МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Аліна САВЧЕНКО
«_____»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ДИПЛОМНА РОБОТА, ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

"МАГІСТРА"

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: «Вебсервіс для пасажирських перевезень»

Виконав: Загурський Максим Валерійович

Керівник: професор Зіатдінов Юрій Кашафович

Нормоконтролер _____Ігор РАЙЧЕВ

Київ – 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *Комп'ютерних наук та технологій*

Кафедра *Комп'ютерних інформаційних технологій*

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Аліна САВЧЕНКО

"___"_____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи студента

Загурського Максима Валерійовича

(прізвище, ім'я, по батькові)

1. **Тема роботи:** «Вебсервіс для пасажирських перевезень», затверджена наказом ректора від 29 вересня 2023 за № 1976/ст.

2. **Термін виконання роботи: :** з 02 жовтня 2023 по 31 грудня 2023.

3. **Вихідні дані до роботи:** основні підходи та аналіз створення веб-додатків, теоретичні відомості веб-розробки, інструментальні засоби для веб-застосунків..

4. **Зміст пояснювальної записки (перелік питань, що підлягають розробці):** вступ, дослідження та аналіз веб-додатків, опис та вибір інструментів та технологій для реалізації веб-додатку, проектування веб-додатку, розробка веб-сервісу.

5. **Перелік обов'язкового графічного матеріалу:** слайди презентації в MS PowerPoint.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Дослідження та аналіз предметної області	02.10.23 – 10.10.23	
2	Проведення консультацій з науковим керівником	10.10.23 – 18.10.23	
3	Написання та підготовка розділу 1	19.10.23 – 25.10.23	
4	Написання та підготовка розділу 2	26.10.23 – 06.11.23	
5	Написання та підготовка розділу 3	07.11.23 – 15.11.23	
6	Оформлення пояснювальної записки	15.11.23 – 25.11.23	
7	Написання та підписання рецензії та відгуку	26.11.23 – 02.12.23	
8	Підготовка презентації та доповіді	02.12.23 – 12.12.23	
9.	Підготовка до захисту та попередній захист дипломної роботи	13.12.23 – 20.12.23	

7. Дата видачі завдання: «02» жовтня 2023 р.

Керівник дипломної роботи _____ Юрій ЗІАТДІНОВ

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Максим ЗАГУРСЬКИЙ

(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Вебсервіс для пасажирських перевезень» викладена на 83 сторінках і містить 37 рисунків. Список бібліографічних посилань складається з 13 найменувань.

Ключові слова: ВЕБСЕРВІС ПАСАЖИРСЬКИХ ПЕРЕВЕЗЕНЬ, ВЕБ-ДОДАТОК, MERN, ПРОЕКТУВАННЯ ВЕБ-СИСТЕМИ, БАЗИ ДАНИХ.

Об’єкт дослідження – веб-додатки для пасажирських перевезень.

Предмет дослідження – оптимізація процесу бронювання, покращення взаємодії користувачів, розробка заходів для забезпечення конфіденційності та безпеки інформації, використання нових технічних аспектів розробки.

Мета роботи є детальне вивчення, аналіз та розробка веб-сервісу, спрямованого на оптимізацію та поліпшення процесів пасажирських перевезень, як для пасажирів так і для перевізників.

Актуальність дипломної роботи полягає в створенні більш сучасного та ефективного вебсервісу, який полегшить взаємодію приватних перевізників з клієнтами, а саме з пасажирами на пряму без участі сторонніх осіб.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

SPA – Single Page Application (односторінковий застосунок)

MPA – Multi Page Application (багатосторінковий застосунок)

PWA – Progressive Web Application (прогресивний веб-додаток)

HTML – Hyper Text Markup Language (мова розмітки гіпертексту)

API – Application programming interface (інтерфейс прикладного програмування)

JSON – JavaScript Object Notation (текстовий формат обміну даними)

IDE – Integrated Development Environment (інтегроване середовище розробки)

URL – Uniform Resource Locator (уніфікований покажчик інформаційного ресурсу)

HTTP – Hypertext Transfer Protocol (протокол передачі гіпертексту)

HTTPS – Hypertext Transfer Protocol Secure (захищений протокол передачі гіпертексту)

ВСТУП

Відзначаючи важливість пасажирських перевезень у розвитку національної економіки та забезпеченні мобільності населення, важливо висвітлити етапи їхнього розвитку в Україні. Протягом останніх десятиліть у країні спостерігаються суттєві зміни у сфері транспортної інфраструктури, які вплинули на покращення та розширення мережі пасажирських перевезень.

Реформи та інвестиції у транспортну галузь сприяли зростанню конкуренції серед перевізників та покращенню якості послуг. Розвиток швидкісних залізниць, модернізація автопарку та впровадження новітніх технологій у сфері авіаційних та автомобільних перевезень стали ключовими напрямками у вдосконаленні пасажирських перевезень.

Зростаюча потреба в ефективному транспорті та росту міських агломерацій підкреслює важливість досконалення системи пасажирських перевезень. Зокрема, розвиток штучного інтелекту, впровадження електромобільності та створення інтегрованих систем квитування сприяють створенню зручних та доступних для населення транспортних рішень.

Однак, разом з усім цим, виникає необхідність гарантувати безпеку пасажирських перевезень. Впровадження сучасних систем безпеки, навчання персоналу та вдосконалення правил експлуатації транспортних засобів стають важливими кроками у забезпеченні безпеки пасажирів.

Пасажирські перевезення в Україні не лише відіграють ключову роль у розвитку економіки та забезпеченні мобільності населення, але й перебувають на етапі постійного вдосконалення та адаптації до сучасних викликів та потреб суспільства.

РОЗДІЛ 1. АНАЛІЗ, РОЗРОБКА ТА БЕЗПЕКА ВЕБ-ДОДАТКІВ

1.1. Веб-додаток його переваги та недоліки.

Веб-додаток - це програмне забезпечення, розроблене для використання через веб-браузери користувачів. Ці додатки працюють на веб-серверах та надають можливість взаємодії з користувачами, обробляти дані, зберігати інформацію на сервері та надавати різноманітні функціональні можливості через інтерфейс веб-сторінки. Вони стали невід'ємною частиною сучасного інтернет-світу і грають ключову роль у забезпеченні доступу до різноманітних послуг та ресурсів через веб-браузери. Ці додатки надають користувачам широкий спектр функціональності, однак вони також мають свої переваги і недоліки.

Основними перевагами веб-додатків є:

- Доступність з будь-якого пристрою. Це дозволяє користувачам легко отримувати доступ до важливих даних та сервісів зі смартфонів, планшетів, ноутбуків тощо;
- Не потребує встановлення що полегшує їх використання;
- Веб-додатки легко оновлюються централізовано на сервері, тому користувачам не потрібно перейматися про оновлення на своїх пристроях;
- Мультиплатформенність. Це дозволяє використовувати їх на різних операційних системах і пристроях.

Основними недоліками веб-додатків є:

- Залежність від Інтернету. Відсутність доступу до Інтернету може призвести до неможливості використовувати їх;

- Можливість затримок. Швидкість роботи веб-додатків залежить від швидкості та стабільності Інтернет-з'єднання, тому користувачі можуть стикатися з затримками під час завантаження сторінок чи взаємодії з додатком;
- Обмежена взаємодія з пристроями. Веб-додатки не завжди можуть використовувати всі можливості пристроїв (наприклад, камери або сенсори) так, як це роблять мобільні додатки;
- Безпека. Веб-додатки можуть бути більш вразливими до кіберзагроз, так як їх виконання відбувається на віддалених серверах.

У підсумку, веб-додатки є потужним інструментом для забезпечення доступу до різноманітних сервісів та даних через веб-браузери. Вони надають безліч переваг, таких як доступність з будь-якого пристрою та економічність розробки. Однак вони також мають свої недоліки, такі як залежність від Інтернету та обмежена функціональність.

1.2. Відмінність веб-додатку від веб-сайту.

У сучасному світі, коли Інтернет став неодмінною частиною нашого повсякденного життя, терміни "веб-додаток" і "веб-сайт" часто вживаються замінно або сприймаються як схожі поняття. Однак різниця між ними є важливою та ділиться на різні аспекти, включаючи функціональність, цільову аудиторію, способи взаємодії та технічну реалізацію. У цьому дослідженні ми розглянемо ці ключові різниці між веб-додатками та веб-сайтами, щоб краще зрозуміти їхню природу та роль у віртуальному просторі.

Веб-сайт - це колекція веб-сторінок, які містять інформацію, текст, графіку, відео та інші ресурси. Ось деякі ключові характеристики веб-сайту:

- Статичний або інформаційний характер: Веб-сайти зазвичай мають статичний характер і призначені для надання інформації користувачам. Це можуть бути блоги, новинні сайти, сайти-візитки компаній, сайти для відображення фотографій тощо;

- Обмежена взаємодія: Веб-сайти зазвичай не мають великої взаємодії з користувачем. Вони надають інформацію, і користувачі можуть переглядати цю інформацію, але зазвичай не можуть виконувати складних дій;
- Зазвичай не вимагає авторизації: В більшості випадків веб-сайти не вимагають від користувачів авторизації чи входу в систему;
- Призначені для публічного доступу: Веб-сайти призначені для публічного доступу, і їх вміст доступний для всіх користувачів Інтернету;
- Спрощена архітектура: Архітектура веб-сайту взагалі є спрощеною, і він може бути побудований за допомогою стандартних технологій, таких як HTML, CSS ТА JavaScript.

Веб-додаток, навпаки, має більше функціональних можливостей і взаємодії з користувачем. Ось деякі основні характеристики веб-додатка:

- Динамічний та інтерактивний характер: Веб-додатки зазвичай мають динамічний характер і надають можливість користувачам виконувати різні дії та взаємодіяти з системою. Це можуть бути соціальні мережі, електронні поштові сервіси, онлайн-магазини, інструменти для спільної роботи тощо;
- Складні функції та обробка даних: Веб-додатки зазвичай включають багато функцій, таких як авторизація, робота з базами даних, обробка форм, сповіщення, аналітика тощо;
- Може вимагати авторизації: Деякі веб-додатки можуть вимагати, щоб користувачі авторизувалися перед отриманням доступу до певних функцій або даних;
- Може бути призначений для конкретного користувача або спільноти: Веб-додатки можуть бути розроблені для обмеженої аудиторії або спеціалізованого використання;
- Складніша архітектура: Архітектура веб-додатка зазвичай складніша, і вона може включати серверну частину, бази даних, клієнтську логіку та інші компоненти.

Для кращого розуміння різниці веб-додатка та веб-сайту давайте розглянемо декілька прикладів:

- Веб-сайт: Блог одного автора, який публікує статті та інформацію про свої заходи. Користувачі можуть переглядати статті та залишати коментарі, але взаємодія обмежена;
- Веб-додаток: Онлайн-магазин електроніки, де користувачі можуть переглядати товари, додавати їх до кошика, здійснювати покупки та виконувати оплату. Додаток має складну логіку обробки замовлень та вимагає авторизації;
- Веб-сайт: Сайт-візитка для невеликої компанії, який містить інформацію про послуги та контактну інформацію. Взаємодія обмежена, і користувачі можуть використовувати сайт лише для перегляду інформації;
- Веб-додаток: Соціальна мережа, де користувачі можуть створювати профілі, додавати друзів, публікувати повідомлення та взаємодіяти один з одним. Додаток надає багато функцій для спілкування та спільної діяльності.

Веб-додаток і веб-сайт - це два різних типи веб-ресурсів, і їх вибір залежить від конкретних потреб і цілей. Веб-сайт зазвичай надає інформацію та обмежену взаємодію, тоді як веб-додаток дозволяє виконувати більше завдань та взаємодіяти з користувачами на більш глибокому рівні. Розуміння цієї різниці допоможе вам краще визначити, який тип веб-ресурсу підходить для вашого проекту.

1.3. Основні види веб-додатків.

Веб-додатки стали невід'ємною частиною нашого сучасного світу. З плином часу розробники зрозуміли, що існують різні підходи до створення веб-додатків, які відрізняються за функціональністю та архітектурою. У цьому розділі ми розглянемо три основних види веб-додатків: PWA (Progressive Web App), SPA (Single Page Application) та MPA (Multi-Page Application). Ми також проаналізуємо їхні переваги та недоліки.

PWA - це новий підхід до створення веб-додатків, який поєднує в собі найкращі риси веб-сайтів і мобільних додатків. Основні особливості PWA:

Переваги PWA:

- Кросплатформенність: PWA працюють на будь-якому пристрої з веб-браузером, що робить їх кросплатформенними;
- Офлайн-режим: PWA можуть працювати в офлайн-режимі, завдяки можливості кешування контенту;
- Інсталяція на головний екран: Користувачі можуть додати PWA на головний екран свого пристрою, подібно до звичайних мобільних додатків;
- Оновлення безпеки: PWA оновлюються автоматично, що дозволяє розробникам надавати безпечні версії додатків.

Недоліки PWA:

- Обмежені функції пристрою: PWA можуть мати обмежений доступ до функцій пристрою, таких як камера та геолокація;
- Підтримка браузера: Деякі функції PWA можуть не підтримуватися всіма браузерами.

SPA - це веб-додаток, який завантажує одну сторінку та динамічно оновлює її вміст за допомогою JavaScript. Основні особливості SPA:

Переваги SPA:

- Плавність інтерфейсу: SPA надають користувачам плавну і відзивчиву інтерактивність завдяки асинхронному завантаженню даних;
- Модульність: SPA легко підрозділяються на модулі, що полегшує розробку і підтримку.

Недоліки SPA:

- Проблеми з SEO: Індексація SPA пошуковими системами може бути складною через динамічність завантаження контенту;
- Ініціалізація: Ініціалізація SPA може займати більше часу, ніж у звичайних мульти-сторінкових додатках.

MPA - це класичний підхід до створення веб-додатків, при якому кожна сторінка завантажується окремо. Основні особливості MPA:

Переваги MPA:

- SEO-дружність: MPA мають легше індексуються пошуковими системами завдяки окремим URL для кожної сторінки;
- Простота розробки: Розробка MPA може бути менш складною, оскільки вони не вимагають великої кількості JavaScript.

Недоліки MPA:

- Оновлення сторінокк: При переході між сторінками MPA повторно завантажують весь HTML-код, що може затримувати завантаження;
- Відсутність реактивності: MPA можуть бути менш реагуючими в порівнянні з SPA через потребу у повному перезавантаженні сторінок.

1.4. Важливість вибору архітектури для веб-додатку.

Обов'язковість вибору архітектури для веб-додатку є важливою складовою процесу розробки програмного забезпечення. Вибір архітектури допомагає визначити структуру та організацію вашого веб-додатку, що в свою чергу впливає на його продуктивність, безпеку, розширюваність та інші аспекти.

Ось кілька обґрунтувань, чому вибір архітектури є обов'язковим етапом розробки веб-додатку:

- Оптимізація продуктивності: Різні архітектури мають різний вплив на продуктивність. Вибір оптимальної архітектури дозволяє забезпечити швидку та ефективну роботу веб-додатку;
- Забезпечення безпеки: Деякі архітектурні рішення можуть забезпечити кращий рівень безпеки для вашого додатку. Наприклад, використання розділення на клієнт-сервер може допомогти уникнути певних видів атак;
- Розширюваність: Архітектура веб-додатку повинна бути готовою до масштабування. Вибір правильної архітектури може полегшити майбутнє розширення функціональності та вмісту додатку;

- Підтримка обслуговування: Архітектура також повинна враховувати питання обслуговування та підтримки. Вибір архітектури може вплинути на вартість та ефективність обслуговування додатку;
- Масштабованість розробки: Розробка великих веб-додатків зазвичай ведеться командами розробників. Архітектурні рішення допомагають узгодити роботу різних команд та зберігати однорідність коду;
- Документація та розуміння коду: Архітектурні рішення сприяють структурації коду, що робить його більш зрозумілим для розробників та підтримувачів.

Важливо враховувати, що вибір архітектури повинен бути обґрунтованим і враховувати потреби вашого проекту. Різні види веб-додатків (наприклад, багатосторінкові сайти, односторінкові додатки, веб-сервіси тощо) можуть вимагати різних архітектурних підходів. Перед вибором архітектури рекомендується провести аналіз вимог і оцінку можливих варіантів для забезпечення оптимальності та ефективності вашого веб-додатку.

1.5. Архітектура MVC, MVP, веб-компонентів.

MVC-архітектура складається з трьох ключових компонентів, а саме: модель, представлення та контролер (Model-View-Controller).

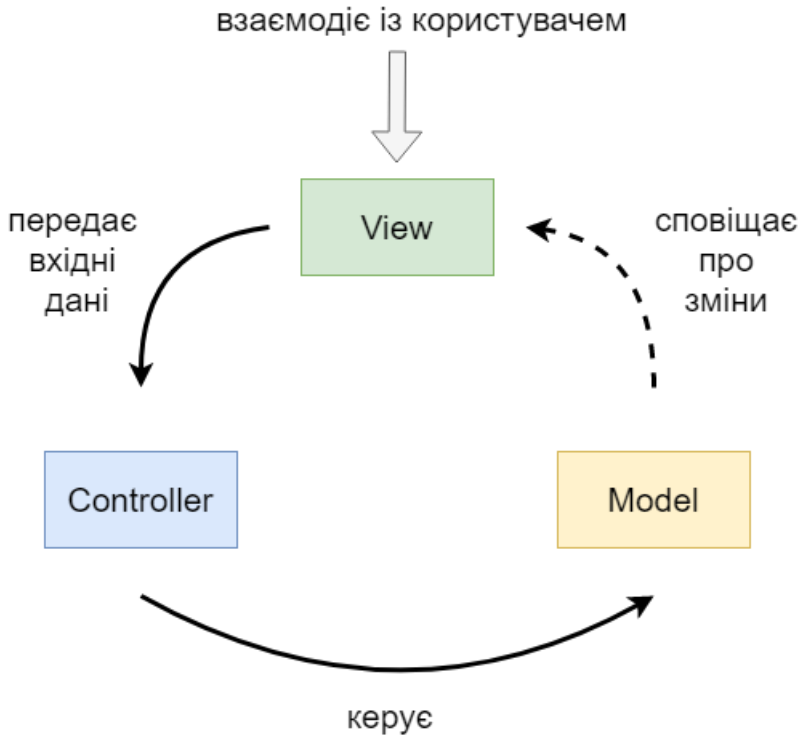

Рисунок 1.1. Діаграма взаємодії між компонентами шаблону MVC

Модель використовується у веб-додатку для керування бізнес-даними. Вона створює зручний формат для даних, не зв'язуючись при цьому з представленням та інтерфейсом користувача. Коли модель змінюється, через оновлення формату даних, вона сповіщає представлення, щоб забезпечити правильну реакцію на ці зміни. Представлення відображає дані моделі та залежить від її поточного стану. Користувачі мають можливість взаємодіяти з представленнями, які надають зручний інтерфейс для операцій читання та редагування. Важливо зазначити, що представлення тісно пов'язане із концепцією шаблонування у JavaScript. Цей метод часто використовується для запобігання створенню великих HTML-розміток у пам'яті, шляхом конкатенації рядків. Він базується на використанні змінних у розмітці, які розділені спеціальним синтаксисом. Важливо зазначити, що шаблони не є представленнями, але можуть відображати об'єкт представлення повністю або частково. Посередником між моделлю та представленням виступає контролер. Він відповідає за оновлення моделі після користувацьких дій та управління логікою веб-застосунку. Використання моделі, представлення та контролера у шаблоні MVC

сприяє модульності в розробці, що дозволяє уникнути повторного дублювання коду і розділити основну логіку клієнтської сторони від інтерфейсу користувача.

Архітектура MVP (Model-View-Presenter) була розроблена на основі архітектури MVC. Основною ідеєю її створення було покращення логіки представлення.

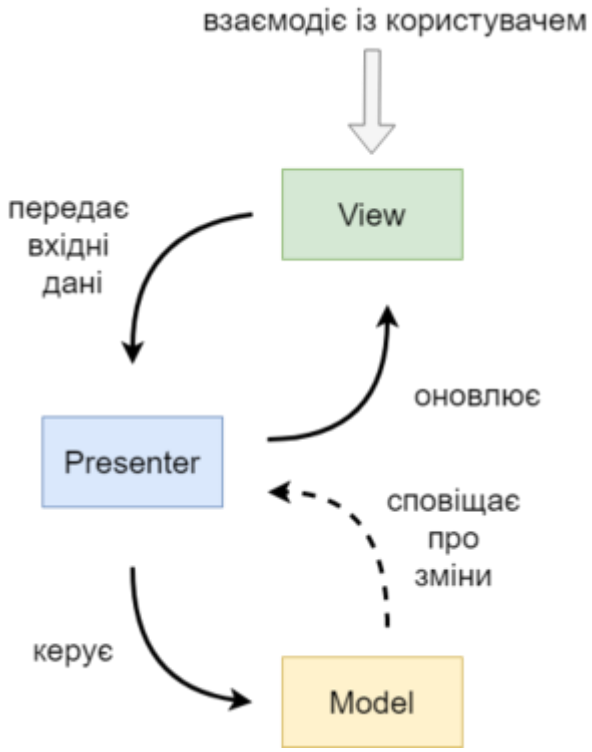

Рисунок 1.2. Діаграма взаємодії між компонентами MVP

У порівнянні з архітектурою MVC, MVP відрізняється наявністю нового компонента - пред'явника, який замінює контролер. Пред'явник відповідає за зв'язок між представленням та моделлю та включає в себе бізнес-логіку користувацького інтерфейсу, необхідну для відображення даних. Пред'явник спостерігає за моделлю та оновлює представлення, коли дані змінюються, тим самим керуючи його станом. Найпоширенішою реалізацією MVP є пасивне представлення, яке містить мінімальну кількість логіки. Пред'явники визначають, що повинно бути відображено в представленні, виходячи з обробки подій та отриманих від користувача даних. При цьому взаємодія між представленням та моделлю здійснюється за закритими правилами. Варіант з використанням наглядового контролера також можливий для

прив'язки даних у MVP. Ця варіація архітектури близька до шаблонів MVC та MVVM. MVP-архітектура надає більше чіткого розмежування між моделлю та представленням та зазвичай використовується у великих корпоративних програмах, де важлива взаємодія з користувачем та використання багатої логіки представлення, побудованої на обробці користувацьких дій.

Архітектура веб-компонентів. Востаннє, веб-компоненти, як одна з архітектурних концепцій, набули великої популярності в сфері веб-розробки і зазнали значних змін у процесі створення веб-додатків. Вони є стандартом, розробленим консорціумом W3C (World Wide Web Consortium) і служать основою для компонентно-орієнтованого програмування, дозволяючи створювати веб-додатки з інкапсульованими та повторно використовуваними віджетами. Веб-компоненти взаємодіють з іншими технічними стандартами, такими як шаблони, користувацькі елементи, Shadow DOM (тіньова модель документу) та HTML imports (методи імпортування HTML документів в інші документи). Ці поняття є важливими складовими веб-компонентів. З метою спрощення використання веб-компонентів розробляються бібліотеки, ідея яких надихнула численні сучасні фреймворки на створення власних реалізацій веб-компонентів з використанням різних підходів.

1.6. Процес створення веб-додатку.

