

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ОНЛАЙН ОРГАНАЙЗЕРУ
ДЛЯ СТУДЕНТІВ. SOFTWARE IMPLEMENTATION OF
AN ONLINE TASK ORGANIZER FOR STUDENTS

Виконав: студент 4 курсу, групи КНТ-127
Спеціальності 121 Інженерія програмного
забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Дзугань А.О.

(прізвище та ініціали)

Керівник Колпакова Т.О.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (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(КИ)

Дзуганя Андрія Олександровича
 (прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Програмна реалізація онлайн органайзеру для студентів. Software Implementation of an Online Task Organizer for Students
 керівник проекту (роботи) Колпакова Тетяна Олексіївна, 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 проекту (роботи) 17 травня 2021 року

3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір технологій для розробки веб-застосунку. 3. Засоби програмної реалізації веб-застосунку. 4. Програмна реалізація веб-застосунку 5. Експлуатація та тестування веб-застосунку. 6. Технічне завдання. 7. Текст програми.

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

6. Консультанти розділів проекту (роботи)

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

7. Дата видачі завдання 15 березня 2021 р.**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка архітектури програми.	4 тиждень	Розділ 2
4	Розробка програми.	5–6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи.	9 тиждень	

Студент _____ Дзугань А.О. _____
(підпис) (прізвище та ініціали)Керівник проекту (роботи) _____ Колпакова Т.О. _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
86 с., 1 табл., 27 рис., 3 дод., 34 джерела.

ОНЛАЙН-ОРГАНАЙЗЕР, НАВЧАЛЬНИЙ ПРОЦЕС, ВЕБ-ЗАСТОСУНОК, JAVASCRIPT, REACTJS, NODE.JS, EXPRESS, MONGODB.

Об'єкт дослідження – процес організації графіку освітнього процесу для студентів та викладачів.

Предмет дослідження – онлайн органайзер для зручного контролю та актуалізації інформації щодо графіку освітнього процесу.

Мета роботи – створення веб-застосунку для надання інформації щодо графіку освітнього процесу для студентів та викладачів.

Матеріали, методи та технічні засоби: структурне, об'єктно-орієнтоване та функціональне програмування, мова програмування JavaScript, середовище виконання Node.js, веб-фреймворк Express, система керування базою даних MongoDB, персональний комп'ютер з процесором Intel Core 2 Duo під управлінням операційної системи Linux Kubuntu 20.04.

Результати: розроблено онлайн органайзер, який дозволяє отримувати та керувати інформацією щодо графіку навчального процесу. Онлайн органайзер може бути використаний у вищих навчальних закладах, необхідно лише ввести процес оновлення даних з розкладу занять того чи іншого вузу.

Висновки. Розроблено онлайн органайзер для студентів та викладачів, який дозволяє користувачеві ефективно керувати навчальним процесом.

Галузь використання: навчальні заклади системи освіти.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 86 pages, 1 tables, 27 figures, 3 appendixes, 34 sources.

ONLINE ORGANIZER, LEARNING PROCESS, WEB APPLICATION, JAVASCRIPT, REACTJS, NODE.JS, EXPRESS, MONGODB.

The object of research is the process of organizing the schedule of the educational process for students and teachers.

The subject of the research is an online organizer for convenient control and updating information on the schedule of the educational process.

The purpose of the work is to create a web application to provide information on schedule of educational process for students and teachers.

Materials, methods and technical means: structural, object-oriented and functional programming, programming language JavaScript, Node.js runtime, web framework Express, database management system MongoDB, a personal computer with a processor Intel Core 2 Duo running Linux Kubuntu 20.04.

Results. Developed an online organizer that allows you to receive and manage information about the schedule of the educational process. Online the organizer can be used in higher educational institutions, you only need to enter the process of updating the data from the schedule of classes or another university.

Conclusions. Developed an online organizer for students and teachers, which allows the user to effectively manage the learning process.

The field of use is a educational institutions of the education system.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Опис об'єкта дослідження	11
1.2 Опис предмета дослідження	13
1.3 Висновки за розділом 1	21
2 Вибір технологій для розробки веб-застосунку	22
2.1 Мова програмування JavaScript	22
2.2 Серверна платформа Node.js	24
2.3 Фреймворк Express.js	26
2.4 База даних MongoDB	26
2.5 Формат передачі даних JSON	30
2.6 Висновки за розділом 2	32
3 Засоби програмної реалізації веб-застосунку	33
3.1 Функціональні вимоги до веб-застосунку	33
3.2 Опис архітектури та обраних технологій	34
3.3 Висновки за розділом 3	42
4 Програмна реалізація веб-застосунку	44
4.1 Функціональні вимоги до веб-застосунку	44
4.2 Процес програмної реалізації веб-застосунку	45
4.3 Структурна схема веб-застосунку	45
4.4 Моделювання бази даних	47
4.5 Безпека веб-застосунку	52
4.6 Висновки за розділом 4	54
5 Експлуатація та тестування веб-застосунку	55
5.1 Вхід до системи	55
5.2 Створення нової групи	58
5.3 Створення нового оголошення	59

5.4 Розклад занять.....	60
5.5 Сторінка «Моя група».....	62
5.6 Висновки за розділом 5	63
Висновки.....	64
Перелік джерел посилань.....	66
Додаток А. Технічне завдання.....	69
Додаток Б. Текст програми.....	72
Додаток В. Слайди презентації	79

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
BSON	– Binary JavaScript Object Notation;
DOM	– Document Object Model;
HTML	– HyperText Markup Language;
JS	– JavaScript;
JSON	– JavaScript Object Notation;
NoSQL	– Not Only Structured Query Language;
PIM	– Personal Information Manager;
REST	– Representational State Transfer;
SPA	– Single Page Application;
URL	– Uniform Resource Locator;
БД	– база даних;
ПЗ	– програмне забезпечення;
СКБД	– система керування базами даних.

ВСТУП

Сучасний етап розвитку суспільства вимагає інтенсивного запровадження інформаційно-комунікаційних технологій у всіх сферах діяльності, зокрема у системі освіти. В умовах зростання інформаційних потоків, формування, передавання та адекватне використання інформації є однією з обов'язкових складових управлінської та навчально-виховної діяльності. Ефективне функціонування вищого навчального закладу вимагає створення єдиного інформаційного середовища, яке надає доступ до актуальної та повної інформації учасникам освітнього процесу та є необхідним інструментом діяльності співробітників вузу та навчання студентів.

Організація навчального процесу це достатньо складна робота, що потребує багато часу та зусиль. Вирішенням даної проблеми є використання інформаційних технологій, що здатні пришвидшити та полегшити цей процес [1].

Оптимізувати та підвищити ефективність навчального процесу дозволяє використання студентами методів та технологій тайм-менеджменту. З розвитком інформаційних технологій паперові прототипи засобів тайм-менеджменту видозмінилися в електронні органайзери, потім в комп'ютерні програми та онлайн-органайзери. До основних функцій сучасних органайзерів можна віднести: роботу з календарем, планування завдань, нагадування та сповіщення, наявність контактної книги, можливість створення нотаток.

Наразі існує достатньо велика кількість систем, які дозволяють зручно планувати свій час, але зазвичай вони створені лише для одного користувача. Студент може розпланувати свій час згідно графіку навчального процесу за допомогою портативного (або стаціонарного) пристрою, але зважаючи на те, що розклад у будь-якому навчальному закладі – структура динамічна, то при кожній зміні у ньому до органайзеру буде потрібно вносити правки.

Розробка такого органайзера у вигляді веб-застосунку заощаджує ресурси пристрою та підтримує кросплатформінг, що є ключовою перевагою над звичайними застосунками.

Онлайн-органайзери, як засоби управління часом є більш зручними, оскільки мають додаткові функції (синхронізація та захист інформації, нагадування про майбутні події), що ефективно впливає на планування діяльності.

Основною метою програмної реалізації онлайн-органайзеру для студентів є надання користувачам зручного та надійного засобу для отримання та керування інформацією щодо графіку навчального процесу.

Для досягнення поставленої мети у роботі вирішувалися такі завдання:

- проаналізувати предметну область та аналогічні органайзери;
- проаналізувати методи розробки веб-застосунків;
- обрати найзручніші та найактуальніші технології для їх використання при розробці онлайн органайзеру;
- спроектувати веб-застосунок для студентів та викладачів враховуючи всі особливості веб-технологій;
- провести тестування розробленого веб-застосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис об'єкта дослідження

Більшість видів діяльності вимагає від працівників високого рівня індивідуального розвитку, який є неможливим без уміння самостійно планувати та організовувати свою роботу, розставляти пріоритети, визначати найбільш важливі цілі, а також досягати їх з найменшими витратами власних зусиль і часу.

Для здобувача освіти опанування навичками раціонального використання часу (одного з найцінніших ресурсів) та ефективної організації діяльності є важливим етапом особистісного розвитку та становлення як фахівця.

Сучасний освітній процес вимагає зміни позиції студента з об'єкта навчання в активного суб'єкта навчально-професійної діяльності, а також активізує самостійну навчально-пізнавальну діяльність студентів. Студент повинен бути готовий самостійно організовувати свою навчальну діяльність і час, здобувати знання та вміло застосовувати їх на практиці [2].

Застосування технік тайм-менеджменту в організації навчальної діяльності дає студентам можливість більш ефективно використовувати процес навчання для підготовки до майбутньої професійної діяльності, а час вільний від навчання для самореалізації та саморозвитку.

З появою інформаційних технологій тайм-менеджмент перейшов в нову фазу розвитку. Персональні комп'ютери, Інтернет, супутні інформаційні технології дозволяють більш точно фіксувати витрати часу і полегшують їх аналіз.

Для автоматизації процесів тайм-менеджменту зручно використовувати сучасні технічні засоби. Сьогодні навіть найпростіші мобільні телефони містять такий набір інструментів і засобів тайм-менеджера, як: контакти (телефонну книгу), календар, список завдань, можливість робити нотатки, аудіозаписи і т. д. Можливість постійно мати доступ до електронної пошти та Інтернету –

незамінна функція, яка дозволяє повністю звільнитися від прив'язки до робочого місця [2].

Проте, не слід плутати інструменти організації часу з їх носіями. Так, ПК, ноутбук, мобільний телефон або простий органайзер – це лише представники електронних та паперових носіїв інструментарію організації часу.

Програмне забезпечення – один з інструментів планування, який дозволяє чітко розписати особистий та робочий час.

Однією з головних проблем організації навчального процесу з якою стикається навчальний заклад є гармонізація розкладу занять з наявним аудиторним фондом та можливостями викладачів та швидке сповіщення студентів та співробітників щодо змін [1].

Вирішенням даної проблеми є використання інформаційних технологій, що здатні пришвидшити та полегшити цей процес, а саме Personal Information Manager (PIM). Технологія Personal Information Manager включає в себе широкий набір необхідних інструментів для керування контактами, календарями, завданнями, зустрічами та іншими особистими даними. Інструменти PIM розрізняються в залежності від потреб користувача та вартості продукту [3].

Personal Information Manager зазвичай мають такий функціональний набір:

- адресна книга;
- списки (наприклад, списки завдань);
- календар;
- нотатки;
- планування зустрічей та завдань;
- сповіщення та попередження;
- електронна пошта та система миттєвих повідомлень;
- факсимільний зв'язок;
- голосова пошта;
- хмарне сховище;
- організація спільного доступу до керування даними [3].

Онлайн-сервіси для планування подій, зустрічей, справ з прив'язкою до календаря мають значні переваги перед оффлайновими аналогами, найголовніша з них – їх мобільність та доступ до інформації, яка зберігається в Інтернеті.

1.2 Опис предмета дослідження

Існує велика кількість програмних продуктів: додатків, веб-сайтів та мобільних додатків, спрямованих на спрощення виконання завдань, пов'язаних з організацією часу. Всі додатки мають свій унікальний дизайн та багато різного функціоналу, який хоч і відрізняється і є унікальним, але має одну ідею та мету.

Електронні календарі, органайзери, планувальники: Google Calendar, Microsoft Outlook, C-Organizer, AM Noteboo, Diary from Enot, WinOrganizer, EssentialPIM, Evernote, LeaderTask тощо – програмні засоби для планування і організації особистої та спільної діяльності, ефективного розподілу робочого часу та узгодження термінів виконання окремих етапів дослідження.

Багато з даних програм, хоча і мають необхідний функціонал, але не призначені саме для вирішення проблеми організації індивідуального навчального графіку. Окрім того, мають обмеження, які докорінно будуть впливати на вибір додатку та його впровадження. Нижче розглянемо деякі з них.

1.2.1 Сервіс Google Calendar

Google Calendar – це безкоштовний сервіс, який допомагає у застосуванні тайм-менеджменту та дає можливість створювати календар-органайзер для власних потреб. Дане програмне забезпечення було розроблено американською транснаціональною корпорацією Google. В квітні 2006 року було випущено десктопну версію продукту, а в липні 2009 року була

представлена фінальна версія у вигляді веб-застосунку та мобільних додатків для платформ Android та iOS. Щоб отримати можливість використання застосунку, користувач повинен мати аккаунт Google (<https://www.google.com>) [4].

Даний програмний продукт надає змогу користувачам створювати, змінювати, видаляти та проводити інші операції з своїми подіями. Налаштування сповіщень можна підв'язати до їх типу та відповідно часу. Окрім того до подій можливо додавати контакти та координати місцезнаходження майбутньої зустрічі. Для своїх потреб користувачі можуть налаштувати додаткові особливості Google Calendar, такі як: підключення спеціальних календарів, додання дні народження контактів та автоматичне відсилка вітань та листівок з цього приводу, календарі для конкретних країн, для яких уже підв'язані усі святкові та вихідні дні.

Було впроваджено машинне навчання, яке надало Google Calendar нові можливості. Одна з них – це інтеграція події з Gmail, де вся головна інформація про подію, така як назва, час, місце та інші з повідомлень користувача Gmail автоматично з'являється у календарі. Інша особливість – це нагадування про подію, яка надсилається користувачеві. Також існує пункт Розумні пропозиції, де веб-застосунок надає варіанти при заповненні даних, таких як назва, контакти та місце при створенні подій. Крім того, якщо користувач в пункті Цілі вказує свою мету додавання події, то штучний інтелект автоматично надає найбільш оптимальні часи для активності [5].

Мобільні додатки Google Calendar отримали різні відгуки. Деякі критики вважали, що дизайн додатку є недоречним та займає дуже багато простору, але інші високо оцінили графіку, зокрема чистий та сміливий графічний інтерфейс. Протилежні відгуки також отримала функція «розумного підбору». Проте, інтеграції між Google Calendar та Gmail було надано найвищі оцінки критиків та звичайних користувачів.

Робота з Google Calendar здійснюється у вікні браузера за допомогою веб-інтерфейсу, а всі дані зберігаються на сервері Google, тому отримати

доступ до розкладу занять, заходів та інших подій можна з будь-якого комп'ютера, що підключений до мережі Інтернет. Головний екран додатку зображений на рис. 1.1.

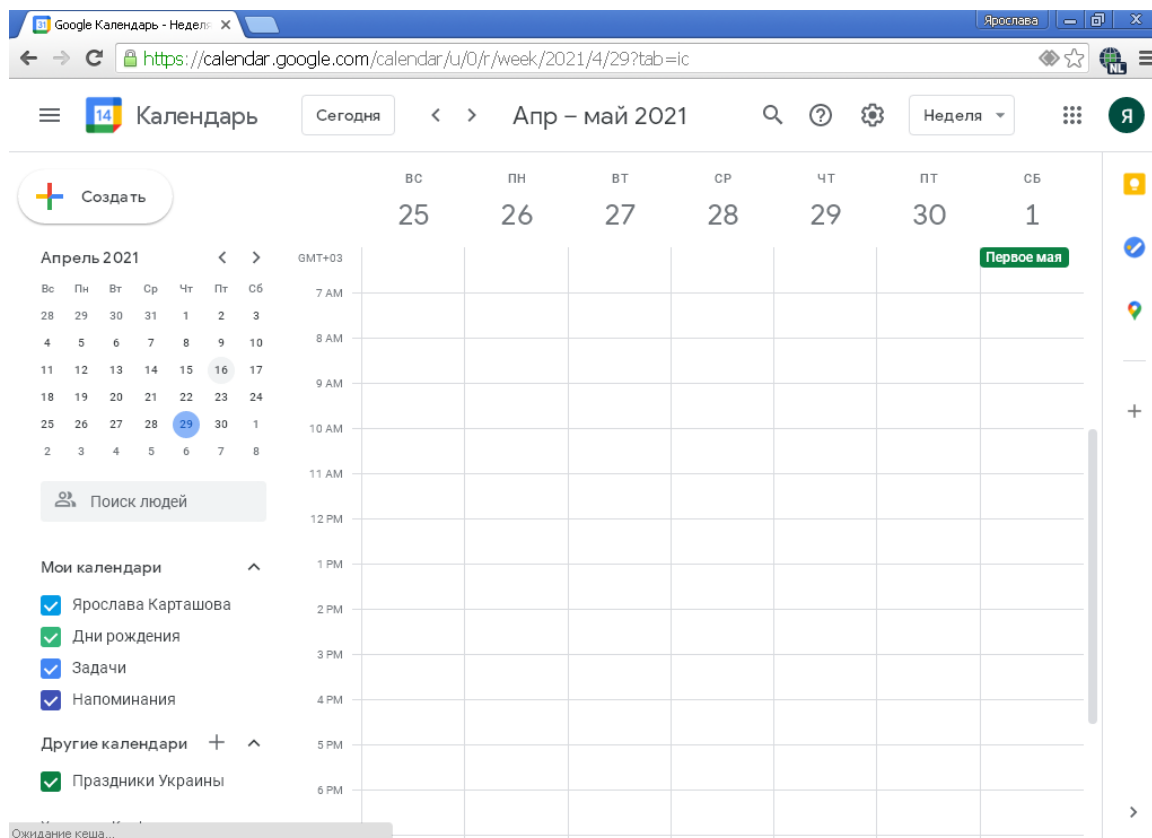


Рисунок 1.1 – Головний екран Google Calendar

В інтерфейсі активно використовуються технології JavaScript і AJAX, що прискорює реакцію системи і дозволяє користуватися нею з тією ж легкістю, як і програмою, встановленою локально [4].

