

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

**«Система управління навчальним процесом для
закладів освіти»**

Виконав студент групи ІПЗ-20-2
БУБЕНКО Олексій Вікторович

Керівник роботи:
БАКАЛЮК Тетяна Анатоліївна

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2024

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

«ЗАТВЕРДЖУЮ»
Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БУБЕНКО Олексій Вікторович**
Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**
Тема роботи: **Система управління навчальним процесом для закладів освіти».**
затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №74/с**
Вихідні дані для роботи: **оптимізація навчального процесу в закладах освіти, зокрема процес ведення навчальних курсів.**
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	27.02.2024р.	26.02.2024р.
2	Вакалюк Т.А.	27.02.2024р.	12.05.2024р.
3	Вакалюк Т.А.	27.02.2024р.	30.05.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебдодатка платформи та пояснювальної записки. Пояснювальна записка до випускної роботи містить текстову частину, викладену на 90 сторінках друкованого тексту. Список використаних джерел містить 25 найменувань і займає 3 сторінки. У роботі наведено 44 рисунки та 47 таблиць. Також у роботі є 31 сторінок додатків. Загальний обсяг роботи – 123 сторінки.

Метою роботи є створення системи управління для закладів освіти із застосуванням мікросервісної та чистої архітектур.

В першому розділі поставлено задачу на кваліфікаційну роботу. Визначено ключовий функціонал програмного рішення, на основі порівняння з існуючими системами, а також обґрунтовано вибір інструментів та технологій розробки.

В другому розділі були визначені вимоги, змодельовані діаграми використання та класів, розроблені схеми баз даних для мікросервісів, спроектовані алгоритми, а також реалізована функціональність на серверній і клієнтській частинах з наданням важливих фрагментів програмного коду.

У третьому розділі було описано процес розгортання мікросервісної системи, користувацький інтерфейс застосунку та проведено тестування.

В результаті було розроблено програмне рішення у вигляді вебсайту з адаптивним інтерфейсом, що надає користувачам інструменти для оптимізації навчального процесу в закладах освіти.

КЛЮЧОВІ СЛОВА: НАВЧАЛЬНИЙ ПРОЦЕС, МІКРОСЕРВІСНА АРХІТЕКТУРА, НАВЧАЛЬНИЙ ЗАКЛАД, RABBITMQ

					ІПЗ.КР.Б-121-24-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Система управління навчальним процесом для закладів освіти Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Бубенко О.В.					3	90
Керівник		Вакалюк Т.А.						
Н. контр.		Тичина О.П.				Житомирська політехніка, група ІПЗ-20-2		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's thesis consists of a web-based platform application and an explanatory note. The explanatory note to the graduation thesis contains a textual part set out on 90 pages of printed text. The list of references includes 25 titles and covers 3 pages. The work contains 44 figures and 47 tables. There are also 31 pages of appendices. The total volume of the work is 123 pages.

The goal is to develop a management system for educational institutions using microservice and clean architectures.

In the first section, the task for the qualification work is set. The key functionality of the software solution is defined, based on a comparison with existing systems, and the choice of development tools and technologies is justified.

In the second section, requirements were defined, usage and class diagrams were modeled, database schemes for microservices were developed, algorithms were designed, and functionality was implemented on the server and client sides, providing important program code snippets.

The third section describes the process of deploying a microservice system, the application user interface, and testing.

The result was a software solution in the form of a website with an adaptive interface that provides users with tools to optimize the educational process in educational institutions.

KEYWORDS: EDUCATIONAL PROCESS, MICROSERVICE ARCHITECTURE, EDUCATIONAL INSTITUTION, RABBITMQ

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ ДЛЯ ЗАКЛАДІВ ОСВІТИ	10
1.1 Постановка задачі	10
1.2 Аналіз аналогів програмного продукту	12
1.3 Вибір архітектури системи управління навчальним процесом для закладів освіти	18
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	23
Висновки до першого розділу.....	27
РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ	29
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	29
2.2 Розробка бази даних системи	43
2.3 Проєктування та реалізація алгоритмів системи	56
2.4 Реалізація функціоналу системи управління навчальним процесом 58	
Висновки до другого розділу.....	68
РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СИСТЕМОЮ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ.....	69
3.1 Порядок встановлення та налаштування параметрів платформи	69
3.2 Структура інтерфейсу платформи.....	71

3.3 Тестування платформи	84
Висновки до третього розділу	86
ВИСНОВКИ	87
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	88
ДОДАТКИ.....	91

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

СУБД – Система управління базами даних

AMQP – Advanced Message Queuing Protocol

UML – Unified Modeling Language

ORM – Object-Relational Mapping

SQL – Structured Query Language

AWS – Amazon Web Services

JSX – JavaScript Syntax eXtension

SSR – Server-Side Rendering

HTTP – HyperText Transfer Protocol

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

ВСТУП

Актуальність теми. Сучасний світ переживає глобальні трансформації та зміни, які впливають на всі сфери життя, зокрема й на освіту. Сьогодні освітні заклади стикаються зі складністю навчального процесу, який вимагає постійної адаптації до нових умов та викликів. У цьому контексті система управління навчальним процесом стає головним інструментом, який дозволяє забезпечити ефективну та якісну освітню діяльність.

Система управління навчальним процесом закладу освіти — це комплекс програмних та технічних рішень, які допомагають організувати, контролювати, аналізувати навчальний процес в онлайн-режимі. Вона надає студентам та викладачам, учням та вчителям можливість отримувати та надавати освіту з будь-якого куточку світу, дозволяє зробити процес навчання більш індивідуальним, сприяє гнучкому навчанню, враховуючи потреби кожного. Також впровадження системи управління навчальним процесом може відкрити можливості для отримання освіти студентами та учнями, які перебувають на тимчасово окупованих територіях. Тому розробка такої системи є особливо актуальною в наш час.

Метою випускної кваліфікаційної роботи бакалавра є розробка системи управління навчальним процесом для закладів освіти.

Для реалізації поставленої мети потрібно виконати наступні **завдання**:

- провести аналіз аналогів системи, що розробляється та визначити їхні переваги та недоліки
- розробити архітектуру та загальну структуру системи й визначити ключові її компоненти
- вибір інструментів та технологій для реалізації системи
- реалізація функціоналу
- тестування платформи

Об'єктом дослідження є оптимізація навчального процесу в закладах освіти, зокрема процес ведення навчальних курсів.

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

Предметом дослідження є застосування сучасних веб-технологій у контексті мікросервісного та чистого архітектурних підходів для розробки системи управління навчальним процесом для закладів освіти.

За темою випускної кваліфікаційної роботи бакалавра було опубліковано тези [1].

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ ДЛЯ ЗАКЛАДІВ ОСВІТИ

1.1 Постановка задачі

Призначення системи полягає в інтеграції її в освітні заклади з метою забезпечення доступу до системи управління навчальним процесом через глобальну мережу Інтернет. Ця система дозволяє вчителям та викладачам створювати навчальні простори з різних навчальних дисциплін, завантажувати на них теоретичні матеріали та практичні завдання для отримувачів освітніх послуг.

Мета впровадження системи управління навчальним процесом для закладів освіти полягає у створенні доступного середовища для освіти, спрямованого на полегшення доступу до освітніх послуг для отримувачів освіти у навчальних закладах через глобальну мережу Інтернет. Ця платформа націлена сприяти викладачам та вчителям у створенні та розробці навчальних курсів, а учням та студентам надати можливість доступу до різноманітних навчальних матеріалів. Все це, зі свого боку, стимулюватиме покращення якості освіти та навчального процесу загалом.

Необхідна реалізація наступних кроків:

1. Аналіз функціональних вимог:

- провести аналіз потреб та очікувань різних груп користувачів системи управління навчальним процесом, таких як: студенти, опікуни, викладачі, адміністрація закладу тощо;
- визначити найважливіші функції та можливості, які повинна мати система для покращення якості навчального процесу.

2. Вибір технічних інструментів:

- обрати найбільш ефективніший підхід для розробки системи, враховуючи специфіку та складність проекту, а також ресурси та існуючі обмеження;

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

- обрати оптимальний стек технологій для розробки веб-застосунку, визначити та обґрунтувати використання фреймворку ASP.NET Core для розробки серверної логіки та API, бібліотеки React та фреймворку NextJS для клієнтської частини та бази даних PostgreSQL для зберігання даних, а також інші інструменти та бібліотеки, які будуть необхідні для реалізації функціональних вимог.

3. Розробка концепції та архітектури системи:

- розробити концептуальну модель системи, використовуючи діаграми UML, такі як діаграми прецедентів, класів, послідовності, активності тощо;
- розробити архітектуру системи, використовуючи шаблони проєктування, принципи та стандарти.

4. Реалізація серверної та клієнтської частин системи:

- реалізувати серверну частину додатку на базі фреймворку ASP.NET Core та Entity Framework Core для забезпечення надійного та швидкого API, використовуючи базу даних PostgreSQL для ефективного зберігання даних про різноманітний навчальний контент, користувачів тощо;
- реалізувати клієнтську частину додатку на базі бібліотеки React, фреймворку NextJS для створення зручного та адаптивного інтерфейсу, забезпечити користувачам можливість легкої навігації, пошуку та доступу до навчальних матеріалів.

5. Реалізація функціональності багатопрофільного акаунт та ведення навчального простору декількома викладачами:

- реалізувати функціональність декількох профілів для одного акаунту на освітній платформі, оскільки користувачі часто виконують кілька ролей у навчальному процесі. Наприклад, одна особа може бути одночасно опікуном та викладачем, і здатність легко перемикатися між цими ролями без необхідності створення окремих акаунтів значно підвищує зручність користування платформою;
- реалізувати функціональності ведення навчального простору декількома викладачами, оскільки спільне ведення навчального простору

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

декількома викладачами відповідає сучасним освітнім тенденціям, де колаборація або заміна викладачами один одного є важливим елементом освітнього процесу.

6. Тестування та оптимізація продукту:

- виконати мануальні тестування всіх функціональних компонентів. Реалізувати юніт-тести для окремих функціональних компонентів, щоб усунути помилки на ранніх етапах розробки та забезпечити стабільність і надійність програмного забезпечення;
- вдосконалення продуктивності та ефективності додатку досягається за допомогою оптимізації, що включає в себе використання кешування, мініфікацію та інші методи.

1.2 Аналіз аналогів програмного продукту

На сьогодні, в освітній сфері, системи управління навчальним процесом виступають основним інструментом для забезпечення ефективного функціонування освітніх закладів. З метою створення багатофункціональної та конкурентоздатної системи управління, що відповідає викликам сучасної освіти, важливим етапом є порівняння існуючих аналогів.

Аналіз подібних систем дозволить визначити основні функціональні можливості. Детальне порівняння аналогів управління навчальним процесом допоможе виділити важливі аспекти, такі як адміністрування, створення навчальних матеріалів, перегляд аналітичної інформації тощо. Ці аспекти надалі визначатимуть успішність та конкурентоздатність нашого програмного продукту.

«**HUMAN Школа**» — система управління навчанням, що об'єднує всіх учасників освітнього процесу та допомагає вчителям та керівникам шкіл організувати внутрішні бізнес-процеси закладу освіти [2]. Система HUMAN надає розширені можливості для централізації освітніх процесів у навчальному закладі. Зокрема, можливість ведення всієї документації в

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

електронному форматі і зосередження її на єдиній платформі спрощує рутинні завдання.

Однією з ключових особливостей цього рішення є можливість створення власної шкільної соцмережі. Адміністратори та викладачі можуть здійснювати ефективну комунікацію шляхом поширення новин, оголошень, створення подій та проведення опитувань, надаючи учасникам освітнього процесу актуальну інформацію.

Система HUMAN також відзначається можливістю моніторингу та аналізу якості освіти в закладі. Адміністратор отримує доступ до інформації про діяльність педагогічних працівників, дозволяючи переглядати навчальні плани, матеріали та домашні завдання у режимі реального часу. Завдяки аналітиці системи, адміністратор навчального закладу має змогу відстежити різноманітні показники та аспекти функціонування заклад, такі як: успішність та відвідуваність учнів, ефективність роботи вчителів та викладачів. Це забезпечує високий рівень контролю освітнього процесу і дозволяє своєчасно впливати на його якість.

Переваги:

- централізованість всіх необхідних інструментів для управління освітнім процесом;
- гнучкість для дистанційного або змішаного навчання з різними форматами;
- поглиблена аналітика про успішність та відвідуваність учнів;
- інструменти для рефлексії учнів для персоналізованого навчання та розвитку талантів;
- універсальний розклад для планування та проведення занять;
- адміністрування системи для створення та керування класами, курсами, користувачами тощо;
- підтримка та консультація від команди HUMAN для адаптації шкіл до сучасних освітніх інновацій.

Недоліки:

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

- дуже обмежена підтримка типів файлів, які можна завантажувати та переглядати на платформі;
- відсутність дати перевірки домашнього завдання вчителем;
- відсутність функціональності, яка дозволяє переглянути чи прочитане повідомлення у внутрішній соціальній мережі.

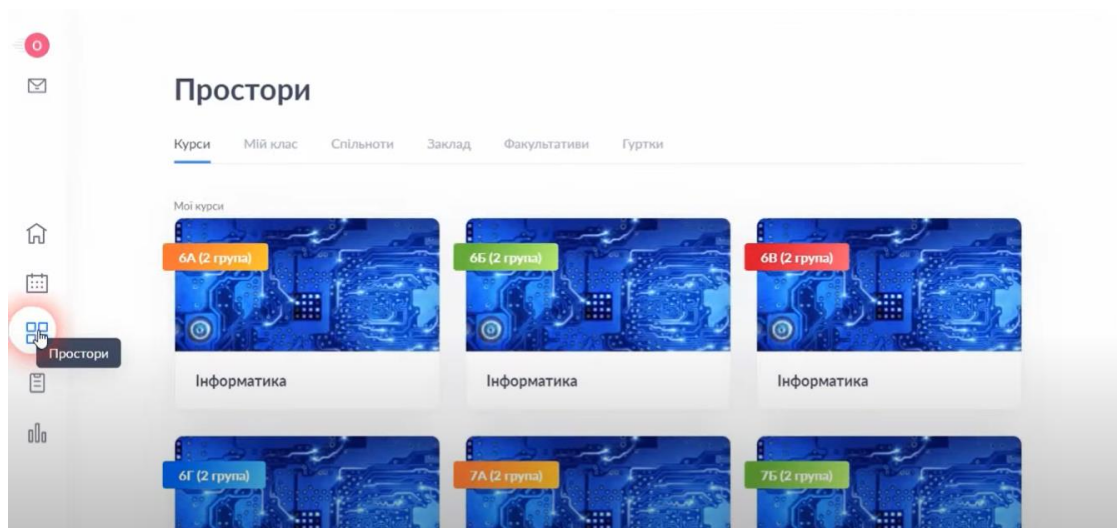


Рис. 1.1. – Інтерфейс сторінки навчальних просторів платформи «HUMAN»

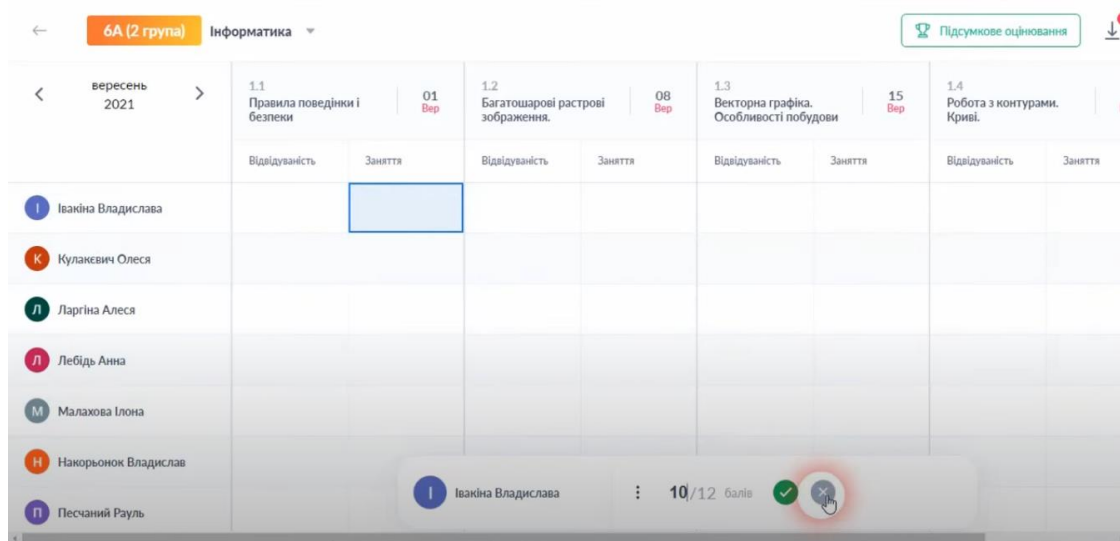


Рис. 1.2. – Інтерфейс сторінки журналу оцінок на платформі «HUMAN»

«Нові знання» — платформа для електронних щоденників та журналів з інструментами для дистанційного навчання. Надає можливості створення уроків, виставлення оцінок та аналізу успішності учнів, класів, школи.

Платформа «Нові знання» дозволяє впроваджувати щоденники, які надають учням та їхнім батькам постійний доступ до історії отриманих

оцінок та домашніх завдань. Ця функція дозволяє проводити аналіз успішності за тривалим періодом [3].

Також платформа «Нові знання» оснащена розгалуженими звітами, які забезпечують гнучкі механізми аналізу та графічного відображення навчальних досягнень учнів, класу, школи та роботи вчителя. Додатково, платформа реалізує можливості дистанційного навчання, дозволяючи виконувати, перевіряти та обговорювати практичні завдання онлайн.

Переваги:

- є можливість завантажувати власні завдання, користуватися електронними підручниками та бібліотекою уроків;
- автоматичне розрахування та формування звітів для всіх видів оцінок, включаючи тематичні, семестрові, річні оцінки;
- є можливість роздрукувати журнал або окрему сторінку предмета.

Недоліки:

- сервери можуть бути недоступними при великій кількості одночасно працюючих користувачів;
- певні елементи інтерфейсу є застарілими.

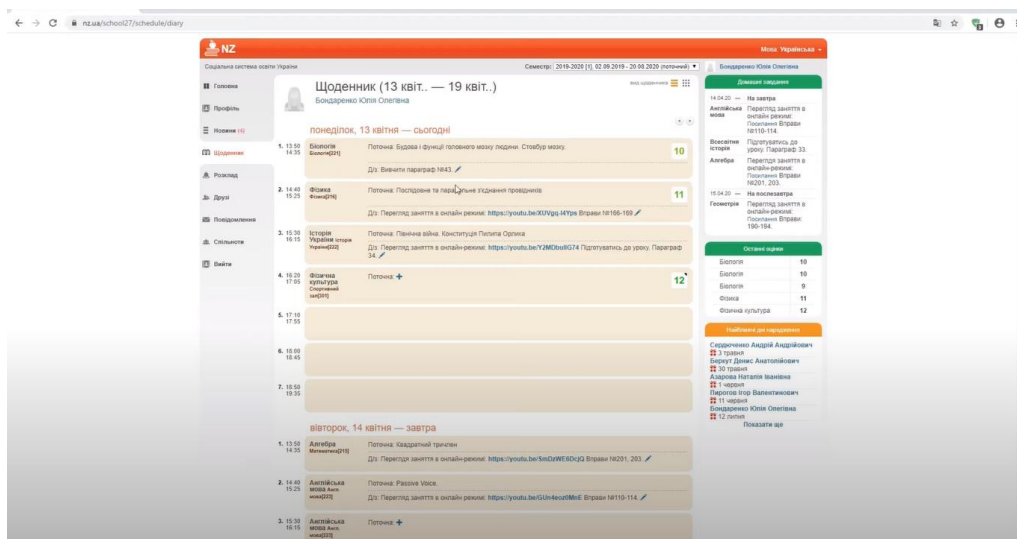


Рис. 1.3. – Інтерфейс «Нові знання»

«Smart school» — це система автоматизації освітнього процесу для закладів загальної середньої освіти, професійно-технічних навчальних закладів та вищих навчальних закладів I–II рівнів акредитації [4].

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

«Smart School» — це комплексна освітня платформа, яка спрямована на оптимізацію адміністративних процесів, покращення комунікації між учасниками освітнього процесу та забезпечення безперервного навчання. Платформа надає можливості для адміністраторів, викладачів, учнів та батьків.

Адміністратори можуть керувати розкладом занять, мають доступ до журналів успішності всіх класів, можуть переглядати звітність, керувати списками учнів, батьків та викладачів, тарифікувати викладачів та розподіляти робочий час.

Викладачі можуть працювати з журналами, зв'язуватися з батьками, розміщувати навчальні матеріали, створювати власні навчальні плани, керувати класом та створювати онлайн-тести.

Учні та їх батьки можуть отримувати актуальну інформацію про оцінки, відвідування занять, розклад занять та його зміни, домашні завдання, графіки успішності, зв'язок з вчителями та проходити онлайн-тестування.

Переваги:

- електронний журнал дозволяє автоматично генерувати звіти про успішність та зручно вести облік відвідування занять;
- редактор розкладу занять дозволяє легко формувати графік проведення занять та легко вносити зміни до нього;
- можливість формувати різноманітні типи звітів для навчальних закладів, що спрощує процес аналізу та моніторингу навчальних досягнень.

Недоліки:

- перевантажений інтерфейс таких сторінок, як журнал оцінок;
- різні посилання для входу на платформу для учнів та батьків і вчителів.

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

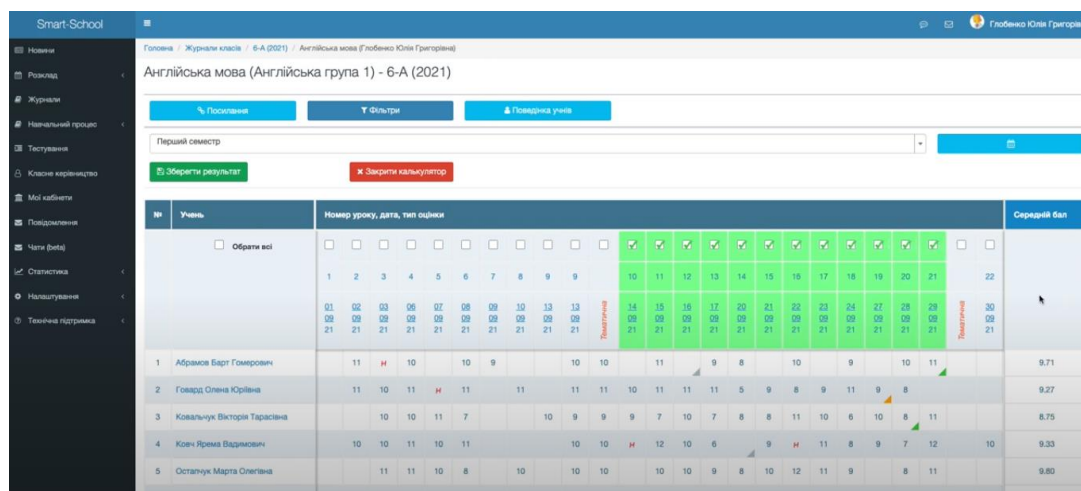
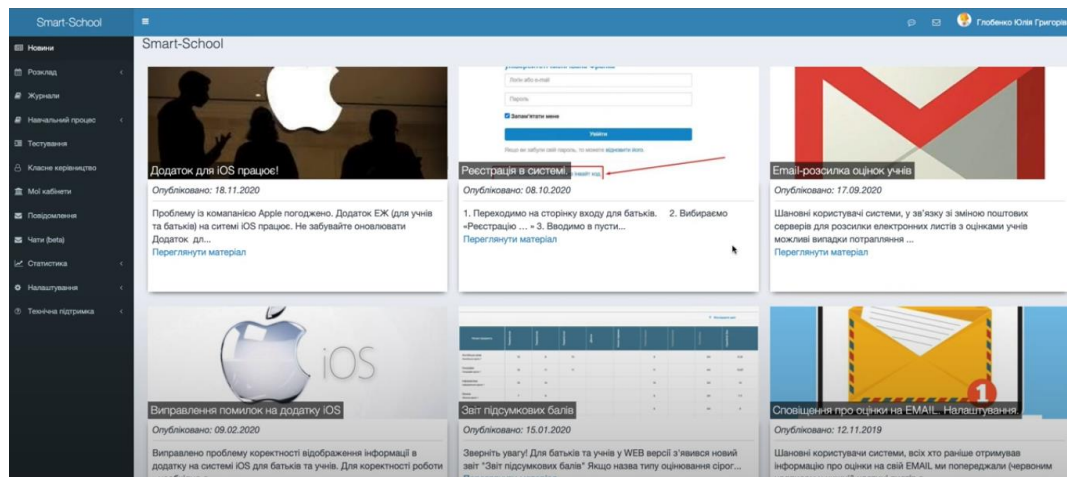


Рис. 1.5. Інтерфейс журналу оцінок «Smart school»

Таблиця 1.1

Порівняння аналогів

Характеристика	«Human Школа»	«Нові знання»	«Smart school»
Комунікація учнів та вчителів	+	-	+
Відмовостійкість	+	-	+
Система сповіщень	±	-	+
Відстеження прогресу учнів	+	+	+
Можливості тестування учнів	+	-	+
Завантаження власного контенту вчителем на сторінку курсу	+	-	+

Після аналізу освітніх платформ «Human Школа», «Нові знання» та «Smart School» вдалось виявити кілька основних характеристик для врахування при розробці нашого веб-застосунку. Отже, головні функції, на які варто звернути увагу, включають керування учнями, вчителями, навчальними курсами тощо, ведення сторінок навчальних просторів — наповнення їх теоретичними матеріалами, практичними завданнями, а також можливість для користувача мати декілька профілів.

1.3 Вибір архітектури системи управління навчальним процесом для закладів освіти

Вибір архітектури для системи управління навчальним процесом є важливим етапом, який впливає на успіх проєкту. Архітектура визначає структуру системи, включаючи способи взаємодії між різними компонентами. Вона впливає на багато аспектів системи, включаючи її продуктивність, надійність, масштабованість, безпеку тощо. Вибір архітектури — це рішення, яке впливає на довгострокову ефективність та гнучкість системи [5].

Розглянемо три основні архітектурні підходи: монолітний, мікросервісний та сервісно-орієнтований. Кожен з них має свої особливості, переваги та недоліки, які ми проаналізуємо детальніше.

Монолітна архітектура — це традиційний архітектурний підхід, де весь функціонал розташований в одному місці. У цій архітектурі всі компоненти застосунку спільно працюють із однією базою даних та іншими ресурсами, що робить розробку та розгортання простішими, але може призвести до складнощів з масштабованістю та підтримкою доступності для великої кількості користувачів [6].

