ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка Web-застосунку для подорожей світом з використанням HTML, CSS, JS та PHP»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Артем МАНУШКІН
(підпис)

Виконав: здобувач вищої освіти групи ПД-42

Артем МАНУШКІН

Керівник: Тимур ДОВЖЕНКО
к.т.н.

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Манушкіну Артему Євгеновичу

1. Тема кваліфікаційної роботи: «Розробка Web-застосунку для подорожей світом з використанням HTML, CSS, JS та PHP»
керівник кваліфікаційної роботи к.т.н, доцент кафедри ІПЗ Тимур ДОВЖЕНКО, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи роботи веб-застосунок для подорожей світом,

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

1. Огляд та аналіз існуючих методів та технологій при розробці сайту на якому зібрані всі можливі поїздки світом.
2. Проектування веб-застосунку для подорожей світом.
3. Програмна реалізація та опис функціонування застосунку для веб-застосунку для подорожей світом.
4. Тестування застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Блок-схеми.
3. Структура бази даних.
4. Мапа веб-застосунку.
5. Екранні форми.
6. Апробація результатів дослідження.

6. Дата видачі завдання«28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Огляд існуючих алгоритмів розробки веб-застосунка для подорожей світом	14.03-21.03.2024	
4	Проектування веб-застосунку для подорожей світом	23.03-28.03.2024	
5	Програмна реалізація застосунку	01.04-10.04.2024	
6	Тестування застосунку	12.04-26.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Артем МАНУШКІН
(підпис)

Керівник
кваліфікаційної роботи _____ Тимур ДОВЖЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 57 стор., 11 рис., 7 джерел.

Мета роботи – Розробити інформаційну архітектуру сайту, включаючи базу даних для зберігання інформації про користувачів, подорожі, місця та послуги.

Об'єкт дослідження – процес побудови веб-застосунку для подорожей світом.

Предмет дослідження – програмне забезпечення для розробки веб-застосунку для подорожу світом.

Короткий зміст роботи: Ця дипломна робота присвячена розробці веб-застосунку для подорожей світом з використанням HTML, CSS, JavaScript та PHP. Метою роботи є створення функціонального та зручного інструменту, який дозволить користувачам планувати свої подорожі, знаходити та бронювати туристичні послуги, а також ділитися своїм досвідом з іншими мандрівниками. У процесі роботи було проведено аналіз ринку та вимог користувачів, спроектовано архітектуру застосунку та базу даних, розроблено користувацький інтерфейс та серверну частину. Особлива увага приділялася безпеці та захисту даних користувачів. Веб-застосунок був протестований та оптимізований для забезпечення стабільної та швидкої роботи. Результатом роботи є інтерактивний веб-застосунок, який сприяє покращенню користувацького досвіду у сфері планування подорожей та надає якісні онлайн-сервіси.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУНОК, ПОДОРОЖІ, HTML, CSS, JAVASCRIPT, PHP, ПЛАНУВАННЯ ПОДОРОЖЕЙ.

ЗМІСТ

1 АНАЛІЗ ПРОБЛЕМИ ТА ІСНУЮЧИХ ЗАСОБІВ

1.1Визначення проблеми та актуальність теми

У сучасному світі зростає попит на інформацію про подорожі та можливості для їх планування. Індустрія туризму постійно розвивається, пропонуючи нові напрямки, сервіси та зручності для мандрівників. В контексті цього розвитку велике значення набирає доступність та зручність інструментів для планування подорожей.

Зростаюча доступність до Інтернету та мобільних пристроїв дозволяє користувачам з легкістю шукати інформацію про подорожі, бронювати готелі, купувати квитки та взагалі керувати своїми мандрівками. Однак існуючі ресурси часто обмежені в своїх можливостях, або недостатньо зручні для використання.

Розробка веб-застосунку для подорожей стає актуальною задачею в контексті постійного підвищення вимог до функціональності та зручності використання. Спрощення процесу планування подорожей та забезпечення доступу до всієї необхідної інформації на одній платформі може значно полегшити життя мандрівників та зробити подорожі більш доступними та приємними.

Однією з основних проблем, яку вирішує ця бакалаврська робота, є недостатня функціональність та нестача зручних інструментів для планування подорожей на існуючих веб-платформах. Багато з них пропонують лише базовий функціонал, не забезпечуючи повного спектру можливостей для користувачів. Також існує проблема розпорошеності інформації: користувачам доводиться витрачати багато часу на пошук та порівняння різних сервісів та пропозицій.

Іншою проблемою є нестабільність деяких існуючих веб-додатків та їх неадаптованість до різних типів пристроїв та розмірів екранів. Це може призвести до незручностей для користувачів, які хочуть планувати подорожі на ходу за допомогою мобільних пристроїв.

Також важливо враховувати проблему безпеки даних користувачів. При зборі та обробці особистої інформації (такої як дані банківських карток, адреси проживання тощо) додатки повинні забезпечувати високий рівень захисту, щоб уникнути можливості крадіжки чи зламу цих даних.

Отже, розробка веб-застосунку для подорожей з врахуванням цих проблем та вирішення ними стає актуальною та важливою задачею, що може принести значний внесок у сферу туризму та покращити досвід користувачів.

1.2 Аналіз існуючих web-додатків для подорожей

У світі туризму існує велика кількість веб-додатків, які пропонують різноманітні сервіси для планування та організації подорожей. Давайте розглянемо деякі з найпопулярніших сервісів та їх особливості:

Booking.com

Booking.com є одним з найбільших веб-сайтів для бронювання готелів, апартаментів та інших видів житла по всьому світу. Він пропонує великий вибір варіантів розміщення за різними ціновими категоріями та зручними фільтрами для пошуку.

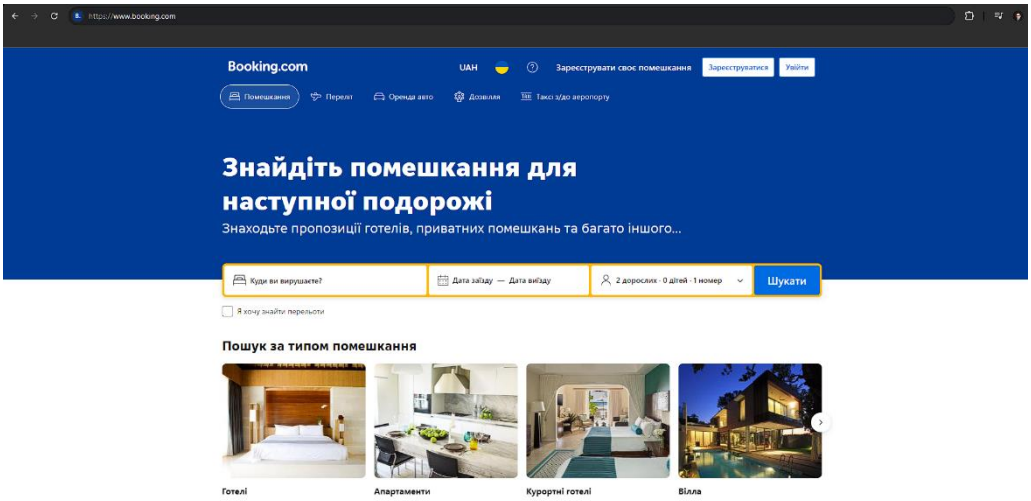


Рис. 1.2.1 Головна сторінка веб-застосунку Booking.com

Airbnb

Airbnb – це платформа для бронювання помешкань в приватних будинках та квартирах. Вона надає можливість орендувати житло від місцевих мешканців, що дозволяє подорожуючим отримати більш аутентичний досвід місцевого життя.

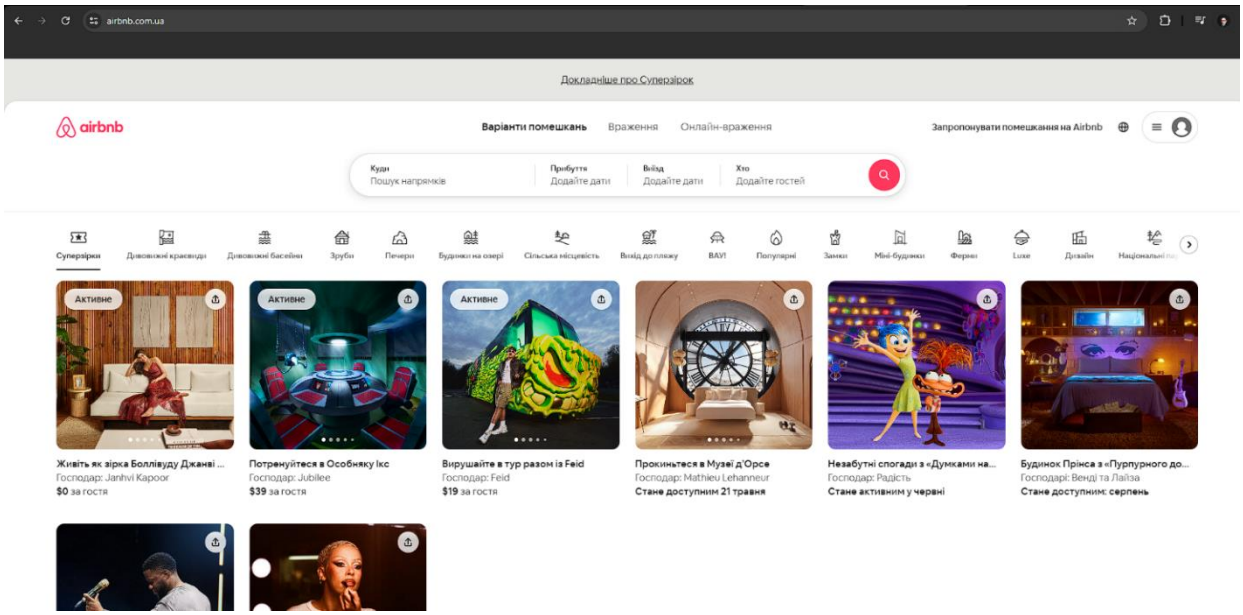


Рис. 1.2.2 Головна сторінка веб-застосунку Airbnb

TripAdvisor

TripAdvisor – це веб-сайт, який зосереджений на відгуках та рекомендаціях від користувачів щодо готелів, ресторанів, екскурсій та інших послуг пов’язаних з подорожами. Він надає корисні відгуки та оцінки, які допомагають подорожуючим зробити кращі вибори.

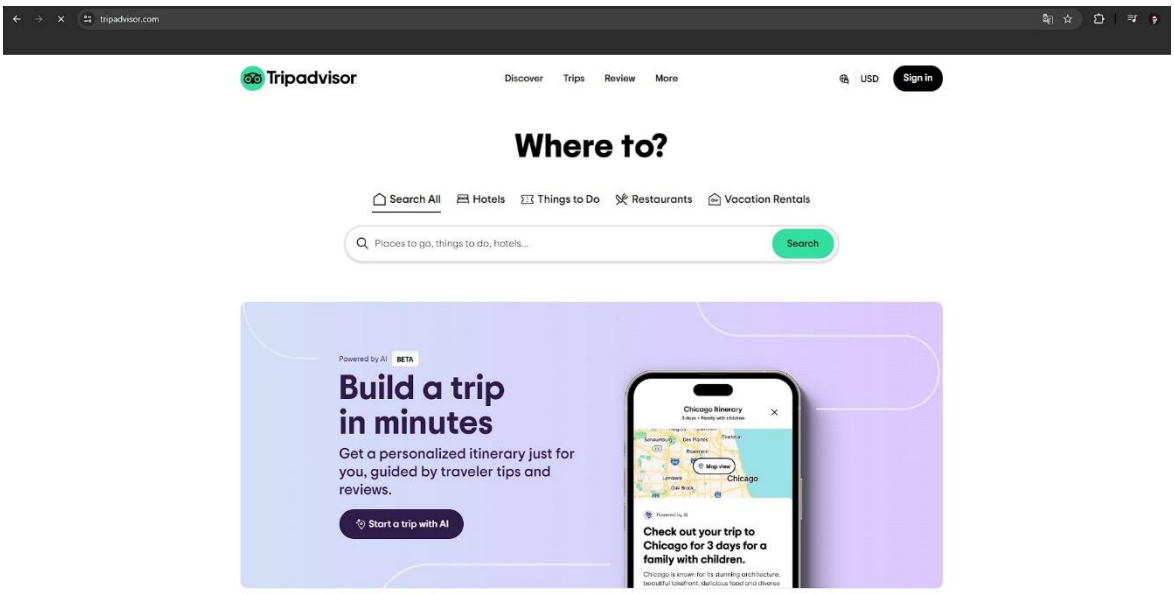


Рис. 1.2.3 Головна сторінка веб-застосунку TripAdvisor

Google Подорожі

Google Подорожі – це веб-застосунок від Google, який надає користувачам можливість створювати свої власні маршрути, отримувати рекомендації щод місць для відвідування та зберігати всю необхідну інформацію про подорож на одному місці.