Процес створення веб-додатку може бути складним і включати багато етапів. Першим кроком є визначення мети додатку та його функціональних вимог. Важливо зрозуміти, які завдання додаток повинен вирішувати та для кого він буде призначений. Далі йде етап вибору архітектури для веб-додатку та його проектування. На цьому етапі розробляється архітектура додатку. Визначається структура, базові технології та розташування даних. Також вибирається інтерфейс користувача (UI) і створюється макет (wireframe). Наступна дія яку потрібно зробити це розпочати роботу над серверною частиною. Розробка Back-End включає в себе створення серверів, баз даних, API та логіки бізнес-логіки. Вибір технологій, мов програмування (наприклад, Python, Ruby, Java, Node.js) та фреймворків важливий на

цьому етапі. Після розпочинається не менш важлива робота над клієнтською частиною додатку. Front-End розробка включає в себе створення веб-інтерфейсу, який бачить користувач. Використовуються HTML, CSS та JavaScript, а також фреймворки, такі як React, Angular, або Vue.js. Закінчивши роботу над серверною та клієнтською частиною програміст підключає базу даних до проекту. Дизайн бази даних включає в себе створення схеми, таблиць та зв'язків для зберігання даних. Використовуються системи керування базами даних (СКБД) такі як MySQL, PostgreSQL, або NoSQL рішення, наприклад MongoDB. Наступним кроком є розробка бізнес-логіки. На цьому етапі програмісти реалізують логіку, необхідну для виконання функціональних вимог додатку. Це включає обробку запитів, автентифікацію, авторизацію та інші бізнес-правила. Потім додаток переходить у стадію тестування. Він піддається різним видам тестування, включаючи функціональне тестування, тестування безпеки, тестування продуктивності. Внутрішній тестинг включає тестування додатку командою розробників та QA-інженерами. Всі виявлені помилки виправляються. Реліз та розгортання веб-додатку є слідуючим етапом. Готовий до використання застосунок розгортається на серверах. Важливо встановити необхідні заходи безпеки та моніторингу для успішного функціонування. Після релізу додатку він потребує постійного обслуговування, виправлення помилок і оновлень, щоб підтримувати його актуальність і працездатність. При зростанні користувацької бази застосунок може вимагати масштабування серверів та оптимізації для забезпечення високої продуктивності. Після успішного завершення всіх попередніх пунктів та тестів, додаток готовий до використання користувачами.

Важливо пам'ятати, що розробка веб-додатку - це ітеративний процес, і після запуску потрібно продовжувати вдосконалювати його відповідно до потреб користувачів і змін на ринку.

Давайте розглянемо процеси які слід робити після запуску веб-сервісу:

- Підтримка і оновлення: Після запуску додатку важливо надавати підтримку користувачам. Це включає в себе вирішення їхніх запитів та проблем, виправлення помилок, а також розгляд оновлень і нових функцій;

- Збір і аналіз даних від користувачів може надати цінний інсайт для подальшого вдосконалення додатку. Використовуються інструменти аналітики, такі як Google Analytics;
- Оптимізація продуктивності: З часом можуть виникнути проблеми з продуктивністю додатку під великим навантаженням. Оптимізація серверів, баз даних та клієнтської частини може бути необхідною;
- Розвиток і оновлення: Зміни в вимогах користувачів або на ринку можуть вимагати розширення функціоналу додатку. Розробка нових функцій і їх впровадження - це постійний процес;
- Забезпечення безпеки: Слід надавати особливу увагу забезпеченню безпеки додатку. Це включає в себе виявлення потенційних загроз і вжиття заходів для їх запобігання;
- Супровідні послуги: В залежності від типу додатку і його мети, можуть надаватися супровідні послуги, такі як технічна підтримка, навчання користувачів і консультації;
- Моніторинг і реагування: Важливо постійно моніторити роботу додатку і реагувати на можливі неполадки або відмови. Моніторинг може бути автоматизованим.

Двома найважливішими частинами створення веб-додатків є розробка клієнтської та серверної частини.

Клієнтська частина веб-додатку - це та частина, яку бачить і з якою взаємодіє користувач. Вона відповідає за відображення інформації, з якою користувач може взаємодіяти. Інтерфейс користувача (UI) охоплює дизайн, макет та відображення інформації на веб-сторінках, маючи на меті створити інтуїтивно зрозумілий та комфортний для користувачів спосіб взаємодії.

В сфері веб-технологій для реалізації клієнтської частини використовуються HTML для структури сторінки, CSS для стилізації та JavaScript для надання динаміки та взаємодії. Також фреймворки, такі як React, Angular і Vue.js, які спрощують процес розробки та підвищують продуктивність. Для отримання даних і відправлення запитів до серверної частини, клієнтська частина використовує API (Application Programming

Interface). Зазвичай це AJAX запити або використання бібліотек, таких як Axios або Fetch, для взаємодії зі сторонніми службами.

З метою забезпечення безпеки та обмеження доступу користувачів до ресурсів важливо впроваджувати системи автентифікації та авторизації. Якщо додаток призначений для глобального ринку, то локалізація та міжнародна підтримка є необхідними, дозволяючи проводити переклад та адаптацію для різних мов і регіонів. Тестування та налагодження клієнтської частини є важливим етапом, спрямованим на виявлення помилок та підтвердження її правильної роботи на різних браузерах та пристроях.

Серверна частина веб-додатку відповідає за обробку запитів, обчислення, збереження та надання даних клієнтській частині.

Вибір серверів та хостингу для розгортання серверної частини додатку є важливою та вирішальною задачею. Це може включати в себе вибір між фізичним сервером та хмарними платформами, такими як AWS, Azure чи Google Cloud.

Створення та налаштування бази даних також важливий етап розробки. Це може бути SQL база даних, така як MySQL або PostgreSQL, або MongoDB. Для взаємодії з клієнтською частиною необхідно розробити API та впровадити логіку бізнес-процесів, включаючи обробку запитів та взаємодію з базою даних. Забезпечення безпеки серверної частини включає захист від різних атак, таких як SQL-ін'єкції та міжсайтове сценаріювання (XSS). Масштабування серверної частини планується і реалізується для забезпечення високої доступності та продуктивності.

Збереження даних та регулярне створення резервних копій є важливим елементом надійного зберігання і запобігання втратам інформації. Тестування серверної частини виконується для виявлення помилок та оптимізації продуктивності. Моніторинг роботи серверної частини, а також ведення журналу подій, допомагають вчасно виявляти проблеми та забезпечують дані для подальшого аналізу.

Клієнтська та серверна частини взаємодіють між собою через мережу Інтернет, надсилаючи запити та отримуючи відповіді. Успішний веб-додаток вимагає добре

спроектованих і функціонуючих клієнтської та серверної частин, які спільно забезпечують користувачам зручний і безпечний досвід.

1.7. Аналіз фреймворків для клієнтської частини веб-додатку.

Фреймворк - це структура або платформа для розробки програмного забезпечення, яка надає базовий функціонал та архітектурний каркас для роботи. Фреймворк може включати в себе набір бібліотек, стандартів, конвенцій та інших ресурсів, які допомагають розробникам створювати програми швидше та ефективніше.Перед початком вибору фреймворку для веб-додатку необхідно провести детальний аналіз вимог до проекту. Це включає в себе визначення функціональних, нефункціональних та технічних вимог до додатку. Функціональні вимоги описують, які функції повинен виконувати додаток, нефункціональні вимоги - вимоги до продуктивності, безпеки, масштабованості та інші характеристики. Технічні вимоги визначають, які технології, мови програмування та бази даних повинні використовуватися. Для ефективного вибору фреймворку необхідно визначити набір критеріїв, за якими будуть оцінюватися доступні варіанти. Для цього можна використовувати наступні критерії: продуктивність, масштабованість, безпека, функціональність. На сьогоднішній день існує багато популярних фреймворків та бібліотек для розробки клієнтської сторони веб-додатків. Найпопулярніші з них є React, Vue, Angular.

React, також відомий як React JS, є бібліотекою JavaScript для створення користувацьких інтерфейсів, яка розробляється і підтримується META (Facebook Inc). Головна мета React - спростити створення інтерфейсів шляхом розділення сторінок на окремі фрагменти, які часто називаються вебкомпонентами. Вона дозволяє змінювати дані на сторінці без її повного перезавантаження. React використовується для розробки як мобільних додатків, так і односторінкових веб-застосунків, які базуються на одному документі HTML. Важливо відзначити, що React не є повним MVC-фреймворком, але бібліотекою з відкритим вихідним кодом, яка використовується для рендерингу представлень та має власну архітектуру. React

дозволяє розробникам вибирати структуру веб-застосунку, але він пропонує компонентний підхід для розробки. Також важливо розрізняти між вебкомпонентами та React. Вебкомпоненти розроблені для інкапсуляції функціональності, тоді як React створений для синхронізації об'єктної моделі документа DOM з даними веб-застосунку. Обидва підходи можуть співіснувати, і React може бути використаний у вебкомпонентах або для їх створення, що робить його потужним інструментом для розробки веб-застосунків.

Vue.js останнім часом набуває все більшої популярності серед веб-розробників. Цей інструмент може використовуватися як фреймворк або бібліотека функцій в залежності від потреб вашого веб-застосунку. Він вражає швидкістю та продуктивністю і є ідеальним вибором для створення інтерфейсів користувача та односторінкових застосунків. Vue.js складається з основної бібліотеки, яка в основному спрямована на рівень представлення, та набору допоміжних бібліотек, що розширюють його можливості. Цей фреймворк також використовує систему компонентів, що вносить певну подібність з React. Архітектура Vue.js ґрунтується на шаблоні MVVM, проте не реалізує його повністю, фокусуючись переважно на моделі та ViewModel. Розробники віддають перевагу Vue.js завдяки його гнучкості та легкості вивчення.

Angular розроблено як покращена версія AngularJS, що був розроблений командою Google. Після того, як його архітектура була перероблена з нуля, Angular став окремим фреймворком, який набув значного популярності. Цей перехід супроводжувався виправленням численних помилок та вразливостей безпеки, а також істотними змінами в структурі. Angular написаний на TypeScript і вимагає використання цієї мови під час написання коду. На відміну від своєї попередньої версії, AngularJS, він не є повністю фреймворком MVC, але базується на компонентному підході. Кожен компонент, створений під час розробки із врахуванням всіх необхідних параметрів, можна розглядати як окрему реалізацію шаблону MVC, оскільки він включає залежності, деталі представлення і оголошення класу з відповідною логікою, що може бути розглянуто як контролер. Розділення представлення та логіки дозволяє спростити структуру та розмітку. У фреймворку

доступно багато інструментів для розробки веб-застосунків, але це може збільшити складність вивчення. Angular підтримує розробку для різних платформ, включаючи десктопні та мобільні додатки. Завдяки великій спільноті розробників, Angular продовжує активно розвиватися та удосконалюватися.

1.8. Аналіз інструментів для серверної частини веб-додатку.

Вибір мови програмування для серверної частини додатку є ключовим етапом в розробці, і його важливість визначається рядом факторів. Ефективність, масштабованість, швидкість розробки та підтримка – це всі аспекти, які варто враховувати при виборі мови програмування.

Перш за все, ефективність виконання завдань тісно пов'язана з характером самого додатку. Різні мови мають свої особливості і переваги в різних сценаріях використання, починаючи від веб-розробки та закінчуючи обробкою великих обсягів даних.

Масштабованість грає важливу роль у виборі мови програмування, оскільки необхідно обирати ту, яка забезпечить зручну можливість розширення функціональності та обслуговування зростаючої кількості користувачів чи об'єму даних. Швидкість розробки та підтримка також важливі, оскільки вони впливають на терміни випуску та ефективність обслуговування кодовой бази у майбутньому.

Ось коротка характеристика деяких мов програмування для серверної частини додатку:

JavaScript (Node.js) відзначається зручністю для веб-розробки, орієнтованою на події та асинхронною, і часто використовується для створення легких та швидких веб-серверів.

Python, завдяки своїй простоті та читабельності, знаходить застосування в різноманітних завданнях, включаючи веб-розробку, і підтримує різні фреймворки, такі як Django та Flask.

Ruby, яка є простою та елегантною, основним чином використовується з фреймворком Ruby on Rails для швидкого розгортання веб-додатків.

Java, масштабована та надійна, застосовується головним чином в корпоративному середовищі для великих проектів.

C# (ASP.NET), інтегрована з платформою Microsoft, є зручною для розробки під Windows-сервери та корпоративні додатки.

Go, ефективна та швидка, використовується для створення високоефективних веб-серверів.

PHP, широко використовується в веб-розробці та взаємодіє з різними базами даних, такими як MySQL та PostgreSQL.

Кожна мова має свої сильні та слабкі сторони, і вибір залежить від конкретних вимог проекту, навичок розробників та його масштабу.

Фреймворки допомагають спростити розробку, організуючи базові структури та надаючи готові інструменти. Деякі популярні серверні фреймворки включають Express.js (для Node.js), Django (для Python), Ruby on Rails (для Ruby), Spring (для Java) та Laravel (для PHP).

Розглянемо кожен з цих фреймворків:

Express.js є легким та гнучким фреймворком для розробки веб-додатків на мові програмування JavaScript, спеціально створеним для середовища Node.js. Легкість використання робить Express мінімалістичним та добре організованим фреймворком, що спрощує створення серверних додатків, не нав'язуючи багато власних правил і дозволяючи розробникам вибирати інструменти та архітектурні підходи. Рутинізація маршрутів забезпечує потужні засоби для визначення маршрутів (routes), спрощуючи обробку різних запитів і визначення поведінки додатку.

Middlewares в Express дозволяють виконувати певний код перед обробкою запиту або відправленням відповіді, забезпечуючи модульність та розширюваність додатків. Є вбудована підтримка обробки статичних файлів, таких як зображення, CSS та JavaScript, що сприяє ефективній роботі з ресурсами клієнта. Щодо шаблонізаторів (template engines), таких як EJS та Pug (раніше відомих як Jade), Express дозволяє легко виводити HTML на стороні сервера.

Розширюваність фреймворка відбувається за рахунок великої кількості сторонніх модулів та middleware. Express має активну спільноту та постійно

оновлюється, роблячи його популярним вибором для розробників. Express.js вважається одним із найпопулярніших та ефективних фреймворків для створення серверних додатків на Node.js, і він знаходить застосування в широкому спектрі веб-проектів.

Django - це високорівневий фреймворк для розробки веб-додатків на мові програмування Python. Він вирізняється своєю ефективністю, швидкістю розробки та забезпечує великий набір готових компонентів для побудови повнофункціональних веб-застосунків. Django використовує архітектурний шаблон Модель-Вид-Шаблон (MVC), що дозволяє ефективно розділити логіку додатку, представлення та обробку даних. Також, фреймворк надає вбудований ORM для спрощення взаємодії з базою даних. Однією з ключових особливостей є вбудований інтерфейс адміністратора, який автоматично генерується на основі моделей додатку, спрощуючи управління адміністративною частиною веб-застосунків.

Для роботи з фронтендом використовується система шаблонів та вбудований двигун шаблонів Django, що дозволяє легко створювати динамічний HTML-контент. Також, Django вбудовує різноманітні заходи безпеки, такі як захист від атак на основі SQL-ін'єкцій, міжсайтового сценаріювання (XSS) та CSRF (Cross-Site Request Forgery). Завдяки великій кількості готових компонентів та автоматичному створенню адміністративної панелі, Django сприяє швидкій розробці веб-додатків. Фреймворк також легко розширюється за допомогою сторонніх пакетів та доповнень. Загалом, Django є потужним інструментом для розгортання веб-додатків, використовується в різних проектах і користується великою та активною спільнотою розробників.

Ruby on Rails, який часто називають просто Rails, є високорівневим фреймворком для розробки веб-додатків на мові програмування Ruby. Заснований на парадигмі "Конвенція над конфігурацією", він спрямований на прискорення процесу розробки та забезпечення стандартів у структурі проекту.

Один із ключових елементів - активний запис (Active Record), спрощений шар роботи з базою даних, що дозволяє взаємодіяти з нею за допомогою об'єктно-орієнтованого коду. Визначальною рисою є концепція "Конвенція над

конфігурацією", яка полегшує життя розробникам завдяки автоматизованим налаштуванням та стандартам у фреймворку. Також, динамічні RESTful маршрути, які дозволяють ефективно управляти шляхами запитів та ресурсами. Rails також включає у себе команду "scaffold", яка автоматично створює базовий код для моделі, контролера та виду для обраних ресурсів. Засоби автоматизації тестування вбудовані для створення тестів та забезпечення стабільності додатку.

Для розширення функціональності використовуються "геми" (gems) - розширення та пакети, які полегшують інтеграцію з різними засобами та сервісами. Засоби безпеки вбудовані, а підтримка AJAX спрощує реалізацію динамічної взаємодії на сторінках. Усе це робить Ruby on Rails потужним інструментом для швидкої та ефективної розробки веб-додатків.

Spring - це високорівневий фреймворк для розробки Java-додатків, призначений для створення ефективних, масштабованих та підтримуваних застосунків. Він використовує принцип інверсії управління, де контейнер Spring керує об'єктами життєвим циклом та взаємозв'язками.

Фреймворк надає управління бобами (Bean Management), де контейнер відповідає за створення, конфігурацію та знищення об'єктів. Spring підтримує аспектно-орієнтоване програмування, що дозволяє визначати аспекти (перерізи функціональності) та впроваджувати їх в код додатку. Контекст додатка Spring надає середовище для конфігурації та управління компонентами, а також для внедрення залежностей. Spring Boot, розширення Spring, дозволяє швидко створювати готові до виробництва додатки з мінімальними налаштуваннями. Фреймворк також пропонує модель програмування за допомогою анотацій, що спрощує роботу з різними компонентами.

Spring забезпечує підтримку транзакцій та легко інтегрується з іншими технологіями, такими як Hibernate або Spring Security. Він орієнтований на модульність, а офіційна документація є добре структурованою та оновлюється. Spring є потужним інструментом для розробки Java-додатків, і його екосистема пропонує різні інструменти для вирішення різноманітних вимог проектів.

Laravel - це високорівневий фреймворк для розробки веб-додатків на мові програмування PHP. Заснований на архітектурному шаблоні MVC, Laravel пропонує елегантний синтаксис та широкий спектр інструментів для швидкої та продуктивної розробки. Однією з ключових особливостей є Artisan, консольна команда Laravel, яка автоматизує рутинні завдання. Вбудований ORM Eloquent дозволяє взаємодіяти з базою даних за допомогою зрозумілого синтаксису мови програмування.

Laravel пропонує механізм міграцій для управління структурою бази даних, а Blade - простий та ефективний шаблонізатор для створення виглядів. Middleware використовується для обробки HTTP-запитів, дозволяючи впроваджувати логіку до обробників запитів. Цей фреймворк включає інструменти для безпеки, такі як CSRF-захист та хешування паролів. Laravel є потужним інструментом для ефективної розробки PHP-додатків.

1.9. Безпека веб-додатку.

Основні загрози для веб-додатку.

Безпека веб-додатків є надзвичайно важливою та актуальною проблемою в сучасному інформаційному суспільстві. Веб-додатки стали невід'ємною частиною життя, використовуються для різних цілей, включаючи електронну комунікацію, фінансові операції, зберігання особистих даних та багато іншого.

Однак разом зі зростанням популярності веб-додатків збільшується і кількість загроз, які вони стикаються. Забезпечення безпеки веб-додатків має вирішальне значення. Тому що веб-додатки зберігають та обробляють конфіденційну інформацію, таку як особисті дані користувачів, фінансові дані, медичні записи тощо. Якщо ця інформація потрапить до рук зловмисників, то можуть виникнути серйозні наслідки для приватності та безпеки осіб, які її стосується.

Також зловмисники можуть намагатися модифікувати дані веб-додатку або вплинути на його функціональність. Це може призвести до втрати даних або спотворення інформації, що може завдати збитків як користувачам, так і організації.

Щоб забезпечити безпеку веб-додатків потрібно ознайомитися з основними загрозами для веб додатків:

- SQL-ін'єкція (SQL Injection) - це атака, під час якої зловмисник вставляє SQL-код у веб-додаток через ввідні поля, що може призвести до витоку, видалення або модифікації даних у базі даних. Наприклад, зловмисник може отримати доступ до конфіденційних даних користувачів або витягнути дані про кредитні картки.
- Кросс-сайт скриптинг (Cross-Site Scripting, XSS) - це атака, при якій зловмисник вставляє зловісний JavaScript-код у веб-додаток через ввідні дані, які потім виконуються у браузері іншого користувача. Це може призвести до викрадення сесійних файлів, зміни вмісту сторінки або зловживання даними користувачів.
- Кросс-сайт реквест піддавання (Cross-Site Request Forgery, CSRF) CSRF - це атака, під час якої зловмисник використовує авторизований сеанс користувача для виконання дій без його згоди. Наприклад, зловмисник може виконати операції від імені користувача, якщо він увійшов до системи.

Забезпечення безпеки веб-додатків є надзвичайно важливим завданням у сучасному інтернет-середовищі. Розробники та адміністратори повинні бути усвідомлені щодо потенційних загроз та приймати відповідні заходи для захисту веб-додатків від атак та збереження конфіденційності, цілісності та доступності даних.

Методи запобігання SQL-ін'єкціям у веб-додатку.

Некоректна перевірка вхідних даних є найбільш поширеною слабкістю безпеки веб-застосунків. Для того, щоб захистити веб-додаток від SQL-ін'єкцій є методи та практики забезпечення безпеки. Першим і найважливішим кроком у запобіганні SQL-ін'єкціям є валідація та фільтрація вхідних даних.

Валідація полягає в перевірці вхідних даних на відповідність очікуваним форматам та типам. Фільтрація включає в себе видалення або заміну небезпечних символів у вхідних даних. Використання параметризованих запитів є ефективним методом запобігання SQL-ін'єкціям. Замість вставки вхідних даних безпосередньо в SQL-

запити, параметризовані запити використовують параметри, які передаються окремо в запиті і автоматично захищаються від SQL-ін'єкцій. Об'єктно-реляційне відображення (ORM) дозволяє розробникам працювати з базами даних, використовуючи об'єктно-орієнтований підхід, а не писати SQL-запити вручну. ORM автоматично генерує і обробляє SQL-запити, що робить SQL-ін'єкції неможливими.

Ескейпінг спеціальних символів, таких як одинарні лапки та двокрапки. Це дозволяє базі даних розрізняти між введеним текстом і SQL-кодом. Всі ці методи та практики дозволять вчасно виявляти вирішувати можливі атаки SQL-ін'єкції, забезпечуючи надійність та захищеність веб-додатків.

Методи запобігання Кросс-сайт скриптингу (XSS) у веб-додатку.

Не менш поширеною загрозою безпеці веб-застосунку є XSS. Першим і важливим кроком у запобіганні XSS є валідація та фільтрація вхідних даних. Валідація допомагає перевірити, чи відповідають вхідні дані очікуваному формату (наприклад, електронна адреса, URL). Фільтрація дозволяє вилучити небезпечні символи із вхідних даних, що може завадити виконанню шкідливих скриптів. Екранування виведення даних (Escaping), цей метод означає що всі дані, які виводяться на сторінку, повинні бути перетворені у безпечний формат, який не може бути використаний для виконання JavaScript-коду. Наприклад, замість прямого вставлення даних користувача в HTML, їх можна перетворити у HTML-сутності. Використання HTTP-заголовків безпеки, такі як Content Security Policy (CSP) і HTTP Strict Transport Security (HSTS), дозволяють встановлювати правила, які обмежують виконання скриптів на сторінці та забороняють використання незахищеного з'єднання. Вони зменшують ризик XSS-атак і допомагають забезпечити безпеку веб-додатку. Використання бібліотек і фреймворків, які мають вбудовану підтримку для запобігання XSS, є розумним підходом. Ці інструменти зазвичай мають функції екранування та інші безпекові механізми, які допомагають уникнути XSS-атак.

