

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ФОРМУВАННЯ
РОЗКЛАДУ НАВЧАЛЬНИХ ЗАНЯТЬ
SOFTWARE IMPLEMENTATION OF THE METHODS
FOR CLASS TIMETABLING

Виконав: студент 4 курсу, групи КНТ-227

Спеціальності 122 Комп'ютерні
науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

Медведєв С.Р.

(прізвище та ініціали)

Керівник Олійник А. О.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20__ року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Медведєва Сергія Романовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація методів формування розкладу навчальних занять. Software Implementation of the Methods for Class Timetabling.

керівник проєкту (роботи) Олійник Андрій Олександрович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 30 ” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 17 травня 2021 року
 3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Олійник А.О., доцент		
Нормоконтроль	Андрєєв М. О., асистент		

7. Дата видачі завдання “05 ” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	1 тиждень	
2	Розробка та затвердження технічного завдання	2 тиждень	Технічне завдання
3	Проектування структури системи	3 тиждень	Структурна схема
4	Розробка схеми функціонування системи та модулів програми	4 тиждень	Функціональна схема. Модулі
5	Створення програмного коду системи	5-6 тижні	Текст програми
6	Тестування та відлагодження програми	7 тиждень	Працездатна система
7	Розробка розділів пояснювальної записки	8 тиждень	ПЗ
8	Розробка слайдів презентації	9 тиждень	Слайди презентації
9	Захист роботи	10 тиждень	

Студент

_____ Медведєв С.Р.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ Олійник А.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 132 с., 47 рис., 13 табл., 4 дод., 27 джерел.

СИСТЕМА ФОРМУВАННЯ РОЗКЛАДУ, МЕТОД РОЗФАРБУВАННЯ ГРАФА, ГЕНЕТИЧНИЙ АЛГОРИТМ, МЕТОД ГІЛОК ТА МЕЖ, МЕТОД ІМІТАЦІЇ ВІДПАЛУ, МУРАШИННИЙ АЛГОРИТМ, JAVASCRIPT, ВЕБ-ЗАСТОСУНОК.

Об'єкт дослідження – системи формування розкладу.

Предмет дослідження – методи формування розкладу навчальних занять.

Мета роботи – програмна реалізація методів формування розкладу навчальних занять.

Актуальність та проблематика роботи полягає в тому, що в наш час дуже важко формувати оптимальний розклад занять у навчальних закладах з-за потреби великих затрат ресурсів та часу. Тому, з'явилася потрібність у виборі методу для формування оптимального розкладу.

Матеріали, методи та технічні засоби: мова програмування JavaScript, середовище розробки Visual Studio Code, платформа Node.js, бібліотеки React та Redux, персональний комп'ютер з процесором Intel Core i5 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено веб-застосунок у вигляді односторінкового додатку (SPA – Single Page Application), який формує оптимальний розклад занять та виводить його на сторінку для перегляду.

Висновки. Розроблено веб-застосунок для формування оптимального розкладу занять.

Галузь використання – формування розкладу занять у вищих навчальних закладах

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 132 pages, 47 figures, 13 tables, 4 appendixes, 27 sources.

SCHEDULE FORMATION SYSTEM, GRAPH COLORING METHOD, GENETIC ALGORITHM, BRANCH AND BOUNDARY METHOD, ANNEAL SIMULATION METHOD, ANT ALGORITHM, JAVASCRIPT, WEB APPLICATION.

The object of research is the scheduling systems.

The subject of research – methods for class timetabling.

The purpose of the work is the software implementation for the methods for class timetabling.

The relevance and problems of the work is that nowadays it is very difficult to form an optimal schedule of classes in educational institutions due to the need for large expenditures of resources and time. Therefore, there is a need to choose a method to form the optimal schedule formation.

Materials, methods and technical means: JavaScript programming language, Visual Studio Code development environment, Node.js platform, React and Redux libraries, personal computer with Intel Core i5 processor running Microsoft Windows 10 operating system.

Results. A web application has been created in the form of a single-page application (SPA), which forms the optimal schedule of classes and displays it on a page for viewing.

Conclusions. A web application has been developed to create an optimal schedule of classes.

Field of use – the schedule formation of classes in higher education.

ЗМІСТ

	С.
Перелік скорочень та умовних позачень	7
Вступ.....	8
1 Аналіз предметної області	9
1.1 Аналіз аналогічних програмних засобів	9
1.2 Огляд існуючих методів	16
1.3 Постановка завдання до роботи	25
1.4 Висновки за розділом 1.....	27
2 Вибір та обґрунтування структури системи, яка проєктується	28
2.1 Вибір інструментарію розробки програмного продукту	28
2.2 Структурна схема програмного продукту	37
2.3 Висновки за розділом 2.....	43
3. Основні рішення щодо реалізації компонентів системи.....	45
3.1 Обробка даних.....	45
3.2 Порівняння обраних методів для вирішення завдання формування розкладу.....	46
3.3 Висновки за розділом 3.....	51
4. Керівництво програміста	52
4.1 Призначення й умови використання веб-застосунку.....	52
4.2 Характеристики програмного продукту	52
4.3 Звертання до програми (файли проєкту)	53
4.4 Вхідні і вихідні дані	55
4.5 Повідомлення	55
4.6 Висновки за розділом 4.....	56
5 Керівництво оператора	57
5.1 Призначення й умови виконання веб-застосунку	57
5.2 Виконання програмного продукту	57
5.3 Повідомлення оператору	61
5.4 Висновки за розділом 5.....	61
Висновки.....	62
Перелік джерел посилання	64
Додаток А Технічне завдання.....	67
Додаток Б Опис програмного продукту.....	70
Додаток В Лістинги файлів роботи.....	76
Додаток Г Слайди презентації.....	120

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗАЧЕНЬ

ACID	– Atomicity, Consistency, Isolation, Durability;
API	– Application Programming Interface;
ER	– Entity-Relationship;
GPL	– General Public License;
IDE	– Integrated Development Environment;
JRE	– Java Runtime Environment;
JSON	– JavaScript Object Notation;
JS	– JavaScript;
MIT	– Massachusetts Institute of Technology;
SPA	– Single Page Application;
БД	– база даних;
ГА	– генетичний алгоритм
ОС	– операційна система;
ПП	– програмний продукт;
СУБД	– система управління базами даних;

ВСТУП

В наш час, проблема складання розкладу занять в навчальних закладах, від шкіл до університетів, має величезну важливість. Без оптимального розкладу неможливо організувати навчання таким чином, щоб викладачам і учням було комфортно і зручно. Складання розкладу вручну займає дуже багато часу та сил, і рідко підсумковий результат буде задовольняти вимогам і бажанням обох сторін. Звідси і виникає потреба автоматизації процесу складання розкладу.

Для складання розкладу потрібно враховувати безліч змінних (об'єктів), таких як: кількість навчальних днів, список викладачів, список предметів, список груп тощо. Завдяки зв'язкам об'єктів один з одним, можна отримати можливість створити програму з прозорою структурою і мінімальним шансом помилки через людський фактор.

Для вирішення цієї проблеми, було виконано дослідження методів складання розкладу, зроблено їх порівняльний аналіз а також обрано метод, який подалі використовується у розробленому веб-застосунку.

Було розроблено веб-застосунок, який призначений для автоматизації процесу складання розкладу у вищих навчальних закладах. Веб-застосунок надає користувачеві можливість до перегляду актуальних розкладів занять для різних груп, а для викладачів – можливість задавати вхідні дані та отримувати сформований оптимальний розклад, який надалі можна редагувати.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз аналогічних програмних засобів

1.1.1. Free Timetabling Software

FET (Free Timetabling Software) – це безкоштовне програмне забезпечення для автоматичного планування розкладу роботи школи, середньої школи чи університету. Він використовує швидкий та ефективний алгоритм розкладу руху. Він ліцензований відповідно до Загальної публічної ліцензії GNU Affero версії 3 або пізнішої [1].

FET було розроблено Лівіу Лалеску та Фолькером Дірром і на сьогоднішній день має актуальну версію FET-5.49.1, яка була випущена 9 березня 2021 року [1]. FET Має власну документацію з посиланнями та відкритим вихідним кодом. Знімок екрану програмного забезпечення наведений на рисунку 1.1.

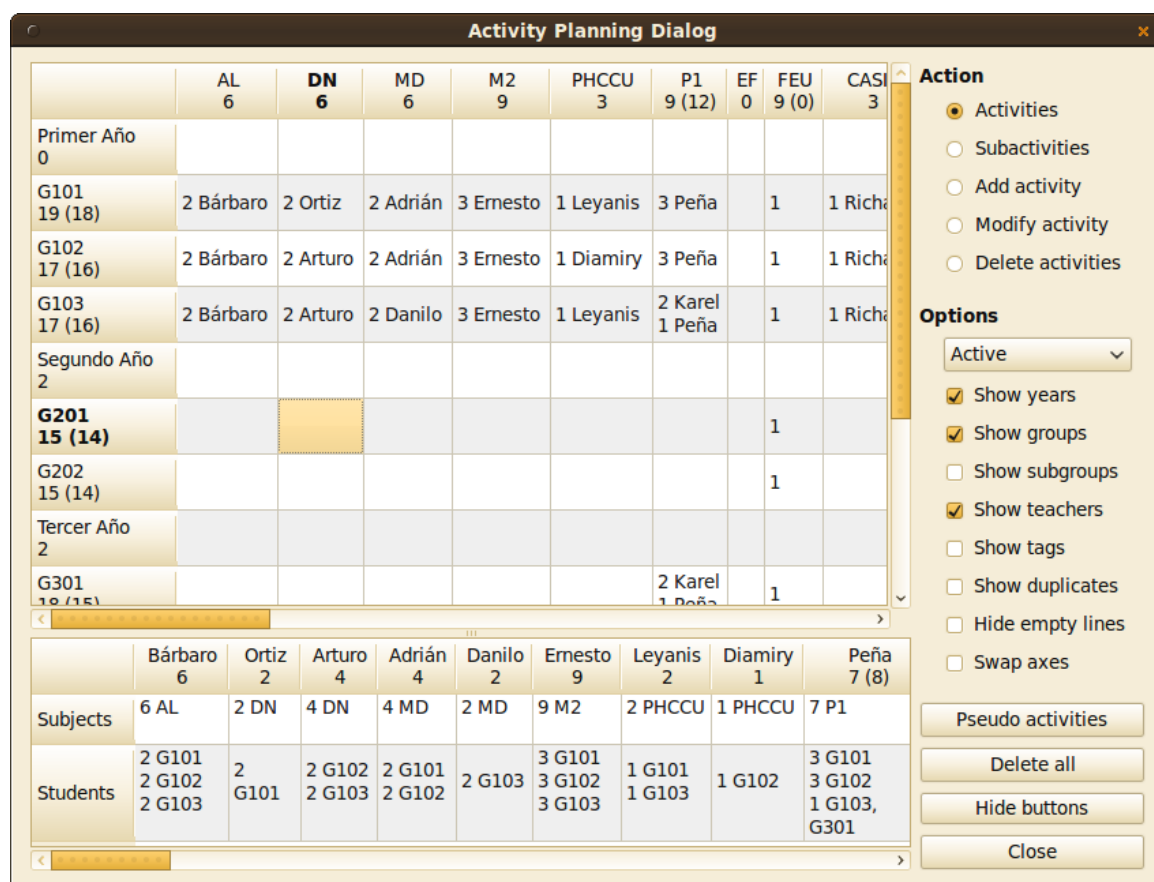


Рисунок 1.1 – Знімок екрану програмного забезпечення FET

Зазвичай FET може вирішити складний графік максимум за 5-20 хвилин. Для більш простих розкладів часу може знадобитися коротший час - менше 5 хвилин (в деяких випадках - секунд). Для надзвичайно складних розкладів це може зайняти більше часу, за лічені години [1].

FET – це програма, яка написана на мові C++ і використовує Qt, набір інструментів C++ для розробки крос-платформних додатків. FET можна запускати на будь-якій платформі, що підтримується Qt (GNU / Linux, Windows, Mac OS X та, можливо, інші) [2].

FET Має можливість імпорту та експорту файлів формату CSV, повністю автоматичний алгоритм генерації, що дозволяє також напівавтоматичний або ручний розподіл; Гнучка структура студентів, організована за наборами: роки, групи та підгрупи. FET дозволяє перекривати роки та групи та неперекриваючі підгрупи [3].

1.1.2 ПЗ «Расписание»

ПЗ «Расписание» – це комерційне програмне забезпечення, яке повністю автоматизує процес формування розкладу занять в різних освітніх установах, враховуючи особливості навчальних планів і організацію навчального процесу [4].

ПЗ «Расписание» було розроблено ТОВ «KostanaySoft» і на сьогоднішній день має актуальну версію 6.0.21, яка була випущена 27 березня 2021 року [4]. «Расписание» має власну документацію, з якою можна ознайомитись на сайті товариства-розробника. Знімок екрану програмного забезпечення наведений на рисунку 1.2.

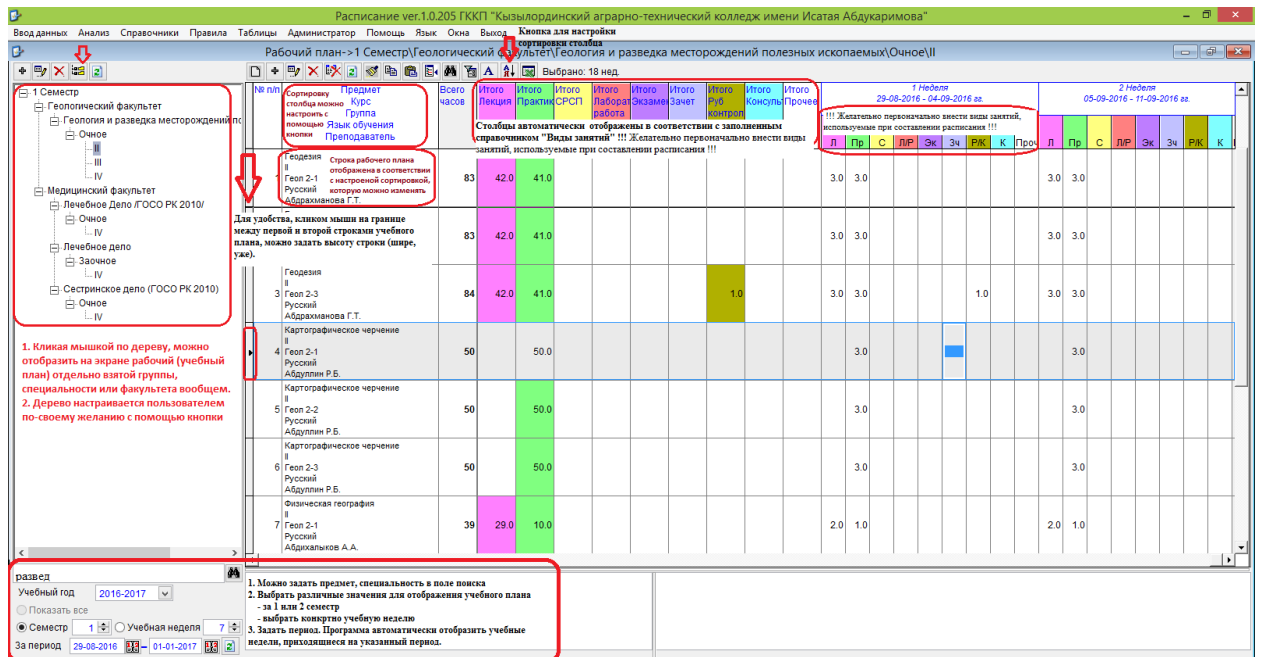


Рисунок 1.2 – Знімок екрану програмного забезпечення «Расписание»

ПЗ «Расписание» має можливість імпорту та експорту файлів у форматі Excel, можливість групування розкладів по спеціальностям факультетів, можливість проводити аналіз зайнятості викладачів тощо. Також, «Расписание» має модуль адміністрування, який забезпечує конфіденційність документів, що потрапляють в систему: реалізовані функції розмежування прав доступу користувачів до різних документів [4].

1.1.3 ПЗ «НИКА-Люкс 7»

Программа составления расписания занятий «НИКА-Люкс 7» - це комерційне програмне забезпечення, яке повністю автоматизує процес формування розкладу занять в освітніх установах різного статусу і профілю (школи, гімназії, коледжі, училища), враховуючи особливості навчальних планів і організації навчального процесу, а також вимоги методичного, санітарно-гігієнічного та індивідуально-психологічного характеру. Програма надзвичайно легка в освоєнні і зручна в роботі. Результатом роботи програми є розклад занять, складений з мінімальною кількістю «вікон» для викладачів [5].

ПЗ «НИКА-Люкс 7» було розроблено компанією «НИКА-Софт» і на сьогоднішній день має актуальну версію 7.6.1 [5]. ПЗ має власну документацію, з якою можна ознайомитись на офіційному сайті розробника. Знімок екрану програмного забезпечення наведений на рисунку 1.3.



Рисунок 1.3 – Знімок екрану програмного забезпечення «НИКА-Люкс 7»

ПЗ «НИКА-Люкс 7» має багато різних можливостей, наприклад: викладання одним учителем двох і більше предметів, імпорт та експорт розкладу в Excel, зберігання в одному файлі декількох розкладів на різні періоди навчального року, експорт розкладу для відображення на сайті школи і на інформаційні кіоски тощо [6].

ПЗ «НИКА-Люкс 7» складається з декілька модулів, які можна отримати окремо від основної програми. Такими модулями є модуль «Заміни», який призначений для коригування щоденного розкладу занять в зв'язку з відсутністю вчителів, та модуль «НИКА-Онлайн», який призначений для миттєвого відображення розкладу і замін на сайті школи або на інформаційному кіоску [6].

1.1.4 ПЗ «1С:Университет» з модулем автоматизованого складання розкладу

ПЗ «1С:Университет» - це комерційне програмне забезпечення, для автоматизування роботи вищих навчальних закладів. В її склад входить модуль автоматизованого складання розкладу. З його допомогою складати розклад можна в автоматичному, ручному і змішаному режимах з урахуванням багатьох обмежень і умов. При цьому можна побудувати як допустимий розклад, так і оптимізоване, в якому скорочено кількість вікон або кількість використовуваних приміщень [7].

ПЗ «1С:Университет» було розроблено компанією «1С» і на сьогоднішній день має актуальну версію 1.1.3 [7]. «1С:Университет» має власну документацію, з якою можна ознайомитись на офіційному сайті розробника. Знімок екрану програмного забезпечення можна побачити на рисунку 1.4.

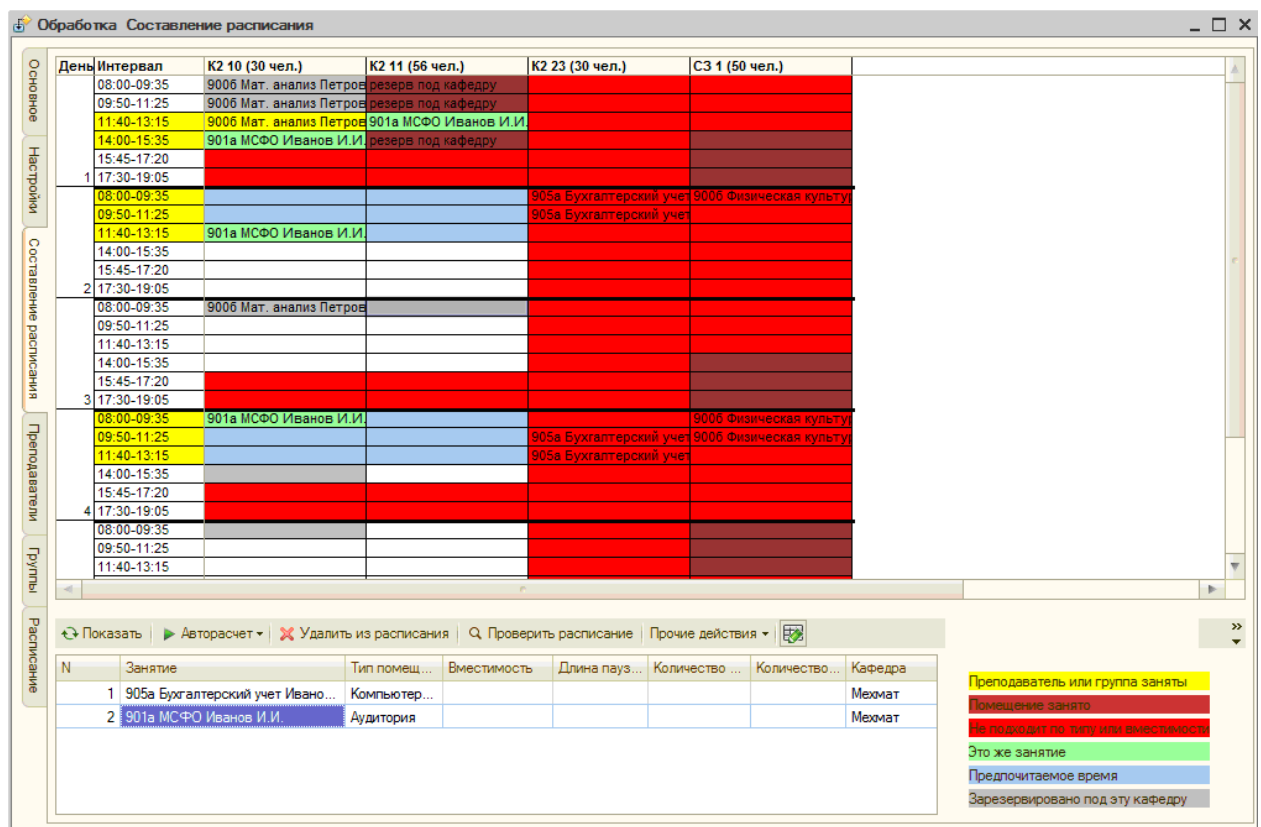


Рисунок 1.4 – Знімок екрану програмного забезпечення «1С:Университет»

У ПЗ «1С:Університет» є багато можливостей, наприклад: зручна форма "шахматка" для швидкої ручної модифікації розкладу перетягуванням "drag & drop", складання кількох розкладів і вибір кращого, врахування побажань і можливостей викладачів, груп студентів, приміщень, імпорт та експорт файлів формату Excel, перегляд розкладів і введення переваг по web-інтерфейсу тощо [7].

