкладу перед створенням або редагуванням заняття. Зокрема, система перевіряє, чи не використовується вибрана аудиторія, викладач або академічна група в той самий часовий проміжок у межах поточної версії розкладу. Якщо виявлено накладання, користувач отримує повідомлення про помилку, а операція збереження не виконується.

У спрощеному вигляді виклик перевірки конфлікту аудиторій під час створення заняття виглядає так:

```
if ($collisionError = $this->checkRoomCollisions(
    $scheduleVersionId,
    $weekId,
    $dayId,
    $hourId,
    $roomIds
)) {
    return $collisionError;
}
```

У реальній реалізації перевірка є складнішою, оскільки враховує не лише аудиторії, а й викладачів, групи та підгрупи. Для академічних груп додатково перевіряється, чи заняття стосується всієї групи або лише окремої підгрупи. Це

дозволяє уникати помилкових конфліктів у випадках, коли різні підгрупи однієї академічної групи можуть мати різні заняття в один і той самий час.

Також у модулі реалізовано перегляд розкладу за різними сутностями. Для цього використовуються окремі контролери та сервіси, які формують табличне або спискове представлення даних. Розклад може переглядатися за академічною групою, викладачем або аудиторією. При цьому використовується однакова базова структура даних, але змінюється основний критерій фільтрації: group_id, staff_id або room_id.

Окрему увагу в реалізації приділено сторінці «Мій розклад», яка використовується викладачами для перегляду власного розкладу занять. На цій сторінці розклад формується на основі зв'язку між заняттями та працівниками, що зберігається в таблиці schedule_pair_staff. Система визначає поточного авторизованого користувача, зіставляє його з відповідним записом працівника та формує вибірку занять, у яких він зазначений як викладач.

У спрощеному вигляді отримання розкладу викладача можна подати так:

```
$pairs = SchedulePair::find()
    ->joinWith(['schedulePairStaff'])
    ->where(['schedule_pair_staff.staff_id' => $staffId])
    ->andWhere(['schedule_pair.schedule_version_id'                    =>
$scheduleVersionId])
    ->all();
```

Такий підхід дозволяє викладачу швидко отримувати актуальну інформацію про власні заняття без необхідності вручну шукати їх у загальному розкладі груп або аудиторій. Залежно від наданих прав доступу викладач може лише переглядати розклад або виконувати окремі дії з редагування своїх занять.

У модулі також реалізовано сторінку «Контроль», яка призначена для перегляду всіх пар, наявних у системі, з можливістю використання фільтрів. На відміну від сторінок перегляду розкладу за групами, викладачами або аудиторіями, сторінка контролю надає більш узагальнене адміністративне представлення даних. Вона дозволяє переглядати заняття за різними критеріями, зокрема за групою, викладачем, аудиторією, дисципліною, типом заняття, днем, тижнем, номером пари або версією розкладу.

Приклад побудови фільтрованого запиту можна подати так:

```
$query = SchedulePair::find()
    ->joinWith([
        'schedulePairGroups',
        'schedulePairStaff',
        'schedulePairRooms',
        'subject',
        'activityTag'
    ])
    ->andFilterWhere(['schedule_pair.schedule_version_id'          =>
$versionId])
    ->andFilterWhere(['schedule_pair_group.group_id' => $groupId])
    ->andFilterWhere(['schedule_pair_staff.staff_id' => $staffId])
    ->andFilterWhere(['schedule_pair_room.room_id' => $roomId])
    ->andFilterWhere(['schedule_pair.day_id' => $dayId])
    ->andFilterWhere(['schedule_pair.hour_id' => $hourId]);
```

Сторінка «Контроль» є корисною для адміністрації, оскільки дозволяє швидко знаходити потрібні заняття, перевіряти коректність заповнення розкладу, аналізувати навантаження та виявляти потенційні помилки. Такий інструмент особливо важливий після імпорту FET/XML-файлу, коли необхідно перевірити, чи всі заняття були коректно перенесені у внутрішню структуру системи.

Сторінка «Контроль» не призначена для редагування посилань або нотаток до занять. Її основним завданням є адміністративний перегляд усіх пар розкладу з можливістю фільтрації та подальшого експорту даних. Завдяки фільтрам користувач може швидко знайти потрібні заняття за семестром, версією розкладу, академічною групою, викладачем, аудиторією, дисципліною, днем, тижнем або номером пари. Це дає змогу перевіряти коректність імпорту, аналізувати заповнення розкладу, знаходити потенційні помилки та формувати потрібну вибірку для експорту.

Керування посиланнями та нотатками до занять виконується на сторінках, де розклад відображається у вигляді списку. До таких сторінок належать спискове представлення розкладу, розклад викладача та сторінка «Мій розклад». У цьому форматі заняття подаються послідовним переліком, що дозволяє зручно працювати з додатковою інформацією до конкретної пари.

Посилання можуть використовуватися для переходу до онлайн-заняття, електронного курсу або іншого навчального ресурсу. Нотатки можуть містити уточнення щодо проведення заняття, службову інформацію або додаткові

пояснення. Після додавання або редагування ці дані зберігаються у відповідних таблицях бази даних і відображаються у розкладі разом з основною інформацією про заняття.

У спрощеному вигляді створення додаткової інформації до заняття можна подати так:

```
$pair = SchedulePair::findOne($pairId);

$link = new SchedulePairLink();
$link->pair_id = $pair->id;
$link->staff_id = Yii::$app->user->identity->staff_id;
$link->url = $request->post('url');
$link->type = $request->post('type');
$link->save();

$note = new SchedulePairNote();
$note->pair_id = $pair->id;
$note->staff_id = Yii::$app->user->identity->staff_id;
$note->text = $request->post('note');
$note->week_number = $request->post('week_number');
$note->save();
```

У наведеному фрагменті додаткова інформація не зберігається безпосередньо в таблиці основного заняття, а виноситься в окремі пов’язані таблиці. Такий підхід дозволяє зберігати декілька допоміжних записів для однієї пари та не перевантажувати основну структуру розкладу. Після додавання або редагування посилань і нотаток ця інформація може відображатися безпосередньо у розкладі користувача разом з основними даними про заняття.

Окремим функціональним елементом є сторінка керування версіями розкладу. Вона дозволяє переглядати всі створені версії, визначати поточну активну версію, керувати видимістю розкладу для користувачів та обмежувати можливість редагування окремих версій. Це дає змогу адміністрації працювати з декількома варіантами розкладу, не впливаючи на той розклад, який уже відображається студентам і викладачам.

У скороченому вигляді вибір активної версії розкладу може бути представлений так:

```
$scheduleVersionId = $request->get('schedule_version_id')
    ?? ScheduleVersion::find()
        ->where([
            'is_current' => true,
            'is_visible' => true
```

```
])
->select('id')
->scalar();
```

За замовчуванням на основних сторінках модуля відображається лише активна та видима версія розкладу. Це зроблено для того, щоб студенти, викладачі та інші користувачі системи бачили актуальну інформацію без необхідності самостійно обирати потрібний варіант. Водночас зі сторінки «Версії» користувачі з відповідними правами можуть перейти до перегляду конкретної версії розкладу. Такий підхід дозволяє одночасно підтримувати стабільне відображення актуального розкладу та зберігати можливість аналізу або редагування інших версій.

Робота з версіями також важлива під час тестування змін. Адміністрація може підготувати новий варіант розкладу, перевірити його на конфлікти, внести необхідні коригування, а вже після цього зробити його поточним і видимим для користувачів. Це зменшує ризик випадкової публікації неперевіреної інформації.

Доступ до функціоналу модуля обмежується відповідно до ролей користувачів. Студенти мають доступ лише до перегляду власного персоналізованого розкладу, який формується на основі розкладу академічної групи та вибіркових дисциплін. Викладачі можуть переглядати власний розклад на сторінці «Мій розклад», а також, за наявності відповідних прав, виконувати дії з редагування занять, що належать до їхньої зони відповідальності.

Адміністрація має вищий рівень доступу, тому отримує всі можливості, доступні викладачам, а також додатковий адміністративний функціонал. До таких можливостей належать імпорт FET/XML-файлу, перегляд сторінки «Контроль», керування версіями розкладу, перевірка конфліктів, експорт розкладу та редагування ширшого набору занять. Таким чином, права адміністрації розширюють права викладача, а не замінюють їх окремим незалежним набором дій.

Власник системи або технічний адміністратор має доступ до функцій, пов’язаних із керуванням доступами користувачів, налаштуванням системи та контролем стабільності її роботи. Такий розподіл ролей дозволяє забезпечити

безпечну експлуатацію модуля, обмежити доступ до критичних дій та зменшити ризик несанкціонованого редагування розкладу.

Важливою частиною розробленого функціоналу є формування персоналізованого розкладу студента. На відміну від звичайного групового розкладу, студентський розклад повинен враховувати не лише заняття академічної групи, а й вибіркові дисципліни, які обрав конкретний студент. Для цього використовується сервіс StudentScheduleCacheService.

Основна точка входу для отримання студентського розкладу має такий вигляд:

```
public static function get(int $educationId, string $viewType = 'table'): array
{
    $viewType = self::normalizeViewType($viewType);
    $redis = RedisCacheService::getInstance();

    if (!$redis->connect()) {
        return self::buildFromDb($educationId, $viewType);
    }

    try {
        return self::getFromRedis($redis, $educationId, $viewType);
    } catch (NotFoundHttpException $e) {
        throw $e;
    } catch (Throwable $e) {
        Yii::warning(
            "Redis schedule cache error, falling back to DB: " . $e->getMessage(),
            'schedule-cache'
        );

        return self::buildFromDb($educationId, $viewType);
    }
}
```

У цьому фрагменті реалізовано спробу отримати розклад із Redis. Якщо підключення до Redis недоступне або під час читання кешу виникає помилка, система не припиняє роботу, а формує розклад безпосередньо з бази даних. Такий підхід підвищує надійність модуля, оскільки відмова кешу не блокує доступ студента до розкладу.

Для зменшення кількості повторних обчислень використовується не індивідуальний кеш для кожного студента, а кешування за принципом когорти. Якщо кілька студентів належать до однієї академічної групи та мають однаковий

набір вибіркових дисциплін і потоків, для них може використовуватися один спільний кешований розклад. Формування ключів кешу винесено в окремий допоміжний клас ScheduleRedisHelper.

Фрагмент формування ключів Redis наведено нижче:

```
public static function studentEnrollmentsKey(int $educationId): string
{
    return "student_schedule:{$educationId}:electives";
}

public static function cohortKey(
    int $groupId,
    string $viewType,
    string $comboHash
): string {
    return "schedule:cohort:{$groupId}:{$viewType}:{$comboHash}";
}

public static function tagGroupKey(int $groupId): string
{
    return "tag:group:{$groupId}";
}
```

Окремо формується хеш комбінації вибіркових дисциплін. Завдяки цьому студенти з однаковим набором вибіркових дисциплін можуть отримувати розклад із одного кешованого запису:

```
public static function comboHash(int $groupId, array $enrollments):
string
{
    $pairs = array_map(
        fn($e) => implode(':', [
            $e['elective_subject_id'] ?? $e['elective_id'] ?? '0',
            $e['activity_type_id'] ?? '0',
            $e['flow_number'] ?? $e['flow_id'] ?? '0',
        ]),
        $enrollments
    );

    sort($pairs);

    return md5($groupId . '|' . implode(',', $pairs));
}
```

Такий механізм дозволяє суттєво зменшити навантаження на базу даних під час масового перегляду розкладу студентами. Замість повторного формування розкладу для кожного окремого користувача система може використовувати вже підготовлений результат для всієї групи студентів з однаковими параметрами.

Формування персонального розкладу студента відбувається шляхом поєднання двох джерел даних. Першим джерелом є загальний розклад академічної групи, який формується на основі таблиць schedule_pair, schedule_pair_group, schedule_pair_staff та schedule_pair_room. Другим джерелом є дані про вибіркові дисципліни, які зберігаються в таблицях elective_result_student, elective_result_student_flow та elective_result_schedule. У результаті студент бачить лише ті вибіркові заняття, які відповідають його індивідуальному вибору.

Окремим функціональним елементом є експорт відфільтрованих даних зі сторінки «Контроль» у форматі Excel. Ця можливість використовується для перевірки, аналізу та передавання інформації структурним підрозділам університету. Перед експортом адміністративний користувач може сформувати потрібну вибірку за допомогою фільтрів за групою, викладачем, аудиторією, дисципліною, днем, тижнем, номером пари або версією розкладу. Після цього система отримує відповідні записи з бази даних, доповнює їх пов’язаною інформацією та формує файл у табличному форматі.

Таким чином, у межах розробки модуля «Розклад занять» було реалізовано основні функціональні можливості, необхідні для управління навчальним розкладом у закладі вищої освіти. Система підтримує імпорт FET/XML-файлів, створення, редагування, копіювання та перенесення занять, керування версіями розкладу, перевірку конфліктів, перегляд розкладу викладача на сторінці «Мій розклад», адміністративний контроль усіх пар за фільтрами, формування персоналізованого студентського розкладу, кешування даних у Redis та експорт інформації до Excel. Використання сервісного шару, транзакцій, рольового розмежування доступу та кешування дозволило підвищити надійність, продуктивність і зручність супроводу розробленого програмного забезпечення.

Висновки до другого розділу

У процесі проєктування модуля «Розклад занять» інформаційної системи онлайн-менеджменту освітніх процесів Digital University UA було визначено

основні функціональні та нефункціональні вимоги до програмного забезпечення. На цьому етапі сформульовано ключові цілі розробки, визначено основні категорії користувачів системи та описано варіанти їхньої взаємодії з модулем. Особливу увагу приділено ролям студентів, викладачів, адміністрації та власника системи, оскільки кожна з цих категорій має різний рівень доступу до функціоналу розкладу.

У межах аналізу варіантів використання було визначено, що студенти повинні мати доступ до персоналізованого розкладу, сформованого на основі розкладу академічної групи та вибіркових дисциплін. Викладачі отримують можливість переглядати власний розклад на сторінці «Мій розклад», а адміністрація має розширений набір можливостей, що охоплює імпорт розкладу, редагування, копіювання та перенесення занять, перегляд сторінки «Контроль», керування версіями, перевірку конфліктів і експорт даних. Такий розподіл функцій дозволяє забезпечити контрольований доступ до критичних операцій та підвищити безпеку роботи системи.

У ході проєктування було обґрунтовано вибір системи керування базами даних MySQL, яка відповідає реляційній природі даних модуля «Розклад занять». Розклад занять складається з великої кількості взаємопов’язаних сутностей, серед яких заняття, версії розкладу, академічні групи, викладачі, аудиторії, навчальні дисципліни, дні, тижні та часові проміжки. Використання реляційної бази даних дозволило забезпечити цілісність цих зв’язків, підтримку складних запитів, транзакційність операцій та зручну інтеграцію з фреймворком Yii2 через механізм ActiveRecord.

На етапі розробки бази даних було визначено основні таблиці модуля, їхні атрибути та взаємозв’язки. Центральною сутністю виступає таблиця занять, яка пов’язується з таблицями версій розкладу, академічних груп, викладачів та аудиторій. Окреме використання проміжних таблиць для зв’язку занять із групами, працівниками та аудиторіями дозволило реалізувати гнучку модель даних, у якій одне заняття може мати кілька пов’язаних груп, викладачів або аудиторій. Також було враховано підтримку підгруп, вибіркових дисциплін,

приміток, посилань та обмежень, необхідних для коректного функціонування модуля.

Окрему увагу було приділено механізму імпортування розкладу з FET/XML-файлу. Оскільки первинна генерація одного з варіантів розкладу може виконуватися стороннім програмним забезпеченням, у системі передбачено процес зчитування зовнішнього файлу, перетворення його структури, зіставлення отриманих даних із внутрішніми сутностями бази даних і подальше збереження інформації у таблицях модуля. Такий підхід дозволяє поєднати переваги автоматизованої генерації розкладу із можливостями подальшого редагування, контролю, затвердження та відображення даних у межах Digital University UA.

У межах реалізації функціоналу було розглянуто основні програмні механізми модуля. Зокрема, описано створення та редагування занять, копіювання й перенесення пар, перевірку конфліктів між групами, викладачами та аудиторіями, відображення розкладу за різними сутностями, а також роботу сторінок «Мій розклад», «Контроль» і «Версії». Додатково враховано можливість прикріплення до занять посилань і нотаток, які відображаються у розкладі та можуть адмініструватися через сторінку «Контроль». Важливою особливістю є те, що за замовчуванням на основних сторінках системи відображається лише активна та видима версія розкладу, тоді як користувачі з відповідними правами можуть перейти до перегляду конкретних версій зі сторінки керування версіями.

