Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: **«Розробка веб-застосунку підбору маршрутів для мандрівок з використанням технології React»**

Виконав студент 4-го року навчання

спеціальності 122

«Комп’ютерні науки»

Онопрійчук Артур Едуардович

Керівник: Борозенний С. О.

кандидат _____

Рецензент _____

(прізвище та ініціали)

Кваліфікаційна робота захищена

з оцінкою _____

Секретар ЕК_____

(підпис)

«_____»_____2023р.

Київ - 2023

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапу роботи	Термін виконання	Примітка
1.	Отримання завдання на роботу.	29.06.2022	
2.	Огляд літератури за темою роботи.	липень - серпень 2022	
3.	Огляд застосунків та системи, що мають суміжний функціонал, вибір технології та інструментів для розробки.	вересень 2022	
4.	Розробка практичної частини.	жовтень - грудень 2022	
5.	Написання текстової частини.	січень - березень 2023	
6.	Корегування роботи за результатами перевірки керівником.	квітень 2023	
7.	Створення презентації для доповіді.	травень 2023	
8.	Подання роботи на кафедру для перевірки на плагіат.	22.05.2023	
9.	Захист кваліфікаційної роботи.	30.05.2023	

Студент Онопрійчук Артур Едуардович

Керівник Борозенний С. О.

"_____"_____

Вступ

У зв’язку зі зростаючою популярністю міжнародних подорожей та актуальністю пошуку локацій для мандрівок, за мету була поставлена розробка алгоритму для пошуку складних маршрутів подорожей з урахуванням вподобань користувача та створення веб-сайту з використанням відповідного алгоритму.

Наш веб-сайт буде мати простий та інтуїтивно зрозумілий інтерфейс, без зайвої сторонньої інформації, що дозволить користувачам легко орієнтуватися по застосунку та вводити параметри пошуку, такі як країни, бюджет мандрівки, сезон, тип туризму, популярність серед інших туристів та кількість бажаних зупинок у маршруті. Крім того, сайт буде містити детальну інформацію про кожне місто та країну на маршруті, що допоможе користувачам зробити вибір.

Застосунок буде пропонувати користувачам різноманітні маршрути, що включають в себе як відомі туристичні місця, так і менш відомі природні та культурно пам'ятні місця. Таким чином, користувачі зможуть знайти унікальні місця для відвідування та насолодитися незабутнім досвідом подорожі.

Аудиторія застосунку є всі користувачі Інтернет зацікавленні у пошуку місць для подорожі, що відповідають їхнім потребам та вподобанням.

Для реалізації застосунку, потрібно створити алгоритм, який буде вираховувати та обробляти дані. Вхідними даними для алгоритму є фільтраційні дані вподобань користувача, отримані від його безпосереднього вводу у форму на веб-сайті. Вихідними даними є перелік маршрутів релевантних під запит користувача. Маршрути, які отримує користувач як кінцевий результат, складаються з переліку міст, додаткового опису та корисних посилань, які як найкраще підходять під визначений пошук.

Об’єктами досліджень є:

- Статистика туристичних подорожей з відкритих джерел, статей та документацій;
- Застосунки, що мають суміжний функціонал до розроблюваної системи;
- Технології та інструменти для розробки програмного продукту.

Програмний продукт можна поділити на такі логічні складові:

- Серверна частина;
- Клієнтська частина;
- База даних.

Під час розробки використано багато різних бібліотек та додаткових фреймворків. Алгоритм, як і вся серверна частина застосунку написана на мові JavaScript. Варто виділити JavaScript-бібліотеку, призначену для створення інтерфейсів користувача React, що займає провідну роль та є основним інструментом у розробці клієнтської частини. У ролі бази даних використано документо-орієнтовану базу даних MongoDB.

Також, було використано бібліотеку Express для створення сервера застосунку, Mongoose для забезпечення простої та зручної взаємодії з базою даних, Axios для виконання запитів з клієнтської частини на серверну частину застосунку, а також бібліотеки TailwindCSS та DaisyUI для швидкого та гармонійного застосування стилів та стандартизації вигляду веб-сайту. Застосунок було розроблено з використанням архітектурного підходу «MVC» (Model-View-Controller), що дозволило зручно розділити логіку та відображення даних на клієнті та сервері.

Кваліфікаційна робота складається зі вступу, трьох основних розділів, висновка та списку використаних джерел.

1. Аналіз предметної області. Постановка завдання кваліфікаційної роботи

1.1. Аналіз сучасного стану питання та обґрунтування теми

Туризм є невід'ємною частиною нашого світу. Мільйони людей подорожують по світу кожен рік. Завдяки всесвітньої туристичної організації ООН можемо побачити зростаючу популярність саме туристичних подорожей [1]. Графік можна побачити на рисунку 1.1.

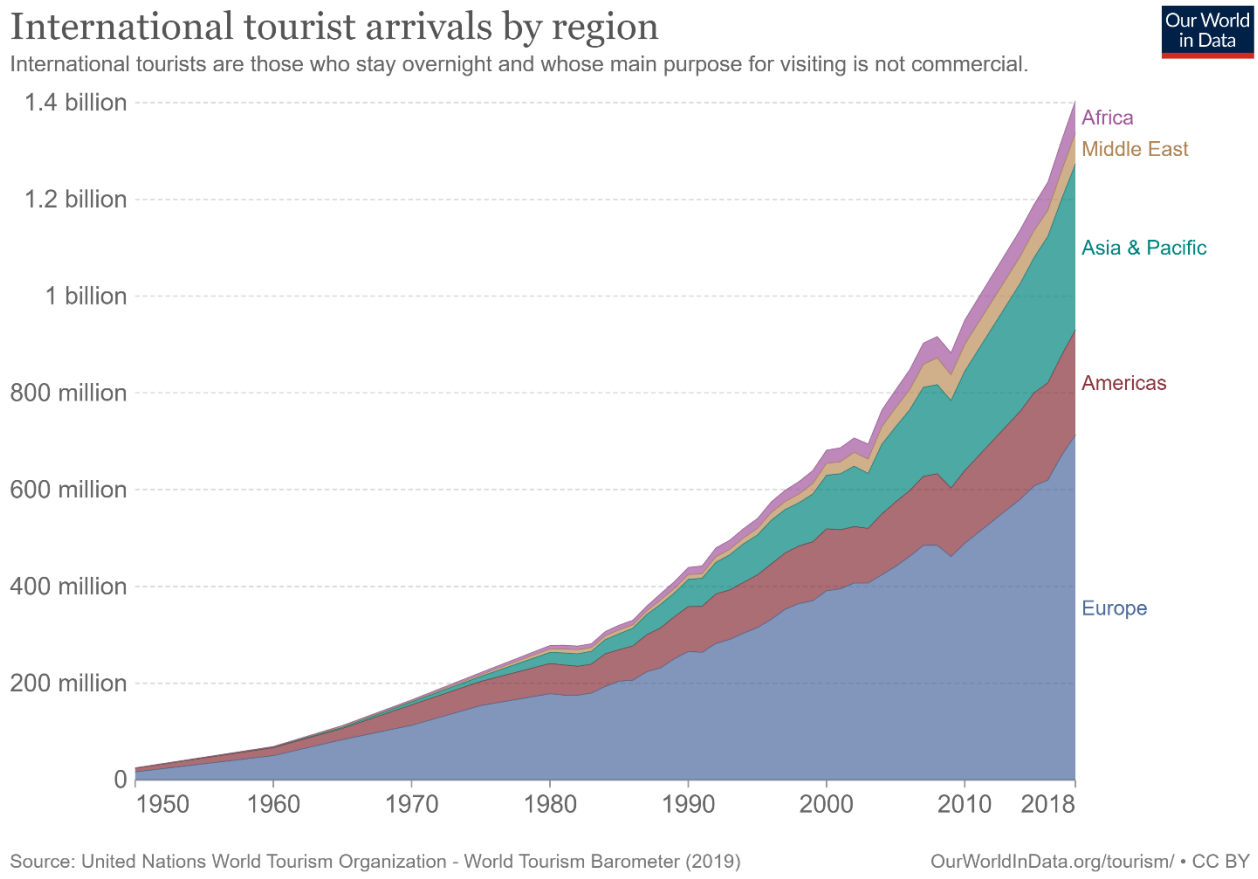


Рисунок 1.1 – Статистика туристичних подорожей

У 1950 загальна кількість міжнародних подорожей складала лише 25 мільйонів. Бачимо у 80-х роках 20 сторіччя початок різкого росту. Тільки за 2018 рік маємо 1 мільярд 400 мільйонів туристичних поїздок. Популярність подорожей до Африки, Близького Сходу, Азії та Океанії зростає з декількох сотень тисяч до десятків та сотень мільйонів за 70 років. Тож, бачимо

тенденції на ще більшу «близькість світу» та інтерес у людей подорожувати на всі континенти.

Всесвітня туристична організація ООН, також, надає деяку інформацію щодо зниження статистики туризму через COVID-19 на протязі 2020-2021 років з поступовим поверненням показників на до епідемічний рівень [2].

З такої важливої статистики бачимо, що людство за останні 70 років створило можливості для міжнародного на міжконтинентальних подорожей. Туризм, який до цього був можливий тільки для невеликої групи людей, зараз доступний для сотень мільйонів людей. Зрозуміло, що попит на туристичні подорожі є величезним та продовжує зростати.

Зазвичай, туристи мають ідею, який вид туризму вони хочуть, бюджет поїздки, сезон тощо. Але не всі знають, як підібрати країну або місто, щоб їх побажання та можливості перетнулися. Зручний та функціональний сайт пошуку найліпших та найоптимальніших туристичних маршрутів дозволяє допомогти користувачу знайти саме те, що підходить йому, з корисною інформацією та посиланнями.

Такий веб-застосунок підбору маршрутів для мандрівок повинен мати:

- Зручний та сучасний інтерфейс;
- Пошук маршруту мандрівки;
- Контакти при виникненні проблем у користуванні;
- Інформація про країну та місто через який проходить маршрут;
- Посилання на мапу маршруту та наявні місця у готелі.

1.2.Огляд існуючих аналогів розробки

Аналогів з подібною концепцією існує небагато, але, все ж таки, один сервіс виділяється і є доволі популярним. «Where and When» - веб-сайт, який має дуже багато сервісів та інформації пов’язаної з туристичними

подорожами [3]. За словами розробників: «Сайт, який дає вам знайти, куди і коли поїхати у відпустку: знайдіть погоду, адаптовану до активності, якою ви хочете зайнятися, і вашого бюджету, залежно від того, який місяць ви виберете для подорожі». Інтерфейс даного веб-сайту можна побачити на рисунку 1.2.

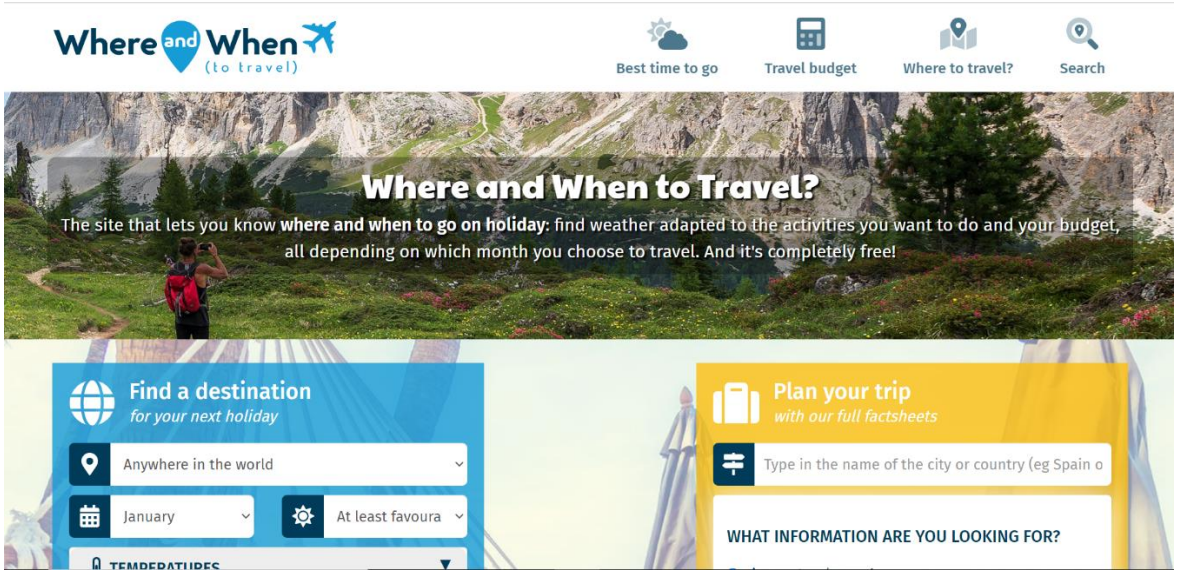


Рисунок 1.2 – Веб-сайт «Where and When»

Деякі сервіси цього сайту:

- Підбір локації для подорожі за користувацькими фільтрами;
- Інформація про погодні умови для певного міста або країни;
- Інформація щодо орієнтовного бюджету туристичного перебування у певному місті або країні.

