

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 календарного планування
виконання ІТ-проєктів

Виконав студент IV курсу, групи ІІ-92
(шифр групи)
Медведєв Михайло Євгенович
(прізвище, ім'я, по батькові) _____ (підпис)

Керівник ст.викл., Халус О.А.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Консультант
з графічної
документації доцент, к.т.н., доц., Ліщук К.І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Рецензент д.т.н., проф., проф. каф ІСТ Теленик С.Ф.
(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

(підпис) Едуард ЖАРІКОВ
(ім'я прізвище)

“ ____ ” _____ 2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Медведєву Михайлу Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Програмне забезпечення для календарного планування виконання ІТ-проєктів

керівник проєкту Халус Олена Андріївна, ст.викл.
(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 використання

2) Схема структурна класів програмного забезпечення

3) Схема структурна компонентів програмного забезпечення

4) 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	18.03.2023	
3	Постановка та формалізація задачі	29.03.2023	
4	Розробка інформаційного забезпечення	29.03.2023	
5	Алгоритмізація задачі	30.03.2023	
6	Обґрунтування вибору використаних технічних засобів	30.03.2023	
7	Розробка програмного забезпечення	03.04.2023	
8	Налагодження програми	08.04.2023	
9	Виконання графічних документів	10.04.2023	
10	Оформлення пояснювальної записки	02.05.2023	
11	Подання ДП на попередній захист	24.05.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент

(підпис)

Михайло МЕДВЕДЄВ

(ініціали, прізвище)

Керівник

(підпис)

Олена ХАЛУС

(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 93 таблиці, 16 рисунків та 18 джерел – загалом 85 сторінки.

Дипломний проєкт присвячений розробці програмного забезпечення для створення календарних планувань та їх оптимізації за допомогою сучасних алгоритмів.

Метою розробки є спрощення створення ефективних розкладів за рахунок створення ефективного програмного забезпечення, що реалізує конструктивні алгоритми оптимізації багатоетапних задач календарного планування, що дозволяє підвищити ефективність роботи команд у IT-проєктах.

Розділ аналізу вимог до програмного забезпечення присвячений опису основних понять теорії розкладів, а також розроблених ПДС-алгоритмів для точного розв'язання розглянутих задач календарного планування. Окрім цього, у даному розділі було проаналізовано предметну область цільового програмного забезпечення та поставлено ряд функціональних та нефункціональних вимог.

У розділі моделювання та конструювання програмного забезпечення описано розробку архітектури програмного забезпечення, а також детальний опис кожної з частин отриманого застосунку.

Розділ аналізу якості та тестування програмного забезпечення присвячений аналізу якості програмного забезпечення отриманого додатку та опису процесу його тестування.

Розділ впровадження та супроводу програмного забезпечення зосереджений на процесі розгортання отриманого застосунку, а також на підтримці отриманого програмного забезпечення.

КЛЮЧОВІ СЛОВА: ТЕОРІЯ РОЗКЛАДІВ, КАЛЕНДАРНЕ ПЛАНУВАННЯ, ЕФЕКТИВНИЙ РОЗКЛАД, IT-ПРОЄКТ, ВЕБ-ЗАСТОСУНОК, БАЗА ДАНИХ.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 93 tables, 16 figures and 18 sources – in total 85 pages.

The diploma project is dedicated to the development of software for creating calendar plans and their optimization using modern algorithms.

The purpose of the development is to simplify the creation of effective schedules by creating effective software that implements constructive algorithms for optimizing calendar planning multi-stage tasks, which allows to increase the efficiency of teams in IT projects.

The software requirements analysis section is devoted to the description of the main concepts of the theory of schedules, as well as the developed PDS algorithms which can be used to find accurate solutions of the considered calendar planning problems. In addition, in this section the subject area of the target software was analyzed and a number of functional and non-functional requirements were set.

The software modeling and design section describes the development of the software architecture, as well as a detailed description of each part of the application.

The section on software quality analysis and testing is devoted to software quality analysis of the received application and a description of its testing process.

The software implementation and support section is focused on the process of deploying the resulting application, as well as supporting the resulting software.

KEYWORDS: SCHEDULE THEORY, CALENDAR PLANNING, EFFICIENT SCHEDULE, IT PROJECT, WEB APPLICATION, DATABASE.

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

Програмне забезпечення для календарного планування виконання

ІТ-проектів

Технічне завдання

КПІ.ПІ-9213.045490.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена ХАЛУС

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Михайло МЕДВЕДСВ

Київ – 2023

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ.....	3
2	ПІДСТАВА ДЛЯ РОЗРОБКИ.....	4
3	ПРИЗНАЧЕННЯ РОЗРОБКИ.....	5
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	5
4.1	Вимоги до функціональних характеристик	6
4.1.1	Для користувача:.....	6
4.1.2	Додаткові вимоги:.....	8
4.2	Вимоги до надійності	8
4.3	Умови експлуатації.....	8
4.3.1	Вид обслуговування	8
4.3.2	Обслуговуючий персонал	8
4.4	Вимоги до складу і параметрів технічних засобів	9
4.5	Вимоги до інформаційної та програмної сумісності	9
4.5.1	Вимоги до вхідних даних.....	9
4.5.2	Вимоги до вихідних даних	9
4.5.3	Вимоги до мови розробки.....	9
4.5.4	Вимоги до середовища розробки	10
4.5.5	Вимоги до представлення вихідних кодів	10
4.6	Вимоги до маркування та пакування.....	10
4.7	Вимоги до транспортування та зберігання	10
4.8	Спеціальні вимоги	10
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	11
5.1	Попередній склад програмної документації	11
5.2	Спеціальні вимоги до програмної документації	11
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ.....	12
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	13

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для календарного планування виконання ІТ-проектів.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення для створення календарних плануваль ІТ-проектів та їх оптимізації за допомогою сучасних алгоритмів. Програмне забезпечення призначене для менеджерів, обов'язками яких є планування виконання ІТ-проектів для підвищення ефективності роботи команди розробників.

					КПІ.ІП-9213.045490.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		3

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмного забезпечення для календарного планування виконання ІТ-проектів є завдання на дипломне проектування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

					КПІ.ІП-9213.045490.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для менеджерів, обов'язками яких є планування виконання ІТ-проєктів для підвищення ефективності роботи команди розробників.

Метою розробки є підвищення якості процесу планування виконання ІТ-проєктів за рахунок створення ефективних алгоритмів календарного планування та відповідного їм програмного забезпечення.

					КПІ.ІП-9213.045490.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Для користувача:

- реєстрація користувача за допомогою електронної пошти та пароллю;
 - 1) перевірка валідності електронної пошти;
 - 2) перевірка, чи є електронна пошта унікальною у системі;
 - 3) перевірка пароллю на відповідність поставленим обмеженням (пароль повинен мати щонайменше 7 символів, з щонайменше одну літеру верхнього регістру, одну літеру нижнього регістру та один спеціальний символ);
- авторизація користувача;
 - 1) авторизація користувача за допомогою електронної пошти та пароллю;
 - а) перевірка електронної адреси та пароллю;
 - 2) авторизація за допомогою сервісу Google;
 - а) автоматична реєстрація користувача при першій авторизації через Google;
- редагування особистих даних користувача;
 - 1) оновлення імені та прізвища користувача;
 - 2) оновлення пароллю користувача;
- створення нового розкладу;
 - 1) вибір типу нового розкладу;
 - 2) введення назви розкладу;
- збереження розкладу;
 - 1) перевірка створеного розкладу на наявність помилок;

- завантаження збереженого розкладу;
- редагування розкладу;
 - 1) введення нової назви розкладу;
- видалення розкладу;
- додавання задач за допомогою графічного інтерфейсу;
- редагування задач за допомогою графічного інтерфейсу;
 - 1) введення нової назви задачі;
 - 2) введення значення директивного терміну задачі;
- видалення задач за допомогою графічного інтерфейсу;
- додавання операцій до задач за допомогою графічного інтерфейсу;
- редагування операцій за допомогою графічного інтерфейсу;
 - 1) введення назви операції;
 - 2) введення значення тривалості операції;
- видалення операцій за допомогою графічного інтерфейсу;
- додавання рівнів розкладу за допомогою графічного інтерфейсу;
- редагування рівнів розкладу за допомогою графічного інтерфейсу;
 - 1) введення нової назви рівня;
- видалення рівнів розкладу за допомогою графічного інтерфейсу;
- додавання приладів за допомогою графічного інтерфейсу;
- редагування приладів за допомогою графічного інтерфейсу;
 - 1) введення нової назви приладу;
 - 2) введення значення моменту початку роботи приладу;
- призначення операцій приладам за допомогою графічного інтерфейсу;
- переміщення операцій між задачами за допомогою графічного інтерфейсу;
- оптимізація розкладів;

- 1) оптимізація багаторівневих розкладів за допомогою швидкого алгоритму;
- 2) оптимізація багаторівневих розкладів за допомогою комплексного алгоритму;
- 3) оптимізація однорівневих розкладів за допомогою алгоритму з автоматичним розподілом операцій між пристроями.

4.1.2 Додаткові вимоги:

- Графічний інтерфейс веб-додатку повинен бути зрозумілим для користувача;
- Застосунок повинен працювати у більшості сучасних веб-браузерів.

4.2 Вимоги до надійності

Передбачити перевірку введеної інформації. Передбачити захист від некоректних дій користувача. Передбачити збереження цілісності інформації в базі даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Обслуговуючий персонал даного програмного забезпечення повинен вміти проводити такі операції, як підключення до платформи хостингу та запуск скриптів у командному рядку.

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: 4 Гб;
- підключення до мережі Інтернет зі швидкістю від 10 мегабіт.

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i9;
- об'єм ОЗП: 16 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows XP, Windows NT і т.д.) або Unix.

4.5.1 Вимоги до вхідних даних

Особливі вимоги до вхідних даних не висуваються.

4.5.2 Вимоги до вихідних даних

Особливі вимоги до вихідних даних не висуваються.

4.5.3 Вимоги до мови розробки

Розробку веб-застосунку виконати мовою програмування TypeScript, клієнтську частину веб-додатку реалізувати за допомогою бібліотеки React.js, для серверної частини використати фреймворк Nest.js. Оптимізаційні інструменти реалізувати за допомогою мови програмування Python.

					КПІ.ІП-9213.045490.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		9

4.5.4 Вимоги до середовища розробки

Розробку виконати за допомогою середовища розробки Visual Studio Code.

4.5.5 Вимоги до представлення вихідних кодів

Вихідні коди повинні бути представлені у вигляді текстових файлів.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред'являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

					КПІ.ІП-9213.045490.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		10

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.

5.2 Спеціальні вимоги до програмної документації

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

					КПІ.ІП-9213.045490.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		11

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	19.03	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.03	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.04	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.04	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.04	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.04	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

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

					КПІ.ІП-9213.045490.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		13

Пояснювальна записка до дипломного проєкту

на тему: Програмне забезпечення для календарного планування виконання
ІТ-проєктів

КПІ.ПІ-9213.045490.02.81

Київ – 2023

ЗМІСТ

ЗМІСТ	2
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	4
ВСТУП	5
1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	7
1.1 Загальні положення	7
1.2 Змістовний опис і аналіз предметної області	9
1.3 Аналіз існуючих технології та успішних ІТ-проектів	16
1.4 Аналіз вимог до програмного забезпечення	24
1.5 Постановка задачі	42
Висновки до розділу	44
2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	45
2.1 Моделювання та аналіз програмного забезпечення.....	45
2.2 Архітектура програмного забезпечення.....	46
2.3 Конструювання програмного забезпечення.....	55
2.4 Аналіз безпеки даних	58
Висновки до розділу	60
3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 61	
3.1 Аналіз якості ПЗ.....	61
3.2 Опис процесів тестування.....	61
3.3 Опис контрольного прикладу	75
Висновки до розділу	78
4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .	79
4.1 Розгортання програмного забезпечення.....	79
4.2 Підтримка програмного забезпечення.....	80
Висновки до розділу	81
ВИСНОВКИ.....	82

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 83

ДОДАТОК А ЗВІТ ПОДІБНОСТІ..... 85

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

BPMN	– Business Process Model and Notation – Модель і нотація бізнес-процесів
MPMM	– Мінімізація різниці між мінімальним та максимальним моментами закінчення роботи приладами
ДОО	– Допустима ознака оптимальності
ПДС	– Алгоритм з поліноміальною та декомпозиційною складовою
SSL	– Secure Sockets Layer
HTTPS	– HyperText Transfer Protocol Secure
ТР	– Теорія розкладів
ПЗ	– Програмне забезпечення

					КПІ.ІП-9213.045490.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

ВСТУП

Поняття розкладу (календарного плану, графіку роботи і т.д.) знає кожна людина з дитинства. Багато речей, які відбуваються у житті як дитини, так і дорослої особи, пов'язані з плануванням цих подій та розподілі часу. Для того, щоб все встигати, дітей змалку вчать правильно розподіляти свою роботу у часі. Кожна людина у нашому світі фактично кожного дня створює свій графік роботи на основі своїх потреб та планів інших людей.

Деякі ж люди мають створювати календарні планування за обов'язком своєї робочої діяльності. Такі розклади є надважливими у багатьох галузях, оскільки вони забезпечують неперервну та максимально продуктивну роботу багатьох людей чи підприємств. До таких галузей можна віднести планування руху транспорту, роботу заводів, проведення навчання, планування розробки ІТ-проєктів тощо.

Теорія розкладів – це єдина наука, яка вивчає задачі на розподілення завдань у часі за умови, що він є обмеженим, а також методи створення оптимальних календарних планувань. Запропоновані методи даною наукою знаходять своє застосування не тільки в системах управління по виробництву матеріалів, а й у розгалужених системах, які розподіляють час та ресурси конструкторських бюро та центрів обчислювальних потужностей.

Розвиток теорії розкладів характеризувався розробкою ефективних алгоритмів для складання оптимальних календарних планувань за чітко поставленими критеріями. Оскільки задача створення розкладу є NP-складною, створені алгоритми можуть значно прискорити процес створення та оптимізації графіків роботи. Однак, попри те, що постановки задач побудови календарних планувань різних типів є доволі простими, сьогодні існують лише декілька алгоритмів, які можуть розв'язувати дані задачі з достатньою точністю.

Автоматизація створення розкладів дозволяє значно спростити планування проєктів та підвищити продуктивність працівників, однак сучасні

					КПІ.ІП-9213.045490.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

платформи для календарного планування є платними та не мають в собі інструментів для оптимізації отриманих розкладів виконання проєктів.

Тому темою даної роботи є реалізація застосунку з графічним інтерфейсом задля спрощення створення календарних планувань виконання ІТ-проєктів. Дана розробка матиме можливість застосовувати сучасні ПДС-алгоритми для автоматичної оптимізації розкладів та зберігати їх для подальшого використання.

					КПІ.ІП-9213.045490.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		6

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Теорія розкладів базується на декількох основних поняттях, з яких вона складається. Такими поняттями є операція, задача, прилад, при цьому фізична природа даних об'єктів не є важливою[1].

Найменшою одиницею у розкладі є операція – деяка дія, яка має свою тривалість (довжину) та належить до деякої задачі. Задача ж є набором операцій у деякій послідовності з чітко встановленим директивним строком, при цьому кожна операція у задачі має бути виконана не раніше попередньої операції даної задачі. Директивним строком останньої операції у задачі є директивний строк всієї задачі.

Прилад є виконавцем набору операцій у деякому порядку за умови, що, почавши виконання першої своєї операції, він повинен робити всі операції по черзі без пауз.

Основною характеристикою задач теорії розкладів є цільова функція (критерій оптимальності), значення якої повинно бути мінімізоване або максимізоване на множині директивних термінів. В задачах ТР дана цільова функція може бути знайдена за допомогою заданого набору штрафних функцій, які підраховуються після створення розкладу з розподіленими у часі та між приладами операціями[3].

Можливі варіанти цільових функцій:

- C_j – момент завершення обробки операції j ;
- $L_j = C_j - d_j$ – часове відхилення;
- $T_j = \max\{0; C_j - d_j\}$ – запізнення;
- $E_j = \max\{0; d_j - C_j\}$ – випередження;
- U_j – показує, чи запізнюється виконання операції j , тобто дорівнює 1, якщо виконання запізнюється ($C_j > d_j$), і дорівнює 0, якщо завдання не запізнюється ($C_j \leq d_j$).

В теорії розкладів завдання можна розрізняти за наступними критеріями:

- за способом задання вхідної інформації[2]:
 - 1) детерміновані (в початкову систему надходить деяка фіксована кількість операцій (робіт), після чого на основі початкових даних будується розклад);
 - 2) динамічні (операції можуть надходити у систему у будь-який час після початку створення розкладу);
- за типом розв’язку, має бути знайдений[4]:
 - 1) задачі упорядкування (розподіл операцій задано заздалегідь між всіма приладами, визначені необхідні параметри робіт, задачею є складення розкладу, у якому всі операції у всіх приладах розміщені у оптимальному порядку та для них задані такі параметри, як початок чи кінець виконання операції);
 - 2) задачі узгодження (вибір тривалості операцій, моментів надходження операцій у розклад та інших параметрів);
 - 3) задачі розподілу (знаходження оптимального розподілу операцій по всім приладам; в контексті даної задачі вважається, що операції можуть виконуватись різними виконавцями);
- за типом цільової функції[3]:
 - 1) задачі з сумарними критеріями оптимізації (цільова функція направлена на мінімізацію чи максимізацію суми деяких значень частин розкладу);
 - 2) задачі з мінімаксними критеріями оптимізації (цільова функція направлена на мінімізацію лише максимального значення);
 - 3) багатокритеріальні задачі оптимізації (задачі такого типу мають декілька цільових функцій);
 - 4) задачі на побудову допустимого розкладу (створення допустимого календарного планування, при якому всі умови допустимості будуть виконані);

- за способом зображення календарних планувань[3]:
 - 1) табличне зображення;
 - 2) графічне зображення за допомогою діаграми Ганта;
 - 3) векторне зображення розкладу за допомогою вектору (вказується лише порядок виконання завдань).

1.2 Змістовний опис і аналіз предметної області

Оскільки дана розробка передбачає створення та оптимізацію календарних планувань різних типів для ІТ-проектів, потрібно провести аналіз складових частин даних розкладів.

Оскільки розклад у загальному визначенні складається із задач, операцій, приладів та рівнів, слід визначити, що відображатимуть дані об'єкти в області ІТ-проектів. Операцією у таких проєктах є задачі програмування, створення дизайну, знаходження та виправлення помилок у коді тощо. Задачею ж є деякий набір завдань для команди, який демонструє деякий функціонал, що повинен бути доданий до продукту.

Аналогами приладів у ІТ-проектах будуть програмісти, дизайнери, тестувальники та інші працівники, які беруть участь у роботі над проєктом.

Рівні в розкладі відповідатимуть за набір працівників одного рівня кваліфікації та відповідальності на проєкті. Більш відповідальні операції будуть виконані більш кваліфікованими спеціалістами на перших рівнях. До такого типу завдань можна віднести створення архітектури програмного забезпечення, налаштування загальних конфігурацій майбутнього проєкту, розробка моделі бази даних тощо. Менш відповідальні операції будуть виконуватись на останніх рівнях працівниками, які несуть значно меншу відповідальність. До таких завдань можна віднести вирішення невеликих проблем у коді, виправлення недоліків у дизайні, реалізація невеликих частин функціоналу тощо.

У розкладі з математичної точки зору час є числом, яке відображає положення деякого моменту відносно початку часу. При створенні

					КПІ.ІП-9213.045490.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		9

календарних планувань для ІТ-проектів час можна вимірювати від створення самого розкладу, а детальність часу у розкладі задається в залежності від характеристик створеного розкладу.

У даній роботі розглядатимуться алгоритми по оптимізації задач двох типів. Перша задача має наступну постановку: задана множина незалежних робіт $J = \{1, 2, \dots, n\}$, кожна з яких складається з однієї роботи. Для кожної задачі j відомі її тривалість виконання p_j та директивний термін d_j . Переривання виконання робіт недопустимо. Всі роботи потрапляють у систему одночасно перед формуванням розкладу. Процес виконання робіт є неперервним: після виконання першої операції приладом він повинен виконувати всі інші операції згідно розподілу. Необхідно знайти допустимий розклад, у якому сумарне випередження моментів закінчення виконання операцій відносно директивних термінів приймає мінімальне значення.

Для вирішення поставленої задачі можуть використовуватись інноваційні ПДС-алгоритми, такі як А, А1 та А2.

Алгоритм А має наступний вигляд:

- побудувати розклад $\sigma = \{1, 2, \dots, n\}$, в якому завдання відсортовані у порядку неспадання їх директивних термінів;
- визначити момент запуску виконання завдань $r = d_{max} - \sum_{i=1}^n l_i$, де $d_{max} = \max_{j \in J} d_j$. Визначити значення запізнення всіх завдань у отриманому розкладі;
- якщо немає завдань, які запізнюються – кінець алгоритму. Інакше визначити завдання l з максимальним значенням запізнення і новий момент початку виконання завдань $r_{max} = r - T_l$. Скласти новий розклад відносно нового моменту початку виконання операцій приладом.

Покращенням даного алгоритму є ПДС-алгоритм А1, який складається з наступних кроків:

- 1) побудувати початковий розклад за допомогою алгоритму А;
- 2) серед завдань, директивні терміни яких більші або дорівнюють моменту закінчення роботи всіх завдань, обрати завдання з найменшою тривалістю та поставити обране завдання на останнє місце у списку операцій на виконання приладом:

$$l_{j_n} = \min_{s/d_s \geq \sum_{j=1}^n l_j + r} \{l_s\}$$

- 3) перейти до позиції $i = n - 1$;
 - 4) визначити завдання, яке буде виконуватись на позиції i ;
- знайти момент закінчення роботи завдання j_i :

$$C_{j_n} = \sum_{j=1}^n l_j - \sum_{l=0}^{n-i-1} l_{j_{n-1}} + r$$

- з множини завдань, для яких директивні терміни не менше моменту закінчення роботи завдання j_i , обрати завдання з мінімальною тривалістю:

$$l_{j_n} = \min_{s/d_s \geq \sum_{j=1}^n l_j - \sum_{l=0}^{n-i-1} l_{j_{n-1}} + r} \{l_s\}$$

- розмістити завдання j_i на позицію i ;
- 5) перейти на наступну позицію: $i = i - 1$;
 - 6) якщо $i = 0$ – кінець алгоритму, інакше перейти на крок 4.

Алгоритми А та А1 ефективно розв'язують поставлену задачу, отримуючи як результат календарне планування, початок роботи приладу в якому завжди дорівнює при $r = r_{max}$. Однак не завжди розклади за такої умови є оптимальними.

Існує й інший алгоритм, який розв'язує поставлену задачу у загальному випадку. Алгоритм А2 дозволяє знаходити оптимальний розклад за критерієм мінімізації сумарного випередження.

Алгоритм А2 складається з наступних кроків:

- 1) побудувати розклад за допомогою алгоритму А1 для $r = r_{max}$;
- 2) побудувати розклад за допомогою алгоритму А1 для $r = 0$;
- 3) визначити сумарні випередження отриманих розкладів $E(r_{max})$ та $E(0)$ відповідно;
- 4) якщо $E(r_{max}) + nr_{max} = E(0)$ – то отриманий перший розклад є оптимальним, кінець алгоритму;
- 5) інакше $l = 0$ – порядковий номер зміни структури розкладу;
- 6) $l = l + 1$;
- 7) знайти довжину l -того інтервалу за наступною формулою:

$$\Delta_l = \min_{2 \leq i \leq n} \{C_{ji} - \max_{\substack{s \\ p_s < p_{ji} \\ d_s < C_{ji}}} \{d_s\}\}$$

- 8) перемістити вліво поточну послідовність робіт на довжину Δ_l ;
- 9) побудувати новий розклад за допомогою алгоритму А1 при новому значенні моменту початку роботи приладу;
- 10) якщо не виконується рівність $E(r_{max}) + nr_{max} = E(0)$ – повернутись до кроку 6;
- 11) серед отриманих розкладів на всіх кроках знайти розклад з найменшим значенням сумарного випередження.

Всі описані алгоритми можуть ефективно розв'язувати задачу для одного пристрою та набору операцій. Дані алгоритми можуть бути також використані у багатоетапній задачі календарного планування. Розглянемо її загальну постановку.

Задана множина незалежних задач $J_1, J_2 \dots J_n$, кожна з яких складається з p операцій. Кожна задача має свій директивний термін $d_i, i = \overline{1, n}$. Задані також тривалості виконання кожної операції у задачах $l_i^k, i = \overline{1, n}, k = \overline{1, p}$. Також задана множина приладів на p рівнях, які можуть виконувати операції лише свого рівня. При цьому прилади k -того рівня отримують свої операції

від $k - 1$ рівня. Необхідно створити оптимальне календарне планування за критерієм мінімізації сумарного випередження.

Технологічний процес p -рівневої системи для даної поставленої задачі зображений на рисунку 1.1.

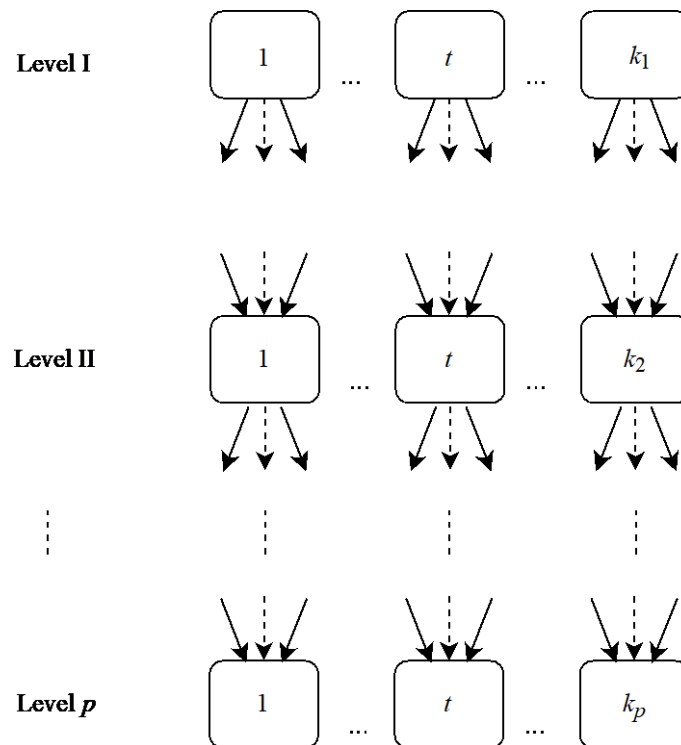


Рисунок 1.1 – Структура процесу p -рівневої системи

Алгоритм оптимізації багаторівневого розкладу має наступний вигляд[7]:

- 1) $i = p - 1$ – порядковий номер оптимізованого рівня;
- 2) кожен пристрій на i -тому рівні необхідно оптимізувати за допомогою алгоритму A2 або A1;
- 3) якщо $i = 0$ (всі рівні оптимізовано) – кінець алгоритму. Інакше визначити директивні терміни операцій $i - 1$ рівня як моменти початку виконання операцій i -того рівня їх задач;
- 4) $i = i - 1$, повернутись до кроку 2.

У теорії розкладів мають місце не лише задачі оптимізації одного приладу за призначеними на нього операціями та їх директивними строками. Розглянемо задачу теорії розкладів для оптимізації і розподілу операцій другого типу.

Задана множина незалежних завдань $J = \{1, 2, \dots, n\}$, множина приладів однакової продуктивності $I = \{1, 2, \dots, m\}$, для кожного завдання відома тривалість його виконання. Моменти запуску приладів $s_i \geq 0, i \in I$ можуть не співпадати. Після початку роботи прилад повинен виконати всі операції у призначеному порядку без зупинок. Необхідно побудувати календарне планування за критерієм мінімізації різниці між максимальним та мінімальним моментами закінчення виконання завдань приладами (MPMM).

Для побудови оптимального календарного планування за поставленою задачею використовується алгоритм А0, який складається з двох частин – побудову початкового розкладу та ітеративну оптимізацію отриманого розкладу. Розглянемо детально кожен з них.

Алгоритм побудови початкового розкладу σ_0 складається з наступних кроків[8]:

- 1) відсортувати список всіх завдань з множини J у порядку незростання;
- 2) відсортувати прилади у порядку неспадання значень початку їх роботи s_i ;
- 3) встановити час звільнення приладів як $C_i = s_i \quad \forall i = \overline{1, m}$;
- 4) обрати завдання з максимальної тривалістю виконання з непризначених завдань та призначити на прилад з мінімальним значенням моменту звільнення приладу;
- 5) визначити новий час звільнення приладу $C_i = C_i + l_j$;
- 6) якщо всі завдання розподілені між приладами – кінець алгоритму; інакше повернутись на крок 4.

Другий етап алгоритму А0 є послідовною оптимізацією отриманого розкладу за рахунок перестановок, які сприяють покращенню критерію МРММ розкладу. Всі можливі перестановки зображені у таблицях 1.1 та 1.2.

Таблиця 1.1 – Властивості перестановок типу А

Тип перестановки	Прилади, які приймають участь у перестановці		θ ($\theta > 0$)	Умова, при якій виконується перестановка	Характеристики результуючого розкладу σ^1
	h	s			
1-1А	$h \in I_{\Delta}(\sigma),$ $j_1 \in J_h$	$s \in I_R(\sigma),$ $j_2 \in J_s$	$l_{j_1} - l_{j_2}$	$\theta \leq \Delta_h(\sigma),$ $\theta \leq R_s(\sigma)$	$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta,$ $R_s(\sigma^1) = R_s(\sigma) - \theta.$
1-2А	$h \in I_{\Delta}(\sigma),$ $j_1 \in J_h$	$s \in I_R(\sigma),$ $j_2, j_3 \in J_s$	$l_{j_1} - (l_{j_2} + l_{j_3})$		
2-1А	$h \in I_{\Delta}(\sigma),$ $j_1, j_2 \in J_h$	$s \in I_R(\sigma),$ $j_3 \in J_s$	$(l_{j_1} + l_{j_2}) - l_{j_3}$		
2-2А	$h \in I_{\Delta}(\sigma)$ $j_1, j_2 \in J_h$	$s \in I_R(\sigma),$ $j_3, j_4 \in J_s$	$(l_{j_1} + l_{j_2}) - (l_{j_3} + l_{j_4})$		

Таблиця 1.2 – Властивості перестановок типу Б

Тип перестановки	Прилади, які приймають участь у перестановці		θ ($\theta > 0$)	Умова, при якій виконується перестановка	Характеристики результуючого розкладу σ^1
	h	s			
1-1Б	$h \in I_{\Delta}(\sigma),$ $j_1 \in J_h$	$s \in I_R(\sigma),$ $j_2 \in J_s$	$l_{j_1} - l_{j_2}$	$\theta \leq \Delta_h(\sigma),$ $\theta > R_s(\sigma)$	$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta,$ $\Delta_s(\sigma^1) = \theta - R_s(\sigma)$
1-2Б	$h \in I_{\Delta}(\sigma),$ $j_1 \in J_h$	$s \in I_R(\sigma),$ $j_2, j_3 \in J_s$	$l_{j_1} - (l_{j_2} + l_{j_3})$		
2-1Б	$h \in I_{\Delta}(\sigma),$ $j_1, j_2 \in J_h$	$s \in I_R(\sigma),$ $j_3 \in J_s$	$(l_{j_1} + l_{j_2}) - l_{j_3}$		
2-2Б	$h \in I_{\Delta}(\sigma),$ $j_1, j_2 \in J_h$	$s \in I_R(\sigma),$ $j_3, j_4 \in J_s$	$(l_{j_1} + l_{j_2}) - (l_{j_3} + l_{j_4})$		

Для опису алгоритму А0 введемо додаткові допоміжні позначення:

$$C_{cp} = \frac{\sum_{j=1}^n l_j + \sum_{i=1}^m s_i}{m};$$

$\Delta_i(\sigma) = \max\{0, C_i(\sigma) - C_{cp}\}$ – запізнення приладу $i \in I$;

$R_i(\sigma) = \max\{0, C_{cp} - C_i(\sigma)\}$ – випередження приладу $i \in I$;

$I_{\Delta}(\sigma)$ – множина приладів розкладу σ , на яких $\Delta_i(\sigma) > 0$;

$I_R(\sigma)$ – множина приладів розкладу σ , на яких $R_i(\sigma) > 0$.

Для визначення оптимальності розкладу вводиться допустима ознака оптимальності (ДОО), яка має наступний вигляд: якщо $C_i(\sigma) = \text{const } \forall i \in I$, то розклад σ є оптимальним.

Алгоритм виконання другого етапу алгоритму A_0 складається з наступних кроків[8]:

- 1) побудувати початковий розклад σ_0 , $\sigma = \sigma_0$;
- 2) визначити множини $I_\Delta(\sigma)$ та $I_R(\sigma)$;
- 3) перевірка виконання ДОО для задачі МРММ; якщо дана ознака виконується – розклад є оптимальним, кінець алгоритму;
- 4) визначити прилад h , у якого значення запізнення є максимальним;
- 5) для приладу h , перебираючи прилади $s \in I_R(\sigma)$, виконати перестановку типу А чи типу Б. Якщо знайшлась хоча б одна така перестановка – повернутись до кроку 2;
- 6) отриманий розклад σ є оптимальним, кінець алгоритму.

1.3 Аналіз існуючих технології та успішних ІТ-проектів

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Теорія розкладів містить у собі велику кількість різних задач календарного планування, які базуються на встановлених критеріях. Найпопулярнішим критерієм у таких задачах є мінімізація сумарного запізнення роботи приладами. У таких задачах прилади можуть виконувати операції після встановлених директивних термінів, а метою алгоритмів, що розв'язують подібні задачі, є зменшення запізнення всіма приладами. При цьому критерій оптимальності може бути сформованим як мінімізація сумарного зваженого запізнення приладами виконання задач.

Для застосування алгоритмів оптимізації з теорії розкладів на виробництво було також розроблено деякі багатоетапні задачі календарного планування, які включають в себе різні методи оптимізації на різних рівнях розкладу. Такі задачі допомагають моделювати та аналізувати процеси таких

видів виробництва, як «під ключ», робочий цех, будівничі виробництва, системи планування та управління проектами. Виробництва такого типу відносяться до виробничих систем з мережевим представленням технологічних процесів і обмеженими ресурсами. Системи такого типу мають такі особливості[6]:

- велика кількість різних типів приладів;
- значна різниця між технологічними маршрутами виробів;
- конструктивна складність номенклатури виробів;
- наявність циклів у процесі виробництва;
- необхідність управління виготовленнями запасних виробів;
- обмеженість ресурсів, необхідних для виконання поставлених задач у часі.

Багатоетапні задачі календарного планування розв’язуються одразу багатьма сучасними алгоритмами, при чому дані алгоритми використовуються як для часткових задач планування на всіх частинах розкладу, так і для глобальної оптимізації отриманого розкладу.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Найпопулярнішою мовою програмування для імплементації алгоритмів на основі математичної бази є Python. Дана мова має велику кількість вбудованих методів та інструментів для проведення аналізу даних та написання високоефективних алгоритмів. До того ж сьогодні більшість бібліотек для роботи з великими об’ємами даних, автоматизації процесів і т.д. написана саме для використання у мові Python[11].

Для швидкої роботи з даними та імплементації високоефективних алгоритмів сьогодні також використовуються такі мови, як C++ та R. Основні відмінності розглянутих мов програмування наведені у таблиці 1.3.

					КПІ.ІП-9213.045490.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		17

Таблиця 1.3 – Відмінності мов програмування Python[11], C++[12] та R[13]

Характеристика	Python	C++	R
Легкість у використанні мови	Мова достатньо легка у використанні	Мова є досить складною для вивчення та роботи	Синтаксис мови є дуже специфічним, мова є доволі складною
Швидкість роботи	Код програми виконується по одному рядку, тому швидкість виконання є меншою за аналоги, однак достатньою для роботи з простими математичними обчисленнями	Оскільки мова є достатньо низькорівневою, швидкість роботи є надзвичайно високою	Швидкість виконання програми є достатньо високою
Вбудований функціонал	Мова має широкий вибір вбудованих функцій для зручної роботи з даними	Мова має достатньо невеликий об'єм інструментів для роботи з даними	Мова має дуже великий набір вбудованих методів та інструментів

Продовження таблиці 1.3

Характеристика	Python	C++	R
Сторонні бібліотеки	Існує багато бібліотек для роботи з даною мовою, більшість з них підтримуються і зараз	Має достатньо бібліотек, однак більшість з них є достатньо старими та не підтримуються	Мова має дуже мало сторонніх бібліотек для розширення функціоналу мови
Документація та підтримка	Мова має значну підтримку як зі сторони розробників мови, так і зі сторони спільноти	Мова має невелику документацію від розробників	Мова має достатню документацію і значну підтримку спільнотою

Зважаючи на факт, що для імплементації та роботи з розглянутими алгоритмами програма не потребує великих обчислювальних можливостей, було прийняте рішення обрати мову програмування Python для реалізації оптимізаційних інструментів для застосунку.

Серед інструментів для створення клієнтської частини веб-застосунку найпопулярнішими сьогодні є бібліотека React.js та фреймворки Angular та Vue. Дані інструменти якнайкраще підходять для створення односторінкових застосунків – веб-сайтів спеціального типу, у яких верстка не завантажується повністю окремо для кожної сторінки, а генерується завдяки коду, написаному на мові програмування JavaScript. Основні відмінності даних бібліотек та фреймворків описано у таблиці 1.4.

Таблиця 1.4 – Відмінності React.js[15], Angular[16] та Vue[17]

Характеристика	React.js	Angular	Vue
Структура коду застосунку	Не має фіксованої структури коду	Має строгу структуру коду з модулів, компонентів та сервісів	Має мінімальні вимоги до структури застосунку
Можливість статичної типізації	На вибір розробника	Завжди обов'язкова	На вибір розробника
Підтримка	Бібліотека має високий рівень підтримки як зі сторони розробників, так і спільнотою	Фреймворк має достатній рівень підтримки розробниками	Фреймворк у більшості підтримується спільнотою
Легкість у використанні мови	Легка у вивченні і використанні	Має дуже багато вбудованих інструментів, тому вивчення фреймворку займає багато часу	Достатньо легкий у вивченні фреймворк, однак більшість джерел інформації для вивчення написана китайською мовою

Проаналізувавши всі переваги та недоліки, було обрано саме React.js у якості бібліотеки для створення клієнтської частини веб-застосунку та мову програмування TypeScript для статичної типізації.

Для створення серверної частини веб-застосунку потрібно синхронізувати всі наявні типи на клієнтській частині з типами серверної частини, і коли дані частини написані на різних мовах – відбувається дублювання коду. Однак, якщо серверна частина буде написана також на мові програмування TypeScript, більшість типів клієнтської частини можна буде перевикористати у серверній частині. Саме тому було обрано імплементація серверу на основі Node.js та з використанням мови програмування TypeScript.

Для двигуна Node.js існує достатньо велика кількість бібліотек для реалізації веб-серверу, таких як Express.js, Koa, Fastify та Nest.js. Для реалізації серверної частини веб-застосунку було обрано використання саме Nest.js за наступними критеріями[18]:

- достатньо строга структура коду та чітко визначений стиль коду;
- велика кількість інструментів для ефективної роботи зі статичною типізацією мови програмування TypeScript;
- використання інтерфейсу командного рядка для роботи з кодом застосунку;
- легкість у збірці та розгортанні серверу.

1.3.3 Аналіз відомих програмних продуктів

Існують аналоги програмного забезпечення, що реалізують проведення календарного планування виконання ІТ-проектів, такі як Jira[9] та Trello[10]. Дані сервіси використовуються для створення задач, їх опису та призначення між виконавцями – програмістами, дизайнерами, тестувальниками тощо. Приклади роботи відповідних платформ можна побачити на рисунках 1.2-1.3.

