math_2024_055

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Факультет математики та інформатики
Кафедра математичного моделювання

РОЗРОБКА АДМІНІСТРАТИВНОЇ ЧАСТИНИ З ВИКОРИСТАННЯМ ЗОВНІШНЬОГО WebAPI ДЛЯ СТВОРЕННЯ РОЗКЛАДУ

Кваліфікаційна робота

Рівень вищої освіти – перший (бакалаврський)

Виконав:
студент 4 курсу, 407 групи
Урсулян Андрій Георгійович
Керівник:
кандидат фізико-математичних наук,
доцент Горбатенко М.Ю.

До захисту допущено
на засіданні кафедри
протокол № 17 від 6 червня 2024 р.
Зав. кафедрою _____ проф. Черевко І.М.

Чернівці – 2024

АНОТАЦІЯ

У роботі досліджено важливість наявності панелі адміністратора, для ефективного управління розкладом занять. Детально описано основні функції, які вона має забезпечувати, а саме: редагування, видалення, додавання розкладу, керування ресурсами, оповіщення користувачів, тощо. Особлива увага приділяється використанню Blazor WebAssembly, котрий безпосередньо використовуються для реалізації веб-додатку на мові C#. Результатом роботи є реалізована адмін панель, яка відповідає всім потребам керування розкладом, та має потенціал для подальшого розвитку та реалізації додаткових функції відповідно до побажань та зауважень користувачів.

Ключові слова: C#, Blazor, WebAssembly, Адміністративна панель, веб-додаток, веб-сайт, ASP.NET, JWT, WebAPI.

ABSTRACT

The study investigates the importance of having an administrator panel for effective schedule management. It details the main functions it should provide, such as editing, deleting, adding schedules, managing resources, notifying users, and more. Special attention is given to the use of Blazor WebAssembly, which is directly used to implement the web application in C#. The result of the work is an implemented admin panel that meets all the needs of schedule management and has the potential for further development and implementation of additional features according to user wishes and feedback.

Key words: C#, Blazor, WebAssembly, Administator panel, web application, website, ASP.NET, JWT, WebAPI.

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

_____ А.Г. Урсулян

ЗМІСТ

ВСТУП

Суспільству, у якому ми живемо, притаманно багато факторів, але одним із найголовніших є швидке розповсюдження нових інформаційних технологій, зокрема адаптація до конкретних умов та факторів, які є на даний момент. Значний вплив інноваційних технологій зараз спостерігається у сфері дистанційного навчання, де мобільність, зручність та актуальність грають ключову роль.

Умовою невпинного розвитку онлайн навчання в сучасних умовах – це розробка, впровадження та реалізація інноваційних веб сторінок, та забезпечення відповідної якості продукту. У реаліях дистанційного навчання інформація про актуальний розклад занять, та зручність його редагування стає ключовим фактором у зручності та оптимізації буденності.

Мета написання кваліфікаційної роботи – дослідження та створення адмін панелі з використанням .NET та Blazor WebAssembly

Завдання:

- описати історію та функціонування адмін панелей;
- розглянути схожі проекти;
- описати застосування фреймворку ASP.NET;
- виявити особливості використання в проєкті Blazor WebAssembly
- розробити адмін панель з використанням ASP.NET та Blazor WebAssembly та описати роботу програми.

РОЗДІЛ 1. СПЕЦИФІКА РЕАЛІЗАЦІЇ АДМІН ПАНЕЛЕЙ

1.1. Історія та функціонування адмін панелей

Адміністративні панелі, є невід'ємною частиною майже кожного сайту та програми. Вони призначені для управління сайтом, або програмою, надаючи адміністратору зручний та інтуїтивний інтерфейс для того щоб він міг з легкістю додати, видалити, або редагувати вміст сторінки, керувати користувачами, переглядати статистику та інші подібні цим функції.

Історія адмін-панелей тягнеться з початку розвитку веб індустрії в загальному. Спочатку адміністрування сайтів відбувалося через написання коду вручну, або за допомогою простих текстових інтерфейсів. Проте, з розвитком технологій у сфері веб індустрії, та зростання масштабності та складності веб-проектів стало очевидним, що потрібно створювати та впроваджувати технологічніші та потужніші інструменти для адміністрування.

В загальному, історію адмін-панелей можна розподілити на декілька проміжків (90-ті, початок 2000-х, Середина 2000-х, кінець 00-х та початок 10-х, безпосередньо 10-ті роки, та 2020-ті), розберемо кожен з них детальніше.

1.1.1. 1990-ті

Перші адміністративні-панелі були досить простими, їх в загальному адмін-панелями важко назвати, адже на перших веб-сайтах управління контентом відбувалось безпосередньо через редагування HTML-файлів на сервері, що, погодьтесь, викликало певний дискомфорт. Також у цей період почались з'являтись перші серверні технології, такі як CGI та Perl, які дозволяли створювати динамічні веб-сайти і примітивні адмін-панелі для редагування контенту, що було вже набагато зручніше, ніж редагування через корінні файли веб-сторінки.

1.1.2. Початок 2000-х

На початку 00-х активного розвитку почала набирати так звана платформа LAMP (Linux, Apache, MySQL, PHP), адже вона набагато сильніше прискорювала роботу над веб-додатком, та безпосередньо над створенням та налаштуванням адмін-панелі для нього. Також слід зазначити що цей період можна відзначити ще й появою Content Management Systems, або ж CMS. Виникли перші системи управління контентом, такі як WordPress, Joomla, Drupal. Вони так би мовити мали по замовчуванню базові адмін панелі для управління контентом.

1.1.3. Середина 2000-х

Поява візуальних редакторів, та їх плавна інтеграція в робочий процес, робить управління проектом більш доступним для користувача без технічних навичок. До найпопулярніших на той час візуальних редакторів, можна віднести такі як TinyMCE, CKEditor, тощо. Також варто зазначити, що творці CMS не стояли на місці, та активно модернізували свій проєкт, що призвело до збільшення його потужності, з’являються нові можливості, такі як, управління користувачами, правами доступу, ролями, тощо.

1.1.4. Кінець 2000-х – Початок 2010-х

Революція у сфері адмін-панелей, поява на ринку JavaScript та AJAX, що призвело до кардинального збільшення функціоналу, і як результат, поява фреймворків та бібліотек, таких як JQuery AngularJS, котрі значно полегшили розробку складних інтерфейсів, де від адміністратора потребують значний функціонал в роботі з веб-сервісом. Використання AJAX (Asynchronous JavaScript and XML) дозволило створювати більш інтерактивні адмін панелі. AJAX дозволяв взаємодіяти з сервером без необхідності перезавантаження сторінки, що значно покращило користувацький досвід.

1.1.5. 2010-ті

Фреймворки такі як AngularJS, React, Vue.JS, дозволили створювати одно-сторінкові додатки (SPA(Single Page Application)). В ньому весь контент динамічно завантажується на одній сторінці, що робить роботу з адмін панелями більш швидкою та зручною. Також варто відзначити що в цей період популярності почав набирати так званий API-first підхід. Він передбачає розробку бекенду як набору API, які потім можуть бути використані для створення різних фронтенд інтерфейсів. Це дозволяє розробникам створювати більш гнучкі та масштабовані системи. Також, завдяки тому що стрімку популярність почали набирати мобільні пристрої, виникла потреба створювати окрему версію і для них. Звідси й виникли перші мобільні адмін-панелі, які дозволили керувати контентом з будь якого місця. Вони на сьогоднішній день стали стандартом для багатьох сучасних CMS.

1.1.6. 2020-ті

Спочатку декілька слів про No-code та Low-code платформи. До них належать такі платформи як Webflow, Wix, Bubble, тощо. Вони дозволяють користувачам створювати веб-застосунки з адмін панелями без необхідності писати код, свого роду конструктор. Це значно збільшило аудиторію, яка може створювати та керувати власним веб-застосунком. Також варто зазначити що велику увагу почали приділяти покращенню користувацького інтерфейсу та досвіду. Це включає створення адаптивного дизайну, який працює на різних пристроях, а також впровадження сучасних дизайн-принципів для забезпечення інтуїтивного та зручного управління для користувача.

Питання безпеки на сьогоднішній день є одним із ключових у сфері адміністративних панелей. Через зростання кібератак питання безпеки стало критично важливим. Сучасні адміністративні-панелі впроваджують складні механізми аутентифікації, наприклад, двофакторна аутентифіція, шифрування

даних, регулярні оновлення та інші заходи для захисту від несанкціонованого доступу до особистих даних користувачів, або розробників.

1.2 Майбутні перспективи

Адміністративні панелі в перспективі можуть досягти ще більших планок, враховуючий стрімкий розвиток ШІ (штучного інтелекту), машинного навчання, голосових інтерфейсів. та збільшення попиту на мобільність.

Інтеграція штучного інтелекту та машинного навчання може значно покращити автоматизацію та персоналізацію контенту на веб-сторінках. Штучний інтелект може допомагати в аналізі даних, беручи до уваги статистику, і виходячи з цього автоматично адаптувати контент на сторінці для користувачів, виходячи з їх побажань та інтересів.

Голосові інтерфейси з розвитком технологій розпізнання голосу можуть стати наступним кроком в еволюції роботи адміністративних панелей, дозволяючи користувачу управляти контентом за допомогою голосових команд.

З подальшим зростанням використання мобільних пристроїв, з’являється потреба в адаптації буденної роботи адміністратора під сучасний лад. Саме тому розробка компактних, зручних, та функціональних адмін панелей, які орієнтовані на використання за допомогою мобільних пристроїв буде залишатися важливим напрямком у цій сфері.

Адміністративні панелі продовжуватимуть свій розвиток, інтегруючи всі новітні технології та адаптуючись до змінюваних потреб користувачів, не забуваючи при цьому про безпеку, зручність та ефективність в управлінні контентом.

1.3. Огляд найрозповсюдженіших адміністративних панелей, що є у відкритому доступі

У відкритому доступі можна знайти безліч адміністративних панелей різного рівня складності та призначення. Нижче розглянемо декілька

популярних, безкоштовних та відкритих адміністративних панелей, які можна використати для управління контентом та адміністрування веб-сайтів та веб-застосунків:

1) CMS із вбудованими адмін панелями.

До них можна віднести:

- WordPress – найпопулярніша CMS у світі, з величезною кількістю плагінів та тем. Використовується для створення блогів, новинних сайтів, корпоративних веб-сайтів та інтернет-магазинів. Завдяки своїй гнучкості та масштабованості, WordPress дозволяє легко налаштовувати дизайн і функціональність сайтів, інтегрувати різноманітні сервіси, а також забезпечує високу продуктивність і безпеку. Це робить його ідеальним вибором як для початківців, так і для досвідчених розробників.

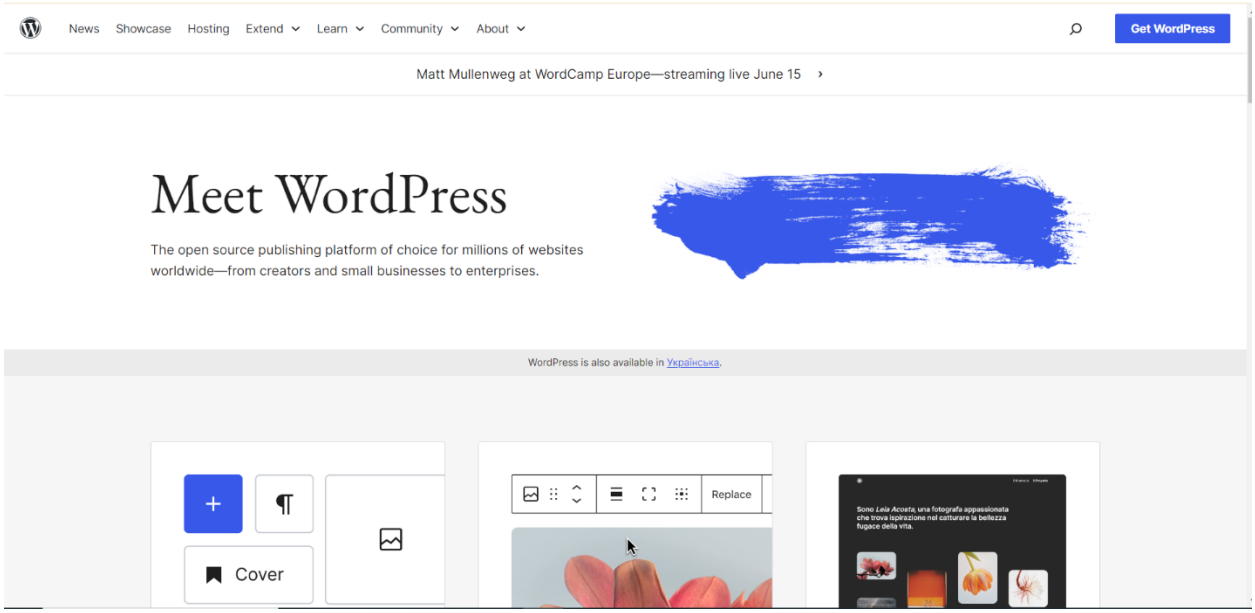

Рис. 1.1. WordPress

- Joomla – гнучка і розширювана CMS для створення складних сайтів та веб-застосунків. Використовується зазвичай для корпоративних веб-сайтів, інтернет-магазинів та соціальних мереж. Завдяки своїй модульній архітектурі, Joomla дозволяє легко додавати нові функції та налаштовувати сайти під конкретні потреби. Вона забезпечує зручний інтерфейс для управління контентом, інтеграцію з різними сервісами, а також високий рівень безпеки. Це робить Joomla чудовим вибором для реалізації проектів будь-якої складності.

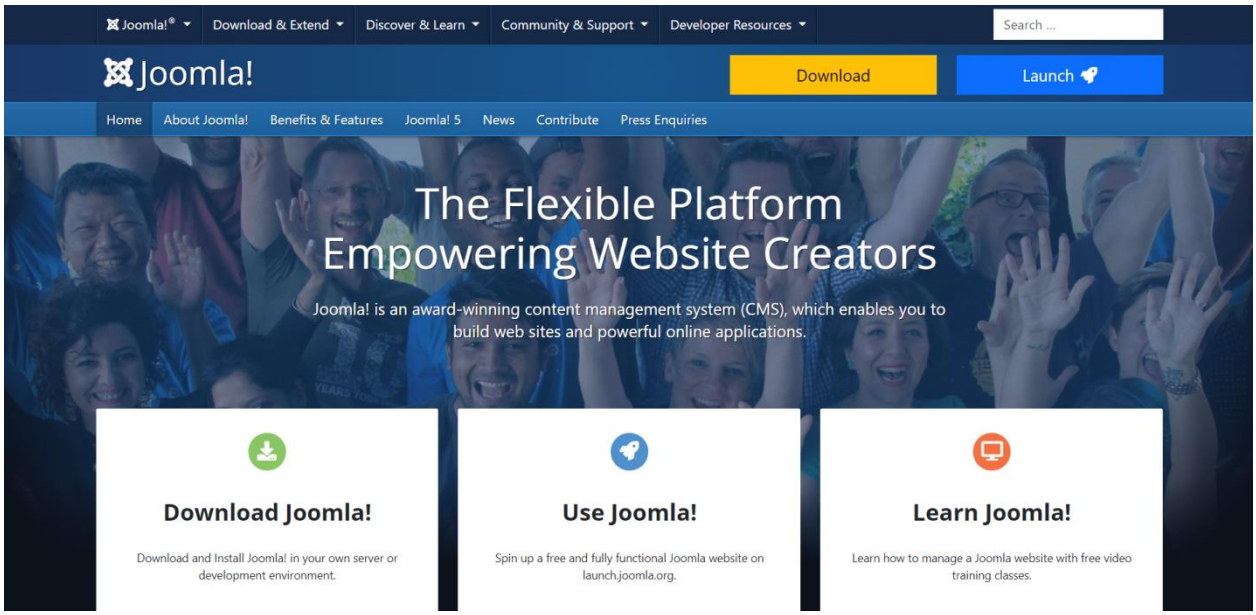

Рис. 1.2. Joomla

- Drupal – потужна та гнучка CMS, яка використовується для створення складних і багаторівневих сайтів та веб-застосунків. Часто використовується для створення соціальних мереж, корпоративних веб-сайтів, державних порталів та інших великих проектів. Завдяки своїй високій масштабованості та безпеці, Drupal дозволяє розробникам легко налаштовувати і розширювати функціональність сайтів, інтегрувати їх з різноманітними сервісами та забезпечувати зручне управління

контентом. Це робить Drupal ідеальним вибором для проектів з високими вимогами до гнучкості та надійності.