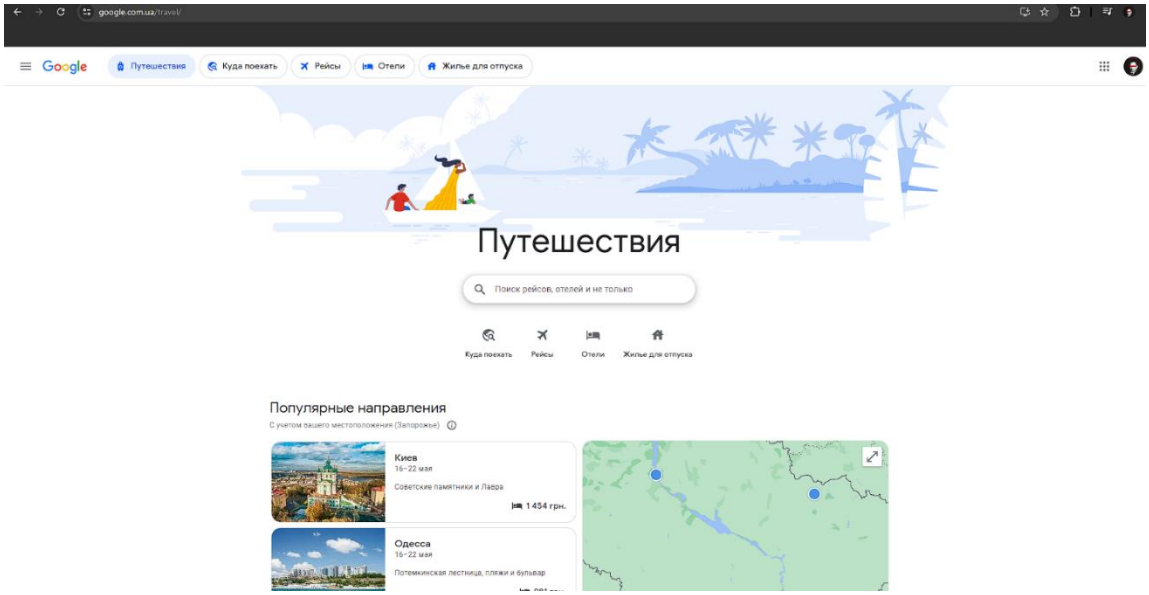


Рис. 1.2.4 Головна сторінка веб-застосунку Google Подорожі

Expedia

Expedia – це веб-сайт для бронювання готелів, авіаквитків, пакетних турів та інших послуг для подорожей. Він пропонує широкий вибір варіантів та можливість зберігати гроші завдяки пакетним пропозиціям.

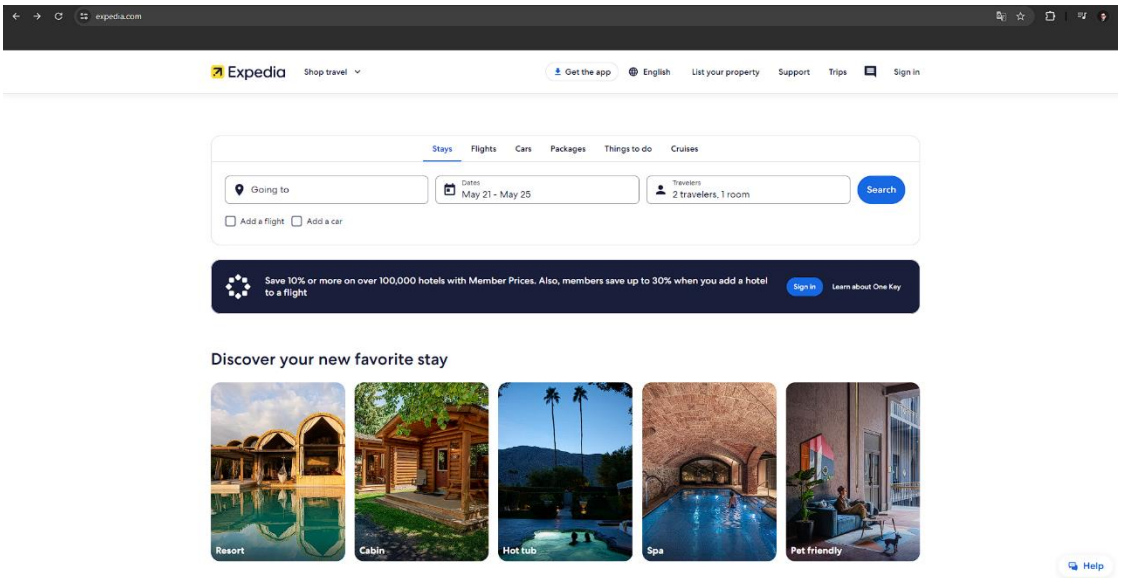


Рис. 1.2.5 Головна сторінка веб-застосунку Expedia

Після представлення існуючих веб-додатків розглянемо основні переваги та недоліки представлених аналогів для розробки нашого власного застосунку

Переваги:

- **Широкий вибір**: Більшість існуючих веб-додатків пропонують широкий вибір варіантів розміщення, транспорту та розваг для подорожуючих.
- **Корисні рекомендації**: Деякі сервіси надають корисні рекомендації та відгуки від інших користувачів, що допомагає зробити кращі вибори.
- **Зручність використання**: Багато сервісів мають інтуїтивно зрозумілий і легкий у використанні інтерфейс, що полегшує процес планування подорожей.

Недоліки:

- **Обмежена функціональність**: Деякі сервіси можуть бути обмежені у своїх можливостях, не надаючи всіх необхідних функцій для повного планування подорожі.
- **Високі комісійні внески**: Деякі платформи можуть брати великі комісійні внески за бронювання, що може збільшити вартість подорожі для користувачів.
- **Нестабільність додатків**: Деякі веб-додатки можуть бути нестабільними або неадаптованими до різних пристроїв, що може призвести до незручностей для користувачів.

Існуючі веб-додатки для подорожей мають свої переваги та недоліки. Розуміння цих аспектів дозволяє розробникам створити більш ефективні та зручні інструменти для планування та організації подорожей.

1.3 Визначення основних вимог до web-застосунку

Функціональні вимоги визначають основні функції та можливості, які має мати веб-застосунок для подорожей. Розробка цих вимог є ключовим етапом у процесі планування та розробки продукту.

Можливість пошуку та бронювання розміщення

Користувачі повинні мати можливість шукати та бронювати різноманітні варіанти розміщення (готелі, апартаменти, помешкання на Airbnb тощо). Застосунок повинен надати можливість фільтрувати результати за різними критеріями, такими як ціна, рейтинг, розташування тощо.

Планування маршрутів та екскурсій

Веб-застосунок повинен надати користувачам інструменти для планування своїх маршрутів та відвідування цікавих місць. Це може включати в себе можливість створення списків пам’яток, визначення оптимальних маршрутів та доступ до інформації про транспортні засоби та розклади.

Інтеграція з картами та навігаційними сервісами

Для зручності користувачів важливо мати доступ до інтерактивних карт, які дозволяють відображати розташування різних об’єктів та маршрутів. Також важлива можливість використання навігаційних сервісів для планування маршрутів та переміщення по місту чи країні.

Збереження особистої інформації та історії подорожей

Веб-застосунок повинен надати можливість зберігати особисту інформацію користувачів, таку як історія бронювань, попередні маршрути та уподобання. Це дозволить користувачам швидше здійснювати бронювання та отримувати персоналізовані рекомендації.

Забезпечення безпеки даних

Однією з ключових функцій є забезпечення безпеки особистих даних користувачів. Застосунок повинен використовувати захистні протоколи для збереження конфіденційної інформації, такої як дані банківських карток та особисті дані.

Нефункціональні вимоги визначають якість та характеристики, які має мати веб-застосунок, а не його функціональність безпосередньо. Ці вимоги визначають якість користувацького досвіду та стійкість системи.

Продуктивність

Застосунок повинен працювати ефективно та швидко, навіть при великому обсязі даних та одночасному доступі великої кількості користувачів.

Надійність

Система повинна бути надійною та стійкою до відмов, забезпечуючи безперебійну роботу навіть при можливих ситуаціях відмов.

Сумісність з різними пристроями та браузерами

Застосунок повинен бути сумісним з різними типами пристроїв (комп’ютери, планшети, смартфони) та різними браузерами для забезпечення зручного користування для всіх користувачів.

Інтерфейс користувача

Інтерфейс користувача повинен бути зручним та інтуїтивно зрозумілим для користувачів будь-якого рівня технічної підготовки.

Безпека

Система повинна мати високий рівень безпеки для захисту особистих даних користувачів та запобігання несанкціонованому доступу до них.

Визначення функціональних та нефункціональних вимог є важливим етапом у процесі розробки веб-застосунку для подорожей. Ці вимоги визначають обсяг та характеристики продукту, який має задовольнити потреби користувачів та забезпечити їм приємний та безпечний досвід подорожування.

1.4 Постановка задач бакалаврської роботи

Метою цієї бакалаврської роботи є розробка веб-застосунку для подорожей, який надає користувачам зручні та ефективні інструменти для планування, бронювання та організації подорожей світом. Основним завданням дослідження є створення функціонального та ефективного веб-застосунку, який забезпечить користувачам високий рівень зручності, надійності та безпеки під час планування та здійснення подорожей.

- **Аналіз існуючих веб-додатків для подорожей**: Провести огляд та аналіз популярних сервісів для планування подорожей, визначити їх переваги та недоліки, а також виявити недоліки, які можна врахувати при розробці власного застосунку.
- **Визначення вимог до веб-застосунку**: Сформулювати функціональні та нефункціональні вимоги до веб-застосунку, що відповідають потребам користувачів та забезпечують його якість та ефективність.
- **Проектування системи**: Розробити архітектуру веб-застосунку, спроектувати інтерфейс користувача та базу даних, визначити основні модулі та функціональність системи.
- **Реалізація веб-застосунку**: Розробити програмний код веб-застосунку, забезпечити його функціональність та ефективність, здійснити інтеграцію з внутрішніми та зовнішніми сервісами.

- **Тестування та оптимізація**: Провести тестування розробленого веб-застосунку для перевірки його працездатності, безпеки та ефективності, а також здійснити оптимізацію для підвищення продуктивності та стійкості системи.
- **Документування та підготовка звіту**: Підготувати документацію з розробки веб-застосунку, включаючи технічний опис системи, пояснення архітектури та використані технології, а також підготувати звіт про виконану роботу.
- **Оцінка результатів та висновки**: Проаналізувати отримані результати, визначити переваги та недоліки розробленого веб-застосунку, зробити висновки щодо його ефективності та можливостей подальшого розвитку.

Ці завдання дослідження спрямовані на досягнення мети бакалаврської роботи та розвиток веб-застосунку для подорожей, який відповідає сучасним вимогам користувачів та забезпечує їм зручні та ефективні інструменти для планування та організації подорожей.

2 ОПИС ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Архітектура веб-застосунку: Вибір архітектурного підходу

При проектуванні веб-додатків важливо визначити правильний архітектурний підхід, який забезпечить ефективність, масштабованість та зручність розробки та підтримки. У цьому розділі ми розглянемо різні архітектурні підходи та обгрунтуємо вибір підходу для нашого веб-застосунку для подорожей.

Одномодельна архітектура є традиційним підходом до розробки веб-додатків. У такій архітектурі усі компоненти застосунку, включаючи серверну логіку, базу даних, інтерфейс користувача та інші, розглядаються як один монолітний блок програмного забезпечення.

Переваги:

- **Простота розробки**: Розробка та впровадження відбувається в межах одного проекту, що спрощує розробку та підтримку.
- **Легка масштабованість**: При потребі можна просто масштабувати застосунок, додаючи додаткові ресурси.

Недоліки:

- **Обмежена гнучкість**: Велика кількість коду та функцій може зробити розробку та розгортання складними.
- **Погана масштабованість команд**: Розробка великих проектів може бути складною для розподілу між великою кількістю розробників.

Сервіс-орієнтована архітектура (SOA) розглядає застосунок як набір незалежних сервісів, кожен з яких відповідає за виконання певного функціоналу. Ці сервіси можуть бути незалежно розгорнуті та масштабовані.