Запобігання XSS-атакам вимагає комплексного підходу та використання різних методів забезпечення безпеки. Тому правильна реалізація цих заходів допоможе забезпечити надійну захист веб-додатків від XSS-атак і зберегти безпеку користувачів та їх даних.

Методи запобігання Кросс-сайт реквест піддавання (CSRF) у веб-додатку.

Кросс-сайт реквест піддавання (CSRF) є однією з важливих загроз безпеці веб-додатків. Щоб забезпечити безпеку веб-додатку від CSRF використовуються специфікацію SameSite для файлів cookie, подвійну перевірку куки та перевірку походження запиту. Специфікація SameSite для файлів cookie - це один з методів, який допомагає запобігти атакам CSRF. Завдяки цій специфікації, веб-розробники можуть встановити атрибут "SameSite" для файлів cookie, які визначають, як ці файли cookie повинні взаємодіяти з вебсайтами з іншого джерела. Подвійна перевірка куки (Double-Submit Cookie) при цьому методі веб-додаток генерує унікальний токен (токен CSRF) і включає його як файл cookie та як приховане поле в форму на сторінці. При подачі форми на сервер користувач повинен включити цей токен у запит, і він порівнюється з токеном, збереженим в файлі cookie. Якщо токени відповідають один одному, то запит вважається припустимим. Зловмисники не можуть отримати доступ до токену, оскільки він знаходиться як файл cookie на браузері користувача і не може бути змінений з іншого джерела через обмеження схеми SameSite. Перевірка походження запиту (Origin Checking) цей метод перевіряти заголовок "Origin" запиту та порівнювати його з очікуваним джерелом запиту. Якщо походження запиту не відповідає очікуваному, то запит може бути відхилений. Комбінування специфікації SameSite для файлів cookie, подвійної перевірки куки, перевірки походження запиту може значно зменшити ризик атак CSRF і покращити безпеку веб-сервісу та його користувачів.

Висновки до розділу 1

У розділі дипломної роботи, присвяченому веб-додаткам, проведено ретельний аналіз ключових аспектів їх розробки та функціонування. Здійснено детальний розгляд переваг та недоліків веб-додатків, розкрито їх відмінність від веб-сайтів та виокремлено основні види цих додатків.

Особлива увага приділена важливому питанню – вибору оптимальної архітектури для веб-додатку. Для цього були розглянуті концепції архітектурних шаблонів, таких як MVC, MVP, а також вивчені особливості веб-компонентів. Крім того, детально розглянуто процес створення веб-додатків, включаючи аналіз фреймворків для клієнтської та серверної частини.

Окремий блок вивчення присвячено питанню безпеки веб-додатків, де висвітлено основні принципи та заходи для встановлення безпеки, необхідні для збереження надійності та конфіденційності інформації.

Отримані в ході дослідження знання і висновки дозволяють визначити оптимальний підхід до розробки та управління веб-додатками.

РОЗДІЛ 2. АНАЛІЗ ВИКОРИСТАНИХ ТЕХНОЛОГІЙ В РОЗРОБЦІ ВЕБ-СЕРВІСУ

2.1. Стек технологій MERN.

Сучасна реальність веб-розробки вимагає від розробників надійних та ефективних інструментів для створення інноваційних веб-додатків. MERN, або стек технологій MERN, став невід'ємною частиною цієї екосистеми, пропонуючи інтегрований пакет інструментів, який відкриває безмежні можливості для розробників. Завдяки своїй простій архітектурі, високій ефективності та активній спільноті, MERN стає ключовим інструментом для тих, хто прагне створити високоякісні веб-додатки, готові відповідати сучасним та майбутнім викликам технологій.

MERN не просто надає розробникам технічний арсенал, але втілює філософію розробки, що поєднує в собі ефективність, гнучкість та інновації. На підставі простоти архітектури MERN, розробники можуть швидко та ефективно створювати високопродуктивні та інтерактивні веб-додатки, що відповідають найвищим стандартам у світі веб-програмування.

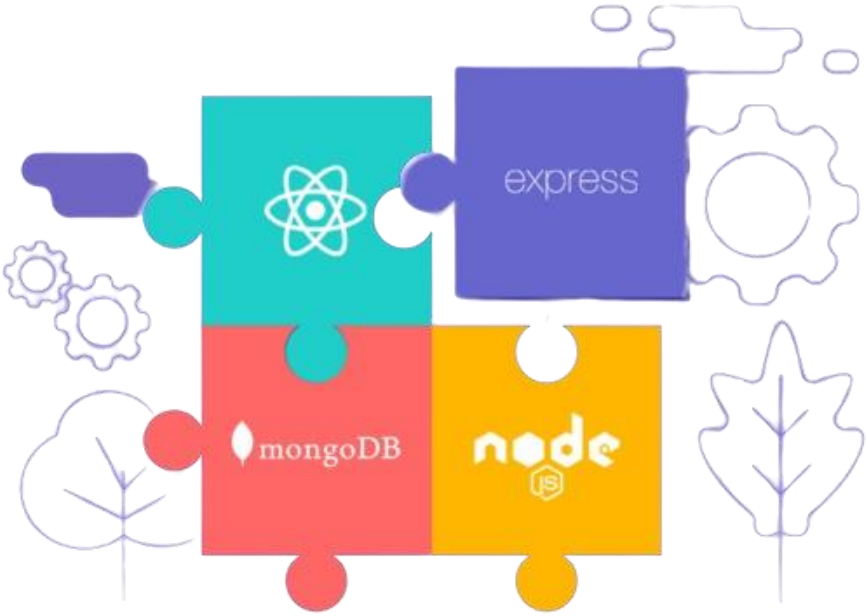

Рисунок 1.3. Стек технологій MERN

Використовуючи цей стек, вони отримують доступ до різноманітних інструментів для оптимізації робочого процесу та підтримки ефективної взаємодії з серверною частиною, базою даних та динамічним управлінням станом клієнтської частини. Однією з найактуальніших переваг MERN є його здатність впроваджувати нові функції швидко та безпроблемно, що стає важливою перевагою в умовах швидко змінюючогося веб-середовища.

Від зручності роботи з кодом до швидкості впровадження нових функцій, MERN став важливим партнером для тих, хто бажає створювати інноваційні та конкурентоздатні веб-додатки. Наголос на актуальних інструментах та можливостях MERN робить його важливою технологією для сучасного веб-розробника, що має на меті високоякісний та швидкий розвиток веб-додатків.

MERN - це абревіатура, яка вказує на стек технологій, що використовується для розробки веб-додатків. MERN складається з наступних частин:

- M - MongoDB: NoSQL база даних, яка зберігає дані у форматі BSON (бінарний JSON). MongoDB використовується для збереження документів та JSON-подібних даних;

- E - Express.js: Веб-фреймворк для Node.js, який використовується для розробки серверної частини додатку. Він допомагає створювати API та обробляти HTTP-запити;
- R - React: Бібліотека для створення користувацьких інтерфейсів. React використовується для розробки клієнтської частини додатку, де створюються інтерфейси та відображаються дані;
- N - Node.js: Заснована на JavaScript середовище виконання, яке використовується для роботи на серверній стороні. Node.js дозволяє виконувати JavaScript на стороні сервера.

Історія створення MERN:

MERN став популярним стеком для розробки завдяки поєднанню цих технологій. React був розроблений Facebook, Express.js - як фреймворк для Node.js, завдяки своїй простоті та широкій підтримці, став популярним серед розробників. MongoDB, в свою чергу, став вибором для багатьох завдяки своїй гнучкості та можливості зберігати дані у форматі, який ближчий до JSON.

У цьому контексті, давайте розглянемо основні переваги та можливості цього захоплюючого технологічного стеку.

Плюси MERN:

- Швидкість розробки: MERN дозволяє розробникам швидко створювати веб-додатки завдяки готовим компонентам та бібліотекам, які спрощують роботу;
- Універсальність: React може використовуватися як для створення клієнтських веб-додатків, так і для розробки мобільних додатків;
- Масштабованість: MongoDB дозволяє легко масштабувати базу даних, а Node.js може обробляти багато одночасних запитів;
- JavaScript: Всі частини MERN використовують JavaScript, що спрощує розробку та підтримку коду.

Мінуси MERN:

- Відсутність вбудованого ORM (Object-Relational Mapping): MongoDB не має вбудованого ORM, що може ускладнити взаємодію з базою даних для деяких розробників;
- Брак конвенції у розробці: MERN не накладає жодних конвенцій розробки, тому команди повинні самостійно визначити структуру проекту та найкращі практики;
- Наявність JavaScript на сервері: Хоча Node.js має свої переваги, він може не бути найкращим вибором для всіх випадків, особливо коли розробляється додаток з великою обчислювальною навантаженістю.

Загалом MERN є потужним та популярним стеком для розробки веб-додатків, але він може не підходити для всіх видів проектів та вимагати додаткових знань і рішень у певних випадках.

2.1.1. Node.js.

Node.js — це не просто середовище виконання JavaScript на серверному боці, це технологія, яка революціонізує підхід до розробки серверних додатків. Започаткований з метою оптимізації виконання JavaScript-коду на сервері, Node.js став відкритим вікном у світ ефективного, швидкого та масштабованого програмування. Дозвольте вам зануритися в унікальність Node.js, де асинхронність та швидкодія грають роль у створенні потужних серверних додатків. У цьому світі, JavaScript виходить за межі браузера, стаючи ключовим інструментом для побудови високопродуктивних, надійних та ефективних серверів. Давайте розглянемо, як Node.js переписує правила гри у сфері серверного програмування та чому він заслуговує на увагу кожного розробника.

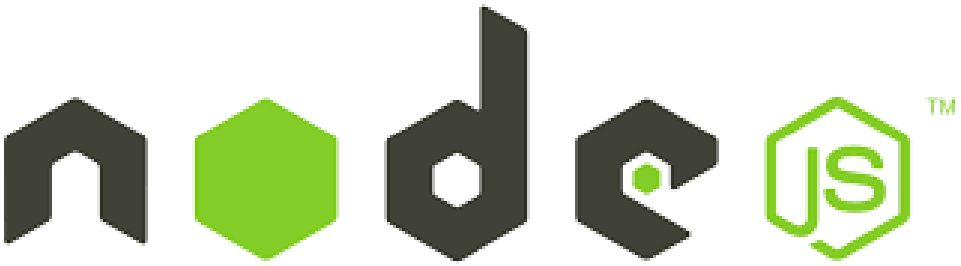

Рисунок 1.4. Середовище виконання Node.js

Головною метою Node.js є забезпечення можливості розробки масштабованих мережевих застосунків та оптимізація обробки багатоопераційних введень-виведень. Він вирізняється своєю асинхронною та подієвою моделлю, яка дозволяє ефективно обробляти багато операцій, не блокуючи основний потік виконання програми.

Node.js використовує асинхронний підхід, щоб уможливити виконання операцій паралельно та забезпечити високу продуктивність, що особливо корисно при обробці багатьох запитів одночасно. Вбудований пакетний менеджер npm робить використання та обмін модулями серед розробників простішими, сприяючи швидкому вибору інструментів для розробки. Використання рушія V8 від Google гарантує високу швидкість виконання коду JavaScript, що робить Node.js одним з найефективніших виконавців.

Node.js знайшов широке застосування в різних галузях технологій та розробки. Node.js є відмінним вибором для розробки веб-серверів та веб-додатків. Здатність обробляти багато одночасних з'єднань робить його ідеальним для серверної розробки, де важлива висока продуктивність та швидка відповідь на запити. Завдяки своїй легкості та ефективності, Node.js використовується для створення програм для пристроїв Інтернет речей (IoT). Він може обробляти дані з різних сенсорів та пристроїв, забезпечуючи зручну розробку для цього сегмента. Node.js служить основою для багатьох інструментів та фреймворків розробки, таких як Express.js та Nest.js. Його модульність та легкість розширення роблять його привабливим для

розробників, які шукають гнучкість у виборі інструментів для своїх проектів. Асинхронні можливості Node.js роблять його ідеальним для створення систем реального часу, таких як чати та гіперпотужні застосунки, де необхідно миттєве оброблення та передача даних. Використання Node.js у мікросервісній архітектурі дозволяє створювати ефективні та легко масштабовані сервіси, що особливо важливо в сучасних розробках програмного забезпечення. Node.js використовується для взаємодії з різними базами даних, такими як MongoDB, MySQL та інші, забезпечуючи зручний інтерфейс для розробників. Розробники можуть використовувати Node.js для створення власних інструментів розробки завдяки його модульності та широкому спектру пакетів, які допомагають у рутинних завданнях. Node.js може бути використаний для створення хмарних служб та застосунків, що взаємодіють з іншими компонентами хмарних рішень, надаючи розробникам можливість створювати розподілені системи зручно та ефективно. Давайте також розглянемо переваги та недоліки Node.js:

Переваги:

- Ефективність: Висока швидкість виконання завдяки рушію V8;
- Модульність: Легкість у використанні та розширенні завдяки пакетному менеджеру npm;
- Асинхронність: Забезпечує ефективну обробку багатоопераційних введень-виведень.

Недоліки:

- Однопоточність: Відсутність багатопоточності в стандартному розумінні;
- Втрата часу під час важких обчислень: Може бути неефективним для важких обчислень через однопоточний характер.

2.1.2. Express.js.

Express.js - це веб-фреймворк, призначений для розробки серверних додатків на Node.js. Запущений у 2010 році, Express став популярним завдяки своїй простоті, швидкості та можливості легкого розширення. Цей фреймворк надає базовий набір

інструментів для створення веб-сайтів і додатків, дозволяючи розробникам швидко розпочати свої проекти та легко розширювати їх функціонал.

Рисунок 1.5. Фреймворк Express.js

Головна ідея Express полягає в тому, щоб надати розробникам високий рівень вільності в тому, як вони хочуть будувати свої серверні застосунки. Фреймворк надає лише основні інструменти для маршрутизації, обробки запитів та управління видами відповідей. Це робить його ідеальним вибором для тих, хто шукає простий та ефективний інструмент для розробки веб-застосунків.

Простота та мінімалізм - це основна філософія Express, яка полягає в тому, щоб надавати лише необхідні функції для швидкої розробки веб-додатків, дозволяючи розробникам ефективно створювати та налаштовувати проекти. Маршрутизація вражає своєю потужністю, спрощуючи обробку різноманітних запитів та визначення необхідних дій для кожного типу запиту. Middleware представляє собою ключовий елемент Express, що дозволяє розробникам використовувати функції, які мають доступ до об'єктів запиту та відповіді, виконуючи певні операції перед передачею управління маршрутам. Шаблонізатори в Express підтримують використання різноманітних шаблонізаторів, полегшуючи створення сторінок та представлень, таких як EJS, Pug і Handlebars. Статична обробка файлів, включаючи зображення та скрипти клієнта, також є зручною функціональністю Express. Middleware та обробка запитів включають можливість легко визначати, як програма повинна реагувати на

різні типи запитів HTTP, а розробники можуть створювати власні middleware для виконання специфічних операцій. Express спрощує роботу з базами даних, підтримуючи легку інтеграцію з різними системами управління базами даних та використання ORM, як от Mongoose для MongoDB.

Express.js взаємодіє з Node.js у декількох основних аспектах. Express використовує можливості Node.js для роботи з протоколом HTTP, приймання та обробки HTTP-запитів, а також відправки HTTP-відповідей. Як фреймворк, він працює в асинхронному та нестоповому режимі, що дозволяє ефективно обробляти багато одночасних запитів. Ключовий компонент Express - це middleware, який використовується для обробки запитів та викликається послідовно для їхньої обробки. Концепція middleware ідеально вписується в асинхронну природу Node.js. Express може використовувати модульну систему Node.js для підключення різних модулів та компонентів, використовуючи, наприклад, require. Окрім цього, Express взаємодіє з HTTP методами, такими як GET, POST, PUT, DELETE, щоб взаємодіяти з різними типами HTTP-запитів.

Також цей фреймворк має свої мінуси:

Мінуси Express.js:

- Ручне Управління Компонентами: Так як Express є мінімалістичним, іноді розробнику доводиться вручну керувати компонентами, що може бути завданням для великих проектів;
- Відсутність Стандартів: У порівнянні з іншими фреймворками, Express не накладає чітких стандартів структури проекту, що може вести до розбіжностей у розробці;
- Брак Вбудованих Засобів: У порівнянні з іншими фреймворками, Express може виглядати обмеженим у вбудованих інструментах для деяких завдань, що вимагають додаткових бібліотек;
- Недостатня Стандартизація Middleware: У випадку деяких middleware може виникати проблема недостатньої стандартизації, що призводить до варіативності в їхній якості та функціоналі;

- Асинхронність: Для новачків може бути складно зрозуміти асинхронний підхід Node.js та Express, що може викликати труднощі в розробці;
- Відсутність Вбудованих Засобів для Реактивного Програмування: В порівнянні з іншими фреймворками, Express не має вбудованих інструментів для реактивного програмування, що може бути важливим для деяких проектів.

2.1.3. MongoDB.

Зберігання даних користувачів в базі даних стає важливою задачею, і для досягнення цієї мети використання MongoDB разом із вибраним серверним середовищем виявиться оптимальним рішенням. MongoDB - це NoSQL документно-орієнтована база даних, спроектована для зберігання обширних обсягів інформації. На відміну від традиційних реляційних баз даних, які використовують таблиці та рядки, MongoDB оперує з колекціями та документами. Документи представляють собою пари ключ-значення, що становлять основну одиницю даних в MongoDB. Колекції, у свою чергу, містять набори документів та функціонують як еквівалент реляційних таблиць. MongoDB у 2009 році і з того часу вона стала визнаним рішенням для зберігання та управління великими обсягами різноманітних даних.

Рисунок 1.6. База даних MongoDB

Однією з ключових рис MongoDB є відсутність фіксованої схеми, що робить її ідеальним вибором для проектів, де структура даних може часто змінюватися. Це

полегшує розробку, оскільки розробники можуть працювати з даними у вигляді об'єктів JSON.

Ця база даних також відома своєю здатністю ефективно масштабуватися горизонтально, тобто легко адаптується до зростання обсягів даних та навантаження, розподіляючи їх на кілька серверів.

У сучасному технологічному світі MongoDB застосовується в різних областях, включаючи веб-розробку, аналітику даних та системи керування контентом. Її розширена функціональність, швидкість та гнучкість роблять її ключовим інструментом для розробників та архітекторів даних у сучасному програмуванні.

Додаткові особливості MongoBD:

- У більш пізніх версіях MongoDB додана підтримка транзакцій, що робить її більш придатною для застосунків, де важлива консистентність даних;
- MongoDB надає потужні інструменти для агрегації даних та виконання операцій Map-Reduce, що дозволяє виконувати складні операції обробки та аналізу даних.

2.1.4. React.

React був розроблений командою інженерів у Facebook, під керівництвом Джордана Вальке та Джо Ревса. Почавши як внутрішній інструмент для Facebook у 2011 році, React зробив вражаючий шлях відкритого програмного забезпечення.

Рисунок 1.7. Бібліотека React

Однією з ключових мет React була необхідність покращити швидкість та ефективність взаємодії з користувачем на великих веб-сайтах з великою кількістю даних. React відмінно справляється з цією задачею, дозволяючи розробникам ефективно управляти станом додатка та автоматизувати процес оновлення відображення. На сьогодні React є ключовим інструментом для фронтенд-розробників. Широкі можливості та стабільний розвиток роблять його важливим елементом для сучасної веб-розробки. Однією з ключових особливостей React є його компонентний підхід. Відмінною рисою в порівнянні з традиційним підходом до розробки, де весь інтерфейс розділяється на різні файли та елементи, є те, що React дозволяє розділяти інтерфейс на невеликі, автономні компоненти. Це сприяє легкості розробки, підтримці та повторному використанню коду.

Ще однією важливою особливістю є використання віртуального DOM (Document Object Model). React використовує свій власний віртуальний DOM, що дозволяє ефективно взаємодіяти з реальним DOM, забезпечуючи високу продуктивність та швидкість оновлення інтерфейсу.

Багатофункціональність дозволяють розробникам React вирішувати різноманітні завдання, починаючи від створення простих веб-сайтів і закінчуючи складними односторінковими додатками. React легко інтегрується з різними

інструментами та бібліотеками, роблячи його відмінним вибором для сучасної фронтенд-розробки.

Завдяки своїй зручності використання, широкому спектру можливостей та постійному розвитку, React залишається популярним серед розробників. В поєднанні з активною спільнотою він стає ключовим інструментом для творчого втілення ідей у сфері веб-розробки.

React використовує декларативний підхід, де розробник лише описує, як повинен виглядати інтерфейс у різних станах. React автоматично вирішує, як це досягти, що сприяє реактивності інтерфейсу. Компоненти, основний будівельний блок React, розділяють інтерфейс на малі, автономні модулі, полегшуючи розробку, підтримку та перевикористання коду.

Використання віртуального DOM дозволяє React ефективно взаємодіяти з реальним DOM, забезпечуючи високу продуктивність та швидкість оновлення інтерфейсу. Це особливо корисно для створення односторінкових застосунків, де весь контент завантажується один раз, а навігація відбувається без перезавантаження сторінки.

Компоненти React можуть приймати дані від своїх батьків через props, що є незмінними та визначається зовні, і також мати внутрішній стан, який може змінюватися в результаті взаємодії користувача чи інших подій. Життєвий цикл компонентів React включає методи, такі як componentDidMount, componentDidUpdate, componentWillUnmount, які викликаються на різних етапах життя компонента.

Управління формами та введенням даних у React відбувається легко за допомогою власного стану та обробників подій. Загалом, принцип роботи React полягає в створенні декларативних, компонентних інтерфейсів, які ефективно взаємодіють з віртуальним DOM для швидкого та динамічного відображення веб-додатків. React має широку підтримку та інтеграцію з багатьма іншими інструментами та бібліотеками.

Також бібліотека має свої недоліки:

Недоліки:

- Великий поріг входу: Для новачків може бути складно зрозуміти певні концепції, такі як віртуальний DOM та JSX;
- Необхідність вивчення додаткових інструментів: Для повноцінної розробки з React часто потрібно вивчати та використовувати додаткові інструменти, такі як Webpack або Babel;
- Іноді складно зіставити з іншими бібліотеками чи фреймворками: Залежно від конкретного випадку використання, може бути ситуація, коли інший фреймворк або бібліотека були б більш підходящими;
- Ініціалізація проекту: Налаштування проекту з використанням React може займати більше часу порівняно з іншими інструментами;
- Децентралізований стан: Управління станом додатку може стати складнішим, особливо великими проектами, якщо не використовувати додаткові бібліотеки для цього.

2.1.5. Next.js.

Next.js - це потужний фреймворк для розробки веб-додатків, який базується на вже відомій бібліотеці React. Він був створений командою Vercel, яка відповідала на потреби розробників у більш зручних та продуктивних інструментах для створення веб-додатків.

Next.js виник у 2016 році і відразу ж привернув увагу розробників своєю метою спростити процес розробки та оптимізації веб-додатків на основі React. Фреймворк був спроектований так, щоб полегшити впровадження принципів React у процес розробки, забезпечуючи при цьому ефективність та простоту використання.

