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)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

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

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмне забезпечення для підготовки та роботи з розкладом занять

Виконав студент IV курсу, групи ІТ-92
(шифр групи)
Яцевський Олександр Ігорович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Фіногенов О.Д. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст. викл.,Вітковська І.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент, к.т.н., доц., Філіппова М.В. _____
(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 –2023

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)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Яцевському Олександру Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Програмне забезпечення для підготовки та роботи з розкладом занять

керівник проєкту Фіногенов Олексій Дмитрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31» травня 2023 р. № 2101-с

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.
4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна компонентів програмного забезпечення

3) Схема бази даних

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «10» березня 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання задачі	17.04.2023	
3	Постановка та формалізація задачі	19.04.2023	
4	Розробка інформаційного забезпечення	20.04.2023	
5	Алгоритмізація задачі	24.04.2023	
6	Обґрунтування вибору використаних технічних засобів	29.04.2023	
7	Розробка програмного забезпечення	05.05.2023	
8	Налагодження програми	13.05.2023	
9	Виконання графічних документів	20.05.2023	
10	Оформлення пояснювальної записки	28.05.2023	
11	Подання ДП на попередній захист	02.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент _____ Олександр ЯЦЕВСЬКИЙ
(підпис) (ініціали, прізвище)

Керівник _____ Олександр ФІНОГЕНОВ
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 31 таблицю, 10 рисунків та 8 джерел – загалом 58 сторінки.

Дипломний проєкт присвячений розробці програмного забезпечення для підготовки та роботи з розкладом занять. Метою створення системи є розробка системи для підготовки та роботи з розкладом занять для нової системи розкладу занять НТУУ КПІ ім. Ігоря Сікорського.

У розділі аналізу вимог до програмного забезпечення розглянуто наявне програмне забезпечення для заповнення та відображення розкладу занять, проаналізовано вимоги до такого ПЗ, виконано постановку задачу для цього проєкту.

Розділ моделювання та конструювання присвячений дослідженню вхідних даних та моделюванню сутностей даних, розробці архітектури компонентів ПЗ, аналізу наявних алгоритмів для вирішення задач та розробці бази даних.

У розділі аналізу якості та тестування розглянуто процеси оцінки якості , використані для розробки ПЗ, розроблено процес мануального тестування ПЗ.

Розділ впровадження та супровід присвячений інструкціям по розгортанню та підтримці програмного забезпечення.

Програмне забезпечення впроваджено в НТУУ КПІ ім. Ігоря Сікорського.

КЛЮЧОВІ СЛОВА: РОЗКЛАД ЗАНЯТЬ, БАЗА ДАНИХ, ВЕБ СЕРВІСИ, ПЗ ДЛЯ УНІВЕРСИТЕТУ

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 31 tables, 10 figures and 8 sources – in total 58 pages.

The diploma project is devoted to the development of software for preparing and working with the schedule of classes. The purpose of creating the system is to develop a system for preparing and working with the class schedule for the new class schedule system of National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

In the software requirements analysis section, the available software for filling and displaying the class schedule was considered, the requirements for such software were analyzed, and the task set for this project was completed.

The modeling and design section is devoted to the study of input data and modeling of data entities, the development of the architecture of software components, the analysis of existing algorithms for solving problems, and the development of a database.

In the quality analysis and testing section, quality assessment processes used for software development are considered, and the process of manual software testing is developed.

The Implementation and Support section provides instructions for deploying and maintaining the software.

The software is implemented in NTUU KPI named after Igor Sikorsky.

KEY WORDS: SCHEDULE, DATABASE, WEB SERVICES, UNIVERSITY SOFTWARE

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4	КПІ.ІТ-9230.045440.00.90	Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-9230.045440.01.91	Технічне завдання	17	
3	А4	КПІ.ІТ-9230.045440.02.81	Пояснювальна записка	60	
4	А4	КПІ.ІТ-9230.045440.03.12	Текст програми	47	
5	А4	КПІ.ІТ-9230.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-9230.045440.05.34	Керівництво користувача	8	
7	А3	КПІ. ІТ-9230.045440.06.99	Схема структурна варіантів використань	1	
7	А3	КПІ. ІТ-9230.045440.06.99	Схема структурна компонентів програмного	1	
8	А3	КПІ.ІТ-9230.045440.06.99	Схема бази даних	1	

					КПІ.ІТ-9230.045440.00.90		
Змін.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Яцевський О.І.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Фіногенов О.Д.					
Н.контр.		Вітковська І.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-92	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023
р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДГОТОВКИ ТА РОБОТИ З РОЗКЛАДОМ ЗАНЯТЬ

Технічне завдання

КПІ.ІТ-9230.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____Олександр ЯЦЕВСЬКИЙ

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для підготовки та роботи з розкладом занять.

Галузь застосування:

Наведене технічне завдання поширюється на розробку “програмне забезпечення для підготовки та роботи з розкладом занять” КПІ.ІТ-9230.045440.02.81, котра використовується для заповнення та редагування даних розладу занять, надання графічного інтерфейсу для управління розкладами занять, інтеграції з системами ЄІС НТУУ КПІ ім. Ігоря Сікорського та призначена для управління даними розкладу занять та синхронізації бази даних розкладів занять ЄІС.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки “Програмного забезпечення для підготовки та роботи з розкладом занять” 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 інтеграції нової системи розкладу з Єдиним Інформаційним Середовище та виведення з експлуатації попередньої системи розкладу занять.

Метою розробки є:

– інтеграція нової системи розкладу занять;
– синхронізація даних з єдиним інформаційним середовищем;
– надання інтерфейсу для управління даними розкладу.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Синхронізація бази даних

Основними етапами є:

– розробка бази даних для збереження даних розкладу занять;

– механізм автоматичної синхронізації даних з сервісами ЄІС;

– механізм обробки синхронізації БД в умовах нестабільної роботи датацентру.

4.1.2 Користувацький інтерфейс

– при відкритті головної сторінки інтерфейсу адміністратора, користувача зустрічає форма авторизації (рис.4.1);

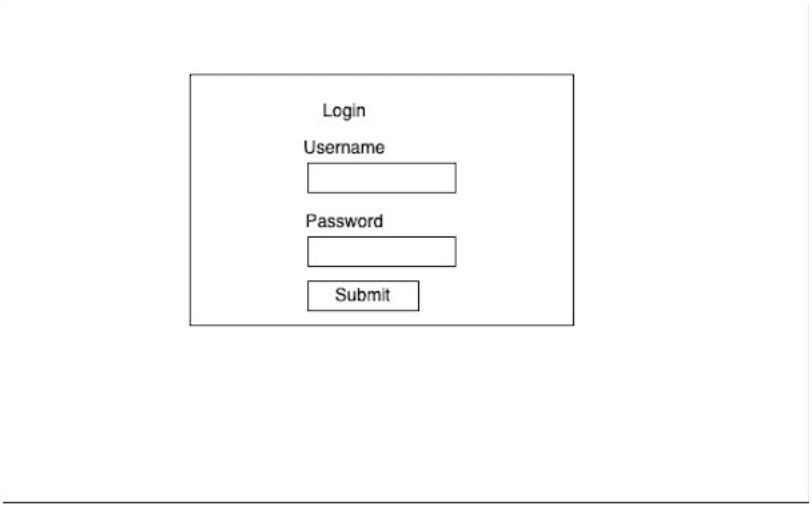


Рисунок 4.1 – Форма авторизації

– після успішної авторизації перед користувачем відкривається головна сторінка додатку (рис 4.2);

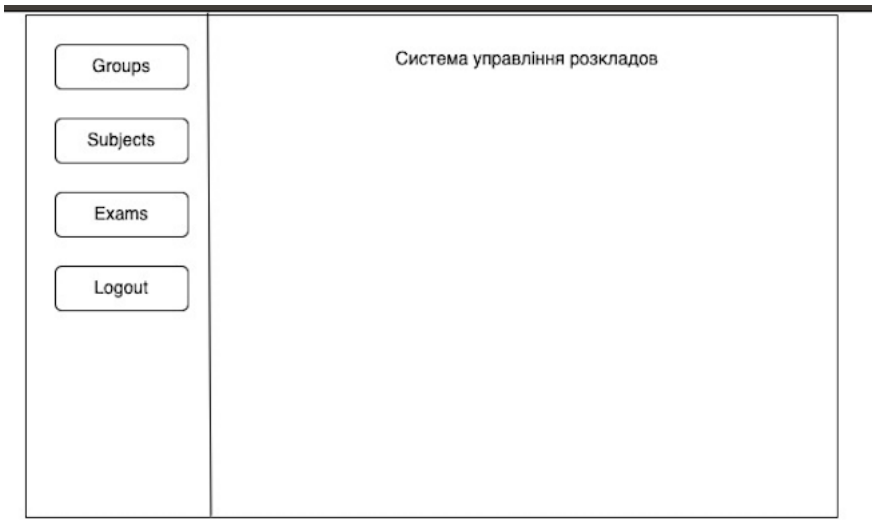


Рисунок 4.2 – Головна сторінка додатку

– На боковій панелі відображається три кнопки, які ведуть на відповідні сторінки: “Groups” (сторінка “Групи”), “Subjects” (сторінка “Розклад занять”) та “Exams” (сторінка “Розклад Екзаменів”);

– для перегляду списку груп користувач може натиснути кнопку “Groups” у боковій панелі, перед ним відкриється список груп, наявних в системі. Для кожної групи наявна кнопка “Edit” (Редагувати) та “Remove” (Видалити)(рис 4.3);

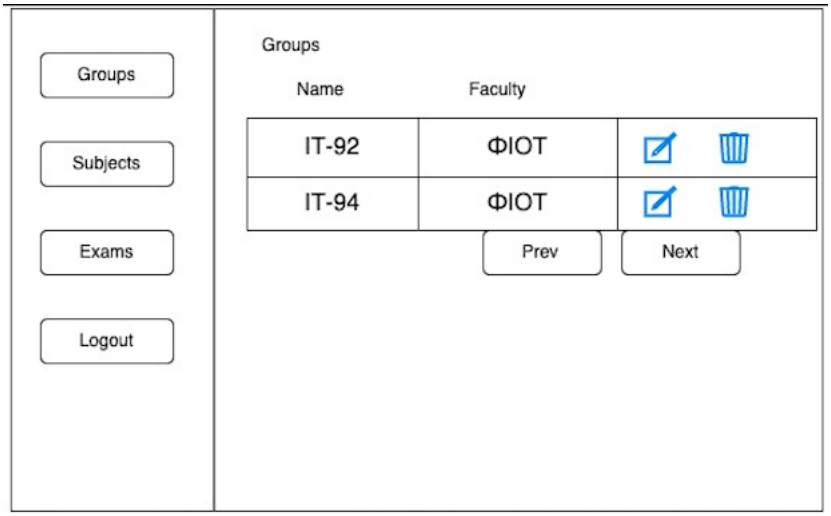


Рисунок 4.3 – Сторінка “Groups”

– на сторінці “Groups”, перед користувачем відкривається список груп, наявних в системі, та форма для заповнення даних нової групи. Після того, як користувач заповнив дані групи, він може натиснути кнопку “Create Group”, після чого група буде додана до бази даних. Крім того, для кожної групи на сторінці “Groups” наявна кнопка “Edit” (Редагувати), яку можна використовувати для редагування даних групи, та кнопка “Remove” (Видалити), яку можна використовувати для видалення групи з бази даних (рис 4.4);

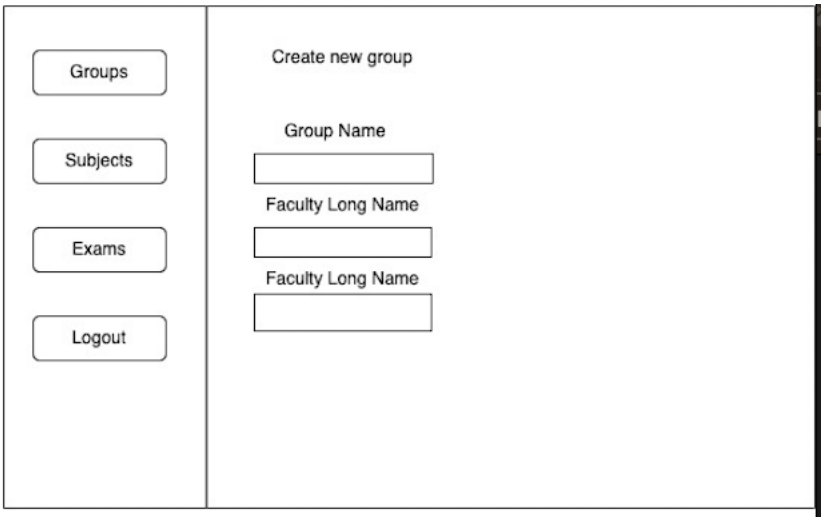


Рисунок 4.4 – Сторінка “Create Group”

– для перегляду розкладу занять користувач може натиснути кнопку “Subjects” у боковій панелі, перед ним відкриється розклад занять для всіх груп в системі. Для кожного предмету на сторінці “Subjects” наявна кнопка “Edit” (Редагувати), яку можна використовувати для редагування даних предмету, та кнопка “Remove” (Видалити), яку можна використовувати для видалення предмету з бази даних. (рис 4.5);

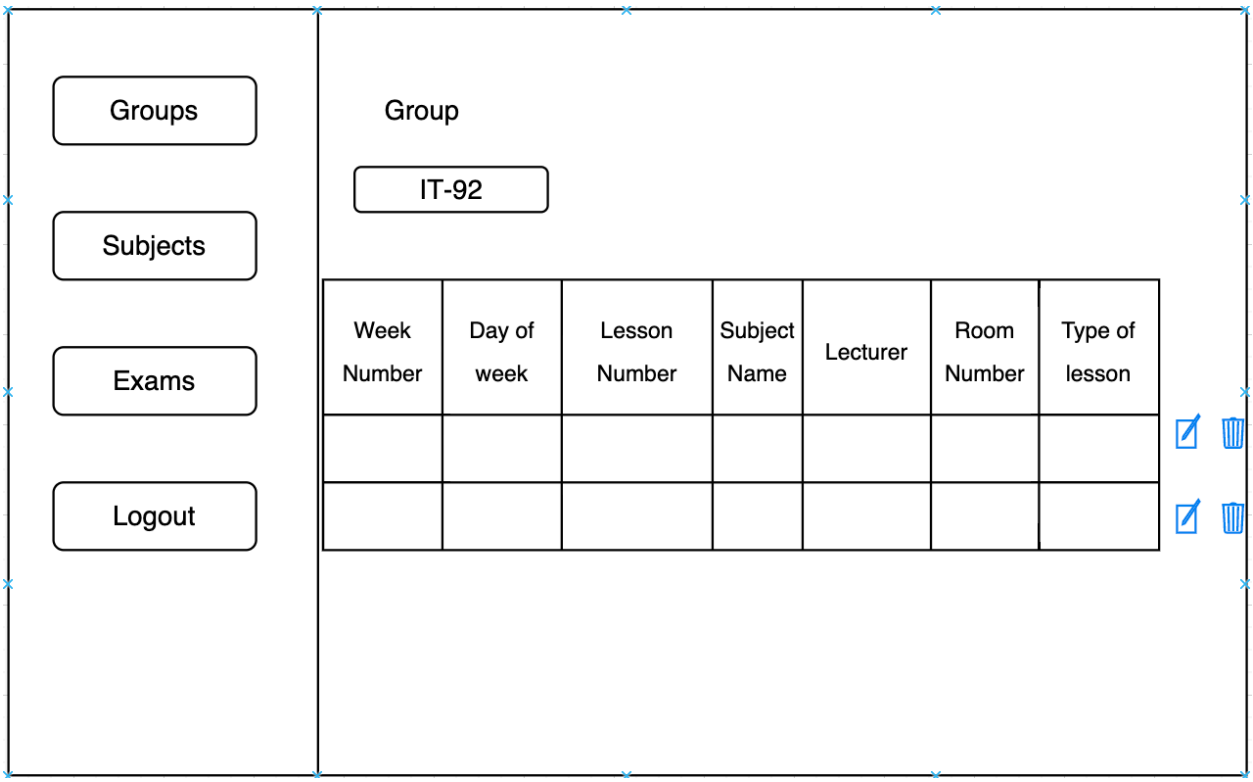


Рисунок 4.5 – Сторінка “Subjects”

– для перегляду розкладу екзаменів користувач може натиснути кнопку “Exams” у боковій панелі, перед ним відкриється розклад екзаменів для всіх груп в системі (рис 4.6);

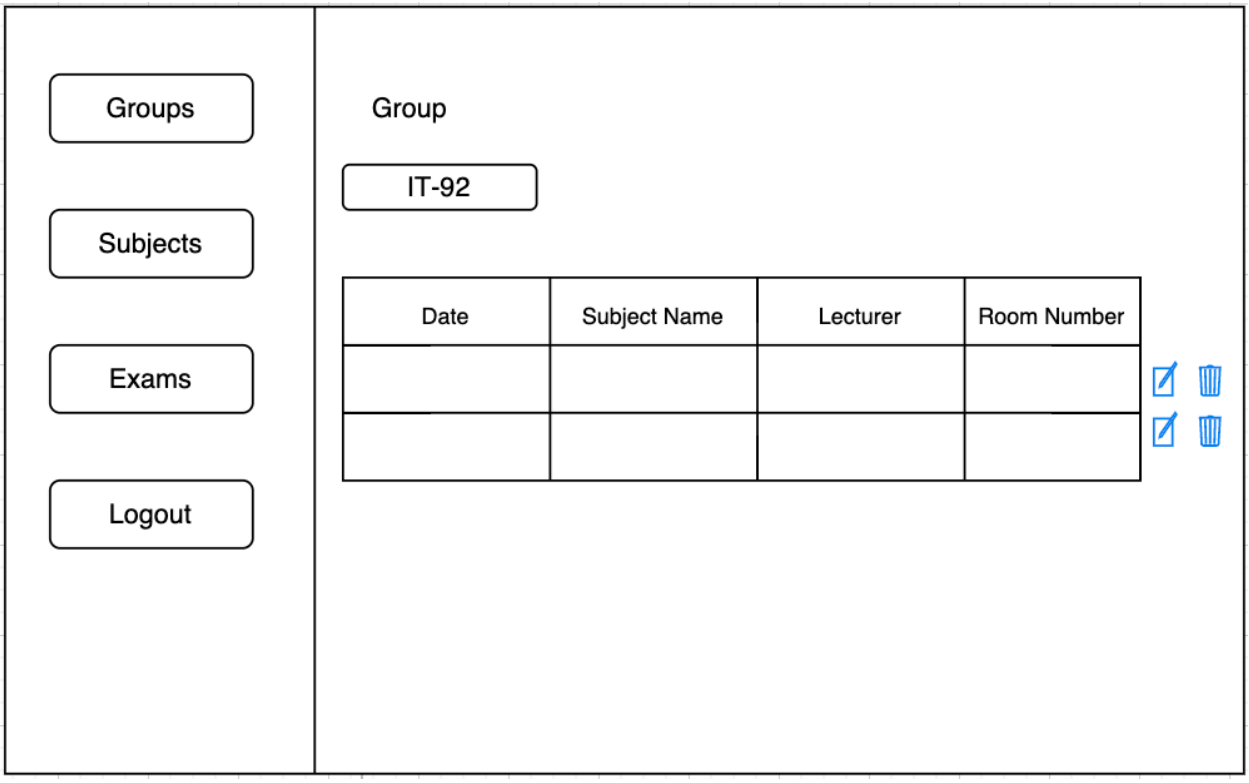


Рисунок 4.6 – Сторінка “Exams”

– на сторінці “Subjects” користувач може переглянути список всіх предметів, які доступні в системі, та додати новий предмет, натиснувши кнопку “Add Subject” (рис 4.7);

– на сторінці “Exams” користувач може переглянути список всіх предметів, які доступні в системі, та додати новий предмет, натиснувши кнопку “Add Exam” (рис 4.8);

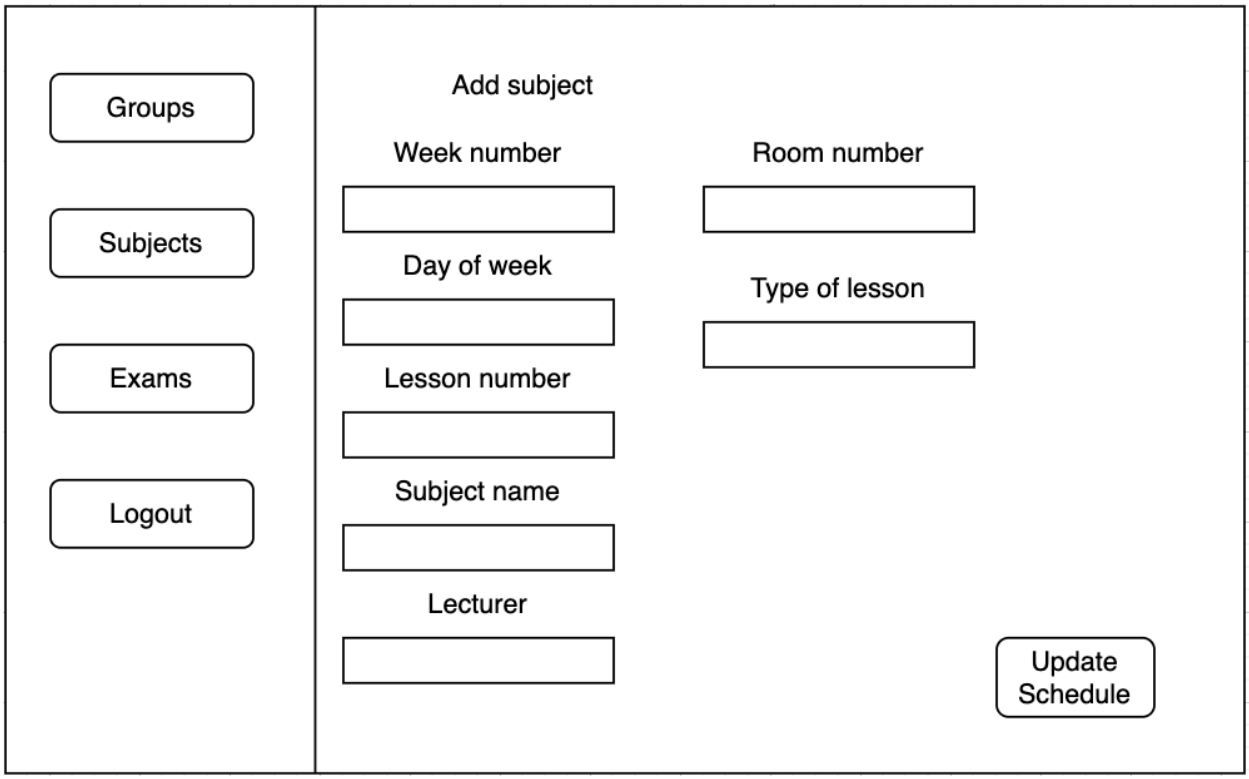


Рисунок 4.7 – Сторінка “Create Subject”

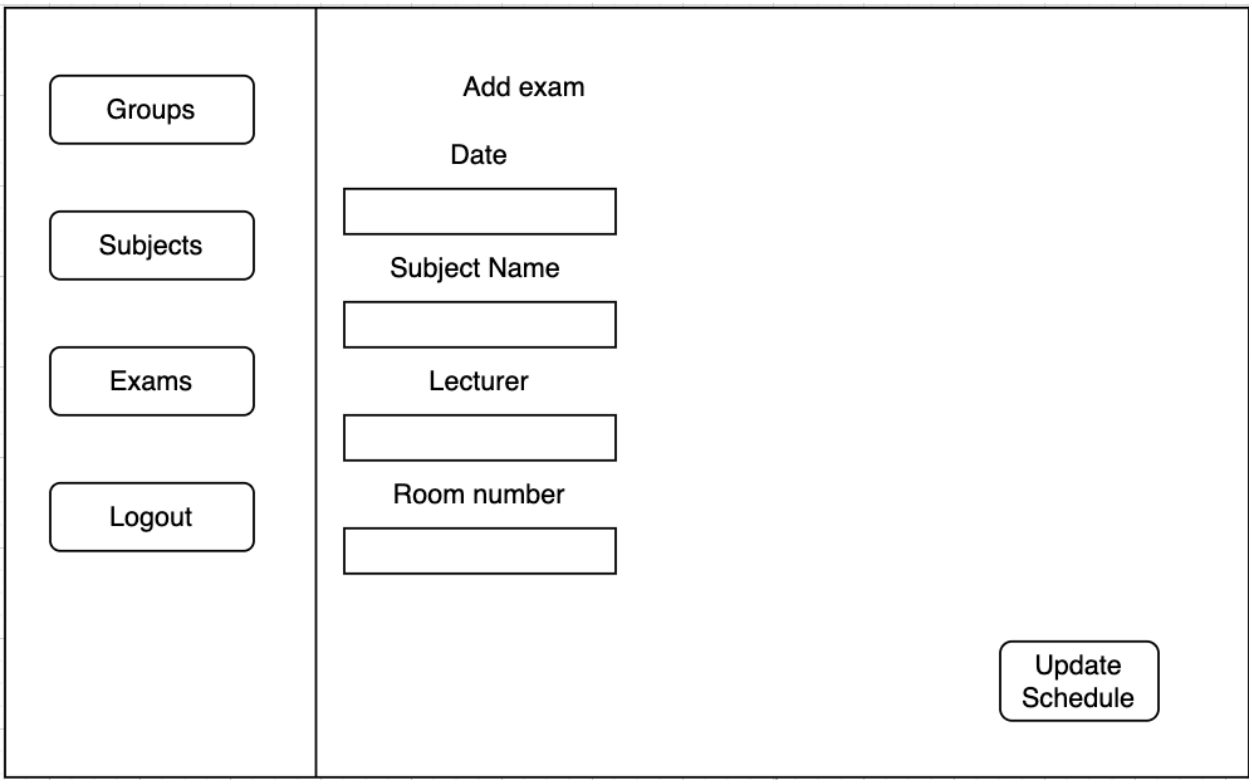


Рисунок 4.8 – Сторінка “Add Exam”

– після того, як користувач натисне кнопку “Add Subject”, він буде перенаправлений на сторінку з формою для додавання нового предмету. Щоб додати предмет, користувач повинен заповнити всі поля в формі та натиснути кнопку “Create Subject”;

– для редагування даних предмету на сторінці “Subjects” користувач повинен натиснути кнопку “Edit” біля потрібного предмету. Після цього він буде перенаправлений на сторінку з формою для редагування даних предмету. Після внесення необхідних змін користувач повинен натиснути кнопку “Save Changes”;

– для видалення предмету на сторінці “Subjects” користувач повинен натиснути кнопку “Remove” біля потрібного предмету. Після цього він буде попереджений, що видалення предмету є незворотнім процесом, та якщо він погоджується з цим, то повинен натиснути кнопку “Delete”;

– для редагування даних групи на сторінці “Exams” користувач повинен натиснути кнопку “Edit” біля потрібної групи. Після цього він буде перенаправлений на сторінку з формою для редагування даних групи. Після внесення необхідних змін користувач повинен натиснути кнопку “Update Schedule”;

– для видалення групи на сторінці “Exams” користувач повинен натиснути кнопку “Remove” біля потрібної групи. Після цього він буде попереджений, що видалення групи є незворотнім процесом, та якщо він погоджується з цим, то повинен натиснути кнопку “Update Schedule”;

4.1.3 Для користувача (адміністратора системи розкладу):

– авторизація користувача;
– відображення списку груп;
– відображення списку занять;

– відображення списку екзамені;
– можливість додавання групи;
– можливість доповнення та редагування розкладу занять;
– можливість доповнення та редагування розкладу екзаменів.

