Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра комп'ютерних інтелектуальних технологій та систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)
Клієнт-серверне програмне забезпечення для планування адаптивного навчального розкладу студентів та викладачів
(тема)

Виконав:
здобувач IV року навчання,
групи КІУКІ-21-10
Віталій КУЧЕРЕНКО
(власне ім'я, прізвище)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма
Комп'ютерна Інженерія
(повна назва освітньої програми)

Керівник ас. каф. КІТС Андрій ТАТАРНИКОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ Олег РУДЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Комп'ютерних інтелектуальних технологій та систем

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 25 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Кучеренку Віталію Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Клієнт-серверне програмне забезпечення для планування адаптивного навчального розкладу студентів та викладачів

затверджена наказом по університету від " 21 " травня 2025 р. № 399 Ст

2. Термін подання студентом роботи до екзаменаційної комісії

3. Вхідні дані до роботи

1. Документація мови програмування C#.
2. Документація технології WPF (Windows Presentation Foundation).
3. Документація SQL Server та SQL-запитів.
4. Середовище розробки Visual Studio (з підтримкою .NET).
5. Бібліотеки для візуалізації даних у WPF.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз предметної області.
2. Аналіз використовуваних технологій.
3. Програмна реалізація інтерактивного інтерфейсу та роботи з даними.
4. Інструкція користувача.
5. Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
Слайд – презентація – 16 слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	26.05.2025-28.05.2025	Виконано
2	Вибір технологій розробки,	29.05.2025-31.05.2025	Виконано
	інструментальних засобів		
3	Створення тренувальних наборів даних	01.06.2025-02.06.2025	Виконано
4	Тренування моделей	03.06.2025-07.06.2025	Виконано
5	Розробка серверної та клієнтської частин	08.06.2024-10.06.2025	Виконано
6	Тестування застосунку	11.06.2025-12.06.2025	Виконано
7	Оформлення матеріалів атестаційної роботи	13.06.2025-16.06.2025	Виконано
8	Подання атестаційної роботи керівникові та	17.06.2025-18.06.2025	Виконано
	її попередній захист		
9	Подання роботи на рецензування	19.06.2025-20.06.2025	Виконано

Дата видачі завдання 26 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Андрій ТАТАРНИКОВ
(підпис) (посада, власне ім'я, прізвище

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 75 с., 34 рис., 3 табл., 2 дод., 30 джерел.

РОЗКЛАД, СТУДЕНТ, ВИКЛАДАЧ, АУДИТОРІЯ, ДИСЦИПЛІНА, БАЗА ДАНИХ, АДАПТАЦІЯ, WPF, SQL SERVER, АЛГОРИТМ.

Метою атестаційної роботи є клієнт-серверне програмне забезпечення для планування адаптивного навчального розкладу студентів та викладачів.

У ході виконання атестаційної роботи були проаналізовані існуючі підходи до автоматизації планування навчального розкладу та визначені їхні ключові недоліки – зокрема, обмежена адаптивність до змін та складність у врахуванні індивідуальних побажань. Це стало основою для формування функціональних та технічних вимог до розробленого застосунку.

Розроблена система була створена з використанням мови програмування C# та технології Windows Presentation Foundation для створення графічного інтерфейсу, Entity Framework для доступу до даних та SQL Server як засіб зберігання інформації. Система забезпечує автоматизоване формування розкладу з урахуванням доступності викладачів та побажань студентів, підтримує ручне редагування, виявлення конфліктів і надає аналітику змін. Застосунок пройшов повноцінне тестування з використанням реальних даних та може бути впроваджений в освітнє середовище.

ABSTRACT

Bachelor’s thesis: 75 pages, 34 figures, 3 tables, 2 appendices, 30 sources.

SCHEDULE, STUDENT, TEACHER, CLASSROOM, SUBJECT, DATABASE, ADAPTATION, WPF, SQL SERVER, ALGORITHM.

The goal of this thesis is to develop client-server software for planning adaptive class schedules for students and instructors.

During the development of this thesis, existing approaches to schedule automation were analyzed, and their key limitations were identified — in particular, limited adaptability to changes and difficulties in accounting for individual preferences. These findings served as the foundation for defining the functional and technical requirements of the application.

The developed system was implemented using the C# programming language and Windows Presentation Foundation (WPF) technology to create the graphical user interface, Entity Framework for data access, and SQL Server as the data storage platform. The system supports automated schedule generation based on instructor availability and student preferences, manual editing, conflict detection, and provides built-in analytics for tracking changes. The application has been fully tested using real data and is ready for deployment in an academic environment.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІС – інформаційна система

БД – база даних

GUI – графічний інтерфейс користувача (англ. Graphical User Interface)

WPF – Windows Presentation Foundation (технологія побудови інтерфейсу в середовищі .NET)

EF – Entity Framework (ORM-технологія доступу до даних у C#)

SQL – Structured Query Language, мова структурованих запитів

CRUD – операції створення, читання, оновлення та видалення даних (Create, Read, Update, Delete)

.NET – програмна платформа Microsoft для розробки застосунків

UI – користувацький інтерфейс (User Interface)

UX – досвід користувача (User Experience)

API – інтерфейс прикладного програмування (Application Programming Interface)

IDE – інтегроване середовище розробки (Integrated Development Environment)

ПЗ – програмне забезпечення

ПО – програмне оточення

КСЗІ – комплексна система захисту інформації

JSON – формат обміну даними (JavaScript Object Notation)

HTTP – протокол передачі гіпертексту (HyperText Transfer Protocol)

REST – архітектурний стиль для створення вебсервісів

MVC – архітектурний шаблон Model–View–Controller

FET – Free Timetabling Software

ХНУРЕ – Харківський Національний Університет Радіоелектроніки

ООП – об'єктно-орієнтоване програмування

IT – інформаційні технології

LMS – система управління навчанням (Learning Management System)

CLR – загальне середовище виконання (Common Language Runtime)

IIS – інформаційна служба Інтернету (Internet Information Services)

MVVM – Модель - Представлення - Модель Представлення (Model–View–ViewModel)

ВСТУП

Сучасні навчальні заклади стикаються з проблемою ефективного планування навчального процесу, розподілу аудиторного фонду та контролю за виконанням розкладу занять. Ручне складання розкладу вимагає значних витрат часу та ресурсів, а також часто призводить до помилок та неоптимального використання наявних ресурсів навчальних закладів.

Мета роботи полягає у розробці інформаційної системи управління розкладом навчального закладу, яка дозволяє автоматизувати процеси планування занять, оптимізувати використання аудиторного фонду та забезпечити ефективний контроль за навчальним процесом.

Для розроблення системи та досягнення поставленої мети необхідно:

– провести розгорнутий аналіз сучасних підходів, які використовуються для автоматизації управління навчальним розкладом;
– розробити архітектуру інформаційної системи;
– створити базу даних для зберігання необхідної інформації;
– реалізувати алгоритми автоматичного формування розкладу;
– створити систему контролю конфліктів у розкладі;
– розробити зручний користувацький інтерфейс;
– провести тестування та оптимізацію.

Використання створеної системи управління розкладом дозволить підвищити якість планування навчального процесу, оптимізувати використання ресурсів та покращити контроль за виконанням розкладу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз процесу планування навчального розкладу

Процес створення навчального розкладу розпочинається з отримання базової інформації від кафедр. Завідувачі кафедр надають дані про навантаження викладачів, кількість годин з кожної дисципліни та побажання щодо проведення занять. Методисти опрацьовують цю інформацію, перевіряючи відповідність навчальним планам та нормативним вимогам. Формується список аудиторій з урахуванням їх оснащення.

Наступним етапом виступає початкове планування розкладу на семестр. Методисти аналізують наявний аудиторний фонд та його завантаженість у попередні періоди. Складається матриця доступності аудиторій по днях тижня та парах. Враховуються особливості проведення лекційних, практичних та лабораторних занять. Перевіряється можливість об'єднання потоків студентів для проведення лекцій[1].

Процес узгодження розкладу з викладачами потребує детального опрацювання. Збираються побажання викладачів щодо зручного для них часу проведення занять. Враховується зайнятість викладачів на різних факультетах та в інших навчальних закладах. Перевіряється можливість проведення занять у вказаний викладачами час [2].

1.2 Огляд існуючих рішень для створення розкладу

1.2.1 ASC Timetables

Система ASC Timetables пропонує комплексне рішення для автоматизації створення розкладу . Програмний продукт містить зручний інтерфейс для введення даних про викладачів та предмети. Алгоритм

автоматично перевіряє можливі конфлікти при складанні розкладу. База даних зберігає інформацію про завантаженість аудиторій та викладачів.

Система генерує різні варіанти розкладу. Користувачі мають можливість вносити корективи в автоматично створений розклад. Програма дозволяє використовувати дані в різних форматах [3].

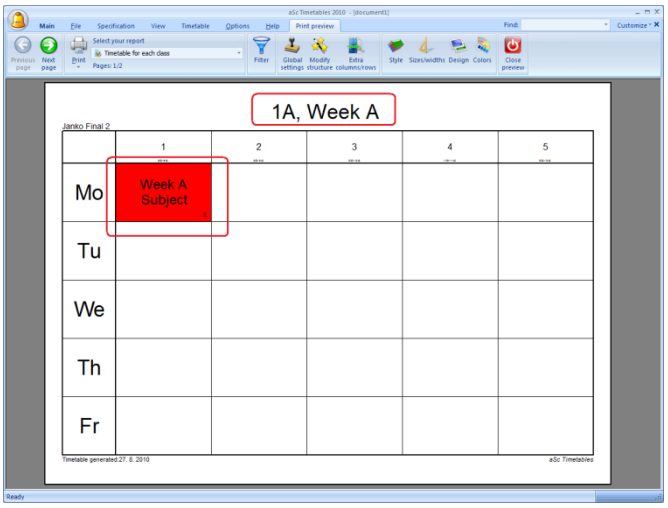

Рисунок 1.1 – Зовнішній вигляд інтерфейсу ASC Timetables

Одним з недоліків ASC Timetables є те, що зміни в розкладі оновлюються лише один раз на день, тому користувачам необхідно самостійно відслідковувати актуальність інформації. Крім того, система краще підходить для типової організації навчальних тижнів і потребує додаткових налаштувань у разі використання нетипових форматів. Треба також брати до уваги, що можливості створення складних індивідуальних розкладів обмежені.

1.2.2 FlexSchedule

Програмний продукт «FlexSchedule» забезпечує гнучке планування навчального процесу та індивідуальних графіків навчання. Користувачі можуть налаштовувати параметри планування під свої потреби. Передбачено

можливість роботи з модульною системою навчання [4]. Застосунок забезпечує гнучкий підхід до організації контрольних заходів.

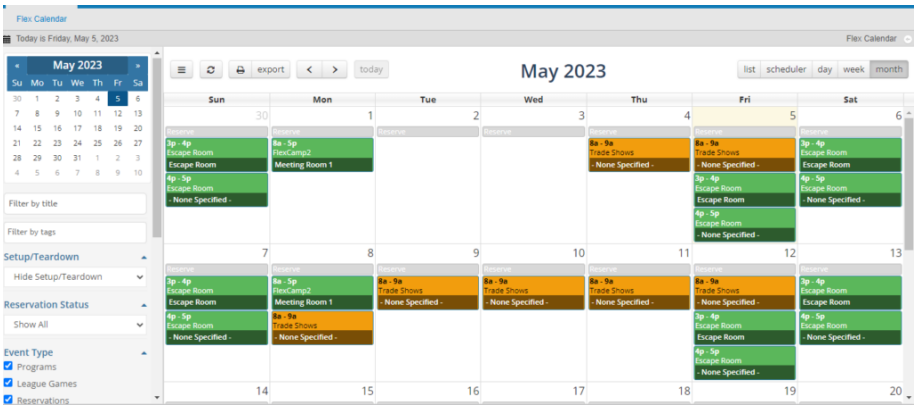

Рисунок 1.2 – Зовнішній вигляд інтерфейсу FlexSchedule

До недоліків FlexSchedule можна віднести те, що система не пристосована до стандартної моделі розкладу з парами, яка є звичайною для більшості університетів. Через те, що система призначена для блочної або модульної організації навчання традиційним закладам зазвичай необхідно внесення значних змін у внутрішніх процесах. Крім того, інтеграція з LMS-платформами іноді потребує технічної підтримки або використання додаткових модулів, що ускладнює ситуацію для закладів з обмеженими IT-ресурсами.

1.2.3 Untis

Untis – одна з найбільш популярних програм для створення розкладів у навчальних закладах, яка дозволяє швидко вводити інформацію про викладачів, предмети, групи, аудиторії та інші елементи навчального процесу, не потребуючи спеціальних навичок для користування.

До основних переваг можна віднести автоматичну перевірку наявності конфліктів у розкладі, наприклад, якщо у одного з викладачів заплановані

заняття у декількох місцях одночасно або якщо не вистачає вільних аудиторій. Також при роботі з Untis можуть братися до уваги побажання викладачів та різні обмеження, які встановлені адміністрацією.