Той факт, що кожен місяць цей сайт відвідує близько пів мільйона користувачів говорить про те, що попит на подібний сервіс існує [4]. Потенційним туристам цікаво підібрати собі країну або місто для подорожі за фільтрами місяць, ціна, сезон, активність тощо. А також, користувачі

цікавляться окремо погодними умовами та бюджетом поїздки для певної локації.

«Where and When» є веб-порталом з великою базою даних майже про всі міста, країни, регіони та континенти. Також, даний сайт має авторські статті та описи цих локацій. «Where and When» фокусований більше на погодній статистиці та інформації. Він дозволяє зрозуміти коли краще за всього та куди поїхати, в принципі що випливає з його назви.

Але треба зазначити, що даний сайт та розроблюваний веб-застосунок підбору маршрутів для мандрівок виконують трохи різну функцію. Розробка є більш вузько направленою та виконує функцію підбору маршруту мандрівок, який складається з декількох локацій. Хоча за допомогою розробки можна шукати одну локацію, вибравши «кількість зупинок - 1», головна ідея саме комбінувати різні міста та країни для того, щоб створити унікальний, оптимальний та цікавий маршрут подорожі серед тих локацій, які підходять користувачу після фільтрації.

Плюси та мінуси даного сайту:

З плюсів можна виділити:

- Сучасний та інтуїтивно зрозумілий дизайн;
- Велика база даних міст, країн, регіонів та континентів;
- Цікаві авторські статті;
- Широкий функціонал.

З мінусів можна виділити:

- Відсутність можливості підбору маршрутів з декількох локацій;
- Відсутність вибору країни при фільтрації;
- Відсутність фільтрації по популярності локації.

1.3.Постановка задачі

У результаті аналізу предметної області та порівнянні з аналогом розроблюваного веб-застосування сформовано перелік вимог до створюваної системи.

1. Для функціонування веб-застосунок повинен мати клієнтську частину, серверну частину та базу даних;
2. Застосунок повинен забезпечувати користувачам підбір маршрутів мандрівок за такими характеристиками:
   - Вибір країни або декількох країн або будь-якої країни за замовчуванням;
   - Вибір бюджету на один день у доларах США або будь-якого бюджету за замовчуванням (від 0 до 500 доларів США на день);
   - Вибір сезону (зима, весна, літо, осінь) або декількох сезонів або будь-якого сезону за замовчуванням;
   - Вибір типу туризму (активний-пригодницький, культурний, пляжний, на природі) або декількох типів туризму або будь-якого типу туризму за замовчуванням;
   - Вибір рейтингу популярності локації (1, 2, 3, 4, 5) або декількох рейтингів популярності локації або будь-якого рейтингу популярності локації за замовчуванням.
   - Вибір кількості зупинок (1, 2, 3, 4, 5, 6, 7) або дві зупинки за замовчуванням
3. Застосунок повинен включати в себе алгоритм пошуку найліпшого та найоптимальнішого маршруту мандрівки за аналізом характеристик подорожі, який ввів користувач;
4. Застосунок повинен мати додаткову інформацію про кожну країну та місто у вигляді короткого опису локації;

5. Кожен маршрут мандрівки повинен мати посилання на мапу з маршрутом на сервісі «Google Maps» [5], а також, посилання на доступні готелі та апартаменти у вибраній локації на сервісі «Booking.com» [6];
6. Застосунок повинен мати коротку інформації саме що його роботи та посилання на зворотній зв’язок;
7. Нефункціональні вимоги до застосунку:
    - Сучасний та мінімалістичний дизайн;
    - Інтуїтивно зрозумілий інтерфейс;
    - Адаптація сторінок під різні розширення екрану;
    - Кольорова гамма, що слідує новим тенденціям та виділяє важливі елементи.

2. Теоретичні відомості

2.1. Вибір структури веб-сайту

Для правильного підходу до проектування та створення веб-сайту, потрібно розібратися у видах та класифікаціях веб-сайтів.

Коли користувач заходить на сайт, йому важливо швидко знайти потрібну інформацію або скористатися сервісом. Інтуїтивно зрозуміла організація сайту значно полегшує пошук для користувача. Важливо, щоб шлях до будь-якої інформації на сайті займав не більше кількох кліків користувача. Для комерційних сайтів цей аспект особливо важливий, оскільки складна і нечітка структура сайту відштовхує і веде потенційних клієнтів до конкурентів. Також продумана структура сайту дозволяє змінювати або масштабувати ресурс у майбутньому.

Сайти можна класифікувати за доступністю сервісів, фізичним розташуванням і призначенням [7].

1. За доступністю сервісів їх поділяють на:

- Відкриті – усі сервіси повністю доступні для будь-яких користувачів;
- Напіввідкриті – для доступу потрібно зареєструватися (зазвичай безкоштовно);
- Закриті – повністю закриті службові сайти організацій (наприклад, корпоративні сайти), особисті сайти приватних осіб. Такі сайти доступні для вузького кола людей. Доступ нових людей можливий через запрошення або входження у систему за кодами доступу.

2. За фізичним розташуванням:
   - Зовнішні – якщо сайт доступний користувачам з Інтернету;
   - Внутрішнім – доступ до якого можуть здійснювати лише користувачі локальної мережі. Прикладом внутрішнього сайту може бути корпоративний сайт підприємства в локальній мережі провайдера.
3. За призначенням сайти поділяються на:
   - Бізнес-сайти – сайти, які містять інформацію про компанії та їхні послуги;
   - Інформаційні сайти – призначені для інформування користувачів, новин, тематичні сайти, енциклопедії, словники тощо;
   - Сайти соціальних мереж – це інтерактивні багатокористувацькі веб-сайти, які заповнюються самими учасниками мережі;
   - Веб-портали – це універсальні сайти, за допомогою яких можна отримати доступ до інших ресурсів Інтернет;
   - Сайти-сервіси - сайти-сервіси, які існують в мережі Інтернет, зокрема сайти пошукові служби, поштові сайти, онлайн-сховища дані, зберігання відео тощо;
   - Та інші.

Отже, розуміючи види веб-сайтів можемо класифікувати розробку як сайт-сервіс, який є відкритим та доступним для всіх користувачів Інтернету.

2.2.Структура веб-сторінки

Веб-сторінка є одиницею веб-сайту. Структура веб-сторінки є важливим аспектом, бо вона визначає вигляд цієї сторінки. Кожна сторінка може значно відрізнятися одна від одної, але всі вони, мають подібні стандартні компоненти [8]:

- Заголовок: це великий прямокутний блок зверху сторінки. Може мати логотип, слоган тощо. Зазвичай, заголовок не змінюється від сторінки до сторінки та залишається статичним;
- Панель навігації: теж великий прямокутний блок, який знаходиться одразу після заголовку або є його частиною. Панель навігації містить посилання на головні розділи сайту, вибір мови тощо. Він може бути представленим у вигляді кнопок меню або випадаючого списку. Як і заголовок, цей блок зазвичай є статичним;
- Основний зміст: великий блок розташований по центру сторінки та містить основний контент. Ця частина є унікальною та буде відрізнятися на різних сторінках сайту;
- Бічна панель: необов'язковий блок, який може знаходитися зліва або справа від основного змісту. Представляє собою додаткову інформацію або рекламу. Також, існують випадки, коли бічна панель повторюється або використовується як допоміжна навігаційна панель;
- Футер: прямокутний блок внизу сторінки, який містить повідомлення про авторські права, контактну або додаткову інформацію. Футер зазвичай є статичним.

Приклад структури з веб-сайту «MDN» можна побачити на рисунку 2.1.

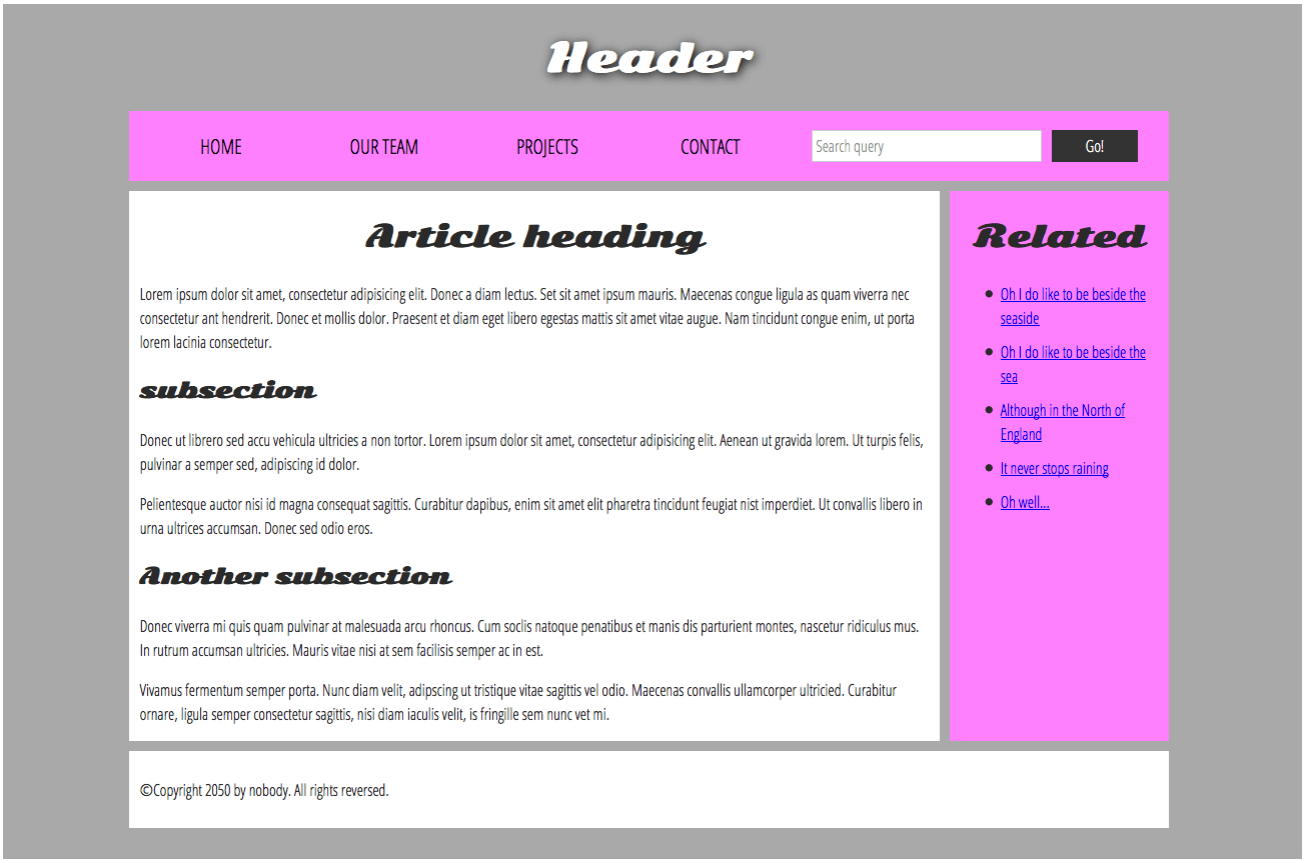


Рисунок 2.1 – Структура веб-сайту

Веб-сторінка є важливою складовою будь-якого веб-сайту, оскільки її структура визначає вигляд та організацію контенту. Однак, щоб веб-сторінка була ефективною, вона повинна бути доступною та привабливою для будь-якої аудиторії, була зрозуміла якомога більшій кількості національностей, молодим, дорослим та людям похилого віку.

Для залучення максимальної кількості аудиторії потрібно обов’язково використовувати інтернаціональну мову, або ту користувачів якої найбільше на планеті. Так як у даного сайту не визначена строго категорія аудиторії, то найкращим варіантом буде використовувати англійську мову. Вона дозволить сотням мільйонів людей розуміти наповнення сторінок, а іншим легко переводити сторінку за допомогою плагінів у браузері, або вручну.

Для молодої аудиторії, веб-сторінка повинна бути інтерактивною та цікавою, з використанням анімацій. Також, важливо мати чітку навігаційну структуру, яка допоможе користувачам швидко знайти необхідну інформацію.

Для дорослої аудиторії, веб-сторінка повинна виглядати достатньо солідно та серйозно.

Для старшої аудиторії, веб-сторінка повинна бути простою та інтуїтивно зрозумілою, з чіткою структурою. Кольори та фон повинні бути приємними для ока та не викликати напруження.

Авжеж, сторінка повинна мати адаптивний дизайн для того щоб коректно працювати та гарно виглядати на будь-якому пристрої.

Таким чином, створюючи веб-сторінку, важливо враховувати потреби різних груп, щоб забезпечити їм зручний та приємний досвід користування.

