

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ЧАСУ
ТА ПЛАНУВАННЯ ПОДІЙ

DEVELOPMENT OF WEB APPLICATION FOR TIME MANAGEMENT
AND EVENT PLANNING

Виконав(ла): студент(ка) 4 курсу, групи КНТ-131

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

БОЛУЧЕВСЬКИЙ М.Б.

(ПРИЗВИЩЕ та ініціали)

Керівник ФЕДОРОНЧАК Т.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ТИМЕНКО К.І.

(ПРИЗВИЩЕ та ініціали)

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

Факультет КНТ
 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.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БОЛУЧЕВСЬКОГО Максима Борисовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебзастосунку для організації часу та планування подій. Development of a Web Application for Time Management and Event Planning.
- керівник проєкту (роботи) к.т.н., доцент ФЕДОРОНЧАК Тетяна Василівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)
- затверджені наказом закладу вищої освіти від “28” квітня 2025 року № 209
2. Строк подання студентом проєкту (роботи) 02 червня 2025 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ФЕДОРОНЧАК Т.В., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

(підпис) Максим БОЛУЧЕВСЬКИЙ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна ФЕДОРОНЧАК
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 90с., 11 табл., 43 рис., 3 дод., 34 джерела.

ПЛАНУВАННЯ, ВЕБЗАСТОСУНОК, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КОРИСТУВАЧ, МІКРОСЕРВІС, КОНТЕЙНЕР, JAVA WEB API, SPRING BOOT, SPRING AUTHORIZATION SERVER.

Об'єкт дослідження – процес розробки програмного забезпечення для організації часу і планування подій.

Предмет дослідження – вебзастосунки для організації часу та планування подій.

Мета роботи – розробка вебзастосунку для особистого планування подій.

Матеріали, методи і технічні засоби: мови програмування Java і Typescript, фреймворки Spring Boot і Angular, IDE IntelliJ IDEA, система управління контейнерами Docker.

Результати. Створено програмний вебзастосунок на мові Java, з клієнтською частиною на Typescript. Програма має зрозумілі функції і зручний інтерфейс, який дозволяє користувачам робити планування.

Висновки. Розроблений програмний вебзастосунок для планування подій, дає змогу створювати плановані події, додавати інформацію про них, встановлювати нагадування на запланований час, і залучати контакти.

Галузь використання – особисте планування, організація власного часу і контактів.

ABSTRACT

Explanatory note to the bachelor's diploma qualification work: 90p., 11 tables, 43 figures, 3 appendixes, 34 sources.

PLANNING, WEB APPLICATION, SOFTWARE, USER, MICROSERVICE, CONTAINER, JAVA WEB API, SPRING BOOT, SPRING AUTHORIZATION SERVER.

The object of research is the process of developing software for time management and event planning.

The subject of the research is web applications for time management and event planning.

The purpose of the work is to develop a web application for personal event planning.

Materials, methods and technical means: Java and Typescript programming languages, Spring Boot and Angular frameworks, IntelliJ IDEA IDE, Docker container management system.

Results. A software web application in Java was created, with a client part in Typescript. The program has clear functions and a convenient interface that allows users to make plans.

Conclusions. A web application for event planning has been developed, allowing you to create scheduled events, add information about them, set reminders for the scheduled time, and engage contacts.

The field of use – personal planning, organization of your own time and contacts.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	8
Вступ.....	9
1 Аналіз предметної області	10
1.1 Актуальність використання планувальників	10
1.2 Огляд аналогів	11
1.3 Постановка завдання роботи	19
1.4 Висновки з розділу 1	19
2 Вибір та обґрунтування матеріалів та методів	21
2.1 Вибір мови програмування	21
2.2 Вибір фреймворку для створення ПЗ	25
2.3 Вибір середовища розробки	27
2.4 Вибір системи управління базами даних	30
2.5 Висновки з розділу 2	31
3 Опис архітектури та розробки програми	32
3.1 Проектування архітектури програмного забезпечення.....	32
3.2 Структура розроблюваного вебзастосунку для організації часу та	
планування подій	34
3.3 Огляд функціонування програмного забезпечення.....	37
3.4 Огляд структури бази даних	44
3.5 Основні рішення щодо проектування інтерфейсу програмного.....	
забезпечення	46
3.6 Висновки до розділу 3.....	53
4 Експлуатація, тестування та експериментальне дослідження програми	54
4.1 Призначення й умови застосування програми	54
4.2 Розгортання проекту	55
4.3 Вхідні та вихідні дані	56
4.4 Методика та результати тестування.....	57

4.5 Висновки до розділу 4.....	62
Висновки.....	63
Перелік джерел посилання	65
Додаток А Технічне завдання.....	68
Додаток Б Текст програми.....	72
Додаток В Слайди презентації	80

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ACID – Atomicity, Consistency, Isolation, Durability;

API – Application Programming Interface;

CLI – Command line interface;

CRUD – Create, Read, Update, Delete;

GPL – General Public License;

JVM – Java Virtual Machine;

JWT – JSON Web Token;

IDE – Integrated Development Environment;

IT – Information Technology;

SPA – Single Page Application;

URL – Uniform Resource Locator;

UX – User experience;

БД – база даних;

ПЗ – програмне забезпечення;

СКБД – система керування базами даних.

ВСТУП

У сучасному світі питання ефективного управління власним часом набуває дедалі більшої актуальності. В умовах постійного зростання обсягів інформації, навантаження та темпу життя, людина стикається з необхідністю раціонального планування повсякденних справ, дотримання дедлайнів та збереження балансу між особистим і професійним життям.

Варто розуміти, що ефективний тайм-менеджмент допомагає оптимізувати час, а не поміщати себе в рамки, в яких неможливо жити [1].

Ці питання та виклики спонукають до активного пошуку інструментів для підвищення особистої продуктивності.

Особливої популярності набувають електронні планнери, оскільки традиційні засоби організації часу: щоденники, паперові нотатники, таблиці – не завжди відповідають вимогам гнучкості, інтеграції та мобільності, які виникають в сучасному інформаційному середовищі. А їх модифіковані цифрові аналоги дозволяють зручно керувати розкладом, зберігати контакти, створювати події та зустрічі, встановлювати нагадування, а також синхронізувати дані між різними пристроями.

Метою даної кваліфікаційної роботи є розробка вебзастосунку для планування та керування власним часом, який надає користувачу інтуїтивно зрозумілий інтерфейс, систему подій і нагадувань, а також можливість керування контактами. Розроблене програмне забезпечення має підтримувати базові функції календаря, обробку персональних даних та функціонал, що сприятиме підвищенню особистої ефективності користувача.

Актуальність теми полягає не лише в її прикладному значенні, а й у широких можливостях для подальшого розвитку системи: впровадження штучного інтелекту для аналізу активності користувача, інтеграція з іншими сервісами, що може суттєво розширити функціональність та покращити користувацький досвід.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність використання планувальників

У сучасному світі, де інформаційне навантаження та темп життя постійно зростають, ефективне управління власним часом стає критично важливим. Це дуже актуально у студентів, фахівців сфери ІТ, підприємців та всіх, хто прагне високої продуктивності та балансу між особистим і професійним життям. Програмне забезпечення для планування та керування часом виступає незамінним інструментом в такій ситуації [2].

Використання цифрових планувальників може суттєво збільшити ефективність використання як особистого так і робочого часу, що сприятиме зниженню стресу та підвищенню продуктивності. Для прикладу, навіть такі прості інструменти як Google Tasks, TickTick дозволяють організовувати задачі, встановлювати нагадування. Вони доступні будь-де і будь-коли, що допомагає завжди бути в курсі своїх планів та не забувати виконувати справи.

1.1.1 Основні тенденції у розробці планувальників

Підтримка персональних уподобань є досить важливою в застосунках для управління часом. Користувачі повинні мати можливість налаштовувати нагадування та сповіщення для запланованих справ, зустрічей або важливих етапів. Це підвищує гарантію, що нічого не буде пропущено [3].

Наразі все більше роблять акцент на мінімалістичний дизайн та зручний користувацький досвід (який часто позначають як UX). Перевантаженість інтерфейсу великою кількістю елементів, тексту або анімацій може відволікати та викликати незадоволення. Розробники прагнуть створити інтуїтивно зрозумілі, чисті та приємні у використанні застосунки, що не потребують багато часу на освоєння. Ергономічність та візуальна привабливість все частіше стають важливими конкурентними перевагами.

1.2 Огляд аналогів

Перед створенням власного проекту для планування подій важливо проаналізувати існуючі рішення, що вже користуються попитом. Варто розглянути найпоширеніші функціональні можливості, підходи до інтерфейсу, технічні реалізації, а також виявити сильні та слабкі сторони продуктів.

Важливо порівняти наявність і методи використання тих чи інших функцій, та які пропозиції надають альтернативні застосунки. Наприклад встановлення нагадування для бажаної задачі, або відправлення запрошення чи сповіщення користувачу на пошту, якщо його залучено до планованих активностей тощо. Якщо певний базовий корисний функціонал не реалізовано, або реалізовано складним чином, чи взагалі винесено до окремих можливостей застосунку, що доступний лише після внесення грошової оплати чи підписки (звісно якщо ми не говоримо про глибокі корпоративні можливості та складні інтеграції, де такий підхід можна вважати доцільним), це може бути сприйнято як недолік та знизить зацікавленість користувача.

1.2.1 Планувальник Google Tasks

Google Tasks – це простий застосунок для ведення задач, розроблений компанією Google. Він інтегрований в Google Workplace та добре підходить для тих хто вже користується середовищем Google [4].

Справи створені у Google Task синхронізуються між усіма пристроями користувача де використовується google акаунт. Це дозволяє бачити свої завдання з будь-якого пристрою та місця [5].

До планів можна додавати деталі та створювати певні підзадачі, що допомагає більш структуровано організувати завдання для себе. Окрім цього, інтегрованість з Google Workplace дозволяє створювати завдання з інших рішень Google [5].

Він також має можливість встановлювати запланований час для події та нагадування, щоб користувач не пропускав та не губив важливі для себе задачі.

А ще задачі можна відмічати як виконані, та сортувати за назвою чи датою. Загалом функціонал Google Task є невеликим, але цього достатньо для звичайного особистого планування повсякденних справ.

Окрім цього варто відзначити що Google Tasks є повністю безкоштовним.

Інтерфейс застосунку є зручним і зрозумілим. Елементів небагато, немає нічого зайвого. Більшу частину вікна займає список задач створених користувачем. В цілому простий ненав'язливий дизайн, з яким розбереться навіть той хто ніколи раніше не працював з подібними додатками.

На рис. 1.1 зображено інтерфейс планувальника Google Tasks [6].

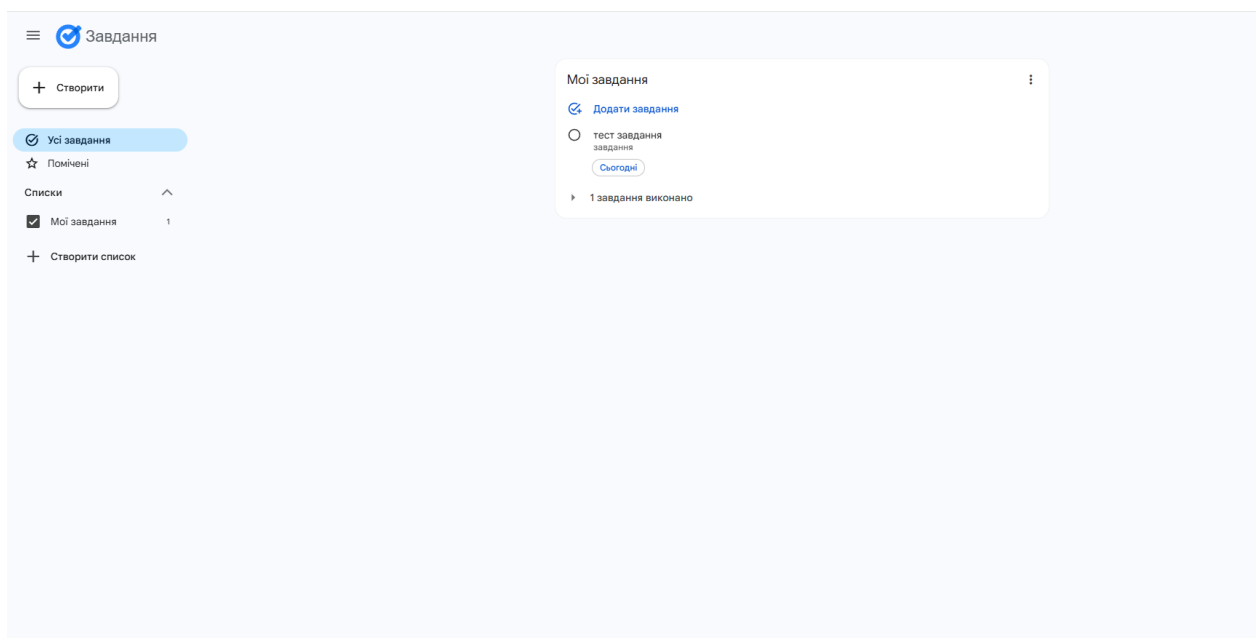


Рисунок 1.1 – Інтерфейс Google Tasks [6]

1.2.2 Нотатник Google Keep

Google Keep – це нотатник, розроблений компанією Google. Він включений до пакету вебредакторів документів Google.

На відміну від Google Tasks цей застосунок має ширше коло можливостей і функцій. Окрім створення нотатки та встановлення нагадування на певний час, тут можна змінити колір її фону, додати співавтора, а також додати зображення, але підтримуються лише файли GIF, JPEG, JPG та PNG [7].

Тут, як і в попередньому випадку усі нотатки та нагадування синхронізуються між пристроями.

Застосунок також має цікаву функцію, що дозволяє розпізнати текст з фотографії, щоб додати його у нотатку. Але це працює лише з якісними зображеннями, і не підходить для сканування складних і великих документів [7].

Інтерфейс застосунку схожий на Google Tasks, але з урахуванням елементів для використання вже перерахованих функціональних можливостей. Нотатки розташовуються вільно в умовних колонках, їх також можна закріплювати, щоб одні знаходилися над іншими.

Загалом Google Keep розроблений для швидкого фіксування назрілих думок для їх подальшого використання. Іноді його порівнюють з холодильником, на який невпорядковано наліплено стікери [7].

Google Keep є повністю безкоштовним.

На рис. 1.2 зображено інтерфейс планувальника Google Keep [8].

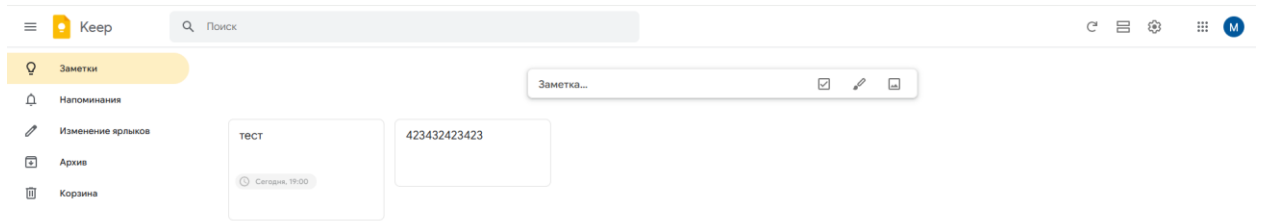


Рисунок 1.2 – Інтерфейс Google Keep [8]

1.2.3 Застосунок для керування завданнями Any.do

Any.do – це платформа для ефективного управління завданнями і проектами, яку заснували Омер Перчик, Йоні Лінденфельд та Ітай Кахана у 2011 році.

Він доступний як у вебверсії та як окремий десктопний застосунок для ОС Windows, macOS, на мобільних пристроях для iOS та Android.

Безкоштовною версією Any.do, може користуватися кожен. Однак вона має обмежені функції, що не в повній мірі розкривають користувацький досвід застосунку. З нею доступні можливості створити завдання, встановити нагадування до завдання, додати до нього коментарі та підзадачі [9].

Після успішної реєстрації користувач зможе ознайомитися з базовими функціями і почати створення завдань. На вкладках Наступні 7 днів, Всі мої завдання є кнопка «Додати завдання». Щоб створити завдання потрібно натиснути на неї та ввести необхідну інформацію. Після створення завдання до нього можна додати нотакти та встановити підзавдання. Також можна прикріпити файли до завдання. Розмір файлів обмежений 1.5 Мб. Є можливість додавати мітки до завдання, але у безкоштовну плані, доступна

лише одна мітка «Пріоритетність». Доступна також можливість встановити нагадування, щоб програма вчасно сповіщала про необхідність виконання завдання [9].

Можна групувати кожне завдання за певними категоріями, тобто створювати різні списки і додавати в них завдання. Це може знадобитися для сортування завдань.

Можливості для командної співпраці, залучення учасників, керування проектами, відслідковування робочого простору команди та інших більш цікавих інструментів таких як встановлення повторюваних завдань, кольорові мітки, інтеграція з певними сервісами, доступні лише на основі відповідних тарифних планів, які користувачі можуть визначати і використовувати відповідно до своїх потреб.

Інтерфейс Any.do є досить сучасним, але він вже має набагато більше елементів ніж розглянуті до цього застосунки. Для використання розширених можливостей програми, можливо потрібно буде докласти значних зусиль. Наприклад, керування кількома інтеграціями зі сторонніми програмами в Any.do може виявитися складним для пересічного користувача. Інтерпретація значної кількості інтерактивних значків та іконок також може викликати невеликі труднощі [9]-[10].

Окрім цього подекуди шрифти елементів можна вважати недостатньо читабельними, наприклад, досить важлива кнопка «Позначити як виконане» сама є невеликою та має маленький шрифт.

Any.do надає можливість зв'язатися зі службою підтримки через онлайн-форму. У довідковому центрі програми користувачі можуть шукати та переглядати базу знань для отримання довідкової документації. Там можна отримати інформацію про робочі області, навчальні посібники, відповіді на поширені запитання та загальні ресурси спільноти [10].

На рис. 1.3 зображено інтерфейс застосунку керування завданнями Any.do [11].

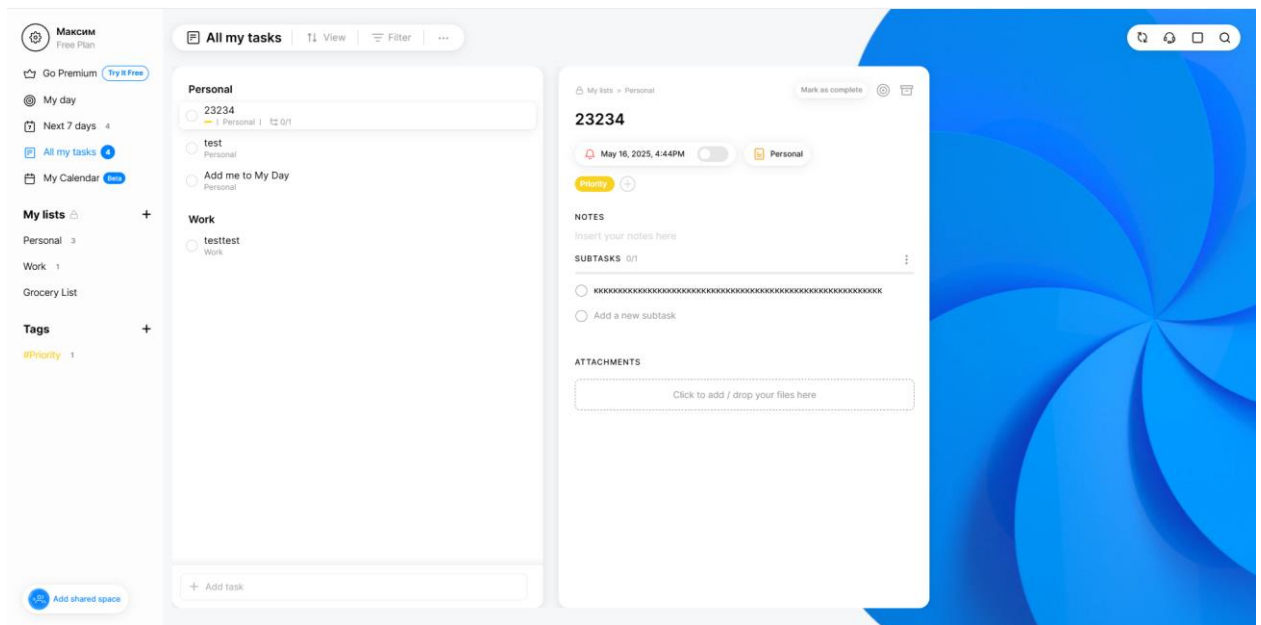


Рисунок 1.3 – Інтерфейс Any.do [11]

1.2.4 Вебзастосунок для планування TickTick

TickTick – це кросплатформний застосунок для створення планів, запущений у 2013 році одноіменною компанією.