Переваги:

- **Гнучкість та масштабованість**: Сервіси можуть бути розгорнуті та масштабовані незалежно один від одного.
- **Відокремленість функцій**: Кожен сервіс відповідає за конкретну функціональність, що сприяє розподілу роботи між різними командами.

Недоліки:

- **Складність управління**: Управління багатьма незалежними сервісами може бути складним та вимагати додаткових інструментів та процедур.

Мікросервісна архітектура є розширенням SOA, де кожен сервіс є невеликим, самостійним додатком, який виконує конкретну функцію. Кожен мікросервіс має власну базу даних та може бути розгорнутий та масштабований окремо.

Переваги:

- **Висока гнучкість та масштабованість**: Кожен мікросервіс може бути масштабований незалежно від інших, що забезпечує гнучкість та ефективність.
- **Відокремленість функцій**: Кожен мікросервіс відповідає за конкретну функціональність, що спрощує розробку та підтримку.

Недоліки:

- **Складність управління та координації**: Управління багатьма незалежними мікросервісами може бути складним та вимагати додаткових інструментів та процедур.

Для нашого веб-застосунку для подорожей ми вибрали **мікросервісну архітектуру**. Цей вибір обумовлений потребою високої гнучкості та масштабованості, щоб забезпечити зручність та ефективність розробки та підтримки застосунку. Крім того, відокремленість функцій у мікросервісній архітектурі дозволить нам легко розподіляти роботу між різними командами розробників та швидко впроваджувати нові функції та оновлення.

Ми також розглянемо можливість використання контейнеризації (наприклад, Docker) та оркестрації контейнерів (наприклад, Kubernetes) для спрощення розгортання та керування мікросервісами. Це дозволить нам забезпечити гнучкість та ефективність в управлінні нашим додатком для подорожей.

2.2 Дизайн інтерфейсу користувача

Дизайн інтерфейсу користувача є ключовим аспектом веб-застосунку для подорожей, оскільки він визначає спосіб взаємодії користувача з додатком та впливає на його враження та задоволення від використання. У цьому розділі ми розглянемо визначення структури та зовнішнього вигляду застосунку для забезпечення максимальної зручності та естетичності.

Структура веб-застосунку для подорожей повинна бути логічною та зручною для користувача. Вона включає в себе розташування основних елементів та розділів, навігаційні меню, сторінки та їх взаємозв’язки. Розглянемо основні компоненти структури застосунку:

Головна сторінка (Головний екран)

На головній сторінці користувач повинен мати доступ до основних функцій застосунку, таких як пошук місць для подорожей, планування маршрутів, відображення рекомендацій та акцій, а також можливість увійти в особистий кабінет.

Сторінки для пошуку та бронювання розміщення

Ці сторінки мають надати користувачу можливість шукати та вибирати різні варіанти розміщення (готелі, апартаменти, помешкання на Airbnb тощо) за допомогою фільтрів та критеріїв.

Сторінки для планування маршрутів та екскурсій

Ці сторінки мають надати користувачу можливість планувати свої маршрути та екскурсії, додавати цікаві місця до списків, переглядати карти та розклади транспорту.

Особистий кабінет

У особистому кабінеті користувач повинен мати можливість переглядати свої бронювання, історію подорожей, змінювати особисту інформацію та налаштування облікового запису.

Зовнішній вигляд застосунку для подорожей має бути привабливим та сучасним, а також відповідати тематиці подорожей та манері використання. Ось деякі ключові аспекти зовнішнього вигляду:

Кольорова палітра

Вибір кольорової палітри може бути пов’язаний з елементами, які асоціюються з подорожами: блакитний колір неба, зелений колір природи, піскові відтінки пустелі тощо. Важливо також врахувати контрастність та читабельність тексту на фоні.

Ілюстрації та фотографії

Використання відповідних ілюстрацій та фотографій може створити атмосферу подорожей та зробити застосунок більш привабливим для користувача. Фотографії місць, пам’яток, природних красот та інших елементів можуть створити емоційне зв’язок з користувачем.

Типографіка

Вибір шрифтів та їх розміщення має велике значення для зручності читання та естетичного вигляду. Рекомендується використовувати чіткі та легко читабельні шрифти, а також правильно розташовувати текстові блоки на сторінці.

Важливим аспектом дизайну є його адаптивність до різних типів пристроїв та розмірів екранів. Застосунок повинен оптимально виглядати як на комп’ютері,

так і на планшеті або смартфоні. Це досягається за допомогою використання резинового дизайну, медіазапитів та інших технік адаптивного дизайну.

Дизайн інтерфейсу користувача відіграє важливу роль у привабливості та зручності веб-застосунку для подорожей. Визначення структури та зовнішнього вигляду допомагає забезпечити ефективну взаємодію з користувачем та позитивне враження від використання застосунку. З метою забезпечення успішної реалізації цих аспектів, ми будемо використовувати сучасні техніки та кращі практики у дизайні інтерфейсу користувача.

2.3 База даних: Проектування схеми та вибір технологій

База даних веб-застосунку для подорожей є одним з ключових компонентів, що забезпечує збереження та організацію даних про користувачів, подорожі, розміщення, маршрути та багато іншого. У цьому розділі ми розглянемо проектування схеми бази даних для нашого застосунку та вибір технологій для роботи з нею.

Для ефективної організації та збереження даних потрібно спроектувати відповідну схему бази даних. Ось ключові сутності та їх взаємозв’язки, які необхідно врахувати:

Користувачі (Users)

- Інформація про користувачів, така як ім’я, електронна пошта, пароль, дата реєстрації тощо.
- Зв’язок з іншими сутностями, наприклад, подорожі, бронювання розміщення тощо.

Подорожі (Trips)

- Деталі про кожну подорож, такі як назва, дата початку та завершення, маршрут, опис тощо.

- Зв’язок з користувачами, що створили подорож, а також з маршрутами, розміщенням тощо.

Маршрути (Routes)

- Інформація про маршрути подорожі, така як назва, точки маршруту, відстані, часи тощо.
- Зв’язок з подорожами, до яких вони відносяться.

Розміщення (Accommodations)

- Деталі про розміщення, такі як назва, адреса, тип (готель, апартаменти тощо), вартість, доступність тощо.
- Зв’язок з подорожами, до яких вони відносяться, а також з бронюванням.

Бронювання (Bookings)

- Інформація про бронювання розміщення користувачами, така як дата, тривалість, ціна тощо.
- Зв’язок з користувачами, які здійснили бронювання, та з розміщенням, яке заброньоване.

При виборі технологій для роботи з базою даних необхідно врахувати такі фактори, як швидкодія, масштабованість, надійність, зручність в роботі та вартість. Ось деякі з найпопулярніших технологій баз даних та їх характеристики:

Relational Database Management Systems (RDBMS)

RDBMS є традиційними системами керування базами даних, які використовують SQL для роботи з даними. Найпопулярніші RDBMS включають MySQL, PostgreSQL, SQLite та Microsoft SQL Server.

Вибір: Для проекту можна обрати PostgreSQL, оскільки він має високу швидкодію, масштабованість та підтримку геоданих, що може бути корисним для подорожного застосунку.

NoSQL Databases

NoSQL бази даних призначені для зберігання та роботи з невідносинними даними, такими як документи, ключ-значення, колонки тощо. Популярні NoSQL бази даних включають MongoDB, Cassandra, Redis та Amazon DynamoDB.

Вибір: MongoDB може бути вибором для проекту, оскільки він дозволяє зберігати структуровані та невідносинні дані, що може бути корисним для зберігання різноманітної інформації про подорожі та користувачів.

Проектування схеми бази даних та вибір технологій для роботи з нею є важливим етапом у розробці веб-застосунку для подорожей. Вирішення правильних архітектурних та технологічних рішень дозволяє забезпечити ефективну роботу з даними та забезпечити високу продуктивність та надійність застосунку. З урахуванням особливостей нашого проекту, ми вибрали PostgreSQL як систему керування базою даних та MongoDB для потреб зберігання невідносинних даних.

Далі розглянемо саму структуру бази даних для нашого застосунку. Розроблена в найпростішому браузерному графічному редакторі. Для прикладу – draw.io

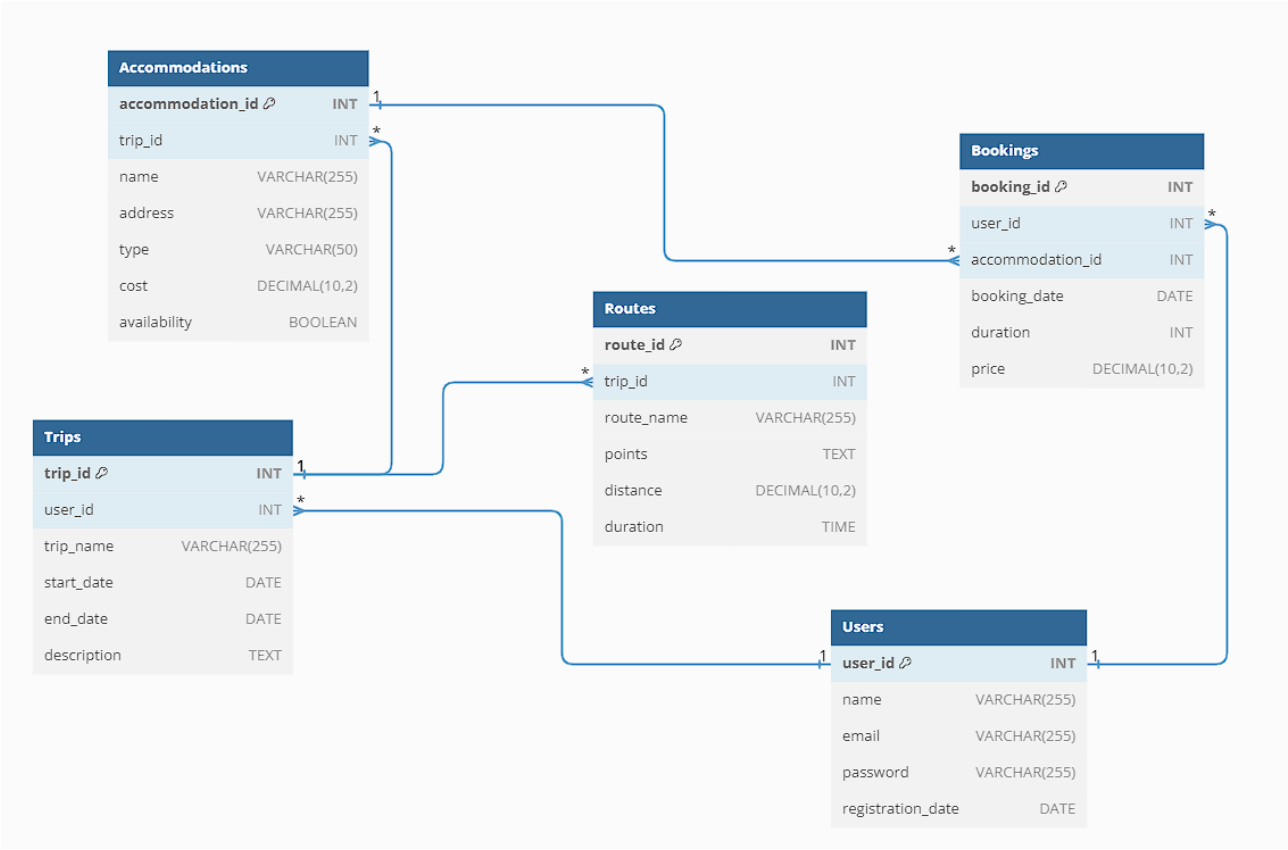


Рис. 2.3.1 ER-діаграма розробленої бази даних для веб-застосунку планування подорожей світом

2.4 Функціональність застосунку

Веб-застосунок для подорожей повинен мати широкий спектр функціональності, щоб задовольнити потреби користувачів у плануванні, бронюванні та відвідуванні різних місць по всьому світу. У цьому розділі ми розглянемо основні можливості застосунку та їх реалізацію.

Реєстрація та авторизація користувачів:

Опис функціональності:

- Користувачі можуть створювати облікові записи за допомогою електронної пошти та пароля або використовувати сторонні сервіси для швидкої реєстрації (наприклад, Google, Facebook).
- Після реєстрації користувачі можуть увійти в свій обліковий запис, використовуючи електронну пошту та пароль або інші методи авторизації.

Реалізація:

- Використання системи аутентифікації, такої як Oauth або JWT, для забезпечення безпеки та авторизації користувачів.
- Зберігання облікових записів у базі даних з хешованими паролями для забезпечення безпеки.