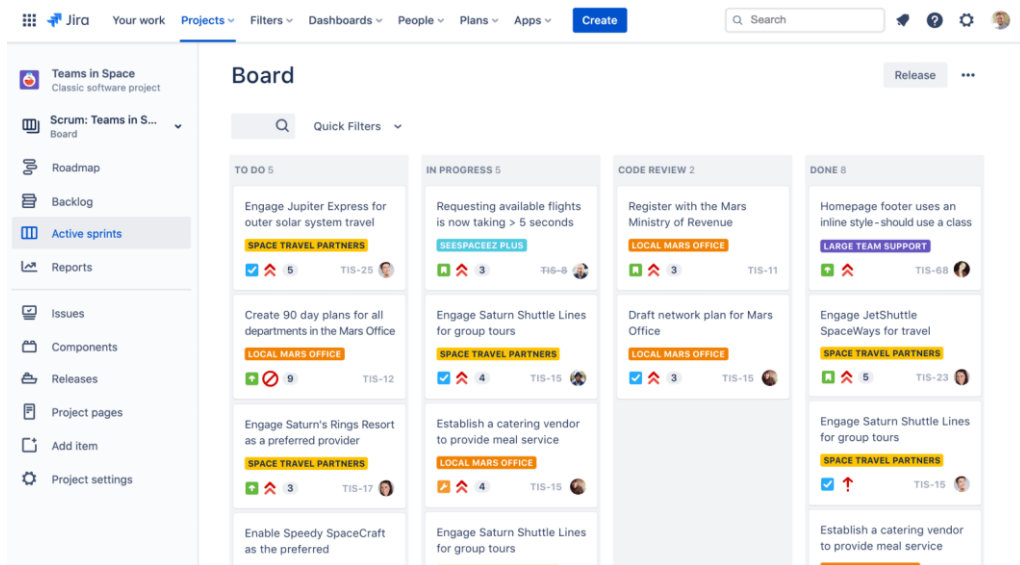


Рисунок 1.2 – Головне вікно Jira[9]

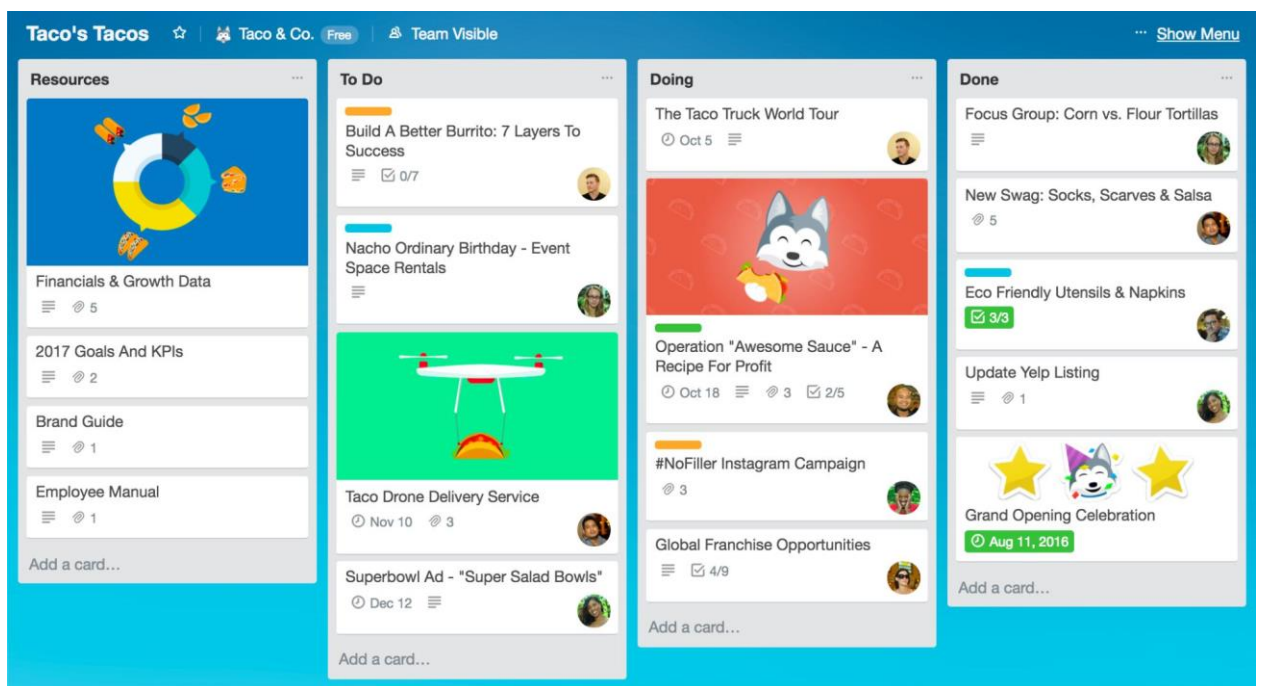


Рисунок 1.3 – Головне вікно Trello[10]

Порівняння функціоналу Jira, Trello та даного дипломного проєкту наведено у таблиці 1.5.

Таблиця 1.5 – Порівняння роботи Jira, Trello та дипломного проєкту

Характеристика	Jira	Trello	Дипломний проєкт
Створення розкладів	Може створювати розгалужені розклади та гнучко описувати характеристики його частин	Може створювати прості календарні планування	Може створювати однорівневі та багаторівневі розклади, а також гнучко змінювати характеристики всіх його частин
Взаємодія між користувачами	Менеджери у системі можуть надсилати завдання та призначати їх на інших виконавців	Всі користувачі можуть призначати завдання один одному	Користувачі можуть редагувати лише свої розклади, функціоналу для взаємодії декількох користувачів над одним розкладом немає
Вхід у систему	Лише за допомогою електронної пошти та пароллю	Лише за допомогою електронної пошти та пароллю	Можлива авторизація як за допомогою електронної пошти та пароллю, так і за допомогою сервісу Google

Продовження таблиці 1.5

Характеристика	Jira	Trello	Дипломний проєкт
Оптимізація розкладів	Немає інструментів для оптимізації розкладів	Немає інструментів для оптимізації розкладів	Має функціонал для оптимізації календарних планувань різних типів
Графічний інтерфейс	Має зручний графічний інтерфейс	Має зручний графічний інтерфейс	Має зручний графічний інтерфейс
Доступність	Представлена безкоштовна демо-версія з пробним періодом використання	Представлена безкоштовна демо-версія з пробним періодом використання	Використання програмного забезпечення є безкоштовним

1.4 Аналіз вимог до програмного забезпечення

1.4.1 Розроблення функціональних вимог

Створений застосунок повинен надавати користувачу можливість створення календарних планувань та їх оптимізації. Для цього користувач повинен за допомогою графічного інтерфейсу створити всі необхідні частини розкладу, такі як задачі, операції, пристрої та рівні. Після цього користувач повинен мати можливість застосувати один з наявних типів оптимізації та зберегти створене календарне планування. Діаграма варіантів використання наведена у додатках.

В таблицях 1.6-1.23 наведені варіанти використання програмного забезпечення.

Таблиця 1.6 – Варіант використання УС-1

Назва	Увійти в систему
Опис	Авторизація користувача в системі
Учасники	Користувач
Передумови	Користувач повинен бути зареєстрованим у системі за умови входу за допомогою електронної адреси та паролю
Постумови	Створення нової авторизаційної сесії користувача, перехід на сторінку профілю
Основний сценарій	Користувач переходить на сторінку входу в систему та обирає метод авторизації – за допомогою електронної пошти та паролю або за допомогою Google. Після успішної авторизації користувач перенаправляється на сторінку профілю
Розширення сценаріїв	-

Таблиця 1.7 – Варіант використання УС-2

Назва	Увійти в систему через електронну адресу та пароль
Опис	Авторизація користувача в системі за допомогою електронної пошти та паролю
Учасники	Користувач
Передумови	Користувач повинен бути зареєстрованим у системі за допомогою електронної адреси та паролю
Постумови	Створення нової авторизаційної сесії користувача, перехід на сторінку профілю

Продовження таблиці 1.7

Основний сценарій	Користувач на сторінці входу вводить у відповідні поля електронну адресу та пароль. Після успішної авторизації користувач перенаправляється на сторінку профілю
Розширення сценаріїв	Якщо введені користувачем дані є невірними, користувач отримає повідомлення про помилку авторизації

Таблиця 1.8 – Варіант використання УС-3

Назва	Зареєструватись за допомогою електронної адреси та паролю
Опис	Реєстрація користувача в системі за допомогою електронної пошти та паролю
Учасники	Користувач
Передумови	-
Постумови	Створення нового користувача у системі та нової авторизаційної сесії користувача, перехід на сторінку профілю
Основний сценарій	Користувач переходить на сторінку реєстрації та вводить всі необхідні для неї дані: ім'я, прізвище, електронна адреса, пароль, пароль повторно. Після успішної реєстрації користувач перенаправляється на сторінку профілю
Розширення сценаріїв	Якщо введені користувачем дані не відповідають всім вимогам, користувач отримає інформативне повідомлення з описом про помилку

Таблиця 1.9 – Варіант використання UC-4

Назва	Увійти в систему за допомогою Google
Опис	Авторизація користувача в системі за допомогою Google
Учасники	Користувач
Передумови	-
Постумови	Створення нової авторизаційної сесії користувача, якщо користувач не був зареєстрований у системі – створення нового користувача у системі, перехід на сторінку профілю
Основний сценарій	Користувач на сторінці входу натискає на кнопку Вхід за допомогою Google. У відкритому вікні користувач повинен дотримуватись вказівок по вибору акаунту для входу в систему. Після успішної авторизації у Google, користувач перенаправляється на сторінку профілю
Розширення сценаріїв	Якщо користувач не був зареєстрованим у системі за допомогою Google – у системі автоматично проводиться реєстрація нового користувача

Таблиця 1.10 – Варіант використання UC-5

Назва	Вийти з системи
Опис	Завершення поточної авторизаційної сесії користувача
Учасники	Користувач
Передумови	Користувач повинен бути авторизованим у системі
Постумови	Авторизаційна сесія користувача завершується, користувач перенаправляється на сторінку входу в систему

Продовження таблиці 1.10

Основний сценарій	Користувач у верхньому меню натискає кнопку Вийти з системи. Після цього активна сесія завершується та користувача перенаправляє на сторінку входу в систему
Розширення сценаріїв	-

Таблиця 1.11 – Варіант використання УС-6

Назва	Відкрити сторінку профілю користувача
Опис	Перегляд особистих даних профілю користувача
Учасники	Користувач
Передумови	Користувач повинен бути авторизованим у системі
Постумови	-
Основний сценарій	Користувач переходить на сторінку профілю, де він може переглянути свої особисті дані
Розширення сценаріїв	-

Таблиця 1.12 – Варіант використання УС-7

Назва	Змінити особисті дані користувача
Опис	Оновлення особистих даних користувача
Учасники	Користувач
Передумови	Користувач повинен відкрити сторінку профілю
Постумови	Особисті дані користувача повинні бути оновлені

Продовження таблиці 1.12

Основний сценарій	Користувач на сторінці профілю натискає кнопку Змінити особисті дані. Після цього на сторінці з'являється модальне вікно, де користувач може змінити такі дані, як ім'я та прізвище користувача. При натисканні кнопки Зберегти користувач перенаправляється на сторінку профілю з оновленими даними
Розширення сценаріїв	Якщо користувач залишає будь яке поле у модальному вікні порожнім – він буде проінформований про помилку

Таблиця 1.13 - Варіант використання UC-8

Назва	Змінити пароль користувача
Опис	Оновлення паролю користувача
Учасники	Користувач
Передумови	Користувач повинен відкрити сторінку профілю
Постумови	Пароль користувача повинен бути оновленим, всі інші сесії користувача завершуються
Основний сценарій	Користувач на сторінці профілю натискає кнопку Змінити пароль. Після цього на сторінці з'являється модальне вікно, де користувач повинен ввести старий пароль та двічі новий пароль. При натисканні кнопки Зберегти користувач перенаправляється на сторінку профілю
Розширення сценаріїв	Якщо виникає помилка при зміні паролю – користувач отримує інформативне повідомлення про помилку

Таблиця 1.14 – Варіант використання УС-9

Назва	Переглянути список створених розкладів
Опис	Огляд всіх створених користувачем календарних планувань
Учасники	Користувач
Передумови	Користувач повинен бути авторизованим у системі
Постумови	-
Основний сценарій	Користувач переходить на сторінку розкладів. На ній користувач може переглянути список всіх створених ним календарних планувань та їх загальні характеристики, такі як тип розкладу, кількість рівнів, пристроїв та операцій
Розширення сценаріїв	-

Таблиця 1.15 – Варіант використання УС-10

Назва	Відкрити сторінку перегляду розкладу
Опис	Огляд календарного планування, створеного користувачем
Учасники	Користувач
Передумови	Користувач повинен бути авторизованим у системі
Постумови	-
Основний сценарій	Користувач переходить на сторінку перегляду розкладу. На ній користувач може оглянути всі частини розкладу, їх характеристики та розподіл
Розширення сценаріїв	-

Таблиця 1.16 – Варіант використання УС-11

Назва	Ввести необхідні параметри частин розкладу та змінити їх розподіл
Опис	Змінення характеристик частин розкладу та зміна їх розподілу
Учасники	Користувач
Передумови	Користувач повинен бути на сторінці перегляду розкладу
Постумови	Розклад на сторінці матиме відредагований вигляд
Основний сценарій	Користувач натискає на кнопку Редагувати розклад. Після цього біля кожної частини розкладу з'являються кнопки для їх редагування та видалення, а також створення нових. При натисканні на кнопку редагування з'являється модальне вікно, де користувач може ввести оновлені дані поточної частини розкладу. При натисканні кнопки видалення поточна частина розкладу зникає. При натисканні кнопки додавання у список частин даного типу у розкладі з'являється нова частина у вигляді за замовчуванням. Операції можна переміщувати між пристроями одного рівня перетаскуванням за допомогою комп'ютерної миші. Також операції можна переміщувати між завданнями аналогічним чином

Продовження таблиці 1.16

Розширення сценаріїв	Якщо операція була додана до задачі, вона також буде відображатись у своєму рівні у блоці «Не використані операції». Якщо операція була розміщена на виконання у деякому пристрої і при переміщенні операції між задачами або у межах однієї задачі операція змінює свій порядковий номер у задачі, дана операція зникає з пристрою та з'являється у блоці «Не використані операції» нового рівня. При видаленні задачі операції, що були створені у даній задачі, також видаляються. При видаленні пристрою, всі операції, які були призначені на виконання на нього, переміщуються у блок «Не використані операції». При видаленні рівня у розкладі всі створені у ньому приладу також видаляються
----------------------	--

Таблиця 1.17 – Варіант використання UC-12

Назва	Зберегти розклад
Опис	Збереження створеного користувачем розкладом у системі
Учасники	Користувач
Передумови	Користувач повинен бути на сторінці перегляду розкладу
Постумови	Розклад зберігається у базі даних
Основний сценарій	Користувач натискає на кнопку «Зберегти та закрити». Після цього користувач перенаправляється на сторінку перегляду списку створених розкладів
Розширення сценаріїв	Якщо розклад сформований не повністю – користувач отримає інформативне повідомлення про помилку

Таблиця 1.18 – Варіант використання УС-13

Назва	Оптимізувати розклад
Опис	Оптимізація створеного розкладу за допомогою спеціальних алгоритмів
Учасники	Користувач
Передумови	Користувач повинен бути на сторінці перегляду розкладу
Постумови	Розклад розміщується у часі оптимальним чином та відображається на сторінці у новому вигляді
Основний сценарій	Користувач натискає на кнопку Оптимізувати. Якщо існує декілька алгоритмів оптимізації – користувач обирає у відкритому модальному вікні метод оптимізації. Після проведення оптимізації розкладу користувач на сторінці може побачити новий розклад, у якому всі операції мають такі атрибути, як час початку виконання, кінець виконання, час затримки та час випередження
Розширення сценаріїв	Якщо розклад сформований не повністю – користувач отримає інформативне повідомлення про помилку

Таблиця 1.19 – Варіант використання УС-14

Назва	Оптимізувати багаторівневий розклад за допомогою швидкого алгоритму
Опис	Проведення оптимізації створеного багаторівневого розкладу за допомогою швидкого алгоритму
Учасники	Користувач

Продовження таблиці 1.19

Передумови	Користувач повинен бути на сторінці перегляду розкладу, поточний розклад повинен бути багаторівневим
Постумови	Оптимізований розклад з'являється на сторінці його перегляду
Основний сценарій	Користувач натискає на кнопку Оптимізувати. У модальному вікні користувач обирає метод оптимізації Швидкий. Після цього оптимізований розклад з'являється на поточній сторінці
Розширення сценаріїв	Якщо розклад сформований не повністю – користувач отримає інформативне повідомлення про помилку

Таблиця 1.20 – Варіант використання UC-15

Назва	Оптимізувати багаторівневий розклад за допомогою комплексного алгоритму
Опис	Проведення оптимізації створеного багаторівневого розкладу за допомогою комплексного алгоритму
Учасники	Користувач
Передумови	Користувач повинен бути на сторінці перегляду розкладу, поточний розклад повинен бути багаторівневим
Постумови	Оптимізований розклад з'являється на сторінці його перегляду
Основний сценарій	Користувач натискає на кнопку Оптимізувати. У модальному вікні користувач обирає метод оптимізації Комплексний. Після цього оптимізований розклад з'являється на поточній сторінці.
Розширення сценаріїв	Якщо розклад сформований не повністю – користувач отримає інформативне повідомлення про помилку

Таблиця 1.21 – Варіант використання UC-16

Назва	Оптимізувати однорівневий розклад з автоматичним розподілом операцій між пристроями
Опис	Проведення оптимізації створеного однорівневого розкладу за допомогою алгоритму з автоматичним розподілом операцій між всіма пристроями
Учасники	Користувач
Передумови	Користувач повинен бути на сторінці перегляду розкладу, поточний розклад повинен бути однорівневим
Постумови	Оптимізований розклад з'являється на сторінці його перегляду
Основний сценарій	Користувач натискає на кнопку Оптимізувати. Після цього оптимізований розклад з'являється на поточній сторінці
Розширення сценаріїв	Якщо розклад сформований не повністю – користувач отримає інформативне повідомлення про помилку

Таблиця 1.22 – Варіант використання UC-17

Назва	Обрати тип розкладу
Опис	Вибір типу розкладу при його створенні користувачем
Учасники	Користувач
Передумови	Користувач повинен бути на сторінці створення нового розкладу
Постумови	Створений розклад обраного типу з'являється на сторінці його перегляду

Продовження таблиці 1.22

Основний сценарій	Користувач у модальному вікні обирає тип нового розкладу. Після натискання кнопки «Створити та продовжити» користувач перенаправляється на сторінку перегляду розкладу
Розширення сценаріїв	-

Таблиця 1.23 – Варіант використання UC-18

Назва	Видалити розклад
Опис	Видалення розкладу, створеного поточним користувачем
Учасники	Користувач
Передумови	Користувач повинен бути на сторінці перегляду розкладу
Постумови	Розклад повинен бути видалений, користувач має бути перенаправлений на сторінку перегляду створених розкладів
Основний сценарій	Користувач натискає на кнопку видалення. Після цього користувач перенаправляється на сторінку перегляду всіх створених ним календарних планувальників
Розширення сценаріїв	-

Функціональні вимоги до програмного забезпечення для проведення календарних планувальників ІТ-проектів наведені у таблицях 1.24-1.51. Матриця трасування вимог зображена на рисунку 1.4.

Таблиця 1.24 – Функціональна вимога FR-1

Назва	Авторизація користувача за допомогою електронної адреси та пароллю
Опис	Система повинна надавати можливість авторизації користувача за допомогою наступних даних: електронна пошта, пароль

Таблиця 1.25 – Функціональна вимога FR-2

Назва	Авторизація користувача за допомогою Google
Опис	Система повинна надавати можливість авторизації користувача за допомогою сервісу Google.

Таблиця 1.26 – Функціональна вимога FR-3

Назва	Реєстрація користувача
Опис	Система повинна надавати можливість реєстрації користувача за умови введення наступних даних: ім'я, прізвище, електронна пошта, пароль, підтвердження паролю.

Таблиця 1.27 – Функціональна вимога FR-4

Назва	Вихід користувача з системи
Опис	Система повинна надавати можливість користувачу завершити активну авторизаційну сесію та вийти з системи.

Таблиця 1.28 – Функціональна вимога FR-5

Назва	Зміна особистих даних користувача
Опис	Система повинна надавати можливість користувачу оновлювати такі його дані, як ім'я та прізвище.

Таблиця 1.29 – Функціональна вимога FR-6

Назва	Зміна паролю користувача
Опис	Система повинна надавати можливість користувачу оновлювати пароль користувача.

Таблиця 1.30 – Функціональна вимога FR-7

Назва	Перегляд створених розкладів
Опис	Система повинна надавати можливість користувачу переглянути всі створені ним календарні планування.

Таблиця 1.31 – Функціональна вимога FR-8

Назва	Створення нового розкладу
Опис	Система повинна надавати можливість користувачу створювати нові розклади двох типів.

Таблиця 1.32 – Функціональна вимога FR-9

Назва	Редагування розкладу
Опис	Система повинна надавати можливість користувачу змінювати ім'я розкладу.

Таблиця 1.33 – Функціональна вимога FR-10

Назва	Збереження розкладу
Опис	Система повинна надавати можливість користувачу зберігати створені розклади у системі.

Таблиця 1.34 – Функціональна вимога FR-11

Назва	Видалення розкладу
Опис	Система повинна надавати можливість користувачу видаляти збережені у системі свої розклади.

Таблиця 1.35 – Функціональна вимога FR-12

Назва	Додання задачі
Опис	Система повинна надавати можливість користувачу додавати нові задачі до розкладу.

Таблиця 1.36 – Функціональна вимога FR-13

Назва	Редагування задачі
Опис	Система повинна надавати можливість користувачу редагувати такі характеристики задач, як ім'я та директивний термін.

Таблиця 1.37 – Функціональна вимога FR-14

Назва	Видалення задачі
Опис	Система повинна надавати можливість користувачу видаляти задачі з розкладу.

Таблиця 1.38 – Функціональна вимога FR-15

Назва	Додання операції
Опис	Система повинна надавати можливість користувачу додавати операції до створених задач.

Таблиця 1.39 – Функціональна вимога FR-16

Назва	Редагування операції
Опис	Система повинна надавати можливість користувачу редагувати такі характеристики операцій, як ім'я та тривалість.

Таблиця 1.40 – Функціональна вимога FR-17

Назва	Видалення операції
Опис	Система повинна надавати можливість користувачу видаляти операції з розкладу.

Таблиця 1.41 – Функціональна вимога FR-18

Назва	Додання рівня
Опис	Система повинна надавати можливість користувачу додавати рівні до розкладу.

Таблиця 1.42 – Функціональна вимога FR-19

Назва	Редагування рівня
Опис	Система повинна надавати можливість користувачу редагувати імена рівнів розкладу.

Таблиця 1.43 – Функціональна вимога FR-20

Назва	Видалення рівня
Опис	Система повинна надавати можливість користувачу видаляти рівні з розкладу.

Таблиця 1.44 – Функціональна вимога FR-21

Назва	Додання приладу
Опис	Система повинна надавати можливість користувачу додавати прилади до рівнів розкладу.

Таблиця 1.45 – Функціональна вимога FR-22

Назва	Редагування приладу
Опис	Система повинна надавати можливість користувачу редагувати такі характеристики приладу, як ім'я та час початку роботи.

Таблиця 1.46 – Функціональна вимога FR-23

Назва	Видалення приладу
Опис	Система повинна надавати можливість користувачу видалення приладу з розкладу.

Таблиця 1.47 – Функціональна вимога FR-24

Назва	Переміщення операції між задачами
Опис	Система повинна надавати можливість користувачу переміщення створених операції між задачами.

Таблиця 1.48 – Функціональна вимога FR-25

Назва	Розподіл операцій між приладами
Опис	Система повинна надавати можливість користувачу розподілу створених операції між приладами у розкладі.

Таблиця 1.49 – Функціональна вимога FR-26

Назва	Оптимізація багаторівневого розкладу швидким алгоритмом
Опис	Система повинна надавати можливість користувачу проведення оптимізації створеного розкладу швидким алгоритмом.

Таблиця 1.50 – Функціональна вимога FR-27

Назва	Оптимізація багаторівневого розкладу комплексним алгоритмом
Опис	Система повинна надавати можливість користувачу проведення оптимізації створеного розкладу комплексним алгоритмом.

Таблиця 1.51 – Функціональна вимога FR-28

Назва	Оптимізація однорівневого розкладу алгоритмом з автоматичним розподіленням операцій між приладами
Опис	Система повинна надавати можливість користувачу проведення оптимізації однорівневого розкладу за допомогою алгоритму з розподіленням операцій між приладами.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16	FR-17	FR-18	FR-19	FR-20	FR-21	FR-22	FR-23	FR-24	FR-25	FR-26	FR-27	FR-28
UC-1	+																											
UC-2	+																											
UC-3			+																									
UC-4		+																										
UC-5				+																								
UC-6																												
UC-7					+																							
UC-8						+																						
UC-9							+																					
UC-10																												
UC-11								+	+			+	+	+	+	+	+	+	+	+	+	+	+	+	+	+		
UC-12									+																			
UC-13																												
UC-14																												
UC-15																										+		
UC-16																											+	
UC-17								+																				+
UC-18											+																	

Рисунок 1.4 – Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

Розроблене програмне забезпечення повинне відповідати наступним нефункціональним вимогам:

- веб-застосунок повинен бути простим у використанні та мати зрозумілий графічний інтерфейс;
- застосунок повинен мати можливість працювати у більшості сучасних веб-браузерів.

1.5 Постановка задачі

Програмне забезпечення повинно надавати користувачу можливість створення календарних планувальників та їх оптимізації за допомогою сучасних алгоритмів. Для цього застосунок надає можливість користувачу створювати, редагувати та видаляти частини розкладу, а також розподіляти дані частини у розкладі за допомогою графічного інтерфейсу. Після заповнення всіх частин розкладу, користувач може оптимізувати створений розклад за допомогою декількох видів оптимізаційних алгоритмів.

Метою розробки є підвищення якості процесу планування виконання ІТ-проектів за рахунок створення ефективних алгоритмів календарного планування та відповідного їм програмного забезпечення.

Розробка призначена для менеджерів, обов'язками яких є планування виконання ІТ-проектів для підвищення ефективності роботи команди розробників.

Мета даної роботи досягається за рахунок розв'язання наступних задач:

- реалізація функціоналу для реєстрації та авторизації користувача за допомогою електронної пошти та паролю або за допомогою Google;
- реалізації інструментів для перегляду, створення, редагування і видалення календарних планувальників;
- створення графічного інтерфейсу для редагування частин розкладу та зміни їх взаємного розташування;
- реалізація функціоналу для оптимізації багаторівневих розкладів за допомогою швидкого та комплексного алгоритмів з використанням людино-комп'ютерної системи ітераційного розподілення задач між виконавцями;
- реалізація функціоналу для отримання оптимального плану одноетапної задачі календарного планування з паралельними ідентичними приладами і одним директивним терміном.

					КПІ.ІП-9213.045490.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		43

Висновки до розділу

В першому розділі дипломного проєкту було детально описано основи теорії розкладів, її основних понять та термінів, на основі яких побудовані ПДС-алгоритми для ефективного створення оптимального календарного планування для різного типу поставлених задач. Також було проведено детальний аналіз предметної області та надано її змістовний опис.

Для аналізу успішних ІТ-проєктів було проаналізовано набір відомих технічних рішень, що використовувались для створення аналогів програмного забезпечення. Для цього було порівняно популярні інструменти для побудови всіх частин майбутнього застосунку та обрано найкращі саме для поставленої задачі. Також було проведено аналіз існуючих аналогів та виділено основні відмінності дипломного проєкту від них.

Задля опису можливостей застосунку було детально описано всі функціональні вимоги та варіанти використання, а також було створено діаграму Use Case. Для створених описів функціональних вимог та варіантів використання було створено матрицю трасування вимог. Також було описано всі нефункціональні вимоги до програмного забезпечення.

					КПІ.ІП-9213.045490.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		44

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Бізнес-процеси реєстрації користувача, створення розкладу багаторівневої системи та створення розкладу однорівневої системи з автоматичним розподіленням задач описуються за допомогою BPMN діаграм, що зображено на рисунках 2.1-2.3.

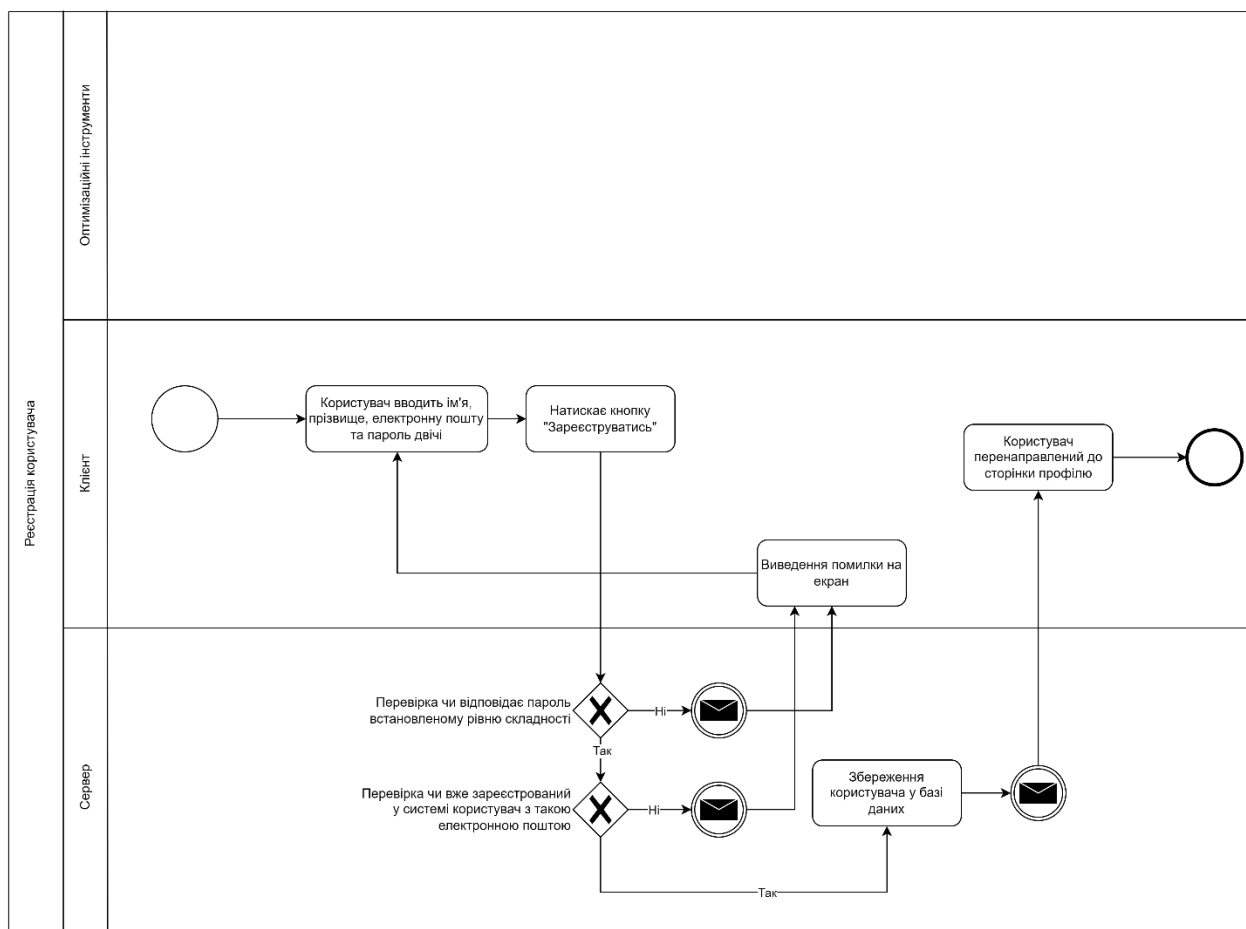


Рисунок 2.1 – BPMN діаграма реєстрації користувача

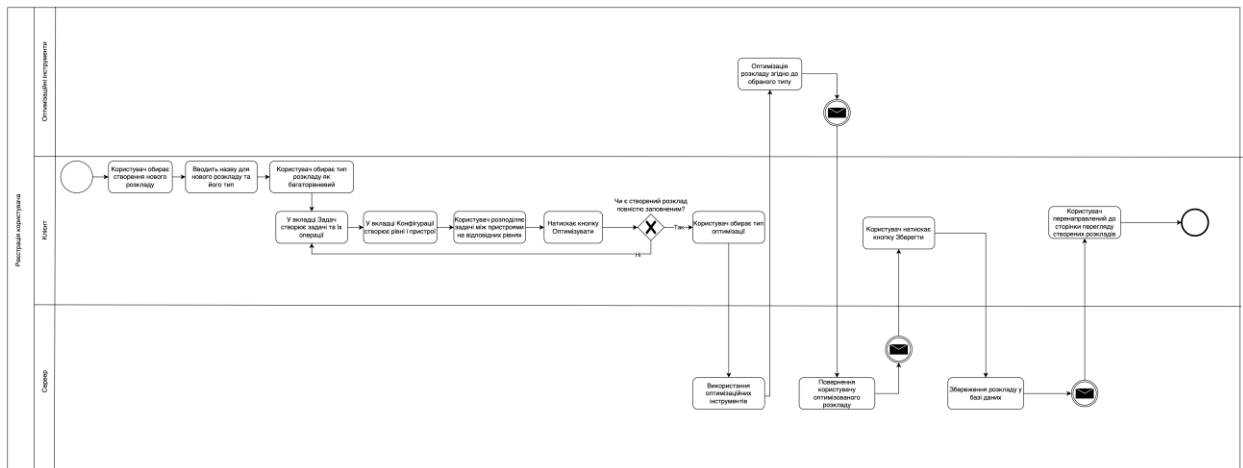


Рисунок 2.2 – BPMN діаграма створення багаторівневого розкладу

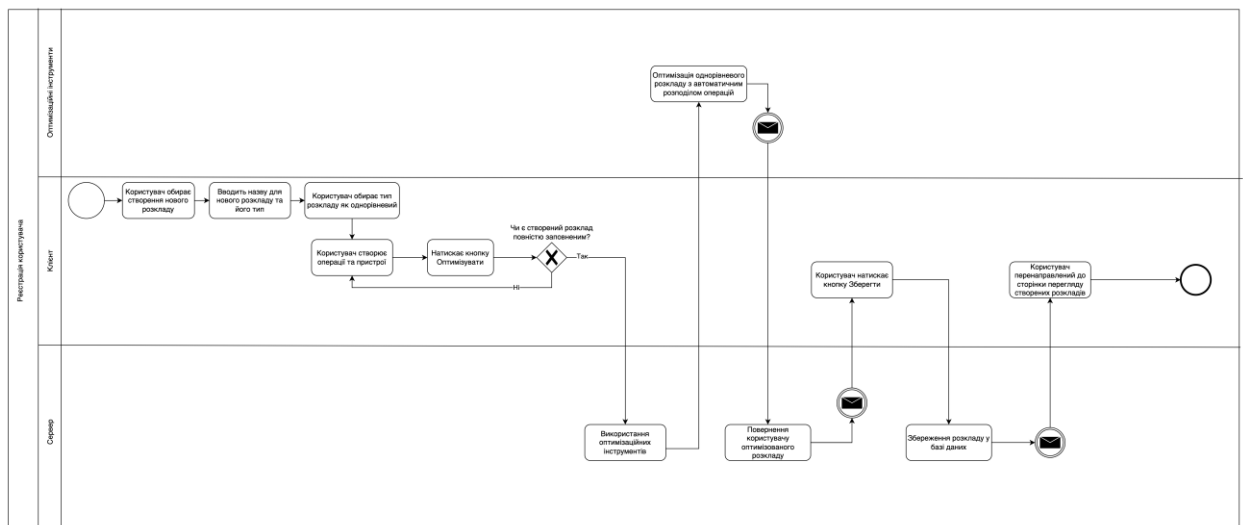


Рисунок 2.3 – BPMN діаграма створення однорівневого розкладу з автоматичним розподіленням операцій між пристроями

2.2 Архітектура програмного забезпечення

Програмне забезпечення для календарного планування містить у собі веб-застосунок та набір оптимізаційних інструментів, що зображено на рисунку 2.4.

Веб-застосунок складається з клієнтської та серверної частин. За допомогою графічного інтерфейсу користувач має змогу реєструватись у додатку, створювати розклади та оптимізувати їх.

Оптимізаційні інструменти використовуються серверною частиною веб-застосунку для оптимізації створених багаторівневих розкладів та

однорівневих розкладів з автоматичним розподіленням операцій між приладами.

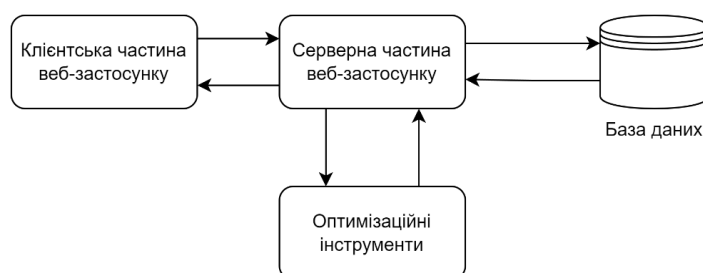


Рисунок 2.4 – Схема взаємодії частин ПЗ

Архітектура клієнтської частини веб-застосунку

У клієнтській частині застосунку для завантаження залежностей використовується менеджер пакетів yarn. Використані бібліотеки вказані у таблиці 2.1.

Таблиця 2.1 – Залежності клієнтської частини проєкту

Залежність	Опис
react	бібліотека для створення веб-застосунків
mobx	бібліотека для роботи з глобальним стану веб-застосунку та слідкуванням на його змінами
typescript	залежність, що дозволяє транспілювати код, написаний на мові програмування TypeScript, у код на мові програмування JavaScript, зрозумілий вбудованому двигуну у Node.js
uuid	бібліотека для генерування унікальних ідентифікаторів

У бібліотеці React.js основними частинами застосунку є компоненти, сервіси та глобальний стан. Схему з їх зображеннями можна побачити у графічній документації креслення «Схема структурна компонентів програмного забезпечення». Роботу основних компонентів і сервісів описано у таблицях 2.2 та 2.3 відповідно.

Таблиця 2.2 – Компоненти клієнтської частини веб-застосунку

Компонент	Опис
App	основний компонент, що відповідає за відображення сторінок веб-застосунку в залежності від адреси
LoginPage	відображає сторінку входу в систему застосунку
RegistrationPage	відображає сторінку реєстрації користувача у застосунку
ProfilePage	відображає сторінку профілю користувача та форми для його редагування
SchedulesPage	містить відображення всіх створених розкладів поточним користувачем
SingleDeviceSchedule Form	містить у собі форму для роботи з багаторівневим розкладом
MultiDeviceScheduleForm	містить у собі форму для роботи з однорівневим розкладом з автоматичним розподілом операцій між пристроями
TasksList	відображає список завдань у розкладі
TaskDnd	відображає картку завдання зі створеними операціями в ній
LayersList	відображає список рівнів у багаторівневому розкладі
LayerDnd	відображає картку рівня зі створеними пристроями в ньому

Продовження таблиці 2.2

Компонент	Опис
DeviceDnd	відображає картку пристрою з призначеними на нього операціями
OperationCard	відображає картку властивостей операції у вигляді картки
NavigationBar	відображає верхнє глобальне меню для переходу у різні частини веб-застосунку

Таблиця 2.3 – Сервіси клієнтської частини веб-застосунку

Сервіс	Опис
AuthService	надає можливість здійснювати реєстрацію, вхід в систему за допомогою електронної пошти і пароллю та за допомогою Google
UserService	містить у собі набір функцій для отримання даних про поточного користувача, редагування його особистих даних та зміни пароллю
ScheduleService	містить у собі набір функцій для створення, редагування та видалення календарних планувальників, отримання списку створених розкладів поточним користувачем та оптимізацію різних видів розкладів

Архітектура серверної частини веб-застосунку

Серверна частина застосунку побудована за допомогою фреймворку Nest.js. У даному фреймворку основними частинами є модулі, контролери і сервіси. Залежності серверної частини застосунку наведено у таблиці 2.4. Схему з зображенням основних частин серверного застосунку зображено у графічній документації креслення «Схема структурна компонентів програмного забезпечення». Робота основних сервісів і контролерів описана у таблицях 2.5 та 2.6 відповідно.

Таблиця 2.4 – Залежності серверної частини веб-застосунку

Залежність	Опис
nestjs	фреймворк для створення серверних застосунків
nestjs/jwt	адаптована бібліотека для Nest.js, за допомогою якої можна працювати з JSON Web Tokens для авторизації
pg	бібліотека для роботи з базою даних PostgreSQL
passport	бібліотека для роботи з авторизацією користувачів та перевіркою активності сесій
class-validator	бібліотека для перевірки даних, що надходять на сервер, перед тим, як вони будуть обробляться застосунком
class-transformer	бібліотека для трансформації даних, що надходять на сервер, у потрібний для серверу формат
typescript	залежність, що дозволяє транспілювати код, написаний на мові програмування TypeScript, у код на мові програмування JavaScript, зрозумілий вбудованому двигуну у Node.js

Таблиця 2.5 – Сервіси серверної частини застосунку

Сервіс	Опис
AuthService	дозволяє проводити реєстрацію користувача та вхід у систему застосунку за допомогою електронної адреси і паролю та за допомогою Google
AuthProvidersService	дозволяє працювати з даними про провайдерів авторизації користувача
PgService	дозволяє генерувати та здійснювати запити у базу даних PostgreSQL
PythonService	відповідає за зв'язок серверної частини застосунку з оптимізаційними інструментами, написаними на мові програмування Python
RefreshSessionsService	дозволяє працювати з сесіями користувачів, перевіряти активність сесії та оновлювати її час роботи
SchedulesService	дозволяє створювати, редагувати, видаляти та оптимізовувати календарні планування, а також отримувати список розкладів за користувачем
ScryptService	дозволяє зашифровувати та розшифровувати дані за допомогою вбудованої бібліотеки scrypt
UsersService	дозволяє створювати, редагувати та видаляти користувачів у системі, а також отримувати профіль поточного користувача

Таблиця 2.6 – Контролери серверної частини веб-застосунку

AuthController		
POST	/auth/login	авторизує користувача
POST	/auth/register	реєструє користувача у системі
POST	/auth/refresh-token	оновлює токени авторизації для продовження активної сесії
POST	/auth/google	авторизує користувача через Google
POST	/auth/logout	завершує поточну авторизаційну сесію користувача
SchedulesController		
POST	/schedule	створює новий розклад для поточного користувача
PATCH	/schedule	оновлює розклад поточного користувача
DELETE	/schedule/:id	видаляє розклад по його ідентифікатору
GET	/schedules/me	повертає список всіх створених розкладів поточним користувачем
GET	/schedules/:id	повертає розклад, знайдений за даним ідентифікатором
POST	/schedules/optimize/single	оптимізує багаторівневий розклад

Продовження таблиці 2.6

SchedulesController		
POST	/schedules/optimize/multi	оптимізує однорівневий розклад з автоматичним розподілом операцій між пристроями
UsersController		
GET	/user/me	повертає профіль поточного користувача
PATCH	/user/me	оновлює профіль поточного користувача
PATCH	/user/password	оновлює пароль поточного користувача

Архітектура оптимізаційних інструментів проєкту

Оптимізаційні інструменти для застосунку побудовані за допомогою мови програмування Python. Діаграма класів даної частини застосунку наведена на рисунку 2.5.