Він є доступним як окремий застосунок для iOS, Android, Windows, Linux, та як вебверсія у будь-якому браузері. Саме її і буде надалі розглянуто.

TickTick як і Any.do має безкоштовний та преміальний тарифні плани. І відповідно безкоштовна версія застосунку має неповний функціонал, але достатній для базового планування задач.

У безкоштовній версії користувач може створювати задачі, додавати задачі до списків, та додавати учасників до списку. Доступно для створення лише 9 списків, з 99 завданнями в кожному, та 19 підзавдань для кожного завдання. До кожного списку можна додати лише одного учасника [12].

До завдань можна додавати нагадування на певну дату та час, визначати за який час до настання нагадати та встановлювати повторюваність. Є можливість встановлювати пріоритетність та закріплювати задачі.

Інтерфейс вебзастосунку TickTick є найменш привабливим і зручним серед усіх розглянутих. Дуже багато маленьких шрифтів та іконок. Плейсхолдери для створення завдання, фільтрів, підзавдань та коментарів є ледва помітними. Окрім цього деякі функції розташовуються в неочевидних місцях, наприклад меню текстового редагування. Також варто зазначити що з призначенням деяких іконок та функцій пересічному користувачу буде нелегко розібратись. Загалом вебверсія цього застосунку забезпечує не найкращий користувацький досвід.

На рис. 1.4 зображено інтерфейс вебзастосунку TickTick [13].

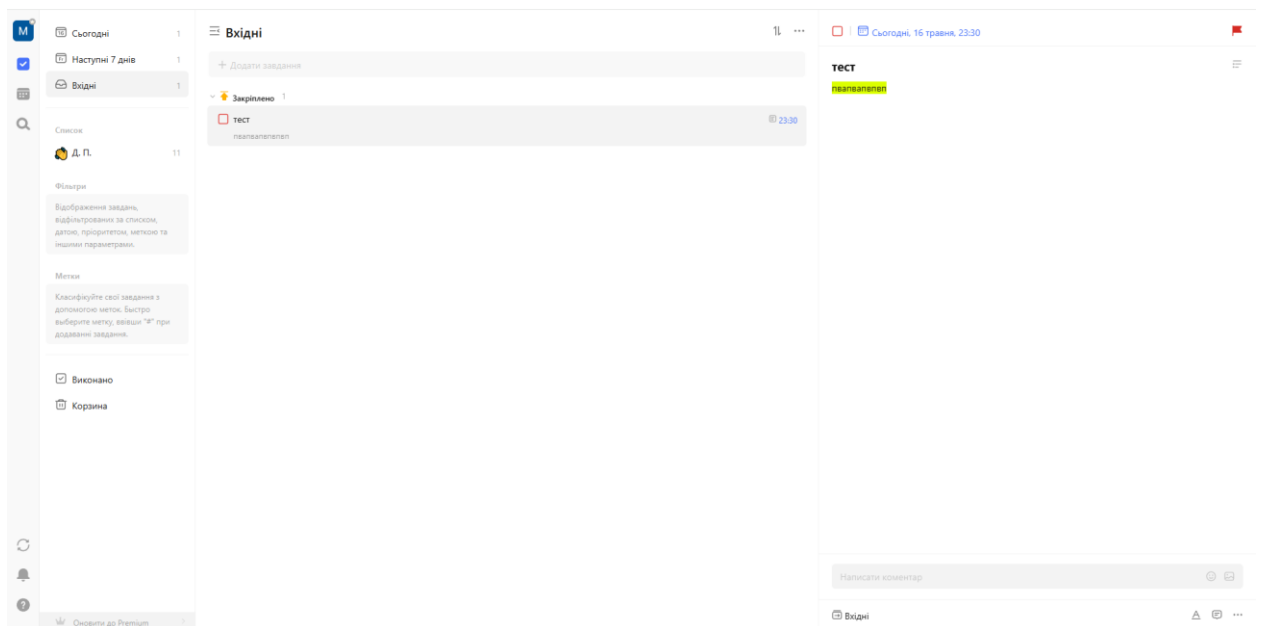


Рисунок 1.4 – Інтерфейс TickTick [13]

1.2.5 Порівняння існуючих та розроблюваного продуктів

Вище було розглянуто та описано найбільш відомі та використовувані вебсервіси для планування завдань та організації власного часу. Загалом вони всі є досить схожими, і надають достатній базовий набір можливостей для використання у розглянутій проблемній області. Звісно що деякі з них мають певні недоліки. Наприклад, відсутність або обмеженість функцій у

безкоштовній версії, або недостатньо зрозумілий інтерфейс, що може ускладнити намагання користувача пізнати функціонал застосунку.

В табл. 1.1 наведено порівняння розглянутих аналогів (їх безкоштовний версій, якщо існують платні тарифи) та програмного забезпечення, що буде розроблено в ході виконання роботи – ВЗОЧПП (Вебзастосунок для організації часу та планування подій).

Таблиця 1.1 – Порівняння програмного забезпечення для планування

Параметр	Google Tasks	Google Keep	Any.do	TickTick	ВЗОЧ ПП
CRUD-операції з подіями	+	+	+	+	+
Налаштування нагадувань	+	+	+	+	+
Залучення інших користувачів до задачі	—	+	+—	+—	+
Можливість додавати файли	—	+	+	—	—
Кросплатформність/ Кросбраузерність	+	+	+	+	+
Простота і зручність інтерфейсу	+	+	+—	+—	+
Безкоштовність	+	+	+—	+—	+

1.3 Постановка завдання роботи

За результатами вивчення предметної області та проведеного аналізу наявних робочих і застосовуваних аналогічних рішень було сформульовано та поставлено задачу на подальшу роботу.

Метою даної роботи є підвищення користувацького досвіду у плануванні щоденних справ, організації особистого часу, створенні подій, нагадувань і зустрічей, а також зберіганні та керуванні контактами.

Основними функціональними вимогами до проєкту є:

- внесення планів на певну дату і час;
- можливість додавання подій;
- можливість редагування та видалення подій;
- можливість додавання контактів;
- можливість редагування та видалення контактів;
- можливість додати нотатки до події;
- можливість вказати учасників та надіслати їм повідомлення;
- налаштування сповіщень про майбутні події, що скоро настануть;
- реєстрація та авторизація користувачів.

Інтерфейс розроблюваного вебзастосунку має бути простим, інтуїтивно зрозумілим і ефективним. Він має надавати користувачам змогу без зайвих зусиль виконувати необхідну роботу, та зосереджуватися на власних справах а не на «боротьбі» із системою.

Продукт має бути розроблено з урахуванням можливостей подальшого покращення та розширення.

1.4 Висновки з розділу 1

У даному розділі було розглянуто та проаналізовано предметну область виконуваної роботи, а саме призначення і актуальність використання

планувальників, що характеризується тенденціями стрімкого зростання попиту на цифрові засоби планування і ведення завдань, як загалом майже у всіх сферах нашого життя на сьогоднішній день. Визначено що використання цих інструментів дозволяє підвищити як індивідуальну так і командну продуктивність, уникати зайвих витрат ресурсів і часу, та запобігати вигоранню.

Розглянуті під час дослідження існуючі програмні рішення мають багато спільних рис. Загалом їх можна охарактеризувати як достатньо непогані інструменти що надають необхідний набір функцій для створення і ведення хоча б мінімального особистого розпорядку. Звісно, як і будь-який програмний продукт, вони мають свої недоліки, що в цілому можна охарактеризувати як такі, що впливають на користування не занадто критично, хоча звісно все залежить від потреб користувача.

Таким чином, після детального аналізу і порівнянь було сформульовано мету подальшої роботи, основні функціональні можливості вебзастосунку та вимоги до інтерфейсу.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МАТЕРІАЛІВ ТА МЕТОДІВ

Даний розділ містить розгляд основних проектних рішень щодо системи що розробляється, опис обраних мов програмування та середовища розробки, а також певні архітектурні рішення системи.

2.1 Вибір мови програмування

При виборі мови програмування для розробки застосунків варто звертати увагу на різні аспекти, основними з яких є зручність, ефективність, легкість підтримки та масштабування, а також безпекові фактори та наявність великої кількості інструментів. Окрім цього варто враховувати наявність якісних фреймворків та активної підтримки спільноти. Для кожної частини проекту, вибір мови програмування є досить важливим кроком в процесі розробки.

Бекенд потрібно розглядати як серверну частину застосунку що здійснює обробку даних, керує базами даних, виконує автентифікацію користувачів та забезпечує безперебійну роботу проекту. Якщо розглядати вебзастосунок як айсберг, то бекенд – це саме та частина, що, умовно кажучи, прихована під водою, і непомітна та недоступна звичайному користувачу [14].

Фронтенд, в свою чергу – це та частина, яку користувач бачить та безпосередньо взаємодіє. Це наприклад, форми, поля, кнопки, чекбокси, навігація та зображення тощо. Окрім звичного відображення та краси ця частина відповідає і за підвантаження, адаптивність, попередню валідацію даних, скрипти що обробляють та надсилають запити [15].

Мову програмування Java було обрано для розробки серверної частини вебзастосунку. Java – високорівнева об'єктно орієнтована мова програмування, яку використовують для різних платформ. Вона є дуже популярною серед веброзробників, відповідальних за бекенд. Вона має досить

зрозумілий і зручний синтаксис, який схожий на C та C++, але досить спрощений та більш читабельний [16]-[17].

Java має достатньо жорстку типізацію даних що сприяє написанню зрозумілого коду та уникненню помилок і допомагає знаходити несправності ще на етапі компіляції. Окрім цього вона не потребує ручного управління пам'яттю завдяки автоматичному збору і вивільненню [16].

До того ж ця мова є портативною. Її досить відомий принцип, що дослівно перекладається як «Пиши один раз, запускай будь-де» описує, що будь-яка програма написана на Java, може працювати в будь-якому середовищі яке підтримує JVM (Віртуальна машина Java) [17].

Java має досить якісні вбудовані механізми безпеки, що сприяє розробці захищених вебзастосунків. До того ж в ній досить багато вбудованих інструментів та бібліотек, зокрема і для розробки вебсервісів.

Важливо зазначити що Java має досить широку екосистему потужних фреймворків, таких як Spring, Hibernate, JSF, Struts, що роблять розробку набагато ефективнішою та швидшою.

Однак для веброзробки використовують і інші мови програмування, тому додатково до вже наведеного обґрунтування, далі представлено порівняння мов у табл. 2.1

Таблиця 2.1 – Порівняння мов для веброзробки

Критерій	Java	Python	JavaScript (Node.js)	PHP
Продуктивність/ швидкість	+	+—	+—	+—
Зручність	+	++	+	+
Кросплатформність	+	+	+	+

Продовження таблиці 2.1

Критерій	Java	Python	JavaScript (Node.js)	PHP
Механізми безпеки	+	+	+—	+—
Фреймворки	++	++	++	+
Робота з БД	++	+	+	+
Масштабованість	++	+	+	+—
Безкоштовність	+	+	+	+

До того ж Java має дуже активну спільноту по всьому світу та широку якісну документацію, що робить використання цієї мови ще більш досконалим. Java використовують для розробки ПЗ такі відомі компанії як Amazon, Netflix, Google, Spotify, LinkedIn [17].

Для розробки клієнтської частини частини застосунку було використано мову програмування TypeScript.

TypeScript це не просто окрема мова програмування, це певна надбудова над JavaScript. Тобто TypeScript підтримує усі функції та можливості JavaScript але має певне доповнення – систему типів. Ця мова підтримує статичну типізацію, що допомагає виокремлювати помилки та несподівану поведінку ще на етапі компіляції, а не виконання програми. Це дозволяє створювати більш надійний і підтримуваний код [18].

Доповнюючи наведений опис варто сказати що, TypeScript ніколи не змінює поведінку коду JavaScript під час виконання. Це означає, що при перенесенні коду з JavaScript у TypeScript, він гарантовано працюватиме так само, навіть якщо TypeScript вважатиме, що код містить помилки типу [19].

TypeScript підтримує інтерфейси, узагальнення, успадкування та модифікатори доступу до методів, що робить код більш передбачуваним та безпечним для типів. Інтерфейси визначають формальні угоди між різними частинами коду, узагальнення забезпечують перевірку під час компіляції та можливість повторного використання, а успадкування полегшує повторне використання і дозволяючи новим класам розширювати ті, що вже є [20].

Порівняння та відмінності TypeScript і JavaScript наведено у табл. 2.2.

Таблиця 2.2 – Порівняння TypeScript і JavaScript

Характеристика	TS	JS
Типізація	Сильна, статична типізація; підтримує динамічну.	Лише динамічна типізація.
Підтримка ООП	Повна (класи, інтерфейси, дженеріки, модифікатори доступу тощо).	Обмежена (класи ES6, нестрога типізація)
Компіляція	Компілюється в JS.	Непотрібна.
Сумісність з IDE	Підтримує автодоповнення, рефакторинг, визначення помилок	Базова підтримка, більш обмежена.
Робота з фреймворками	Широке коло зручних фреймворків.	Здебільшого ті ж фреймворки, але деякі не оптимізовано або не працюють взагалі (NestJS, tRPC).
Підтримка спільноти	Постійно зростаюча спільнота.	Найбільша спільнота серед усіх мов програмування.

2.2 Вибір фреймворку для створення ПЗ

В якості фреймворку для розробки серверної частини вебзастосунку для планування подій на мові програмування Java було обрано Spring, зокрема Spring Boot.

Spring Framework – один з найпопулярніших і найпотужніших фреймворків для створення вебзастосунків. Spring Framework надає комплексну модель програмування та конфігурації для сучасних застосунків на базі Java, що робить розробку швидшою, масштабованішою та зручнішою в обслуговуванні.

Маючи в своїй основі інверсію керування (IoC) та впровадження залежностей (DI), фреймворк забезпечує міцне підґрунття для широкого набору функцій і можливостей. Інверсія керування – це принцип проектування, що використовується в об'єктно-орієнтованому програмуванні, де керування створенням об'єктів та управлінням залежностями передається з коду програми до зовнішнього фреймворку або контейнера. Такий підхід зменшує складність ручного керування залежностями та дозволяє створювати більш модульний та гнучкий код. Впровадження залежностей – це шаблон проектування, який використовується в розробці програмного забезпечення для реалізації інверсії керування. Він дозволяє класу отримувати свої залежності із зовнішнього джерела, а не створювати їх усередині класу. Це зменшує залежність між класами та робить систему більш зручною в обслуговуванні [21]-[22].

Spring Boot, в свою чергу, надає інструменти для легкого створення окремих застосунків, які можна відразу легко запускати. Він сприяє мінімізації труднощів конфігурації та інтеграції з основним фреймворком Spring [23].

Однією з найважливіших особливостей Spring Boot є автоконфігурація, що робить можливим налаштування проекту більш автоматично, ґрунтуючись на його структурі і залежностях. Це мінімізує необхідність ручного втручання,

але не суперечить йому, тобто розробник може визначати і власну конфігурацію [23].

Spring Boot ще спрощує розробку, пропонуючи вбудовування серверів застосунків, таких як Tomcat, Jetty або UnderFlow. Ці вбудовані сервери попередньо налаштовані та дозволяють розробникам запускати застосунки без необхідності зовнішнього розгортання. Вбудовані сервери є однією з визначних особливостей Spring Boot, що дозволяє програмам запускатися без необхідності налаштовувати зовнішній сервер. Spring Boot автоматично налаштовує Tomcat на основі залежностей, а створену програму можна пакувати у JAR-файл і запускати будь-де, що також сприяє контейнеризації застосунків та їх подальшому розгортанню [24]-[25].

Фреймворк забезпечує багато корисних можливостей для розробки. Наприклад технології Spring Security, що надають широкий спектр засобів для безпеки вебзастосунків. Вони допомагають налаштовувати авторизацію, автентифікацію, інтеграцію JWT - токенів, та керувати сесансами. Або Spring Data що забезпечує зручний доступ до даних та покращує роботу з базами даних.

Документація фреймворку є дуже широкою і зрозумілою, і забезпечує якісне розуміння функцій і технологій, значно полегшуючи навчання.

До того ж існує дуже зручний і популярний інструмент для швидкого створення Spring Boot проектів з усіма необхідними залежностями, це – Spring Initializr. Цей вебінструмент допомагає розробникам налаштувати новий проект докладаючи мінімальних зусиль. Він підтримує мови Java, Kotlin, Groovy та збірки Maven і Gradle [26].

Для розробки клієнтської частини проекту було вибрано фреймворк Angular.

Angular – це серйозний фреймворк, побудований на TypeScript, що дозволяє розробляти швидкі і надійні застосунки. Підтримуваний командою Google, він пропонує комплексне рішення для фронтенд-розробки з інструментами для обробки форм, маршрутизації, HTTP-сервісів тощо [27].

Однією з особливостей Angular є компонентний підхід. Компоненти – це окремі блоки, що складаються з коду TypeScript, що визначає основну поведінку, HTML-шаблонів, які контролюють що відображається в DOM та стилів CSS. Разом ці блоки утворюють користувацький інтерфейс програми. Компоненти легко повторно використовувати в різних частинах проекту, що покращує підтримуваність і читабельність [28]-[29].

Також варто згадати про Angular CLI – інструмент інтерфейсу командного рядка, який дозволяє створювати, розробляти, тестувати, розгортати та підтримувати застосунки за допомогою команд в консолі [30].

Angular підтримує двостороннє зв'язування даних, тобто можливість зв'язувати дані між компонентами і шаблонами. Це сприяє динамічній зміні даних в застосунку, коли вони є в даних моделі. Сервіси визначають загальну логіку і функції програми. А ін'єкції залежностей, в свою чергу, дають змогу компонентам отримувати доступ до сервісів без ручного створення їхніх екземплярів. Окрім цього в Angular є зручний механізм маршрутизації, що зв'язує компоненти з потрібними URL для забезпечення переходів між різними сторінками. Загалом, Angular надає усі потрібні інструменти для створення SPA. SPA – односторінкові застосунки, функції яких існують на одній HTML – сторінці, а браузеру потрібно відображати лише ті частини, які важливі для користувача, замість завантаження нової сторінки. Саме такого підходу і притримується клієнтська частина розроблюваного застосунку [31].

Цей фреймворк надає зручну та зрозумілу структуру проекту, і дозволяє підтримувати її такою, завдяки якісним інструментам та простим командам. А встановити його можна досить просто за допомогою `npm`, використовуючи команду `npm install -g @angular/cli`.

2.3 Вибір середовища розробки

Як обрати правильне і зручне середовище розробки це досить важливе питання для створення майже будь-якого проекту.

Тому, враховуючи вже обрані і описані рішення, вибір припав на IntelliJ IDEA IDE. IntelliJ IDEA, створена компанією JetBrains є найпотужнішим рішенням для розробки на Java та Kotlin.

78% розробників Java обирають IntelliJ IDEA [32].

IntelliJ IDEA надає повну підтримку вже загаданого Spring і SpringBoot. Інтелектуальне автодоповнення коду допомагає швидше писати якісний код на основі Spring. IDE враховує контекст і пропонує лише найдоречніші варіанти як у файлах Java (або Kotlin) оснований на Spring, але й у файлах властивостей конфігурації, посиланнях на URL-шляхи та багатьох інших випадках. IntelliJ IDEA виявляє помилки та ймовірні баги у Spring-проекті ще до компіляції та запуску. Вона надає пояснення та пропонує можливі виправлення, має багато перевірок специфічних для Spring, що охоплюють широкий спектр проблем. Ще тут є повна підтримка Lombok, яка включає навігацію до автоматично підключених залежностей або кандидатів, а також швидке виправлення відсутніх залежностей компонентів [33].

Варто зазначити що розумне автодоповнення це загальна можливість цієї IDE, що працює майже для будь-якого коду. Вона також має якісну підтримку REST API, надаючи автоматичне розпізнавання контролерів і допомогу в побудові маршрутів.

Окрім Java та Kotlin, IntelliJ IDEA підтримує JavaScript (TypeScript), HTML, CSS та JSON. Тому цю ж IDE використано для розробки клієнтської частини вебзастосунку з Angular. Оскільки вона має вбудовані консольні інструменти, тут є зручна інтеграція з AngularCLI.

Порівняння IntelliJ IDEA з іншими IDE для роботи з Java наведено у табл. 2.3.

IntelliJ IDEA, як і більшість продуктів JetBrains, має шикарний зручний і сучасний інтерфейс, з ергономічним розташуванням вікон і інструментів, що робить розробку в ній високопродуктивною.

