НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Вебзастосунок для організації дистанційного навчання

Виконав студент IV курсу, групи ІП-02
(шифр групи)

Демченко Олексій Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник асистент Тихонов С.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., Ткач М.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Київ – 2024

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Демченку Олексію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Вебзастосунок для організації дистанційного навчання
керівник проєкту Тихонов Сергій Вадимович, асистент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 27 » травня 2024 р. №2112-с

2. Термін подання студентом проєкту « 17 » червня 2024 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Передпроєктне обстеження предметної області: аналіз предметної області, аналіз існуючих рішень, аналіз відомих програмних продуктів, аналіз відомих технічних рішень, опис бізнес-процесів, постановка задачі.
2) Розроблення вимог до програмного забезпечення: варіанти використання програмного забезпечення, розроблення функціональних вимог, розроблення нефункціональних вимог.
3) Конструювання та розроблення програмного забезпечення: архітектура програмного забезпечення, обґрунтування вибору засобів розробки, конструювання програмного забезпечення, аналіз безпеки даних.
4) Аналіз якості та тестування програмного забезпечення: аналіз якості програмного забезпечення, опис процесів тестування, опис контрольного прикладу.
5) Розгортання та супровід програмного забезпечення: розгортання програмного забезпечення, супровід програмного забезпечення.

5. Перелік графічного матеріалу

1) Креслення вигляду екранних форм
2) Схема структурна класів програмного забезпечення
3) Схема структурна класів програмного забезпечення
4) Схема бази даних
5) Схема варіантів використання

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «11» березня 2024 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	11.03.2024	
2	Аналіз існуючих методів розв’язання задачі	20.03.2024	
3	Постановка та формалізація задачі	25.03.2024	
4	Розробка інформаційного забезпечення	30.03.2024	
5	Алгоритмізація задачі	05.04.2024	
6	Обґрунтування вибору використаних технічних засобів	10.04.2024	
7	Розробка програмного забезпечення	15.04.2024	
8	Налагодження програми	10.05.2024	
9	Виконання графічних документів	20.05.2024	
10	Оформлення пояснювальної записки	29.06.2024	
11	Подання ДП на попередній захист	02.06.2024	
12	Подання ДП рецензенту	10.06.2024	
13	Подання ДП на основний захист	14.06.2024	

Студент _____ Олексій ДЕМЧЕНКО
(підпис) (ініціали, прізвище)

Керівник _____ Сергій ТИХОНОВ
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п’ятьох розділів, містить 89 таблиць, 67 рисунків та 18 джерел – загалом 133 сторінки.

Дипломний проєкт присвячений розробці програмного забезпечення, що забезпечує можливість зручного дистанційного навчання наданням необхідної функціональності для викладачів та студентів.

Мета: підвищення якості освіти та спрощення організаційних питань з приводу дистанційного навчання.

Об'єкт дослідження: вебзастосунок для організації дистанційного навчання.

Предмет дослідження: забезпечення якості, впровадження і супроводження вебзастосунку.

У розділі передпроєктного обстеження предметної області наведено аналіз предметної області та існуючих технічних рішень з дистанційного навчання, а також сформовано постановку задачі дипломного проєкту.

Розділ розроблення вимог до програмного забезпечення складається з варіантів використання програмного забезпечення, сформованих функціональних та нефункціональних вимог.

Розділ конструювання та розроблення програмного забезпечення містить розробку архітектури програмного забезпечення, обґрунтування обраних програмних засобів, конструювання та аналіз безпеки програмного забезпечення.

У розділі аналізу якості та тестування програмного забезпечення перевіряється якість розробки, тестується розроблене програмне забезпечення відповідно до розроблених тестів та наведено контрольний приклад.

Розділ розгортання та супроводу програмного забезпечення містить детальний опис розгортання за допомогою інструменту Docker, описано процес супроводу забезпечення.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, ДИСТАНЦІЙНЕ НАВЧАННЯ, ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД, ОНЛАЙН НАВЧАННЯ, ПЛАТФОРМА ДЛЯ НАВЧАННЯ.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 89 tables, 67 figures and 18 sources – in total 133 pages.

The diploma project is dedicated to the development of software that enables convenient remote learning by providing the necessary functionality for teachers and students.

Goal: to improve the quality of education and simplify organizational issues related to distance learning.

Object of research: a web application for managing remote learning.

Subject of research: quality assurance, implementation and maintenance of the web application.

The section of the pre-project survey of the subject area provides an analysis of the subject area and existing technical solutions for distance learning, as well as formulating the task of the diploma project.

The section on developing software requirements consists of software use cases, functional and non-functional requirements.

The software design and development section includes software architecture development, justification of the selected software tools, software security design and analysis.

The section on quality analysis and software testing checks the quality of development, tests the developed software in accordance with the developed tests, and provides a control example.

The section on software deployment and maintenance contains a detailed description of deployment using the Docker tool, and the process of software maintenance is described.

KEYWORDS: WEB APPLICATION, DISTANCE LEARNING, HIGHER EDUCATION INSTITUTION, ONLINE LEARNING, PLATFORM FOR STUDYING.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-0123.045440.01.91	Технічне завдання	18	
3	А4	КПІ.ІП-0123.045440.02.81	Пояснювальна записка	133	
4	А4	КПІ.ІП-0123.045440.03.12	Текст програми	106	
5	А4	КПІ.ІП-0123.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-0123.045440.05.34	Керівництво користувача	15	
7	А3	КПІ.ІП-0123.045440.06.99	Графічний матеріал	5	

КПІ.ІП-0217.045440.00.90

Змін.	Арк.	№ докум.	Підп.	Дата
Розроб.		Демченко О.С.		
Перевір.		Тихонов С.В.		
Н.контр.		Шулькевич Т.В.		
Затв.		Жаріков Е.В.		

Відомість дипломного проєкту

Літ.	Аркуш	Аркушів
	1	

КПІ ім. Ігоря Сікорського
ФІОТ каф. ІПІ гр. ІП-02

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНОГО НАВЧАННЯ

Технічне завдання

КПІ.ІП-0217.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Сергій ТИХОНОВ

Нормоконтроль:
_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:
_____ Олексій ДЕМЧЕНКО

Київ – 2024

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: вебзастосунок для організації дистанційного навчання.

Галузь застосування: викладачі та студенти під час навчання або викладання у вищому навчальному закладі.

Наведене технічне завдання поширюється на розробку вебзастосунку для орагіназції дистанційного навчання КПІ.ІП-0217.045440, котрий використовується для зручного дистанційного навчання завдяки наданню необхідних функціональних можливостей для студентів та викладачів, призначена для навчання або викладання у вищому навчальному закладі.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки вебзастосунку для організації дистанційного навчання є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для дистанійного навчання та викладання у вищому навчальному закладі.

Метою розробки є підвищення якості освіти та спрощення організаційних питань з приводу дистанційного навчання у вищих навчальних закладах.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

– сторінка реєстрації відповідно до рисунку 4.1;

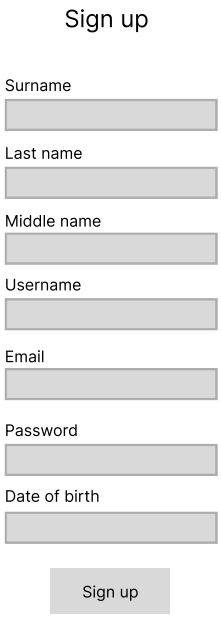


Рисунок 4.1 – Екранна форма сторінки реєстрації

– сторінка авторизації відповідно до рисунку 4.2;

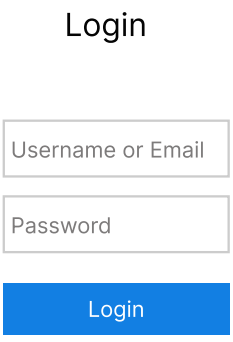


Рисунок 4.2 – Екранна форма сторінки авторизації

– сторінка чатів відповідно до рисунку 4.3;

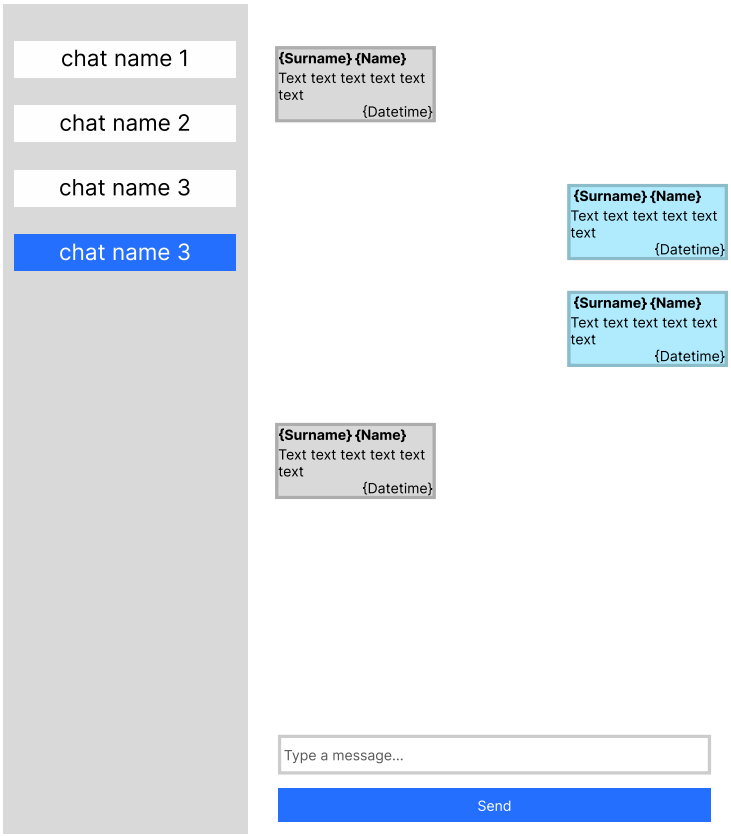


Рисунок 4.3 – Екранна форма сторінки чату

– сторінка адміністратора відповідно до рисунку 4.4;

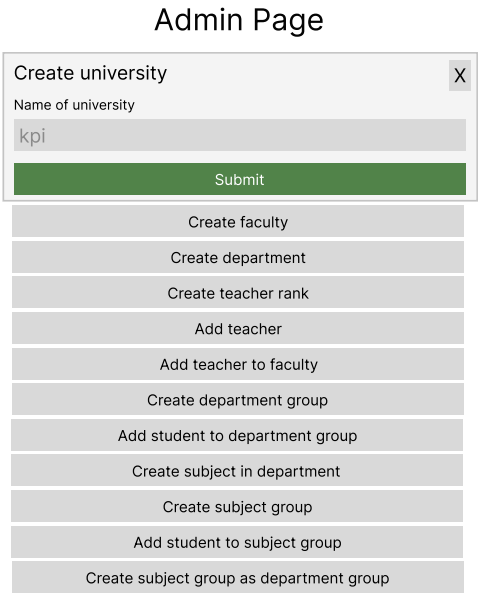


Рисунок 4.4 – Екранна форма сторінки адміністратора

– сторінка завдань викладача відповідно до рисунку 4.5;

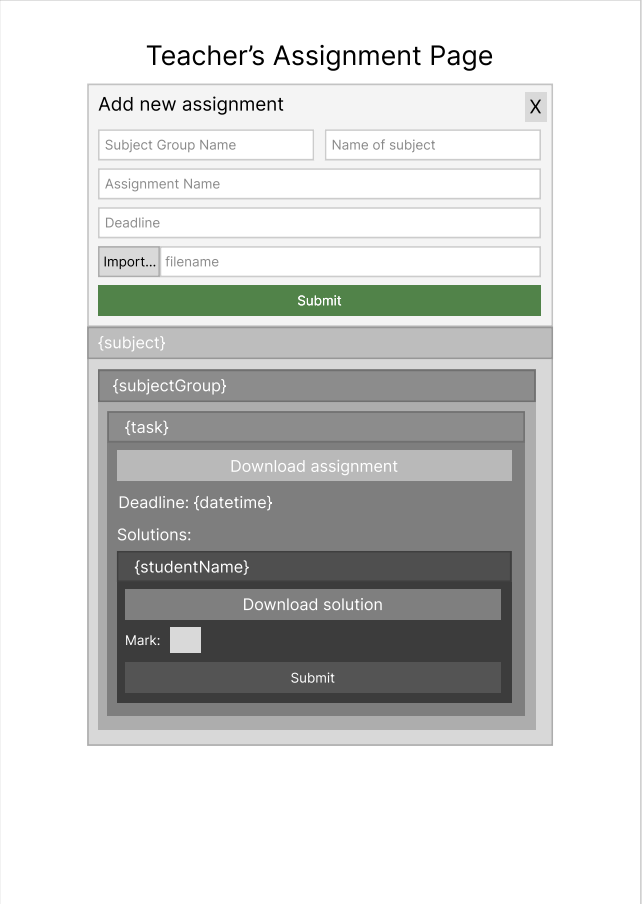


Рисунок 4.5 – Екранна форма сторінки завдань викладача

– сторінка завдань студента відповідно до рисунку 4.6;

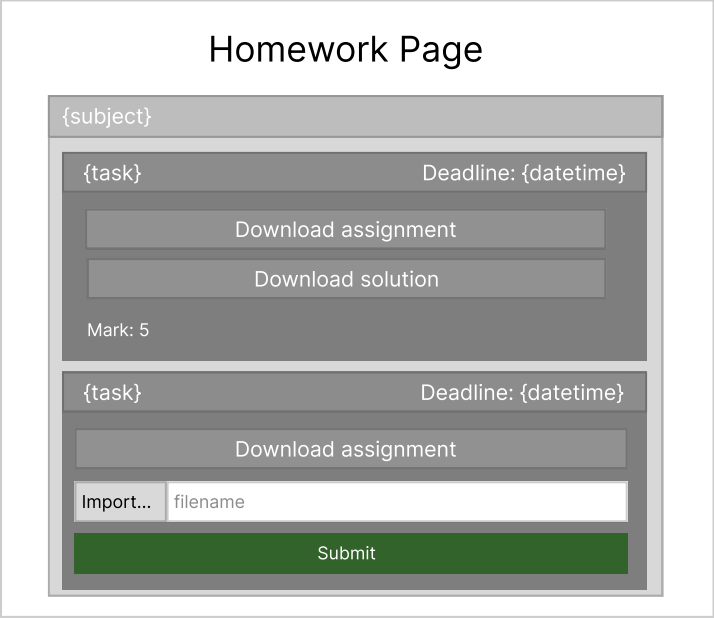


Рисунок 4.6 – Екранна форма сторінки завдань студента

– сторінка щоденника для студента згідно рисунку 4.7.

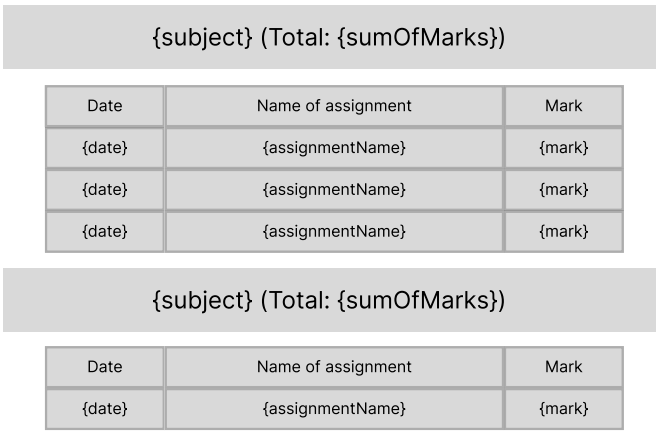


Рисунок 4.7 – Екранна форма сторінки щоденника

4.1.2 Для користувача:

– можливість реєстрації;
– можливість авторизації.

4.1.3 Для адміністратора системи:

– можливість створити університет;
– можливість створити факультет;
– можливість створити кафедру;
– можливість створити рівень вчителя;
– можливість надати користувачу права вчителя;
– можливість додати вчителя до факультету;
– можливість створити групу на кафедрі;
– можливість додати студента до групи на кафедрі;
– можливість створити предмет на кафедрі;
– можливість створити навчальну групу для предмету;
– можливість додати студента до навчальної групи премдету;
– можливість створити навчальну групу, як групу з кафедри.

4.1.4 Для викладача:

– можливість створити завдання для групи предмету;

– можливість створити завантажувати власні завдання у вигляді файлу;

– можливість переглядати список власних завдань;

– можливість оцінювати завдання рішень студентів до власних завдань;

– можливість завантажувати рішення студентів до власних завдань на пристрій у вигляді файлу;

– можливість переглядати повідомлення в чатах, в яких викладає предмет;

– можливість відправляти повідомлення в чатах, в яких викладає предмет.

4.1.5 Для студента:

– можливість переглядати список завдань, наданих його групі предмету;

– можливість завантажувати завдання викладача у вигляді файлу;

– можливість прикріплювати рішення до завдання у вигляді файлу;

– можливість завантажувати власне рішення до завдання у вигляді файлу;

– можливість переглядати оцінки у вигляді таблиці з кожного предмету окремо;

– можливість переглядати сумарну кількість балів з кожного предмету окремо;

– можливість переглядати повідомлення в чатах, в яких навчається на предметі;

– можливість відправляти повідомлення в чатах, в яких навчається на предметі.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висовуються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висовуються.

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 4 Гб;
– підключення до мережі Інтернет зі швидкістю від 10 мегабіт.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;
– об‘єм ОЗП: 16 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем Windows та Unix-подібних, що мають можливість пілключатися до мережі Інтернет та підтримують веб-браузер.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні повинні бути надані користувачем за допомогою вебзастосунку.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені у вигляді екранних форм на інтерфейсі користувача.

4.5.3 Вимоги до мови розробки

Розробку виконати на мовах програмування C++ для серверної частини та JavaScript для клієнтської частини.

4.5.4 Вимоги до середовища розробки

Розробку виконати за допомогою середовища CLion.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді двох репозиторіїв на GitHub.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема бази даних;
– дві схеми структурних класів програмного забезпечення;
– креслення вигляду екранних форм.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	11.03	
2.	Аналіз існуючих методів розв’язання задачі	20.03	
3.	Постановка та формалізація задачі	25.03	Специфікації програмного забезпечення
4.	Розробка інформаційного забезпечення	30.03	Технічна документація
5.	Алгоритмізація задачі	05.04	Схема структурна програмного забезпечення та специфікація компонентів
6.	Обгрунтування вибору використаних технічних засобів	10.04	Технічна документація
7.	Розробка програмного забезпечення	15.04	Тексти програмного забезпечення
8.	Налагодження програми	10.05	Тести, результати тестування
9.	Виконання графічних документів	20.05	Графічний матеріал проєкту
10.	Оформлення пояснювальної записки	29.05	Пояснювальна записка

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: **Вебзастосунок для організації дистанційного навчання**

КПІ.ІП-0217.045440.02.81

Київ – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс
SDK	–	Software development kit
IT	–	Інформаційні технології
ER	–	Entity-Relation diagram
ОС	–	Операційна система.
БД	–	База даних
HTTP	–	протокол передачі даних, що використовується в комп'ютерних мережах. Протокол передачі гіпертекстових документів
AWS	–	Amazon Web Services
AWS S3	–	сервіс-сховище даних, один з Amazon Web Services. Сервіс надає можливість для зберігання й отримання будь-якого обсягу даних, у будь-який час з будь-якої точки мережі, тобто так званий файловий хостинг.
REST API	–	це архітектура інтерфейсу прикладних програм (API), що використовує HTTP-запити для доступу до ресурсів та їхнього використання
Фреймворк	–	це програмне середовище, яке спрощує та прискорює створення програмного забезпечення
Мессенджер	–	це застосунок для обміну миттєвими повідомленнями

ВСТУП

Сучасна динаміка освітнього процесу та зростання викликів, пов'язаних з його організацією та доступністю, народжує необхідність вдосконалення інструментів та платформ, які забезпечують якісне та ефективне навчання на всіх рівнях. У світлі цих викликів, розробка веб-застосунку для організації дистанційного навчання стає актуальним завданням, що спрямоване на забезпечення зручності, доступності та ефективності навчального процесу.

Поступове перетворення сучасного освітнього середовища у віртуальне, посилене впровадженням технологій та інтерактивних методів навчання, відкриває перед нами безліч можливостей для створення інноваційних інструментів, спрямованих на поліпшення навчального процесу. Веб-застосунок, який об'єднує в собі функціональні можливості месенджера для спілкування між викладачами та студентами, сторінки для розміщення домашніх завдань та особистих щоденників, відповідає сучасним потребам освіти та сприяє підвищенню якості та доступності навчання.

Подальше дослідження та розробка веб-застосунку для дистанційного навчання має на меті забезпечити ефективність та зручність навчального процесу, що є важливим кроком у сфері розвитку освіти у сучасному цифровому світі.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Вебзастосунок (іноді вебдодаток) — розподілений застосунок, в якому клієнтом виступає браузер, а сервером — вебсервер. Браузер може бути реалізацією так званих тонких клієнтів — логіка застосунку зосереджується на сервері, а функція браузера полягає переважно у зображенні інформації, завантаженої мережею з сервера, і передачі назад даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому вебзастосунки є міжплатформовими сервісами. Унаслідок цієї універсальності й відносної простоти розробки вебзастосунки стали широко популярними в кінці 1990-х — початку 2000-х років [1].

Дистанційне навчання - це форма навчання, при якій студент та викладач розташовані в різних місцях, а навчальний матеріал передається за допомогою інформаційних технологій через Інтернет або інші, не такі зручні або не такі популярні, засоби зв'язку [2].

Вебзастосунок для організації дистанційного навчання – це застосунок, який дозволяє облегшити організаційні питання, підвищити зручність навчання за допомогою об’єднання способів взаємодії викладачів та студентів, онлайн щоденника, видачі будь-яких завдань студенту: тести після завершення теми, курсові роботи, домашні завдання та інші, та виставлення оцінок за завдання.

Предметна область, що стосується організації дистанційного навчання, включає в себе широкий спектр аспектів, пов'язаних з освітнім процесом, інформаційними технологіями та взаємодією між учасниками навчального процесу. Ця область обумовлена швидким розвитком технологій та змінами в суспільстві, що вимагають нових підходів до навчання та набуття знань.

Напрямки розробок у сфері дистанційного навчання охоплюють різноманітні аспекти, такі як розробка платформ для навчання онлайн,

створення інтерактивних навчальних матеріалів, розробка засобів відстеження успішності студентів та інші.

На сьогоднішній день існують численні платформи та інструменти для дистанційного навчання, які ми розглянемо в наступному розділі, проте багато з них мають обмежені можливості, не завжди задовольняють потреби користувачів через те, що не є об’єднаними у один єдиний або не підтримують навчання саме для університету шляхом розбиття на університети, факультети, кафедри, групи, лекторів та практиків тощо. Також серед існуючих рішень мала доступність вибору мов, де рідко можна зустріти українську. Недоліками поточного стану справ можуть бути недостатня функціональність, складність використання, відсутність інтеграції з іншими інструментами тощо.

Було проведено опитування серед студентів (загальна кількість учасників у опитуванні 59) та визначено, що через те, що кожен викладач використовує свої застосунки для викладання виникає плутанина з приводу організаційних питань: де можна знайти домашнє завдання, куди відправляти вже готове завдання, де знайти список дедлайнів та інколи навіть як зв’язатися

з викладачем (рисунок 1.1).

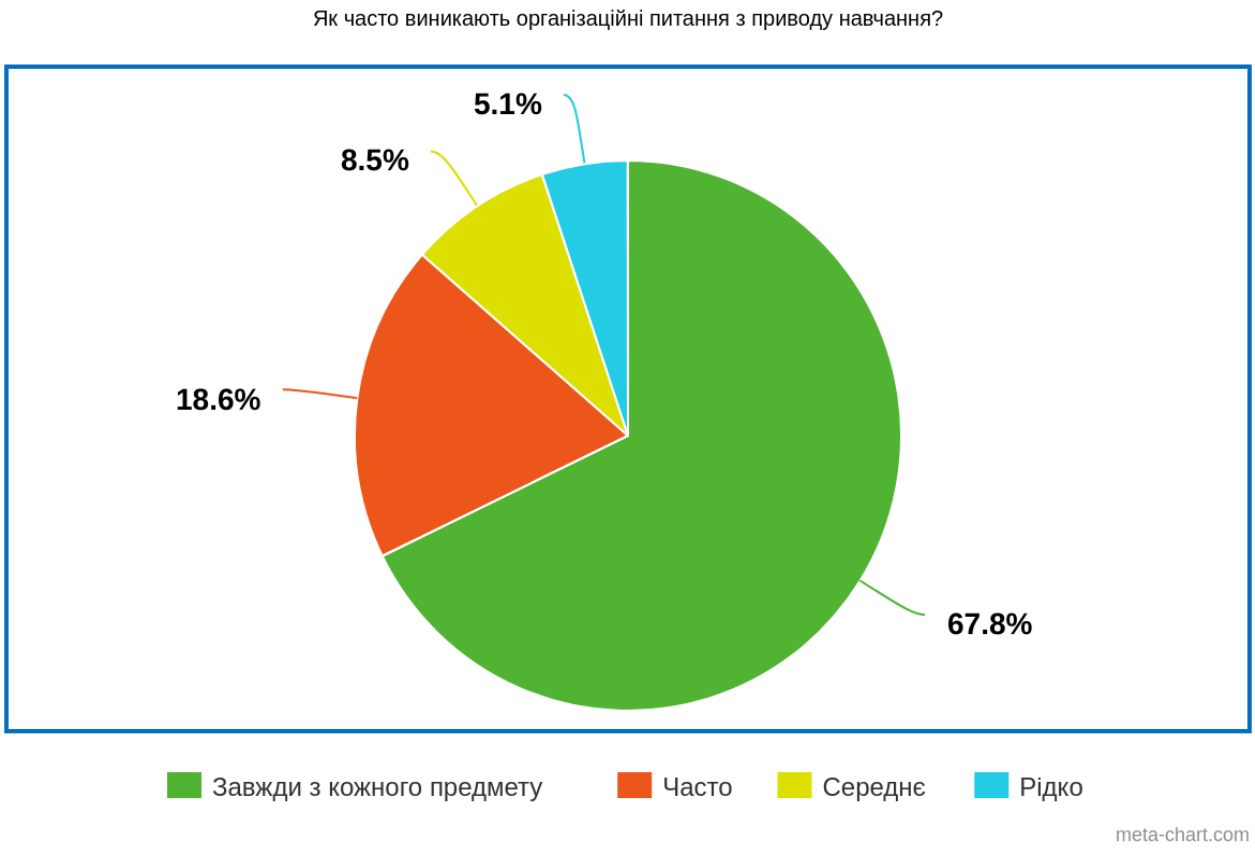


Рисунок 1.1 – Опитування серед студентів з приводу організаційних питань

Шляхи покращення ситуації у сфері розробок для дистанційного навчання можуть включати розробку інноваційних платформ, що поєднують у собі різноманітні функціональні можливості та забезпечують зручність та ефективність користувачам. Іншими напрямками можуть бути вдосконалення інтерфейсів, забезпечення інтеграції з іншими інструментами та розробка персоналізованих рішень.

У рамках дипломного проєкту буде розроблено веб-застосунок, який об'єднає в собі функціональні можливості месенджера для спілкування між викладачами та студентами, можливість розміщення домашніх завдань, їх рішень та особистих щоденників для студентів. Виходячи з аналізу поточного стану справ у сфері дистанційного навчання, цей застосунок буде спрямований на забезпечення зручності, доступності та ефективності навчання, а також на вирішення недоліків та вдосконалення у цій сфері.

1.2 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації вебзастоунку для організації дистанційного навчання. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.2.1 Аналіз відомих програмних продуктів

Серед програмних продуктів будуть розглядатися найвідоміші та найзручніші реалізації застосунків, їх кращі та гірші сторони.

Google Classroom - вебсервіс, створений Google для навчальних закладів з метою спрощення створення, поширення і класифікації завдань безпаперовим шляхом. Основна мета сервісу — прискорити процес поширення файлів між педагогами та здобувачами освіти. Може використовуватися вчителями та учнями у школах, або у закладах вищої освіти викладачами та студентами. Даний застосунок реалізовує процес видачі завдань, закриття завдань через просрочення здачі, здачу завдання та онлайн перегляд завдань. Також він реалізовує комунікацію, але не є саме створений для цього. Сама комунікація не виглядає як Real-Time, а скоріше, як просто написання коментарів до завдання, до рішення завдання [3].

Moodle – вебсервіс, що є аналогом Google Classroom, створений для публікації матеріалів, завдань, проходження онлайн тестів та виставлення оцінок. Процес комунікації аналогічний: через коментарі до завдань, що ускладнює коммунікацію [4].

Microsoft Teams – вебсервіс для Real-Time спілкування людей методами мессенджеру або через мікрофон, але не має можливості виставлення оцінок, викладення завдань, дедлайнів тощо, не є платформою, що спеціалізується на навчанні у вищому навчальному закладі [5].

Telegram – вебсервіс, що реалізовує мессенджер та простоту зв’язку через його популярність, тому що даний додаток є найпопулярнішим

мессенджером телефоні. Його проблемою для навчання є, що він зовсім не призначений для дистанційного навчання [6].

Campus – вебсервіс, що дозволяє переглядати свої оцінки, тобто є електронним щоденником, але його основною проблемою є, що він ніяк не інтегрований через сервіси, наведені вище, через це викладач може або відразу не копіювати з іншого сервісу в даний щоденник або взагалі забувати виставити. У випадку копіювання викладач марнує дорогоцінний час, який міг би витратити на важливіші справи або на своїх студентів.

Вище наведені застосунки вже викликають плутанину через не інтегрування одного з іншим між собою.

Для порівняння проєкту з аналогами можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння дипломного проєкту з найвідомішими аналогами

Функціонал та особливості	Вебзастосунок для організації дистанційного навчання	Google classroom	Moodle	Microsoft Teams	Telegram	Campus	Пояснення
Електронний щоденник	Присутній	Відсутній	Присутній	Відсутній	Відсутній	Присутній	В Google Classroom та Moodle можна переглядати свої оцінки, але вони є не фінальним варіантом, бо частіше університет виставляє оцінки в свій окремий застосунок, наприклад в КПІ, це - Campus

Функціонал та особливості	Вебзастосунок для організації дистанційного навчання	Google classroom	Moodle	Microsoft Teams	Telegram	Campus	Пояснення
Завантаження завдань та їх рішень	Присутній	Присутній	Присутній	Відсутній	Відсутній	Відсутній	Завантажити завдання та їх рішення можна в усіх, крім Campus, але це буде виглядати, як звичайне повідомлення у мессенджері, не буде дедлайну здачі, збору всіх рішень в одну купу тощо
Мессенджер	Присутній	Відсутній	Відсутній	Присутній	Присутній	Відсутній	Серед існуючих мессенджерів є дуже зручна реалізація спілкування. Найкращим є Telegram

1.2.2 Аналіз відомих алгоритмічних та технічних рішень

Перед початком проектуванням архітектури та розробкою завжди постає питання вибору платформи, під яку буде розроблятися застосунок. В даній задачі розглядається вебзастосунок, що є кросс-платформенною середою, тому що запуск застосунку виконується через браузер, що реалізований на всіх відомих найпопулярніших платформах та є світовим стандартом [7].

На наведеному рисунку 1.2 перегляньмо популярність операційних систем.

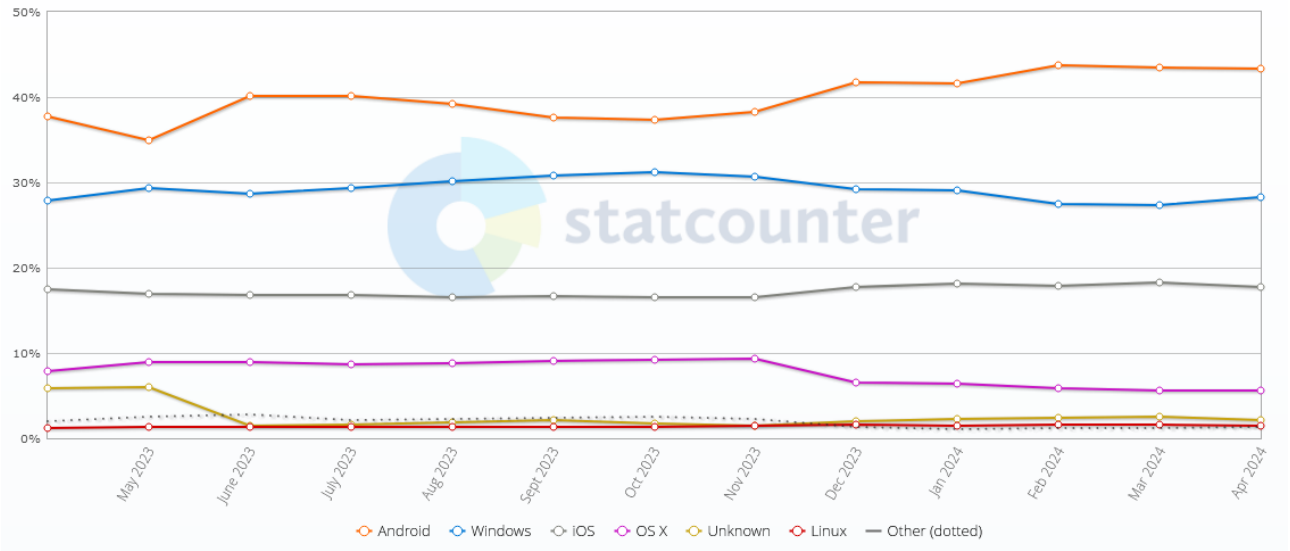


Рисунок 1.2 – Графік популярності операційних систем

На рисунку 1.2 можемо бачити, що операційні системи всі по-своєму популярні та кожен обирає те, що йому зручно, тому був обраний саме вебзастосунок через те, що неважливо, яка саме операційна система буде у користувача, у кожної операційної системи реалізований браузер, який буде комунікувати з майбутнім сервером через клієнт-серверну архітектуру.

Для розробки бекенду була обрана операційна система Unix-подібної системи Linux під назвою Ubuntu через її популярність (рисунок 1.3), простоту та підтримуваність. Користувачу не є важливим, яку платформу підтримує бекенд, через те, що клієнт з сервером буде комунікувати всесвітнім стандартом HTTP, який також підтримується будь-яким браузером [8].

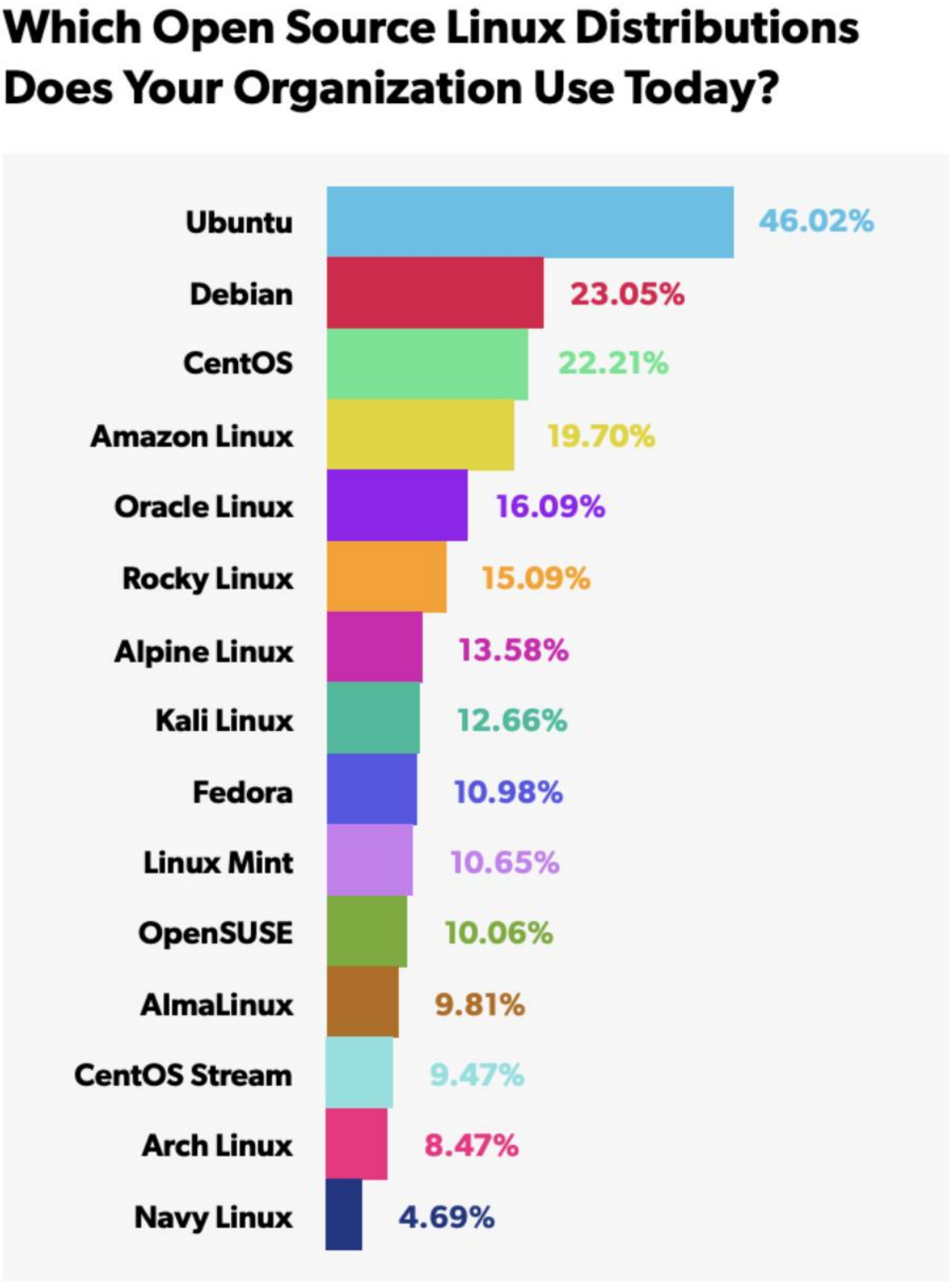


Рисунок 1.3 – Популярність Unix-подібних систем

Серед відомих архітектур є вибір проектування та розробки серед монолітною, клієнт-серверною та peer-to-peer. Тому порівняймо їх між собою:

Монолітна архітектура:

Переваги:

- є простішою в розумінні та розробці, оскільки весь функціонал додатка знаходиться в одному місці;
- управління монолітним додатком може бути простіше, оскільки немає необхідності керувати декількома окремими сервісами;
- розгортання монолітного додатка на віддаленному сервері може бути швидшим, оскільки весь функціонал розглядається як єдине ціле.

Недоліки:

- менша масштабованість через те, що весь функціонал знаходиться в одному місці, що призводить до плутанини;
- є небезпечною у використанні з БД через те, що користувач може отримати доступ до всіх даних та робити з ними, шо заманеться.

Клієнт-серверна архітектура:

Переваги:

- простота управління і супроводу;
- легке масштабування та досить швидке розгортання застосунку;
- більша безпека, оскільки доступ до даних має лише сервер та сервер контролює доступ до даних між користувачами.

Недоліки:

- залежність від доступності сервера;
- збільшене навантаження через велику кількість клієнтів.

Peer-to-peer архітектура:

Переваги:

- більша резистентність до відмов: відсутність одного центрального сервера;
- необмежена масштабованість.

Недоліки:

- складніше управління та супровід;
- потребує більшої кількості ресурсів;
- важке розгортання.

Серед вище наведенних архітектур була обрана клієнт-серверна, що є не досить складною в написанні, як peer-to-peer та набагато безпечнішою за монолітну.

Далі потрібно обрати протокол обміну інформації між клієнтом та сервером. Серед найвідоміших є gRPC та HTTP.

gRPC:

Переваги :

- gRPC використовує бінарний формат передачі даних Protocol Buffers, який забезпечує високу швидкість передачі та зменшення обсягу даних порівняно з текстовими форматами, такими як JSON;
- gRPC надає можливість автоматичної генерації коду для клієнтів та серверів з використанням специфікацій API, що спрощує розробку;
- gRPC підтримує багато мов програмування.

Недоліки:

- бінарний формат даних та код, що генерується автоматично, можуть ускладнити процес розробки порівняно з текстовими форматами, які використовуються в HTTP;
- у випадку зміни специфікації API може знадобитися перегенерація та розгортання коду, що може призвести до проблем з оновленнями.

HTTP:

Переваги:

- є простим у використанні та розумінні протоколом, що дозволяє легко розробляти та налагоджувати веб-застосунки;
- є стандартом для взаємодії у вебі та має широку підтримку у браузерах, веб-серверах та клієнтських додатках;
- дозволяє легко інтегрувати різні компоненти додатків та забезпечує високу сумісність із різноманітними системами.

Недоліки:

- передає дані у текстовому форматі, що може призвести до зайвої накладності у вигляді заголовків та іншої мета-інформації, що впливає на швидкодію передачі.

Серед даних двох був обраний HTTP через його простоту, підтримуваність та стандартизацію.

Вибір бази данних пав на PostgreSQL, яка презентує себе як «Найдосконаліша у світі реляційна база даних з відкритим кодом». Серед вибору були дві найвідоміші MySQL та PostgreSQL. Обидві дані бази даних є реаліційними, доступними, з гарною документацією, але PostgreSQL через свою активну спільноту, широкі можливості використання розширень, та відмінну підтримку складних SQL запитів вважається більш потужним і надійним вибором для багатьох проєктів [9].

PostgreSQL має деякі переваги порівняно з MySQL:

- строго дотримується стандартів SQL, що робить його більш передбачуваним у роботі з базами даних та переносимим між різними платформами;
- має більш потужний оптимізатор запитів, що дозволяє ефективно виконувати складні запити навіть у великих обсягах даних.
- має розширену підтримку транзакцій та механізми блокування, що робить його більш підходящим для великих, високонавантажених проєктів.
- має широкий вибір розширень та можливостей для розширення функціональності, що дозволяє пристосувати базу даних під конкретні потреби проєкту.
- активна спільнота та підтримка: PostgreSQL користується широкою популярністю у спільноті, що означає активний розвиток, швидке виправлення помилок та доступність різноманітних ресурсів для підтримки.

Хоча і MySQL є потужною базою даних, PostgreSQL зазвичай вважається більш гнучким вибором, особливо для великих або складних проєктів.

1.3 Опис бізнес-процесів

Для опису бізнес процесів програмного забезпечення використовується BPMN модель. В даному розділі будуть розглянуті основні процеси використання майбутнього вебзастосунку.

Опис послідовності створення та виконання запиту від користувача на реєстрацію у застосунку (рисунок 1.4):

- користувач заповнює обов’язкові поля для реєстрації;
- натискає на кнопку зареєструватись;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

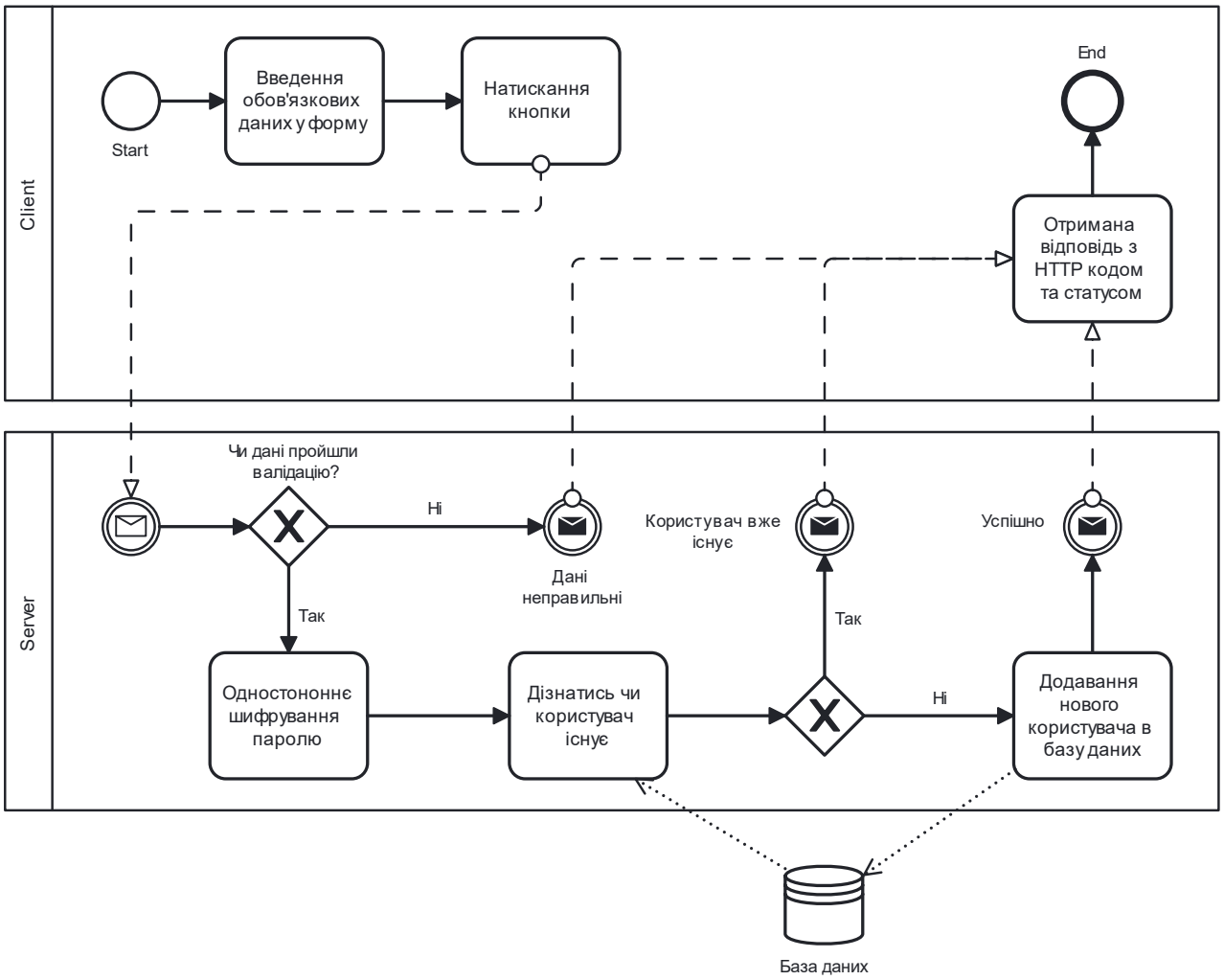


Рисунок 1.4 – Опис бізнес процесу реєстрації за допомогою моделі BPMN

Опис послідовності створення та виконання запиту від користувача на вхід у застосунок (рисунок 1.5):

- користувач заповнює обов’язкові поля для входу;
- натискає на кнопку увійти;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

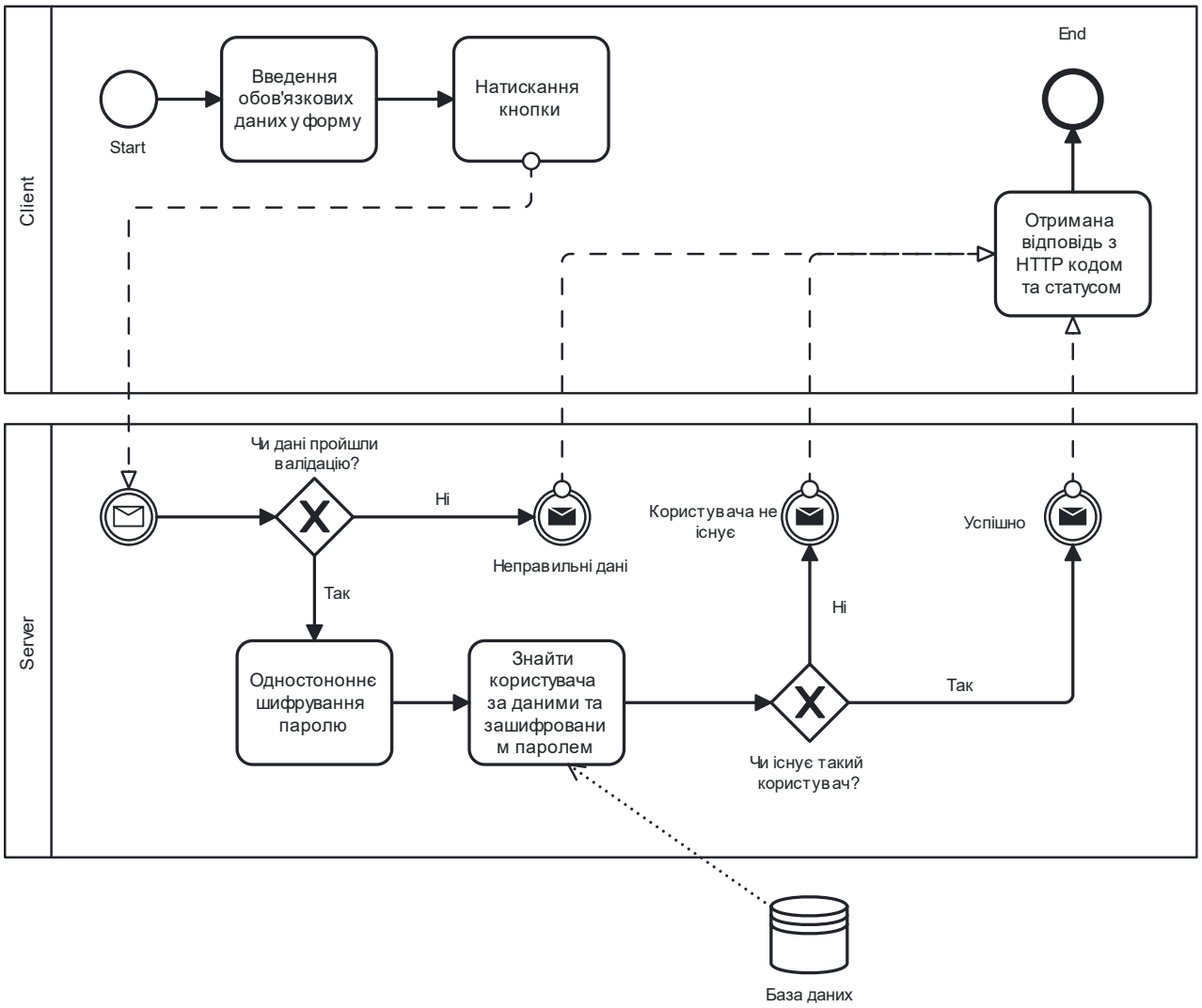


Рисунок 1.5 – Опис бізнес процесу входу за допомогою моделі BPMN

Опис послідовності створення та виконання запиту від користувача на отримання електронного щоденнику (рисунок 1.6):

- користувач відкриває сторінку електронного щоденника;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

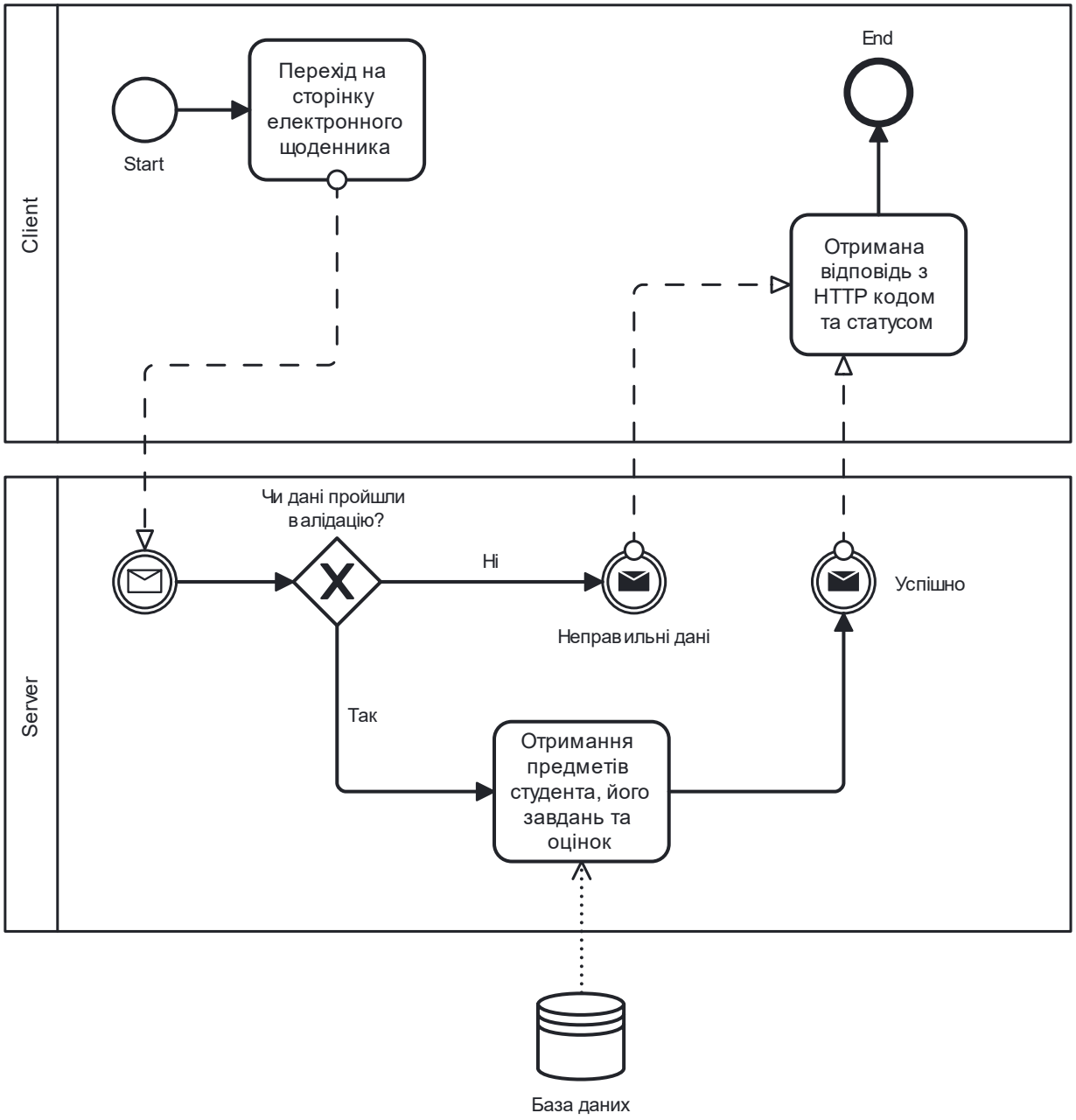


Рисунок 1.6 – Опис бізнес процесу отримання електронного щоденника за допомогою моделі BPMN

Опис послідовності створення та виконання запиту від користувача на отримання списку доступних чатів для спілкування (рисунок 1.7):

- користувач відкриває сторінку чатів;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;

- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

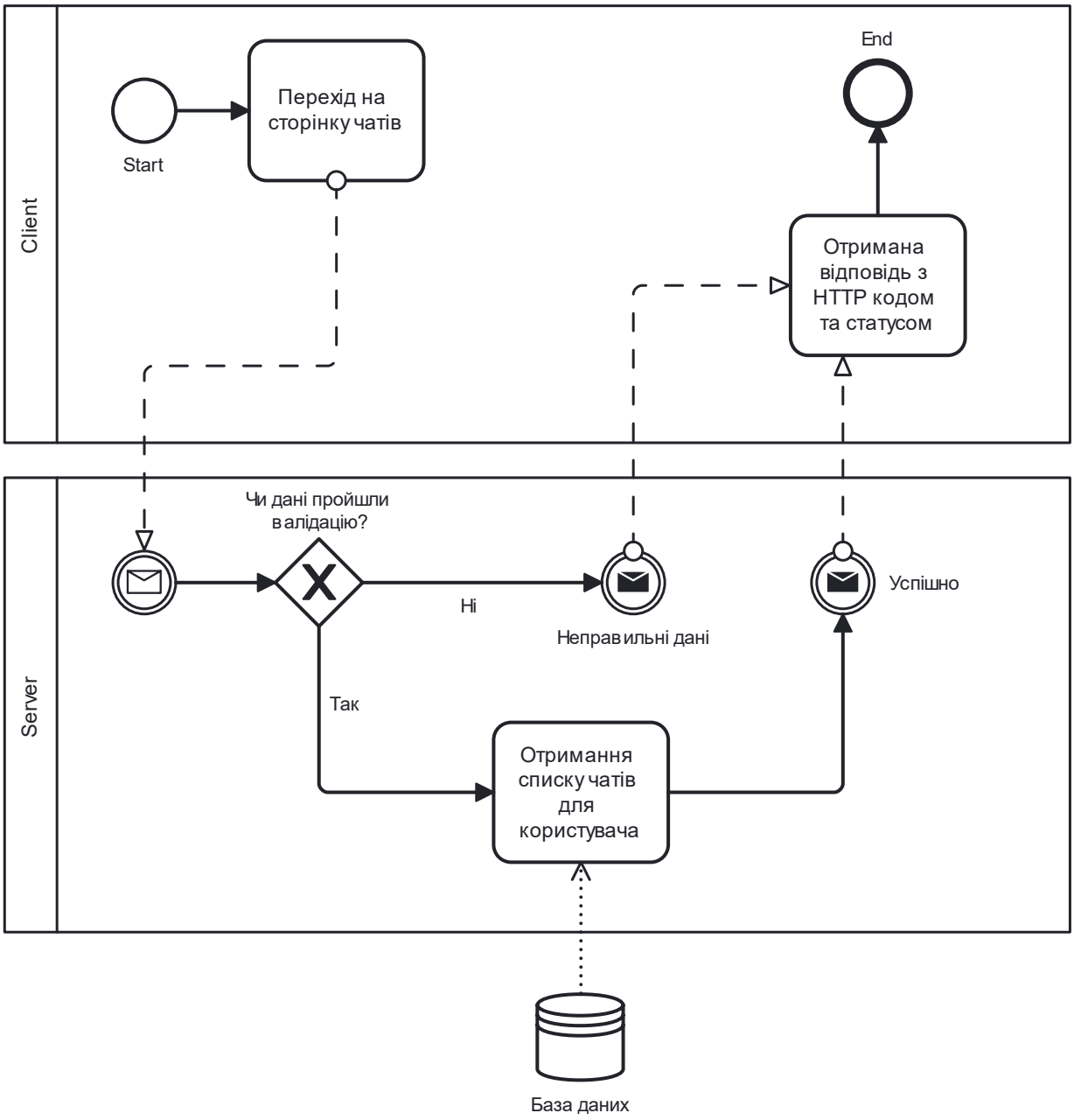


Рисунок 1.7 – Опис бізнес процесу отримання списку доступних чатів для спілкування за допомогою моделі BPMN

Опис послідовності створення та виконання запиту від користувача на отримання всіх повідомлень чату групи (рисунок 1.8):

- користувач натискає на чат з панелі вибору чатів;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію

клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;

- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

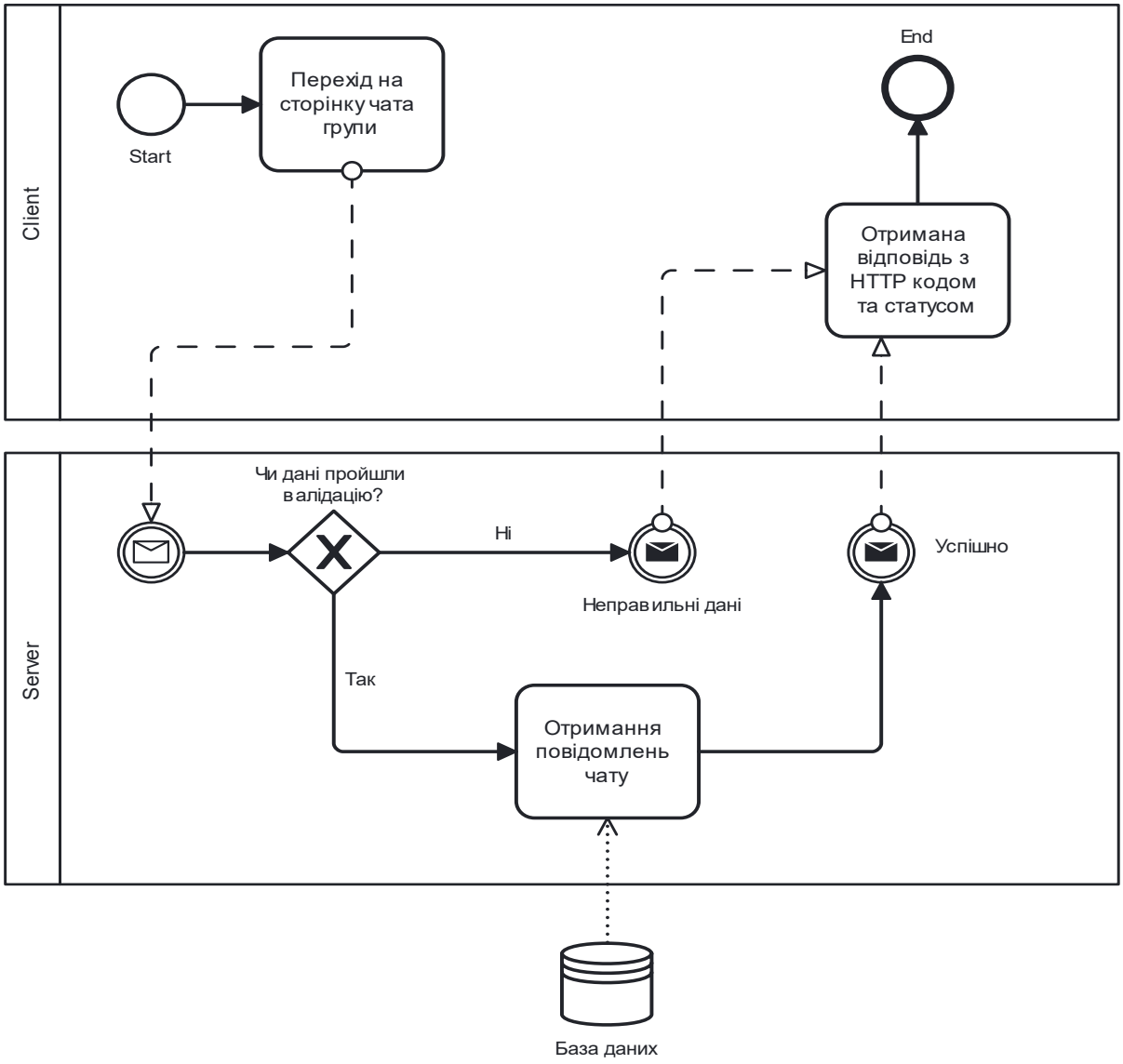


Рисунок 1.8 – Опис бізнес процесу отримання отримання всіх повідомлень чату групи за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на завантаження завдання викладача (рисунок 1.9):

- користувач натискає на назву файлу завдання;
- клієнт виконує запит на сервер;

- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

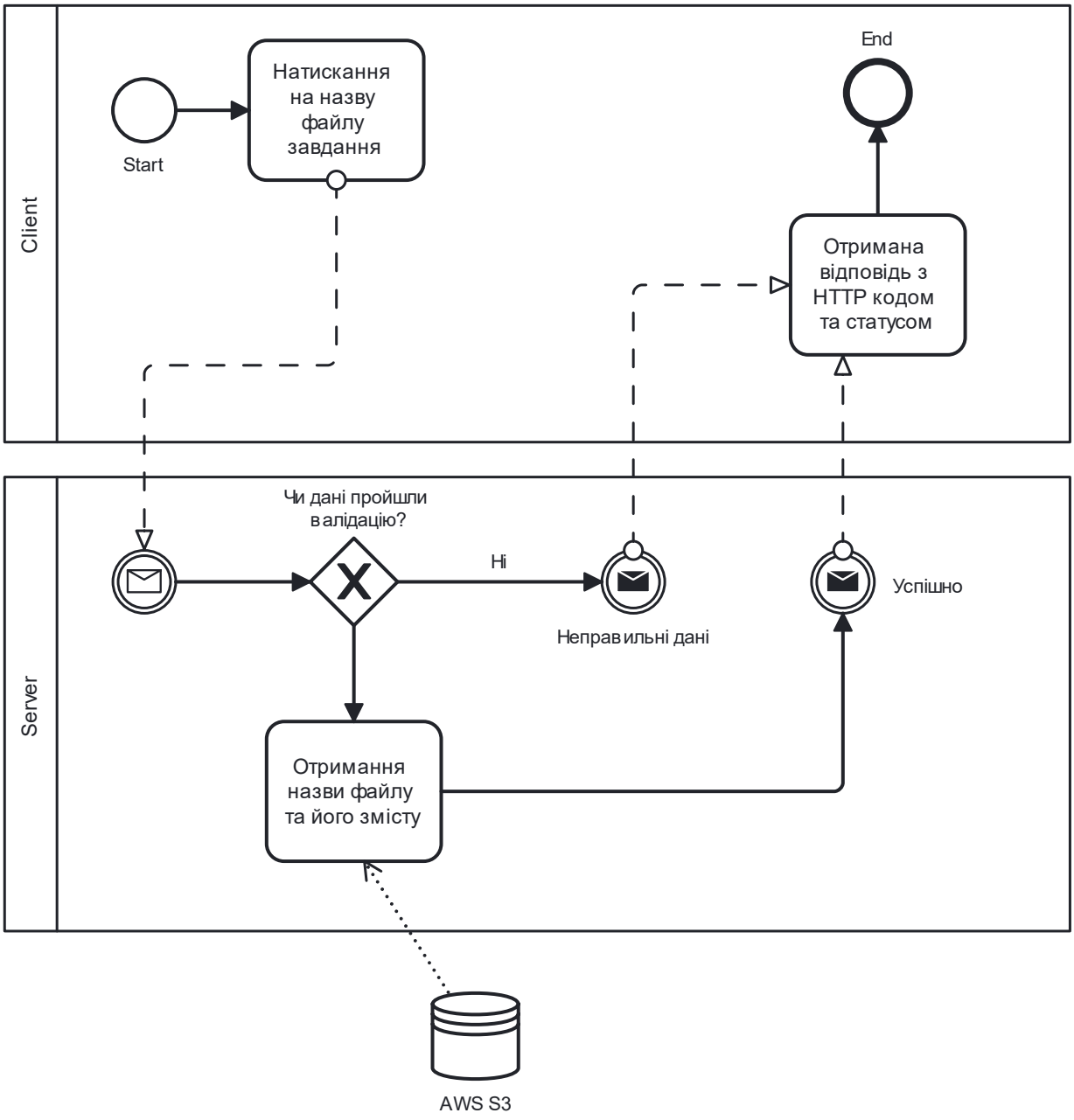


Рисунок 1.9 – Опис бізнес процесу завантаження файлу завдання за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на завантаження завдання викладача (рисунок 1.10):

- користувач натискає на назву файлу рішення завдання;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

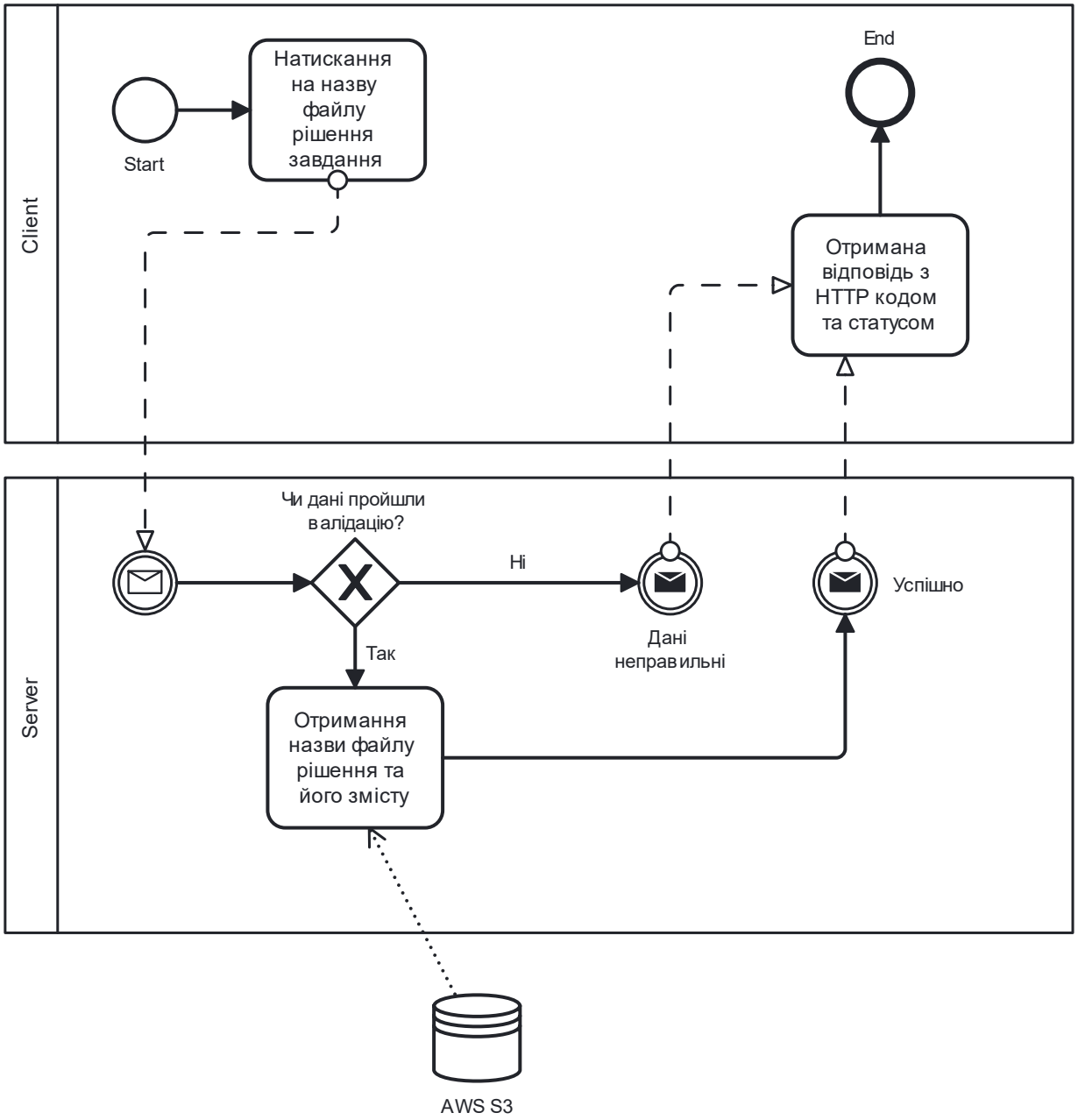


Рисунок 1.10 – Опис бізнес процесу завантаження файлу рішення завдання за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на завантаження рішення завдання студентом (рисунок 1.11):

- користувач друкує повідомлення та натискає на кнопку відправити (Enter);
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту,

в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;

- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

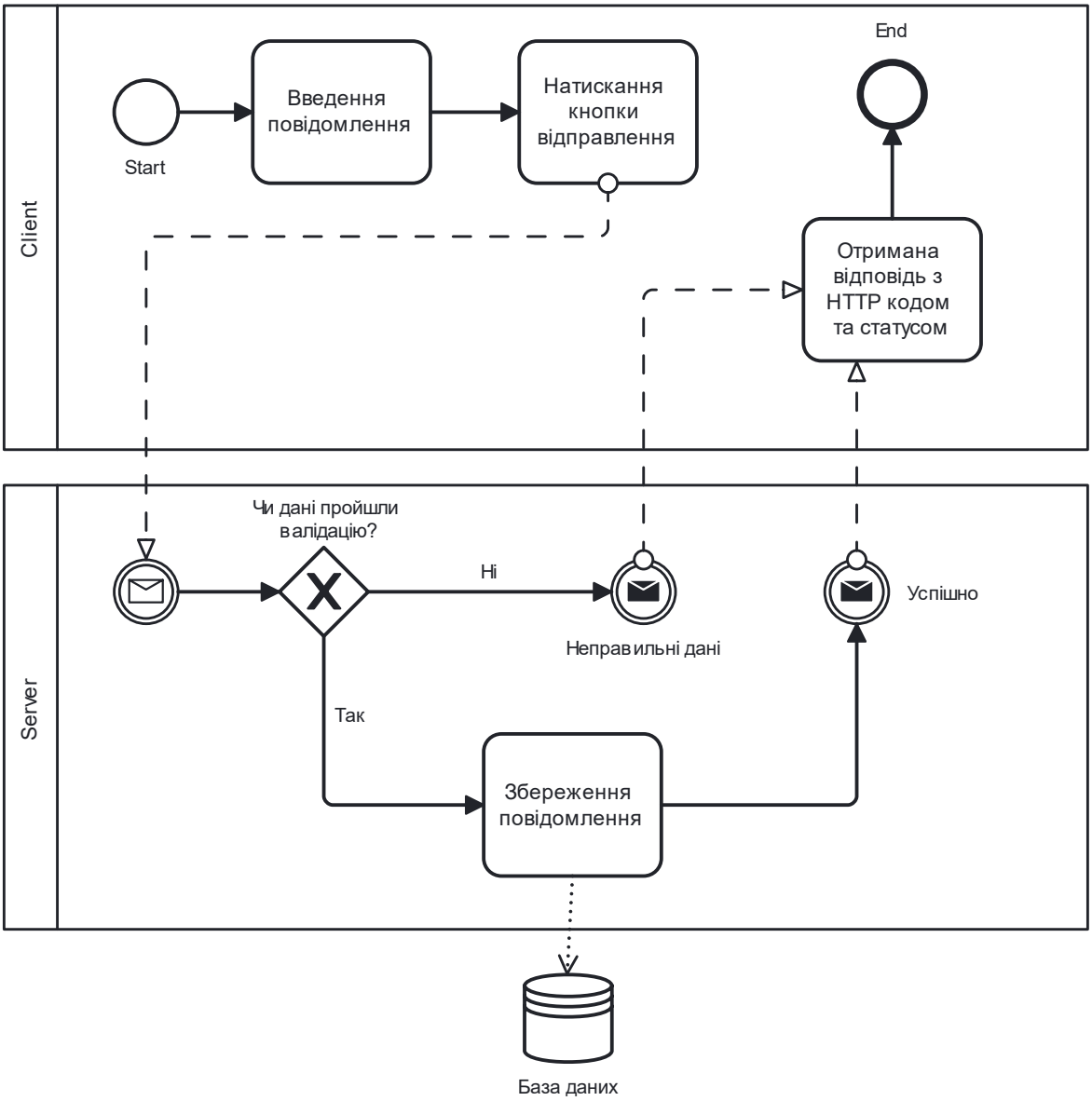


Рисунок 1.11 – Опис бізнес процесу відправлення повідомлення за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на отримання списку завдань викладача, рішень студента та оцінок (рисунок 1.12):

- студент заходить на сторінку завдань;
- клієнт виконує запит на сервер;

- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

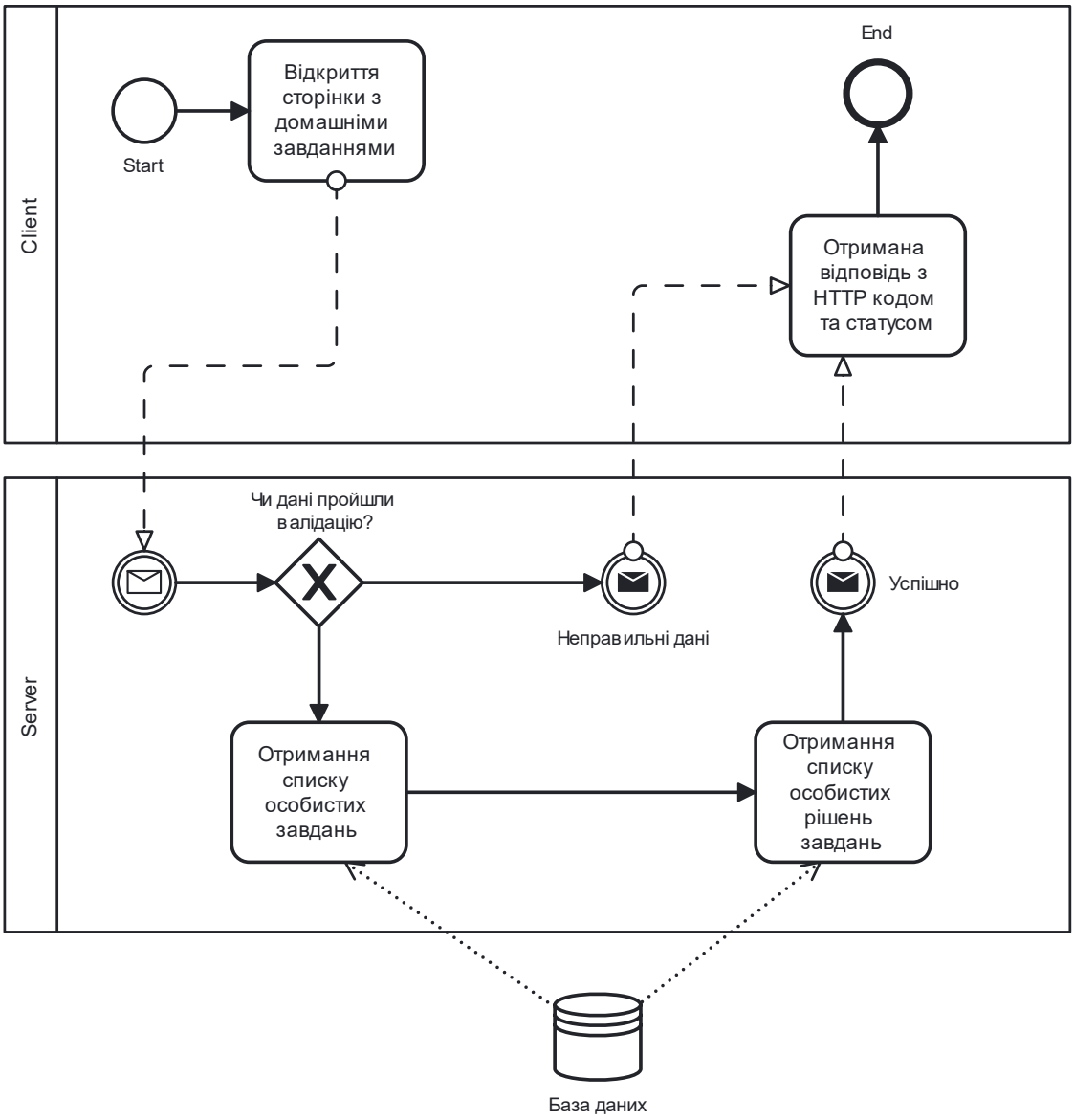


Рисунок 1.12 – Опис бізнес процесу отримання списку завдань викладача, рішень студента та оцінок за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на відправлення рішення завданя викладача (рисунок 1.13):

- студент заповнює форму рішення завдання;

- натискає на кнопку відправити;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

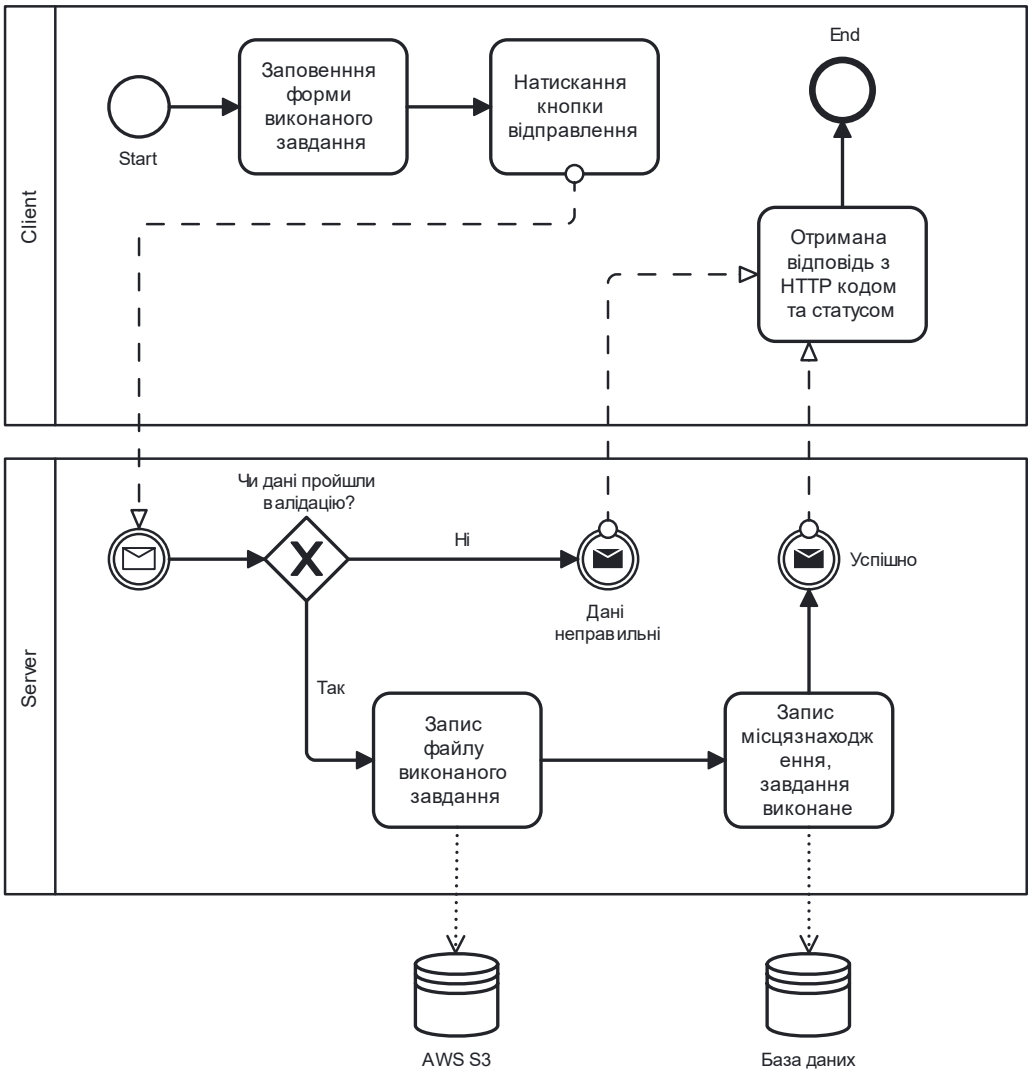


Рисунок 1.13 – Опис бізнес процесу відправлення рішення завдання викладача за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на отримання всіх завдань викладача та всіх рішень студентів до завдання (рисунок 1.14):

- викладач відкриває сторінку завдань;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

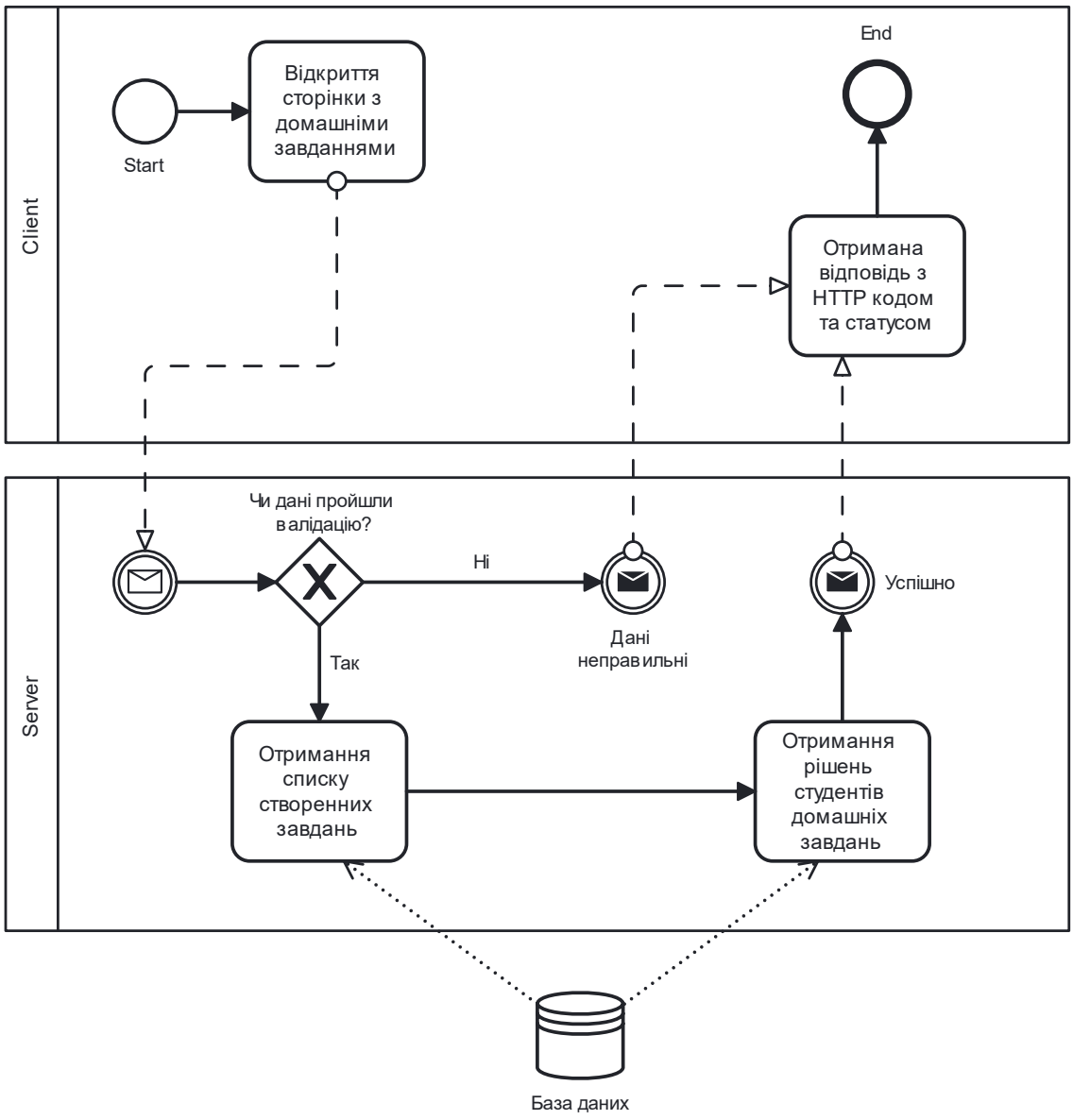


Рисунок 1.14 – Опис бізнес процесу отримання всіх завдань викладача та всіх рішень студентів до завдання за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на відправлення завдання викладачем (рисунок 1.15):

- викладач відкриває сторінку завдань;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

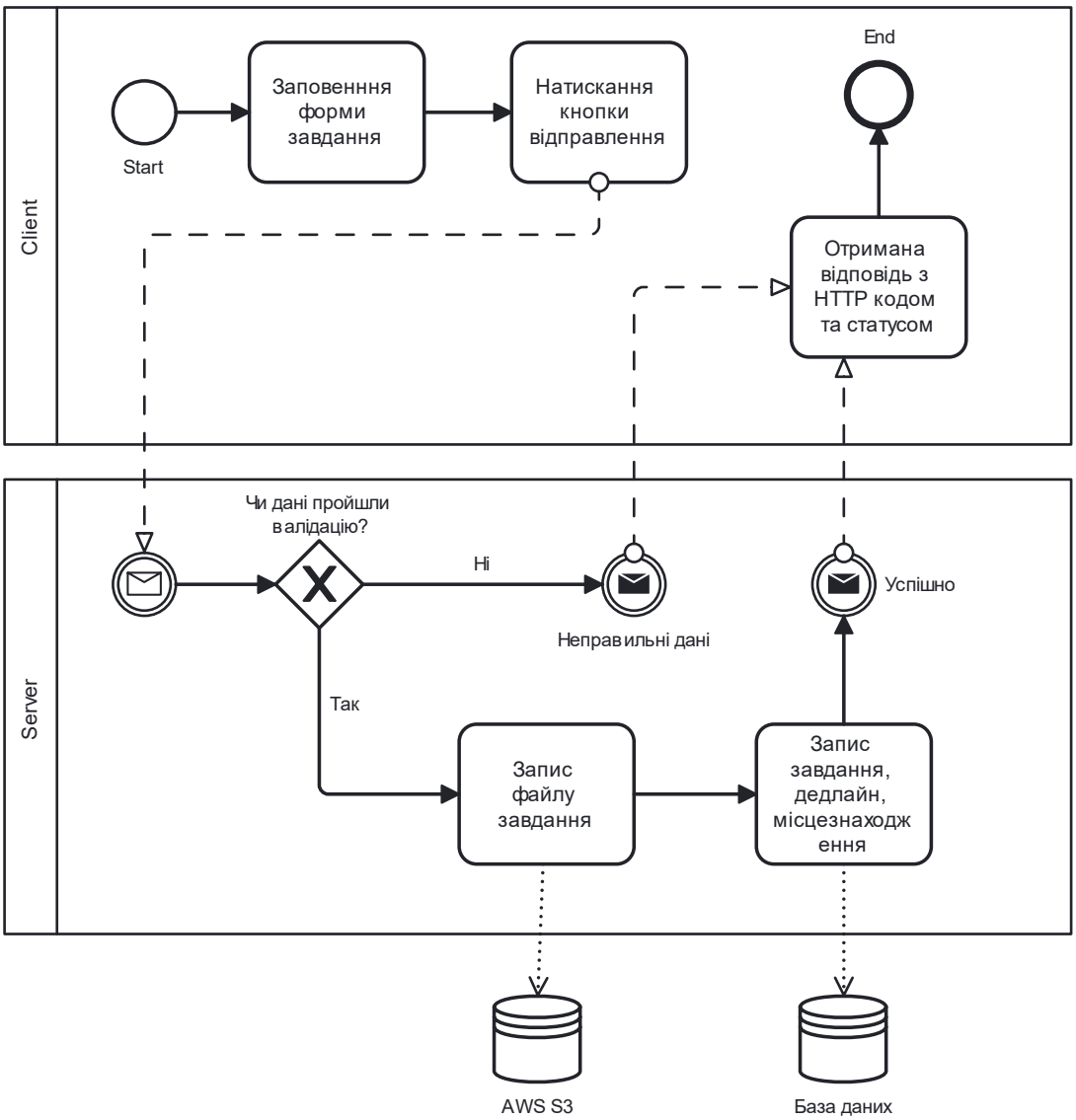


Рисунок 1.15 – Опис бізнес процесу відправлення завдання викладачем за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на відправлення оцінки за виконання завдання студента (рисунок 1.16):

- викладач відкриває сторінку завдань;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

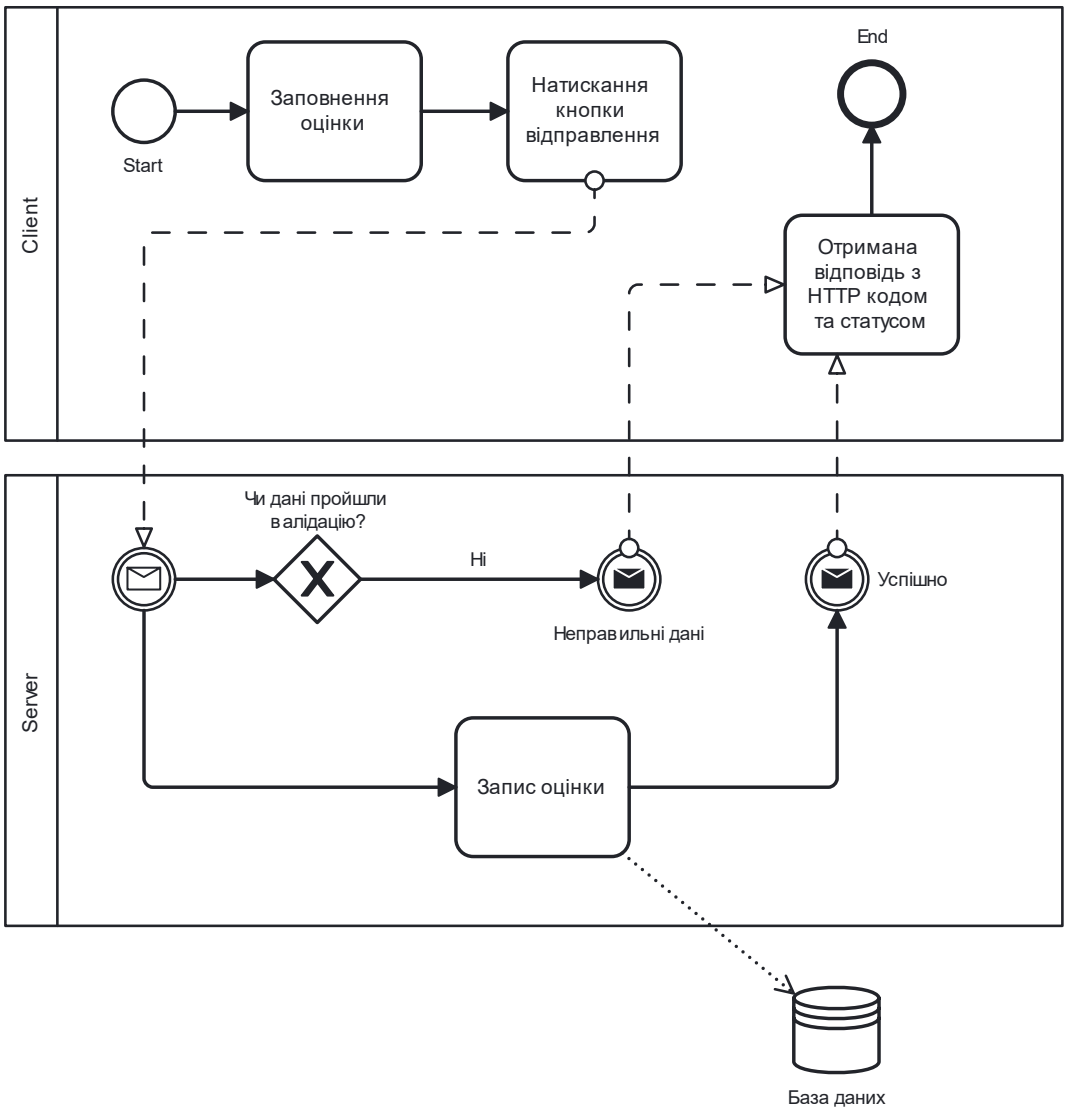


Рисунок 1.16 – Опис бізнес процесу відправлення оцінки за виконання завдання студентом за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на створення університету (рисунок 1.17):

- адміністратор заповнює форму;
- натискає на кнопку відправити;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

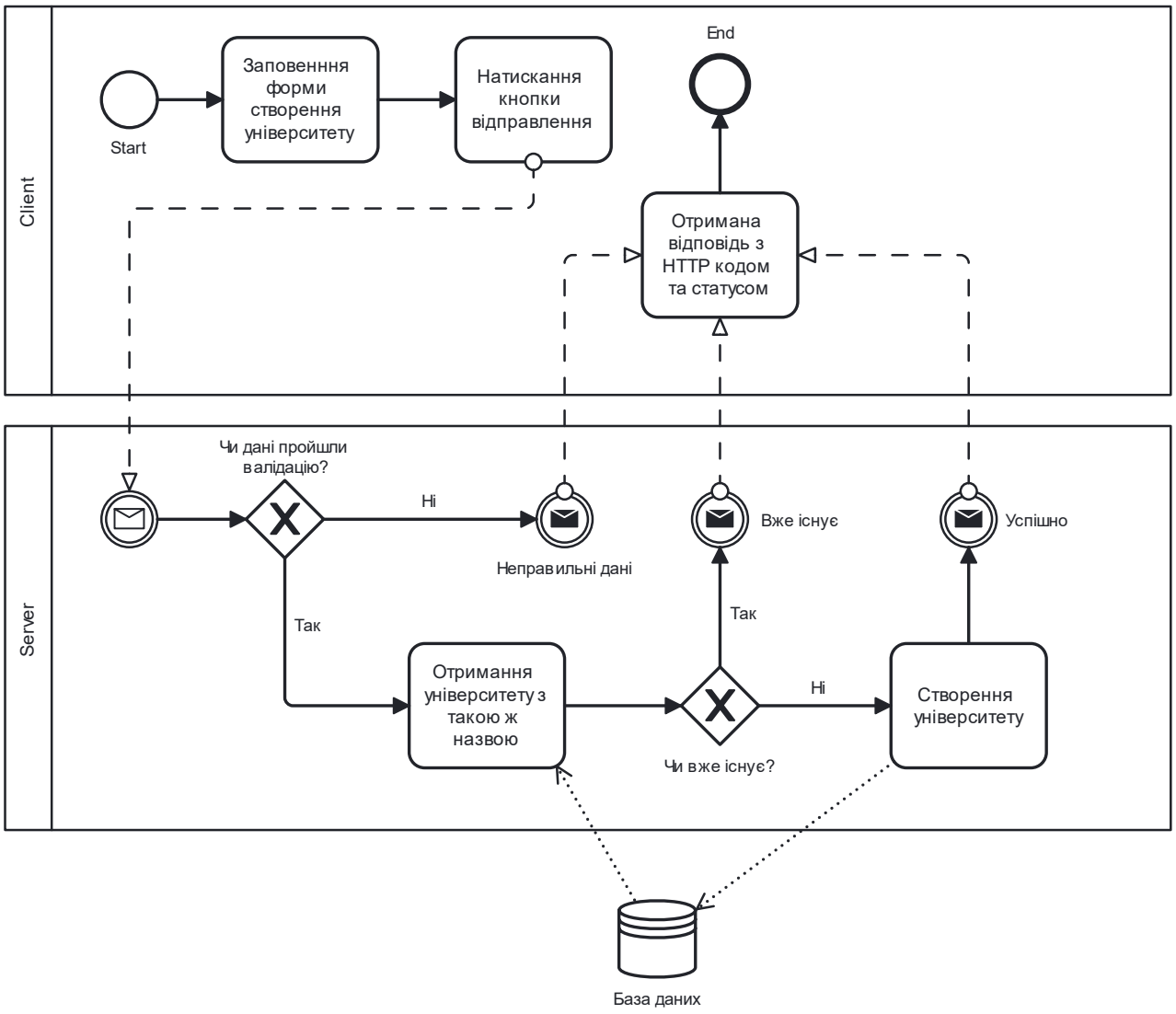


Рисунок 1.17 – Опис бізнес процесу створення університету за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на створення ступеня викладача (рисунок 1.18):

- адміністратор заповнює форму;
- натискає на кнопку відправити;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

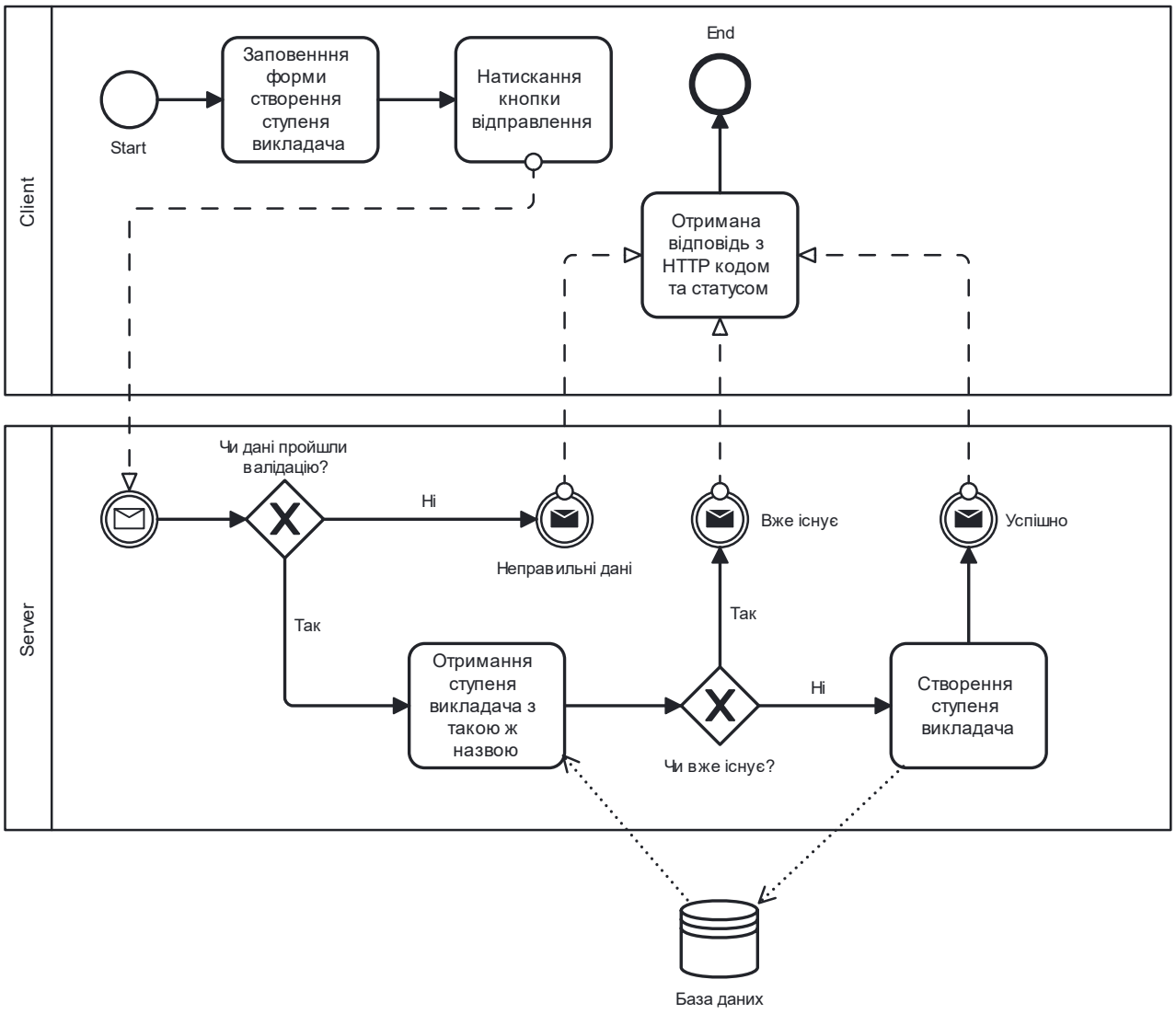


Рисунок 1.18 – Опис бізнес процесу створення ступеня викладача за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на створення предмету на кафедрі (рисунок 1.19):

- адміністратор заповнює форму;
- натискає на кнопку відправити;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

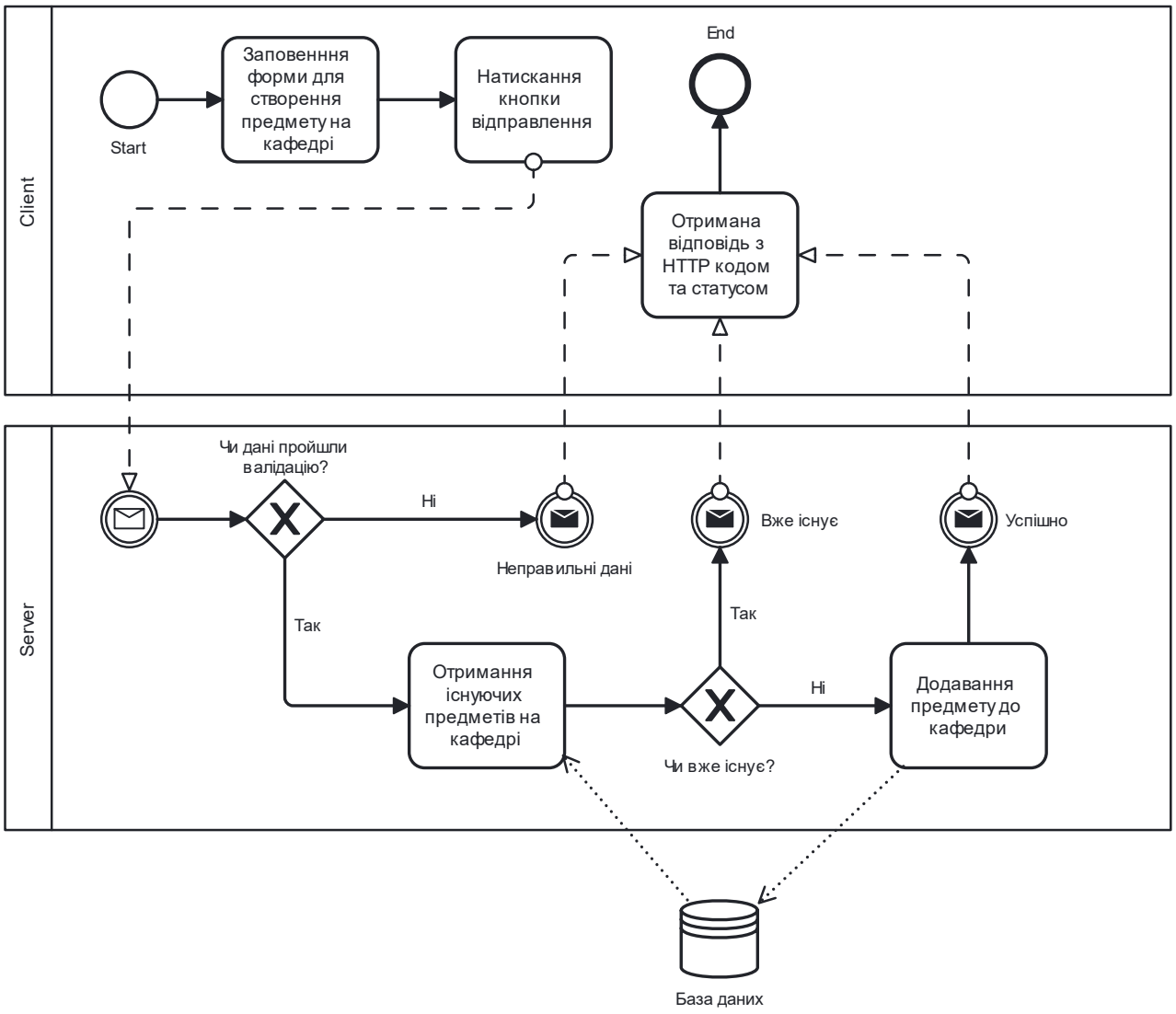


Рисунок 1.19 – Опис бізнес процесу створення предмету на кафедрі за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на створення групи для предмету, як групи на кафедрі, наприклад, група ІП-02 для предмету «Основи програмування» (рисунок 1.20):

- адміністратор заповнює форму;
- натискає на кнопку відправити;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;

- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

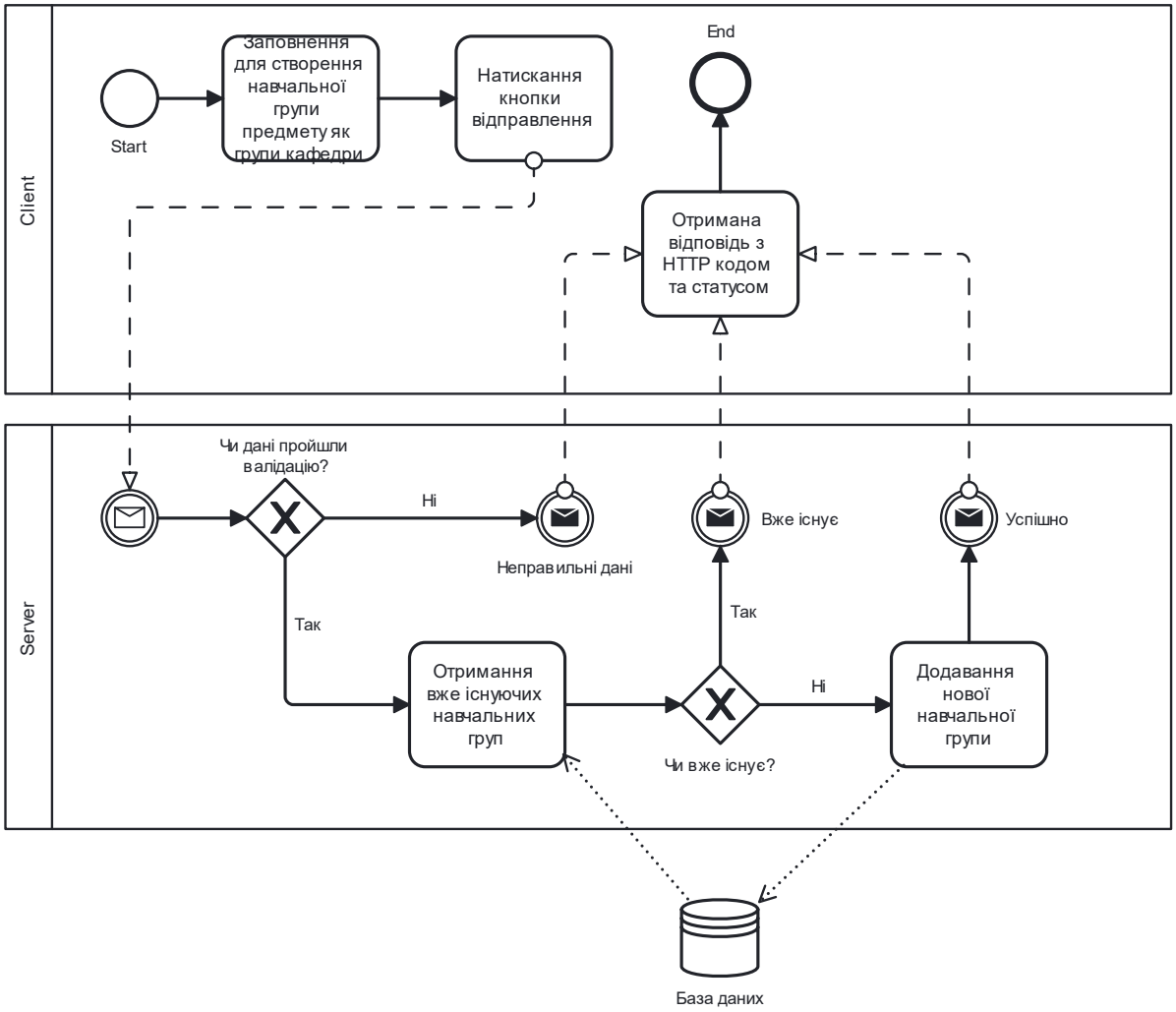


Рисунок 1.20 – Опис бізнес процесу створення групи для предмету, як групи на кафедрі за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на створення пустої групи для предмету на кафедрі (рисунок 1.21):

- адміністратор заповнює форму;
- натискає на кнопку відправити;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;

- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

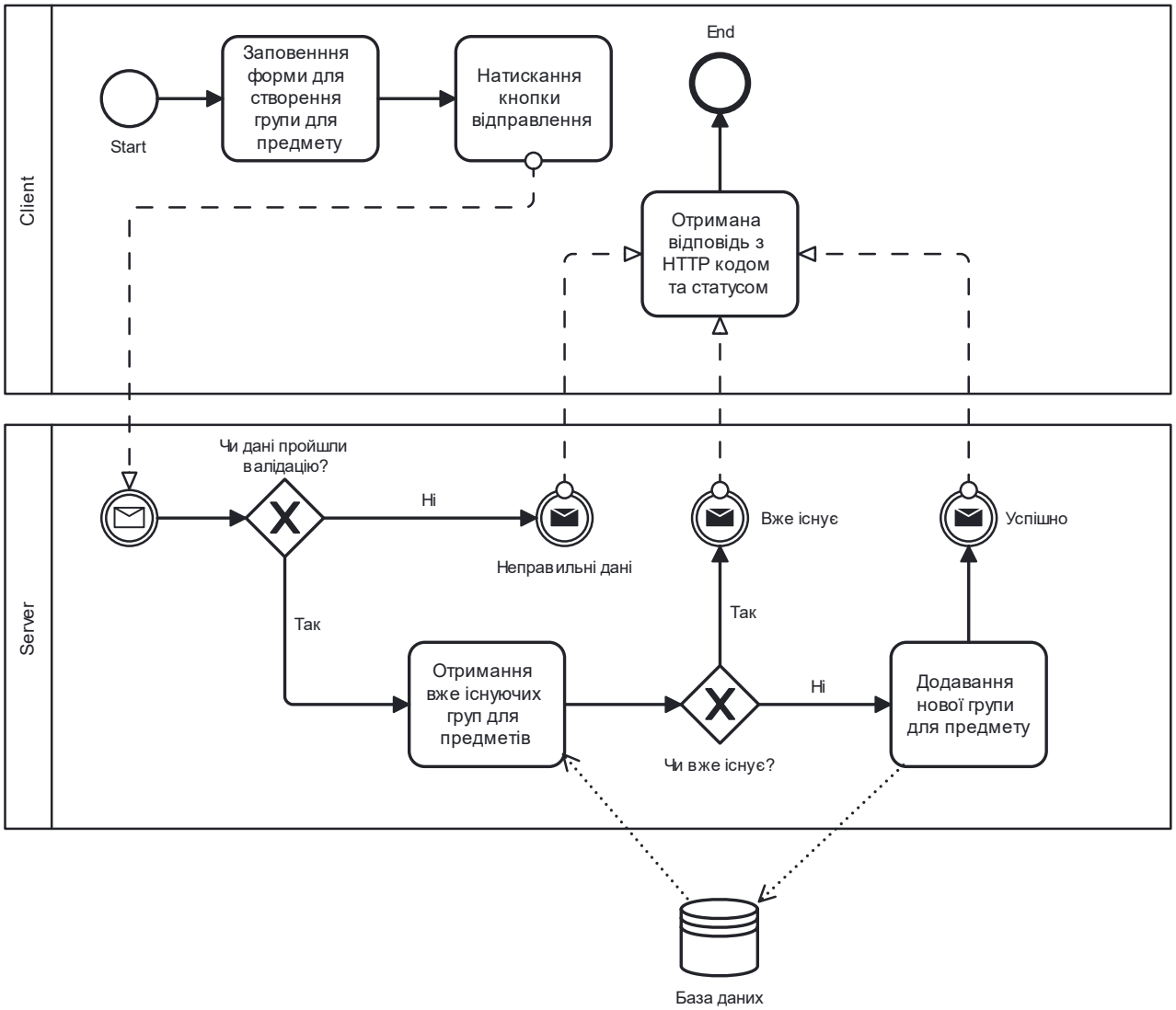


Рисунок 1.21 – Опис бізнес процесу створення пустої групи для предмету на кафедрі за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на створення факультету (рисунок 1.22):

- адміністратор заповнює форму;
- натискає на кнопку відправити;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту,

в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;

- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

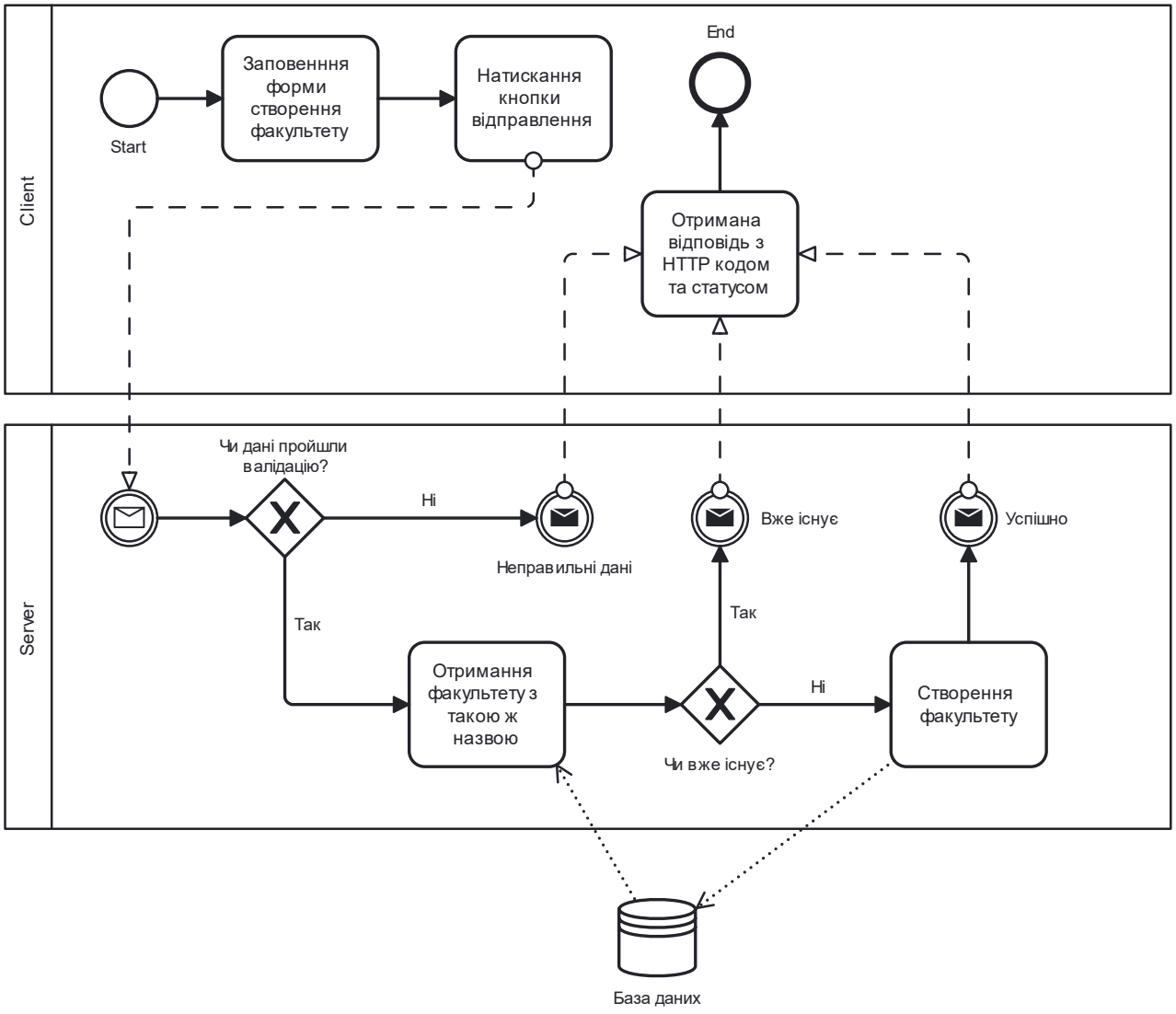


Рисунок 1.22 – Опис бізнес процесу створення факультету за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на створення групи на кафедрі, наприклад, ІП-02 (рисунок 1.23):

- адміністратор заповнює форму;
- натискає на кнопку відправити;
- клієнт виконує запит на сервер;

- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

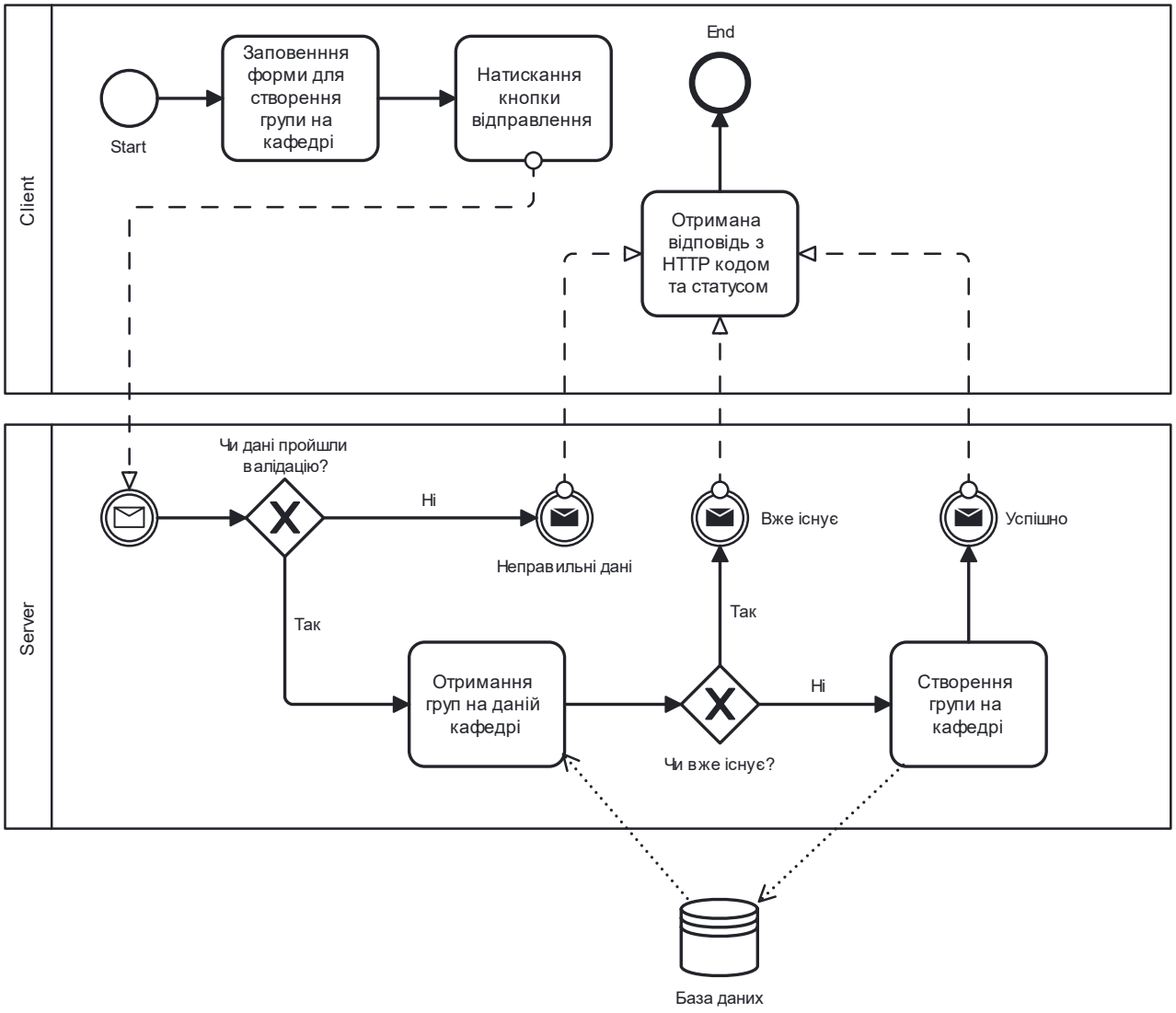


Рисунок 1.23 – Опис бізнес процесу на створення групи на кафедрі за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на створення кафедри (рисунок 1.24):

- адміністратор заповнює форму;
- натискає на кнопку відправити;

- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

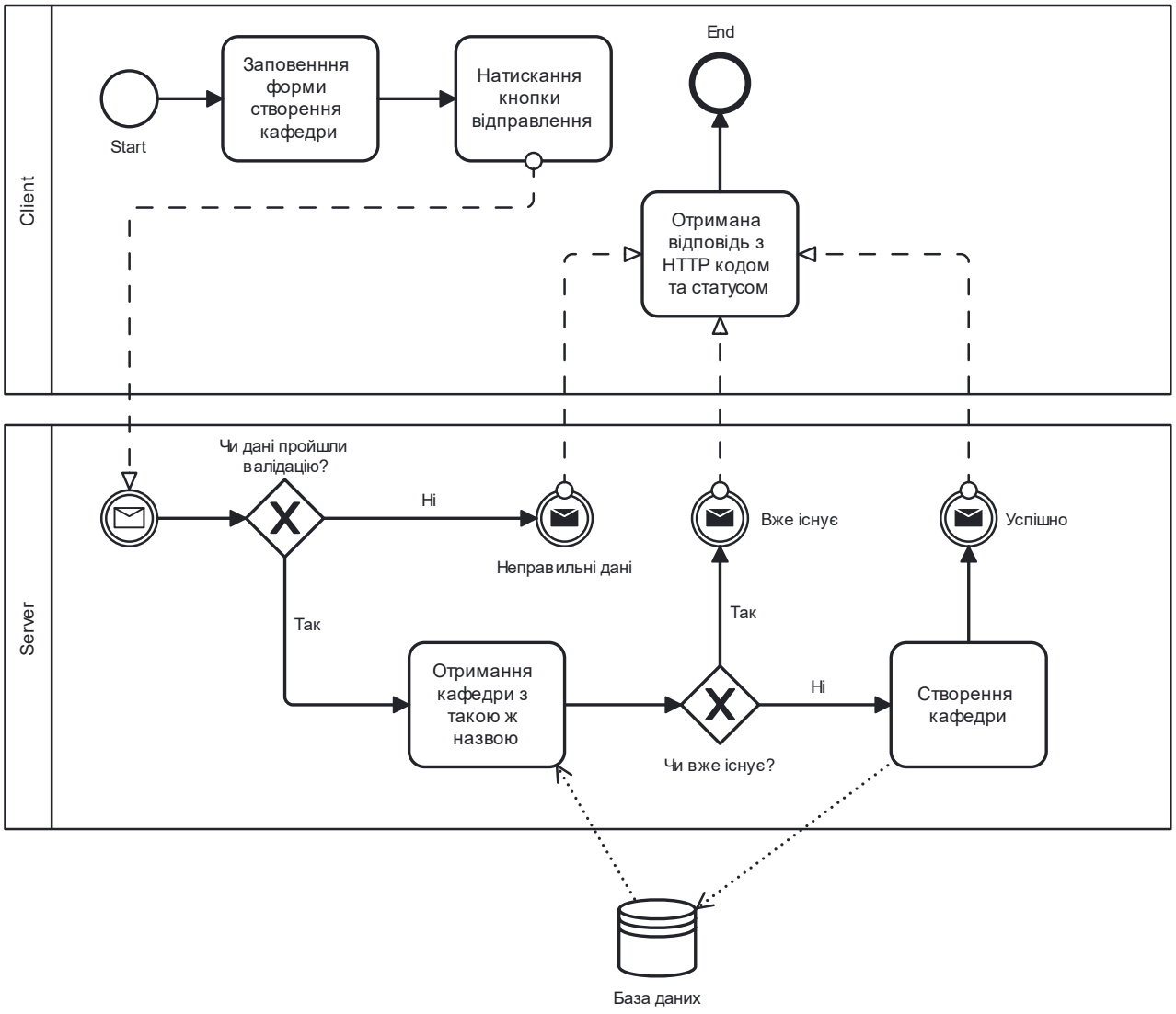


Рисунок 1.24 – Опис бізнес процесу на створення кафедри за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на видачу прав викладача користувачу (рисунок 1.25):

- адміністратор заповнює форму;

- натискає на кнопку відправити;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

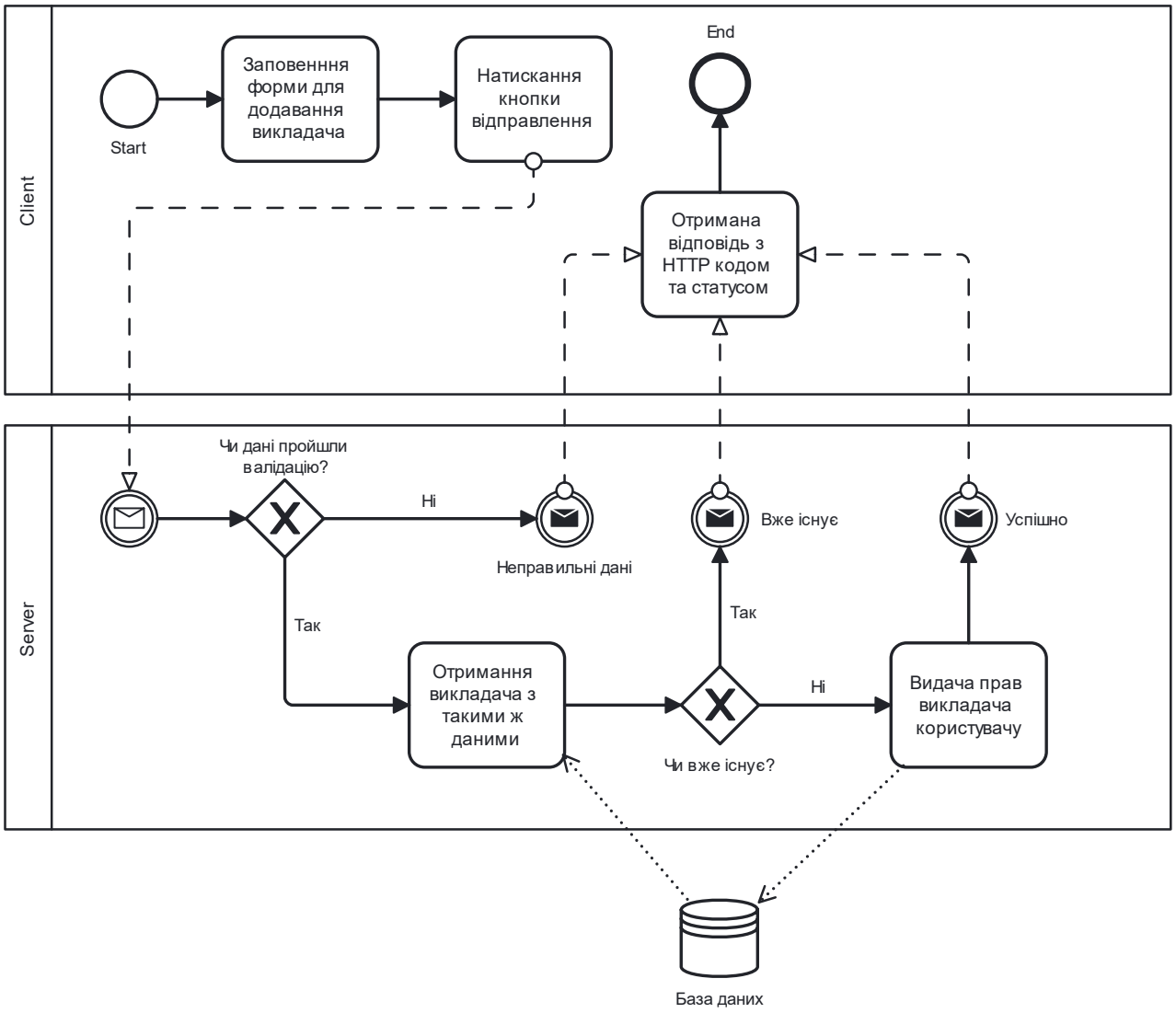


Рисунок 1.25 – Опис бізнес процесу на видачу прав викладача користувачу за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на ручне додавання студента до групи предмету (рисунок 1.26):

- адміністратор заповнює форму;
- натискає на кнопку відправити;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

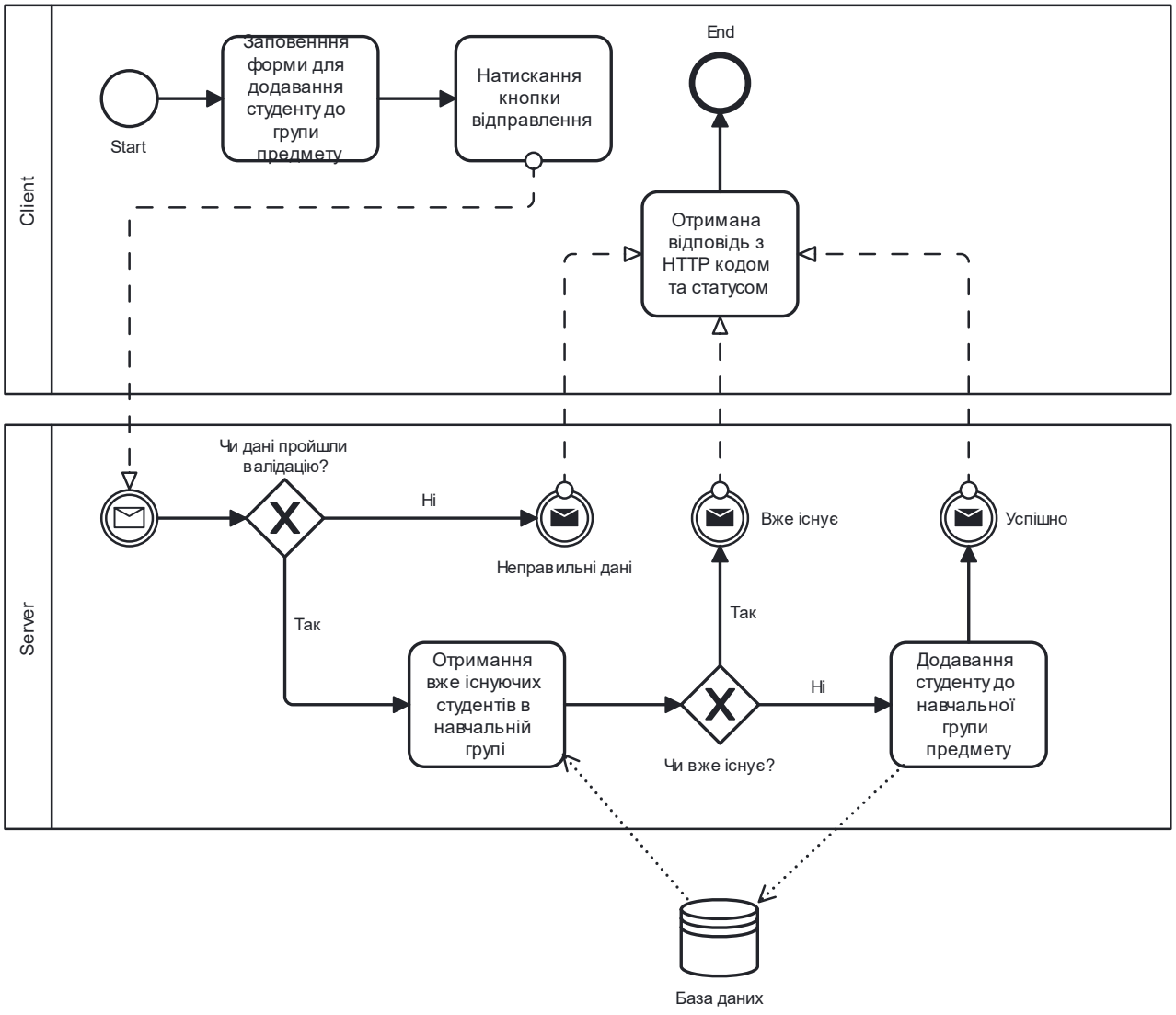


Рисунок 1.26 – Опис бізнес процесу на ручне додавання студента до групи предмету за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на ручне додавання студента до навчальної групи на кафедрі, наприклад, ІП-02 (рисунок 1.27):

- адміністратор заповнює форму;
- натискає на кнопку відправити;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

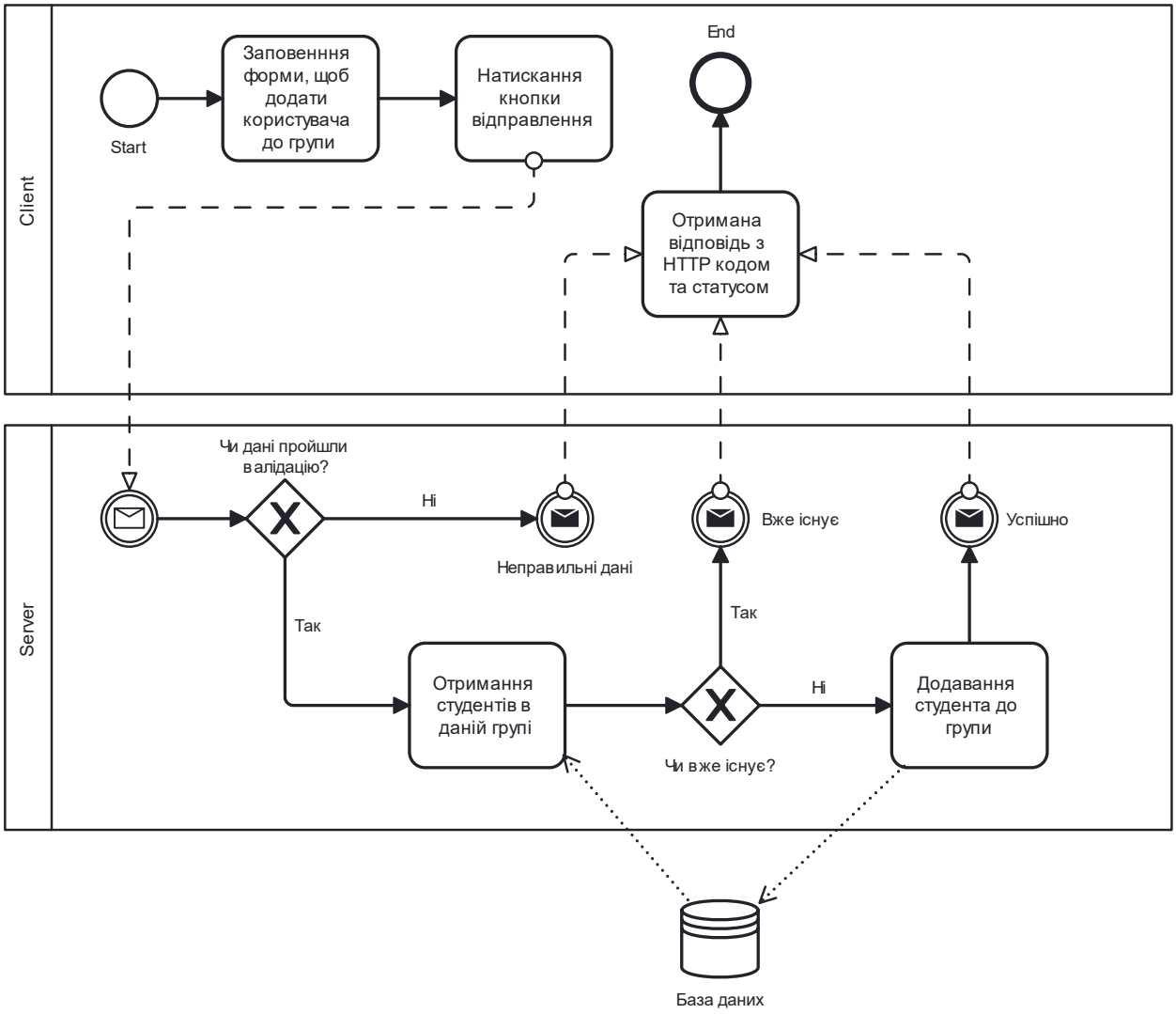


Рисунок 1.27 – Опис бізнес процесу на ручне додавання студента до навчальної групи на кафедрі за допомогою моделі BPMN

Опис послідовності створення та виконання запиту на додавання викладача до факультету (рисунок 1.28):

- адміністратор заповнює форму;
- натискає на кнопку відправити;
- клієнт виконує запит на сервер;
- якщо параметри вказані неправильно або виникне інша невідома помилка, то сервер віддасть відповідний код помилки та інформацію клієнту, в якій буде сказано, що саме пішло не так, а клієнт повідомить користувачу про помилку;
- якщо всі параметри вказані вірно, то клієнт отримає відповідь з успішним кодом 2xx та повідомить про це користувача.

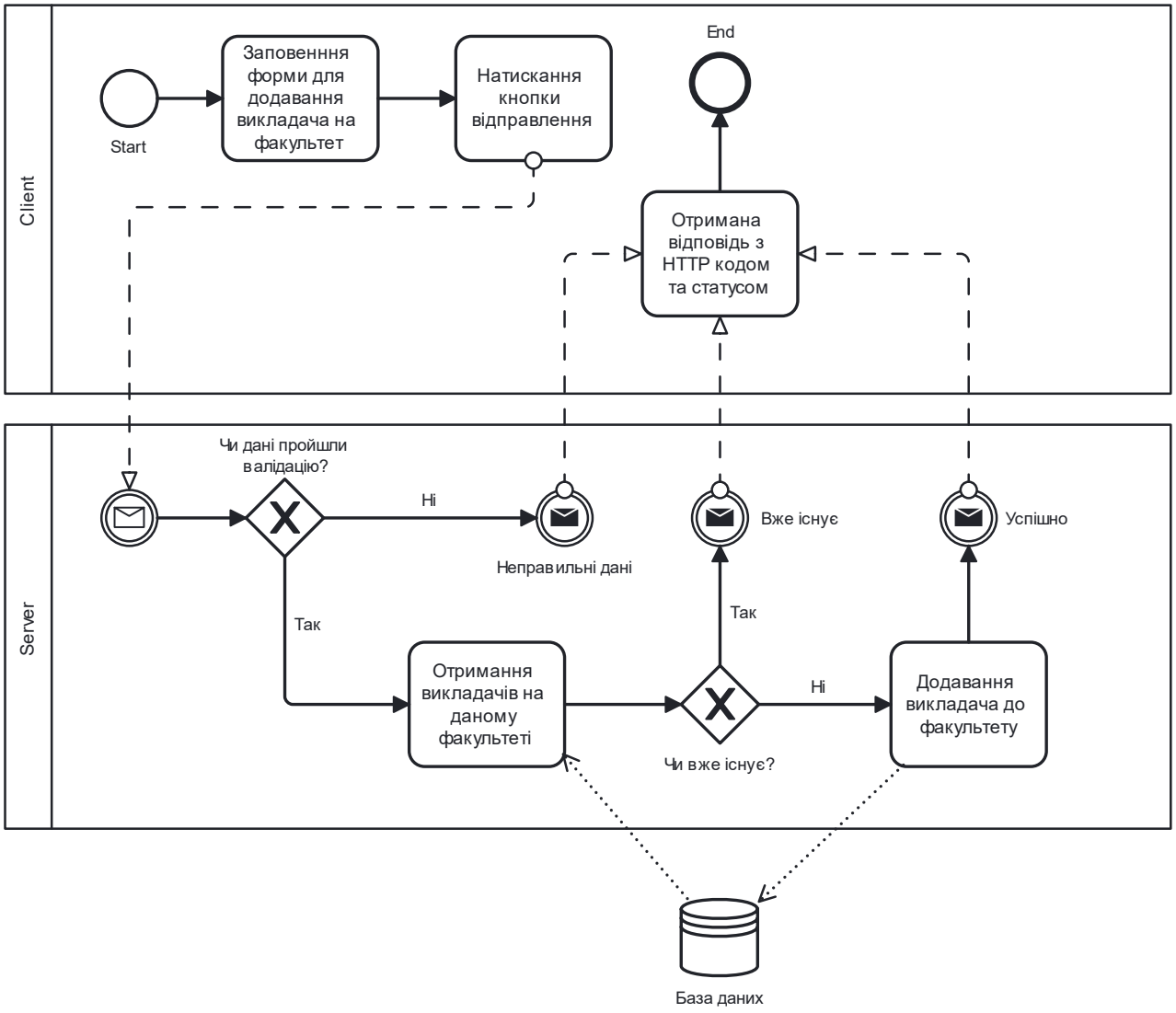


Рисунок 1.28 – Опис бізнес процесу на додавання викладача до факультету за допомогою моделі BPMN

1.4 Постановка задачі

Розробка даного вебзастосунку передбачає собою реалізацію платформи для дистанційного навчання, який полегшить спосіб дистанційного навчання через об’єднання всіх необхідних вимог щодо якісного навчання.

Головною метою є підвищення якості освіти та спрощення організаційних питань з приводу дистанційного навчання. Для досягнення поставленої мети потрібно реалізувати наступні задачі:

- спільний та захищений чат для всіх курсів, де викладачі та студенти можуть спілкуватися: це забезпечить більш ефективну взаємодію та обмін інформацією;
- єдина система завдань, де викладачі зможуть публікувати домашні завдання, а студенти матимуть можливість відправляти свої роботи: це полегшить процес організації та відстеження завдань;
- модуль для відстеження оцінок, який дозволить викладачам швидко оновлювати інформацію, а студентам легко переглядати свої оцінки.

Висновки до розділу

Під час виконання даного розділу було проведено аналіз предметної області, де було розглянуто всі основні поняття і терміни. Було проведено опитування серед студентів та визначено, що більшість студентів стикаються з очевидними організаційними питаннями через плутанину, що виникає через велику кількість різних застосунків для навчання, які кожен викладач обирає окремо для себе.