Пошук місць для подорожей:

Опис функціональності:

- Користувачі можуть шукати місця для подорожей за допомогою різних критеріїв, таких як місцевість, тип розваги, відстань від поточного місцезнаходження тощо.
- Можливість перегляду рейтингу та відгуків інших користувачів про місця.

Реалізація:

- Використання алгоритмів пошуку та фільтрації даних для представлення користувачеві релевантних результатів пошуку.
- Інтеграція з сервісами карт (наприклад, Google Maps) для відображення місць на карті та визначення відстані до них від користувача.

Планування маршрутів:

Опис функціональності:

- Користувачі можуть планувати свої маршрути подорожей, додаючи до них цікаві місця, пам’ятки, ресторани тощо.
- Можливість зберігання та поділу маршрутів з іншими користувачами.

Реалізація:

- Розробка інтерактивних інструментів для створення та редагування маршрутів на основі карт.
- Зберігання маршрутів у базі даних та використання системи дозволів для управління доступом до них.

Бронювання розміщення:

Опис функціональності:

- Можливість бронювання готелів, апартаментів, помешкань на Airbnb та інших видів розміщення.
- Перегляд доступності та цін на різні дати та місця.

Реалізація:

- Інтеграція з платіжними системами для прийому платежів та підтвердження бронювань.
- Використання API сервісів бронювання розміщення (наприклад, Booking.com, Airbnb) для отримання актуальної інформації та здійснення бронювань.

Особистий кабінет користувача:

Опис функціональності:

- Можливість перегляду історії подорожей, бронювань, зміни особистих даних та налаштувань облікового запису.
- Зберігання фотографій, відгуків та оцінок користувачів.

Реалізація:

- Створення особистого кабінету для кожного користувача з можливістю авторизації та захисту персональних даних.
- Реалізація функцій завантаження та відображення фотографій, огляду історії та редагування особистих даних.

Функціональність веб-застосунку для подорожей включає широкий спектр можливостей, що відповідають потребам користувачів у плануванні, бронюванні та відвідуванні різних місць по всьому світу. Реалізація цих можливостей вимагає ретельного проектування та розробки функцій для забезпечення зручного та ефективного використання застосунку. З врахуванням вищезазначених функцій, ми будемо розробляти веб-застосунок для подорожей з максимальною увагою до деталей та користувацького досвіду.

3 ОПИС РЕАЛІЗАЦІЇ СИСТЕМИ

3.1 Діаграма класів

Діаграма класів є важливим інструментом для моделювання структури програми та взаємозв'язків між її компонентами. У цьому розділі ми розглянемо основні класи нашого веб-застосунку для подорожей та їх взаємозв'язки.

Клас "Користувач" відповідає за представлення користувачів у системі. Він містить основні атрибути та методи для роботи з обліковими записами користувачів, а також взаємодію з іншими частинами системи.

Атрибути:

- id: унікальний ідентифікатор користувача
- email: електронна пошта користувача
- password: хешований пароль користувача
- name: ім'я користувача
- created_at: дата та час створення облікового запису
- last_login_at: дата та час останнього входу в систему

Методи:

- register(): метод для реєстрації нового користувача в системі
- login(): метод для входу в систему з використанням електронної пошти та пароля
- update_profile(): метод для зміни особистих даних користувача
- view_history(): метод для перегляду історії подорожей та бронювань

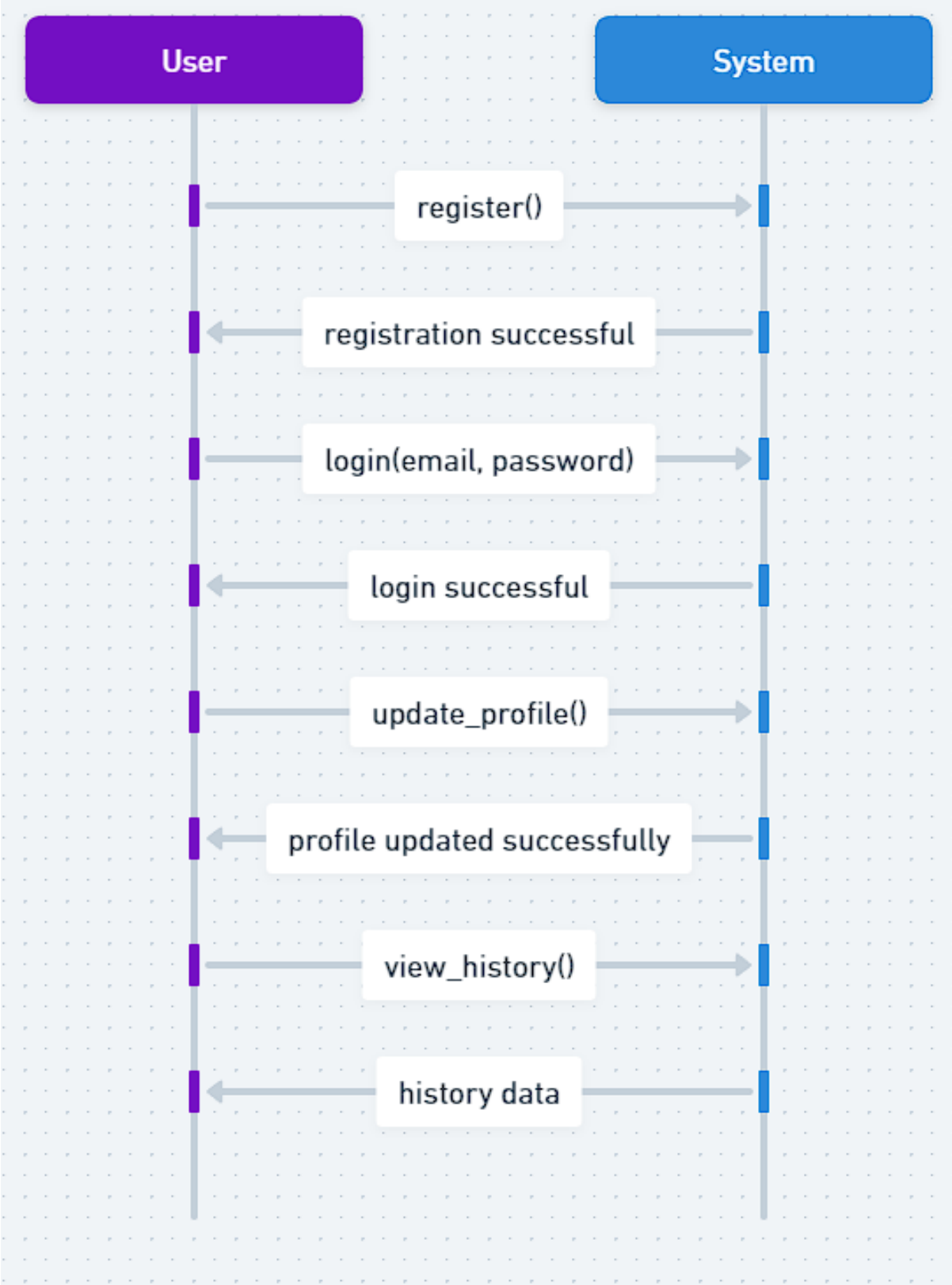


Рис. 3.1 Діаграму класу «Користувач»

Клас "Подорож" відповідає за представлення подорожей у системі. Він містить інформацію про назву подорожі, дату початку та завершення, маршрут та інші деталі, а також забезпечує взаємодію з користувачами.

Атрибути:

- id: унікальний ідентифікатор подорожі
- name: назва подорожі
- start_date: дата початку подорожі
- end_date: дата завершення подорожі
- route: маршрут подорожі
- description: опис подорожі
- creator_id: ідентифікатор користувача, який створив подорож

Методи:

- create(): метод для створення нової подорожі
- edit(): метод для редагування деталей подорожі
- delete(): метод для видалення подорожі
- add_place(): метод для додавання нового місця до маршруту

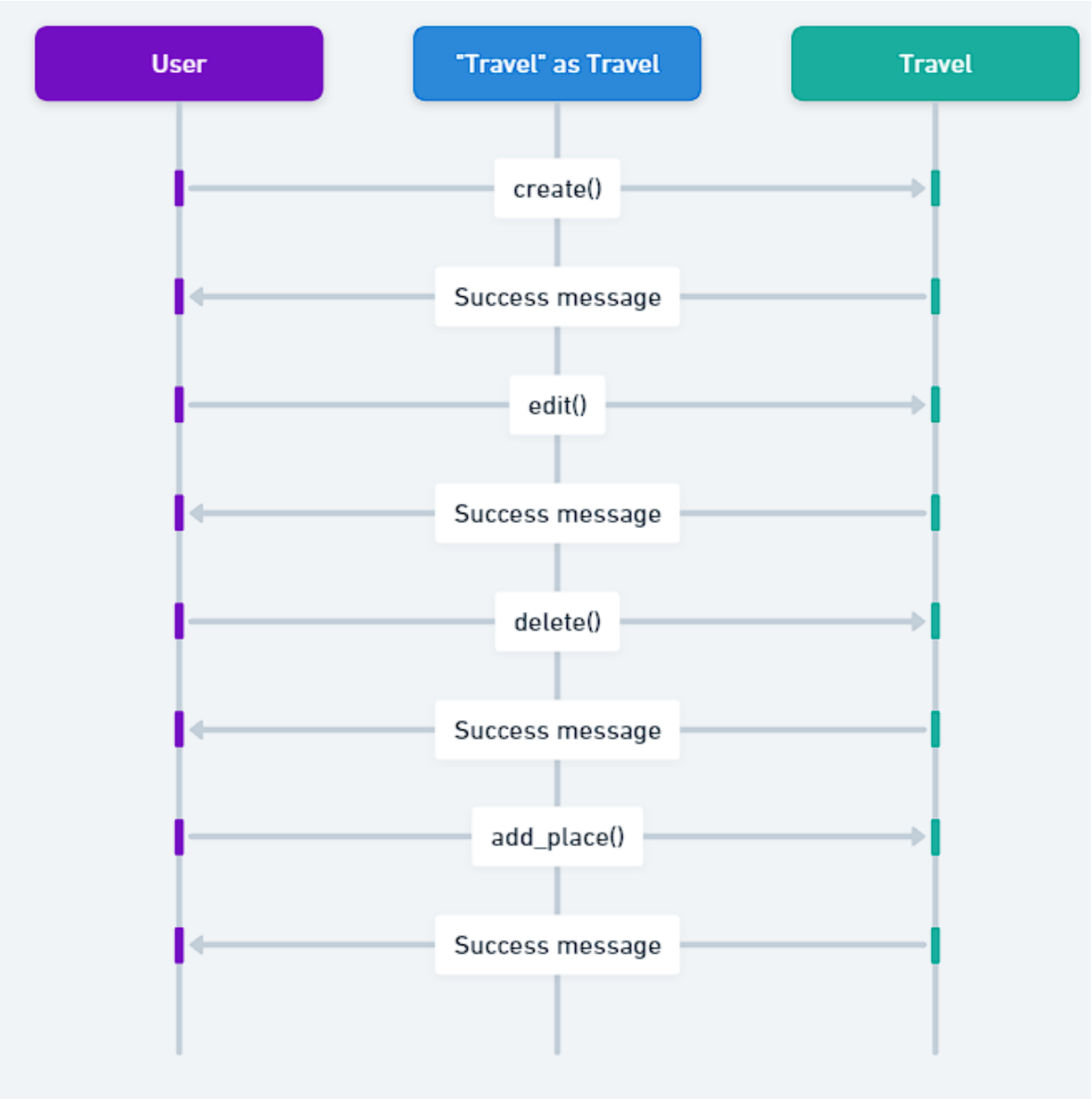


Рис. 3.2 Діаграма класу «Подорож»

Клас "Місце" відповідає за представлення різних місць для подорожей у системі. Він містить інформацію про назву, опис, географічні координати та інші характеристики місць.

Атрибути:

- id: унікальний ідентифікатор місця
- name: назва місця
- description: опис місця
- coordinates: географічні координати місця (широта та довгота)

Методи:

- get_details(): метод для отримання додаткових деталей про місце
- rate(): метод для оцінки місця користувачем
- leave_review(): метод для залишення відгуку про місце

Рис. 3.3 Діаграма класу «Місце»

Клас "Бронювання" відповідає за представлення бронювань розміщення користувачами. Він містить інформацію про дату, тривалість, ціну та інші деталі бронювання.