Також у другому розділі було описано використання Redis для кешування персоналізованих розкладів студентів. Оскільки студентський розклад формується на основі загального розкладу академічної групи та вибіркових дисциплін, його повторне обчислення може створювати додаткове навантаження на базу даних. Використання кешування дозволяє пришвидшити відображення розкладу, зменшити кількість повторних запитів до MySQL та забезпечити комфортну роботу користувачів при великій кількості одночасних звернень.

Таким чином, у другому розділі було виконано проєктування основних складових модуля «Розклад занять», визначено варіанти використання системи, обґрунтовано вибір MySQL як основної системи керування базами даних, описано структуру бази даних, механізм імпорту FET/XML-файлу та ключові елементи реалізації функціоналу. Отримані результати підтверджують, що запропонована структура модуля відповідає вимогам до сучасної інформаційної системи управління освітнім процесом, забезпечує підтримку версійності, рольового доступу, контролю конфліктів, персоналізованого відображення розкладу та подальшого розвитку програмного продукту.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ОНЛАЙН-МЕНЕДЖМЕНТУ ОСВІТНІХ ПРОЦЕСІВ

3.1. Порядок встановлення та налаштування параметрів додатку

Перед запуском програмного продукту було визначено загальну схему розгортання системи, що демонструє взаємодію основних компонентів вебзастосунку та допоміжних сервісів. Оскільки модуль «Розклад занять» є складовою частиною інформаційної системи Digital University UA, його розгортання окремо від основного проєкту не передбачається. Модуль використовує спільну систему авторизації, базу даних, довідники, ролі користувачів та інші компоненти платформи, тому його встановлення виконується в межах загального розгортання системи.

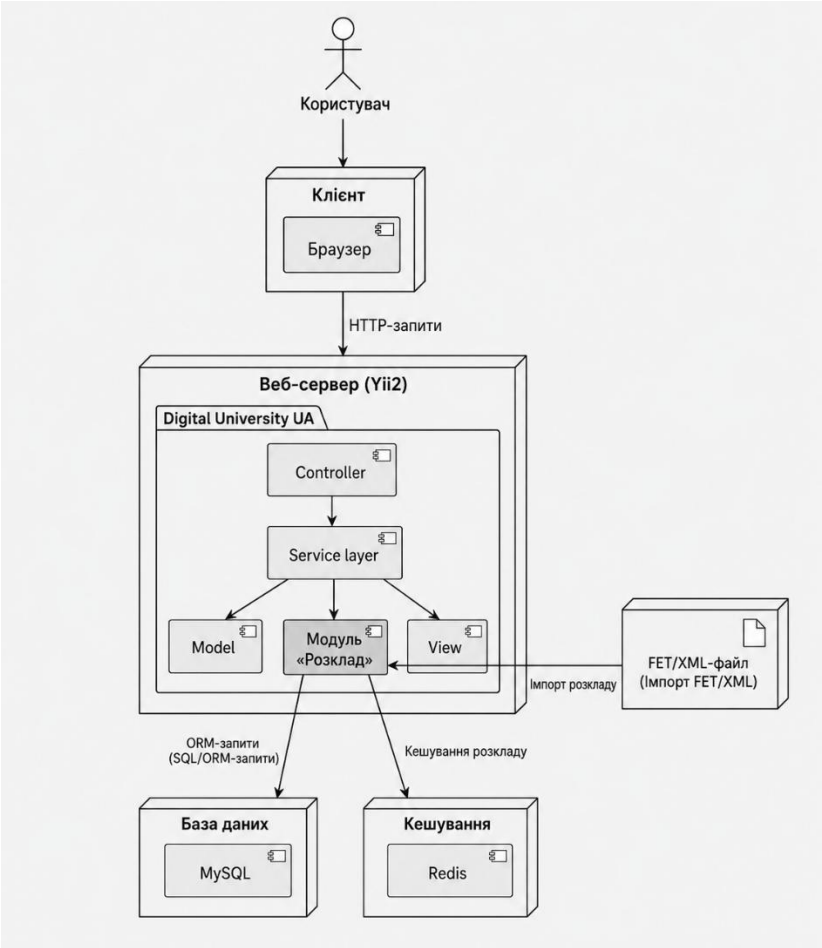

Рис. 3.1 – Діаграма розгортання інформаційної системи Digital University UA з модулем «Розклад занять»

Діаграма розгортання демонструє архітектуру веборієнтованої інформаційної системи, у межах якої функціонує модуль «Розклад занять». Користувач взаємодіє з програмним продуктом через веббраузер, надсилаючи HTTP-запити до серверної частини. На сервері розгорнуто Yii2-застосунок, який реалізує логіку роботи системи за архітектурним шаблоном MVC. Обробка запитів виконується контролерами, взаємодія з базою даних забезпечується моделями, а формування HTML-сторінок здійснюється за допомогою представлень.

Основними компонентами серверної частини є:

- контролери — приймають HTTP-запити, виконують маршрутизацію та викликають відповідні сервіси модуля;
- моделі — забезпечують взаємодію з таблицями бази даних через механізм Active Record;
- представлення — формують сторінки інтерфейсу користувача;
- сервісний шар — містить основну бізнес-логіку модуля, зокрема формування розкладу, імпорт FET/XML-файлів, перевірку конфліктів, кешування, експорт і роботу з версіями розкладу.

Основні дані системи зберігаються в реляційній базі даних MySQL. У ній міститься інформація про користувачів, студентів, викладачів, академічні групи, дисципліни, аудиторії, версії розкладу, навчальні заняття, зв’язки пар із групами, викладачами та аудиторіями, а також додаткові дані у вигляді посилань і нотаток до занять.

Для підвищення продуктивності використовується Redis. У модулі «Розклад занять» Redis застосовується для кешування персоналізованих студентських розкладів, що дозволяє зменшити кількість повторних запитів до MySQL та пришвидшити відображення інформації в особистих кабінетах користувачів[26;27]. Налаштування Redis передбачає визначення хоста, порту, параметрів підключення та префіксів ключів кешу, які використовуються для збереження даних розкладу.

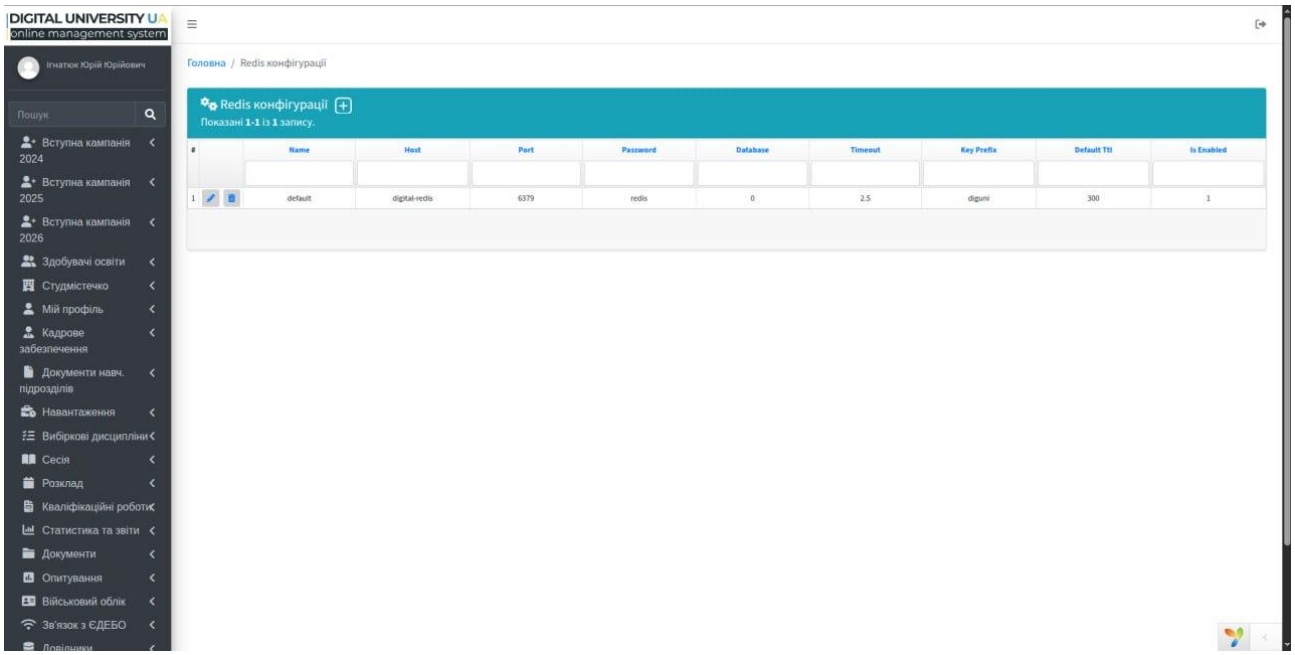

Рис. 3.2 – Приклад налаштування підключення до Redis у конфігурації системи

Розгортання програмного продукту виконується за допомогою Docker[25]. Такий підхід дозволяє забезпечити однакове середовище виконання на різних етапах розробки, тестування та експлуатації. У Docker-середовищі можуть бути окремо налаштовані контейнери для вебзастосунку, бази даних MySQL та Redis. Це спрощує запуск системи, оновлення залежностей і перенесення проєкту між різними серверами.

Загальний порядок запуску системи передбачає такі етапи:

- підготовка серверного середовища з підтримкою Docker та Docker Compose;
- налаштування конфігураційних параметрів проєкту;
- запуск контейнерів вебзастосунку, MySQL та Redis;
- застосування міграцій бази даних;
- перевірка підключення до бази даних і кеш-сховища;
- налаштування ролей користувачів та доступів до функціоналу;
- підготовка довідників, необхідних для роботи розкладу;
- імпорт початкового варіанта розкладу з FET/XML-файлу.

```
NAME                          IMAGE                 COMMAND                  SERVICE    CREATED        STATUS                  PORTS
digital-local-diia-microservice   local-template-diia   "docker-entrypoint.s…"   diia       2 months ago   Up 43 hours
digital-local-edebo-microservice  local-template-edebo  "docker-entrypoint.s…"   edebo      2 months ago   Up 43 hours             3000/tcp
digital-local-mongodb-server      mongo:5.0             "docker-entrypoint.s…"   mongodb    2 months ago   Up 43 hours             0.0.0.0:27017->27017/tcp, [::]:27017->27017/tcp
digital-local-mysql-server        local-template-mysql  "docker-entrypoint.s…"   mysql      2 months ago   Up 43 hours             0.0.0.0:3306->3306/tcp, [::]:3306->3306/tcp, 33060/tcp
digital-local-nginx               local-template-nginx  "/docker-entrypoint.…"   nginx      2 months ago   Up 43 hours             0.0.0.0:80->80/tcp, [::]:80->80/tcp, 443/tcp
digital-local-rabbitmq            rabbitmq:3-management "docker-entrypoint.s…"   rabbitmq   2 months ago   Up 43 hours (healthy)   4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, [::]:5672->5672/tcp, 15671/tcp, 15691-15692/tcp,
25672/tcp, 0.0.0.0:15672->15672/tcp, [::]:15672->15672/tcp
digital-local-website             local-template-website "/bin/sh -c '    mkd…"   website    43 hours ago   Up 43 hours             9000/tcp
digital-redis                     redis:7-alpine        "docker-entrypoint.s…"   redis      5 weeks ago    Up 43 hours             6379/tcp
```

Рис. 3.3 – Приклад запущених контейнерів програмного середовища

Перед початком роботи з модулем необхідно переконатися, що в системі наявні базові довідкові дані. До них належать навчальні роки, семестри, академічні групи, викладачі, аудиторії, навчальні дисципліни, типи занять, дні тижня, часові проміжки та навчальні тижні. Без цих даних імпортований розклад не зможе бути коректно зіставлений із внутрішніми сутностями системи.

Початкове наповнення модуля розкладом виконується шляхом імпорту FET/XML-файлу. Такий файл є результатом первинної генерації розкладу стороннім програмним забезпеченням. Після завантаження файл обробляється сервісами модуля, а отримані дані зіставляються з наявними записами бази даних. У результаті створюється відповідна версія розкладу, яка надалі може бути переглянута, відредагована, перевірена на конфлікти та опублікована для користувачів.

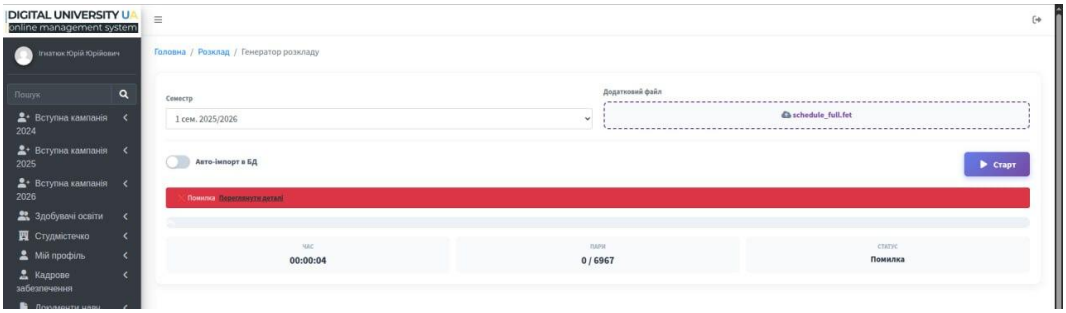

Рис. 3.4 – Сторінка імпорту розкладу з FET/XML-файлу

Під час імпорту розкладу система може виявити помилки, пов’язані з некоректністю або неповною відповідністю даних у FET/XML-файлі внутрішнім правилам системи. Приклад такої ситуації зображено на рисунку 3.4, де після запуску імпорту відображається повідомлення про помилку. Такі помилки можуть виникати у випадках, коли сформований у зовнішньому програмному забезпеченні розклад порушує обмеження, попередньо задані в системі.

До таких обмежень належать правила, що зберігаються в межах schedule restrictions і можуть стосуватися академічних груп, викладачів або аудиторій. Наприклад, система може зафіксувати невідповідність, якщо у файлі розкладу

певна група, викладач або аудиторія призначені на часовий проміжок, який був обмежений або заборонений у налаштуваннях. Це дозволяє виявляти проблеми ще на етапі імпорту та не переносити до бази даних некоректний варіант розкладу без попередньої перевірки.

Для зручності адміністративного користувача передбачено можливість перегляду деталей помилки. Крім того, отримані повідомлення можуть бути експортовані у текстовий файл формату .txt. Це дає змогу зберегти перелік знайдених проблем, передати його відповідальним особам або використати для подальшого коригування початкового FET/XML-файлу. Після виправлення помилок файл може бути повторно імпортований у систему.

Після імпорту розкладу адміністративний користувач може перейти до сторінки керування версіями. На цій сторінці визначається, яка версія є активною, видимою для користувачів і доступною для редагування. За замовчуванням на основних сторінках модуля відображається саме активна та видима версія розкладу. Це дозволяє студентам і викладачам бачити лише актуальну інформацію, тоді як адміністрація може окремо працювати з іншими версіями.

