Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Ю.В. Кравченко
«_____»_____2021 року

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

галузі знань 17 «Електроніка та телекомунікації»
за спеціальністю 172 «Телекомунікації та радіотехніка»

на тему:

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ «ЕЛЕКТРОННИЙ РОЗКЛАД ФАКУЛЬТЕТУ»

Виконав: студент групи МІТ-41

Старчевий А. О.
(прізвище ім'я по-батькові) (підпис)

Керівник: асистент кафедри мережевих та інтернет технологій
к.т.н. Махович О. І.
(посада, прізвище ім'я по-батькові) (підпис)

Київ 2021

Міністерство освіти і науки України
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Ю.В. Кравченко
«_____»_____2021 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ

Здобувачу вищої освіти Старчевому Анатолію Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи:
«Розробка інформаційної системи "Електронний розклад факультету"»

затверджена на засіданні кафедри МІТ «__4__» ___грудня_____ 2020 р. протокол № _8__
2. Термін здачі закінченої роботи «31» травня 2021 р
3. Вихідні дані до проекту (роботи)

4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити, обсяг – 35-40 стор.)
1. Дослідження методів та засобів розробки інформаційних систем.
2. Аналіз предметної області та проектування схеми бази даних.
3. Розробка Веб-додатку «Електронний розклад факультету».

5. Перелік графічного матеріалу 8-10 слайдів
Короткий огляд засобів розробки, фреймворку ASP.NET Core, шаблону проектування MVC, результат аналізу предметної області, схема бази даних, загальна структура проекту, особливості використання Entity Framework, приклади реалізації основних операцій з даними, авторизація та автентифікація користувачів, інтерфейс користувача.

Дата видачі завдання
Керівник роботи Асистент кафедри мережевих та інтернет технологій к.т.н. О.І. Махович
(підпис) (посада, прізвище, ім'я, по батькові)

Завдання прийняв до виконання
(підпис) (прізвище, ім'я, по батькові)

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Підготовчий	29.01.2021	
2	Розділ 1	01.03.2021	
3	Розділ 2	01.04.2021	
4	Розділ 3	01.05.2021	
5	Доповідь та слайди	27.05.2021	
6	Пояснювальна записка	31.05.2021	

Здобувач вищої освіти _____
(підпис) (прізвище, ім'я, по батькові)

Керівник _____Махович Олександр Іванович
(підпис) (прізвище, ім'я, по батькові)

РЕФЕРАТ

Пояснювальна записка: 41 с., 29 Рисунок, 5 табл., 1 додаток, 21 джерел.

Об’єкт дослідження: інформаційна система «Електронний розклад факультету»

Предмет дослідження: розробка веб-додатку «Електронний розклад факультету»

Мета роботи (проекту): провести теоретичний аналіз методичної літератури по розробці інформаційної системи. Проаналізувати предметну область. Використати набуті знання та методичний інструментарій для проектування схеми бази даних. Використати спроектовану схему бази даних для розробки веб-додатку «Електронний розклад факультету».

В роботі проведено аналіз методичної літератури по розробці інформаційних систем та веб-додатків.

Спроектовано та побудовано базу даних для інформаційної системи «Електронний розклад факультету».

Розроблено веб-додаток, на якому можна продивлятися розклад, інформацію про викладачів, пари, а також адміністратору змінювати, видаляти та додавати будь-яку інформацію.

Результати здійснених у дипломному проекті досліджень можуть бути використані у розробці сайту або бази даних для розкладу факультету або деякої іншої учбової установи.

Ключові слова: СУБД MICROSOFT SQL SERVER, БАЗА ДАНИХ, ІНФОРМАЦІЙНА СИСТЕМА, РОЗКЛАД ФАКУЛЬТЕТУ, ASP.NET, ENTITY FRAMEWORK, MVC, РОЗМЕЖУВАННЯ ПРАВ ДОСТУПУ

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних

СУБД – Система управління базами даних

MVC – Model-View-Controller

T-SQL – Transact-SQL

HTML – HyperText Markup Language

ВСТУП

У зв’язку з поширенням та розвитком дистанційного навчання в останні роки є дуже важливою інформатизація та автоматизація навчання. Електронний розклад факультету є частиною інформатизації навчання. Не дивлячись на велику поширеність та потребу ця технологія є недостатньо вивченою. Велика кількість університетів та факультетів, навіть якщо і мають сайт з електронним розкладом, то він доволі незручний або не містить достатньо функціонала. Зручний електронний розклад є дуже важливим, оскільки під час навчання часто відбуваються заміни або відміни пар, аудиторій, викладачів та іншого. А багатофункціональний і зручний розклад дозволить уникнути неприємностей, пов’язаних з цим, наприклад коли студент або викладач прийшов на пару, якої немає, тому що йому забули сказати або він прослухав.

Головною частиною роботи електронного розкладу є зручна та гарно спроектована база даних. Виділимо найпопулярніші компанії, які працюють в області створення систем управління для баз даних. Серед них є такі як: Oracle, Microsoft та співтовариство PostgreSQL.

Практичним значенням одержаних результатів є зручна, гарно розроблена та спроектована база даних, яка може використовуватися як основа для розробки схеми бази даних для розкладу будь-якого навчального закладу, так і вже повністю робочий прототип. Також серед основних результатів буде створений прототип веб-додатку, який працюватиме на основі спроектованої схеми бази даних. Цей сайт також можна буде використовувати як прототип веб-додатку для сайту з розкладом.

1 ДОСЛІДЖЕННЯ МЕТОДІВ ТА ЗАСОБІВ РОЗРОБЦІ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 SQL

SQL являється інструментом, призначеним для організації, управління, вибірки і оброки інформації, яка зберігається в базі даних [1]. SQL має велику кількість функціональних можливостей. Розглянемо їх:

1. Вибірка даних. Можливість вибирати частину або всю інформацію з бази даних;
2. Обробка даних. Дозволяє обробляти базу даних. Наприклад, додавати, видаляти або відновлювати дані;
3. Спільне використання. У SQL можна працювати декільком користувачам одночасно, і при цьому зміни, які робить один із користувачів не будуть знищувати дані, які вносить інший користувач;
4. Визначення даних. В SQL є можливість визначати структуру і організацію даних, що зберігаються та взаємозв’язки між елементами даних;
5. Управління доступом. В SQL є можливість в деякій мірі обмежити користувача. Наприклад, заборонити йому додавання і зміну даних. Це може захистити базу даних від несанкціонованого доступу;
6. Цілісність даних.

SQL складається з п’яти частин:

DML – мова керування даними. Зазначено у таблиці 1.1.

Таблиця 1.1 – DML

	Команда	Опис
1	DELETE	Видаляє запис
2	UPDATE	Змінює запис
3	INSERT	Створює запис

DDL – мова визначення даних. Зазначено у таблиці 1.2.

Таблиця 1.2 - DDL

	Команда	Опис
1	CREATE	Створити нову таблицю або інший об’єкт в базі даних
2	DROP	Видаляю цілу таблицю або інший об’єкт у базі даних
3	ALTER	Змінює існуючий об’єкт, наприклад таблицю

DCL – мова контролю даними. Зазначено у таблиці 1.3.

Таблиця 1.3 - DCL

	Команда	Опис
1	GRANT	Дає привілеї користувачеві
2	REVOKE	Відбирає привілеї від користувача

TCL – група операторів для управління транзакціями. Зазначено у таблиці 1.4.

Таблиця 1.4 - TCL

	Команда	Опис
1	COMMIT	Застосовує транзакцію
2	ROLLBACK	Відкатує всі зміни, зроблені в даній транзакції
3	SAVE POINT	Робить проміжну точку збереження в транзакції

DQL – мова структурованих запитів, за допомогою неї пишуться спеціальні запити SELECT до бази даних.

1.1.1 Архітектура SQL

В будь-якій СУБД, коли ви виконуєте запит SQL, СУБД інтерпретує цей запит. Цей процес включає такі компоненти:

1. Диспетчер запитів;
2. Рушій СУБД;
3. Парсер та оптимайзер;
4. SQL Query Engine та інші.

Розглянемо на рисунку 1.1 архітектуру SQL [2].

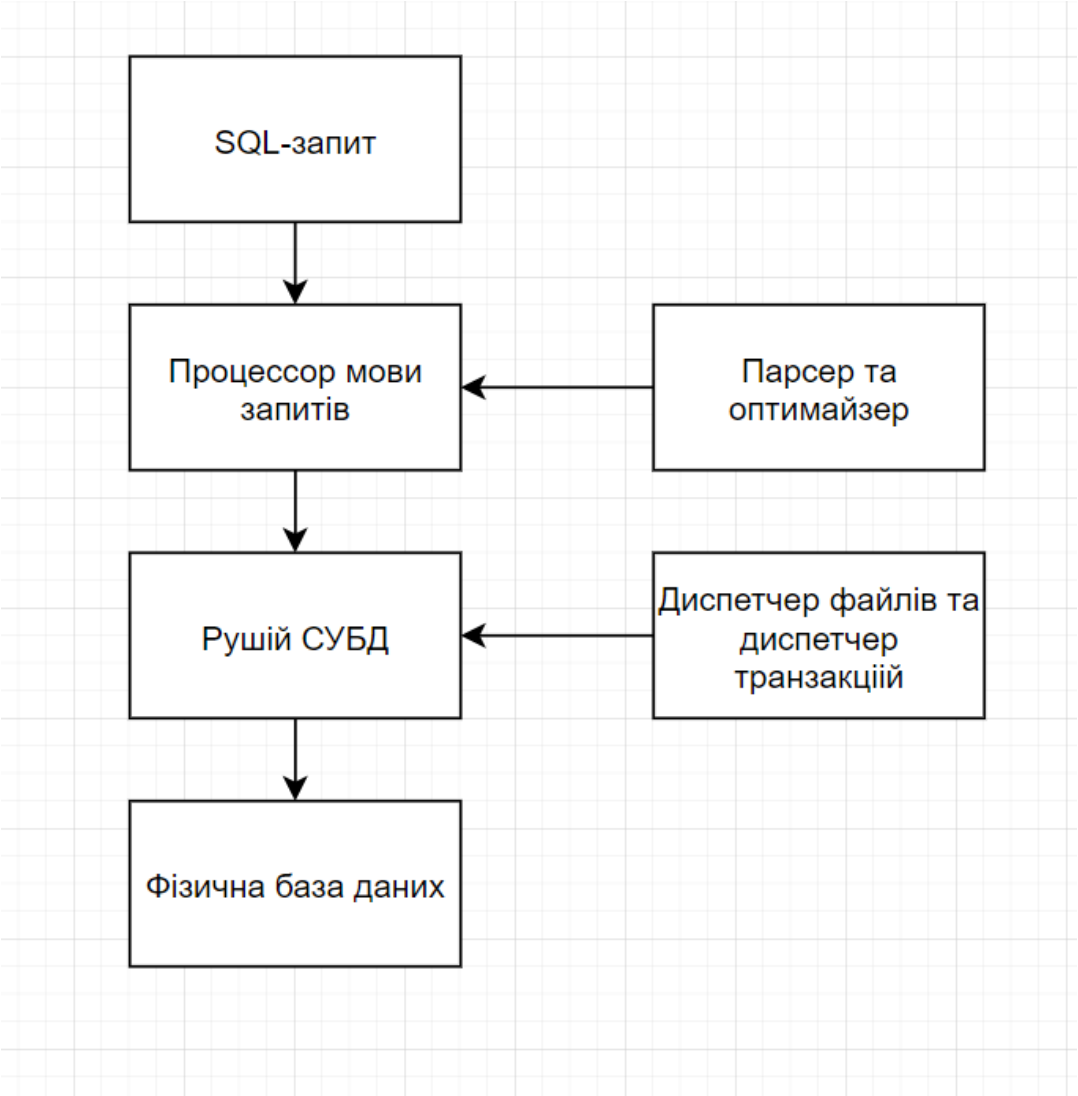


Рисунок 1.1 – Архітектура SQL

1.2 Системи управління базами даних

База даних – це організований набір даних, які зазвичай зберігаються та мають електронний доступ з комп’ютерної системи [3]. Система управління базами даних – це програмне забезпечення, яке взаємодіє з кінцевими користувачами, програмами та самою базою даних для збору та аналізу даних [4]. СУБД має дуже велику кількість функціональних можливостей. Розглянемо головні з них на рисунку 1.2.