Найголовнішою особливістю Google Calendar є те, що користувачі мають право створювати, змінювати та видаляти події. Кожна з подій має час старту та час завершення події, також є подія, яка триває цілий день.

Інші особливості Google Calendar:

- можливість повторення події, яку може запустити користувач, щоб не заповнювати дані кожного разу;
- висвітлення подій в різному кольоровому фоні для пріоритизації завдань;

- можливість використання різних форми перегляду календаря, такі як денний, тижневий, місячний, річний та порядок денний;
- можливість додання місцезнаходження для кожної з майбутніх подій;
- можливість налаштування сповіщення на власну електронну пошту або мобільний пристрій, що забезпечує дотримання плану подій;
- можливість запрошення до подій інших користувачів за посиланням, запрошені в свою чергу мають вибір чи погодитися.

Одна з ключових переваг цього сервісу – можливість спільного використання Google Calendar. Можна надати доступ до власного Google Calendar обраним користувачам, щоб вони дізнавалися про ваш вільний час, або дивилися інформацію про заплановані вами зустрічі. Можна легко звіритися з розкладом колег, щоб обрати зручний час для наради або спільної зустрічі [6].

1.2.2 Застосунок-органайзер Microsoft Outlook

Microsoft Outlook (MS Outlook) входить до пакету офісних програм Microsoft Office, створеного корпорацією Microsoft для операційних систем Windows, macOS, Android та iOS.

MS Outlook – це поштовий клієнт, який має потужний функціонал повноцінного електронного органайзеру, а саме: календар, планувальник завдань, записна книжка, менеджер контактів. та менеджер групової роботи. Особливістю даного PIM є можливість інтеграції з іншими продуктами Microsoft Office, хмарним сховищем One Drive та можливість спільної одночасної роботи над проектами і автоматичного складання щоденника роботи [7].

Outlook може використовуватися як окремий застосунок, так і виступати в ролі клієнта для поштового сервера Microsoft Exchange Server, що надає додаткові функції для спільної роботи користувачів однієї організації: загальні поштові скриньки, теки завдань, календарі, конференції, планування і резервування часу загальних зустрічей, узгодження документів. Microsoft

Outlook і Microsoft Exchange Server є платформою для організації документообігу, оскільки вони забезпечені системою розробки призначених для користувача плагінів і скриптів, за допомогою яких можливе програмування додаткових функцій документообігу (і не тільки документообігу), не передбачених в стандартному постачанні. За допомогою розділу Посилання можна знайти інформацію про деякі плагіни для Microsoft Outlook [8].

Характерної рисою Outlook є можливість переглянути інформацію адресної книги про всіх користувачів системи і їх контактну інформацію. Хмарне сховище One Drive, дозволяє створити сховище документів до яких можна надати доступ для спільної роботи, редагування або перегляду. Система сповіщення інформує користувача про події які він запланував або в які його запросили, про нові повідомлення на електронній пошті. Календар у Outlook дозволяє створювати наради, планувати свій робочий час та надавати доступ до свого календаря іншим користувачам. Календар Outlook зображений на рис. 1.2.

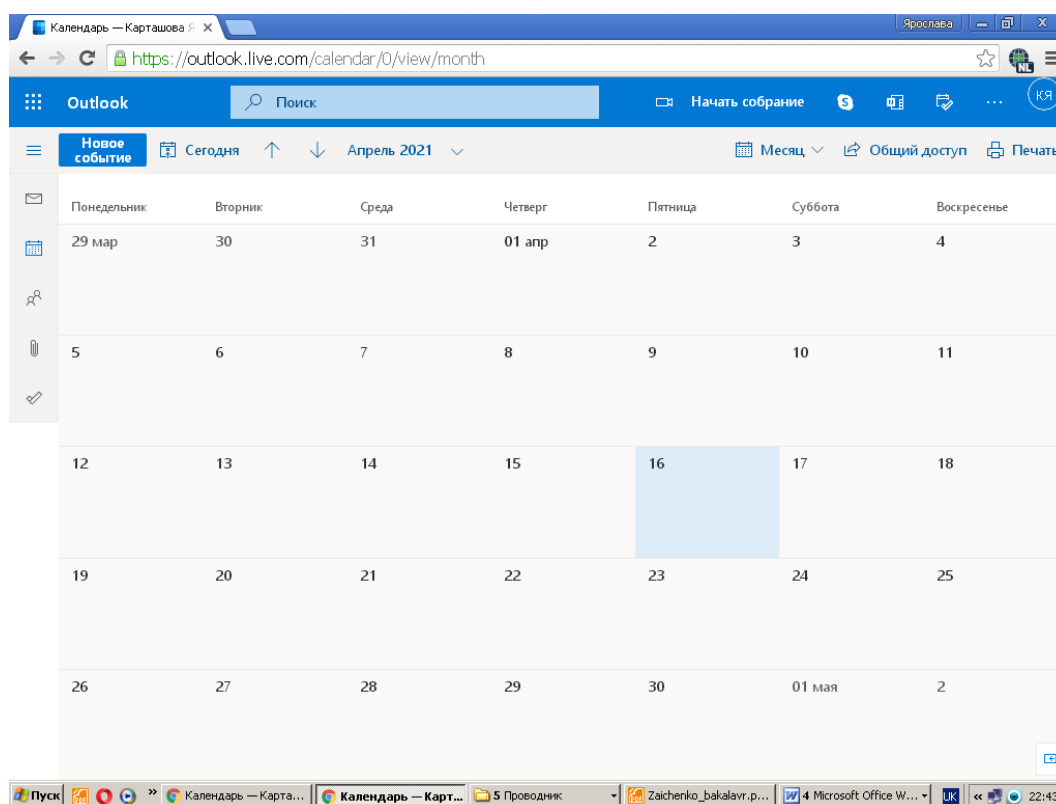


Рисунок 1.2 – Календар Microsoft Outlook

Вище вказані функції Outlook дають можливість для вирішення таких проблем, як: проблеми з наявністю вільних аудиторій, рівномірний розподіл навчального часу, можливість підтримувати зв'язок викладачів зі студентами. На підставі цього, в 2014 році Microsoft Outlook було впроваджено в Донецькому національному університеті імені Василя Стуса. Також було запропоновано створення загально-факультетського календаря, де зазначався б розклад занять; номер аудиторії, що використовується; тип заняття, наприклад, практичне заняття; викладача та присутніх студентів. Тобто за допомогою Outlook стало можливо синхронізувати навчальний процес між викладачами зі студентами з однієї сторони та аудиторним фондом з іншої. Це надало такі переваги, як:

- вирішення проблем відсутність необхідних аудиторій;
- отримання даних про розмір групи або груп студентів для підбору оптимального розміру аудиторії для занять;
- можливість швидкого сповіщення про зміни у розкладі або заміна чи перенесення занять викладачем;
- отримання даних для створення картини усього навчального процесу для можливості швидкої модернізації або зміни.

За допомогою Outlook викладачі мають змогу підтримувати зв'язок зі студентами у будь-який час, надсилати їм додаткові матеріали, попереджати про зміни у розкладі [1].

Microsoft Outlook має доволі захищену систему, але з деякими особливостями. Програма має стандартні функції, які захищають користувача, в тому числі блокування зовнішніх веб-сайтів та непідтверджених зображень, виключення вкладених файлів та не допущення виконуючих файлів. Проте, при необхідності, користувач може легко обійти цю заборону, але за такі дії Outlook не бере на себе відповідальність.

1.2.3. Менеджер персональної інформації EssentialPIM

EssentialPIM – це персональний органайзер, який дозволяє управляти календарем, завданнями, справами, нотатками, контактами, паролями та електронною поштою. Програму розроблено незалежною технологічною компанією Astonsoft (Таллінн, Естонія) для Windows та мобільних платформ Android та iOS [9].

EssentialPIM для Windows дозволяє керувати інформацією на декількох платформах і хмарних додатках. Наявність переносної версії, яка може працювати з USB-накопичувача або інших переносних пристроїв, та широкі можливості синхронізації, імпорту та експорту надають повний контроль над вашими даними [10].

EssentialPIM – зручний і швидкий веб-застосунок, який має зрозумілий інтерфейс, що дозволяє відкривати нові функції під час використання програми. Інтерфейс EssentialPIM представлено на рис. 1.3.

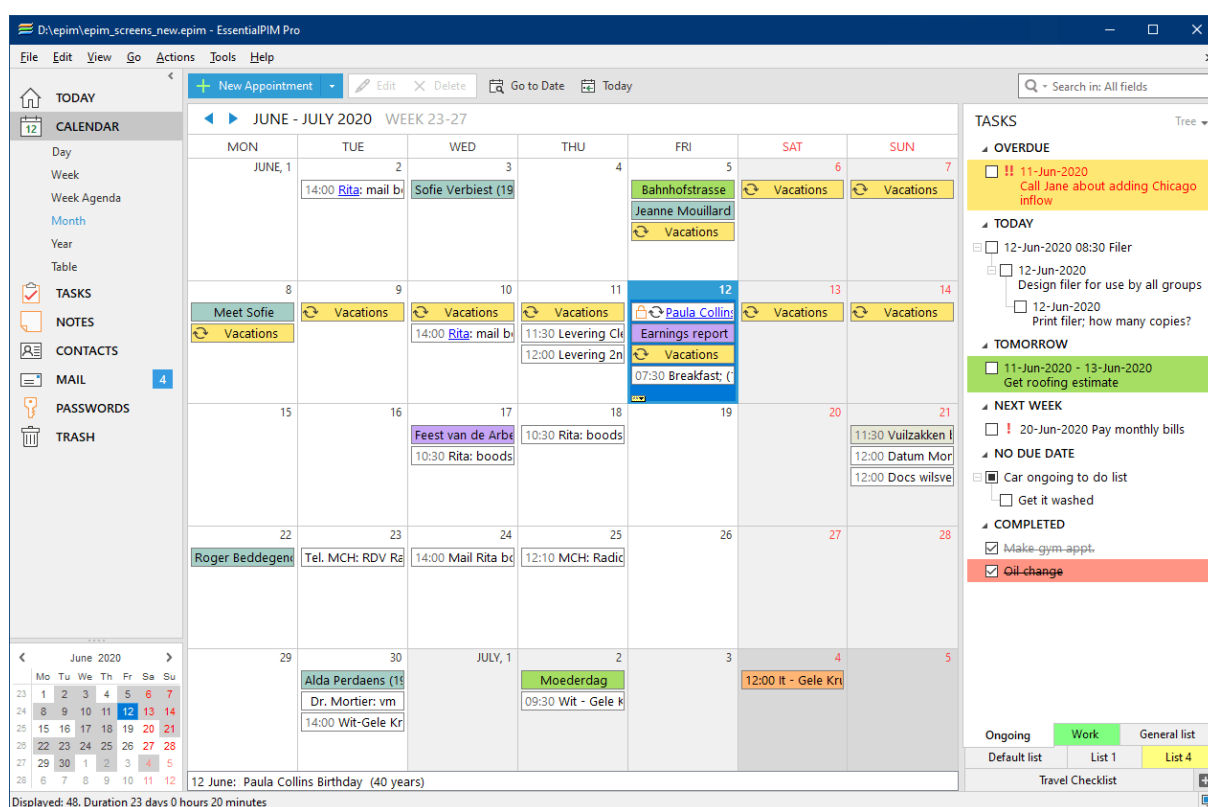


Рисунок 1.3 – Інтерфейс EssentialPIM [6]

Інтерфейс EssentialPIM крім англійської мови, також доступний російською, білоруською та українською мовами.

Додаток EssentialPIM для Android може бути синхронізоване з версією EssentialPIM для Windows, а також з відповідними сервісами Google, включаючи: Календар, Завдання, Контакти, Диск.

EssentialPIM для iPhone / iPad дозволяє зберігати всі дані EssentialPIM і дистанційно керувати ними. Додаток керує і забезпечує синхронізацію наступних даних EssentialPIM: контактів, календаря, завдань і паролів (перші два типи даних синхронізуються з внутрішніми програмами iOS).

EPIM Cloud зберігає всі дані користувача EssentialPIM у безпечному і надійному хмарному сховищі та синхронізує їх в фоновому режимі.

Дані (за вибором користувача) може бути зашифровано з використанням AES (Rijndael) 256-bit. Синхронізація з усіма онлайн-сервісами проводиться тільки по SSL (HTTPS) протоколу. Таким чином гарантується повна безпека даних в будь-яких ситуаціях.

Також EssentialPIM пропонує розширені можливості імпорту та експорту даних. До переліку форматів, які підтримує EssentialPIM відноситься EML, XML, HTML, RTF, CSV, TXT, iCal, vCard і інші.

EssentialPIM використовує базу даних з відкритим вихідним кодом – Firebird. Це означає, що дані користувачів завжди доступні, не має незрозумілих і закритих форматів. Користувач також може самостійно писати плагіни та програми, що працюють з базою EssentialPIM [10].

Результати аналізу аналогів було систематизовано та зроблено висновки щодо їх особливостей, недоліків та переваг, які наведено в таблиці 1.1.

Таблиця 1.1 – Аналіз аналогів веб-застосунку

Характеристики	Google Calendar	Microsoft Outlook	EssentialPIM
Веб-застосунок	+	+	+
Мобільна версія	+	+	+
Десктопная версія	+	-	+
Необхідність наявності акаунту	+	-	-
Календар	+	+	+
Сповіщення	+	+	+
Синхронізація	+	-	+
Імпорт та експорт подій	+	-	+
Спільний доступ до подій	+	+	+
Хмарне сховище	+	-	+
Створення додаткового календаря за необхідності	+	-	-
Безпека	+	+	+
Зручний графічний інтерфейс	+	-	+

1.3 Висновки за розділом 1

У 1 розділі було розглянуто деякі існуючі Personal Information Manager, які дають можливість застосування технік тайм-менеджменту в організації навчальної діяльності.

Незважаючи на різноманіття функціоналу органайзерів-аналогів, можна зробити висновок про те, що не один з них не вирішує проблематику надання актуальної інформації щодо графіку навчального процесу в повному обсязі. Тому програмна реалізація онлайн органайзера з наданням користувачам зручного та надійного засобу для отримання та керування інформацією щодо графіку навчального процесу є доцільною.

2 ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ

В даний період існує великий спектр вибору мов програмування та технологій для розроблення програмного забезпечення, зокрема веб-застосунків. Для прискорення розробки програмних систем досить часто використовуються вже створені технології розробки – платформи, бібліотеки, фреймворки. Це не тільки прискорює розробку, а й дозволяє абстрагуватися до рівня вирішення проблеми, а не написання програмного коду.

Розглянемо основні особливості та переваги обраних мови програмування та технологій.

2.1 Мова програмування JavaScript

JavaScript (скорочено JS) – динамічна, об'єктно-орієнтована прототипна мова програмування, є реалізацією стандарт мови програмування ECMAScript, який затверджений міжнародною організацією ECMA згідно зі специфікацією ECMA-262 [11].

JavaScript розроблено Бренданом Айком (Ейхом) (Brendan Eich) для веб-браузера Netscape Navigator (розробник американська корпорація Netscape Communications) [12].

Слід відрізнити мови Java та JavaScript – це дві різні мови, які мають різний концепт, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Обидві мови наслідують синтаксис мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme [11].

JavaScript – це інтерпретована мова програмування з об'єктно-орієнтованими можливостями. З точки зору синтаксису базова мова JavaScript нагадує C, C ++ і Java такими програмними конструкціями, як інструкція if, цикл while та оператор &&. Однак ця подібність обмежується синтаксичною схожістю [13].

JavaScript є нетипізованою мовою, тому в неї не потрібно визначати типи змінних. Об'єкти в JavaScript відображають імена властивостей на довільні значення. Цим вони більше нагадують асоціативні масиви Perl, ніж структури C або об'єкти C ++ або Java [13].

Механізм об'єктно-орієнтованого наслідування JavaScript скоріш схожий на механізм прототипів в таких маловідомих мовах, як Self, та сильно відрізняється від механізму наслідування в C ++ та Java.

Як і Perl, JavaScript – це мова, що інтерпретується, і деякі її інструменти, наприклад регулярні вирази та засоби роботи з масивами, реалізовані за прикладом мови Perl.

Ядро мови JavaScript підтримує роботу з такими простими типами даних, як числа, рядки і булеві значення. Крім цього він має вбудовану підтримку масивів, дат і об'єктів регулярних виразів.