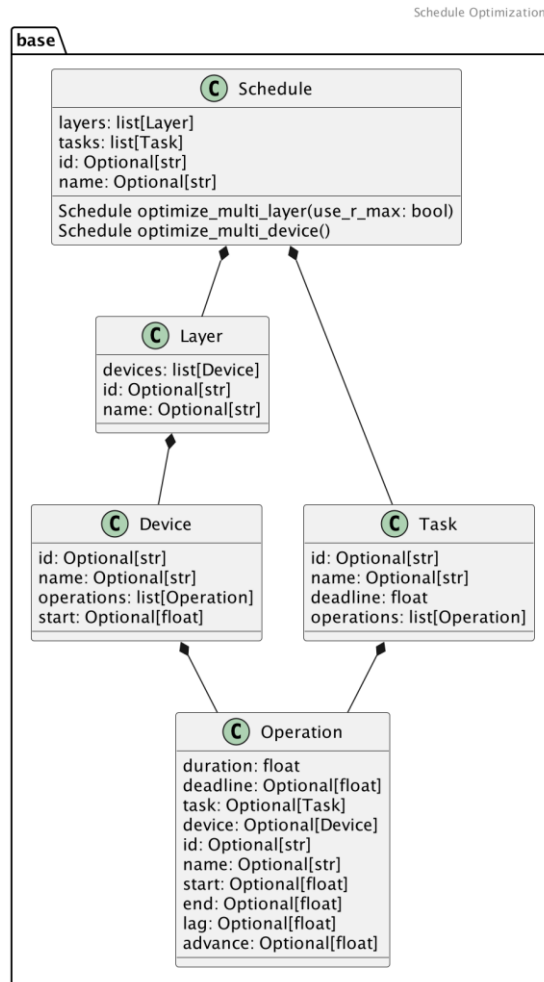


Рисунок 2.5 – Структура оптимізаційної частини застосунку

Дана структура класів дозволяє ефективно зберігати дані про календарне планування та оптимізувати їх. Кожен об’єкт з даних класів має такі поля, як “id” та “name”, що дозволяє маркувати об’єкти для зручності та ідентифікувати їх після оптимізації розкладу.

Метод “optimize_multi_layer” класу Schedule відповідає за оптимізацію багаторівневого розкладу за допомогою алгоритмів A1 та A2. Даний метод повертає копію початкового розкладу після впорядкування всіх операцій у їх пристроях на всіх рівнях та їх розподілення у часі.

Метод “optimize_multi_device” класу Schedule оптимізує однорівневий розклад, розподіляючи операції між приладами та задаючи їх початковий час виконання за допомогою алгоритму A0. Даний метод також повертає копію початкового розкладу після оптимізації.

2.3 Конструювання програмного забезпечення

У розробленому застосунку в якості системи управління базами даних використовується PostgreSQL. База даних, створена для даного застосунку, призначена для зберігання даних про користувачів, їх авторизаційні сесії, а також даних про створені календарні планування користувачем. Опис таблиць бази даних описано у таблицях 2.7-2.14. Модель бази даних наведена у графічній документації креслення «Схема бази даних».

Таблиця 2.7 – Опис таблиці Users

Поле	Тип даних	Опис
userId	UUID	унікальний ідентифікатор користувача
firstName	VARCHAR	ім'я користувача
lastName	VARCHAR	прізвище користувача
email	VARCHAR	електронна пошта користувача

Таблиця 2.8 – Опис таблиці AuthProviders

Поле	Тип даних	Опис
userId	UUID	ідентифікатор користувача
authName	VARCHAR	тип авторизації
password	VARCHAR	хеш паролю користувача

Таблиця 2.9 – Опис таблиці RefreshSessions

Поле	Тип даних	Опис
refreshToken	UUID	токен оновлення
userId	UUID	ідентифікатор користувача
userAgent	VARCHAR	дані про конфігурації системи користувача
fingerprint	VARCHAR	унікальний хеш пристрою користувача
expiresIn	BIGINT	час роботи токена
createAt	TIMESTAMP	дата створення токена

Таблиця 2.10 – Опис таблиці Schedules

Поле	Тип даних	Опис
id	UUID	ідентифікатор розкладу
name	VARCHAR	назва розкладу
ownerId	UUID	ідентифікатор користувача
type	VARCHAR	тип розкладу
timestamp	TIMESTAMP	час створення розкладу

Таблиця 2.11 – Опис таблиці ScheduleLayers

Поле	Тип даних	Опис
id	UUID	ідентифікатор рівня розкладу
name	VARCHAR	назва рівня розкладу
index	INT	порядковий номер рівня
scheduleId	UUID	ідентифікатор розкладу

Таблиця 2.12 – Опис таблиці ScheduleDevices

Поле	Тип даних	Опис
id	UUID	ідентифікатор приладу
name	VARCHAR	назва приладу
index	INT	порядковий номер приладу
layerId	UUID	ідентифікатор рівня розкладу
start	NUMERIC	момент початку роботи приладу

Таблиця 2.13 – Опис таблиці ScheduleTasks

Поле	Тип даних	Опис
id	UUID	ідентифікатор задачі
name	VARCHAR	назва задачі
scheduleId	UUID	ідентифікатор розкладу
deadline	NUMERIC	директивний термін задачі

Таблиця 2.14 – Опис таблиці ScheduleOperations

Поле	Тип даних	Опис
id	UUID	ідентифікатор операції
name	VARCHAR	назва приладу
taskId	UUID	ідентифікатор задачі
deviceId	UUID	ідентифікатор приладу
index	INT	порядковий номер у задачі
deviceIndex	INT	порядковий номер у приладі

Продовження таблиці 2.14

deadline	NUMERIC	директивний термін операції
duration	NUMERIC	тривалість операції
start	NUMERIC	момент початку виконання операції

Опис утиліт, бібліотек та іншого програмного забезпечення, що використовується у розробці, наведено у таблиці 2.15.

Таблиця 2.15 – Опис утиліт

Назва утиліти	Опис застосування
Visual Studio Code	Середовище розробки для всіх частин застосунку, а також для тестування та розгортання програмного забезпечення
Google Chrome Developer Tools	Інструменти для виявлення помилок у клієнтській частині застосунку
Postman	Програмне забезпечення для проведення запитів за допомогою протоколу HTTP для тестування серверної частини застосунку
pgAdmin	Програмне забезпечення для доступу до бази даних PostgreSQL за допомогою графічного інтерфейсу

2.4 Аналіз безпеки даних

У серверній частині застосунку реалізовано безпечну авторизацію користувача за допомогою електронної пошти та паролю або за допомогою сервісу Google. Дані методи авторизації було реалізовано з урахуванням можливих вразливостей до безпеки даних.

При реєстрації за допомогою електронної адреси та пароллю у серверній частині застосунку пароль користувача хешується та зберігається у базі даних у захешованому вигляді. Завдяки цьому пароль користувача неможливо отримати після його хешування.

При авторизації за допомогою електронної пошти та пароллю введений пароль також хешується та зрівнюється з хешем, який збережений у базі даних у користувача з введеною електронною адресою. Якщо дані хеши співпадають, то система може вважати введений пароль вірним та створить авторизаційну сесію для користувача.

Для авторизації в системі за допомогою Google користувач, після натискання кнопки «Login with Google» та введення коректних даних акаунту Google, отримує на сторінці одноразовий ключ, який надсилається на сервер, де даний токен використовується для отримання особистих даних про користувача, таких як ім'я та електронна адреса. При цьому отримати дані про користувача можна лише один раз.

Після входу в систему кожен користувач отримує два токени – access token та refresh token. Access token має невеликий термін роботи та використовується для доступу до функціоналу застосунку від імені користувача. Refresh token має довгий термін роботи і використовується для оновлення access token. Завдяки даному набору токенів сесія користувача є найбільш захищеною, оскільки токен для доступу до даних є дійсним лише невеликий час, а токен для його оновлення – захищений та збережений у базі даних. Для підвищення безпеки користування сесія користувача на сервері також містить у собі fingerprint – унікальний ідентифікатор пристрою користувача, завдяки чому можна уникнути вразливостей у безпеці системи при перехопленні запитів користувача.

Таким чином, у процесі розробки програмного забезпечення для календарного планування виконання ІТ-проєкту було виконано захист всіх вразливих частин застосунку та його даних.

Висновки до розділу

В другому розділі дипломного проекту було розглянуто:

- схему роботи і взаємодії частин розробленого ПЗ;
- процес реєстрації користувача та створення календарних планувальників різних видів;
- набір компонентів і сервісів, за допомогою яких побудована клієнтська частина веб-застосунку;
- структуру бази даних та її таблиць;
- набір модулів, сервісів та контролерів, розроблених у серверній частині застосунку;
- структуру класів оптимізаційних інструментів, що використовуються у застосунку;
- схему бази даних та детальний опис всіх її таблиць;
- аналіз безпеки даних у отриманому програмному забезпеченні.

					КПІ.ІП-9213.045490.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		60

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналізуючи якість програмного забезпечення, було проведено перевірку його на відповідність поставленим функціональним та нефункціональним вимогам. Детальний опис перевірки нефункціональних вимог у отриманому програмному забезпеченні наведений у таблиці 3.1.

Таблиця 3.1 – Перевірка нефункціональних вимог

Зручний графічний інтерфейс	Веб-застосунок має доволі простий, зручний та лаконічний графічний інтерфейс
Сумісність з сучасними браузерами	Веб-додаток був перевірений на останніх версіях браузерів Chrome, Firefox, Edge та Safari

Всі функціональні вимоги були перевірені за допомогою мануального тестування. Матриця відповідності функціональних вимог і тестів, які перевіряють виконання поставлених вимог, наведено на рисунку 3.1.

3.2 Опис процесів тестування

Опис всіх тестів, які були проведені мануально на отриманому програмному забезпеченні, наведено у таблицях 3.2-3.27.

Таблиця 3.2 – Тест 1

Тест	Авторизація користувача за допомогою електронної пошти та паролю
Початковий стан системи	Користувач перебуває на сторінці входу в систему
Вхідні дані	Електронна пошта, пароль

Продовження таблиці 3.2

Опис проведення тесту	На сторінці авторизації заповнюються наступні поля: електронна адреса, пароль. Після цього натискається кнопка Вхід
Очікуваний результат	Авторизація відбувається успішно, користувач перенаправляється на сторінку профілю
Фактичний результат	Авторизація відбувається успішно, користувач перенаправляється на сторінку профілю

Таблиця 3.3 – Тест 2

Тест	Авторизація користувача за допомогою неправильного паролю
Початковий стан системи	Користувач перебуває на сторінці входу в систему
Вхідні дані	Електронна пошта, пароль
Опис проведення тесту	На сторінці авторизації заповнюються наступні поля: електронна адреса, пароль (заповнюється неправильним). Після цього натискається кнопка Вхід
Очікуваний результат	Відбувається невдала спроба авторизації, користувач отримує повідомлення про помилку авторизації
Фактичний результат	Відбувається невдала спроба авторизації, користувач отримує повідомлення про помилку авторизації

Таблиця 3.4 – Тест 3

Тест	Авторизація користувача за допомогою сервісу Google
Початковий стан системи	Користувач перебуває на сторінці входу в систему
Вхідні дані	Електронна пошта та пароль від Google-акаунту

Продовження таблиці 3.4

Опис проведення тесту	На сторінці входу в систему натискається кнопка «Вхід за допомогою Google», після чого користувач перенаправляється на сторінку авторизації у сервісі Google. Після введення всіх необхідних даних користувач підтверджує авторизацію у застосунку
Очікуваний результат	Користувач успішно авторизується у системі та перенаправляється на сторінку перегляду профілю
Фактичний результат	Користувач успішно авторизується у системі та перенаправляється на сторінку перегляду профілю

Таблиця 3.5 – Тест 4

Тест	Реєстрація користувача
Початковий стан системи	Користувач перебуває на сторінці реєстрації
Вхідні дані	Ім'я, прізвище, електронна пошта, пароль, пароль повторно
Опис проведення тесту	На сторінці реєстрації заповнюються наступні поля: ім'я, прізвище, електронна адреса, пароль та підтвердження паролю. Після цього натискається кнопка Реєстрація
Очікуваний результат	Реєстрація відбувається успішно, користувач автоматично входить в систему та перенаправляється на сторінку профілю
Фактичний результат	Реєстрація відбувається успішно, користувач автоматично входить в систему та перенаправляється на сторінку профілю

Таблиця 3.6 – Тест 5

Тест	Невдала реєстрація користувача
Початковий стан системи	Користувач перебуває на сторінці реєстрації
Вхідні дані	Ім'я, прізвище, електронна пошта, пароль, підтвердження паролю
Опис проведення тесту	На сторінці реєстрації заповнюються наступні поля: ім'я, прізвище, електронна адреса, пароль та підтвердження пароль (заповнюється неправильно). Після цього натискається кнопка Реєстрація
Очікуваний результат	Реєстрація відбувається невдало, користувач отримує інформативне повідомлення про помилку при реєстрації
Фактичний результат	Реєстрація відбувається невдало, користувач отримує інформативне повідомлення про помилку при реєстрації

Таблиця 3.7 – Тест 6

Тест	Перегляд списку створених календарних планувань
Початковий стан системи	Користувач повинен бути авторизованим у системі
Вхідні дані	-
Опис проведення тесту	Користувач переходить на сторінку перегляду створених розкладів
Очікуваний результат	На сторінці користувачеві демонструється список всіх створених календарних планувань та їх характеристики, такі як кількість рівнів, операцій, пристроїв та задач
Фактичний результат	На сторінці користувачеві демонструється список всіх створених календарних планувань та їх характеристики, такі як кількість рівнів, операцій, пристроїв та задач

Таблиця 3.8 – Тест 7

Тест	Створення нового розкладу
Початковий стан системи	Користувач повинен перебувати на сторінці перегляду створених розкладів
Вхідні дані	Назва розкладу, його тип
Опис проведення тесту	Користувач натискає на кнопку додання нового розкладу. У модальному вікні користувач вводить назву нового розкладу та його тип. Після цього користувач натискає кнопку «Створити та продовжити»
Очікуваний результат	Користувач перенаправляється на сторінку редагування нового порожнього проєкту
Фактичний результат	Користувач перенаправляється на сторінку редагування нового порожнього проєкту

Таблиця 3.9 – Тест 8

Тест	Збереження розкладу
Початковий стан системи	Користувач повинен перебувати на сторінці редагування календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку Зберегти у нижньому меню
Очікуваний результат	Користувач перенаправляється на сторінку перегляду створених розкладів. У списку наявний збережений розклад
Фактичний результат	Користувач перенаправляється на сторінку перегляду створених розкладів. У списку наявний збережений розклад

Таблиця 3.10 – Тест 9

Тест	Зміна імені розкладу
Початковий стан системи	Користувач повинен перебувати на сторінці редагування календарного планування
Вхідні дані	Назва розкладу
Опис проведення тесту	Користувач натискає на кнопку редагування поруч з іменем користувача. У модальному вікні користувач вводить нову назву розкладу та натискає кнопку Зберегти
Очікуваний результат	Користувач перенаправляється на сторінку редагування розкладу з новим іменем
Фактичний результат	Користувач перенаправляється на сторінку редагування розкладу з новим іменем

Таблиця 3.11 – Тест 10

Тест	Додання нового рівня у розклад
Початковий стан системи	Користувач повинен перебувати на сторінці редагування багаторівневого календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку «Додати новий рівень»
Очікуваний результат	У розкладі з'являється новий рівень з назвою за замовчуванням
Фактичний результат	У розкладі з'являється новий рівень з назвою за замовчуванням

Таблиця 3.12 – Тест 11

Тест	Редагування рівня розкладу
Початковий стан системи	Користувач повинен перебувати на сторінці редагування багаторівневого календарного планування
Вхідні дані	Назва рівня
Опис проведення тесту	Користувач натискає на кнопку редагування обраного рівня. У модальному вікні користувач заповнює поле назви з новим значенням та натискає кнопку Зберегти
Очікуваний результат	Обраний рівень у розкладі відображається з новою назвою
Фактичний результат	Обраний рівень у розкладі відображається з новою назвою

Таблиця 3.13 – Тест 12

Тест	Видалення рівня розкладу
Початковий стан системи	Користувач повинен перебувати на сторінці редагування багаторівневого календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку видалення обраного рівня
Очікуваний результат	Обраний рівень видалився у розкладі та більше не відображається у ньому, всі пристрої, які були створені у даному рівні, також видалились
Фактичний результат	Обраний рівень видалився у розкладі та більше не відображається у ньому, всі пристрої, які були створені у даному рівні, також видалились

Таблиця 3.14 – Тест 13

Тест	Створення пристрою у розкладі
Початковий стан системи	Користувач повинен перебувати на сторінці редагування календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку «Додати новий пристрій»
Очікуваний результат	У розклад додається новий пристрій з характеристиками за замовчуванням
Фактичний результат	У розклад додається новий пристрій з характеристиками за замовчуванням

Таблиця 3.15 – Тест 14

Тест	Редагування пристрою
Початковий стан системи	Користувач повинен перебувати на сторінці редагування календарного планування
Вхідні дані	Назва пристрою, момент початку роботи
Опис проведення тесту	Користувач натискає на кнопку редагування у обраному пристрої. У модальному вікні користувач заповнює такі поля, як нова назва та момент початку роботи пристрою (якщо розклад є однорівневим)
Очікуваний результат	Обраний пристрій з'являється у розкладі з оновленими характеристиками
Фактичний результат	Обраний пристрій з'являється у розкладі з оновленими характеристиками

Таблиця 3.16 – Тест 15

Тест	Видалення пристрою
Початковий стан системи	Користувач повинен перебувати на сторінці редагування календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку видалення у обраному пристрої
Очікуваний результат	Обраний пристрій видаляється з календарного планування
Фактичний результат	Обраний пристрій видаляється з календарного планування

Таблиця 3.17 – Тест 16

Тест	Створення нової задачі у розкладі
Початковий стан системи	Користувач повинен перебувати на сторінці редагування багаторівневого календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку «Додати нову задачу»
Очікуваний результат	У розклад додається нова задача з параметрами за замовчуванням
Фактичний результат	У розклад додається нова задача з параметрами за замовчуванням

Таблиця 3.18 – Тест 17

Тест	Редагування задачі
Початковий стан системи	Користувач повинен перебувати на сторінці редагування багаторівневого календарного планування
Вхідні дані	Назва задачі, директивний термін
Опис проведення тесту	Користувач натискає на кнопку редагування обраної задачі. У модальному вікні користувач вводить такі дані, як нову назву задачі та її директивний термін. Після цього користувач натискає на кнопку Зберегти
Очікуваний результат	Обрана задача оновлюється у поточному календарному плануванні
Фактичний результат	Обрана задача оновлюється у поточному календарному плануванні

Таблиця 3.19 – Тест 18

Тест	Видалення задачі з розкладу
Початковий стан системи	Користувач повинен перебувати на сторінці редагування багаторівневого календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку видалення обраної задачі
Очікуваний результат	Задача видаляється з розкладу; всі операції, які були створені у даній задачі, також видаляються
Фактичний результат	Задача видаляється з розкладу; всі операції, які були створені у даній задачі, також видаляються

Таблиця 3.20 – Тест 19

Тест	Створення нової операції у розкладу
Початковий стан системи	Користувач повинен перебувати на сторінці редагування календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку створення нової операції
Очікуваний результат	Нова операція додається до розкладу у форматі за замовчуванням
Фактичний результат	Нова операція додається до розкладу у форматі за замовчуванням

Таблиця 3.21 – Тест 20

Тест	Редагування операції
Початковий стан системи	Користувач повинен перебувати на сторінці редагування календарного планування
Вхідні дані	Назва операції, тривалість
Опис проведення тесту	Користувач натискає на кнопку редагування операції. У модальному вікні користувач вводить такі дані, як нову назву операції та її тривалість. Після цього користувач натискає кнопку Зберегти
Очікуваний результат	Операція оновлюється у розкладі та відображається з оновленими характеристиками
Фактичний результат	Операція оновлюється у розкладі та відображається з оновленими характеристиками

Таблиця 3.22 – Тест 21

Тест	Видалення операції
Початковий стан системи	Користувач повинен перебувати на сторінці редагування календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку видалення операції
Очікуваний результат	Обрана операція видаляється з розкладу та зникає зі сторінки
Фактичний результат	Обрана операція видаляється з розкладу та зникає зі сторінки

Таблиця 3.23 – Тест 22

Тест	Переміщення операції між задачами
Початковий стан системи	Користувач повинен перебувати на сторінці редагування багаторівневого календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач за допомогою миші переміщує обрану операції до іншої задачі
Очікуваний результат	Операція відображається у новій задачі; якщо порядковий номер операції у задачі був змінений і операція була призначена на пристрій на виконання, то операція перестає бути призначеною на даний пристрій
Фактичний результат	Операція відображається у новій задачі; якщо порядковий номер операції у задачі був змінений і операція була призначена на пристрій на виконання, то операція перестає бути призначеною на даний пристрій

Таблиця 3.24 – Тест 23

Тест	Переміщення операції між пристроями
Початковий стан системи	Користувач повинен перебувати на сторінці редагування багаторівневого календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач за допомогою миші переміщує операцію до іншого пристрою одного рівня
Очікуваний результат	Операція відображається у новому пристрої та зникає з попереднього пристрою
Фактичний результат	Операція відображається у новому пристрої та зникає з попереднього пристрою

Таблиця 3.25 – Тест 24

Тест	Оптимізація багаторівневого розкладу швидким алгоритмом
Початковий стан системи	Користувач повинен перебувати на сторінці редагування багаторівневого календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач за натискає на кнопку оптимізації. У модальному вікні користувач обирає оптимізацію швидким алгоритмом
Очікуваний результат	Оптимізований розклад з'являється на сторінці; такі об'єкти розкладу, як операції та пристрою, демонструють додаткові характеристики, такі як моменти початку та кінця роботи, а також затримку та випередження у даному розкладі

Продовження таблиці 3.25

Фактичний результат	Оптимізований розклад з'являється на сторінці; такі об'єкти розкладу, як операції та пристрою, демонструють додаткові характеристики, такі як моменти початку та кінця роботи, а також затримку та випередження у даному розкладі
---------------------	---

Таблиця 3.26 – Тест 25

Тест	Оптимізація багаторівневого розкладу комплексним алгоритмом
Початковий стан системи	Користувач повинен перебувати на сторінці редагування багаторівневого календарного планування
Вхідні дані	-
Опис проведення тесту	Користувач за натискає на кнопку оптимізації. У модальному вікні користувач обирає оптимізацію комплексним алгоритмом
Очікуваний результат	Оптимізований розклад комплексним алгоритмом з'являється на сторінці
Фактичний результат	Оптимізований розклад комплексним алгоритмом з'являється на сторінці

Таблиця 3.27 – Тест 26

Тест	Оптимізація однорівневого розкладу алгоритмом з автоматичним розподілом операцій між пристроями
Початковий стан системи	Користувач повинен перебувати на сторінці редагування однорівневого календарного планування
Вхідні дані	-

Продовження таблиці 3.27

Опис проведення тесту	Користувач за натискає на кнопку оптимізації у нижньому меню
Очікуваний результат	Оптимізований однорівневий розклад з'являється на сторінці; операції та пристрою демонструють додаткові параметри, такі як моменти початку та кінця роботи, а також затримку та випередження у даному розкладі
Фактичний результат	Оптимізований однорівневий розклад з'являється на сторінці; операції та пристрою демонструють додаткові параметри, такі як моменти початку та кінця роботи, а також затримку та випередження у даному розкладі

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16	FR-17	FR-18	FR-19	FR-20	FR-21	FR-22	FR-23	FR-24	FR-25	FR-26	FR-27	FR-28
Тест 1	+																											
Тест 2	+																											
Тест 3		+																										
Тест 4			+																									
Тест 5			+																									
Тест 6							+				+																	
Тест 7								+																				
Тест 8									+																			
Тест 9									+																			
Тест 10																		+										
Тест 11																			+									
Тест 12																				+								
Тест 13																					+							
Тест 14																						+						
Тест 15																							+					
Тест 16												+																
Тест 17													+															
Тест 18														+														
Тест 19															+													
Тест 20																+												
Тест 21																	+											
Тест 22																								+				
Тест 23																									+			
Тест 24																										+		
Тест 25																											+	
Тест 26																												+

Рисунок 3.1 – Матриця відповідності функціональних вимог та проведених тестів

3.3 Опис контрольного прикладу

За основу контрольного прикладу було обрано опис тесту 7 «Створення нового розкладу».

Для створення нового розкладу користувач повинен бути зареєстрованим у системі та перебувати на сторінці перегляду створених календарних планувань.



Рисунок 3.2 – Сторінка перегляду створених розкладів

При натисканні кнопки з позначкою «+» користувач потрапляє на сторінку створення нового розкладу, де потрібно ввести такі дані, як назву нового розкладу та його тип.

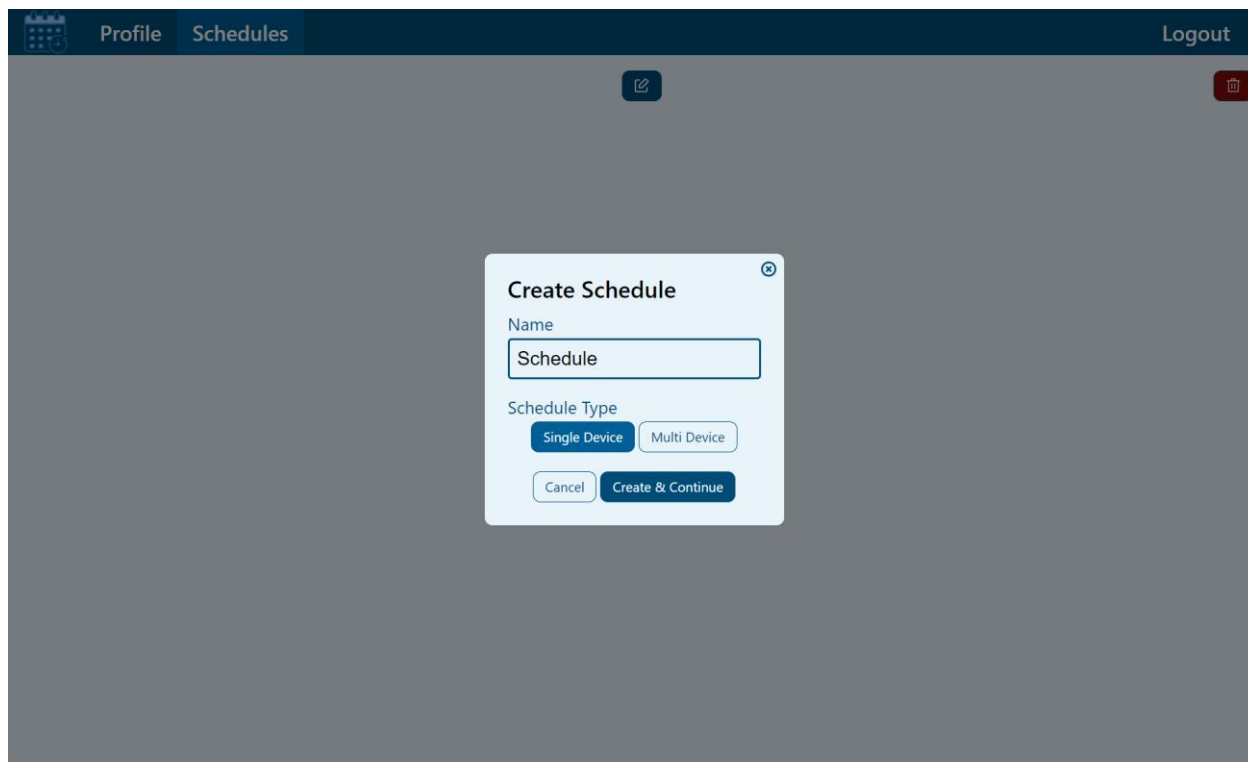


Рисунок 3.3 – Сторінка створення нового розкладу

Після введення всіх необхідних даних, користувач повинен натиснути кнопку «Create & Continue», після чого користувач переходить на сторінку редагування розкладу, який був створений у формі за замовчуванням в залежності від обраного типу.

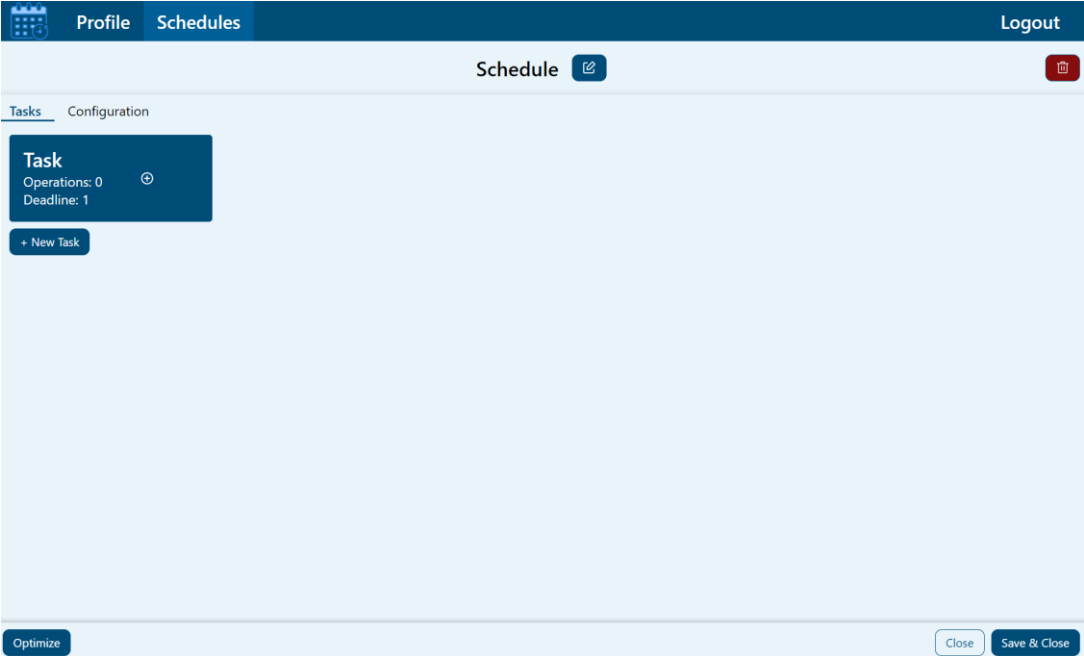


Рисунок 3.4 – Сторінка редагування багаторівневого розкладу

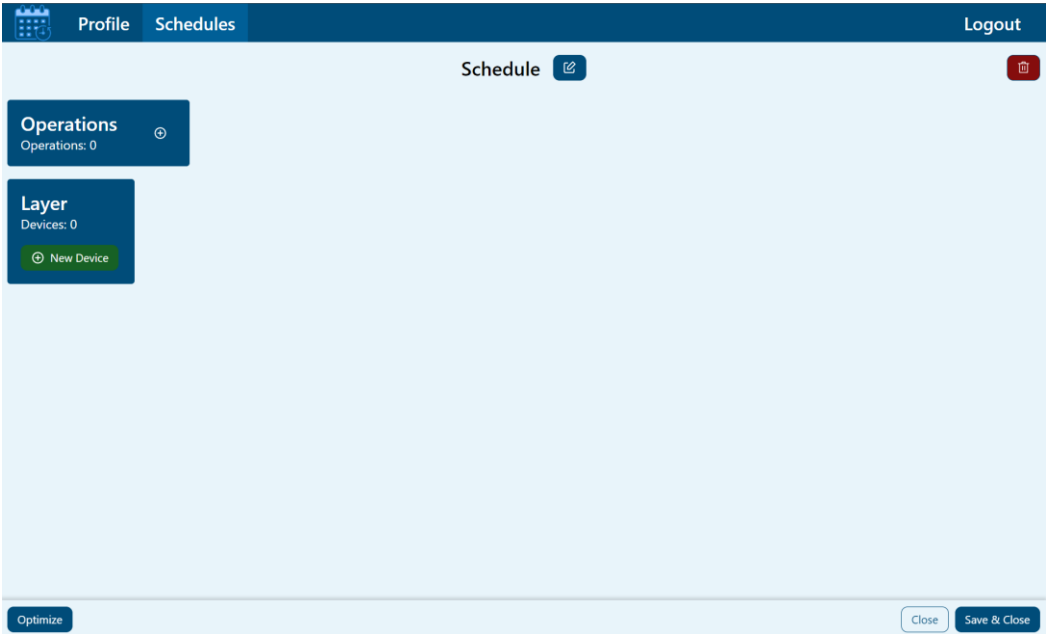


Рисунок 3.5 – Сторінка редагування однорівневого розкладу

Висновки до розділу

У даному розділі було проаналізовано якість отриманого програмного забезпечення, а саме перевірено дотримання всіх поставлених функціональних та нефункціональних вимог.

Оцінка виконання нефункціональних вимог була проведена за такими поставленими характеристиками, як зручність графічного інтерфейсу застосунку та його сумісність з більшістю сучасним веб-браузерів. В результаті аналізу дотримання нефункціональних вимог було визначено, що всі наведені характеристики відповідають поставленим вимогам.

Дотримання функціональних вимог було перевірено за допомогою мануального тестування. Наведений набір тестів для даного виду тестування дає можливість протестувати всі поставлені функціональні вимоги. Помилки під час тестування не було виявлено.

					КПІ.ІП-9213.045490.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		78

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання всіх частин отриманого застосунку, а саме клієнтську та серверну частини веб-застосунку і оптимізаційні інструменти, було обрано сервіс Digital Ocean. Дана платформа дозволяє створювати хмарні віртуальні машини, підключатись до них та розгортати на них будь який тип програмного забезпечення[14].

Розгортання відбувається мануально на створеній віртуальній машині на хмарі. Для повноцінного розгортання необхідно запустити спеціальний скрипт `setup.sh`, який завантажить останню версію коду з репозиторію на платформі Github, встановить необхідні залежності для всіх частин застосунку та запустить веб застосунок. Для того, щоб зупинити роботу застосунку, необхідно запустити скрипт `stop.sh`.

Для вільного доступу до застосунку у мережі Інтернет для публічний IP віртуальної машини було додано доменне ім'я `https://schedule-project.tech`. Також для шифрування даних при передачі їх мережею Інтернет для клієнтської та серверної частини було додано SSL-сертифікати, які дозволяють використовувати захищений метод передачі даних в Інтернеті HTTPS.

Роботу розгорнутого застосунку на віртуальній машині на хмарній платформі Digital Ocean продемонстровано на рисунку 4.1.

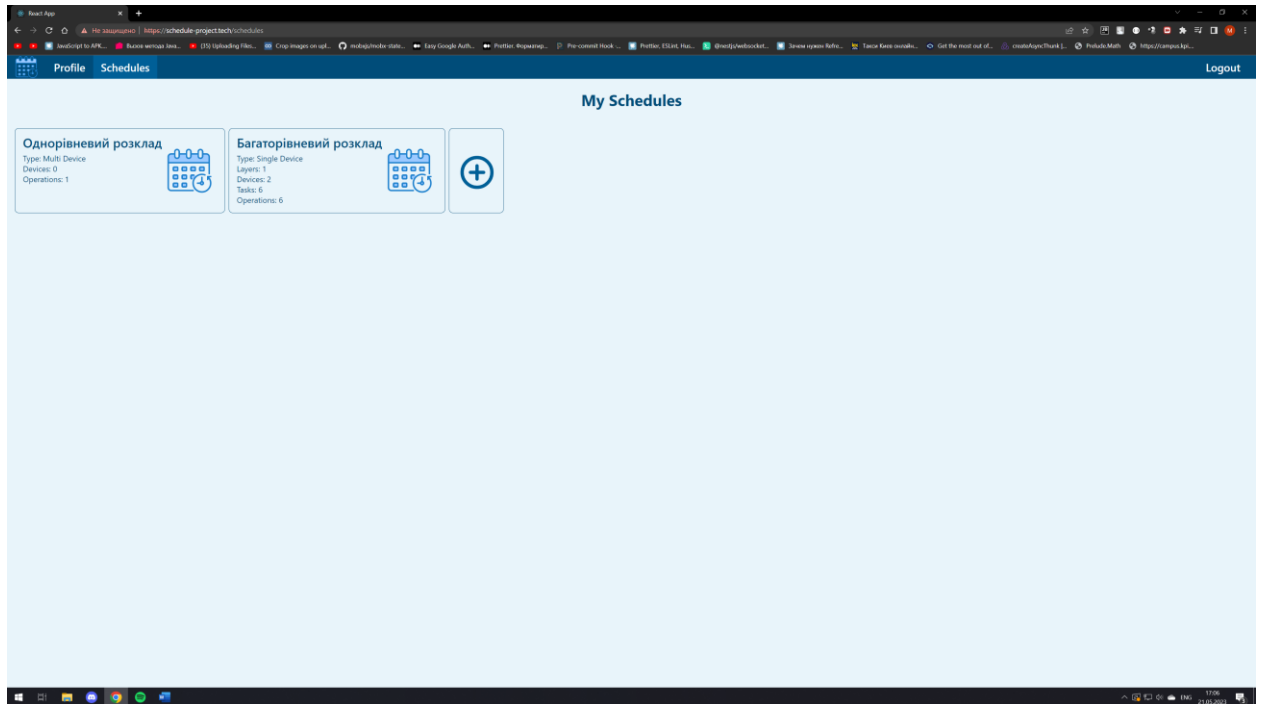


Рисунок 4.1 – Робота розгорнутого програмного забезпечення

4.2 Підтримка програмного забезпечення

Користувачі даного програмного забезпечення повинні отримувати його нові версії у майбутньому. Для оновлення програмного забезпечення адміністратор ресурсу, де розгортається застосунок, повинен зупинити застосунок за допомогою скрипту `stop.sh` та запустити його за допомогою `setup.sh`. Під час запуску застосунку остання версія продукту буде завантажена, встановлена та буде доступна у мережі Інтернет.

```
root@schedule-vm:~/schedule-project# ./stop.sh
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sigspec]
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sigspec]
root@schedule-vm:~/schedule-project# ./setup.sh
From https://github.com/MedvedMichael/schedule-project
* branch          master      -> FETCH_HEAD
Already up to date.
yarn install v1.22.19
[1/4] Resolving packages...
success Already up-to-date.
Done in 1.41s.
yarn run v1.22.19
yarn run v1.22.19
$ yarn workspace server run start:dev
$ yarn workspace client run start
$ react-scripts start
$ nest start --watch
```

Рисунок 4.2 – Оновлення застосунку та його успішний перезапуск

Висновки до розділу

У даному розділі було розглянуто процеси розгортання отриманого програмного забезпечення та його підтримка. Для розгортання застосунку було використано можливості хостингової платформи Digital Ocean. Запуск веб-застосунку здійснюється за допомогою спеціального скрипту.

Підтримка програмного забезпечення застосунку проводиться у ручному режимі адміністратором платформи, де розміщений додаток, за допомогою спеціальних скриптів.

					КПІ.ІП-9213.045490.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		81

ВИСНОВКИ

В рамках дипломного проєкту було реалізовано програмне забезпечення для створення календарних планувань IT-проєктів та їх оптимізації за допомогою сучасних алгоритмів.

Перш за все було проаналізовано предметну область та визначено важливі моменти, які враховувались у проєктуванні системи. Також було сформовано функціональні та нефункціональні вимоги до програмного продукту, проаналізовано аналоги та обрано технологічний стек для розробки.

Реалізований застосунок складається з веб-застосунку, який включає в себе клієнтську та серверну частини, та оптимізаційних інструментів. Всі частини веб-додатку написані на мову програмування TypeScript, а оптимізаційні інструменти реалізовані за допомогою мови програмування Python. Отриманий застосунок надає користувачу можливість за допомогою графічного інтерфейсу створювати календарні планування, редагувати всі їх частини та оптимізовувати отримані розклади за допомогою різних методів. Програмне забезпечення застосунку було проаналізовано на наявність вразливостей, що можуть впливати на безпеку даних.

Для виявлення проблем у програмному забезпеченні було проведене мануальне тестування додатку за допомогою поставленого набору тестів. Застосунок був протестований у таких сучасних браузерах, як Chrome, Firefox, Edge та Safari. Також було наведено детальний опис контрольного прикладу для тестування отриманого застосунку.

Після тестування застосунок було розгорнуто на віртуальній машині хмарного сервісу Digital Ocean. Також було детально описано процес підтримки та оновлення програмного забезпечення.

