МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

ім. Ю.М. Потебні

ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)

(рівень вищої освіти)

на тему **Побудова веб-сайту надання послуг з Node.js та Express.js**

Виконала студентка 4 курсу, групи 6.1219-пзс-з

спеціальності 121 Інженерія програмного забезпечення

(код і назва спеціальності)

освітньої програми Програмне забезпечення систем

(код і назва освітньої програми)

С.Ю Гнилицька

(підпис, ініціали та прізвище)

Керівник доцент, к.ф.-м.н. В.І. Попівщий

(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Рецензент директор ТОВ «Дискус» П.О. Лютий

(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя

2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ім. Ю.М. Потебні ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення (код та назва)
Освітня програма Програмне забезпечення систем (код та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри Тетяна КРИТСЬКА
“_01_” __березня 2024 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Гнилицькій Світлані Юріївні

(прізвище, ім’я, по батькові)

1.Тема роботи Побудова веб-сайту надання послуг з Node.js та Express.js

керівник роботи Попівщий Василь Іванович, доцент, к.ф.-м.н.

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від 26.12.2023 р. № 2212-с

Строк подання студентом кваліфікаційної роботи 28 травня 2024 р.

3. Вихідні дані бакалаврської роботи

- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми збору та аналізу даних з веб-сторінок;
- створення програмного продукту та його опис;
- дослідження поставленої проблеми та розробка висновків.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи бакалавра	Примітка
1	Аналіз предметної області	28.11 - 05.12.2023	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	06.12 - 25.12.2023	виконано
3	Аналіз існуючих методів рішення	26.12 - 10.01.2024	виконано
4	Дослідження існуючих засобів для реалізації Веб-сайтів	11.01 - 01.02.2024	виконано
5	Узгодження подальших дій з науковим керівником	19.02 - 01.03.2024	виконано
6	Реалізація застосунку на основі Node.js	02.03 - 05.04.2024	виконано
7	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	02.05 - 10.05.2024	виконано
8	Усунення додаткових проблем та «багів»	11.05 - 13.05.2024	виконано
9	Тестування та порівняння показників	14.05 - 15.05.2024	виконано
10	Розгортання сайту	16.05 - 17.05.2024	виконано
11	Оформлення звіту	18.05 - 23.05.2024	виконано

Студентка С.Ю. Гнилицька

(підпис) (ініціали та прізвище)

Керівник роботи В.І. Попівщий

(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер І.А. Скрипник

(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Сторінок - 78

Рисунків - 17

Джерел - 16

Гнилицька С.Ю. Побудова веб-сайту надання послуг з Node.js та Express.js: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник В. І. Попівщий. Запоріжжя : ЗНУ, 2023. 78 с.

Мета полягає у розробці функціонального веб-сайту для надання послуг, використовуючи Node.js та Express.js для забезпечення ефективної серверної логіки та обробки запитів.

У процесі розробки була розглянута проблема створення повнофункціонального веб-сайту для надання туристичних послуг, який включав би різноманітні сторінки для відгуків, турів, гідів, та забезпечував би можливість користувачам подавати заявки і залишати коментарі. У результаті був розроблений веб-сайт, який використовує Node.js та Express.js для серверної логіки та маршрутизації, має базу даних на основі Sequelize для зберігання та обробки даних заявок і відгуків. Веб-сайт включає сторінки для перегляду турів, гідів, цікавих місць, та подання заявок, забезпечує можливість користувачам залишати відгуки, які відображаються на сайті. Розроблений веб-сайт має адміністративний інтерфейс для перегляду заявок, захищений паролем. Має налаштовані стилі для зручного та привабливого інтерфейсу користувача.

Ключові слова: *Node.js, Express.js, веб-сайт, туристичні послуги, маршрутизація, база даних, Sequelize, заявки, відгуки, інтерфейс користувача, адміністративний інтерфейс, стилі, коментарі, тури, гіди, цікаві місця, захист даних, форми submission, асинхронна обробка, Full StacDevelopmen*

ABSTRACT

Pages - 78

Drawings - 17

Sources - 16

Hnylytska S.Yu. Development of a Service Provision Website Using Node.js and Express.js: Bachelor's Thesis in the Specialty 121 "Software Engineering" / Scientific Supervisor V.I. Popivshyi. Zaporizhzhia: ZNU, 2023. 78 p.

The goal is to develop a functional website for service provision using Node.js and Express.js to ensure efficient server-side logic and request handling.

During the development process, the challenge of creating a fully functional website for providing tourism services was addressed. The website was designed to include various pages for reviews, tours, and guides, and to allow users to submit applications and leave comments. As a result, a website was developed using Node.js and Express.js for server-side logic and routing, with a Sequelize-based database for storing and processing application and review data. The website includes pages for viewing tours, guides, and points of interest, as well as submitting applications. It allows users to leave reviews, which are displayed on the site. The developed website has an administrative interface for viewing applications, which is password-protected, and it features customized styles for a user-friendly and attractive interface.

Keywords: *Node.js, Express.js, website, tourism services, routing, database, Sequelize, applications, reviews, user interface, administrative interface, styles, comments, tours, guides, points of interest, data protection, form submission, asynchronous processing, Full Stack Development.*

ЗМІСТ

ВСТУП

Актуальність теми

Актуальність цієї теми обумовлена зростаючим попитом на цифрові рішення в туристичній індустрії. У сучасному світі клієнти очікують зручності, швидкості та ефективності при плануванні подорожей і бронюванні послуг. Розробка веб-сайтів з використанням технологій Node.js та Express.js забезпечує високу продуктивність, масштабованість і гнучкість у розробці веб-додатків, що є важливими для конкурентоспроможності на ринку.

На сьогоднішній день стрімко зростає попит на цифрові рішення в туристичній індустрії. Створення функціональних веб-сайтів, що забезпечують ефективну обробку запитів і зберігання даних, сприяє покращенню якості обслуговування клієнтів і розширенню бізнес-можливостей. Використання сучасних технологій, таких як Node.js та Express.js, дозволяє створювати масштабовані та надійні сервіси, що відповідають вимогам сучасного ринку.

Основними проблемами, які виникають при розробці веб-сайтів для надання туристичних послуг, є:

1. Забезпечення ефективної обробки великої кількості запитів від користувачів.
2. Інтеграція з різними зовнішніми API та сервісами.
3. Збереження і обробка великого обсягу даних заявок і відгуків.
4. Забезпечення безпеки даних користувачів.
5. Забезпечення зручного і інтуїтивно зрозумілого інтерфейсу для користувачів і адміністративного персоналу.

Основні переваги використання Node.js та Express.js для розробки веб-сайтів включають високу продуктивність та швидкість обробки запитів завдяки асинхронній обробці, легку масштабованість додатків для обробки

великої кількості одночасних запитів, широку екосистему пакетів та модулів, що спрощує інтеграцію з іншими сервісами, зручність використання однієї мови програмування (JavaScript) для фронтенду та бекенду, гнучкість у розробці і налаштуванні серверної логіки та маршрутизації. Загалом, розробка веб-сайту для надання туристичних послуг з використанням Node.js та Express.js є ефективним рішенням, яке дозволяє створювати сучасні, продуктивні та безпечні веб-додатки. Це сприяє покращенню якості обслуговування клієнтів, підвищенню конкурентоспроможності бізнесу і відповідає вимогам сучасного цифрового ринку.

Мета дослідження

Розробка функціонального, зручного та інтуїтивного зрозумілого інтерфейсу веб-сайту для надання туристичних послуг.

Завдання дослідження

Проаналізувати вимоги до функціонального веб-сайту для надання туристичних послуг. Розробка архітектури веб-сайту, що використовує Node.js та Express.js. Реалізувати серверну логіку та маршрутизацію запитів з використанням Node.js та Express.js та інтегрувати базу даних для зберігання і управління даними заявок та відгуків, використовуючи Sequelize. Розробити інтерфейс користувача, який включатиме сторінки для перегляду турів, гідів та подання заявок. Забезпечити можливість користувачам залишати відгуки та коментарі, які відображатимуться на сайті. Створення адміністративного інтерфейсу для перегляду заявок і керування контентом, захищений паролем. Забезпечити належний рівень безпеки для захисту даних користувачів, впровадити стильове оформлення для зручного та привабливого інтерфейсу. Тестувати і оптимізувати веб-сайт для забезпечення його продуктивності і надійності.

Об’єкт дослідження

Об'єктом дослідження є процес розробки веб-сайту надання туристичних послуг з Node.js та Express.js.

Предмет дослідження

Предметом дослідження є вивчення основних принципів роботи з Node.js та Express.js у Microsoft Visuale Studio Code.

Методи дослідження

Аналіз літератури та огляд існуючих рішень. Дослідження наявних методів та інструментів для розробки веб-сайтів, зокрема з використанням Node.js та Express.js, а також аналіз сучасних тенденцій у сфері туристичних послуг.

Практичне значення одержаних результатів

Практичне значення одержаних результатів полягає в конкретних користувальницьких і підприємницьких вигодах, які можуть бути здобуті завдяки розробленому веб-сайту для надання туристичних послуг.

Можливість швидко та зручно знаходити та бронювати туристичні послуги через веб-сайт. Адміністратори отримають інструменти для управління замовленнями, відгуками та іншою інформацією. Зручний та зрозумілий інтерфейс на веб-сайті допоможе створити позитивний клієнтський досвід, що важливо для підтримання лояльності клієнтів.

Глосарій

Веб-сайт - це колекція веб-сторінок, що доступні через Інтернет і розміщені на одному домені. Веб-сайт може містити різноманітний контент, такий як текст, зображення, відео, форми зв'язку та інші елементи, призначені для комунікації з користувачем.

Node.js - це середовище виконання JavaScript, яке дозволяє запускати код JavaScript на сервері. Node.js базується на подіях та асинхронному введенні/виведенні, що робить його ефективним для розробки масштабованих та швидких веб-додатків.

Express.js - це веб-фреймворк для Node.js, який дозволяє швидко створювати веб-додатки та веб-сервери. Express.js надає простий та ефективний спосіб створення маршрутів для обробки HTTP-запитів, обробки шаблонів, керування сеансами та багато іншого, що спрощує розробку веб-додатків на Node.js.

Sequelize - це ORM (Object-Relational Mapping) для Node.js, яке дозволяє зручно взаємодіяти з базою даних, використовуючи об'єктно-орієнтований підхід.

Адміністративний інтерфейс - це частина веб-сайту, призначена для адміністраторів, де вони можуть переглядати та керувати контентом, відстежувати замовлення та здійснювати інші адміністративні функції.

Маршрутизація - процес визначення шляхів URL і визначення того, який код виконуватиметься для кожного URL на веб-сайті.

Захист даних - набір практик та методів, спрямованих на забезпечення безпеки даних, включаючи шифрування, аутентифікацію, авторизацію та інші заходи безпеки.

База даних (Database) - структурована колекція даних, яка зберігається та управляється за допомогою комп'ютерної системи.

SQL (Structured Query Language) - мова запитів, яка використовується для взаємодії з реляційними базами даних, такими як MySQL, PostgreSQL, SQLite тощо.

ORM (Object-Relational Mapping) - технологія програмування, яка дозволяє зв'язувати об'єкти програми з записами у базі даних, що спрощує роботу з даними та уникнення роботи з SQL-запитами безпосередньо.

Модель даних (Data Model) - структура, що визначає спосіб організації та представлення даних у базі даних, включаючи сутності, атрибути та відносини між ними.

Схема бази даних (Database Schema) - формальний опис структури та організації даних у базі даних, включаючи таблиці, поля, відносини та обмеження.

HTML/CSS - Мови розмітки та стилізації, які використовуються для створення структури та дизайну веб-сторінок.

JavaScript - мова програмування, яка використовується для додавання інтерактивності та динамічних функцій на веб-сторінці.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд літературних джерел

Мета створення туристичної веб-сторінки полягає в наданні користувачам зручного та ефективного інструменту для планування та бронювання подорожей та туристичних послуг. Ця веб-сторінка може слугувати як платформа для представлення туристичних атракцій, маршрутів, послуг гідів та іншої корисної інформації для подорожуючих. Актуальність туристичної веб-сторінки пояснюється зростанням популярності туризму як відпочинку. За останні роки туризм став однією з найбільш динамічно розвиваючихся галузей, що викликає зростання попиту на інформацію про подорожі та послуги. Така веб-сторінка може стати важливим інструментом як для індивідуальних подорожей, так і для бізнес-подорожей та організації туристичних подій.

Основні теми, які слід приділити увагу при створенні туристичної веб-сторінки, включають:

1. Каталог турів та послуг: Зручний та організований перегляд доступних турів, екскурсій, відпочинку та інших послуг.
2. Бронювання: Можливість користувачам зареєструватися на сайті та бронювати тури та послуги онлайн.
3. Відгуки: Можливість користувачів залишати відгуки про свої подорожі та оцінювати якість послуг.
4. Інформація про місцевість: Представлення корисної інформації про місцевість, її культуру, традиції, гастрономію та інші цікаві аспекти.
5. Безпека та конфіденційність: Забезпечення високого рівня безпеки та захисту персональних даних користувачів, зокрема під час проведення транзакцій та обробки платежів.

Дослідження та написання туристичної веб-сторінки мають велику важливість з різних точок зору: Задоволення потреб користувачів, розвиток туристичного бізнесу, продвиження місцевого туризму. Туристична веб-сторінка надає користувачам зручний та ефективний інструмент для планування та бронювання подорожей та туристичних послуг. Вона допомагає задовольнити потреби користувачів у пошуку та отриманні інформації про подорожі, а також у резервуванні турів та послуг. Туристична веб-сторінка може сприяти розвитку туристичного бізнесу, привертаючи більше клієнтів. Вона може стати важливим інструментом для туристичних агентств, готелів, туроператорів та інших гравців на ринку туризму. Туристична веб-сторінка може сприяти просуванню місцевого туризму, привертаючи увагу до різноманітних туристичних атракцій та послуг у певній локації. Це може сприяти розвитку туристичної індустрії та збільшенню доходів для місцевих підприємств та громад.

Задача написання туристичної веб-сторінки полягає у створенні функціонального, зручного та привабливого веб-додатку, який відповідає потребам користувачів та бізнесу. Це включає в себе розробку інтерфейсу користувача, реалізацію функціональності пошуку та бронювання турів та послуг, інтеграцію з платіжними системами, забезпечення безпеки та конфіденційності даних користувачів та багато іншого. У великій мірі успіх туристичної веб-сторінки залежить від якості та ефективності її реалізації.

Користування туристичним веб-сайтом може виникнути ряд проблем, з якими користувачі можуть зіткнутися, як складність навігації: Якщо веб-сайт має складну структуру або неясну навігацію, користувачі можуть мати проблеми з пошуком потрібної інформації або функціоналу.

Користувачі також можуть стикатися з проблемами бронювання. Технічні недоліки або складнощі з формами бронювання можуть призвести до невдалих спроб здійснення бронювання послуг. Невідповідність очікуванням також є поширеною проблемою. Якщо інформація на веб-сайті не відповідає

реальності або якості послуг не відповідає очікуванням користувачів, це може призвести до негативного досвіду користування.

У розробці туристичного веб-сайту можуть виникнути такі проблеми: складність інтеграції зовнішніх сервісів, безпека та захист даних, оптимізація швидкості завантаження. Розробка безпечного механізму аутентифікації, захисту особистих даних користувачів та забезпечення безпеки транзакцій є важливою, але складною задачею. Великий обсяг зображень та даних на веб-сайті може призвести до повільної швидкості завантаження сторінок, що може вплинути на користувальний досвід.

Розуміння цих проблем дозволяє розробникам створити більш зручний та ефективний туристичний веб-сайт, який задовольнить потреби користувачів і максимально уникне можливих проблем.