JavaScript найчастіше використовується у браузері, що надає можливість виконання коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

Структурно JavaScript можна представити у вигляді об'єднання трьох частин: ядра (ECMAScript), об'єктної моделі браузера та об'єктної моделі документа. Як зрозуміло з назви, ядро становить основу даної мови програмування. В ньому визначені основні типи даних, ключові та зарезервовані слова, конструкції, оператори, об'єкти та інші основні компоненти даної мови.

Об'єктна модель браузера є проміжним шаром між ядром та об'єктною моделлю документа. Вона дозволяє здійснювати вплив на браузер, а саме: здійснювати зациклювання коду, виконувати затримку у виконанні коду, наприклад, для очікування певних дій від користувача, здійснювати переадресацію користувача на вказану url адресу, отримувати інформацію про розмір вікна браузера та багато іншого.

Об'єктна модель документа є складовою частиною JavaScript та дозволяє виконувати маніпуляцію над HTML сторінкою або сторінкою, що створена за допомогою мови розмітки. Вона дозволяє змінювати структуру певного блоку, видаляти його вміст, копіювати та вставляти в інше місце. Завдяки такій складовій структурі дана мова програмування дозволяє виконувати велику кількість задач у браузері.

JavaScript має синтаксис схожий на синтаксис мов програмування C та Java. В даній мові назва змінної може починатися з букви, символу долара, підкреслення; для оформлення однорядкових коментарів використовуються `//`, а для багаторядкових `/**/`. До її особливостей відноситься динамічна типізація, яка дає змогу оголошувати змінні з одним типом даних, а потім записувати в них дані іншого типу, автоматичне управління пам'яттю та прототипне наслідування [13].

JavaScript використовується практично на всіх сучасних веб-сайтах. У всіх сучасних браузерах і клієнтських пристроях – настільних комп'ютерах, ігрових консолях, планшетах і смартфонах – є інтерпретатори JavaScript, що робить його найбільш поширеною мовою в історії програмування. Він входить в базову тріаду технологій, які необхідно знати всім розробникам веб-застосунків [7].

З появою Node.js мова програмування JavaScript також використовується для програмування на стороні сервера (подібно до таких мов програмування, як Java, PHP), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF-документів тощо [7].

2.2 Серверна платформа Node.js

Node.js – платформа з відкритим кодом, що використовується для створення клієнтських та серверних застосунків [14].

Один з найважливіших плюсів платформи – це асинхронність в поєднанні з подієвим підходом. На відміну від потокового, подієво-орієнтоване програмування засноване на зовнішніх діях. Тобто виконання програми залежить від дій користувача. Таке рішення спрощує програмування інтерактивних додатків, тобто заснованих на роботі з використанням введення-виведення. Це можуть бути онлайн-чати або ігрові програми, барвисті інтерактивні веб-сервіси або всілякі калькулятори, рейтинги, голосування і т.д.

Крім того, до переваг платформи Node.js відносять:

- простий і широко відомий JavaScript. Звичайно, платформа передбачає власні інструменти та особливості, наприклад, тут немає браузерних API, cookie або DOM, однак присутні власні бібліотеки. В основному використовуються можливості і синтаксис всім звичного JavaScript;

- велику стандартну бібліотеку. Платформа від самого початку мала широкий набір можливостей, а в нових версіях бібліотека поповнюється і поліпшується;

- велику кількість зовнішніх бібліотек і готових модулів. Використання пакетного менеджера NPM дозволяє постійно розвивати екосистему Node. Сьогодні число пакетів в ньому перевищило за цифру 500 тисяч і постійно зростає.

Важливо розуміти, що на відміну від звичайного браузерного JavaScript, із застосуванням Node додаток може звертатися до широкого переліку глобальних об'єктів, в тому числі document або window. В результаті програма може звертатися до вінчестера і файлової системи користувача, а також до бібліотек і програмним рішенням, що знаходяться на комп'ютері, що помітно розширює можливості взаємодії.

Найчастіше цю платформу використовують для створення веб-сервісів, що вимагають інтенсивного обміну інформацією з користувачами, в тому числі, для реалізації чатів, систем спільної роботи, соціальних мереж і т.д. Багато програм, створені на Node.js, складаються з серверної і клієнтської частин.

Для розуміння причин такого використання варто трохи докладніше зупинитися на асинхронності і можливостях подієво-орієнтованої парадигми. Суть програмування при цьому зводиться до того, що ви вказуєте програмі, які дії вона повинна виконати, а також яку функцію вона повинна викликати після виконання цих дій. Поки Node.js чекає виконання коду і виклику функції, вона може виконувати інші завдання. При цьому потік в очікуванні запиту блокуватися не буде.

Таким чином, програма може звернутися із запитом до бази даних сервера, а поки очікується відповідь, обробити інші запити. В результаті одночасно обробляються тисячі з'єднань, кожне з яких витрачає мало часу і ресурсів. Це помітно прискорює обмін інформацією з сервером і, як підсумок, роботу браузерного додатку.

2.3 Фреймворк Express.js

Express.js, або просто Express, – каркас веб-застосунків для Node.js, реалізований як вільне і відкрите програмне забезпечення під ліцензією MIT. Він спроектований для створення веб-застосунків і API. Де-факто є стандартним каркасом для Node.js. Автор фреймворка, TJ Holowaychuk, описує його як створений на основі написаного на мові Ruby каркаса Sinatra, маючи на увазі, що він мінімалістичний і включає велику кількість додаткових плагінів. Express може бути бек-ендом для програмного стека MEAN, разом з базою даних MongoDB і каркасом Vue для фронт-енду [15].

2.4 База даних MongoDB

MongoDB є відкритою системою управління базами даних (СКБД), яка використовує документно-орієнтовану модель бази даних, що підтримує різні форми даних [16].

Це одна з численних не реляційних технологій баз даних, які виникли в середині 2000-х років під банером NoSQL для використання у великих додатках та інших місцях обробки, що містять дані, які не вписуються в жорстку реляційну модель. Замість використання таблиць і рядків, як у реляційних базах даних, архітектура MongoDB складається з колекцій і документів.

Запис в MongoDB є документом, який є структурою даних, що складається з пар поля-значення. Документи MongoDB подібні до об'єктів JavaScript Object Notation, але використовують варіант Binary JSON (BSON), який вміщує більше типів даних. Поля в документах близькі до стовпців реляційної бази даних, а значення, які вони містять, можуть бути різними типами даних, включаючи інші документи, масиви документів, згідно з керівництвом користувача MongoDB.

Документи, які також повинні включати первинний ключ як унікальний ідентифікатор, є основною одиницею даних у MongoDB. Колекції містять набори документів і функціонують як еквівалент реляційних таблиць бази даних. Колекції можуть містити будь-який тип даних, але обмеження полягає в тому, що дані в колекції не можуть бути розподілені між різними базами даних.

Оболонка mongo – це інтерактивний інтерфейс JavaScript для MongoDB, який дозволяє користувачам запитувати і оновлювати дані, а також проводити адміністративні операції. Оболонка є стандартним компонентом розподілів з відкритим вихідним кодом MongoDB. Після встановлення MongoDB користувачі підключають оболонку mongo до запущених MongoDB-екземплярів [16].

Формат зберігання та обміну даними BSON, що використовується в MongoDB, забезпечує двійкове представлення документів, подібних до JSON. Автоматична функція sharding – це ще одна ключова особливість, яка дозволяє розподілити дані в колекції MongoDB по декількох системах для горизонтальної масштабованості, оскільки обсяги даних і вимоги до пропускної здатності зростають.

СКБД NoSQL використовує єдину архітектуру майстра для узгодженості даних, з вторинними базами даних, які підтримують копії первинної бази

даних. Операції автоматично копіюються до цих вторинних баз даних для автоматичного переключення. Найновіші технології MongoDB орієнтовані на хмарні та мобільні пристрої.

Як і інші бази даних NoSQL, MongoDB не вимагає попередньо визначених схем і зберігає будь-які типи даних. Це дає користувачам можливість створювати будь-яку кількість полів у документі, що полегшує масштабування баз даних MongoDB у порівнянні з реляційними базами даних.

Однією з переваг використання документів є те, що ці об'єкти відображаються у нативні типи даних у ряді мов програмування. Крім того, наявність вбудованих документів зменшує потребу в об'єднанні баз даних, що може зменшити витрати.

Основною функцією MongoDB є його горизонтальна масштабованість, що робить її корисною базою даних для компаній, що працюють з великими додатками даних. Крім того, sharding дозволяє базі даних поширювати дані по кластеру машин. Нові версії MongoDB також підтримують створення зон даних на основі клавіші shard. MongoDB підтримує ряд механізмів зберігання даних і надає інтерфейси API, які дозволяють третім сторонам розробляти власні механізми зберігання для MongoDB [16].

СКБД також має вбудовані можливості агрегації, які дозволяють користувачам запускати код MapReduce безпосередньо в базі даних, а не запускати MapReduce на Hadoop. MongoDB також має власну файлову систему GridFS, подібну до Hadoop Distributed File System (HDFS), в першу чергу для зберігання файлів, розмір яких перевищує обмеження розміру BSON на 16 МБ на документ. Ці подібності дозволяють використовувати MongoDB замість Hadoop, хоча програмне забезпечення бази даних інтегрується з Hadoop, Spark та іншими фреймворками обробки даних.

Хоча переваг багато, є деякі недоліки для MongoDB. Завдяки автоматичній стратегії відмови користувач встановлює лише один головний вузол у кластері MongoDB. Якщо майстер виходить з ладу, підлеглий вузол автоматично перетвориться на новий. Цей перемикач забезпечує

безперервність, але не миттєву – цей процес може тривати до однієї хвилини. Для порівняння, база даних Cassandra NoSQL підтримує кілька головних вузлів. Один головний вузол MongoDB також обмежує швидкість запису даних у базу даних. Дані запису повинні бути записані на майстер і запис нової інформації в базу даних обмежений ємністю цього головного вузла. Інша потенційна проблема полягає в тому, що MongoDB не забезпечує повну цілісність посилення за допомогою обмежень зовнішнього ключа, які можуть вплинути на узгодженість даних. Крім того, аутентифікація користувача за замовчуванням не включена в бази даних MongoDB, що відповідає популярності розробників за технологією [16].

MongoDB доступний у спільноті та комерційних версіях через постачальника MongoDB Inc. MongoDB Community Edition є відкритим випуском, в той час як MongoDB Enterprise Server надає додаткові функції безпеки, механізм зберігання в пам'яті, функції адміністрування та аутентифікації та можливості моніторингу через Ops Manager.

Графічний інтерфейс користувача (GUI) під назвою MongoDB Compass надає користувачам можливість працювати з структурою документів, виконувати запити, індексні дані та багато іншого. MongoDB Connector для BI дозволяє користувачам підключати базу даних NoSQL до своїх засобів бізнес-аналітики для візуалізації даних і створення звітів за допомогою SQL-запитів.

Слідом за сторонами інших постачальників баз даних NoSQL, MongoDB Inc. запустила в 2016 році базу даних у хмарі як сервіс під назвою MongoDB Atlas. Atlas працює на AWS, Microsoft Azure та Google Cloud Platform. Зовсім недавно MongoDB випустила платформу Stitch для розробки додатків на MongoDB Atlas, з планами розширити її на локальні бази даних.

2.5 Формат передачі даних JSON

JSON — текстовий формат обміну даними, заснований на мультипарадигмальній мові програмування. Його основне призначення полягає у зберіганні і передачі структурованого потоку інформації [17].

За допомогою простих правил формування конструкцій із символів в JavaScript, людина може забезпечити легкий і надійний спосіб зберігання будь-якого виду інформації. Це може бути звичайне число, цілі рядки або величезна кількість різних об'єктів, виражених в простому тексті.

Крім цього, формат JSON використовується для об'єднання між собою об'єктів і структури даних у вигляді набору компонентів, формуючи тим самим програмні одиниці, що дозволяють зберігати і обробляти складні записи, що складаються з декількох змінних різного типу. Після того як файл створений, рядки що містяться в ньому досить легко перенаправляти в просторі Інтернет через будь-які шляхи передачі даних. Це пов'язано з тим, що рядок являє собою звичайний текст. Незважаючи на можливість використання практично у всіх скриптових мовах, його назва відноситься до JavaScript [17].

Інструмент має такі переваги:

- охоплює порівняно невеликий обсяг, компактний;
- текстовий зміст може легко створюватися і використовуватися обчислювальною технікою і людиною;
- можна без особливих зусиль перетворити в структуру практично для всіх видів формальних мов, що використовуються для створення комп'ютерних програм;
- більшість мов програмування, таких як JavaScript, Ruby, Python або PHP, наділені функціями і спеціальними інструментами для читання і редагування файлу.

У переважній більшості випадків формат JSON використовується для передачі інформації від сервера до браузера. Цей процес, як правило, відбувається в «фоновому» режимі обміну браузера з веб-сервером, а доставка

здійснюється за допомогою AJAX. Це обумовлюється тим, що в процесі доставки даних відсутня необхідність перезавантажувати сторінку. У JSON типи даних підрозділяються на кілька категорій: прості і складні. До першого виду можна віднести, перш за все, текстові рядки і числа, до другого – об'єкти.

В цілому виділяють шість основних типів.

Числовий тип. При цьому числа можуть бути як без знаковими цілими, так і цілими зі знаком. Зокрема, тут може міститися дробова частина і представлення дійсних чисел у вигляді дробової частини логарифма і порядку. Цей спосіб застосовується в JavaScript для всіх числових значень без винятків, проте в інших математичних бібліотеках, в яких він використовується, кодування може відбуватися з використанням зовсім інших алгоритмів.

Довільна послідовність (рядок) символів латинського алфавіту, цифр та елементів пунктуації (від нуля і символів юнікод). Кожний наступний рядок відділяється від попереднього рядка за допомогою парного розділового знака – лапок («текст») або з використанням символу, з написанням, зворотним по відношенню до звичайного символу, косою риси.

Літерали або константи, що включаються безпосередньо в текст. Це може бути будь-яке значення з true і false або їх аналогів.

Масив. Він являє собою упорядкований перелік символів від нуля і далі. Кожен символ може бути представлений в будь-якій формі.

Об'єкт. Це хаотично складений набір пар ключ-значення. Виходячи з того, що основна функція об'єктів полягає в поданні абстрактного типу даних, рекомендується, щоб ключі були унікальними.

Пусте значення, тобто «Null». Інтервали між символами допускаються, якщо вони будуть використані між синтаксичними одиницями. Для цього застосовуються кілька символів: звичайне відступ, горизонтальні текстові вкладки і коса риска.

2.6 Висновки за розділом 2

У 2 розділі було розглянуто основні особливості та переваги деяких технологій розробки програмного забезпечення. Нами було зроблено висновок, що для побудови веб-застосунку є доцільним використання таких технологій: мова програмування JavaScript, серверна платформа Node.js, фреймворк EXPRESS, база даних MongoDB та формат передачі даних JSON. Дані технології досить поширені в наш час, мають добре описану та повноцінну документацію, велику кількість публікацій та є досить простими та зручними у використанні.

3 ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ВЕБ-ЗАСТОСУНКУ

3.1 Функціональні вимоги до веб-застосунок

Розроблений програмний продукт – віртуальний органайзер вільного перегляду та комфортного редагування інформації щодо графіку навчального процесу та розкладу занять у будь-який момент часу. Веб-застосунок має назву «Student Organizer».

Щоб отримати можливість використовувати веб-застосунок, користувачеві необхідно пройти аутентифікацію та авторизацію. Це надає програмному забезпеченню більш високий рівень надійності та безпеки.

Авторизовані користувачі (студенти) мають дану основну функціональність та доступ до таких веб-сторінок:

- головна сторінка з повідомленнями;
- перегляд розкладу занять;
- перегляд учасників своєї групи;
- створення нового повідомлення для своєї групи.

За для забезпечення ефективного та якісного управління веб-застосунком, у системі також наявні ролі куратора та старости, які мають дещо більшу функціональність ніж у користувачів-студентів. Куратор та староста виконують функції адміністраторів та мають можливість:

- додавання нових користувачів (лише куратор);
- редагування розкладу занять для окремої групи;
- створення нової групи;
- перегляд усієї інформації про користувачів системи.

Вимоги до технічного забезпечення користувача мінімальні, потрібна лише можливість працювати у веб-браузері, пристрої вводу та виведення інформації.

3.2 Опис архітектури та обраних технологій

3.2.1 Загальна архітектура веб-застосунку

Для даного веб-сервісу було обрано клієнт-серверну архітектуру. Клієнт-серверна архітектура умовно поділяється на дві частини: клієнт та сервер. Схему клієнт-серверної архітектури зображено на рис 3.1.