Атрибути:

- id: унікальний ідентифікатор бронювання
- user_id: ідентифікатор користувача, який здійснив бронювання
- accommodation_id: ідентифікатор розміщення, яке було заброньоване
- check_in_date: дата заїзду
- check_out_date: дата виїзду
- price: вартість бронювання

Методи:

- make_booking(): метод для здійснення бронювання розміщення
- cancel_booking(): метод для скасування бронювання

Після розгляду всіх класів у розробленій базі даних, пропонуємо розглянути зв’язки між нашими таблицями

- Користувачі можуть створювати подорожі та бронювати розміщення.
- Користувачі можуть додавати місця до своїх подорожей.
- Кожна подорож може містити декілька місць та бронювань розміщення.
- Користувачі можуть залишати відгуки та оцінки для місць.

Діаграма класів відображає структуру програми та взаємозв'язки між її компонентами. За допомогою цієї діаграми ми можемо краще зрозуміти, як класи взаємодіють один з одним та які функції вони виконують у системі. Ретельне проектування класів дозволить нам створити добре організовану та ефективну програму для нашого веб-застосунку для подорожей.

3.2 Специфікація ключових класів та їх методів

У цьому розділі ми розглянемо детальну специфікацію ключових класів нашого веб-застосунку для подорожей та їх методів. Кожен клас буде описаний з точки зору його функціональності та взаємодії з іншими частинами системи.

Клас "Користувач" (User)

Цей клас відповідає за представлення користувачів у системі та управління їх обліковими записами. Ось опис його основних методів:

Метод register()

Цей метод дозволяє користувачам зареєструвати новий обліковий запис у системі. Він отримує на вхід дані користувача, такі як ім'я, електронна пошта та пароль, і перевіряє їх на валідність. Після цього створюється новий об'єкт класу "Користувач" у базі даних.

Розглянемо код розроблений на мові програмування php:

```php
<?php

class UserRegistration {

    private $dbConnection;

    // Конструктор класу, приймає підключення до бази даних
    public function __construct($dbConnection) {
        $this->dbConnection = $dbConnection;
    }

    // Метод для реєстрації нового користувача
    public function registerUser($name, $email, $password) {
        // Перевірка валідності введених даних
        if ($this->validateUserData($name, $email, $password)) {
            // Хешування паролю
            $hashedPassword = password_hash($password, PASSWORD_DEFAULT);

            // Додавання користувача до бази даних
            $query = "INSERT INTO users (name, email, password) VALUES (?, ?, ?)";
            $statement = $this->dbConnection->prepare($query);
            $statement->bind_param("sss", $name, $email, $hashedPassword);
            $statement->execute();

            // Повертаємо інформацію про успішну реєстрацію
            return "Користувач успішно зареєстрований!";
```

```
        } else {
            // Повертаємо повідомлення про помилку у валідації даних
            return "Неправильні дані користувача!";
        }
    }

    // Приватний метод для перевірки валідності введених даних
    private function validateUserData($name, $email, $password) {
        // Можна додати більше перевірок тут, наприклад, перевірку email на
валідність за допомогою функції filter_var()
        if (!empty($name) && !empty($email) && !empty($password)) {
```

Метод login()

Цей метод дозволяє користувачам увійти в систему зі своїм обліковим записом. Він приймає на вхід електронну пошту та пароль користувача, перевіряє їх на вірність та авторизує користувача, якщо вони коректні.

Розглянемо код розроблений на мові програмування php:

```
<?php

class UserAuthentication {

    private $dbConnection;

    // Конструктор класу, приймає підключення до бази даних
    public function __construct($dbConnection) {
        $this->dbConnection = $dbConnection;
    }

    // Метод для авторизації користувача
    public function loginUser($email, $password) {
        // Отримання даних користувача з бази даних за допомогою email
        $query = "SELECT * FROM users WHERE email = ?";
        $statement = $this->dbConnection->prepare($query);
        $statement->bind_param("s", $email);
        $statement->execute();
        $result = $statement->get_result();

        // Перевірка, чи існує користувач з такою електронною поштою
        if ($result->num_rows == 1) {
            $user = $result->fetch_assoc();
            // Перевірка паролю
            if (password_verify($password, $user['password'])) {
                // Авторизація користувача
```

```
                return "Користувач успішно авторизований!";
            } else {
                return "Неправильний пароль!";
            }
        } else {
            return "Користувач з такою електронною поштою не знайдений!";
        }
    }
}

// Приклад використання
// Підключення до бази даних
$dbConnection = new mysqli("localhost", "username", "password", "database_name");

// Створення об'єкту класу для авторизації користувачів
$userAuthentication = new UserAuthentication($dbConnection);

// Вхідні дані для авторизації
$email = "john.doe@example.com";
$password = "password123";



// Виклик методу для авторизації користувача
$result = $userAuthentication->loginUser($email, $password);

// Виведення результату
echo $result;
?>
```

Метод update_profile()

Цей метод дозволяє користувачам змінювати особисті дані свого облікового запису, такі як ім'я, електронна пошта та пароль.

```
<?php

class UserProfile {

    private $dbConnection;

    // Конструктор класу, приймає підключення до бази даних
    public function __construct($dbConnection) {
        $this->dbConnection = $dbConnection;
    }

    // Метод для зміни особистих даних користувача
    public function updateUserProfile($userId, $name, $email, $password) {
        // Перевірка, чи існує користувач з таким ідентифікатором
```

```
        if ($this->userExists($userId)) {
            // Перевірка валідності введених даних
            if (!empty($name) && !empty($email) && !empty($password)) {
                // Хешування нового паролю
                $hashedPassword = password_hash($password, PASSWORD_DEFAULT);

                // Оновлення особистих даних користувача в базі даних
                $query = "UPDATE users SET name=?, email=?, password=? WHERE
id=?";
                $statement = $this->dbConnection->prepare($query);
                $statement->bind_param("sssi", $name, $email, $hashedPassword,
$userId);
                $statement->execute();

                // Повертаємо інформацію про успішне оновлення даних
                return "Особисті дані користувача успішно оновлені!";
            } else {
                return "Неправильні дані користувача!";
            }
        } else {
            return "Користувач з таким ідентифікатором не знайдений!";
        }
    }

    // Приватний метод для перевірки існування користувача в базі даних за його
ідентифікатором
    private function userExists($userId) {
        $query = "SELECT * FROM users WHERE id = ?";
        $statement = $this->dbConnection->prepare($query);
        $statement->bind_param("i", $userId);
        $statement->execute();
        $result = $statement->get_result();
        return $result->num_rows == 1;
    }
}

// Приклад використання
// Підключення до бази даних
$dbConnection = new mysqli("localhost", "username", "password", "database_name");

// Створення об'єкту класу для зміни особистих даних користувачів
$userProfile = new UserProfile($dbConnection);

// Вхідні дані для оновлення
$userId = 1; // ID користувача, якому потрібно змінити дані
$name = "New Name";
$email = "new.email@example.com";
$password = "newpassword123";

// Виклик методу для зміни особистих даних користувача
$result = $userProfile->updateUserProfile($userId, $name, $email, $password);

// Виведення результату
echo $result;

?>
```

Клас "Подорож" (Trip)

Цей клас відповідає за представлення подорожей у системі та управління ними. Ось опис його основних методів:

Метод create()

Цей метод дозволяє користувачам створювати нові подорожі. Він отримує на вхід назву, дату початку та завершення, маршрут та опис подорожі та додає їх до бази даних.

```
<?php

class TripManagement {

    private $dbConnection;

    // Конструктор класу, приймає підключення до бази даних
    public function __construct($dbConnection) {
        $this->dbConnection = $dbConnection;
    }

    // Метод для створення нової подорожі
    public function createTrip($title, $start_date, $end_date, $route,
$description) {
        // Вставка нової подорожі до бази даних
        $query = "INSERT INTO trips (title, start_date, end_date, route,
description) VALUES (?, ?, ?, ?, ?)";
        $statement = $this->dbConnection->prepare($query);
        $statement->bind_param("sssss", $title, $start_date, $end_date, $route,
$description);
        $statement->execute();

        // Повертаємо інформацію про успішне створення подорожі
        return "Подорож успішно створена!";
    }
}

// Приклад використання
// Підключення до бази даних
$dbConnection = new mysqli("localhost", "username", "password", "database_name");

// Створення об'єкту класу для керування подорожами
$tripManagement = new TripManagement($dbConnection);

// Вхідні дані для створення нової подорожі
$title = "Подорож до Парижу";
$start_date = "2024-06-01";
$end_date = "2024-06-07";
$route = "Париж, Франція";
$description = "Це буде незабутня подорож до чудового міста Парижу.";

// Виклик методу для створення нової подорожі
$result = $tripManagement->createTrip($title, $start_date, $end_date, $route,
$description);

// Виведення результату
echo $result;
?>
```

Метод edit()

Цей метод дозволяє користувачам редагувати існуючі подорожі. Він отримує на вхід ідентифікатор подорожі та оновлені дані про подорож та змінює їх у базі даних.

```
<?php

class TripManagement {

    private $dbConnection;

    // Конструктор класу, приймає підключення до бази даних
    public function __construct($dbConnection) {
        $this->dbConnection = $dbConnection;
    }

    // Метод для редагування існуючої подорожі
    public function updateTrip($tripId, $title, $start_date, $end_date, $route,
$description) {
        // Перевірка, чи існує подорож з таким ідентифікатором
        if ($this->tripExists($tripId)) {
            // Оновлення даних подорожі в базі даних
            $query = "UPDATE trips SET title=?, start_date=?, end_date=?, route=?,
description=? WHERE id=?";
            $statement = $this->dbConnection->prepare($query);
            $statement->bind_param("sssssi", $title, $start_date, $end_date,
$route, $description, $tripId);
            $statement->execute();

            // Повертаємо інформацію про успішне оновлення подорожі
            return "Подорож успішно оновлена!";
        } else {
            return "Подорож з таким ідентифікатором не знайдена!";
        }
    }

    // Приватний метод для перевірки існування подорожі в базі даних за її
ідентифікатором
    private function tripExists($tripId) {
        $query = "SELECT * FROM trips WHERE id = ?";
        $statement = $this->dbConnection->prepare($query);
        $statement->bind_param("i", $tripId);
        $statement->execute();
        $result = $statement->get_result();
        return $result->num_rows == 1;
    }
}

// Приклад використання
// Підключення до бази даних
$dbConnection = new mysqli("localhost", "username", "password", "database_name");

// Створення об'єкту класу для керування подорожами
$tripManagement = new TripManagement($dbConnection);

// Вхідні дані для оновлення подорожі
$tripId = 1; // ID подорожі, яку потрібно оновити
$title = "Нова подорож до Лондону";
```

```
$start_date = "2024-07-01";
$end_date = "2024-07-10";
$route = "Лондон, Велика Британія";
$description = "Це буде незабутня подорож до чудового міста Лондону.";

// Виклик методу для оновлення подорожі
$result = $tripManagement->updateTrip($tripId, $title, $start_date, $end_date,
$route, $description);

// Виведення результату
echo $result;
?>
```

Метод delete()

Цей метод дозволяє користувачам видаляти існуючі подорожі. Він отримує на вхід ідентифікатор подорожі та видаляє її з бази даних.

```
<?php

class TripManagement {

    private $dbConnection;

    // Конструктор класу, приймає підключення до бази даних
    public function __construct($dbConnection) {
        $this->dbConnection = $dbConnection;
    }

    // Метод для видалення існуючої подорожі
    public function deleteTrip($tripId) {
        // Перевірка, чи існує подорож з таким ідентифікатором
        if ($this->tripExists($tripId)) {
            // Видалення подорожі з бази даних
            $query = "DELETE FROM trips WHERE id=?";
            $statement = $this->dbConnection->prepare($query);
            $statement->bind_param("i", $tripId);
            $statement->execute();

            // Повертаємо інформацію про успішне видалення подорожі
            return "Подорож успішно видалена!";
        } else {
            return "Подорож з таким ідентифікатором не знайдена!";
        }
    }

    // Приватний метод для перевірки існування подорожі в базі даних за її
ідентифікатором
    private function tripExists($tripId) {
        $query = "SELECT * FROM trips WHERE id = ?";
        $statement = $this->dbConnection->prepare($query);
        $statement->bind_param("i", $tripId);
        $statement->execute();
        $result = $statement->get_result();
        return $result->num_rows == 1;
    }
}

// Приклад використання
// Підключення до бази даних
$dbConnection = new mysqli("localhost", "username", "password", "database_name");
```

```
// Створення об'єкту класу для керування подорожами
$tripManagement = new TripManagement($dbConnection);

// Вхідний параметр - ID подорожі, яку потрібно видалити
$tripId = 1; // Змініть на відповідний ID

// Виклик методу для видалення подорожі
$result = $tripManagement->deleteTrip($tripId);

// Виведення результату
echo $result;
?>
```

