Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

РОЗРОБКА ВЕБ-СЕРВІСУ ДЛЯ ПЛАНУВАННЯ ПОДОРОЖЕЙ

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення»

Керівник курсової роботи:
к. ф.-м. н. Гречко А. В.

Виконала студентка:
Кірдяєва О.О.

Київ 2020

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри інформатики,

к. ф.-м. н. С. С. Гороховський

_____

(підпис)

“____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентки Кірдяєвої Ольги Олександрівни факультету інформатики 3 курсу

Тема: Розробка веб-сервісу для планування подорожей

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Календарний план

Зміст

Перелік умовних позначень

Вступ

Розділ 1 Аналіз предметної області. Постановка завдання курсової роботи

Розділ 2 Теоретичні відомості

Розділ 3 Опис реалізації програмного продукту

Висновки

Перелік використаних джерел

Дата видачі “___” _____ 2020 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання курсової роботи
Тема: Розробка веб-сервісу для планування подорожей

Календарний план виконання роботи:

№ п/п	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	10.10.2019	
2.	Ознайомлення з існуючою інформацією по темі	17.01.2020	
3.	Ознайомлення з існуючими системами-аналогами роботи	01.05.2020	
4.	Початок створення практичної частини	05.05.2020	
5.	Початок написання теоретичної частини	06.05.2020	
6.	Подання проміжної версії практичної частини	10.05.2020	
7.	Аналіз практичної частини; її корегування	10.05.2020	
8.	Остаточне завершення написання теоретичної частини роботи та розробки практичної частини; корегування	11.05.2020	
9.	Створення презентації	11.05.2020	
10.	Захист курсової роботи	20.05.2020	

Студент Кірдяєва О.О.

Керівник Гречко А.В.

“_____” _____

Зміст

Перелік термінів та умовних позначень

Фреймворк – це програмна платформа, що визначає структуру системи і полегшує розробку програмного забезпечення.

Нереляційна база даних – це така реалізація бази даних, яка не використовує у своїй структурі реляції та дозволяє більш гнучко обробляти дані.

Пакетний менеджер – це програма, що дозволяє ефективно керувати встановленням та налаштуванням програмного забезпечення

Шаблонізатор – це програмне забезпечення, яке дозволяє за допомогою шаблонів зручно відділяти рівень представлення від інших рівнів.

Вступ

PlanYourTrip – це простий у використанні веб-сервіс для планування подорожей, що буде корисним для будь-якої людині, яка любить подорожувати. Він дозволяє за допомогою лаконічного інтерфейсу створювати автомобільні маршрути із багатьма пунктами призначення і ділитись враженнями щодо подорожі із іншими користувачами.

Актуальність обраної теми полягає в тому, що, незважаючи на значну кількість різноманітних аналогів сервісу, важко знайти такий, що задовольнив би потреби кожного мандрівника.

Мета курсової роботи – допомогти мандрівникам планувати маршрути своїх подорожей, забезпечити єдине місце для зберігання пунктів призначення, надати можливість вести власний щоденник подорожей і ділитись враженнями з іншими користувачами.

Завдання курсової роботи – створити такий сервіс для планування подорожей, що задовольнятиме основні потреби мандрівників і буде простим та зручним у використанні.

Об’єктом дослідження є питання планування подорожей та побудови маршрутів. Предметом дослідження є питання наявності на ринку актуальних сервісів та створення власної альтернативи.

Практичне значення одержаних результатів дослідження полягає в тому, що аналіз наявних сервісів дозволяє врахувати їхні переваги та недоліки і на основі цих даних розробити власний сервіс, що буде поєднувати в собі сильні сторони наявних платформ.

Для створення веб-сервісу PlanYourTrip було використане таке програмне забезпечення:

Мова програмування – JavaScript, із використанням програмної платформи Node.js і фреймворку Express для написання серверної частини.

Усі дані зберігаються у нереляційній базі даних MongoDB із використанням бібліотеки Mongoose, що дозволяє створювати моделі даних для кращої структури коду.

Для встановлення та структурування пакетів був використаний пакетний менеджер NPM. Представлення відбувається за допомогою шаблонізатора Pug і клієнтського Javascript включно з бібліотекою jQuery. За візуальну складову відповідають фреймворк Bootstrap і стилі CSS.

Робота із картами була реалізована за допомогою Google Maps API.

Курсова робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАННЯ КУРСОВОЇ РОБОТИ

1.1. Аналіз сучасного стану питання та обґрунтування теми.

Як зазначено у статті [1], на сьогоднішній день індустрія туризму перебуває у кризовому стані. У зв’язку з останніми подіями люди повністю відмовились

від подорожей заради збереження власної безпеки. Тим не менш, незважаючи на поточну ситуацію у світі, подорожі все одно залишатимуться актуальною галуззю, коли загроза підхопити коронавірус піде нанівець. Можна прогнозувати, що після тривалого перебування вдома люди будуть переповнені жагою мандрувати, і тому, щойно карантинні заходи будуть послаблені, слід очікувати на значний спалах подорожей у всьому світі. Цьому посприяють також знижки на авіаперельоти та проживання у готелях, які зараз активно запроваджують різні компанії, щоб не позбутись прибутків. У зв'язку з цим ринок туризму може сильно змінитись, мандрівники надаватимуть перевагу маршрутам із вигідними цінами і, як наслідок, досліджуватимуть нові напрямки. Окрім того, із високою ймовірністю підвищиться попит на подорожі всередині країни. Отже, як наслідок усіх вищезазначених факторів, сервіси для планування подорожей у найближчому майбутньому мають значні шанси опинитись на піку популярності.

На сьогоднішній день існує багато засобів для планування подорожей. Найбільш яскравими представниками цього напряму є такі сервіси:

1) Google Trips
2) Waytips
3) Youroute
4) Tropinki
5) Triphearts

Перераховані сервіси будуть детальніше розглянуті у наступному підпункті.

1.2. Огляд існуючих аналогів розробки.

На жаль, більшість альтернативних сервісів не доступні українською мовою, але, тим не менш, вони є доцільними прикладами, адже вдало ілюструють основні функції.

1.2.1. Google Trips [2]

Мабуть, найвідоміший сервіс серед інших, адже його розробником є корпорація Google. На основній сторінці (рис. 1-2) можна переглянути основні функції сервісу:

1) Пропозиції актуальних маршрутів
2) Авіарейси
3) Готелі
4) Створення подорожі

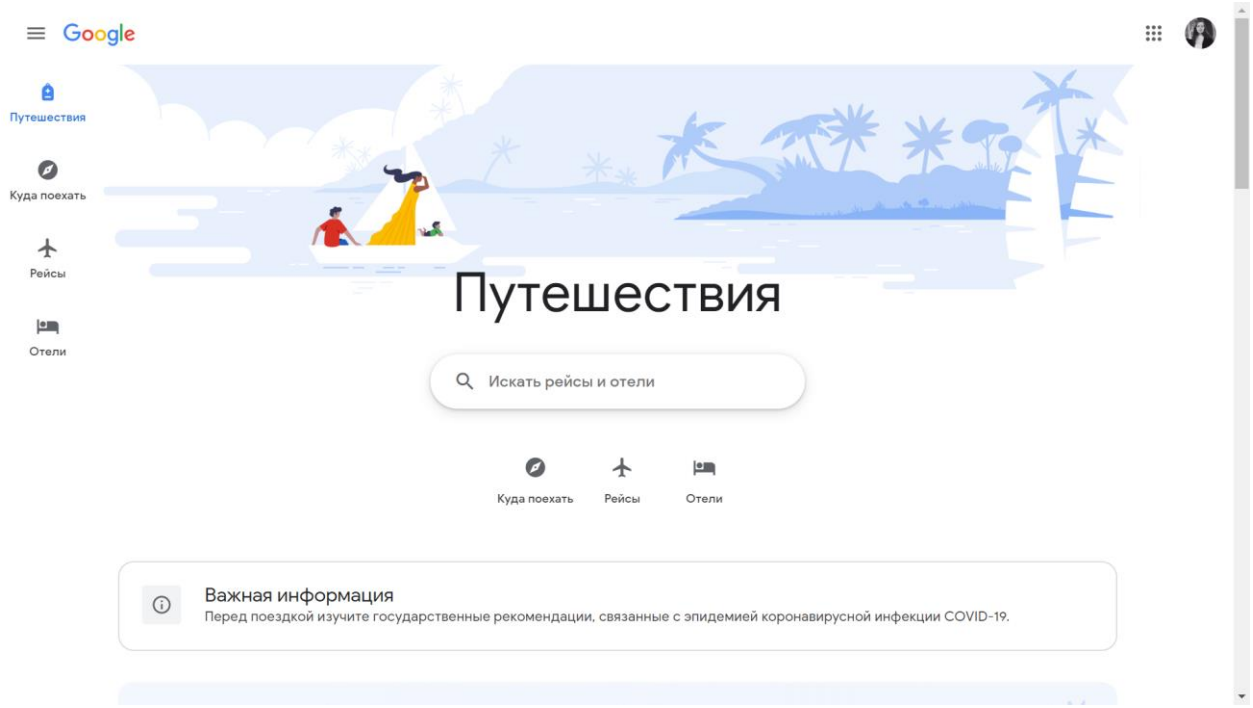

Рис. 1. Основна сторінка

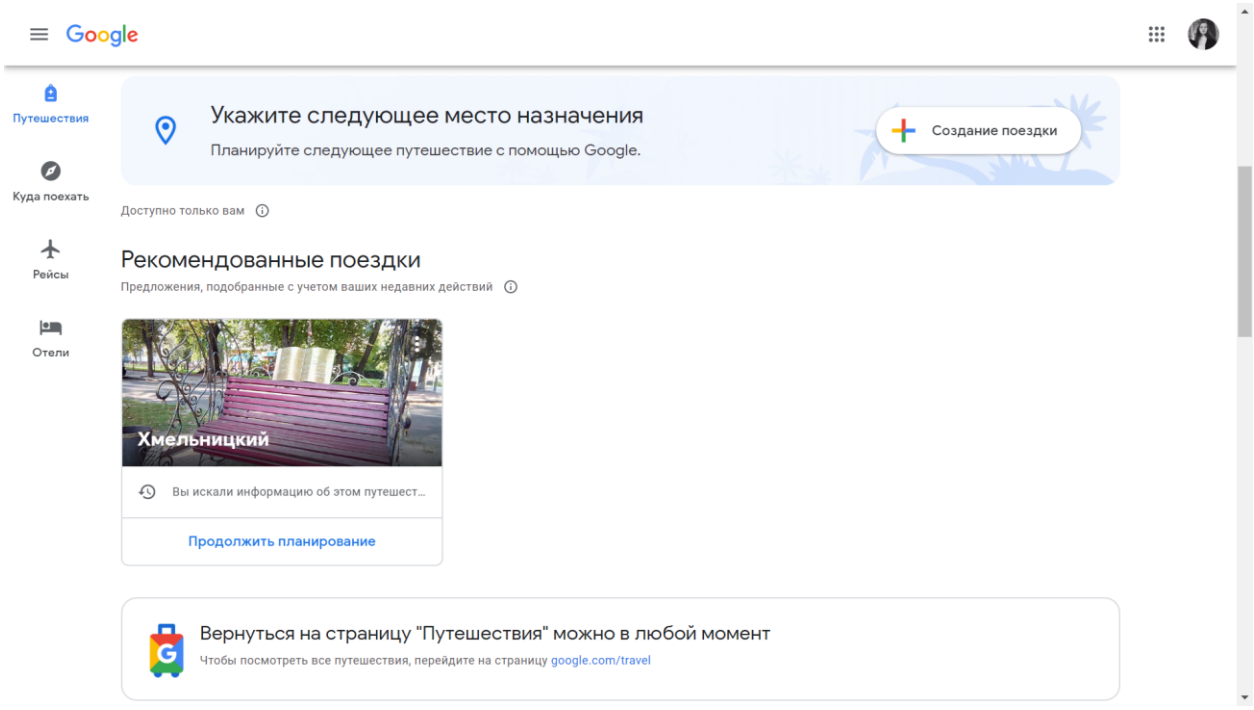

Рис. 2. Рекомендації

На рис. 3 зображено вікно із формою для створення подорожі. Користувач має обрати міста і дати своєї мандрівки.

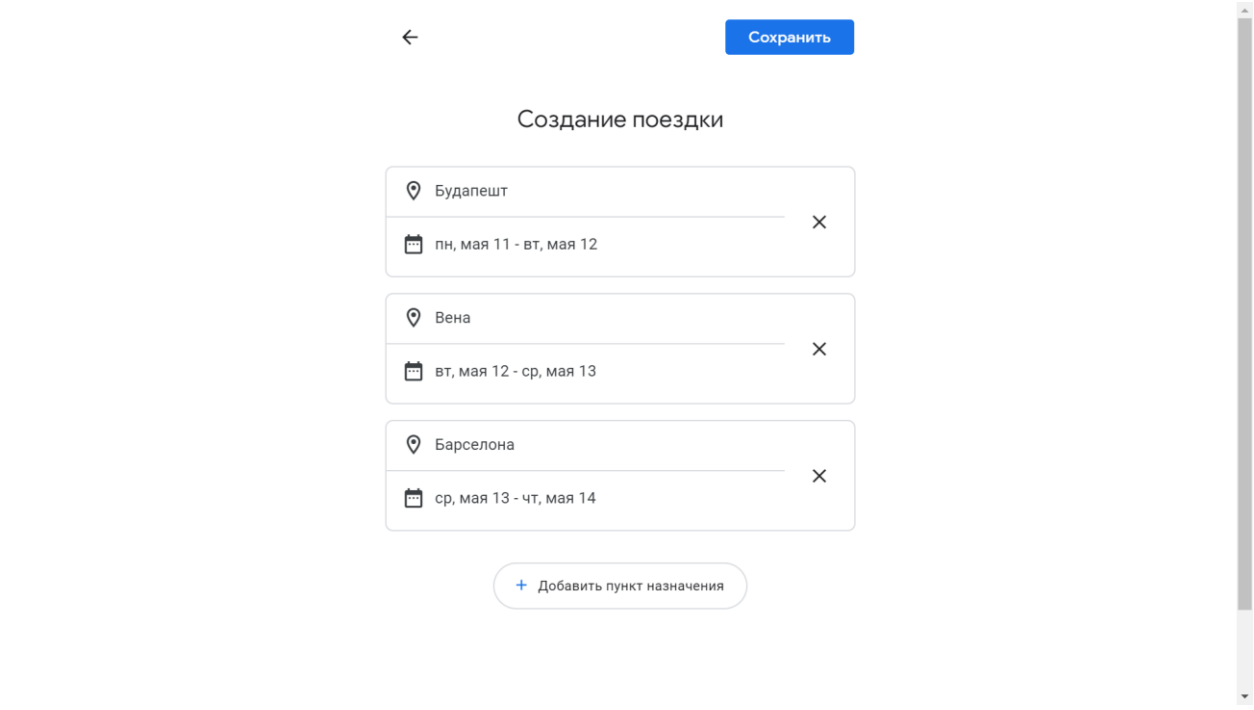

Рис. 3. Створення подорожі

На рис. 4-5 можна побачити готову подорож із переліком обраних місць, пропозиціями щодо відвідувань та можливістю одразу забронювати готелі.

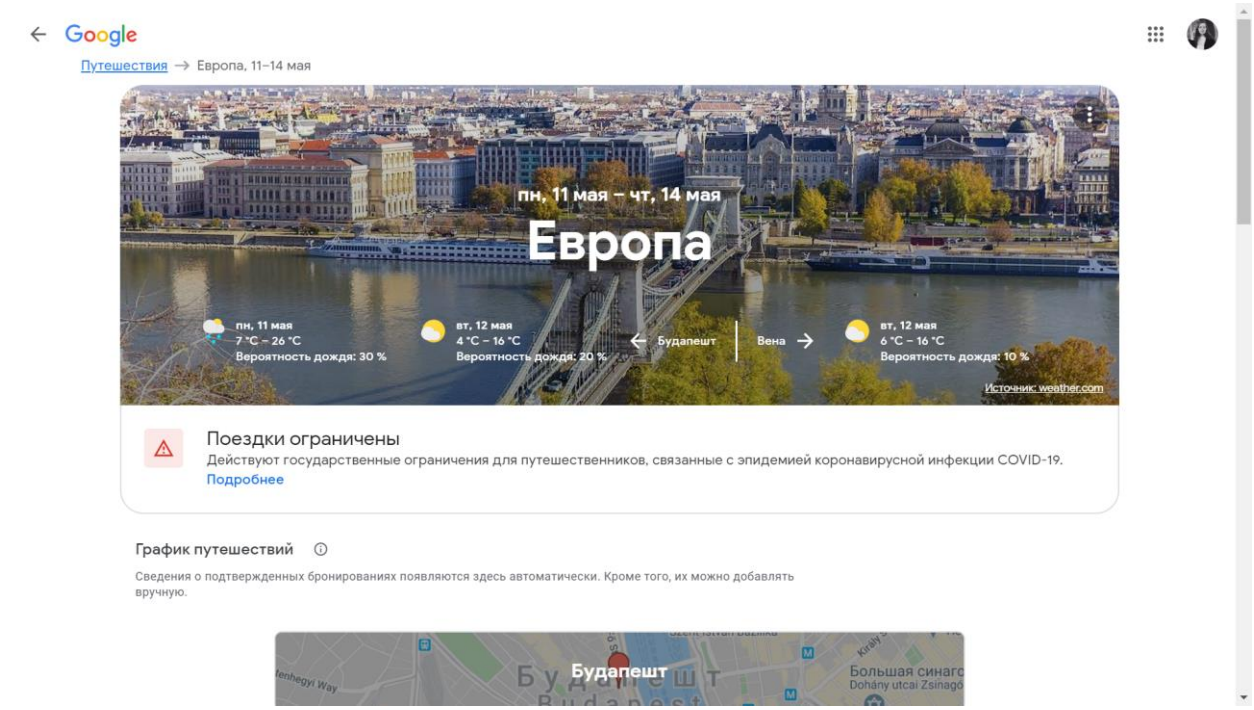

Рис. 4. Сторінка подорожі

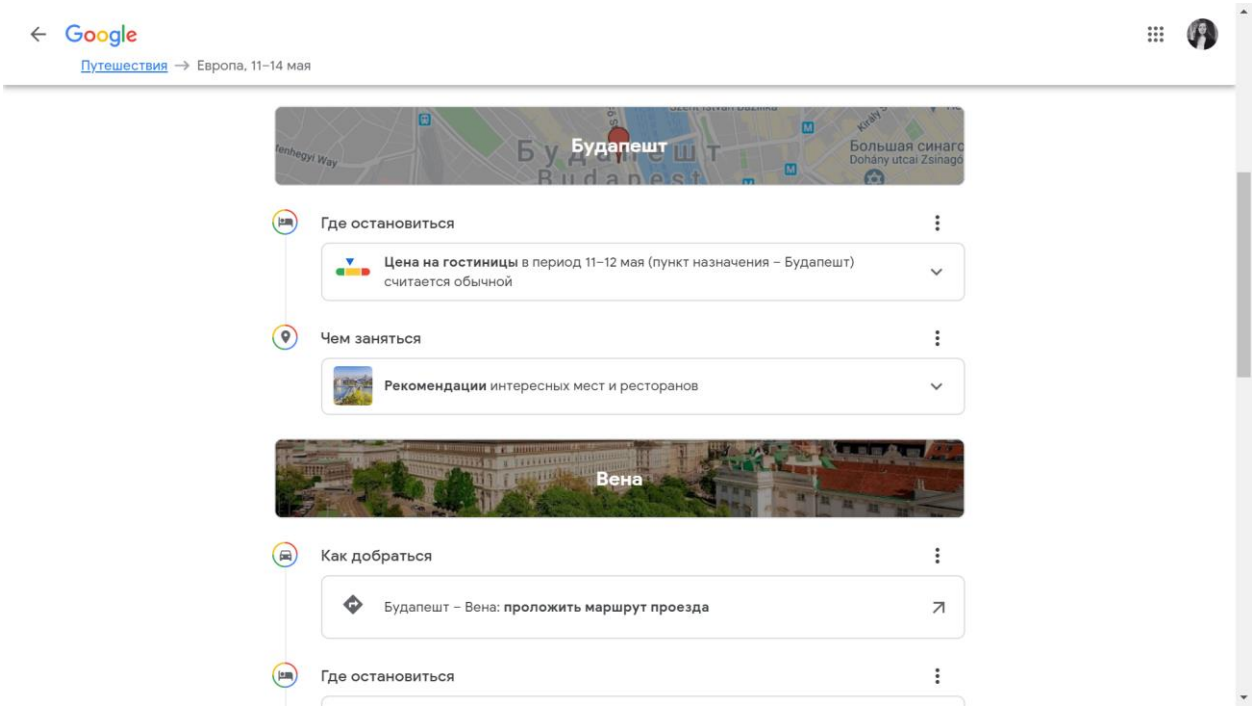

Рис. 5. Перелік пунктів призначення

1.2.2. Waytips [3]

Цей сервіс не настільки відомий, але він поєднує у собі декілька можливостей: він дозволяє користувачам створювати маршрути, переглядати інформацію про різні міста та пам’ятки, бронювати авіаквитки, і при цьому має соціальну складову, адже користувачі можуть створювати пости із враженнями.