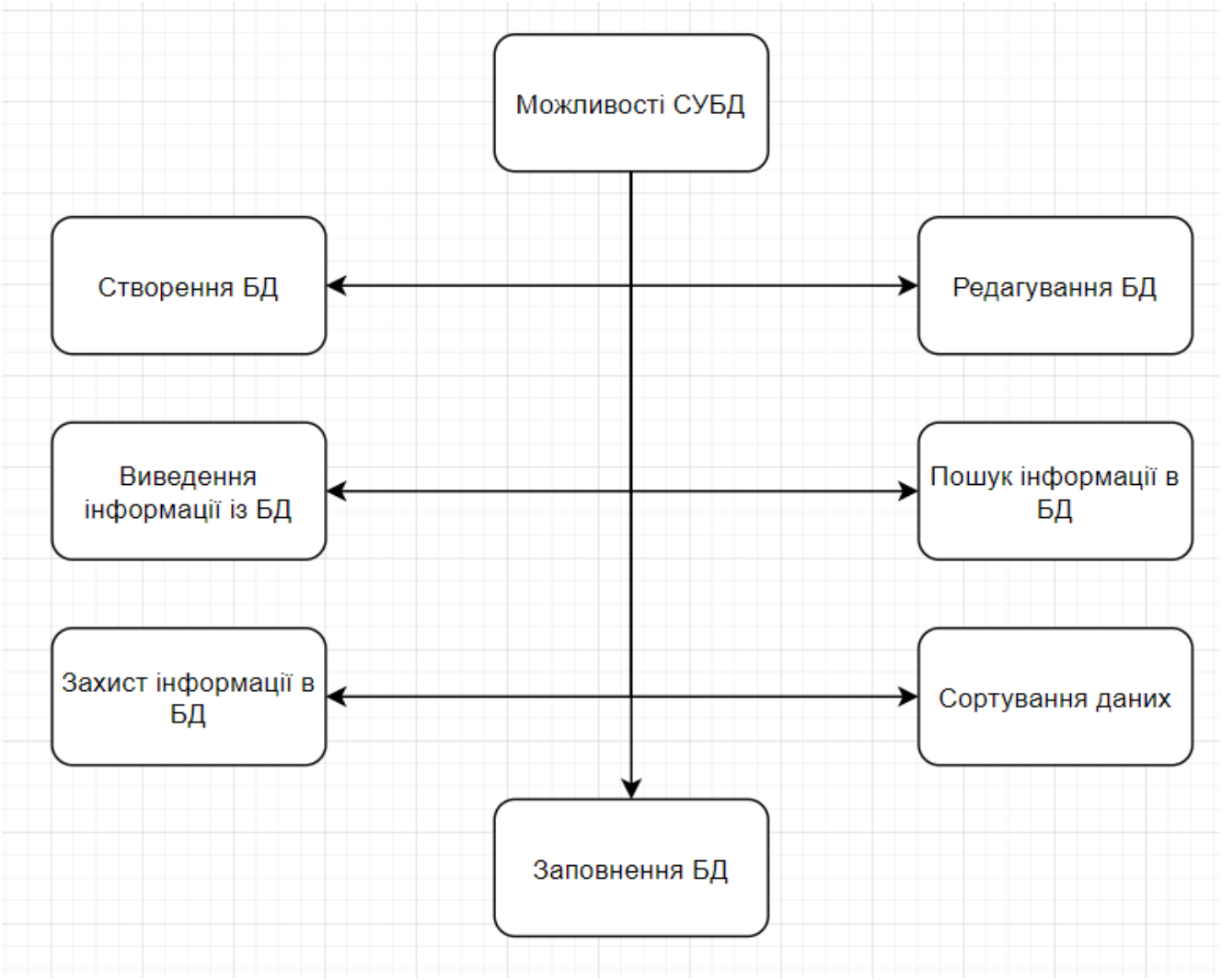


Рисунок 1.2 - Функціональні можливості СУБД

1.2.1 Вибір СУБД

1.2.1.1 Oracle Database

Oracle Database – це об’єктно-реляційна система управління базами даних, створена компанією Oracle [5]. Ідея створення цієї СУБД виникла у Ларрі Еллінсон у 1977-му році.

Дана СУБД має свої переваги:

1. Може підтримувати велику кількість користувачів. Також в базі даних цієї СУБД можуть працювати декілька користувачів одночасно;
2. Немає суперництва між різними видами даних;
3. В СУБД Oracle є підтримка транзакцій;
4. Також серед переваг варто виділити, що Oracle є доволі відмовостойкою. Будь-які збої не призводять до зупинки бази даних. Також є можливість не відключати всю систему, коли треба перезавантажити дані якогось певного додатка;
5. Бази даних в цій СУБД дуже легко переносяться між різними операційними системами з невеликими змінами.

Також важливо виділити деякі недоліки даної СУБД. Серед них є такі:

1. Занадто висока вартість, особливо для невеликих компаній.
2. Невелика можливість масштабування. Oracle може зажадати великої кількості ресурсів одразу після встановлення.

1.2.1.2 PostgreSQL

PostgreSQL – це система управління реляційними базами даних із відкритим кодом, яка підтримується співтовариством PostgreSQL, та поширюється за допомогою ліцензії PostreSQL. Автори підкреслюють масштабованість та відповідність SQL [6].

В PostgreSQL інтегровані транзакції із властивостями Atomicity, Consistency, Isolation, Durability (ACID), зовнішні ключі, тригери, автоматично оновлюванні подання та збережені процедури. PostgreSQL призначений як для обробки невеликих баз даних на персональних комп’ютерах, так і для веб-сервісів із великою кількістю одночасних користувачів. Основною операційною системою для PostgreSQL є macOS, але вона також доступна і для Linux та Windows. Розглянемо головні можливості цієї СУБД.

Можливості PostgreSQL

Розглянемо основні можливості PostgreSQL у таблиці 1.5.

Таблиця 1.5 – Можливості PostgreSQL

Функції	Виконуються на сервері, а не в клієнті БД.
Тригери	Можливість визначати тригери
Індекси	GIN, хеш, R-дерево, GIST, B-дерево
Multiversion Concurrency Control	Механізм MVCC, за допомогою якого можуть працювати паралельно декілька користувачів. Вони не будуть бачити зміни один одного до моменту фіксації транзакції.
Розширення	Підтримується можливість створення модифікацій.
Спадкування	Можливість успадкування від інших таблиць

1.2.1.3 MySQL

MySQL – це система управління реляційними базами даних з відкритим кодом, яка підтримується компанією Oracle. Вона поширюється безкоштовно, за допомогою «Загальної публічної ліцензії» [7].

Також СУБД MySQL Server має важливі переваги, такі як:

1. СУБД може управляти користувачами;
2. Полегшує тестування цілісності даних;
3. Універсальність та поширеність;
4. Простота, а наявність плагінів ще більше спрощує роботу з БД;

5. Масштабованість;
6. Швидкість.

Також система MySQL має і деякі недоліки, серед яких виділимо такі: як і будь який продукт із відкритим кодом, має недостатню надійність. Також серед недоліків є низька швидкість розробки.

1.2.1.4 Microsoft SQL Server

Microsoft SQL Server – це система управління реляційними базами даних, розроблена корпорацією Майкрософт [8]. Майкрософт має дуже велику кількість видань, спрямованих на різних клієнтів. Ці видання спрямовані як на невеликі додатки, спрямовані на декілька користувачів, так і на величезні, на яких будуть працювати цілі корпорації. Ця СУБД має масу переваг. Розглянемо деякі з них:

1. Ця СУБД має зручну інтеграцію з іншими продуктами Microsoft: Visual Studio та Access;
2. Розмір бази даних до 8 Мегабайт. Це дозволяє створювати великі бази даних;
3. Microsoft SQL Server легко масштабується, що дозволяє працювати як на персональному комп’ютері, так і серверах деякої компанії;
4. Велика кількість рутинних завдань автоматизовано;
5. Можна створювати профілі;
6. Вбудовано пошук по тексту, фразах, словах;

7. Підтримується синхронізація між різними комп'ютерами за допомогою інтернету.

Як і будь який продукт, Microsoft SQL Server має свої недоліки. Навіть при ретельному налаштуванні, вона може зайняти велику кількість ресурсів комп’ютера. Також ця СУБД є досить дорогою.

В результаті обмірковування всіх переваг і недоліків усіх чотирьох систем управління базою даних було прийнято рішення використовувати MS SQL Server у цій дипломній роботі. Основним аспектом при виборі цієї СУБД було те, що вона легко інтегрується у Visual Studio.

Якщо ми вирішіли робити дипломну роботу на базі MS SQL Server, важливо буде розповісти про Transact-SQL.

1.3 Transact-SQL

Transact-SQL – це набір розширень для програмування від Microsoft до мови SQL для СУБД Microsoft SQL Server [9]. Розширення включає в себе:

1. Локальні та глобальні змінні;
2. Аутентифікація за допомогою Microsoft Windows;
3. Управляючі оператори.

Мова T-SQL є головним компонентом використання MS SQL Server.

Розглянемо принцип створення бази даних в T-SQL на прикладі створення таблиці Teacher із даної роботи. Про схему бази даних даної роботи розказано в 2 розділі. Спочатку створюється база даних за допомогою коду:

Create Database {Назва бази даних} Go

Далі створюємо таблицю в базі даних та заповняємо її даними. Це виконується за допомогою коду:

```
CREATE TABLE Teacher(
        Id int Primary Key,
        FullName nvarchar(50) NOT NULL,
        img nvarchar(50) NULL,
) GO


INSERT Teacher
(FullName, img)
VALUES
('Махович Олександр Іванович', '/images/pic02.jpg'),
('Герасименко Оксана Юрієвна', '/images/pic02.jpg'),
('Кравченко Юрій Васильович', '/images/pic02.jpg'),
GO
```

Отже, розглянемо результати роботи коду вище на рисунку 1.3.

Results | Messages

	Id	FullName	img
1	1	Махович Олександр Іванович	/images/pic02.jpg
2	2	Герасименко Оксана Орієвна	/images/pic02.jpg
3	3	Кравченко Юрій Васильович	/images/pic02.jpg

Рисунок 1.3 – Результат створення таблиці в T-SQL

1.4 ASP.NET Core та .NET Framework

ASP.NET Core це платформа від компанії Microsoft, що служить для створення веб-сайтів і веб-додатків. В ній можна писати код на всіх мовах, які входять в .NET Framework [10]. ASP.NET Core являється якісною зміною платформи ASP.NET, повною її революцією. На основі цієї платформи можна створювати сайти різної складності і тематики.

ASP.NET Core швидше ніж звичайні скриптові мови. Причиною цього є те, що мова C#, на якій працює ця платформа, є компільованою мовою програмування, а тому вона компілюється лише раз, при першому запуску проекту, а далі запускається з кешу.

ASP.NET Core є доволі популярною платформою. Є велика кількість популярних сайтів, які на написані на даній платформі. Серед цих сайтів можна виділити такі:

1. сайт компанії Microsoft;
2. сервіс для реєстрації доменних імен GoDaddy;
3. StackOverflow - це найбільший онлайн форум;
4. сайт Dell та інші.

На рисунку 1.4 можна розглянути статистику використання різних фреймворків на 2020 рік.

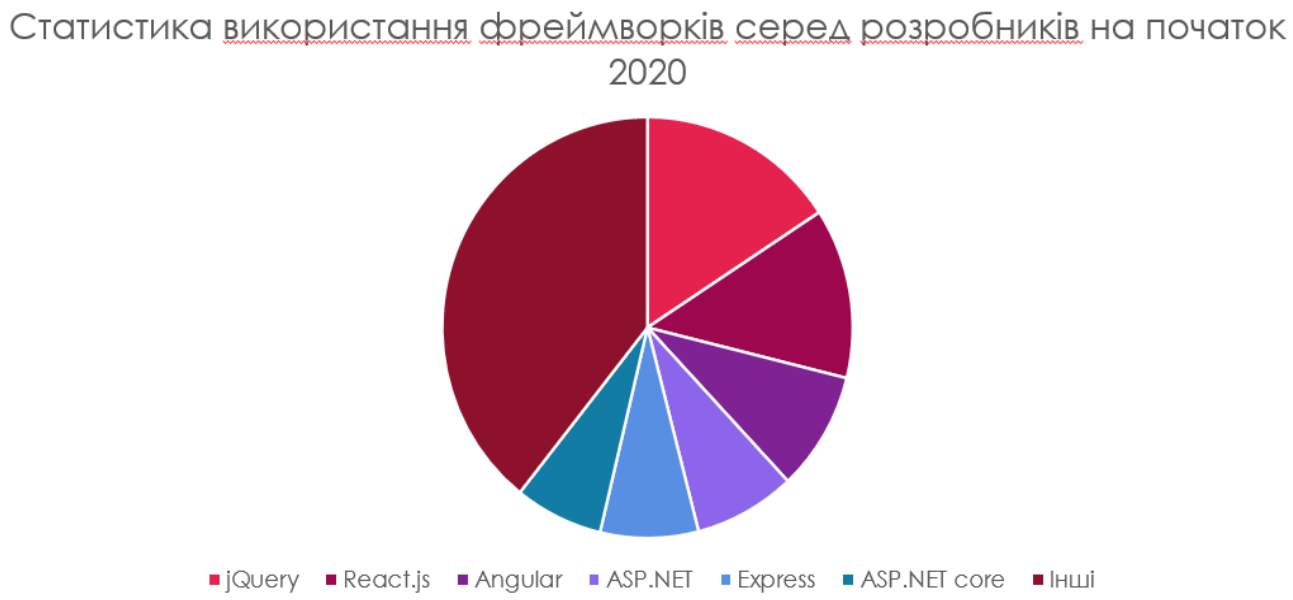


Рисунок 1.4 – Статистика використання різних фреймворків [11]

В ASP.NET Core використовується схему Model-View-Controller (Модель-Представлення-Контролер). Кожний компонент в MVC відповідає за певні дії. Розглянемо принцип роботи цієї схеми на рисунку 1.5. Клієнт виконує певну дію, посилаючи запит на контролер. Потім контролер трактує дію клієнта і відправляє запит у модель. Модель в свою чергу надає дані з сервера бази даних або додає, видаляє, змінює існуючі дані. Представлення же являється кодом Razor, який відповідає за зовнішній вигляд сайту.