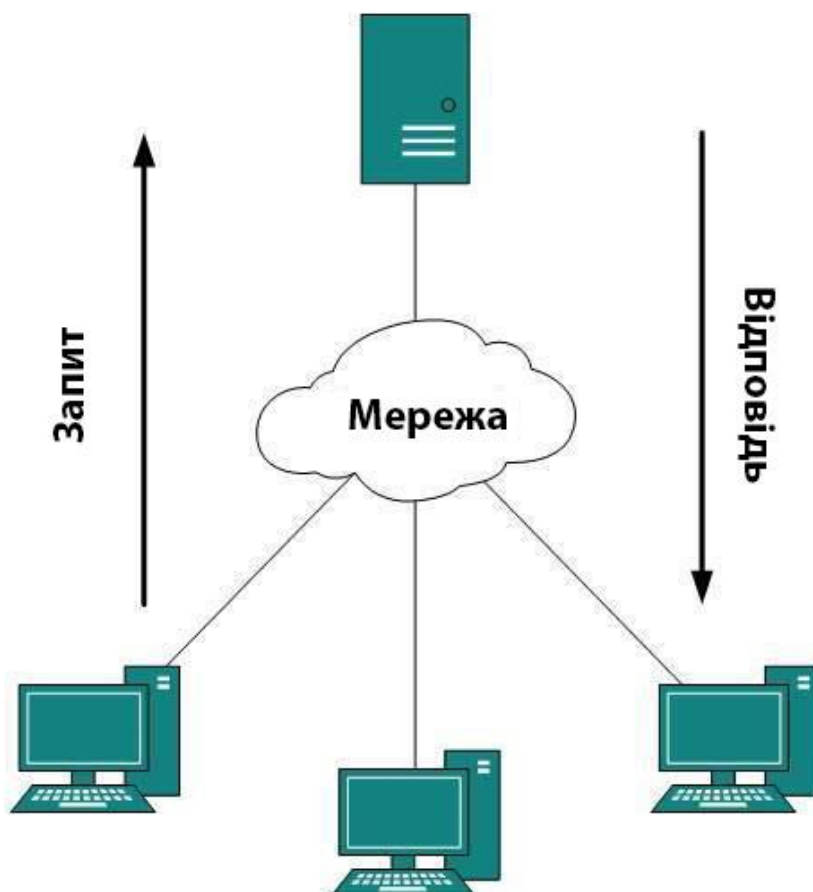


Рисунок 3.1 – Схема клієнт-серверної архітектури

Клієнти роблять запити до сервера, який оброблює, зберігає та повертає дані, які необхідні користувачеві, тому його можна вважати головною частиною даної архітектури. Існує припущення, що сервер за необхідності може існувати без юзерів, у той час як користувачі без сервера вже ніяк не здатні функціонувати. Клієнтів зазвичай набагато більше ніж серверів. Наприклад, до єдиного сервера, який працює з сайтом підключається величезна

кількість браузерів. Користувачі за допомогою програми-клієнта спілкуються з сервером, але клієнтом може також виступати робот або штучний інтелект [18].

У навантажених веб-застосунках також може бути велика кількість серверів, але це не означає, що клієнтів менше, а навпаки на такій системі найбільш імовірно дуже великий трафік.

Переваги клієнт-серверної архітектури [19]:

- виконання більшої частини роботи потужною серверною частиною при мінімумі навантаженні на клієнта;
- основна частина даних зберігається на сервері. При цьому, як правило, він краще захищений від різного виду загроз, ніж звичайний клієнтський ПК;
- можливість більш чіткого розмежування повноважень доступу до різних рівнів інформаційної системи (кожному клієнту – свій рівень доступу);
- кросплатформеність, завдяки якій клієнт може працювати з ресурсами сервера незалежно від операційної системи;
- зменшення навантаження на Інтернет-мережу з огляду на те, що клієнт в основному передає серверу команди, який їх виконує.

Недоліки клієнт-серверної архітектури [19]:

- вихід з ладу сервера може привести до непрацездатності всієї системи, що його використовує;
- висока вартість серверного обладнання та його обслуговування (зокрема, може знадобитися окремий фахівець для обслуговування);
- високе навантаження на серверне обладнання і канал зв'язку до нього.

У підсумку, можна сказати що клієнт-серверна архітектура не тільки поділяє машини на клієнтські та серверні, а має більш глибоке значення. Дана архітектура розподіляє навантаження і розділяє роботу на серверній та клієнтській частині, що надає більшій швидкості та ефективності роботи клієнтів.

3.2.2 Основні принципи технології клієнтської частини

Клієнтська частина (front-end) веб-застосунку була створена за принципом SPA (Single Page Application), тобто односторінкового застосунку.

Односторінковий застосунок SPA (Single Page Application), або односторінковий інтерфейс SPI (Single Page Interface) – це веб-застосунок або веб-сайт, який розміщується на одній веб-сторінці з метою забезпечити користувачу імітацію роботи схожою до користування десктопною програмою [20].

В односторінковому застосунку весь необхідний код (HTML, JavaScript та CSS) завантажується разом зі сторінкою, або динамічно довантажується за потребою, зазвичай у відповідь на дії користувача. Сторінка не оновлюється і не перенаправляє користувача до іншої сторінки у процесі роботи з нею. Взаємодія з односторінковим застосунком часто включає в себе динамічний зв'язок з веб-сервером [13].

На даний період існує досить багато фреймворків та бібліотек для створення SPA. Для розробки клієнтської частини проекту нами було обрано JavaScript-бібліотеку для написання односторінкових застосунків React (ReactJS), яка була створена Джорданом Валке, розробником програмного забезпечення в компанії Facebook [21].

Бібліотеку React було обрано з ряду причин [22]:

- легкість вивчення React, що обумовлено простотою його синтаксису. Інженери лише повинні знати JavaScript та згадати свої навички написання HTML. Немає потреби в глибокому вивченні TypeScript, як у випадку з Angular;
- високий рівень гнучкості і максимальна чутливість React;
- віртуальна DOM (document object model), яка дозволяє впорядковувати документи форматів HTML, XHTML або XML в дерево, яке найкраще підходить веб-браузерам для аналізу різних елементів веб-застосунку;

- можливість легко працювати при високих навантаженнях у поєднанні з ES6 / 7 ReactJS;
- зв'язування даних від великих до менших. Це означає такий потік даних, при якому дочірні елементи не можуть впливати на батьківські дані;
- наявність стовідсоткової JavaScript-бібліотеки з відкритим вихідним кодом, яка отримує безліч щоденних оновлень та покращень відповідно до відгуків розробників по всьому світу;
- неймовірна легкість, так як дані, які виконуються на стороні користувача, можуть легко бути представлені на стороні сервера в той же самий час;
- як правило, дуже проста міграція між версіями.

Для дизайну серед безлічі CSS-фреймворків, наборів стилізованих компонентів та дизайн-систем нами було обрано Ant Design. Ця дизайн-система має у собі досить багато вже стилізованих компонентів для обраної раніше бібліотеки ReactJS, дає можливість дуже легко їх кастомізувати та додавати необхідну функціональність. На офіційному сайті наявна дуже детальна документація з великою кількістю прикладів, що є незаперечним плюсом на догоду скорочення часу на розробку дизайну [23].

3.2.3 Основні принципи технології серверної частини

Серверна частина (back-end) веб-застосунку представлена REST API (Representational State Transfer Application Programming Interface) – архітектурним стилем взаємодії компонентів розподіленого додатку в мережі.

REST (Representational State Transfer) є підходом до архітектури мережеских протоколів, який був описаний Роем Філдінгом у дисертації Architectural Styles and the Design of Network-based Software Architectures (2000 p.) [24].

При використанні архітектурного стилю REST кожна одиниця даних (Ресурс) визначається за допомогою URL (Uniform Resource Locator), який є ключем до цієї одиниці інформації. В більшості випадків, дані відправляються в

форматі JSON (JavaScript Object Notation), але можна використовувати і інші формати як HTML, PNG, PDF і інші.

Особливістю даного виду архітектури є те, що вона заснована на протоколі HTTP, через що управління ресурсами веб-застосунку відбувається з допомогою таких методів, як GET (отримання даних), POST (надсилання даних), PUT (вставка, оновлення даних), DELETE (видалення даних).

GET – метод, який застосовується для запиту вмісту ресурсу. Параметри запиту передаються після символу «?» (/api/resource?name=su). якщо ресурс не був знайдений, повертається відповідь 404 (Not found).

POST – використовується для завантаження нових ресурсів на сервер. Вміст даних при завантаженні поміщається в тіло запиту. При успішному завантаженні даних сервер повертає 200 (Ok) або 201 (Created).

PUT – оновлює ресурс по заданому ідентифікатором. тіло запиту повинно містити оновлені дані оригіналу. Якщо ресурс не існує, створює його і передає відповідь 201 (Created). В іншому випадку, змінює його з відповіддю 200 (Ok).

DELETE – видаляє ресурс, вказаний за допомогою ідентифікатора. як правило, нічого не повертає.

Переваги REST-архітектури [25]:

- компоненти систем взаємодіють в набагато більшому масштабі;
- всі інтерфейси загальні;
- частини можливо впроваджувати незалежно одну від одної;
- наявні проміжні елементи, які знижують відсоток затримки і посилюють безпеку з'єднання.

Недоліки REST-архітектури [26]:

- надмірна вибірка: найчастіше кінцева точка API надає на запит зайву інформацію, яка не потрібна клієнтові для здійснення дій (необхідно отримати тільки ім'я користувача, але метод повідомляє ще його вік, телефон та електронну пошту);
- недостатня вибірка: клієнту доводиться робити декілька запитів, щоб отримати всю інформацію, яка йому потрібна (з однієї кінцевої точки API

клієнт отримує особисті дані користувача, а з іншої – інформацію щодо зроблених ним замовлень).

На рисунку 3.2 зображена схема роботи веб-застосунку, що відповідає REST-архітектурі.

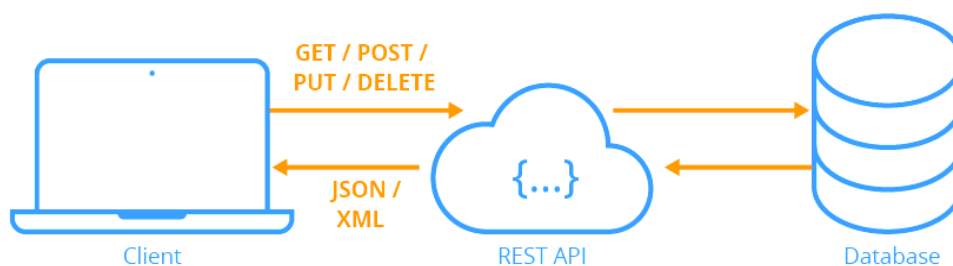


Рисунок 3.2 – Схема REST архітектури

Для скорочення терміну роботи та підвищення комфорту під час розробки для серверної частини нами було обрано фреймворк Express.

Express – це лаконічний та гнучкий веб-фреймворк для додатків Node.js, що пропонує великий комплект функцій для мобільних і веб-застосунків. Маючи в своєму арсеналі багато службових методів HTTP і проміжних оброблювачів, він надає можливість розробити надійний API просто та швидко [15].

Node.js – середовище виконання JavaScript, платформа з відкритим кодом, що використовується для реалізації серверної частини застосунків [14].

Однією з переваг Node.js є асинхронність в поєднанні з подієво-орієнтованим підходом до програмування, який заснований на зовнішніх діях. Тобто виконання програми залежить від дій користувача. Таке рішення спрощує програмування веб-застосунків (онлайн-чатів, ігрові програм, інтерактивних веб-сервісів), які засновані на роботі з використанням введення-виведення.

Також, до переваг платформи Node.js відносять:

- простий і широко відомий JavaScript. Звичайно, платформа передбачає власні інструменти та особливості, наприклад, тут немає браузерних

API, cookie або DOM, однак присутні власні бібліотеки. В основному використовуються можливості і синтаксис всім звичного JavaScript;

- велика стандартна бібліотека. Платформа від самого початку мала широкий набір можливостей, а в нових версіях бібліотека поповнюється і поліпшується;

- велика кількість зовнішніх бібліотек і готових модулів.

Використання пакетного менеджера NPM дозволяє постійно розвивати екосистему Node. Сьогодні число пакетів в ньому перевищило 500 тисяч і постійно зростає. Потрібно розуміти, що на відміну від JavaScript, що виконується у браузері, під час застосування Node.js додаток має можливість звертатися до великої кількості глобальних об'єктів, в тому числі document та window. Як наслідок, програма може звертатися до вінчестера і файлової системи користувача, а також до програмних рішень та бібліотек, що знаходяться на комп'ютері. Така особливість платформи Node помітно підвищує можливості її взаємодії з зовнішніми компонентами.

Зазвичай цю платформу використовують для створення веб-сервісів, яким необхідний інтенсивний інформаційний обмін з користувачами, а також для реалізації чатів, систем спільної роботи, соціальних мереж і т.д. Велика кількість програм, розроблених на Node.js мають дві складові – серверну та клієнтську частину. Для розуміння причин такого використання варто трохи докладніше зупинитися на асинхронності і можливостях подієво-орієнтованої парадигми. При цьому програмування зводиться до того, що ви вказуєте програмі, які дії вона повинна виконати, а також яку функцію вона повинна викликати після виконання цих дій. Поки Node.js чекає виконання коду і виклику функції, вона може виконувати інші завдання. При цьому потік в очікуванні запиту блокуватися не буде. Таким чином, програма може звернутися із запитом до бази даних сервера, а поки очікується відповідь, обробити інші запити. В результаті одночасно обробляються тисячі з'єднань, кожне з яких витрачає мало часу і ресурсів. Це помітно прискорює обмін інформацією з сервером і, як підсумок, роботу браузерного додатку.

Одним із найважливіших компонентів серверної частини веб-застосунку є база даних. Нами було розглянуто такі типи баз даних, як реляційна та нереляційна.

Реляційні бази даних використовуються для зберігання і надання доступу до взаємопов'язаних елементів інформації. Реляційні бази даних засновані на реляційній моделі – інтуїтивно зрозумілому, наочному табличному способі представлення даних. Кожен рядок, що міститься в таблиці такої бази даних, є записом з унікальним ідентифікатором, який називають ключем. Стовпці таблиці мають атрибути даних, а кожен запис зазвичай містить значення для кожного атрибута, що дає можливість легко встановлювати взаємозв'язок між елементами даних [27].

У нереляційній базі даних, на відміну від більшості традиційних систем баз даних, не використовується таблична схема рядків і стовпців. У базах даних такого типу застосовується модель зберігання, яка оптимізована під конкретні вимоги типу даних, що зберігаються. Наприклад, дані можуть зберігатися як прості пари "ключ – значення", документи JSON або граф, що складається з ребер і вершин [28].

При розробці веб-застосунку було вирішено відійти від стандарту та застосувати нереляційну базу даних NoSQL (Not only SQL), яка є документно-орієнтованою. NoSQL бази даних, у порівнянні з SQL, набагато краще масштабуються, не накладають обмежень на типи даних, які можна зберігати разом, і дозволяють додавати нові типи при зміні потреб. Кожне рішення у рамках технології NoSQL використовує свою власну модель. NoSQL бази даних все більше і більше використовуються у веб-застосунках.

Акцент було зроблено на MongoDB – відкритому програмному забезпеченні, яке є вільною від схем, документно-орієнтованою, розширюваною та високопродуктивною базою даних, що використовує мову програмування C++. MongoDB зберігає всі дані у форматі бінарного JSON (BSON) [29].

MongoDB розробляється з жовтня 2007 року компанією 10gen, яку було організовано керівництвом DoubleClick, та часто застосовується у сучасних проєктах [16].

Основні характеристики та можливості MongoDB:

- підтримує масштабованість та відмовостійкість;
- зручно формувати запити завдяки гнучкій мові;
- можливість формувати динамічні запити;
- швидкі оновлення;
- підтримує MapReduce;
- прості та потужні JSON-подібні схеми (документо-орієнтоване сховище);
- повністю підтримує індекси;
- операції, що модифікують дані в БД журналюються;
- профілюються запити.

Для взаємодії Node.js додатку з базою даних MongoDB було обрано ODM Mongoose [30].

Mongoose – це JavaScript бібліотека моделювання об'єктів для MongoDB в середовищі NodeJS. Mongoose забезпечує взаємодію додатків Node.js з MongoDB. Mongoose надає схеми, які визначають структуру і тип даних документів, тим самим забезпечують необхідну структуру БД. Крім того, mongoose на підставі схеми забезпечує створення моделі, після збереження якої створюється документ з властивостями, які визначені в схемі.

3.3 Висновки за розділом 3

В розділі 3 розглянуто засоби програмної реалізації веб-застосунку. Для веб-застосунку нами було обрано клієнт-серверну архітектуру, однією з переваг якої є кросплатформність.

Клієнтська частина (front-end) веб-застосунку була створена за принципом SPA. При розробці було використано JavaScript-бібліотеку для

написання односторінкових застосунків React (ReactJS). Підставою для цього було легкість вивчення React, що обумовлено простотою його синтаксису, та високий рівень гнучкості і максимальна чутливість React.

Серверна частина (front-end) веб-застосунку представлена REST API. Архітектурний стиль REST було обрано завдяки загальності інтерфейсів та можливості впровадження частин незалежно одну від одної. Для розробки серверної частини нами було обрано платформу Node JS, яка використовує можливості і синтаксис JavaScript та має велику кількість зовнішніх бібліотек і готових модулів. Для скорочення терміну роботи та підвищення комфорту під час розробки для серверної частини було обрано фреймворк Express. Підставами для обрання бази даних MongoDB було документо-орієнтоване сховище (проста та потужна JSON-подібна схема даних) та гнучка мова для формування запитів.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ

4.1 Функціональні вимоги до веб-застосунку

Розроблений програмний продукт – віртуальний органайзер вільного перегляду та комфортного редагування інформації щодо графіку навчального процесу та розкладу занять у будь-який момент часу. Веб-застосунок має назву «Student Organizer».