4.1.4 Для адміністратора системи :

– можливість відновлення даних з резервного зберігання;
– можливість налаштування автоматичної синхронізації;
– можливість часткової синхронізації.

4.1.5 Додаткові вимоги:

Додаткових вимог немає.

4.2 Вимоги до надійності

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

Система має зберігати всі важливі дані в базі даних, на диску. Не зберігати стан проведення операції в оперативній пам’яті. При процедури синхронізації необхідна наявність контрольний точок, можливість проведення синхронізації частинами, створення резервних копій даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються»

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 8 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;
– об‘єм ОЗП: 8 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 сімейства Unix. Передбачено розгортання за допомогою ПЗ “Докер”.

4.5.1 Вимоги до вхідних даних

Для обмінну даних використовується стандарт JSON.

Також вхідні данні вводяться за допомогою веб-застосунку.

4.5.2 Вимоги до вихідних даних

Результати повинні бути графічно представлені на інтерфейсі користувача.

Для обмінну даних використовується стандарт JSON.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування C# та TypeScript.

4.5.4 Вимоги до середовища розробки

Розробку виконати на допомогою середовище JetBrains Rider, PostgreSQL.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді файлів із розширеннями cs, ts, html, css.

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;
– керівництво системного адміністратора.

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

– схема структурна варіантів використання;
– схема структурна діяльності;
– схема структурна компонентів програмного забезпечення;
– схема бази даних;
– архітектура 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	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

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

Пояснювальна записка
до дипломного проєкту

на тему: Програмне забезпечення для підготовки та роботи з розкладом занять

КПІ.ІТ-9230.045440.02.81

Київ – 2023

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.
ЄІС		Єдине інформаційне середовище

ВСТУП

В сучасному світі освіта є дуже важливою складовою суспільства, все більшим стає попит на дипломованих спеціалістів - відповідно зростають вимоги до навчальних закладів. Все більшими стають вимоги до організації навчання студентів, бо зростає їх кількість і через це ускладняється організація навчального процесу. З 2020 року світові системи освіти постали перед серйозним викликом - організація навчального процесу в умовах пандемії COVID-19. Це вимагає від навчальних закладів впровадження нових технологій, які дозволять забезпечити навчання студентів в умовах дистанційного навчання.

Одним з основних елементів організації навчального процесу дистанційно є перенесення систем адміністрації університету в онлайн середовище. Такі системи як журнал оцінок, начальні матеріали, розклад занять мають бути максимально доступними для студентів і викладачів.

Тому в закладів освіти виникає потреба у відповідному програмному забезпеченні для того, щоб відповідати цим вимогам. До цього програмного забезпечення відповідно ставляться певні вимоги, такі як доступність, безпека, зручність використання, можливість розширення функціоналу, можливість розгортання на різних платформах, максимально широка підтримка користувачів.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

На даний момент існує велика кількість різноманітних систем управління навчальним процесом, які використовуються в університетах. Вони можуть бути розроблені як власними зусиллями університету, так і створені сторонніми компаніями. Ці системи можуть бути розроблені для використання в конкретному університеті, або ж бути універсальними. Існують готові рішення такі як Google Clasroom, Moodle, Blackboard, які можуть бути використані в будь-якому університеті.

Ці системи підходять до організації навчального процесу для асинхронної роботи, але не можуть бути використані для організації синхронних занять. Тому виникає потреба у створенні системи, яка буде використовуватися для організації синхронних занять.

Синхронне навчання забезпечує швидкий та безпосередній зворотний зв'язок між учителем та учнями. Відповіді та реакції надаються в режимі реального часу, надаючи можливість організувати безпосередню взаємодію учнів у малих групах, швидко обговорити питання та прийняти рішення. Однак, синхронне навчання вимагає онлайн-присутності в чітко визначений час та може стати проблемою для тих, хто має накладаючись графіки.

Натомість, асинхронний режим дозволяє працювати за власним графіком та у власному темпі, максимально використовуючи переваги змішаного навчання. Опанування матеріалу залежить від власного розуміння, а не темпу решти групи. Однак, асинхронний режим може викликати відчуття ізольованості та знизити відчуття навчальної спільноти, якщо спеціально не підтримувати його. Крім того, він вимагає від учнів високої самодисципліни та вміння керувати своїм часом.

Отже, враховуючи особливості цих форматів навчання, можна визначити види та форми діяльності, які дозволять ефективно використовувати кожен з режимів.

Синхронне навчання може бути організоване у вигляді лекцій, семінарів, практичних занять, лабораторних робіт, консультацій, індивідуальних занять, заліків, іспитів, захистів курсових робіт, дипломних робіт, випускних робіт, а також інших форм навчальної діяльності, які вимагають присутності учасників навчального процесу у одному місці та одночасно.

Системи розкладу направлені в першу чергу для координації процесу синхронного навчання. Розклад занять має включати всю необхідну інформацію про лекції, семінари, практичні заняття, лабораторні роботи, консультації, індивідуальні заняття, заліки, іспити, захисти курсових робіт, дипломних робіт, випускних робіт, а також інші форми навчальної діяльності, які вимагають присутності учасників навчального процесу у одному місці та одночасно.

Також немало важливою я необхідність інтеграції з внутрішніми системами університету. Це необхідно для інтеграції з внутрішньо університетською системою організації занять, забезпечення актуальності інформації про викладачів та студентів, а також для інтеграції з системою електронного документообігу.

1.2 Змістовний опис і аналіз предметної області

На даний момент існує достатньо велика кількість готових рішень, доступних на рівні інтеграції університету, курсу, чи окремого студента. Більшість наявних рішень є широко спеціалізованими або дуже вузько спеціалізованими і не відповідають потребам університету, курсу, чи окремого студента.

Такі рішення орієнтовані на асинхронний процес навчання і підходять до проблеми достатньо комплексно. Вони включають в себе різноманітні

інструменти для викладачів, студентів, адміністраторів, та інших учасників навчального процесу.

З технічної точки зору більшість систем використовують сучасні технології, такі як: Java, Python, PHP, JavaScript, HTML, CSS, та інші. Це дозволяє розробникам використовувати сучасні підходи до розробки, такі як: мікросервісна архітектура, розподілені системи, та інші. Більшість систем для організації навчального процесу, які використовуються в університетах України, є застарілими та не відповідають сучасним вимогам до програмного забезпечення з точки зору оптимізації, швидкодії та дизайну. Існують рішення, використовуванні як сервіси. Цей підхід далеко не завжди оптимальний для навчальних закладів, з різних причин - недостатньо можливостей для інтеграції, відсутність підтримки базового функціоналу (локалізація), дорогі тарифи, неможливість використання сторонніх ресурсів, та інші.

Відсутність універсального рішення, яке б відповідало потребам університету, курсу, чи окремого студента, є проблемою, яку необхідно вирішити. Це можна зробити шляхом розробки нового програмного забезпечення, яке відповідатиме потребам університету, курсу, чи окремого студента.

Розробка системи є комплексною задачею, яку складно вирішити кожному навчальному закладу окремо. Є певний список готових рішень для окремих університетів, але ці рішення часто не мають можливості інтеграції.

В університеті НТУУ КПІ існує готова система розкладу занять, яку можна знайти за посиланням http://rozklad.kpi.ua/. Ця система є інтегрованою з інфраструктурою та іншими програмним забезпеченням університету під назвою Єдине Інформаційне Середовище (далі - ЄІС).[2] Сайт підтримує весь базовий необхідний функціонал для розкладу занять, а саме:

- вибір групи;
- перегляд занять для обраної групи;
- перегляд розкладів для викладачів;
- перегляд екзаменів для групи;

– Розподіл розкладів занять на перший та другий навчальний тиждень.

– На даний момент, ця система вважається застарілою та такою, що не відповідає існуючим вимогам для ПЗ. З недоліків можна назвати:

– застарілість технологій;

– зав’язаність на іншій існуючій інфраструктурі;

– недостатня продуктивність системи;

– відсутність підтримки сучасних стандартів для веб-сервісів.

Через перелічений список недоліків практично неможливо реалізувати новий функціонал без шкоди для іншого ПЗ. Це утворює сильну зв’язність для наявної архітектури, тому що сервіс не може працювати без бази даних, яка в свою чергу тримає інші дані університету, які, можливо, є досить таки вразливими.

Необхідність використання загально університетської БД ускладнює процедуру адміністрування даних розкладів. Через вразливість даних в цій БД, підсистема заповнення даних розкладів є закритою та інтегрованою з іншими підсистемами. Через це дані розкладів є підмножиною інших даних університету, і не виділяються в окрему категорію, через це неможливо відокремити цей функціонал в наявній системі.

Залежність від інших складових та іншого ПЗ унеможливлює розгортання додатку окрему, що суперечить сучасним принципам організації ефективного ПЗ.

Як альтернатива, були розглянути інші сервіси, такі як Google Calendar, Moodle, Microsoft Outlook.

Ці сервіси підтримують фунціонал календарів, і мають підтримку окремих подій, циклічних подій, а також інтеграцію з навчальними платформами.

Проте жоден з перелічених сервісів не має можливості організації даних в потрібному форматі і з відповідним групуванням за типом користувача.

Через це, окрім ускладнення використання на боці користувача, виникають проблеми потенційної міграції і управління даними.

1.3 Аналіз існуючих технологій та успішних IT-проектів

Ця система за будовою є типовою системою з веб сервісів, що є дуже популярним програмним продуктом, тому існує широкий вибір інструментів для розробки цього продукту.

Основними інструментами є:

– мова програмування: основними мовами програмування для веб-сервісів є C#, Java, Python, Golang вибір мови програмування залежить від багатьох факторів, таких як уміння розробника, швидкість виконання, читаємість синтаксису, наявність підтримки;

– інтегроване середовище розробки (IDE). Є велика кількість IDE які підтримують для розробки більшість сучасних мов програмування, такі як – Intelij IDEA, Rider, Visual Studio, Visual Studio Code;

– фреймоворки та бібліотеки: для розробки веб-сервісу необхідно оброблювати велику кількість логіки для взємодії з протоклами HTTP та HTTPs, тому доцільнішим буде використання фреймоврку, який буде підтримувати цей функціонал;

– база даних: база даних необхідна практично в будь-якому веб-проекті, який забезпечує роботу з даними. Популярні бази даних діляться на реляційні та нереляційні. До реляційних баз даних відносяться PostgresSQL, MySQL, SQLite.

Мова програмування C#

Як основна мова програмування була обрана мова C#.

C# — це об’єктно-орієнтована мова програмування, назва якої вимовляється як «see sharp».[7] Назва походить від музичної ноти C#, яка є нотою C, підвищеною на півтону. По суті, C# — це оновлена та вдосконалена версія мови C.

C# має сувору типізацію та включає автоматичне збирання сміття, що звільняє програміста від тягаря ручного керування пам’яттю. Пізніше мова додала динамічне зв’язування, асинхронні методи та шаблони.

C# схожий на Java і є його головним конкурентом. Обидві мови розвивають концепції C++, одночасно спрощуючи її. Ці дві мови поділяють однакову філософію, але є технічні нюанси, які їх відрізняють. Ці відмінності пов’язані з тим, що C# було розроблено для екосистеми Microsoft, тоді як Java була розроблена як середовище з відкритим кодом. Наприклад, C# має різні підходи до реалізації параметризованих типів і не має перевірених винятків Java.

Основною перевагою, через яку був обраний саме C# є широкий інструментарій мови, для більшості можливих труднощів скоріше за все буде наявне рішення. А саме такі речі, як:

– широкий вибір бібліотек, які можна використовувати в проекті. З справді корисних можна виділити бібліотеки для передачі даних по протоколу HTTP, бібліотеки для роботи з JSON і XML файлами, бібліотеки для підключення і операцій над базою даних, логування. Багато з цих рішень, як наприклад, бібліотека для роботи з JSON файлами і логуванням, входять в стандартну поставку .NET Core, що значно спрощує роботу з ними. Через широку підтримку данної мови програмування для практично всієї множини задач, з якими можна стикнутись при розбробці веб-сервісу, будуть мати готове стандартне рішення. Через таку широку підтримку, відпадає необхідність в самостійному написанні бібліотек і перевірці на надійність і доцільність використання, що значно прискорює розробку і додає надійності до проекту "з коробки";

– потужна підтримка ASP.NET. Для побудови веб-сервісу значна частина коду буде відповідати за роботу з протоколом HTTP, а також за підтримку стандартного функціоналу веб-сервісу, такого як: роутинг, авторизація, аутентифікація, робота з сесіями, робота з файлами, робота з базою даних, логування. Для реалізації цих задач в ASP.NET є готові рішення,

які можна використовувати в проекті. Це значно спрощує розробку і дозволяє зосередитись на реалізації бізнес-логіки проекту, а не на реалізації стандартного функціоналу;

– спільнота спеціалістів, які займаються розвитком технології а також відкритих проектів. Це я важливим фактором, тому що як сказано вище, технологія C# є дуже популярною в Україні і тому у випадку якщо проект буде мати відритий код, його підтримка і розширення функціоналу буде значно простішою для сторонніх спеціалістів. За останній десять років, значний прогрес відбувся в сфері відкритого коду. Даний проект задумується як університетська система, і звісно навіть після завершення розробки й тестування скоріше за все буде виникати потреба в новому функціоналі та виправлення помилок, тому підтримка з боку незалежних спеціалістів є дуже важливою.

IDE Rider

Як основна IDE був обраний JetBrains Rider. Цей інструмент надає дуже зручні інструменти для написання і аналізування коду на мові C#. Оскільки для розробки використовується система MacOs, це унеможливлює використання Visual Studio 2022, тому дана IDE є найкращою альтернативою.

ASP.NET Core

ASP.NET є технологією створення веб-додатків та веб-сервісів, яка розробляється компанією Microsoft та є частиною платформи Microsoft .NET. Вона є еволюцією старішої технології Microsoft ASP, і має багато спільного зі своїм попередником, що дозволяє розробникам легко перейти на нову технологію. Проте, внутрішній механізм ASP.NET відрізняється від ASP, оскільки він заснований на платформі .NET та використовує всі нові можливості, які ця платформа надає. Останньою версією технології на даний момент є ASP.NET Core 6.0.[5]

PostgreSQL

PostgreSQL — це об’єктно-реляційна система керування базами даних (ORDBMS). Це альтернатива як комерційним СУБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СУБД з відкритим кодом (MySQL, Firebird, SQLite).

В предметній області розкладу занять можливо виокремити ряд сутностей, пов'язаних між собою, такі як лектори, уроки, аудиторії, також корисним може бути функціонал для оптимізації пошукових запитів за допомогою індексів, системи резервації данних за допомогою процедур, використання транзакцій для забезпечення консистенції даних. Тому для обраного додатку реляційна база даних є найбільш оптимальним варіантом. Для реалізації бази даних було обрано PostgreSQL, оскільки вона є вільною, має великий набір функцій, відповідає вимогам ACID, має велику кількість додаткових модулів, які можна використовувати для розширення функціоналу.

1.4 Аналіз вимог до програмного забезпечення

Програмне забезпечення ділиться на модулі: інтерфейс користувача, серверна частина інтерфейсу, серверна частина для синхронізації бази даних.

В програмному забезпеченні передбачені 2 актори: Адміністратор системи розкладу і Системний адміністратор (рис. 1.1).

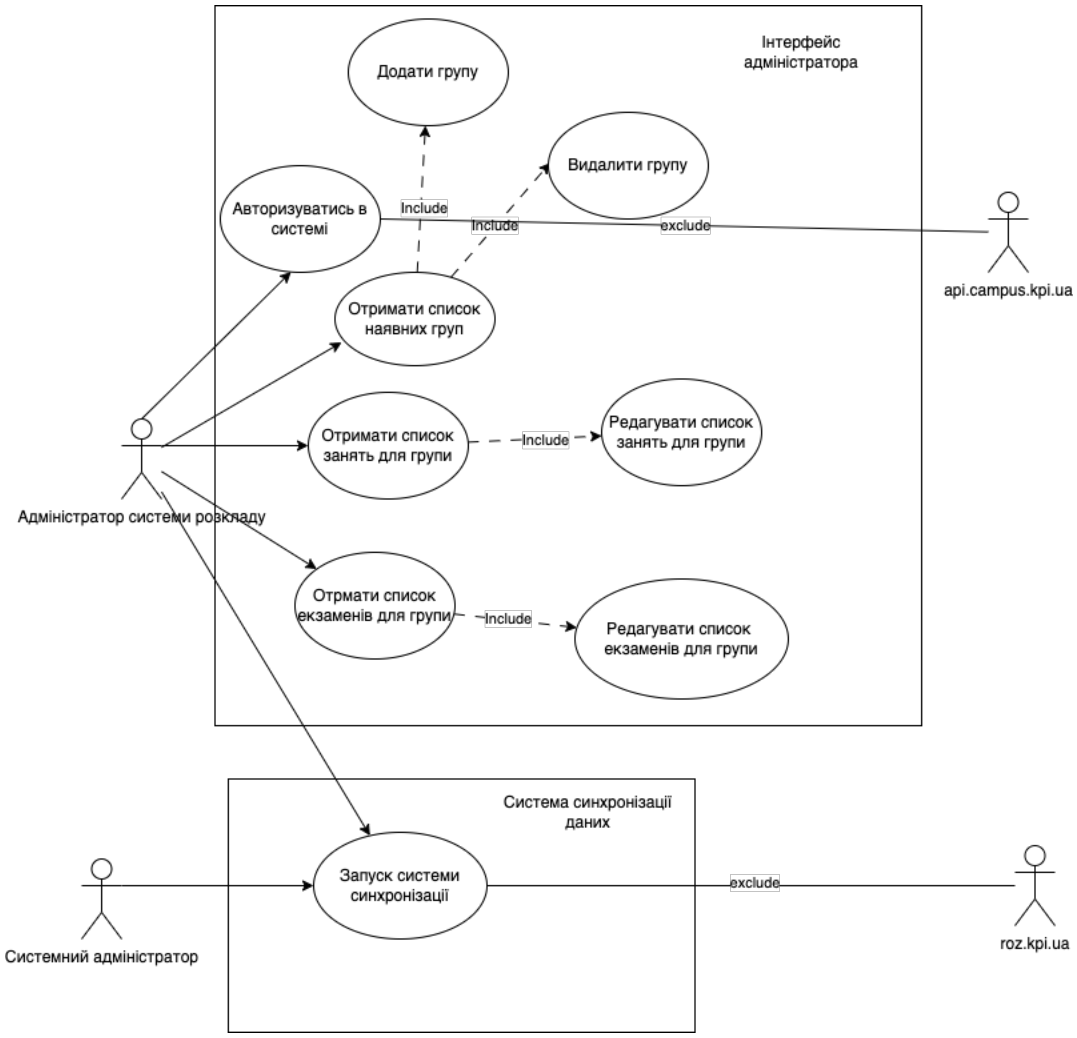


Рисунок 1.1 – Діаграма варіантів використання

Варіанти використання наведені в таблицях 1.1 – 1.9.

Таблиця 1.1 – Варіант використання UC-01

Use case name	Автентифікація користувача до інтерфейсу адміністратора
Use case ID	UC-01
Goals	Логін користувача
Actors	Користувач з правами адміністратора
Trigger	Користувач використати функціонал інтерфейсу адміністратора
Pre-conditions	1. Данні користувача присутні в системі електронний кампус.
Flow of Events	1. Користувач переходить на сторінку логіна. 2. Користувач вводить логін і пароль.
Extension	-
Post-Condition	Користувач ввійшов до інтерфейсу адміністратора

Таблиця 1.2 – Варіант використання UC-02

Use case name	Автоматизований процес синхронізації розкладу занять
Use case ID	UC-02
Goals	Автоматизований регулярний процес синхронізації бази даних розкладу занять
Actors	Системний адміністратор
Trigger	Потреба регулярної синхронізації даних розкладів
Pre-conditions	1. Користувач має доступ до сервера з налаштованою системою синхронізації бази даних 2. Користувач має досвід роботи з командним рядком
Flow of Events	1. Користувач заходить на сервер за допомогою командного рядка 2. Користувач виконує скрипт вказаний в інструкції
Extension	-
PostCondition	База даних розкладу синхронізована

Таблиця 1.3 – Варіант використання UC-03

Use case name	Перевірка наявності групи в базі даних за допомогою інтерфейсу адміністратора
Use case ID	UC-03
Goals	Перевірити наявність групи в базі даних
Actors	Користувач з правами адміністратора
Trigger	Користувач хоче перевірити наявність групи в БД
Pre-conditions	1. Користувач авторизований
Flow of Events	1. Перейти на сторінку "Групи". 2. Ввести назву групи в поле.
Extension	-
Post-Condition	Можливість додавати нові групи в систему

Таблиця 1.4 – Варіант використання UC-04

Use case name	Переглянути список занять для групи
Use case ID	UC-04
Goals	Відображення списку занять для обраної групи
Actors	Користувач з правами адміністратора
Trigger	Користувач бажає переглянути список занять для розкладу обраної групи
Pre-conditions	1. Користувач авторизований 2. Група повинна бути існуючою в базі даних
Flow of Events	1. Користувач вибирає групу зі списку доступних 2. Відображається список уроків для обраної групи
Extension	-
Post-Condition	Користувач може переглядати список уроків для групи

Таблиця 1.5 – Варіант використання UC-05

Use case name	Додати групу
Use case ID	UC-05
Goals	Відображення списку уроків для обраної групи
Actors	Користувач з правами адміністратора
Trigger	Користувач бажає переглянути список уроків для групи
Pre-conditions	1. Користувач авторизований 2. Група повинна бути існуючою в базі даних
Flow of Events	1. Користувач вибирає групу зі списку доступних 2. Відображається список уроків для обраної групи
Extension	-
Post-Condition	Користувач може переглядати список уроків для групи

Таблиця 1.6 – Варіант використання UC-06

Use case name	Редагувати заняття
Use case ID	UC-06
Actors	Користувач з правами адміністратора
Trigger	Користувач бажає редагувати заняття в розкладі для обраної групи
Pre-conditions	1. Користувач авторизований 2. Група повинна бути існуючою в базі даних 3. Урок повинен існувати в розкладі для групи
Flow of Events	1. Користувач вибирає групу зі списку доступних 2. Користувач обирає урок для редагування
Extension	-
Post-Condition	Урок успішно відредагований в розкладі для групи

Таблиця 1.7 – Варіант використання UC-07

Use case name	Додати заняття для групи
Use case ID	UC-07
Goals	Оновити розклад групи
Actors	Користувач з правами адміністратора
Trigger	Користувач бажає оновити розклад групи
Pre-conditions	1. Користувач авторизований 2. Група повинна бути існуючою в базі даних
Flow of Events	1. Користувач вибирає групу зі списку доступних 2. Користувач вводить дані заняття в відповідні поля в формі для нового заняття
Extension	-
Post-Condition	Заняття в успішно додано до розкладу групи.

Таблиця 1.8 – Варіант використання UC-08

Use case name	Переглянути список екзаменів для групи
Use case ID	UC-08
Goals	Відображення списку екзаменів для обраної групи
Actors	Користувач з правами адміністратора
Trigger	Користувач бажає переглянути список екзаменів для розкладу обраної групи
Pre-conditions	1. Користувач авторизований 2. Група повинна бути існуючою в базі даних
Flow of Events	1. Користувач вибирає групу зі списку доступних 2. Відображається список екзаменів
Extension	-
Post-Condition	Користувач може переглядати список екзаменів для групи

Таблиця 1.9 – Варіант використання UC-09

Use case name	Додати екзамен для групи
Use case ID	UC-09
Goals	Оновити розклад екзаменів для обраної групи
Actors	Користувач з правами адміністратора
Trigger	Користувач бажає оновити список екзаменів для розкладу обраної групи
Pre-conditions	1. Користувач авторизований 2. Група повинна бути існуючою в базі даних
Flow of Events	1. Користувач вибирає групу зі списку доступних 2. Відображається список екзаменів
Extension	-
Post-Condition	Користувач може переглядати список екзаменів для групи

Дослідимо функціональні вимоги для системи (табл. 1.9).

Таблиця 1.9 – Функціональні вимоги

Код вимоги	Опис вимоги	Опис
FR-1	Авторизація користувача	Користувач може ввести логін і пароль та виконати процес авторизації.
FR-2	Запуск процесу синхронізації	Користувач може запустити процес синхронізації бази даних.
FR-3	Отримання списку груп	Користувач може переглянути список груп в системі.

Кінець таблиці 1.9

FR-4	Отримання списку уроків	Користувач може переглянути список уроків в системі.
FR-5	Отримання списку екзаменів	Користувач може переглянути список екзаменів в системі.
FR-6	Додати групу	Користувач може додати групу.
FR-7	Оновити список уроків для групи	Користувач може додати, видалити та редагувати список занять для групи.
FR-8	Оновити список екзаменів для групи	Користувач може додати, видалити та редагувати список екзаменів для групи.

Дослідимо таблицю вимог в загальному.