Був проведений аналіз відомих програмних продуктів, де було визначено, що немає жодного застосунку, що спеціалізується саме на навчанні у вищому начальному закладі, реалізує в собі всі необхідні для навчання речі: мессенджер, модуль відстеження оцінок та єдина система завдань, що інтегруються один з одним.

Також було виконано аналізування відомих алгоритмічних та технічних рішень, серед якого було обрано клієнт-серверну архітектуру програмного

забезпечення, спосіб реалізації методом вебзастосунку через його кросс-платформеність, методом комунікації клієнта та сервера за допомогою протоколу HTTP через його простоту, документованність та поширеність. Реляційну базу даних PostgreSQL через її популярність, масштабованість та строгого дотримування стандартизації мови SQL. Для розробки бекенду була обрана unix-подібна платформа Ubuntu через її популярність, документованність, простоту та підтримуваність.

Були описані всі необхідні бізнес-процеси за допомогою представлення у моделях BPMN, що спростить подальшу розробку та опис необхідних функціональних вимог.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головною функцією програмного забезпечення є надання користувачеві можливостей для полегшення організаційних питянь з приводу дистанційного навчання, більше функцій можна побачити на рисунку 2.1.

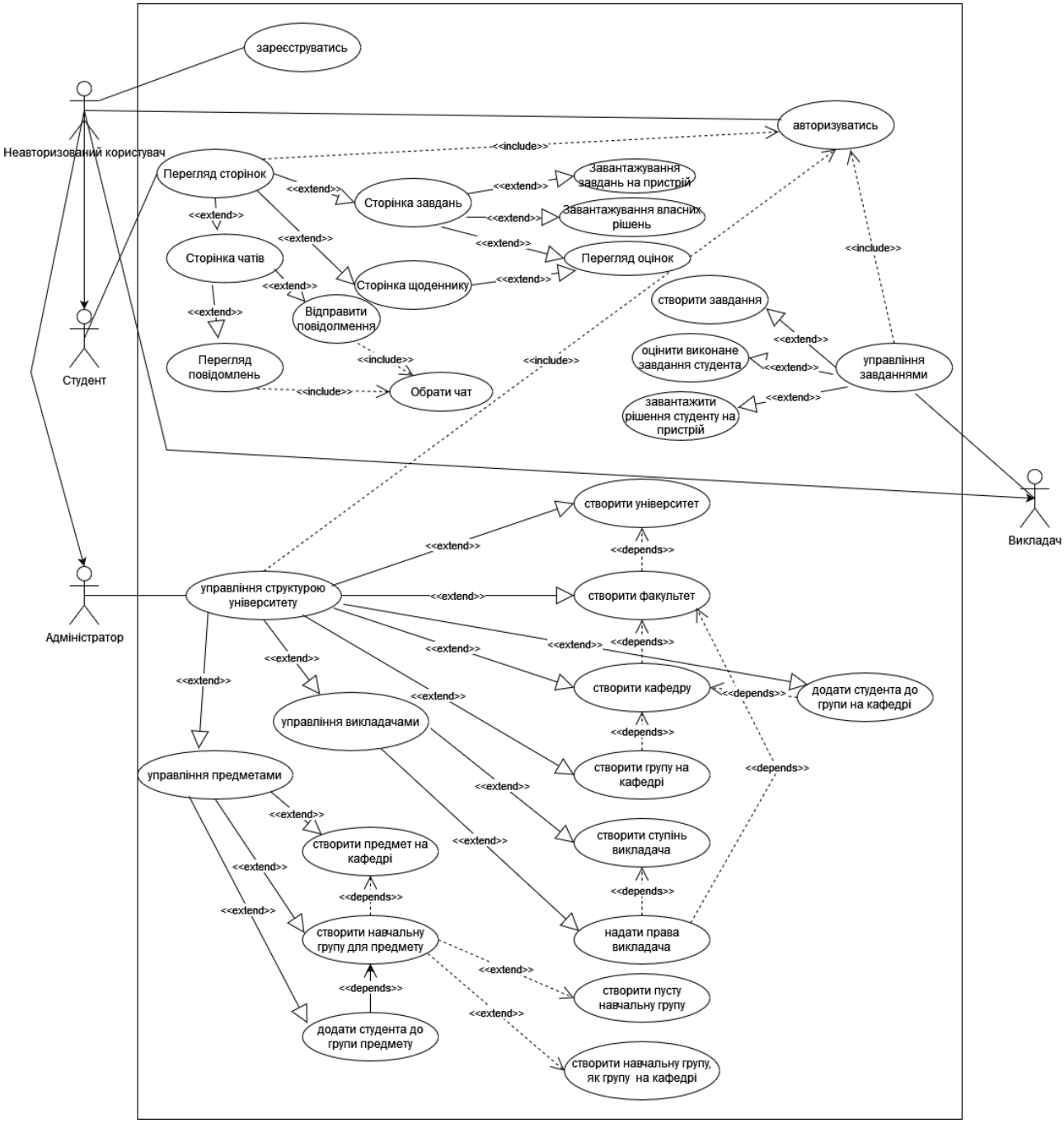


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.2 - 2.38 наведені варіанти використання програмного забезпечення.

Таблиця 2.2 - Варіант використання UC-01

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Реєстрація нового користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає зареєструватися
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: прізвище, ім’я, по-батькові, ім’я користувача, електронна пошта користувача, пароль в системі та дату народження. Після заповнення даних користувача натискає кнопку реєстрації. Після цього з’являється повідомлення про успішну реєстрацію, і користувач перенаправляється на сторінку входу.
Extension	В випадку введення не коректних даних користувач отримує помилку, що відображається під кнопкою зареєструватись.
Post-Condition	Створення користувача у базі даних, перехід на сторінку входу

Таблиця 2.3 - Варіант використання UC-02

Use case name	Реєстрація користувача вже існуючого користувача
Use case ID	UC-02
Goals	Отримання помилки про те, що користувач вже існує
Actors	Користувач
Trigger	Користувач бажає зареєструватися
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: прізвище, ім’я, по-батькові, ім’я користувача, електронна пошта користувача, пароль в системі та дату народження. Після заповнення даних

Продовження таблиці 2.3

	користувача натискає кнопку реєстрації. Після цього з’являється повідомлення про неуспішну реєстрацію.
Extension	В випадку введення не коректних даних користувач отримує помилку, що відображається під кнопкою зареєструватись.
Post-Condition	-

Таблиця 2.4 - Варіант використання UC-03

Use case name	Вхід існуючого користувача
Use case ID	UC-03
Goals	Вхід користувача в систему
Actors	Зареєстрований користувач
Trigger	Користувач бажає увійти у вебдодаток
Pre-conditions	Користувач зареєструвався
Flow of Events	Користувач переходить на сторінку входу. В поля для входу вводяться відповідні дані: ім’я користувача або електронна пошта та пароль. Після заповнення даних користувача натискає кнопку входу. Після цього з’являється повідомлення про успішний вхід, і користувач перенаправляється на сторінку чатів.
Extension	В випадку введення не коректних даних користувач отримує помилку, що відображається під кнопкою входу.
Post-Condition	Перехід на сторінку чатів

Таблиця 2.5 - Варіант використання UC-04

Use case name	Вхід не існуючого користувача
Use case ID	UC-04
Goals	Вхід неіснуючого користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає увійти у вебдодаток
Pre-conditions	Користувач не є зареєстрованним

Продовження таблиці 2.5

Flow of Events	Користувач переходить на сторінку входу. В поля для входу вводяться відповідні дані: ім’я користувача або електронна пошта та пароль. Після заповнення даних користувача натискає кнопку входу. Після цього з’являється повідомлення про неправильні дані для входу.
Extension	В випадку введення не коректних даних або не введення даних користувач отримує помилку, що відображається під кнопкою входу.
Post-Condition	-

Таблиця 2.6 - Варіант використання UC-05

Use case name	Створення університету
Use case ID	UC-05
Goals	Успішне створення університету
Actors	Адміністратор
Trigger	Адміністратор бажає створити університет
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводиться назва університету. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення над панеллю адміністратора про успішну дію.
Extension	-
Post-Condition	Створення запису у базі даних

Таблиця 2.7 - Варіант використання UC-06

Use case name	Створення вже існуючого університету
Use case ID	UC-06
Goals	Адміністратор отримає помилку про те, що університет з даною назвою вже існує
Actors	Адміністратор

Продовження таблиці 2.7

Trigger	Адміністратор бажає створити університет
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводиться назва університету. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення про те, що університет вже існує.
Extension	-
Post-Condition	-

Таблиця 2.8 - Варіант використання UC-07

Use case name	Створення факультету
Use case ID	UC-07
Goals	Успішне створення факультету
Actors	Адміністратор
Trigger	Адміністратор бажає створити факультет
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводиться назва факультету. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення над панеллю адміністратора про успішну дію.
Extension	-
Post-Condition	Створення запису у базі даних

Таблиця 2.9 - Варіант використання UC-08

Use case name	Створення вже існуючого факультету
Use case ID	UC-08
Goals	Адміністратор отримає помилку про те, що факультет з даною назвою вже існує

Продовження таблиці 2.9

Actors	Адміністратор
Trigger	Адміністратор бажає створити факультет
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводиться назва факультету та університету, для якої бажає створити факультет. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення про те, що факультет вже існує.
Extension	-
Post-Condition	-

Таблиця 2.10 - Варіант використання UC-09

Use case name	Створення кафедри
Use case ID	UC-09
Goals	Успішне створення кафедри
Actors	Адміністратор
Trigger	Адміністратор бажає створити кафедру
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводиться назва кафедри та факультету, для якої бажає створити кафедру. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення над панеллю адміністратора про успішну дію.
Extension	-
Post-Condition	Створення запису у базі даних

Таблиця 2.11 - Варіант використання UC-10

Use case name	Створення вже існуючої кафедри
Use case ID	UC-10
Goals	Адміністратор отримає помилку про те, що кафедра з даною назвою вже існує
Actors	Адміністратор
Trigger	Адміністратор бажає створити кафедру
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводиться назва кафедри та факультету, для якої бажає створити кафедру. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення про те, що кафедра вже існує.
Extension	-
Post-Condition	-

Таблиця 2.12 - Варіант використання UC-11

Use case name	Створення рівня викладача
Use case ID	UC-11
Goals	Успішне створення рівня викладача
Actors	Адміністратор
Trigger	Адміністратор бажає створити рівень викладача
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводиться назва рівню викладача. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення над панеллю адміністратора про успішну дію.
Extension	-
Post-Condition	Створення запису у базі даних

Таблиця 2.13 - Варіант використання UC-12

Use case name	Створення вже існуючого рівня викладача
Use case ID	UC-12
Goals	Адміністратор отримає помилку про те, що рівень викладача з даною назвою вже існує
Actors	Адміністратор
Trigger	Адміністратор бажає створити рівень викладача
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводиться назва рівню викладача. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення про те, що рівень викладача вже існує.
Extension	-
Post-Condition	-

Таблиця 2.14 - Варіант використання UC-13

Use case name	Видача користувачеві прав викладача
Use case ID	UC-13
Goals	Успішна видача користувачеві прав викладача
Actors	Адміністратор
Trigger	Адміністратор бажає надати права викладача
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться ім’я користувача або пошта та рівень викладача. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення над панеллю адміністратора про успішну дію.
Extension	-
Post-Condition	Створення запису у базі даних

Таблиця 2.15 - Варіант використання UC-14

Use case name	Видача викладачу прав викладача
Use case ID	UC-14
Goals	Адміністратор отримає помилку про те, що користувач вже є викладачем
Actors	Адміністратор
Trigger	Адміністратор бажає надати користувачеві права викладача
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться ім’я або пошта користувача та рівень. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення про те, що користувач вже є викладачем.
Extension	-
Post-Condition	-

Таблиця 2.16 - Варіант використання UC-15

Use case name	Додача викладача до факультету
Use case ID	UC-15
Goals	Успішна додача викладача до факультету
Actors	Адміністратор
Trigger	Адміністратор бажає додати викладача до факультету
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться ім’я користувача або пошта та назва факультету. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення над панеллю адміністратора про успішну дію.
Extension	-
Post-Condition	Створення запису у базі даних

Таблиця 2.17 - Варіант використання UC-16

Use case name	Додача вже існуючого викладача на факультеті
Use case ID	UC-16
Goals	Адміністратор отримає помилку про те, що викладач вже існує на даному факультеті
Actors	Адміністратор
Trigger	Адміністратор бажає додати викладача до факультету
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться ім’я або пошта користувача та назва факультету. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення про те, що викладач вже є викладачем на даному факультеті.
Extension	-
Post-Condition	-

Таблиця 2.18 - Варіант використання UC-17

Use case name	Створення групи на кафедрі
Use case ID	UC-17
Goals	Успішне створення групи на кафедрі
Actors	Адміністратор
Trigger	Адміністратор бажає створити групу на кафедрі
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться назва кафедри та назва групи. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення над панеллю адміністратора про успішну дію.
Extension	-
Post-Condition	Створення запису у базі даних

Таблиця 2.19 - Варіант використання UC-18

Use case name	Створення вже існуючої групи на кафедрі
Use case ID	UC-18
Goals	Адміністратор отримає помилку про те, що група вже існує на кафедрі
Actors	Адміністратор
Trigger	Адміністратор бажає створити групу на кафедрі
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться назва кафедри та назва групи. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення про те, що викладач вже є викладачем на даному факультеті.
Extension	-
Post-Condition	-

Таблиця 2.20 - Варіант використання UC-19

Use case name	Додача студента до групи на кафедрі
Use case ID	UC-19
Goals	Успішна додача студента до групи на кафедрі
Actors	Адміністратор
Trigger	Адміністратор бажає додати студента до групи на кафедрі
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться ім’я користувача або пошта та назва групи. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення над панеллю адміністратора про успішну дію.
Extension	Якщо групи не існує адміністратор отримає помилку про це.
Post-Condition	Створення запису у базі даних

Таблиця 2.21 - Варіант використання UC-20

Use case name	Додача студента до групи на кафедрі, в якій студент вже навчається
Use case ID	UC-20
Goals	Адміністратор отримає помилку про те, що студент вже навчається в групі
Actors	Адміністратор
Trigger	Адміністратор бажає додати студента до групи на кафедрі
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться ім’я користувача або пошта та назва групи. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення про те, що студент вже навчається у групі.
Extension	Якщо групи не існує адміністратор отримає помилку про це.
Post-Condition	-

Таблиця 2.22 - Варіант використання UC-21

Use case name	Створення предмету на кафедрі
Use case ID	UC-21
Goals	Успішне створення предмету на кафедрі
Actors	Адміністратор
Trigger	Адміністратор бажає створити предмет на кафедрі
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться ім’я користувача або пошта лектора, назва кафедри та назва предмету. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення над панеллю адміністратора про успішну дію.
Extension	Якщо лектора, кафедри або предмету не існує, то отримає відповідну помилку
Post-Condition	Створення запису у базі даних

Таблиця 2.23 - Варіант використання UC-22

Use case name	Створення вже існуючого на кафедрі предмету
Use case ID	UC-22
Goals	Адміністратор отримає помилку про те, що предмет на кафедрі вже існує
Actors	Адміністратор
Trigger	Адміністратор бажає створити предмет на кафедрі
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться ім’я користувача або пошта лектора, назва кафедри та предмету. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення про те, що предмет вже існує.
Extension	Якщо кафедри, лектора або предмету не існує, то отримає відповідну помилку
Post-Condition	-

Таблиця 2.24 - Варіант використання UC-23

Use case name	Створення навчальної групи предмету
Use case ID	UC-23
Goals	Успішне створення навчальної групи предмету на кафедрі
Actors	Адміністратор
Trigger	Адміністратор бажає створити навчальну групу предмету
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться ім’я користувача або пошта практика, назва предмету та назва групи предмету. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення про успішну дію.
Extension	Якщо практика, кафедри або предмету не існує або група вже існує, то отримає відповідну помилку
Post-Condition	Створення запису у базі даних

Таблиця 2.25 - Варіант використання UC-24

Use case name	Додача студента до групи предмету
Use case ID	UC-24
Goals	Успішна додача студента до групи предмету
Actors	Адміністратор
Trigger	Адміністратор бажає додати студента до групи предмету
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться ім’я користувача або пошта студента, назва гурпи предмету. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення про успішну дію
Extension	Якщо групи предмету або студента не існує, то отримає відповідну помилку
Post-Condition	Створення запису у базі даних

Таблиця 2.26 - Варіант використання UC-25

Use case name	Створення навчальної групи предмету як групу на кафедрі
Use case ID	UC-25
Goals	Успішне створення навчальної групи предмету на кафедрі
Actors	Адміністратор
Trigger	Адміністратор бажає створити навчальну групу предмету
Pre-conditions	Адміністратор увійшов у застосунок
Flow of Events	Адміністратор переходить на сторінку адміністратора. В поля для вводу вводяться ім’я користувача або пошта практика, назва предмету та назва групи на кафедрі. Після заповнення даних адміністратор натискає на кнопку виконати. Після цього з’являється повідомлення над панеллю адміністратора про успішну дію.
Extension	Якщо практика, предмету або групи на кафедрі не існує або група предмету вже існує, то отримає відповідну помилку
Post-Condition	Створення запису у базі даних

Таблиця 2.27 - Варіант використання UC-26

Use case name	Отримання списку чатів
Use case ID	UC-26
Goals	Успішне отримання чатів
Actors	Студент або викладач
Trigger	Коритсувач заходить на сторінку чатів
Pre-conditions	Користувач є авторизованим
Flow of Events	Користувач переходить на сторінку чатів. Після цього на екрані відображається список чатів.
Extension	-
Post-Condition	-

Таблиця 2.28 - Варіант використання UC-27

Use case name	Відкриття чату
Use case ID	UC-27
Goals	Успішне відкриття чату та отримання повідомлень
Actors	Студент або викладач
Trigger	Користувач бажає відкрити чат групи
Pre-conditions	Користувач увійшов у застосунок та відкрив сторінку чатів
Flow of Events	Користувач натискає на назву чату на відповідній сторінці після чого отримує список усіх повідомлень.
Extension	-
Post-Condition	-

Таблиця 2.29 - Варіант використання UC-28

Use case name	Відправлення повідомлення
Use case ID	UC-28
Goals	Успішне відправлення повідомлення
Actors	Студент або викладач
Trigger	Користувач бажає надіслати повідомлення у чат

Продовження таблиці 2.29

Pre-conditions	Користувач увійшов у застосунок, відкрив сторінку чатів та натиснув на бажаючий чат
Flow of Events	Користувач друкує повідомлення та натискає на кнопку надіслати (Enter). Повідомлення автоматично відправляється, сторінка оновлюється. На сторінці з’являється повідомлення, його відправник, дата та час відправлення.
Extension	-
Post-Condition	Створення запису у базі даних

Таблиця 2.30 - Варіант використання UC-29

Use case name	Отримання оцінок
Use case ID	UC-29
Goals	Успішне отримання оцінок
Actors	Студент
Trigger	Користувач бажає отримати список своїх оцінок
Pre-conditions	Користувач увійшов у застосунок, відкрив сторінку оцінок
Flow of Events	Користувач відкрив сторінку оцінок та на екрані відображені предмети, на кожен може натиснути, щоб згорнути або розгорнути та подивитися, за які завдання та скількі балів він отримав
Extension	-
Post-Condition	-

Таблиця 2.31 - Варіант використання UC-30

Use case name	Отримання завдань
Use case ID	UC-30
Goals	Успішне отримання завдань
Actors	Студент
Trigger	Користувач бажає отримати список своїх завдань
Pre-conditions	Користувач увійшов у застосунок, відкрив сторінку завдань

Продовження таблиці 2.31

Flow of Events	Користувач відкрив сторінку завдань та на екрані відображені предмети та групи, в яких він навчається, на кожен може натиснути, щоб згорнути або розгорнути та подивитися список особистих завдань: назва, дедлайн та можливість завантажити майбутнє рішення,
Extension	-
Post-Condition	-

Таблиця 2.32 - Варіант використання UC-31

Use case name	Отримання завдань
Use case ID	UC-31
Goals	Успішне отримання завдань
Actors	Викладач
Trigger	Викладач бажає отримати список своїх завдань та рішень студентів своїх груп
Pre-conditions	Користувач увійшов у застосунок, відкрив сторінку завдань
Flow of Events	Користувач відкрив сторінку завдань та на екрані відображені предмети та групи, в яких викладає, на кожен може натиснути, щоб згорнути або розгорнути та подивитися список особистих завдань: назва,
Extension	-
Post-Condition	-

Таблиця 2.33 - Варіант використання UC-32

Use case name	Відправлення завдання для групи предмету
Use case ID	UC-32
Goals	Успішне відправлення завдання
Actors	Викладач
Trigger	Користувач бажає відправити завдання групі предмету
Pre-conditions	Користувач увійшов у застосунок, відкрив сторінку завдань

Продовження таблиці 2.33

Flow of Events	Користувач відкрив сторінку завдань та натискає на предмет та групу, для якої бажає завантажити завдання, заповнює форму: назва завдання, обирає файл, виставляє дедлайн, натискає на кнопку відправити
Extension	Дедлайн має бути майбутньою датою
Post-Condition	Створюється запис в базі даних, файл відправляється до окремого сервісу

Таблиця 2.34 - Варіант використання UC-33

Use case name	Відправлення рішення завдання викладачеві
Use case ID	UC-33
Goals	Успішне відправлення рішення завдання
Actors	Студент
Trigger	Студент бажає відправити рішення до завдання
Pre-conditions	Користувач увійшов у застосунок, відкрив сторінку завдань
Flow of Events	Користувач відкрив сторінку завдань обрав предмет, обрав завдання та заповнює форму для рішення: додає файл рішення та натискає на кнопку відправити, після цього його рішення є прикріпленим до завдання.
Extension	При не заповненні форми отримає помилку, форму заповнити можна лише до дедлайну, інакше не буде можливості завантажити рішення.
Post-Condition	-

Таблиця 2.35 - Варіант використання UC-34

Use case name	Отримання рішень студентів
Use case ID	UC-34
Goals	Успішне отримання рішень студентів
Actors	Викладач
Trigger	Користувач бажає отримати рішення студентів
Pre-conditions	Користувач увійшов у застосунок, відкрив сторінку завдань

Продовження таблиці 2.35

Flow of Events	Користувач відкрив сторінку завдань та натискає на предмет та групу, в якій викладає, відкриває завдання, на екрані з’являються рішення
Extension	-
Post-Condition	-

Таблиця 2.36 - Варіант використання UC-35

Use case name	Виставлення оцінки за рішення студента
Use case ID	UC-35
Goals	Успішне виставлення оцінки
Actors	Викладач
Trigger	Користувач бажає оцінити рішення до власного завдання
Pre-conditions	Користувач увійшов у застосунок, відкрив сторінку завдань, обрав предмет та групу, обрав завдання та отримав список рішень
Flow of Events	Користувач переглянув виконання завдання, заповнив поле оцінки та натиснув на кнопку відправити. На екрані поле стало недоступним та зникла кнопка відправити.
Extension	-
Post-Condition	Створюється запис в базі даних

Таблиця 2.37 - Варіант використання UC-36

Use case name	Завантаження файлу завдання
Use case ID	UC-36
Goals	Успішне завантаження файлу завдання
Actors	Викладач або студент
Trigger	Користувач завантажити завдання
Pre-conditions	Користувач увійшов у застосунок, відкрив сторінку завдань, натиснув на бажане завдання
Flow of Events	Користувач натискає на назву файлу та розпочинається автоматичне завантажування файлу
Extension	-

Продовження таблиці 2.36

Post-Condition	-

Таблиця 2.38 - Варіант використання UC-37

Use case name	Завантаження файлу рішення завдання
Use case ID	UC-37
Goals	Успішне завантаження рішення завдання
Actors	Викладач або студент
Trigger	Користувач завантажити завдання
Pre-conditions	Користувач увійшов у застосунок, відкрив сторінку завдань, натиснув на бажане завдання та на рішення
Flow of Events	Користувач натискає на назву файлу та розпочинається автоматичне завантажування файлу
Extension	-
Post-Condition	-

2.2 Розроблення функціональних вимог

Вебзастосунок розділений на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.39 наведено загальну модель вимог, а в таблицях 2.40 – 2.47 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.3.

Таблиця 2.39 – Загальна модель вимог

№	Назва	ID вимоги	Пріоритети	Ризики
1	Аутентифікація користувача	FR-1	Високий	Низький
1.1	Реєстрація користувача	FR-2	Високий	Низький
1.2	Авторизація користувача	FR-3	Високий	Низький

Продовження таблиці 2.39

№	Назва	ID вимоги	Пріоритети	Ризики
2	Спільний та захищений чат між викладачами та студентами	FR-4	Високий	Середній
2.1	Відображення списку чатів	FR-5	Середній	Низький
2.1.1	Відкриття бажаного чату	FR-6	Високий	Низький
2.1.2	Перегляд існуючих повідомлень	FR-7	Високий	Середній
2.1.3	Відправлення повідомлення	FR-8	Високий	Низький
3	Єдина система завдань	FR-9	Високий	Високий
3.1	Створення завдання викладачем	FR-10	Високий	Високий
3.1.1	Завантаження існуючого завдання на пристрій	FR-11	Високий	Високий
3.2	Прикріплення рішення до завдання	FR-12	Високий	Середній
3.2.1	Завантаження існуючого рішення завдання на пристрій	FR-13	Високий	Середній
3.2.2	Оцінення виконаного завдання	FR-14	Середній	Низький
3.2.3	Перегляд оцінки за виконане завдання	FR-15	Високий	Низький
4	Модуль відстеження оцінок	FR-16	Низький	Низький
4.1	Перегляд списку завдань та оцінок	FR-17	Низький	Низький
4.2	Перегляд суми оцінок з кожного предмету окремо	FR-18	Низький	Низький

Продовження таблиці 2.39

№	Назва	ID вимоги	Пріоритети	Ризики
5	Панель адміністратора	FR-19	Високий	Низький
5.1	Створення університету	FR-20	Високий	Низький
5.2	Створення факультету	FR-21	Високий	Низький
5.3	Створення кафедри	FR-22	Високий	Низький
5.4	Створення рівня викладача	FR-23	Високий	Низький
5.5	Надання прав викладача користувачеві	FR-24	Високий	Низький
5.6	Додавання викладача до факультету	FR-25	Високий	Низький
5.7	Створення групи на кафедрі	FR-26	Високий	Низький
5.8	Додавання студента до групи на кафедрі	FR-27	Високий	Низький
5.9	Створення предмету на кафедрі	FR-28	Високий	Низький
5.10	Створення групи для предмету	FR-29	Високий	Низький
5.11	Додавання студента до групи предмету	FR-30	Середній	Низький
5.12	Створення групи для предмету як групи на кафедрі	FR-31	Високий	Низький

Таблиця 2.40 – Перелік функціональних вимог

ID вимоги	Назва та опис
FR-1	Аутентифікація користувача. Користувач має змогу зараєструватись або увійти

Продовження таблиці 2.40

ID вимоги	Назва та опис
FR-2	Реєстрація користувача. Неаутентифікований користувач може зареєструватися, відкривши необхідну сторінку або перейти на сторінку входу.
FR-3	Авторизація користувача. Неавторизований користувач може увійти, відкривши необхідну сторінку або перейти на сторінку реєстрації.
FR-4	Спільний та захищений чат між викладачами та студентами. Аутентифікований користувач має можливість зайти на сторінку чатів.
FR-5	Відображення списку чатів. Аутентифікований користувач на сторінці чатів має список доступних йому чатів.
FR-6	Відкриття бажаного чату. Аутентифікований користувач має можливість на сторінці чатів обрати бажаний чат.
FR-7	Перегляд існуючих повідомлень. Аутентифікований користувач має можливість переглядати існуючі повідомлення у обраному ним чаті.
FR-8	Відправлення повідомлення. Аутентифікований користувач має можливість надсилати повідомлення у обраному чаті.
FR-9	Єдина система завдань. Аутентифікований користувач має можливість відкрити сторінку завдань.
FR-10	Створення завдання викладачем. Викладач має можливість створити нове завдання для своєї групи, для якої викладає предмет.

Продовження таблиці 2.40

ID вимоги	Назва та опис
FR-11	Завантаження існуючого завдання на пристрій. Студент або викладач мають можливість завантажити своє завдання на пристрій.
FR-12	Прикріплення рішення до завдання. Студент має можливість прикріпити рішення до завдання викладача до дедлайну.
FR-13	Завантаження існуючого рішення завдання на пристрій. Студент або викладач мають можливість завантажити рішення завдання на пристрій. Для студента – особисте рішення, для викладача – рішення студенту, до свого завдання.
FR-14	Оцінення виконаного завдання. Викладач має можливість оцінити завдання, виконане студентом.
FR-15	Перегляд оцінки за виконане завдання. Викладач або студент мають можливість переглядати виставлені оцінки за завдання, які їм належать.
FR-16	Модуль відстеження оцінок. Студент має можливість відкрити сторінку особистих оцінок.
FR-17	Перегляд списку завдань та оцінок. Студент має можливість переглядати список оцінок у вигляді таблиці, де відображені завдання та оцінки за них.
FR-18	Перегляд суми оцінок з кожного предмету окремо. Студент має можливість переглядати суму балів з кожного предмету окремо для легшого орієнтування з приводу допуску до екзамену.
FR-19	Панель адміністратора. Адміністратор має можливість відкрити сторінку для управління застосунком.

Продовження таблиці 2.40

ID вимоги	Назва та опис
FR-20	Створення університету. Адміністратор має можливість створити університет, в якому будуть проводитись навчання.
FR-21	Створення факультету. Адміністратор має можливість створити факультет.
FR-22	Створення кафедри. Адміністратор має можливість створити кафедру.
FR-23	Створення рівня викладача. Адміністратор має можливість створити рівень викладача.
FR-24	Надання прав викладача користувачеві. Адміністратор має можливість надати права викладача користувачеві.
FR-25	Додавання викладача до факультету. Адміністратор має можливість додати викладача до факультету.
FR-26	Створення групи на кафедрі. Адміністратор має можливість створити групу на кафедрі.
FR-27	Додавання студента до групи на кафедрі. Адміністратор має можливість додати студента до групи на кафедрі.
FR-28	Створення предмету на кафедрі. Адміністратор має можливість створити предмет на кафедрі.
FR-29	Створення групи для предмету. Адміністратор має можливість створити групу для предмету. Це вирішить проблему виборних предметів.
FR-30	Додавання студента до групи предмету. Адміністратор має можливість додати студента до групи предмету.

Продовження таблиці 2.40

ID вимоги	Назва та опис
FR-31	Створення групи для предмету як групи на кафедрі. Адміністратор має можливість створити групу для предмету як групу на кафедрі.

	UC-01	UC-02	UC-03	UC-04	UC-05	UC-06	UC-07	UC-08	UC-09	UC-10	UC-11	UC-12	UC-13	UC-14	UC-15	UC-16	UC-17	UC-18	UC-19	UC-20	UC-21	UC-22	UC-23	UC-24	UC-25	UC-26	UC-27	UC-28	UC-29	UC-30	UC-31	UC-32	UC-33	UC-34	UC-35	UC-36	UC-37
FR-01	+	+	+	+																																	
FR-02	+	+																																			
FR-03			+	+																																	
FR-04																										+	+	+									
FR-05																										+											
FR-06																											+										
FR-07																											+										
FR-08																												+									
FR-09																														+	+	+	+	+	+	+	+
FR-10																																+					
FR-11																														+	+					+	+
FR-12																																	+				
FR-13																																					+
FR-14																																		+	+		
FR-15																													+								
FR-16																													+								
FR-17																													+								
FR-18																													+								
FR-19					+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+												
FR-20					+	+																															
FR-21							+	+																													
FR-22									+	+																											
FR-23											+	+																									
FR-24													+	+																							
FR-25															+	+																					
FR-26																	+	+																			
FR-27																			+	+																	
FR-28																					+	+															
FR-29																							+														
FR-30																								+													
FR-31																									+												

Рисунок 2.3 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

Нефункціональні вимоги

До нефункціоналих вимог дуже важливо віднести продуктивність та безпеку, бо, на мою думку, ці дві складові є найважливішими.

Продуктивність:

- час відповіді сервера не повинен перевищувати двух секунд при навантаженні до тисячі користувачів одночасно. Це забезпечить швидкодію та кращий досвід користування вебзастосунком;
- сторінки повинні завантажуватися за три секунди або менше.

Безпека:

- особисті важливі дані такі, як пароль користувача мають бути односторонньо зашифрованими;
- дані користувачів мають передаватися по зашифрованому протоколу HTTPS.

Висновки до розділу

Під час виконання даного розділу була створена UML діаграма варіантів використання, на якій були зображені всі основні можливості майбутнього користувача вебзастосунку, а саме: були винесені три основні ролі: студент, викладач та адміністратор, для кожного окремо були описані всі їхні майбутні можливості. Також варіанти використання були детально розписані за допомогою таблиць, що в подальшому допоможе у різних ситуаціях при розробці: у виключних та успішних.

Далі були розроблені функціональні вимоги програмного забезпечення: розписані основні модулі вебзастосунку та можливості, які будуть надані у кожному з них. Основними майбутніми модулями є аутентифікація користувача, спільний та захищений чат, єдина система звдань та відстеження оцінок. Створена матриця трасування вимог, в якій чітко відображено, який варіант використання покриває спеціальну функціональну вимогу. На матриці чітко видно, що всі функціональні вимоги були у кінцевому результаті успішно покриті варіантами використання.

Розроблено нефункціональні вимоги, в яких виокремлено дві основні: безпека та швидкодія, що є найголовнішими у використанні вебзастосунку.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Архітектура вже була частково розглянута під час аналізу відомих технічних рішень: була обрана клієнт-серверна архітектура через свою масштабованість та безпечність управління даними. Методом взаємодії був обраний протокол HTTP через свою простоту, документованість та швидкість розробки. База даних була обрана реляційна, а саме: PostgreSQL через свою масштабованість, сторогої дотримуваності до стандарту мови запитів SQL та великої кількості розширень. Для зображення архітектури застосунку була використана C4 Модель, зображена на рисунках (3.1-3.5).

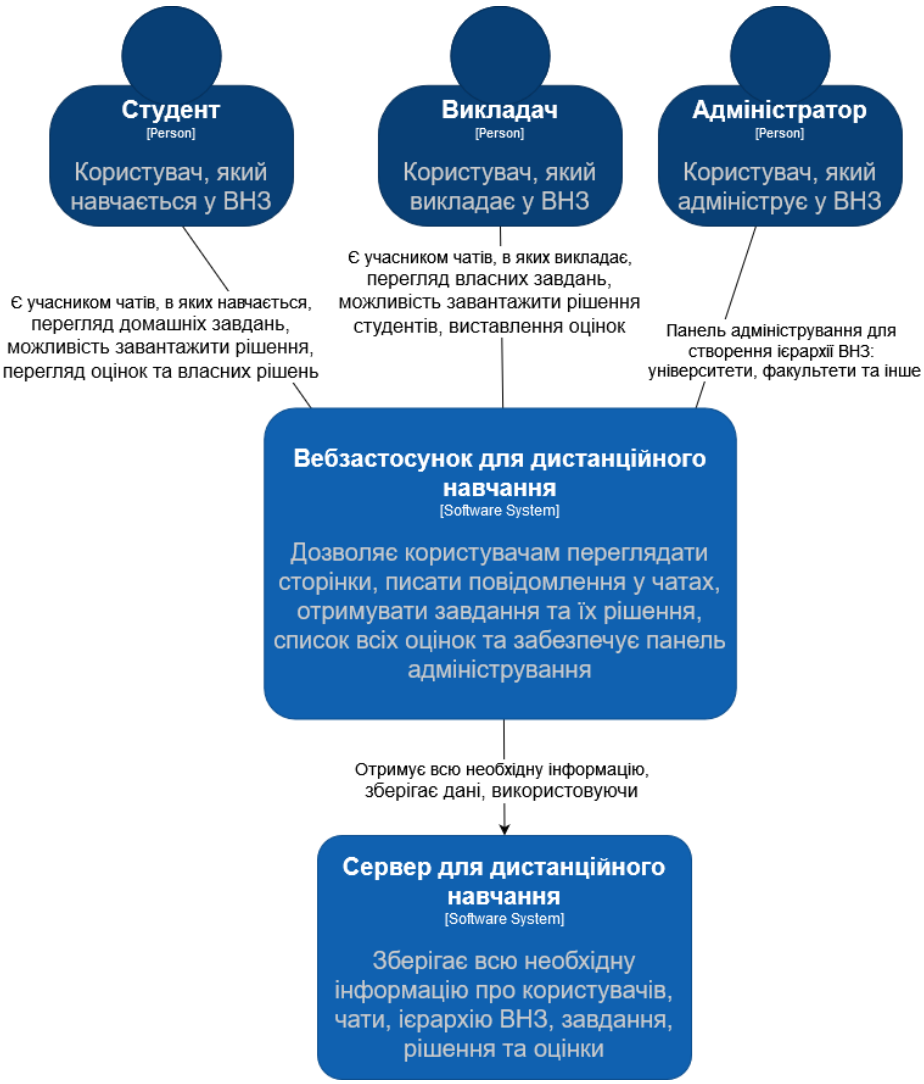


Рисунок 3.1 – Представлення високорівневої архітектури через контекстний рівень моделі C4

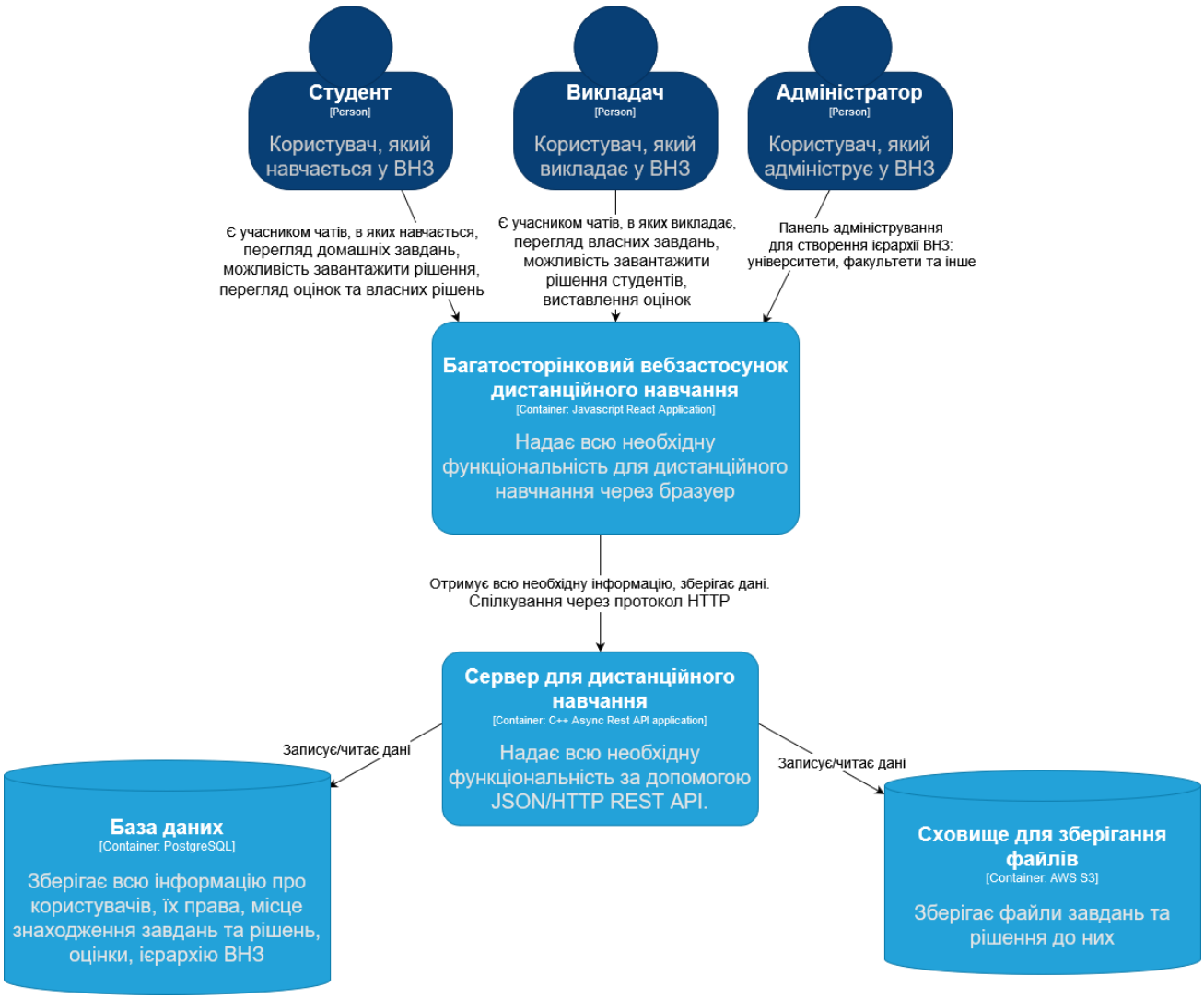


Рисунок 3.2 – Представлення високорівневої архітектури через контейнерний рівень моделі C4

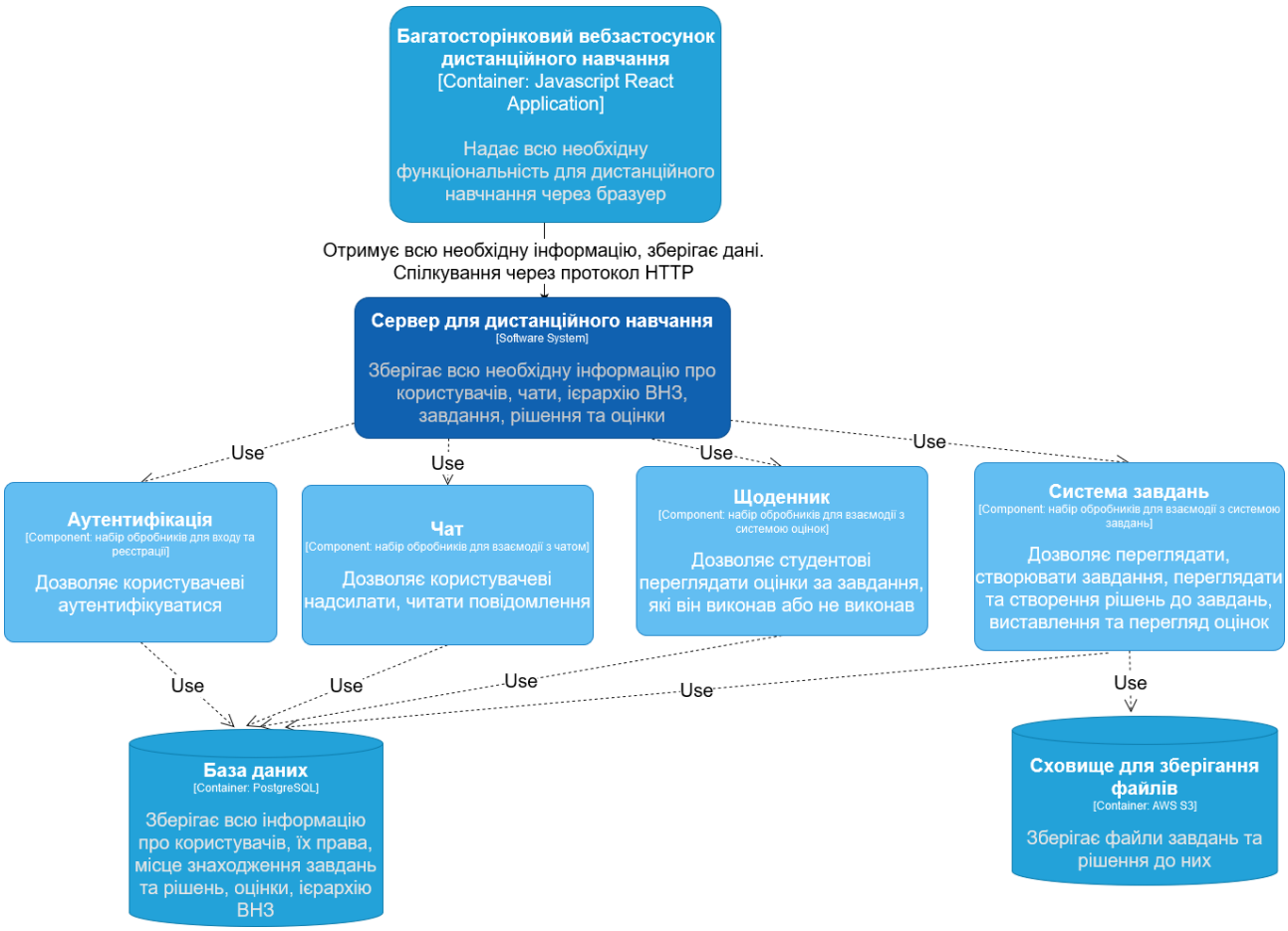


Рисунок 3.3 – Детальне представлення архітектури через компонентний рівень моделі C4

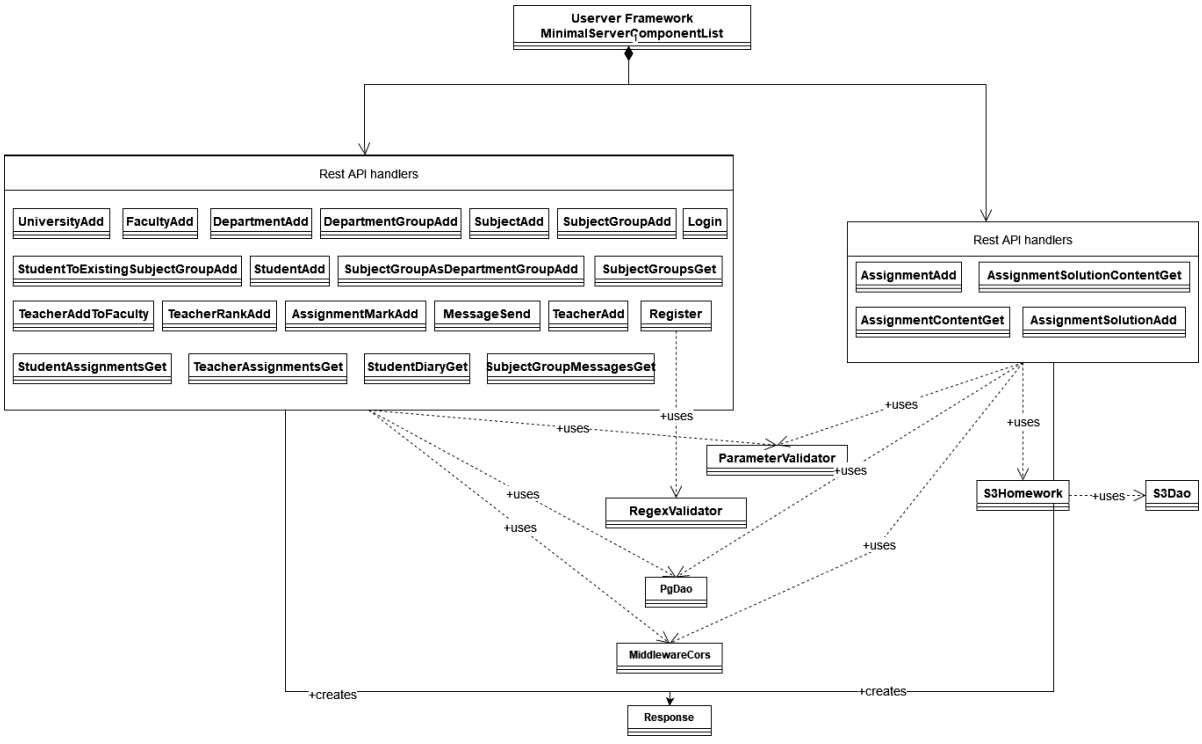


Рисунок 3.4 – Детальне представлення архітектури через кодовий рівень моделі C4 серверної частини

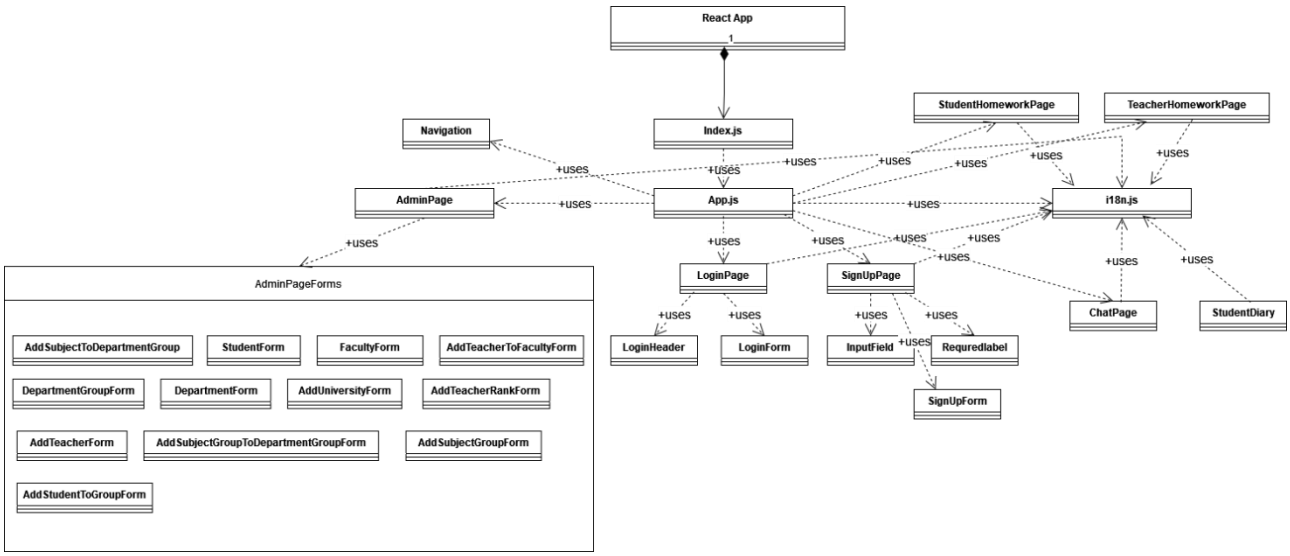


Рисунок 3.5 – Детальне представлення архітектури через кодовий рівень моделі C4 клієнтської частини

Для модулю авторизації було обрано підхід через зберігання даних у реляційній базі даних PostgreSQL. Це забезпечить збереження всієї важливої інформації для кожного користувача, щоб не можна було випадково втратити її. Єдина система завдань зберігає файли завдань та їх рішень я в окремому сховищі AWS S3, а їх шлях до бази даних. Це вирішить проблему втрати даних з усіх робіт, виконаних викладачем та студентом.

Продуктивність вебзастосунку забезпечується завдяки рішенню зробити сервер асинхронним, завдяки цьому серверна частина не буде блокуватися та очікувати, поки попередній запит буде обробленим, щоб розпочати обробку нового.

Безбека вебзастосунку вирішується завдяки використанню технології CORS, що дозволяє лише вебзастосунку відправляти запити на сервер для майбутньої обробки. Зв’язок між клієнтом та сервером виконується через зашифрований протокол HTTPS та важливі дані користувача, такі, як пароль, мають бути односторонньо зашифровані в базі даних.

Далі будуть наведений технологічний стек, а саме дві таблиці, таблиця 3.1 для відображення серверного стеку та таблиця 3.2 для клієнтського вебзастосунку.

Таблиця 3.1 – Технологічний стек серверної частини

Тип технології	Назва технології
Операційна система	Ubuntu 22.04
Архітектурний патерн	Моноліт
База даних	PostgreSQL
Розбиття сервісів на контейнери	Docker

Таблиця 3.2 – Технологічний стек клієнтської частини

Тип технології	Назва технології
Операційна система	Будь-яка з графічним інтерфейсом
Архітектурний патерн	Моноліт

3.2 Обґрунтування вибору засобів розробки

Для того, щоб завершити технологічний стек, потрібно обрати мову програмування, середовище для зручної та швидкої розробки. Для початку важливо обрати мову програмування. Для серверної частини будуть розглядатися мови C++, Golang та JavaScript.

C++

Переваги:

- швидкість та ефективність: C++ забезпечує високий рівень продуктивності та низькі затримки завдяки близькості до апаратного забезпечення;
- контроль над ресурсами: дає можливість точного контролю над пам'яттю і ресурсами, що є важливим для оптимізації;
- зрілість та надійність: довга історія розвитку та використання у промисловості забезпечує стабільність та надійність;
- статична типізація даних: дозволяє уникнути помилок через те, що мова є компільованою.

Недоліки:

- складність у вивченні: високий поріг входу через управління пам’яттю.

Golang

Переваги:

- конкурентність: вбудована підтримка конкуретного програмування, що спрощує розробку багатопотокових додатків;
- статична типізація: через те, що мова є компільованною не виникають проблем з типізацією даних.

Недоліки:

- менша продуктивність: хоче дана мова програмування вважається швидкою, їй дуже далеко до C++;
- менший контроль: через менший котроль мова сама працює з оперативною пам’яттю, що може бути критичним для високопродуктивних задач.

JavaScript

Переваги:

- швидкий старт та розробка: завдяки великому набору бібліотек та їх документації пришвидшує процес розробки;
- одномовний стек: можливість використання однієї мови для клієнту та серверу.

Недоліки:

- обмежена продуктивність: при виликої кількості користувачів можуть виникати проблеми зі швидкістю;
- однопоточність: хоча мова використовує асинхронність для обробки задач, мова досі залишається однопоточною.

Після даного порівняння приходимо до висновку, що C++ є так званим «швейцарським ножем», який підходить для будь-якої задачі. Через свою швидкість є найращим вибором для великих проектів особливо для серверних рішень. Для клієнтської частини вебзастосунку був обраний JavaScript, що є

світовим стандартом через те велику кількість матеріалів та швидкість розробки. Кожен браузер з коробки підтримує дану мову.

Для серверної частини на мові C++ був обраний найновітніший фреймворк Userver. Серед вибору були Userver та Drogon. На рисунку 3.6 відображена порівняльна таблиця на сайті розробника фреймворку Userver. На ній можемо бачити, що є лише два фреймворку для розробки C++ серверної частини. Це Userver та Drogon, також явно видно, що Userver виграє по всім поріянням через його масштабованість. Даний фреймворк позіціоную себе як «простота мови Python, швидкість мови C++» [10].

Feature	userver	go-micro 4.7.0	dapr 1.5.3	actix 0.13.0 + tokio 1.19.2	drogon 1.7.5
Programming model for IO-bound apps	stackful coroutines	stackful coroutines	actors	stackless coroutines	callbacks / stackless coroutines
Programming language to use	C++	Go-lang	Python, JS, .Net, PHP, Java, Go	Rust	C++
Caching data from remote or DB	✓ [↗]	✗	✗	✗	✗
Dynamic Config [1]	✓ [↗]	✓ [↗]	✗	✗	✗
Unit testing	✓ C++ [↗]	✓ via Go-lang	✓ PHP [↗]	✓	✓ [↗]
Functional Testing [2]	✓ [↗]	✗	✗ [↗]	✗ [↗]	✗ [↗]
Async synchronization primitives	✓ [↗]	✓ via Go-lang	✗ forces turn based access	✓ [↗]	✗
Dist locks	✓	✓ [↗]	✗ [↗]	± third-party libs	✗
Async HTTP client	✓ [↗]	✓	✓	✓	✓ [↗]
Async HTTP server	✓ [↗]	✓	✓	✓	✓
Async gRPC client	✓ [↗]	✓	✓	± third-party libs	✗
Async gRPC server	✓ [↗]	✓	✓	± third-party libs	✗
Async PostgreSQL	✓ [↗]	± third-party driver	✓ [↗]	✗ manual offloading	✓ [↗]
PostgreSQL pipelining, binary protocol	✓ [↗]	✗	✗	± third-party libs	✗
Async Redis	✓ [↗]	± third-party driver	✓ [↗]	± third-party libs	✓ [↗]
Async Mongo	✓ [↗]	± third-party driver	✓ [↗]	✗ manual offloading	✗ [↗]
Async ClickHouse	✓ [↗]	± third-party driver	✗	± third-party libs	✗ [↗]
Async MySQL	✓ MySQL Driver - EXPERIMENTAL	± third-party driver	✓ [↗]	✗ [↗]	✓ [↗]
Metrics	✓ [↗]	± third-party driver	✓ [↗]	✗	✗
No args evaluation for disabled logs	✓ [↗]	✗	✗	± third-party libs	✗
Secrets Management	± [↗]	?	✓	?	?
Distributed Tracing	✓ [↗]	?	✓ [↗]	± third-party libs	✗
JSON, BSON, YAML	✓ [↗]	± third-party libs	± third-party libs	± third-party libs	± only JSON
Content compression/decompression	✓	✓	?	✓	✓
Service Discovery	✓ DNS, DB topology discovery	✓ [↗]	?	?	?
Async TCP/UDP	✓ [↗]	✓	?	✓ [↗]	✗
Async TLS Socket	✓ [↗]	✓	?	± third-party libs	✗
Async HTTPS client	✓ [↗]	✓	?	✓	?
Async HTTPS server	✓ [↗]	?	?	✓	?
WebSockets Server	✓ [↗]	± third-party libs	✗ [↗]	± third-party libs	✓ [↗]
Deadlines and Cancellations	✓	?	?	?	± [↗]
Retries and Load Balancing	✓	✓ [↗]	✓	?	?

Рисунок 3.6 – Порівняння фреймворків

Даний фреймворк містить в собі все необхідне для розробки асинхронних, масштабованих серверних застосунків. Але все ж знадобляться ще дві окремі бібліотки. Перша – jwt-cpp для модулю аутентифікації, що є кращим вибором серед бібліотек, що підтримують стандарт JWT. Друга – AWS SDK, яка не має аналогів, тому що дана бібліотека створена розробниками Amazon задля використання своїх сервісів через будь-яку мову програмування. Дана

бібліотека знадобиться для модулю завдань, тому що завдання десь потрібно зберігати. Вони будуть зберігатися у сховищі AWS S3, а доступ до даних буде виконуватись через дану бібліотеку.

Серед IDE буде винесено найпопулярніші Visual Studio Code, Vim та CLion.

Visual Studio Code

Переваги:

- розширюваність: велика кількість розширень для різних мов програмування, включаючи підтримку синтаксису, інтеграцію з системами контролю версій, налагоджувачі, підтримка Docker, AWS та інших технологій;
- інтуїтивно зрозумілий інтерфейс: зручний графічний інтерфейс з підтримкою вкладок, інтеграцією терміналу та зручними інструментами для налагодження;
- велика спільнота: активна спільнота користувачів та розробників, що створюють та підтримують численні плагіни та розширення.
- підтримка роботи з хмарними сервісами: легка інтеграція з AWS, що корисно для вашого проекту.

Недоліки:

- немає зручної інтеграції з CMake: потрібно самому вручну налаштовувати процес перетворення коду у бінарний файл;
- немає автоматичної інтеграції з відладкою коду: потрібно вручну налаштовувати підключення відладки до бінарного файлу.

Vim

Переваги:

- легкість та швидкість: дуже легкий редактор, швидкий старт та низькі вимоги до системних ресурсів;
- можливість кастомізації: велика кількість плагінів та скриптів для налаштування під власні потреби;
- портативність: працює на будь-якій операційній системі, навіть на серверних системах без графічного інтерфейсу;

- ефективність роботи з текстом: потужні команди для редагування тексту, що дозволяють швидко та ефективно працювати з кодом.

Недоліки:

- крута крива навчання: потребує значного часу для освоєння команд та налаштувань;
- обмежений графічний інтерфейс: відсутність графічного інтерфейсу, що може бути незручно для деяких розробників;

Мало підтримки сучасних інструментів: менше інтеграцій з сучасними інструментами та сервісами, такими як AWS, у порівнянні з VS Code;

мало підтримки сучасних інструментів: менше інтеграцій з сучасними інструментами та сервісами, такими як AWS, у порівнянні з VS Code.

CLion

Переваги:

- інтеграція з CMake: ідеальний для проектів на C++ з використанням CMake, що може бути корисно, якщо ваш проект включає компоненти на C++;
- потужний налагоджувач: вбудований потужний налагоджувач, який дозволяє зручно відслідковувати помилки та виконання коду;
- рефакторинг коду: інструменти для автоматичного рефакторингу коду, що допомагають підтримувати чистоту та організованість коду;
- підтримка різних мов: підтримує не лише C/C++, але й Python, JavaScript та інші мови через розширення.

Недоліки:

- ціна: платна ліцензія, що може бути недосяжною для невеликих команд або індивідуальних розробників;
- вимоги до ресурсів: високі вимоги до системних ресурсів, що може уповільнювати роботу на слабких машинах;
- обмеженість у порівнянні з VS Code: Менше підтримки розширень та інтеграцій у порівнянні з VS Code.

Для розробки серверної частини обраний CLion через швидкість налаштування проєкту: все працює з коробки, відладник коду, безкоштовна ліцензія для студентів та автоматична інтеграція з CMake.

А для розробки клієнтської частини був обраний Visual Studio Code через його легкість та адаптивність.

3.3 Конструювання програмного забезпечення

База даних була вже обрана у попередніх розділах, а саме PostreSQL. Далі буде представлена логічна модель (рисунок 3.7) майбутьої БД за допомогою представлення у вигляді ER діаграми.

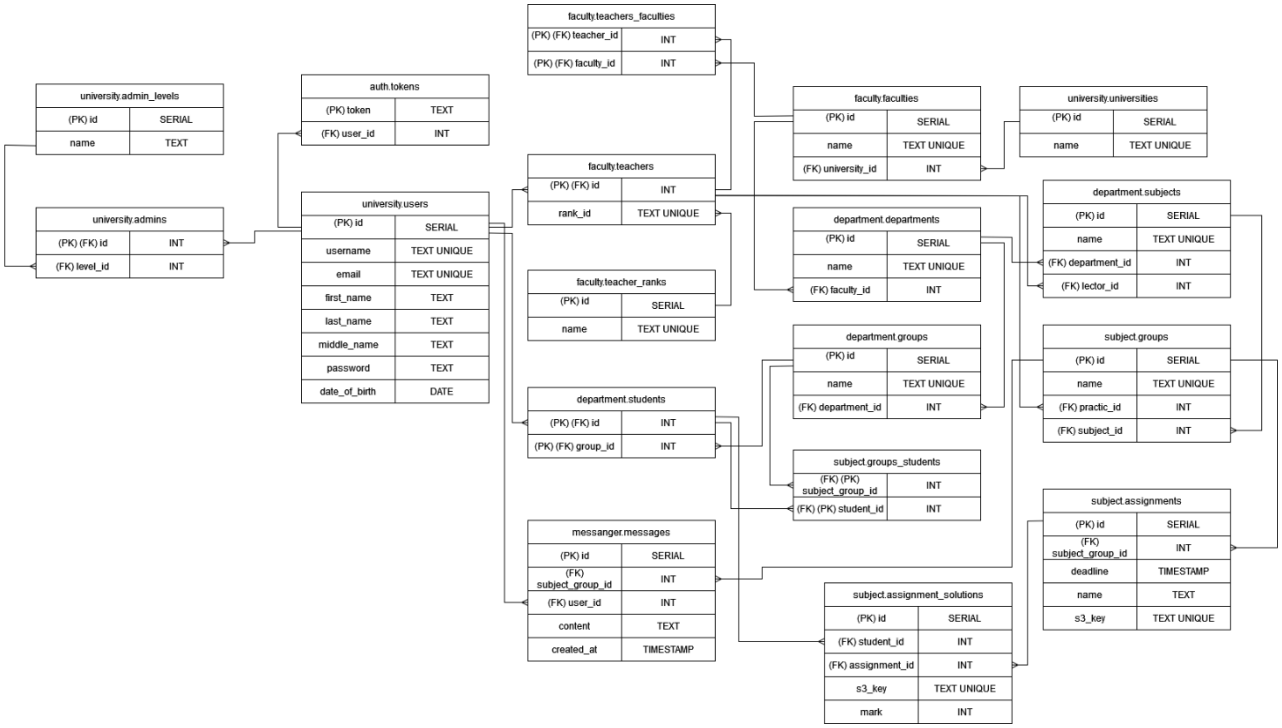


Рисунок 3.7 – ER діаграма сутностей до БД

Детальна структура класів вже була представлена у попередньому розділі, можна ознайомитися на рисунку 3.4.

Опис таблиць бази даних наведено у таблицях 3.3 - 3.20. Модель бази даних наведена на рисунку 3.7.

Таблиця 3.3 – Опис таблиці university.universities

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер університету
name	text	назва університету

Таблиця 3.4 – Опис таблиці university.users

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер користувача
username	text	ім’я акаунту
email	text	електронна пошта користувача
first_name	text	ім'я користувача
last_name	text	прізвище користувача
middle_name	text	по-батькові користувача
password	text	пароль
date_of_birth	date	дата народження

Таблиця 3.5 – Опис таблиці faculty.faculties

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер факультету
name	text	назва факультету
university_id	int	ідентифікаційний номер університету

Таблиця 3.6 – Опис таблиці faculty.teacher_ranks

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер
name	text	назва ступеня викладача

Таблиця 3.7 – Опис таблиці department.departments

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер кафедри
name	text	назва
faculty_id	int	ідентифікаційний номер факультету

Таблиця 3.8 – Опис таблиці department.groups

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер групи на кафедрі
name	text	назва групи
department_id	int	ідентифікаційний номер кафедри

Таблиця 3.9 – Опис таблиці faculty.teachers

Назва поля	Тип даних	Опис
id	int	ідентифікаційний номер користувача
rank_id	int	ідентифікаційний номер ступеня викладача

Таблиця 3.10 – Опис таблиці faculty.teachers_faculties

Назва поля	Тип даних	Опис
teacher_id	int	ідентифікаційний номер вчителя
faculty_id	int	ідентифікаційний номер факультету

Таблиця 3.11 – Опис таблиці department.students

Назва поля	Тип даних	Опис
id	int	ідентифікаційний номер користувача
group_id	int	ідентифікаційний номер групи на кафедрі

Таблиця 3.12 – Опис таблиці department.subjects

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер предмету
name	text	назва
department_id	int	ідентифікаційний номер кафедри
lector_id	int	ідентифікаційний номер викладача (лектора)

Таблиця 3.13 – Опис таблиці subject.groups

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер групи предмету
name	text	назва
subject_id	int	ідентифікаційний номер предмету
practic_id	int	ідентифікаційний номер викладача (практика)

Таблиця 3.14 – Опис таблиці subject.groups_students

Назва поля	Тип даних	Опис
subject_group_id	int	ідентифікаційний номер групи предмету
student_id	int	ідентифікаційний номер студента

Таблиця 3.15 – Опис таблиці subject.assignments

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер завдання
subject_group_id	int	ідентифікаційний номер групи предмету
deadline	timestamp	срок здачі завдання
name	text	назва
s3_key	text	локація зберігання файлу завдання

Таблиця 3.16 – Опис таблиці subject.assignment_solutions

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер рішення завдання
student_id	int	ідентифікаційний номер студента
assignment_id	int	ідентифікаційний номер завдання
s3_key	text	локація зберігання файлу рішення
mark	int	оцінка за завдання

Таблиця 3.17 – Опис таблиці auth.tokens

Назва поля	Тип даних	Опис
token	text	токен для авторизації користувача
user_id	int	ідентифікаційний номер користувача

Таблиця 3.18 – Опис таблиці messanger.messages

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер повідомлення
subject_group_id	int	ідентифікаційний номер групи предмету
user_id	int	ідентифікаційний номер користувача
content	text	текст повідомлення
created_at	timestamp	дата та час створення запису повідомлення

Таблиця 3.19 – Опис таблиці university.admin_levels

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер рівню адміна
name	text	назва рівню адміністрування

Таблиця 3.20 – Опис таблиці university.admins

Назва поля	Тип даних	Опис
id	INT	ідентифікаційний номер користувача
level_id	INT	ідентифікаційний номер рівню адміністрування

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.21.

Таблиця 3.21 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	CLion	Головне середовище розробки програмного забезпечення серверної частини дипломної роботи.
2	Visual Studio Code	Головне середовище розробки програмного забезпечення клієнтської частини дипломної роботи.
3	Postman	Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів.

Продовження таблиці 3.21

№ п/п	Назва утиліти	Опис застосування
4	Psql	Програмне забезпечення яке надає командний інтерфейс CLI для доступу до бази даних.

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Безпека даних є однією з найважливіших складових у програмному забезпеченні, оскільки дані користувачів, файли завдань та рішень потрібно зберігати та не бути взразливими. Візьмемо ситуацію: злоумисник отримав дані користувачів і теперь може дізнатися їх паролі на ішних сервісах, якщо вони всюди їх дублюють або ситуація, в якій злоумисник отримає всі файли, необхідні для навчання та чужі рішення до них. Саме через дані проблеми потрібно бути готовими до даних випадків. Для запобігання проблем проведемо основні вразливості вже створеного ПЗ.

Розглянемо вразливість SQLInjection. Дана вразливість є дуже популярною в наш час. Здійснюється вона шляхом передачі параметрів на сервер у вигляді запиту до БД. Через це зловмисник може зробити так звану «ін’єкцію» та отримати доступ до всіх даних сервісу. Дана проблема вирішується на серверній частині, всі поля для запиту детально перевіраються на строгий тип, допустимі значення та інші [11].

Розглянемо випадок, якщо все ж якимось чином злоумисник отримав дані користувачів через не захищенний модуль, хоча модулі всі є захищені (описано вище). В такому випадку злоумисник отримає односторонньо зашифровані дані (пароль) та облікові дані зашифрованні двосторонньо, які

можна розшифрувати, маючи лише спеціальний ключ, до якого злоумисник доступу не буде мати.

Далі розглянемо випадок, що злоумисник намагатиметься дістатися до сховища файлів AWS S3. По-перше, у нього це мало-вірогідно вийде, через те, що локально створена окрема мережа, яка не має доступу до Інтернету, а лише сервер має до нього доступ. По-друге, сховище захищене методом CORS, який дозволяє лише окремим доменним ім’ям або айпі-адресам під’єднуватися. По-третє, якщо якимось чином злоумисник все ж дістався файлів, файли виявляться знову зашифрованими, дешифрувати він не сможе через те, що не буде мати ключ для цього.

Дані між клієнтом і сервером передаються по протоколу HTTPS. Тобто злоумисник, якщо навіть зміг перехватити дані, також отримає їх у зашифрованному вигляді, що зробить ці дані повністю непригодними до використання.

Висновки до розділу

У результаті написання даного розділу була детально описана архітектура програмного забезпечення за допомогою C4 Model, представлені чотири рівні. Перші два є високорівневими: контекстний, контейнерний. Інші два є більш деталізованими: компонентний та кодовий рівні.

Далі був обгрунтований технологічний стек. Для серверу, по-перше, була обрана сучасна мова програмування C++, яка є найшвидшою високорівневою мовою у світі, фреймворк для пришвидшення розробки Userver, декілька окремих бібліотек, перша для авторизації – jwt-cpp та друга для підключення до сховища файлів AWS SDK. Для клієнту була обрана мова JavaScript, що є найкращою мовою для вебзастосунків та найвідоміший фреймворк для пришвидшення розробки React.

Для розробки були обрані дані IDE для зручного та швидкого написання коду. Для клієнту це Visual Studio Code, а для серверу – Clion. База даних

PostgreSQL була представлена у вигляді ER-діаграми, описані всі окремі сутності, їх поля та зв’язки між ними.

На прикінці розділу був проведений аналіз безпеки даних, в якому було виявлено та вирішенні всі небезпечні місця у програмному забезпеченні.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

У наш час аналіз якості програмного забезпечення (ПЗ) є дуже важливим для забезпечення надійності, ефективності та безпеки систем. Важливим завданням аналізу якості ПЗ є визначення метрик, за допомогою яких буде виконано перевірку ПЗ. У наукових дослідженнях зазначено, що список метрик якості коду є дуже великим і кожну з них перевірити неможливо. Тому в ході цієї роботи будуть використані наступні метрики: повторюваність коду, складність коду та проблеми в коді. Отже, можемо визначити подальшу мету розділу.