Важливо знати, що модуль не можна використовувати без основної платформи «1С». Треба купувати ліцензію на платформу та на її розширення та модулі.

1.1.5 ПЗ «АВТОРасписание»

ПЗ «АВТОРасписание» – комерційне програмне забезпечення, яке призначено для швидкого, зручного і якісного складання розкладів занять і супроводу їх протягом всього навчального року. ПЗ працює з розкладами будь-якого ступеня складності. Розмірність навчального закладу не обмежена. Можна скласти тижневі, двотижневі, семестрові розклади, а також розкладу сесій заочників з плаваючим графіком навчання та розкладу разових заходів [8].

ПЗ «АВТОРасписание» було розроблено компанією «MMIS LAB» і на сьогоднішній день має версію від 13.07.2020 року [8]. ПЗ має власну документацію, яку можна отримати у комплекті з програмою при її покупці. Знімок екрану програмного забезпечення можна побачити на рисунку 1.5.

Таблиця 1.1 – Порівняльна характеристика існуючих програмних засобів

Можливості	FET	«Расписание»	«НИКА»	«АВТОР»	«ІС»
Наявність користувацького інтерфейсу	Так	Так	Так	Так	Так
Кросплатформеність	GNU OS	Так	Так	Так	Так
Наявність зрозумілої та повної документації	Так	Так	Так	Так	Так
Комерційний проект	Ні	Так	Так	Так	Так
Десктопний додаток	Так	Так	Так	Так	Так
Треба встановлювати ПП на кожен комп'ютер	Так	Так	Так	Так	Так
Можливість імпорту/експорту файлів формату Excel	Тільки CSV	Так	Так	Так	Так
Рівень можливостей	+	++	+++	+++	++++

Зробивши аналіз аналогів програмного забезпечення для формування розкладу, я дійшов висновку, що приведені аналоги можуть підійти під розв'язування даної задачі, але для цього треба встановлювати на кожен комп'ютер програмне забезпечення.

Для вирішення задачі формування розкладу, на мою думку, треба розробити веб-застосунок, користуватись яким можна на будь-якому пристрої, у якого є доступ к мережі Інтернет.

1.2 Огляд існуючих методів

В даному розділі розглядаються методи рішення задачі формування розкладу, та обирається найбільш ефективний метод вирішення задачі формування розкладу.

1.2.1 Метод гілок та меж

Метод гілок та меж є загальним алгоритмічним методом вирішення оптимізаційних задач. Він часто використовується для таких NP-повних задач, як задача комівояжера та задача о ранці. Даний метод є модифікацією повного перебору при якій відкидаються підмножини допустимих рішень, що не містять значень цільової функції, які оптимальні [10].

В методі гілок та меж використовуються дві операції: розгалуження та знаходження оцінок (меж) [10].

За допомогою операції розгалуження множина допустимих рішень на кожному етапі розділяється на підмножини меншої розмірності. Ці множини стають вузлами дерева пошуку [10].

Операція знаходження оцінок (меж) визначає верхні та нижні межі оптимального значення на підмножинах допустимих рішень. Якщо для знайденого вузла дерева пошуку верхня межа співпадає з нижньою, то це значення є оптимальним [10].

Головний недолік методу гілок і меж полягає у відсутності ознаки оптимальності і, як наслідок, в необхідності повністю розв'язувати задачі лінійного програмування, пов'язані з кожною вершиною багатогранника допустимих рішень. Оптимальний розв'язок може бути отримано задовго до зупинки алгоритму, але спостерігати це в загальному випадку не можна, оскільки оптимальність встановлюється лише після вичерпання списку завдань. Для задач великої розмірності це вимагає значних і, певною мірою, непотрібних з практичної точки зору витрат часу. Крім того, для певних типів задач проблематичним може бути отримання оцінок для здійснення операції розгалуження [10].

1.2.2 Метод імітації відпалу

Алгоритм імітації відпалу – загальний алгоритмічний метод розв'язання задачі глобальної оптимізації; є одним із прикладів методів Монте-Карло. У процесі пошуку оптимального рішення з деякою ймовірністю дозволяється перехід у стан із більш високим значенням цільової функції. Ця особливість дозволяє вийти з локальних оптимумів. На початку роботи алгоритму ймовірність переходу в стан із більш високим значенням цільової функції повинна бути достатньо велика, щоб був шанс переходу від обраного початкового наближення до будь-якого іншого рішення. У процесі роботи алгоритму ймовірність переходу плавно зменшується [10].

Недоліком методу є висока обчислювальна складність. Це зумовлено тим, що для отримання потрібного рішення треба досить повільне зниження ймовірності перевести в стан із більш високим значенням цільової функції, який призводить до великої кількості ітерацій алгоритму. Алгоритми імітації відпалу можуть використовуватися для побудови розкладів реалізації прикладних програм на всіх стадіях проектування обчислювальних систем реального часу. Однак для практичного застосування методів відпалу актуальною залишається задача амортизації обчислювальної складності алгоритмів [10].

Величезною перевагою метода імітації відпалу є властивість запобігти "пастки" в локальних мінімумах оптимізованої функції, і продовжити пошук глобального мінімуму. Це реалізується за рахунок прийняття не тільки змін параметрів, що стають причиною зменшення значення функції, але і деяких змін, що збільшують її значення, в залежності від т. зв. температури T – характеристики модельованого процесу. Чим вище температура, тим більші "погіршуючі" зміни (аналогічні випадковим флуктуаціям в нагрітій речовині) допустимі, і більше їх ймовірність [11].

Ще одною перевагою є те, що навіть в умовах невеликої кількості обчислювальних ресурсів для знаходження глобального мінімуму, метод імітації відпалу, як правило, видає ненайгірше рішення (один з локальних мінімумів) [11].

1.2.3 Генетичні алгоритми

Генетичний алгоритм є методом, що відображає природну еволюцію методів розв’язування проблем, і в першу чергу задач оптимізації. Генетичні алгоритми — це операції пошуку, засновані на механізмах природного відбору і спадкоємства. У них використовується еволюційний принцип виживання більш за все пристосованих особин. Вони відрізняються від традиційних методів оптимізації декількома базовими елементами [13].

Зокрема, генетичні алгоритми:

- опрацьовують не значення параметрів самого завдання, а їх закодовану форму;
- здійснюють пошук розв’язку виходячи не з єдиної точки, а з їх деякої популяції;
- використовують тільки цільову функцію, а не її похідні або іншу допоміжну інформацію;
- застосовують імовірнісні, а не детерміновані правила вибору [13].

При описанні генетичних алгоритмів використовуються визначення, запозичені з генетики. Наприклад, мова йде про популяцію особин, а в якості базових понять застосовуються ген, хромосома, генотип, фенотип, алель [13].

Популяція — це кінцева множина особин. Особини, що входять в популяцію, у генетичних алгоритмах представляються хромосомами з закодованими в них множинами параметрів задачі, тобто розв’язків, які інакше називаються точками в просторі пошуку [13].

Хромосоми (інші назви — ланцюжки або кодові послідовності) — це впорядковані послідовності генів. Ген (який також називається властивістю,

знаком чи детектором) – це атомарний елемент генотипу, зокрема, хромосоми. Генотип або структура – це набір хромосом даної особи. Отже, особинами популяції можуть бути генотипи або одиничні хромосоми (в широко відомому випадку, коли генотип складається з однієї хромосоми) [13].

Фенотип – це набір значень, які відповідає даному генотипу, тобто декодована структура або безліч параметрів задачі (розв’язок, точка простору пошуку) [13].

Алель – це значення конкретного гена, також визначається як значення властивості або різновид властивості. Локус чи позиція вказує місце розміщення даного гена в хромосомі (ланцюжку). Множина позицій генів – це локи [13].

Генетичні алгоритми (ГА) дозволяють розв’язувати широке коло складних задач багатокритеріальної оптимізації шляхом випадкового підбору, комбінування та зміни параметрів моделювання способами, що аналогічні до біологічної еволюції (наслідування, мутація, відбір). Застосовують ці алгоритми й для рішення задач на графах [10].

Генетичні алгоритми оперують сукупністю особин (популяцією), які кодують одно з можливих рішень задачі. Цим генетичний алгоритм відрізняється від більшості інших алгоритмів оптимізації, які працюють лише з одним рішенням, покращуючи його [10].

Існує багато модифікацій генетичних алгоритмів. Але всім їм властива універсальність. Тобто від задачі, яку треба розв’язати, залежить визначення функції пристосованості й спосіб кодування рішень. Решта же етапів генетичних алгоритмів виконується однаково для будь-яких задач [10].

Генетичні алгоритми можуть бути використані для широкого класу задач, у тому числі об’ємної розмірності, й характеризуються стійкістю до потрапляння у локальні оптимуми поверхні можливих рішень. Ще однією важливою особливістю ГА є те, що вони можуть бути використані в завданнях зі змінним середовищем [10].

Генетичні методи мають такі переваги:

- відсутність необхідності в специфічних знаннях про розв’язувану задачу. Проте у випадку, якщо додаткова інформація про досліджувану систему, об’єкт або процес є відомою, то вона може бути використана в процесі пошуку;
- концептуальна простота та прозорість реалізації;
- можливість розпаралелювання;
- легкість кодування вхідної і вихідної інформації. Некритичність до виду параметрів досліджуваних систем (можливість використання експертної, емпіричної, довідкової та іншої інформації про об’єкт, поданої різними типами даних);
- можливість застосування до великого кола задач без внесення вагомих змін у внутрішню структуру методу;
- можливість адаптації параметрів генетичного пошуку до особливостей розв’язуваної задачі;
- менша ймовірність попадання і зациклення в локальному оптимумі, яка досягається за рахунок використання популяційного підходу;
- можливість застосування в методі інших пошукових процедур.

Проблематично використовувати ГА у випадках:

- коли необхідно знайти точний глобальний оптимум;
- час виконання функції оцінки великий;
- необхідно знайти всі рішення задачі, а не одне з них;
- кодування рішення надто складне [10];
- якщо простір пошуку є дискретним і невеликим за розміром;
- якщо цільова функція в пошуковому просторі є гладкою і унімодальною;
- якщо про простір пошуку відома деяка додаткова інформація [14].

1.2.4 Мурашиний алгоритм

Мурашиний алгоритм (ant colony optimization, ACO) – один з ефективних поліноміальних алгоритмів для знаходження наближених рішень задачі комівояжера, а також аналогічних задач пошуку маршрутів на графах. Суть підходу полягає в використанні моделі поведінки мурах, що шукають шляхи від колонії до джерела живлення. Схожими з мурашиними алгоритмами є методи колективного (ройового) інтелекту, які описують комплексну колективну поведінку децентралізованої системи, що самоорганізується. Ці методи розглядаються в теорії штучного інтелекту як метод оптимізації [10].

Мурашині алгоритми гарантують знаходження розв'язку задачі, хоча час його отримання визначити практично неможливо. На відміну від ГА, при наступній ітерації використовується пам'ять всіх поколінь, а не лише попереднього. Зазвичай потребують застосування додаткових методів, таких як локальний пошук тощо. Для кожного конкретного типу задач потребують визначення моделюючих параметрів, які можуть бути отримані лише шляхом дослідження [10].

1.2.5 Метод розфарбовування графа

Алгоритм розфарбовування графа дозволяє знаходити (точне або наближене) значення хроматичного числа довільного графа і відповідну цьому значенню розмальовку вершин [15].

Розглянемо простий алгоритм розфарбовування графа.

Крок 1. Впорядкувати вершини по зменшенню ступеня.

Крок 2. Пофарбувати першу вершину в колір 1.

Крок 3. Вибрати колір забарвлення 1.

Крок 4. Пофарбувати в обраний колір всяку вершину, що не суміжно з іншого, вже пофарбованої в цей колір.

Крок 5. Вибрати колір.

Крок 6 Поки не пофарбовані всі вершини, повторювати кроки 4-5 [15].

У цьому простому з методів вершини спочатку розташовуються в порядку незростання їх ступенів. Перша вершина забарвлюється в колір 1; потім список вершин проглядається зверху вниз (по незростання ступенів) і в колір 1 забарвлюється всяка вершина, що не суміжна з іншою, вже пофарбовану в цей колір. Потім повертаємося до першої в списку нерозфарбованої вершини, фарбуємо її в колір 2 і знову переглядаємо список вершин зверху вниз, фарбуючи в колір 2 будь-яку нерозфарбовану вершину, яка не з'єднана ребром з іншою, вже пофарбованою в колір 2 вершиною. Аналогічно діємо з кольорами 3, 4 і т. Д., Поки не будуть зафарбовані всі вершини. Число використаних кольорів буде тоді наближеним значенням хроматичного числа графа [15].

Наведений вище алгоритм можна модифікувати. Для цього після кожного кроку потрібно упорядковувати нефарбовані вершини. Проста модифікація описаної вище евристичної процедури полягає в переупорядковуванні нефарбованих вершин по незростанню їх відносних ступенів. Під відносними ступенями розуміються ступеня відповідних вершин в нерозфарбованому підграфі вихідного графа [15].

У даній модифікації передбачалося, що якщо дві вершини мають однакові ступені, то порядок таких вершин випадковий. Такі вершини можна також упорядкувати, але вже по двокроковими ступенями. Двокроковий ступінь визначимо як суму відносних ступенів інцидентних вершин. Аналогічно можна надходити і далі [15].

При постановці задачі складання розкладу навчальних занять як задачі розфарбовування графу будують граф, в якому кожна з вершин є заняттям. У випадку конфліктів між двома вершинами, їх з'єднують ребром. Це означає заборону проведення цих занять в один час. Конфлікти виникають у випадках, коли обидва заняття проводяться в один час, одним викладачем, тощо [12].

Задача складання розкладу занять розглядається як мінімізація кількості кольорів, що необхідні для розфарбовування графа. При цьому враховується представлення кожного кольору одному періоду занять [12].

1.2.7 Порівняльний аналіз існуючих методів вирішення завдання

В ході роботи, було проведено порівняльний аналіз досліджених методів для формування розкладу занять (табл.1.2).

Завдання формування розкладу занять відноситься до комбінаторного завдання і вирішуються за допомогою методів дискретної оптимізації. В теорії, розв'язати комбінаторне завдання не є проблемою, оскільки вхідні дані (плани) це скінченне число, і тому це завдання можна вирішити шлях перебору усіх планів. Проблема у іншому: чи не занадто велике число планів буде подаватися на вхід, щоб рішення цього завдання можна було реалізувати на практиці.

Таблиця 1.2 – Порівняльна характеристика існуючих методів для формування розкладу занять

Можливості	Метод гілок та меж	Метод імітації відпаду	Генетичний алгоритм	Мурашиний алгоритм	Метод розфарбування графа
1	2	3	4	5	6
Простота та прозорість	Так	Ні	Так	Ні	Так
Адаптивність до особливостей вирішувальної задачі	Ні	Ні	Так	Так	Так
Велика витратність часу при великій розмірності початкових даних	Так	Так	Ні	Так	Ні
Велика точність розв'язку вирішувальної задачі	Ні	Так	Так	Так	Так

Продовження таблиці 1.2

1	2	3	4	5	6
Висока похибка між бажаними і отриманими результатами	Так	Ні	Ні	Так	Ні

Виходячи з таблиці 1.2, можна побачити, що усі методи мають свої недоліки та переваги. Але серед них можна виділити ГА та метод розфарбування графа. З невеликим обсягом даних краще впорається метод розфарбування графа, коли для великих об'ємів даних ГА буде видавати кращий результат, хоч і використовуючи при цьому трохи більше ресурсів. Отже, підсумовуючи вищесказане, було прийняте рішення використовувати один з цих двох методів для розв'язування завдання формування розкладу занять.

1.3 Постановка завдання до роботи

Формування розкладу є важливим завданням для усіх навчальних закладів. Задача формування розкладу вирішується у багатьох галузях. Насамперед, при плануванні дискретного виробництва, організації пасажирських та товарних перевезень, проектуванні та проведенні навчальних занять у середній, професійно-технічній та вищій школі. В її основу покладено необхідність забезпечення оптимального розподілу робіт серед виконавців, враховуючи просторові обмеження за часом[9].

Розклад сам по собі залежить від багатьох факторів. Їх можна поділити на об'єктивні(жорсткі) та суб'єктивні(нежорсткі) параметри. Об'єктивні — це база даних вищого навчального закладу, в якій зберігається інформація про аудиторії та предмети. Суб'єктивні — це вимоги та побажання студентів та викладачів [9].

В даній роботі ми будемо зосереджувати свою увагу саме на розкладі занять навчального закладу, для якого жорсткими умовами будуть:

- предмети, які треба провести;
- викладачі, які ці предмети ведуть;
- студенти;
- аудиторії, де заняття будуть проходити [9].

Також не дуже очевидним ресурсом буде час, який буде розподілений між робочими днями та парами [9].

До нежорстких умов треба віднести:

- вимоги і побажання викладачів;
- вимоги і побажання студентів [9].

Метою роботи є програмна реалізація методів формування розкладу занять.

Для досягнення поставленої мети мають бути виконані наступні завдання:

- порівняти методи за певними критеріями та обрати кращий з них;
- реалізувати серверну частину на основі обраного методу для обробки вхідних даних;
- реалізувати клієнтську частину для завантаження вхідних даних і відображення результатів;
- додати можливість змінювати результати вручну, якщо вони нас не влаштовують;
- перевірити працездатність системи з тестовими та реальними даними.

Математичне представлення поставленої задачі подано у формулі (1.1):

$$f = \{N, T, S, C, W, Lt, Ls, P, D, Db, Tr, M\} \quad , \quad (1.1)$$

де N – предметні години, які треба відпрацювати;

T – викладачі;

S – студенти (групи);

C – аудиторії, де будуть проводитись завдання;

W – кількість можливих вікон;

Lt – предмети, які ведуть викладачі;

Ls – предмети, які є у студентів;

P – кількість пар;

D – кількість робочих днів;

Db – список неробочих днів (свята, події тощо);

Tr – посада викладача;

M – розмір курсу (семестра) для групи студентів (кількість тижнів).

1.4 Висновки за розділом 1

В даному розділі було проаналізовано аналогічні системи вирішення завдання формування розкладу занять, та наведено їх порівняльний аналіз, який показав, що ці аналоги не підходять до нашого випадку.

Проаналізовано основні методи вирішення поставленого завдання та наведено порівняльний аналіз характеристик цих методів, який показав, які алгоритми слід використовувати для вирішення завдання.

Було сформульовано і поставлено завдання до роботи.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного продукту

Наступний етап розробки ПП це вибір інструментарію та використання необхідних технологій у проєкті.

Основна задача ПП – це програмна реалізація методів формування розкладу занять.

Розроблюваний ПП повинен містити такі модулі:

- модуль для розрахунків;
- модуль вводу-виводу даних;
- модуль зберігання даних;
- модуль для зв'язку всіх елементів ПП.

Для проектування ПП необхідно обрати наступні засоби: мову програмування для фронтенду (front-end), мову програмування для бекенду (back-end), середовище розробки (IDE – Integrated Development Environment) та систему управління базами даних (СУБД).

2.1.1 Вибір мови програмування

Для вирішення завдання формування розкладу занять, яка полягає в реалізації методів оптимізації, які були обрані у першому пункті, треба проаналізувати існуючі мови програмування, зробити їх порівняльну характеристику та на її основі обрати ту, яка краще допоможе нам в реалізації рішення завдання. Нижче наведемо опис популярних мов програмування для розробки веб-застосунків, розглянемо їх переваги та недоліки.

Python – це проста в освоєнні, потужна мова програмування. Вона має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис та

динамічне введення тексту Python, разом з інтерпретованою природою, роблять його ідеальною мовою для створення сценаріїв та швидкої розробки додатків у багатьох областях на більшості платформ [16].

Популярність Python обумовлена тим, що дана мова здатна вирішувати широке коло завдань і його застосування можливе на всіх популярних на сьогоднішній день платформах, за винятком мобільного сегменту, на якому щільно вкоренилися специфічні мови. В останні роки Python придбав свою популярність за рахунок ефективності в таких сферах розробки як Machine Learning і Data Science завдяки своїй розширюваності і гнучкості, що так необхідно в даному сегменті програмування. У той же час Python широко застосовується в WEB, в якому дана мова відчуває себе відмінно з початку 2000-х років і до цього дня розвиває і вдосконалює цей напрямок [17].

Переваги мови Python:

- гнучкість – це основна перевага мови, так як завдяки своїй гнучкості мова отримала популярність серед багатьох розробників;
- можливість розширення – існують бібліотеки і фреймворки під будь-який тип завдань і потреб;
- простота синтаксису – код чистий і зрозумілий, без зайвих дужок і виразів. Для того, щоб вивчити синтаксис багато часу не знадобиться;
- Open Source – код інтерпретатора Python є відкритим, що дозволяє будь-кому, хто зацікавлений у розвитку мови взяти участь в його розробці і поліпшити його. Якщо дивитися деталі релізу однією з версій мови, то можна помітити, що величезні частини нового функціоналу реалізовані сторонніми розробниками [17];
- універсальність в застосуванні – на мові Python можна розробляти, як десктопні додатки, так і веб-застосунки. Також, мова дозволяє проводити наукові дослідження, створювати комп'ютерні ігри, мобільні додатки тощо.