Таблиця 1.10 – Модель вимог в загальному вигляді

№	Назва	ID Вимоги	Пріоритет	Ризик	Статус
1	Авторизація користувача	FR-1	Високий	Низький	Підверджено
2	Запуск процесу синхронізації	FR-2	Високий	Низький	Підверджено
3	Отримання списку груп	FR-3	Низький	Низький	Підверджено
4	Отримання списку уроків	FR-4	Низький	Низький	Підверджено

Кінець таблиці 1.10

5	Отримання списку екзаменів	FR-5	Низький	Низький	Підверджено
6	Додати групу	FR-6	Низький	Низький	Підверджено
7	Оновити список уроків для групи	FR-7	Низький	Низький	Підверджено
8	Оновити список екзаменів для групи	FR-8	Низький	Низький	Підверджено

Розглянемо матрицю трасування вимог (табл. 1.11).

Таблиця 1.11 – Матриця трасування вимог

	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9
FR-1	+		+	+	+	+	+	+	+
FR-2		+							
FR-3			+		+				
FR-4				+		+	+		
FR-5								+	+
FR-6					+				
FR-7						+			
FR-8								+	+

Постановка задачі

Перед цим проектом стоять наступні задачі:

1. Інтеграція нової системи розкладу з Єдиною Інформаційною Системою університету та виведення з експлуатації попередньої системи розкладу занять.

2. У зв’язку з проблемами, спричиненими знеструмленням датацентру, забезпечення резервації даних і можливість перенесення системи в хмарне середовище.

Висновки до розділу

В цьому розділі було проведено аналіз наявних програмних рішень для задачі онлайн розкладу занять. Був проведений аналіз вимог до такого програмного рішення а також можливості для реалізації. В результаті був отриманий загальний опис вимог до системи, а саме:

– розробка окремої системи для організації онлайн розкладу занять;

– підтримки сумісності і релевантної даних з наявними внутрішніми системами ЄІС. Розробка зберігання та резервування даних;

– розробка інтерфейсу адміністратора для ручного редагування розкладу та управління;

– розробка системи авторизації для обмеження доступу до інтерфейсу адміністратора.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1Моделювання та аналіз програмного забезпечення

Головна задача, яка стоїть перед цим проектом, це інтеграція з внутрішніми сервісами університету (ЄІС), але без сильної зв'язності між сервісами інфраструктури університету. Для цього було прийнято рішення налаштувати окрему базу даних, де могли б зберігатися оброблені данні. Для того щоб налагодити синхронізацію даних для цієї бази даних, було вирішено розробити сервіс для синхронізації. Також буде розроблено інтерфейс адміністратора для заповнення даних вручну, якщо це буде необхідно.

Так як цей проєкт є доповненням для системи “Електронний Кампус”, то основний стек був обраний таким чином, щоб він відповідав основному проєкту[1]. Таким чином було обрано платформу .NET і базу даних PostgreSQL.

2.1.1 Аналіз структури даних

Головною складовою цієї системи є модель даних. Тому першим кроком в моделюванні ПЗ було створено діаграму сутностей (рис. 2.1)

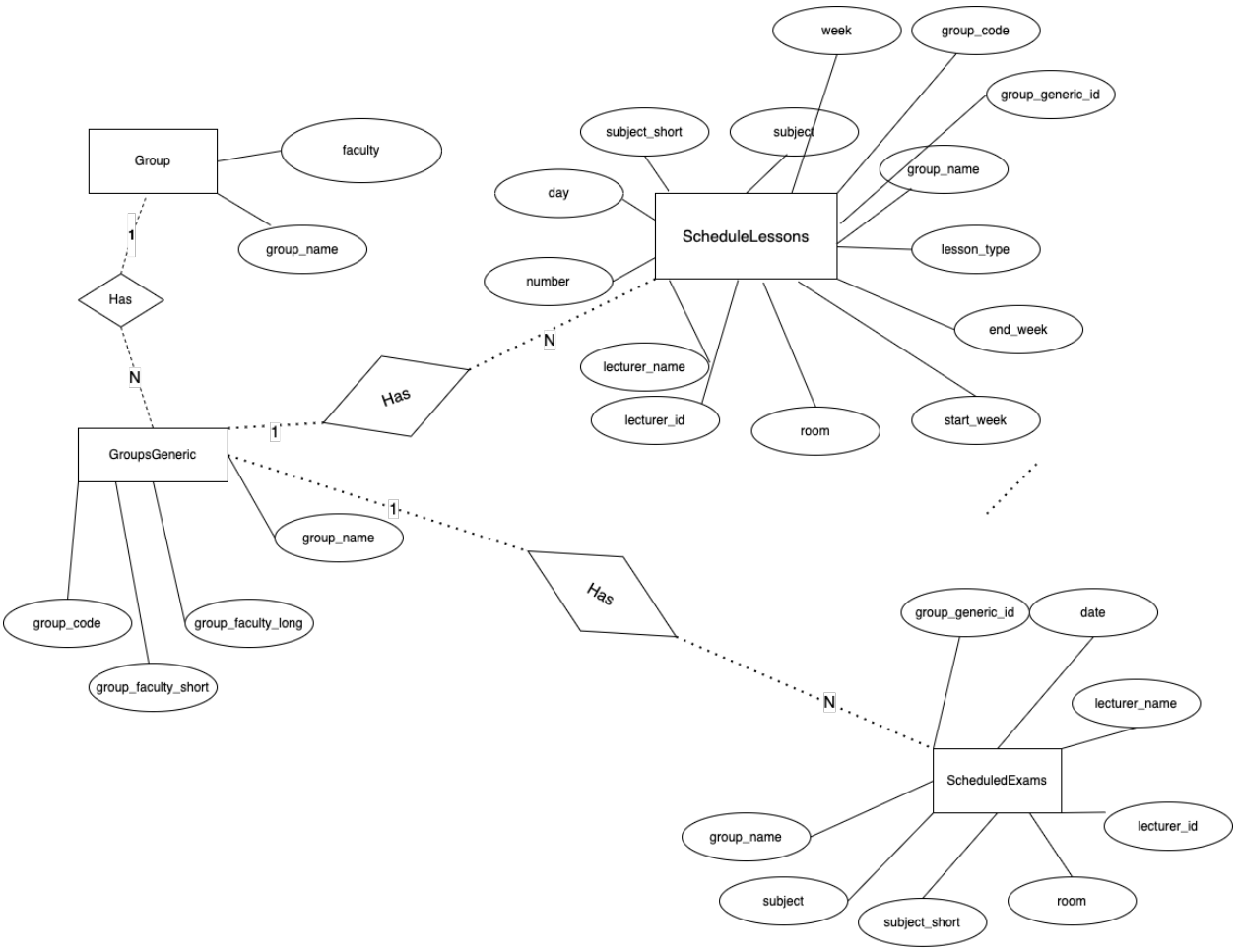


Рисунок 2.1 – Діаграма сутностей

Як можна бачити, основні сутності цього проекту є: Група, Урок і Екзамен. Сутності були визначені відповідно до вимог до проекту, тобто мають зберігати достатньо інформації щоб відображати розклад занять. Сутності організовані таким чином, щоб зберегти сумісність з внутрішніми моделями даних сервісів.

На рисунку 2.1 зображено, що ключовою сутністю є група, до неї сходяться зв’язки Занять (“ScheduledLessons”) та Екзаменів (“ScheduledExams”). Можна виокремити сутності Група (“Group”) та Генерована група “GroupGeneric”. Ці сутності несуть схожі дані, але мають різне призначення:

– “Group” є сутністю, яка використовується для конфігурації синхронізації даних. Множина цих елементів є множиною текстових

найменувань груп. Назва групи не є обов'язково унікальною в межах університету, тому що групи можуть повторюватись в межах різних факультетів. Тому це поле не можна використовувати для групування даних розкладу;

– "GroupGeneric" є сутністю унікальною для групи в рамках університету. Поле "group_code" містить унікальний ідентифікатор для групи, тому цей елемент може бути використаний для групування інших елементів для розкладу групи;

– "ScheduledLessons" відповідає за збереження даних для окремого заняття, пари;

– "ScheduleExams" відповідає за збереження даних для екзамену або заліку.

2.1.2 Аналіз механізму синхронізації даних

Ця частина є основною складовою процесу синхронізації з системами університету. Так як повноцінна заміна системи для заповнення розкладу з самого початку не була можлива, було розроблено сервіс, який би синхронізував данні з базою даних університету. За рахунок цього забезпечується сумісність і актуальність даних.

Для початку синхронізації даних необхідно мати повний список груп в системі. Потім за допомогою алгоритму черги буде необхідно отримати розклад для кожної групи, агрегувати данні і зберегти в базі даних.

Важливим елементом є механізми резервації даних. Так як ця система синхронізації була розрахована на роботу в умовах технічних і логічних збоїв системи, важливим є резервація наявних даних для того, щоб у разі проваленої процедури синхронізації, данні не були втрачені і система могла продовжити роботу. Для чого використовуються процедури в базі даних, які зберігають усі наявні данні в окремій таблиці, і в разі проваленої процедури синхронізації,

данні можуть бути відновлені. Блок схему цього процесу можна побачити на рисунку 2.2.

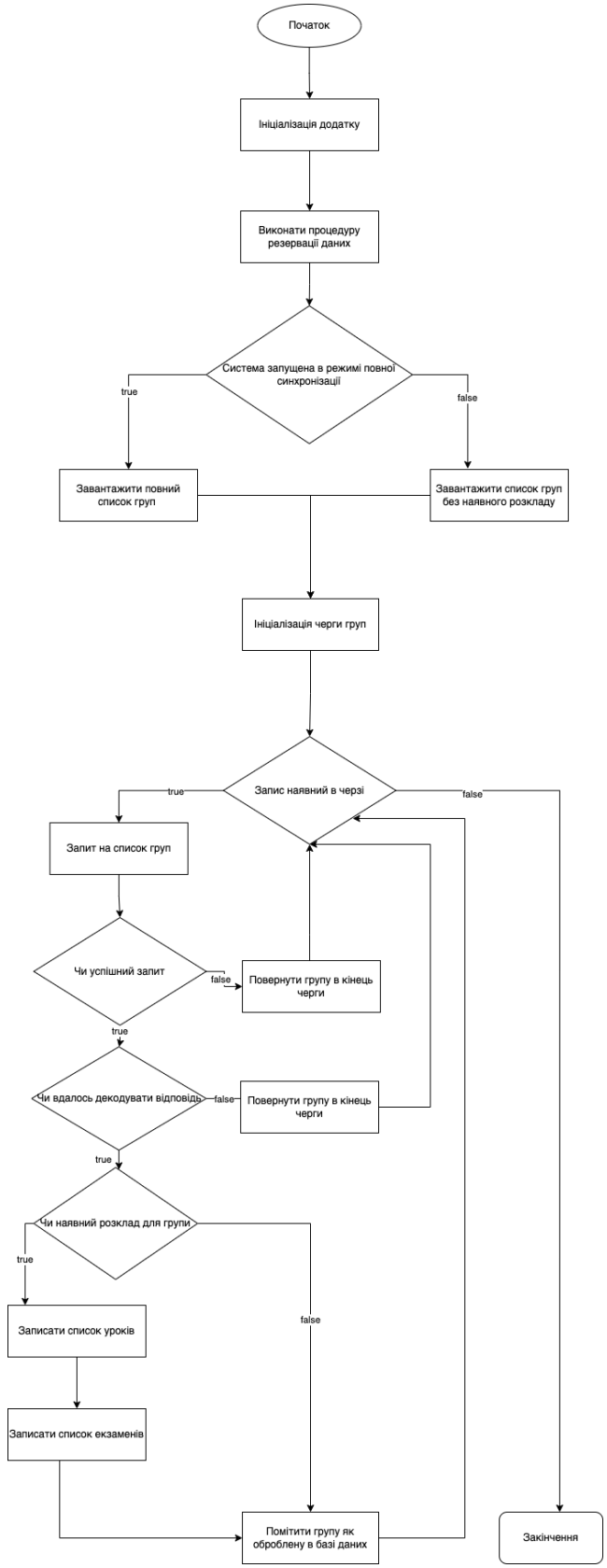


Рисунок 2.2 – Алгоритм процесу синхронізації

На діаграмі зображена логіка синхронізації даних з бази даних університету, вона включає:

– можливість повного оновлення даних та режим часткового оновлення;

– формування черги з груп для оновлення даних;

– логіку обробки помилок в обробці.

Цей сервіс має підтримувати режим повного оновлення даних та режим часткового оновлення даних. Це є важливим для забезпечення роботи в умовах нестабільного доступу до систем університету.

Режим повного оновлення може бути використаний регулярно для забезпечення актуальності розкладів занять, він передбачає завантаження розкладів для всіх конфігурований груп і може використовуватись автоматично. В даному випадку всі наявні розклади переносяться в резерв і відбувається заповнення “з нуля”.

Режим часткового оновлення не передбачає резервацію усіх наявних даних, і відповідає лише за заповнення пропусків. Цей режим може бути використаний, коли неможливо провести повну процедуру оновлення. Така потреба може виникнути за певних обставин, наприклад коли через відсутність стабільного електропостачання системи університету не працюють певний час, і проміжок часу, коли ці системи доступні онлайн менший за час повного оновлення.

Інтерфейс адміністратора

Цей модуль складається з серверного АПІ і клієнта у вигляді фронтенд додатку. Використання цієї системи передбачено для ручного контролю процесу синхронізації, заповнення / редагування даних вручну. За допомогою фронтенду користувач може переглядати данні розкладу. АПІ дозволяє виконувати CRUD операції над даними розкладу. Обидва сервіси мають працювати з однією базою даних.

Доступ під іменем адміністратора передбачає авторизацію користувача. Імплементація авторизації буде виконана за допомогою JWT токенів. Імплементація авторизації це дуже комплексна з точки зору безпеки тема, яка вимагає окремого дослідження. Тому в цьому проекті буде використаний провайдер авторизації з основного проекту “Електронний Кампус”. За допомогою цього в додатку немає необхідності обробляти запити щодо безпеки збереження даних користувачів.

2.2 Архітектура програмного забезпечення

Це ПЗ спроектувати як частину оновленої інфраструктури університету, проекту “Електронний Кампус”. Тому було обрано архітектуру модулів для цієї системи.

Програмне забезпечення складається з 4 частини :

– Campus.Core – базова бібліотека

– Campus.Schedule.Dashboard – модуль для API інтерфейсу адміністратора;

– Campus.Schedule.Parser – модуль синхронізації даних;

– Клієнтська частина;

– База даних.

Архітектуру зображено на Рисунку 2.3.

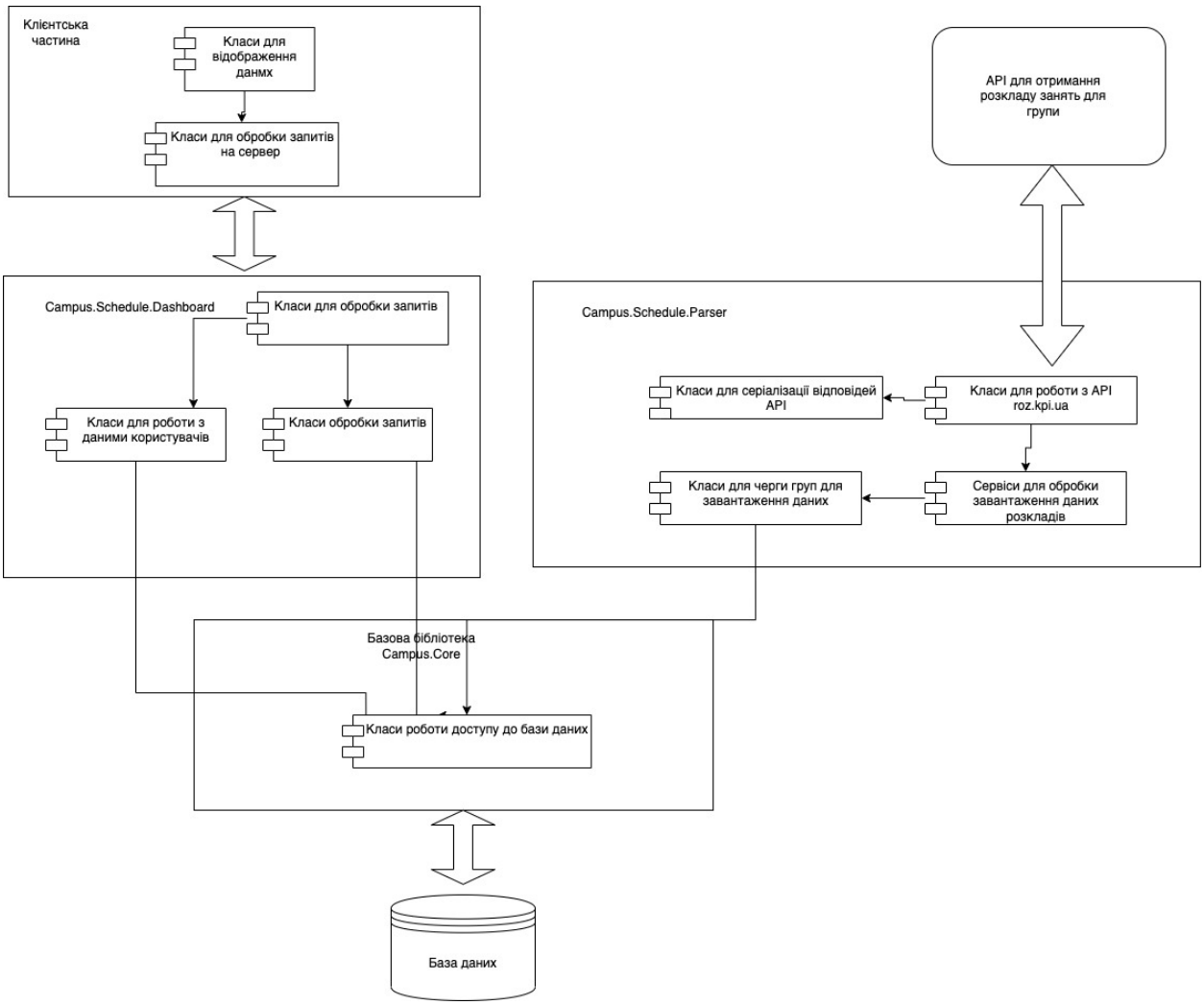


Рисунок 2.3 – Архітектура ПЗ

На рисунку можна побачити взаємне розташування модулів на вза’ємодію з зовнішніми API.

2.3 Конструювання програмного забезпечення

Для розробки бази даних був обраний підхід Database First.[5] Database First передбачає створення в першу черги бази даних за допомогою засобів управління базами даних, тобто скоріш за все у даному випадку - мови запитів SQL. Відповідно до таблиць бази даних створюються моделі в бізнес логікі проекту.

Це дозволяє знизити рівень абстракції при роботі з базою даних. Для цього ПЗ я вважаю це перевагою, тому що це додає гнучкості і прибирає залежність на технології, які мають високі вимоги до організації кодової бази певним чином. Хоч це і вимагає написання і тестування більшої кількості коду,

але позитивно впливає на прозорість системи, спрощуючи її подальшу підтримку.

Для доступу до бази даних було використано бібліотеку Dapper і драйвер для мови C# для бази даних Postgres. Використання цього підходу вимагає написання SQL запитів до бази даних і написання конвертації з типу Cursor у відповідний тип у бізнес логікі додатку. Схему бази даних зображено на рисунку 2.4.

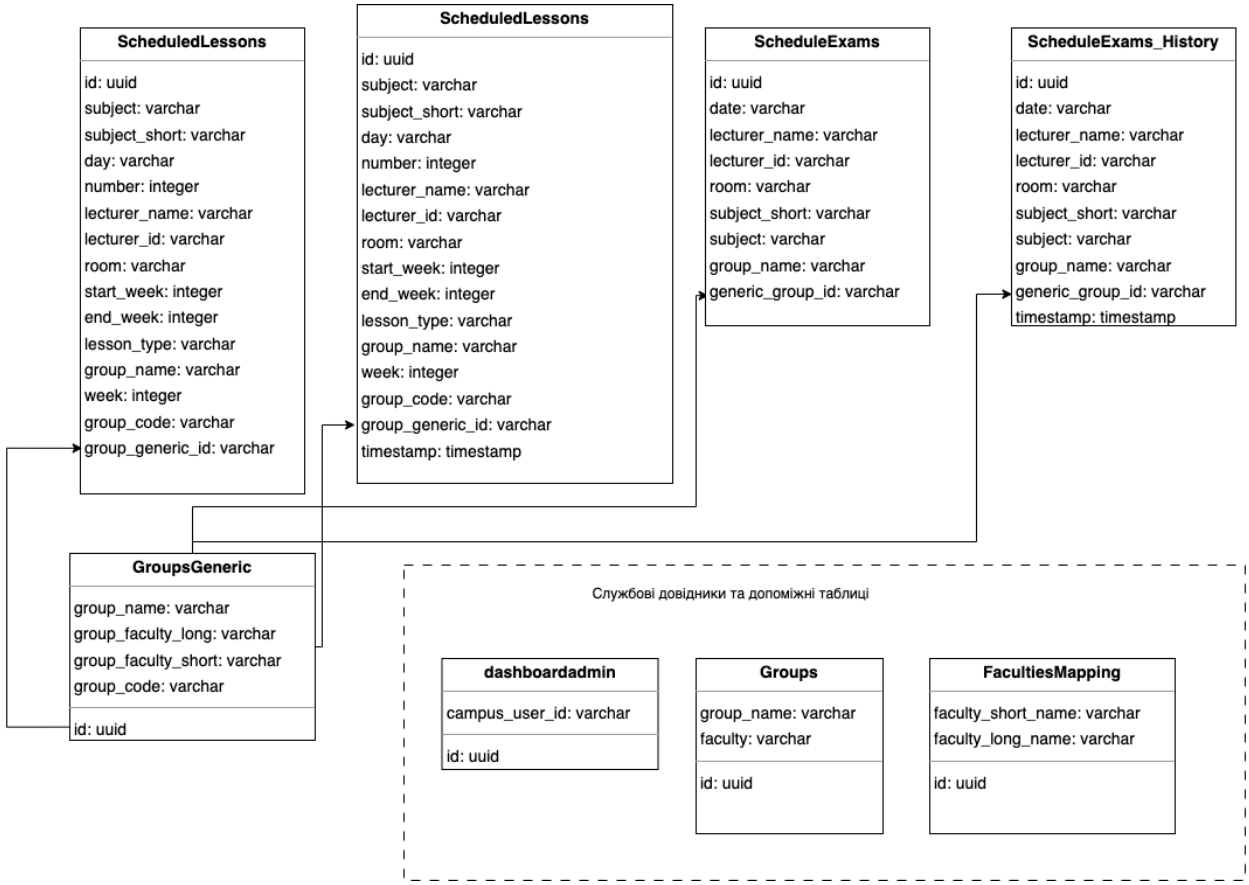


Рисунок 2.4 – Схема бази даних

Опис таблиць бази даних наведено в таблицях 2.1 – 2.8.

Таблиця 2.1 – Опис таблиці Groups

Поле	Тип даних	Опис
id	uuid	Універсальний унікальний ідентифікатор групи
group_name	varchar	Назва групи

Кінець таблиці 2.1

Faculty	varchar	Назва факультету

Таблиця 2.2 – Опис таблиці GroupsGeneric

Поле	Тип даних	Опис
id	uuid	Універсальний унікальний ідентифікатор групи
group_name	varchar	Назва групи
group_faculty_long	varchar	Повна назва факультету групи
group_faculty_short	varchar	Скорочена назва факультету групи
group_code	varchar	Код групи

Таблиця 2.3 – Опис таблиці FacultiesMapping

Поле	Тип даних	Опис
id	uuid	Універсальний унікальний ідентифікатор для запису
faculty_short_name	varchar	Абревіатура від назви факультету
faculty_long_name	varchar	Повна назва факультету

Таблиця 2.4 – Опис таблиці DashboardAdmin

Поле	Тип даних	Опис
id	uuid	Універсальний унікальний ідентифікатор для запису
campus_user_id	varchar	Ідентифікатор користувача в системі “Електронний Кампус”

Таблиця 2.5 – Опис таблиці ScheduledLessons

Поле	Тип даних	Опис

Кінець таблиці 2.5

Id	uuid	Універсальний унікальний ідентифікатор заняття
subject	varchar	Назва предмета
subject_short	varchar	Скорочена назва предмета
day	varchar	День тижня (Пн / Вв / Ср / Чт / Пт / Сб)
number	integer	Номер заняття
lecturer_name	varchar	Ім'я викладача
lecturer_id	varchar	Ідентифікатор викладача у форматі GUID - 128-розрядне ціле число, яке має дуже низьку ймовірність дублювання.
room	varchar	Аудиторія (хх-х-ххх / хх-хх / хх-ххх та інші формати)
start_week	integer	Номер початкового тижня
end_week	integer	Номер кінцевого тижня
lesson_type	varchar	Тип заняття (Прак on-line / Лаб / 0,5 Лек та інші)
group_name	varchar	Назва групи
week	integer	Номер тижня
group_code	varchar	Унікальний код групи
group_generic_id	varchar	Ідентифікатор групи в "GroupsGeneric"

Таблиця 2.6 – Опис таблиці ScheduledLessons_History

Поле	Тип даних	Опис
id	uuid	Універсальний унікальний ідентифікатор заняття

Кінець таблиці 2.6

Subject	varchar	Назва предмета
subject_short	varchar	Скорочена назва предмета
day	varchar	День тижня (Пн / Вв / Ср / Чт / Пт / Сб)
number	integer	Номер заняття
lecturer_name	varchar	Ім'я викладача
lecturer_id	varchar	Ідентифікатор викладача у форматі GUID - 128-розрядне ціле число, яке має дуже низьку ймовірність дублювання.
room	varchar	Аудиторія (хх-х-ххх / хх-хх / хх-ххх та інші формати)
start_week	integer	Номер початкового тижня
end_week	integer	Номер кінцевого тижня
lesson_type	varchar	Тип заняття (Прак on-line / Лаб / 0,5 Лек та інші)
group_name	varchar	Назва групи
week	integer	Номер тижня
group_code	varchar	Унікальний код групи
group_generic_id	varchar	Ідентифікатор групи в "GroupsGeneric"
timestamp	timestamp	Дата процедури резервації

Таблиця 2.7 – Опис таблиці ScheduleExams

Поле	Тип даних	Опис
id	uuid	Універсальний унікальний ідентифікатор заняття
date	timestamp	Дата екзамену

Кінець таблиці 2.7

lecturer_name	varchar	Ім'я викладача
lecturer_id	varchar	Ідентифікатор викладача у форматі GUID
room	varchar	Аудиторія (xx-x-xxx / xx-xx / xx-xxx та інші формати)
subject_short	varchar	Скорочена назва предмету
subject	varchar	Повна назва предмету
group_name	varchar	Назва групи
generic_group_id	varchar	Ідентифікатор групи в "GroupsGeneric"

Таблиця 2.8 – Опис таблиці ScheduleExams_History

Поле	Тип даних	Опис
id	uuid	Універсальний унікальний ідентифікатор заняття
date	timestamp	Дата екзамену
lecturer_name	varchar	Ім'я викладача
lecturer_id	varchar	Ідентифікатор викладача у форматі GUID
room	varchar	Аудиторія (xx-x-xxx / xx-xx / xx-xxx та інші формати)
subject_short	varchar	Скорочена назва предмету
subject	varchar	Повна назва предмету
group_name	varchar	Назва групи
generic_group_id	varchar	Ідентифікатор групи в "GroupsGeneric"
timestamp	timestamp	Дата процедури резервації

База даних серверу призначена для зберігання даних про групи, дисципліни і екзамени.