1.2 Аналіз програмного забезпечення

1.2.1 Node.js

Node.js - середовище виконання JavaScript для серверного програмування, яке дозволяє створювати масштабовані та високопродуктивні серверні додатки.

Node.js включає в себе кілька важливих інструментів і функцій, які роблять його потужним середовищем для серверного програмування:

1. NPM (Node Package Manager): Менеджер пакетів, який дозволяє легко встановлювати, оновлювати та управляти бібліотеками та модулями, що використовуються в проекті.
2. Бібліотека стандартних модулів: Node.js постачається з набором вбудованих модулів (наприклад, http, fs, `path`), які забезпечують базову функціональність для роботи з HTTP-запитами, файловою системою, шляхами та іншими необхідними задачами.

3. Подійно-орієнтована модель: Node.js використовує подійно-орієнтовану архітектуру, що дозволяє обробляти асинхронні операції (такі як запити до бази даних або файлової системи) без блокування основного потоку виконання.
4. Асинхронна обробка: Завдяки використанню неблокуючого вводу/виводу (non-blocking I/O), Node.js може обробляти велику кількість одночасних з'єднань, що робить його ідеальним для створення масштабованих мережевих додатків.

Node.js використовує мову програмування JavaScript, яка є однією з найпопулярніших мов програмування в світі. Node.js дозволяє розробляти серверні додатки. Завдяки можливості запуску JavaScript на сервері, Node.js дозволяє створювати серверні додатки, які можуть обробляти HTTP-запити, керувати сесіями, працювати з базами даних та виконувати інші серверні задачі.

За допомогою Node.js можна створювати високопродуктивні мережеві додатки - завдяки подійному циклу та неблокуючому вводу/виводу, Node.js підходить для розробки додатків, що потребують обробки великої кількості одночасних з'єднань, таких як чати, реальні ігри та інші реального часу додатки. Використовування однієї мови для фронтенду та бекенду є дуже зручним. Оскільки JavaScript використовується як для клієнтської (фронтенд), так і для серверної (бекенд) розробки, це спрощує розробку, дозволяючи використовувати одну мову для всього стеку додатку. Node.js дозволяє швидко створювати та розгортати додатки. Завдяки великій кількості доступних модулів та бібліотек у NPM, а також простоті JavaScript, розробка додатків на Node.js може бути швидкою та ефективною.

Ці особливості роблять Node.js потужним інструментом для сучасної веб-розробки, дозволяючи створювати масштабовані, продуктивні та зручні додатки.

HTML (HyperText Markup Language) та CSS (Cascading Style Sheets) є основними технологіями для створення веб-сторінок, і вони відіграють важливу роль у розробці фронтенду, тоді як Node.js зазвичай використовується для бекенду.

HTML є мовою розмітки, яка використовується для створення структури веб-сторінок. Вона визначає різні елементи, такі як заголовки, параграфи, зображення, посилання та форми. HTML є основою будь-якої веб-сторінки, оскільки він надає базову структуру, яку браузери відображають користувачам.

CSS є мовою стилів, яка використовується для оформлення та розташування елементів HTML на веб-сторінці. CSS дозволяє визначати кольори, шрифти, відступи, розміри, макети та інші візуальні аспекти веб-сторінки. Це дозволяє створювати привабливі та зручні інтерфейси користувача.

Node.js зазвичай використовується для створення серверної частини веб-додатків. Node.js може використовуватися для серверного рендерингу HTML. Це означає, що сервер генерує HTML-код, який потім надсилається клієнту (браузеру). Наприклад, за допомогою шаблонних двигунів, таких як Handlebars або EJS, можна динамічно створювати HTML на основі даних, які отримуються з бази даних або інших джерел. Node.js разом з Express.js може обробляти HTTP-запити від клієнтів (браузерів) і надсилати відповідні відповіді у вигляді HTML-сторінок або інших типів даних (наприклад, JSON для API).

Node.js може слугувати для роздачі статичних файлів, таких як HTML, CSS, JavaScript та зображення. Це дозволяє зберігати всі ресурси веб-додатку на сервері та надавати їх користувачам за запитом. Node.js може створювати API, які забезпечують динамічний вміст для веб-сторінок. Наприклад, JavaScript на фронтенді може робити запити до серверного API, створеного за

допомогою Node.js, щоб отримати та відобразити дані без необхідності перезавантаження сторінки.

1.2.2 Express.js

Express.js — це мінімалістичний і гнучкий веб-фреймворк для Node.js, який полегшує розробку серверних додатків і API. Він є популярним вибором для створення серверних частин веб-додатків завдяки своїй простоті, зручності використання та багатому набору функцій.

Express.js використовується для створення серверних додатків, які можуть обробляти HTTP-запити, маршрутизувати їх до відповідних обробників, взаємодіяти з базами даних, а також надавати статичні ресурси (HTML, CSS, JavaScript файли) клієнтам (браузерам).

Express.js дозволяє виконувати ряд важливих завдань у розробці серверних додатків:

1. Маршрутизація: - Express.js надає потужну систему маршрутизації, яка дозволяє визначати маршрути для обробки різних HTTP-запитів (GET, POST, PUT, DELETE тощо).
2. Обробка запитів і відповідей: - Express.js дозволяє легко обробляти дані з запитів (наприклад, параметри запиту, тіла POST-запитів) і формувати відповіді клієнтам. - Підтримка middleware дозволяє обробляти запити на різних етапах їх обробки (наприклад, аутентифікація, логування, обробка помилок).
3. Шаблонізація: - Express.js підтримує використання шаблонних двигунів (наприклад, Pug, EJS, Handlebars), що дозволяє динамічно створювати HTML-сторінки на основі даних.
4. Статичні файли: - Express.js може використовуватися для роздачі статичних файлів (наприклад, HTML, CSS, JavaScript, зображення) клієнтам.

5. Підтримка RESTful API: - Express.js ідеально підходить для створення RESTful API, що дозволяє легко створювати, читати, оновлювати та видаляти ресурси через HTTP-запити.

Переваги Express.js

- Простота та гнучкість: Express.js є простим у використанні і легко налаштовується, що робить його чудовим вибором як для початківців, так і для досвідчених розробників.
- Масштабованість: Завдяки своїй модульній структурі, Express.js дозволяє легко додавати нову функціональність за допомогою middleware та інших плагінів.
- Широке прийняття та підтримка: Велика спільнота та багата екосистема пакетів роблять Express.js надійним та добре підтримуваним інструментом.

Express.js є ключовим інструментом у стеку MEAN/MERN (MongoDB, Express.js, Angular/React, Node.js) і широко використовується для розробки сучасних веб-додатків.

1.2.3 Sequelize

Sequelize — це ORM (Object-Relational Mapping) бібліотека для Node.js, яка дозволяє розробникам працювати з реляційними базами даних, такими як MySQL, PostgreSQL, SQLite та MSSQL, використовуючи об'єктно-орієнтований підхід. Це значно спрощує процес взаємодії з базами даних, дозволяючи уникнути написання великої кількості сирих SQL-запитів і забезпечуючи зручну роботу з даними у вигляді JavaScript об'єктів.

Основні можливості Sequelize:

1. Моделі: Sequelize дозволяє визначати моделі, які представляють таблиці в базі даних. Кожна модель описує структуру таблиці та її атрибути.
2. Запити: Sequelize надає методи для виконання CRUD (Create, Read, Update, Delete) операцій і складних запитів, таких як фільтрація, сортування та агрегація даних.
3. Валідація: Вбудована валідація даних на рівні моделей дозволяє перевіряти дані перед їх збереженням у базі даних.
4. Асоціації: Sequelize дозволяє визначати зв'язки між моделями, такі як один-до-одного, один-до-багатьох та багато-до-багатьох, що полегшує роботу зі складними реляційними структурами.
5. Міграції: Sequelize має систему міграцій, яка дозволяє управляти змінами в структурі бази даних, відслідковувати ці зміни та застосовувати їх поетапно.
6. Синхронізація: Sequelize може автоматично створювати таблиці на основі визначених моделей, що значно спрощує початкове налаштування бази даних.

Переваги використання Sequelize є зручність роботи з базою даних: Sequelize дозволяє працювати з базою даних на високому рівні абстракції, зменшуючи кількість коду, який потрібно написати для взаємодії з базою даних; об'єктно-орієнтований підхід: Це дозволяє працювати з даними у вигляді JavaScript об'єктів, що робить код більш зрозумілим і підтримуваним; масштабованість: Sequelize підтримує складні зв'язки між таблицями та складні запити, що дозволяє створювати масштабовані та функціональні додатки; підтримка міграцій: Система міграцій дозволяє легко управляти змінами в базі даних і зберігати її структуру в актуальному стані.

Sequelize є потужним інструментом для розробників, які працюють з реляційними базами даних у середовищі Node.js, і значно спрощує процес розробки, управління даними та забезпечення цілісності даних у додатках.

1.3 Постановка завдання

Мета цієї роботи Метою цієї роботи є розробка повнофункціонального веб-сайту для надання туристичних послуг, використовуючи Node.js та Express.js для забезпечення ефективної серверної логіки та обробки запитів. Веб-сайт повинен дозволяти користувачам переглядати доступні тури та гідів, подавати заявки на тури, залишати відгуки та коментарі, а також включати адміністративний інтерфейс для керування заявками.

Етапи дослідження

1. Аналіз вимог: - Визначення основних функцій і можливостей, які повинен мати веб-сайт. - Вивчення потреб користувачів і аналіз схожих існуючих рішень.
2. Проектування архітектури: - Розробка архітектури веб-сайту, включаючи серверну частину на базі Node.js та Express.js. - Вибір технологій для фронтенду, бази даних та інших необхідних компонентів.
3. Розробка бази даних: - Проектування схеми бази даних для зберігання інформації про тури, гідів, користувачів, заявки та відгуки. - Вибір та налаштування системи управління базами даних (наприклад, MySQL або PostgreSQL). - Використання ORM (Sequelize) для взаємодії з базою даних.
4. Розробка серверної частини: - Створення RESTful API для обробки запитів користувачів. - Реалізація маршрутизації запитів за допомогою Express.js. - Забезпечення обробки даних та взаємодії з базою даних.
5. Розробка клієнтської частини: - Створення інтерфейсу користувача за допомогою HTML, CSS та JavaScript. - Забезпечення зручної навігації та інтуїтивно зрозумілого користувацького досвіду.
6. Інтеграція та тестування: - Інтеграція клієнтської та серверної частин. - Проведення функціонального тестування для забезпечення коректної

роботи всіх компонентів веб-сайту. - Тестування на відповідність вимогам безпеки та продуктивності.

7. Зворотний зв'язок та покращення: - Збір відгуків від користувачів та аналіз результатів. - Внесення покращень на основі зворотного зв'язку та подальше оновлення веб-сайту.

1.4 Висновки до розділу 1

1. Було оглянуто процес розробки повнофункціонального туристичного веб-сайту за допомогою Node.js та Express.js.
2. Проаналізовані етапи розробки, включаючи аналіз вимог, проектування архітектури, створення бази даних, розробку серверної та клієнтської частин, інтеграцію, тестування, розгортання і подальше покращення веб-сайту.
3. Визначена постановка завдання дослідження.

2 ТЕХНОЛОГІЇ РОЗРОБКИ ВЕБ-САЙТІВ

2.1 Аналіз сучасних технологій для розробки веб-сайту

У процесі розробки сучасних веб-сайтів важливо вибирати відповідні технології, які забезпечують ефективність, масштабованість, зручність у використанні та підтримку. Тому було обрано основні технології, які використовуються для розробки веб-сайтів:

1. Node.js: Серверне середовище виконання JavaScript, яке дозволяє розробникам використовувати JavaScript для серверної частини веб-додатків. Асинхронна обробка запитів, висока продуктивність, багата екосистема пакетів NPM, одна мова для серверної і клієнтської частин. Ідеально підходить для побудови реальних додатків, RESTful API, і додатків з високими вимогами до масштабованості.
2. Express.js: Легкий і гнучкий фреймворк для Node.js, який спрощує розробку серверних додатків і API. Простота і гнучкість, багатий набір функцій для маршрутизації, обробки запитів і відповіді, підтримка middleware. Використовується для створення серверної логіки, обробки запитів та створення RESTful API.
3. Sequelize: ORM бібліотека для Node.js, яка дозволяє взаємодіяти з реляційними базами даних, такими як MySQL, PostgreSQL, SQLite та MSSQL. Спрощує роботу з базами даних, об'єктно-орієнтований підхід, підтримка асоціацій, валідацій та міграцій. Використовується для управління базою даних, зберігання і обробки даних заявок і відгуків.
4. HTML & CSS: HTML (HyperText Markup Language) використовується для створення структури веб-сторінок, а CSS (Cascading Style Sheets) — для стилізації і оформлення. Відповідають стандартам веб-розробки, широке використання і підтримка, гнучкість у створенні дизайну та

макетів. Використовуються для створення і оформлення користувацького інтерфейсу веб-сайту.

5. JavaScript (на клієнтській стороні): Мова програмування, яка використовується для створення динамічної і інтерактивної поведінки на веб-сторінках. Виконується безпосередньо в браузері, дозволяє створювати інтерактивні елементи і покращувати користувацький досвід. Використовується для реалізації інтерактивності на клієнтській частині веб-сайту.
6. Реляційні бази даних (наприклад, MySQL, PostgreSQL): Бази даних, які використовують SQL (Structured Query Language) для управління даними. Висока продуктивність, підтримка складних запитів, забезпечення цілісності даних. Зберігання інформації про тури, гідів, користувачів, заявки та відгуки.

Використання Node.js, Express.js, Sequelize, HTML, CSS, JavaScript та реляційних баз даних забезпечує ефективний і надійний процес розробки сучасних веб-сайтів. Ці технології дозволяють створювати масштабовані, продуктивні та зручні для користувачів веб-додатки, що відповідають сучасним вимогам.

По темі побудови веб-сайту надання послуг з використанням Node.js та Express.js я обрала тематику туризму. Цей вибір обумовлений актуальністю та популярністю туризму як галузі, яка активно розвивається та потребує сучасних технологічних рішень.

Туризм є однією з найбільш динамічних і перспективних сфер, що сприяє економічному розвитку та культурному обміну. Веб-сайт, орієнтований на надання туристичних послуг, може значно полегшити процес планування подорожей, пропонуючи користувачам зручний інструмент для пошуку турів, бронювання послуг та отримання відгуків.

Таким чином, створення туристичного веб-сайту за допомогою Node.js та Express.js дозволяє продемонструвати можливості цих технологій у контексті реального і значущого прикладу.

2.2 Аналіз вимог до користувачів туристичного веб-сайту

Аналіз вимог до користувачів є важливим етапом у розробці туристичного веб-сайту, оскільки він дозволяє визначити, які функції і сервіси потрібні для задоволення потреб користувачів. Вимоги користувачів можна розділити на функціональні та нефункціональні.

Функціональні вимоги

1. Перегляд інформації про тури:
    1.1. Користувачі повинні мати доступ до списку доступних турів з детальною інформацією про кожен тур.
    1.2. Можливість перегляду фотографій, описів, маршрутів і цін на тури.
2. Бронювання та подання заявок:
    2.1. Можливість бронювання турів онлайн через заповнення форми заявки.
    2.2. Зворотній зв'язок після подання заявки, включаючи підтвердження бронювання.
3. Відгуки :
    3.1. Користувачі повинні мати можливість залишати відгуки та оцінки турів.
    3.2. Відгуки повинні відображатися на сторінках турів для інших користувачів.
4. Інформація про гідів:
    4.1. Сторінки з інформацією про туристичних гідів, їх досвід.

4.2. Можливість перегляду доступних гідів та вибору гіда для конкретного туру.