Таблиця 2.3 – Порівняння IDE

Параметр	IntelliJ IDEA	Eclipse	ApacheNetBeans
Підтримка мов	Java, Kotlin, Groovy, JS (TS), Python, HTML	Java, C/C++, PHP, JS	Java, C/C++, PHP, JS, HTML
Кросплатформність	+	+	+
Автодоповнення	Найкраще, рівня ІІІ	Непогане	Середнє
Продуктивність	Швидка, висока оптимізація	Середня, потребує налаштувань	Середня
Підтримка Git	Вбудована і дуже зручна	Доступна через плагіни	Вбудована, але посередня
Підтримка фреймворків	Найкраща, велике коло фреймворків, якісна підтримка	Добра, іноді потребує плагінів	Порівняно обмежена
Безкоштовність	Common – безкоштовна, Ultimate – платна	Безкоштовна	Безкоштовна

Хоча Ultimate версія IntelliJ IDEA, яка власне і має більшість описаних переваг, є платною, вона має безкоштовну пробну ліцензію, яка забезпечує усе перераховане і доступна вона протягом 30 днів.

А використання однієї IDE для створення усіх частин проекту дозволяє фокусуватися на вирішенні нагальних питань, і допомагає швидше відслідковувати помилки що виникають під час розробки.

2.4 Вибір системи управління базами даних

Використання систем керування базами даних, має велику вагу при розробці майже будь-якого застосунку в наш час, незалежно від його призначення та масштабу, оскільки це забезпечує зручну і надійну роботу з даними. Тому при виборі СКБД варто особливо звертати увагу на якість механізмів безпеки і можливість забезпечити стабільну роботу і широкий функціонал.

Для даної роботи було обрано СКБД MySQL. MySQL – найпопулярніша в світі реляційна система керування базами даних, в основі якої закладено мову SQL. Вона зберігає дані в таблицях рядків і стовпців, організованих у схеми, що визначають, як дані організовані та збережені і описує зв'язок між різними таблицями. Завдяки цьому легко зберігати, отримувати та аналізувати багато типів даних [34].

MySQL підтримує ACID транзакції. Це гарантує, що операції з базою даних виконуються повністю або не виконуються взагалі, дані залишаються в послідовному стані, операції не заважають одна одній, а зміни зберігаються навіть у разі збоїв. Це є фундаментальним принципом надійності для будь-якої бізнес-логіки [34].

Ця СКБД дуже добре інтегрується як і з IntelliJ IDEA, так і зі Spring Boot з JPA/Hibernate, що використовуються в проекті. Це дає змогу абстрагуватися від низькорівневих SQL-запитів і працювати з об'єктами Java, що значно прискорює розробку та підтримуваність коду.

MySQL має відкритий вихідний код, і є повністю безкоштовним. Розповсюджується під загальною публічною ліцензією GNU (GPL).

Завдяки своїй відкритості і популярності MySQL має величезну спільноту і широкий доступ до різних ресурсів підтримки.

У проекті її використання відбувалося шляхом розгортання контейнеру в Docker, з відповідного образу mysql, що відкрито доступний у Docker Hub. Такий підхід забезпечує простоту використання, шляхом зручних команд та

усуненням необхідності встановлювати та налаштовувати різні необхідні інструменти для роботи з БД і створення сервера, а також швидкість запуску, портативність та масштабованість.

2.5 Висновки з розділу 2

У цьому розділі було розглянуто і обґрунтовано обрані рішення щодо використання мов програмування, фреймворків, IDE та СКБД.

Для серверної частини використовується мова програмування Java та фреймворк Spring Boot, через свої широкі можливості для створення вебзастосунків, надійність і при цьому зручність використання.

Для клієнтської частини було обрано TypeScript та Angular, тому що TypeScript забезпечує чітку типізацію, і допомагає уникати помилок. Angular, в свою чергу це потужний фреймворк, що надає чітку структуру і широке коло зручних інструментів для різних можливостей вебзастосунку.

В якості IDE було обрано IntelliJ IDEA за його продуктивність, ергономічність та якісну підтримку усіх необхідних функцій.

Вибір MySQL, в якості системи керування базами даних, обґрунтовано її надійністю, продуктивністю та перевіреністю часом.

3 ОПИС АРХІТЕКТУРИ ТА РОЗРОБКИ ПРОГРАМИ

3.1 Проектування архітектури програмного забезпечення

В основі архітектури даного вебзастосунку покладено шаблон проектування MVC – Model-View-Controller. MVC поділяє програму на три основні логічні компоненти, а саме: model, що відповідає за роботу з даними та бізнес-логіку, view – представлення, що відповідає за всю логіку відображення інтерфейсу та інформації, controller, який здійснює обробку запитів, взаємодіє з моделлю для маніпуляції з даними, та робить подальший вибір відповідного представлення і відправку відповіді.

MVC є одним з найпопулярніших архітектурних шаблонів для розробки вебзастосунків, завдяки своїй логічності, простоті, масштабованості і можливості працювати над кожним фрагментом окремо, знижуючи таким чином кількість помилок і їх впливовість. Його підтримує велика кількість фреймворків.

Архітектуру вебзастосунку поділено на три основних модулі: клієнтська частина, серверна частина REST API, що виконує основні функції, та сервер авторизації, що відповідає за створення нових користувачів та авторизацію існуючих, тобто видачу і перевірку JWT-токенів. Кожна серверна частина має свою окрему БД.

Клієнтська частина, розроблена за допомогою фреймворку Angular, являє собою односторінковий застосунок (SPA). SPA (Single page application) – вебзастосунок, який завантажує лише одну HTML-сторінку, при першому візиті користувача, зазвичай це index.html. В подальшому при взаємодії з елементами та навігації по застосунку, контент такої сторінки оновлюється динамічно, за допомогою JavaScript (TypeScript). Коли користувач перемикається між різними представленнями, Angular не робить запит на сервер за новою сторінкою. Для цього використовується служба Router. Він перехоплює зміну URL, визначає, який компонент потрібно відобразити і динамічно завантажує його у відповідному місці (як правило, в середині тегу

<router-outlet>). Цей процес здійснюється на стороні клієнта і не потребує повного перезавантаження сторінки чи створення нових.

Серверна частина REST API, розроблена за допомогою фреймворку Spring Boot, який широко підтримує MVC, та забезпечує автоконфігурацію стартових пакетів, що дозволяє легко впроваджувати необхідні залежності та бібліотеки. Вона оброблює запити авторизованого користувача, реалізує основну бізнес-логіку функціональності застосунку та взаємодіє з базою даних. Такий підхід забезпечує швидкість розробки, гнучкість і безпеку. А також сприяє масштабованості та легкій розширюваності.

Серверна частина авторизації, створена за допомогою Spring Authorization Server. Spring Authorization Server – це спеціальний окремий модуль на базі Spring Security, який підтримує OAuth2 та OIDC (OpenID Connect) для Spring застосунків. OAuth2 та OIDC це протоколи авторизації та аутентифікації, що дозволяють клієнтському застосунку отримувати доступ до ресурсів користувача, які знаходяться на окремому сервері.

Рівень бази даних представлений реляційною СКБД MySQL. Це забезпечує водночас простоту, надійність і продуктивність. Широка підтримка зі сторони Spring Boot, зокрема MySQL драйверів, інтеграцій Spring Data JPA і Hibernate, що забезпечують потужну абстракцію, та транзакціональності, що спрощує атомарність при роботі з базою, забезпечує швидкий і зручний доступ до даних, та дозволяє зосередитись на бізнес-логіці.

Загалом архітектуру розроблюваного застосунку можна охарактеризувати як клієнт-серверну з мікросервісним підходом. Схематичне зображення архітектури наведено на рис. 3.1.

Такий архітектурний підхід забезпечує високу продуктивність, ефективну обробку запитів, зручну роботу з інтерфейсом. А зрозумілий розподіл функціонування вебзастосунку спрощує підтримку та розширення кожного компонента.

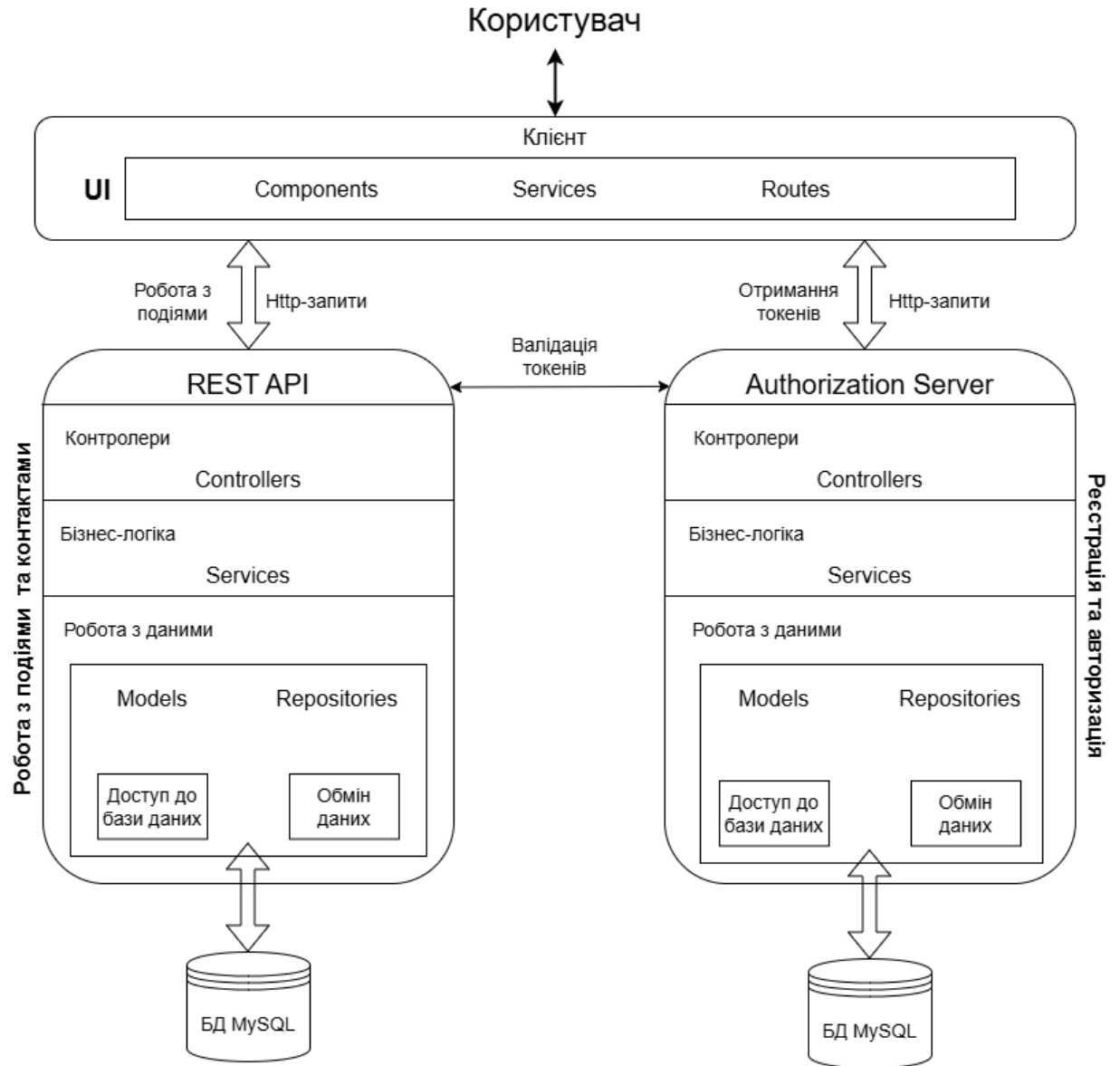


Рисунок 3.1 – Архітектура вебзастосунку

3.2 Структура розроблюваного вебзастосунку для організації часу та планування подій

Структуру вебзастосунку можна показати у вигляді UML-діаграми класів.

UML (Unified Modeling Language) – це стандарт візуального моделювання, що використовується для представлення архітектур, структур,

специфікацій різного роду систем, а також відображення різноманітних процесів.

Діаграма класів – це структурні діаграми, що найчастіше використовуються для об’єктно орієнтованих систем. Вони показують класи, їхні атрибути та методи, а також відношення і зв’язки між класами.

Основними класами вебзастосунку для організації часу та планування подій є: подія, яка планується (Event), контакт користувача, який також може бути пов’язаний з подією (Contact), сповіщення для запланованої події, яке надсилається користувачу, якщо він обрав таку опцію (Notification), та власне користувач, що є власником цього всього (User).

Зображення діаграми класів, створеної для програмної розробки показано на рис. 3.2.

Іншими класами що використовуються в програмі здебільшого є контролери (наприклад EventController) що відповідають за прийом і обробку HTTP-запитів (GET, POST, PUT, DELETE), відправку відповідей, сервіси (наприклад EventService) що виконують основну бізнес-логіку, репозиторіїв (наприклад EventRepository), які є інтерфейсами, що організовують взаємодію з базою даних для відповідних сутностей.

Загальну структуру проекту наведено за допомогою схеми, яка демонструє, як саме влаштовані дерева проектів розроблюваних частин програми.

Схему структури проекту, у вигляді дерев показано на рис. 3.3.

Варто зазначити що .java файли в деревах серверних частин це основні класи проектів, з анотацією @SpringBootApplication, що забезпечує автоматичне налаштування, сканування компонентів та основні функції Spring Boot.

А файли pom.xml являють собою об’єктну модель проекту та містять основну інформацію щодо проекту, конфігурації збірки, підключені залежності та плагіни.

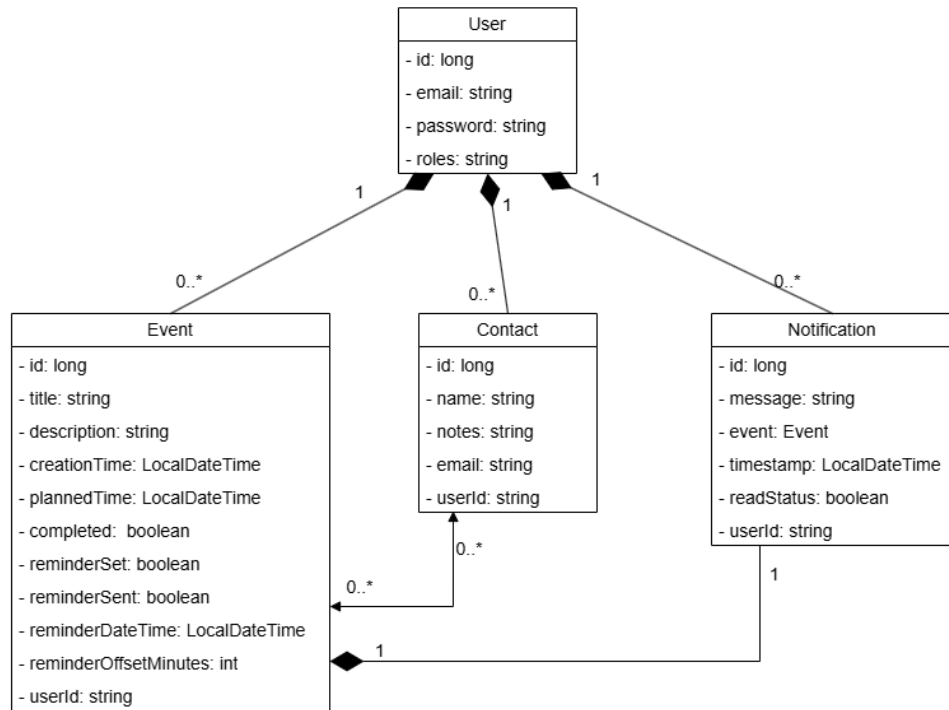


Рисунок 3.2 – Зображення діаграми класів

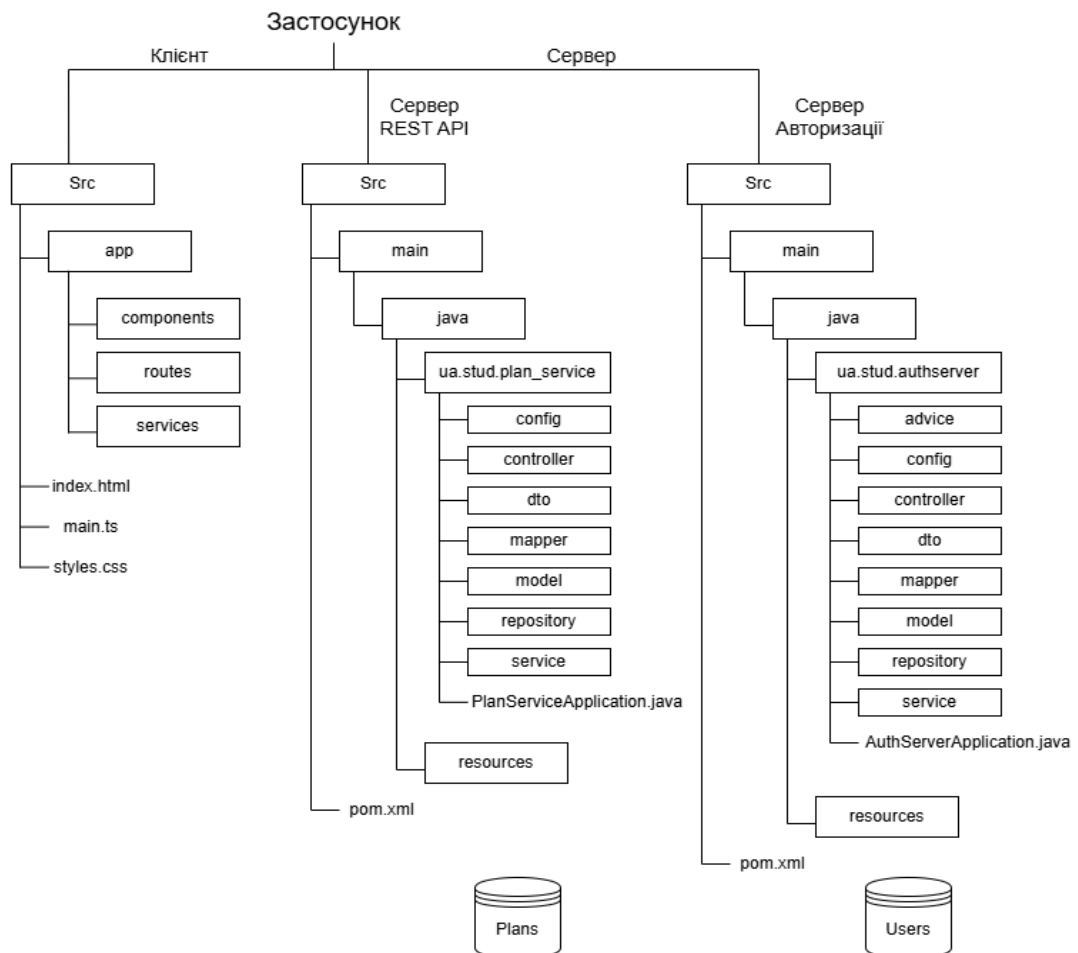


Рисунок 3.3 – Зображення дерев проекту

3.3 Огляд функціонування програмного забезпечення

Для початку варто розглянути функціонування з точки зору взаємодії користувача із системою. Загалом її можна розглядати з точки зору двох, певним чином пов'язаних варіантів використання – робота з подіями і робота з контактами.