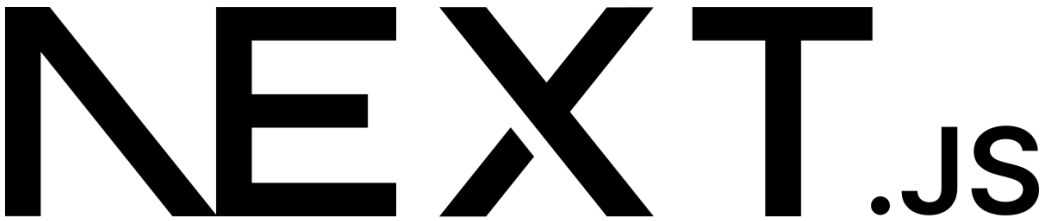

Рисунок 1.8. Фреймворк Next.js

Завдяки своїй гнучкості та інструментам для як CSR, так і SSR, Next.js дозволяє розробникам вибирати оптимальний метод рендерингу залежно від вимог їхніх проектів. Ключовою особливістю є вбудована підтримка рендерингу як на стороні клієнта (Client-Side Rendering, CSR), так і на стороні сервера (Server-Side Rendering, SSR), що робить його ідеальним вибором для розробки як статичних, так і динамічних веб-сайтів. Next.js дозволяє розробникам концентруватися на створенні функціонального та привабливого інтерфейсу, залишаючи за собою складності оптимізації та управління станом.

Багатофункціональність Next.js включає в себе автоматичну оптимізацію зображень, передвантаження сторінок, вбудовану підтримку маршрутизації. Фреймворк має вбудовану систему маршрутизації, що робить визначення маршрутів та сторінок зручним. Крім того, Next.js дозволяє генерувати як статичні, так і динамічні сторінки, що дозволяє поліпшити продуктивність та SEO. Розробники можуть використовувати компонентний підхід для розділення інтерфейсу на невеликі та автономні компоненти. Фреймворк також забезпечує захист від різноманітних атак та автоматичну оптимізацію та обробку зображень для покращення швидкості завантаження сторінок. Додатково, Next.js дозволяє гнучко налаштовувати та розширювати конфігурацію згідно з потребами проекту. Загалом, фреймворк допомагає розробникам швидко створювати високопродуктивні та ефективні веб-додатки, використовуючи сучасні підходи до розробки та оптимізації.

Давайте підсумуємо плюси та розглянемо мінуси даної технології:

Переваги:

- гнучка система рендерингу;
- вбудована маршрутизація;
- підтримка статичного та динамічного рендерингу;
- компонентний підхід;
- висока продуктивність;
- безпека;
- оптимізація зображень.

Мінуси:

- малий робочий запас інструментів;
- обмежена гнучкість конфігурації;
- велика вартість навчання спеціалістів.

Загалом, переваги Next.js переважають недоліки, але вибір фреймворку повинен залежати від конкретних потреб та ресурсів команди розробників.

2.2. Середовище розробки.

Visual Studio Code, відомий як VS Code, - це текстовий редактор, розроблений Microsoft. Вирізняється своєю легкістю використання та широким спектром можливостей, стаючи ключовим інструментом для багатьох розробників. Давайте розглянемо історію його створення та загальну інформацію.

VS Code був вперше представлений Microsoft у травні 2015 року. Завдяки своїй легкості та ефективності, редактор швидко завоював увагу користувачів своєю здатністю працювати на різних операційних системах, таких як Windows, macOS та Linux. Легкість використання визначається інтуїтивним інтерфейсом та швидкою установкою, що дозволяє швидко почати роботу.

Рисунок 1.9. Редактор початкового коду VS Code

Редактор має розширену екосистему розширень та плагінів, які полегшують інтеграцію з різними мовами програмування та технологіями. Вбудований відладчик

та підтримка системи контролю версій Git роблять зручною роботу з кодовою базою. Присутній вбудований термінал для виконання команд та завдань безпосередньо з редактора. Широкі можливості налаштувань забезпечують велику кількість параметрів для адаптації до індивідуальних потреб користувачів.

Редактор підтримує широкий спектр мов програмування, включаючи JavaScript, TypeScript, Python, Java та інші. Також, він забезпечує інтеграцію з хмарними сервісами, такими як Azure, AWS та Google Cloud.

Давайте виділимо основні переваги VS Code та її недоліки:

Переваги:

- кросплатформенність;
- легкість використання;
- розширення та плагіни;
- відладка та контроль версій;
- термінал;
- широкі можливості налаштувань.

Недоліки:

- велика кількість функцій;
- використання великих об'ємів ресурсів;
- низька швидкість роботи з великими проектами.

Висновки до розділу 2

У даному розділі дипломної роботи було виконано аналіз та комплексне вивчення стеку технологій MERN (MongoDB, Express.js, React, Node.js), а також зосереджено увагу на середовищі розробки Visual Studio Code (VS Code). Підкреслено, що вибір відповідних технологій та середовища розробки є стратегічно важливим етапом у процесі створення високопродуктивного та функціонального програмного забезпечення.

Виведені висновки вказують на те, що використання Node.js та Express.js у сполученні з MongoDB забезпечує не лише високий рівень продуктивності, але й гнучкість у розробці серверної частини веб-додатків. Розглянуто та визначено внесок реактивної бібліотеки React та фреймворку Next.js у забезпечення ефективної та швидкої роботи клієнтської сторони додатка.

Детально розглянуті можливості, які пропонує середовище розробки VS Code, підкреслюють його переваги у контексті зручності та ефективності в роботі з кодом. Враховані вбудовані інструменти для розробників та потужність можливостей розширення функціоналу, що робить VS Code обґрунтованим вибором для професійної розробки програмного забезпечення.

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБ-СЕРВІСУ ДЛЯ ПАСАЖИРСЬКИХ ПЕРЕВЕЗЕНЬ

3.1. Огляд предметної області.

3.1.1. Загальна інформація.

Пасажирські перевезення в сфері веб-сервісів пропонують різноманітні послуги для людей, які подорожують. Це включає легке онлайн-бронювання місць та замовлення транспортних послуг через веб-інтерфейс. Ви можете обрати тип транспорту і визначити оптимальний маршрут.

Важливою частиною таких сервісів є отримання актуальної інформації про розклади руху транспорту та найкращі маршрути. Технологія GPS використовується для відстеження розташування транспортних засобів у реальному часі.

Онлайн-оплата послуг та видача електронних квитків забезпечують зручність для користувачів і дозволяють уникнути фізичних білетів. Є також системи знижок для різних категорій користувачів та програми лояльності, які спрямовані на привертання та утримання клієнтів.

Безпека є однією з пріоритетних задач, і тому сервіси забезпечують відстеження маршрутів і використовують камери для забезпечення безпеки користувачів. Мобільні додатки розроблені для зручності користувачів під час подорожей і співпраці з іншими послугами, такими як готелі, ресторани та аеропорти, щоб створити повний маршрут подорожі.

3.1.2. Аналіз та порівняння аналогічних онлайн сервісів для пасажирських перевезень.

BlaBlaCar - це онлайн-платформа для карпулінгу, яка сприяє спільній поїздці водіїв і пасажирів. На цій платформі водії можуть пропонувати вільні місця в своїх автомобілях, а пасажири можуть знаходити підходящі поїздки у тому ж напрямку.

Основні аспекти BlaBlaCar включають можливість взаємодії між водіями та пасажирами, систему рейтингів і відгуків для забезпечення довіри, а також перевірку безпеки та достовірності профілів користувачів. Водії можуть встановлювати вартість поїздки, яка зазвичай є вигідною порівняно з іншими транспортними варіантами. Сервіс також дозволяє користувачам обговорювати умови подорожі та визначати деталі до від'їзду.

Завдяки широкій мережі користувачів, BlaBlaCar став популярним серед людей, які шукають економічний та екологічний спосіб подорожувати. Перед використанням сервісу рекомендується уважно вивчити умови та перевірити відгуки інших користувачів для забезпечення безпеки та задоволення від поїздки.

Розглянемо основні переваги та недоліки BlaBlaCar:

Переваги:

- Економічність. Карпулінг з BlaBlaCar може бути вигідним з фінансової точки зору, оскільки вартість поїздки зазвичай менша, ніж вартість інших транспортних послуг чи повноцінного квитка на громадський транспорт;
- Екологічна турбота: Карпулінг допомагає зменшити кількість автомобілів на дорогах, сприяючи зменшенню викидів CO2 і зниженню негативного впливу на навколишнє середовище;
- Гнучкість і співпраця: Користувачі можуть обговорювати умови подорожі, включаючи маршрут, графік і вартість. Це робить сервіс гнучким та придатним для різних потреб;
- Можливість завести нових знайомих: Під час подорожі користувачі можуть познайомитися з новими людьми, що робить досвід більш соціальним і цікавим.

Недоліки:

- Необхідність довіряти незнайомцям: Для пасажирів і водіїв існує елемент ризику, оскільки вони подорожують чи приймають незнайомців у свої автомобілі;
- Не завжди доступно: Сервіс може бути менш доступним у деяких регіонах або в певні часи, що може ускладнити знаходження підходящих поїздок;
- Нестійкість планів: Зміни у планах або скасування поїздки можуть виникнути в будь-який момент, що може створити незручності для інших користувачів;
- Обмежені можливості для вантажів: BlaBlaCar зазвичай спрямований на пасажирські поїздки, і він може бути менш практичним для тих, хто шукає послуги перевезення вантажів;
- Залежність від інтернет-з'єднання: Використання платформи вимагає наявності інтернет-з'єднання, що може бути проблематичним в деяких ситуаціях або регіонах з обмеженим покриттям.

Busfor - це відмінна платформа для онлайн-бронювання автобусних квитків. Тут можна легко шукати доступні маршрути, порівнювати ціни і миттєво забронювати квитки.

Сервіс надає докладну інформацію про маршрути, включаючи графіки руху і місця зупинок, що полегшує процес планування вашої поїздки. Важливим аспектом є відгуки та оцінки інших користувачів, які надають вам інформацію про досвід подорожі та надійність автобусних компаній та маршрутів. Система оплати робить процес придбання квитків максимально зручним, пропонуючи різні варіанти оплати для вашого комфорту. Мобільний додаток дозволяє вам шукати та бронювати квитки прямо зі свого смартфона, що робить використання сервісу ще більш зручним. Якщо виникають питання або потрібна допомога, служба підтримки завжди готова надати вам необхідну інформацію та вирішити будь-які питання. Перед вибором Busfor рекомендується вивчити умови користування, політику повернення коштів і відгуки

інших користувачів для забезпечення приємного та надійного маневрування платформою.

Недоліки користування онлайн сервісом BusFor:

- Обмежена географічна доступність: Busfor може не бути таким покритим у всіх регіонах, що обмежує доступність сервісу для деяких користувачів;
- Варіативність якості послуг: Як з будь-яким сервісом, який пропонує різні автобусні компанії, якість послуг може варіюватися, і не завжди гарантується однаково високий стандарт;
- Можливість змін у графіку руху: Зміни у графіку руху або скасування поїздок можуть виникнути, що може вплинути на плани користувачів;
- Потреба в інтернет-з'єднанні: Використання Busfor вимагає наявності інтернет-з'єднання, що може бути не зручно в умовах обмеженого доступу до мережі;
- Можливість змін у ціновій політиці: Ціни на квитки можуть змінюватися, і можуть виникнути непередбачувані зміни у ціновій політиці автобусних компаній.

FlixBus - це популярний сервіс для міжміських автобусних подорожей у Європі. Він спеціалізується на забезпеченні доступних та зручних маршрутів для подорожуючих. За допомогою зручної онлайн-платформи ви можете легко шукати, порівнювати ціни та бронювати квитки, що робить процес організації поїздок максимально простим. Широка мережа маршрутів FlixBus охоплює різні локації у Європі, що дозволяє вам ефективно переміщатися між різними місцями. Однією з головних переваг є доступні ціни на квитки, що робить FlixBus конкурентоспроможним порівняно з іншими видами транспорту. На борту автобусів є різноманітні зручності, такі як безкоштовний Wi-Fi, розетки для заряджання та кондиціювання. FlixBus також активно працює над екологічною ініціативою для зменшення впливу перевезень на навколишнє середовище. Пасажири можуть залишати відгуки та ставити оцінки, надаючи іншим користувачам корисну інформацію. Завдяки спеціальним пропозиціям, акціям та знижкам на квитки, подорожі з FlixBus стають ще більш доступними та привабливими.

Основні недоліки FlixBus:

- Можливість змін у графіку руху: Час від часу можливі зміни у графіку руху або скасування маршрутів, що може вплинути на плани подорожуючих;
- Обмежена доступність: В деяких регіонах та країнах сервіс може бути менш доступним, обмежуючи вибір для користувачів;
- Обмеження багажу: Є обмеження щодо розміру та ваги багажу, що може бути неприємним для тих, хто подорожує з великим обсягом або важкими речами;
- Нестабільність якості послуг: Як і у багатьох транспортних підприємств, які працюють з різними підприємцями, які здійснюють перевезення, якість послуг може варіюватися.

Отже, BlaBlaCar, Busfor та FlixBus представляють собою онлайн-платформи, що спеціалізуються на пасажирських перевезеннях та пропонують різноманітні види транспорту. Користувачі можуть створювати облікові записи на цих платформах для зручного бронювання подорожей. Вони поділяють спільні риси, такі як надання міжнародних маршрутів, робота як онлайн-платформи та акцент на пасажирському транспорті.

Проте їхні відмінності полягають у типі транспорту, який вони примарно пропонують. BlaBlaCar спеціалізується на карпулінгу, де пасажири ділять витрати на подорожі з приватними водіями. Busfor зосереджений на автобусних подорожах, включаючи регіональні та міжнародні маршрути. FlixBus діє як автономний оператор із власним автопарком, пропонуючи автобусні маршрути в Європі та інших регіонах.

Інша відмінність полягає у їхній співпраці із перевізниками. BlaBlaCar сприяє співпраці між приватними особами, які подорожують разом, тоді як Busfor співпрацює з автобусними компаніями та операторами. FlixBus, з іншого боку, діє як автономний оператор із власним автопарком.

Крім того, їхні бізнес-моделі різняться. BlaBlaCar ґрунтується на карпулінгу та діленні витрат, Busfor отримує комісію за продаж квитків на автобусні маршрути, а FlixBus працює за моделлю продажу квитків для свого власного автопарку.

Географічно BlaBlaCar має глобальну присутність та активний у багатьох країнах світу. Busfor спеціалізується на регіональних та міжнародних маршрутах, зокрема у Східній Європі. FlixBus широко представлений в Європі та інших частинах світу.

Загалом, хоча ці платформи мають спільні риси в онлайн-бронюванні та пасажирських перевезеннях, їхні відмінності у типах транспорту, бізнес-моделях та географічній фокусуванні роблять їх підходящими для різноманітних потреб у подорожах та виборів користувачів.

3.2. Проектування веб-сервісу.

В сучасному цифровому світі веб-додатки визначають нові стандарти у взаємодії користувачів з інформацією та сервісами. Проєктування веб-додатків вимагає глибокого розуміння потреб користувачів, технічних вимог та найсучасніших технологій. Цей процес виявляється не лише технічним викликом, але й творчим завданням, яке передбачає гармонійне поєднання функціональності та естетики.

Починаючи з аналізу вимог та закінчуючи етапом впровадження, проектування веб-додатку вимагає від команди розробників інноваційного мислення, системного підходу та уваги до деталей. У цьому контексті, важливо зазначити, що успішне проектування веб-додатків не тільки визначає його функціональність, але й створює приємний та ефективний інтерфейс для користувача.

Цей процес необхідно розглядати як невід'ємну частину великого екосистеми сучасних технологій, що пропонують нові можливості для взаємодії та розвитку. В умовах швидкого темпу змін та росту веб-середовища, проектування веб-додатків є ключовим елементом успіху та конкурентоспроможності.

Проєктування веб-додатку перед його реалізацією є важливим етапом у розробці програмного забезпечення. На цьому етапі визначаються конкретні цілі та вимоги до додатку для спільного розуміння всіма учасниками проекту.

Даний підхід також допомагає ефективно управляти ресурсами, включаючи час, гроші та людські ресурси. Визначення архітектури та структури допомагає створити масштабовану та ефективну систему, яка легко розширюється в майбутньому.

Проєктування сприяє вдосконаленню користувацького досвіду, дозволяючи визначити оптимальний інтерфейс та функціонал. Такий підхід спрощує розробку, адже з чітким планом можна ефективно працювати над окремими частинами проекту.

Цей етап також допомагає зменшити ризики, адже аналіз і проєктування дозволяють ідентифікувати можливі проблеми ще до початку реалізації. Виявлення та виправлення помилок на етапі проєктування вартіше, ніж на етапі реалізації, що сприяє ефективному управлінню вартістю проекту.

3.2.1. Діаграма варіантів використання.

Діаграма варіантів використання є частиною UML (Unified Modeling Language) та служить для моделювання функціоналу системи з огляду на її взаємодію з різними зовнішніми агентами, такими як користувачі або інші системи.

У цій діаграмі актори представляють зовнішні сутності, такі як користувачі чи інші системи, і варіанти використання описують конкретні функціональні можливості системи. Актори розташовані за межами системи, а варіанти використання розміщені всередині системи і пов'язані лініями з акторами.

Ця діаграма використовується для моделювання функцій системи, визначення вимог та полегшення комунікації між розробниками та іншими учасниками процесу розробки програмного забезпечення. Її структура спрощує визначення того, як система взаємодіє з акторами, що робить її корисною для аналізу та документування функціоналу системи.

Для того що побачити повну картину роботи веб-сервісу було реалізовано діаграму варіантів використання:

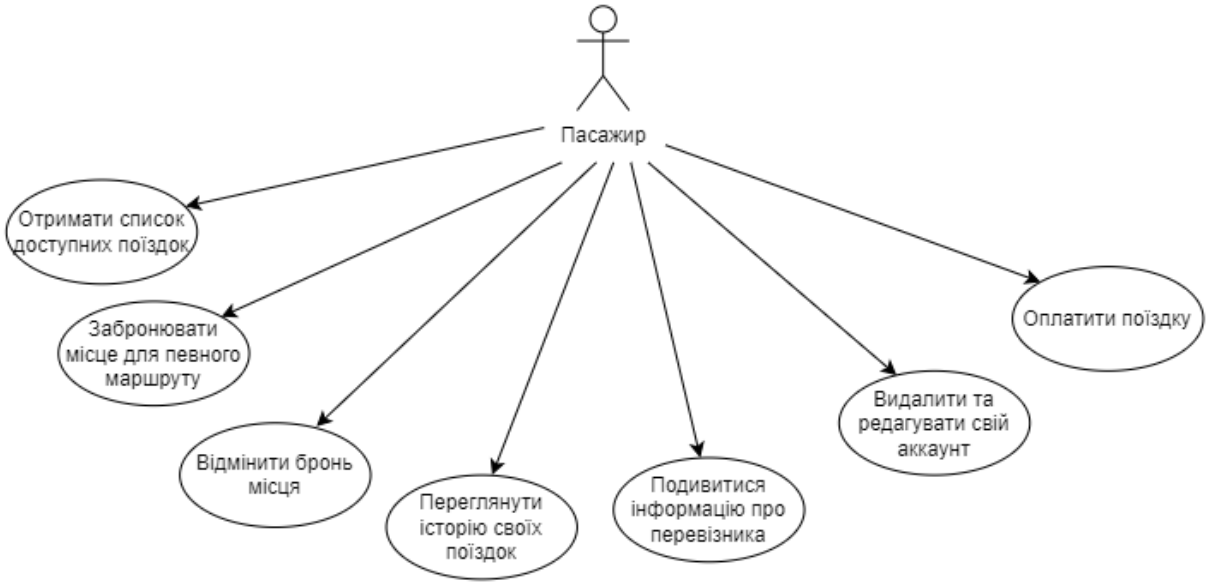

Рисунок. 1.10. Діаграма варіантів використання пасажира

В пасажирській перевізницькій системі, веб-сервіс грає ключову роль у забезпеченні зручності та ефективності взаємодії між пасажирами та перевізниками. Пасажири використовують цей сервіс для великого спектру дій, починаючи від пошуку оптимальних маршрутів та завершуючи відстеженням історії своїх поїздок.

Пасажир має можливість детального пошуку квитків за вказаним маршрутом, враховуючи різні параметри, такі як час відправлення та прибуття, зручність посадки, і інші. Важливою функцією є можливість бронювання квитків онлайн, що значно полегшує пасажирам процес планування та придбання квитків. Також, історія поїздок зберігається в особистому кабінеті, де пасажір може переглядати деталі минулих подорожей та визначати свої уподобання.

Окрім того, інформація про перевізників, доступна в системі, надає пасажирам можливість оцінювати та обирати оптимальних перевізників для своїх поїздок. Це може включати рейтинги, відгуки від інших користувачів та деталі про послуги.

Спільно з цим, можливість створення та редагування особистого кабінету дає пасажирам можливість зберігати та керувати своїми особистими налаштуваннями, що робить використання сервісу ще більш індивідуалізованим та зручним.

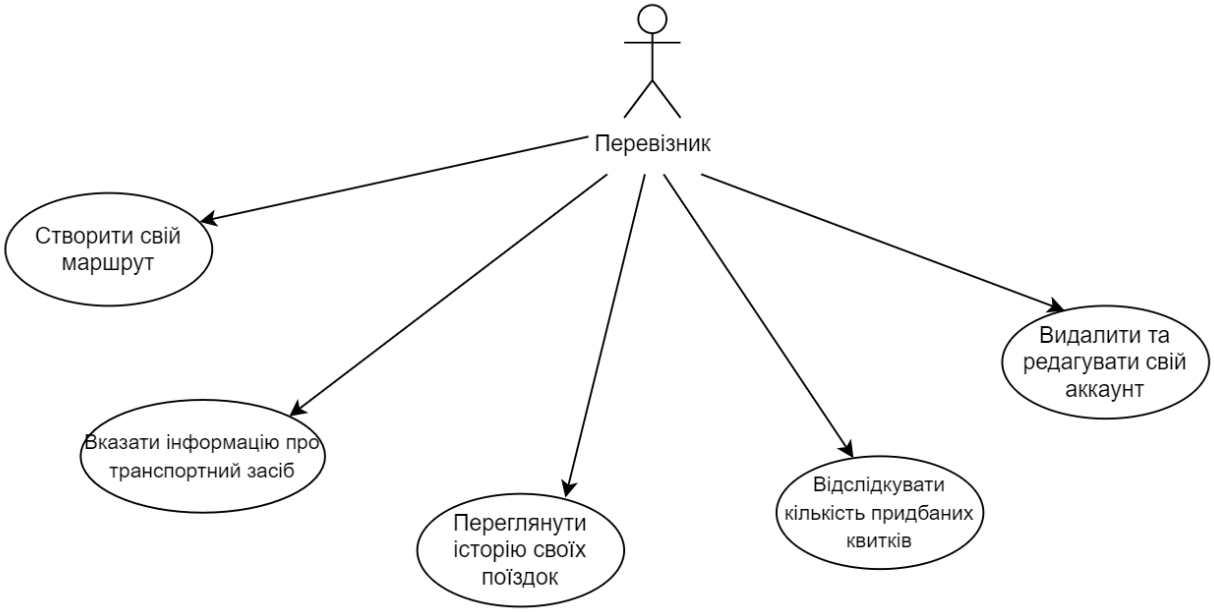

Рисунок 1.11. Діаграма варіантів використання перевізника

Перевізники, з іншого боку, можуть використовувати веб-сервіс для управління своїми маршрутами та транспортними засобами. Створення нових маршрутів включає визначення пунктів відправлення та призначення, а також часу руху. Перевізники можуть редагувати та оновлювати інформацію про свої транспортні засоби, вказуючи характеристики та вартість квитків.