5. Адміністративний інтерфейс:
   5.1. Адміністратори повинні мати доступ до управління заявками, відгуками та інформацією про тури.
   5.2. Захищений паролем доступ до адміністративної панелі.

Нефункціональні вимоги

1. Зручність використання:
   1.1. Інтуїтивно зрозумілий інтерфейс для користувачів різного рівня підготовки.
   1.2. Зручна навігація по сайту, швидкий доступ до основних функцій.
2. Продуктивність: -
   2.1. Швидкий час завантаження сторінок і обробки запитів.
   2.2. Сайт повинен витримувати високі навантаження під час пікових періодів.
3. Безпека:
   3.1. Захист даних користувачів, включаючи особисту інформацію та дані платіжних карток.
   3.2. Захист від зовнішніх загроз, таких як SQL-ін'єкції та міжсайтовий скриптинг (XSS).

Аналіз вимог до користувачів туристичного веб-сайту дозволяє визначити ключові функції та характеристики, які необхідні для забезпечення зручного, безпечного та ефективного користування сайтом. Врахування цих вимог під час розробки допоможе створити конкурентоспроможний продукт, який відповідатиме потребам туристів і сприятиме розвитку туристичних послуг.

2.3 Аналіз вимог до програмного забезпечення

Для успішної розробки туристичного веб-сайту необхідно чітко визначити вимоги до програмного забезпечення, яке забезпечить необхідну функціональність та ефективність системи. Вимоги до програмного забезпечення включають як функціональні, так і нефункціональні аспекти.

Функціональні вимоги

1. Користувацький інтерфейс (UI):
    1.1. Каталог турів: Сторінка для перегляду всіх доступних турів з детальною інформацією про кожен тур.
    1.2. Бронювання турів: Форми для подання заявок на бронювання турів.
    1.3. Відгуки та коментарі: Можливість користувачів залишати відгуки та переглядати відгуки інших.
    1.4. Інформація про гідів: Профілі туристичних гідів з описом їх досвіду та відгуками клієнтів.
    1.5. Адміністративна панель: Інтерфейс для адміністраторів для управління контентом, заявками та відгуками.
2. Серверна логіка
    2.1. Обробка запитів: Асинхронна обробка запитів для забезпечення: швидкої реакції сервера.
    2.2. Маршрутизація: Визначення маршрутів для різних запитів і сторінок.
3. База даних: Схема бази даних: Проектування таблиць для зберігання інформації про тури, користувачів, гідів, заявки та відгуки.

Нефункціональні вимоги

1. Продуктивність:

1.1. Швидкість завантаження: Забезпечення швидкого завантаження сторінок та обробки запитів.

1.2. Оптимізація запитів: Ефективне використання бази даних для мінімізації часу обробки запитів.

2. Безпека:

2.1. Захист даних: Захист персональних даних користувачів, включаючи шифрування паролів та захист даних платіжних карток.

2.2. Безпека передачі даних: Використання SSL для захисту даних під час їх передачі між клієнтом і сервером.

3. Сумісність: Адаптивний дизайн: Підтримка різних розмірів екранів для зручного використання на мобільних пристроях і планшетах.

4. Зручність використання:

4.1. Інтуїтивний інтерфейс: Простий та зрозумілий інтерфейс для користувачів різного рівня підготовки.

4.2. Навігація: Легка навігація по веб-сайту, з чітко визначеними розділами та функціями.

Аналіз вимог до програмного забезпечення туристичного веб-сайту дозволяє визначити критичні аспекти, необхідні для успішної реалізації проекту. Функціональні вимоги забезпечують основну функціональність веб-сайту, тоді як нефункціональні вимоги гарантують безпеку, продуктивність, масштабованість та зручність використання, що є ключовими факторами для створення ефективного та надійного туристичного веб-сайту.

2.4 Висновки до розділу 2

1. Були проаналізовані сучасні технології для розробки веб-сайту.
2. Аналіз та дослідження вимог до користувачів туристисного веб-сайту.
3. Розглянуті загальні вимоги до програмного забезпечення для веб-сайту.

3 РОЗРОБКА ТУРИСТИЧНОГО ВЕБ-САЙТУ

3.1 Опис предметної області

Предметна область цього дослідження охоплює розробку туристичного веб-сайту, який забезпечує користувачам можливість планувати, бронювати та оцінювати туристичні послуги. Ця область включає кілька ключових аспектів, таких як управління інформацією про тури, гідів, відгуки користувачів, а також обробку заявок і забезпечення взаємодії між користувачами та адміністраторами.

Метою розробки цього туристичного веб-сайту є створення функціональної платформи, яка дозволить користувачам легко знаходити та бронювати тури, залишати відгуки, а також отримувати всю необхідну інформацію для організації подорожей. Веб-сайт має забезпечити зручний інтерфейс для користувачів, ефективну обробку даних та надійний захист інформації.

Ця робота зосереджена на створенні інтуїтивно зрозумілого та зручного інтерфейсу для користувачів, реалізації функціоналу, який дозволяє користувачам переглядати, фільтрувати та шукати тури, а також бронювати їх онлайн, забезпеченні можливості залишати відгуки та оцінки, а також переглядати відгуки інших користувачів та розробці адміністративного інтерфейсу для управління контентом, заявками та відгуками. У цій роботі буде розглянуто: - Архітектура та дизайн веб-сайту. - Вибір та використання технологій для розробки, зокрема Node.js, Express.js та Sequelize. - Процес інтеграції з базою даних. - Забезпечення безпеки даних. - Тестування функціональності та продуктивності веб-сайту. - Зручність використання та адаптивність інтерфейсу. Ця робота спрямована на забезпечення користувачів зручним інструментом для планування та організації подорожей, підвищення ефективності взаємодії між користувачами та туристичними операторами,

демонстрацію можливостей сучасних веб-технологій у розробці функціональних та надійних веб-сайтів, створення бази для подальшого розвитку та розширення функціоналу веб-сайту, враховуючи потреби та зворотний зв'язок користувачів.

Цей проект має на меті не лише створення функціонального веб-сайту, але й показати важливість інтеграції сучасних технологій у сферу туризму для покращення користувацького досвіду та підвищення конкуренто спроможності на ринку туристичних послуг.

3.2 Архітектура веб-сайту

Архітектура веб-сайту про подорожі включає кілька основних компонентів, які забезпечують його функціональність, продуктивність та надійність. Ця архітектура базується на сучасних веб-технологіях та включає серверну, клієнтську частини, базу даних. Основні компоненти архітектури

1. Клієнтська частина (Front-end):
    1.1. HTML/CSS: Використовуються для створення структури сторінок (HTML) та стилізації інтерфейсу (CSS), забезпечуючи зручність та естетичний вигляд.
    1.2. JavaScript: Забезпечує динамічність та інтерактивність веб-сайту, зокрема обробку подій на стороні клієнта.
    1.3. React.js або Vue.js (опціонально): Можуть використовуватися для створення SPA (Single Page Application) з високою реактивністю та оптимізованою продуктивністю.
2. Серверна частина (Back-end): - Node.js: Використовується для запуску серверної частини, забезпечуючи асинхронну обробку запитів і високу продуктивність. - Express.js: Фреймворк для Node.js, який спрощує створення серверної логіки та маршрутизації запитів.

3. База даних (Data Storage): - PostgreSQL: Реляційна база даних, яка використовується для зберігання інформації про тури, користувачів, гідів, заявки та відгуки. - Sequelize: ORM (Object-Relational Mapping) бібліотека для Node.js, яка полегшує роботу з базою даних, забезпечуючи об'єктно-орієнтований доступ до даних.
4. Зовнішні сервіси та API: - Google Maps API: Для відображення маршрутів та цікавих місць на карті. - Email-сервіси: Для відправки підтверджень бронювання та інших повідомлень користувачам.

Детальна архітектура (Дивись Рис.1)

1. Клієнтський рівень (Client Layer):
    1.1. Інтерфейс користувача: Користувач взаємодіє з веб-сайтом через браузер, де відображається інтерфейс, створений за допомогою HTML, CSS і JavaScript.
    1.2. Запити до сервера: Всі дії користувача (наприклад, пошук турів, бронювання) призводять до відправки запитів до серверної частини.
2. Серверний рівень (Server Layer):
    2.1. Express.js маршрутизація: Обробка HTTP-запитів від клієнта, маршрутизація запитів до відповідних контролерів.
    2.2. Контролери: Логіка обробки запитів, виконання необхідних дій (наприклад, отримання даних з бази, створення нових записів).
    2.3. Middlewares: Проміжні обробники запитів для виконання аутентифікації, авторизації, обробки помилок тощо.
3. Рівень зберігання даних (Data Layer):
    3.1. PostgreSQL: Зберігає всі структуровані дані веб-сайту.
    3.2. Sequelize моделі: Визначення структур даних у вигляді моделей, які відображаються на відповідні таблиці в базі даних.

4. Зовнішні інтеграції (External Integrations): API інтеграції: Використання зовнішніх API для додаткових функціональностей (наприклад, відображення карт, відправка email-повідомлень).

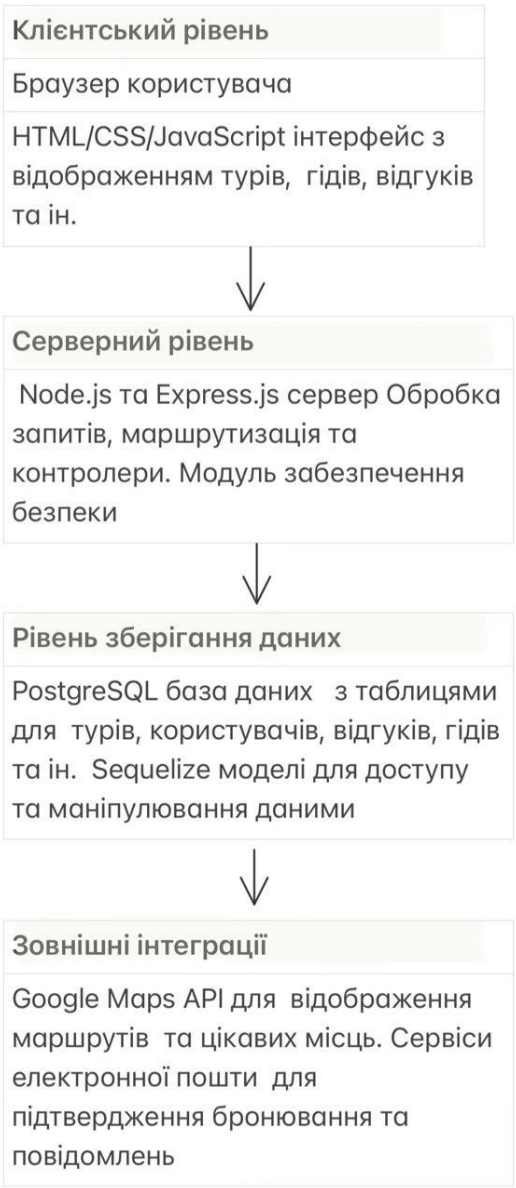


Рисунок 1 – Архітектура веб-сайту

Архітектура веб-сайту про подорожі базується на поєднанні сучасних технологій для забезпечення інтерактивності, масштабованості та надійності. Використання Node.js та Express.js на серверному рівні дозволяє ефективно обробляти запити, а PostgreSQL з ORM Sequelize забезпечує надійне та зручне

зберігання даних. Клієнтська частина, створена за допомогою HTML, CSS і JavaScript, надає користувачам зручний інтерфейс для взаємодії з сайтом. Інтеграція з зовнішніми сервісами додатково розширює функціональність веб-сайту, роблячи його корисним та зручним для користувачів.

На Рис.2 Зображено Use-Case для Користувача та на Рис.3 зображено Use-case для Адміністратора.

Рисунок 2 – Use-Case для Користувача

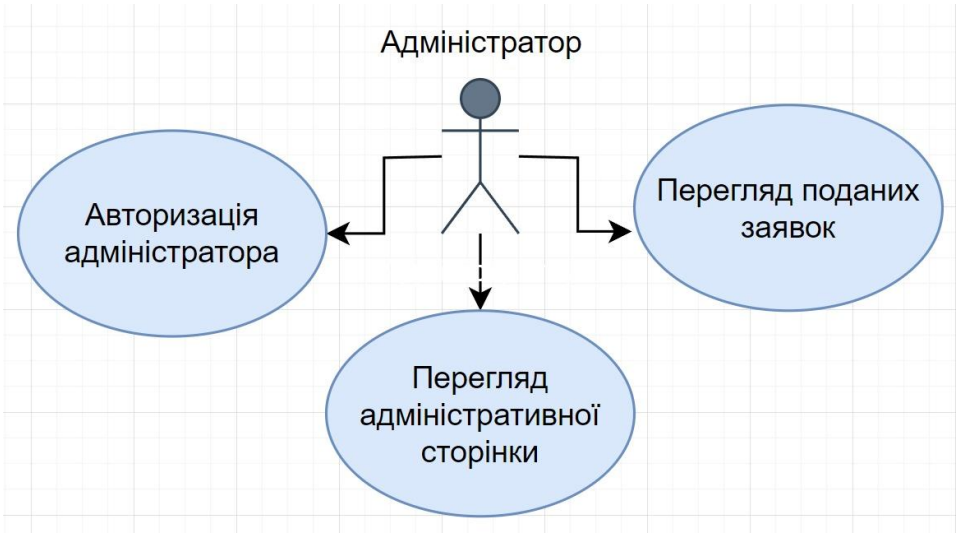


Рисунок 3 – Use-case для Адміністратора

3.3 Функціональні вимоги системи

Функціональні вимоги системи визначають основні можливості та функції, які має виконувати веб-сайт про подорожі для задоволення потреб користувачів.

Деякі загальні функціональні вимоги включають:

1. Перегляд турів та пакетів подорожей:
    1.1. Користувачі можуть переглядати різноманітні тури, екскурсії та подорожі по регіонам України.
    1.2. Інформація про кожний тур повинна включати опис, фотографії, маршрут, ціни та умови бронювання.
2. Бронювання турів:
    2.1. Користувачі можуть бронювати тури та пакети подорожей онлайн.
    2.2. Можливість обрати дату, кількість учасників та додаткові послуги.
3. Подання відгуків та оцінок: - Користувачі можуть залишати відгуки та оцінки після завершення подорожі. - Можливість оцінювати якість туру, послуг гідів та загальний досвід.
    3.1. Керування адміністратором: Адміністратор може додавати, редагувати та видаляти тури, гідів, цікаві місця та іншу інформацію на сайті.
    3.2. Можливість переглядати та керувати заявками на бронювання та відгуками користувачів. Ці функціональні вимоги є базовими та необхідними для створення функціонального та корисного туристичного веб-сайту.

3.4 Вимоги до інтерфейсу

Вимоги до інтерфейсу включають:

- Зручність використання: Інтуїтивний та легкий для користувача інтерфейс, який дозволяє швидко знаходити необхідну інформацію та виконувати дії.
- Естетика та привабливість: Привабливий дизайн, який привертає увагу користувачів та робить взаємодію з сайтом приємною.
- Адаптивність: Забезпечення коректного відображення сайту на різних пристроях і розмірах екранів, включаючи комп'ютери, планшети та смартфони.
- Навігація: Чітка та легко доступна навігація, яка дозволяє користувачам швидко переходити між різними розділами та функціями сайту.
- Взаємодія з користувачем: Можливість взаємодії з сайтом через різні елементи інтерфейсу, такі як кнопки, форми та меню.
- Доступність: Забезпечення доступності сайту для всіх користувачів, включаючи людей з обмеженими можливостями.
- Швидкість та продуктивність: Оптимізація швидкості завантаження сторінок та відгуків на сайті, щоб забезпечити зручність користувачів.
- Відображення інформації: Чітке та структуроване відображення інформації про тури, гідів, цікаві місця та інші елементи сайту.