Щоби отримати можливість використовувати веб-застосунок, користувачеві необхідно пройти аутентифікацію та авторизацію. Це надає програмному забезпеченню більш високий рівень надійності та безпеки.

Авторизовані користувачі (студенти) мають дану основну функціональність та доступ до таких веб-сторінок:

- головна сторінка з повідомленнями;
- перегляд розкладу занять;
- перегляд учасників своєї групи;
- створення нового повідомлення для своєї групи.

За для забезпечення ефективного та якісного управління веб-застосунком, у системі також наявні ролі куратора та старости, які мають дещо більшу функціональність ніж у користувачів-студентів. Куратор та староста виконують функції адміністраторів та мають можливість:

- додавання нових користувачів (лише куратор);
- редагування розкладу занять для окремої групи;
- створення нової групи;
- перегляд усієї інформації про користувачів системи.

Вимоги до технічного забезпечення користувача мінімальні, потрібна лише можливість працювати у веб-браузері, пристрої вводу та виведення інформації.

4.2 Процес програмної реалізації веб-застосунку

Для спрощення процесу та можливості повернутись до попередньої версії коду було застосовано систему GIT.

Було створено 2 публічних репозиторія на github.com:

- Student-organizer-back (https://github.com/mrdzukan/student-organizer-back.git);
- Student-organizer-front (https://github.com/mrdzukan/student-organizer-front.git).

Перейшовши на посилання можна ознайомитись з вихідним кодом та деталями процесу розробки (кнопка commits, рис. 4.1).

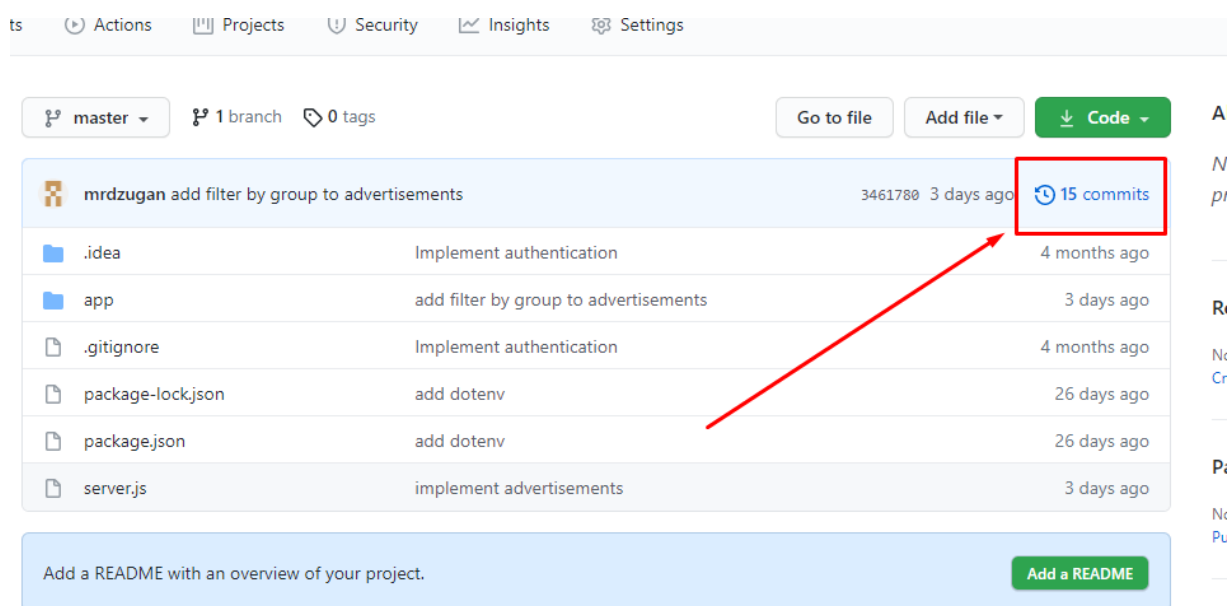


Рисунок 4.1 – Кнопка commits

4.3 Структурна схема веб-застосунку

Розглянемо структурну схему веб-застосунку, зокрема схеми серверної та клієнтської частин.

Структурну схему серверної частини зображено на рис 4.2.

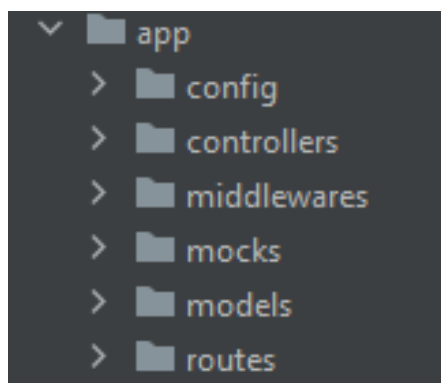


Рисунок 4.2 – Структурна схема серверної частини

Основні компоненти серверної частини згруповані по папкам у каталозі `app`:

- `config` – містить конфігураційні файли;
- `controllers` – містить всі контролери програми, в яких виконується бізнес-логіка;
- `middlewares` – містить усі файли проміжних обробок;
- `mocks` – містить усі заготовлені дані, які буде перенесено до бази даних під час першого запуску проекту;
- `models` – містить усі моделі (сутності);
- `routes` – файли з описом усіх інтерфейсів (endpoints), що будуть доступні у програмі.

Структурну схему серверної частини зображено на рис 4.3.

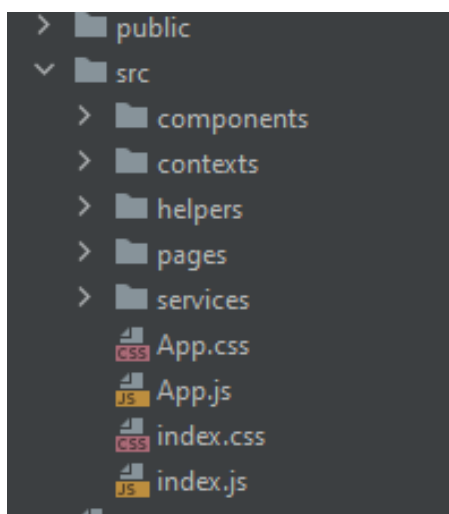


Рисунок 4.3 – Структурна схема клієнтської частини

Основні компоненти серверної частини згруповані по папкам у каталозі `src`:

- `components` – містить React-компоненти, що можуть використовуватися повсюдно;
- `contexts` – містить всі контексти клієнтської частини, які дозволяють передавати дані по компонентах;
- `helpers` – містить всі файли з допоміжними функціями, котрі також можна перевикористати будь-де;
- `services` – містить файли з функціями, що звертаються до серверної частини;
- `pages` – містить React-компоненти основних сторінок проекту;
- `index.js` – вхідна точка проекту;
- `App.js` – файл з маршрутизацією по сторінках клієнтської частини;
- `App.css` & `index.css` - відповідають за глобальні стилі.

4.4 Моделювання бази даних

Спираючись на функціонал, який повинен забезпечувати проект, база даних потребує наявності сутностей, які було визначено у даному розділі.

Користувач – особа, що виконує певні дії у системі. У базі даних було визначено сутність під назвою `User`, яка має наступні поля:

- `App.css` & `index.css` - відповідають за глобальні стилі;
- `_id` – ідентифікатор користувача (генерується автоматично під час створення нового користувача) ;
- `name` – ім'я користувача;
- `surname` – прізвище користувача;
- `faculty` – ідентифікатор факультету, на якому вчиться або працює користувач;
- `group` – ідентифікатор групи, у якій вчиться користувач;
- `email` – поштова адреса користувача;

- password – пароль користувача;
- roles – набір ролей (рівнів доступу), що має користувач.

Схему моделі User зображено на рис. 4.4.

```
const User = mongoose.model(
  'User',
  new mongoose.Schema({
    name: String,
    surname: String,
    faculty: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Faculty'
    },
    group: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Group'
    },
    email: String,
    password: String,
    roles: [{
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Role'
    }]
  })
);
```

Рисунок 4.4 – Схема моделі User

Оголошення – сутність, яка дозволяє сповістити користувачів щодо актуальних новин стосовно навчання. Сутність Advertisement має наступні поля:

- _id – ідентифікатор оголошення (генерується автоматично під час створення нового оголошення);
- title – заголовок оголошення;
- description – основний текст оголошення;
- user – ідентифікатор користувача, що створив оголошення;
- group – ідентифікатор групи, членам якої буде висвітлено це оголошення.

Схему моделі Advertisement зображено на рис. 4.5.

```
const Advertisement = mongoose.model(
  'Advertisement',
  new mongoose.Schema({
    title: String,
    description: String,
    user: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'User'
    },
    group: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Group'
    }
  }), { timestamps: true })
);
```

Рисунок 4.5 – Схема моделі Advertisement

Факультет – це сутність факультету, який наявний в університеті. Сутність Faculty має наступні поля:

- `_id` – ідентифікатор факультету (генерується автоматично під час створення нового факультету);
- `fullName` – повна назва факультету;
- `abbreviation` – аббревіатура назви факультету;
- `groups` – набір ідентифікаторів груп, які належать до цього факультету.

Схему моделі Faculty зображено на рис. 4.6.

```
const Faculty = mongoose.model(
  'Faculty',
  new mongoose.Schema({
    fullName: String,
    abbreviation: String,
    groups: [{
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Group'
    }]
  })
);
```

Рисунок 4.6 – Схема моделі Faculty

Група – це сутність групи, яка належить до певного факультету у університеті. Сутність Group має наступні поля:

- `_id` – ідентифікатор групи (генерується автоматично під час створення нової групи);
- `name` – назва (або номер) групи;
- `faculty` – ідентифікатор факультету, до якого належить група;
- `members` – набір ідентифікаторів користувачів, які є членами цієї групи.

Схему моделі Group зображено на рис. 4.7.

```
const Group = mongoose.model(
  'Group',
  new mongoose.Schema({
    name: String,
    faculty: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Faculty'
    },
    members: [
      {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'User'
      }
    ]
  })
);
```

Рисунок 4.7 – Схема моделі Group

Роль – це сутність, яка надає користувачам певні рівні доступу. Сутність Role має лише 2 поля:

- `_id` – ідентифікатор ролі (генерується автоматично під час створення нової ролі);
- `name` – ім'я ролі.

Схему моделі Role зображено на рис. 4.8.

```
const Role = mongoose.model(
  'Role',
  new mongoose.Schema({
    name: String
  })
);
```

Рисунок 4.8 – Схема моделі Role

Розклад – це найскладніша сутність у проекті, що включає у себе розклад занять по чисельнику та знаменнику. Кожне заняття включає у себе назву пари та ім'я викладача, що її викладає. Сутність Timetable має наступні поля:

- `_id` – ідентифікатор розкладу (генерується автоматично під час створення нового розкладу);
- `group` – ідентифікатор групи, до якої відноситься цей розклад
- `schedule` – розклад занять (пар);
- `numerator` – розклад занять по чисельнику. Являє собою масив з днями тижня (з понеділка по неділю);
- `dayOfWeek` – день тижня;
- `lessons` – набір занять у цей день тижня;
- `subject` – назва пари;
- `teacher` – ім'я викладача;
- `denominator` – розклад занять по знаменнику. Являє собою масив з днями тижня (з понеділка по неділю);
- `dayOfWeek` – день тижня;
- `lessons` – набір занять у цей день тижня;
- `subject` – назва пари;
- `teacher` – ім'я викладача.

Схему моделі Timetable зображено на рис. 4.9.

```

const Timetable = mongoose.model(
  'Timetable',
  new mongoose.Schema({
    group: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Group'
    },
    schedule: {
      numerator: [
        {
          dayOfWeek: Number,
          lessons: [{
            subject: String,
            teacher: String
          }]
        }
      ],
      denominator: [
        {
          dayOfWeek: Number,
          lessons: [{
            subject: String,
            teacher: String
          }]
        }
      ]
    }
  })
);

```

Рисунок 4.9 – Схема моделі Timetable

4.5 Безпека веб-застосунку

Будь який веб-застосунок, як правило, має бути безпечним [31].

Під безпекою веб-сервісів розуміється забезпечення збереження даних і їх недоступність для сторонніх осіб, а також здатність додатки зберігати працездатність при кібератаках і не піддаватися зараження вірусами. Для відповідності цим принципам було застосовано такі процеси, як авторизація та аутентифікація [32].

Авторизація – надання певній особі або групі осіб прав на виконання певних дій.

Аутентифікація – процедура перевірки автентичності, наприклад перевірка справжності користувача шляхом порівняння введеного ним пароля з паролем, збереженим в базі даних [33].

До веб-застосунку «Student Organizer» нами було застосовано технологію JWT (JSON Web Token).

JSON Web Token (JWT) – це JSON об'єкт, який визначений у відкритому стандарті RFC 7519. Він вважається одним з безпечних способів передачі інформації між двома учасниками. Для його створення необхідно визначити заголовок (header) із загальною інформацією по токени, корисні дані (payload), такі як ід користувача, його роль і т.д. і підписи (signature) [34].

Алгоритм застосування JWT зображено на рис. 4.10.



Рисунок 4.10 – Алгоритм застосування JWT

Веб-застосунок «Student Organizer» використовує JWT для перевірки аутентифікації користувача в такий спосіб.

Спершу користувач заходить на сервер аутентифікації за допомогою аутентифікаційного ключа (це може бути пара логін / пароль, або Facebook ключ, або Google ключ, або ключ від іншого облікового запису).

Потім сервер аутентифікації створює JWT і відправляє його користувачеві.

Коли користувач робить запит до API додатка, він додає до нього отриманий раніше JWT.

Коли користувач робить API запит, додаток може перевірити за поданою із запитом JWT чи є користувач тим, за кого себе видає. У цій схемі серверна частина додатку налаштована так, що зможе перевірити, чи є вхідний JWT саме тим, що був створений сервером аутентифікації.

4.6 Висновки за розділом 4

В розділі 3 розглянуто структурну схему веб-застосунку, зокрема схеми серверної та клієнтської частин та застосування технології JWT.

5 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ

В даному розділі розглядається вигляд та функціонал створеного веб-застосунку «Student Organizer».

Користувач взаємодіє зі сторінкою веб-застосунку та обирає те, що йому потрібно. Органайзер має простий і легкий для використання інтерфейс, якому немає аналогів. При оновленні веб-застосунку не виникне ніяких труднощів, оскільки загальна структурна схема залишається незмінною.

5.1 Вхід до системи

Головною сторінкою веб-застосунку «Student Organizer» виступає сторінка оголошень, але щоб отримати до неї доступ потрібно увійти в систему за допомогою форми авторизації (рис. 5.1).

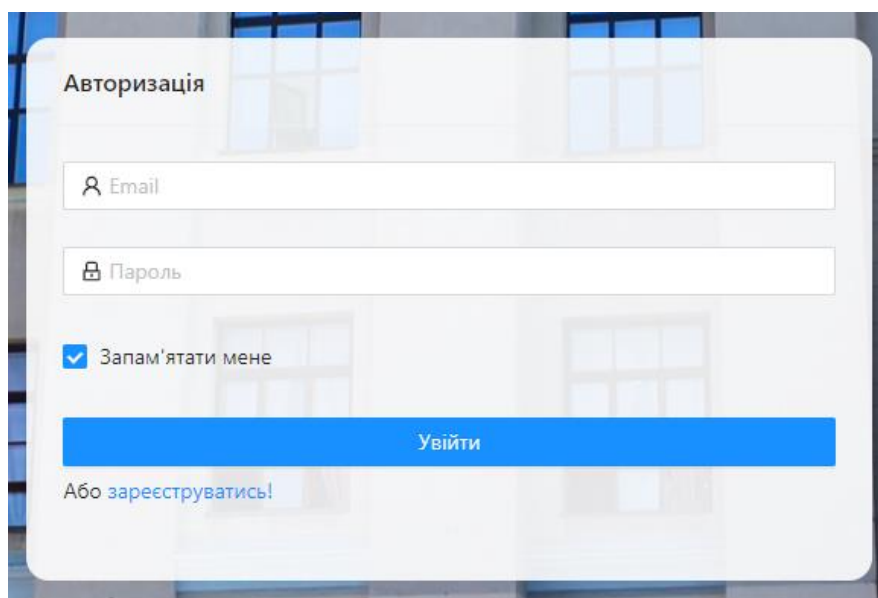
The image shows a login form titled "Авторизація" (Authorization) overlaid on a blurred background of a building. The form contains two input fields: "Email" with an envelope icon and "Пароль" (Password) with a lock icon. Below these is a checkbox labeled "Запам'ятати мене" (Remember me) which is checked. A large blue button labeled "Увійти" (Login) is positioned below the checkbox. At the bottom of the form, there is a link that says "Або зареєструватись!" (Or register!).

Рисунок 5.1 – Форма авторизації

Якщо користувач вперше користується додатком, або ще не створив особистого акаунту в системі, він повинен пройти форму реєстрації, усі поля якої є обов'язковими для заповнення. Користувачу пропонується обрати серед

існуючих у базі факультетів та групу, у якій він навчається. Передбачається, що потрібний користувачу факультет буде наявний у списку для вибору, а натомість групи, у якій навчається користувач може не існувати у базі даних. В такому випадку користувач може обрати пункт «У списку немає моєї групи», і йому буде запропоновано створити нову групу одразу після реєстрації. Форма реєстрації зображена на рис. 5.2.