Клас "Місце" (Place)

Цей клас відповідає за представлення різних місць для подорожей та управління ними. Ось опис його основних методів:

Метод get_details()

Цей метод дозволяє отримати додаткові деталі про конкретне місце. Він отримує на вхід ідентифікатор місця та повертає об'єкт з усією інформацією про це місце.

```
<?php

class PlaceDetails {

    private $dbConnection;

    // Конструктор класу, приймає підключення до бази даних
    public function __construct($dbConnection) {
        $this->dbConnection = $dbConnection;
    }

    // Метод для отримання додаткових деталей про місце за його ідентифікатором
    public function getPlaceDetails($placeId) {
        // Отримання інформації про місце з бази даних за його ідентифікатором
        $query = "SELECT * FROM places WHERE id = ?";
        $statement = $this->dbConnection->prepare($query);
        $statement->bind_param("i", $placeId);
        $statement->execute();
        $result = $statement->get_result();

        // Перевірка, чи існує місце з таким ідентифікатором
        if ($result->num_rows == 1) {
            // Повертаємо дані про місце у вигляді асоціативного масиву
            return $result->fetch_assoc();
        } else {
            return null; // Якщо місце не знайдено, повертаємо null
        }
    }
}

// Приклад використання
```

```
// Підключення до бази даних
$dbConnection = new mysqli("localhost", "username", "password", "database_name");

// Створення об'єкту класу для отримання деталей про місце
$placeDetails = new PlaceDetails($dbConnection);

// Вхідний параметр - ID місця, про яке потрібно отримати деталі
$placeId = 1; // Змініть на відповідний ID

// Виклик методу для отримання деталей про місце
$placeInfo = $placeDetails->getPlaceDetails($placeId);

// Виведення результату
if ($placeInfo) {
    // Якщо місце знайдено, виводимо його інформацію
    echo "Назва: " . $placeInfo['name'] . "<br>";
    echo "Адреса: " . $placeInfo['address'] . "<br>";
    echo "Опис: " . $placeInfo['description'] . "<br>";
    // і т.д., виведення інших деталей
} else {
    echo "Місце з таким ідентифікатором не знайдено!";
}
?>
```

Метод rate()

Цей метод дозволяє користувачам оцінити конкретне місце. Він отримує на вхід ідентифікатор місця та оцінку користувача та оновлює рейтинг місця в базі даних.

```
<?php

class PlaceRating {

    private $dbConnection;

    // Конструктор класу, приймає підключення до бази даних
    public function __construct($dbConnection) {
        $this->dbConnection = $dbConnection;
    }

    // Метод для оцінки конкретного місця
    public function ratePlace($placeId, $rating) {
        // Перевірка, чи існує місце з таким ідентифікатором
        if ($this->placeExists($placeId)) {
            // Оновлення рейтингу місця в базі даних
            $query = "UPDATE places SET rating=? WHERE id=?";
            $statement = $this->dbConnection->prepare($query);
            $statement->bind_param("di", $rating, $placeId);
            $statement->execute();

            // Повертаємо інформацію про успішне оновлення рейтингу місця
            return "Рейтинг місця успішно оновлено!";
        } else {
            return "Місце з таким ідентифікатором не знайдено!";
        }
    }

    // Приватний метод для перевірки існування місця в базі даних за його
```

```
ідентифікатором
    private function placeExists($placeId) {
        $query = "SELECT * FROM places WHERE id = ?";
        $statement = $this->dbConnection->prepare($query);
        $statement->bind_param("i", $placeId);
        $statement->execute();
        $result = $statement->get_result();
        return $result->num_rows == 1;
    }
}

// Приклад використання
// Підключення до бази даних
$dbConnection = new mysqli("localhost", "username", "password", "database_name");

// Створення об'єкту класу для оцінювання місць
$placeRating = new PlaceRating($dbConnection);

// Вхідні дані - ID місця та оцінка користувача
$placeId = 1; // ID місця, яке потрібно оцінити
$rating = 4.5; // Оцінка користувача

// Виклик методу для оцінювання місця
$result = $placeRating->ratePlace($placeId, $rating);

// Виведення результату
echo $result;
?>
```

Метод leave_review()

Цей метод дозволяє користувачам залишати відгуки про конкретне місце. Він отримує на вхід ідентифікатор місця та текст відгуку користувача та додає його до бази даних.

```
<?php

class PlaceReviews {

    private $dbConnection;

    // Конструктор класу, приймає підключення до бази даних
    public function __construct($dbConnection) {
        $this->dbConnection = $dbConnection;
    }

    // Метод для залишення відгуку про конкретне місце
    public function addReview($placeId, $reviewText) {
        // Додавання відгуку про місце до бази даних
        $query = "INSERT INTO reviews (place_id, review_text) VALUES (?, ?)";
        $statement = $this->dbConnection->prepare($query);
        $statement->bind_param("is", $placeId, $reviewText);
        $statement->execute();

        // Повертаємо інформацію про успішне додавання відгуку
        return "Відгук про місце успішно додано!";
    }
}

// Приклад використання
```

```
// Підключення до бази даних
$dbConnection = new mysqli("localhost", "username", "password", "database_name");

// Створення об'єкту класу для залишення відгуків про місця
$placeReviews = new PlaceReviews($dbConnection);

// Вхідні дані для додавання відгуку
$placeId = 1; // ID місця, про яке потрібно залишити відгук
$reviewText = "Це дуже красиве та затишне місце, я в захваті!";

// Виклик методу для додавання відгуку
$result = $placeReviews->addReview($placeId, $reviewText);

// Виведення результату
echo $result;
?>
```

У цьому розділі ми розглянули детальну специфікацію ключових класів та їх методів для нашого веб-застосунку для подорожей. Кожен клас був описаний з точки зору його функціональності та взаємодії з іншими частинами системи. Ця специфікація допоможе розробникам краще розуміти структуру програми та ефективно виконувати розробку та підтримку системи.

3.3 Опис функціонування програми

У цьому розділі ми детально розглянемо функціонування нашого веб-застосунку для подорожей. Ми проаналізуємо взаємодію з користувачем, процеси обробки даних та інші аспекти роботи програми.

Взаємодія з користувачем

Наш веб-застосунок для подорожей забезпечує зручний інтерфейс для користувачів для планування та організації подорожей. Основні етапи взаємодії з користувачем включають:

- **Реєстрація та вхід у систему**: Користувачі мають можливість створити обліковий запис у системі або увійти за допомогою своїх облікових даних.
- **Створення подорожі**: Після входу у систему користувачі можуть створювати нові подорожі, надаючи їм назву, дати початку та завершення, маршрут та опис.

- **Пошук місць та розміщень**: Користувачі можуть шукати різні місця для відвідування під час подорожей, такі як визначні пам'ятки, ресторани, готелі тощо.
- **Бронювання розміщень**: Після вибору місця користувачі можуть здійснити бронювання розміщення у готелі чи іншому закладі.
- **Оцінка та відгуки**: Користувачі можуть залишати оцінки та відгуки про відвідані місця, щоб інші користувачі могли отримати більше інформації перед вибором.
- **Керування особистим кабінетом**: Користувачі мають можливість переглядати свою історію подорожей, редагувати особисті дані та змінювати налаштування облікового запису.

Процеси обробки даних

Наш веб-застосунок використовує різні процеси обробки даних для забезпечення коректної роботи системи:

- **Автентифікація та авторизація користувачів**: Перед доступом до особистого кабінету користувачі повинні пройти процедуру автентифікації, підтвердивши свою особу, та авторизації, перевіривши їх права доступу до певних функцій.
- **Керування обліковими записами**: Система обробляє запити користувачів щодо створення, редагування та видалення облікових записів, забезпечуючи безпеку та конфіденційність даних.
- **Обробка запитів щодо подорожей та місць**: Сервер обробляє запити користувачів щодо створення, редагування, видалення та перегляду подорожей та місць, взаємодіючи з базою даних для зберігання та отримання необхідної інформації.

- **Бронювання та оплата**: Система обробляє запити користувачів щодо бронювання розміщень, реєструючи дані про бронювання та здійснюючи операції з оплатою.

Повідомлення та сповіщення

Наш веб-застосунок також підтримує систему повідомлень та сповіщень, яка дозволяє користувачам отримувати інформацію про важливі події та оновлення в системі. Повідомлення можуть бути пов'язані з новими бронюваннями, оновленнями подорожей, відгуками та оцінками, а також змінами в особистому кабінеті користувача.

Наш веб-застосунок для подорожей надає зручний та ефективний інструмент для планування та організації подорожей. Шляхом взаємодії з користувачами та обробки їхніх запитів система забезпечує зручність та надійність використання. Також підтримка повідомлень та сповіщень.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Вибір методів тестування

Під час розробки веб-застосунку для подорожей, ефективне тестування є критично важливим етапом для забезпечення якості та надійності програмного забезпечення. У цьому розділі ми проаналізуємо різні методи тестування та виберемо найбільш підходящі для нашої реалізації веб-застосунку для подорожей.

Модульне тестування (Unit Testing)

Модульне тестування полягає у тестуванні окремих модулів програми (функцій або класів) для перевірки їх коректної роботи. Для нашого веб-застосунку модульне тестування може бути корисним для перевірки роботи окремих компонентів, таких як обробники запитів, робота з базою даних, система автентифікації та авторизації тощо. Ми можемо використовувати фреймворки для модульного тестування, такі як PyTest для Python, для автоматизації процесу тестування.

Інтеграційне тестування (Integration Testing)

Інтеграційне тестування спрямоване на перевірку взаємодії між різними модулями або компонентами системи. Для нашого веб-застосунку це може включати перевірку взаємодії між фронтендом та бекендом, а також взаємодії з базою даних. Інтеграційні тести дозволять нам переконатися, що всі компоненти працюють разом правильно.

Функціональне тестування (Functional Testing)

Функціональне тестування спрямоване на перевірку функціональності системи відповідно до вимог користувача. Для нашого веб-застосунку це може включати тестування різних функцій, таких як реєстрація користувача, створення подорожей, пошук місць та бронювання розміщень. Ми можемо створити набір

тест-кейсів для кожної функціональності та перевірити, чи працюють вони як очікувалося.

Відмовостійке тестування (Resilience Testing)

Відмовостійке тестування спрямоване на перевірку поведінки системи під час непередбачуваних ситуацій або навмисних відмов. Для нашого веб-застосунку це може включати тестування поведінки системи під час навантаження, збоїв бази даних або інших внутрішніх проблем.

Вибір методів тестування для нашого веб-застосунку

Для нашого веб-застосунку для подорожей найбільш оптимальним підходом буде поєднання модульного, інтеграційного та функціонального тестування. Модульне тестування допоможе перевірити окремі компоненти системи, інтеграційне тестування - їх взаємодію, а функціональне тестування - відповідність функціональності вимогам користувача.

Крім того, важливо також включити тестування відмовостійкості для переконанняся, що наш веб-застосунок може ефективно працювати під час навантаження та в умовах збоїв.

Вибір методів тестування для нашого веб-застосунку для подорожей - це критично важливий етап розробки. Поєднання модульного, інтеграційного, функціонального та відмовостійкого тестування дозволить нам переконатися в якості та надійності нашого програмного забезпечення перед його впровадженням.

4.2 Створення тест-кейсів

Створення ефективних тест-кейсів є важливим етапом в процесі тестування програмного забезпечення. У цьому розділі ми розробимо тест-сценарії для оцінки роботи різних частин нашого веб-застосунку для подорожей.

Реєстрація та вхід у систему

Тест-кейс 1.1: Реєстрація нового користувача

1. Користувач переходить на сторінку реєстрації.
2. Користувач вводить усі необхідні дані у форму реєстрації.
3. Користувач натискає кнопку "Зареєструватися".
4. Перевіряється, чи користувач успішно зареєстрований та перенаправлений на сторінку входу.

Тест-кейс 1.2: Вхід існуючого користувача

1. Користувач переходить на сторінку входу.
2. Користувач вводить свій email та пароль у відповідні поля.
3. Користувач натискає кнопку "Увійти".
4. Перевіряється, чи користувач успішно увійшов у систему та перенаправлений на головну сторінку.