Основною таблицею є таблиця зі списком груп. Цей список має бути початково заповнений адміністратором. Модель групи складається з назви групи та назви факультету. Для кожної групи може бути унікальною лише в

рамках одного факультету. Таким чином, назва групи є унікальною в межах одного факультету, вона складається з коду в форматі XX-NN, де літери це код спеціальності в рамках факультету, і цифри відповідають за нумерацію курсу.

Таблиця з групами використовується для отримання даних з внутрішньої бази даних університету. На етапі обробки даних отриманих з внутрішньої БД заповнюється таблиця “GroupsGeneric”. Ця модель тримає доповнені дані про групу, такі як повна назва факультету і код групи. Код групи - це поле, яке відповідає ідентифікатору групи в БД університету. Таким чином, кожен запис в таблиці “GroupsGeneric” відповідає унікальній групі. Ця таблиця необхідна для групування груп і дисциплін та екзаменів.

Таблиця “ScheduledLessons” зберігає інформацію про список дисциплін. Для кожної дисципліни зберігаються всі дані відповідно до бази даних університету. Це необхідно як для організації розкладу, так і для підтримки сумісності з БД університету, для цього таблиця має такі поля як “group_code”, “lecturer_id”. За аналогією працює таблиця з “ScheduledExams”.

Для процедури синхронізації бази є вимога щодо можливості відновлення попереднього стану розкладу, для цього були додані таблиці “ScheduledLessons_History” та “ScheduledExams_History”. Ці таблиці є копіями вище згаданих таблиць, але з додатковим полем “timestamp”. Поле “timestamp” - це час виконання процедури резервації даних. Для резервації даних з таблиць “ScheduledLessons” та “ScheduledExams” використовується збережена процедура, яка переносить всі наявні дані у резервні таблиці і видаляє дані з основних таблиць.

Такий підхід був обраний, як стратегія обробки помилок в процесі синхронізації. Процес синхронізації може займати досить багато часу, від кількох годин, і в зв'язку з можливими перебоями в функціонуванні інфраструктури, можуть виникати помилки, які вплинуть на валідність даних. Для того щоб уникнути використання великих транзакцій в базі даних, було прийнято рішення резервувати дані з ручною можливістю відновлення даних з останньої резервної копії.

Операція резервування на рівні бази даних включає створення копій існуючих таблиць, якщо вони ще не існують, копіювання даних з наявних таблиць з та встановлення відбитку часу, коли ця операція була виконана. Для процесу резервування даних було обрано метод збереженої процедури в базі даних.

Для системи синхронізації було обрано архітектуру “worker”. Тобто додаток має виконувати одну функцію і зупинятись після завешення операції. Таким чином, цей додаток має відповідний lifecycle – ініціалізація, збереження наявних даних у резервні таблиці, завантаження списку груп з бази даних, отримання даних для групи, збереження даних в базі даних, вихід з додатку. Цей процес зображено на рисунку 2.5.

Рисунок 2.5 - Життєвий цикл сервісу синхронізації

При роботі з АПІ розкладу було виявлено, що при великій кількості запитів можуть виникати помилки на окремих запитах, що можу вплинути на коректність даних в результаті. Тому при виборі з наявних алгоритмів потрібно враховувати необхідність подвійної обробки даних для однієї групи. Тому використання звичайного списку та циклу не підходить під цю вимогу,

тому що при ітерації по масиву C# неможливо додати елементи в масив, що ітерується. Тому далі можемо розглядати Стек і Чергу. Якщо при виконанні запиту для певної групи відбулася помилка, то найбільш доцільним буде спробувати отримати ці дані через певний проміжок часу. У випадку зі стеком, новий об’єкт буде доданий на початок, тому у випадку проблеми з запитом на певну групу, процес не просунеться далі. Тому найбільш доцільним є використання черги, бо елемент буде доданий в кінець, і таким чином запит повториться через певний час.

Для серверної частини інтерфейсу адміністратора була використана архітектура серверного API та клієнтського додатку. Для серверної частини було використано мову програмування C#, переважно через вже наявну кодову базу з використанням цих технологій, що значно спрощує розробку, так як більшість логіки доступу до даних є схожою для обох серверних додатків.

API побудованою за архітектурою Model-View-Controller. Для кожної сутності використовується окремий контролер, який обробляє лише данні характерні для окремої сутності. Структуру класу контролерів зображено на рисунку 2.6.

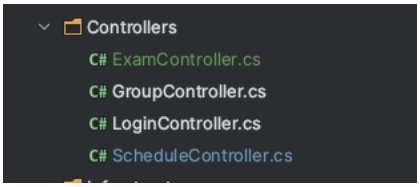


Рисунок 2.6 – Структура серверного API

Клас контролера виконує запити до класу ScheduleRepository, який відповідає за формування запитів до бази даних.

Для підтримки процесу аутентифікації розроблено middleware, клас який використовується фільтрування запитів до контролерів. У даному випадку даний клас виконує перевірку авторизації користувача, у разі непроходження перевірки, запит завершиться зі HTPP статусом 401.

Для запитів виконується перевірка вхідних даних, це необхідно для забезпечення безпеки додатку.

Для запитів на отримання розкладу екзаменів та розкладу занять, виконується перевірка на пустий ввід, а також на присутність ідентифікатора групи в базі даних. Приклад коду перевірки зображено на рисунку 2.7.

```
[HttpGet]
[Route(template: "")]
public async Task<IActionResult> GetExamsByGroupId([FromQuery] string groupId)
{
    if (string.IsNullOrWhiteSpace(groupId))
    {
        return BadRequest(error: $"Required parameter `{nameof(groupId)}` missing");
    }
```

Рисунок 2.7 – Валідація вхідних даних для класу контролера.

Для запитів на оновлення розкладу занять створено один метод, який відповідає за оновлення списку розкладів для певної групи. Переваги вибору такого підходу це наявність єдиної логіки для валідації будь-яких змін в списку занять, а також можливість використання того ж інтерфейсу для додавання, видалення, редагування списку занять.

Список умов для перевірки:

– Коректний ідентифікатор групи
– Група існує в базі даних
– Для кожного заняття день тижня коректний
– Для кожного заняття порядковий номер заняття коректний
– Номер тижня вказаний правильно.
– Після проходження перевірки даних для оновлення, список занять для обраної групи видаляється і додаються нові заняття з зв’язком до об’єкту групи.

Для клієнтської частини було обрано Next.js через зручність використання та інтеграції. Цей фреймворк відразу включає велику кількість інструментів, такі як роутинг, SSR.

Next.js використовує бібліотеку React для клієнтської частини додатку. Основною перегою цього фреймворку є Server Side Rendering, тобто технологія, яка дозволяє виконувати процедуру рендерингу на стороні сервера, тоді як клієнт отримує заповнений html документ, на відмінну від типового Single Page Application.

Для цього додатку це є перевагою, хоча й не є критичною особливістю, а лише спрощує розробку додатку.

Next.js також підтримує API Routes, які дозволяють розробникам легко створювати логіку, яка виконується на стороні серверу в межах своїх Next.js додатків. Це означає, що ви можете легко інтегрувати функціонал серверної частини безпосередньо у свій додаток, що відкриває широкий спектр можливостей для взаємодії з даними.

Для авторизації в інтерфейсі адміністратора було обрано аутентифікацію за методом JWT. Імплементація JWT аутентифікації є досить базовим функціоналом для .NET додатку, проте це підвищує безпекові вимоги до додатку. Таким чином у разі розробки власної системи генерації JWT і зберігання даних користувача, виникають безпекові вимоги щодо зберігання приватних ключів для генерації JWT, збереження даних користувача, в тому числі пароль, а також додаткові засоби безпеки, генерації JWT токенів.

Для генерації JWT токенів було використано ПЗ “Електронний Кампус”. Таким чином, під час здійснення процедури логіну користувач надсилає форму з логіном і паролем в API “Електронний Кампус”, після чого отримую JWT токен. Запит для направляється на серверну частину клієнтського додатку, після чого направляється в API “Електронний Кампус”, і після цього відповідь обробляється на клієнтській частині додатку. У цього підходу є ряд переваг – екранування запиту на аутентифікацію, зменшений ризик перехоплення логіну та паролю користувача. Для підвищення безпеки додатку, виконується перевірка токена на рівні клієнтського додатку та на рівні API.

Веб-додаток також потребує певної стилізації для покращення використання. Так як для даного ПЗ не стоїть вимога реалізації певного дизайну або слідування стандартам дизайну в рамках університету, створено мінімальну стилізацію для забезпечення функціональності веб-додатку. Для цього було використано бібліотеку стилів tailwind.css.

На стороні серверної частини додатку необхідно виконати перевірку публічного ключа токена на відповідність вказаним сертифікатами для “Електронний Кампус”. У разі відповідності, з публічної частини ключа отримуються ідентифікатор користувача, за яким можна перевірити на наявність прав адміністратора системи.

JWT токен необхідної для будь-який запитів до API додатку.

Для отримання доступу до бази даних університету було використано api https://reg.kpi.ua. Цей сервіс має закритий доступ, тому запити здійснюються з авторизованим ключем. З усіх наявних методів даного API було використано запит “/InteropGetGroupSchedulesAsJson?”. Цей запит повертає данні розкладу та розкладу сесії для однієї групи. Приклад результату для запиту зображено на рисунку 2.6. Відповідь повертається в форматі xml. Даний документ має одне поле типу string, в якому зберігається об’єкт розкладу для окремої групи.

Об’єкт розкладу зберігається в форматі подібному до JSON, в якому замість символу “”” використовується символ “’”. Тому для обробки цих даних стояла задача нормалізації даних до формату JSON. Для обробки відповіді в такому форматі був розроблений наступний алгоритм:

1. для початкової обробки відповіді використовується парсер XML;
2. виконується запит на парсер для отримання полів за типом “string”;
3. для обраного об’єкта обирається перший child-елементів, який очікується як тип string;

4. Для обробки цього об’єкта був створений клас SingleQuoteJsonReader, який наслідує вбудований JsonTextReader, та додає можливість обробки документа з символом “””.

5. Виконується запит до класу SingleQuoteJsonReader, після чого в результаті ми отримуємо необхідний об’єкт розкладу.

```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://np/">{'InteropScheduleList':[{'Код':'37627639-f487-4548-a27b-6f939e0351a5','ТипСеместру':'Весняний',
'Факультет':'Хіміко-технологічний                                    ','Курс':'1','ScheduleItems':[
{'Код':'01216941-b94b-4788-a320-931d9c8f9b48','ДисциплінаНазва':'Філософські засади наукової діяльності 2. Філософська гносеологія та
епістемологія','ДисциплінаСкороченаНазва':'','ДисциплінаКодРНП':'332034405570ed634471bb5fc1eec7bd4843','День':'Чт','Пара':'3',
'Тиждень':'1','Викладач':'Щирица Тетяна Володимирівна','ВикладачКод':'fa1219ea-aa1c-41f3-a839-a0c07b9ac16f','Аудиторія':null,
'ТижденьПочатку':null,'ТижденьКінця':null,'ТипЗаняття':'Прак on-line','Група':'ХП-21ф'},{'Код':'9d1ccdf5-1289-4f49-a99e-52b4a09cdc03',
'ДисциплінаНазва':'Філософські засади наукової діяльності 2. Філософська гносеологія та епістемологія','ДисциплінаСкороченаНазва':'',
'ДисциплінаКодРНП':'332034405570ed634471bb5fc1eec7bd4843','День':'Чт','Пара':'4','Тиждень':'1','Викладач':'Препотенська Марина Петрівна',
'ВикладачКод':'119f3cad-7d6e-430c-aeef-92351c0e1bd9','Аудиторія':null,'ТижденьПочатку':null,'ТижденьКінця':null,'ТипЗаняття':'Лек
on-line','Група':'ХП-21ф'},{'Код':'c3a11864-3002-43cf-b4e0-5a16660cd70f','ДисциплінаНазва':'Філософські засади наукової діяльності 2.
Філософська гносеологія та епістемологія','ДисциплінаСкороченаНазва':'','ДисциплінаКодРНП':'332034405570ed634471bb5fc1eec7bd4843',
'День':'Чт','Пара':'5','Тиждень':'2','Викладач':'Щирица Тетяна Володимирівна','ВикладачКод':'fa1219ea-aa1c-41f3-a839-a0c07b9ac16f',
'Аудиторія':null,'ТижденьПочатку':null,'ТижденьКінця':null,'ТипЗаняття':'Прак on-line','Група':'ХП-21ф'},
{'Код':'b2b05afd-e734-44b6-88e7-fd547bc67d56','ДисциплінаНазва':'Іноземна мова для наукової діяльності. Частина 2. Наукова комунікація',
'ДисциплінаСкороченаНазва':'','ДисциплінаКодРНП':'28306927ad5be7944ec381853ec84c052efc','День':'Чт','Пара':'1','Тиждень':'1',
'Викладач':'Гудманян Артур Грантович','ВикладачКод':'2b539e54-51b0-4793-8431-a5ef93342eee','Аудиторія':null,'ТижденьПочатку':null,
'ТижденьКінця':null,'ТипЗаняття':'Прак on-line','Група':'ХП-21ф'},{'Код':'39ab25c7-a99f-4f21-aa7f-dd817a883d1f',
'ДисциплінаНазва':'Іноземна мова для наукової діяльності. Частина 2. Наукова комунікація','ДисциплінаСкороченаНазва':'',
'ДисциплінаКодРНП':'28306927ad5be7944ec381853ec84c052efc','День':'Чт','Пара':'1','Тиждень':'2','Викладач':'Гудманян Артур Грантович',
'ВикладачКод':'2b539e54-51b0-4793-8431-a5ef93342eee','Аудиторія':null,'ТижденьПочатку':null,'ТижденьКінця':null,'ТипЗаняття':'Прак
on-line','Група':'ХП-21ф'}],'SessionScheduleItems':[{'Код':'1481a576-c5dc-4393-8cb6-53765095b380','Дата':"\/Date(1686816000000+0300)\/",
'Викладач':'Гудманян Артур Грантович','ВикладачКод':'2b539e54-51b0-4793-8431-a5ef93342eee','Аудиторія':'04-ВХ',
'ДисциплінаНазва':'Іноземна мова для наукової діяльності. Частина 2. Наукова комунікація','ДисциплінаСкороченаНазва':'',
'ДисциплінаКодРНП':'28306927ad5be7944ec381853ec84c052efc','Група':'ХП-21ф'},{'Код':'d230f109-3c1b-415b-b98c-e8b8630364d1','Дата':"\/Date
(1687154400000+0300)\/","Викладач':'Препотенська Марина Петрівна','ВикладачКод':'119f3cad-7d6e-430c-aeef-92351c0e1bd9',
'Аудиторія':'04-ВХ','ДисциплінаНазва':'Філософські засади наукової діяльності 2. Філософська гносеологія та епістемологія',
'ДисциплінаСкороченаНазва':'','ДисциплінаКодРНП':'332034405570ed634471bb5fc1eec7bd4843','Група':'ХП-21ф'}]}]}</string>
```


Рисунок 2.8 – Приклад відповіді на запит “/InteropGetGroupSchedulesAsJson?”.

При роботі з даним API, потрібно обробляти різні випадки відповіді. Можливі кілька варіантів відповіді з даного API:

1. запит виконано успішно, для групи наявний розклад занять;
2. запит виконано успішно, відсутній розклад занять для групи;
3. запит виконано успішно, відсутні дані розкладу екзаменів для групи;
4. запит виконано успішно, відсутні дані групи;
5. запит виконано успішно, відповідь в неправильному форматі;
6. сервер не відповів на запит.

Випадки 1 – 4 обробляються відповідно до отриманих даних, тобто виконуються операції для оновлення розкладу групи.

У випадку 5 подальша обробка запиту для цієї групи неможлива. За документацією, наданою інженерним відділом, відповідальним за це API,

така відповідь можлива у випадку відсутності даних для поточної групи, тому група при

Випадок 6 може виникнути, якщо в роботі датацентру виникли помилки або API обробляє надмірну кількість запитів.

У випадку надмірної кількості запитів, логічним рішенням було б спробувати провести запит через певну кількість часу. Під час роботи з даним ПЗ експериментальним шляхом було визначено, що при обробці великої кількості запитів певні запити можуть викликати помилку в системі. Тому кожен, наприклад, 10 запит обробляється неправильно, що призводить до виникнення втрати даних по завершенню процесу синхронізації.

Тому для обробки цього випадку передбачена затримка між запитами на розклад груп, щоб не спричиняти надмірну кількість запитів. Це значення може змінюватись системним адміністратором, в залежності від існуючої потреби зі зменшення кількості запитів та швидкодії системи.

Для випадку, коли помилки спричинені перебоями в роботі датацентру, для обробки цього випадку необхідне втручання системного адміністратор для налаштування процесу синхронізації.

2.4 Аналіз безпеки даних

Розроблений додаток не передбачає збереження особистих даних кінцевих користувачів або іншої вразливої персональної інформації.

Для підтримки аутентифікації користувачів з правами адміністратора зберігається лише ідентифікатор користувача в системі Електронний Кампус. Система не потребує доступу для секретних ключів для генерації JWT токена.

Передача даних по мережі відбувається з підтримкою SSL сертифікатів.

Доступ до даних захищений на багатьох рівнях :

– Аутентифікація користувача на рівні інтерфейсу адміністратора. Для будь-яких запитів необхідно зберігати JWT токен;

– Аутентифікація на рівні СУБД. Для доступу до БД використовується захищений протокол та доступ за ім’ям та паролем користувача. В додатку передбачений механізм збереження приватних ключів в змінних оточення;

У випадку, якщо зловмисник отримає доступ до інтерфейсу адміністратора, він зможе редагувати данні в таблицях занять та екзаменів, що може вплинути на працездатність системи. Проте зловмисник не зможе отримати доступ до жодної не публічної інформації.

Якщо зловмисник отримає доступ до СУБД системи, то він зможе пошкодити записи в БД, проте при процедурі синхронізації запити будуть автоматично оновленні. Але у разі пошкодження записів у резервних таблицях, ці дані неможливо буде відновити. Тому данні доступу до СУБД мають зберігатися тільки на сервері і у відповідному ПЗ для менеджменту вразливої інформації.

Висновки по розділу

У цьому розділі було розглянуто процес моделювання та конструювання програмного забезпечення. За допомогою діаграми сутностей було визначено основні сутності даних – Група, Заняття, Екзамен, а також зв’язки між ними. З урахуванням цієї інформацію було спроектовано модель роботи з даними.

Було розглянуто архітектуру ПЗ. ПЗ ділиться на модулі, кожен з яких відповідає за функціонал окрему частину функціоналу та бізнес логіки, а саме: загальна логіка роботи з СУБД, інтеграція з сервісами Єдиної Інформаційної Системи, серверна та клієнтська частини інтерфейсу адміністратора.

Виконано дослідження інтеграції ПЗ з системами ЄІС, а саме синхронізацію даних розкладів занять та бізнес логіку процесу синхронізації.

Розглянуто механізми для роботи системи з урахуванням можливих помилок на рівні інтеграції з ЄІС.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Якісь програмного тестування визначається набором характеристик, які мають відповідати встановленим вимогам продукту.

Для оцінки якості ПЗ було використано сервіс codescene.io.[6] Цей сервіс оцінює програмний код за наступними метриками:

– вразливості в коді;
– підтримка коду;
– використання “шкідливих” патернів розробки;
– зв’язність коду;
– використання ресурсів;
– обробка помилок;
– зрозумілість коду;
– дублювання коду.

Результат сканування зображено на рисунку 3.1

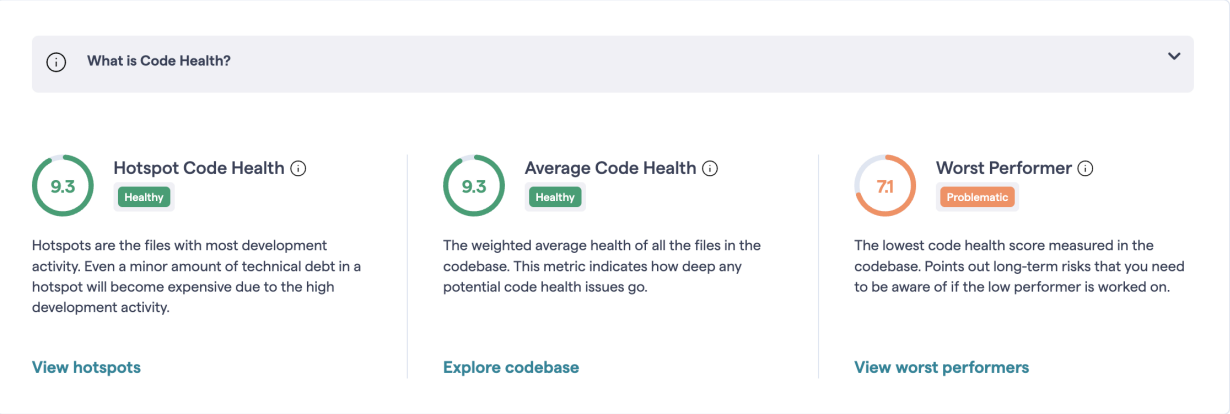


Рисунок 3.1 – Результат сканування коду

Як можна зрозуміти з результатів перевірки, код даного ПЗ задовольняє вимоги за цими показниками.

Одним з способів перевірка відповідності вимогам є тестування. Для тестування будуть використані варіанти використання описані в розділі 1.4 і за результатами тестів виконана перевірка функціональних вимог.

3.2 Опис процесів тестування

В таблицях 3.1 – 3.13 розглянуто серію мануальних тестів.

Таблиця 3.1 – Тест 1.1

Тест	Початкова синхронізація даних розкладів
Модуль	Модуль синхронізації даних (“Campus.Schedule.Parser”)
Номер тесту	1.1
Початковий стан системи	Створена база даних, відсутні дані розкладів. Створена вся необхідна конфігурація для синхронізації даних
Вхідні данні	-
Опис проведення тесту	У командному рядку вводиться назва програми.
Очікуваний результат	В базі даних заповнені таблиці “GroupsGeneric”, “ScheduleLessons”, “ScheduleExams”. Дані відповідають даним на наявному розкладі ЄІС.
Фактичний результат	В базі даних заповнені таблиці “GroupsGeneric”, “ScheduleLessons”, “ScheduleExams”. Дані відповідають даним на наявному розкладі ЄІС.

Таблиця 3.2 – Тест 1.2

Тест	Автоматизований процес синхронізації розкладу занять
Модуль	Модуль синхронізації даних (“Campus.Schedule.Parser”)
Номер тесту	1.2

Кінець таблиці 3.2

Початковий стан системи	Створена база даних, заповнені дані розкладів. Налаштований автоматичний запуск програми за інтервалом часу.
Вхідні данні	-
Опис проведення тесту	Процес запускається автоматично відповідно до розкладу запуску програми
Очікуваний результат	В базі даних заповнені таблиці “GroupsGeneric”, “ScheduledLessons”, “ScheduleExams”. Дані відповідають даним на наявному розкладі ЄІС. Заповнені таблиці “ScheduledLessons_History”, “ScheduledExams_History”
Фактичний результат	При перевірці у вказаний період часу дані відповідають даним у наявному розкладі ЄІС. В таблицях “ScheduledLessons_History”, “ScheduledExams_History” збережені дані з попередніх ітерацій системи.

Таблиця 3.3 – Тест 1.3

Тест	Аутентифікація та авторизація користувача в інтерфейсі адміністратора
Модуль	Модуль синхронізації даних (Campus.Schedule.Dashboard)
Номер тесту	1.3
Початковий стан системи	Користувач заходить на адресу сторінки інтерфейсу адміністратора
Вхідні данні	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться електрона пошта та пароль. Користувач натискає кнопку “Login”.

Кінець таблиці 3.3

Очікуваний результат	Користувач успішно виконав вхід, відкрилась головна сторінка інтерфейсу адміністратора
Фактичний результат	Користувач успішно виконав вхід, відкрилась головна сторінка інтерфейсу адміністратора

Таблиця 3.4 – Тест 1.4

Тест	Відображення списку груп
Модуль	Інтерфейс адміністратора (Campus.Schedule.Dashboard)
Номер тесту	1.4
Початковий стан системи	Користувач авторизований
Вхідні данні	-
Опис проведення тесту	Користувач переходить на сторінку Groups
Очікуваний результат	Перед користувачем відображається повний список груп
Фактичний результат	Перед користувачем відображається повний список груп, данні для яких синхронізовані в даний момент

Таблиця 3.5 – Тест 1.5

Тест	Додавання групи в базу даних
Модуль	Інтерфейс адміністратора (Campus.Schedule.Dashboard)
Номер тесту	1.5
Початковий стан системи	Користувач авторизований
Вхідні данні	-

Кінець таблиці 3.5

Опис проведення тесту	Перед користувачем відображається повний список груп. У відповіді поля користувач вводить назву групи, абревіатуру назви факультету, повну назву факультету, натискає кнопку “Create group”.
Очікуваний результат	Група додалась до списку з існуючими групами.
Фактичний результат	Група додалась до списку з існуючими групами. Після перезавантаження сторінки група залишається в списку груп.

Таблиця 3.6 – Тест 1.6

Тест	Переглянути список занять для групи
Модуль	Інтерфейс адміністратора (Campus.Schedule.Dashboard)
Номер тесту	1.6
Початковий стан системи	Користувач авторизований
Вхідні данні	-
Опис проведення тесту	Користувач переходить на сторінку Subjects. Зі списку груп користувач обирає необхідну групу.
Очікуваний результат	Перед користувачем відкривається список занять для обраної групи.
Фактичний результат	Перед користувачем відкривається список занять для обраної групи.

