НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“До захисту допущено”
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

“___”_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Цифрові технології в енергетиці”
зі спеціальності *122 “Комп’ютерні науки”*

на тему: Інформаційний портал для студентів кафедри

Виконав:
студент IV курсу, групи ТР-01

БЕКІРОВ Владислав Васильович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник: *доцент каф. цифрових технологій в енергетиці*
доцент, к.т.н., Юлія СИДОРЕНКО _____
(посада, вчене звання ,науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент: доцент каф. ТАЕ, к.т.н., Рачинський Артур Юрійович
_____ _____
(посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Н.контроль: *старший викладач Ольга Беспала* _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА_

«__»_____2024 р.

ЗАВДАННЯ
на дипломну роботу студенту

БЕКІРОВУ Владиславу Васильовичу
(прізвище, ім’я, по батькові)

1. Тема роботи **Інформаційний портал для студентів кафедри**

керівник роботи ***Сидоренко Юлія Всеволодівна, к.т.н., доцент***
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__»_____ 202__ р. №_____

2. Термін подання роботи студентом **07.06.2024 року**

3. Вихідні дані до роботи мова програмування C#, мова програмування TypeScript, фреймворки ASP.NET та Angular, СКБД PostgreSQL, середовище розробки Rider.

4. Перелік питань, які потрібно розробити _____
1) провести аналіз існуючих інформаційних порталів вищих навчальних закладів_____
2) аргументувати обрані технології та алгоритми для реалізації програмного забезпечення_____
3) розробити архітектуру бази даних та сервера_____
4) створити веб-застосунок_____

5) продемонструвати роботу користувача з розробленою програмною системою

5. Орієнтований перелік ілюстративного матеріалу діаграми, що описують архітектуру системи, бази даних та серверу, способи взаємодії користувача з системою.

6. Дата видачі завдання «15» вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі	17-20.04.24	Виконано
3.	Розробка архітектури та загальної структури системи	21-25.04.24	Виконано
4.	Розробка частин окремих підсистем	25.04-02.05.24	Виконано
5.	Програмна реалізація системи	03-07.05.24	Виконано
6.	Оформлення пояснювальної записки	08-12.05.24	Виконано
7.	Захист програмного продукту	13-17.05.24	Виконано
8.	Передзахист	03-06.06.24	Виконано
9.	Подання готової роботи на кафедру	07.06.2024	Виконано
10.	Захист	17-21.06.2024	Виконано

Студент _____ **Владислав БЕКІРОВ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ **Юлія СИДОРЕНКО**
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота виконана на 53 сторінках, містить 38 ілюстрацій, 10 таблиць, 1 додаток, 15 джерел в переліку посилань.

Метою даної роботи є створення інформаційного веб-порталу для студентів кафедри, що дозволить оптимізувати освітній процес.

Для розробки було використано сучасні технології, а саме: СКДБ PostgreSQL, мови програмування C# та JavaScript, фреймворки ASP.NET та Angular та бібліотеку компонент користувацького інтерфейсу Angular Material.

У результаті було отримано веб-портал з розділом новин та оголошень, календарем подій, бібліотекою навчально-методичних матеріалів.

Ключові слова**:** Інформаційний портал, ASP.NET Core, Angular, Entity Framework Core, PostgreSQL, TailwindCSS, Seq.

ABSTRACT

The thesis consists of 53 pages, 38 illustrations, 10 tables, 1 appendix, 15 sources in the list of references.

The purpose of this work is to create an information web portal for students of the department, which will optimise the educational process.

Modern technologies were used for the development, namely: PostgreSQL database, C# and JavaScript programming languages, ASP.NET and Angular frameworks, and the Angular Material user interface component library.

The result is a web portal with a news and announcements section, an events calendar, and a library of educational materials.

Keywords: Information portal, ASP.NET Core, Angular, Entity Framework Core, PostgreSQL, TailwindCSS, Seq.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Фреймворк – набір інструментів та бібліотек для розробки програмного забезпечення.

СКБД – система керування базами даних.

ORM – об'єктно-реляційне відображення. Технологія, що спрощує роботу з базами даних, дозволяючи маніпулювати даними у вигляді об'єктів.

UI – інтерфейс користувача.

Backend – сервера частина програмного забезпечення.

API – програмний інтерфейс програми.

Middleware – проміжне програмне забезпечення, що забезпечує взаємодію між різними компонентами системи.

ВСТУП

Сучасний освітній процес швидко розвивається все більше інтегруючи інформаційно-комунікаційні технології. Разом з цим з’являються і нові виклики. Одним із ключових викликів є розрізненість інформаційних ресурсів. Інформація, що стосується навчального процесу, часто розподілена по різних джерелах: паперових документах, онлайн платформах та веб-сайтах інституту, що ускладнює процес пошуку потрібної студентам та викладачам інформації. Через це студенти змушені витрачати час на пошук матеріалів замість того, щоб зосередитись на навчанні. Це може знизити мотивацію студентів, що негативно вплине на загальну атмосферу навчання.

Тому виникає потреба в створенні зручного та функціонального інформаційного порталу для студентів. Такий портал може допомогти вирішити вищезазначені проблеми та стати централізованим джерелом інформації, де студенти зможуть знайти все, що їм потрібно для навчання. Інформація на порталі буде завжди актуальною, а завдяки зручній навігації студенти зможуть легко знаходити потрібні їм дані.

Метою цієї роботи є створення програмного забезпечення, що дозволить оптимізувати освітній процес і зробити його більш зручним та ефективним для всіх учасників, завдяки використанню сучасних технологій розробки.

Для досягнення мети роботи необхідно виконати наступні завдання:

- дослідити існуючі програмні системи, що мають схожий функціонал та виділити їх переваги і недоліки;

- проаналізувати сучасні технології розробки баз даних, серверів та веб-застосунків;

- спроектувати та розробити базу даних;

- розробити сервер, що буде відповідати за збереження та опрацювання даних;

- розробити інтерфейс користувача;

- провести комплексне тестування розробленої програмної системи.

Підсумовуючи, ця робота спрямована на створення інформаційного веб-порталу, що допоможе студентам, надаючи їм ресурси, що необхідні для досягнення успіху в навчанні. Усуваючи недоліки існуючих систем і використовуючи потенціал сучасних технологій, цей проект має на меті створити ефективне та доступне навчальне середовище.

1 ПОСТАНОВКА ЗАДАЧІ ВЕБ-ПОРТАЛУ ДЛЯ СТУДЕНТІВ КАФЕДРИ

У цьому розділі розглядається постановка задачі створення веб-порталу для студентів кафедри. Портал має адресувати проблеми аналогів і відповідати певним функціональним і нефункціональним вимогам.

1.1 Аналіз предметної області інформаційного порталу для студентів кафедри

За останні два десятиліття освітній процес зазнав значних трансформацій зумовлених розвитком інформаційних технологій. Заклади вищої навчальної освіти стали все частіше інтегрувати різні навчальні платформи: Moodle, Talent LMS, Blackboard Learn тощо.

Незважаючи на ці досягнення сучасний освітній процес все ще стикається з багатьма проблемами. Навчально-методичні матеріали, такі як інструкції до лабораторних та практичних робіт, посібники, конспекти лекцій, часто розкидані по різних платформах, що ускладнює пошук студентами та викладачами необхідної інформації. Комунікація між студентами та викладачами проходить використовуючи різні платформи, багато з яких є застарілими. Без єдиного місця, де можна дізнатися про всі важливі новини та події на кафедрі, студенти можуть пропустити терміни здачі робіт. Крім того, якщо немає можливості дізнатися про семінари, тренінги або інші цікаві події, що відбуваються на кафедрі, то студенти можуть втратити можливості для розвитку у обраній сфері. Це також ускладнює роботу викладачам і керівництву кафедри, адже їм доводиться витрачати додатковий час на розсилку електронних листів або створення оголошень на різних платформах щоб повідомити студентів про події на кафедрі.

Через це постає питання створення веб-порталу, який стане централізованим

джерелом актуальної інформації про події на кафедрі, для студентів та викладачів. Інтеграція такого порталу у навчальний процес кафедри зможе вирішити зазначені вище проблеми і сприяти покращенню академічної успішності студентів.

1.2 Огляд існуючих веб-порталів кафедр

Успішним прикладом інформаційного веб-портал кафедри є портал кафедри комп'ютерних наук Кембриджського університету [1]. Цей портал надає широкий спектр інформації і ресурсів для студентів, викладачів та абітурієнтів.

Цей веб-сайт розбито на багато розділів, кожен з яких містить інформацію для різних категорій користувачів. Розділ новин регулярно оновлюється останніми новинами кафедри та оголошеннями щодо майбутніх воркшопів і лекцій. Розділ для відвідувачів включає детальну інформацію про бакалаврські, магістерські та докторські програми, вимоги до вступу, курси, навчальні плани і інші ресурси для майбутніх студентів. Секція досліджень містить огляд основних напрямків і проектів кафедри, публікації та наукові роботи, а також інформацію щодо співпраці та фінансування. Головну сторінку наведено на рисунку 1.1.

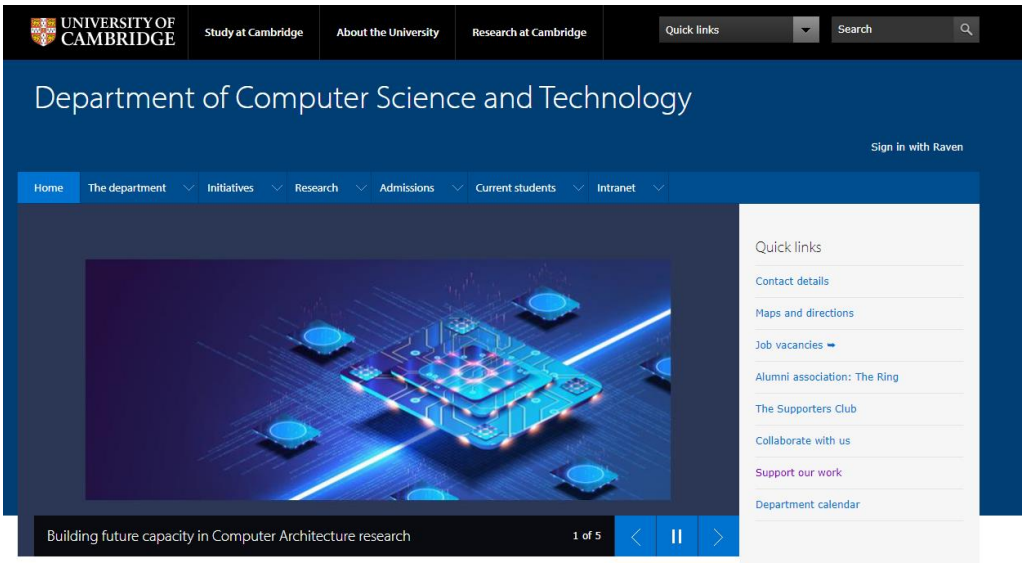


Рисунок 1.1 — Веб-портал кафедри Кембриджського університету

Але даний портал має і недоліки: він перевантажений непотрібною інформацією і не має розділу з освітніми матеріалами, що ускладнює пошук необхідної інформації студентами та викладачами. Крім цього, до списку недоліків можна додати відсутність локалізації, що ускладнює роботу з порталом для тих, хто не володіє англійською мовою і відсутність інтерактивних функцій, таких як форумів для студентів і абітурієнтів.

Дана робота адресує ці проблеми і покращує загальний користувацький досвід за допомогою використання більш сучасних технологій.

1.3 Постановка задачі

Була поставлена задача створити веб-портал для студентів кафедри, що зможе об’єднати усі необхідні для навчання матеріали в одному місці. Цей портал має на меті забезпечити студентам та викладачам легкий доступ до новин кафедри, календарю подій і іншим пов’язаним з навчальним процесом матеріалам. Для цього було визначено наступні функціональні та нефункціональні вимоги до розроблюваної системи.

1.3.1 Функціональні вимоги

Функціональні вимоги визначають основні функції системи, які необхідно реалізувати для забезпечення її коректної роботи та задоволення потреб користувачів. Було визначено наступні функціональні вимоги:

- система має підтримувати 4 типи ролей: студент, представник студентської ради, викладач та адміністратор. В залежності від ролі, користувач має доступ до різного функціоналу;

- усі користувачі можуть переглядати інформацію про себе та за потреби редагувати її;

- усі користувачі можуть переглядати розділ новин та оголошень, навчальні матеріали, календар подій;

- усі користувачі мають доступ до форми зв’язку зі студентською радою;

- адміністратор повинен мати можливість реєструвати нових користувачів у системи;

- адміністратор повинен мати можливість додавати, видаляти та редагувати новини, оголошення, події у календарі;

- адміністратор повинен мати доступ до інформаційної панелі, що містить різноманітні параметри успішності кафедри.

1.3.2 Нефункціональні вимоги

Нефункціональні вимоги визначають ключові характеристики системи, що впливають на її продуктивність, безпеку і надійність. Було визначено наступні нефункціональні вимоги:

- веб-портал повинен мати зручний та інтуїтивно зрозумілий інтерфейс;

- веб-портал має використовувати протокол HTTPS для забезпечення безпеки даних;

- веб-портал має стабільно працювати на всіх версіях браузерів;

- серверна частина порталу має стабільно працювати при великих навантаженнях;

- система має мати можливість легкого розширення функціоналу без значної зміни існуючого коду.

2 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмний продукт, що розглядається у цій роботі буде складатись з трьох компонентів: бази даних, сервера та веб-застосунку. База даних була створена використовуючи СКБД PostgreSQL, сервер – за допомогою фреймворку ASP.NET, а веб-застосунок за допомогою веб-фреймворку Angular. При розробці також було використано шаблон проектування «чиста архітектура».

2.1 Архітектура системи

Архітектура програмного забезпечення визначає спосіб структурування модулів застосунку та взаємодію між ними. Вибір архітектури застосунку є вкрай важливим, бо чітка структура робить код програми легшим для підтримки та розширення і допомагає керувати залежностями та оптимізувати процес розробки.

Чиста архітектура - це набір принципів проектування програмного забезпечення, що ґрунтуються на чіткому розділенні відповідальності та незалежності компонентів [2]. При використанні підходу “Чиста архітектура” застосунок поділяється на логічні шари, кожен з яких відповідає за певну функціональність. Це дозволяє змінювати різні компоненти, наприклад, бази даних, інтерфейс користувача, без впливу на основну логіку програми.

Внутрішній рівень чистої архітектури, містить бізнес-логіку та сутності застосунку. Цей рівень не залежить від будь-яких зовнішніх систем або фреймворків, що гарантує, що основна функціональність програми не буде залежати від деталей реалізації. Навколо цього шару розташовані зовнішні шари, які відповідають за деталі реалізації, такі як користувацький інтерфейс, доступ до бази даних тощо. Ізоляція внутрішнього рівню від цих шарів дозволяє замінювати їх, не впливаючи на функціональність застосунку.

Важливою перевагою чистої архітектури є легкість тестування застосунків,

які її використовують. Ізоляція шарів бізнес-логіки спрощує написання автоматизованих та інтеграційних тестів для ключових компонентів системи. Тести можна писати для основної логіки без взаємодії з базами даних, користувацькими інтерфейсами чи зовнішніми сервісами, що робить їх простішими та надійнішими.

2.2 Вибір інструментів для розробки бази даних

2.2.1 Система керування базами даних PostgreSQL

PostgreSQL - це об’єктно-реляційна система керування базами даних з відкритим кодом, активна розробка якої триває понад 35 років, завдяки чому вона заслужила репутацію надійної і продуктивної СКБД [3]. PostgreSQL підтримує велику кількість типів даних, включаючи примітивні (INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR), структуровані (DATE, TIME, TIMESTAMP, INTERVAL), а також підтримує JSON і XML.