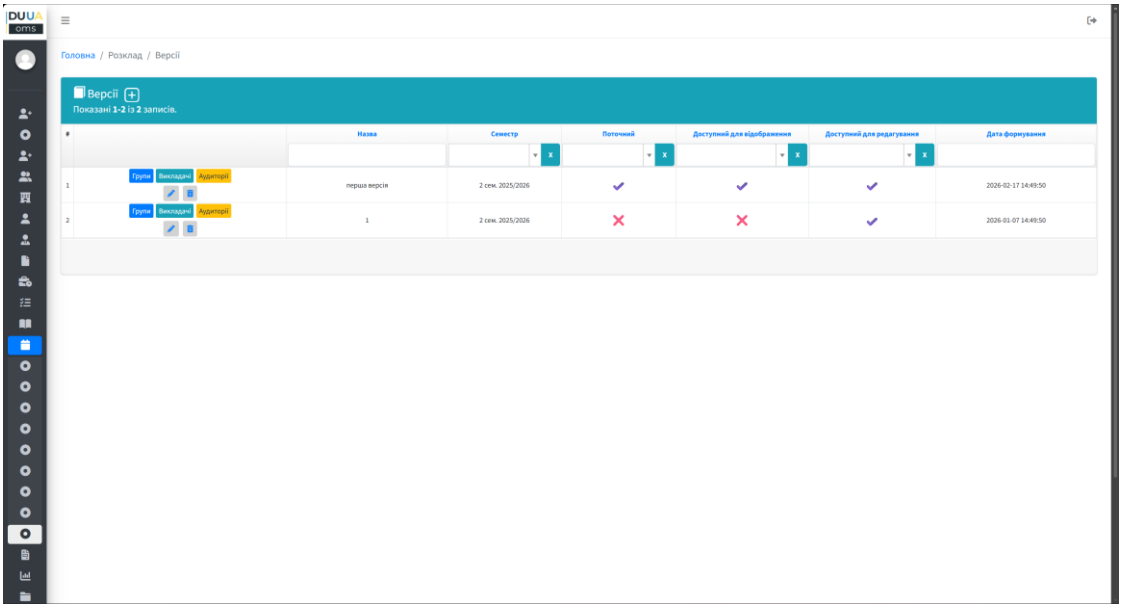

Рис. 3.5 – Сторінка керування версіями розкладу

Окремим етапом налаштування є перевірка прав доступу користувачів. Студенти повинні мати доступ лише до перегляду персоналізованого розкладу.

Викладачі отримують можливість переглядати власний розклад на сторінці «Мій розклад», а також виконувати окремі дії відповідно до наданих прав. Адміністрація має розширений доступ до імпорту, редагування, копіювання, перенесення пар, перегляду сторінки «Контроль», керування версіями, перевірки конфліктів, управління посиланнями й нотатками до занять та експорту розкладу.

Після завершення налаштування та імпорту початкових даних система готова до використання.

Таким чином, порядок встановлення та налаштування модуля «Розклад занять» передбачає розгортання всієї інформаційної системи Digital University UA, налаштування підключень до MySQL і Redis, запуск Docker-середовища, підготовку довідкових даних, імпорт початкового розкладу та визначення активної версії. Після виконання цих дій модуль готовий до використання студентами, викладачами та адміністративними користувачами відповідно до їхніх ролей і прав доступу.

3.2. Структура інтерфейсу додатку

Розроблена система онлайн-менеджменту освітніх процесів Digital University UA має веборієнтований інтерфейс, доступ до якого здійснюється через браузер[23,24]. Користувачеві не потрібно встановлювати додаткове програмне забезпечення на власний пристрій, оскільки вся взаємодія із системою відбувається через вебсторінки порталу. Інтерфейс побудований таким чином, щоб забезпечити швидкий доступ до основних модулів системи та водночас розмежувати функціональність відповідно до ролі користувача.

Для персоналізованого доступу до системи передбачено процедуру автентифікації. Перед початком роботи користувач повинен ввести облікові дані у відповідну форму входу. Після успішної авторизації система визначає роль користувача та надає доступ лише до тих розділів і дій, які відповідають його правам.

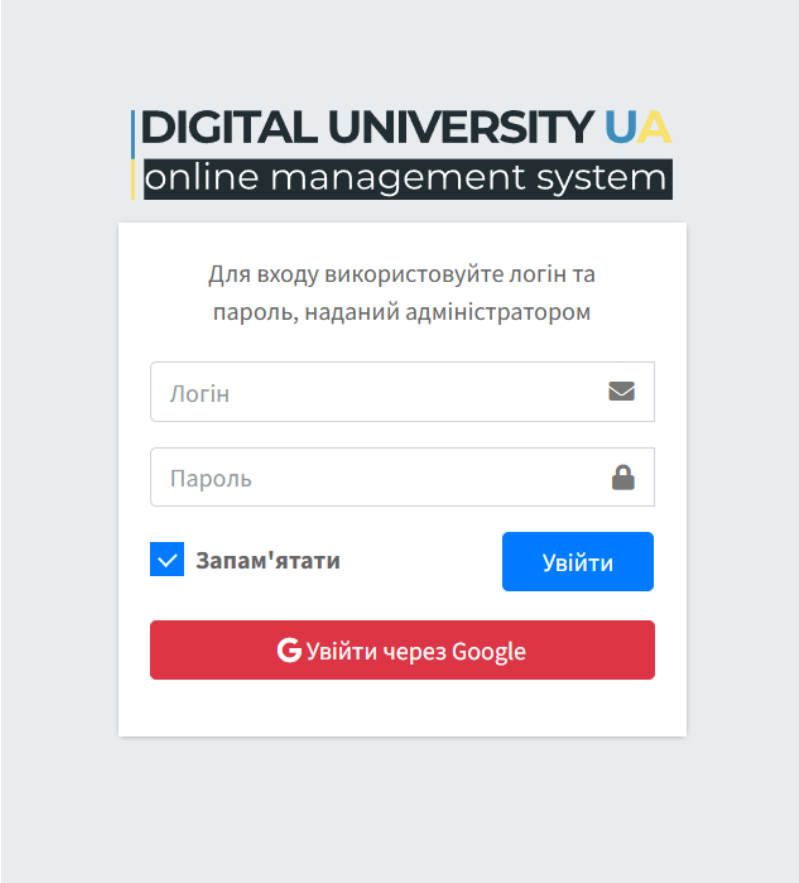

Рис. 3.6 – Форма входу до інформаційної системи Digital University UA

З міркувань безпеки система не передбачає відкритої самостійної реєстрації користувачів через вебінтерфейс[32]. Облікові записи створюються централізовано відповідальними працівниками університету. Такий підхід дозволяє контролювати доступ до освітнього середовища та зменшує ризик несанкціонованого використання системи.

Після проходження автентифікації користувач потрапляє до головного інтерфейсу порталу. Основна навігація розміщена в лівій частині екрана у вигляді вертикального меню. У цьому меню згруповано основні модулі системи, серед яких окреме місце займає модуль «Розклад занять». Наявність єдиної навігаційної панелі спрощує перехід між розділами та дозволяє користувачу швидко знайти потрібну функціональність.

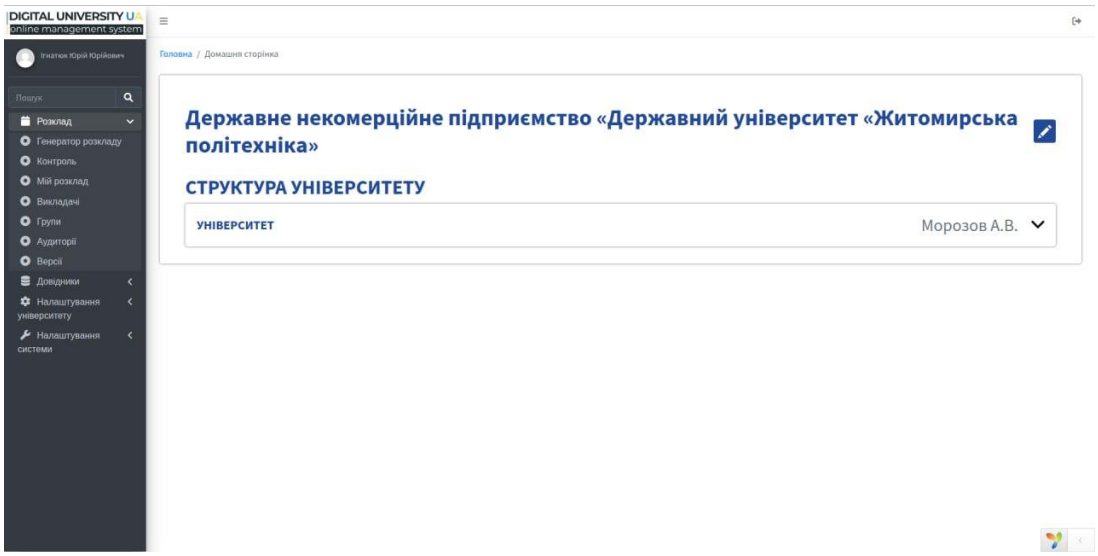

Рис. 3.7 – Головна сторінка системи з навігаційною панеллю

У лівому меню модуль «Розклад занять» представлений як окремий розділ, що містить набір підсторінок для різних сценаріїв роботи. До них належать генератор розкладу, сторінка контролю, сторінка «Мій розклад», перегляд розкладу за викладачами, академічними групами й аудиторіями, сторінка версій, а також сторінки налаштування обмежень. Набір доступних пунктів меню може відрізнятися залежно від ролі користувача.

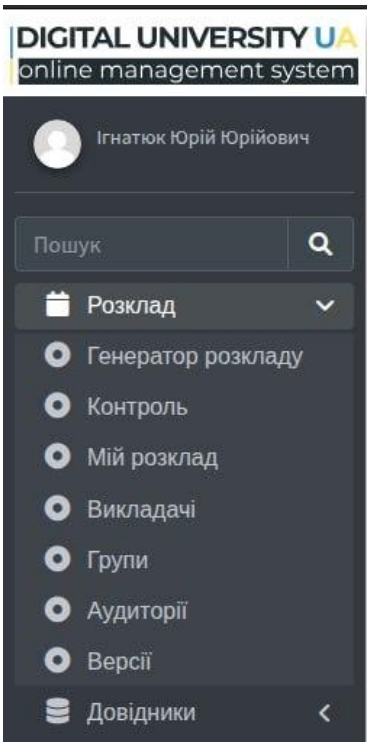

Рис. 3.8 – Розташування модуля «Розклад занять» у навігаційному меню системи

Огляд інтерфейсу модуля доцільно виконувати від імені користувача з адміністративними правами доступу, оскільки саме йому доступна більшість функцій, пов’язаних із формуванням, імпортом, перевіркою та редагуванням розкладу. Водночас окремі сторінки модуля мають різний рівень доступності залежно від ролі користувача. Наприклад, студент працює переважно з персоналізованим розкладом в особистому кабінеті, викладач має доступ до сторінки «Мій розклад», а адміністративний користувач може працювати з генератором, версіями, сторінками перегляду розкладу, фільтрами, експортом та іншими інструментами адміністрування.

Першим етапом роботи з модулем після підготовки довідкових даних є перехід до сторінки генератора розкладу. Ця сторінка призначена для завантаження FET/XML-файлу, який містить попередньо сформований варіант розкладу. У верхній частині інтерфейсу користувач обирає семестр, до якого буде прив’язано імпортований розклад, після чого завантажує файл у відповідне поле. Додатково на сторінці може бути доступний перемикач автоматичного імпорту в базу даних, який визначає, чи потрібно одразу зберігати отримані дані після обробки файлу.

Після натискання кнопки запуску система починає аналіз структури файлу, зчитує інформацію про заняття, групи, викладачів, аудиторії, навчальні дні та часові проміжки. У процесі виконання операції користувач бачить службову інформацію про стан імпорту: час виконання, кількість оброблених пар та поточний статус. Це дозволяє контролювати перебіг процесу та швидко визначити, чи завершився імпорт успішно.

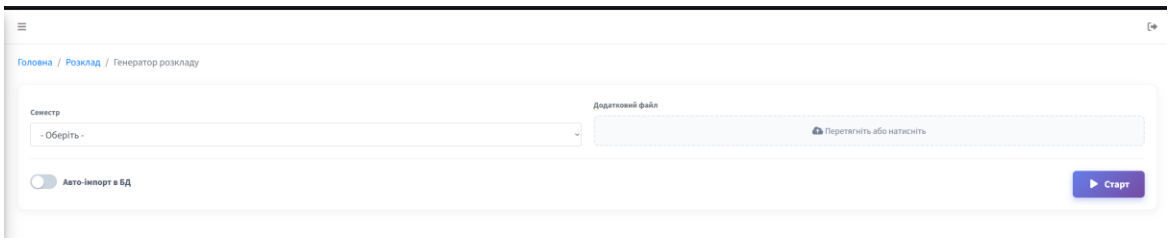

Рис. 3.9 – Сторінка генератора розкладу та імпорту FET/XML-файлу

Під час імпорту система може завершити обробку файлу як успішно, так і з помилкою. Успішне завершення означає, що файл було коректно зчитано, а

дані розкладу були опрацьовані та збережені відповідно до внутрішньої структури системи. У такому випадку користувач бачить повідомлення про успішний результат, а також загальні показники виконаної операції.

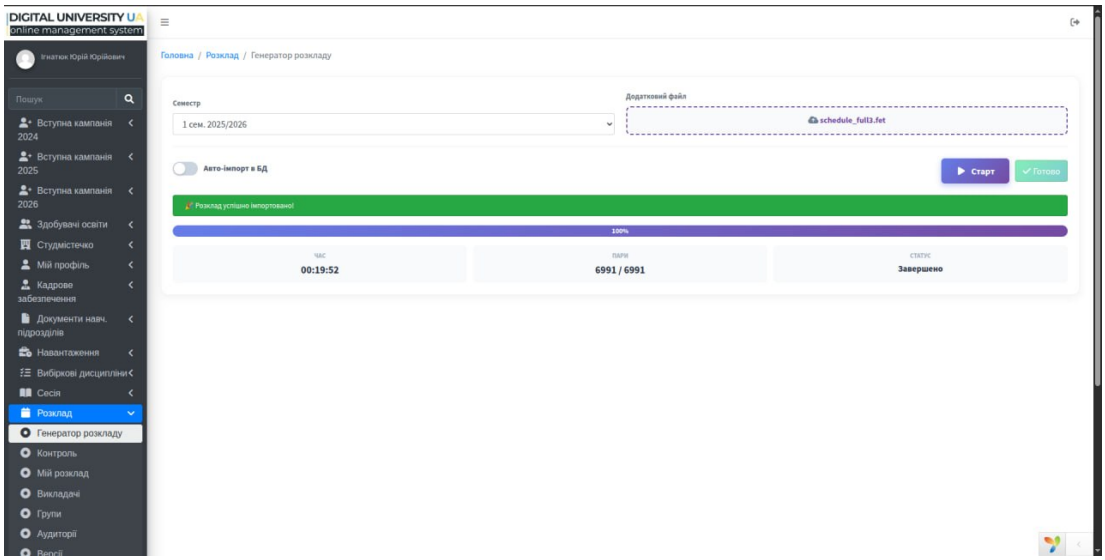

Рис. 3.10 – Приклад успішного завершення імпорту розкладу

У випадку помилки система відображає відповідне повідомлення на сторінці генератора. Такі помилки можуть виникати через некоректну структуру файлу, відсутність потрібних відповідників у довідниках або порушення обмежень, попередньо налаштованих у модулі. Наприклад, імпортований розклад може містити заняття, призначене на часовий проміжок, який заборонений для певного викладача, академічної групи або аудиторії. Це дозволяє виявити проблему ще до публікації розкладу для користувачів.

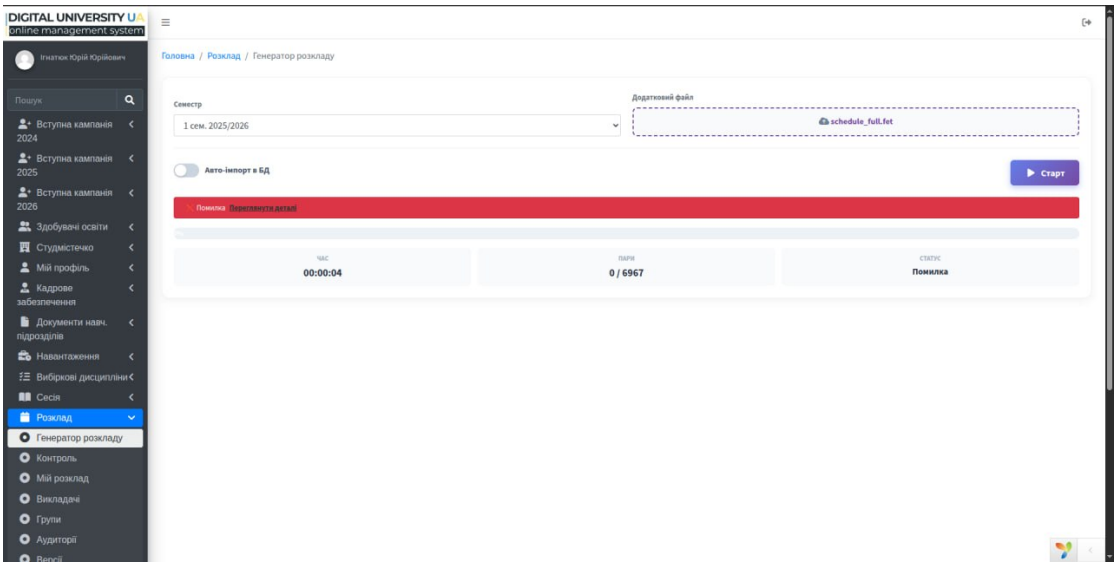

Рис. 3.11 – Приклад повідомлення про помилку під час імпорту розкладу

Для зручності адміністративного користувача система надає можливість переглянути деталі помилки. Це особливо важливо у випадках, коли імпортований файл містить значну кількість занять, а помилка стосується лише окремих записів. Деталізація дозволяє зрозуміти, які саме пари були оброблені некоректно, які обмеження були порушені або які дані не вдалося зіставити з внутрішніми довідниками системи.