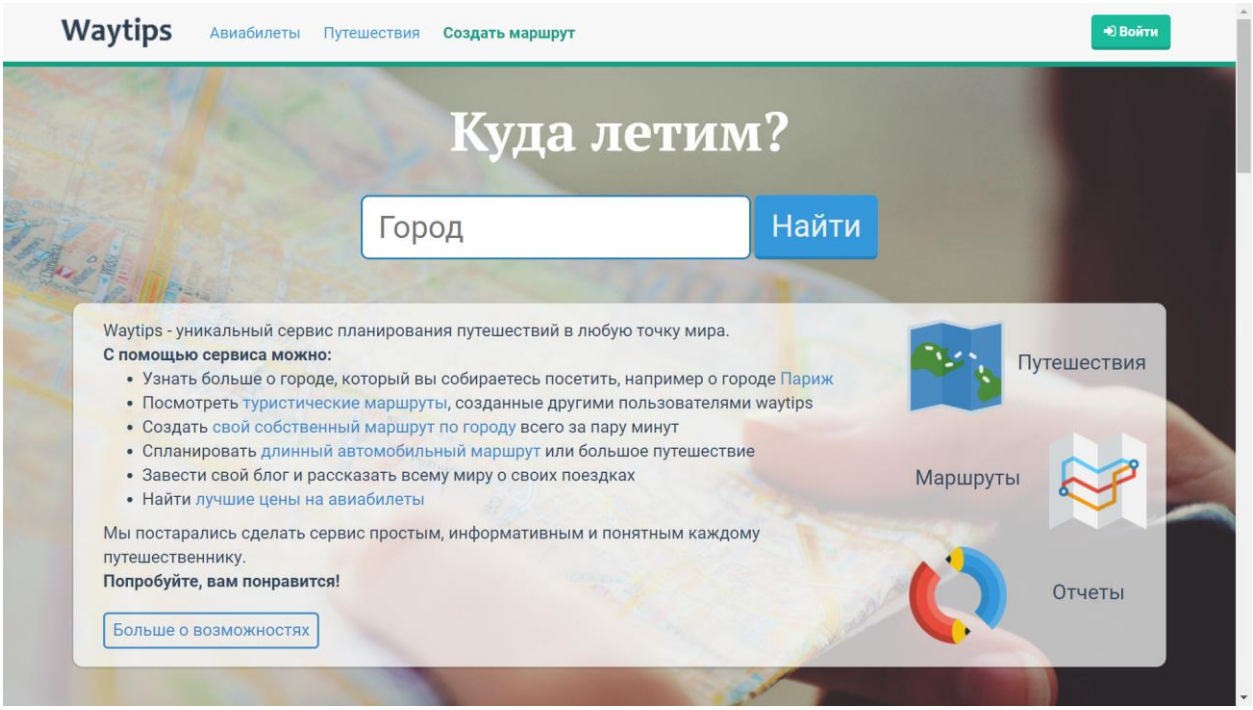

Рис. 6. Основна сторінка

На основній сторінці (рис. 6-8) розташовані всі необхідні посилання та вікна.

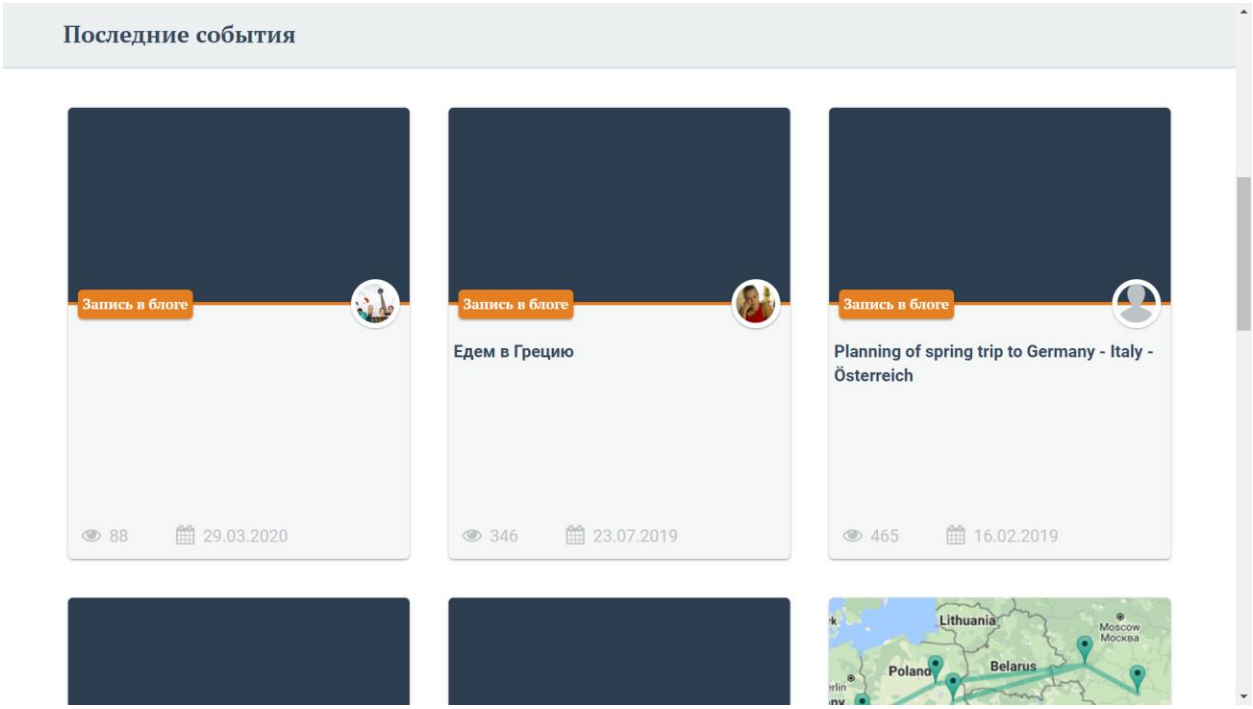

Рис. 7. Пости і маршрути користувачів

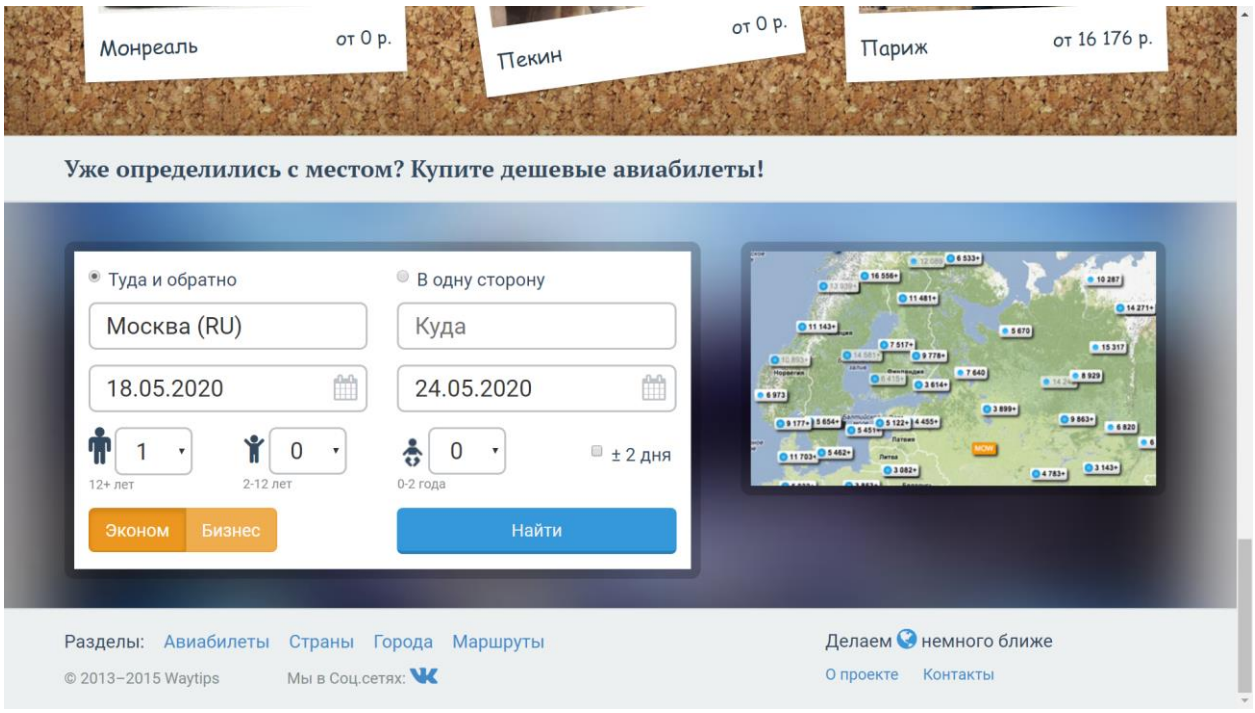

Рис. 8. Авіабілети

Тим не менш, сайт містить декілька недоліків. Він і має достатньо простий та зрозумілий інтерфейс, але не всі посилання розташовані інтуїтивно. Також він має проблеми із Google Maps API, адже мапи відображаються некоректно

(рис.9, 15). На рис. 9-11 можна побачити створений деяким користувачем маршрут, його деталізацію та поле для коментарів.

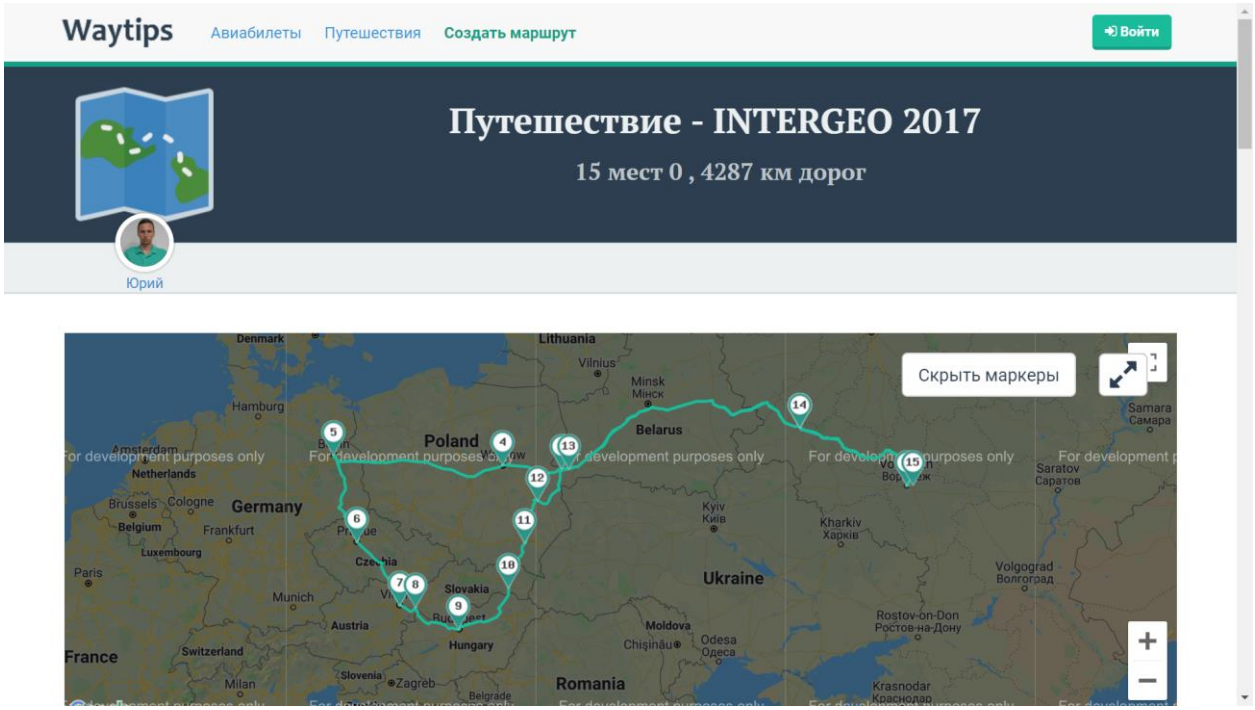

Рис. 9. Приклад створеного маршруту

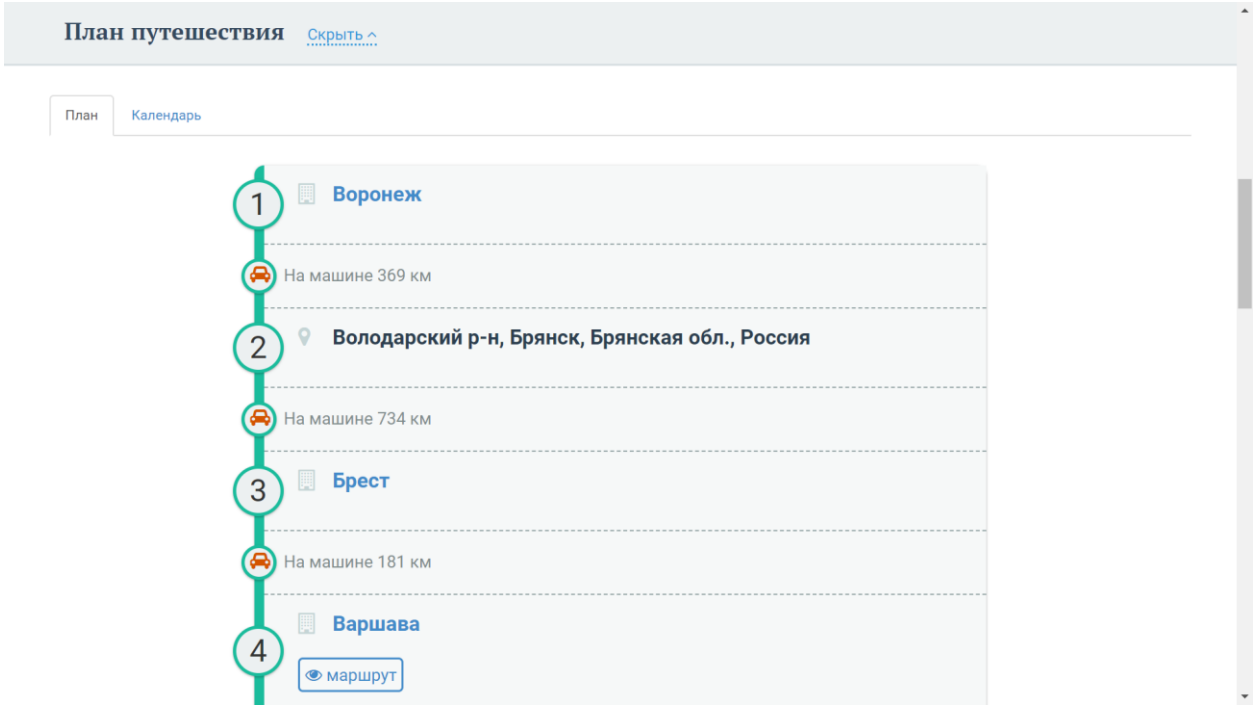

Рис. 10. Деталізація маршруту

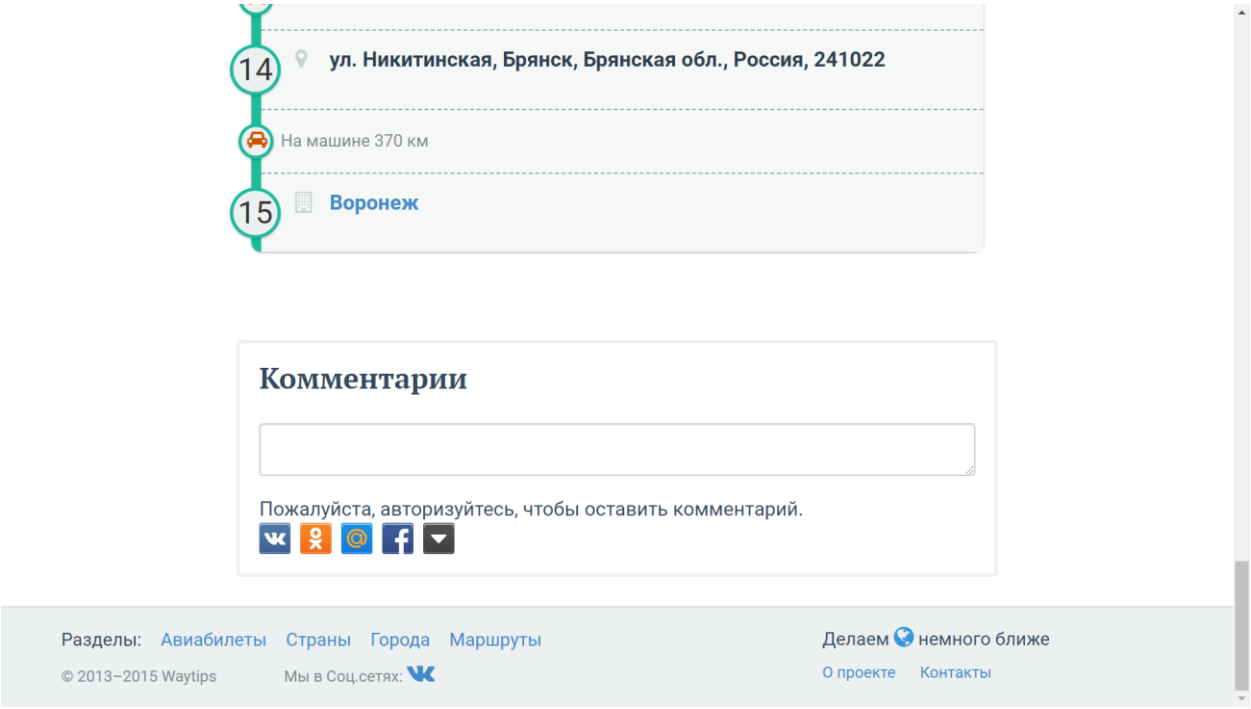

Рис. 11. Коментарі

На рис. 12 видно результат натискання кнопки «Создать маршрут». Користувач має обрати одне місто, а після цього сервіс запропонує варіанти культурних пам'яток, які варто відвідати (рис.13-15).