Схему взаємодії користувача із запланованими подіями показано, за допомогою діаграми прецедентів на рис. 3.4

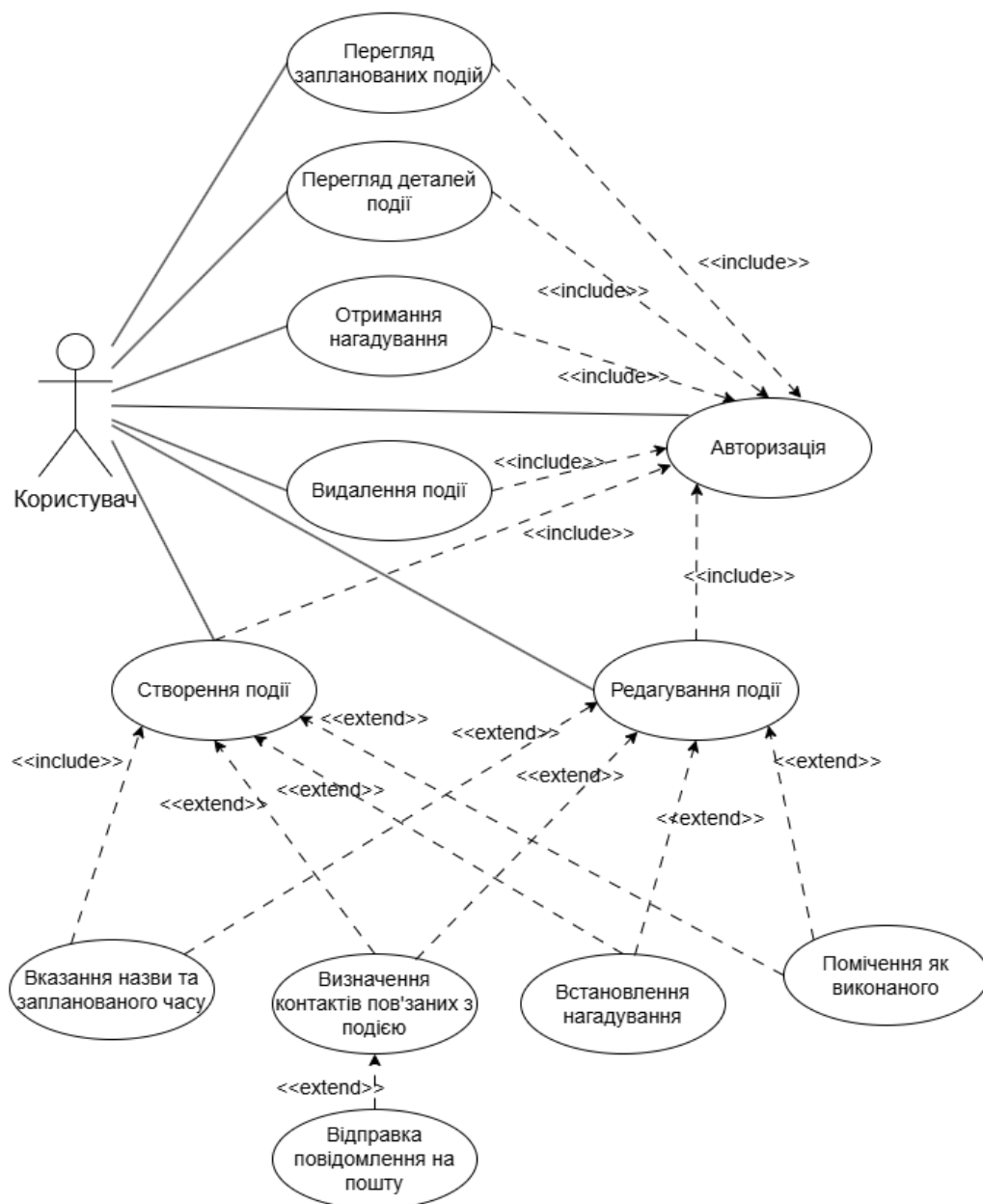


Рисунок 3.4 – Діаграма прецедентів взаємодії користувача з подіями

Для кращого розуміння алгоритмів роботи з програмою, нижче наведено потік виконання для створення події.

Передумови: система працює, користувач авторизований.

Етапи основного потоку:

- перейти на форму створення;
- ввести назву події і заплановану дату;
- натиснути «Створити подію»;
- подія відображається в списку.

Етапи альтернативного потоку:

- перейти на форму створення;
- ввести назву події і заплановану дату;
- встановити нагадування;
- обрати час за який нагадати;
- натиснути «Створити подію»;
- подія відображається в списку;
- приходить нагадування на запланований час.

Помилковий потік:

Якщо, користувач вводить забагато символів у назві, або некоректний час, йому буде виведено повідомлення про некоректність даних.

Постумова: створено заплановану подію.

Схему взаємодії користувача із контактами показано, за допомогою діаграми прецедентів на рис. 3.5

Для прикладу нижче наведено потік видалення контакту.

Передумови: система працює, користувач авторизований.

Етапи основного потоку:

- перейти до списку контактів;
- обрати контакт для видалення;
- натиснути «Видалити»;
- підтвердити видалення.

Етапи альтернативного потоку:

- перейти до списку контактів;
- обрати контакт для видалення;
- натиснути «Видалити»;

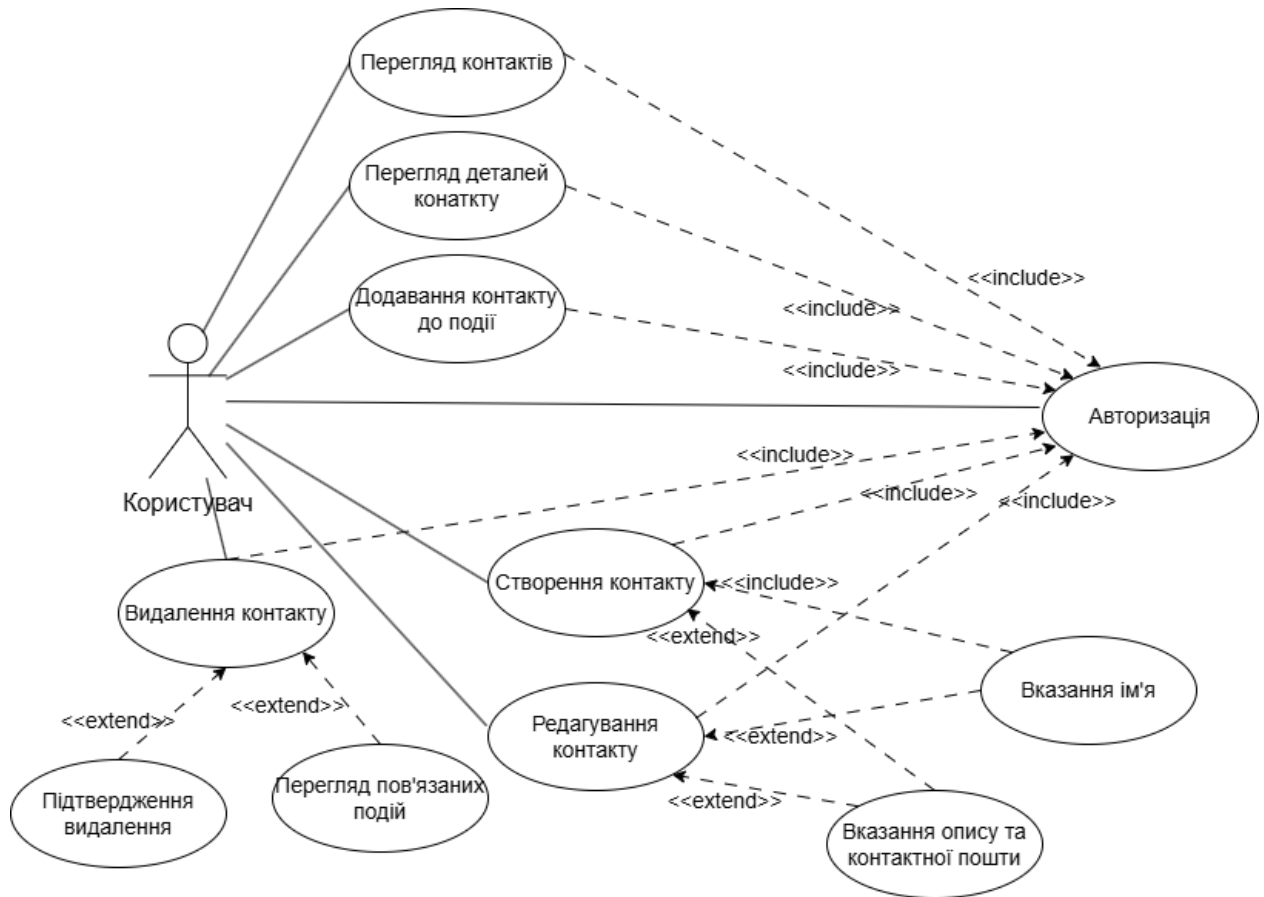


Рисунок 3.5 – Діаграма прецедентів взаємодії користувача з контактами

- підтвердити видалення;
- ознайомитись зі списком пов'язаних подій;
- видалити або скасувати видалення.

Постумова: видалено контакт або скасовано видалення.

Також варто зазначити що якщо користувач, при створенні контакту вкаже валідну контактну електронну поштову адресу у відповідне поле, то при додаванні контакту до події, вказаний адресант отримає відповідне повідомлення на свою електронну пошту.

Процес використання системи візуалізовано за допомогою блок-схеми на рис. 3.6.

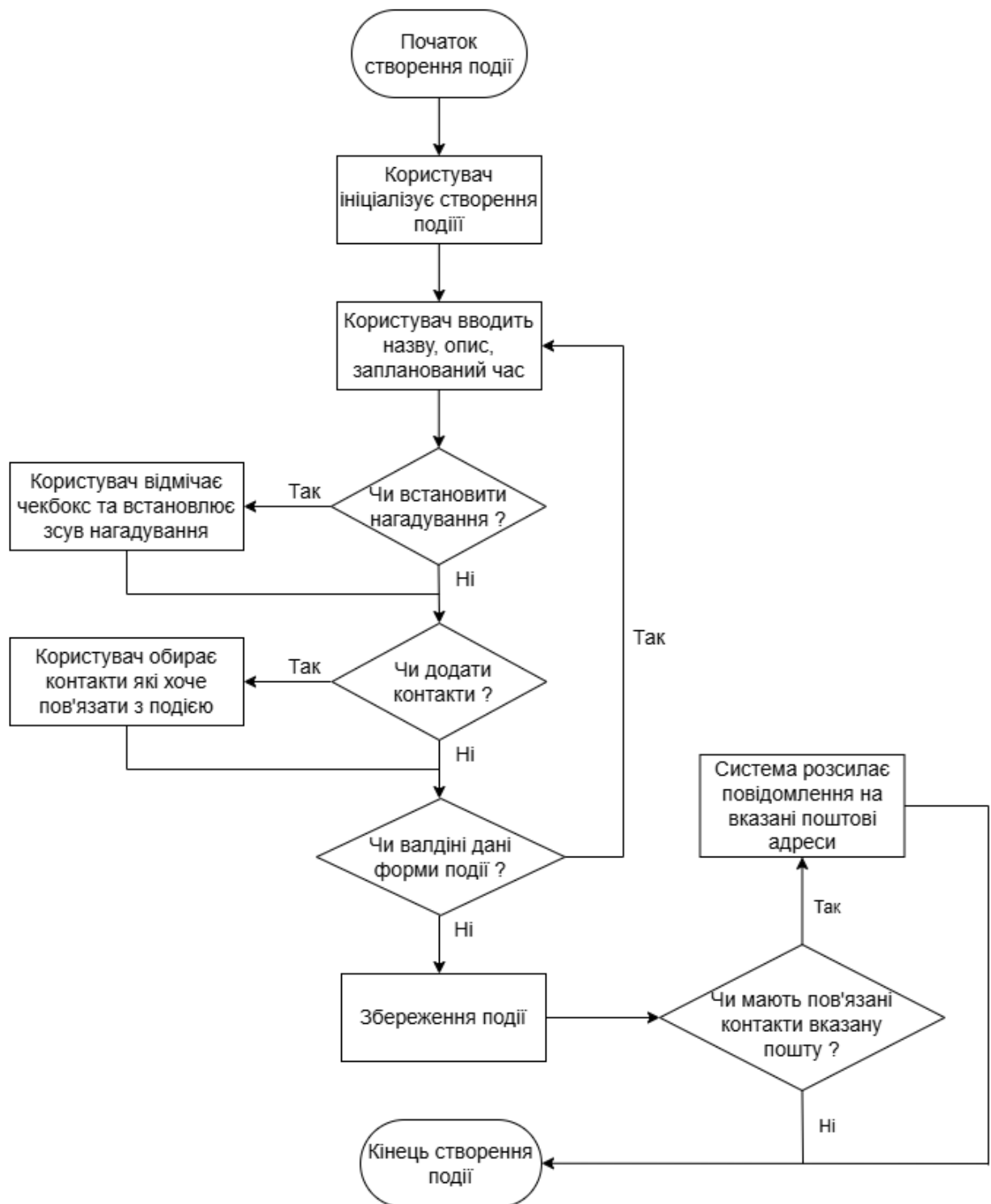


Рисунок 3.6 – Блок-схема створення події

В ході розробки було імплементовано класи та методи необхідні для обробки запитів та функціонування бізнес-логіки. Деякі з них наведено нижче.

Для серверної частини:

Варто зазначити що для передачі даних з бекенду на фронтенд було використано класи DTO (Data Transfer Object), які чітко визначають, які дані очікуються на вхід та що буде повернуто на вихід, незалежно від внутрішньої структури сутностей. Це сприяє розділенню відповідальності, безпеці та уникненню відправки зайвих даних.

Клас `EventController` відповідає за прийом запитів і відправку відповідей, що стосуються роботи з подіями. Усі методи потребують авторизованості користувача. Методи класу наведено в табл. 3.1.

Таблиця 3.1 – Методи класу `EventController`

Сигнатура	Призначення
<code>getUserIdFromAuthentication(Authentication authentication)</code>	Метод призначений для витягування значення <code>userId</code> з токена запиту.
<code>getAllEvents(Authentication authentication)</code>	Функція, що повертає список всіх подій що належать автентифікованому користувачу.
<code>createEvent(@Valid @RequestBody EventDTO eventDTO, Authentication authentication)</code>	Метод, що приймає дані створення події, мапнить їх в об'єкт класу сутностей, і викликає відповідний метод в сервісі для обробки.
<code>getEventById(@PathVariable Long id, Authentication authentication)</code>	Отримання деталей конкретної події за її <code>id</code> .
<code>updateEvent(@PathVariable Long id, @Valid @RequestBody EventDTO eventDTO, Authentication authentication)</code>	Метод, що приймає оновлені дані та ідентифікатор події, мапнить їх в об'єкт класу сутностей, і викликає відповідний метод в сервісі для обробки.

Продовження таблиці 3.1

Сигнатура	Призначення
<code>deleteEvent(@PathVariable Long id, Authentication authentication)</code>	Отримує id події та викликає відповідний метод в сервісі для видалення події.

Клас `EventServiceImpl` представляє шар бізнес логіки з основними правилами та операціями, пов'язаними з подіями. Він взаємодіє з репозиторіями для доступу до даних та може викликати інші сервіси. Методи класу наведено в табл. 3.2.

Таблиця 3.2 – Методи класу `EventServiceImpl`

Сигнатура	Призначення
<code>findAllEventsForUser(String userId)</code>	Викликає метод в репозиторії для отримання подій відповідного користувача з бази. Повертає список подій.
<code>findEventByIdAndUserId(Long eventId, String userId)</code>	Викликає метод в репозиторії для отримання конкретної події відповідного користувача з бази. Повертає конкретну подію.
<code>createEvent(Event event, String userId, List<Long> participantIds)</code>	Встановлює до події користувача що створив, час створення, та учасників якщо є. Викликає метод в репозиторії для збереження події.
<code>updateEvent(Long eventId, Event updatedEventDetails, String userId, List<Long> participantIds)</code>	Викликає метод в репозиторії для отримання (перевірки існування) конкретної події відповідного користувача з бази. Встановлює усі передані зміни. Викликає метод в репозиторії для збереження події.

Продовження таблиці 3.2

Сигнатура	Призначення
<code>deleteEvent(Long eventId, String userId)</code>	Викликає метод в репозиторії для отримання (перевірки існування) конкретної події відповідного користувача з бази. Викликає метод репозиторію для видалення з бази.

Нагадування організовано за допомогою опитування на стороні сервера. Для цього є відповідний клас `ReminderNotificationServiceImpl` – основний сервіс що відповідає за спрацьовування нагадувань. Він має один метод `checkAndSendReminders()`, що анотований `@Scheduled`. Це означає, що метод запускається кожен раз за розкладом. Він робить запит до `EventRepository`, щоб знайти всі події, для яких `reminderDateTime` настав або вже минув, і для яких нагадування ще не було відправлено. Визначає повідомлення та користувача на основі даних кожної події, та ініціалізує створення нагадувань для кожної. Після цього показник відправленості сповіщення для всіх цих сповіщень встановлюється `true`.

В свою чергу на клієнтській стороні, за допомогою сервісу `notification`, з певною періодичністю (30 секунд) виконується опитування, надсилається запит на сервер для отримання нових нагадувань користувача.

Клас `EmailServiceImpl` відповідає за відправку повідомлень на електронні поштові адреси, які вказано для контактів, яких додано до певної події. Він має один метод `sendNotificationToEmail(String Email, Event event)`, який формує повідомлення та надсилає його на вказану поштову адресу. Це відбувається за допомогою класу `SimpleMailMessage` (для формування повідомлення) та інтерфейсу `JavaMailSender` (для відправки на пошту), які доступні завдяки підключенню залежності `spring-boot-starter-mail`. Поштовий сервер та відправник налаштовуються за допомогою елементів конфігурації з простору імен `spring.mail` у файлі `application.properties` проєкту.

3.4 Огляд структури бази даних

Для роботи з даними та їх зберігання було обрано MySQL, як надійну, зручну, безпечну і широко функціональну систему керування базами даних, дані в якій добре організовані та структуровані.

Основними сутностями в системі є:

- користувачі (users);
- плановані події (events);
- контакти (contacts);
- сповіщення (notifications).

Як вже було зазначено, сервер авторизації та сервер обробки запитів мають кожен свою базу даних, що сприяє незалежності сервісів та підвищенню безпеки. Тож база даних серверу авторизації має лише одну таблицю users, в якій зберігаються електронна пошта та хешований пароль зареєстрованих користувачів. Всі інші сутності належать базі основного серверу.

Поля таблиці users та їх опис представлено в табл. 3.3.

Таблиця 3.3 – Поля таблиці users

Назва	Тип даних	Опис
id	bigint	Порядковий номер користувача в базі.
email	varchar	Поштова адреса, використовується як ідентифікатор при авторизації.
password	varchar	Пароль після BCrypt хешування.
roles	varchar	Роль користувача. Навіть якщо вона одна, потрібно для конфігурацій безпеки та генерації токенів.

Поля таблиці events та їх опис представлено в табл. 3.4.

Таблиця 3.4 – Поля таблиці events

Назва	Тип даних	Опис
id	bigint	Порядковий номер події в базі.
title	varchar	Назва подій (її суть).
description	varchar	Детальний опис.
creationTime	datetime	Дата та час створення.
plannedTime	datetime	Запланована дата і час.
completed	bool	Показник чи виконано подію.
reminderSet	bit	Показник чи встановлено нагадування.
reminderSent	bit	Показник чи відправлено нагадування.
reminderDateTime	datetime	Час нагадування.
reminderOffsetMinutes	int	Зсув від планованого часу події.
userId	varchar	Користувач (власник) події.

Поля таблиці contacts та їх опис представлено в табл. 3.5.

Таблиця 3.5 – Поля таблиці contacts

Назва	Тип даних	Опис
id	bigint	Порядковий номер контакту в базі.
name	varchar	Ім'я контактної особи.
notes	varchar	Детальний опис.
email	varchar	Пошта для повідомлення.
userId	varchar	Користувач (власник) контакту.

Поля таблиці notifications та їх опис представлено в табл. 3.6.

Таблиця 3.6 – Поля таблиці notifications

Назва	Тип даних	Опис
id	bigint	Порядковий сповіщення у в базі.
message	varchar	Ім'я контактної особи.
timestamp	datetime	Час надсилання сповіщення.
readstatus	bit	Статус чи прочитано сповіщення користувачем.
eventId	bigint	Ідентифікатор пов'язаної події.
userId	varchar	Користувач (власник) контакту.

Структуру створених баз даних показано на рис. 3.7

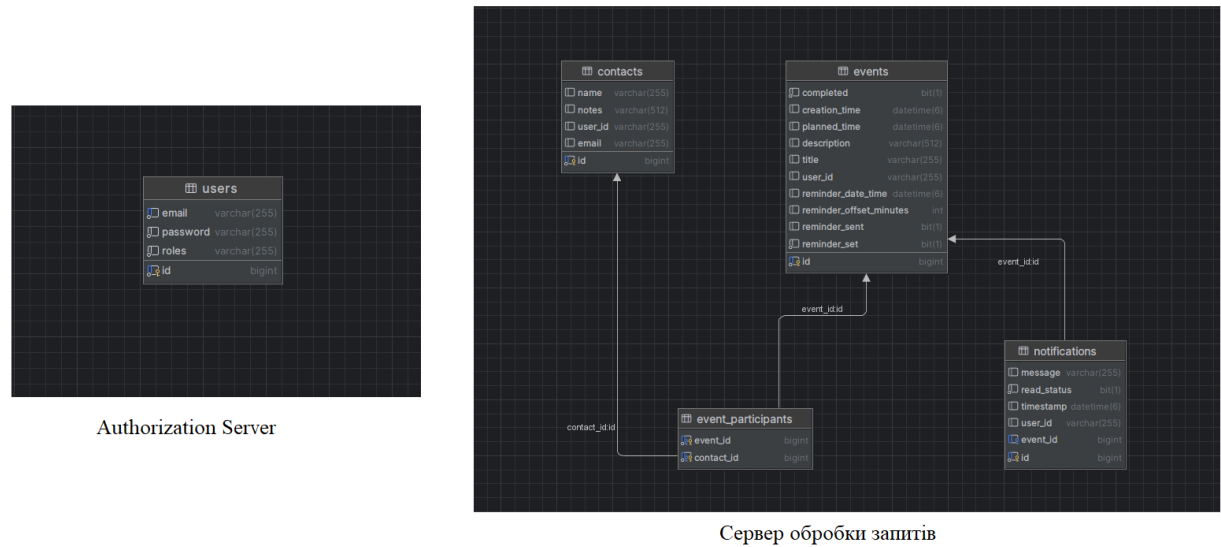


Рисунок 3.7 – Структура баз даних

3.5 Основні рішення щодо проєктування інтерфейсу програмного забезпечення

Інтерфейс даного програмного забезпечення спроектовано з урахуванням функціональних вимог і вимог до інтерфейсу з технічного завдання. Він є досить простим та зрозумілим. Користувач може легко орієнтуватись в застосунку та використовувати всі функції.