Рис. 1.3. Drupal

2) Фреймворки з вбудованими адміністративними панелями.

До них належать:

- Django Admin - це вбудована адміністративна панель у Django, популярному фреймворку мови Python, яка використовується для розробки веб-застосунків різного рівня складності. Вона надає готове рішення для швидкого створення потужних адміністративних інтерфейсів, дозволяючи легко керувати даними, користувачами та правами доступу. Django Admin інтегрується з моделями даних Django, що спрощує створення адміністративних панелей для будь-яких моделей даних у проекті. Крім того, вона підтримує розширення за допомогою сторонніх пакетів, що дозволяє розширити її функціональність за потребами конкретного проекту. Django Admin

забезпечує ефективне управління даними та користувачами, спрощуючи процес розробки і підвищуючи продуктивність розробників.

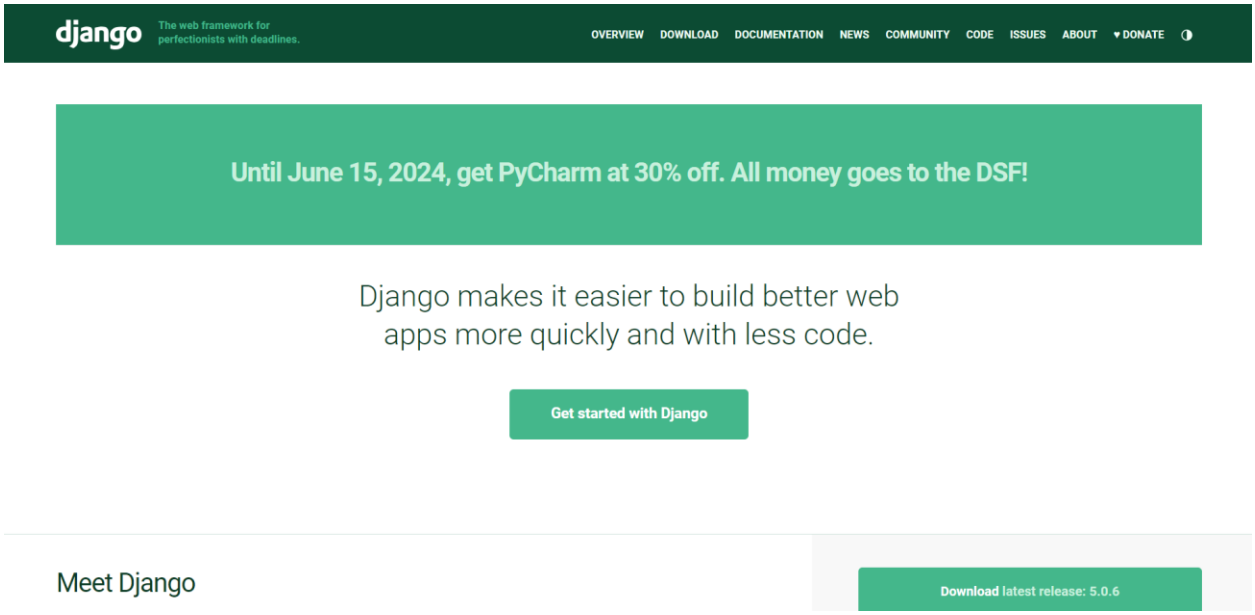

Рис. 1.4. Django Admin

- Laravel Nova - це адміністративна панель для Laravel, одного з найпопулярніших PHP-фреймворків. Вона спеціально розроблена для використання з Laravel-застосунками і надає розробникам потужний інструмент для швидкого створення адміністративних інтерфейсів. Laravel Nova дозволяє легко керувати даними, редагувати моделі, налаштовувати поля та відображати інформацію у зручному форматі. Вона забезпечує розширення за допомогою власних компонентів та сторонніх пакетів, що дозволяє розширити її функціональність за потребами конкретного проекту. Laravel Nova спрощує процес розробки адміністративних інтерфейсів і забезпечує ефективне управління даними для Laravel-застосунків.

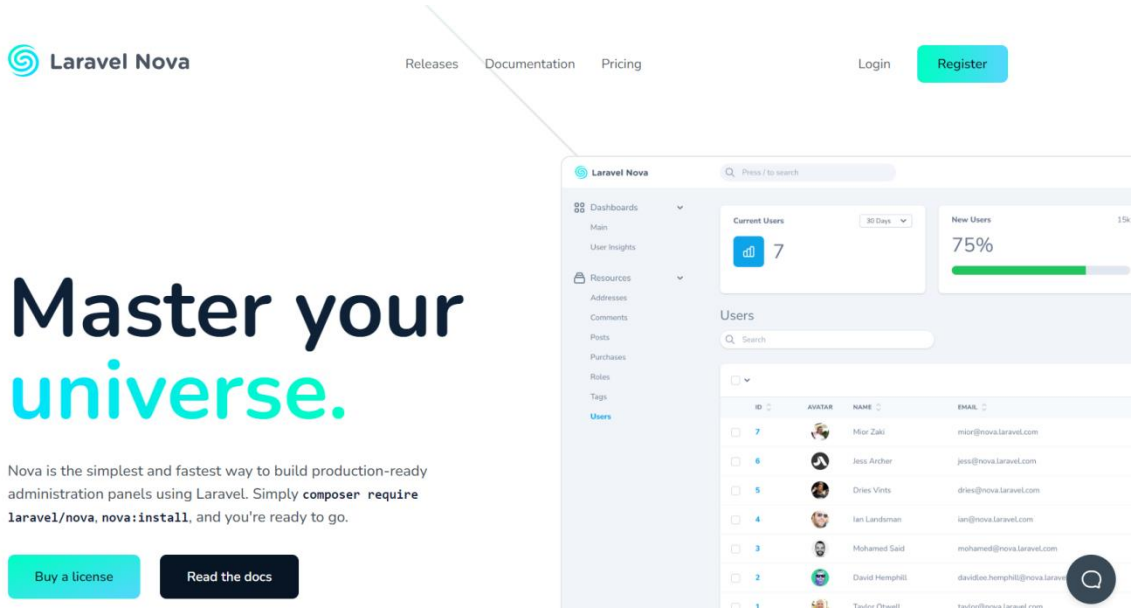

Рис. 1.5. Laraver Nova

3) Окремі адміністративні панелі.

З них можна згадати про:

- AdminLTE - це популярний шаблон адміністративної панелі, побудований на основі Bootstrap. Він ідеально підходить для будь-якої веб-сторінки або веб-застосунку, який потребує адміністративного інтерфейсу. AdminLTE пропонує чистий та сучасний дизайн, а також розширений набір компонентів і стилів, що дозволяють швидко розробляти стильні та функціональні адміністративні панелі. Його висока розширюваність і гнучкість роблять його популярним вибором для розробників, які шукають швидке та ефективне рішення для своїх проектів.

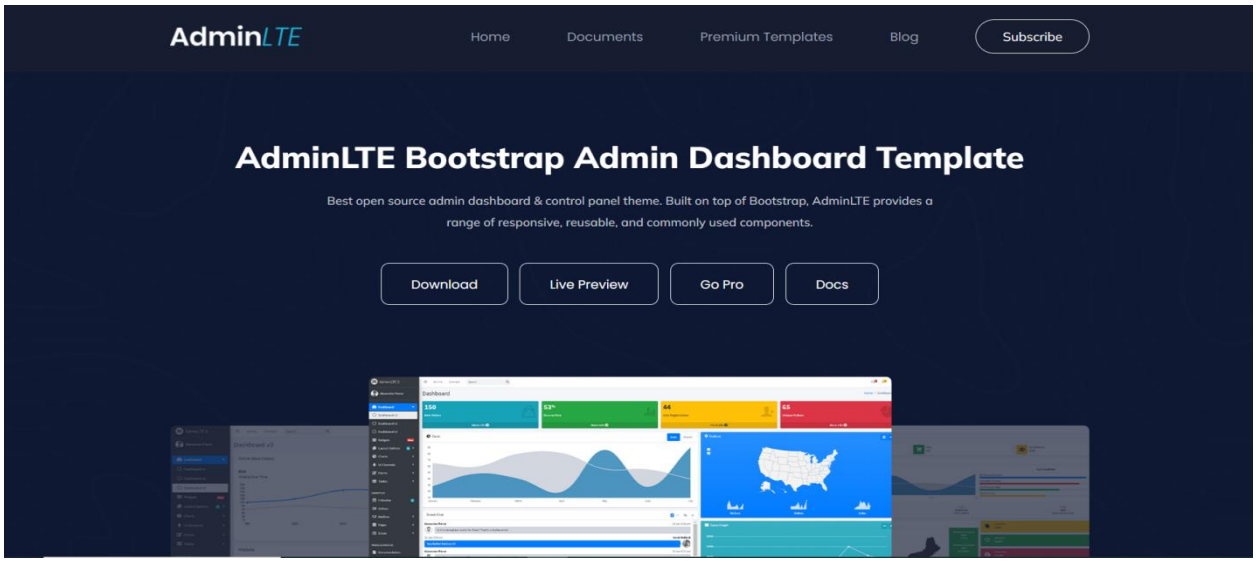

Рис 1.6. AdminLTE

- React-admin - це фреймворк для побудови адміністративних панелей, розроблений на базі React. Його використання ідеально підходить для створення веб-застосунків на основі React, які вимагають адміністративного інтерфейсу. React-admin надає розробникам гнучкий і потужний інструментарій для швидкого створення і налаштування адміністративних панелей. Він має багатий набір компонентів і можливостей, які дозволяють ефективно керувати даними, користувачами та ролями доступу. Завдяки використанню React, цей фреймворк забезпечує високу продуктивність і зручний інтерфейс для розробників, що робить його популярним вибором для створення адміністративних панелей у веб-застосунках на React.

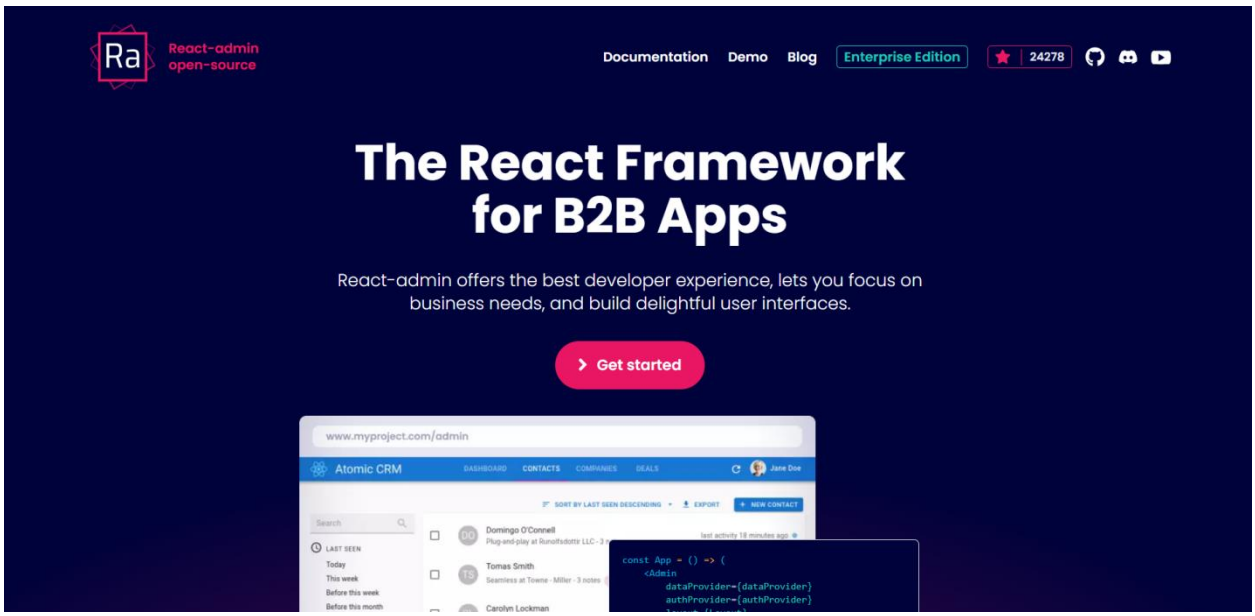

Рис. 1.7. React-admin

- CoreUI - це відкрите та безкоштовне адміністративне рішення з багатьма вбудованими функціями, що використовується для розробки веб-застосунків та інформаційних панелей. Цей інструмент надає розробникам широкий набір компонентів, які допомагають забезпечити швидку та ефективну розробку адміністративних інтерфейсів. CoreUI пропонує гнучкий та стильний дизайн, який можна легко налаштувати під потреби конкретного проекту. Він підтримує розширення за допомогою додаткових плагінів та компонентів, що дозволяє розширити

його функціональність за бажанням розробника. CoreUI є популярним вибором для розробників, які шукають швидке та зручне рішення для створення адміністративних інтерфейсів для своїх веб-застосунків та інформаційних панелей.

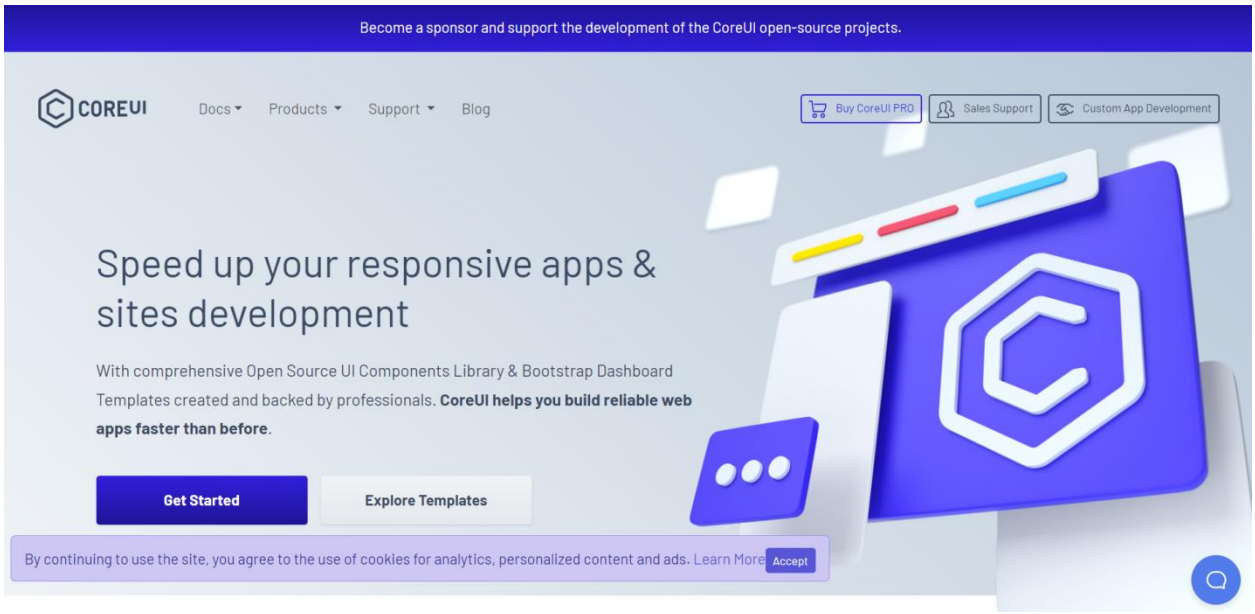

Рис. 1.8. CoreUI

Ці платформи та фреймворки пропонують багатий набір функцій для адміністраторів і розробників. Вони дозволяють не лише створювати, оптимізувати та керувати веб-застосунками і веб-сайтами будь-якої складності, але й забезпечують інтеграцію з іншими сервісами, моніторинг продуктивності, налаштування безпеки, а також зручний інтерфейс для управління користувачами і контентом. Завдяки цим інструментам можна автоматизувати рутинні завдання, підвищити ефективність роботи і швидко адаптуватися до змін у вимогах проекту.