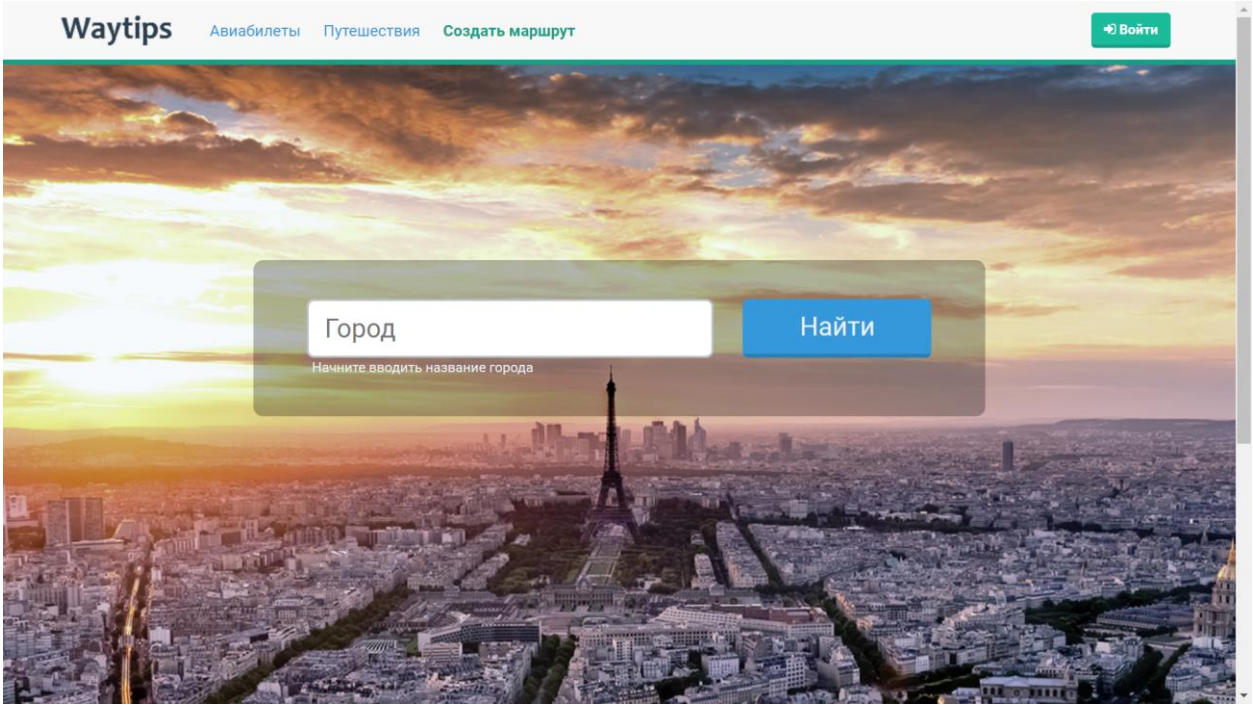

Рис. 12. Пошук місця призначення

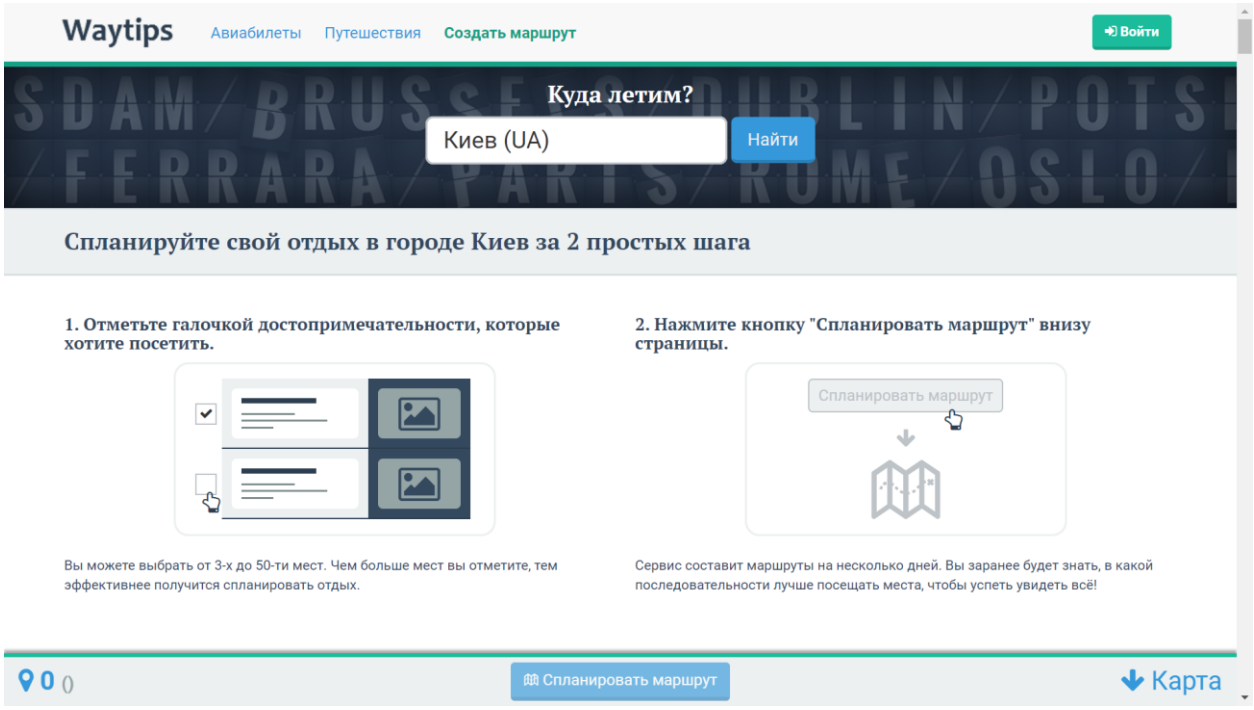

Рис. 13. Планування подорожі

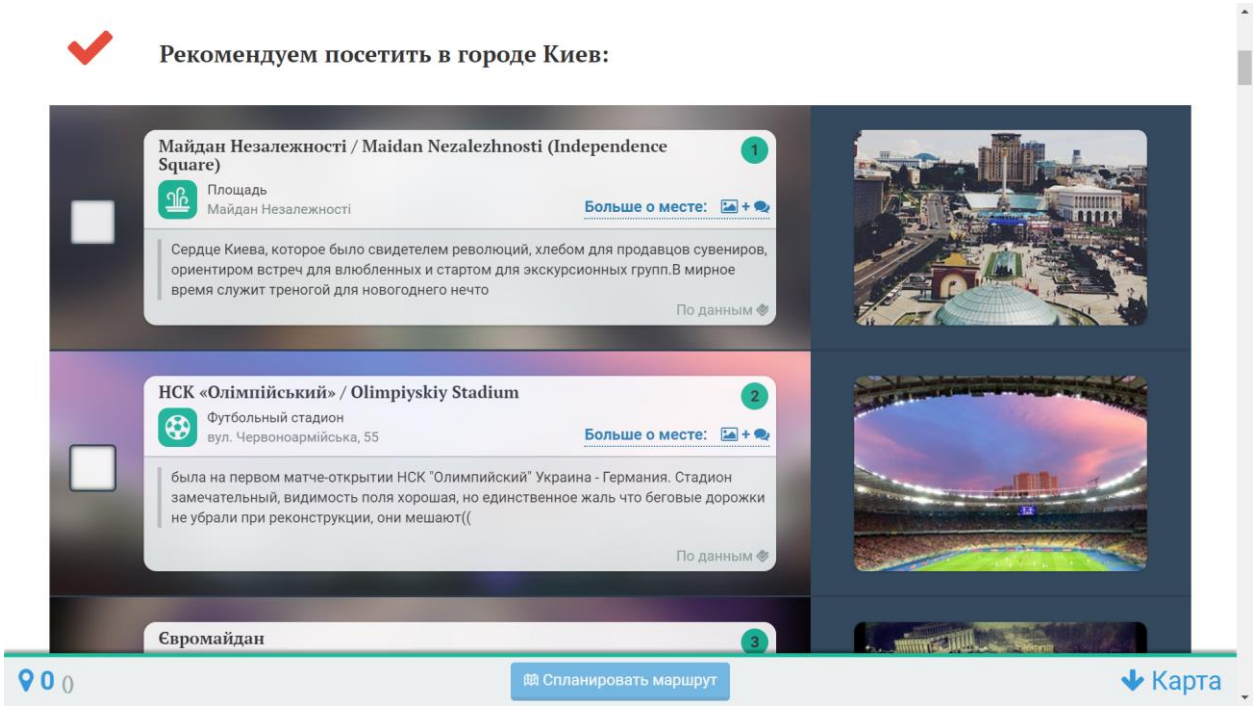

Рис. 14. Туристичні пропозиції

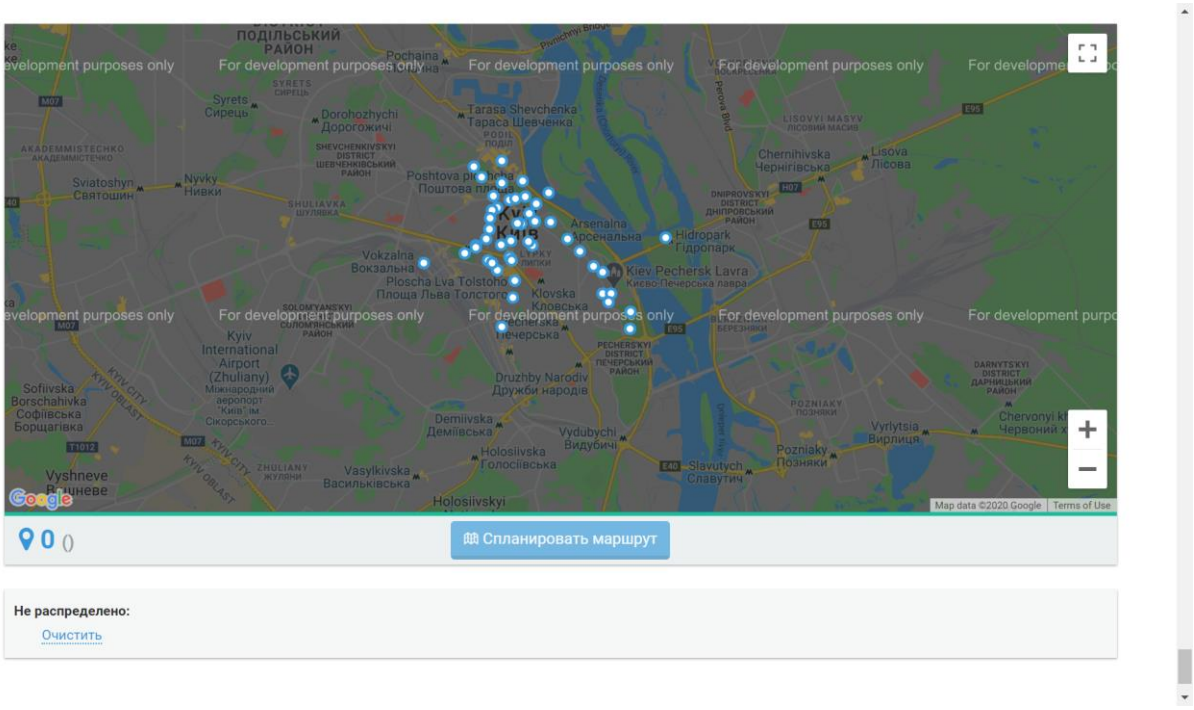

Рис. 15. Туристичні місця на карті

1.2.3. Youroute [4]

Ще один сервіс для планування подорожей із досить обширними можливостями. На основній сторінці (рис. 16) розташовані посилання на різні можливості, але не всі з них робочі. Посилання «Авиабилеты» відкриває пусту сторінку, а посилання «Отели» перенаправляє на сайт Booking.com [5].

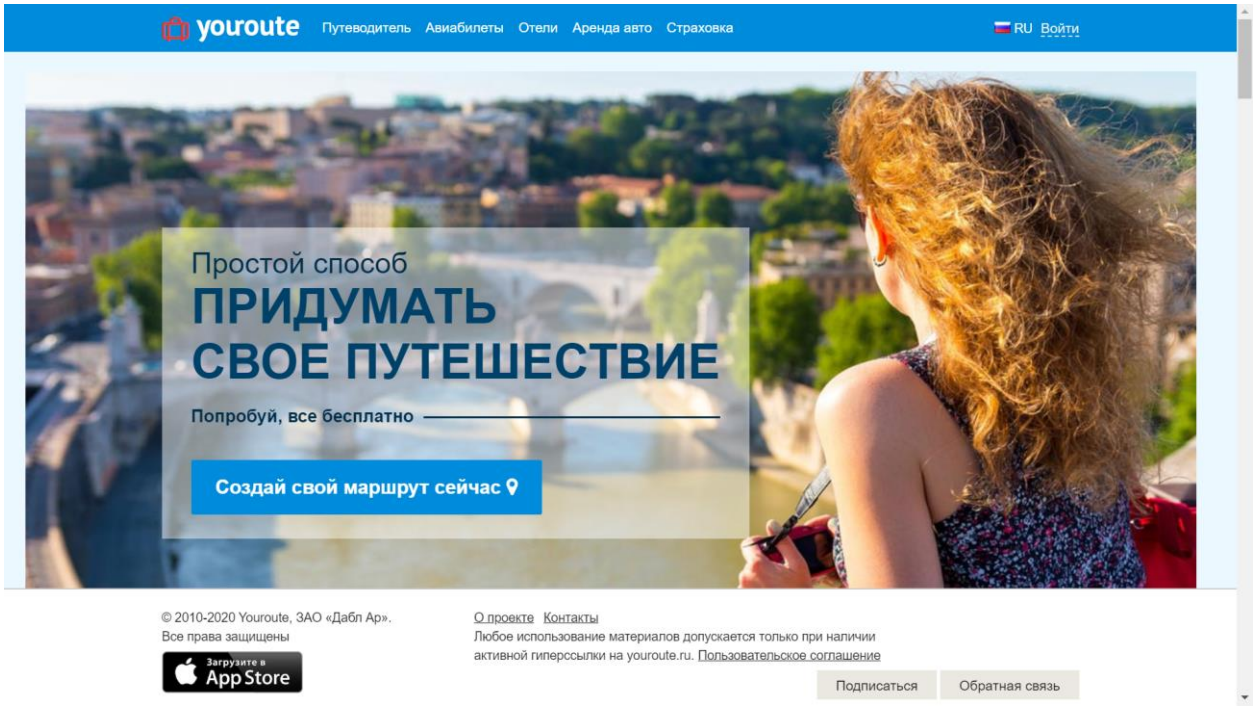

Рис. 16. Основна сторінка

На рис. 17-19 показані основні етапи побудови маршруту. На рис. 20 зображений результат пошуку за містом із найцікавішими культурними пам’ятками.