Ця СКБД має ряд переваг порівняно зі своїми конкурентами. Її архітектура дозволяє ефективно масштабуватися, забезпечуючи стабільну продуктивність навіть при значних обсягах даних. PostgreSQL надає різні засоби для контролю доступу, включаючи автентифікацію, шифрування даних, управління ролями і правами доступу на різних рівнях (рядки, таблиці, бази даних) [4]. Розробники можуть створювати та інтегрувати свої доповнення, щоб розширити її можливості. Така гнучкість дозволяє PostgreSQL адаптуватися до конкретних бізнес-вимог та легко інтегруватися з іншими технологіями.

Крім того, PostgreSQL користується сильною підтримкою спільноти та активно розвивається. Її відкритий вихідний код сприяє співпраці між розробниками по всьому світу, що призводить до регулярних оновлень, виправлення помилок і підвищення продуктивності. Це гарантує, що PostgreSQL буде залишатися передовим рішенням для баз даних, що постійно розвивається, щоб задовольнити потреби користувачів.

Нарешті, PostgreSQL не залежить від платформи і працює на різних

операційних системах, включаючи Linux, Windows, macOS і інші Unix-подібні системи. Це робить її доступною для широкого кола користувачів і полегшує розгортання в різних середовищах.

Ця СКБД підтримує інтеграцію з платформою ASP.NET, яку буде використано для створення сервера веб-порталу.

2.2.2 pgAdmin – інструмент для адміністрування баз даних

pgAdmin - це програмне забезпечення, призначене для управління та адміністрування базами даних. Його основною функцією є полегшення роботи з базами даних, дозволяючи користувачам легко й ефективно виконувати широкий спектр задач [5].

pgAdmin дає можливість переглядати і редагувати таблиці баз даних, виконувати запити, а також управляти користувачами та їх рівнями доступу. Крім цього, pgAdmin пропонує ряд додаткових функцій, призначених для керування базами даних, таких як резервне копіювання та відновлення баз даних. Ці можливості забезпечують цілісність і безпеку даних.

2.3 Вибір інструментів для розробки серверної частини системи

2.3.1 C# - сучасна мова програмування платформи .NET

C# - це універсальна багатоплатформна мова програмування, що має мільйони користувачів і є найпоширенішою мовою для платформи .NET [6]. Мова C# заснована на об’єктно-орієнтованих принципах, але також включає елементи інших парадигм програмування, зокрема функціонального програмування.

Мова програмування C# займає провідні позиції серед мов платформи .NET. Програмне забезпечення розроблене на мові C# може працювати на широкому спектрі пристроїв і операційних систем. Це включає мобільні додатки, настільні програми тощо.

Також слід зазначити, що платформа .NET постійно розвивається, що

дозволяє C# залишатися на передовій сучасних технологій розробки програмного забезпечення. Розробники можуть скористатися різноманітними бібліотеками і фреймворками, що значно спрощують процес створення додатків різної складності, забезпечуючи їхню стабільність та продуктивність.

2.3.2 Entity Framework Core – технологія доступу до даних

Entity Framework Core - це сучасний об'єктно-реляційний маппер з відкритим вихідним кодом для платформи .NET, призначений для полегшення доступу до даних при розробці програмного забезпечення [7]. Він дозволяє розробникам взаємодіяти з базою даних за допомогою об'єктів .NET.

Цей фреймворк підтримує інтеграцію з різноманітними СКБД, включаючи PostgreSQL, MySQL, MSSQL та інші. Ця дозволяє обирати базу даних, що найкраще відповідає вимогам програмної системи.

Однією з ключових особливостей Entity Framework є підтримка LINQ - мови запитів, інтегрованої в мови програмування платформи .NET. LINQ дозволяє розробникам писати строго типізовані запити за допомогою обраної мови програмування. При виконанні, ці запити транслюються в запити до бази даних мовою SQL.

Крім цього, Entity Framework включає численні функції, що спрощують управління даними та забезпечують їх цілісність. Автоматичне відстеження змін відстежує кожен отриманий з бази даних запис, що дозволяє ефективно обробляти оновлення бази даних [8]. Каскадне видалення гарантує, що при видаленні запису, пов'язані дані видаляються автоматично без необхідності написання додаткового коду.

Entity Framework також включає в себе інструменти для управління міграціями, що дозволяє розробникам легко вносити зміни в схему бази даних при розширенні функціоналу системи. Крім цього, Entity Framework надає можливість робити асинхронні запити до баз даних, без яких неможливе створення програмного забезпечення, що виконує операції з великим обсягом даних.

2.3.3 ASP.NET – фреймворк для розробки серверних API

ASP.NET - це фреймворк, розроблений компанією Microsoft, за допомогою якого розробники можуть створювати багатофункціональні веб-додатки, використовуючи будь-яку з мов програмування платформи .NET [9].

Цей фреймворк пропонує широкий спектр інструментів для виконання таких поширених завдань, як автентифікація користувачів, перевірка даних і управління сеансами. Крім того, ASP.NET легко інтегрується з іншими технологіями Microsoft, наприклад з Entity Framework Core.

З точки зору продуктивності, ASP.NET значно обходить усі аналоги . Він включає вбудовані механізми кешування даних для мінімізації кількості запитів до бази даних і зменшення навантаження на сервер, а також підтримує асинхронні операції для обробки великої кількості одночасних запитів [10].

2.3.4 Seq – журнал структурованих логів

Seq - це сервер для збереження і обробки логів, розроблений спеціально для роботи зі структурованими логами, що робить його важливим інструментом для розробників, яким потрібно відстежувати, аналізувати і реагувати на поведінку своїх програмних систем. Його основною функцією є централізація логів з різних джерел, що значно спрощує процес моніторингу додатків, виявлення проблем і отримання цінної інформації про стан системи [11].

Однією з особливостей Seq є його багата мова запитів. Вона дозволяє фільтрувати і шукати в журналах з винятковою точністю, дозволяючи точно визначити конкретні події або тенденції, що можуть вказувати на проблеми в роботі програмного забезпечення.

Seq також підтримує можливість налаштування оповіщень, які можна використовувати для сповіщення про досягнення певних умов у даних журналу. Наприклад, можна налаштувати сповіщення, які спрацьовуватимуть, коли кількість помилок перевищить певний поріг. Ці сповіщення можуть бути інтегровані з іншими інструментами моніторингу та оповіщення, гарантуючи, що розробники завжди будуть в курсі появи критичних проблем в роботі системи.

2.4 Вибір інструментів для розробки клієнтської частини системи

2.4.1 TypeScript – розширення для мови JavaScript

TypeScript - це строго типізована мова програмування, що базується на JavaScript і призначена для покращення та розширення можливостей JavaScript [12]. Вона дозволяє розробникам виявляти помилки на ранніх стадіях процесу розробки веб-застосунків за допомогою перевірки типів. Це особливо корисно у великих проектах, де раннє виявлення помилок з типізацією може заощадити значний час і ресурси.

Здатність керувати складними проектами та зменшувати кількість помилок під час виконання робить TypeScript привабливим вибором для компаній, які прагнуть створювати надійні та зручні в обслуговуванні веб-додатки. Система типів TypeScript допомагає забезпечити дотримання контрактів у коді, полегшуючи його розуміння, що має вирішальне значення для підтримки програмних систем.

2.4.2 Angular – фреймворк для створення веб-застосунків

Angular - це фреймворк розроблений компанією Google призначений для створення веб-додатків. Angular використовує компонентну архітектуру, що дозволяє розробникам створювати багаторазові компоненти. Це допомагає полегшити розробку та підтримку великих додатків.

Angular пропонує власну систему прив'язування даних, що дозволяє автоматично синхронізувати модель і представлення, гарантуючи, що інтерфейс користувача завжди буде відповідати стану додатку [13]. Окрім цього цей фреймворк, надає потужний маршрутизатор, що підтримує ліниве завантаження, вкладеність маршрутів і інші корисні функції.

2.4.3 Angular Material – бібліотека компонент інтерфейсу користувача

Angular Material - це бібліотека компонентів для розробки інтерфейсу користувача, створена спеціально для веб-фреймворку Angular. Вона пропонує широкий спектр компонентів, що легко налаштовуються під потреби додатку.

Бібліотека включає багато різноманітних компонент, таких як календарі, кнопки, піктограми, картки тощо [14].

Angular Material дозволяє розробникам швидко створювати веб-додатки із зручним інтерфейсом. Усі компоненти бібліотеки є адаптивними, що гарантує, що розроблені веб-додатки будуть функціонувати на широкому спектрі.

2.4.4 TailwindCSS – сучасний CSS-фреймворк

Tailwind CSS - це CSS-фреймворк, розроблений, щоб допомогти розробникам створювати сучасні веб-дизайни. На відміну від традиційних CSS-фреймворків, які надають заздалегідь визначені компоненти та стилі, Tailwind CSS пропонує повний набір низькорівневих класів, які можна застосовувати безпосередньо до елементів HTML. Ці утиліти охоплюють широкий спектр потреб дизайну, включаючи шрифти, інтервали, кольори тощо, що дозволяє розробникам створювати застосунки без написання CSS.

Tailwind надає можливість легко налаштовувати систему дизайну під конкретні вимоги проекту за допомогою конфігураційних файлів. Так, наприклад, можна змінювати колірні схеми, налаштування інтервалів та адаптивних точок розриву [15].

Tailwind CSS побудований за філософією mobile-first, що означає, що спочатку пріоритет надається дизайну для невеликих екранів, а потім поступово покращується дизайн для великих екранів. Такий підхід гарантує, що веб-додатки будуть адаптивними за замовчуванням, забезпечуючи кращий користувацький досвід на різних пристроях.

Tailwind CSS також включає потужні функції, такі як PurgeCSS, яка допомагає оптимізувати остаточний пакет CSS, видаляючи невикористовувані стилі у виробництві. Це призводить до зменшення розміру CSS-файлів, що покращує продуктивність веб-додатків.

2.4.5 NgRX Signals – бібліотека для збереження стану застосунку

NgRX - це бібліотека призначена для управління станом веб-додатків, розроблена спеціально для Angular. Вона дозволяє розробникам створювати реактивні додатки, централізуючи усю логіку управління станом застосунку.

NgRx Signals використовує додану у останніх версіях Angular технологію сигналів, що призводить до ефективного рендерингу і кращої продуктивності в складних додатках.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Розроблена система складається з 3 компонент: бази даних, сервера та веб-порталу. В цьому розділі буде детально розглянута реалізація кожного з них.

3.1 Розробка бази даних та серверу

Для розробки серверу було обрано фреймворк ASP.NET. Для початку роботи було створено новий ASP.NET Web API проект у інтегрованому середовищі розробки Rider.

Далі було створено пусту базу даних в застосунку pgAdmin. Після цього на сервері було встановлено пакети Entity Framework Core. Наступним кроком було описано всі необхідні таблиці за допомогою класів мови C#. Приклад опису таблиці Groups наведено на рисунку 3.1.

```
public class Group
{
    2 usages
    public long Id { get; set; }

    3 usages
    public required string Name { get; set; }

    9 usages
    public int Year { get; set; }

    2 usages
    public long SpecialtyId { get; set; }

    5 usages
    public GroupType GroupType { get; set; }

    6 usages
    public Specialty Specialty { get; set; } = null!;

    6 usages
    public List<User> Users { get; set; } = new List<User>();
}
```

Рисунок 3.1 — Приклад опису таблиці Groups

Усього було розроблено 8 таблиць. Схему бази даних представлено на рисунку 3.2.

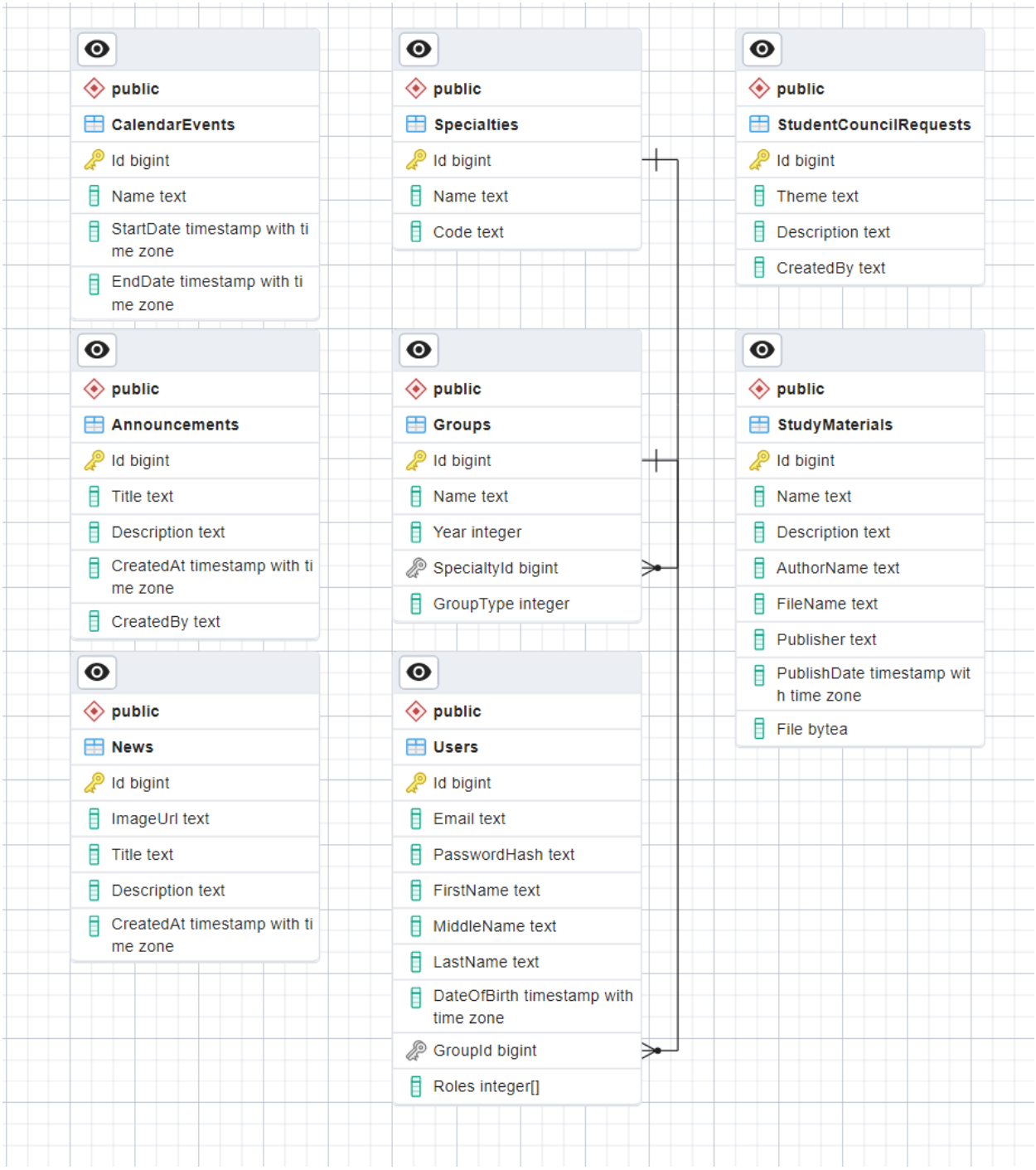


Рисунок 3.2 — Схема бази даних веб-порталу

Розглянемо основні таблиці бази даних і їх поля.