Мікросервісна архітектура – це архітектурний підхід, коли система розбивається на невеликі, незалежні сервіси, які можуть бути розгорнуті окремо. Кожен сервіс відповідає за певний функціонал та має власну базу

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

даних. Ця архітектура забезпечує гнучкість, масштабованість та можливість швидкого реагування на зміни, але приносить додаткову складність у керуванні та взаємодії між сервісами [7].

Сервісно-орієнтована архітектура — це архітектурний підхід, коли функціональність розбивається на незалежні сервіси, які виконують конкретні функції та можуть бути використані в різних контекстах. Ця архітектура сприяє повторному використанню коду, забезпечує гнучкість та складається з сервісів, які можуть бути використані через систему керування пакунками. Сервісно-орієнтована архітектура може стати причиною складнощів у конфігурації, відлагодженні та керуванні залежностями між сервісами, що потребує додаткових зусиль для забезпечення ефективного функціонування системи [8].

Вибір мікросервісної архітектури для системи управління навчальним процесом обґрунтований кількома важливими факторами. Перш за все, ця архітектура забезпечує гнучкість і розширюваність, дозволяючи легко адаптувати систему до зростаючих потреб системи. Кожен логічний компонент може бути розглянутий як окремий сервіс, що дозволяє незалежно розвивати та масштабувати його.

Другим важливим аргументом є ефективне управління та підтримка системи. У мікросервісній архітектурі окремі сервіси можуть бути розроблені, протестовані та підтримувані окремо, що спрощує процес розробки та забезпечує швидке впровадження змін. Крім того, ця архітектура забезпечує більшу стабільність, оскільки проблеми в одному сервісі не впливають на решту системи. Такий підхід полегшує виявлення та виправлення помилок, що виникають у процесі роботи системи.

Отже, компоненти веб-орієнтованої мікросервісної архітектури включають перелік незалежних сервісів, які взаємодіють для забезпечення повноцінного функціоналу системи. Кожен з цих сервісів спрямований на виконання конкретної функції та може бути розгорнутий і масштабований окремо від інших [9]. Серед найважливіших компонентів можна виділити

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

сервіси автентифікації та авторизації, які відповідають за перевірку ідентифікації користувачів та надання доступу до ресурсів системи.

Додатковим важливим компонентом веб-орієнтованої мікросервісної архітектури є API шлюз. Це сервіс, який виступає в якості центральної точки входу для всіх запитів до системи. Основні функції API шлюзу включають в себе маршрутизацію запитів до відповідних мікросервісів, забезпечення автентифікації та авторизації клієнтів, перетворення та агрегацію даних та логування запитів [10].

Кожен мікросервіс має власну базу даних, що дозволяє мікросервісам мати свою власну структуру даних та зручно керувати ними, а також це сприяє зменшенню залежностей між мікросервісами [11]. Також кожен з мікросервісів може використовувати технології та схеми баз даних, які найкраще відповідають його потребам, що забезпечує оптимальну продуктивність та ефективність. Тож такий підхід дозволяє кожному сервісу бути гнучким та адаптуватися до власних конкретних вимог.

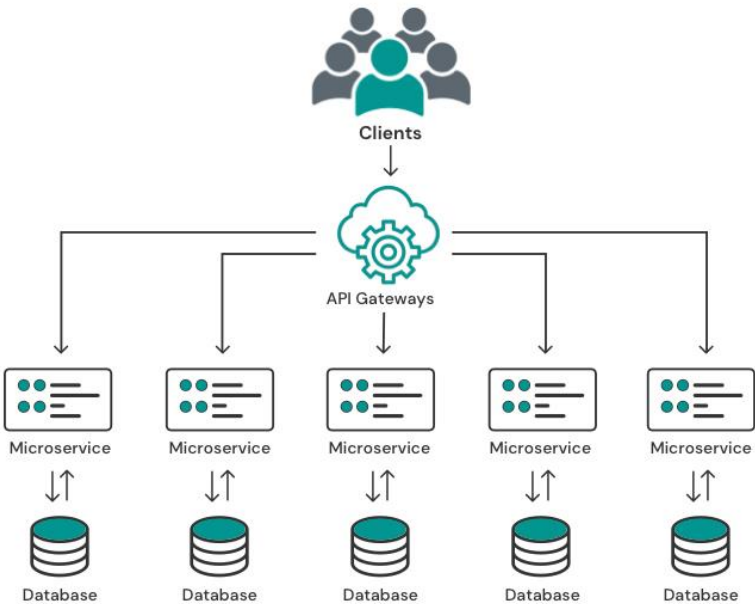


Рис. 1.6. – Мікросервісна архітектура

Важливо, щоб найбільші мікросервіси були спроектовані за певним архітектурним підходом, що дозволить ефективно підтримувати та

розширювати їх. Два поширені підходи — тришарова архітектура та чиста архітектура — займають визначальне місце серед варіацій організації програмних рішень. В обґрунтуванні вибору одного з них для проекту, потрібно ретельно зважити їхні переваги та обмеження.

Тришарова архітектура є найпопулярнішим підходом для побудови програмних систем. Цей підхід використовує розділення програми на три основні шари: презентаційний, бізнес-логіку та доступ до даних. Презентаційний шар відповідає за відображення інтерфейсу користувача та обробку вхідних запитів. Бізнес-логіка містить логіку, що визначає правила та функціональність застосунку. Шар доступу до даних відповідає за взаємодію з базою даних або зовнішніми джерелами даних. Використання такої архітектури полегшує розподіл функцій, але може призвести до взаємозалежності між шарами, що ускладнює модифікацію окремих компонентів [12].

Тришарова архітектура є популярною через свою простоту та зручність у розподілі обов'язків між компонентами програми. Проте, цей підхід може спричинити появу проблем, пов'язаних зі зростанням взаємозалежності шарів, що потенційно може ускладнити розширення та підтримку системи, що розроблюється в майбутньому.

Чиста архітектура, яка включає у себе концепцію відокремлення компонентів, спрямована на створення системи з низьким рівнем залежностей між різними її частинами. Вона складається з різних рівнів абстракції, що акцентують увагу на важливості відокремлення логіки програми на незалежні модулі. Розглянемо шари чистої архітектури детальніше.

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

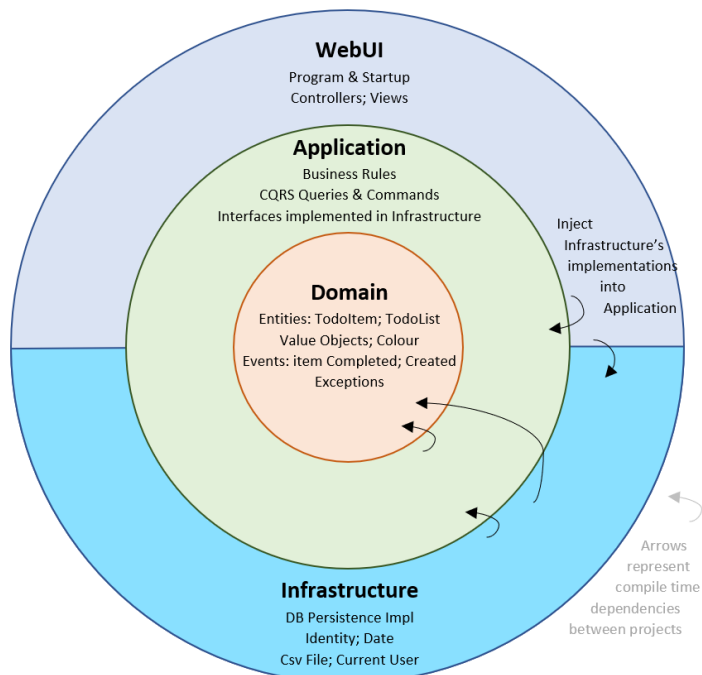


Рис. 1.7.. – Візуалізація чистої архітектури

Шар Domain включає в себе сутності та правила застосунку, що представляють унікальні для системи об'єкти та концепції. Цей шар має бути незалежним від інших, у тому числі від будь-яких сторонніх фреймворків чи бібліотек.

Шар Application містить логіку застосунку, яка відповідає за обробку користувацьких запитів на основі визначених прецедентів. Цей шар посиляється на шар Domain, виконуючи, як вже було сказано вище, визначені бізнес-логікою операції [13].

Шар Infrastructure відповідає за взаємодію із зовнішніми системами, такими як бази даних, зовнішні API та інші сервіси. Він відповідає за комунікацію зовнішніх ресурсів із застосунком.

Шар Presentation повинен реалізовувати користувацький інтерфейс та спосіб взаємодії з програмним рішенням. Це може бути веб-інтерфейс, мобільні додатки, API та інші інтерфейси призначені для взаємодії з користувачем.

У порівнянні з традиційною тришаровою архітектурою, чиста архітектура має покращену модульність та гнучкість. Тришарова архітектура, яка зазвичай включає презентаційний шар, бізнес-логіку та шар даних, може

призвести до прямих залежностей між шарами, що ускладнює зміни та розширення системи. Тоді як чиста архітектура, навпаки, використовує принцип інверсії залежностей, де зовнішні шари залежать від внутрішніх, а не навпаки. Це означає, що зміни в базі даних, користувацькому інтерфейсі або будь-яких зовнішніх фреймворках та бібліотеках не вплинуть на бізнес-логіку, яка знаходиться в ядрі архітектури. Такий підхід дозволить ефективно впроваджувати нові зміни без необхідності при цьому змінювати бізнес-логіку.

Отже, зважаючи на проаналізовані чинники, переваги та недоліки найпопулярніших архітектурних рішень для програмних систем було обрано мікросервісну архітектуру, в якій найголовніші сервіси будуть побудовані за принципами чистої архітектури.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибір інструментів та технологій є надзвичайно важливим етапом у розробці проєкту, оскільки від цього вибору залежить успішність та ефективність його виконання.

Насамперед вибір системи управління базами даних — один з найвідповідальніших кроків при розробці застосунку будь-якого рівня складності. Існує велика їх кількість, однак кожна з них має різні можливості та функціонал. У процесі вибору системи управління базами даних було прийнято рішення на користь реляційної системи. Це обумовлено кількома причинами. По-перше, реляційні бази даних вирізняються своєю надійністю та стабільністю в роботі, що є критичним для безперебійної роботи системи. По-друге, реляційні системи управління базами даних забезпечують гнучкий доступ до даних, дозволяючи ефективно працювати з різними типами інформації та виконувати складні запити. Важливо також відзначити їх масштабованість та механізми безпеки даних. Тому ці фактори

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

підтверджують обрання реляційної системи як найбільш відповідного та оптимального варіанту для такого проєкту, як система управління навчальним процесом.

У процесі вибору системи управління реляційними базами даних було розглянуто два варіанти: Microsoft SQL Server та PostgreSQL. Обидві СУБД є надійними рішеннями для роботи з реляційними даними, проте PostgreSQL була обрана через кілька вагомих причин. По-перше, PostgreSQL є об'єктно-орієнтованою системою, що дозволяє ефективно працювати з таблицями бази даних в об'єктно-орієнтованому стилі та полегшує моделювання складних даних і їх взаємозв'язків [14]. По-друге, PostgreSQL підтримує ширший спектр типів даних, що робить її більш гнучкою та адаптивною для різних технічних потреб. Крім того, PostgreSQL має активну спільноту розробників і відкритий вихідний код, що сприяє швидкому вирішенню проблем, які можуть виникнути при роботі з цією СУБД. Враховуючи ці чинники, PostgreSQL була обрана як найбільш оптимальна система управління базами даних для проєкту, що розробляється.

У якості серверної мови програмування було обрано C# з кількох вагомих причин. Ця мова програмування інтегрована з платформою .NET, що забезпечує широкий спектр для вибору бібліотек та інструментів, значно спрощуючи процес створення програмного забезпечення, надаючи готові рішення для тривіальних задач. Також мова програмування C# відзначається своєю високою продуктивністю, що є критично важливим для надійної роботи такої системи, як система управління навчальним процесом. Крім того, оскільки C# є об'єктно-орієнтованою мовою, вона чудово підходить для роботи з обраною раніше СУБД PostgreSQL, забезпечуючи ефективне моделювання даних та їх взаємозв'язків.

Оскільки система, що розробляється є веб-рішенням, було обрано фреймворк ASP.NET Core, на основі якого буде створено серверну частину застосунку. Цей фреймворк так як і мова програмування C# є інтегрованим в платформу .NET, а також дозволяє розробляти високопродуктивні,

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

кросплатформні рішення, ефективно масштабувати та підтримувати їх. Ще однією вагомою перевагою є широка екосистема та підтримка з боку компанії Microsoft, що робить його безальтернативним рішенням для нашого проєкту.

Об'єктно-реляційне відображення є найголовнішим елементом для взаємодії між серверним застосунком та базою даних у веб-розробці. Для платформи .NET існує дві основні бібліотеки, які надають функціонал ORM: Dapper та Entity Framework. Dapper є легковаговим ORM, що надає простий синтаксис та ручне керування SQL-запитами, що робить його вразливим до змін у структурі бази даних. Entity Framework, навпаки, забезпечує високий рівень абстракції та спрощує взаємодію з базою даних завдяки концепції Code First, дозволяючи працювати з об'єктами напряду через об'єктно-орієнтований програмний код. Отже, у нашому випадку було обрано Entity Framework через його здатність надавати високий рівень абстракції та автоматизованої роботи з реляційними даними, що сприяє підвищенню продуктивності розробки та зниженню ризику виникнення синтаксичних помилок на рівні SQL-запитів.

Також використаємо бібліотеку LanguageExt з метою використання переваг функціонального програмування в об'єктно-орієнтованій мові програмування C#. Застосування цієї бібліотеки дозволить більш гнучко описувати інтерфейси програмних функцій та методів. Наприклад, бібліотека LanguageExt надає можливість методам, в залежності від бізнес-логіки, повертати або успішний результат з відповідним об'єктом, або об'єкт, що представляє помилку.

Для комунікації між мікросервісами, у ролі брокера повідомлень, було обрано RabbitMQ та бібліотеку MassTransit для ефективної роботи з нею. RabbitMQ, який є одним з найпопулярніших брокерів повідомлень, підтримує стандарт AMQP, що надає інтерфейс асинхронного обміну повідомленнями між мікросервісами [15]. Вибір цих інструментів також сприяє відмовостійкості системи, оскільки повідомлення можуть бути повторно

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

доставлені у випадку збоїв, та забезпечує високу гарантію доставки повідомлень.

MassTransit надає додатковий рівень абстракції, спрощуючи інтеграцію та взаємодію з RabbitMQ. Це дозволяє зосередитися на бізнес-логіці, не вдаючись у складнощі низькорівневої взаємодії з брокером повідомлень, тим самим підвищуючи продуктивність розробки.

Для збереження файлів було використано хмарний сервіс Amazon S3. Цей сервіс дозволяє зберігати файли у хмарному сховищі Amazon Web Services (AWS) з можливістю доступу до них через мережу Інтернет. Він забезпечує надійність, швидкість для зберігання та отримання файламів, що робить його оптимальним вибором для потреб проєкту, на відміну від збереження файлів безпосередньо на сервері.

Для підтримки якості програмного коду та завчасного виявлення потенційних проблемних місць на серверній частині проєкту було обрано бібліотеку SonarAnalyzer. Ця бібліотека здійснює аналіз програмного коду з метою виявлення можливих помилок та недоліків, що дозволяє усунути або оптимізувати їх до виникнення проблем в експлуатаційній фазі. Отже, використання SonarAnalyzer сприяє підвищенню якості та надійності кодової бази [16].

Тепер визначимо інструменти та засоби розробки для клієнтської частини застосунку. Як мову програмування було обрано TypeScript, адже, на відміну від JavaScript, адже вона має статичну типізацію. Статична типізація дозволить будувати клієнтську частину застосунку, на основі чітко описаних моделей та контрактів даних. Це, зі свого боку, дозволить більш ефективно масштабувати та підтримувати проєкт [17].

Для побудови інтерфейсу застосунку були обрані бібліотека React та фреймворк Next.js. React дозволяє будувати інтерфейс на основі компонентів, використовуючи спеціальний JSX-синтаксис, що є найбільш продуктивним способом для розробки користувацького інтерфейсу. Фреймворк Next.js, як обгортка над бібліотекою React, забезпечує швидке відображення сторінок

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

завдяки підтримці Server-Side Rendering (SSR). Це означає, що HTML код сторінки генерується на сервері кожен раз, коли користувач запитує її, і це дозволяє швидше відображення контенту користувачу. Крім того, цей підхід також позитивно впливає на SEO, оскільки пошукові системи можуть краще індексувати сторінки, що веде до покращення позицій в результатах пошуку [18].

Для управління станом даних у клієнтську застосунку було обрано бібліотеку Zustand. Вона має модульний підхід, що дозволяє створювати окремі модулі стану для кожного компонента або функціональної області, у порівнянні з аналогічною бібліотекою Redux, де стан є централізованим. Це робить Zustand ідеальним вибором для мікросервісної архітектури, де кожен сервіс може мати свій власний набір даних і, відповідно, повинен мати власний стан.

Для асинхронних запитів до API було обрано бібліотеку Tanstack Query, яка, на відміну від нативних клієнтських інструментів, такі як fetch має вбудовану підтримку кешування та повторну відправку запитів на сервер для отримання актуальних даних.

Організація робочого процесу є важливим елементом розробки будь-якого великого проекту і система управління навчальним процесом не є виключенням. Для цієї мети, було обрано хостинг репозиторіїв GitLab, оскільки ця платформа надає зручні інструменти для відстеження, управління завданнями через функціональність (Issues), відображення статусів завдань на дошці задач (Issues boards), а також можливість визначення важливих етапів або цілей за допомогою Milestones.

Висновки до першого розділу

У першому розділі бакалаврської роботи було поставлено задачу на кваліфікаційну роботу бакалавра.

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

Виконано порівняння існуючих рішень з формуванням порівняльної таблиці для визначення головної функціональності системи управління навчальним процесом, що розробляється.

Було проведено аналіз та вибір архітектурного підходу для реалізації застосунку та обрано мікросервісну архітектуру, в якій найголовніші сервіси побудовані за принципами чистої архітектури.

Обрано такі технології як C#, ASP .NET, EntityFramework, брокер повідомлень RabbitMQ та реляційну базу даних PostgreSQL для серверної частини застосунку та фреймворк NextJS, бібліотеки React, Zustand, Tanstack Query для клієнтської частини.

Таблиця 2.2

Опис прецеденту «Відхиляти запити на приєднання»

Назва варіанту використання	Відхиляти запити на приєднання
Основні актори	Адміністратор
Короткий опис	Адміністратор може відхиляти запити навчальних закладів на приєднання до платформи

Таблиця 2.3

Опис прецеденту «Переглядати навчальні заклади платформи»

Назва варіанту використання	Переглядати навчальні заклади платформи
Основні актори	Адміністратор
Короткий опис	Адміністратор може переглядати всі навчальні заклади, які додані на платформу

Таблиця 2.4

Опис прецеденту «Приймати запити на приєднання»

Назва варіанту використання	Приймати запити на приєднання
Основні актори	Адміністратор
Короткий опис	Адміністратор може схвалювати запити навчальних закладів на приєднання до платформи

Таблиця 2.5

Опис прецеденту «Видаляти навчальні заклади»

Назва варіанту використання	Видаляти навчальні заклади
Основні актори	Адміністратор
Короткий опис	Адміністратор може від'єднувати навчальні заклади від платформи

Таблиця 2.6

Опис прецеденту «Керування акаунтом»

Назва варіанту використання	Керування акаунтом
Основні актори	Адміністратор, Студент, Опікун, Куратор, Викладач, Адміністратор навчального закладу, Користувач без профілю
Короткий опис	Користувачі можуть керувати власним акаунтом

Таблиця 2.7

Опис прецеденту «Зміна паролю»

Назва варіанту використання	Зміна паролю
Основні актори	Адміністратор, Студент, Опікун, Куратор, Викладач, Адміністратор навчального закладу, Користувач без профілю
Короткий опис	Користувачі можуть змінювати або відновлювати пароль до акаунту

Таблиця 2.8

Опис прецеденту «Зміна інформації про акаунт»

Назва варіанту використання	Зміна інформації про акаунт
Основні актори	Адміністратор, Студент, Опікун, Куратор, Викладач, Адміністратор навчального закладу, Користувач без профілю
Короткий опис	Користувачі можуть змінювати контактні дані свого акаунту — адресу електронної пошти або ім'я

Таблиця 2.9

Опис прецеденту «Створювати запити на приєднання»

Назва варіанту використання	Створювати запити на приєднання
Основні актори	Користувач без профілю
Короткий опис	Користувачі без профілю можуть створювати запит на приєднання навчального закладу до платформи

Таблиця 2.10

Опис прецеденту «Створити профіль за запрошувальним посиланням»

Назва варіанту використання	Створити профіль за запрошувальним посиланням
Основні актори	Користувач без профілю
Короткий опис	Користувачі без профілю можуть створювати різні види профілів за запрошувальним посиланням — опікун, студент, викладач, адміністратор навчального закладу, куратор

Таблиця 2.11

Опис прецеденту «Переглядати свій навчальний заклад»

Назва варіанту використання	Переглядати свій навчальний заклад
Основні актори	Студент, Адміністратор навчального закладу, Куратор, Викладач
Короткий опис	Користувачі можуть переглядати інформацію про навчальний заклад, до якого вони належать

Таблиця 2.12

Опис прецеденту «Зміна активного профілю»

Назва варіанту використання	Зміна активного профілю
Основні актори	Студент, Адміністратор, Опікун, Викладач, Адміністратор навчального закладу, Куратор
Короткий опис	Користувачі можуть обирати активний профіль

Таблиця 2.13

Опис прецеденту «Завантаження виконаних робіт»

Назва варіанту використання	Завантаження виконаних робіт
Основні актори	Студент
Короткий опис	Студент може завантажувати виконані практичні завдання

Таблиця 2.14

Опис прецеденту «Керувати профілями власними»

Назва варіанту використання	Керувати профілями власними
Основні актори	Студент, Опікун, Куратор, Викладач, Адміністратор навчального закладу
Короткий опис	Користувачі можуть змінювати інформацію про свої профілі

Таблиця 2.15

Опис прецеденту «Перегляд курсів дітей»

Назва варіанту використання	Перегляд курсів дітей
Основні актори	Опікун
Короткий опис	Опікут може переглядати навчальні курси своїх дітей

Таблиця 2.16

Опис прецеденту «Керування дітьми»

Назва варіанту використання	Керування дітьми
Основні актори	Опікун
Короткий опис	Опікун може додавати та видаляти дітей

Таблиця 2.17

Опис прецеденту «Керування оголошеннями в групі»

Назва варіанту використання	Керування оголошеннями в групі
Основні актори	Опікун, Студент, Викладач, Куратор
Короткий опис	Користувачі можуть створювати, видаляти та редагувати оголошення

Таблиця 2.18

Опис прецеденту «Перегляд курсу»

Назва варіанту використання	Перегляд курсу
Основні актори	Студент, Адміністратор навчального закладу, Викладач, Куратор
Короткий опис	Користувачі можуть переглядати наповнення навчального курсу

Таблиця 2.19

Опис прецеденту «Пошук оголошень в групі»

Назва варіанту використання	Пошук оголошень в групі
Основні актори	Опікун, Студент, Адміністратор навчального закладу, Викладач, Куратор
Короткий опис	Користувачі можуть використовувати пошук для того, щоб знайти потрібне оголошення

Таблиця 2.20

Опис прецеденту «Керування заняттями»

Назва варіанту використання	Керування заняттями
Основні актори	Викладач
Короткий опис	Викладач може створювати та редагувати план занять

Таблиця 2.21

Опис прецеденту «Перегляд виконання робіт»

Назва варіанту використання	Перегляд виконання робіт
Основні актори	Викладач
Короткий опис	Викладач може переглядати завантажені на перевірку роботи

Таблиця 2.22

Опис прецеденту «Керування практичними матеріалами»

Назва варіанту використання	Керування практичними матеріалами
Основні актори	Викладач
Короткий опис	Викладач може створювати практичні завдання

Таблиця 2.23

Опис прецеденту «Керування теоретичними матеріалами»

Назва варіанту використання	Керування теоретичними матеріалами
Основні актори	Викладач
Короткий опис	Викладач може створювати теоретичні параграфи та прикріплювати файли до них

Таблиця 2.24

Опис прецеденту «Генерувати запрошення для куратора»

Назва варіанту використання	Генерувати запрошення для куратора
Основні актори	Адміністратор навчального закладу
Короткий опис	Адміністратор навчального закладу може генерувати посилання для додавання куратора для групи

Таблиця 2.25

Опис прецеденту «Керування групами»

Назва варіанту використання	Керування групами
Основні актори	Адміністратор навчального закладу
Короткий опис	Адміністратор навчального закладу може створювати та редагувати навчальні групи закладу

Таблиця 2.26

Опис прецеденту «Керування курсами»

Назва варіанту використання	Керування курсами
Основні актори	Адміністратор навчального закладу, Викладач
Короткий опис	Користувачі можуть створювати навчальні курси

Таблиця 2.27

Опис прецеденту «Керування навчальними періодами»

Назва варіанту використання	Керування навчальними періодами
1	2
Основні актори	Адміністратор навчального закладу

1	2
Короткий опис	Адміністратор навчального закладу може створювати та редагувати навчальні періоди закладу

Таблиця 2.28

Опис прецеденту «Керування профілем навчального закладу»

Назва варіанту використання	Керування профілями навчального закладу
Основні актори	Адміністратор навчального закладу
Короткий опис	Адміністратор навчального закладу може переглядати та редагувати всі профілі свого навчального закладу

Отже, проаналізувавши всі варіанти використання системи управління навчальним процесом, було визначено основні потреби користувачів, які потрібно врахувати, щоб реалізувати застосунок. Відповідно було складено вимоги до системи.

Вимоги користувачів:

1. Перегляд навчальних закладів, які приєднані до платформи.
2. Створення навчальних просторів та призначати до них декількох викладачів та декілька навчальних груп.
3. Перегляд запитів навчальних закладів на приєднання та можливість схвалити або відхилити ці запити.
4. Створення декількох профілів, щоб поєднувати різні ролі — опікун, викладач тощо.
5. Можливість змінювати активний профіль, перебуваючи на будь-якій з сторінок сайту.
6. Можливість генерувати запрошувальні посилання для адміністраторів навчального закладу, викладачів, студентів та кураторів груп.

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

7. Створення навчальних груп.
8. Створення оголошення в навчальних групах та пошук по ним.
9. Розміщення плану занять на сторінці навчального простору.
10. Розміщення теоретичних матеріалів, що належать до певного заняття.
11. Створення практичних завдань.
12. Завантаження виконаних практичних завдань.
13. Перегляд виконаних студентами практичних завдань.