Крім перегляду помилок в інтерфейсі, передбачено можливість експорту списку помилок у текстовий файл формату .txt. Такий файл може бути переданий відповідальній особі для виправлення початкового FET/XML-файлу або використаний для подальшого аналізу причин невдалого імпорту. Після внесення змін файл можна повторно завантажити до системи.

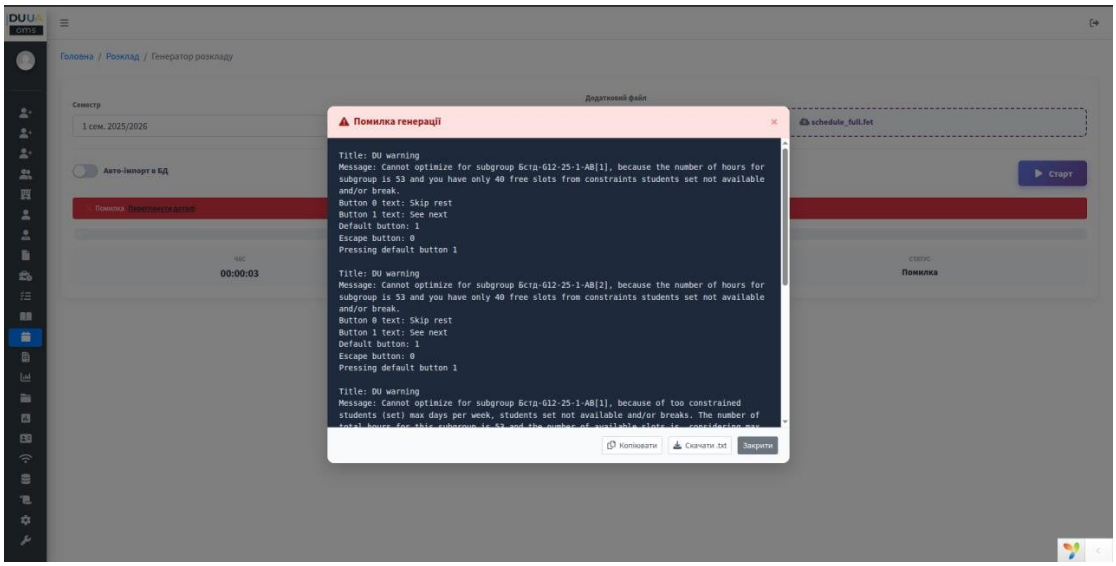

Рис. 3.12 – Перегляд деталей помилки та експорт повідомлень імпорту у текстовий файл

Після імпорту розкладу користувач може перейти до сторінки керування версіями. Ця сторінка відображає перелік створених версій розкладу із зазначенням їхньої назви, семестру, дати створення, статусу активності, видимості та можливості редагування. Версійність є важливою частиною модуля, оскільки дозволяє працювати з кількома варіантами розкладу, не змінюючи одразу той, який уже доступний студентам і викладачам.

За замовчуванням на основних сторінках модуля відображається активна та видима версія розкладу. Це зменшує ризик плутанини серед користувачів,

оскільки студенти й викладачі бачать лише актуальний варіант. Водночас користувач із відповідними правами може перейти до перегляду конкретної версії, щоб перевірити її зміст, порівняти з іншими варіантами або внести зміни перед публікацією.

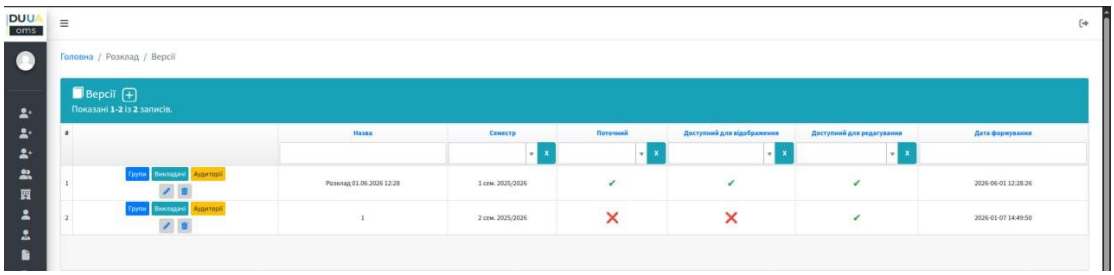

Рис. 3.13 – Сторінка керування версіями розкладу

Редагування версії дозволяє змінювати її основні службові параметри, зокрема назву, семестр, активність, видимість для користувачів та доступність для редагування. Такий механізм дає змогу адміністрації підготувати новий варіант розкладу, перевірити його та лише після цього зробити доступним для студентів і викладачів. Окреме керування ознаками активності та видимості дозволяє розмежувати технічну роботу з версією та її фактичну публікацію.

Завдяки цьому адміністратор може зберігати в системі декілька варіантів розкладу для одного семестру. Наприклад, одна версія може бути опублікованою та видимою для користувачів, тоді як інша може перебувати на етапі перевірки або доопрацювання. Це підвищує контрольованість процесу оновлення розкладу та зменшує ризик випадкового оприлюднення неперевірених змін.

Рис. 3.14 – Редагування параметрів версії розкладу

Одним із базових представлень модуля є сторінка перегляду розкладу за академічними групами. На індексній сторінці користувач може обрати потрібну

академічну групу або скористатися доступними елементами навігації для переходу до її розкладу. Такий підхід зручний для адміністрації, деканату та інших працівників, які працюють із розкладом конкретних студентських груп.

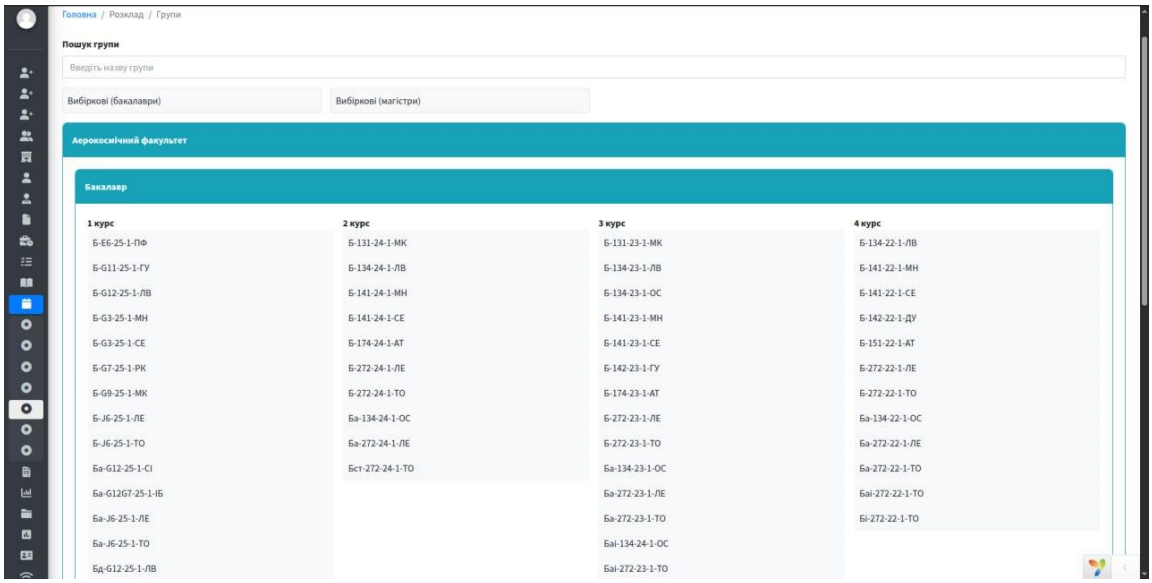

Рис. 3.15 – Сторінка вибору академічної групи для перегляду розкладу

Після вибору групи відкривається сторінка перегляду її розкладу. Основним форматом відображення є таблиця, у якій заняття розміщуються відповідно до навчальних днів, тижнів та часових пар. У комірках розкладу відображаються дисципліни, типи занять, викладачі, аудиторії та інша необхідна інформація. Табличний формат дає змогу швидко оцінити завантаженість групи протягом тижня та виявити можливі накладки.

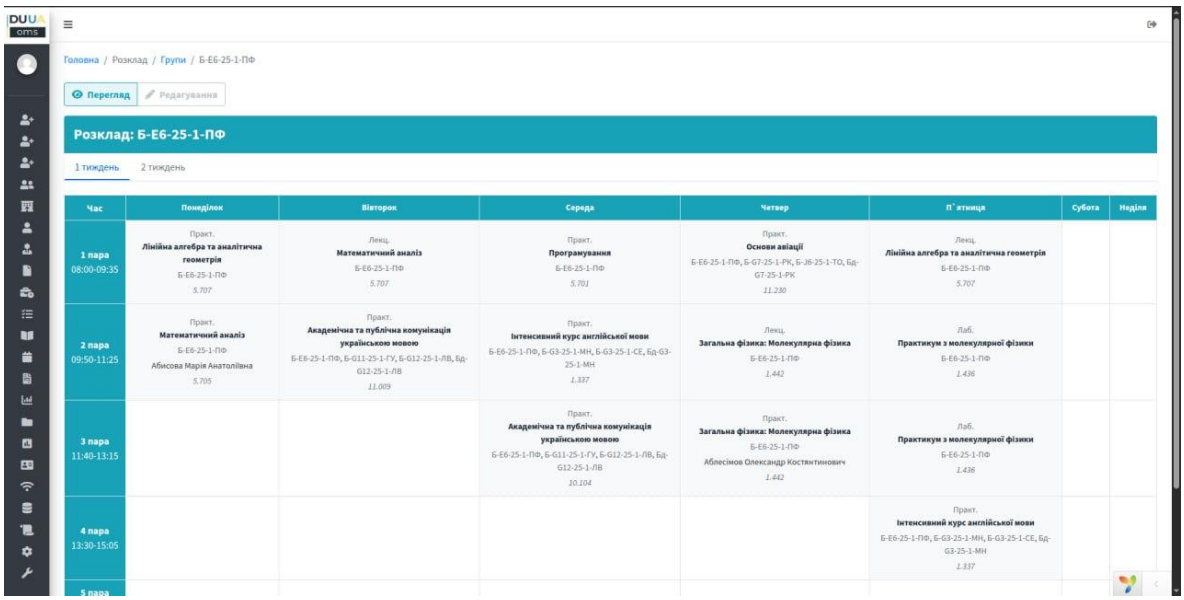

Рис. 3.16 – Табличне представлення розкладу академічної групи

Окреме представлення реалізовано для перегляду розкладу за аудиторіями. На індексній сторінці користувач може вибрати конкретну аудиторію та перейти до перегляду її зайнятості. Такий функціонал є важливим для контролю аудиторного фонду, оскільки дозволяє швидко визначити, які заняття проводяться в певному приміщенні та в які часові проміжки.

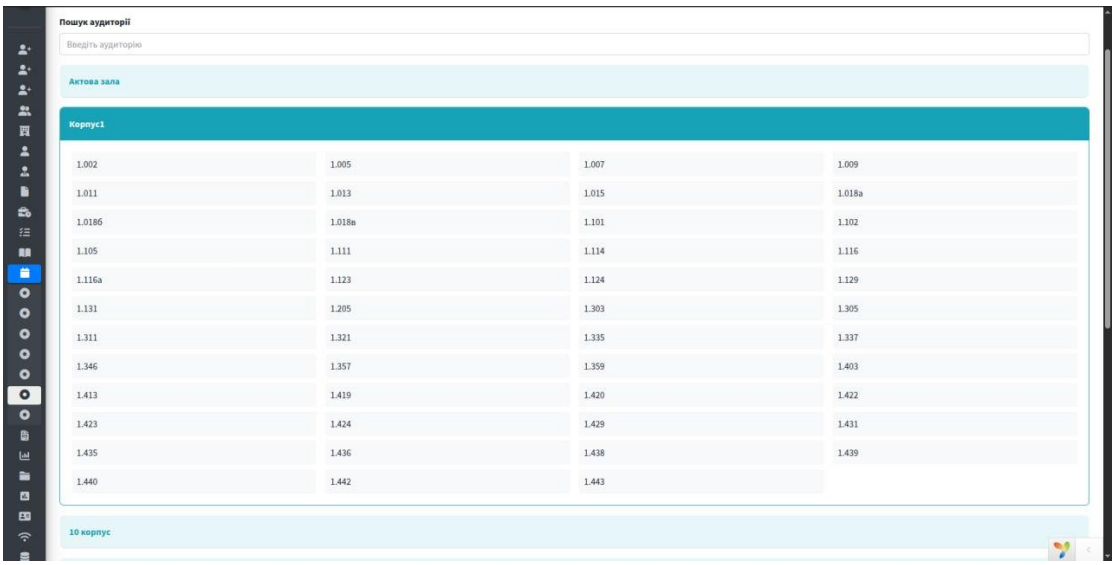

Рис. 3.17 – Сторінка вибору аудиторії для перегляду розкладу

Сторінка перегляду розкладу аудиторії дає змогу оцінити її використання протягом навчального тижня. У таблиці відображаються дисципліни, групи, викладачі та час проведення занять. Це спрощує пошук вільних аудиторій, перевірку коректності призначень і виявлення ситуацій, коли одна аудиторія може бути помилково зайнята кількома заняттями одночасно.

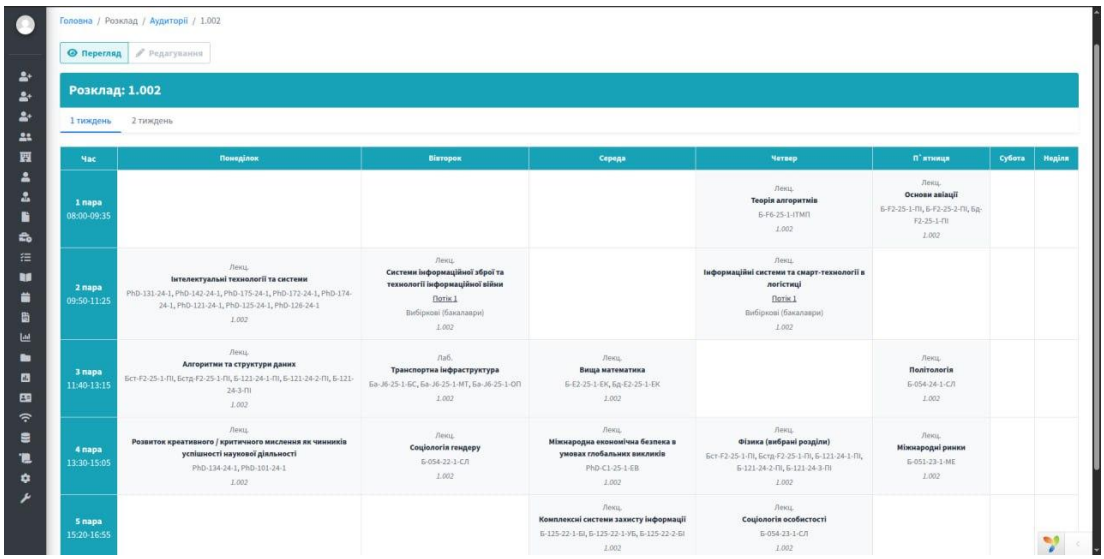

Рис. 3.18 – Табличне представлення розкладу аудиторії

Також у модулі реалізовано перегляд розкладу за викладачами. На індексній сторінці користувач може обрати конкретного працівника та перейти до його розкладу. Такий режим корисний для адміністрації, деканату, кафедр та інших відповідальних осіб, які контролюють навантаження викладачів або перевіряють наявність накладок у їхньому розкладі.