Таблиця 3.1 – Опис таблиці Users

Поле	Тип поля	Опис
Id	bigInt	Унікальний ідентифікатор
Email	text	Електрона пошта користувача
PasswordHash	text	Пароль користувача
FirstName	text	Ім’я користувача
MiddleName	text	По-батькові користувача
LastName	text	Прізвище користувача
DateOfBirth	timestamp with time zone	Дата народження користувача
GroupId	bigint	Унікальний ідентифікатор групи користувача. Це поле є зовнішнім ключем до таблиці Groups. Це поле не є обов’язковим.
Roles	integer[]	Масив ролей користувача

Таблиця 3.2 – Опис таблиці Groups

Поле	Тип поля	Опис
Id	bigInt	Унікальний ідентифікатор
Name	text	Назва групи

Кінець таблиці 3.2

Поле	Тип поля	Опис
SpecialtyId	bigInt	Ідентифікатор спеціальності
GroupType	integer	Тип групи. Можливі значення 0 – Бакалаври і 1 - Магістри

Таблиця 3.3 – Опис таблиці Specialties

Поле	Тип поля	Опис
Id	bigInt	Унікальний ідентифікатор спеціальності
Name	text	Назва спеціальності
Code	text	Код спеціальності

Таблиця 3.4 – Опис таблиці Announcements

Поле	Тип поля	Опис
Id	bigInt	Унікальний ідентифікатор оголошення
Title	text	Тема оголошення
Description	text	Текст оголошення
CreatedAt	timestamp with time zone	Дата створення оголошення
CreatedBy	text	Пошта користувача створившого оголошення

Таблиця 3.5 – Опис таблиці StudentCounsilRequests

Поле	Тип поля	Опис
Id	bigInt	Унікальний ідентифікатор запиту
Theme	text	Тема запиту
Description	text	Опис запиту
CreatedBy	text	Електрона пошта користувача створившого запит

Таблиця 3.6 – Опис таблиці StudyMaterials

Поле	Тип поля	Опис
Id	bigInt	унікальний ідентифікатор навчального матеріалу
Name	text	Назва навчального матеріалу
Description	text	Короткий опис навчального матеріалу
AuthorName	text	Перелік авторів
FileName	text	Назва файлу
Publisher	text	Видавництво
PublishDate	timestamp with time zone	Дата видання

Таблиця 3.7 – Опис таблиці NewsRecords

Поле	Тип поля	Опис
Id	bigInt	Унікальний ідентифікатор новини
ImageUrl	text	Посилання на зображення
Title	text	Заголовок новини
Description	text	Текст новини
CreatedAt	timestamp with time zone	Дата створення новини

Таблиця 3.8 – Опис таблиці CalendarEvents

Поле	Тип поля	Опис
Id	bigInt	Унікальний ідентифікатор події
Name	text	Назва події
StartDate	timestamp with time zone	Дата початку події
EndDate	timestamp with time zone	Дата закінчення події

Після створення усіх таблиць розроблену базу даних було приєднано до сервера використовуючи ASP.NET Middleware.

```
public static IServiceCollection AppendDatabase(this IServiceCollection services, IConfiguration configuration)
{
    var connectionString = configuration.GetConnectionString(name: "DefaultConnection");

    services.AddDbContext<ApplicationDbContext>(optionsAction: opt :DbContextOptionsBuilder => opt.UseNpgsql(connectionString));

    return services;
}
```

Рисунок 3.3 — Налаштування підключення між базою даних та сервером

Щоб закінчити інтеграцію бази даних з сервером було створено міграцію використовуючи інструмент управління міграціями, що пропонується інтегрованим середовищем розробки Rider (рисунок 3.4).

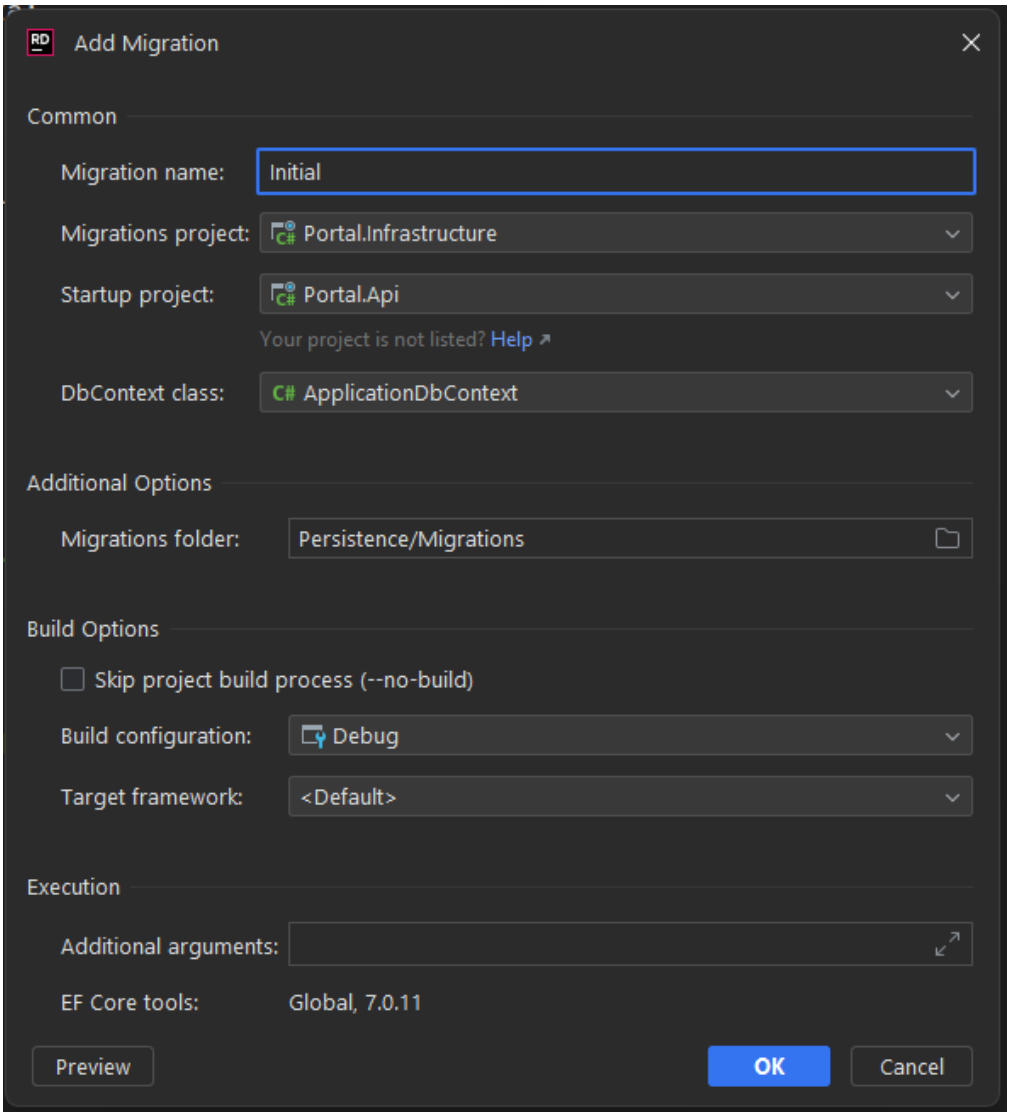
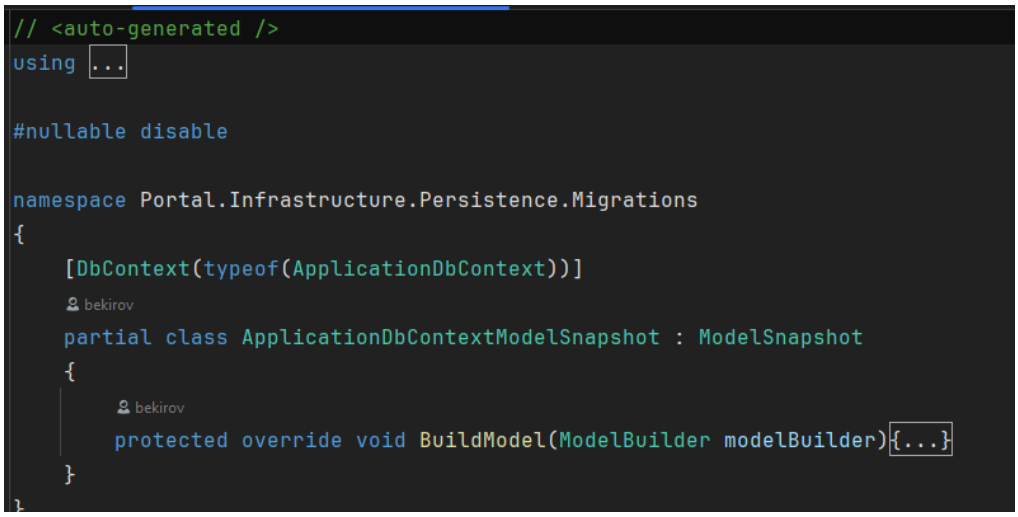


Рисунок 3.4 — Створення початкової міграції

Після виконання міграції було згенеровано наступні файли (рисунок 3.5-3.6). Далі, за потреби, можна використати все той же інструмент для управління міграціями і ці файли щоб повернутися на минулі версії бази даних.

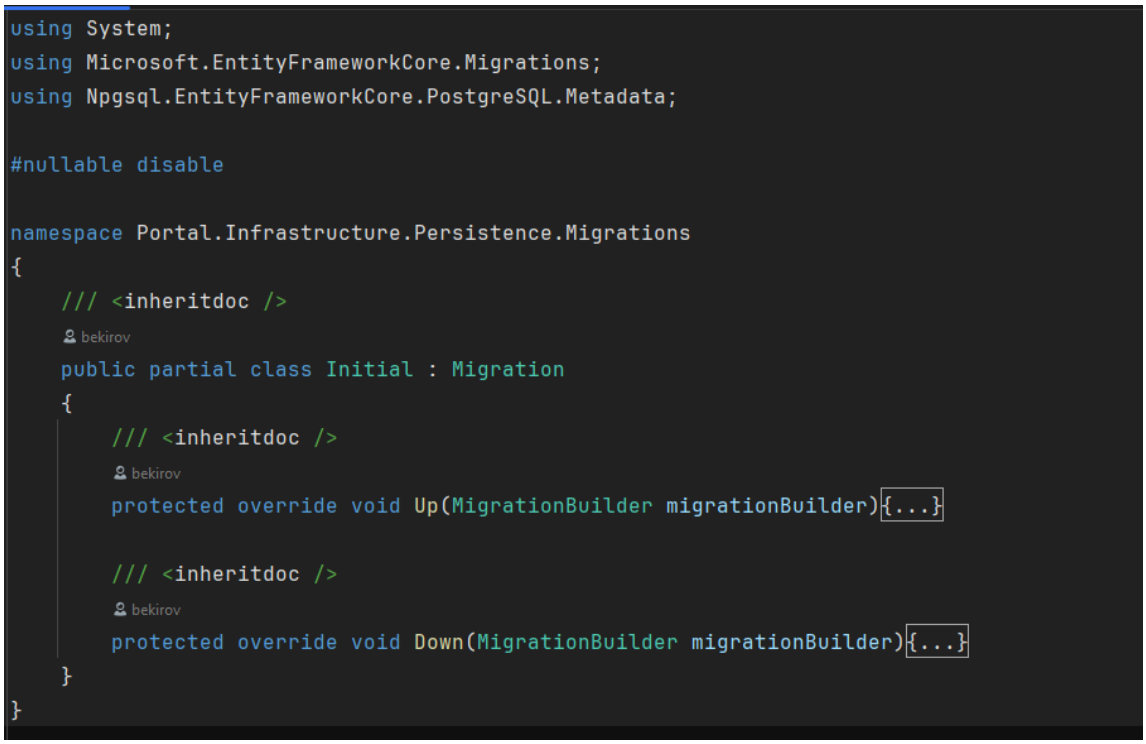


```
// <auto-generated />
using ...

#nullable disable

namespace Portal.Infrastructure.Persistence.Migrations
{
    [DbContext(typeof(ApplicationDbContext))]
    bekirov
    partial class ApplicationDbContextModelSnapshot : ModelSnapshot
    {
        bekirov
        protected override void BuildModel(ModelBuilder modelBuilder){...}
    }
}
```

Рисунок 3.5 — Файл міграції



```
using System;
using Microsoft.EntityFrameworkCore.Migrations;
using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

#nullable disable

namespace Portal.Infrastructure.Persistence.Migrations
{
    /// <inheritdoc />
    bekirov
    public partial class Initial : Migration
    {
        /// <inheritdoc />
        bekirov
        protected override void Up(MigrationBuilder migrationBuilder){...}

        /// <inheritdoc />
        bekirov
        protected override void Down(MigrationBuilder migrationBuilder){...}
    }
}
```

Рисунок 3.6 — Файл міграції

Наступним кроком було налаштовано інтеграцію логера Serilog з сервером. Serilog буде відстежувати помилки та іншу інформацію, що може допомогти з діагностикою проблем програмного забезпечення.

Щоб інтегрувати Serilog з ASP.NET, спочатку потрібно інсталювати пакети

Serilog через менеджер пакетів NuGet, який використовується платформою .NET. Після цього можна задати параметри конфігурації логеру.

На рисунку 3.7 представлена конфігурація логеру, що використовується розробленим сервером. Ця конфігурація налаштовує мінімальний рівень помилок, які відстежуються логером, встановлені плагіни та вихідні джерела. В даному випадку це консоль та Seq сервер.

```
"MinimumLevel": {
  "Default": "Information",
  "Microsoft": "Warning",
  "System": "Warning"
},
"Enrich": [
  "FromLogContext",
  "WithExceptionDetails"
],
"WriteTo": [
  {
    "Name": "Console",
    "Args": {
      "theme": "Serilog.Sinks.SystemConsole.Themes.AnsiConsoleTheme::Code, Serilog.Sinks.Console"
    }
  },
  {
    "Name": "Seq",
    "Args": {
      "serverUrl": "http://localhost:60443",
      "apiKey": "bvNwyHl7vjqYkvGTEIio",
      "restrictedToMinimumLevel": "Warning"
    }
  }
]
```

Рисунок 3.7 — Конфігурація Serilog

Після закінчення налаштування можна підключитися до Seq використовуючи веб-інтерфейс (рисунок 3.8). За допомогою веб-інтерфейсу можна переглядати, аналізувати та керувати журналами подій у реальному часі, що значно полегшить моніторинг веб-порталу.