При розробці інтерфейсу було використано такі елементи як: кнопки, поля, списки, чекбокси, календар, іконки, блочні контейнери, рядкові контейнери, інтерактивні форми та діалогові вікна.

Основним шрифтом застосунку є Montserrat з GoogleFonts. Він є досить популярним у веброзробці, виглядає достатньо сучасно та забезпечує добру читабельність.

Повідомлення користувачу відображаються інтерактивно, в залежності від його дій. Наприклад помилки при введені невалідних або надмірних даних у певні поля. А нагадування щодо подій відображаються асинхронно і не потребують ручного перезавантаження сторінки.

Загалом інтерфейс розробленого вебзастосунку для організації часу планування подій складається з наступних компонентів:

- компонент з вибором увійти чи авторизуватись;
- сторінка реєстрації;
- сторінка авторизації;
- головна сторінка з потижневими списками запланованих подій;
- форма створення та редагування події;
- компонент зі списком контактів;
- компонент із повним списком всіх подій;
- форма створення та редагування контактів;
- компонент підтвердження видалення контакту;
- форма виходу.

Для створення і редагування, як подій так і контактів, використовуються одні й ті ж форми відповідно. Але в залежності від опції (чи створюється новий об'єкт, чи редагується існуючий) вони трохи видозмінюються.

Інтерфейс сторінки вибору входу показано на рис. 3.8.

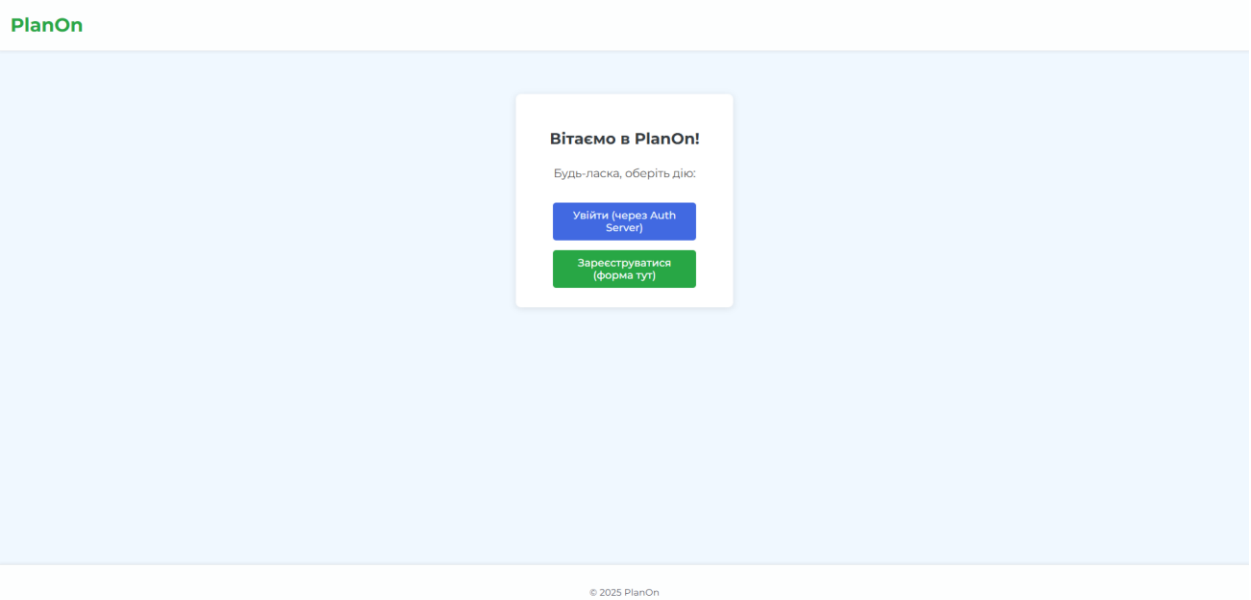


Рисунок 3.8 – Інтерфейс сторінки вибору входу

Інтерфейс сторінки реєстрації показано на рис. 3.9.

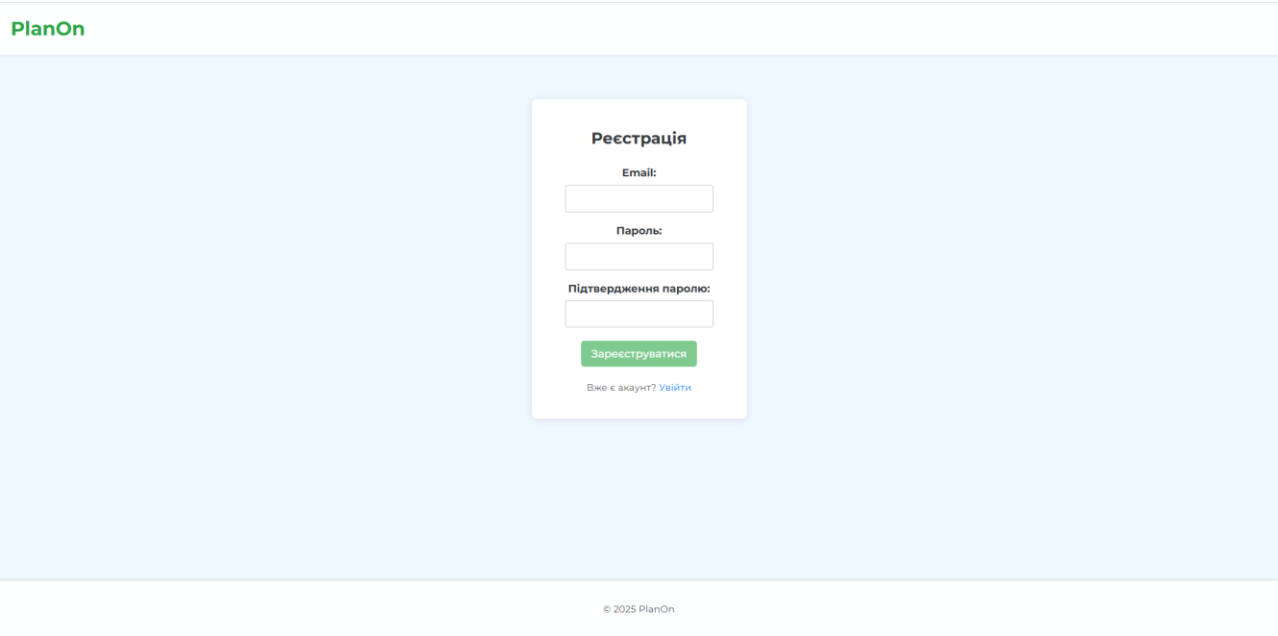


Рисунок 3.9 – Інтерфейс сторінки реєстрації

Інтерфейс сторінки авторизації показано на рис. 3.10.

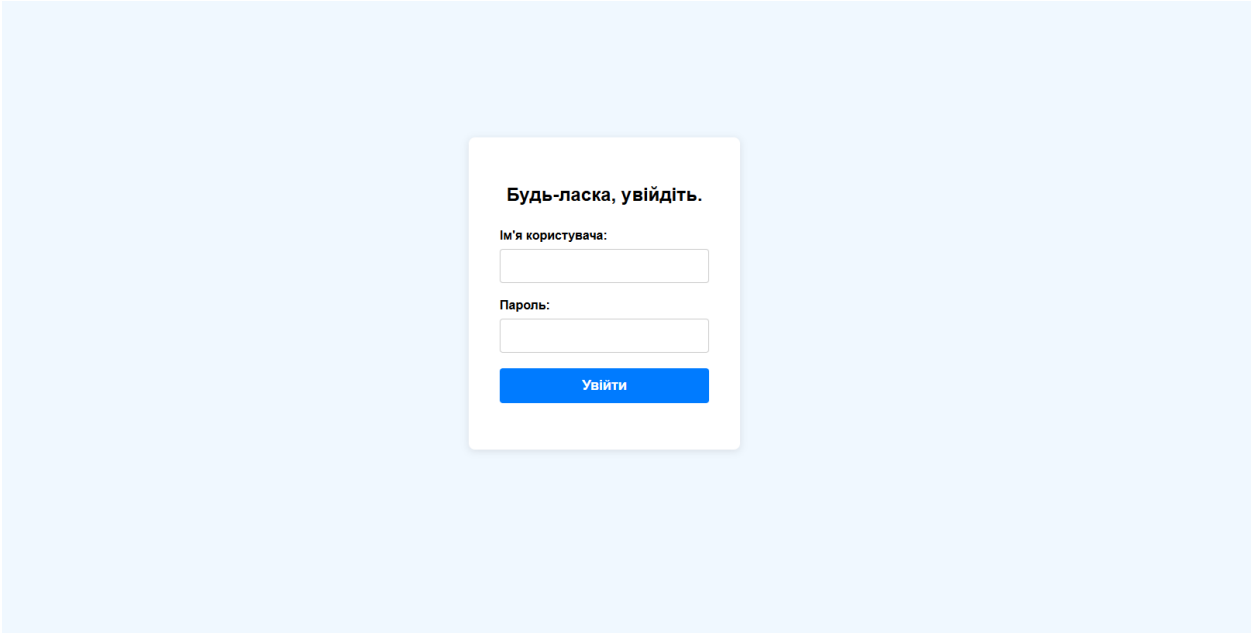


Рисунок 3.10 – Інтерфейс сторінки авторизації

Інтерфейс головної сторінки з потижневими списками запланованих подій показано на рис. 3.11.

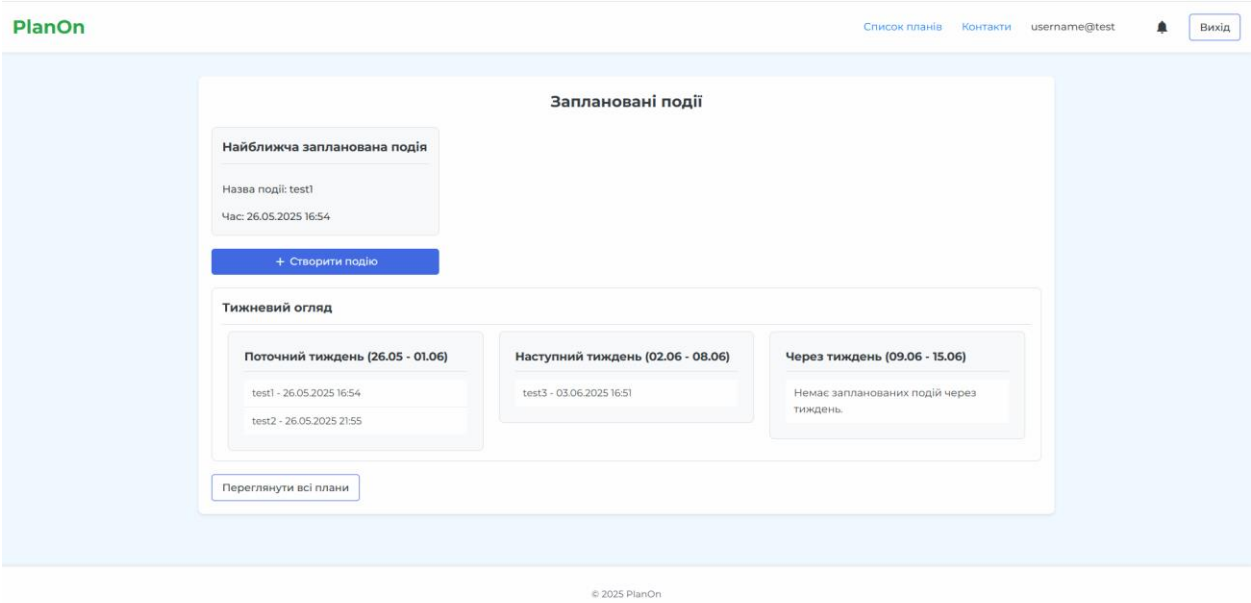


Рисунок 3.11 – Інтерфейс головної сторінки

Інтерфейс форми створення та редагування події на рис. 3.12.

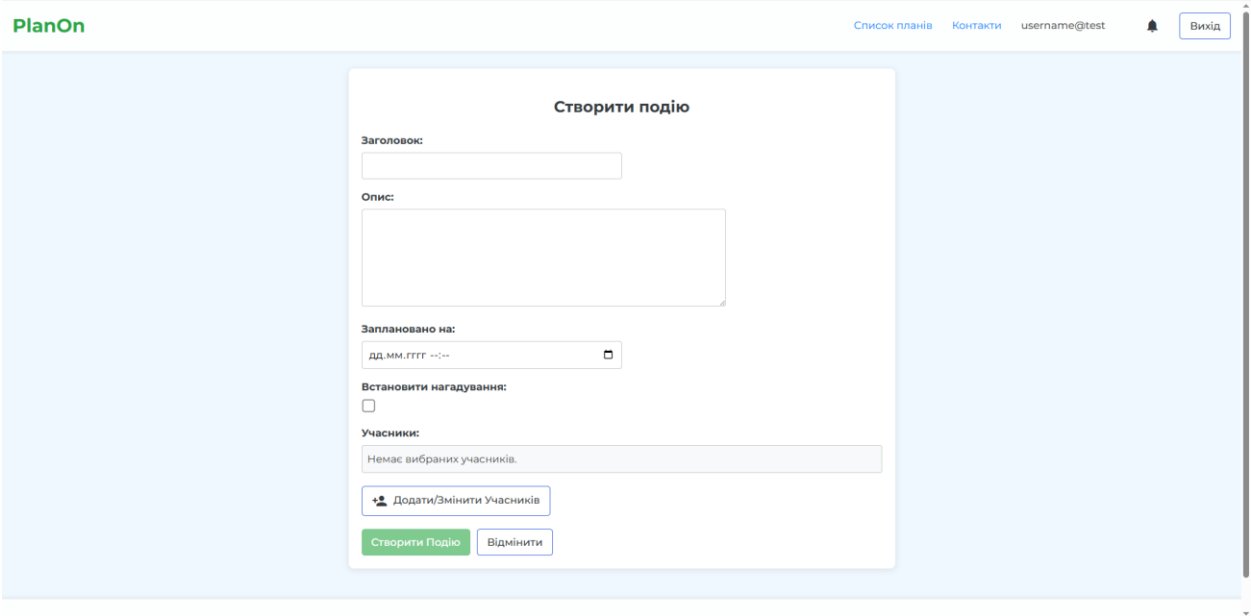


Рисунок 3.12 – Інтерфейс форми створення або редагування події

Інтерфейс сторінки зі списком контактів показано на рис. 3.13.

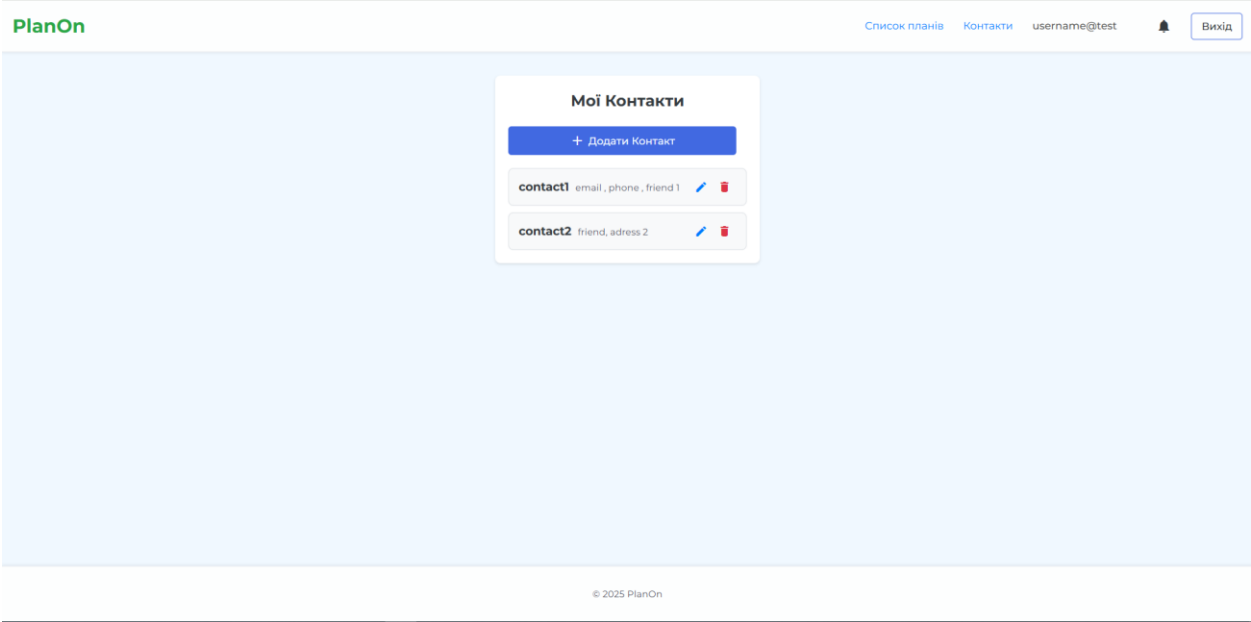


Рисунок 3.13 – Інтерфейс сторінки зі списком контактів

Інтерфейс сторінки з повним списком всіх запланованих подій показано на рис. 3.14.

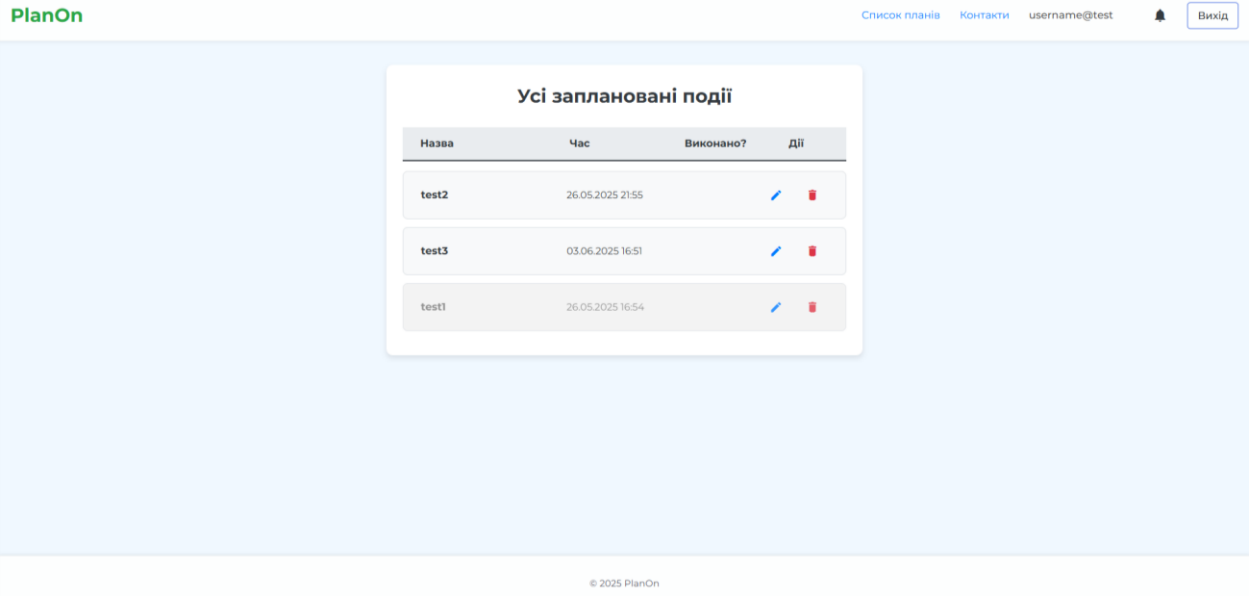


Рисунок 3.14 – Інтерфейс сторінки з повним списком всіх запланованих подій

Більшість форм реалізовано як модальні, за допомогою сервісу mat-dialog для Angular. Приклади таких наведено нижче.

Інтерфейс модальної форми створення та редагування контактів показано на рис. 3.15.