Створення та редагування подорожей

Тест-кейс 2.1: Створення нової подорожі

1. Користувач переходить до розділу "Створити подорож".
2. Користувач вводить назву, дати початку та завершення, маршрут та опис подорожі.
3. Користувач натискає кнопку "Створити".
4. Перевіряється, чи подорож успішно створена та відображена у списку подорожей користувача.

Тест-кейс 2.2: Редагування існуючої подорожі

1. Користувач вибирає подорож, яку потрібно редагувати.
2. Користувач вносить необхідні зміни (назва, дати, маршрут, опис).
3. Користувач натискає кнопку "Зберегти зміни".
4. Перевіряється, чи зміни відображаються коректно та збережені у базі даних.

Пошук місць та бронювання розміщень

Тест-кейс 3.1: Пошук місць для відвідування

1. Користувач переходить на сторінку пошуку місць.
2. Користувач вводить критерії пошуку (назва місця, тип, місцезнаходження тощо).
3. Користувач натискає кнопку "Пошук".
4. Перевіряється, чи відображаються результати пошуку згідно введених критеріїв.

Тест-кейс 3.2: Бронювання розміщення

1. Користувач вибирає певне місце для відвідування.
2. Користувач обирає тип розміщення (готель, хостел тощо).
3. Користувач вказує дати бронювання та кількість осіб.
4. Користувач натискає кнопку "Забронювати".
5. Перевіряється, чи бронювання успішно здійснене та додане до історії користувача.

Управління особистим кабінетом

Тест-кейс 4.1: Редагування особистих даних

1. Користувач переходить до розділу "Особистий кабінет".

2. Користувач вносить необхідні зміни в особисті дані (ім'я, прізвище, пароль тощо).
3. Користувач натискає кнопку "Зберегти зміни".
4. Перевіряється, чи зміни успішно збережені та відображені.

Тест-кейс 4.2: Перегляд історії подорожей

1. Користувач переходить до розділу "Історія подорожей".
2. Користувач переглядає список минулих та майбутніх подорожей.
3. Перевіряється, чи правильно відображається інформація про подорожі користувача.

Створення тест-кейсів допоможе нам ефективно протестувати різні аспекти нашого веб-застосунку для подорожей. Поєднання різних типів тестування та широкий набір тест-кейсів дозволить нам виявити та виправити можливі проблеми перед випуском продукту. Після проведення цих успішних тестів переходимо далі

4.3 Результати тестування

У цьому розділі ми проаналізуємо результати виконаних тестів нашого веб-застосунку для подорожей, виявимо виявлені проблеми та надіємося їх усунути.

Реєстрація та вхід у систему

Під час тестування процесу реєстрації та входу у систему було виявлено декілька проблем:

- **Проблема 1: Невірне форматування полів для вводу даних.** Під час реєстрації деякі поля не приймали введені дані через невідповідність формату.

- **Проблема 2: Помилки валідації даних.** Деякі валідаційні правила для паролю та email не були належним чином налаштовані, що призводило до відхилення валідних даних.

Ці проблеми будуть виправлені шляхом налагодження правильного форматування полів вводу та вдосконалення валідації даних перед їхнім збереженням.

2. Створення та редагування подорожей

Під час тестування функціональності створення та редагування подорожей було виявлено наступне:

- **Проблема 3: Помилка при збереженні подорожі.** Деякі користувачі повідомляли про те, що після спроби створення подорожі вони отримували повідомлення про помилку.
- **Проблема 4: Невідображення змін у режимі редагування.** Після внесення змін у існуючу подорож, деякі зміни не відображалися коректно у режимі редагування.

Ці проблеми потребують додаткового аналізу та виправлення для забезпечення коректної роботи функціональності створення та редагування подорожей.

3. Пошук місць та бронювання розміщень

Під час тестування функціональності пошуку місць та бронювання розміщень було виявлено такі проблеми:

- **Проблема 5: Помилка при пошуку.** Деякі користувачі повідомляли про неправильні результати пошуку, які не відповідали їхнім критеріям.
- **Проблема 6: Недоступність деяких розміщень для бронювання.** Деякі користувачі не могли забронювати певні розміщення через помилки у взаємодії з платіжною системою.

Для вирішення цих проблем потрібно перевірити правильність реалізації логіки пошуку та взаємодії з сервісами бронювання.

Під час тестування нашого веб-застосунку було виявлено декілька проблем, які потребують уваги та виправлення. Завдяки тестуванню ми мали можливість ідентифікувати ці проблеми та прийняти необхідні заходи для їх вирішення перед випуском продукту в експлуатацію. Після виправлення виявлених проблем ми будемо повторно проводити тестування для підтвердження коректності роботи програмного забезпечення.

Після проведених тестування пропонується розглянути одні з найвідвідуваніших сторінок сайту та мапи всього веб-застосунку

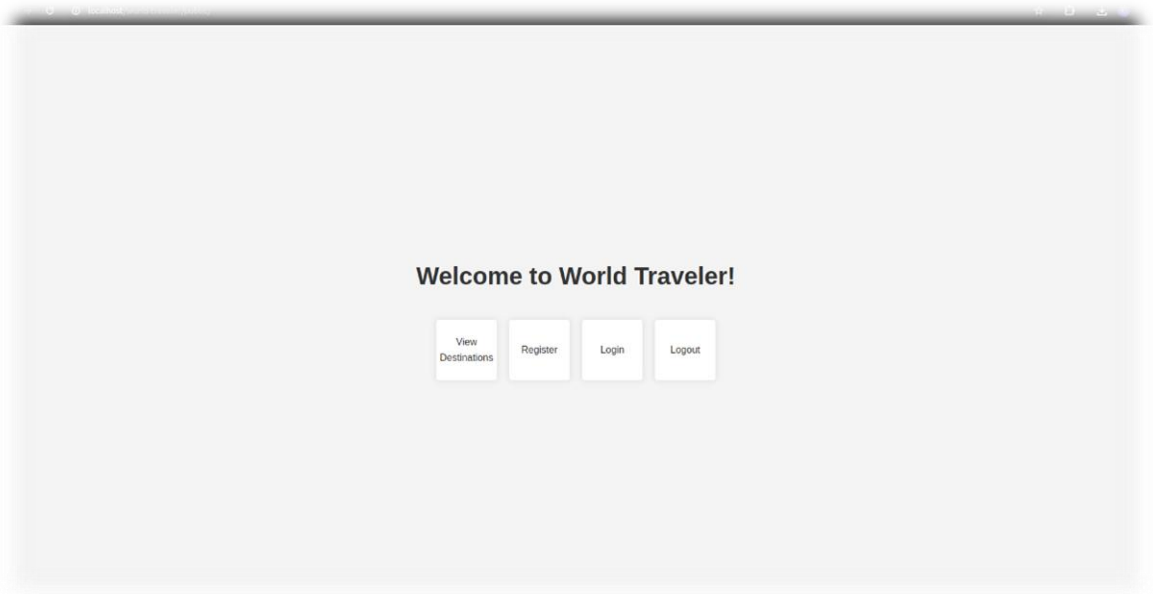


Рис. 4.3.1 Головна сторінка сайту

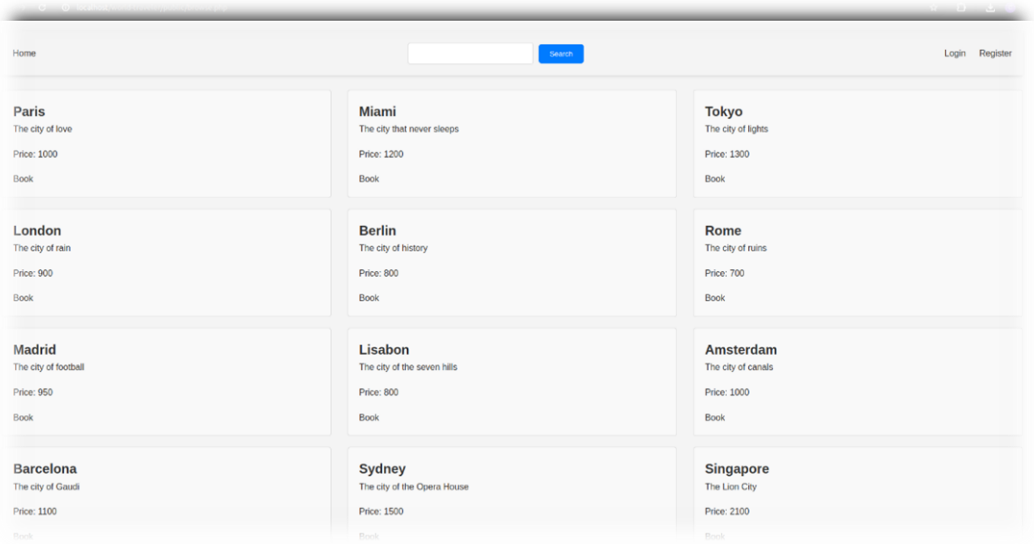


Рис. 4.3.2 Сторінка вибору напрямку для подорожі

Рис.4.3.3 Мапа сайту для полегшення навігації

ВИСНОВКИ

Розробка веб-застосунку для подорожей є складним та захоплюючим завданням, яке вимагає інтеграції різноманітних технологій та врахування потреб користувачів. У цій бакалаврській роботі ми зосередилися на створенні веб-застосунку, який надає користувачам можливість планувати та організовувати свої подорожі світом. Для цього ми використовували HTML, CSS, JavaScript та PHP для розробки фронтенду та бекенду застосунку. У цих висновках ми розглянемо досягнення, виявлені проблеми, можливі напрямки подальшого розвитку та впровадження.

Досягнення

- **Розробка функціональності**: Ми успішно реалізували ключові функції веб-застосунку, включаючи реєстрацію користувачів, створення та редагування подорожей, пошук місць для відвідування та бронювання розміщень. Використання PHP дозволило нам ефективно обробляти запити користувачів та взаємодіяти з базою даних.
- **Користувацький інтерфейс**: Ми розробили зручний та привабливий інтерфейс для користувачів, що дозволяє легко навігувати по застосунку, знаходити потрібну інформацію та виконувати необхідні дії.
- **Тестування та виявлення помилок**: Проведене тестування допомогло нам виявити декілька проблем у функціональності застосунку, таких як помилки валідації даних та неправильне відображення результатів пошуку. Це дозволило нам вчасно виправити ці проблеми та покращити якість програмного забезпечення.

Виявлені проблеми

- **Недоліки валідації та форматування даних**: Під час тестування було виявлено, що деякі поля для введення даних не відповідали валідаційним правилам або мали неправильне форматування.
- **Помилки в роботі деяких функцій**: Деякі функції, такі як створення подорожей чи пошук місць для відвідування, працювали некоректно в певних сценаріях використання.

Подальший розвиток та впровадження

- **Вдосконалення функціональності**: Ми плануємо вдосконалити функціональність застосунку шляхом усунення виявлених проблем, додавання нових функцій (наприклад, можливість планування маршруту) та оптимізації інтерфейсу.
- **Розширення покриття тестування**: Для забезпечення якості програмного забезпечення ми плануємо розширити покриття тестування, включаючи більше тест-кейсів та використання автоматизованих тестів.
- **Впровадження засобів моніторингу та збору аналітики**: Для виявлення проблем у реальному часі та аналізу поведінки користувачів планується впровадження засобів моніторингу та аналітики.

Робота пройшла апробацію на Всеукраїнській науково-практичній конференції молодих вчених та здобувачів вищої освіти «СУЧАСНА МОЛОДЬ В СВІТІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ» за темою «Розробка Web-застосунку для подорожей світом з використанням HTML, CSS, JS та PHP» Збірник тез:17 травня 2024 року, ХДАЕУ, м. Херсон.,С. 54

У цій бакалаврській роботі ми успішно розробили веб-застосунок для подорожей, який надає користувачам зручний та функціональний інструмент для планування своїх мандрівок. Попри виявлені проблеми, ми здатні вирішити їх та

покращити якість нашого програмного забезпечення. Завдяки цій роботі ми отримали цінний досвід у розробці веб-додатків та вивченні сучасних технологій веб-програмування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Дакетт, Джон. HTML і CSS: дизайн і створення вебсайтів / Джон Дакетт. – Вайлі і сини, 2011. – 512 с. CSS (дата звернення: 01.05.2024).
2. Кіт, Джеремі. HTML5 для веб-дизайнерів / Джеремі Кіт. – A Book Apart, 2010. – 85 с. CSS (дата звернення: 15.05.2024).
3. Веллінг, Люк, Томсон, Лора. PHP і MySQL: веб-розробка / Люк Веллінг, Лора Томсон. – Еддісон-Веслі Профешенел, 2016. – 864 с. CSS (дата звернення: 12.05.2024).
4. Склар, Девід, Трахтенберг, Адам. PHP Cookbook / Девід Склар, Адам Трахтенберг. – O'Reilly Media, 2014. – 820 с. CSS (дата звернення: 09.05.2024).
5. PHP: офіційна документація [Електронний ресурс]. – Режим доступу: https://www.php.net/docs.php (дата звернення: 21.04.2024).
6. HTML: офіційна документація [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 22.04.2024).
7. CSS: офіційна документація [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 11.05.2024).