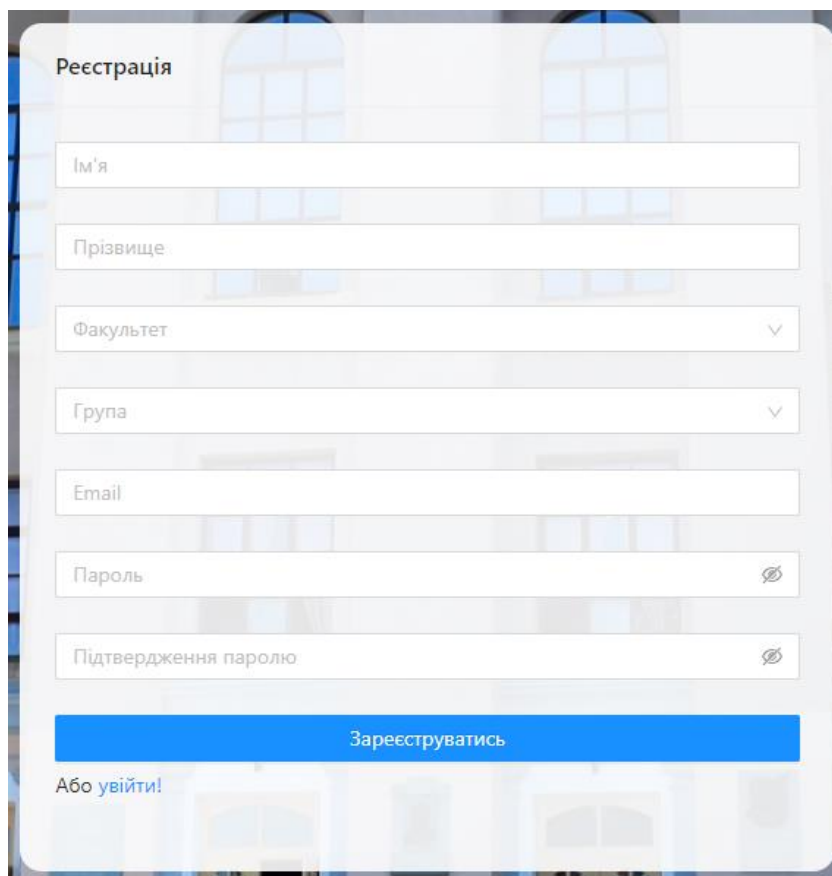
The image shows a registration form titled "Реєстрація" (Registration). It contains several input fields: "Ім'я" (Name), "Прізвище" (Surname), "Факультет" (Faculty) with a dropdown arrow, "Група" (Group) with a dropdown arrow, "Email", "Пароль" (Password) with an eye icon, and "Підтвердження паролю" (Confirm password) with an eye icon. Below these fields is a blue button labeled "Зареєструватись" (Register). At the bottom left, there is a link that says "Або увійти!" (Or log in!). The background of the form is a blurred image of a building with arched windows.

Рисунок 5.2 – Форма реєстрації

Після успішної реєстрації ми попадаємо на сторінку оголошень, нас зустрічає привітальне повідомлення у правому верхньому кутку, яке зображено на рис. 5.3.

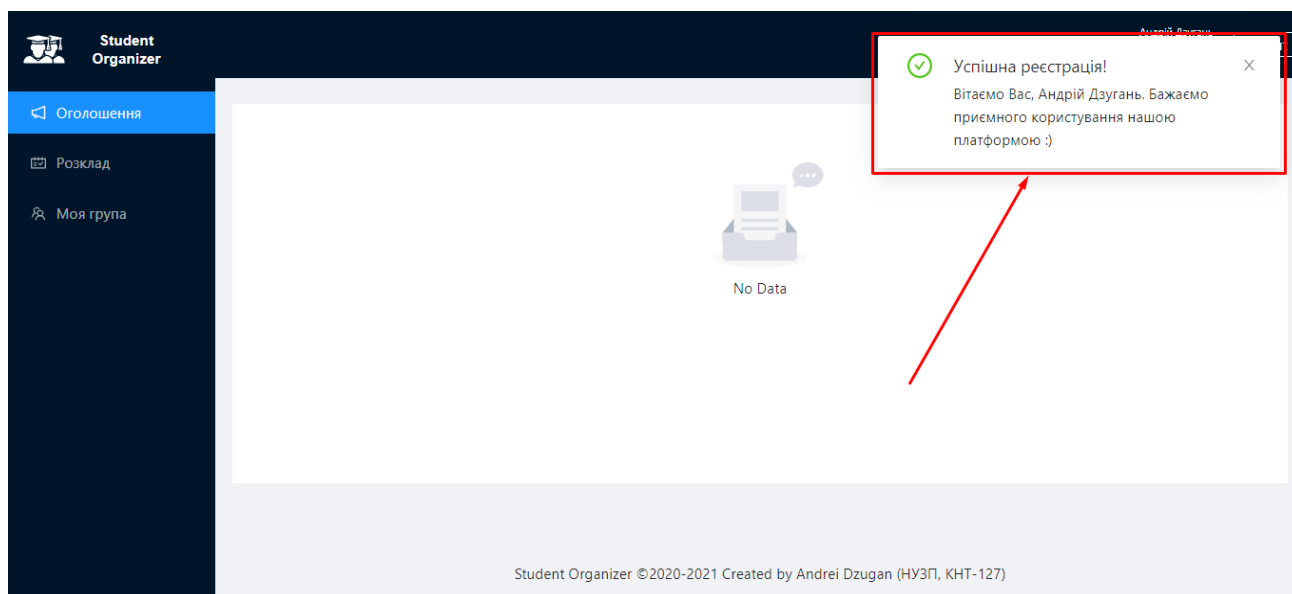


Рисунок 5.3 – Сторінка оголошень, привітальне повідомлення

Для досконального тестування, до особистого акаунту нашого користувача було присвоєно усі існуючі ролі: студент, староста, куратор.

Одразу після закриття або автоматичного зникнення повідомлення у правому верхньому кутку ми бачимо ім'я та прізвище користувача, якого створили раніше, та кнопки «Створити групу» і «Вийти». Цей блок зображено на рис. 5.4.

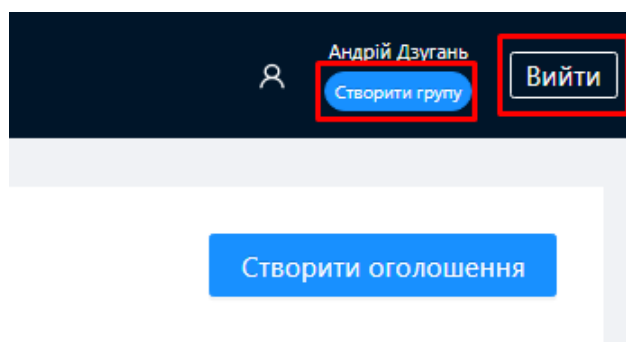


Рисунок 5.4 – Інформативний блок користувача та кнопки «Створити групу» і «Вийти»

5.2 Створення нової групи

Почнемо з створення нової групи, натиснувши на відповідну кнопку. Перед нами з'являється модальне вікно, що пропонує заповнити лише одне поле – назву групи. Назва та аббревіатура факультету була автоматично додана згідно обраного під час реєстрації факультету. Модальне вікно створення нової групи зображено на рис. 5.5.

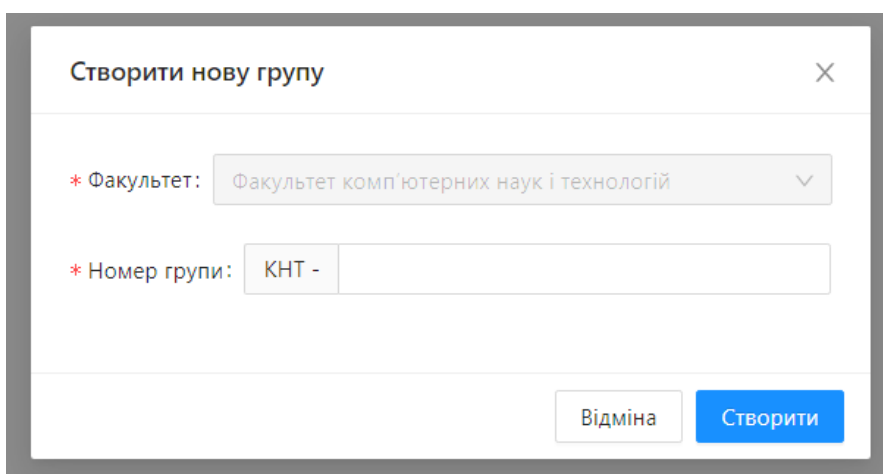


Рисунок 5.5 –Модальне вікно створення нової групи

Після вводу назви (номера) групи та натискання кнопки «Створити», система сповістить нас про успішне створення нової групи. Варто зазначити, що одразу після цього поточного користувача буде додано до створеної групи. Оновлений інформативний блок користувача зображено на рисунку 5.6.

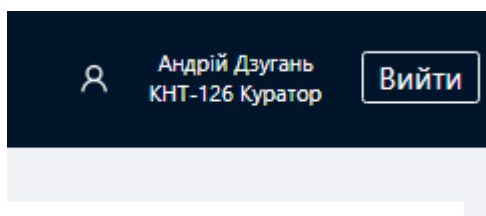


Рисунок 5.6 – Оновлений інформативний блок користувача після створення нової групи з номером «126»

5.3 Створення нового оголошення

Відтепер нам надається можливість сповна протестувати функціонал додатку. На головній сторінці ще немає жодного оголошення, тому пропоную створити його. Натискаємо на кнопку «Створити оголошення», перед нами з'являється модальне вікно створення оголошення (рис. 5.7). Варто звернути увагу на те, що основний текст оголошення (повідомлення) є необов'язковим, можна заповнити лише заголовок.

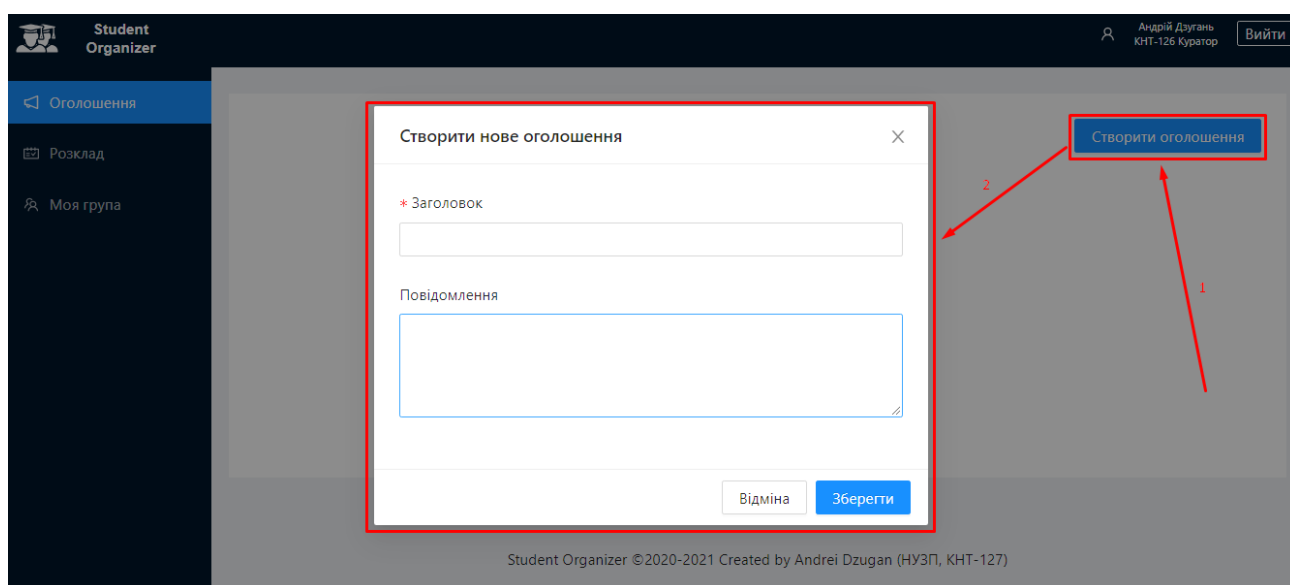


Рисунок 5.7 – Модальне вікно створення нового оголошення

Створивши нове оголошення, за допомогою кнопки «Зберегти», ми повинні побачити на сторінці щойно створене оголошення, яке зображено на рис. 5.8.

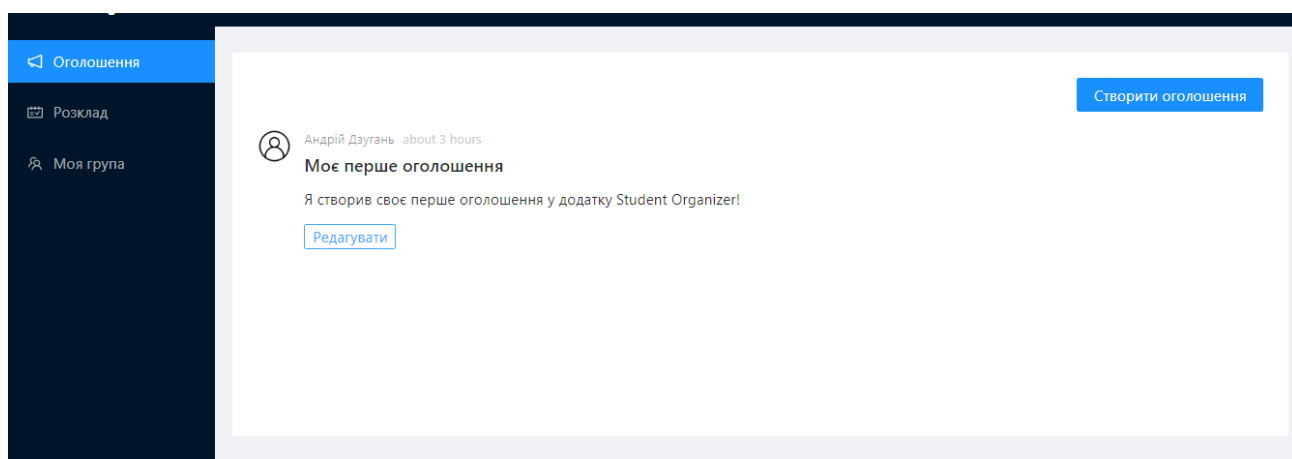


Рисунок 5.8 – Створене оголошення

5.4 Розклад занять

Оскільки сторінку оголошень ми вже розглянули, можна перейти до сторінки з розкладом занять. Так як для нашої групи ще не існує розкладу, нам пропонується його створити (рис. 5.9).

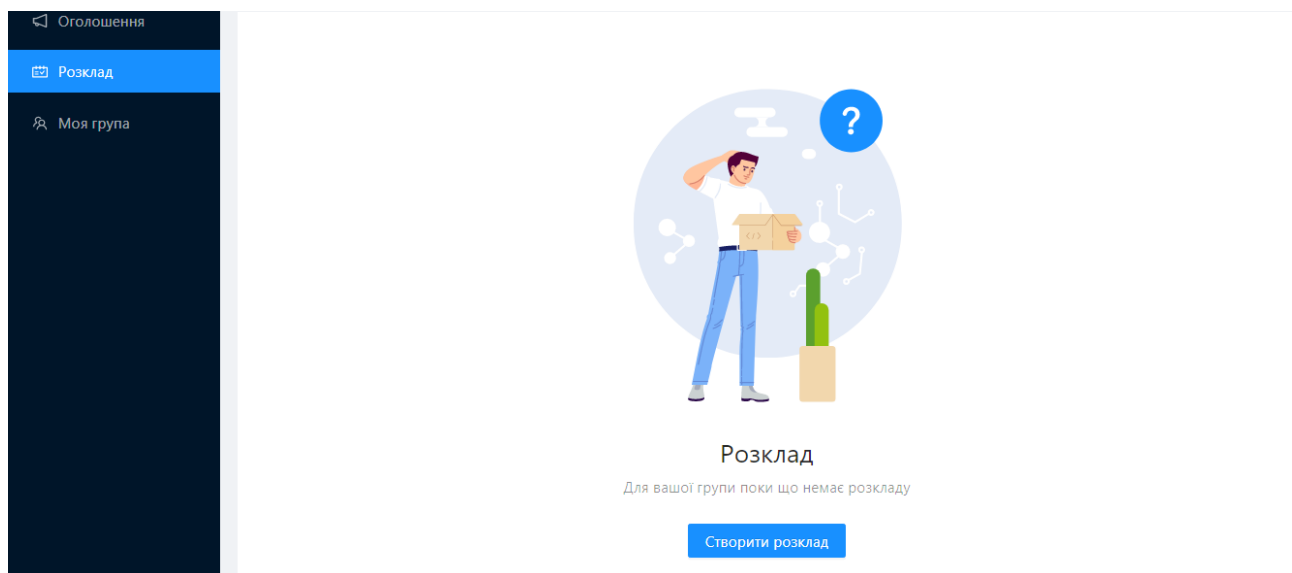


Рисунок 5.9 – Сторінка розкладу, без фактично існуючого розкладу занять

Натискаємо на кнопку «Створити розклад», перед нами з'являється модальне вікно створення розкладу. У ньому можна знайти невеликий блок з підказками, та кнопку «Додати пару», що додає новий рядок розкладу у цей

день. Внизу наявні дві кнопки – «Відміна» та «Наступний день». Відміна припиняє процес створення розкладу без збереження введених даних. Якщо натиснути на кнопку «Наступний день», буде відкрито створення розкладу на наступному дні. Спочатку йде 7 днів тижня по чисельнику, а потім 7 днів по знаменнику відповідно. Модальне вікно зображено на рис. 5.10.