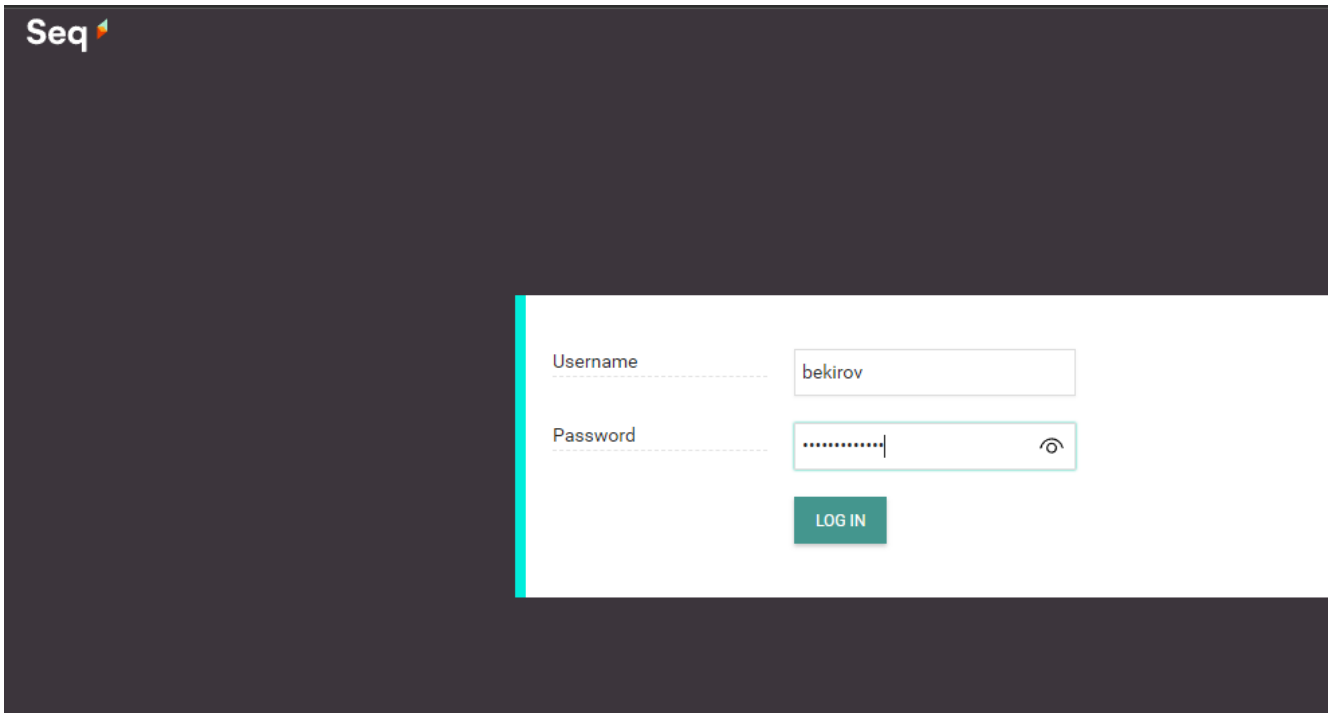


Рисунок 3.8 — Автентифікація у Seq

Після авторизації можна побачити інформаційну панель на якій містяться графіки, що показують кількість помилок та інших подій (рисунок 3.9).

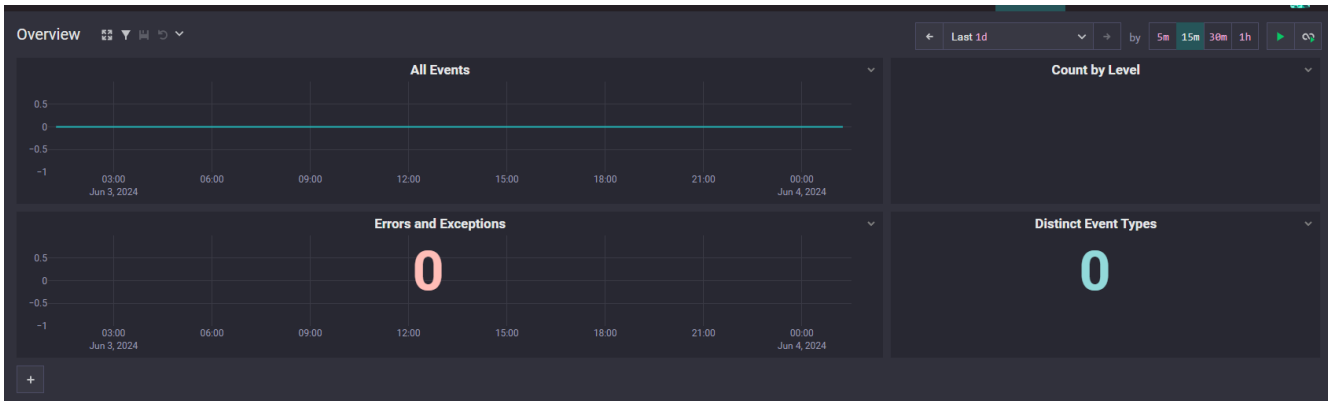


Рисунок 3.9 — Інформаційна панель SEQ

На цій інформаційній панелі можна побачити помилки, попередження, інформаційні повідомлення тощо.

Також можна фільтрувати події журналу на основі різних критеріїв, таких як

часовий діапазон, рівень тощо. Це допомагає визначити конкретні проблеми в даних журналу.

Далі було створено набір класів для роботи з базою даних. Основну роль у цьому відіграє клас Repository, який забезпечує методи для виконання різноманітних операцій над записами в таблицях бази даних. Цей клас включає в себе методи для читання даних, видалення існуючих записів, створення нових і редагування наявних записів. Кожен з цих методів відповідає за конкретну операцію, що дозволяє ефективно взаємодіяти з даними у таблицях.

На рисунку 3.10 представлено UML-діаграму класу Repository.

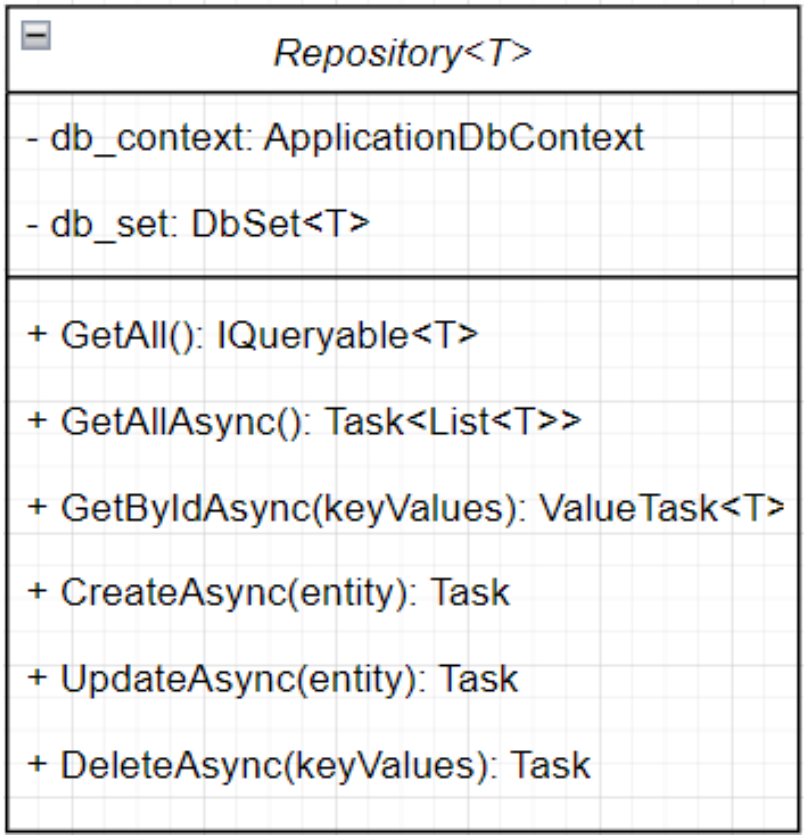


Рисунок 3.10 — UML-діаграма класу Repository

Далі було розроблено сервіси, які відповідають за основну логіку серверної части. Опис основних сервісів представлено у таблиці 3.9.

Таблиця 3.9 – Опис основних сервісів серверної частини веб-порталу

Сервіс	Функції
UserService	Управління користувачами: визначає методи для реєстрації нових користувачів у системі, авторизації користувачів, отримання та оновлення профіля користувача, отримання статистики по користувачам на порталі.
JwtService	Управління авторизацією за допомогою Json Web Tokens: визначає методи для створення JWT токенів, що використовуються для авторизації користувача у системі.
AnnouncementService	Управління оголошеннями: визначає методи для отримання оголошень відсортованих за датою, створення нових оголошень, оновлення та видалення існуючих оголошень.
NewsService	Управління новинами: визначає методи для отримання останніх новин відсортованих за датою, створення, оновлення та видалення новин.
CalendarEventService	Управління подіями на кафедрі: визначає методи для отримання останніх подій, створення та видалення подій.

Кінець таблиці 3.9

Сервіс	Функції
StudyMaterialService	Управління навчально-методичними матеріалами: визначає методи для отримання усіх матеріалів, отримання за унікальним ідентифікатором, створення та видалення матеріалів.
StudentCouncilService	Управління запитами до студентської ради: визначає методи для створення нових запитів та перегляду існуючих.
GroupsService	Управління навчальними групами: визначає методи для отримання списку усіх груп та створення нових.
SpecialtiesService	Управління спеціальностями: визначає методи для отримання списку усіх спеціальностей та створення нових.

Нарешті було створено контролери. Контролери дозволяють розділити бізнес-логіку застосунку та його представлення, забезпечуючи взаємодію між користувачем та застосунком. UML-діаграми основних контролерів представлено на рисунках 3.11-3.14. Ці діаграми відображають взаємодію контролерів з сервісами та їх структуру, що допоможе краще візуалізувати архітектурні аспекти серверної частини веб-порталу.

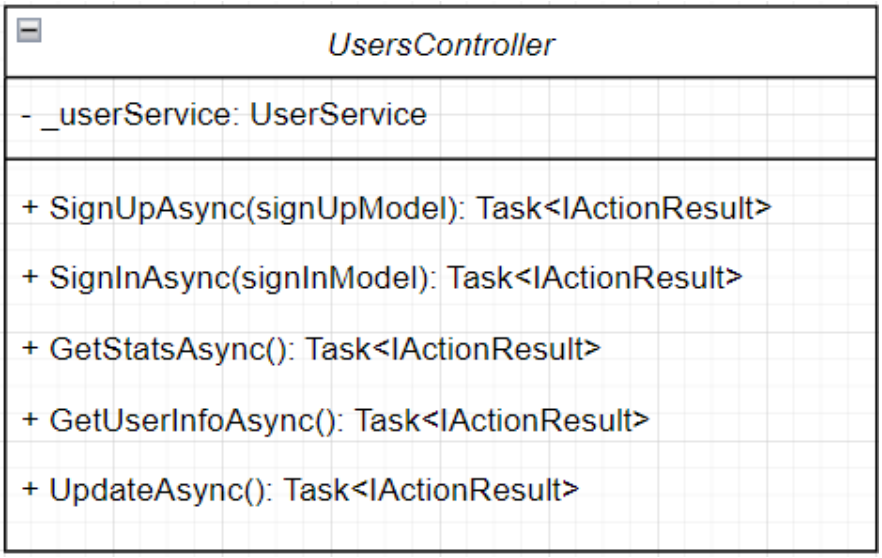


Рисунок 3.11 — UML-діаграма класу UsersController

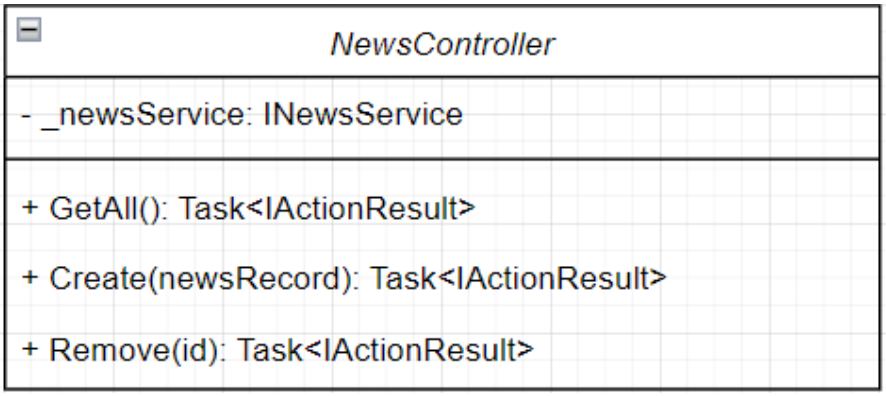


Рисунок 3.12 — UML-діаграма класу NewsController

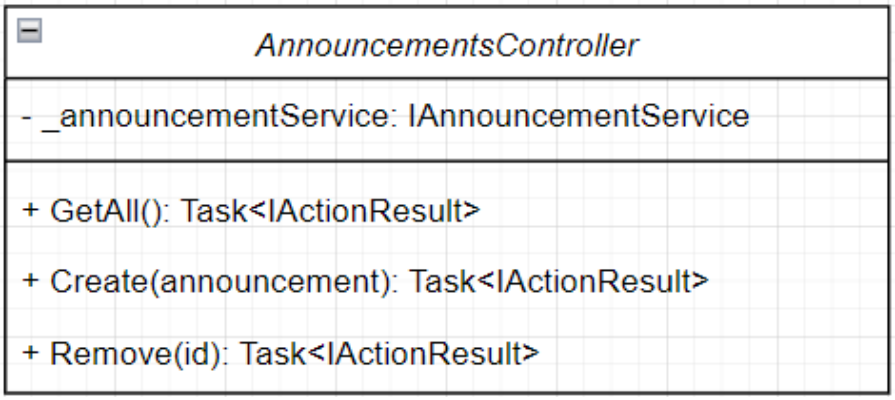


Рисунок 3.13 — UML-діаграма класу AnnouncementsController

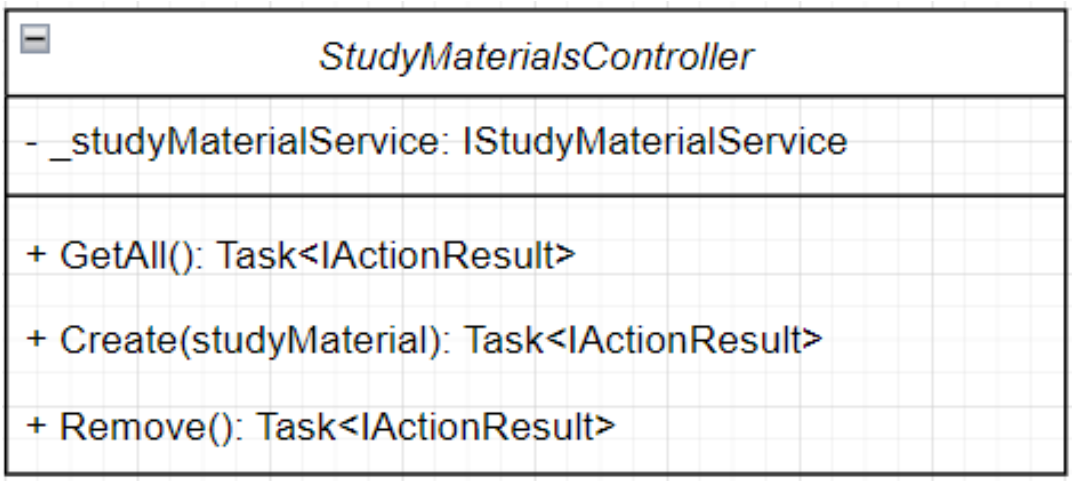


Рисунок 3.14 — UML-діаграма класу StudyMaterialsController

У ASP.NET контролери відповідають за опрацювання HTTP-запитів та відправку відповіді на клієнт.

3.2 Розробка веб-порталу

Спочатку було створено новий Angular проект використовуючи Angular CLI. Після цього було встановлено пакет @ngrx/signals який буде використано для управління станом застосунку і @angular/material для створення користувацького інтерфейсу.

Після завершення встановлення необхідних пакетів, було розроблено компоненти застосунку Кожен компонент був ретельно протестований для забезпечення його коректної роботи в межах застосунку.