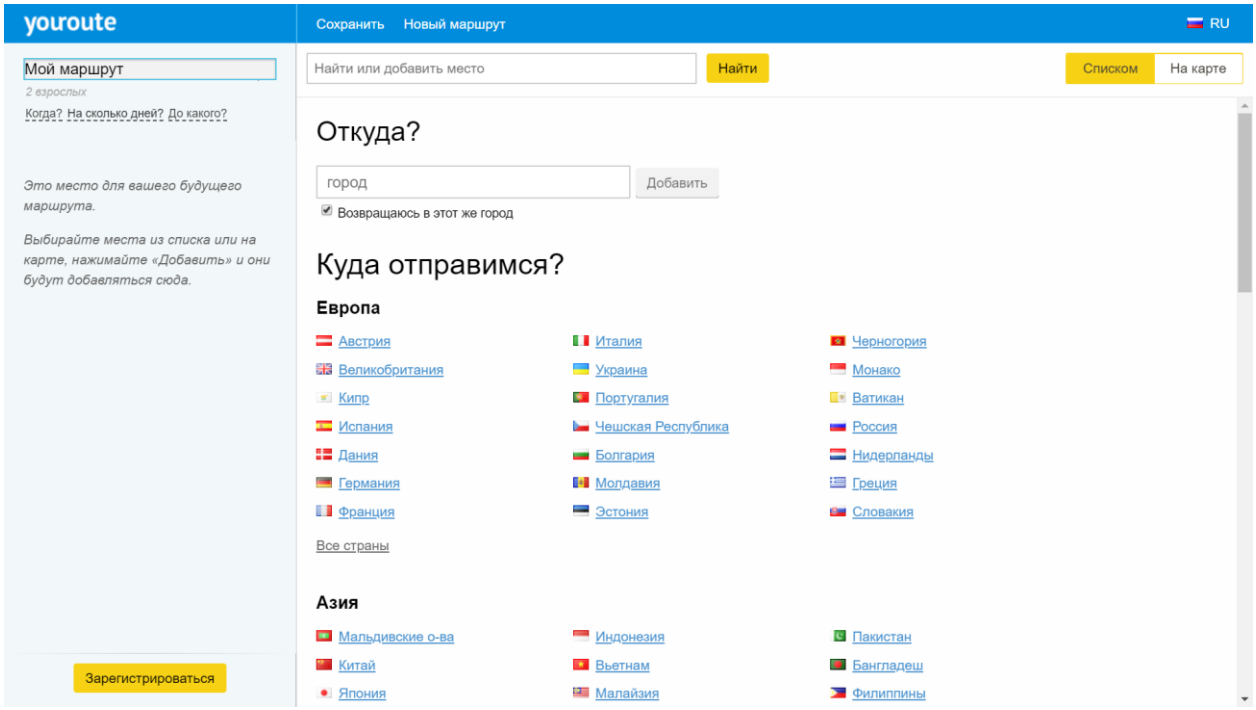

Рис. 17. Конструктор маршрутів

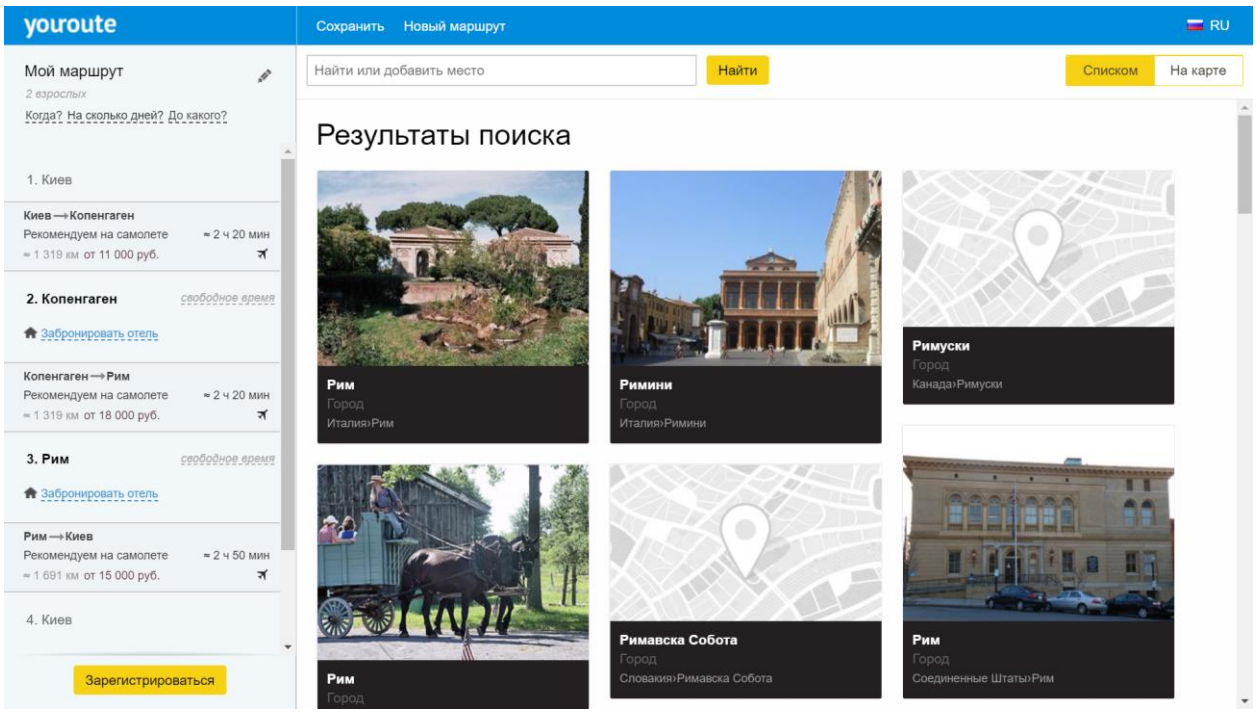

Рис. 18. Результати пошуку

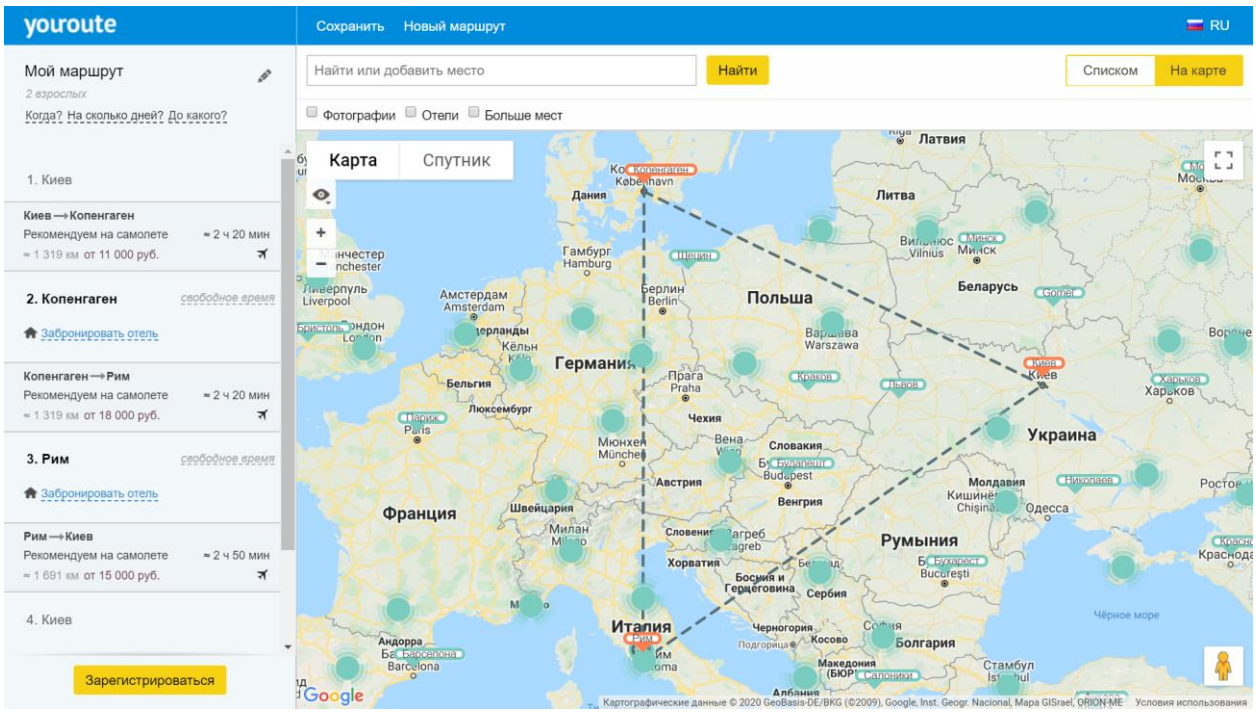

Рис. 19. Прямий маршрут на мапі

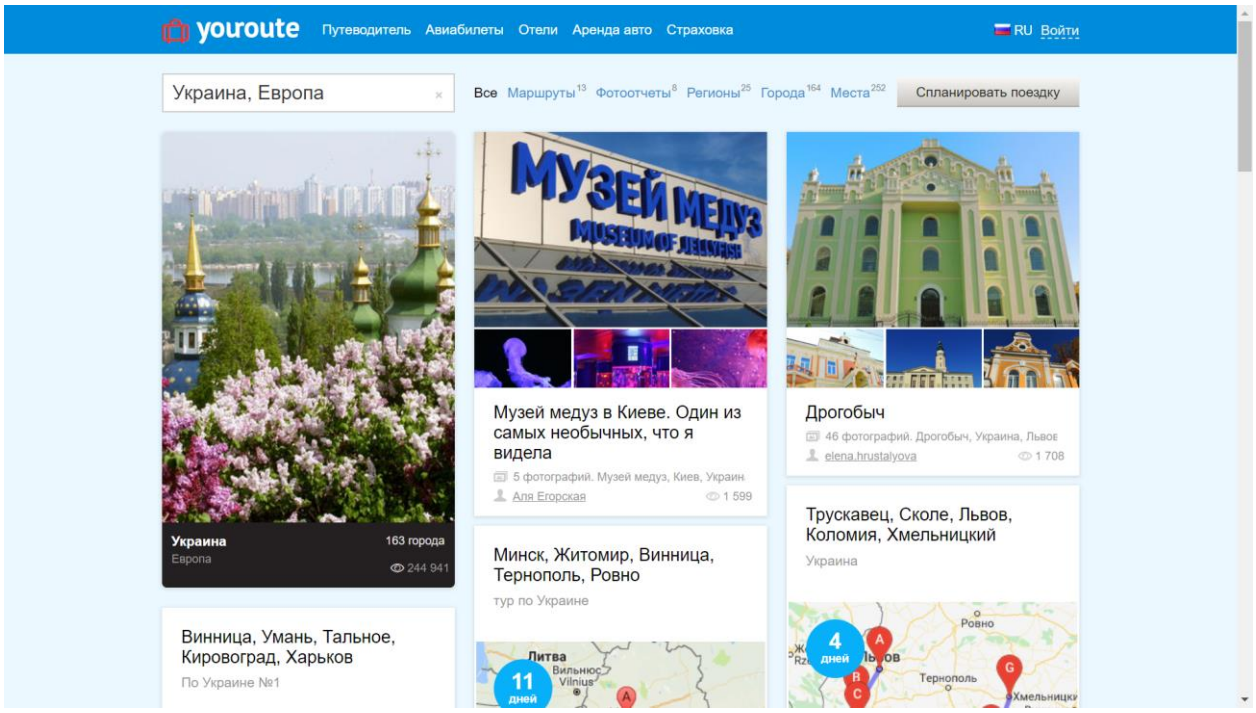

Рис. 20. Цікаві місця

1.2.4. Tropinki [6]

Вузькоспеціалізований сервіс, що пропонує лише створення маршруту подорожі, зате містить багато параметрів. У першу чергу розрахований на мандрівників, що люблять ходити в походи. На рис. 1 зображена основна сторінка веб-сайту. На рис. 2 зображена сторінка із генерацією маршруту.

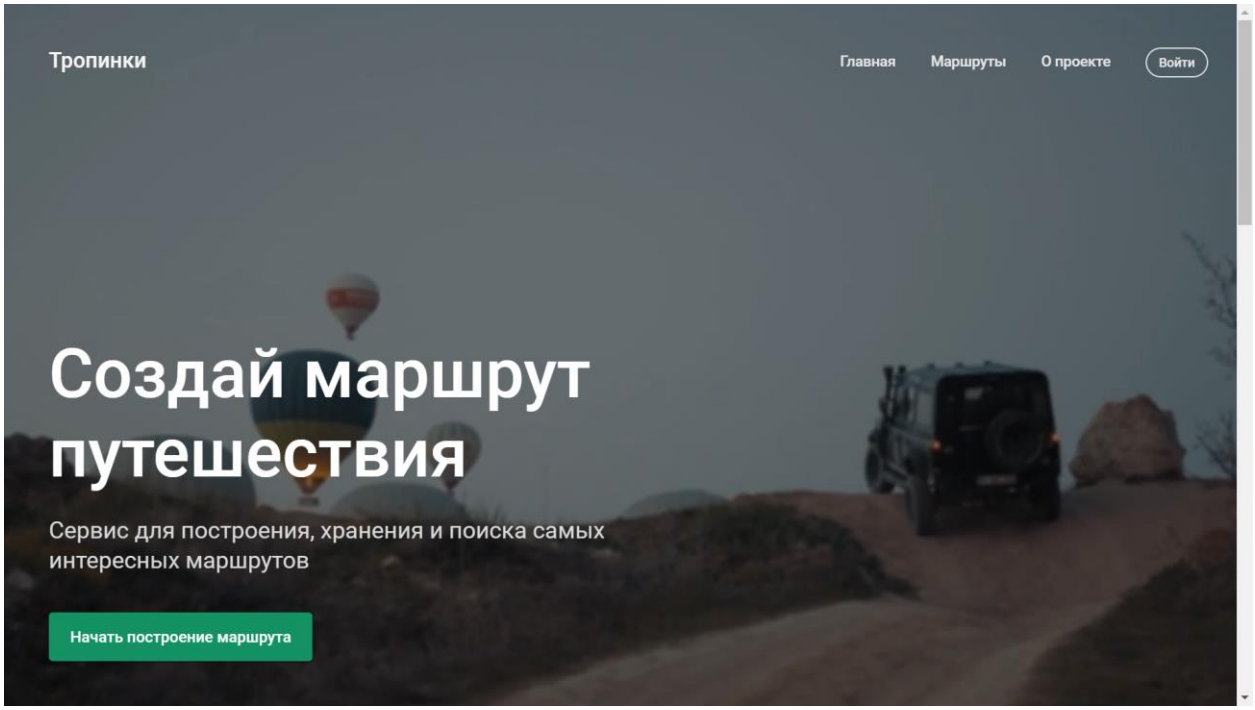

Рис. 21. Основна сторінка

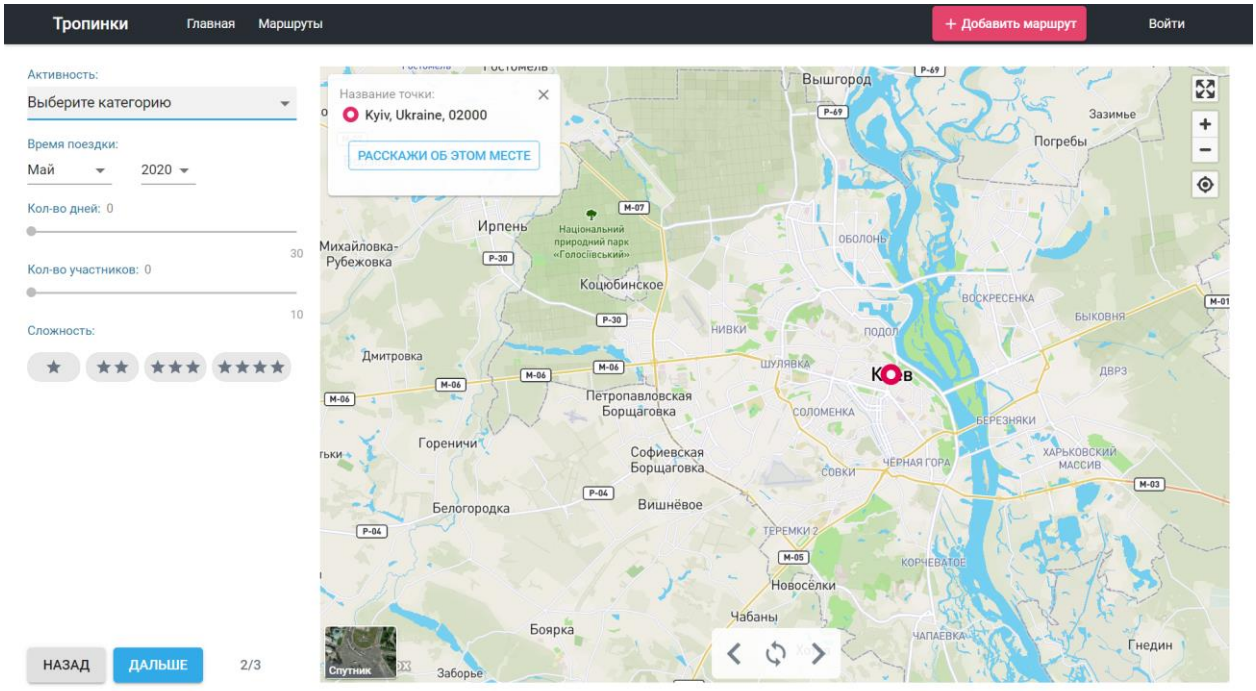

Рис. 22. Побудова маршруту

1.2.5. Triphearts [7]

Дуже зручний та приємний у використанні сайт, що також доступний українською мовою (рис. 23). Дозволяє створювати авіамаршрути між різними

містами а також зберігати на мапі історію подорожі разом із нотатками (рис.24-26).