Таблиця 3.7 – Тест 1.7

Тест	Додати заняття до списку занять для групи
Модуль	Інтерфейс адміністратора (Campus.Schedule.Dashboard)
Номер тесту	1.7

Кінець таблиці 3.7

Початковий стан системи	Користувач авторизований
Вхідні данні	Код групи, номер тижня для нового заняття, день тижня для нового заняття, номер заняття на день, назва заняття, ім’я лектора, номер аудиторії, тип заняття.
Опис проведення тесту	Користувач переходить на сторінку Subjects. Зі списку груп користувач обирає необхідну групу. В форму AddSubject користувач заповнює відповідні дані для нового заняття, натискає кнопку “Add row” та кнопку “Update Schedule”
Очікуваний результат	Заняття додано до розкладу групи
Фактичний результат	Заняття додано до розкладу групи.

Таблиця 3.8– Тест 1.8

Тест	Редагувати заняття до списку занять для групи
Модуль	Інтерфейс адміністратора (Campus.Schedule.Dashboard)
Номер тесту	1.8
Початковий стан системи	Користувач авторизований
Вхідні данні	Код групи, номер тижня для нового заняття, день тижня для нового заняття, номер заняття на день, назва заняття, ім’я лектора, номер аудиторії, тип заняття.
Опис проведення тесту	Користувач переходить на сторінку Subjects. Зі списку груп користувач обирає необхідну групу. Зі списку занять обирає одне для редагування. В формі заповнює оновлену інформацію. Натискає кнопку “Update Row” та “Update Schedule”.

Кінець таблиці 3.8

Очікуваний результат	Заняття оновлено для розкладу групи.
Фактичний результат	Заняття оновлено для розкладу групи.

Таблиця 3.9 – Тест 1.9

Тест	Видалити заняття зі списку занять для групи
Модуль	Інтерфейс адміністратора (Campus.Schedule.Dashboard)
Номер тесту	1.9
Початковий стан системи	Користувач авторизований
Вхідні данні	Код групи.
Опис проведення тесту	Користувач переходить на сторінку Subjects. Зі списку груп користувач обирає необхідну групу. Зі списку занять обирає одне для видалення. Натискає кнопку “Remove”
Очікуваний результат	Заняття видалено для розкладу групи.
Фактичний результат	Заняття видалено для розкладу групи.

Таблиця 3.10 – Тест 1.10

Тест	Переглянути список екзаменів для групи
Модуль	Інтерфейс адміністратора (Campus.Schedule.Dashboard)
Номер тесту	1.10
Початковий стан системи	Користувач авторизований
Вхідні данні	Код групи.

Кінець таблиці 3.10

Опис проведення тесту	Користувач переходить на сторінку Exams. Зі списку груп користувач обирає необхідну групу.
Очікуваний результат	Перед користувачем відкривається список екзаменів для обраної групи.
Фактичний результат	Перед користувачем відкривається список екзаменів для обраної групи.

Таблиця 3.11 – Тест 1.11

Тест	Додати екзамен до списку занять для групи
Модуль	Інтерфейс адміністратора (Campus.Schedule.Dashboard)
Номер тесту	1.11
Початковий стан системи	Користувач авторизований
Вхідні данні	Код групи, дата нового екзамену, назва нового екзамену, ім’я лектора, номер аудиторії.
Опис проведення тесту	Користувач переходить на сторінку “Exams”. Зі списку груп користувач обирає необхідну групу. В форму “AddExam” користувач заповнює відповідні дані для нового заняття, натискає кнопку “Add row” та кнопку “Update Schedule”
Очікуваний результат	Екзамен додано до розкладу групи
Фактичний результат	Екзамен додано до розкладу групи

Таблиця 3.12 – Тест 1.12

Тест	Редагувати екзамен до списку занять для групи

Кінець таблиці 3.12

Модуль	Інтерфейс адміністратора (Campus.Schedule.Dashboard)
Номер тесту	1.12
Початковий стан системи	Користувач авторизований
Вхідні данні	Код групи, дата нового екзамену, назва нового екзамену, ім’я лектора, номер аудиторії
Опис проведення тесту	Користувач переходить на сторінку “Exams”. Зі списку груп користувач обирає необхідну групу. Зі списку екзаменів обирає одне для редагування. В формі заповнює оновлену інформацію. Натискає кнопку “Update Row” та “Update Schedule”.
Очікуваний результат	Екзамен оновлено для розкладу групи.
Фактичний результат	Екзамен оновлено для розкладу групи.

Таблиця 3.13– Тест 1.13

Тест	Видалити заняття зі списку занять для групи
Модуль	Інтерфейс адміністратора (Campus.Schedule.Dashboard)
Номер тесту	1.13
Початковий стан системи	Користувач авторизований
Вхідні данні	Код групи.
Опис проведення тесту	Користувач переходить на сторінку “Exams”. Зі списку груп користувач обирає необхідну групу. Зі списку екзаменів обирає один для видалення. Натискає кнопку “Remove”

Кінець таблиці 3.13

Очікуваний результат	Екзамен видалено для розкладу групи.
Фактичний результат	Екзамен видалено для розкладу групи.

3.3 Опис контрольного прикладу

В процесі підтримки даного ПЗ важливим є процес перевірки системи на коректність конфігурації. Для перевірки коректності роботи розгорнутої системи можна провести наступний тест:

1. В таблицю “Groups” потрібно додати назви груп, для яких необхідно завантажити розклади;
2. В командному рядку необхідно виконати команду для запуску проекту “Campus.Schedule.Parser”. Після цього у виводі за допомогою системи логування можна побачити поточний статус процесу;
3. Після завершення виконання програми, таблиці “ScheduledLessons” та “ScheduledExams” мають бути заповнені.
4. Виконати вхід під іменем користувача з правами адміністратора системи розклад в інтерфейс адміністратора.
5. На сторінці “Groups” можна побачити список груп, який має відповідати списку, зконфігурованому у пункті 1.
6. На сторінці “Subjects” для кожної з обраних груп має бути присутній розклад занять.

Висновки по розділу

В цьому розділі виконано аналіз програмного забезпечення. Проведено мануальне тестування на відповідність функціональним вимогам. Розроблено

контрольний приклад для перевірки працездатності розгорнутого програмного забезпечення з холодного старту.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1Розгортання програмного забезпечення

Клієнтську і серверну частину додатку було вирішено розгорнути на приватному віртуальному сервері (VPS). Для розгортання було використано програмне забезпечення Docker та сервіс GitHub Actions забезпечення постійної інтеграції.

Розгортання починається коли новий код додається в репозиторій додатку. Тоді у середовищі GitHub Actions на базі вказаного у відповідному файлі конфігурації Dockerfile створюється новий docker image.

Для розгортання системи на сервері користувач має підключитись до сервер за допомогою протоколу SSH.

Додаток потребує наявність бази даних, для цього був використаний докер контейнер Postgresql. Для розгортання інтерфейсу адміністратора необхідно за допомогою вказаних команд запустити відповідні докер контейнери, вказавши данні для підключення до бази даних як змінні оточення.

Для розгортання системи синхронізації було використано вказану команду для запуску докер контейнера та утиліту крон, в конфігурації для запуску контейнера щодня вночі.

4.2 Підтримка програмного забезпечення

У випадку внесення змін в кодову базу, за допомогою GitHub Actions буде опубліковано нову версію docker image для кожного модуля системи.

Оновлення версії розгорнутої системи передбачено системним адміністратором системи.

Висновки по розділу

В цьому розділі було розглянуто впровадження та супровід програмного забезпечення. Було описано кроки необхідні для налаштування системи і механізми підтримки програмного забезпечення.

ВИСНОВКИ

В результаті виконання дипломного проекту було спроектовано програмне забезпечення для підготовки та роботи з розкладом занять. Метою ПЗ була розробка програмного рішення для заміни наявної системи розкладу, яке відповідало б сучасним вимогам до веб-додатків.

ПЗ передбачає завантаження розкладів занять з сервісів ЄІС, управління процесом синхронізації а також засоби для внесення і редагування розкладів занять. На даний момент це ПЗ інтегровано та успішно використовується в системах університету. Це програмне рішення є частиною оновленої інфраструктури ЄІС, що дозволяє подальший розвиток, підтримку та інтеграцію системи розкладу занять. Система пристосована для інтеграції в ЄІС та в систему “Електронний Кампус”, що спрощує подальшу інтеграції з іншими сервісами університету.

В першому розділі були розглянути вимоги до програмного забезпечення, розглянута предметна область, варіанти використання та функціональні вимоги до проекту. В результаті було сформовано вимоги для цього ПЗ:

1. Інтеграція нової системи розкладу з Єдиною Інформаційною Системою університету та виведення з експлуатації попередньої системи розкладу занять.

2. У зв’язку з проблемами, спричиненими знеструмленням датацентру, забезпечення резервації даних і можливість перенесення системи в хмарне середовище.

В другому розділі було розглянуто бізнес процеси додатку, моделі даних які використовуються. Було побудовано архітектуру ПЗ, яка складається з чотирьох модулів: база даних, система синхронізації, серверна та клієнтська частини інтерфейсу адміністратора.

Для бази даних використана СУБД PostgreSQL. В базі даних передбачені таблиці для збереження даних груп, занять, екзаменів, а також допоміжні

таблиці для збереження даних для доступу адміністраторів, необхідна конфігурація для налаштування системи синхронізації.

Як основна мова програмування використана мова C#, що дозволяє використання потужних фреймворків та бібліотек для роботи з данними та мережею.

Для клієнтської частини інтерфейсу адміністратора використаний фреймворк Next.js, який дозволяє використовувати потужну вбудовану бібліотеку для побудови клієнтських додатків.

В третьому розділі розглянуто процес аналізу якості системи, проведено мануальне тестування системи. Було розроблено контрольний тест, який дозволяє пересвідчитись в функціональності системи для холодного старту.

В четвертому розділі розглянуто процес розгортання та підтримки системи. Це вимагає присутності системного адміністратора.

Перевагами цього програмного забезпечення є:

– використання сучасної технологічної бази;

– інтеграція з системою ЄІС;

– інтеграція з системою “Електронний Кампус”;

– простота в подальшої підтримки та інтеграції ПЗ.

– Вперше:

– пристосовано систему для функціонування в умовах перебоїв в роботі датацентру університету;

– розроблено рішення, яке дозволяє розгортання системи розкладу занять в хмарних датацентрах;

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) GENERAL ARCHITECTURE OF THE SYSTEM “ELECTRONIC CAMPUS” [Електронний ресурс]- https://www.sworld.com.ua/konfer31/813.pdf

2) Підсистема обліку електронного методичного забезпечення навчального процесу інститутів і факультетів НТУУ “КПІ” [Електронний ресурс] - https://www.academia.edu/89210076/Підсистема_обліку_електронного_методичного_забезпечення_навчального_процесу_інститутів_і_факультетів_НТУУ_КПІ_

3) Moodle reference [Електронний ресурс] https://docs.moodle.org/402/en/Main_page

4) Web Programming ASP.NET core [Електронний Ресурс] - https://www.halvorsen.blog/documents/programming/csharp/textbook/aspnet/Web%20Programming%20-%20ASP.NET%20Core.pdf

5) Architecting Cloud Native .NET Applications for Azure [Електронний Ресурс] - https://learn.microsoft.com/en-us/dotnet/architecture/cloud-native/

6) CodeScene Reference [Електронний ресурс] - https://codescene.io/docs/guides/technical/code-health.html#

7) C# language specification [Електронний ресурс] - https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/readme.

8) Next.js Reference [Електронний ресурс] - https://nextjs.org/docs

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015400419

Дата перевірки:
02.06.2023 18:50:59 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
02.06.2023 18:51:34 EEST

ID користувача:
76913

Назва документа: **IT-92_Яцевський_ПЗ**

Кількість сторінок: **56** Кількість слів: **9056** Кількість символів: **71183** Розмір файлу: **1.23 MB** ID файлу: **1015064292**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

6.93%
Схожість

Найбільша схожість: **3.17%** з джерелом з Бібліотеки (ID файлу: **1014967607**)

3.83% Джерела з Інтернету	254	Сторінка 58
6.55% Джерела з Бібліотеки	271	Сторінка 60

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	3
Підозріле форматування	10 сторінок

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Програмне забезпечення для підготовки та роботи з розкладом занять

Текст програми

КПІ.ІТ-9230.045440.03.13

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Олександр ЯЦЕВСЬКИЙ

Київ – 2023

Файл Settings.cs

```
 namespace Campus.Schedule.Parser;

public class Settings
{
    public string ScheduleApiPassword { get; set; }
    public string ConnectionString { get; set; }
    public bool EraseDb { get; set; }
    public bool GroupUpsert { get; set; }
    public int ParsingTimeout { get; set; } = 2000;
}
```

Файл Program.cs

```
using System.Net;
using System.Threading.Tasks;
using Campus.Core.Data;
using Campus.Core.Mappers;
using Campus.Core.Repositories;
using Campus.DAL;
using Campus.DAL.Queries;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using RestSharp;

namespace Campus.Schedule.Parser;

class Program
{
    public static Task Main(string[] args) =>
        Host
            // TODO add env variables
            // TODO configure timetable repo - add mock repo first, then
connect db one
            .CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
            {
                logging.ClearProviders();
                logging.AddConsole();
            })
            .ConfigureServices((hostBuilder, services) =>
            {
                var settings = LoadSettings(hostBuilder.Configuration);

                // TODO add services here
                services
                    .AddSingleton<Settings>(settings)
                    .AddScoped<IRestClient>(_ =>
                    {
                        var client = new RestClient(options =>
                        {
                            options.ThrowOnAnyError = false;
                            options.RemoteCertificateValidationCallback +=
(_, _, _, _) => true;
                        });
```

```
ServicePointManager.ServerCertificateValidationCallback +=  (_, _, _, _) =>
true;
                        return client;
                    })
                    .AddScoped<IFetchData>(serviceProvider =>
                    {
                        var client =
serviceProvider.GetRequiredService<IRestClient>();
                        var logger =
serviceProvider.GetService<ILogger<FetchGroup>>();
                        return new FetchGroup(client,
settings.ScheduleApiPassword, logger);
                    })
                    // services.AddScoped<ITimetableRepository,
TimetableRepositoryMemory>();
                    .AddSingleton<QueryManager>(p =>
                    {
                        var logger = p.GetService<ILogger<QueryManager>>();
                        return new
QueryManager(hostBuilder.HostingEnvironment.ContentRootPath, logger);
                    })
                    .AddSingleton<ScheduleQueryManager>()
                    .AddSingleton<ScheduledMapper>()
                    .AddSingleton<LecturerMapper>()
                    .AddSingleton<GroupMapper>()
                    .AddSingleton<FacultyMapper>()
                    .AddScoped<IDatabase, Database>(p =>
                    {
                        var logger =
p.GetRequiredService<ILogger<PostgresDatabase>>();
                        return new
PostgresDatabase(settings.ConnectionString, logger);
                    })
                    .AddScoped<IScheduleRepository, ScheduleRepository>()
                    .AddScoped<IDataManager, ParserDataManager>()
                    .AddHostedService<ParserHostedService>();
            })
            .UseConsoleLifetime()
            .Build()
            .RunAsync();

    private static Settings LoadSettings(IConfiguration configuration)
    {
        var settings = new Settings();
        configuration.Bind(settings);
        settings.ConnectionString =
configuration.GetConnectionString("DefaultConnection");
        return settings;
    }
}
```

Файл ParserHostedService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using System.Xml;
using Campus.Models.Schedule;
using Campus.Schedule.Parser.Entities;
using Microsoft.Extensions.Hosting;
```

```
using Microsoft.Extensions.Logging;

namespace Campus.Schedule.Parser;

public class ParserHostedService : IHostedService
{
    private readonly ILogger _logger;
    private readonly IHostApplicationLifetime _hostApplicationLifetime;
    private readonly IDataManager _dataManager;
    private readonly IFetchData _fetchGroup;
    private readonly Settings _settings;

    public ParserHostedService(
        IDataManager dataManager,
        IFetchData fetchGroup,
        IHostApplicationLifetime hostApplicationLifetime,
        ILogger<ParserHostedService> logger,
        Settings settings)
    {
        _logger = logger;
        _dataManager = dataManager;
        _fetchGroup = fetchGroup;
        _hostApplicationLifetime = hostApplicationLifetime;
        _settings = settings;
    }

    public async Task StartAsync(CancellationToken cancellationToken)
    {
        if (_settings.GroupUpsert)
        {
            _logger.LogInformation("Start in upsert mode");
        };

        // db won't be erased in upsert mode
        if (_settings.EraseDb && !_settings.GroupUpsert)
        {
            _logger.LogInformation("Lessons and Exams erase");
            _dataManager.EraseDb();
        }
        else
        {
            _logger.LogInformation("Skip Lessons and Exams erase");
        }

        try
        {
            _logger.LogInformation("Start parsing");

            if (!_settings.GroupUpsert)
            {
                _dataManager.LoadGroups();
            }
            else
            {
                _dataManager.LoadGroupsForUpsert();
            }

            _logger.LogInformation("Loaded groups, count : " +
_dataManager.GroupsToParseCount);

            while (_dataManager.GroupsToParseCount > 0)
            {
                Thread.Sleep(_settings.ParsingTimeout);
```

```
                var groupName = _dataManager.GetNextGroupId();
                _logger.LogInformation("Parsing group : " + groupName);
                List<GroupTimetable> groupTimetables;

                try
                {
                    groupTimetables = (await
_fetchGroup.LoadData(groupName)).ToList();
                }
                catch (XmlException e)
                {
                    _logger.LogError("Trouble with loading data for : " +
groupName);
                    _dataManager.Requeue(groupName);
                    continue;
                }
                catch (System.Text.Json.JsonException e)
                {
                    _logger.LogError("Trouble with loading data for : " +
groupName + " :  " + e);
                    _dataManager.Requeue(groupName);
                    continue;
                }

                if (groupTimetables.Count == 0)
                {
                    // TODO handle that
                    continue;
                }

                foreach (var timetable in groupTimetables)
                {
                    var lessons = new List<ScheduledLesson>();
                    var exams = new List<ScheduleExam>();

                    {
                        foreach (var subject in timetable.ScheduleItems)
                        {
                            var lesson = subject.AsScheduledLesson();
                            lessons.Add(lesson);
                        }

                        foreach (var scheduleExam in timetable.Exams)
                        {
                            var exam = scheduleExam.AsScheduleExam();
                            exams.Add(exam);
                        }

                        var timetableUpdate = new TimetableUpdate()
                        {
                            GroupId = groupName,
                            Lessons = lessons,
                            Exams = exams
                        };

                        _dataManager.UpdateCurrentGroup(timetableUpdate,
timetable.Faculty, timetable.Code.ToString());
                    }
                }
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Parsing error");
```

```
        }
        finally
        {
            _hostApplicationLifetime.StopApplication();
        }
    }

    public Task StopAsync(CancellationToken cancellationToken)
    {
        return Task.CompletedTask;
    }
}
```

Файл GroupNotFoundException.cs

```
using System;

namespace Campus.Schedule.Parser;

public class GroupNotFoundException : Exception
{
}

Файл  FetchData.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Threading.Tasks;
using Campus.Schedule.Parser.Entities;
using Microsoft.Extensions.Logging;
using RestSharp;

namespace Campus.Schedule.Parser;

/// <summary>
/// Load data from API
/// </summary>
public interface IFetchData
{
    public Task<IEnumerable<GroupTimetable>> LoadData(string id);
}


public abstract class FetchEntity : IFetchData
{
    protected string Password { get; private set; }

    private readonly IRestClient _client;

    private readonly ILogger _logger;

    public FetchEntity(IRestClient client, string password, ILogger logger)
    {
        Password = password;

        _client = client;
        _logger = logger;
    }

    /// <summary>
    /// mapping of method and query params
    /// </summary>
```

```
    /// <param name="id">identifier of entity</param>
    /// <returns></returns>
    protected abstract string ConstructQuery(string id);

    public virtual async Task<IEnumerable<GroupTimetable>> LoadData(string
id)
    {
        var query = ConstructQuery(id);
        _logger.LogInformation("QueryBefore : " + query);

        var request = new RestRequest(query)
        {
            Timeout = 60000
        };

        //var res = _client.Get<GroupTimetable>(request);
        var res = await _client.GetAsync(request);

        if (res.StatusCode == HttpStatusCode.NotFound)
        {
            throw new GroupNotFoundException();
        }


        if (!res.IsSuccessful)
        {
            _logger.LogError(res.ErrorException?.Message);
        }

        var deseralizer = new ApiXmlDeserializer();
        GroupTimetableWrapper data;

        try
        {
            data = deseralizer.Deserialize<GroupTimetableWrapper>(res);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex.Message);
            return new List<GroupTimetable>();
        }

        if (data.InteropScheduleList == null)
        {
            return new List<GroupTimetable>();
        }

        return data.InteropScheduleList;
    }
}

public class FetchGroup : FetchEntity
{
    public FetchGroup(IRestClient client, string password, ILogger logger) :
base(client, password, logger)
    {
    }

    protected override string ConstructQuery(string id)
    {
        var queryUrl = "https://reg.kpi.ua/NP.Dev/WebService.asmx" +
"/InteropGetGroupSchedulesAsJson?groupName=" + id +
                       "&password=" + Password;
        return queryUrl;
```

```
    }

    public override async Task<IEnumerable<GroupTimetable>> LoadData(string
id)
    {
        var data = await base.LoadData(id);

        var timetables = new List<GroupTimetable>();

        foreach (var d in data)
        {
            if (d == null)
            {
                timetables.Add(new GroupTimetable
                {
                    ScheduleItems = new List<ScheduleItem>()
                });
            }
            else
            {
                timetables.Add(d);
            }
        }

        return timetables;
    }
}
```

Файл DataManager.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Campus.Core.Repositories;
using Campus.Models.Schedule;

namespace Campus.Schedule.Parser;

public interface IDataManager
{
    void EraseDb();
    void LoadGroups();
    void LoadGroupsForUpsert();

    string GetNextGroupId();

    public void Requeue(string groupName);

    void UpdateCurrentGroup(TimetableUpdate timetableUpdate, string faculty,
string group_code);

    int GroupsToParseCount { get; }
}

public record TimetableUpdate
{
    public string GroupId { get; set; }
    public IEnumerable<ScheduledLesson> Lessons { get; set; }
    public IEnumerable<ScheduleExam> Exams { get; set; }
}

public class ParserDataManager : IDataManager
{
```

```
    private readonly IScheduleRepository _repo;

    private Queue<Group> _groups;
    private List<Group> _groupsList;
    private Group _currentGroup;
    private List<FacultyMapping> _faculties;

    public void EraseDb()
    {
        _repo.DeleteAndReserveCurrentLessonsAndExams();
    }

    public ParserDataManager(IScheduleRepository repo)
    {
        _repo = repo;
    }

    public void LoadGroups()
    {
        var groups = _repo.GetGroups();
        _groups = new Queue<Group>(groups);
        _groupsList = groups.ToList();
        _faculties = _repo.GetFacultyMapping().ToList();
    }

    public void LoadGroupsForUpsert()
    {
        var groups = _repo.GetGroupsMissing();
        _groups = new Queue<Group>(groups);
        _groupsList = groups.ToList();
        _faculties = _repo.GetFacultyMapping().ToList();
    }

    public string GetNextGroupId()
    {
        var group = _groups.Dequeue();
        _currentGroup = group;

        return _currentGroup.Name;
    }

    public void Requeue(string groupName)
    {
        _groups.Enqueue(new Group
        {
            Id = Guid.NewGuid(),
            Name = groupName,
            Faculty = ""
        });
    }

    public void UpdateCurrentGroup(TimetableUpdate timetableUpdate, string
faculty, string group_code)
    {
        // TODO add group generic to reserve or make it skip insert
        var facultyNameMapping = _faculties.FirstOrDefault(f =>
f.FacultyLongName.Trim() == faculty.Trim());

        var facultyShortName = facultyNameMapping != null ?
facultyNameMapping.FacultyShortName : "";
        var genericGroupId = Guid.NewGuid();
        _repo.AddGroupGeneric(new GroupGeneric
        {
            Id = genericGroupId,
```

```
            GroupCode = group_code,
            GroupFacultyLong = faculty,
            GroupName = _currentGroup.Name,
            GroupFacultyShort = facultyShortName
        });

        var groupId = _currentGroup.Id;
        // TODO add group_code field
        var exams = _repo.GetExamsByGroupName(_currentGroup.Name);
        foreach (var lesson in timetableUpdate.Lessons)
        {
            _repo.AddGroupScheduleLesson(genericGroupId, lesson);
        }

        foreach (var exam in timetableUpdate.Exams)
        {
            exam.GenericGroupId = genericGroupId;
            _repo.AddGroupExam(genericGroupId, exam);
        }
    }

    public int GroupsToParseCount => _groups.Count;
}
```

Файл ApiXmlDeserializer.cs

```
using Campus.Schedule.Parser.Entities;
using RestSharp;
using RestSharp.Serializers.Xml;

namespace Campus.Schedule.Parser;

public class ApiXmlDeserializer : IXmlDeserializer
{
    public string RootElement { get; set; }
    public string Namespace { get; set; }
    public string DateFormat { get; set; }

    public T Deserialize<T>(RestResponse response)
    {
        var content = response.Content;
        var responseDeserialized = new ApiXmlResponse<T>(content);

        return responseDeserialized.Data;
    }
}
```

Файл ApiXmlResponse.cs

```
using System;
using System.IO;
using System.Text;
using System.Text.Json;
using System.Text.RegularExpressions;
using System.Xml;
using Newtonsoft.Json;
using JsonSerializer = Newtonsoft.Json.JsonSerializer;

namespace Campus.Schedule.Parser.Entities;

public record ApiXmlResponse<T>
{
    public T Data { get; }
```

```
    public ApiXmlResponse(string xml)
    {
        var xmlDoc = new XmlDocument();
        xmlDoc.LoadXml(xml);

        var json = xmlDoc.GetElementsByTagName("string")[0].InnerXml;

        var sr = new StringReader(json);
        var reader = new SingleQuoteJsonReader(sr);
        var serializer = new JsonSerializer();

        Data = serializer.Deserialize<T>(reader);
    }
}
```