Ці вимоги допомагають забезпечити зручний та привабливий для користувачів інтерфейс, який підвищує задоволення від використання веб-сайту і сприяє досягненню його цілей.

3.6 Реалізація

У рамках підготовки до проекту я спочатку налаштувала середовище розробки та створила базову структуру проекту. Для підтримки якості коду та його читабельності, я встановила розширення ESLint для перевірки синтаксису JavaScript, а також Prettier - Mode formatter для автоматичного форматування коду. Для забезпечення зручності при роботі з Node.js, я використала Node.js Extension Pack, що включає в себе різноманітні додаткові інструменти. Після налаштування середовища розробки, я ініціалізувала проект та встановила необхідні пакети, готуючи його до подальшого розвитку та реалізації функціональності.

Після налаштування середовища розробки, створила основний серверний файл та визначила необхідні маршрути для різних сторінок мого веб-сайту. Додала обробку POST-запитів, щоб забезпечити можливість обробки даних, що надходять від користувачів, і виконати відповідні дії. Також була додана обробка статичних файлів, щоб забезпечити коректне відображення статичного контенту на сторінках веб-сайту.

На Рис.4 зображено структуру розробленого туристичного веб-сайту.

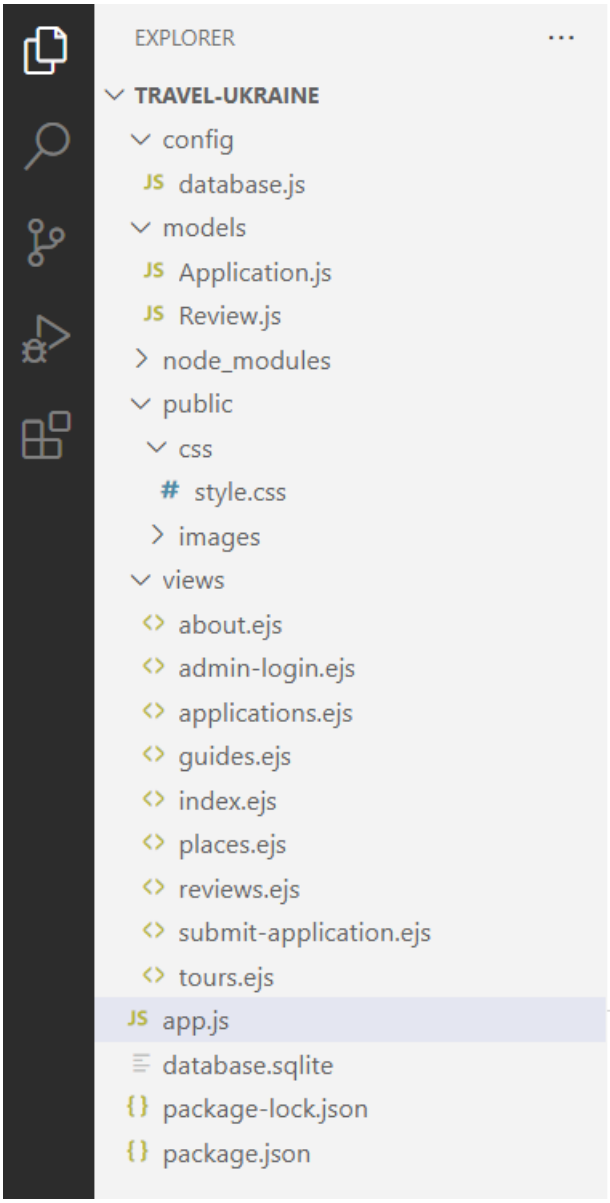


Рисунок 4 – Структура веб-сайту.

У файл app.js було додано кілька ключових функціональних елементів для забезпечення роботи веб-сайту: Підключено модулі Express для створення серверу, модуль body-parser для обробки POST-запитів, Sequelize для роботи з базою даних.

Для налаштування середовища встановлено механізм обробки статичних файлів для відображення CSS та інших ресурсів. Налаштовано систему перегляду EJS для рендерингу динамічних сторінок.

Визначення маршрутів - додано маршрути для різних сторінок веб-сайту, включаючи головну сторінку, сторінки про нас, тури, гіди та цікаві місця. Також додано маршрути для обробки форм заявок та відгуків користувачів, забезпечивши збереження цих даних у базі даних.

Обробка POST-запитів: додано обробку POST-запитів для збереження даних форм у базі даних та маршрути для обробки POST-запитів, що дозволяє користувачам подавати заявки та залишати відгуки. Впроваджено систему аутентифікації для доступу до адміністративної панелі.

Ці зміни дозволили забезпечити коректну роботу веб-сайту та реалізувати необхідний функціонал для користувачів та адміністраторів. На лістингу 1 зображено код файлу «app.js»

Лістинг 1: Код "app.js"

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');
const sequelize = require('./config/database');
const Application = require('./models/Application');

const app = express();

// Налаштування EJS як шаблонізатора
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

// Підключення статичних файлів
app.use(express.static(path.join(__dirname, 'public')));

// Парсинг даних з форм
app.use(bodyParser.urlencoded({ extended: true }));
```

```
// Синхронізація бази даних
sequelize.sync().then(() => {
    console.log('База даних синхронізована');
}).catch(err => {
    console.error('Помилка синхронізації бази даних:', err);
});

const Review = require('./models/Review');

// Маршрут для головної сторінки
app.get('/', (req, res) => {
    res.render('index', { title: 'Welcome to Travel Ukraine'
});
});

// Маршрут для сторінки 'Про нас'
app.get('/about', (req, res) => {
    res.render('about', { title: 'Про нас' });
});

// Маршрут для сторінки 'Відгуки'
app.get('/reviews', async (req, res) => {
    const reviews = await Review.findAll({
        order: [['createdAt', 'DESC']] // сортування за датою
створення у зворотному порядку
    });
    res.render('reviews', { reviews, title: 'Відгуки' });
});


// Маршрут для обробки подачі нових відгуків
```

```
app.post('/reviews', async (req, res) => {
    const { name, review } = req.body;
    await Review.create({ name, review });
    res.redirect('/reviews');
});

// Маршрут для сторінки 'Гіди'
app.get('/guides', (req, res) => {
    res.render('guides', { title: 'Гіди' });
});

// Маршрут для сторінки 'Цікаві місця'
app.get('/places', (req, res) => {
    res.render('places', { title: 'Цікаві місця' });
});

// Маршрут для сторінки 'Тури'
app.get('/tours', (req, res) => {
    res.render('tours', { title: 'Тури' });
});

// Маршрут для форми подачі заявки
app.get('/submit-application', (req, res) => {
    res.render('submit-application', { title: 'Подати
заявку' });
});

// Обробка заявки
app.post('/submit-application', async (req, res) => {
    try {
        const { firstName, lastName, phone, email, date,
tour, guide } = req.body;
```

```
        await Application.create({ firstName, lastName, phone, email, date, tour, guide });
        res.send('<h1 style="text-align: center;">Заявка отримана</h1>');
    } catch (error) {
        console.error('Помилка створення заявки:', error);
        res.status(500).send('Помилка сервера');
    }
});

// Маршрут для показу форми введення пароля
app.get('/admin', (req, res) => {
    res.render('admin-login', { title: 'Вхід для адміністратора' });
});

// Обробка форми введення пароля
app.post('/admin/login', (req, res) => {
    const password = req.body.password;
    if (password === 'adminpassword') {
        res.redirect('/admin/applications');
    } else {
        res.redirect('/admin');
    }
});

// Маршрут для перегляду заявок (тільки для адміністратора)
app.get('/admin/applications', async (req, res) => {
    try {
        const applications = await Application.findAll();
        res.render('applications', { applications, title: 'Заявки' });
```

```
    } catch (error) {
        console.error('Помилка отримання заявок:', error);
        res.status(500).send('Помилка сервера');
    }
});

// Налаштування порту та запуск сервера
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Сервер працює на http://localhost:${PORT}`);
});
```

На головну сторінку було додано Заголовок, Наші пропозиції, Розділ "Чому обирають нас", Розділ "Відгуки", Кнопку "Заповнити анкету". У лістингу 2 коротко записан код Головної сторінки. Див. Рис. 5

Лістинг 2: вміст сторінки "index.ejs"

```
<!DOCTYPE html>
<html lang="en">
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="/css/style.css">
<title><%= title %></title>
    <header>
        <nav class="navbar">
            <ul>
                <li><a href="/">Головна</a></li>
                <li><a href="/about">Про нас</a></li>
                <li><a href="/reviews">Відгуки</a></li>
                <li><a href="/guides">Гіди</a></li>
```

```
                <li><a href="/places">Цікаві місця</a></li>
                <li><a href="/tours">Тури</a></li>
                <li><a     href="/submit-application">Подати
заявку</a></li>
            </ul>
        </nav>
    </header>
    <div class="img-uk">
        <!-- ... Привітальний тури--!>
     <div  class="proposals"><!--...    Запропоновані  тури--
!></div>
    </div>

    <section class="advantages-reviews">
        <div class="advantages">
            <h2>Чому обирають нас</h2>
            <ul><!--... Перелік переваг--!></ul>
        </div>
        <div class="testimonials">
            <!-- ... Відгуки клієнтів--!></div>
    </section>

    <section class="cta">
        <h2>Готові до подорожі?</h2>
        <a  href="/submit-application"  class="btn">Заповнити
анкету</a>
    </section>

    <footer>
        <p>&copy;    2024    Travel    Ukraine.    Всі    права
захищені.</p>
    </footer>
```

```
</body>
</html>
```

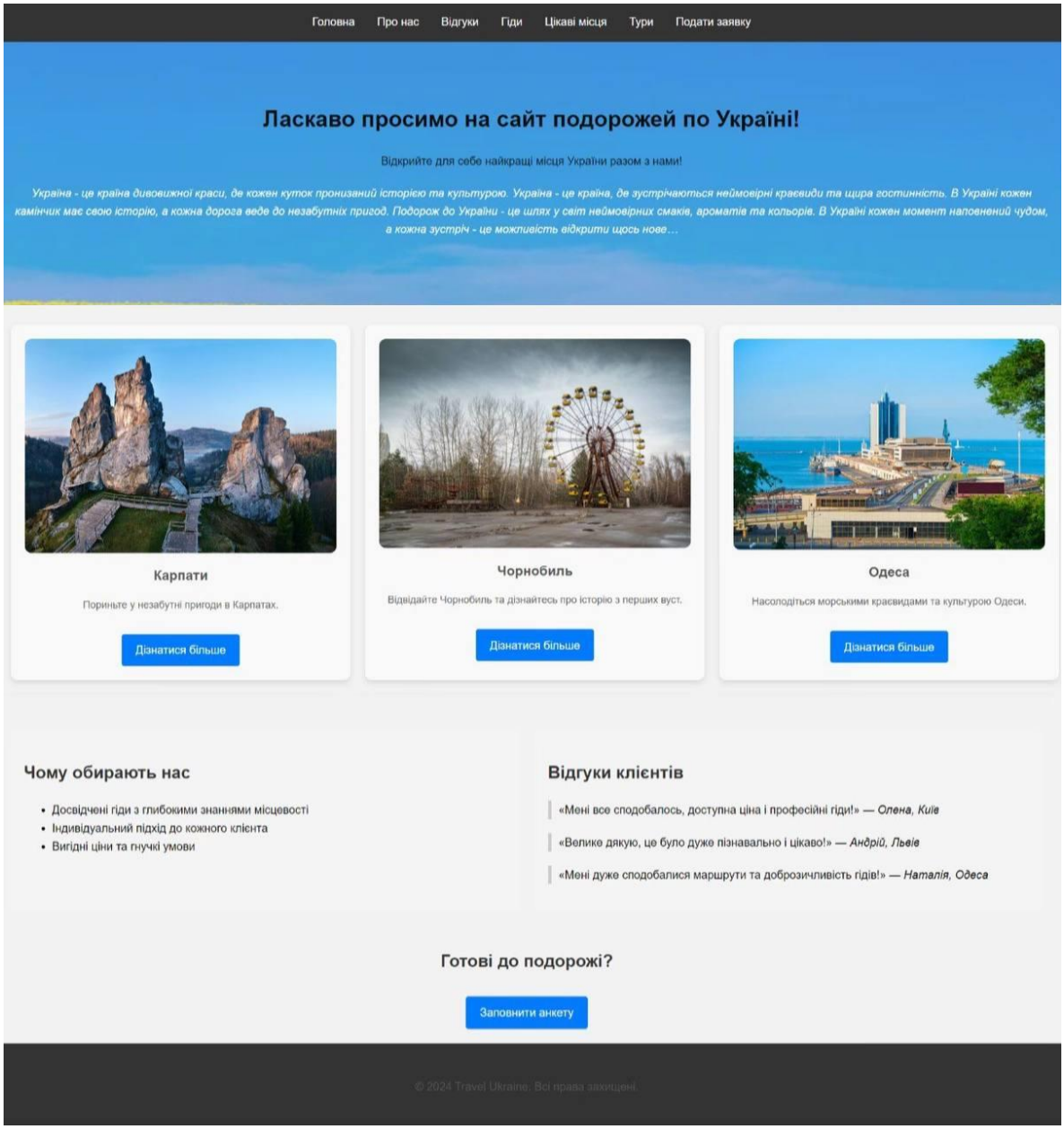


Рисунок 5 – Головна сторінка

На сторінку «Про нас» було додано наступні елементи: Кнопка «Заповнити анкету»: додано кнопку, яка перенаправляє користувачів на сторінку заповнення анкети. Це полегшує процес подачі заявок та сприяє активнішій взаємодії користувачів з нашим сервісом.

Додано нові блоки тексту, що детально описують місію компанії, її історію та ключові досягнення. Цей контент допомагає користувачам краще зрозуміти, хто ми є і чим займаємося.

Додано секцію з інформацією про нашу команду. Інтерактивні елементи - впроваджено інтерактивні елементи, такі як спливаючі вікна та анімації, для покращення користувацького досвіду та залучення уваги до важливих частин сторінки.

Ці зміни зробили сторінку «Про нас» більш інформативною, привабливою та зручною для користувачів, сприяючи кращому розумінню цінностей та діяльності нашої компанії. У лістингу 3 надаю інформацію щодо змісту сторінки “Про нас”. Див. Рис.6

Лістинг 3: вміст “about.ejs”

```
<!DOCTYPE html>
<!--... HTML-шаблон--!>
</head>
<body>
    <header>
        <nav class="navbar">
            <ul>
                    <li><a href="/">Головна</a></li>
                    <!-- .... Посилання на інші сторінки--!>
            </ul>
        </nav>
    </header>
    <main>
        <div class="content-wrapper">
            <div class="left-content">
                <h1><%= title %></h1>
```

```
                <p>Наша компанія займається організацією подорожей по Україні. Ми прагнемо показати вам найкрасивіші та найцікавіші місця нашої країни!</p>
                <section>
                <h2>Наша місія</h2>
                <p>Наша місія – зробити ваші подорожі незабутніми та пізнавальними!</p>
                </section>
                <section>
                <h2>Наша команда</h2>
                <p>У нас працює команда професіоналів, які люблять свою справу і готові допомогти вам з організацією вашого відпочинку!</p>
                </section>
                <section>
                    <h2>Контактна інформація</h2>
                    <p>Телефон: +380664461417</p>
                    <p>E-mail: travel-ua@gmail.com</p>
                    <address>Адреса: вул. Хрещатик, 50, Київ,Україна</address>
                  <!-- Кнопка для переходу на сторінку "Заповнити анкету" -->
                    <div class="apply-button">
                        <a href="/submit-application" class="btn">Заповнити анкету</a>
                    </div>
                </section>
            </div>
            <div class="right-content">
                <img src="/images/photo.jpg">
                <q class="nice-quote">
```