Метою тестування веб-застосунку є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- перевірка сумісності веб-застосунку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox тощо);
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу.

Метриками для оцінки якості ПЗ обрано наступні:

- Дублювання коду: вимірюється у відсотках і відображає, скільки коду повторюється в різних місцях проекту. Високий відсоток дублювання коду може вказувати на необхідність рефакторингу для покращення підтримуваності та зменшення технічного боргу.
- Складність коду: визначає складність реалізації логіки програмного забезпечення. Ця метрика допомагає виявити складні ділянки коду, які можуть бути важкими для розуміння та підтримки. Висока складність коду може призвести до більшої кількості помилок та ускладнює процес тестування.

- Попередження: відображає відсоток попереджень, виявлених статичним аналізатором коду, що можуть вказувати на потенційні проблеми. Високий відсоток попереджень свідчить про наявність потенційних ризиків, які слід усунути для покращення якості коду.

Такі метрики дозволяють всебічно оцінити якість веб-застосунку, виявити слабкі місця та забезпечити їх усунення для досягнення високих стандартів якості програмного забезпечення.

Такі метрики були обрані, тому що вони всі в тому чи іншому ступені перевіряються за допомогою інструментів аналізу. Існує безліч інструментів для аналізу якості коду, деякі з яких мають більше метрик, інші — менше, але в загальному названі метрики присутні майже у всіх статичних аналізаторах коду.

Для аналізу якості коду на C++ та React (де C++ використовується для бекенду, а React для фронтенду) можна використовувати спеціальні програмні засоби. До них належать лінтери та інші інструменти статичного аналізу коду, такі як clang-tidy, ESLint та інші.

clang-tidy — це інструмент для статичного аналізу коду C++, який допомагає знайти та виправити помилки, покращити кодову базу та дотримуватися найкращих практик програмування. Він забезпечує перевірку коду на наявність потенційних проблем, включаючи оптимізацію продуктивності та безпеки [12].

ESLint — це інструмент для аналізу коду JavaScript та TypeScript, який широко використовується в проектах на React. Він допомагає знайти помилки у коді та дотримуватися встановлених правил стилю програмування, що значно покращує читабельність та підтримуваність коду [13].

Codacy — це хмарний сервіс для автоматизованого аналізу коду, який дозволяє виявляти помилки, вразливості та проблеми в коді. Він підтримує багато мов програмування та інтегрується з різними системами безперервної інтеграції [14].

Таблиця 4.1 – Порівняння інструментів для аналізу на якость коду

Властивість	Eslint	clang-tidy	Codacy
Безкоштовний	+	+	+
Підтримка C++	-	+	+
Підтримка JavaScript	+	-	+
Статичний аналіз коду	+	+	+
Безпека	+	+	+
Робота у хмарі	-	-	+
Автоматична інтеграція з системою контролю версій	-	-	+

Після порівняння інструментів в таблиці 4.1 можемо дійти до висновку, що логічніше за все було б використовувати інструмент, який підтримує обидві мови програмування та працює у хмарному середовищі, що інтегрується з системою контролю версій. Саме тому був обраний Codacy. На рисунку 3.1 можна побачити результат аналізу даного інструменту.

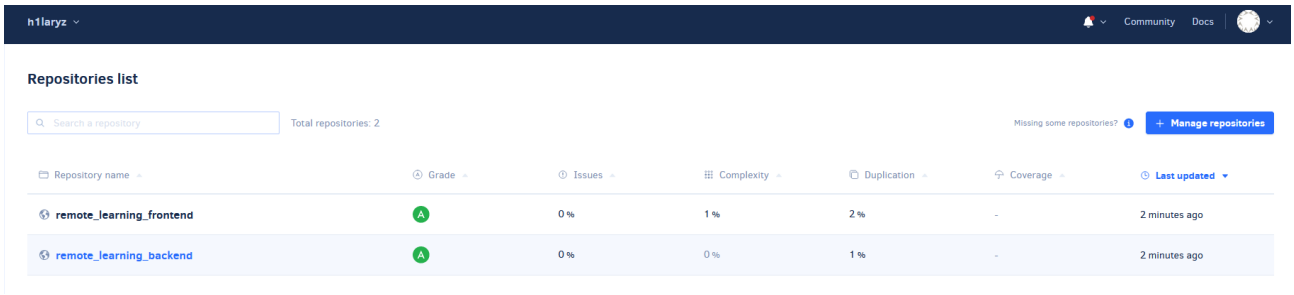


Рисунок 4.1 – Аналіз якості коду за допомгою веб-застосунку Codacy

Робимо висновок, що якість коду є задоволеню з відміткою A. Відстоток проблем у фронтенді та бекенді близько 0%, складність є лише у фронтенді, що є лише 1% всього коду та кількість повторюваного коду є допустимою 1% та 2% відповідно.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Для подальшого тестування був обраний мануальний вид тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.2 – 4.27.

Таблиця 4.2 – Тест 1.1 Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Прізвище, ім'я, по-батькові, ім'я користувача, електронна пошта, пароль, дата народження.
Опис проведення тесту	У відповідні поля вводяться: електронна пошта, яка до цього не була зареєстрована в системі, пароль, що має хоча б одну велику літеру, хоч б один спеціальний символ, хоча б одна цифра та розмір паролю має бути більше за 10 символів; ім’я, по-батькові (необов’язково), прізвище та дата народження Після цього натискається кнопка підтвердження реєстрації.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.3 – Тест 1.2 Авторизація користувача

Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні дані	Ім’я користувача або електронна пошта та пароль
Опис проведення тесту	У відповідні поля вводяться: електронна пошта або ім’я користувача у відповідне поле та пароль, що був використаний під час реєстрації.
Очікуваний результат	Авторизація проходить успішно, користувач потрапляє на сторінку чатів.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 2.1 Авторизація користувача

Початковий стан системи	Користувач пройшов авторизацію, автоматично потрапив на сторінку чатів або натиснув у навігації на кнопку для чатів.
Вхідні дані	-
Опис проведення тесту	Користувач переходить на сторінку чатів.
Очікуваний результат	Користувач, що є викладачем або студентом отримує список доступних для нього чатів, що знаходяться на лівій панелі навігації по сторінці.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 2.1.1 Відкриття бажаного чату

Початковий стан системи	Користувач пройшов авторизацію, автоматично потрапив на сторінку чатів або натиснув у навігації на кнопку для чатів.
Вхідні дані	-
Опис проведення тесту	Авторизований користувач на панелі навігації чатів натискає на бажаючу назву чату.
Очікуваний результат	Користувач бачить список вже відправлених раніше повідомлень, має можливість надрукувати та відправити своє повідомлення.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 2.1.2 Перегляд існуючих повідомлень

Початковий стан системи	Авторизований користувач натиснув на назву чату на сторінці чатів.
Вхідні дані	-
Опис проведення тесту	Користувач натискає на бажаючий чат та починає скролити мишкою або використовує слайдер для перегляду попередніх повідомлень.
Очікуваний результат	Користувач успішно передивляється список попередніх повідомлень.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 2.1.3 Відправлення повідомлення

Початковий стан системи	Авторизований користувач натиснув на назву чату на сторінці чатів.
Вхідні дані	Надруковане повідомлення у відповідному полі.
Опис проведення тесту	Користувач натискає на поле для друкування, друкує повідомлення, яке бажає надіслати та натискає Enter або на відповідну кнопку мишкою.
Очікуваний результат	Повідомлення успішно відправлене. Відображається дата, час повідомлення, текст, прізвище та ім’я відправника.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.8 – Тест 3.1 Створення завдання викладачем

Початковий стан системи	Авторизований викладач натискає на панелі навігації на кнопку «Завдання».
Вхідні дані	Назва групи предмету, назва предмету, назва завдання, дедлайн, файл.
Опис проведення тесту	Викладач натискає на кнопку «додати нове завдання», заповнює форму: назва групи предмету, назва предмету, назва завдання, дедлайн, обирає файл зі свого пристрою та натискає «виконати».
Очікуваний результат	Завдання успішно завантажене та відображається у викладача після завантаження.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.9 – Тест 3.1.1 Завантаження існуючого завдання на пристрій

Початковий стан системи	Авторизований користувач переходить на сторінку завдання.
Вхідні дані	-
Опис проведення тесту	Користувач натискає на назву предмету та групу (якщо є викладачем), обирає завдання та натискає на кнопку «завантажити». Файл завантажується на пристрій.
Очікуваний результат	Файл успішно завантажився на пристрій.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.10 – Тест 3.2 Прикріплення рішення до завдання

Початковий стан системи	Авторизований студент натискає на панелі навігації на кнопку «Завдання».
Вхідні дані	Файл рішення завдання
Опис проведення тесту	Студент натискає на назву предмета, на бажаюче завдання, до якого хоче прикріпити рішення, дедлайн якого ще не вийшов. Натискає на кнопку «обзор», назва кнопки залежить від локалі операційної системи. Обирає файл на пристрої та натискає на кнопку «надіслати».
Очікуваний результат	Рішення до завдання успішно прикріплене.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.11 – Тест 3.2.1 Завантаження існуючого рішення завдання на пристрій

Початковий стан системи	Авторизований студент або викладач натискає на панелі навігації на кнопку «Завдання».
Вхідні дані	-
Опис проведення тесту	Користувач натискає на назву предмета, групи (якщо є викладачем), назву завдання та натискає на кропку «завантажити рішення»
Очікуваний результат	Рішення до завдання успішно завантажене на пристрій.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.12 – Тест 3.2.2 Оцінення виконаного завдання

Початковий стан системи	Авторизований викладач натискає на панелі навігації на кнопку «Завдання».
Вхідні дані	Оцінка
Опис проведення тесту	Викладач натискає на назву предмета, назву групи, на завдання, на студента, вводить в поле «оцінка» відповідне число від 0 до 100 та натискає на кнопку «виконати».
Очікуваний результат	Оцінка успішно виставлена.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.13 – Тест 3.2.3 Перегляд оцінки за виконане завдання

Початковий стан системи	Авторизований студент натискає на панелі навігації на кнопку «Завдання».
Вхідні дані	-
Опис проведення тесту	Авторизований студент натискає на назву предмету, назву завдання та бачить у полі «оцінка» число, що і є його оцінкою за виставлене завдання, якщо оцінка ще не виставлена, то «ще не виставлено» замість числа.
Очікуваний результат	Поле з оцінкою відображається коректно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.14 – Тест 4.1 Перегляд списку завдань та оцінок

Початковий стан системи	Авторизований студент натискає на панелі навігації на кнопку «щоденник».
Вхідні дані	-
Опис проведення тесту	Студент натискає на кнопку щоденник на панелі навігації та бачить перед собою у вигляді таблиці: предмет, дату виставлення оцінки, оцінка та назву завдання, за яке отримав оцінку або пусту таблицю, якщо оцінок ще немає.
Очікуваний результат	Оцінки відображаються у вигляді таблиці.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.15 – Тест 4.2 Перегляд суми оцінок з кожного предмету

Початковий стан системи	Авторизований студент натискає на панелі навігації на кнопку «щоденник».
Вхідні дані	-
Опис проведення тесту	Студент натискає на кнопку щоденник на панелі навігації та бачить перед собою у вигляді таблиць оцінки за предмети, біля назви є сумарна кількість балів з даного предмету.
Очікуваний результат	Сумарна кількість балів відображається успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.16 – Тест 5.1 Створення університету

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Назва університету.
Опис проведення тесту	Адміністратор натискає на кнопку «створити університет», відкривається форма, в якій необхідно заповнити поле «назва університету».
Очікуваний результат	Університет створений успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.17 – Тест 5.2 Створення факультету

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Назва університету, назва факультету.
Опис проведення тесту	Адміністратор натискає на кнопку «створити факультет», відкривається форма, в якій необхідно заповнити поля «назва університету», «назва факультету».
Очікуваний результат	Факультет створений успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.18 – Тест 5.3 Створення кафедри

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Назва факультету, назва кафедри.
Опис проведення тесту	Адміністратор натискає на кнопку «створити факультет», відкривається форма, в якій необхідно заповнити поля «назва кафедри», «назва факультету».
Очікуваний результат	Кафедра створена успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.19 – Тест 5.4 Створення рівня викладача

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Рівень.
Опис проведення тесту	Адміністратор натискає на кнопку «створити рівень викладача», відкривається форма, в якій необхідно заповнити поле «рівень».
Очікуваний результат	Кафедра створений успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.20 – Тест 5.5 Надання користувачу права вчителя

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Ім'я користувача або пошта, рівень.
Опис проведення тесту	Адміністратор натискає на кнопку «надати користувачу права вчителя», відкривається форма, в якій необхідно заповнити поля «ім'я користувача або пошта», «рівень».
Очікуваний результат	Права викладача надані успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.21 – Тест 5.6 Додавання викладача до факультету

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Ім'я користувача або пошта, назва факультету.
Опис проведення тесту	Адміністратор натискає на кнопку «додати викладача до факультету», відкривається форма, в якій необхідно заповнити поля «ім'я користувача або пошта», «назва факультету».
Очікуваний результат	Викладач успішно доданий до факультету.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.22 – Тест 5.7 Створення групи на кафедрі

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Назва групи та назва кафедри.
Опис проведення тесту	Адміністратор натискає на кнопку «створити групу на кафедрі», відкривається форма, в якій необхідно заповнити поля «назва групи», «назва кафедри».
Очікуваний результат	Група на кафедрі створена успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.23 – Тест 5.8 Додавання студента до групи на кафедрі

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Ім'я користувача або пошта та назва групи
Опис проведення тесту	Адміністратор натискає на кнопку «додати студента до групи на кафедрі», відкривається форма, в якій необхідно заповнити поля «назва групи», «ім’я користувача або пошта».
Очікуваний результат	Студент успішно доданий до групи на кафедрі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.24 – Тест 5.9 Створення предмету на кафедрі

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Ім'я користувача або пошта лектора, назва кафедри, назва предмету.
Опис проведення тесту	Адміністратор натискає на кнопку «створити групу на кафедрі», відкривається форма, в якій необхідно заповнити поля «ім'я користувача або пошта лектора», «назва кафедри», «назва предмету».
Очікуваний результат	Предмет створений успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.25 – Тест 5.10 Створення групи для предмету

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Ім'я користувача або пошта практика, назва предмету, назва групи предмету.
Опис проведення тесту	Адміністратор натискає на кнопку «створити групу на кафедрі», відкривається форма, в якій необхідно заповнити поля «ім'я користувача або пошта практика», «назва предмету», «назва групи предмету».
Очікуваний результат	Група для предмету створена успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.26 – Тест 5.11 Додавання студента до групи предмету

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Ім'я користувача або пошта, назва групи предмету.
Опис проведення тесту	Адміністратор натискає на кнопку «створити групу на кафедрі», відкривається форма, в якій необхідно заповнити поля «ім'я користувача або пошта», «назва групи предмету».
Очікуваний результат	Студент успішно доданий до навчальної групи предмету.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.27 – Тест 5.12 Створенення групи для предмету як групи на кафедрі

Початковий стан системи	Авторизований адміністратор натискає на кропку «панель адміністратора» на панелі навігації.
Вхідні дані	Ім'я користувача або пошта практика, назва предмету, назва групи.
Опис проведення тесту	Адміністратор натискає на кнопку «створити навчальну групу, як групу з кафедри», відкривається форма, в якій необхідно заповнити поля «Ім'я користувача або пошта практика», «назва предмету», «назва групи».
Очікуваний результат	Навчальна група для предмету успішно створена, як група з кафедри.
Фактичний результат	Збігається з очікуваним.

4.3 Опис контрольного прикладу

Користувач (майбутній адміністратор) відкриває веб-браузер (рисунок 4.2), вводить адресу сайту, потрапляє на сторінку реєстрації (рисунок 4.3), має можливість перейти на сторінку авторизації, заповнює усі необхідні поля у форму та натискає на кнопку зареєструватися (рисунок 4.4), потрапляє автоматично на сторінку авторизації (рисунок 4.5), вводить необхідні поля для авторизації, що були використанні під час реєстрації. Даний акаунт немає доступу, поки ні до чого, поки адміністратор не відправить необхідний запит до серверу для надання доступу до адміністративної панелі (рисунок 4.6).

Користувач стає адміністратором з відповідним рівнем доступу. Переходить на сторінку «адміністративна панель», натискає на кнопку «створити університет», заповнює форму та натискає надіслати (рисунок 4.7), аналогічно натискає на «створити факультет» та заповнює дані, знову натискає

надіслати (рисунок 4.8), далі по аналогії створює всю необхідну ієрархію університету для навчання.

Наступний користувач (майбутній студент) створює акаунт на вікні зареєструватися (рисунок 4.9). Ще один користувач (майбутній викладач) створює акаунт на вікні зареєструватися (рисунок 4.10).

Адміністратор надає права доступу на сторінці адміністратора для студента та викладача (рисунки 4.11-4.12).

Студент перезаходить в акаунт, щоб оновились коректно права (рисунок 4.13). Викладач виконує аналогічну дію (рисунок 4.14). Обох користувачів перенаправляє на сторінку чатів, де вже є група предмету, що створив адміністратор (рисунок 4.15). Викладач або студент можуть натиснути на свою групу предмету та переглядати всі надіслані раніше повідолмення у дану групу (рисунок 4.16). Викладач або студент друкує повідомлення у необхідне поле та натискає на кнопку надіслати (рисунок 4.17). Викладач відкриває сторінку «завдання» у панелі навігації та бачить список створених раніше завдань, поки що їх немає (рисунок 4.18), натискає на кнопку «створити нове завдання», заповнює відповідну форму, натискає «надіслати» (рисунок 4.19). Завдання було успішно додане (рисунок 4.20). Студент заходить через панель навігації на сторінку «завдання» та бачить прикріплене завдання до своєї групи (рисунок 4.21). Студент обирає файл рішення завдання, натискає «надіслати» (рисунок 4.22). Рішення успішно завантажене. Викладач отримує нове рішення, може його розгорнути та завантажити на пристрій, виставляє оцінку (рисунок 4.23) та натискає «надіслати».

Студент натискає в панелі навігації на кнопку «щоденник» та бачить виставлену оцінку, а саме дату, коли була виставлена оцінка, назву завдання, предмету та саму оцінку, також сумарну кількість біля назви предмету (рисунок 4.24).

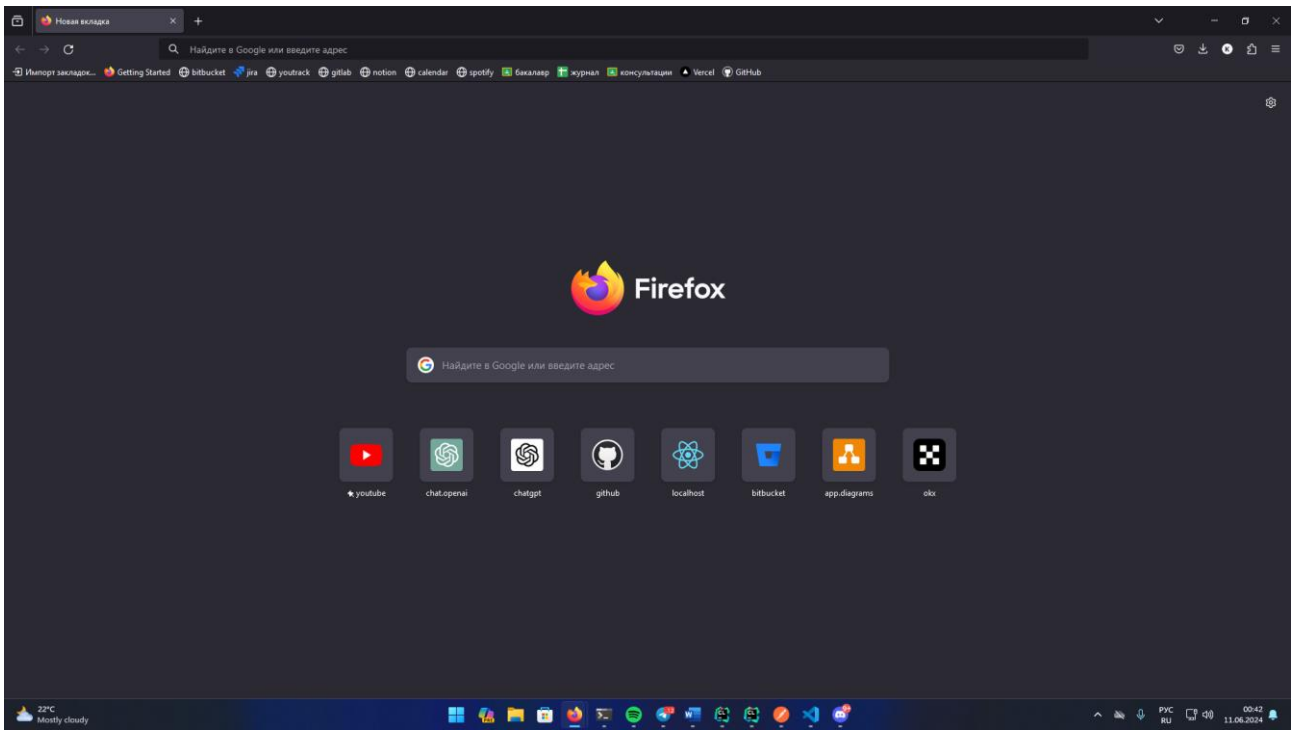


Рисунок 4.2 – Відкриття веб-браузеру

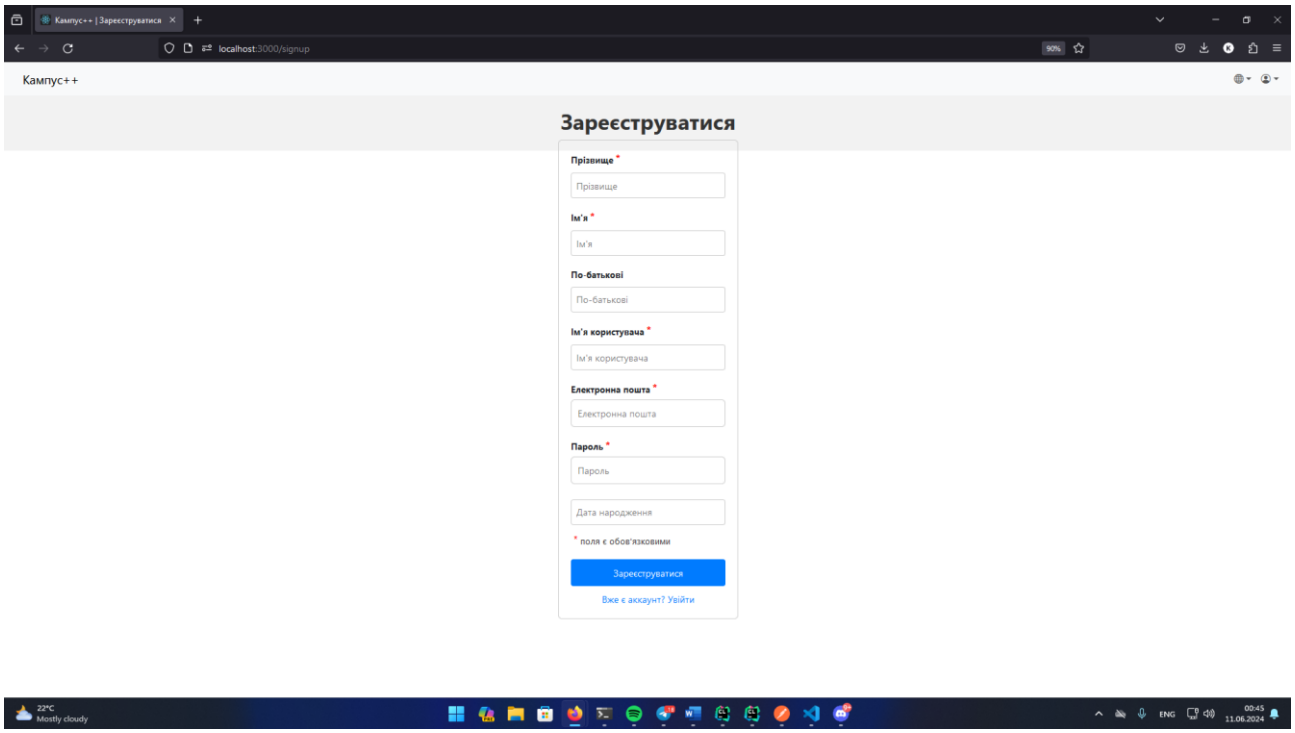


Рисунок 4.3 – Майбутній адміністратор на сторінці реєстрації

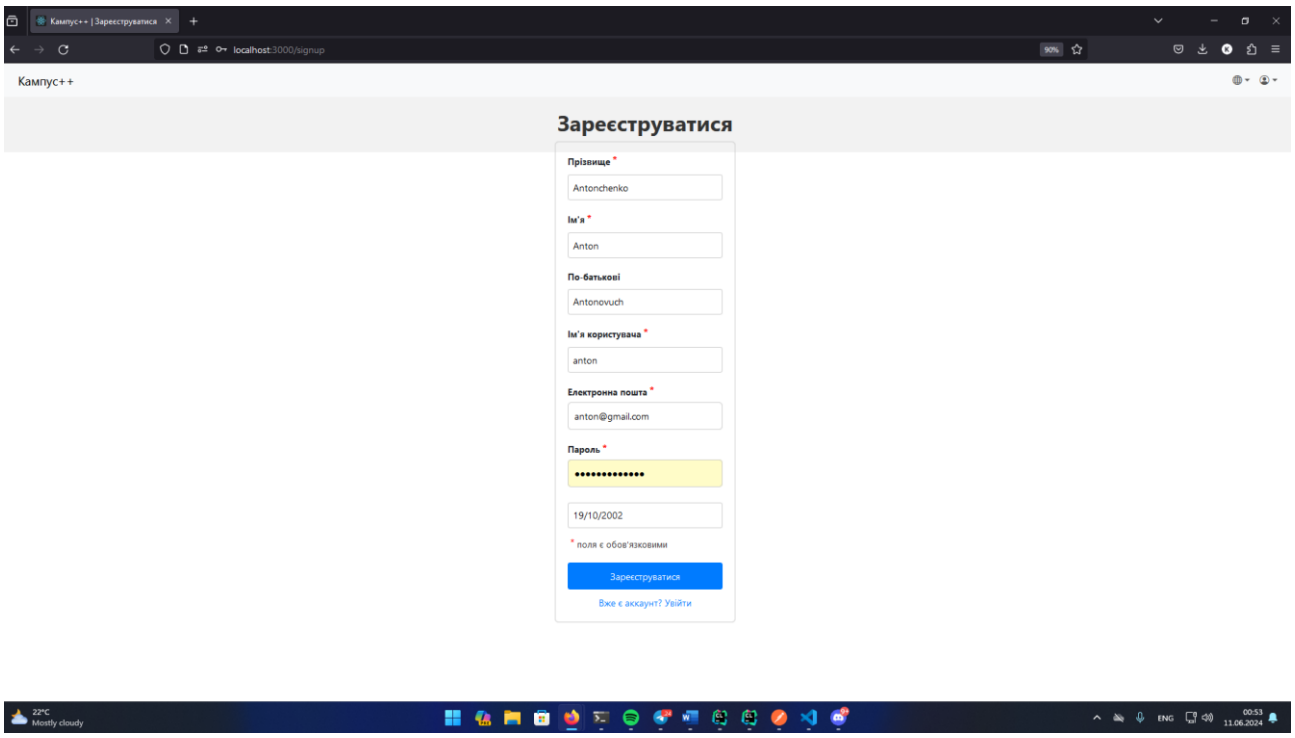


Рисунок 4.4 – Заповнення форми для реєстрації та натиск на кнопку «зареєструватися»

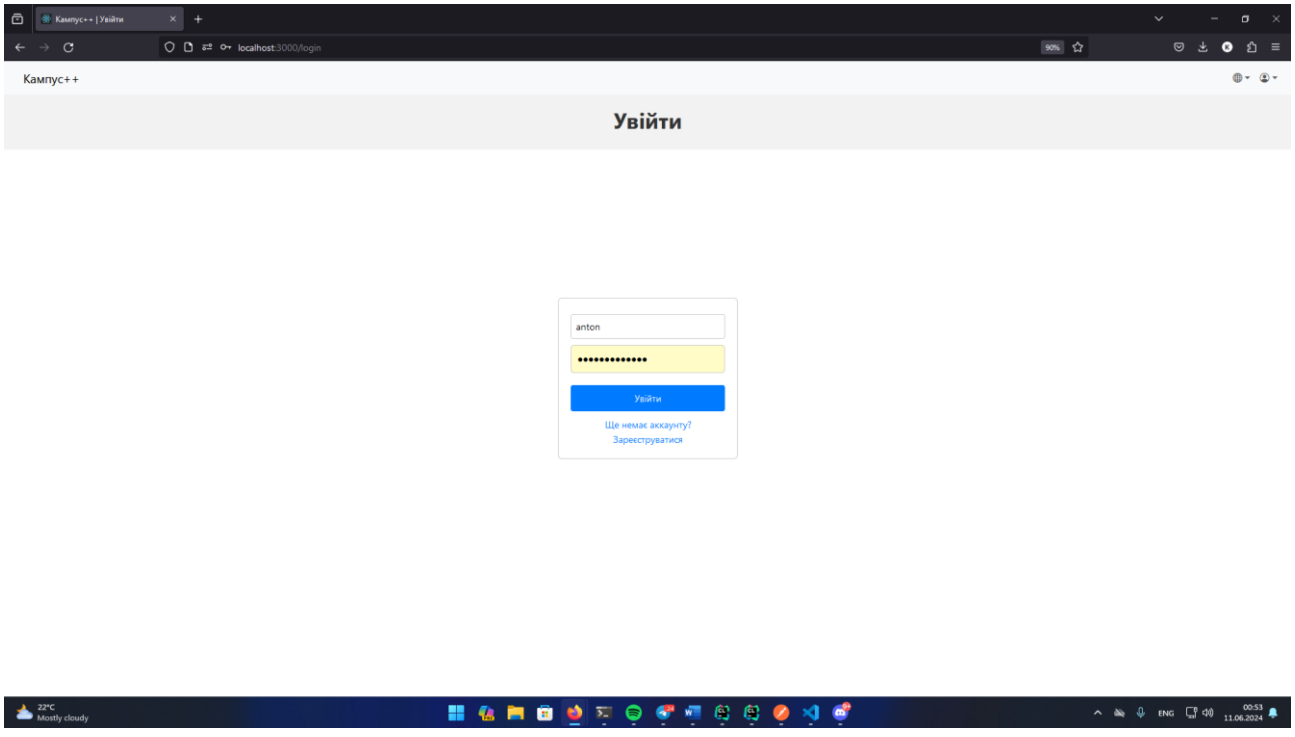


Рисунок 4.5 – Перенаправлення на сторінку авторизації після успішної реєстрації

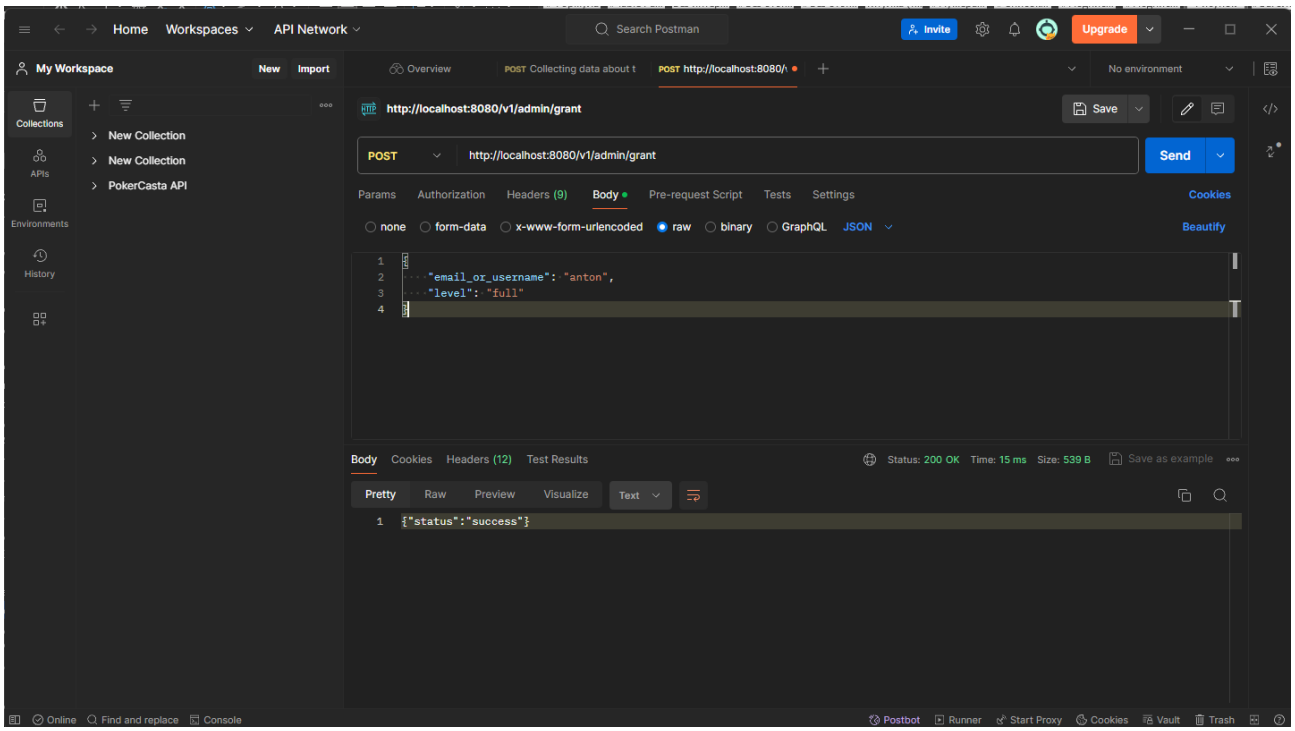


Рисунок 4.6 – Головний адміністратор відправляє запит на сервер для надання прав адміністратору користувачеві

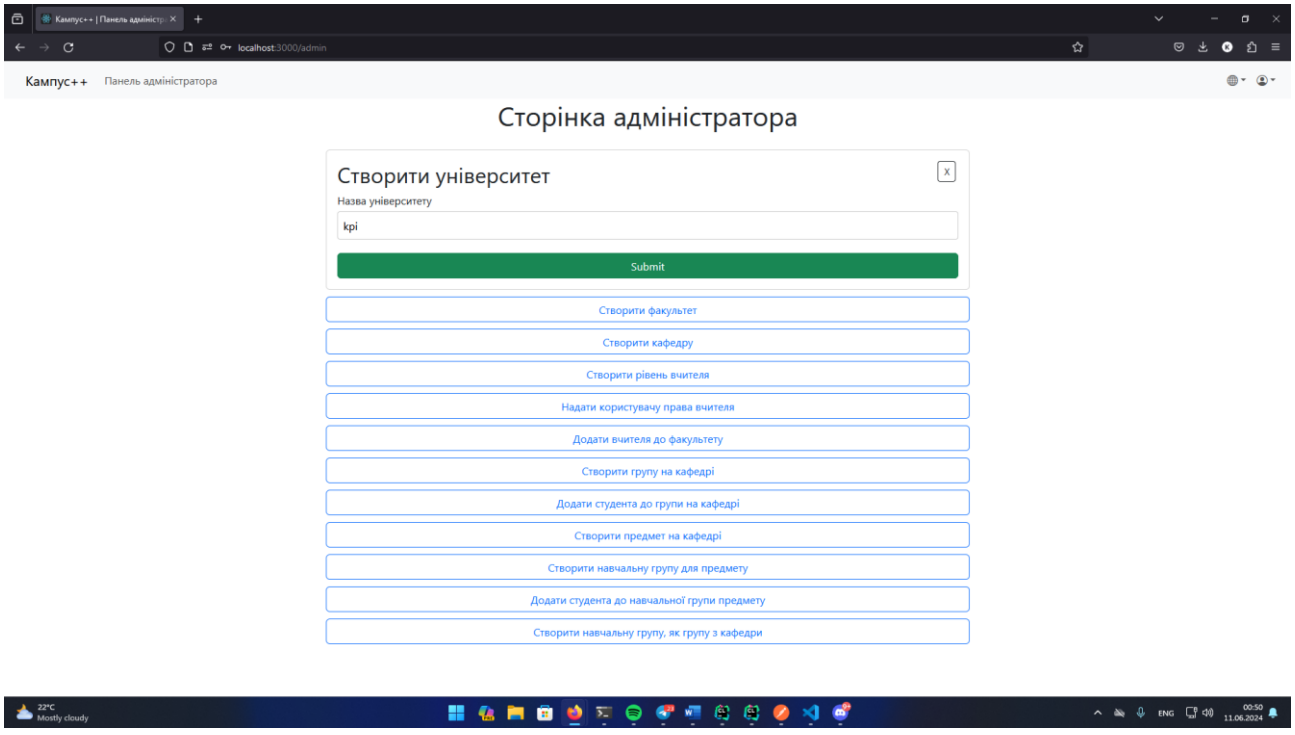


Рисунок 4.7 – Адміністратор створює університет з назвою «kpi»

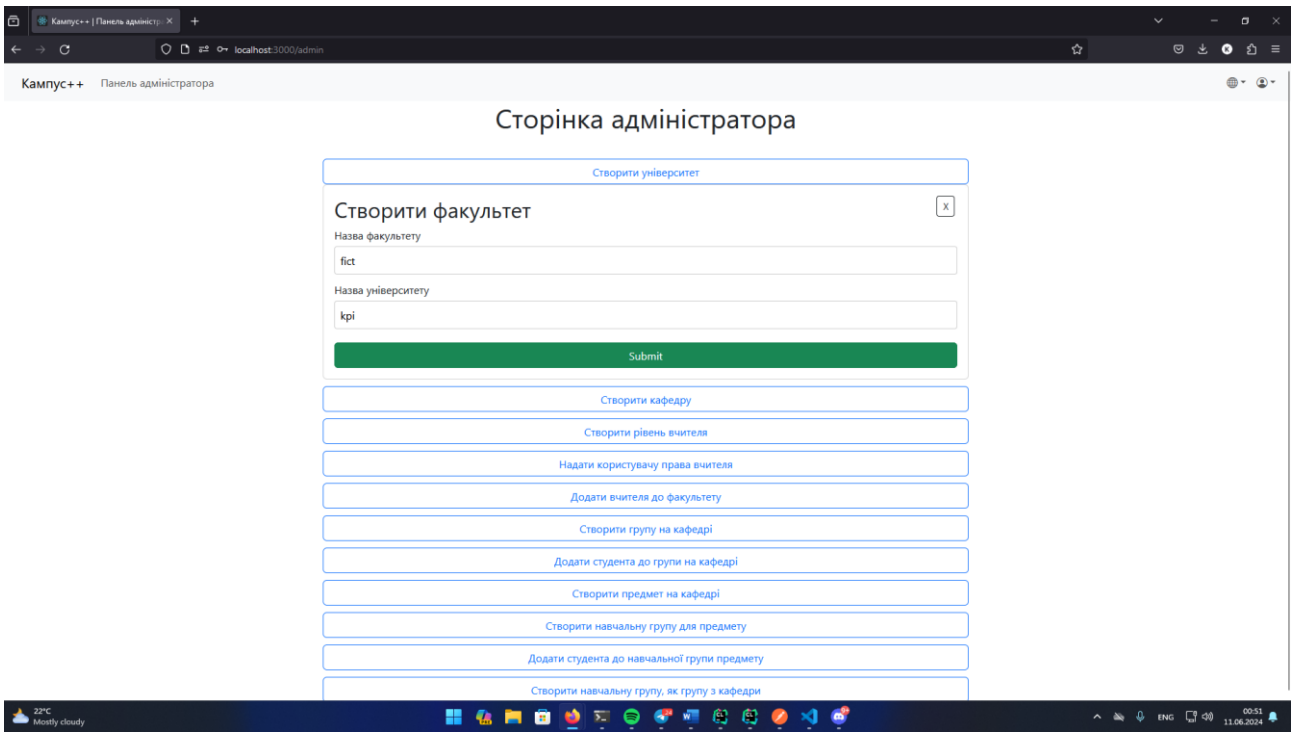


Рисунок 4.8 – Адміністратор створює факультет «fict» в університеті «kpi»

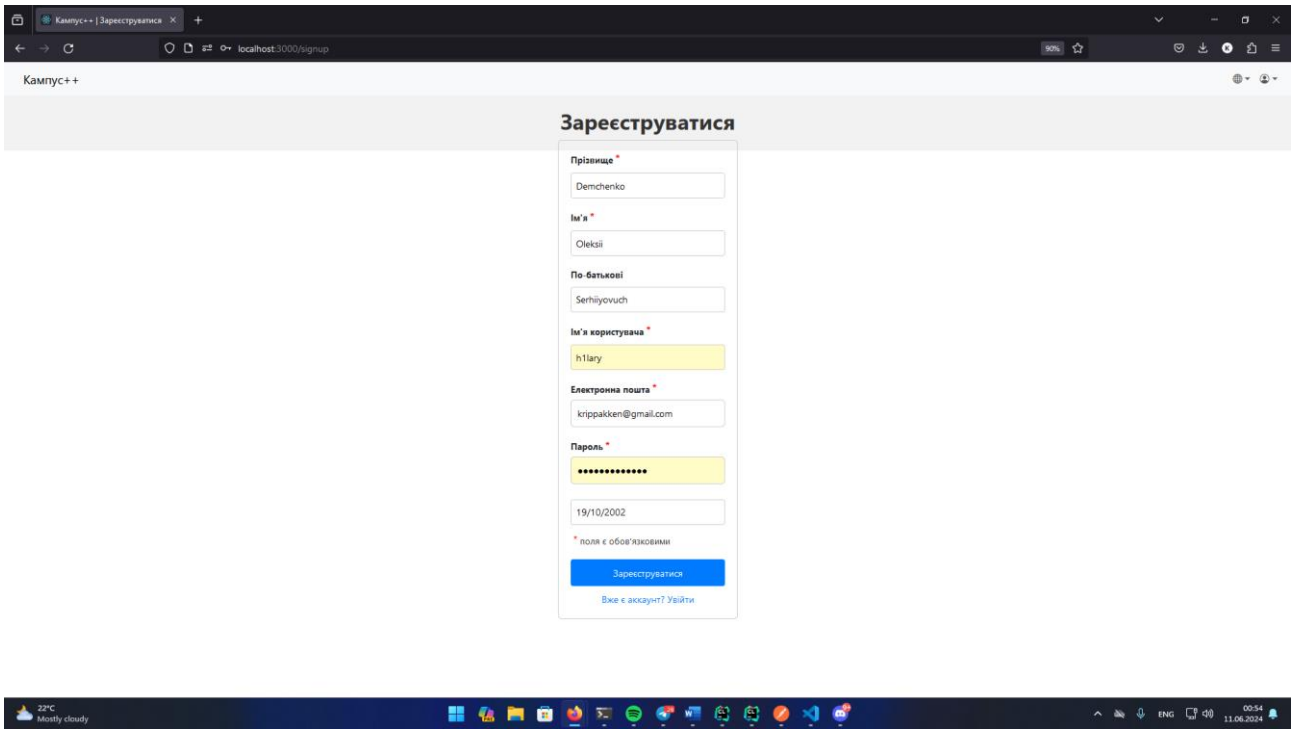


Рисунок 4.9 – Студент створює акаунт на сторінці реєстрації

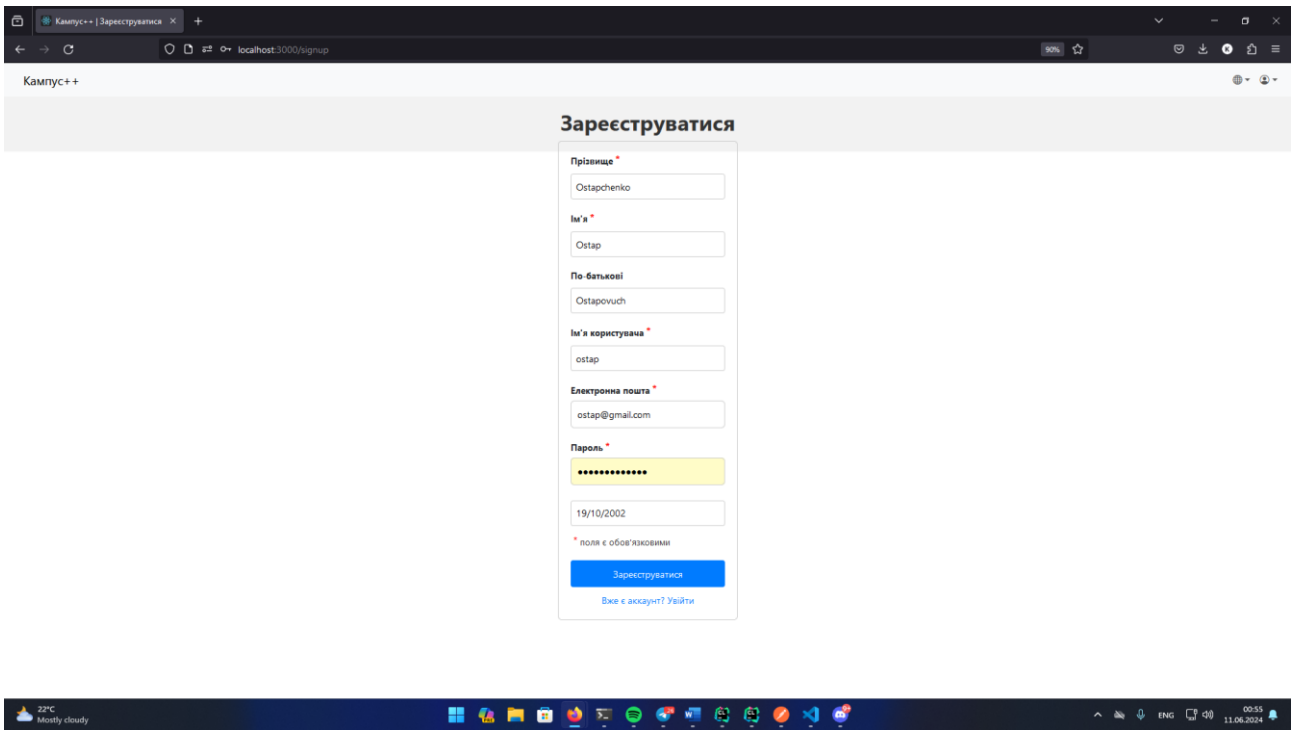


Рисунок 4.10 – Викладач створює акаунт на сторінці реєстрації

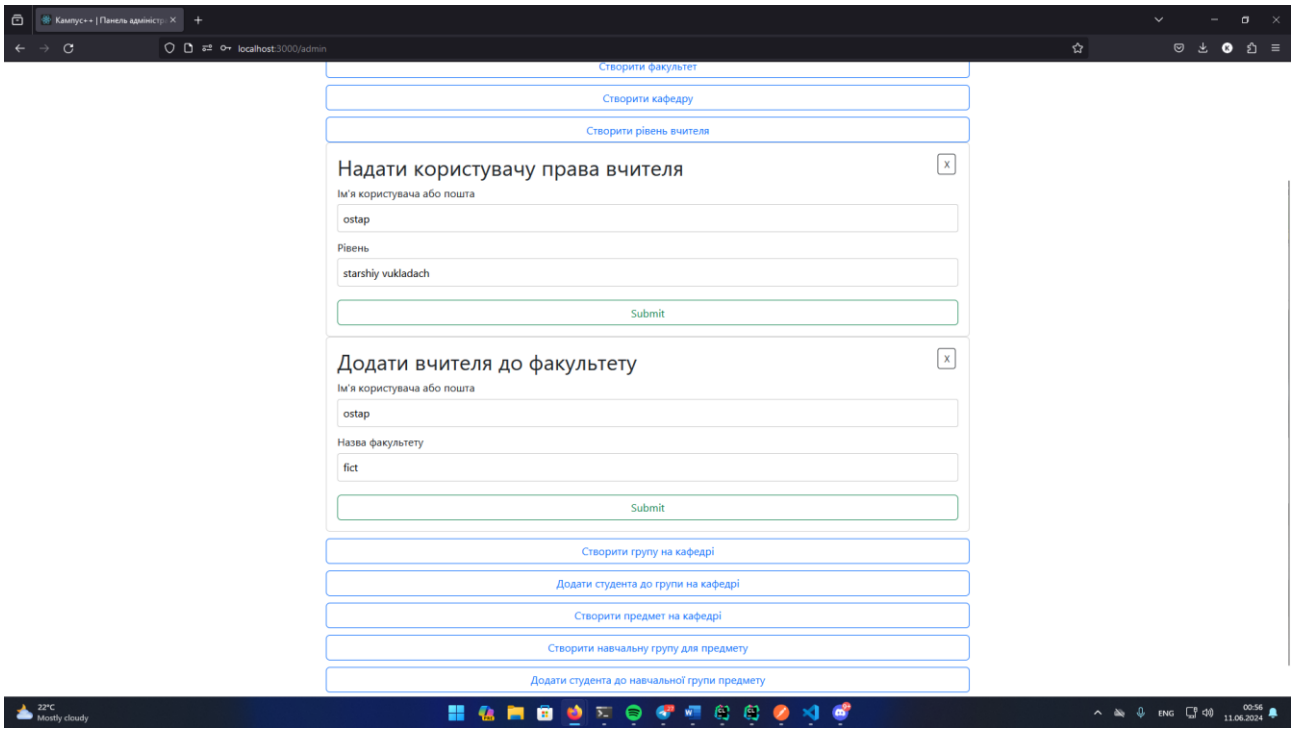


Рисунок 4.11 – Адміністратор надає викладачеві з ім’ям користувача права вчителя

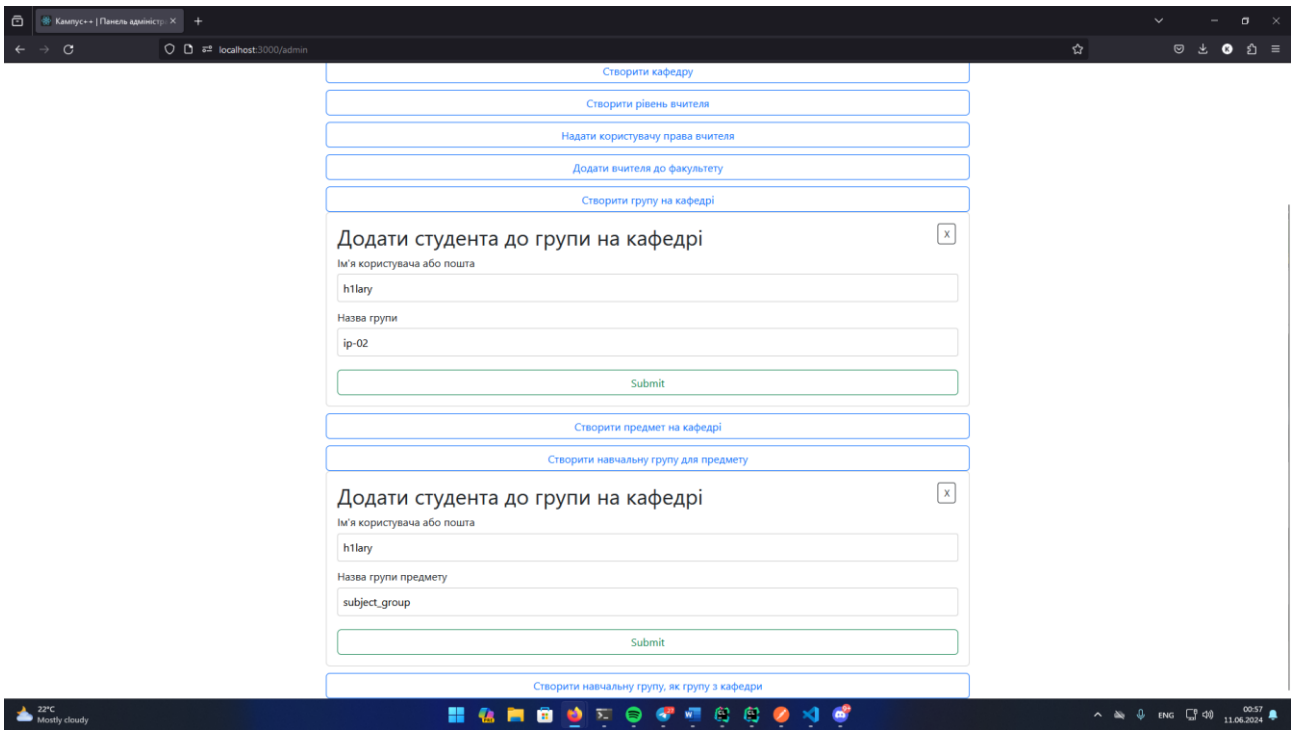


Рисунок 4.12 – Адміністратор додає користувача «h1lary» до групи на кафедрі та до групи предмету на кафедрі

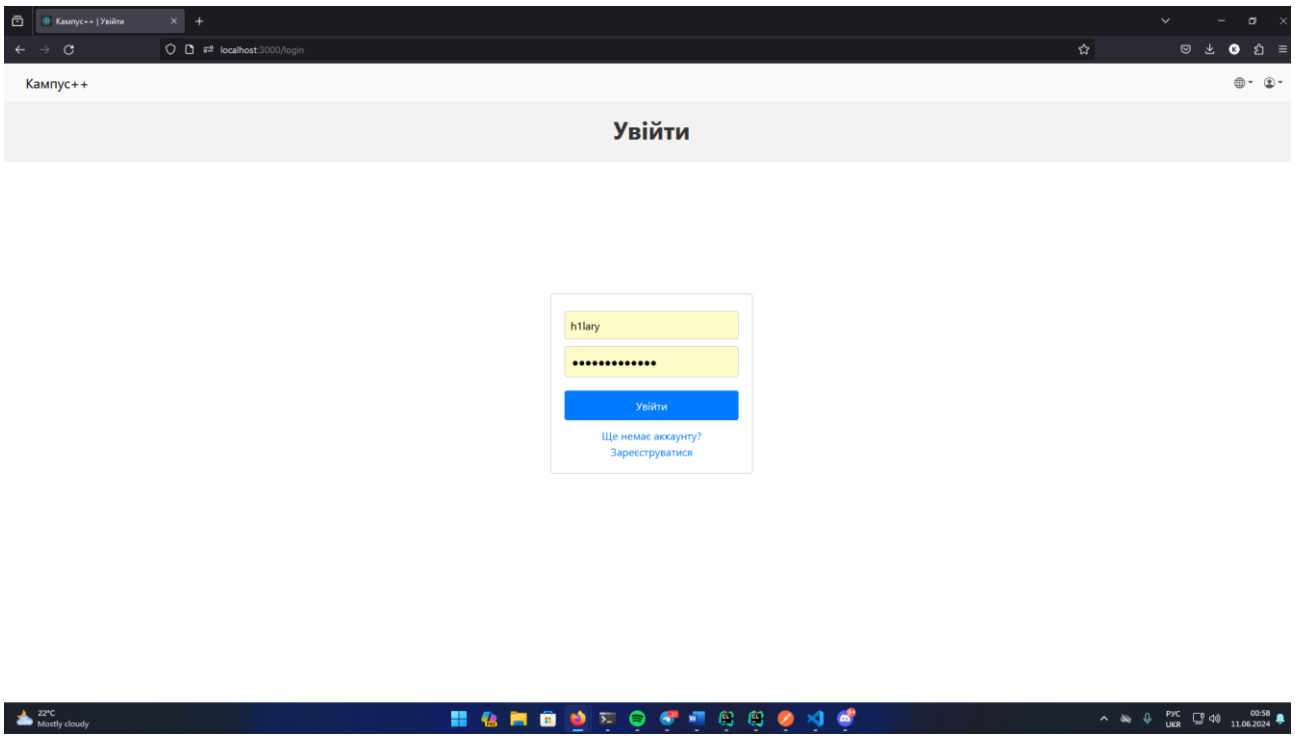


Рисунок 4.13 – Студент перезаходить у свій акаунт, щоб оновити права доступу

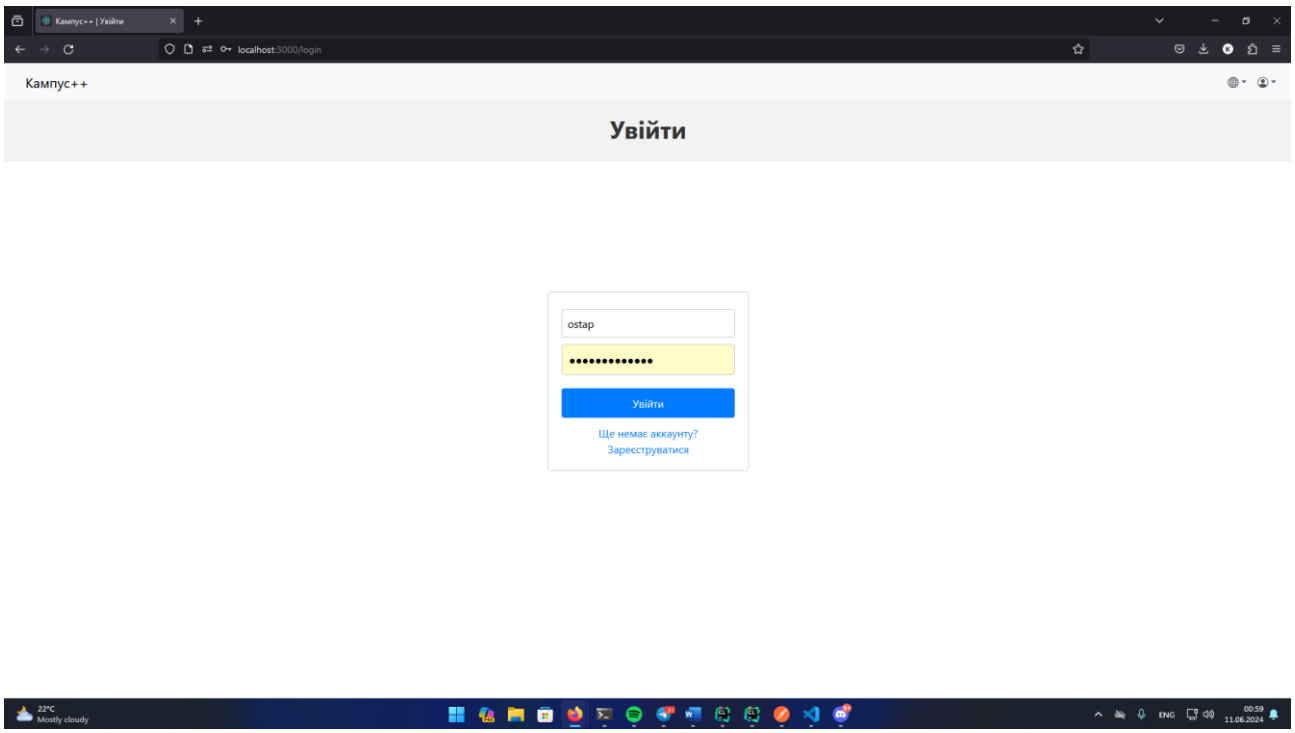


Рисунок 4.14 – Викладач перезаходить у свій акаунт, щоб оновити права доступу

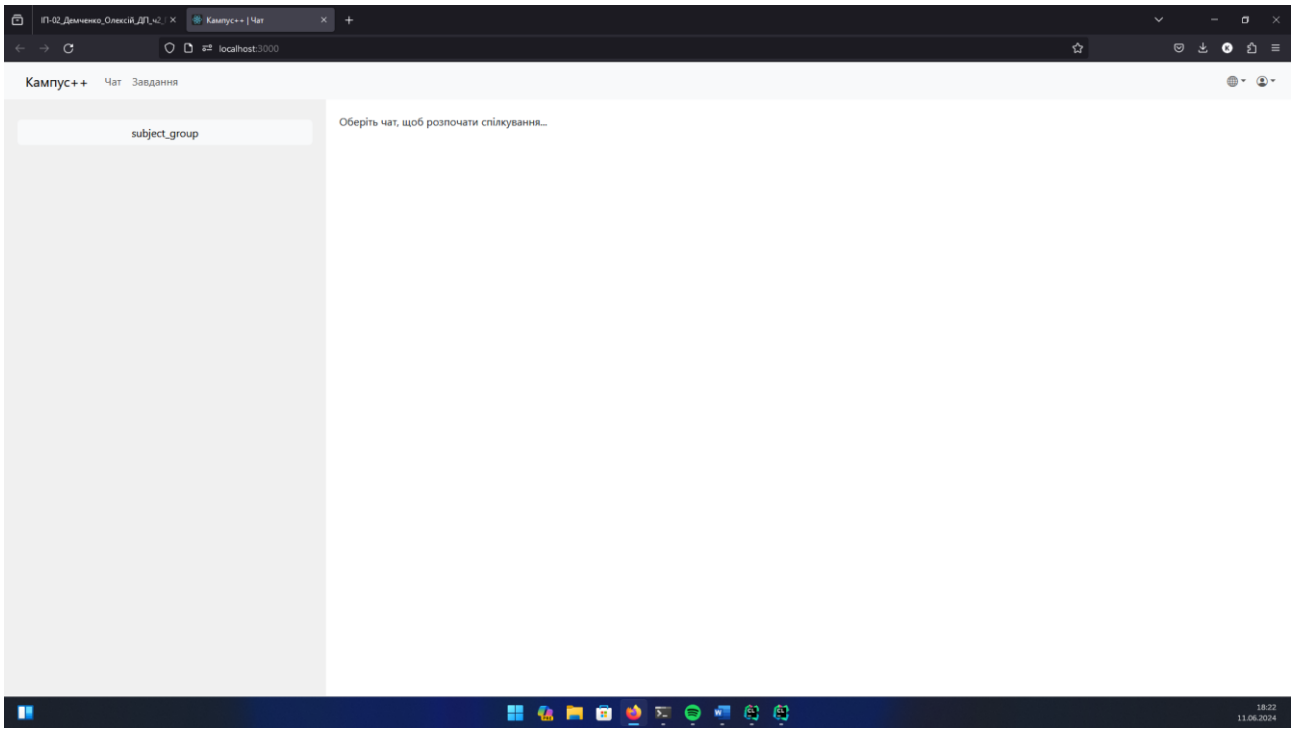


Рисунок 4.15 – Викладач або студент після авторизації перенаправлений на сторінку чатів

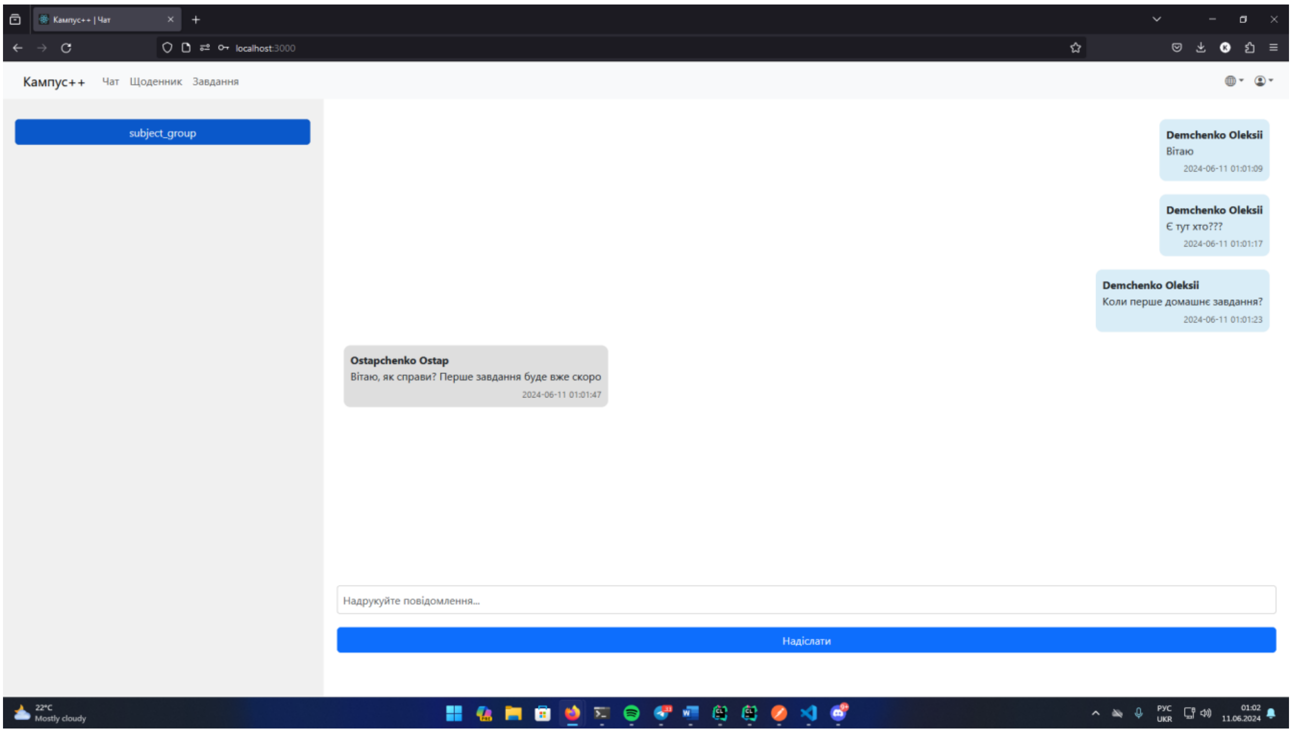


Рисунок 4.16 – Студент відкриває чат групи та може преглядати старі повідомлення (справа – власні, зліва – чужі)

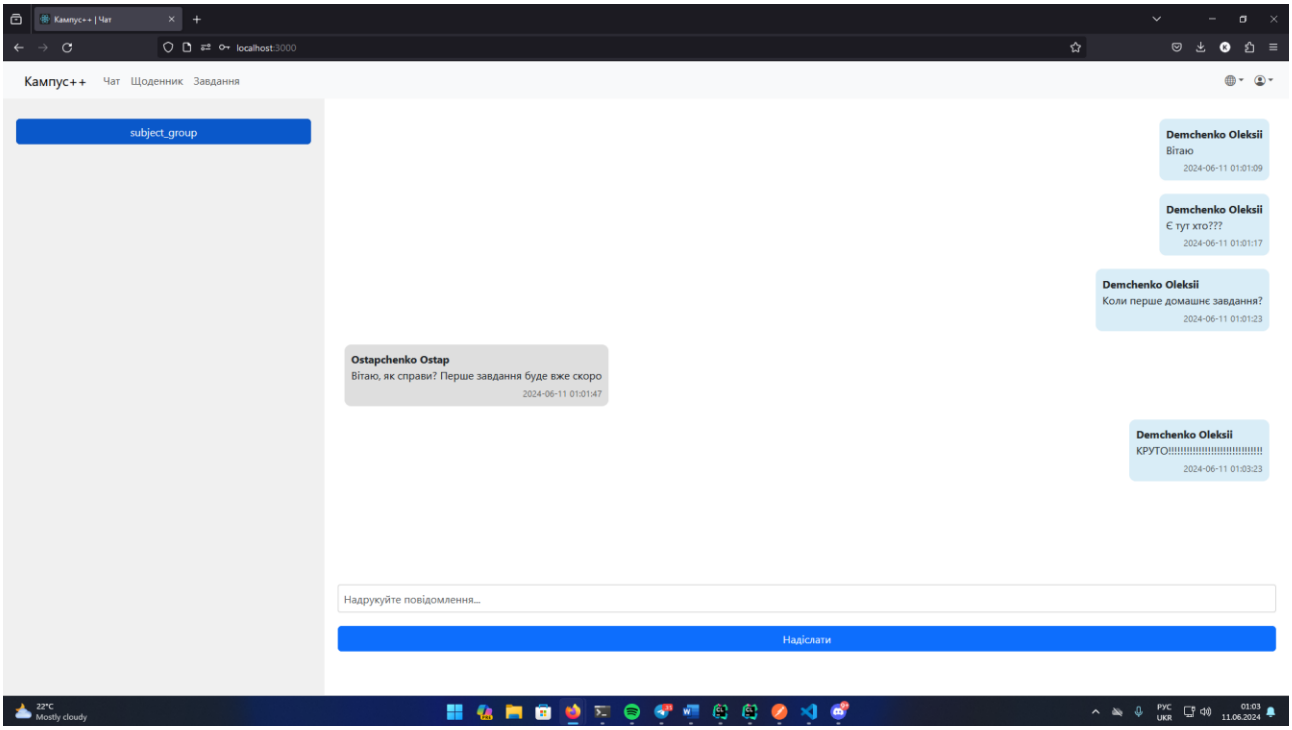


Рисунок 4.17 – Студент друкує повідомлення та натискає на кнопку надіслати

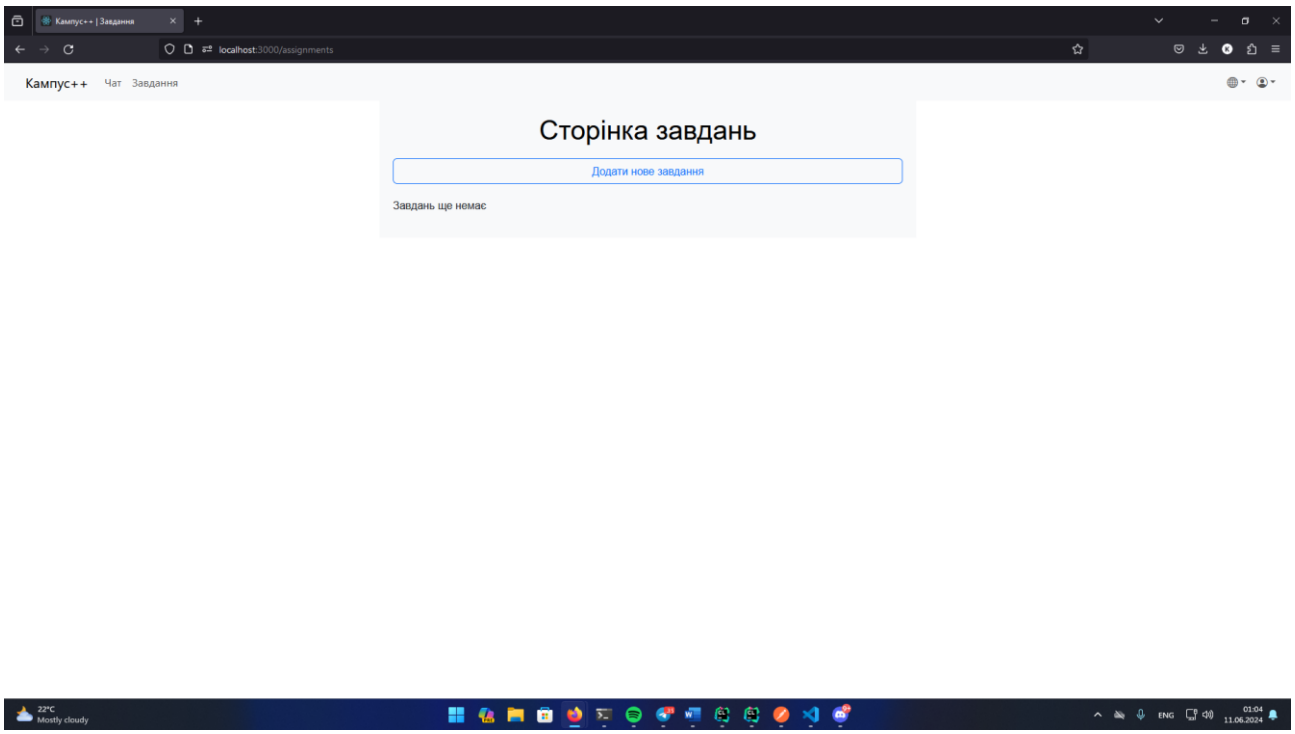


Рисунок 4.18 – Викладач відкриває сторінку завдань та бачить, що завдань ще немає