Файл GroupTimetable.cs

```
using System;
using System.Collections.Generic;
using System.Text.Json.Serialization;
using Campus.Models.Schedule;
using Newtonsoft.Json;

namespace Campus.Schedule.Parser.Entities;

public record TimetableWrapper<T>
{
    public IEnumerable<T> InteropScheduleList { get; set; }
}

public record GroupTimetableWrapper
{
    public GroupTimetable[] InteropScheduleList { get; set; }
}

public record GroupTimetable
{
    [JsonPropertyName("Код")]
    [JsonProperty("Код")]
    public Guid Code { get; set; }

    [JsonPropertyName("ТипСеместру")]
    [JsonProperty("ТипСеместру")]
    public string SemesterType { get; set; }

    [JsonPropertyName("Факультет")]
    [JsonProperty("Факультет")]
    public string Faculty { get; set; }

    [JsonPropertyName("Курс")]
    [JsonProperty("Курс")]
    public string Course { get; set; }

    [JsonPropertyName("ScheduleItems")]
    [JsonProperty("ScheduleItems")]
    public List<ScheduleItem> ScheduleItems { get; set; }

    [JsonPropertyName("SessionScheduleItems")]
    [JsonProperty("SessionScheduleItems")]
    public List<Exam> Exams { get; set; }
}
```

```
public record Exam
{
    [JsonPropertyName("Код")]
    [JsonProperty("Код")]
    public Guid Code { get; set; }

    [JsonPropertyName("Дата")]
    [JsonProperty("Дата")]
    public string  Date { get; set; }
    [JsonPropertyName("Викладач")]
    [JsonProperty("Викладач")]
    public string Lecturer { get; set; }
    [JsonPropertyName("ВикладачКод")]
    [JsonProperty("ВикладачКод")]
    public string LecturerId { get; set; }
    [JsonPropertyName("Аудиторія")]
    [JsonProperty("Аудиторія")]
    public string Auditory { get; set; }
    [JsonPropertyName("ДисциплінаНазва")]
    [JsonProperty("ДисциплінаНазва")]
    public string SubjectName { get; set; }
    [JsonPropertyName("ДисциплінаСкороченаНазва")]
    [JsonProperty("ДисциплінаСкороченаНазва")]
    public string SubjectShortName { get; set; }
    [JsonPropertyName("ДисциплінаКодРНП")]
    [JsonProperty("ДисциплінаКодРНП")]
    public string SubjectCodeRNP { get; set; }
    [JsonPropertyName("Група")]
    [JsonProperty("Група")]
    public string Group { get; set; }


    public ScheduleExam AsScheduleExam()
    {
        // 'Дата': "/Date(1641384000000+0200)/"
        var timestamp = Date.Split("(")[1].Split(")")[0].Split("+")[0];
        var timestampDouble = double.Parse(timestamp);
        return new ScheduleExam
        {
            Id = Guid.NewGuid(),
            Date = new DateTime(1970, 1,
1).AddMilliseconds(timestampDouble).AddHours(2),
            LecturerName = Lecturer,
            LecturerId = LecturerId,
            Room = Auditory,
            Subject = SubjectName,
            SubjectShort = SubjectShortName,
            Group = Group
        };
    }
}

public record ScheduleItem
{
    [JsonPropertyName("Код")]
    [JsonProperty("Код")]
    public Guid Code { get; set; }

    [JsonPropertyName("ДисциплінаНазва")]
    [JsonProperty("ДисциплінаНазва")]
    public string SubjectName { get; set; }

    [JsonPropertyName("ДисциплінаСкороченаНазва")]
    [JsonProperty("ДисциплінаСкороченаНазва")]
```

```
    public string SubjectShortName { get; set; }

    [JsonPropertyName("ДисциплінаКодРНП")]
    [JsonProperty("ДисциплінаКодРНП")]
    public string SubjectCodeRNP { get; set; }

    [JsonPropertyName("День")]
    [JsonProperty("День")]
    public string Day { get; set; }

    [JsonPropertyName("Пара")]
    [JsonProperty("Пара")]
    public string Lesson { get; set; }

    [JsonPropertyName("Викладач")]
    [JsonProperty("Викладач")]
    public string Lecturer { get; set; }

    [JsonPropertyName("ВикладачКод")]
    [JsonProperty("ВикладачКод")]
    public string LecturerId { get; set; }

    [JsonPropertyName("Аудиторія")]
    [JsonProperty("Аудиторія")]
    public string Auditory { get; set; }

    [JsonPropertyName("ТижденьПочатку")]
    [JsonProperty("ТижденьПочатку")]
    public int? WeekStart { get; set; }

    [JsonPropertyName("ТижденьКінця")]
    [JsonProperty("ТижденьКінця")]
    public int? WeekEnd { get; set; }

    [JsonPropertyName("ТипЗаняття")]
    [JsonProperty("ТипЗаняття")]
    public string LessonType { get; set; }

    [JsonPropertyName("Група")]
    [JsonProperty("Група")]
    public string Group { get; set; }

    [JsonPropertyName("Тиждень")]
    [JsonProperty("Тиждень")]
    public string Week { get; set; }

    public ScheduledLesson AsScheduledLesson()
    {
        // TODO generate id
        return new ScheduledLesson
        {
            Id = Code,
            Day = Day,
            Subject = SubjectName != null ? SubjectName.Trim() : "",
            SubjectShort = SubjectShortName != null ? SubjectShortName.Trim()
: "",
            Number = int.Parse(Lesson),
            Room = Auditory,
            EndWeek = WeekEnd.HasValue  ?  WeekEnd.Value : 0,
            StartWeek = WeekStart.HasValue ? WeekStart.Value  : 0,
            LecturerName = Lecturer != null ? Lecturer.Trim() : "",
            LecturerId = LecturerId,
            LessonType = LessonType != null ? LessonType.Trim() : "",
            GroupName = Group != null ? Group.Trim() : "",
```

```
            Week = int.Parse(Week)
        };
    }
}
```

Файл SingleQuoteJsonReader.cs

```
using System.IO;
using Newtonsoft.Json;

namespace Campus.Schedule.Parser.Entities;

public class SingleQuoteJsonReader : JsonTextReader
{
    public SingleQuoteJsonReader(TextReader reader) : base(reader)
    {
        QuoteChar = '\'';
    }
}
```

Файл Settings.cs

```
namespace Campus.Schedule.Dashboard;

public class Settings
{
    public string ConnectionString { get; set; }

    public JwtSettings Jwt { get; set; } = new JwtSettings();
    public class JwtSettings
    {
        public string Issuer { get; set; }
        public string Audience { get; set; }
        public string Secret { get; set; }

        public TimeSpan AccessTokenExpire => TimeSpan.FromDays(1);
    }
}
```

Файл Program.cs

```
using System.Text;
using Campus.Core.Data;
using Campus.Core.Mappers;
using Campus.Core.Repositories;
using Campus.DAL;
using Campus.DAL.Queries;
using Campus.Models;
using Campus.Schedule.Dashboard;
using Campus.Schedule.Dashboard.Infrastructure;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Identity;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

const string swaggerDocumentName = "campus schedule dashboard";

// Load application settings
var settings = new Settings();
builder.Configuration.Bind(settings);
```

```
settings.ConnectionString =
builder.Configuration.GetConnectionString("DefaultConnection");

// Add services to the container.
builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();

// builder.Services.AddAuthentication(config =>
//     {
//         // config.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
//         config.DefaultChallengeScheme =
JwtBearerDefaults.AuthenticationScheme;
//     })
//     .AddJwtBearer(options =>
//     {
//         options.SaveToken = true;
//         options.RequireHttpsMetadata = false;
//
//         options.TokenValidationParameters = new TokenValidationParameters
//         {
//             ValidateIssuer = false,
//             ValidateAudience = false,
//             ValidateLifetime = false,
//             ValidateIssuerSigningKey = false,
//             ValidateActor = false,
//             ValidIssuer = settings.Jwt.Issuer,
//             ValidAudience = settings.Jwt.Audience,
//         };
//     });

builder.Services.AddSwaggerGen()
    .AddSingleton<Settings>(settings)
    .AddSwaggerGen(
        c => { c.SwaggerDoc("v1", new OpenApiInfo { Title =
"Campus.Schedule.Dashboard", Version = "v1" }); })
    .AddSingleton<QueryManager>(p =>
    {
        var logger = p.GetService<ILogger<QueryManager>>();
        return new QueryManager(
            builder.Environment.ContentRootPath.TrimEnd('/'),
            logger);
    })
    .AddSingleton<ScheduleQueryManager>()
    .AddSingleton<ScheduledMapper>()
    .AddSingleton<LecturerMapper>()
    .AddSingleton<GroupMapper>()
    .AddSingleton<FacultyMapper>()
    .AddScoped<IDatabase, Database>(p =>
    {
        var logger = p.GetRequiredService<ILogger<PostgresDatabase>>();

        return new PostgresDatabase(settings.ConnectionString, logger);
    })
    .AddScoped<IScheduleRepository, ScheduleRepository>()
    .AddScoped<IAdminService, AdminService>()
    .AddScoped<AuthMiddleware>()
    .AddScoped<CampusAuthClient>()
    .AddControllers();

var app = builder.Build();
```

```
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI(
        c => c.SwaggerEndpoint($"/swagger/v1/swagger.json", "campus schedule
dashboard")
    );
}

app.UseHttpsRedirection();

app.UseCors(x => x
    .AllowAnyMethod()
    .AllowAnyHeader()
    .SetIsOriginAllowed(origin => true) // allow any origin
    .AllowCredentials()); // allow credentials

// app.UseAuthorization();

// app.UseAuthentication();

app.UseMiddleware<AuthMiddleware>();

app.MapControllers();

app.Run();
```

Файл Campus.Schedule.Dashboard.csproj

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer"
Version="6.0.5" />
    <PackageReference Include="Swashbuckle.AspNetCore" Version="6.5.0" />
    <None CopyToOutputDirectory="Always" Update="./appsettings.json" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\Campus.Core\Campus.Core.csproj" />
    <ProjectReference Include="..\Campus.Web\Campus.Web.csproj" />
  </ItemGroup>

  <ItemGroup>
    <PackageReference
Include="Microsoft.AspNetCore.Cryptography.KeyDerivation" Version="7.0.5" />
    <PackageReference Include="Microsoft.Extensions.Caching.Memory"
Version="7.0.0" />
    <PackageReference Include="Microsoft.Extensions.Logging" Version="7.0.0"
/>
    <PackageReference Include="Microsoft.IdentityModel.Tokens"
Version="6.30.0" />
    <PackageReference Include="RestSharp" Version="110.2.0" />
    <PackageReference Include="System.IdentityModel.Tokens.Jwt"
Version="6.30.0" />
    <PackageReference Include="JWT" Version="10.0.2" />
```

```
  </ItemGroup>

  <ItemGroup>
    <Compile Remove="Models\UploadMultipartModel.cs" />
    <Compile Remove="Models\PasswordRecoveryRequest.cs" />
    <Compile Remove="Models\KPIIDAuthorizationRequest.cs" />
    <Compile Remove="Models\AuthorizationResponse.cs" />
    <Compile Remove="Models\AuthorizationRequest.cs" />
    <Compile Remove="Models\AuthCheckResponse.cs" />
    <Compile Remove="Models\AccountInformationUpdateRequest.cs" />
    <Compile Remove="Infrastructure\ProfileAuthorizationAttribute.cs" />
    <Compile Remove="Infrastructure\PascalCase.cs" />
    <Compile Remove="Controllers\AccountController.cs" />
  </ItemGroup>

</Project>
```

Файл AuthMiddleware.cs

```
using Campus.Schedule.Dashboard.Infrastructure;

namespace Campus.Schedule.Dashboard;

public class AuthMiddleware : IMiddleware
{
    private readonly CampusAuthClient _authClient;
    private readonly IAdminService _adminService;

    public AuthMiddleware(CampusAuthClient authClient, IAdminService
adminService)
    {
        _authClient = authClient;
        _adminService = adminService;
    }

    public async Task InvokeAsync(HttpContext context, RequestDelegate next)
    {
        var accessToken =
context.Request.Headers["Authorization"].FirstOrDefault()?.Split(" ").Last();

        if (string.IsNullOrEmpty(accessToken))
        {
            context.Response.StatusCode = 401;
            return;
        }

        var userId = await _authClient.Auth(accessToken);

        Console.WriteLine($"User id: {userId}");

        if (string.IsNullOrEmpty(userId))
        {
            context.Response.StatusCode = 401;
            return;
        }

        var adminExists = _adminService.VerifyAdmin(userId);

        if (adminExists)
        {
            await next(context);
```

```
        }
        else
        {
            context.Response.StatusCode = 401;
            return;
        }
    }
}
```

Файл CreateScheduleDto.cs

```
using Campus.Models.Schedule;

namespace Campus.Schedule.Dashboard.Models;

public record CreateScheduleDto
{
    public string GroupId { get; init; }

    public IEnumerable<LessonCreateDto> Lessons { get; init; }
}
```

Файл GroupCreateDto.cs

```
namespace Campus.Schedule.Dashboard.Models;

public record GroupCreateDto
{
    public string GroupName { get; set; }
    public string FacultyNameShort { get; set; }
    public string FacultyNameLong { get; set; }
}
```

Файл LessonCreateDto.cs

```
using System.Diagnostics.CodeAnalysis;

namespace Campus.Schedule.Dashboard.Models;

public record LessonCreateDto
{
    public string Subject { get; init; }
    public string SubjectShort { get; init; }
    public string Day { get; init; }
    public int Number { get; init; }
    public string LecturerName { get; init; }

    [AllowNull]
    public string LecturerId { get; init; }

    [AllowNull]
    public string Room { get; init; }

    public int? StartWeek { get; init; }
    public int? EndWeek { get; init; }
    public string LessonType { get; init; }

    public int Week { get; init; }

}
```

Файл AdminService.cs

```
using Campus.Core.Repositories;

namespace Campus.Schedule.Dashboard.Infrastructure;

public interface IAdminService
{
    bool VerifyAdmin(string userId);
}

public class AdminService : IAdminService
{
    private readonly IScheduleRepository _repo;

    public AdminService(IScheduleRepository repo)
    {
        _repo = repo;
    }

    public bool VerifyAdmin(string userId)
    {
        return _repo.CheckForDashboardAdmin(userId);
    }
}
```

Файл ExamController.cs

```
using Campus.Core.Repositories;
using Campus.Models.Schedule;
using Campus.Web;
using Microsoft.AspNetCore.Mvc;

namespace Campus.Schedule.Dashboard.Controllers;


[ApiController]
[Route("admin/schedule/exams")]
public class ExamController : BaseApiController
{
    private readonly IScheduleRepository _repository;

    public ExamController(IScheduleRepository repository,
ILogger<ScheduleController> logger) : base(logger)
    {
        _repository = repository;
    }

    [HttpGet]
    [Route("")]
    public async Task<IActionResult> GetExamsByGroupId([FromQuery] string
groupId)
    {
        if (string.IsNullOrWhiteSpace(groupId))
        {
            return BadRequest($"Required parameter `{nameof(groupId)}`
missing");
        }

        var lessons = await GetLessons(groupId);

        return await Result(lessons);
    }
```

```
    private async Task<IReadOnlyCollection<ScheduleExam>> GetLessons(string
groupId)
    {
        if (!string.IsNullOrWhiteSpace(groupId))
        {
            return _repository.GetExamsByGenericGroupId(groupId);
        }

        throw new ArgumentException($"Required parameter `{nameof(groupId)}`
missing");
    }
}
```

Файл GroupController.cs

```
using Campus.Core.Repositories;
using Campus.Models.Schedule;
using Campus.Schedule.Dashboard.Models;
using Campus.Web;
using Microsoft.AspNetCore.Mvc;

namespace Campus.Schedule.Dashboard.Controllers;

[ApiController]
[Route("admin/schedule/groups")]
public class GroupController : BaseApiController
{
    private readonly IScheduleRepository _repository;

    public GroupController(IScheduleRepository repository,
ILogger<GroupController> logger) : base(logger)
    {
        _repository = repository;
    }

    [HttpGet]
    [Route("")]
    public async Task<IActionResult> GetGroupsList([FromQuery] string?
filter)
    {
        var groups = !string.IsNullOrWhiteSpace(filter)
            ? _repository.GetGroupsGenericFiltered(filter)
            : _repository.GetGroupsGeneric();

        return await Result(groups);
    }

    [HttpPost]
    [Route("")]
    public async Task<IActionResult> CreateGroup([FromBody] GroupCreateDto
group)
    {
        var genericGroup = new GroupGeneric
        {
            Id = Guid.NewGuid(),
            GroupName = group.GroupName,
            GroupFacultyShort = group.FacultyNameShort,
            GroupFacultyLong = group.FacultyNameLong,
            GroupCode = "" // TODO: figure out that relation
        };

        var result = _repository.AddGroupGeneric(genericGroup);
```

```
        return await Result(result);
    }
}
```

Файл LoginController**.cs**

```
using Campus.Core.Repositories;
using Campus.Web;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Campus.Schedule.Dashboard.Controllers;

[Route("login")]
// [Authorize]
public class LoginController : BaseApiController
{
    private readonly IScheduleRepository _repository;

    public LoginController(IScheduleRepository repository,
ILogger<GroupController> logger) : base(logger)
    {
        _repository = repository;
    }

    [HttpGet]
    [Route("check")]
    public async Task<IActionResult> CheckLogin()
    {
        return await Result(true);
    }
}
```

Файл ScheduleController.cs

```
using Campus.Core.Repositories;
using Campus.Models.Schedule;
using Campus.Schedule.Dashboard.Models;
using Campus.Web;
using Microsoft.AspNetCore.Mvc;

namespace Campus.Schedule.Dashboard.Controllers;

[ApiController]
[Route("admin/schedule/lessons")]
public class ScheduleController : BaseApiController
{
    private readonly IScheduleRepository _repository;

    public ScheduleController(IScheduleRepository repository,
ILogger<ScheduleController> logger) : base(logger)
    {
        _repository = repository;
    }

    [HttpGet]
    [Route("")]
    public async Task<IActionResult> GetLessonsById([FromQuery] string
groupId)
    {
        if (string.IsNullOrWhiteSpace(groupId))
        {
```

```
            return BadRequest($"Required parameter `{nameof(groupId)}`
missing");
        }

        var lessons = await GetLessons(groupId);

        return await Result(lessons);
    }

    /// <summary>
    /// Used to upload schedule for group.
    /// Group should exists already in database.
    /// All lessons for group will be replaced with new ones.
    /// </summary>
    /// <param name="scheduleDto"></param>
    /// <returns></returns>
    [HttpPost]
    [Route("")]
    public async Task<IActionResult> CreateOrReplaceSchedule([FromBody]
CreateScheduleDto scheduleDto)
    {
        if (!Guid.TryParse(scheduleDto.GroupId, out var _))
        {
            throw new ArgumentException($"Required parameter
`{nameof(scheduleDto.GroupId)}` is not valid guid");
        }

        var group = _repository.GetGroupGeneric(scheduleDto.GroupId);

        if (group == null)
        {
            throw new ArgumentException($"Group with id
`{scheduleDto.GroupId}` not found");
        }

        var lessons = new List<ScheduledLesson>();

        foreach (var lDto in scheduleDto.Lessons)
        {
            if (!new[] { "Пн", "Вв", "Ср", "Чт", "Пт", "Сб", "Нд"
}.Contains(lDto.Day))
            {
                throw new ArgumentException(
                    $"Group with id `{scheduleDto.GroupId}` has invalid day
number `{lDto.Day}`");
            }

            if (lDto.Number is not (< 7 and > 0))
            {
                throw new ArgumentException(
                    $"Group with id `{scheduleDto.GroupId}` has invalid
lesson number `{lDto.Number}`");
            }

            if (lDto.Week is not (1 or 2))
            {
                throw new ArgumentException(
                    $"Group with id `{scheduleDto.GroupId}` has invalid week
number `{lDto.Week}`");
            }

            lessons.Add(new ScheduledLesson
            {
                Id = Guid.NewGuid(),
```

```
                Day = lDto.Day,
                Number = lDto.Number,
                Subject = lDto.Subject,
                SubjectShort = lDto.SubjectShort,
                LecturerName = lDto.LecturerName,
                LecturerId = lDto.LecturerId,
                Room = lDto.Room,
                StartWeek = lDto.StartWeek,
                EndWeek = lDto.EndWeek,
                LessonType = lDto.LessonType,
                Week = lDto.Week,
                GroupId = scheduleDto.GroupId,
                GroupName = group.Name
            });
        }


_repository.RemoveGroupScheduleLessonsAll(Guid.Parse(scheduleDto.GroupId));

        foreach (var l in lessons)
        {

_repository.AddGroupScheduleLesson(Guid.Parse(scheduleDto.GroupId), l);
        }

        return await Result(null);
    }

    private async Task<IReadOnlyCollection<ScheduledLesson>>
GetLessons(string groupId)
    {
        if (!string.IsNullOrWhiteSpace(groupId))
        {
            return await _repository.GetLessonsByGroupId(groupId);
        }

        throw new ArgumentException($"Required parameter `{nameof(groupId)}`
missing");
    }
}
```

Файл ./src/util/dayOfWeek.tsx

```
const daysOfWeek: { [dayName: string]: number } = {
    'Пн': 1,
    'Вв': 2,
    'Ср': 3,
    'Чт': 4,
    'Пт': 5,
    'Сб': 6,
};

const sortLessons = (rows: any[]) =>
    rows.sort((a, b) => a.weekNumber - b.weekNumber)
        .sort((a, b) => daysOfWeek[a.dayOfWeek] - daysOfWeek[b.dayOfWeek])
        .sort((a, b) => a.lessonsNumber - b.lessonsNumber);


export
{
    daysOfWeek, sortLessons
}
```

Файл ./src/styles/login.css

```
.login-form-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
  }

  .login-form {
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 100%;
    max-width: 450px;
    padding: 2rem;
    border-radius: 0.5rem;
    box-shadow: 0 5px 10px rgba(0, 0, 0, 0.1);
    background-color: #fff;
  }

  .login-form label {
    font-size: 1.2rem;
    font-weight: bold;
    margin-bottom: 0.5rem;
  }

  .login-form input {
    width: 100%;
    padding: 0.5rem;
    margin-bottom: 1rem;
    border: none;
    border-radius: 0.25rem;
    background-color: #f5f5f5;
  }

  .login-form button[type="submit"] {
    width: 100%;
    padding: 0.75rem;
    margin-top: 1rem;
    border: none;
    border-radius: 0.5rem;
    background-color: #007aff;
    color: #fff;
    font-size: 1.2rem;
    font-weight: bold;
    text-align: center;
    text-transform: uppercase;
    cursor: pointer;
  }

  .login-form button[type="submit"]:hover,
  .login-form button[type="submit"]:focus {
    background-color: #0055ff;
    outline: none;
  }
```

Файл ./src/styles/globals.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

```
/* :root {
  --foreground-rgb: 0, 0, 0;
  --background-start-rgb: 214, 219, 220;
  --background-end-rgb: 255, 255, 255;
}

@media (prefers-color-scheme: dark) {
  :root {
    --foreground-rgb: 255, 255, 255;
    --background-start-rgb: 0, 0, 0;
    --background-end-rgb: 0, 0, 0;
  }
}

body {
  color: rgb(var(--foreground-rgb));
  background: linear-gradient(
      to bottom,
      transparent,
      rgb(var(--background-end-rgb))
    )
    rgb(var(--background-start-rgb));
} */
```

Файл ./src/components/WithAuth.tsx

```
import { useRouter } from "next/router";
import { useEffect, useState } from "react";
import jwt from "jsonwebtoken";

export function withAuth(WrappedComponent: React.ComponentType<any>) {
  return (props: any) => {
    const router = useRouter();
    const [verified, setVerified] = useState<boolean>(false);

    useEffect(() => {
      const token = localStorage.getItem("token");

      if (!token) {
        setVerified(false);
        router.push("/auth/login");
        return;
      }

      let decodeResult: any;

      try {
        decodeResult = jwt.decode(token);
      } catch (err) {
        localStorage.removeItem("token");
        setVerified(false);
        router.push("/auth/login");
        return;
      }

      console.dir(decodeResult);

      if (!decodeResult || decodeResult.exp > Date.now()) {
        localStorage.removeItem("token");
        setVerified(false);
        router.push("/auth/login");
        return;
      }
```

```
      setVerified(true);
    }, [router]);

    return verified && <WrappedComponent {...props} />;
  };
}
```

Файл ./src/components/ExamTable.tsx

```
import { getExams, getLessons, Lesson, updateSchedule } from
"@/services/schedule";
import { useEffect, useState } from "react";

type RowData = {
    id: string;
    date: Date;
    lecturerName: string;
    room: string;
    subjectShort: string;
    subject: string;
    group: string;
};

export default function ExamTable({ groupId }: { groupId: string }) {
  const [rows, setRows] = useState<RowData[]>([]);

  useEffect(() => {
    if (groupId) {
      getExams(groupId, 0, 0).then((schedule) =>
        setRows(
          schedule.data.map((s) => ({
            id: s.id,
            room: s.room,
            date : s.date,
            subjectShort: s.subjectShort,
            group: s.group,
            lecturerName : s.lecturerName,
             subject : s.subject,
          }))
        )
      );
    }
  }, [groupId]);

//   const [newRowData, setNewRowData] = useState<RowData>({
//     id: "",
//     weekNumber: 1,
//     dayOfWeek: "Пн",
//     subjectName: "",
//     lecturer: "",
//     roomNumber: "",
//     lessonType: "Лек",
//   });

  // Sort rows by weekNumber and dayOfWeek
//   const sortedRows = rows.sort((a, b) => {
//     if (a.date !== b.weekNumber) {
//       return a.weekNumber - b.weekNumber;
//     }

//     // Add more sorting logic for dayOfWeek if needed
//     return a.dayOfWeek.localeCompare(b.dayOfWeek);
//   });
```

```
//   const handleNewRowInputChange = (
//     e: React.ChangeEvent<HTMLInputElement | HTMLSelectElement>,
//     field: keyof RowData
//   ) => {
//     setNewRowData({ ...newRowData, [field]: e.target.value });
//   };

//   const addRow = () => {
//     const newRow = {
//       ...newRowData,
//       id: "",
//     };

//     const updatedRows = [...rows, newRow];
//     setRows(updatedRows);

//     setNewRowData({
//       ...newRowData,
//       id: "",
//       subjectName: "",
//       lecturer: "",
//       roomNumber: "",
//     });
//   };