РОЗДІЛ 2. ВИКОРИСТАНІ ТЕХНОЛОГІЇ ДЛЯ СТВОРЕНОЇ АДМІНІСТРАТИВНОЇ ПАНЕЛІ

2.1. Визначення фреймворку ASP.NET

ASP.NET — це веб-фреймворк, розроблений компанією Microsoft, який дозволяє створювати динамічні веб-додатки, веб-сайти і веб-сервіси. Він є частиною платформи .NET і надає розробникам широкий набір інструментів і бібліотек для розробки продуктивних і масштабованих веб-додатків.

Основні компоненти на особливості ASP.NET:

1) ASP.NET Core – сучасна, крос-платформова, високопродуктивна версія ASP.NET, яка може працювати на Windows, macOS і Linux.

Особливості ASP.NET Core:

- Крос-платформність: Підтримка різних операційних систем.
- Висока продуктивність: Оптимізований для швидкої роботи і низького використання ресурсів.
- Модульність: Можливість підключення тільки потрібних компонентів, що зменшує розмір і підвищує продуктивність додатків.
- Open Source: Відкрите програмне забезпечення з активною спільнотою розробників.

2) ASP.NET MVC - Архітектурний шаблон Model-View-Controller (MVC) для створення веб-додатків.

Особливості ASP.NET MVC**:**

- Розділення завдань: Відокремлює логіку додатку, користувацький інтерфейс і управління даними.
- Контрольованість: Легше тестування і відладка завдяки розділенню логіки додатку.
- Чистий код: Зручність у підтримці та розширенні додатку.

3) ASP.NET Web API - Фреймворк для створення HTTP-служб, які можуть обробляти запити з різних клієнтів, включаючи браузери та мобільні пристрої.

Особливості ASP.NET Web API:

- Легкість інтеграції: Підтримка роботи з різними клієнтами, такими як мобільні додатки, AJAX-запити та інші.
- RESTful API: Легко створювати та керувати RESTful сервісами.

4) ASP.NET Web forms – традиційний компонент ASP.NET, що дозволяє створювати динамічні веб-сторінки з мінімальним написанням коду.

Особливості ASP.NET Web forms:

- Простота у використанні**:** Зручний для розробників, які звикли до моделі "drag-and-drop".
- Швидка розробка: Можливість швидкого створення прототипів.

5) Blazor – фреймворк для створення інтерактивних веб-інтерфейсів з викристанням C# замість JavaScript.

Особливості Blazor:

- WebAssembly: використовується для виконання коду в браузері.
- Односторінкові додатки: створення сучасних односторінкових додатків з інтерактивним інтерфейсом.

Основні переваги ASP.NET:

- Висока продуктивність: Оптимізовані процеси та низький час відгуку.
- Крос-платформовість: Можливість розгортання додатків на різних операційних системах.
- Безпека: Вбудовані засоби захисту, такі як автентифікація та авторизація.
- Розширюваність: Легкість інтеграції з іншими бібліотеками та сервісами

- Сучасний стек технологій: Підтримка новітніх стандартів і технологій веб-розробки.

Використання ASP.NET:

ASP.NET підходить для різних типів веб-додатків, включаючи корпоративні портали, інтернет-магазини, соціальні мережі, API-сервіси, та інтерактивні односторінкові додатки. Завдяки своїй гнучкості, продуктивності та багатофункціональності, ASP.NET залишається одним з найбільш популярних виборів для веб-розробників по всьому світу.

2.2. Blazor WebAssembly

Blazor WebAssembly — це фреймворк від Microsoft для створення інтерактивних односторінкових веб-додатків (SPA) з використанням C# замість JavaScript. Blazor WebAssembly використовує технологію WebAssembly для виконання .NET-коду безпосередньо в браузері, що дозволяє розробникам писати клієнтську логіку на мові C#.

Основні концепції та особливості Blazor WebAssembly:

1) WebAssembly – це технологія, яка дозволяє запускати .NET код у браузері без необхідності використання плагінів, або надбудов. Blazor використовує WebAssembly для запуску .NET Runtime в браузері, що дозволяє виконувати .NET-код безпосередньо на стороні клієнта.
2) Розробка на C# - замість традиційного використання JavaScript для розробки клієнтської частини, Blazor дозволяє використовувати C# для написання інтерактивної логіки.
3) Компонентна модель – Blazor базується на компонентній архітектурі, де компоненти – це повторно використані елементи інтерфейсу.

Основні переваги Blazor WebAssembly:

- Крос-платформовість: Підтримка всіх сучасних браузерів без необхідності використання плагінів.

- Висока продуктивність: Завдяки WebAssembly, Blazor WebAssembly забезпечує високу продуктивність виконання коду.
- Скорочення обсягу коду: Використання єдиного коду для клієнтської та серверної частин зменшує дублювання логіки.
- Строга типізація: Використання C# забезпечує сильну типізацію, що знижує кількість помилок під час компіляції.
- Інтеграція з .NET екосистемою: Повна сумісність з іншими компонентами .NET, такими як Entity Framework? Dependency Injection, тощо.

Використання:

Blazor WebAssembly підходить для створення різних типів веб-додатків, включаючи корпоративні додатки, інтерактивні користувацькі інтерфейси, та навіть гри. Завдяки можливості використовувати C# для всієї логіки додатку, Blazor WebAssembly може значно спростити розробку та підтримку додатків, особливо для тих команд, які вже використовують .NET на сервері.

РОЗДІЛ 3. ОПИС РОБОТИ ПРОГРАМИ

У кваліфікаційній роботі на тему “Розробка адміністративної частини з використанням зовнішнього WebAPI для створення розкладу” було реалізовано інтерактивну адміністративну панель, завдяки якій є можливість редагувати контент на веб-сторінці.

Для реалізації поставленої задачі було використано C# та його модулі ASP.NET і Blazor WebAssembly.

Головною метою було створити адмін панель зі простим та інтуїтивно зрозумілим інтерфейсом. Було розроблено сайт для того, щоб користувачі могли зручно і швидко створювати групи, розклад, аудиторії тощо. Користуватись сайтом можуть тільки авторизовані користувачі. Передбачено 3 види авторизованих користувачів:

- Адміністатор розкладу: має змогу додавати розклад для певного факультету, доступні сторінки для перегляду груп, викладачів, аудиторій, навчальні програми та кафедр.
- Адміністратор факультету: має доступ до вищезазначених сторінок , а також може додавати, видаляти та змінювати інформацію про групи, викладачів, аудиторій, навчальних програм, кафедр.
- Адміністратор: може створювати та редагувати університети та факультети.

Передбачається, що кожен викладач матиме власну роль, будь то адміністратор розкладу, адміністратор факультету чи адміністратор університету. Адміністратором з найбільшими повноваженнями буде одна, або декілька осіб, що дозволить правильно розподілити ролі та забезпечити безпеку.

Адміністратор розкладу відповідає за створення та управління розкладами занять, адміністратор факультету контролює діяльність у межах свого факультету, а адміністратор університету має повний доступ до всієї системи та можливість змінювати глобальні налаштування. Такий розподіл обов’язків

дозволяє ефективно керувати ресурсами та забезпечувати контроль над різними аспектами роботи навчального закладу.

Доступ до створення розкладу для певного факультету здійснюється за допомогою зчитаного ідентифікатора факультету з JWT токена, який належить викладачу. Цей механізм забезпечує, що викладачі можуть створювати і редагувати розклади лише для своїх факультетів, виключаючи можливість несанкціонованих змін у розкладах інших факультетів.

JWT токен містить інформацію про роль користувача і його приналежність до певного факультету, що дозволяє системі точно визначати, які дії дозволені кожному конкретному користувачеві. Такий підхід забезпечує високий рівень безпеки та захисту даних, оскільки кожен користувач має доступ лише до тієї інформації та функцій, які відповідають його ролі.

Завдяки такому багаторівневому підходу до управління ролями та доступами, система забезпечує ефективне управління розкладами, захист даних і підвищений рівень безпеки для всіх учасників навчального процесу.

Сторінки створення розкладу:

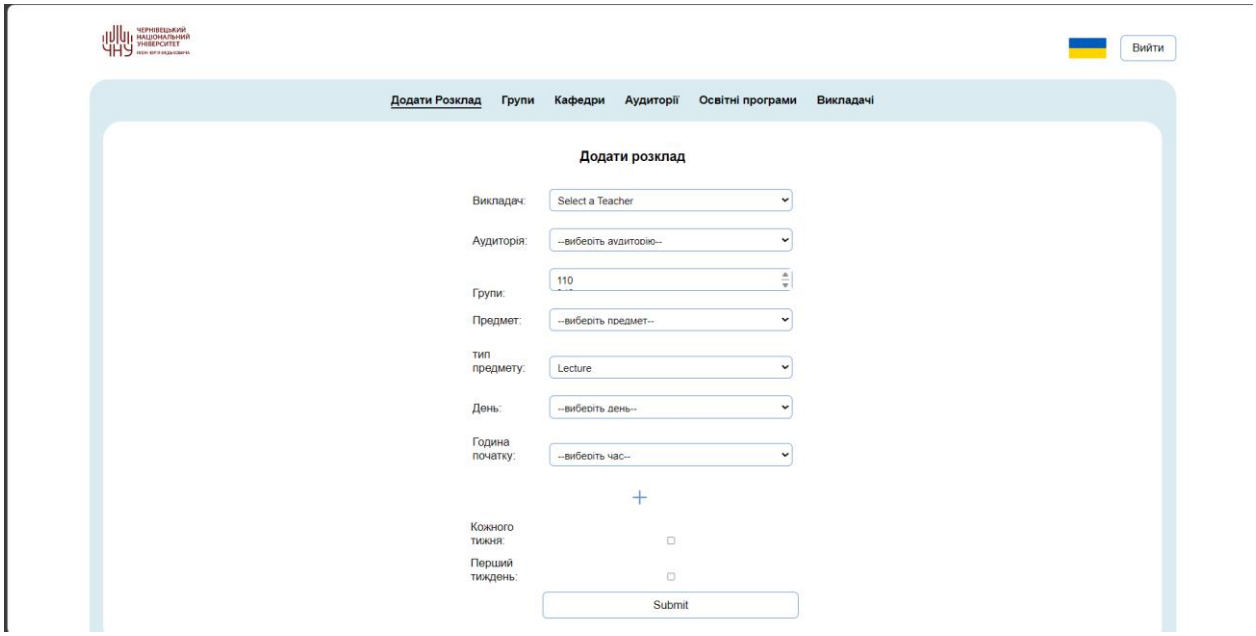

Рис 3.1. Додавання розкладу

У вкладці ‘Групи’ була реалізована можливість виконувати різні дії з групами для освітніх програм. Тепер користувачі мають змогу не лише додавати нові групи, але й редагувати наявні, вносячи зміни у їхні налаштування чи опис. Крім того, існує можливість видаляти групи, які більше не є актуальними чи потрібними. Це значно спрощує управління освітніми програмами, забезпечуючи гнучкість та зручність в організації навчального процесу.

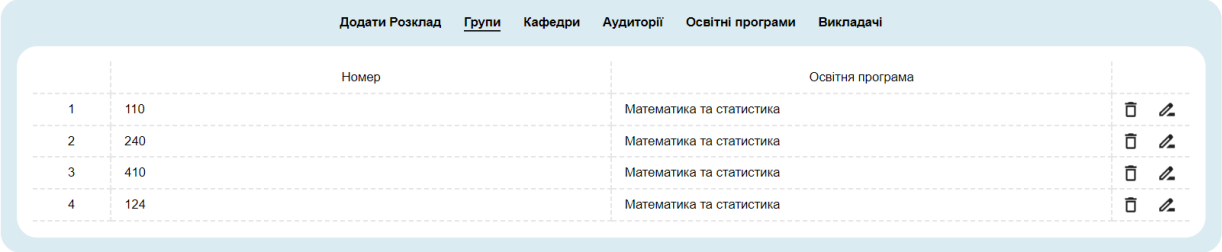

Рис. 3.2. Додавання групи

У вкладці ‘Кафедри’ була реалізована можливість виконувати різні операції з кафедрами, для освітніх програм. Тепер користувачі можуть не тільки додавати нові кафедри, але й редагувати вже існуючі, змінюючи їхню структуру, назву, або опис. Крім того, є можливість видаляти кафедри, які більше не є актуальними або необхідними. Це робить процес управління освітніми програмами набагато ефективнішим, дозволяючи краще організовувати та координувати академічну діяльність.

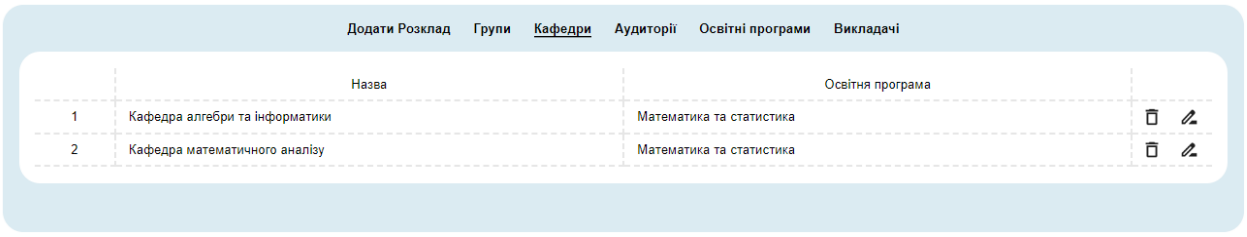

Рис. 3.3. Додавання Кафедри

У вкладці 'Аудиторії' була реалізована можливість виконувати різні дії з аудиторіями для освітніх програм. Тепер користувачі можуть додавати нові аудиторії, редагувати наявні, вносячи зміни в їхні характеристики, та видаляти ті, що більше не потрібні. Крім того, можна зазначити наявність проектора та вказати вмістимість кожної аудиторії. Це дає змогу краще розподіляти ресурси та оптимізувати використання навчальних приміщень, забезпечуючи комфортні умови для проведення занять.