					КПІ.ІП-9213.045490.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		82

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Юсупова Н.И. Об одной классификации задач составления расписаний / Юсупова Н.И., Сметанина О.Н., Ахтариев А.А. // Вестник Уфимского государственного авиационного технического университета. – 2007 – 86-87 с.
- 2) Жданова Е.Г. Методичные указания к самостоятельной работе по дисциплине “Теория расписаний” / Жданова Е.Г. – 2001 – 6-7 с.
- 3) Лазарев О.А. Теория расписаний. Задачи и алгоритмы / Лазарев О.А., Гафаров С.Р. – 2011 – 23-35 с.
- 4) Данилин М.Н. Аналитическая классификация моделей задач теории расписаний / Данилин М.Н. – 2010
- 5) Павлов О.А. Составление допустимого расписания выполнения работ на одном приборе оптимального по критерию минимизации суммарного опережения работ / Павлов О.А., Халус О.А. – 2014
- 6) Павлов О.А. Труднорешаемые задачи комбинаторной оптимизации в планировании и принятии решений / Павлов О.А., Згуровский М.З. – 2016 – 17-31 с.
- 7) Павлов О.А. Автоматизация планирования в дискретных системах с заданой структурой технологических процессов / Павлов О.А., Халус О.А., Борисенко I.B.
- 8) Теленик С.Ф. Высокоэффективные алгоритмы составления расписаний на идентичных параллельных приборах, содержащие достаточные признаки оптимальности получаемых решений / Теленик С.Ф., Павлов А.А., Мисюра Е.Б., Мельников О.В., Халус Е.А.
- 9) Посібник по продукту Atlassian Jira Електронний ресурс. – URL: <https://www.atlassian.com/ru/software/jira/guides/getting-started/introduction> (дата звернення: 15.03.2023)
- 10) Документація по Atlassian Trello Електронний ресурс – URL: <https://trello.com/ru> (дата звернення: 15.03.2023)

- 11) Документація по мові програмування Python Електронний ресурс – URL: <https://www.python.org/doc/> (дата звернення: 15.03.2023)
- 12) C++ Language Reference Електронний ресурс – URL: <https://learn.microsoft.com/en-us/cpp/cpp/cpp-language-reference?view=msvc-170> (дата звернення: 15.03.2023)
- 13) The R Project for Statistical Computing Електронний ресурс – URL: <https://www.r-project.org/about.html> (дата звернення: 15.03.2023)
- 14) Getting Started with Digital Ocean Електронний ресурс – URL: <https://docs.digitalocean.com/products/getting-started/> (дата звернення: 15.03.2023)
- 15) React.js Api Reference Електронний ресурс – URL: <https://react.dev/reference/react> (дата звернення: 15.03.2023)
- 16) AngularJS Api Docs Електронний ресурс – URL: <https://docs.angularjs.org/api> (дата звернення: 15.03.2023)
- 17) Getting Starter with Vue.js Електронний ресурс – URL: <https://cn.vuejs.org/guide/quick-start.html> (дата звернення: 15.03.2023)
- 18) NestJS – Documentation Електронний ресурс – URL: <https://docs.nestjs.com/> (дата звернення: 15.03.2023)

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



Ім'я користувача:
Лісовиченко Олег Іванович

Дата перевірки:
02.06.2023 14:28:48 EEST

Дата звіту:
02.06.2023 14:46:50 EEST

ID перевірки:
1015394126

Тип перевірки:
Doc vs Internet + Library

ID користувача:
76913

Назва документа: ІП-92_Медведев_ПЗ

Кількість сторінок: 80 Кількість слів: 11568 Кількість символів: 92550 Розмір файлу: 10.01 MB ID файлу: 1015058537

4.96%
Схожість

Найбільша схожість: 1.17% з джерелом з Бібліотеки (ID файлу: 1015035555)

2.78% Джерела з Інтернету	168	Сторінка 82
4.39% Джерела з Бібліотеки	227	Сторінка 84

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	34
------------------	----

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

**Програмне забезпечення для календарного планування виконання ІТ-
проєктів**

Текст програми

КПІ. ІІ-9213.045490.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена ХАЛУС

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Михайло МЕДВЕДСВ

Київ – 2023

Файл math/base.py

```
class Operation:
    def __init__(self, duration: float, deadline: float = None, task: 'Task'
= None,
                device: 'Device' = None,
                start: float = None, id: str = None, name: str = None):
        self.duration = duration
        self.deadline = deadline
        self.task = task
        self.device = device
        self.id = id
        self.name = name
        self.start = start

        self.lag: float | None = None
        self.advance: float | None = None

    @classmethod
    def copy(cls, operation: 'Operation'):
        return cls(operation.duration, operation.deadline, operation.task,
operation.device,
                    operation.start, operation.id, operation.name)

    def __str__(self) -> str:
        return 'Operation' + (' ' + self.name if self.name else '') + ' -
duration: ' + str(self.duration) + \
            (' deadline: ' + str(self.deadline) if self.deadline is not None
else '') + \
            (' start: ' + str(self.start) + ' end: ' + str(self.end) + '
lag:' + str(self.lag) + ' advance: ' + str(
                self.advance) if self.start is not None else '')

    @property
    def end(self) -> float:
        return self.start + self.duration if self.start is not None else None

    @property
    def start(self) -> float:
        return self.__start

    @start.setter
    def start(self, start_time: float):
        self.__start = start_time
        if start_time is not None and self.deadline is not None:
            diff = self.start + self.duration - self.deadline
            self.lag = max(diff, 0)
            self.advance = max(-diff, 0)
        else:
            self.lag = None
            self.advance = None

class Task:
    def __init__(self, operations: list[Operation], deadline: float = None,
id: str = None, name: str = None):
        self.id = id
        self.name = name
        self.deadline = deadline

        self.__operations = operations
        for operation in operations:
```

					КПІ.ІП-9213.045490.03.12	Арк.
						2
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        operation.task = self

    @classmethod
    def copy(cls, task: 'Task'):
        return cls([Operation.copy(operation) for operation in
task.operations], task.id, task.name)

    @property
    def operations(self):
        return self.__operations

    @operations.setter
    def operations(self, operations: list[Operation]):
        self.__operations = operations

class Device:
    def __init__(self, operations: list[Operation] = None, start: float =
None, id: str = None, name: str = None):
        self.id = id
        self.name = name
        self.operations: list[Operation] = \
            operations if operations is not None else []

        for operation in self.operations:
            operation.device = self

        self.start = start

    @classmethod
    def copy(cls, device: 'Device'):
        return cls([Operation.copy(operation) for operation in
device.operations], device.start, device.id, device.name)

    def __str__(self) -> str:
        return 'Device: \n' + str('\n'.join([str(e) for e in
self.operations])) + \
            ('\nStart: ' + str(self.start) + ', End: ' + str(self.end) if
self.start is not None else '') + \
            ('\nTotal lag: ' + str(self.lag) +
            '\nTotal advance: ' + str(self.advance)
            if self.start is not None and self.lag is not None and
self.advance is not None else '')

    @property
    def lag(self) -> float | None:
        try:
            return sum([execution.lag for execution in self.operations])
        except TypeError:
            return None

    @property
    def advance(self) -> float | None:
        try:
            return sum([execution.advance for execution in self.operations])
        except TypeError:
            return None

    @property
    def operations(self) -> list[Operation]:
        return self.__executions

    @operations.setter
    def operations(self, operations: list[Operation]):

```

```

        self.__executions = operations
        for execution in self.operations:
            execution.device = self

        if len(operations) != 0:
            self.start = operations[0].start
            self.end = operations[-1].start + operations[-1].duration \
                if operations[-1].start is not None else None
        else:
            self.start = None
            self.end = None

    @property
    def start(self) -> float:
        return self.__start

    @start.setter
    def start(self, start_time: float):
        self.__start = start_time
        if start_time is None:
            for execution in self.operations:
                execution.start = None
            self.end = None
            return

        for execution in self.operations:
            execution.start = start_time
            start_time += execution.duration

        if len(self.operations) != 0 and self.start is not None:
            self.end = self.operations[-1].start + self.operations[-1].duration

    @dataclass(eq=False)
    class Layer:
        devices: list[Device]
        id: Optional[str] = None
        name: Optional[str] = None

        @classmethod
        def copy(cls, layer: 'Layer'):
            return cls([Device.copy(device) for device in layer.devices],
                layer.id, layer.name)

        @property
        def delta_devices(self):
            max_device = max(self.devices, key=lambda device: device.end)
            min_device = min(self.devices, key=lambda device: device.end)
            return max_device.end - min_device.end

    @dataclass(eq=False)
    class Schedule:
        layers: list[Layer]
        tasks: list[Task]
        id: Optional[str] = None
        name: Optional[str] = None

        @classmethod
        def copy(cls, schedule: 'Schedule'):
            return cls([Layer.copy(layer) for layer in schedule.layers],
                [Task.copy(task) for task in schedule.tasks],
                schedule.id,

```

```

        schedule.name)

def optimize_multi_layer(self, use_r_max: bool):
    multi_device = importlib.import_module('multi_device')
    return multi_device.get_multi_layer_schedule(self, use_r_max)

def optimize_multi_device(self):
    multi_device = importlib.import_module('multi_device')
    return multi_device.get_multi_device_schedule_A0(self)
    }
gameImport_t;

```

Файл math/call.py

```

class ArgumentType(Enum):
    Schedule = 'schedule'
    Device = 'device'

def to_float(num):
    return float(num) if num is not None else None

def parse_schedule(schedule_str: str) -> Schedule:
    obj_schedule = json.loads(schedule_str)
    tasks: list[Task] = []
    operations: list[Operation] = []

    for obj_task in obj_schedule['tasks']:
        current_operations = [
            Operation(duration=op.get('duration'),
                deadline=op.get('deadline'), id=op.get('id'), name=op.get('name'))
            for op in obj_task.get('operations')]

        task = Task(operations=current_operations,
            deadline=obj_task.get('deadline'), id=obj_task.get('id'),
            name=obj_task.get('name'))
        tasks.append(task)

        operations += current_operations

    layers: list[Layer] = []

    for obj_layer in obj_schedule['layers']:
        current_devices: list[Device] = []

        for obj_device in obj_layer['devices']:

```

					КПІ.ІП-9213.045490.03.12	Арк.
Вмін.	Арк.	№ докум.	Підп.	Дата.		5

```

        operation_ids = [op.get('id')
                           for op in obj_device.get('operations')]
        device_operations = [
            operation for operation in operations if operation.id in
operation_ids
        ]
        current_devices.append(
            Device(operations=device_operations, id=obj_device.get('id'),
name=obj_device.get('name'),
                    start=obj_device.get('start')))

        layers.append(Layer(devices=current_devices,
                             id=obj_layer.get('id'),
name=obj_layer.get('name')))

    return Schedule(layers=layers, tasks=tasks,
                    id=obj_schedule.get('id'), name=obj_schedule.get('name'))

def stringify_schedule(schedule: Schedule):
    obj_tasks = [
        {
            'id': task.id,
            'name': task.name,
            'scheduleId': schedule.id,
            'deadline': task.deadline,
            'operations': [{
                'id': operation.id,
                'name': operation.name,
                'duration': operation.duration,
                'deadline': operation.deadline,
                'deviceId': operation.device.id,
                'taskId': task.id,
                'start': operation.start,
                'end': operation.end
            } for operation in task.operations]
        }
        for task in schedule.tasks
    ]

    obj_schedule = {
        'id': schedule.id,
        'name': schedule.name,

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		6

```

'tasks': obj_tasks,
'layers': [{
    'id': layer.id,
    'name': layer.name,
    'scheduleId': schedule.id,
    'devices': [{
        'id': device.id,
        'name': device.name,
        'layerId': layer.id,
        'start': device.start,
        'end': device.end,
        'operations': [{
            'id': operation.id,
            'name': operation.name,
            'duration': operation.duration,
            'deadline': operation.deadline,
            'deviceId': operation.device.id,
            'taskId': operation.task.id,
            'start': operation.start,
            'end': operation.end
        } for operation in device.operations]
    }
    for device in layer.devices
],
}
for layer in schedule.layers
]
}

return json.dumps(obj_schedule)

```

```

def pre_get_multi_layer_schedule(obj_schedule: str, use_r_max: str):
    schedule = parse_schedule(obj_schedule)
    use_r_max = json.loads(use_r_max)
    result = get_multi_layer_schedule(schedule, use_r_max)
    print(stringify_schedule(result))

```

```

def get_multi_device_schedule(obj_schedule: str):
    schedule = parse_schedule(obj_schedule)
    result = get_multi_device_schedule_A0(schedule)
    print(stringify_schedule(result))

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		7

```

functions_by_names = {
    'get_multi_layer_schedule': pre_get_multi_layer_schedule,
    'get_multi_device_schedule': get_multi_device_schedule
}

try:
    functions_by_names[sys.argv[1]](*sys.argv[2:])
except Exception as error:
    print(error)

sys.stdout.flush()

```

Файл math/multi_device.py

```

def get_multi_layer_schedule(schedule: Schedule, use_r_max=True) -> Schedule:
    new_schedule = Schedule(layers=[], tasks=schedule.tasks,
                             name=schedule.name, id=schedule.id)
    for i in range(len(schedule.layers) - 1, -1, -1):
        algo = get_schedule_A2 if i == len(
            schedule.layers) - 1 and not use_r_max else get_schedule_A1
        layer = schedule.layers[i]
        scheduled_layer = Layer(devices=[algo(device)
                                          for device in layer.devices],
                                id=layer.id, name=layer.name)
        new_schedule.layers.insert(0, scheduled_layer)

    if i != 0:
        for device in scheduled_layer.devices:
            for operation in device.operations:
                task = operation.task
                prev_index = task.operations.index(operation) - 1
                prev_operation = task.operations[prev_index]
                prev_operation.deadline = operation.start

    return new_schedule

def make_permutation_1_1(lagged_device: Device, advanced_device: Device,
                          average_deadline: float) -> bool:
    for i in range(len(lagged_device.operations)):

```

					КПІ.ІП-9213.045490.03.12	Арк.
Вмін.	Арк.	№ докум.	Підп.	Дата.		8

```

        for j in range(len(advanced_device.operations)):
            theta = lagged_device.operations[i].duration -
advanced_device.operations[j].duration
            if theta <= 0:
                continue

            if theta <= lagged_device.end - average_deadline:
                lagged_operation = lagged_device.operations.pop(i)
                advanced_operation = advanced_device.operations.pop(j)
                lagged_device.operations.insert(i, advanced_operation)
                advanced_device.operations.insert(j, lagged_operation)
                return True

    return False

def make_permutation_1_2(lagged_device: Device, advanced_device: Device,
average_deadline: float) -> bool:
    for i in range(len(lagged_device.operations)):
        for j in range(len(advanced_device.operations)):
            for k in range(j + 1, len(advanced_device.operations)):
                theta = lagged_device.operations[i].duration - (
                    advanced_device.operations[j].duration +
advanced_device.operations[k].duration)
                if theta <= 0:
                    continue

                if theta <= lagged_device.end - average_deadline:
                    lagged_operation = lagged_device.operations.pop(i)
                    advanced_operation_1 = advanced_device.operations.pop(j)
                    advanced_operation_2 = advanced_device.operations.pop(k -
1)

                    lagged_device.operations.insert(i, advanced_operation_1)
                    lagged_device.operations.insert(i, advanced_operation_2)
                    advanced_device.operations.insert(j, lagged_operation)

                return True

    return False

def make_permutation_2_1(lagged_device: Device, advanced_device: Device,
average_deadline: float) -> bool:

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		9

```

    for i in range(len(lagged_device.operations)):
        for j in range(i + 1, len(lagged_device.operations)):
            for k in range(len(advanced_device.operations)):
                theta = lagged_device.operations[i].duration +
lagged_device.operations[j].duration - \
                    advanced_device.operations[k].duration
                if theta <= 0:
                    continue

                if theta <= lagged_device.end - average_deadline:
                    lagged_operation_1 = lagged_device.operations.pop(i)
                    lagged_operation_2 = lagged_device.operations.pop(j - 1)
                    advanced_operation = advanced_device.operations.pop(k)
                    lagged_device.operations.insert(i, advanced_operation)
                    advanced_device.operations.insert(k, lagged_operation_1)
                    advanced_device.operations.insert(k, lagged_operation_2)

                return True

    return False

def make_permutation_2_2(lagged_device: Device, advanced_device: Device,
average_deadline: float) -> bool:
    for i in range(len(lagged_device.operations)):
        for j in range(i + 1, len(lagged_device.operations)):
            for k in range(len(advanced_device.operations)):
                for x in range(k + 1, len(advanced_device.operations)):
                    theta = (lagged_device.operations[i].duration +
lagged_device.operations[j].duration) - \
                        (advanced_device.operations[k].duration +
advanced_device.operations[x].duration)
                    if theta <= 0:
                        continue

                    if theta <= lagged_device.end - average_deadline:
                        lagged_operation_1 = lagged_device.operations.pop(i)
                        lagged_operation_2 = lagged_device.operations.pop(j -
1)

                        advanced_operation_1 =
advanced_device.operations.pop(k)
                        advanced_operation_2 =
advanced_device.operations.pop(x - 1)

```

```

        lagged_device.operations.insert(i,
advanced_operation_1)
        lagged_device.operations.insert(j,
advanced_operation_2)
        advanced_device.operations.insert(k,
lagged_operation_1)
        advanced_device.operations.insert(x,
lagged_operation_2)

        return True

    return False

def get_multi_device_heuristic_schedule(schedule: Schedule):
    operations = [task.operations[0] for task in schedule.tasks]

    operations.sort(key=lambda op: op.duration, reverse=True)

    new_devices = [Device.copy(device) for device in
schedule.layers[0].devices]
    new_devices.sort(key=lambda device: device.start)

    schedule.layers[0].devices = new_devices

    for operation in operations:
        min_end_time_device = min(new_devices, key=lambda device: device.end
if device.end else 0)
        min_end_time_device.operations.append(operation)
        min_end_time_device.start = min_end_time_device.start

    return schedule

def get_multi_device_schedule_A0(schedule: Schedule):
    new_schedule = get_multi_device_heuristic_schedule(schedule)
    new_devices = new_schedule.layers[0].devices
    operations = [task.operations[0] for task in new_schedule.tasks]

    average_deadline = (sum([device.start for device in new_devices]) + sum(
        [operation.duration for operation in operations])) / len(new_devices)

    while True:

```

```

temp_end = new_devices[0].end

if all([device.end == temp_end for device in new_devices]):
    new_schedule.layers[0].devices = new_devices
    break

advanced_devices = [device for device in new_devices if device.end <
average_deadline]
lagged_devices = [device for device in new_devices if device.end >
average_deadline]

max_lagged_device = max(lagged_devices, key=lambda device: device.end
- average_deadline)
found_permutation = False

for advanced_device in advanced_devices:
    if make_permutation_1_1(max_lagged_device, advanced_device,
average_deadline) or \
        make_permutation_1_2(max_lagged_device, advanced_device,
average_deadline) or \
        make_permutation_2_1(max_lagged_device, advanced_device,
average_deadline) or \
        make_permutation_2_2(max_lagged_device, advanced_device,
average_deadline):
        found_permutation = True
        break

for device in new_devices:
    device.start = device.start

if not found_permutation:
    break

for device in new_schedule.layers[0].devices:
    for operation in device.operations:
        operation.device = device

return new_schedule

```

Файл math/single_device.py

```

def __get_max_start_time(operations: list[Operation]):
    max_deadline = max([operation.deadline for operation in operations])
    total_duration = sum([operation.duration for operation in operations])

```

					КПІ.ІП-9213.045490.03.12	Арк.
						12
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

return max_deadline - total_duration

def __get_heuristic_schedule(device: Device) -> Device:
    new_operations = [operation for operation in device.operations]
    new_operations.sort(key=lambda x: x.deadline)
    start_time = __get_max_start_time(new_operations)
    return Device(operations=new_operations, start=start_time, id=device.id,
name=device.name)

def get_schedule_A(device: Device, start_time: float = None) -> Device:
    scheduled_device = __get_heuristic_schedule(device)
    max_lag = max([execution.lag for execution in
scheduled_device.operations])
    new_start_time = scheduled_device.start - \
        max_lag if start_time is None else start_time

    scheduled_device.start = new_start_time
    return scheduled_device

def get_schedule_A1(device: Device, start_time: float = None) -> Device:
    scheduled_device = get_schedule_A(device, start_time)
    r_max = scheduled_device.start
    executions = scheduled_device.operations
    end_time = scheduled_device.end
    executions_after_deadline = [
        execution for execution in executions if execution.deadline >=
end_time]
    if len(executions_after_deadline) == 0:
        return scheduled_device

    new_execution = min(executions_after_deadline, key=lambda x: x.duration)

    new_execution.start = end_time - new_execution.duration
    executions.remove(new_execution)
    executions.append(new_execution)

    i = len(executions) - 2
    while i > 0:
        last_start_moment = executions[i + 1].start

```

```

        executions_after_deadline = [execution for execution in executions[:i
+ 1] if

                                execution.deadline >= last_start_moment]
    if len(executions_after_deadline) > 0:
        new_execution = min(executions_after_deadline,
                                key=lambda x: x.duration)
        new_execution.start = last_start_moment - new_execution.duration
        executions.remove(new_execution)
        executions.insert(i, new_execution)

    i -= 1

    scheduled_device.start = r_max

    return scheduled_device

def get_schedule_A2(device: Device) -> Device:
    oldest_schedule = get_schedule_A1(Device.copy(device), 0)
    current_schedule = get_schedule_A1(device)
    found_schedules = [(current_schedule.start, current_schedule.advance)]
    current_r = current_schedule.start

    while current_schedule.advance + len(device.operations) *
current_schedule.start != oldest_schedule.advance:
        delta_l = min([
            current_schedule.operations[i].end -
            max([
                operation.deadline
                for operation in current_schedule.operations
                if operation.duration <
current_schedule.operations[i].duration and
                    operation.deadline <
current_schedule.operations[i].end
                ] + [-float('inf')])
            for i in range(1, len(current_schedule.operations))
        ])
        if delta_l == float('inf'):
            break
        current_r -= delta_l
        current_schedule = get_schedule_A1(
            current_schedule, start_time=current_r)

```

```

        found_schedules.append((current_schedule.start,
current_schedule.advance))

result_start = min(found_schedules, key=lambda x: x[1])[0]
return get_schedule_A1(
        current_schedule, start_time=result_start)

```

Файл server/db/tables.sql

```

CREATE TABLE IF NOT EXISTS "Users"(
    "userId" UUID PRIMARY KEY,
    "firstName" VARCHAR(255),
    "lastName" VARCHAR(255),
    "email" VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS "AuthProviders"(
    "userId" UUID REFERENCES "Users" ON DELETE CASCADE,
    "authName" VARCHAR (100) NOT NULL,
    "password" TEXT,
    PRIMARY KEY("userId", "authName")
);

CREATE TABLE IF NOT EXISTS "RefreshSessions" (
    "refreshToken" UUID PRIMARY KEY,
    "userId" UUID REFERENCES "Users" ON DELETE CASCADE,
    "userAgent" VARCHAR(200) NOT NULL,
    "fingerprint" VARCHAR(200) NOT NULL,
    "expiresIn" BIGINT NOT NULL,
    "createdAt" TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS "Schedules" (
    "id" UUID PRIMARY KEY,
    "name" VARCHAR(255),
    "ownerId" UUID REFERENCES "Users" ON DELETE CASCADE,
    "type" VARCHAR(20),
    "timestamp" TIMESTAMP WITH TIME ZONE
);

CREATE TABLE IF NOT EXISTS "ScheduleTasks" (
    "id" UUID PRIMARY KEY,
    "name" VARCHAR(255),
    "scheduleId" UUID REFERENCES "Schedules" ON DELETE CASCADE,
    "deadline" NUMERIC
);

```

```

CREATE TABLE IF NOT EXISTS "ScheduleLayers" (
    "id" UUID PRIMARY KEY,
    "name" VARCHAR(255),
    "index" INT,
    "scheduleId" UUID REFERENCES "Schedules" ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS "ScheduleDevices" (
    "id" UUID PRIMARY KEY,
    "name" VARCHAR(255),
    "index" INT,
    "layerId" UUID REFERENCES "ScheduleLayers" ON DELETE CASCADE,
    "start" NUMERIC
);

CREATE TABLE IF NOT EXISTS "ScheduleOperations" (
    "id" UUID PRIMARY KEY,
    "name" VARCHAR(255),
    "taskId" UUID REFERENCES "ScheduleTasks" ON DELETE CASCADE,
    "deviceId" UUID REFERENCES "ScheduleDevices" ON DELETE CASCADE,
    "index" INT,
    "deviceIndex" INT,
    "duration" NUMERIC,
    "deadline" NUMERIC,
    "start" NUMERIC
);

```

Файл server/src/modules/auth/auth.service.ts

```

@Injectable()
export class AuthService {
    private maxNumberOfSessions = 10;

    constructor(
        private scriptService: ScriptService,
        private jwtService: JwtService,
        private usersService: UsersService,
        private refreshSessionsService: RefreshSessionsService,
        private authProvidersService: AuthProvidersService,
    ) {}

    async register(
        {
            password: pass,
            email,

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		16

```

        firstName,
        lastName,
        astrologicalToken,
    }: RegisterParams,
    authName: AuthProviderName,
    userAgent = '',
): Promise<{ accessToken: string; refreshToken: string }> {
    const oldUser = await this.usersService.findByEmail(email);
    if (oldUser) {
        throw new BadRequestException('User with such email exists already');
    }
    const password =
        authName === 'local'
            ? await this.scrptService.hash(pass as string)
            : undefined;
    const userId = uuid();

    await this.usersService.registerUser({
        email,
        firstName,
        lastName,
        userId,
    });

    const { accessToken, refreshToken } = await
this.createNewRefreshSession({
        userId,
        userAgent,
        fingerprint: astrologicalToken,
    });

    await this.authProvidersService.saveProvider({
        userId,
        authName,
        password,
    });
    return { accessToken, refreshToken };
}

async login(
    { email, password, astrologicalToken: fingerprint }: LoginParams,
    userAgent: string,
    authName: AuthProviderName,

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		17

```

): Promise<AuthTokensPair> {
    const user = await this.userService.findByEmail(email);
    if (!user) {
        throw new NotFoundException();
    }
    const { userId: userId } = user;
    const provider = await this.authProvidersService.findOne(userId,
authName);
    if (!provider) {
        throw new NotFoundException();
    }
    const isValid =
        authName === 'local'
            ? await this.scrptService.verify(
                password as string,
                provider.password as string,
            )
            : true;

    if (!isValid) {
        throw new BadRequestException();
    }

    return await this.createNewRefreshSession({
        userId,
        userAgent,
        fingerprint,
    });
}

async checkSession(
    refreshToken: string,
    fingerprint: string,
): Promise<Session> {
    const session = await this.refreshSessionsService.findOne(refreshToken);
    if (!session) {
        throw new UnauthorizedException('INVALID_TOKEN');
    }

    const { fingerprint: oldFingerprint, createdAt, expiresIn } = session;
    await this.refreshSessionsService.delete(refreshToken);

    const expiredDate = new Date(createdAt.getTime() + +expiresIn);

```

```

const now = new Date();
if (
    expiredDate.getTime() <= now.getTime() ||
    oldFingerprint !== fingerprint
) {
    throw new UnauthorizedException('INVALID_REFRESH_SESSION');
}

return session;
}

async refreshTokens(
    refreshToken: string,
    fingerprint: string,
): Promise<AuthTokensPair> {
    const session = await this.checkSession(refreshToken, fingerprint);
    const { userId, userAgent } = session;

    return await this.createNewRefreshSession({
        userId,
        userAgent,
        fingerprint: fingerprint,
    });
}

async createNewRefreshSession({
    userId,
    fingerprint,
    userAgent,
}): RefreshSessionDTO): Promise<AuthTokensPair> {
    const accessToken = this.jwtService.sign({ userId });
    const refreshToken = uuid();

    const sessions = (await this.refreshSessionsService.find(userId)).sort(
        (a, b) => a.createdAt.getTime() - b.createdAt.getTime(),
    );

    const oldSameSession = sessions.find((s) => s.fingerprint ===
fingerprint);
    if (oldSameSession) {
        await this.refreshSessionsService.delete(oldSameSession.refreshToken);
    }
}

```

```

        if (sessions.length > this.maxNumberOfSessions - 1) {
            await this.refreshSessionsService.delete(sessions[0].refreshToken);
        }
        await this.refreshSessionsService.saveSession({
            refreshToken,
            userId,
            fingerprint,
            expiresIn: 5.184e9,
            createdAt: new Date(),
            userAgent,
        });
        return { accessToken, refreshToken };
    }

    async googleAuth(
        userData: GoogleResponse,
        userAgent: string,
        astrologicalToken: string,
    ): Promise<AuthTokensPair> {
        try {
            return await this.login(
                {
                    email: userData.email,
                    astrologicalToken,
                },
                userAgent ? userAgent : '',
                'google',
            );
        } catch (error) {
            if (error.status === 400) {
                throw new BadRequestException();
            }
            return await this.register(
                {
                    email: userData.email,
                    firstName: userData.given_name,
                    lastName: userData.family_name,
                    astrologicalToken,
                },
                'google',
                userAgent,
            );
        }
    }

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		20

```

    }

    async logout(refreshToken: string): Promise<void> {
        await this.refreshSessionsService.delete(refreshToken);
    }
}

```

Файл server/src/modules/auth/auth.module.ts

```

@Module({
    imports: [
        forwardRef(() => UsersModule),
        JwtModule.register({
            secret: process.env.JWT_SECRET,
            signOptions: { expiresIn: '30m' },
        }),
    ],
    providers: [
        AuthService,
        ScryptService,
        UsersService,
        PgService,
        JwtStrategy,
        RefreshSessionsService,
        AuthProvidersService,
    ],
    controllers: [AuthController],
    exports: [
        AuthService,
        ScryptService,
        UsersService,
        PgService,
        JwtModule,
        JwtStrategy,
        RefreshSessionsService,
        AuthProvidersService,
    ],
})
export class AuthModule {}

```

Файл server/src/modules/auth/auth.controller.ts

					КПІ.ІП-9213.045490.03.12	Арк.
Вмін.	Арк.	№ докум.	Підп.	Дата.		21

```

const userAgentName = 'user-agent';
@Controller(AuthRoute)
export class AuthController {
    constructor(private authService: AuthService) {}

    @Post(RegisterRoute)
    @UsePipes(new ValidationPipe({ transform: true }))
    async register(
        @Body() registerDTO: RegisterParams,
        @Res({ passthrough: true }) response: Response,
        @Req() { headers }: Request,
    ): Promise<void> {
        const userAgent = String(headers[userAgentName]);
        const pair = await this.authService.register(
            registerDTO,
            'local',
            userAgent,
        );

        sendTokensPair(response, pair);
    }

    @Post(LoginRoute)
    @UsePipes(new ValidationPipe({ transform: true }))
    async login(
        @Body() authDTO: LoginParams,
        @Res({ passthrough: true }) response: Response,
        @Req() { headers }: Request,
    ): Promise<void> {
        const userAgent = String(headers[userAgentName]);
        const pair = await this.authService.login(authDTO, userAgent, 'local');
        sendTokensPair(response, pair);
    }

    @Post(RefreshTokensRoute)
    async refreshTokens(
        @Body() { astrologicalToken }: RefreshTokensParams,
        @Req() request: Request,
        @Res({ passthrough: true }) response: Response,
    ): Promise<void> {
        const { refreshToken: oldToken } = request.cookies;
        if (!oldToken) throw new UnauthorizedException('No refresh token');
        const pair = await this.authService.refreshTokens(

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		22

```

        oldToken,
        astrologicalToken,
    );
    sendTokensPair(response, pair);
}

@Post(GoogleRoute)
async googleAuth(
    @Body() body: GoogleParams,
    @Req() { headers }: Request,
    @Res({ passthrough: true }) response: Response,
): Promise<void> {
    const userData = (await fetch(
        process.env.GOOGLE_RESOURCE_ID + body.tokenId,
    ).then((res) => res.json())) as GoogleResponse;

    if (!userData.email_verified) {
        throw new BadRequestException('Not confirmed google account!');
    }
    const userAgent = String(headers[userAgentName]);
    const pair = await this.authService.googleAuth(
        userData,
        userAgent,
        body.astrologicalToken,
    );
    sendTokensPair(response, pair);
}

@Post(LogoutRoute)
async logout(
    @Req() request: Request,
    @Res({ passthrough: true }) response: Response,
): Promise<void> {
    const { refreshToken } = request.cookies;
    await this.authService.logout(refreshToken);
    cleanCookies(response);
}
}

```

Файл server/src/modules/python/python.service.ts

```

@Injectable()
export class PythonService {
    call(funcName: string, ...args: any[]): Promise<Object> {

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		23

```

return new Promise((res, rej) => {
  const process = spawn('python', [
    '../math/call.py',
    funcName,
    ...args.map((arg) => JSON.stringify(arg)),
  ]);
  (process.stdout as any).on('data', (data: Buffer) => {
    try {
      const result = JSON.parse(data.toString());
      res(result);
    } catch (error) {
      console.error(data.toString());
    }
  });
  (process.stderr as any).on('data', (data: Buffer) => {
    console.error(data.toString());
  });
});
}
}

```

Файл server/src/modules/users/users.service.ts

```

@Injectable()
export class UsersService {
  tableName = 'Users';
  constructor(
    private pgService: PgService,
    private authProvidersService: AuthProvidersService,
    private scryptService: ScryptService,
    private refreshSessionsService: RefreshSessionsService,
    @Inject(forwardRef(() => AuthService))
    private authService: AuthService,
  ) {}

  async registerUser(registerData: RegisterData): Promise<void> {
    await this.pgService.create({
      values: [registerData],
      tableName: this.tableName,
    });
  }

  async findByEmail(email: string): Promise<User> {
    const user = await this.pgService.findOne<UserEntity>({

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		24

```

        tableName: this.tableName,
        where: { email },
    });
    if (!user) {
        return user;
    }
    const { firstName, lastName, userId: userId } = user;
    const authProviders = (await this.authProvidersService.find(userId)).map(
        (a) => a.authName,
    );
    return {
        userId: userId,
        firstName,
        lastName,
        email,
        authProviders,
    };
}

async findById(userId: string): Promise<User> {
    const user = await this.pgService.findOne<UserEntity>({
        tableName: this.tableName,
        where: { userId },
    });
    if (!user) {
        return user;
    }

    const { firstName, lastName, email } = user;
    const authProviders = (await this.authProvidersService.find(userId)).map(
        (a) => a.authName,
    );
    return {
        userId: userId,
        firstName,
        lastName,
        email,
        authProviders,
    };
}

async patchUser(userId: string, updates: UserUpdates): Promise<void> {
    await this.pgService.update<Partial<UserEntity>>>({

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		25

```

        tableName: this.tableName,
        updates: updates as Record<string, unknown>,
        where: { userId },
    });
}

async changePassword(
    userId: string,
    oldPassword: string,
    newPassword: string,
    fingerprint: string,
    userAgent: string,
): Promise<AuthTokensPair> {
    const user = await this.pgService.findOne<User>({
        tableName: this.tableName,
        where: { userId },
    });
    if (!user) throw new NotFoundException();

    const provider = await this.authProvidersService.findOne(
        user.userId,
        'local',
    );

    const isValid = await this.scryptService.verify(
        oldPassword,
        provider.password as string,
    );
    if (!isValid) {
        throw new BadRequestException();
    }

    await this.refreshSessionsService.deleteAll(userId);

    const newPass = await this.scryptService.hash(newPassword);
    await this.authProvidersService.changePassword(userId, newPass);

    return await this.authService.createNewRefreshSession({
        userId,
        fingerprint,
        userAgent,
    });
}

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		26

```
}
```

Файл server/src/modules/users/users.controller.ts

```
@Controller(UserRoute)
export class UsersController {
  constructor(private userService: UsersService) {}

  @UseGuards(JwtAuthGuard)
  @Get(MyProfileRoute)
  async getMyProfile(
    @Request() { user }: JwtValidationOutput,
  ): Promise<MyProfileRouteResponse> {
    return await this.userService.findById(user.userId);
  }

  @UseGuards(JwtAuthGuard)
  @Get(UserByIdRoute)
  async getUserGyId(
    @Param('id') userId: string,
  ): Promise<UserByIdRouteResponse> {
    return await this.userService.findById(userId);
  }

  @UseGuards(JwtAuthGuard)
  @Patch(PatchMyProfileRoute)
  async patchUserProfile(
    @Request() { user }: JwtValidationOutput,
    @Body('updates') updates: UserUpdates,
  ): Promise<void> {
    await this.userService.patchUser(user.userId, updates);
  }

  @UseGuards(JwtAuthGuard)
  @UsePipes(new ValidationPipe({ transform: true }))
  @Patch(ChangeMyPasswordRoute)
  async changePassword(
    @Request() { user }: JwtValidationOutput,
    @Body()
    { oldPassword, newPassword, astrologicalToken }:
    ChangeMyPasswordRouteProps,
    @Req() { headers }: ExpressRequest,
```

					КПІ.ІП-9213.045490.03.12	Арк.
						27
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

    @Res() res: Response,
  ): Promise<void> {
    const userAgent = String(headers['user-agent']);
    const pair = await this.usersService.changePassword(
      user.userId,
      oldPassword,
      newPassword,
      astrologicalToken,
      userAgent,
    );

    sendTokensPair(res, pair);
  }
}

```

Файл server/src/modules/schedules/schedules.service.ts

```

@Injectable()
export class SchedulesService {
  tableName = 'Schedules';
  layersTableName = 'ScheduleLayers';
  devicesTableName = 'ScheduleDevices';
  tasksTableName = 'ScheduleTasks';
  operationsTableName = 'ScheduleOperations';
  constructor(
    private pgService: PgService,
    private pythonService: PythonService,
  ) {}

  async createSchedule({
    id,
    name,
    ownerId,
    type,
    tasks,
    layers,
  }: Schedule): Promise<void> {
    await this.pgService.create<ScheduleDto>({
      tableName: this.tableName,
      values: [
        { id, name, ownerId, type, timestamp: new Date().toISOString() },

```

```

    ],
  });

  await Promise.all([
    this.pgService.create<LayerDto>({
      tableName: this.layersTableName,
      values: layers.map(({ id, name, scheduleId }, index) => ({
        id,
        name,
        scheduleId,
        index,
      })),
    }),
    this.pgService.create<TaskDto>({
      tableName: this.tasksTableName,
      values: tasks.map(({ id, name, scheduleId, deadline }) => ({
        id,
        name,
        scheduleId,
        deadline,
      })),
    }),
  ]);

  for (const layer of layers) {
    if (layer.devices.length !== 0) {
      await this.pgService.create<DeviceDto>({
        tableName: this.devicesTableName,
        values: layer.devices.map(({ id, name, layerId, start }, index) =>
        ({
          id,
          name,
          layerId,
          index,
          start,
        })),
      });
    }
  }

  for (const task of tasks) {
    await this.pgService.create<OperationDto>({
      tableName: this.operationsTableName,

```

```

        values: task.operations.map((operation, index) => {
            let foundDevice: Device;

            for (const layer of layers) {
                const device = layer.devices.find(
                    (device) => device.id === operation.deviceId,
                );
                if (device) {
                    foundDevice = device;
                    break;
                }
            }
            return {
                id: operation.id,
                name: operation.name,
                duration: operation.duration,
                deviceId: operation.deviceId,
                taskId: operation.taskId,
                start: operation.start,
                index,
                deviceIndex: operation.deviceId
                    ? foundDevice.operations.findIndex((op) => op.id ===
operation.id)
                    : null,
            };
        })),
    });
}

async findById(id: string) {
    const scheduleDto = await this.pgService.findOne<ScheduleDto>({
        tableName: this.tableName,
        where: { id },
    });
    if (!scheduleDto) {
        throw new NotFoundException();
    }

    const layerDtos = await this.pgService.find<LayerDto>({
        tableName: this.layersTableName,
        where: {
            scheduleId: id,

```

					КПІ.ІП-9213.045490.03.12	Арк.
Вмін.	Арк.	№ докум.	Підп.	Дата.		30

```

    },
    orderBy: 'index',
  });

const layers: Layer[] = layerDtos.map(({ index, ...layerDto }) => ({
  ...layerDto,
  devices: [],
}));

const taskDtos = await this.pgService.find<TaskDto>({
  tableName: this.tasksTableName,
  where: {
    scheduleId: id,
  },
});

const tasks: Task[] = taskDtos.map(({ deadline, ...taskDto }) => ({
  ...taskDto,
  deadline: deadline !== null ? +deadline : null,
  operations: [],
}));

for (const task of tasks) {
  const operationDtos = await this.pgService.find<OperationDto>({
    tableName: this.operationsTableName,
    where: {
      taskId: task.id,
    },
    orderBy: 'index',
  });
  task.operations = operationDtos.map(
    ({ index, start, duration, ...operation }) => ({
      ...operation,
      deadline: null,
      start: start ? +start : null,
      duration: +duration,
      // @ts-ignore
      end: start ? parseFloat(start) + parseFloat(duration) : null,
    }),
  );
}

for (const layer of layers) {
  layer.devices.push(

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		31

```

... (
  await this.pgService.find<DeviceDto>({
    tableName: this.devicesTableName,
    where: {
      layerId: layer.id,
    },
    orderBy: 'index',
  })
).map(({ index, start, ...deviceDto }) => ({
  ...deviceDto,
  start: start !== null ? +start : null,
  end: null,
  operations: [],
})),
);

for (const device of layer.devices) {
  const operationDtos = await this.pgService.find<OperationDto>({
    tableName: this.operationsTableName,
    where: {
      deviceId: device.id,
    },
    orderBy: 'deviceIndex',
  });
  device.operations = operationDtos.map(
    ({ index, start, duration, ...operation }) => ({
      ...operation,
      deadline: null,
      start: start ? +start : null,
      duration: +duration,
      // @ts-ignore
      end: start ? parseFloat(start) + parseFloat(duration) : null,
    }),
  );
}

const schedule: Schedule = { ...scheduleDto, tasks, layers };

return schedule;
}

async deleteById(scheduleId: string): Promise<void> {

```

```

    await this.pgService.delete<ScheduleDto>({
      tableName: this.tableName,
      where: { id: scheduleId },
      cascade: true,
    });
  }

  async updateSchedule(schedule: Schedule) {
    // TODO maybe add deep checking
    await this.deleteById(schedule.id);
    await this.createSchedule(schedule);
  }

  async getSchedulesByUser(userId: string): Promise<ScheduleStats[]> {
    const { rows } = await this.pgService.useQuery(
      `
      SELECT s.id, s.name, s.type, COUNT(DISTINCT sl.id) AS layers,
      COUNT(DISTINCT sd.id) AS devices, COUNT(DISTINCT st.id) AS tasks,
      COUNT(DISTINCT so.id) AS operations FROM "Schedules" s
      LEFT JOIN "ScheduleLayers" sl ON s.id = sl."scheduleId"
      LEFT JOIN "ScheduleDevices" sd ON sl.id = sd."layerId"
      LEFT JOIN "ScheduleOperations" so ON sd.id = so."deviceId"
      LEFT JOIN "ScheduleTasks" st ON s.id = st."scheduleId"
      WHERE s."ownerId" = $1
      GROUP BY s.id, s.name
      ORDER BY s.timestamp DESC
      `,
      [userId],
    );

    return rows.map((row) => ({
      id: row.id,
      name: row.name,
      type: row.type,
      counts: {
        operations: +row.operations,
        tasks: +row.tasks,
        layers: +row.layers,
        devices: +row.devices,
      },
    }));
  }
}

```