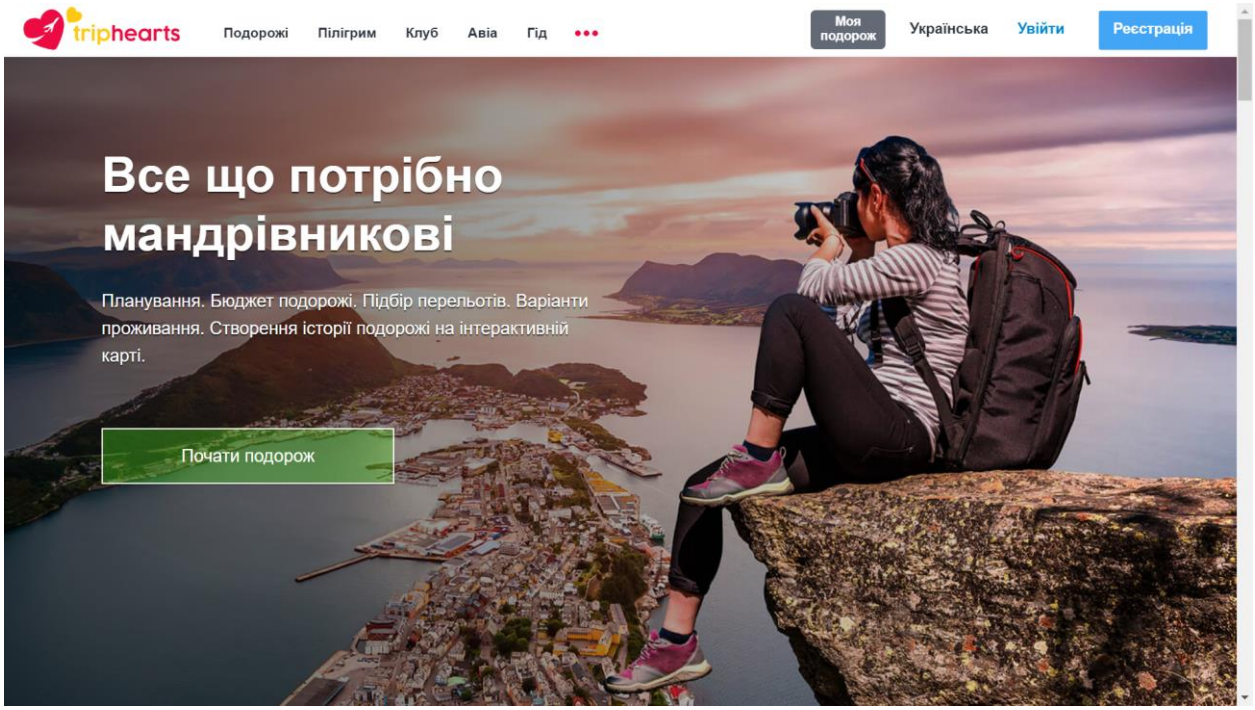

Рис. 23. Основна сторінка

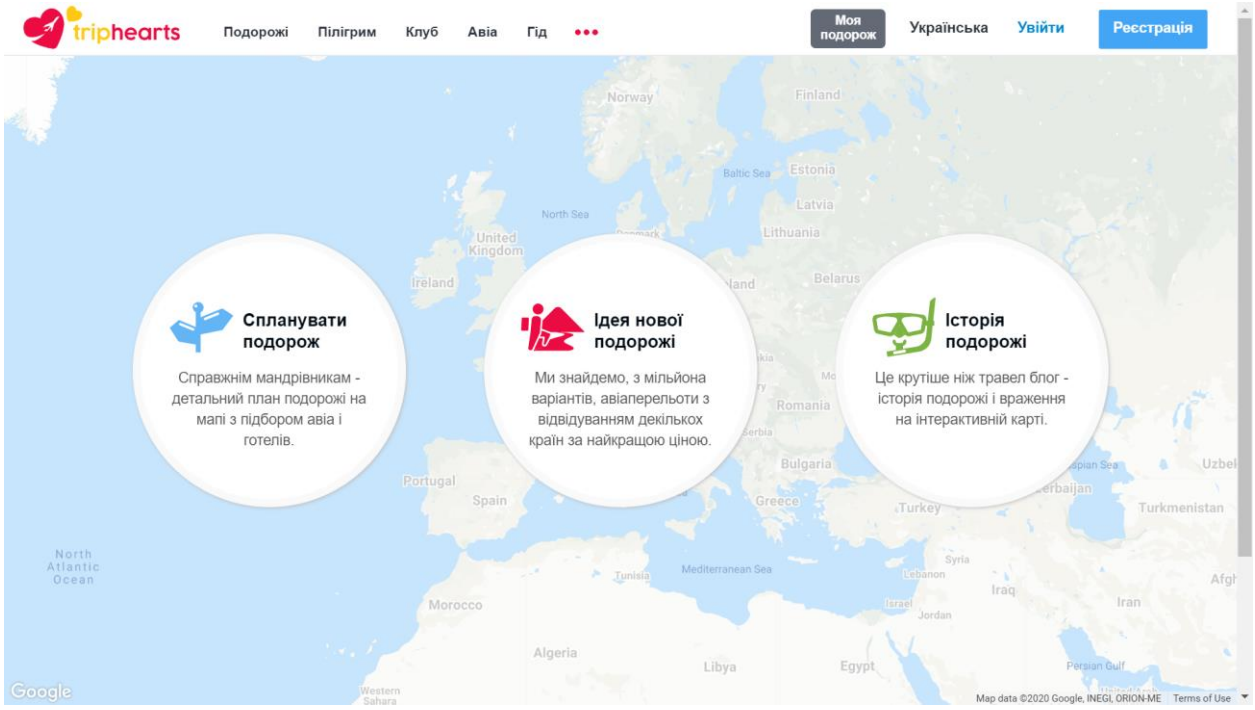

Рис. 24. Перелік функцій

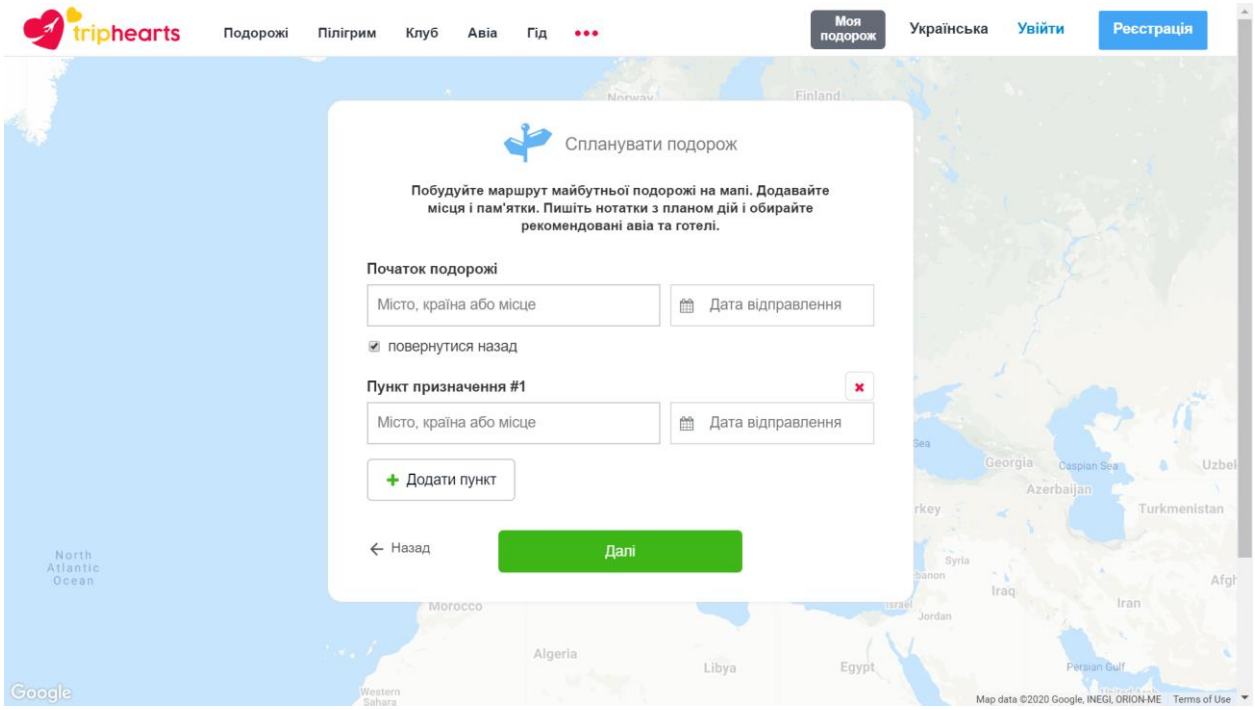

Рис. 25. Створення подорожі

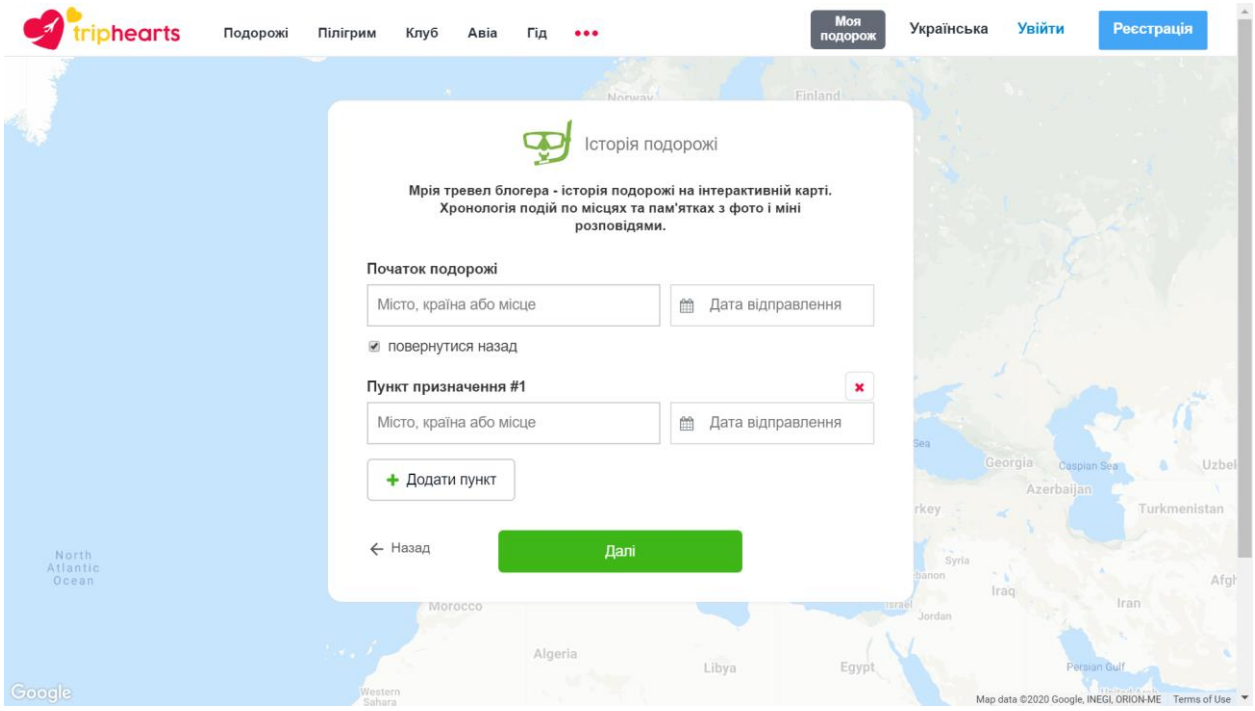

Рис. 26. Створення історії вражень на мапі

1.3.Постановка завдання.

Основним завданням курсової роботи є розробка веб-сервісу для планування подорожей, що допоможе користувачам структурувати їхні заплановані маршрути поїздок, зберігати враження і спогади про минулі мандрівки і

ділитись досвідом з іншими користувачами. Таким чином можна сформувати перелік необхідних функцій для повноцінної роботи сервісу:

1) Реєстрація користувачів.
2) Безпечне збереження інформації про користувачів у базі даних.
3) Захист даних за допомогою аутентифікації.
4) Створення маршруту подорожі, що складається із багатьох точок призначення.
5) Перегляд маршруту на мапі
6) Збереження маршруту до бази даних
7) Перегляд своїх маршрутів
8) Перегляд маршрутів інших користувачів
9) Створення текстових постів із власним досвідом та враженнями
10) Збереження постів у базу даних
11) Перегляд своїх постів
12) Перегляд постів інших користувачів

2. ТЕОРЕТИЧНІ ВІДОМОСТІ

Існує багато різних типів веб-сайтів, кожен із яких має власну аудиторію і слугує для різних цілей. Спеціалісти з рекламної компанії Internetdevels [8] виділяють такі найпопулярніші категорії:

1) Сайт-візитка. Це простий сайт, що найчастіше використовується для знайомства із діяльністю особи або фірми. Його можна вважати рекламним матеріалом для поширення інформації про деякі послуги. Також, такі сайти забезпечують надання контактної інформації, за допомогою якої можна зв’язатись із власником. Зазвичай вони не забезпечені широким функціоналом та надають користувачеві зробити кілька простих дій: надіслати повідомлення, підписатись на розсилку, скачати презентацію тощо.

2) Сайт компанії з каталогом продукції. Це тип сайту, що дуже схожий на сайт-візитку, але він містить більше інформації про діяльність компанії або особи, адже на ньому розміщений каталог продукції. Так він надає більше інформації і, як наслідок, краще приваблює клієнтів. На них приходять різні люди з різними потребами: одні - щоб дізнатись щось про компанію, інші - почитати про продукти тощо. Тому такий сайт надає користувачам набагато ширший спектр можливих дій. Проте це все ще сайт, що не передбачає існування багатьох користувачів.
3) Інтернет-магазин. На сайтах такого типу продукцію можна не тільки переглядати, а ще й замовляти. Такі сайти є надзвичайно популярними, адже мають великий попит у сучасному світі. Особливо це помітно в умовах карантину, адже більшість компаній змушені переходити в онлайн-режим задля того, щоб мати змогу надавати свої послуги під час глобальної самоізоляції. При цьому інтернет-магазини мають значні прибутки навіть у спокійні часи, адже в епоху панування інтернету можливість швидко та зручно замовити товари онлайн є надзвичайно важливою. На таких сайтах користувачам надається можливість зареєструватися та авторизуватись на сайті, зберігати вподобані товари до корзини, отримувати зворотній зв'язок від продавців, читати відгуки інших клієнтів.
4) Новинні та пошукові портали. Ці типи сайтів також мають значну популярність, адже всім важливо знати, що відбувається у світі. Останнім часом помітна тенденція відмови від телебачення, особливо серед молоді, тому із часом інтернет-джерела інформації завойовують все більшу аудиторію і, як наслідок, отримують значні прибутки завдяки розміщенню реклами. Це може бути електронна газета чи журнал, веб-сайт каналу новин, також сюди відносять деякі блоги - тобто всі місця в інтернеті, звідки користувач може дізнатися достовірну інформацію про те, що робиться у світі. Також до новинних порталів можна додати канали з новинами в таких соціальних мережах, як Facebook та Twitter.

Пошукові портали – веб-застосування, що містять посилання на інші Інтернет-ресурси, релевантні до пошукового запиту, введеного користувачем. Зазвичай такі сайти надають користувачам тільки дві можливі дії – пошук та переадресація на інші сторінки, при цьому обробляючи величезні об'єми інформації, щоб видати користувачу сайти, що найкраще відповідають його запиту. Найбільш відомими такими «двигунами» є Google, Bing, Yahoo, DuckDuckGo та Яндекс.

5) Інформаційні портали. Ці сайти є найбільш важливими для навчання та саморозвитку. Вони є необхідними джерелами інформації і поєднують у собі статті на різну тематику. Також найчастіше існують за рахунок реклами. До цього типу сайтів відносять всі, основним контентом яких є інформація – статті, відгуки, фото та відео. Зазвичай це блоги, соціальні мережі, сайти для перегляду фільмів та фото. До них можна віднести YouTube, Wikipedia, Pinterest та дуже багато інших.
6) Веб-сайти обліку товарів. Це спеціальні сайти, що використовуються деякою фірмою і доступ до яких мають лише її співробітники. Такі сайти мають найвищі вимоги до рівня безпеки даних і високі показники надійності та доступності, адже від цих параметрів залежить збереження конфіденційності користувачів, а також репутація компанії.

Веб-сервіс для планування подорожей надає користувачам можливість реєструватися на сайті та виконувати великий спектр дій, отже, це не сайт-візитівка. При цьому PlanYourTrip не пропонує відвідувачам каталог доступної продукції і не створений для розвитку бізнесу. Тому його не можна віднести до жодного з сайтів компаній, обліку товарів та інтернет-магазину. Даний сервіс не призначений для пошуку інших інтернет-ресурсів та не надає користувачам достовірних новин, тому це не новинний і не пошуковий портал. Отже, методом виключення бачимо. що веб-сервіс PlanYourTrip - інформаційний веб-сайт. І справді, даний сервіс пропонує користувачам усю можливу інформацію про геолокації світу (за допомогою GoogleMaps), надає

можливість переглядати маршрути та враження від подорожей інших користувачів для особистого використання. Також сервіс надає можливість ділитися власними враженнями та складати свої маршрути, що робить його схожим на блог.

3. ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ

3.1. Аналіз технічного завдання

PlanYourTrip – це сервіс, який призначений для будь-кого, хто любить подорожувати, тобто користувачі можуть не бути спеціалістами. Отже, спосіб реалізації має бути максимально зрозумілим та доступним. Вимоги до інтерфейсу аналогічні, він також має бути інтуїтивно зрозумілим і не перевантаженим. Саме тому найкращим і найбільш доступним варіантом реалізації є веб-сайт. Користувач заходить на сторінку і одразу має зареєструватись, щоб мати можливість зберігати результати роботи.

На вхід для реєстрації подається така інформація користувача:

1) Ім’я (воно не має бути унікальним, адже його єдине призначення – для відображення інформації про автора постів та маршрутів)
2) Електронна пошта (використовується для входу в систему)
3) Пароль (необхідний елемент для аутентифікації користувача)

Зі збереженням паролю завжди постає необхідність додаткового захисту даних, тому необхідно забезпечити хешування паролів, щоб вони не зберігались у базі даних у відкритому вигляді.

Для задач такого типу зручно використовувати поєднання NodeJS, Express та MongoDB разом із додатковими модулями для конкретних задач. До MongoDB найкраще доступатись за допомогою Mongoose, адже цей засіб робить код більш інтуїтивно зрозумілим та структурованим за рахунок використання моделей об’єктів. Для тимчасового збереження даних користувача доречно використовувати сесії, звідки можна зручно діставати ідентифікатор

користувача. Саме за допомогою них можна перевірити, чи користувач авторизований і чи має доступ до сторінок сервісу.

Прості приклади використання цих технологій можна знайти за посиланнями [9] і [10]

Для створення маршрутів доречно використовувати Google Maps API. Завдяки чіткій документації можна досить швидко розібратись із тим, як правильно працювати із картами. Окрім того, ці карти є досить популярними, а отже нескладно знайти і відповіді на нестандартні питання. Приклади роботи з Google Maps API можна знайти за посиланнями [11], [12], [13] і [14].

Задля відображення даних для користувача зручно використовувати бібліотеку jQuery або просто доступатись до елементів сторінки за допомогою властивостей DOM – об'єктної моделі документа.

3.2. Обґрунтування алгоритму й структури програми

Відповідно до поставленого завдання веб-сервіс має складатись із бекенду (для запитів до бази даних) та фронтенду (для взаємодії з користувачем).

Таким чином програма розділяється на три логічних рівні. Бекенд реалізований за допомогою NodeJS та Express. База даних реалізована за допомогою MongoDB та Mongoose. Фронтенд працює за допомогою JavaScript, jQuery та Ajax. Підключення до серверу та бази даних відбувається у файлі **start.js**, який власне запускає виконання програми. Необхідні моделі для роботи із базами даних визначені у файлах **post.js**, **route.js**, **user.js** і зберігаються у директорії models. Всі залежності та параметри програми визначені у файлі **app.js**. У ньому ж підключається файл із усіма необхідними GET та POST запитами – **index.js**. У цьому файлі генеруються всі запити та передаються необхідні параметри для шаблонів, які в свою чергу відображають кінцевий вміст сторінок. Усі шаблони зібрані у директорії views: **layout.pug**, **index.pug**, **main.pug**, **map.pug**. Клієнтський JavaScript розташований у директорії public/js: **client.js**, **map.js**. У файлі client.js

визначені усі функції для роботи зі сторінкою та динамічного генерування карток зі збереженими даними. У файлі map.js визначені всі функції для роботи із картами за допомогою Google Maps API. Стилі програми можна знайти у директорії public/css.

Таким чином для клієнта програма складається із таких сторінок:

index.pug – тут відбувається реєстрація та вхід до системи

main.pug – тут відображаються всі збережі в базу даних об’єкти та розташовані оновні посилання

map.pug – тут відбувається створення маршруту за допомогою великої інтерактивної карти і полів для введення адрес із автодоповненням.

3.3.Обґрунтування вибору засобів розробки

Перелік усіх засобів розробки:

1) JavaScript – зручна і розповсюджена мова, що чудово підходить для веб-програмування
2) NodeJS – програмна платформа, що дозволяє створювати бекенд-сервер за допомогою JavaScript
3) NPM – менеджер пакетів, що використовується разом із NodeJS і дозволяє структурувати всі необхідні для коректної роботи програми залежності
4) Express – гнучкий фреймворк для NodeJS, що дозволяє розробляти веб-додатки
5) MongoDB – проста нереляційна база даних, яка не потребує для роботи складних запитів і при цьому ефективно зберігає необхідну інформацію
6) Mongoose – бібліотека, що дозволяє створювати моделі на основі даних та зручно із ними взаємодіяти
7) Pug – інтуїтивно зрозумілий шаблонізатор, що дозволяє створювати представлення веб-сторінок із меншими зусиллями

8) Bootstrap – бібліотека елементів, що дозволяє швидко і без особливих зусиль привести сторінку до візуально приємного вигляду
9) jQuery – бібліотека, що дозволяє простіше оновлювати інформацію на сторінках
10) Ajax – технологія, що дозволяє відправляти асинхронні запити на сервер і отримувати необхідну інформацію.
11) Google Maps API – доступний та поширений інтерфейс для розробки мап

3.4. Опис розробки програми

Перелік етапів розробки програми:

1) Підключення серверу та бази даних у файлі start.js.
2) Реєстрація та збереження даних користувачів за допомогою Mongoose.
3) Забезпечення хешування паролів за допомогою модуля bcrypt.
4) Логін та аутентифікація за допомогою сесій із використанням модуля express-session.
5) Генерування сторінок для клієнта за допомогою шаблонізатора Pug.
6) Отримання Google Maps API KEY.
7) Коректне відображення карти на сторінці.
8) Підключення автодоповнення до статичних полів форми.
9) Підключення автодоповнення до динамічно згенерованих полів форми.
10) Успішне створення маршруту із двох точок.
11) Успішне створення маршруту із максимум 27 точок (у зв’язку із обмеженням Google Maps API).
12) Збереження точок маршруту до бази даних.
13) Створення та збереження постів.
14) Написання GET запитів для отримання інформації з бази даних.
15) Динамічне генерування карток із постами та маршрутами за допомогою DOM, Ajax та jQuery.

16) Невдала спроба реалізувати статичні карти для відображення маршрутів.

17) Фінальні штрихи

3.5. Створення об'єктів і розробка головної програми

Створення об’єктів відбувається за допомогою моделей Mongoose. У програмі визначені схеми об’єктів User, Route та Post. Об’єкт User створюється під час реєстрації, потім він використовується для ідентифікації під час входу до системи. Об’єкт Route створюється на сторінці map.pug після натискання на кнопку «Зберегти маршрут». Дані з нього дістаються динамічно на сторінці main.pug для відображення карток із маршрутами. Об’єкт Post створюється під час збереження назви і тексту поста і потім так само відображається динамічно у вигляді карток.

3.6. Опис файлів даних та інтерфейсу програми

Інтерфейс програми був розроблений із ідеєю лаконічності. Перша сторінка, яку бачить користувач – це сторінка із назвою сервісу, кнопками реєстрації та входу до системи та відео на тему подорожей для привернення уваги (рис. 27).

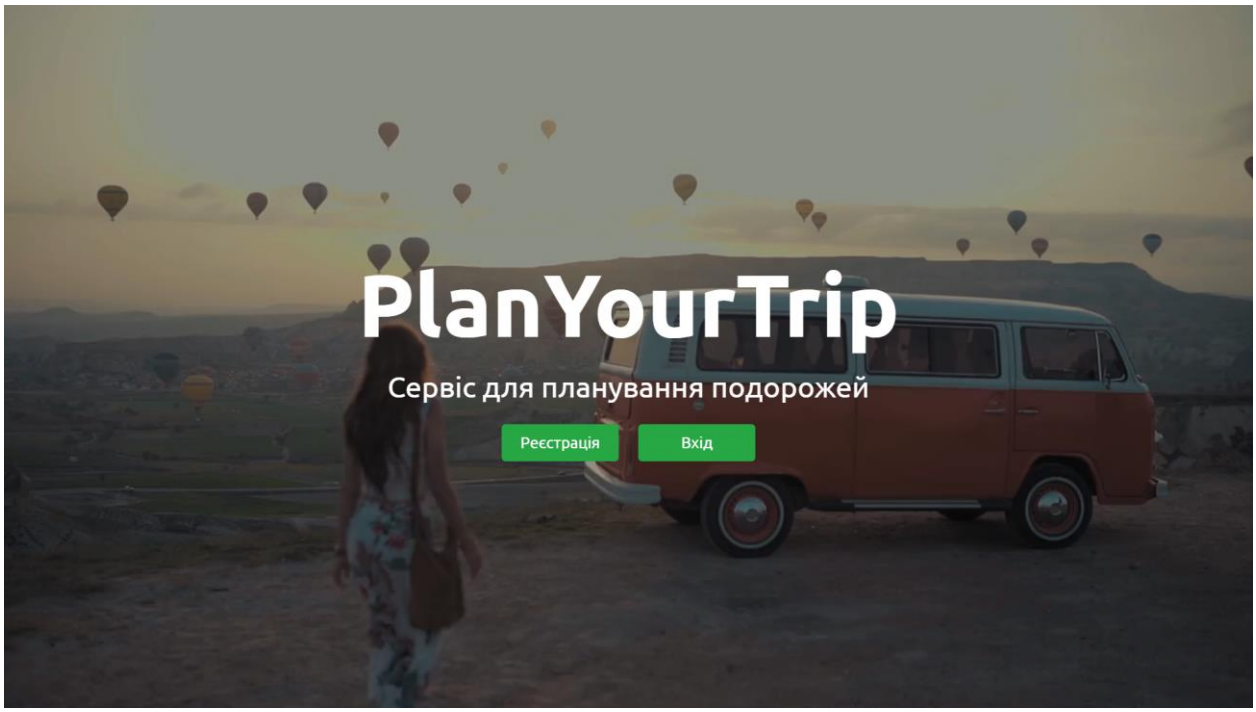

Рис. 27. Основна сторінка

Новим користувачам необхідно натиснути на кнопку «Реєстрація» та зареєструватись (рис. 28), а якщо користувач вже має акаунт, то він може натиснути на кнопку «Вхід» та ввести свої дані (рис. 29).