Вся інформація зберігається в одній базі даних, що дозволяє швидко отримувати актуальні дані про використання ресурсів і приймати обґрунтовані управлінські рішення. У програмі Untis можна створити декілька варіантів розкладу, із яких можна вибрати найбільш зручний [5].

У разі потреби користувач може вручну вносити зміни в автоматично створений розклад – це особливо корисно в разі змін або непередбачуваних обставин. Розклад можна зберегти у форматах PDF, Excel або HTML, для того щоб була можливість зручної публікації в Інтернеті.

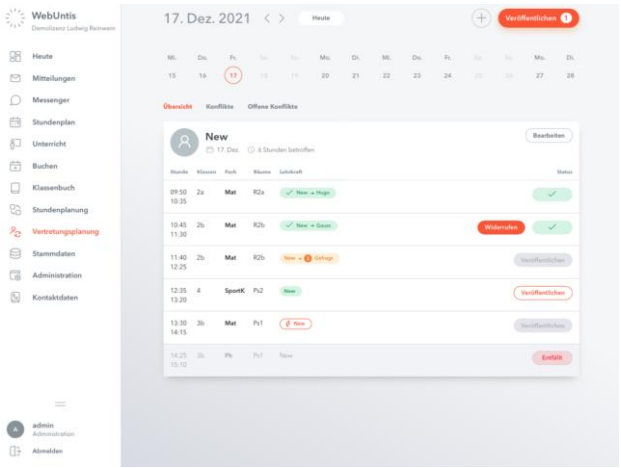

Рисунок 1.3 – Зовнішній вигляд інтерфейсу Untis

Одним з недоліків Untis є те, що для отримання повного доступу до всіх можливостей необхідно придбати ліцензію, що може стати значним фінансовим тягарем для деяких закладів. Крім цього, через велику кількість функцій і параметрів інтерфейс програми є досить складним, тому користувачам потрібен певний час на його оволодіння. Під час налаштування складних розкладів може знадобитися допомога технічних фахівців або звернення до служби підтримки.

1.2.4 FET (Free Timetabling Software)

FET – це безкоштовна платформа для складання розкладів. Ця програма є зручною для використання в школах та університетах, де потрібно враховувати багато змінних, починаючи від кількості аудиторій закінчуючи побажаннями викладачів. Користувач має можливість ввести всі потрібні дані: перелік предметів, груп, викладачів, обмеження за часом. Програма автоматично створює розклад, уникаючи конфліктів.

Усі дані зберігаються у форматі XML, що спрощує їх збереження, передачу та обробку. План включає декілька варіантів графіку, а саме користувач вирішує, який з них найбільш підходить. При потребі є можливість вносити зміни, що є корисним у випадку обмеженості часу.

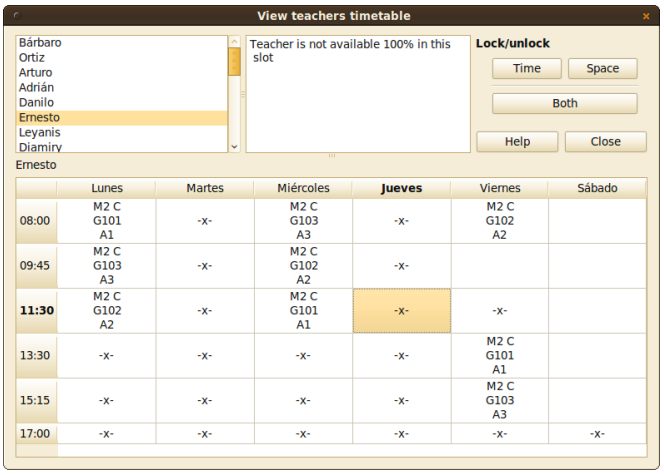

Рисунок 1.4 – Зовнішній вигляд інтерфейсу Free Timetabling Software

Серед недоліків FET слід зазначити складність налаштування великої кількості обмежень і пріоритетів, що потребує глибокого розуміння логіки роботи програми. Незважаючи на те, що система є безкоштовною, вона не оновлює розклад в реальному часі та не має вбудованих засобів для інтеграції з LMS або системами електронного обліку студентів, що робить її менш ефективною для великих навчальних закладів [6].

1.2.5 CIST Nure

У Харківському національному університеті радіоелектроніки функціонує система CIST, яка централізовано управляє навчальним розкладом і підтримує освітній процес у цифровому форматі. Ця онлайн-платформа дозволяє адміністрації університету планувати навчальне навантаження, координувати ресурси та надавати доступ до актуальної інформації всім учасникам навчального процесу.

Вся необхідна інформація зберігається в системі: перелік предметів, викладачів, груп, аудиторій та навчальні плани. Розклад автоматично створюється з урахуванням багатьох параметрів, таких як доступність викладачів до типу занять та пріоритетність приміщень [7].

Рисунок 1.5 – Зовнішній вигляд інтерфейсу CIST Nure

Інтерфейс CIST доступний з будь-якого пристрою або браузера, де є можливість переглянути актуальний розклад в будь-який час, оскільки його оновлення відбувається автоматично після внесення змін. До недоліків можна віднести те, що зміни в розкладі відбуваються один раз на день, через що, необхідно перевіряти можливі зміни кожного дня.

1.2.6 Результати аналізу існуючих рішень

Серед сучасних засобів автоматизації створення та управління розкладами в закладах вищої освіти виділяється кілька варіантів, кожен з яких має свої переваги та особливості впровадження в навчальний процес.

ASC Timetables надає зручний спосіб виявлення конфліктів у розкладі і підтримує мобільний застосунок EduPage, через який студенти та викладачі можуть легко отримувати своєчасну інформацію. Ця програма дозволяє створювати декілька варіантів розкладу та швидко вносити зміни вручну, що є корисним в умовах постійних змін.

FlexSchedule призначений для навчальних закладів з нестандартною організацією навчального процесу – блокові курси, змінні формати занять та інше. Програма чудово інтегрується з освітніми платформами типу Moodle або Canvas, що дозволяє синхронізувати розклад з навчальним планом.

Untis має потужну систему обробки замін та можливість оновлювати розклад у режимі реального часу. У програмі враховуються індивідуальні особливості вчителів і груп, що дозволяє створювати персоналізовані розклади. Завдяки WebUntis люди можуть отримати доступ до актуальної інформації у будь-який час.

FET – це безкоштовний застосунок з відкритим вихідним кодом. Ця програма має багато опцій налаштувань і може враховувати обмеження, але її використання вимагає більш детальних налаштувань, що може бути проблемою для користувачів без технічних знань.

CIST Nure – це внутрішня система університету ХНУРЕ, яка об'єднує створення розкладу з іншими освітніми модулями, такими як облік успішності, електронні журнали та інше. Система працює через веб-браузер і автоматично оновлює дані, при цьому розклад доступний для студентів, викладачів і адміністрації з урахуванням їхніх ролей. Однак ця система призначена для конкретного використання в ХНУРЕ і не може бути використана за його межами.

Загалом, всі ці системи мають спільну мету – автоматизувати створення розкладу з урахуванням різних параметрів. ASC та Untis – це комерційні продукти, які вимагають ліцензії. FET не завжди є зручним для початківців. А гнучкий графік не є оптимальним для класичних розкладів з парними предметами [8].

Таблиця 1.1 – Порівняльняння характеристик серед аналогів

Основні характеристики	Розглянути застосунки				
	Timetables	FlexSchedule	Untis	FET	CIST Nure
Автоматичне створення розкладу	+	+	+	+	+
Підтримка зміни/замін	+/-	+	+	-	+
Налаштування обмежень	+/-	+	+	+	+/-
Інтеграція з іншими системами	EduPage	LMS	WebUntis	-	Інтегрована
Експорт у різні формати	+	+	+	+	+
Платформа з відкритим кодом	-	-	-	+	-

1.3. Аналіз вимог до системи планування розкладу

Система планування розкладу потребує надійної бази даних для зберігання інформації. Структура бази даних має забезпечувати швидкий доступ до інформації. Механізми резервного копіювання захищають від

втрати даних. Передбачено функції відновлення інформації після збоїв завдяки архівації даних попередніх періодів навчання.

Програмний продукт забезпечує захист особистої інформації користувачів. Система розмежовує права доступу до різних функцій та інформації. Механізми авторизації контролюють дії користувачів у системі. Передбачено багаторівневу систему аутентифікації. Програма зберігає всі зміни, внесені користувачами та повідомляє адміністратора про підозрілі дії.

Користувацький інтерфейс відрізняється зручністю та інтуїтивною зрозумілістю. Система пропонує різні варіанти відображення розкладу занять. Користувачі можуть налаштовувати інтерфейс під свої потреби. Програма забезпечує швидку навігацію між різними розділами [9].

Функціонал застосунку включає механізми виявлення та вирішення конфліктів, завдяки автоматичній перевірці доступності аудиторій у вказаний час та використанню актуального розкладу викладачів.

Модуль аналітики надає інструменти для оцінки ефективності розкладу. Система збирає статистичні дані про використання аудиторного фонду. Програма аналізує розподіл навантаження викладачів. Користувачі отримують звіти про якість складеного розкладу.

Механізм внесення змін до розкладу забезпечує гнучкість планування. Система дозволяє оперативно реагувати на відсутність викладачів та дозволяє швидко вносити корективи до розкладу. Передбачено функції масового перенесення занять. Система зберігає історію всіх внесених змін, а механізм повідомлень інформує про зміни в розкладі.

Система сповіщень забезпечує оперативне інформування користувачів. Програма надсилає повідомлення про зміни в розкладі. Користувачі отримують нагадування про майбутні заняття. Передбачено різні канали комунікації з користувачами. Механізм повідомлень працює в режимі реального часу та автоматично зберігає історію всіх сповіщень.

Модуль роботи з викладачами забезпечує врахування їхніх побажань. Система дозволяє вказувати зручний час для проведення занять. Програма

враховує зайнятість викладачів у інших закладах. Користувачі можуть планувати час для методичної роботи. Передбачено функції обліку підвищення кваліфікації. Система контролює рівномірність навантаження викладачів. Механізм планування враховує наукову діяльність викладачів.

1.4. Результати аналізу

Вивчення процесу планування розкладу виявило низку ключових характеристик. Процес потребує координацію багатьох підрозділів навчального закладу. Створення розкладу вимагає врахування різноманітних факторів та обмежень. Ручне складання розкладу займає час та ресурси. Поточні методи роботи не забезпечують оптимального використання аудиторного фонду. Процес внесення змін до розкладу характеризується складністю та повільністю [10].

Дослідження наявних програмних рішень показало їхні переваги та недоліки. Більшість систем пропонують базові функції створення розкладу. Автоматизація планування реалізована на різних рівнях складності. Можливості адаптації розкладу обмежені закладеними алгоритмами.

Аналіз вимог користувачів визначив основні напрямки розробки. Система має забезпечувати гнучке планування навчального процесу, простоту у використанні наявних інструментів для роботи з розкладом. Функціонал повинен підтримувати різні сценарії використання. Механізми автоматизації мають враховувати специфіку закладу.

Технічні вимоги до системи охоплюють різні рівні реалізації. База даних має забезпечувати надійне зберігання інформації. Система повинна підтримувати одночасну роботу великої кількості користувачів. Програмний код потребує модульної структури для масштабування.

Дослідження користувацького досвіду виявило очікування від системи. Користувачі потребують простого та зрозумілого інтерфейсу. Функціонал має відповідати реальним сценаріям використання. Система повинна

надавати швидкий доступ до потрібної інформації. Механізми пошуку мають працювати ефективно та точно. Сповіщення про зміни повинні бути своєчасними. Налаштування інтерфейсу має враховувати переваги користувачів. Документація системи повинна бути повною та зрозумілою.

Аналіз потреб викладачів показав необхідні інструменти. Планування навантаження має бути зручним та гнучким. Система повинна враховувати побажання щодо розкладу. Заміни викладачів мають плануватися за мінімальний час. Методична робота потребує окремого планування. Наукова діяльність має враховуватися при складанні розкладу [11].

1.5. Постановка задачі розробки

1.5.1 Системні вимоги

Технічні характеристики серверного обладнання визначають базові вимоги для розгортання системи. Комп’ютер потребує процесор з тактовою частотою не менше 3.0 GHz. Кількість ядер процесора має становити мінімум 8 одиниць. Оперативна пам'ять повинна мати обсяг від 32 GB. Дисковий простір розраховується з урахуванням обсягу бази даних та з урахуванням необхідності збереження інформації про навчальні програми.

Програмне забезпечення включає такі компоненти. Операційна система Windows Server 2019 або новіша версія. Система керування базами даних Microsoft SQL Server 2019. Веб-сервер IIS версії 10 для розгортання застосунку. Фреймворк .NET 6.0 або новіший для виконання програми. Системи резервного копіювання для захисту даних. Антивірусне програмне забезпечення для безпеки сервера. Засоби моніторингу для контролю роботи системи та оперативного виявлення помилок.

Програмне оточення користувача включає стандартні компоненти. Операційна система Windows 10 або новіша версія. Браузер можна викоритовувати в залежності від уподобань коирстувача. Офісний пакет

Microsoft Office для документів . Програма для перегляду PDF-файлів. Антивірусне програмне забезпечення для захисту даних. Додаткове програмне забезпечення за потребами користувача.