```
                    Людину роблять щасливою три речі: любов,
цікава робота і можливість подорожувати.
                </q>
                <i                  class="nice-quote"><sub>I.
Бунін</sub></i>
            </div>
        </div>
        <iframe
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d
2540.0171050983755!2d30.518093915694318!3d50.44738279567482
!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x40d4ce
e0e0289a45%3A0xe6b3a28b3e5e9e19!2z0J_Rg9GI0LrQtdC90YIsINCU0
LXQsNC90LjRjywgNTAsINCa0YDQvtC_0YDQvtCy0YHQutCw0Y8g0L7QsdC7
LiwgMDIwMDA!5e0!3m2!1suk!2sua!4v1620231424657!5m2!1suk!2sua
"      width="600"      height="450"      style="border:0;"
allowfullscreen="" loading="lazy">
        </iframe>
        <section>
            <p><!--... Вітальний текст--!></p>
        </section>
    </main>
    <footer>
    <p>&copy; 2024 Travel Ukraine. Всі права захищені.</p>
    </footer>
</body>
</html>
```

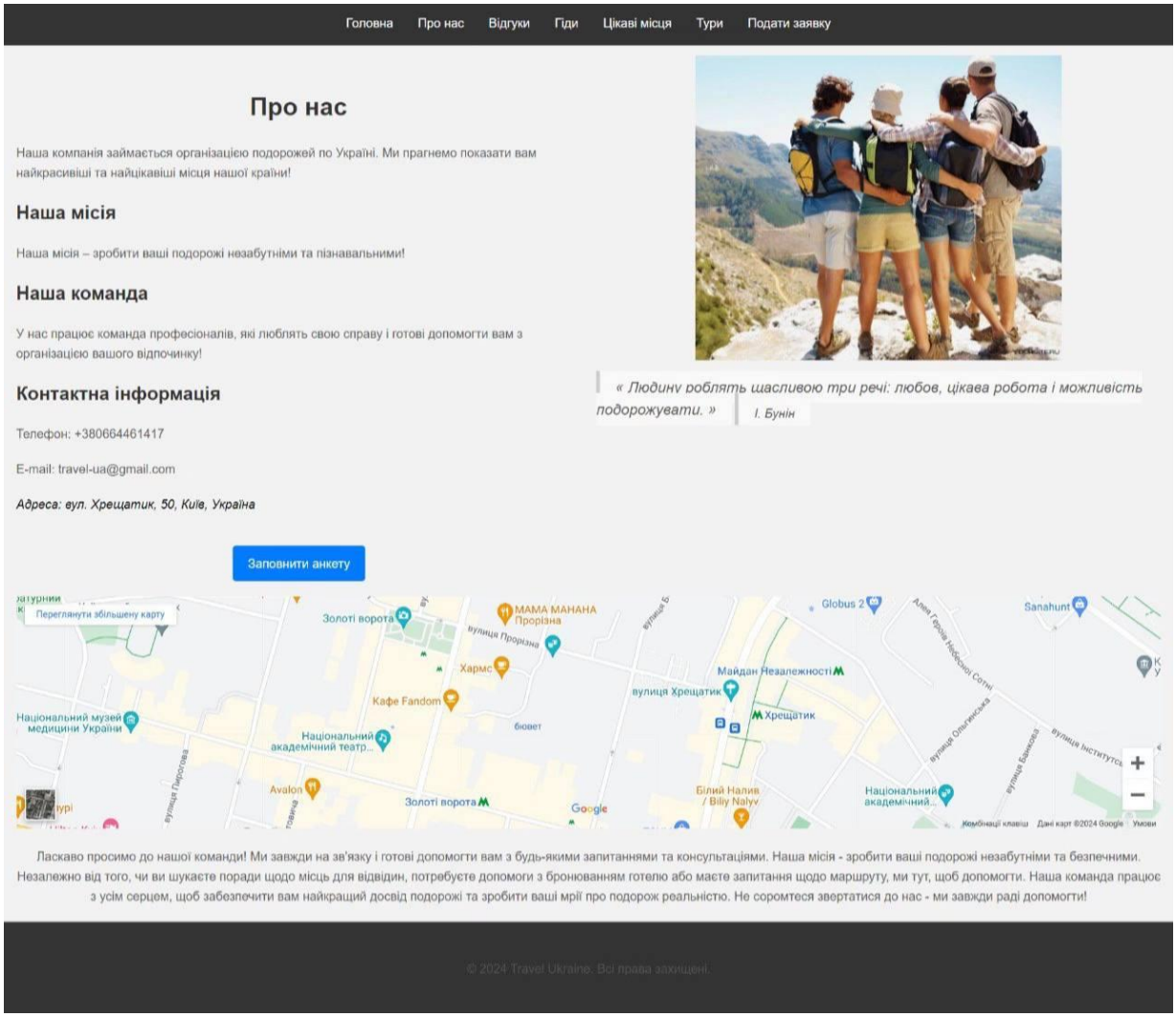


Рисунок 6 – Сторінка “Про нас”

Було створено сторінку «Відгуки», де було додано форму, що дозволяє користувачам залишати свої відгуки про наші послуги. Форма включає поля для введення імені, електронної пошти та самого тексту відгуку.

Реалізування відображення всіх відгуків користувачів, збережених у базі даних. Відображення нових відгуків на початку списку, що дозволяє користувачам бачити найактуальнішу інформацію. Додавання повідомлення, яке з’являється після успішного подання відгук, інформуючи користувача про те, що його відгук було прийнято та збережено.

Реалізовано базовий захист від спаму та забезпечення конфіденційності даних користувачів.

Ці зміни дозволили створити зручну та інтуїтивно зрозумілу платформу для залишення та перегляду відгуків, підвищивши рівень взаємодії користувачів з нашим сайтом та надавши їм можливість ділитися своїм досвідом з іншими. У Лістингу 4 зображено необхідний код. Див. Рис.7

Лістинг 4: Вміст файлу "reviews.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!--... HTML-шаблон--!>
</head>
<body>
    <header>
        <nav class="navbar">
            <ul>
                <li><a href="/">Головна</a></li>
                <!-- .... Посилання на інші сторінки--!></ul>
        </nav>
    </header>
    <div class="container">
        <h1>Відгуки</h1>
        <h3>Ми завжди раді отримати ваші відгуки про наші тури. <br> Будь ласка, залишайте чесні відгуки, щоб ми могли вдосконалювати наші послуги. <br> Дякуємо за вибір нашого сайту!</h3>

        <!-- Форма для подачі нового відгуку -->
        <form action="/reviews" method="post">
            <label for="name">Ваше ім'я:</label>
            <input type="text" id="name" name="name" required>
```

```
            <label for="review">Ваш відгук:</label>
            <textarea id="review" name="review"
class="review-textarea" required></textarea>

            <button type="submit">Надіслати</button>
        </form>

        <!-- Відображення наявних відгуків -->
        <div class="reviews-list">
            <% reviews.forEach(review => { %>
                <div class="review-item">
                    <h3><%= review.name %></h3>
                    <p><%= review.review %></p>
                </div>
            <% }); %>
        </div>
    </div>
</body>
</html>
”
```

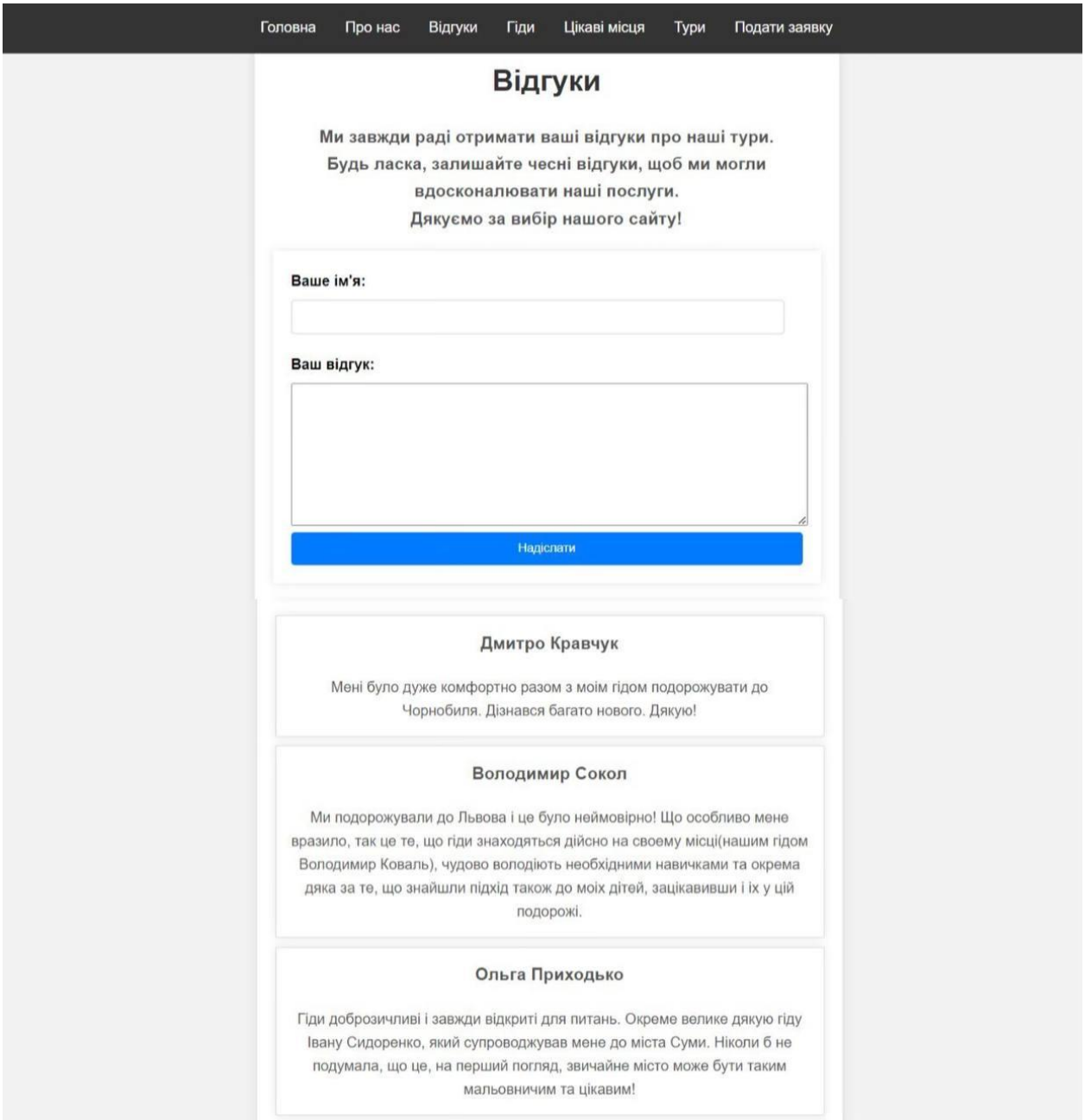


Рисунок 7 – Сторінка “Відгуки”

Для налаштування бази даних та роботи з моделями даних були створені config/database.js та models/Review.js

Файл config/database.js було створено для налаштування підключення до бази даних за допомогою Sequelize. Основною ціллю налаштування підключення до бази даних було встановлення конфігурації для підключення до бази даних SQLite, включаючи ім'я бази даних та інші параметри.

Ініціалізація Sequelize з використанням встановлених параметрів підключення дозволяє використовувати ORM для взаємодії з базою даних.

Файл models/Review.js було створено для визначення моделі «Review» за допомогою Sequelize. Основними цілями було визначення структури таблиці «Review» та використання моделі для взаємодії з таблицею «Review» (створення, читання, оновлення та видалення записів у таблиці «Review» за допомогою Sequelize).

config/database.js налаштовує підключення до бази даних і створює екземпляр Sequelize для взаємодії з базою даних.
У Лістингу 5 зображено викорастаний код для отримання необхідних результатів.

models/Review.js визначає модель «Review», яка представляє таблицю «Review» у базі даних і дозволяє здійснювати операції CRUD на цій таблиці. У Лістингу 6 зображено необхідний для цього код.

Ці файли є ключовими для роботи з базою даних у проекті, забезпечуючи організовану та структуровану взаємодію з даними.

Лістинг 5: Вміст config/database.js

```
const { Sequelize } = require('sequelize');

const sequelize = new Sequelize({
    dialect: 'sqlite',
    storage: './database.sqlite'
});

module.exports = sequelize;
```

```
const { Sequelize, DataTypes } = require('sequelize');
const sequelize = require('../config/database');

const Review = sequelize.define('Review', {
```

```
    name: {
        type: DataTypes.STRING,
        allowNull: false
    },
    review: {
        type: DataTypes.TEXT,
        allowNull: false
    }
}, {
    timestamps: true, // це додасть поля createdAt та updatedAt
});
module.exports = Review;
```

Наступним кроком було додавання файлу «guides.ejs». Цей файл використовується для відображення інформації про гідів, які надають послуги в рамках туру. До нього було додано код, який дозволяє відобразити список гідів та їхні описи.

Було додано заголовок сторінки, секцію для відображення кожного гіда з його ім'ям, фото та коротким описом. Використано CSS класи для стилізації елементів сторінки, таких як фото та опис гідів.

Використано цикл для відображення кожного гіда з масиву guides. Відображено фото, ім'я та опис гіда.