Зокрема, можливість відслідковування заброньованих квитків на їхніх маршрутах дозволяє перевізникам ефективно планувати та керувати заповненістю транспортних засобів. Також, наявність особистого кабінету дозволяє перевізникам зручно управляти інформацією про свою компанію, переглядати відгуки та вносити необхідні зміни.

Така система взаємодії створює інтегроване середовище, де як пасажири, так і перевізники можуть максимально зручно та ефективно користуватися всіма можливостями пасажирських перевезень.

3.2.2. Діаграма послідовності.

Діаграма послідовності - це інструмент моделювання, який використовується для візуалізації взаємодії об'єктів в системі в часовому порядку. На діаграмі

відображаються об'єкти, лінії життя, стрілки послідовності та повідомлення, щоб ілюструвати порядок подій. Кожен об'єкт представлений вертикальною лінією, яка називається лінією життя. Стрілки послідовності вказують на порядок передачі повідомлень або викликів між об'єктами. На стрілках можуть бути вказані повідомлення, що передаються від одного об'єкта до іншого. Діаграма послідовності використовується для аналізу та документування взаємодії в системі, полегшуючи розуміння логіки та хронології подій.

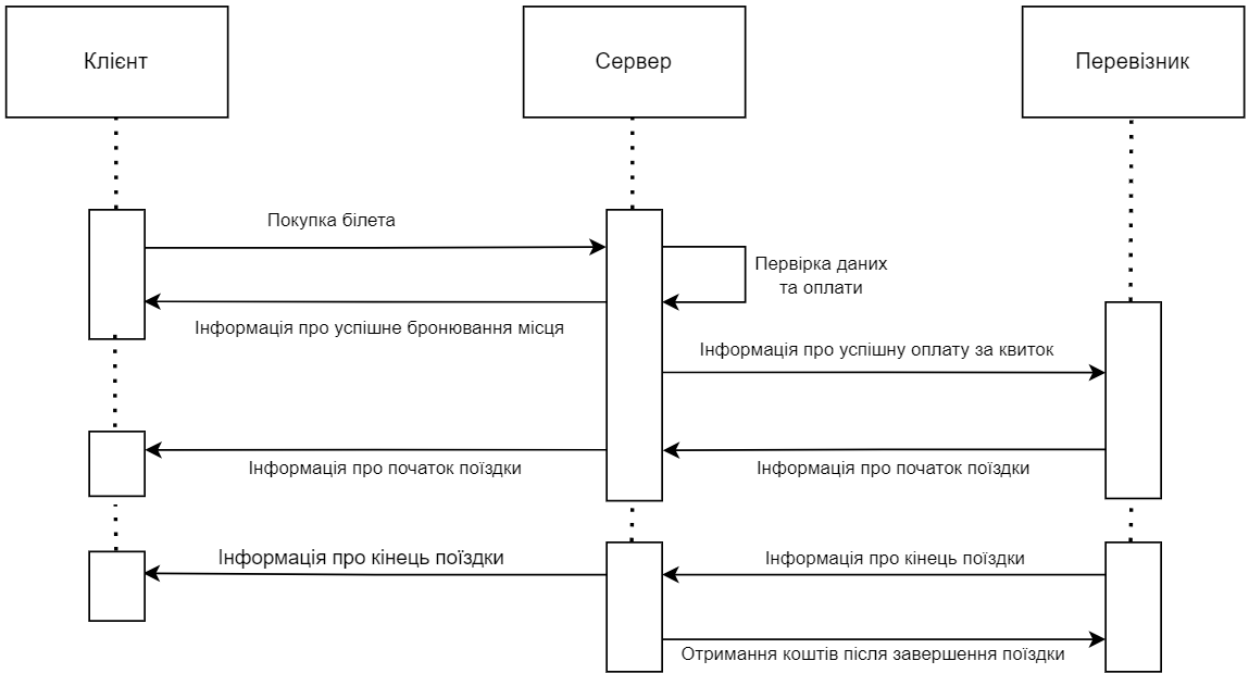

Рисунок 1.12. Діаграма послідовності успішної поїздки

Зв'язок між пасажиром, бекенд-сервером та перевізником в системі онлайн-покупки квитка на автобус включає кілька ключових кроків. Перш за все, пасажир використовує веб-сайт для вибору маршруту, дати та інших параметрів подорожі.

Далі, пасажир обирає зручний для себе час та місце посадки. Після цього відбувається передача запиту на бекенд-сервер, де клієнт формує запит інформації про наявність вільних місць на обраному автобусі та ціну квитка.

На етапі бекенд-сервера відбувається обробка запиту пасажира, взаємодія з базою даних для перевірки наявності вільних місць та інших деталей, і відповідь на запит пасажира інформацією про наявність місць та ціни. Після отримання відповіді

пасажир підтверджує свій вибір та здійснює оплату через інтегровану платіжну систему. Після успішної оплати бекенд-сервер генерує електронний квиток та відправляє пасажирові підтвердження оплати та електронний квиток.

Інформація про купівлю та пасажира передається перевізнику, щоб вони могли підготуватися до прибуття пасажира. Нарешті, на автобусній зупинці пасажир пред'являє електронний квиток або інформацію про покупку для посадки в автобус. Далі після завершення поїздки водій передає інформацію на сервер і автоматично цю інформацію отримує пасажир від бекенду. Потім через декілька хвилин перевізник отримує кошти за кожного пасажира та за весь маршрут.

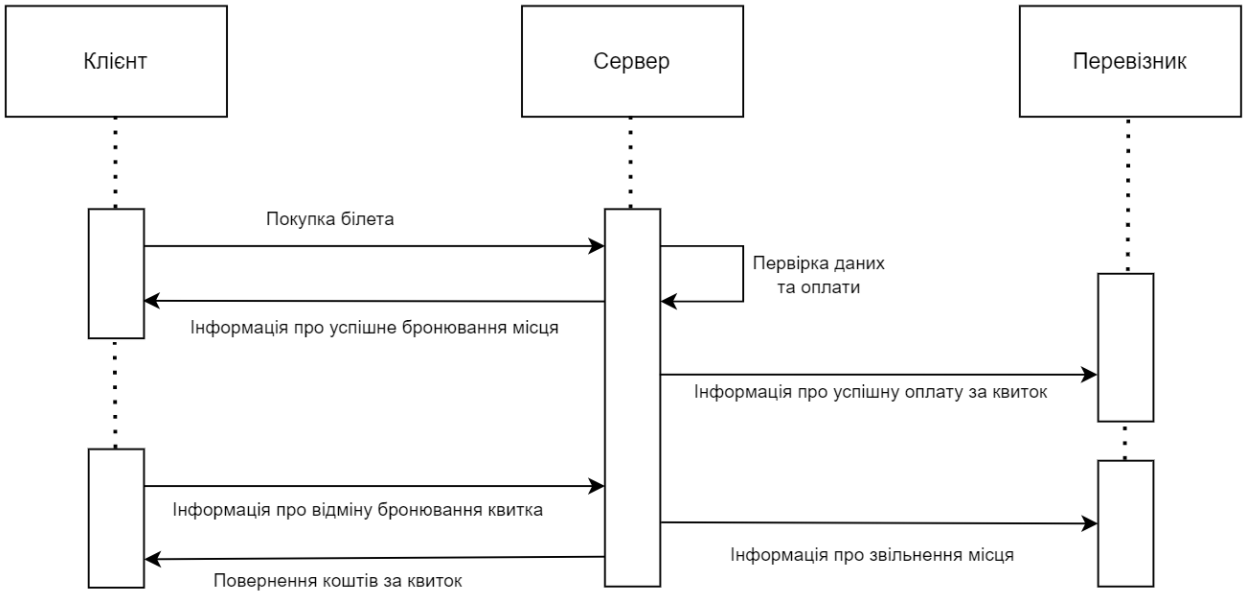

Рисунок 1.13. Діаграма послідовності неуспішної поїздки

На даній діаграмі виокремлюється, що процес придбання та отримання інформації аналогічний попередній діаграмі, проте наступні кроки відрізняються: замість початку подорожі пасажир скасовує бронювання квитка та відмовляється від подорожі. Одразу після отримання сервером цього запиту, відбувається передача інформації перевізнику про вакантне місце в результаті скасування бронювання. По закінченню цього процесу пасажиру повертаються кошти на його реквізити.

3.2.3. Структура веб-сервісу для пасажирських перевезень.

Веб-сервіс пасажирських перевезень має складну структуру, що залежить від його конкретних функцій і потреб. Додаток сформований з п’яти основних частин тісно пов’язаних між собою:

- сторінка верифікації;
- головна сторінка;
- особистий кабінет перевізника;
- особистий кабінет пасажира;
- сторінка оформлення та оплати квитка.

Рисунок 1.14. Структурна схема сервісу

Сторінка верифікації служить першою точкою входу для користувачів, які мають можливість скористатися однією з таких функцій: авторизація, реєстрація. Далі йде головна сторінка де клієнти можуть швидко використовувати функцію пошуку, фільтрації та сортування для вибору напрямку подорожі. Форма пошуку надає можливість введення точок відправлення та призначення, а також вказання дати і часу, аби отримати ідеально підібрані результати.

Результати пошуку представляються у вигляді доступних варіантів перевезень з деталями, такими як час відправлення, призначення, тип транспорту та вартість. Користувачі можуть обирати конкретний варіант і здійснювати бронювання та сплату через безпечний механізм оплати, що інтегрований у систему.

База даних зберігає інформацію про користувачів, їхні історії бронювань, розклади, маршрути, особисту інформацію та інші важливі дані, які можна переглянути в особистому кабінеті. Система повідомлень надсилає підтвердження,

сповіщення про зміни у рейсах та інші важливі повідомлення користувачам. Наприкінці, основа безпеки включає шифрування та захист даних, а також механізми захисту від потенційних атак.

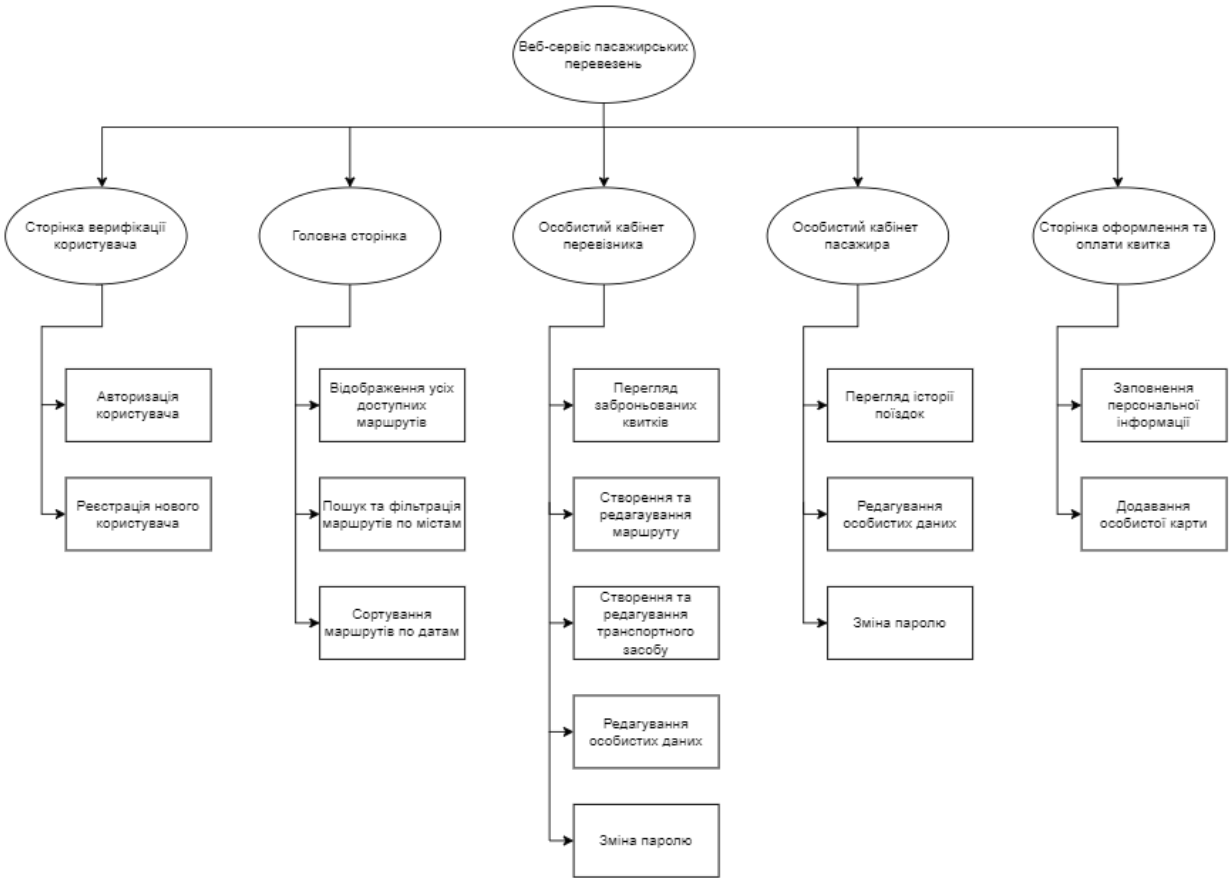

Рисунок 1.15. Функціональна схема сервісу

3.2.4. Проектування бази даних.

Проектування баз даних – це важлива частина створення програмного забезпечення. Цей процес допомагає визначити, як будуть зберігатися та організовані дані у системі. На початку проводиться аналіз вимог, де ми отримуємо від користувачів вимоги та розуміємо основні об'єкти та їхні атрибути.

Далі створюється концептуальна модель за допомогою інструментів, таких як діаграми сутностей-зв'язків (ERD). Потім йде нормалізація даних, щоб уникнути непорозумінь та покращити використання пам'яті.

Після цього концептуальну модель перетворюється в логічну, де визначаються таблиці, їхні ключі, індекси та інші елементи бази даних. Важливим етапом є визначення типів даних для кожного атрибута. Потім ми переходимо до фізичного дизайну, обираємо систему управління базами даних та налаштовуємо параметри для оптимізації продуктивності.

Фізичний дизайн включає створення структури бази даних, розміщення таблиць, індексів та інших об'єктів. Після цього проводиться оптимізація та тестування для підвищення продуктивності та перевірки правильності роботи бази даних.

Завершальним етапом є документування, де створюється документація, що описує всі аспекти бази даних. Цей процес гарантує ефективність, надійність та зручність у користуванні базою даних на протязі усього життєвого циклу системи.

В проекті веб-сервіс для пасажирських перевезень існують шість колекцій бази даних, які взаємодіють між собою. Перша колекція, "User" (Користувач), використовується для зберігання інформації про користувача системи – пасажира або перевізника. У неї входять атрибути, такі як id - унікальний ідентифікатор клієнта, userRole - роль користувача, firstName - ім'я, secondName - прізвище, email - адреса електронної пошти та phone - номер телефону.

Сутність "UserHistorys" (Історія користувача), зберігає історію поїздок пасажира і включає в себе атрибути: id - ідентифікатор, bookingIds - список ідентифікаторів бронювань та userId - ідентифікатор користувача.

Третя колекція, "Bookings" (Бронювання), відповідає за інформацію про бронювання квитків і включає атрибути, такі як id - унікальний ідентифікатор квитка, status - статус, userId - ідентифікатор користувача, routeId - ідентифікатор маршруту, start - місце початку і end - закінчення маршруту, та price - вартість квитка.

Четверта колекція, "Vehicles" (Транспортні засоби), має інформацію про доступні транспортні засоби перевізника, включаючи id - неповторний ідентифікатор транспорту, name - назву автобусу, image - зображення автобусу, registrationId - реєстраційний номер транспорту та userId - ідентифікатор перевізника.

П'ята сутність, "Routes" (Маршрути), зберігає дані про доступні маршрути перевізника. Вона включає такі атрибути, як id - унікальний ідентифікатор маршруту, startLocation - місце початку і endLocation - закінчення маршруту, userId - ідентифікатор користувача, price - вартість одного посадочного місця на маршрут, vehicleId - ідентифікатор транспортного засобу призначеного для маршруту, та routeStopIds - список ідентифікаторів зупинок на маршруті.

Шоста колекція, "RouteStops" (Зупинки маршруту), відповідає за інформацію про зупинки на маршруті та містить в собі id - ідентифікатор зупинки, location - місце розташування зупинки та userId - ідентифікатор користувача.