Функціональні вимоги:

1. Реєстрація та авторизація користувачів: можливість реєстрації та авторизації користувачів, використовуючи адресу електронної пошти та пароль.

2. Приєднання до платформи та від'єднання від неї: застосунок повинен забезпечувати можливість перегляду та обробки запитів від навчальних закладів, які мають бажання приєднатись до платформи.

3. Ведення навчального простору: застосунок повинен надавати можливість вносити зміни щодо плану занять курсу, а також створювати практичні завдання та теоретичні матеріали.

4. Завантаження виконаних завдань: система повинна надавати можливість користувачати завантажувати виконані роботи з можливістю прикріплювати файли.

Нефункціональні вимоги:

1. Сприйняття

- час, необхідний для адаптації та дослідження всіх можливостей платформи — не більше 1 години.

- час відгуку для типових запитів — не більше 5 секунд, для складних — не більше 20 секунд

- інтерфейс повинен бути простим, адаптивним та інтуїтивно зрозумілим

2. Надійність

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

- доступність
- середній час безвідмовної роботи — 15 робочих днів;
- максимальна норма помилок або дефектів — 1 помилка на 1000 запитів користувача;

3. Продуктивність

- система повинна підтримувати мінімум 300 одночасно працюючих користувачів, пов'язаних спільною базою даних.

З огляду на обраний технологічний стек, що передбачає використання засобів об'єктно-орієнтованого програмування, ключовим завданням при проєктуванні системи є створення об'єктно-орієнтованої моделі. Для її зображення використовуватимемо діаграму класів. Діаграма класів показує структуру системи шляхом зображення класів, інтерфейсів, перерахувань, атрибутів, методів та зв'язків між об'єктами діаграми [19]. Для початку розглянемо виділені доменні сутності мікросервісу, що відповідає за керування навчальними закладами.

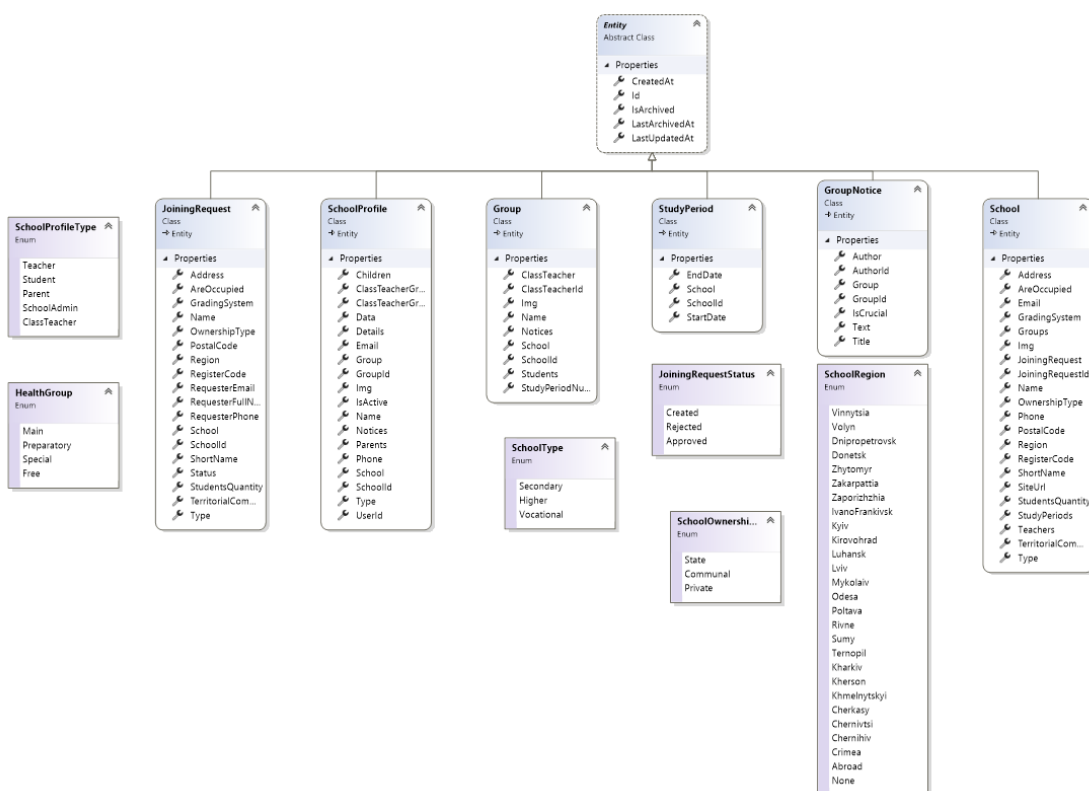


Рис. 2.2. – Діаграма класів доменного шару мікросервісу, що відповідає за керування навчальними закладами

- Entity – абстрактний клас, що містить спільні властивості для всіх доменних сутностей, такі як ідентифікатор, дати створення та оновлення. Цей клас успадковують всі класи, які представляють доменні сутності;

- JoiningRequest – клас, який представляє сутність запиту на приєднання, який використовується для обробки запитів на приєднання навчального закладу та містить детальну інформацію про заявника та власне навчальний заклад;

- SchoolProfile – клас, який представляє сутність шкільного профілю;

- Group – клас, який представляє сутність групи та призначений для керування групами учнів та кураторами цих груп;

- StudyPeriod – клас, який представляє сутність навчального періоду та визначає часові періоди навчання у навчальному закладі, включаючи дати початку та закінчення;

- GroupNotice – клас, який представляє сутність оголошення в групі та включає в себе інформацію про оголошення, включаючи автора, заголовок, текст повідомлення та чи є це оголошення критично важливим;

- School – клас, який представляє сутність навчального закладу та використовується для керування загальною інформацією про навчальний заклад, включаючи адреси, контактні дані, типи власності тощо;

- SchoolProfileType – перерахування для визначення типів профілів у школі, таких як вчитель, учень, опікун, адміністратор навчального закладу та куратор групи;

- JoiningRequestStatus – перерахування для визначення статусу запиту на приєднання навчального закладу;

- HealthGroup – перерахування для класифікації груп здоров'я студентів на наступні: основна, підготовча, спеціальна та звільнено від фізичних занять;

- SchoolType – перерахування для класифікації типів навчальних закладів;

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

- SchoolOwnershipType – перерахування для визначення типу власності навчального закладу, таких як державна, комунальна та приватна;
- SchoolRegion – перерахування для визначення регіонів, у яких знаходяться навчальні заклади.

Тепер розглянемо спроектовану діаграму класів доменного шару мікросервісу, що відповідає за керування навчальними просторами (рис. 2.3)

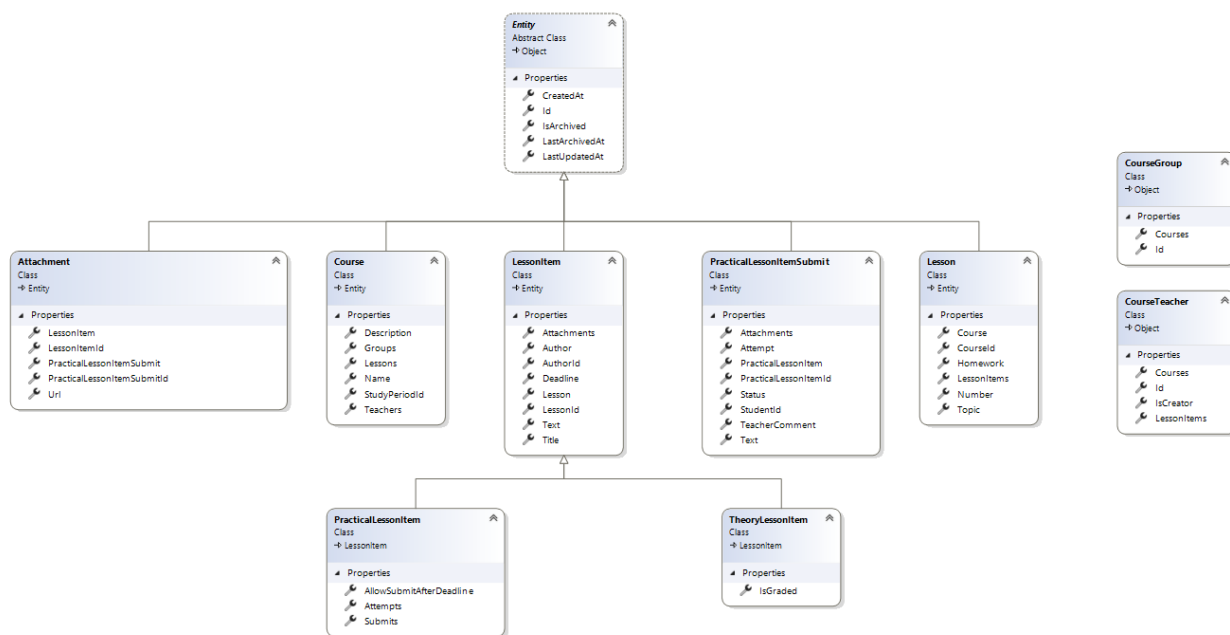


Рис. 2.3. – Діаграма класів доменного шару мікросервісу, що відповідає за керування навчальними просторами

- Entity – аналогічний до попередньої діаграми абстрактний клас, що містить спільні властивості для всіх доменних сутностей;
- Attachment – клас, який представляє сутність інформації про прикріпленний файл та використовується для зберігання посилань на зовнішнє хмарне сховище;
- Course – клас, який представляє сутність навчального простору. Він містить інформацію про групи, заняття назву та викладачів. Слід зазначити, що один навчальний простір може мати декілька викладачів та декілька навчальних груп;

- Lesson – клас, який представляє сутність заняття та використовується для організації плану занять у межах певного навчального простору;

- LessonItem – клас, який представляє сутність елементу заняття, наприклад, завдання, лекційний параграф або інші навчальні матеріали та використовується для структурування контенту занять;

- PracticalLessonItem – клас, який представляє сутність практичного елементу заняття. Він включає в себе властивості, пов'язані з виконанням практичних завдань, зокрема має властивість, щоб обмежити завантаження результатів виконання до певної дати. Цей клас успадковується від класу LessonItem;

- PracticalLessonItemSubmit – клас, що представляє сутність виконаного студентом певного практичного завдання;

- TheoryLessonItem – клас, що представляє теоретичний параграф заняття. Цей клас також успадковується від класу LessonItem і включає додаткову властивість, яка визначає, чи підлягає теоретичне завдання оцінюванню. Він використовується для структуризації теоретичних матеріалів у межах певного заняття;

- CourseGroup – клас, що представляє зв'язок між навчальними просторами та групами, зокрема містить інформацію про те, до яких навчальних просторів додана певна група;

- CourseTeacher – клас, що представляє зв'язок між навчальними просторами та викладачами й містить інформацію про те, чи є викладач автором курсу.

При створенні великих і складних програмних систем, зокрема, таких як система управління навчальним процесом, використання шаблонів проектування є критично важливим для забезпечення якості і довгострокової підтримки застосунку. Ці шаблони дозволяють стандартизувати підхід до розв'я-

зання типових завдань, що знижує ризик виникнення помилок і неочікуваної поведінки системи.

Основним з шаблонів проєктування системи було виділено CQRS (рис. 2.4), який розділяє всі операції на дві групи: зміни стану (команди) з використанням CommandContext та отримання даних (запити) через QueryContext, що дозволяє незалежно масштабувати та оптимізувати обидва види операцій.

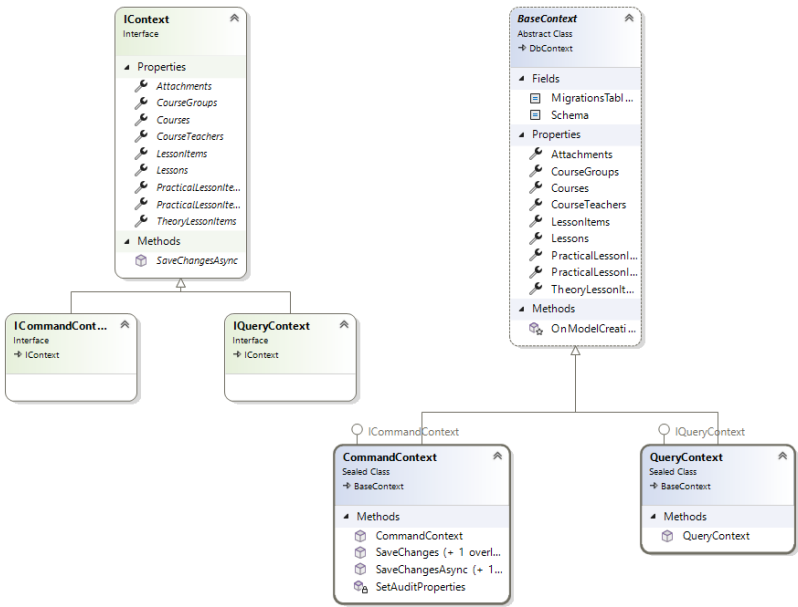


Рис. 2.4. – Реалізація шаблону проєктування CQRS

У традиційних системах, де читання та запис виконуються через один і той же контекст, можуть виникати затримки, оскільки трудомісткі операції запису можуть блокувати швидкі запити на читання. Застосування шаблону CQRS дозволяє зберігати аудитні дані, такі як дати створення та оновлення, виключно для команд, що змінюють стан системи, а клас QueryContext надає можливість виконувати операції без відстеження об'єктів, що значно поліпшує продуктивність та швидкість виконання запитів на читання даних [20].

2.2 Розробка бази даних системи

Система, що розробляється, використовує мікросервісну архітектуру, і одним з ключових підходів у цьому випадку є створення окремих баз даних для кожного з мікросервісів. Цей підхід забезпечує ізоляцію даних та

розподіл відповідальностей, що сприяє підвищенню гнучкості, масштабованості та надійності всієї системи.

Тому мікросервіси системи, що розробляється також повинні мати окремі бази даних. Розділення баз даних дозволить кожному мікросервісу зосередитися на своїх конкретних завданнях та оптимізувати доступ і обробку даних відповідно до своїх унікальних потреб.

Для мікросервісу, який керує навчальними закладами, важлива надійність бази даних, оскільки вона містить значну кількість важливої інформації, такої як структура закладів, їхні профілі та контактні дані. Ці дані мають залишатися стабільними та надійними для забезпечення ефективного функціонування системи у довгостроковій перспективі.

У той же час, для мікросервісу керування навчальними просторами, найголовнішими пріоритетами є швидкість та ефективне використання ресурсів бази даних. Це означає, що дані про навчальні простори та план занять повинні зчитуватися та оновлюватися швидко, щоб забезпечити безперервну роботу системи та задовольнити потреби користувачів.

Таким чином, в обох випадках база даних відіграє важливу роль, але з різними акцентами. Для мікросервісу навчальних закладів надійність та стабільність є ключовими, тоді як для навчальних просторів швидкість та ефективність використання ресурсів бази даних важливі для забезпечення продуктивності системи.

Для початку розглянемо таблиці бази даних, які використовує система управління навчальним процесом.

База даних мікросервісу, що відповідає за керування навчальними закладами складається з таблиць, що зображені на наступній діграмі (рис. 2.5).

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

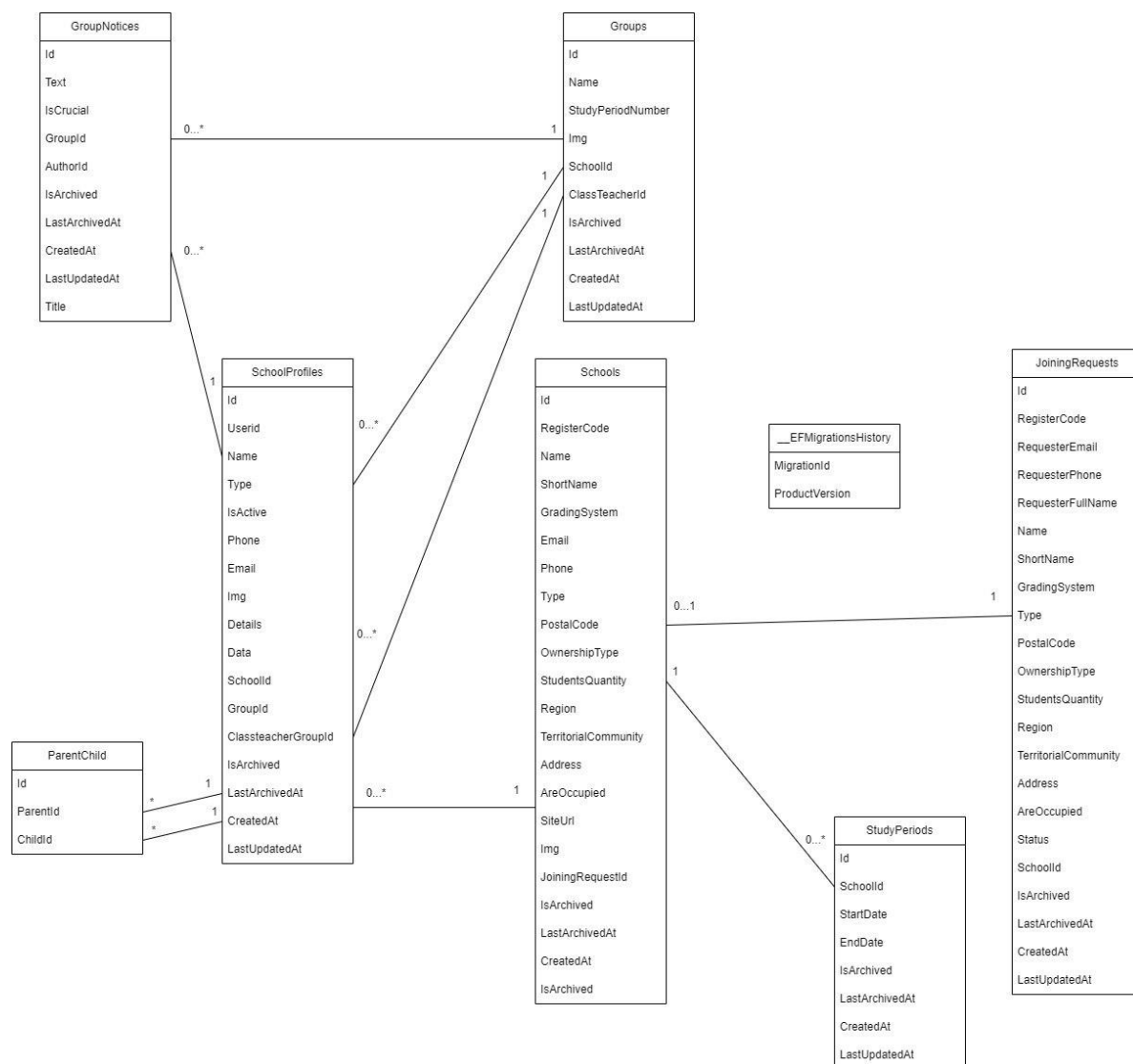


Рис. 2.5. – Діаграма бази даних мікросервісу, що відповідає за керування навчальними

- __EFMigrationsHistory — таблиця, створена фреймворком EntityFramework Core для історії міграцій;
- Schools — таблиця навчальних закладів;
- Groups — таблиця груп;
- SchoolProfiles — таблиця профілів;
- GroupNotices — таблиця оголошень групи;
- ParenChild — зв’язна таблиця для збереження опікунів та дітей;
- StudyPeriods — навчальні періоди;
- JoiningRequests — запити навчальних закладів на приєднання.

Далі наведено опис кожної з таблиць спроектованої бази даних (табли.
2.29 – 2.35)

Таблиця 2.29

Опис таблиці “Schools”

Поле	Тип	Призначення
Id	uuid	Ідентифікатор
RegisterCode	character varying	Код ЄДРПОУ
Name	character varying	Назва
ShortName	character varying	Коротка назва
GradingSystem	integer	Система оцінювання
Email	character varying	Електронна адреса
Phone	character varying	Номер телефону
Type	integer	Тип навчального закладу
PostalCode	character varying	Поштовий індекс
OwnershipType	character varying	Тип власності
StudentsQuantity	integer	Орієнтовна кількість студентів
Region	character varying	Регіон
TerritorialCommunity	character varying	Територіальна громада
Address	character varying	Адреса
AreOccupied	boolean	Відповідає за те, чи є заклад окупований
SiteUrl	character varying	Посилання на сайт закладу
Img	character varying	Зображення
JoiningRequestId	uuid	Ідентифікатор запиту на приєднання
CreatedAt	timestamp	Дата створення
LastUpdatedAt	timestamp	Дата останнього оновлення

Опис таблиці “JoiningRequests”

Поле	Тип	Призначення
1	2	3
Id	uuid	Ідентифікатор
RegisterCode	character varying	Код ЄДРПОУ
RequesterEmail	character varying	Електронна адреса ініціатора запиту
RequesterPhone	character varying	Номер телефону ініціатора запиту
RequesterFullName	character varying	Ініціали ініціатора запиту
Name	character varying	Назва навчального закладу
ShortName	character varying	Коротка назва навчального закладу
GradingSystem	character varying	Система оцінювання навчального закладу
Type	character varying	Тип навчального закладу
PostalCode	character varying	Поштовий індекс навчального закладу
OwnershipType	character varying	Тип власності навчального закладу
StudentsQuantity	integer	Орієнтовна кількість студентів навчального закладу
Region	character varying	Регіон навчального закладу
TerritorialCommunity	character varying	Територіальна громада навчального закладу

1	2	3
Address	character varying	Адреса навчального закладу
AreOccupied	character varying	Відповідає за те, чи є заклад окупований
Status	character varying	Статус обробки запиту
SchoolId	uuid	Ідентифікатор навчального закладу
CreatedAt	timestamp	Дата створення
LastUpdatedAt	timestamp	Дата останнього оновлення

Таблиця 2.31

Опис таблиці “StudyPeriods”

Поле	Тип	Призначення
Id	uuid	Ідентифікатор
SchoolId	uuid	Ідентифікатор навчального закладу
StartDate	timestamp	Початок
EndDate	timestamp	Кінець
CreatedAt	timestamp	Дата створення
LastUpdatedAt	timestamp	Дата останнього оновлення

Таблиця 2.32

Опис таблиці “SchoolProfiles”

Поле	Тип	Призначення
1	2	3
Id	uuid	Ідентифікатор

Продовження табл. 2.32

1	2	3
UserId	uuid	Ідентифікатор користувача
Name	character varying	Ім'я
Type	character varying	Тип
IsActive	character varying	Відповідає за те, чи є профіль активним
Phone	character varying	Номер телефону
Email	character varying	Електронна адреса
Img	character varying	Зображення
Details	character varying	Додаткова інформація
Data	character varying	Серіалізовані дані для кожного з типів профілів
SchoolId	uuid	Ідентифікатор навчального закладу
GroupId	uuid	Ідентифікатор групи
ClassTeacherGroupId	character varying	Ідентифікатор групи для куратора
CreatedAt	timestamp	Дата створення
LastUpdatedAt	timestamp	Дата останнього оновлення

Таблиця 2.33

Опис таблиці “ParentChild”

Поле	Тип	Призначення
1	2	3
Id	uuid	Ідентифікатор

Продовження табл. 2.33

1	2	3
ParentId	uuid	Ідентифікатор опікуна
ChildId	uuid	Ідентифікатор дитини

Таблиця 2.34

Опис таблиці “GroupNotices”

Поле	Тип	Призначення
Id	uuid	Ідентифікатор
Text	character varying	Текст
IsCrucial	boolean	Відповідає за те, чи є оголошення критичним
GroupId	uuid	Ідентифікатор групи
AuthorId	uuid	Ідентифікатор автора
CreatedAt	timestamp	Дата створення
LastUpdatedAt	timestamp	Дата останнього оновлення
Title	character varying	Заголовок

Таблиця 2.35

Опис таблиці “Groups”

Поле	Тип	Призначення
1	2	3
Id	uuid	Ідентифікатор
Name	character varying	Назва
StudyPeriodNumber	character varying	Номер навчального періоду
Img	character varying	Зображення
SchoolId	uuid	Ідентифікатор навчального закладу

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

1	2	3
ClassTeacherId	uuid	Ідентифікатор куратора
CreatedAt	timestamp	Дата створення
LastUpdatedAt	timestamp	Дата останнього оновлення

Таблиця 2.36

Опис таблиці “__EFMigrationsHistory”

Поле	Тип	Призначення
MigrationId	uuid	Ідентифікатор міграції
ProductVersion	character varying	Версія Entity Framework Core

База даних мікросервісу, що відповідає за керування навчальними просторами складається з таблиць, що зображені на наступній діграмі (рис. 2.6).

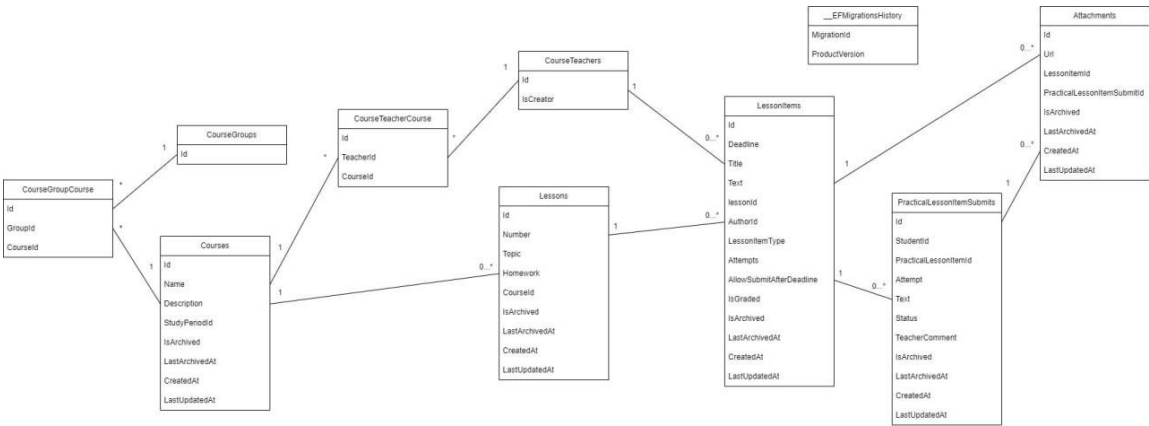


Рис. 2.6. – Діаграма бази даних мікросервісу, що відповідає за керування навчальними просторами

- __EFMigrationsHistory — таблиця, створена фреймворком EntityFramework Core для історії міграцій;
- Courses — таблиця навчальних просторів;

- CourseGroupCourse — зв’язна таблиця;
- CourseGroup — таблиця сутності групи;
- CourseTeacherCourse — зв’язна таблиця;
- CourseTeachers — таблиця сутності викладача;
- Lessons — таблиця занять;
- LessonItems — таблиця елементів занять;
- PracticalLessonItemSubmits — таблиця завантажених робіт;
- Attachments — таблиця для прикріплень.