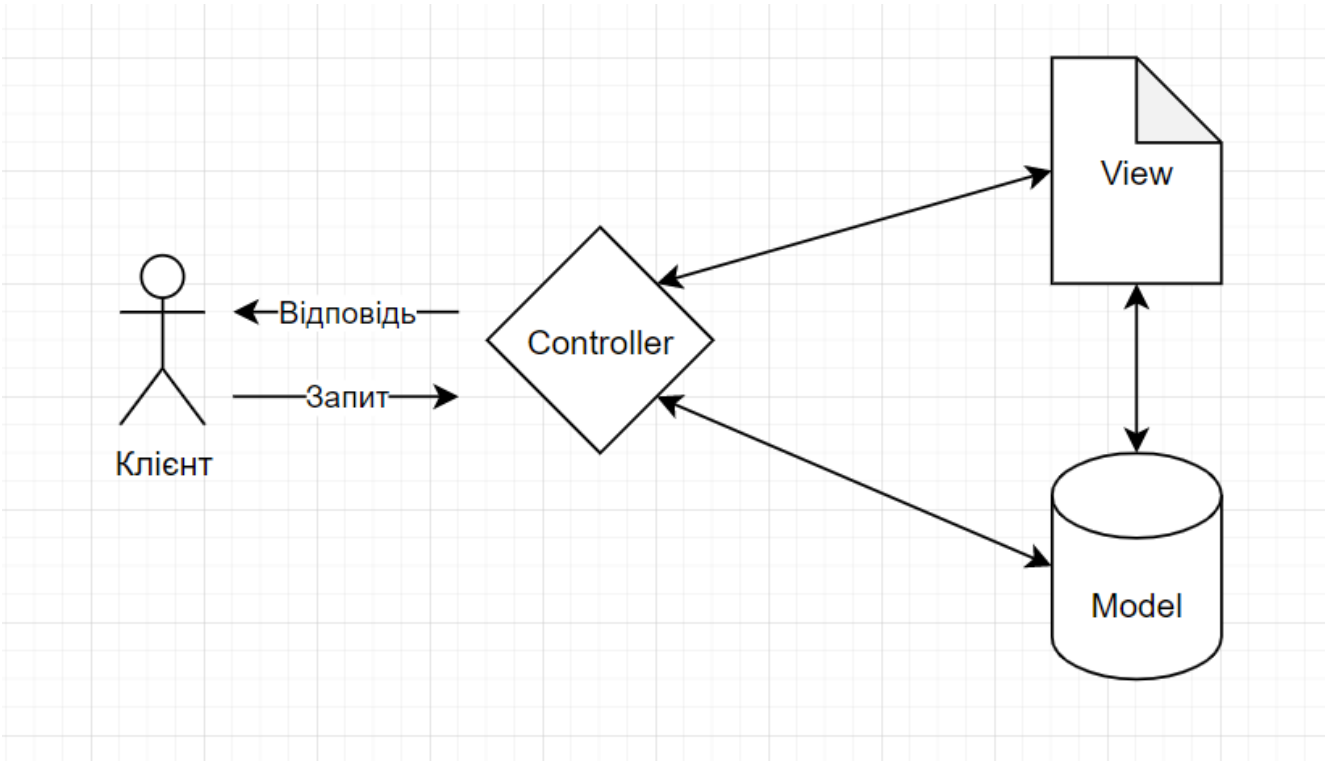


Рисунок 1.5 – Модель Model-View-Controller

Модель MVC має багато переваг. Так, розробку MVC легко розбити на етапи. За рахунок цього етап тестування стає простішим. Також таку модель легко масштабувати, просто додаючи нові моделі на контролери. За рахунок цього модель MVC є дуже популярною і використовується в багатьох проектах не тільки на платформі ASP.NET Core, але і в PHP, Java, JavaScript та інших популярних мовах програмування.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОЕКТУВАННЯ СХЕМИ БАЗИ ДАНИХ

2.1 Аналіз предметної області

Предметна область електронного розкладу факультету при проектуванні бази даних включає велику кількість елементів і взаємозв'язків між ними. На початковому етапі проектування інформаційної системи необхідно ретельно проаналізувати існуючі дані і упорядкувати. Нижче опишемо підхід для подальшого проектування бази даних.

Множина задач обробки даних:

Оновлення даних

Видалення даних

Додавання даних

Вибірка даних

Множина інформаційних елементів бази даних буде включати 18 елементів:

Кількість аудиторних годин

Кількість неаудиторних годин

Викладач

Вид заняття

Форма навчання

Назва пари

Номер аудиторії

Номер пари

Час початку пари

Час закінчення пари

Номер курсу

Кафедра

Підгрупа

Група

Освітня програма

День тижня

Номер пари

Коментар до пари (необов’язковий)

2.1.1 Проектування схеми бази даних розкладу

Розділимо дану інформацію на 3 основні таблиці: пара на рисунку 2.1, група - рисунок 2.2 та розклад – рисунок 2.3. Давайте спочатку створимо 3 основні таблиці в нашій базі даних для відображення цих 3-ьох основних пунктів.

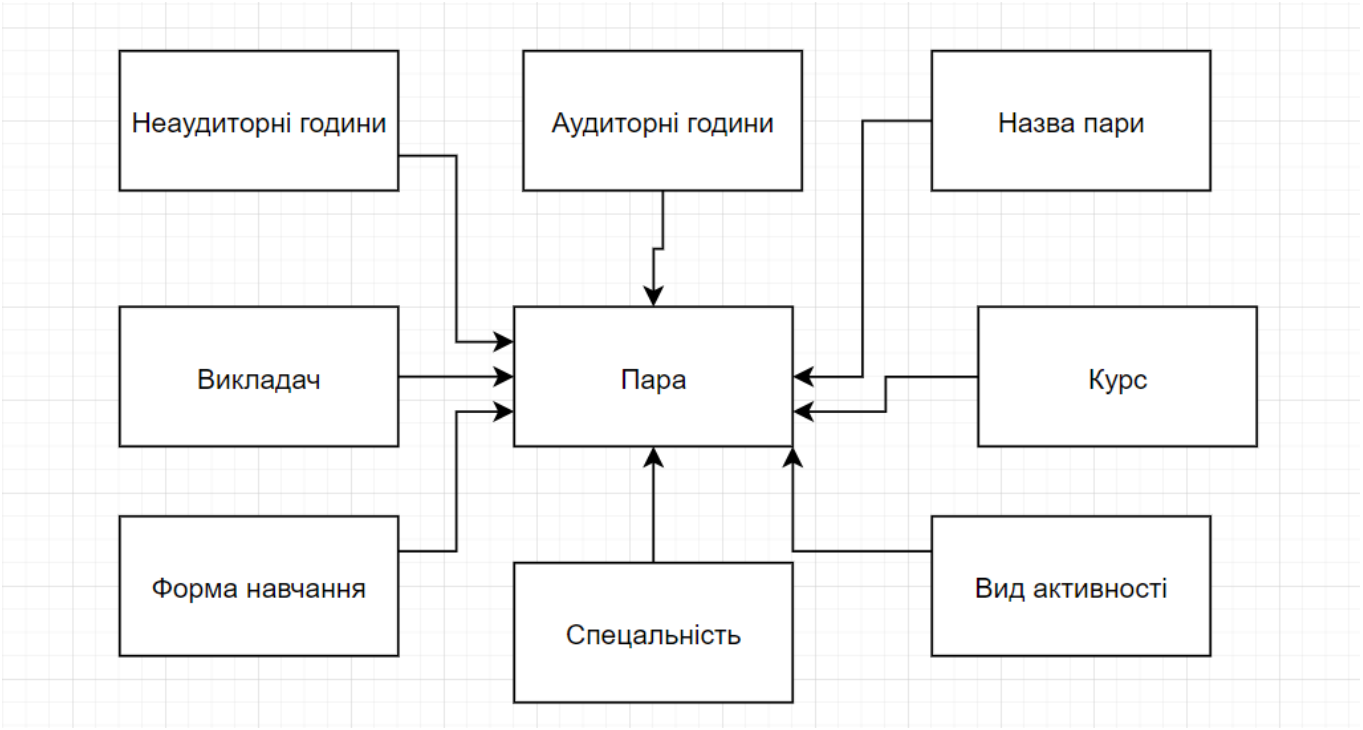


Рисунок 2.1 – Таблиця пара

Отже, виберемо інформаційні елементи, які ми підключимо до таблиці пара: неаудиторні години, аудиторні години, назва пари, курс, спеціальність, форма навчання, вид активності, викладач.

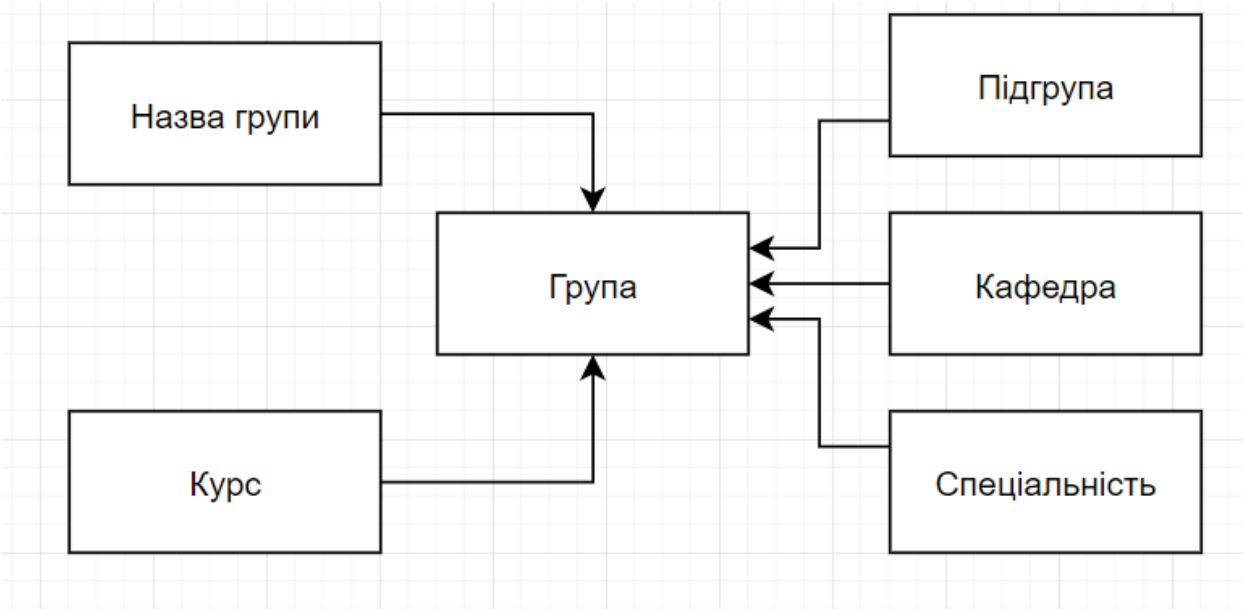


Рисунок 2.2 – Таблиця група

До таблиці група підключимо такі елементи як: підгрупа, кафедра, спеціальність, курс, назва групи.

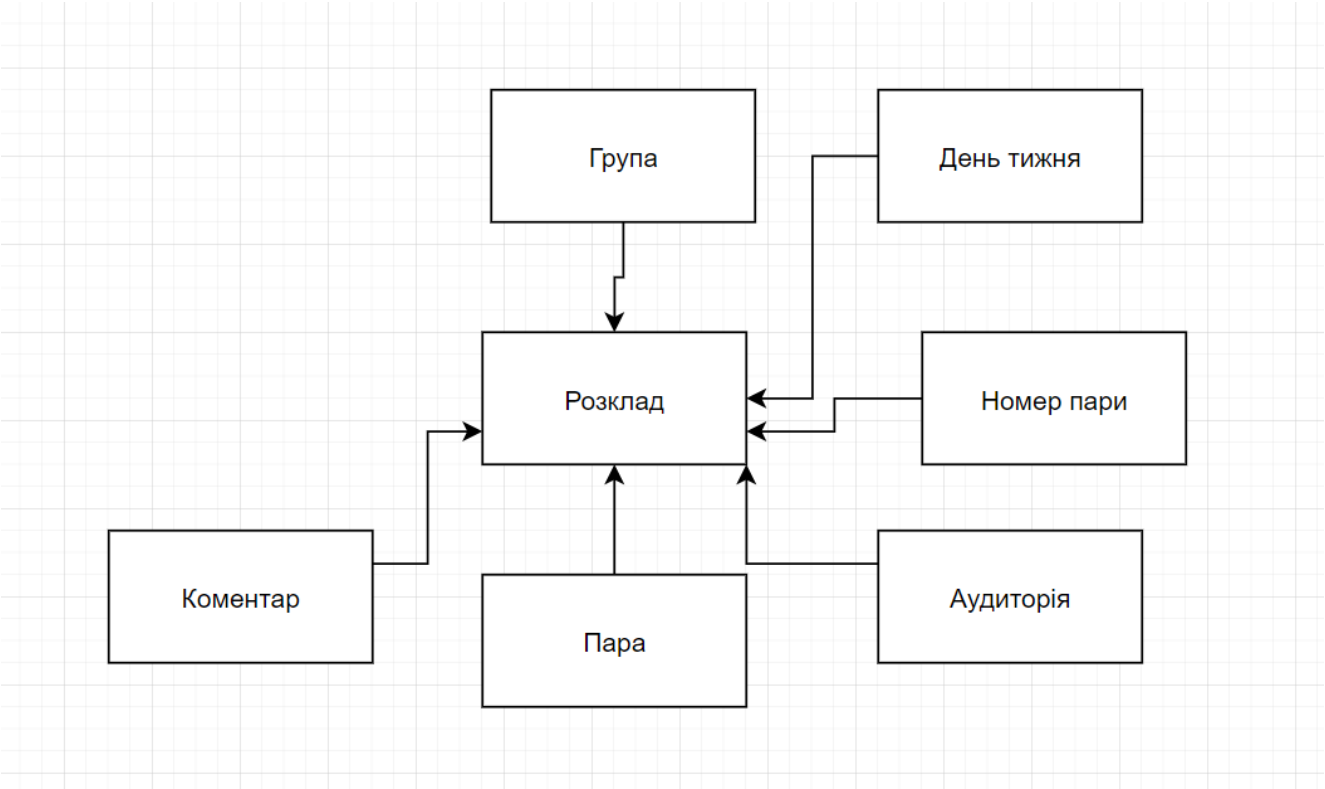


Рисунок 2.3 – Таблиця розклад

Також створимо таблицю розклад та підключимо такі структурні елементи: група, день тижня, парність, номер пари, аудиторія, пара та коментар до пари.

Отже, на основі даних 3 таблиць БД, створимо фінальну версію схеми, об’єднавши ці 3 схеми в одну на рисунку 2.4.