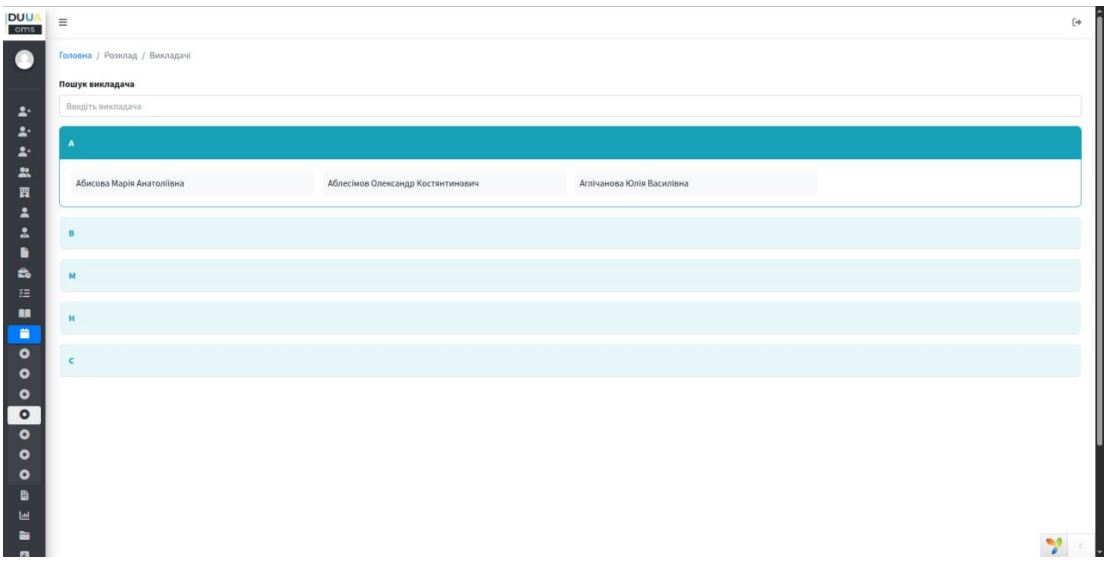

Рис. 3.19 – Сторінка вибору викладача для перегляду розкладу

Розклад викладача може відображатися у табличному та списковому форматах. Табличне представлення дозволяє побачити розподіл занять за днями і парами, а спискове представлення є зручним для перегляду занять послідовним переліком. Саме у списковому форматі доступна робота з додатковою інформацією до занять, зокрема з посиланнями та нотатками.

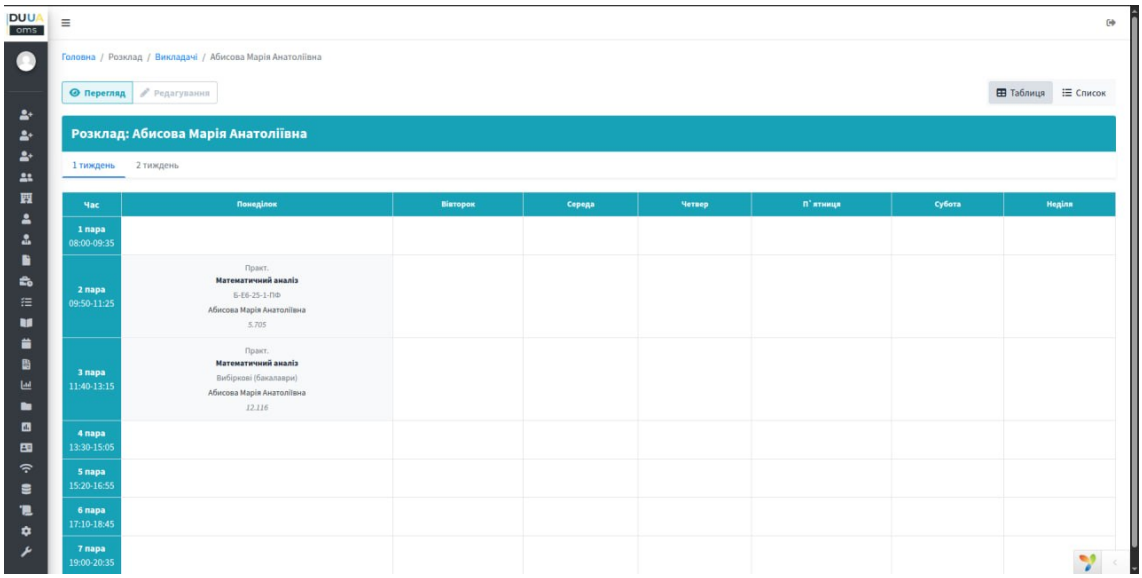

Рис. 3.20 – Табличне представлення розкладу викладача

Посилання можуть використовуватися для переходу до онлайн-заняття, електронного курсу або іншого навчального ресурсу. Нотатки дозволяють додати уточнення щодо проведення заняття, службову інформацію або інші коментарі. Доступ до таких дій перевіряється відповідно до прав користувача, зокрема з урахуванням того, чи належить відповідний розклад викладачеві або його підлеглому.

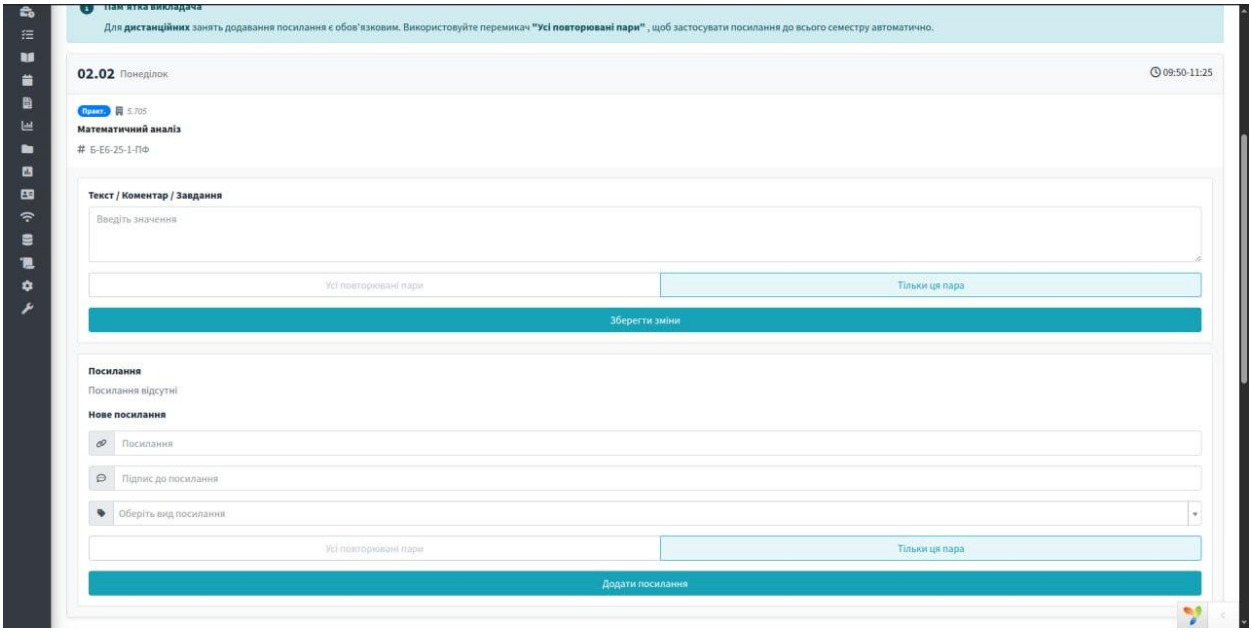

Рис. 3.21 – Спискове представлення розкладу викладача з посиланнями та нотатками

Для сторінок перегляду розкладу за групами, аудиторіями та викладачами передбачено спільний набір дій із заняттями. Користувач із відповідними правами може створювати нові пари, редагувати вже наявні заняття, копіювати їх або переносити в інший часовий проміжок. Це дозволяє працювати з розкладом без необхідності переходити до окремого технічного інтерфейсу для кожної дії.

Створення нової пари виконується через окрему форму або модальне вікно. У ній користувач заповнює необхідні параметри заняття, зокрема дисципліну, тип заняття, академічні групи, викладачів, аудиторії, навчальний тиждень, день і номер пари. Після збереження система створює основний запис заняття та пов’язані записи з групами, викладачами й аудиторіями.

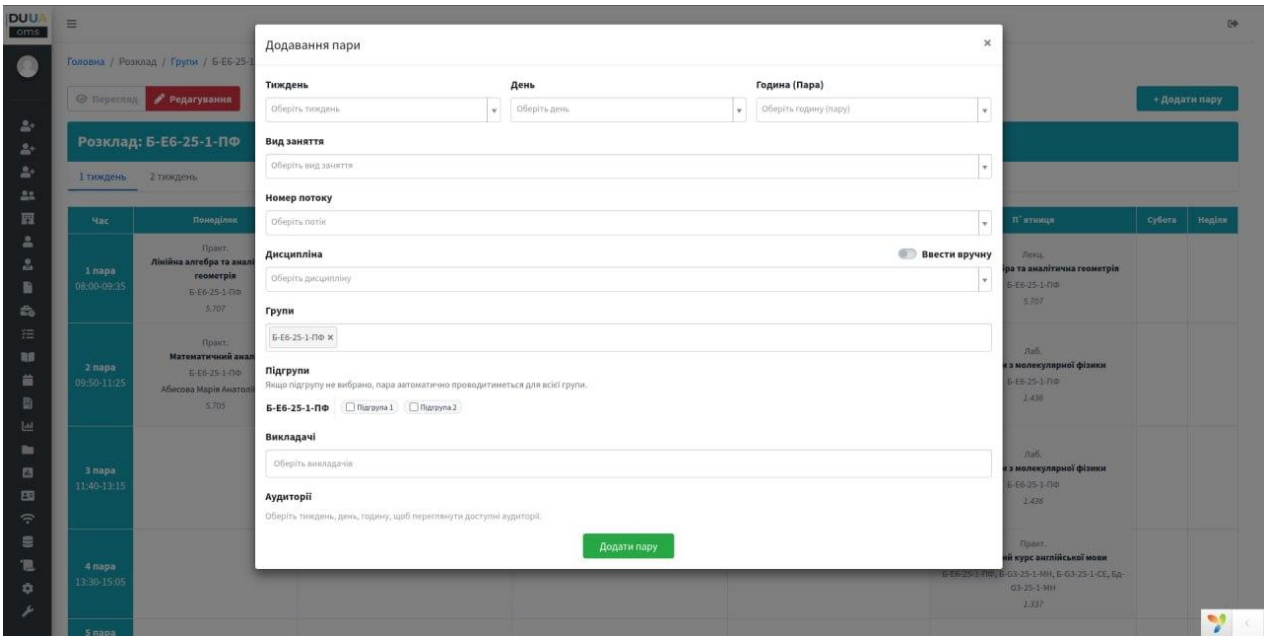

Рис. 3.22 – Модальне вікно створення навчального заняття

Редагування пари також виконується через модальне вікно, у якому можна змінити основні параметри заняття: дисципліну, тип заняття, викладачів, академічні групи, аудиторії, навчальний тиждень, день та номер пари. Використання модального вікна робить процес редагування швидшим, оскільки користувач залишається в контексті поточного розкладу та одразу бачить, яке заняття змінює.

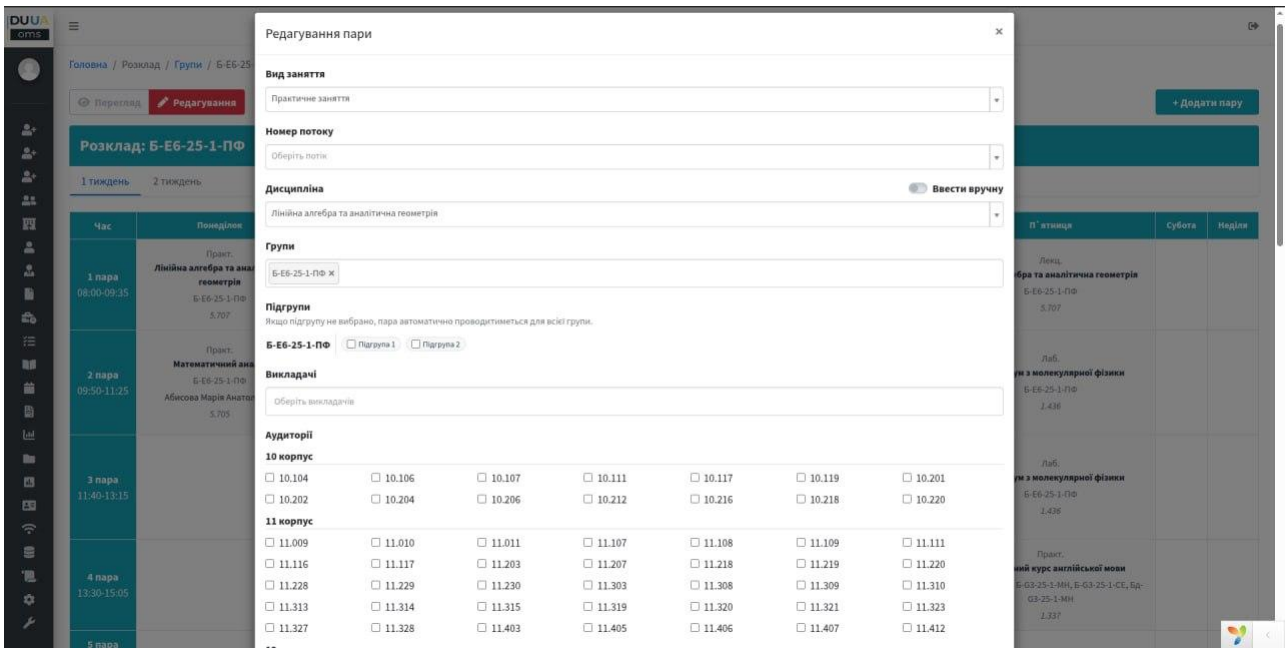

Рис. 3.23 – Модальне вікно редагування навчального заняття

Окрім створення та редагування, інтерфейс підтримує копіювання і перенесення пар. Копіювання використовується тоді, коли потрібно швидко створити нове заняття на основі вже наявного. Перенесення дозволяє змінити часову позицію пари, зберігши її основні зв’язки з групами, викладачами та аудиторіями. Такі дії є важливими для оперативного внесення змін у розклад без потреби повторного введення всіх даних вручну. Для цього достатньо натиснути на клітинку, в яку користувач бажає скопіювати чи перенести пару.

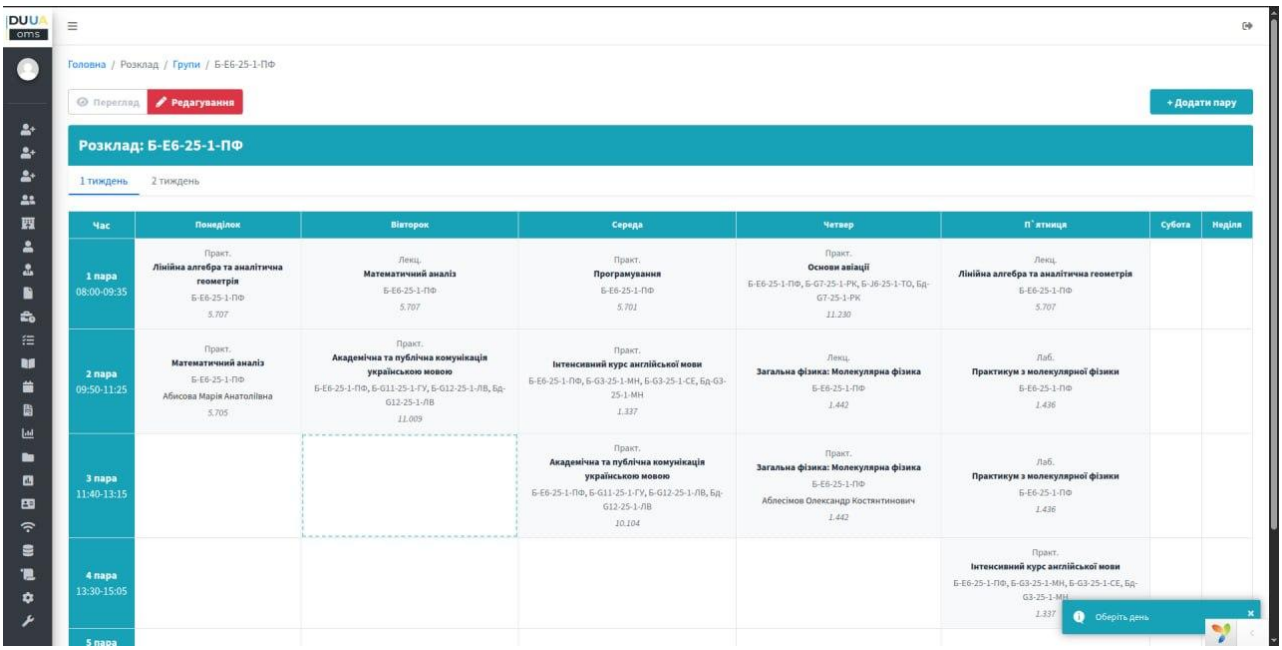

Рис. 3.24 – Елементи керування для копіювання та перенесення пари

Для викладачів у системі передбачено окрему сторінку «Мій розклад». Вона призначена для швидкого перегляду власних занять без необхідності вручну шукати себе у списку викладачів. Після авторизації система визначає поточного користувача, зіставляє його з відповідним працівником і автоматично формує розклад занять, у яких він зазначений як викладач.