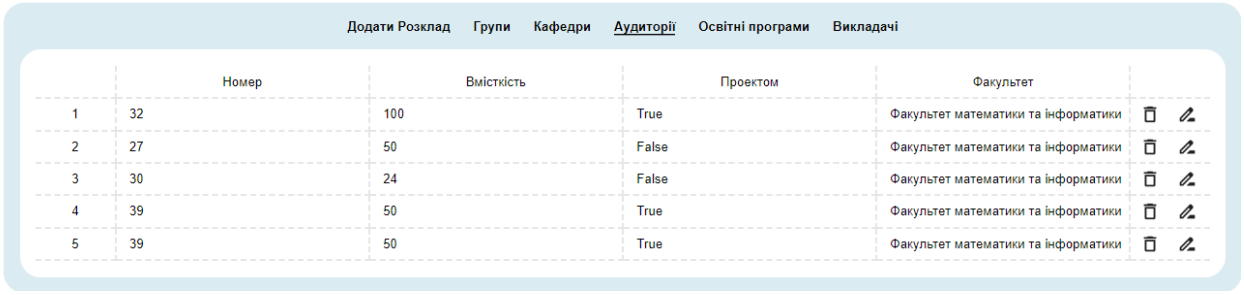

Рис. 3.4. Додавання аудиторії

У вкладці 'Освітні програми' була реалізована можливість виконувати різні операції з програмами для освітніх установ. Тепер користувачі можуть додавати нові освітні програми, редагувати їхні назви, назви спеціальностей, коди спеціальностей, спеціалізації, професійні кваліфікації, компетенції, номери сертифікатів, факультети та предмети, а також видаляти ті програми, які більше не потрібні. Це забезпечує комплексний підхід до управління освітніми програмами, дозволяючи легко оновлювати і підтримувати актуальність інформації.

Рис. 3.5. Додавання освітньої програми

У вкладці 'Викладачі' була реалізована можливість виконувати різноманітні дії з інформацією про викладачів. Тепер користувачі можуть додавати нових викладачів, редагувати їхні дані, такі як повне ім'я, звання, кафедра та електронна пошта, а також видаляти інформацію про викладачів, які більше не працюють. Це значно полегшує управління кадровими ресурсами, забезпечуючи актуальність і точність інформації про викладацький склад.

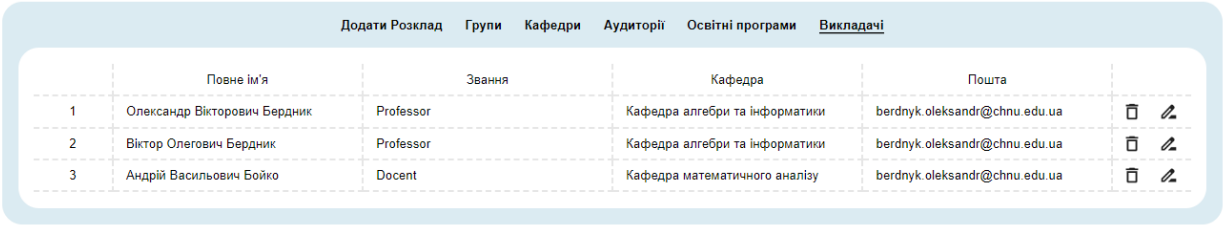

Рис. 3.6. Додавання викладачів

Було реалізовано також адаптивний дизайн, що автоматично пристосовується до різних розмірів екрану користувача. Це забезпечує зручний перегляд та використання системи на будь-яких пристроях, включаючи комп'ютери, планшети та смартфони. Адаптивний дизайн гарантує, що інтерфейс залишається зручним та ефективним навіть на пристроях з невеликими екранами.

Крім того, був забезпечений доступ до бази даних, яка підключена через WebAPI. Ця база даних використовується для зберігання всієї інформації, доданої користувачами до системи. За допомогою WebAPI користувачі можуть взаємодіяти з базою даних, додавати, редагувати та видаляти дані. Це забезпечує надійне зберігання та обробку інформації, а також дозволяє здійснювати розширені функції, такі як пошук, фільтрація та сортування даних.

Загалом, завдяки адаптивному дизайну і доступу до бази даних через WebAPI, система стає ще більш зручною та функціональною для користувачів. Ці функції дозволяють пристосовувати веб-додаток під індивідуальні потреби користувачів та забезпечують надійну та ефективну роботу системи.

ВИСНОВКИ

За останні три десятиліття у веб-технологіях відбулися досить кардинальні зміни. Розробникам довелося знайти найбільш радикальні та інтенсивні рішення вже існуючих проблем.

Адмін панелі можуть бути різних форм і розмірів, та використовуватися для різних цілей. Іноді може знадобитися інформативна панель, яка може допомогти адміністраторам зрозуміти основні метрики роботи системи, або додатку, і, спрямувати їх до прийняття ефективних рішень. Також адмін панелі можна використовувати для моніторингу та керування контентом, інтерактивного аналізу даних, або отримання зворотного зв’язку від користувачів. Вони можуть викликати інтерес до продукту серед внутрішніх команд, або дати вам уявлення про пріоритети та потреби ваших користувачів.

Для створення власної адміністративної панелі для сайту розкладу було розглянуто безліч ресурсів, які надають можливість адмініструвати сторінку, такі як: React-admin, CoreUI, Blazor, Django admin та інші. Проаналізувавши структуру та роботу цих ресурсів ми мали можливість обрати кращий з них, для реалізації власної адмін панелі.

У наш час та в наших реаліях люди не можуть жити без мобільних і веб-додатків. Зі стрімким переходом навчальних закладів на онлайн, або комбіновану форму навчання, наявність зручного розкладу, а також можливості зручно та оперативно його редагувати, є невід’ємною частиною життя. Наявність зручних та простих в розумінні фреймворків, значно спрощують ці задачі. Одними з таких фреймворків є ASP.NET, та Blazor.

У кваліфікаційній роботі на тему “Розробка адміністративної частини з використанням зовнішнього WebAPI для створення розкладу” було реалізовано інтерактивну адміністративну панель, завдяки якій є можливість редагувати контент на веб-сторінці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Документація Microsoft по Blazor [Електронний ресурс]: – Режим доступу: ASP.NET Core Blazor | Microsoft Learn.

2. Документація по ASP.NET Core [Електронний ресурс]: – Режим доступу: ASP.NET documentation | Microsoft Learn

3. Офіційний блог Microsoft:[Електронний ресурс]: – Режим доступу: ASP.NET - .NET Blog (microsoft.com)

4. Введення в WebAssembly: [Електронний ресурс]: WebAssembly | MDN (mozilla.org)

5. Microsoft Learn - Введення в Blazor:[Електронний ресурс]: – Режим доступу: Build your first web app with Blazor - Training | Microsoft Learn

6. Офіційний репозиторій GitHub для Blazor: [Електронний ресурс]: – Режим доступу: GitHub - dotnet/blazor: Blazor moved to https://github.com/dotnet/aspnetcore

7. Stack Overflow - обговорення Blazor: [Електронний ресурс]: – 2021. – Режим доступу: Newest 'blazor' Questions - Stack Overflow

8. Документація по JSON Web Token (JWT):[Електронний ресурс]: – Режим доступу: Latest JWT.io topics - Auth0 Community

9. Документація по Entity Framework Core:[Електронний ресурс]: – Режим доступу: Overview of Entity Framework Core - EF Core | Microsoft Learn

10. Microsoft Learn - Зовнішній WebAPI:[Електронний ресурс]: – Режим доступу: Create a web API with ASP.NET Core controllers - Training | Microsoft Learn

11. Blazor University - Online Tutorials: [Електронний ресурс]: – Режим доступу: Blazor University - Introduction (blazor-university.com)

12. Awesome Blazor - GitHub репозиторій з ресурсами та прикладами[Електронний ресурс]: – Режим доступу: GitHub - AdrienTorris/awesome-blazor: Resources for Blazor, a .NET web framework using C#/Razor and HTML that runs in the browser with WebAssembly.

13. Microsoft Build 2023 - Сесії та відео з Blazor – Режим доступу: Your home for Microsoft Build

14. JetBrains - Ресурси та статті про Blazor [Електронний ресурс]: – Режим доступу: blazor | The JetBrains Blog

15. Microsoft Learn - Advanced Blazor WebAssembly [Електронний ресурс]: – Режим доступу: Tooling for ASP.NET Core Blazor | Microsoft Learn

16. Бердник О.В., Горбатенко М.Ю. Онлайн розклад як складова цифрової трансформації університету. V міжнародна науково-практична конференція “Інформаційні моделюючі технології, системи та комплекси” (18 – 19 квітня 2024 року) Черкаси, Україна. С. 83-84. [Електронний ресурс]: - Режим доступу: https://fotius.cdu.edu.ua/wp-content/uploads/2024/05/Book_IMTCK_2024.pdf

ДОДАТОК

Auth.Provoder.cs:

```
using Microsoft.AspNetCore.Components.Authorization;
using System.Net.Http.Headers;
using System.Security.Claims;
using System.Text.Json;

namespace ScheduleAdmin
{
   public class AuthProvider : AuthenticationStateProvider
   {
      public override async Task<AuthenticationState>
GetAuthenticationStateAsync()
      {
         //no access
         //var identity = new ClaimsIdentity();

         //schedule admin
         string token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy5taWN
yb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvY2xhaW1zL3JvbGUiOi
JTY2hlZHVsZSBhZG1pbmlzdHJhdG9yIiwiaHR0cDovL3NjaGVtYXMueG1sc29
hcC5vcmcvd3MvMjAwNS8wNS9pZGVudGl0eS9jbGFpbXMvbmFtZSI6IkpvaG4
gRG9lIiwiZmFjdWx0eUlkIjoiMDdhMTYwNzItZDQ2MS00MWQxLThiNjgtY2
MwYjllZWJiNzg1IiwiaWF0IjoxNTE2MjM5MDIyfQ.5dcJ-
Jz3q63mrC1ppZPjYOXFGZhVk0SHiP4N0HytX8E";

         //falucty admin
         //string token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy5taWN
yb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvY2xhaW1zL3JvbGUiOi
JGYWN1bHR5IGFkbWluaXN0cmF0b3IiLCJodHRwOi8vc2NoZW1hcy54bWxzb
2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1lIjoiSm9ob
iBEb2UiLCJmYWN1bHR5SWQiOiIwN2ExNjA3Mi1kNDYxLTQxZDEtOGI2O
C1jYzBiOWVlYmI3ODUiLCJpYXQiOjE1MTYyMzkwMjJ9.YTzQ5CWVKPEIv
NjLfE5yRW43fTkNU0RUu9_69ISrM7Q";

         // admin
         //string token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy5taWN
yb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvY2xhaW1zL3JvbGUiOi
JBZG1pbmlzdHJhdG9yIiwiaHR0cDovL3NjaGVtYXMueG1sc29hcC5vcmcvd3M
vMjAwNS8wNS9pZGVudGl0eS9jbGFpbXMvbmFtZSI6IkpvaG4gRG9lIiwiZmFj
dWx0eUlkIjoiMDAwMDAwMDAtMDAwMC0wMDAwLTAwMDAtMDAwMD
```

```
AwMDAwMDAwIiwiaWF0IjoxNTE2MjM5MDIyfQ.9XZZqTowWA7ekU_AxOt
18ghHl0drLfE6SOP23osMJZE";

            var identity = new ClaimsIdentity(ParseClaimsFromJwt(token), "jwt");

            var user = new ClaimsPrincipal(identity);
            var state = new AuthenticationState(user);

            NotifyAuthenticationStateChanged(Task.FromResult(state));

            return state;
        }

        public static IEnumerable<Claim> ParseClaimsFromJwt(string jwt)
        {
            var payload = jwt.Split('.')[1];
            var jsonBytes = ParseBase64WithoutPadding(payload);
            var keyValuePairs = JsonSerializer.Deserialize<Dictionary<string,
object>>(jsonBytes);
            return keyValuePairs.Select(kvp => new Claim(kvp.Key,
kvp.Value.ToString()));
        }

        private static byte[] ParseBase64WithoutPadding(string base64)
        {
            switch (base64.Length % 4)
            {
                case 2: base64 += "=="; break;
                case 3: base64 += "="; break;
            }
            return Convert.FromBase64String(base64);
        }
    }
}
```