Компоненти можна повторно використовувати в різних частинах програми, що сприяє зменшенню дублювання коду. Кожен компонент в Angular інкапсулює свою логіку та стилі, гарантуючи, що зміни в ньому не впливатимуть на інші компоненти. Приклад компоненти, що відповідає за профіль користувача наведено на рисунку 3.15.

```
<form class="flex flex-col items-center pt-5 gap-y-5" [formGroup]="form">
    <div class="flex flex-col border border-gray-400 rounded-lg w-[500px] p-10">
        <h1 class="text-xl font-medium mb-2.5">Основна інформація</h1>
        <mat-form-field>
            <mat-label>Ім'я</mat-label>
            <input matInput formControlName="firstName" />
        </mat-form-field>
        <mat-form-field>
            <mat-label>По-батькові</mat-label>
            <input matInput formControlName="middleName" />
        </mat-form-field>
        <mat-form-field>
            <mat-label>Прізвище</mat-label>
            <input matInput formControlName="lastName" />
        </mat-form-field>
        @if(userStore.user()?.group) {
        <span class="mb-5">Група: {{ userStore.user()?.group?.name }}</span>
        <span
            >Спеціальність: {{ userStore.user()?.group?.specialty?.name }}</span
        >
        }
    </div>
    <div class="flex flex-col border border-gray-400 rounded-lg w-[500px] p-10">
        <h1 class="text-xl font-medium mb-2.5">Контакти</h1>
        <mat-form-field>
            <mat-label>Пошта</mat-label>
            <input matInput formControlName="email" />
        </mat-form-field>
    </div>
    <button mat-flat-button color="primary" (click)="edit()">Зберегти</button>
</form>
```

Рисунок 3.15 — компонента Profile

Архітектура даного веб-застосунку побудована на модулях. Кожен функціональний модуль містить усі необхідні компоненти, сервіси та інші супутні класи, характерні для цієї функції. Так, наприклад, модуль новин містить усі компоненти, що відповідають за відображення розділу новин, а також сервіси, що зберігають і оброблюють дані пов’язані з новинами.

Для підвищення продуктивності веб-застосунку було реалізовано ліниве завантаження, яке гарантує, що модулі програми завантажуватимуться лише тоді, коли це необхідно.

Структуру клієнтської частини веб-порталу представлено на рисунку 3.16.

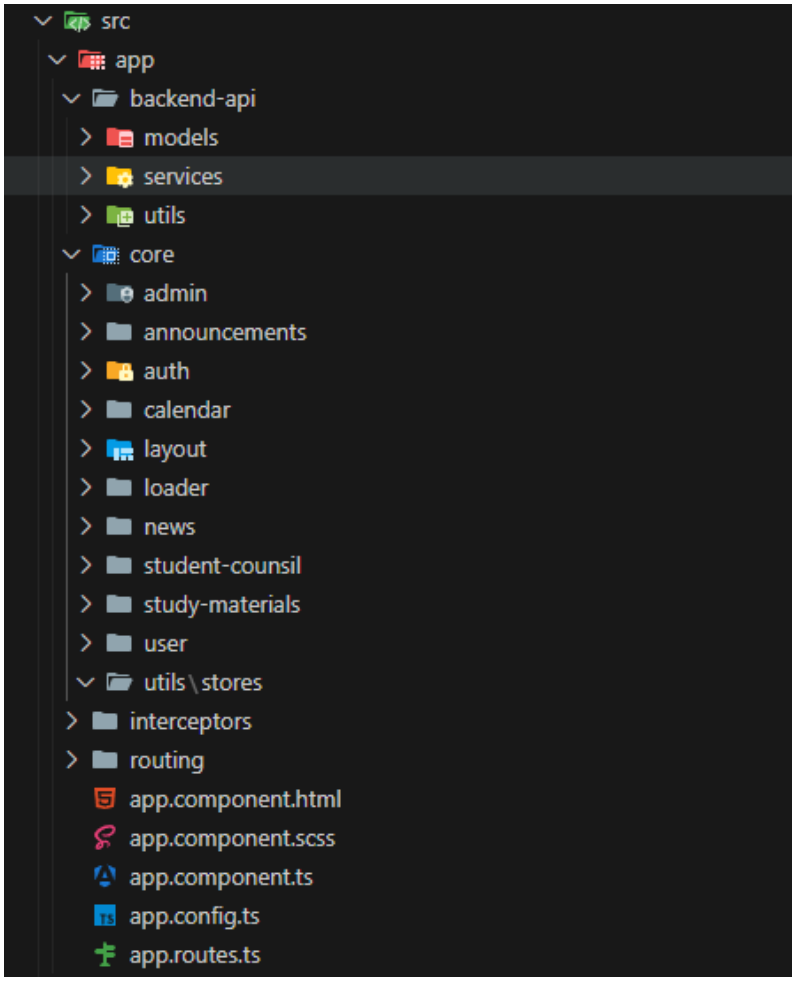


Рисунок 3.16 — Архітектура веб-застосунку

У таблиці 3.10 наведено основні компоненти веб-порталу кафедри разом з їх функціоналом та залежностями.

Таблиця 3.10 – Опис основних компонент веб-застосунку

Сторінка	Функціонал	Залежності
NewsList	Перегляд розділу новин, створення та видалення новин.	MatDialog, MatSnackBar, NewsStore, UserStore
AnnouncementsList	Перегляд розділу оголошень створення та видалення оголошень.	MatDialog, MatSnackBar, AnnouncementsStore, UserStore

Кінець таблиці 3.10

AdminDashboard	Реєстрація нових користувачів, груп, спеціальностей у системі. Перегляд статистики.	MatDialog, MatSnackBar, SpecialtiesStore, GroupsStore, AdminStore
Calendar	Перегляд календарю подій, створення і видалення подій	MatDialog, CalendarEventsStore, UserStore

Наведена таблиця наочно відображає основні сторінки веб-порталу, їх функціональні можливості та ключові залежності між компонентами. Це дозволяє легко орієнтуватися у структурі проекту та взаємозв'язках між його елементами, що сприяє ефективному управлінню розробкою веб-порталу.

4 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

У цьому розділі буде розглянута робота користувача з системою. Розроблений веб-портал має 4 ролі: студент, представник студентської ради, викладач та адміністратор. Було виконано авторизацію у системі в якості користувача з роллю “Адміністратор”. Таким чином можна буде одразу побачити весь функціонал застосунку.

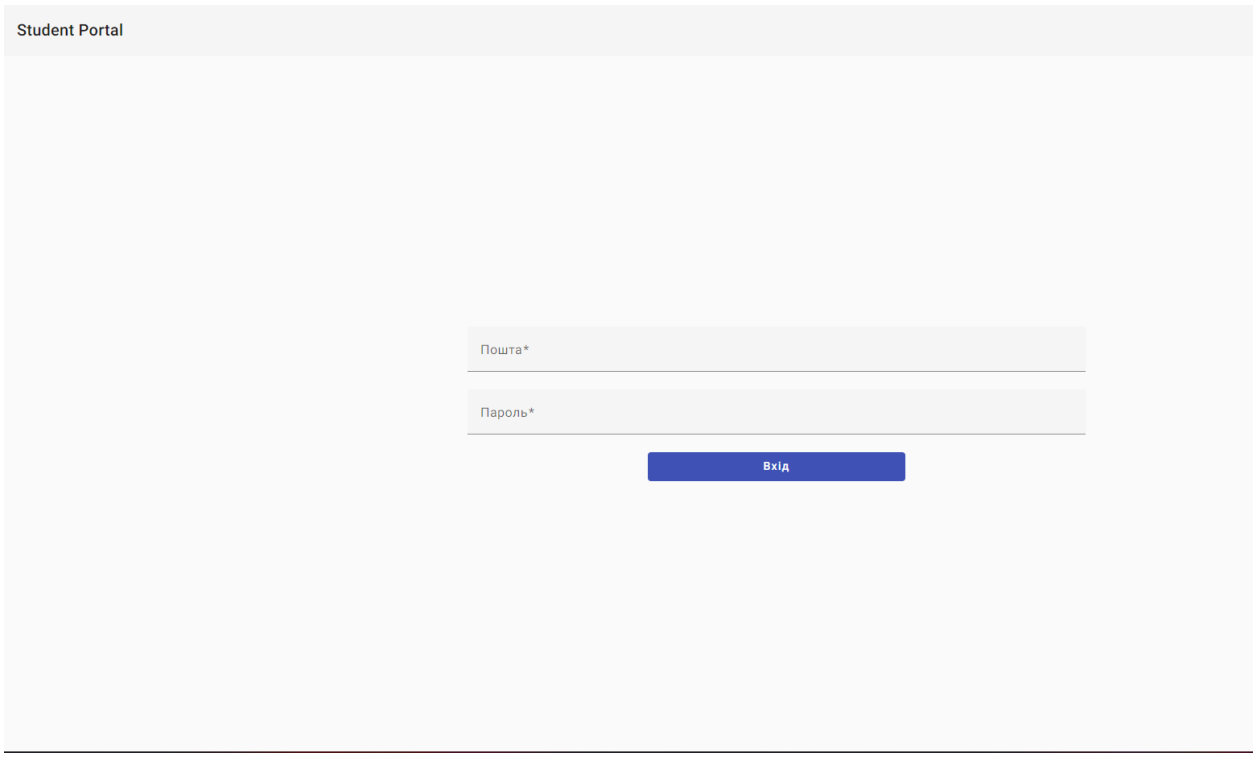


Рисунок 4.1 — Авторизація у веб-порталі кафедри

Після авторизації користувач потрапляє на сторінку зі своїм профілем. Там він може побачити інформацію про себе, спеціальність та групу (якщо користувач є студентом) та свої контактні дані. За потреби користувач може оновити своє ПІБ та контактні дані.

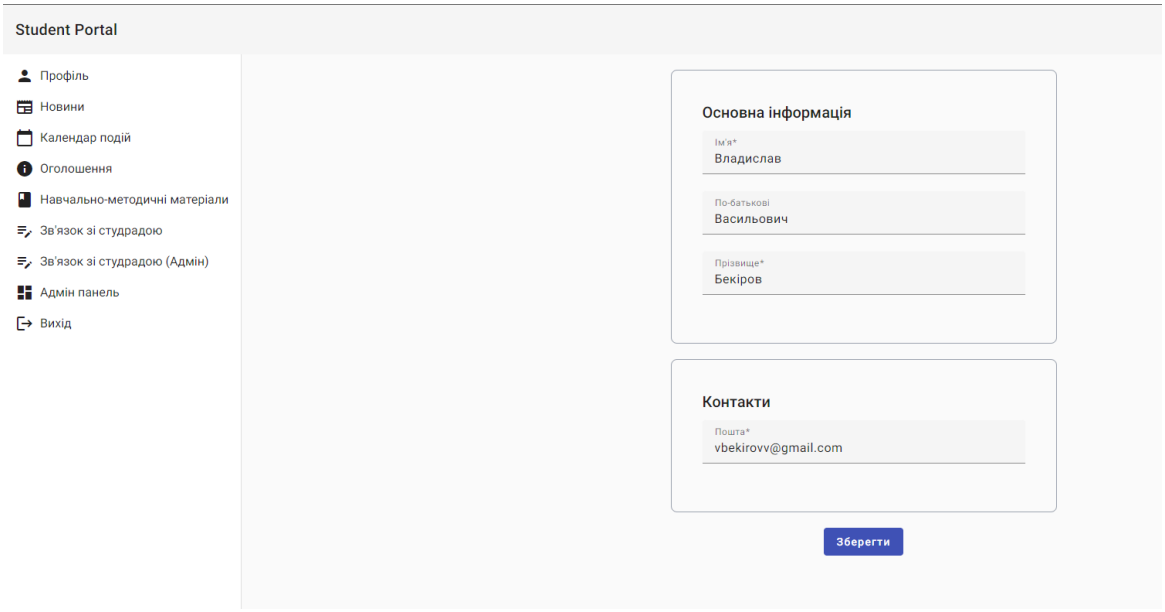


Рисунок 4.2 — Профіль користувача

Скориставшись панеллю навігації користувач може потрапити на різні сторінки веб-порталу. Перейшовши на сторінку з новинами можна побачити новини упорядковані по даті.

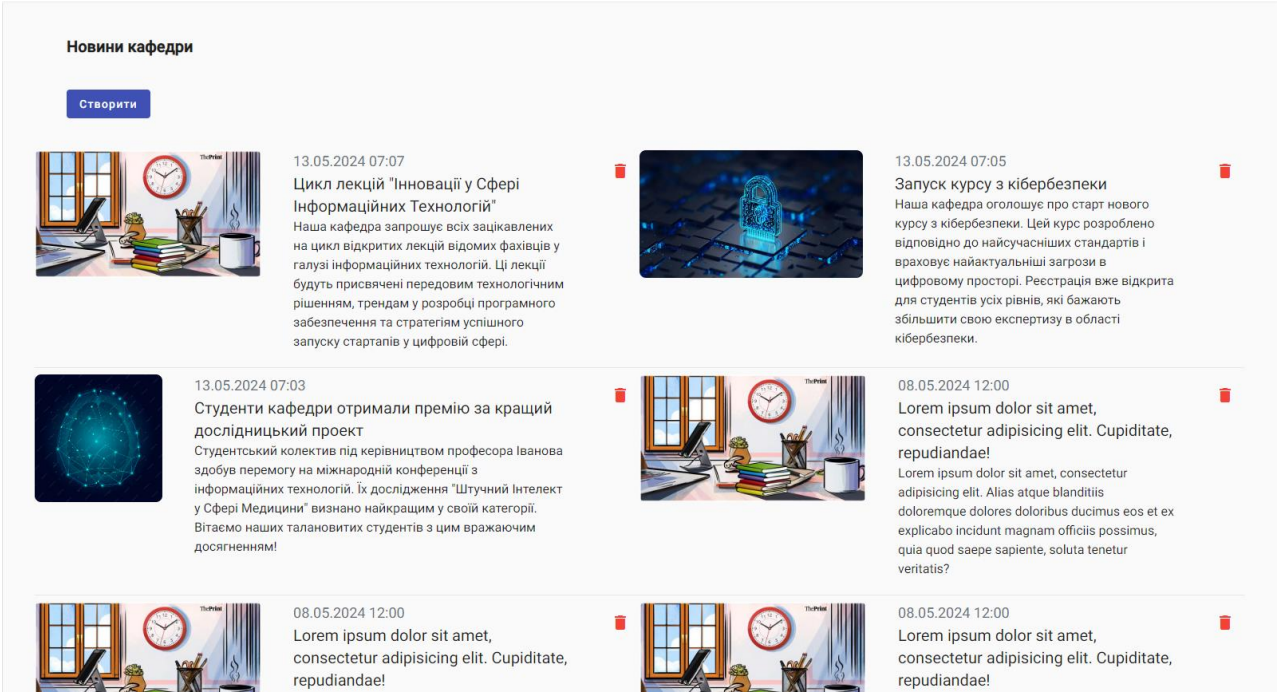


Рисунок 4.3 — Розділ з новинами

Користувач має роль “Адміністратор”, тому має змогу видаляти існуючі новини або створювати нові. Для створення новини необхідно вказати заголовок, текст новини та посилання на зображення.