Недоліки мови Python:

- швидкість роботи – програмні продукти на мові Python недостатньо швидкі. Також, мова не дуже підходить для вирішення задач з великим обсягом пам'яті;

- складність переходу на інші мови – якщо першою мовою програмування вивчати Python, то надалі перейти на інші мови буде складніше;

- помилки виконання – так як у мові Python динамічна типізація, можуть виникнути помилки виконання, які не були передбачені розробником ПП.

JavaScript (JS) – це мова програмування що дозволяє зробити Web - сторінку інтерактивною, тобто такою що реагує на дії користувача [18].

JavaScript - об'єктно-орієнтована скриптова мова програмування і є діалектом мови ECMAScript [18].

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить у браузерях як мова сценаріїв для надання інтерактивності веб-сторінкам [18].

Переваги мови JavaScript:

- незамінність для веб-розробки – підтримка скриптів усіма популярними браузерами. Повна інтеграція з версткою сторінок (frontend) і серверною частиною (backend);

- швидкість роботи і продуктивність – мова JavaScript дозволяє частково обробляти веб-сторінки на комп'ютерах користувача без запитів до сервера. Це економить час і трафік, знижує навантаження на сервер;

- простота і раціональність застосування – просте завдання можна вирішити за 5 хвилин, не треба робити зайву роботу. Для складних завдань є варіанти вирішення, можна адаптувати;

- легкість освоєння – простота синтаксису і раціональність, інтуїтивна зрозумілість коду [19].

Недоліки мови JavaScript:

- відсутність можливості читання та завантаження файлів – це обмеження функціональності на стороні клієнта за міркуваннями безпеки;
- нестрога типізація і вільне трактування – має місце різна інтерпретація даних. Немає можливості раннього виявлення помилок. Всі недоліки виявляються вже на етапі роботи;
- доступність для зломисників – у вільну скриптову мову можна найпростіше вбудувати фрагмент шкідливого коду, який може нашкодити користувачеві [19].

Java – об’єктно-орієнтована мова програмування загального призначення. Відноситься к мовам з сильною типізацією. Творці реалізували принцип WORA: write once, run anywhere або «пиши один раз, запускай де завгодно». Це означає, що написаний на Java додаток можна запустити на будь-якій платформі, якщо на ній встановлене середовище виконання Java (JRE, Java Runtime Environment) [20].

На мові Java написано багато веб-додатків. Популярні фреймворки, в тому числі Spring та Stuts, використовуються для створення різних додатків в інтернеті: від комерційних проектів до великих порталів, від освітніх платформ до урядових ресурсів [20].

Переваги мови Java:

- незалежність від платформи – написаний на Java додаток можна запустити на будь-якій платформі, якщо на ній встановлене середовище виконання Java JRE;
- легкість освоєння – простота синтаксису і раціональність, інтуїтивна зрозумілість коду;
- повна об’єктна орієнтованість – мова Java повністю об’єктно-орієнтована: всі сутності це об’єкти, а функції, в основному це методи класів;
- висока надійність – мова Java створена таким чином, щоб розробник коду не зміг створити проблемний програмний продукт. Для цього в Java є багато різних внутрішніх механізмів.

Недоліки мови Java:

- низька продуктивність – з-за того, що Java має багато різних внутрішніх механізмів, вона має низьку продуктивність;
- відсутність нативного дизайну - Java-інструменту для створення нативного дизайну немає;
- багатослівний і складний код – коли коду стає забагато, то він становиться складним для читання та розуміння [21].

C Sharp – відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C ++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML [22].

Переваги мови C Sharp:

- підтримка корпорації Майкрософт – на відміну від Java, якій не пішов на користь перехід у власність Oracle, C # добре розвивається завдяки зусиллям Microsoft;
- багато синтаксичного цукру. Синтаксичний цукор - це такі конструкції, які створені для полегшення написання і розуміння коду (особливо якщо це код іншого програміста) і не грають ролі при компіляції;
- середній поріг входження. Синтаксис схожий на C, C ++ або Java полегшує перехід для інших програмістів. Для новачків це також один з найперспективніших мов для вивчення [23].

Недоліки мови C Sharp:

- орієнтованість, в основному, тільки на .NET (на Windows платформу);
- безкоштовність тільки для невеликих компаній, учнів і програмістів-одинаків. Для великих команд покупка ліцензій обійдеться недешево;
- наявність оператора go to [23];

Проаналізувавши роботи [16]-[23], було зроблено порівняльну характеристику мов програмування для вирішення поставленого завдання, яка представлена у табл. 2.1.

Таблиця 2.1 – Порівняльна характеристика мов програмування для вирішення поставленого завдання

Критерії порівняння	Python	JavaScript	Java	C Sharp
Ліцензія	Python Software Foundation License	New BSD	GNU GPL	GNU GPL
Кросплатформеність	+	+	+	+
Легкість освоєння	++	++	+	+
Рівень можливостей	++	++	+	+
Можна використовувати для frontend (клієнтська частина)	+	+++	+	+
Можна використовувати для backend (серверна частина)	+++	+++	+	+
Входить в топ 3 мов програмування веб-застосунків	+++	++	++	+

З табл. 2.1. можна побачити, що мови Python та JavaScript мають майже однакові характеристики. Якщо Python краще використовувати для бекенду, то Javascript підходить як для фронтенду, так і для бекенду.

Проаналізувавши платформи та бібліотеки для цих мов програмування, я дійшов висновку, що краще всього, для розробки веб-застосунку підійде мова JavaScript з платформою Node.js та бібліотекою React, які дозволяють розробляти як фронтенд, так і бекенд. Також, з ними зручно працювати за самого початку, завдяки доступній і зрозумілій документації та іншим інструментам, які можуть полегшити входження для

новачків. Отже, підбиваючи висновки, було обрано мову програмування JavaScript та платформу Node.js для розробки бекенду та бібліотеку React для розробки фронтенду.

2.1.2 Вибір середовища розробки

Для того щоб писати код на будь-якій мові програмування, потрібно відповідне середовище розробки.

Для мови JavaScript існує багато різних середовищ та редакторів коду, але я виділив лише декілька цікавих для нас: Sublime Text, Visual Studio Code та WebStorm.

Кожне з цих середовищ розробки мають графічний інтерфейс та корисні і різноманітні інструменти для створення великих проектів на мові JavaScript, будь то мобільний додаток або веб-застосунок. Розглянемо детальніше наведені вище IDE.

Sublime Text – це платний редактор з безкоштовним пробним періодом. У ньому немає великої кількості встановлених плагінів, але взагалі ці плагіни є, у всьому їх різноманітті, для будь-яких потреб. Деякі пакети, такі як SideBarEnhancements (дозволяє перейменовувати, переміщати, копіювати і вставляти файли і папки), мабуть, повинні бути вбудовані в редактор, але ви можете їх завантажити і встановити. На первинне налаштування цього редактора може знадобитися якийсь час [24].

Переваги середовища Sublime Text:

- легкий в освоєнні;
- популярний серед спільноти;
- можна налаштовувати під користувача;
- компактний та ефективний [24]-[25].
- Недоліки середовища Sublime Text:
- для роботи потребує завантаження додаткових пакетів;
- комерційний проект з урізаною функціональністю [24]-[25].

Visual Studio Code (VSCode) – це безкоштовний редактор коду від Microsoft. Він має відкритий вихідний код і легковагий при установці. VSCode відмінно підходить для початківців JavaScript-розробників, оскільки має хороший набір функціоналу «з коробки», без необхідності встановлювати додаткові плагіни. Але цей редактор популярний не тільки серед початківців програмістів. Він може стати ідеальним вибором для більш просунутих користувачів, яким потрібно просто швидко приступити до роботи, не витрачаючи зайвий час на налаштування [24].

Унікальна особливість VSCode - можливість використовувати його в браузері. Таким чином ви можете користуватися редактором на планшетах і при цьому мати таку ж середу, до якої звикли в десктопній версії [24].

Переваги середовища Visual Studio Code:

- регулярні оновлення та підтримка;
- безкоштовна;
- досить маленьке споживання пам'яті;
- простий у використанні;
- має вбудований термінал [24]-[25].

Недоліки середовища Visual Studio Code:

- робота терміналу не завжди працює задовільно;
- для роботи потребує завантаження додаткових пакетів [24]-[25].

WebStorm виділяється із загального списку, тому що це багата інтегроване середовище розробки для JavaScript. Розробник цієї IDE – JetBrains [24].

Це середовище відмінно підходить для різних платформ, таких як React, Angular, Vue.js і т. Д. З її допомогою можна займатися налагодженням node-скриптів і запускати тести на вбудованому сервері. Також можна запускати і налагоджувати прм-скрипти (за допомогою деревоподібного інтерфейсу). І для всього цього не потрібні ніякі плагіни [24].

Проте, плагіни доступні для деяких більш специфічних функцій, які не вбудованих в саму IDE. Але в більшості випадків все, що вам може

знадобитися, в IDE уже є. Найприємніше в цьому те, що ви можете відкрити для себе нові, невідомі вам перш функції, і оцінити, наскільки здорово, що вони є [24].

У WebStorm за замовчуванням налаштований автосохранение файлів по ходу роботи над ними. Коли після цієї IDE переходиш на який-небудь інший редактор без подібного функціоналу, починаєш гостро відчувати, наскільки його не вистачає. Втім, це не унікальна особливість WebStorm, просто тут реалізація трохи приємніше [24].

Переваги середовища WebStorm:

- багато різноманітних інструментів для розробки проектів на мові JavaScript;
- розроблено професіоналами спеціально для мови JavaScript [24]-[25].

Недоліки середовища WebStorm:

- потребує наявності багатого обсягу оперативної пам'яті для нормального функціонування;
- потребує самостійного налаштування під різні проекти [24]-[25].

Проаналізувавши джерела [24]-[25] було зроблено порівняльну характеристику середовищ розробки для вирішення поставленого завдання, яка представлена у табл. 2.2.

Таблиця 2.2 – Порівняльна характеристика середовищ розробки для вирішення поставленого завдання

Критерії порівняння	Sublime Text	Visual Studio Code	WebStorm
Ліцензія	Proprietary Software	MIT	Apache License 2.0
Кросплатформеність	+	+	+
Наявність автодоповнення та автовиправлення	±	+	+
Підтримка різних репозиторіїв	+	+	+
Зручний інтерфейс користувача	+	+	+
Рівень функціоналу	+	++	+++
Безкоштовний IDE	-	+	-
Створений для розробки на JavaScript	-	±	+

Як видно з таблиці 2.2. більшість переваг має саме Visual Studio Code, так як безкоштовний та має необхідний функціонал для розробки веб-застосунків на мові JavaScript. Навіть в безкоштовній версії рівень функціоналу досить великий. Також в Visual Studio Code є різноманітні інструменти та вбудовані функції, які полегшують написання коду та створення проектів. Отже, підбиваючи висновки, було обрано середовище розробки Visual Studio Code для вирішення поставленого завдання.

2.2 Структурна схема програмного продукту

Розроблюваний ПП повинен обов'язково містити такі модулі:

- розрахунковий;
- зберігання даних;
- вводу-виводу даних;
- модуль зв'язку всіх елементів ПП.

Розрахунковий модуль – це модуль на серверній частині (бекенд), який повинен приймати запити від клієнта (фронтенд), і керуючи даними з БД, за допомогою відповідних функцій обробляти інформацію і генерувати оптимальну відповідь.

Модуль зберігання даних – це модуль бази даних, який забезпечує керування та управління даними у БД.

Модуль вводу-виводу даних – це модуль, який відповідає за введення даних користувача та виведення результатів у заданому форматі відображення.

Модуль зв'язку всіх елементів ПП – це модуль, який пов'язує роботи усіх компонентів системи и забезпечує правильне її функціонування.

Розроблюваний ПП використовує дані про аудиторії, групи, викладачів, предмети тощо, та робить з цих даних оптимальний розклад. Ці дані можуть бути представлені у вигляді файлів Excel, текстових файлів (txt,

xml, csv тощо) а також у вигляді нереляційної бази даних (MongoDB, DynamoDB Cassandra тощо).

Обраний формат зберігання повинен відповідати таким вимогам:

- мати здатність зручно компонувати інформацію та витягувати її у разі необхідності;
- мати здатність зберігати великий об'єм даних;
- багаторазове використання даних;
- швидкий пошук і отримання інформації за запитом користувача;
- простота оновлення даних;
- захист даних від несанкціонованого доступу, спотворення і знищення;
- зменшення зайвої надмірності даних [26].

Цим вимогам відповідають як таблиці реляційних баз даних (БД), так і сутності нереляційних баз даних (NoSQL). Але між ними є багато відмінностей. Порівняльна характеристика реляційних та нереляційних БД представлена у табл. 2.3.

Таблиця 2.3 – Порівняння реляційних та нереляційних БД [27]

Порівняльні характеристики	Реляційні бази даних	Нереляційні бази даних (NoSQL)
1	2	3
Відповідні робочі навантаження	Реляційні БД призначені для транзакційних та строго несуперечливих додатків обробки транзакцій в режимі реального часу і добре підходять для аналітичної обробки в режимі реального часу	Бази даних NoSQL призначені для роботи з цілою низкою шаблонів доступу до даних, в тому числі додатків з низькою затримкою. Пошукові БД NoSQL призначені для аналітики частково структурованих даних

Продовження таблиці 2.3

1	2	3
Модель даних	Реляційна модель нормалізує дані і перетворює їх в таблиці, що складаються з рядків і стовпців. Схема жорстко задає таблиці, рядки, стовпці, індекси, відносини між таблицями та інші елементи бази даних. Така БД забезпечує цілісність посилованих даних у відносинах між таблицями	Бази даних NoSQL надають різноманітні моделі даних, такі як пари «ключ-значення», документи і графи, оптимізовані для високої продуктивності і масштабованості
Властивості ACID	Реляційні бази даних забезпечують набір властивостей ACID: атомарність, неуперечливість, ізолюваність, надійність	Бази даних NoSQL часто пропонують компроміс, пом'якшуючи жорсткі вимоги властивостей ACID заради більш гнучкої моделі даних, яка допускає горизонтальне масштабування
Продуктивність	Продуктивність головним чином залежить від дискової підсистеми. Для забезпечення максимальної продуктивності часто потрібно оптимізація запитів, індексів і структури таблиці	Продуктивність зазвичай залежить від розміру кластера базового апаратного забезпечення, затримки мережі і викликає додатки

Продовження таблиці 2.3

1	2	3
Масштабування	Реляційні бази даних зазвичай масштабуються шляхом збільшення обчислювальних можливостей апаратного забезпечення або додавання окремих копій для робочих навантажень читання	Бази даних NoSQL зазвичай підтримують високу роздільність завдяки шаблонами доступу з можливістю масштабування на основі розподіленої архітектури. Це підвищує пропускну здатність і забезпечує стійку продуктивність майже в необмежених масштабах
API	Запити на запис і вилучення даних складаються на мові SQL. Ці запити аналізує і виконує реляційна база даних	Об'єктно-орієнтовані API дозволяють розробникам додатків без проблем здійснювати запис і витяг структур даних. Завдяки використанню ключів секцій додатки можуть вести пошук по парам «ключ-значення», наборам стовпців тощо

Виходячи з табл. 2.3. можна зробити висновок, що нереляційні БД мають низку переваг перед реляційними. Отже, проаналізувавши види баз даних, я дійшов висновку, що для виконання проектного завдання підійде використання нереляційної бази даних, а саме MongoDB.

Для відображення структури нереляційної бази даних було створено ER-діаграму (рис. 2.1).

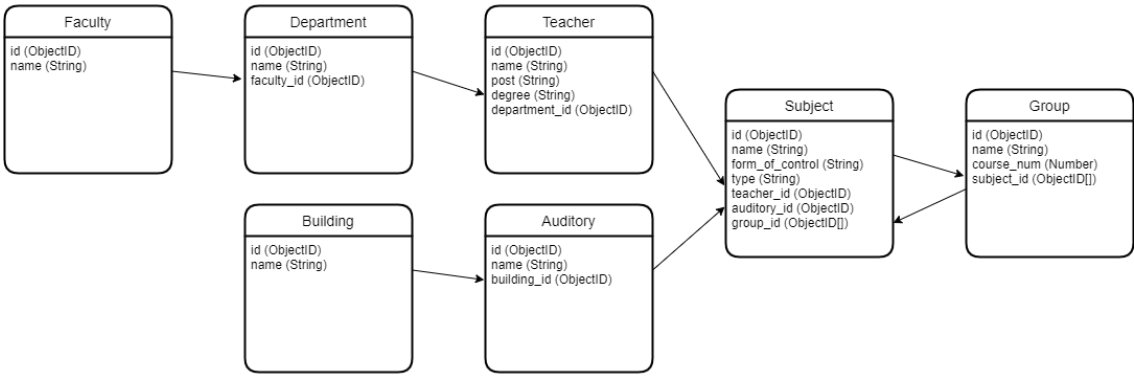


Рисунок 2.1 – ER-діаграма розроблюваної нереляційної бази даних

Опис сутностей та їх атрибути наведені у табл. 2.4.

Таблиця 2.4 – Опис сутностей та їх атрибутів

Сутність	Атрибут	Тип	Опис сутності
1	2	3	4
Faculty	id	Лічильник	Містить інформацію про факультет
	name	Строковий	
Department	id	Лічильник	Містить інформацію про кафедру та якому факультету вона належить
	name	Строковий	
	faculty_id	Лічильник	

Продовження таблиці 2.4

1	2	3	4
Teacher	id	Лічильник	Містить інформацію про викладача, його наукове звання, посаду та якій кафедрі він належить
	name	Строковий	
	post	Строковий	
	degree	Строковий	
	department_id	Лічильник	
Building	id	Лічильник	Містить інформацію про корпус університету, в якому будуть проходити заняття
	name	Строковий	
Auditory	id	Лічильник	Містить інформацію про аудиторію, та якому корпусу вона належить
	name	Строковий	
	building_id	Лічильник	
Subject	id	Лічильник	Містить інформацію про предмет викладання, його форму контролю, тип (лекція/екзамен), викладача, аудиторію, та масив груп, яким цей предмет викладається
	name	Строковий	
	form_of_control	Строковий	
	type	Строковий	
	teacher_id	Лічильник	
	auditory_id	Лічильник	
	group_id	Лічильник	
Group	id	Лічильник	Містить інформацію про групу студентів, їх спеціальність та список предметів, які їм викладаються
	name	Строковий	
	course_num	Числовий	
	subject_id	Лічильник	

Виходячи з рисунку 2.1 та таблиці 2.4 можна побачити, що між сутностями є зв'язки у вигляді поля посилянь (поле типу ObjectID), яке вказує на зв'язок сутності з іншою (одна сутність є частиною іншої тощо).

Структурну схему розроблюваного ПП можна побачити на рисунку 2.2.

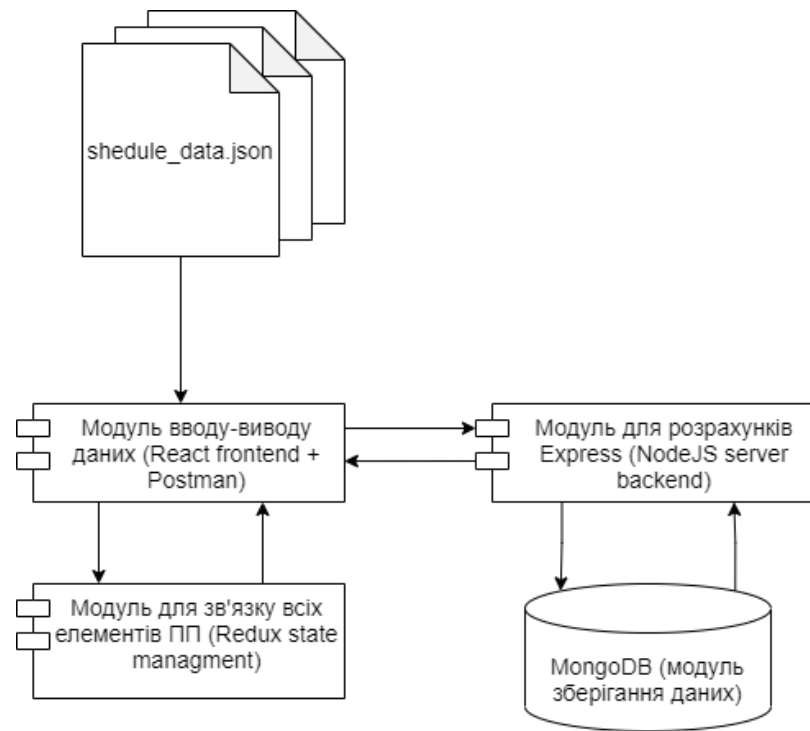


Рисунок 2.2 – Структурна схема розроблюваного програмного продукту

На рисунку 2.2 можна побачити, що веб-застосунок використовує дані у форматі JSON та підгружає їх за допомогою модуля вводу-виводу, який реалізований за допомогою бібліотеки React та додатку Postman. Завантажені дані переходять до модуля розрахунків, який реалізований за допомогою серверу Express (NodeJS), та модуля зберігання даних, у вигляді нереляційної БД MongoDB, яка виділяє нам кластер для наших потреб. Зв'язок між модулями реалізований за допомогою бібліотеки Redux.