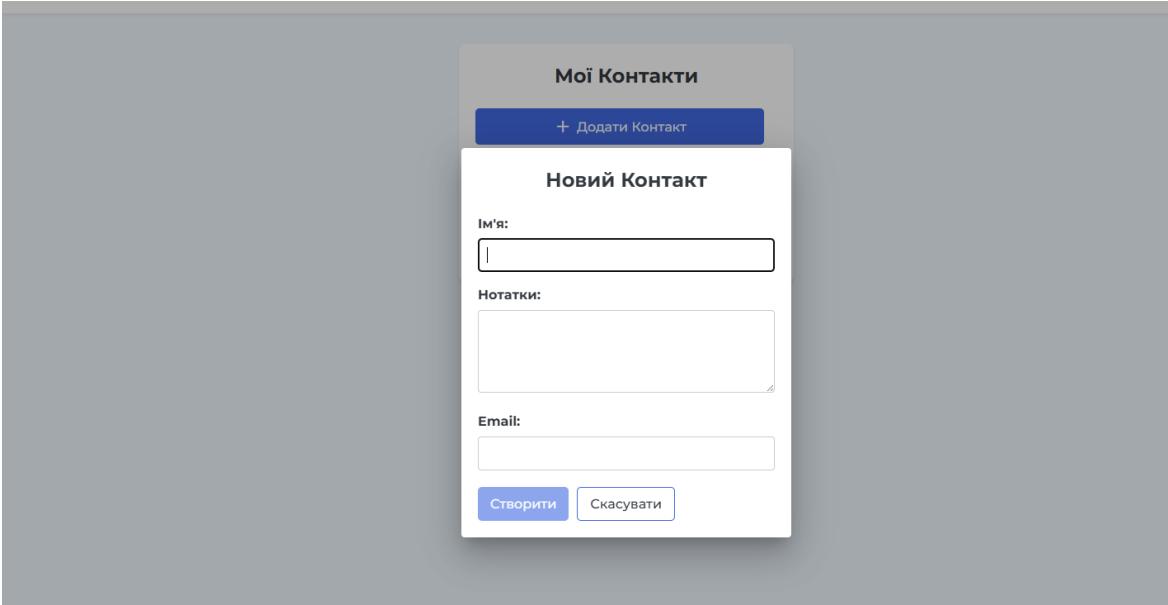


Рисунок 3.15 – Інтерфейс модальної форми створення та редагування контактів

Інтерфейс модальної форми підтвердження видалення контакту показано на рис. 3.16.

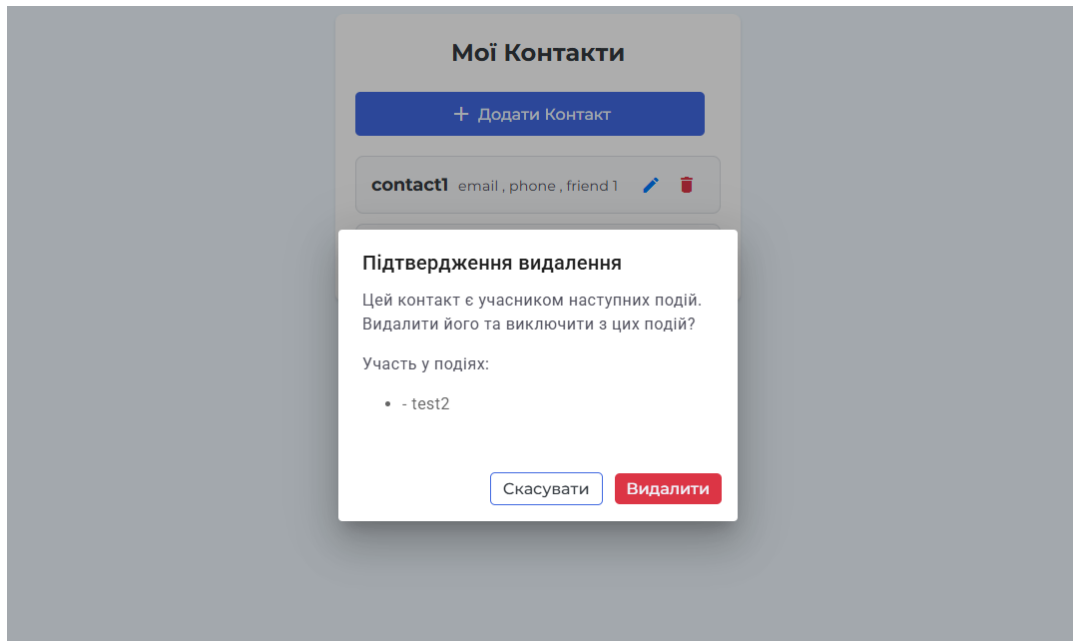


Рисунок 3.16 – Інтерфейс модальної форми підтвердження видалення контакту

Інтерфейс модальної форми виходу показано на рис. 3.17.

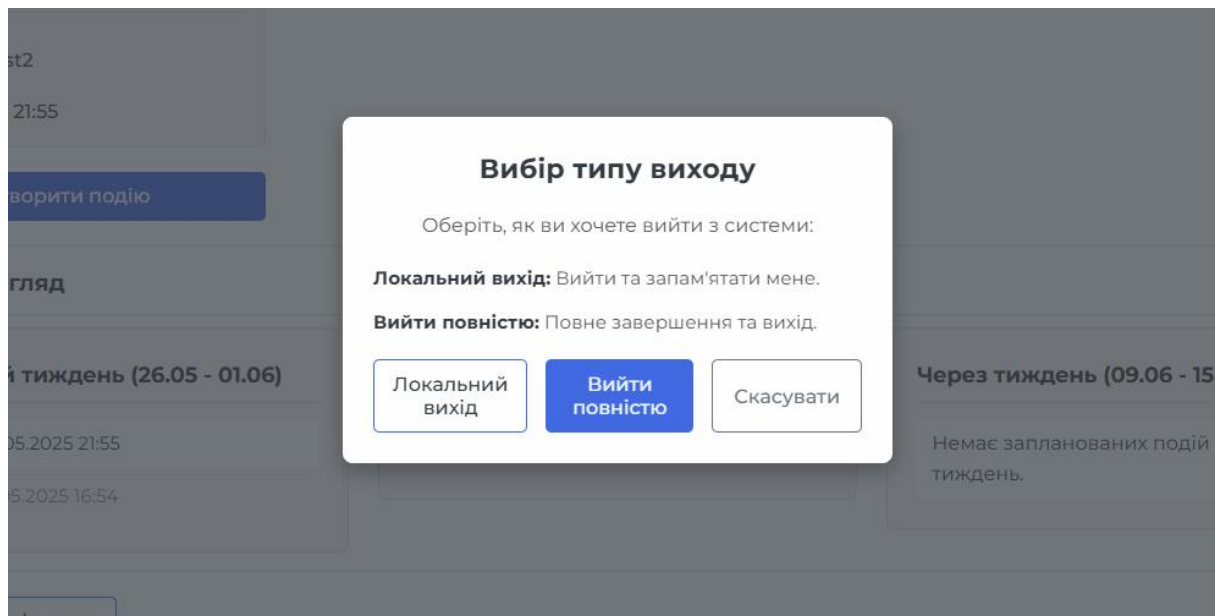


Рисунок 3.17 – Інтерфейс модальної форми виходу

3.6 Висновки до розділу 3

У цьому розділі було розглянуто архітектуру та структуру розробленого вебзастосунку для планування подій та організації часу. Тут приділено увагу деталям розробки та описано основні рішення щодо головних компонентів системи.

Клієнтську частину реалізовано за сучасним підходом Single Page Application, а серверну розділено на два основні сервіси, побудовані на основі фреймворку Spring Boot. Це забезпечує чіткий розподіл відповідальності, що підвищує масштабованість та зручність супроводу системи.

Показано основні функціональні можливості та варіанти взаємодії кінцевого користувача з системою. Побудовано наочні та зрозумілі UML-діаграми. В системі успішно реалізовано основний функціонал проекту: CRUD операції з подіями та контактами, гнучкий механізм встановлення нагадувань, пов'язування учасників (контактів) з подіями та можливість повідомити їх про це за допомогою поштового повідомлення.

Детально описано розроблену базу даних, що являє собою реляційну MySQL. Вона структуровано зберігає дані проекту у відповідних таблицях. Показано опис і відношення цих таблиць.

Інтерфейс застосунку орієнтовано на зручні умови роботи для користувача. Для окремих функціональних блоків використано окремі компоненти. Валідація на формах відбувається відповідно до внесених змін користувача, система попереджає користувача про помилки, а усі неоднозначні дії (видалення, вихід) потребують підтверджень користувача, що мінімізує помилкове користування та покращує користувацький досвід.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Розроблена програма призначена для надання користувачам зручного інструменту для планування особистих подій. Написана в основному на мовах Java (серверна частина) та TypeScript (клієнтська частина). У якості основного фреймворку для серверної частини було обрано Spring Boot, що є потужним і надійним рішенням для швидкого налаштування і створення RESTful API. Окрім цього він надає зручні можливості для роботи з базами даних, особливо MySQL, яку і було використано в якості надійного сховища даних для обох серверних частин. Для клієнтської частини використано фреймворк Angular, який забезпечує стійку типізацію та зручний компонентний підхід для створення SPA застосунків.

Програма орієнтована на дуже широке коло користувачів, оскільки як було визначено під час детального аналізу предметної області, планування особистих справ сприяє налагодженню темпу життя, зменшенню кількості забутих планів та підвищенню власної продуктивності. Це є досить важливим майже для будь-якої людини.

Програмне забезпечення для організації часу та планування подій призначене для виконання наступних функцій:

- внесення планів на певну дату і час;
- додавання подій;
- редагування та видалення подій;
- додавання контактів;
- редагування та видалення контактів;
- додавання нотаток до події;
- можливість вказати пов'язаних учасників та надіслати їм повідомлення;

- налаштування сповіщень про майбутні події, зустрічі що скоро настануть;
- реєстрація та авторизація користувачів.

Для функціонування вебзастосунку, необхідне виконання наступних вимог до системи та програмних засобів:

- ОС основана на Unix або Windows;
- процесор Intel core i5-2500;
- монітор з роздільною здатністю 1920x1080;
- ОЗП ємністю 4 гб або більше;
- Щонайменше 2 Гб;
- браузер останньої версії на рушії Chromium (Google Chrome), Gecko (Mozilla Firefox) або інші (Opera, Microsoft Edge), які підтримують сучасні стандарти HTML, CSS та JS.

4.2 Розгортання проекту

Для розгортання системи користувачу потрібно мати на своєму персональному комп'ютері встановлену платформу Node.js 22.11.0 та менеджер пакетів npm 10.9.0. Це можна робити з офіційного сайту Node.js. Angular після цього можна встановити командою `npm install -g @angular/cli`. Для роботи бекенд частин необхідно мати встановлену JVM 21 версії. Це можна зробити встановивши Java Development Kit 21, з офіційної сторінки Oracle. Також потрібна СКБД MySQL, та будь який сервіс для запуску вебсерверів, наприклад XAMPP, інструмент MySQL Workbench тощо.

Для запуску серверної частини потрібно щоб були вільні порти 8080 (для основного серверу), 9000 (для серверу авторизації) та 3306 (для роботи баз даних). Запуск бекенд сервісів можна здійснити за допомогою відповідних .jar файлів. А локальний сервер баз даних запустити відповідно до обраного інструменту.

Клієнська частина запускається в кореневій директорії проекту з консольного рядка, за допомогою команди `ng serve`. Потрібно щоб був вільним порт 4200.

Або альтернативний підхід розгортання системи. Користувачу потрібно мати на своєму комп'ютері програмне забезпечення для розгортання і управління контейнерами Docker.

Щоб почати роботу користувачу потрібно завантажити образи модулів системи з та образ СКБД MySQL з репозиторію Docker Hub. Для цього потрібно використати команду `docker pull <ім'я образу>`.

Після цього потрібно завантажити теку з конфігураціями з GitHub репозиторію. Потім в директорії цієї теки запустити консоль і виконати команду `docker compose up --build`. З образів буде створено контейнери з усіма необхідними для роботи налаштуваннями.

Для початку експлуатації програми потрібно перейти за посиланням <http://127.0.0.1:4200/>.

4.3 Вхідні та вихідні дані

Вхідними даними при реєстрації та авторизації є логін та пароль користувача. При роботі з основним функціоналом вхідними даними є назви подій, опис подій, обрані заплановані дати та час, встановлення нагадування, час зсуву нагадування, статус завершеності справи, пов'язані контакти, ім'я контакту, нотатки до контакту, контактна електронна пошта.

Вихідними даними застосунку є відповідні зміни в системі та інтерфейсі користувача, оновлення даних в списках, полях, переадресація сторінок тощо.

4.4 Методика та результати тестування

Тестування проводилося за методикою ручного тестування. Цей вид тестування є досить простим, полягає у активній взаємодії з програмним продуктом – так як це б робив потенційний користувач. Перевіряється функціональність, зручність використання.

Тестування відбувалося у трьох браузерах: Google Chrome 136.0.7103.93, Mozilla Firefox 138.0.4 та Microsoft Edge 136.0.3240.92.

Протестовано основні сценарії функціонування вебзастосунку. Деякі з них описано та показано нижче.

Реєстрація з валідними даними. Користувач вводить свою пошту, пароль, та повторює пароль в полі для підтвердження.

Дані. Email: «testuser11@test.com», пароль: «testpass33», повторний пароль «testpass33».

Очікуваний результат: користувача успішно створено, переадресація на сторінку входу.

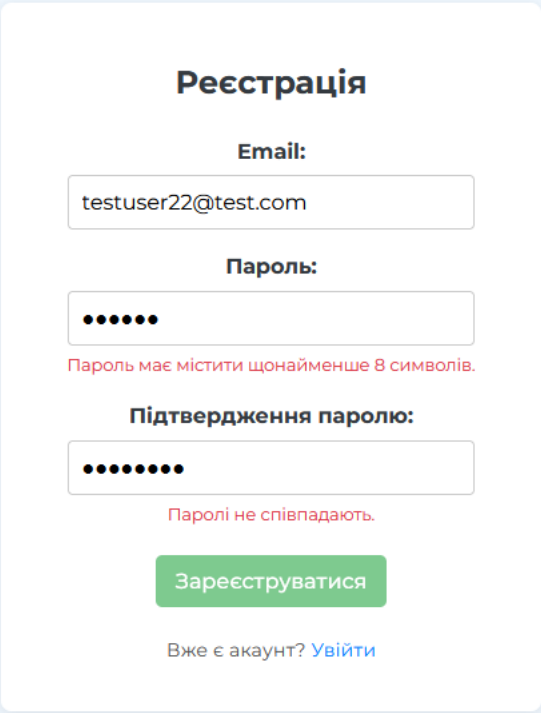
Фактичний результат: користувача успішно створено, переадресація на сторінку входу.

Реєстрація з невалідними даними. Користувач вводить свою пошту, невалідний пароль, та повторює пароль з помилками в полі для підтвердження.

Дані. Email: «testuser22@test.com», пароль: «testpw», повторний пароль «testpass».

Очікуваний результат: користувачу повідомлено про помилки для кожного поля, можливість зареєструватися з такими даними неможлива.

Фактичний результат: користувачу повідомлено про помилки для кожного поля, можливість зареєструватися з такими даними неможлива. Результат показано на рис. 4.1.



The image shows a registration form titled "Реєстрація" (Registration) on a light blue background. The form is white with rounded corners and contains the following elements:

- Email:** A text input field containing "testuser22@test.com".
- Пароль:** A password input field with 7 dots. Below it, a red error message reads: "Пароль має містити щонайменше 8 символів." (Password must contain at least 8 symbols).
- Підтвердження паролю:** A confirmation password input field with 7 dots. Below it, a red error message reads: "Паролі не співпадають." (Passwords do not match).
- Зареєструватися**: A green button with white text.
- Вже є акаунт? Увійти**: A link in blue text.

Рисунок 4.1 – Реєстрація з невалідними даними

Створення події з валідними даними. Користувач переходить на форму створення події. Заповнює коректно поля заголовков, опис, встановлює запланований час майбутній від поточного. Натискає кнопку створити.

Дані. Заголовок: «testtitle», Опис: «testdescription», Заплановано на: «27.05.2025 20:00» (поточний час: 27.05.2025 15:55).

Очікуваний результат: подію успішно створено, переадресація на головну сторінку з потижневими списками запланованих подій.

Фактичний результат: подію успішно створено, переадресація на головну сторінку з потижневими списками запланованих подій.

Створення події з невалідними даними. Користувач переходить на форму створення події. Не заповнює поле заголовку, встановлює запланований час минулий від поточного. Натискає кнопку створити.

Дані. Заголовок: відсутній, Опис: «testdescription», Заплановано на: «26.05.2025 20:00» (поточний час: 27.05.2025 16:00).

Очікуваний результат: користувачу повідомлено про неможливість залишити поле заголовку пустим а дату обрати раніше від поточної.

Фактичний результат: користувачу повідомлено про неможливість залишити поле заголовку пустим а дату обрати раніше від поточної. Результат показано на рис. 4.2

The screenshot displays a web form titled "Створити подію" (Create Event). The form contains several input fields and a checkbox, each with associated validation messages in red text:

- Заголовок:** (Title) - An empty text input field with the error message "Заголовок обов'язковий." (Title is required.)
- Опис:** (Description) - A text area containing "testdescription" with a red squiggly underline indicating a validation error.
- Заплановано на:** (Scheduled for) - A date and time picker showing "26.05.2025, 20:00" with a calendar icon. The error message "Планований час не може бути в минулому." (Scheduled time cannot be in the past.) is displayed below.
- Встановити нагадування:** (Set reminder) - An unchecked checkbox.
- Учасники:** (Participants) - A list box showing "Немає вибраних учасників." (No participants selected.)

At the bottom of the form, there is a button "+ Додати/Змінити Учасників" (Add/Change Participants), a green "Створити Подію" (Create Event) button, and a "Відмінити" (Cancel) button.

Рисунок 4.2 – Створення події з невалідними даними

Отримання нагадування щодо запланованого повідомлення. Користувач переходить на форму створення події. Заповнює коректно поля заголовок, опис, встановлює запланований час майбутній від поточного.

Встановлює чекбокс для нагадування. Вводить за скільки хвилин до події нагадати. Натискає кнопку створити.

Дані. Заголовок: «testtitle», Опис: «testdescription», Заплановано на: «27.05.2025 16:35» (поточний час: 27.05.2025 16:20), відмічений чекбокс «Встановити нагадування», Нагадати за (хвилин до події): 5.

Очікуваний результат: користувачу о 16:30 надійшло повідомлення про подію, що скоро настане.

Фактичний результат: користувачу о 16:30 надійшло повідомлення про подію, що скоро настане. Результат показано на рис. 4.3

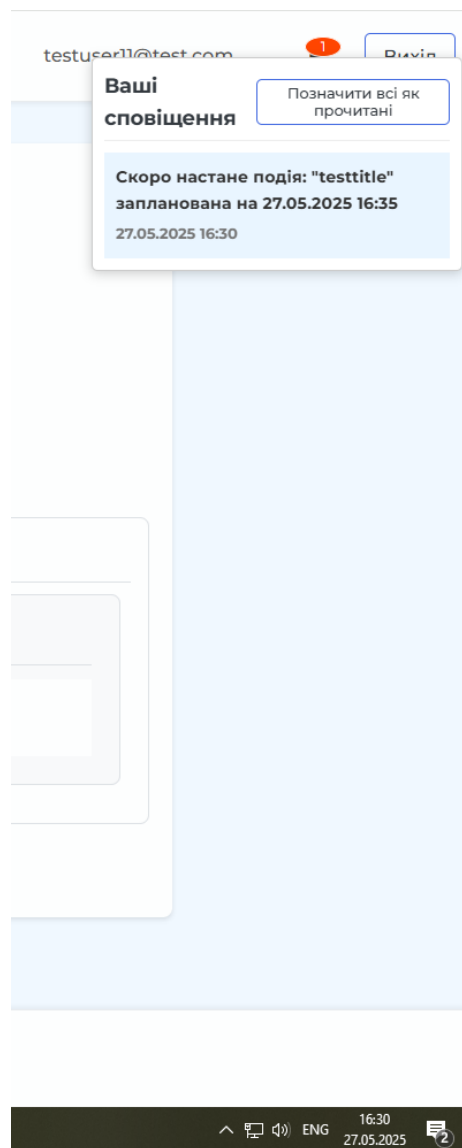


Рисунок 4.3 – Отримання нагадування щодо запланованої події

Під час проведення тестування було сформовано відповідний чекліст, який продемонстровано у табл. 4.1.