```

    async optimizeSingleDeviceSchedule(schedule: Schedule, useRmax = true) {
        // console.log(JSON.stringify(schedule));
        const res = (await this.pythonService.call(
            'get_multi_layer_schedule',
            schedule,
            useRmax,
        )) as Schedule;
        return { ...res, ownerId: schedule.ownerId, type: schedule.type };
    }

    async optimizeMultiDeviceSchedule(schedule: Schedule) {
        // console.log(JSON.stringify(schedule));

        const res = (await this.pythonService.call(
            'get_multi_device_schedule',
            schedule,
        )) as Schedule;
        return { ...res, ownerId: schedule.ownerId, type: schedule.type };
    }
}

```

Файл server/src/modules/schedules/schedules.controller.ts

```

@Controller(ScheduleRoute)
export class SchedulesController {
    constructor(private schedulesService: SchedulesService) {}

    @UseGuards(JwtAuthGuard)
    @Post()
    async createSchedule(
        @Request() { user }: JwtValidationOutput,
        @Body('schedule') serializedSchedule: SerializedSchedule,
    ): Promise<void> {
        if (user.userId !== serializedSchedule.ownerId) {
            throw new UnauthorizedException();
        }

        const schedule = deserializeSchedule(serializedSchedule);
        return await this.schedulesService.createSchedule(schedule);
    }

    @UseGuards(JwtAuthGuard)
    @Patch()

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		34

```

async updateSchedule(
  @Request() { user }: JwtValidationOutput,
  @Body('schedule') serializedSchedule: SerializedSchedule,
): Promise<void> {
  if (user.userId !== serializedSchedule.ownerId) {
    throw new UnauthorizedException();
  }

  const schedule = deserializeSchedule(serializedSchedule);
  return await this.schedulesService.updateSchedule(schedule);
}

@UseGuards(JwtAuthGuard)
@Delete('/:id')
async deleteSchedule(
  @Request() { user }: JwtValidationOutput,
  @Param('id') id: string,
): Promise<void> {
  const schedule = await this.schedulesService.findById(id);
  if (schedule.ownerId !== user.userId) {
    throw new UnauthorizedException();
  }

  return await this.schedulesService.deleteById(id);
}

@UseGuards(JwtAuthGuard)
@Get(MySchedulesRoute)
async getMySchedules(
  @Request() { user }: JwtValidationOutput,
): Promise<ScheduleStats[]> {
  const schedules = await this.schedulesService.getSchedulesByUser(
    user.userId,
  );

  return schedules;
}

@UseGuards(JwtAuthGuard)
@Get('/:id')
async getScheduleById(
  @Request() { user }: JwtValidationOutput,
  @Param('id') id: string,
): Promise<SerializedSchedule> {
  const schedule = await this.schedulesService.findById(id);

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		35

```

        if (schedule.ownerId !== user.userId) {
            throw new UnauthorizedException();
        }

        return serializeSchedule(schedule);
    }

    @UseGuards(JwtAuthGuard)
    @Post(SingleDeviceScheduleOptimizeRoute)
    async optimizeSingleDeviceSchedule(
        @Request() { user }: JwtValidationOutput,
        @Body('schedule') serializedSchedule: SerializedSchedule,
        @Body('useRmax') useRmax: boolean,
    ): Promise<SerializedSchedule> {
        if (user.userId !== serializedSchedule.ownerId) {
            throw new UnauthorizedException();
        }

        const schedule = deserializeSchedule(serializedSchedule);

        const res = await this.schedulesService.optimizeSingleDeviceSchedule(
            schedule,
            useRmax,
        );

        return serializeSchedule(res);
    }

    @UseGuards(JwtAuthGuard)
    @Post(MultiDeviceScheduleOptimizeRoute)
    async optimizeMultiDeviceSchedule(
        @Request() { user }: JwtValidationOutput,
        @Body('schedule') serializedSchedule: SerializedSchedule,
    ): Promise<SerializedSchedule> {
        if (user.userId !== serializedSchedule.ownerId) {
            throw new UnauthorizedException();
        }

        const schedule = deserializeSchedule(serializedSchedule);

        const res = await this.schedulesService.optimizeMultiDeviceSchedule(
            schedule,
        );

        return serializeSchedule(res);
    }
}

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		36

Файл client/src/helpers/schedule.ts

```
export const serializeSchedule = (schedule: Schedule): SerializedSchedule =>
{
  const operations: Operation[] = [];
  schedule.tasks.forEach((task) => {
    operations.push(...task.operations);
    if (task.operations.length !== 0) {
      task.operations[task.operations.length - 1].deadline = task.deadline;
    }
  });
  const serializedSchedule: SerializedSchedule = {
    ...schedule,
    operations,
    layers: schedule.layers.map((layer) => ({
      ...layer,
      devices: layer.devices.map(({ operations, ...device }) => ({
        ...device,
        operationIds: operations.map(({ id }) => id),
      })),
    })),
    tasks: schedule.tasks.map(({ operations, ...task }) => ({
      ...task,
      operationIds: operations.map(({ id }) => id),
    })),
  };

  return serializedSchedule;
};

export const deserializeSchedule = ({
  operations,
  ...serializedSchedule
}: SerializedSchedule): Schedule => {
  const schedule: Schedule = {
    ...serializedSchedule,
    tasks: serializedSchedule.tasks.map(({ operationIds, ...task }) => ({
      ...task,
      operations: operationIds.map(
        (id) => operations.find((operation) => operation.id === id) as
        Operation
      )
    })
  };
};
```

```

    ),
  })),
  layers: serializedSchedule.layers.map((layer) => ({
    ...layer,
    devices: layer.devices.map(({ operationIds, ...device }) => ({
      ...device,
      operations: operationIds.map(
        (id) =>
          operations.find((operation) => operation.id === id) as Operation
      ),
    })),
  })),
};

return schedule;
};

export interface OperationStats {
  lag: number | null;
  advance: number | null;
}

export const getOperationStats = (operation: Operation): OperationStats => {
  if (operation.start === null || operation.deadline === null) {
    return { lag: null, advance: null };
  }

  const end = operation.start + operation.duration;
  const lag = Math.max(0, end - operation.deadline);
  const advance = Math.max(0, operation.deadline - end);
  return { lag, advance };
};

```

Файл client/src/hooks/useForm.ts

```

interface Props<T> {
  initData: T;
  getErrors?: (data: T) => Partial<Record<keyof T, string | null>>;
  dynamic?: boolean;
}

export default function useForm<T>({
  initData,
  getErrors,
  dynamic = true,
}: Props<T>) {

```

```

const [newData, setNewData] = useState({ ...initData });
const [errors, setErrors] = useState<Partial<Record<keyof T, string>>>({});

useEffect(() => {
  if (dynamic) {
    setNewData({ ...initData });
  }
}, [initData]);

const onFieldChange = (field: keyof T, value: T[keyof T] | null) => {
  setErrors(
    ({ [field]: _, ...oldErrors }) =>
      ({ ...oldErrors } as Partial<Record<keyof T, string>>)
  );
  setNewData((oldData) => ({ ...oldData, [field]: value }));
};

const validate = () => {
  const newErrors = getErrors?.(newData) ?? {};

  const filteredErrors = Object.fromEntries(
    Object.entries(newErrors).filter(([_, value]) => value)
  ) as Partial<Record<keyof T, string>>;

  setErrors(filteredErrors);

  return {
    errors: filteredErrors,
    valid: Object.keys(filteredErrors).length === 0,
  };
};

const setError = (field: keyof T, error: string) => {
  setErrors(
    (oldErrors) =>
      ({ ...oldErrors, [field]: error } as Partial<Record<keyof T,
string>>)
  );
};

return { data: newData, errors, onFieldChange, validate, setError };
}

```

Файл client/src/interfaces/schedule.ts

```
export interface Task {
  id: string;
  name: string;
  scheduleId: string;
  deadline: number | null;
  operations: Operation[];
}

export interface Operation {
  id: string;
  name: string;
  duration: number;
  deadline: number | null;
  deviceId: string | null;
  taskId: string;
  start: number | null;
  end: number | null;
}

export interface Device {
  id: string;
  name: string;
  operations: Operation[];
  layerId: string;
  start: number | null;
  end: number | null;
}

export interface Layer {
  id: string;
  name: string;
  scheduleId: string;
  devices: Device[];
}

export enum ScheduleType {
  SingleDevice = 'singleDevice',
  MultiDevice = 'multiDevice',
}

export const ScheduleTypeLabel: Record<ScheduleType, string> = {
```

```

    [ScheduleType.SingleDevice]: 'Single Device',
    [ScheduleType.MultiDevice]: 'Multi Device',
  };

export interface Schedule {
  id: string;
  name: string;
  ownerId: string;
  type: ScheduleType;
  tasks: Task[];
  layers: Layer[];
}

export enum SingleDeviceScheduleOptimizationType {
  Complex = 'complex',
  Fast = 'fast',
}

export interface SerializedSchedule {
  id: string;
  name: string;
  ownerId: string;
  type: ScheduleType;
  operations: Operation[];
  tasks: (Omit<Task, 'operations'> & { operationIds: string[] })[];
  layers: (Omit<Layer, 'devices'> & {
    devices: (Omit<Device, 'operations'> & {
      operationIds: string[];
    })[];
  })[];
}

export interface ScheduleStats {
  id: string;
  name: string;
  type: ScheduleType;
  counts: {
    operations: number;
    tasks: number;
    layers: number;
    devices: number;
  };
}

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		41

```

export const getDefaultOperation = (taskId: string): Operation => ({
  id: uuid(),
  name: 'Operation',
  duration: 1,
  taskId,
  deadline: null,
  deviceId: null,
  start: null,
  end: null,
});

export const getDefaultTask = (scheduleId: string): Task => ({
  id: uuid(),
  name: 'Task',
  scheduleId,
  deadline: 1,
  operations: [],
});

export const getDefaultSingleOperationTask = (scheduleId: string): Task => ({
  ...getDefaultTask(scheduleId),
  name: 'Operations',
  deadline: null,
});

export const getDefaultLayer = (scheduleId: string): Layer => ({
  id: uuid(),
  name: 'Layer',
  scheduleId,
  devices: [],
});

export const getDefaultDevice = (layerId: string): Device => ({
  id: uuid(),
  name: 'Device',
  layerId,
  operations: [],
  start: null,
  end: null,
});

export const NOT_USED_OPERATIONS = 'not-used';

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		42

```

export const getNotUsedOperationsDevice = (
  layerId: string,
  operations: Operation[]
): Device => ({
  id: NOT_USED_OPERATIONS,
  name: 'Not used operations',
  layerId,
  operations,
  start: null,
  end: null,
});

export const getDefaultSchedule = (
  ownerId: string,
  type: ScheduleType = ScheduleType.SingleDevice
): Schedule => {
  const scheduleId = uuid();
  return {
    id: scheduleId,
    name: 'Schedule',
    ownerId,
    tasks: [getDefaultTask(scheduleId)],
    layers: [getDefaultLayer(scheduleId)],
    type,
  };
};

const validateSingleDeviceSchedule = (schedule: Schedule): string | null => {
  if (schedule.layers.length === 0) {
    return 'Schedule should contain at least one layer!';
  }
  if (schedule.tasks.length === 0) {
    return 'Schedule should contain at least one task!';
  }

  const layersNum = schedule.tasks[0].operations.length;
  if (layersNum !== schedule.layers.length) {
    return 'Number of operations in tasks does not match with number of
layers!';
  }
  for (const task of schedule.tasks) {
    if (task.operations.length !== layersNum) {

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		43

```

        return 'All tasks should have the same number of operations (layers)!';
    }
    for (const operation of task.operations) {
        if (!operation.deviceId) {
            return 'All operations should be allocated to devices!';
        }
    }
}

for (const layer of schedule.layers) {
    for (const device of layer.devices) {
        if (device.operations.length === 0) {
            return 'Each device should have at least one associated operation!';
        }
    }
}

return null;
};

const validateMultiDeviceSchedule = (schedule: Schedule): string | null => {
    if (schedule.tasks.length === 0) {
        return 'Schedule should contain at least one operation!';
    }

    return null;
};

export const validateSchedule = (schedule: Schedule): string | null => {
    if (schedule.type === ScheduleType.SingleDevice) {
        return validateSingleDeviceSchedule(schedule);
    } else {
        return validateMultiDeviceSchedule(schedule);
    }
};

```

Файл client/src/pages/LoginPage.tsx

```
const LoginForm = styled.form`
  display: flex;
  flex-direction: column;
  min-width: 540px;
`;
```

					КПІ.ІП-9213.045490.03.12	Арк.
Вмін.	Арк.	№ докум.	Підп.	Дата.		44

```

const LoginDiv = styled.div`
  margin: 0 auto;
  padding: 1rem 5rem;
  display: flex;
  flex-direction: column;
  flex-wrap: wrap;
`;

const ButtonBox = styled.div`
  min-width: 375px;
  max-width: 2560 px;
  height: auto;
  width: 100%;
  display: flex;
  justify-content: space-between;
  margin: 0.5rem 0;

  & > * {
    width: 49%;
  }
`;

const Title = styled.h1`
  text-align: center;
  font-size: calc(3vw + 2rem);
  margin: 1rem;
  color: ${Colors.primary};
`;

const RegisterButton = styled(Button)`
  min-height: 43px;
`;

const LoginPage = (): JSX.Element => {
  const [email, setEmail] = useState<string>('');
  const [password, setPassword] = useState<string>('');
  const [showError, setShowError] = useState<boolean>(false);

  const navigate = useNavigate();

  useEffect(() => {
    const { user } = store;
    if (user) {

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		45

```

        navigate('/users/' + user.userId);
    }
}, [navigate, store?.user]);
const handleGoogleClick = async (res: CredentialResponse): Promise<void> =>
{
    try {
        const { accessToken } = await responseGoogle(res);
        await store.setAccessToken(accessToken);
        navigate('/chat');
    } catch (error) {
        setShowError(true);
    }
};

const setToken = async (): Promise<void> => {
    try {
        const { accessToken } = await login(email, password);
        await store.setAccessToken(accessToken);
        navigate('/chat');
    } catch (error) {
        setShowError(true);
    }
};

const handleSubmit = async (
    event: React.FormEvent<HTMLFormElement>
): Promise<void> => {
    event.preventDefault();
    await setToken();
};

return (
    <LoginDiv>
        <Title>Login</Title>
        <LoginForm onSubmit={handleSubmit}>
            <label>Email</label>
            <Input
                type="email"
                value={email}
                onChange={(ev): void => setEmail(ev.target.value)}
            />

            <label>Password</label>

```

```

    <Input
      type="password"
      value={password}
      onChange={(ev): void => setPassword(ev.target.value)}
    />
    {showError ? (
      <ErrorValidation>Email or passord is wrong!</ErrorValidation>
    ) : null}
    <GoogleOAuthProvider
      clientId={process.env.REACT_APP_GOOGLE_CLIENT_ID as string}
    >
      <ButtonBox>
        <Button>Sign in</Button>

        <GoogleLogin
          onSuccess={(res): Promise<void> => handleGoogleClick(res)}
          onError={() => console.log('ERROR')}
        />
      </ButtonBox>
    </GoogleOAuthProvider>
  </LoginForm>
  <RegisterButton onClick={() : void => navigate('/registration')}>
    Sign up
  </RegisterButton>
</LoginDiv>
);
};

export default observer(LoginPage);

```

Файл client/src/pages/ProfilePage.tsx

```

const Title = styled.h1`
  text-align: center;
  font-size: 2rem;
  margin: 1rem;
  color: ${Colors.primary};
`;

const Form = styled.form`
  display: flex;

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		47

```

    flex-direction: column;

    & > :last-child {
      margin-top: 4px;
    }
  `;

const ProfileBlockView = styled.div`
  display: flex;
  flex-grow: 1;
  flex-wrap: wrap;
  flex-direction: column;
  padding: 0 1rem;
`;

const InfoBlock = styled.div`
  display: flex;
  flex-wrap: wrap;
  padding: 0.5rem;
  justify-content: space-evenly;
`;

const ProfileDetails = styled.div`
  display: flex;
  flex-direction: column;
`;

const DetailInfo = styled.h3`
  text-align: center;
  display: flex;
  margin: 0.25rem;
  color: ${Colors.secondary};
`;

const ButtonBlock = styled.div`
  display: flex;
  justify-content: center;
  & * {
    margin: 0 0.25rem;
  }
`;

const Label = styled.label`

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		48

```

    color: ${Colors.primary};
    margin: 0.25rem;
`;

export default observer(function ProfilePage(): JSX.Element {
    const [user, setUser] = useState<User>(store.user);
    const [showChangePasswordModal, setShowChangePasswordModal] =
    useState(false);
    const [showModal, setShowModal] = useState(false);
    const navigate = useNavigate();
    const params = useParams<{ id: string }>();
    const { id } = params;

    const getProfile = async (): Promise<void> => {
        try {
            const profileUser = await getUserProfile(store.accessToken, id as
            string);
            setUser(profileUser);
        } catch {
            navigate('/');
        }
    };

    useEffect(() => {
        if (store.initialized) {
            if (!store.user) {
                return navigate('/login');
            }
        }
    }, [store.initialized]);

    useEffect(() => {
        if (store.initialized) {
            getProfile();
        }
    }, [store.initialized, params]);

    const {
        data: newUserInfo,
        onFieldChange: onUserInfoFieldChange,
        validate: validateUserInfo,
        errors: newUserInfoErrors,
    } = useForm({

```

```

    initData: user,
    getErrors: (newUser) => ({
      firstName: !newUser.firstName ? EMPTY_ERROR : null,
      lastName: !newUser.lastName ? EMPTY_ERROR : null,
    }),
  });

const changeProfile = async (
  event: React.FormEvent<HTMLFormElement>
): Promise<void> => {
  event.preventDefault();

  const { valid } = validateUserInfo();
  if (!valid) {
    return;
  }

  const { firstName, lastName } = newUserInfo;
  await patchMyProfile(store.accessToken, {
    firstName,
    lastName,
  });
  store.setUser(newUserInfo);
  setUser(newUserInfo);
  setShowModal(false);
};

const {
  data: { oldPassword, newPassword, repeatNewPassword },
  onFieldChange: onPasswordInfoFieldChange,
  validate: validatePasswordInfo,
  errors: newPasswordInfoErrors,
  setError,
} = useForm({
  initData: { oldPassword: '', newPassword: '', repeatNewPassword: '' },
  getErrors: (newPasswords) => ({
    oldPassword: !newPasswords.oldPassword ? EMPTY_ERROR : null,
    newPassword: !newPasswords.newPassword ? EMPTY_ERROR : null,
    repeatNewPassword: !newPasswords.repeatNewPassword
      ? EMPTY_ERROR
      : newPasswords.newPassword !== newPasswords.repeatNewPassword
      ? 'Passwords does not match'
      : null,
  })
});

```

```

   )),
    dynamic: false,
  });

const changePassword = async (event: React.FormEvent): Promise<void> => {
  event.preventDefault();
  const { valid } = validatePasswordInfo();
  if (!valid) {
    return;
  }

  try {
    const { accessToken } = await changeMyPassword(
      store.accessToken,
      oldPassword,
      newPassword
    );
    store.setAccessToken(accessToken);
    setShowChangePasswordModal(false);
  } catch (error) {
    //@ts-ignore
    if (error.message.toLowerCase().includes('password')) {
      setError('newPassword', PASSWORD_ERROR);
    }
  }
};

if (!user) return <div></div>;

const modal = (
  <Modal open={showModal} onClose={() : void => setShowModal(false)}>
    <Form onSubmit={changeProfile}>
      <Label>Enter your First name</Label>
      <Input
        value={newUserInfo?.firstName}
        onChange={({ target }) : void =>
          onUserInfoFieldChange('firstName', target.value)
        }
        type="text"
        error={newUserInfoErrors.firstName}
      />
      <Label>Enter your Last name</Label>
      <Input

```

```

        value={newUserInfo?.lastName}
        onChange={({ target }): void =>
            onUserInfoFieldChange('lastName', target.value)
        }
        type="text"
        error={newUserInfoErrors.lastName}
    />
    <Button>Save changes</Button>
</Form>
</Modal>
);

const changePasswordModal = (
    <Modal
        open={showChangePasswordModal}
        onClose={() : void => setShowChangePasswordModal(false)}
    >
        <Form onSubmit={changePassword}>
            <Label>Confirm your password</Label>
            <Input
                value={oldPassword}
                onChange={({ target }): void =>
                    onPasswordInfoFieldChange('oldPassword', target.value)
                }
                type="password"
                error={newPasswordInfoErrors.oldPassword}
            />

            <Label>Enter new password</Label>
            <Input
                value={newPassword}
                onChange={({ target }): void =>
                    onPasswordInfoFieldChange('newPassword', target.value)
                }
                type="password"
                error={newPasswordInfoErrors.newPassword}
            />

            <Label>Repeat new password</Label>
            <Input
                value={repeatNewPassword}
                onChange={({ target }): void =>
                    onPasswordInfoFieldChange('repeatNewPassword', target.value)

```

```

        }
        type="password"
        error={newPasswordInfoErrors.repeatNewPassword}
      />
      <Button>Change password</Button>
    </Form>
  </Modal>
);

const changePasswordButton = user.authProviders.includes('local') ? (
  <Button onClick={() : void => setShowChangePasswordModal(true)}>
    Change password
  </Button>
) : null;

return (
  <ProfileBlockView>
    {modal}
    {changePasswordModal}
    <Title>Profile</Title>
    <InfoBlock>
      <ProfileDetails>
        <DetailInfo>
          {user?.lastName} {user?.firstName}
        </DetailInfo>
        <DetailInfo>{user?.email}</DetailInfo>
      </ProfileDetails>
    </InfoBlock>
    <ButtonBlock>
      {user?.userId === store?.user?.userId ? (
        <>
          <Button
            onClick={() : void => {
              setShowModal(true);
            }}
          >
            Change profile
          </Button>
          {changePasswordButton}
        </>
      ) : null}
    </ButtonBlock>
  </ProfileBlockView>

```

```
);
});
```

Файл client/src/pages/RegistrationPage.tsx

```
const RegistrationDiv = styled.div`
  margin: 0 auto;
  padding: 1rem 5rem;
  display: flex;
  flex-direction: column;
  flex-wrap: wrap;
`;

const RegistrationForm = styled.form`
  display: flex;
  flex-direction: column;
`;

const Title = styled.h1`
  text-align: center;
  font-size: calc(3vw + 2rem);
  margin: 1rem;
  color: ${Colors.primary};
`;

const RegisterActions = styled.div`
  display: flex;
  justify-content: center;
  gap: 0.5rem;
  margin-top: 0.5rem;
`;

export default observer(function RegistrationPage(): JSX.Element {
  const [serverError, setServerError] = useState<string>();
  const {
    data: { firstName, lastName, email, password },
    validate,
    onFieldChange,
    errors,
    setError,
  } = useForm({
    initData: {
      firstName: '',
      lastName: '',
```

```

        email: '',
        password: '',
    },
    dynamic: false,
    getErrors: (data) => ({
        firstName: !data.firstName ? EMPTY_ERROR : null,
        lastName: !data.lastName ? EMPTY_ERROR : null,
        email: !data.email ? EMPTY_ERROR : null,
        password: !data.password ? EMPTY_ERROR : null,
    }),
    });
const navigate = useNavigate();

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

    setServerError(undefined);
    const { valid } = validate();
    if (!valid) {
        return;
    }

    try {
        const { accessToken } = await register(
            firstName,
            lastName,
            email,
            password
        );
        store.setAccessToken(accessToken);
        navigate('/extra-page');
    } catch (error) {
        //@ts-ignore
        const message: string = error.message;
        if (message.includes('password')) {
            setError('password', PASSWORD_ERROR);
        } else {
            setServerError(message);
        }
    }
};

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		55

```

return (
  <RegistrationDiv>
    <Title>User Registration</Title>
    <RegistrationForm onSubmit={handleSubmit}>
      <label>First Name</label>
      <Input
        value={firstName}
        onChange={(ev): void => onFieldChange('firstName',
ev.target.value)}
        type="text"
        error={errors.firstName}
      />
      <label>Last Name</label>
      <Input
        value={lastName}
        onChange={(ev): void => onFieldChange('lastName', ev.target.value)}
        type="text"
        error={errors.lastName}
      />
      <label>Email</label>
      <Input
        value={email}
        onChange={(ev): void => onFieldChange('email', ev.target.value)}
        type="email"
        error={errors.email}
      />
      <label>Password</label>
      <Input
        value={password}
        onChange={(ev): void => onFieldChange('password', ev.target.value)}
        type="password"
        error={errors.password}
      />
      {serverError ? (
        <ErrorValidation className="small">{serverError}</ErrorValidation>
      ) : null}
      <RegisterActions>
        <Button
          outlined
          className="register"
          onClick={() : void => navigate('/')}>
          >

```

```

        Back to Login page
      </Button>
      <Button className="register" type="submit">
        Register now
      </Button>
    </RegisterActions>
  </RegistrationForm>
</RegistrationDiv>
);
});

```

Файл client/src/pages/SchedulePage.tsx

```

const Label = styled.span`
  ${regularFont('1.25rem')}
  white-space: nowrap;
`;

const StyledSpinner = styled(Spinner)`
  &&&& {
    flex-grow: unset;
    margin: auto;
  }
`;

const ModalBody = styled.div`
  display: flex;
  flex-direction: column;
  align-items: center;
`;

const ModalActions = styled.div`
  margin-top: 1rem;
  display: flex;
  gap: 0.5rem;
`;

const CreateForm = styled.div`
  display: flex;
  flex-direction: column;
  gap: 1rem;
  align-items: stretch;

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		57

```

`;

const TypeButtonsGroup = styled(ButtonGroup) `
  margin: 0.25rem 0;
  align-self: stretch;
`;

const TypeButton = styled(Button)<{ outlined?: boolean }>`
  ${({ outlined }) => (!outlined ? `background: ${Colors.secondary}` :
'white')})
`;

const NameBlock = styled.div`
  display: flex;
  justify-content: center;
  align-items: center;
  margin-top: 1rem;
`;

const NameLabel = styled(Label)`
  ${boldFont('1.5rem')}
  line-height: 1.5rem;
`;

const SaveNameButton = styled(Button)`
  min-height: 43px;
  max-height: 43px;
  margin: 0.25rem;
  align-self: flex-start;
`;

const NameEditButton = styled(EditButton)`
  margin-left: 1rem;
`;

const StyledDeleteButton = styled>DeleteButton)`
  position: absolute;
  right: 0.5rem;
`;

interface Props {
  formType: FormType;
}

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		58

```

export default observer(function SchedulePage({ formType }: Props) {
  const [fetched, setFetched] = useState(false);
  const [showEditModal, setShowEditModal] = useState(false);
  const [showDeleteModal, setShowDeleteModal] = useState(false);
  const [editName, setEditName] = useState(false);

  const location = useLocation();
  const navigate = useNavigate();

  const [showModal, setShowModal] = useState(formType === FormType.New);

  const scheduleId = location.pathname.split('/').slice(-1)[0];
  const schedule = store.schedule;

  const {
    data: { name },
    onFieldChange: onNameFieldChange,
    errors: nameErrors,
    validate,
  } = useForm({
    initData: schedule,
    getErrors: (data) => ({
      name: !data.name ? EMPTY_ERROR : '',
    }),
  });

  const {
    data: newSchedule,
    onFieldChange,
    errors,
  } = useForm({
    initData: {
      name: 'Schedule',
      type: ScheduleType.SingleDevice,
    },
    getErrors: (s) => ({
      name: !s.name ? EMPTY_ERROR : null,
    }),
    dynamic: false,
  });

  const fetchSchedule = async () => {

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		59

```

    await store.getSchedule(scheduleId);
    setFetched(true);
  };

  useEffect(() => {
    if (!fetched && store.accessToken) {
      if (formType === FormType.Edit) {
        fetchSchedule();
      } else {
        store.resetSchedule();
      }
    }
  }, [store.accessToken]);

  const handleCreateCancel = () => {
    setShowModal(false);
    navigate('/schedules');
  };

  const handleCreateSubmit = () => {
    store.createNewSchedule(newSchedule.name, newSchedule.type);
    setShowModal(false);
    setFetched(true);
  };

  const mapPages: Record<ScheduleType, JSX.Element> = {
    [ScheduleType.SingleDevice]: (
      <SingleDeviceScheduleForm
        formType={formType}
        onEdit={() => setShowEditModal(true)}
      />
    ),
    [ScheduleType.MultiDevice]: (
      <MultiDeviceScheduleForm
        formType={formType}
        onEdit={() => setShowEditModal(true)}
      />
    ),
  };

  useEffect(() => {
    if (store.initialized) {
      if (!store.user) {

```

```

        return navigate('/login');
    }
}
}, [store.initialized]);

const handleMakeScheduleEditable = () => {
    store.setCanEditSchedule(true);
    setShowEditModal(false);
};

const handleDeleteClick = async () => {
    await store.deleteSchedule();
    setShowDeleteModal(false);
    navigate('/schedules');
};

const handleNameChange = () => {
    const { valid } = validate();
    if (!valid) {
        return;
    }
    store.schedule.name = name;
    setEditName(false);
};

return (
    <>
        <Modal open={showModal} onClose={handleCreateCancel}>
            <CreateForm>
                <NameLabel>Create Schedule</NameLabel>
                <InputContainer label="Name">
                    <Input
                        value={newSchedule.name}
                        onChange={({ target }) => onFieldChange('name', target.value)}
                        error={errors.name}
                    />
                </InputContainer>
                <InputContainer label="Schedule Type">
                    <TypeButtonsGroup>
                        <TypeButton
                            outlined={newSchedule.type !== ScheduleType.SingleDevice}
                            onClick={() => onFieldChange('type',
ScheduleType.SingleDevice)}

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		61

```

        >
        Single Device
    </TypeButton>
    <TypeButton
        outlined={newSchedule.type !== ScheduleType.MultiDevice}
        onClick={() => onFieldChange('type',
ScheduleType.MultiDevice)}
    >
        Multi Device
    </TypeButton>
</TypeButtonsGroup>
</InputContainer>
<ButtonGroup>
    <Button outlined onClick={handleCreateCancel}>
        Cancel
    </Button>
    <Button onClick={handleCreateSubmit}>Create & Continue</Button>
</ButtonGroup>
</CreateForm>
</Modal>
<Modal open={showEditModal} onClose={() => setShowEditModal(false)}>
    <ModalBody>
        <Label>Are you sure you want to edit schedule?</Label>
        <Label>
            All time parameters like start and end times will be removed!
        </Label>
        <ModalActions>
            <Button outlined onClick={() => setShowEditModal(false)}>
                Cancel
            </Button>
            <Button onClick={handleMakeScheduleEditable}>Submit</Button>
        </ModalActions>
    </ModalBody>
</Modal>
<Modal open={showDeleteModal} onClose={() =>
setShowDeleteModal(false)}>
    <ModalBody>
        <Label>Are you sure you want to delete schedule?</Label>
        <ModalActions>
            <Button outlined onClick={() => setShowDeleteModal(false)}>
                Cancel
            </Button>
            <Button onClick={handleDeleteClick}>Submit</Button>
        </ModalActions>
    </ModalBody>
</Modal>

```

```

        </ModalActions>
      </ModalBody>
    </Modal>
    {!store.loading && (
      <NameBlock>
        {editName ? (
          <>
            <Input
              value={name}
              onChange={({ target }) =>
                onNameFieldChange('name', target.value)
              }
              error={nameErrors.name}
            />
            <SaveNameButton
              onClick={handleNameChange}>Save</SaveNameButton>
            </>
          ) : (
            <>
              <NameLabel>{schedule?.name}</NameLabel>
              <NameEditButton onClick={() => setEditName(true)} />
            </>
          )}
          <StyledDeleteButton onClick={() => setShowDeleteModal(true)} />
        </NameBlock>
      )}
      {store.loading && <StyledSpinner />}
      {store.schedule && mapPages[store.schedule?.type]}
    </>
  );
});

```

Файл client/src/pages/SchedulesPage.tsx

```

const SchedulesPageBlock = styled.div`
  display: flex;
  flex-direction: column;
`;

const Title = styled.h1`
  color: ${Colors.primary};

```

					КПІ.ІП-9213.045490.03.12	Арк.
Вмін.	Арк.	№ докум.	Підп.	Дата.		63

```

    text-align: center;
`;

const SchedulesList = styled.div`
  display: flex;
  gap: 0.5rem;
  padding: 1rem;
  flex-wrap: wrap;
`;

const ScheduleCard = styled.div`
  display: flex;
  padding: 1rem;
  border-radius: 0.5rem;
  border: 1px ${Colors.primary} solid;
  color: ${Colors.primary};
  transition: all 0.2s ease;
  cursor: pointer;
  &:hover {
    background-color: ${Colors.primary};
    color: ${Colors.light};
  }
`;

const ScheduleInfo = styled.div`
  display: flex;
  flex-direction: column;
`;

const ScheduleImage = styled.img`
  width: 6rem;
  height: 6rem;
  margin: auto 0.5rem;
`;

const ScheduleName = styled.span`
  ${boldFont('26px')}
  line-height: 26px;
  margin-bottom: 0.5rem;
`;

const AddCard = styled(ScheduleCard)`
  min-height: 125px;

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		64

```

    &:hover svg path {
      transition: stroke 0.3s ease;
      stroke: ${Colors.light};
    }
  `;

const AddIcon = styled(Icon) `
  align-self: center;
  margin-right: 4px;
`;

export default observer(function SchedulesPage() {
  const { data: schedules, isLoading } = useQuery(
    'schedules',
    () => getMySchedules(store.accessToken),
    {
      enabled: !!store.accessToken,
    }
  );

  const navigate = useNavigate();

  useEffect(() => {
    if (store.initialized) {
      if (!store.user) {
        return navigate('/login');
      }
    }
  }, [store.initialized]);

  return (
    <SchedulesPageBlock>
      <Title>My Schedules</Title>
      <SchedulesList>
        {isLoading && <Spinner />}

        {schedules?.map((schedule) => (
          <ScheduleCard
            key={`schedule-${schedule.id}`}
            onClick={() => navigate(`/schedules/${schedule.id}`)}
          >
            <ScheduleInfo>
              <ScheduleName>{schedule.name}</ScheduleName>
              <span>Type: {ScheduleTypeLabel[schedule.type]}</span>
            </ScheduleInfo>
          </ScheduleCard>
        ))}
      </SchedulesList>
    </SchedulesPageBlock>
  );
});

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		65

```

        {schedule.type === ScheduleType.SingleDevice && (
          <span>Layers: {schedule.counts.layers}</span>
        )}
        <span>Devices: {schedule.counts.devices}</span>
        {schedule.type === ScheduleType.SingleDevice && (
          <span>Tasks: {schedule.counts.tasks}</span>
        )}
        <span>
          {`Operations: ${
            schedule.type === ScheduleType.SingleDevice
              ? schedule.counts.operations
              : schedule.counts.tasks
          }`}
        </span>
      </ScheduleInfo>
      <ScheduleImage src="/schedule.png" />
    </ScheduleCard>
  )}}
  <AddCard onClick={() => navigate(`/schedules/new`)}>
    <AddIcon
      width={75}
      height={75}
      icon={IconType.Add}
      color={Colors.secondary}
    />
  </AddCard>
</SchedulesList>
</SchedulesPageBlock>
);
});

```

Файл client/src/components/schedule/CommonLayout.tsx

```

export const Actions = styled.div`
  display: flex;
  flex-direction: column;
  margin-left: auto;
  justify-content: center;

```

```

    opacity: 0;
    transition: opacity 0.3s ease;
    margin: auto 0 auto auto;
    gap: 0.25rem;
`;

const StyledDeleteButton = styled(Button) `
  &,
  &:hover {
    background: ${Colors.red};
  }
`;

export const DeleteButton = (
  props: ButtonHTMLAttributes<HTMLButtonElement>
) => {
  return (
    <StyledDeleteButton {...props}>
      <Icon icon={IconType.Remove} />
    </StyledDeleteButton>
  );
};

const EditOperation = styled(Button) `
  margin-left: auto;
  align-self: center;
`;

export const EditButton = (props: ButtonHTMLAttributes<HTMLButtonElement>) =>
{
  return (
    <EditOperation {...props}>
      <Icon icon={IconType.Edit} />
    </EditOperation>
  );
};

export const EditForm = styled.form `
  display: flex;
  align-items: center;
  flex-direction: column;
  gap: 0.25rem;
`;

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		67

```

export const FormTitle = styled.h1`
  ${boldFont('2rem')}
  color: ${Colors.primary};
  margin: 0.5rem;
`;

export const CardInfoBlock = styled.div`
  display: flex;
  flex-direction: column;
  color: ${Colors.light};
`;

export const CardNameLabel = styled.span`
  ${boldFont('1.5rem')}
  line-height: 1.5rem;
  margin-bottom: 0.25rem;
`;

export const CardLabel = styled.span<{ bold?: boolean }>`
  ${({ bold }) => (!bold ? regularFont('1rem') : boldFont('1rem'))}
  white-space: nowrap;
`;

const InputContainerLabel = styled.span`
  ${regularFont('18px')}
  line-height: 18px;
  color: ${Colors.primary};
`;

const StyledInputContainer = styled.div`
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  gap: 0.125rem;
`;

interface InputContainerProps {
  children?: React.ReactNode;
  label: string | React.ReactNode;
}

export const InputContainer = ({ children, label }: InputContainerProps) => {

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		68

```

    return (
      <StyledInputContainer>
        <InputContainerLabel>{label}</InputContainerLabel>
        {children}
      </StyledInputContainer>
    );
  };
};

```

Файл client/src/components/schedule/DeviceDnd.tsx

```

const DeviceBlock = styled.div<{ danger: boolean }>`
  display: flex;
  border: 1px ${Colors.light} solid;
  border-radius: 0.25rem;
  padding: 1rem;
  gap: 1rem;
  align-self: stretch;
  transition: all 0.3s ease;
  background-color: ${({ danger }) => (danger ? Colors.red : Colors.green)};

  min-height: 70px;
  position: relative;

  &:hover > ${Actions} {
    opacity: 1;
  }
`;

const DndBlock = styled.div<{ dragged: boolean }>`
  display: flex;
  flex-grow: 1;
  min-width: 250px;
`;

interface Props {
  device: Device;
  layerId: string;
  className?: string;
  moveDisabled?: boolean;
}

export default observer(function DeviceDnd({

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		69

```

    device,
    layerId,
    className,
  }: Props) {
    const [showModal, setShowModal] = useState(false);

    const {
      data: newDevice,
      errors,
      validate,
      onChange,
    } = useForm({
      initData: device,
      getErrors: (newDevice) => ({
        name: !newDevice.name ? EMPTY_ERROR : null,
        start: newDevice.start === null ? EMPTY_ERROR : null,
      }),
    });

    const handleSubmit = (e: FormEvent) => {
      e.preventDefault();
      const { valid } = validate();
      if (!valid) {
        return;
      }
      store.updateDevice(newDevice);
      setShowModal(false);
    };

    const totalStats = device.operations.reduce(
      (prev, operation) => {
        const stats = getOperationStats(operation);
        if (stats.lag === null || stats.advance === null) {
          return prev;
        }

        return {
          lag: (prev.lag ?? 0) + stats.lag,
          advance: (prev.advance ?? 0) + stats.advance,
        };
      },
      { lag: null, advance: null } as OperationStats
    );
  }

```

```

return (
  <>
    <Modal
      open={showModal}
      onClose={() => {
        setShowModal(false);
      }}
    >
      <EditForm onSubmit={handleSubmit}>
        <FormTitle>Edit Device</FormTitle>
        <InputContainer label="Name">
          <Input
            placeholder="Name"
            value={newDevice.name}
            onChange={({ target }) => onFieldChange('name', target.value)}
            error={errors?.name}
          />
        </InputContainer>
        {store.schedule?.type === ScheduleType.MultiDevice && (
          <InputContainer label="Start">
            <Input
              type="number"
              placeholder="Start"
              value={newDevice.start ?? ''}
              onChange={({ target }) =>
                onFieldChange('start', target.value ? +target.value : null)
              }
              error={errors?.start}
            />
          </InputContainer>
        )}
        <Button type="submit">Submit</Button>
      </EditForm>
    </Modal>
    <DeviceBlock
      className={className}
      danger={device.id === NOT_USED_OPERATIONS}
    >
      <CardInfoBlock>
        <CardNameLabel>{device.name}</CardNameLabel>
        {device.id !== NOT_USED_OPERATIONS && (
          <CardLabel>Operations: {device.operations.length}</CardLabel>
        )}
      </CardInfoBlock>
    </DeviceBlock>
  </>
)