//   const handleUpload = async () => {
//     try {
//       const lessonsDto: Lesson[] = rows.map((r) => ({
//         id: r.id,
//         week: r.weekNumber,
//         day: r.dayOfWeek,
//         endWeek: 0,
//         startWeek: 0,
//         subject: r.subjectName,
//         lecturerName: r.lecturer,
//         room: r.roomNumber,
//         lessonType: r.lessonType,
//         groupId: groupId,
//         groupName: "",
//         lecturerId: "",
//         number: 0, // TODO add number
//         subjectShort: "",
//       }));

//       // Call the API to create the new lessons
//       const newLessons = await updateSchedule(groupId, lessonsDto);

//       // TODO: Handle success response from API
//       console.log("New lessons created:", newLessons);

//       // Clear the new row data and reload the schedule
//       setNewRowData({
//         id: "",
//         weekNumber: 1,
//         dayOfWeek: "Monday",
//         subjectName: "",
//         lecturer: "",
//         roomNumber: "",
//         lessonType: "Лек",
//       });
//       setRows(sortedRows);
//     } catch (error) {
//       // TODO: Handle error response from API
```

```
//       console.error("Failed to create new lessons:", error);
//     }
//   };

  return (
    <div className="p-6">
      <h1 className="text-3xl font-bold mb-6">Schedule</h1>

      <div className="mt-6">
        <h2 className="text-xl font-bold mb-2">Subjects</h2>
        <table className="w-full table-auto mb-6">
          <thead>
            <tr className="bg-gray-200 text-gray-600 uppercase text-sm
leading-normal">
              <th className="py-3 px-6 text-left">Date</th>
              <th className="py-3 px-6 text-left">Subject Name</th>
              <th className="py-3 px-6 text-left">Lecturer</th>
              <th className="py-3 px-6 text-left">Room Number</th>
            </tr>
          </thead>
          <tbody className="text-gray-600 text-sm font-light">
            {rows.map((row) => (
              <tr
                key={row.id}
                className="border-b border-gray-200 hover:bg-gray-100"
              >
                <td className="py-3 px-6 text-
left">{row.date.toString()}</td>
                <td className="py-3 px-6 text-left">{row.subject}</td>
                <td className="py-3 px-6 text-left">{row.lecturerName}</td>
                <td className="py-3 px-6 text-left">{row.room}</td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
  );
}
```

Файл ./src/components/GroupCreateForm.tsx

```
import { useState } from "react";
import axios from "axios";
import { createGroup } from "@/services/groups";

type GroupCreateFormProps = {
  onGroupCreated: (group: any) => void;
};

type GroupCreateFormState = {
  groupName: string;
  facultyNameShort: string;
  facultyNameLong: string;
};

export default function GroupCreateForm(props: GroupCreateFormProps) {
  const [formData, setFormData] = useState<GroupCreateFormState>({
    groupName: "",
    facultyNameShort: "",
    facultyNameLong: "",
  });
```

```
const handleInputChange = (event: React.ChangeEvent<HTMLInputElement>) => {
  const { name, value } = event.target;

  setFormData({
    ...formData,
    [name]: value,
  });
};

const handleSubmit = (event: React.FormEvent<HTMLFormElement>) => {
  event.preventDefault();

  const newGroup = {
    groupName: formData.groupName,
    facultyNameShort: formData.facultyNameShort,
    facultyNameLong: formData.facultyNameLong,
  };

  createGroup(newGroup)
    .then((group) => {
      props.onGroupCreated(newGroup);
      setFormData({
        groupName: "",
        facultyNameShort: "",
        facultyNameLong: "",
      });
    })
    .catch((error) => {
      console.error("Error creating group", error);
    });
};

return (
  <form onSubmit={handleSubmit}>
    <div className="mb-4">
      <label htmlFor="group-name" className="block font-bold mb-2">
        Group Name
      </label>
      <input
        type="text"
        name="groupName"
        id="group-name"
        value={formData.groupName}
        onChange={handleInputChange}
        className="border border-gray-400 p-2 rounded-md"
      />
    </div>
    <div className="mb-4">
      <label htmlFor="faculty-short-name" className="block font-bold mb-2">
        Faculty Short Name
      </label>
      <input
        type="text"
        name="facultyNameShort"
        id="faculty-short-name"
        value={formData.facultyNameShort}
        onChange={handleInputChange}
        className="border border-gray-400 p-2 rounded-md"
      />
    </div>
    <div className="mb-4">
      <label htmlFor="faculty-long-name" className="block font-bold mb-2">
        Faculty Long Name
      </label>
```

```
        <input
          type="text"
          name="facultyNameLong"
          id="faculty-long-name"
          value={formData.facultyNameLong}
          onChange={handleInputChange}
          className="border border-gray-400 p-2 rounded-md"
        />
      </div>
      <div className="mt-8">
        <button
          type="submit"
          className="px-4 py-2 bg-blue-500 text-white rounded-md"
        >
          Create Group
        </button>
      </div>
    </form>
  );
}
```

Файл ./src/components/SubjectTable.tsx

```
import {getLessons, Lesson, updateSchedule} from "@/services/schedule";
import {useEffect, useState} from "react";
import {daysOfWeek, sortLessons} from "@/util/dayOfWeek";

type RowData = {
    id: string;
    weekNumber: number;
    dayOfWeek: string;
    lessonsNumber: number;
    subjectName: string;
    lecturer: string;
    roomNumber: string;
    lessonType: string;
};

const newRowDataDefault: RowData = {
    id: "",
    subjectName: "",
    lecturer: "",
    roomNumber: "",
    weekNumber: 1,
    dayOfWeek: "Пн",
    lessonsNumber: 1,
    lessonType: "Лек",
}

export default function SubjectTable({groupId}: { groupId: string }) {
    const [rows, setRows] = useState<RowData[]>([]);
    const [rowEditing, setRowEditing] = useState<number | null>(null);

    const [newRowData, setNewRowData] = useState<RowData>({
        id: "",
        weekNumber: 1,
        dayOfWeek: "Пн",
        lessonsNumber: 1,
        subjectName: "",
        lecturer: "",
        roomNumber: "",
        lessonType: "Лек",
    });
```

```
useEffect(() => {
    if (groupId) {
        getLessons(groupId, 0, 0).then((schedule) =>
            setRows(
                sortLessons(schedule.data.map((s) => ({
                        id: s.id,
                        weekNumber: s.week,
                        dayOfWeek: s.day,
                        lessonsNumber: s.number,
                        subjectName: s.subject,
                        lecturer: s.lecturerName,
                        roomNumber: s.room,
                        lessonType: s.lessonType,
                    }))
                )
            )
        );
    }
}, [groupId]);

useEffect(() => {
    if (rowEditing) {
        setNewRowData({
            ...rows[rowEditing],
        });
    } else {
        // to handle case when editing is cancelled
        if (!newRowData) {
            setNewRowData(newRowDataDefault);
        }
    }
}, [rowEditing])


const handleNewRowInputChange = (
    e: React.ChangeEvent<HTMLInputElement | HTMLSelectElement>,
    field: keyof RowData
) => {
    setNewRowData({...newRowData, [field]: e.target.value});
};

const addRow = () => {
    const newRow = {
        ...newRowData,
        id: "",
    };

    const updatedRows = [...rows, newRow];
    setRows(updatedRows);

    setNewRowData(newRowDataDefault);
};

const updateRow = () => {
    if (rowEditing) {
        const rowsUpdated = JSON.parse(JSON.stringify(rows));
        rowsUpdated[rowEditing] = {...newRowData, id: ""};

        console.dir(rowsUpdated)
        setRows(rowsUpdated)
        setNewRowData(newRowDataDefault);
        setRowEditing(null);
    }
```

```
    }

    const handleUpload = async () => {
        try {
            const lessonsDto: Lesson[] = rows.map((r) => ({
                id: r.id,
                week: r.weekNumber,
                day: r.dayOfWeek,
                endWeek: 0,
                startWeek: 0,
                subject: r.subjectName,
                lecturerName: r.lecturer,
                room: r.roomNumber,
                lessonType: r.lessonType,
                groupId: groupId,
                groupName: "",
                lecturerId: "",
                number: r.lessonsNumber,
                subjectShort: "",
            }));

            // Call the API to create the new lessons
            const newLessons = await updateSchedule(groupId, lessonsDto);

            // TODO: Handle success response from API
            console.log("New lessons created:", newLessons);

            // Clear the new row data and reload the schedule
            setNewRowData(newRowDataDefault);
            setRows(rows);
        } catch (error) {
            // TODO: Handle error response from API
            console.error("Failed to create new lessons:", error);
        }
    };

    const handleRowEdit = (rowIndex: number) => {
        console.log("Editing row:", rowIndex);
        setRowEditing(rowIndex)
    }

    const handleRowDelete = (rowIndex: number) => {
        console.log("Deleting row:", rowIndex);
        const rowsUpdated = [...rows];
        rowsUpdated.splice(rowIndex, 1);
        setRows(rowsUpdated);
    }

    return (
        <div className="p-6">
            <h1 className="text-3xl font-bold mb-6">Schedule</h1>

            <div className="mt-6">
                <h2 className="text-xl font-bold mb-2">Subjects</h2>
                <table className="w-full table-auto mb-6">
                    <thead>
                    <tr className="bg-gray-200 text-gray-600 uppercase text-
sm leading-normal">
                        <th className="py-3 px-6 text-left">Week Number</th>
                        <th className="py-3 px-6 text-left">Day of Week</th>
                        <th className="py-3 px-6 text-left">Lessons
number</th>
                        <th className="py-3 px-6 text-left">Subject Name</th>
                        <th className="py-3 px-6 text-left">Lecturer</th>
```

```
                        <th className="py-3 px-6 text-left">Room Number</th>
                        <th className="py-3 px-6 text-left">Type of
Lesson</th>
                    </tr>
                    </thead>
                    <tbody className="text-gray-600 text-sm font-light">
                    {rows.map((row, i) => (
                        <tr
                            key={row.id}
                            className="border-b border-gray-200 hover:bg-
gray-100"
                        >
                            <td className="py-3 px-6 text-
left">{row.weekNumber}</td>
                            <td className="py-3 px-6 text-
left">{row.dayOfWeek}</td>
                            <td className="py-3 px-6 text-
left">{row.lessonsNumber}</td>
                            <td className="py-3 px-6 text-
left">{row.subjectName}</td>
                            <td className="py-3 px-6 text-
left">{row.lecturer}</td>
                            <td className="py-3 px-6 text-
left">{row.roomNumber}</td>
                            <td className="py-3 px-6 text-
left">{row.lessonType}</td>
                            <td className="py-3 px-6 text-left">
                                <button onClick={() => handleRowEdit(i)}>
Edit</button>
                            </td>
                            <td className="py-3 px-6 text-left">
                                <button onClick={() => handleRowDelete(i)}>
Remove</button>
                            </td>
                        </tr>
                    ))}
                    </tbody>
                </table>
            </div>
            <div className="flex flex-col gap-2">
                <h2 className="text-xl font-bold mb-2">Add subject</h2>
                <label htmlFor="new-week-number">Week Number:</label>
                <select
                    id="new-week-number"
                    className="rounded-lg border-gray-300 shadow-sm
focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-
50"
                    value={newRowData.weekNumber}
                    onChange={(e) => handleNewRowInputChange(e,
"weekNumber")}
                >
                    <option value="1">1</option>
                    <option value="2">2</option>
                </select>
                <label htmlFor="new-day-of-week">Day of Week:</label>
                <select
                    id="new-day-of-week"
                    className="rounded-lg border-gray-300 shadow-sm
focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-
50"
                    value={newRowData.dayOfWeek}
                    onChange={(e) => handleNewRowInputChange(e, "dayOfWeek")}
                >
                    <option value="Пн">Пн</option>
```

```
                    <option value="Вв">Вв</option>
                    <option value="Ср">Ср</option>
                    <option value="Чт">Чт</option>
                    <option value="Пт">Пт</option>
                    <option value="Сб">Сб</option>
                    <option value="Нд">Нд</option>
                </select>
                <label htmlFor="new-lesson-number">Lessons number : </label>
                <select id="new-lesson-number"
                        className="rounded-lg border-gray-300 shadow-sm
focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-
50"
                    value={newRowData.lessonsNumber}
                    onChange={(e) => handleNewRowInputChange(e,
"lessonsNumber")}
                >
                    {Array.from(Array(8).keys()).map((i) => (
                        <option key={i} value={i + 1}>
                            {i + 1}
                        </option>
                    ))}
                </select>
                <label htmlFor="new-subject-name">Subject Name:</label>
                <input
                    id="new-subject-name"
                    type="text"
                    className="rounded-lg border-gray-300 shadow-sm
focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-
50"
                    value={newRowData.subjectName}
                    onChange={(e) => handleNewRowInputChange(e,
"subjectName")}
                />
                <label htmlFor="new-lecturer">Lecturer:</label>
                <input
                    id="new-lecturer"
                    type="text"
                    className="rounded-lg border-gray-300 shadow-sm
focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-
50"
                    value={newRowData.lecturer}
                    onChange={(e) => handleNewRowInputChange(e, "lecturer")}
                />
                <label htmlFor="new-room-number">Room Number:</label>
                <input
                    id="new-room-number"
                    type="text"
                    className="rounded-lg border-gray-300 shadow-sm
focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-
50"
                    value={newRowData.roomNumber}
                    onChange={(e) => handleNewRowInputChange(e,
"roomNumber")}
                />
                <label htmlFor="new-lesson-type">Type of Lesson:</label>
                <select
                    id="new-lesson-type"
                    className="rounded-lg border-gray-300 shadow-sm
focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-
50"
                    value={newRowData.lessonType}
                    onChange={(e) => handleNewRowInputChange(e,
"lessonType")}
                >
```

```
                    <option value="Лек">Лек</option>
                    <option value="Лаб">Лаб</option>
                    <option value="Прак">Прак</option>
                </select>
                {
                    rowEditing ?
                        <button
                            className="bg-indigo-500 hover:bg-indigo-700
text-white font-bold py-2 px-4 rounded focus:outline-none focus:shadow-
outline"
                            onClick={updateRow}
                        >
                            Update Row
                        </button>
                        : <button
                            className="bg-indigo-500 hover:bg-indigo-700
text-white font-bold py-2 px-4 rounded focus:outline-none focus:shadow-
outline"
                            onClick={addRow}
                        >
                            Add Row
                        </button>

                }

            </div>
            <button
                className="bg-indigo-500 hover:bg-indigo-700 text-white font-
bold py-2 px-4 rounded focus:outline-none focus:shadow-outline"
                onClick={handleUpload}
            >
                Update schedule
            </button>
        </div>
    );
}
```

Файл ./src/components/Logout.tsx

```
import { useEffect, useState } from "react";
import { useRouter } from "next/router";

export default function Logout() {
  const router = useRouter();
  const [isLoggingOut, setIsLoggingOut] = useState(false);

  useEffect(() => {
    if (isLoggingOut) {
      localStorage.removeItem("token");
      router.push("/");
    }
  }, [isLoggingOut]);

  const handleLogout = () => {
    setIsLoggingOut(true);
  };

  return (
    <button
      className="px-4 py-2 bg-grey-500 text-black rounded hover:bg-blue-600"
      onClick={handleLogout}
    >
      {isLoggingOut ? "Logging out..." : "Logout"}
```

```
    </button>
  );
}
```

Файл ./src/components/GroupList.tsx

```
import { useState, useEffect } from "react";
import { Group, getGroups } from "../services/groups";
import GroupCreateForm from "./GroupCreateForm";

export default function GroupList() {
  const [groups, setGroups] = useState<Group[]>([]);
  const [currentPage, setCurrentPage] = useState(1);
  const [totalPages, setTotalPages] = useState(0);

  useEffect(() => {
    getGroups(currentPage).then((data) => {
      console.dir(data);
      setGroups(data.data);
      setTotalPages(data.pagination.totalPages);
    });
  }, [currentPage]);

  const handleGroupCreated = (group: any) => {
    const newGroup = {
      id: "",
      name: group.groupName,
      faculty: group.facultyNameShort,
    };

    setGroups([...groups, newGroup]);
  };

  const handlePrevPage = () => {
    setCurrentPage((prevPage) => prevPage - 1);
  };

  const handleNextPage = () => {
    setCurrentPage((prevPage) => prevPage + 1);
  };

  return (
    <div className="ml-4">
      <h2 className="text-xl font-bold mb-2">Groups</h2>

      <table className="table-auto w-full">
        <thead>
          <tr>
            <th className="px-4 py-2">Name</th>
            <th className="px-4 py-2">Faculty</th>
          </tr>
        </thead>
        <tbody>
          {groups.map((group) => (
            <tr key={group.id}>
              <td className="border px-4 py-2">{group.name}</td>
              <td className="border px-4 py-2">{group.faculty}</td>
            </tr>
          ))}
        </tbody>
      </table>

      <div className="flex justify-between items-center mt-4">
```

```
        <div>
          Page {currentPage} of {totalPages}
        </div>
        <div>
          <button
            onClick={handlePrevPage}
            disabled={currentPage === 1}
            className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-
2 px-4 rounded mr-2"
          >
            Prev
          </button>
          <button
            onClick={handleNextPage}
            disabled={currentPage === totalPages}
            className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-
2 px-4 rounded"
          >
            Next
          </button>
        </div>
      </div>

      <div className="mt-4">
        <h2 className="text-xl font-bold mb-2">Create a new group</h2>
        <GroupCreateForm onGroupCreated={handleGroupCreated} />
      </div>
    </div>
  );
}
```

Файл ./src/components/GroupSelector.tsx

```
import { useEffect, useState } from "react";
import SubjectTable from "./SubjectTable";
import { Group, getGroups } from "@/services/groups";

type GroupSelectorProps = {
  onGroupChange: (group: Group) => void;
};

export default function GroupSelector({ onGroupChange }: GroupSelectorProps)
{
  const [groups, setGroups] = useState<Group[]>([]);
  const [selectedGroup, setSelectedGroup] = useState<Group>({
    id: "",
    name: "",
    faculty: "",
  });

  useEffect(() => {
    getGroups(0).then((data) => {
      setGroups(data.data);
      setSelectedGroup(data.data[0]);
    });
  }, []);

  const handleGroupChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
    const groupName = e.target.value;
    const group = groups.find((g) => g.name === groupName);

    if (group) {
      setSelectedGroup(group);
```

```
      onGroupChange(group);
    }
  };

  return (
    <div className="p-6">
      <h1 className="text-3xl font-bold mb-6">Select a group</h1>
      <select
        className="border-gray-300 shadow-sm focus:ring focus:ring-indigo-200
focus:border-indigo-300 rounded-md"
        value={selectedGroup.name}
        onChange={handleGroupChange}
      >
        {groups
          .map((g) => g.name)
          .map((groupName) => (
            <option key={groupName} value={groupName}>
              {groupName}
            </option>
          ))}
      </select>
      {selectedGroup && <SubjectTable groupId={selectedGroup.id} />}
    </div>
  );
}
```

Файл ./src/components/Sidebar.tsx

```
import { useRouter } from "next/router";
import Logout from "./Logout";

type SidebarItemProps = {
  href: string;
  label: string;
};

function SidebarItem({ href, label }: SidebarItemProps) {
  const router = useRouter();

  return (
    <a
      href={href}
      className={`p-2 rounded-md cursor-pointer ${
        router.pathname === href ? "bg-gray-100" : ""
      }`}
    >
      {label}
    </a>
  );
}

export default function Sidebar() {
  return (
    <div className="bg-gray-200 w-48 h-screen flex flex-col items-start py-
4">
      <SidebarItem href="/groups" label="Groups" />
      <SidebarItem href="/subjects" label="Subjects" />
      <SidebarItem href="/exams" label="Exams" />
      <SidebarItem href="/logout" label="Logout" />
      <Logout />
    </div>
  );
}
```

Файл ./src/components/GroupSelectorGeneric.tsx

```
import {getGroups, Group} from "@/services/groups";
import React, {useEffect, useState} from "react";

type GroupSelectorProps = {
    onGroupChange: (group: Group) => void;
    WithGroup: React.ComponentType<WithGroup>;
};

interface WithGroup{
    groupId: string;
}

export default function GroupSelectorGeneric({onGroupChange, WithGroup }:
GroupSelectorProps) {
    const [groups, setGroups] = useState<Group[]>([]);
    const [selectedGroup, setSelectedGroup] = useState<Group>({
        id: "",
        name: "",
        faculty: "",
    });

    useEffect(() => {
        getGroups(0).then((data) => {
            setGroups(data.data);
            setSelectedGroup(data.data[0]);
        });
    }, []);

    const handleGroupChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
        const groupName = e.target.value;
        const group = groups.find((g) => g.name === groupName);

        if (group) {
            setSelectedGroup(group);
            onGroupChange(group);
        }
    };

    return (
        <div className="p-6">
            <h1 className="text-3xl font-bold mb-6">Select a group</h1>
            <select
                className="border-gray-300 shadow-sm focus:ring focus:ring-
indigo-200 focus:border-indigo-300 rounded-md"
                value={selectedGroup.name}
                onChange={handleGroupChange}
            >
                {groups
                    .map((g) => g.name)
                    .map((groupName) => (
                        <option key={groupName} value={groupName}>
                            {groupName}
                        </option>
                    ))}
            </select>
            {selectedGroup && <WithGroup groupId={selectedGroup.id}/>}
        </div>
    );
}
```

Файл ./src/components/GroupSelectorExam.tsx

```
import { useEffect, useState } from "react";
import SubjectTable from "./SubjectTable";
import { Group, getGroups } from "@/services/groups";
import ExamTable from "./ExamTable";

type GroupSelectorProps = {
  onGroupChange: (group: Group) => void;
};

export default function GroupSelectorExam({ onGroupChange }:
GroupSelectorProps) {
  const [groups, setGroups] = useState<Group[]>([]);
  const [selectedGroup, setSelectedGroup] = useState<Group>({
    id: "",
    name: "",
    faculty: "",
  });

  useEffect(() => {
    getGroups(0).then((data) => {
      setGroups(data.data);
      setSelectedGroup(data.data[0]);
    });
  }, []);

  const handleGroupChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
    const groupName = e.target.value;
    const group = groups.find((g) => g.name === groupName);

    if (group) {
      setSelectedGroup(group);
      onGroupChange(group);
    }
  };

  return (
    <div className="p-6">
      <h1 className="text-3xl font-bold mb-6">Select a group</h1>
      <select
        className="border-gray-300 shadow-sm focus:ring focus:ring-indigo-200
focus:border-indigo-300 rounded-md"
        value={selectedGroup.name}
        onChange={handleGroupChange}
      >
        {groups
          .map((g) => g.name)
          .map((groupName) => (
            <option key={groupName} value={groupName}>
              {groupName}
            </option>
          ))}
      </select>
      {selectedGroup && <ExamTable groupId={selectedGroup.id} />}
    </div>
  );
}
```

Файл ./src/pages/index.tsx

```
export default function Home() {
  return <>Система управління розкладом</>;
```

```
}
```

Файл ./src/pages/auth/login.tsx

```
import { ChangeEvent, FormEvent, useState } from "react";
import axios from "axios";
import { useRouter } from "next/router";

type LoginFormProps = {
  // onSubmit: (username: string, password: string) => void;
};

export default function LoginForm({}: LoginFormProps) {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [loginError, setLoginError] = useState("");
  const router = useRouter();

  const handleUsernameChange = (event: ChangeEvent<HTMLInputElement>) => {
    setUsername(event.target.value);
  };

  const handlePasswordChange = (event: ChangeEvent<HTMLInputElement>) => {
    setPassword(event.target.value);
  };

  const handleSubmit = async (event: FormEvent<HTMLFormElement>) => {
    event.preventDefault();
    try {
      const response = await axios.post("/api/auth/login", {
        username,
        password,
      });

      const tokens = response.data.token;
      localStorage.setItem("token", tokens);

      router.push("/");
    } catch (err) {
      setLoginError("Login failed. Please check your username and
password.");
    }
  };

  return (
    <form
      className="max-w-sm mx-auto p-8 bg-white rounded-md shadow-md"
      onSubmit={handleSubmit}
    >
      <h2 className="text-2xl font-bold text-gray-800 mb-6">Login</h2>
      <div className="mb-4">
        <label
          htmlFor="username"
          className="block text-gray-700 font-semibold mb-2"
        >
          Username:
        </label>
        <input
          type="text"
          id="username"
          className="w-full rounded-md border-gray-300 shadow-sm
focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-
50"
```

```
          value={username}
          onChange={handleUsernameChange}
        />
      </div>
      <div className="mb-4">
        <label
          htmlFor="password"
          className="block text-gray-700 font-semibold mb-2"
        >
          Password:
        </label>
        <input
          type="password"
          id="password"
          className="w-full rounded-md border-gray-300 shadow-sm
focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-
50"
          value={password}
          onChange={handlePasswordChange}
        />
      </div>
      {loginError && <p className="">{loginError}</p>}
      <button
        type="submit"
        className="bg-indigo-500 hover:bg-indigo-700 text-white font-bold py-
2 px-4 rounded focus:outline-none focus:shadow-outline"
      >
        Submit
      </button>
    </form>
  );
}
```

Файл ./src/pages/_document.tsx

```
import { Html, Head, Main, NextScript } from 'next/document'

export default function Document() {
  return (
    <Html lang="en">
      <Head />
      <body>
        <Main />
        <NextScript />
      </body>
    </Html>
  )
}
```

Файл ./src/pages/api/auth/login.ts

```
import { NextApiRequest, NextApiResponse } from "next";
import axios from "axios";
import qs from "querystring";

export default async function login(req: NextApiRequest, res:
NextApiResponse) {
  const { username, password } = req.body;

  const dataEncoded = qs.stringify({
    Username: username,
    Password: password,
  });
```

```
  const authResponse = await axios.post(
    `${process.env.AUTH_ENDPOINT}/oauth/token`,
    dataEncoded,
    {
      headers: {
        "Content-Type": "application/x-www-form-urlencoded",
      },
    }
  );

  const accessToken = authResponse.data.access_token;

  const authCheck = await axios.get("http://localhost:5290/login/check", {
    headers: {
      Authorization: `Bearer ${accessToken}`,
    },
  });

  if (!authCheck.data) {
    res.status(401).json({ message: "Unauthorized" });
  }

  res.status(200).json({ token: accessToken });
}
```

Файл ./src/pages/api/auth/index.ts