2.3 Висновки за розділом 2

В даному розділі було проведено аналіз та вибір інструментарію для розробки програмного продукту, було спроектовано ER-діаграму та структуру майбутнього ПП. Проведено порівняльну характеристику мов програмування та встановлено, що для розробки нашого ПП краще за все підійде мова програмування JavaScript і платформа NodeJS для реалізації

серверної частини, та необхідні бібліотеки для реалізації клієнтської частини. Також, було проведено порівняльну характеристику середовищ розробки та встановлено, що для розробки нашого ПП на обраній мові краще за все підійде IDE Visual Studio Code. Також, було проведено порівняльну характеристику реляційних та нереляційних баз даних, та встановлено, що для реалізації більш за все підійде нереляційна БД, а саме MongoDB.

3. ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Обробка даних

Для роботи з існуючими методами, які розглядались у першому розділі, був використаний тестовий файл формату CSV з наступними заголовками: «sub_id» (ідентифікатор заняття), «teacher_id» (ідентифікатор викладача), «group_id» (ідентифікатор групи студентів), «room_id» (ідентифікатор аудиторії) та «subject_name» (назву заняття). Тестові дані зображені на рисунку 3.1.

	A	B	C	D	E	
1	sub_id	teacher_id	group_id	room_id	subject_name	
2	1	1	1	1	sub 1	
3	2	1	2	1	sub 1	
4	3	1	3	1	sub 1	
5	4	2	1	2	sub 2	
6	5	2	2	2	sub 2	
7	6	2	3	2	sub 2	
8	7	3	1	4	sub 3	
9	8	3	2	4	sub 3	
10	9	3	3	4	sub 3	
11	10	3	4	3	sub 4	
12	11	2	4	1	sub 3	
13						

Рисунок 3.1 – Тестові дані з файлу формату CSV

Для тестування було обрано наступне завдання: треба провести 3 предмета (від sub 1 до sub-3) в кожній з трьох груп студентів, а також 1 предмет у четвертій групі. Кожен предмет викладає один з трьох викладачів в одному з чотирьох кабінетів.

3.2 Порівняння обраних методів для вирішення завдання формування розкладу

3.2.1 Методи гілок та меж, імітації відпалу та мурашиний алгоритм

Після обробки даних, метод гілок та мереж видав багато варіантів розкладу. Але перший оптимальний варіант був отриманий десь в середині ітерацій. Наступні після цього ітерації знаходили варіації отриманого розкладу. Перший результат був більш-менш оптимальний, але для його знаходження треба було пройти усі ітерації, що, звісно, потребувало багато зайвих ресурсних витрат.

Щоб використати цей метод для розв'язування задачі формування розкладу занять, треба провести процедуру розгалуження, що не дуже легко, тому що у багатьох груп може бути одне заняття в один момент часу.

Затрати часу методу гілок та меж зі збільшенням об'єму інформації вхідних даних збільшувались в прогресії. Чим більше становилося даних, тим більше часу було потрібно для їх обробки. Також зменшувалась точність формування розкладів. Точність результату роботи цього методу дорівнювала приблизно 82%. Витрачений час на роботу дорівнював приблизно 60 (0.06 мс).

Після обробки даних, метод імітації відпалу видав більш-менш оптимальне рішення при невеликому об'єму вхідних даних. При цьому, він використовував невиправдано багато ресурсів для того, щоб знайти результат. Точність результату роботи цього методу дорівнювала приблизно 72%. Витрачений час на роботу дорівнював приблизно 86 (0.086 мс).

Після обробки даних, мурашиний алгоритм не одразу видав оптимальний результат. Це вийшло тому, що для правильної роботи методу треба було налаштовувати його вручну, змінюючи параметри методу експериментальним шляхом. Після чого, оптимальний розклад був знайдений. Точність результату роботи цього методу дорівнювала приблизно

78%. Витрачений час на роботу дорівнював приблизно 30 (0.03 мс). Графік залежності роботи методів за швидкістю можна побачити на рисунку 3.2.

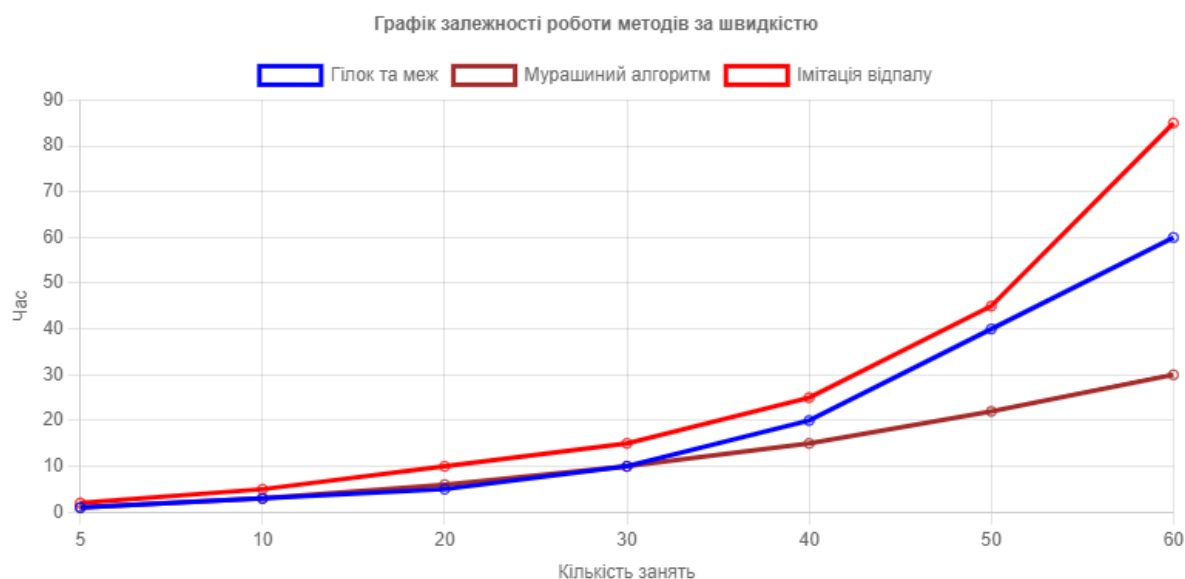


Рисунок 3.2 – Графік залежності роботи наведених методів за швидкістю

Як можна побачити, найкращий метод за швидкістю серед наведених – це мурашиний алгоритм. Графік залежності роботи методів за точністю можна побачити на рисунку 3.3.



Рисунок 3.3 – Графік залежності роботи наведених методів за точністю

Як можна побачити, самий оптимальний метод за точністю серед наведених – це метод гілок та меж. Але, з-за деяких особливостей вирішуваної задачі, ці методи не підходять для використання. Методи, які відповідають вимогам та можуть підійти під особливості вирішуваної задачі це ГА та метод розфарбовування графа. Розглянемо їх у наступному підпункті.

3.2.2 Метод розфарбовування графа та генетичний алгоритм

Після обробки вхідних даних, був розроблений граф з 11 вершинами та 56 ребрами, де вершини відповідають за конкретне заняття, а ребра: зв'язки між заняттями які проводяться у одній аудиторії, викладає один викладач та які проводяться у одній конкретній групі.

Припустимо, що треба сформувати розклад на 1 день, у якому можна провести до 6 пар. Результат роботи методу можна побачити в таблиці 3.1.

Таблиця 3.1 – Результат роботи методу розфарбовування графу

	Група №1	Група №2	Група №3	Група №4
1 Пара	Subject 1	Subject 2	Subject 3	
2 Пара	Subject 3			Subject 3
3 Пара	Subject 2	Subject 3	Subject 1	
4 Пара		Subject 1	Subject 2	Subject 4
5 Пара				
6 Пара				

Як видно з табл. 3.1 усім групам було сформовано розклад, який не спричиняє конфліктів між групами, викладачами та аудиторіями. Також, можна помітити, що кількість пар залежить від кількості навчальних занять. Якщо вхідними даними задати кількість занять N , то мінімальне значення пар $P_{\min}$ буде дорівнювати кількості занять N формула (3.1).

$$P_{\min} = N \quad (3.1)$$

За одну ітерацію алгоритм сформував оптимальний розклад. При цьому початковою вершиною була обрана вершина під номером один. Якщо задати іншу початкову вершину, розклад зміниться, але досі буде оптимальним. За великим обсягом даних алгоритм теж зміг показати оптимальний розклад, але вимагав більшу кількість витраченого часу. У цього алгоритму є головний недолік – він не об'єднує заняття в одну пару, як наприклад для проведення лекції для декількох груп. Але це обмеження можна обійти – з самого початку записати в одне заняття декілька груп разом.

Затрати часу методу розфарбовування графа зі збільшенням об'єму вхідних даних були мінімальні, тобто майже не збільшувались. Точність при цьому не враховувалася, оскільки оптимальних рішень було кілька (різні варіації).

Генетичний алгоритм в свою чергу теж отримує оптимальний розклад, який варіацією розкладу, отриманого у методі розфарбування графу. Але процес формування розкладу у ГА потребував трохи більше часу ніж у його суперника. Графік залежності роботи ГА та методу розфарбування графу за швидкістю можна побачити на рисунку 3.4.

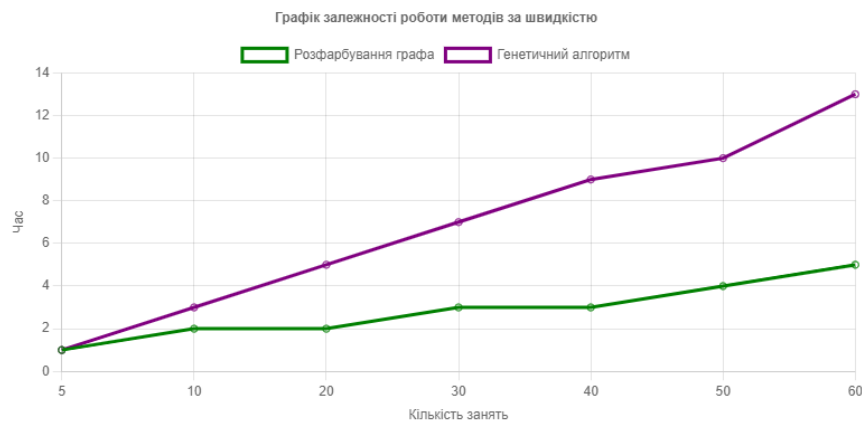


Рисунок 3.4 – Графік залежності роботи ГА та методу розфарбування графу за швидкістю

Як можна побачити на рисунку 3.4, витрати часу у обох методів зростають при збільшенні кількості вхідних даних, але у ГА витрати помітно більше. Графік залежності роботи ГА та методу розфарбування графу за точністю можна побачити на рисунку 3.5.

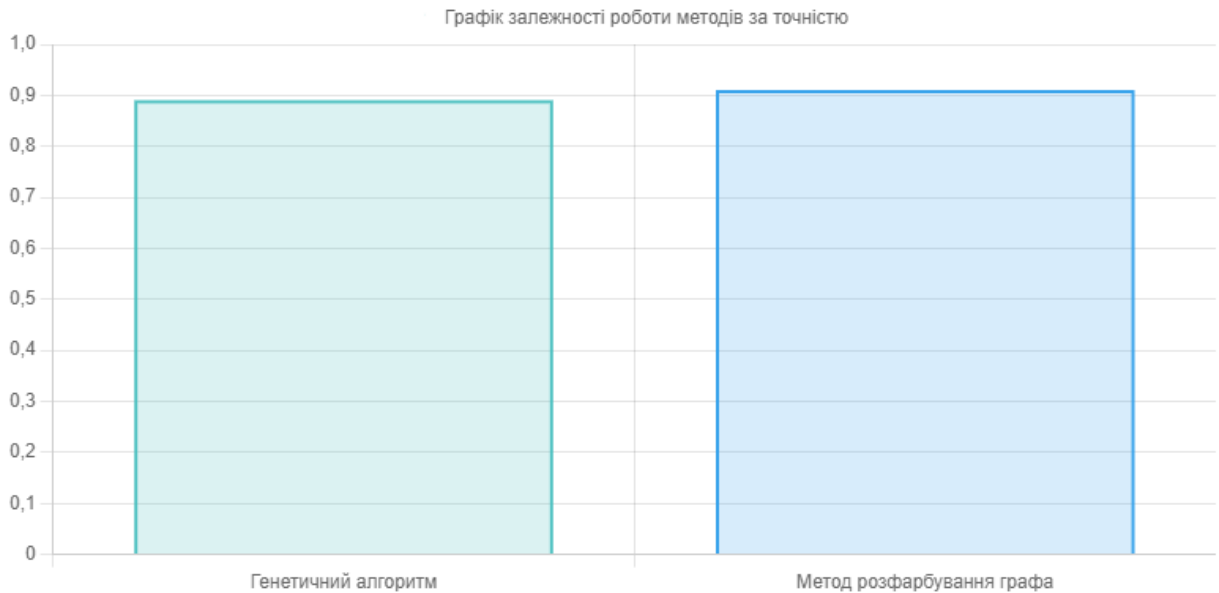


Рисунок 3.5 – Графік залежності роботи ГА та методу розфарбування графу за точністю

Як можна побачити на рисунку 3.5, методи майже ідентичні за точністю, хоч метод розфарбування графу трохи точніше. Але це не говорить про те, що ГА гірше методу розфарбовування графа. Навпаки, він краще методу по багатьом параметрам, але його не рекомендується використовувати для малих вибірок даних. В нашому випадку, ми використовуємо саме такий об'єм вибірки.

Отже, я дійшов висновку, що обидва методи можуть підійти для вирішення задачі формування розкладу занять. Але, для нашого окремого випадку, проаналізувавши обидва методи, було вирішено використовувати метод розфарбування графа, оскільки він дуже добре показав себе при обробці невеликих масивів даних. Для подальших модифікацій та при

зростанні вхідної вибірки, рекомендується вже використовувати генетичний алгоритм.

3.3 Висновки за розділом 3

В даному розділі було проаналізовано вхідні дані у спрощеному вигляді. Також, були проаналізовані на працездатність, точність результатів та швидкість роботи наступні методи: Метод гілок та меж, Метод імітації відпалу, Мурашиний алгоритм, Генетичний алгоритм та Метод розфарбування графа. Для вирішення поставленого завдання було обрано метод розфарбування графа, оскільки він дуже добре показав себе при обробці невеликих масивів даних. Для подальших модифікацій та при зростанні вхідної вибірки, рекомендується вже використовувати генетичний алгоритм.

4. КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови використання веб-застосунку

Розроблений веб-застосунок призначений для вирішення задачі формування розкладу занять. Цим веб-застосунком можуть користуватися як викладачі, так і студенти вищих навчальних закладів.

Для роботи програми, існують мінімальні характеристики системи, які можна побачити в табл. 4.1

Таблиця 4.1 – Мінімальні характеристики та вимоги до клієнтської системи

Характеристика	Значення
Операційна система	Windows 10 x86/x64
Графічний процесор	Intel Core i5
Жорсткий диск (ємкість)	10 Gb
Оперативна пам'ять (ємкість)	8 Gb
Швидкість Ethernet-з'єднання	10-100 Mb

4.2 Характеристики програмного продукту

Розроблюваний веб-застосунок використовує у своїй роботі наступні компоненти:

- середовище Visual Studio Code для написання коду на мові JavaScript;
- проект на платформі Node.js;
- бібліотека React;
- бібліотека Redux;
- додаток Postman;
- доступ к мережі Ethernet.

Розроблюваний веб-застосунок складається з таких модулів:

- модуль зберігання даних. Для зберігання даних використовується нереляційна БД MongoDB з кластером, розміром 500 Mb;
- розрахунковий модуль. Для розрахунків використовуються скрипти на серверній частині (бекенд), які приймають запити від клієнта (фронтенд), і керуючи даними з БД, за допомогою відповідних функцій обробляють інформацію і генерують оптимальну відповідь.
- модуль вводу-виводу. Для вводу даних у форматі JSON відповідає додаток Postman. Для виводу даних використовуються клієнтська частина, яка створена за допомогою бібліотеки React;
- модуль зв'язку усіх елементів ПП. Для зв'язку елементів використовується бібліотека Redux на клієнтській частині.

4.3 Звертання до програми (файли проєкту)

Розроблений веб-додаток на мові програмування JavaScript складається з наступних модулів: модуль фронтенду (клієнтської частини) та модуль бекенду (серверної частини).

Модуль фронтенду містить такі скрипти (субмодулі):

- App.jsx – пов'язує між собою розроблені модулі та повертає розмітку сторінки;
- index.js – повертає компонент App з розміткою та стан веб-застосунку;
- package.json – містить головну інформацію про конфігурацію клієнтської частини проєкту та його залежні модулі;
- package-lock.json – містить детальну інформацію про залежні модулі;
- Content.jsx – компонент, який повертає таблицю з результатами роботи веб-застосунку;
- Content.scss – файл зі стилями для компонента Content;

– index.html – головна сторінка проекту, яка постійно перерисовується завдяки роботі компонентів.

Модуль бекенду містить такі скріпти (субмодулі):

– index.js – скріпт, відповідаючий за підключення до БД MongoDB;
– default.json – файл, який містить в собі головні змінні, які потрібні для підключення до БД;

– auditory.model.js – містить в собі модель «Аудиторія» для БД;
– building.model.js – містить в собі модель «Корпус» для БД;
– day_of_week.model.js – містить в собі модель «День тижня» для БД;

– department.model.js – містить в собі модель «Кафедра» для БД;
– faculty.model.js – містить в собі модель «Факультет» для БД;
– group.model.js – містить в собі модель «Група» для БД;
– pair.model.js – містить в собі модель «Пара» для БД;
– subject.model.js – містить в собі модель «Предмет» для БД;
– teacher.model.js – містить в собі модель «Викладач» для БД;
– auditory.routes.js – містить в собі розроблені запити для колекції «Аудиторія» у БД;

– building.routes.js – – містить в собі розроблені запити для колекції «Корпус» у БД;

– day_of_week.routes.js – – містить в собі розроблені запити для колекції «День тижня» у БД;

– department.routes.js – – містить в собі розроблені запити для колекції «Кафедра» у БД;

– faculty.routes.js – – містить в собі розроблені запити для колекції «Факультет» у БД;

– group.routes.js – – містить в собі розроблені запити для колекції «Група» у БД ;

– pair.routes.js – – містить в собі розроблені запити для колекції «Аудиторія» у БД;

- `subject.routes.js` – містить в собі розроблені запити для колекції «Предмет» у БД;
- `teacher.routes.js` – містить в собі розроблені запити для колекції «Викладач» у БД;
- `package.json` – містить головну інформацію про конфігурацію серверу та його залежні модулі;
- `package-lock.json` – містить детальну інформацію про залежні модулі;
- `proba.js` – скрипт, який містить в собі реалізацію методу розфарбовування графу.

4.4 Вхідні і вихідні дані

Вхідними даними для веб-застосунку є інформація з БД у вигляді пов'язаних між собою колекцій. Ці дані можна занести до БД за допомогою запитів у якості параметрів яких виступають рядки у форматі JSON (JavaScript Object Notation).

Вихідними даними веб-застосунку є оптимальний розклад занять, який відображається на головній сторінці застосунку у вигляді таблиці.

4.5 Повідомлення

Під час роботи веб застосунку генеруються повідомлення, які описані в табл.4.2.

Таблиця 4.2 – Повідомлення та їх опис

Повідомлення	Опис
Server error	Помилка серверу
Server started on port...	Сервер запущений під портом ...

4.6 Висновки за розділом 4

У даному розділі було розглянуто основні функції веб-застосунку, їх опис та характеристики.

Була переглянута структура проекту у вигляді модулів, які складаються з скриптів. Розглянуті вхідні та вихідні дані для веб-застосунку.

Також, були розглянуті основні типи повідомлень, які генеруються під час роботи веб-застосунку.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення й умови виконання веб-застосунку

Розроблений веб-застосунок призначений для вирішення задачі формування розкладу занять. Цим веб-застосунком можуть користуватися як викладачі, так і студенти вищих навчальних закладів.

Застосунок призначений для систем с характеристиками, які можна побачити в табл. 5.1

Таблиця 5.1 – Характеристики системи

Характеристика	Значення
Операційна система	Windows 10 x86/x64
Графічний процесор	Intel Core i5
Жорсткий диск (ємкість)	10 Gb
Оперативна пам'ять (ємкість)	8 Gb
Швидкість Ethernet-з'єднання	10-100 Mb

5.2 Виконання програмного продукту

Для запуску веб-застосунку треба відкрити проект у середовищі Visual Studio Code. Після чого, створити термінал і ввести у ньому команду `cd .\server\` , для переходу у папку серверу. Для запуску сервера треба в терміналі ввести команду `npm run start` , яка запустить сервер та підключиться до БД MongoDB. Після успішного підключення, застосунок згенерує повідомлення у терміналі «Server started on port 5000». Підключення до серверу можна побачити на рисунку 5.1.