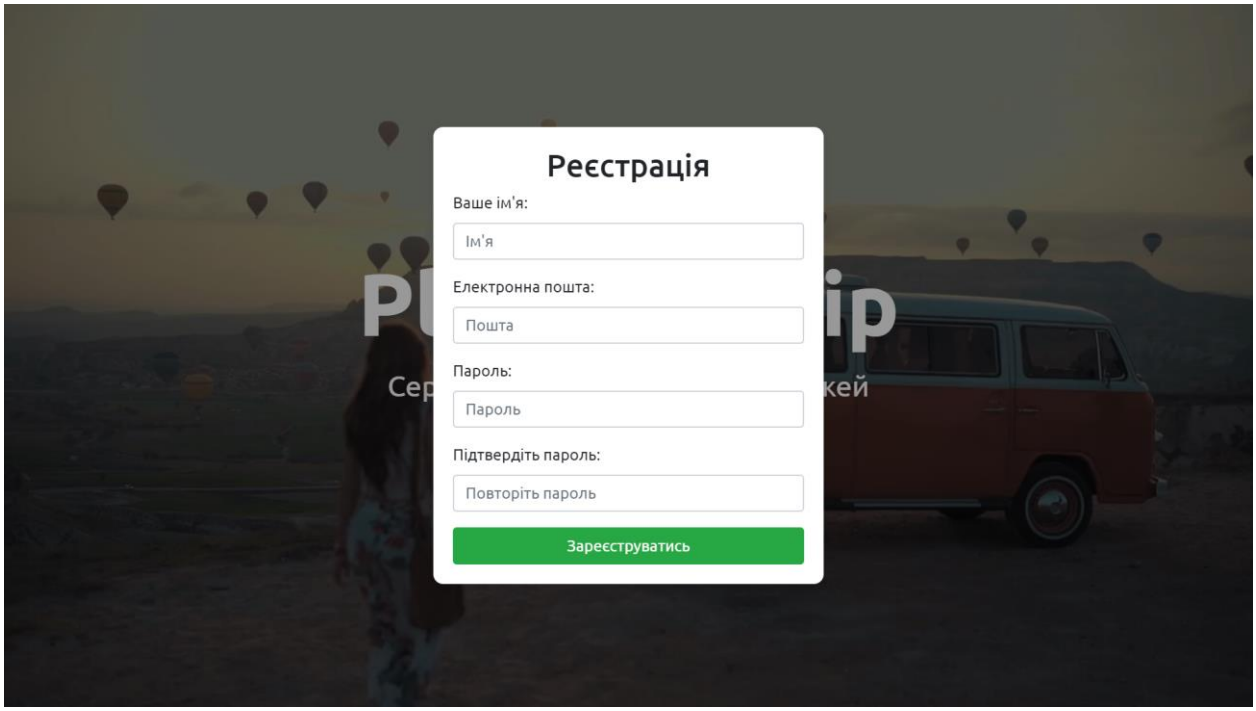

Рис. 28. Реєстрація

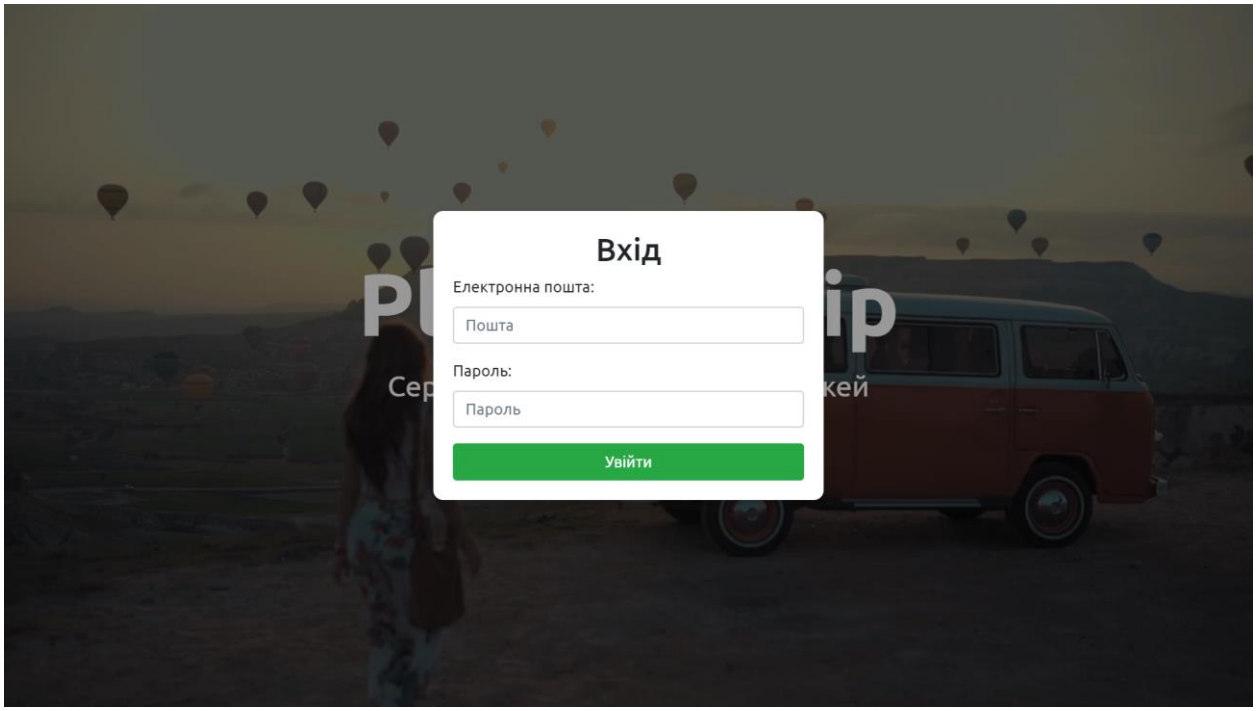

Рис. 29. Вхід

Після аутентифікації користувач опиняється на основній сторінці, де автоматично відображаються всі наявні маршрути користувачів (рис.30).

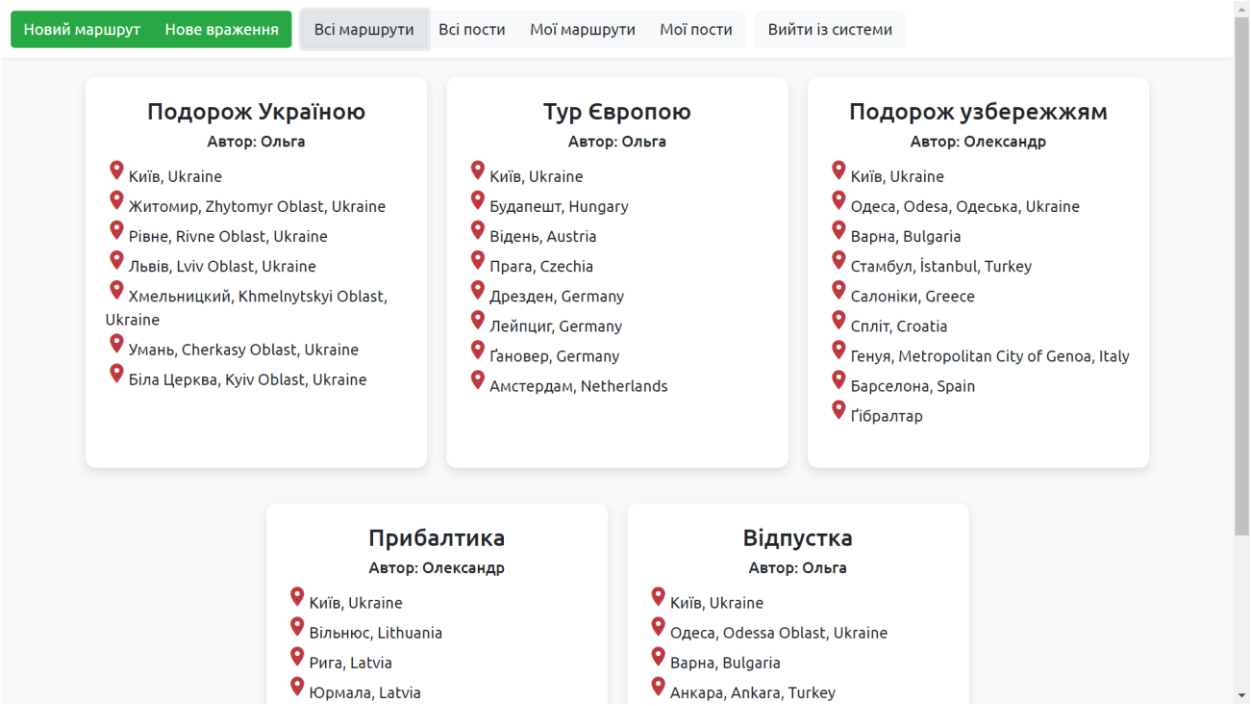

Рис. 30. Перелік маршрутів усіх користувачів

Якщо користувач натискає на кнопку «Новий маршрут», то він опиняється на сторінці із картою (рис. 31-33).

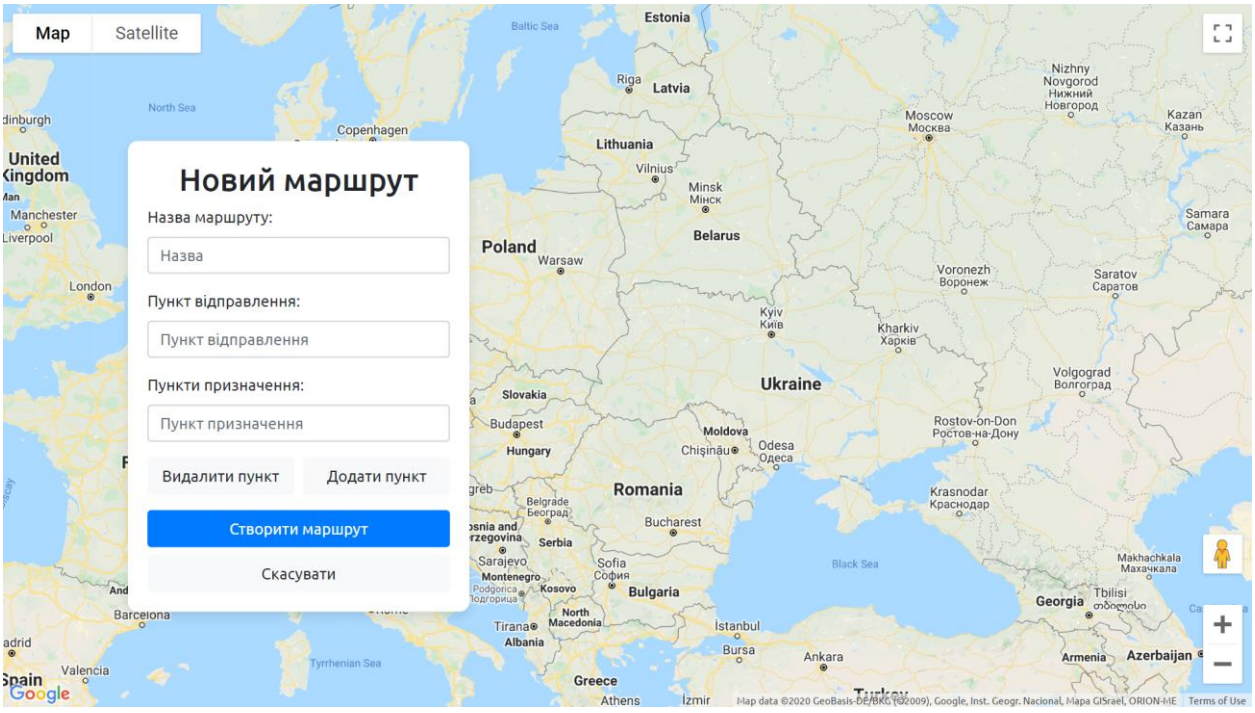

Рис. 31.Створення маршруту

Тут можна додати довільну кількість точок (до 27 включно) (рис. 32).

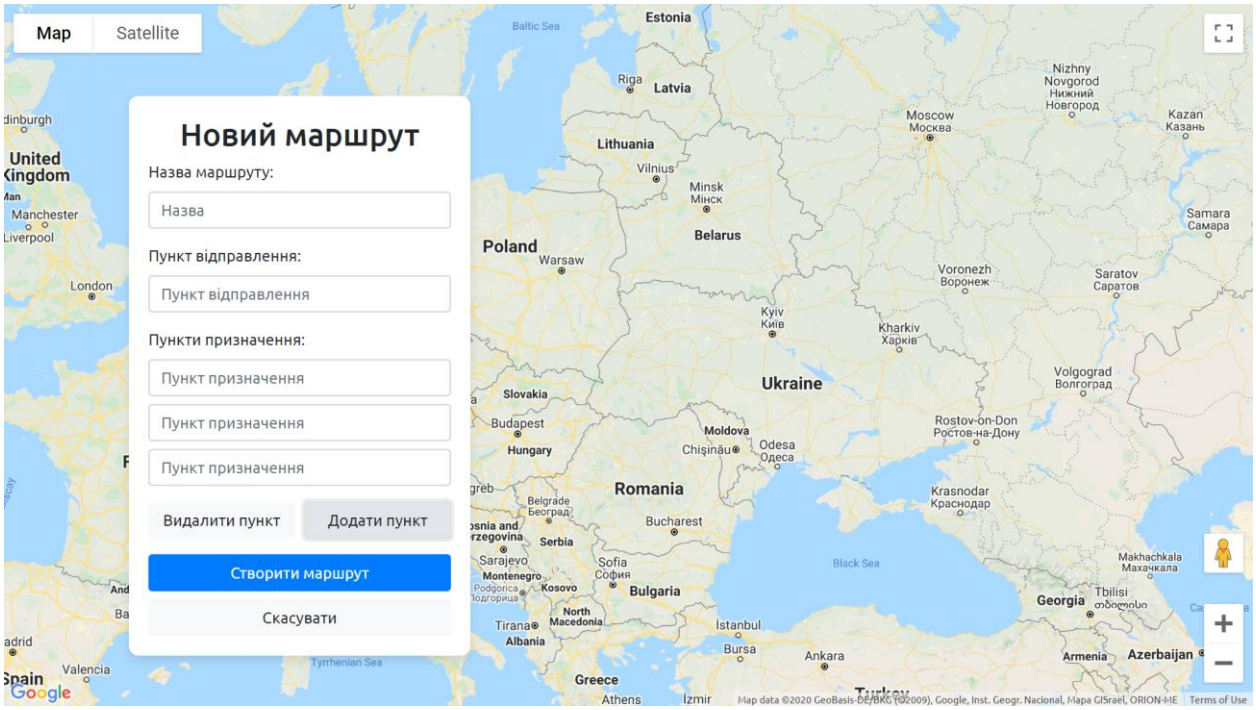

Рис. 32. Додавання декількох полів

Користувач має ввести назву маршруту і всі необхідні пункти за допомогою автодоповнення. Якщо між усіма точками існує автомобільний маршрут, то він буде виведений після натискання кнопки «Створити маршрут» (рис.33).

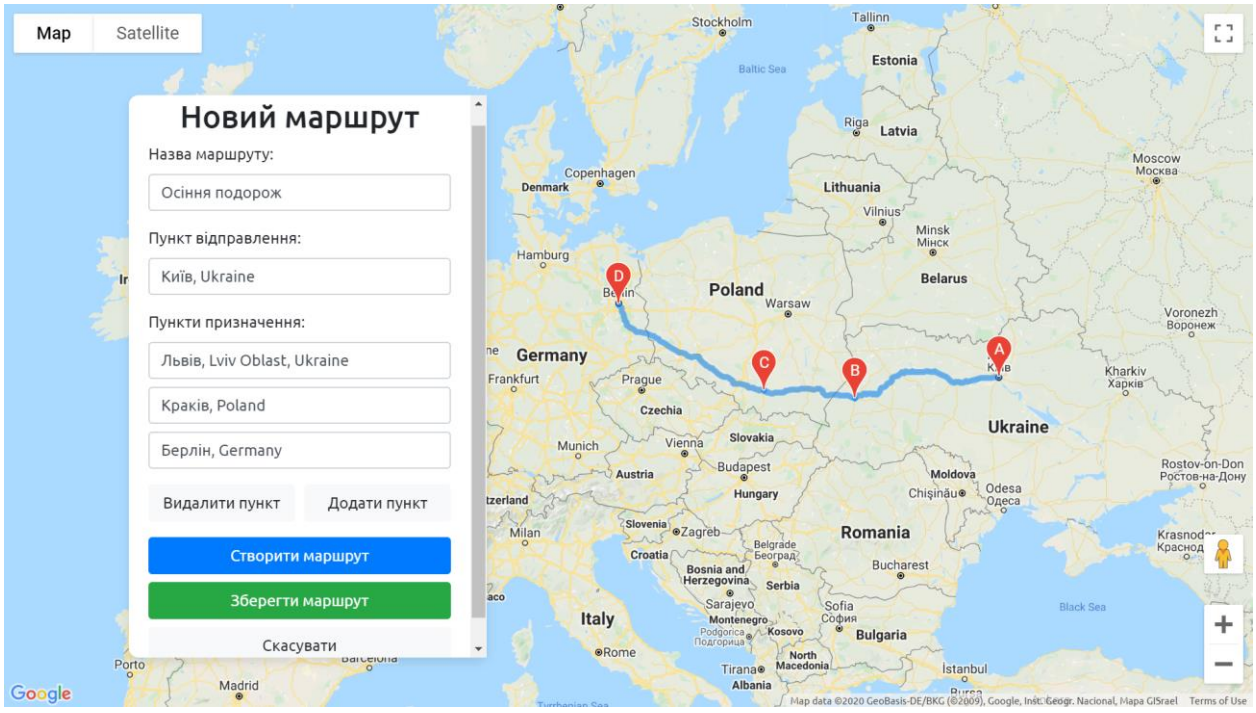

Рис. 33. Відображення маршруту на сторінці

У разі успіху на карті з’явиться маршрут, а у формі – кнопка «Зберегти маршрут». Після натискання цієї кнопки всі точки маршруту будуть збережені у базу даних і на головній сторінці відображатиметься нова картка. За допомогою кнопок «Всі маршрути», «Всі пости», «Мої маршрути» і «Мої пости» можна регулювати відображення карток на головній сторінці (рис. 30, 34, 37, 38).