Системи зберігання даних забезпечують надійність інформації. Сховище даних має підтримувати технологію RAID. Дискова підсистема повинна забезпечувати швидкий доступ. Система резервного копіювання працює за встановленим розкладом. Архівне зберігання даних організовано на окремих носіях. Механізми відновлення даних мають бути протестовані.

Системи безпеки реалізують комплексний захист інформації. Міжмережевий екран контролює мережевий трафік. Антивірусний захист працює на всіх компонентах системи. Система виявлення вторгнень відстежує підозрілу активність. Шифрування даних при передачі через мережу. Механізми автентифікації перевіряють користувачів. Контроль доступу реалізований на рівні застосунку. Аудит безпеки проводиться автоматично за окремим розкладом.

Системи моніторингу забезпечують контроль роботи. Збір метрик продуктивності всіх компонентів системи. Відстеження стану мережевих з'єднань та сервісів. Контроль використання системних ресурсів серверів. Моніторинг доступності всіх функцій системи. Аналіз журналів роботи для виявлення проблем. Сповіщення про критичні події та збої. Збереження історії моніторингу для аналізу.

1.5.2 Опис функцій програмного забезпечення

Модуль автентифікації забезпечує захищений доступ до системи. Користувачі проходять процедуру входу за допомогою логіну та паролю. Система підтримує багатофакторну автентифікацію через SMS або email. Механізм відновлення паролю реалізований через підтверджену електронну пошту. Зберігання паролів відбувається в захищеному вигляді з використанням хешування.

Функціонал управління користувачами надає інструменти адміністрування. Створення нових облікових записів з визначенням прав доступу. Редагування профілів користувачів та їх налаштувань системи. Блокування облікових записів неактивних користувачів. Система розподіляє користувачів за групами та ролями. Механізм успадкування прав спрощує адміністрування доступу. Пошук користувачів реалізований за різними параметрами. Експорт списку користувачів з детальною інформацією [12].

Функціонал контролю відвідуваності забезпечує моніторинг занять. Облік присутності студентів на заняттях різних типів. Формування журналів відвідуваності по групах. Аналіз статистики відвідування за період навчання. Система генерує звіти про пропуски занять як для студентів так і для викладачів [13]. Експорт даних для деканатів та кураторів. Інтеграція з електронним журналом університету.

Модуль обліку успішності відстежує результати навчання. Ведення електронного журналу оцінок з усіх дисциплін. Розрахунок середніх балів та рейтингів студентів. Формування відомостей успішності за контрольні заходи. Система аналізує динаміку успішності студентів.

Модуль статистики та аналітики обробляє дані системи. Збір та аналіз показників використання аудиторного фонду. Розрахунок статистики завантаженості викладачів. Аналіз ефективності складеного розкладу занять. Система виявляє проблемні місця в організації навчання. Генерація аналітичних звітів для керівництва. Експорт даних для подальшої обробки. Візуалізація статистичної інформації.

Модуль адміністрування надає інструменти управління. Налаштування параметрів роботи всіх модулів системи. Управління правами доступу користувачів до функцій. Моніторинг продуктивності та навантаження на систему. Резервне копіювання та відновлення даних. Оновлення програмного забезпечення системи. Ведення журналів роботи та аудиту. Діагностика та усунення проблем [14].

2 ВИБІР ТА ОБҐРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір платформи розробки та мови програмування

2.1.1. Вибір програмних засобів розробки системи

Платформа Microsoft .NET Framework надає потужні інструменти для розробки. Середовище виконання CLR забезпечує керування пам'яттю та безпекою програми. Фреймворк містить велику кількість готових бібліотек для розробки. Система типів .NET підтримує ООП.

Мова програмування C# об'єднує в собі ефективність та стабільність. Завдяки статичній типізації можна виявити помилки ще на етапі збірки. Її синтаксис зрозумілий та послідовний для розробників. Підтримка сучасних парадигм програмування робить код гнучким. Регулярні оновлення додають нові можливості до мови. Велика кількість навчальних матеріалів полегшує освоєння. Розвинена екосистема бібліотек прискорює розробку.

Середовище розробки Visual Studio забезпечує комфортну роботу. Інтегрований контроль якості коду дозволяє ефективно знаходити помилки. Система контролю версій інтегрована в середовище розробки. Візуальний редактор форм спрощує створення інтерфейсу. Засоби стандартизації допомагають покращувати якість коду [15].

Технологія Windows Presentation Foundation (WPF) оптимізує розробку інтерфейсу. Декларативний підхід XAML спрощує створення компонентів. Система компонування забезпечує гнучке розміщення елементів. Механізм прив'язки даних зменшує обсяг коду необхідний для безперебійного функціонування застосунку.

Для обґрунтованого вибору платформи розробки було проведено порівняльний аналіз найпопулярніших технологій, результати якого наведені в таблиці 2.1.

Таблиця 2.1 – Порівняння платформ розробки для реалізації системи

Характеристика	.NET Framework	Java	Python	PHP
Продуктивність	Висока	Висока	Середня	Середня
Масштабованість	Висока	Висока	Середня	Середня
Кросплатформність	Висока	Висока	Висока	Висока
Безпека	Висока	Висока	Середня	Середня
Підтримка GUI	WPF, WinForms	Swing, JavaFX	PyQt, Tkinter	-
Підтримка БД	ADO.NET, Entity Framework	JDBC, Hibernate	SQLAlchemy	PDO
Швидкість розробки	Висока	Середня	Висока	Висока
Вартість розробки	Висока	Середня	Низька	Низька
Зрілість платформи	Висока	Висока	Середня	Середня
Екосистема	Розвинена	Розвинена	Розвинена	Розвинена
Наявність спеціалістів	Висока	Висока	Середня	Висока
Документація	Повна	Повна	Хороша	Хороша

2.1.2. Платформи розгортання та засоби життєвого циклу програмного забезпечення

Система для роботи з SQL надає можливості з надійного зберігання даних. Транзакційний механізм забезпечує цілісність даних. Індексування підвищує швидкість пошуку інформації. Система безпеки необхідна для контролю та забезпечення захисту інформації користувачі застосунку. Інструменти адміністрування спрощують обслуговування бази даних. Механізми реплікації підвищують надійність системи.

Фреймворк Entity Framework спрощує роботу з базою даних. Об'єктно-

реляційне відображення автоматизує доступ до даних. Система міграцій контролює зміни структури бази. LINQ забезпечує безпечні запити до даних. Кешування підвищує продуктивність доступу до інформації. Механізм відстеження змін спрощує оновлення даних. Підтримка асинхронних операцій покращує швидкодію.

Система контролю версій Git дає змогу ефективно управляти кодом. Завдяки розподіленій архітектурі підвищується надійність зберігання, а механізм гілок забезпечує можливість паралельної розробки функціоналу. Система тегів спрощує випуск продуктів.

Середовище IIS оптимізує роботу веб-застосунків. Список застосунків ізолює процеси для надійності. Система модулів розширює можливості сервера. Механізм кешування підвищує продуктивність. Налаштування безпеки захищають від атак [16]. Журналювання допомагає знаходити проблеми. Кластеризація забезпечує високу доступність.

Бібліотека NuGet спрощує управління залежностями проєкту. Централізоване сховище пакетів прискорює розробку. Система версій контролює сумісність компонентів. Автоматичне оновлення підтримує актуальність бібліотек.

Фреймворк для тестування MSTest забезпечує якість коду. Атрибути тестів спрощують написання перевірок. Параметризовані тести розширюють покриття коду. Категорії тестів організують набори перевірок. Звіти про результати допомагають знаходити помилки. Інтеграція з середовищем розробки автоматизує тестування.

Система MSBuild автоматизує процес збірки проєкту. Файли програми описують процес збірки декларативно. Умовна сбірка MSBuild адаптує проєкт під різні платформи. Система цілей організує необхідні етапи. Користувацькі завдання розширюють можливості збірки. Паралельна компіляція прискорює процес.

Система журналювання NLog забезпечує діагностику застосунку. Конфігурація через XML спрощує налаштування. Різні приймачі звітів

розширюють можливості запису. Фільтрація подій зменшує обсяг журналів. Форматування повідомлень покращує розуміння коду. Асинхронний запис підвищує продуктивність. Ротація файлів контролює розмір журналів.

Фреймворк ASP.NET Core розширює можливості веб-розробки. Модульна архітектура забезпечує гнучкість системи. Middleware спрощує обробку HTTP-запитів. Конфігурація через код підвищує контроль над застосунком. Dependency Injection організує зв'язки компонентів. Кросплатформність розширює середовище розгортання. Висока продуктивність оптимізує роботу.

Система автоматизації Azure DevOps керує життєвим циклом. Репозиторії коду централізують зберігання проєкту. Конвеєри збірки автоматизують компіляцію та тестування. Система версій контролює розгортання застосунку. Дошки завдань організують процес розробки [17]. Тестові плани систематизують перевірку якості.

Платформа SignalR забезпечує двонаправлену комунікацію. WebSocket оптимізує обмін даними в реальному часі. Автоматичне відновлення з'єднання підвищує надійність. Масштабування через Redis забезпечує високе навантаження. Групи з'єднань організують розсилку повідомлень. Клієнтські бібліотеки спрощують інтеграцію. Моніторинг допомагає відстежувати проблеми.

2.2. Обґрунтування вибору технологій розробки

Вибір платформи .NET Framework зумовлений потребами розлобленного проєкту. Платформа надає комплексний набір інструментів та можливостей для розробки різноманітних програмних компонентів. Середовище виконання забезпечує надійну та безперебійну роботу програми. Вбудовані механізми безпеки автоматично захищають дані користувачів від потеційних атак. Розвинена екосистема розробників прискорює процеси створення та інтеграції програмних продуктів.

Таблиця 2.2 – Характеристики обраних технологій розробки

Технологія	Призначення	Ключові переваги	Особливості використання
C#	Мова програмування	Статична типізація, ООП, надійність коду	Основна мова розробки
WPF	UI фреймворк	XAML, векторна графіка, стилізація	Створення інтерфейсу
SQL Server	СУБД	Транзакції, масштабованість, надійність	Зберігання даних
Entity Framework	ORM	Автоматичне відображення, міграції, LINQ	Доступ до даних
Git	Система контролю версій	Розподіленість, гілки, історія змін	Керування кодом
Visual Studio	IDE	Відладка, рефакторинг, інтеграції	Середовище розробки
MSTest	Фреймворк тестування	Атрибути, моки, параметризація	Тестування коду
SignalR	Бібліотека комунікації	WebSocket, масштабування, надійність	Обмін даними
NLog	Система логування	Конфігурація, фільтрація, ротація	Журналювання подій

Для реалізації системи було обрано комплекс технологій, основні характеристики яких наведені в таблиці 2.2. Кожна з обраних технологій відіграє свою роль у забезпеченні надійності та ефективності розробленого програмного рішення.

Мова C# обрана через технічні характеристики та переваги. Статична типізація забезпечує надійність програмного коду. Об'єктно-орієнтований підхід структурує архітектуру застосунку. Розвинений синтаксис спрощує написання програм. Регулярні оновлення додають сучасні можливості і автоматичну оптимізацію продуктивності програми. Інтеграція з платформою .NET розширює функціональність.

Технологія WPF обрана для створення користувацького інтерфейсу. Векторна графіка забезпечує якісне відображення елементів. Система стилів дозволяє стандартизувати дизайн програми. Механізм прив'язки даних зменшує обсяг коду. Підтримка анімацій покращує користувацький досвід. Компонування елементів автоматично адаптується до розмірів вікна користувача, з можливістю змін.

SQL Server обрано як систему керування базами даних завдяки її транзакційному механізму, що гарантує цілісність інформації. Продуктивність сервера забезпечує швидку роботу з даними. Інструменти адміністрування спрощують обслуговування. Механізми дублювання підвищують надійність системи. Вбудована аналітика оптимізує запити до бази. Масштабованість дозволяє розширювати систему [18].

Entity Framework спрощує взаємодію з базою даних. Об'єктно-реляційне відображення автоматизує роботу з даними. Система міграцій контролює зміни структури бази. Кешування підвищує швидкість доступу до даних. Відстеження змін автоматизує оновлення інформації. Асинхронні операції покращують відгук системи.

Git обраний для керування версіями програмного коду. Розподілена архітектура забезпечує надійність зберігання. Система гілок підтримує паралельну розробку функцій та дозволяє зберігати історію змін проєкту. Можливості об’єднання коду автоматизують процес інтеграції. Система тегів спрощує управління версіями. Графічні інструменти контролю гілок полегшують роботу з репозиторієм, надаючи можливість редагування без використання консольних команд [19].

2.3. Архітектурні патерни та підходи до розробки

Патерн Model-View-ViewModel формує основу архітектури застосунку. Розділення логіки та представлення покращує структуру програми. Модель даних залишається незалежною від інтерфейсу користувача. ViewModel забезпечує зв'язок між Model та View.