2.3.Вибір кольорів веб-сайту

Вибір кольорової гами для інтерфейсу веб-сайту є важливим аспектом, бо впливає на його сприйняття та читабельність. Для того, щоб користувачу було приємно переглядати сторінки сайту та легко знаходити посилання, інформацію тощо, важливо підібрати відповідні кольори. Постійно з’являються нові тренди щодо кольорів, які також важливо брати до уваги та використовувати.

Для веб-сайту вибрано кольорову гамму яка складається з відтінків темно-синього кольору, чорного, сірого та білого кольорів. Кожен колір відповідає за свою функцію. Дизайнери описують даний набір так: «Поєднання синього і чорного справить незабутнє враження, а пастельні допоможуть додати тепло і врівноваженість. Ця кольорова палітра є новим подихом до вже традиційної чорно-білої кольорової гами» [9]. Також, для фону сторінок використовуються

різні патерни, які не заважають користувачу бачити контент, але додають особливості та оригінальності до дизайну веб-сайту.

3. Опис реалізації програмного продукту

3.1. Аналіз технічного завдання

У цьому розділі описано та доповнено деталі з розділу 1.3, його особливості, важливі аспекти алгоритму та взаємодію користувача з веб-сайтом.

Веб-застосунок повинен надавати можливість користувачу шукати маршрут для мандрівок за допомогою фільтрів, у спеціальній формі вводу на веб-сайті, за параметрами описаними у розділі 1.3 та переглядати маршрути з додатковою інформацією та корисними посиланнями.

Маршрути складаються з переліку міст, які як найкраще підходять під визначений пошук. Для реалізації цього, потрібно створити алгоритм, який буде вираховувати та обробляти дані. Основні кроки алгоритму:

1. Алгоритм отримує запит та шукає міста, які підходять по параметрам фільтрації у базі даних;
2. Алгоритм перемішує дані у випадковому порядку для створення ефекту випадкового маршруту;
3. Алгоритм обирає певну кількість релевантних даних, таку кількість, яка дозволить повернути результат не довше ніж за 10 секунд;
4. Алгоритм за допомогою комбінаторної формули обчислює всі можливі маршрути зі взятих даних;
5. Алгоритм обробляє та модифікує дані додаючи до кожного маршруту певні властивості та формуючи з нього фінальний об’єкт-маршрут;

6. Алгоритм сортує об’єкти-маршрути за кількістю міст у маршруті за спаданням та кількість міст з одного континенту (для вираховування відносної відстані) за спаданням;
7. Алгоритм формує фінальний об’єкт в якому знаходиться масив об’єктів-маршрутів та відправляє його на клієнт.

Відносна відстань, згадана у кроці 6 вираховується для поліпшення та оптимізації маршруту. Також зазначена кількість міст також важлива, адже алгоритм намагається знайти маршрут по всім вказаним країнам від користувача (або кількості зупинок, якщо користувач не вказував певні країни).

Таким чином, алгоритм шукає та обробляє пошукові дані для того щоб зберегти всі бажання користувача, а також оптимізувати маршрут та вивести зверху вниз списком починаючи з найрелевантнішого.

Інтерфейс веб-сайту повинен бути інтуїтивно зрозумілим та без зайвих елементів. Як тільки користувач потрапив на веб-сайт – він повинен відразу зрозуміти де знаходиться основний контент, а саме пошукова форма.

3.2.Архітектура застосунку

Після аналізу технічного завдання, було вирішено створити застосунок на клієнт-серверній архітектурі. На рисунку 3.1 можна побачити схему даної архітектури.

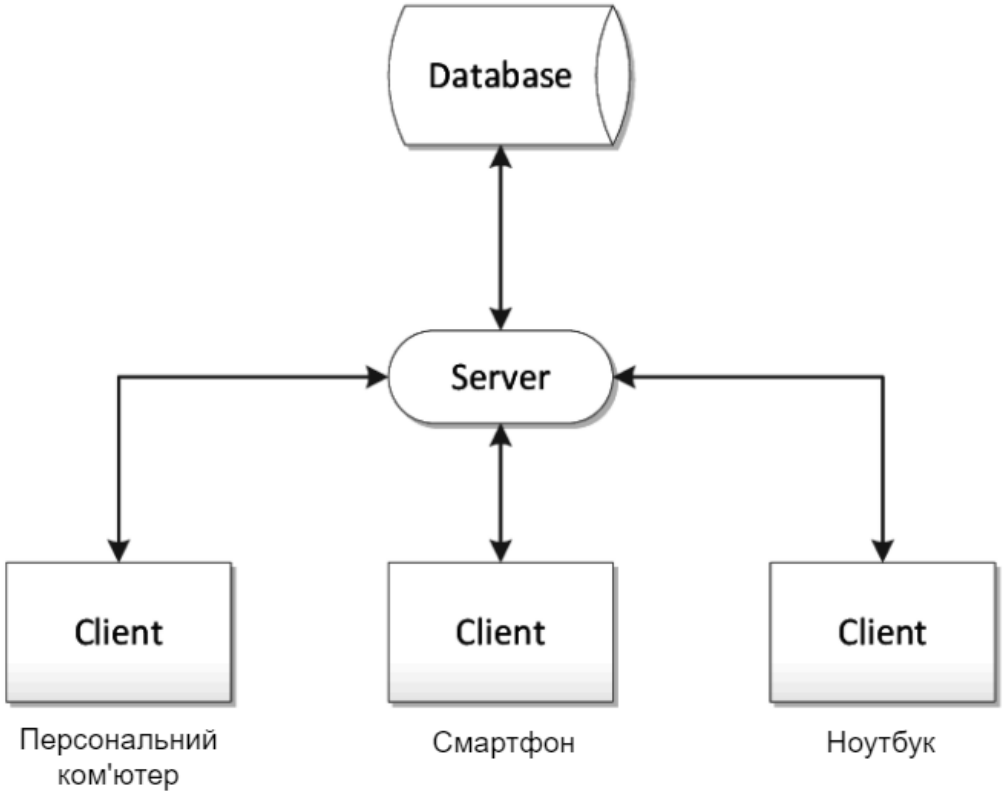


Рисунок 3.1 - Схема клієнт-серверної архітектури

Клієнт-серверна архітектура складається з:

- Клієнтський веб-браузер;
- Веб-сервер;
- База даних.

У цій архітектурі клієнтом є браузер, який використовує особа-користувач. Оскільки програма є веб-сайтом і розміщена в Інтернеті, як вже було зазначено у частині 2.1, будь-хто, хто має підключення до Інтернету, може отримати доступ до сервісу. На схемі представлені найпопулярніші гаджети для перегляду Інтернету, а саме: ПК, смартфон та ноутбук.

Спочатку клієнтський веб-браузер робить запит на сервер у формі HTTP-запиту. Після отримання запиту сервер робить запит до бази даних і отримує відповідні та актуальні дані, потім обробляє їх, за потреби робить обчислення

та маніпуляції з даними, та відправляє як відповідь клієнту знову через протокол HTTP. Клієнт отримує певні дані та відображає їх у веб-браузері.

Крім того, важливо зазначити, що веб-застосунок побудовано відповідно до патерну проектування MVC (Model-View-Controller). Патерн передбачає поділ архітектури програми на 3 частини: контролер (Controller), модель даних (Model) та інтерфейс (View). Всі 3 частини з'єднані та взаємодіють одна з іншою.

- Інтерфейс – це те, що людина побачить під час використання програми, у нашому випадку під час перегляду сторінок веб-сайту. Інтерфейси взаємодіють з контролерами, надсилаючи їм запити.
- Контролер – це логіка програми, яка взаємодіє з інтерфейсом і відповідає на його запити, а також маніпулює моделлю даних.
- Модель даних – отримує певний запит від контролера та надсилає інтерфейсу певні дані.

Отже, у даній схемі маємо такі ролі:

- Роль інтерфейсу (View) виконує клієнтський веб-браузер;
- Роль контролера (Controller) виконує сервер;
- Роль моделі даних (Model) виконує база даних.

На рисунку 3.2 представлена схема MVC.

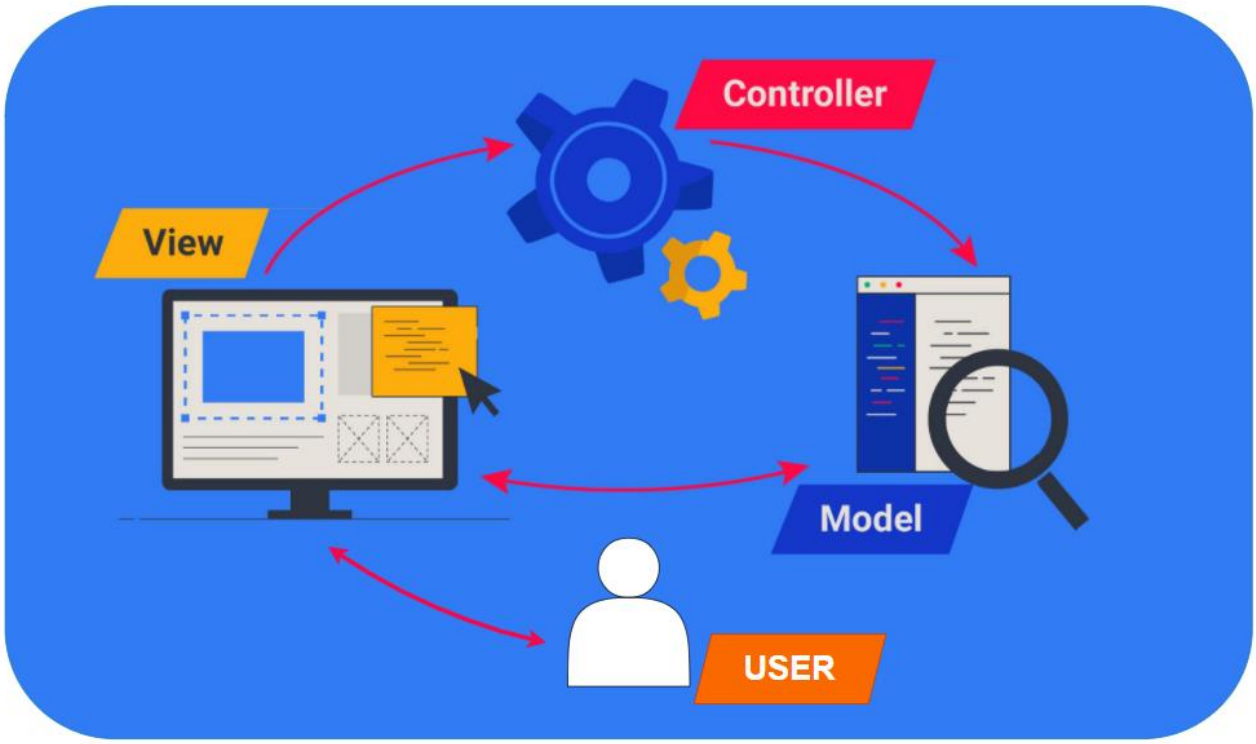


Рисунок 3.2 - Схема патерну проектування MVC

За допомогою MVC патерну можна розділити застосунок на логічні компоненти, що полегшує розробку, тестування та підтримку коду. Крім того, цей підхід дозволяє змінювати один компонент, не впливаючи на інші, що дозволяє швидко та ефективно розвивати додаток.

У нашому випадку головні складові MVC патерну є:

- Model – нереляційна база даних MongoDB, доступ до якої буде у серверної частини застосунку;
- View – бібліотека React, клієнтська частина застосунку, яка відображатиме отримані дані користувачеві у браузері;
- Controller – бібліотека Express для створення сервера на основі Node.js для взаємодії з клієнтом та базою даних.

3.3.Обґрунтування вибору засобів розробки

3.3.1. Засоби розробки клієнтської частини

Для розробки клієнтської частини застосунку було обрано мову програмування JavaScript. JavaScript – це прототипно-орієнтована, мультипарадигменна мова з динамічною типізацією, яка підтримує декілька стилів програмування: об'єктно-орієнтовану, імперативну та декларативну [10]. Ця мова є по суті єдиною, що використовується як основа для програмування на стороні браузера. З іншого боку JavaScript використовується в інших сферах програмування, наприклад, у створенні простих комп’ютерних ігор.

Для створення веб-інтерфейсів було використано JavaScript-бібліотеку React. React – це декларативна, ефективна і гнучка JavaScript-бібліотека, призначена для створення інтерфейсів користувача. Вона дозволяє компонувати складні інтерфейси з невеликих окремих частин коду – «компонентів» [11]. Побудова структури сторінки відбувається за допомогою спеціального синтаксису «JSX» який дозволяє писати код JavaScript всередині HTML розмітки. React наразі є найпопулярнішою бібліотекою JavaScript для розробки веб-інтерфейсів з тисячами додаткових бібліотек, які дають змогу побудувати унікальний продукт поєднуючи їх.