ScheduleService.cs:

```
using Newtonsoft.Json;
using ScheduleAdmin.Pages;
using ScheduleWebAPI.Domain.DTOs;
using ScheduleWebAPI.Domain.Entities;
using System;
using System.Net.Http.Json;
using System.Text;
using System.Xml.Linq;
```

```
namespace ScheduleAdmin.Schedule
{
   public class ScheduleService : IScheduleService
   {
      private readonly HttpClient _httpClient;
      private readonly string _baseUrl;

      private const string UniversitiesEndpoint = "/universities";
      private const string FacultiesEndpoint = "/universities/{0}/faculties";
      private const string AudiencesEndpoint =
"/universities/faculties/{0}/audiences";
      private const string DepartmentsEndpoint = "/departments";
      private const string EducationalProgramsEndpoint = "/educationalprograms";
      private const string SubjectsEndpoint = "/subjects/{0}";
      private const string SubjecNamesEndpoint = "/subjects/{0}/names";
      private const string GroupsEndpoint = "/groups?facultyId={0}";
      private const string AddGroupEndpoint = "/groups";
      private const string StudentsEndpoint = "/people/students";
      private const string TeachersEndpoint = "/people/teachers?facultyId={0}";
      private const string GroupLessonsEndpoint = "/Lessons?groupId={0}";
      private const string TeacherLessonsEndpoint = "/Lessons?teacherId={0}";
      private const string LessonsEndpoint = "/Lessons";

      public ScheduleService(HttpClient httpClient, string baseUrl)
      {
         _httpClient = httpClient;
         _baseUrl  = baseUrl;
      }

      public async Task<List<UniversityResponseDTO>?> GetUniversities()
      {
         return await
_httpClient.GetFromJsonAsync<List<UniversityResponseDTO>>(_baseUrl +
UniversitiesEndpoint);
      }

      public async Task AddUniversity(string name)
      {
         var json = JsonConvert.SerializeObject(name);
         var content = new StringContent(json, Encoding.UTF8, "application/json");

         var response = await _httpClient.PostAsync(_baseUrl + FacultiesEndpoint,
content);

         response.EnsureSuccessStatusCode();
```

```
    }

    public async Task<List<FacultyResponseDTO>?> GetFaculties(Guid
universityId)
    {
        string url = _baseUrl + string.Format(FacultiesEndpoint, universityId);
        return await
_httpClient.GetFromJsonAsync<List<FacultyResponseDTO>>(url);
    }

    public async Task AddFaculty(FacultyRequestDTO facultyDTO, Guid
universityId)
    {
        string url = _baseUrl + string.Format(FacultiesEndpoint, universityId);

        var json = JsonConvert.SerializeObject(facultyDTO);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        var response = await _httpClient.PostAsync(url, content);

        response.EnsureSuccessStatusCode();
    }

    public async Task<List<AudienceResponseDTO>?> GetAudiences(Guid
facultyId)
    {
        string url = _baseUrl + string.Format(AudiencesEndpoint, facultyId);
        return await
_httpClient.GetFromJsonAsync<List<AudienceResponseDTO>>(url);
    }

    public async Task AddAudience(AudienceRequestDTO
audienceRequestDTO, Guid facultyId)
    {
        string url = _baseUrl + string.Format(AudiencesEndpoint, facultyId);

        var json = JsonConvert.SerializeObject(audienceRequestDTO);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        var response = await _httpClient.PostAsync(url, content);

        response.EnsureSuccessStatusCode();
    }
```

```
        public async Task<List<DepartmentResponseDTO>?>
GetDepartmentsByFacultyId(Guid facultyId)
        {
            string url = _baseUrl + DepartmentsEndpoint + "?facultyId=" + facultyId;
            return await
_httpClient.GetFromJsonAsync<List<DepartmentResponseDTO>>(url);
        }

        public async Task<DepartmentResponseDTO?>
GetDepartmentByEducationalProgramId(Guid educationalProgramId)
        {
            string url = _baseUrl + DepartmentsEndpoint + "?educationalProgramId="
+ educationalProgramId;
            try
            {
                return await
_httpClient.GetFromJsonAsync<DepartmentResponseDTO>(url);
            }
            catch
            {
                return null;
            }
        }

        public async Task AddDepartment(DepartmentRequestDTO
departmentRequestDTO)
        {
            var json = JsonConvert.SerializeObject(departmentRequestDTO);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            var response = await _httpClient.PostAsync(_baseUrl +
DepartmentsEndpoint, content);

            response.EnsureSuccessStatusCode();
        }

        public async Task<List<EducationalProgramResponseDTO>?>
GetEducationalPrograms(Guid facultyId)
        {
            string url = _baseUrl + EducationalProgramsEndpoint;
            return await
_httpClient.GetFromJsonAsync<List<EducationalProgramResponseDTO>?>(url);
        }
```

```
    public async Task AddEducationalProgram(EducationalProgramRequestDTO
educationalProgramDTO)
    {
        var json = JsonConvert.SerializeObject(educationalProgramDTO);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        var response = await _httpClient.PostAsync(_baseUrl +
EducationalProgramsEndpoint, content);

        response.EnsureSuccessStatusCode();
    }

    public async Task<List<SubjectResponseDTO>?> GetSubjects(Guid
educationalProgramId)
    {
        string url = _baseUrl + string.Format(SubjectsEndpoint,
educationalProgramId);
        return await
_httpClient.GetFromJsonAsync<List<SubjectResponseDTO>>(url);
    }

    public async Task AddSubject(SubjectRequestDTO subject, Guid
educationalProgramId)
    {
        string url = _baseUrl + string.Format(SubjectsEndpoint,
educationalProgramId);

        var json = JsonConvert.SerializeObject(subject);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        var response = await _httpClient.PostAsync(url, content);

        response.EnsureSuccessStatusCode();
    }

    public async Task AddSubjectName(SubjectNameRequestDTO subject, Guid
subjectId)
    {
        string url = _baseUrl + string.Format(SubjecNamesEndpoint, subjectId);

        var json = JsonConvert.SerializeObject(subject);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        var response = await _httpClient.PostAsync(url, content);
```

```
            response.EnsureSuccessStatusCode();
        }

        public async Task<List<GroupResponseDTO>?> GetGroups(Guid facultyId)
        {
            string url = _baseUrl + string.Format(GroupsEndpoint, facultyId);
            return await
_httpClient.GetFromJsonAsync<List<GroupResponseDTO>?>(url);
        }

        public async Task AddGroup(GroupRequestDTO groupDTO)
        {
            var json = JsonConvert.SerializeObject(groupDTO);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            var response = await _httpClient.PostAsync(_baseUrl +
AddGroupEndpoint, content);

            response.EnsureSuccessStatusCode();
        }

        public async Task AddStudent(StudentRequestDTO studentDTO)
        {
            var json = JsonConvert.SerializeObject(studentDTO);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            var response = await _httpClient.PostAsync(_baseUrl + StudentsEndpoint,
content);

            response.EnsureSuccessStatusCode();
        }

        public async Task<List<TeacherResponseDTO>?> GetTeachers(Guid
facultyId)
        {
            string url = _baseUrl + string.Format(TeachersEndpoint, facultyId);
            return await
_httpClient.GetFromJsonAsync<List<TeacherResponseDTO>?>(url);
        }

        public async Task AddTeacher(TeacherRequestDTO teacherDTO)
        {
            var json = JsonConvert.SerializeObject(teacherDTO);
            var content = new StringContent(json, Encoding.UTF8, "application/json");
```

```
            var response = await _httpClient.PostAsync(_baseUrl + TeachersEndpoint,
content);

            response.EnsureSuccessStatusCode();
        }

        public async Task<List<LessonResponseDTO>?>
GetLessonsByGroupId(Guid groupId)
        {
            string url = _baseUrl + string.Format(GroupLessonsEndpoint, groupId);
            return await
_httpClient.GetFromJsonAsync<List<LessonResponseDTO>?>(url);
        }

        public async Task<List<LessonResponseDTO>?>
GetLessonsByTeacherId(Guid TeacherId)
        {
            string url = _baseUrl + string.Format(TeacherLessonsEndpoint, TeacherId);
            return await
_httpClient.GetFromJsonAsync<List<LessonResponseDTO>?>(url);
        }

        public async Task AddLesson(LessonRequestDTO lesson)
        {
            var json = JsonConvert.SerializeObject(lesson);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            var response = await _httpClient.PostAsync(_baseUrl + LessonsEndpoint,
content);

            response.EnsureSuccessStatusCode();
        }
    }
}
```

ISheduleService.cs:

```
using ScheduleWebAPI.Domain.DTOs;
using System.Threading.Tasks;

namespace ScheduleAdmin.Schedule
{
    public interface IScheduleService
    {
        public Task<List<UniversityResponseDTO>?> GetUniversities();
        public Task AddUniversity(string name);
```

```
        public Task<List<FacultyResponseDTO>?> GetFaculties(Guid universityId);
        public Task AddFaculty(FacultyRequestDTO facultyDTO, Guid
universityId);
        public Task<List<AudienceResponseDTO>?> GetAudiences(Guid facultyId);
        public Task AddAudience(AudienceRequestDTO audienceRequestDTO,
Guid facultyId);
        public Task<List<DepartmentResponseDTO>?>
GetDepartmentsByFacultyId(Guid facultyId);
        public Task<DepartmentResponseDTO?>
GetDepartmentByEducationalProgramId(Guid educationalProgramId);
        public Task AddDepartment(DepartmentRequestDTO
departmentRequestDTO);
        public Task<List<EducationalProgramResponseDTO>?>
GetEducationalPrograms(Guid facultyId);
        public Task AddEducationalProgram(EducationalProgramRequestDTO
educationalProgramDTO);
        public Task<List<SubjectResponseDTO>?> GetSubjects(Guid
educationalProgramId);
        public Task AddSubject(SubjectRequestDTO subject, Guid
educationalProgramId);
        public Task AddSubjectName(SubjectNameRequestDTO subject, Guid
subjectId);
        public Task<List<GroupResponseDTO>?> GetGroups(Guid facultyId);
        public Task AddGroup(GroupRequestDTO groupDTO);
        public Task AddStudent(StudentRequestDTO studentDTO);
        public Task<List<TeacherResponseDTO>?> GetTeachers(Guid facultyId);
        public Task AddTeacher(TeacherRequestDTO teacherDTO);
        public Task<List<LessonResponseDTO>?> GetLessonsByGroupId(Guid
groupId);
        public Task<List<LessonResponseDTO>?> GetLessonsByTeacherId(Guid
TeacherId);
        public Task AddLesson(LessonRequestDTO lesson);

    }
}
```

Header.razor:

```
<header>
    <nav>
        <section class="navbar-left">
            <img class="logo" src="/Logo-chnu.svg">
        </section>
        <section class="navbar-right">
            <div class="menu-items" id="menu-items">
```

```
            <div class="language-toggle" id="language-toggle"
onclick="toggleLanguage()">
                <img class="icon" id="language-icon" src="/UA.png">
            </div>
            <a class="button">Вийти</a>
        </div>
    </section>
  </nav>
</header>

@code {

}
```

MainLayout.razor:

```
@inherits LayoutComponentBase

<Header/>

<div class="schedule-section">
  <NavMenu/>
  <div class="schedule-div">
    @Body
  </div>
</div>
```

NavMenu.razor:

```
@inject UserInfo User
@inject NavigationManager Navigation

<div class="navigation">
  <AuthorizeView Roles="Faculty administrator, Schedule administrator">
    <Authorized>
      <NavLink class="navigation-link" Match="NavLinkMatch.All"
href="AddLessons" ActiveClass="navigation-active-link">
        Додати Розклад
      </NavLink>
      <NavLink class="navigation-link"
href="@($"/universities/faculties/{User.FacultyId}/groups")"
ActiveClass="navigation-active-link">
        Групи
      </NavLink>
      <NavLink class="navigation-link"
href="@($"/universities/faculties/{User.FacultyId}/departments")"
ActiveClass="navigation-active-link">
```

```
            Кафедри
        </NavLink>
        <NavLink class="navigation-link"
href="@($"/universities/faculties/{User.FacultyId}/audiences")"
ActiveClass="navigation-active-link">
            Аудиторії
        </NavLink>
        <NavLink class="navigation-link"
href="@($"/universities/faculties/{User.FacultyId}/educationalprograms")"
ActiveClass="navigation-active-link">
            Освітні програми
        </NavLink>
        <NavLink class="navigation-link"
href="@($"/universities/faculties/{User.FacultyId}/teachers")"
ActiveClass="navigation-active-link">
            Викладачі
        </NavLink>
    </Authorized>
  </AuthorizeView>
  <AuthorizeView Roles="Administrator">
    <Authorized>
        <NavLink class="navigation-link" Match="NavLinkMatch.All" href="/"
ActiveClass="navigation-active-link">
            Університети
        </NavLink>

    </Authorized>
  </AuthorizeView>

</div>

@code {
  protected override async Task OnInitializedAsync()
  {
  }
}
```

Program.cs:

```
using Microsoft.AspNetCore.Components.Authorization;
using Microsoft.AspNetCore.Components.Web;
using Microsoft.AspNetCore.Components.WebAssembly.Hosting;
using ScheduleAdmin;
using ScheduleAdmin.Schedule;
using System.Security.Claims;
```

```
var builder = WebAssemblyHostBuilder.CreateDefault(args);
builder.RootComponents.Add<App>("#app");
builder.RootComponents.Add<HeadOutlet>("head::after");
builder.Services.AddScoped<AuthenticationStateProvider, AuthProvider>();
builder.Services.AddAuthorizationCore();
builder.Services.AddScoped<UserInfo>(serviceProvider =>
{
    var authState =
serviceProvider.GetRequiredService<AuthenticationStateProvider>().GetAuthenti
cationStateAsync().Result;
    var user = authState.User;
    var facultyId = Guid.Parse(user.FindFirst("facultyId")!.Value);
    var role = user.FindFirst(ClaimTypes.Role)?.Value ?? "DefaultRole";
    return new UserInfo(user.Identity.Name, user.Identity.Name, role, facultyId);
});

builder.Services.AddScoped<IScheduleService, ScheduleService>(sp =>
{
    var httpClient = new HttpClient();
    var baseUrl = builder.Configuration.GetValue<string>("BaseUrl");

    if (!string.IsNullOrEmpty(baseUrl))
    {
        return new ScheduleService(httpClient, baseUrl);
    }
    else
    {
        throw new ArgumentNullException("BaseUrl", "Base URL is not specified in
the configuration.");
    }
});

builder.Services.AddScoped<AuthenticationStateProvider, AuthProvider>();
builder.Services.AddAuthorizationCore();

await builder.Build().RunAsync();
```

Userinfo.cs:

```
namespace ScheduleAdmin
{
    public class UserInfo
    {
        public string Name { get; private set; }
        public string Email { get; private set; }
        public string Role { get; private set; }
```

```
        public Guid FacultyId { get; private  set; }

        public UserInfo(string name, string email, string role, Guid facultyID)
        {
            Name = name;
            Email = email;
            Role = role;
            FacultyId = facultyID;
        }
    }
}
```

Pages:
AddAudience.razor

```
@page "/addaudience"
@inject UserInfo admin
@inject IScheduleService service
@attribute [Authorize(Roles = "Faculty administrator")]

<PageTitle>Add audience</PageTitle>

<div class="title">
    <h3>Додати аудиторію</h3>
</div>

<EditForm class="add-edit-form" Model="_audienceDTO"
OnValidSubmit="Submit">
    <div class="div-form">
        <label class="label-form">
            Номер:
        </label>
        <InputNumber class="input-form" @bind-Value="_audienceDTO.Number"
min="1" max="1000"></InputNumber>
    </div>
    <div class="div-form">
        <label class="label-form">
            Вмісткість:
        </label>
        <InputNumber class="input-form" @bind-Value="_audienceDTO.Capacity"
min="1" max="1000"></InputNumber>
    </div>
    <div class="div-form">
        <label class="label-form">
            Наявність проектора:
```

```
        </label>
        <InputCheckbox class="input-form" @bind-
Value="_audienceDTO.HasProjector"></InputCheckbox>
    </div>
    <div>
        <button type="submit" class="button submit-button">Submit</button>
    </div>
</EditForm>

@code {
    [SupplyParameterFromForm]
    private AudienceRequestDTO _audienceDTO { get; set; } = null!;

    protected override void OnInitialized()
    {
        _audienceDTO = new(0, 0, false);
    }

    private async Task Submit()
    {
        await service.AddAudience(_audienceDTO, admin.FacultyId);
    }
}
```

AddDepartment.razor

```
@page "/adddepartment"
@inject UserInfo admin
@inject IScheduleService service
@attribute [Authorize(Roles = "Faculty administrator")]

<PageTitle>Add department</PageTitle>

<div class="title">
    <h3>Додати кафедру</h3>
</div>

@if (_loading)
{
    <p>Loading...</p>
}
else
{
    <EditForm class="add-edit-form" Model="_departmentDTO"
OnValidSubmit="Submit">
        <div class="div-form">
            <label class="label-form">
```

```
            Навчальна програма:
        </label>
        <InputSelect class="input-form" @bind-
Value="_departmentDTO.EducationalProgramId" required>
            <option disabled hidden value="@Guid.Empty">-виберіть навчальну
програму-</option>
            @foreach (var educationalprogram in educationalprogramDTOs)
            {
                <option value="@educationalprogram.Id">
                    @educationalprogram.Name
                </option>
            }
        </InputSelect>
    </div>

    <div class="div-form">
        <label class="label-form">
            Назва:
        </label>
        <InputText class="input-form" @bind-Value="_departmentDTO.Name"
minlength="3" maxlength="100" required></InputText>
    </div>

    <div>
        <button type="submit" class="button submit-button">Submit</button>
    </div>
</EditForm>
}

@code {
    private bool _loading { get; set; } = true;

    [SupplyParameterFromForm]
    private DepartmentRequestDTO _departmentDTO { get; set; } = null!;

    private List<EducationalProgramResponseDTO> educationalprogramDTOs {
get; set; } = null!;

    protected override async Task OnInitializedAsync()
    {
        educationalprogramDTOs = await
service.GetEducationalPrograms(admin.FacultyId);
        _departmentDTO = new(Guid.Empty, "");

        _loading = false;
```

```
    }

    private async Task Submit()
    {
        await service.AddDepartment(_departmentDTO);
    }
}
```