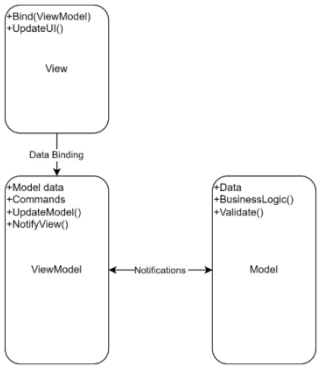

Рисунок 2.1 – Структура та взаємодія компонентів патерну MVVM

Механізм прив'язки даних автоматизує оновлення інтерфейсу. Команди інкапсулюють дії користувача в окремі об'єкти. Тестування компонентів стає простішим завдяки розділенню відповідальності.

Підхід Domain-Driven Design структурує бізнес-логіку системи. Предметна область описується через чітко визначені сутності та об'єкти. Агрегати відповідають за цілісність взаємопов’язаних даних, сховища абстрагують взаємодію з базою даних, а сервіси реалізують логіку складних бізнес-процесів. Фабрики створюють об'єкти предметної області. Події доменну передають інформацію між компонентами.

Патерн Repository ізолює роботу з базою даних. Абстрактний інтерфейс приховує деталі зберігання даних. Репозиторії забезпечують доступ до колекцій об'єктів. Реалізація може змінюватись без впливу на бізнес-логіку. Кешування результатів запитів підвищує продуктивність. Транзакції забезпечують цілісність даних.

Принцип Dependency Injection забезпечує слабке зв'язування компонентів. Залежності передаються об'єктам через конструктор або властивості. Контейнер DI керує життєвим циклом об'єктів. Сервіси реєструються в контейнері з визначеним часом життя. Інтерфейси описують контракти взаємодії компонентів. Тестування спрощується завдяки можливості підміни реалізацій [20].

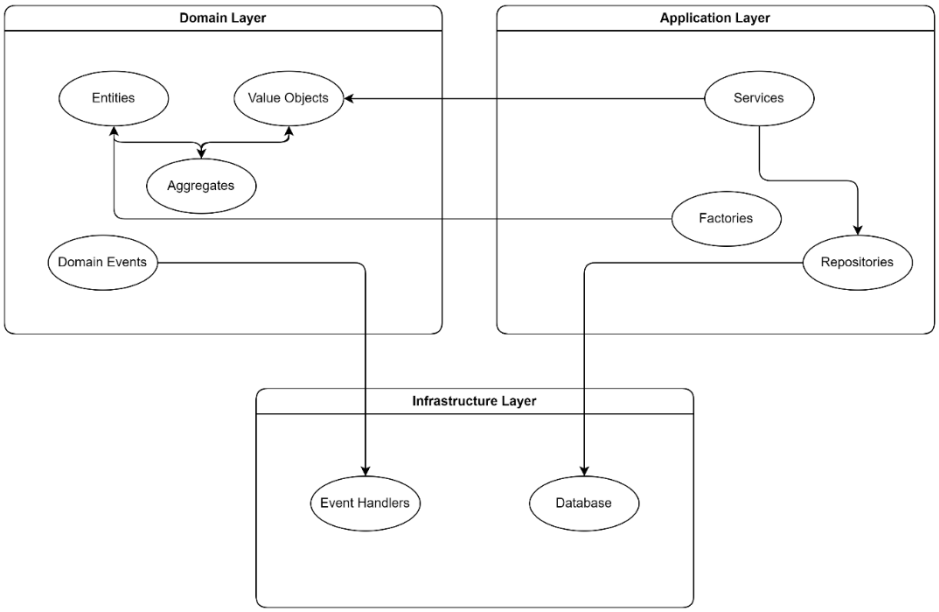

Рисунок 2.2 – Архітектурні шари та компоненти в Domain-Driven Design

Патерн Unit of Work координує операції з даними, забезпечуючи їх узгоджене збереження. Транзакції збирають операції оновлення бази даних в один запит. Відстеження змін об'єктів відбувається автоматично. Перевірка даних виконується автоматично перед збереженням. Обробка помилок забезпечує цілісність даних. Можливість повернутись до попередньої версії при виникненні проблем.

Підхід Command Query Responsibility Segregation розмежовує операції читання та запису: команди змінюють стан системи без повернення результату, а запити лише отримують дані, не впливаючи на стан. Асинхронна обробка команд сприяє підвищенню масштабованості системи.

Події інформують про зміни в системі. Журналювання команд спрощує проведення аудиту системи.

Патерн Observer забезпечує оновлення залежних компонентів. Підписники отримують сповіщення про зміни в об'єктах. Слабке зв'язування досягається через інтерфейси. Видавці не залежать від конкретних підписників. Асинхронна доставка подій покращує продуктивність.

Підхід Event Sourcing зберігає історію змін системи. Події описують всі зміни стану об'єктів. Відновлення стану відбувається через програвання подій. Знімки стану прискорюють відновлення об'єктів. Події забезпечують аудит усіх змін у системі [21]. Паралельна обробка подій підвищує продуктивність.

Патерн Mediator централізує взаємодію компонентів. Медіатор інкапсулює протоколи обміну даними. Компоненти не залежать один від одного. Маршрутизація повідомлень відбувається через медіатор. Обробка помилок централізована в одному місці. Розширення функціональності через додавання обробників.

Підхід Specification інкапсулює правила вибірки даних. Специфікації комбінуються через логічні операції. Повторне використання критеріїв пошуку спрощується. Читабельність коду запитів покращується.

Патерн Strategy визначає набір взаємозамінних алгоритмів, кожен із яких інкапсулює окрему реалізацію функціональності, що забезпечує гнучкість у виборі поведінки під час виконання. Вибір алгоритму можливий під час виконання програми. Додавання нових стратегій не змінює існуючий код. Конфігурація системи стає більш гнучкою.

Підхід Chain of Responsibility формує ланцюг обробників, у якому кожен елемент самостійно вирішує, чи обробляти запит, чи передати його далі по ланцюгу. Запити передаються по ланцюжку до відповідного обробника. Додавання нових обробників не змінює існуючий код. Порядок обробки може змінюватись динамічно. Розділення відповідальності між обробниками покращує структуру [22].

Патерн Factory Method інкапсулює створення об'єктів. Фабричні методи приховують деталі створення екземплярів. Підкласи можуть змінювати тип створюваних об'єктів. Повторне використання коду ініціалізації об'єктів.

Підхід Template Method визначає структуру алгоритму. Базовий клас задає послідовність кроків операції. Підкласи реалізують конкретні кроки алгоритму. Повторне використання коду покращується. Розширення функціональності через створення нових підкласів.

Патерн State керує поведінкою об'єкта при зміні стану. Стани інкапсулюють поведінку об'єкта в різних умовах. Перехід між станами відбувається автоматично. Додавання нових станів не змінює існуючий код. Логіка поведінки локалізована в класах станів [23]. Тестування кожного стану проводиться окремо. Моніторинг переходів спрощує діагностику.

2.4. Технології реалізації застосунку

2.4.1 Середовище розробки Visual Studio Code

Visual Studio має велику кількість різноманітних інструментів розробки, різноманітні шаблони для створення проєктів, що дає змогу швидко створювати застосунки на основі технології WPF для розробки графічного інтерфейсу. Середовище також пропонує розширені засоби роботи з XAML, спрощуючи створення вікон і керуючих елементів [24].

Інтеграція з системами керування базами даних, зокрема Microsoft SQL Server, дає змогу безпосередньо за допомогою вбудованих інструментів редактора створювати, переглядати та змінювати бази даних. У Visual Studio також є засоби для налагодження: встановлення точок зупинки у коді, перегляд стану змінних, аналіз списку викликів методів та відстження різноманітних об'єктів у пам'яті, що значно спрощує пошук та виправлення потеційних помилок [25].

2.4.2 Мова програмування C#

C# являє собою сучасну об’єктно-орієнтовану мову програмування, розроблену для платформи .NET, яка поєднує зрозумілий синтаксис, жорстку типізацію та підтримку кількох парадигм, що забезпечило її популярність у створенні різноманітних застосунків.

Однією з основних переваг C# є повна підтримка об'єктно-орієнтованого підходу, що дозволяє створювати масштабовані та легкі в обслуговуванні рішення. Мова втілює основні принципи ООП, а саме успадкування, інкапсуляцію та поліморфізм, які сприяють структуруванню коду та полегшують його повторне використання.

У C# також є підтримка елементів функціонального програмування, таких як лямбда-вирази, анонімні методи та делегати, що дозволяє створювати більш гнучкі й ефективні рішення.

Механізм обробки виключень є важливою складовою для ефективного контролю за помилками під час виконання програми. Крім того, за допомогою підтримки асинхронного програмування та ключових слів async та await, мова програмування дає змогу створювати продуктивні програми з паралельною обробкою завдань.

У розробці системи планування розкладу C# використовується для реалізації бізнес-логіки, роботи з базою даних через Entity Framework, а також для побудови графічного інтерфейсу за допомогою WPF.

2.4.3 Технологія створення інтерфейсу

WPF яляє собою сучасну технологію, яка використовуєтсья для розробки графічних інтерфейсів зостосунків у середовищі Windows.

Однією з ключових переваг є використання можливостей для незалежного визначеня структури інтерфейсу, відокремлюючи його від програмної логіки.

WPF працює за моделлю програмування на основі подій, підтримує двостороннє зв'язування даних (data binding) та надає можливість використовувати шаблони і стилі для стандартизації вигляду елементів [26].

2.4.4 ORM-технологія Entity Framework

Entity Framework (EF) – це популярна технологія ORM (Object-Relational Mapping) від Microsoft, яка дозволяє працювати з базами даних через об'єктно-орієнтовані моделі [27].

Основна перевага Entity Framework полягає в автоматизації операцій зчитування, додавання, оновлення та видалення даних за допомогою об’єктів і класів. Технологія підтримує кілька підходів до роботи: Database First (створення моделей з наявної бази даних), Code First (створення бази даних на основі коду) і Model First (створення моделі в графічному редакторі з подальшим створенням бази даних). Для розробки системи планування навчального розкладу використовується Entity Framework для роботи з даними про студентів, викладачів, предмети та розклади.

2.4.5 Збереження та керування даними

Одною з основних переваг SQL Server є висока продуктивність, підтримка транзакцій, забезпечення цілісності даних і розвинуті можливості адміністрування.

SQL Server має тісну інтеграцію з Visual Studio, що дозволяє працювати з базами даних безпосередньо у середовищі розробки.

У Microsoft SQL Server використовується для зберігання інформації про студентів, викладачів, навчальні предмети, аудиторії та розклади в системі планування. Надійність та масштабовість системи управління базами даних (СКБД) гарантують стійку роботу системи та забезпечують здатність обробляти велику кількість одночасних запитів [28].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1. Проєктування бази даних

Структура бази даних розроблена з урахуванням вимог системи. Таблиці організовані за принципом нормалізації даних. Зв'язки між таблицями забезпечують цілісність інформації. Індекси оптимізують швидкість пошуку та вибірки даних. Обмеження контролюють коректність даних. Представлення спрощують доступ до пов'язаної інформації. Збережені процедури реалізують складну логіку обробки.

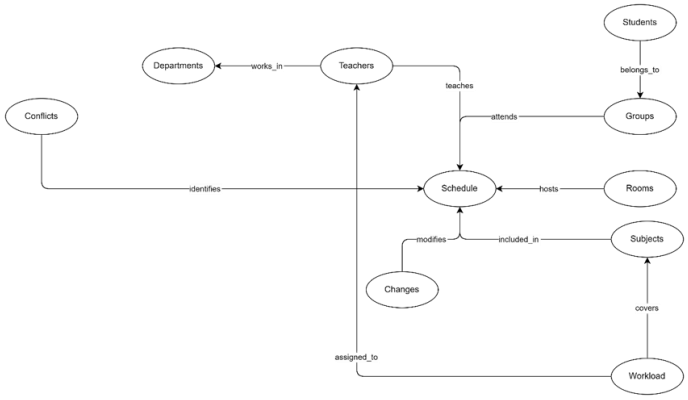

Рисунок 3.1 – ER-діаграма основних таблиць бази даних

Таблиця «Teachers» зберігає інформацію про викладачів. Поля містять особисті дані та контактну інформацію. Записи включають дані про наукові ступені та звання. Зберігається інформація про спеціалізацію викладачів. Поле доступності вказує можливий час проведення занять. Зовнішні ключі пов'язують з кафедрами та дисциплінами.

Таблиця «Students» містить дані про навчальний розклад студентів. Записи включають особисту інформацію та контакти. Поля групи та спеціальності визначають спеціалізацію студента. Зберігається інформація

про форму навчання та рік вступу. Зовнішні ключі пов'язують з академічними групами. Є можливість контролю переведення між групами.

Таблиця «Subjects» зберігає перелік навчальних дисциплін. Поля описують назву та тип дисципліни. Записи містять кількість годин різних видів занять. Зберігається інформація про семестр викладання. Зовнішні ключі пов'язують з кафедрами та викладачами. Індекси прискорюють пошук за назвою [29].

Таблиця «Rooms» містить дані про навчальні приміщення. Записи включають номер та тип аудиторії. Поля описують місткість та технічне оснащення. Зберігається інформація про розташування приміщень. Зовнішні ключі пов'язують з корпусами закладу.