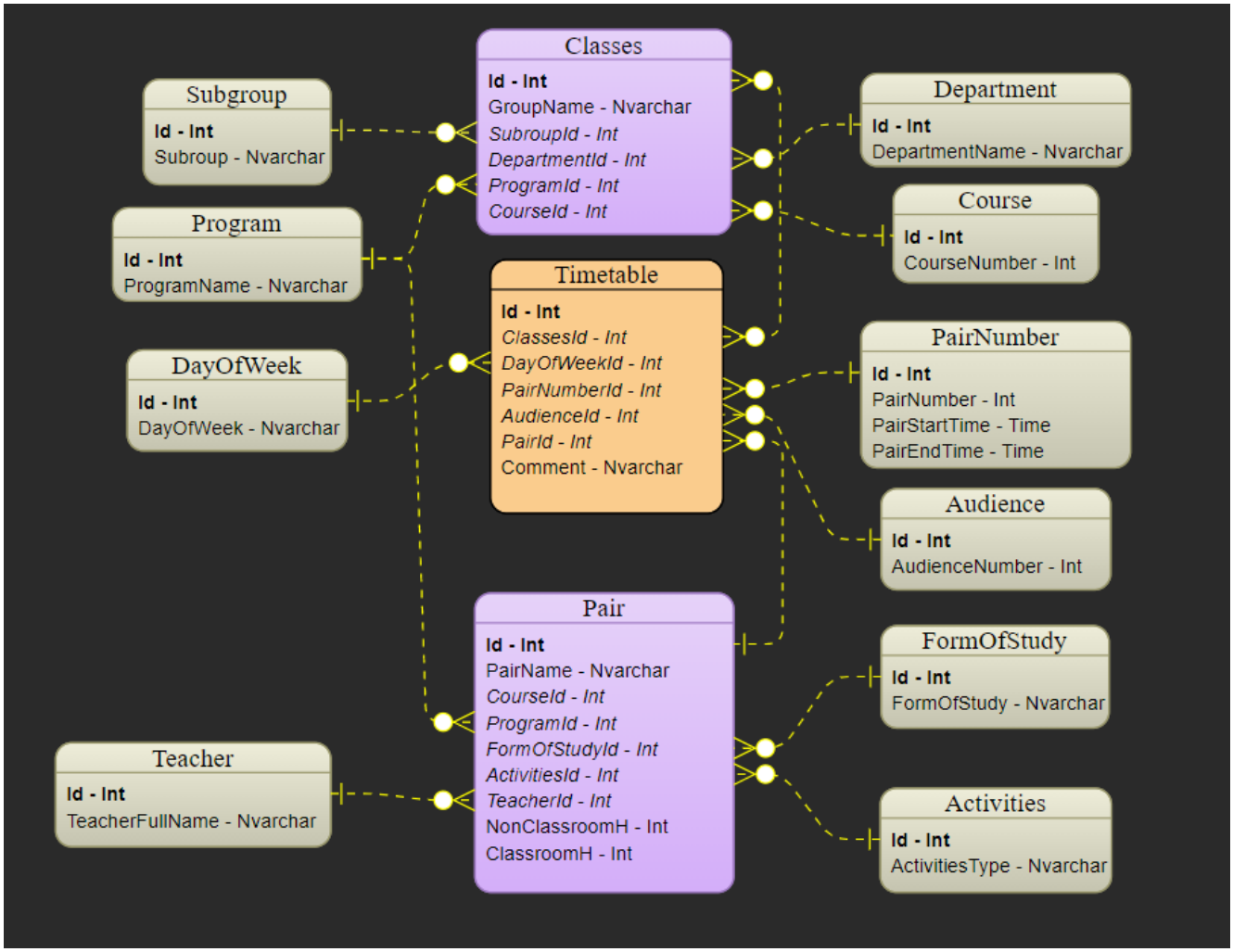


Рисунок 2.4 – Схема БД

Отже, схема бази даних цієї дипломної роботи буде складатися з 13 таблиць, які ми назвемо: Subgroup, Program, DayOfWeek, Teacher, Activities, FormOfStudy, Audience, PairNumber, Course, Department, Pair, TimeTable, Classes. Кожна схема буде включати Primary Key з назвою Id.

Також в цій схемі будуть Foreign Key. В таблиці Pair: CourseId, ProgramId, FormOfStudyId, ActivitiesId, TeacherId. В таблиці Classes: SubgroupId, DepartmentId, ProgramId, CourseId. В таблиці TimeTable: ClassesId, DayOfWeekId, PairNumberId, AudienceId, PairId.

Всі схеми між собою будуть зв’язані зв’язками один до багатьох.

Запропонований в цьому підрозділі підхід дозволяє впорядкувати множину інформаційних елементів за рівнями ієрархії. Це дозволяє виділити основну таблицю та таблиці, які займають проміжне положення.

2.1.2 Проектування схеми бази даних користувачів

Спроектуємо схему бази даних, яка буде зберігати користувачів. Також створимо розподілення на ролі. Таблиці будуть з’єднані зв’язком один до багатьох, оскільки багато користувачів можуть мати одну роль.

Отже, множина елементів схеми БД «користувачі» буде включати: нікнейм, справжнє ім’я, пошту, пароль та роль. Представимо нашу схему користувачів на рисунку 10.

Запропонована схема є дуже зручною, оскільки дає можливість легко масштабувати елементи користувачів.

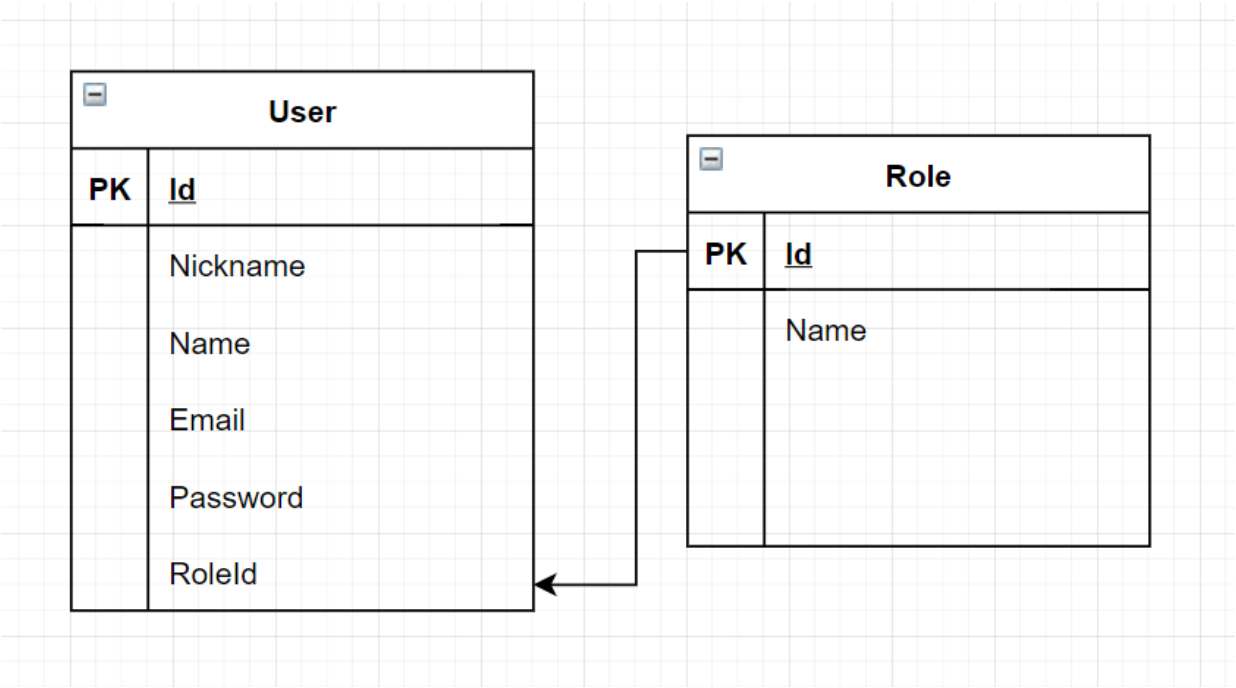
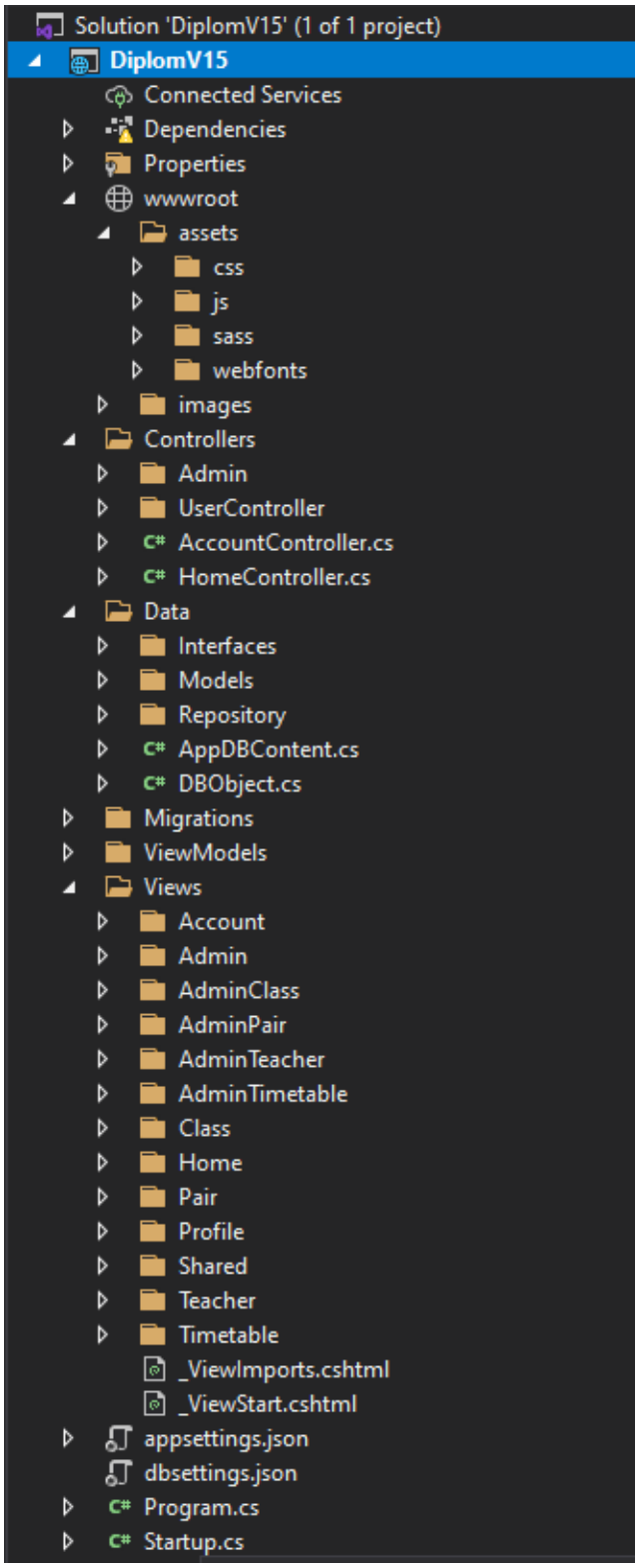


Рисунок 2.5 – Схема бази даних користувачів

2.2 Структура проекту

Зручна структура проекту э дуже важливою частиною розробки сайту на ASP.NET Core. Є два основних популярних підходи для структури проекту в структурі MVC. Перший називається Tech Folder, який встановлено в проект ASP.NET по замовченню. Він є дуже популярним для невеликих проектів, оскільки має логічну організацію. Всі моделі, інтерфейси, репозиторії та контролери розподілені кожен по своїй папці. Серед переваг цієї структури також є те, що вона дуже проста та зручна.

Також серед популярних підходів є Feature Folders, де для кожної моделі, контролера, інтерфейсу та репозиторію створюється своя папка. Це підвищує зв'язність файлів. Також це дає легко масштабувати проект, оскільки всі файли, зв'язані з одною моделлю, знаходяться в одній папці [12].



Після аналізу та порівняння цих двох підходів ми прийшли к висновку використовувати підхід Tech Folder.

Розглянемо на рисунку 2.6 структуру нашого проекту.

В папці wwwroot зберігаються всі статичні файли: це CSS, JS, файли шрифтів та зображення. Для кожного типу файлів створена своя папка.

Далі йде папка Controllers. Вона вміщує всі контролери. В свою чергу контролери розподілені на ті, представлення з яких бачить тільки адміністратор, тільки авторизований користувач та всі користувачі.

Далі знаходиться папка Data, в якій зберігаються всі інтерфейси, моделі та репозиторії. Для кожного пункту створена своя папка. Також тут знаходиться контекст нашої бази даних.

Нижче зберігаються всі міграції. В папці ViewModels знаходяться наші моделі представлень.

Останньою папкою є папка Views. В ній зберігаються всі представлення. Кожна папка називається так, як називається контролер, до якої вона відноситься. При цьому в папці Shared знаходиться наш Layout. Файл dbsettings.json та appsettings.json відповідають за настройки проекту та підключення до бази даних.

Рисунок 2.6 - Структура проекту

2.3 Entity Framework

Entity Framework (EF) – це рішення яке використовується в .NET для роботи з базами даних [13]. Воно дозволяє працювати з СУБД за допомогою сутностей. З допомогою EF робота з базою даних виконується значно швидше. Більша частина роботи, така як підключення, транзакції, запити та інше, виконується автоматично.