Невід’ємна частина браузерної частини є бібліотека Redux, що дозволяє глобально керувати станом усього застосунку. Це є дуже корисним через специфіку React та розширює можливості стандартного підходу за допомогою архітектури Redux [12].

Для побудови адаптивної розмітки, яка має добре виглядати на будь-якій роздільній здатності екрану було використано зв’язку бібліотек Tailwind CSS та DaisyUI, які дозволяють з легкістю будувати інтерфейси з готових блоків та

з налаштуванням усієї логіки адаптивності. Варто зазначити якісну інтеграцію з бібліотекою React.

Список основних бібліотек використаних на клієнтській частині застосунку (назва: версія):

- axios: 0.27.2;
- daisyui: 2.24.0;
- rc-slider: 10.0.1;
- react: 18.2.0;
- react-dom: 18.2.0;
- react-redux: 8.0.2;
- react-router-dom: 6.3.0;
- react-select: 5.4.0;
- redux: 4.2.0;
- redux-thunk: 2.4.1;
- tailwindcss: 3.1.8;
- та інші.

Всі бібліотеки завантажені за допомогою Node Package Manager, що є менеджером пакунків для мови програмування JavaScript.

3.3.2. Засоби розробки серверної частини

Завдяки платформі NodeJS – є можливість створювати серверну частину застосунку за допомогою тієї ж мови програмування JavaScript. Node.js – це програмна платформа, яка транслює скрипт JavaScript у машинний код і такий код може виконуватися на стороні сервера [13]. Це дуже зручно, адже для розробки повноцінного застосунку як клієнтської, так і серверної частини – за основу береться одна мова. Також, треба зазначити, що існує велика кількість бібліотек у тому ж Node Package Manager, які дозволять створювати і сервера.

Основні бібліотеки, що використовується на стороні сервера це express, mongoose та axios. Express – це фреймворк, який реалізує шар функцій, необхідних для створення API та має готові функції обробки HTTP запитів. Mongoose – це модуль, за допомогою якого відбувається з'єднання, створення схем даних та подальша робота з базою даних MongoDB за допомогою вбудованого API. Axios – бібліотека, основна функція якої HTTP запити з NodeJS.

Перелік бібліотек використаних на серверній частині застосунку (назва: версія):

- axios: 0.27.2;
- body-parser: 1.20.0;
- cors: 2.8.5;
- dotenv: 16.0.2;
- express: 4.18.1;
- mongoose: 6.5.0.

3.3.3. Засоби розробки бази даних

Для зберігання даних було обрано документо-орієнтован базу даних MongoDB. Ця база даних класифікується як нереляційна (NoSQL), що заперечує більшість підходів притаманних реляційним базам даних. Кожна база даних має певну кількість колекцій, їх можна порівняти з таблицями у реляційних базах даних. Кожна колекція має свої документи, що містять певні властивості. У подальших пунктах базу даних застосунку буде детально описано.

На відміну баз даних типу SQL, MongoDB, завдяки своїм документам – працює швидше, має кращу масштабованість та легша у використанні [14]. Дана БД використовує JSON-подібний формат, що у свою чергу є JavaScript

Object Notation. Ця особливість допомагає працювати з базою даних, адже по суті, це ті ж самі об’єкти, що ми маємо у мові JavaScript. Інтеграція з серверною частиною є легкою та зручною.

Для зберігання даних у хмарі було використано Mongo Atlas, що дає змогу отримувати дані з БД MongoDB звертаючись до їх сервісу через Інтернет.

3.3.4. Середовище розробки

Як середовище розробки було обрано саме WebStorm від компанії JetBrains [15]. На сьогодні це середовище є найкращим програмним забезпеченням для розробки веб-сайтів та взагалі роботи з JavaScript і суміжними бібліотеками. Даний редактор має зручний функціонал:

- розумний редактор коду;
- вбудовані інструменти для та підказки розробнику;
- швидка навігація та глобальний пошук по проекту і окремий пошук по модулю чи файлу;
- кастомізація середовища;
- а також вбудована систему контролю версій Git з підтримкою перегляду змін прямо у файлах з кодом.

3.3.5. Засоби розгортання веб-застосунку в Інтернет

Для того, щоб користувачі мали доступ до веб-застосунку у мережі Інтернет, потрібно розгорнути дані застосунки у хмарі, а також, надати унікальний домен для зручності.

Клієнтська частина застосунку розгорнута на платформі Netlify. «Netlify допоможе вам об’єднати ваші улюблені інструменти та API для створення найшвидших сайтів, магазинів і додатків для комбінованої мережі» [16]. Дана

платформа має багато рідного функціоналу і вважається однією з найкращих та найпростіших способів розгорнути браузерну частину веб-сайту.

Другою невід'ємною частиною розгортання застосунку є його серверна частина, що розгорнута на платформі Vercel. «Vercel – це платформа для веб-розробників, яка забезпечує швидкість і надійність, необхідні інноваторам у момент натхнення. Vercel має підтримку без зайвої конфігурації для більше 35 веб фреймворків» [17]. Ця платформа надає можливість розгортати веб-сервер безкоштовно, але з певними обмеженнями. Саме тому у частині 3.1 зазначено про тривалість отримання результату не більше 10 секунд, адже, саме безкоштовна версія дозволяє мати затримку у відповіді не більше цього часу.

Також, варто зазначити веб-сайт «namecheap.com», на якому було придбано доменне ім'я для застосунку. Користувачі Інтернет можуть отримати доступ до веб-застосунку за адресою https://travbase.net/.

3.4.Опис бази даних

У MongoDB дані зберігаються у вигляді документів з певними властивостями, документи збираються у колекції і формують базу даних. Для зберігання даних у застосунку було створено дві колекції: міста (cities) та країни (countries).

Розглянемо властивості документа з колекції міст:

- _id: унікальний ідентифікатор;
- name: назва міста;
- popularity: популярність міста (1, 2, 3, 4, 5);
- pricePerDay: середня ціна перебування у міста на день (у доларах США);
- tourismType: масив з переліком типів туризму (активний-пригодницький, культурний, пляжний, на природі);

- seasons: масив з переліком типів сезону (зима, весна, літо, осінь);
- info: додаткова текстова інформація про місто;
- image: посилання на картинку;
- country: країна, де знаходиться місто.

Розглянемо властивості документа з колекції країн:

- _id: унікальний ідентифікатор;
- name: назва країни;
- continent: континент, де знаходиться країна;
- info: додаткова текстова інформація про країну;
- image: посилання на картинку.

На наступних двох рисунках 3.3 та 3.4 зображено приклади документів з кожної колекції, їх властивості та типи.

Рисунок 3.3 - Приклад документу у колекції cities

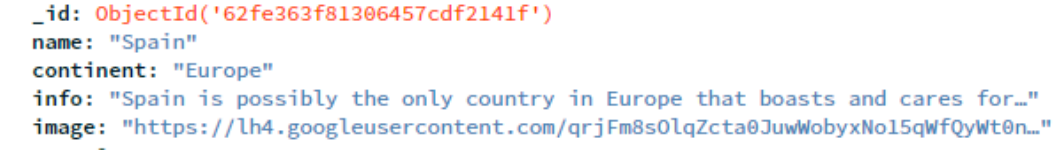


Рисунок 3.4 - Приклад документу у колекції countries

3.5.Опис розробки програми

3.5.1. Структура проекту

Структура проекту складається з двох головних папок «client» та «server», що мають файли клієнтської та серверної частини відповідно. Скріншот можна побачити на рисунку 3.5. Також, розглянемо деталі з кожної частини.

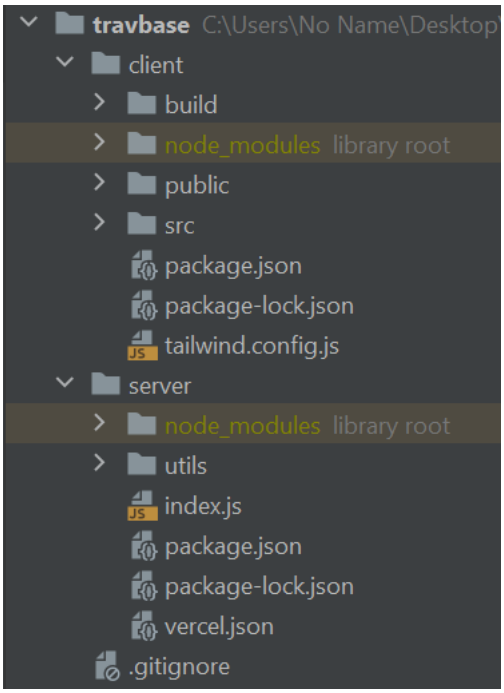


Рисунок 3.5 – Структура проекту

Клієнтська частина складається з таких каталогів та файлів:

- каталог «build», де знаходиться підготовлена версія проекту для завантаження на хмару;
- каталог «node_modules», де зберігаються встановлені бібліотеки;
- каталог «public», де зберігаються картинки та іконки, а також, початковий HTML файл;
- каталог «src», де знаходиться логіка браузерної частини, у тому числі компоненти React, архітектура Redux та допоміжні стилі CSS;

- конфігураційні файли «package.json» та «package-lock.json» для Node Package Manager;
- конфігураційний файл «tailwind.config.js» для бібліотеки Tailwind CSS.

Серверна частина складається з таких каталогів та файлів:

- каталог «node_modules», де зберігаються встановлені бібліотеки;
- каталог «utils», де зберігаються функціонал обробки даних при запиті пошуку за фільтром та створення об’єкту відповіді;
- файл «index.js», який є головним на сервері, відповідає за з’єднання з базою даних та обробкою запитів від браузера;
- конфігураційні файли «package.json» та «package-lock.json» для Node Package Manager;
- конфігураційний файл «vercel.json» для роботи з сервісом Vercel.

3.5.2. Опис розробки клієнтської частини

Головні будуючи блоки клієнтської частини є компоненти React. React дозволяє по різному створювати ці компоненти, але, в даній роботі використовуються саме функціональні компоненти. Це дозволило значно скоротити та спростити код за допомогою «хуків». Компоненти поділені на 4 логічні частини: опис початкової сторінки (Home), опис сторінки переліку запропонованих маршрутів (FindRoutes), опис сторінки маршруту (Route) та опис фільтра, який використовується на сторінках Home та FindRoutes (Filter).

Також, варто зазначити Redux, який відповідає за зберігання всього стану застосунку. Можна виділи 4 логічні частини: дані про країни (застосовуються для підвантаження актуального списку країн до першого поля фільтру), дані про результати пошуку маршрутів, дані чи видимий фільтр на сторінці FindRouter, дані про обрані параметри фільтрації. Останні два використовуються для того, щоб користувачу було зручно повертатися до

попередніх фільтрів при перегляді певного маршруту або виходу до початкової сторінки.

3.5.3. Опис розробки серверної частини

Головним файлом серверу є «index.js», а також допоміжні файли з функціями обробки даних.

Робота сервера забезпечується HTTP-запитами:

- POST запит «/add-country» - додавання нової країни до базу даних;
- POST запит «/add-city» - додавання нового міста до бази даних;
- GET запит «/countries» - отримання усіх країн з бази даних;
- GET запит «/countries/:name» - отримання країни за її іменем з бази даних;
- GET запит «/cities» - отримання усіх міст з бази даних;
- POST запит «/filter» - отримання списку маршрутів після використання фільтру.

Головні функції обробки даних:

- «citiesFilter» - функція, що фільтрує всі міста по запиту користувача;
- «shuffleArray» - функція, що перемішує відфільтровані міста для отримання ефекту випадковості;
- «createRouts» - функція, що реалізує алгоритм створення маршрутів, має в собі багато підфункцій які виконують різні дії: створення комбінацій, повторний запит до сервера, створення об’єктів маршруту, сортування та інше.

3.5.4. Опис розробки алгоритму пошуку мандрівок

У цьому розділі буде описано кроки алгоритму з розділу 3.1 з технічної сторони.