Файл guides.ejs був оновлений для відображення списку гідів, включаючи їх фото, імена та описи, з використанням шаблонних виразів EJS для динамічного виведення даних. У Лістингу 7 зображено викорастаний для цього код. Див. Рис. 8

Лістинг 7: Вміст файлу «guides.ejs»

```
<!DOCTYPE html>
<html lang="uk">
<head>
<!--... HTML-шаблон--!>
</head>
<body>
    <header>
        <nav class="navbar">
            <ul>
                <li><a href="/">Головна</a></li>
                <!-- .... Посилання на інші сторінки--!>
            </ul>
        </nav>
    </header>
    <main>
        <h1><%= title %></h1>
        <p  class="welcome-message">Ми  завжди  готові  бути
поруч та показати Україну з нових сторін!</p>
        <div class="guides">
            <div class="guide-card">
            <img src="/images/guide1.jpg" alt="Гід 1">
            <h2>Олександр Петренко</h2>
            <p>Олександр – досвідчений гід з більш ніж 10-
річним  стажем.  Його  екскурсії  завжди  цікаві  та
інформативні.</p>
            </div>
            <!--... Картки інших гідів--!>
        </div>
    </main>
    <script src="/js/script.js"></script>
```

```
    <footer>
    <p>&copy; 2024 Travel Ukraine. Всі права захищені.</p>
    </footer>
</body>
</html>
```

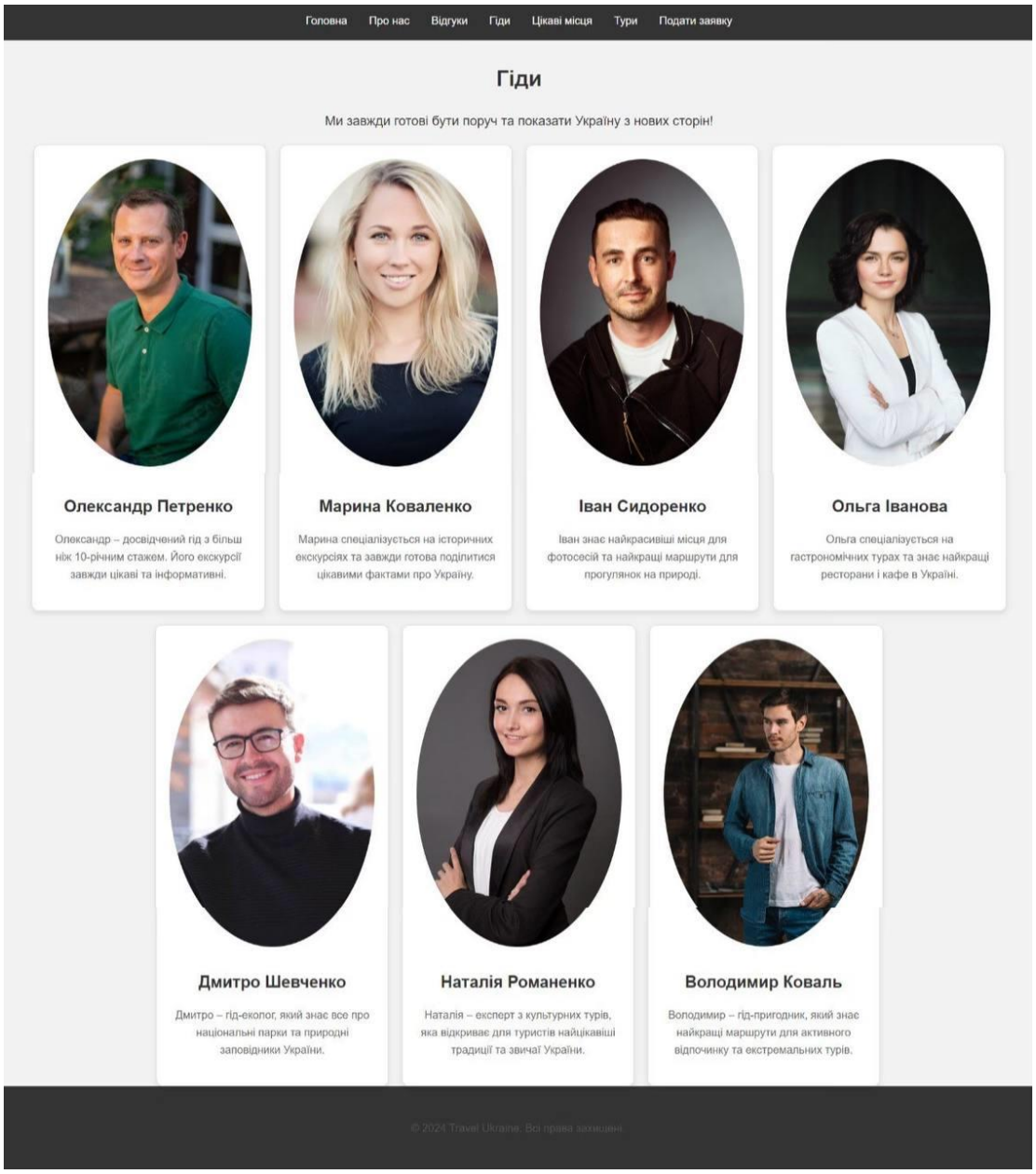


Рисунок 8 – Сторінка “Гіди”

Наступним кроком я додала сторінку з цікавими місцями «places.ejs»

Цей файл використовується для відображення інформації про цікаві місця, які можна відвідати в рамках турів. До нього було додано код, який дозволяє відобразити список таких місць з їхніми фото, назвами та описами.

Додано заголовок сторінки, секцію для відображення кожного місця з його назвою, фото та коротким описом. Використано CSS класи для стилізації елементів сторінки, таких як фото та опис місць.

Використано цикл для відображення кожного місця з масиву places. Відображено фото, назву та опис місця.У Лістингу 8 додаю код, який я використовувала для написання цієї сторінки. Див. Рис. 9

Лістинг 8: Вміст файлу «places.ejs»

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <!--... HTML-шаблон--!>
</head>
<body>
    <header>
        <nav class="navbar">
            <ul>
                <li><a href="/">Головна</a></li>
                <!-- .... Посилання на інші сторінки--!>
            </ul>
        </nav>
    </header>
    <main>
        <h1><%= title %></h1>
        <div class="places-container">
            <section class="place">
                <img src="/images/kyiv.jpg">
```

```
                <h2>Київ – столиця України</h2>
                <p>Київ – це серце України, де збереглися унікальні історичні пам'ятки, такі як Софійський собор та Києво-Печерська лавра. Місто також славиться своєю зеленою зоною, прекрасними парками та набережними.</p>
            </section>
            <!--...Секції з іншими цікавими місцями--!>
        </div>
    </main>
    <footer>
    <p>&copy; 2024 Travel Ukraine. Всі права захищені.</p>
    </footer>
</body>
</html>
```

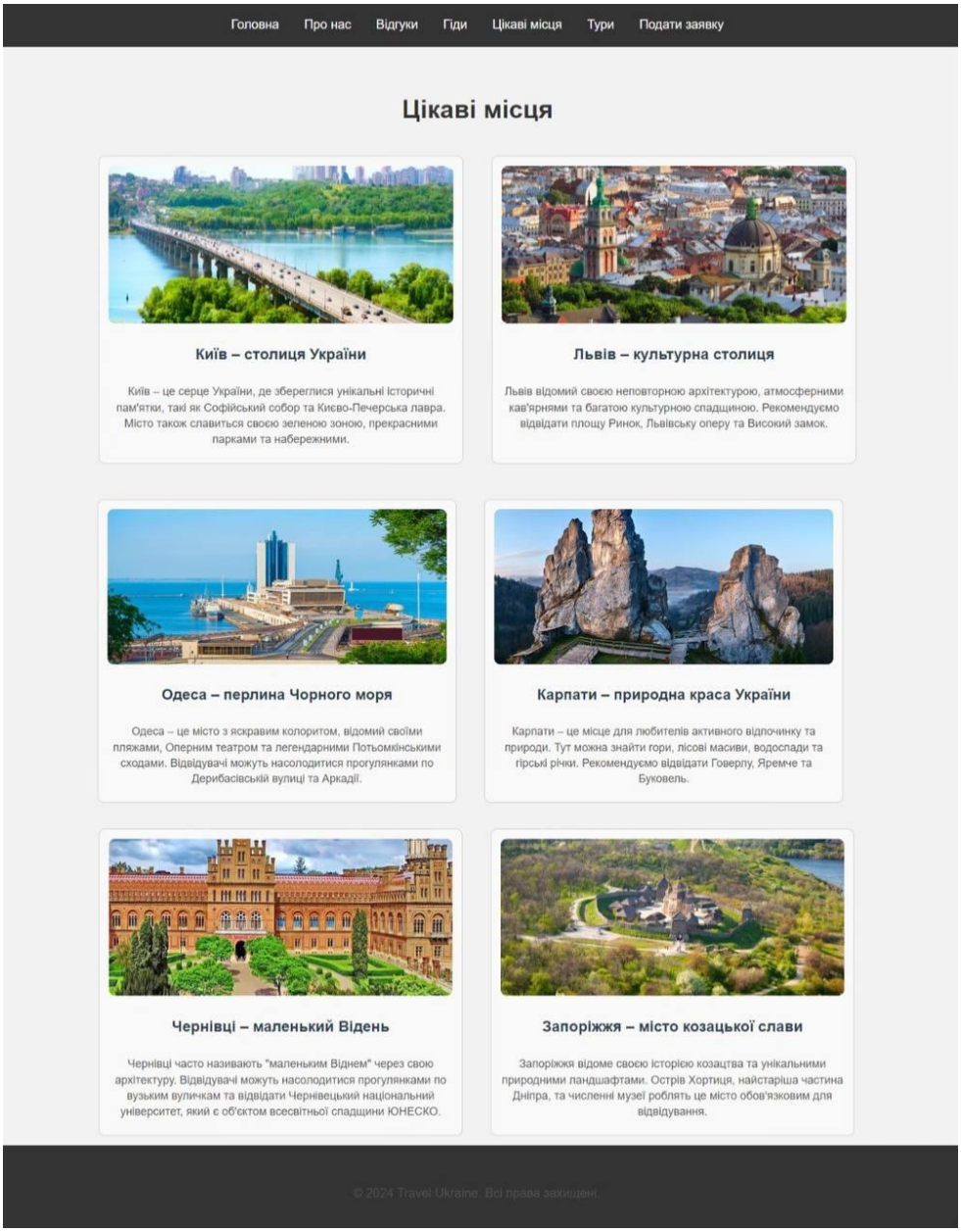


Рисунок 9 – Сторінка “Цікаві місця”

Потім створення файлу «tours.ejs», куди було додано HTML-код для представлення списку турів, кожен тур містить назву, опис, ціну та кнопку для бронювання. Використано класи для стилізації елементів, таких як tour для кожного окремого туру і container для загальної структури сторінки.

Додано блоки з детальною інформацією про кожен тур, включаючи назву, опис, ціну та кнопку бронювання. У Лістингу 9 зображено код для сторінки «Тури». Див. Рис. 10

Лістинг 9: Вміст файлу «tours.ejs»

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <!--... HTML-шаблон--!>
</head>
<body>
    <header>
        <nav class="navbar">
            <ul>
                <li><a href="/">Головна</a></li>
                <!-- .... Посилання на інші сторінки--!>
            </ul>
        </nav>
    </header>
    <main>
        <h1>Наші Тури</h1>
        <div class="tours-container">
        <div class="tour">
                <h2>Тур 1: Київ </h2>
                <img src="/images/kyiv.jpg" alt="Київ">
                <p>Ціна: 1800 грн</p>
                <p>Цей тур включає в себе відвідування
основних визначних пам'яток Києва, таких як Софійський собор,
Києво-Печерська лавра, Майдан Незалежності та багато інших.
</p>
                <a href="/submit-application"
class="btn">Бронювати</a>
            </div>
        <!--... Таким самим чином додано інші тури--!>
        </div>
    </main>
```

```
    <script src="/js/script.js"></script>

    <footer>
        <p>&copy; 2024 Travel Ukraine. Всі права захищені.</p>
    </footer>
</body>
</html>
```

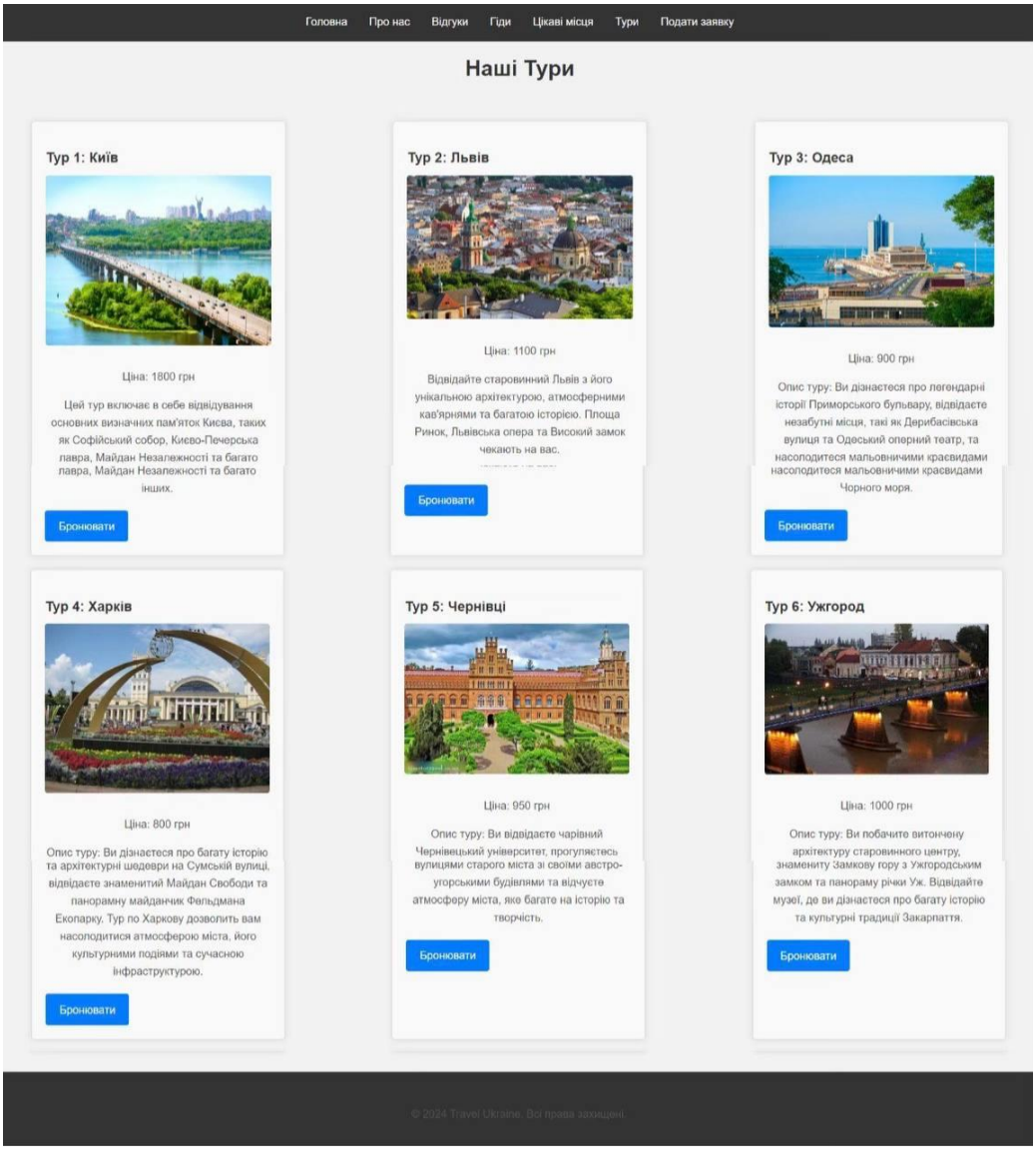


Рисунок 10 – Сторінка “Тури”

Створення та додача до файлу submit-application.ejs HTML-коду для створення форми, яка дозволяє користувачам подавати заявки на тури.

Додано поля для збору необхідної інформації від користувача, такі як ім'я, електронна пошта, телефон та повідомлення. Додано кнопку для відправлення заповненої форми.

Опис елементів форми

- Ім'я (name): поле для введення імені користувача.
- Електронна пошта (email): поле для введення електронної адреси користувача.
- Телефон (phone): поле для введення номеру телефону користувача. - Повідомлення (message): текстова область для введення додаткової інформації або повідомлення від користувача.
- Кнопка "Надіслати": кнопка для відправлення форми.