Кожну таблицю бази даних в Entity Framework прийнято виділяти в окрему модель. Моделі описують логіку даних. Для прикладу створимо модель, яка буде відповідати таблиці Classes в нашій базі даних на рисунку 2.7.

```
public class Classes
{
    public int Id { get; set; }

    public string groupName { get; set; }

    public int subgroupId { get; set; }

    public int departmentId { get; set; }

    public int programId { get; set; }

    public int courseId { get; set; }

    public bool isFavourite { get; set; }

    public List<Timetable> class_list { get; set; }

    public virtual Course course { get; set; }

    public virtual Subgroup subgroup { get; set; }

    public virtual Department department { get; set; }

    public virtual Programs program { get; set; }
}
```

Рисунок 2.7 – Модель таблиці Timetable

Для початку створимо первинний ключ для моделі. Це робиться за допомогою рядку:

public int Id { get; set; }

Для створення звичайного поля використаємо код:

public {тип даних} {назва змінної} { get; set; }

Додаємо всі зовнішні ключи. Вони отримуються за допомогою коду:

public int {назва таблиці}Id { get; set; }

Також для отримання зв’язку один до багатьох для кожного зовнішнього ключа створимо підключення до інших моделей. За допомогою цієї віртуальної змінної ми будемо отримувати доступ до даних інших моделей через зовнішні ключі. Це робиться за допомогою наступного коду:

public virtual {назва моделі} {назва змінної} { get; set; }

Цей код ми використовуємо в ситуаціях, коли ми підключаємо інші моделі до нашої. В ситуаціях, коли ми підключаємо нашу модель до іншої використаємо наступний код:

public List<{Таблиця, до якої підключаємося}> {назва змінної} { get; set; }

Далі створимо контекст бази даних. Для цього треба створити клас, який буде наслідувати клас DbContext – це клас, який визначає контекст даних для взаємодії з БД. За допомогою коду

public DbSet<{назва моделі}> {назва змінної} { get; set; }

представимо набір моделей, які будуть зберігатися в нашій базі даних.

Для підключення до нашої бази даних створимо json файл dbsettings. Додаємо в нього код ConnectionString:

"DefaultConnection":"Server=.; Database=Imperius; Trusted_Connection=True; MultipleActiveResultSets=true"

Оскільки ми використовуємо локальну БД, то назва сервера буде =“.”. Також додаємо рядок підключення в файлі Startup.cs на рисунку 2.8, який буде вказувати використовувати саме файл dbsettings.json для підключення до БД:

```
public Startup(IWebHostEnvironment hostEnv)
{
    _confString = new ConfigurationBuilder().SetBasePath(hostEnv.ContentRootPath).AddJsonFile("dbsettings.json").Build();
}
```

Рисунок 2.8 - Рядок підключення в файлі Startup.cs

Linq додає в Entity Framework синтаксис [14], подібний до SQL. Наприклад, ось так виглядає метод Where в Linq:

Var pair = db.pairs.Where(i => i.activity.Id.Equals(1));

Для прикладу наведемо код із даної роботи для додавання, видалення та отримання пари за Id в репозиторії TimetableRepository на рисунку 2.9.

```
public void AddPair(Timetable timetable)
{
    appDBContent.Timetable.Add(timetable);

    appDBContent.SaveChanges();
}

public Timetable GetPair(int id) => appDBContent.Timetable.FirstOrDefault(p => p.Id == id);

public void DeletePair(int id)
{
    Timetable timetable = GetPair(id);

    appDBContent.Timetable.Remove(timetable);

    appDBContent.SaveChanges();
}
```

Рисунок 2.9 – Додавання, отримання пари за Id та видалення пари за допомогою Linq

Entity Framework Fluent API використовується для настройки класів [15]. В ньому використовується клас OnModelCreating, за допомогою якого ми можемо настроїти такі речі, як:

1. Настроїти сутності: відношення «один до одного», «один до багатьох» та «багато до багатьох», первинні та зовнішні ключі, змінити назву таблиці.
2. Настроїти властивості: зіставлення стовпців та значення за замовчуванням.
3. Настроїти модель: додати атрибути анотації, настроїти схему за замовчуванням та багато іншого.

Для прикладу представимо код із нашого контексту бази даних для додавання значення за замовчуванням при створення моделей User та Role в нашому проекті на рисунку 2.10.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    string adminRoleName = "admin";
    string userRoleName = "user";

    string adminEmail = "admin@mail.ru";
    string adminPassword = "123456";

    Role adminRole = new Role { Id = 1, Name = adminRoleName };
    Role userRole = new Role { Id = 2, Name = userRoleName };
    User adminUser = new User { Id = 1, Email = adminEmail, Password = adminPassword, RoleId = adminRole.Id };

    modelBuilder.Entity<Role>().HasData(new Role[] { adminRole, userRole });
    modelBuilder.Entity<User>().HasData(new User[] { adminUser });
    base.OnModelCreating(modelBuilder);
}
```

Рисунок 2.10 – Значення за замовчуванням в моделях User та Role за допомогою Fluent API

В заключення скажемо, що Entity Framework дозволяє значно зменшити кількість коду при роботі з базами даних. Він представляє дуже велику кількість функціональних можливостей. Але, варто пам’ятати, що EF також збільшує навантаження на систему і зменшує продуктивність. В невеликих проектах, таких

як дана дипломна робота, це не сильно впливає, але в великих проектах рекомендовано використовувати ADO.NET [16].

2.4 Razor

Razor – це синтаксис розмітки для вбудовування серверного коду у веб-сторінки [17].

Серед переваг Razor можна виділити такі:

1. Razor дуже простий;
2. Вбудована підтримка в Visual Studio;
3. Код є більш стислим при порівнюванні с чистим HTML.

Розглянемо роботу розмітки Razor на прикладі коду із даного проекту на рисунку 2.11.

```
@model Pair

<div class="row mt-5 mb-2">
    <p>Назва: @Model.PairName</p>
    <p>Курс: @Model.course.CourseNumber.ToString()</p>
    <p>Група: @Model.programs.ProgramName</p>
    <p>Аудиторні години: @Model.ClassroomH</p>
    <p>Неаудиторні години: @Model.NonClassroomH</p>
    <p><a href="/AdminPair/Delete/@Model.Id">Видалити</a></p>
    <p><a href="/AdminPair/Edit/@Model.Id">Змінити пару</a></p>
</div>
```

Рисунок 2.11 – Приклад роботи розмітки Razor

Як можна побачити, синтаксис роботи Razor складається з коду C# та HTML розмітки. За допомогою @Model {Назва моделі} ми можемо підключити модель, а потім отримувати доступ до змінних цієї моделі. Про моделі буде розказано нижче.

3 РОЗРОБКА ВЕБ-ДОДАТКУ «ЕЛЕКТРОННИЙ РОЗКЛАД ФАКУЛЬТЕТУ»

3.1 Створення інтерфейсів та репозиторіїв. Валідація моделі.

Також для кожної моделі в даній роботі необхідно провести валідацію. Проведемо валідацію на прикладі моделі Classes в нашій БД рисунках 3.1 та 3.2.

```
[Key]
public int Id { get; set; }

[Display(Name = "Введіть назву")]
[StringLength(10)]
[Required(ErrorMessage = "Не вказана назва групи")]
public string groupName { get; set; }

[Display(Name = "Введіть Id підгрупи")]
[Range(1, 3, ErrorMessage = "Введіть Id від 1-го до 3-ьох")]
[Required(ErrorMessage = "Не вказана підгрупа")]
public int subgroupId { get; set; }
```

Рисунок 3.1 – Приклад валідації моделі БД

Спочатку додаємо атрибут Key для первинного ключа в нашій моделі. Він вказує, що поле Id являється первинним ключем. Атрибут Display додає строку, яка відображається як ім’я поля. Атрибут StringLength додає максимальну можливу довжину поля String. Також існує атрибут Range(x, y), за допомогою якого можна додати діапазон значень від x до y. Одним з найважливіших атрибутів є атрибут Required, який робить будь-яке поле обов’язковим. Також можна вказати ErrorMessage та MinimumLength – це іменовані параметри. ErrorMessage виводить повідомлення при порушенні атрибута, а MinimumLength задає мінімальну довжину поля [18].

```
[BindNever]
[ScaffoldColumn(false)]
public FormOfStudy formOfStudy { get; set; }
```

Рисунок 3.2 – Приклад валідації зв’язку між моделями

Також дуже важливо додати атрибут ScaffoldColumn(false) до полів, які додають підключення до інших моделей. Цей атрибут повністю приховує ці поля.

Наступним етапом буде створення інтерфейсів для кожної моделі. Розберемо інтерфейси на прикладі інтерфейсу для моделі Timetable нашої БД на рисунку 3.3.

```
public interface ITimetable
{
    IEnumerable<Timetable> timetables { get; }

    void AddPair(Timetable timetable);

    Timetable GetPair(int id);

    void DeletePair(int id);

    void EditPair(Timetable timetable);
}
```

Рисунок 3.3 – Інтерфейс ITimetable

Спочатку створимо перелік IEnumerable. IEnumerable – це перелік, який підтримує просту ітерацію по колекції. Він буде повертати список із всіх даних з таблиці Timetable в нашій базі даних. Також додамо 4 функції, повернення пари по id, видалення пари, зміна існуючої пари та додавання нової пари в розклад. Оскільки це інтерфейс, то не будемо реалізовувати ці функцій.

Для реалізації наших інтерфейсів для кожного інтерфейсу створимо репозиторій. Розглянемо реалізацію інтерфейсу на прикладі репозиторію TimetableRepository на рисунку 3.4.

```
private readonly AppDBContent appDBContent;

public TimetableRepository(AppDBContent appDBContent)
{
    this.appDBContent = appDBContent;
}

public IEnumerable<Timetable> timetables => appDBContent.Timetable.Include(c => c.classes).Include(c => c.dayWeek)
    .Include(c => c.parity).Include(c => c.pairNumber).Include(c => c.audience).Include(c => c.pair);

public void AddPair(Timetable timetable)
{
    appDBContent.Timetable.Add(timetable);

    appDBContent.SaveChanges();
}

public Timetable GetPair(int id) => appDBContent.Timetable.FirstOrDefault(p => p.Id == id);

public void DeletePair(int id)
{
    Timetable timetable = GetPair(id);

    appDBContent.Timetable.Remove(timetable);

    appDBContent.SaveChanges();
}

public void EditPair(Timetable timetable)
{
    appDBContent.Timetable.Update(timetable);

    appDBContent.SaveChanges();
}
```

Рисунок 3.4 – Репозиторій моделі Timetable

Він буде наслідувати інтерфейс ITimetable. Спочатку ми створюємо змінну типу контексту нашої бази даних та конструктор за замовчуванням для нашого репозиторію.

Напишемо реалізацію списку IEnumerable. Цей код буде повертати всі дані з таблиці в БД. За допомогою ключового слова Include та змінних в нашій моделі ми підключаємо інші таблиці до цієї. Також напишемо реалізацію функцій AddPair, GetPair, DeletePair та EditPair.

3.2 Створення контролерів та маршрутів для них

Контролер – це центральний компонент в архітектурі MVC. В ньому зберігаються спеціальні функції (такі, як ViewResult, ActionResult та інші), які при виклику будуть повертати представлення [19].

Розглянемо на рисунках 3.5 – 3.7 створення контролера на прикладі контролера AdminPairController, який на сайті відповідає за відображення та зміну інформації про пару адміністратором.

```
public class AdminPairController : Controller
{
    private readonly IPair m_IPair;

    public AdminPairController(IPair _iPair)
    {
        m_IPair = _iPair;
    }
```

Рисунок 3.5 – Створення контролера PairController

Для початку підключимо наслідування базового класу Controller з класу AspNetCore.MVC, і створимо змінну типу інтерфейсу пари. Також напишемо конструктор по замовчуванню.

```
[Authorize(Roles = "admin")]
[Route("AdminPair/List")]
[Route("AdminPair/List/{pairInfo}")]
public ViewResult List(string pairInfo)
{
    string _program = pairInfo;
    IEnumerable<Pair> pair = null;

    string currActivities = "";
    if (string.IsNullOrEmpty(pairInfo))
    {
        pair = m_IPair.pairs.OrderBy(i => i.PairName);
        currActivities = "Всі види";
    }
    if (string.Equals("lecture", pairInfo, StringComparison.OrdinalIgnoreCase))
    {
        pair = m_IPair.pairs.Where(i => i.activity.Id.Equals(1)).OrderBy(i => i.Id);
        currActivities = "Лекції";
    }
    if (string.Equals("practice", pairInfo, StringComparison.OrdinalIgnoreCase))
    {
        pair = m_IPair.pairs.Where(i => i.activity.Id.Equals(2)).OrderBy(i => i.Id);
        currActivities = "Практики";
    }
```

Рисунок 3.6 – Створення функції List в контролері PairController

За допомогою коду *[Authorize(Roles = "admin")]* відмітимо, що сторінкою у веб-додатку, яке повертає дана функція можуть користуватися тільки адміністратори. Про ролі буде розказано нижче.

Далі ми додаємо маршрут для нашого контролера за допомогою коду *[Route(“Pair/List/{pairInfo}”)],* де pairInfo – змінна, яка буде відображати тип активності (лекція, практика, лабораторна). Створимо перелік IEnumerable, який і буде списком пар, які будуть виводитися на сайті. Блок коду If відповідає саме за те, які з пар будуть відображатися. Якщо рядок pairInfo пустий, то будуть відображатися всі пари. Якщо ж pairInfo буде дорівнювати наприклад lecture – то будуть відображатися тільки пари, які мають Id, який дорівнює виду активності – «Лекція». Код StringComparsion.OriginalIgnoreCase відповідає за те, щоб не

враховувався регістр букв. currActivities – це змінна типу string, яка буде на сайті відображати тип активності, пари якої зараз відображаються.

```
    var classObj = new ClassListViewModel
    {
        AllInfo = pair,
        currActivities = currActivities
    };

    ViewBag.Title = "Сторінка с групами";

    return View(classObj);
}
```

Рисунок 3.7 – Блок коду з поверненням представлення в PairController

Для повернення значення у View, створимо об’єкт classObj, який буде об’єднувати всі пари, які необхідно вивести, а також рядок currActivities. І потім напишемо код, який повертає classObj у View. Далі розглянемо код у контролері, який буде відповідати за зміну пари на рисунку 3.8.