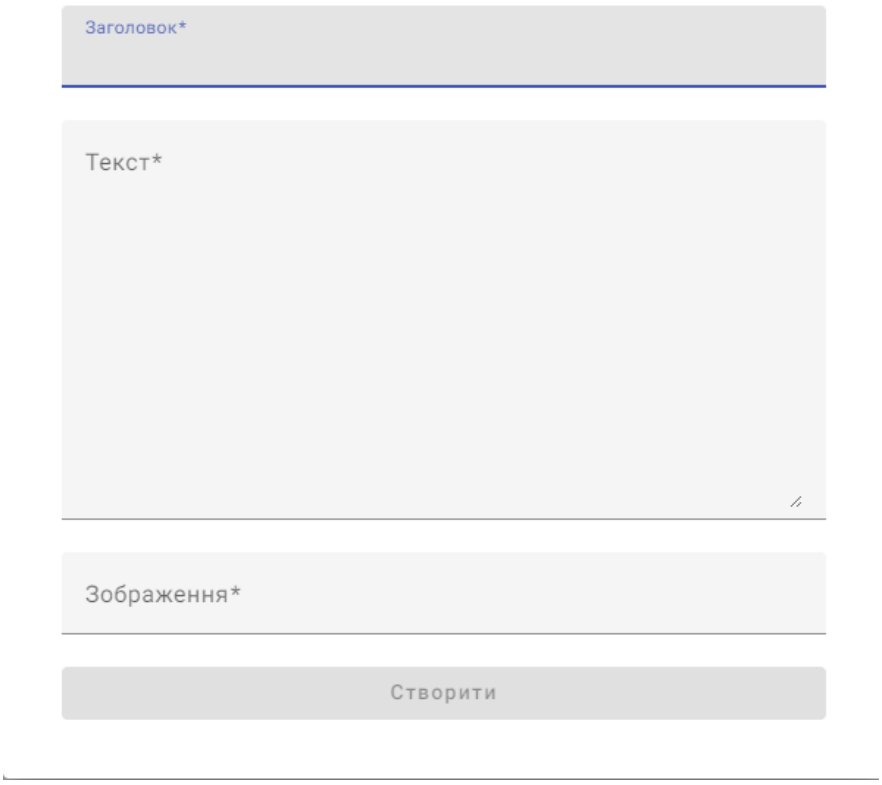


Рисунок 4.4 — Створення новини

Далі було здійснено перехід на сторінку з оголошеннями. Тут можна ознайомитись з останніми оголошеннями, що опубліковані викладачами або адміністраторами.

Кожне з них містить заголовок, дату публікації та текст повідомлення, що дозволяє студентам швидко отримувати актуальну інформацію.

Маючи роль “Адміністратор” можна додавати або видаляти оголошення. Для створення нового оголошення потрібно вказати лише заголовок та його текст. Роботу користувача з розділом оголошень наведено на рисунках 4.5-4.6.

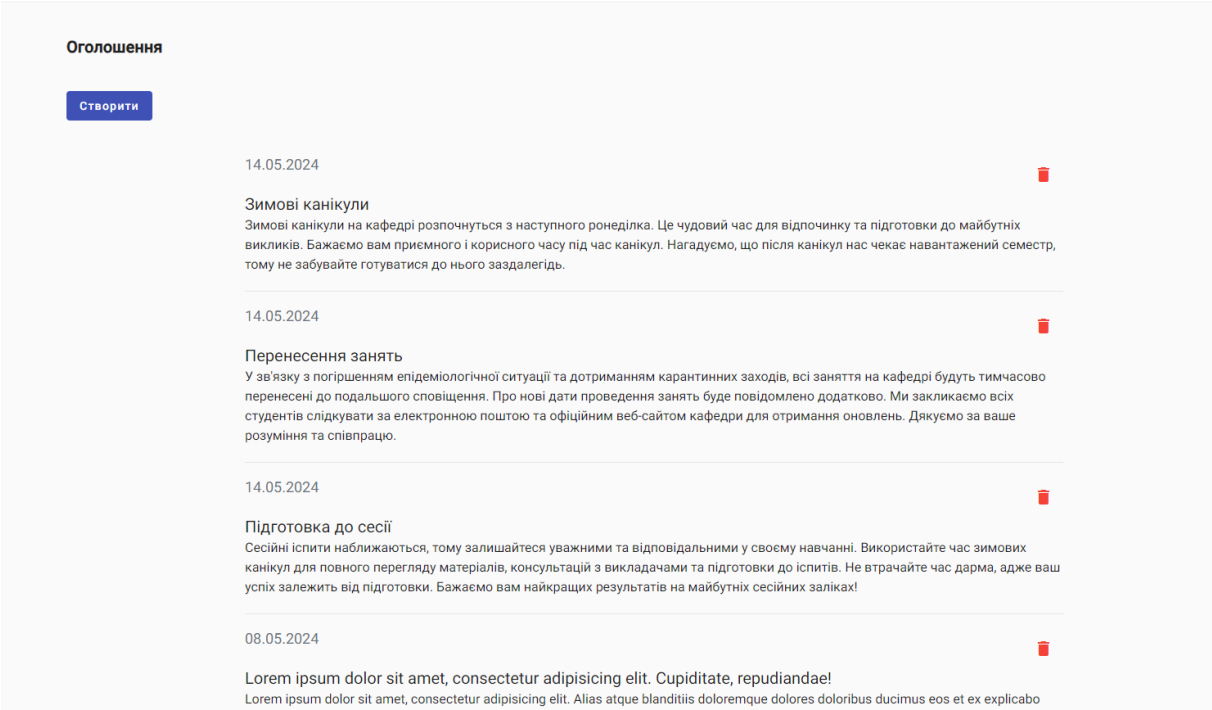


Рисунок 4.5 — Розділ з оголошеннями

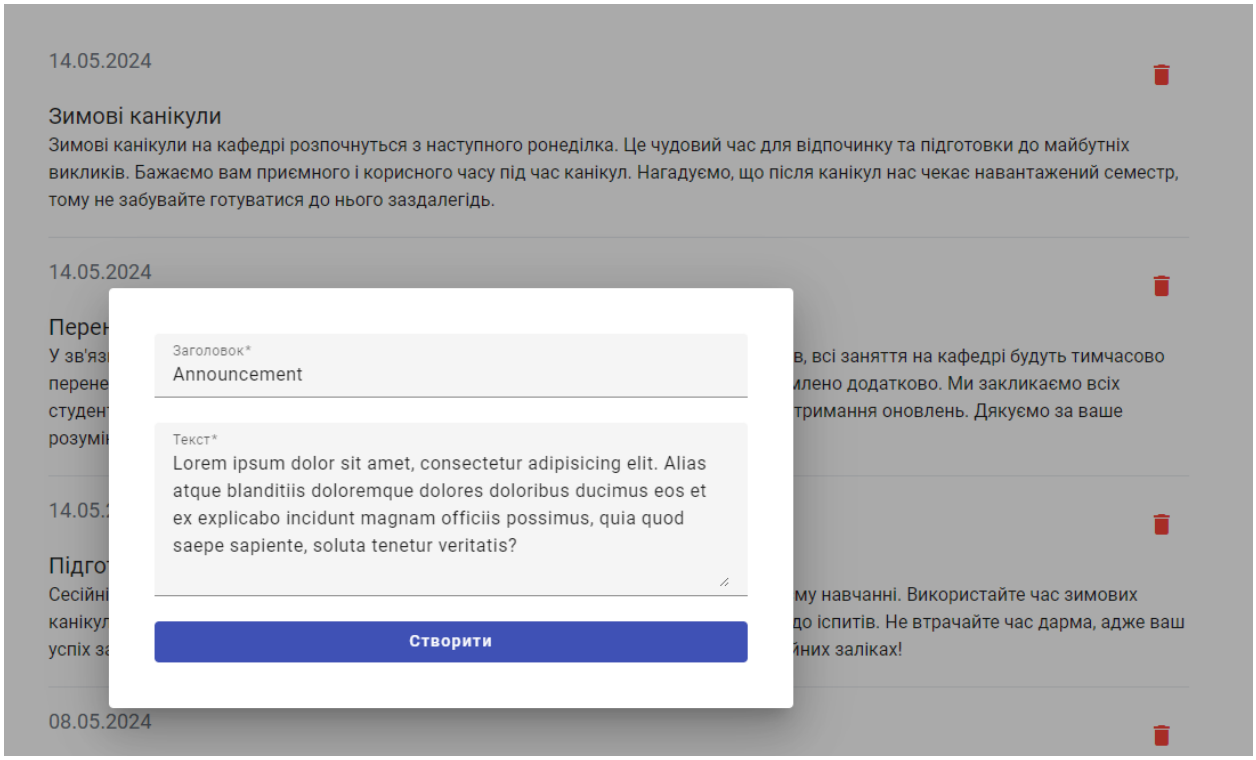


Рисунок 4.6 — Створення нового оголошення

Перейшовши на сторінку з календарем подій, користувач може побачити найближчі події, які будуть відбуватись на кафедрі.

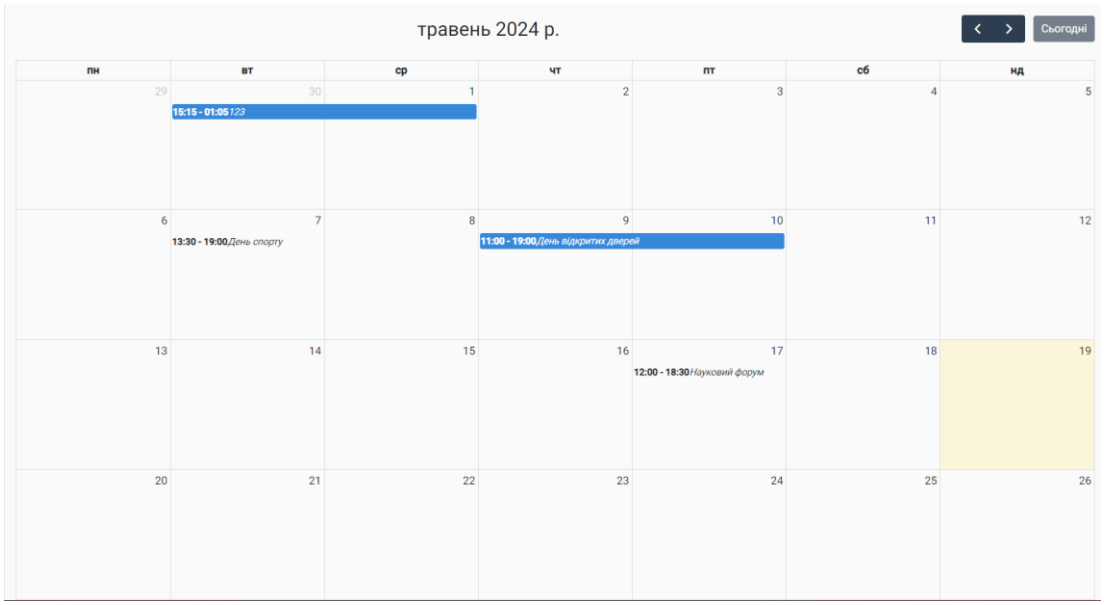


Рисунок 4.7 — Календар подій

Дозволяється виділити на календарі дату щоб створити нову подію або натиснути на існуючу щоб видалити її.

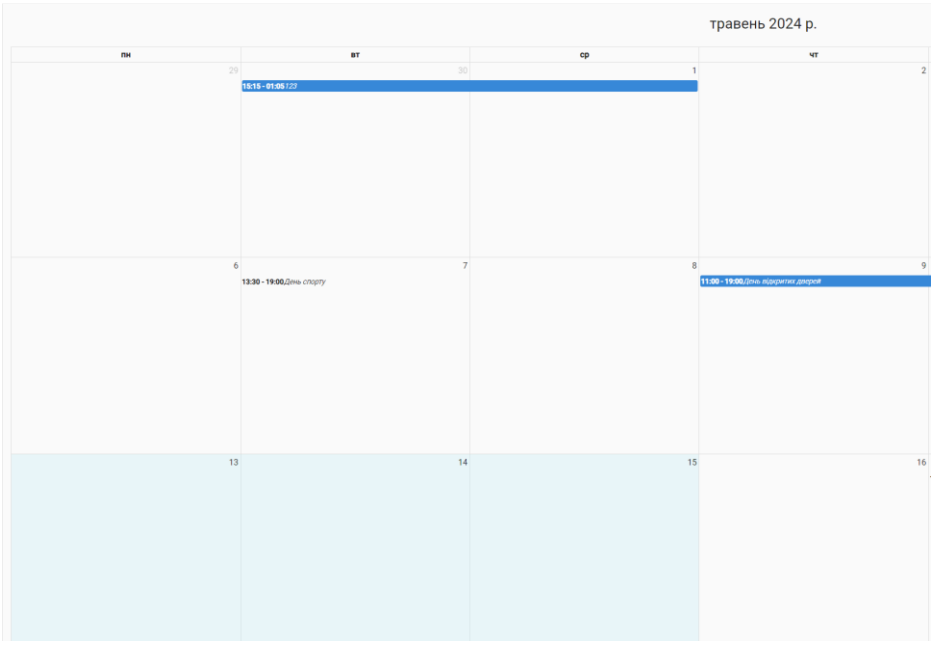


Рисунок 4.8 — Вибір дат для створення події

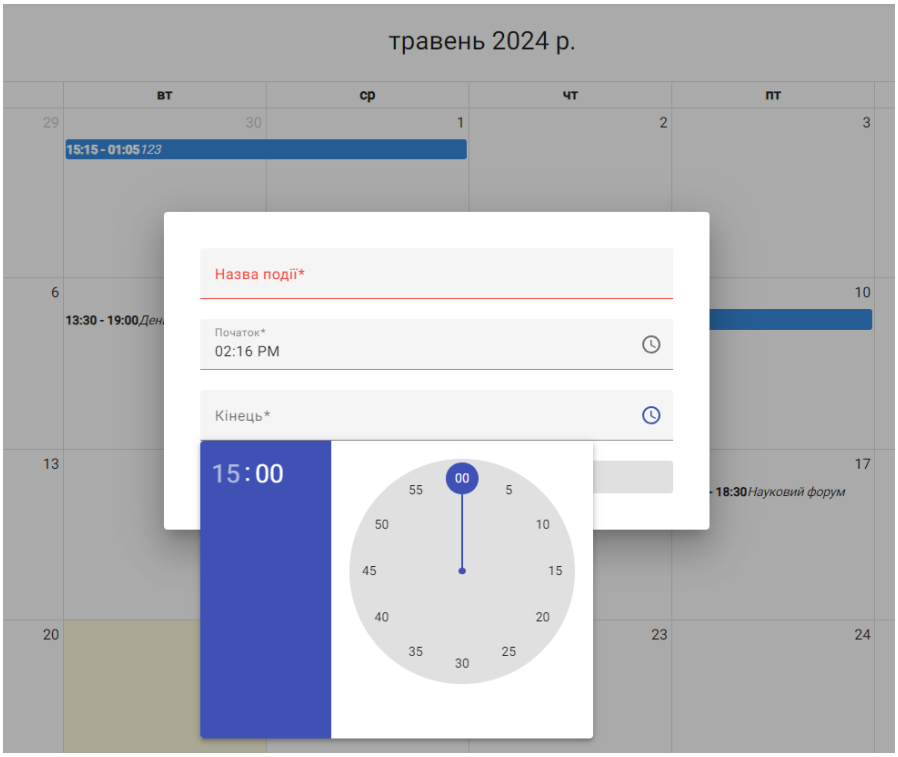


Рисунок 4.9 — Створення нової події

Для перегляду навчально-методичних матеріалів необхідно перейти на відповідну сторінку скориставшись панеллю навігації.