Алгоритм складається з таких кроків, кожен з яких відповідає коротким крокам опису алгоритму з розділу 3.1:

1. Алгоритм починається з того, що сервер отримує POST запит «/filter», оброблюючи його, він у свою чергу робить запит до бази даних з параметром у вигляді країн, що користувач обрав. Даними, що повертає база даних є масив об’єктів, які представляють собою опис певного міста. Отримуючи відповідь від бази даних, дані відправляються до функції «citiesFilter», що вже фільтрує отримані результати з бази даних по іншим параметрам, що користувач зазначив. Варто зазначити, що, якщо отриманих кількість релевантних об’єктів є меншою за кількість «зупинок» у маршруті, або ж, релевантні дані взагалі відсутні, то користувачу видається повідомлення про неможливість знайдення подібного маршруту та прохання повторити запит з іншими параметрами;
2. Отримуючи усі релевантні дані ми не можемо відобразити користувачу, адже після створення комбінацій (крок 4) в нас може створитися сотні маршрутів. Навіть, якщо ми можемо повернути таку велику кількість даних клієнту, в цьому немає потреби, адже користувач не буде переглядати кожен з маршрутів. Функція «shuffleArray» перемішує відфільтровані міста для отримання випадкового набору релевантних даних;
3. Алгоритм обирає певну кількість релевантних даних, таку кількість, яка дозволить повернути результат не довше ніж за 10 секунд, а саме 5 об’єктів (міст) для подальшої обробки та відправляє його до функції «createRouts». Дана кількість вибірки також пов’язана з кроком 4;
4. За допомогою функції «createCombinations» з наданих об’єктів за допомогою комбінаторної формули формуються усі можливі комбінації маршрутів та повертається масив з «маршрутів», що у свою чергу є

масивом об’єктів з описом міст. Дана частина алгоритму застосовує рекурсію. Було проведено тестування з різною кількістю даних та як результат знайдено оптимальну кількість об’єктів для формування комбінацій як 5. Також, як зазначено у розділі 3.3.5, дана кількість обрана ще через обмеженість сервісу хостингу сервера;

5. Для кожного «маршруту» алгоритм обробляє та модифікує дані кожного об’єкта та додає певні властивості країни в якому заходиться це місто та його континент для формування фінального об’єкту-маршруту. Дані, що додаються, отримуються з того ж серверу застосунку за запитом GET «/countries/:name», що повертає як країну так і континент за певним містом;
6. Далі застосовуються два сортування для надання користувачу спочатку найрелевантніших маршрутів сторінки, а вже потім менш релевантних. Алгоритм сортує об’єкти-маршрути за кількістю міст у маршруті за спаданням та кількості міст з одного континенту (для вираховування відносної відстані) за спаданням. Це дозволяє продемонструвати користувачу найкращий та найкоротший маршрут по його бажаним параметрам;
7. Алгоритм формує фінальний об’єкт в якому знаходиться масив об’єктів-маршрутів та відправляє його на клієнт, що і відображає цей список маршрутів.

3.6.Опис файлів даних та інтерфейсу програми

У даному розділі буде розглянуто інтерфейс застосунку та можливості користувача.

Треба почати з того, що користувач може отримати доступ до застосунку просто ввівши адресу сайту у пошукову стрічку браузера, а саме «travbase.net».

Відкривши сайт, користувач потрапляє на розділ «Home» домашньої сторінки веб-сайту на якій зображена назва, слоган, та кнопка-якорь до розділу пошуку, яка анімовано показує напрям. Також користувач може бачити навігаційну панель зверху на якій є:

- Кнопка-лого «travbase», яка буде існувати на кожній сторінці веб-сайту. При натисканні кнопки, користувач перенаправляється на домашню сторінку застосунку;
- Кнопка-якорь «Home», яка також повертає користувача на початкову частину домашньої сторінки;
- Кнопка-якорь «Search», яка направляє користувача до розділу пошуку;
- Кнопка-якорь «About us», яка направляє користувача до нижнього розділу сторінки з додатковою інформацією про застосунок.

Розділ «Home» домашньої сторінки сайту можна побачити на рисунку 3.6.

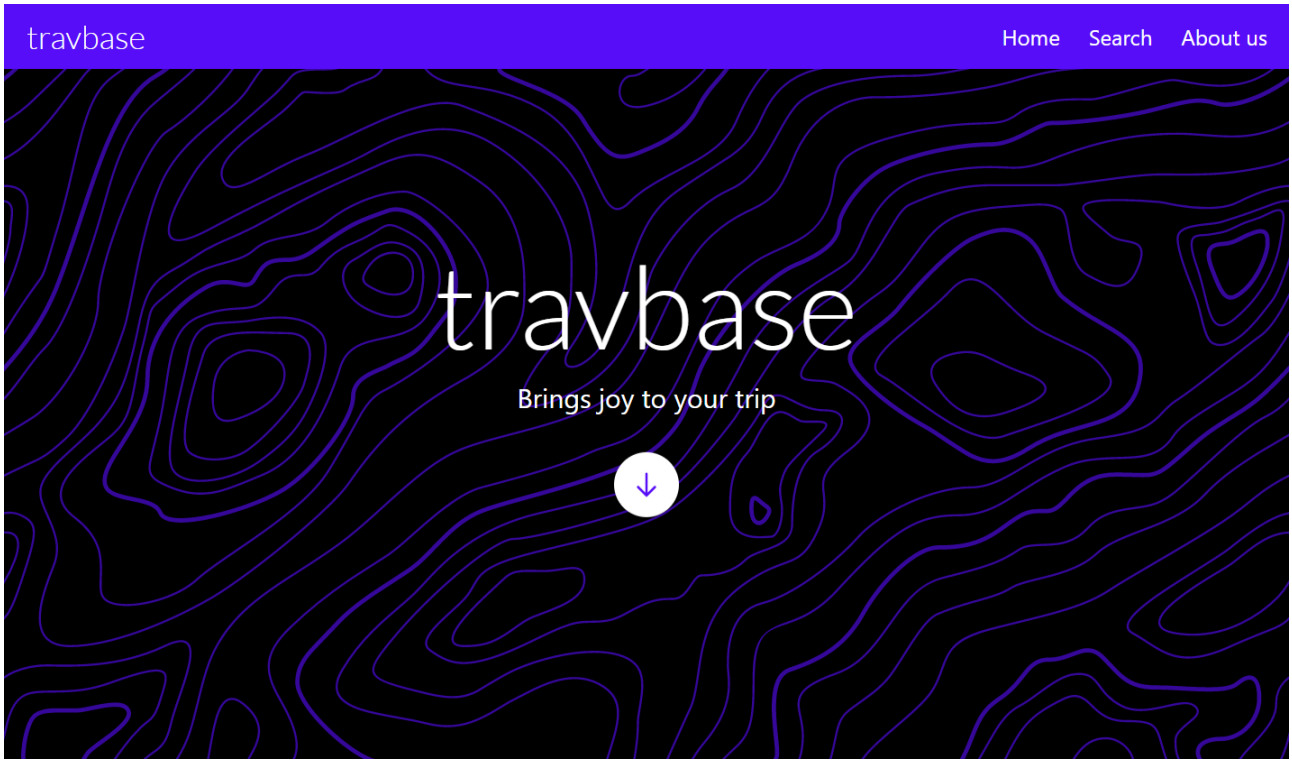


Рисунок 3.6 – Розділ «Home» домашньої сторінки веб-сайту

Користувач може перейти до розділу «Search» натискаючи на анімовану кнопку-якорь по середині екрану, натискаючи на відповідну кнопку-якорь на навігаційній панелі або просто пролистуючи веб-сайт вниз.

Тут користувач бачить форму для заповнення. Єдине поле форми яке є обов'язковим це «Stops», воно по замовчуванню має значення «2» та може бути змінене від «1» до «7». Усі інші поля форми не є обов'язковими і користувач може спираючись на свої сподобання та бажання заповнювати тільки ті поля які йому потрібні.

Форма має такі можливості вводу:

- Вибір країни або декількох країн або будь-якої країни за замовчуванням;
- Вибір бюджету на один день у доларах США або будь-якого бюджету за замовчуванням (від 0 до 500 доларів США на день);

- Вибір сезону (зима, весна, літо, осінь) або декількох сезонів або будь-якого сезону за замовчуванням;
- Вибір типу туризму (активний-пригодницький, культурний, пляжний, на природі) або декількох типів туризму або будь-якого типу туризму за замовчуванням;
- Вибір рейтингу популярності локації (1, 2, 3, 4, 5) або декількох рейтингів популярності локації або будь-якого рейтингу популярності локації за замовчуванням;
- Вибір кількості зупинок (1, 2, 3, 4, 5, 6, 7) або дві зупинки за замовчуванням.

Після заповнення форми, користувач натискає кнопку «SEARCH» та автоматично переходить на сторінку відображення підібраних маршрутів.

Розділ «Search» домашньої сторінки сайту можна побачити на рисунку 3.7.

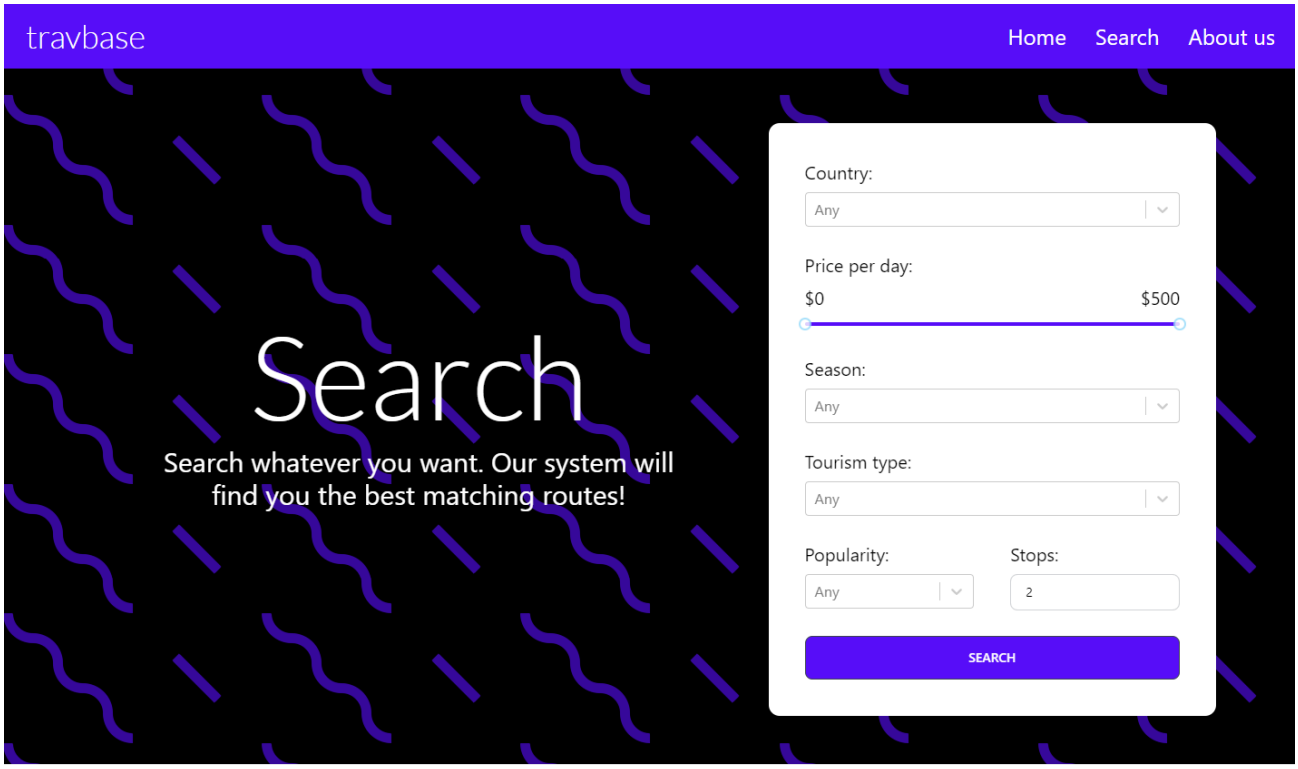


Рисунок 3.7 – Розділ «Search» домашньої сторінки веб-сайту

Якщо користувач хоче побачити додаткову інформацію про застосунок та зв’язатися з власником веб-сайту він може перейти до розділу «About us».

Розділ «About us» домашньої сторінки сайту можна побачити на рисунку 3.8.