```
[Authorize(Roles = "admin")]
[HttpGet]
[ActionName("Edit")]
public ViewResult Edit(int id)
{
    Pair pair = m_IPair.GetPair(id);

    return View(pair);
}

[Authorize(Roles = "admin")]
[HttpPost]
public IActionResult Edit(Pair pair)
{
    if (ModelState.IsValid)
    {
        m_IPair.EditPair(pair);

        return Redirect("/AdminPair/List");
    }
    else
    {
        return View();
    }
```

Рисунок 3.8 – функція Edit контролера AdminPairController

Функція складається з 2-х частин. Перша – це HttpGet частина, яка повертає представлення на пару. А друга, HttpPost частина, виконується після натискання кнопки «Зберегти» на сайті та змінює пару у нашій базі даних [20].

Також для кожного контролера створимо маршрути в коді класу Startup.cs:

routes.MapRoute(name: "{Назва змінної}", template: "{назва контролера}/{action}", defaults: new { Controller = "{Назва контролера}", action = "{Назва View}" });

У цьому розділі ми написали код, який відповідає за відображення та маршрутизацію контенту на нашому сайті.

3.3 Створення представлень.

Представлення – це код користувацького інтерфейсу, оснований на HTML. Розглянемо створення представлень на прикладі представлення list для контролера PairController.

List:

```
<h3><p align="center">@Model.currActivities</p></h3>
<div class="col-lg">
   @{
      foreach (Pair pair in Model.AllInfo)
      {
         @Html.Partial("AdminPairList", pair)
      }
   }
</div>
```

Можна побачити, що це представлення повертає код файлу AdminPairList, який ми розглядали в дургому розділі на рисунку 16. Також створимо файл ViewImports з кодом:

```
@{
   Layout = "_Layout";
}
```

, який буде відповідати за підключення до нашого Layout, а також _ViewImport, в якому будуть зберігатися імпорти в наші представлення.

Представимо результат роботи представлення на нашому сайті на рисунку 25.

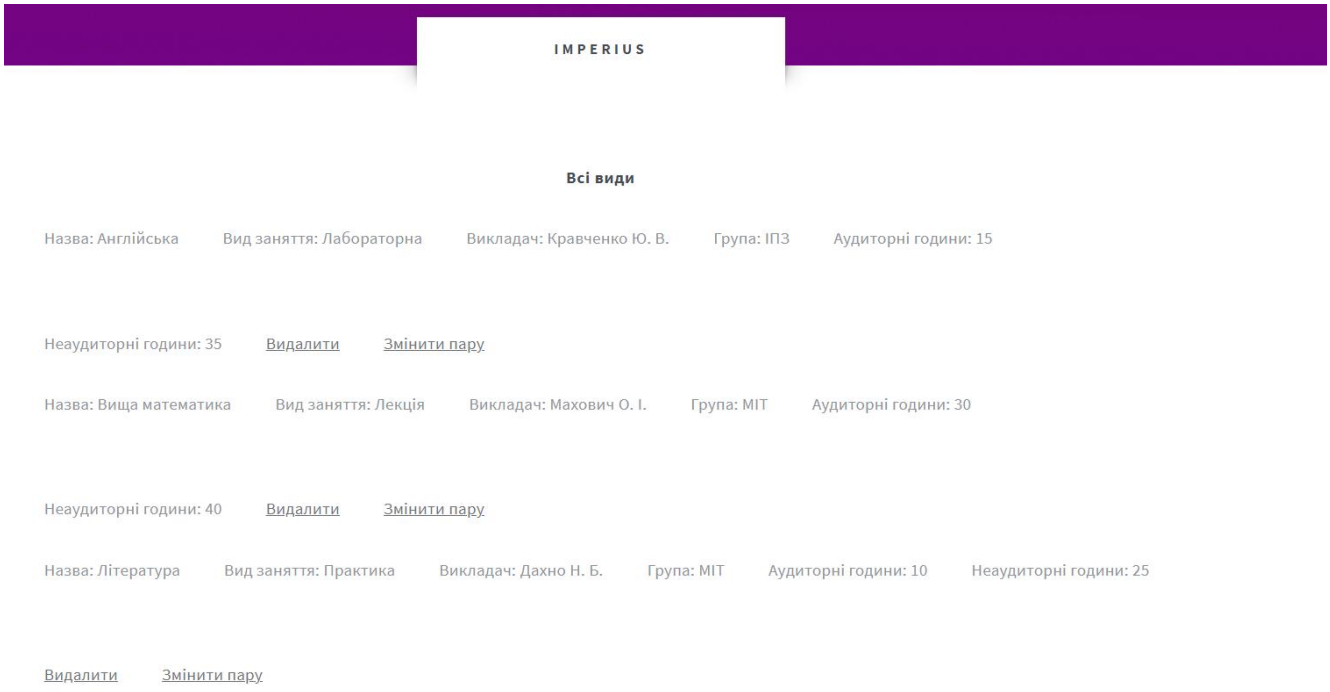


Рисунок 6 – Результат представлення List у контролері AdminPairController

Також представимо представлення Edit, яке відповідає на зміну інформації у контролері AdminTeacherController на рисунку 3.9 та результат його роботи на сайті на рисунку 3.10.

```
@model Teacher

<form asp-action="Edit" asp-controller="AdminTeacher" asp-route-id="@Model.Id">
    <div class="form-group">
        <label asp-for="FullName" class="control-label">Имя</label>
        <input type="text" asp-for="FullName" class="form-control" />
    </div>
    <div class="form-group">
        <label asp-for="img" class="control-label">Фото</label>
        <input type="text" asp-for="img" class="form-control" />
    </div>
    <div class="form-group">
        <input type="submit" value="Зберегти" class="btn btn-default" />
    </div>
</form>
```

Рисунок 3. 9 – Представлення Edit контролера AdminTeacherController

Рисунок 3.10 – Результат роботи представлення на сайті

3.4 Авторизація та аутентифікація. Розподілення на ролі

Ця дипломна робота включає 2 ролі. Користувач та адмін. Користувач зможе переглядати інформацію на сайті, а адмін зможе її редагувати. Незалогіненний користувач зможе лише зайти на сайт та побачити початкову сторінку.

Створимо моделі для реєстрації та авторизації. Модель для реєстрації буде майже повторювати модель User, але матиме валідацію для коректного створення нового користувача.

За аутентифікацію та авторизацію на нашому сайті відповідає AccountController. Цей контролер включає в себе, реєстрацію, авторизацію, аутентифікацію та вихід з аккаунту. Розповім про роботу AccountController`а на прикладі реєстрації на рисунку 3.11.

```
[HttpGet]
public IActionResult Register()
{
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Register(RegisterModel model)
{
    if (ModelState.IsValid)
    {
        User user = iUser.GetRegisterUser(model);

        if (user == null)
        {
            user = new User { Email = model.Email, Password = model.Password, NickName = model.NickName, Name = model.Name };
            Role userRole = await dBContent.Roles.FirstOrDefaultAsync(r => r.Name == "user");
            if (userRole != null)
                user.Role = userRole;

            dBContent.Users.Add(user);

            await dBContent.SaveChangesAsync();

            await Authenticate(user);

            return RedirectToAction("Index", "Home");
        }
        else
            ModelState.AddModelError("", "Некоректний логін та(або) пароль");
    }
    return View(model);
}
```

Рисунок 3.11 – Реєстрація в контролері AccountrController

Спочатку Get-запит повертає представлення на форму реєстрації. А Post-запит за домогою *user = new User {}* створює нового користувача, додає його до бази даних та зберігає її. Фільтр *[ValidateAntiforgeryToken]* призначений для протидії підробці міжсайтових запросів. Потім за допомогою *await Autheticate(user)* мы проводимо аутентифікацію цього користувача.

Представимо принцип аутентифікації нашої дипломної роботи в контролері AccountController на рисунку 3.12.

```
private async Task Authenticate(User user)
{
    var claims = new List<Claim>
    {
        new Claim(ClaimsIdentity.DefaultNameClaimType, user.Email),
        new Claim(ClaimsIdentity.DefaultRoleClaimType, user.Role?.Name)
    };

    ClaimsIdentity id = new ClaimsIdentity(claims, "ApplicationCookie", ClaimsIdentity.DefaultNameClaimType,
        ClaimsIdentity.DefaultRoleClaimType);