Таблиця «Schedule» реалізує основний розклад занять. Записи зв'язують викладачів, групи та дисципліни. Поля визначають час та місце проведення занять. Зберігається тип заняття та періодичність проведення. Зовнішні ключі забезпечують цілісність даних.

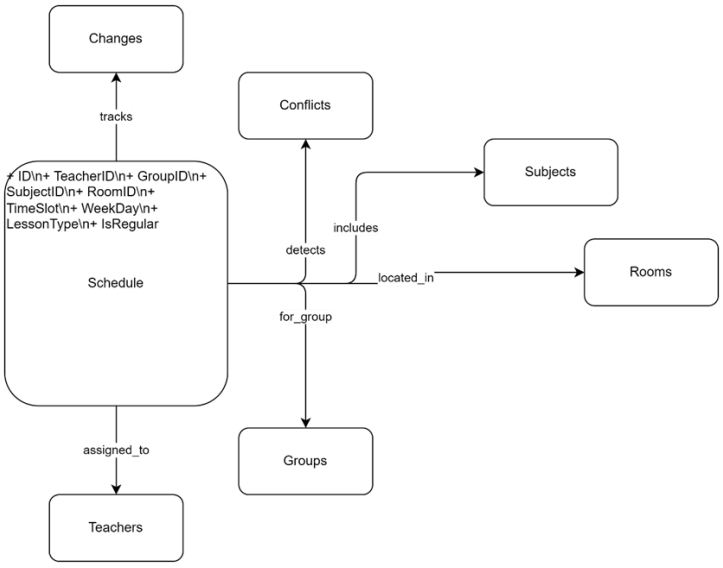

Рисунок 3.2 – Структура та зв'язки таблиці Schedule

Таблиця «Groups» зберігає інформацію про навчальні групи. Поля містять назву та спеціальність групи. Записи включають курс та форму

навчання. Зберігається кількість студентів у групі. Зовнішні ключі пов'язують з факультетами. Тригери контролюють зміни складу груп. Індекси оптимізують вибірку за спеціальністю.

Таблиця «Departments» містить дані про кафедри закладу. Записи включають назву та код кафедри. Поля описують спеціалізацію та керівництво. Зберігається контактна інформація підрозділу. Зовнішні ключі пов'язують з факультетами. Обмеження контролюють унікальність кодів. Індекси прискорюють пошук за назвою.

Таблиця «Changes» відстежує зміни в розкладі занять. Записи фіксують дату та тип змін. Поля містять причину та ініціатора змін. Зберігається статус затвердження змін. Зовнішні ключі пов'язують з основним розкладом. Є можливість модифікації розкладу.

Таблиця «Workload» зберігає дані про навантаження викладачів. Поля містять кількість годин різних видів занять. Записи включають семестр та навчальний рік. Зберігається інформація про виконання навантаження. Зовнішні ключі пов'язують з викладачами та дисциплінами. Тригери оновлюють дані при змінах розкладу.

Таблиця «Equipment» містить перелік технічного оснащення. Записи описують тип та характеристики обладнання. Поля включають дату встановлення та стан. Зберігається інформація про обслуговування. Зовнішні ключі пов'язують з аудиторіями. Тригери відстежують переміщення обладнання. Індекси оптимізують пошук за типом.

Таблиця «Conflicts» фіксує виявлені помилки в розкладі. Поля включають дату виявлення та статус. Зберігається спосіб вирішення конфлікту. Зовнішні ключі пов'язують з заняттями розкладу. Є автоматичне створення записів при перевірці розкладу. Індекси прискорюють аналіз та вирішення конфліктних ситуацій.

Таблиця «Settings» зберігає налаштування системи. Записи містять параметри та їх значення. Поля включають опис та тип налаштувань. Зберігається інформація про останні зміни.

Таблиця «Logs» накопичує журнал подій системи. Записи фіксують час та тип події. Поля містять детальний опис операції. Зберігається користувач та результат дії. Зовнішні ключі пов'язують з об'єктами системи. Тригери очищують старі записи журналу.

Таблиця «Notifications» керує розсилкою повідомлень. Записи містять текст та тип сповіщення. Поля включають час створення та відправки. Зберігається статус доставки повідомлення.

3.2. Архітектура програми

Система реалізована на основі трьохрівневої архітектурної моделі. Рівень представлення (Presentation Layer) забезпечує взаємодію з користувачем і створений за допомогою WPF. Інтерфейс включає головне вікно, модальні форми для створення, редагування та видалення записів, а також екрани авторизації й аналітики.

Для виконання SQL-запитів використовуються об'єкти SqlConnection та SqlCommand, що забезпечує надійну та контрольовану роботу з даними.

Рівень бізнес-логіки (Business Logic Layer) реалізує основну функціональність застосунку, включаючи обробку подій, перевірку введених користувачем даних та виконання фонових завдань, зокрема роботу сервісу сповіщень.

3.2.1 Основні діалогові вікна та аналітика

У даній частині проєкту було створено основні діалогові вікна для додавання нових елементів, таких як групи (AddGroupDialog.xaml.cs), аудиторії (AddRoomDialog.xaml.cs), розклади (AddScheduleDialog.xaml.cs), предмети (AddSubjectDialog.xaml.cs) і користувачі (AddUserDialog.xaml.cs).

У кожному з цих вікон було розроблено автоматичний алгоритм перевірки введеної інформації та взаємодії з базою даних, щоб забезпечити

коректну роботу системи та уникнути можливих помилок при невірному вводі логіну та паролю.

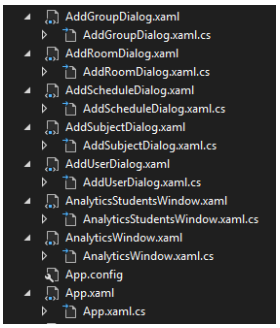

Рисунок 3.3 – Файлова структура діалоговий вікн та аналітики

AnalyticsWindow.xaml.cs і AnalyticsStudentsWindow.xaml.cs – вікна для аналітики . Вони були використані для збору статистики про студентів.

3.2.2 Редагування та звіти

У цій групі містяться файли, що відповідають за редагування раніше створених об'єктів.

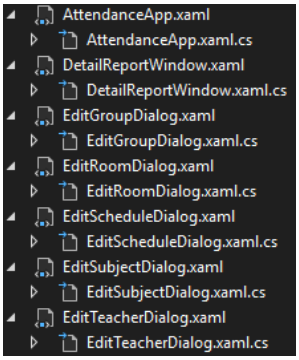

Рисунок 3.4 – Файлова структура редагування та звітів

Файли EditGroupDialog.xaml.cs, EditRoomDialog.xaml.cs, EditScheduleDialog.xaml.cs, EditSubjectDialog.xaml.cs та EditTeacherDialog.xaml.cs – кожен з

них реалізує логіку для завантаження існуючих даних, редагування інформації та збереження змін.

AttendanceApp.xaml.cs відповідає за основну логіку реєстрації відвідуваності.

3.2.3 Інтерфейс та сервіси

Ця група містить файли, що пов'язані з основним інтерфейсом, авторизацією та сервісами повідомлень.

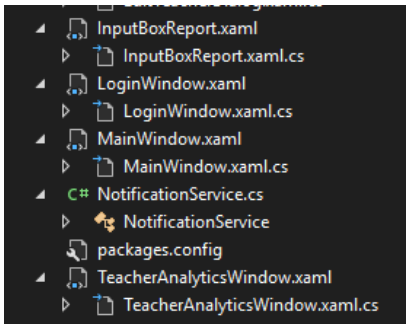

Рисунок 3.5 – Файлова структура сервісів та інтерфейсу

MainWindow.xaml.cs – представляє головне вікно програми, яке використовується для переміщення між основними функціями, такими як аналітика, розклад, редагування і т.д.

LoginWindow.xaml.cs – це вікно, призначене для аутентифікації користувача. У цьому файлі реалізується перевірка введених облікових даних, встановлюється з’єднання з базою даних і виконується відкриття головного вікна (MainWindow) у разі успішного входу.

NotificationService.cs – сервіс відображення повідомлень користувачу.

InputBoxReport.xaml.cs – введення параметрів для формування звітів.

TeacherAnalyticsWindow.xaml.cs – це вікно аналітики, яке спрямоване на викладачів. У цьому документі здійснено збір і відображення статистики щодо навантаження викладачів, їх занять і розкладу.

3.3. Реалізація інтерактивного інтерфейсу

Інтерактивний інтерфейс користувача в системі UniversitySQL_System реалізовано за допомогою технології WPF. Інтерфейс використовується для вікон авторизації, головного вікна управління системою, CRUD-операціями та інше [30].

3.3.1 Авторизація

Після запуску програми користувач перенаправляється до вікна LoginWindow, у якому здійснюється перевірка логіна та пароля шляхом звернення до бази даних. У разі успішного входу автоматично відкривається MainWindow (лістинг 3.1) .

Лістинг 3.1 – Початок роботи основного меню

```
      public MainWindow(){
          InitializeComponent();
          LoadData();
          _notificationService.StartNotifications(_currentUserId,
_currentUserRole);}
```

3.3.2 Основне вікно

MainWindow містить елементи з таблицями (DataGrid) для користувачів, груп, предметів та розкладу. Кожен елемент списку має кнопки «Додати», «Редагувати», «Видалити» (лістинг 3.2).

Лістинг 3.2 – Функція AddRoom_Click

```
          private void AddRoom_Click(object sender, RoutedEventArgs e){
              if (App.CurrentUserRole != "Admin"){
                  MessageBox.Show("Ви не маєте дозволу на додавання кімнат.");
                  return;}
              var dialog = new AddRoomDialog();
              if (dialog.ShowDialog() == true)
```

```
            {
                string query = "INSERT INTO Rooms (Name, Capacity, Type)
VALUES (@Name, @Capacity, @Type)";
                SqlParameter[] parameters =

            {
             new SqlParameter("@Name", dialog.RoomName),
             new SqlParameter("@Capacity", dialog.Capacity),
             new SqlParameter("@Type", dialog.RoomType)
        };
                ExecuteNonQuery(query, parameters);

                LoadData(); // Refresh the rooms list
            }
        }
```

3.3.3 Аналітичні панелі

Вікна AnalyticsStudentsWindow і TeacherAnalyticsWindow дозволяють користувачу обрати параметри фільтрації та переглянути графіки відвідуваності або завантаженості. Побудова графіків виконується за допомогою бібліотеки LiveCharts (лістинг 3.3).

Лістинг 3.3 – Функція для малювання графіку відвідуваності

```
    private void DrawChart(List<AttendanceData> attendance){
        var values = new ChartValues<int>(attendance.Select(x =>
x.PresentCount));

        SeriesCollection = new SeriesCollection{
            new ColumnSeries

            {
                Title = "Відвідуваність",
                Values = values}};
        Labels = attendance.Select(x => x.SubjectName).ToArray();
        DataContext = this;}
```

3.3.4 Перевірка на конфлікти в розкладі

Одна з важливих функцій інтерактивного інтерфейсу є перевірка наявності конфліктів у розкладі при додаванні або редагуванні пар. Це дозволяє уникнути ситуацій, коли одна й та сама аудиторія або викладач мають одночасно кілька занять.

Функція конфліктів реалізована у класі EditScheduleDialog.xaml.cs, у вигляді методу CheckConflicts і викликається до виконання SQL-запиту, що в свою чергу блокує збереження некоректного розкладу (листінг 3.4)

Лістинг 3.4 – Функція для перевірки на конфлікти в розкладі

```
      private bool CheckConflicts(int scheduleId, int roomId, string
dayOfWeek, string timeSlot)
      {
          string query = "SELECT COUNT(*) FROM Schedule " +
                         "WHERE RoomId = @RoomId AND DayOfWeek = @DayOfWeek
AND TimeSlot = @TimeSlot " +
                         "AND ScheduleId != @ScheduleId";
          using (SqlConnection connection = new
SqlConnection(connectionString)){
              SqlCommand command = new SqlCommand(query, connection);
              command.Parameters.AddWithValue("@RoomId", roomId);
              command.Parameters.AddWithValue("@DayOfWeek", dayOfWeek);
              command.Parameters.AddWithValue("@TimeSlot", timeSlot);
              command.Parameters.AddWithValue("@ScheduleId", scheduleId);
              connection.Open();
              int count = (int)command.ExecuteScalar();
              return count > 0;
      }
      }
```

Цей метод виконує перевірку на наявність іншого запису в таблиці Schedule з тим самим днем тижня, часом і номером аудиторії, але з іншим ідентифікатором (ScheduleId). Якщо такий запис існує — повертається true, що означає наявність конфлікту. Якщо конфлікт не виявлено — зміни дозволяється зберегти.

Лістинг 3.5 – Виклик перевірки конфлікту перед збереженням розкладу

```
      if (CheckConflicts(_scheduleId, int.Parse(RoomIdTextBox.Text),
DayOfWeekTextBox.Text, TimeSlotTextBox.Text))
      {
          MessageBox.Show("У вибраний час ця аудиторія вже зайнята!",
                          "Конфлікт у розкладі", MessageBoxButton.OK,
MessageBoxImage.Warning);
          return;
      }
```

При натисканні кнопки збереження система викликає перевірку CheckConflicts, передаючи поточні значення з полів редагування: RoomId, DayOfWeek, TimeSlot та ScheduleId. Якщо функція повертає true, відображається повідомлення про помилку, а діалог редагування не закривається — користувач може скорегувати дані вручну.