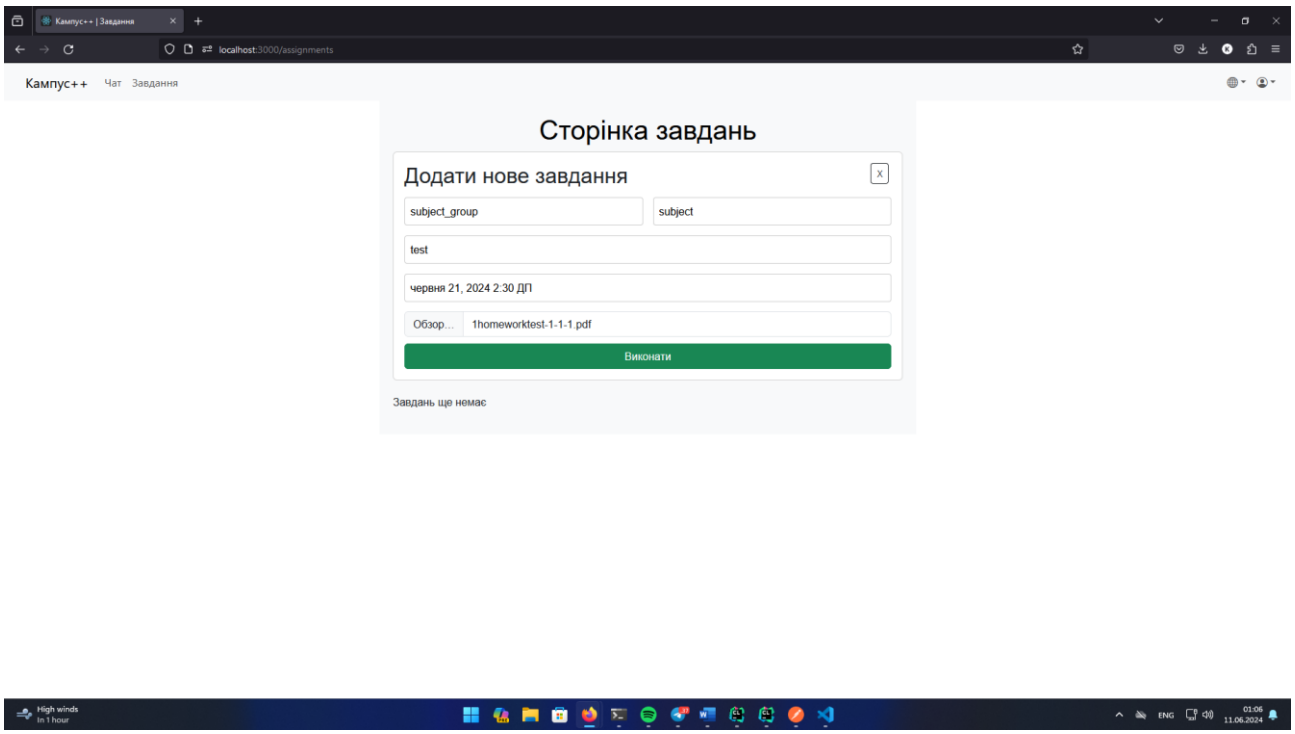


Рисунок 4.19 – Викладач заповнює форму завдання та натискає на кнопку «надіслати»

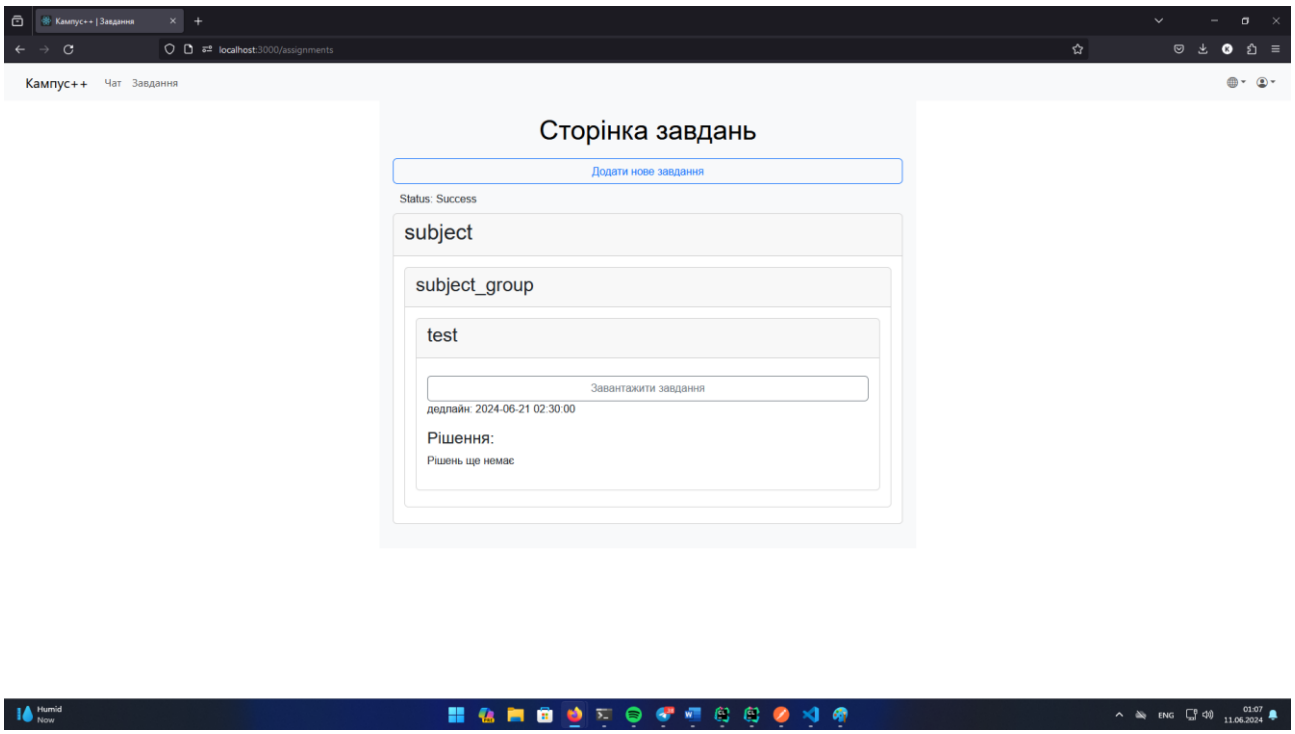


Рисунок 4.20 – Викладач бачить створене завдання

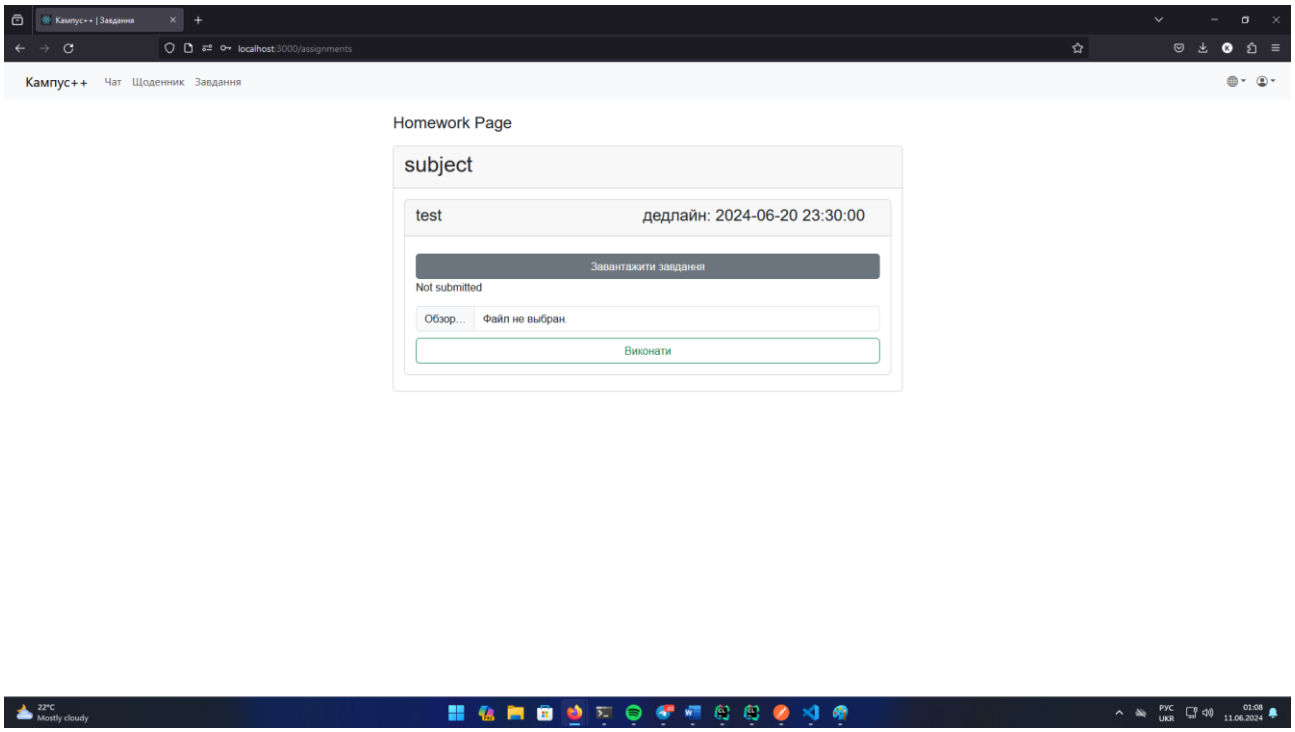


Рисунок 4.21 – Студент бачить на сторінці завдань нове завдання, прикріплене до своєї групи

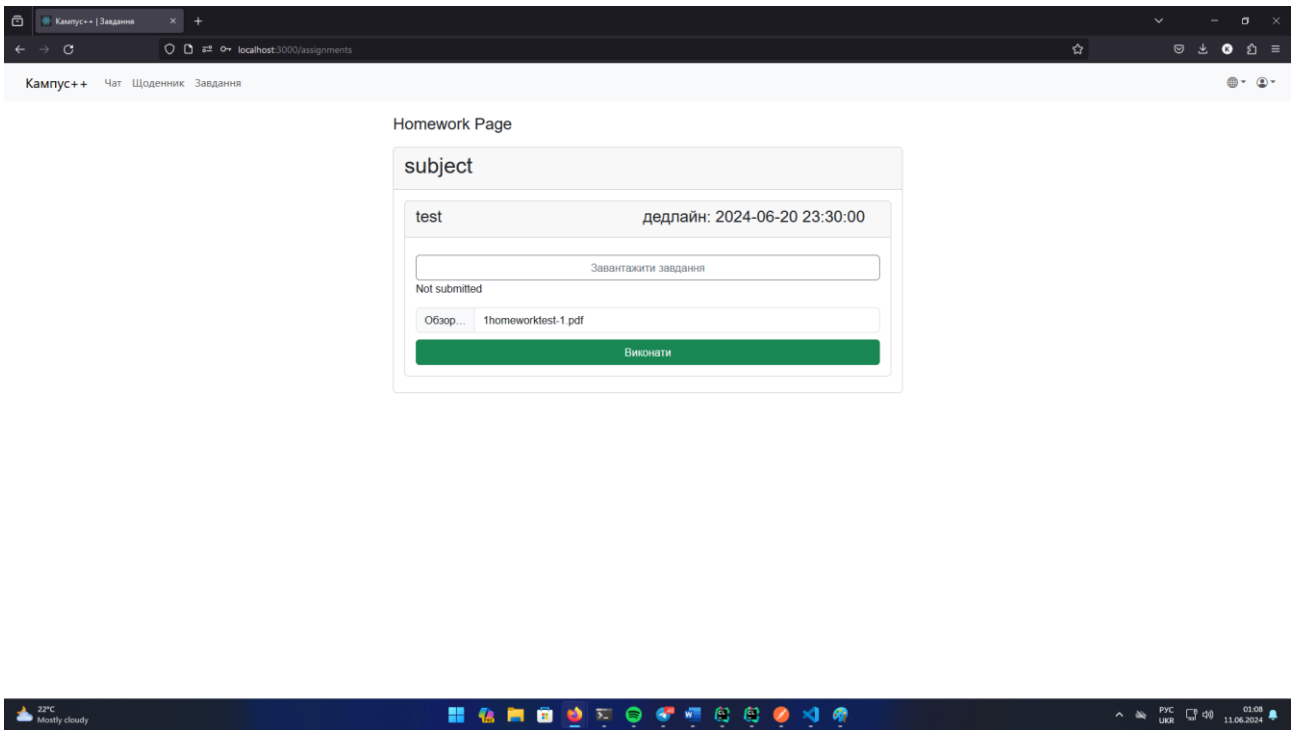


Рисунок 4.22 – Студент завантажує рішення до завдання (файл) та натискає «виконати»

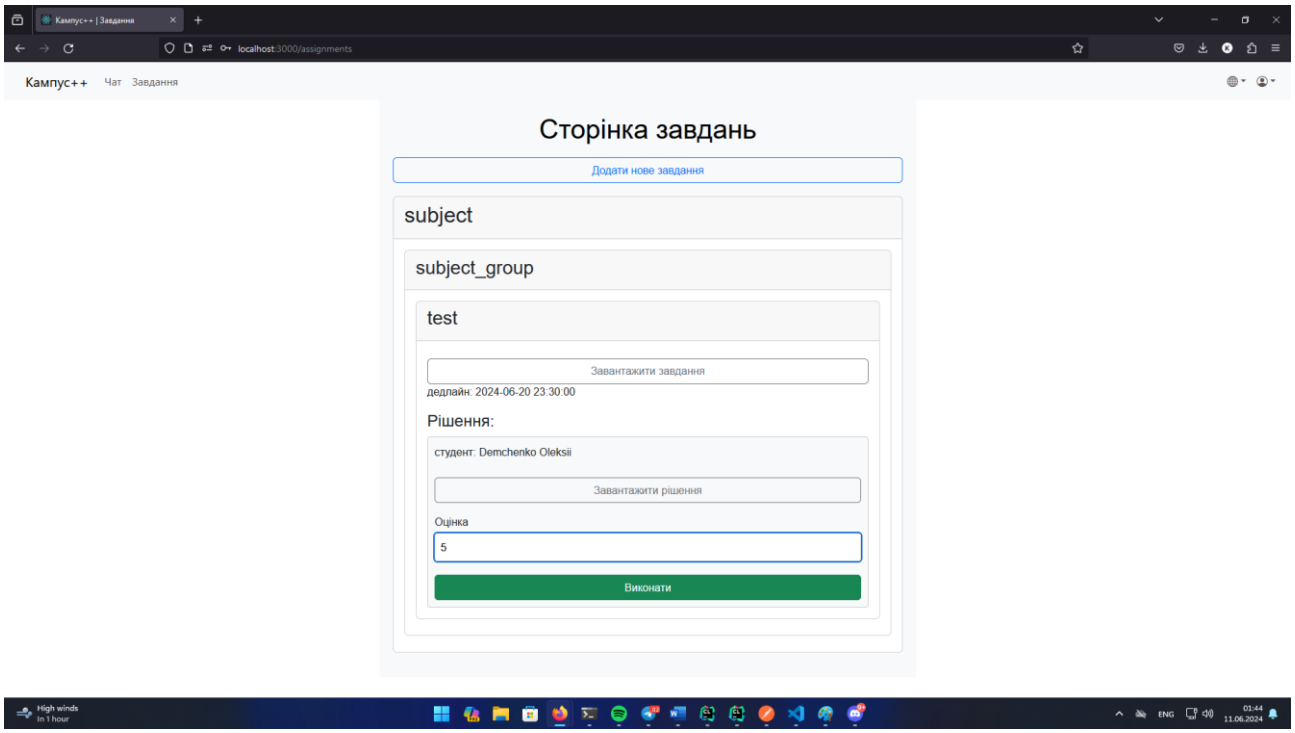


Рисунок 4.23 – Викладач виставляє оцінку до завдання студенту та натискає на кнопку «виконати»

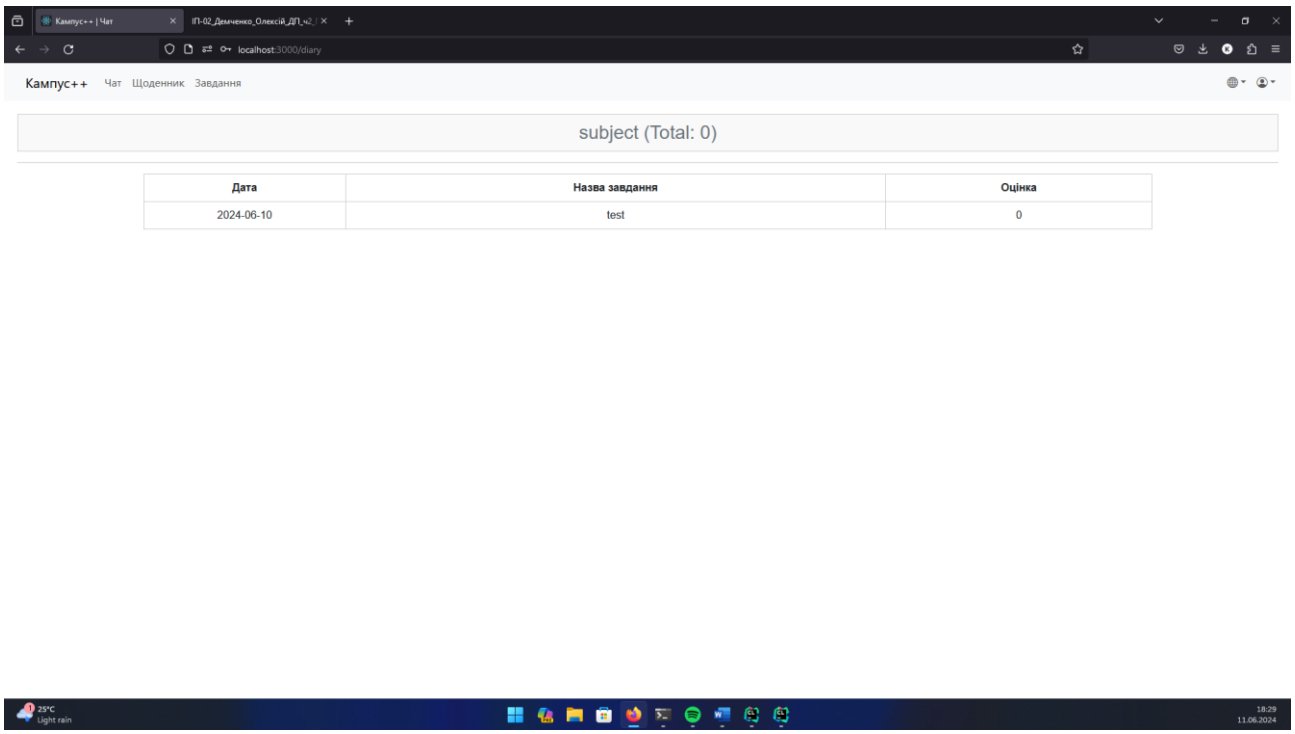


Рисунок 4.24 – Студент перейшов на сторінку щоденник через панель навігації та бачить виставлений бал за попереднє завдання

Висновки до розділу

Отже, під час виконання даного розділу був проведений аналіз якості програмного забезпечення. Для початку було порівняно декілька іструментів для цього, з яких був обраний Codacy через його кращі сторони. Завдяки даному хмарному сервісу було перевірено програмний код на складність, повторюваність та кількість попереджень. Перевірка пройшла успішно з мінімальними показниками та найбільшою оцінкою «А».

Далі, було проведено мануальне тестування програмного забезпечення, були описані всі тест-кейси, які мають покривати функціональні вимоги. Всі тести були пройдені успішно. А саме було перевірено аутентифікацію, єдиний модуль завдань, щоденник, адміністративну панель та чат.

В останньому підрозділі було описано контрольний приклад у вигляді рисунків зі сторони адміністратора, викладача та студента, щоб надати уявлення, як саме працює вебзастосунок.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Серед засобів розгортання був обраний Docker та його розширення Docker Compose.

Docker є надзвичайно потужним інструментом для розгортання додатків завдяки таким перевагам:

- Ізоляція середовища: Docker дозволяє створювати контейнери, які ізолюють додаток та його залежності від операційної системи хосту. Це означає, що додаток завжди буде працювати однаково, незалежно від середовища, в якому він запускається.
- Легкість міграції та масштабування: Завдяки контейнеризації, додатки можна легко переносити між різними середовищами (розробка, тестування, продакшн) без необхідності вносити зміни в їх код. Це значно полегшує масштабування додатків.
- Кросплатформенність: Docker дозволяє запускати додатки на будь-якій операційній системі, що підтримує Docker. Це означає, що немає потреби перевстановлювати операційну систему чи налаштовувати специфічні залежності для кожного середовища.
- Простота в налаштуванні та використанні: Створення Dockerfile та побудова образів є відносно простим процесом, який може автоматизувати багато рутинних завдань, пов'язаних з налаштуванням середовища [15].

Docker Compose також є важливим інструментом у арсеналі розробника та адміністратора систем:

- Оркестрація контейнерів: Docker Compose дозволяє легко керувати багатоконтейнерними додатками. За допомогою одного YAML файлу можна описати всі сервіси, які потрібні для роботи додатку (бази даних, кеш, веб-сервери тощо) та їх взаємодію.

- Автоматизація розгортання: З Docker Compose можна автоматизувати процес розгортання додатків. Один командний рядок (docker-compose up) запускає всі необхідні сервіси, налаштовує мережу та волюми.
- Управління конфігураціями: Docker Compose дозволяє зберігати конфігурації середовища в одному місці, що полегшує їх зміну та управління. Це зменшує ризик помилок та забезпечує консистентність середовищ розробки, тестування та продакшн.
- Спрощення тестування: За допомогою Docker Compose можна легко створювати тестові середовища, які повністю повторюють продакшн середовища. Це полегшує процес тестування та забезпечує високу якість програмного забезпечення [16].

Docker та Docker Compose є ключовими інструментами для сучасного розгортання та управління додатками, забезпечуючи ефективність, гнучкість та надійність процесів.

Для того, щоб розгорнути серверну частину достатньо завантажити папку з проєктом з репозиторію на Github. Зайти в папку та написати в термінал команду «make docker-start-release» (рисунок 5.1). Дана команда автоматично, використовуючи Docker, розгорне серверну частину, а саме, базу даних PostgreSQL, сам сервер та AWS S3, тобто все необхідне для роботи серверу. Даний метод стає доступним завдяки використанню програми make, що зберігає всі необхідні скрипти в одному місці, що дуже є зручним методом в наш час [17].

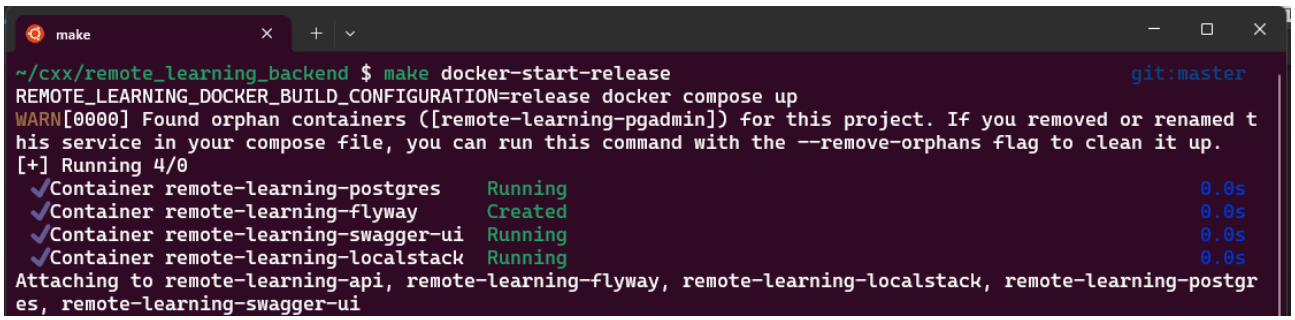


Рисунок 5.1 – Розгортання серверної частини за допомогою використання Docker Compose

Залишилось лише розгорнути клієнтську частину, для неї точно так само використовується Docker Compose. Для цього достатньо завантажити папку з репозиторієм собі на пристрій та запустити завдяки команді «docker compose build && docker compose up» (рисунок 5.2).

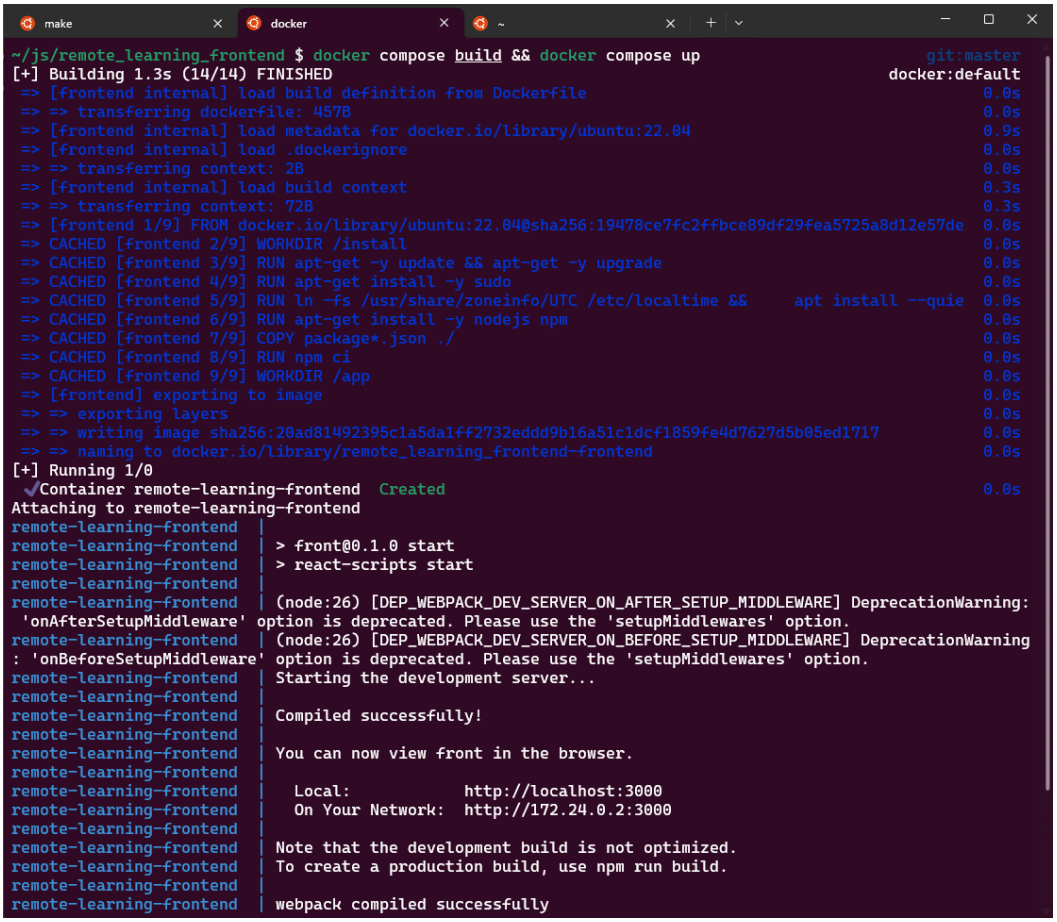


Рисунок 5.2 – Розгортання клієнтської частини

5.2 Супровід програмного забезпечення

Для супроводу програмного забезпечення використовується GitHub Actions. Даний інструмент дозволяє автоматично на кожну зміну виконувати

певну послідовність дій, яка задається в окремому файлі, що зазвичай лежить в проєкті. Інструмент реагує на кожну зміну в репозиторії (commit) та запускає дану послідовність (рисунок 5.3) [18]. В даному проєкті для перевірки, що програмний код є робочим: компілюється без помилок, був доданий скрипт, який намагається зкомпілювати проєкт за допомогою докеру. Якщо перевірки не будуть пройдені програміст не зможе додати свої зміни до головної гілки за допомогою pull request. Дана дія запобігає тому, щоб програміст випадково нічого не зламав. Дану процедуру можна виконати локально (рисунок 5.4) за допомогою команди «make docker-build-debug»: це й робить GitHub Actions автоматично на кожну зміну.

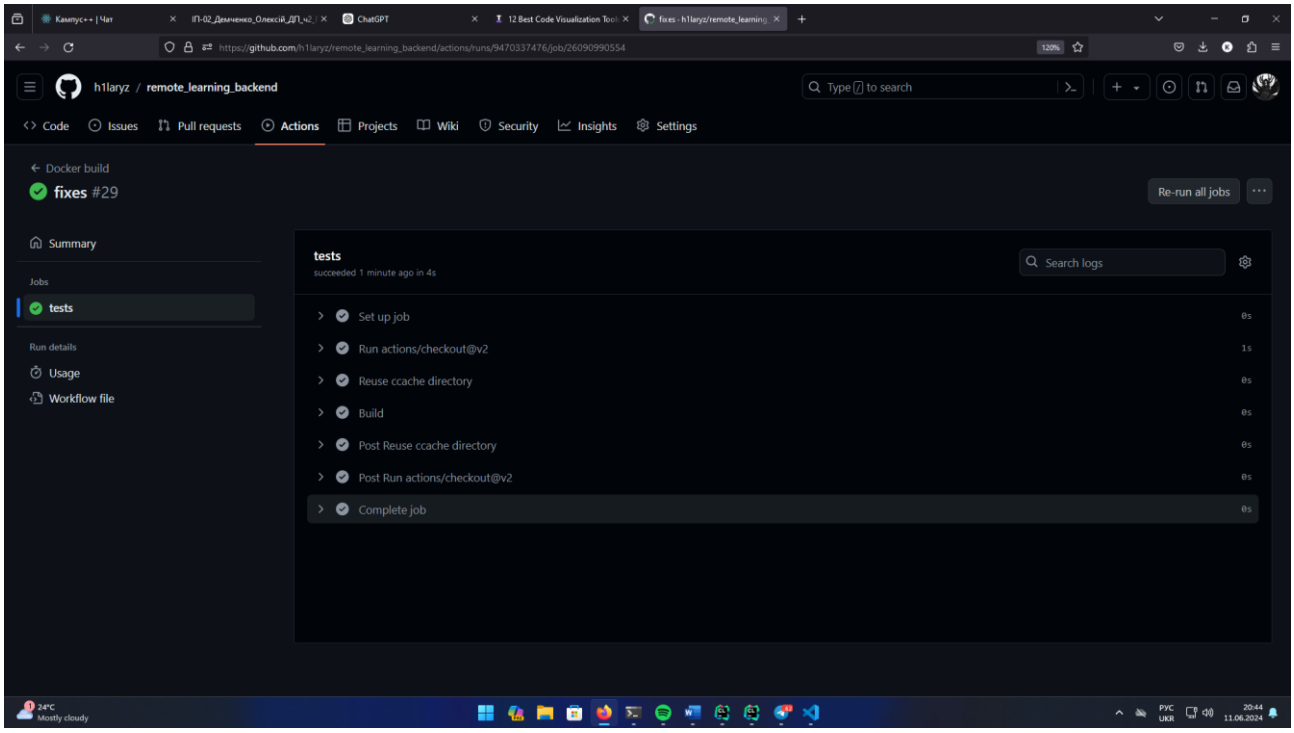


Рисунок 5.3 – Автоматичний запуск спроби скомпілювати проєкт на серверній частині, використовуючи GitHub Actions

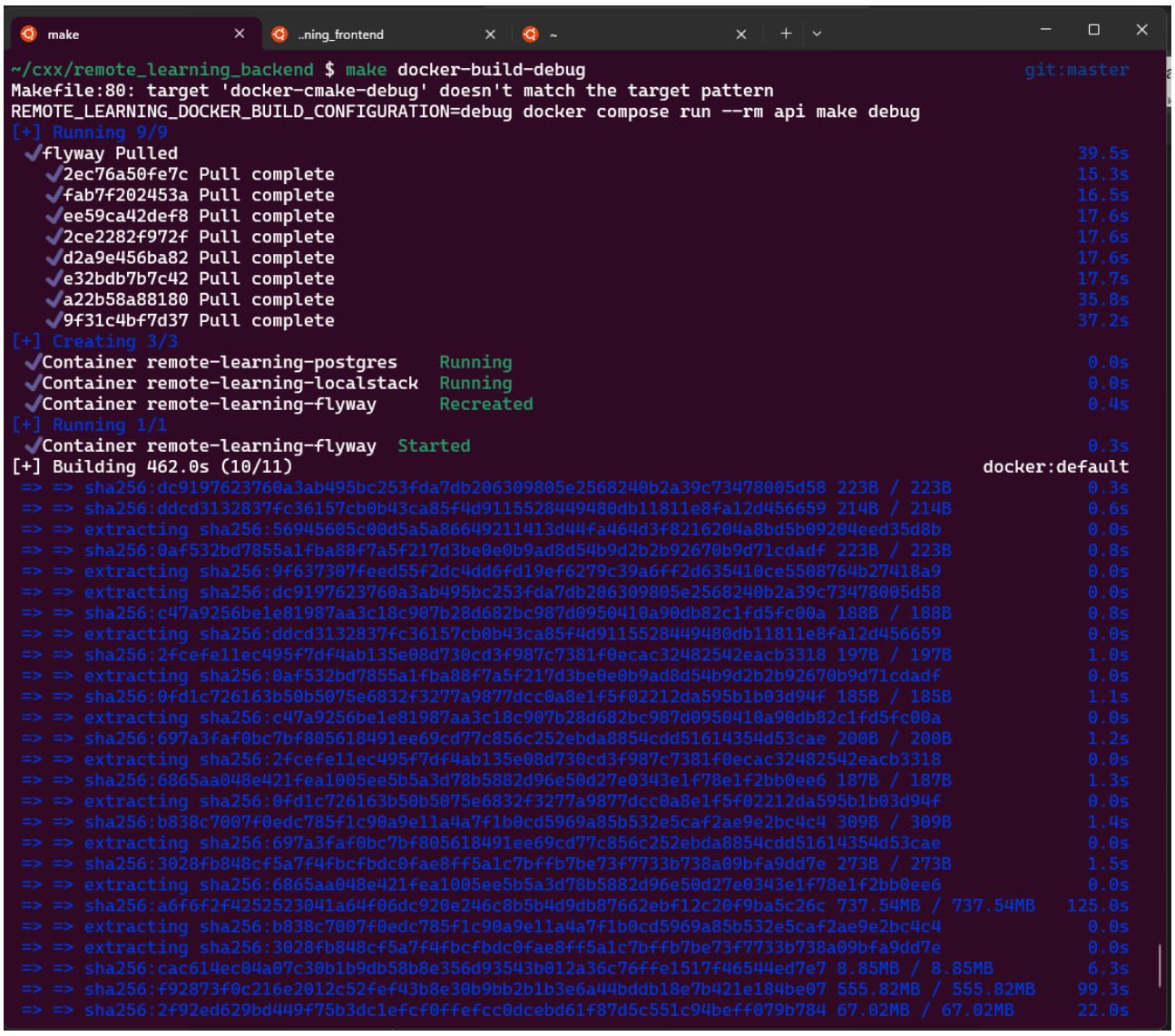


Рисунок 5.4 – Локальна спроба скомпілювати проєкт з нуля на серверній частині

Далі буде продемонстрована аналогічна дія з клієнтською частиною (рисунок 5.5). За допомгою GitHub Actions була реалізована спроба зібрати проєкт в докері. Бачимо успішний результат.

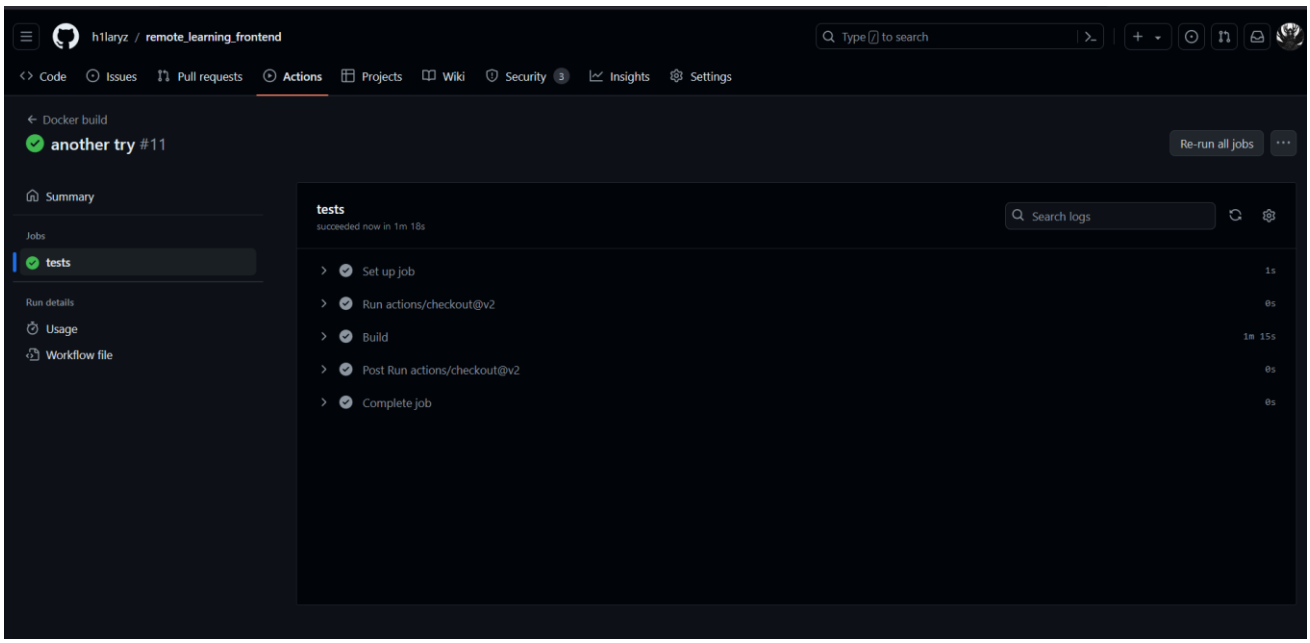


Рисунок 5.5 – GitHub Actions для клієнтської частини вебзастосунку

Висновки до розділу

Отже, в даному розділі був проведений детальний опис розгортання програмного забезпечення. Для цього була обрана найсучасніша технологія Docker та його доповнення Compose, завдяки якому можна за допомогою лише однієї команди автоматично все запустити, як на локальному пристрої так і не будь-якому сервері, де вже встановлена дана технологія. Дана процедура була продумана як для серверної частини, так й для клієнтської.

Для супроводу програмного забезпечення був обраний GitHub Actions, що автоматично на кожну зміну перевіряє чи не був випадково зламаний вебзастосунок, чи проходить він компіляцію та чи зможе він взагалі бути запущеним. Дана методологія була реалізована також за допомогою технології Docker.

Усі необхідні файли для розгортання та супроводу знаходяться у проєкті в репозиторії GitHub.

ВИСНОВКИ

У результаті виконання дипломного проєкту було реалізовано вебзастосунок для організації дистанційного навчання, що призначений для створення всіх необхідних ресурсів для навчання: чат зі студентами та викладачами, єдина система завдань та щоденник.

Для розробки була обрана клієнт-серверна архітектура для того, шоб забезпечити безпеку до даних шляхом обробки запитів через сервер. Розробка здійснювалась на двух мовах програмуваннях C++ та Javascript. Перша для серверної частини, а друга для клієнтської. Для забезпечення збереження даних була обрана БД PostgreSQL, а для збереження файлів домашніх завдань технологія AWS S3.

Після цього був проведений аналіз якості програмного забезпечення, а для цього був використаний інструмент Codacy, що продемонстрував якість, а саме: відсутність дублікацій, попереджень та складності в програмному коді. Було проведено мануальне тестування вебзастосунку, при якому не було виявлено жодних проблем.

Було забезпечене легке розгортання проєкту за допомогою технології Docker, що дозволяє на будь-якому локальному пристрої легко та доступно все запустити. Для супроводу програмного забезпечення був використаний GitHub Actions для перевірки на компіляцію коду, що забезпечить відсутність випадкових помилок у проєкті.

Розроблений вебзастосунок може бути використаний у будь-якому вищому навчальному закладі задля усунення всіх можливих організаційних питань та марного витрачення дорогоцінного часу викладачів і студентів. Даний застосунок надає гарний користувацький інтерфейс, що не викликає жодних питань з приводу організації. Також застосунок підтримує дві мови: українську та англійську, що дозволяє використовувати його не тільки в українському вищому навчальному закладі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Учасники проектів Вікімедіа. Вебзастосунок – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Вебзастосунок (дата звернення: 10.05.2024).

2) Про організацію роботи дистанційних класів. Управління освіти Чернігівської міської ради. URL: http://osvita.ch.ua/distanciyna_osvita/ (дата звернення: 04.05.2024).

3) «Google Classroom» – ідеальний ресурс для дистанційної роботи з класом. Освітній проект «На Урок» для вчителів. URL: https://naurok.com.ua/post/google-classroom-idealniy-resurs-dlya-distanciyno-roboti-z-klasom (дата звернення: 03.05.2024).

4) Освіта.ua. Організація дистанційного навчання в Moodle. Освіта.UA. URL: https://osvita.ua/vnz/high_school/72285/ (дата звернення: 03.05.2024).

5) Організація дистанційного навчання за допомогою Microsoft Teams – Інститут післядипломної освіти. Інститут післядипломної освіти – Навчання впродовж життя. URL: https://ipo.kpi.ua/microsoft-teams/ (дата звернення: 03.05.2024).

6) Використання сервісу Telegram за умов дистанційного навчання - Кузня. Кузня контенту. URL: https://content.hneu.edu.ua/s/2u6CTRcSD (дата звернення: 03.05.2024).

7) Fanchi C. Everything You Need To Know About Web App Development. Backendless. URL: https://backendless.com/everything-you-need-to-know-about-web-app-development/ (date of access: 03.05.2024).

8) Cross Browser Compatibility Score of HTTP/2 protocol. Next-Generation Mobile Apps and Cross Browser Testing Cloud | LambdaTest. URL: https://www.lambdatest.com/web-technologies/http2 (date of access: 03.05.2024).

9) PostgreSQL: the #1 open-source database with endless benefits – Fusion Infotech Limited. Fusion Infotech Limited – Fusion Infotech is primarily a Software Development & System Integration company. URL:

https://www.fusioninfotechltd.com/postgresql-open-source-database/ (date of access: 03.05.2024).

10) userver: Feature Comparison with other Frameworks. userver: userver. URL: https://userver.tech/da/d26/md_en_2userver_2framework__comparison.html (date of access: 03.05.2024).

11) What is SQL Injection? Tutorial & Examples | Web Security Academy. Web Application Security, Testing, & Scanning - PortSwigger. URL: https://portswigger.net/web-security/sql-injection (date of access: 03.05.2024).

12) A Brief Introduction To Clang-Tidy And Its Role in Visual Assist - Tomato Soup. Tomato Soup - Visual Assist Team Blog. URL: https://www.wholetomato.com/blog/2021/01/08/a-brief-introduction-to-clang-tidy-and-its-role-in-visual-assist/ (date of access: 03.05.2024).

13) Gupta S. ESLint: What, Why, When, How. DEV Community. URL: https://dev.to/shivambmgupta/eslint-what-why-when-how-5f1d (date of access: 03.05.2024).

14) Codacy. Everything you need to know about static code analysis. Medium. URL: https://medium.com/@codacy/everything-you-need-to-know-about-static-code-analysis-3e759db79658 (date of access: 03.05.2024).

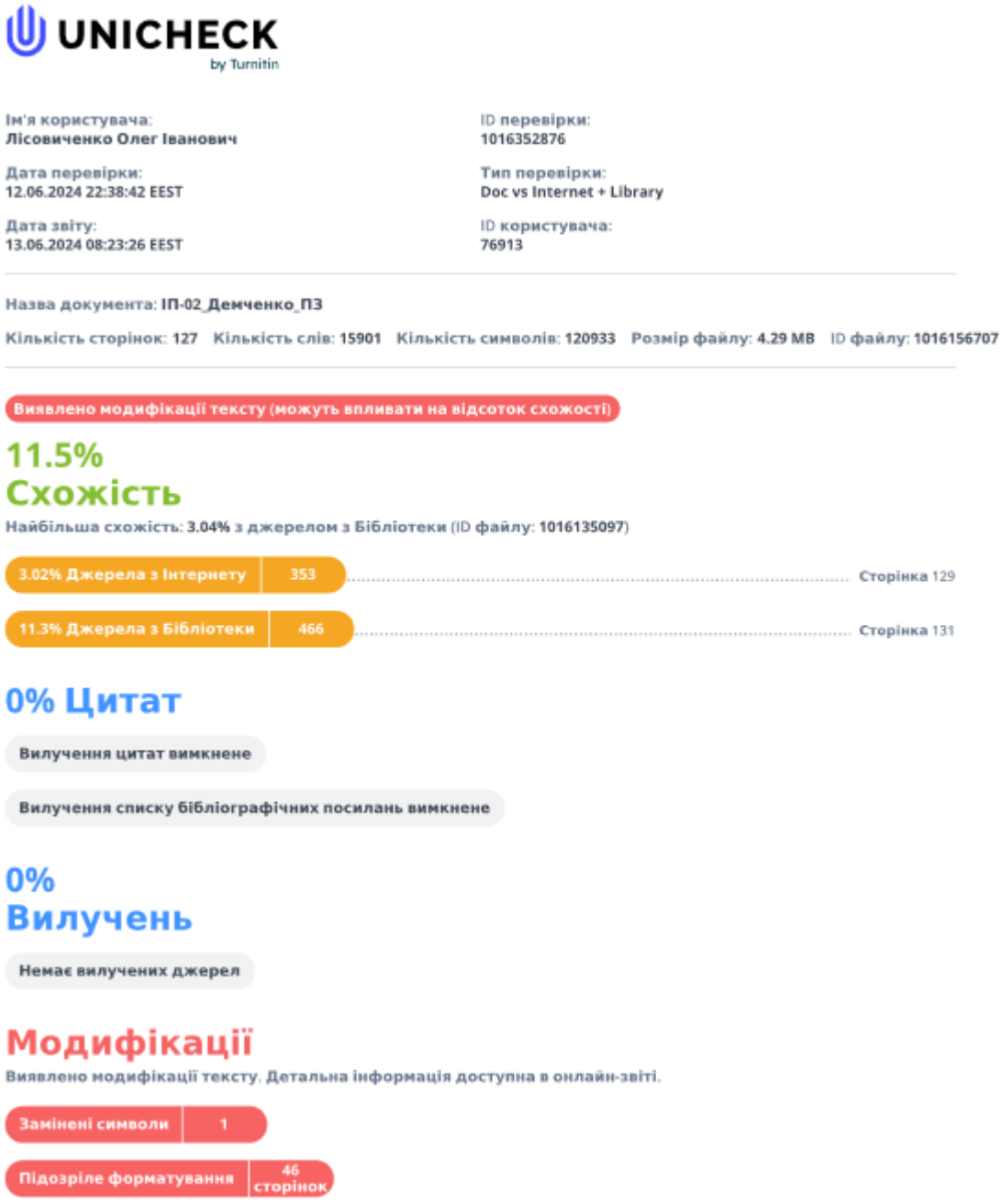
15) What Is Docker? | IBM. IBM - United States. URL: https://www.ibm.com/topics/docker (date of access: 03.05.2024).

16) B A. What is Docker Compose? How to Use it with an Example. freeCodeCamp.org. URL: https://www.freecodecamp.org/news/what-is-docker-compose-how-to-use-it/ (date of access: 03.05.2024).

17) What is a Makefile and how does it work?. Opensource.com. URL: https://opensource.com/article/18/8/what-how-makefile (date of access: 03.05.2024).

18) What is GitHub Actions? Automated CI/CD for GitHub. InfoWorld. URL: https://www.infoworld.com/article/3698188/what-is-github-actions-automated-cicd-for-github.html (date of access: 03.05.2024).

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



UNICHECK
by Turnitin

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1016352876

Дата перевірки:
12.06.2024 22:38:42 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
13.06.2024 08:23:26 EEST

ID користувача:
76913

Назва документа: ІП-02_Демченко_ПЗ

Кількість сторінок: 127 Кількість слів: 15901 Кількість символів: 120933 Розмір файлу: 4.29 MB ID файлу: 1016156707

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

11.5%
Схожість

Найбільша схожість: 3.04% з джерелом з Бібліотеки (ID файлу: 1016135097)

3.02% Джерела з Інтернету 353 ..... Сторінка 129

11.3% Джерела з Бібліотеки 466 ..... Сторінка 131

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 1

Підозріле форматування 46 сторінок

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНОГО НАВЧАННЯ

Текст програми

КПІ.ІП-0217.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Сергій ТИХОНОВ

Нормоконтроль:
_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:
_____ Олексій ДЕМЧЕНКО

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду
Серверна частина: https://github.com/h1laryz/remote_learning_backend
Клієнтська частина: https://github.com/h1laryz/remote_learning_frontend

Файл register.hpp та register.cpp SignUpPage.jsx SignUpFrom.jsx
Реалізація реєстрації користувача

```
import React from 'react';
import SignUpHeader from './Header/SignUpHeader';
import SignUpForm from './Form/SignUpForm';
import { Helmet } from 'react-helmet';
import { I18nextProvider, useTranslation } from 'react-i18next';
import Navigation from '../../navigation/Navigation';

const SignUpPage = () => {
  const { t } = useTranslation();

  return (
    <div>
      <Helmet>
        <title>{t("nameOfProject")} | {t("signUp")}</title>
      </Helmet>
      <Navigation />
      <SignUpHeader />
      <SignUpForm />
    </div>
  );
};

export default SignUpPage;

import React, { useState } from 'react';
import { useTranslation } from 'react-i18next';
import { Link } from 'react-router-dom';
import './SignUpForm.css';
import InputField from './InputField/InputField';
import DatePicker, { registerLocale } from "react-datepicker";
import enLocale from 'date-fns/locale/en-US';
import ukLocale from 'date-fns/locale/uk';
import "react-datepicker/dist/react-datepicker.css";
import RequiredLabel from './RequiredLabel/RequiredLabel';

registerLocale('en', enLocale);
registerLocale('uk', ukLocale);

function formatRFC3339Date(date) {
  const isoString = date.toISOString(); // Получаем строку в формате ISO8601

  if (isoString === null)
  {
    return isoString;
  }

  return isoString.split('T')[0]; // Обрезаем время и возвращаем только дату
}

const SignUpForm = () => {
  const { t, i18n } = useTranslation();
  const [dateOfBirth, setDateOfBirth] = useState(null);
  const [loading, setLoading] = useState(false);
  const [ error, setError ] = useState(null);

  const handleSubmit = async (event) => {
    event.preventDefault();

    // Set loading state to true
    setLoading(true);

    // Get form data
    const formData = new FormData(event.target);
    const surname = formData.get('surname');
    const lastName = formData.get('lastName');
    const middleName = formData.get('middleName');
    const username = formData.get('username');
```

```
  const email = formData.get('email');
  const password = formData.get('password');

  // Create request body
  const body = {
    surname,
    last_name: lastName,
    middle_name: middleName,
    date_of_birth: formatRFC3339Date(dateOfBirth),
    username,
    email,
    password,
  };

  try {
    // Make request
    const response = await fetch('http://localhost:8080/v1/register', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(body),
    });

    // Check if request was successful
    if (response.ok) {
      // Get the token from the response
      const token = await response.text();

      // Store the token in localStorage
      localStorage.setItem('token', token);

      // Redirect to the homepage
      window.location.href = '/';
    } else {
      // Get the error message from the response
      const errorResponse = await response.json();
      const errorMessage = errorResponse.error;

      // Translate the error message to the current language
      const translatedErrorMessage = t(`error.${errorMessage}`);

      // Set error message
      setError(translatedErrorMessage);

      // Set loading state to false
      setLoading(false);
    }
  } catch (error) {
    setError(t('error.serverNotResponding'));

    // Set loading state to false
    setLoading(false);
  }
};

return (
  <form className="form-container" onSubmit={handleSubmit}>
    <InputField label={t('surname')} required>
      <input type="text" name="surname" placeholder={t('surname')} />
    </InputField>
    <InputField label={t('lastName')} required>
      <input type="text" name="lastName" placeholder={t('lastName')} />
    </InputField>
    <InputField label={t('middleName')}>
      <input type="text" name="middleName" placeholder={t('middleName')} />
    </InputField>
    <InputField label={t('username')} required>
      <input type="text" name="username" placeholder={t('username')} />
    </InputField>
    <InputField label={t('email')} required>
      <input type="email" name="email" placeholder={t('email')} />
    </InputField>
    <InputField label={t('password')} required>
      <input type="password" name="password" placeholder={t('password')} />
    </InputField>
    <DatePicker
      selected={dateOfBirth}
      onChange={(date) => setDateOfBirth(date)}
      wrapperClassName="datePicker"
      dateFormat="dd/MM/yyyy"
```

```
        placeholderText={t('dateOfBirth')}
        locale={i18n.language}
      />
      <RequiredLabel />
      <button type="submit" disabled={loading}>{loading? 'Loading...' : t('signUp')}</button>
      {/* Add a Link to navigate to the login form */}
      <Link to="/login" className="form-link">{t('alreadyHaveAccount')} {t('login')}</Link>
      {/* Error message */}
      {error && <div className="error-message">{error}</div>}
    </form>
  );
};

export default SignUpForm;
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>
#include <userver/utils/regex.hpp>

namespace rl::handlers
{
class Register final : public userver::server::handlers::HttpHandlerBase
{
public:
    Register( const userver::components::ComponentConfig& config,
              const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-register";
    using HttpHandlerBase::HttpHandlerBase;

    std::string
    HandleRequestThrow( const userver::server::http::HttpRequest& request,
                        userver::server::request::RequestContext& request_context ) const
override;

    [[nodiscard]] bool registerImpl( std::string_view surname,
                                     std::string_view last_name,
                                     std::string_view middle_name,
                                     std::string_view password,
                                     std::string_view username,
                                     std::string_view email,
                                     std::string_view date_of_birth ) const;

private:
    [[nodiscard]] bool isUsernameFree( std::string_view username ) const;
    [[nodiscard]] bool isEmailFree( std::string_view email ) const;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers

 #include "api/handlers/register.hpp"

#include <jwt-cpp/jwt.h>
#include <userver/crypto/hash.hpp>
#include <userver/formats/json.hpp>

#include <magic_enum_all.hpp>

#include "api/handlers/errors/Error.h"
#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/handlers/validators/RegexValidator.hpp"

namespace
{
constexpr std::string_view kUsername{ "username" };
constexpr std::string_view kMail{ "email" };
constexpr std::string_view kPassword{ "password" };
constexpr std::string_view kDateOfBirth{ "date_of_birth" };
constexpr std::string_view kMiddleName{ "middle_name" };
constexpr std::string_view kSurname{ "surname" };
```

```
constexpr std::string_view kLastName{ "last_name" };
} // namespace

namespace rl::handlers
{
Register::Register( const userver::components::ComponentConfig& config,
                    const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string Register::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                          userver::server::request::RequestContext& ) const
{
    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kUsername, kPassword, kMail, kDateOfBirth, kLastName, kSurname } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& username{ request_body[ kUsername ].As< std::string >() };
    if ( !isUsernameFree( username ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "usernameAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    if ( !validators::RegexValidator::isValidUsername( username ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "usernameRegexError";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto& mail{ request_body[ kMail ].As< std::string >() };
    if ( !isEmailFree( mail ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "emailAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    if ( !validators::RegexValidator::isValidEmail( mail ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "emailRegexError";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto& password{ request_body[ kPassword ].As< std::string >() };
    if ( !validators::RegexValidator::isValidPassword( password ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "passwordRegexError";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto& date_of_birth{ request_body[ kDateOfBirth ].As< std::string >() };
    if ( !validators::RegexValidator::isValidDateOfBirth( date_of_birth ) )
    {
```

```
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "dateOfBirthRegexError";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto& last_name{ request_body[ kLastName ].As< std::string >() };
    if ( !validators::RegexValidator::isValidLastName( last_name ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "lastNameRegexError";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto& surname{ request_body[ kSurname ].As< std::string >() };
    if ( !validators::RegexValidator::isValidSurname( surname ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "surnameRegexError";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto& middle_name{ request_body[ kMiddleName ].As< std::string >( "" ) };
    if ( !middle_name.empty() && !validators::RegexValidator::isValidMiddleName( middle_name ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "middleNameRegexError";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    if ( !registerImpl( surname, last_name, middle_name, password, username, mail, date_of_birth
) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "internalServerError";

        request.SetResponseStatus( userver::server::http::HttpStatus::kInternalServerError );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "registerSuccess";
    return userver::formats::json::ToString( response.ExtractValue() );
}

bool Register::registerImpl( std::string_view surname,
                             std::string_view last_name,
                             std::string_view middle_name,
                             std::string_view password,
                             std::string_view username,
                             std::string_view mail,
                             std::string_view date_of_birth ) const
{
    if ( !middle_name.empty() )
    {
        const auto query{
            "INSERT INTO university.users(username, email, last_name, surname, middle_name, "
            "password, date_of_birth) "
            "VALUES ($1, $2, $3, $4, $5, $6, $7::DATE) "
            "RETURNING id;"
        };

        auto result{ pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                           query,
                                           username,
                                           mail,
                                           last_name,
                                           surname,
                                           middle_name,
                                           userver::crypto::hash::Sha512( password ),
                                           date_of_birth ) };

        if ( result.IsEmpty() )
        {
```

```
            return false;
        }

        return true;
    }

    const auto query{
        "INSERT INTO university.users(username, email, last_name, surname, password, "
        "date_of_birth) "
        "VALUES ($1, $2, $3, $4, $5, $6::DATE) "
        "RETURNING id;"
    };

    auto result{ pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                       query,
                                       username,
                                       mail,
                                       last_name,
                                       surname,
                                       userver::crypto::hash::Sha512( password ),
                                       date_of_birth ) };

    if ( result.IsEmpty() )
    {
        return false;
    }

    return true;
}

bool Register::isUsernameFree( std::string_view username ) const
{
    const auto query{
        "SELECT id FROM university.users "
        "WHERE username = $1"
    };

    auto result{ pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                       query,
                                       username ) };

    if ( result.IsEmpty() )
    {
        return true;
    }

    return false;
}

bool Register::isEmailFree( std::string_view email ) const
{
    const auto query{
        "SELECT id FROM university.users "
        "WHERE email = $1"
    };

    auto result{
        pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster, query, email
)
    };

    if ( result.IsEmpty() )
    {
        return true;
    }

    return false;
}
} // namespace rl::handlers
```

Файл login.hpp login.cpp LoginPage.jsx LoginForm.jsx

Реалізація функціональної вимоги Авторизація користувача

```
import React from 'react';
import { useNavigate } from 'react-router-dom'; // Импортируем useNavigate
import LoginHeader from './Header/LoginHeader';
import LoginForm from './Form/LoginForm';
import { Helmet } from 'react-helmet';
import { I18nextProvider, useTranslation } from 'react-i18next';
import Navigation from '../../navigation/Navigation';
```

```
const LoginPage = ({ onLogin }) => {
  const navigate = useNavigate(); // Инициализируем useNavigate

  const HandleLogin = (jwtToken) => {
    onLogin(jwtToken); // Вызываем функцию onLogin из App.js
    navigate('/'); // Перенаправляем на основную страницу
  };

  const { t } = useTranslation();

  return (
    <div>
      <Helmet>
        <title>{t("nameOfProject")} | {t("login")}</title>
      </Helmet>
      <Navigation/>
      <LoginHeader />
      <LoginForm onLogin={HandleLogin} /> {/* Передаем handleLogin в LoginForm */}
    </div>
  );
};

export default LoginPage;

import React, { useState } from 'react';
import { useTranslation } from 'react-i18next';
import { Link } from 'react-router-dom';
import axios from 'axios'; // Импортируем axios для отправки запросов
import './LoginForm.css';

const LoginForm = ({ onLogin }) => {
  const { t } = useTranslation();
  const [error, setError] = useState(null);

  const handleSubmit = async (event) => {
    event.preventDefault();
    const formData = new FormData(event.target);
    const data = {
      email_or_username: formData.get('email_or_username'),
      password: formData.get('password')
    };

    try {
      const response = await axios.post('http://localhost:8080/v1/login', data);
      const { token, error } = response.data;
      if (token) {
        onLogin(token); // Вызываем функцию onLogin из родительского компонента
      } else {
        setError(error); // Устанавливаем сообщение об ошибке
      }
    } catch (error) {
      setError('Something went wrong'); // Устанавливаем сообщение об ошибке, если запрос не
удался
    }
  };

  return (
    <form className="form-container" onSubmit={handleSubmit}>
      <input type="text" name="email_or_username" placeholder={t('usernameOrEmail')} />
      <input type="password" name="password" placeholder={t('password')} />
      <button type="submit">{t('login')}</button>
      {error && <div className="error-message">{error}</div>} {/* Отображаем сообщение об ошибке,
если оно есть */}
      <Link to="/signup" className="form-link">{t('dontHaveAccount')} {t('signUp')}</Link>
    </form>
  );
};

export default LoginForm;
 #pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>
```

```
namespace rl::handlers
{
class Login final : public userver::server::handlers::HttpHandlerBase
{
public:
    Login( const userver::components::ComponentConfig& config,
           const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-login";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

    std::optional< int > getUserIdViaDb( const std::string& emailOrUsername,
                                         const std::string& password ) const;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers

#include "api/handlers/login.hpp"

#include <jwt-cpp/jwt.h>
#include <userver/crypto/hash.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/result_set.hpp>

#include "api/handlers/validators/RegexValidator.hpp"
#include "api/pg/PgDao.hpp"

namespace
{
constexpr std::string_view kUsername{ "username" };
constexpr std::string_view kPassword{ "password" };
constexpr std::string_view kMail{ "email" };
constexpr std::string_view kEmailOrUsername{ "email_or_username" };
} // namespace

namespace rl::handlers
{
Login::Login( const userver::components::ComponentConfig& config,
              const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string Login::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                       userver::server::request::RequestContext& ) const
{
    if ( request.RequestBody().empty() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );

        userver::formats::json::ValueBuilder response;
        response[ "description" ] = fmt::format( R"(Body keys "{}" or "{}" and "{}" are
required)",
                                                 kMail,
                                                 kUsername,
                                                 kPassword );

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto& emailOrUsername{ request_body[ kEmailOrUsername ].As< std::string >( "" ) };
    const auto& password{ request_body[ kPassword ].As< std::string >( "" ) };

    if ( emailOrUsername.empty() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
```

```
        userver::formats::json::ValueBuilder response;
        response[ "description" ] =
            fmt::format( R"(Body key "{}" or "{}" is required)", kMail, kUsername );

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    if ( password.empty() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );

        userver::formats::json::ValueBuilder response;
        response[ "description" ] = fmt::format( R"(Body key "{}" is required)", kPassword );

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto opt_user_id{ getUserIdViaDb( emailOrUsername, password ) };

    if ( !opt_user_id.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "description" ] = fmt::format( R"(Invalid "{}" or "{}")", kUsername, kPassword
);

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto user_id = opt_user_id.value();

    const auto role = [ pg_dao, user_id ]() -> std::string
    {
        if ( pg_dao.isUserStudent( user_id ) )
        {
            return "student";
        }
        if ( pg_dao.isUserTeacher( user_id ) )
        {
            return "teacher";
        }
        if ( pg_dao.isUserAdmin( user_id ) )
        {
            return "admin";
        }
        return "";
    }();

    const auto token = [ pg_dao, user_id, role ]() -> std::string
    {
        const auto level = [ pg_dao, user_id, role ]() -> std::string
        {
            if ( role == "admin" )
            {
                return pg_dao.getAdminLevel( user_id ).AsSingleRow< std::string >();
            }
            return {};
        }();

        return jwt::create()
            .set_type( "JWT" )
            .set_id( "rsa-create-example" )
            .set_issued_now()
            .set_expires_in( std::chrono::seconds{ 36000 } )
            .set_payload_claim( "role", jwt::claim( role ) )
            .set_payload_claim( "level", jwt::claim( level ) )
            .set_payload_claim( "user_id", jwt::claim( std::to_string( user_id ) ) )
            .sign( jwt::algorithm::hs256( "secret" ) );
    }();

    const auto pg_query_save_token{
        "INSERT INTO auth.tokens(token, user_id) "
        "VALUES ($1, $2);"
    };

    const auto result{ pg_cluster_->Execute(
userver::storages::postgres::ClusterHostType::kMaster,
                                            pg_query_save_token,
```

```
                                            token,
                                            user_id ) };

    userver::formats::json::ValueBuilder response;
    response[ "token" ] = token;

    return userver::formats::json::ToString( response.ExtractValue() );
}

std::optional< int > Login::getUserIdViaDb( const std::string& emailOrUsername,
                                            const std::string& password ) const
{
    const auto pg_query =
        "SELECT id FROM university.users "
        "WHERE username = $1 OR email = $1"
        "AND password = $2;";

    const auto result{ pg_cluster_->Execute(
userver::storages::postgres::ClusterHostType::kMaster,
                                            pg_query,
                                            emailOrUsername,
                                            userver::crypto::hash::Sha512( password ) ) };

    if ( result.IsEmpty() )
    {
        return {};
    }

    return result.AsSingleRow< int >();
}
} // namespace rl::handlers
```

Файл subject_groups_get.hpp subject_groups_get.cpp

Реалізація функціональної вимоги Відображення списку чатів

```
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class SubjectGroupsGet final : public userver::server::handlers::HttpHandlerBase
{
public:
    SubjectGroupsGet( const userver::components::ComponentConfig& config,
                      const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-subject-groups-get";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers

#include "api/handlers/subject_groups_get.hpp"


#include <userver/http/common_headers.hpp>


#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"


#include <tuple>
```

```
namespace
{
using GroupRow = std::tuple< std::string >;
} // namespace

namespace rl::handlers
{
SubjectGroupsGet::SubjectGroupsGet( const userver::components::ComponentConfig& config,
                                    const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string SubjectGroupsGet::HandleRequestThrow( const userver::server::http::HttpRequest&
request,
                                                  userver::server::request::RequestContext& )
const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto user_id_result{ pg_dao.getUserIdViaJwt( jwt ) };
    if ( user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto user_id{ user_id_result.AsSingleRow< int >() };

    userver::formats::json::ValueBuilder response;

    if ( pg_dao.isUserStudent( user_id ) )
    {
        const auto subject_groups_result{ pg_dao.getSubjectGroupsForStudent( user_id ) };
        if ( subject_groups_result.IsEmpty() )
        {
            request.SetResponseStatus( userver::server::http::HttpStatus::kNoContent );
            return {};
        }

        const auto subject_groups{ subject_groups_result.AsSetOf< GroupRow >(
            userver::storages::postgres::kRowTag ) };

        for ( const auto subject_group : subject_groups )
        {
```

```
            const auto [ subject_name ] = subject_group;

            response.PushBack( subject_name );
        }
    }
    else if ( pg_dao.isUserTeacher( user_id ) )
    {
        const auto subject_groups_result{ pg_dao.getSubjectGroupsForTeacher( user_id ) };
        if ( subject_groups_result.IsEmpty() )
        {
            request.SetResponseStatus( userver::server::http::HttpStatus::kNoContent );
            return {};
        }

        const auto subject_groups{ subject_groups_result.AsSetOf< GroupRow >(
            userver::storages::postgres::kRowTag ) };

        for ( const auto subject_group : subject_groups )
        {
            const auto [ subject_name ] = subject_group;

            response.PushBack( subject_name );
        }
    }
    else
    {
        response[ "error" ] = "userIsNotATeacherOrStudent";

        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    return userver::formats::json::ToString( response.ExtractValue() );
}

} // namespace rl::handlers
```

Файл subject_group_messages_get.hpp subject_group_messages_get.cpp

Реалізація функціональної вимоги Відкриття бажаного чату

```
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
```

```
class SubjectGroupMessagesGet final : public userver::server::handlers::HttpHandlerBase
{
public:
    SubjectGroupMessagesGet( const userver::components::ComponentConfig& config,
                             const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-subject-group-messages-get";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/subject_group_messages_get.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <tuple>

namespace
{
constexpr std::string_view kSubjectGroupName{ "subject_group_name" };
const auto kSurname{ "surname" };
const auto kLastName{ "last_name" };
const auto kTimestamp{ "timestamp" };
const auto kContent{ "content" };
const auto kUserId{ "user_id" };
using MessageRow = std::tuple< std::string, std::string, std::string, std::string, int >;
} // namespace

namespace rl::handlers
{
SubjectGroupMessagesGet::SubjectGroupMessagesGet(
    const userver::components::ComponentConfig& config,
    const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string
SubjectGroupMessagesGet::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                             userver::server::request::RequestContext& ) const
{
```

```
const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

const auto& subject_group_name{ request.GetPathArg( kSubjectGroupName ) };
if ( subject_group_name.empty() )
{
    userver::formats::json::ValueBuilder response;
    response[ "error" ] = "requiredPathParamIsNotPassed";

    request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const pg::PgDao pg_dao{ pg_cluster_ };

const auto user_id_result{ pg_dao.getUserIdViaJwt( jwt ) };
if ( user_id_result.IsEmpty() )
{
    userver::formats::json::ValueBuilder response;
    response[ "error" ] = "userDoesntExist";

    request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const auto user_id{ user_id_result.AsSingleRow< int >() };

const auto subject_group_id_result{ pg_dao.getSubjectGroupId( subject_group_name ) };
if ( subject_group_id_result.IsEmpty() )
{
    userver::formats::json::ValueBuilder response;
    response[ "error" ] = "subjectGroupDoesntExist";

    request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const auto subject_group_id{ subject_group_id_result.AsSingleRow< int >() };

const auto subject_group_messages_result{ pg_dao.getMessagesForSubjectGroup(
    subject_group_id ) };
if ( subject_group_messages_result.IsEmpty() )
{
    request.SetResponseStatus( userver::server::http::HttpStatus::kNoContent );
    return {};
}

userver::formats::json::ValueBuilder response;

const auto subject_messages{ subject_group_messages_result.AsSetOf< MessageRow >(
    userver::storages::postgres::kRowTag ) };
for ( const auto message : subject_messages )
{
    const auto [ surname, last_name, timestamp, content, sender_user_id ] = message;
```

```
        userver::formats::json::ValueBuilder message_json;
        message_json[ kSurname ]   = surname;
        message_json[ kLastName ]  = last_name;
        message_json[ kTimestamp ] = timestamp;
        message_json[ kContent ]   = content;
        message_json[ kUserId ]    = sender_user_id;

        response.PushBack( std::move( message_json ) );
    }

    return userver::formats::json::ToString( response.ExtractValue() );
}

} // namespace rl::handlers
```

Файл message_send.hpp message_send.cpp

Реалізація функціональної вимоги Відправлення повідомлення

```
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class MessageSend final : public userver::server::handlers::HttpHandlerBase
{
public:
    MessageSend( const userver::components::ComponentConfig& config,
                 const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-subject-group-message-send";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/message_send.hpp"


#include <userver/http/common_headers.hpp>


#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"


namespace
{
std::string_view kSubjectGroupName{ "subject_group_name" };
std::string_view kContent{ "content" };
} // namespace


namespace rl::handlers
{
```

```
MessageSend::MessageSend( const userver::components::ComponentConfig& config,
                          const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string MessageSend::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                             userver::server::request::RequestContext& ) const
{
    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kSubjectGroupName, kContent } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    const auto& subject_Group_name{ request_body[ kSubjectGroupName ].As< std::string >() };
    const auto& content{ request_body[ kContent ].As< std::string >() };
    if ( content.empty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "contentIsEmpty";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto user_id_result{ pg_dao.getUserIdViaJwt( jwt ) };
    if ( user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto user_id{ user_id_result.AsSingleRow< int >() };

    const auto subject_group_id_result{ pg_dao.getSubjectGroupId( subject_Group_name ) };
    if ( subject_group_id_result.IsEmpty() )
```

```
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectGroupDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto subject_group_id{ subject_group_id_result.AsSingleRow< int >() };

    const auto send_message_result{ pg_dao.sendMessage( subject_group_id, user_id, content ) };
    if ( !send_message_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "internalServerError";

        request.SetResponseStatus( userver::server::http::HttpStatus::kInternalServerError );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}

} // namespace rl::handlers
```