    await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(id));
}
```

Рисунок 3.12 – Реалізація аутентифікації

Також для авторизації та аутентифікація в нашому проекті додамо код в клас Startup.cs в метод Configure:

app.UseAuthentication();
app.UseAuthorization();

ВИСНОВКИ

1. У першому розділі була проаналізована методична література по розробці інформаційної системи. Спочатку було проаналізовано стандарт SQL, його архітектуру та основи. Далі були проаналізовані всі найпопулярніші системи управління базами даних на ринку та порівняні між собою, виведені головні переваги та недоліки кожної та вибрана одна, яка підходитиме для даної дипломної роботи найкраще. Далі була проаналізована документація по головним технологіям, які ми використовували в даній дипломній роботи. Серед головних: Transact-SQL, ASP-NET, Razor та інші. Робота, яка була пророблена в даному розділі допоможе нам в проектуванні схеми бази даних для даної дипломної роботи і написанні веб-додатку.
2. У наступному розділі була проаналізована предметна область «Електронного розкладу факультету». Була виділена множина задач обробки даних та множина інформаційних елементів бази даних. Далі були застосовані здобуті знання з першого розділу та інформація з другого для проектування схеми бази даних, на якій в подальшому і буде працювати наш веб-додаток. Наступним кроком було проаналізовано та вибрано структуру нашого проекту. Структура є дуже важливою частиною написання коду, оскільки вона допоможе в подальшому не плутатися в нашому власному коду та може підвищити швидкість розробки проекту. Далі була проаналізована технологія Entity Framework. Ця технологія допоможе зменшити кількість коду та підвищити швидкість та простоту написання коду та підключення до бази даних.
3. В останньому розділі на основі набутих знань та спроектованої бази даних ми розробили веб-додаток «Електронний розклад факультету». Спочатку

були створені моделі, а також інтерфейси та репозиторії до кожної моделі, проведена валідація моделі. Потім був створений контекст бази даних. Наступними двома кроками були створені контролери та представлення. Далі ми створили базу даних користувачів та розподілення на ролі. Були створені авторизація, аутентифікація та реєстрація.

4. Робота, проведена в цих трьох розділах дозволила нам спроектувати та створити ефективну базу даних, та на основі неї створити робочий прототип веб-додатку «Електронний розклад факультету». Спроектовану в цій роботі схему бази даних можна використовувати як основу для розробки власного веб-додатку за темою розклад факультету або будь-якого іншого закладу освіти. Робочий прототип веб-сайту також можна використовувати для розробки сайту розкладу навчального закладу

ПЕРЕЛІК ПОСИЛАНЬ

1. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель, SQL Полное руководство Третье издание – 959 с.
2. [Електронний ресурс]. – Режим доступу: https://coderlessons.com/tutorials/bazy-dannykh/uchit-sql/sql-obzor
3. [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85
4. [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85
5. Офіційний сайт Oracle Database [Електронний ресурс]. – Режим доступу: https://www.oracle.com/ru/index.html
6. Офіційний сайт PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/
7. Офіційний сайт MySQL [Електронний ресурс]. – Режим доступу: https://www.mysql.com/