Цей підхід забезпечує базову цілісність розкладу та дозволяє уникати логічних конфліктів у межах однієї аудиторії. Код реалізований з використанням параметризованих запитів, що гарантує безпечне звернення до бази даних, запобігаючи SQL-ін’єкціям. Це рішення є ефективним і простим у підтримці, а його реалізація не потребує сторонніх бібліотек.

3.3.5 Запуск сповіщень при наближенні занять

У застосунку реалізовано механізм фонових сповіщень, що нагадують користувачам (студентам або викладачам) про початок пари. Сповіщення реалізовані через клас NotificationService, який працює як окремий фоновий потік, запущений після входу користувача в систему.

Сервіс сповіщень запускається у MainWindow.xaml.cs після успішної авторизації (лістинг 3.5)

Лістинг 3.6 – Запуск сервісу сповіщень

```
_notificationService.StartNotifications(_currentUserId,_currentUserRole)
```

Це викликає асинхронну перевірку розкладу для поточного користувача. Метод CheckAndNotify, що входить до складу класу NotificationService, виконує періодичну перевірку розкладу для поточного користувача та працює у фоновому потоці, не блокуючи головний інтерфейс програми, і відображає сповіщення, якщо заняття має розпочатися протягом наступних 15 хвилин (лістинг 3.6).

Лістинг 3.7 – Логіка методу CheckAndNotify

```
      private void CheckAndNotify(){while (true){
              var now = DateTime.Now;
              var nextClass = GetNextClassForUser(_userId, _role);
              if (nextClass != null && (nextClass.Time - now).TotalMinutes <=
15){Application.Current.Dispatcher.Invoke(() =>{
                      MessageBox.Show(
                          $"Нагадування:       скоро       почнеться       пара
({nextClass.Subject})                          в                              ауд.
{nextClass.Room}","Сповіщення",MessageBoxButton.OK,
                              MessageBoxImage.Information);});}
              Thread.Sleep(60000); }}
```

Користувач отримує повідомлення з інформацією про предмет, час та аудиторію де буде проводитися заняття. Сповіщення працює автоматично, незалежно від дій користувача.

3.4 Конфігурація, збирання та запуск WPF-застосунку

WPF – застосунок програми має стандартну структуру, притаманну C#-проєктам на .NET Framework. Основні етапи підготовки до запуску включають налаштування стартового вікна, опис ресурсів, зберігання параметрів у конфігураційному файлі та підключення NuGet-пакетів

3.4.1 App.xaml – стартове вікно та ресурси

Файл App.xaml використовується для задання загальних ресурсів і стартової точки запуску інтерфейсу. У проєкті налаштоване головне вікно MainWindow як початкове (лістинг 3.7).

Лістинг 3.8 – Фрагмент файлу App.xaml із визначенням стартового вікна LoginWindow

```
      <Application x:Class="UniversitySQL_System.App"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"Startup
Uri="LoginWindow.xaml">
      <Application.Resources>
```

```
<!-- Глобальні стилі або словники ресурсів --></Application.Resources>
</Application>
```

Файл забезпечує автоматичне завантаження вікна авторизації першим у черзі запитів.

3.4.2 App.config – підключення до бази даних

Файл App.config містить рядок підключення до SQL Server, який використовується у всій системі через ConfigurationManager (лістинг 3.8).

Лістинг 3.9 — фрагмент App.config:

```
<configuration><connectionStrings>
<add name="ConnectionString"
connectionString="Data
Source=localhost;Initial
Catalog=UniversityDB;Integrated                                    Security=True"
providerName="System.Data.SqlClient" />
</connectionStrings>
</configuration>
```

Це дозволяє централізовано змінювати підключення до бази без необхідності внесення додаткових змін в код застосунку.

Застосунок додатково використовує секцію конфігурації для зберігання загальних параметрів середовища

Лістинг 3.10 — Додаткові параметри конфігурації

```
<appSettings>
  <add key="DefaultLanguage" value="uk-UA" />
  <add key="EnableLogging" value="true" />
  <add key="LogFilePath" value="logs\app.log" />
</appSettings>
```

Ці параметри використовуються в коді для встановлення стандартної мови інтерфейсу та активації ведення журналів подій.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Перший запуск застосунку

Запустивши рішення universitySQL_System.sln у середовищі Visual Studio 2022 починаємо роботу з програмою планування навчального розкладу.

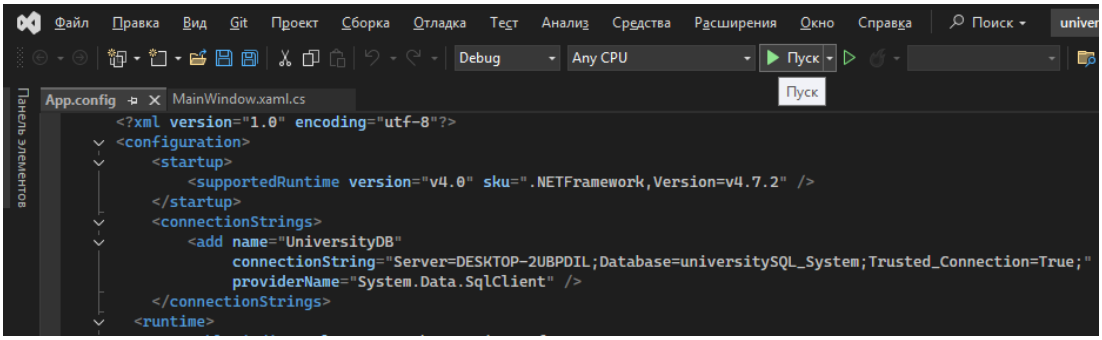

Рисунок 4.1 – Запуск програми у Visual Studio 2022

Натискаємо на кнопку «Пуск» та запускаємо програму. Під час першого запуску може знадобитися час, щоб всі необхідні для роботи бібліотеки завантажилися.

4.2 Початок роботи з системою

Після запуску програми ми бачимо вікно авторизації у систему. Після цього потрібно ввести логін та пароль від облікового запису.

Натискаємо кнопку «Увійти», якщо було введено вірні дані, то входимо у програму. У разі винекнення помилки в логіні або паролі користувач побачить повідомлення «Невірний логін або пароль» і після цього запропонує ввести дані ще раз. Якщо користувач неправильно вводить данні для авторазиції 3 рази, доступ до застосунку блокується на 1 годину.

4.3 Інтерфейс та функціонал адміністратора

Після авторизації з правами адміністратора, користувач отримує доступ до всіх функцій системи керування навчальним процесом.

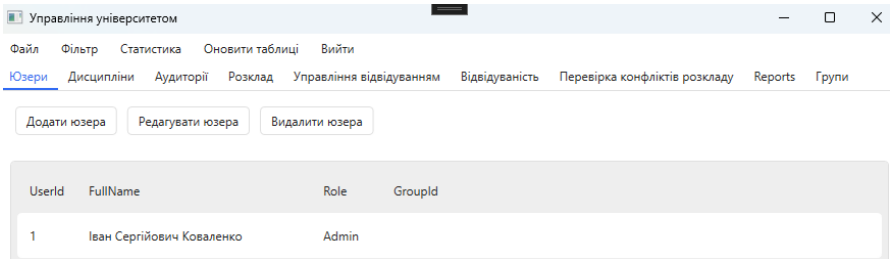

Рисунок 4.2 – Інтерфейс адміністратора

У головному меню розміщені елементи: «Користувачі», «Дисципліни», «Аудиторії», «Розклад», «Управління відвідуванням», «Відвідуваність», «Перевірка конфліктів розкладу», «Reports» та «Групи».

4.3.1 Розділ «Користувачі»

У цьому розділі адміністратор може побачити всіх користувачів та має можливість додавати, редагувати та видаляти їх за необхідностю.

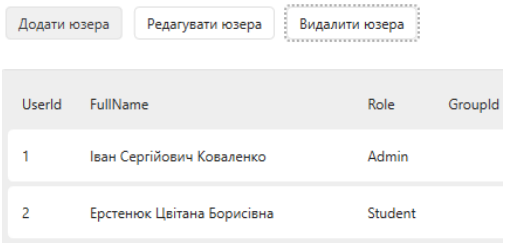

Рисунок 4.3 – Відобрадення користувачів

В меню «Додати користувача» є можливість вказати інформацію про нового користувача. До необхідої інформації відноситься ім’я, пароль для

доступу у застосунок, ріень доступу до системи (студент, вчитель або адміністратор), повне ім’я та GroupId, який містить інформацію про групу, до якої відноситься користувач.

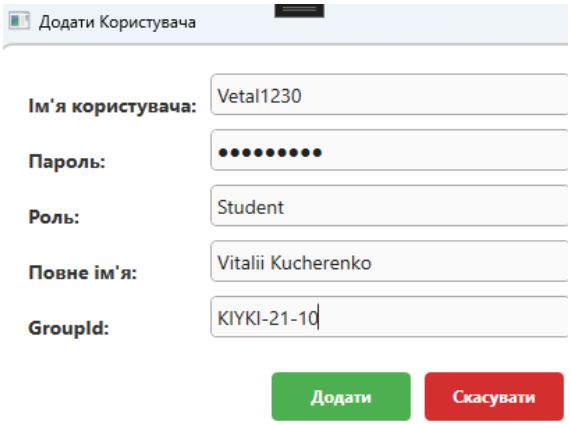

Рисунок 4.4 – Додавання нового користувача у систему

За допомогою кнопки «Редагувати» можемо змінювати дані вже наявного користувача. Також можемо використати функцію «Видалити», щоб прибрати непотрібного користувача.

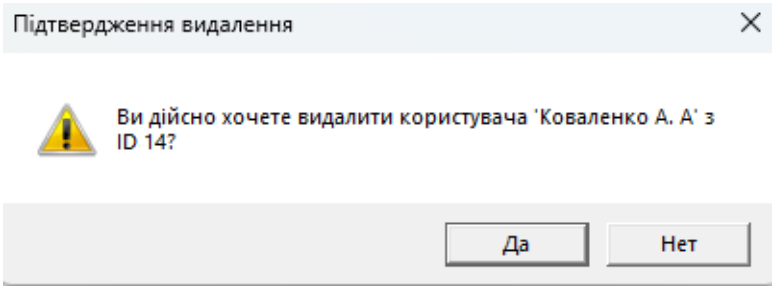

Рисунок 4.5 – Підтвердження видаленя користувача

4.3.2 Розділ роботи з дисциплінами

У розділі «Дисципліни» знаходиться інофрмація про всі наявні дисципліни з можливістю додавати нові та редагувати ті, що були додані до

цього. За допомогою цього блоку виконується робота з підготовки навчальних матеріалів та їх редагування в процесі навчання.

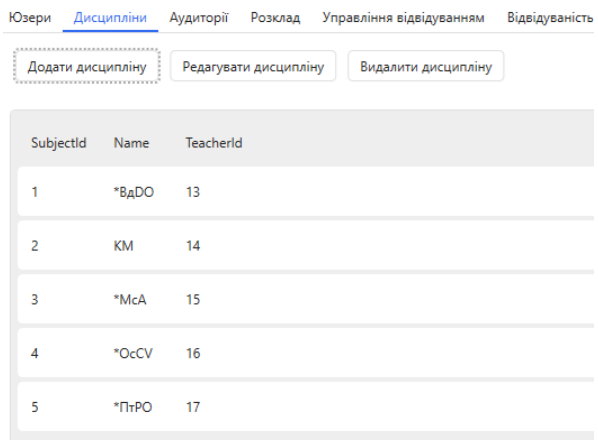

Рисунок 4.6 – Відобрадення дисциплін

В меню «Додати дисципліну» вводимо назву дисципліну та ім’я викладача, який буде проводити заняття. Інформація про назву дисципліни та виклада надається відповідним відділом унівеситету, та може буди змінена за необхідністю в продовж навчального семестру.

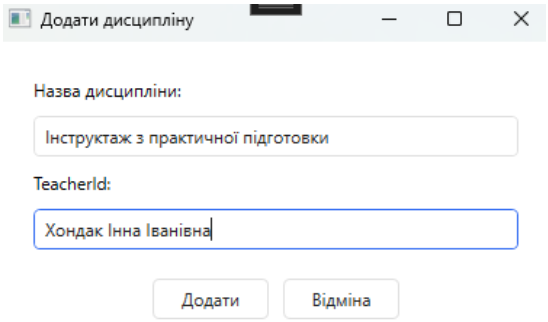

Рисунок 4.7 – Додавання нової дисципліни

За допомогою кнопки «Редагувати» змінюємо інформацію про наявну в системі дисцпліну, а за допомогою «Видалити» відповідно можемо прибати її з системи.

4.3.3 Розділ роботи з аудиторіями

На сторінці «Аудиторії» можно побачити всі наявні аудіторії. І ще маємо можливість додати новую аудіторії, при її наявності, змінити дані або повністю вадилати відповідні дані про аудіторію.