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Попробуйте нову кроссплатформенну оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\Sergio\Desktop\schedule_project> cd .\server\
PS C:\Users\Sergio\Desktop\schedule_project\server> npm run start

> server@1.0.0 start C:\Users\Sergio\Desktop\schedule_project\server
> nodemon index.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Server started on port 5000

```

Рисунок 5.1 – Запуск серверу та підключення до БД MongoDB

Щоб побачити, які дані зберігає база даних MongoDB, треба перейти на її офіційний сайт, увійти в свій обліковий запис та перейти до створеного кластеру під назвою scheduleDB. Сторінку з кластером можна побачити на рисунку 5.2.

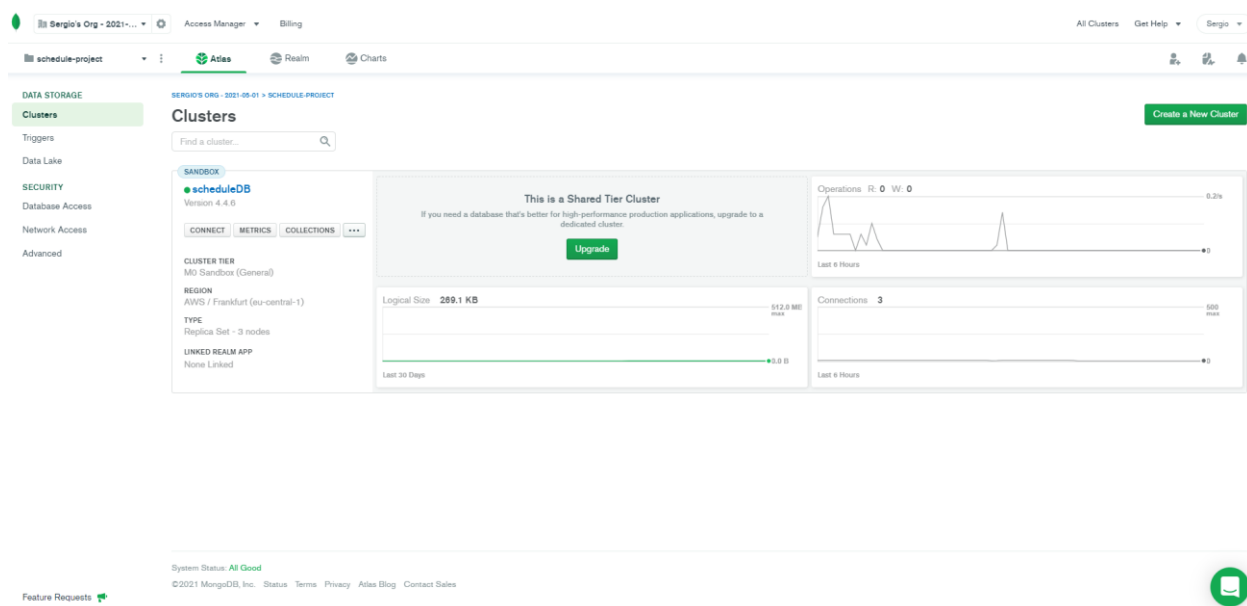


Рисунок 5.2 – Головна сторінка з кластером scheduleDB

Щоб побачити список створених колекцій, треба під назвою кластеру натиснути кнопку «Collections». Список створених колекцій зображений на рисунку 5.3.

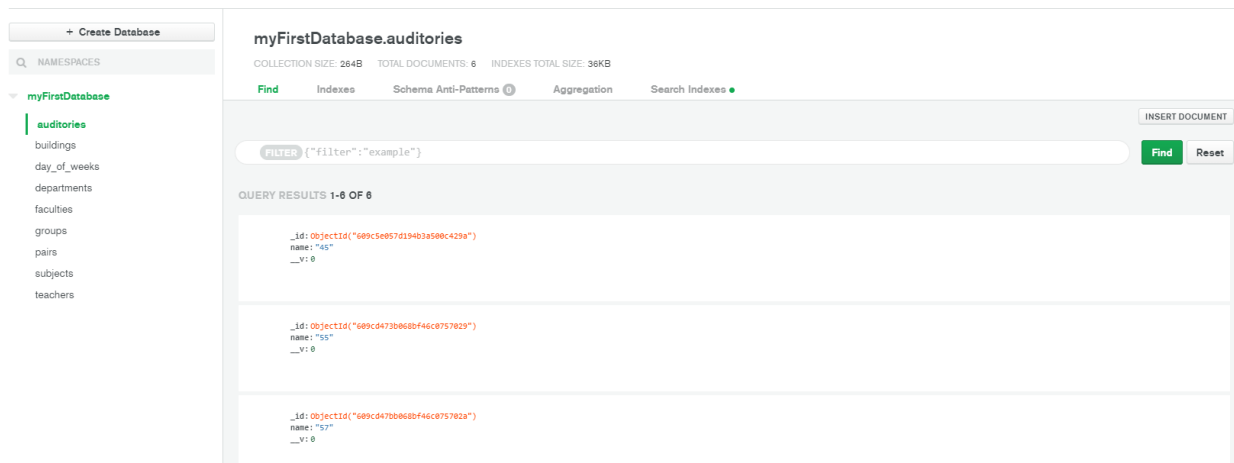


Рисунок 5.3 – Список створених колекцій у кластері scheduleDB

Заповнення бази даних значеннями здійснюється за допомогою застосунку «Postman», який дозволяє імітувати виконання запитів до БД. Графічний інтерфейс застосунку Postman можна побачити на рисунку 5.4

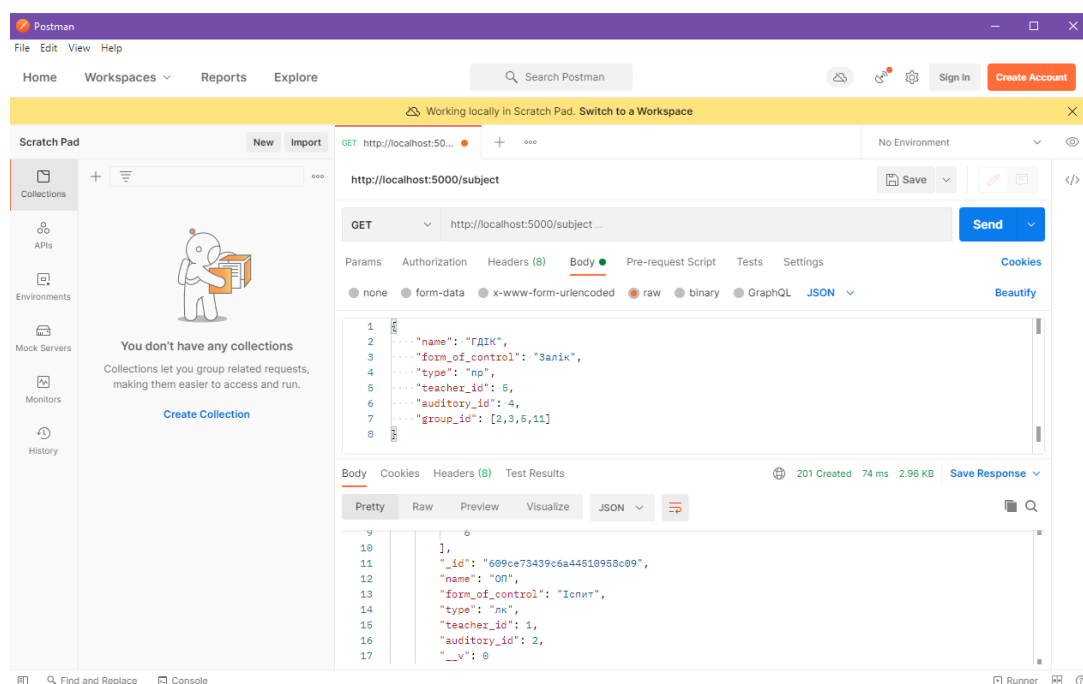


Рисунок 5.4 – Графічний інтерфейс застосунку «Postman»

Далі, треба запустити клієнтську частину. Для цього потрібно відкрити нове вікно у середовищі Visual Studio Code комбінацією клавіш CTRL+SHIFT+N та створити новий термінал. Після цього, треба перейти до папки клієнтської частини виконавши команду `cd .\client\`. Далі, треба виконати команду `npm start` для запуску застосунку. Після успішного запуску, в браузері за замовчування відкриється головна сторінка розробленого веб-застосунку з результатами у вигляді таблиці з розкладом. Головну сторінку веб-застосунку можна побачити на рис. 5.5 та рис. 5.6.

[illegible]

Рисунок 5.5 – Головна сторінка веб-застосунку

[illegible]

Рисунок 5.6 – Головна сторінка веб-застосунку та її «підвал»

5.3 Повідомлення оператора

Веб-застосунок може повідомляти оператора у випадках, які наведені у таблиці 5.2.

Таблиця 5.2 – Повідомлення оператора

Повідомлення	Опис
Server error	Помилка серверу
Server started on port...	Сервер запущений під портом ...

Також, повідомлення про помилки відображаються у консолі браузера, яку можна відкрити при натисканні на клавішу F12.

5.4 Висновки за розділом 5

В даному розділі було описано керівництво оператора для роботи з веб-застосунком, його серверною та клієнтською частиною. Були приведені характеристики системи для запуску веб-застосунку.

Також, було наведено приклади роботи веб-застосунку, інструкція та поради для його запуску у системі оператора.

Веб-застосунок був розроблений на мові програмування JavaScript з використанням платформи Node.js та бібліотек React та Redux. Зберігання інформації було реалізовано у вигляді колекцій нереляційної бази даних MongoDB.

ВИСНОВКИ

При виконанні дипломної кваліфікаційної роботи бакалавра було проаналізовано аналогічні системи вирішення завдання формування розкладу занять, та наведено їх порівняльний аналіз, який показав, що ці аналоги не підходять до нашого випадку.

Проаналізовано основні методи вирішення поставленого завдання та наведено порівняльний аналіз характеристик цих методів, який показав, які алгоритми слід використовувати для вирішення завдання.

Проведено аналіз та вибір інструментарію для розробки програмного продукту, було спроектовано ER-діаграму та структуру майбутнього ПП. Проведено порівняльну характеристику мов програмування та встановлено, що для розробки нашого ПП краще за все підійде мова програмування JavaScript і платформа NodeJS для реалізації серверної частини, та необхідні бібліотеки для реалізації клієнтської частини. Також, було проведено порівняльну характеристику середовищ розробки та встановлено, що для розробки ПП на обраній мові краще за все підійде IDE Visual Studio Code.

Також, було проведено порівняльну характеристику реляційних та нереляційних баз даних, та встановлено, що для реалізації більш за все підійде нереляційна БД, а саме MongoDB.

Проаналізовано вхідні дані у спрощеному вигляді. Також, були проаналізовані на працездатність, точність результатів та швидкість роботи наступні методи: Метод гілок та меж, Метод імітації відпалу, Мурашиний алгоритм, Генетичний алгоритм та Метод розфарбування графа. Для вирішення поставленого завдання було обрано метод розфарбування графа, оскільки він дуже добре показав себе при обробці невеликих масивів даних. Для подальших модифікацій та при зростанні вхідної вибірки, рекомендується вже використовувати генетичний алгоритм.

Розробка та тестування ПП здійснювалось на персональному комп'ютері з наступними характеристиками системи: Операційна система

Windows 10 64-bit, Графічний процесор Intel Core i5, Жорсткий диск з ємкістю у 100 GB, Оперативна пам'ять ємкістю у 8 GB, Швидкість Інтернет-з'єднання 100 Мб.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. FET – Free Timetabling Software [Electronic resource]. – Access mode : <https://lalescu.ro/liviu/fet/>.
2. FET – Download [Electronic resource]. – Access mode : <https://lalescu.ro/liviu/fet/download.html>.
3. FET – Features [Electronic resource]. – Access mode : <https://lalescu.ro/liviu/fet/features.html>.
4. Универсальные программные комплексы [Электрон. ресурс]. – Режим доступа : <http://www.kostanaysoft.com/index/raspisanie/0-94>.
5. Программа составления расписания занятий «Ника» [Электрон. ресурс]. – Режим доступа : <https://allsoft.ua/software/vendors/nika-soft/programma-sostavleniya-raspisaniya-zanyatiy-nika/>.
6. Ника-Люкс – НИКА-софт [Электрон. ресурс]. – Режим доступа : <https://nikasoft.ru/nika-lux/>.
7. 1С:Автоматизированное составление расписания. Университет [Электрон. ресурс]. – Режим доступа : https://solutions.1c.ru/catalog/asp_univer/features.
8. Авторасписание [Электрон. ресурс]. – Режим доступа : <https://www.mmis.ru/programs/avtor>.
9. Мулява, І. А. Система формування розкладу навчального заняття з використанням суб'єктивних переваг [Текст] / І. А. Мулява // Міжнародний науковий журнал. – 2016. – №7. – С. 22– 27.
10. Аналіз методів складання розкладу занять для вдосконалення процесу підготовки льотного складу авіації повітряних сил збройних сил України [Текст] / [П. М. Оніпченко, М. А. Павленко, О. І. Тимочко и др.] // Системи обробки інформації. – 2015. – №12. – С. 199–201.
11. Лопатин А. С. Метод отжига [Текст] / А. С. Лопатин // Метод отжига. – 2005. – С. 133–149.

12. Волков, С. В. Інформаційна система генерації та доступу до розкладу навчальних занять у комп'ютерних класах вищого навчального закладу [Текст] / С. В. Волков, Д. А. Коваленко // Вісник Східноукраїнського національного університету імені Володимира Даля. – 2018. – №6 – С. 31–37.

13. Генетичні алгоритми. Ключові поняття і методи реалізації генетичних алгоритмів [Електрон. ресурс]. – Режим доступу : http://www.znannya.org/?view=ga_general.

14. Субботін С. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: Монографія / С. О. Субботін, А. О. Олійник, О. О. Олійник ; під заг. ред. С.О. Субботіна. – Запоріжжя : ЗНТУ, 2009. – 375 с.

15. Алгоритм раскраски графа – Библиотека алгоритмов на графах [Электрон. ресурс]. – Режим доступа : <http://urban-sanjoo.narod.ru/colours.html>.

16. The Python Tutorial – Python 3.9.5 documentation [Electronic resource]. – Access mode : <https://docs.python.org/3/tutorial/index.html>.

17. Python: преимущества и недостатки языка Python [Электрон. ресурс]. – Режим доступа : <https://blog.ithillel.ua/articles/preimushchestva-i-nedostatki-yazyka-python>.

18. Мова JavaScript та її можливості [Електрон. ресурс]. – Режим доступу : <https://sites.google.com/site/webtehnologiietawebdizajn/mova-javascript-ta-ieie-mozlivosti>.

19. Язык программирования Javascript: особенности и преимущества [Электрон. ресурс]. – Режим доступа : <https://vc.ru/hr/145461-yazyk-programmirovaniya-javascript-osobennosti-i-preimushchestva>.

20. Язык программирования Java: особенности, популярность, ситуация на рынке [Электрон. ресурс]. – Режим доступа : <https://ru.hexlet.io/blog/posts/yazyk-programmirovaniya-java-osobennosti-populyarnost-situatsiya-na-rynke-truda>.

21. Плюсы и минусы программирования на Java [Электрон. ресурс]. – Режим доступа : <https://cutt.ly/JbKj3pq>.

22. C Sharp – Викиучебник [Электрон. ресурс]. – Режим доступа : https://ru.wikibooks.org/wiki/C_Sharp.
23. Языки программирования. C Sharp [Электрон. ресурс]. – Режим доступа : <https://learn-code.ru/yazyki-programmirovaniya/c-sharp>.
24. Эмпатия в программировании [Электрон. ресурс]. – Режим доступа : <https://techrocks.ru/2019/12/05/best-code-editors-for-javascript-developers/>.
25. Python IDE – какой редактор кода выбрать? [Электрон. ресурс]. – Режим доступа : <https://pythonru.com/baza-znaniy/python-ide-kakoj-redaktor-koda-vybrat>.
26. Требования, предъявляемые к базе данных [Электрон. ресурс]. – Режим доступа : <http://bseu.by/it/tohod/lekcii4.htm>.
27. Что такое NoSQL? [Электрон. ресурс]. – Режим доступа : <https://aws.amazon.com/ru/nosql/>.

ДОДАТОК А
Технічне завдання

Вступ

В наш час, проблема складання розкладу занять в навчальних закладах, від шкіл до університетів, має величезну важливість. Без оптимального розкладу неможливо організувати навчання таким чином, щоб викладачам і учням було комфортно і зручно. Складання розкладу вручну займає дуже багато часу та сил, і рідко підсумковий результат буде задовольняти вимогам і бажанням обох сторін. Звідси і виникає потреба автоматизації процесу складання розкладу.

Для складання розкладу потрібно враховувати безліч змінних (об'єктів), таких як: кількість навчальних днів, список викладачів, список предметів, список груп тощо. Завдяки зв'язкам об'єктів один з одним, можна отримати можливість створити програму з прозорою структурою і мінімальним шансом помилки через людський фактор.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломний проєкт бакалавра на тему “Програмна реалізація методів формування розкладу навчальних занять”, затверджене наказом Національного університету «Запорізька політехніка» №131 від 30 березня 2021 року.

A.2 Призначення розробки

Завдання дипломної кваліфікаційної роботи бакалавра:

- аналіз існуючих аналогічних програмних продуктів;
- огляд існуючих методів вирішення завдання;
- вибір середовища для розробки та його аналіз;
- вибір мови програмування та її аналіз;

– аналіз роботи методів вирішення завдання та вибір одного з них для розробки веб-застосунку формування розкладу занять.

А.3 Основні вимоги до програми, яка розробляється

Розроблений веб-застосунок призначений для вирішення задачі формування розкладу занять. Цим веб-застосунком можуть користуватися як викладачі, так і студенти вищих навчальних закладів.

Застосунок призначений для систем с характеристиками, які можна побачити в табл. А.1

Таблиця А.1 – Характеристики системи

Характеристика	Значення
Операційна система	Windows 10 x86/x64
Графічний процесор	Intel Core i5
Жорсткий диск (ємкість)	10 Gb
Оперативна пам'ять (ємкість)	8 Gb
Швидкість Ethernet-з'єднання	10-100 Mb

А.4 Вимоги до програмної документації

Стадії та етапи:

- розбір предметної області в деталях;
- аналіз існуючих аналогів програмного продукту;
- огляд та аналіз існуючих методів вирішення завдання;
- вибір мови програмування для розробки;
- вибір середовища розробки;
- розробка серверної частини веб-застосунку;
- розробка клієнтської частини веб-застосунку;
- нормоконтроль пояснювальної записки;
- захист дипломної роботи.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Назва програмного продукту – програмна реалізація методу розфарбовування графа для вирішення задачі формування розкладу занять, ScheduleProject.

Веб-застосунок був розроблений за допомогою мови програмування JavaScript, платформи Node.js та бібліотек React та Redux.

Б.2 Функціональне призначення

Вхідними даними для веб-застосунку є інформація з БД у вигляді пов'язаних між собою колекцій. Ці дані можна занести до БД за допомогою запитів у якості параметрів яких виступають рядки у форматі JSON (JavaScript Object Notation).

Вихідними даними веб-застосунку є оптимальний розклад занять, який відображається на головній сторінці застосунку у вигляді таблиці.

Б.3 Повідомлення

Під час роботи веб застосунку генеруються повідомлення, які описані в табл.4.2.

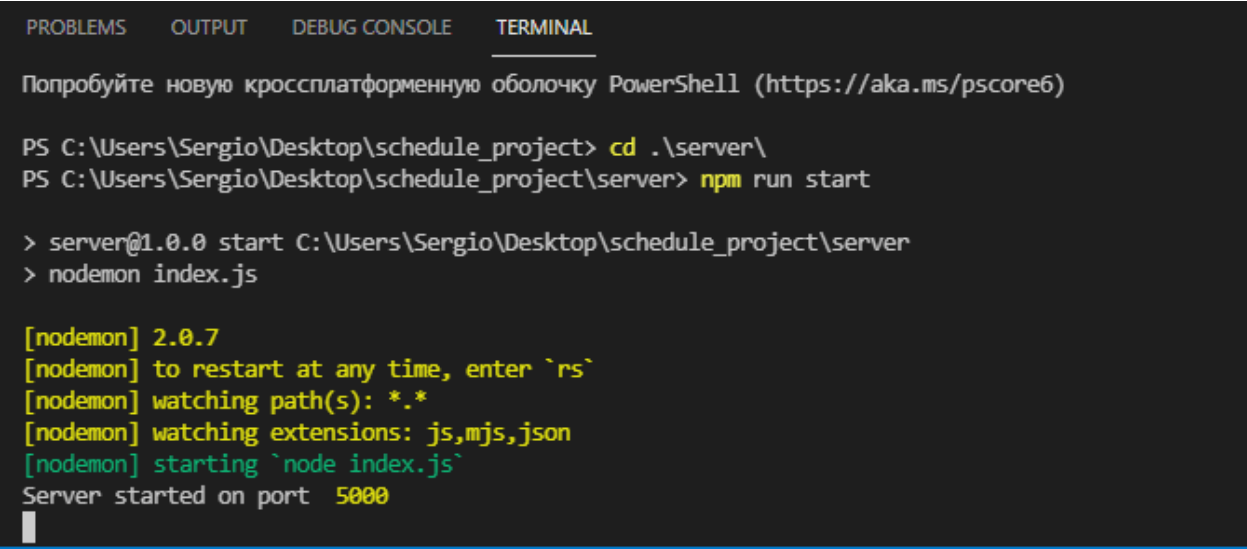
Таблиця 4.2 – Повідомлення та їх опис

Повідомлення	Опис
Server error	Помилка серверу
Server started on port...	Сервер запущений під портом ...

Також, повідомлення про помилки відображаються у консолі браузера, яку можна відкрити при натисканні на клавішу F12.

Б.4 Виконання програмного продукту

Для запуску веб-застосунку треба відкрити проект у середовищі Visual Studio Code. Після чого, створити термінал і ввести у ньому команду `cd .\server\`, для переходу у папку серверу. Для запуску сервера треба в терміналі ввести команду `npm run start`, яка запустить сервер та підключиться до БД MongoDB. Після успішного підключення, застосунок згенерує повідомлення у терміналі «Server started on port 5000». Підключення до серверу можна побачити на рисунку Б.1.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Попробуйте нову кроссплатформенну оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\Sergio\Desktop\schedule_project> cd .\server\
PS C:\Users\Sergio\Desktop\schedule_project\server> npm run start

> server@1.0.0 start C:\Users\Sergio\Desktop\schedule_project\server
> nodemon index.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Server started on port 5000
```

Рисунок Б.1 – Запуск серверу та підключення до БД MongoDB

Щоб побачити, які дані зберігає база даних MongoDB, треба перейти на її офіційний сайт, увійти в свій обліковий запис та перейти до створеного кластера під назвою scheduleDB. Сторінку з кластером можна побачити на рисунку Б.2.