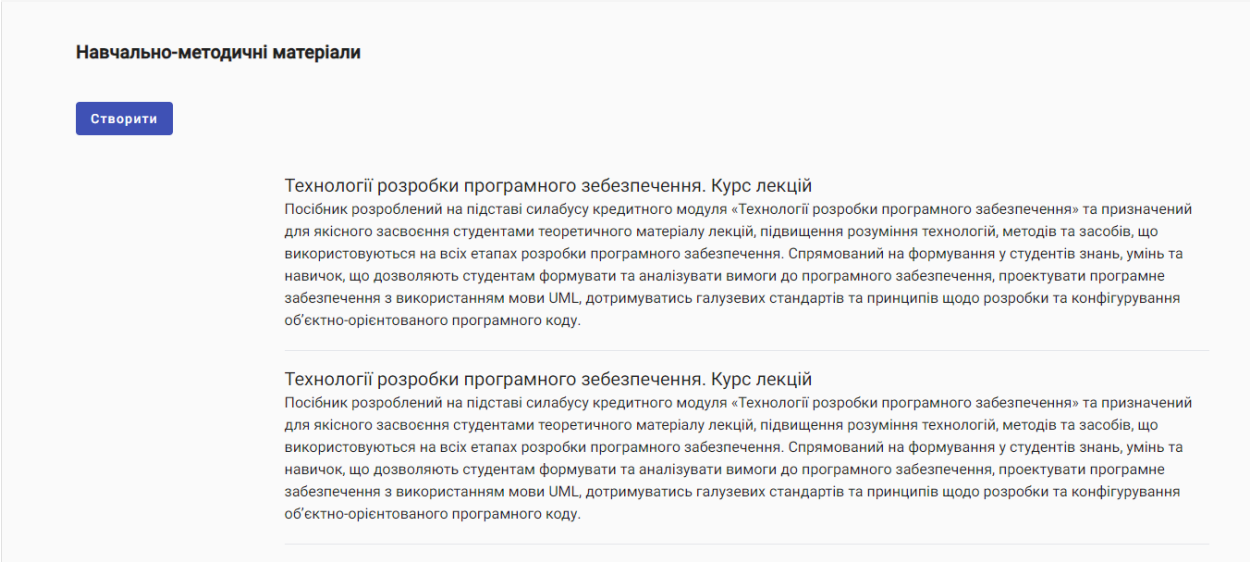


Рисунок 4.10 — Список навчально-методичних матеріалів

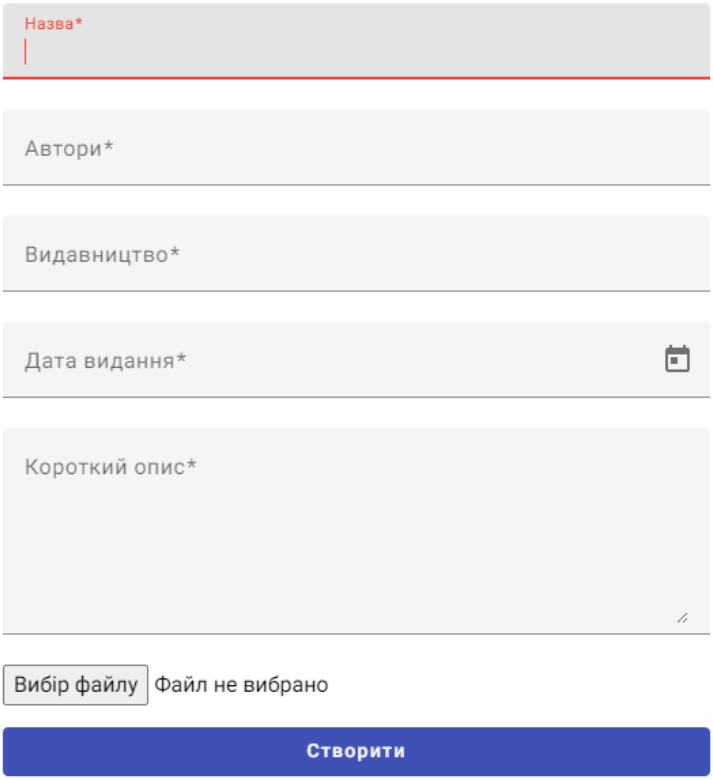


Рисунок 4.11 — Створення нового навчально-методичного матеріалу

Натиснувши на будь-який матеріал зі списку, користувач потрапляє на сторінку з більш детальним описом матеріалу, а також матиме змогу завантажити його.

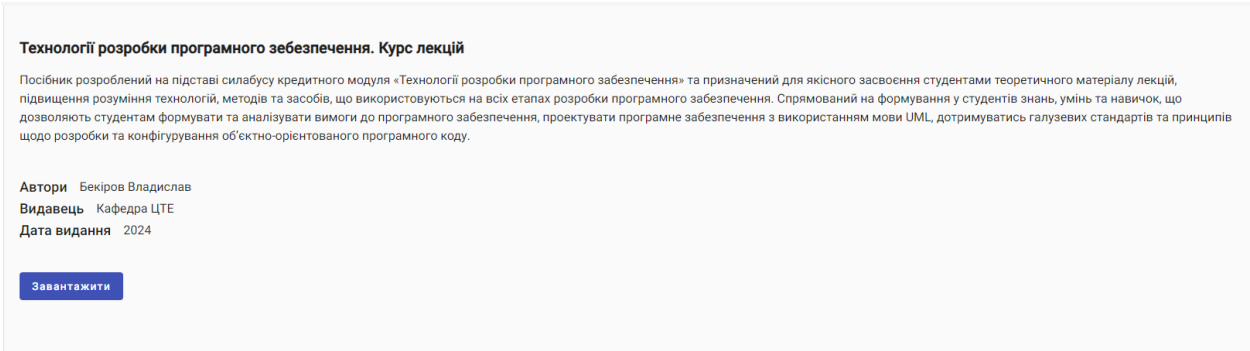


Рисунок 4.12 — Навчально-методичний матеріал

Кожен користувач має змогу скористуватися формою зв’язку зі студентською радою щоб запропонувати свої ідеї щодо поліпшення навчального процесу або поскаржитись на проблеми.

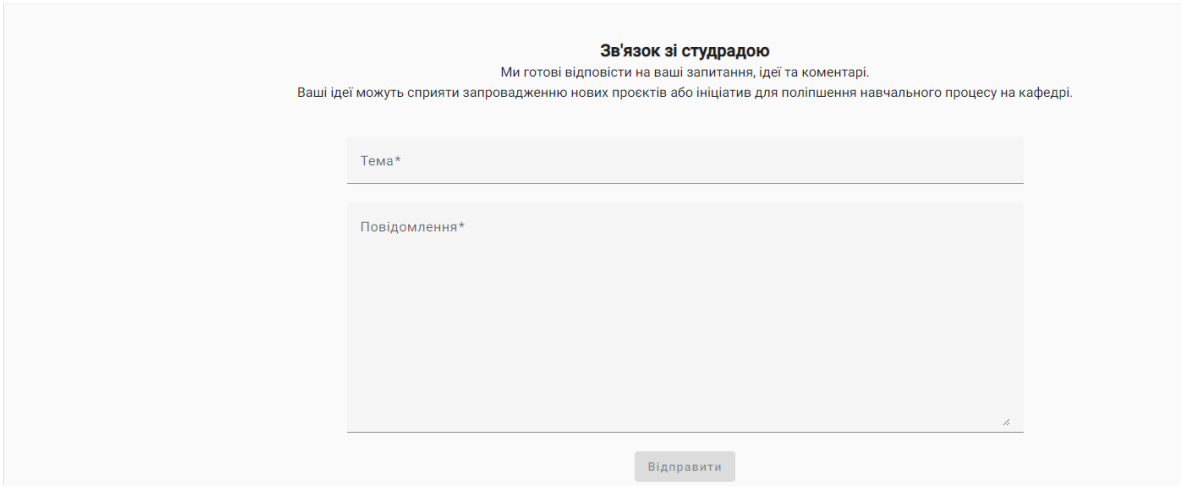


Рисунок 4.13 — Форма зв’язку зі студентською радою

Маючи роль “Представник студентської ради” або “Адміністратор” можна потрапити на сторінку де будуть показані усі заяви разом з їх авторами.

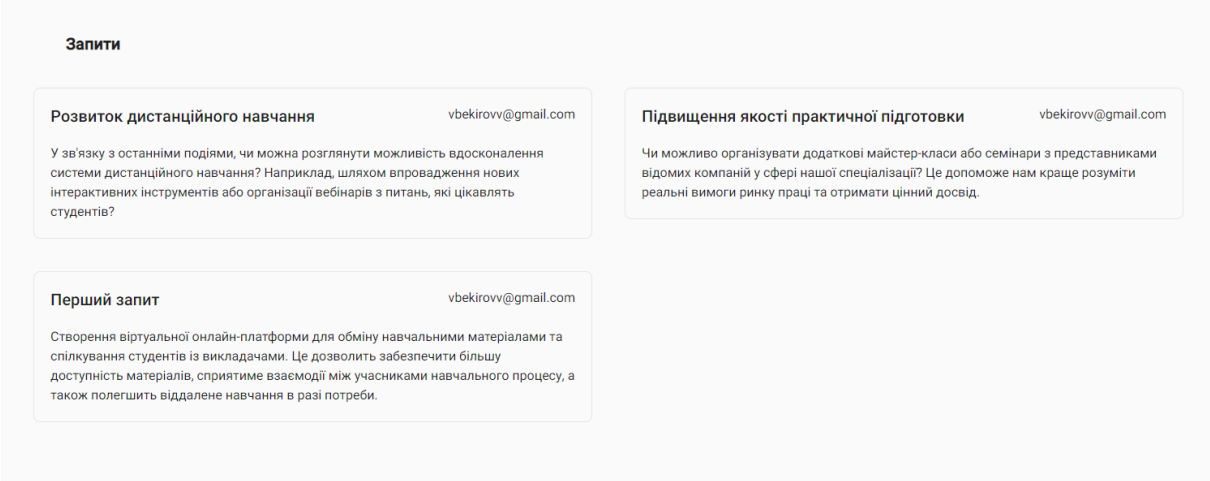


Рисунок 4.14 — Запити до студентської ради

Також розроблений веб-портал має панель адміністратора. На цій сторінці

можна зареєструвати в системі нові спеціальності, групи та користувачів. Також є можливість подивитись статистику по кількості студентів за спеціальностями або типом навчання.

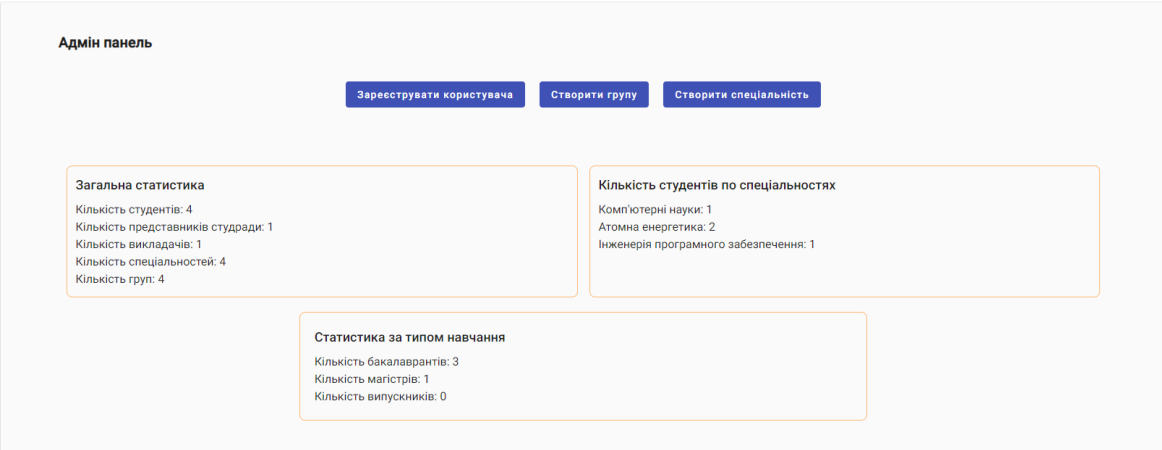


Рисунок 4.15 — Панель адміністратора

При натисканні на кнопку “Створити спеціальність” відкривається вікно, де можна ввести назву та код нової спеціальності.

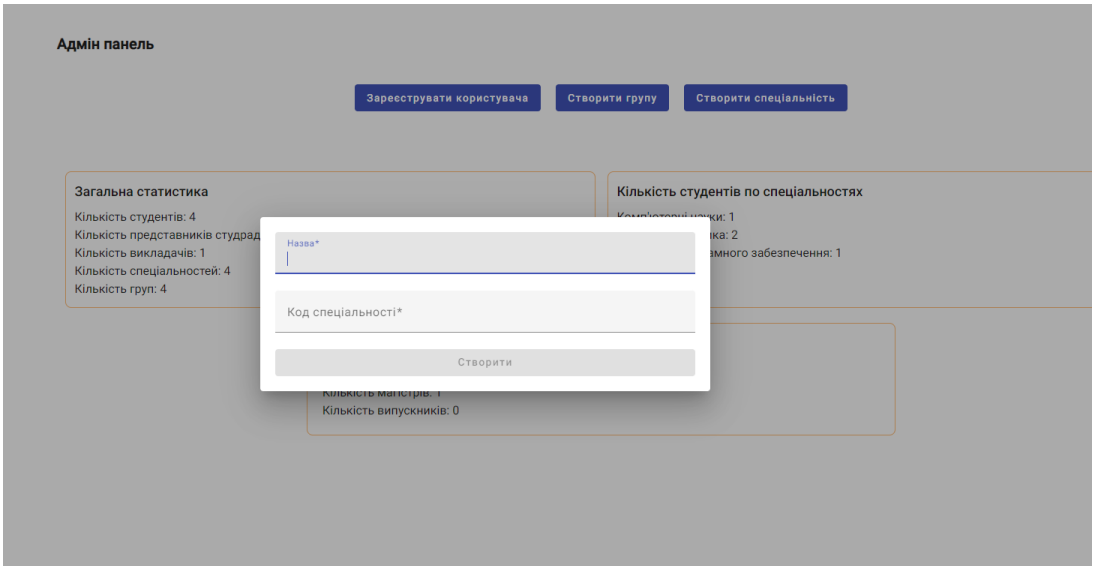


Рисунок 4.16 — Створення спеціальності

При натисканні на кнопку “Створити групу” відкривається схоже на

попереднє вікно, де можна ввести дані нової групи. До обов’язкових полів входять назва, рік початку навчання, спеціальність і тип групи.

Після цього система перевірить введені дані на коректність і, у разі успіху, створить нову групу.