Таблиця 4.1 – Чекліст тестування

Функціонал	Google Chrome 136.0.7103.93	Mozilla Firefox 138.0.4	Microsoft Edge 136.0.3240.92
Реєстрація користувача	Пройшов	Пройшов	Пройшов
Авторизація користувача	Пройшов	Пройшов	Пройшов
Перегляд списку запланованих справ на найближчі 3 тижні	Пройшов	Пройшов	Пройшов
Перегляд повного списку усіх запланованих справ	Пройшов	Пройшов	Пройшов
Створення події	Пройшов	Пройшов	Пройшов
Редагування події	Пройшов	Пройшов	Пройшов
Видалення події	Пройшов	Пройшов	Пройшов
Встановлення нагадування	Пройшов	Пройшов	Пройшов
Отримання нагадування	Пройшов	Пройшов	Пройшов
Створення контакту	Пройшов	Пройшов	Пройшов
Редагування контакту	Пройшов	Пройшов	Пройшов
Видалення непов'язаного контакту	Пройшов	Пройшов	Пройшов
Видалення контакту пов'язаного з подіями	Пройшов	Пройшов	Пройшов
Відправка повідомлення на пошту контакту, якого додано до події	Пройшов	Пройшов	Пройшов

За результатами детального і уважного тестування можна впевнено сказати, що розроблений за допомогою сучасних підходів і інструментів, вебзастосунок для організації часу та планування подій працює злагоджено, стабільно і безпечно, а головне підтримує усі визначені функціональні вимоги.

Проведені дослідження показують, що система справна готова до розгортання та реальної експлуатації користувачами.

4.5 Висновки до розділу 4

У даному розділі описано призначення розробленого програмного забезпечення, його основний функціонал. Визначено основні умови необхідні для запуску та експлуатації продукту. Детально розглянуто варіанти процесів встановлення та розгортання системи.

Проведено ретельне тестування вебзастосунку у різних браузерях. Описано різні сценарії взаємодії користувача та відповіді системи на них.

Результати тестування показали високий рівень працездатності та зручності програмного продукту та готовність до справжнього використання.

Розроблена система готова для подальшого розширення та впровадження нових функцій.

ВИСНОВКИ

В ході виконання кваліфікаційної дипломної роботи бакалавра було детально проаналізовано та досліджено процес розробки програмного забезпечення для організації часу та планування подій.

Під час ґрунтовного розгляду предметної області було визначено, що необхідність використання подібних інструментів є постійно зростаючою в сучасному світі, кількість інформації в якому безупинно збільшується, а темп життя стає все більш навантаженим. Застосування цифрових рішень для планування повсякденного життя дозволяє розподілити час якомога зручним способом і уникнути пропусків важливих подій, забуття побутових справ і втраті корисних контактів.

Аналогічне програмне забезпечення є достатньо варіативним. Іноді воно занадто перевантажене функціоналом, а подекуди має не зовсім зручний інтерфейс, який може неправильно акцентувати увагу користувача на різних елементах. Тож визначено що розробка програмного забезпечення для організації часу та планування завдань є актуальним напрямком, і сформульовано відповідні функціональні та інтерфейсні вимоги.

Визначено мови програмування для розробки вебзастосунку. Java для бекенд, що має широкі можливості для створення серверних застосунків та підтримується дуже потужними і водночас зручними фреймворками. Для фронтенд – TypeScript, вдосконалений JavaScript зі статичною типізацією, підтримкою авторефакторингу і ООП.

В якості середовища розробки для даного програмного забезпечення обрано сучасну та швидку IntelliJ IDEA, яка має якісну підтримку обраних мов та велику кількість інструментів та інтеграції для продуктивної розробки.

Визначено архітектуру вебзастосунку, основною особливістю якої є чітке розподілення обов'язків для кожної частини системи. Клієнтська частина – взаємодія з користувачем, відображення змін відправлення запитів. Серверна частина авторизації – обробка даних користувачів, відправка та

перевірка JWT доступу до ресурсів. Серверна частина RESTful API – обробка запитів для роботи з основним функціоналом програми, здійснення CRUD операцій та взаємодія з відповідною базою даних. Такий архітектурний підхід сприяє зручному супроводу і вдосконаленню застосунку, достатньому рівню безпеки та масштабованості.

Для зберігання даних використовується дві бази даних (для зберігання даних користувачів та їх ресурсів відповідно) MySQL, яка забезпечує надійність зберігання даних та практичний доступ до них. Спроектовано структуру бази даних, що зрозуміло відображає сутності системи та відношення між ними.

Зображено та описано функціонування програмного забезпечення для організації часу та планування подій, що включає в себе роботу з подіями, встановлення нагадувань та роботу з контактами.

Спроектовано та реалізовано зручний та адаптивний інтерфейс взаємодії користувача із системою, який забезпечує приємний користувацький досвід.

Проведено ретельне тестування розробленого програмного забезпечення для організації часу та планування подій, що продемонструвало повну справність програмного продукту і готовність до експлуатації кінцевими користувачами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Тайм-менеджмент: 5 порад для тих, хто нічого не встигає [Електрон. ресурс]. – Режим доступу: <https://happymonday.ua/jeffektivnyj-tajm-menedzhment-5-sovetov>.
2. Чому керівники бізнесу не можуть ігнорувати автоматизацію тайм-менеджменту у 2025 році [Електрон. ресурс]. – Режим доступу: <https://yaware.com.ua/uk/blog/chomu-kerivniki-biznesu-ne-mozhut-ignoruvati-avtomatizacziyu-tajm-menedzhmentu-u-2025-rocz/>.
3. How to Develop a Time Management App? [Electronic resource]. – Access mode: <https://www.matellio.com/blog/time-management-app-development>.
4. Google Tasks vs. Todoist [Electronic resource]. – Access mode: <https://www.usemotion.com/blog/google-tasks-vs-todoist>.
5. Докладніше про Google Завдання [Електрон. ресурс]. – Режим доступу: <https://support.google.com/tasks/answer/7675772>.
6. Google Tasks [Electronic resource]. – Access mode: <https://tasks.google.com/tasks/>.
7. Google Keep Review [Electronic resource]. – Access mode: <https://www.pcmag.com/reviews/google-keep-web>.
8. Google Keep [Electronic resource]. – Access mode: <https://keep.google.com/>.
9. Any.do [Electronic resource]. – Access mode: <https://www.techradar.com/reviews/anydo>.
10. Any.do Review: Features, Pricing, Pros, and Cons [Electronic resource]. – Access mode: <https://www.techrepublic.com/article/any-do-review/>.
11. Any.do [Electronic resource]. – Access mode: <https://app.any.do/tasks/all>.
12. TickTick Review do [Electronic resource]. – Access mode: <https://www.pcmag.com/reviews/ticktick>

13. TickTick [Electronic resource]. – Access mode: <https://ticktick.com/webapp>.
14. Top 11 Backend Programming Languages in 2025 [Electronic resource]. – Access mode: <https://webandcrafts.com/blog/backend-languages>.
15. Розробка з боку Front end – що це таке і чим відрізняється від Back end? [Електрон. ресурс]. – Режим доступу: <https://dan-it.com.ua/uk/blog/rozrobka-z-boku-front-end-shho-ce-take-i-chim-vidriznjaetsja-vid-back-end/>.
16. Що таке Java і де вона використовується [Електрон. ресурс]. – Режим доступу: <https://goit.global.ua/articles/shcho-take-java-i-de-vona-vykorystovuietsia/>
17. Java Tutorial [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/java/>.
18. Що таке Typescript і навіщо він потрібен [Електрон. ресурс]. – Режим доступу: <https://foxminded.ua/typescript/>.
19. TypeScript for the New Programmer [Electronic resource]. – Access mode: <https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html>.
20. What is TypeScript? Definition, History, Features and Uses [Electronic resource]. – Access mode: <https://invedus.com/blog/what-is-typescript-definition-history-features-and-uses-of-typescript/>.
21. Why Spring? [Electronic resource]. – Access mode: <https://spring.io/why-spring>.
22. Вступ до інверсії керування та впровадження залежностей за допомогою Spring [Електрон. ресурс]. – Режим доступу: <https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring>.
23. Як Spring Boot допомагає прискорити розробку додатків [Електрон. ресурс]. – Режим доступу: <https://foxminded.ua/spring-boot/>.
24. Spring Boot [Electronic resource]. – Access mode: <https://spring.io/projects/spring-boot>.

25. Enable or Disable Embedded Tomcat with Profile in Spring Boot [Electronic resource]. – Access mode: <https://www.baeldung.com/spring-boot-enable-disable-embedded-tomcat>.
26. Spring Initializr [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/spring-initializr/>.
27. What is Angular? [Electronic resource]. – Access mode: <https://angular.dev/overview>.
28. Angular проти React [Electronic resource]. – Access mode: <https://foxminded.ua/angular-vs-react>.
29. Anatomy of a component [Electronic resource]. – Access mode: <https://angular.dev/guide/components>.
30. The Angular CLI [Electronic resource]. – Access mode: <https://angular.dev/tools/cli>.
31. Перші кроки в Angular: усе, що потрібно знати розробнику-початківцю [Електрон. ресурс]. – Режим доступу: <https://foxminded.ua/angular/>.
32. The IDE for Professional Development [Electronic resource]. – Access mode: <https://www.jetbrains.com/idea/>.
33. The IDE for Spring developers [Electronic resource]. – Access mode: <https://www.jetbrains.com/idea/spring/>.
34. MySQL: Understanding What It Is and How It's Used [Electronic resource]. – Access mode: <https://www.oracle.com/ua/mysql/what-is-mysql/>.

ДОДАТОК А
Технічне завдання

Вступ

Назва програми – онлайн-планер «PlanOn». Розроблювана система призначена для зручного планування подій та нотування важливої інформації, що збільшить ефективність розподілу часу в житті користувачів.

A.1 Підстава для розробки

Підставою для розробки слугує завдання на дипломну кваліфікаційну роботу на тему «Розробка вебзастосунку для організації часу та планування подій» затверджену наказом Національного університету «Запорізька політехніка» №209 від 28.04.25 року.

A.2 Призначення розробки

Основною метою є допомога користувачам у плануванні повсякденних справ, організації особистого та робочого часу, створенні подій, нагадувань і зустрічей, а також зберіганні та керуванні контактами.

Система спрямована на підвищення особистої ефективності, покращення самоорганізації за рахунок зручного інтерфейсу та зрозумілого функціоналу.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Основними функціональними вимогами до проєкту є:

- внесення планів на певну дату і час;
- можливість додавання подій;
- можливість видалення подій;

- можливість додавання контактів;
- можливість видалення контактів;
- можливість вказати учасників та додати нотатки до події;
- налаштування push-повідомлень про майбутні події, зустрічі що скоро настануть;
- реєстрація користувачів.

A.3.2 Вимоги до інтерфейсу

Основним принципом розробки інтерфейсу має простота та інтуїтивність, щоб користувач легко орієнтувався без додаткових підказок.

Основні функції (календар, події, нагадування, контакти) повинні бути швидко доступними з головного меню. Також варто продумати зручність введення даних: форми для створення та редагування подій, зустрічей і контактів мають бути максимально простими й зрозумілими.

Також потрібно досягти візуальної чіткості за рахунок читабельних шрифтів, гарно підібраних кольорів і зрозумілих піктограм. Дотримання мінімалізму допоможе уникнути перевантаження інтерфейсу зайвими елементами.

За замовчуванням інтерфейс повинен підтримувати українську мову.

A.3.3 Вимоги до зберігання даних

Інформація про користувачів та події що кожен з них створює, та їх контакти зберігаються у відповідній базі даних, що керується СУБД MySQL.

Безпека даних досягається за рахунок надійного процесу авторизації та автентифікації, а також кодування токенів та паролей.

A.3.4 Вимоги до складу та параметрів технічних засобів

Мінімальні вимоги до системи та програмних засобів, необхідні для функціонування вебзастосунку:

- будь-яка ОС основана на Unix або Windows;
- процесор Intel core i3-2500;
- монітор з роздільною здатністю 1920x1080;
- ОЗП ємністю 4 гб;
- встановлений браузер (веб-оглядач) останньої версії на рушії Chromium (Google Chrome), Gecko (Mozilla Firefox) або інших (Opera, Microsoft Edge), що підтримують сучасні стандарти HTML, CSS та JS.
- підключення до мережі Інтернет.

ДОДАТОК Б
Текст програми

Б.1 Серверна частина

Б.1.1 Код класу контролера для операцій над подіями

```

@RestController
@RequestMapping("/api/v1/events")
@PreAuthorize("isAuthenticated()")
@AllArgsConstructor
public class EventController {

    private final EventService eventService;
    private final EventMapper eventMapper;

    // Функція витягування userId з JWT
    private String getUserIdFromAuthentication(Authentication authentication) {
        if (authentication != null && authentication.getPrincipal() instanceof Jwt) {
            Jwt token = (Jwt) authentication.getPrincipal();
            return token.getSubject();
        }
        throw new IllegalStateException("User not authenticated or principal isn't JWT");
    }

    @GetMapping("/")
    public List<EventDTO> getAllEvents(Authentication authentication) {
        String userId = getUserIdFromAuthentication(authentication);
        List<Event> events = eventService.findAllEventsForUser(userId);
        return eventMapper.toDtoList(events); //Конвертація списку Entity в список DTO
    }

    @PostMapping("event")
    @ResponseStatus(HttpStatus.CREATED)
    public EventDTO createEvent(@Valid @RequestBody EventDTO eventDTO,
        Authentication authentication) {
        String userId = getUserIdFromAuthentication(authentication);
        List<Long> participantIds = null;
        if (eventDTO.getParticipantIds() != null) {
            participantIds = eventDTO.getParticipantIds();
        }
    }

```

```

Event event = eventMapper.toEntity(eventDTO); //Мапп DTO в Entity

Event createdEvent = eventService.createEvent(event, userId, participantIds);

return eventMapper.toDto(createdEvent); //Мапп Entity назад DTO
}

```

```

@GetMapping("/{id}")
public ResponseEntity<EventDTO> getEventById(@PathVariable Long id,
Authentication authentication) {
    String userId = getUserIdFromAuthentication(authentication);

    Optional<Event> event = eventService.findEventByIdAndUserId(id, userId);

    return event.map(eventMapper::toDto) //Якщо Entity знайдено, мапп в DTO
        .map(ResponseEntity::ok)
        .orElseGet(() -> ResponseEntity.notFound().build());
}

```

```

@PutMapping("event/{id}")
public ResponseEntity<EventDTO> updateEvent(@PathVariable Long id,
                                             @Valid @RequestBody EventDTO eventDTO,
Authentication authentication) {
    String userId = getUserIdFromAuthentication(authentication);
    List<Long> participantIds = null;
    if (eventDTO.getParticipantIds() != null) {
        participantIds = eventDTO.getParticipantIds();
    }
    Optional<Event> existingEventOpt =
eventService.findEventByIdAndUserId(id, userId);

    if (existingEventOpt.isPresent()) {
        Event existingEvent = existingEventOpt.get();

        eventMapper.updateEventFromDto(eventDTO, existingEvent); //маппер
для оновлення Entity даним з DTO

        Optional<Event> updatedEventOpt = eventService.updateEvent(id,
existingEvent, userId, participantIds);

        return updatedEventOpt.map(eventMapper::toDto)
            .map(ResponseEntity::ok)

```

```

        .orElseGet(() ->
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build());
//Fallback
    } else {

        return ResponseEntity.notFound().build();
    }
}

@DeleteMapping("event/{id}")
public ResponseEntity<Void> deleteEvent(@PathVariable Long id,
Authentication authentication) {
    String userId = getUserIdFromAuthentication(authentication);
    boolean deleted = eventService.deleteEvent(id, userId);

    if (deleted) {
        return ResponseEntity.noContent().build();
    } else {
        return ResponseEntity.notFound().build();
    }
}
}

```

Б.1.2 Код класу сервіса для бізнес-логіки роботи з подіями:

```

@Service
@AllArgsConstructor
public class EventServiceImpl implements EventService {

    private final EventRepository eventRepository;
    private final ContactRepository contactRepository;
    private final EmailServiceImpl emailService;

    @Override
    public List<Event> findAllEventsForUser(String userId) {

        return eventRepository.findByUserId(userId);
    }

    @Override

```

```

public Optional<Event> findEventByIdAndUserId(Long eventId, String userId)
{
    return eventRepository.findByIdAndUserId(eventId, userId);
}

@Override
public Event createEvent(Event event, String userId, List<Long> participantIds)
{
    event.setCreationTime(LocalDateTime.now());
    event.setUserId(userId);

    //Встановлення учасників
    Set<Contact> participants = new HashSet<>();
    if (participantIds != null && !participantIds.isEmpty()) {
        participants = participantIds.stream()
            .map(contactId -> contactRepository.findByIdAndUserId(contactId,
userId))
            .filter(Optional::isPresent) //відкинути контакти, які не знайдено
            або не належать користувачу
            .map(Optional::get) //взяти Contact з Optional
            .collect(Collectors.toSet()); //зібрати в Set
    }
    //Встановити знайдених та перевірених учасників до сутності Event
    event.setParticipants(participants);

    if (event.isReminderSet()) {
        if (event.getReminderOffsetMinutes() != null && event.getPlannedTime()
!= null) {
            event.setReminderDateTime(event.getPlannedTime().minusMinutes(event.getRem
inderOffsetMinutes())); //розрахунок часу нагадування
        } else if (event.getPlannedTime() != null) {
            event.setReminderDateTime(event.getPlannedTime());
            event.setReminderOffsetMinutes(0);
        } else {
            event.setReminderSet(false);
        }
    } else {
        event.setReminderDateTime(null);
        event.setReminderOffsetMinutes(null);
    }
}

```

```

    }
    if (!participants.isEmpty()) {
        //Розсилка листів кожному учаснику події (пошту яких надано)
        for (Contact participant : participants) {
            if (participant.getEmail() != null && !participant.getEmail().isEmpty())
{
                emailService.sendNotificationToEmail(participant.getEmail(), event);
            }
        }
    }
    return eventRepository.save(event);
}

```

```

@Override
@Transactional
public Optional<Event> updateEvent(Long eventId, Event updatedEventDetails,
String userId, List<Long> participantIds) {

```

```

    Optional<Event> existingEventOpt =
eventRepository.findByIdAndUserId(eventId, userId);

```

```

    if (existingEventOpt.isPresent()) {
        Event existingEvent = existingEventOpt.get();

```

```

        existingEvent.setTitle(updatedEventDetails.getTitle());
        existingEvent.setDescription(updatedEventDetails.getDescription());
        existingEvent.setPlannedTime(updatedEventDetails.getPlannedTime());
        existingEvent.setCompleted(updatedEventDetails.isCompleted());
        existingEvent.setReminderSet(updatedEventDetails.isReminderSet());

```

```

existingEvent.setReminderOffsetMinutes(updatedEventDetails.getReminderOffset
Minutes());

```

```

        Set<Contact> newParticipants = new HashSet<>();
        if (participantIds != null && !participantIds.isEmpty()) {
            newParticipants = participantIds.stream()
                .map(contactId -> contactRepository.findByIdAndUserId(contactId,
userId))
                .filter(Optional::isPresent)
                .map(Optional::get)
                .collect(Collectors.toSet());
        }

```

```

        existingEvent.setParticipants(newParticipants);

        if (existingEvent.isReminderSet()) {
            if (existingEvent.getReminderOffsetMinutes() != null &&
existingEvent.getPlannedTime() != null) {

existingEvent.setReminderDateTime(existingEvent.getPlannedTime().minusMinut
es(existingEvent.getReminderOffsetMinutes()));
            } else if (existingEvent.getPlannedTime() != null) {

existingEvent.setReminderDateTime(existingEvent.getPlannedTime());
            existingEvent.setReminderOffsetMinutes(0);
            } else {
                existingEvent.setReminderSet(false);
            }
        } else {
            existingEvent.setReminderDateTime(null);
            existingEvent.setReminderOffsetMinutes(null);
        }

        Event savedEvent = eventRepository.save(existingEvent);

        if (!newParticipants.isEmpty()) {
            for (Contact participant : newParticipants) {
                if (participant.getEmail() != null &&
!participant.getEmail().isEmpty()) {
                    emailService.sendNotificationToEmail(participant.getEmail(),
savedEvent);
                }
            }
        }

        return Optional.of(savedEvent);

    } else {
        return Optional.empty();
    }
}

@Override
@Transactional
public boolean deleteEvent(Long eventId, String userId) {

```

```

    Optional<Event> eventOpt = eventRepository.findByIdAndUserId(eventId,
userId);

    if (eventOpt.isPresent()) {
        eventRepository.deleteById(eventId);
        return true;
    } else {

        return false;
    }
}
}

```

Б.1.3 Код репозиторію для взаємодії з базою даних для роботи з подіями

```

public interface EventRepository extends JpaRepository<Event, Long> {

    List<Event> findByIdAndUserId(Long eventId, String userId);

    List<Event> findByIdAndUserId(Long eventId, String userId);

    List<Event> findByParticipants_Id(Long contactId);

    List<Event>
    findByReminderDateTimeLessThanEqualAndReminderSetTrueAndReminderSent
    FalseAndCompletedFalse(LocalDateTime thresholdTime);
}

```

ДОДАТОК В
Слайди презентації

Національний Університет «Запорізька політехніка»
Кафедра програмних засобів

Розробка вебзастосунку для організації часу та планування подій

Виконав
Ст. гр. КНТ-131
Керівник
к.т.н., доцент

Максим БОЛУЧЕВСЬКИЙ
Тетяна ФЕДОРОНЧАК

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес розробки програмного забезпечення для організації часу і планування подій.