```

```

    })
    {device.start !== null && (
      <CardLabel>Start: {device.start}</CardLabel>
    )}

    {totalStats.lag !== null && (
      <
        <CardLabel bold>Lag: {totalStats.lag}</CardLabel>
        <CardLabel bold>Advance: {totalStats.advance}</CardLabel>
      </>
    )}
  </CardInfoBlock>
  <Draggable
    type={device.layerId}
    isDropDisabled={device.id === NOT_USED_OPERATIONS}
    direction="horizontal"
    draggableId={getDeviceFullId(device)}
  >
    {(provided, snapshot) => (
      <DndBlock
        {...provided.draggableProps}
        ref={provided.innerRef}
        dragged={snapshot.isDraggingOver}
      >
        {device.operations.map((operation, index) => (
          <OperationCard
            key={operation.id}
            operation={operation}
            parent={device}
            index={index}
          />
        ))}
        {provided.placeholder}
      </DndBlock>
    )}
  </Draggable>
  {device.id !== NOT_USED_OPERATIONS && store.canEditSchedule && (
    <Actions>
      <EditButton onClick={() => setShowModal(true)} />
      <DeleteButton
        onClick={() => store.removeDevice(layerId, device.id)}
      />
    </Actions>
  )}

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		72

```

    })
  </DeviceBlock>
</>
);
});

```

Файл client/src/components/schedule/LayerDnd.tsx

```

const Header = styled.div`
  display: flex;
  align-self: stretch;
`;

const LayerBlock = styled.div`
  display: flex;
  flex-direction: column;
  padding: 1rem;
  border: 1px ${Colors.primary} solid;
  border-radius: 0.25rem;
  gap: 1rem;
  align-items: flex-start;
  min-width: 125px;
  transition: all 0.3s ease;
  background-color: ${Colors.primary};

  &:hover > ${Header} > ${Actions} {
    opacity: 1;
  }
`;

const AddDevice = styled(Button)`
  display: flex;
  gap: 0.5rem;
  min-width: 120px;
  border-color: ${Colors.green};
  &,
  &:hover {
    background: ${Colors.green};
  }
`;

const NotUsedOperationsDevice = styled(DeviceDnd)`

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		73

```

    margin-top: auto;
`;

interface Props {
    layer: Layer;
    notUsedOperations?: Operation[];
    moveDisabled?: boolean;
}

export default observer(function LayerDnd({
    layer,
    notUsedOperations = [],
    moveDisabled,
}: Props) {
    const [showModal, setShowModal] = useState(false);

    const {
        data: newLayer,
        errors,
        validate,
        onFieldChange,
    } = useForm({
        initData: layer,
        getErrors: (newLayer) => ({
            name: !newLayer.name ? EMPTY_ERROR : null,
        }),
    });

    const notUsedOperationsDevice = getNotUsedOperationsDevice(
        layer.id,
        notUsedOperations
    );

    const handleSubmit = (e: FormEvent) => {
        e.preventDefault();
        const { valid } = validate();
        if (!valid) {
            return;
        }
        store.updateLayer(newLayer);
        setShowModal(false);
    };

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		74

```

return (
  <>
    <Modal
      open={showModal}
      onClose={() => {
        setShowModal(false);
      }}
    >
      <EditForm onSubmit={handleSubmit}>
        <FormTitle>Edit Layer</FormTitle>
        <InputContainer label="Name">
          <Input
            placeholder="Name"
            value={newLayer.name}
            onChange={({ target }) => onFieldChange('name', target.value)}
            error={errors?.name}
          />
        </InputContainer>
        <Button type="submit">Submit</Button>
      </EditForm>
    </Modal>
    <LayerBlock>
      <Header>
        <CardInfoBlock>
          <CardNameLabel>{layer.name}</CardNameLabel>
          <CardLabel>Devices: {layer.devices.length}</CardLabel>
        </CardInfoBlock>
        {store.canEditSchedule && !moveDisabled && (
          <Actions>
            <EditButton onClick={() => setShowModal(true)} />
            <DeleteButton onClick={() => store.removeLayer(layer.id)} />
          </Actions>
        )}
      </Header>
      {layer.devices.map((device) => (
        <DeviceDnd
          moveDisabled
          layerId={layer.id}
          key={device.id}
          device={device}
        />
      ))}
    </LayerBlock>
  </>
)

```

```

        {store.canEditSchedule && (
          <AddDevice onClick={() => store.addNewDevice(layer.id)}>
            <Icon icon={IconType.Add} /> New Device
          </AddDevice>
        )}
        {notUsedOperations.length !== 0 && (
          <NotUsedOperationsDevice
            key={notUsedOperationsDevice.id}
            device={notUsedOperationsDevice}
            layerId={layer.id}
          />
        )}
      </LayerBlock>
    </>
  );
});

```

Файл client/src/components/schedule/LayersList.tsx

```

const Layers = styled.div`
  display: flex;
  padding: 2rem 1rem 3rem 1rem;
  gap: 1rem;
`;

const AddLayer = styled(Button)`
  align-self: flex-start;
  display: flex;
  gap: 0.5rem;
  min-width: 120px;
`;

interface Props {
  layers: Layer[];
  tasks: Task[];
}

export default observer(function LayersList({ tasks, layers }: Props) {
  const handleDragEnd: OnDragEndResponder = (result) => {
    if (!result.destination) {
      return;
    }

    const sourceIds = getPartsFromFullId(result.source.droppableId);

```



```

border: 1px solid ${Colors.light};
cursor: pointer;
width: 250px;
margin-right: 0.25rem;
transition: background 0.2s ease, color 0.2s ease;

&:hover ${Actions} {
  opacity: 1;
}
`;

interface Props {
  operation: Operation;
  parent: Device | Task | string;
  index: number;
  moveDisabled?: boolean;
}

export default observer(function OperationCard({
  operation,
  index,
  parent,
  moveDisabled,
}: Props) {
  const [showModal, setShowModal] = useState(false);
  const {
    data: newOperation,
    errors,
    validate,
    onFieldChange,
  } = useForm({
    initData: operation,
    getErrors: (newOperation) => ({
      name: !newOperation.name ? EMPTY_ERROR : null,
      duration: !newOperation.duration ? EMPTY_ERROR : null,
    }),
  });

  const handleSubmit = (e: FormEvent) => {
    e.preventDefault();
    const { valid } = validate();

    if (!valid) {

```

					КПІ.ІП-9213.045490.03.12	Арк.
						78
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        return;
    }
    const { name, duration } = newOperation;
    store.updateOperation({ ...operation, name, duration });
    setShowModal(false);
};

const stats = getOperationStats(operation);
return (
    <>
        <Modal
            open={showModal}
            onClose={() => {
                setShowModal(false);
            }}
        >
            <EditForm onSubmit={handleSubmit}>
                <FormTitle>Edit Operation</FormTitle>
                <InputContainer label="Name">
                    <Input
                        placeholder="Name"
                        value={newOperation.name}
                        onChange={({ target }) => onFieldChange('name', target.value)}
                        error={errors?.name}
                    />
                </InputContainer>
                <InputContainer label="Duration">
                    <Input
                        type="number"
                        placeholder="Duration"
                        value={newOperation.duration}
                        onChange={({ target }) =>
                            onFieldChange('duration', target.value ? +target.value :
null)
                        }
                        error={errors?.duration}
                    />
                </InputContainer>
                <Button type="submit">Submit</Button>
            </EditForm>
        </Modal>
        <Draggable
            isDragDisabled={!store.canEditSchedule || moveDisabled}

```

```

draggableId={getOperationFullId(operation, parent)}
index={index}
>
{ (provided) => (
  <Card
    ref={provided.innerRef}
    {...provided.draggableProps}
    {...provided.dragHandleProps}
    style={provided.draggableProps.style}
  >
    <CardInfoBlock>
      <CardNameLabel>{operation.name}</CardNameLabel>
      <CardLabel>Duration: {operation.duration}</CardLabel>
      {operation.deadline !== null && (
        <CardLabel>Deadline: {operation.deadline}</CardLabel>
      )}
      {operation.start !== undefined && operation.start !== null && (
        <>
          <CardLabel>Start: {operation.start}</CardLabel>
          <CardLabel>End: {operation.end}</CardLabel>
        </>
      )}
      {stats.lag !== null && (
        <>
          <CardLabel bold>Lag: {stats.lag}</CardLabel>
          <CardLabel bold>Advance: {stats.advance}</CardLabel>
        </>
      )}
    </CardInfoBlock>
    {store.canEditSchedule && (
      <Actions>
        <EditButton onClick={() => setShowModal(true)} />
        <DeleteButton
          onClick={() =>
            store.removeOperation(operation.taskId, operation.id)
          }
        />
      </Actions>
    )}
  </Card>
)}
</Draggable>
</>

```

```
);  
});
```

Файл client/src/components/schedule/TaskDnd.tsx

```
const TaskBlock = styled.div`  
  display: flex;  
  padding: 1rem;  
  border: 1px ${Colors.primary} solid;  
  border-radius: 0.25rem;  
  background-color: ${Colors.primary};  
  gap: 1rem;  
`;  
  
const DndBlock = styled.div`  
  display: flex;  
  flex-grow: 1;  
  &:hover > ${Actions} {  
    opacity: 1;  
  }  
`;  
  
const AddOperation = styled(Button)`  
  align-self: center;  
  margin-left: 1rem;  
`;  
  
interface Props {  
  task: Task;  
  moveDisabled?: boolean;  
}  
  
export default observer(function TaskDnd({ task, moveDisabled }: Props) {  
  const [showModal, setShowModal] = useState(false);  
  
  const {  
    data: newTask,  
    errors,  
    validate,  
    onFieldChange,  
  } = useForm({  
    initData: task,  
    getErrors: (newTask) => ({
```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		81

```

        name: !newTask.name ? EMPTY_ERROR : null,
        deadline: !newTask.deadline ? EMPTY_ERROR : null,
    }},
    });

const handleSubmit = (e: FormEvent) => {
    e.preventDefault();
    const { valid } = validate();
    if (!valid) {
        return;
    }
    store.updateTask(newTask);
    setShowModal(false);
};

(() => store.canEditSchedule)();

return (
    <>
        <Modal
            open={showModal}
            onClose={() => {
                setShowModal(false);
            }}
        >
            <EditForm onSubmit={handleSubmit}>
                <FormTitle>Edit Task</FormTitle>
                <InputContainer label="Name">
                    <Input
                        placeholder="Name"
                        value={newTask.name}
                        onChange={({ target }) => onFieldChange('name', target.value)}
                        error={errors?.name}
                    />
                </InputContainer>
                <InputContainer label="Deadline">
                    <Input
                        type="number"
                        placeholder="Deadline"
                        value={newTask.deadline ?? ''}
                        onChange={({ target }) =>
                            onFieldChange('deadline', target.value ? +target.value :
null)
                    </Input>
                </InputContainer>
            </EditForm>
        </Modal>
    </>
);

```

					КПІ.ІП-9213.045490.03.12	Арк.
Вмін.	Арк.	№ докум.	Підп.	Дата.		82

```

        }
        error={errors?.deadline}
      />
    </InputContainer>
    <Button type="submit">Submit</Button>
  </EditForm>
</Modal>
<TaskBlock>
  <CardInfoBlock>
    <CardNameLabel>{task.name}</CardNameLabel>
    <CardLabel>Operations: {task.operations.length}</CardLabel>
    {task.deadline !== null && (
      <CardLabel>Deadline: {task.deadline}</CardLabel>
    )}
  </CardInfoBlock>
  <Draggable
    direction="horizontal"
    draggableId={task.id}
    isDropDisabled={moveDisabled}
  >
    {(provided) => (
      <DndBlock {...provided.draggableProps} ref={provided.innerRef}>
        {task.operations.map((operation, index) => (
          <OperationCard
            key={operation.id}
            operation={operation}
            index={index}
            parent={task}
            moveDisabled={moveDisabled}
          />
        ))}
        {provided.placeholder}
        {store.canEditSchedule && (
          <>
            <AddOperation onClick={() =>
store.addNewOperation(task.id)}>
              <Icon icon={IconType.Add} />
            </AddOperation>
            {!moveDisabled && (
              <Actions>
                <EditButton onClick={() => setShowModal(true)} />
                <DeleteButton onClick={() => store.removeTask(task.id)}
              />
            )}
          </>
        )}
      )}
    )}
  </>

```

```

        </Actions>
      )}
    </>
  )}
</DndBlock>
)}
</Droppable>
</TaskBlock>
</>
);
});

```

Файл client/src/components/schedule/TasksList.tsx

```

const Tasks = styled.div`
  display: flex;
  flex-direction: column;
  padding: 1rem;
  gap: 0.5rem;
  padding-bottom: 5rem;
`;

const AddTask = styled(Button)`
  align-self: flex-start;
`;

interface Props {
  tasks: Task[];
}

export default observer(function TasksList({ tasks }: Props) {
  const handleDragEnd: OnDragEndResponder = (result) => {
    if (!result.destination) {
      return;
    }

    const sourceId = result.source.droppableId;
    const destinationId = result.destination.droppableId;

    store.moveTaskOperation(
      sourceId,
      destinationId,
      result.source.index,

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		84

```

        result.destination.index
    );
};

return (
    <DragDropContext onDragEnd={handleDragEnd}>
        <Tasks>
            {tasks?.map((task) => (
                <TaskDnd key={task.id} task={task} />
            ))}
            {store.canEditSchedule && (
                <AddTask onClick={() => store.addNewTask()}>+ New Task</AddTask>
            )}
        </Tasks>
    </DragDropContext>
);
});

```

Файл client/src/components/schedule/utils.ts

```

export enum ItemTypes {
    Operation = 'operation',
    Device = 'device',
}

const DELIMITER = ':' as const;

export const getDeviceFullId = (device: Device) => {
    return [device.layerId, device.id].join(DELIMITER);
};

export const getOperationFullId = (
    operation: Operation,
    parent: Device | Task | string
) => {
    if (typeof parent === 'string') {
        return [parent, operation.id].join(DELIMITER);
    }
    if ('layerId' in parent) {
        return [parent.layerId, parent.id, operation.id].join(DELIMITER);
    }
    return [parent.id, operation.id].join(DELIMITER);
};

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		85

```
export const getPartsFromFullId = (fullId: string): string[] => {
  return fullId.split(DELIMITER);
};

export const EMPTY_ERROR = 'Should not be empty';
```

Файл client/src/components/schedule/SingleDeviceScheduleForm.tsx

```
const FOOTER_HEIGHT = '3rem';

const ScheduleEditBlockWrapper = styled.div`
  height: calc(100vh - ${NAVIGATION_BAR_HEIGHT} - ${TABS_HEADER_HEIGHT});

  display: flex;
  flex-direction: column;
  padding: 1rem 0;
  & > div {
    flex-grow: 1;
    & > div:last-child {
      flex-grow: 1;
    }
  }
`;

const EditBlockWrapper = styled.div`
  flex-grow: 1;
  width: 100vw;
  height: calc(100vh - ${NAVIGATION_BAR_HEIGHT} - ${TABS_HEADER_HEIGHT});
  display: flex;
  position: relative;
`;

const EditBlock = styled.div`
  flex-grow: 1;
  display: flex;

  overflow: auto;
  align-items: flex-start;
  margin-bottom: ${FOOTER_HEIGHT};
```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		86

```
`;

const ConfigurationEditContent = styled.div`
  display: flex;
`;

const Label = styled.span`
  ${regularFont('1.25rem')}
  white-space: nowrap;
`;

const StyledSpinner = styled(Spinner)`
  &&& {
    flex-grow: unset;
    margin: auto;
  }
`;

const Footer = styled.div`
  position: sticky;
  bottom: 0;
  left: 0;
  width: 100%;
  min-height: ${FOOTER_HEIGHT};
  max-height: ${FOOTER_HEIGHT};

  box-shadow: 0px -3px 4px rgba(0, 0, 0, 0.15);
  background-color: ${Colors.background};

  display: flex;
  align-items: center;
  justify-content: flex-end;
  gap: 0.5rem;
  & > :last-child {
    margin-right: 0.5rem;
  }
  & > :first-child {
    margin-left: 0.5rem;
  }
`;

const LeftButton = styled(Button)`
  margin-right: auto;
```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		87

```
`;

const ModalBody = styled.div`
  display: flex;
  flex-direction: column;
  align-items: stretch;

  ${ButtonGroup} {
    align-self: stretch;
    justify-content: center;
    padding: 0.5rem;
    gap: 0.5rem;
    ${Button} {
      flex-grow: 1;
    }
  }
`;

enum ScheduleTab {
  Tasks = 'tasks',
  Configuration = 'configuration',
}

const ScheduleTabLabels: Record<ScheduleTab, string> = {
  [ScheduleTab.Tasks]: 'Tasks',
  [ScheduleTab.Configuration]: 'Configuration',
};

const ErrorLabel = styled.span`
  color: ${Colors.red};
  ${boldFont('18px')}
  margin-right: auto;
  border: 2px solid ${Colors.red};
  border-radius: 0.25rem;
  padding: 0.25rem 0.5rem;
`;

interface Props {
  formType: FormType;
  onEdit: () => void;
}

export default observer(function SingleDeviceScheduleForm({
```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		88

```

    formType,
    onEdit,
  }: Props) {
    const [activeTab, setActiveTab] = useState(ScheduleTab.Tasks);
    const [error, setError] = useState<string | null>();
    const [showModal, setShowModal] = useState(false);
    const location = useLocation();
    const navigate = useNavigate();
    const schedule = store.schedule;

    const saveSchedule = async () => {
      const error = validateSchedule(schedule);
      setError(error);
      if (error) {
        return;
      }
      if (formType === FormType.New) {
        await createSchedule(
          { ...schedule, ownerId: store.user.userId },
          store.accessToken
        );
      } else {
        await updateSchedule(schedule, store.accessToken);
      }

      navigate('/schedules');
    };

    const tabs: TabItem<ScheduleTab>[] = [
      {
        key: ScheduleTab.Tasks,
        label: ScheduleTabLabels.tasks,
        component: (
          <EditBlockWrapper>
            <EditBlock>
              <ConfigurationEditContent>
                <TasksList tasks={schedule?.tasks as Task[]} />
              </ConfigurationEditContent>
            </EditBlock>
          </EditBlockWrapper>
        ),
      },
    ],
    {

```

```

        key: ScheduleTab.Configuration,
        label: ScheduleTabLabels.configuration,
        component: (
            <EditBlockWrapper>
                <EditBlock>
                    <ConfigurationEditContent>
                        <LayersList
                            tasks={schedule.tasks}
                            layers={schedule?.layers as Layer[]}
                        />
                    </ConfigurationEditContent>
                </EditBlock>
            </EditBlockWrapper>
        ),
    },
];

const handleTabChange = (tab: ScheduleTab) => {
    navigate(`?tab=${tab}`, { relative: 'route', replace: true });
};

useEffect(() => {
    const currentTab = new URLSearchParams(location.search).get('tab');
    if (!currentTab) {
        handleTabChange(ScheduleTab.Tasks);
        return;
    }
    setActiveTab(currentTab as ScheduleTab);
}, [location.search]);

const handleOptimizeClick = async (
    type: SingleDeviceScheduleOptimizationType
) => {
    const error = validateSchedule(schedule);
    setError(error);
    if (error) {
        return;
    }
    await store.optimizeSingleDeviceSchedule(type);
    setShowModal(false);
};

return (

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		90

```

<>
<Modal open={showModal} onClose={() => setShowModal(false)}>
  <ModalBody>
    <Label>Choose optimization method</Label>
    <ButtonGroup>
      <Button
        onClick={() =>
          handleOptimizeClick(SingleDeviceScheduleOptimizationType.Fast)
        }
      >
        Fast
      </Button>
      <Button
        onClick={() =>
          handleOptimizeClick(
            SingleDeviceScheduleOptimizationType.Complex
          )
        }
      >
        Complex
      </Button>
    </ButtonGroup>
  </ModalBody>
</Modal>
<ScheduleEditBlockWrapper>
  {store.loading ? (
    <StyledSpinner />
  ) : (
    <>
      <Tabs
        items={tabs}
        selected={activeTab}
        onSelect={handleTabChange}
      />
      <Footer>
        {!store.canEditSchedule && (
          <Button onClick={() => onEdit()}>Edit Schedule</Button>
        )}

        <LeftButton onClick={() => setShowModal(true)}>
          Optimize
        </LeftButton>
      </Footer>
    </>
  )}

```

```

        {error && <ErrorLabel>{error}</ErrorLabel>}
        <Button outlined onClick={() => navigate('/schedules')}>
            Close
        </Button>
        <Button onClick={saveSchedule}>Save & Close</Button>
    </Footer>
</>
    )}
</ScheduleEditBlockWrapper>
</>
);
});

```

Файл client/src/components/schedule/MultiDeviceScheduleForm.tsx

```

const ScheduleEditBlockWrapper = styled.div`
    height: calc(100vh - ${NAVIGATION_BAR_HEIGHT} - ${TABS_HEADER_HEIGHT});

    display: flex;
    flex-direction: column;
    padding: 1rem 0;
    & > div {
        flex-grow: 1;
        & > div:last-child {
            flex-grow: 1;
        }
    }
`;

const EditBlockWrapper = styled.div`
    flex-grow: 1;
    width: 100vw;
    height: calc(100vh - ${NAVIGATION_BAR_HEIGHT} - ${TABS_HEADER_HEIGHT});
    display: flex;
    position: relative;
`;

const EditBlock = styled.div`
    flex-grow: 1;
    display: flex;

    overflow: auto;
    align-items: flex-start;

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		92

```

    margin-bottom: ${FOOTER_HEIGHT};
`;

const ConfigurationEditContent = styled.div`
  display: flex;
`;

const Footer = styled.div`
  position: sticky;
  bottom: 0;
  left: 0;
  width: 100%;
  min-height: ${FOOTER_HEIGHT};
  max-height: ${FOOTER_HEIGHT};

  box-shadow: 0px -3px 4px rgba(0, 0, 0, 0.15);
  background-color: ${Colors.background};

  display: flex;
  align-items: center;
  justify-content: flex-end;
  gap: 0.5rem;
  & > :last-child {
    margin-right: 0.5rem;
  }
  & > :first-child {
    margin-left: 0.5rem;
  }
`;

const LeftButton = styled(Button)`
  margin-right: auto;
`;

const ErrorLabel = styled.span`
  color: ${Colors.red};
  ${boldFont('18px')}
  margin-right: auto;
  border: 2px solid ${Colors.red};
  border-radius: 0.25rem;
  padding: 0.25rem 0.5rem;
`;

```

```

const FormWrapper = styled.div`
  display: flex;
  flex-direction: column;
  padding: 0.5rem;
  padding-bottom: 1.5rem;
  align-items: flex-start;
  gap: 1rem;
`;

interface Props {
  formType: FormType;
  onEdit: () => void;
}

export default observer(function MultiDeviceScheduleForm({
  formType,
  onEdit,
}: Props) {
  const [error, setError] = useState<string | null>();
  const navigate = useNavigate();

  const schedule = store.schedule;

  const saveSchedule = async () => {
    const error = validateSchedule(schedule);
    setError(error);
    if (error) {
      return;
    }
    if (formType === FormType.New) {
      await createSchedule(
        { ...schedule, ownerId: store.user.userId },
        store.accessToken
      );
    } else {
      await updateSchedule(schedule, store.accessToken);
    }

    navigate('/schedules');
  };

  const handleOptimizeClick = async () => {
    const error = validateSchedule(schedule);

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		94

```

    setError(error);
    if (error) {
      return;
    }
    store.optimizeMultiDeviceSchedule();
  };

  return (
    <ScheduleEditBlockWrapper>
      <EditBlockWrapper>
        <EditBlock>
          <ConfigurationEditContent>
            <div>
              <DragDropContext onDragEnd={() => {}}>
                <FormWrapper>
                  <TaskDnd
                    moveDisabled
                    task={{
                      ...(schedule.tasks[0] ??
                        getDefaultSingleOperationTask(schedule.id)),
                      operations: schedule.tasks.map(
                        (task) => task.operations[0]
                      ),
                    }}
                  />
                <LayerDnd layer={schedule.layers[0]} moveDisabled />
              </FormWrapper>
            </DragDropContext>
          </div>
        </ConfigurationEditContent>
      </EditBlock>
    </EditBlockWrapper>
    <Footer>
      {!store.canEditSchedule && (
        <Button onClick={onEdit}>Edit Schedule</Button>
      )}

      <LeftButton onClick={handleOptimizeClick}>Optimize</LeftButton>
      {error && <ErrorLabel>{error}</ErrorLabel>}
      <Button outlined onClick={() => navigate('/schedules')}>
        Close
      </Button>
      <Button onClick={saveSchedule}>Save & Close</Button>
    </Footer>
  );

```

```

        </Footer>
      </ScheduleEditBlockWrapper>
    );
  });

```

Файл client/src/services/schedule.service.ts

```

export const getScheduleById = async (
  id: string,
  accessToken: string
): Promise<Schedule> => {
  const res = await fetch(
    `${process.env.REACT_APP_SERVER_URL}${FullSchedulesByIdRoute}${id}`,
    {
      mode: "cors",
      headers: getHeaders(accessToken),
    }
  );
};

const serializedSchedule: SerializedSchedule = await res.json();
return deserializeSchedule(serializedSchedule);
// return SCHEDULE;
};

export const createSchedule = async (
  schedule: Schedule,
  accessToken: string
) => {
  await fetch(process.env.REACT_APP_SERVER_URL + FullCreateScheduleRoute, {
    method: "POST",
    body: JSON.stringify({ schedule: serializeSchedule(schedule) }),
    headers: getHeaders(accessToken),
    mode: "cors",
  });
};

export const updateSchedule = async (
  schedule: Schedule,
  accessToken: string
) => {
  await fetch(process.env.REACT_APP_SERVER_URL + FullUpdateScheduleRoute, {
    method: "PATCH",
    body: JSON.stringify({ schedule: serializeSchedule(schedule) }),

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		96

```

        headers: getHeaders(accessToken),
        mode: "cors",
    });
};

export const removeSchedule = async (
    scheduleId: string,
    accessToken: string
) => {
    await fetch(
        process.env.REACT_APP_SERVER_URL +
        FullDeleteScheduleRoute +
        "/" +
        scheduleId,
        {
            method: "DELETE",
            headers: getHeaders(accessToken),
            mode: "cors",
        }
    );
};

export const getMySchedules = async (
    accessToken: string
): Promise<ScheduleStats[]> => {
    const res = await fetch(
        process.env.REACT_APP_SERVER_URL + FullMySchedulesRoute,
        {
            headers: getHeaders(accessToken),
            mode: "cors",
        }
    );

    return await res.json();
};

export const optimizeSingleDeviceSchedule = async (
    schedule: Schedule,
    type: SingleDeviceScheduleOptimizationType,
    accessToken: string
) => {
    const res = await fetch(
        process.env.REACT_APP_SERVER_URL + FullSingleDeviceScheduleOptimizeRoute,

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		97

```

    {
      method: "POST",
      body: JSON.stringify({
        schedule: serializeSchedule(schedule),
        useRmax: type === SingleDeviceScheduleOptimizationType.Fast,
      }),
      headers: getHeaders(accessToken),
      mode: "cors",
    }
  );

  return deserializeSchedule(await res.json());
};

export const optimizeMultiDeviceSchedule = async (
  schedule: Schedule,
  accessToken: string
) => {
  const res = await fetch(
    process.env.REACT_APP_SERVER_URL + FullMultiDeviceScheduleOptimizeRoute,
    {
      method: "POST",
      body: JSON.stringify({
        schedule: serializeSchedule(schedule),
      }),
      headers: getHeaders(accessToken),
      mode: "cors",
    }
  );

  return deserializeSchedule(await res.json());
};

```

Файл client/src/store/store.ts

```

export class Store {
  accessToken!: string;
  online = false;
  user!: User;
  initialized = false;
  schedule!: Schedule;
  loading = false;

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		98

```

canEditSchedule = false;

constructor() {
  makeAutoObservable(this);
  this.initStore();

  setInterval(() => this.initStore(), 100000);
}

initStore() {
  refresh()
    .then(async ({ accessToken }) => {
      this.setAccessToken(accessToken);
      await this.setMyProfile();
      this.schedule.ownerId = this.user.userId;
    })
    .catch(() => {
      runInAction(() => {
        this.online = false;
        this.initialized = true;
      });
    });
}

setUser(newUserInfo: User): void {
  this.user = newUserInfo;
}

async setAccessToken(accessToken: string): Promise<void> {
  this.accessToken = accessToken;
  await this.setMyProfile();
}

async setMyProfile(): Promise<void> {
  try {
    const user = await getMyProfile(this.accessToken);
    runInAction(() => {
      this.user = user;
    });
  } catch (err) {
  } finally {
    runInAction(() => {
      this.initialized = true;
    });
  }
}

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		99

```

    });
  }
}

resetSchedule() {
  //@ts-ignore
  this.schedule = null;
}

async getSchedule(scheduleId: string) {
  runInAction(() => {
    this.loading = true;
  });
  const newSchedule = await getScheduleById(scheduleId, this.accessToken);
  runInAction(() => {
    this.schedule = newSchedule;
    this.loading = false;
  });
  this.setCanEditSchedule(false);
}

createNewSchedule(name: string, type: ScheduleType) {
  this.schedule = { ...getDefaultSchedule(store.user.userId, type), name };
  if (type === ScheduleType.MultiDevice) {
    this.schedule.tasks = [];
  }
  this.setCanEditSchedule(true);
}

setCanEditSchedule(value: boolean) {
  this.canEditSchedule = value;
  if (value) {
    if (this.schedule.type === ScheduleType.SingleDevice) {
      this.schedule.tasks.forEach((task) =>
        task.operations.forEach((operation, index) => {
          operation.start = null;
          operation.end = null;
          if (index !== task.operations.length - 1) {
            operation.deadline = null;
          }
        })
      );
    }
  }
};

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		100

```

    this.schedule.layers.forEach((layer, index) =>
      layer.devices.forEach((device) => {
        device.start = null;
        device.end = null;
        device.operations.forEach((operation) => {
          operation.start = null;
          operation.end = null;
          if (index !== this.schedule.layers.length - 1) {
            operation.deadline = null;
          }
        });
      });
    );
  } else {
    this.schedule.tasks.forEach((task) => {
      task.operations.forEach((operation) => {
        operation.deviceId = null;
        operation.start = null;
        operation.end = null;
      });
    });
    for (const i in this.schedule.layers[0].devices) {
      this.schedule.layers[0].devices[i] = {
        ...this.schedule.layers[0].devices[i],
        operations: [],
        end: null,
      };
    }
  }
}

addNewLayer() {
  this.schedule.layers.push(getDefaultLayer(this.schedule?.id as string));
}

updateLayer(layer: Layer) {
  const index = this.schedule.layers.findIndex((l) => layer.id === l.id);
  this.schedule.layers.splice(index, 1, layer);
}

removeLayer(layerId: string) {
  const [layer] = this.schedule.layers.splice(

```

```

        this.schedule.layers.findIndex((layer) => layer.id === layerId),
        1
    );

    layer.devices.forEach((device) =>
        this.schedule.tasks.forEach((task) => {
            task.operations.forEach((operation) => {
                if (device.operations.find((op) => op.id === operation.id)) {
                    operation.deviceId = null;
                }
            });
        });
    );
}

addNewDevice(layerId: string) {
    const newDevice = getDefaultDevice(layerId);
    if (this.schedule.type === ScheduleType.MultiDevice) {
        newDevice.start = 0;
    }
    this.schedule.layers
        ?.find((layer) => layer.id === layerId)
        ?.devices?.push(newDevice);
}

updateDevice(device: Device) {
    const layer = this.schedule.layers.find(
        (layer) => layer.id === device.layerId
    ) as Layer;

    const index = layer.devices.findIndex((d) => d.id === device.id);
    layer.devices.splice(index, 1, device);
}

removeDevice(layerId: string, deviceId: string) {
    const layer = this.schedule.layers.find(
        (layer) => layer.id === layerId
    ) as Layer;
    const [device] = layer?.devices.splice(
        layer.devices.findIndex((device) => device.id === deviceId),
        1
    );
};

```

```

        this.schedule.tasks.forEach((task) => {
            task.operations.forEach((operation) => {
                if (device.operations.find((op) => op.id === operation.id)) {
                    operation.deviceId = null;
                }
            });
        });
    });
}

addNewTask() {
    this.schedule.tasks.push(getDefaultTask(this.schedule.id));
}

updateTask(task: Task) {
    const index = this.schedule.tasks.findIndex((t) => t.id === task.id);
    this.schedule.tasks.splice(index, 1, task);
    if (task.operations.length !== 0) {
        task.operations[task.operations.length - 1].deadline = task.deadline;
        this.updateOperation(task.operations[task.operations.length - 1]);
    }
}

removeTask(taskId: string) {
    const [task] = this.schedule.tasks.splice(
        this.schedule.tasks.findIndex((task) => task.id === taskId),
        1
    );
    this.schedule.layers.forEach((layer) => {
        layer.devices.forEach((device) => {
            device.operations = device.operations.filter(
                (operation) => !task.operations.find((op) => op.id ===
operation.id)
            );
        });
    });
}

addNewOperation(taskId: string) {
    if (this.schedule.type === ScheduleType.SingleDevice) {
        const task = this.schedule.tasks.find(
            (task) => task.id === taskId
        ) as Task;
        task.operations.push(getDefaultOperation(taskId));
    }
}

```

```

        this.updateTaskOperationDeadlines(task);
    } else {
        const task: Task = getDefaultSingleOperationTask(this.schedule.id);
        const operation = getDefaultOperation(task.id);
        task.operations = [operation];
        this.schedule.tasks.push(task);
    }
}

updateOperation(operation: Operation) {
    const task = this.schedule.tasks.find(
        (task) => task.id === operation.taskId
    ) as Task;

    const index = task.operations.findIndex((op) => op.id === operation.id);

    const presetOperation = {
        ...operation,
        ...(index === task.operations.length - 1
            ? { deadline: task.deadline }
            : {}),
    };

    const [oldOperation] = task.operations.splice(index, 1, presetOperation);

    if (presetOperation.deviceId) {
        for (const layer of this.schedule.layers) {
            const device = layer.devices.find(
                (device) => device.id === presetOperation.deviceId
            );
            device?.operations.splice(
                device.operations.findIndex((op) => op.id === presetOperation.id),
                1,
                presetOperation
            );
        }
    } else if (oldOperation.deviceId) {
        for (const layer of this.schedule.layers) {
            const device = layer.devices.find(
                (device) => device.id === oldOperation.deviceId
            );
            device?.operations.splice(
                device.operations.findIndex((op) => op.id === oldOperation.id),

```

```

        1
    );
}
}

removeOperation(taskId: string, operationId: string) {
    if (this.schedule.type === ScheduleType.SingleDevice) {
        const task = this.schedule.tasks.find(
            (task) => task.id === taskId
        ) as Task;
        const [operation] = task.operations.splice(
            task.operations.findIndex((operation) => operation.id ===
operationId),
            1
        );
        if (operation.deviceId) {
            for (const layer of this.schedule.layers) {
                const device = layer.devices.find(
                    (device) => device.id === operation.deviceId
                );
                if (device) {
                    device.operations.splice(
                        device.operations.findIndex((op) => op.id === operation.id),
                        1
                    );
                }
            }
            this.updateTaskOperationDeadlines(task);
        } else {
            const index = this.schedule.tasks.findIndex((task) => task.id ===
taskId);
            this.schedule.tasks.splice(index, 1);
        }
    }

    updateTaskOperationDeadlines(task: Task) {
        task.operations.forEach(({ deadline, ...operation }, index) => {
            this.updateOperation({
                ...operation,
                deadline: index === task.operations.length - 1 ? task.deadline :
null,

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		105

```

    });
  });
}

get notUsedOperations() {
  return this.schedule.tasks.reduce((prev, curr) => {
    curr.operations.forEach((operation, index) => {
      const layerId = this.schedule.layers[index]?.id ?? 'new-layer';
      if (!operation.deviceId) {
        if (!prev[layerId]) {
          prev[layerId] = [operation];
        } else {
          prev[layerId].push(operation);
        }
      }
    });
    return prev;
  }, {} as Record<string, Operation[]>);
}

moveOperation(
  layerId: string,
  deviceId: string,
  newDeviceId: string,
  operationIndex: number,
  newIndex: number
) {
  if (deviceId === newDeviceId && operationIndex === newIndex) {
    return;
  }
  let operation: Operation;
  if (deviceId === NOT_USED_OPERATIONS) {
    operation = this.notUsedOperations[layerId][operationIndex];
  } else {
    const sourceDevice = this.schedule.layers
      .find((layer) => layer.id === layerId)
      ?.devices.find((device) => device.id === deviceId) as Device;
    [operation] = sourceDevice.operations.splice(operationIndex, 1);
  }
  const destinationDevice = this.schedule.layers
    .find((layer) => layer.id === layerId)
    ?.devices.find((device) => device.id === newDeviceId) as Device;

```

```

        operation.deviceId = destinationDevice.id;
        destinationDevice?.operations.splice(newIndex, 0, operation);
        this.updateOperation(operation);
    }

    moveTaskOperation(
        taskId: string,
        newTaskId: string,
        operationIndex: number,
        newOperationIndex: number
    ) {
        if (taskId === newTaskId && operationIndex === newOperationIndex) {
            return;
        }
        const sourceTask = this.schedule.tasks.find(
            (task) => task.id === taskId
        ) as Task;
        const destinationTask = this.schedule.tasks.find(
            (task) => task.id === newTaskId
        ) as Task;

        const [operation] = sourceTask.operations.splice(operationIndex, 1);
        destinationTask.operations.splice(newOperationIndex, 0, {
            ...operation,
            taskId: destinationTask.id,
        });

        if (taskId === newTaskId) {
            for (
                let i = Math.min(operationIndex, newOperationIndex);
                i <= Math.max(operationIndex, newOperationIndex);
                i++
            ) {
                this.updateOperation({ ...sourceTask.operations[i], deviceId: null
            });
            }
        } else {
            for (let i = operationIndex; i < sourceTask.operations.length; i++) {
                this.updateOperation({ ...sourceTask.operations[i], deviceId: null
            });
            }
            for (
                let i =

```

```

        operationIndex !== newOperationIndex
        ? newOperationIndex
        : newOperationIndex + 1;
    i < destinationTask.operations.length;
    i++
  ) {
    this.updateOperation({
      ...destinationTask.operations[i],
      deviceId: null,
    });
  }
}

this.updateTaskOperationDeadlines(sourceTask);
this.updateTaskOperationDeadlines(destinationTask);
}

async optimizeSingleDeviceSchedule(
  type: SingleDeviceScheduleOptimizationType
) {
  const newSchedule = await optimizeSingleDeviceSchedule(
    this.schedule,
    type,
    this.accessToken
  );
  this.schedule = newSchedule;
  this.setCanEditSchedule(false);
}

async optimizeMultiDeviceSchedule() {
  const newSchedule = await optimizeMultiDeviceSchedule(
    {
      ...this.schedule,
      layers: this.schedule.layers.map((layer) => ({
        ...layer,
        devices: layer.devices.map((device) => ({
          ...device,
          operations: [],
        })),
      })),
    },
    this.accessToken
  );
}

```

```

        this.schedule = newSchedule;
        this.setCanEditSchedule(false);
    }

    async deleteSchedule() {
        await removeSchedule(this.schedule.id, this.accessToken);
        this.resetSchedule();
    }
}

let store = new Store();

export function reloadChatStore(): void {
    store = new Store();
}

export default store;

```

Файл client/src/App.tsx

```

const MainLayout = styled.div`
    display: flex;
    flex-direction: column;
    height: 100vh;
    background: ${Colors.background};
    @media (max-width: 50rem) {
        flex-direction: column;
    }
`;

const AppView = styled.div`
    min-width: none;
    height: 100vh;
    color: ${Colors.dark};
    overflow: hidden;
`;

export default observer(function App() {
    return (
        <AppView>
            <Router>
                <MainLayout>
                    {store?.user ? <NavigationBar /> : null}

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		109

```

    <Routes>
      <Route path="/" element={<LoginPage />}></Route>
      <Route path="registration" element={<RegistrationPage
/>}></Route>
      <Route path="users/:id" element={<ProfilePage />} />
      <Route path="schedules" element={<SchedulesPage />} />
      <Route
        path="schedules/new"
        element={<SchedulePage formType={FormType.New} />}
      />
      <Route
        path="schedules/:id"
        element={<SchedulePage formType={FormType.Edit} />}
      />
      <Route path="*" element={<Navigate to="/" replace />} />
    </Routes>
  </MainLayout>
</Router>
</AppView>
);
});

```

					КПІ.ІП-9213.045490.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		110

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

**Програмне забезпечення для календарного планування виконання
ІТ-проектів**

Програма та методика тестування

КПІ. ІІ-9213.045490.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена ХАЛУС

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Михайло МЕДВЕДЄВ

Київ – 2023

ЗМІСТ

1	ОБ’ЄКТ ВИПРОБУВАНЬ.....	3
2	МЕТА ТЕСТУВАННЯ	4
3	МЕТОДИ ТЕСТУВАННЯ.....	5
4	ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ	6

					КПІ.ІП-9213.045490.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		2

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є програмне забезпечення для календарного планування виконання ІТ-проєктів. Створений застосунок був розроблений для роботи у більшості сучасних веб-браузерів, таких як Chrome, Firefox, Edge чи Safari.

					КПІ.ІП-9213.045490.04.51	Арк.
						3
Змін	Арк.	№ докум.	Підп.	Дата.		

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

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

					КПІ.ІП-9213.045490.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		4

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- системне тестування – перевіряється усе програмне забезпечення в цілому;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

					КПІ.ІП-9213.045490.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		5

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в сумісності додатку з більшістю сучасних веб-браузерів та зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- статичне тестування коду програмного забезпечення;
- динамічне тестування на відповідність функціональним вимогам;
- тестування роботи застосунку у різних веб-браузерах;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування інтерфейсу користувача;
- тестування зручності використання.

					КПІ.ІП-9213.045490.04.51	Арк.
						6
Змін	Арк.	№ докум.	Підп.	Дата.		