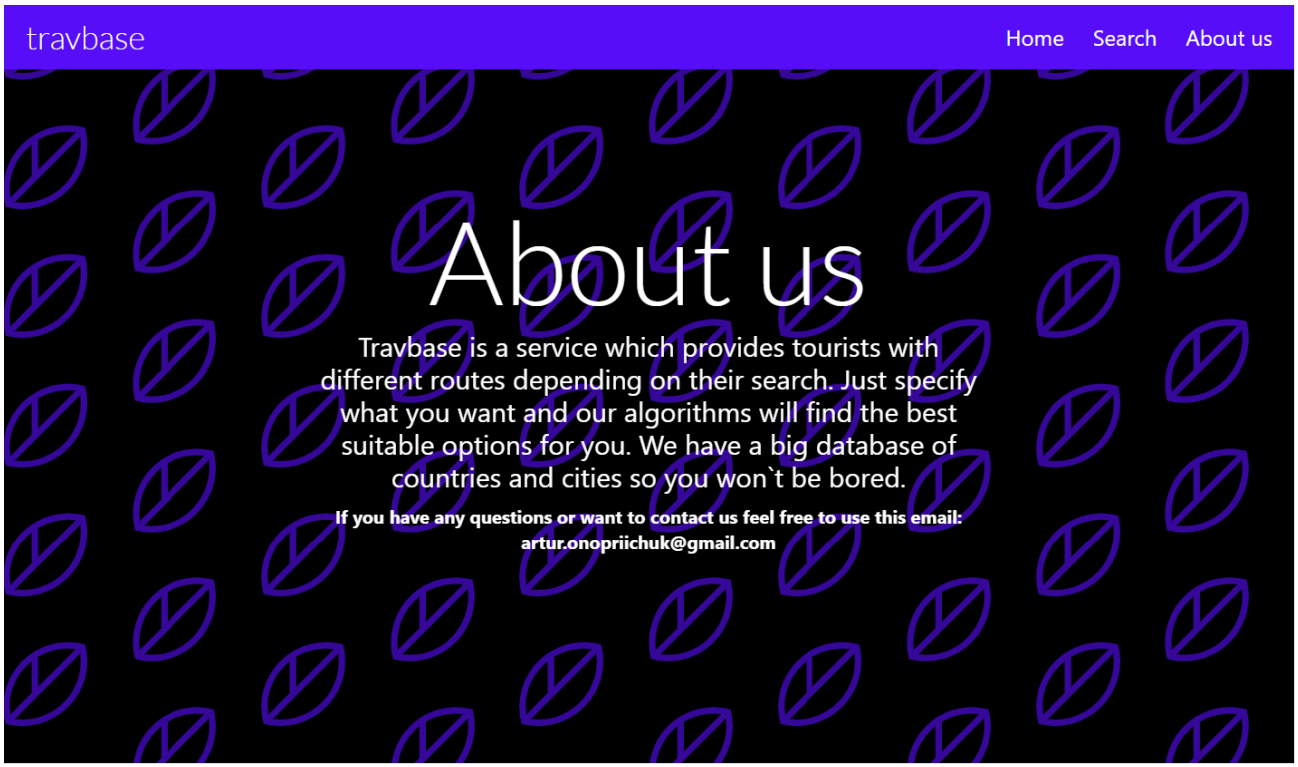


Рисунок 3.8 – Розділ «About us» домашньої сторінки веб-сайту

Після автоматичного переходу на сторінку відображення підібраних маршрутів користувач бачить декілька варіантів підібраних маршрутів за його вподобаннями.

Для прикладу, розглянемо такий фільтра пошуку:

- Вибір будь-якої країни за замовчуванням;
- Вибір бюджету від 0 до 145 доларів США на день;
- Вибір сезону літо;
- Вибір типу туризму активний-пригодницький та пляжний;
- Вибір рейтингу популярності локації;
- Вибір двох зупинок за замовчуванням.

Зазначений приклад фільтру можна побачити на рисунку 3.9.

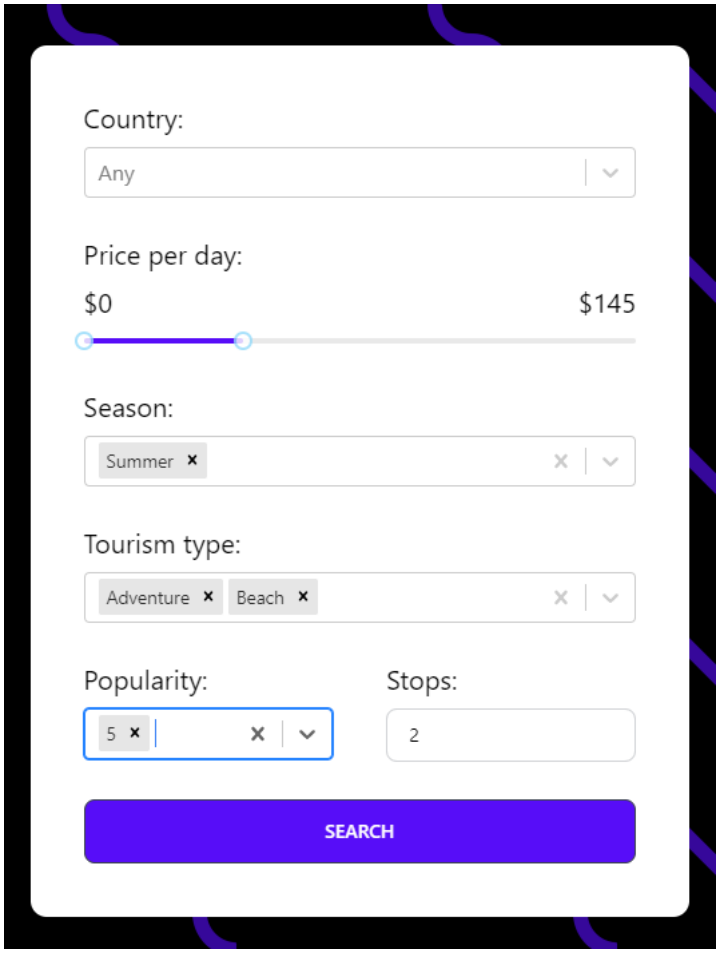


Рисунок 3.9 – Приклад фільтру користувача

З даними фільтрами маємо 10 різних маршрутів, які підпадають під задані вподобання. Переглянемо останні 3 зі списку:

- Маршрут через Бразилію та Іспанію;

- Маршрут через Японію та Іспанію;
- Маршрут через Японію з двома внутрішніми локаціями.

Результат пошуку можна побачити на рисунку 3.10.

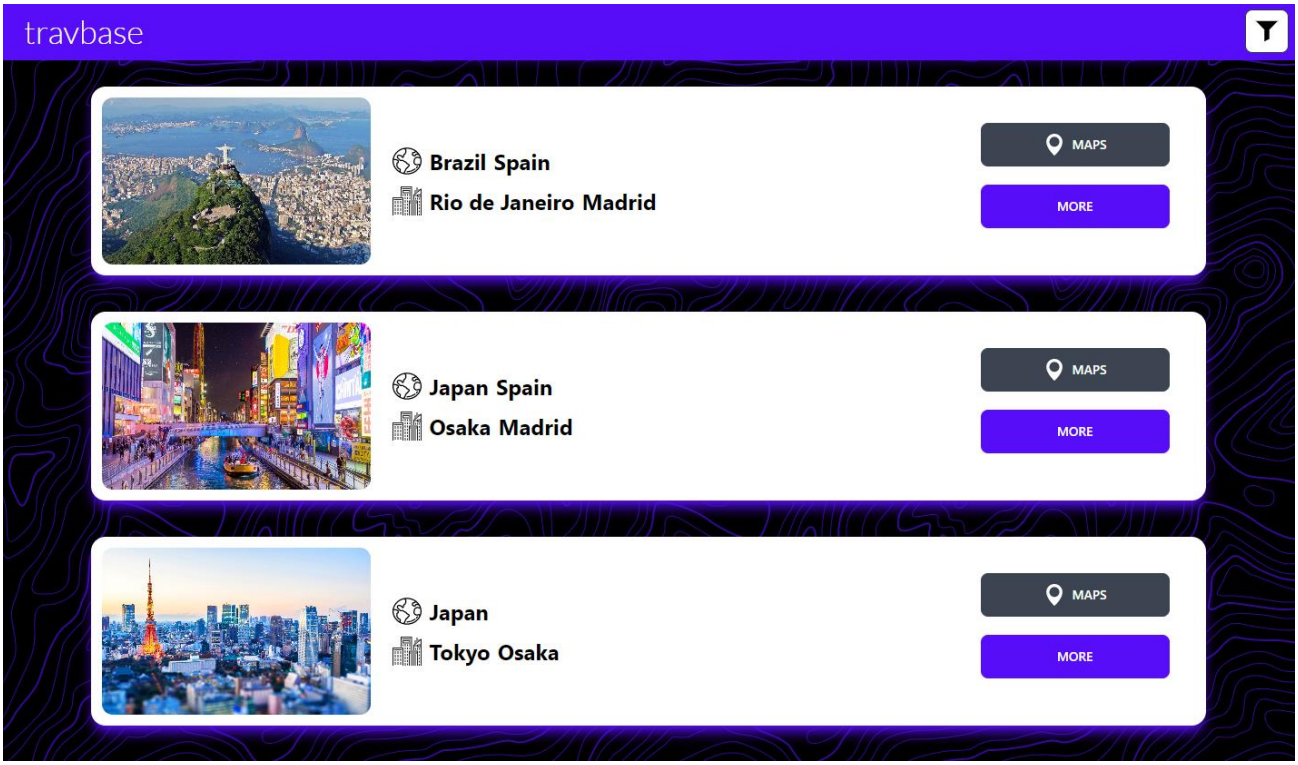


Рисунок 3.10 – Результат пошуку за прикладом фільтрів користувача

У користувача є можливість відкривати фільтр на сторінці списку маршрутів та змінювати пошуковий запит. Кнопка для відкриття\закриття форми фільтрації знаходиться у правій частині навігаційної панелі зверху сторінки. Форма вже заповнена актуальним запитом користувача, що допомагає швидко та зручно редагувати попередній фільтраційний запит.

Також, варто зазначити, що фільтраційні параметри зберігаються протягом усього знаходження користувача на веб-сайті без його перезавантаження, отже, навіть якщо користувач повернеться на домашню сторінку, його фільтраційні параметри зберігаються та будуть відображенні у формі.

Фільтраційна форма на сторінці списку маршрутів можна побачити на рисунку 3.11.

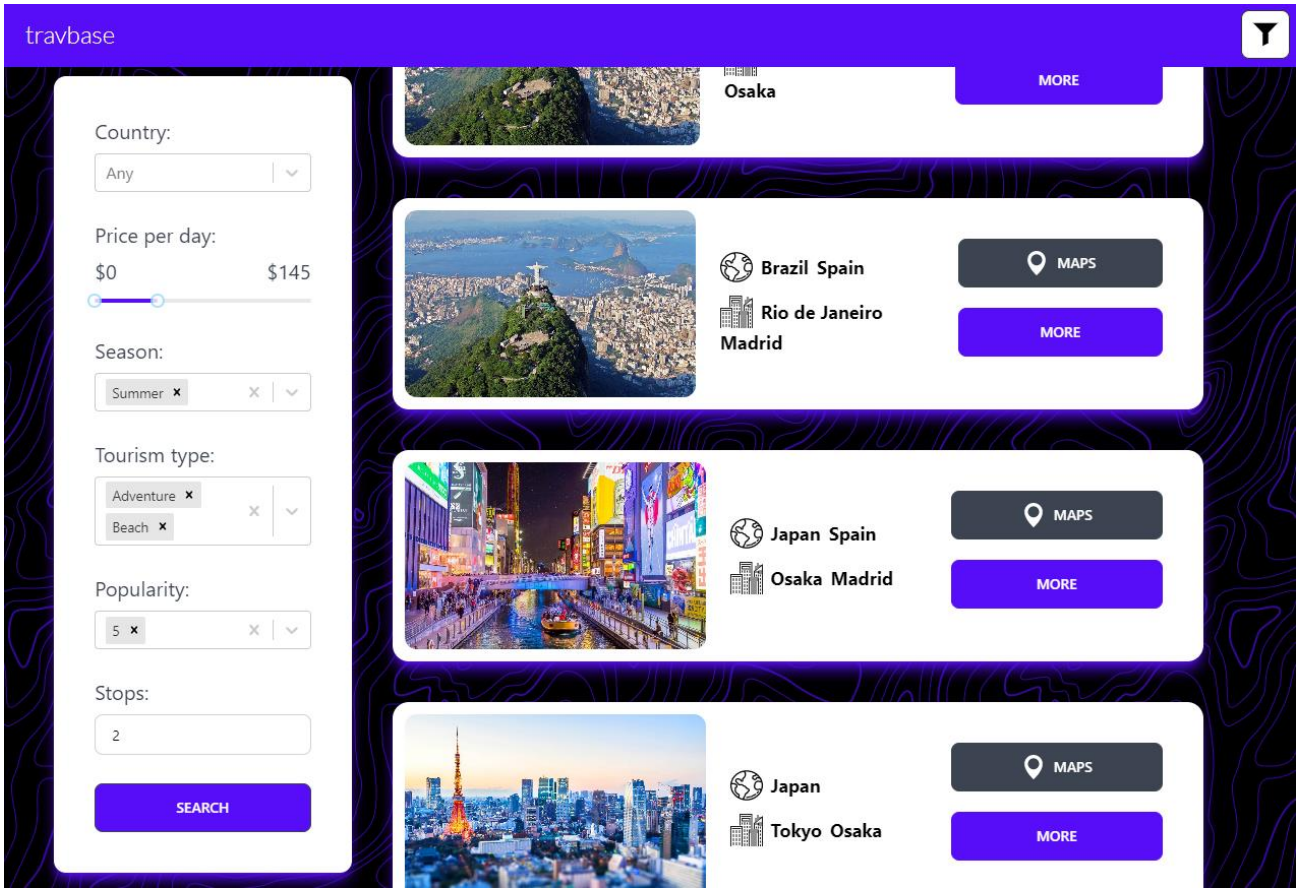


Рисунок 3.11 – Фільтраційна форма на сторінці списку маршрутів

Маршрути відображаються у вигляді карток з короткою інформацією щодо країн та міст які входять у даний маршрут, картинкою та двома кнопками. Кпопка-посилання «MAPS» відповідає за посилання на Google карту. При натисканні ка неї користувач отримує готовий маршрут на карті з усіма переліченими зупинками на маршруті. Карта відкривається у окремому вікну браузера. Кнопка «MORE» відкриває детальну сторінку для певного маршруту.