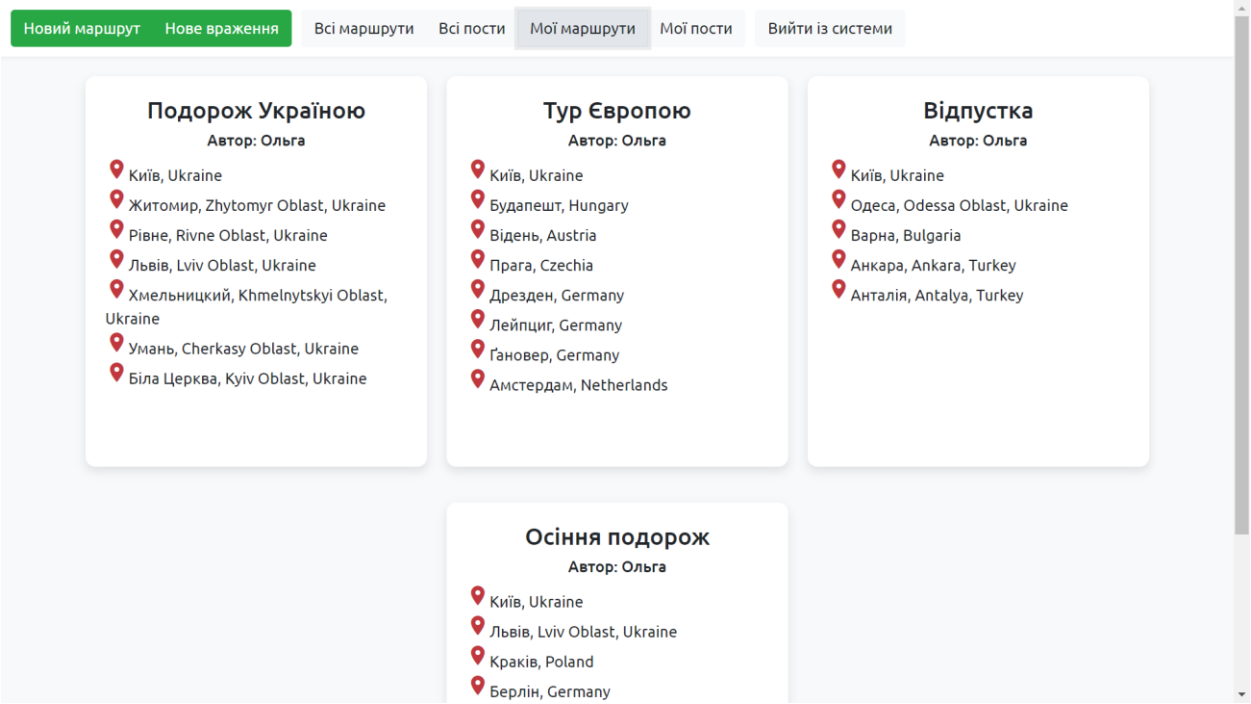

Рис. 34. Перегляд маршрутів поточного користувача

За допомогою кнопки «Нове враження» можна створити новий пост із назвою та довільним текстом (рис. 35-36). Натиснувши кнопку «Зберегти пост» і обравши необхідне відображення за допомогою відповідної кнопки, користувач побачить, що його пост з’явився на головній сторінці (рис. 37-38). Вийти із системи і тим самим завершити сесію можна за допомогою відповідної кнопки.

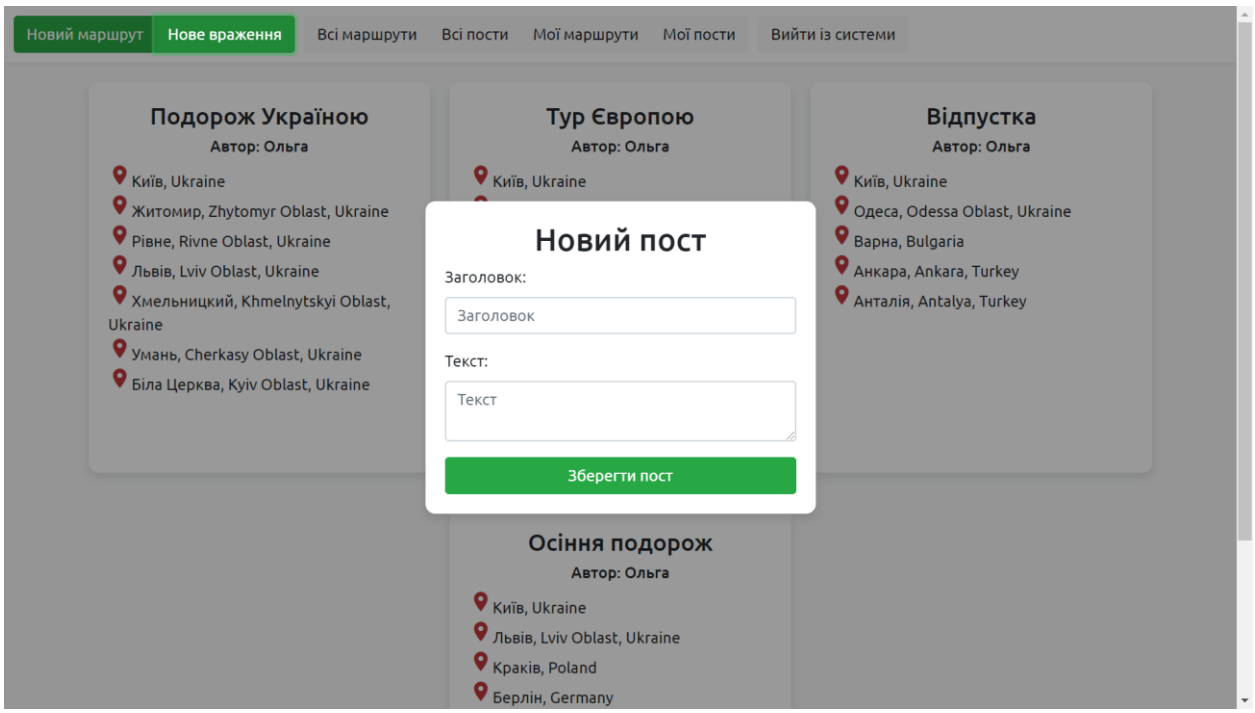

Рис. 35.Вікно створення нового поста

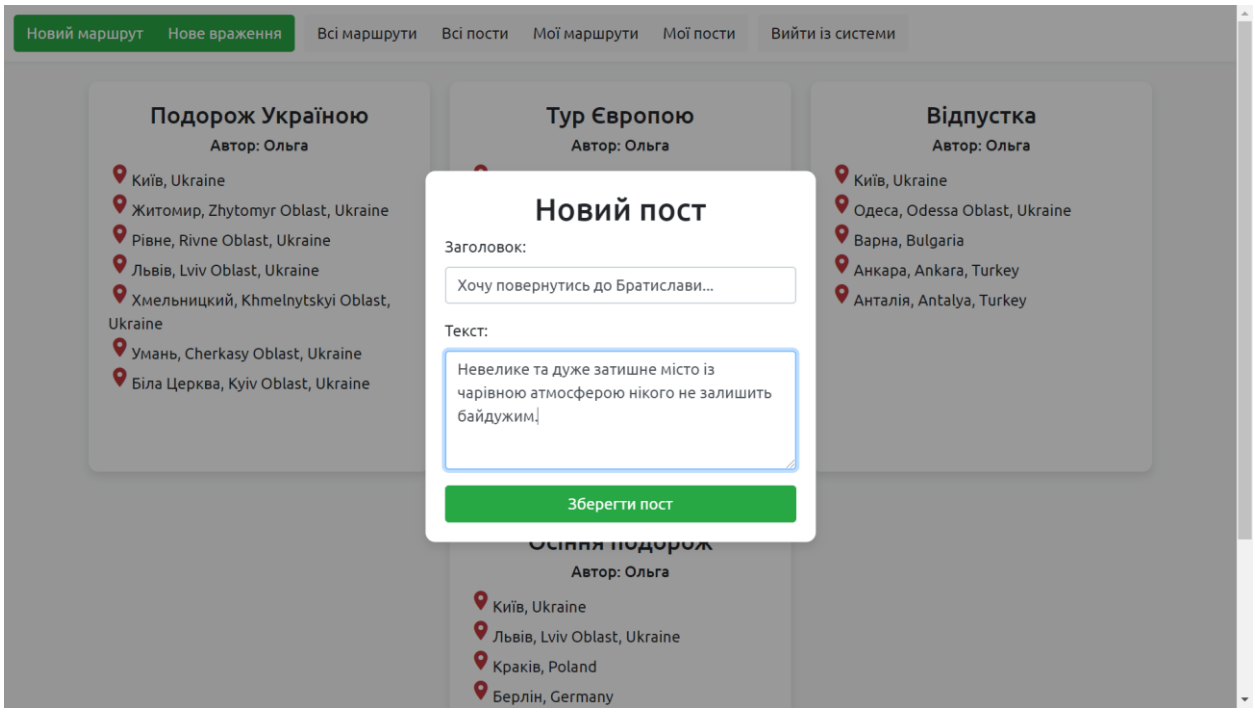

Рис. 36. Заповнення форми

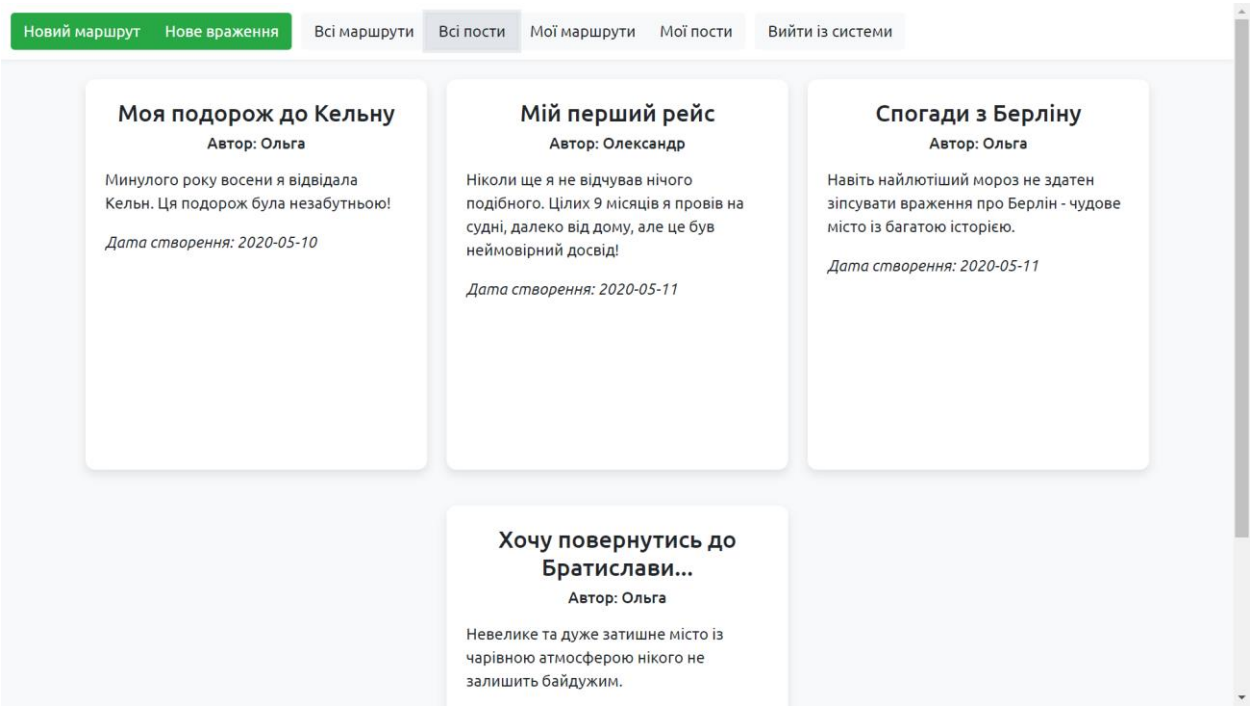

Рис. 37. Перегляд постів усіх користувачів

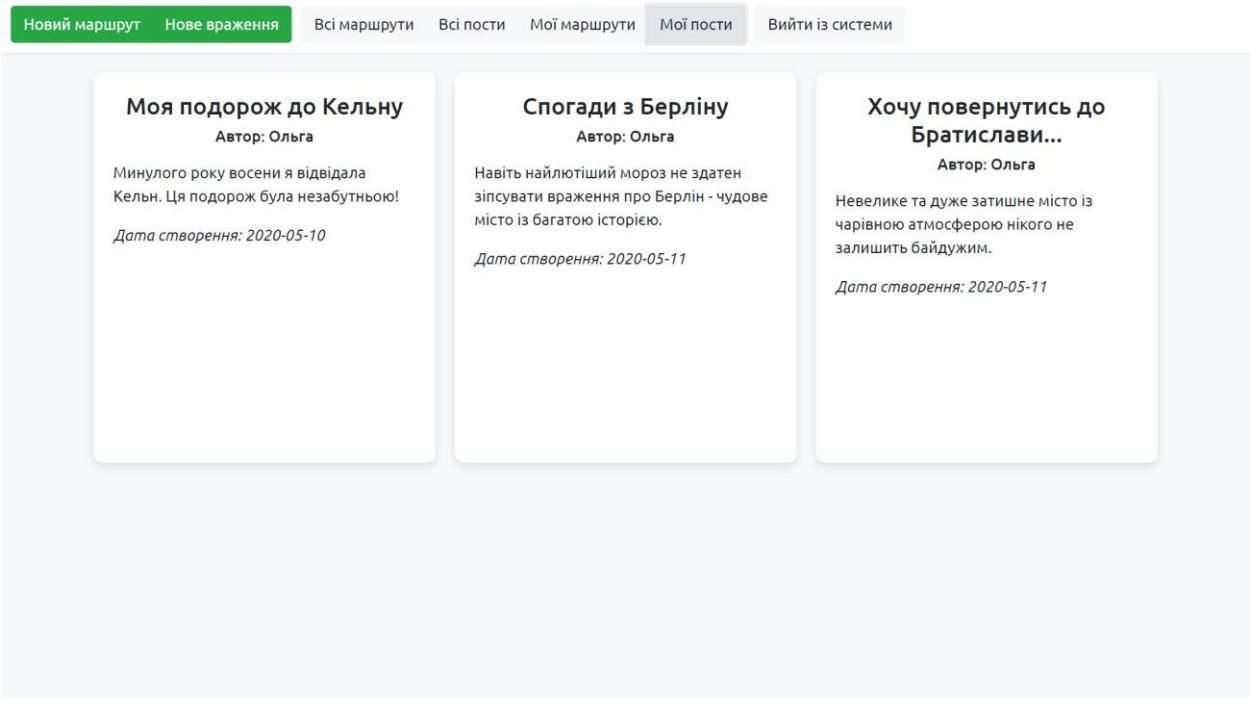

Рис. 38. Перегляд постів поточного користувача

Висновки

У результаті виконання цієї роботи було створено сервіс, що виконує такі завдання, як створення та збереження маршрутів подорожей за допомогою інтерактивної карти і написання постів із враженнями, які можуть переглядати всі користувачі. Таким чином було досягнуто поставлену мету щодо розробки

працездатного сайту із простим та зручним інтерфейсом і корисними для мандрівників можливостями. За допомогою цієї роботи було досліджено сучасний стан туристичної галузі у світі та проведено аналіз основних додатків для планування подорожей. Цей сайт став внеском у розробку альтернативних засобів для планування маршрутів і має багато перспектив для вдосконалення та подальшого розвитку.

Список використаної літератури

1) У відпустку. Туризм після коронавірусу. [Електронний ресурс]. Режим доступу: https://www.uvidpustku.com/podorozhi-pislya-coronavirusa/
2) Google Trips. [Електронний ресурс]. Режим доступу: https://www.google.com/travel/
3) Waytips. [Електронний ресурс]. Режим доступу: http://waytips.com/
4) Youroute. [Електронний ресурс]. Режим доступу: https://youroute.ru/
5) Booking. [Електронний ресурс]. Режим доступу: https://www.booking.com/index.uk.html
6) Tropinki. [Електронний ресурс]. Режим доступу: https://ttrails.ru/
7) Triphearts. [Електронний ресурс]. Режим доступу: https://triphearts.com/ua/

8) Internetdevels. Найпопулярніші типи сайтів. [Електронний ресурс]. Режим доступу: https://internetdevels.ua/blog/most-common-types-of-websites
9) Sitepoint. Побудова простого додатку за допомогою Node, Bootstrap і MongoDB. [Електронний ресурс]. Режим доступу: https://www.sitepoint.com/build-simple-beginner-app-node-bootstrap-mongodb/
10) Medium. Приклад аутентифікації із використанням Node та MongoDB. [Електронний ресурс]. Режим доступу: https://medium.com/createdd-notes/starting-with-authentication-a-tutorial-with-node-js-and-mongodb-25d524ca0359
11) Google Developers. Документація з використання автодоповнення полів. [Електронний ресурс]. Режим доступу: https://developers.google.com/maps/documentation/javascript/examples/places-autocomplete
12) Stack Overflow. Приклад додавання декількох полів із автодоповненням на одну сторінку. [Електронний ресурс]. Режим доступу: https://stackoverflow.com/questions/20416058/adding-multiple-instances-of-google-places-on-same-page
13) Google Developers. Документація з використання маршрутів. [Електронний ресурс]. Режим доступу: https://developers.google.com/maps/documentation/javascript/examples/places-autocomplete-directions
14) Google Developers. Документація з використання маршрутів, що складаються із багатьох пунктів. [Електронний ресурс].

Режим доступу: https://developers.google.com/maps/documentation/javascript/examples/directions-waypoints

Додатки

Програмний код

start.js:

```
require('dotenv').config();
const mongoose = require('mongoose');

//connect to database
mongoose.connect(process.env.DATABASE, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
  useCreateIndex: true
});

//success and error messages
mongoose.connection
  .on('open', () => {
    console.log('Mongoose connection open');
  })
  .on('error', (err) => {
    console.log('Connection error: ' +  err.message);
  });

const app = require('./app');

//connect to server
const port = process.env.PORT;
const server = app.listen(port, () => {
  console.log('Server is running on port ' + port);
});
```

app.js:

```
const express = require('express');
const app = express();

//initialize bodyParser
const bodyParser = require('body-parser');
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

//set template engine
const path = require('path');
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

//set static directory
app.use(express.static('public'));

//initialize session
const session = require('express-session');
app.use(session({
  secret: 'plan and travel',
  resave: true,
  saveUninitialized: false
}));

const routes = require('./routes/index');
app.use('/', routes);

module.exports = app;
```