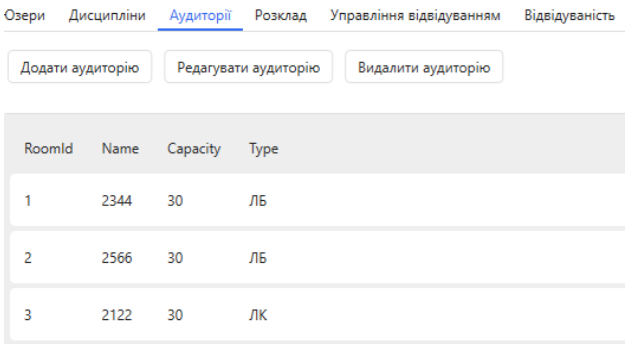

Рисунок 4.8 – Відобрадення аудіторій

В меню «Додати аудиторію» обираємо назву аудіторії, її вмістимість та тип занять, які будуть проводиться в процесі навчального семестру.

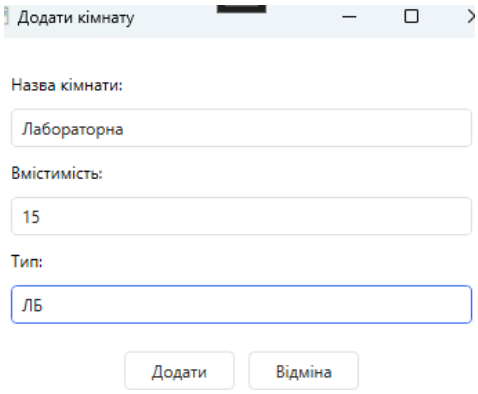

Рисунок 4.9 – Додавання аудиторії

При додаванні аудиторії для проведення занять система автоматично перевіряє зайнятість даної аудиторії, та при виникненні конфліктів, автоматично попереджає про це адміністратора.

4.3.4 Розділ Розклад

У розділі «Розклад» можно побачити актуальну інформацію про розклад, а також є можливість додати нове заняття, змінити його або видалити з системи. Якщо потрібно попередити про непередбачувані зміни інших адміністраторів або викладачів необхідно натиснути на «Репорт».

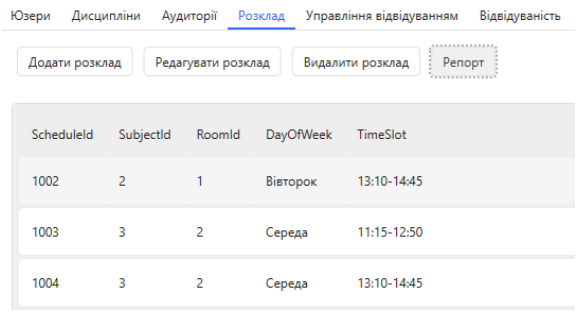

Рисунок 4.10 – Відобрадення розкладу

У вікні «Додати розклад» необхідно вести дані про предмет, аудіторію, день тижня, час проведення заннятя та інформацію про викладача.

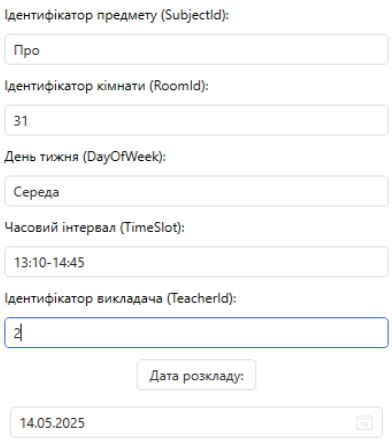

Рисунок 4.11 – Додавання нового заннятя

Також є можливість обрати дату проведення за допомогою вбудованого календаря: У вікні «Reports» користувачі мають можливість написати про

зміну аудиторії, дату та час проведення занять або про зміну відповідального викладача. Натискаємо на цю кнопку та вводимо повідомлення, яке побачать користувачі з нижчим рівнем доступу (викладачи та студенти).

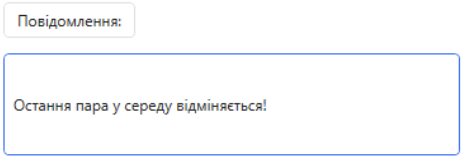

Рисунок 4.12 – Відправлення репорту

4.3.5 Управління відвідуванням та відвідування

В розділі «Управління відвідуванням» можемо додати потрібного студента до заняття.

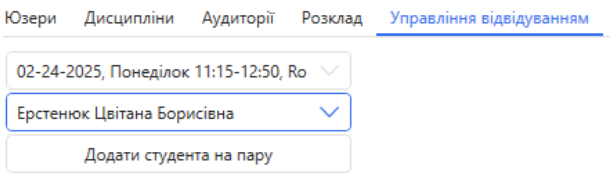

Рисунок 4.13 – Додання студента

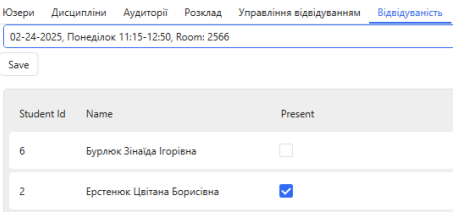

Рисунок 4.14 – Відмітка наявності студента

Також є можливість зафіксувати присутність. За допомогою цього інструмену викладачі можуть відслідковувати відвідуваність студентів.

4.3.6 Перевірка конфліктів у розкладі

За допомогою цього розділу можливо виявити конфлікти викладачів та аудиторій у розкладі в режимі реального часу.

Schedule1	Schedule2	TeacherId	TeacherName	DayOfWeek	TimeSlot	ScheduleDate
1026	1028	5	Гопкало Рудан Бажанович	Середа	10:00-12:00	5/14/2025 12:00:00 AM
1028	1026	5	Гопкало Рудан Бажанович	Середа	10:00-12:00	5/14/2025 12:00:00 AM

Рисунок 4.15 – Конфлікт викладачів

Schedule1	Schedule2	RoomId	RoomName	DayOfWeek	TimeSlot	ScheduleDate
1026	1028	3	2122	Середа	10:00-12:00	5/14/2025 12:00:00 AM
1028	1026	3	2122	Середа	10:00-12:00	5/14/2025 12:00:00 AM

Рисунок 4.16 – Конфлікт аудіторій

При виникненні конфліктних ситуацій застосунок автоматично сповіщає викладачів про виявлені проблеми. За допомогою цього інструменту запобігаються ситуації коли в одній аудиторії може проводитися одночасно дві дисципліни.

4.3.7 Розділ обробки повідомлень користувачів

У розділі «Reports» є можливість переглянути повідомлення, пов’язані зі змінами або проблемами у розкладі. Розділ призначений для збереження всіх повідомлень у зручному форматі та дозволяє вирішувати всі нагальні питання та проблеми більш оперативно, не чекаючи доки відповідний відділ опрацює отриману інформацію.

4.3.8 Розділ роботи з групами студентів

Останнім доступним для роботи користувача є пункт меню «Групи», який дозволяє керувати академічними групами. Це необхідно для можливсті оперативного редагування відповідної інформації.

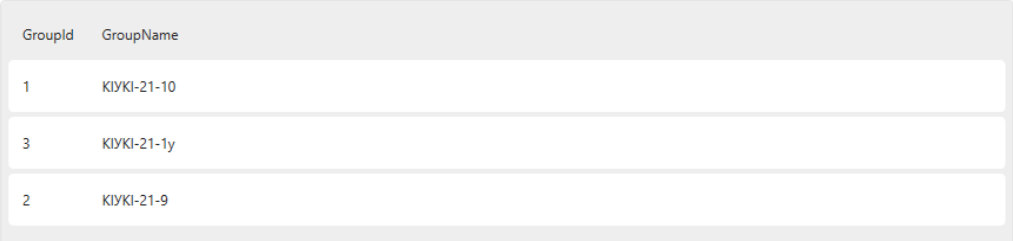

Рисунок 4.17 – Розділ Групи

4.3.9 Функції верхнього меню адміністратору

У верхній панелі головного вікна адміністратора розташовано набір додаткових функціональних кнопок, які покращують зручність взаємодії з даними та програмою:

– Файл – містить опції для імпорту/експорту даних, резервного копіювання та завантаження збереженого стану бази в разі необхідності резервного копіювання або відновлення даних;

– Фільтр – дозволяє встановити умови для фільтрації даних у таблицях та дозволяє проводити фільтрацію даних по параметрам, дозволяючи знаходити дані;

– Статистика – відкриває вікно перегляду аналітичних даних статистики відвідуваності груп, проведених занять викладачами та іншу інформацію про роботу студентів в процесі проходження ними навчальної програми;

– Оновити таблиці – виконує перезавантаження усіх таблиць, отримуючи актуальні дані з бази без необхідності повторного запуску

програми. Дана функція доступна користувачам з рівнем доступу «Адміністратор» та дозволяє отримати актуальні дані зі сторонніх сервісів, які підключені до застосунку;

– Вийти – завершує поточний сеанс роботи та повертає користувача на екран авторизації.

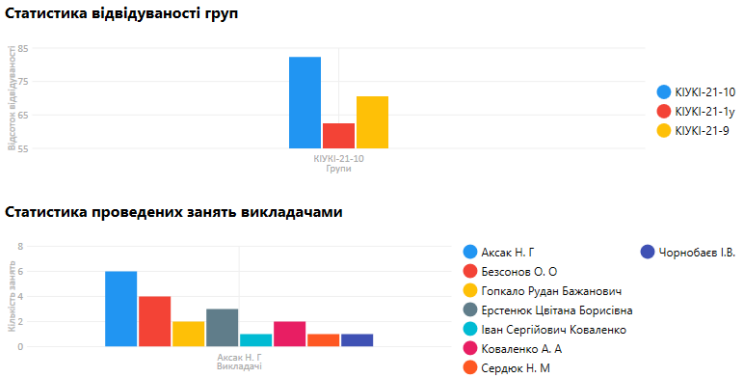

Рисунок 4.18 – Статистика відвідування та проведеня занять

4.4 Інтерфейс та функціонал викладача

Інтерфейс викладача є спрощеною версією адміністративного та орієнтований на перегляд розкладу, роботу з відвідуваністю та обробку повідомлень від студентів.

Викладач може лише переглядати дані про студентів, дисципліни та аудиторії. Але доступ до можливостей редагування інформації про користувачів, аудиторний фонд та розклад обежений і не дає змоги самостіно змінювати цю інформацію.

4.5 Інтерфейс та функціонал студента

Інтерфейс студента має найпростіший вигляд і надає доступ лише до перегляду особистого розкладу, закріплених дисциплін, списку викладачів та аудиторій. Усі дії виконуються лише в режимі читання – редагування даних

недоступне. Студент також має змогу переглядати повідомлення про зміни у розкладі. Такий інтерфейс мінімізує ризик помилок і зосереджує користувача лише на отриманні актуальної інформації.

Студент має можливість самостійно відмітитись на парі. Що дозволяє спростити процесс контролю за рівнем відвідування занять.

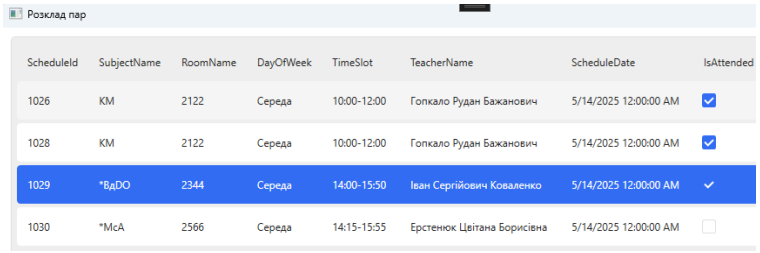

Рисунок 4.19 – Самостійна відмітка студентом присутності на парі

Також студент має свою додаткову статистику, яка надає інформацію про відсоток та кількість відвіданих занять.

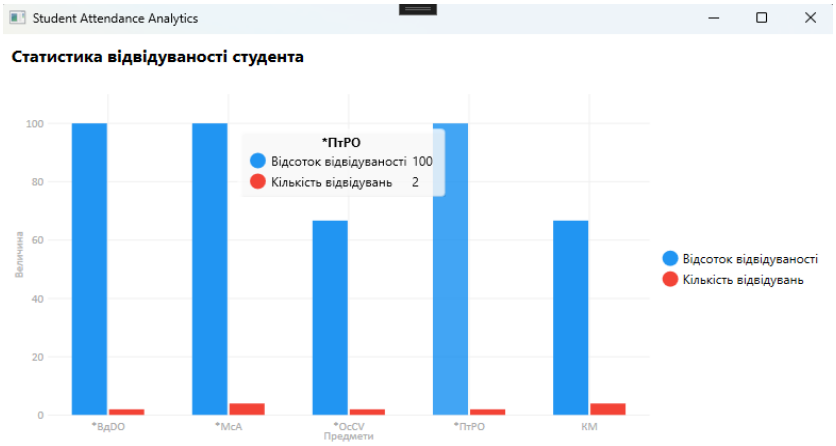

Рисунок 4.20 – Інтерфейс програми студента

4.6 Система сповіщень

У програмі реалізовано систему сповіщень, яка інформує користувача про наближення або початок заняття. Повідомлення автоматично

з’являються на екрані та містять інформацію про назву дисципліни, час та номер аудиторії. Це дозволяє краще організувати студентів та уникнути їх запізнення на заняття.