Ці зміни дозволяють користувачам легко заповнити та подати заявку на тур через даний веб-сайт. У Лістингу 10 зображення код для сторінки «Подати заявку». Див. Рис. 1

Лістинг 10: Вміст файлу «submit-applications.ejs»

```
<!DOCTYPE html>
<html lang="uk">
<head>
<!--... HTML-шаблон--!>
</head>
<body>
    <header>
        <nav class="navbar">
            <ul>
                <li><a href="/">Головна</a></li>
                <!-- .... Посилання на інші сторінки--!>
            </ul>
```

```
        </nav>
    </header>

    <div class="container">
        <h1>Подати заявку</h1>
        <form action="/submit-application" method="POST">
            <div class="form-group">
                <label for="firstName">Ім'я:</label>
                <input    type="text"    id="firstName"    name="firstName"
required>
            </div>
            <div class="form-group">
                <label for="lastName">Прізвище:</label>
                <input type="text" id="lastName" name="lastName" required>
            </div>
            <div class="form-group">
                <label for="phone">Номер телефону:</label>
                <input type="tel" id="phone" name="phone" required>
            </div>
            <div class="form-group">
                <label for="email">Електронна пошта:</label>
                <input type="email" id="email" name="email" required>
            </div>
            <div class="form-group">
                <label for="date">Виберіть дату:</label>
                <input type="date" id="date" name="date" required>
            </div>
            <div class="form-group">
                <label for="tour">Виберіть тур:</label>
                <select id="tour" name="tour" required>
                    <option value="kyiv">Київ</option>
                    <option value="lviv">Львів</option>
                    <option value="odessa">Одеса</option>
                    <option value="kharkiv">Харків</option>
                    <option value="chernivtsi">Чернівці</option>
                    <option value="uzhhorod">Ужгород</option>
                    <option value="poltava">Полтава</option>
                    <option                  value="ivano-frankivsk">Івано-
Франківськ</option>
                    <option value="karpaty">Карпати</option>
```

```
                    <option value="sumy">Суми</option>
                    <option value="zaporizhzhia">Запоріжжя</option>
                    <option value="chernobyl">Чорнобиль</option>
                </select>
            </div>
            <div class="form-group">
                <label for="guide">Виберіть гіда:</label>
                <select id="guide" name="guide" required>
                    <option value="guide1">Гід 1 - Олександр
Петренко</option>
                    <option value="guide2">Гід 2 - Марина
Коваленко</option>
                    <option value="guide3">Гід 3 - Іван Сидоренко</option>
                    <option value="guide4">Гід 4 - Ольга Іванова</option>
                    <option value="guide5">Гід 5 - Дмитро Шевченко</option>
                    <option value="guide6">Гід 6 - Наталія
Романенко</option>
                    <option value="guide7">Гід 7 - Володимир
Коваль</option>
                </select>
            </div>
            <button type="submit">Подати заявку</button>
        </form>
    </div>

    <footer>
        <p>&copy; 2024 Travel Ukraine. Всі права захищені.</p>
    </footer>
</body>
</html>
```

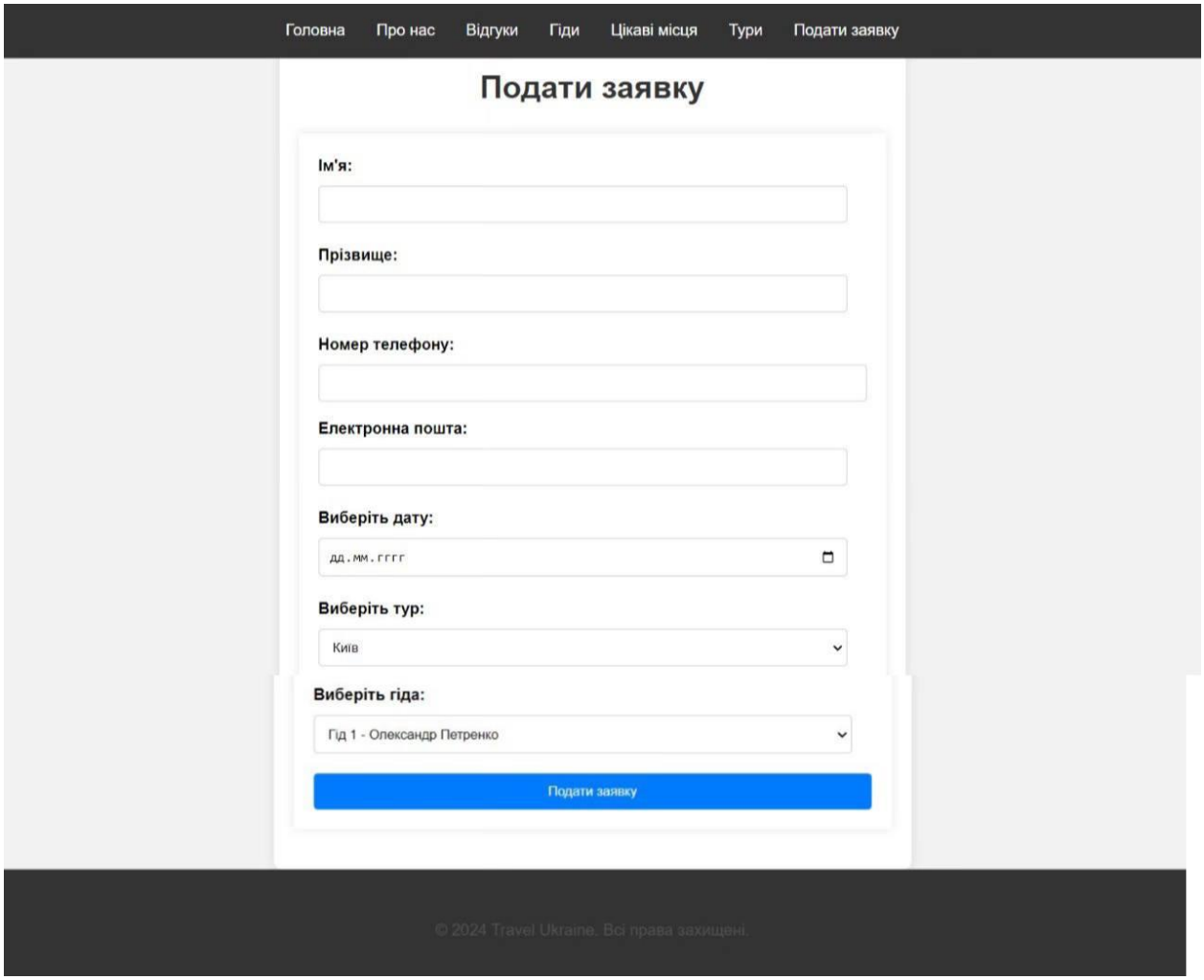


Рисунок 11 – Сторінка ”Заповніть заявку”

Файл `views/applications.ejs` був доданий для відображення списку поданих заявок. Ця сторінка дозволяє адміністратору переглядати всі заявки, які були надіслані користувачами через форму на сторінці "Подати заявку".

Відображення списку заявок, файл містить HTML-код, який створює таблицю або список для відображення заявок. Використовується серверний шаблон EJS для вставки даних з бази даних в HTML.

Таблиця відображає дані з бази даних у вигляді таблиці з колонками для імені, електронної пошти, телефону та повідомлення. Якщо заявок немає, відображається повідомлення "Немає жодних заявок".

Файл `applications.ejs` потрібен для того, щоб адміністратор міг зручно переглядати та керувати заявками, які надійшли від користувачів. Це допомагає систематизувати та швидко обробляти запити, що важливо для ефективного управління турами та взаємодії з клієнтами. У Лістингу 11 зображено необхідний для цього код.

Лістинг 11 Вміст «applications.ejs»

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta    name="viewport"    content="width=device-width,
initial-scale=1.0">
    <link rel="stylesheet" href="/css/style.css">
    <title><%= title %></title>
</head>
<body>
    <h1>Заявки</h1>
    <table>
        <thead>
            <tr>
                <th>Ім'я</th>
                <th>Прізвище</th>
                <th>Номер телефону</th>
                <th>Пошта</th>
                <th>Дата</th>
                <th>Тур</th>
                <th>Гід</th>
            </tr>
        </thead>
        <tbody>
```

```
            <% applications.forEach(application => { %>
                <tr>
                    <td><%= application.firstName %></td>
                    <td><%= application.lastName %></td>
                    <td><%= application.phone %></td>
                    <td><%= application.email %></td>
                    <td><%= application.date %></td>
                    <td><%= application.tour %></td>
                    <td><%= application.guide %></td>
                </tr>
            <% }); %>
        </tbody>
    </table>
</body>
</html>
```

Створили файл `models/Application.js` для управління даними заявок в базі даних. Цей файл визначає структуру та властивості моделі заявки, а також дозволяє нам виконувати CRUD-операції (створення, читання, оновлення, видалення) з заявками в базі даних за допомогою Sequelize ORM.

Модель `Application` визначає, які поля будуть зберігатися в базі даних для кожної заявки (наприклад, ім'я, електронна пошта, телефон, повідомлення).

Модель визначає типи даних для кожного поля (наприклад, STRING для імені та електронної пошти, TEXT для повідомлення).

Використовуючи цю модель, ми можемо легко створювати нові записи, читати існуючі, оновлювати та видаляти заявки з бази даних.

Пояснення :

- name: Поле для зберігання імені користувача, обов'язкове для заповнення.
- email: Поле для зберігання пошти користувача, обов'язкове для заповнення.
- phone: Поле для зберігання номера телефону користувача, обов'язкове для заповнення.
- message: Поле для зберігання повідомлення від користувача, обов'язкове для заповнення.
- timestamps: Включення автоматичного створення полів `createdAt` та `updatedAt` для запису часу створення та оновлення заявки.

Завдяки цьому файлу ми можемо легко взаємодіяти з базою даних, наприклад:

Створення `models/Application.js` дозволяє нам централізовано керувати всіма аспектами роботи з заявками в базі даних, забезпечуючи ефективне зберігання та обробку даних користувачів. У Лістингу 12 зображено необхідний для цього код.

Лістинг 12: Вміст models/Applications.js

```
const { DataTypes } = require('sequelize');
const sequelize = require('../config/database');

const Application = sequelize.define('Application', {
    firstName: {
        type: DataTypes.STRING,
        allowNull: false
    },
    lastName: {
```

```
        type: DataTypes.STRING,
        allowNull: false
    },
    phone: {
        type: DataTypes.STRING,
        allowNull: false
    },
    email: {
        type: DataTypes.STRING,
        allowNull: false
    },
    date: {
        type: DataTypes.DATE,
        allowNull: false
    },
    tour: {
        type: DataTypes.STRING,
        allowNull: false
    },
    guide: {
        type: DataTypes.STRING,
        allowNull: false
    }
});