AddEducationalProgram.razor

```
@page "/addeducationalprogram"
@using ScheduleAdmin.Schedule
@inject UserInfo admin
@inject IScheduleService service
@attribute [Authorize(Roles = "Faculty administrator")]

<PageTitle>Add educational program</PageTitle>

<div class="title">
    <h3>Додати освітню програму</h3>
</div>

<EditForm class="add-edit-form" Model="_educationalProgramDTO"
OnValidSubmit="Submit">
    <div class="div-form">
        <label class="label-form">
            Назва програми:
        </label>
        <InputText class="input-form" @bind-
Value="_educationalProgramDTO.Name" minlength="10" maxlength="100"
required></InputText>
    </div>
    <div class="div-form">
        <label class="label-form">
            Назва спеціальності:
        </label>
        <InputText class="input-form" @bind-
Value="_educationalProgramDTO.SpecialtyName" minlength="10"
maxlength="100" required></InputText>
    </div>
    <div class="div-form">
        <label class="label-form">
            Код спеціальності:
        </label>
        <InputNumber class="input-form" @bind-
Value="_educationalProgramDTO.SpecialtyCode" min="100"
max="999"></InputNumber>
```

```
    </div>
    <div class="div-form">
        <label class="label-form">
            Спеціалізація:
        </label>
        <InputText class="input-form" @bind-
Value="_educationalProgramDTO.Specialization"></InputText>
    </div>
    <div class="div-form">
        <label class="label-form">
            Кваліфікація:
        </label>
        <InputText class="input-form" @bind-
Value="_educationalProgramDTO.Qualification" required></InputText>
    </div>
    <div class="div-form">
        <label class="label-form">
            Професійна кваліфікація:
        </label>
        <InputText class="input-form" @bind-
Value="_educationalProgramDTO.ProfessionalQualification"></InputText>
    </div>
    <div class="div-form">
        <label class="label-form">
            Компетенції:
        </label>
        <InputText class="input-form" @bind-
Value="_educationalProgramDTO.Competencies" required></InputText>
    </div>
    <div class="div-form">
        <label class="label-form">
            Номер сертифікату:
        </label>
        <InputNumber class="input-form" @bind-
Value="_educationalProgramDTO.CertificateNumber" min="1000"
max="9999"></InputNumber>
    </div>
    <div class="div-form">
        <label class="label-form">
          Фото:
        </label>
        <InputFile class="input-form" OnChange="LoadPhoto" multiple required />
    </div>
    <div>
        <button type="submit" class="button submit-button">Submit</button>
```

```
    </div>
</EditForm>

@code {
    [SupplyParameterFromForm]
    private EducationalProgramRequestDTO _educationalProgramDTO { get; set; }
= null!;

    protected override void OnInitialized()
    {
        _educationalProgramDTO = new("",
            "",
            0,
            null,
            " ",
            null,
            "",
            0,
            "",
            admin.FacultyId);
    }

    private async void LoadPhoto(InputFileChangeEventArgs e)
    {
        var fileFormat = "image/png";

        var imageFiles = e.GetMultipleFiles();

        var buffer = new byte[imageFiles[0].Size];
        await imageFiles[0].OpenReadStream().ReadAsync(buffer);

        _educationalProgramDTO.CertificatePhoto = $"data:{fileFormat}; base64,
{Convert.ToBase64String(buffer)}";
    }

    private async Task Submit()
    {
        await service.AddEducationalProgram(_educationalProgramDTO);
    }
}
```

AddFaculty.razor

```
@page "/addfaculty"
@inject UserInfo admin
@inject IScheduleService service
```

```
@attribute [Authorize(Roles = "Administrator")]

<PageTitle>Add faculty</PageTitle>

<div class="title">
    <h3>Додати факультет</h3>
</div>

@if (_loading)
{
    <p>Loading...</p>
}
else
{
    <EditForm class="add-edit-form" Model="_facultyDTO"
OnValidSubmit="Submit">
        <div class="div-form">
            <label class="label-form">
                Університет:
            </label>
            <select class="input-form" @oninput="(e) => SelectedUniversity(e)"
required>
                <option disabled hidden value="@Guid.Empty">-виберіть навчальну
програму-</option>
                @foreach (var university in _universityDTOs)
                {
                    <option value="@university.Id">
                        @university.Name
                    </option>
                }
            </select>
        </div>

        <div class="div-form">
            <label class="label-form">
                Назва:
            </label>
            <InputText class="input-form" @bind-Value="_facultyDTO.Name"
minlength="10" maxlength="80" required></InputText>
        </div>

        <div class="div-form">
            <label class="label-form">
                Номер:
            </label>
```

```
            <InputNumber class="input-form" @bind-Value="_facultyDTO.Number"
min="0" max="999"></InputNumber>
        </div>

        <div>
            <button type="submit" class="button submit-button">Submit</button>
        </div>
    </EditForm>
}

@code {
    private bool _loading { get; set; } = true;
    private Guid _univerityId { get; set; }

    [SupplyParameterFromForm]
    private FacultyRequestDTO _facultyDTO { get; set; } = null!;

    private List<UniversityResponseDTO> _universityDTOs { get; set; } = null!;

    protected override async Task OnInitializedAsync()
    {
        _universityDTOs = await service.GetUniversities();
        _facultyDTO = new("",0);

        _loading = false;
    }

    private async Task SelectedUniversity(ChangeEventArgs e)
    {
        if (Guid.TryParse(e.Value!.ToString(), out Guid selectedUniversity))
        {
            _univerityId = selectedUniversity;
        }
    }

    private async Task Submit()
    {
        await service.AddFaculty(_facultyDTO, _univerityId);
    }
}
```

AddGroup.razor

```
@page "/addgroup"
@using ScheduleAdmin.Schedule
@using ScheduleWebAPI.Domain.Enums
```

```
@inject UserInfo admin
@inject IScheduleService service

<PageTitle>Add group</PageTitle>

<div class="title">
    <h3>Додати групу</h3>
</div>

@if (_loading)
{
    <p>Loading...</p>
}
else
{
    <EditForm class="add-edit-form" Model="_group"
OnValidSubmit="Submit">
        <div class="div-form">
            <label class="label-form">
                Номер:
            </label>
            <InputNumber class="input-form" @bind-Value="_group.Number"
min="1" max="1000"></InputNumber>
        </div>

        <div class="div-form">
            <label class="label-form">
                Навчальна програма:
            </label>
            <InputSelect class="input-form" @bind-
Value="_group.EducationalProgramId" required>
                <option disabled hidden value="@Guid.Empty">-виберіть навчальну
програму-</option>
                @foreach (var educationalprogram in _educationalProgramDTOs)
                {
                    <option value="@educationalprogram.Id">
                        @educationalprogram.Name
                    </option>
                }
            </InputSelect>
        </div>

        <div>
            <button type="submit" class="button submit-button">Submit</button>
        </div>
```

```
    </EditForm>
}

@code {
    private bool _loading { get; set; } = true;

    [SupplyParameterFromForm]
    private GroupRequestDTO _group { get; set; } = null!;

    private List<EducationalProgramResponseDTO> _educationalProgramDTOs {
get; set; } = null!;

    protected override async Task OnInitializedAsync()
    {
        _educationalProgramDTOs = await
service.GetEducationalPrograms(admin.FacultyId);
        _group = new(0, Guid.Empty);

        _loading = false;
    }

    private async Task Submit()
    {
        await service.AddGroup(_group);
    }
}
```

AddLesson.razor

```
@page "/addlessons"
@using ScheduleAdmin.Schedule
@using ScheduleWebAPI.Domain.Enums
@inject UserInfo admin
@inject IScheduleService service

<PageTitle>Add lessons</PageTitle>

<div class="title">
    <h3>Додати розклад</h3>
</div>

@if (_loading)
{
    <p>Loading...</p>
}
else
```

```
{
   <EditForm class="add-edit-form" Model="_lesson"
OnValidSubmit="Submit">
      <div class="div-form">
         <label class="label-form">
            Викладач:
         </label>
         <InputSelect class="input-form" @bind-Value="_lesson.TeacherId">
               <option disabled hidden value="@Guid.Empty">Select a
Teacher</option>
               @foreach (var teacher in _teacherDTOs)
               {
                  <option value="@teacher.Id">
                     @($"{teacher.Firstname} {teacher.Surname}
{teacher.Patronymic}")
                  </option>
               }
         </InputSelect>
      </div>

      <div class="div-form" >
         <label class="label-form">
            Аудиторія:
         </label>
         <InputSelect class="input-form" @bind-Value="_lesson.AudienceId">
            <option disabled hidden value="@Guid.Empty">--виберіть аудиторію-
-</option>
            @foreach (var audience in _audienceDTOs)
            {
               <option value="@audience.Id">
                  @audience.Number
               </option>
            }
         </InputSelect>
      </div>

      <div class="div-form">
         <label class="label-form">
            Групи:
         </label>
         <select multiple class="input-form" @onchange="SelectedGroup">
            @foreach (var group in _groupDTOs)
            {
               <option value="@group.Id">
                     @group.Number
```

```
                </option>
            }
        </select>
    </div>

    <div class="div-form">
        <label class="label-form">
            Предмет:
        </label>
        <InputSelect class="input-form" @bind-Value="_lesson.SubjectId">
            <option disabled hidden value="@Guid.Empty">--виберіть предмет--
</option>
            @if (_subjectGroups != null && _subjectGroups.Count()>0)
            {
                @foreach (var group in _subjectGroups)
                {
                    <option value="@group.Key">
                        @foreach (var item in group)
                        {
                            <text>@item.Name; </text>
                        }
                    </option>
                }
            }
        </InputSelect>
    </div>

    <div class="div-form">
        <label class="label-form">
            тип предмету:
        </label>
        <InputSelect class="input-form" @bind-Value="_lesson.Type" required>
            <option value="@LessonType.Lecture">
                @LessonType.Lecture.ToString()
            </option>
            <option value="@LessonType.Practical">
                @LessonType.Practical.ToString()
            </option>
            <option value="@LessonType.Laboratory">
                @LessonType.Laboratory.ToString()
            </option>
            <option value="@LessonType.Seminar">
                @LessonType.Seminar.ToString()
            </option>
        </InputSelect>
```

```
        </div>
        @foreach (var block in blocks)
        {
            <div class="div-form">
                <label class="label-form">
                    День:
                </label>
                <select class="input-form" @oninput="(e) => SelectedDay(e, block-1)"
required>
                    <option disabled selected value>--виберіть день--</option>
                    <option value="@WeekDay.Monday">
                        @WeekDay.Monday.ToString()
                    </option>
                    <option value="@WeekDay.Tuesday">
                        @WeekDay.Tuesday.ToString()
                    </option>
                    <option value="@WeekDay.Wednesday">
                        @WeekDay.Wednesday.ToString()
                    </option>
                    <option value="@WeekDay.Thursday">
                        @WeekDay.Thursday.ToString()
                    </option>
                    <option value="@WeekDay.Friday">
                        @WeekDay.Friday.ToString()
                    </option>
                    <option value="@WeekDay.Saturday">
                        @WeekDay.Saturday.ToString()
                    </option>
                </select>
            </div>

            <div class="div-form">
                <label class="label-form">
                    Година початку:
                </label>
                <select class="input-form" @oninput="(e) => SelectedTime(e, block-1)"
required>
                    <option disabled selected value>--виберіть час--</option>
                    <option value="08:20:00">
                        8:20
                    </option>
                    <option value="09:50:00">
                        9:50
                    </option>
                    <option value="11:30:00">
```

```
                    11:30
                </option>
                <option value="13:00:00">
                    13:00
                </option>
                <option value="14:40:00">
                    14:40
                </option>
                <option value="16:10:00">
                    16:10
                </option>
            </select>
        </div>
    }

    <div class="div-form">
        <button class="button add-days-button" @onclick="AddBlock">
            <img src="/Add.png" class="add-img" />
        </button>
    </div>

    <div class="div-form">
        <label class="label-form">
            Кожного тижня:
        </label>
        <InputCheckbox class="input-form" @bind-
Value="_lesson.EveryWeek"></InputCheckbox>
    </div>

    <div class="div-form">
        <label class="label-form">
            Перший тиждень:
        </label>
        <InputCheckbox class="input-form" @bind-
Value="_lesson.FirstWeek"></InputCheckbox>
    </div>

    <div>
        <button type="submit" class="button submit-button">Submit</button>
    </div>
</EditForm>
}


@code {
```

```
    private bool _loading { get; set; } = true;
    private List<int> blocks = new List<int> { 1 };

    [SupplyParameterFromForm]
    private LessonRequestDTO _lesson { get; set; } = null!;

    private List<TeacherResponseDTO>? _teacherDTOs;
    private List<AudienceResponseDTO>? _audienceDTOs;
    private List<GroupResponseDTO>? _groupDTOs;

    private List<SubjectResponseDTO>? _subjectsDTOs;
    private IEnumerable<IGrouping<Guid, SubjectResponseDTO>>
_subjectGroups = null!;

    protected override async Task OnInitializedAsync()
    {
        _teacherDTOs = await service.GetTeachers(admin.FacultyId);
        _audienceDTOs = await service.GetAudiences(admin.FacultyId);
        _groupDTOs = await service.GetGroups(admin.FacultyId);

        _lesson = new(
            Guid.Empty,
            Guid.Empty,
            Guid.Empty,
            new WeekDay[1],
            new TimeOnly[1],
            0,
            false,
            false,
            new List<Guid>());

        _loading = false;
        await base.OnInitializedAsync();
    }

    private async Task SelectedGroup(ChangeEventArgs e)
    {
        var selectedGroups = ((string[])e.Value!).Select(Guid.Parse).ToArray();
        _lesson.GroupsId = selectedGroups;
        _subjectsDTOs = new List<SubjectResponseDTO>();

        foreach (var group in selectedGroups)
        {
            var res = await service.GetSubjects(_groupDTOs.Where(x => x.Id ==
group).First().EducationalProgramId);
```

```
            _subjectsDTOs.AddRange(res);
        }

        if (selectedGroups.Length>1)
        {
            _subjectGroups = _subjectsDTOs
                        .GroupBy(x => x.SubjectId)
                        .Where(g => g.Count() > 1);
        }
        else
        {
            _subjectGroups = _subjectsDTOs
                        .GroupBy(x => x.SubjectId);
        }

    }

    private void SelectedDay(ChangeEventArgs e, int index)
    {
        if (Enum.TryParse<WeekDay>(e.Value.ToString(), out var selectedDay))
        {
            if (_lesson.Days.Length>index)
            {
                _lesson.Days[index] = selectedDay;
                return;
            }
            int currentLength = _lesson.Days.Length;
            WeekDay[] newDays = new WeekDay[currentLength + 1];

            Array.Copy(_lesson.Days, newDays, currentLength);

            newDays[currentLength] = selectedDay;
            _lesson.Days = newDays;
        }
    }

    private void SelectedTime(ChangeEventArgs e, int index)
    {
        Console.WriteLine(index);
        if (TimeOnly.TryParse(e.Value.ToString(), out var selectedTime))
        {
            if (_lesson.Times.Length > index)
            {
                _lesson.Times[index] = selectedTime;
                return;
```

```
            }
            int currentLength = _lesson.Times.Length;
            TimeOnly[] newTimes = new TimeOnly[currentLength + 1];

            Array.Copy(_lesson.Times, newTimes, currentLength);

            newTimes[currentLength] = selectedTime;
            _lesson.Times = newTimes;
        }
    }

    private void AddBlock()
    {
        blocks.Add(blocks.Count + 1);
    }

    private async Task Submit()
    {
        await service.AddLesson(_lesson);
    }
}
```