Далі наведено опис кожної з таблиць спроектованої бази даних (табли. 2.37 – 2.46).

Таблиця 2.37

Опис таблиці “Courses”

Поле	Тип	Призначення
Id	uuid	Ідентифікатор
Name	character varying	Назва
Description	character varying	Опис
StudyPeriodId	uuid	Ідентифікатор навчального періоду
CreatedAt	character varying	Дата створення
LastUpdatedAt	character varying	Дата останнього оновлення

Таблиця 2.38

Опис таблиці “CourseGroupCourse”

Поле	Тип	Призначення
Id	uuid	Ідентифікатор
GroupId	uuid	Ідентифікатор групи
CourseId	uuid	Ідентифікатор навчального простору

Таблиця 2.39

Опис таблиці “CourseGroups”

Поле	Тип	Призначення
Id	uuid	Ідентифікатор

Таблиця 2.40

Опис таблиці “CourseTeacherCourse”

Поле	Тип	Призначення
Id	uuid	Ідентифікатор
TeacherId	uuid	Ідентифікатор викладача
CourseId	uuid	Ідентифікатор навчального курсу

Таблиця 2.41

Опис таблиці “CourseTeachers”

Поле	Тип	Призначення
Id	uuid	Ідентифікатор
IsCreator	character varying	Відповідає за те чи є викладач засновником простору

Таблиця 2.42

Опис таблиці “Lessons”

Поле	Тип	Призначення
1	2	3
Id	uuid	Ідентифікатор
Number	character varying	Номер
Topic	character varying	Тема

Продовження табл. 2.42

1	2	3
Homework	character varying	Домашнє завдання
CourseId	uuid	Ідентифікатор навчального простору
CreatedAt	character varying	Дата створення
LastUpdatedAt	character varying	Дата останнього оновлення

Таблиця 2.43

Опис таблиці “LessonItems”

Поле	Тип	Призначення
Id	uuid	Ідентифікатор
Deadline	character varying	Кінцева дата
Title	character varying	Заголовок
Text	character varying	Текст
LessonId	uuid	Ідентифікатор заняття
AuthorId	uuid	Ідентифікатор автора
LessonItemType	character varying	Тип заняття
AllowSubmitAfterDeadline	character varying	Дозволено завантажувати виконані роботи після кінцевої дати
IsGraded	character varying	Відповідає за те чи це завдання на оцінку
CreatedAt	character varying	Дата створення
LastUpdatedAt	character varying	Дата останнього оновлення

Таблиця 2.44

Опис таблиці “PracticalLessonItemSubmits”

Поле	Тип	Призначення
Id	uuid	Ідентифікатор
StudentId	uuid	Ідентифікатор студента
PracticalLessonItemId	uuid	Ідентифікатор елемента заняття
Text	character varying	Текст
CreatedAt	character varying	Дата створення
LastUpdatedAt	character varying	Дата останнього оновлення

Таблиця 2.45

Опис таблиці “Attachments”

Поле	Тип	Призначення
Id	uuid	Ідентифікатор
Url	character varying	Шлях на сервері S3
LessonItemId	uuid	Ідентифікатор елемента заняття
PracticalLessonItemSubmitId	uuid	Ідентифікатор виконаного завдання
CreatedAt	character varying	Дата створення
LastUpdatedAt	character varying	Дата останнього оновлення

Таблиця 2.46

Опис таблиці “__EFMigrationsHistory”

Поле	Тип	Призначення
1	2	3

1	2	3
MigrationId	uuid	Ідентифікатор міграції
ProductVersion	character varying	Версія Entity Framework Core

2.3 Проєктування та реалізація алгоритмів системи

При створенні якісної програмної системи важливим етапом є проєктування та реалізація алгоритмів, які вона використовуватиме. Для цього необхідно побудувати діаграми для найголовніших варіантів використання, щоб чітко визначити послідовність кроків їх виконання.

Насамперед було реалізовано алгоритм завантаження виконаного практичного завдання студентом, що зображений на діаграмі активностей (рис. 2.7).

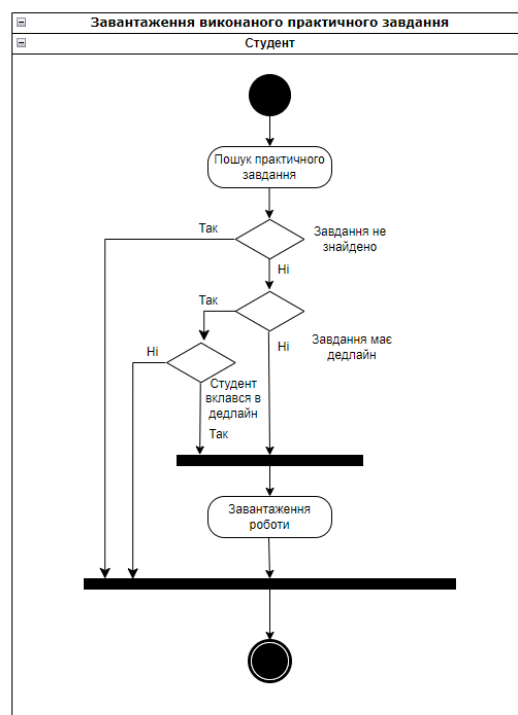


Рис. 2.7. – Діаграма активності завантаження виконаного практичного завдання

Отже, алгоритм виконання практичного завдання починається з етапу пошуку: студент шукає необхідне завдання. Якщо завдання не знайдено, система автоматично відхиляє спробу завантаження роботи. У випадку, коли завдання знайдено, наступний крок — це перевірка наявності дедлайну. Відсутність дедлайну дозволяє студенту відразу виконати завантаження своєї роботи. Якщо ж дедлайн встановлено, відбувається перевірка на те, чи завантаження відбувається вчасно. У разі дотримання термінів, робота буде завантажена. Проте, якщо студент пропустив дедлайн, спроба завантаження буде відхилена, щоб забезпечити дотримання встановлених викладачем термінів виконання завдання.

Наступним етапом є реалізація алгоритму створення та обробки запиту на приєднання навчального закладу, відображеного на діаграмі послідовності (Рис. 2.8.).

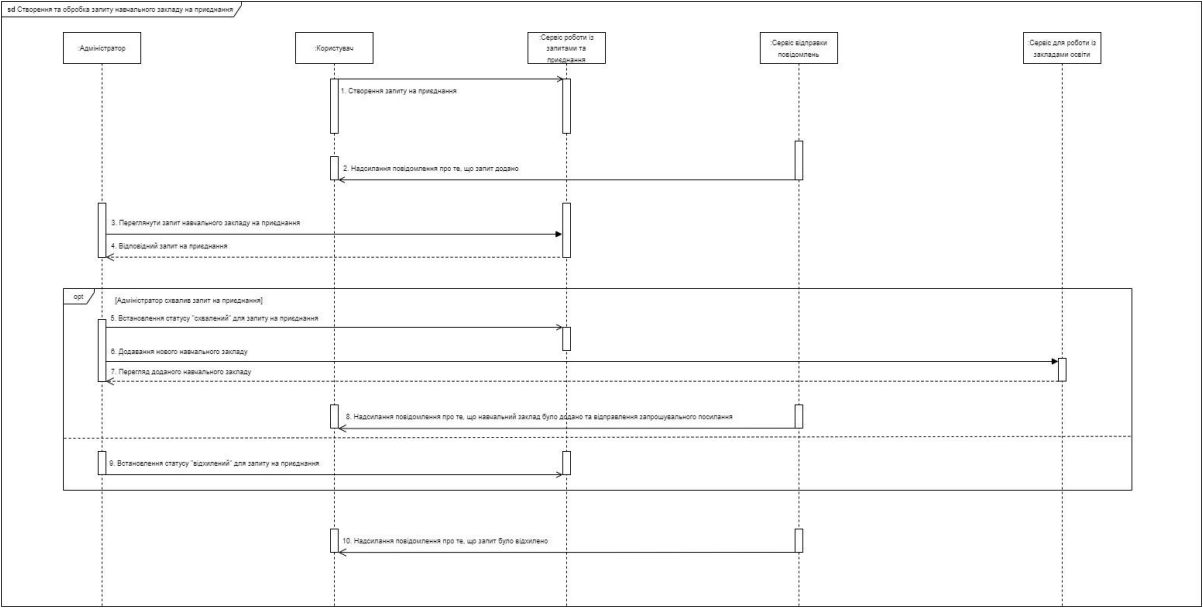


Рис. 2.8. – Діаграма послідовності створення та обробки запиту навчального закладу на приєднання

Спочатку користувач створює запит на приєднання. Якщо вхідні дані успішно проходять валідацію, відповідний сервіс створює запит на приєднання навчального закладу, а сервіс відправки електронних повідомлень надсилає повідомлення на вказану електронну адресу, в якому

зазначено, що запит успішно відправлено. Обидві ці операції виконуються асинхронно.

Адміністратор платформи надсилає запит до сервісу обробки запитів на приєднання, щоб переглянути створений користувачем запит. Адміністратор отримує відповідь з даними про цей запит. Обидві ці операції виконуються синхронно. Відбувається розгалуження залежно від рішення адміністратора щодо запиту на приєднання.

Якщо адміністратор схвалює запит, спочатку відбувається асинхронна дія встановлення статусу «схвалений». Виконуються синхронні дії: додавання нового навчального закладу шляхом надсилання команди до сервісу роботи з навчальними закладами та перегляд доданого навчального закладу викладачем. Сервіс відправки електронних повідомлень надсилає користувачу повідомлення про те, що навчальний заклад було додано до платформи, включаючи спеціальне запрошувальне посилання для доступу до роботи на платформі з власним навчальним закладом.

Якщо ж адміністратор відхиляє запит на приєднання, відбуваються асинхронні дії: встановлення статусу запиту «відхилений» та надсилання сервісом відправки електронних повідомлень інформації про те, що запит на приєднання навчального закладу було відхилено.

2.4 Реалізація функціоналу системи управління навчальним процесом

Реалізація функціональної частини системи, що розробляється включає в себе реалізацію серверної та клієнтської частин застосунку, а також програмного коду для реалізації інфраструктури, яка відповідає за збереження файлів, спільного коду в реєстрі пакетів і налаштування комунікації між мікросервісами.

Для початку розглянемо програмний метод `SetAuditProperties`, реалізований у класі `CommandContext`, який виконується для операцій на зміну да-

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

них. Цей метод встановлює властивості аудиту для всіх змінених сутностей у контексті даних, ітеруючись через всі сутності, які були змінені, та в залежності від їх стану (додані або модифіковані), оновлює відповідні часові мітки. Для доданих сутностей для властивості CreatedAt встановлюється значення поточного часу, що відображає час їх створення. Для модифікованих сутностей значення властивості LastUpdatedAt оновлюється відповідно до поточного часу, що відображає час останньої модифікації. Оскільки клас CommandContext використовується для всіх операцій на зміну даних у системі, цей підхід забезпечує єдину та централізовану логіку для відстеження створення та оновлення сутностей, що сприяє консистентності функціональності аудиту та дозволяє уникнути дублювання програмного коду в різних частинах застосунку.

```
private void SetAuditProperties()
{
    var now = DateTime.UtcNow;

    foreach (var changedEntity in ChangeTracker.Entries())
        if (changedEntity.Entity is Entity entity)
            switch (changedEntity.State)
            {
                case EntityState.Added:
                    entity.CreatedAt = now;
                    break;

                case EntityState.Modified:
                    Entry(entity).Property(x => x.CreatedAt).IsModified = false;
                    entity.LastUpdatedAt = now;
                    break;
            }
}
```

Для забезпечення валідації даних команд та запитів у програмному коді було обрано використання бібліотеки FluentValidation. Ця бібліотека надає можливість легко та ефективно виокремити логіку перевірок з програмного коду, який відповідає за бізнес-логіку, і визначити їх у вигляді простих валідаційних правил.

Нижче наведено лістинг програмного коду для валідації запити на отримання всіх груп навчального закладу, який містить перевірки на ідентифікатор користувача та властивості для пагінації.

```
public class GetAllGroupsQueryValidator : AbstractValidator<GetAllGroupsQuery>
{
```

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

```

public GetAllGroupsQueryValidator()
{
    RuleFor(x => x.UserId)
        .NotNull()
        .WithErrorCode(ErrorTitles.Common.Null)
        .NotEqual(Guid.Empty)
        .WithErrorCode(ErrorTitles.Common.Empty);

    RuleFor(x => x.StudyPeriodNumber)
        .GreaterThanOrEqualTo((byte)0)
        .WithErrorCode(ErrorTitles.Common.LowerZero);

    RuleFor(x => x.Skip)
        .NotNull()
        .WithErrorCode(ErrorTitles.Common.Null)
        .GreaterThanOrEqualTo((uint)0)
        .WithErrorCode(ErrorTitles.Common.LowerZero);

    RuleFor(x => x.Take)
        .GreaterThanOrEqualTo((uint)0)
        .WithErrorCode(ErrorTitles.Common.LowerZero);
}
}

```

Для забезпечення зручного інтерфейсу для клієнтських застосунків важним етапом є стандартизація обробки помилок. Для досягнення цієї мети реалізовано абстрактний клас `Error` та конкретні класи, які описують різні типи помилок. Це дозволяє уніфікувати формати та повідомлення про помилки, спрощуючи їх обробку для користувачів API.

Нижче наведено лістинг класів `Error` та `NotFoundError`.

```

public abstract class Error
{
    public virtual string Detail { get; set; } = "Unknown error.";
    public virtual string Title => ErrorTitles.Common.Unknown;
    public virtual string Type => ErrorTypes.Unknown;
    public virtual List<string> Params { get; set; } = new();
}

public class NotFoundError(string resourceName = "resource") : Error
{
    public override string Detail { get; set; } = $"The {resourceName} not found.";
    public override string Type => ErrorTypes.NotFound;
}

```

Для того, щоб відповідь сервера у разі неочікуваної поведінки застосунку була в одному форматі, реалізуємо фільтр `ConvertToProblemDetailsFilter`, який буде уніфікувати всі запити, що містять помилку. Фільтр перевіряє, чи є відповідь сервера помилкою, і якщо так, то

повертає об'єкт типу `ProblemDetails`, який надається платформою ASP.NET Core з відповідним статусом виконання та деталями помилки.

Нижче наведено лістинг фільтру `ConvertToProblemDetailsFilter`.

```
public class ConvertToProblemDetailsFilter : IActionFilter
{
    public void OnActionExecuting(ActionExecutingContext context)
    {
    }

    public void OnActionExecuted(ActionExecutedContext context)
    {
        if (context.Result is not ObjectResult { Value: Error error } result)
            return;

        var problemDetails = new ProblemDetails
        {
            Detail = error.Detail,
            Title = error.Title,
            Status = result.StatusCode,
            Type = error.Type,
            Extensions =
            {
                { "params", error.Params }
            }
        };

        context.Result = new ObjectResult(problemDetails) { StatusCode = result.StatusCode };
    }
}
```

Для покращення аналізу та відлагодження роботи застосунку, завдяки реалізації класу `LoggingBehavior`, було впроваджено логування всіх вхідних та вихідних запитів до системи. Це надає можливість вести детальний журнал дій користувачів і виявляти можливі проблеми та недоліки в роботі системи. Наведено лістинг класу `LoggingBehavior`.

```
public class LoggingBehavior<TRequest, TResponse>
    : IPipelineBehavior<TRequest, TResponse> where TRequest : IRequest<TResponse>
{
    public async Task<TResponse> Handle(TRequest request, RequestHandlerDelegate<TResponse> next, CancellationToken cancellationToken)
    {
        var requestName = typeof(TRequest).Name;

        Log.Information("School incoming request: {Name} {@Request}", requestName, request);

        var response = await next();
        return response;
    }
}
```

При реалізації бізнес-логіки застосунку всі прецеденти можна розділити на дві основні категорії: читання та запис. Спочатку розглянемо операції на запис, які ще часто називають командами.

Нижче подано лістинг команди `CreatePracticalLessonItemSubmit`, яка відповідає за завантаження виконаного практичного завдання.

```
public class CreatePracticalLessonItemSubmitCommand:
 IRequest<Either<PracticalLessonItemSubmitModelResponse, Error>>
{
    public Guid UserId { get; set; }

    public Guid PracticalLessonItemId { get; set; }

    public string? Text { get; set; }

    public ICollection<AttachmentModelRequest>? Attachments { get; set; }
}
```

Варто звернути увагу на те, що результуючий тип обробки команди — це монада `Either` з бібліотеки `LanguageExt`, яка дозволяє залежно від бізнес-логіки обробника команди повертати різні результати.

Ця команда потрапляє до обробника цієї команди `CreatePracticalLessonItemSubmitHandler`, який містить логіку для обробки та збереження даних, що надходять від користувача під час завантаження виконаного практичного завдання в систему. Нижче наведено фрагмент програмного коду, який відповідає за збереження результатів виконаного практичного завдання. У цьому фрагменті використовується спеціальний контекст для виконання операцій запису даних.

```
var entity =
_mapper.Map<Domain.Entities.PracticalLessonItemSubmit>(request);
entity.PracticalLessonItem = practicalLessonItem;
entity.Attempt = (uint)existedEntities.Count + 1;
entity.Status = PracticalLessonItemSubmitStatus.Submitted;
entity.StudentId = activeProfile.Id;

await _commandContext.PracticalLessonItemSubmits.AddAsync(entity, cancella-
tionToken);

try
{
    await _commandContext.SaveChangesAsync(cancellationTokens);
}
catch (Exception exception)
{
    Log.Error(exception, "An error occurred while creating the practical
lesson item submit with values {@Request}.", request);
    return new InvalidDatabaseOperationError("lesson");
}
```

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

Тепер розглянемо деталі реалізації операцій на зчитування на прикладі отримання певного навчального простору. Нижче наведено лістинг класу `GetOneCourseQuery`, який приймає ідентифікатори простору й користувача та може повернути знайдений навчальний простір або помилку.

```
public record GetOneCourseQuery(Guid Id, Guid UserId) :
    IRequest<Either<CourseModelResponse, Error>>;
```

Обробники запитів на читання даних використовують інший контекст для взаємодії з базою даних, відмінний від контексту, який використовують обробники команд. Контекст для читання не зберігає інформацію, пов'язану з аудитом, що знижує навантаження на систему. Крім того, для пришвидшення виконання запитів він не відслідковує отримані об'єкти, дозволяючи здійснювати швидкий доступ до необхідної інформації.

Нижче наведено лістинг конструктора класу `QueryContext`:

```
public QueryContext(DbContextOptions<QueryContext> options) : base(options)
{
    ChangeTracker.QueryTrackingBehavior = QueryTrackingBehavior.NoTracking;
}
```

Команди та запити надсилаються контролерами за допомогою бібліотеки `MediatR`. Використання цієї бібліотеки дозволяє значно спростити архітектуру системи, уникнувши великої кількості залежностей між різними компонентами. `MediatR` діє як посередник, який отримує запити та направляє їх до відповідних обробників та централізовано керує запитами й командами [21].

Нижче наведено лістинг базового контролера `BaseController`. Від цього абстрактного класу успадковуються всі реалізовані контролери.

```
[ApiController]
[Route("api/[controller]/")]
public abstract class BaseController : ControllerBase
{
    private IMediator? _mediator;

    protected IMediator Mediator => _mediator ??=
        HttpContext.RequestServices.GetService<IMediator>(!);
}
```

Отже, найпершими обробниками запитів до API є методи контролерів. Вони відповідають не лише за обробку вхідних запитів, але й за повернення

відповідей клієнтській стороні. Нижче наведено лістинг методу, який обробляє запит на створення запрошення для викладачів, який містить атрибут ProfileIdentify для ідентифікації активного профілю користувача, лістинг цього атрибуту наведено в Додатку А. Також можна помітити, що результат повертається клієнтській стороні за допомогою методу Match з бібліотеки LanguageExt. У випадку успішного виконання (Left), створюється HTTP-відповідь зі статусом 201 (Created) за допомогою методу CreatedAtAction. У випадку помилки (Right), викликається метод Handle з класу ActionResultHandler, який формує HTTP-відповідь з інформацією про помилку.

```
[Authorize]
[ProfileIdentify([Constants.SchoolAdmin], true)]
[HttpPost("[action]/")]
[ProducesResponseType(StatusCodes.Status201Created, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
public async Task<IActionResult> TeacherInvitation([FromBody] CreateInvitation-
Request invitationRequest)
{
    var userId = User.Identity?.GetId();
    var userRole = User.Identity?.GetRole();

    if (userId is null || userRole is null)
        return ActionResultHandler.Handle(new InvalidError("user"));

    var commandUserId = userRole == Constants.AdminRole ? null : userId;
    var command = new CreateTeacherInvitationCommand(
        invitationRequest.SchoolId,
        commandUserId
    );

    var result = await Mediator.Send(command);
    return result.Match(
        Left: response => CreatedAtAction(nameof(Create), response),
        Right: ActionResultHandler.Handle
    );
}
```

Для збереження файлів було реалізовано клас S3Service який завантажує файли на сервери сервісу S3. Нижче наведено лістинг методу GetPreSignedUrl, який генерує тимчасове посилання на завантажений файл. Повний лістинг класу S3Service розміщено в Додатку А.

```
private string GetPreSignedUrl(string bucket, string name, int
urlExpirationInMin = 100)
{
    var preSignedUrlRequest = new GetPreSignedUrlRequest
    {
        BucketName = bucket,
```

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		64


```

        Key = name,
        Expires = DateTime.UtcNow.AddMinutes(urlExpirationInMin)
    };

    var preSignedUrl = s3Client.GetPreSignedURL(preSignedUrlRequest);
    return preSignedUrl;
}

```

Для конвертування об'єктів у проєкті використовується бібліотека AutoMapper. Вона автоматизує процес конвертації властивостей між об'єктами, що значно зменшує кількість ручного коду. Нижче наведено лістинг програмного коду, який визначає глобальні правила, за якими будуть конвертуватись перерахування в рядкові значення.

```

CreateMap<Enum, string>().ConvertUsing
(e => e.ToString().ToSnakeCase());

```

Операції з сутністю профіля було реалізовано в класі SchoolProfileManager, повний лістинг якого знаходиться в Додатку А. Одне із найголовніших завдань цього класу – це забезпечення кешування профілів. Це важливо, оскільки активний профіль користувача запитується в більшості наявного функціоналу. Кешування профілів дозволяє зберігати в пам'яті інформацію про активний профіль і використовувати її безпосередньо з кешу, замість постійних запитів до бази даних. Це сприяє покращенню продуктивності та швидкодії роботи системи, оскільки уникнуто зайвих навантажень на джерело даних і зменшено час очікування відповіді.

Нижче наведено лістинг методу CacheProfiles, який виконує кешування профілю.

```

public async Task<SchoolProfileModelResponse?> CacheProfiles(Guid userId, Guid
currentProfileId)
{
    ClearCache(userId);
    var cacheKey = GetCacheKey(userId);

    var userProfiles = await _commandContext.SchoolProfiles
        .Where(profile => profile.UserId == userId)
        .ToListAsync();

    if (!userProfiles.Any())
    {
        _memoryCache.Set<IEnumerable<SchoolProfileModelResponse?>>(cacheKey,
null);
        return null;
    }

    ActivateProfile(userProfiles, currentProfileId);
}

```

```

var profilesToCache = await MapToResponses(userProfiles);
_memoryCache.Set(cacheKey, profilesToCache);

return profilesToCache?.FirstOrDefault(p => p.IsActive);
}

```

Для того, щоб надсилати захищені посилання на приєднання, було використано криптографічний метод AES та відповідно реалізовано клас AesChipher, повний лістинг якого наведено в Додатку А. Нижче наведені конструктор класу AesChipher та лістинг методу Encrypt, який виконує шифрування вхідного рядка.

```

public AesCipher(IConfiguration configuration)
{
    var encryptionKeyBase64 = configura-
tion["EncryptionSettings:EncryptionKeyBase64"]!;
    _encryptionKeyBytes = Convert.FromBase64String(encryptionKeyBase64);

    var initializationVectorBase64 = configura-
tion["EncryptionSettings:InitializationVectorBase64"]!;
    _initializationVectorBytes = Con-
vert.FromBase64String(initializationVectorBase64);
}

public string Encrypt(string plainText)
{
    using var aes = AesAlgorithm.Create();
    aes.Key = _encryptionKeyBytes;
    aes.IV = _initializationVectorBytes;

    var encryptor = aes.CreateEncryptor(aes.Key, aes.IV);
    var plainBytes = Encoding.UTF8.GetBytes(plainText);

    using var memoryStream = new MemoryStream();
    using (var cryptoStream = new CryptoStream(memoryStream, encryptor, Cryp-
toStreamMode.Write))
    {
        cryptoStream.Write(plainBytes, 0, plainBytes.Length);
    }

    return Convert.ToBase64String(memoryStream.ToArray());
}

```

У мікросервісній архітектурі важливим етапом є реалізація API. У нашому випадку API шлюз реалізовано за допомогою Reverse Proxy. Reverse Proxy – це тип проксі-сервера, який використовується в комп'ютерних мережах для обробки запитів, що надходять із зовнішнього середовища до внутрішньої мережі. Він діє як посередник між зовнішніми клієнтами і внутрішніми серверами, приховуючи внутрішню структуру мережі від зовнішніх користувачів [22].

Основна перевага використання Reverse Proxy у ролі API шлюзу полягає у тому, що він дозволяє централізовано керувати всіма зовнішніми запитами до системи. Нижче наведено лістинг налаштування для запитів, які стосуються операцій з навчальними закладами.

```
school": {
  "ClusterId": "school",
  "AuthorizationPolicy": "default",
  "Match": {
    "Path": "/school/{**catch-all}"
  },
  "Transforms": [
    {
      "PathPattern": "api/school/{**catch-all}"
    }
  ]
}
```

На клієнтській частині застосунку розглянемо надсилання запитів до серверу. Нижче наведено лістинг функції create, яка виконує HTTP-POST запит до API.

```
async function create(
  url: string,
  body: any,
  isFormData = false,
  schoolProfileId = null,
) {
  const requestOptions = {
    method: 'POST',
    headers: await getHeaders(isFormData, schoolProfileId),
    body: isFormData ? body : JSON.stringify(body),
  };

  const response = await fetch(baseUrl + url, requestOptions);
  return await handleResponse(response);
}
```

Ця функція виконує асинхронний HTTP POST-запит до вказаного URL-адреси API-шлюзу. Спочатку створюється об'єкт requestOptions, який містить метод, заголовки та тіло запиту, що формується з урахуванням параметра isFormData. Потім виконується HTTP-запит до сервера, отримана відповідь обробляється та повертається як результат виконання функції create. Таким чином, функція відправляє дані до сервера, обробляє отриману відповідь та повертає результат запиту.