module.exports = Application;
```

Також додаю папку, у якій зберігаю фото для цього проекту, як зображено на Рис.12

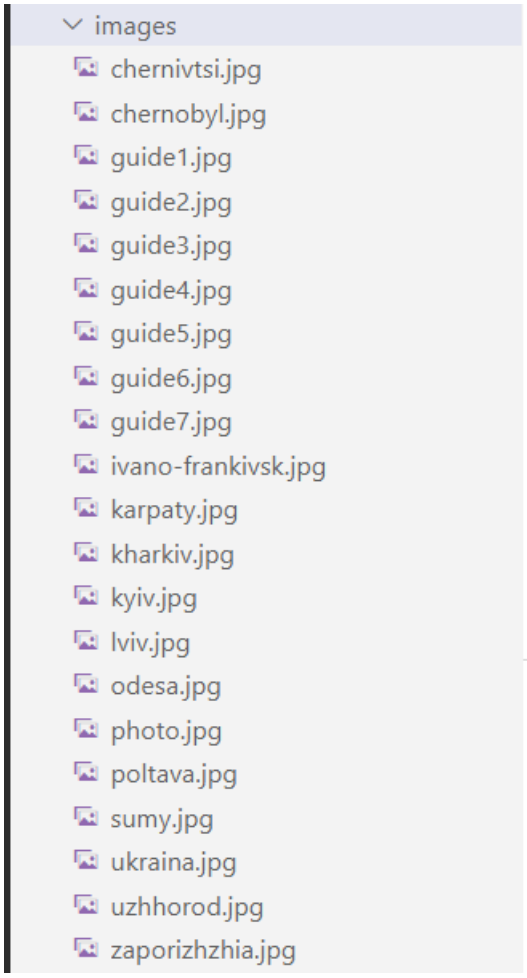


Рисунок 12 – Файл з необхідними фото

До стилів CSS додала наступне:

1. Стилі для адмінської сторінки (логін і перегляд заявок)
    a. Центрування та оформлення форми логіну
    b. Список заявок у вигляді таблиці
    c. Загальні стилі для кнопок та заголовків.
2. Стилі для форми подання заявки та відгуків:
    a. Оформлення форми.
    b. Поліпшення полів вводу і текстових областей.
    c. Стилі для кнопки відправки.
3. Загальні покращення
    a. Фоновий колір.

b. Вирівнювання тексту.
c. Стилі для заголовків.

Опис доданих стилів:

1. Основні стилі:
   1.1. Встановлено базові стилі для всього сайту, включаючи загальні стилі для тіла (`body`), заголовків, контейнерів, тощо.
   1.2. Встановлено стилі для шапки сайту (`header`), включаючи кольори, відступи, та вирівнювання.
2. Стилі для адмінської сторінки:
   2.1. Створено стилі для форми входу адміністраторів, щоб вона була по центру екрану з приємним оформленням.
   2.2. Створено стилі для таблиці заявок, щоб дані були чітко структуровані та читабельні.
3. Стилі для форм подання заявки та відгуків:
   3.1. Створено стилі для форми подання заявки та форми подання відгуків з чіткими полями вводу та кнопками відправки.
4. Стилі для відображення відгуків:
   4.1. Відгуки мають приємне оформлення з відступами, тінями та виділеними заголовками.

3.7 Тестування

Підготовка середовища для тестування

1. Встановлення всіх залежностей:
   Виконали команду `npm install`, щоб встановити всі необхідні пакети.
2. Запуск сервера:
   Використали команду `npm start` для запуску сервера.

Перевірили, чи сервер правильно працює, перейшовши на http://localhost:3000 у веб-браузері.

Тестування функціональності веб-сайту

1. Перевірка головної сторінки:
    a. Перейшли на головну сторінку і переконалися, що вона правильно завантажується, і всі посилання на інші сторінки працюють.
    b. Перевірили наявність основних елементів, таких як заголовки, навігаційне меню, і переконалися, що вони правильно стилізовані.
2. Тестування сторінки «Про нас»:
    a. Перевірили, чи сторінка правильно завантажується і містить всю необхідну інформацію про компанію.
    b. Переконалися, що стилі застосовані коректно, і сторінка виглядає естетично привабливою.
3. Тестування сторінки «Гіди»:
    a. Перевірили, чи сторінка відображає інформацію про гідів.
    b. Переконалися, що стилі відображаються правильно, і інформація зручна для читання.
4. Тестування сторінки «Цікаві місця»:
    a. Перевірили, чи сторінка містить інформацію про цікаві місця для відвідування.
    b. Переконалися, що контент сторінки відображається коректно і відповідає стилям сайту.
5. Тестування сторінки «Тури»:
    a. Перевірили, чи сторінка містить інформацію про доступні тури.
    b. Переконалися, що сторінка правильно стилізована і зручна для користувачів.
6. Тестування форми подання відгуків:

a. Перейшли на сторінку відгуків, заповнили форму та відправили її. Результати коректної роботи зображено на Рис.13
b. Перевірили, чи з’являється новий відгук у списку відгуків на сторінці. Результати коректної роботи зображено на Рис.14

7. Тестування форми подання заявок:
    a. Перейшли на сторінку подання заявок, заповнили форму та відправили її.
    b. Перевірили, чи з’являється повідомлення про успішне відправлення заявки. Результати коректної роботи зображено на Рис.15
    c. Перевірили базу даних, щоб переконатися, що заявка збережена правильно.
8. Тестування адмінської сторінки:
    a. Перейшли на сторінку адміністратора http://localhost:3000/admin.
    b. Ввели пароль для входу і переконалися, що доступ до сторінки обмежений тільки для авторизованих користувачів. Результати коректної роботи зображено на Рис.16
    c. Перевірили сторінку перегляду заявок, щоб переконатися, що всі заявки відображаються правильно. Результати коректної роботи зображено на Рис.17

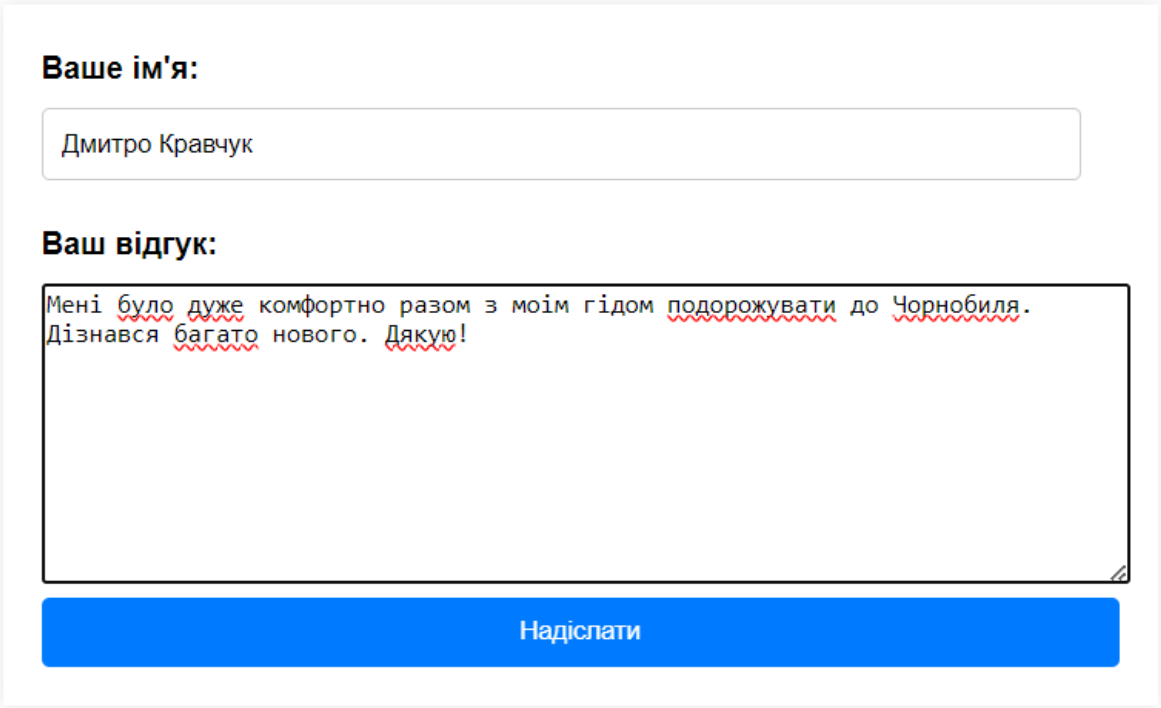


Рис. 13 Відгук

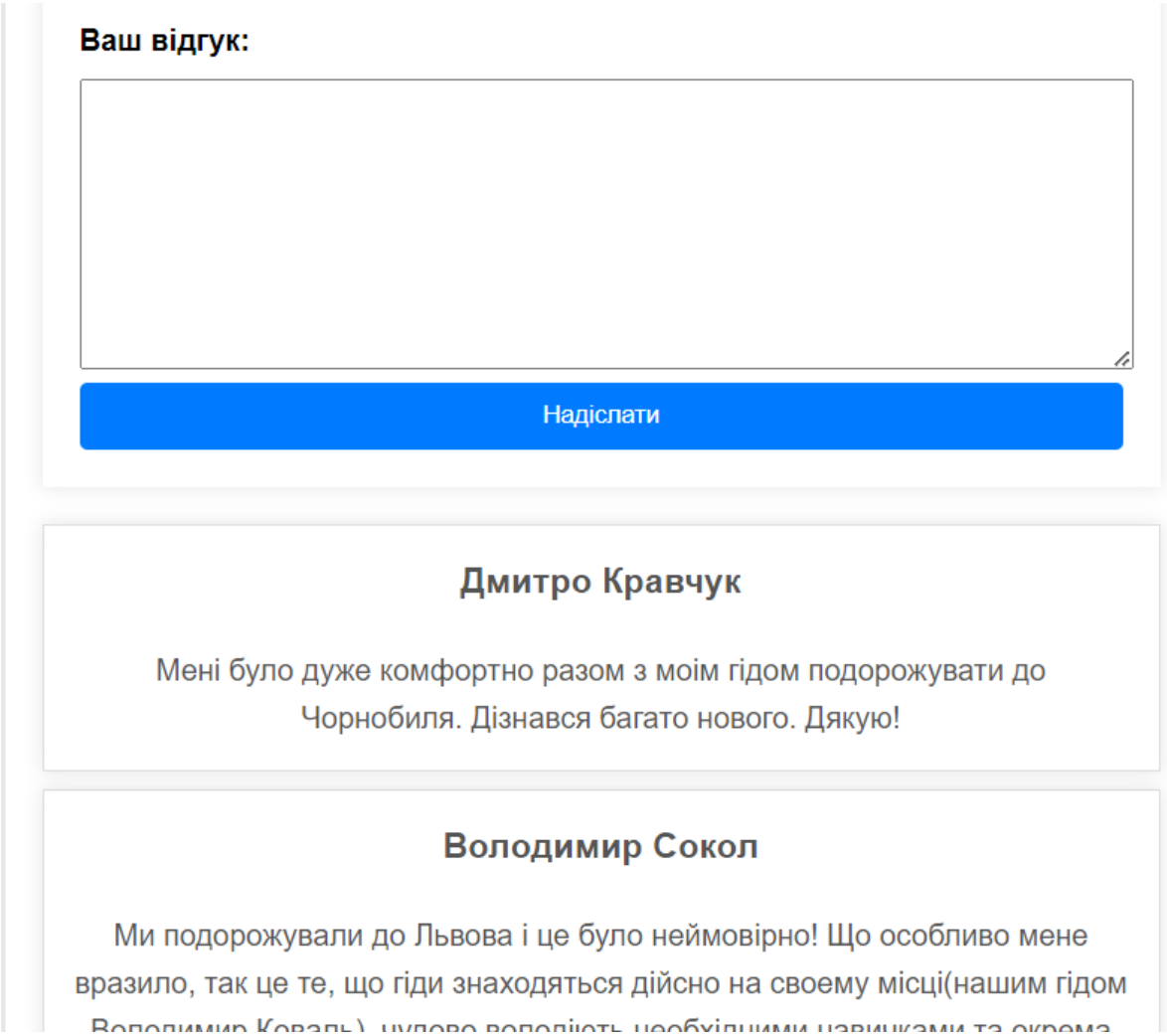


Рис. 14 Публікація відгуку

Рис. 15 Повідомлення про успішне відправлення заявки

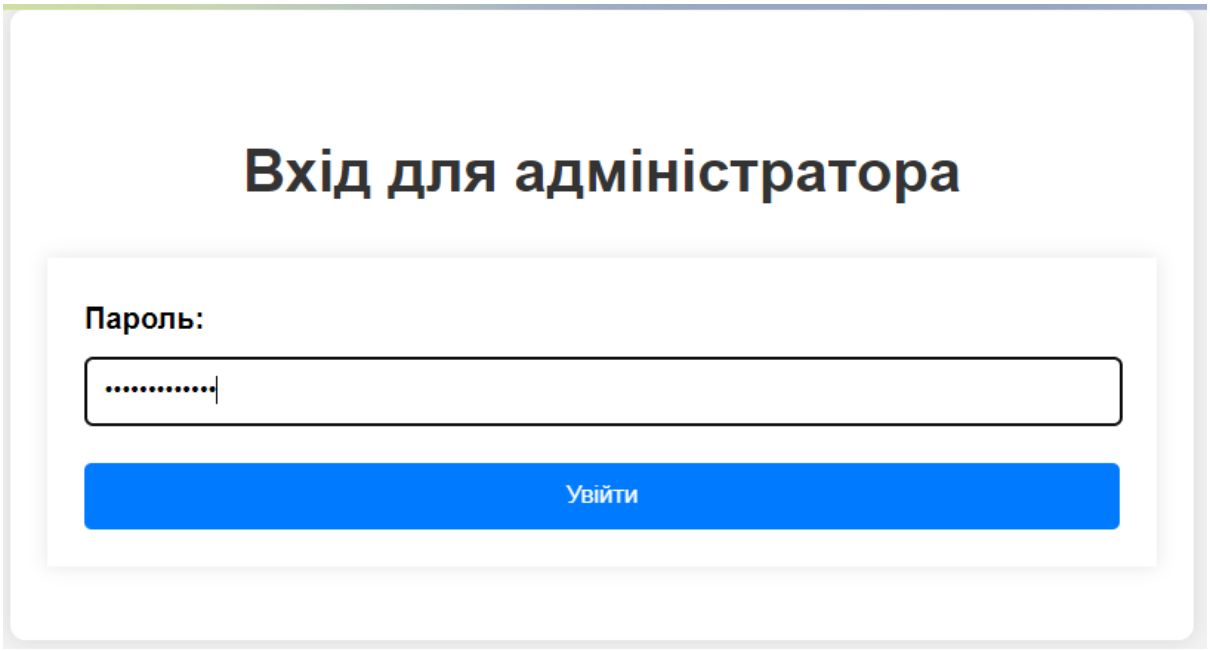
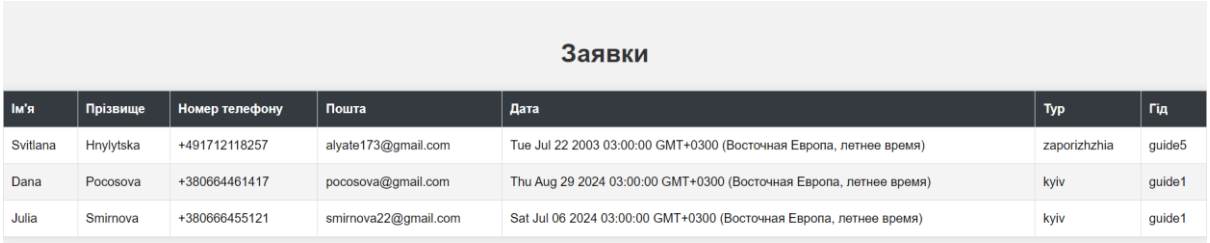


Рис. 16 Доступ до сторінки Адміністратора



Заявки

Ім'я	Прізвище	Номер телефону	Пошта	Дата	Тур	Гід
Svitlana	Hnylytska	+491712118257	alyate173@gmail.com	Tue Jul 22 2003 03:00:00 GMT+0300 (Восточная Европа, летнее время)	zaporizhzhia	guide5
Dana	Pocosova	+380664461417	pocosova@gmail.com	Thu Aug 29 2024 03:00:00 GMT+0300 (Восточная Европа, летнее время)	kyiv	guide1
Julia	Smirnova	+380666455121	smirnova22@gmail.com	Sat Jul 06 2024 03:00:00 GMT+0300 (Восточная Европа, летнее время)	kyiv	guide1

Рис. 17 Сторінка перегляду заявок

Інтеграційне тестування

1. Тестування переходів між сторінками:
    a. Перевірили, чи всі посилання та маршрути працюють правильно.

   b. Переконалися, що користувач може легко переходити між сторінками без помилок.
2. Тестування обробки помилок:
   a. Викликали помилкові маршрути та перевірили, чи відображаються відповідні сторінки помилок (404, 500).
   b. Переконалися, що сервер правильно обробляє некоректні запити.

Тестування продуктивності

1. Перевірка часу завантаження сторінок:
   a. Використали інструменти розробника у веб-браузері для перевірки часу завантаження сторінок.
   b. Оптимізували статичні файли та зображення для покращення продуктивності.
2. Тестування під навантаженням:
   a. Використали інструменти для навантажувального тестування, такі як Apache JMeter, для перевірки поведінки сервера під час високих навантажень.
   b. Переконалися, що сервер може обробляти кілька запитів одночасно без збоїв.

Тестування веб-сайту виявило декілька початкових помилок, які були успішно виправлені. В результаті сайт працює стабільно, надає всі необхідні функції та забезпечує хороший користувацький досвід.

3.8 Висновки до розділу 3

1. Було оглянуто опис предметної області та визначені загальні вимоги, щодо реалізації туристичного веб-сайту.
2. Визначені основні вимоги до інтерфейсу веб-сайту.
3. Розглянута реалізація веб-сайту та тестування на коректність роботи веб-сайту та бази даних зі збереженням заявок та публікацій відгуків.

ВИСНОВКИ

1. В результаті виконання кваліфікаційної роботи було створено туристичний веб-сайт про подорожі по Україні з використанням Node.js, Express.js, EJS та базами даних.
2. Проект розробки веб-сайту потребує чіткого планування структури та логіки додатку. Початкове налаштування середовища розробки, створення базової структури проекту, ініціалізація необхідних пакетів та налаштування роутів є важливими етапами, що забезпечують основу для подальшої роботи.
3. Використання таких інструментів, як ESLint, Prettier, і Node.js Extension Pack, значно підвищує якість коду, робить його більш читабельним і підтримуваним. Sequelize ORM спрощує роботу з базою даних, дозволяючи абстрагуватися від низькорівневих SQL-запитів.
4. При створенні додатків, що працюють з персональними даними користувачів, особливу увагу слід приділяти їх захисту. В цьому проекті було реалізовано обмежений доступ до конфіденційних даних через адмінську панель з паролем, що забезпечує збереження приватної інформації.
5. Окрім функціональності, велике значення має зовнішній вигляд та зручність користування веб-сайтом. Сторінки було стилізовано за допомогою CSS, що забезпечило приємний та інтуїтивно зрозумілий інтерфейс для користувачів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Database https://moodle.znu.edu.ua/pluginfile.php/402598/mod_resource/content/1/%D0%9F%D1%96%D0%B4%D1%80%D1%83%D1%87%D0%BD%D0%B8%D0%BA.PDF (дата звернення 20.05.2024 р).
2. Web-programming https://moodle.znu.edu.ua/pluginfile.php/380213/mod_resource/content/1/%D0%9F%D1%96%D0%B4%D1%80%D1%83%D1%87%D0%BD%D0%B8%D0%BA.pdf (дата звернення 20.05.2024 р).
3. Web-programming https://sites.znu.edu.ua/webprog/(дата звернення 20.05.2024 р).
4. Node JS https://blog.skillfactory.ru/glossary/node-js/(дата звернення 20.05.2024 р).
5. Node JS https://dan-it.com.ua/blog/chto-jeto-takoe-node-js-prostymi-slovami/(дата звернення 20.05.2024 р).
6. Webside https://skillbox.ru/media/marketing/chto-takoe-vebsayt-kak-on-rabotaet-i-kak-sozdayut-sayty/ (дата звернення 20.05.2024 р).
7. HTML and CSS https://www.w3schools.com/html/html_css.asp (дата звернення 20.05.2024 р).
8. Express JS https://www.codecademy.com/article/what-is-express-js
9. Sequelize https://www.digitalocean.com/community/tutorials/how-to-use-sequelize-with-node-js-and-mysql (дата звернення 20.05.2024 р).
10. MDN Web Docs https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs (дата звернення 20.05.2024 р).
11. Node.js Official Documentation https://nodejs.org/en/docs/ (дата звернення 20.05.2024 р).
12. Express.js Official Documentation https://expressjs.com/en/4x/api.html (дата звернення 20.05.2024 р).

13. FreeCodeCamp https://www.freecodecamp.org/learn/apis-and-microservices/ (дата звернення 20.05.2024 р).
14. Udemy - The Complete Node.js Developer Course (3rd Edition https://www.udemy.com/course/the-complete-nodejs-developer-course-2/
15. https://www.planttext.com/#google_vignette (дата звернення 20.05.2024 р).
16. Coursera - Server-side Development with NodeJS, Express and MongoDB https://www.coursera.org/learn/server-side-nodejs (дата звернення 20.05.2024 р).

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Гнилицька Світлана Юріївна, студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ, спеціальність 121 Інженерія програмного забезпечення освітньої програми “Програмне забезпечення систем”, адреса електронної пошти ipz20z-01@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему « Побудова веб-сайту надання послуг з Node.js та Express.js» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

-згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 20.05.2024 _____ _Гнилицька Світлана Юріївна
(студент)

Дата 20.05.2024 _____ Попівщий Василь Іванович)
(керівник)