```
import login from "./login";

export { login };
```

Файл ./src/pages/api/hello.ts

```
// Next.js API route support: https://nextjs.org/docs/api-routes/introduction
import type { NextApiRequest, NextApiResponse } from 'next'

type Data = {
  name: string
}

export default function handler(
  req: NextApiRequest,
  res: NextApiResponse<Data>
) {
  res.status(200).json({ name: 'John Doe' })
}
```

Файл ./src/pages/_app.tsx

```
import "@/styles/globals.css";
import type {AppProps} from "next/app";
import Sidebar from "../components/Sidebar";
import {withAuth} from "@/components/WithAuth";
import React from "react";
import {useRouter} from "next/router";

function MyApp({Component, pageProps}: AppProps) {
    const router = useRouter();

    return (
        <div className="flex">
            {router.pathname !== "/login" && <Sidebar/>}
```

```
            <div className="flex-1">
                <Component {...pageProps} />
            </div>
        </div>
    );
}

export default MyApp;
```

Файл ./src/pages/exams.tsx

```
import GroupSelectorGeneric from "@/components/GroupSelectorGeneric";
import ExamTable from "@/components/ExamTable";

export default function Exams() {
    return <GroupSelectorGeneric onGroupChange={() => console.log("Group
changed")} WithGroup={ExamTable}/>;
}
```

Файл ./src/pages/groups.tsx

```
import GroupList from "@/components/GroupList";
import { withAuth } from "@/components/WithAuth";

function Groups() {
  return <GroupList />;
}

export default withAuth(Groups);
```

Файл ./src/pages/subjects.tsx

```
import SubjectTable from "@/components/SubjectTable";
import {withAuth} from "@/components/WithAuth";
import GroupSelectorGeneric from "@/components/GroupSelectorGeneric";

function Subjects() {
    return <GroupSelectorGeneric onGroupChange={() => console.log("Group
changed")} WithGroup={SubjectTable}/>;
}

export default withAuth(Subjects);
```

Файл ./src/services/schedule.ts

```
import axios from "axios";

const BASE_URL = "http://localhost:5290";

export interface Lesson {
  id: string;
  subject: string;
  subjectShort: string;
  day: string;
  number: number;
  lecturerName: string;
  lecturerId: string;
  room: string;
  startWeek: number;
  endWeek: number;
  lessonType: string;
  groupId: string | null;
```

```
  groupName: string;
  week: number;
}

export interface Exam {
  id: string;
  date: Date;
  lecturerName: string;
  lecturerId: string;
  room: string;
  subjectShort: string;
  subject: string;
  group: string;
}

export const getLessons = async (
  groupId: string,
  pageNumber: number,
  pageSize: number
) => {
  const response = await axios.get(`${BASE_URL}/admin/schedule/lessons`, {
    params: {
      groupId,
      pageNumber,
      pageSize,
    },
    headers: {
      Authorization: `Bearer ${localStorage.getItem("token")}`,
    },
  });
  return {
    data: response.data.data as Lesson[],
    pagination: response.data.paging,
  };
};

export const getExams = async (
  groupId: string,
  pageNumber: number,
  pageSize: number
) => {
  const response = await axios.get(`${BASE_URL}/admin/schedule/exams`, {
    params: {
      groupId,
    },
    headers: {
      Authorization: `Bearer ${localStorage.getItem("token")}`,
    },
  });

  return { data: response.data.data as Exam[] };
};

export const updateSchedule = async (groupId: string, lessons: Lesson[]) => {
  const response = await axios.post(
    `${BASE_URL}/admin/schedule/lessons`,
    {
      groupId,
      lessons,
    },
    {
      headers: {
        Authorization: `Bearer ${localStorage.getItem("token")}`,
      },
```

```
    }
  );

  return response.data;
};
```

Файл ./src/services/groups.ts

```
import axios from "axios";

export type Group = {
  id: string;
  name: string;
  faculty: string;
};

export type GroupCreateDTO = {
  groupName: string;
  facultyNameShort: string;
  facultyNameLong: string;
};

const BASE_URL = "http://localhost:5290";

export const getGroups = async (page: number) => {
  const response = await axios.get(`${BASE_URL}/admin/schedule/groups`, {
    headers: {
      Authorization: `Bearer ${localStorage.getItem("token")}`,
    },
  });
  return {
    data: response.data.data as Group[],
    pagination: response.data.paging,
  };
};

export const createGroup = async (group: GroupCreateDTO): Promise<Group> => {
  const response = await axios.post(
    `${BASE_URL}/admin/schedule/groups`,
    group,
    {
      headers: {
        Authorization: `Bearer ${localStorage.getItem("token")}`,
      },
    }
  );
  return response.data;
};
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДГОТОВКИ ТА РОБОТИ З РОЗКЛАДОМ ЗАНЯТЬ

Програма та методика тестування

КПІ.ІТ-9230.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____Олександр ЯЦЕВСЬКИЙ

Київ – 2023

ЗМІСТ

ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є програмне забезпечення для підготовки та роботи з розкладом занять. Система складається з СУБД, модуля синхронізації даних та інтерфейсу адміністратора. Система відповідає за інтеграцію нової інфраструктури розкладу занять з наявними компонентами системи.

МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних.

МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– 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;

ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Для того щоб перевірити працездатність і відмово стійкість застосунку необхідно провести наступні етапи тестування для різних модулів системи:

– тестування синхронізації даних при великій кількості параметрів для синхронізації;

– тестування синхронізації даних в умовах нестабільної інфраструктури - не постійної доступності API інфраструктури ЄІС;

– тестування синхронізації даних частинами з інтервалом у часі;

– тестування інтерфейсу адміністратора на підтримку різних браузерів;

– динамічне тестування на відповідність функціональним вимогам.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023
р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДГОТОВКИ ТА РОБОТИ З РОЗКЛАДОМ ЗАНЯТЬ

Керівництво користувача

КПІ.ІТ-9230.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____Олександр ЯЦЕВСЬКИЙ

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

“Програмне забезпечення для підготовки та роботи з розкладом занять” – це програмний продукт для інтеграції нової системи розкладу в Єдине інформаційне середовище університету. Дане ПЗ складається з кількох модулів для серверної та клієнтської частин. В функціонал ПЗ входить:

- провадження бази даних для розладу занять;
- синхронізація бази даних з сервісами ЄІС;
- надання веб-інтерфейсу для управління розкладом занять.

1. ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

1.1 Системні вимоги для коректної роботи

Для розгортання серверної частини застосунку необхідні вимоги :

– підтримка docker контейнерів;
– наявність доступу в інтернет;
– не менше 4 ГБ вільної пам’яті на диску;
– не менше 4 ГБ оперативної пам’яті.

Для використання клієнтської частини додатку:

– наявність доступу в інтернет;
– наявність актуальної версії браузера.

1.2 Завантаження застосунку

На даний момент установлення ПЗ доступно як docker image. Для розгортання необхідно завантажити файл docker-compose.yml з репозиторія проекту, вказати необхідну конфігурацію і запустити систему контейнерів.

1.3 Перевірка коректної роботи

Для перевірки коректності роботи розгорнутої системи можна провести наступний тест:

1. В таблицю “Groups” потрібно додати назви груп, для яких необхідно завантажити розклади;
2. В командному рядку необхідно виконати команду для запуску проекту “Campus.Schedule.Parser”. Після цього у виводі за допомогою системи логування можна побачити поточний статус процесу;
3. Після завершення виконання програми, таблиці “ScheduledLessons” та “ScheduledExams” мають бути заповнені.

4. Виконати вхід під іменем користувача з правами адміністратора системи розклад в інтерфейс адміністратора.

5. На сторінці “Groups” можна побачити список груп, який має відповідати списку, зконфігурованому у пункті 1.

6. На сторінці “Subjects” для кожної з обраних груп має бути присутній розклад занять.

2. ВИКОНАННЯ ПРОГРАМИ

3.1 Виконання системи для системного адміністратора

Після розгортання сервісів і створення бази даних, наступним кроком потрібно провести синхронізацію бази даних.

Для початку синхронізації, в базу даних необхідно імпортувати данні списку груп, для яких відбудеться процедура синхронізації (рис 3.1).

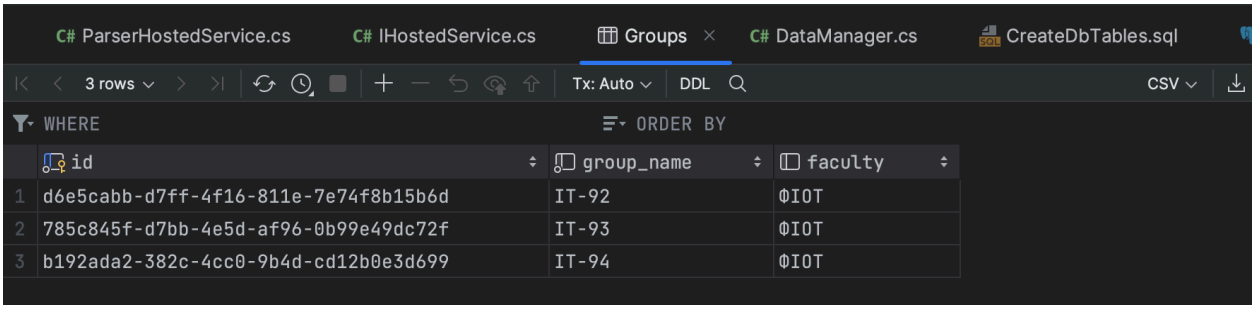


Рисунок 3.1 – Список груп для синхронізації.

Після цього необхідно запустити додаток Campus.Schedule.Parser. (рис 3.2)

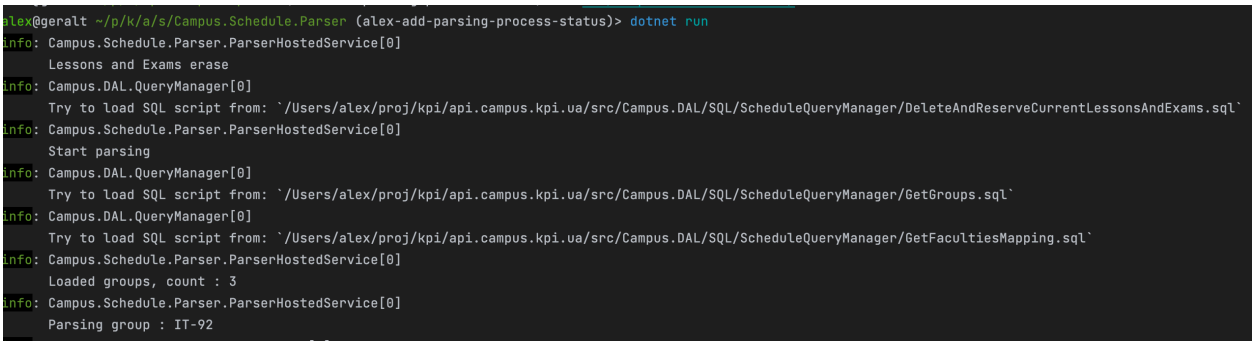


Рисунок 3.2 – Вивід сервісу синхронізації.

По завершенню робити програми у зконфігурованній базі даних мають бути заповнені дані для груп, занять та екзаменів. Перегляд даних можливий з веб-додатку для інтерфейсу адміністратора, робота з яким описана в пункті 3.2.

3.2 Виконання програми для користувача

При запуску застосунку перед користувачем відкривається сторінка авторизації. Необхідно ввести ім’я користувача та пароль.

У випадку, якщо дані введені неправильно, перед користувач побачить повідомлення про некоректність даних.

Якщо дані коректні, перед користувачем відкриється початкова сторінка системи.

Для того, щоб відобразити список груп, користувач має натиснути на кнопку Groups в боковій панелі. На екрані зобразиться список груп, наявних в базі даних.

Для того, щоб додати групу, користувач має змогу ввести данні групи у відповідні поля форми і натиснути на кнопку “Create Group”.

Для того щоб видалити групу, користувач може натиснути на кнопку “Remove” в рядку обраної групи.

Після внесення змін у список груп, предметів чи екзаменів, на сторінці користувача з'явиться кнопка "Update Schedule". Користувач може натиснути на неї, щоб зберегти внесені зміни та переглянути оновлений розклад занять і екзаменів.

Для того, щоб відобразити список предметів, користувач має натиснути на кнопку "Subjects" в боковій панелі. На екрані зобразиться розклад занять для всіх груп в системі. Для кожного предмету на сторінці "Subjects" наявна кнопка "Edit" (Редагувати), яку можна використовувати для редагування даних предмету, та кнопка "Remove" (Видалити), яку можна використовувати для видалення предмету з бази даних. Щоб додати новий предмет, користувач має натиснути на кнопку "Add Subject" на сторінці "Subjects" і заповнити форму з даними нового предмету, після чого натиснути кнопку "Create Subject". Для редагування даних предмету,

користувач має натиснути кнопку "Edit" біля потрібного предмету на сторінці "Subjects". Після цього він буде перенаправлений на сторінку з формою для редагування даних предмету. Після внесення необхідних змін користувач повинен натиснути кнопку " Update Schedule ". Для видалення предмету, користувач має натиснути кнопку "Remove" біля потрібного предмету на сторінці "Subjects". Після цього він буде попереджений, що видалення предмету є незворотнім процесом, та якщо він погоджується з цим, то повинен натиснути кнопку "Delete".

Для того, щоб відобразити розклад екзаменів, користувач має натиснути на кнопку "Exams" у боковій панелі. На екрані зображений розклад екзаменів для всіх груп в системі. Для кожного екзамену на сторінці "Exams" наявна кнопка "Edit" (Редагувати), яку можна використовувати для редагування даних екзамену, та кнопка "Remove" (Видалити), яку можна використовувати для видалення екзамену з бази даних.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДГОТОВКИ ТА РОБОТИ З РОЗКЛАДОМ ЗАНЯТЬ

Графічний матеріал

КПІ.ІХ-9230.045440.06.99

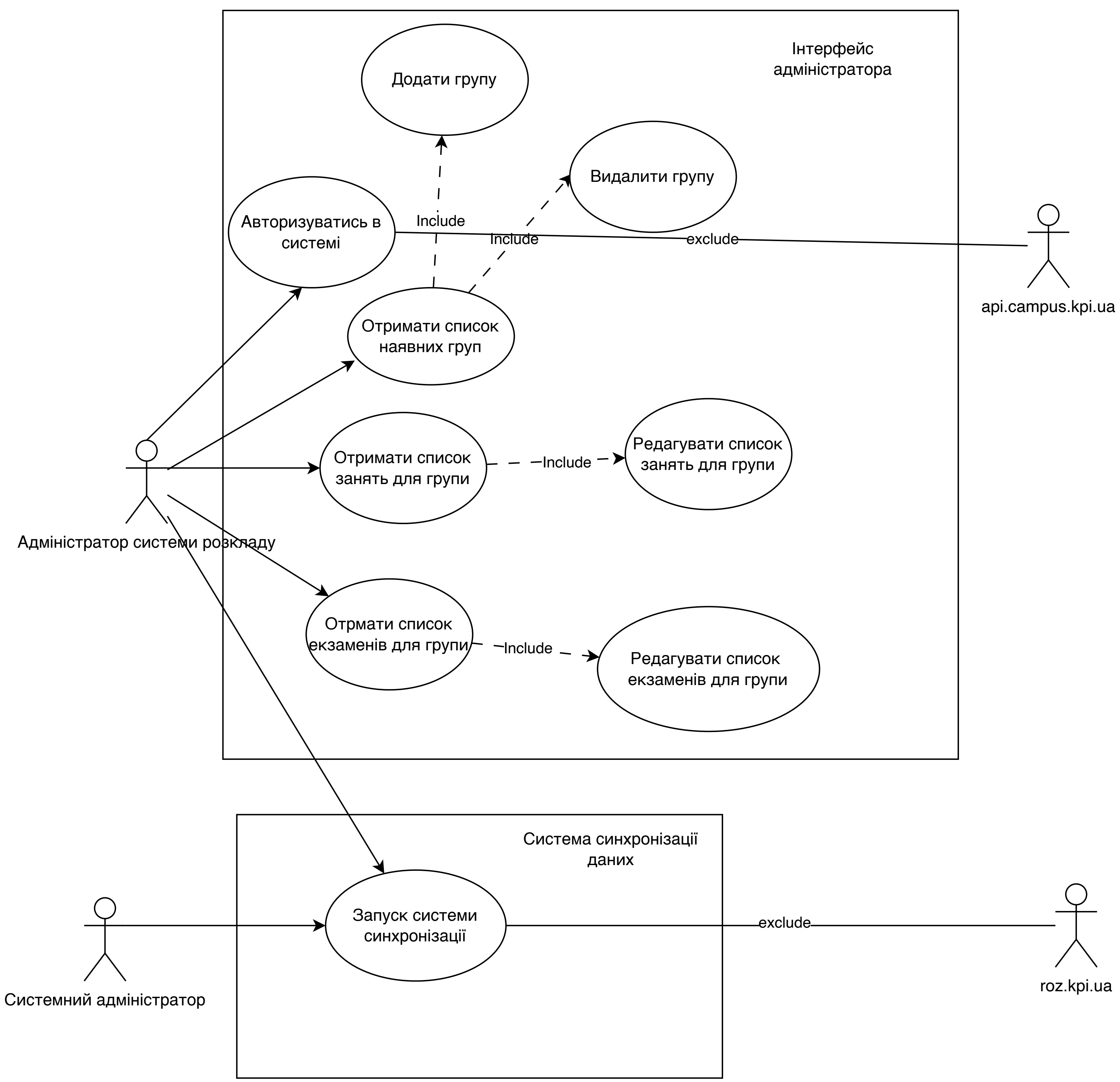
“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____Олександр ЯЦЕВСЬКИЙ

Київ – 2023

КПІ.ІП-9230.045440.06.99 ССВ



Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Яцевський О.І		
Перевірив		Фіногенов О.Д.		
Т. кон.				
Н. кон.		Вітковська І.І.		
Затвердив		Фіногенов О.Д.		

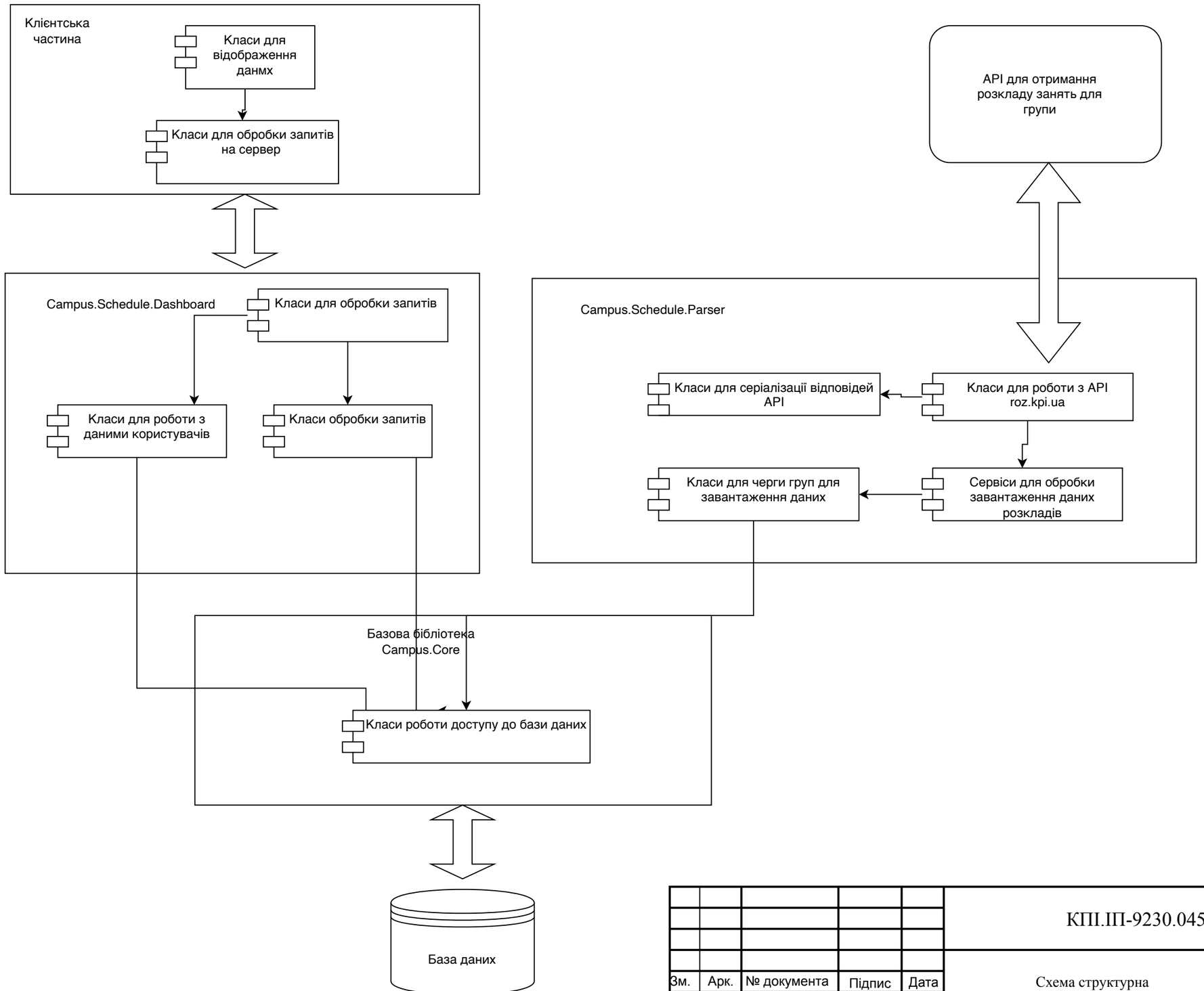
КПІ.ІП-9230.045440.06.99 ССВ

Діаграма варіантів використання

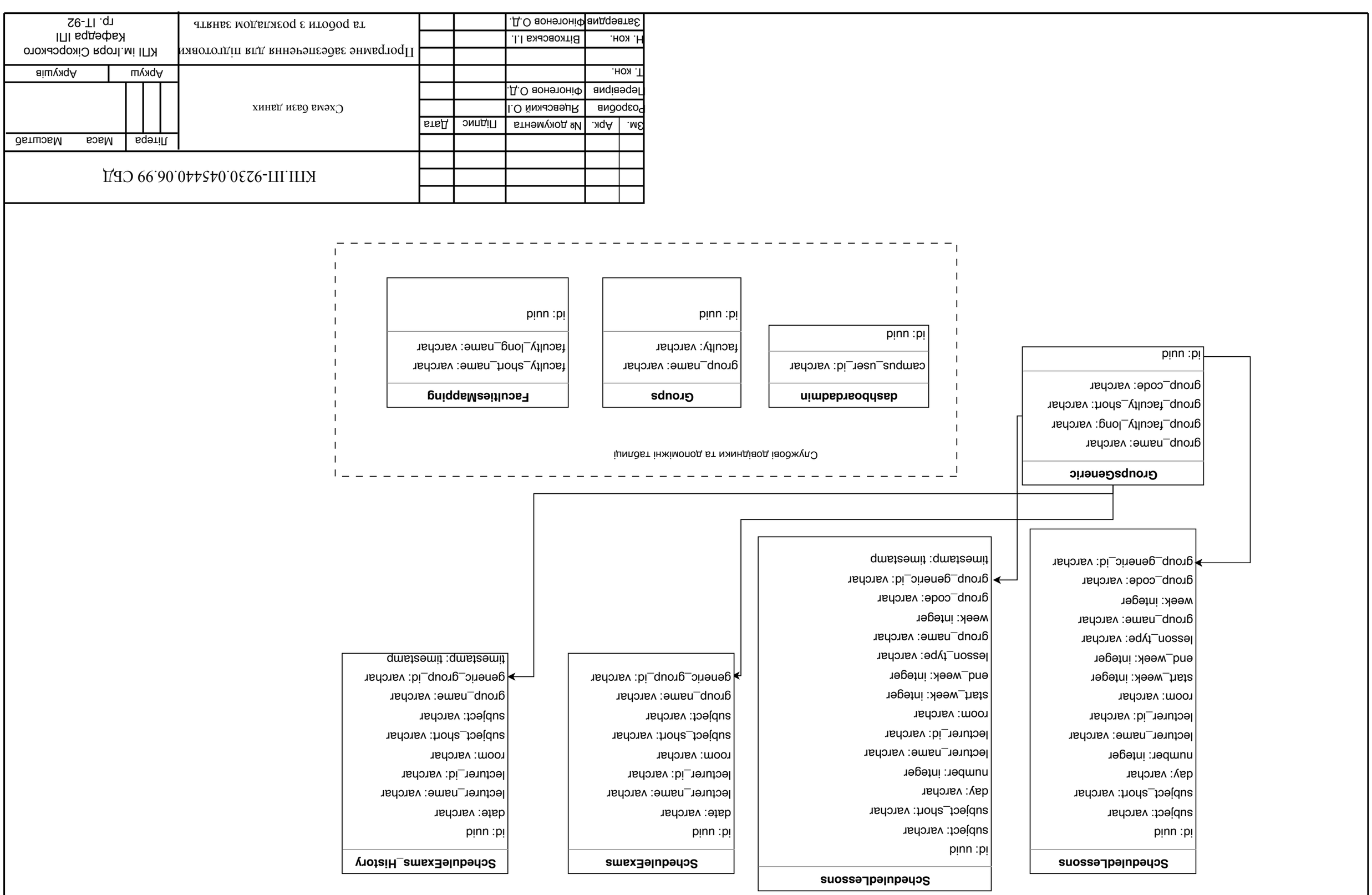
Літера	Маса	Масштаб
Аркуш	Аркушів	

Програмне забезпечення для підготовки та роботи з розкладом занять

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІТ-92



					КПІ.ІП-9230.045440.06.99 ССМ	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Яцевський О.І				
Перевірив		Фіногенов О.Д.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Вітковська І.І.			Програмне забезпечення для підготовки та роботи з розкладом занять	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-92
Затвердив		Фіногенов О.Д.				



КПІ.ІП-9230.045440.06.99 СБД

Зм.	Арк.	№ документа	Підпис	Дата	КПІ.ІП-9230.045440.06.99 СБД		
Розробив		Яцевський О.І.				Літера	Маса Масштаб
Перевірив		Фіногенов О.Д.			Схема бази даних		
Т. контр.						Аркуш	Аркушів
Н. контр.		Бітковська І.І.					
Затвердив		Фіногенов О.Д.			Програмне забезпечення для підготовки та роботи з розкладом занять	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-92	