За своєю логікою сторінка «Мій розклад» повторно використовує механізм перегляду розкладу викладача, однак застосовує автоматичну фільтрацію за поточним користувачем. Це робить сторінку зручною для щоденного використання викладачем. У списковому форматі на цій сторінці також може бути доступна робота з посиланнями та нотатками до занять, якщо користувач має відповідні права.

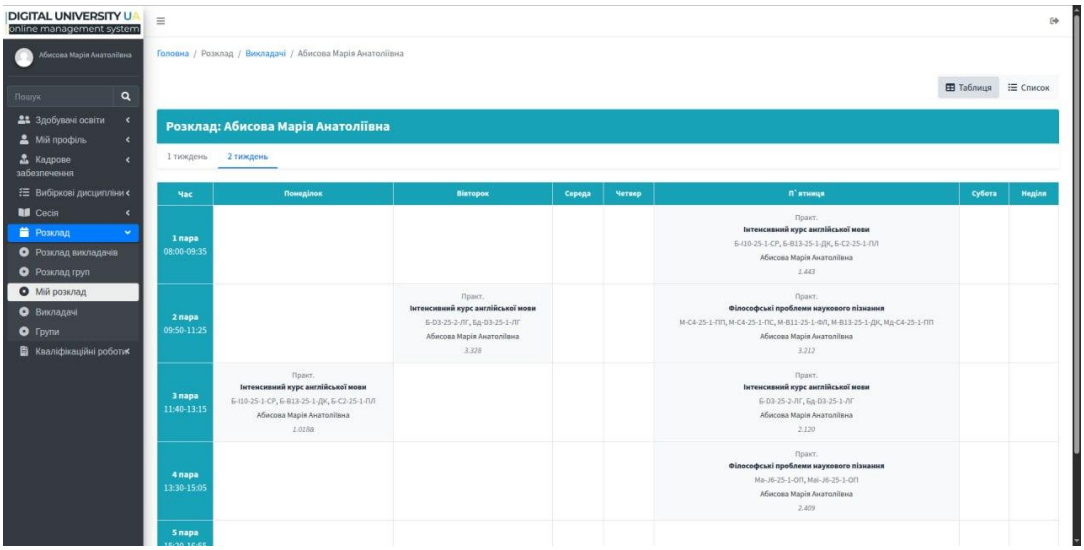

Рис. 3.25 – Сторінка «Мій розклад» для викладача

Окрему роль у модулі відіграє сторінка «Контроль». Вона призначена для адміністративного перегляду всіх пар розкладу, що містяться в системі. На цій сторінці доступні фільтри за семестром, версією розкладу, академічною групою, викладачем, аудиторією, дисципліною, днем, тижнем та номером пари. Завдяки цьому користувач може швидко знайти потрібні заняття, перевірити коректність імпортованих даних або сформувати вибірку для подальшого експорту.

Важливо зазначити, що сторінка «Контроль» не використовується для редагування посилань і нотаток до занять. Її основне призначення полягає у перевірці пар за фільтрами, аналізі результатів імпорту, виявленні можливих неточностей та підготовці даних до експорту. Такий підхід дозволяє відокремити адміністративний контроль розкладу від роботи з додатковою інформацією, яка виконується у спискових представленнях розкладу.

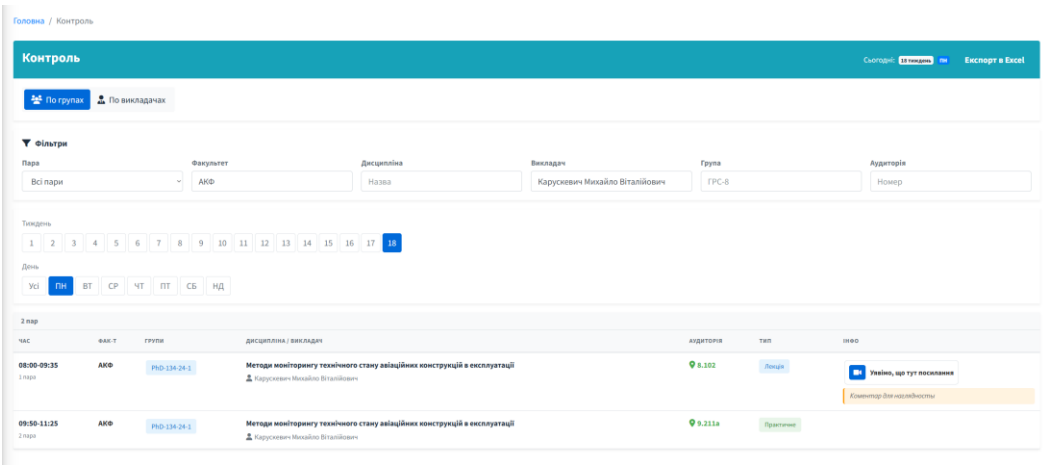

Рис. 3.26 – Сторінка «Контроль» з фільтрацією пар розкладу

На сторінці «Контроль» передбачено можливість експорту відфільтрованих даних у табличному форматі. Це дозволяє адміністративному користувачу спочатку сформувати потрібну вибірку за допомогою фільтрів, а потім зберегти її для подальшої перевірки, підготовки службових матеріалів або передавання інформації відповідальним підрозділам університету.

Експорт зі сторінки «Контроль» орієнтований саме на роботу з вибіркою даних за заданими параметрами. Наприклад, користувач може експортувати заняття певної академічної групи, викладача, аудиторії, семестру або конкретної версії розкладу. Це дає змогу не вивантажувати весь розклад повністю, а працювати лише з тією частиною даних, яка потрібна для перевірки або подальшої обробки.

A	B	C	D	E	F	G	H	I	J	K	L	M
Дата	День	Час	Навчальна дисципліна	Вид заняття	Аудиторія	Дистанційно / очно	Факультет викладача	Факультет групи	Академічні групи	К-ть студентів	Викладач	Інформація
01.06.2026	ПН 2 тиждень	08:00-09:35	Методи моніторингу технічного стану авіаційних конструкцій в експлуатації	Лекція	8.102	Очно	АКФ	АКФ	PhD-134-24-1	2	Карусевич Михайло Віталійович	Посилання: Уважно, що тут посилання: #placeholder# | Примітка: Коме
01.06.2026	ПН 2 тиждень	09:50-11:25	Методи моніторингу технічного стану авіаційних конструкцій в експлуатації	Практичне	9.211а	Очно	АКФ	АКФ	PhD-134-24-1	2	Карусевич Михайло Віталійович	

Рис. 3.27 – Експорт відфільтрованих пар зі сторінки «Контроль»

Для налаштування правил побудови розкладу в системі передбачені сторінки обмежень. Обмеження можуть задаватися окремо для академічних груп, викладачів та аудиторій. Вони використовуються для визначення часових проміжків, у які проведення занять є небажаним або забороненим. Надалі ці дані враховуються під час перевірки імпортованого FET/XML-файлу та допомагають виявляти помилки ще до публікації розкладу для користувачів.

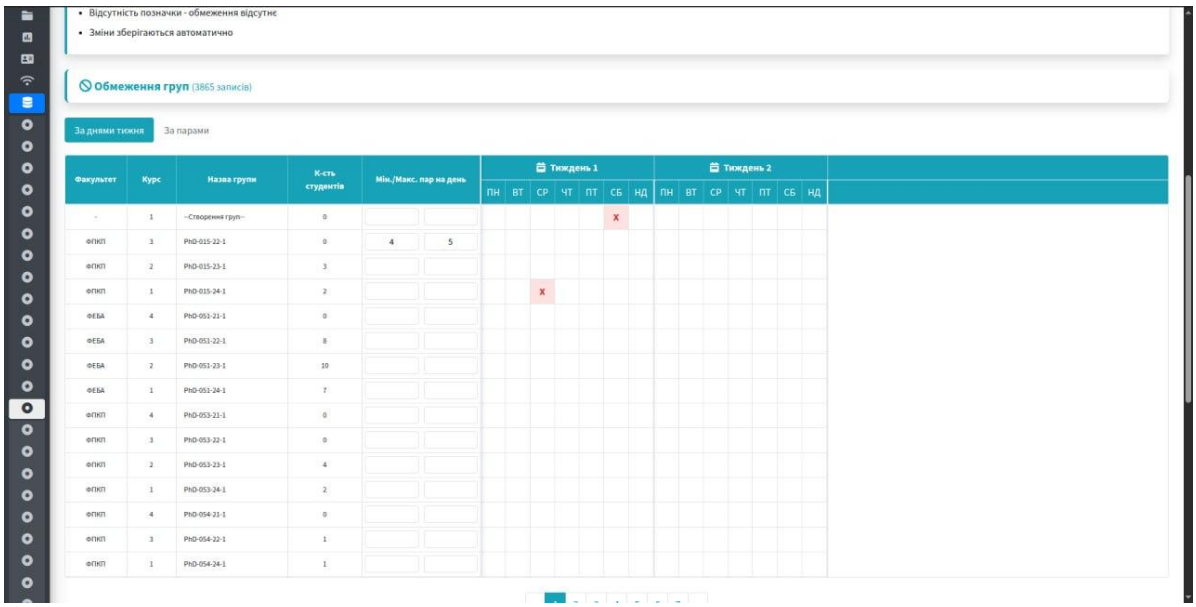

Рис. 3.28 – Сторінка налаштування обмежень для академічних груп

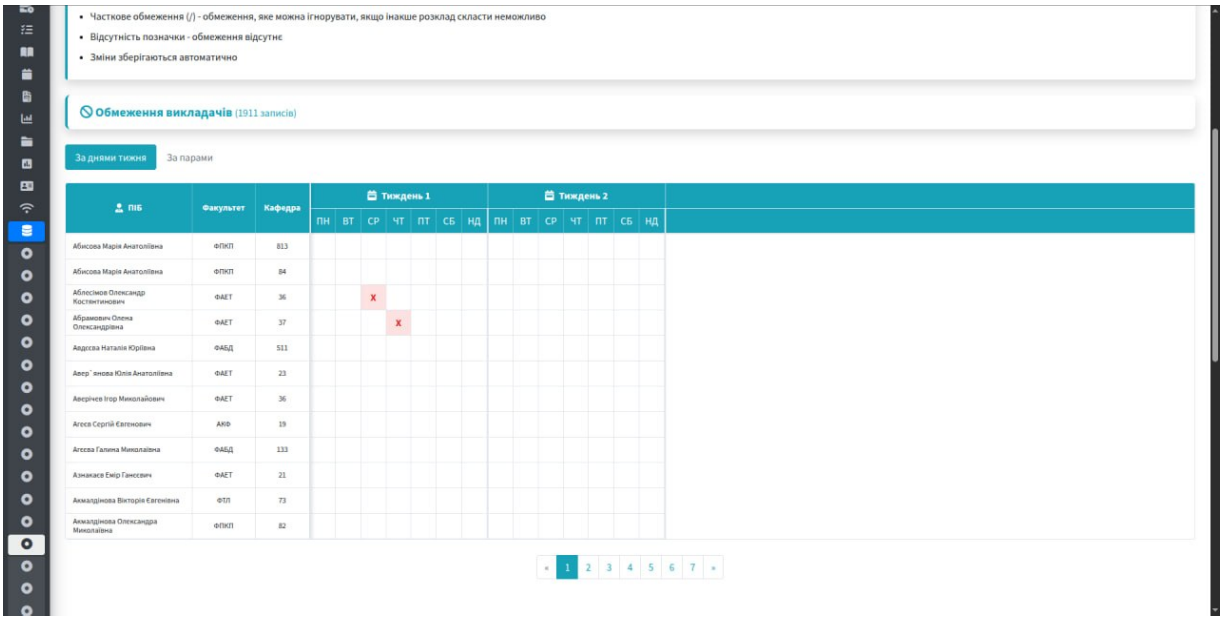

Рис. 3.29 – Сторінка налаштування обмежень для викладачів

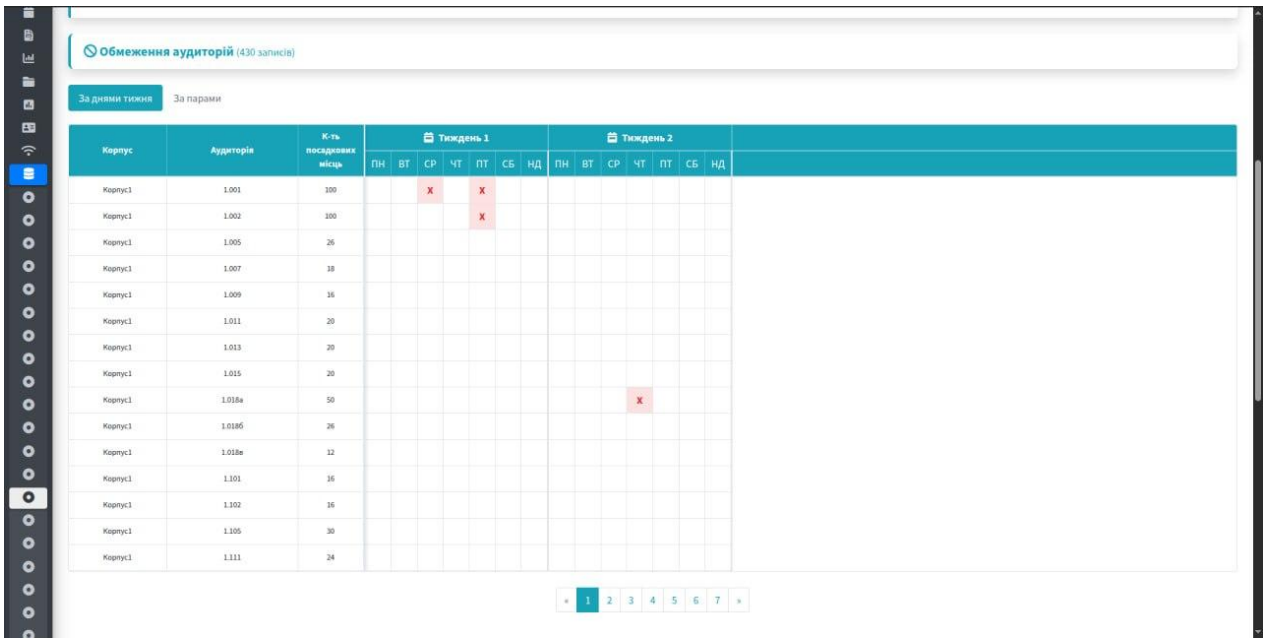

Рис. 3.30 – Сторінка налаштування обмежень для аудиторій

У випадку порушення обмежень або виникнення накладок система може відображати відповідні повідомлення. Наприклад, конфлікт може виникнути тоді, коли одна академічна група, викладач або аудиторія призначені на кілька занять в один і той самий час. Такі повідомлення допомагають користувачу швидко виявити проблему та внести необхідні зміни до розкладу.