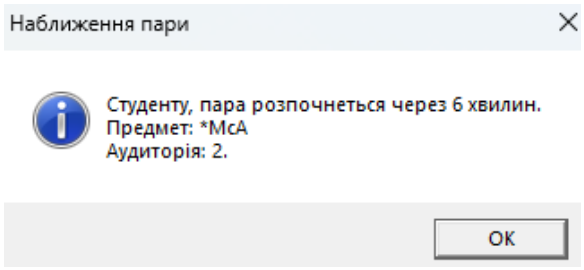

Рисунок 4.21 – Інформація про наближення початку заняття

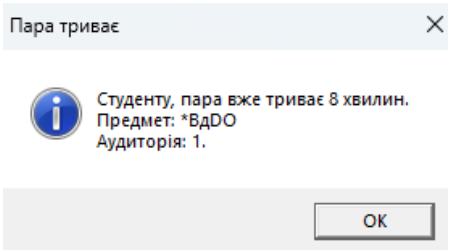

Рисунок 4.22 – Інформація про поточний статус заняття для студента

У процесі використання розробленого програмного рішення викладачі та студенти зазначали позитивні зміни в організації навчального процесу. Спостерігалося покращення показників відвідуваності занять, що пов’язано з підвищенням доступності інформації про зміни в розкладі та зручністю його перегляду. Завдяки оновленню даних у системі та виявленню конфліктів, зменшилась кількість випадкових накладок і перенесень пар, що сприяло стабільності навчального процесу. Користувачі також відзначали скорочення часу, необхідного для отримання відповіді або уточнення з боку викладача, завдяки чіткій структурі системи та можливості швидко переглядати актуальну інформацію. Підвищення зручності користування системою позитивно вплинуло на дисципліну студентів.

ВИСНОВКИ

Метою цієї роботи є створення клієнт-серверного програмного забезпечення для автоматизованого формування адаптивного навчального розкладу для студентів і викладачів. У процесі розробки було досліджено існуючі системи планування, що дало змогу виявити їхні основні недоліки, серед яких: складність врахування індивідуальних побажань, недостатня гнучкість до змін у навчальному процесі, відсутність зручної візуалізації даних і проблеми з інтеграцією зовнішнішніх сервісів.

У відповідь на ці виклики розроблено інформаційну систему з графічним інтерфейсом користувача, створену з використанням WPF і мови програмування C#. Програма побудована за архітектурною моделлю MVVM і працює з базою даних SQL Server через ORM-технологію Entity Framework. Застосунок забезпечує автоматичне формування розкладу з урахуванням ресурсів навчального закладу, перевірку на конфлікти, можливість налаштувань, ведення історії змін, а також аналіз ефективності планування.

Усі поставлені цілі кваліфікаційної роботи було виконано. Розроблене програмне забезпечення повінстю відповідає вимогам функціональності, підтримує масштабування і готове до впровадження у реальному освітньому середовищі.

Подальший розвиток проєкту передбачає розширення функціоналу – інтеграцію з LMS-платформами, підтримку мобільного доступу, синхронізацію з зовнішніми календарями, удосконалення аналітичного модуля і впровадження сценаріїв для змішаного та дистанційного навчання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Карплюк С. О., Тетяна Анатоліївна Вакалюк "Огляд функціональних можливостей програмного забезпечення для управління освітнім процесом закладу вищої освіти." Інформаційні технології і засоби навчання, 65.3 (2018): 262-276. URL: http://eprints.zu.edu.ua/27263/1/Карплюк_Вакалюк_WOS.pdf (дата звернення: 25.04.2025)

2. Боровик С. М., Жежерунов І. Ю. Адаптивні інформаційні системи у сфері вищої освіти. // Таврійський науковий вісник. – 2023. – № 4(38). – С. 43–47. – URL: http://tnv.org.ua/2023/issue38/tnv_38_2023_43.pdf (дата звернення: 27.04.2025)

3. Сакалюк О. Ю. "Реалізація проекту розробки програмного забезпечення автоматизованого керування процесом формування розкладу навчальних занять засобами пакета Gantt Project." Automation of technological and business processes, 13.3 (2021): 57-63. URL: https://journals.ontu.edu.ua/index.php/atbp/article/view/2142 (дата звернення: 27.04.2025)

4. Сакалюк О. Ю., Трішин Ф. А. "Аналіз процесу формування розкладу навчальних занять." (2019). URL: https://card-file.ontu.edu.ua/handle/123456789/10116 (дата звернення: 30.04.2025)

5. Hanzha A. S., Antonenko S. V., Izmailova M. K. "Огляд існуючих автоматизованих систем управління освітніми закладами." Актуальні проблеми автоматизації та інформаційних технологій, 26 (2022). URL: https://actualproblems.dp.ua/index.php/APAIT/article/view/213 (дата звернення: 01.05.2025)

6. Сімоненко А. "Метод спрямованого пошуку рішення для складання розкладу заняттів у навчальних закладах." Адаптивні системи

автоматичного управління, 2.43 (2023): 131-139. URL: https://asac.kpi.ua/article/view/292263 (дата звернення: 01.05.2025)

7. Медведєв М. Є. "Програмне забезпечення для календарного планування виконання IT-проєктів." (2023). URL: https://ela.kpi.ua/items/eaaaafc4-b3ac-47db-a970-81c8ccee2d46 (дата звернення: 05.05.2025)

8. Хоанг Шон. "Розробка інформаційної системи розкладу занять закладу вищої освіти." (2018). URL: http://dp.knute.edu.ua/jspui/bitstream/123456789/3603/1/Диплом%20Хоанг%20Шон%20ФОАІС%204-7.pdf (дата звернення: 05.05.2025)

9. Паньків Н. І., Романюк І. О. Використання WPF у реалізації сучасних графічних інтерфейсів користувача. // Вісник Хмельницького національного університету. – 2023. – № 1. – С. 112–117. – URL: https://visnyk.khnu.km.ua/article/view/2334 (дата звернення: 10.05.2025)

10. Кучеренко В. В. Клієнт-серверне рішення для адаптивного планування навчального розкладу студентів. // Матеріали Міжнародної науково-практичної конференції «Інтердисциплінарні наукові дослідження: стан, проблеми, перспективи», 6 грудня 2024 року. – URL: https://archive.liga.science/index.php/conference-proceedings/issue/view/inter-06.12.2024/108 (дата звернення: 11.05.2025)

11. Кучеренко В. В. Інформаційна система адаптивного навчального планування для студентів і викладачів. // Матеріали Міжнародної науково-практичної конференції «Інноваційні підходи в науці та освіті», 4 квітня 2025 року. – URL: https://archive.liga.science/index.php/conference-proceedings/issue/view/inter-04.04.2025/125 (дата звернення: 13.05.2025)

12. Семенов А. І., Левченко О. П. Застосування технології Entity Framework у проєктуванні клієнт-серверних додатків. // Комп’ютерні науки та інформаційні технології. – 2023. – № 2. – С. 88–92. – URL: https://csit.sumdu.edu.ua/archive/2023_2/csit_2023_2_88.pdf (дата звернення: 13.05.2025)

13. Міщук Н. Д., Багнюк Н. В. Серверне рішення для розумного будинку на базі Android: інтеграція з Home Assistant та OpenAI. // Матеріали конференції «Інтегровані технології у виробництві», 11–17 березня 2024 р. – Луцьк: ЛНТУ, 2024. – С. 57. – URL: https://conference.ikto.net/pub/akit_2024_11-17march_1.pdf (дата звернення: 15.05.2025)

14. Коваль А. І., та ін. "Порівняння об’єктно-орієнтованої та функційної парадигм програмування у проектуванні програмного забезпечення." (2021). URL: https://elar.khmnu.edu.ua/items/534e4b11-57f6-4c61-9cc2-e72ecc27cad8 (дата звернення: 20.05.2025)

15. Якушенко О. С., та ін. "Програмне забезпечення для моделювання статичних нейронних мереж." (2019). URL: https://er.nau.edu.ua/handle/NAU/40470 (дата звернення: 19.05.2025)

16. Крепич С. Я., Співак І. Я. "Якість програмного забезпечення та тестування: базовий курс." (2020). URL: http://dspace.wunu.edu.ua/bitstream/316497/39773/1/Навчальний%20посібник%20з%20якості%20ПЗ%20та%20тестування%20(1).pdf (дата звернення: 19.05.2025)

17. Голубничий Д. Ю., Колесник М. Ю. "Використання платформи .NET для розробки застосунків." (2024). URL: https://openarchive.nure.ua/entities/publication/ae62b1c8-b735-4cb4-a402-ea7cf33e0c72 (дата звернення: 19.05.2025)

18. Ichanska N., Ulko S. "Основні аспекти створення мобільних додатків та вибір інструментів їх розробки." Системи управління, навігації та зв’язку. Збірник наукових праць, 1.59 (2020): 74-78. URL: https://journals.nupp.edu.ua/sunz/article/view/1786 (дата звернення: 19.05.2025)

19. Карагодін Р. В. "Розробка програмної системи аналізу контенту WEB-сайтів на предмет унікальності з використанням мови програмування JavaScript на платформі Node.js." MS thesis. ТНТУ ім. І. Пулюя, 2021. URL: https://elartu.tntu.edu.ua/handle/lib/36828 (дата звернення: 25.05.2025)

20. Назаренко С. М. "Економічна оцінка аутсорсингу інформаційних технологій." Diss. Національний технічний університет "Харківський політехнічний інститут", 2022. URL: https://repository.kpi.kharkov.ua/items/36532139-46bc-4ee4-b023-9ed2eb8570fa (дата звернення: 30.05.2025)

21. Хорольський А. О., Косенко А. В. "Розробка та реалізація моделі для обґрунтування оптимальних технологічних схем відпрацювання викидонебезпечних вугільних пластів." Науковий вісник ДонНТУ, 1.8-2.9 (2022): 193-205. URL: https://www.researchgate.net/profile/Andrii-Khorolskyi/publication/368308671_Rozrobka_ta_realizacia_modeli_dla_obgruntuvanna_optimalnih_tehnologicnih_shem_vidpracuvanna_vikidonebezpecnih_vugilnih_plastiv/links/63e235482f0d126cd1946e19/Rozrobka-ta-realizacia-modeli-dla-obgruntuvanna-optimalnih-tehnologicnih-shem-vidpracuvanna-vikidonebezpecnih-vugilnih-plastiv.pdf (дата звернення: 01.06.2025)

22. Резанова В. Г., Марченко В. І. "Архітектурні патерни при розробці клієнт-серверних додатків." Інформаційні технології в науці, виробництві та підприємництві (2020). URL: https://er.knutd.edu.ua/handle/123456789/16387 (дата звернення: 01.06.2025)

23. Астістова Т. І., Мирець Р. В. "Використання патернів проектування для розробки програмного забезпечення." Інформаційні технології в науці, виробництві та підприємництві (2023). URL: https://er.knutd.edu.ua/handle/123456789/24121 (дата звернення: 02.06.2025)

24. Гавришко Я. "Моделювання за допомогою DDD та архітектурні патерни." (2021). URL: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/8c7db7ac-2781-4002-b0df-9b11c92cb5c2/content (дата звернення: 05.06.2025)

25. Прокопенко Н. О. Підходи до створення інформаційних систем для управління навчальними процесами. // Вісник ХНУРЕ. – 2022. – № 1. – С. 75–79. – URL: https://visnyk.khnu.km.ua/article/view/2543 (дата звернення: 07.06.2025)

26. Кіяшко Д. Г. "Вивчення засобів та архітектури для взаємодії мікросервісів у вебзастосунках." (2024). URL: https://openarchive.nure.ua/entities/publication/9383ab10-fae6-4ace-9501-e9c07c556f8b (дата звернення: 07.06.2025)

27. Ковальчук І. "Аналіз сучасних середовищ розробки програмного забезпечення." Наукові праці ХНУРЕ, 1.59 (2020): 74-78. URL: https://openarchive.nure.ua/bitstreams/553b4358-4ce0-4cf6-908d-bf85e9460095/download (дата звернення: 10.06.2025)

28. Томка В. "Windows Presentation Foundation: практичні рецепти." *Чернівецький національний університет*, (2023): с. 5–35. URL: https://archer.chnu.edu.ua/handle/123456789/6705 (дата звернення: 12.06.2025)

29. Рамазанов Р. Ш. "Дослідження технологій доступу до реляційних баз даних." *ХНУРЕ*, (2023): с. 88–92. URL: https://openarchive.nure.ua/bitstream/handle/123456789/18015/Robota_BD.pdf (дата звернення: 12.06.2025)

30. Мисько Б. В., Петришин В. С., Зелінська О. В. "Логічне проектування баз даних." Комп’ютерні технології обробки даних (2022): 217-218. URL: https://jktod.donnu.edu.ua/article/view/13101 (дата звернення: 14.06.2025)