Приклад картки маршруту можна побачити на рисунку 3.12.

Рисунок 3.12 – Приклад картки маршруту

Приклад мапи маршруту можна побачити на рисунку 3.13.

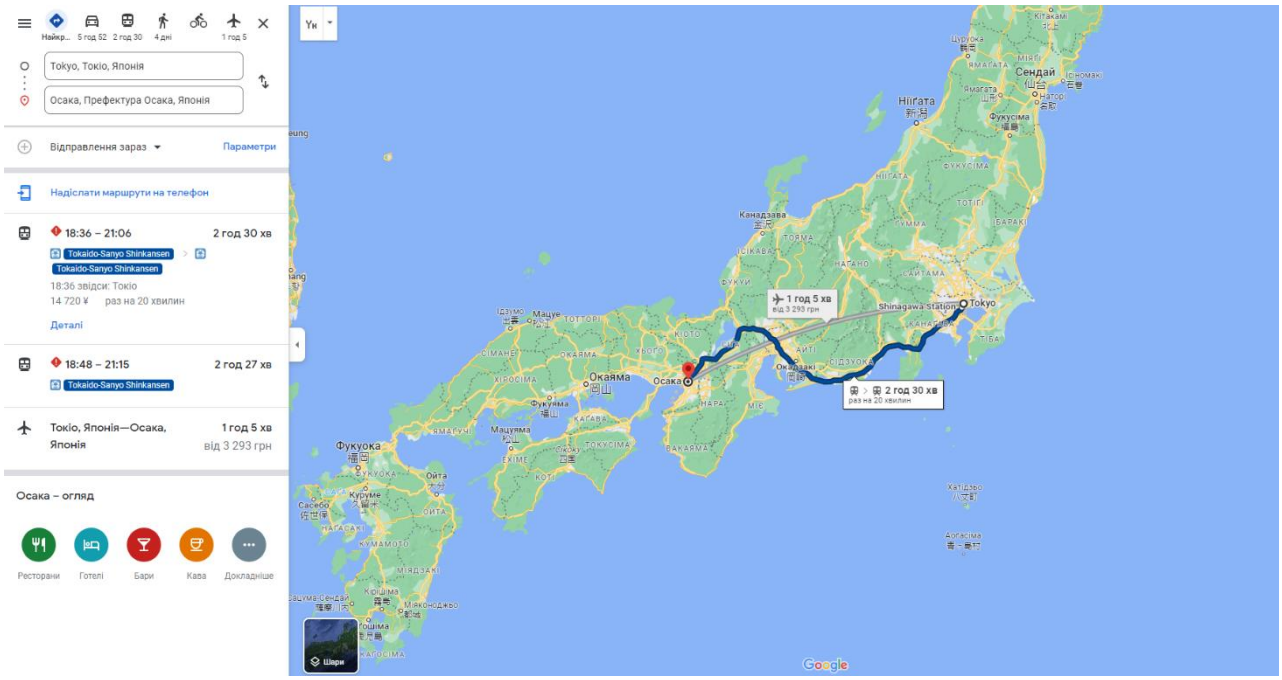


Рисунок 3.13 – Приклад мапи маршруту

При переході на сторінку детальної інформації ми бачимо дві головні секції:

- Секція з інформацією про всі країни які перелічені у маршруті. Країни демонструються у вигляді карток з картинкою, інформацією про назву країни, континент, на якому знаходиться дана країна та короткий туристичний опис;
- Маршрут, який складається з міст. Кожне місто має інформацію про:
    - Зображення;
    - Назву;
    - Країну;
    - Середню ціну туристичного дня;
    - Популярність серед туристів;
    - Найкращий сезон для відвідування;
    - Можливі типи туризму;
    - Опис;
    - Кнопку-посилання «HOTELS ON BOOKING» на варіанти житла у даному місті згідно платформи Booking.com.

  Також, в кінці маршруту користувачу знову надається можливість перейти до перегляду маршруту на карті від Google за допомогою кнопки-посилання «VIEW ON GOOGLE MAPS».

Приклад сторінки з детальною інформацією щодо певного маршруту можна побачити на рисунку 3.14.

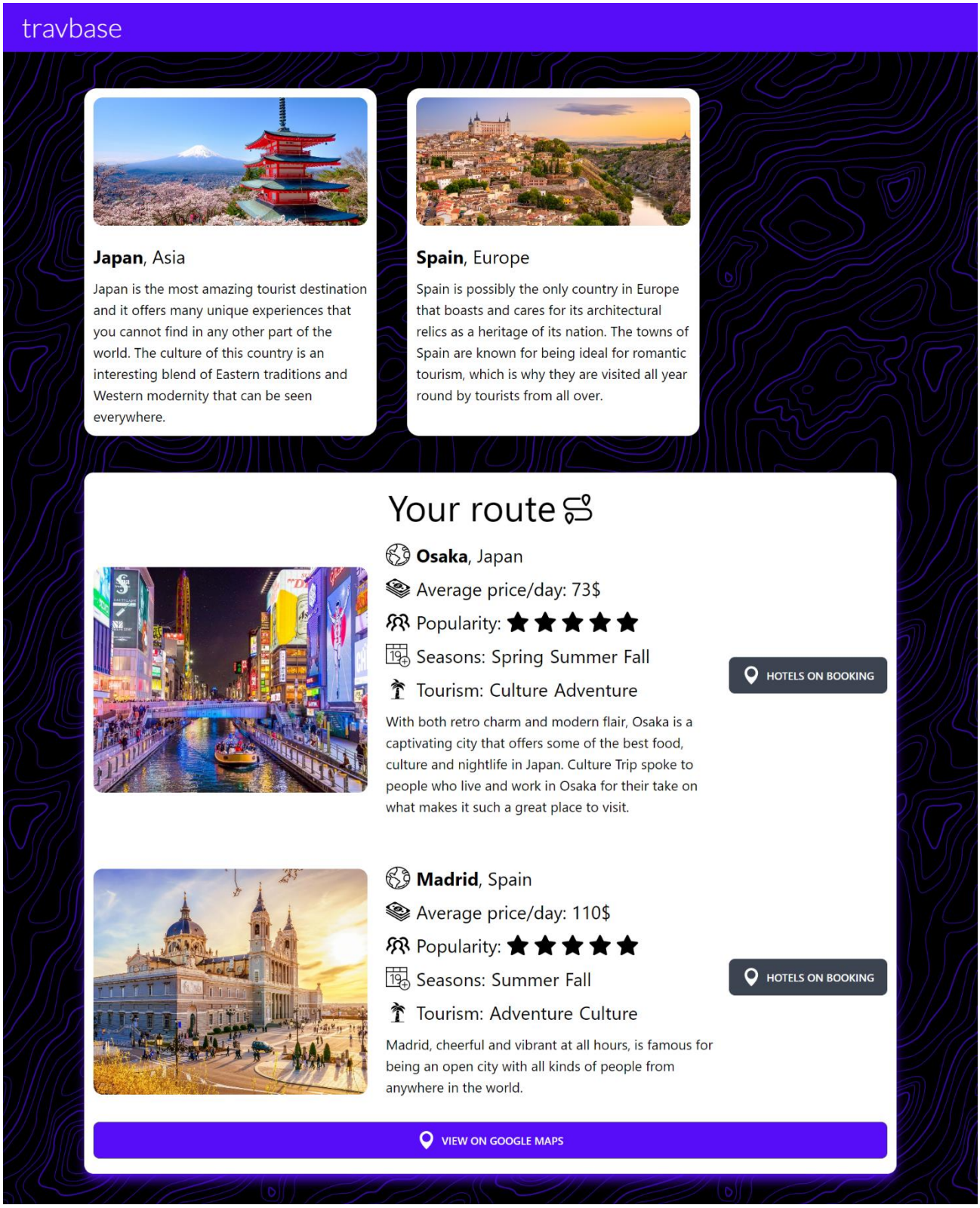


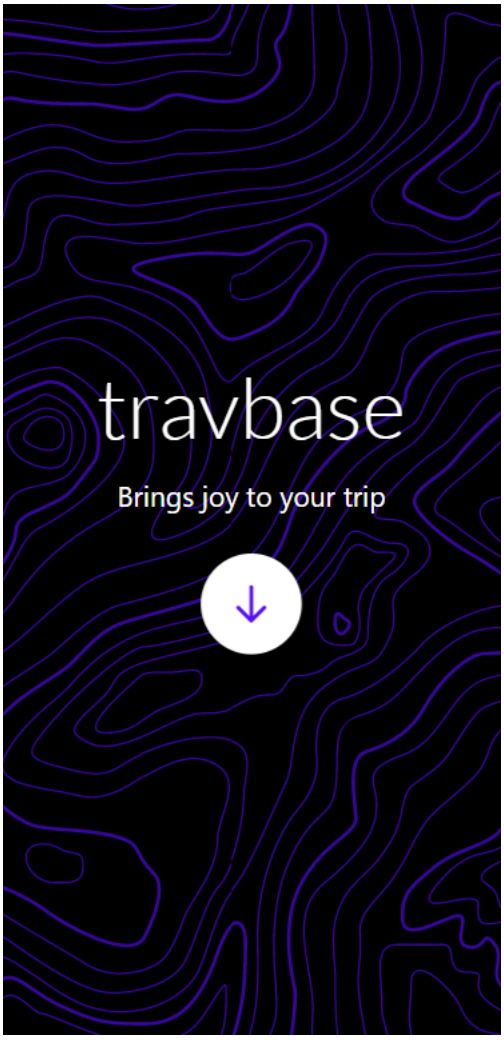
Рисунок 3.14 – Приклад сторінки з детальною інформацією щодо певного маршруту

Користувач може виходити з перегляду детальної інформації певного маршруту. Натискаючи кнопку «назад» у браузері, користувач повертається на попередню сторінку та бачить перелік маршрутів, які є релевантними для нього. Якщо користувач хоче побачити іншу підбірку маршрутів, йому достатньо просто натиснути на кнопку «SEARCH» знову, що автоматично перезавантажить список маршрутів з новими варіантами, фільтраційні параметри яких є ідентичними до попередніх.

Як можна побачити, використовування застосунку є інтуїтивно зрозумілим та простим. Користувач швидко знайде потрібну інформацію та маршрути.

Адаптивність є дуже важливим аспектом у користуванні веб-сайтом. Даний застосунок є повністю адаптивним під всі екрани користувачів, починаючи від малих смартфонів і закінчуючи великими моніторами, та навіть екранами телевізорів. Важливо зробити досвід користувача максимально успішним. Деякі елементи інтерфейсу потрібно прибрати, а деякі змістити чи редагувати так, щоб все гармонійно виглядало.

Приклади адаптації сторінок для екрану телефону (360 пікселів на 740 пікселів) побачити на рисунках 3.15, 3.16, 3.17 та 3.18.