8. Офіційний сайт MS SQL Server[Електронний ресурс]. – Режим доступу: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019?ranMID=24542&ranEAID=a1LgFw09t88&ranSiteID=a1LgFw09t88-HpXfoeALYk1aVD1J3DuwoA&epi=a1LgFw09t88-HpXfoeALYk1aVD1J3DuwoA&irgwc=1&OCID=AID2000142_aff_7593_1243925&tduid=%28ir__16rjrcdbqwkfq2k1kk0sohz3
9. Офіційна документація Transact-SQL [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/sql/t-sql/language-reference?ranMID=24542&ranEAID=a1LgFw09t88&ranSiteID=a1LgFw09t88-h9wqH4u9Q_FEMjqEQ0hOeQ&epi=a1LgFw09t88-h9wqH4u9Q_FEMjqEQ0hOeQ&irgwc=1&OCID=AID2000142_aff_7593_1243925&tduid=(ir__16rjrcdbqwkfq2k1kk0sohz3wf2xugilivc2mf2b00)(7593)(1243925)(a1LgFw09t88-h9wqH4u9Q_FEMjqEQ0hOeQ)()&irclickid=_16rjrcdbqwkfq2k1kk0sohz3wf2xugilivc2mf2b00&view=sql-server-ver15
10. Офіційна документація ASP.NET Core [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/?ranMID=24542&ranEAID=a1LgFw09t88&ranSiteID=a1LgFw09t88-MVk0aIKJ4VxQI7m8miEXwQ&epi=a1LgFw09t88-

MVk0aIKJ4VxQI7m8miEXwQ&irgwc=1&OCID=AID2000142_aff_7593_1243925&tduid=(ir__16rjrcdbqwkfq2k1kk0sohz3wf2xugilguc2mf2b00)(7593)(1243925)(a1LgFw09t88-MVk0aIKJ4VxQI7m8miEXwQ)()&irclickid=_16rjrcdbqwkfq2k1kk0sohz3wf2xugilguc2mf2b00&view=aspnetcore-5.0

11. [Електронний ресурс]. – Режим доступу: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/
12. [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/321392/
13. Офіційна документація Entity Framework [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/ef/?ranMID=24542&ranEAID=a1LgFw09t88&ranSiteID=a1LgFw09t88-K07AtVgvOTbdvFphGO61qA&epi=a1LgFw09t88-K07AtVgvOTbdvFphGO61qA&irgwc=1&OCID=AID2000142_aff_7593_1243925&tduid=%28ir__16rjrcdbqwkfq2k1kk0sohz3wf2xuginjec2mf2b00%29%287593%29%281243925%29%28a1LgFw09t88-K07AtVgvOTbdvFphGO61qA%29%28%29&irclickid=_16rjrcdbqwkfq2k1kk0sohz3wf2xuginjec2mf2b00
14. [Електронний ресурс]. – Режим доступу: https://www.entityframeworktutorial.net/querying-entity-graph-in-entity-framework.aspx
15. [Електронний ресурс]. – Режим доступу: https://www.entityframeworktutorial.net/efcore/fluent-api-in-entity-framework-core.aspx
16. [Електронний ресурс]. – Режим доступу: https://skillbox.ru/media/code/entity_framework/
17. [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-5.0
18. [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/aspnet5/19.1.php
19. [Електронний ресурс]. – Режим доступу: https://itproger.com/course/asp-net
20. [Електронний ресурс]. – Режим доступу: https://professorweb.ru/my/ASP_NET/gamestore/level2/2_10.php

Додаток А. Фото сторінок сайту

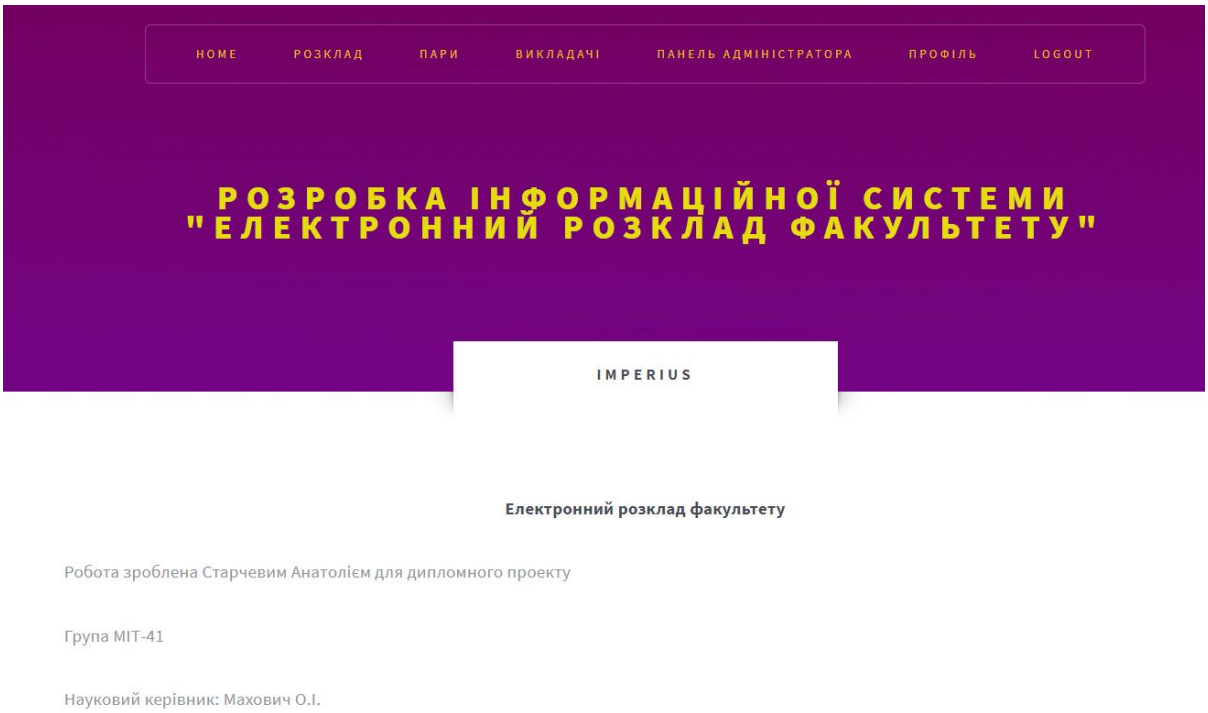


Рисунок 4.1 - Головна сторінка сайту

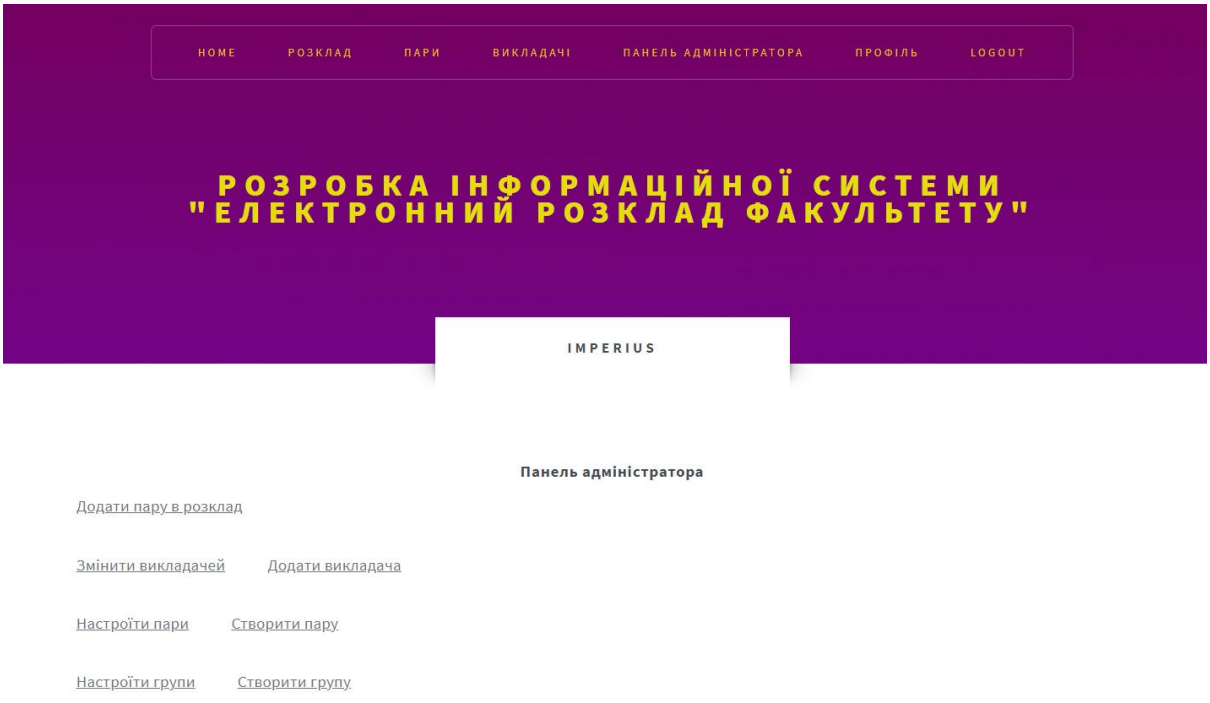


Рисунок 4.2 - Панель адміністратора

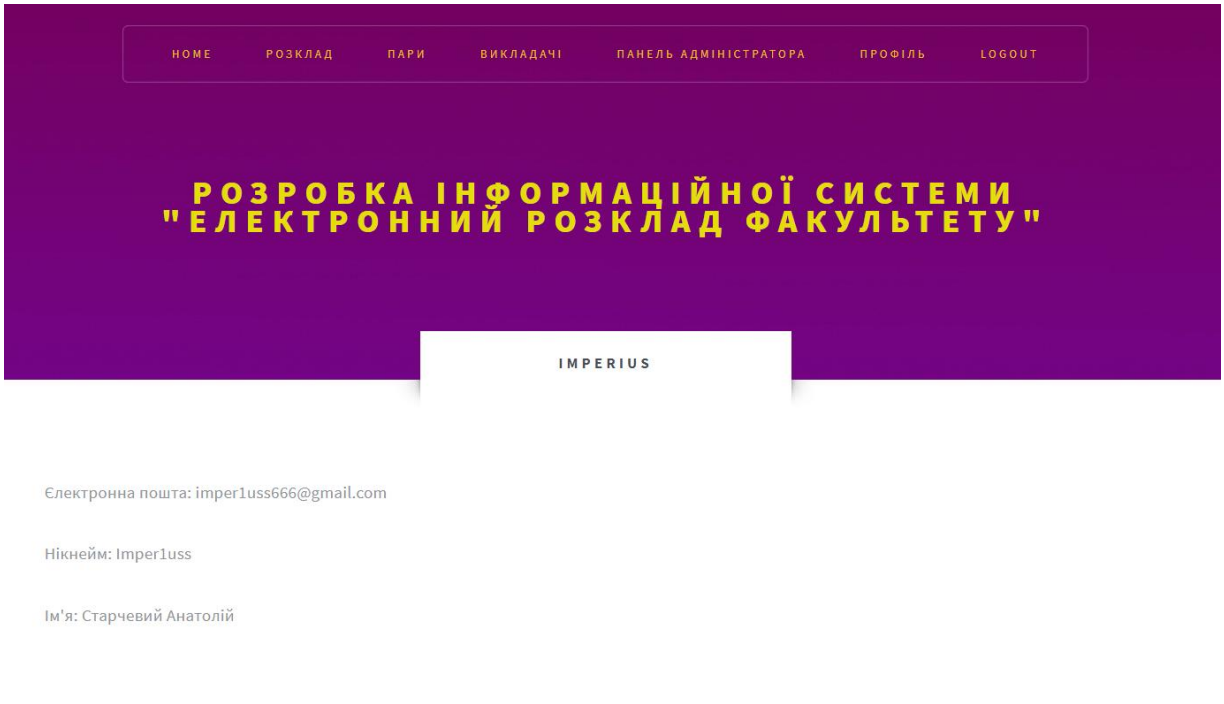


Рисунок 4.3 – Профіль

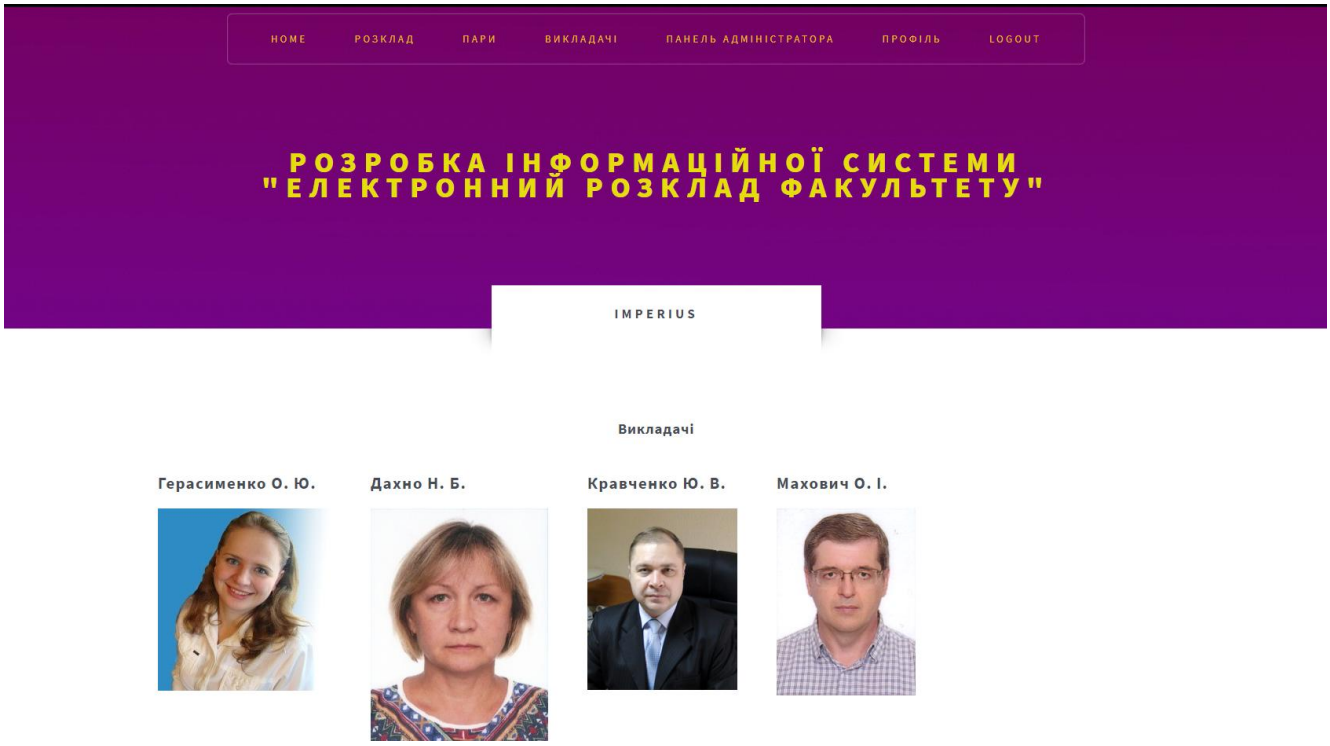


Рисунок 4.4 - Список викладачів

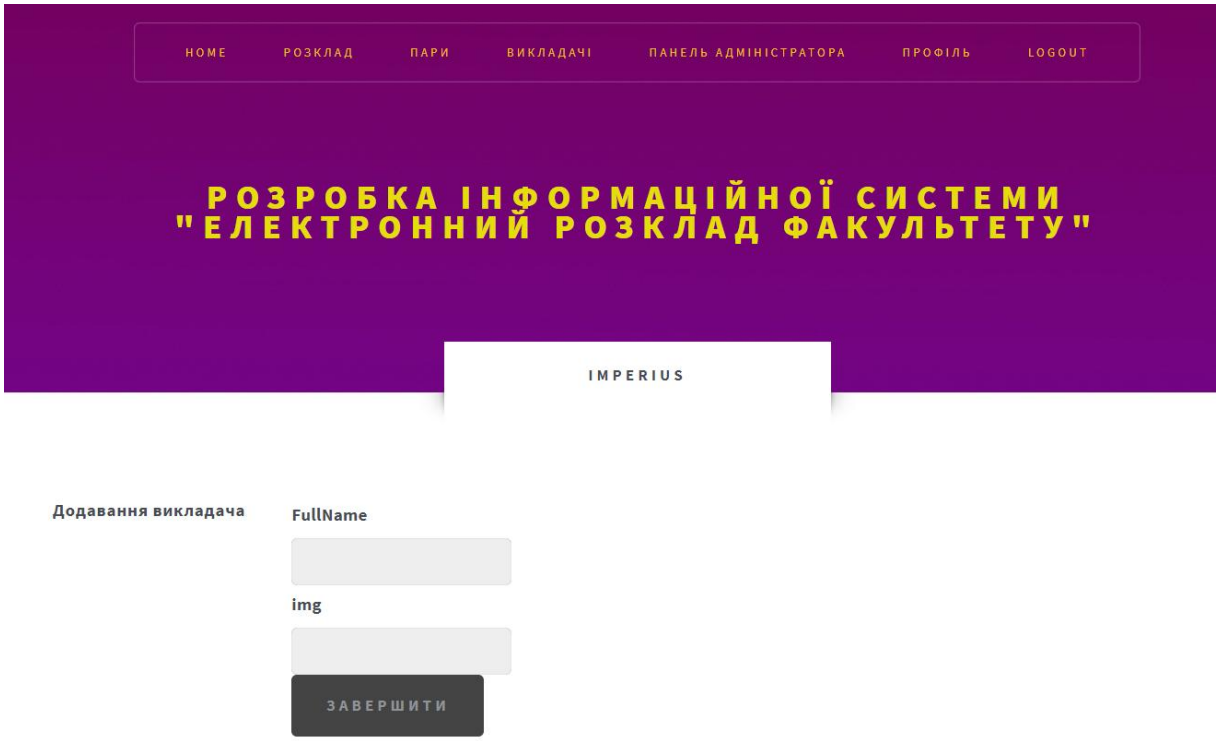


Рисунок 4.4 - Додавання викладача

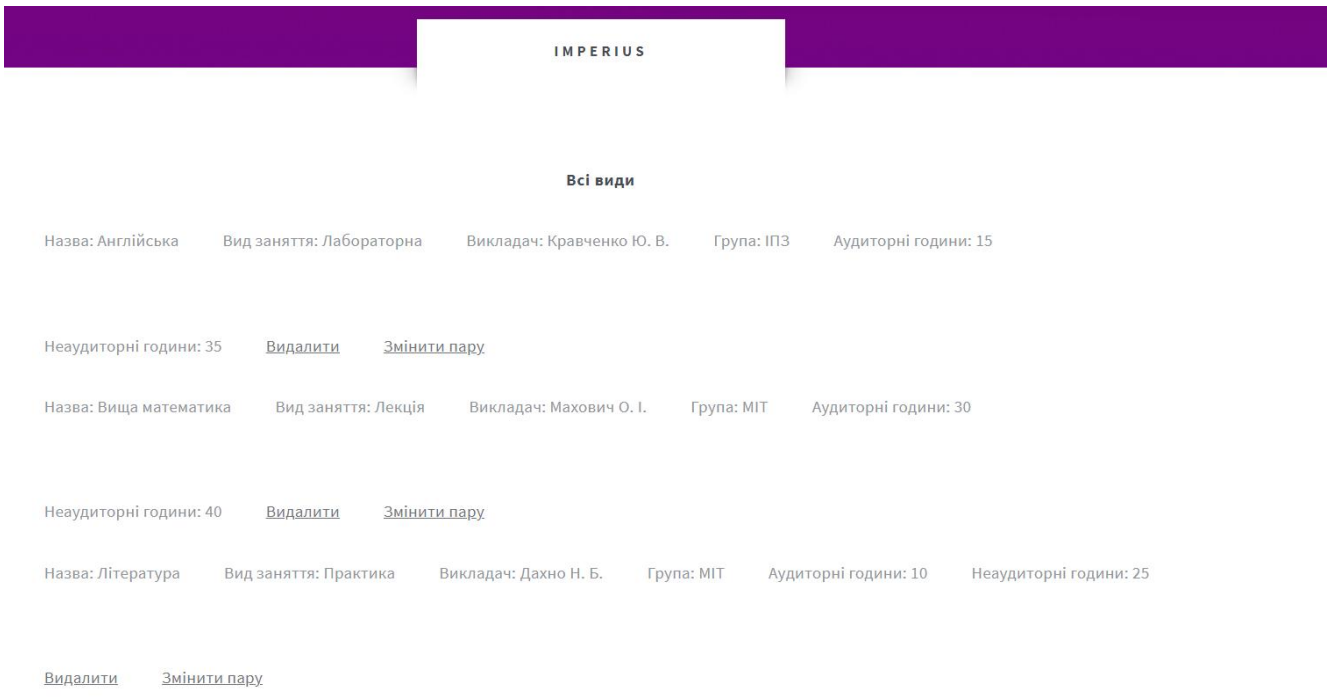


Рисунок 4.5 - Інформація про пари

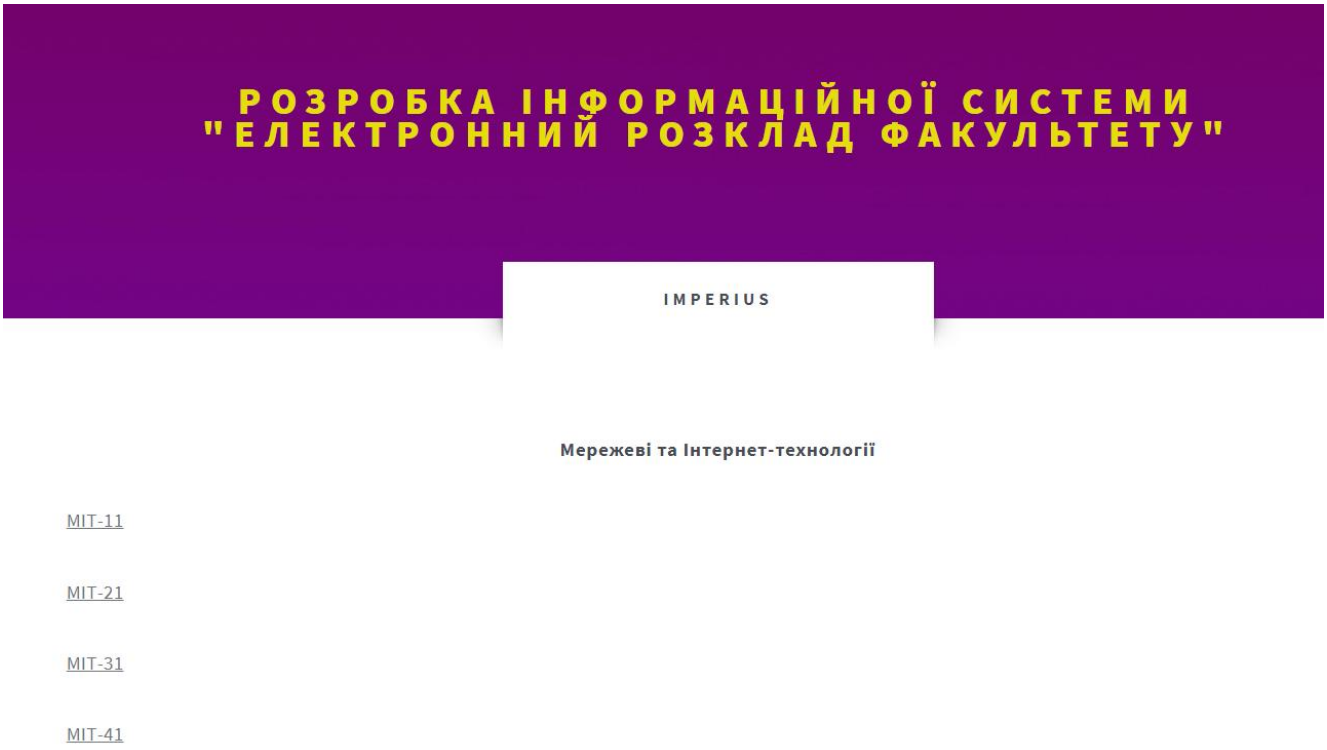


Рисунок 4.6 - Список груп програми "МІТ"

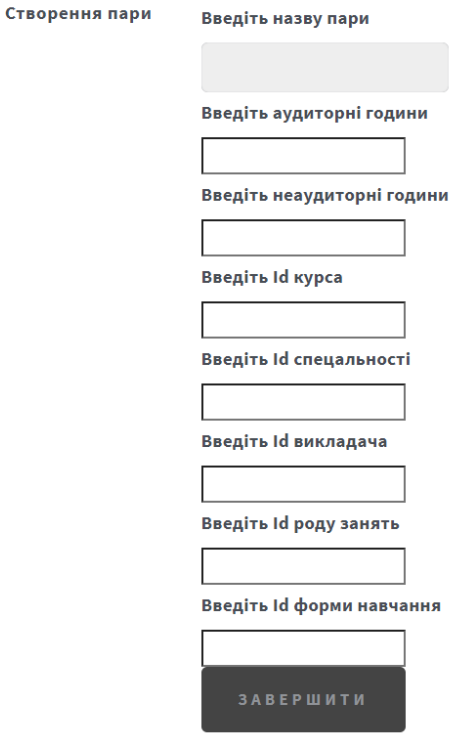


Рисунок 4.7 - Створення пари

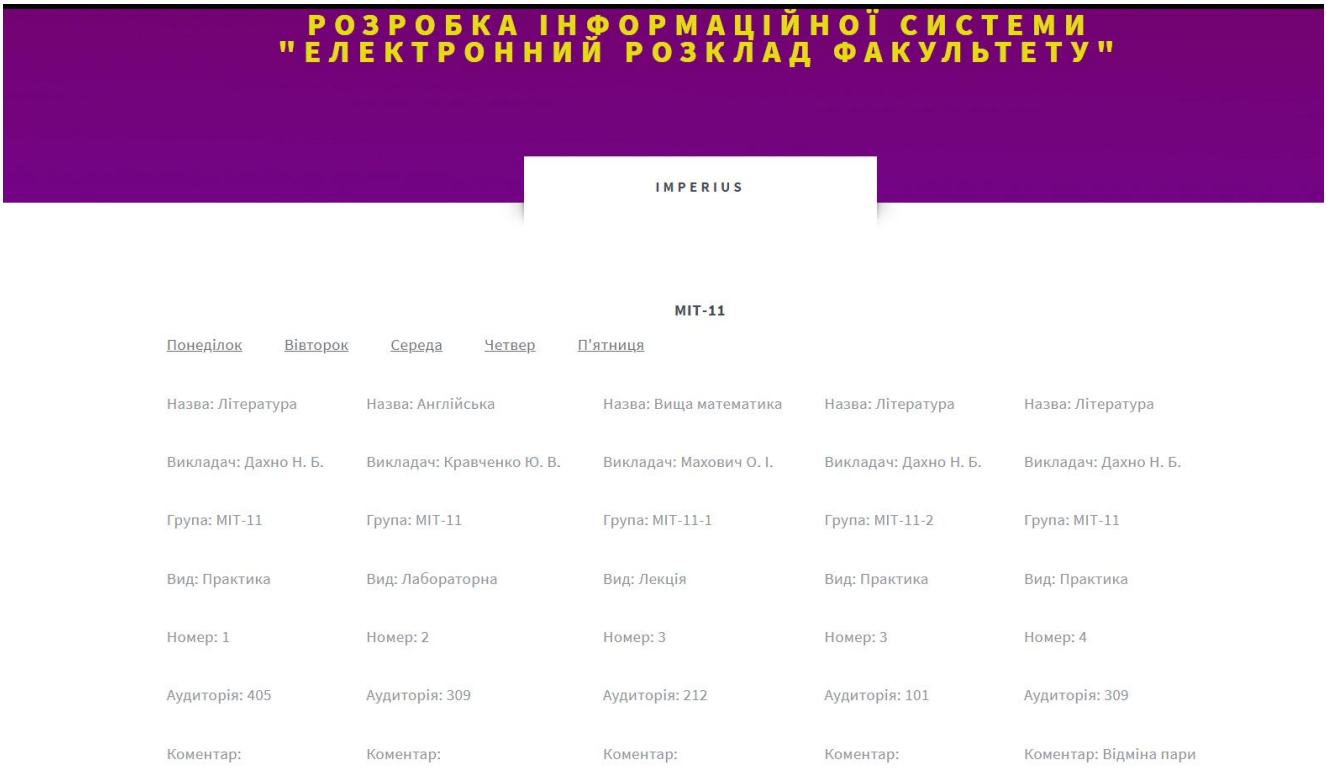


Рисунок 4.8 - Розклад для групи МІТ-11 у вівторок

Рисунок 4.9 - Додавання групи

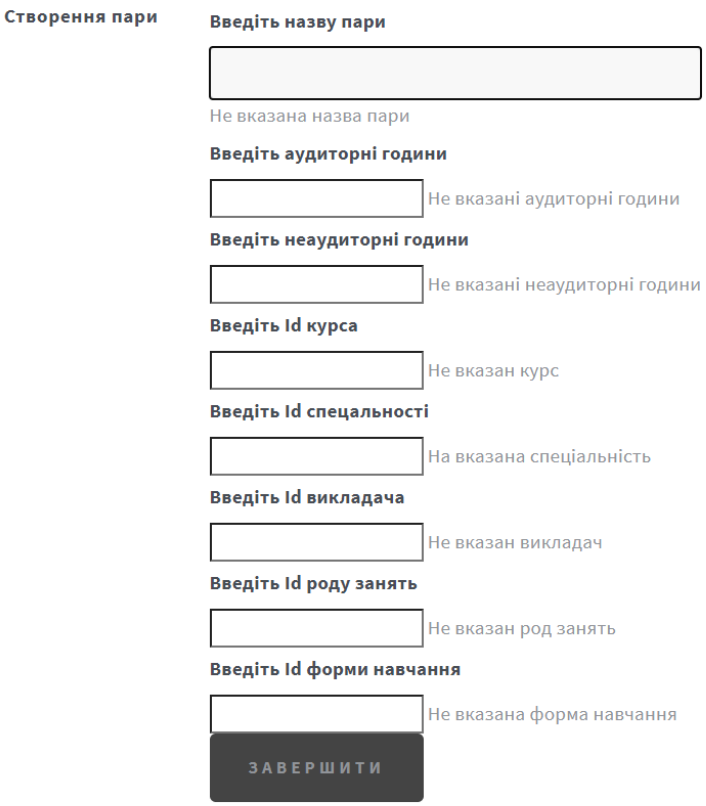


Рисунок 4.10 - Результат роботи валідації