(Чисельник) Понеділок

- Якщо пара випадає на вікно, залишайте обидва поля у рядку порожніми
 - Якщо у цей день немає пар, пропустіть натиснувши "Наступний день"

1	ТВШНМ лк	Субботін С.О.	⊖
2	Предмет	Викладач	⊖
3	АОП лб	Качан О.І.	⊖

+ Додати пару

Відмінити Наступний день

Рисунок 5.10 – Модальне вікно створення розкладу

Тепер ми можемо ознайомитись та відредагувати щойно створений розклад. На сторінці з розкладом існує 2 вкладки, що переключають розклад на «по-чисельнику» та «по-знаменнику». Поточний день тижня підсвічується голубим кольором, а вихідні дні (субота та неділя) мають червоні грані.

Успішно створений розклад занять зображений на рис. 5.11.

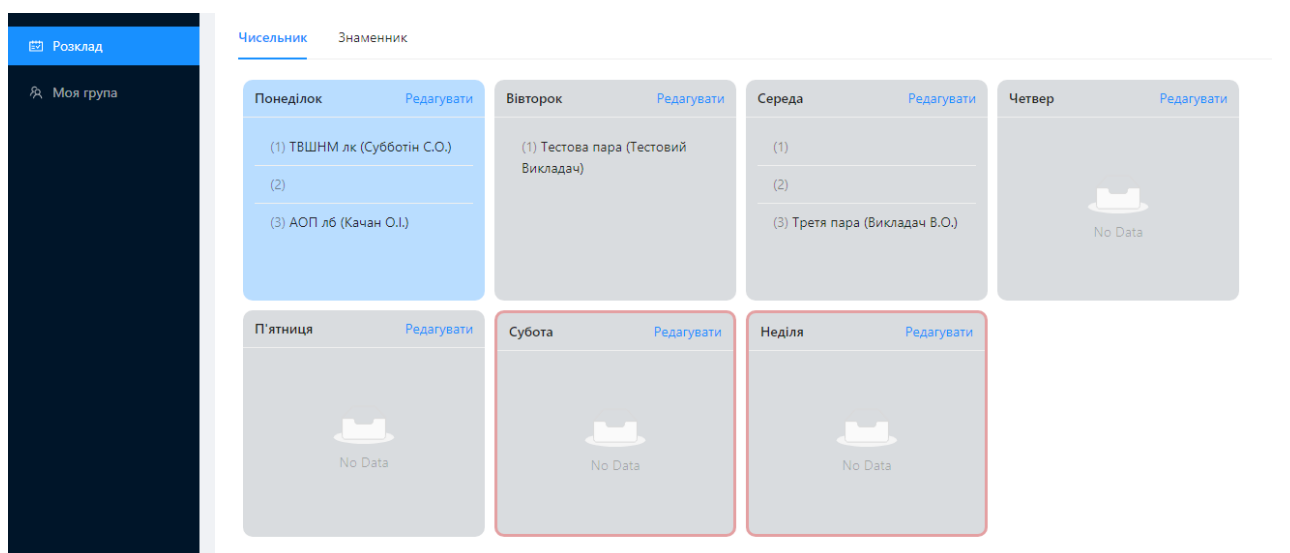


Рисунок 5.11 – Успішно створений розклад

5.5 Сторінка «Моя група»

На завершення розглянемо більш інформативну, ніж функціональну сторінку «Моя група». У ній зображено таблицю зі усіма студентами та куратором, що включені до поточної групи користувача. У таблиці наведені ім'я та прізвища користувачів, їх поштові адреси для особистого листування та їх ролі у системі веб-застосунку «Student Organizer». Сторінку «Моя група» зображено на рис. 5.12.

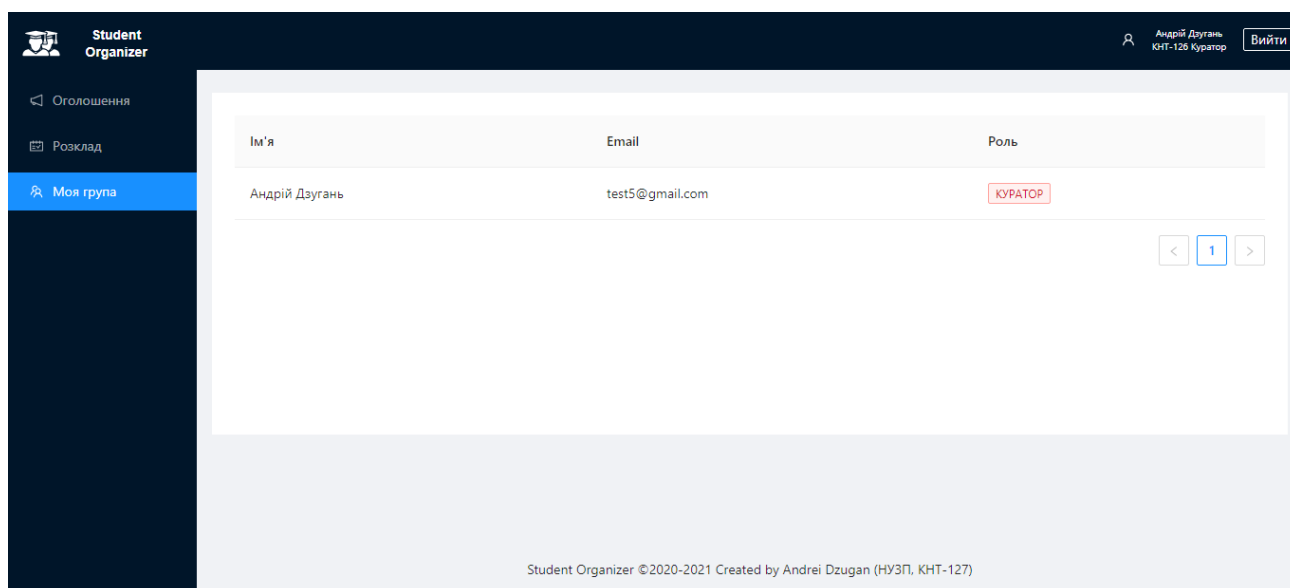


Рисунок 5.12 – Сторінка «Моя група»

5.6 Висновки за розділом 5

У розділі 5 було розглянуто реалізований веб-застосунок, його функціональність для усіх груп користувачів. Було наведено скріншоти працюючої програми, щоб довести працездатність та її ефективність. Проведено дослідження клієнтського графічного інтерфейсу та розписано інструкцію користувача.

ВИСНОВКИ

У ході виконання даної роботи було розроблено онлайн органайзер для студентів та викладачів. Створений веб-застосунок «Student Organizer» дозволяє отримувати та керувати інформацією щодо графіку навчального процесу.

Онлайн-органайзер розроблено згідно з сучасними вимогами та тенденціями. Вимоги до веб-застосунку, який розроблювався нами, було сформовано в ході аналізу сучасних аналогів та на основі порівняння їх переваг та недоліків.

Під час роботи над проектом проаналізовано та застосовано основні досягнення у галузі Веб-технологій. В ході аналізу доступних технологій реалізації веб-застосунку, було обрано створення клієнт-серверної архітектури з використанням мови програмування Javascript, зокрема фреймворка Express та бібліотеки React JS. В ролі СКБД обрано MongoDB. Було виконано проектування веб-застосунку та написання коду.

Веб-застосунок відповідає всім вимогам, які були поставлені на етапі розробки технічного завдання, та містить необхідний мінімум функціоналу. Зокрема, забезпечено повний функціонал для актуалізації інформації щодо навчального процесу; реалізована відповідна авторизація та аутентифікація; забезпечено надійну та безпечну роботу продукту.

Веб-застосунок «Student Organizer» розроблений у доступній безкоштовній версії. Основною перевагою цього веб-застосунку в порівнянні з аналогами є відсутність довгих і трудомістких налаштувань перед початком роботи та надлишкового функціоналу.

За допомогою сучасних технологій було створено зручний користувацький інтерфейс. При проектування веб-застосунку також було враховано адаптивність до будь-якого пристрою, на якому буде працювати користувач, що підвищує його мобільність.

Було проведено мануальне тестування веб-застосунку на локальному сервері Node.js.

Онлайн-органайзер може бути використаний у навчальних закладах, необхідно лише ввести процес оновлення даних з розкладу занять того чи іншого вузу. Застосування розробленого веб-застосунку для актуалізації інформації щодо навчального процесу дозволить створити єдине інформаційне середовище для студентів та викладачів.

Веб-застосунок спроектовано із урахуванням потенційного розширення функціоналу у майбутньому.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Косякова, А. В. Використання Personal Information Manager для оптимізації навчального процесу [Текст] / А. В. Косякова, О. С. Щербіна // Вісник студентського наукового товариства Донецького національного університету імені Василя Стуса, – 2017. – № 9. – Том 1. – С. 93–96.
2. Тайм-менеджмент как основа успешного обучения в учреждении высшего образования [Текст] / [сост. : Н. Л. Кучинская]. – Минск : 2015. – 37 с.
3. Personal Information Manager (PIM) [Electronic resource]. – Access mode : <https://www.techopedia.com/definition/24752/personal-information-manager-pim>
4. Вікіпедія [Електрон. ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Google_Calendar
5. Google Календарь [Электрон. ресурс]. – Режим доступа : <https://www.google.com/calendar/about/>
6. Кільченко, А. В. Використання сервісу Google Календар для планування та організації науково-дослідної роботи в науковій установі / А. В. Кільченко, Д. М. Климчук // Звітна науково-практична конференція Інституту інформаційних технологій і засобів навчання НАПН України, Київ, 27 березня 2018 р. : збірник матеріалів наукової конференції. – Київ : ІТЗН НАПН України, 2018. – С. 18–23
7. Вікіпедія [Електрон. ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Microsoft_Outlook
8. Microsoft Outlook [Электрон. ресурс]. – Режим доступа : <https://www.microsoft.com/ru-ru/microsoft-365/outlook/email-and-calendar-software-microsoft-outlook>
9. Astonsoft [Электрон. ресурс]. – Режим доступа : <https://www.astonsoft.ee/ru/>
10. EssentialPIM [Электрон. ресурс]. – Режим доступа : <https://www.essentialpim.com/ru/>

11. JavaScript [Электрон. ресурс]. – Режим доступа : <http://xn--80adth0aefm3i.xn--j1amh/javascript>
12. Вікіпедія [Электрон. ресурс]. – Режим доступа : https://uk.wikipedia.org/wiki/Брендан_Айк
13. Флэнаган, Д. JavaScript. Подробное руководство [Текст] / Д. Флэнаган. – СПб: Символ-Плюс, 2008. – 992 с.
14. Node.js [Электрон. ресурс]. – Режим доступа : <https://nodejs.org/uk/about/>
15. Express [Электрон. ресурс]. – Режим доступа : <https://expressjs.com/ru/>
16. MongoDB [Электрон. ресурс]. – Режим доступа : <https://www.mongodb.com/what-is-mongodb>
17. Введение в JSON [Электрон. ресурс]. – Режим доступа : <https://www.json.org/json-ru.html>
18. Камер, Дуглас Э. Сети TCP/IP [Текст] / Дуглас Э. Камер, Дэвид Л. Стивене. – М. : Издательский дом "Вильямс". – Том. 3 Разработка приложений типа клиент/сервер для Linux/POSIX 2002. – 592 с.
19. Понятие клиент-серверных систем [Электрон. ресурс]. – Режим доступа : <http://bourabai.kz/dbt/client1.htm>
20. Bajaj P. Overview of Single Page Application (SPA) [Electronic resource] / P. Bajaj – Access mode : <https://www.c-sharpcorner.com/blogs/overview-of-single-page-application-spa1>
21. React [Electronic resource]. – Access mode : <https://reactjs.org/>
22. Krill, P. React: Making faster, smoother UIs for data-driven Web apps [Electronic resource] P. Krill / – Access mode : <https://www.infoworld.com/article/2608181/react--making-faster--smoother-uis-for-data-driven-web-apps.html>
23. Ant Design [Electronic resource]. – Access mode: <https://ant.design/>
24. Fielding, Roy, T. Architectural Styles and the Design of Network-based Software Architectures: diss ... doctor of philosophy / Roy T. Fielding. – Irvine:

University of California, 2000. – 162 p.

25. REST API: что это такое простыми словами, примеры запросов, варианты использования сервиса, методы [Электрон. ресурс] – Режим доступа : <https://www.cleverence.ru/articles/elektronnaya-kommertsiya/rest-api-chto-eto-takoe-prostymi-slovami-primery-zaprosov-varianty-ispolzovaniya-servisa-metody/>

26. Хватит использовать REST для API! [Электрон. ресурс] – Режим доступа : <https://proglib.io/p/hvatit-ispolzovat-rest-dlya-api-2020-12-12>

27. Что такое реляционная база данных? [Электрон. ресурс] – Режим доступа : <https://www.oracle.com/ru/database/what-is-a-relational-database/>

28. Нереляционные данные и базы данных NoSQL [Электрон. ресурс] – Режим доступа : <https://docs.microsoft.com/ru-ru/azure/architecture/data-guide/big-data/non-relational-data>

29. MongoDB - нереляційна база даних [Електрон. ресурс]. – Режим доступу : <http://energyfirefox.blogspot.com/2012/07/mongodb.html>

30. Ткаченко, В. Технологии создания веб-приложений на основе JS в среде Node.js для взаимодействия с MongoDB (Mongoose) [Электрон. ресурс] / В. Ткаченко. – Режим доступа : <https://www.lessons-tva.info/articles/informat/44.html>

31. Безопасность сайтов и веб-приложений [Электрон. ресурс] – Режим доступа : https://web-creator.ru/articles/about_web_sites_security

32. Идентификация, аутентификация и авторизация – в чем разница? [Электрон. ресурс] – Режим доступа : <https://www.kaspersky.ru/blog/identification-authentication-authorization-difference/29123/>

33. Что такое Аутентификация: Методы и Элементы [Электрон. ресурс] – Режим доступа : <https://sendpulse.com/ru/support/glossary/authentication#>

34. Пять простых шагов для понимания JSON Web Tokens (JWT) [Электрон. ресурс] – Режим доступа : <https://habr.com/ru/post/340146/>

ДОДАТОК А
Технічне завдання

Вступ

Даний дипломний проєкт поширюється на розробку та тестування веб-застосунку «Student Organizer». Область застосування: заклади освіти.

А.1 Підстави для розробки

Підставою для розробки є завдання на виконання дипломної кваліфікаційної роботи кваліфікаційно-освітнього рівня бакалавр, на тему «Програмна реалізація онлайн органайзеру для студентів» затверджене наказом Національного університету «Запорізька політехніка» № 103 від 30 березня 2021 р.

А.2 Призначення розробки

Розробка призначена для використання при організації навчального процесу у вищих навчальних закладах.

А.3 Вимоги до програмного виробу

Система має забезпечувати такі основні функції:

- можливість перегляду, створення та оновлення повідомлень на головній сторінці. Повідомлення повинні відображатися тільки учасникам конкретної групи;
- перегляд, створення та редагування розкладу занять по чисельнику та знаменнику;
- можливість перегляду учасників своєї групи;
- можливість створення нової групи, за умови що її не було обрано при реєстрації;
- зручний та зрозумілий для використання інтерфейс.

A.4 Вимоги до програмної документації

У процесі виконання дипломного проєкту (роботи) повинна бути розроблена документація у складі технічного завдання, пояснювальної записки, програми та методики тестування.

A.5 Техніко-економічні показники

Для забезпечення роботи веб-застосунку необхідно мати будь-який електронний носій, який під'єднано до мережі Інтернет та браузер з мінімальними вимогами.

Особливих вимог до експлуатації застосунку не представляється.

A.6 Порядок контролю та приймання

Дипломна кваліфікаційна робота допускається до захисту керівником, нормоконтролером, затверджується завідувачем кафедри та захищається в присутності членів екзаменаційної комісії.