AddStudent.razor

```
@page "/addstudent"
@using ScheduleAdmin.Schedule
@using ScheduleWebAPI.Domain.Enums
@inject UserInfo admin
@inject IScheduleService service

<PageTitle>Add student</PageTitle>

<div class="title">
    <h3>Додати студента</h3>
</div>

@if (_loading)
{
    <p>Loading...</p>
}
else
{
    <EditForm class="add-edit-form" Model="_student"
OnValidSubmit="Submit">
        <div class="div-form">
            <label class="label-form">
```

```
            Ім'я:
          </label>
          <InputText class="input-form" @bind-Value="_student.Firstname"
minlength="3" maxlength="100" required></InputText>
        </div>
        <div class="div-form">
          <label class="label-form">
            По батькові:
          </label>
          <InputText class="input-form" @bind-Value="_student.Patronymic"
minlength="3" maxlength="100" required></InputText>
        </div>
        <div class="div-form">
          <label class="label-form">
            Прізвище:
          </label>
          <InputText class="input-form" @bind-Value="_student.Surname"
minlength="3" maxlength="100" required></InputText>
        </div>
        <div class="div-form">
          <label class="label-form">
            Email:
          </label>
          <InputText class="input-form" type="email" @bind-
Value="_student.Email" required></InputText>
        </div>
        <div class="div-form">
          <label class="label-form">
            Група:
          </label>
          <InputSelect class="input-form" @bind-Value="_student.GroupId">
            <option disabled hidden value="@Guid.Empty">--виберіть групу--
</option>
            @foreach (var group in _groupDTOs)
            {
              <option value="@group.Id">
                @group.Number|@group.EducationalProgramName
              </option>
            }
          </InputSelect>
        </div>
        <div class="div-form">
          <label class="label-form">
            День народження:
          </label>
```

```
            <InputDate class="input-form" @bind-Value="_student.Birthday"
minlength="10" maxlength="100" required></InputDate>
        </div>
        <div>
            <button type="submit" class="button submit-button">Submit</button>
        </div>
    </EditForm>
}

@code {
    private bool _loading { get; set; } = true;

    [SupplyParameterFromForm]
    private StudentRequestDTO _student { get; set; } = null!;

    private List<GroupResponseDTO>? _groupDTOs;

    protected override async Task OnInitializedAsync()
    {
        _groupDTOs = await service.GetGroups(admin.FacultyId);
        _student = new StudentRequestDTO("", "", "", "", Guid.Empty,
DateOnly.FromDateTime(DateTime.Now));

        _loading = false;
    }

    private async Task Submit()
    {
        await service.AddStudent(_student);
    }
}
```

AddSubject.razor

```
@page "/addsubject"
@inject UserInfo admin
@inject IScheduleService service
@attribute [Authorize(Roles = "Faculty administrator")]

<PageTitle>Add subject</PageTitle>

<div class="title">
    <h3>Додати предмет</h3>
</div>

@if (_loading)
```

```
{
    <p>Loading...</p>
}
else
{
    <EditForm class="add-edit-form" Model="_subjestDTO"
OnValidSubmit="Submit">
        <div class="div-form">
            <label class="label-form">
                Навчальна програма:
            </label>
                <select class="input-form" @oninput="(e) =>
SelectedEducationalProgramId(e)" required>
                <option disabled hidden value="@Guid.Empty">-виберіть навчальну
програму-</option>
                @foreach (var educationalprogram in educationalprogramDTOs)
                {
                    <option value="@educationalprogram.Id">
                        @educationalprogram.Name
                    </option>
                }
            </select>
        </div>

        <div class="div-form">
            <label class="label-form">
                Назва:
            </label>
            <InputText class="input-form" @bind-Value="_subjestDTO.Name"
minlength="3" maxlength="100" required></InputText>
        </div>
        <div class="div-form">
            <label class="label-form">
                Обов'язковий:
            </label>
            <InputCheckbox @bind-
Value="_subjestDTO.IsCompulsory"></InputCheckbox>
        </div>

        <div>
            <button type="submit" class="button submit-button">Submit</button>
        </div>
    </EditForm>
}
```

```
@code {
    private bool _loading { get; set; } = true;
    private List<int> blocks = new List<int>();
    private Guid _educationalProgramId { get; set; }

    [SupplyParameterFromForm]
    private SubjectRequestDTO _subjestDTO { get; set; } = null!;
    private SubjectNameRequestDTO _subjestNameDTO { get; set; } = null!;

    private List<EducationalProgramResponseDTO> educationalprogramDTOs {
get; set; } = null!;

    protected override async Task OnInitializedAsync()
    {
        educationalprogramDTOs = await
service.GetEducationalPrograms(admin.FacultyId);
        _subjestDTO = new("",false);

        _loading = false;
    }

    private async Task SelectedEducationalProgramId(ChangeEventArgs e)
    {
        if (Guid.TryParse(e.Value.ToString(), out Guid selectedProgram))
        {
            _educationalProgramId = selectedProgram;
        }
    }

    private void AddBlock()
    {
        blocks.Add(blocks.Count + 1);
    }

    private async Task Submit()
    {
        await service.AddSubject(_subjestDTO, _educationalProgramId);
    }
}
```

AddSubjectName.razor

```
@page "/addsubjectname"
@inject UserInfo admin
@inject IScheduleService service
@attribute [Authorize(Roles = "Faculty administrator")]
```

```
<PageTitle>Add subject name</PageTitle>

<div class="title">
   <h3>Додати назву предмету</h3>
</div>

@if (_loading)
{
   <p>Loading...</p>
}
else
{
   <EditForm class="add-edit-form" Model="_subjestNameDTO"
OnValidSubmit="Submit">
      <div class="div-form">
         <label class="label-form">
           Предмет:
         </label>
         <select class="input-form" @oninput="(e) =>
SelectedEducationalProgramSubject(e)" required>
            <option selected disabled hidden value="@Guid.Empty">-виберіть
навчальну програму-</option>
            @foreach (var subject in subjectDTOs)
            {
               <option value="@subject.SubjectId">
                  @subject.Name; @subject.EducationalProgramName
               </option>
            }
         </select>
      </div>
      <div class="div-form">
         <label class="label-form">
            Навчальна програма:
         </label>
         <InputSelect class="input-form"  @bind-
Value="_subjestNameDTO.EducationalProgramId" required>
            <option disabled hidden value="@Guid.Empty">-виберіть навчальну
програму-</option>
            @foreach (var educationalprogram in educationalprogramDTOs)
            {
               <option value="@educationalprogram.Id">
                  @educationalprogram.Name
               </option>
            }
```

```
            </InputSelect>
        </div>

        <div class="div-form">
            <label class="label-form">
                Назва:
            </label>
            <InputText class="input-form" @bind-Value="_subjestNameDTO.Name"
minlength="3" maxlength="100" required></InputText>
        </div>

        <div class="div-form">
            <label class="label-form">
                Обов'язковий:
            </label>
            <InputCheckbox @bind-
Value="_subjestNameDTO.IsCompulsory"></InputCheckbox>
        </div>

        <div>
            <button type="submit" class="button submit-button">Submit</button>
        </div>
    </EditForm>
}

@code {
    private bool _loading { get; set; } = true;
    private List<int> blocks = new List<int>();

    private Guid _subjectId { get; set; }

    [SupplyParameterFromForm]
    private SubjectNameRequestDTO _subjestNameDTO { get; set; } = null!;

    private List<EducationalProgramResponseDTO> educationalprogramDTOs {
get; set; } = null!;
    private List<SubjectResponseDTO> subjectDTOs { get; set; } = null!;

    protected override async Task OnInitializedAsync()
    {
        educationalprogramDTOs = await
service.GetEducationalPrograms(admin.FacultyId);
        subjectDTOs = new();
        _subjectId = Guid.Empty;
```

```
        foreach (var program in educationalprogramDTOs)
        {
            subjectDTOs.AddRange(await service.GetSubjects(program.Id));
        }

        _subjestNameDTO = new("",Guid.Empty, false);

        _loading = false;
    }

    private async Task SelectedEducationalProgramSubject(ChangeEventArgs e)
    {
        if (Guid.TryParse(e.Value.ToString(), out Guid selectedSubject))
        {
            _subjectId = selectedSubject;
        }
    }

    private void AddBlock()
    {
        blocks.Add(blocks.Count + 1);
    }

    private async Task Submit()
    {
        await service.AddSubjectName(_subjestNameDTO, _subjectId);
    }
}
```

AddTeacher.razor

```
@page "/addteacher"
@using ScheduleWebAPI.Domain.Enums
@inject UserInfo admin
@inject IScheduleService service
@attribute [Authorize(Roles = "Faculty administrator")]

<PageTitle>Add teacher</PageTitle>

<div class="title">
    <h3>Додати Викладача</h3>
</div>

@if (_loading)
{
    <p>Loading...</p>
```

```
}
else
{
   <EditForm class="add-edit-form" Model="_teacherDTO"
OnValidSubmit="Submit">
      <div class="div-form">
         <label class="label-form">
            Ім'я:
         </label>
         <InputText class="input-form" @bind-Value="_teacherDTO.Firstname"
minlength="3" maxlength="100" required></InputText>
      </div>

      <div class="div-form">
         <label class="label-form">
            Прізвище:
         </label>
         <InputText class="input-form" @bind-Value="_teacherDTO.Surname"
minlength="3" maxlength="100" required></InputText>
      </div>
      <div class="div-form">
         <label class="label-form">
            По батькові:
         </label>
         <InputText class="input-form" @bind-Value="_teacherDTO.Patronymic"
minlength="3" maxlength="100" required></InputText>
      </div>

      <div class="div-form">
         <label class="label-form">
            Електронна пошта:
         </label>
         <InputText type="email" class="input-form" @bind-
Value="_teacherDTO.Email" t minlength="3" maxlength="100"
required></InputText>
      </div>

      <div class="div-form">
         <label class="label-form">
           Кафедра:
         </label>
         <InputSelect class="input-form" @bind-
Value="_teacherDTO.DepartmentId">
            <option disabled hidden value="@Guid.Empty">-виберіть кафедру-
</option>
```

```
            @foreach (var department in _departmentDTOs)
            {
                <option value="@department.Id">
                    @department.Name
                </option>
            }
        </InputSelect>
    </div>

    <div class="div-form">
        <label class="label-form">
            Звання:
        </label>
        <InputSelect class="input-form" @bind-Value="_teacherDTO.Rank"
required>
            <option
value="@TeacherRank.Docent">@TeacherRank.Docent.ToString()</option>
            <option
value="@TeacherRank.Professor">@TeacherRank.Professor.ToString()</option>
            <option
value="@TeacherRank.Assistant">@TeacherRank.Assistant.ToString()</option>
        </InputSelect>
    </div>

    <div>
        <button type="submit" class="button submit-button">Submit</button>
    </div>
</EditForm>
}

@code {
    private bool _loading { get; set; } = true;

    private Guid _educationalProgramId { get; set; }

    [SupplyParameterFromForm]
    private TeacherRequestDTO _teacherDTO { get; set; } = null!;

    private List<DepartmentResponseDTO> _departmentDTOs { get; set; } = null!;

    protected override async Task OnInitializedAsync()
    {
        _departmentDTOs = await
service.GetDepartmentsByFacultyId(admin.FacultyId);
        _teacherDTO = new("", "", "", "", Guid.Empty, 0);
```

```
        _loading = false;
    }

    private async Task Submit()
    {
        await service.AddTeacher(_teacherDTO);
    }
}
```

AddUniversity.razor

```
@page "/adduniversity"
@using ScheduleAdmin.Schedule
@inject IScheduleService service
@attribute [Authorize(Roles = "Administrator")]

<PageTitle>Create university</PageTitle>

<div class="title">
    <h3>Додати університет</h3>
</div>

<EditForm class="add-edit-form" Model="UniversityName"
OnValidSubmit="Submit">
    <div class="div-form">
        <label class="label-form">
            Назва:
        </label>
        <InputText class="input-form" @bind-Value="UniversityName"
minlength="3" maxlength="100" required />
    </div>
    <div>
        <button type="submit" class="button submit-button">Submit</button>
    </div>
</EditForm>

@code {
    [SupplyParameterFromForm]
    public string UniversityName { get; set; } = String.Empty;

    private async Task Submit()
    {
        await service.AddUniversity(UniversityName);
    }
}
```

Audiences.razor

```
@page "/universities/faculties/{facultyId:guid}/audiences"
@inject IScheduleService service

<PageTitle>Audiences</PageTitle>


<AuthorizeView Roles="Faculty administrator">
    <Authorized>
        <NavLink class="button add-button" href="/addaudience">
            Додати аудиторію
        </NavLink>
    </Authorized>
</AuthorizeView>

@if (_audienceDTOs == null)
{
    <p>No data</p>
}
else
{
    <table class="schedule-table">
        <thead>
            <tr class="schedule-days">
                <th class="border-right-bottom"> </th>
                <th class="border-right-bottom">Номер</th>
                <th class="border-right-bottom">Вмісткість</th>
                <th class="border-right-bottom">Проектом</th>
                <th class="border-right-bottom">Факультет</th>
                <th class="border-bottom"> </th>
            </tr>
        </thead>
        <tbody>
            @foreach (var audience in _audienceDTOs)
            {
                <tr>
                    <th class="border-right-bottom">@_counter</th>
                    <td class="border-right-bottom">@audience.Number</td>
                    <td class="border-right-bottom">@audience.Capacity</td>
                    <td class="border-right-bottom">@audience.HasProjector</td>
                    <td class="border-right-bottom">@audience.FacultyName</td>
                    <td class="border-bottom">
                        <button class="delete-button">
```

```
                    <img src="/delete.png" class="icon " />
                </button>
                <button class="edit-button">
                    <img src="/edit.png" class="icon " />
                </button>
            </td>
        </tr>
        _counter++;
    }
</tbody>
</table>


}

@code {
    [Parameter]
    public Guid FacultyId { get; set; }

    private int _counter;
    private List<AudienceResponseDTO>? _audienceDTOs;

    protected override async Task OnInitializedAsync()
    {
        _counter = 1;
        _audienceDTOs = await service.GetAudiences(FacultyId);
    }
}
```

Counter.razor

```
@page "/counter"
@attribute [Authorize(Roles = "Schedule administrator")]

<PageTitle>Counter</PageTitle>

<h1>Counter</h1>

<p role="status">Current count: @currentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

<AuthorizeView Roles="Admin">
    <Authorized>
        <span>you are @context.User.Identity.Name</span>
    </Authorized>
```

```
</AuthorizeView>


@code {
    private int currentCount = 0;

    private void IncrementCount()
    {
        currentCount++;
    }
}
```

Departments.razor

```
@page "/universities/faculties/{facultyId:guid}/departments"
@page "/educationalprogram/{educationalProgramId:guid}/departments"
@inject IScheduleService service

<PageTitle>Departments</PageTitle>

<AuthorizeView Roles="Faculty administrator">
    <Authorized>
        <NavLink class="button add-button" href="/adddepartment">
            Додати кафедру
        </NavLink>
    </Authorized>
</AuthorizeView>

@if (_departmentDTOs == null)
{
    <p>No data</p>
}
else
{
    <table class="schedule-table">
        <thead>
            <tr class="schedule-days">
                <th class="border-right-bottom"> </th>
                <th class="border-right-bottom">Назва</th>
                <th class="border-right-bottom">Освітня програма</th>
                <th class="border-bottom"> </th>
            </tr>
        </thead>
        <tbody>
            @foreach (var department in _departmentDTOs)
```

```
            {
                <tr>
                    <th class="border-right-bottom">@_counter</th>
                    <td class="border-right-bottom">@department.Name</td>
                    <td class="border-right-
bottom">@department.EducationalProgramName</td>
                    <td class="border-bottom">
                        <button class="delete-button">
                            <img src="/delete.png" class="icon " />
                        </button>
                        <button class="edit-button">
                            <img src="/edit.png" class="icon " />
                        </button>
                    </td>
                </tr>
                _counter++;
            }
        </tbody>
    </table>


}

@code {
    [Parameter]
    public Guid FacultyId { get; set; }
    [Parameter]
    public Guid EducationalProgramId { get; set; }

    private int _counter;
    private List<DepartmentResponseDTO>? _departmentDTOs;

    protected override async Task OnInitializedAsync()
    {
        _counter = 1;
        if (FacultyId != Guid.Empty)
        {
            _departmentDTOs = await service.GetDepartmentsByFacultyId(FacultyId);
        }
        else
        {
            DepartmentResponseDTO? response = await
service.GetDepartmentByEducationalProgramId(EducationalProgramId);
            if (response != null)
            {
```

```
            _departmentDTOs = new List<DepartmentResponseDTO>() { response
};
        }
    }
  }
}
```