Файл assignment_add.hpp assignment_add.cpp

Реалізація функціональної вимоги Авторизація користувача

```
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

#include "api/aws/s3.hpp"

namespace rl::handlers
{
class AssignmentAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    AssignmentAdd( const userver::components::ComponentConfig& config,
                   const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-assignment-add";
```

```
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
    aws::s3::S3Homework s3_homework_;
};
} // namespace rl::handlers
#include "api/handlers/assignment_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

namespace
{
constexpr std::string_view kAssignmentFile{ "assignment_file" };
constexpr std::string_view kDeadline{ "deadline" };
constexpr std::string_view kSubjectGroupName{ "subject_group_name" };
constexpr std::string_view kSubjectName{ "subject_name" };
constexpr std::string_view kAssignmentName{ "assignment_name" };
} // namespace

namespace rl::handlers
{
AssignmentAdd::AssignmentAdd( const userver::components::ComponentConfig& config,
                              const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string AssignmentAdd::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                               userver::server::request::RequestContext& ) const
{
    const auto content_type =
        userver::http::ContentType( request.GetHeader( userver::http::headers::kContentType ) );
    if ( content_type != "multipart/form-data" )
    {
        request.GetHttpResponse().SetStatus( userver::server::http::HttpStatus::kBadRequest );
        return "Expected 'multipart/form-data' content type";
    }

    validators::ParameterValidator::getErrorIfNotPassedFormDataParameters(
        request.FormDataArgNames(),
        { kAssignmentFile, kAssignmentName, kDeadline, kSubjectGroupName, kSubjectName } );

    const auto& assignment_file{ request.GetFormDataArg( kAssignmentFile ) };
```

```
    const auto& deadline{ request.GetFormDataArg( kDeadline ).value };
    const auto& subject_group_name{ request.GetFormDataArg( kSubjectGroupName ).value };
    const auto& subject_name{ request.GetFormDataArg( kSubjectName ).value };
    const auto& assignment_name{ request.GetFormDataArg( kAssignmentName ).value };

    if ( !assignment_file.filename.has_value() )
    {
        request.GetHttpResponse().SetStatus( userver::server::http::HttpStatus::kBadRequest );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "homeworkFileIsNotAFile";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto subject_group_id_result{ pg_dao.getSubjectGroupId( subject_group_name ) };
    if ( subject_group_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectGroupDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto subject_group_id{ subject_group_id_result.AsSingleRow< int >() };

    const auto subject_id_result{ pg_dao.getSubjectId( subject_name ) };
    if ( subject_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    // TODO: addAssignment transaction because if it puts into s3, but not into db what to do
next
    // ???????????????

    const auto s3_key_location{ s3_homework_.addAssignment( subject_name,
                                                            subject_group_name,
                                                            assignment_file.filename.value(),
                                                            assignment_file.value ) };

    if ( !s3_key_location.has_value() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "internalServerError";
```

```
        request.SetResponseStatus( userver::server::http::HttpStatus::kInternalServerError );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto subject_assignment_result{ pg_dao.addSubjectAssignment( subject_group_id,
                                                                       deadline,
                                                                       assignment_name,
                                                                       s3_key_location.value() )
};

    if ( !subject_assignment_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectAssignmentAlreadyExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ]      = "success";
    response[ "s3_location" ] = s3_key_location.value();
    return userver::formats::json::ToString( response.ExtractValue() );
}

} // namespace rl::handlers
```

Файл assignment_content_get.hpp assignment_content_get.cpp

Реалізація функціональної вимоги Завантаження існуючого завдання на пристрій

```
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

#include "api/aws/s3.hpp"

namespace rl::handlers
{
class AssignmentContentGet final : public userver::server::handlers::HttpHandlerBase
{
public:
    AssignmentContentGet( const userver::components::ComponentConfig& config,
                          const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-assignment-content-get";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
    aws::s3::S3Homework s3_homework_;
```

```
};
} // namespace rl::handlers

#include "api/handlers/assignment_content_get.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

namespace
{
constexpr std::string_view kS3Key{ "s3_key" };
} // namespace

namespace rl::handlers
{
AssignmentContentGet::AssignmentContentGet( const userver::components::ComponentConfig& config,
                                            const userver::components::ComponentContext& context
)
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string
AssignmentContentGet::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                          userver::server::request::RequestContext& ) const
{
    const auto& s3_key{ request.GetPathArg( kS3Key ) };
    if ( s3_key.empty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "requiredPathParamIsNotPassed";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto assignment{ s3_homework_.getAssignment( s3_key ) };
    if ( !assignment.has_value() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "assignmentDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "filename" ] = assignment->file_name;
    response[ "content" ]  = assignment->content;
    return userver::formats::json::ToString( response.ExtractValue() );
}

} // namespace rl::handlers
```

Файл assignment_solution_add.hpp assignment_solution_add.cpp

Реалізація функціональної вимоги Прикріплення рішення до завдання

```
#pragma once


#include <string>

#include <string_view>


#include <userver/components/component_config.hpp>

#include <userver/components/component_context.hpp>

#include <userver/server/handlers/http_handler_base.hpp>

#include <userver/server/http/http_request.hpp>

#include <userver/server/request/request_context.hpp>

#include <userver/storages/postgres/postgres.hpp>
```

```
#include "api/aws/s3.hpp"

namespace rl::handlers
{
class AssignmentSolutionAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    AssignmentSolutionAdd( const userver::components::ComponentConfig& config,
                           const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-assignment-solution-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
    aws::s3::S3Homework s3_homework_;
};
} // namespace rl::handlers
#include "api/handlers/assignment_solution_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

namespace
{
constexpr std::string_view kSubjectGroupName{ "subject_group_name" };
constexpr std::string_view kAssignmentName{ "assignment_name" };
constexpr std::string_view kAssignmentSolutionFile{ "assignment_solution_file" };
using SubjectResultRow = std::tuple< int, std::string >;
} // namespace

namespace rl::handlers
{
AssignmentSolutionAdd::AssignmentSolutionAdd( const userver::components::ComponentConfig& config,
                                              const userver::components::ComponentContext&
context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string
AssignmentSolutionAdd::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                           userver::server::request::RequestContext& ) const
{
```

```
const auto content_type =
    userver::http::ContentType( request.GetHeader( userver::http::headers::kContentType ) );
if ( content_type != "multipart/form-data" )
{
    request.GetHttpResponse().SetStatus( userver::server::http::HttpStatus::kBadRequest );
    return "Expected 'multipart/form-data' content type";
}

validators::ParameterValidator::getErrorIfNotPassedFormDataParameters(
    request.FormDataArgNames(),
    { kAssignmentSolutionFile, kAssignmentName, kSubjectGroupName } );

const auto& assignment_solution_file{ request.GetFormDataArg( kAssignmentSolutionFile ) };
const auto& subject_group_name{ request.GetFormDataArg( kSubjectGroupName ).value };
const auto& assignment_name{ request.GetFormDataArg( kAssignmentName ).value };
const auto& jwt{ request.GetHeader(
    userver::http::headers::kAuthorization ) }; // TODO: split bearer

if ( !assignment_solution_file.filename.has_value() )
{
    request.GetHttpResponse().SetStatus( userver::server::http::HttpStatus::kBadRequest );

    userver::formats::json::ValueBuilder response;
    response[ "error" ] = "homeworkFileIsNotAFile";

    request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const pg::PgDao pg_dao{ pg_cluster_ };

const auto user_id_result{ pg_dao.getUserIdViaJwt( jwt ) };
if ( user_id_result.IsEmpty() )
{
    userver::formats::json::ValueBuilder response;
    response[ "error" ] = "jwtIsNotValid";

    request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const auto user_id{ user_id_result.AsSingleRow< int >() };

if ( !pg_dao.isUserStudent( user_id ) )
{
    userver::formats::json::ValueBuilder response;
    response[ "error" ] = "userIsNotAStudent";

    request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const auto subject_group_id_result{ pg_dao.getSubjectGroupId( subject_group_name ) };
if ( subject_group_id_result.IsEmpty() )
```

```
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectGroupDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto subject_group_id{ subject_group_id_result.AsSingleRow< int >() };

    const auto subject_result{ pg_dao.getSubjectBySubjectGroup( subject_group_name ) };
    if ( subject_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto [ subject_id, subject_name ] =
        subject_result.AsSingleRow< SubjectResultRow >( userver::storages::postgres::kRowTag );

    // TODO: addAssignment transaction because if it puts into s3, but not into db what to do
next

    const auto assignment_id_result{ pg_dao.getAssignmentId( subject_group_id, assignment_name )
};
    if ( assignment_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "assignmentDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto assignment_id{ assignment_id_result.AsSingleRow< int >() };

    const auto s3_key_location{ s3_homework_.addAssignmentSolution(
        subject_name,
        subject_group_name,
        assignment_solution_file.filename.value(),
        assignment_solution_file.value,
        user_id ) };

    if ( !s3_key_location.has_value() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "internalServerError";

        request.SetResponseStatus( userver::server::http::HttpStatus::kInternalServerError );
        return userver::formats::json::ToString( response.ExtractValue() );
    }
```

```
    const auto subject_assignment_result{
        pg_dao.addAssignmentSolution( user_id, assignment_id, s3_key_location.value() )
    };

    if ( !subject_assignment_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectAssignmentSolutionAlreadyExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ]      = "success";
    response[ "s3_location" ] = s3_key_location.value();
    return userver::formats::json::ToString( response.ExtractValue() );
}

} // namespace rl::handlers
```

Файл assignment_solution_content_get.hpp assignment_solution_content_get.cpp

Реалізація функціональної вимоги Завантаження існуючого рішення завдання на пристрій

```
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

#include "api/aws/s3.hpp"

namespace rl::handlers
{
class AssignmentSolutionContentGet final : public userver::server::handlers::HttpHandlerBase
{
public:
    AssignmentSolutionContentGet( const userver::components::ComponentConfig& config,
                                  const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-assignment-solution-content-get";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
    aws::s3::S3Homework s3_homework_;
};
} // namespace rl::handlers

#include "api/handlers/assignment_solution_content_get.hpp"

#include <userver/http/common_headers.hpp>
```

```
#include "api/handlers/validators/ParameterValidator.hpp"

namespace
{
constexpr std::string_view kS3Key{ "s3_key" };
} // namespace

namespace rl::handlers
{
AssignmentSolutionContentGet::AssignmentSolutionContentGet(
    const userver::components::ComponentConfig& config,
    const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string
AssignmentSolutionContentGet::HandleRequestThrow( const userver::server::http::HttpRequest&
request,
                                                  userver::server::request::RequestContext& )
const
{
    const auto& s3_key{ request.GetPathArg( kS3Key ) };
    if ( s3_key.empty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "requiredPathParamIsNotPassed";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto assignment{ s3_homework_.getAssignmentSolution( s3_key ) };
    if ( !assignment.has_value() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "assignmentSolutionDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "filename" ] = assignment->file_name;
    response[ "content" ]  = assignment->content;
    return userver::formats::json::ToString( response.ExtractValue() );
}

} // namespace rl::handlers
```

Файл assignment_mark_add.hpp assignment_mark_add.cpp

Реалізація оцінення виконаного завдання

```
#pragma once


#include <string>

#include <string_view>


#include <userver/components/component_config.hpp>

#include <userver/components/component_context.hpp>

#include <userver/server/handlers/http_handler_base.hpp>

#include <userver/server/http/http_request.hpp>

#include <userver/server/request/request_context.hpp>

#include <userver/storages/postgres/postgres.hpp>


#include "api/aws/s3.hpp"


namespace rl::handlers

{
```

```
class AssignmentMarkAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    AssignmentMarkAdd( const userver::components::ComponentConfig& config,
                       const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-assignment-mark-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
    aws::s3::S3Homework s3_homework_;
};
} // namespace rl::handlers
#include "api/handlers/assignment_mark_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

namespace
{
constexpr std::string_view kMark{ "mark" };
constexpr std::string_view kS3Location{ "s3_location" };
} // namespace

namespace rl::handlers
{
AssignmentMarkAdd::AssignmentMarkAdd( const userver::components::ComponentConfig& config,
                                      const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string
AssignmentMarkAdd::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                       userver::server::request::RequestContext& ) const
{
    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters( request_body,
                                                                           { kS3Location, kMark }
)
    };
```

```
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& s3_location{ request_body[ kS3Location ].As< std::string >() };
    const auto& mark{ request_body[ kMark ].As< int >() };

    if ( mark > 100 || mark < 0 )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "markBadRange";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto insert_mark_result{ pg_dao.setMarkForSolutionByS3Key( s3_location, mark ) };
    if ( !insert_mark_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "markAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}

} // namespace rl::handlers
```

Файл student_assignments_get.hpp student_assignments_get.cpp teacher_assignments_get.hpp teacher_assignments_get.cpp

Реалізація функціональної вимоги Перегляд оцінки за виконане завдання

```
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

#include "api/aws/s3.hpp"

namespace rl::handlers
{
class StudentAssignmentsGet final : public userver::server::handlers::HttpHandlerBase
{
public:
```

```
    StudentAssignmentsGet( const userver::components::ComponentConfig& config,
                           const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-student-assignments-get";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
    aws::s3::S3Homework s3_homework_;
};
} // namespace rl::handlers
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

#include "api/aws/s3.hpp"

namespace rl::handlers
{
class TeacherAssignmentsGet final : public userver::server::handlers::HttpHandlerBase
{
public:
    TeacherAssignmentsGet( const userver::components::ComponentConfig& config,
                           const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-teacher-assignments-get";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
    aws::s3::S3Homework s3_homework_;
};
} // namespace rl::handlers
#include "api/handlers/teacher_assignments_get.hpp"

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <userver/http/common_headers.hpp>

namespace
{
constexpr std::string_view kJwt{ "jwt" };
constexpr auto kSurname{ "surname" };
constexpr auto kLastName{ "last_name" };
constexpr auto kMiddleName{ "middle_name" };
constexpr auto kUsername{ "username" };
using AssignmentsSet =
    std::tuple< int, std::string, std::string, std::string, std::string, std::string >;
using SolutionsSet = std::
    tuple< std::string, std::string, std::string, std::string, std::string, std::optional< int >
>;
constexpr auto kAssignmentName{ "assignment_name" };
constexpr auto kMark{ "mark" };
constexpr auto kS3Location{ "s3_location" };
constexpr auto kDeadline{ "deadline" };
constexpr auto kSolutions{ "solutions" };
constexpr auto kSubjectName{ "subject_name" };
constexpr auto kSubjectGroupName{ "subject_group_name" };
} // namespace

namespace rl::handlers
{
```

```
TeacherAssignmentsGet::TeacherAssignmentsGet( const userver::components::ComponentConfig& config,
                                              const userver::components::ComponentContext&
context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string
TeacherAssignmentsGet::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                           userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader(
        userver::http::headers::kAuthorization ) }; // TODO: split bearer

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto user_id_result{ pg_dao.getUserIdViaJwt( jwt ) };
    if ( user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "jwtIsNotValidDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto user_id{ user_id_result.AsSingleRow< int >() };

    const auto teacher_assignments_result{ pg_dao.getTeacherAssignments( user_id ) };
    if ( teacher_assignments_result.IsEmpty() )
    {
        return {};
    }

    const auto teacher_assignments{ teacher_assignments_result.AsSetOf< AssignmentsSet >(
        userver::storages::postgres::kRowTag ) };

    userver::formats::json::ValueBuilder response;
    for ( const auto& assignment : teacher_assignments )
    {
        const auto [ assignment_id,
                     assignment_name,
                     s3_key_location,
                     deadline,
                     subject_name,
                     subject_group_name ] = assignment;

        userver::formats::json::ValueBuilder assignment_json;
        assignment_json[ kAssignmentName ]   = assignment_name;
        assignment_json[ kS3Location ]       = s3_key_location;
        assignment_json[ kDeadline ]         = deadline;
        assignment_json[ kSubjectName ]      = subject_name;
        assignment_json[ kSubjectGroupName ] = subject_group_name;

        const auto solutions_for_assignment_result{ pg_dao.getAllSolutionsForAssignment(
            assignment_id ) };
        if ( !solutions_for_assignment_result.IsEmpty() )
        {
            const auto solutions_for_assignment{
                solutions_for_assignment_result.AsSetOf< SolutionsSet >(
                    userver::storages::postgres::kRowTag )
            };
            for ( const auto& solution : solutions_for_assignment )
            {
                const auto [ student_surname,
                             student_last_name,
                             student_middle_name,
                             student_username,
                             student_s3_key,
                             mark ] = solution;

                userver::formats::json::ValueBuilder solution_json;

                solution_json[ kSurname ]    = student_surname;
                solution_json[ kLastName ]   = student_last_name;
                solution_json[ kMiddleName ] = student_middle_name;
                solution_json[ kUsername ]   = student_username;
                solution_json[ kS3Location ] = student_s3_key;
```

```
                if ( mark.has_value() )
                {
                    solution_json[ kMark ] = mark.value();
                }

                assignment_json[ kSolutions ].PushBack( std::move( solution_json ) );
            }
        }

        response.PushBack( std::move( assignment_json ) );
    }

    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
#include "api/handlers/student_assignments_get.hpp"

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <userver/http/common_headers.hpp>

namespace
{
constexpr std::string_view kJwt{ "jwt" };
using AssignmentsSet =
    std::tuple< int, std::string, std::string, std::string, std::string, std::string >;
using AssignmentSolutionRow = std::tuple< std::string, std::optional< int >, std::string >;
constexpr auto kAssignmentName{ "assignment_name" };
constexpr auto kMark{ "mark" };
constexpr auto kS3Location{ "s3_location" };
constexpr auto kDeadline{ "deadline" };
constexpr auto kSubjectName{ "subject_name" };
constexpr auto kSubjectGroupName{ "subject_group_name" };
} // namespace

namespace rl::handlers
{
StudentAssignmentsGet::StudentAssignmentsGet( const userver::components::ComponentConfig& config,
                                              const userver::components::ComponentContext&
context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string
StudentAssignmentsGet::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                           userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader(
        userver::http::headers::kAuthorization ) }; // TODO: split bearer

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto user_id_result{ pg_dao.getUserIdViaJwt( jwt ) };
    if ( user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "jwtIsNotValidDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto user_id{ user_id_result.AsSingleRow< int >() };

    const auto student_assignments_result{ pg_dao.getStudentAssignments( user_id ) };
    if ( student_assignments_result.IsEmpty() )
    {
        return {};
    }

    const auto student_assignments{ student_assignments_result.AsSetOf< AssignmentsSet >(
        userver::storages::postgres::kRowTag ) };

    userver::formats::json::ValueBuilder response;
    for ( const auto item : student_assignments )
```

```
    {
        const auto [ assignment_id,
                     assignment_name,
                     s3_key_location,
                     deadline,
                     subject_name,
                     subject_group_name ] = item;

        userver::formats::json::ValueBuilder json_obj;
        json_obj[ kAssignmentName ]   = assignment_name;
        json_obj[ kS3Location ]       = s3_key_location;
        json_obj[ kDeadline ]         = deadline;
        json_obj[ kSubjectName ]      = subject_name;
        json_obj[ kSubjectGroupName ] = subject_group_name;

        const auto student_assignment_solution_result{
            pg_dao.getStudentAssignmentSolution( user_id, assignment_id )
        };
        if ( !student_assignment_solution_result.IsEmpty() )
        {
            const auto [ solution_s3_key, mark, assignment_date_time ] =
                student_assignment_solution_result.AsSingleRow< AssignmentSolutionRow >(
                    userver::storages::postgres::kRowTag );
            json_obj[ "solution" ][ kS3Location ] = solution_s3_key;
            json_obj[ "assignment_date" ]         = assignment_date_time;

            if ( mark.has_value() )
            {
                json_obj[ "solution" ][ kMark ] = mark.value();
            }
        }

        response.PushBack( std::move( json_obj ) );
    }

    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл student_diary_get.hpp student_diary_get.cpp StudentDiary.jsx

Реалізація функціональної вимоги Перегляд списку завдань та оцінок, Перегляд суми оцінок з кожного предмету окремо

```
import React, { useState, useEffect } from 'react';
import Navigation from '../navigation/Navigation';
import './StudentDiary.css';
import { t } from 'i18next';

const StudentDiary = () => {
  const [subjects, setSubjects] = useState({});
  const [expandedSubjects, setExpandedSubjects] = useState({});

  useEffect(() => {
    fetchDiaryData();
  }, []);

  const fetchDiaryData = async () => {
    try {
      const response = await fetch('http://localhost:8080/v1/student/diary', {
        method: 'GET',
        headers: {
          'Authorization': localStorage.getItem('jwtToken')
        }
      });
      if (!response.ok) {
        throw new Error('Failed to fetch data');
      }
      const data = await response.json();
      setSubjects(data);
      // Устанавливаем изначально все предметы раскрытыми
      setExpandedSubjects(Object.fromEntries(Object.keys(data).map(subject => [subject, true])));
    } catch (error) {
      console.error('Error:', error);
    }
  };

  const toggleSubject = (subject) => {
```

```
    setExpandedSubjects(prevState => ({
      ...prevState,
      [subject]: !prevState[subject]
    }));
  };

  const getTotalScore = (assignments) => {
    return assignments.reduce((total, assignment) => {
      return total + (assignment.mark || 0);
    }, 0);
  };

  const formatDate = (dateTime) => {
    return dateTime ? dateTime.split(' ')[0] : ''; // Возвращаем только дату, игнорируя время
  };

  const renderAssignmentTable = (assignments) => (
    <table className="assignment-table">
      <thead>
        <tr>
          <th className="date-column">{t('date')}</th>
          <th>{t('nameOfAssignment')}</th>
          <th>{t('mark')}</th>
        </tr>
      </thead>
      <tbody>
        {assignments.map(assignment => (
          <tr key={assignment.assignment_name}>
            <td>{formatDate(assignment.assignment_date_time)}</td>
            <td>{assignment.assignment_name}</td>
            <td>{assignment.mark !== undefined ? assignment.mark : <span className="undefined-
mark">ще не виставлено</span>}</td>
          </tr>
        ))}
      </tbody>
    </table>
  );

  return (
    <div>
      <Navigation jwtToken={localStorage.getItem('jwtToken')} />
      <div className="diary-container">
        {Object.keys(subjects).map(subjectName => (
          <div key={subjectName} className="subject-container">
            <h3
              className="subject-header"
              onClick={() => toggleSubject(subjectName)}
            >
              {subjectName} (Total: {getTotalScore(subjects[subjectName])})
            </h3>
            {expandedSubjects[subjectName] && (
              <div>
                <hr /> {/* Полоса между таблицами */}
                {subjects[subjectName] && subjects[subjectName].length > 0
                  ? renderAssignmentTable(subjects[subjectName])
                  : <p>No assignments available</p>}
              </div>
            )}
          </div>
        ))}
      </div>
    </div>
  );
};

export default StudentDiary;
```

```
#pragma once


#include <string>
#include <string_view>


#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
```

```
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class StudentDiaryGet final : public userver::server::handlers::HttpHandlerBase
{
public:
    StudentDiaryGet( const userver::components::ComponentConfig& config,
                     const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-student-diary-get";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/student_diary_get.hpp"

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <userver/formats/serialize/common_containers.hpp>
#include <userver/http/common_headers.hpp>

namespace
{
using AssignmentsSet =
    std::tuple< int, std::string, std::string, std::string, std::string, std::string >;
using AssignmentSolutionRow = std::tuple< std::string, std::optional< int >, std::string >;
constexpr auto kAssignmentName{ "assignment_name" };
constexpr auto kMark{ "mark" };
constexpr auto kS3Location{ "s3_location" };
constexpr auto kDeadline{ "deadline" };
constexpr auto kSubjectName{ "subject_name" };
constexpr auto kSubjectGroupName{ "subject_group_name" };
} // namespace

namespace rl::handlers
{
StudentDiaryGet::StudentDiaryGet( const userver::components::ComponentConfig& config,
                                  const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}
```

```
std::string StudentDiaryGet::HandleRequestThrow( const userver::server::http::HttpRequest&
request,
                                                 userver::server::request::RequestContext& )
const
{
    const auto& jwt{ request.GetHeader(
        userver::http::headers::kAuthorization ) }; // TODO: split bearer

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto user_id_result{ pg_dao.getUserIdViaJwt( jwt ) };
    if ( user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "jwtIsNotValidDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto user_id{ user_id_result.AsSingleRow< int >() };

    if ( !pg_dao.isUserStudent( user_id ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAStudent";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto student_assignments_result{ pg_dao.getStudentAssignments( user_id ) };
    if ( student_assignments_result.IsEmpty() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kNoContent );
        return {};
    }

    const auto student_assignments{ student_assignments_result.AsSetOf< AssignmentsSet >(
        userver::storages::postgres::kRowTag ) };

    userver::formats::json::ValueBuilder response;

    // Group assignments by subject
    std::unordered_map< std::string, std::vector< userver::formats::json::ValueBuilder > >
        groupedAssignments;

    for ( const auto item : student_assignments )
    {
        const auto [ assignment_id,
                     assignment_name,
                     s3_key_location,
                     deadline,
```

```
                    subject_name,
                    subject_group_name ] = item;

        userver::formats::json::ValueBuilder json_obj;
        json_obj[ kAssignmentName ] = assignment_name;

        const auto student_assignment_solution_result{
            pg_dao.getStudentAssignmentSolution( user_id, assignment_id )
        };

        if ( !student_assignment_solution_result.IsEmpty() )
        {
            const auto [ solution_s3_key, mark, assignment_date_time ] =
                student_assignment_solution_result.AsSingleRow< AssignmentSolutionRow >(
                    userver::storages::postgres::kRowTag );

            json_obj[ "assignment_date_time" ] = assignment_date_time;
            // Если есть оценка, добавляем её в JSON объект
            if ( mark.has_value() )
            {
                json_obj[ kMark ] = mark.value();
            }
            else
            {
                json_obj[ kMark ] = 0; // Если оценка не задана, по умолчанию 0
            }
        }

        // Добавляем задание в группу по предмету
        groupedAssignments[ subject_name ].push_back( std::move( json_obj ) );
    }

    // Создаем JSON объекты для каждого предмета
    for ( const auto& [ subject_name, assignments ] : groupedAssignments )
    {
        userver::formats::json::ValueBuilder subjectObj;
        for ( auto assignment : assignments )
        {
            subjectObj.PushBack( std::move( assignment ) );
        }
        response[ subject_name ] = std::move( subjectObj );
    }

    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл university_add.hpp university_add.cpp AddUniversityForm.jsx

Реалізація функціональної вимоги Створення університету

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'

const AddUniversityForm = ({ onSubmit, onChange, formData, setShowForm }) => {
```

```
  const { t } = useTranslation();

  return (
    <div className='card'>
      <div className="card-body">
        <form onSubmit={onSubmit}>
        <h2 className='card-title'>{t('createUniversity')}
          <button onClick={() => setShowForm(false)}  type="button" class="btn btn-outline-dark
btn-sm">X</button>
        </h2>
          <div>
            <label>{t('nameOfUniversity')}</label>
            <input type="text" name="university_name" value={formData.university_name || ''}
onChange={onChange}
            placeholder='kpi' />
          </div>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
        </form>
      </div>
    </div>
  );
};

export default AddUniversityForm;
```

```
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class UniversityAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    UniversityAdd( const userver::components::ComponentConfig& config,
                   const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-university-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/admin/university_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
```

```
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kUniversityName{ "university_name" };
}

namespace rl::handlers
{
UniversityAdd::UniversityAdd( const userver::components::ComponentConfig& config,
                              const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string UniversityAdd::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                               userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend() || levelIt->second.to_str() != "full" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }
```

```
    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters( request_body,
                                                                           { kUniversityName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& university_name{ request_body[ kUniversityName ].As< std::string >() };

    // TODO: addAssignment jwt

    const pg::PgDao pgDao{ pg_cluster_ };
    auto result{ pgDao.createUniversity( university_name ) };
    if ( !result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "universityAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}

} // namespace rl::handlers
```

Файл faculty_add.hpp faculty_add.cpp FacultyForm.jsx

Реалізація функціональної вимоги Створення факультету

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'

const FacultyForm = ({ onSubmit, onChange, formData, setShowForm }) => {
  const { t } = useTranslation();

  return (
    <div className='card'>
      <div className="card-body">
        <form onSubmit={onSubmit}>
        <h2 className='card-title'>{t('createFaculty')}
          <button onClick={() => setShowForm(false)}  type="button" class="btn btn-outline-dark
btn-sm">X</button>
        </h2>
          <label>{t('facultyName')}</label>
          <input type="text" name="faculty_name" value={formData.faculty_name || ''}
onChange={onChange}
            placeholder="fict"/>

          <label>{t('universityName')}</label>
          <input type="text" name="university_name" value={formData.university_name || ''}
onChange={onChange}
            placeholder="kpi"/>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
```

```
        </form>
      </div>
    </div>
  );
};

export default FacultyForm;
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class FacultyAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    FacultyAdd( const userver::components::ComponentConfig& config,
                const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-faculty-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/admin/faculty_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kFacultyName{ "faculty_name" };
constexpr std::string_view kUniversityName{ "university_name" };
} // namespace

namespace rl::handlers
{
```

```
FacultyAdd::FacultyAdd( const userver::components::ComponentConfig& config,
                        const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string FacultyAdd::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                            userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kFacultyName, kUniversityName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
```

```
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& faculty_name{ request_body[ kFacultyName ].As< std::string >() };
    const auto& university_name{ request_body[ kUniversityName ].As< std::string >() };

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto get_university_id_result{ pg_dao.getUniversityId( university_name ) };
    if ( get_university_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "universityDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto university_id{ get_university_id_result.AsSingleRow< int >() };

    const auto insert_faculty_query_result{ pg_dao.addFaculty( faculty_name, university_id ) };
    if ( !insert_faculty_query_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "facultyAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл department_add.hpp department_add.cpp DepartmentForm.jsx

Реалізація функціональної вимоги Створення кафедри

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'

const DepartmentForm = ({ onSubmit, onChange, formData, setShowForm }) => {
  const { t } = useTranslation();

  return (
    <div className='card'>
      <div className="card-body">
        <form onSubmit={onSubmit}>
        <h2 className='card-title'>{t('createDepartment')}
          <button onClick={() => setShowForm(false)}  type="button" class="btn btn-outline-dark
btn-sm">X</button>
        </h2>
          <div>
            <label>{t('facultyName')}</label>
            <input type="text" name="faculty_name" value={formData.faculty_name || ''}
onChange={onChange}
              placeholder='fict'/>
          </div>
          <div>
```

```
            <label>{t('departmentName')}</label>
            <input type="text" name="department_name" value={formData.department_name || ''}
onChange={onChange}
            placeholder='ipi'/>
          </div>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
        </form>
      </div>
    </div>
  );
};

export default DepartmentForm;
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class DepartmentAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    DepartmentAdd( const userver::components::ComponentConfig& config,
                   const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-department-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/admin/department_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kFacultyName{ "faculty_name" };
constexpr std::string_view kDepartmentName{ "department_name" };
} // namespace

namespace rl::handlers
{
DepartmentAdd::DepartmentAdd( const userver::components::ComponentConfig& config,
                              const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string DepartmentAdd::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                               userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
```

```
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || ( levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty"
              && levelIt->second.to_str() != "department" ) )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kFacultyName, kDepartmentName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& faculty_name{ request_body[ kFacultyName ].As< std::string >() };
    const auto& department_name{ request_body[ kDepartmentName ].As< std::string >() };

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto get_faculty_id_result{ pg_dao.getFacultyId( faculty_name ) };
    if ( get_faculty_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "facultyDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto faculty_id{ get_faculty_id_result.AsSingleRow< int >() };

    const auto insert_faculty_query_result{ pg_dao.addDepartment( department_name, faculty_id )
};
    if ( !insert_faculty_query_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "departmentAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл teacher_rank_add.hpp teacher_rank_add.cpp TeacherRankForm.jsx

Реалізація функціональної вимоги Створення рівня викладача

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'
```

```
const AddTeacherRankForm = ({ onSubmit, onChange, formData, setShowForm }) => {
  const { t } = useTranslation();

  return (
    <div className='card'>
      <div className="card-body">
        <form onSubmit={onSubmit}>
          <h2 className='card-title'>{t('createTeacherRank')}
              <button onClick={() => setShowForm(false)}  type="button" class="btn btn-outline-
dark btn-sm">X</button>
            </h2>
          <div>
            <label>{t('rank')}</label>
            <input type="text" name="teacher_rank" value={formData.teacher_rank || ''}
onChange={onChange}
            placeholder='starshiy vukladach'/>
          </div>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
        </form>
      </div>
    </div>
  );
};

export default AddTeacherRankForm;
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class TeacherRankAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    TeacherRankAdd( const userver::components::ComponentConfig& config,
                    const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-teacher-rank-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/admin/teacher_rank_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kTeacherRank{ "teacher_rank" };
}

namespace rl::handlers
{
TeacherRankAdd::TeacherRankAdd( const userver::components::ComponentConfig& config,
                                const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
```

```
}

std::string TeacherRankAdd::HandleRequestThrow( const userver::server::http::HttpRequest&
request,
                                                userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters( request_body,
                                                                           { kTeacherRank } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& teacher_rank{ request_body[ kTeacherRank ].As< std::string >() };

    const pg::PgDao pg_dao{ pg_cluster_ };
    const auto result{ pg_dao.createTeacherRank( teacher_rank ) };

    if ( !result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "teacherRankAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл teacher_add.hpp teacher_add.cpp AddTeacherForm.jsx

Реалізація функціональної вимоги Надання прав викладача користувачеві

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'

const AddTeacherForm = ({ onSubmit, onChange, formData, setShowForm }) => {
  const { t } = useTranslation();
```

```
  return (
    <div className='card'>
      <div className="card-body">
        <form onSubmit={onSubmit}>
          <h2 className='card-title'>{t('addTeacher')}
            <button onClick={() => setShowForm(false)} type="button" class="btn btn-outline-dark
btn-sm">X</button>
          </h2>
          <div>
            <label>{t('usernameOrEmail')}</label>
            <input type="text" name="email_or_username" value={formData.email_or_username || ''}
onChange={onChange}
            placeholder='ostap'/>
          </div>
          <div>
            <label>{t('rank')}</label>
            <input type="text" name="rank_name" value={formData.rank_name || ''}
onChange={onChange}
              placeholder='starshiy vukladach'/>
          </div>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
        </form>
      </div>
    </div>
  );
};

export default AddTeacherForm;
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class TeacherAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    TeacherAdd( const userver::components::ComponentConfig& config,
                const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-teacher-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/admin/teacher_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kEmailOrUsername{ "email_or_username" };
constexpr std::string_view kRankName{ "rank_name" };
} // namespace

namespace rl::handlers
{
TeacherAdd::TeacherAdd( const userver::components::ComponentConfig& config,
                        const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
```

```
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string TeacherAdd::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                            userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kEmailOrUsername, kRankName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& email_or_username{ request_body[ kEmailOrUsername ].As< std::string >() };
    const auto& rank_name{ request_body[ kRankName ].As< std::string >() };

    const pg::PgDao pg_dao{ pg_cluster_ };
    const auto get_user_id_result{ pg_dao.getUserIdViaEmailOrUsername( email_or_username ) };

    if ( get_user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto user_id{ get_user_id_result.AsSingleRow< int >() };

    const auto get_rank_id_result{ pg_dao.getRankId( rank_name ) };

    if ( get_rank_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "rankDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }
```

```
    const auto rank_id{ get_rank_id_result.AsSingleRow< int >() };

    const auto insert_teacher_query_result{ pg_dao.addTeacher( user_id, rank_id ) };
    if ( !insert_teacher_query_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "teacherAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл teacher_add_to_faculty.hpp teacher_add_to_faculty.cpp AddTeacherToFacultyForm.jsx

Реалізація функціональної вимоги Додавання викладача до факультету

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'
import { t } from 'i18next';

const AddTeacherToFacultyForm = ({ onSubmit, onChange, formData, setShowForm }) => {
  return (
    <div className='card'>
      <div className="card-body">
        <form onSubmit={onSubmit}>
        <h2 className='card-title'>{t('addTeacherToFaculty')}
          <button onClick={() => setShowForm(false)}  type="button" class="btn btn-outline-dark
btn-sm">X</button>
        </h2>
          <div>
            <label>{t('usernameOrEmail')}</label>
            <input type="text" name="email_or_username" value={formData.email_or_username || ''}
onChange={onChange} />
          </div>
          <div>
            <label>{t('facultyName')}</label>
            <input type="text" name="faculty_name" value={formData.faculty_name || ''}
onChange={onChange} />
          </div>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
        </form>
      </div>
    </div>
  );
};

export default AddTeacherToFacultyForm;
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class TeacherAddToFaculty final : public userver::server::handlers::HttpHandlerBase
{
public:
    TeacherAddToFaculty( const userver::components::ComponentConfig& config,
                         const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-teacher-add-to-faculty";
    using HttpHandlerBase::HttpHandlerBase;
```

```
    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/admin/teacher_add_to_faculty.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kEmailOrUsername{ "email_or_username" };
constexpr std::string_view kFacultyName{ "faculty_name" };
} // namespace

namespace rl::handlers
{
TeacherAddToFaculty::TeacherAddToFaculty( const userver::components::ComponentConfig& config,
                                          const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string
TeacherAddToFaculty::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                         userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kEmailOrUsername, kFacultyName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
```

```
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& email_or_username{ request_body[ kEmailOrUsername ].As< std::string >() };
    const auto& faculty_name{ request_body[ kFacultyName ].As< std::string >() };

    const pg::PgDao dao{ pg_cluster_ };
    const auto user_id_result{ dao.getUserIdViaEmailOrUsername( email_or_username ) };

    if ( user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto user_id{ user_id_result.AsSingleRow< int >() };
    if ( !dao.isUserTeacher( user_id ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotATeacher";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto get_faculty_query_result{ dao.getFacultyId( faculty_name ) };
    if ( get_faculty_query_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "facultyDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto faculty_id{ get_faculty_query_result.AsSingleRow< int >() };

    const auto connect_teacher_to_faculty_query_result{ dao.connectTeacherToFaculty( user_id,
                                                                                     faculty_id )
};
    if ( !connect_teacher_to_faculty_query_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "teacherAlreadyExistsOnThatFaculty";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл department_group_add.hpp delartment_group_add.cpp DepartmentGroupForm.jsx

Реалізація функціональної вимоги Створення групи на кафедрі

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'
import { t } from 'i18next';

const DepartmentGroupForm = ({ onSubmit, onChange, formData, setShowForm }) => {
  return (
    <div className='card'>
      <div className="card-body">
        <form onSubmit={onSubmit}>
        <h2 className='card-title'>{t('createDepartmentGroup')}
          <button onClick={() => setShowForm(false)}  type="button" class="btn btn-outline-dark
btn-sm">X</button>
        </h2>
          <div>
```

```
            <label>{t('departmentName')}</label>
            <input type="text" name="department_name" value={formData.department_name || ''}
onChange={onChange} />
          </div>
          <div>
            <label>{t('groupName')}</label>
            <input type="text" name="department_group_name" value={formData.department_group_name
|| ''} onChange={onChange} />
          </div>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
        </form>
      </div>
    </div>
  );
};

export default DepartmentGroupForm;
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class DepartmentGroupAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    DepartmentGroupAdd( const userver::components::ComponentConfig& config,
                        const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-department-group-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/admin/department_group_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kDepartmentName{ "department_name" };
constexpr std::string_view kDepartmentGroupName{ "department_group_name" };
} // namespace

namespace rl::handlers
{
DepartmentGroupAdd::DepartmentGroupAdd( const userver::components::ComponentConfig& config,
                                        const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string
DepartmentGroupAdd::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                        userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };
```

```
    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || ( levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty"
              && levelIt->second.to_str() != "department" ) )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kDepartmentGroupName, kDepartmentName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& department_group_name{ request_body[ kDepartmentGroupName ].As< std::string >()
};
    const auto& department_name{ request_body[ kDepartmentName ].As< std::string >() };

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto get_department_id_result{ pg_dao.getDepartmentId( department_name ) };
    if ( get_department_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "departmentDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto department_id{ get_department_id_result.AsSingleRow< int >() };

    const auto insert_department_group_query_result{
        pg_dao.addDepartmentGroup( department_group_name, department_id )
    };
    if ( !insert_department_group_query_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "departmentGroupAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл student_add.hpp student_add.cpp StudentForm.jsx

Реалізація функціональної вимоги Додавання студента до групи на кафедрі

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'

const StudentForm = ({ onSubmit, onChange, formData, setShowForm }) => {
  const {t} = useTranslation();

  return (
    <div className='card'>
      <div className="card-body">
        <form onSubmit={onSubmit}>
          <h2 className='card-title'>{t('addStudentToDepartmentGroup')}
            <button onClick={() => setShowForm(false)}  type="button" class="btn btn-outline-dark
btn-sm">X</button>
          </h2>
          <div>
            <label>{t('usernameOrEmail')}</label>
            <input type="text" name="email_or_username" value={formData.email_or_username || ''}
onChange={onChange} />
          </div>
          <div>
            <label>{t('groupName')}</label>
            <input type="text" name="department_group_name" value={formData.department_group_name
|| ''} onChange={onChange} />
          </div>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
        </form>
      </div>
    </div>
  );
};

export default StudentForm;
```

```
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class StudentAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    StudentAdd( const userver::components::ComponentConfig& config,
                const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-student-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers

#include "api/handlers/admin/student_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
```

```
{
constexpr std::string_view kEmailOrUsername{ "email_or_username" };
constexpr std::string_view kDepartmentGroupName{ "department_group_name" };
} // namespace

namespace rl::handlers
{
StudentAdd::StudentAdd( const userver::components::ComponentConfig& config,
                        const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string StudentAdd::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                            userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || ( levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty"
              && levelIt->second.to_str() != "department" ) )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kEmailOrUsername, kDepartmentGroupName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& email_or_username{ request_body[ kEmailOrUsername ].As< std::string >() };
    const auto& department_group_name{ request_body[ kDepartmentGroupName ].As< std::string >()
};

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto get_user_id_result{ pg_dao.getUserIdViaEmailOrUsername( email_or_username ) };

    if ( get_user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }
```

```
    const auto user_id{ get_user_id_result.AsSingleRow< int >() };

    const auto get_group_id_result{ pg_dao.getDepartmentGroupId( department_group_name ) };
    if ( get_group_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "departmentGroupDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto department_group_id{ get_group_id_result.AsSingleRow< int >() };

    const auto insert_student_result{ pg_dao.addStudent( user_id, department_group_id ) };
    if ( !insert_student_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "studentAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл subject_add.hpp subject_add.cpp AddSubjectToDepartmentGroup.jsx

Реалізація функціональної вимоги Створення предмету на кафедрі

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'

const AddSubjectToDepartmentForm = ({ onSubmit, onChange, formData, setShowForm }) => {
  const { t } = useTranslation();

  return (
    <div className='card'>
      <div className="card-body">
        <form onSubmit={onSubmit}>
          <h2 className='card-title'>{t('createSubjectToDepartment')}
            <button onClick={() => setShowForm(false)}  type="button" class="btn btn-outline-dark
btn-sm">X</button>
          </h2>
          <div>
            <label>{t('lectorUsernameOrEmail')}</label>
            <input type="text" name="lector_email_or_username"
value={formData.lector_email_or_username || ''} onChange={onChange} />
          </div>
          <div>
            <label>{t('departmentName')}</label>
            <input type="text" name="department_name" value={formData.department_name || ''}
onChange={onChange} />
          </div>
          <div>
            <label>{t('subjectName')}</label>
            <input type="text" name="subject_name" value={formData.subject_name || ''}
onChange={onChange} />
          </div>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
        </form>
      </div>
    </div>
  );
};

export default AddSubjectToDepartmentForm;
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
```

```
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class SubjectAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    SubjectAdd( const userver::components::ComponentConfig& config,
                const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-subject-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/admin/subject_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kLectorEmailOrUsername{ "lector_email_or_username" };
constexpr std::string_view kDepartmentName{ "department_name" };
constexpr std::string_view kSubjectName{ "subject_name" };
} // namespace

namespace rl::handlers
{
SubjectAdd::SubjectAdd( const userver::components::ComponentConfig& config,
                        const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string SubjectAdd::HandleRequestThrow( const userver::server::http::HttpRequest& request,
                                            userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || ( levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty"
              && levelIt->second.to_str() != "department" ) )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );
```

```
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kLectorEmailOrUsername, kDepartmentName, kSubjectName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& lector_email_or_username{
        request_body[ kLectorEmailOrUsername ].As< std::string >()
    };
    const auto& subject_name{ request_body[ kSubjectName ].As< std::string >() };
    const auto& department_name{ request_body[ kDepartmentName ].As< std::string >() };

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto get_user_id_result{ pg_dao.getUserIdViaEmailOrUsername( lector_email_or_username )
};

    if ( get_user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto user_id{ get_user_id_result.AsSingleRow< int >() };

    if ( !pg_dao.isUserTeacher( user_id ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotATeacher;";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto department_result{ pg_dao.getDepartmentId( department_name ) };
    if ( department_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "departmentDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto department_id{ department_result.AsSingleRow< int >() };

    const auto add_subject_result{ pg_dao.addSubject( subject_name, department_id, user_id ) };
    if ( !add_subject_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл subject_group_add.hpp subject_group_add.cpp AddSubjectGroupForm.jsx

Реалізація функціональної вимоги Створення групи для предмету

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'

const AddSubjectGroupForm = ({ onSubmit, onChange, formData, setShowForm }) => {
  const {t} = useTranslation();

  return (
    <div className='card'>
      <div className="card-body">
        <form onSubmit={onSubmit}>
          <h2 className='card-title'>{t('createSubjectGroup')}
            <button onClick={() => setShowForm(false)}  type="button" class="btn btn-outline-dark
btn-sm">X</button>
          </h2>
          <div>
            <label>{t('usernameOrEmailOfPractic')}</label>
            <input type="text" name="practic_email_or_username"
value={formData.practic_email_or_username || ''} onChange={onChange} />
          </div>
          <div>
            <label>{t('subjectName')}</label>
            <input type="text" name="subject_name" value={formData.subject_name || ''}
onChange={onChange} />
          </div>
          <div>
            <label>{t('subjectGroupName')}</label>
            <input type="text" name="subject_group_name" value={formData.subject_group_name ||
''} onChange={onChange} />
          </div>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
        </form>
      </div>
    </div>
  );
};

export default AddSubjectGroupForm;
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class SubjectGroupAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    SubjectGroupAdd( const userver::components::ComponentConfig& config,
                     const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName = "handler-subject-group-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/admin/subject_group_add.hpp"

#include <userver/http/common_headers.hpp>
```

```
#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kPracticUsernameOrEmail{ "practic_email_or_username" };
constexpr std::string_view kSubjectGroupName{ "subject_group_name" };
constexpr std::string_view kSubjectName{ "subject_name" };
} // namespace

namespace rl::handlers
{
SubjectGroupAdd::SubjectGroupAdd( const userver::components::ComponentConfig& config,
                                  const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string SubjectGroupAdd::HandleRequestThrow( const userver::server::http::HttpRequest&
request,
                                                 userver::server::request::RequestContext& )
const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || ( levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty"
              && levelIt->second.to_str() != "department" ) )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kPracticUsernameOrEmail, kSubjectName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& practic_email_or_username{
        request_body[ kPracticUsernameOrEmail ].As< std::string >()
    };
    const auto& subject_name{ request_body[ kSubjectName ].As< std::string >() };

    const pg::PgDao pg_dao{ pg_cluster_ };
```

```
    const auto get_user_id_result{ pg_dao.getUserIdViaEmailOrUsername(
        practic_email_or_username ) };

    if ( get_user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto user_id{ get_user_id_result.AsSingleRow< int >() };

    if ( !pg_dao.isUserTeacher( user_id ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotATeacher;";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto subject_result{ pg_dao.getSubjectId( subject_name ) };
    if ( subject_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto subject_id{ subject_result.AsSingleRow< int >() };

    const auto subject_group_name = [ request_body, pg_dao, subject_name, subject_id ]()
    {
        if ( request_body.HasMember( kSubjectGroupName ) )
        {
            return request_body[ kSubjectGroupName ].As< std::string >();
        }

        const auto subject_groups_amount{ pg_dao.getSubjectGroupsCount( subject_id ) };

        return subject_name + '-' + std::to_string( subject_groups_amount + 1 );
    }();

    const auto add_subject_group_result{
        pg_dao.addSubjectGroup( subject_group_name, subject_id, user_id )
    };
    if ( !add_subject_group_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectGroupAlreadyExists";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл student_to_existing_subject_group_add.hpp student_to_existing_subject_group_add.cpp AddStudentToGroupForm.jsx

Реалізація функціональної вимоги Додавання студента до групи предмету

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'

const AddStudentToGroupForm = ({ onSubmit, onChange, formData, setShowForm }) => {
  const {t} = useTranslation();

  return (
    <div className='card'>
      <div className="card-body">
```

```
        <form onSubmit={onSubmit}>
          <h2 className='card-title'>{t('addStudentToDepartmentGroup')}
            <button onClick={() => setShowForm(false)}  type="button" class="btn btn-outline-dark
btn-sm">X</button>
          </h2>
          <div>
            <label>{t('usernameOrEmail')}</label>
            <input type="text" name="email_or_username" value={formData.email_or_username || ''}
onChange={onChange} />
          </div>
          <div>
            <label>{t('subjectGroupName')}</label>
            <input type="text" name="subject_group_name" value={formData.subject_group_name ||
''} onChange={onChange} />
          </div>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
        </form>
      </div>
    </div>
  );
};

export default AddStudentToGroupForm;
#pragma once

#include <string>
#include <string_view>

#include <userver/components/component_config.hpp>
#include <userver/components/component_context.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/server/http/http_request.hpp>
#include <userver/server/request/request_context.hpp>
#include <userver/storages/postgres/postgres.hpp>

namespace rl::handlers
{
class StudentToExistingSubjectGroupAdd final : public userver::server::handlers::HttpHandlerBase
{
public:
    StudentToExistingSubjectGroupAdd( const userver::components::ComponentConfig& config,
                                      const userver::components::ComponentContext& context );

public:
    static constexpr std::string_view kName =
        "handler-teacher-student-to-existing-subject-group-add";
    using HttpHandlerBase::HttpHandlerBase;

    std::string HandleRequestThrow( const userver::server::http::HttpRequest&,
                                    userver::server::request::RequestContext& ctx ) const
override;

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::handlers
#include "api/handlers/admin/student_to_existing_subject_group_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kEmailOrUsername{ "email_or_username" };
constexpr std::string_view kSubjectGroupName{ "subject_group_name" };
} // namespace

namespace rl::handlers
{
StudentToExistingSubjectGroupAdd::StudentToExistingSubjectGroupAdd(
    const userver::components::ComponentConfig& config,
    const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
```

```
}

std::string StudentToExistingSubjectGroupAdd::HandleRequestThrow(
    const userver::server::http::HttpRequest& request,
    userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || ( levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty"
              && levelIt->second.to_str() != "department" ) )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kEmailOrUsername, kSubjectGroupName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& student_email_or_username{ request_body[ kEmailOrUsername ].As< std::string >()
};
    const auto& subject_group_name{ request_body[ kSubjectGroupName ].As< std::string >() };

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto get_user_id_result{ pg_dao.getUserIdViaEmailOrUsername(
        student_email_or_username ) };

    if ( get_user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto user_id{ get_user_id_result.AsSingleRow< int >() };

    if ( !pg_dao.isUserStudent( user_id ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAStudent";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }
```

```
    const auto subject_group_id_result{ pg_dao.getSubjectGroupId( subject_group_name ) };

    if ( subject_group_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectGroupDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto subject_group_id{ subject_group_id_result.AsSingleRow< int >() };

    const auto student_add_to_subject_group_result{
        pg_dao.addStudentToSubjectGroup( subject_group_id, user_id )
    };
    if ( !student_add_to_subject_group_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "studentIsAlreadyInSubjectGroup";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
```

Файл subject_group_as_department_group_add.hpp subject_group_as_department_group_add.cpp AddSubjectGroupToDepartmentGroupForm.jsx

Реалізація функціональної вимоги Створення групи для предмету як групи на кафедрі

```
import React from 'react';
import { useTranslation } from 'react-i18next';

import './Form.css'

const AddSubjectGroupToDepartmentGroupForm = ({ onSubmit, onChange, formData, setShowForm }) => {
  const {t} = useTranslation();

  return (
    <div className='card'>
      <div className="card-body">
        <form onSubmit={onSubmit}>
          <h2 className='card-title'>{t('createSubjectGroupAsDepartmentGroup')}
            <button onClick={() => setShowForm(false)}  type="button" class="btn btn-outline-dark
btn-sm">X</button>
          </h2>
          <div>
            <label>{t('usernameOrEmailOfPractic')}</label>
            <input type="text" name="practic_email_or_username"
value={formData.practic_email_or_username || ''} onChange={onChange} />
          </div>
          <div>
            <label>{t('subjectName')}</label>
            <input type="text" name="subject_name" value={formData.subject_name || ''}
onChange={onChange} />
          </div>
          <div>
            <label>{t('groupName')}</label>
            <input type="text" name="department_group_name" value={formData.department_group_name
|| ''} onChange={onChange} />
          </div>
          <button className='btn btn-outline-success' type="submit">{t('Submit')}</button>
        </form>
      </div>
    </div>
  );
};

export default AddSubjectGroupToDepartmentGroupForm;

#include "api/handlers/admin/student_to_existing_subject_group_add.hpp"
```

```
#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kEmailOrUsername{ "email_or_username" };
constexpr std::string_view kSubjectGroupName{ "subject_group_name" };
} // namespace

namespace rl::handlers
{
StudentToExistingSubjectGroupAdd::StudentToExistingSubjectGroupAdd(
    const userver::components::ComponentConfig& config,
    const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}

std::string StudentToExistingSubjectGroupAdd::HandleRequestThrow(
    const userver::server::http::HttpRequest& request,
    userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || ( levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty"
              && levelIt->second.to_str() != "department" ) )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kEmailOrUsername, kSubjectGroupName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& student_email_or_username{ request_body[ kEmailOrUsername ].As< std::string >()
};
    const auto& subject_group_name{ request_body[ kSubjectGroupName ].As< std::string >() };

    const pg::PgDao pg_dao{ pg_cluster_ };
```

```
    const auto get_user_id_result{ pg_dao.getUserIdViaEmailOrUsername(
        student_email_or_username ) };

    if ( get_user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto user_id{ get_user_id_result.AsSingleRow< int >() };

    if ( !pg_dao.isUserStudent( user_id ) )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAStudent";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto subject_group_id_result{ pg_dao.getSubjectGroupId( subject_group_name ) };

    if ( subject_group_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "subjectGroupDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto subject_group_id{ subject_group_id_result.AsSingleRow< int >() };

    const auto student_add_to_subject_group_result{
        pg_dao.addStudentToSubjectGroup( subject_group_id, user_id )
    };
    if ( !student_add_to_subject_group_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "studentIsAlreadyInSubjectGroup";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    userver::formats::json::ValueBuilder response;
    response[ "status" ] = "success";
    return userver::formats::json::ToString( response.ExtractValue() );
}
} // namespace rl::handlers
#include "api/handlers/admin/subject_group_as_department_group_add.hpp"

#include <userver/http/common_headers.hpp>

#include "api/handlers/validators/ParameterValidator.hpp"
#include "api/pg/PgDao.hpp"

#include <jwt-cpp/jwt.h>

namespace
{
constexpr std::string_view kPracticUsernameOrEmail{ "practic_email_or_username" };
constexpr std::string_view kSubjectName{ "subject_name" };
constexpr std::string_view kDepartmentGroupName{ "department_group_name" };
} // namespace

namespace rl::handlers
{
SubjectGroupAsDepartmentGroupAdd::SubjectGroupAsDepartmentGroupAdd(
    const userver::components::ComponentConfig& config,
    const userver::components::ComponentContext& context )
    : userver::server::handlers::HttpHandlerBase{ config, context }
    , pg_cluster_{
        context.FindComponent< userver::components::Postgres >( "auth-database" ).GetCluster()
    }
{
}
```

```
std::string SubjectGroupAsDepartmentGroupAdd::HandleRequestThrow(
    const userver::server::http::HttpRequest& request,
    userver::server::request::RequestContext& ) const
{
    const auto& jwt{ request.GetHeader( userver::http::headers::kAuthorization ) };

    auto decodedJwt = jwt::decode( jwt );

    auto verifier = jwt::verify().allow_algorithm( jwt::algorithm::hs256{ "secret" } );

    verifier.verify( decodedJwt );
    const auto payload{ decodedJwt.get_payload_json() };

    const auto roleIt{ payload.find( "role" ) };
    if ( roleIt == payload.cend() || roleIt->second.to_str() != "admin" )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userIsNotAdmin";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto levelIt{ payload.find( "level" ) };
    if ( levelIt == payload.cend()
         || ( levelIt->second.to_str() != "full" && levelIt->second.to_str() != "faculty"
              && levelIt->second.to_str() != "department" ) )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kUnauthorized );

        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "noAccess";

        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto request_body{ userver::formats::json::FromString( request.RequestBody() ) };

    const auto body_validation_error{
        validators::ParameterValidator::getErrorIfNotPassedBodyParameters(
            request_body,
            { kPracticUsernameOrEmail, kSubjectName, kDepartmentGroupName } )
    };
    if ( body_validation_error.has_value() )
    {
        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( body_validation_error.value() );
    }

    const auto& practic_email_or_username{
        request_body[ kPracticUsernameOrEmail ].As< std::string >()
    };
    const auto& subject_name{ request_body[ kSubjectName ].As< std::string >() };
    const auto& department_group_name{ request_body[ kDepartmentGroupName ].As< std::string >()
};

    const pg::PgDao pg_dao{ pg_cluster_ };

    const auto department_group_id_result{ pg_dao.getDepartmentGroupId( department_group_name )
};
    if ( department_group_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "departmentGroupDoesntExist";

        request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
        return userver::formats::json::ToString( response.ExtractValue() );
    }

    const auto department_group_id{ department_group_id_result.AsSingleRow< int >() };

    const auto get_user_id_result{ pg_dao.getUserIdViaEmailOrUsername(
        practic_email_or_username ) };

    if ( get_user_id_result.IsEmpty() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "userDoesntExist";
```

```
    request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const auto user_id{ get_user_id_result.AsSingleRow< int >() };

if ( !pg_dao.isUserTeacher( user_id ) )
{
    userver::formats::json::ValueBuilder response;
    response[ "error" ] = "userIsNotATeacher;";

    request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const auto subject_result{ pg_dao.getSubjectId( subject_name ) };
if ( subject_result.IsEmpty() )
{
    userver::formats::json::ValueBuilder response;
    response[ "error" ] = "subjectDoesntExist";

    request.SetResponseStatus( userver::server::http::HttpStatus::kBadRequest );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const auto subject_id{ subject_result.AsSingleRow< int >() };
const auto subject_group_name{ subject_name + '-' + department_group_name };

const auto add_subject_group_result{
    pg_dao.addSubjectGroup( subject_group_name, subject_id, user_id )
};
if ( !add_subject_group_result.RowsAffected() )
{
    userver::formats::json::ValueBuilder response;
    response[ "error" ] = "subjectGroupAlreadyExists";

    request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const auto subject_group_id_result{ pg_dao.getSubjectGroupId( subject_group_name ) };
if ( subject_group_id_result.IsEmpty() )
{
    userver::formats::json::ValueBuilder response;
    response[ "error" ] = "internalServerError";

    request.SetResponseStatus( userver::server::http::HttpStatus::kInternalServerError );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const auto subject_group_id{ subject_group_id_result.AsSingleRow< int >() };

const auto students_result{ pg_dao.getStudentsIdsInDepartmentGroup( department_group_id ) };
if ( students_result.IsEmpty() )
{
    userver::formats::json::ValueBuilder response;
    response[ "error" ] = "noStudentsInDepartmentGroup";

    request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
    return userver::formats::json::ToString( response.ExtractValue() );
}

const auto students_set{ students_result.AsSetOf< int >() };
for ( const auto student_id : students_set )
{
    const auto student_add_to_subject_group_result{
        pg_dao.addStudentToSubjectGroup( subject_group_id, student_id )
    };
    if ( !student_add_to_subject_group_result.RowsAffected() )
    {
        userver::formats::json::ValueBuilder response;
        response[ "error" ] = "studentFromDepartmentGroupIsAlreadyInSubjectGroup";

        request.SetResponseStatus( userver::server::http::HttpStatus::kConflict );
        return userver::formats::json::ToString( response.ExtractValue() );
    }
}

userver::formats::json::ValueBuilder response;
response[ "status" ] = "success";
return userver::formats::json::ToString( response.ExtractValue() );
```

```
}
} // namespace rl::handlers

Pgdao.hpp
Використовується у всіх функціональних вимогах
#pragma once

#include <userver/storages/postgres/cluster.hpp>

namespace rl::pg
{
class PgDao
{
public:
    explicit PgDao( userver::storages::postgres::ClusterPtr pg_cluster )
        : pg_cluster_{ std::move( pg_cluster ) }
    {
    }

    [[nodiscard]] auto createUniversity( std::string_view name ) const
    {
        const auto query{
            "INSERT INTO university.universities (name) "
            "VALUES ($1) "
            "ON CONFLICT DO NOTHING;"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                     query,
                                     name );
    }

    [[nodiscard]] auto createTeacherRank( std::string_view rank ) const
    {
        const auto query{
            "INSERT INTO faculty.teacher_ranks (name) "
            "VALUES ($1) "
            "ON CONFLICT DO NOTHING;"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                     query,
                                     rank );
    }

    [[nodiscard]] auto getUserIdViaEmailOrUsername( std::string_view email_or_username ) const
    {
        const auto get_user_id_query{
            "SELECT id FROM university.users "
            "WHERE LOWER(email) = LOWER($1) OR LOWER(username) = LOWER($2);"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                     get_user_id_query,
                                     email_or_username,
                                     email_or_username );
    }

    [[nodiscard]] bool isUserTeacher( int user_id ) const
    {
        const auto get_is_teacher_query{ "SELECT 1 FROM faculty.teachers WHERE id = $1;" };

        const auto result{ pg_cluster_->Execute(
            userver::storages::postgres::ClusterHostType::kMaster,
            get_is_teacher_query,
            user_id ) };

        return !result.IsEmpty() && result.AsSingleRow< int >() == 1;
    }

    [[nodiscard]] bool isUserAdmin( int user_id ) const
    {
        const auto get_is_admin_query{ "SELECT 1 FROM university.admins WHERE id = $1;" };

        const auto result{ pg_cluster_->Execute(
            userver::storages::postgres::ClusterHostType::kMaster,
            get_is_admin_query,
            user_id ) };

        return !result.IsEmpty() && result.AsSingleRow< int >() == 1;
    }
```

```
[[nodiscard]] bool isUserStudent( int user_id ) const
{
    const auto query{ "SELECT 1 FROM department.students WHERE id = $1;" };

    const auto result{ pg_cluster_->Execute(
        userver::storages::postgres::ClusterHostType::kMaster,
        query,
        user_id ) };

    return !result.IsEmpty() && result.AsSingleRow< int >() == 1;
}

[[nodiscard]] auto getFacultyId( std::string_view name ) const
{
    const auto get_faculty_query{
        "SELECT id FROM faculty.faculties "
        "WHERE LOWER(name) = LOWER($1);"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 get_faculty_query,
                                 name );
}

[[nodiscard]] auto getAdminLevel( int user_id ) const
{
    const auto get_faculty_query{
        "SELECT name FROM university.admins "
        "JOIN university.users ON university.users.id = university.admins.id "
        "JOIN university.admin_levels ON university.admins.level_id = "
        "university.admin_levels.id "
        "WHERE university.admins.id = $1;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 get_faculty_query,
                                 user_id );
}

[[nodiscard]] auto connectTeacherToFaculty( int teacher_id, int faculty_id ) const
{
    const auto connect_teacher_to_faculty_query{
        "INSERT INTO faculty.teachers_faculties (teacher_id, faculty_id) "
        "VALUES ($1, $2) "
        "ON CONFLICT DO NOTHING;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 connect_teacher_to_faculty_query,
                                 teacher_id,
                                 faculty_id );
}

[[nodiscard]] auto getRankId( std::string_view name ) const
{
    const auto get_rank_id_query{
        "SELECT id FROM faculty.teacher_ranks "
        "WHERE LOWER(name) = LOWER($1);"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 get_rank_id_query,
                                 name );
}

[[nodiscard]] auto addTeacher( int user_id, int rank_id ) const
{
    const auto insert_teacher_query{
        "INSERT INTO faculty.teachers (id, rank_id) "
        "VALUES ($1, $2) "
        "ON CONFLICT DO NOTHING;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 insert_teacher_query,
                                 user_id,
                                 rank_id );
}

[[nodiscard]] auto getDepartmentGroupId( std::string_view name ) const
```

```
{
    const auto get_group_id_query{
        "SELECT id FROM department.groups "
        "WHERE LOWER(name) = LOWER($1);"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 get_group_id_query,
                                 name );
}

[[nodiscard]] auto addStudent( int user_id, int department_group_id ) const
{
    const auto add_student_query{
        "INSERT INTO department.students (id, group_id) "
        "VALUES ($1, $2) "
        "ON CONFLICT DO NOTHING;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 add_student_query,
                                 user_id,
                                 department_group_id );
}

[[nodiscard]] auto getAdminLevelId( std::string_view level ) const
{
    const auto get_admin_level_id{
        "SELECT id FROM university.admin_levels "
        "WHERE LOWER(name) = LOWER($1);"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 get_admin_level_id,
                                 level );
}

[[nodiscard]] auto addAdmin( int user_id, int admin_level_id ) const
{
    const auto add_student_query{
        "INSERT INTO university.admins (id, level_id) "
        "VALUES ($1, $2) "
        "ON CONFLICT DO NOTHING;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 add_student_query,
                                 user_id,
                                 admin_level_id );
}

[[nodiscard]] auto getUniversityId( std::string_view name ) const
{
    const auto get_university_id_query{
        "SELECT id FROM university.universities "
        "WHERE LOWER(name) = LOWER($1)"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 get_university_id_query,
                                 name );
}

[[nodiscard]] auto addFaculty( std::string_view name, int university_id ) const
{
    const auto insert_faculty_query{
        "INSERT INTO faculty.faculties (name, university_id) "
        "VALUES ($1, $2) "
        "ON CONFLICT DO NOTHING;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 insert_faculty_query,
                                 name,
                                 university_id );
}

[[nodiscard]] auto getDepartmentId( std::string_view name ) const
{
    const auto get_department_id_query{
        "SELECT id FROM department.departments "
```

```
        "WHERE LOWER(name) = LOWER($1)"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 get_department_id_query,
                                 name );
}

[[nodiscard]] auto addDepartmentGroup( std::string_view name, int department_id ) const
{
    const auto insert_department_group_query{
        "INSERT INTO department.groups (name, department_id) "
        "VALUES ($1, $2) "
        "ON CONFLICT DO NOTHING;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 insert_department_group_query,
                                 name,
                                 department_id );
}

[[nodiscard]] auto addDepartment( std::string_view name, int faculty_id ) const
{
    const auto insert_department_query{
        "INSERT INTO department.departments (name, faculty_id) "
        "VALUES ($1, $2) "
        "ON CONFLICT DO NOTHING;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 insert_department_query,
                                 name,
                                 faculty_id );
}

[[nodiscard]] auto
addSubject( std::string_view subject_name, int department_id, int lector_id ) const
{
    const auto insert_subject_query{
        "INSERT INTO department.subjects (name, department_id, lector_id) "
        "VALUES ($1, $2, $3) "
        "ON CONFLICT DO NOTHING;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 insert_subject_query,
                                 subject_name,
                                 department_id,
                                 lector_id );
}

[[nodiscard]] auto getSubjectId( std::string_view subject_name ) const
{
    const auto insert_subject_query{
        "SELECT id FROM department.subjects "
        "WHERE LOWER(name) = LOWER($1);"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 insert_subject_query,
                                 subject_name );
}

[[nodiscard]] auto getSubjectBySubjectGroup( std::string_view subject_group_name ) const
{
    const auto query{
        "SELECT department.subjects.id, department.subjects.name FROM department.subjects "
        "JOIN subject.groups ON subject.groups.subject_id = department.subjects.id "
        "WHERE subject.groups.name = $1;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 query,
                                 subject_group_name );
}

[[nodiscard]] int getSubjectGroupsCount( int subject_id ) const
{
    const auto subject_groups_query{
        "SELECT COUNT(*) "
```

```
        "FROM subject.groups "
        "WHERE subject_id = $1;"
    };

    return pg_cluster_
        ->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                   subject_groups_query,
                   subject_id )
        .AsSingleRow< int >();
}

[[nodiscard]] auto getSubjectGroupId( std::string_view name ) const
{
    const auto subject_groups_query{
        "SELECT id FROM subject.groups "
        "WHERE LOWER(subject.groups.name) = LOWER($1);"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 subject_groups_query,
                                 name );
}

[[nodiscard]] auto
addSubjectGroup( std::string_view subject_group_name, int subject_id, int practic_id ) const
{
    const auto insert_subject_query{
        "INSERT INTO subject.groups (name, subject_id, practic_id) "
        "VALUES ($1, $2, $3) "
        "ON CONFLICT DO NOTHING;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 insert_subject_query,
                                 subject_group_name,
                                 subject_id,
                                 practic_id );
}

[[nodiscard]] auto getStudentsIdsInDepartmentGroup( int department_group_id ) const
{
    const auto insert_subject_query{
        "SELECT id FROM department.students "
        "WHERE group_id = $1"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 insert_subject_query,
                                 department_group_id );
}

[[nodiscard]] auto addStudentToSubjectGroup( int subject_group_id, int student_id ) const
{
    const auto query{
        "INSERT INTO subject.groups_students(subject_group_id, student_id) "
        "VALUES($1, $2) "
        "ON CONFLICT DO NOTHING"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 query,
                                 subject_group_id,
                                 student_id );
}

[[nodiscard]] auto addSubjectAssignment( int subject_group_id,
                                         std::string_view deadline,
                                         std::string_view name,
                                         std::string_view s3_key ) const
{
    const auto query{
        "INSERT INTO subject.assignments(subject_group_id, deadline, name, s3_key) "
        "VALUES ($1, $2::timestamp, $3, $4) "
        "ON CONFLICT DO NOTHING;"
    };

    return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                 query,
                                 subject_group_id,
                                 deadline,
                                 name,
```

```
                                    s3_key );
    }

    [[nodiscard]] auto getAssignmentId( int subject_group_id, std::string_view name ) const
    {
        const auto query{
            "SELECT id FROM subject.assignments "
            "WHERE subject_group_id = $1 AND name = $2"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                    query,
                                    subject_group_id,
                                    name );
    }

    [[nodiscard]] auto updateMarkForAssignment( int assignment_id, int student_id, int mark )
const
    {
        const auto query{
            "UPDATE subject.assignment_solutions "
            "SET mark = $1"
            "WHERE assignment_id = $2 AND student_id = $3;"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                    query,
                                    mark,
                                    assignment_id,
                                    student_id );
    }

    [[nodiscard]] auto
    addAssignmentSolution( int student_id, int assignment_id, std::string_view s3_key ) const
    {
        const auto query{
            "INSERT INTO subject.assignment_solutions (student_id, assignment_id, s3_key) "
            "VALUES ($1, $2, $3) "
            "ON CONFLICT DO NOTHING;"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                    query,
                                    student_id,
                                    assignment_id,
                                    s3_key );
    }

    [[nodiscard]] auto getUserIdViaJwt( std::string_view jwt ) const
    {
        const auto query{ "SELECT user_id FROM auth.tokens WHERE token = $1" };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                    query,
                                    jwt );
    }

    [[nodiscard]] auto getStudentAssignments( int user_id ) const
    {
        const auto query{
            "SELECT subject.assignments.id, subject.assignments.name, subject.assignments.s3_key,
"
            "TO_CHAR(subject.assignments.deadline AT TIME ZONE 'UTC', 'YYYY-MM-DD HH24:MI:SS
TZ'), "
            "department.subjects.name, subject.groups.name "
            "FROM subject.assignments "
            "JOIN subject.groups ON subject.assignments.subject_group_id = subject.groups.id "
            "JOIN subject.groups_students ON subject.groups.id = "
            "subject.groups_students.subject_group_id "
            "JOIN department.subjects ON department.subjects.id = subject.groups.subject_id "
            "WHERE subject.groups_students.student_id = $1;"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                    query,
                                    user_id );
    }

    [[nodiscard]] auto getStudentAssignmentSolution( int user_id, int assignment_id ) const
    {
        const auto query{
```

```
            "SELECT subject.assignment_solutions.s3_key, subject.assignment_solutions.mark, "
            "TO_CHAR(subject.assignments.created_at, 'YYYY-MM-DD HH24:MI:SS TZ') "
            "FROM subject.assignments "
            "JOIN subject.assignment_solutions ON subject.assignments.id = "
            "subject.assignment_solutions.assignment_id "
            "WHERE subject.assignment_solutions.student_id = $1 AND subject.assignments.id = $2;"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                     query,
                                     user_id,
                                     assignment_id );
    }

    [[nodiscard]] auto getTeacherAssignments( int user_id ) const
    {
        const auto query{
            "SELECT subject.assignments.id, subject.assignments.name, subject.assignments.s3_key,
"
            "TO_CHAR(subject.assignments.deadline AT TIME ZONE 'UTC', 'YYYY-MM-DD HH24:MI:SS
TZ'), "
            "department.subjects.name, subject.groups.name "
            "FROM subject.assignments "
            "JOIN subject.groups ON subject.groups.id = subject.assignments.subject_group_id "
            "JOIN department.subjects ON department.subjects.id = subject.groups.subject_id "
            "WHERE subject.groups.practic_id = $1;"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                     query,
                                     user_id );
    }

    [[nodiscard]] auto getAllSolutionsForAssignment( int assignment_id ) const
    {
        const auto query{
            "SELECT university.users.surname, university.users.last_name, "
            "university.users.middle_name, university.users.username, "
            "subject.assignment_solutions.s3_key, subject.assignment_solutions.mark "
            "FROM subject.assignments "
            "JOIN subject.assignment_solutions ON subject.assignments.id = "
            "subject.assignment_solutions.assignment_id "
            "JOIN university.users ON university.users.id = "
            "subject.assignment_solutions.student_id "
            "WHERE subject.assignments.id = $1"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                     query,
                                     assignment_id );
    }

    [[nodiscard]] auto setMarkForSolutionByS3Key( std::string_view s3_key, double mark ) const
    {
        const auto query{
            "UPDATE subject.assignment_solutions "
            "SET mark = $1 "
            "WHERE s3_key = $2 AND mark IS NULL;"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                     query,
                                     mark,
                                     s3_key );
    }

    [[nodiscard]] auto
    sendMessage( int subject_group_id, int user_id, std::string_view content ) const
    {
        const auto query{
            "INSERT INTO messenger.messages (subject_group_id, user_id, content) "
            "VALUES ($1, $2, $3) "
            "ON CONFLICT DO NOTHING;"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                     query,
                                     subject_group_id,
                                     user_id,
                                     content );
    }
```

```
    [[nodiscard]] auto getMessagesForSubjectGroup( int subject_group_id ) const
    {
        const auto query{
            "SELECT university.users.surname, university.users.last_name, "
            "TO_CHAR(messenger.messages.created_at, 'YYYY-MM-DD HH24:MI:SS TZ'), "
            "messenger.messages.content, university.users.id "
            "FROM messenger.messages "
            "JOIN university.users ON messenger.messages.user_id = university.users.id "
            "WHERE messenger.messages.subject_group_id = $1"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                     query,
                                     subject_group_id );
    }

    [[nodiscard]] auto getSubjectGroupsForStudent( int user_id ) const
    {
        const auto query{
            "SELECT subject.groups.name "
            "FROM subject.groups "
            "JOIN subject.groups_students ON subject.groups_students.subject_group_id = "
            "subject.groups.id "
            "WHERE subject.groups_students.student_id = $1"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                     query,

                                     user_id );
    }

    [[nodiscard]] auto getSubjectGroupsForTeacher( int user_id ) const
    {
        const auto query{
            "SELECT subject.groups.name "
            "FROM subject.groups "
            "WHERE subject.groups.practic_id = $1"
        };

        return pg_cluster_->Execute( userver::storages::postgres::ClusterHostType::kMaster,
                                     query,
                                     user_id );
    }

private:
    userver::storages::postgres::ClusterPtr pg_cluster_;
};
} // namespace rl::pg
```