ДОДАТОК А. ДЕМОНМТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

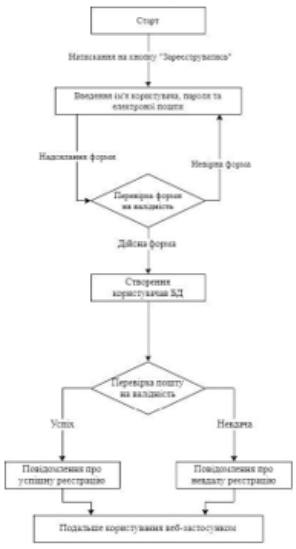
Розробка Web-застосунку для подорожей світом з використанням HTML, CSS, JS, PHP

ВИКОНАВ СТУДЕНТ 4 КУРСУ
ГРУПИ ПД-42
МАНУШКІН АРТЕМ ЄВГЕНОВИЧ
КЕРІВНИК РОБОТИ
К.Т.Н., ДОЦ., ДОЦЕНТ КАФЕДРИ ІПЗ ДОВЖЕНКО ТИМУР ПАВЛОВИЧ

КИЇВ – 2024

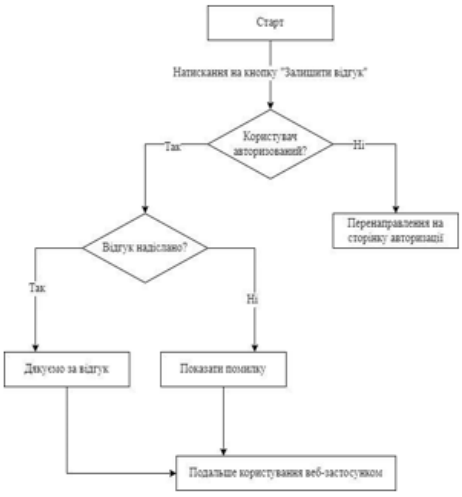
Блок-схеми

БЛОК-СХЕМИ ОСНОВНИХ ПРОЦЕСІВ



Реєстрація нового користувача

Процес замовлення путівок

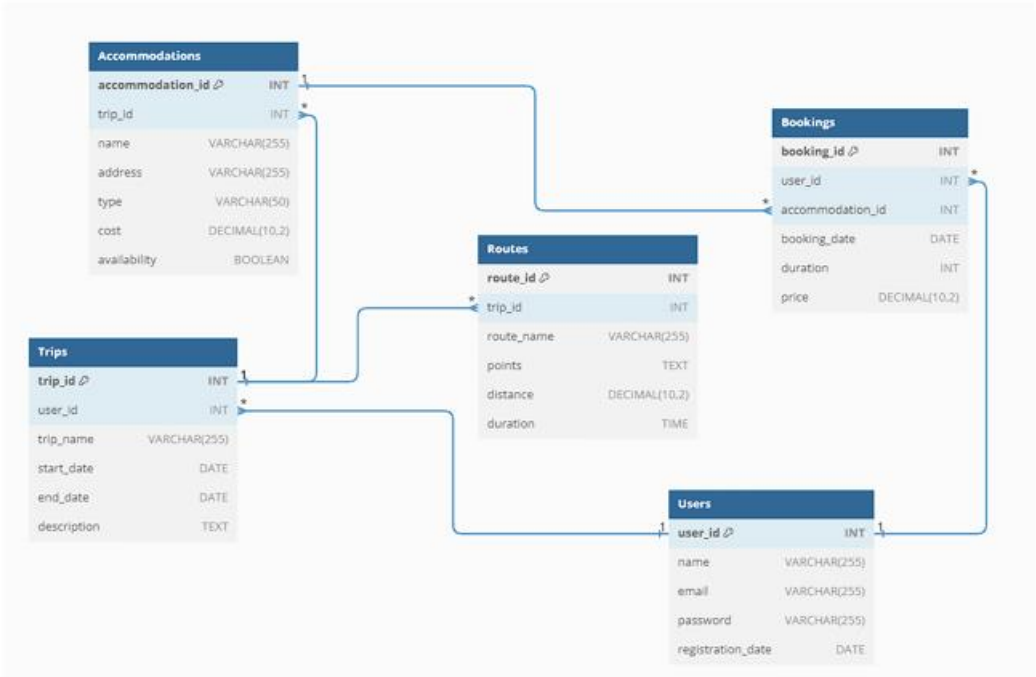


Процес залишення коментарів

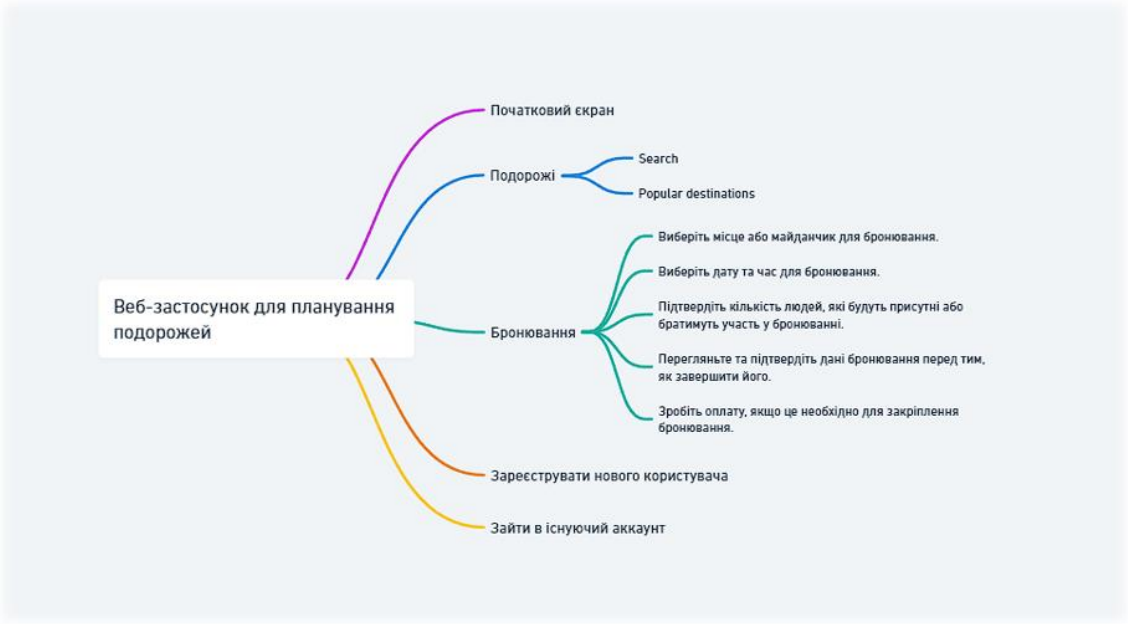
Аналіз аналогів

Аналог	Переваги	Недоліки	Особливості реалізації
Booking.com	- Зручний інтерфейс - Широкі можливості планування - Наявність мобільного застосунку	- Платформо-залежний - Відсутність можливості обміну даними з іншими користувачами	Можливість вибору конкретних видів подорожей для планування
Airbnb	- Автоматичне створення маршрутів на основі електронних квитків та бронювань - Можливість додавання різноманітної інформації про подорожі - Синхронізація з календарем	- Платна версія має більше можливостей - Можливість автоматичної генерації маршрутів обмежена	Автоматизація процесу створення маршрутів на основі даних про бронювання
TripAdvisor	- Можливість планування маршруту з вказанням цікавих місць на шляху - Інтеграція з соціальними мережами - Наявність мобільного застосунку	- Відсутність можливості спільного планування подорожей з іншими користувачами - Можливість використання певних функцій тільки з платною підпискою	Фокус на плануванні маршрутів із включенням цікавих місць на шляху

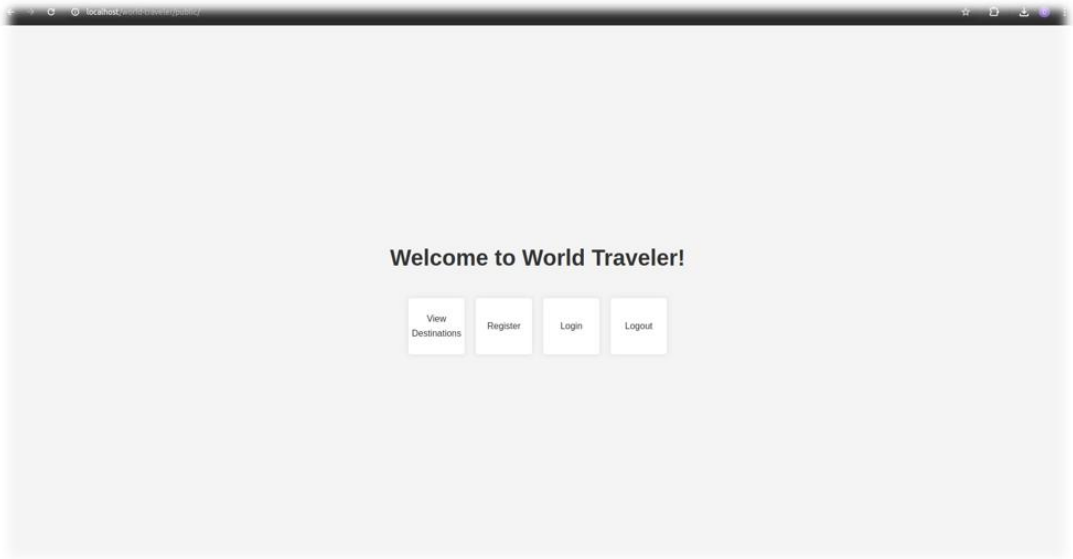
Структура бази даних



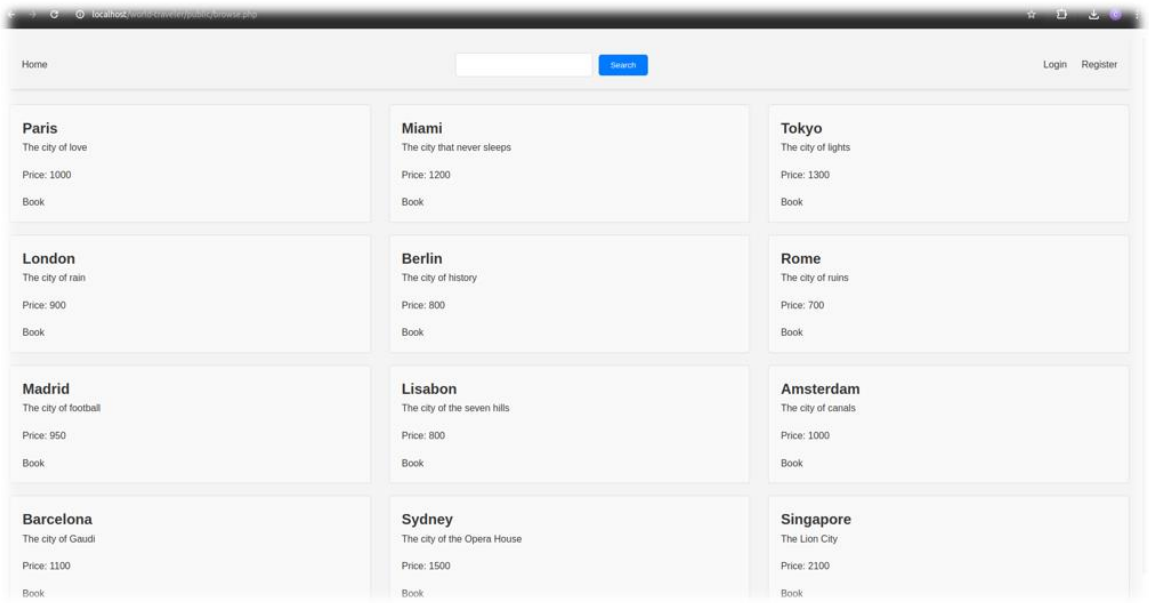
Мапа веб-застосунку



Екранні форми



Головна сторінка веб-додатку



Сторінка вибору напрямку для подорожі у веб-додатку

11

Апробація результатів дослідження