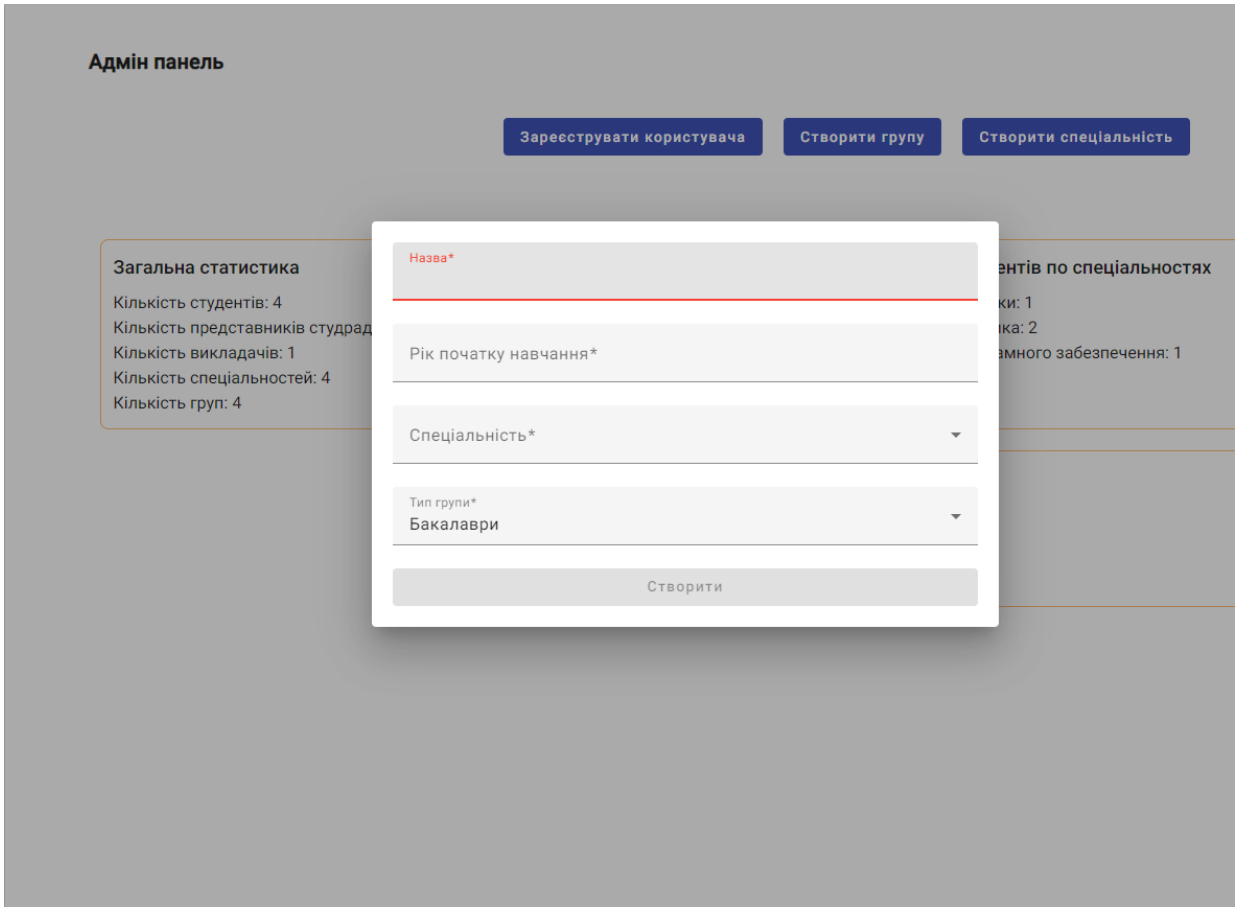


Рисунок 4.17 — Створення групи

Також можна зареєструвати нового користувача в системі, натиснувши відповідну кнопку. Після цього відкриється вікно у якому необхідно вказати пошту, пароль, ПІБ і дату народження. Після введення цих даних потрібно натиснути кнопку "Створити", після чого система автоматично створить обліковий запис для нового користувача.

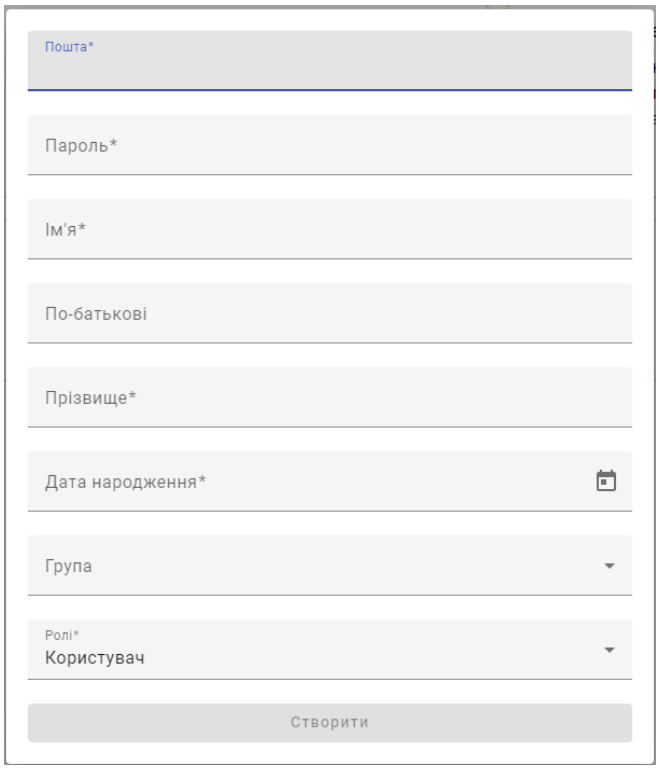


Рисунок 4.18 — Реєстрація нового користувача

Також можна обрати групу та ролі користувача з відповідних випадаючих списків.

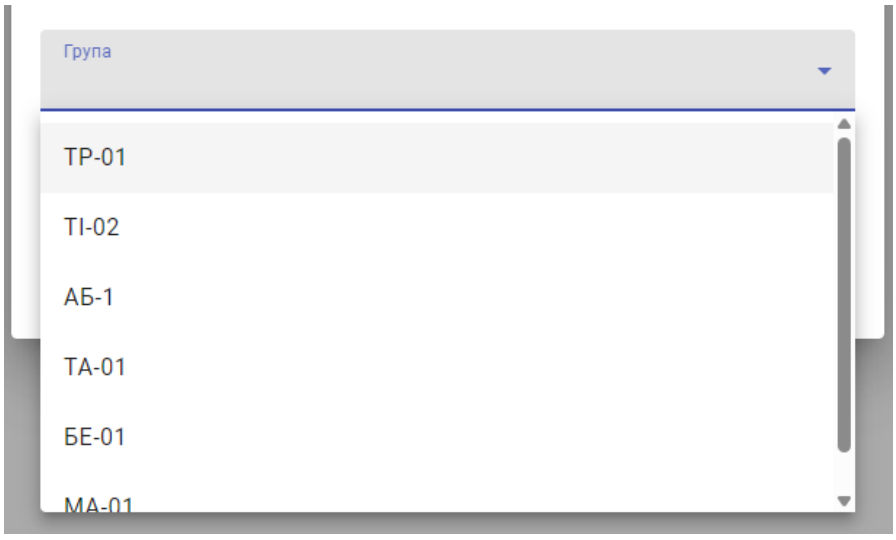


Рисунок 4.19 — Вибір групи

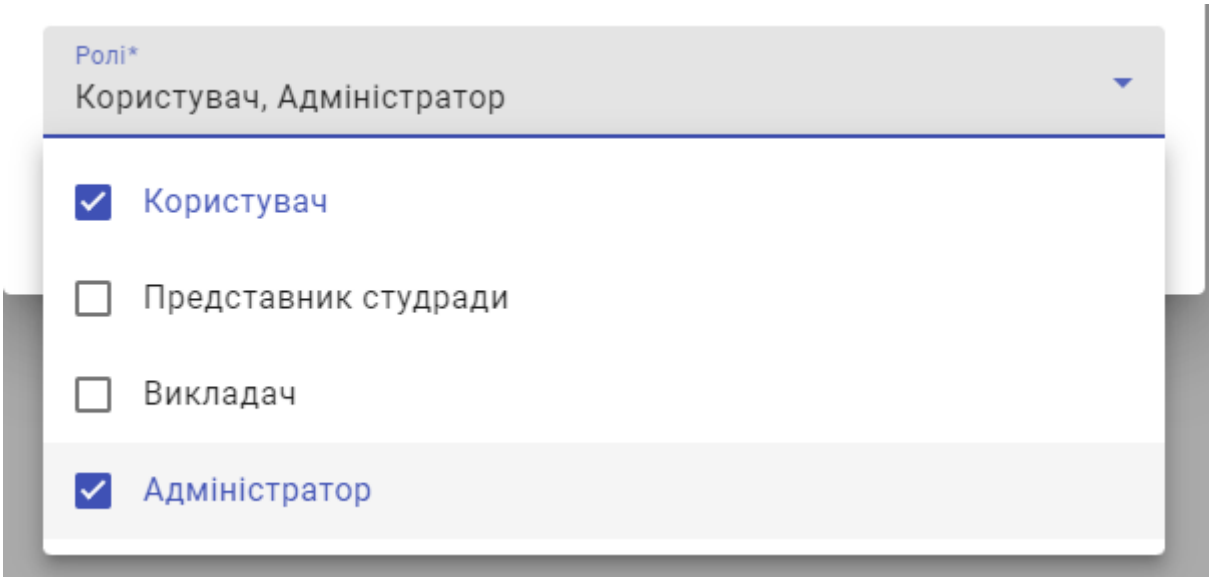


Рисунок 4.20 — Вибір ролей

Для покращення користувацького досвіду, у разі успішно виконаної операції або виникнення помилки, веб-портал відображає відповідне повідомлення.

Рисунок 4.21 – Результат реєстрації користувача

Це повідомлення автоматично зникне через 3 секунди, але його можна прибрати самостійно натиснувши відповідну кнопку.

ВИСНОВКИ

У даній роботі було розроблено інформаційний веб-портал для студентів кафедри. Створений портал забезпечує централізований доступ до навчальних матеріалів, новин, оголошень та інших ресурсів, необхідних для успішного навчання.

У ході проектування було проаналізовано існуючи аналоги та враховано їх недоліки. На основі цього було побудовано систему, що складається з бази даних, серверу та веб-застосунку. Розроблена система враховує недоліки існуючих аналогів та додає багато оригінальних функцій.

Для створення програмного коду системи були використано сучасні технології, такі як: PostgreSQL, ASP.NET, Angular, TailwindCSS та інші. Завдяки цим технологіям було реалізовано такі функції:

- система має декілька типів ролей. В залежності від ролі користувач має доступ до різного контенту на веб-порталі;
- користувачі мають доступ до новин кафедри, оголошень, календарю подій;
- адміністратор має доступ до інформаційної панелі, де він може реєструвати нових користувачів, завантажувати новий контент на веб-портал або переглядати статистику по різноманітних параметрах успішності кафедри.

Велику увагу було приділено тестуванню розробленої системи і огляду взаємодії користувача з нею.

В програмному коді порталу закладена можливість легко масштабувати існуючий застосунок щоб додавати нові функції що забезпечить його довгострокову актуальність та відповідність вимогам часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Department of Computer Science and Technology. URL: https://www.cst.cam.ac.uk/.

2. Robert C. Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design, Pearson 2017. 432 p.

3. PostgreSQL: The world's most advanced open source database. URL: https://www.postgresql.org/.

4. Regina Obe. PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database. O'Reilly Media 2017. 312 p.

5. pgAdmin - PostgreSQL Tools. URL: https://www.pgadmin.org/.

6. A tour of the C# language. URL: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview.

7. Entity Framework documentation hub. URL: https://learn.microsoft.com/en-us/ef/

8. Smith J. Entity Framework Core in Action, Second Edition. Manning., 2021. 1285 с.

9. APIs with ASP.NET Core. URL: https://dotnet.microsoft.com/en-us/apps/aspnet/apis.

10. Create Web APIs with ASP.NET Core. Microsoft Learn: веб-сайт. URL: https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-8.0

11. Seq – centralized structured logs. URL: https://docs.datalust.co/docs/an-overview-of-seq.

12. Stefan Baumgartner. TypeScript Cookbook: Real World Type-Level Programming. O'Reilly Media. 419с

13. Angular Understanding binding. URL: https://angular.io/guide/binding-syntax.

14. Angular material UI component library. URL: https://material.angular.io/.

15. A guide to configuring and customizing your Tailwind installation. URL: https://tailwindcss.com/docs/configuration.

ДОДАТОК А

ТЕКСТ ПРОГРАМНОГО МОДУЛЯ

«ПРОГРАМНА РЕАЛІЗАЦІЯ ЛОГІКИ ДОСТУПУ ДО БАЗИ ДАНИХ»

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ІАТЕ_ЦТЕ_ТР-01_24Б

Аркушів 6

Київ-2024

Програмні засоби:
– мова програмування – C#;
– об’єктно-реляційний маппер – Entity Framework Core

```
using Microsoft.EntityFrameworkCore;
using Portal.Domain.Entities;
using Portal.Infrastructure.Persistence.Configurations;

namespace Portal.Infrastructure.Persistence.DataAccess;

public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
    {
    }

    public DbSet<User> Users { get; set; } = null!;

    public DbSet<Specialty> Specialties { get; set; } = null!;

    public DbSet<Group> Groups { get; set; } = null!;

    public DbSet<NewsRecord> News { get; set; } = null!;

    public DbSet<Announcement> Announcements { get; set; } = null!;

    public DbSet<StudentCouncilRequest> StudentCouncilRequests { get; set; } = null!;
```

```
        public DbSet<CalendarEvent> CalendarEvents { get; set; } = null!;

        public DbSet<StudyMaterial> StudyMaterials { get; set; } = null!;

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);


modelBuilder.ApplyConfigurationsFromAssembly(typeof(UserConfiguration).Assembl
y);
        }
    }

    namespace Portal.Infrastructure.Persistence.DataAccess;

    public interface IRepository<TEntity> where TEntity: class
    {
        IQueryable<TEntity> GetAll();

        Task<List<TEntity>> GetAllAsync();

        ValueTask<TEntity?> GetByIdAsync(params object?[]? keyValues);

        Task CreateAsync(TEntity entity);

        Task UpdateAsync(TEntity entity);

        Task DeleteAsync(params object?[]? keyValues);
    }
```

```
using Microsoft.EntityFrameworkCore;

namespace Portal.Infrastructure.Persistence.DataAccess;

public class Repository<TEntity> : IRepository<TEntity> where TEntity: class
{
    private readonly ApplicationDbContext _dbContext;
    private readonly DbSet<TEntity> _dbSet;

    public Repository(ApplicationDbContext dbContext)
    {
        _dbContext = dbContext;

        _dbSet = _dbContext.Set<TEntity>();
    }

    public IQueryable<TEntity> GetAll()
    {
        return _dbSet;
    }

    public Task<List<TEntity>> GetAllAsync()
    {
        return _dbSet.ToListAsync();
    }

    public ValueTask<TEntity?> GetByIdAsync(params object?[]? keyValues)
    {
        return _dbSet.FindAsync(keyValues);
```

```
    }

    public Task CreateAsync(TEntity entity)
    {
        _dbSet.Add(entity);

        return _dbContext.SaveChangesAsync();
    }

    public Task UpdateAsync(TEntity entity)
    {
        _dbSet.Update(entity);

        return _dbContext.SaveChangesAsync();
    }

    public async Task DeleteAsync(params object?[]? keyValues)
    {
        var entity = await _dbSet.FindAsync(keyValues);

        if (entity is not null)
        {
            _dbSet.Remove(entity);

            await _dbContext.SaveChangesAsync();
        }
    }
}

using Microsoft.EntityFrameworkCore;
```

```
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Portal.Infrastructure.Persistence.DataAccess;

namespace Portal.Infrastructure.Extensions;

public static class DatabaseExtensions
{
    public static IServiceCollection AppendDatabase(this IServiceCollection services, IConfiguration configuration)
    {
        var connectionString = configuration.GetConnectionString("DefaultConnection");

        services.AddDbContext<ApplicationDbContext>(opt => opt.UseNpgsql(connectionString));

        return services;
    }
}

using Portal.Domain.Entities;
using Portal.Infrastructure.Model;

namespace Portal.Infrastructure.Extensions;

public static class MapExtensions
{
    public static UserViewModel Map(this User user)
    {
```

```
    return new UserViewModel
    {
        Id = user.Id,
        Email = user.Email,
        FirstName = user.FirstName,
        MiddleName = user.MiddleName,
        LastName = user.LastName,
        DateOfBirth = user.DateOfBirth,
        Group = user.Group is not null ? user.Group.Map() : null,
        Roles = user.Roles,
    };
}

public static GroupViewModel Map(this Group group)
{
    return new GroupViewModel
    {
        Id = group.Id,
        Name = group.Name,
        Year = group.Year,
        Specialty = group.Specialty.Map()
    };
}
```