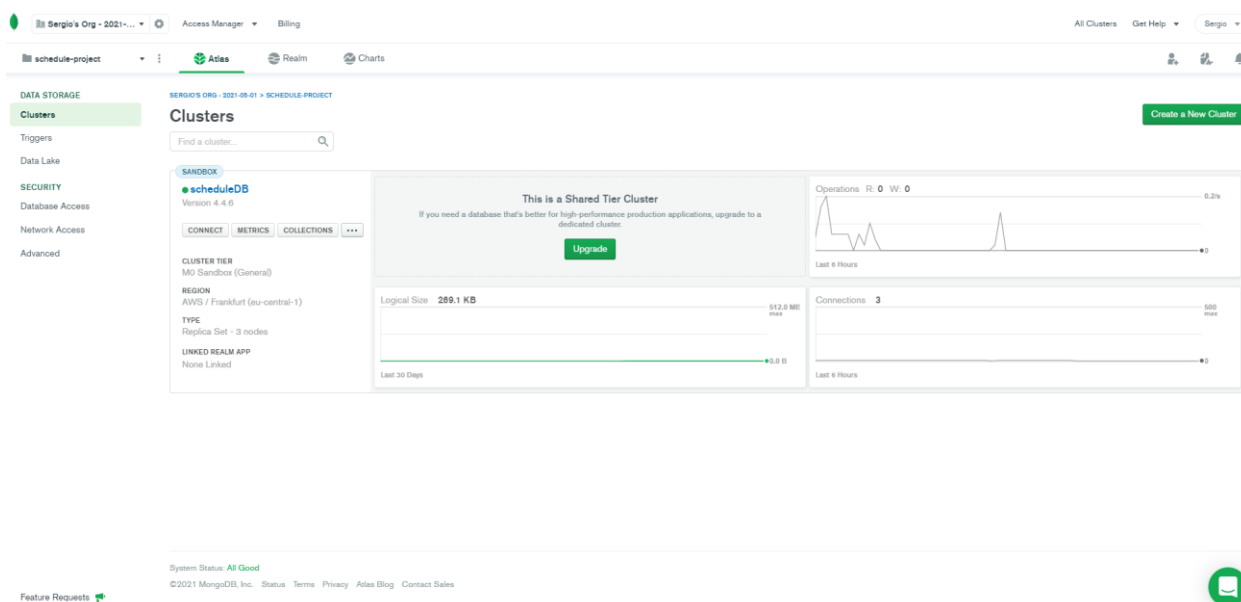


Рисунок Б.2 – Головна сторінка з кластером scheduleDB

Щоб побачити список створених колекцій, треба під назвою кластеру натиснути кнопку «Collections». Список створених колекцій зображений на рисунку Б.3.

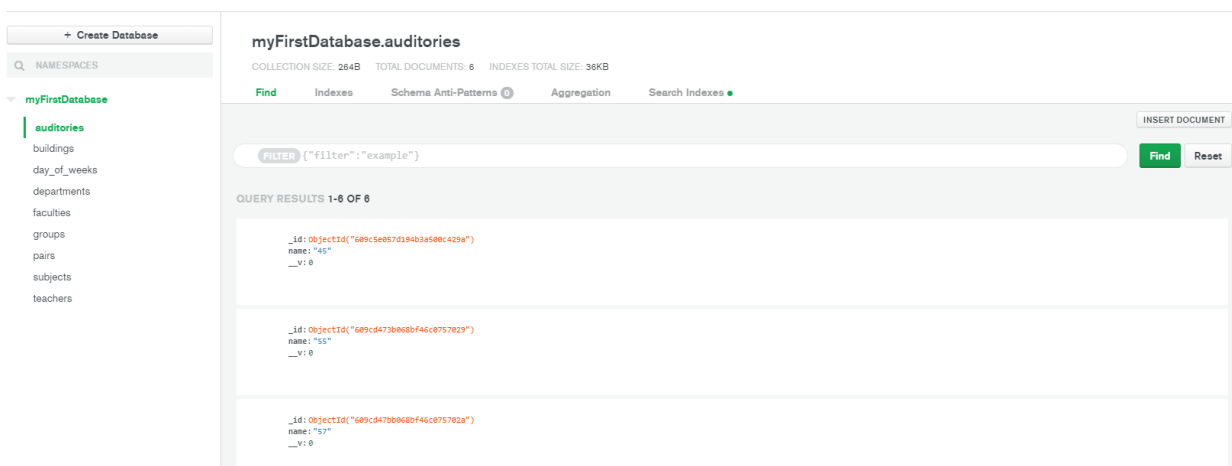


Рисунок Б.3 – Список створених колекцій у кластері scheduleDB

Заповнення бази даних значеннями здійснюється за допомогою застосунку «Postman», який дозволяє імітувати виконання запитів до БД. Графічний інтерфейс застосунку Postman можна побачити на рисунку Б.4

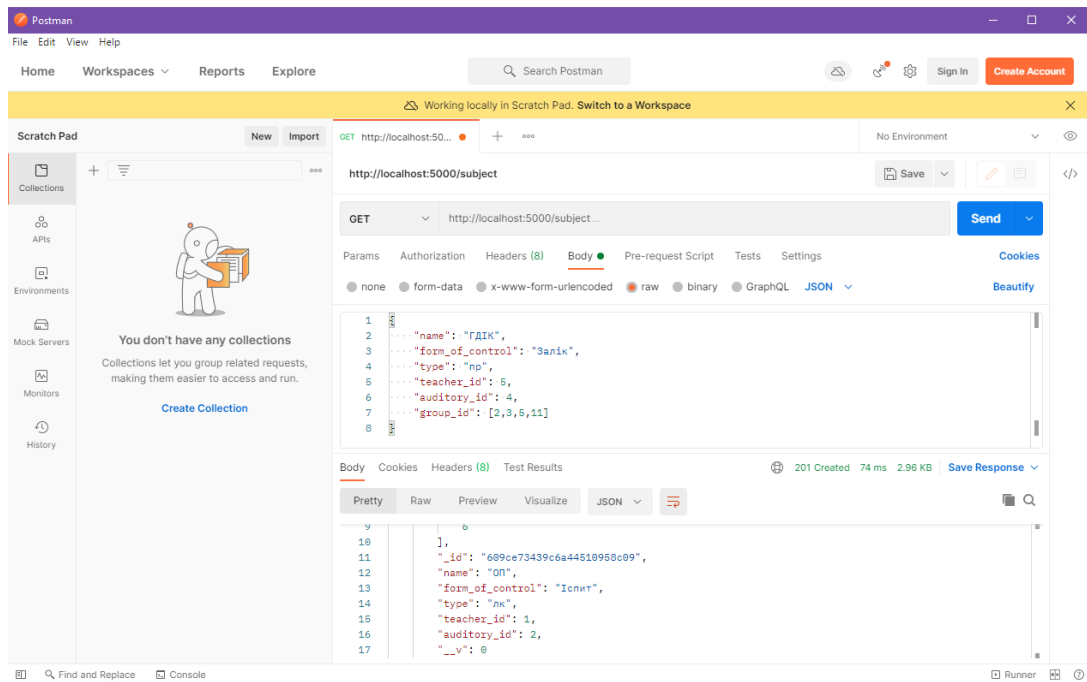


Рисунок Б.4 – Графічний інтерфейс застосунку «Postman»

Далі, треба запустити клієнтську частину. Для цього потрібно відкрити нове вікно у середовищі Visual Studio Code комбінацією клавіш CTRL+SHIFT+N та створити новий термінал. Після цього, треба перейти до папки клієнтської частини виконавши команду `cd .\client\`. Далі, треба виконати команду `npm start` для запуску застосунку. Після успішного запуску, в браузері за замовчування відкриється головна сторінка розробленого веб-застосунку з результатами у вигляді таблиці з розкладом. Головну сторінку веб-застосунку можна побачити на рис. Б.5 та рис. Б.6.

ScheduleProject		Graph coloring method											
		Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7	Group 8	Group 9	Group 10	Group 11	Group 12
Monday	I	ОП [м] Табунцки Г.В. Auditory: 45	ОП [м] Табунцки Г.В. Auditory: 45	ОП [м] Табунцки Г.В. Auditory: 45	ОП [м] Табунцки Г.В. Auditory: 45	ОП [м] Табунцки Г.В. Auditory: 45	ОП [м] Табунцки Г.В. Auditory: 45	ООП [нр] Меронова Н.О. Auditory: 57	ООП [нр] Меронова Н.О. Auditory: 57	ООП [нр] Дейнега Л.Ю. Auditory: 49			
	II						ОП [нр] Дейнега Л.Ю. Auditory: 49						
	III		ГДК [нр] Касрін Д.А. Auditory: 49	ГДК [нр] Касрін Д.А. Auditory: 49		ГДК [нр] Касрін Д.А. Auditory: 49						ГДК [нр] Касрін Д.А. Auditory: 49	
	IV												
	V												
	VI												
Tuesday	I	ОП [нр] Меронова Н.О. Auditory: 57	ОП [нр] Меронова Н.О. Auditory: 57	ОП [нр] Меронова Н.О. Auditory: 57	ОП [нр] Дейнега Л.Ю. Auditory: 49	ОП [нр] Дейнега Л.Ю. Auditory: 49		ООП [м] Табунцки Г.В. Auditory: 55	ООП [м] Табунцки Г.В. Auditory: 55	ООП [м] Табунцки Г.В. Auditory: 55			
	II	ГДК [нр] Касрін Д.А. Auditory: 49											
	III												

Рисунок Б.5 – Головна сторінка веб-застосунку

	III						Auditory: 49						
	IV												
	V												
	VI												
Friday	I	ГДК [м] Льоскін В.М. Auditory: 45	ГДК [м] Льоскін В.М. Auditory: 45	ГДК [м] Льоскін В.М. Auditory: 45	ГДК [м] Льоскін В.М. Auditory: 45	ГДК [м] Льоскін В.М. Auditory: 45	ГДК [м] Льоскін В.М. Auditory: 45		ООП [нр] Меронова Н.О. Auditory: 57	ГДК [м] Льоскін В.М. Auditory: 45	ГДК [м] Льоскін В.М. Auditory: 45		
	II									ГДК [нр] Касрін Д.А. Auditory: 49			
	III												
	IV												
	V												
	VI												
Saturday	I												
	II												
	III												
	IV												
	V												
	VI												

Web-Application "ScheduleProject" was created by Sergio Medvedev
Copyright © Sergio Medvedev 2021. All Rights Reserved.

Рисунок Б.6 – Головна сторінка веб-застосунку та її «підвал»

ДОДАТОК В

Лістинги файлів роботи

В.1 Лістинг файлу index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <title>Schedule Project</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

В.2 Лістинг файлу Content.jsx

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';
import './Content.scss'

const Content = () => {
  const [subjectsList, setSubjectsList] = useState([]);
  useEffect(()=>{
    axios.get('http://localhost:5000/subject').then((resp) => {
      const allPersons = resp.data;
      setSubjectsList(allPersons);
    });
  },[])

  let newSubjectsList = [];
  let newSubjectsTypeList = [];

  subjectsList.map(item =>
    newSubjectsList.push(item.name)
```

```

)

subjectsList.map(item =>
  newSubjectsTypeList.push(item.type)
)

const [daysList, setDayList] = useState([]);
useEffect(()=>{
  axios.get('http://localhost:5000/day').then((resp) => {
    const allPersons = resp.data;
    setDayList(allPersons);
  });
},[])

let newDaysList = [];

daysList.map(item =>
  newDaysList.push(item.name)
)

const [pairsList, setPairList] = useState([]);
useEffect(()=>{
  axios.get('http://localhost:5000/pair').then((resp) => {
    const allPersons = resp.data;
    setPairList(allPersons);
  });
},[])

let newPairsList = [];

pairsList.map(item =>
  newPairsList.push(item.name)
)

const [teachersList, setTeacherList] = useState([]);
useEffect(()=>{
  axios.get('http://localhost:5000/teacher').then((resp) => {
    const allPersons = resp.data;
    setTeacherList(allPersons);
  });
},[])

```

```

let newTeachersList = [];

teachersList.map(item =>
  newTeachersList.push(item.name)
)

const [auditoriesList, setAuditoryList] = useState([]);
useEffect(()=>{
  axios.get('http://localhost:5000/auditory').then((resp) => {
    const allPersons = resp.data;
    setAuditoryList(allPersons);
  });
},[])

let newAuditoriesList = [];

auditoriesList.map(item =>
  newAuditoriesList.push(item.name)
)

// console.log(newSubjectsList)
// console.log(newAuditoriesList)
// console.log(newTeachersList)

return (

<div className="App">
  <div className="wrapper">
    <table className="brd">
      <tr>
        <th colSpan="2"></th>
        <th>Group 1</th>
        <th>Group 2</th>
        <th>Group 3</th>
        <th>Group 4</th>
        <th>Group 5</th>
        <th>Group 6</th>
        <th>Group 7</th>
        <th>Group 8</th>
        <th>Group 9</th>
        <th>Group 10</th>
        <th>Group 11</th>

```

```

    <th>Group 12</th>
</tr>
<tr>
    <td rowspan="7"><b>{newDaysList[0]}</b></td>
</tr>
<tr>
    <td>{newPairsList[0]}</td>
    <td>
        <p>{newSubjectsList[0]} [{newSubjectsTypeList[0]}]</p>
        <p>{newTeachersList[0]}</p>
        <p>Auditory: {newAuditoriesList[0]}</p>
    </td>
    <td>
        <p>{newSubjectsList[0]} [{newSubjectsTypeList[0]}]</p>
        <p>{newTeachersList[0]}</p>
        <p>Auditory: {newAuditoriesList[0]}</p>
    </td>
    <td>
        <p>{newSubjectsList[0]} [{newSubjectsTypeList[0]}]</p>
        <p>{newTeachersList[0]}</p>
        <p>Auditory: {newAuditoriesList[0]}</p>
    </td>
    <td>
        <p>{newSubjectsList[0]} [{newSubjectsTypeList[0]}]</p>
        <p>{newTeachersList[0]}</p>
        <p>Auditory: {newAuditoriesList[0]}</p>
    </td>
    <td>
        <p>{newSubjectsList[0]} [{newSubjectsTypeList[0]}]</p>
        <p>{newTeachersList[0]}</p>
        <p>Auditory: {newAuditoriesList[0]}</p>
    </td>
    <td>
        <p>{newSubjectsList[0]} [{newSubjectsTypeList[0]}]</p>
        <p>{newTeachersList[0]}</p>
        <p>Auditory: {newAuditoriesList[0]}</p>
    </td>
    <td>
        <p>{newSubjectsList[8]} [{newSubjectsTypeList[8]}]</p>
        <p>{newTeachersList[1]}</p>
        <p>Auditory: {newAuditoriesList[2]}</p>
    </td>

```

```

<td>
  <p>{newSubjectsList[8]} [{newSubjectsTypeList[8]]}</p>
  <p>{newTeachersList[1]}</p>
  <p>Auditory: {newAuditoriesList[2]}</p>
</td>
<td>
  <p>{newSubjectsList[11]} [{newSubjectsTypeList[11]]}</p>
  <p>{newTeachersList[2]}</p>
  <p>Auditory: {newAuditoriesList[3]}</p>
</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
  <td>{newPairsList[1]}</td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td>
    <p>{newSubjectsList[6]} [{newSubjectsTypeList[6]]}</p>
    <p>{newTeachersList[2]}</p>
    <p>Auditory: {newAuditoriesList[3]}</p>
  </td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
</tr>
<tr>
  <td>{newPairsList[2]}</td>
  <td></td>
  <td>
    <p>{newSubjectsList[17]} [{newSubjectsTypeList[17]]}</p>
    <p>{newTeachersList[4]}</p>
    <p>Auditory: {newAuditoriesList[3]}</p>
  </td>
  <td>

```

```

        <p>{newSubjectsList[17]} [{newSubjectsTypeList[17]}]</p>
        <p>{newTeachersList[4]}</p>
        <p>Auditory: {newAuditoriesList[3]}</p>
    </td>
</td>
<td>
        <p>{newSubjectsList[17]} [{newSubjectsTypeList[17]}]</p>
        <p>{newTeachersList[4]}</p>
        <p>Auditory: {newAuditoriesList[3]}</p>
    </td>
</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>
        <p>{newSubjectsList[17]} [{newSubjectsTypeList[17]}]</p>
        <p>{newTeachersList[4]}</p>
        <p>Auditory: {newAuditoriesList[3]}</p>
    </td>
</td>
</tr>
<tr>
    <td>{newPairsList[3]}</td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    </tr>
<tr>
    <td>{newPairsList[4]}</td>
    <td></td>
    <td></td>
    <td></td>

```

	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
</tr>	
<tr>	
	<td>{newPairsList[5]}</td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
	<td></td>
</tr>	
<tr>	
	<td rowSpan="7">{ newDaysList[1]}</td>
</tr>	
<tr>	
	<td>{newPairsList[0]}</td>
	<td>
	{newSubjectsList[1]} [{newSubjectsTypeList[1]}]
	{newTeachersList[1]}
	Auditory: {newAuditoriesList[2]}
	</td>
	<td>
	{newSubjectsList[1]} [{newSubjectsTypeList[1]}]
	{newTeachersList[1]}
	Auditory: {newAuditoriesList[2]}
	</td>
	<td>
	{newSubjectsList[1]} [{newSubjectsTypeList[1]}]

```

        <p>{newTeachersList[1]}</p>
        <p>Auditory: {newAuditoriesList[2]}</p>
    </td>
    <td>
        <p>{newSubjectsList[4]} [{newSubjectsTypeList[4]}]</p>
        <p>{newTeachersList[2]}</p>
        <p>Auditory: {newAuditoriesList[3]}</p>
    </td>
    <td>
        <p>{newSubjectsList[4]} [{newSubjectsTypeList[4]}]</p>
        <p>{newTeachersList[2]}</p>
        <p>Auditory: {newAuditoriesList[3]}</p>
    </td>
    <td></td>
    <td>
        <p>{newSubjectsList[7]} [{newSubjectsTypeList[7]}]</p>
        <p>{newTeachersList[0]}</p>
        <p>Auditory: {newAuditoriesList[1]}</p>
    </td>
    <td>
        <p>{newSubjectsList[7]} [{newSubjectsTypeList[7]}]</p>
        <p>{newTeachersList[0]}</p>
        <p>Auditory: {newAuditoriesList[1]}</p>
    </td>
    <td>
        <p>{newSubjectsList[7]} [{newSubjectsTypeList[7]}]</p>
        <p>{newTeachersList[0]}</p>
        <p>Auditory: {newAuditoriesList[1]}</p>
    </td>
    <td></td>
    <td></td>
    <td></td>
</tr>
<tr>
    <td>{newPairsList[1]}</td>
    <td>
        <p>{newSubjectsList[13]} [{newSubjectsTypeList[13]}]</p>
        <p>{newTeachersList[4]}</p>
        <p>Auditory: {newAuditoriesList[3]}</p>
    </td>
    <td></td>
    <td></td>

```

```

        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
    </tr>
    <tr>
        <td>{newPairsList[2]}</td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
    </tr>
    <tr>
        <td>{newPairsList[3]}</td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
    </tr>
    <tr>
        <td>{newPairsList[4]}</td>

```

```

<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{newPairsList[5]}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td rowspan="7"><b>{newDaysList[2]}</b></td>
</tr>
<tr>
<td>{newPairsList[0]}</td>
<td>
<p>{newSubjectsList[2]} [{newSubjectsTypeList[2]}]</p>
<p>{newTeachersList[2]}</p>
<p>Auditory: {newAuditoriesList[3]}</p>
</td>
<td>
<p>{newSubjectsList[2]} [{newSubjectsTypeList[2]}]</p>
<p>{newTeachersList[2]}</p>
<p>Auditory: {newAuditoriesList[3]}</p>

```

[illegible]

[illegible]

```

<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{newPairsList[5]}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td rowspan="7"><b>{newDaysList[3]}</b></td>
</tr>
<tr>
<td>{newPairsList[0]}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>
<p>{newSubjectsList[5]} [{newSubjectsTypeList[5]}]</p>
<p>{newTeachersList[1]}</p>
<p>Auditory: {newAuditoriesList[2]}</p>
</td>
<td>
<p>{newSubjectsList[9]} [{newSubjectsTypeList[9]}]</p>
<p>{newTeachersList[2]}</p>
<p>Auditory: {newAuditoriesList[3]}</p>
</td>
<td>
<p>{newSubjectsList[9]} [{newSubjectsTypeList[9]}]</p>
<p>{newTeachersList[2]}</p>

```