Для забезпечення можливості зміни активної вкладки на всіх сторінках додатку реалізовано функцію useUserStore, яка використовує інструменти бібліотеки Zustand для збереження стану.

Лістинг функції useUserStore наведено нижче

```
export const useUserStore = create<UserStore>()(
  immer(
    devtools(
      (set) => ({
        currentTab: CurrentTab.Profile,
        setCurrentTab: (tab: CurrentTab) =>
          set((state) => {
            state.currentTab = tab;
          }),
      })),
      { name: 'User store' },
    ),
  ),
);
```

Висновки до другого розділу

Під час розробки системи управління навчальним процесом було визначено найголовніші прецеденти та вимоги до розробленої системи. На основі цих вимог було змодельовано діаграму використання. Після визначення варіантів використання було розроблено діаграму класів на основі визначених доменних сутностей та виділено шаблони проєктування для роботи з ними.

Найважливішим етапом була розробка схем баз даних для двох найголовніших мікросервісів та опису зв'язків між описаними таблицями.

Наступним етапом було проєктування алгоритмів системи, які представлені діаграмами активності та послідовності.

У результаті, відповідно до визначених вимог, було реалізовано функціональність системи як на серверній, так і на клієнтській частинах та наведено найголовніші фрагменти програмного коду.

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СИСТЕМОЮ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

3.1 Порядок встановлення та налаштування параметрів платформи

Система, що розробляється, використовує мікросервісну архітектуру, тому кожен з мікросервісів може бути розміщений на окремому сервері. Це означає, що кожен мікросервіс працює у власному середовищі та не залежить від інших мікросервісів, що робить більш ефективними процеси керування та масштабування.

Для кращого розуміння процесу розгортання було змодельовано діаграму використання (рис. 3.1)

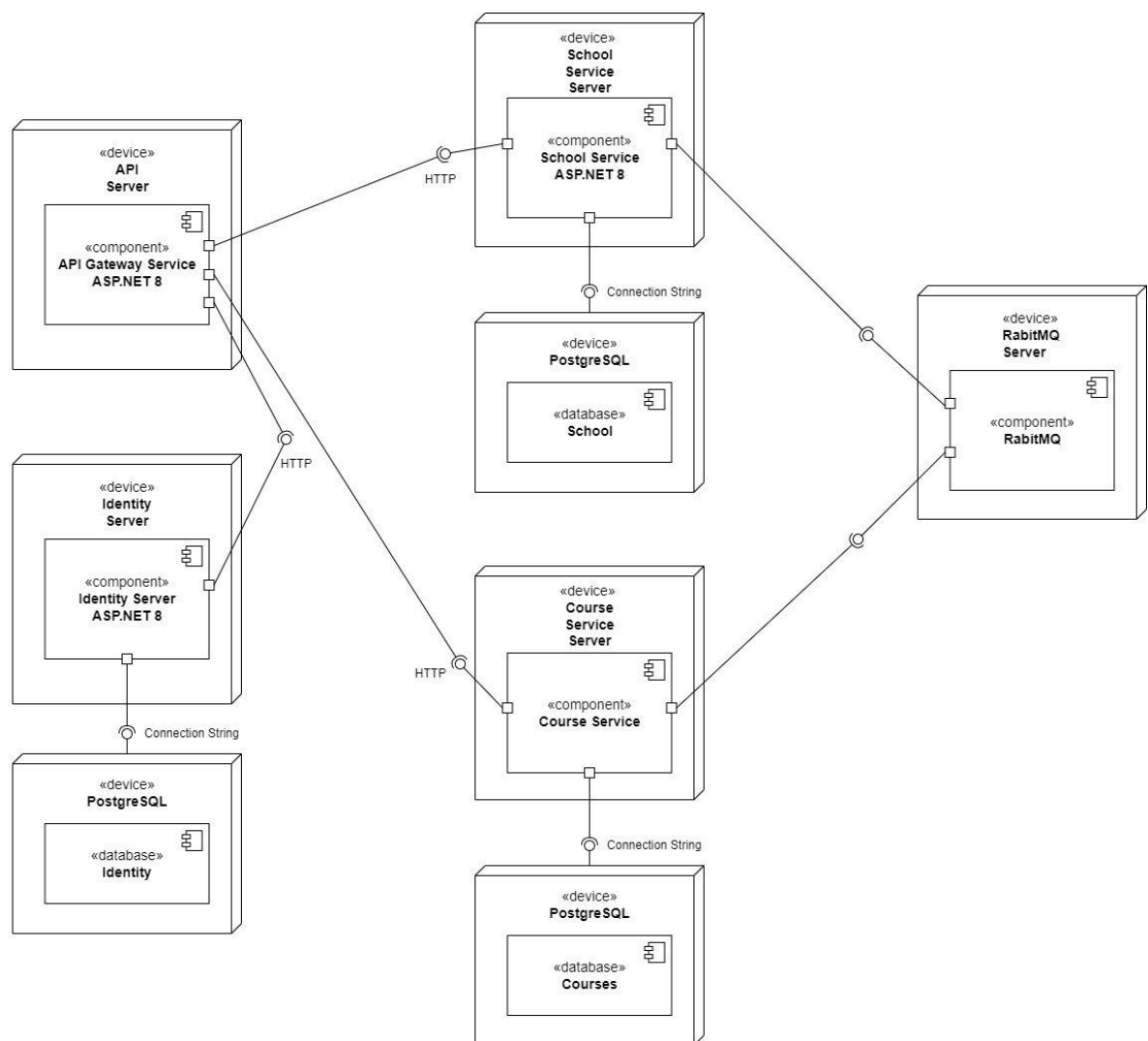


Рис. 3.1. Діаграма розгортання

На діаграмі було представлено кілька серверів, на яких розміщено окремі мікросервіси, а також сервери баз даних та сервер брокера повідомлень.

Ізоляція дозволяє легко виявляти та виправляти помилки, оскільки збої в одному мікросервісі не впливають на роботу інших. Незалежність роботи мікросервісів також означає, що кожен з них може бути розроблений, розгорнутий і оновлений окремо, без необхідності зупиняти або змінювати інші частини системи. Це полегшує впровадження нового функціоналу, а також підвищує стійкість системи до відмов [23].

Розміщення мікросервісів на окремих серверах також сприяє кращому розподілу ресурсів, оскільки кожен мікросервіс може бути налаштований для оптимального використання обчислювальних потужностей, пам'яті та інших ресурсів, що дозволяє ефективніше керувати навантаженням і забезпечувати високу продуктивність усієї системи.

Розгорнути серверну частину системи необхідно на сервері з такими мінімальними характеристиками:

- чотириядерний процесор з тактовою частотою 3 ГГц або вище;
- 8 ГБ оперативної пам'яті;
- 16 ГБ вільного місця на жорсткому диску.

Для розгортання серверної частини програмної системи потрібно виконати наступні операції.

1. Завантажити та встановити .NET SDK з офіційного сайту Microsoft. Це дозволить компілювати та запускати ASP.NET додатки.

2. Виконати команду `dotnet publish -c Release -o out` для компіляції мікросервісу API Gateway.

3. Скопіювати вміст папки `out` на сервер, де буде розміщено цей мікросервіс.

4. Скопіювати вміст папки `out` на сервер для Identity Service.

5. Налаштувати з'єднання з базою даних PostgreSQL (Identity) через рядок підключення.

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

6. Виконати команду `dotnet publish -c Release -o out` для компіляції мікросервісу School Service.

7. Скопіювати вміст папки out на сервер для School Service.

8. Налаштувати з'єднання з базою даних PostgreSQL (School) через рядок підключення.

9. Виконати команду `dotnet publish -c Release -o out` для компіляції мікросервісу Course Service.

10. Скопіювати вміст папки out на сервер для Course Service.

11. Налаштувати з'єднання з базою даних PostgreSQL (Course) через рядок підключення.

12. Встановити Docker, якщо він ще не встановлений.

13. Завантажити RabbitMQ за допомогою Docker, виконавши команду `docker pull rabbitmq:3-management`.

14. Запустити контейнер RabbitMQ: `docker run -d --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3-management`

15. Виконати команду `dotnet run` для запуску кожного мікросервісу після розгортання на сервері.

Розроблена система, для взаємодії з користувачем, підтримує використання будь-якого сучасного веб-браузера, включаючи Google Chrome (версія 45 і вище), Mozilla Firefox (версія 38 і вище), Microsoft Edge, Safari та інші популярні браузери.

3.2 Структура інтерфейсу платформи

Взаємодія з системою управління навчальним процесом для закладів освіти здійснюється через веб-браузер.

Перейшовши на головну сторінку, користувач бачить банер та має можливість зареєструватись, увійти або створити запит на приєднання навчального закладу.

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

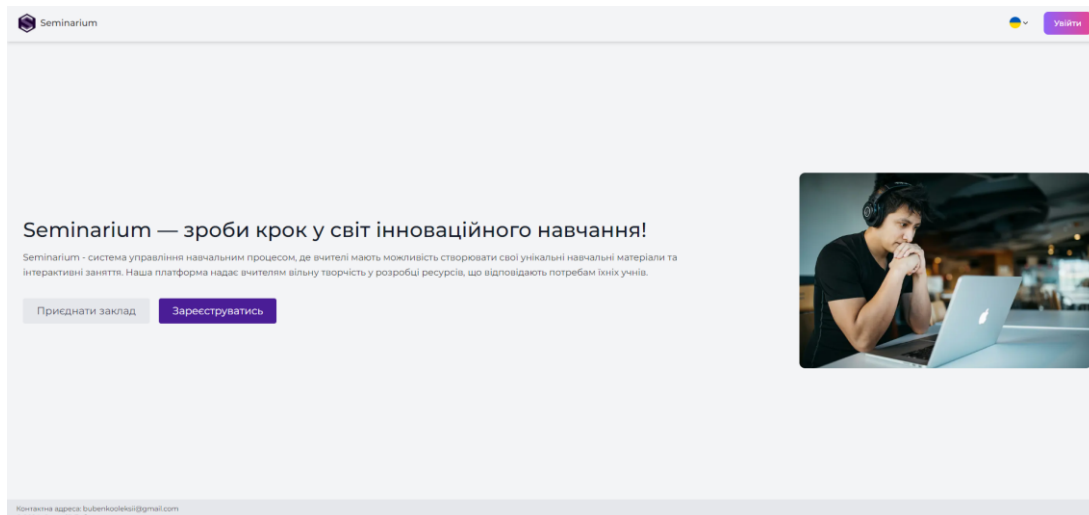


Рис. 3.2. – Головна сторінка системи

Натиснувши кнопку «Зареєструватись», користувачу відображається форма реєстрації (рис. 3.3). Якщо користувач ввів коректні дані, то акаунт буде створено, а користувачу потрібно буде підтвердити свою реєстрацію, перейшовши за посиланням, яке буде надіслано на електронну пошту.

Рис. 3.3. – Форма реєстрації

У випадку, коли користувач здійснює вхід до системи та забув пароль, система надає можливість його відновити, заповнивши певну форму (рис. 3.4)

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

Лист для відновлення паролю надіслано на **ipz202_bov@student.ztu.edu.ua**

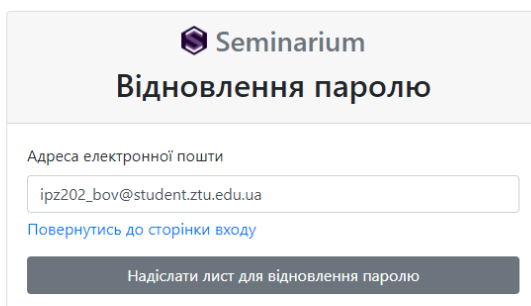


Рис. 3.4. – Відновлення паролю

Як у авторизованого, так і у не авторизованого користувача є можливість створити запит на приєднання навчального закладу, заповнивши форму (рис. 3.5.)

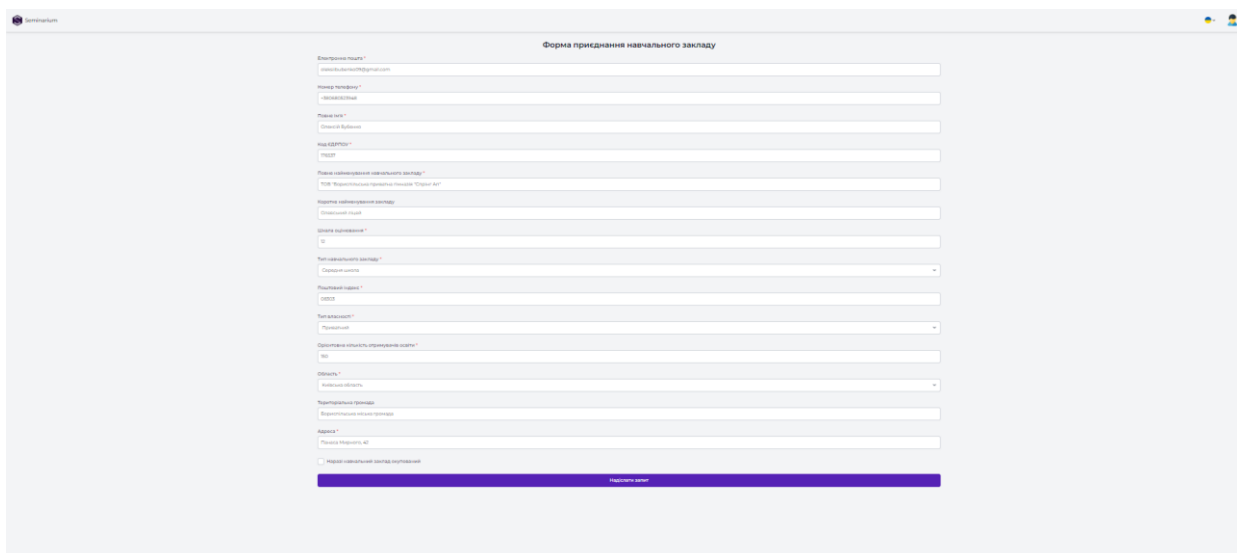


Рис. 3.5. – Створення запиту на приєднання навчального закладу

Адміністратор платформи має можливість переглядати запити на приєднання на відповідній сторінці (рис. 3.6)

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

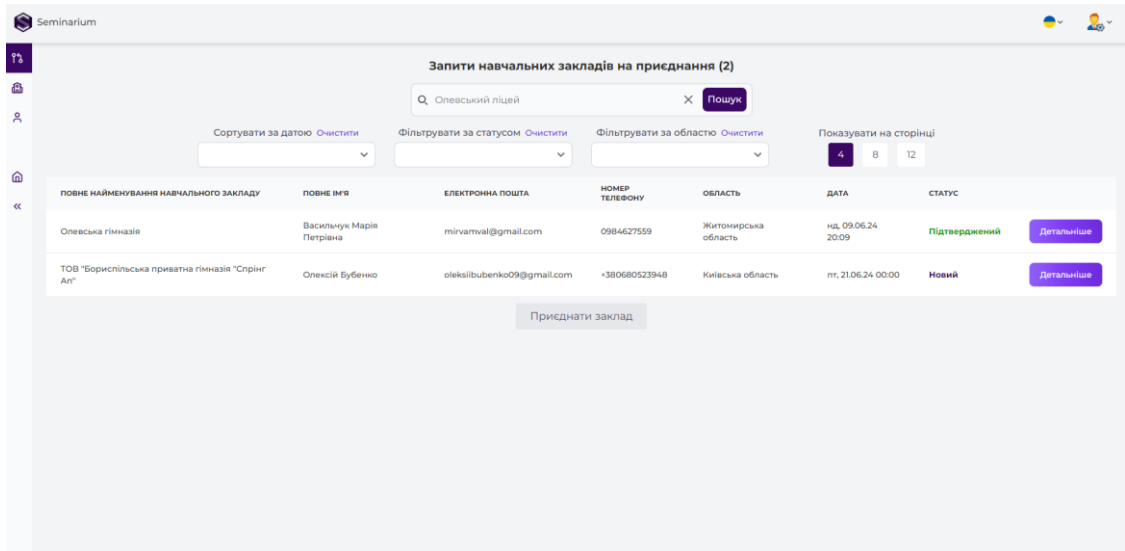


Рис. 3.6. – Перегляд запитів на приєднання зі сторони адміністратора

Перейшовши до деталей, адміністратор може переглянути інформацію про навчальний заклад, який має намір додаться до платформи, в держреєстрі, відхилити або схвалити запит на приєднання на окремій сторінці (рис. 3.7).

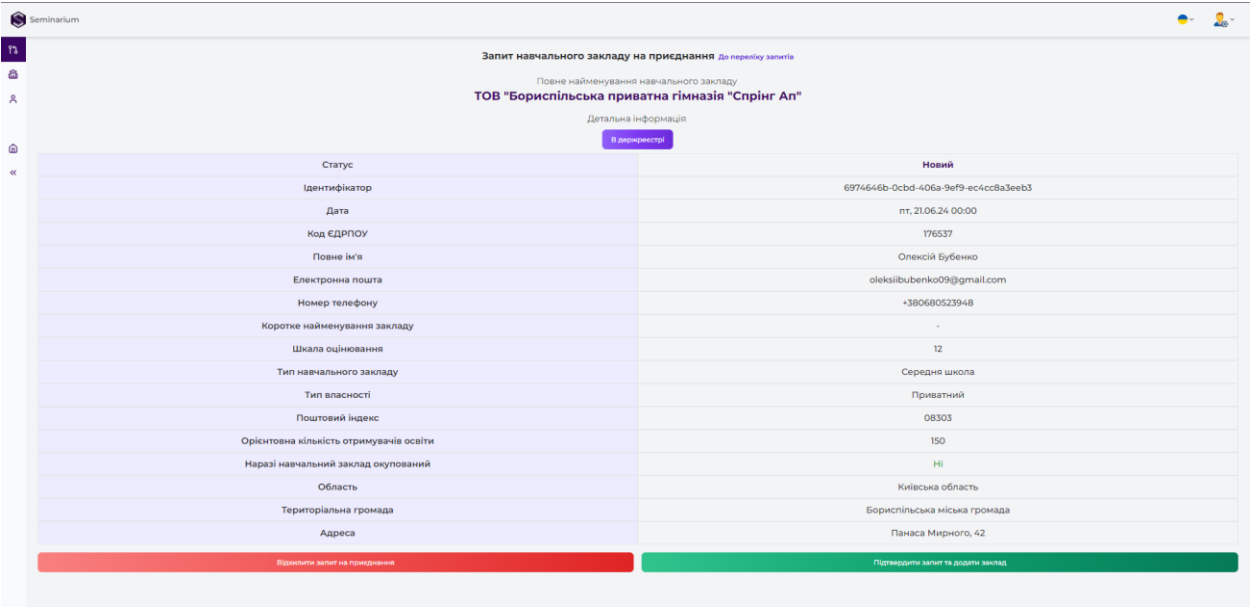


Рис. 3.7. – Перегляд запиту на приєднання

Якщо адміністратор системи вирішить схвалити запит на приєднання, то перед ним з'явиться форма для створення навчального закладу, в якій продублюються значення із запиту на приєднання. Після створення навчального закладу буде відображено його сторінку (рис. 3.8).

Навчальний заклад до вступу на навчання

Під час набірних конкурсів надсилайте дані заводу

ТОВ "Бориспільська приватна гімназія "Світанок АРТ"

Інформація про підприємство-засновника

Згенерувати інформацію для банківської форми

Детальніша інформація

Увага!

Код ЄДРПОУ	770527
коротке найменування заводу	-
Штатне підприємство	12
Тип навчального закладу	Середня школа
Тип власності	Приватний
Податковий індекс	08303
Кількість організованих освітян	150
Область	Київська область
Територіальна громада	Бориспільська міська громада
Адреса	Гімназія Миколицька, 42
Дата припинення	вт. 20.06.24 10:03
Ідентифікатор	актвсблр-тфпр-маш-ассн-ГДХ301640eac
Електронна пошта	-
Номер телефону	-
Сайт заводу	-
Наразі навчальний заклад функціонує	НІ


Вибрати заклад



Детальніше про підприємство

Детальніша інформація

Рис. 3.8. – Доданий навчальний заклад після схвалення запиту на приєднання

У адміністратора платформи є можливість згенерувати запрошувальне посилання для адміністратора навчального закладу, а також запрошувальне посилання для адміністратора навчального закладу автоматично надсилається при схваленні запиту. Перейшовши за цим посиланням, користувач бачить форму (рис. 3.9) та може створити профіль, який буде прив'язано до його навчального закладу.

 Seminarium



Створення профілю адміністратора навчального закладу

Ім'я (обов'язково)

Олексій Бубенко

Телефон (необов'язково)

Електронна пошта (необов'язково)

lp202_bov@student.stu.edu.ua

Корисні деталі про Вас (необов'язково)

Маю вади слуху...

Створити профіль

Контактна адреса: bubenkodekisi@gmail.com

Рис. 3.9. – Створення профілю адміністратора навчального закладу

Після створення профілю, адміністратор навчального закладу матиме змогу керувати групами, зокрема створювати (рис. 3.10), видаляти та редагувати групи.

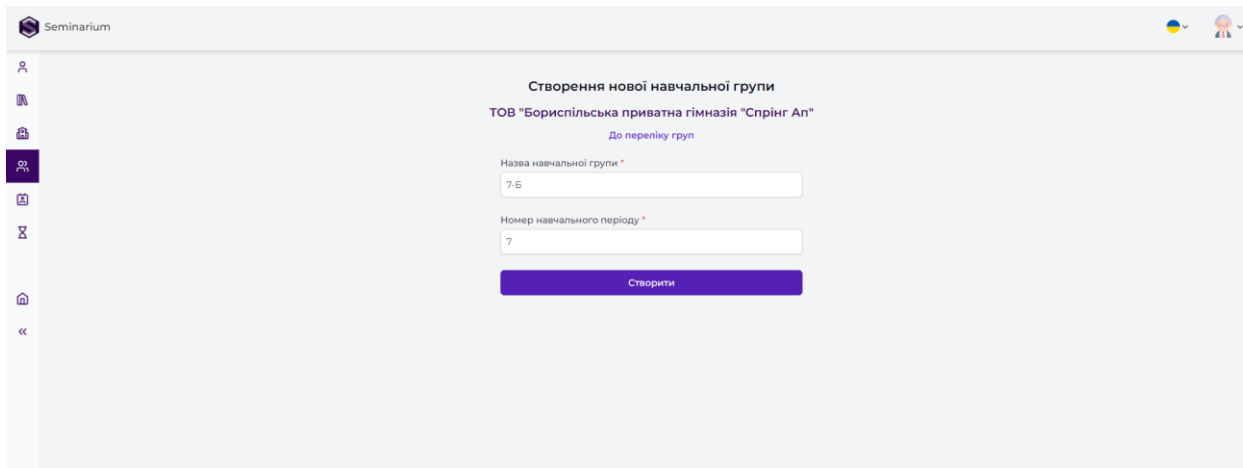


Рис. 3.10. – Створення навчальної групи

Також у адміністратора навчального закладу є можливість переглядати всі навчальні групи, які є на платформі (рис. 3.11), обравши відповідний пункт у бічному меню.

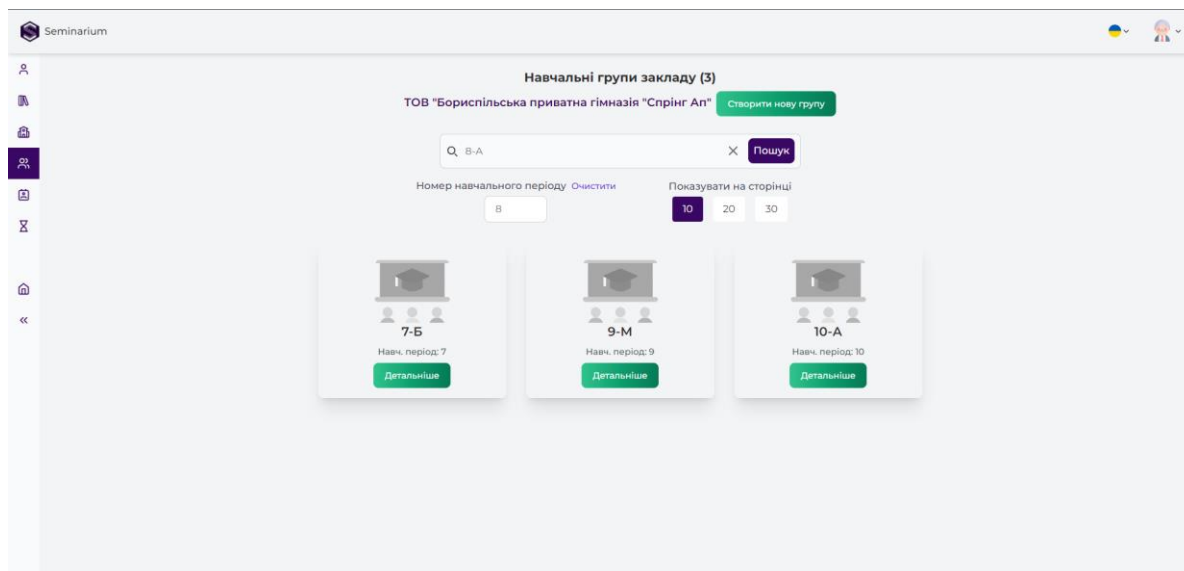


Рис. 3.11. – Перегляд навчальних груп закладу

На сторінці конкретної групи адміністратор навчального закладу може згенерувати запрошувальне посилання для студентів або куратора за яким користувачі зможуть створити нові профілі прив'язані до цієї групи. Отже, клікнувши на кнопку «Згенерувати запрошення для студентів», користувачу показується модальне вікно, де він може скопіювати згенероване посилання або зберегти QR-код (рис. 3.12), щоб передати його отримувачам освіти.