Файл s3.hpp

Використовується у функціональних вимогах з єдиної системи завдань

```
#pragma once

#include <aws/auth/credentials.h>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsRequest.h>
#include <aws/s3/model/PutObjectRequest.h>

#include <magic_enum_all.hpp>

namespace rl::aws::s3
{
struct Homework
{
    std::string subject;  // Предмет
    std::string group;    // Группа
    std::string filename; // Название файла
    std::string content;  // Содержимое файла
};

enum class S3Bucket
{
    homework = 0
};
```

```
class S3Dao
{
public:
    S3Dao()
    {
        clientConfig_.endpointOverride = "http://s3.localhost.localstack.cloud:4566";
        clientConfig_.verifySSL        = false;
        clientConfig_.region           = "us-east-1";
    }

    bool createBucket( const Aws::String& bucketName ) const
    {
        Aws::S3::S3Client client( clientConfig_ );
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket( bucketName );

        // TODO(user): Change the bucket location constraint enum to your target Region.
        if ( clientConfig_.region != "us-east-1" )
        {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
            createBucketConfig.SetLocationConstraint(

Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                    clientConfig_.region ) );
            request.SetCreateBucketConfiguration( createBucketConfig );
        }

        Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket( request );
        if ( !outcome.IsSuccess() )
        {
            auto& err = outcome.GetError();
            LOG_ERROR() << "Error: CreateBucket: " << err.GetExceptionName() << ": "
                        << err.GetMessage();
        }
        else
        {
            LOG_INFO() << "Created bucket " << bucketName << " in the specified AWS Region.";
        }

        return outcome.IsSuccess();
    }

    bool listBuckets() const
    {
        Aws::S3::S3Client client( clientConfig_ );

        auto outcome = client.ListBuckets();

        bool result = true;
        if ( !outcome.IsSuccess() )
        {
            LOG_CRITICAL() << "Failed with error: " << outcome.GetError();
            result = false;
        }
        else
        {
            LOG_INFO() << "Found " << outcome.GetResult().GetBuckets().size() << " buckets\n";
            for ( auto&& b : outcome.GetResult().GetBuckets() )
            {
                LOG_INFO() << b.GetName();
            }
        }

        return result;
    }

public:
    [[nodiscard]] const Aws::Client::ClientConfiguration& getClientConfig() const
    {
        return clientConfig_;
    }

private:
    Aws::Client::ClientConfiguration clientConfig_;
};

class S3Homework
{
public:
    S3Homework() = default;
```

```
    [[nodiscard]] std::optional< std::string >
    addAssignment( std::string_view subject,
                   std::string_view group,
                   std::string_view file_name,
                   std::string_view homework_content ) const
    {
        // Формируем ключ (путь к файлу) в S3 в формате "group/subject/homework"
        std::string key = std::string( subject ) + "/" + std::string( group ) + "/" +
"assignments/"
            + std::string( file_name );

        // Инициализируем S3 клиент
        Aws::S3::S3Client client( s3dao_.getClientConfig() );

        // Создаем объект PutObjectRequest для загрузки файла в S3
        Aws::S3::Model::PutObjectRequest request;
        const auto bucketName{ Aws::String{ "homework" } };
        request.SetBucket( bucketName );
        request.SetKey( Aws::String{ key } );

        // Задаем содержимое файла в качестве данных для загрузки
        auto data = Aws::MakeShared< Aws::StringStream >( "homework" );
        *data << homework_content;
        request.SetBody( data );

        // Выполняем загрузку файла в S3
        auto outcome = client.PutObject( request );

        // Проверяем результат загрузки
        if ( outcome.IsSuccess() )
        {
            LOG_INFO() << "Uploaded homework file '" << file_name << "' successfully to S3.";
            return key;
        }

        auto& err = outcome.GetError();
        LOG_ERROR() << "Error uploading homework file '" << file_name
                    << "' to S3: " << err.GetExceptionName() << ": " << err.GetMessage();
        return std::nullopt;
    }

    [[nodiscard]] std::optional< std::string >
    addAssignmentSolution( std::string_view subject,
                           std::string_view group,
                           std::string_view file_name,
                           std::string_view homework_content,
                           int student_id ) const
    {
        // Формируем ключ (путь к файлу) в S3 в формате "group/subject/homework"
        std::string key = std::string( subject ) + "/" + std::string( group ) + "/" +
"solutions/"
            + std::to_string( student_id ) + std::string( file_name );

        // Инициализируем S3 клиент
        Aws::S3::S3Client client( s3dao_.getClientConfig() );

        // Создаем объект PutObjectRequest для загрузки файла в S3
        Aws::S3::Model::PutObjectRequest request;
        const auto bucketName{ Aws::String{ "homework" } };
        request.SetBucket( bucketName );
        request.SetKey( Aws::String{ key } );

        // Задаем содержимое файла в качестве данных для загрузки
        auto data = Aws::MakeShared< Aws::StringStream >( "homework" );
        *data << homework_content;
        request.SetBody( data );

        // Выполняем загрузку файла в S3
        auto outcome = client.PutObject( request );

        // Проверяем результат загрузки
        if ( outcome.IsSuccess() )
        {
            LOG_INFO() << "Uploaded homework file '" << file_name << "' successfully to S3.";
            return key;
        }

        auto& err = outcome.GetError();
        LOG_ERROR() << "Error uploading homework file '" << file_name
                    << "' to S3: " << err.GetExceptionName() << ": " << err.GetMessage();
```

```
    return std::nullopt;
}

// Тип, представляющий набор домашних заданий
using HomeworkSet = std::vector< Homework >;

struct FileContent
{
    std::string file_name;
    std::string content;
};

[[nodiscard]] std::optional< FileContent > getAssignment( std::string_view s3_key ) const
{
    // Инициализируем S3 клиент
    Aws::S3::S3Client client( s3dao_.getClientConfig() );

    // Создаем объект GetObjectRequest для получения файла из S3
    Aws::S3::Model::GetObjectRequest request;
    const auto bucketName{ Aws::String{ "homework" } };
    request.SetBucket( bucketName );
    request.SetKey( Aws::String{ s3_key } );

    // Выполняем запрос на получение файла из S3
    auto outcome = client.GetObject( request );

    // Проверяем результат запроса
    if ( outcome.IsSuccess() )
    {
        // Читаем содержимое файла
        Aws::StringStream ss;
        ss << outcome.GetResult().GetBody().rdbuf();
        std::string content = ss.str();

        // Возвращаем структуру FileContent
        return FileContent{ std::string( s3_key ), content };
    }

    auto& err = outcome.GetError();
    LOG_ERROR() << "Error getting assignment file '" << s3_key
                << "' from S3: " << err.GetExceptionName() << ": " << err.GetMessage();
    return std::nullopt;
}

[[nodiscard]] std::optional< FileContent >
getAssignmentSolution( std::string_view s3_key ) const
{
    // Инициализируем S3 клиент
    Aws::S3::S3Client client( s3dao_.getClientConfig() );

    // Создаем объект GetObjectRequest для получения файла из S3
    Aws::S3::Model::GetObjectRequest request;
    const auto bucketName{ Aws::String{ "homework" } };
    request.SetBucket( bucketName );
    request.SetKey( Aws::String{ s3_key } );

    // Выполняем запрос на получение файла из S3
    auto outcome = client.GetObject( request );

    // Проверяем результат запроса
    if ( outcome.IsSuccess() )
    {
        // Читаем содержимое файла
        Aws::StringStream ss;
        ss << outcome.GetResult().GetBody().rdbuf();
        std::string content = ss.str();

        // Возвращаем структуру FileContent
        return FileContent{ std::string( s3_key ), content };
    }

    auto& err = outcome.GetError();
    LOG_ERROR() << "Error getting assignment solution file '" << s3_key
                << "' from S3: " << err.GetExceptionName() << ": " << err.GetMessage();
    return std::nullopt;
}

std::string generatePresignedUrl( const std::string& objectKey ) const
{
    Aws::S3::S3Client s3_client( s3dao_.getClientConfig() );
```

```
        Aws::String bucket = "homework";
        Aws::String key    = objectKey.c_str();

        Aws::String presignedUrl = s3_client.GeneratePresignedUrl( bucket,
                                                                   key,
Aws::Http::HttpMethod::HTTP_GET,
                                                                   3600 ); // URL valid for 1
hour

        return std::string( presignedUrl.c_str() );
    }

private:
    S3Dao s3dao_;
    const S3Bucket bucket_{ S3Bucket::homework };
};
} // namespace rl::aws::s3
```

Файл ChatPage.jsx

Використовується для реалізації функціональних вимог зі сторінкою чатів

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { useJwt } from "react-jwt";
import moment from 'moment';
import { useTranslation } from 'react-i18next';
import Navigation from '../navigation/Navigation';
import { Helmet } from 'react-helmet';
import './ChatPage.css';

const BASE_URL = 'http://localhost:8080';

const ChatPage = () => {
  const { t, i18n } = useTranslation();
  const [chatList, setChatList] = useState([]);
  const [activeChat, setActiveChat] = useState(null);
  const [messages, setMessages] = useState([]);
  const [messageContent, setMessageContent] = useState('');
  const [loading, setLoading] = useState(true);
  const [chatsAvailable, setChatsAvailable] = useState(false);
  const { decodedToken } = useJwt(localStorage.getItem('jwtToken'));

  useEffect(() => {
    const fetchChatList = async () => {
      try {
        const jwtToken = localStorage.getItem('jwtToken');
        const response = await axios.get(`${BASE_URL}/v1/subject_groups`, {
          headers: { Authorization: jwtToken }
        });
        console.log('Chat list response:', JSON.stringify(response));
        if (response && !response.data)
        {
          setChatsAvailable(false);
        }
        setChatList(response.data);
        setChatsAvailable(true);
      } catch (error) {
        console.error('Error fetching chat list:', error);
        setChatsAvailable(false);
      }
    };

    fetchChatList();
  }, []);

  useEffect(() => {
    let interval;
    if (activeChat) {
      const fetchMessages = async () => {
        try {
          const jwtToken = localStorage.getItem('jwtToken');
          const response = await axios.get(`${BASE_URL}/v1/chat/messages/${activeChat}`, {
            headers: { Authorization: jwtToken }
          });
          console.log('Messages response:', JSON.stringify(response));

          const processedMessages = response.data.map(message => ({
            ...message,
```

```
            timestamp: moment.utc(message.timestamp, 'YYYY-MM-DD HH:mm:ss').local().format('YYYY-
MM-DD HH:mm:ss')
          }));

          const sortedMessages = processedMessages.sort((b, a) => moment(b.timestamp).valueOf() -
moment(a.timestamp).valueOf());

          setMessages(sortedMessages);
        } catch (error) {
          console.error('Error fetching messages:', error);
        }
      };

      fetchMessages();
      interval = setInterval(fetchMessages, 2000);
    } else {
      clearInterval(interval);
      setMessages([]);
    }

    return () => clearInterval(interval);
  }, [activeChat]);

  const sendMessage = async () => {
    try {
      const jwtToken = localStorage.getItem('jwtToken');
      await axios.post(`${BASE_URL}/v1/chat/send_message`, {
        subject_group_name: activeChat,
        content: messageContent
      }, {
        headers: { Authorization: jwtToken }
      });
      setMessageContent('');
    } catch (error) {
      console.error('Error sending message:', error);
    }
  };

  return (
    <div>
      <Helmet>
        <title>{t("nameOfProject")} | {t("chat")}</title>
      </Helmet>

      <Navigation jwtToken={localStorage.getItem('jwtToken')} />
      <div className="container-fluid chat-page">
        <div className="row">
          <div className="col-md-3 chat-panel">
            {chatsAvailable && chatList.length > 0 ? (
              chatList.map(chat => (
                <button
                  key={chat}
                  onClick={() => setActiveChat(chat)}
                  className={`btn btn-block ${chat === activeChat ? 'btn-primary active' : 'btn-
light'}`}
                >
                  {chat}
                </button>
              ))
            ) : (
              <div>{t('noChatsYet')}</div>
            )}
          </div>
          <div className="col-md-9 chat-area">
            <div className="messages">
              {!activeChat && chatsAvailable && <div className="hello-
world">{t('helloWorldChat')}</div>}
              {messages && messages.map((message, index) => (
                <div key={index} className={`message ${message.user_id.toString() ===
decodedToken.user_id ? 'own' : 'other'}`}>
                  <div className="message-content">
                    <strong>{message.surname} {message.last_name}</strong><br />
                    {message.content}
                    <div className="message-timestamp">{moment(message.timestamp).format('YYYY-
MM-DD HH:mm:ss')}</div>
                  </div>
                </div>
              ))}
            </div>
            {activeChat && (
              <div className="message-input">
```

```
                <input
                  type="text"
                  value={messageContent}
                  onChange={e => setMessageContent(e.target.value)}
                  onKeyDown={e => {
                    if (e.key === 'Enter') sendMessage();
                  }}
                  placeholder={t("chatTypeMsg")}
                  className="form-control"
                />
                <button onClick={sendMessage} className="btn btn-
primary">{t('chatSend')}</button>
              </div>
            )}
          </div>
        </div>
      </div>
    </div>
  );
};

export default ChatPage;
```

Файл StudentHomeworkPage.jsx

Реалізовує функціональні вимоги з приводу єдиної системи завдань

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import Navigation from "../navigation/Navigation"
import { Helmet } from 'react-helmet';
import './StudentHomeworkPage.css';
import { useTranslation } from 'react-i18next';

const StudentHomeworkPage = () => {
  const { t } = useTranslation();
  const [subjects, setSubjects] = useState([]);
  const [isLoading, setIsLoading] = useState(true);
  const [file, setFile] = useState(null);
  const [expandedSubjects, setExpandedSubjects] = useState(new Set());
  const [expandedAssignments, setExpandedAssignments] = useState(new Set());

  const [requestSuccess, setRequestSuccess] = useState(false);
  const [errorMessage, setErrorMessage] = useState('');
  const [messageVisible, setMessageVisible] = useState(false);

  useEffect(() => {
    const fetchSubjects = async () => {
      try {
        const jwtToken = localStorage.getItem('jwtToken');
        const response = await axios.get('http://localhost:8080/v1/subject/assignments/student',
{
          headers: {
            Authorization: `${jwtToken}`
          }
        });

        const groupedSubjects = groupAssignmentsBySubjects(response.data);
        setSubjects(groupedSubjects);
        setIsLoading(false);
      } catch (error) {
        console.error('Error fetching assignments:', error);
        setIsLoading(false);
      }
    };

    fetchSubjects();
  }, []);

  const groupAssignmentsBySubjects = (assignments) => {
    const subjectsMap = new Map();
    assignments.forEach((assignment) => {
      if (!subjectsMap.has(assignment.subject_name)) {
        subjectsMap.set(assignment.subject_name, []);
      }
      const subjectAssignments = subjectsMap.get(assignment.subject_name);
      subjectAssignments.push(assignment);
    });

    const groupedSubjects = [];
```

```
    subjectsMap.forEach((assignments, subjectName) => {
      groupedSubjects.push({
        subject_name: subjectName,
        assignments: assignments
      });
    });

    return groupedSubjects;
  };

  const handleSubmit = async (e, assignment) => {
    e.preventDefault();
    try {
      const jwtToken = localStorage.getItem('jwtToken');
      const formData = new FormData();
      formData.append('subject_group_name', assignment.subject_group_name);
      formData.append('assignment_name', assignment.assignment_name);
      formData.append('assignment_solution_file', file);

      const response = await
axios.post('http://localhost:8080/v1/subject/assignment/solution/add', formData, {
        headers: {
          'Authorization': `${jwtToken}`,
          'Content-Type': 'multipart/form-data'
        }
      });

      if (response.status === 200) {
        setRequestSuccess(true);
        await fetchSubjects(); // Fetch updated data after successful submission
      } else {
        setRequestSuccess(false);
        if (response.status === 404) {
          setErrorMessage('Not found');
        } else {
          const errorData = await response.data;
          setErrorMessage(errorData.error || 'Unknown error');
          console.error('Ошибка:', errorData);
        }
      }
    } catch (error) {
      setRequestSuccess(false);
      setErrorMessage('Unknown error');
    }
    setMessageVisible(true);
  };

  const fetchSubjects = async () => {
    try {
      const jwtToken = localStorage.getItem('jwtToken');

      const response = await axios.get('http://localhost:8080/v1/subject/assignments/student', {
        headers: {
          Authorization: `${jwtToken}`
        }
      });

      const groupedSubjects = groupAssignmentsBySubjects(response.data);
      setSubjects(groupedSubjects);
    } catch (error) {
      console.error('Error fetching assignments:', error);
    }
  };

  const handleFileChange = (e) => {
    setFile(e.target.files[0]);
  };

  const toggleSubject = (subjectName) => {
    const expandedSubjectsCopy = new Set(expandedSubjects);
    if (expandedSubjectsCopy.has(subjectName)) {
      expandedSubjectsCopy.delete(subjectName);
    } else {
      expandedSubjectsCopy.add(subjectName);
    }
    setExpandedSubjects(expandedSubjectsCopy);
  };

  const toggleAssignment = (assignmentId) => {
    const expandedAssignmentsCopy = new Set(expandedAssignments);
    if (expandedAssignmentsCopy.has(assignmentId)) {
```

```
      expandedAssignmentsCopy.delete(assignmentId);
    } else {
      expandedAssignmentsCopy.add(assignmentId);
    }
    setExpandedAssignments(expandedAssignmentsCopy);
  };

  const isDeadlineExpired = (deadline) => {
    const currentDateTime = new Date();
    const deadlineDateTime = new Date(deadline);
    return currentDateTime > deadlineDateTime;
  };

  // Fetch presigned URL for a specific S3 key
  const fetchPresignedUrl = async (s3Key) => {
    try {
      const response = await axios.get(`http://localhost:8080/v1/presigned-
url?s3_key=${encodeURIComponent(s3Key)}`, {
        headers: {
          Authorization: `${localStorage.getItem('jwtToken')}`
        }
      });
      return response.data.url;
    } catch (error) {
      console.error('Error fetching presigned URL:', error);
      return null;
    }
  };

  // Handle file download
  const handleFileDownload = async (s3Key) => {
    const presignedUrl = await fetchPresignedUrl(s3Key);
    if (presignedUrl) {
      window.open(presignedUrl, '_blank');
    } else {
      alert('Failed to generate download link. Please try again later.');
    }
  };

  return (
    <div>
      <Helmet>
        <title>{t("nameOfProject")} | {t("assignments")}</title>
      </Helmet>
      <Navigation jwtToken={localStorage.getItem('jwtToken')} />
      <div className="student-homework-page">
        <h1>Homework Page</h1>
        {isLoading ? (
          <p>Loading...</p>
        ) : (
          <div>
            {subjects.length === 0 ? (
              <p>No assignments available</p>
            ) : (
              subjects.map((subject, index) => (
                <div key={index} className="card mb-3">
                  <div className="card-header clickable" onClick={() =>
toggleSubject(subject.subject_name)}>
                    <h2 className="card-title">{subject.subject_name}</h2>
                  </div>
                  {expandedSubjects.has(subject.subject_name) && (
                    <div className="card-body">
                      {subject.assignments.map((assignment, assignmentIndex) => (
                        <div key={assignmentIndex} className="card mb-2">
                          <div className="card-header clickable" onClick={() =>
toggleAssignment(assignment.id)}>
                            <h4 className="card-title">
                              <div className="assignment-title">
                                {assignment.assignment_name}
{t('deadline')}: {assignment.deadline}
                              </div>
                            </h4>
                          </div>
                          {expandedAssignments.has(assignment.id) && (
                            <div className="card-body">
                              <button className="btn btn-outline-secondary" onClick={() =>
handleFileDownload(assignment.s3_location)}>
                                {t('downloadAssignment')}
                              </button>
                              <p>{assignment.solution ? <button className="btn btn-outline-
secondary" onClick={() =>
```

```
handleFileDownload(assignment.solution.s3_location)}>{t('downloadSolution')}</button> : 'Not
submitted'}</p>
                              {assignment.solution ? (
                                <p>{t('mark')}: {assignment.solution.mark ?
assignment.solution.mark : t('notYet')}</p>
                              ) : (
                                isDeadlineExpired(assignment.deadline) ? (
                                  <p>{t('deadlineExpired')}</p>
                                ) : (
                                  <form onSubmit={(e) => handleSubmit(e, assignment)}>
                                    <input type="file" className="form-control"
onChange={handleFileChange}></input>
                                    <button type="submit" className="btn btn-outline-
success">{t('submit')}</button>
                                  </form>
                                )
                              )}
                            </div>
                          )}
                        </div>
                      ))}
                    </div>
                  )}
                </div>
              ))
            )}
          </div>
        )}
        <div className="message">
          {messageVisible ? (requestSuccess ? 'Status: Success' : `Status: ${errorMessage}`) :
''}
        </div>
      </div>
    </div>
  );
};

export default StudentHomeworkPage;
```

Файл TeacherHomeworkPage.jsx

Реалізовує функціональні вимоги з приводу єдиного модулю завдань викладача

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import DatePicker from "react-datepicker";
import "react-datepicker/dist/react-datepicker.css";
import { useTranslation } from "react-i18next";
import Navigation from "../navigation/Navigation"
import { Helmet } from 'react-helmet';
import 'bootstrap/dist/css/bootstrap.min.css';

import './TeacherHomeworkPage.css';

const TeacherHomeworkPage = () => {
  const [assignments, setAssignments] = useState([]);
  const [isLoading, setIsLoading] = useState(true);
  const [deadline, setDeadline] = useState(null);
  const [newAssignmentData, setNewAssignmentData] = useState({
    assignment_file: null,
    subject_group_name: '',
    subject_name: '',
    assignment_name: ''
  });
  const [requestSuccess, setRequestSuccess] = useState(false);
  const [errorMessage, setErrorMessage] = useState('');
  const [messageVisible, setMessageVisible] = useState(false);
  const [markInputs, setMarkInputs] = useState({});
  const [isAddingAssignment, setIsAddingAssignment] = useState(false);
  const [expandedSubjects, setExpandedSubjects] = useState({});
  const [expandedGroups, setExpandedGroups] = useState({});
  const [expandedAssignments, setExpandedAssignments] = useState({});

  const { t, i18n } = useTranslation();

  // Fetch assignments from the backend
  useEffect(() => {
    fetchAssignments();
  }, []);
```

```
  const fetchAssignments = async () => {
    try {
      const response = await axios.get('http://localhost:8080/v1/subject/assignments/teacher', {
        headers: {
          Authorization: `${localStorage.getItem('jwtToken')}`
        }
      });
      const fetchedAssignments = response.data.map(assignment => ({
        ...assignment,
        solutions: assignment.solutions || []
      }));
      setAssignments(fetchedAssignments);
      setIsLoading(false);
    } catch (error) {
      console.error('Error fetching assignments:', error);
      setIsLoading(false);
    }
  };

  const handleAddAssignment = () => {
    setIsAddingAssignment(prevState => !prevState);
  };

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setNewAssignmentData({
      ...newAssignmentData,
      [name]: value
    });
  };

  const handleFileChange = (e) => {
    setNewAssignmentData({
      ...newAssignmentData,
      assignment_file: e.target.files[0]
    });
  };

  const handleSubmit = async (e) => {
    setMessageVisible(false);
    e.preventDefault();
    try {
      const jwtToken = localStorage.getItem('jwtToken');
      const formData = new FormData();
      formData.append('assignment_file', newAssignmentData.assignment_file);
      formData.append('deadline', deadline.toISOString());
      formData.append('subject_group_name', newAssignmentData.subject_group_name);
      formData.append('subject_name', newAssignmentData.subject_name);
      formData.append('assignment_name', newAssignmentData.assignment_name);

      const response = await axios.post('http://localhost:8080/v1/subject/assignment/add',
formData, {
        headers: {
          'Authorization': `${jwtToken}`,
          'Content-Type': 'multipart/form-data'
        }
      });

      const formatDeadline = (deadline) => {
        let date = new Date(deadline);
        let year = date.getFullYear();
        let month = String(date.getMonth() + 1).padStart(2, '0');
        let day = String(date.getDate()).padStart(2, '0');
        let hours = String(date.getHours()).padStart(2, '0');
        let minutes = String(date.getMinutes()).padStart(2, '0');
        let seconds = String(date.getSeconds()).padStart(2, '0');
        return `${year}-${month}-${day} ${hours}:${minutes}:${seconds}`;
      };

      if (response.status === 200) {
        setRequestSuccess(true);
        const newAssignment = response.data;
        const assignmentToAdd = {
          id: newAssignment.id,
          assignment_name: newAssignmentData.assignment_name,
          subject_name: newAssignmentData.subject_name,
          subject_group_name: newAssignmentData.subject_group_name,
          deadline: formatDeadline(deadline.toISOString()),
          s3_location: newAssignment.s3_location,
          solutions: []
```

```
        };

        setAssignments(prevAssignments => [...prevAssignments, assignmentToAdd]);

        setNewAssignmentData({
          assignment_file: null,
          subject_group_name: '',
          subject_name: '',
          assignment_name: ''
        });
      } else {
        setRequestSuccess(false);
        setErrorMessage(response.data.error || 'Unknown error');
      }
    } catch (error) {
      setRequestSuccess(false);
      setErrorMessage(error.response ? error.response.data.error : 'Unknown error');
    }
    setMessageVisible(true);
  };

  const handleMarkChange = (e, uniqueKey) => {
    const value = e.target.value;
    setMarkInputs({
      ...markInputs,
      [uniqueKey]: value
    });
  };

  const handleMarkSubmit = async (assignmentId, solutionId, s3Location) => {
    const uniqueKey = `${assignmentId}-${solutionId}`;
    const mark = markInputs[uniqueKey];
    if (!mark || isNaN(mark) || parseInt(mark) < 0) {
      alert('Please enter a valid positive number for the mark.');
      return;
    }
    try {
      const jwtToken = localStorage.getItem('jwtToken');
      const response = await axios.post(`http://localhost:8080/v1/subject/assignment/mark/add`, {
        mark: parseInt(mark),
        s3_location: s3Location
      }, {
        headers: {
          'Authorization': `${jwtToken}`
        }
      });
      if (response.status === 200) {
        await fetchAssignments(); // Fetch updated data after successful submission
        setMarkInputs({ ...markInputs, [uniqueKey]: '' }); // Clear the mark input for the
submitted solution
      } else {
        console.error('Error submitting mark:', response.data);
      }
    } catch (error) {
      console.error('Error submitting mark:', error);
    }
  };

  const toggleSubject = (subjectName) => {
    setExpandedSubjects({
      ...expandedSubjects,
      [subjectName]: !expandedSubjects[subjectName]
    });
  };

  const toggleGroup = (subjectName, groupName) => {
    setExpandedGroups({
      ...expandedGroups,
      [subjectName + '-' + groupName]: !expandedGroups[subjectName + '-' + groupName]
    });
  };

  const toggleAssignment = (assignmentId) => {
    setExpandedAssignments({
      ...expandedAssignments,
      [assignmentId]: !expandedAssignments[assignmentId]
    });
  };

  // Fetch presigned URL for a specific S3 key
  const fetchPresignedUrl = async (s3Key) => {
```

```
    try {
      const response = await axios.get(`http://localhost:8080/v1/presigned-
url?s3_key=${encodeURIComponent(s3Key)}`, {
        headers: {
          Authorization: `${localStorage.getItem('jwtToken')}`
        }
      });
      return response.data.url;
    } catch (error) {
      console.error('Error fetching presigned URL:', error);
      return null;
    }
  };

  // Handle file download
  const handleFileDownload = async (s3Key) => {
    const presignedUrl = await fetchPresignedUrl(s3Key);
    if (presignedUrl) {
      window.open(presignedUrl, '_blank');
    } else {
      alert('Failed to generate download link. Please try again later.');
    }
  };

  return (
    <div>
      <Helmet>
        <title>{t("nameOfProject")} | {t("assignments")}</title>
      </Helmet>
      <Navigation jwtToken={localStorage.getItem('jwtToken')} />
      <div className="teacher-homework-page">
        <h1>{t('teacherAssignmentPage')}</h1>
        {!isAddingAssignment && <button className='btn btn-outline-primary'
onClick={handleAddAssignment}>{t('addNewAssignment')}</button>}
        {isAddingAssignment && (
          <div className='card'>
            <div className="card-body">
              <h2 className='card-title'>{t('addNewAssignment')}
                {isAddingAssignment && <button onClick={handleAddAssignment} type="button"
className="btn btn-outline-dark btn-sm">X</button>}
              </h2>
              <form onSubmit={handleSubmit}>
                <div className="subject">
                  <input type="text" name="subject_group_name"
placeholder={t('subjectGroupName')} value={newAssignmentData.subject_group_name}
onChange={handleInputChange} />
                  <input type="text" name="subject_name" placeholder={t('subjectName')}
value={newAssignmentData.subject_name} onChange={handleInputChange} />
                </div>
                <input type="text" name="assignment_name" placeholder={t('assignmentName')}
value={newAssignmentData.assignment_name} onChange={handleInputChange} />
                <DatePicker
                  selected={deadline}
                  onChange={(date) => setDeadline(date)}
                  wrapperClassName="datePicker"
                  showTimeSelect
                  dateFormat="MMMM d, yyyy h:mm aa"
                  placeholderText={t('deadline')}
                  locale={i18n.language}
                />
                <input type="file" className="form-control" name="assignment_file"
onChange={handleFileChange}></input>
                <button className='btn btn-outline-success' type="submit">{t('submit')}</button>
              </form>
            </div>
          </div>
        )}
        <div className="message">
          {messageVisible ? (requestSuccess ? 'Status: Success' : `Status: ${errorMessage}`) :
''}
        </div>
        {isLoading ? (
          <p>Loading...</p>
        ) : (
          <div>
            {assignments.length === 0 ? (
              <p>{t('noAssignmentsAvailable')}</p>
            ) : (
              Object.values(assignments.reduce((acc, assignment) => {
                const key = `${assignment.subject_name}-${assignment.subject_group_name}`;
                if (!acc[key]) {
```

```
                  acc[key] = [];
                }
                acc[key].push(assignment);
                return acc;
              }, {})).map((groupedAssignments) => (
                <div key={groupedAssignments[0].subject_name + '-' +
groupedAssignments[0].subject_group_name} className="card mb-3">
                  <div className="card-header clickable" onClick={() =>
toggleSubject(groupedAssignments[0].subject_name)}>
                    <h2 className="card-title">{groupedAssignments[0].subject_name}</h2>
                  </div>
                  {expandedSubjects[groupedAssignments[0].subject_name] && (
                    <div className="card-body">
                      {groupedAssignments.map((assignment) => (
                        <div key={assignment.subject_group_name} className="card mb-2">
                          <div className="card-header clickable" onClick={() =>
toggleGroup(assignment.subject_name, assignment.subject_group_name)}>
                            <h3 className="card-title">{assignment.subject_group_name}</h3>
                          </div>
                          {expandedGroups[assignment.subject_name + '-' +
assignment.subject_group_name] && (
                            <div className="card-body">
                              {groupedAssignments.map((assignment) => (
                                <div key={assignment.id} className="card mb-2">
                                  <div className="card-header clickable" onClick={() =>
toggleAssignment(assignment.id)}>
                                    <h3 className="card-title">{assignment.assignment_name}</h3>
                                  </div>
                                  {expandedAssignments[assignment.id] && (
                                    <div className="card-body">
                                      <button className="btn btn-outline-secondary" onClick={()
=> handleFileDownload(assignment.s3_location)}>
                                        {t('downloadAssignment')}
                                      </button>
                                      <p>{t('deadline')}: {assignment.deadline}</p>
                                      <h4>{t('solutions')}:</h4>
                                      {assignment.solutions.length > 0 ? (
                                        assignment.solutions.map((solution, i) => {
                                          const uniqueKey = `${assignment.id}-${solution.id}`;
                                          return (
                                            <div key={i} className="solution">
                                              <p>{t('student')}: {solution.surname}
{solution.last_name}</p>
                                              <p>
                                                <button className="btn btn-outline-secondary"
onClick={() => handleFileDownload(solution.s3_location)}>
                                                  {t('downloadSolution')}
                                                </button>
                                              </p>
                                              {solution.mark ? (
                                                <p>{t('mark')}: {solution.mark}</p>
                                              ) : (
                                                <div>
                                                  <label>{t('mark')}</label>
                                                  <input
                                                    type="text"
                                                    placeholder="Input mark"
                                                    value={markInputs[uniqueKey] || ''}
                                                    onChange={(e) => handleMarkChange(e,
uniqueKey)}
                                                  />
                                                  <button className="btn btn-outline-success"
onClick={() => handleMarkSubmit(assignment.id, solution.id,
solution.s3_location)}>{t('submit')}</button>
                                                </div>
                                              )}
                                            </div>
                                          );
                                        })
                                      ) : (
                                        <p>{t('noSolutionsAvailable')}</p>
                                      )}
                                    </div>
                                  )}
                                </div>
                              ))}
                            </div>
                          )}
                        </div>
                      ))}
                    </div>
                  )}
                </div>
```

```
                )}
              </div>
            ))
          )}
        </div>
      )}
    </div>
  </div>
  );
};

export default TeacherHomeworkPage;
```

Файл AdminPage.jsx

Реалізовує адміністративну панель

```
import React, { useState } from 'react';
import DepartmentForm from './forms/DepartmentForm';
import DepartmentGroupForm from './forms/DepartmentGroupForm';
import FacultyForm from './forms/FacultyForm'; // Импортируем новую форму
import StudentForm from './forms/StudentForm';
import AddStudentToGroupForm from './forms/AddStudentToGroupForm';
import AddSubjectToDepartmentForm from './forms/AddSubjectToDepartmentGroup'
import AddSubjectGroupForm from './forms/AddSubjectGroupForm';
import AddSubjectGroupToDepartmentGroupForm from './forms/AddSubjectGroupToDepartmentGroupForm';
import AddTeacherForm from './forms/AddTeacherForm';
import AddTeacherToFacultyForm from './forms/AddTeacherToFacultyForm';
import AddTeacherRankForm from './forms/AddTeacherRankForm'; // Импортируем новую форму
import AddUniversityForm from './forms/AddUniversityForm';
import Navigation from '../navigation/Navigation';
import { useTranslation } from 'react-i18next';
import { useJwt } from "react-jwt";
import { Helmet } from 'react-helmet';

import './AdminPage.css'

const AdminPage = () => {
  // Состояния для хранения данных форм
  const [departmentFormData, setDepartmentFormData] = useState({});
  const [groupFormData, setGroupFormData] = useState({});
  const [facultyFormData, setFacultyFormData] = useState({}); // Добавляем состояние для данных
формы факультета
  const [studentFormData, setStudentFormData] = useState({});
  const [studentToGroupFormData, setStudentToGroupFormData] = useState({});
  const [subjectToDepartmentFormData, setSubjectToDepartmentFormData] = useState({});
  const [subjectGroupFormData, setSubjectGroupFormData] = useState({}); // Добавляем состояние
для данных формы добавления группы предметов
  const [subjectGroupToDepartmentGroupFormData, setSubjectGroupToDepartmentGroupFormData] =
useState({}); // Добавляем состояние для данных формы добавления группы предметов в качестве
группы отдела
  const [teacherFormData, setTeacherFormData] = useState({});
  const [teacherToFacultyFormData, setTeacherToFacultyFormData] = useState({}); // Добавляем
состояние для данных формы добавления учителя на факультет
  const [teacherRankFormData, setTeacherRankFormData] = useState({}); // Добавляем состояние для
данных формы добавления звания учителя
  const [universityFormData, setUniversityFormData] = useState({});
  const [requestSuccess, setRequestSuccess] = useState(false);
  const [errorMessage, setErrorMessage] = useState('');
```

```
  const [messageVisible, setMessageVisible] = useState(false);

  // Состояния для отслеживания видимости форм
  const [showDepartmentForm, setShowDepartmentForm] = useState(false);
  const [showGroupForm, setShowGroupForm] = useState(false);
  const [showFacultyForm, setShowFacultyForm] = useState(false); // Добавляем состояние для
отображения формы факультета
  const [showStudentForm, setShowStudentForm] = useState(false); // Добавляем состояние для
отображения формы студента
  const [showStudentToGroupForm, setShowStudentToGroupForm] = useState(false); // Добавляем
состояние для отображения формы добавления студента в группу предметов
  const [showSubjectToDepartmentForm, setShowSubjectToDepartmentForm] = useState(false);
  const [showSubjectGroupForm, setShowSubjectGroupForm] = useState(false); // Добавляем состояние
для отображения формы добавления группы предметов
  const [showSubjectGroupToDepartmentGroupForm, setShowSubjectGroupToDepartmentGroupForm] =
useState(false); // Добавляем состояние для отображения формы добавления группы предметов в
качестве группы отдела
  const [showTeacherForm, setShowTeacherForm] = useState(false);
  const [showTeacherToFacultyForm, setShowTeacherToFacultyForm] = useState(false); // Добавляем
состояние для отображения формы добавления учителя на факультет
  const [showTeacherRankForm, setShowTeacherRankForm] = useState(false); // Добавляем состояние
для отображения формы добавления звания учителя
  const [showUniversityForm, setShowUniversityForm] = useState(false); // Добавляем состояние для
отображения формы добавления университета

  // Обработчики изменения данных форм
  const handleDepartmentInputChange = (event) => {
    const { name, value } = event.target;
    setDepartmentFormData({ ...departmentFormData, [name]: value });
  };

  const handleGroupInputChange = (event) => {
    const { name, value } = event.target;
    setGroupFormData({ ...groupFormData, [name]: value });
  };

  const handleFacultyInputChange = (event) => { // Добавляем обработчик изменения данных формы
факультета
    const { name, value } = event.target;
    setFacultyFormData({ ...facultyFormData, [name]: value });
  };

  const handleStudentInputChange = (event) => { // Добавляем обработчик изменения данных формы
студента
    const { name, value } = event.target;
    setStudentFormData({ ...studentFormData, [name]: value });
  };

  const handleStudentToGroupInputChange = (event) => { // Добавляем обработчик изменения данных
формы добавления студента в группу предметов
    const { name, value } = event.target;
    setStudentToGroupFormData({ ...studentToGroupFormData, [name]: value });
  };
```

```
  const handleSubjectToDepartmentInputChange = (event) => { // Добавляем обработчик изменения данных формы добавления предмета в отдел
    const { name, value } = event.target;
    setSubjectToDepartmentFormData({ ...subjectToDepartmentFormData, [name]: value });
  };

  const handleSubjectGroupInputChange = (event) => { // Добавляем обработчик изменения данных формы добавления группы предметов
    const { name, value } = event.target;
    setSubjectGroupFormData({ ...subjectGroupFormData, [name]: value });
  };

  const handleSubjectGroupToDepartmentGroupInputChange = (event) => { // Добавляем обработчик изменения данных формы добавления группы предметов в качестве группы отдела
    const { name, value } = event.target;
    setSubjectGroupToDepartmentGroupFormData({ ...subjectGroupToDepartmentGroupFormData, [name]: value });
  };

  const handleTeacherInputChange = (event) => {
    const { name, value } = event.target;
    setTeacherFormData({ ...teacherFormData, [name]: value });
  };

  const handleTeacherToFacultyInputChange = (event) => { // Добавляем обработчик изменения данных формы добавления учителя на факультет
    const { name, value } = event.target;
    setTeacherToFacultyFormData({ ...teacherToFacultyFormData, [name]: value });
  };

  const handleTeacherRankInputChange = (event) => { // Добавляем обработчик изменения данных формы добавления звания учителя
    const { name, value } = event.target;
    setTeacherRankFormData({ ...teacherRankFormData, [name]: value });
  };

  const handleUniversityInputChange = (event) => { // Добавляем обработчик изменения данных формы добавления университета
    const { name, value } = event.target;
    setUniversityFormData({ ...universityFormData, [name]: value });
  };

  // Функции для отправки данных на сервер
  const handleDepartmentSubmit = async (event) => {
    event.preventDefault();
    try {
      const response = await fetch('http://localhost:8080/v1/admin/department/add', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': localStorage.getItem('jwtToken')
        },
        body: JSON.stringify(departmentFormData),
      });
```

```
    if (response.ok)
    {
      setRequestSuccess(true);
      const data = await response.json();
      console.log('Ответ сервера:', data);
      // Очистка данных формы после успешной отправки
      setDepartmentFormData({});
    }
    else
    {
      setRequestSuccess(false);
      if (response.status === 404) {
        setErrorMessage('Not found');
      } else {
        const errorData = await response.json();
        setErrorMessage(errorData.error || 'Unknown error');
        console.error('Ошибка:', errorData);
      }
    }
  } catch (error) {
    setErrorMessage('Unknown error');
    setRequestSuccess(false);
    console.error('Ошибка:', error);
  }
  setMessageVisible(true);
};

const handleGroupSubmit = async (event) => {
  event.preventDefault();
  try {
    const response = await fetch('http://localhost:8080/v1/admin/department/group/add', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': localStorage.getItem('jwtToken')
      },
      body: JSON.stringify(groupFormData),
    });
    if (response.ok)
      {
        setRequestSuccess(true);
        const data = await response.json();
        console.log('Ответ сервера:', data);
        // Очистка данных формы после успешной отправки
        setGroupFormData({});
      }
      else
      {
        setRequestSuccess(false);
        if (response.status === 404) {
          setErrorMessage('Not found');
        } else {
          const errorData = await response.json();
          setErrorMessage(errorData.error || 'Unknown error');
```

```
            console.error('Ошибка:', errorData);
          }
        }
      } catch (error) {
        setRequestSuccess(false);
        setErrorMessage('Unknown error');
      }
      setMessageVisible(true);
  };

  const handleFacultySubmit = async (event) => { // Добавляем функцию для отправки данных формы факультета
    event.preventDefault();
    try {
      const response = await fetch('http://localhost:8080/v1/admin/faculty/add', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': localStorage.getItem('jwtToken')
        },
        body: JSON.stringify(facultyFormData),
      });
      if (response.ok)
        {
          setRequestSuccess(true);
          const data = await response.json();
          console.log('Ответ сервера:', data);
          // Очистка данных формы после успешной отправки
          setFacultyFormData({});
        }
        else
        {
          setRequestSuccess(false);
          if (response.status === 404) {
            setErrorMessage('Not found');
          } else {
            const errorData = await response.json();
            setErrorMessage(errorData.error || 'Unknown error');
            console.error('Ошибка:', errorData);
          }
        }
      } catch (error) {
        setRequestSuccess(false);
        setErrorMessage('Unknown error');
      }
      setMessageVisible(true);
  };

  const handleStudentSubmit = async (event) => { // Добавляем функцию для отправки данных формы студента
    event.preventDefault();
    try {
      const response = await fetch('http://localhost:8080/v1/admin/student/add', {
        method: 'POST',
```

```
        headers: {
          'Content-Type': 'application/json',
          'Authorization': localStorage.getItem('jwtToken')
        },
        body: JSON.stringify(studentFormData),
      });
      if (response.ok)
        {
          setRequestSuccess(true);
          const data = await response.json();
          console.log('Ответ сервера:', data);
          // Очистка данных формы после успешной отправки
          setStudentFormData({});
        }
        else
        {
          setRequestSuccess(false);
          if (response.status === 404) {
            setErrorMessage('Not found');
          } else {
            const errorData = await response.json();
            setErrorMessage(errorData.error || 'Unknown error');
            console.error('Ошибка:', errorData);
          }
        }
      } catch (error) {
        setRequestSuccess(false);
        setErrorMessage('Unknown error');
      }
      setMessageVisible(true);
  };

  const handleStudentToGroupSubmit = async (event) => { // Добавляем функцию для отправки данных
формы добавления студента в группу предметов
    event.preventDefault();
    try {
      const response = await fetch('http://localhost:8080/v1/admin/department/group/add_student',
{
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': localStorage.getItem('jwtToken')
        },
        body: JSON.stringify(studentToGroupFormData),
      });
      if (response.ok)
        {
          setRequestSuccess(true);
          const data = await response.json();
          console.log('Ответ сервера:', data);
          // Очистка данных формы после успешной отправки
          setStudentToGroupFormData({});
        }
        else
```

```
        {
          setRequestSuccess(false);
          if (response.status === 404) {
            setErrorMessage('Not found');
          } else {
            const errorData = await response.json();
            setErrorMessage(errorData.error || 'Unknown error');
            console.error('Ошибка:', errorData);
          }
        }
      } catch (error) {
        setRequestSuccess(false);
        setErrorMessage('Unknown error');
      }
      setMessageVisible(true);
  };

  const handleSubjectToDepartmentSubmit = async (event) => { // Добавляем функцию для отправки
данных формы добавления предмета в отдел
    event.preventDefault();
    try {
      const response = await fetch('http://localhost:8080/v1/admin/department/subject/add', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': localStorage.getItem('jwtToken')
        },
        body: JSON.stringify(subjectToDepartmentFormData),
      });
      if (response.ok)
        {
          setRequestSuccess(true);
          const data = await response.json();
          console.log('Ответ сервера:', data);
          // Очистка данных формы после успешной отправки
          setSubjectToDepartmentFormData({});
        }
        else
        {
          setRequestSuccess(false);
          if (response.status === 404) {
            setErrorMessage('Not found');
          } else {
            const errorData = await response.json();
            setErrorMessage(errorData.error || 'Unknown error');
            console.error('Ошибка:', errorData);
          }
        }
      } catch (error) {
        setRequestSuccess(false);
        setErrorMessage('Unknown error');
      }
      setMessageVisible(true);
  };
```

```
  const handleSubjectGroupSubmit = async (event) => { // Добавляем функцию для отправки данных формы добавления группы предметов
    event.preventDefault();
    try {
      const response = await fetch('http://localhost:8080/v1/admin/subject/group/add', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': localStorage.getItem('jwtToken')
        },
        body: JSON.stringify(subjectGroupFormData),
      });
      if (response.ok)
        {
          setRequestSuccess(true);
          const data = await response.json();
          console.log('Ответ сервера:', data);
          // Очистка данных формы после успешной отправки
          setSubjectGroupFormData({});
        }
        else
        {
          setRequestSuccess(false);
          if (response.status === 404) {
            setErrorMessage('Not found');
          } else {
            const errorData = await response.json();
            setErrorMessage(errorData.error || 'Unknown error');
            console.error('Ошибка:', errorData);
          }
        }
      } catch (error) {
        setRequestSuccess(false);
        setErrorMessage('Unknown error');
      }
      setMessageVisible(true);
  };

  const handleSubjectGroupToDepartmentGroupSubmit = async (event) => { // Добавляем функцию для отправки данных формы добавления группы предметов в качестве группы отдела
    event.preventDefault();
    try {
      const response = await fetch('http://localhost:8080/v1/admin/subject/group/department_group_add', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': localStorage.getItem('jwtToken')
        },
        body: JSON.stringify(subjectGroupToDepartmentGroupFormData),
      });
      if (response.ok)
        {
```

```
        setRequestSuccess(true);
        const data = await response.json();
        console.log('Ответ сервера:', data);
        // Очистка данных формы после успешной отправки
        setSubjectGroupToDepartmentGroupFormData({});
      }
      else
      {
        setRequestSuccess(false);
        if (response.status === 404) {
          setErrorMessage('Not found');
        } else {
          const errorData = await response.json();
          setErrorMessage(errorData.error || 'Unknown error');
          console.error('Ошибка:', errorData);
        }
      }
    } catch (error) {
      setRequestSuccess(false);
      setErrorMessage('Unknown error');
    }
    setMessageVisible(true);
};

const handleTeacherSubmit = async (event) => {
  event.preventDefault();
  try {
    const response = await fetch('http://localhost:8080/v1/admin/teacher/add', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': localStorage.getItem('jwtToken')
      },
      body: JSON.stringify(teacherFormData),
    });
    if (response.ok)
      {
        setRequestSuccess(true);
        const data = await response.json();
        console.log('Ответ сервера:', data);
        // Очистка данных формы после успешной отправки
        setTeacherFormData({});
      }
      else
      {
        setRequestSuccess(false);
        if (response.status === 404) {
          setErrorMessage('Not found');
        } else {
          const errorData = await response.json();
          setErrorMessage(errorData.error || 'Unknown error');
          console.error('Ошибка:', errorData);
        }
      }
```

```
      } catch (error) {
        setRequestSuccess(false);
        setErrorMessage('Unknown error');
      }
      setMessageVisible(true);
  };

  const handleTeacherToFacultySubmit = async (event) => { // Добавляем функцию для отправки
данных формы добавления учителя на факультет
    event.preventDefault();
    try {
      const response = await fetch('http://localhost:8080/v1/admin/faculty/add_teacher', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': localStorage.getItem('jwtToken')
        },
        body: JSON.stringify(teacherToFacultyFormData),
      });
      if (response.ok)
        {
          setRequestSuccess(true);
          const data = await response.json();
          console.log('Ответ сервера:', data);
          // Очистка данных формы после успешной отправки
          setTeacherToFacultyFormData({});
        }
        else
        {
          setRequestSuccess(false);
          if (response.status === 404) {
            setErrorMessage('Not found');
          } else {
            const errorData = await response.json();
            setErrorMessage(errorData.error || 'Unknown error');
            console.error('Ошибка:', errorData);
          }
        }
      } catch (error) {
        setRequestSuccess(false);
        setErrorMessage('Unknown error');
      }
      setMessageVisible(true);
  };

  const handleTeacherRankSubmit = async (event) => { // Добавляем функцию для отправки данных
формы добавления звания учителя
    event.preventDefault();
    try {
      const response = await fetch('http://localhost:8080/v1/admin/teacher/rank/add', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': localStorage.getItem('jwtToken')
```

```
        },
        body: JSON.stringify(teacherRankFormData),
      });
      if (response.ok)
        {
          setRequestSuccess(true);
          const data = await response.json();
          console.log('Ответ сервера:', data);
          // Очистка данных формы после успешной отправки
          setTeacherRankFormData({});
        }
        else
        {
          setRequestSuccess(false);
          if (response.status === 404) {
            setErrorMessage('Not found');
          } else {
            const errorData = await response.json();
            setErrorMessage(errorData.error || 'Unknown error');
            console.error('Ошибка:', errorData);
          }
        }
      } catch (error) {
        setRequestSuccess(false);
        setErrorMessage('Unknown error');
      }
      setMessageVisible(true);
  };

  const handleUniversitySubmit = async (event) => { // Добавляем функцию для отправки данных
формы добавления университета
    event.preventDefault();
    try {
      const response = await fetch('http://localhost:8080/v1/admin/university/add', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': localStorage.getItem('jwtToken')
        },
        body: JSON.stringify(universityFormData),
      });
      if (response.ok)
        {
          setRequestSuccess(true);
          const data = await response.json();
          console.log('Ответ сервера:', data);
          // Очистка данных формы после успешной отправки
          setUniversityFormData({});
        }
        else
        {
          setRequestSuccess(false);
          if (response.status === 404) {
            setErrorMessage('Not found');
```

```
          } else {
            const errorData = await response.json();
            setErrorMessage(errorData.error || 'Unknown error');
            console.error('Ошибка:', errorData);
          }
        }
      } catch (error) {
        setRequestSuccess(false);
        setErrorMessage('Unknown error');
      }
      setMessageVisible(true);
  };

  // Функция для переключения видимости формы
  const toggleTeacherFormVisibility = () => {
    setMessageVisible(false);
    setShowTeacherForm(!showTeacherForm);
  };


  // Функции для переключения видимости форм
  const toggleDepartmentFormVisibility = () => {
    setMessageVisible(false);
    setShowDepartmentForm(!showDepartmentForm);
  };

  const toggleGroupFormVisibility = () => {
    setMessageVisible(false);
    setShowGroupForm(!showGroupForm);
  };

  const toggleFacultyFormVisibility = () => { // Добавляем функцию для переключения видимости
формы факультета
    setMessageVisible(false);
    setShowFacultyForm(!showFacultyForm);
  };


  const toggleStudentFormVisibility = () => { // Добавляем функцию для переключения видимости
формы студента
    setMessageVisible(false);
    setShowStudentForm(!showStudentForm);
  };

  const toggleStudentToGroupFormVisibility = () => { // Добавляем функцию для переключения
видимости формы добавления студента в группу предметов
    setMessageVisible(false);
    setShowStudentToGroupForm(!showStudentToGroupForm);
  };

  const toggleSubjectToDepartmentFormVisibility = () => { // Добавляем функцию для переключения
видимости формы добавления предмета в отдел
    setMessageVisible(false);
    setShowSubjectToDepartmentForm(!showSubjectToDepartmentForm);
```

```
  };

  const toggleSubjectGroupFormVisibility = () => { // Добавляем функцию для переключения
видимости формы добавления группы предметов
    setMessageVisible(false);
    setShowSubjectGroupForm(!showSubjectGroupForm);
  };

  const toggleSubjectGroupToDepartmentGroupFormVisibility = () => { // Добавляем функцию для
переключения видимости формы добавления группы предметов в качестве группы отдела
    setMessageVisible(false);
    setShowSubjectGroupToDepartmentGroupForm(!showSubjectGroupToDepartmentGroupForm);
  };

  const toggleTeacherToFacultyFormVisibility = () => { // Добавляем функцию для переключения
видимости формы добавления учителя на факультет
    setMessageVisible(false);
    setShowTeacherToFacultyForm(!showTeacherToFacultyForm);
  };

  const toggleTeacherRankFormVisibility = () => { // Добавляем функцию для переключения видимости
формы добавления звания учителя
    setMessageVisible(false);
    setShowTeacherRankForm(!showTeacherRankForm);
  };

  const toggleUniversityFormVisibility = () => { // Добавляем функцию для переключения видимости
формы добавления университета
    setMessageVisible(false);
    setShowUniversityForm(!showUniversityForm);
  };

  const { t } = useTranslation();

  const jwtToken = localStorage.getItem('jwtToken');
  const { decodedToken, isExpired } = useJwt(jwtToken);
  const level = decodedToken?.level;
  console.log("level: ", level);
  console.log("decodedToken: ", decodedToken);

  return (
    <div>
      <Helmet>
        <title>{t("nameOfProject")} | {t("admin")}</title>
      </Helmet>

      <Navigation jwtToken={localStorage.getItem('jwtToken')} />
      <div className='AdminPage'>
        <h1>{t('adminPage')}</h1>
        <div className="message">
          {messageVisible ? (requestSuccess ? 'Status: Success' : `Status: ${errorMessage}`) :
''}
        </div>
        <div className='button-matrix'>
```

```
          { level === "full" && !showUniversityForm &&
            <button onClick={toggleUniversityFormVisibility} className='btn btn-outline-
primary'>{t('createUniversity')}</button>
          }
          {showUniversityForm && (
              <AddUniversityForm onSubmit={handleUniversitySubmit}
onChange={handleUniversityInputChange} formData={universityFormData}
setShowForm={setShowUniversityForm}/>
          )}

          { ((level === "full" || level === "faculty") && !showFacultyForm) &&
            <button onClick={toggleFacultyFormVisibility} className='btn btn-outline-
primary'>{t('createFaculty')}</button>
          }
          {showFacultyForm && (
              <FacultyForm onSubmit={handleFacultySubmit} onChange={handleFacultyInputChange}
formData={facultyFormData} setShowForm={setShowFacultyForm} />
          )}

          { ((level === "full" || level === "faculty" || level === "department") &&
!showDepartmentForm) &&
            <button onClick={toggleDepartmentFormVisibility} className='btn btn-outline-
primary'>{t('createDepartment')}</button>
          }
          {showDepartmentForm && (
              <DepartmentForm onSubmit={handleDepartmentSubmit}
onChange={handleDepartmentInputChange} formData={departmentFormData}
setShowForm={setShowDepartmentForm}/>
          )}

          { ((level === "full" || level === "faculty") && !showTeacherRankForm) &&
            <button onClick={toggleTeacherRankFormVisibility} className='btn btn-outline-
primary'>{t('createTeacherRank')} </button>
          }
          {showTeacherRankForm && (
              <AddTeacherRankForm onSubmit={handleTeacherRankSubmit}
onChange={handleTeacherRankInputChange} formData={teacherRankFormData}
setShowForm={setShowTeacherRankForm}/>
          )}

          { ((level === "full" || level === "faculty") && !showTeacherForm) &&
            <button onClick={toggleTeacherFormVisibility} className='btn btn-outline-
primary'>{t('addTeacher')}</button>
          }
          {showTeacherForm && (
              <AddTeacherForm onSubmit={handleTeacherSubmit} onChange={handleTeacherInputChange}
formData={teacherFormData} setShowForm={setShowTeacherForm}/>
          )}

          { ((level === "full" || level === "faculty") && !showTeacherToFacultyForm) &&
            <button onClick={toggleTeacherToFacultyFormVisibility} className='btn btn-outline-
primary'>{t('addTeacherToFaculty')}</button>
          }
          {showTeacherToFacultyForm && (
```

```
              <AddTeacherToFacultyForm onSubmit={handleTeacherToFacultySubmit}
onChange={handleTeacherToFacultyInputChange} formData={teacherToFacultyFormData}
setShowForm={setTeacherToFacultyFormData}/>
          )}

          { ((level === "full" || level === "faculty" || level === "department") &&
!showGroupForm) &&
            <button onClick={toggleGroupFormVisibility} className='btn btn-outline-
primary'>{t('createDepartmentGroup')}</button>
          }
          {showGroupForm && (
              <DepartmentGroupForm onSubmit={handleGroupSubmit} onChange={handleGroupInputChange}
formData={groupFormData} setShowForm={setGroupFormData}/>
          )}

          { ((level === "full" || level === "faculty" || level === "department") &&
!showStudentForm) &&
            <button onClick={toggleStudentFormVisibility} className='btn btn-outline-
primary'>{t('addStudentToDepartmentGroup')}</button>
          }
          {showStudentForm && (
              <StudentForm onSubmit={handleStudentSubmit} onChange={handleStudentInputChange}
formData={studentFormData} setShowForm={setStudentFormData}/>
          )}

          { ((level === "full" || level === "faculty" || level === "department") &&
!showSubjectToDepartmentForm) &&
            <button onClick={toggleSubjectToDepartmentFormVisibility} className='btn btn-outline-
primary'>{t('createSubjectToDepartment')}</button>
          }
          {showSubjectToDepartmentForm && (
              <AddSubjectToDepartmentForm onSubmit={handleSubjectToDepartmentSubmit}
onChange={handleSubjectToDepartmentInputChange} formData={subjectToDepartmentFormData}
setShowForm={setSubjectToDepartmentFormData}/>
          )}

          { ((level === "full" || level === "faculty" || level === "department") &&
!showSubjectGroupForm) &&
            <button onClick={toggleSubjectGroupFormVisibility} className='btn btn-outline-
primary'>{t('createSubjectGroup')}</button>
          }
          {showSubjectGroupForm && (
              <AddSubjectGroupForm onSubmit={handleSubjectGroupSubmit}
onChange={handleSubjectGroupInputChange} formData={subjectGroupFormData}
setShowForm={setSubjectGroupFormData}/>
          )}

          { ((level === "full" || level === "faculty" || level === "department") &&
!showStudentToGroupForm) &&
            <button onClick={toggleStudentToGroupFormVisibility} className='btn btn-outline-
primary'>{t('addStudentToSubjectGroup')}</button>
          }
          {showStudentToGroupForm && (
```

```
              <AddStudentToGroupForm onSubmit={handleStudentToGroupSubmit}
onChange={handleStudentToGroupInputChange} formData={studentToGroupFormData}
setShowForm={setStudentToGroupFormData}/>
          )}

          { ((level === "full" || level === "faculty" || level === "department") &&
!showSubjectGroupToDepartmentGroupForm) &&
            <button onClick={toggleSubjectGroupToDepartmentGroupFormVisibility} className='btn
btn-outline-primary'>{t('createSubjectGroupAsDepartmentGroup')}</button>
          }
          {showSubjectGroupToDepartmentGroupForm && (
              <AddSubjectGroupToDepartmentGroupForm
onSubmit={handleSubjectGroupToDepartmentGroupSubmit}
onChange={handleSubjectGroupToDepartmentGroupInputChange}
formData={subjectGroupToDepartmentGroupFormData}
setShowForm={setSubjectGroupToDepartmentGroupFormData}/>
          )}
        </div>
      </div>
    </div>
  );
};

export default AdminPage;
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНОГО НАВЧАННЯ

Програма та методика тестування

КПІ.ІХ-0217.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Сергій ТИХОНОВ

Нормоконтроль:
_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:
_____ Олексій ДЕМЧЕНКО

Київ – 2024

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є вебзастосунок для організації дистанційного навчання «Campus++», який призначений для дистанційного навчання у будь-якому вищому навчальному закладі.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox, …);

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;

– функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

– системне тестування – перевіряється усе програмне забезпечення в цілому;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

– тестування «сірої скриньки» – об’єктом тестування тут є деякі особливості внутрішньої поведінки програми. Перевіряється коректність вихідних даних при заданих вхідних, застосовується для тестування окремих алгоритмів (функцій).

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально для перевірки правильності роботи виконання функціональних вимог у програмному забезпеченні та зручність користування наданим вебінтерфейсом. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- динамічне тестування на відповідність функціональним вимогам;
- тестування на пристроях з різною роздільною здатністю;
- тестування на Windows та Unix-подібних операційних системах;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування інтерфейсу користувача;
- тестування зручності використання;
- тестування локалізації.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНОГО НАВЧАННЯ

Керівництво користувача

КПІ.ІП-0217.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Сергій ТИХОНОВ

Нормоконтроль:
_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:
_____ Олексій ДЕМЧЕНКО

Київ – 2024

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Вебзастосунок для організації дистанційного навчання – це застосунок, який дозволяє облегшити організаційні питання, підвищити зручність навчання за допомогою об’єднання способів взаємодії викладачів та студентів, онлайн щоденника, видачі будь-яких завдань студенту: курсові роботи, домашні завдання та інші, та виставлення оцінок за завдання.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішного розгортання застосунку необхідне виконання наступних вимог:

- ПК з процесором Intel Core I5 2500 або кращим;
- мінімальний обсяг оперативної пам’яті 2 ГБ, оптимальний – 8 ГБ, рекомендований – 16 ГБ;
- наявність доступу до мережі Інтернет: мінімальна швидкість – 10 МБіт/с, рекомендована – 100 МБіт/с;
- наявність операційної системи, що підтримує інструмент Docker, наприклад, Windows, Linux та його дистрибутиви;
- наявність встановленної програми Docker.

2.2 Завантаження застосунку

На даний момент застосунок можна встановити власноруч. Для цього необхідно виконати наступну послідовність дій:

- завантажити програмний код серверу з платформи GitHub;
- перейти в папку з кодом серверу;
- виконати команду в термінал «make docker-start-release»;
- завантажити програмний код клієнту з платформи Github;
- перейти в папку з кодом клієнту;
- виконати команду в термінал «docker compose build && docker compose up».

2.3 Перевірка коректної роботи

По завершенню виконання попередньої послідовності дій можна перевірити успішність розгортання. Щоб зробити це, достатньо подивитися в термінал, куди вводились команди, якщо Docker не зупинив свою дію, то в термінал повинні виводитися логи з серверу та клієнту. Також можна

перевірити успішність за допомогою команди «docker ps». Дана команда виводить список всіх працюючих контейнерів у термінал. Серед них можна побачити наступні: remote-learning-api, remote-learning-frontend, remote-learning-db. Якщо список даних контейнерів відповідає дійсності, то це свідчить про те, що розгортання пройшло успішно. Далі можна впевнетись у працездатності відкривши браузер та написавши http://localhost:3000 у пошук будь-якого браузера. Якщо все пройшло успішно користувач потрапить на сторінку реєстрації.

3 ВИКОНАННЯ ПРОГРАМИ

При запуску вебзастосунку буде відображена сторінка для авторизації користувача (рисунок 3.1).

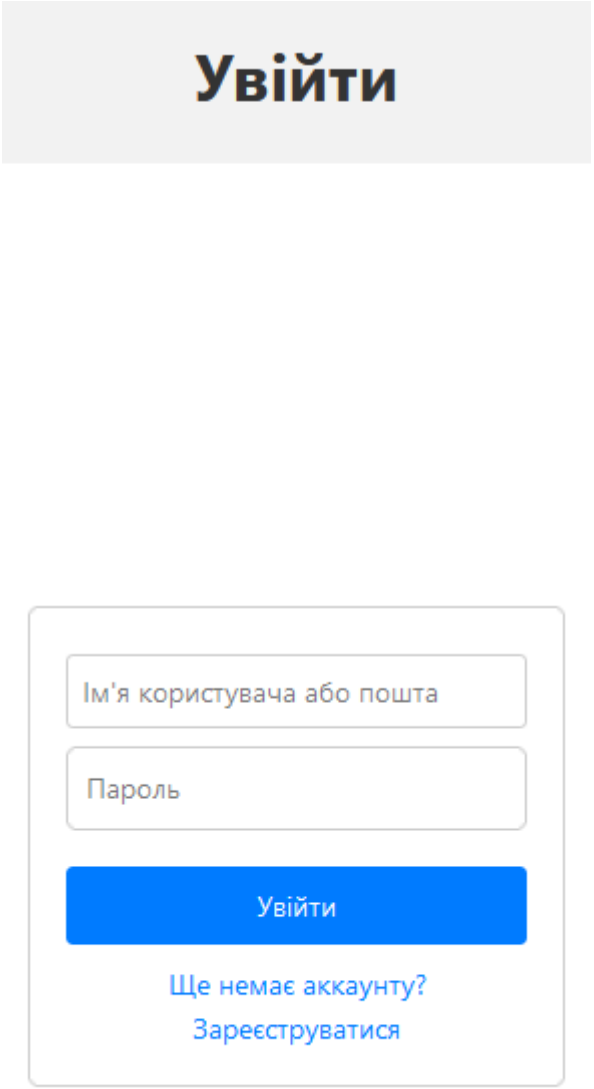


Рисунок 3.1 – Сторінка авторизації користувача

Користувач має змогу перейти на сторінку реєстрації за бажанням та зареєструватись спочатку, якщо ще не зареєстрований (рисунок 3.2).