```

        <td></td>
    </tr>
    <tr>
        <td rowspan="7"><b>{newDaysList[4]}</b></td>
    </tr>
    <tr>
        <td>{newPairsList[0]}</td>
        <td>
            <p>{newSubjectsList[12]} [{newSubjectsTypeList[12]}]</p>
            <p>{newTeachersList[3]}</p>
            <p>Auditory: {newAuditoriesList[0]}</p>
        </td>
        <td>
            <p>{newSubjectsList[12]} [{newSubjectsTypeList[12]}]</p>
            <p>{newTeachersList[3]}</p>
            <p>Auditory: {newAuditoriesList[0]}</p>
        </td>
        <td>
            <p>{newSubjectsList[12]} [{newSubjectsTypeList[12]}]</p>
            <p>{newTeachersList[3]}</p>
            <p>Auditory: {newAuditoriesList[0]}</p>
        </td>
        <td>
            <p>{newSubjectsList[12]} [{newSubjectsTypeList[12]}]</p>
            <p>{newTeachersList[3]}</p>
            <p>Auditory: {newAuditoriesList[0]}</p>
        </td>
        <td>
            <p>{newSubjectsList[12]} [{newSubjectsTypeList[12]}]</p>
            <p>{newTeachersList[3]}</p>
            <p>Auditory: {newAuditoriesList[0]}</p>
        </td>
        <td>
            <p>{newSubjectsList[12]} [{newSubjectsTypeList[12]}]</p>
            <p>{newTeachersList[3]}</p>
            <p>Auditory: {newAuditoriesList[0]}</p>
        </td>
        <td>
            <p>{newSubjectsList[12]} [{newSubjectsTypeList[12]}]</p>
            <p>{newTeachersList[3]}</p>
            <p>Auditory: {newAuditoriesList[0]}</p>
        </td>
        <td>
            <p>{newSubjectsList[12]} [{newSubjectsTypeList[12]}]</p>
            <p>{newTeachersList[3]}</p>
            <p>Auditory: {newAuditoriesList[0]}</p>
        </td>
        <td>
            <p>{newSubjectsList[10]} [{newSubjectsTypeList[10]}]</p>
            <p>{newTeachersList[1]}</p>

```



```

        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
    </tr>
    <tr>
        <td>{newPairsList[3]}</td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
    </tr>
    <tr>
        <td>{newPairsList[4]}</td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
    </tr>
    <tr>
        <td>{newPairsList[5]}</td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
    </tr>

```


[illegible]

```

        <td></td>
        <td></td>
    </tr>
    <tr>
        <td>{newPairsList[5]}</td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
    </tr>
</table>
</div>
<div>
</div>
<div className="footer">
    Web-Application "ScheduleProject" was created by Sergio Medvedev <br/>
    Copyright © Sergio Medvedev 2021. All Rights Reserved.
</div>
</div>
);
}

```

```
export default Content;
```

В.3 Лістинг файлу Content.scss

```

.App{
    display: flex;
    flex-direction: column;
    background-color: rgb(255, 255, 255);

    .header{
        display: flex;
    }
}

```

```
justify-content: space-between;
background-color: rgb(173, 231, 159);
padding: 0px 10px 0px 10px;
height: 80px;
font-size: 30px;
border-radius: 20px;

.header-logo{
  cursor: pointer;
  display: flex;
  align-items: center;
}

.header-auth{
  display: flex;
  flex-direction: row;
  align-items: center;
  text-shadow: black;

  .header-auth-username{
    color: crimson;
  }
}

.wrapper{
  height: auto;
  display: flex;
  justify-content: center;
  .shedule-subject {
    border: 1px solid rgb(73, 139, 106);
  }
}

.footer{
  background-color: rgb(182, 155, 155);
  color: black;
  height: 100px;
  display: flex;
  justify-content: center;
  align-items: center;
```

```

        text-align: center;
        border-radius: 20px;
    }
}

// table {
//     border-collapse: collapse;
//     margin-top: 40px;
// }
// table th,
// table td {
//     padding: 0 3px;
// }
// table.brd th,
// table.brd td {
//     border: 1px solid #000;
// }
// .bot {
//     border-bottom: 2px solid black;
// }

table {
    font-family: "Lucida Sans Unicode", "Lucida Grande", Sans-Serif;
    font-size: 12px;
    border-radius: 10px;
    border-spacing: 0;
    text-align: center;
}
th {
    background: #4355bd;
    color: white;
    text-shadow: 0 1px 1px #2D2020;
    padding: 10px 20px;
}
th, td {
    border-style: solid;
    border-width: 0 1px 1px 0;
    border-color: white;
}
th:first-child, td:first-child {
    text-align: left;
}

```

```

th:first-child {
border-top-left-radius: 10px;
}
th:last-child {
border-top-right-radius: 10px;
}
td {
padding: 10px 20px;
background: #F8E391;
}
tr:last-child td:first-child {
border-radius: 0 0 0 10px;
}
tr:last-child td:last-child {
border-radius: 0 0 10px 0;
}

b {
font-size: large;
}

```

В.4 Лістинг файлу Forms.jsx

```

import React from 'react';

const Forms = () => {
  return (
    <div className="Forms">
      ABOBA
    </div>
  );
};

export default Forms;

```

В.5 Лістинг файлу reduces\index.js

```

import {applyMiddleware, combineReducers, createStore} from 'redux';
import {composeWithDevTools } from 'redux-devtools-extension';

```

```
import scheduleReducer from "./subjectReducer";
import thunk from 'redux-thunk';

const rootReducer = combineReducers({
  subjects: scheduleReducer,
});

export const store = createStore(rootReducer, composeWithDevTools(applyMiddleware(thunk)))
```

В.6 Лістинг файлу subjectReducer.js

```
const defaultState = {

}

export default function scheduleReducer(state = defaultState, action) {
  switch(action.type) {

    default:
      return state
  }
}
```

В.7 Лістинг файлу App.jsx

```
import React from 'react';
import Content from './components/content/Content';
import {BrowserRouter, Redirect, Route, Switch} from 'react-router-dom';
import Forms from './components/forms/Forms';
import Logo from "./assets/img/calendar.png";
import styled from "styled-components";

const StyledImage = styled.img`
  width: 64px;
  height: 64px;
`;

const StyledHeaderName = styled.div`
  margin-left: 20px;
```

```
`;

const StyledFormName = styled.div`
  margin-right: 20px;
`;

const App = () => {
  return (
    <div className="App">
      <div className="header">
        <div className="header-logo">
          <StyledImage src={Logo} alt="logo"/>
          <StyledHeaderName>ScheduleProject</StyledHeaderName>
        </div>

        <div className="header-auth">
          <StyledFormName>Graph coloring method</StyledFormName>
        </div>
      </div>
      <BrowserRouter>
        <Switch>
          <Route path = "/"content" component = {Content} />
          <Route path = "/"forms" component = {Forms} />
          <Redirect to = "/"content" />
        </Switch>
      </BrowserRouter>
    </div>
  );
};

export default App;
```

В.8 Лістинг файлу index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import {store} from './reduces'
import {Provider} from 'react-redux'

ReactDOM.render(
```

```

    <Provider store={store}>
      <App />
    </Provider>,

    document.getElementById('root')
  );

```

В.9 Лістинг файлу client\package.json

```

{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.12.0",
    "@testing-library/react": "^11.2.6",
    "@testing-library/user-event": "^12.8.3",
    "less": "^4.1.1",
    "node-sass": "^4.14.1",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-redux": "^7.2.4",
    "react-scripts": "4.0.3",
    "redux": "^4.1.0",
    "redux-devtools-extension": "^2.13.9",
    "redux-thunk": "^2.3.0",
    "web-vitals": "^1.1.2"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  }
}

```

```

},
"browserslist": {
  "production": [
    ">0.2%",
    "not dead",
    "not op_mini all"
  ],
  "development": [
    "last 1 chrome version",
    "last 1 firefox version",
    "last 1 safari version"
  ]
}
}

```

B.10 Лістинг файлу `auditory.model.js`

```

const {Schema, model, ObjectId} = require("mongoose");

const Auditory = new Schema ({
  name: {type: String, required: true},
  building_id: {type: ObjectId, ref: "Building"}
})

module.exports = model('Auditory', Auditory)

```

B.11 Лістинг файлу `building.model.js`

```

const {Schema, model, ObjectId} = require("mongoose");

const Building = new Schema ({
  name: {type: String, required: true}
})

module.exports = model('Building', Building)

```

B.12 Лістинг файлу `day_of_week.model.js`

```

const {Schema, model, ObjectId} = require("mongoose");

```

```
const Day_of_Week = new Schema ({
  name: {type: String, required: true},
  schedule_id: {type: ObjectId, ref: "Schedule"}
})

module.exports = model('Day_of_Week', Day_of_Week)
```

В.13 Лістинг файлу department.model.js

```
const {Schema, model, ObjectId} = require("mongoose");

const Department = new Schema ({
  name: {type: String, required: true},
  faculty_id: {type: ObjectId, ref: "Faculty"}
})

module.exports = model('Department', Department)
```

В.14 Лістинг файлу faculty.model.js

```
const {Schema, model, ObjectId} = require("mongoose");

const Faculty = new Schema ({
  name: {type: String, required: true}
})

module.exports = model('Faculty', Faculty)
```

В.15 Лістинг файлу group.model.js

```
const {Schema, model, ObjectId} = require("mongoose");

const Group = new Schema ({
  name: {type: String, required: true},
  course_num: {type: Number, required: true},
  department_id: {type: ObjectId, ref: "Subject"}
})
```

```
module.exports = model('Group', Group)
```

В.16 Лістинг файлу pair.model.js

```
const {Schema, model, ObjectId} = require("mongoose");
```

```
const Pair = new Schema ({  
  name: {type: String, required: true},  
  day_id: {type: ObjectId, ref: "Day_of_Week"}  
})
```

```
module.exports = model('Pair', Pair)
```

В.17 Лістинг файлу subject.model.js

```
const {Schema, model, ObjectId} = require("mongoose");
```

```
const Subject = new Schema ({  
  name: {type: String, required: true},  
  form_of_control: {type: String, required: true},  
  type: {type: String, required: true},  
  teacher_id: {type: Number, required: true},  
  auditory_id: {type: Number, required: true},  
  group_id: [{type: Number, required: true}]  
})
```

```
module.exports = model('Subject', Subject)
```

В.18 Лістинг файлу teacher.model.js

```
const {Schema, model, ObjectId} = require("mongoose");
```

```
const Teacher = new Schema ({  
  name: {type: String, required: true},  
  post: {type: String, required: true},  
  degree: {type: String, required: true},  
  department_id: {type: ObjectId, ref: "Department"}
```

```
})
```

```
module.exports = model('Teacher', Teacher)
```

В.19 Лістинг файлу `auditory.routes.js`

```
const Router = require("express");
const router = new Router();
const Auditory = require("../models/auditory.model");
```

```
router.get('/', async (req, res) => {
  try {
    let auditories = await Auditory.find();
    res.status(201).json(auditories);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})
```

```
router.post('/', async (req, res) => {
  const { name } = req.body
  const newAuditory= new Auditory({ name })
  try {
    await newAuditory.save();
    res.status(201).json(newAuditory);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})
```

```
module.exports = router
```

В.20 Лістинг файлу `building.routes.js`

```
const Router = require("express");
const router = new Router();
const Building = require("../models/building.model");
```

```

router.get('/', async (req, res) => {
  try {
    let buildings = await Building.find();
    res.status(201).json(buildings);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

router.post('/', async (req, res) => {
  const { name } = req.body
  const newBuilding = new Building({ name })
  try {
    await newBuilding.save();
    res.status(201).json(newBuilding);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

module.exports = router

```

В.21 Лістинг файлу day_of_week.router.js

```

const Router = require("express");
const router = new Router();
const Day_of_Week = require("../models/day_of_week.model");

router.get('/', async (req, res) => {
  try {
    let days = await Day_of_Week.find();
    res.status(201).json(days);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
}

```

```

    })

    router.post('/', async (req, res) => {
      const { name } = req.body
      const newDay = new Day_of_Week({ name })
      try {
        await newDay.save();
        res.status(201).json(newDay);
      } catch (e) {
        console.log(e)
        res.send({message: "Server error"})
      }
    })

    module.exports = router

```

В.22 Лістинг файлу department.routes.js

```

const Router = require("express");
const router = new Router();
const Department = require("../models/department.model");

router.get('/', async (req, res) => {
  try {
    let departments = await Department.find();
    res.status(201).json(departments);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

router.post('/', async (req, res) => {
  const { name } = req.body
  const newDepartment = new Department({ name })
  try {
    await newDepartment.save();
    res.status(201).json(newDepartment);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

```

```

    }
  })

  module.exports = router

```

В.23 Лістинг файлу `faculty.routes.js`

```

const Router = require("express");
const router = new Router();
const Faculty = require("../models/faculty.model");

router.get('/', async (req, res) => {
  try {
    let faculties = await Faculty.find();
    res.status(201).json(faculties);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

router.post('/', async (req, res) => {
  const { name } = req.body
  const newFaculty = new Faculty({ name })
  try {
    await newFaculty.save();
    res.status(201).json(newFaculty);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

module.exports = router

```

В.24 Лістинг файлу `group.routes.js`

```

const Router = require("express");
const router = new Router();
const Group = require("../models/group.model");

```

```

router.get('/', async (req, res) => {
  try {
    let groups = await Group.find();
    res.status(201).json(groups);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

router.post('/', async (req, res) => {
  const { name, course_num } = req.body
  const newGroup = new Group({ name, course_num })
  try {
    await newGroup.save();
    res.status(201).json(newGroup);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

module.exports = router

```

B.25 Лістинг файлу pair.routes.js

```

const Router = require("express");
const router = new Router();
const Pair = require("../models/pair.model");

router.get('/', async (req, res) => {
  try {
    let pairs = await Pair.find();
    res.status(201).json(pairs);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

```

```

router.post('/', async (req, res) => {
  const { name } = req.body
  const newPair = new Pair({ name })
  try {
    await newPair.save();
    res.status(201).json(newPair);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

```

```

module.exports = router

```

В.26 Лістинг файлу subject.routes.js

```

const Router = require("express");
const router = new Router();
const Subject = require("../models/subject.model");

```

```

router.get('/', async (req, res) => {
  try {
    let subjects = await Subject.find();
    res.status(201).json(subjects);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

```

```

router.post('/', async (req, res) => {
  const { name, form_of_control, type, teacher_id, auditory_id, group_id } = req.body
  const newSubject = new Subject({ name, form_of_control, type, teacher_id, auditory_id, group_id })
  try {
    await newSubject.save();
    res.status(201).json(newSubject);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

```

```
module.exports = router
```

В.27 Лістинг файлу teacher.routes.js

```
const Router = require("express");
const router = new Router();
const Teacher = require("../models/teacher.model");

router.get('/', async (req, res) => {
  try {
    let teachers = await Teacher.find();
    res.status(201).json(teachers);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

router.post('/', async (req, res) => {
  const { name, post, degree } = req.body
  const newTeacher = new Teacher({ name, post, degree })
  try {
    await newTeacher.save();
    res.status(201).json(newTeacher);
  } catch (e) {
    console.log(e)
    res.send({message: "Server error"})
  }
})

module.exports = router
```

В.28 Лістинг файлу server\index.js

```
const express = require("express")
const mongoose = require("mongoose")
const config = require('config')
const buildingRouter = require('../routes/building.routes')
```

```

const auditoryRouter = require('./routes/auditory.routes')
const facultyRouter = require('./routes/faculty.routes')
const dayRouter = require('./routes/day_of_week.routes')
const departmentRouter = require('./routes/department.routes')
const groupRouter = require('./routes/group.routes')
const pairRouter = require('./routes/pair.routes')
const teacherRouter = require('./routes/teacher.routes')
const subjectRouter = require('./routes/subject.routes')
const cors = require("cors")

```

```

const app = express()
const PORT = config.get("serverPort")

```

```

app.use(express.json())
app.use(cors())
app.use("/building", buildingRouter)
app.use("/auditory", auditoryRouter)
app.use("/faculty", facultyRouter)
app.use("/day", dayRouter)
app.use("/department", departmentRouter)
app.use("/group", groupRouter)
app.use("/pair", pairRouter)
app.use("/teacher", teacherRouter)
app.use("/subject", subjectRouter)

```

```

const start = async () => {
  try {
    await mongoose.connect(config.get("dbUrl"), {
      useNewUrlParser:true,
      useUnifiedTopology:true
    })

    app.listen(PORT, () => {
      console.log('Server started on port ', PORT)
    })
  } catch (error) {
    console.log(error)
  }
}

```

```

start()

```

В.29 Лістинг файлу proba.js

```

const csv = require('csv-parser');
const fs = require('fs')
const createCsvWriter = require('csv-writer').createObjectCsvWriter;

function createArray(length) {
    var arr = new Array(length || 0),
        i = length;

    if (arguments.length > 1) {
        var args = Array.prototype.slice.call(arguments, 1);
        while(i--) arr[length-1 - i] = createArray.apply(this, args);
    }

    return arr;
}

class Lesson {

    constructor(id, teacher_id, group_id, room_id, name, type, color) {
        this.id = id;
        this.teacher_id = teacher_id;
        this.group_id = group_id;
        this.room_id = room_id;
        this.name = name;
        this.type = type;
        this.color = color;
    }
}

let n = 18;
let sub_1 = new Lesson(1, 1, [1,2,3,4,5,6], [1], "OOP_1", 1, 0);
let sub_2 = new Lesson(1, 2, [1,2,3], [3], "OOP_1", 0, 0);
let sub_3 = new Lesson(1, 3, [1,2,3], [4], "OOP_1", 0, 0);
let sub_4 = new Lesson(1, 2, [4,5], [3], "OOP_1", 0, 0);
let sub_5 = new Lesson(1, 3, [4,5], [4], "OOP_1", 0, 0);
let sub_6 = new Lesson(1, 2, [6], [3], "OOP_1", 0, 0);
let sub_7 = new Lesson(1, 3, [6], [4], "OOP_1", 0, 0);
let sub_8 = new Lesson(2, 1, [7,8,9], [2], "OOP_2", 1, 0);

```

```

let sub_9 = new Lesson(2, 2, [7,8], [3], "OOP_2", 0, 0);
let sub_10 = new Lesson(2, 3, [7,8], [4], "OOP_2", 0, 0);
let sub_11 = new Lesson(2, 2, [9], [3], "OOP_2", 0, 0);
let sub_12 = new Lesson(2, 3, [9], [4], "OOP_2", 0, 0);
let sub_13 = new Lesson(3, 4, [1,4,6,10,11,2,5,3], [1], "GDIK", 1, 0);
let sub_14 = new Lesson(3, 5, [1], [4], "GDIK", 0, 0);
let sub_15 = new Lesson(3, 5, [4], [4], "GDIK", 0, 0);
let sub_16 = new Lesson(3, 5, [6], [4], "GDIK", 0, 0);
let sub_17 = new Lesson(3, 5, [10], [4], "GDIK", 0, 0);
let sub_18 = new Lesson(3, 5, [11,2,3,5], [4], "GDIK", 0, 0);
mas =
[sub_1,sub_2,sub_3,sub_4,sub_5,sub_6,sub_7,sub_8,sub_9,sub_10,sub_11,sub_12,sub_13,sub_14,sub_15,sub_16,sub_17,sub_18];

```

```

arr = createArray(n*n);
arr.fill(0);
arr_pol = arr.length/n;
// console.log(arr_pol);

let matrix = []
let helpa = 0;
let zero = 0;
// Собираем матрицу смежности
for(let i = 0; i < arr_pol; i++) {
  for(let j = 0; j < arr_pol; j++) {
    if (i != j) {
      if (mas[i].teacher_id == mas[j].teacher_id || mas[i].room_id[0] == mas[j].room_id[0]){
        helpa = 1;
        matrix.push(helpa);
      } else {
        for (let k = 0; k < mas[i].group_id.length; k++) {
          for (let f = 0; f < mas[j].group_id.length; f++){
            if (mas[i].group_id[k] == mas[j].group_id[f]) {
              zero = 1;
            } else {
              zero = 0;
            }
          }
          if (zero == 1){
            break;
          }
        }
        if (zero == 1){

```

```

        break;
    }
}
helpa = zero;
matrix.push(helpa);
}
} else if (i == j) {
    helpa = 0;
    matrix.push(helpa);
}
}
}

fs.writeFileSync("matrix.txt", matrix.toString(), "ascii");

console.log(matrix);

//Преобразование массива в двумерный
const chunkArray = (ar, cnt) => ar.reduce((prev, cur, i, a) => !(i % cnt) ? prev.concat([a.slice(i, i + cnt)]) :
prev, []);

let matrix2 = chunkArray(matrix, arr_pol);

let matrix_test = [
    0,1,0,0,0,
    1,0,1,0,0,
    0,1,0,1,0,
    0,0,1,0,1,
    0,0,0,1,0
]

function prove (color_v, stroka) {
    for (let i = 0; i < color_v.length; i++) {
        //console.log(color_v[i].toString());
        if (color_v[i].toString() == stroka.toString()) {
            return true;
        }
    }
    return false
}

let test = 5;
let matrix3 = chunkArray(matrix_test, test);

```

```

//Присваиваем цвета занятиям (раскраска графа по матрице смежности)
let cc = 0;
let cc_pred = 0;
let color_v = [];
let c_helpa = 0;
let c_lenght = 0;
let dop_mas = []
for(let i = 0; i < arr_pol; i++) {
  cc_pred = cc;
  cc += 1;
  for(let j = 0; j < arr_pol; j++) {
    if (matrix2[i][j] == 0) {
      if (mas[j].color == 0) {
        if (color_v.length == 0) {
          mas[j].color = cc;
          color_v.push([j, mas[j].color]);
          console.log("massiv: "+color_v)
        } else {
          c_lenght = color_v.length;
          for (let k = 0; k < c_lenght; k++) {
            console.log("error: "+ color_v[k][0])
            c_helpa = color_v[k][0];
            if (matrix2[j][c_helpa] == 0) { //mas[c_helpa].color != cc
              mas[j].color = cc;
              proverka = prove(color_v,[j, mas[j].color]);
              if (proverka == false) {
                color_v.push([j, mas[j].color]);
              }
              //console.log("Length: "+color_v.length)
              //continue;
            } else if (matrix2[j][c_helpa] == 1) {
              if (mas[c_helpa].color != cc && color_v[k][1] != cc) {
                mas[j].color = cc;
                proverka = prove(color_v,[j, mas[j].color]);
                if (proverka == false) {
                  color_v.push([j, mas[j].color]);
                }
                //color_v.push([j, mas[j].color]);
              } else {
                mas[j].color = 0;
                dop_mas = color_v.splice(k+1,k);
                console.log(dop_mas)
              }
            }
          }
        }
      }
    }
  }
}

```


ДОДАТОК Г
Слайди презентації

Національний університет “Запорізька Політехніка”

Кафедра програмних засобів

Дипломна кваліфікаційна робота

Програмна реалізація методів формування розкладу навчальних занять

Виконав
ст. гр. КНТ-227

С.Р. Медведєв

Керівник
к.т.н., доцент

А.О. Олійник

Рисунок Г.1 – Слайд №1

Мета, об’єкт, предмет дослідження

Об’єкт дослідження – системи формування розкладу.