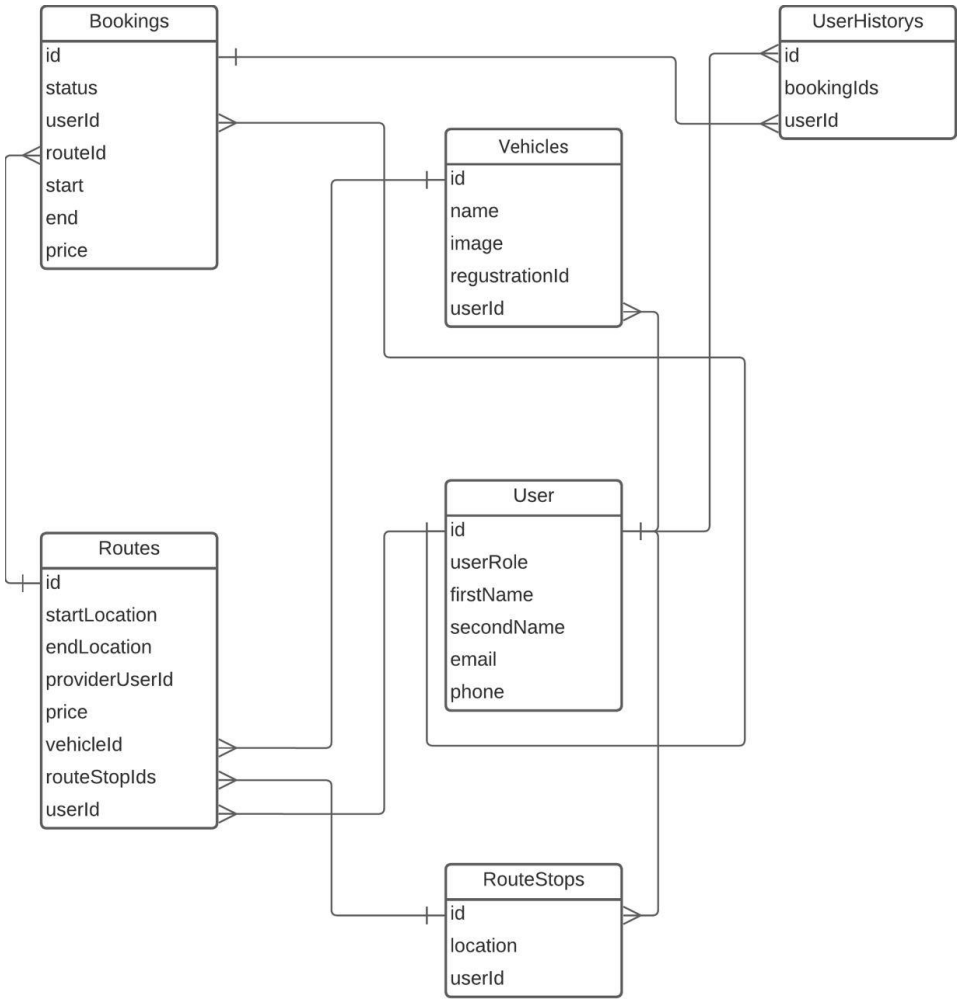

Рисунок 1.16. Загальна схема бази даних

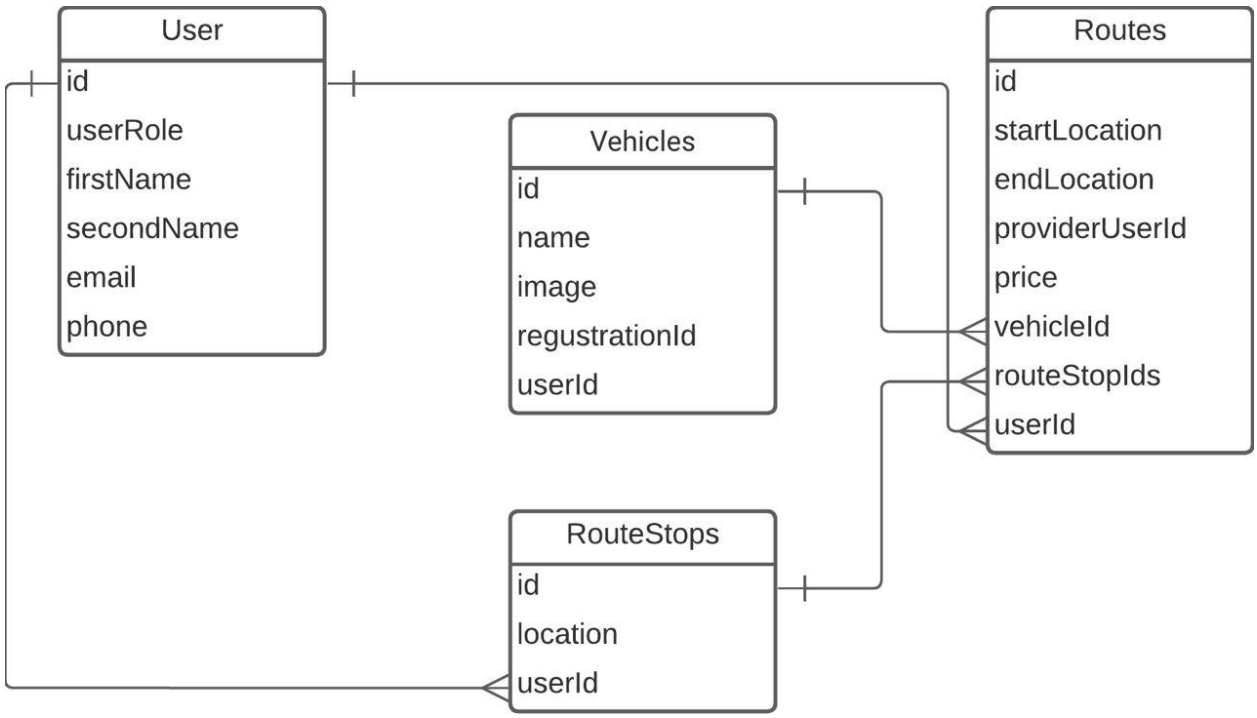

Рисунок 1.17. Процес створення колекції Routes

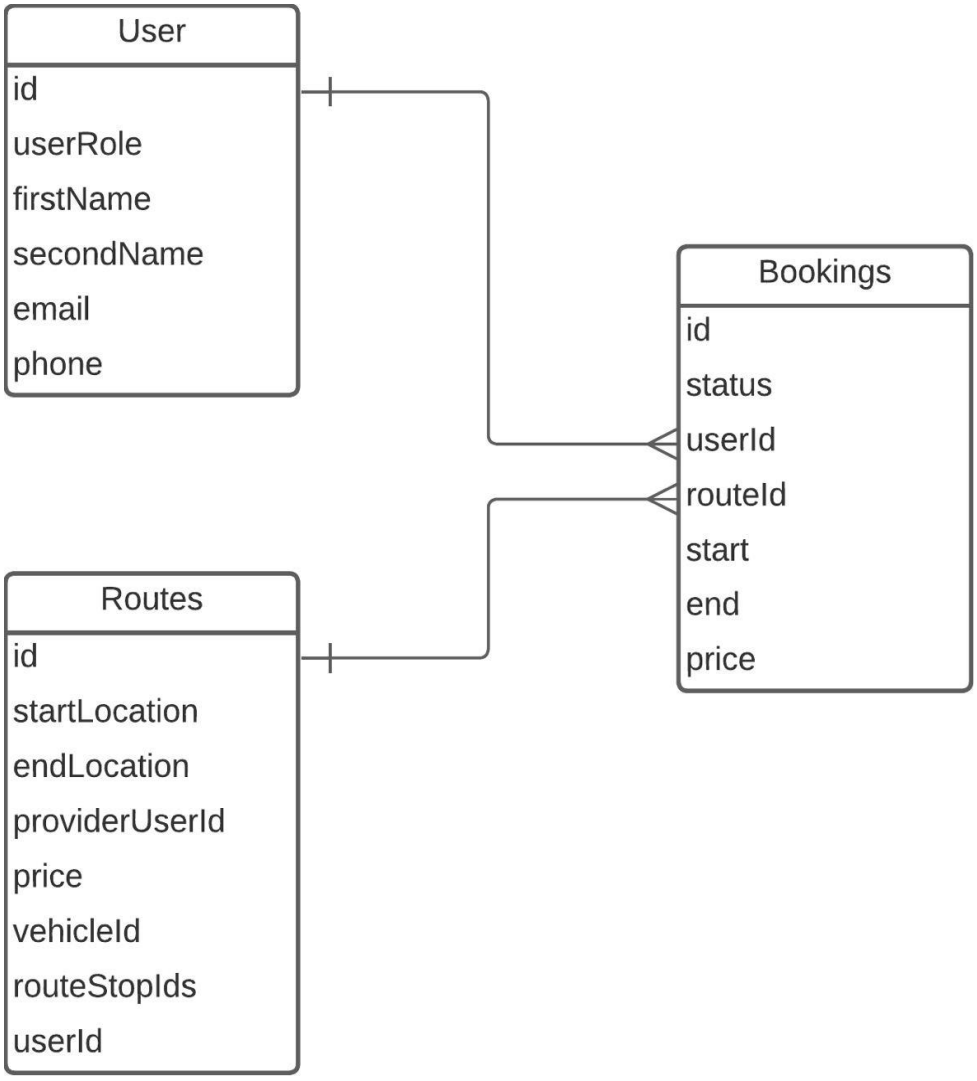

Рисунок 1.18. Процес створення колекції Bookings

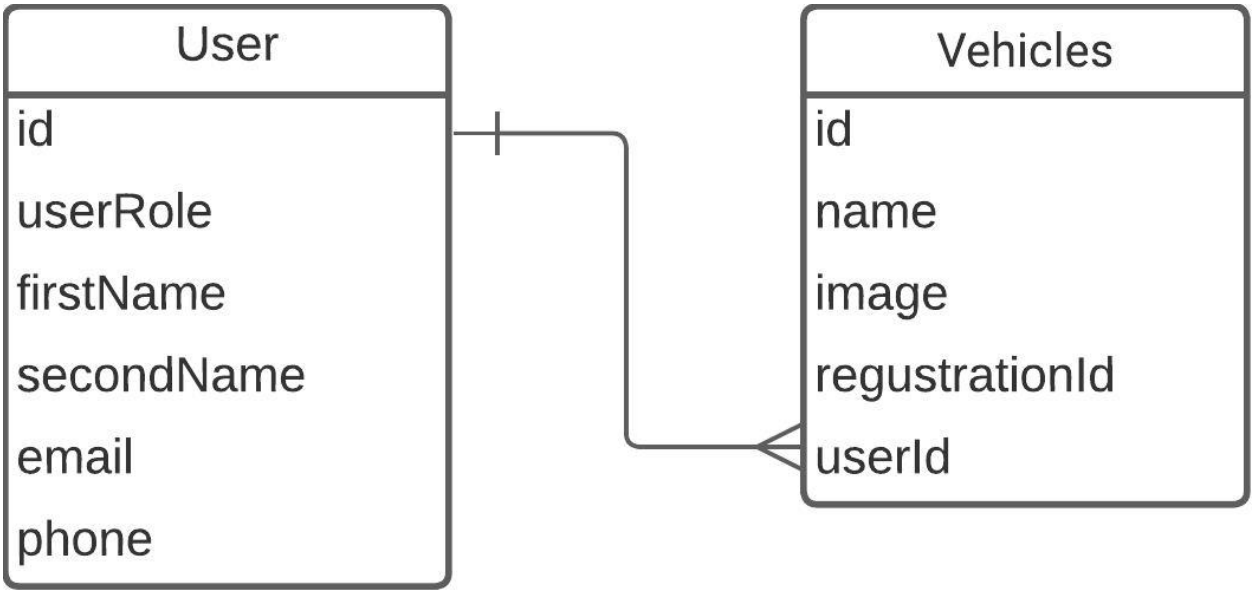

Рисунок 1.19. Процес створення колекції Vehicles

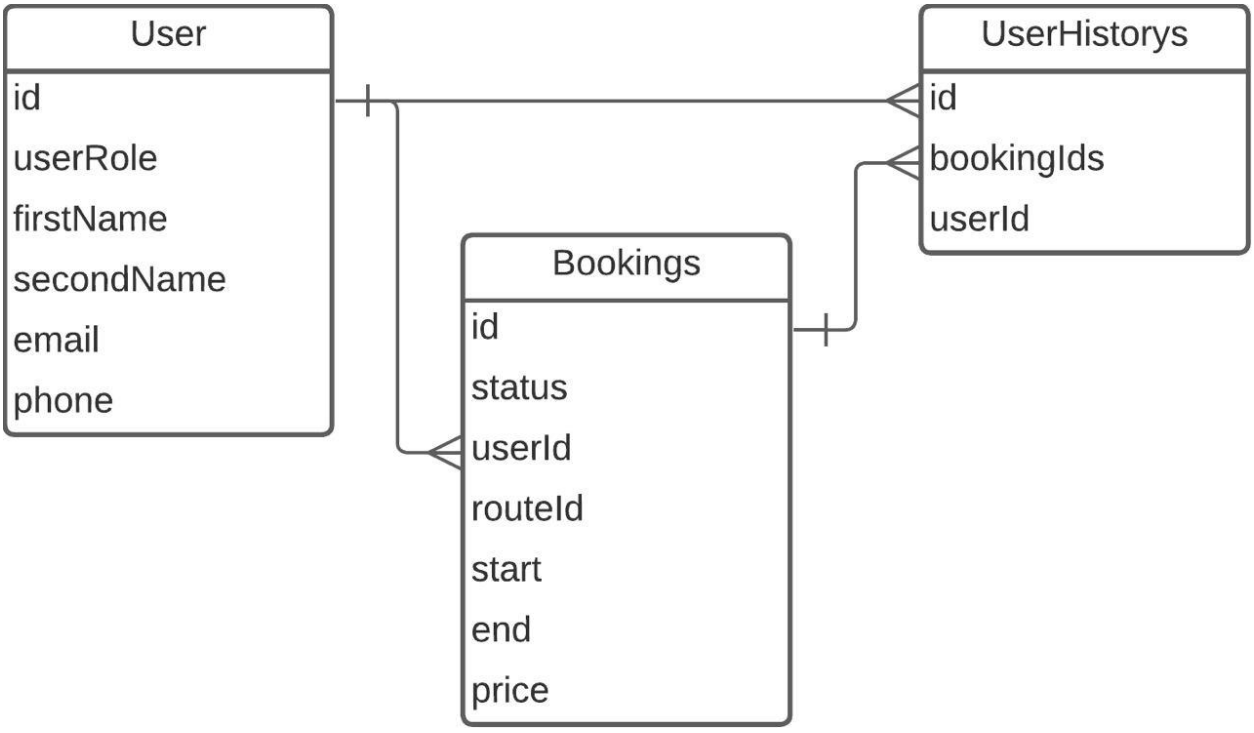

Рисунок 1.20. Процес створення колекції UserHistorys

3.3. Реалізація проекту.

Головна сторінка

Головна сторінка веб-додатку є віртуальним вітальним майданчиком, який відображає сутність та функціональні можливості сервісу. Це перше місце, де користувачі знаходяться при відвідуванні веб-додатку, тому великою мірою визначає їх перший враження та взаємодію з платформою. Перша сторінка повинна бути ефективним засобом навігації, візуально привабливою та інформаційно насиченою.

Вона відображає ключові функції та переваги додатку, надає короткий огляд його можливостей та сервісів. Важливо забезпечити легку навігацію і відобразити основні елементи інтерфейсу, щоб користувачі могли швидко зорієнтуватися та знайти потрібну інформацію.

У кінцевому підсумку, домашня сторінка має бути добре збалансованою, враховуючи як візуальні, так і інформаційні аспекти, для того щоб створювати позитивне враження і заохочувати користувачів до подальшої взаємодії з веб-додатком.

Головна сторінка веб-додатку пасажирських перевезень має сучасний та зручний дизайн, який сприяє легкості користування. Верхня частина сторінки включає в себе логотип, який ідентифікує сервіс, а також кнопку входу в особистий кабінет. По середині сторінки розташована панель пошуку маршруту. Ця панель включає поля для введення назви населеного пункту початку поїздки та міста призначення. Крім того, є поле для вказання дати поїздки, де користувач може вибрати потрібну дату з календаря. Наведенням курсора на поле вводу дати може виводитися календарний віджет для більш зручного вибору дати. Правіше полів введення розташована кнопка "Пошук", яка ініціює пошук оптимального маршруту в зазначену дату. Результати пошуку виводяться на новій сторінці.

Цей інтерфейс спрямований на простоту та зручність використання, надаючи користувачам легкий доступ до основних функцій сервісу.

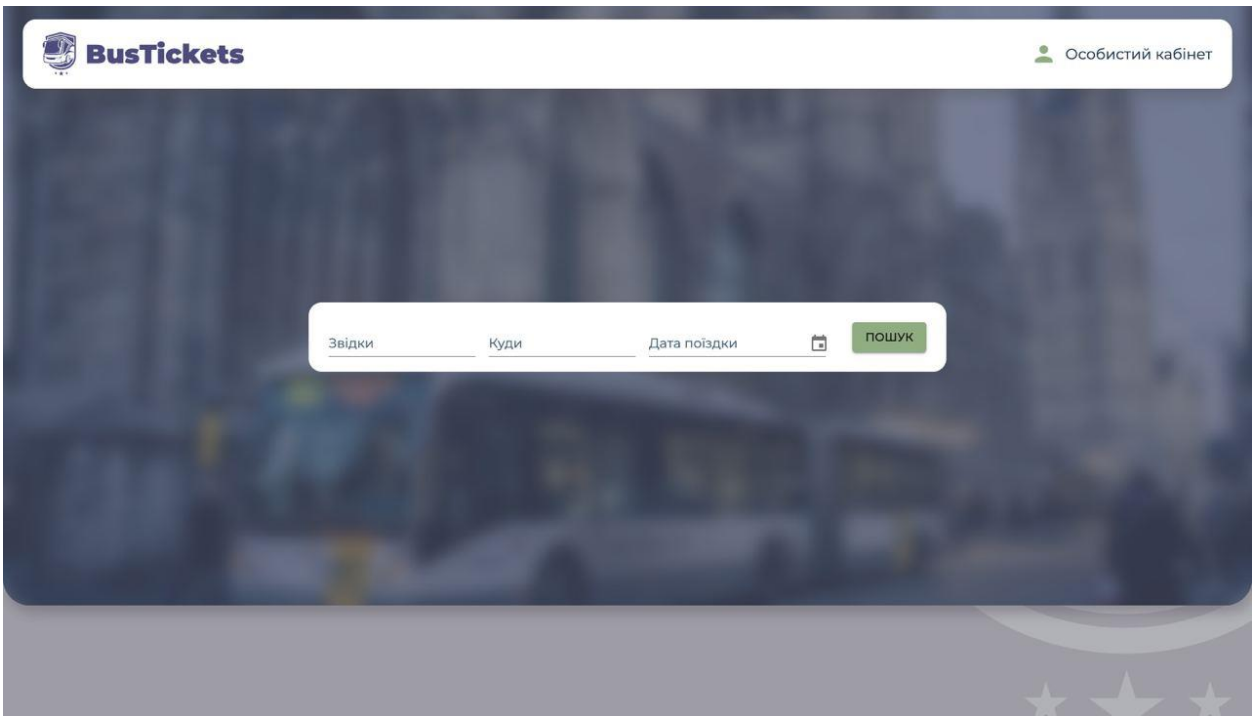

Рисунок 1.21. Головна сторінка

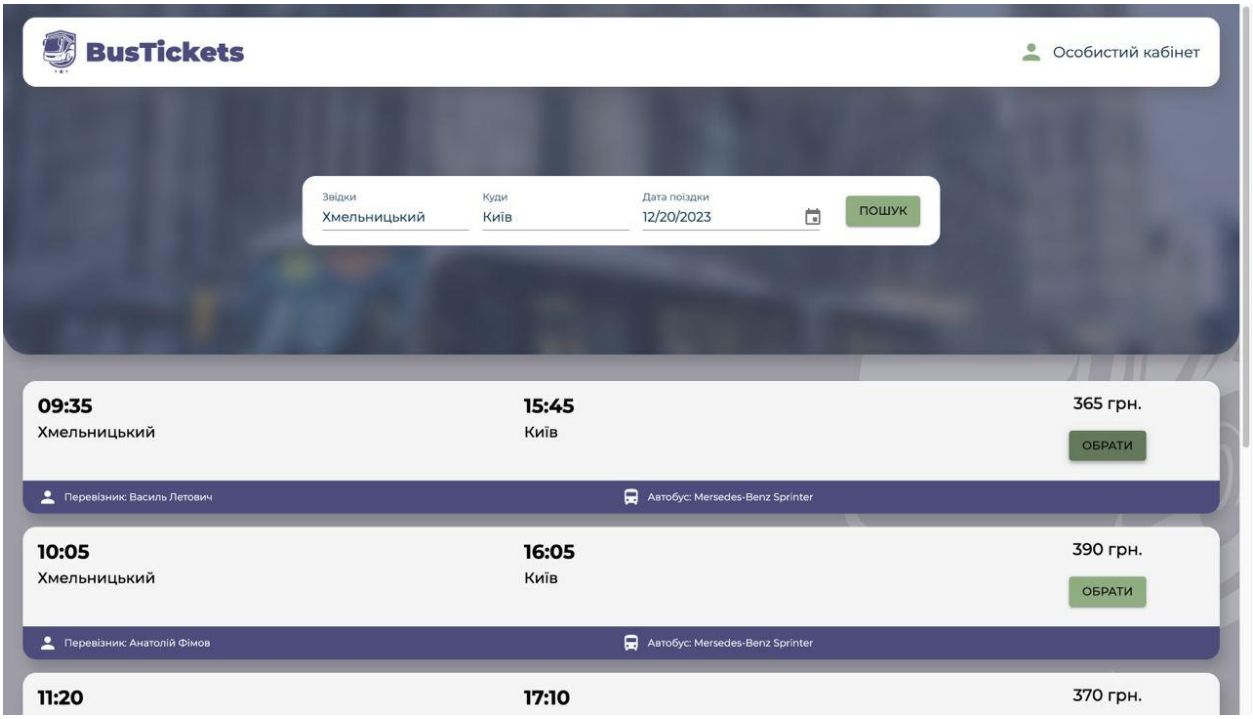

Рисунок 1.22. Головна сторінка після виконання пошуку

Сторінка реєстрації

Створення сторінки реєстрації є важливим етапом у розробці онлайн-сервісів або веб-додатків. На цій сторінці користувачі можуть створити обліковий запис для

використання функціоналу системи. Основні елементи сторінки реєстрації включають форму з обов'язковими полями, такими як ім'я, прізвище, електронна пошта та пароль.

Форма реєстрації також може містити додаткові елементи, наприклад, поля для підтвердження паролю та вибору способу авторизації, такого як за допомогою соціальних мереж. На сторінці зазвичай присутня кнопка "Зареєструватися" або аналогічна, яку користувачі натискують для завершення реєстрації.

Додатково, може бути включене посилання для тих, хто вже має обліковий запис, і хоче увійти. Також важливо враховувати заходи безпеки, такі як використання капчі або reCAPTCHA для уникнення автоматизованих реєстрацій та забезпечення конфіденційності даних. У разі необхідності може бути запит на підтвердження облікового запису через електронну пошту або SMS-повідомлення.

Забезпечення адаптивного дизайну для мобільних пристроїв також є важливим аспектом, щоб зробити процес реєстрації зручним для користувачів на різних пристроях. Крім того, слід додати елементи аналітики, такі як Google Analytics, для відстеження активності користувачів на сторінці реєстрації. Такий комплексний підхід сприяє створенню ефективної та безпечної сторінки реєстрації веб-додатку.

Сторінка реєстрації даного проекту включає в себе центрально розташовану форму, яка надає можливість користувачам реєструватися. Форма реєстрації включає наступні поля: ім'я, прізвище, телефон, електронна пошта, пароль, а також чекбокси для вибору ролі - пасажира чи перевізника. Нижче полів розташована кнопка "Зареєструватися", яка ініціює процес реєстрації. Після натискання цієї кнопки інформація з форми передається на сервер та зберігається у базу даних. Та з цього випливає, що наступного разу користувачу не потрібно буде знову вписувати свої дані, а достатньо буде пройти авторизацію. Якщо ж користувач вже раніше був зареєстрований, то він одразу може авторизуватися у веб-сервісі натиснувши на посилання “Авторизація” у правому верхньому куті сторінки.

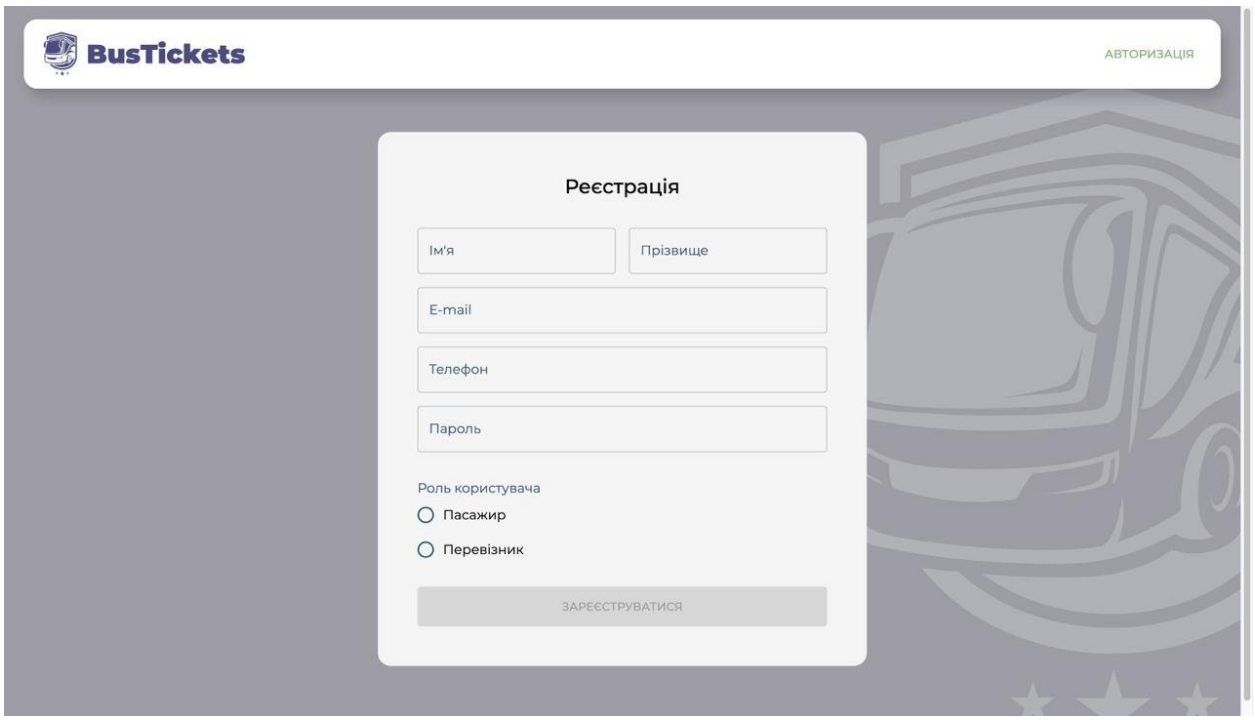

Рисунок 1.23. Сторінка реєстрації

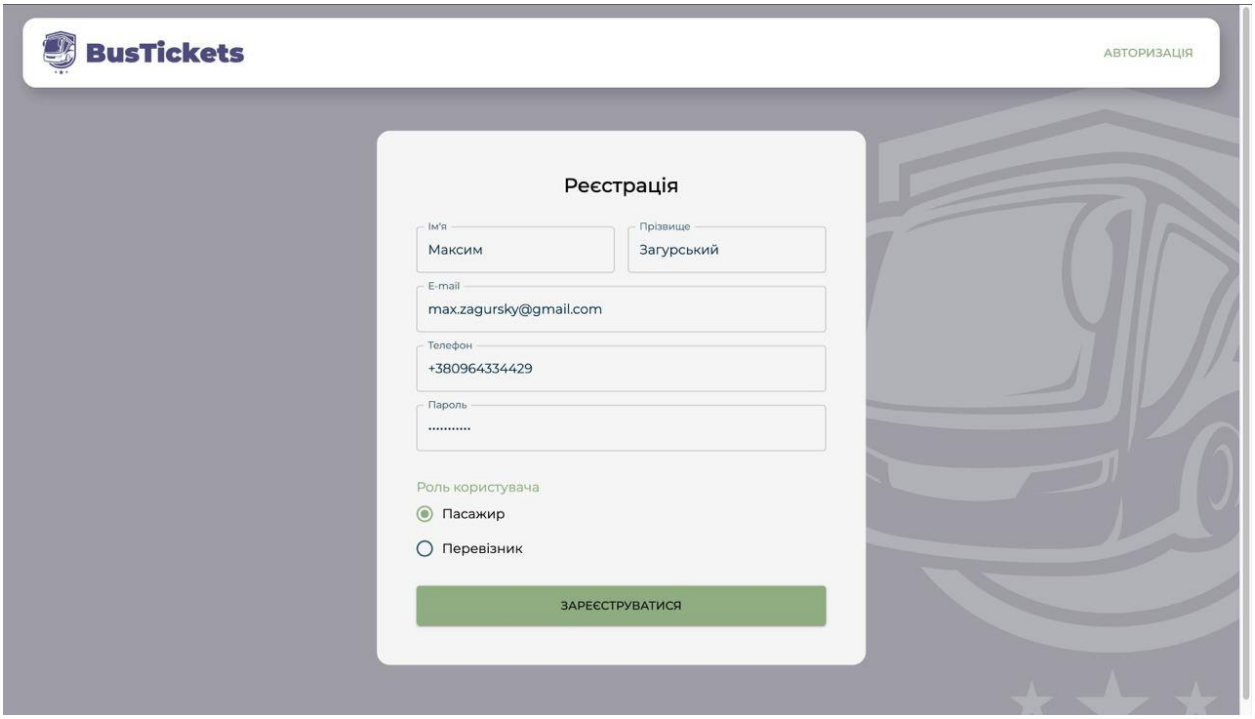

Рисунок 1.24. Сторінка реєстрації з заповненими даними

Сторінка авторизації

Сторінка авторизації веб-сервісу пасажирських перевезень містить форму для введення облікових даних. Форма авторизації включає такі поля: E-mail та пароль.