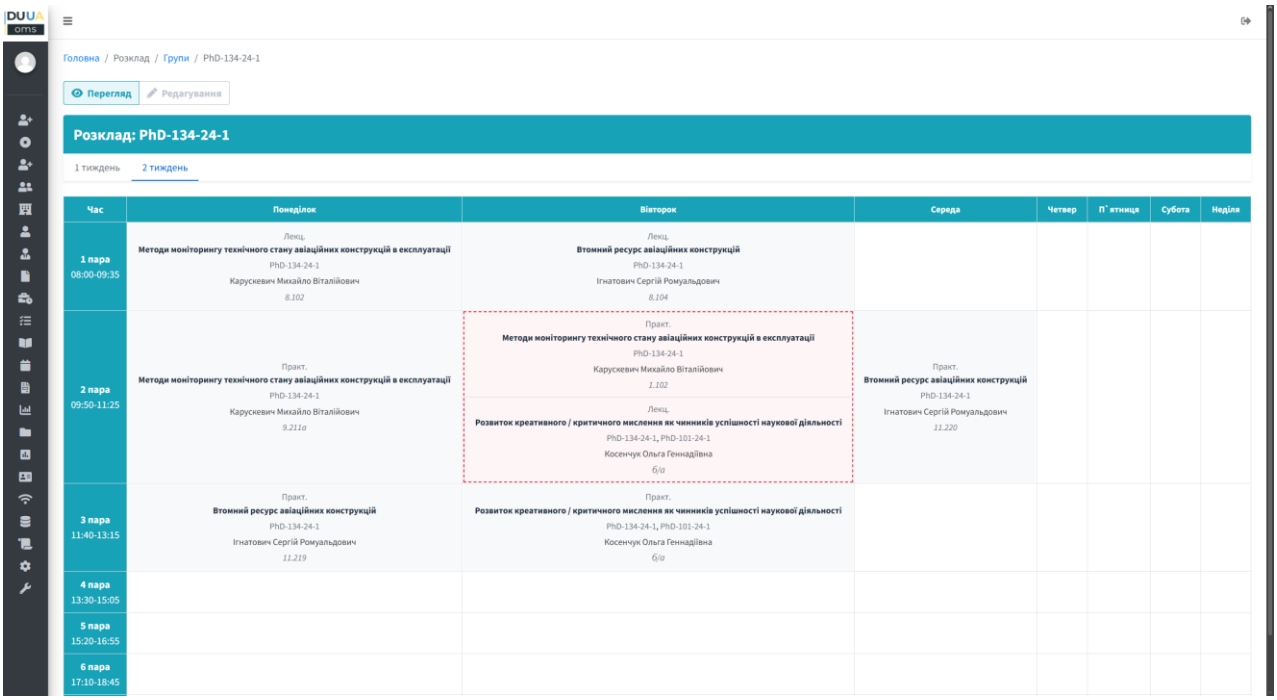

Рис. 3.31 – Відображення конфліктів у розкладі

Для студентів основним сценарієм роботи з модулем є перегляд персоналізованого розкладу в особистому кабінеті. Такий розклад формується на основі занять академічної групи та вибіркових дисциплін, які обрав конкретний студент. Завдяки цьому в особистому кабінеті відображаються лише ті заняття, які стосуються конкретного здобувача освіти.

Студентський розклад може відображатися у двох форматах: табличному та списковому. Табличне представлення є основним і дозволяє переглядати заняття за днями, тижнями та часовими парами. Спискове представлення є додатковим і подає заняття послідовним переліком, що може бути зручним для швидкого перегляду найближчих занять або використання на пристроях із меншою шириною екрана.

У персоналізованому розкладі також можуть відображатися додаткові дані до занять, зокрема посилання на онлайн-ресурси або нотатки, якщо вони були додані користувачами з відповідними правами. Це робить розклад не лише таблицею занять, а й зручним інформаційним інструментом для щоденної роботи студента.

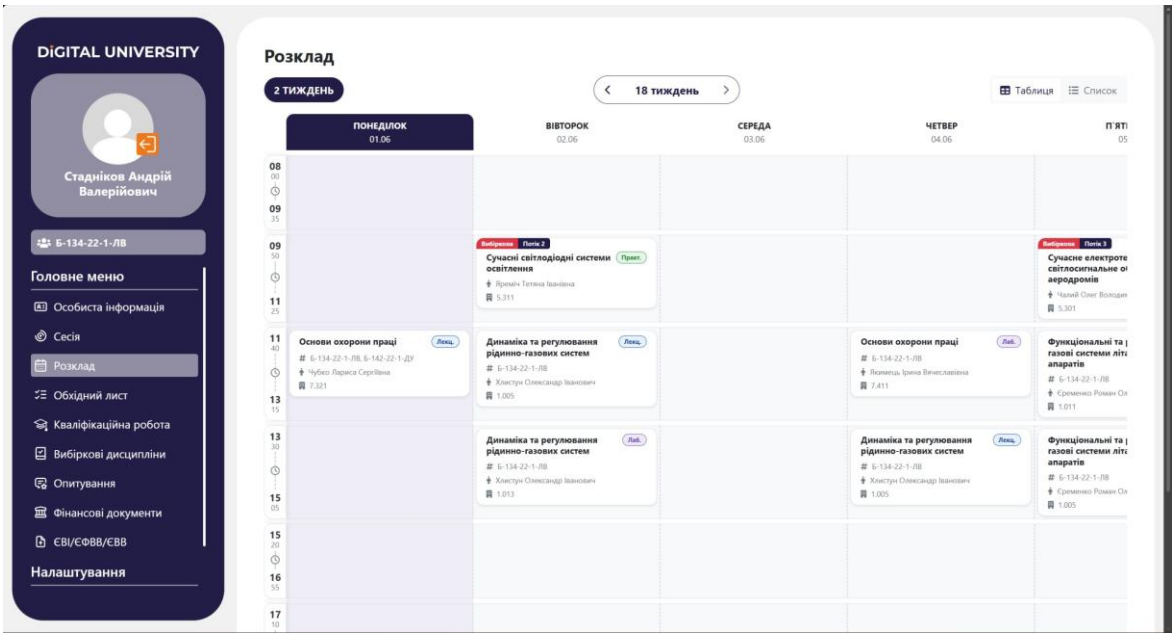

Рис. 3.32 – Табличне представлення персоналізованого розкладу студента

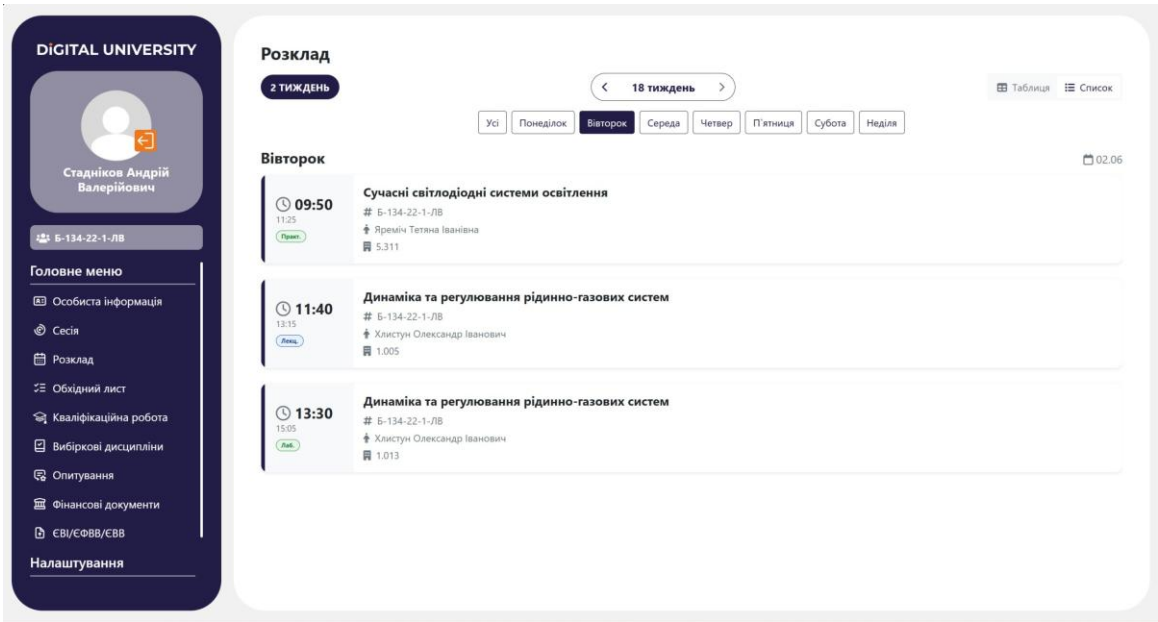

Рис. 3.33 – Списокове представлення персоналізованого розкладу студента

Таким чином, інтерфейс модуля «Розклад занять» забезпечує повний набір інструментів для роботи з навчальним розкладом у межах інформаційної системи Digital University UA. Користувачі можуть переглядати актуальний розклад відповідно до своєї ролі, а адміністративні працівники — імпортувати дані, керувати версіями, редагувати заняття, працювати з фільтрами, налаштовувати обмеження та контролювати конфлікти. Окремо на сторінці «Контроль» реалізовано експорт відфільтрованих пар у табличному форматі. Додатково модуль підтримує табличне і списокове представлення розкладу, сторінку «Мій

розклад» для викладачів, персоналізований розклад студента, а також роботу з посиланнями й нотатками у тих представленнях, де доступний списковий формат.

3.3. Тестування роботи розробленого додатку

У межах розробки модуля «Розклад занять» інформаційної системи Digital University UA було проведено ручне функціональне тестування, що передбачає перевірку основного функціоналу без використання автоматизованих тестових скриптів[31]. Такий підхід дозволив наблизити процес перевірки до реальних умов використання системи, оцінити коректність роботи інтерфейсу, логіку навігації та стабільність виконання основних операцій різними категоріями користувачів.

Основною метою тестування було підтвердження того, що модуль коректно виконує ключові функції, пов’язані з імпортом розкладу з FET/XML-файлу, відображенням розкладу за групами, викладачами та аудиторіями, формуванням персоналізованого розкладу студента, редагуванням занять, керуванням версіями, перевіркою конфліктів, роботою з посиланнями й нотатками, кешуванням даних та експортом відфільтрованої інформації у форматі Excel.

На початковому етапі тестування було перевірено авторизацію користувачів та доступ до модуля відповідно до ролей. Для цього виконувалася перевірка входу до системи від імені студента, викладача та адміністративного користувача. Після введення коректних облікових даних система успішно перенаправляла користувача до особистого кабінету та відображала доступні розділи відповідно до його прав. Студент отримував доступ до персоналізованого розкладу, викладач — до сторінки «Мій розклад», а адміністративний користувач — до розширеного набору сторінок модуля, зокрема генератора розкладу, сторінки «Контроль», сторінки версій та сторінок перегляду розкладу за групами, викладачами й аудиторіями. Перевірка

підтвердила, що авторизація працює коректно, а доступ до функціоналу відповідає ролі користувача.

Окремо було протестовано процес імпорту розкладу з FET/XML-файлу на сторінці генератора розкладу. Адміністративний користувач обирав семестр, завантажував файл і запускав процес імпорту. У ході перевірки встановлено, що система коректно приймає файл, обробляє його структуру, зіставляє отримані дані з внутрішніми довідниками та формує відповідну версію розкладу в базі даних. Додатково було перевірено відображення інформації про хід імпорту, зокрема часу виконання, кількості оброблених пар і статусу завершення операції. Після успішного завершення імпорту створена версія розкладу ставала доступною для подальшого перегляду, редагування та перевірки.

Під час тестування імпорту також було перевірено обробку помилок. Для цього використовувався FET/XML-файл, у якому окремі заняття порушували попередньо встановлені обмеження розкладу. Зокрема, перевірялися ситуації, коли заняття було призначене на часовий проміжок, обмежений для академічної групи, викладача або аудиторії. Система коректно виявляла такі порушення та відображала повідомлення про помилку на сторінці імпорту. Крім того, було перевірено можливість експорту деталей помилки у текстовий файл формату .txt. Отриманий файл містив перелік виявлених проблем і міг бути використаний для подальшого виправлення початкового файлу розкладу.

Наступним етапом було перевірено відображення розкладу за академічною групою. Після вибору потрібної групи система формувала табличне представлення занять із зазначенням днів тижня, номерів пар, дисциплін, типів занять, викладачів та аудиторій. Особливу увагу було приділено тому, що за замовчуванням на сторінці відображалася саме активна та видима версія розкладу. Після зміни активної версії на сторінці керування версіями розклад групи оновлювався відповідно до нового актуального варіанта. Це підтвердило коректність роботи механізму версійності та правильне відображення актуальної інформації для користувачів.

Для викладачів було перевірено роботу сторінки «Мій розклад». Після авторизації система автоматично визначала поточного користувача, зіставляла його з відповідним працівником у системі та формувала перелік занять, у яких він зазначений як викладач. Тестування показало, що викладач бачить власний розклад без необхідності ручного вибору себе зі списку. У розкладі коректно відображалися дисципліни, академічні групи, аудиторії, час проведення занять, а також додаткова інформація у вигляді посилань або нотаток, якщо вони були прикріплені до відповідної пари.

Також було протестовано сторінки перегляду розкладу за викладачами та аудиторіями. На сторінці розкладу викладача система дозволяла обрати потрібного працівника та переглянути всі пов'язані з ним заняття. На сторінці аудиторій користувач міг обрати конкретну аудиторію та перевірити її зайнятість у межах навчального тижня. Під час перевірки встановлено, що фільтрація працює коректно, а дані відображаються відповідно до обраної сутності. Це дозволяє використовувати такі сторінки не лише для перегляду, а й для контролю можливих накладок у розкладі.

У межах перевірки адміністративного функціоналу було протестовано ручне створення нового заняття через інтерфейс модуля. Користувач із відповідними правами заповнював форму, у якій зазначав тиждень, день, номер пари, дисципліну, тип заняття, академічні групи, викладачів та аудиторії. Після збереження система створювала основний запис заняття та пов'язані записи для груп, викладачів і аудиторій. Додатково було перевірено ситуації з неповним заповненням форми. Якщо обов'язкові поля залишалися порожніми, система не дозволяла зберегти заняття та відображала повідомлення про помилку.

Окремо було перевірено редагування, копіювання та перенесення вже створених занять. У процесі редагування змінювалися дисципліна, викладач, аудиторія, академічна група, день та номер пари. Після збереження зміни коректно відображалися у відповідних представленнях розкладу. Під час копіювання система створювала нову пару на основі існуючої, що дозволяло швидко додавати однотипні заняття без повторного введення всіх параметрів.

Перенесення також виконувалося коректно: після зміни тижня, дня або номера пари заняття відображалося в новій позиції розкладу.

Важливою частиною тестування стала перевірка механізму виявлення конфліктів. Для цього створювалися ситуації, коли одна академічна група, викладач або аудиторія були призначені на кілька занять в один і той самий час. Система коректно визначала такі накладання та відображала повідомлення про конфлікт. Окремо було перевірено роботу з підгрупами. Якщо заняття були призначені для різних підгруп однієї академічної групи, система не визначала це як помилку за умови, що вони не перетиналися із заняттям для всієї групи. Така логіка відповідає реальній організації навчального процесу.

Сторінка «Контроль» перевірялася як інструмент адміністративного перегляду всіх пар розкладу за допомогою фільтрів. Під час тестування використовувалися фільтри за семестром, версією розкладу, академічною групою, викладачем, аудиторією, дисципліною, днем, тижнем і номером пари. Система коректно звужувала вибірку відповідно до заданих параметрів і відображала лише ті заняття, які відповідали умовам фільтрації. Це підтвердило зручність сторінки для адміністративного контролю, пошуку помилок, аналізу результатів імпорту та підготовки даних до експорту.

Функціонал додавання та редагування посилань і нотаток було перевірено у спискових представленнях розкладу, зокрема на сторінці розкладу викладача та на сторінці «Мій розклад». До окремої пари було додано посилання на онлайн-ресурс і текстову нотатку. Після збереження ці дані коректно відображалися у відповідному занятті. Також було протестовано зміну вже доданих посилань і нотаток. Система коректно оновлювала інформацію, а зміни ставали доступними в інтерфейсі розкладу без порушення основних параметрів пари.

Під час тестування сторінки керування версіями було перевірено можливість перегляду списку створених версій, визначення активної версії, керування її видимістю та редагованістю. Після зміни активної версії відповідні сторінки перегляду розкладу відображали новий актуальний варіант. Також було перевірено можливість переходу до перегляду конкретної версії зі сторінки

«Версії», що дозволяє адміністрації аналізувати або редагувати не лише активну версію, а й інші збережені варіанти розкладу.

Для студентів було перевірено формування персоналізованого розкладу в особистому кабінеті. Система формувала такий розклад на основі загального розкладу академічної групи та вибіркових дисциплін, обраних конкретним студентом. У процесі тестування встановлено, що студент бачить лише ті заняття, які стосуються його академічної групи та індивідуального вибору. Вибіркові дисципліни, які не були обрані студентом, у персональному розкладі не відображалися.