Предмет дослідження – методи формування розкладу навчальних занять.

Мета роботи – програмна реалізація методів формування розкладу навчальних занять.

Рисунок Г.2 – Слайд №2

Задачі для роботи

- аналіз існуючих аналогічних програмних продуктів;
- огляд існуючих методів вирішення завдання;
- вибір середовища для розробки та його аналіз;
- вибір мови програмування та її аналіз;
- аналіз роботи методів вирішення завдання та вибір одного з них для розробки веб-застосунку формування розкладу занять.

Рисунок Г.3 – Слайд №3

Порівняльна характеристика аналогічних програмних продуктів

Можливості	FET	«Расписание»	«НИКА»	«АВТОР»	«1С»
Наявність користувацького інтерфейсу	Так	Так	Так	Так	Так
Кросплатформеність	GNU OC	Так	Так	Так	Так
Наявність зрозумілої та повної документації	Так	Так	Так	Так	Так
Комерційний проект	Ні	Так	Так	Так	Так
Десктопний додаток	Так	Так	Так	Так	Так
Треба встановлювати ПП на кожен комп'ютер	Так	Так	Так	Так	Так
Можливість імпорту/експорту файлів формату Excel	Тільки CSV	Так	Так	Так	Так
Рівень можливостей	+	++	+++	+++	++++

Рисунок Г.4 – Слайд №4

ER-діаграма розроблюваної нереляційної бази даних

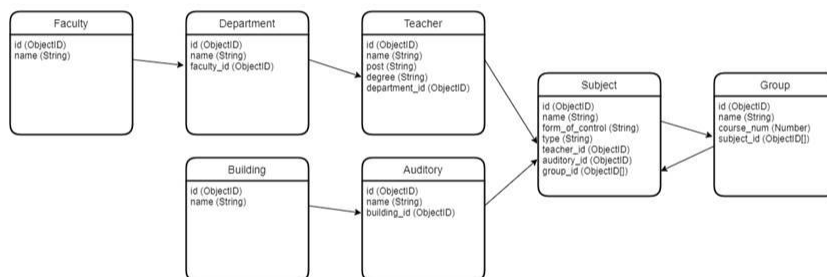


Рисунок Г.5 – Слайд №5

Структурна схема розроблюваного програмного продукту

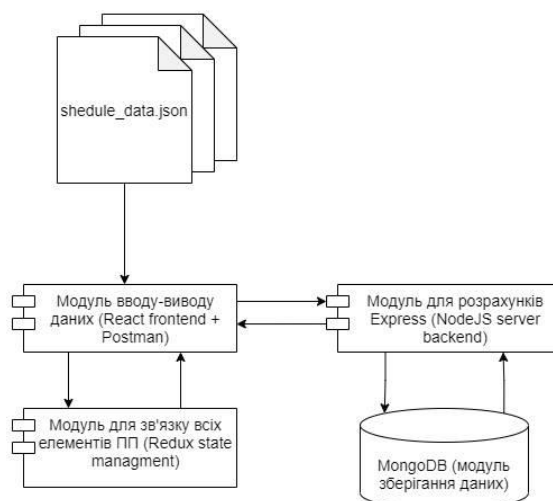


Рисунок Г.6 – Слайд №6

Метод гілок та меж

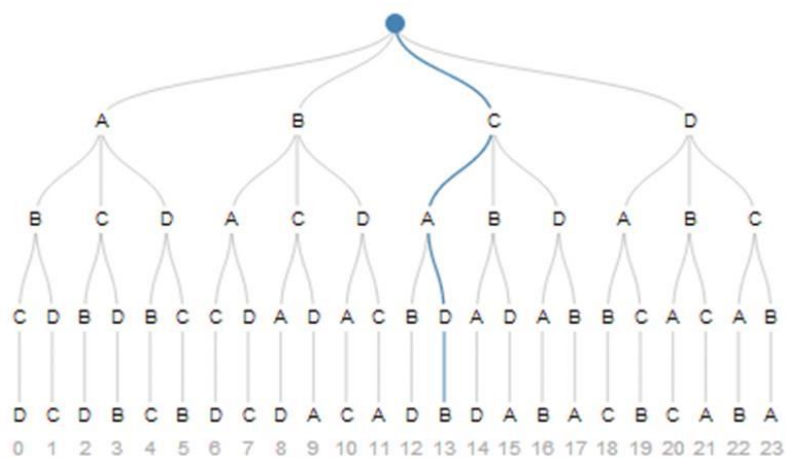


Рисунок Г.7 – Слайд №7

Метод імітації відпалу

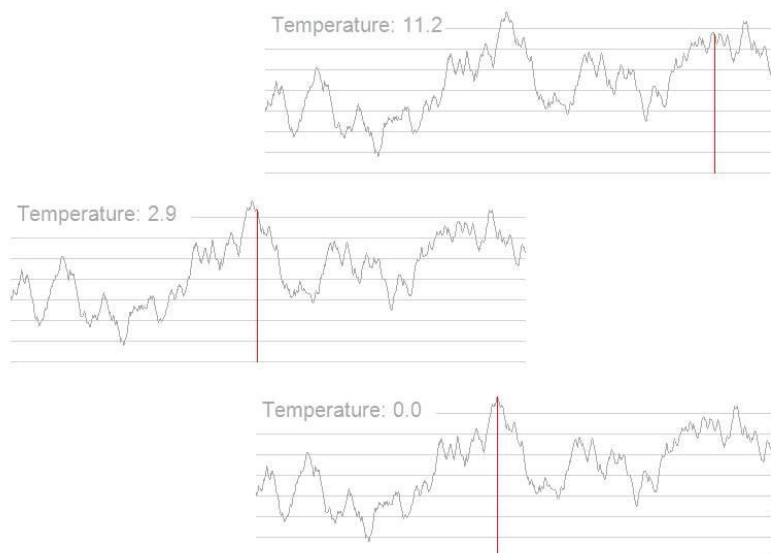


Рисунок Г.8 – Слайд №8

Мурашиний алгоритм

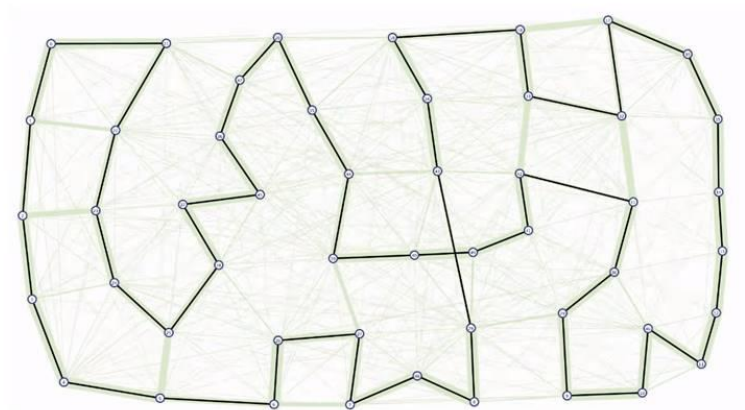


Рисунок Г.9 – Слайд №9

Генетичний алгоритм

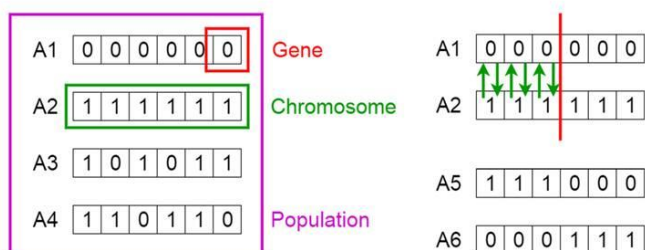


Рисунок Г.10 – Слайд №10

Метод розфарбовування графа

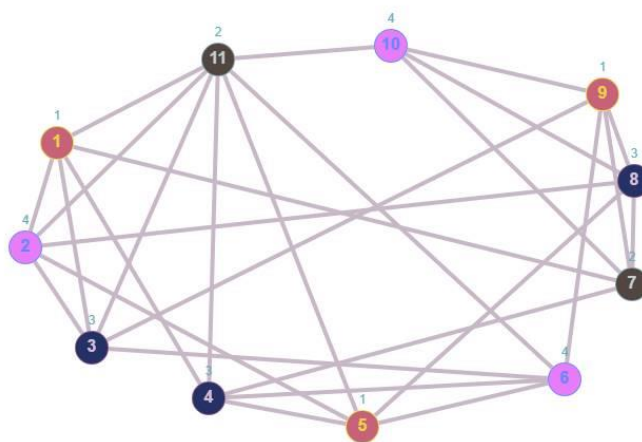


Рисунок Г.11 – Слайд №11

Порівняльна характеристика існуючих методів для формування розкладу занять

Можливості	Метод гілок та меж	Метод імітації відпалу	Генетичний алгоритм	Мурашиний алгоритм	Метод розфарбування графа
Простота та прозорість	Так	Ні	Так	Ні	Так
Адаптивність до особливостей вирішувальної задачі	Ні	Ні	Так	Так	Так
Велика витратність часу при великій розмірності початкових даних	Так	Так	Ні	Так	Ні
Велика точність розв'язку вирішувальної задачі	Ні	Так	Так	Так	Так
Висока похибка між бажаними і отриманими результатами	Так	Ні	Ні	Так	Ні

Рисунок Г.12 – Слайд №12

Графік залежності роботи наведених методів за швидкістю

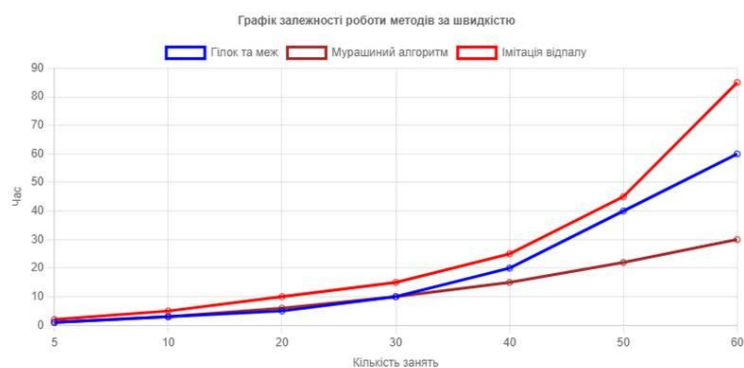


Рисунок Г.13 – Слайд №13

Графік залежності роботи наведених методів за точністю



Рисунок Г.14 – Слайд №14

Графік залежності роботи ГА та методу розфарбування графу за швидкістю

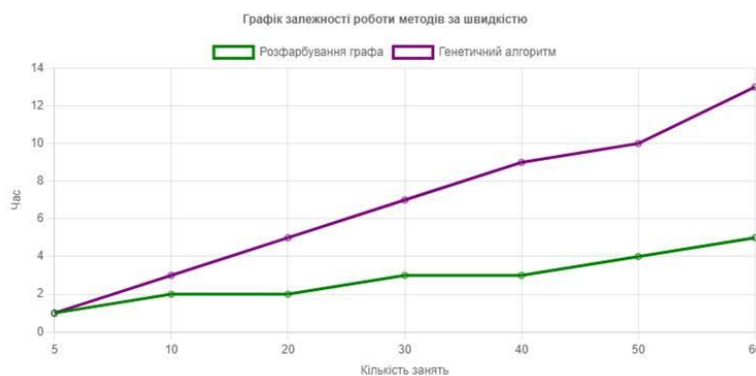


Рисунок Г.15 – Слайд №15

Графік залежності роботи ГА та методу розфарбування графу за точністю

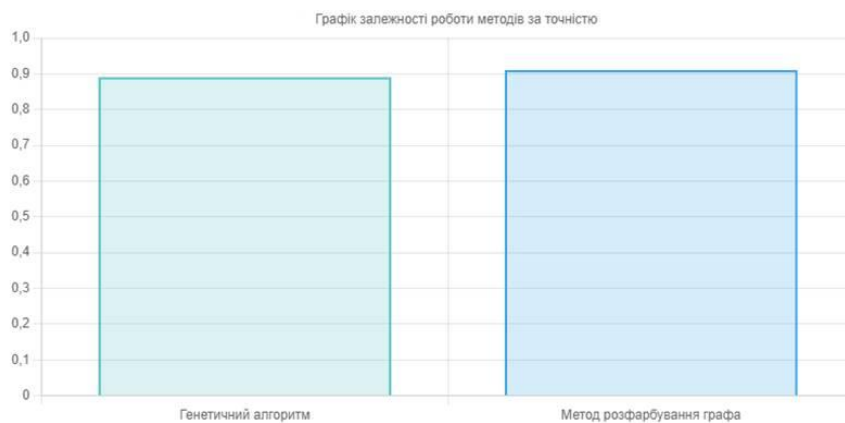


Рисунок Г.16 – Слайд №16

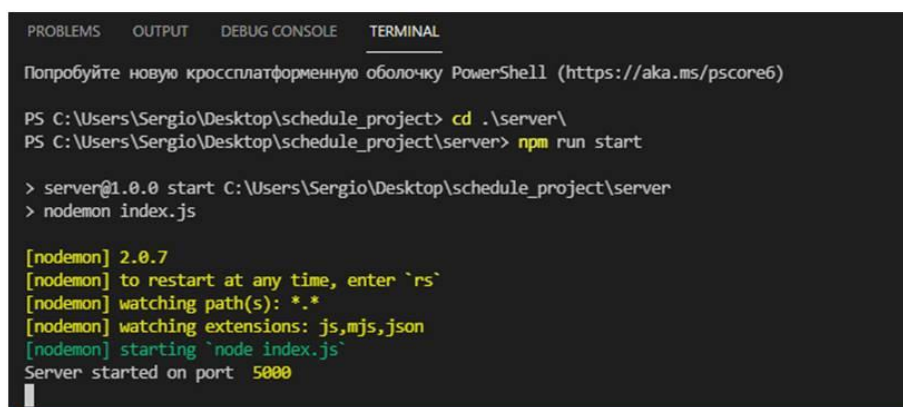
Вибір методу для розробки

Для вирішення поставленого завдання було обрано метод розфарбування графа, оскільки він дуже добре показав себе при обробці невеликих масивів даних.

Для подальших модифікацій та при зростанні вхідної вибірки, рекомендується вже використовувати генетичний алгоритм.

Рисунок Г.17 – Слайд №17

Запуск серверу та підключення до БД MongoDB



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Попробуйте нову кроссплатформенну оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\Sergio\Desktop\schedule_project> cd .\server\
PS C:\Users\Sergio\Desktop\schedule_project\server> npm run start

> server@1.0.0 start C:\Users\Sergio\Desktop\schedule_project\server
> nodemon index.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Server started on port 5000
```

Рисунок Г.18 – Слайд №18

Головна сторінка з кластером «scheduleDB»

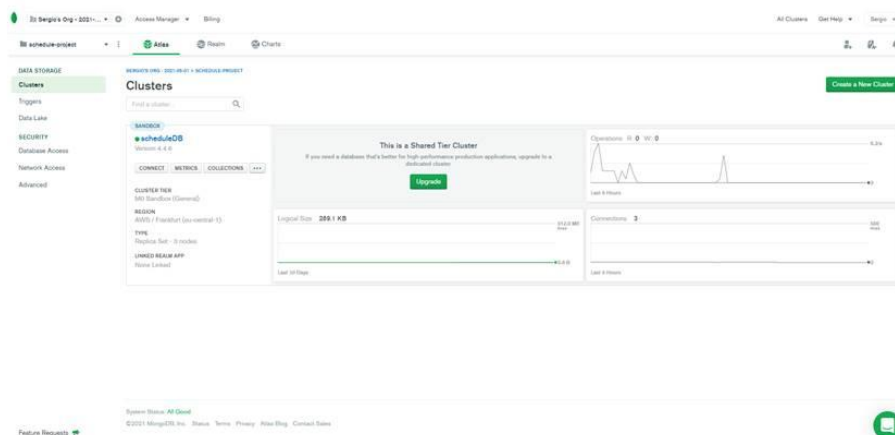


Рисунок Г.19 – Слайд №19

Список створених колекцій у кластері «scheduleDB»

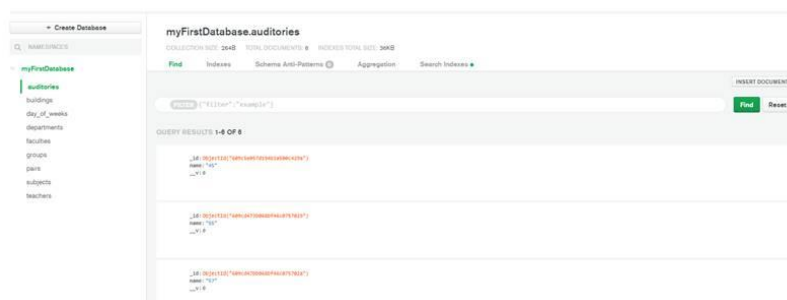


Рисунок Г.20 – Слайд №20

Графічний інтерфейс застосунку «Postman»

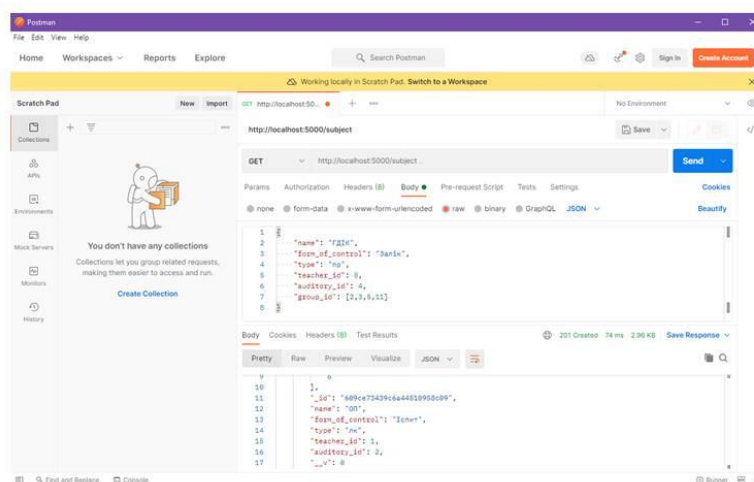


Рисунок Г.21 – Слайд №21

Головна сторінка веб-застосунку з результатами роботи методу розфарбовування графа

ScheduleProject		Graph coloring method											
		Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7	Group 8	Group 9	Group 10	Group 11	Group 12
Monday	I	OT (m) Tefimov F.B. Auditory 45	OT (m) Tefimov F.B. Auditory 45	OT (m) Tefimov F.B. Auditory 45	OT (m) Tefimov F.B. Auditory 45	OT (m) Tefimov F.B. Auditory 45	OT (m) Tefimov F.B. Auditory 45	OOT (m) Morozov H.O. Auditory 57	OOT (m) Morozov H.O. Auditory 57	OOT (m) Delneva R.O. Auditory 49			
	II						OT (m) Delneva R.O. Auditory 49						
	III		ГДК (m) Kazim D.A. Auditory 49	ГДК (m) Kazim D.A. Auditory 49		ГДК (m) Kazim D.A. Auditory 49						ГДК (m) Kazim D.A. Auditory 49	
	IV												
	V												
	VI												
Tuesday	I	OT (m) Morozov H.O. Auditory 57	OT (m) Morozov H.O. Auditory 57	OT (m) Morozov H.O. Auditory 57	OT (m) Delneva R.O. Auditory 49	OT (m) Delneva R.O. Auditory 49		OOT (m) Tefimov F.B. Auditory 55	OOT (m) Tefimov F.B. Auditory 55	OOT (m) Tefimov F.B. Auditory 55			
	II	ГДК (m) Kazim D.A. Auditory 49											
	III												

Рисунок Г.22 – Слайд №22

Висновки

При виконанні дипломної кваліфікаційної роботи бакалавра було проаналізовано аналогічні системи вирішення завдання формування розкладу занять, та наведено їх порівняльний аналіз, який показав, що ці аналоги не підходять до випадку з невеликим обсягом даних.

Проведено аналіз та вибір інструментарію для розробки програмного продукту, було спроектовано ER-діаграму та структуру майбутнього ПП. Для розробки ПП було обрано мову програмування JavaScript, платформу Node.js та бібліотеки React та Redux. Також, було обрано середовище для розробки Visual Studio Code.

Проаналізовано вхідні дані у спрощеному вигляді. Також, були проаналізовані на працездатність, точність результатів та швидкість роботи наступні методи: Метод гілок та меж, Метод імітації відпаду, Мурашиний алгоритм, Генетичний алгоритм та Метод розфарбування графа. Для вирішення поставленого завдання було обрано метод розфарбування графа, оскільки він дуже добре показав себе при обробці невеликих масивів даних. Для подальших модифікацій та при зростанні вхідної вибірки, рекомендується вже використовувати генетичний алгоритм.

Рисунок Г.23 – Слайд №23