EducationalPrograms.razor

```
@page "/universities/faculties/{facultyId:guid}/educationalprograms"
@inject IScheduleService service
@inject NavigationManager Navigation

<PageTitle>EducationalPrograms</PageTitle>


<AuthorizeView Roles="Faculty administrator">
  <Authorized>
    <NavLink class="button add-button" href="/addeducationalprogram">
      Додати освітню програму
    </NavLink>
  </Authorized>
</AuthorizeView>

@if (_educationalProgramDTOs == null)
{
  <p>No data</p>
}
else
{
  <table class="schedule-table">
    <thead>
      <tr class="schedule-days">
        <th class="border-right-bottom"> </th>
        <th class="border-right-bottom">Назва</th>
        <th class="border-right-bottom">Назва спеціальності</th>
        <th class="border-right-bottom">Код спеціальності</th>
        <th class="border-right-bottom">Спеціалізація</th>
        <th class="border-right-bottom">Кваліфікація</th>
        <th class="border-right-bottom">Професійна Кваліфікація</th>
        <th class="border-right-bottom">Компетенції</th>
        <th class="border-right-bottom">Номер сертифікату</th>
        <th class="border-right-bottom">Факультет</th>
        <th class="border-right-bottom">Предмети:</th>
        <th class="border-bottom"> </th>
      </tr>
```

```
        </thead>
        <tbody>
            @foreach (var educationalProgram in _educationalProgramDTOs)
            {
                <tr>
                    <th class="border-right-bottom">@_counter</th>
                    <td class="border-right-bottom">@educationalProgram.Name</td>
                    <td class="border-right-
bottom">@educationalProgram.SpecialtyName</td>
                    <td class="border-right-
bottom">@educationalProgram.SpecialtyCode</td>
                    <td class="border-right-
bottom">@educationalProgram.Specialization</td>
                    <td class="border-right-
bottom">@educationalProgram.Qualification</td>
                    <td class="border-right-
bottom">@educationalProgram.ProfessionalQualification</td>
                    <td class="border-right-
bottom">@educationalProgram.Competencies</td>
                    <td class="border-right-
bottom">@educationalProgram.CertificateNumber</td>
                    <td class="border-right-
bottom">@educationalProgram.FacultyName</td>
                    <td class="border-right-bottom">
                        <NavLink class="button add-button"
@onclick="@(()=>NavigateToPage("/subjects/{0}", @educationalProgram.Id))">
                            Переглянути
                        </NavLink>
                    </td>
                    <td class="border-bottom">
                        <button class="delete-button">
                            <img src="/delete.png" class="icon " />
                        </button>
                        <button class="edit-button">
                            <img src="/edit.png" class="icon " />
                        </button>
                    </td>
                </tr>
                _counter++;
            }
        </tbody>
    </table>


}
```

```
@code {
    [Parameter]
    public Guid FacultyId { get; set; }

    private int _counter;
    private List<EducationalProgramResponseDTO>? _educationalProgramDTOs;

    protected override async Task OnInitializedAsync()
    {
        _counter = 1;
        _educationalProgramDTOs = await
service.GetEducationalPrograms(FacultyId);
    }

    private void NavigateToPage(string route, Guid educationalProgramId)
    {
        route = string.Format(route, educationalProgramId);
        Navigation.NavigateTo(route);
    }
}
```

Faculties.razor

```
@page "/university/{universityId:guid}/Faculties"
@inject IScheduleService service

<PageTitle>Faculties</PageTitle>


<AuthorizeView Roles="Administrator">
    <Authorized>
        <NavLink class="button add-button" href="/addfaculty">
            Додати факультет
        </NavLink>
    </Authorized>
</AuthorizeView>



@if (_facultyDTOs == null)
{
    <p>No data</p>
}
else
{
```

```
    <div class="title">
        <h3>@_facultyDTOs[0]!.UniversityName, Факультети:</h3>
    </div>

    <table class="schedule-table">
        <thead>
            <tr class="schedule-days">
                <th class="border-right-bottom"> </th>
                <th class="border-right-bottom">Назва</th>
                <th class="border-bottom"> </th>
            </tr>
        </thead>
        <tbody>
            @foreach (var faculty in _facultyDTOs)
            {
                <tr>
                    <th class="border-right-bottom">@_counter</th>
                    <td class="border-right-bottom">@faculty.Name</td>
                    <td class="border-bottom">
                        <button class="delete-button">
                            <img src="/delete.png" class="icon " />
                        </button>
                        <button class="edit-button">
                            <img src="/edit.png" class="icon " />
                        </button>
                    </td>
                </tr>
                _counter++;
            }
        </tbody>
    </table>


}

@code {
    [Parameter]
    public Guid UniversityId { get; set; }

    private int _counter;
    private List<FacultyResponseDTO>? _facultyDTOs;

    protected override async Task OnInitializedAsync()
    {
        _counter = 1;
```

```
        _facultyDTOs = await service.GetFaculties(UniversityId);
    }
}
```

Groups.razor

```
@page "/universities/faculties/{facultyId:guid}/groups"
@inject IScheduleService service

<PageTitle>Groups</PageTitle>

<AuthorizeView Roles="Faculty administrator">
    <Authorized>
        <NavLink class="button add-button" href="/addgroup">
            Додати групу
        </NavLink>
        <NavLink class="button add-button" href="/addstudent">
            Додати студента
        </NavLink>
    </Authorized>
</AuthorizeView>

@if (_groupDTOs == null || _groupDTOs.Count==0)
{
    <p>No data</p>
}
else
{
    <table class="schedule-table">
        <thead>
            <tr class="schedule-days">
                <th class="border-right-bottom"> </th>
                <th class="border-right-bottom">Номер</th>
                <th class="border-right-bottom">Освітня програма</th>
                <th class="border-bottom"> </th>
            </tr>
        </thead>
        <tbody>
            @foreach (var group in _groupDTOs)
            {
                <tr>
                    <th class="border-right-bottom">@_counter</th>
                    <td class="border-right-bottom">@group.Number</td>
                    <td class="border-right-
bottom">@group.EducationalProgramName</td>
                    <td class="border-bottom">
```

```
                    <button class="delete-button">
                        <img src="/delete.png" class="icon " />
                    </button>
                    <button class="edit-button">
                        <img src="/edit.png" class="icon " />
                    </button>
                </td>
            </tr>
            _counter++;
        }
    </tbody>
</table>


}

@code {
    [Parameter]
    public Guid FacultyId { get; set; }

    private int _counter;
    private List<GroupResponseDTO>? _groupDTOs;

    protected override async Task OnInitializedAsync()
    {
        _counter = 1;
        _groupDTOs = await service.GetGroups(FacultyId);
    }
}
```

Home.razor

```
@page "/"
@using ScheduleAdmin.Schedule
@using ScheduleWebAPI.Domain.DTOs
@inject IScheduleService service

<PageTitle>Universities</PageTitle>


<AuthorizeView Roles="Administrator">
    <Authorized>
        <NavLink class="button add-button" href="/adduniversity"
Match="NavLinkMatch.All">
            Додати університет
        </NavLink>
```

```
    </Authorized>
</AuthorizeView>

<div class="title">
    <h3>Університети</h3>
</div>

@if (universityDTOs == null)
{
    <p>No data</p>
}
else
{
    <table class="schedule-table">
        <thead>
            <tr class="schedule-days">
                <th class="border-right-bottom"> </th>
                <th class="border-right-bottom">Назва</th>
                <AuthorizeView Roles="Administrator">
                    <Authorized>
                        <th class="border-right-bottom">Факультети</th>
                    </Authorized>
                </AuthorizeView>
                <th class="border-bottom"> </th>

            </tr>
        </thead>
        <tbody>
            @foreach (var university in universityDTOs)
        {
            <tr>
                <th class="border-right-bottom">@_counter</th>
                <td class="border-right-bottom">@university.Name</td>
                    <AuthorizeView Roles="Administrator">
                        <Authorized>
                            <td class="border-right-bottom">
                                <NavLink class="button"
href="@($"/university/{university.Id}/Faculties")">
                                    Факультети
                                </NavLink>
                            </td>
                        </Authorized>
                    </AuthorizeView>
                <td class="border-bottom">
                    <button class="delete-button">
```

```
                <img src="/delete.png" class="icon " />
            </button>
            <button class="edit-button">
                <img src="/edit.png" class="icon " />
            </button>
        </td>

    </tr>
        _counter++;
    }
    </tbody>
</table>


}

@code {
    private int _counter;
    private List<UniversityResponseDTO>? universityDTOs;

    protected override async Task OnInitializedAsync()
    {
        _counter = 1;
        universityDTOs = await service.GetUniversities();
    }
}
```

Lessons.razor

```
@page "/groups/{groupId:guid}/lessons"
@page "/teachers/{TeacherId:guid}/lessons"
@attribute [Authorize(Roles = "Schedule administrator")]
@inject IScheduleService service

<PageTitle>Lessons</PageTitle>

<AuthorizeView Roles="Admin">
    <NavLink class="button add-button" href="#">
        Додати пару
    </NavLink>
</AuthorizeView>

@if (_lessonDTOs.Count == 0)
{
    <p>No data</p>
}
```

```
else
{
   <table class="schedule-table">
      <thead>
         <tr class="schedule-days">
            <th class="border-right-bottom"> </th>
            <th class="border-right-bottom">Викладач</th>
            <th class="border-right-bottom">Предмет</th>
            <th class="border-right-bottom">Аудиторія</th>
            <th class="border-right-bottom">День</th>
            <th class="border-right-bottom">Година початку</th>
            <th class="border-right-bottom">Тип пари</th>
            <th class="border-right-bottom">Щотижня</th>
            <th class="border-right-bottom">Перший тиждень</th>
            <th class="border-bottom"> </th>
         </tr>
      </thead>
      <tbody>
         @foreach (var lesson in _lessonDTOs)
         {
            @for (int i = 0; i < lesson.Days.Length; i++)
            {
               <tr>
                  <th class="border-right-bottom">@_counter</th>
                  <td class="border-right-bottom">@lesson.TeacherFullName</td>
                  <td class="border-right-bottom">@lesson.SubjectName</td>
                  <td class="border-right-bottom">@lesson.AudienceNumber</td>
                  <td class="border-right-bottom">@lesson.Days[i]</td>
                  <td class="border-right-bottom">@lesson.Times[i]</td>
                  <td class="border-right-bottom">@lesson.Type</td>
                  <td class="border-right-bottom">@lesson.EveryWeek</td>
                  <td class="border-right-bottom">@lesson.FirstWeek</td>
                  <td class="border-bottom">
                     <button class="delete-button">
                        <img src="/delete.png" class="icon " />
                     </button>
                     <button class="edit-button">
                        <img src="/edit.png" class="icon " />
                     </button>
                  </td>
               </tr>
               _counter++;
            }
         }
      </tbody>
```

```
    </table>
}

@code {
    [Parameter]
    public Guid GroupId { get; set; }
    [Parameter]
    public Guid TeacherId { get; set; }

    private int _counter;
    private List<LessonResponseDTO>? _lessonDTOs;

    protected override async Task OnInitializedAsync()
    {
        _counter = 1;
        if (GroupId != Guid.Empty)
        {
            _lessonDTOs = await service.GetLessonsByGroupId(GroupId);
        }
        else
        {
            _lessonDTOs = await service.GetLessonsByTeacherId(TeacherId);
        }
    }
}
```

Subjects.razor

```
@page "/subjects/{educationalprogramId:guid}"
@inject IScheduleService service

<PageTitle>Subjects</PageTitle>


<AuthorizeView Roles="Faculty administrator">
    <Authorized>
        <NavLink class="button add-button" href="addSubject">
            Додати предмет
        </NavLink>
        <NavLink class="button add-button" href="addsubjectName">
            Додати Назву предмету
        </NavLink>
    </Authorized>
</AuthorizeView>

@if (_subjectDTOs == null || _subjectDTOs.Count == 0)
```

```
{
    <p>No data</p>
}
else
{
    <table class="schedule-table">
        <thead>
            <tr class="schedule-days">
                <th class="border-right-bottom"> </th>
                <th class="border-right-bottom">Id предмету</th>
                <th class="border-right-bottom">Назва</th>
                <th class="border-right-bottom">Назва навчальної програми</th>
                <th class="border-right-bottom">Обов'язковий</th>
                <th class="border-bottom"> </th>
            </tr>
        </thead>
        <tbody>
            @foreach (var subject in _subjectDTOs)
            {
                <tr>
                    <th class="border-right-bottom">@_counter</th>
                    <td class="border-right-bottom">@subject.SubjectId</td>
                    <td class="border-right-bottom">@subject.Name</td>
                    <td class="border-right-
bottom">@subject.EducationalProgramName</td>
                    <td class="border-right-bottom">@subject.IsCompulsory</td>
                    <td class="border-bottom">
                        <button class="delete-button">
                            <img src="/delete.png" class="icon " />
                        </button>
                        <button class="edit-button">
                            <img src="/edit.png" class="icon " />
                        </button>
                    </td>
                </tr>
                _counter++;
            }
        </tbody>
    </table>


}

@code {
    [Parameter]
```

```
    public Guid EducationalProgramId { get; set; }

    private int _counter;
    private List<SubjectResponseDTO>? _subjectDTOs;

    protected override async Task OnInitializedAsync()
    {
        _counter = 1;
        _subjectDTOs = await service.GetSubjects(EducationalProgramId);
    }
}
```

Teachers.razor

```
@page "/universities/faculties/{facultyId:guid}/teachers"
@inject IScheduleService service

<PageTitle>Teachers</PageTitle>

<AuthorizeView Roles="Faculty administrator">
    <Authorized>
        <NavLink class="button add-button" href="/addteacher">
            Додати викладача
        </NavLink>
    </Authorized>
</AuthorizeView>

@if (_teacherDTOs == null || _teacherDTOs.Count == 0)
{
    <p>No data</p>
}
else
{
    <table class="schedule-table">
        <thead>
            <tr class="schedule-days">
                <th class="border-right-bottom"> </th>
                <th class="border-right-bottom">Повне ім'я</th>
                <th class="border-right-bottom">Звання</th>
                <th class="border-right-bottom">Кафедра</th>
                <th class="border-right-bottom">Пошта</th>
                <th class="border-bottom"> </th>
            </tr>
        </thead>
        <tbody>
```

```
        @foreach (var teacher in _teacherDTOs)
        {
            <tr>
                <th class="border-right-bottom">@_counter</th>
                <td class="border-right-bottom">@teacher.Firstname
@teacher.Surname @teacher.Patronymic</td>
                <td class="border-right-bottom">@teacher.Rank</td>
                <td class="border-right-bottom">@teacher.DepartmentName</td>
                <td class="border-right-bottom">@teacher.Email</td>
                <td class="border-bottom">
                    <button class="delete-button">
                        <img src="/delete.png" class="icon " />
                    </button>
                    <button class="edit-button">
                        <img src="/edit.png" class="icon " />
                    </button>
                </td>
            </tr>
            _counter++;
        }
    </tbody>
</table>


}

@code {
    [Parameter]
    public Guid FacultyId { get; set; }

    private int _counter;
    private List<TeacherResponseDTO>? _teacherDTOs;

    protected override async Task OnInitializedAsync()
    {
        _counter = 1;
        _teacherDTOs = await service.GetTeachers(FacultyId);
    }
}
```