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

**Програмне забезпечення для календарного планування виконання
ІТ-проектів**

Керівництво користувача

КПІ.ПІ-9213.045490.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена ХАЛУС

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Михайло МЕДВЕДСВ

Київ – 2023

ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	4
2.1	Системні вимоги для коректної роботи.....	4
2.2	Завантаження застосунку.....	4
2.3	Перевірка коректної роботи.....	4
3	ВИКОНАННЯ ПРОГРАМИ	5

					КПІ.ІП-9213.045490.05.34	Арк.
						2
Змін.	Арк.	№ докум.	Підп.	Дата.		

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Програмне забезпечення для календарного планування виконання ІТ-проектів являє собою веб-додаток, у якому за допомогою графічного інтерфейсу користувач може створювати розклади та оптимізовувати їх за допомогою сучасних конструктивних алгоритмів.

Додаток розгорнутий на віртуальній машині на хмарному сервісі і доступний для використання у мережі Інтернет.

					КПІ.ІП-9213.045490.05.34	Арк.
						3
Змін.	Арк.	№ докум.	Підп.	Дата.		

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

- наявність ПК з операційною системою Windows, MacOS або Linux;
- наявність останньої версії браузеру Chrome, Firefox, Edge або Safari;
- наявність доступу до Інтернету.

2.2 Завантаження застосунку

Отримати доступ до розгорнутого додатку можна за посиланням <https://schedule-project.tech>. Для розгортання додатку власноруч потрібно завантажити весь код та запустити скрипт у командному рядку `setup.sh`.

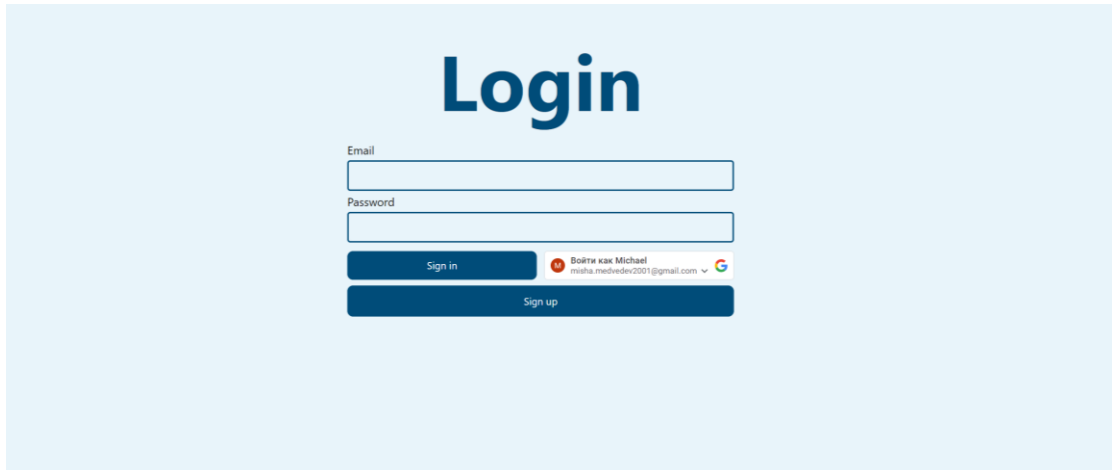
2.3 Перевірка коректної роботи

По завершенню роботи скрипту `setup.sh` додаток буде доступний користувачу у веб-браузері за посиланням <http://localhost>. У разі, якщо за даним посиланням сайт не відображається, то розгортання застосунку відбулось не успішно. Інакше користувач потрапляє на сторінку входу в систему.

					КПІ.ІП-9213.045490.05.34	Арк.
						4
Змін.	Арк.	№ докум.	Підп.	Дата.		

3 ВИКОНАННЯ ПРОГРАМИ

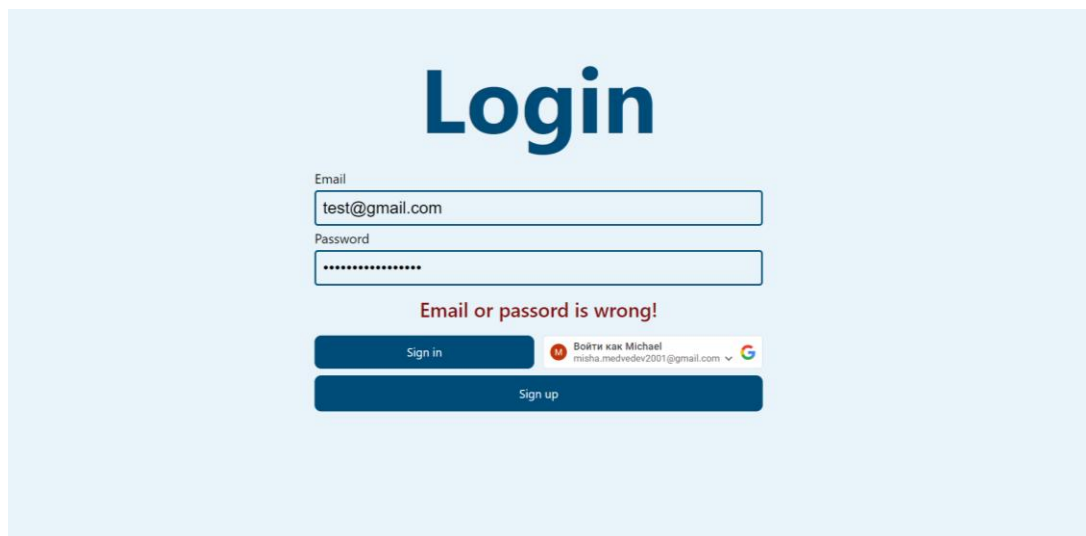
При вході у веб-застосунок користувач потрапляє на сторінку входу в систему. Користувач може обрати, яким методом він може авторизуватись – за допомогою електронної пошти та паролю чи за допомогою Google.



The screenshot shows a light blue background with the word "Login" in large blue font at the top. Below it are two input fields: "Email" and "Password". To the right of the "Sign in" button is a Google login option with the text "Войти как Michael" and "misha.medvedev2001@gmail.com".

Рисунок 3.1 – Сторінка авторизації в системі

Після введення електронної адреси та паролю користувач повинен натиснути кнопку “Sign in” для авторизації. Якщо електронна пошта або пароль були введені невірно – користувач отримає повідомлення про виникнення помилки при спробі входу в систему.



The screenshot shows the same login page as Figure 3.1, but with the "Email" field containing "test@gmail.com" and the "Password" field filled with dots. Below the input fields, a red error message "Email or password is wrong!" is displayed. The "Sign in" button is highlighted.

Рисунок 3.2 – Форма входу в систему з помилкою

Якщо дані будуть введені вірно – користувач буде перенаправлений на сторінку перегляду профілю.

Для того, щоб увійти в систему за допомогою Google, користувач повинен натиснути кнопку “Sign in with Google”. Після цього користувач перенаправляється на сторінку входу в акаунт Google. Якщо вхід буде виконано успішно, користувач буде перенаправлений на сторінку перегляду профілю.

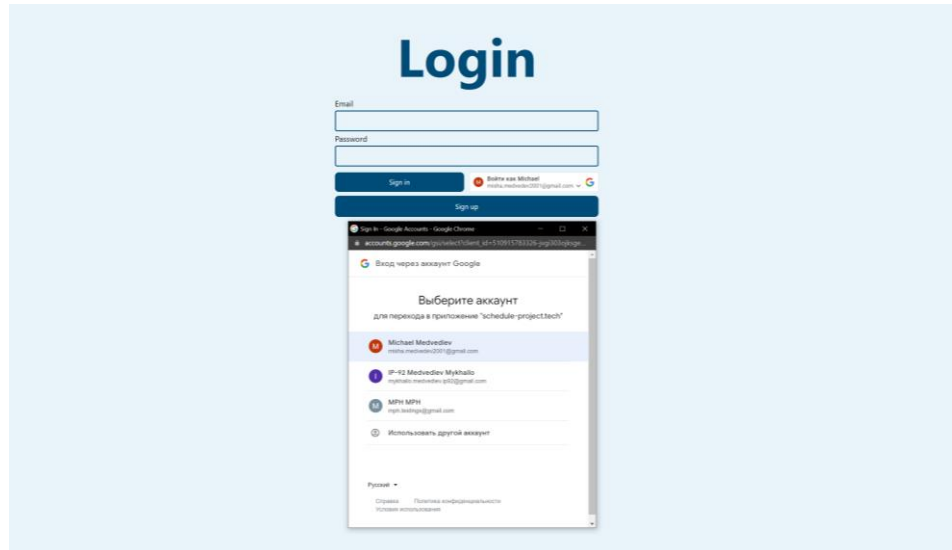


Рисунок 3.3 – Вхід у систему за допомогою сервісу Google

Для реєстрації нового користувача потрібно натиснути кнопку “Sign up”. Після цього користувач перенаправляється на сторінку реєстрації. Для успішної реєстрації користувач повинен заповнити такі поля, як ім’я, прізвище, електронну адресу та пароль. При цьому пароль повинен складатись мінімум з 7 символів та містити в собі щонайменше одну літеру верхнього регістру, одну літеру нижнього регістру та один спеціальний СИМВОЛ.

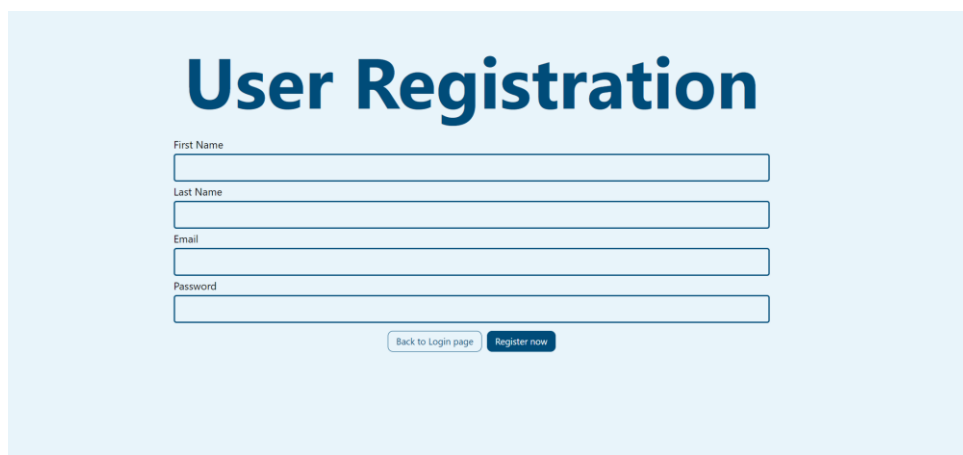
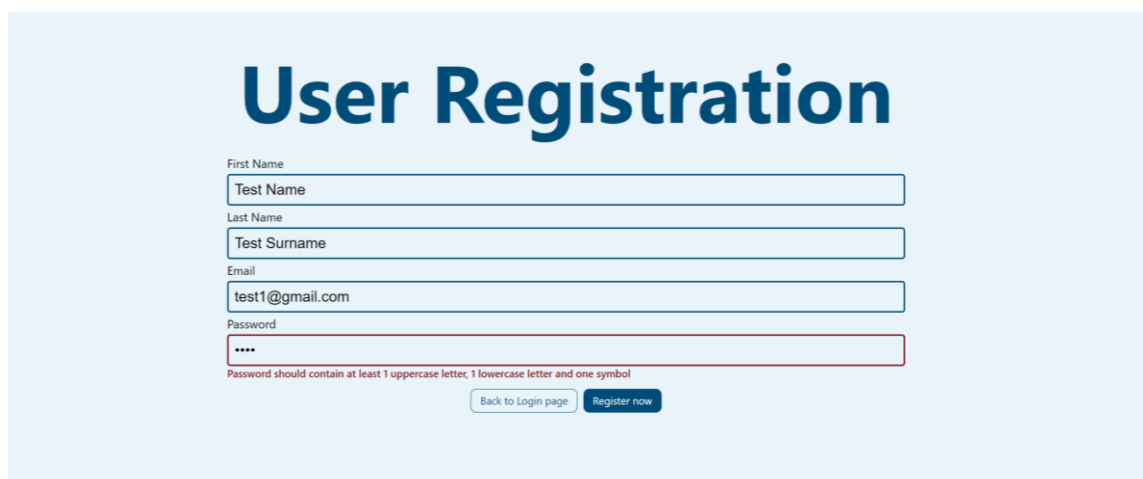


Рисунок 3.4 – Сторінка реєстрації користувача у системі

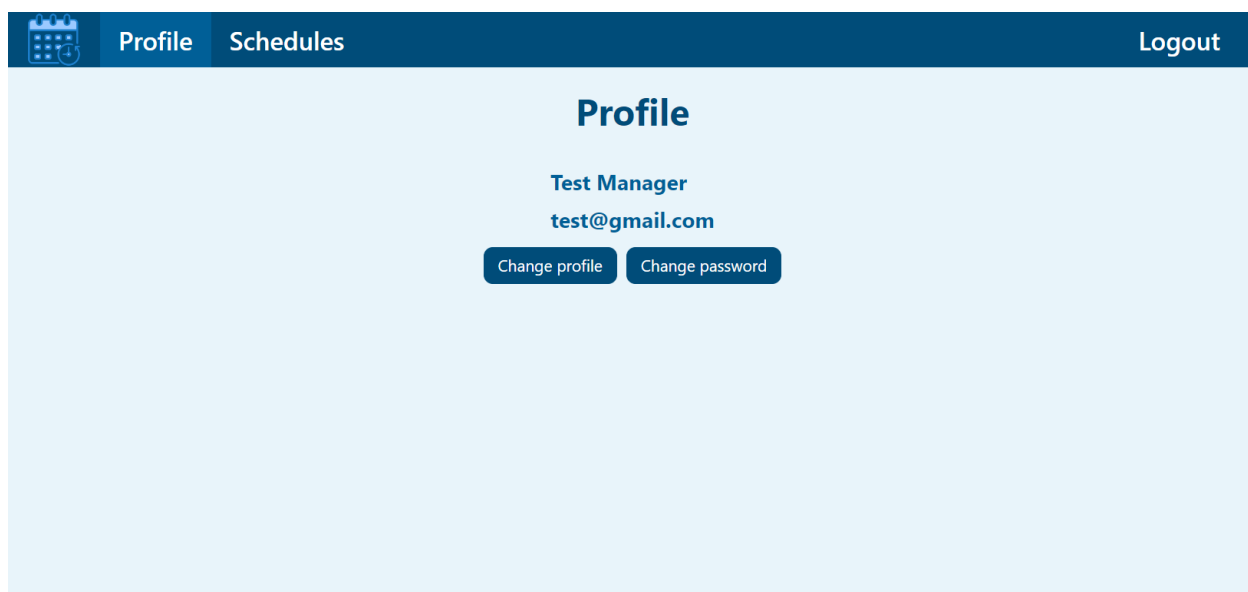
Якщо введений пароль не відповідатиме поставленим вимогам, користувач отримає інформативне повідомлення про помилку.



The image shows a 'User Registration' form on a light blue background. The title 'User Registration' is in a large, bold, dark blue font. Below the title are four input fields: 'First Name' with the value 'Test Name', 'Last Name' with the value 'Test Surname', 'Email' with the value 'test1@gmail.com', and 'Password' with four asterisks. Below the password field is a red error message: 'Password should contain at least 1 uppercase letter, 1 lowercase letter and one symbol'. At the bottom of the form are two buttons: 'Back to Login page' and 'Register now'.

Рисунок 3.5 – Форма реєстрації користувача з повідомленням про помилку

Якщо всі дані введені вірно, користувач автоматично авторизується в системі та потрапляє на сторінку профілю. На даній сторінці користувач може переглянути свої особисті дані у системі та редагувати їх.



The image shows a 'User Profile' page. At the top is a dark blue navigation bar with a calendar icon, the text 'Profile Schedules', and a 'Logout' link. The main content area has a light blue background. The title 'Profile' is in a bold, dark blue font. Below the title, the text 'Test Manager' and 'test@gmail.com' are displayed. At the bottom are two buttons: 'Change profile' and 'Change password'.

Рисунок 3.6 – Сторінка перегляду профілю користувача

Для редагування особистих даних користувач повинен натиснути кнопку “Change profile”. Після цього на сторінці з’явиться модальне вікно, де користувач може ввести нові значення імені та прізвища.

Рисунок 3.7 – Форма редагування особистих даних користувача

Після введення даних у формі, користувач повинен натиснути кнопку “Save changes”. Після цього користувач повертається на сторінку перегляду профілю.

Для зміни паролю користувач повинен на сторінці профілю натиснути кнопку “Change password”. Після цього користувачу відкриється модальне вікно, де користувач повинен ввести такі дані, як старий пароль, новий пароль та підтвердження нового паролю.

					КП.ІП-9213.045490.05.34	Арк.
						8
Змін.	Арк.	№ докум.	Підп.	Дата.		

Рисунок 3.8 – Форма зміни паролю користувача

Після того, як користувач заповнить всі необхідні поля, потрібно натиснути кнопку “Change password”, після чого користувач буде перенаправлений на сторінку профілю.

Для перегляду створених розкладів користувач повинен натиснути на кнопку “Schedules” у верхньому меню. На даній сторінці можна побачити список всіх створених користувачем календарних планувань. Кожен елемент списку представляє собою картку, у якій описано характеристики розкладу, такі як кількість рівнів, приладів, задач та операцій, а також тип розкладу.

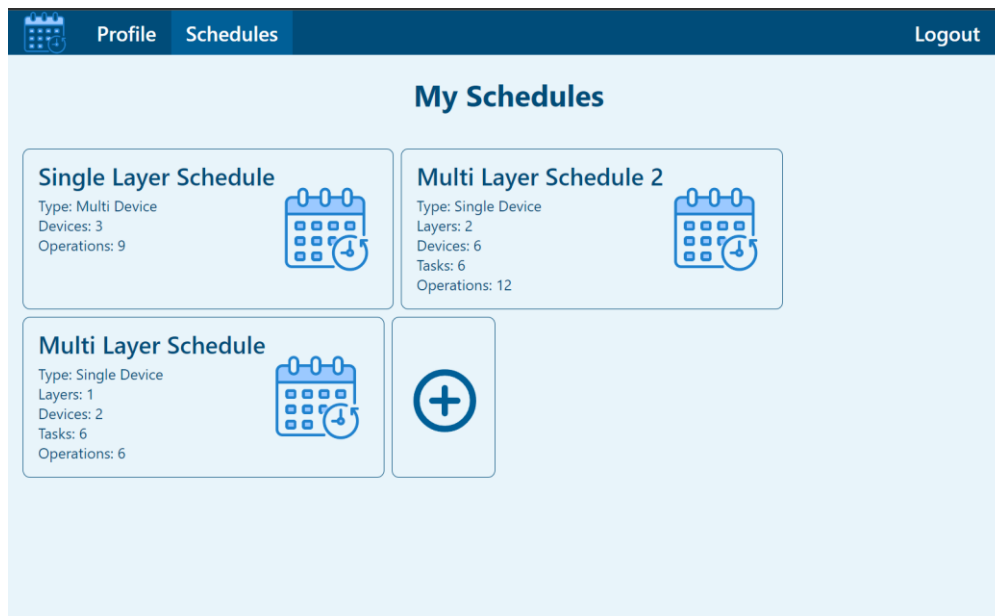


Рисунок 3.9 – Сторінка перегляду створених розкладів

Для створення нового розкладу користувач повинен натиснути на кнопку “+”. Після цього для користувача відкривається модальне вікно, де можна обрати тип нового розкладу та його ім’я.

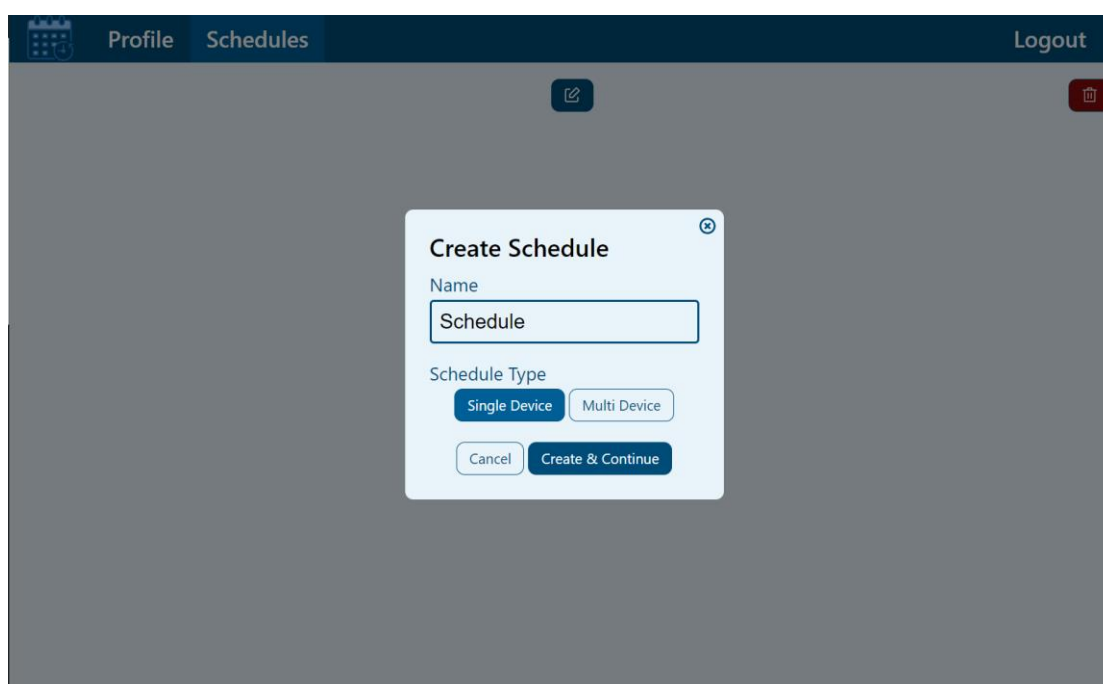


Рисунок 3.10 – Форма створення нового розкладу

Після заповнення всіх необхідних даних користувач повинен натиснути кнопку “Create & Continue”, після чого користувач потрапляє на сторінку редагування частин розкладу.

Якщо новий розклад є багаторівневого типу, то перед користувачем відкриється сторінка редагування багаторівневого розкладу

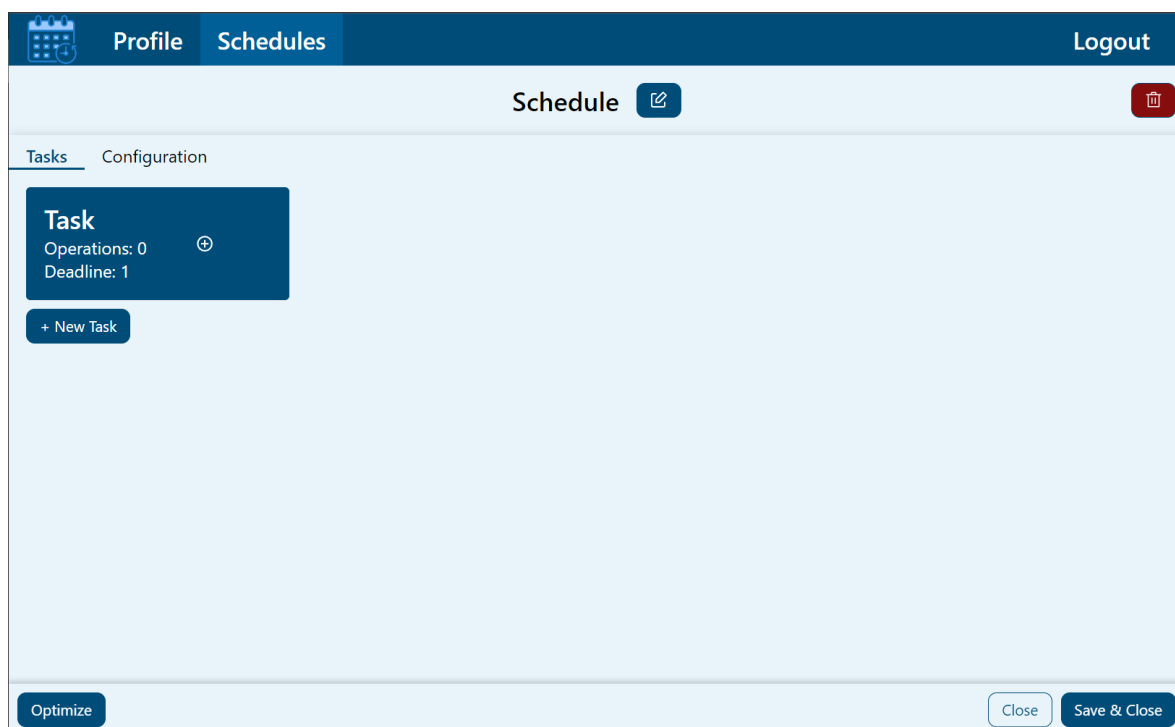


Рисунок 3.11 – Початкова сторінка редагування багаторівневого розкладу

Для редагування назви розкладу потрібно натиснути кнопку у вигляді позначки олівця біля імені розкладу. У результаті даної дії у верхній частині сторінки з'явиться поле для вводу нової назви розкладу. Після редагування назви потрібно натиснути кнопку “Save” для збереження змін.

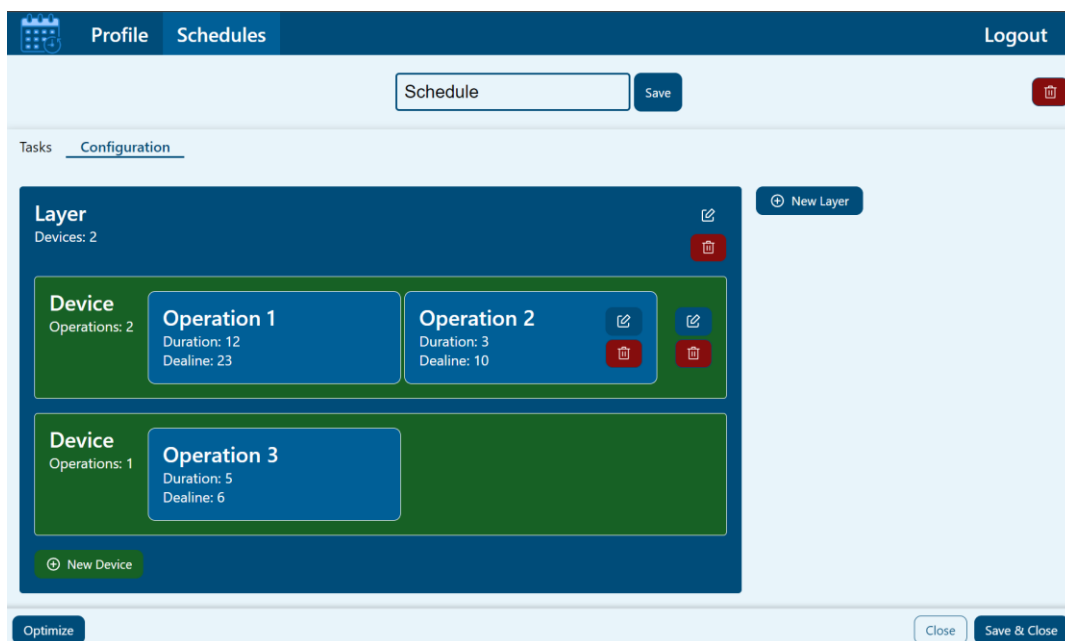


Рисунок 3.12 – Редагування назви розкладу

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІП-9213.045490.05.34

Арк.

11

Сторінка редагування багаторівневого розкладу складається з двох вкладок – вкладки задач та вкладки конфігурації розкладу. У вкладці задач користувач може створювати, редагувати такі частини розкладу, як задачі та операції. Для створення нової задачі користувач повинен натиснути на кнопку “+ New Task”. Після її натискання у розкладі з’являється нова задача у вигляді за замовчуванням.

Для редагування задачі потрібно натиснути на кнопку у вигляді позначки олівця у картці обраної задачі. Після цього користувачеві відкриється модальне вікно, де можна ввести нові характеристики задачі, такі як її назву та директивний термін.

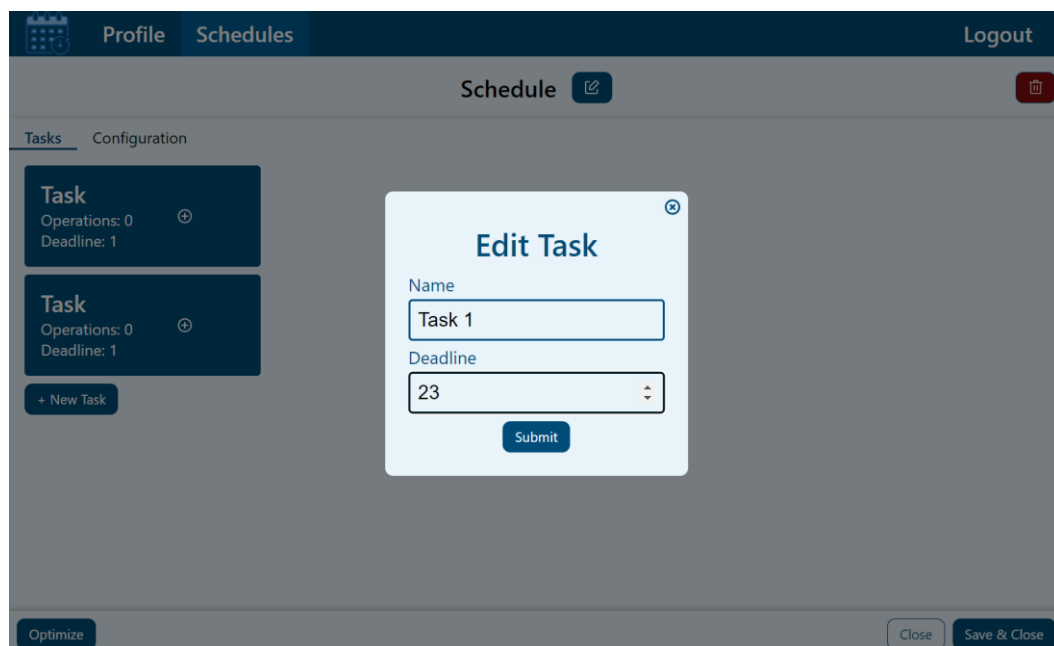


Рисунок 3.13 – Модальне вікно редагування задачі

Після введення всіх необхідних даних користувач повинен натиснути кнопку “Submit”. У результаті даної дії користувач повертається на сторінку редагування розкладу.

Для видалення задачі користувачеві потрібно натиснути кнопку у вигляді позначки корзини. Після даної дії у розкладі зникне обрана задача.

Для додання нової операції до задачі достатньо натиснути кнопку “+” у обраній задачі. В результаті цього у задачі з’явиться нова операція у вигляді за замовчуванням.

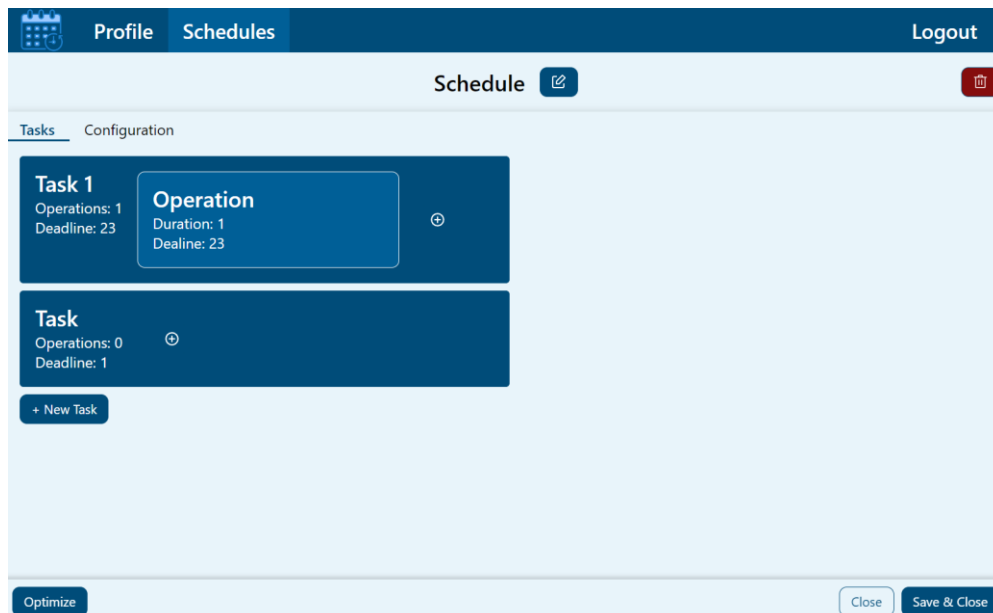


Рисунок 3.14 – Сторінка редагування розкладу після додавання нової операції

Якщо користувач хоче змінити характеристики операції, для цього потрібно натиснути кнопку у вигляді позначки олівця у картці обраної операції. Після цього на екрані з'явиться модальне вікно для редагування параметрів операції, таких як назва операції та її тривалість.

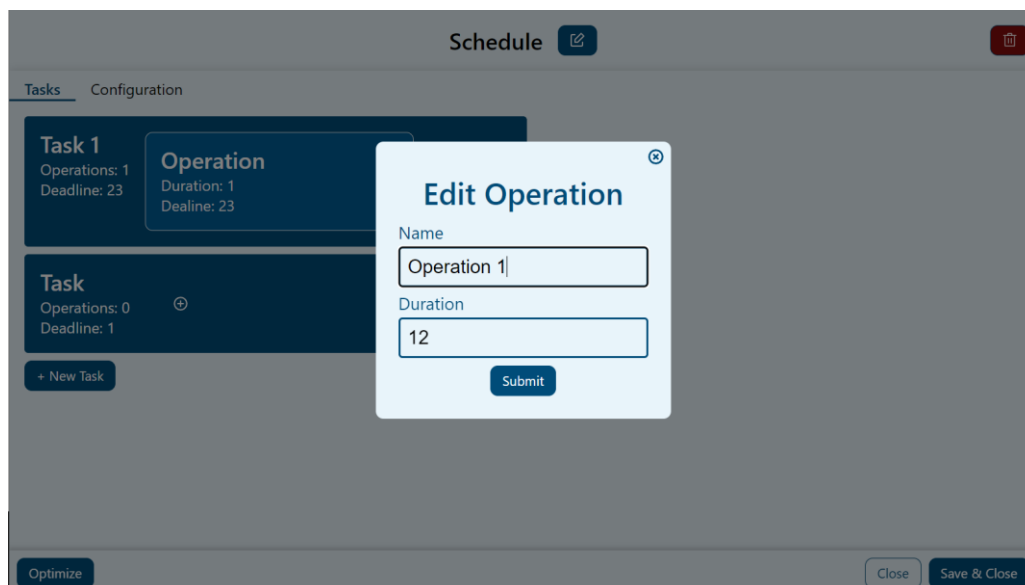


Рисунок 3.15 – Форма редагування операції

Після заповнення необхідних полів вводу користувач повинен натиснути кнопку “Submit”, в результаті чого користувач повертається на сторінку редагування розкладу.

У разі необхідності видалення операції користувач повинен натиснути на кнопку у вигляді корзини у картці обраної операції, після чого дана операція зникає з задачі.

Після створення всіх необхідних задач та операцій, користувач переходить на другу вкладку “Configuration”. У даній вкладці користувач може створювати рівні розкладу, їх пристрої та призначати операції створеним приладам.

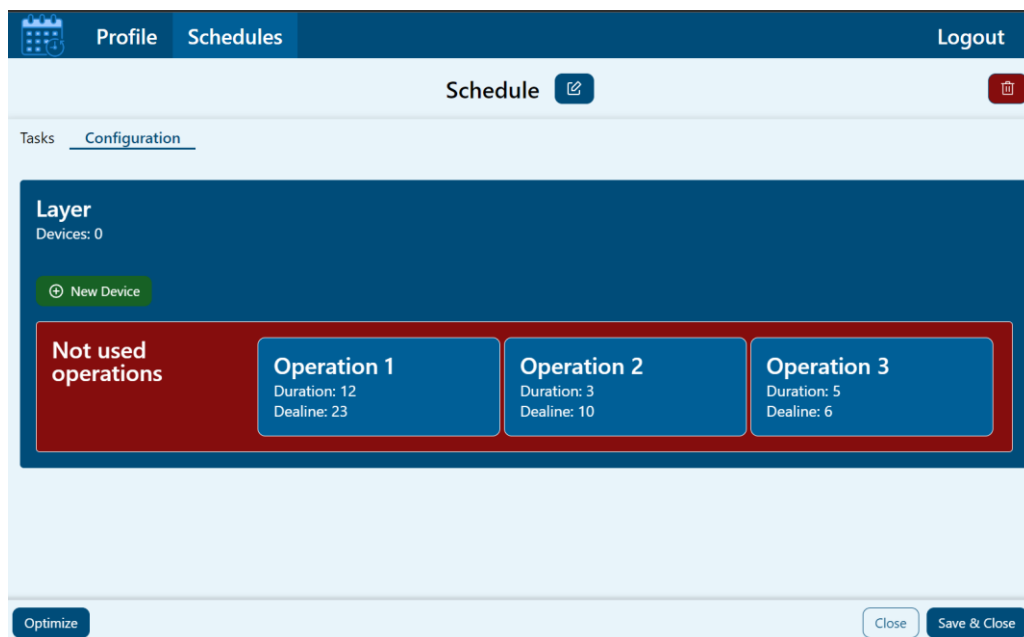


Рисунок 3.16 – Вкладка конфігурації сторінки редагування багаторівневого розкладу

У даній вкладці відображається горизонтальний список всіх рівнів розкладу. Кожен розклад має вертикальний список приладів, створених у ньому. Прилади мають в собі горизонтальні списки операцій, призначених на них, у вигляді карток.

Для створення нового рівня розкладу потрібно натиснути на кнопку “+ New Layer”. У результаті даної дії у розкладі з’явиться новий рівень у кінці їх списку.

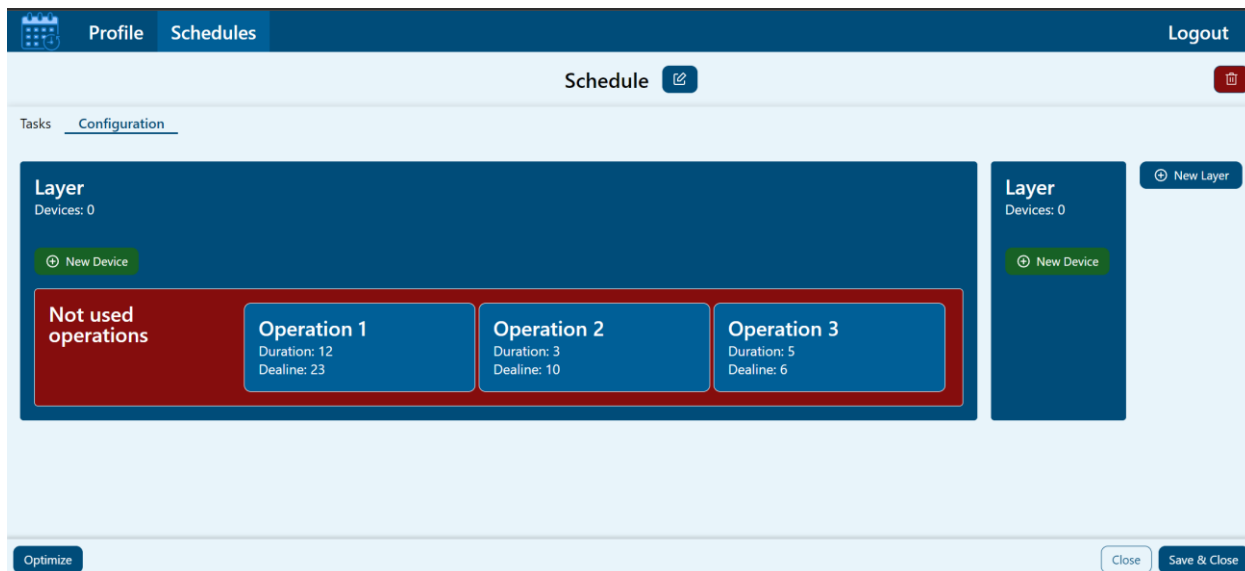


Рисунок 3.17 – Сторінка редагування багаторівневого розкладу після створення нового розкладу

Для редагування параметрів розкладу потрібно натиснути на кнопку у вигляді олівця біля обраного рівня, після чого на сторінці з’явиться модальне вікно, де користувач може ввести нову назву рівня. Після її введення користувач повинен натиснути на кнопку “Submit”, у результаті чого користувач повернеться на сторінку редагування розкладу.

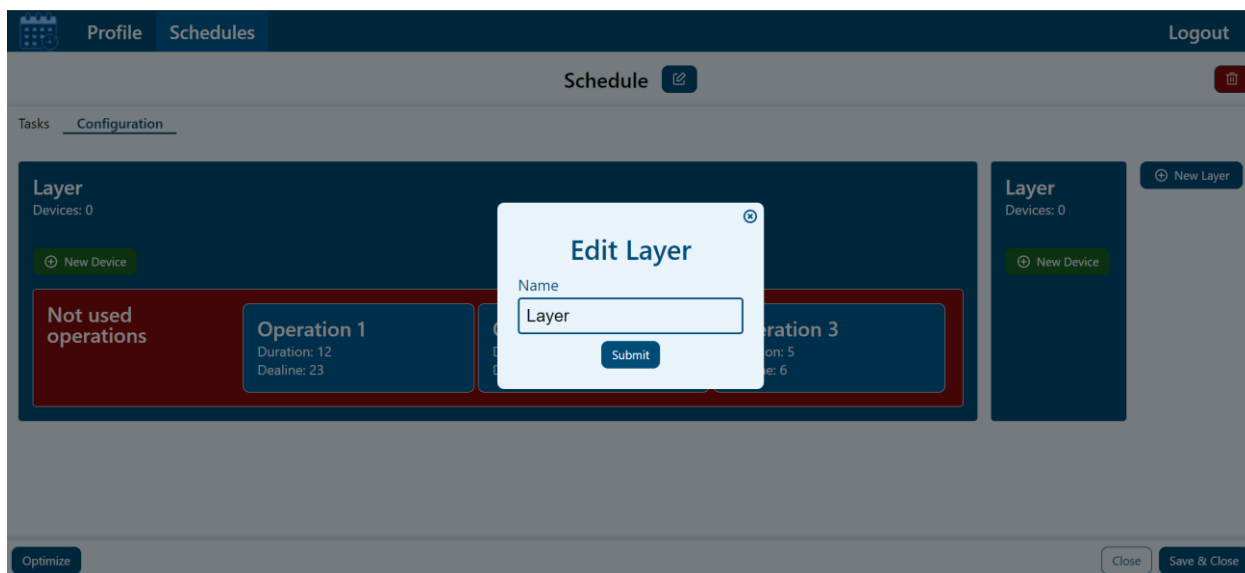


Рисунок 3.18 – Модальне вікно з формою редагування рівня розкладу

Для видалення рівня розкладу потрібно натиснути на кнопку у вигляді корзини поруч з обраним рівнем, у результаті чого даний рівень зникне з відображення розкладу.