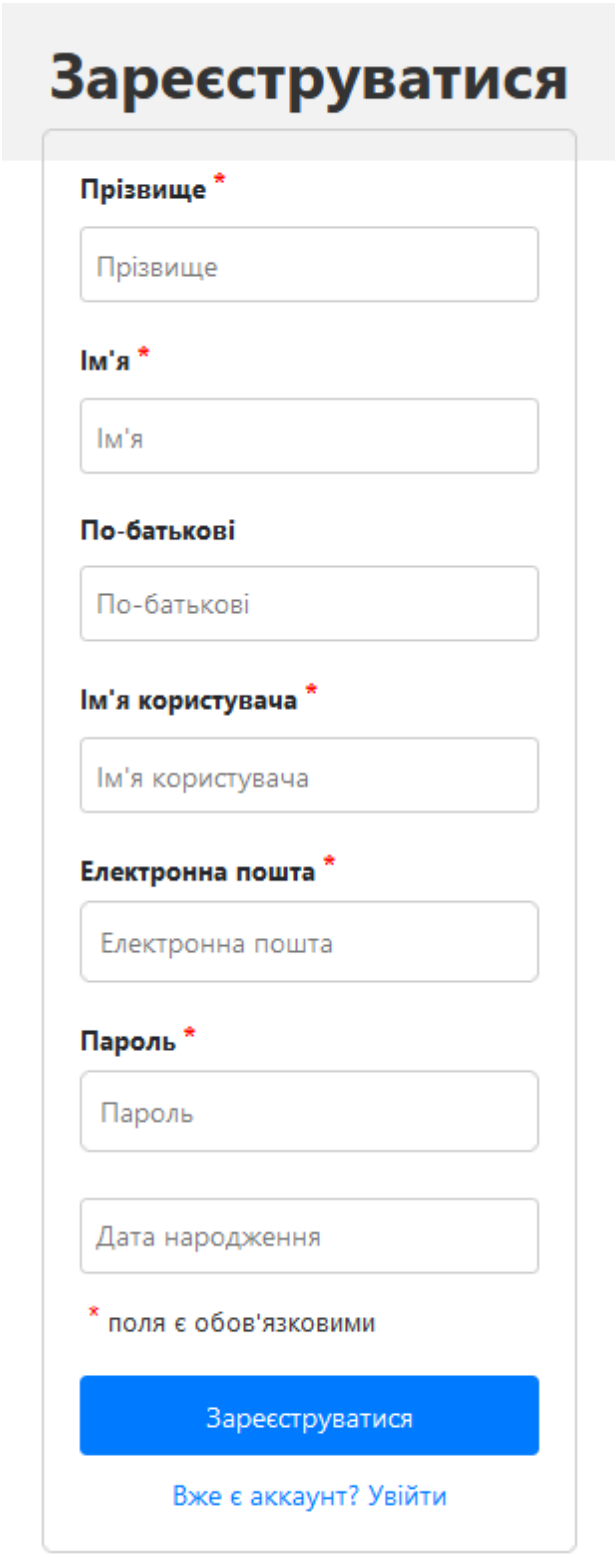


Рисунок 3.2 – Сторінка реєстрації користувача

Авторизовний адміністратор має змогу відкрити сторінку адміністратора задля створення ієрархії ВНЗ (рисунок 3.3).

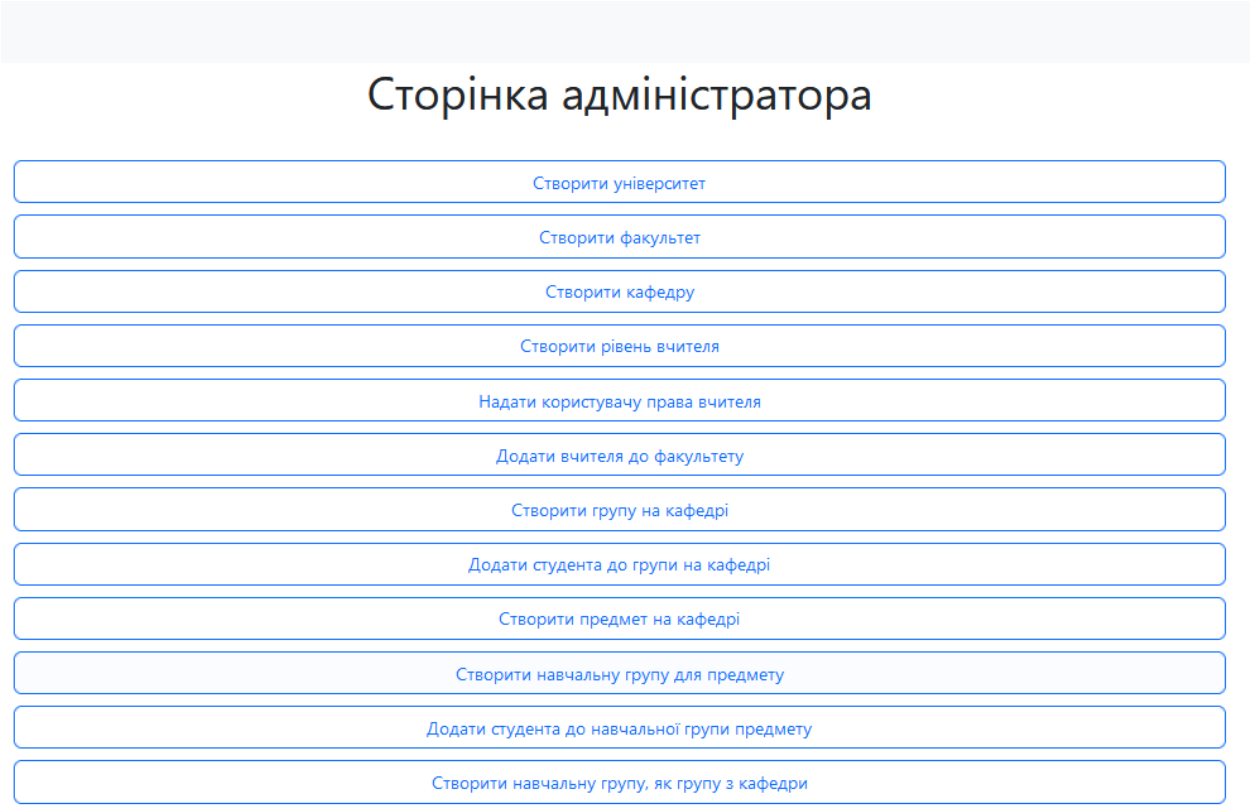


Рисунок 3.3 – Сторінка адміністратора

Адміністратор може створити університет, заповнивши просту форму (рисунок 3.4).

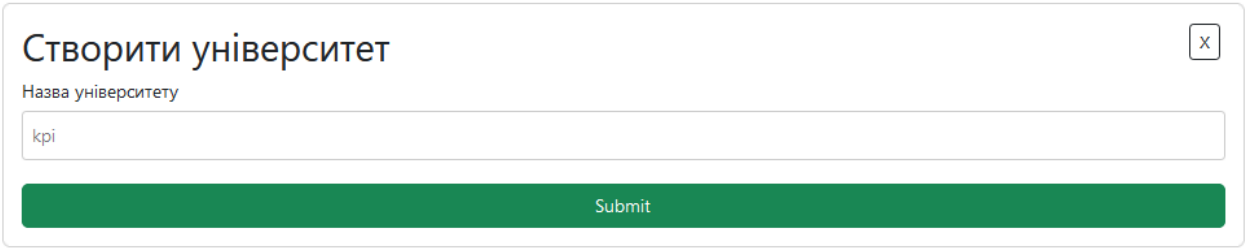


Рисунок 3.4 – Створення університету на сторінці адміністратора

Адміністратор має можливість створити факультет на вже створеному університеті (рисунок 3.5).

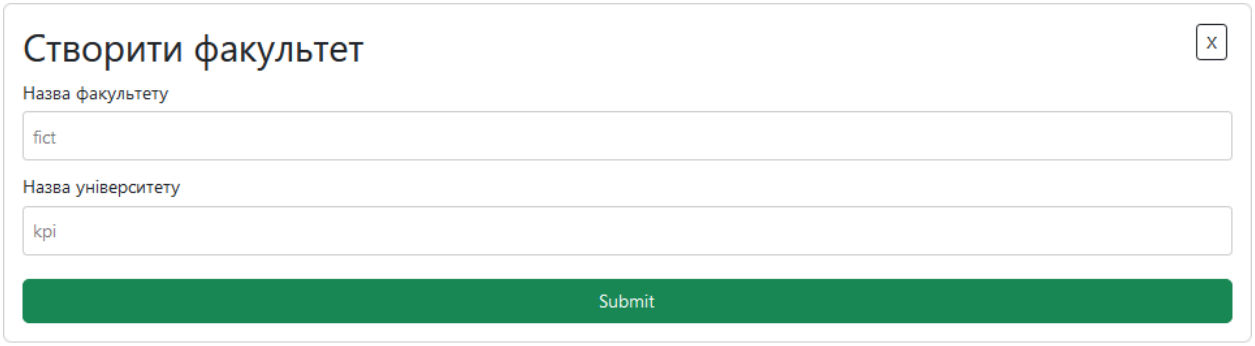


Рисунок 3.5 – Створення факультету на сторінці адміністратора

Адміністратор має можливість створити кафедру на вже існуючому факультеті (рисунок 3.6).

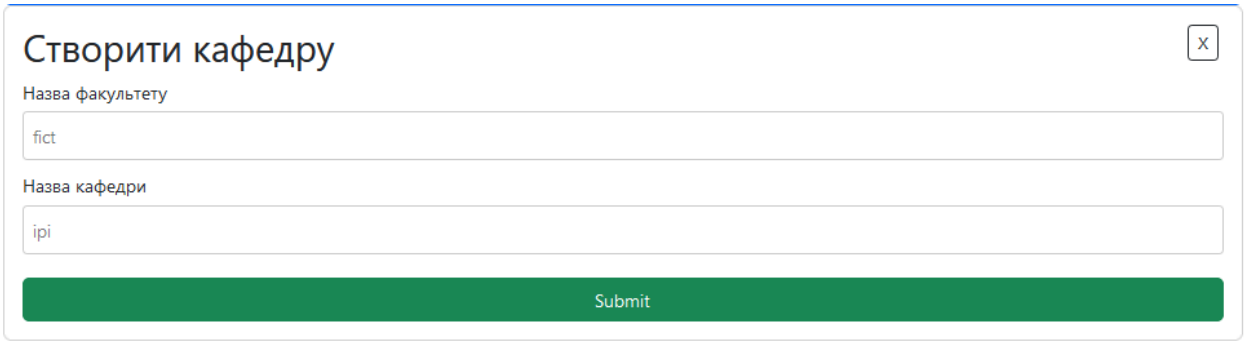


Рисунок 3.6 – Створення кафедри на сторінці адміністратора

Адміністратор має змогу створити рівень викладача (рисунок 3.7).

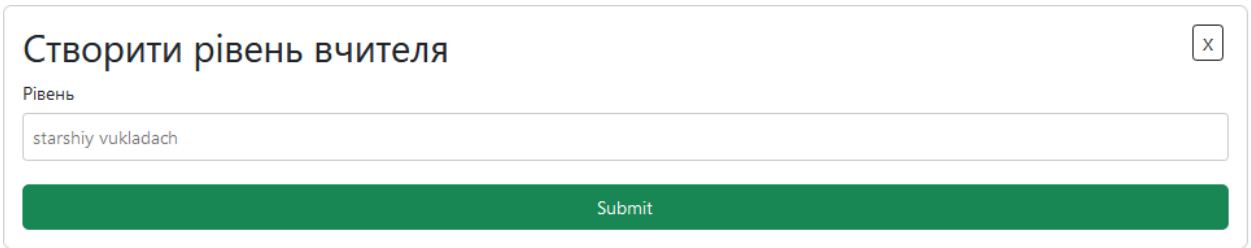


Рисунок 3.7 – Створення рівню викладача на сторінці адміністратора

Адміністратор має можливість надати користувачу права вчителя (рисунок 3.8).

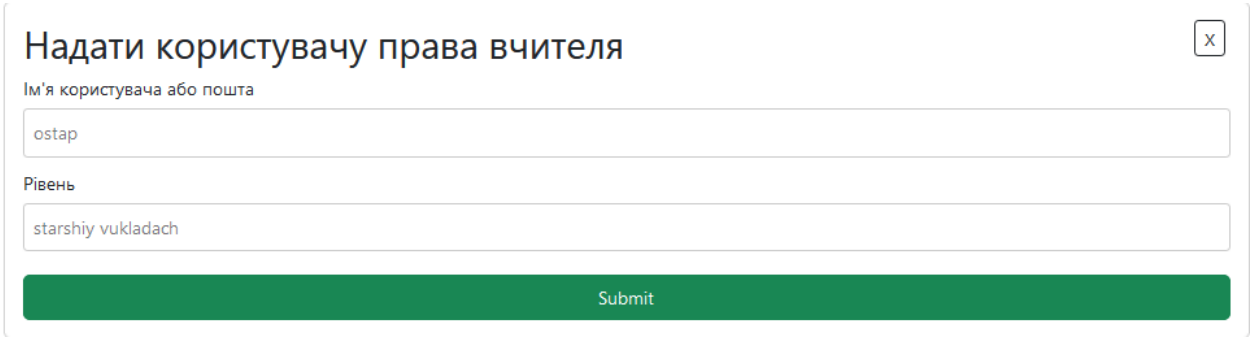


Рисунок 3.8 – Надання користувачеві прав викладача на сторінці адміністратора

Адміністратор має можливість додати викладача до вже існуючого факультету (рисунок 3.9).

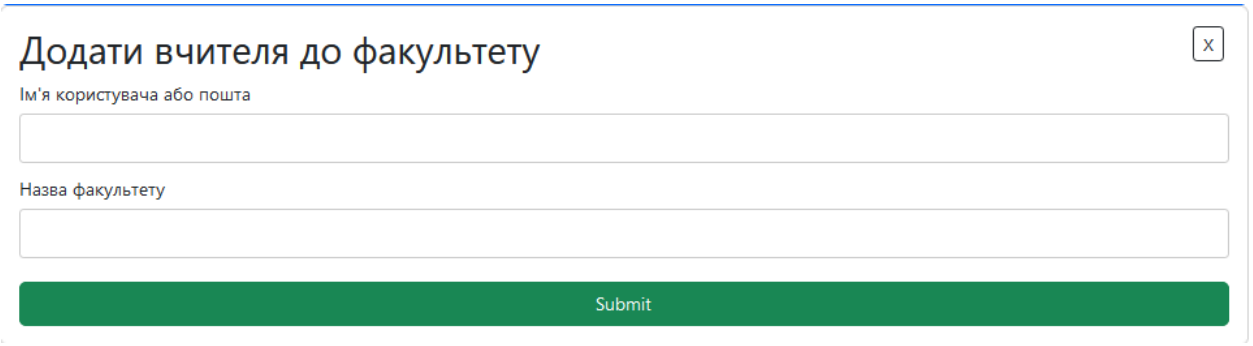


Рисунок 3.9 – Додання вчителя до факультету на сторінці адміністратора

Адміністратор має можливість створити групу на кафедрі (рисунок 3.10).

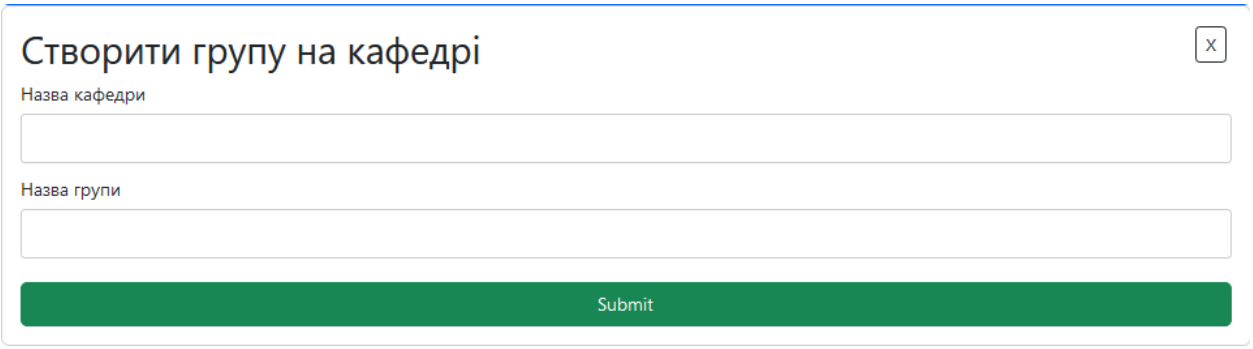


Рисунок 3.10 – Створення групи на кафедрі на сторінці адміністратора

Адміністратор має можливість додати студента до групи на кафедрі (рисунок 3.11).

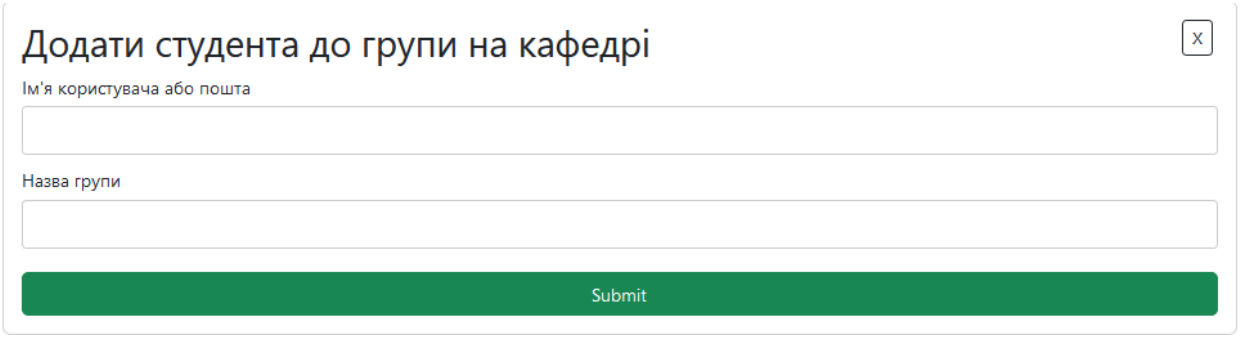


Рисунок 3.11 – Додавання студента до групи на кафедрі на сторінці адміністратора

Адміністратор має можливість створити навчальний предмет на кафедрі (рисунок 3.12).

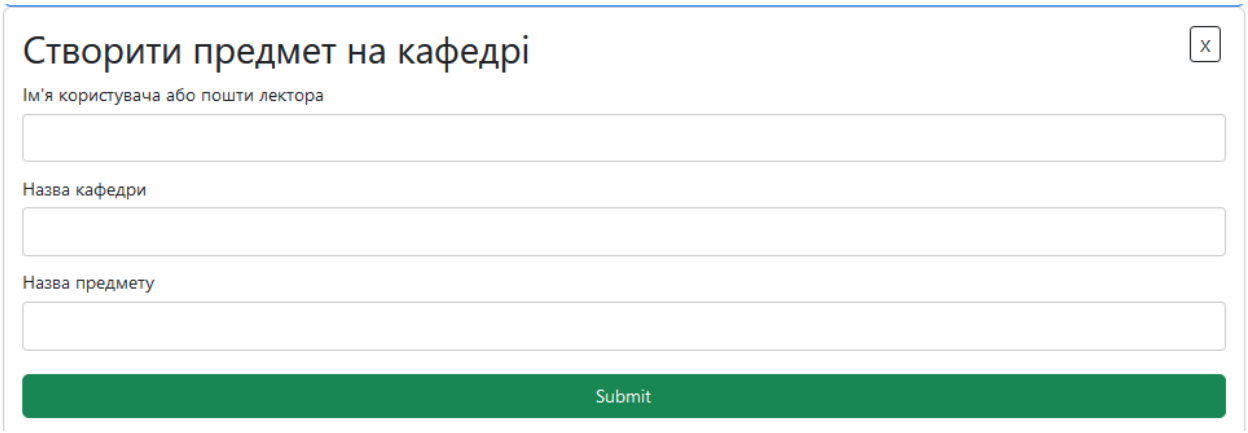


Рисунок 3.12 – Створення предмету на кафедрі на сторінці адміністратора

Адміністратор має можливість створити навчальну групу для предмету (рисунок 3.13).

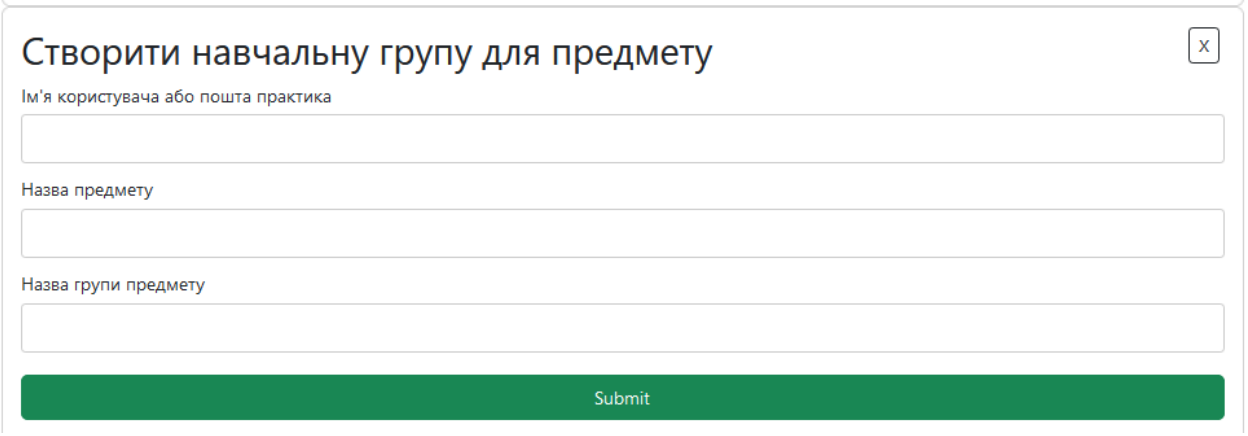


Рисунок 3.13 – Створення навчальної групи для предмету на сторінці адміністратора

Адміністратор має можливість додати студента до навчальної групи предмету (рисунок 3.14).

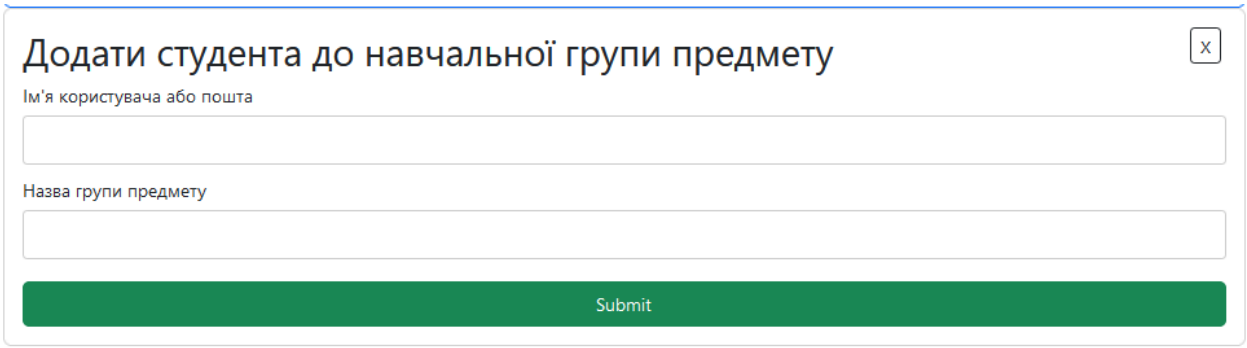


Рисунок 3.14 – Додавання студента до навчальної групи предмету на сторінці адміністратора

Адміністратор має можливість створити навчальну групу предмету, як групу на кафедрі (рисунок 3.15).

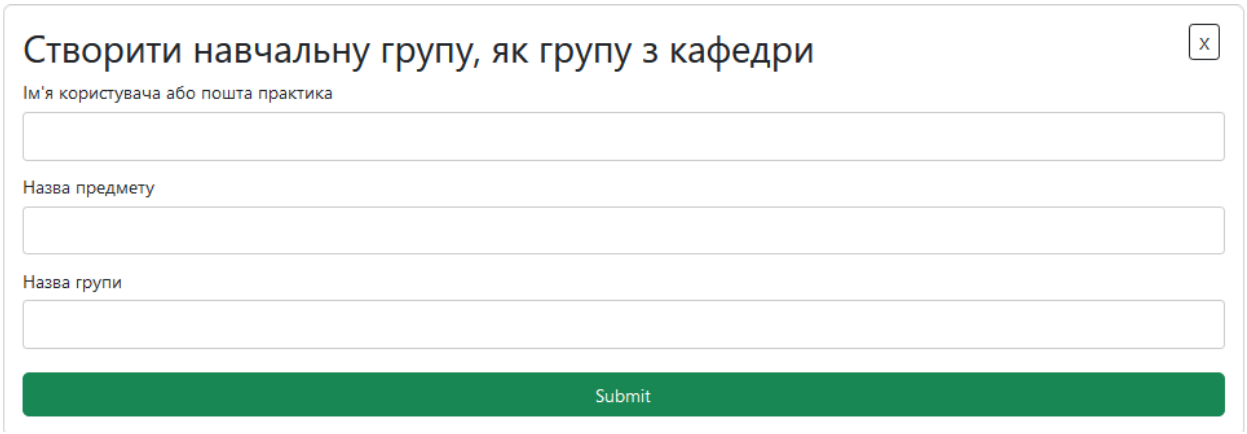


Рисунок 3.15 – Створення навчальної групи предмету, як групи на кафедрі на сторінці адміністратора

Авторизований викладач або студент має можливість відправляти повідомлення в чат групи предмету або переглядати повідомлення (рисунок 3.16).

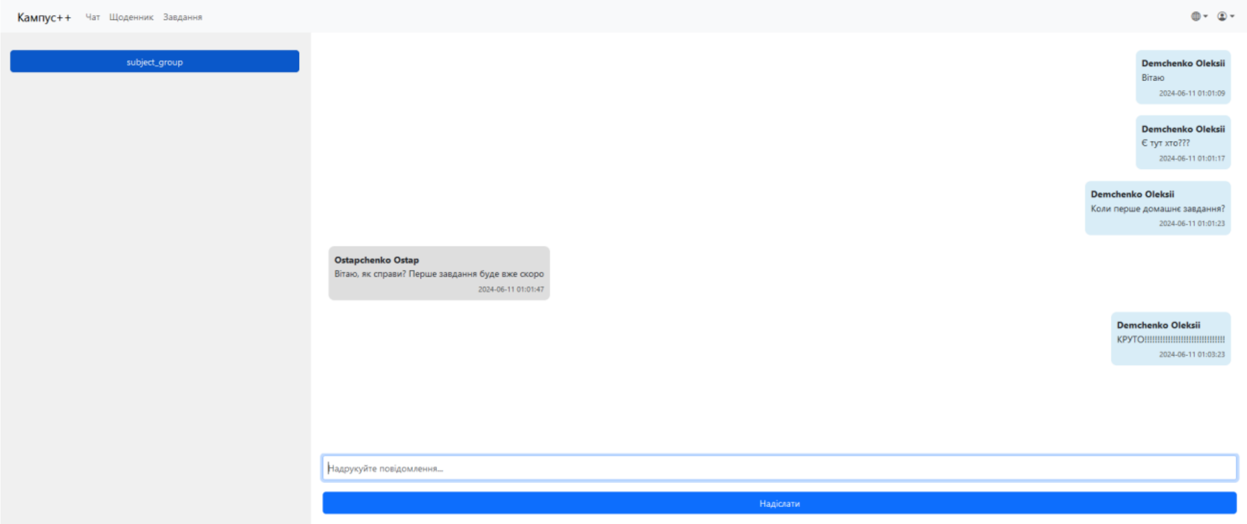


Рисунок 3.16 – Студент або викладач має можливаість надсилати та читати повідомлення в групі, якій викладає або навчається

Авторизований викладач має можливість створювати нове завдання для групи предмету, в якій викладає на сторінці завдань (рисунок 3.17).

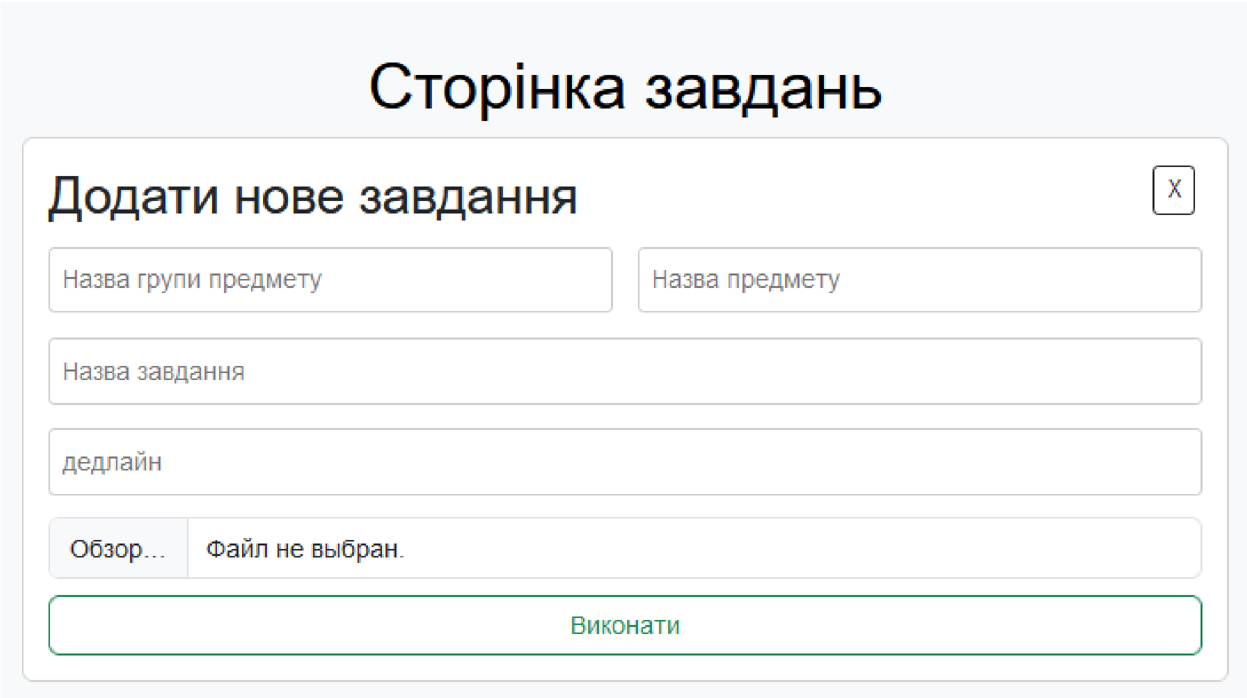


Рисунок 3.17 – Форма для створення завдання для навчальної групи предмету, в якій викладач проводить заняття з практики

Авторизований викладач має можливість переглядати створені завдання та оцінювати рішення, завантажувати файли на локальний пристрій (рисунок 3.18).

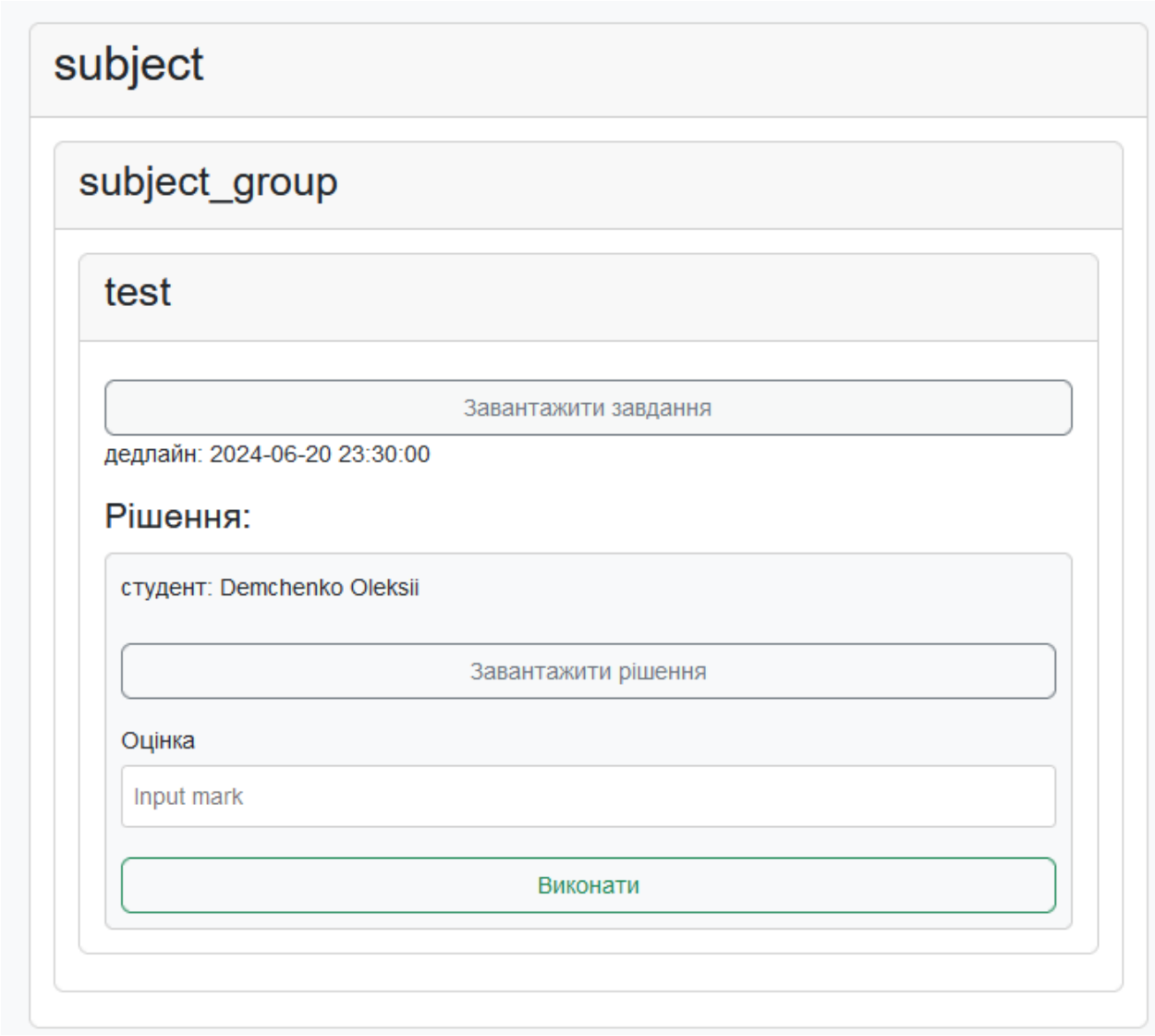


Рисунок 3.18 – Перегляд створених завдань викладачем та прикріплених до них рішень

Авторизований студент має можливість преглядати завдання, дедлайни, завантажувати рішення та переглядати оцінки за виконані решення на сторінці завдань (рисунок 3.19).

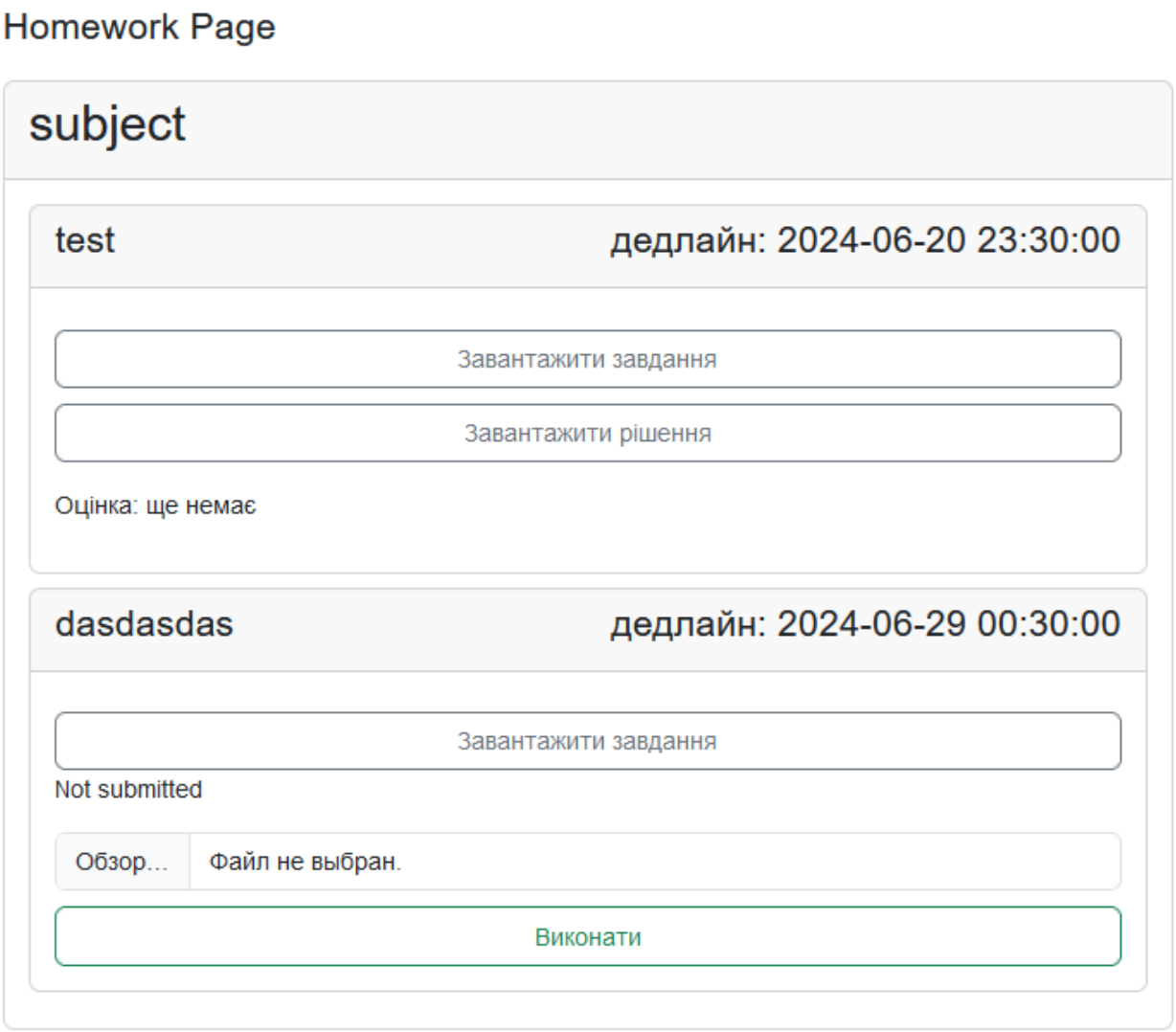


Рисунок 3.19 – Авторизований студент переглядає надіслані для його групи завдання, власні рішення

Студент має можливість на сторінці «щоденник» переглядати оцінки за кожен предмет та суму балів за кожен предмет окремо (рисунок 3.20).

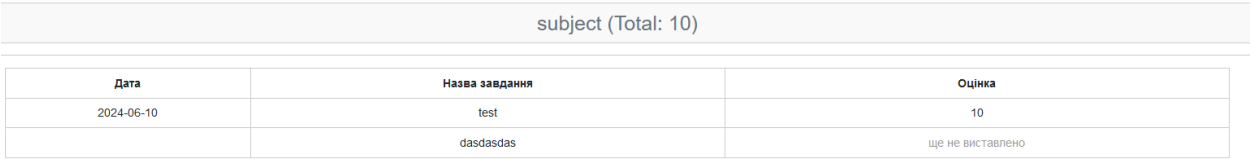


Рисунок 3.20 – Авторизований студент переглядає сторінку щоденник, на якій бачить оцінки за власні рішення

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНОГО НАВЧАННЯ

Графічний матеріал

КПІ.ІП-0217.045440.06.99

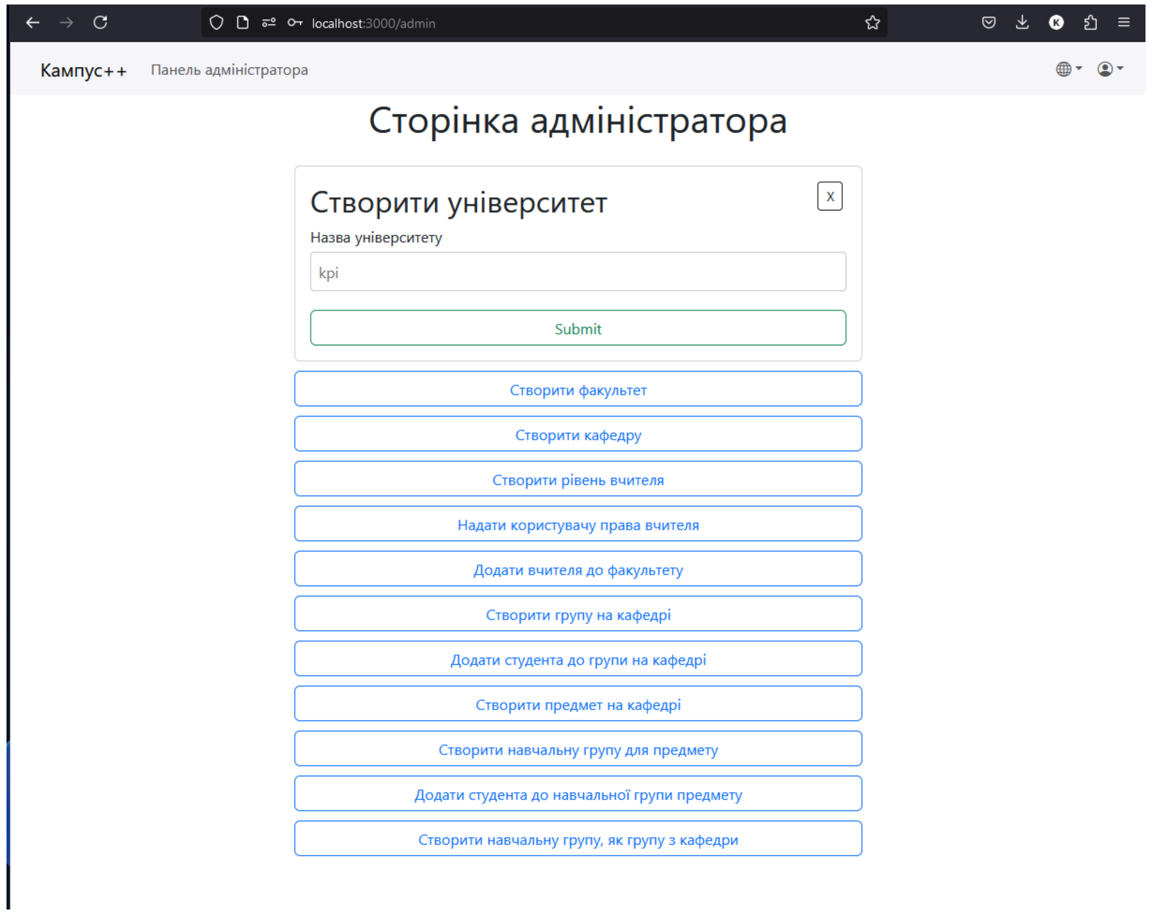
“ПОГОДЖЕНО”
Керівник проєкту:
_____ Сергій ТИХОНОВ

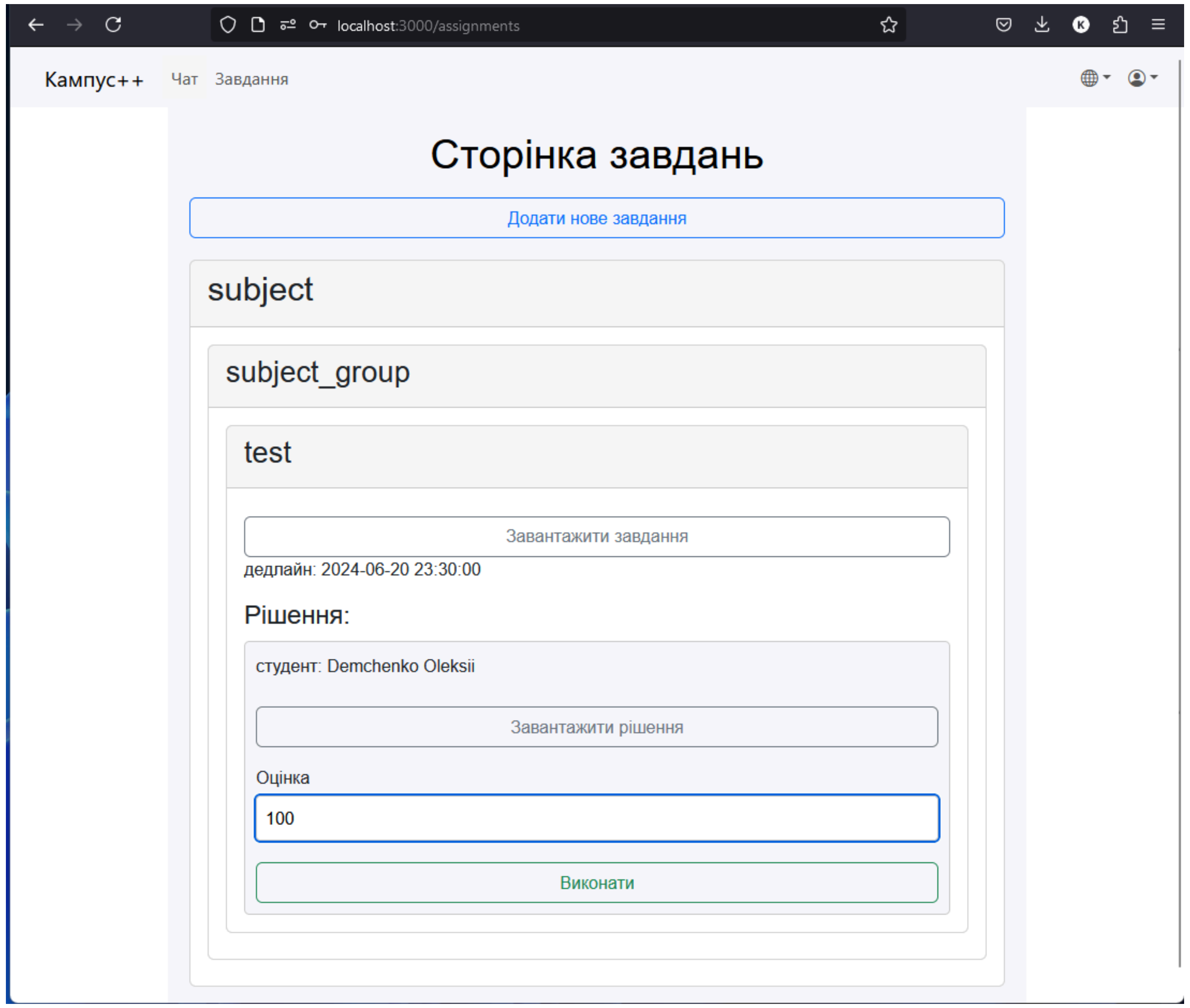
Нормоконтроль:
_____ Тетяна ШУЛЬКЕВИЧ

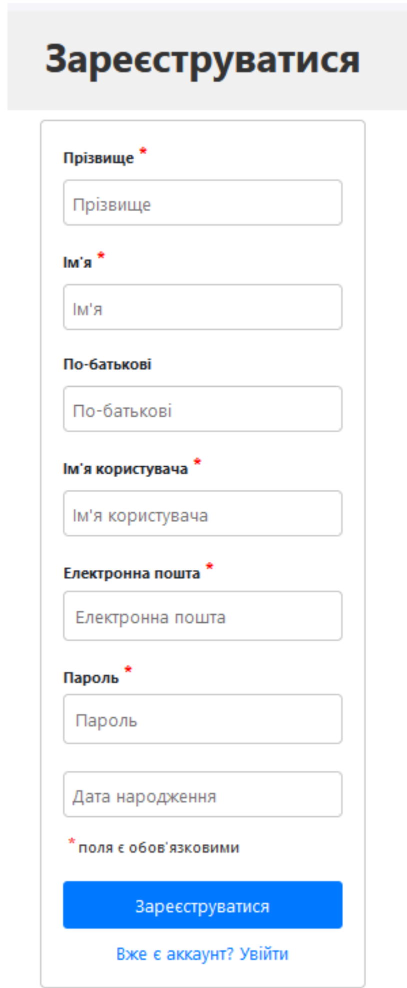
Виконавець:
_____ Олексій ДЕМЧЕНКО

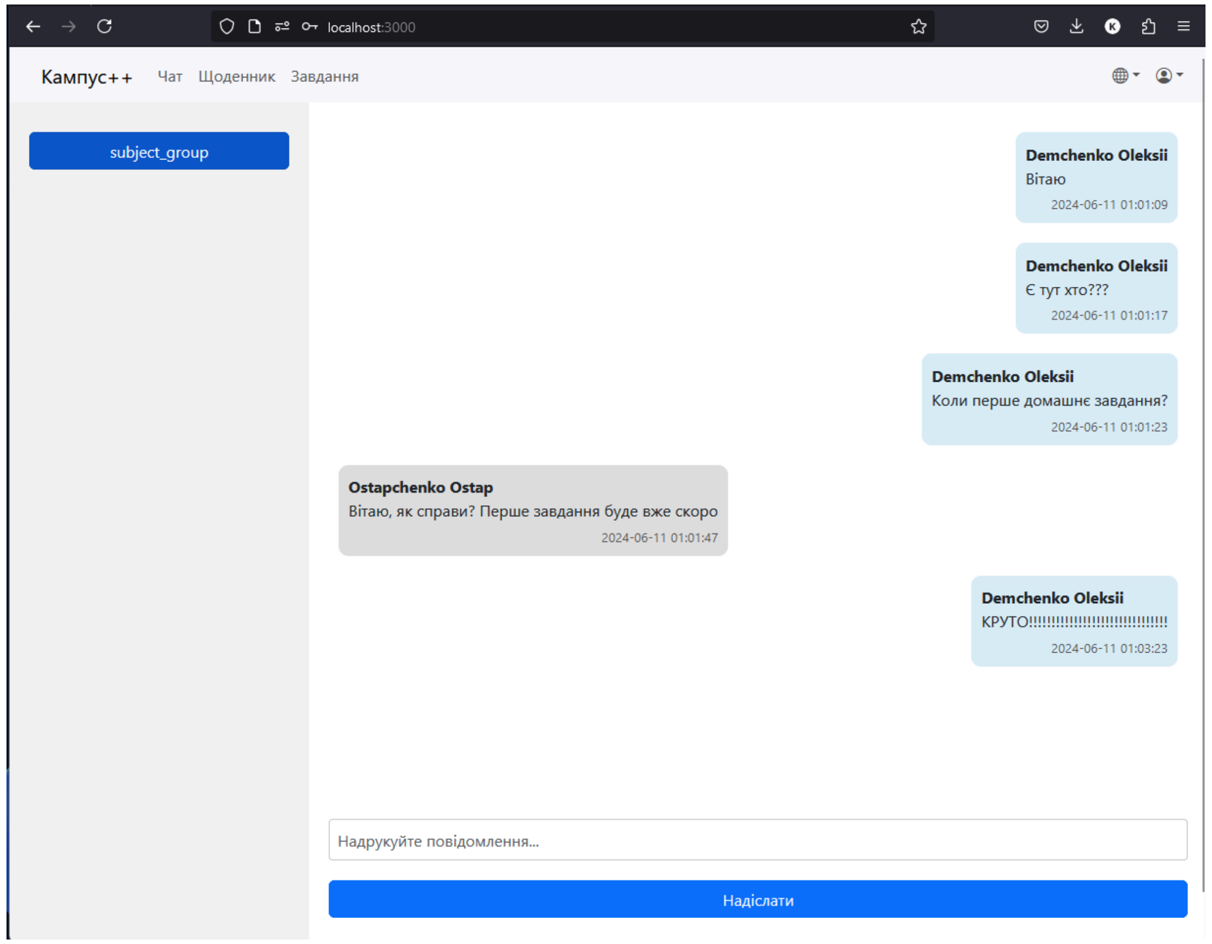
Київ – 2024

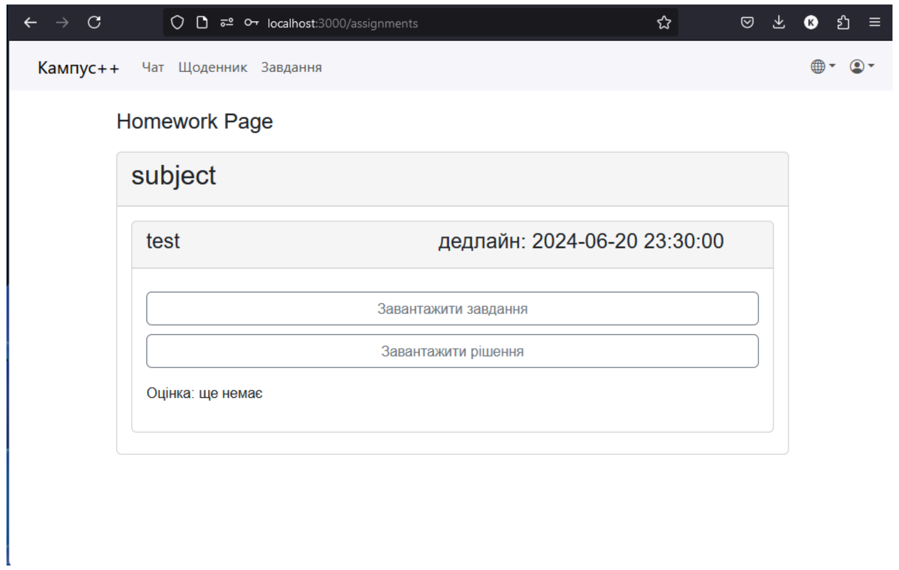
КПІ.ІП-0217.045440.06.99.СКДК

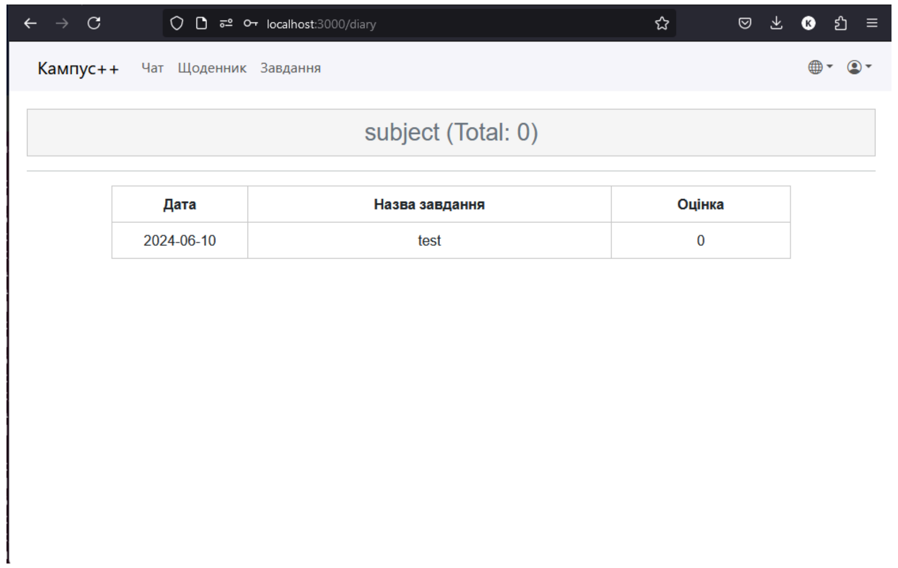


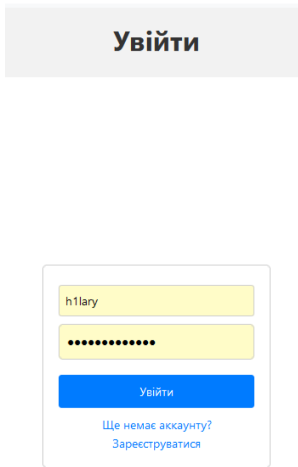






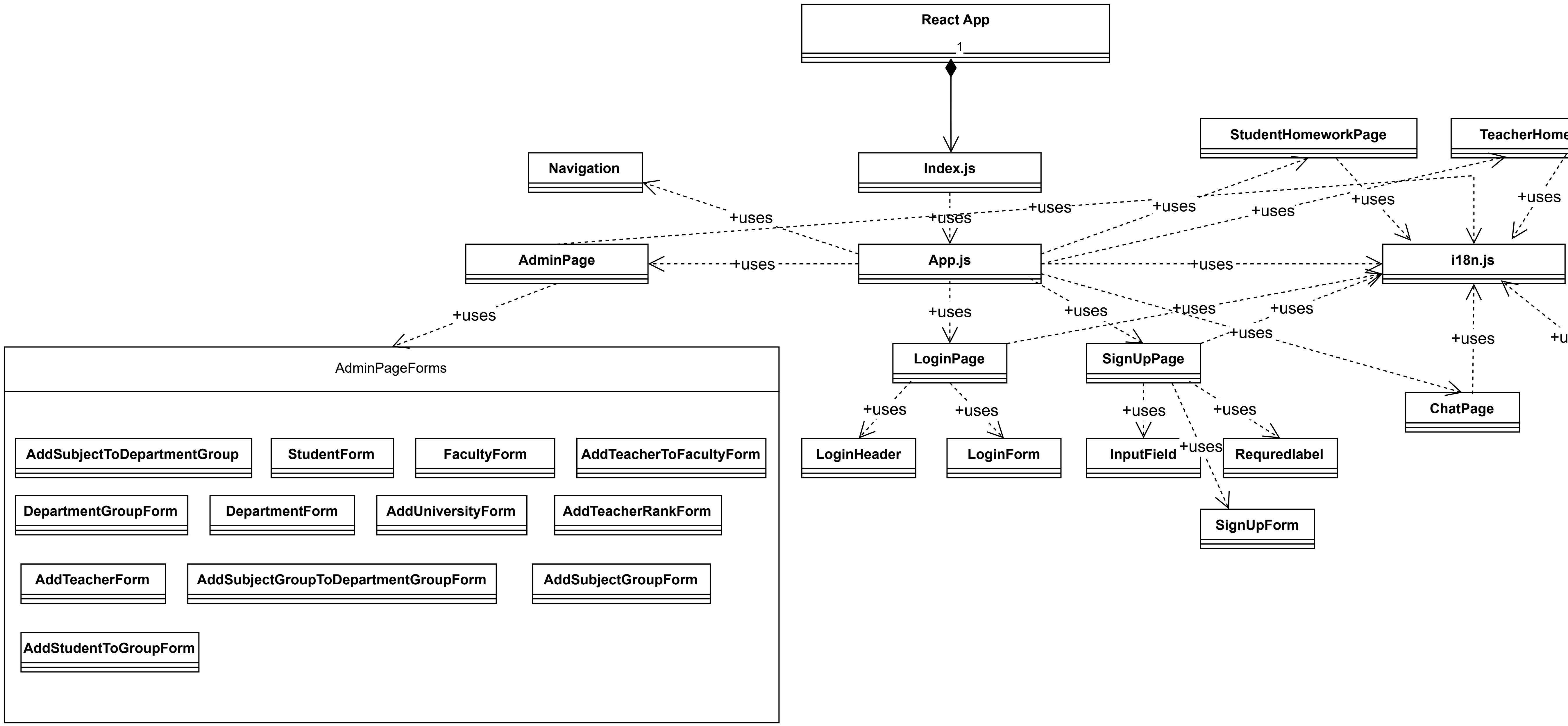






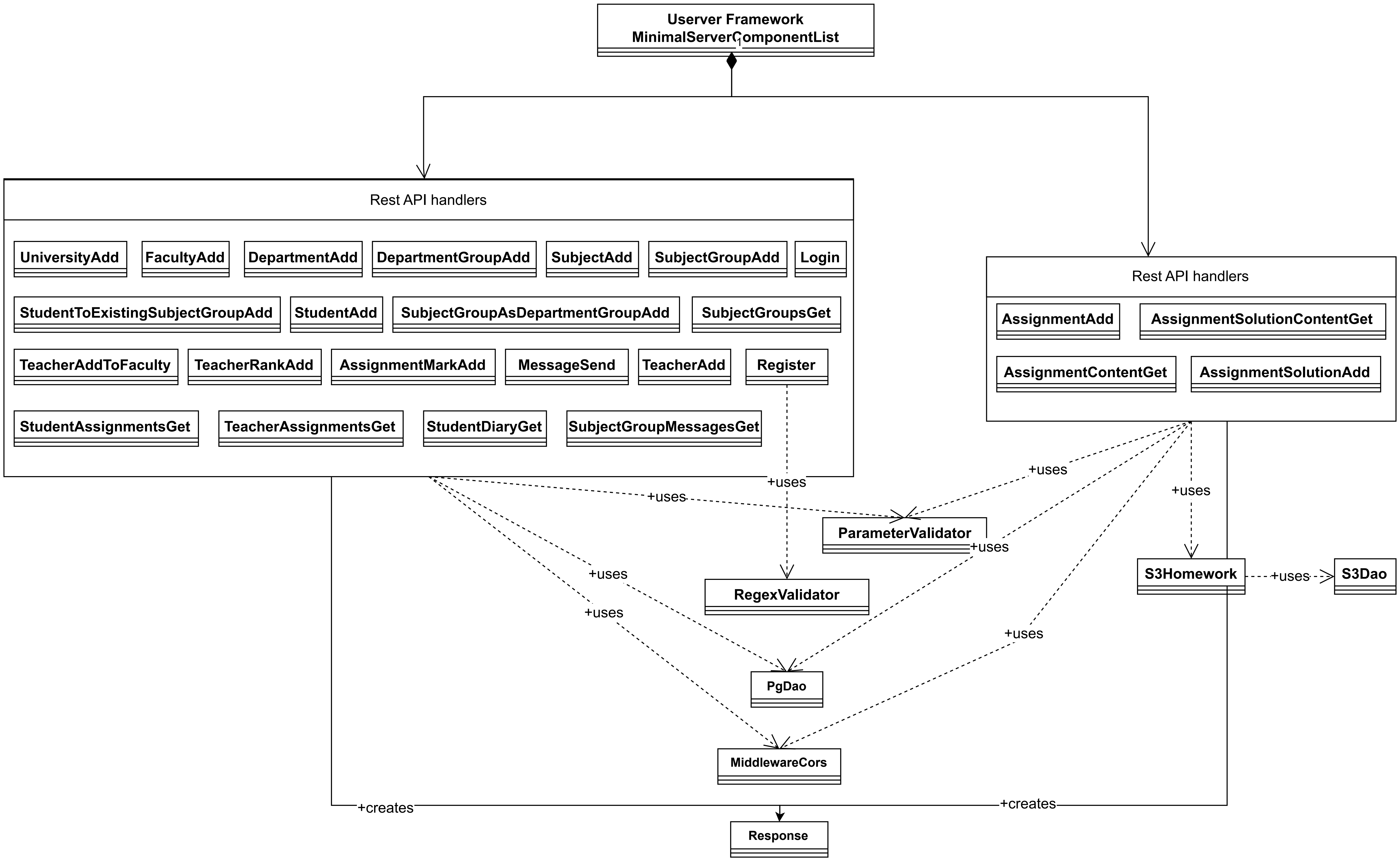
					КПІ.ІП-0217.045440.06.99
Зм.	Арк.	№ документа	Підпис	Дата	
Розробив		Демченко О.С.			Креслення вигляду екранних форм
Перевірив		Тихонов С.В.			Літера / Маса / Масштаб
Т. контр.					Аркуш 1 / Аркушів 5
Н. контр.		Шулькевич Т.В.			Вебзастосунок для організації дистанційного навчання
Затвердив		Жаріков Е.В.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-02

КПІ.ІП-0217.045440.06.99.СКК



					КПІ.ІП-0217.045440.06.99.ССК			
					Схема структурна класів програмного забезпечення	Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Демченко О.С.						
Перевірив		Тихонов С.В.						
Т. контр.						Аркуш 2	Аркушів 5	
					Вебзастосунок для організації дистанційного навчання	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-02		
Н.контр.		Шулькевич Т.В.						
Затвердив		Жаріков Е.В.						

КПІ.ІП-0217.045440.06.99.СКК



Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Демченко О.С.		
Перевірив		Тихонов С.В.		
Т. контр.				
Н. контр.		Шулькевич Т.В.		
Затвердив		Жаріков Е.В.		

КПІ.ІП-0217.045440.06.99.СКК

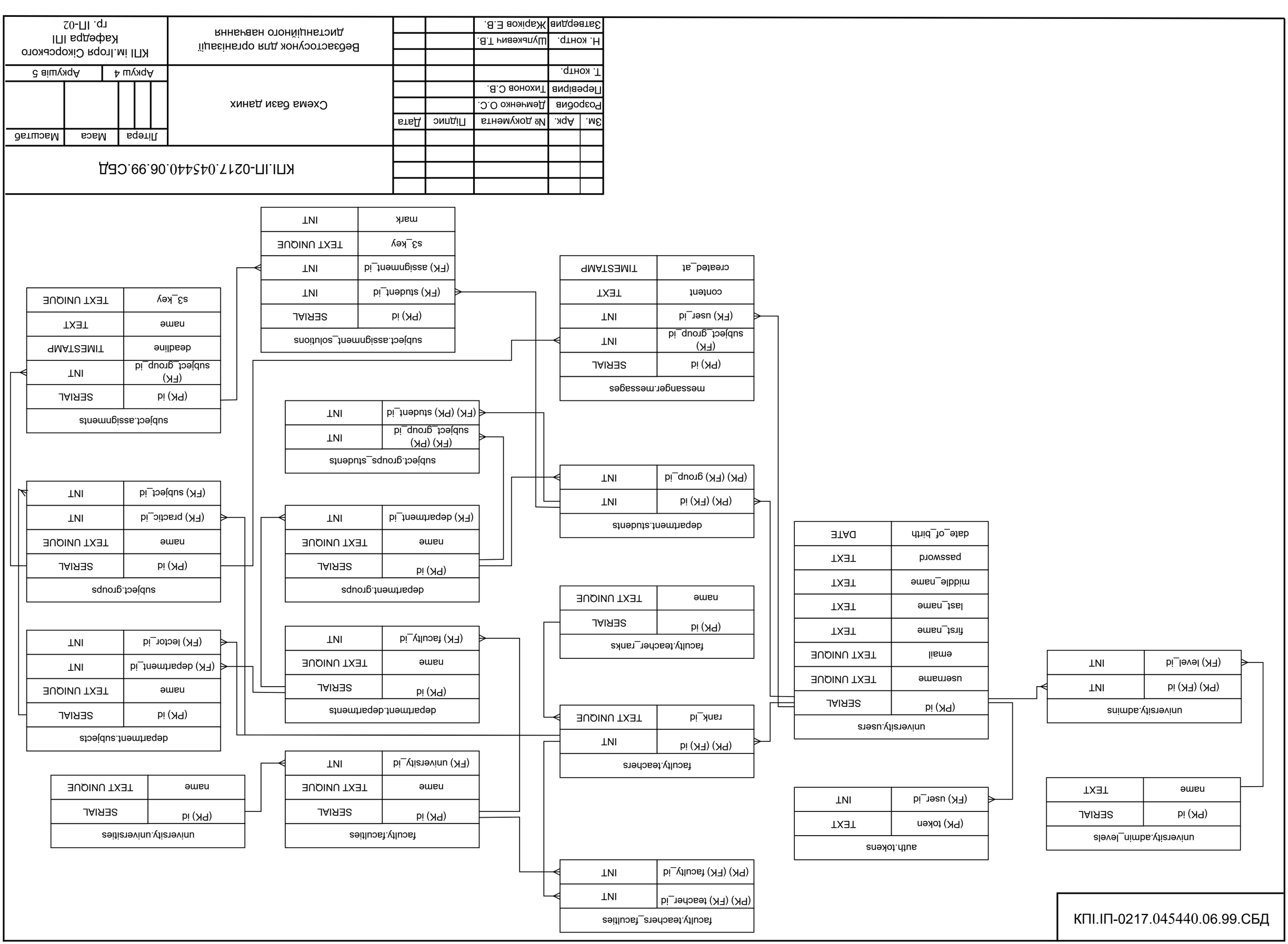
Схема структурна класів програмного забезпечення

Літера	Маса	Масштаб

Аркуш 3 | Аркушів 5

Вебзастосунок для організації дистанційного навчання

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІП-02



КПІ.ІП-0217.045440.06.99.СБД

Літера	Маса	Масштаб

Арк.	Аркушів
4	5

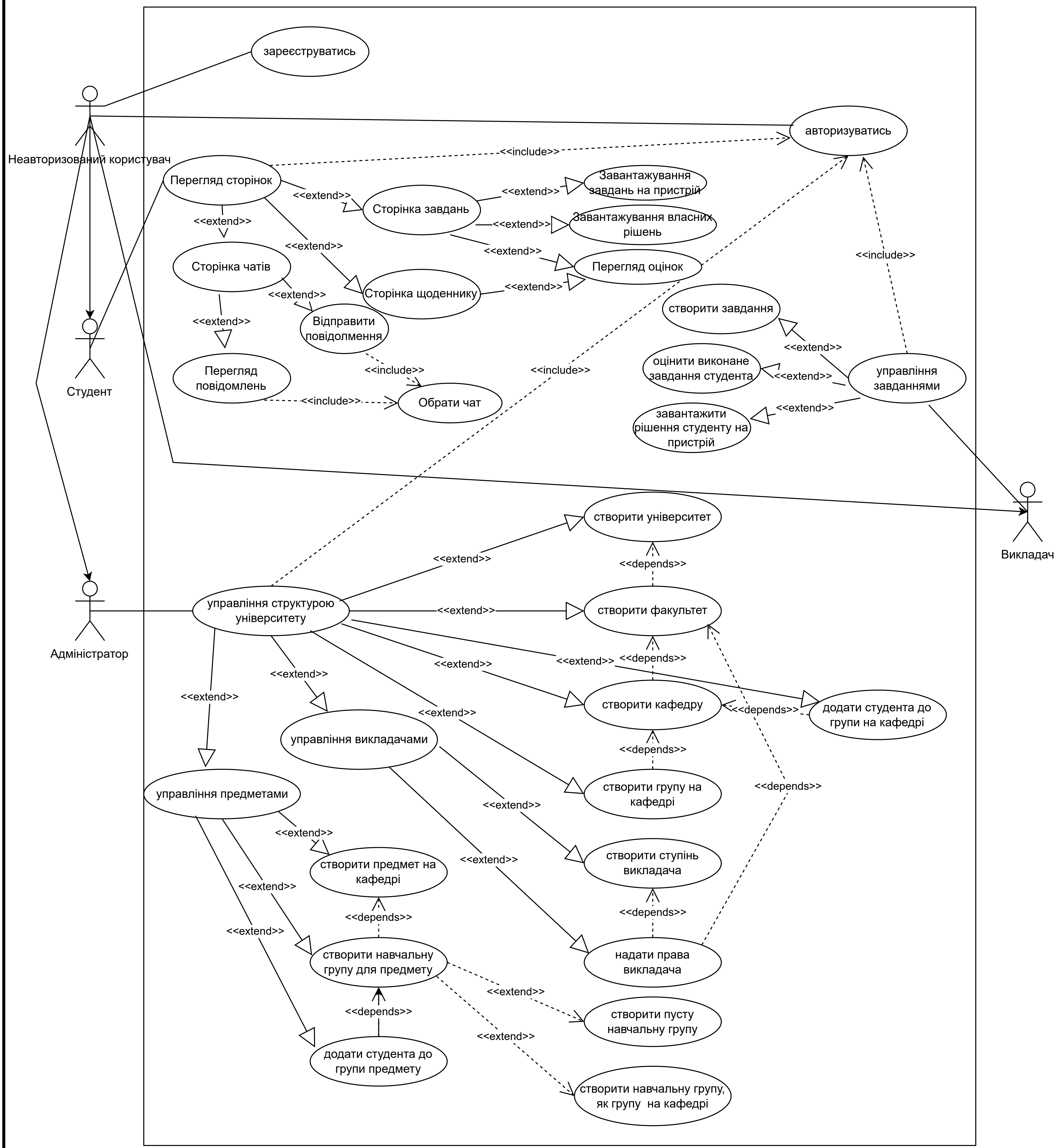
Схема бази даних

Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Демченко О.С.		
Перевірив		Тихонов С.В.		
Т. контр.				
Н. контр.		Шулькевич Т.В.		
Затвердив		Жарікова Е.В.		

Веб-застосунок для організації дистанційного навчання

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІП-02

КПІ.ІП-0217.045440.06.99.СВВ



					КПІ.ІП-0217.045440.06.99.СВВ			
					Схема варіантів використання	Лит.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підп.	Дата				
Розробив		Демченко О.С.						
Перевірив		Тихонов С.В.						
Т. контр.						Аркуш 5		Аркушів 5
					Вебзастосунок для організації дистанційного навчання	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-02		
Н. контр.		Шулькевич Т.В.						
Затвердив		Жаріков Е.В.						