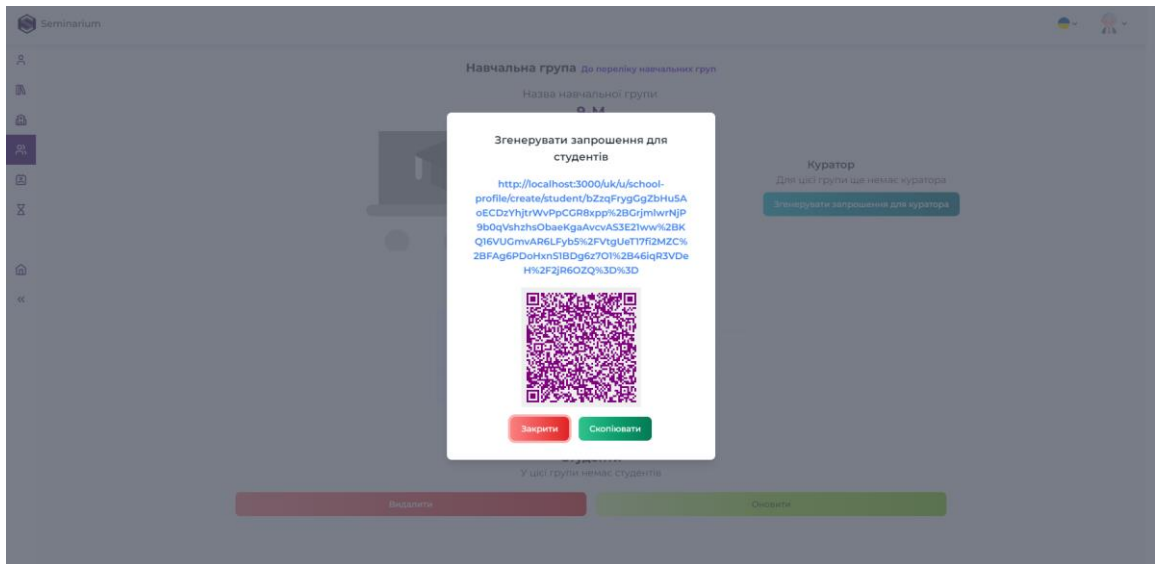


Рис. 3.12. – Генерування запрошення для студентів

Перейшовши за цим посиланням або просканувавши QR-код, студенти мають змогу створити профіль за допомогою спеціальної форми (рис. 3.13).

Створення профілю студента навчального закладу

Ім'я (обов'язково)

Телефон (необов'язково)

Електронна пошта (необов'язково)

Корисні деталі про Вас (необов'язково)

Дата народження (необов'язково)

Ваші особливі здібності (необов'язково)

☒ Ви є старостою
☐ Ви навчатесь індивідуально

Група здоров'я (обов'язково)

Створити профіль

Рис. 3.13. – Створення профілю студента

Після того, як студенти створять свої профілі, відповідна інформація про склад групи буде відображатись на сторінці цієї групи (рис. 3.14). Зокрема, буде відображено список студентів цієї групи та куратора.

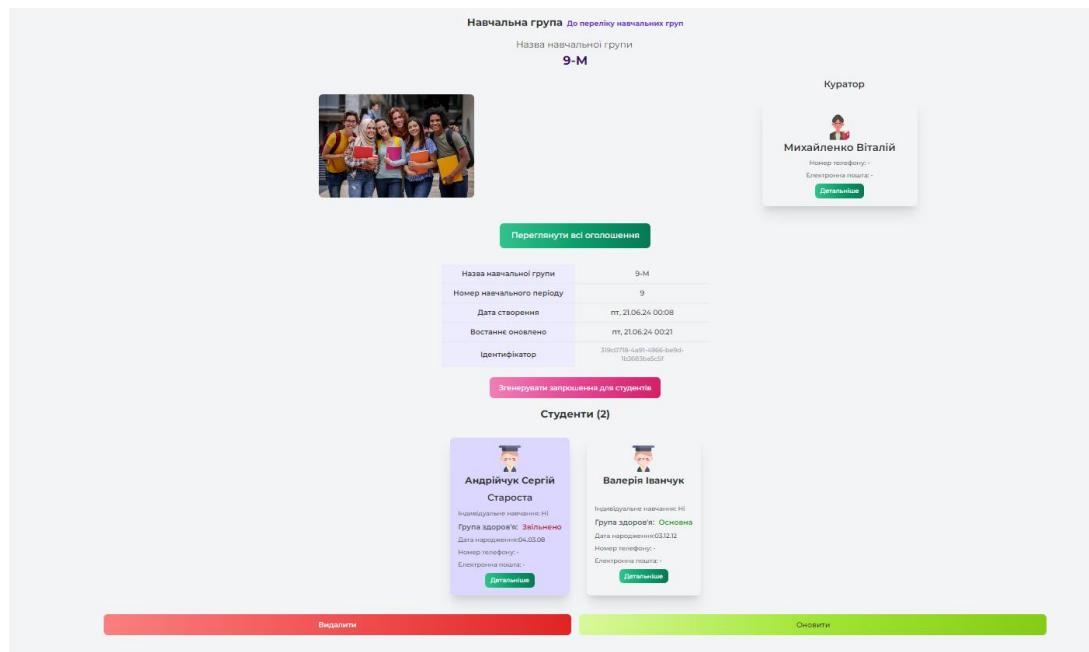


Рис. 3.14. – Перегляд сторінки навчальної групи

У адміністратора навчального закладу, в окремому розділі, є можливість створювати навчальні періоди через спеціальну форму (рис. 3.15). За певним навчальним періодом надалі будуть закріплюватись навчальні простори.

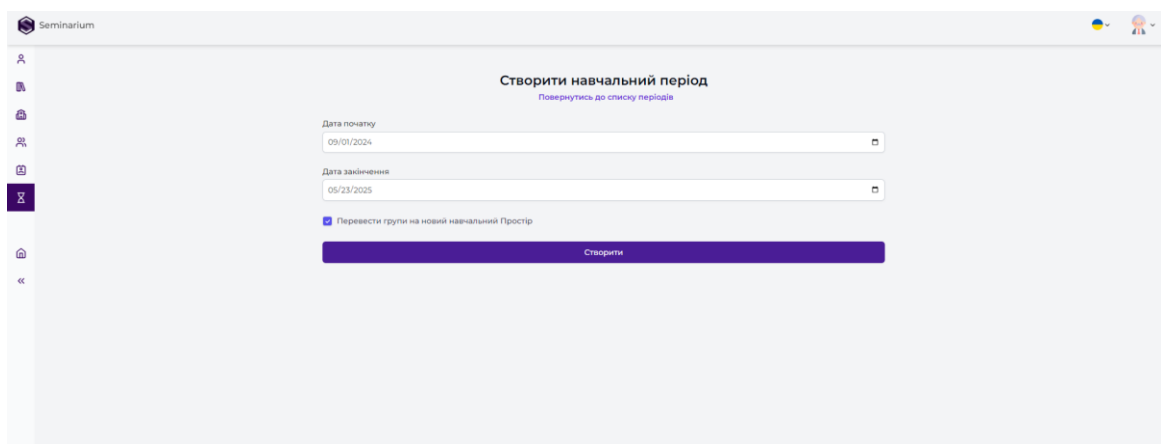


Рис. 3.15. – Створення навчального періоду

Також в учасників навчального закладу на окремій сторінці (рис. 3.16) є можливість переглядати всі профілі, що відносяться до їхнього закладу. Завдяки фільтрам та пошуку можна швидко та легко знайти той профіль, який потрібно.

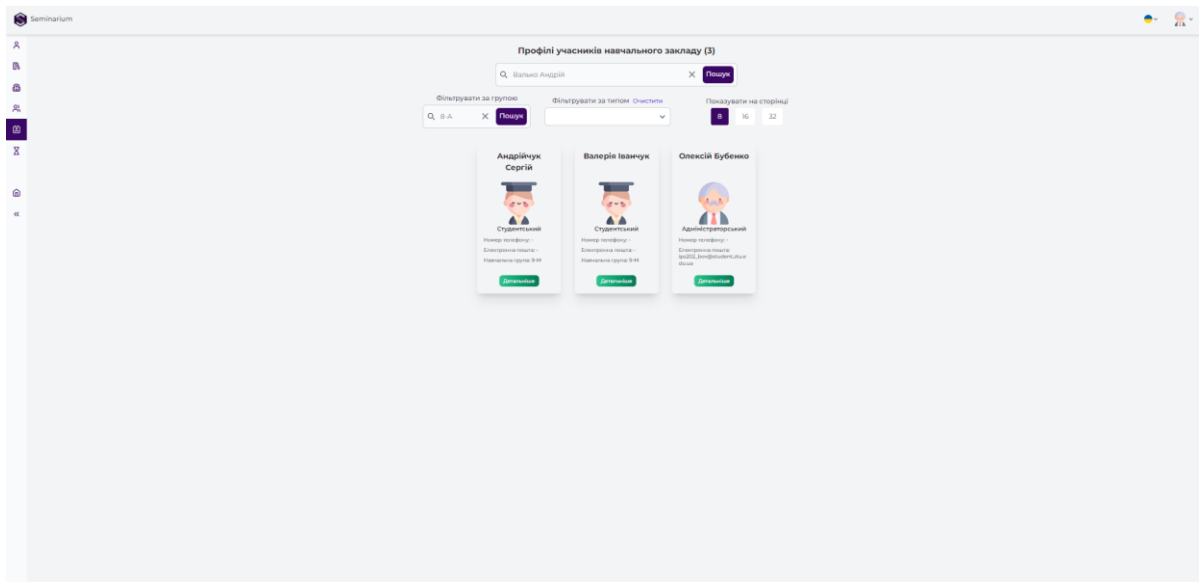


Рис. 3.16. – Перегляд навчальних профілів адміністратором навчального закладу

Для викладачів, при створенні їхнього профілю, відображається спеціальна форма (рис. 3.17), в якій він може вказати детальну інформацію про себе.

Створення профілю викладача навчального закладу

Ім'я (обов'язково)
Антонюк Сергій

Телефон (необов'язково)

Електронна пошта (необов'язково)

Корисні деталі про Вас (необов'язково)
Маю вади слуху...

Дисципліни, які викладаєте (необов'язково)
Інформатика, Креслення

Досвід (необов'язково)
автор понад 50 наукових та навчально-методичних праць

Освіта (необов'язково)
ЖДУ ім. І. Франка

Кваліфікація (необов'язково)
магістр з прикладної математики

Кількість занять на навчальний цикл (обов'язково)

Створити профіль

Рис. 3.17. – Створення профілю викладача

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

В адміністратора навчального закладу та викладачів є можливість створювати навчальні простори за допомогою форми (рис 3.18), до яких згодом прикріплюються навчальні групи.

Семінаріум

Створити навчальний простір

До переліку просторів

ТОВ "Бориспільська приватна гімназія "Спринг Ал"

Навчальний період *

Оберіть навчальний період

Назва простору *

Основи програмування ІП3-26

Опис

Викладач: Патріонко

Створити

Рис. 3.18. – Створення навчального простору

В учасників навчального закладу є можливість переглядати навчальні простори на окемій сторінці (рис. 3.19), здійснювати пошук та фільтрацію, а також можливість переглянути інформацію про конкретний простір.

Семінаріум

Навчальні простори

Створити

Інформатика

Пошук

Оберіть навчальний період

Показувати на сторінці

20 30

Фізична культура 12-А

Детальніше

Основи програмування ІП3-26

Детальніше

Рис. 3.19. – Перегляд навчальних просторів

Натиснувши на кнопку «Детальніше», користувач переходить на сторінку навчального простору, на якій відображено план занять, викладачів та групи закріплені за цим простором (рис. 3.20). У викладача, на цій сторінці є можливість керувати планом занять, а для кожного заняття можна додавати практичні завдання та теоретичні параграфи.

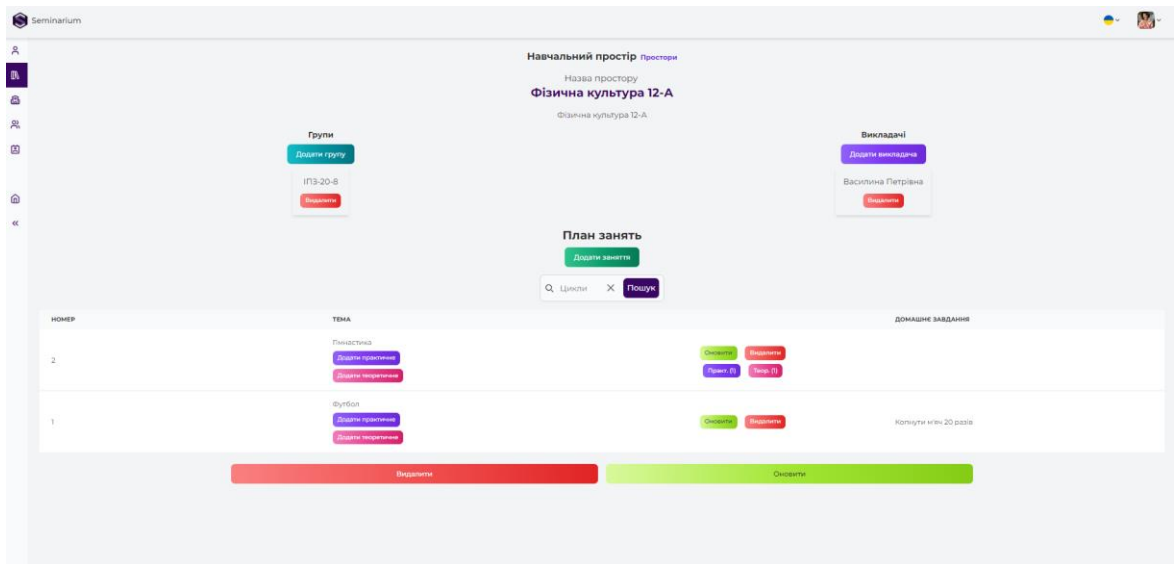


Рис. 3.20. – Перегляд навчального простору

Натиснувши на кнопку «Додати практичне», викладач переходить до форми створення практичного завдання (рис. 3.21). У нього є можливість додати заголовок та текст для практичного завдання, при цьому текст можна форматувати та він може містити як звичайні посилання, так і посилання на платформу YouTube.

Рис. 3.21. – Створення практичного завдання

Студенти, зі свого боку, клікнувши на кнопку «Практ.» на сторінці простору (рис. 3.22), можуть переглянути створене викладачем практичне завдання, а клікнувши на кнопку «Моя робота», в нього є можливість завантажити практичне завдання.

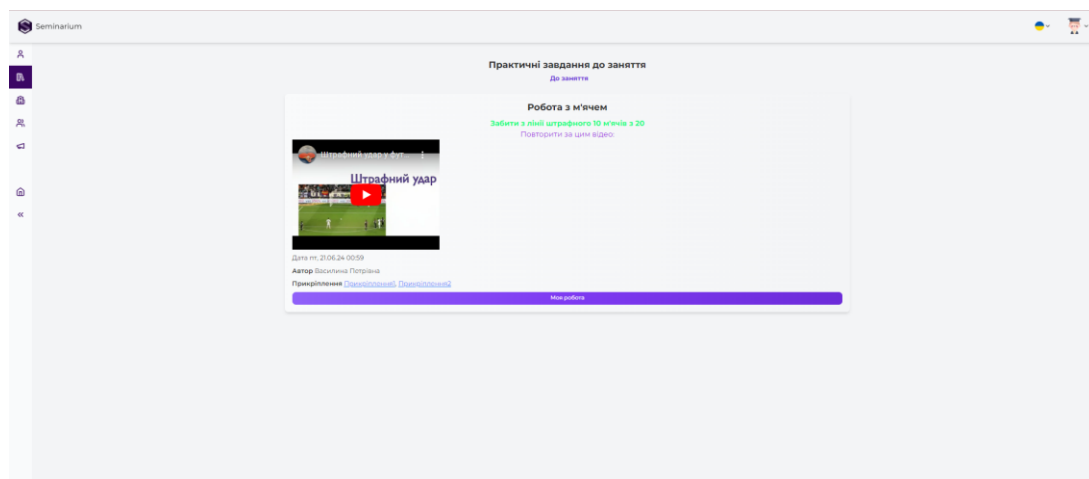


Рис. 3.22. – Перегляд практичного завдання

Після того, як студент завантажив роботу, у викладача є можливість її переглянути на окремій сторінці (рис. 3.23). Також викладач може переглянути завантажені прикріплені файли, якщо такі є.

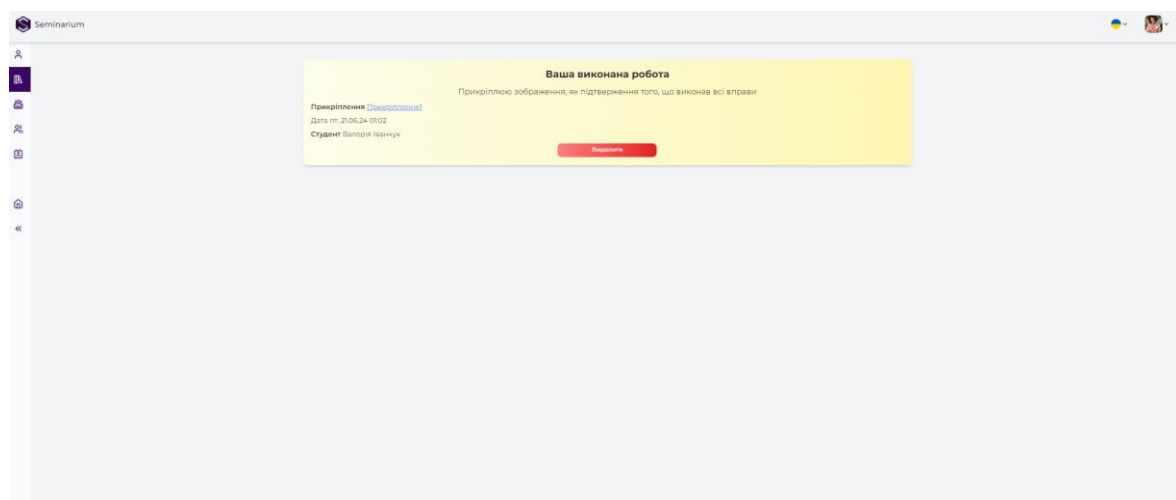


Рис. 3.23. – Перегляд виконаного практичного завдання

У опікуна є можливість переглядати пов'язаних з ним дітей на платформі на спеціальній сторінці (рис. 3.24), де він може клікнути на кнопку «Простори» щоб стежити за заняттями та завданнями, які призначені для цієї дитини.

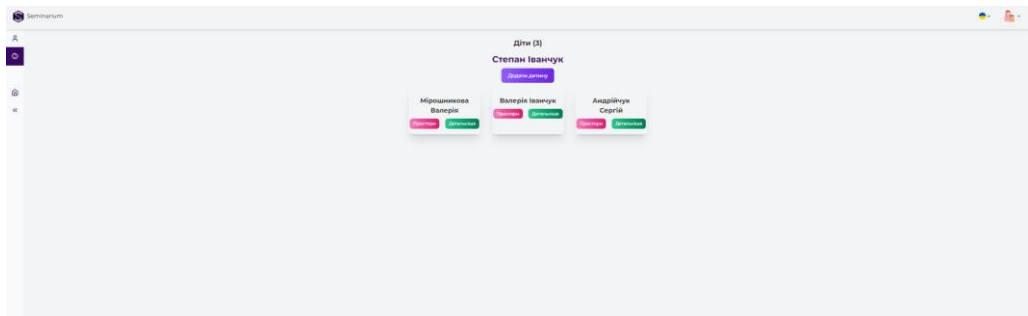


Рис. 3.24. – Перегляд дітей опікуном

Клікнувши на кнопку «Детальніше» на сторінці дітей (рис. 3.25), користувач може переглянути детальну інформацію про студента, зокрема про його батьків та навчальну групу, за якою закріплений студент.

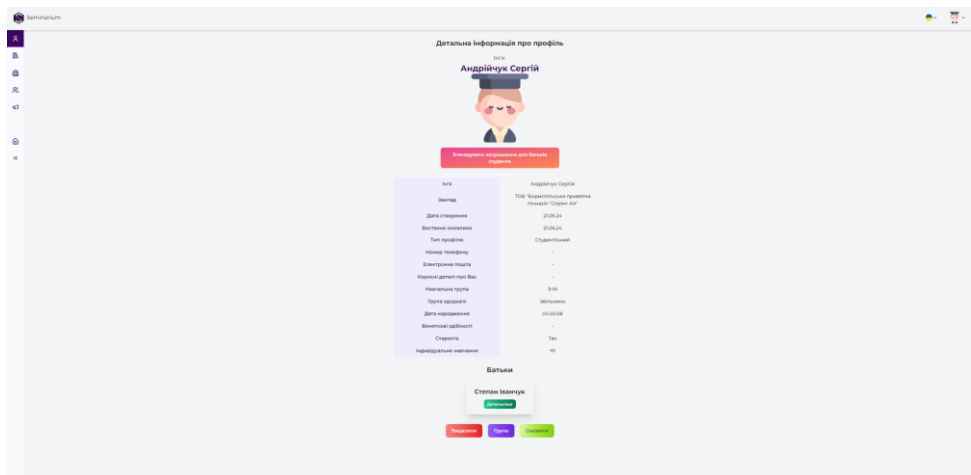


Рис. 3.25. – Перегляд профілю студента

Перейшовши до сторінки оголошень (рис. 3.26), користувачі мають можливість переглядати оголошення певної групи.

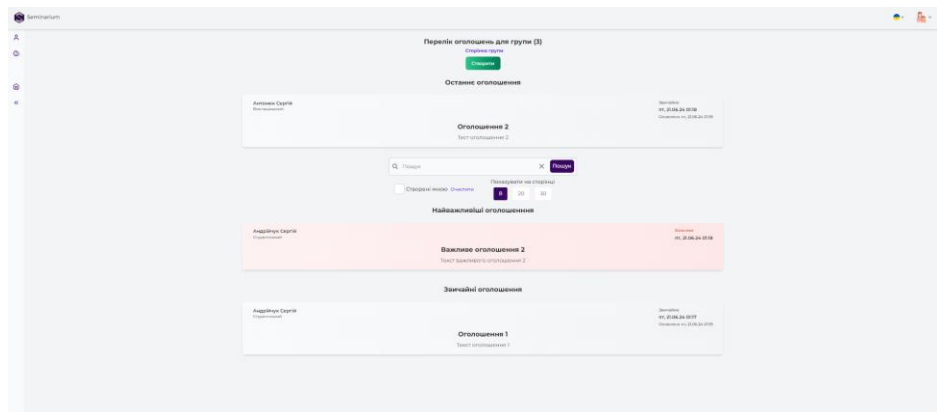


Рис. 3.26. – Перегляд оголошень в групі

Усі оголошення поділяються на два типи – важливі та звичайні. Угорі сторінки завжди відображено останнє оголошення, а нижче можна застосувати пошук та фільтрацію для всіх оголошень цієї групи, однак результати пошуку завжди починаються з найважливіших оголошень.

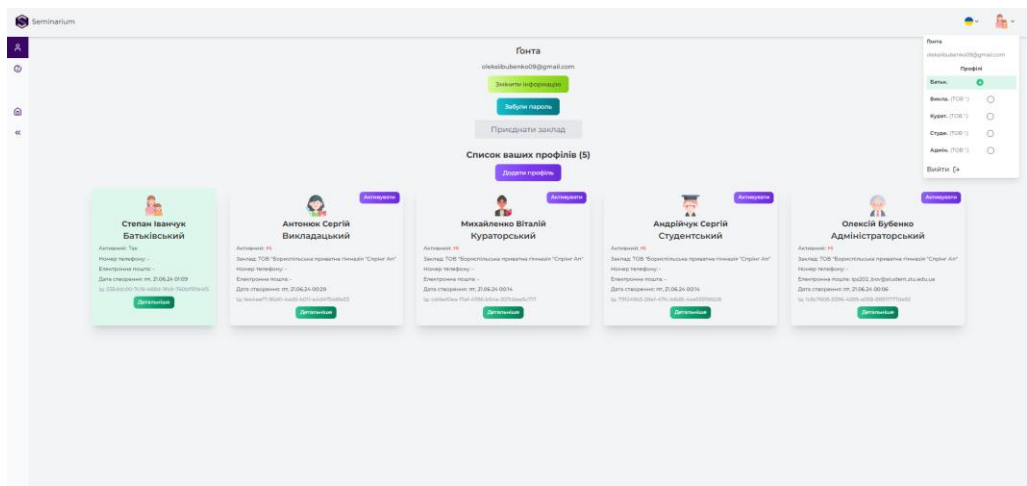


Рис. 3.27. – Зміна активного профілю

Слід зазначити, що користувач має можливість зробити активним той чи інший профіль двома способами: перейшовши на сторінку власних профілів та натиснувши кнопку «Активувати» або натиснувши на відповідний профіль у випадному меню поряд із зображенням профілю.

3.3 Тестування платформи

Тестування — це не лише важлива, але й невід'ємна частина розробки будь-якого великого програмного рішення. Після завершення розробки нашого програмного продукту було проведено комплексне мануальне всіх його компонентів. Головна мета цього процесу полягала у виявленні можливих проблем у функціоналі та впевненості, що програма відповідає всім вимогам, що були поставлені перед нею з самого початку розробки.

Для того, щоб протестувати API серверу було обрано інструмент ApiDog. ApiDog — це інструмент, який призначений, для того, щоб імітувати складні запити з клієнтської частини веб-сайтів [24]. Він надає широкий спектр можливостей для тестування API серверів, включаючи налаштування

параметрів запити, передачу тіла запити, роботу з заголовками та cookies, автоматичну генерацію тестових даних. Цей застосунок допомагає ефективно перевірити функціональність розроблених програмних рішень перед їх впровадженням у реальне середовище.

Після мануального тестування API сервера за допомогою інструмента ApiDog, було виявлено критичні помилки та виправлено їх. На рис. 3.28. зображено приклад надсилання запити до серверу через інструмент ApiDog.

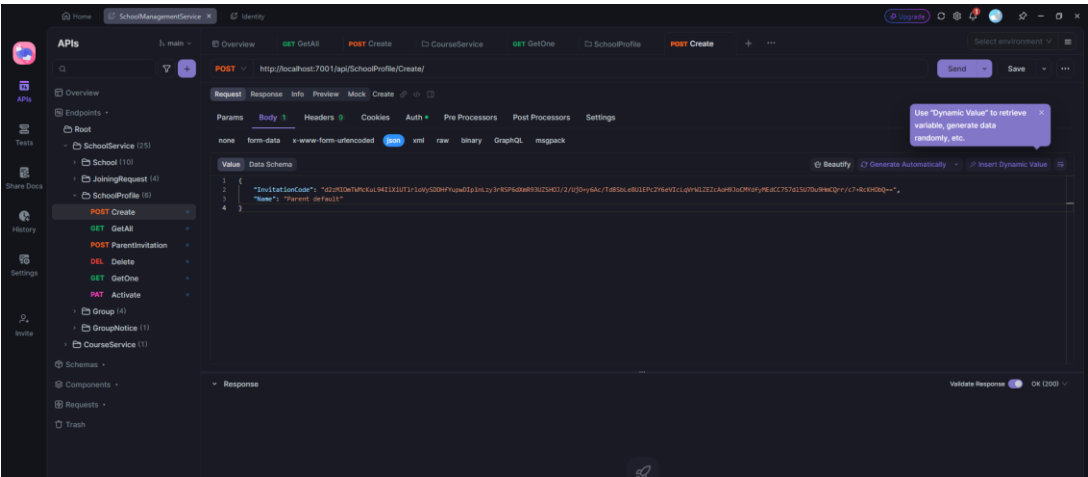


Рис. 3.28. – Приклад запити для тестування API за допомогою програми ApiDog

Під час цього процесу тестування було використано інструмент Lighthouse, розроблений Google, для якісного тестування веб-сторінок. Lighthouse забезпечує детальну звітність щодо різних аспектів веб-проекту, допомагаючи виявити потенційні проблеми та недоліки для поліпшення користувацького досвіду та оптимізації роботи сайту [25]. На рис. 3.29 подано результати тестування за показниками Lighthouse.