Предмет дослідження – вебзастосунки для організації часу та планування подій.

Мета роботи – розробка вебзастосунку для особистого планування подій.

Рисунок В.2 – Слайд 2

Завдання до роботи

- Аналіз предметної області та програмних для планування
- Вибір інструментів та технологій розробки
- Проектування архітектури
- Розробка вебзастосунку для організації часу та планування подій
- Тестування розробленого програмного забезпечення

3

Рисунок В.3 – Слайд 3

Порівняння альтернативних програмних засобів

Параметр	Google Tasks	Google Keep	Any.do	TickTick	ВЗОЧПП
CRUD-операції з подіями	+	+	+	+	+
Налаштування нагадувань	+	+	+	+	+
Залучення інших користувачів до задачі	–	+	+-	+-	+
Можливість додавати файли	–	+	+	–	–
Кросплатформність/Крос браузерність	+	+	+	+	+
Простота і зручність інтерфейсу	+	+	+-	+-	+
Безкоштовність	+	+	+-	+-	+

4

Рисунок В.4 – Слайд 4

Вибір мови програмування

Критерій	Java	Python	JavaScript (Node.js)	PHP
Продуктивність/ швидкість	+	+-	+-	+-
Зручність	+	++	+	+
Кросплатформність	+	+	+	+
Механізми безпеки	+	+	+-	+-
Фреймворки	++	++	++	+
Робота з БД	++	+	+	+
Масштабованість	++	+	+	+-
Безкоштовність	+	+	+	+

5

Рисунок В.5 – Слайд 5

Вибір мови програмування

Характеристика	TS	JS
Типізація	Сильна, статична типізація; підтримує динамічну.	Лише динамічна типізація.
Підтримка ООП	Повна (класи, інтерфейси, дженеріки, модифікатори доступу тощо).	Обмежена (класи ES6, нестрога типізація)
Компіляція	Компілюється в JS.	Непотрібна.
Сумісність з IDE	Підтримує автодоповнення, рефакторинг, визначення помилок	Базова підтримка, більш обмежена.
Робота з фреймворками	Широке коло зручних фреймворків.	Здебільшого ті ж фреймворки, але деякі не оптимізовано або не працюють взагалі (NestJS, tRPC).
Підтримка спільноти	Постійно зростаюча спільнота.	Найбільша спільнота серед усіх мов програмування.

6

Рисунок В.6 – Слайд 6

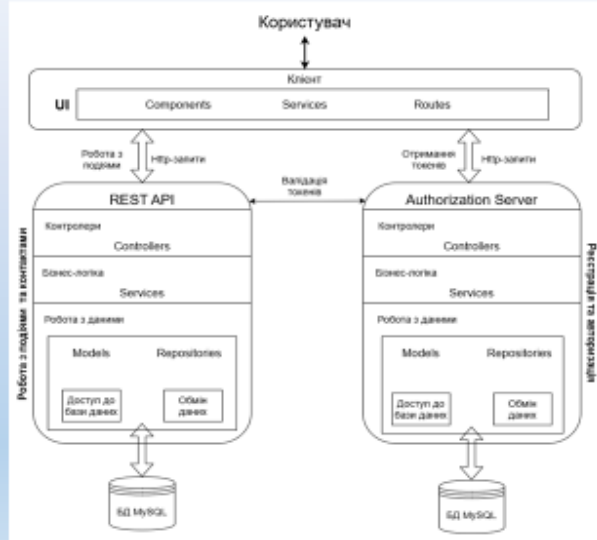
Вибір середовища розробки

Параметр	IntelliJ IDEA	Eclipse	ApacheNetBeans
Підтримка мов	Java, Kotlin, Groovy, JS (TS), Python, HTML	Java, C/C++, PHP, JS	Java, C/C++, PHP, JS, HTML
Кросплатформність	+	+	+
Автодоповнення	Найкраще, рівня ШІ	Непогане	Середнє
Продуктивність	Швидка, висока оптимізація	Середня, потребує налаштувань	Середня
Підтримка Git	Вбудована і дуже зручна	Доступна через плагіни	Вбудована, але посередня
Підтримка фреймворків	Найкраща, велике коло фреймворків, якісна підтримка	Добра, іноді потребує плагінів	Порівняно обмежена
Безкоштовність	Common – безкоштовна, Ultimate – платна	Безкоштовна	Безкоштовна

7

Рисунок В.7 – Слайд 7

Архітектура вебзастосунку



8

Рисунок В.8 – Слайд 8

The diagram illustrates the project structure for three different components:

- Клиент (Client):** Contains a `src` directory with an `app` package. The `app` package includes `components`, `router`, and `services` sub-packages. It also has a `main` package. The project files are `index.html`, `main.js`, and `styles.css`.
- Cepcep REST API:** Contains a `src` directory with a `main` package. The `main` package includes a `java` package. The `java` package includes `ua.rud.plan_service`, `config`, `controller`, `dto`, `mapper`, `model`, `repository`, and `service` sub-packages. It also has a `resources` package. The project files are `planServiceApplication.java` and `plan.xml`.
- Cepcep Aeroportal.ru:** Contains a `src` directory with a `main` package. The `main` package includes a `java` package. The `java` package includes `ua.rud.authserver`, `advice`, `config`, `controller`, `dto`, `mapper`, `model`, `repository`, and `service` sub-packages. It also has a `resources` package. The project files are `AuthServerApplication.java` and `auth.xml`.

All three projects are connected to a central database labeled **БД (Database)**.

10

```

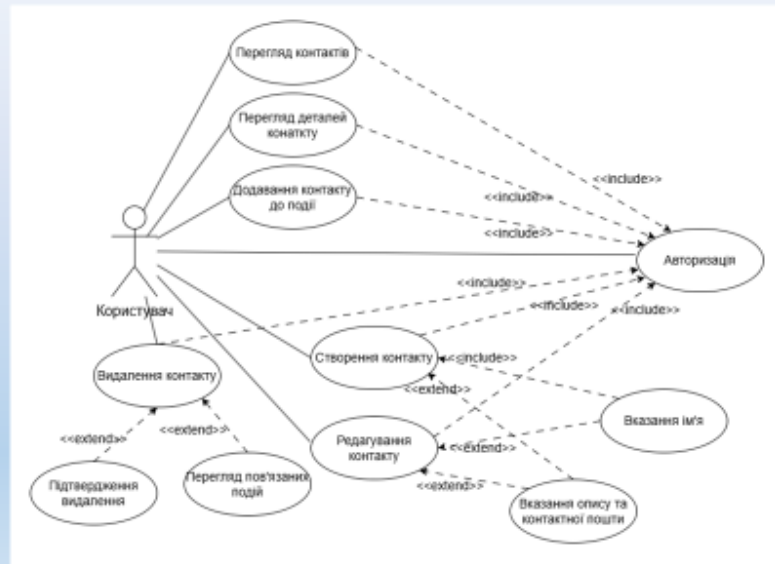
    usecaseDiagram
        actor User as Пользователь
        usecase UC1 as Перевод денежных средств
        usecase UC2 as Перевод деталей билета
        usecase UC3 as Списание на оплату
        usecase UC4 as Возврат билета
        usecase UC5 as Автоплатеж
        usecase UC6 as Сторонний билет
        usecase UC7 as Резервирование билета
        usecase UC8 as Возврат кассы за детализацию чека
        usecase UC9 as Возврат кассы за билет
        usecase UC10 as Возврат кассы за оплату
        usecase UC11 as Получение извещение

        User -- UC1
        User -- UC2
        User -- UC3
        User -- UC4
        User -- UC5
        User -- UC6
        User -- UC7
        User -- UC8
        User -- UC9
        User -- UC10
        User -- UC11

        UC1 -.-> UC5 : <<include>>
        UC2 -.-> UC5 : <<include>>
        UC3 -.-> UC5 : <<include>>
        UC4 -.-> UC5 : <<include>>
        UC5 -.-> UC6 : <<include>>
        UC5 -.-> UC7 : <<include>>
        UC6 -.-> UC7 : <<include>>
        UC7 -.-> UC8 : <<include>>
        UC7 -.-> UC9 : <<include>>
        UC7 -.-> UC10 : <<include>>
        UC7 -.-> UC11 : <<include>>
        UC8 -.-> UC9 : <<include>>
        UC9 -.-> UC10 : <<include>>
        UC10 -.-> UC11 : <<include>>
    
```

Рисунок В.10 – Слайд 10

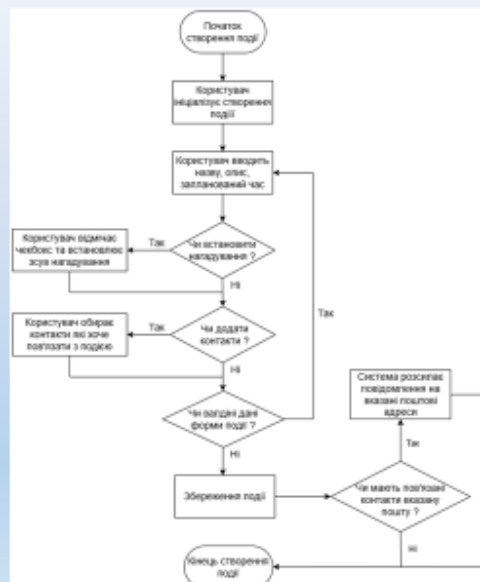
Діаграма прецедентів роботи з контактами



11

Рисунок В.11 – Слайд 11

Блок-схема створення події



12

Рисунок В.12 – Слайд 12

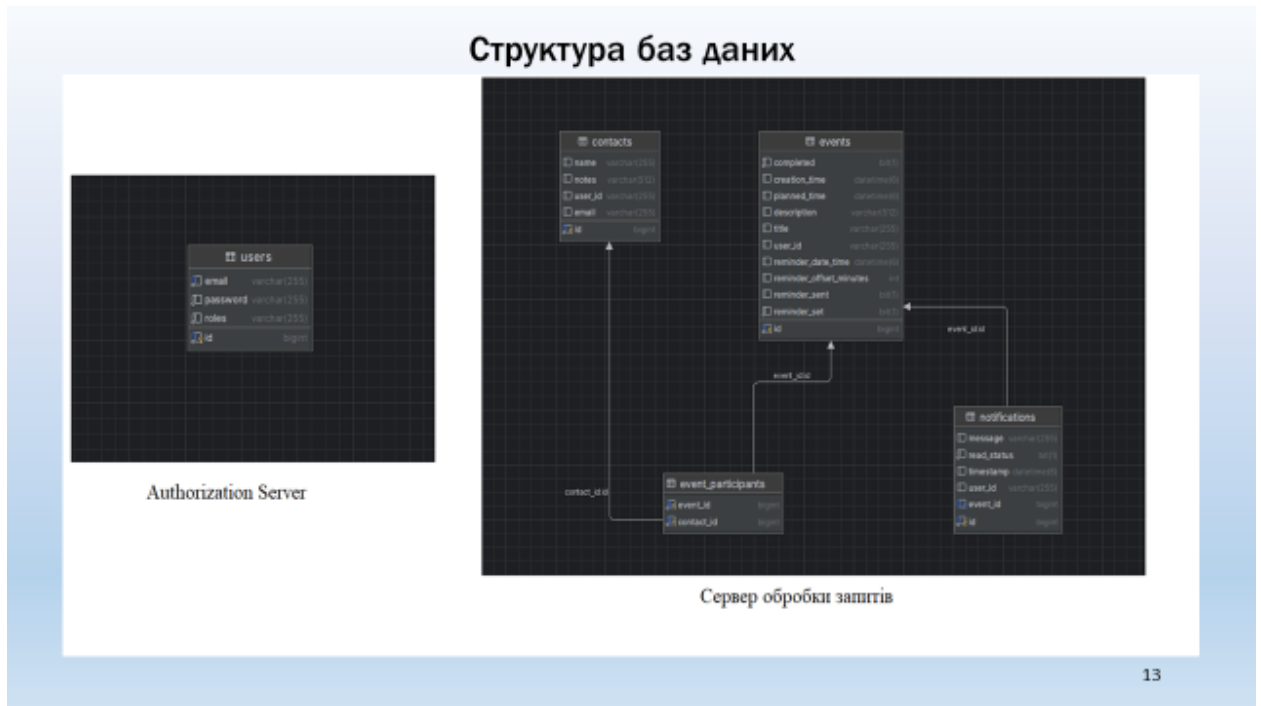


Рисунок В.13 – Слайд 13

Сторінки реєстрації та авторизації

Реєстрація

Емейл:

Пароль:

Підтвердження паролю:

[Зареєструватися](#)

[Вже є акаунт? Логін](#)

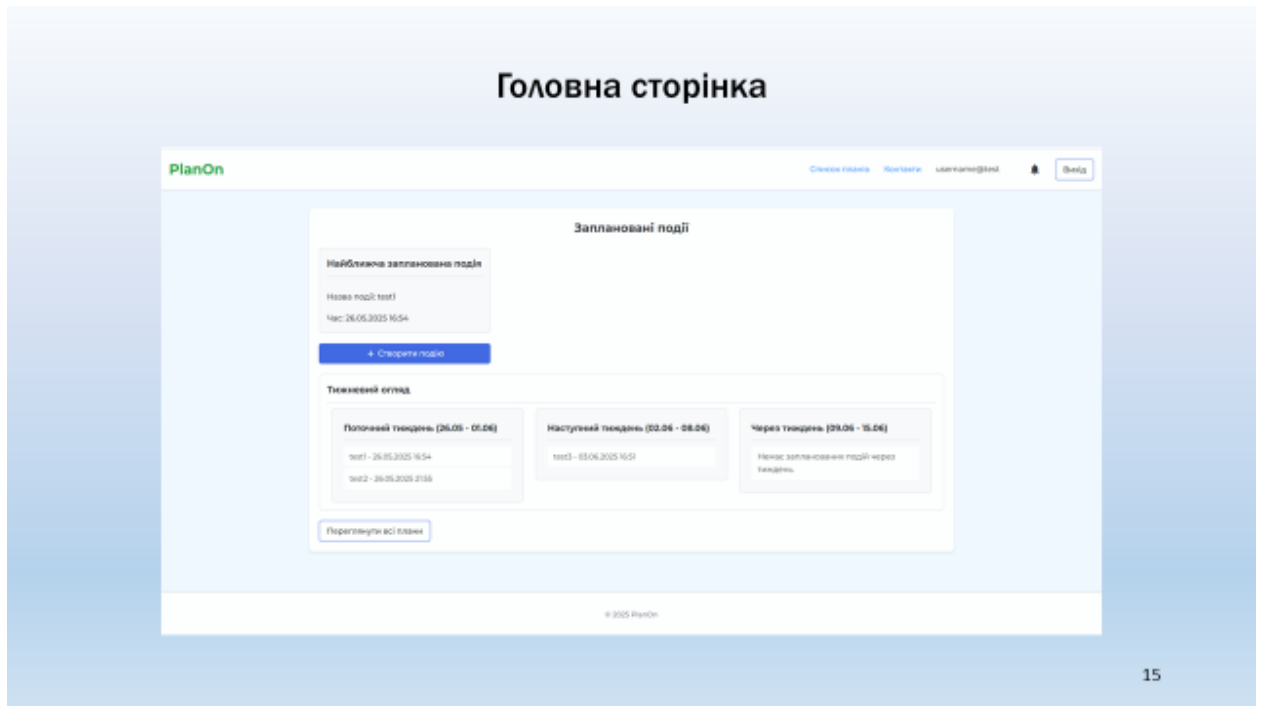
Будь-ласка, увійдіть.

Ім'я користувача:

Пароль:

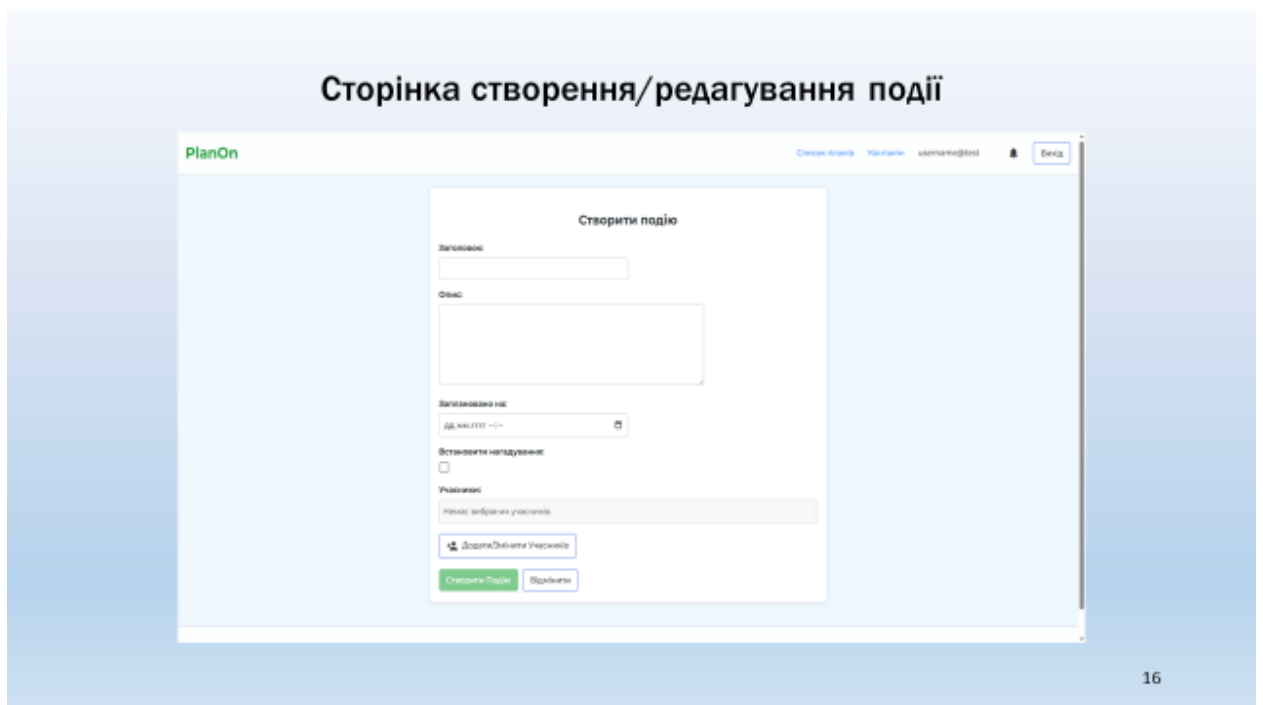
[Увійти](#)

Рисунок В.14 – Слайд 14



15

Рисунок В.15 – Слайд 15



16

Рисунок В.16 – Слайд 16

Тестування вебзастосунку

Функціонал	Google 136.0.7103.93	Chrome	Mozilla Firefox 138.0.4	Microsoft 136.0.3240.92	Edge
Реєстрація користувача	Пройшов		Пройшов		Пройшов
Авторизація користувача	Пройшов		Пройшов		Пройшов
Перегляд списку запланованих справ на найближчі 3 тижні	Пройшов		Пройшов		Пройшов
Перегляд повного списку усіх запланованих справ	Пройшов		Пройшов		Пройшов
Створення події	Пройшов		Пройшов		Пройшов
Редагування події	Пройшов		Пройшов		Пройшов
Видалення події	Пройшов		Пройшов		Пройшов
Встановлення нагадування	Пройшов		Пройшов		Пройшов
Отримання нагадування	Пройшов		Пройшов		Пройшов
Створення контакту	Пройшов		Пройшов		Пройшов
Редагування контакту	Пройшов		Пройшов		Пройшов
Видалення непов'язаного контакту	Пройшов		Пройшов		Пройшов
Видалення контакту пов'язаного з подіями	Пройшов		Пройшов		Пройшов
Відправка повідомлення на пошту контакту, якого додано до події	Пройшов		Пройшов		Пройшов

17

Рисунок В.17 – Слайд 17

Висновки

В ході виконання даної кваліфікаційної дипломної роботи бакалавра було детально проаналізовано та досліджено процес розробки програмного забезпечення для організації часу та планування подій.

Розглянуто предметну область та аналогічні програмні рішення. Сформульовано функціональні та інтерфейсні вимоги. Для реалізації проекту обрано мову програмування Java для серверних частин, TypeScript для клієнтської, та фреймворки Spring Boot та Angular відповідно. Описано архітектуру та структуру розробленого вебзастосунку. Розглянуто основні інтерфейсні рішення.

Протестовано розроблений вебзастосунок для організації часу та планування подій. За результатами тестування було визначено, що розроблена система відповідає поставленим вимогам технічного завдання та може використовуватися кінцевими користувачами за призначенням.

18

Рисунок В.18 – Слайд 18



Рисунок В.19 – Слайд 19