Окремо було перевірено кешування персоналізованих студентських розкладів із використанням Redis. Після першого формування розкладу дані зберігалися в кеші, що дозволяло пришвидшити повторне відкриття сторінки. Також було перевірено, що після редагування заняття або зміни розкладу кешовані записи відповідних академічних груп інвалідуються, тобто видаляються з Redis і повторно формуються під час наступного відкриття розкладу. Це важливо для забезпечення актуальності інформації, оскільки студент після внесення змін повинен бачити оновлений розклад, а не застарілі дані з кешу. У разі недоступності Redis система не припиняла роботу, а формувала розклад безпосередньо з бази даних, що підтвердило стійкість модуля до тимчасових проблем із кеш-сховищем. Такий підхід дозволяє поєднати підвищення швидкодії з надійністю роботи системи навіть за умови часткової недоступності допоміжних сервісів.

Завершальним етапом тестування стала перевірка експорту відфільтрованих даних зі сторінки «Контроль» у форматі Excel[29;31]. Спочатку адміністративний користувач формував потрібну вибірку за допомогою фільтрів, після чого запускав експорт. Система формувала файл, який автоматично завантажувався на пристрій користувача. У файлі коректно відображалися дані про заняття, академічні групи, викладачів, аудиторії, дисципліни та часові проміжки[29]. Додатково було перевірено, що експортований файл містить саме ті записи, які відповідали застосованим фільтрам, без додавання зайвих або

пропущених даних. Це підтвердило коректність зв'язку між механізмом фільтрації на сторінці «Контроль» та логікою формування Excel-файлу. Окремо було перевірено коректність відображення українських символів у згенерованому файлі. Помилок кодування або втрати даних не виявлено.

Узагальнені результати ручного функціонального тестування, для зручності перегляду та порівняння очікуваних і фактичних результатів виконання основних сценаріїв, наведено в таблиці 3.1.

Таблиця 3.1.

Результати тестування модуля «Розклад занять»

№	Сценарій тестування	Очікуваний результат	Фактичний результат
1	Авторизація та перевірка ролей	Користувач бачить доступні сторінки відповідно до ролі	Успішно
2	Імпорт FET/XML-файлу	Розклад імпортується та створюється відповідна версія	Успішно
3	Обробка помилок імпорту	Система показує помилки та дозволяє експортувати їх у .txt	Успішно
4	Перегляд розкладу групи	Відображається розклад обраної академічної групи	Успішно
5	Сторінка «Мій розклад»	Викладач бачить власний розклад	Успішно
6	Перегляд за викладачами та аудиторіями	Дані фільтруються за обраною сутністю	Успішно
7	Створення заняття	Пара створюється з усіма зв'язками	Успішно
8	Редагування, копіювання та перенесення	Зміни коректно відображаються в розкладі	Успішно
9	Перевірка конфліктів	Система виявляє накладання груп, викладачів і аудиторій	Успішно

продовж. таблиці 3.1.

10	Сторінка «Контроль»	Пари фільтруються за заданими параметрами	Успішно
11	Посилання та нотатки	Додаткова інформація зберігається та відображається у розкладі	Успішно
12	Керування версіями	Активна версія відображається за замовчуванням	Успішно
13	Персоналізований розклад студента	Враховуються група та вибіркові дисципліни	Успішно
14	Redis-кешування	Повторне відкриття розкладу виконується швидше, кеш оновлюється після змін	Успішно
15	Експорт в Excel	Файл формується та відкривається без втрати даних	Успішно

Висновки до третього розділу

У третьому розділі було розглянуто порядок розгортання, налаштування та практичного використання модуля «Розклад занять» у межах інформаційної системи Digital University UA. Оскільки модуль є складовою частиною загальної платформи, його встановлення окремо від основного вебзастосунку не передбачається. Було описано, що для коректної роботи системи необхідне налаштоване серверне середовище, база даних MySQL, кеш-сховище Redis, Docker-середовище та наявність базових довідкових даних, зокрема семестрів, академічних груп, викладачів, аудиторій, дисциплін, типів занять і часових проміжків.

Окрему увагу приділено початковому наповненню модуля даними. Було показано, що робота з розкладом починається з імпорту FET/XML-файлу, сформованого стороннім програмним забезпеченням для первинної генерації

розкладу. У процесі імпорту система зіставляє отримані дані з внутрішніми сутностями бази даних, створює відповідну версію розкладу та дозволяє надалі переглядати, редагувати, перевіряти й публікувати її для користувачів. Також було зазначено, що під час імпорту можуть виникати помилки, пов'язані з порушенням установлених обмежень, а перелік таких помилок може бути експортований у текстовий файл для подальшого аналізу.

У межах опису інтерфейсу було продемонстровано основні сторінки модуля «Розклад занять» та порядок взаємодії користувачів із ними. Було розглянуто місце модуля в навігаційному меню системи, сторінку генератора розкладу, сторінку керування версіями, перегляд розкладу за академічними групами, викладачами та аудиторіями, а також сторінку «Мій розклад», яка призначена для швидкого перегляду власних занять викладачем. Для студентів основним сценарієм роботи є перегляд персоналізованого розкладу, сформованого з урахуванням академічної групи та обраних вибіркових дисциплін.

Також у розділі було описано адміністративні можливості модуля. Зокрема, розглянуто сторінку «Контроль», яка дозволяє переглядати всі пари розкладу за фільтрами, перевіряти коректність даних після імпорту та експортувати відфільтровану інформацію. Окремо було описано роботу з посиланнями й нотатками у спискових представленнях розкладу. Було показано, що модуль підтримує створення, редагування, копіювання та перенесення пар, керування активною версією розкладу, налаштування обмежень для груп, викладачів і аудиторій, а також експорт даних у форматі Excel.

У межах тестування було проведено ручну функціональну перевірку ключових сценаріїв роботи модуля. Було перевірено авторизацію користувачів і доступ до функціоналу відповідно до ролей, імпорт розкладу з FET/XML-файлу, обробку помилок імпорту, перегляд розкладу за різними сутностями, роботу сторінки «Мій розклад», створення та редагування занять, копіювання й перенесення пар, перевірку конфліктів, керування версіями, додавання посилань

і нотаток, формування персоналізованого розкладу студента, Redis-кешування та експорт даних у форматі Excel.

Результати тестування підтвердили коректність роботи основного функціоналу модуля «Розклад занять» та його відповідність визначеним вимогам. Усі ключові сценарії були виконані успішно, що свідчить про готовність модуля до використання в межах інформаційної системи Digital University UA. Таким чином, розроблений функціонал забезпечує зручне управління навчальним розкладом, підтримує різні ролі користувачів, підвищує ефективність адміністрування освітнього процесу та забезпечує студентів і викладачів актуальною інформацією про заняття.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було реалізовано повний цикл розробки функціонального модуля «Розклад занять» для інформаційної системи онлайн-менеджменту освітніх процесів Digital University UA. Основна мета роботи полягала у створенні програмного рішення, яке забезпечує централізоване формування, імпортування, редагування, контроль та відображення розкладу навчальних занять для різних категорій користувачів університету.

На початковому етапі було проаналізовано предметну область та визначено основні проблеми, пов’язані з організацією розкладу у закладах вищої освіти. Було встановлено, що ручне ведення розкладу або використання звичайних табличних редакторів не забезпечує належного рівня контролю, версійності, перевірки конфліктів, персоналізованого відображення та інтеграції з іншими складовими освітньої платформи. Саме тому було обґрунтовано доцільність розробки власного модуля, інтегрованого в загальну структуру Digital University UA.

У процесі аналізу аналогів було розглянуто як загальні системи підтримки освітнього процесу, так і спеціалізовані інструменти для формування розкладу. Встановлено, що наявні рішення можуть покривати лише окремі частини поставленої задачі, однак не забезпечують повної інтеграції з внутрішніми даними університету, особистими кабінетами студентів, вибірковими дисциплінами, рольовою моделлю доступу та механізмом керування версіями розкладу. Це підтвердило актуальність створення окремого модуля, адаптованого до потреб конкретної інформаційної системи.

На етапі проєктування було визначено функціональні та нефункціональні вимоги до розроблюваного програмного забезпечення. Було описано основні варіанти використання системи для студентів, викладачів, адміністрації та власників системи. Студенти отримують доступ до персоналізованого розкладу, викладачі — до сторінки «Мій розклад», а адміністративні користувачі — до інструментів імпорту, редагування, копіювання, перенесення пар, керування

версіями, перегляду сторінки «Контроль», перевірки конфліктів, налаштування обмежень, експорту та керування посиланнями й нотатками до занять.

Для реалізації модуля було обрано клієнт-серверну архітектуру з використанням архітектурного шаблону MVC, що відповідає загальній структурі фреймворку Yii2. Серверна частина реалізована мовою PHP з використанням Yii2, для зберігання основних даних застосовано MySQL, а для підвищення продуктивності під час формування персоналізованих розкладів використано Redis. Розгортання системи передбачено в Docker-середовищі, що спрощує налаштування, супровід та перенесення програмного продукту між різними середовищами.

У межах проєктування бази даних було сформовано реляційну структуру, яка забезпечує збереження версій розкладу, навчальних занять, зв’язків пар із академічними групами, викладачами та аудиторіями. Окремо передбачено таблиці для підтримки підгруп, вибіркових дисциплін, навчальних тижнів, днів, часових проміжків, обмежень, посилань і нотаток. Така структура дозволяє уникнути дублювання даних, підтримувати цілісність зв’язків між сутностями та забезпечувати подальший розвиток функціональності модуля.

Важливою частиною розробки стала реалізація імпорту розкладу з FET/XML-файлу. Такий підхід дозволяє використовувати стороннє програмне забезпечення для первинної генерації одного з варіантів розкладу, після чого дані імпортуються до Digital University UA та перетворюються у внутрішню структуру системи. Під час імпорту виконується зіставлення груп, викладачів, аудиторій, дисциплін і часових проміжків із записами бази даних. Також передбачено перевірку порушень обмежень розкладу та можливість експорту виявлених помилок у текстовий файл для подальшого виправлення.

У процесі реалізації було створено функціонал для перегляду розкладу за академічними групами, викладачами та аудиторіями. Окремо реалізовано сторінку «Мій розклад», яка дозволяє викладачу швидко переглядати власні заняття без ручного пошуку у загальному списку. Для студентів реалізовано

персоналізоване відображення розкладу з урахуванням загального розкладу академічної групи та обраних вибіркових дисциплін.

Для адміністративних користувачів реалізовано сторінку «Контроль», яка забезпечує перегляд усіх пар розкладу з використанням фільтрів, перевірку даних і експорт відфільтрованої інформації. Робота з посиланнями та нотатками до занять виконується у спискових представленнях розкладу.

Також реалізовано сторінку керування версіями розкладу, що дозволяє визначати активну, видиму та доступну для редагування версію. За замовчуванням на основних сторінках системи відображається саме активна версія розкладу, тоді як адміністрація може переходити до перегляду конкретних версій.

Окрему увагу було приділено механізму перевірки конфліктів. Система дозволяє виявляти ситуації, коли одна академічна група, викладач або аудиторія одночасно задіяні у кількох заняттях. При цьому враховується логіка підгруп, що дозволяє уникати помилкових конфліктів у випадках, коли різні підгрупи однієї групи мають окремі заняття в один і той самий час. Такий механізм підвищує якість розкладу та зменшує кількість ручних перевірок.

У завершальній частині роботи було розглянуто інтерфейс модуля та порядок роботи з ним. Було описано розгортання системи, налаштування MySQL і Redis, підготовку довідкових даних, імпорт початкового розкладу, роботу зі сторінками перегляду, редагування, контролю, версій та обмежень. Також було проведено ручне функціональне тестування основних сценаріїв роботи модуля. У ході тестування перевірено авторизацію користувачів, імпорт FET/XML-файлу, обробку помилок імпорту, перегляд розкладу, створення, редагування, копіювання та перенесення пар, перевірку конфліктів, роботу з посиланнями й нотатками, кешування в Redis та експорт розкладу у форматі Excel.

Результати тестування підтвердили працездатність розробленого модуля та його відповідність визначеним функціональним вимогам. Усі ключові сценарії були виконані успішно, що свідчить про готовність модуля «Розклад занять» до використання в межах інформаційної системи Digital University UA. Таким

чином, поставлена мета кваліфікаційної роботи була досягнута: розроблено програмний модуль, який забезпечує ефективне управління навчальним розкладом, підтримує різні ролі користувачів, зменшує кількість ручних операцій, підвищує точність даних та забезпечує студентів і викладачів актуальною інформацією про навчальні заняття.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Семеняко Ю. Б., Брюховецька І. В., Бохонько О. Є. Цифровізація у вищій освіті: інституційні підходи до викладання та навчання. Класичний приватний університет, 2023. 186 с.

2. Цифрова трансформація освіти і науки. URL: https://mon.gov.ua/tag/tsifrova-transformatsiya-osviti-i-nauki (дата звернення: 23.06.2026).

3. Moodle. URL: https://moodle.org/ (дата звернення: 23.06.2026).

4. UniTime: Comprehensive Educational Scheduling System. URL: https://www.unitime.org/ (дата звернення: 23.06.2026).

5. UniTime Documentation. URL: https://help.unitime.org/ (дата звернення: 23.06.2026).

6. FET – Free Timetabling Software. URL: https://lalescu.ro/liviu/fet/ (дата звернення: 23.06.2026).

7. aSc Timetables. URL: https://www.asctimetables.com/ (дата звернення: 23.06.2026).

8. The Definitive Guide to Yii 2.0. URL: https://www.yiiframework.com/doc/guide/2.0/en (дата звернення: 23.06.2026).

9. Yii 2.0 API Documentation. URL: https://www.yiiframework.com/doc/api/2.0 (дата звернення: 23.06.2026).

10. PHP Manual. URL: https://www.php.net/manual/en/ (дата звернення: 23.06.2026).

11. Mann E. PHP Cookbook: Modern Code Solutions for Professional Developers. O’Reilly Media, 2023. 420 p.

12. Nixon R. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. O’Reilly Media, 2021. 826 p.

13. Duckett J. PHP & MySQL: Server-side Web Development. Wiley, 2022. 672 p.

14. MySQL 8.4 Reference Manual. URL: https://dev.mysql.com/doc/refman/8.4/en/ (дата звернення: 23.06.2026).
15. Forta B. MySQL Crash Course. Addison-Wesley Professional, 2024. 400 p.
16. Hernandez M. J. Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design. Addison-Wesley Professional, 2021. 672 p.
17. Karwin B. SQL Antipatterns: Avoiding the Pitfalls of Database Programming. Pragmatic Bookshelf, 2010. 328 p.
18. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002. 560 p.
19. Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma, R. Helm, R. Johnson, J. Vlissides. Addison-Wesley Professional, 1994. 416 p.
20. Wiegers K., Beatty J. Software Requirements. Microsoft Press, 2013. 672 p.
21. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional, 2018. 208 p.
22. Skelton M., Paisley A. Creating Software with Modern Diagramming Techniques. Pragmatic Bookshelf, 2021. 280 p.
23. Robbins J. N. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript and Web Graphics. O'Reilly Media, 2021. 640 p.
24. JavaScript Guide – MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 23.06.2026).
25. Docker Documentation. URL: https://docs.docker.com/ (дата звернення: 23.06.2026).
26. Redis Documentation. URL: https://redis.io/docs/latest/ (дата звернення: 23.06.2026).
27. Carlson J. L. Redis in Action. Manning Publications, 2013. 320 p.

28. Kleppmann M. Designing Data-Intensive Applications. O’Reilly Media, 2017. 616 p.
29. PhpSpreadsheet Documentation. URL: https://phpspreadsheet.readthedocs.io/ (дата звернення: 23.06.2026).
30. PHPUnit Documentation. URL: https://phpunit.de/documentation.html (дата звернення: 23.06.2026).
31. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. Wiley, 2011. 256 p.
32. OWASP Web Security Testing Guide. URL: https://owasp.org/www-project-web-security-testing-guide/ (дата звернення: 23.06.2026).
33. Extensible Markup Language (XML) 1.0 Specification. URL: https://www.w3.org/TR/xml/ (дата звернення: 23.06.2026).
34. JSON. URL: https://www.json.org/json-en.html (дата звернення: 23.06.2026).