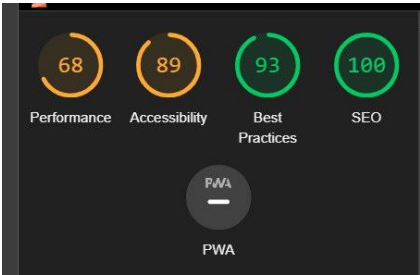


Рис. 3.29. – Результати тестування за показниками Lighthouse

Для перевірки адаптивності інтерфейсу веб-сторінки було використано вбудовані інструменти у браузері. Це дозволило оцінити чи коректно відображаються всі веб-сторінки на різних розмірах екранів.

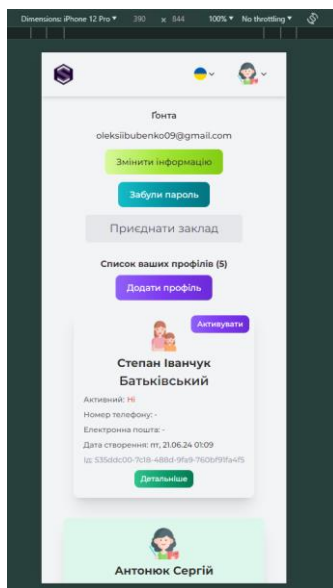


Рис. 3.30. Сторінка профілів користувача зі смартфона

На рис. 3.30. зображено сторінку профілів користувача на смартфоні iPhone 12 Pro.

Висновки до третього розділу

У цьому розділі було розглянуто процес та особливості розгортання мікросервісної програмної системи. Зокрема, було змодельовано діаграму розгортання на якій зображено взаємодію компонентів системи. Також було описано конфігурацію та порядок розгортання системи.

У вигляді зображень проілюстровано інтерфейс клієнтської частини оновного функціоналу застосунку.

У результаті було мануально протестовано реалізовану систему управління навчальним процесом, використовуючи іструмент ApiDog та виправлено виявлені критичні помилки. Також, веб-сайт було протестовано за метриками Lighthouse для оцінки загальної продуктивності та доступності, а за допомогою вбудованих інструментів браузера перевірено адаптивність веб-сайту для різних розширень екрану.

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи бакалавра було проаналізовано функціонал існуючих рішень для того, щоб визначити найголовніші вимоги до системи, що було розроблено. На основі визначених вимог було визначено та обґрунтовано вибір мікросервісної та чистої архітектур, а також інструментальних засобів, таких як мови програмування та фреймворки.

Під час проєктування системи управління навчальним процесом для закладів освіти було виділено таких акторів: Студент, Викладач, Учень, Опікун, Куратор, Адміністратор навчального закладу та Адміністратор платформи. Для кожного з акторів було виокремлено варіанти використання на відповідній діаграмі. Для опису класів доменного рівня основних мікросервісів було змодельовано діаграму класів. Також було розроблено схему двох баз даних, їх таблиць та зв'язків між цими таблицями. При проєктуванні та реалізації алгоритмів системи було змодельовано алгоритми створення та обробки запиту навчального закладу на приєднання та завантаження студентом виконаного практичного завдання.

Встановлено мінімальні вимоги до сервера для розгортання системи, а також розроблена послідовність кроків для процесу розгортання. Проведено демонстрацію інтерфейсу клієнтської частини основного функціоналу системи та проведено тестування застосунку і усунення критичних помилок. В результаті

У результаті виконання роботи розроблена система управління навчальним процесом готова до впровадження та має великий потенціал для розширення, оскільки була розроблена з урахуванням майбутнього розвитку та розширення функціоналу.

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бубенко О.В., Вакалюк Т.А. Проєктування архітектури системи управління спільнотами та подіями. Тези доповідей XIV Міжнародної науково-технічної конференції Інформаційно-комп'ютерні технології - 2023, 28-29 березня 2024 року. Житомир: «Житомирська політехніка», 2024. С.5-6.
2. HUMAN Школа | HUMAN [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.human.ua/schools>.
3. Нові знання – Електронні щоденники та журнали з можливостями дистанційного навчання [Електронний ресурс]. – Режим доступу до ресурсу: <https://nz.ua/>.
4. Smart School Система автоматизації освітнього процесу [Електронний ресурс]. – Режим доступу до ресурсу: <https://smart-school.com.ua/>.
5. Навіщо менеджеру розбиратися в архітектурі IT-проєкту | dev.ua [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://dev.ua/news/navishcho-menedzheru>.
6. Що краще моноліт чи мікросервіси? | IAMPM [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://iampm.club/ua/blog/shho-krashhe-monolit-chi-mikroservisi-yak-obrati-arhitekturu-projektu/>.
7. Microservice Architecture pattern [Електронний ресурс]. – Режим доступу до ресурсу: <https://microservices.io/patterns/microservices.html>.
8. Навіщо потрібна сервіс-орієнтована архітектура | robot_dreams [Електронний ресурс]. – Режим доступу до ресурсу: <https://robotdreams.cc/uk/blog/268-zachem-nuzhna-servis-orientirovannaya-arhitektura>.
9. Про мікросервісну архітектуру [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://foxminded.ua/mikroservisna-arkhitektura/>.

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

10. What is the Role of API gateway in Microservices? [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/what-is-the-role-of-api-gateway-in-microservices/>.
11. 10 Common Microservices Anti-Patterns [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://www.designgurus.io/blog/10-common-microservices-anti-patterns>.
12. What is a 3-tier application architecture? Definition and Examples [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://vfunction.com/blog/3-tier-application/>.
13. From Chaos to Clarity: Unveiling the Secrets of Clean Architecture [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://shashankreddynallu.in/clean-architecture>.
14. Що таке PostgreSQL і для чого використовується? [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://foxminded.ua/postgresql-shcho-tse/>.
15. Introduction to RabbitMQ [Електронний ресурс]. – Режим доступу до ресурсу: <https://jstobigdata.com/rabbitmq/introduction-to-rabbitmq/>.
16. Enhancing Code Quality in .NET Core with SonarAnalyzer.CSharp and StyleCop.Analyzers [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://medium.com/@farkhondepeyali/enhancing-code-quality-in-net-core-with-sonaranalyzer-csharp-and-stylecop-analyzers-17e8f049d7a6>.
17. Чому важлива типізація в JS [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://foxminded.ua/typizaciya-js/>.
18. Next.js SEO: Прийміть майбутнє веб-оптимізації [Електронний ресурс]. – Режим доступу до ресурсу: <https://seo.london/uk/next-js-seo/>.
19. Class Diagram [Електронний ресурс]. – Режим доступу до ресурсу: https://sparxsystems.com/enterprise_architect_user_guide/16.1/modeling_language_s/classdiagram.html.

20. CQRS pattern [Електронний ресурс]. – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs>.
21. Simplifying complexity with MediatR and Minimal APIs [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://q.agency/blog/simplifying-complexity-with-mediatr-and-minimal-apis/>.
22. What is a reverse proxy? | Proxy servers explained [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/>.
23. Мікросервісна архітектура ПЗ [Електронний ресурс]. – Режим доступу до ресурсу: <https://qalight.ua/baza-znaniy/shho-take-mikroservisna-arhitektura-pz/>.
24. REAL API Design-first Development Platform [Електронний ресурс]. – Режим доступу до ресурсу: <https://apidog.com/>.
25. Швидкість завантаження сайту. Все про головні метрики, способи відстеження даних і червневий апдейт від Google [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: <https://golden-web.digital/blog/rubrika-seo/shvidkist-zavantazhennia-saitu-vse-pro-golovni-metriki-sposobi-vidstezhennia-danikh-i-cherhvnevii-apdeit-vid-google/>.

ДОДАТКИ

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		91

Лістинг вихідного коду серверної частини застосунку

```

public class ProfileIdentifyAttribute(string[] allowedProfileTypes, bool allowed-
ForAdmin = false) : ActionFilterAttribute
{
    private const string HeaderKeyName = "SchoolProfileId";

    public override void OnActionExecuting(ActionExecutingContext context)
    {
        var userId = context.HttpContext.User.Identity?.GetId();
        var userRole = context.HttpContext.User.Identity?.GetRole();

        if (userId is null || userRole is null)
        {
            var problemDetail = CreateProblemDetail(
                "Unable to identify the user and their role.",
                "unauthorized",
                StatusCodes.Status401Unauthorized,
                "user"
            );
            SetResult(context, problemDetail);

            return;
        }

        if (userRole == Constants.AdminRole)
        {
            if (!allowedForAdmin)
            {
                var problemDetail = CreateProblemDetail(
                    "This resource is not available to the administrator.",
                    "forbidden",
                    StatusCodes.Status403Forbidden,
                    "user_role"
                );
                SetResult(context, problemDetail);
            }

            return;
        }

        var schoolProfileIdRetrieved =
            context.HttpContext.Request.Headers.TryGetValue(HeaderKeyName, out var
schoolProfileId);
        var isSchoolProfileIdValid = Guid.TryParse(schoolProfileId, out var school-
ProfileGuid);

        if (!schoolProfileIdRetrieved || !isSchoolProfileIdValid)
        {
            var problemDetail = CreateProblemDetail(
                "Unable to identify the user school profile.",
                "forbidden",
                StatusCodes.Status403Forbidden,
                "school_profile"
            );
            SetResult(context, problemDetail);

            return;
        }
    }
}

```

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		92

```

        var schoolProfileManager = context.HttpContext.RequestServices.GetRequiredService<ISchoolProfileManager>();

        var profiles = schoolProfileManager.GetProfiles((Guid)userId).Result;
        var currentProfile = profiles?.FirstOrDefault(p => p.Id == schoolProfileGuid);

        if (currentProfile is null)
        {
            var problemDetail = CreateProblemDetail(
                "User school profile not found.",
                "forbidden",
                StatusCodes.Status403Forbidden,
                "school_profile"
            );
            SetResult(context, problemDetail);

            return;
        }

        if (!currentProfile.IsActive ||
            !SchoolProfileTypeExists(allowedProfileTypes, currentProfile.Type.ToString().ToSnakeCase()))
        {
            var problemDetail = CreateProblemDetail(
                "Incorrect user profile.",
                "invalid_school_profile",
                StatusCodes.Status403Forbidden,
                "school_profile"
            );
            SetResult(context, problemDetail);
        }
    }

    private static ProblemDetails CreateProblemDetail(string detail, string title,
        int statusCode, string extensionKey)
    {
        return new ProblemDetails
        {
            Detail = detail,
            Title = title,
            Status = statusCode,
            Type = "access_denied",
            Extensions =
            {
                { "params", new List<string> { extensionKey } }
            }
        };
    }

    private static void SetResult(ActionExecutingContext context, ProblemDetails problemDetail)
    {
        context.Result = new ObjectResult(problemDetail)
        {
            StatusCode = problemDetail.Status
        };
    }

    public static bool SchoolProfileTypeExists(string[] allowedProfiles, string profile) =>
        allowedProfiles.Exists(s => s.Contains(profile));
}

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		93

```

public class S3Service(IAmazonS3 s3Client) : IS3Service
{
    public Either<FileSuccess, Error> GetOne(GetFileRequest request)
    {
        try
        {
            var preSignedUrl = GetPreSignedUrl(request.Bucket, request.Name, request.UrlExpirationInMin);
            return new FileSuccess(preSignedUrl, request.Name);
        }
        catch
        {
            return new InvalidFileOperationError(request.Name, "retrieving");
        }
    }

    public async Task<Either<FileSuccess, Error>> UploadOne(SaveFileRequest request)
    {
        try
        {
            var putRequest = new PutObjectRequest
            {
                BucketName = request.Bucket,
                Key = request.Name,
                InputStream = request.Stream,
            };

            await s3Client.PutObjectAsync(putRequest);
        }
        catch
        {
            return new InvalidFileOperationError(request.Name, "uploading");
        }

        try
        {
            var preSignedUrl = GetPreSignedUrl(request.Bucket, request.Name, request.UrlExpirationInMin);
            return new FileSuccess(preSignedUrl, request.Name);
        }
        catch
        {
            return new InvalidFileOperationError(request.Name, "retrieving");
        }
    }

    public async Task<Option<Error>> DeleteOne(DeleteFileRequest request)
    {
        try
        {
            var deleteRequest = new DeleteObjectRequest
            {
                BucketName = request.Bucket,
                Key = request.Name
            };

            await s3Client.DeleteObjectAsync(deleteRequest);
        }
        catch
        {
            return new InvalidFileOperationError(request.Name, "deleting");
        }

        return Option<Error>.None;
    }
}

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		94

```

        private string GetPreSignedUrl(string bucket, string name, int urlExpiration-
InMin = 100)
        {
            var preSignedUrlRequest = new GetPreSignedUrlRequest
            {
                BucketName = bucket,
                Key = name,
                Expires = DateTime.UtcNow.AddMinutes(urlExpirationInMin)
            };

            var preSignedUrl = s3Client.GetPreSignedURL(preSignedUrlRequest);
            return preSignedUrl;
        }
    }

```

```
using Exception = System.Exception;
```

```
namespace SchoolService.Application.SchoolProfile.SchoolProfileManager;
```

```
public class SchoolProfileManager : ISchoolProfileManager
{
```

```
    private readonly ICommandContext _commandContext;
```

```
    private readonly IQueryContext _queryContext;
```

```
    private readonly IMemoryCache _memoryCache;
```

```
    private readonly IMapper _mapper;
```

```
    private readonly IFilesManager _filesManager;
```

```
    public SchoolProfileManager(
        ICommandContext commandContext,
        IQueryContext queryContext,
        IMemoryCache memoryCache,
        IMapper mapper,
        IFilesManager filesManager)
    {
        _commandContext = commandContext;
        _queryContext = queryContext;
        _memoryCache = memoryCache;
        _mapper = mapper;
        _filesManager = filesManager;
    }

```

```
    public async Task<Either<Domain.Entities.SchoolProfile, Error>>
CreateSchoolAdminProfile(
    Invitation invitation, CreateSchoolProfileCommand command)
    {
        if (DateTime.UtcNow > invitation.Expired)
            return new InvalidError("invitation");
    }

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		95

```

        var school = await
_commandContext.Schools.FindAsync(invitation.SourceId);
        if (school == null)
            return new InvalidError("school_id");

        var existedProfile = await _commandContext.SchoolProfiles
            .AsNoTracking()
            .Where(p => p.SchoolId == school.Id && p.UserId ==
command.UserId &&
                p.Type == SchoolProfileType.SchoolAdmin)
            .FirstOrDefaultAsync();

        if (existedProfile != null)
            return new AlreadyExistsError("school_profile")
            {
                Params = new List<string>(2) { "user_id", "school_id" }
            };

        var profile = _mapper.Map<Domain.Entities.SchoolProfile>(command);
        profile.School = school;
        profile.Type = invitation.Type;
        profile.IsActive = true;

        return profile;
    }

    public async Task<Either<Domain.Entities.SchoolProfile, Error>>
CreateClassTeacherProfile(Invitation invitation,
CreateSchoolProfileCommand command)
    {
        if (DateTime.UtcNow > invitation.Expired)
            return new InvalidError("invitation");

        var group = await _queryContext.Groups
            .Include(g => g.ClassTeacher)
            .FirstOrDefaultAsync(g => g.Id == invitation.SourceId);
        if (group == null)
            return new InvalidError("group_id");

        if (group.ClassTeacher is not null)
            return new AlreadyExistsError("class_teacher");

        var existedProfile = await _commandContext.SchoolProfiles
            .Where(p => p.GroupId == group.Id && p.UserId ==
command.UserId &&
                p.Type == SchoolProfileType.ClassTeacher)
            .FirstOrDefaultAsync();

        if (existedProfile != null)
            return new AlreadyExistsError("school_profile")

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		96


```

        {
            Params = new List<string>(2) { "user_id", "group_id" }
        };

        var profile = _mapper.Map<Domain.Entities.SchoolProfile>(command);
        profile.ClassTeacherGroupId = group.Id;
        profile.Type = invitation.Type;
        profile.IsActive = true;

        return profile;
    }

    public async Task<Either<Domain.Entities.SchoolProfile, Error>>
    CreateTeacherProfile(
        Invitation invitation, CreateSchoolProfileCommand command)
    {
        if (DateTime.UtcNow > invitation.Expired)
            return new InvalidError("invitation");

        if (command.TeacherLessonsPerCycle is null)
            return new InvalidError("lessons_per_cycle");

        var school = await
        _commandContext.Schools.FindAsync(invitation.SourceId);
        if (school == null)
            return new InvalidError("school_id");

        var existedProfile = await _commandContext.SchoolProfiles
            .Where(p => p.SchoolId == school.Id && p.UserId ==
command.UserId &&
                p.Type == SchoolProfileType.Teacher)
            .FirstOrDefaultAsync();

        if (existedProfile != null)
            return new AlreadyExistsError("school_profile")
            {
                Params = new List<string>(2) { "user_id", "school_id" }
            };

        var profile = _mapper.Map<Domain.Entities.SchoolProfile>(command);

        var data = new TeacherSerializationData(
            command.TeacherSubjects,
            command.TeacherExperience,
            command.TeacherEducation,
            command.TeacherQualification,
            (uint)command.TeacherLessonsPerCycle
        );

        profile.School = school;
    }

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		97

```

        profile.Type = invitation.Type;
        profile.Data = JsonConvert.SerializeObject(data);
        profile.IsActive = true;

        return profile;
    }

    public async Task<Either<Domain.Entities.SchoolProfile, Error>>
    CreateStudentProfile(Invitation invitation, CreateSchoolProfileCommand
    command)
    {
        if (DateTime.UtcNow > invitation.Expired)
            return new InvalidError("invitation");

        if (command.StudentIsIndividually is null)
            return new InvalidError("is_individually");

        if (command.StudentIsClassLeader is null)
            return new InvalidError("is_class_leader");

        if (command.StudentHealthGroup is null)
            return new InvalidError("health_group");

        var group = await
        _queryContext.Groups.FindAsync(invitation.SourceId);
        if (group == null)
            return new InvalidError("group_id");

        var existedProfile = await _commandContext.SchoolProfiles
            .Where(p => p.GroupId == group.Id && p.UserId ==
            command.UserId &&
                p.Type == SchoolProfileType.Student)
            .FirstOrDefaultAsync();

        if (existedProfile != null)
            return new AlreadyExistsError("school_profile")
            {
                Params = new List<string>(2) { "user_id", "group_id" }
            };

        var data = new StudentSerializationData(
            command.StudentDateOfBirth,
            command.StudentAptitudes,
            (bool)command.StudentIsClassLeader,
            (bool)command.StudentIsIndividually,
            (HealthGroup)command.StudentHealthGroup
        );

        var serializationData = JsonConvert.SerializeObject(data);
    }

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		98

```

        var profile = _mapper.Map<Domain.Entities.SchoolProfile>(command);
        profile.GroupId = group.Id;
        profile.Type = invitation.Type;
        profile.IsActive = true;
        profile.Data = serializationData;

        return profile;
    }

    public async Task<(Either<Domain.Entities.SchoolProfile, Error>
    Result, bool IsNew)> CreateParentProfileOrAddChild
        (Invitation invitation, CreateSchoolProfileCommand command)
    {
        if (DateTime.UtcNow > invitation.Expired)
            return (new InvalidError("invitation"), false);

        var child = await _commandContext
            .SchoolProfiles
            .FindAsync(invitation.SourceId);
        if (child is null)
            return (new InvalidError("child_id"), false);

        var existedParent = await _commandContext.SchoolProfiles
            .Include(p => p.Children)
            .FirstOrDefaultAsync(p => p.UserId == command.UserId && p.Type
            == SchoolProfileType.Parent);

        if (existedParent is { Children: { } } &&
            existedParent.Children.Contains(child))
            return (new AlreadyExistsError("child")
            {
                Params = new List<string>(2) { "child_id" }
            }, false);

        if (existedParent == null)
        {
            var serializationData = JsonConvert.SerializeObject(
                new ParentSerializationData(command.ParentAddress)
            );

            var profile =
                _mapper.Map<Domain.Entities.SchoolProfile>(command);

            profile.Type = invitation.Type;
            profile.IsActive = true;
            profile.Data = serializationData;

            await _commandContext.SchoolProfiles.AddAsync(profile);
            await
                _commandContext.SaveChangesAsync(CancellationToken.None);
        }
    }

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		99

```

        profile.Children ??= new
List<Domain.Entities.SchoolProfile>();
        profile.Children.Add(child);

        _commandContext.SchoolProfiles.Update(child);
        _commandContext.SchoolProfiles.Update(profile);

        ClearCache(child.UserId);

        try
        {
            await
_commandContext.SaveChangesAsync(CancellationToken.None);
        }
        catch (Exception exception)
        {
            Log.Error(exception, "An error occurred while saving
school parent profile with values {@Profile}.", profile);
        }

        return (profile, true);
    }

    existedParent.Children ??= new
List<Domain.Entities.SchoolProfile>();
    existedParent.Children.Add(child);
    foreach (var childToClearCache in existedParent.Children)
        ClearCache(childToClearCache.UserId);

    existedParent.IsActive = true;

    try
    {
        _commandContext.SchoolProfiles.Update(existedParent);

        await
_commandContext.SaveChangesAsync(CancellationToken.None);
        return (existedParent, false);
    }
    catch (Exception exception)
    {
        Log.Error(exception, "Excpetion caught while adding child to
parent with id @Id", existedParent.UserId);
        return (new InvalidDatabaseOperationError("school_profile"),
false);
    }
}

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		100

```

        public async Task<SchoolProfileModelResponse?> CacheProfiles(Guid
userId, Guid currentProfileId)
        {
            ClearCache(userId);
            var cacheKey = GetCacheKey(userId);

            var userProfiles = await _commandContext.SchoolProfiles
                .Where(profile => profile.UserId == userId)
                .ToListAsync();

            if (!userProfiles.Any())
            {
                _memoryCache.Set<IEnumerable<SchoolProfileModelResponse?>>(cacheKey,
                null);

                return null;
            }

            ActivateProfile(userProfiles, currentProfileId);

            var profilesToCache = await MapToResponses(userProfiles);
            _memoryCache.Set(cacheKey, profilesToCache);

            return profilesToCache?.FirstOrDefault(p => p.IsActive);
        }

        public async Task<IEnumerable<SchoolProfileModelResponse?>>
        GetProfiles(Guid userId)
        {
            var cacheKey = GetCacheKey(userId);

            if (_memoryCache.TryGetValue(cacheKey, out
            List<SchoolProfileModelResponse?> cachedProfiles))
                return cachedProfiles;

            var profiles = await _queryContext.SchoolProfiles
                .Where(profile => profile.UserId == userId)
                .ToListAsync();

            foreach (var profile in profiles)
            {
                switch (profile.Type)
                {
                    case SchoolProfileType.SchoolAdmin or
                    SchoolProfileType.Teacher:
                        {
                            var school = await
                            _queryContext.Schools.FindAsync(profile.SchoolId);
                            profile.School = school;
                            break;
                        }
                }
            }
        }
    }

```

```

    }
    case SchoolProfileType.ClassTeacher:
    {
        var group = await _queryContext.Groups
            .Include(g => g.School)
            .FirstOrDefaultAsync(g => g.Id ==
profile.ClassTeacherGroupId);
        profile.Group = group;
        profile.School = group?.School;
        break;
    }
    case SchoolProfileType.Student:
    {
        var group = await _queryContext.Groups
            .Include(g => g.School)
            .FirstOrDefaultAsync(g => g.Id ==
profile.GroupId);

        profile.Group = group;
        profile.School = group?.School;

        try
        {
            var allParents = await
_queryContext.SchoolProfiles
                                .Where(p => p.Type ==
SchoolProfileType.Parent)

                                .ToListAsync(CancellationToken.None);

            var allChildren = await
_queryContext.SchoolProfiles
                                .Where(p => p.Type ==
SchoolProfileType.Student)

                                .ToListAsync(CancellationToken.None);

            var filteredParents = allParents
                                .Where(p => allChildren.Exists(child =>
child.Id == profile.Id &&
allParents != null &&
allParents.Exists(parent => parent.Id == p.Id)))
                                .ToList();

            profile.Parents = filteredParents;
        }
        catch (Exception ex)
        {
            profile.Parents = null;

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		102

```

        Log.Error(ex, "An error occurred while
retrieving parents of clhild with values {@Profile}.", profile);
    }

    break;
}
case SchoolProfileType.Parent:
{
    try
    {
        var allParents = await
_queryContext.SchoolProfiles
                .Where(p => p.Type ==
SchoolProfileType.Parent)
                .ToListAsync(Cancellation.Token.None);

        var allChildren = await
_queryContext.SchoolProfiles
                .Where(p => p.Type ==
SchoolProfileType.Student)
                .ToListAsync(Cancellation.Token.None);

        var filteredChildren = allChildren
                .Where(p => allParents.Exists(parent =>
parent.Id == profile.Id &&
allChildren != null &&
allChildren.Exists(child => child.Id == p.Id)))
                .ToList();

        profile.Children = filteredChildren;
    }
    catch (Exception ex)
    {
        profile.Children = null;
        Log.Error(ex, "An error occurred while
retrieving children of parent with values {@Profile}.", profile);
    }

    break;
}
}

var profileResponses = await MapToResponses(profiles);
_memoryCache.Set(cacheKey, profileResponses);
return profileResponses;
}