ДОДАТОК Б
Текст програми

Б.1 Функція контроллера створення нового користувача

```
exports.signup = (req, res) => {

  const user = new User({
    name: req.body.name,
    surname: req.body.surname,
    email: req.body.email,
    faculty: req.body.faculty,
    group: req.body.group,
    authorities: [],
    roles: [],
    password: bcrypt.hashSync(req.body.password, 8)
  });

  if (req.body.roles) {
    Role.find(
      { name: { $in: req.body.roles } },
      (err, roles) => {
        if (err) {
          console.error(err);
          res.status(500).send({ message: err });
          return;
        }
        user.roles = roles.map(role => role._id);
        user.authorities = roles.map(role => role.name);
        registerUser(user, res);
      }
    );
  } else {
    Role.findOne({ name: 'student' }, (err, role) => {
      if (err) {
        console.error(err);
        res.status(500).send({ message: err });
        return;
      }
      user.roles = [role._id];
      user.authorities = [role.name];
      registerUser(user, res);
    });
  }
};
```

Б.2 Функція контроллера аутентифікації існуючого користувача

```
exports.signin = (req, res) => {
  User.findOne({
    email: req.body.email
  })
  .populate('faculty group roles')
  .exec((err, user) => {
    if (err) {
      res.status(500).send({ message: err });
      return;
    }
  })
};
```

```

    if (!user) {
      return res.status(404).send({ message: 'Невірний email або пароль!' });
    }

    const isPasswordValid = bcrypt.compareSync(
      req.body.password,
      user.password
    );

    if (!isPasswordValid) {
      return res.status(401).send({ message: 'Невірний email або пароль!' });
    }

    const token = jwt.sign({ id: user.id }, config.secret, {
      expiresIn: 86400 // 24 hours
    });

    const authorities = user.roles.map(role => `ROLE_${role.name.toUpperCase()}`);

    res.status(200).send({
      message: 'User was successful logged in!',
      user: {
        id: user._id,
        name: user.name,
        surname: user.surname,
        faculty: user.faculty,
        group: user.group,
        email: user.email,
        roles: authorities,
        accessToken: token
      }
    });
  });
};

```

Б.3 Функція контролера оновлення розкладу групи

```

exports.createTimetable = (req, res) => {
  User.findById(req.userId).exec((err, user) => {
    if (err) {
      console.error(err);
      res.status(500).send({ message: err });
      return;
    }

    const timetable = new Timetable({
      group: user.group,
      schedule: req.body.schedule
    });

    timetable.save((err, createdTimetable) => {
      if (err) {
        console.error(err);
        res.status(500).send({ message: err });
        return;
      }
    });
  });
};

```

```

        res.status(201).send(createdTimetable);
    });
});
};

exports.getTimetable = (req, res) => {
    Timetable.findOne({ group: req.params.id })
        .exec((err, timetable) => {
            if (!timetable) {
                return res.status(404).send({ data: { message: 'Timetable is not found' } });
            }
            res.status(200).send({ timetable });
        });
};

exports.updateTimetable = (req, res) => {
    Timetable.findOne({ _id: req.params.id })
        .exec((err, timetable) => {
            if (!timetable) {
                return res.status(404).send({ data: { message: 'Timetable is not found' } });
            }
            console.log(timetable);
            const newSchedule = req.body;
            const weekType = req.body.weekType;
            const dayIndex = timetable.schedule[weekType].findIndex(day => day.dayOfWeek ===
newSchedule.dayOfWeek);
            delete newSchedule.weekType;
            timetable.schedule[weekType][dayIndex] = newSchedule;
            timetable.save((err, updatedTimetable) => {
                if (err) {
                    return res.status(400).send({ data: { message: err } });
                }
                res.status(200).send({ timetable: updatedTimetable });
            });
        });
};
};

```

Б.4 Компонент сторінки з розкладом групи

```

const TimeTable = () => {

    const [timetable, setTimetable] = useState(null);
    const [reloadTrigger, reload] = useState(false);
    const { userInfo } = useContext(AuthContext);

    const fetchTimetable = async () => {
        try {
            const { data: { timetable: fetchedTimetable } } = await
TimetableService.getTimetable(userInfo?.group?._id);
            console.log(fetchedTimetable);
            setTimetable(fetchedTimetable || null);
        } catch(e) {
            console.dir(e);
        }
    };
};

```

```

useEffect(() => {
  fetchTimetable();
  //eslint-disable-next-line
}, [reloadTrigger]);

const [typeOfWeek, setTypeOfWeek] = useState('numerator');

return <AppLayout>
  { timetable ?
    <Tabs
      activeKey={ typeOfWeek }
      onClick={ (type) => setTypeOfWeek(type) }
      animated={ { inkBar: true, tabPane: true } }
    >
      <TabPane tab="Чисельник" key="numerator">
        <Schedule
          reload={ () => reload(v => !v) }
          weekType="numerator"
          timetableId={ timetable._id }
          timetable={ timetable.schedule.numerator }
        />
      </TabPane>
      <TabPane tab="Знаменник" key="denominator">
        <Schedule
          reload={ () => reload(v => !v) }
          weekType="denominator"
          timetableId={ timetable._id }
          timetable={ timetable.schedule.denominator }
        />
      </TabPane>
    </Tabs>
    : <NoSchedule fetchTimetable={ fetchTimetable } /> }
  </AppLayout>;
};

export default TimeTable;

```

Б.5 Компонент модального вікна створення нового розкладу

```

const CreateSchedule = ({ isOpened, closeModal, fetchTimetable }) => {

  const [dayOfWeek, setDayOfWeek] = useState(0);
  const [typeOfWeek, setTypeOfWeek] = useState('numerator');

  const [form] = Form.useForm();

  const changeScheduleValues = (type, values) => {
    const { schedule = [] } = values;
    const parsedSchedule = schedule.map(lesson => lesson || { subject: "", teacher: "" });
    const scheduleWithDay = {
      dayOfWeek,
      lessons: parsedSchedule
    };
    scheduleValues[type] = [...scheduleValues[type], scheduleWithDay];
  };
};

```



```

        </Form.Item>
        <Form.Item
          { ...restField }
          name={ [name, 'teacher'] }
          fieldKey={ [fieldKey, 'teacher'] }
        >
          <Input placeholder="Викладач"/>
        </Form.Item>
        <MinusCircleOutlined onClick={ () => remove(name) }/>
      </Space>
    )) }
    <Form.Item>
      <Button type="dashed" onClick={ () => add() } block icon={ <PlusOutlined/>
        <div>
          Додати пару
        </div>
      </Button>
    </Form.Item>
  </>
) }
</Form.List>
</Form>
</Modal>
);
};

export default CreateSchedule;

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота бакалавра

Програмна реалізація онлайн органайзеру для студентів

Виконав
ст. гр. КНТ-127

А.О. Дзугань

Керівник
к.т.н., доцент

Т.О. Колпакова

2021

Рисунок В.1 – Слайд № 1

МЕТА ТА ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

Мета роботи – розробка онлайн-органайзеру для надання користувачам зручного та надійного засобу для отримання та керування інформацією щодо графіку навчального процесу.

Завдання роботи:

- проаналізувати предметну область та аналогічні органайзери;
- проаналізувати методи розробки веб-застосунків;
- обрати найзручніші та найактуальніші технології для їх використання при розробці онлайн органайзеру;
- спроектувати веб-застосунок для студентів та викладачів враховуючи всі особливості веб-технологій;
- провести тестування розробленого веб-застосунку.

2

Рисунок В.2 – Слайд № 2

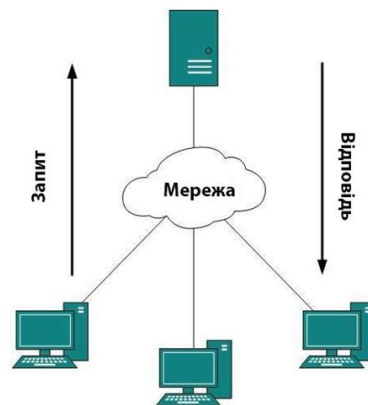
АНАЛІЗ АНАЛОГІВ ВЕБ-ЗАСТОСУНКУ

Характеристики	Google Calendar	Microsoft Outlook	EssentialPIM
Веб-застосунок	+	+	+
Мобільна версія	+	+	+
Десктопная версія	+	-	+
Необхідність наявності акаунту	+	-	-
Календар	+	+	+
Сповіщення	+	+	+
Синхронізація	+	-	+
Імпорт та експорт подій	+	-	+
Спільний доступ до подій	+	+	+
Хмарне сховище	+	-	+
Створення додаткового календаря за необхідності	+	-	-
Безпека	+	+	+
Зручний графічний інтерфейс	+	-	+

3

Рисунок В.3 – Слайд № 3

ЗАГАЛЬНА АРХІТЕКТУРА ВЕБ-ЗАСТОСУНКУ

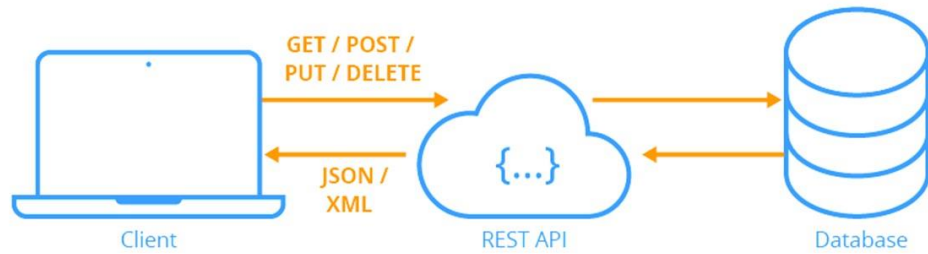


Клієнти роблять запити до сервера, який оброблює, зберігає та повертає дані, які необхідні користувачеві, тому його можна вважати головною частиною клієнт-серверної архітектури.

4

Рисунок В.4 – Слайд № 4

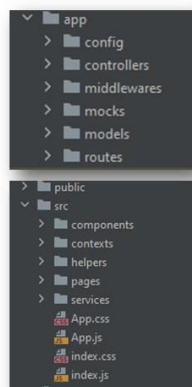
СХЕМА REST АРХІТЕКТУРИ



5

Рисунок В.5 – Слайд № 5

СТРУКТУРНА СХЕМА ТА МОДЕЛІ ЗАСТОСУНКУ



```

const User = mongoose.model(
  'User',
  new mongoose.Schema({
    name: String,
    surname: String,
    faculty: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Faculty'
    },
    group: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Group'
    },
    email: String,
    password: String,
    roles: [
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Role'
    ]
  })
);

const Faculty = mongoose.model(
  'Faculty',
  new mongoose.Schema({
    fullName: String,
    abbreviation: String,
    groups: [
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Group'
    ]
  })
);

const Advertisement = mongoose.model(
  'Advertisement',
  new mongoose.Schema({
    title: String,
    description: String,
    user: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'User'
    },
    group: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Group'
    }
  }, { timestamps: true })
);

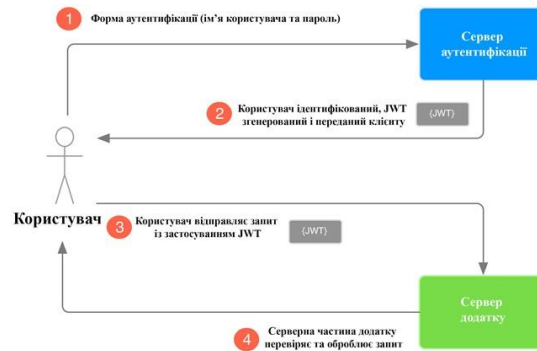
const Group = mongoose.model(
  'Group',
  new mongoose.Schema({
    name: String,
    faculty: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Faculty'
    },
    members: [
      type: mongoose.Schema.Types.ObjectId,
      ref: 'User'
    ]
  })
);

const Timetable = mongoose.model(
  'Timetable',
  new mongoose.Schema({
    group: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Group'
    },
    schedule: {
      numerator: [
        {
          dayOfWeek: Number,
          lessons: [
            {
              subject: String,
              teachers: String
            }
          ]
        }
      ],
      denominator: [
        {
          dayOfWeek: Number,
          lessons: [
            {
              subject: String,
              teachers: String
            }
          ]
        }
      ]
    }
  })
);
  
```

6

Рисунок В.6 – Слайд № 6

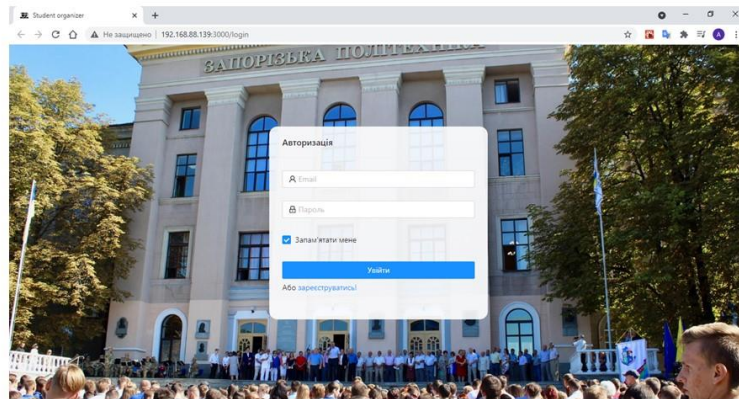
БЕЗПЕКА ВЕБ-ЗАСТОСУНКУ



7

Рисунок В.7 – Слайд № 7

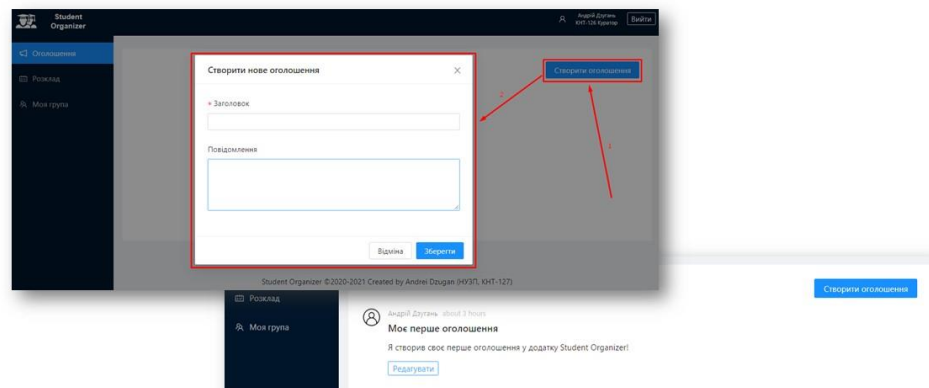
ВЕБ-ЗАСТОСУНОК «STUDENT ORGANIZER». АВТОРИЗАЦІЯ



8

Рисунок В.8 – Слайд № 8

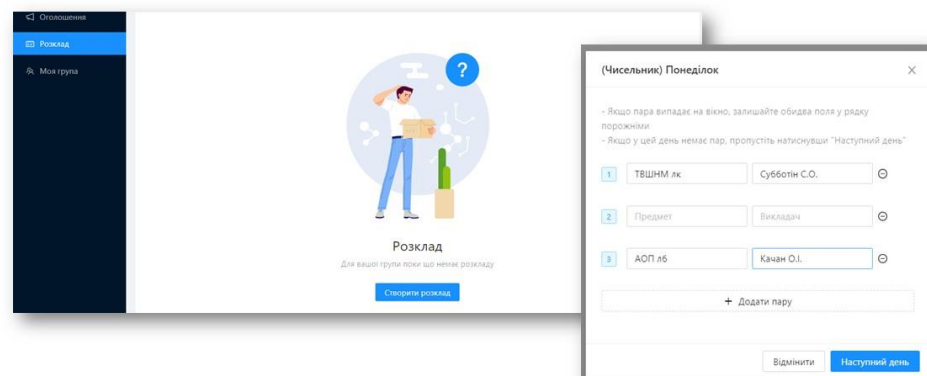
СТВОРЕННЯ НОВИХ ОГолоШЕНЬ



9

Рисунок В.9 – Слайд № 9

СТВОРЕННЯ РОЗКЛАДУ



10

Рисунок В.10 – Слайд № 10

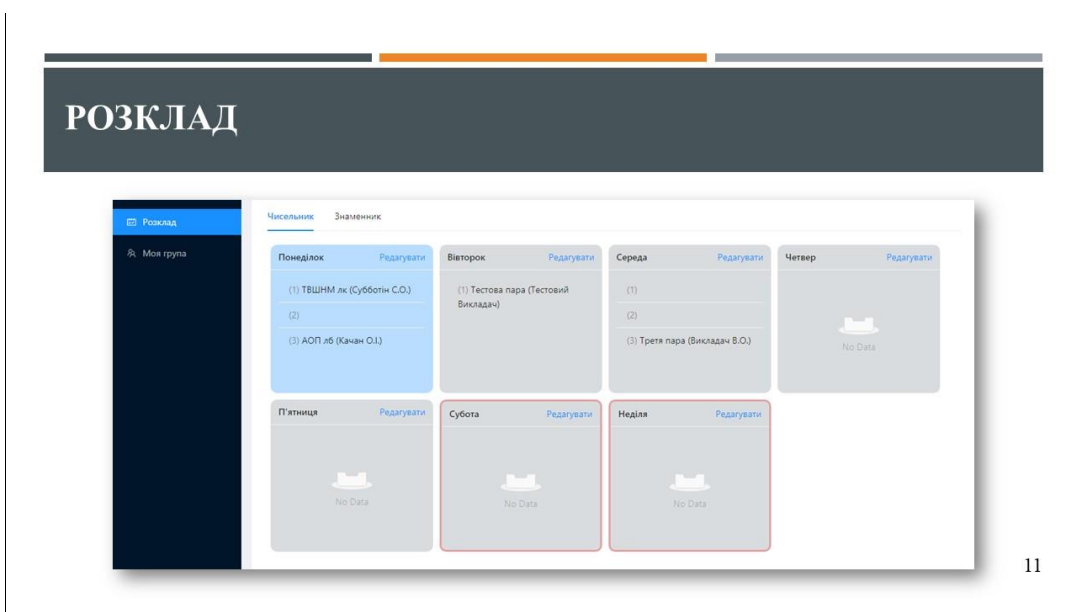


Рисунок В.11 – Слайд № 11



Рисунок В.12 – Слайд № 12



Дякую за увагу!

Рисунок В.13 – Слайд № 13