index.js:

```
const express = require("express");
const router = express.Router();
const ObjectId = require('mongoose').Types.ObjectId;

let User = require('../models/user');
let Post = require('../models/post');
let Route = require('../models/route');
```

```
require('dotenv').config();

//index page
router.get('/', (req, res, next) => {
  User.findById(req.session.userId)
    .exec(function (error, user) {
      if (error) {
        return next(error);
      } else if (user) {
        res.redirect('/main');
      } else {
        res.render('index', {title: 'PlanYourTrip'});
      }
    });
});

//register and login
router.post('/auth', function (req, res, next) {
  //check if user typed the same password twice
  if (req.body.password !== req.body.passwordConf) {
    const err = new Error("Паролі не збігаються");
    err.status = 400;
    return next(err);
  }

  //registration if data is correct
  if (req.body.email &&
    req.body.username &&
    req.body.password &&
    req.body.passwordConf) {
    const userData = {
      email: req.body.email,
      username: req.body.username,
      password: req.body.password,
    }
    User.create(userData, function (error, user) {
```

```
      if (error) {
        return next(error);
      } else {
        req.session.userId = user._id;
        res.redirect('/main');
      }
    });
    //login if data is correct
  } else if (req.body.logemail && req.body.logpassword) {
    User.authenticate(req.body.logemail, req.body.logpassword, function (error, user)
{
      if (error || !user) {
        var err = new Error('У базі даних відсутній користувач із такими даними');
        err.status = 401;
        return next(err);
      } else {
        req.session.userId = user._id;
        res.redirect('/main');
      }
    });
  } else {
    var err = new Error('Будь ласка, заповніть всі поля');
    err.status = 400;
    return next(err);
  }
});

//main page
router.get('/main', (req, res, next) => {
  User.findById(req.session.userId)
    .exec(function (error, user) {
      if (error) {
        return next(error);
      } else {
        if (user === null) {
          res.redirect('/');
        } else {
```

```
          const key = process.env.KEY;
          res.render('main', {title: 'PlanYourTrip', key: key});
        }
      }
    });
});

router.get('/all-routes', (req, res, next) => {
  Route.find({}, function(err, routes) {
    if (err) {
      res.send(err);
    } else {
      res.send(routes);
    }
  });
});

router.get('/all-posts', (req, res, next) => {
  Post.find({}, function(err, posts) {
    if (err) {
      res.send(err);
    } else {
      res.send(posts);
    }
  });
});

router.get('/my-routes', (req, res, next) => {
  Route.find({user_id: req.session.userId}, function(err, routes) {
    if (err) {
      res.send(err);
    } else {
      res.send(routes);
    }
  });
});
```

```
router.get('/my-posts', (req, res, next) => {
  Post.find({user_id: req.session.userId}, function(err, posts) {
    if (err) {
      res.send(err);
    } else {
      res.send(posts);
    }
  });
});

router.get('/username/:user_id', (req, res, next) => {
  User.findOne({_id: req.params.user_id}, function(err, user) {
    if (err) {
      res.send(err);
    } else {
      const username = user.username;
      res.send(username);
    }
  });
});

router.post('/save-post', function (req, res, next) {
  const currentDate = new Date();
  const userId = req.session.userId;
  if (req.body.title &&
    req.body.text) {
    const postData = {
      title: req.body.title,
      text: req.body.text,
      user_id: userId,
      date: currentDate,
    }
    Post.create(postData, function (error, post) {
      if (error) {
        return next(error);
```

```
      } else {
        res.redirect('/main');
      }
    });
  }
});

router.post('/save-route', function (req, res, next) {
  const userId = req.session.userId;
  if (req.body.title &&
    req.body.departure &&
    req.body.waypts) {
    let waypts = req.body.waypts;
    const destination = waypts.pop();
    const routeData = {
      title: req.body.title,
      origin: req.body.departure,
      waypts: waypts,
      destination: destination,
      user_id: userId,
    }
    Route.create(routeData, function (error, route) {
      if (error) {
        return next(error);
      } else {
        res.redirect('/main');
      }
    });
  }
});

//create route page
router.get('/map', (req, res, next) => {
  User.findById(req.session.userId)
    .exec(function (error, user) {
      if (error) {
```

```
        return next(error);
      } else {
        if (user === null) {
          res.redirect('/');
        } else {
          const key = process.env.KEY;
          res.render('map', {title: 'Новий маршрут', key: key});
        }
      }
    });
});

//logout
router.get('/logout', function (req, res, next) {
  if (req.session) {
    //delete session
    req.session.destroy(function (err) {
      if (err) {
        return next(err);
      } else {
        res.redirect('/');
      }
    });
  }
});

module.exports = router;
```

post.js:

```
const mongoose = require('mongoose');

const PostSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
    trim: true,
  },
```

```
  text: {
    type: String,
    required: true,
  },
  user_id: {
    type: mongoose.Schema.Types.ObjectId,
    required: true,
  },
  date: {
    type: Date,
    required: true,
  },
});

let Post = mongoose.model('Post', PostSchema);
module.exports = Post;
```

route.js:

```
const mongoose = require('mongoose');

const RouteSchema = new mongoose.Schema({
  title: {
    type: String
  },
  origin: {
    type: String
  },
  waypts: {
    type: Array
  },
  destination: {
    type: String
  },
  user_id: {
    type: mongoose.Schema.Types.ObjectId,
    required: true,
  },
```

```
});

let Route = mongoose.model('Route', RouteSchema);
module.exports = Route;
```

user.js:

```
const mongoose = require('mongoose');
var bcrypt = require('bcrypt');

const UserSchema = new mongoose.Schema({
  username: {
    type: String,
    required: true,
    trim: true,
  },
  email: {
    type: String,
    unique: true,
    required: true,
    trim: true,
  },
  password: {
    type: String,
    required: true,
  },
});

//authentication
UserSchema.statics.authenticate = function (email, password, callback) {
User.findOne({ email: email })
  .exec(function (err, user) {
    if (err) {
      return callback(err)
    } else if (!user) {
      var err = new Error('User not found.');
      err.status = 401;
      return callback(err);
```

```
    }
    bcrypt.compare(password, user.password, function (err, result) {
      if (result === true) {
        return callback(null, user);
      } else {
        return callback();
      }
    })
  });
}

//hashing password before saving it to database
UserSchema.pre('save', function (next) {
const user = this;
bcrypt.hash(user.password, 10, function (err, hash) {
  if (err) {
    return next(err);
  }
  user.password = hash;
  next();
})
});

let User = mongoose.model('User', UserSchema);
module.exports = User;
```

client.js:

```
$("#register-button").on('click', showRegisterWindow);
$("#login-button").on('click', showLoginWindow);

$("#remove-point-button").on('click', removeRoutePoint);
$("#add-point-button").on('click', addRoutePoint);

$("#create-post-button").on('click', showCreatePostWindow);
$("#all-routes-button").on('click', showAllRoutes);
$("#all-posts-button").on('click', showAllPosts);
$("#my-routes-button").on('click', showMyRoutes);
```

```
$("#my-posts-button").on('click', showMyPosts);

$("#dim-screen-main").on('click', hideWindowMain);
$("#dim-screen").on('click', hideWindow);

$(document).ready(function() {
  $("#all-routes-button").click();
  $("#all-routes-button").focus();
});

function showRegisterWindow() {
  $("#dim-screen").show();
  $("#register-window").show();
}

function showLoginWindow() {
  $("#dim-screen").show();
  $("#login-window").show();
}

function removeRoutePoint() {
  if ($("#destinations-container > input").length == 1) {
    window.alert("Для побудови маршруту необхідно не менше ніж 2 пункти");
  } else {
    $("#destinations-container").children("input:last").remove();
  }
}

function addRoutePoint() {
  if ($("#destinations-container > input").length == 26) {
    window.alert("Ви можете додати не більше ніж 27 пунктів");
  } else {
    let newInput = document.createElement("input");
    newInput.classList.add("form-control");
    newInput.classList.add("point-input");
    newInput.classList.add("waypts");
```

```
    newInput.classList.add("mt-2");
    newInput.setAttribute("type", "text");
    newInput.setAttribute("name", "waypts[]");
    newInput.setAttribute("placeholder", "Пункт призначення");
    $("#destinations-container").append(newInput);
  }
}

function showCreatePostWindow() {
  $("#dim-screen-main").show();
  $("#create-post-window").show();
}

function showAllRoutes() {
  $("#obj-container").empty();
  const url = "http://localhost:8888/all-routes";
  $.ajax({
    url: url
  }).then(function (result) {
    const routes = result;
    routes.forEach((route, i) => {
      const routeId = route._id;
      const title = route.title;
      const origin = route.origin;
      const waypts = route.waypts;
      const destination = route.destination;
      const url = "http://localhost:8888/username/" + route.user_id;
      $.ajax({
        url: url
      }).then(function (result) {
          const username = result;
          createRouteCard(routeId, title, username, origin, waypts, destination);
        });
    });
  });
}
```

```
function createRouteCard(routeId, title, username, origin, waypts, destination) {
  let objCard = document.createElement("div");
  objCard.classList.add("obj-card");

  let cardTitle = document.createElement("h4");
  cardTitle.classList.add("center");
  cardTitle.innerHTML = title;
  objCard.append(cardTitle);

  let cardUser = document.createElement("h6");
  cardUser.classList.add("center");
  cardUser.innerHTML = "Автор: " + username;
  objCard.append(cardUser);

  let placesList = document.createElement("ul");
  let originLi = document.createElement("li");
  let originIcon = document.createElement("i");
  originIcon.classList.add("material-icons");
  originIcon.innerHTML = "place";
  let originText = document.createTextNode(origin);
  originLi.appendChild(originIcon);
  originLi.appendChild(originText);
  placesList.appendChild(originLi);
  for(let i = 0; i < waypts.length; i++) {
    let wayLi = document.createElement("li");
    let wayIcon = document.createElement("i");
    wayIcon.classList.add("material-icons");
    wayIcon.innerHTML = "place";
    let wayText = document.createTextNode(waypts[i]);
    wayLi.appendChild(wayIcon);
    wayLi.appendChild(wayText);

    placesList.appendChild(wayLi);
  }
  let destLi = document.createElement("li");
```

```
  let destIcon = document.createElement("i");
  destIcon.classList.add("material-icons");
  destIcon.innerHTML = "place";
  let destText = document.createTextNode(destination);
  destLi.appendChild(destIcon);
  destLi.appendChild(destText);
  placesList.appendChild(destLi);

  objCard.append(placesList);
  $("#obj-container").append(objCard);
}

function showAllPosts() {
  $("#obj-container").empty();
  const url = "http://localhost:8888/all-posts";
  $.ajax({
    url: url
  }).then(function (result) {
    const posts = result;
    posts.forEach((post, i) => {
      const postId = post._id;
      const title = post.title;
      const text = post.text;
      const date = post.date;
      const url = "http://localhost:8888/username/" + post.user_id;
      $.ajax({
        url: url
      }).then(function (result) {
          const username = result;
          createPostCard(postId, title, username, date, text);
        });
    });
  });
}

function createPostCard(postId, title, username, date, text) {
```

```
  let objCard = document.createElement("div");
  objCard.classList.add("obj-card");

  let cardTitle = document.createElement("h4");
  cardTitle.classList.add("center");
  cardTitle.innerHTML = title;
  objCard.append(cardTitle);

  let cardUser = document.createElement("h6");
  cardUser.classList.add("center");
  cardUser.innerHTML = "Автор: " + username;
  objCard.append(cardUser);

  let postText = document.createElement("p");
  postText.innerHTML = text;
  postText.classList.add("mt-3");
  objCard.append(postText);

  let postDate = document.createElement("i");
  postDate.innerHTML = "Дата створення: " + date.split("T")[0];
  objCard.append(postDate);

  $("#obj-container").append(objCard);
}

function showMyRoutes() {
  $("#obj-container").empty();
  const url = "http://localhost:8888/my-routes";
  $.ajax({
    url: url
  }).then(function (result) {
    const routes = result;
    routes.forEach((route, i) => {
      const routeId = route._id;
      const title = route.title;
      const origin = route.origin;
```

```
      const waypts = route.waypts;
      const destination = route.destination;
      const url = "http://localhost:8888/username/" + route.user_id;
      $.ajax({
        url: url
      }).then(function (result) {
          const username = result;
          createRouteCard(routeId, title, username, origin, waypts, destination);
        });
    });
  });
}

function showMyPosts() {
  $("#obj-container").empty();
  const url = "http://localhost:8888/my-posts";
  $.ajax({
    url: url
  }).then(function (result) {
    const posts = result;
    posts.forEach((post, i) => {
      const postId = post._id;
      const title = post.title;
      const text = post.text;
      const date = post.date;
      const url = "http://localhost:8888/username/" + post.user_id;
      $.ajax({
        url: url
      }).then(function (result) {
          const username = result;
          createPostCard(postId, title, username, date, text);
        });
    });
  });
}
```

```
function hideWindowMain() {
  $("#dim-screen-main").hide();
  $("#create-post-window").hide();
}

function hideWindow() {
  $("#dim-screen").hide();
  $("#register-window").hide();
  $("#login-window").hide();
}
```

map.js:

```
function initMap() {
  const directionsService = new google.maps.DirectionsService;
  const directionsRenderer = new google.maps.DirectionsRenderer;
  let map = new google.maps.Map(document.getElementById('map'), {
    center: new google.maps.LatLng(49.842957, 24.031111),
    zoom: 5,
  });
  initAutocomplete();
  directionsRenderer.setMap(map);
  document.getElementById('create-route-button').addEventListener('click', function()
{
    calculateAndDisplayRoute(directionsService, directionsRenderer);
  });
}

//add autocomplete to each new input field
$("#add-point-button").on('click', initAutocomplete);

function initAutocomplete() {
  const inputFields = document.getElementsByClassName('point-input');
  let autocompletes = [];
  for (let i = 0; i < inputFields.length; i++) {
    let autocomplete = new google.maps.places.Autocomplete(inputFields[i]);
    autocompletes.push(autocomplete);
  }
```

```
}

function calculateAndDisplayRoute(directionsService, directionsRenderer) {
  const origin = document.getElementById('departure-input').value;
  //retrieve all the waypoints from input fields
  let waypts = [];
  const waypointsCollection = document.getElementsByClassName('waypts');
  const waypointsArray = Array.from(waypointsCollection);
  //set the last waypoint as the final destination
  const destination = waypointsArray.pop().value;
  for (var i = 0; i < waypointsArray.length; i++) {
    waypts.push({
      location: waypointsArray[i].value,
      stopover: true
    });
  }

  //build route
  directionsService.route({
    origin: origin,
    destination: destination,
    waypoints: waypts,
    //build route in the given order
    optimizeWaypoints: false,
    //only works for places on the same continent
    travelMode: 'DRIVING'
  }, function(response, status) {
    if (status === 'OK') {
      directionsRenderer.setDirections(response);
      $("#save-route-button").show();
    } else if (status === 'ZERO_RESULTS') {
      window.alert('Помилка ' + status + ': для заданих точок не існує автомобільного
маршруту');
    } else {
      window.alert('Не вдалося побудувати маршрут через помилку ' + status);
    }
  });
```

```
}
```

layout.pug:

```
doctype html
html
  head
    title= title
    link(rel='stylesheet' href='/css/bootstrap.min.css')
    link(rel='stylesheet' href='/css/fonts.css')
    link(rel='stylesheet' href='/css/main.css')

  body.bg-light
    block content
    script(src='/js/jquery.js' type='text/javascript')
    script(src='/js/client.js' type='text/javascript')
    script(src='/js/map.js' type='text/javascript')
```

index.pug:

```
extends layout
block content
  #video-container
    //Video by Taryn Elliott from Pexels
    video.cover(width='100%' height='100%' autoplay loop)
      source(src='/media/video720.mp4' type='video/mp4')

  #welcome-container
    h1 PlanYourTrip
    p Сервіс для планування подорожей
    button#register-button.btn.btn-success Реєстрація
    button#login-button.btn.btn-success Вхід
  #dim-screen
  #register-window
    h2.center Реєстрація
    form(action="/auth" method="POST")
      .form-group
        label(for="name") Ваше ім'я:
```

```
        input.form-control(type="text"  id="name"  name="username"  placeholder="Ім'я" required)
      .form-group
        label(for="email") Електронна пошта:
        input.form-control(type="email"  id="email"  name="email"  placeholder="Пошта" required)
      .form-group
        label(for="password") Пароль:
        input.form-control(type="password"       id="password"       name="password" placeholder="Пароль" required)
      .form-group
        label(for="passwordConf") Підтвердіть пароль:
        input.form-control(type="password"   id="passwordConf"   name="passwordConf" placeholder="Повторіть пароль" required)
      button.btn.btn-block.btn-success(type="submit") Зареєструватись


  #login-window
    h2.center Вхід
    form(action="/auth" method="POST")
      .form-group
        label(for="logemail") Електронна пошта:
        input.form-control(type="email"         id="logemail"         name="logemail" placeholder="Пошта" required)
      .form-group
        label(for="logpassword") Пароль:
        input.form-control(type="password"    id="logpassword"    name="logpassword" placeholder="Пароль" required)
      button.btn.btn-block.btn-success(type="submit") Увійти
```

main.pug:

```
extends layout
block content
  #header
    .btn-group.add-margin
      a.btn.btn-success(href='http://localhost:8888/map') Новий маршрут
      button#create-post-button.btn.btn-success Нове враження
    .btn-group.add-margin
      button#all-routes-button.btn.btn-light Всі маршрути
      button#all-posts-button.btn.btn-light Всі пости
```

```
      button#my-routes-button.btn.btn-light Мої маршрути
      button#my-posts-button.btn.btn-light Мої пости
    a.btn.btn-light.add-margin(href='http://localhost:8888/logout') Вийти із системи
  #dim-screen-main
  #create-post-window
    h2.center Новий пост
    form(action="/save-post" method="POST")
      .form-group
        label(for="title") Заголовок:
        input.form-control(type="text"            id="title"            name="title"
placeholder="Заголовок" required)
      .form-group
        label(for="text") Текст:
        textarea.form-control(type="text"  id="text"  name="text"  placeholder="Текст"
required)
      button.btn.btn-block.btn-success(type="submit") Зберегти пост
  #obj-container
```

map.pug:

```
extends layout
block content
  #map
  .scroll
    #route-window
      form(action='save-route' method='POST')
        h2.center(editable='true') Новий маршрут
        .form-group
          label(for="departure-input") Назва маршруту:
          input#title-input.form-control(type="text"                    name="title"
placeholder="Назва")
        .form-group
          label(for="departure-input") Пункт відправлення:
          input#departure-input.form-control.point-input(type="text" name="departure"
placeholder="Пункт відправлення")
        #destinations-container.form-group
          label(for="first-destination-input") Пункти призначення:
          input#first-destination-input.form-control.point-input.waypts(type="text"
name="waypts[]" placeholder="Пункт призначення")
        .mb-3
```

```
          button#remove-point-button.btn.btn-light(type="button") Видалити пункт
          button#add-point-button.btn.btn-light(type="button") Додати пункт
        button#create-route-button.btn.btn-block.btn-primary(type="button")  Створити маршрут
        button#save-route-button.btn.btn-block.btn-success(type="submit")    Зберегти маршрут
        a.btn.btn-block.btn-light(href='http://localhost:8888/main') Скасувати
  script(src='https://maps.googleapis.com/maps/api/js?key='       +       key       + '&libraries=places&callback=initMap' async defer)
```

main.css:

```
*, .btn {
  font-family: 'Ubuntu', sans-serif;
}

#video-container {
  width: 100%;
  height: 100vh;
}

#video-container video {
  filter: brightness(0.8);
}

.cover {
  object-fit: cover;
}

#welcome-container {
  position: absolute;
  margin-left: auto;
  margin-right: auto;
  left: 0;
  right: 0;
  top: 50%;
  transform: translateY(-50%);
  text-align: center;
}
```

```
#welcome-container h1 {
  color: white;
  font-size: 90px;
  font-weight: bold;
}

#welcome-container p {
  color: white;
  font-size: 30px;
}

#welcome-container .btn {
  width: 120px;
  margin: 0 10px;
}

#register-window, #login-window, #create-post-window {
  display: none;
  width: 400px;
  max-height: 90vh;
  position: absolute;
  margin-left: auto;
  margin-right: auto;
  left: 0;
  right: 0;
  top: 50%;
  transform: translateY(-50%);
  padding: 20px;
  overflow: auto;
  background-color: white;
  border-radius: 10px;
  box-shadow: 0px 5px 10px rgba(0, 0, 0, 0.1);
}

#header {
```

```
  padding: 5px 5px;
  background-color: white;
  box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.1);
}

.add-margin {
  margin: 5px 5px;
}

.to-right {
  float: right;
}

#obj-container {
  text-align: center;
}

.obj-card {
  display: inline-block;
  text-align: left;
  width: 350px;
  height: 400px;
  margin: 20px 10px 10px 10px;
  padding: 20px;
  border-radius: 10px;
  display: inline-block;
  background-color: white;
  overflow: auto;
  box-shadow: 0px 5px 10px rgba(0, 0, 0, 0.1);
}

.obj-card ul {
  padding-left: 0;
  list-style-type: none;
}
```

```
#map {
  position: fixed;
}

#route-window {
  max-height: 80vh;
  position: absolute;
  left: 10%;
  top: 53%;
  transform: translateY(-50%);
  overflow: auto;
  padding: 20px;
  background-color: white;
  border-radius: 10px;
  box-shadow: 0px 5px 10px rgba(0, 0, 0, 0.1);
}

#remove-point-button, #add-point-button {
  width: 150px;
}

#remove-point-button {
  margin-right: 10px;
}

#save-route-button {
  display: none;
}

.center {
  text-align: center;
}

#dim-screen, #dim-screen-main {
  display: none;
  position: fixed;
```

```
  width: 100vw;
  height: 100vh;
  left: 0;
  top: 0;
  background-color: rgba(0, 0, 0, 0.40);
}

#map {
  width: 100vw;
  height: 100vh;
}
```