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		103

```

        public async Task<SchoolProfileModelResponse?> GetActiveProfile(Guid
userId)
        {
            var profiles = await GetProfiles(userId);

            var activeProfile = profiles?.FirstOrDefault(p => p.IsActive);
            return activeProfile;
        }

        public async Task<Option<Error>> ValidateSchoolProfileBySchool(Guid?
userId, Guid schoolId)
        {
            if (userId is null)
                return Option<Error>.None;

            var schoolProfile = await GetActiveProfile((Guid)userId);

            if (schoolProfile?.SchoolId is null)
                return new InvalidError("school_profile");

            var profileSchoolId = (Guid)schoolProfile.SchoolId;
            return profileSchoolId != schoolId
                ? new InvalidError("school")
                : Option<Error>.None;
        }

        public async Task<Option<Error>> ValidateSchoolProfileByGroup(Guid?
userId, Guid groupId)
        {
            if (userId is null)
                return Option<Error>.None;

            var schoolProfile = await GetActiveProfile((Guid)userId);

            if (schoolProfile?.GroupId is null)
                return new InvalidError("school_profile");

            var profileGroupId = (Guid)schoolProfile.GroupId;
            return profileGroupId != groupId
                ? new InvalidError("group")
                : Option<Error>.None;
        }

        public async Task<Option<Error>>
ValidateParentProfileByChildGroup(Guid? userId, Guid groupId)
        {
            if (userId is null)
                return Option<Error>.None;

            var profile = await GetActiveProfile((Guid)userId);

```

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		104


```

        if (profile is null)
            return new InvalidError("school_profile");

        try
        {
            var allParents = await _queryContext.SchoolProfiles
                .Where(p => p.Type == SchoolProfileType.Parent)
                .ToListAsync(Cancellation.Token.None);

            var allChildren = await _queryContext.SchoolProfiles
                .Where(p => p.Type == SchoolProfileType.Student)
                .ToListAsync(Cancellation.Token.None);

            var filteredChildren = allChildren
                .Where(p => allParents.Exists(parent => parent.Id ==
profile.Id &&
allChildren != null
&&
allChildren.Exists(child => child.Id == p.Id)));

            var filteredByGroupChildren = filteredChildren
                .Where(p => p.GroupId == groupId)
                .ToListAsync();

            if (filteredByGroupChildren.Count == 0)
                return new InvalidError("school_profile");
        }
        catch (Exception ex)
        {
            Log.Error(ex, "An error occurred while retrieving children of
parent with values {@Profile}.", profile);
            return new InvalidError("school_profile");
        }

        return Option<Error>.None;
    }

    public async Task<Option<Error>>
ValidateClassTeacherSchoolProfileByGroup(Guid? userId, Guid groupId)
    {
        if (userId is null)
            return Option<Error>.None;

        var schoolProfile = await GetActiveProfile((Guid)userId);

        if (schoolProfile?.ClassTeacherGroupId is null)
            return new InvalidError("school_profile");
    }

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		105

```

        var profileClassTeacherGroupId =
        (Guid)schoolProfile.ClassTeacherGroupId;
        return profileClassTeacherGroupId != groupId
            ? new InvalidError("school_id")
            : Option<Error>.None;
    }

    public void ClearCache(Guid userId)
    {
        var cacheKey = GetCacheKey(userId);
        _memoryCache.Remove(cacheKey);
    }

    private static void
    ActivateProfile(List<Domain.Entities.SchoolProfile> profiles, Guid
    currentProfileId)
        => profiles.ForEach(profile => profile.IsActive = profile.Id ==
    currentProfileId);

    private async Task<IReadOnlyList<SchoolProfileModelResponse>?>
    MapToResponses(IEnumerable<Domain.Entities.SchoolProfile>? entities)
    {
        if (entities is null)
            return null;

        var responses = new List<SchoolProfileModelResponse>();

        foreach (var entity in entities)
        {
            var response =
            _mapper.Map<SchoolProfileModelResponse>(entity);

            if (entity.Img is not null)
            {
                var image = _filesManager.GetFile(entity.Img);
                response.Img = image.IsRight ? null :
                ((FileSuccess)image).Url;
            }

            switch (entity)
            {
                case { Type: SchoolProfileType.SchoolAdmin or
                SchoolProfileType.Teacher }:
                {
                    var school = await
                    _queryContext.Schools.FindAsync(entity.SchoolId);
                    response.SchoolName = school?.Name;
                    response.SchoolId = school?.Id;
                    break;
                }
            }
        }
    }

```

```

        case { Type: SchoolProfileType.Student }:
        {
            var group = await _queryContext.Groups
                .Include(group => group.School)
                .FirstOrDefaultAsync(g => g.Id ==
entity.GroupId);

            response.SchoolName = group?.School.Name;
            response.SchoolId = group?.School.Id;
            break;
        }
        case { Type: SchoolProfileType.ClassTeacher }:
        {
            var group = await _queryContext.Groups
                .Include(group => group.School)
                .FirstOrDefaultAsync(g => g.Id ==
entity.ClassTeacherGroupId);

            response.SchoolName = group?.School.Name;
            response.SchoolId = group?.School.Id;
            break;
        }
    }

    if (string.IsNullOrEmpty(entity.Data))
    {
        responses.Add(response);
        continue;
    }

    switch (entity)
    {
        case { Type: SchoolProfileType.Parent }:
        {
            var data =
JsonConvert.DeserializeObject<ParentSerializationData>(entity.Data);
            response.ParentAddress = data?.ParentAddress;
            break;
        }
        case { Type: SchoolProfileType.Teacher }:
        {
            var data =
JsonConvert.DeserializeObject<TeacherSerializationData>(entity.Data);
            response.TeacherSubjects = data?.TeachersSubjects;
            response.TeacherEducation =
data?.TeachersEducation;
            response.TeacherQualification =
data?.TeachersQualification;
            response.TeacherExperience =
data?.TeachersExperience;
            response.TeacherLessonsPerCycle =
data?.TeachersLessonsPerCycle;

```

		Бубенко О.В.			ІІЗ.КР.Б-121-24-ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		107

```

        break;
    }
    case { Type: SchoolProfileType.Student }:
    {
        var data =
        JsonConvert.DeserializeObject<StudentSerializationData>(entity.Data);
        response.StudentDateOfBirth =
        data?.StudentDateOfBirth;
        response.StudentAptitudes =
        data?.StudentAptitudes;
        response.StudentIsClassLeader =
        data?.StudentIsClassLeader;
        response.StudentIsIndividually =
        data?.StudentIsIndividually;
        response.StudentHealthGroup =
        data?.StudentHealthGroup;
        break;
    }
}

responses.Add(response);
}

return responses.Count != 0 ? responses : null;
}

private static string GetCacheKey(Guid userId)
{
    return $"SchoolProfiles_{userId}";
}
}

using AesAlgorithm = System.Security.Cryptography.Aes;
namespace SchoolService.Application.Common.Cryptography.Aes;

public class AesCipher : IAesCipher
{
    private readonly byte[] _encryptionKeyBytes;
    private readonly byte[] _initializationVectorBytes;

    public AesCipher(IConfiguration configuration)
    {
        var encryptionKeyBase64 = configura-
        tion["EncryptionSettings:EncryptionKeyBase64"]!;
        _encryptionKeyBytes = Convert.FromBase64String(encryptionKeyBase64);

        var initializationVectorBase64 = configura-
        tion["EncryptionSettings:InitializationVectorBase64"]!;
        _initializationVectorBytes = Con-
        vert.FromBase64String(initializationVectorBase64);
    }

    public string Encrypt(string plainText)
    {

```

```

        using var aes = AesAlgorithm.Create();
        aes.Key = _encryptionKeyBytes;
        aes.IV = _initializationVectorBytes;

        var encryptor = aes.CreateEncryptor(aes.Key, aes.IV);
        var plainBytes = Encoding.UTF8.GetBytes(plainText);

        using var memoryStream = new MemoryStream();
        using (var cryptoStream = new CryptoStream(memoryStream, encryptor, Cryptography.CryptoStreamMode.Write))
        {
            cryptoStream.Write(plainBytes, 0, plainBytes.Length);
        }

        return Convert.ToBase64String(memoryStream.ToArray());
    }

    public string Decrypt(string cipherText)
    {
        var cipherBytes = Convert.FromBase64String(cipherText);

        using var aes = AesAlgorithm.Create();
        aes.Key = _encryptionKeyBytes;
        aes.IV = _initializationVectorBytes;

        var decryptor = aes.CreateDecryptor(aes.Key, aes.IV);

        using var memoryStream = new MemoryStream(cipherBytes);
        using var cryptoStream = new CryptoStream(memoryStream, decryptor, Cryptography.CryptoStreamMode.Read);
        using var streamReader = new StreamReader(cryptoStream);
        return streamReader.ReadToEnd();
    }
}

"ReverseProxy": {
  "Routes": {
    "school": {
      "ClusterId": "school",
      "AuthorizationPolicy": "default",
      "Match": {
        "Path": "/school/{**catch-all}"
      },
      "Transforms": [
        {
          "PathPattern": "api/school/{**catch-all}"
        }
      ]
    },
    "joiningRequest": {
      "ClusterId": "school",
      "AuthorizationPolicy": "anonymous",
      "Match": {
        "Path": "/joiningRequest/{**catch-all}"
      },
      "Transforms": [
        {
          "PathPattern": "api/joiningRequest/{**catch-all}"
        }
      ]
    },
    "schoolProfile": {
      "ClusterId": "school",
      "AuthorizationPolicy": "default",
      "Match": {

```

```

        "Path": "/schoolProfile/{**catch-all}"
    },
    "Transforms": [
        {
            "PathPattern": "api/schoolProfile/{**catch-all}"
        }
    ]
},
"group": {
    "ClusterId": "school",
    "AuthorizationPolicy": "default",
    "Match": {
        "Path": "/group/{**catch-all}"
    },
    "Transforms": [
        {
            "PathPattern": "api/group/{**catch-all}"
        }
    ]
},
"groupNotice": {
    "ClusterId": "school",
    "AuthorizationPolicy": "default",
    "Match": {
        "Path": "/groupNotice/{**catch-all}"
    },
    "Transforms": [
        {
            "PathPattern": "api/groupNotice/{**catch-all}"
        }
    ]
},
"studyPeriod": {
    "ClusterId": "school",
    "AuthorizationPolicy": "default",
    "Match": {
        "Path": "/studyPeriod/{**catch-all}"
    },
    "Transforms": [
        {
            "PathPattern": "api/studyPeriod/{**catch-all}"
        }
    ]
}
},
"Clusters": {
    "school": {
        "Destinations": {
            "schoolApi": {
                "Address": "http://localhost:7001"
            }
        }
    }
}
}
}
}

```

ЛІСТИНГ ВИХІДНОГО КОДУ КЛІЄНТСЬКОЇ ЧАСТИНИ ЗАСТОСУНКУ

```

import type { ApiError } from '@shared/types';
import { getCurrentUserToken } from './auth';

const baseUrl = process.env.API_GATEWAY_URL;

async function get(url: string) {
  const requestOptions = {
    method: 'GET',
    headers: await getHeaders(),
  };

  const response = await fetch(baseUrl + url, requestOptions);

  return await handleResponse(response);
}

async function create(
  url: string,
  body: any,
  isFormData = false,
  schoolProfileId = null,
) {
  const requestOptions = {
    method: 'POST',
    headers: await getHeaders(isFormData, schoolProfileId),
    body: isFormData ? body : JSON.stringify(body),
  };

  const response = await fetch(baseUrl + url, requestOptions);
  return await handleResponse(response);
}

async function update(
  url: string,
  body: any,
  isFormData = false,
  schoolProfileId = null,
) {
  const requestOptions = {
    method: 'PUT',
    headers: await getHeaders(isFormData, schoolProfileId),
    body: isFormData ? body : JSON.stringify(body),
  };

  const response = await fetch(baseUrl + url, requestOptions);
  return await handleResponse(response);
}

```

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		111

```

async function partialUpdate(
  url: string,
  body: any,
  isFormData = false,
  schoolProfileId = null,
) {
  const requestOptions = {
    method: 'PATCH',
    headers: await getHeaders(isFormData, schoolProfileId),
    body: isFormData ? body : JSON.stringify(body),
  };

  const response = await fetch(baseUrl + url, requestOptions);
  return await handleResponse(response);
}

async function remove(url: string, schoolProfileId = null) {
  const requestOptions = {
    method: 'DELETE',
    headers: await getHeaders(false, schoolProfileId),
  };

  const response = await fetch(baseUrl + url, requestOptions);

  return await handleResponse(response);
}

async function handleResponse(response: Response) {
  const text = await response.text();

  let data;
  try {
    data = JSON.parse(text);
  } catch (error) {
    data = text;
  }

  if (response.ok) {
    return data || response.statusText;
  } else {
    const error: ApiError = {
      title: data.title,
      type: data.type,
      status: data.status,
      detail: data.detail,
      params: data.params,
      message:
        typeof data === 'string' && data.length > 0
          ? data
    };
  }
}

```

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		112


```

        : response.statusText,
    };

    return { error };
  }
}

async function getHeaders(isFormData = false, schoolProfileId = null) {
  const token = await getCurrentUserToken();
  const headers: any = {};

  if (token) {
    headers.Authorization = 'Bearer ' + token.access_token;
  }

  if (!isFormData) {
    headers['Content-Type'] = 'application/json';
  }

  if (schoolProfileId) {
    headers['SchoolProfileId'] = schoolProfileId;
  }

  return headers;
}

export const api = {
  get,
  create,
  update,
  partialUpdate,
  remove,
};

'use client';

import { useEffect, useState } from 'react';
import { useAdminStore } from '@/features/admin/store/adminStore';
import { useRouter } from 'next/navigation';
import { useSession } from 'next-auth/react';
import { useUserStore } from '@features/user/store/userStore';

export const useAuthRedirectByRole = (activeLocale, requiredRole = null) => {
  const { data: userData, status: userStatus } = useSession();
  const router = useRouter();
  const currentUser = userData?.user;
  const attempts = 2;
  const [attemptCount, setAttemptCount] = useState(0);
  const [isLoading, setIsLoading] = useState(true);

```

```

useEffect(() => {
  if (userStatus === 'loading') return;

  const handleRedirect = () => {
    if (userStatus === 'unauthenticated') {
      router.replace(`/${activeLocale}/access-denied/401`);
    } else if (requiredRole) {
      if (requiredRole === 'admin' && currentUser?.role !== 'admin') {
        router.replace(`/${activeLocale}/access-denied/403`);
      } else if (
        requiredRole === 'user' &&
        !(currentUser?.role === 'user' || currentUser?.role === 'admin')
      ) {
        router.replace(`/${activeLocale}/access-denied/403`);
      } else if (
        requiredRole === 'userOnly' &&
        currentUser?.role !== 'user'
      ) {
        router.replace(`/${activeLocale}/access-denied/403`);
      }
    }
    setIsLoading(false);
  };

  if (
    userStatus === 'unauthenticated' ||
    (userStatus === 'authenticated' &&
      requiredRole &&
      ((requiredRole === 'admin' && currentUser?.role !== 'admin') ||
        (requiredRole === 'user' &&
          !(currentUser?.role === 'user' || currentUser?.role === 'admin')) ||
        (requiredRole === 'userOnly' && currentUser?.role !== 'user')))
  ) {
    if (attemptCount < attempts) {
      setTimeout(() => {
        setAttemptCount(attemptCount + 1);
      }, 1000);
    } else {
      handleRedirect();
    }
  } else {
    setIsLoading(false);
  }
}, [
  userStatus,
  currentUser,
  activeLocale,
  requiredRole,
  attemptCount,
  router,

```

```

]);

useEffect(() => {
  if (
    userStatus === 'authenticated' &&
    (!requiredRole ||
      (requiredRole === 'admin' && currentUser?.role === 'admin') ||
      (requiredRole === 'user' &&
        (currentUser?.role === 'user' || currentUser?.role === 'admin')) ||
      (requiredRole === 'userOnly' && currentUser?.role === 'user'))
  ) {
    setIsLoading(false);
  }
}, [userStatus, currentUser, requiredRole]);

return {
  isUserLoading: userStatus === 'loading' || isLoading,
  isAuthenticated: userStatus === 'authenticated',
  user: currentUser,
};
};

export const useSetCurrentTab = (currentTab) => {
  const setCurrentTab = useAdminStore((store) => store.setCurrentTab);
  const setUserCurrentTab = useUserStore((store) => store.setCurrentTab);

  useEffect(() => {
    setCurrentTab(currentTab);
    setUserCurrentTab(currentTab);
  }, [setCurrentTab, setUserCurrentTab, currentTab]);
};

export const useScrollOffset = () => {
  const [topOffset, setTopOffset] = useState(0);

  const handleScroll = () => {
    const offset =
      window.pageYOffset ||
      document.documentElement.scrollTop ||
      document.body.scrollTop;

    setTopOffset(offset);
  };

  useEffect(() => {
    window.addEventListener('scroll', handleScroll);

    return () => {
      window.removeEventListener('scroll', handleScroll);
    };
  });
};

```

```

    }, []));

    return topOffset;
};

import NextAuth, { type NextAuthOptions, Session } from 'next-auth';
import DuendeIdentityServer6 from 'next-auth/providers/duende-identity-server6';
import jwt from 'jsonwebtoken';
import axios from 'axios';
import { type JWT } from 'next-auth/jwt';

export const authOptions: NextAuthOptions = {
  session: {
    strategy: 'jwt',
  },
  providers: [
    DuendeIdentityServer6({
      id: 'id-server',
      clientId: process.env.AUTH_CLIENT_ID,
      clientSecret: process.env.AUTH_CLIENT_SECRET,
      issuer: process.env.AUTH_ISSUER,
      authorization: {
        params: {
          scope: 'openid profile SeminariumApp',
        },
      },
      idToken: true,
    }),
  ],
  callbacks: {
    async jwt({ token, profile, account }) {
      if (profile) {
        token.username = profile['preferred_username'];
      }
      if (account) {
        token.access_token = account.access_token;
      }

      const decodedToken = jwt.decode(token.access_token);
      token.role = decodedToken.role;
      token.id = decodedToken.sub;

      return token;
    },
    async session({ session, token }) {
      if (token) {
        session.user.id = token.id;
        session.user.role = token.role;
        session.user.username = token.username;
      }
    },
  },
};

```

```

        session.user.email = token.username;
    }

    return session;
},
},
events: {
    async signOut({ session, token }) {
        await axios.get(process.env.AUTH_ISSUER_LOGOT);

        token = {} as JWT;
        session = {} as Session;
    },
},
};

const handler = NextAuth(authOptions);

export { handler as GET, handler as POST };

import { useNavStore } from '@features/nav';
import { CurrentTab } from '@features/user/constants';
import { useUserStore } from '@features/user/store/userStore';
import type { SchoolProfileResponse } from
'@features/user/types/schoolProfileTypes';
import { Tooltip } from 'flowbite-react';
import {
    ChevronsLeft,
    Contact,
    Home,
    Hourglass,
    LibraryBig,
    School,
    User,
    Users,
} from 'lucide-react';
import { useLocale, useTranslations } from 'next-intl';
import Link from 'next/link';
import { FC } from 'react';

interface SchoolAdminContentTabsProps {
    activeProfile: SchoolProfileResponse;
}

const SchoolAdminContentTabs: FC<SchoolAdminContentTabsProps> = ({
    activeProfile,
}) => {
    const activeLocale = useLocale();
    const t = useTranslations('UserContentTabs');

```

```

const currentTab = useUserStore((state) => state.currentTab);
const setSidebarOpen = useNavStore((store) => store.setSidebarOpen);

return (
  <>
    <Tooltip content={t('profile')} placement="right" style="light">
      <Link
        href={`/${activeLocale}/u/`}
        className={`flex h-[50px] w-[50px] items-center justify-center
          ${currentTab === CurrentTab.Profile ? `bg-purple-950` : `bg-gray-50 hover:bg-gray-200`}
          text-gray-800 transition duration-300`}
      >
        <User
          color={` ${currentTab === CurrentTab.Profile ? `#f9fafb` : `#3B0764`} `}
          size={20}
        />
      </Link>
    </Tooltip>

    <Tooltip content={t('courses')} placement="right" style="light">
      <Link
        href={`/${activeLocale}/u/courses`}
        className={`flex h-[50px] w-[50px] items-center justify-center
          ${currentTab === CurrentTab.Course ? `bg-purple-950` : `bg-gray-50 hover:bg-gray-200`}
          text-gray-800 transition duration-300`}
      >
        <LibraryBig
          color={` ${currentTab === CurrentTab.Course ? `#f9fafb` : `#3B0764`} `}
          size={20}
        />
      </Link>
    </Tooltip>

    <Tooltip content={t('school')} placement="right" style="light">
      <Link
        href={`/${activeLocale}/u/my-school/${activeProfile.schoolId}`}
        className={`flex h-[50px] w-[50px] items-center justify-center
          ${currentTab === CurrentTab.School ? `bg-purple-950` : `bg-gray-50 hover:bg-gray-200`}
          text-gray-800 transition duration-300`}
      >
        <School
          color={` ${currentTab === CurrentTab.School ? `#f9fafb` : `#3B0764`} `}
          size={20}
        />
      </Link>
    </Tooltip>
  </>
)

```

```

</Tooltip>

<Tooltip content={t('groups')} placement="right" style="light">
  <Link
    href={`/${activeLocale}/u/groups`}
    className={`flex h-[50px] w-[50px] items-center justify-center
      ${currentTab === CurrentTab.Group ? `bg-purple-950` : `bg-gray-50 hover:bg-gray-200`}
    text-gray-800 transition duration-300`}
  >
    <Users
      color={` ${currentTab === CurrentTab.Group ? `#f9fafb` : `#3B0764`} `}
      size={20}
    />
  </Link>
</Tooltip>

<Tooltip content={t('schoolProfiles')} placement="right" style="light">
  <Link
    href={`/${activeLocale}/u/school-profile/all/`}
    className={`flex h-[50px] w-[50px] items-center justify-center
      ${currentTab === CurrentTab.SchoolProfiles ? `bg-purple-950` : `bg-gray-50 hover:bg-gray-200`}
    text-gray-800 transition duration-300`}
  >
    <Contact
      color={` ${currentTab === CurrentTab.SchoolProfiles ? `#f9fafb` : `#3B0764`} `}
      size={20}
    />
  </Link>
</Tooltip>

<Tooltip content={t('studyPeriods')} placement="right" style="light">
  <Link
    href={`/${activeLocale}/u/study-periods/`}
    className={`flex h-[50px] w-[50px] items-center justify-center
      ${currentTab === CurrentTab.StudyPeriods ? `bg-purple-950` : `bg-gray-50 hover:bg-gray-200`}
    text-gray-800 transition duration-300`}
  >
    <Hourglass
      color={` ${currentTab === CurrentTab.StudyPeriods ? `#f9fafb` : `#3B0764`} `}
      size={20}
    />
  </Link>
</Tooltip>

<Tooltip content={t('home')} placement="right" style="light">

```

```

        <Link
          href={`/${activeLocale}/u/`}
          className={`mt-[50px] flex h-[50px] w-[50px] items-center justify-
center bg-gray-50 text-gray-800
          transition duration-300 hover:bg-gray-200`}
        >
          <Home color={`#3B0764`} size={20} />
        </Link>
      </Tooltip>

      <Tooltip content={t('hide')} placement="right" style="light">
        <div
          onClick={() => setSidebarOpen(false)}
          className={`flex h-[50px] w-[50px] cursor-pointer items-center justify-
center bg-gray-50 text-gray-800
          transition duration-300 hover:bg-gray-200`}
        >
          <ChevronsLeft color={`#3B0764`} size={20} />
        </div>
      </Tooltip>
    </>
  );
};

export { SchoolAdminContentTabs };

'use client';

import { useEffect, useState } from 'react';
import type { ApiResponse } from '@shared/types';
import type { SchoolProfileResponse } from
'@/features/user/types/schoolProfileTypes';
import { useSchoolProfilesStore } from
'@/features/user/store/schoolProfilesStore';
import { get } from '@features/user/api/schoolProfilesApi';

export const useProfiles = () => {
  const profilesStore = useSchoolProfilesStore();

  const [isLoading, setIsLoading] = useState(true);

  const [profiles, setProfiles] =
    useState<ApiResponse<SchoolProfileResponse[]>>(null);
  const [activeProfile, setActiveProfile] =
    useState<SchoolProfileResponse>(null);
  const [isError, setIsError] = useState(false);

  useEffect(() => {
    const fetchData = async () => {
      setIsLoading(true);

```



```

        setIsError(false);

        if (profilesStore.profiles) {
            setProfiles(profilesStore.profiles);
            setActiveProfile(profilesStore.activeProfile);

            setIsLoading(false);
            return;
        }

        try {
            const response = await get();

            if (response && response.error) {
                setIsError(true);
                return;
            }

            setProfiles(response);
            profilesStore.setProfiles(response);

            setActiveProfile(activeProfile);
            return;
        } catch {
            setIsError(true);
        } finally {
            setIsLoading(false);
        }
    };

    fetchData();
}, [
    profilesStore.profiles,
    profilesStore.activeProfile,
    profilesStore.setProfiles,
    profilesStore.changeActiveProfile,
    profilesStore,
    activeProfile,
]);

const currentProfile =
    activeProfile || profiles?.find((profile) => profile.isActive) || null;

return {
    isLoading,
    isError: isError && !profiles,
    profiles,
    activeProfile: currentProfile,
};
};

```

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		121

```

import type { SchoolProfileResponse } from
'@/features/user/types/schoolProfileTypes';
import { create } from 'zustand';
import { devtools } from 'zustand/middleware';
import { immer } from 'zustand/middleware/immer';

interface SchoolProfilesStore {
  activeProfile: SchoolProfileResponse | null;
  profiles: SchoolProfileResponse[] | null;
  setProfiles: (profile: SchoolProfileResponse[]) => void;
  clear: () => void;
  changeActiveProfile: (id: string) => void;
  changeImg: (id: string, newImg: string) => void;
}

export const useSchoolProfilesStore = create<SchoolProfilesStore>()(
  immer(
    devtools(
      (set) => ({
        activeProfile: null,
        profiles: null,
        setProfiles: (profiles: SchoolProfileResponse[]) =>
          set((state) => {
            state.profiles = profiles;
            state.activeProfile = profiles.find((profile) => profile.isActive);
          }),
        changeImg: (id: string, newImg: string) =>
          set((state) => {
            const updatedProfiles = state.profiles.map((profile) => {
              if (profile.id === id) {
                return {
                  ...profile,
                  img: newImg,
                };
              }
              return profile;
            });
            return {
              profiles: updatedProfiles,
              activeProfile: updatedProfiles.find(
                (profile) => profile.isActive,
              ),
            };
          }),
        changeActiveProfile: (id: string) =>
          set((state) => {
            const updatedProfiles = state.profiles.map((profile) => {
              return {
                ...profile,

```

```
        isActive: profile.id === id,
      });
    });

    return {
      profiles: updatedProfiles,
      activeProfile: updatedProfiles.find(
        (profile) => profile.isActive,
      ),
    };
  }),
  clear: () =>
    set((state) => {
      state.profiles = null;
      state.activeProfile = null;
    }),
  { name: 'School profiles store' },
),
),
);
```

		Бубенко О.В.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		123