Нижче полів розташована кнопка "Увійти", яка дозволяє виконати процедуру авторизації.

Для нових користувачів у верхньому правому куті сторінки розміщене посилання "Реєстрація", яке спрямовує на сторінку реєстрації, де можна створити новий обліковий запис.

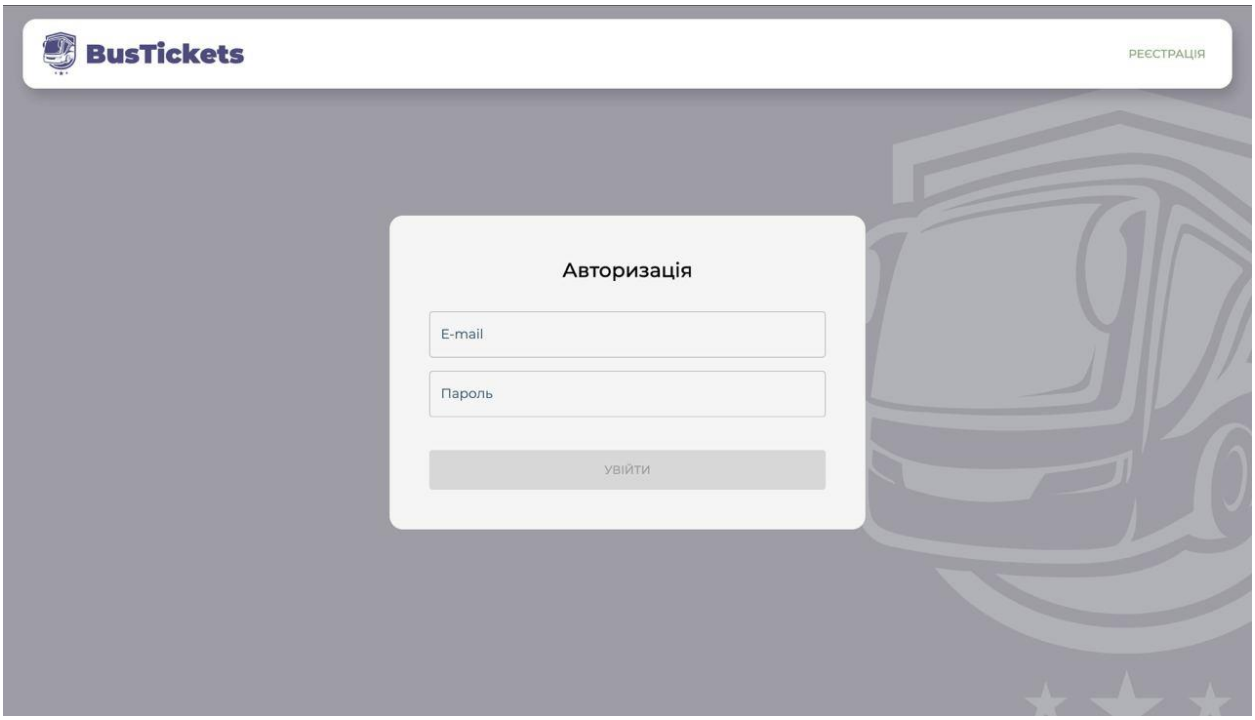

Рисунок 1.25. Сторінка авторизації

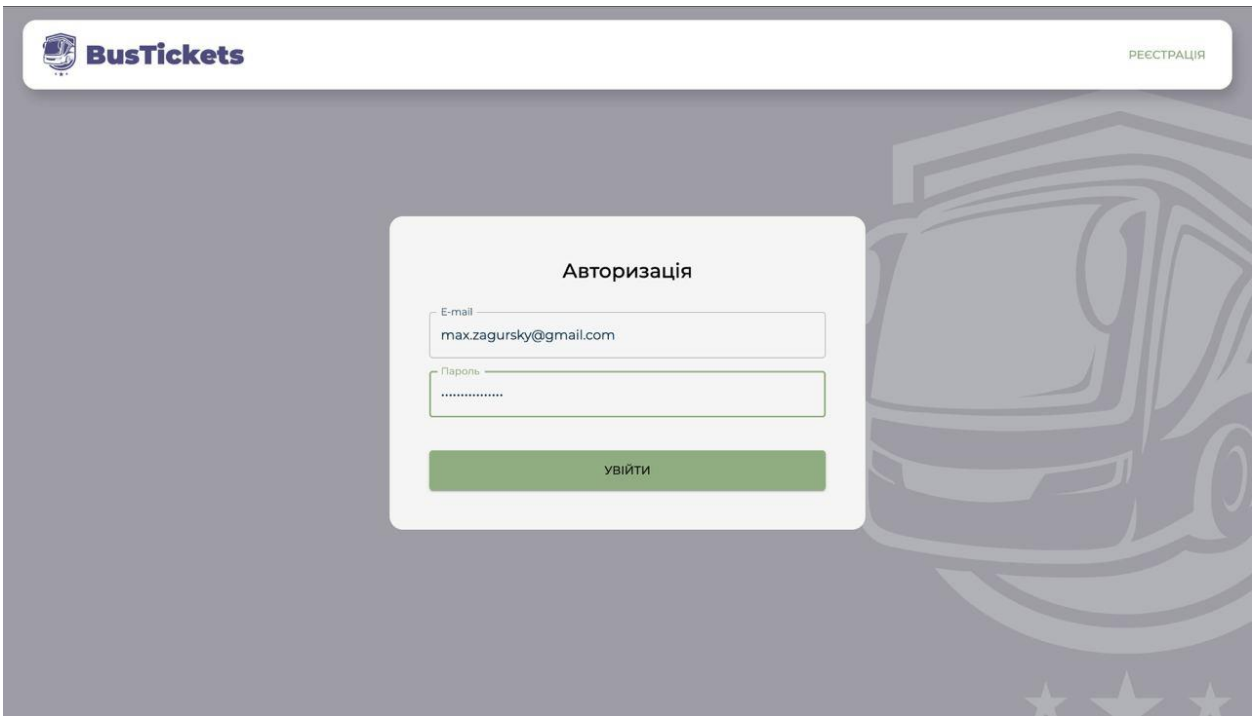

Рисунок 1.26. Сторінка авторизації з заповненими даними

Сторінка оформлення квитка.

Сторінка оформлення квитка розкриває вичерпну інформацію про маршрут, вид, модель транспорту та деталі про перевізника. На цій сторінці розташована також форма для заповнення особистих даних пасажира, включаючи ім'я, прізвище, e-mail та телефон. Після введення особистих даних, система автоматично розраховує вартість поїздки та відображає її нижче. Кнопка "Оплатити квиток" дозволяє легко здійснити оплату. Також для незареєстрованих користувачів можна натиснути на чекбокс “зареєструвати користувача” та система автоматично створить новий обліковий запис. Для зареєстрованих користувачів дані вручну вводити не потрібно, за них це сервіс зробить самостійно підтягнувши дані які були введені під час реєстрації.

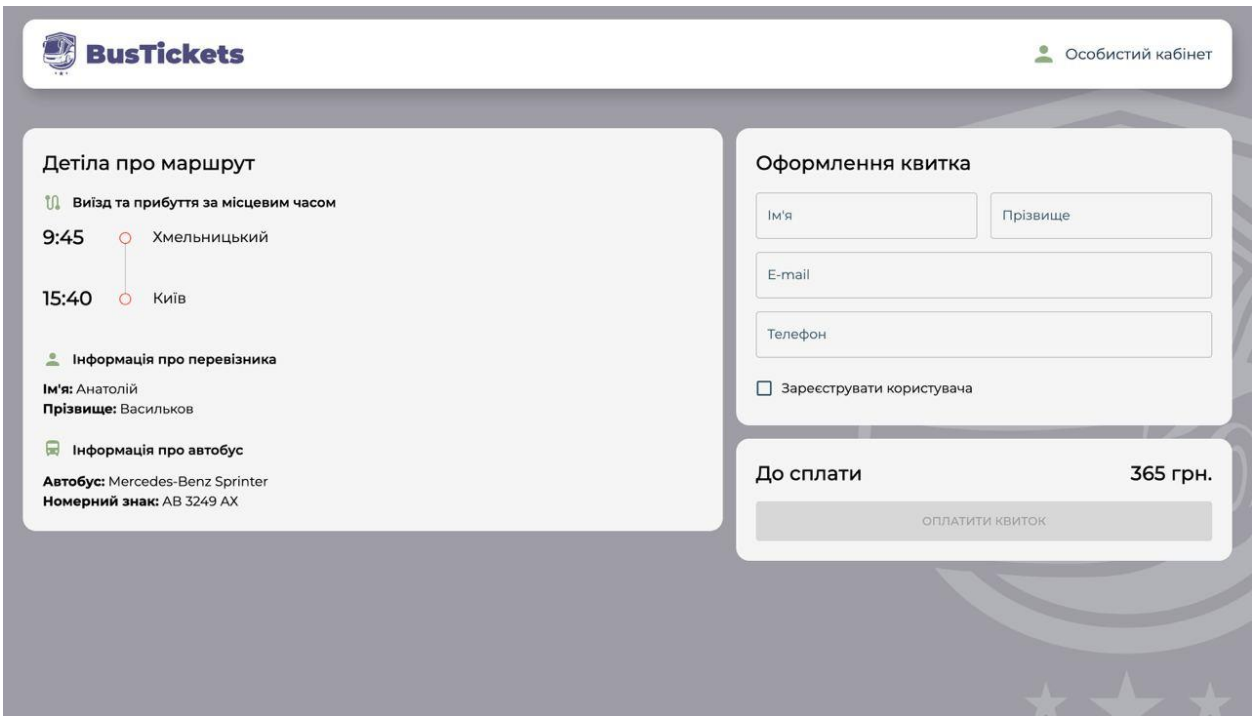

Рисунок 1.27. Сторінка оформлення квитка

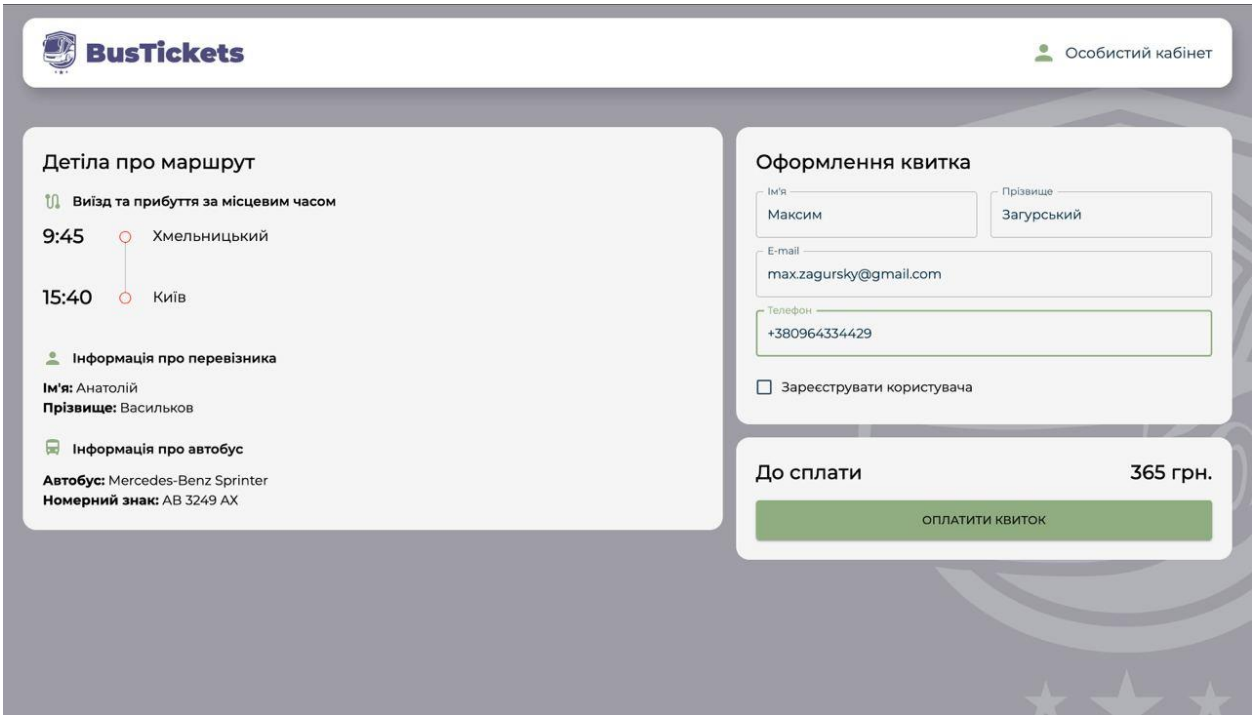

Рисунок 1.28. Сторінка оформлення квитка після заповнення даних

Оплата квитка.

Процес оплати онлайн квитка на бекенді розпочинається з вибору користувачем конкретного маршруту та введення необхідних даних.

Наступним кроком є вибір користувачем способу оплати, такого як кредитна карта, електронний гаманець google pay чи apple pay. Після вибору способу оплати користувач вводить необхідні платіжні дані, такі як номер кредитної карти, термін дії та CVV-код.

Отримавши платіжні дані, система передає їх в платіжний шлюз для безпечної обробки транзакції. Після успішної транзакції користувач отримує підтвердження оплати, яке може включати в себе електронний квиток, номер замовлення та іншу важливу інформацію.

Важливим етапом є оновлення бази даних, де система відзначає оплачені місця як зайняті та видаляє резерв.

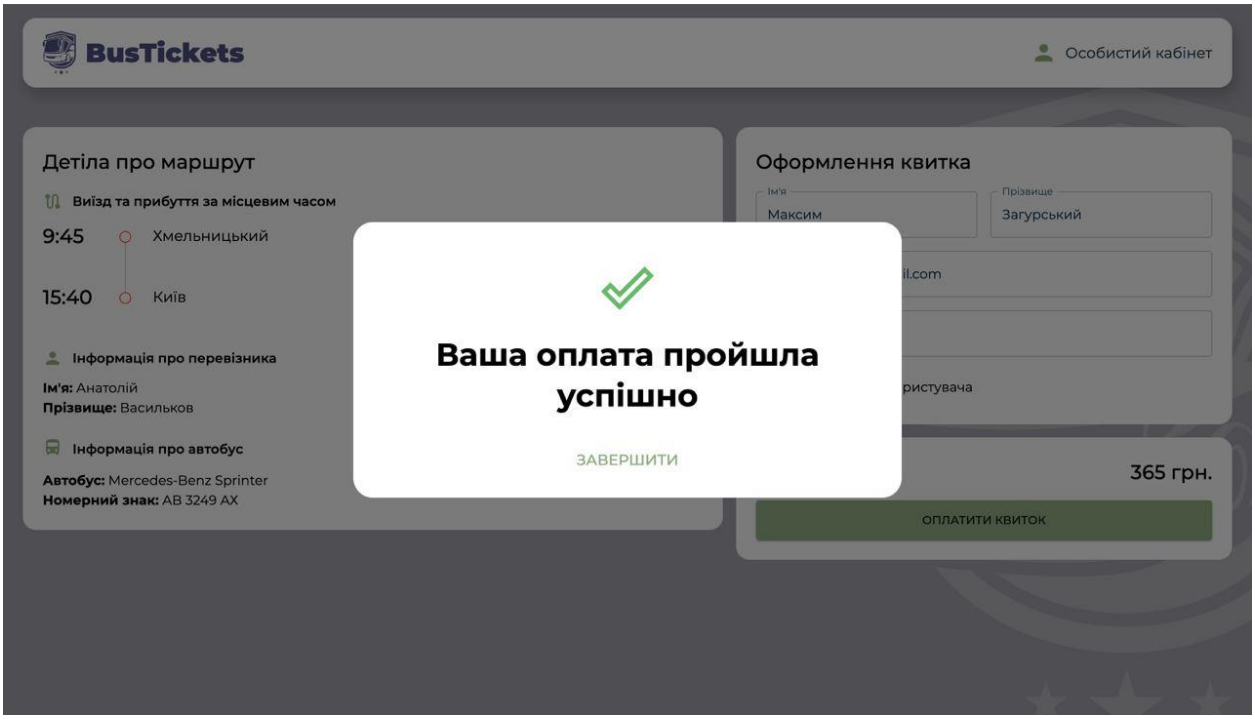

Рисунок 1.29. Вигляд сторінки після успішної оплати квитка

Особистий кабінет пасажира.

У авторизованого та зареєстрованого клієнта є особистий кабінет з історією поїздок, де він може переглядати всі деталі своїх минулих подорожей, включаючи дату, час, місце відправлення та призначення. Крім того, користувач має можливість редагувати свої особисті дані через зручну форму, в яку він може вводити нові або оновлювати існуючі дані, такі як ім'я, прізвище, телефон та електронна пошта.

Також, в особистому кабінеті є функціональність зміни паролю, що дозволяє користувачеві підвищити безпеку свого облікового запису. Користувач вводить старий пароль та записує новий, враховуючи вимоги до безпечного пароля, наприклад, мінімум 8 символів та наявність цифр та спеціальних символів.

Завершуючи сеанс роботи, користувач може безпечно вийти з облікового запису.

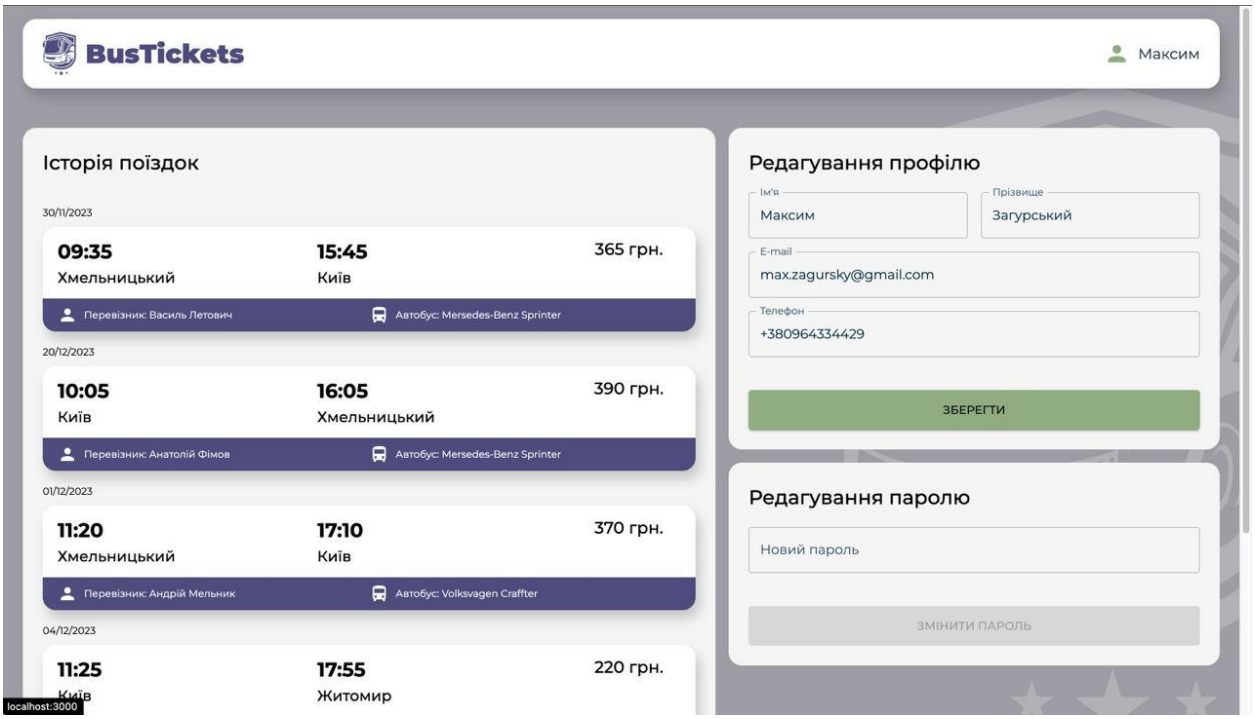

Рисунок 1.30. Особистий кабінет пасажира з історією поїздок

Особистий кабінет перевізника.

Особистий кабінет перевізника представляє собою важливий інструмент, який сприяє ефективному управлінню різними аспектами перевізницької діяльності. Заснований на трьох ключових вкладках, а саме "Квитки", "Маршрути" і "Автобуси", цей інтерфейс надає можливість легко та ефективно керувати своїми поїздками.

На вкладці "Квитки" перевізник може переглядати інформацію про квитки, продані на його маршрутах, а також проаналізувати актуальність майбутніх подорожей та розрахувати прибуток.

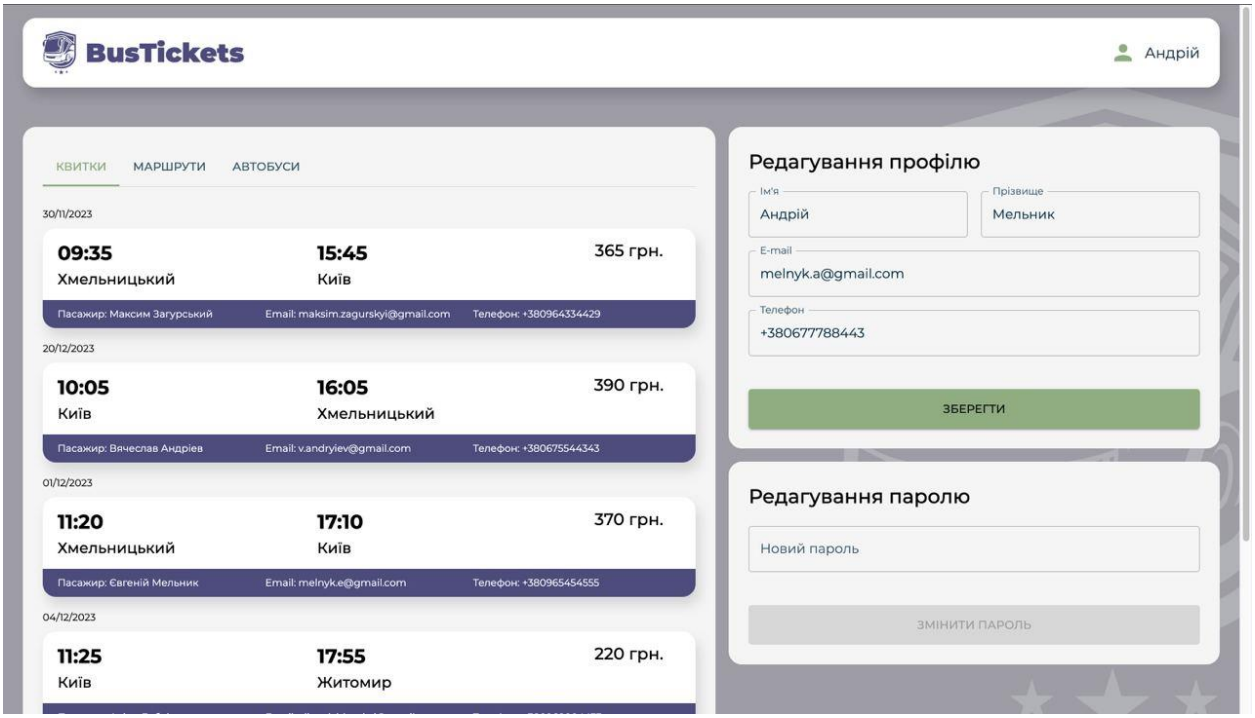

Рисунок 1.31. Профіль перевізника. Список квитків

Розділ "Маршрути" не лише надає можливість перегляду маршрутів, але й дозволяє перевізнику створювати нові маршрути або редагувати існуючі. При створенні нового маршруту можна вказати початкову та кінцеву точки, проміжні зупинки, дату, час, модель автобусу та ціну.