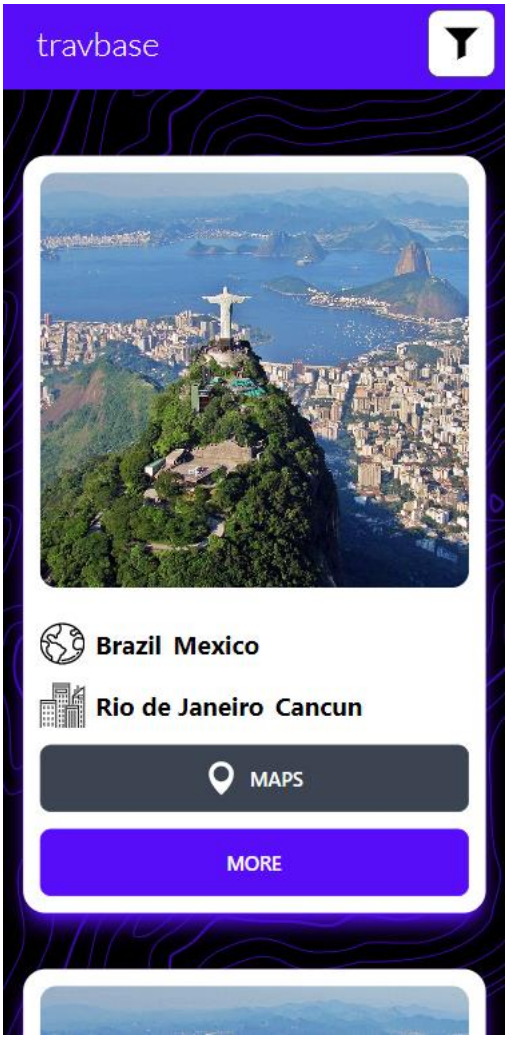


Рисунок 3.15 – Приклад адаптації для екрану телефону сторінки розділу «Home» на домашній сторінці (зліва)

Рисунок 3.16 – Приклад адаптації для екрану телефону сторінки списку маршрутів (справа)

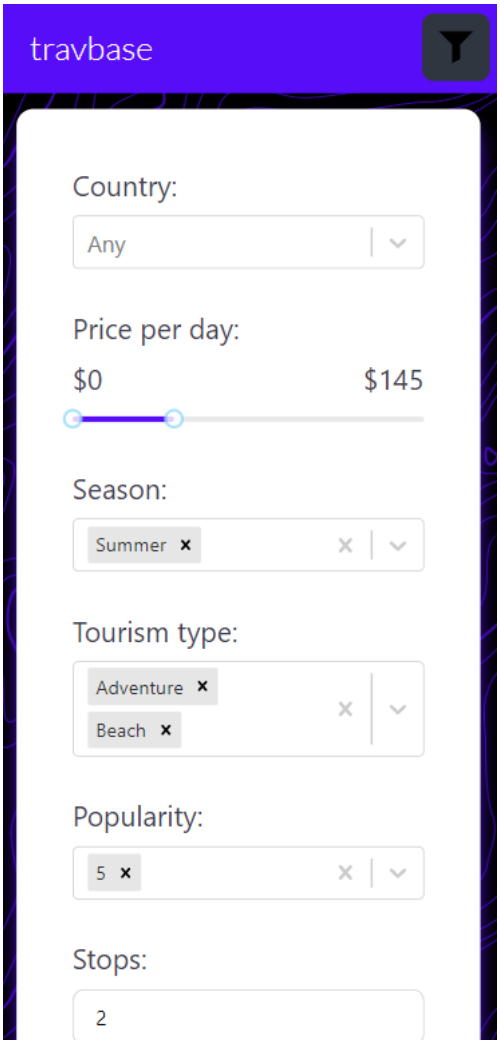


Рисунок 3.17 – Приклад адаптації для екрану телефону фільтраційної форми на сторінці списку маршрутів (зліва)

Рисунок 3.18 – Приклад адаптації для екрану телефону сторінки з детальною інформацією щодо певного маршруту (справа)

Приклади адаптації сторінок для екрану великого монітору (2560 пікселів на 1440 пікселів) побачити на рисунках 3.19 та 3.20.

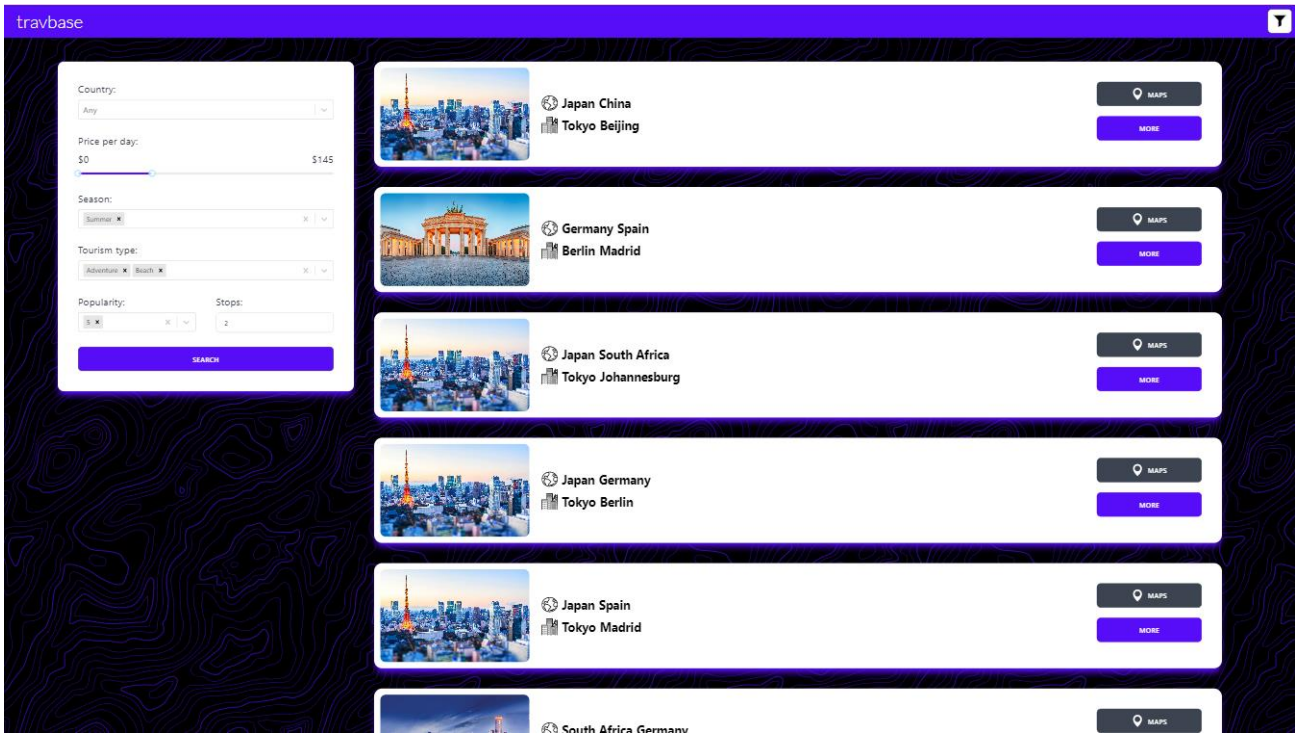


Рисунок 3.19 – Приклад адаптації для екрану великого монітору сторінки списку маршрутів з фільтраційною формою

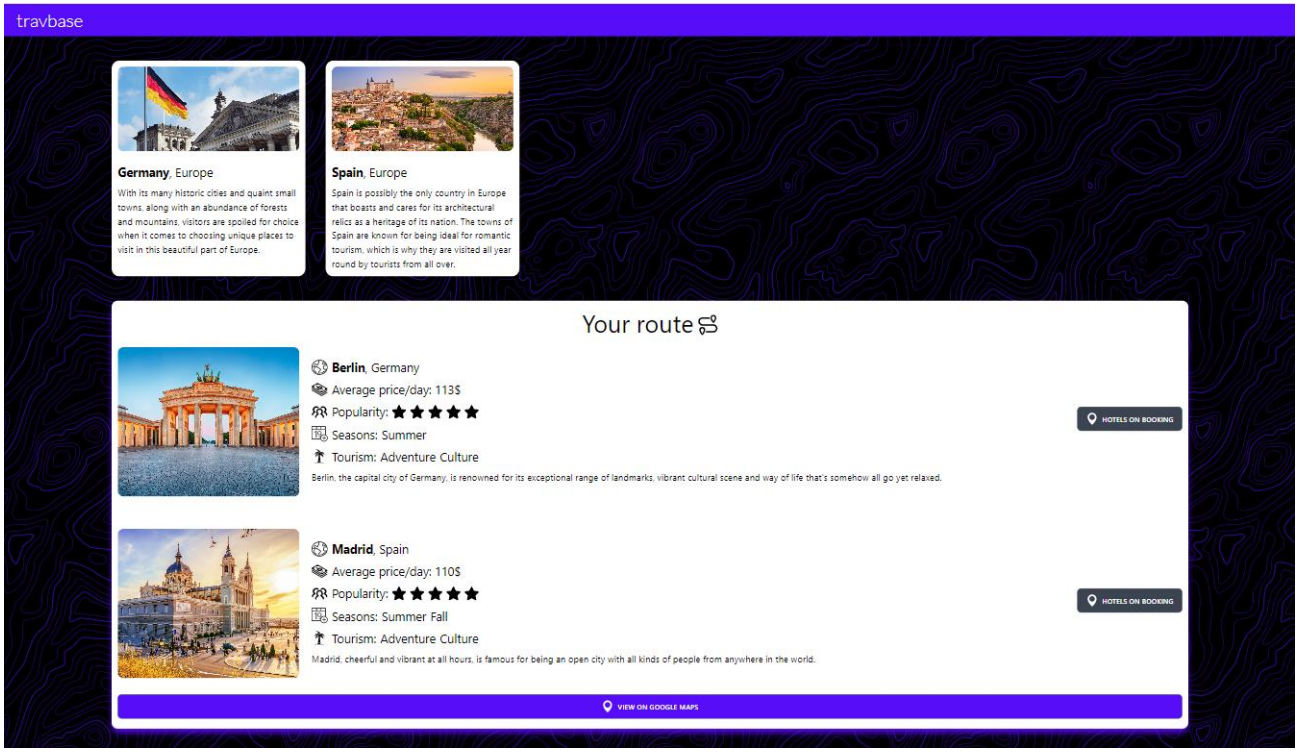


Рисунок 3.20 – Приклад адаптації для екрану великого монітору сторінки з детальною інформацією щодо певного маршруту

Висновки

Під час виконання кваліфікаційної роботи було розроблено алгоритм для пошуку складних маршрутів подорожей з урахуванням вподобань користувача та створено веб-сайт з використанням відповідного алгоритму.

Було проаналізовано сучасні туристичні тенденції та статистичні дані, розглянуто застосунки та системи, що мають суміжний функціонал до застосунку, опрацьовано і знайдено релевантні технології та інструменти для розробки алгоритму та веб-застосунку.

Для розробки клієнтської частини застосунку було обрано мову програмування JavaScript. Для створення веб-інтерфейсів було використано JavaScript-бібліотеку React. Алгоритм, як і вся серверна частина застосунку написана на мові JavaScript. Також, було використано бібліотеку Express для створення сервера застосунку. Було використано багато інших інші допоміжних бібліотек з менеджеру пакетів «Node Package Manager». У ролі бази даних використано документо-орієнтовану базу даних MongoDB. Як середовище розробки було обрано саме WebStorm від компанії JetBrains. Клієнтська частина застосунку розгорнута на платформі Netlify, а серверна частина, на платформі Vercel.

Веб-застосунок надає можливість користувачу шукати складні маршрути для мандрівок за допомогою фільтрів, у спеціальній формі вводу на веб-сайті, яка містить декілька параметрів та у результаті переглядати маршрути. Маршрути, які отримує користувач, складаються з переліку міст, додаткового опису та корисних посилань, які як найкраще підходять під визначений пошук.

Дизайн веб-сайту – мінімалістичний, сучасний, з приємною гаммою кольорів, що слідує новим тенденціям та виділяє важливі елементи. Інтерфейс – інтуїтивно зрозумілий та має адаптивність для різного розширення екрану користувача.

Під час розробки бралося до уваги зручність та зрозумілість використання застосунку, приділялася увага до теоретичних аспектів побудови якісного, сучасного та корисного застосунку з унікальною ідеєю та алгоритмом, що вирішує головне завдання – надання користувачу найрелевантніших маршрутів. Алгоритм розділений на сім умовних кроків, що дає змогу за потреби модифікувати його у майбутньому для ускладнення логіки.

Список використаних джерел

1. «Our World in Data». Tourism by Max Roser and Bastian Herre [Електронний ресурс] https://ourworldindata.org/tourism
2. «United Nations World Tourism Organization». INTERNATIONAL TOURISM AND COVID-19 [Електронний ресурс] https://www.unwto.org/tourism-data/international-tourism-and-covid-19
3. «Where and When» [Електронний ресурс] https://www.whereandwhen.net
4. «Similarweb». whereandwhen.net: Трафік та залученість [Електронний ресурс] https://www.similarweb.com/website/whereandwhen.net/#traffic
5. «Google Maps» [Електронний ресурс] https://www.google.com.ua/maps
6. «Booking.com». [Електронний ресурс] https://www.booking.com
7. «ПОНЯТТЯ, СТРУКТУРА ТА РІЗНОВИДИ ВЕБ-САЙТІВ. АВТОМАТИЗОВАНЕ РОЗРОБЛЕННЯ ВЕБ-САЙТІВ». [Електронний ресурс] http://www.ndu.edu.ua/liceum/web.pdf
8. «MDN Web Docs». [Електронний ресурс] https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Document_and_website_structure
9. «Hubspot». 10 Best Blue Websites in 2022 [Електронний ресурс] https://blog.hubspot.com/website/blue-website
10. «MDN Web Docs». [Електронний ресурс] https://developer.mozilla.org/en-US/docs/Web/JavaScript
11. «React». Посібник: знайомство з React [Електронний ресурс] https://uk.reactjs.org/tutorial/tutorial.html
12. «Redux». Redux Overview and Concepts [Електронний ресурс] https://redux.js.org/tutorials/essentials/part-1-overview-concepts
13. «Node.js». About Node.js [Електронний ресурс] https://nodejs.org/en/about/
14. «MongoDB» MongoDB vs. MySQL Differences [Електронний ресурс] https://www.mongodb.com/compare/mongodb-mysql
15. «WebStorm». [Електронний ресурс] https://www.jetbrains.com/webstorm

16. «Netlify». Welcome to Netlify [Електронний ресурс] https://docs.netlify.com/
17. «Vercel». Introduction to Vercel [Електронний ресурс] https://vercel.com/docs