Для того, щоб створити новий прилад у рівні розкладу, потрібно натиснути на кнопку “+ New Device” у обраному рівні, після чого у даний рівень буде додано новий прилад за замовчуванням.

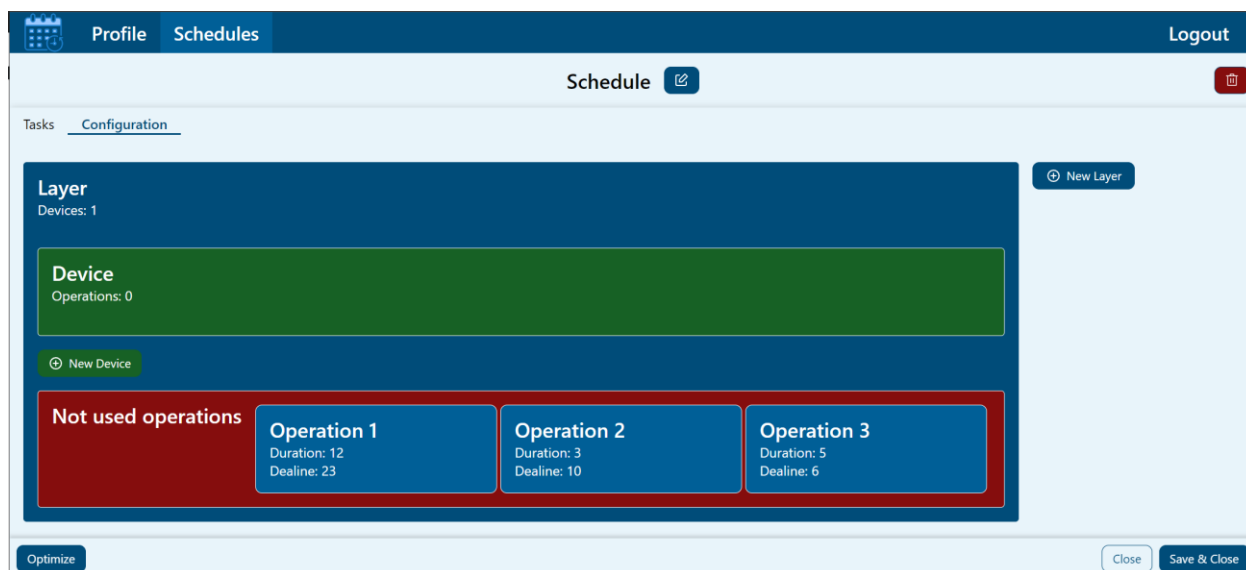


Рисунок 3.19 – Стан форми редагування багаторівневого розкладу після додання нового приладу

Якщо користувачеві потрібно змінити параметри приладу, потрібно натиснути на кнопку у вигляді позначки олівця у картці обраного приладу, у результаті чого з’явиться модальне вікно з формою редагування назви приладу.

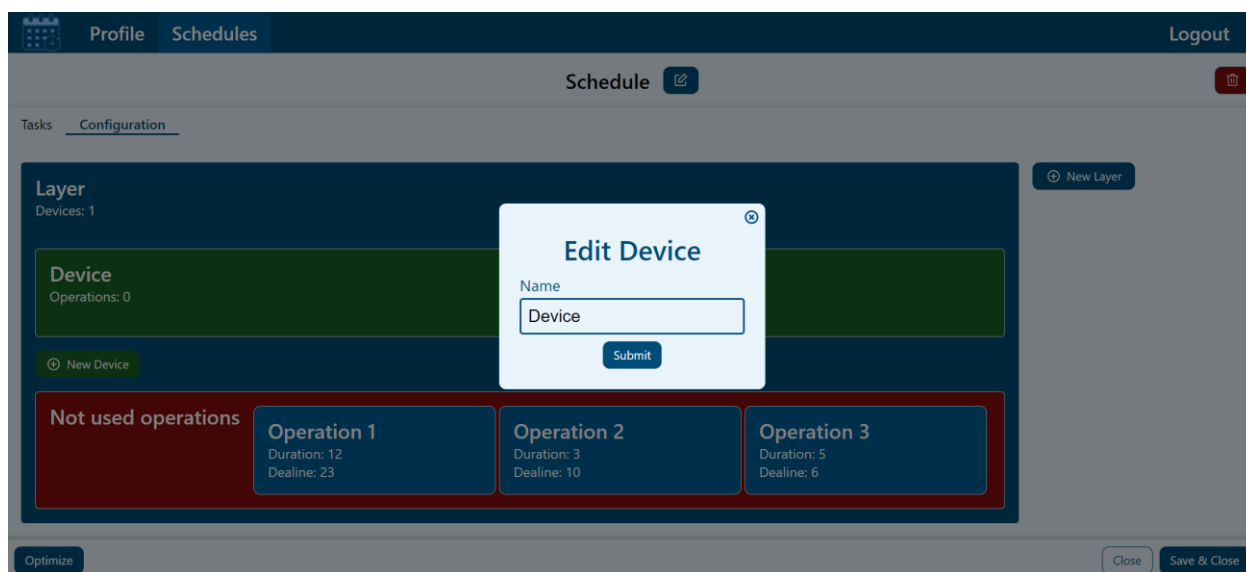


Рисунок 3.20 – Форма редагування імені приладу

Після введення нового імені приладу потрібно натиснути кнопку “Submit”, після чого користувача буде повернений на сторінку редагування багаторівневого розкладу.

Для видалення приладу користувач повинен натиснути на кнопку у вигляді позначки корзини у картці приладу. Після цього обраний прилад зникне зі сторінки редагування.

Для призначення операції приладу потрібно за допомогою миші перенести операцію з блоку “Not used operations” до обраного приладу. Після цієї дії операція буде відображатись у пристрої.

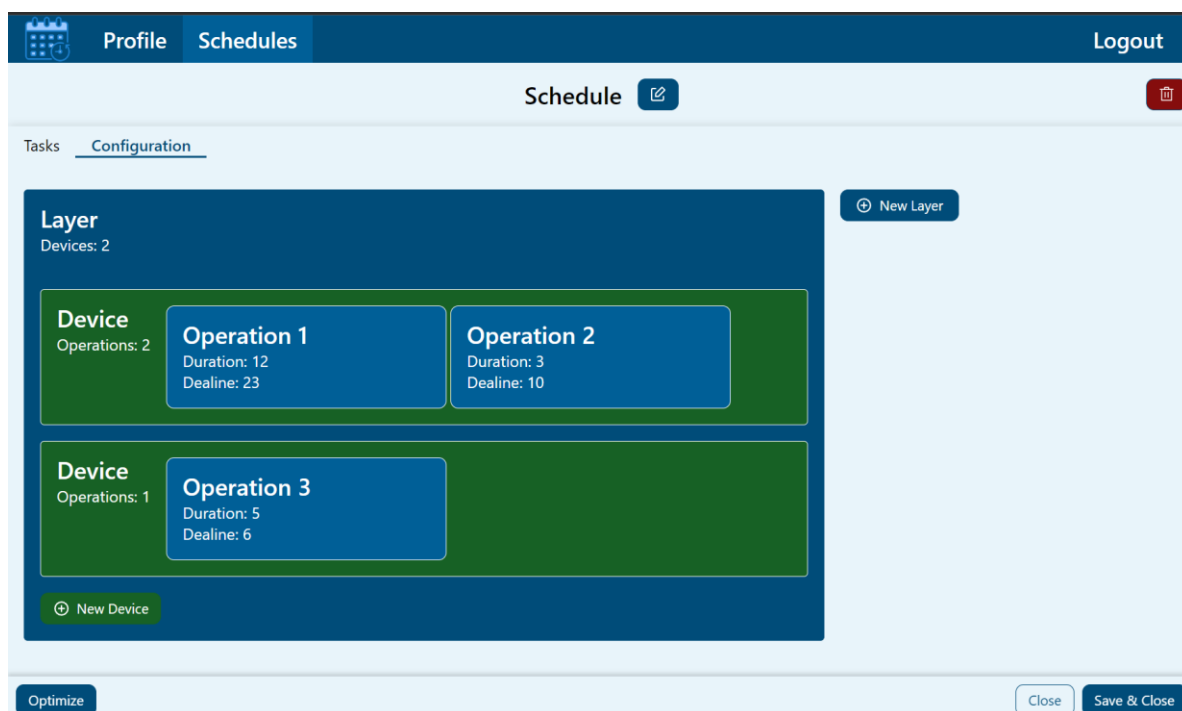


Рисунок 3.21 – Стан сторінки редагування багаторівневого розкладу після призначення всіх операцій приладам

Для оптимізації отриманого розкладу користувач повинен натиснути кнопку “Optimize” у нижньому меню. Після цього з’явиться модальне вікно для вибору алгоритму оптимізації. У результаті його вибору на сторінці з’являється оптимізований розклад, у якого всі операції мають такі характеристики, як моменти початку та кінця роботи, а також час затримки та час випередження роботи.

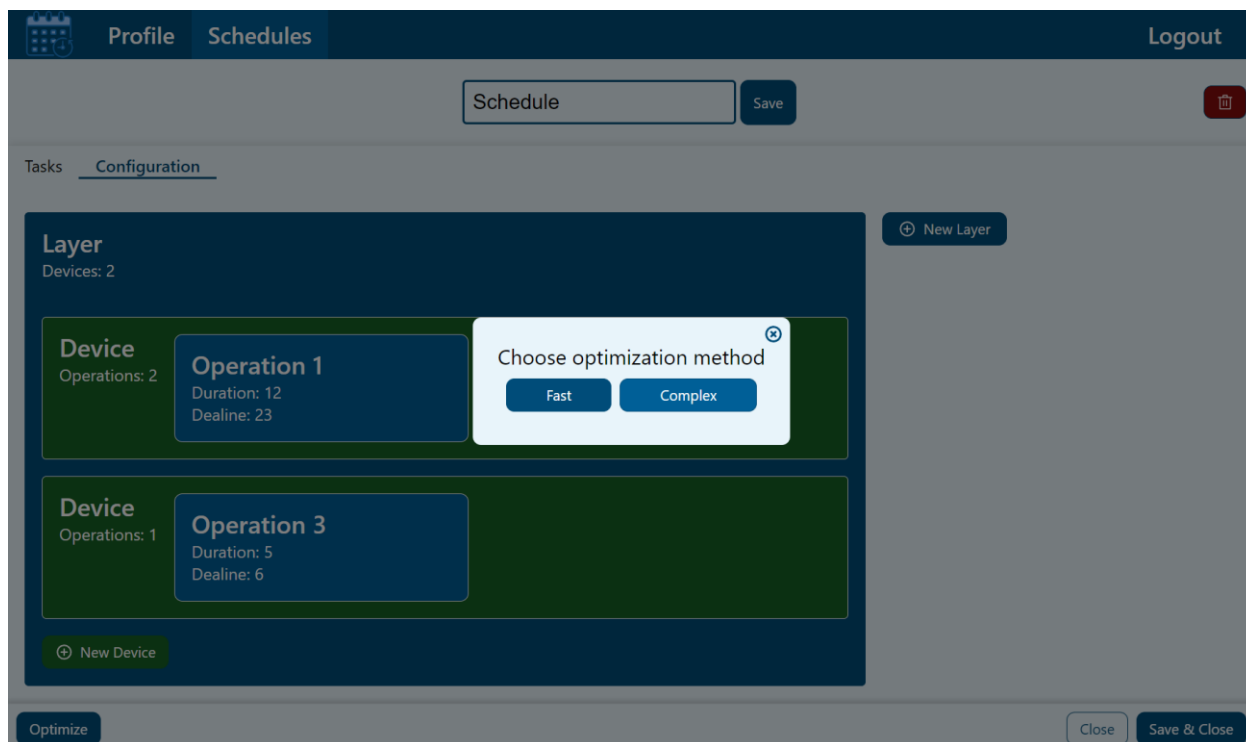


Рисунок 3.22 – Вибір методу оптимізації багаторівневого розкладу

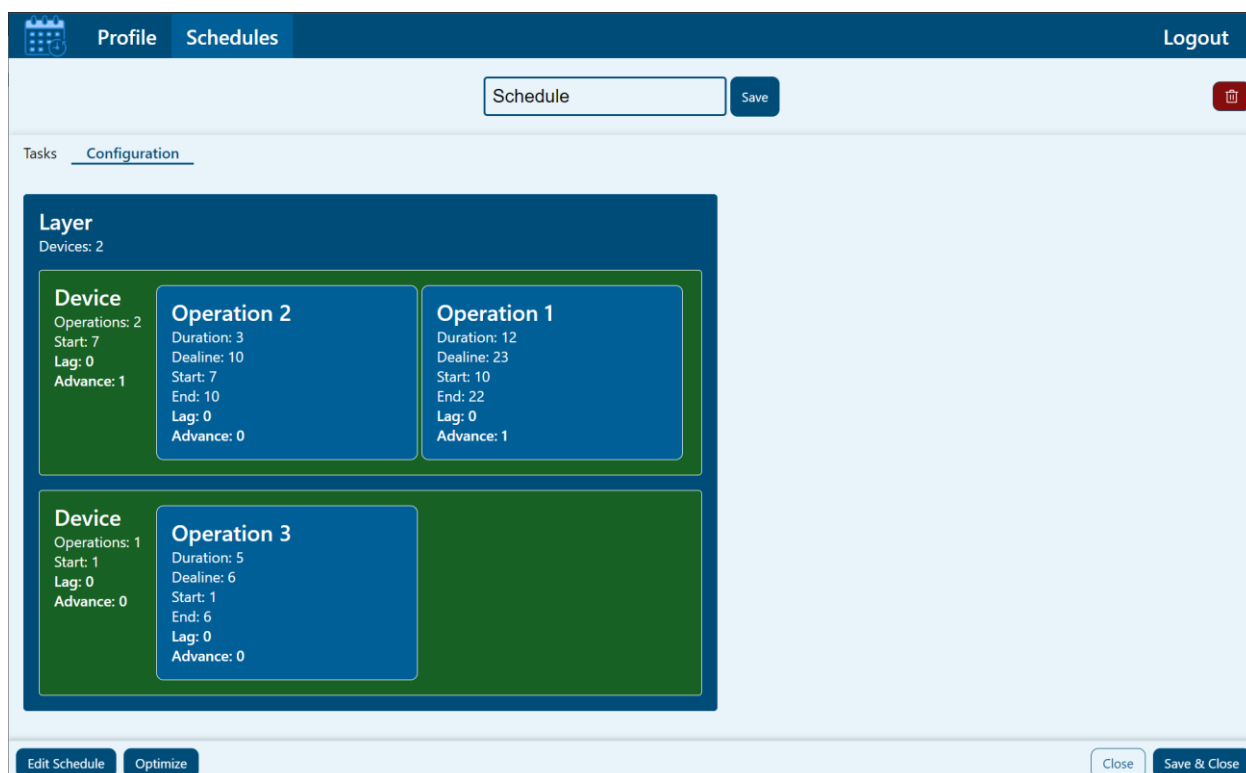


Рисунок 3.23 – Стан сторінки після оптимізації багаторівневого розкладу

Якщо користувачу потрібно зберегти створений розклад, потрібно натиснути кнопку “Save & Close” у нижньому меню. Після цього користувач потрапляє на сторінку перегляду створених розкладів.

У разі створення однорівневого розкладу, користувач потрапить на сторінку редагування саме однорівневого розкладу.

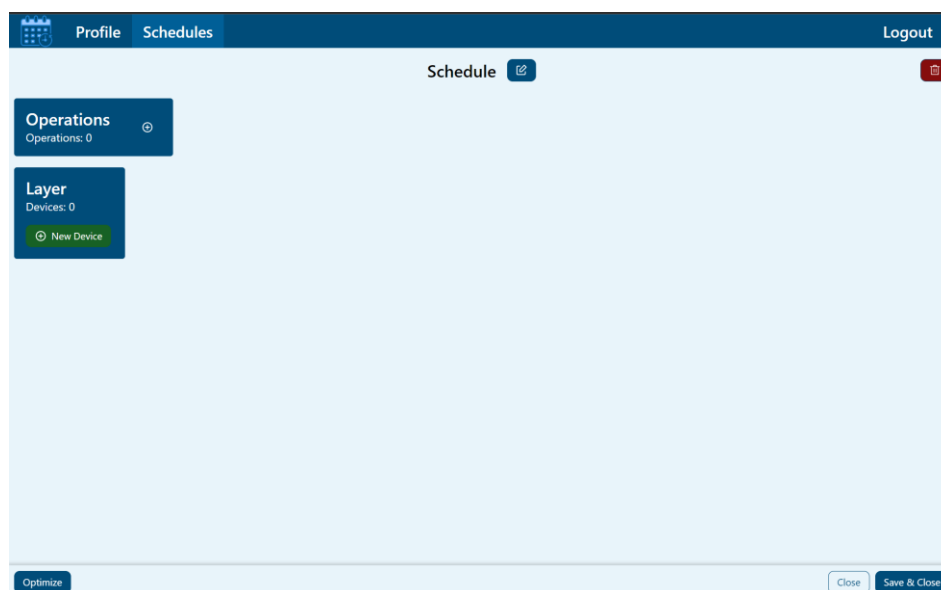


Рисунок 3.24 – Сторінка редагування однорівневого розкладу

Процеси редагування імені однорівневого розкладу, а також створення, редагування та видалення операцій та приладів відбуваються аналогічно до процесів на сторінці редагування багаторівневого розкладу. Єдиною відмінністю є те, що при редагуванні приладу додається нове поле для вводу моменту початку роботи приладу.

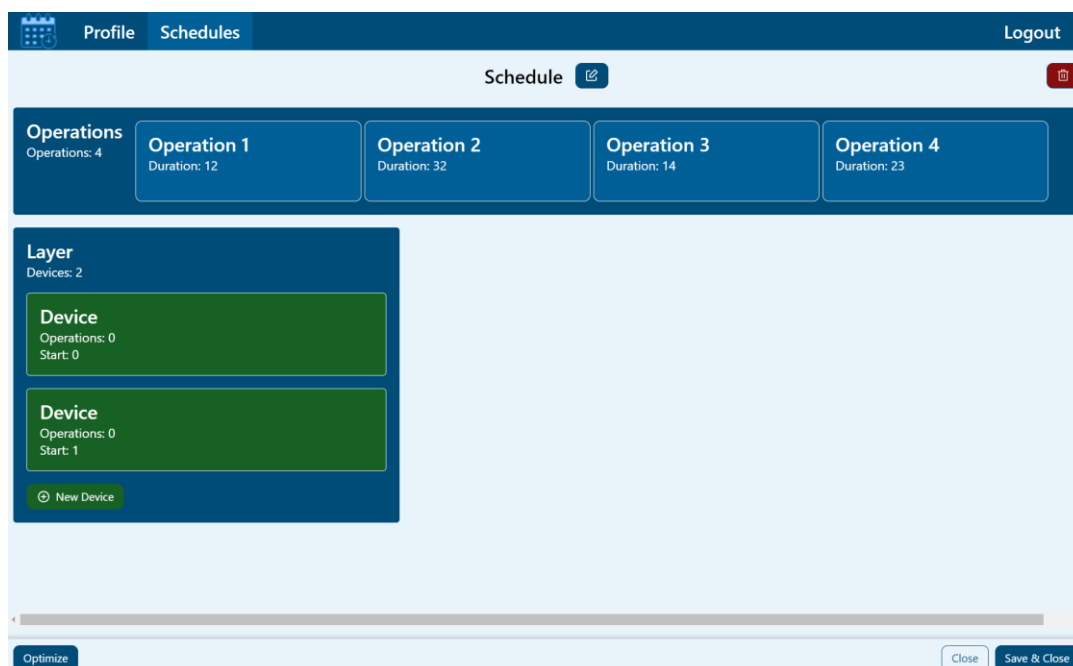


Рисунок 3.25 – Сторінка редагування однорівневого розкладу після створення всіх необхідних операцій та приладів

Для оптимізації однорівневого розкладу потрібно натиснути кнопку “Optimize” у нижньому меню. У результаті даної дії на сторінці буде відображено оптимізований розклад з розподіленими операціями між приладами.

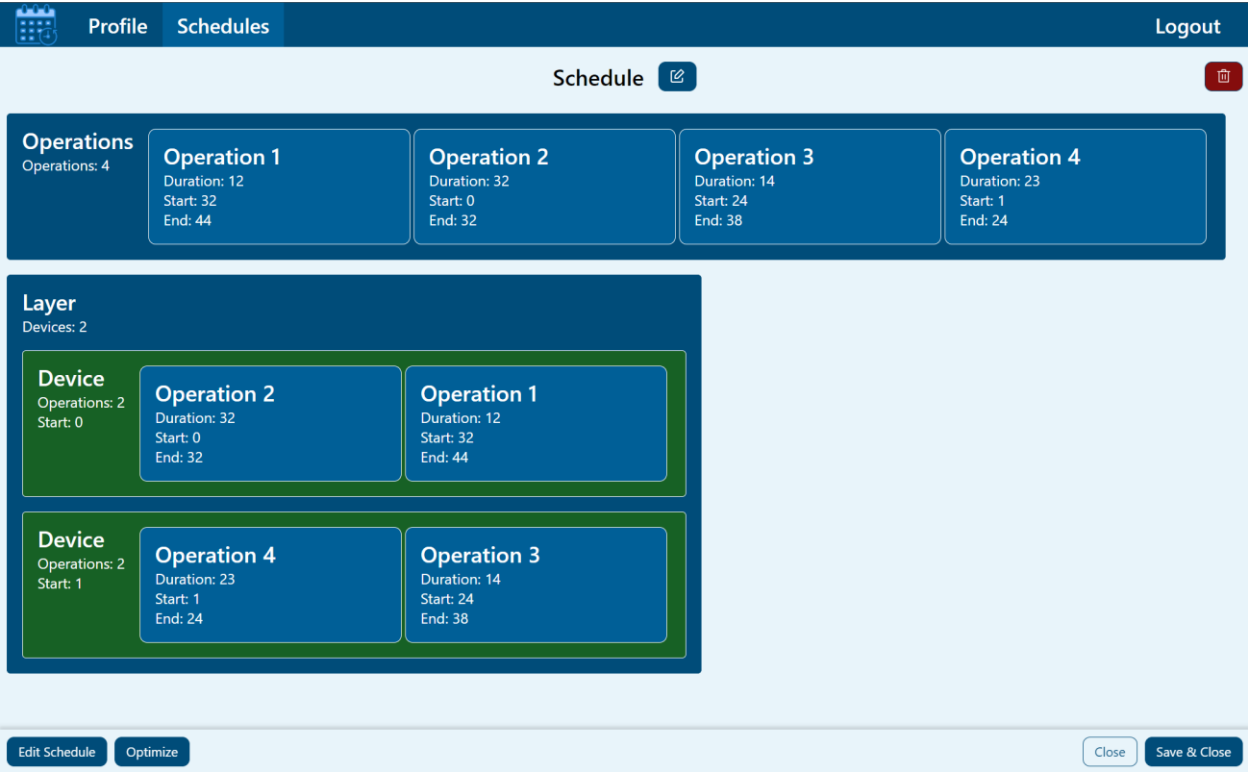


Рисунок 3.26 – Сторінка редагування однорівневого розкладу після проведення оптимізації з автоматичним розподілом операцій між приладами

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

**Програмне забезпечення для календарного планування виконання
ІТ-проектів**

Графічний матеріал

КПІ. ІІ-9213.045490.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена ХАЛУС

Нормоконтроль:

_____ Катерина ЛІЩУК

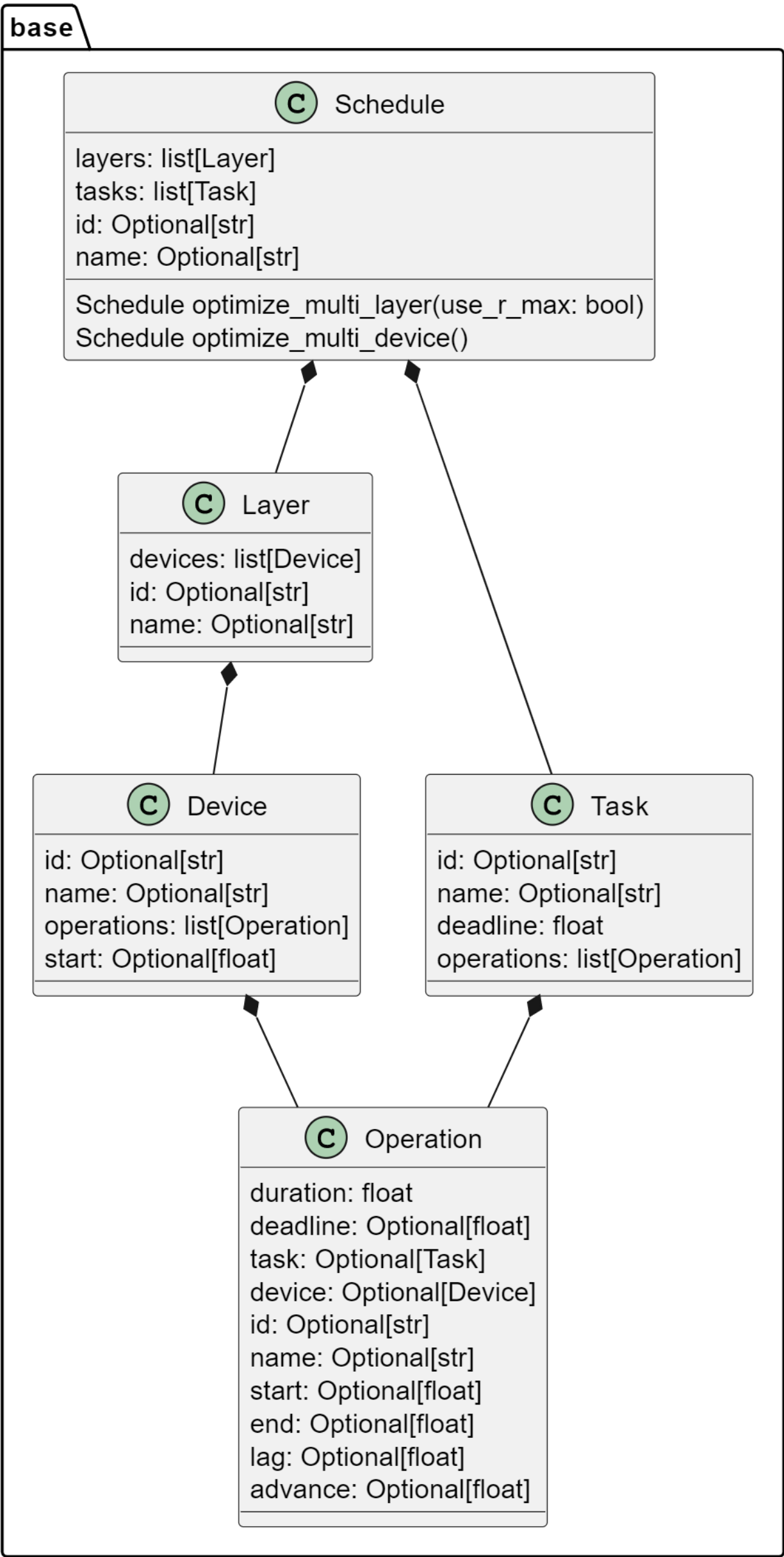
Виконавець:

_____ Михайло МЕДВЕДСВ

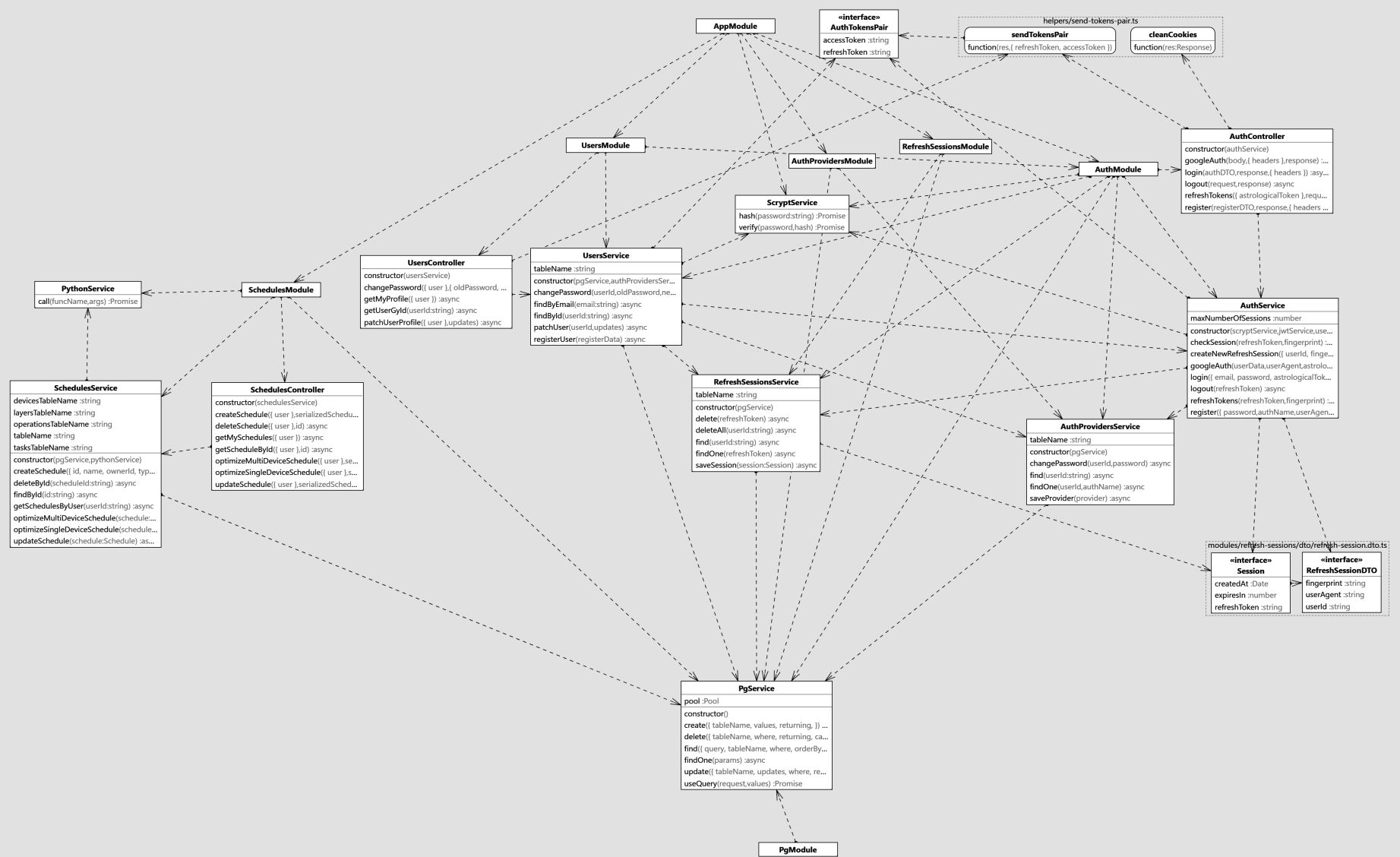
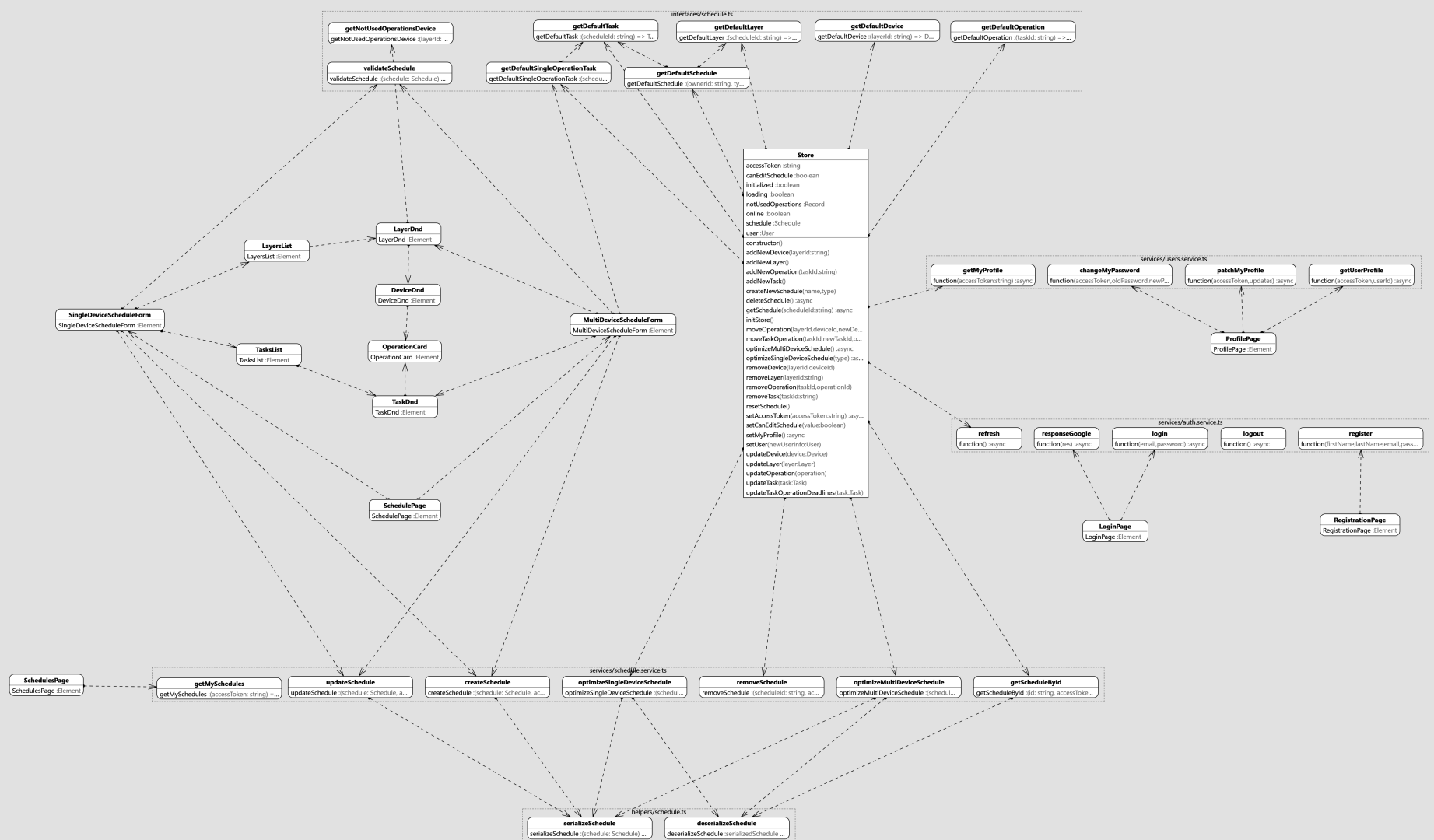
Київ – 2023



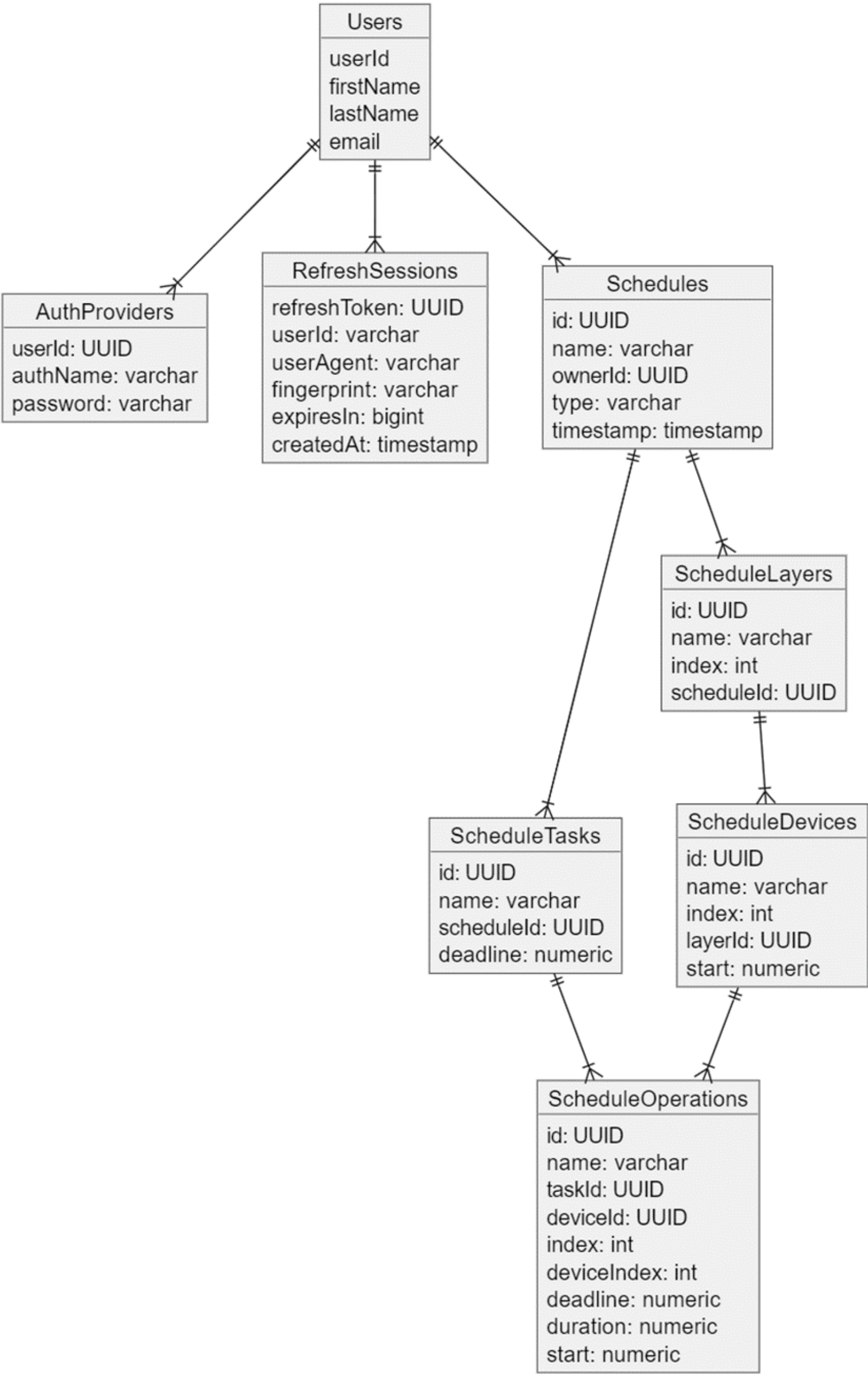
					КПІ.ІП-9213.045490.06.99.CCB					
					Схема структурна варіантів використань	Лит.		Арк.	Аркушів	
Зм.	Арк.	№ докум.	Підп.	Дата						1
Розроб.		Медведев М.Є.								
Перев.		Халус О.А.								
Т. Кон.										
					Програмне забезпечення для ефективного календарного планування виконання ІТ-проектів	Аркуш		Аркушів		
Н. Кон.		Піщук К.І.				КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-92				
Затв.		Жаріков Е.В.								



					КПІ.ІП-9213.045490.06.99.ССК					
					Схема структурна класів програмного забезпечення					
Зм.	Арк.	№ докум.	Підп.	Дата				Лит.	Арк.	Аркушів
Розроб.		Медведев М.Є.								1
Перев.		Халус О.А.								
Т. Кон.								Аркуш		Аркушів
					Програмне забезпечення для ефективного календарного планування виконання ІТ-проектів					
Н. Кон.		Піщук К.І.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-92					
Затв.		Жаріков Е.В.								



					КПІ.ІП-9213.045490.06.99.CCM				
					Схема структурна компонентів програмного забезпечення	Лит.		Арк.	Аркуші
Зм.	Арк.	№ докум.	Підп.	Дата					1
Розроб.		Медведев М.Є.							
Перев.		Халус О.А.							
Т. Кон.						Аркуш		Аркуші	
					Програмне забезпечення для ефективного календарного планування виконання ІТ-проектів	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-92			
Н. Кон.		Ліщук К.І.							
Зам.		Жаріков Е.В.							



					КПІ.ІП-9213.045490.06.99.СБД						
					Схема бази даних	Лит.			Арк.	Аркушів	
Зм.	Арк.	№ докум.	Підп.	Дата						1	
Розроб.		Медведев М.Є.									
Перев.		Халус О.А.									
Т. Кон.											
					Програмне забезпечення для ефективного календарного планування виконання ІТ-проектів	Аркуш			Аркушів		
Н. Кон.		Піщук К.І.				КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-92					
Затв.		Жаріков Е.В.									