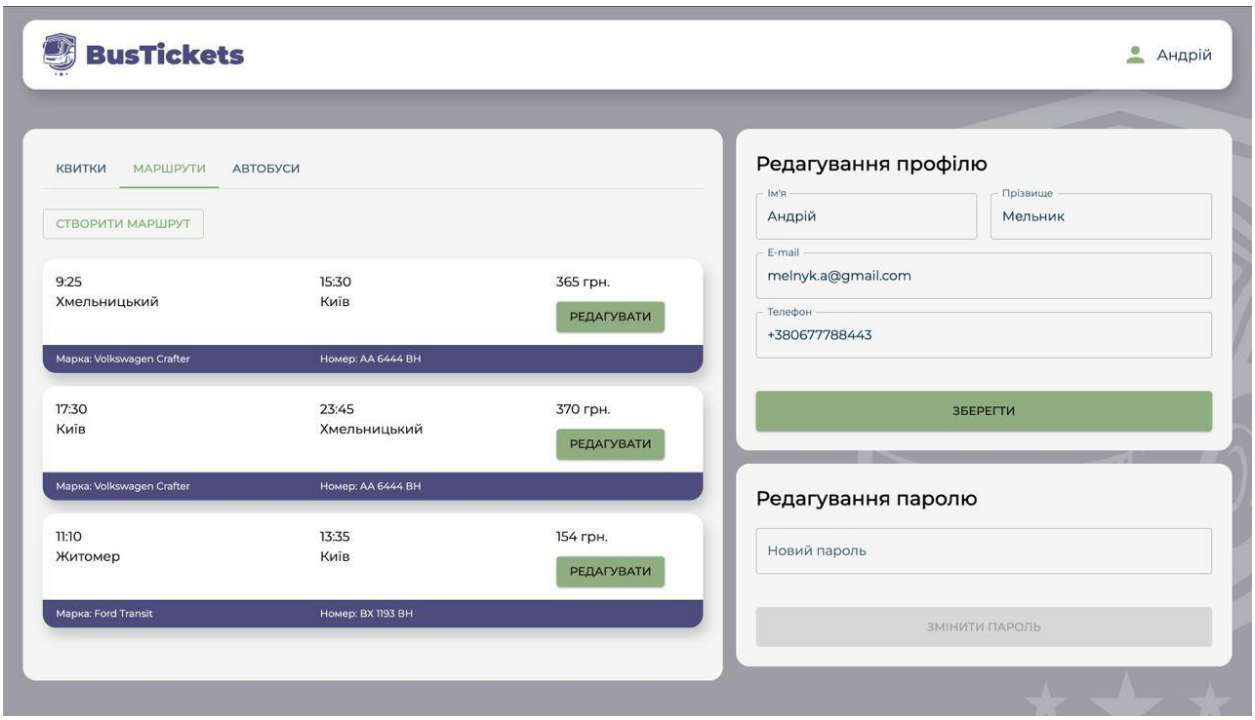

Рисунок 1.32. Список активних маршрутів перевізника

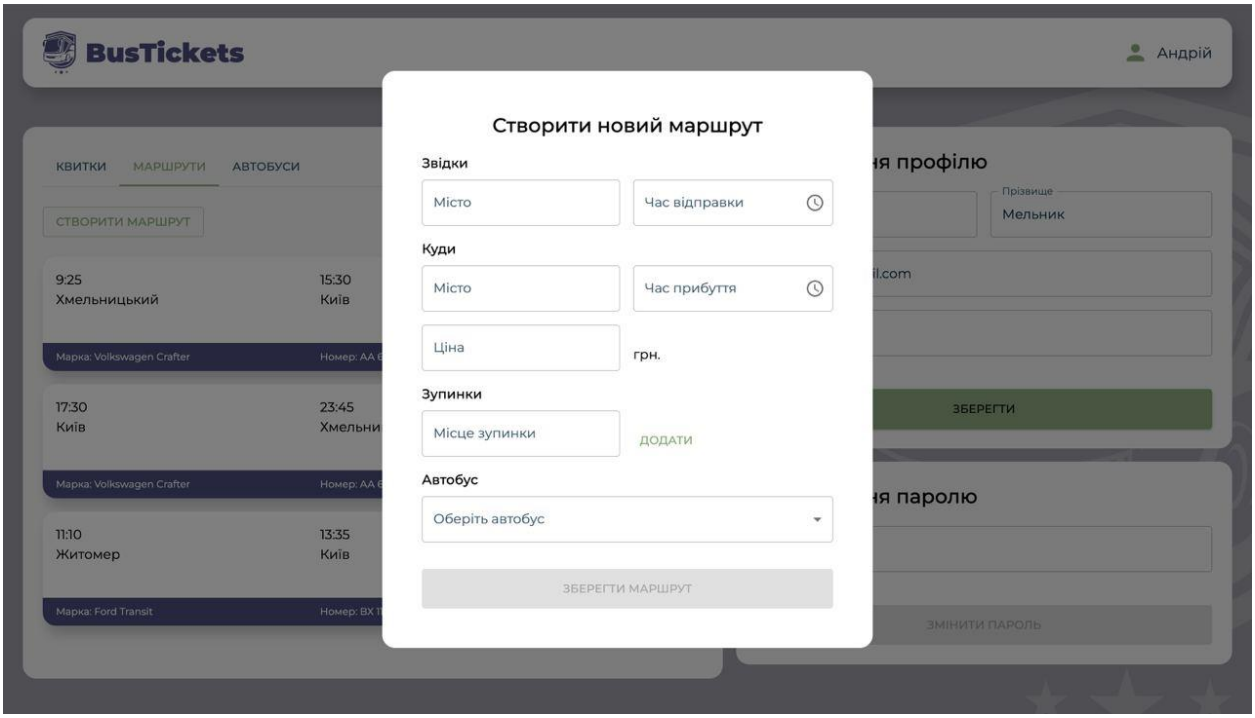

Рисунок 1.33. Створення маршруту

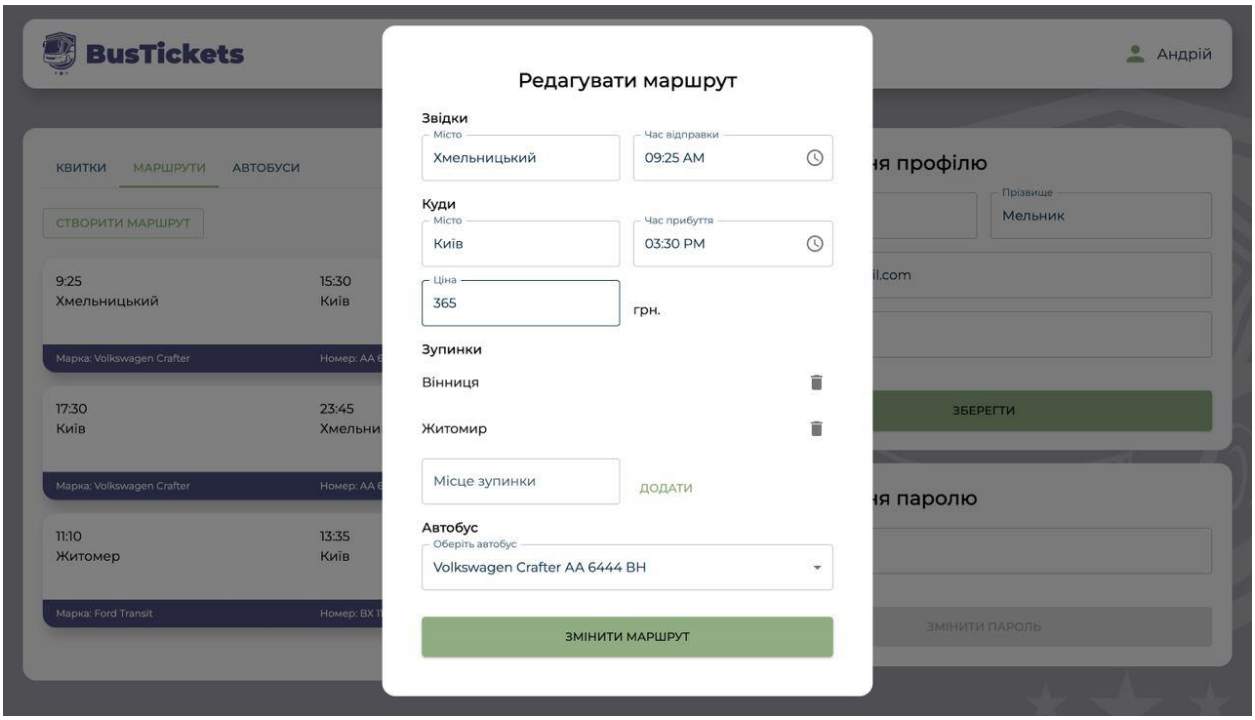

Рисунок 1.34. Редагування маршруту

Вкладка "Автобуси" відкриває перед перевізником можливість управління своїм автобусним парком. Створення нових транспортних засобів та редагування існуючих дозволяє підтримувати високі стандарти безпеки та комфорту для пасажирів.

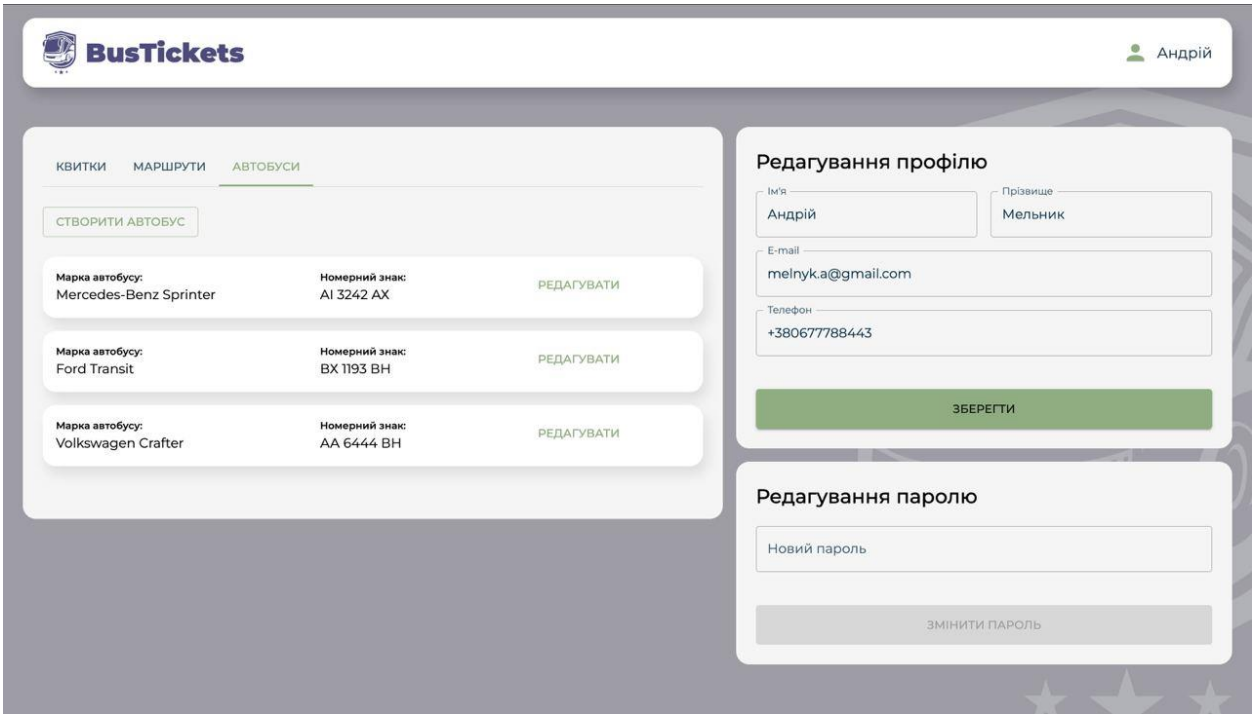

Рисунок 1.35. Список автобусів перевізника

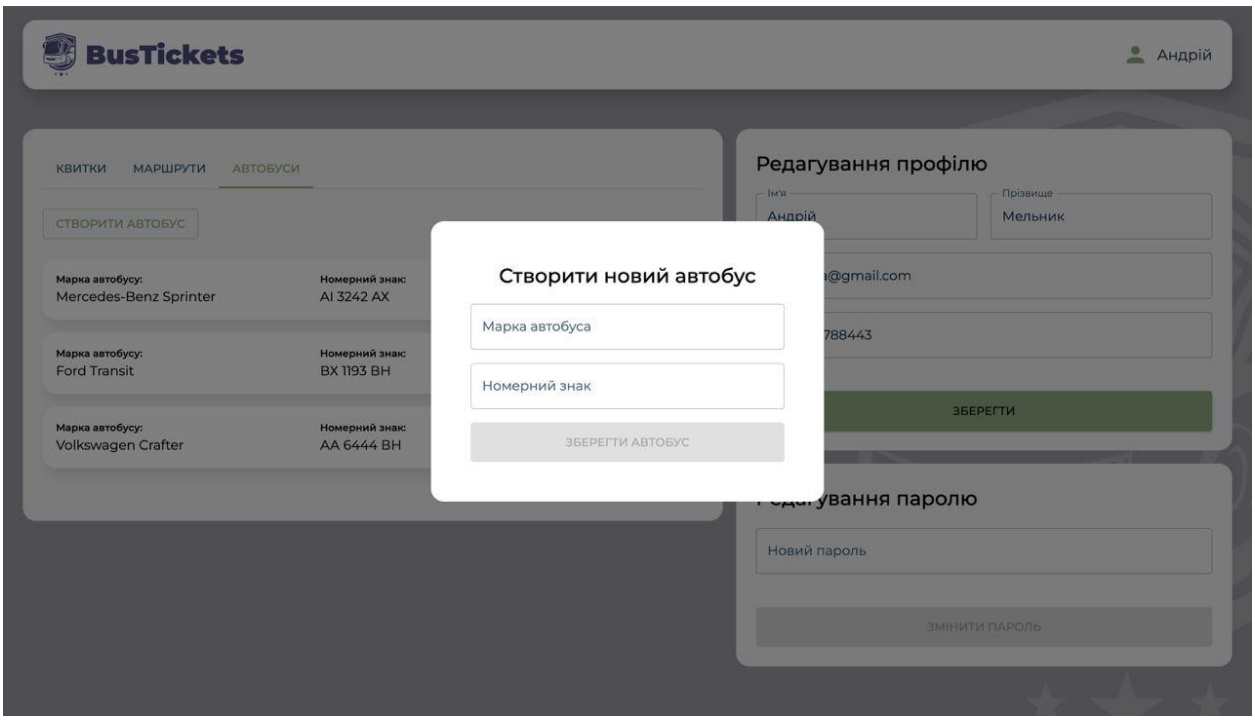

Рисунок 1.36. Створення нового автобусу

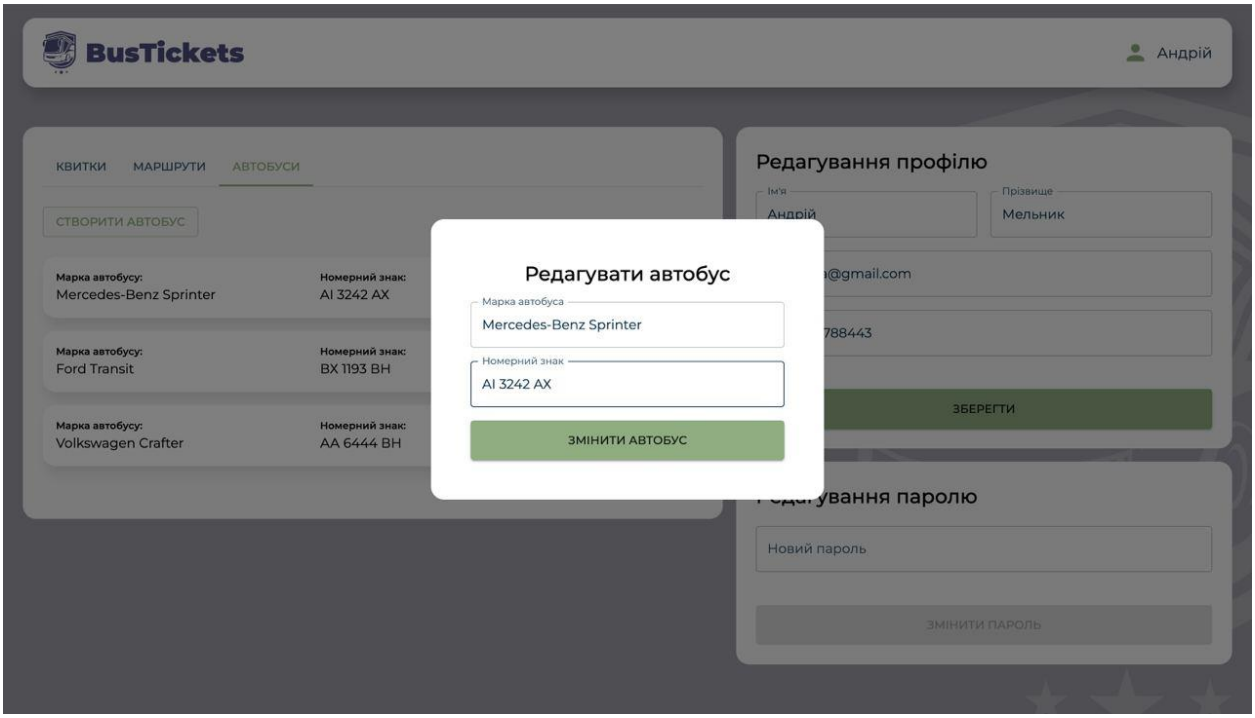

Рисунок 1.37. Редагування автобусу

Крім цього, у особистому кабінеті перевізника можливо редагувати особисті дані та змінювати пароль для підвищення безпеки облікового запису. Це дозволяє забезпечити ефективне керування діяльністю перевізника та зручну адміністрацію його послуг.

Такий особистий кабінет є не лише інструментом управління, але й стратегічним партнером для перевізника, що сприяє його компетентному веденню бізнесу в умовах зростаючого конкурентного середовища.

Висновки до розділу 3

У даному розділі дипломної роботи проведено огляд предметної області, що стосується пасажирських перевезень, і розглянуто загальну інформацію про цей вид транспортних послуг. В результаті аналізу та порівняння аналогічних онлайн сервісів пасажирських перевезень виявлені переваги та недоліки існуючих рішень, що і стало основним фактором у виставлені основних вимоги до веб-додатку.

Представлені діаграми варіантів використання та послідовності, що ілюструють ключові етапи взаємодії користувача з системою.

Структура веб-сервісу пасажирських перевезень описана з точки зору архітектурних компонентів та їх взаємодії, що дозволяє забезпечити оптимальну працездатність системи. Проектування бази даних включає в себе визначення сутностей, їх взаємозв'язків та вибір оптимальних методів зберігання та обробки інформації.

У процесі реалізації проекту були розроблені дизайн та структура, що дозволило створити функціональний та ефективний веб-сервіс для замовлення та керування пасажирськими перевезеннями.

ВИСНОВКИ

У даній дипломній роботі був проведений глибокий аналіз предметної області та аналогічних вебсервісів для розуміння актуальності розробки вебсервісу пасажирських перевезень. Була виконана розробка програмної частини та вивчено всі можливі аспекти безпеки веб-додатків. Поставлену задачу повністю реалізовано та створено стійкий та надійний веб-сервіс для пасажирських перевезень.

У першому розділі роботи визначено важливість вибору архітектури для веб-додатку, детально розглянуті основні види веб-додатків їх переваги та недоліки. Окрема увага приділена архітектурі MVC, MVP, веб-компонентам, а також процесу створення веб-додатку. Проведений аналіз фреймворків для клієнтської та серверної частини веб-додатку визначивши найбільш оптимальні інструменти для реалізації поставлених вимог та завдань.

Другий розділ роботи зосереджений на аналізі використаних технологій у розробці веб-сервісу. Вивчені ключові компоненти стеку технологій MERN (MongoDB, Express.js, React, Node.js), а також проаналізоване середовище розробки VS Code. Це дозволило обґрунтувати вибір конкретних інструментів для успішної реалізації проекту.

У третьому розділі проведено проектування та розробку веб-сервісу для пасажирських перевезень. Визначено діаграми використання та послідовності, описана структура веб-сервісу та розроблена база даних. Реалізація проекту дозволила перевірити та підтвердити сплановані концепції на практиці, що підкреслює актуальність та важливість розглядуваного питання.

Загалом, отримані результати та розроблені рекомендації сприятимуть подальшому розвитку веб-додатків та використанню сучасних технологій у цій сфері. Дана дипломна робота відображає глибоке розуміння теми, високий рівень технічної компетентності та готовність до використання отриманих знань у практичній діяльності.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

1. J. Wang, G. Li, and J. Fe, “Fast-join: An efficient method for fuzzy token matching based string similarity join” 2011. [Електронний ресурс]. – Режим доступу: https://doi.org/10.1109/ICDE.2011.5767865
2. A. Kulkarni, C. More, M. Kulkarni, and V. Bhandekar, “Text Analytic Tools for Semantic Similarity” 2016. [Електронний ресурс]. – Режим доступу: http://imperialjournals.com/index.php/IJIR/article/view/688
3. R. Mihalcea, C. Corley, C. Strapparava, “Corpus-based and knowledge-based measures of text semantic similarity” 2006. [Електронний ресурс]. – Режим доступу: http://www.aaai.org/Papers/AAAI/2006/AAAI06-123.pdf
4. A. Budanitsky , G. Hirst, “Evaluating WordNet-based Measures of Lexical Semantic Relatedness” Comput. Linguist 2006. [Електронний ресурс]. – Режим доступу: https://doi.org/10.1162/coli.2006.32.1.13
5. T. Slimani, “Description and Evaluation of Semantic Similarity Measures Approaches” 2013. [Електронний ресурс]. – Режим доступу: https://doi.org/10.5120/13897-1851
6. J. J. Lastra-Díaz, A. García-Serrano, M. Batet, M. Fernández, and F. Chirigati, “HESML: A scalable ontology-based semantic similarity measures library with a set of reproducible experiments and a replication dataset,” 2017. [Електронний ресурс]. – Режим доступу: https://doi.org/10.1016/j.is.2017.02.002
7. L. Meng, R. Huang, J. Gu, “A review of semantic similarity measures in wordnet,” 2013. [Електронний ресурс]. – Режим доступу: https://pdfs.semanticscholar.org/da95/ceaf335971205f83c8d55f2292463fada4ef.pdf
8. G. Varelas, E. Voutsakis, P. Raftopoulou, E. G. M. Petrakis,E. E. Milios, “Semantic similarity methods in wordNet and their application to information retrieval on the web” 2005. [Електронний ресурс]. – Режим доступу: https://doi.org/10.1145/1097047.1097051

9. Документація серверного середовища. [Електронний ресурс]. – Режим доступу: https://nodejs.org/uk/about/
10. Документація фреймворку серверного середовища. [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/ru/docs/Learn/Serverside/Express_Nodejs/
11. Документація БД. [Електронний ресурс]. – Режим доступу: https://www.guru99.com/what-is-mongodb.html
12. Документація СУБД. [Електронний ресурс]. – Режим доступу: https://mongoosejs.com/
13. Документація фреймоврку клієнта. [Електронний ресурс]. – Режим доступу: https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs