КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Веб-сервіс з пошуку місць для планування туристичних маршрутів»

Виконав _____
(Підпис)
Радченко Володимир Юрійович
(прізвище, ім'я, по батькові)

Керівник Броварець Олександр Олександрович
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")
Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2021

Завдання на бакалаврську роботу

Київський національний університет імені
Тараса Шевченка
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

Назва теми: «Веб-сервіс з пошуку місць для планування туристичних маршрутів»

Освітня програма: Прикладні інформаційні системи
Спеціальність: Комп'ютерні науки

ПІБ	Підпис
Радченко Володмир Юрійович	

Назва роботи українською та англійською мовами

Веб-сервіс з пошуку місць для планування туристичних маршрутів. Web service for searching places to plan tourist routes.

Мета бакалаврської кваліфікаційної роботи, завдання

Мета роботи: проектування та розробка веб-сервісу для планування туристичних маршрутів.

План роботи:

1. Сучасні технології розробки web-сервісів
2. Проектування web-сервісу для побудови туристичних маршрутів
3. Розробка web-сервісу для побудови туристичних маршрутів

ПІБ, ступінь, звання наукового керівника роботи: Броварець Олександр Олександрович, доцент, кандидат технічних наук

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	заява
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	05.03.2021	
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	09.04.2021	
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	07.05.2021	
9.	Подання роботи у першому варіанті	11.05.2021	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**24.05.2021**	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	16.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

РЕФЕРАТ (Анотація)

Радченко Володимир Юрійович

Веб-сервіс з пошуку місця для планування туристичних маршрутів.

Дипломне дослідження складається з трьох розділів та дев'яти підрозділів. Також робота містить вступ, висновки до кожного розділу, списку використаних джерел та додатків. Робота налічує 3 таблиці, 18 рисунків та 3 додатки.

Метою роботи є проектування та розробка веб-сервісу для планування туристичних маршрутів.

Об'єктом дослідження є підходи до побудови веб-сервісів.

Предметом дослідження є розробка веб-сервісу планування туристичних маршрутів.

В першому розділі розглядаються питання впливу сучасних інформаційних технологій на туристичну галузь в цілому, та зокрема зміни ролі споживачів в організації сучасного туризму. Також розглядаються підходи до розробки веб-сервісів. В другому розділі проектується функціональна модель веб-сервісу та розробляється його архітектура. В третьому розділі проектується інтерфейс веб-сервісу для планування туристичних маршрутів та описується розробка і тестування системи.

Практичне значення даного дослідження полягає в наданні користувачам зручного безкоштовного сервісу для побудови туристичних маршрутів.

Ключові слова: веб-сервіс, розробка, планування, проектування, туристичний маршрут.

ABSTRACT

Radchenko Volodymyr Yuriyovych

Web service for finding a place to plan tourist routes.

The honors degree consists of three sections and nine subsections. The work also contains an introduction, conclusions to each section, a list of used sources and appendices. The work has 3 tables, 18 figures and 3 appendices.

The purpose of the work is to design and develop a web service for planning tourist routes.

The object of research is approaches to building web services.

The subject of the study is the development of a web service for planning tourist routes.

The first section examines the impact of modern information technology on the tourism industry in general, and in particular the changing role of consumers in the organization of modern tourism. Approaches to the development of web services are also considered. The second section projects a functional model of a web service and develops its architecture. The third section designs the web service interface for planning tourist routes and describes the development and testing of the system.

The practical significance of this study is to provide users with a convenient free service for building tourist routes.

Keywords: web service, development, planning, design, tourist route.

ЗМІСТ

ВСТУП

Застосування інформаційних технологій змінило функціонування бізнесу у всьому світі. Їх вплив відчувався здебільшого в інформаційно залежних галузях, і туризм належить до таких галузей. Туроператори, туристичні агенти та прямі постачальники туристичних послуг можуть все частіше використовувати інформаційні технології для передачі зображень місця призначення потенційним споживачам.

Також зазначимо, що поява Web 2.0, розповсюдження недорого доступу до Інтернету змінило роль споживачів туристичних послуг. Зокрема інтерактивні форми технологій Web 2.0 дають змогу споживачам взаємодіяти між собою та формувати власні туристичні потреби.

Важлива характеристика веб-сайтів 2.0 передбачає можливість включати різні технології та додатки на веб-сайті, покращувати функціональність. Ця розширена функціональність в першу чергу пов’язана з можливістю таких сайтів публікувати та демонструвати різноманітний вміст, який сформований користувачем або давати сайту можливість залучати інформацію, що поступає від третіх сторін.

Питання використання інформаційних технологій в туристичній галузі досліджувались Гуляєвим В.Г., Чудновським А.Д., Кузиком С.П. та іншими. Так, можливості сучасних інформаційних технологій в туризмі розглядалися Роглєвим Х., Агафоновою Л. В роботі С Мельниченка розглядається практичні інструменти інформаційних технологій, що застосовують в туризмі. В наукових роботах Квартальнова В, Жєлєні М. вказується, що сучасні інформаційні-комунікаційні технології забезпечують стрімке зростання обміну інформацією між учасниками туристичного ринку [12; 13].

Однак зазначимо, в більшості випадків теоретичні дослідження та новітні розробки програмних продуктів в туристичній області в Україні в основному розраховані на використання підприємствами, що працюють в цій області.

Більшість туристичних операторів мають свої сайти, на яких турист може обрати маршрут своєї подорожі, який прокладений саме фірмою. В той же час поки що майже відсутні програмні продукти, що дають туристу самостійно визначити маршрут своєї подорожі, а також обмінюватись думками з приводу того чи іншого маршруту з іншими користувачами.

Такі системи повинні об’єднувати інформацію, яка міститься в різних джерелах. Тому проблема зв'язку таких різнорідних даних, а також створення способів, які дозволяють отримувати їх у зручному для подальшої обробки вигляді. Концепція веб-сервісів призвана вирішити цю задачу об'єднання, інтеграція різнорідних систем на основі відкритих стандартів. Звідси випливає актуальність даної роботи.

Метою роботи є проектування та розробка веб-сервісу для планування туристичних маршрутів.

Для досягнення мети дослідження потрібно вирішити такі завдання:

1. Дослідити вплив інформаційних технологій на туристичну галузь.
2. Вивчити основні поняття та підходи до розробки web-сервісів.
3. Провести проектування програмної розробки на основі об’єктно-орієнтованого підходу.
4. Розробити архітектуру web-сервісу для планування туристичних маршрутів.
5. Розробити web-сервіс для планування туристичних маршрутів та провести його тестування.

Об’єктом дослідження є підходи до побудови веб-сервісів.

Предметом дослідження є розробка веб-сервісу планування туристичних маршрутів.

Наукова новизна цієї роботи полягає у застосуванні сучасних підходів до розробки веб-сервісів для туристичної галузі.

Методи дослідження. У даній дипломній роботі було використано такі методи дослідження: аналіз, порівняння, індукція, дедукція та аналогія.

Практичне значення даного дослідження полягає в наданні користувачам зручного безкоштовного сервісу для побудови туристичних маршрутів.

Враховуючи, що в Україні таких програм мало даний сервіс повинен зайняти нішу програм для туристів, які бажають самостійно організовувати свій відпочинок.

Робота складається з трьох розділів та дев’яти підрозділів. Також робота містить вступ, висновки до кожного розділу, списку використаних джерел та додатків.

В першому розділі розглядаються питання впливу сучасних інформаційних технологій на туристичну галузь в цілому, та зокрема зміни ролі споживачів в організації сучасного туризму. Також розглядаються підходи до розробки веб-сервісів. В другому розділі проектується функціональна модель веб-сервісу та розробляється його архітектура. В третьому розділі проектується інтерфейс веб-сервісу для планування туристичних маршрутів та описується розробка і тестування системи.

РОЗДІЛ 1 СУЧАСНІ ТЕХНОЛОГІЇ РОЗРОБКИ WEB-СЕРВІСІВ

1.1 Використання Web-технологій в туристичній діяльності

Розвиток інформаційно-комунікаційних технологій (ІКТ) суттєво впливає на зміну бізнес-процесів в усіх сферах економічної діяльності, створюючи нові способи управління, змінюючи структуру галузі та формуючи цілий спектр шансів для підприємств. ІКТ, надаючи інструменти для збільшення, управління та розподілу туристичних пропозицій, дозволяють користувачам визначати, модифікувати та купувати туристичні товари [7].

Основною перевагою ІКТ є встановлення більш ефективних зв’язків між різними ланками комерційного ланцюга поставок, пропонуючи легкий доступ у режимі реального часу до даних, що обробляються в певних компонентах даного ланцюга.

ІКТ значно змінили роль кожного гравця в процесі на туристичному ринку, оскільки клієнти туристичних послуг стають не просто споживачами послуг, а активно формують цю послугу, наприклад, створюючи нові види та змінюючи існуючі види туристичних маршрутів [1; 13].

Туризм представляє собою економічну галузь, в якій інформація відіграє надзвичайно важливу роль. Створення, пошук, зберігання, отримання та передача інформації є підґрунтям туристичної галузі, тому технології, що базуються на обробці інформації суттєво впливають на організацію туристичної діяльності [12].

Протягом останніх років відбувається експоненційне збільшення використання Інтернету, а також інструментів, що дозволяють користувачам Інтернету як створювати, так і розповсюджувати різний контент (мультимедіа, текстову інформацію тощо). Ці інструменти, які іноді окреслюють як Web 2.0, можуть розглядатися як інструменти масової співпраці, оскільки вони надають можливість користувачам Інтернету активно брати участь і одночасно

співпрацювати з іншими користувачами Інтернету для виробництво, споживання та розповсюдження інформації та знань, що поширюються через Інтернет.

Розвиток Інтернет-технологій з їх універсальною доступністю та інтерактивним характером змінив поведінка споживачів та ставлення до традиційних моделей туризму (рис.1.1). Зокрема змінився спосіб, яким туристичні товари та послуги поширюються [48]:

– стає все більше прямих продажів споживачам;

– збільшується кількість посередників та третіх сторін, що продають спеціалізовані туристичні товари (наприклад, прокат автомобілів);

– клієнти мають доступ до тієї ж інформації та каналів розповсюдження, що і деякі туроператори всередині існуючої моделі;

– змінюються продукти та послуг, що пропонуються, відповідно до мінливих смаків та вимог споживачів;

– збільшується прозорість витрат.

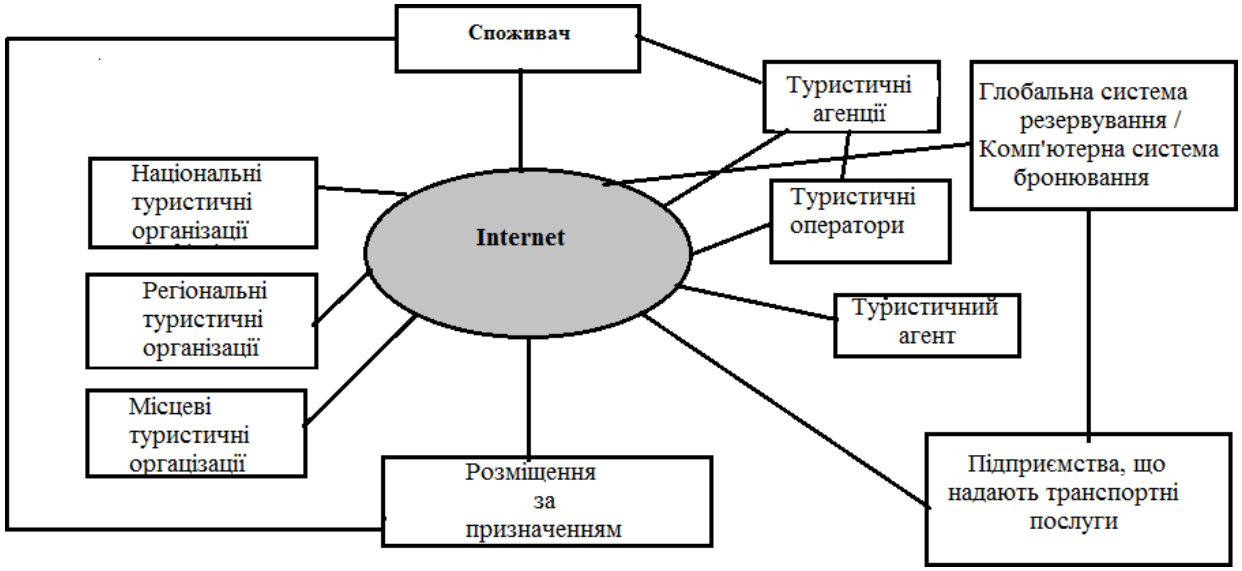

Рисунок 1.1 – Канали розподілу замовлень в туризмі в умовах Інтернету)

Джерело: [47]

Найбільший вплив інформаційних технологій на туристичну індустрію в даний час можна побачити через системи бронювання, що використовуються клієнтами через eTourism. Постачальники послуг проживання – це найбільша

група туризму. Більшість постачальників житла – це малі підприємства, де 95,5% усіх організацій мають менше дев’яти працівників. Більшість таких підприємств завдяки Інтернету отримують розширений доступ споживачів, а також розширюють свої канали збуту [47].

Зазвичай зараз готелі будь-якого розміру мають web-сайти для маркетингу та управління взаємовідносинами з клієнтами (CRM), а також, можливо, включають функцію он-лайн бронювання. Крім того на сайті розміщається реклама, рекламні огляди. Це дозволяє підвищити інтерактивність та комунікацію з клієнтами, максимізувати продажі та підвищити потенціал отримання прибутку.

У більшості готелів будь-якого розміру також вбудовані комп'ютерні системи бронювання (CRS) та системи управління майном (PMS) для управління пропускною спроможністю, контролю запасів приміщень та надання можливостей управління прибутком за рахунок максимального заповнення номерів. Ці системи також включають функції бази даних для загального управління готелем, такі як прогнози та управління фінансами [47].

Впровадження глобальних системи резервування (наприклад, Sabre, Amadeus, Gailileo та Worldspan) для отримання та оприлюднення бронювань дозволяє фірмі охопити глобальну аудиторію, щоб максимізувати бронювання та прибутковість своєї діяльності.

Зміст та інформація, яку генерують користувачі на основі концепції Web 2.0, має значний вплив на профіль, очікування та поведінку людей, що приймають рішення, але також на моделі електронного бізнесу, яку бізнес повинен розвивати та / або адаптуватися. Туристична галузь не є винятком. Навпаки, оскільки інформація є джерелом живлення туристичної індустрії, її використання та розповсюдження через концепцію Web 2.0 мають значний вплив як на туристичний попит, так і на формування нових туристичних продуктів. Тим самим споживач бере участь у виробничому процесі і стає споживачем та виробником одночасно.

У туристичній галузі існують численні приклади загальних та / або спеціалізованих блогів, таких як tripadvisor.com, hotelchatter.com, igougo.com, gazetters.com (Web-журнал B2B для туристичних агентів). Будь-хто може створити web-журнал, використовуючи програмне забезпечення, яке сьогодні пропонується безкоштовно на кількох web-сайтах, його можна використовувати для публікації тексту, зображень, посилань на інші блоги, web-сторінки, аудіо- та відеофайли. Web-журнали стають дуже важливими інструментами, що впливають на пошук інформації, оскільки їх посилання, вміст і популярність можуть диктувати позицію компанії в пошуку пошукової системи. Багато мандрівників-туристів використовують web-журнали як розвагу та / або як спосіб самовираження.

Web-журнали мають силу неупередженої інформації та електронного переказу, який поширюється в Інтернеті, як вірус. Більше того, дуже ймовірно, що під час читання та обміну досвідом подорожей через web-журнали це також створює бажання подорожувати та відвідувати один і той же пункт призначення або постачальників послуг.

З іншого боку, багато туристичних компаній також активно працюють, створюючи та включаючи web-журнали на свої web-сайти. Наприклад, Marriott створив власний web-журнал на своєму Web-сайті, тоді як Starwood створив web-журнал для спілкування з бажаними гостями та підвищення їх лояльності через web-сайт thelobby.com [48].

Web-журнали, ініційовані та модеровані компанією, можуть запропонувати такі переваги: вимагати та збирати відгуки від клієнтів, проводити безкоштовні онлайн-дослідження ринку, стати визнаними експертами з певної теми, спілкуватися та оновлювати своїх клієнтів

Використання технології Wiki дозволяє користувачам створювати свій профіль та запрошувати інших, хто має подібний профіль брати участь у їхній інтернет-спільноті. Найпопулярніші web-сайти, такі як myspace.com та bebo.com відображають готовність користувачів Інтернету трансформувати web-сайти як місця для зібрання людей зі схожими профілями.

Організовані таким чином Web-сайти мають величезний вплив на формування споживчого туристичного досвіду. Сьогодні багато туристів віддають перевагу досвіду інших користувачів зі схожим профілем у поїздці та організації маршруту, що вони запланували, а інші туристи хочуть користуватися Інтернетом для того, щоб планувати групові поїздки зі своїми друзями. Такі вимоги та уподобання туристів породжують нові моделі електронного туристичного бізнесу для розповсюдження та виготовлення туристичних пакетів.

Прикладами таких інтетрнет-посередників за кордоном є сайти goseek.com (рис.1.2), traveltogether.com (рис.1.3).

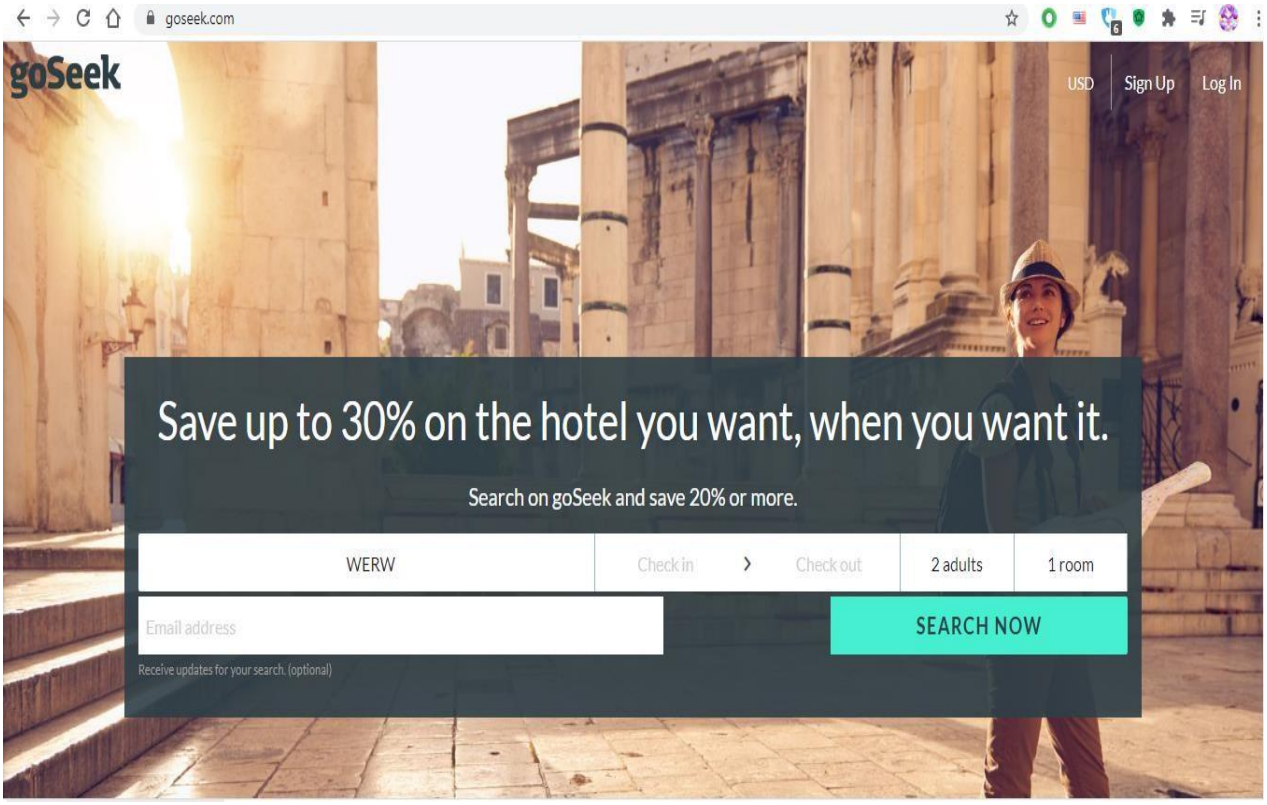

Рисунок 1.2 – Головна сторінка сервісу GoSeek

Вони надають Інтернет-користувачам інструменти для створення онлайн-маршруту, обміну ним та надсилання його електронною поштою друзям для редагування, та групового бронювання.

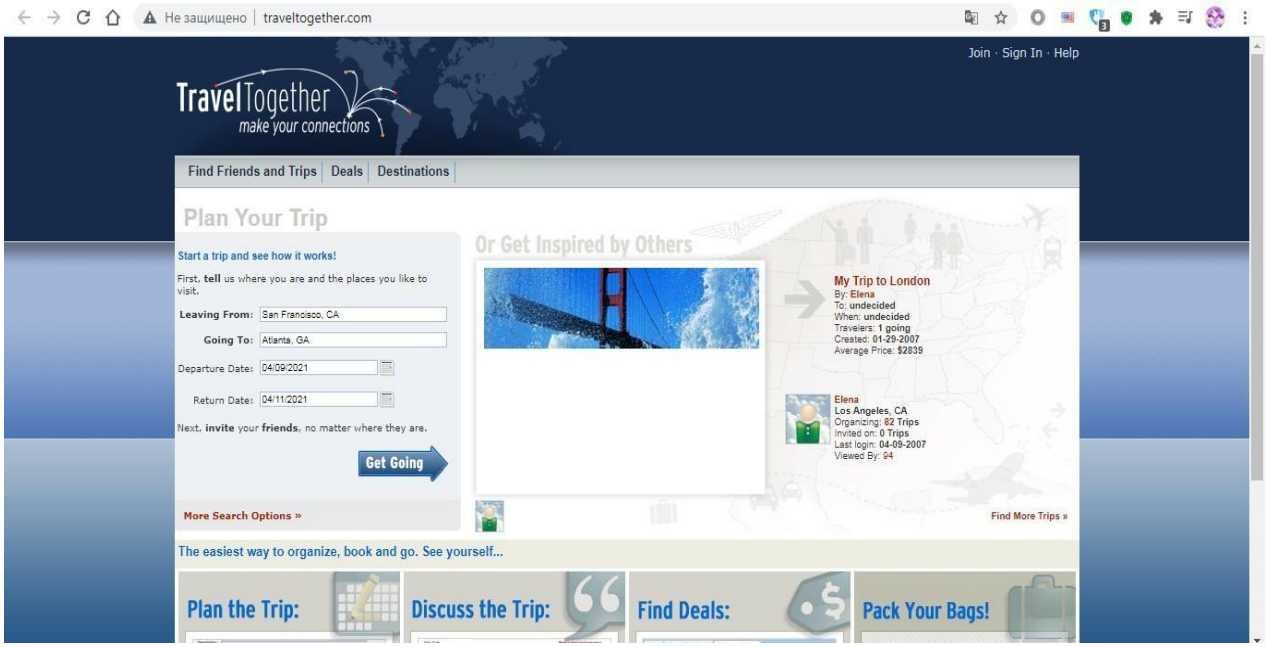

Рисунок 1.3 – Головна сторінка сайту traveltogether.com

З розвитком цифрових карт були створені можливості зв’язати карти з базами даних, які зберігають дані про об’єкти та їх атрибути. Отримання інформації щодо туристичних об’єктів стає більш швидким, повним та ефективним. Таким чином в туристичний бізнес впроваджуються геоінформаційні системи (ГІС) [20].

Важливість ГІС полягає в її здатності пов'язувати просторово пов'язані види діяльності та інтегрувати просторову та інші типи інформації в єдину систему, забезпечуючи тим самим стабільні рамки для загального аналізу.

– застосування ГІС у туризмі базується на таких процесах:

– збір, маніпулювання та зберігання інформації, що стосується району, в якому проводиться інвентаризація туристів та рекреаційних ресурсів,

– тестування стану за допомогою цифрових інтерактивних карт, в процесі визначення найбільш придатних місць для розвитку туристичної галузі,

– інтеграція баз даних в аналіз тенденцій та наслідків туристичної діяльності у районі;

– створення маршрутів та пересування туристів,

– просторовий аналіз зв’язків, що існують між різними туристичними та рекреаційні ресурси,

– фізичне моделювання потенційних впливів туризму на область дослідження.

Таким чином, сучасні Web-технології дозволяють користувачам Інтернету стати співвиробниками, співдизайнерами, співмаркетологами та співрозповсюджувачами туристичного досвіду та послуг. Саме їх використання забезпечує для споживача туристичних послуг їх персоналізацію, швидкість та комфортність отримання,

1.2 Поняття Web-сервісу, принципів його побудови

При розробці сучасного програмного забезпечення для Інтернету все більше уваги приділяється розробці додатків типу Web-сервіс. Саме такі додатки використовуються купівлі квитків, оплати комунальних послуг, бронювання місць в готелі.

Під Web-сервісом розуміють:

– програмну систему, що підтримує інтероперабельну взаємодію між машинами в Мережі;

– програмну систему, яка може бути опублікована, виявлена і використана в Інтернеті за допомогою XML-протоколів;

– спосіб зв'язку програмних систем з використанням XML і HTTP;

– модель бізнесу, коли організація надає доступ по мережі Інтернет до своїх обчислювальних, інтелектуальним і виробничим ресурсам широкому колу клієнтів [2; 19; 23].

У порівнянні з іншими способами зв'язку програмних систем Web-сервіси мають ряд переваг:

– відкриті стандарти;

– робота через один порт по HTTP;

– інтероперабельність, відсутність прив'язки до платформи;

– простота реалізації клієнтів і серверів деяких протоколів Web-сервісів.

Основним недоліком Web-сервісів є великий обсяг мережевого трафіку.

Принципова схема конфігурації сервісів наведена на рис.1.4. Завдяки їй вирішується три основні завдання: публікація в інтернеті відомостей про наявні сервісах, пошук Web-сервісів і взаємодія між сервіс-провайдером і споживачем.

Рисунок 1.4 – Принцип організації роботи Web-сервісів

Джерело: [23]

Робота Web-сервісів побудована на використанні декількох відкритих стандартів: XML – розширювана мова розмітки, призначений для зберігання та передачі структурованих даних; SOAP – протокол обміну повідомленнями на базі XML; WSDL – мова опису зовнішніх інтерфейсів Webсервісів на базі XML; UDDI – універсальний інтерфейс розпізнавання, опису та інтеграції (Universal Discovery, Description, and Integration) [33].

Зв'язок між Web-сервісами та їх клієнтами здійснюється за допомогою повідомлень в форматі XML.

Споживач сервісів відправляє запит на пошук сервісу сервіс-брокеру – спеціалізованій програмі або автоматизованій паспортній системі, яка проводить пошук в довідниках і реєстрах (UDDI), які розглядаються в якості аналогів жовтих сторінок, а при необхідності і безпосередньо в інтернеті [23, с.51].

Universal Description, Discovery, and Integration (UDDI). Щоб полегшити пошук Web-сервісів в глобальній мережі, відомості про них розміщують в реєстрі, загальноприйнятим стандартом якого є заснований на XML формат універсального опису, пошуку і об'єднання сервісів – UDDI [23].

Web Services Flow Language (WSFL). Являє собою мову опису функціональності сервісів, що базується на XML. У стандартній формі він визначає склад і формат даних, які повинні бути введені в запиті на сервіс, а також аналогічний склад вихідних даних [23].

Передбачається, що виробники сервісів публікують дані про їхню наявність у форматі WSDL, після чого їх виявлення стає можливим за допомогою програм-агентів, які здійснюють їх умовний пошук (вносяться в програму пошуку умови до деякої міри аналогічні технічним завданням), і, таким чином, виконають функції брокера. У реальності далеко не всі сервіси доступні за допомогою звернення до реєстрів, і пошук в цьому випадку представляє собою більш складне завдання.

За знаходженні сервісу здійснюється зв'язок між споживачем і провайдером, і сервіс надається споживачеві на узгоджених умовах.

Simple Object Access Protocol (SOAP) – заснований на XML протокол для обміну інформацією в розподіленому обчислювальному середовищі, який складається з трьох частин:

– перша – конверт, який визначає, що міститься в повідомленні, і як його обробляти;

– друга частина – набір правил для визначення запитів на дані, що відповідають додаткам;

– третя – угода про подання процедури віддаленого виклику і відгуку [52].

SOAP може працювати в комбінації з іншими протоколами, особливо HTTP.

Сумісність і відтворюваність сервісів, є властивостями, що забезпечують принципово нову якість і можливості інтеграції в Інтернеті, що виводить глобальну мережу в цілому на новий рівень розвитку.

Архітектурні стилі використання Web-сервісів визначають взаємодію компонентів інформаційних систем. Найбільш поширеними архітектурними стилями при проектуванні Web-сервісів є RPC, SOA і RESTful.

RPC (Remote Procedure Call – віддалений виклик процедур) характеризується тим, що методи, які проектуються, в своїх іменах і параметрах містять як вказівки на бізнес-об'єкт, так і дієслово дії з ним. Як правило, на кожну дію створюється один метод, наприклад: getItems (), doItemOperation(). Параметри методів фіксовані, методи групуються в інтерфейси об'єктів. Методи викликаються синхронно.

В цілому RPC-стиль можна охарактеризувати тим, що створюваний додаток проектується так само, як якби виклики були невіддаленими. Сучасні інструменти приховують складність реалізації віддалених викликів, і програмісти не враховують їх специфіку.

Перевагами даної архітектури є:

– звичність для програмістів;

– широкий спектр інструментаріїв, IDE для швидкої розробки;

– простота проектування і реалізації Web-сервісів.

Недоліками:

– складність масштабування;

– складність побудови стійкої до збоїв системи.

SOA (Service Oriented Architecture – сервіс орієнтована архітектура) – архітектура системи зі слабо пов'язаними компонентами, коли основною одиницею зв'язку є повідомлення, а не операція [51].

Перевагами даної архітектури є:

– зменшення зв'язаності компонентів системи;

– асинхронна робота компонентів системи.

Основним недоліком даного архітектурного стилю є складність реалізації, а саме, необхідність підтримувати черги повідомлень і проектувати асинхронну роботу компонентів системи.

Принципами розробки, підтримки і використання SOA є:

повторне використання (reuse), поділ на частини (Granularity), модульність, сполучуваність сервісів (composability) і компонентизація (componentization);

– сумісність зі стандартами;

– ідентифікація і категоризація Web-сервісів, розгортання (Provisioning and delivery), моніторинг та відстеження (tracking);

– інкапсуляція;

– поділ бізнес-логіки й технології реалізації;

– єдина реалізація і доступність компонентів на рівні підприємства (enterprise-view of components);

– використання існуючих активів завжди, коли є можливість;

– управління життєвим циклом;

– ефективне використання ресурсів системи;

– зрілість і продуктивність сервісів.

RESTful (Representational State Transfer – передача стану уявлень) – проектування системи в термінах ресурсів і передачі їх уявлень методами HTTP [23,50].

Перевагами даного стилю є

– масштабування взаємодії компонентів до масштабів глобальної мережі;

– єдиний інтерфейс взаємодії компонентів, що дає можливість слабо пов'язаним між собою компонентам розвиватися самостійно (наприклад, браузери і Web-сервери);

– наявність готових проміжних компонентів для організації кешування уявлень (наприклад, кешуючий проксі-сервер) [2,28].

Ресурси і передача уявлень ресурсів є ключовими поняттями REST. Інформаційний ресурс – іменована функція R (t), яка в залежності від часу повертає безліч уявлень.

Основні завдання REST включають:

1) масштабування взаємодії компонентів;

2) спільність інтерфейсів (наприклад, розширюваність HTTP своїми заголовками і методами);

3) незалежне розгортання компонент;

4) проміжні компоненти для зменшення затримок, нав'язування безпеки і забезпечення підтримки застарілих систем [28].

Архітектурний стиль REST описує наступні шість обмежень, які можна застосувати до архітектури, але залишає питання реалізації окремих компонент на розсуд розробника [28].

1. Клієнт-сервер. Клієнти відокремлені від серверів єдиним інтерфейсом. Поділ функцій клієнта і сервера означає, що клієнта не хвилюють питання зберігання даних, вони є внутрішніми для кожного сервера. Переносимість клієнтського коду підвищується. Сервер не дбає про інтерфейс / стан користувача, так що сервер може бути простим і добре масштабованим. Сервери та клієнти можуть бути замінені, розроблені незалежно, до тих пір поки інтерфейс не змінюється.

2. Відсутність стану (stateless). Взаємодія клієнта і сервера обмежена таким чином, що контекст клієнта не зберігається на сервері між запитами, кожен запит від клієнта містить всю інформацію, необхідну для обробки запиту, весь стан міститься на клієнті. Це не тільки дозволяє зробити сервери більш доступними для моніторингу (не треба відстежувати стан клієнтів), але також робить їх більш надійними при часткових збоях та поломці мережі, збільшує їх масштабованість.

3. Кешування. Наприклад, в Інтернеті клієнти можуть кешувати відповіді. Таким чином, відповіді від сервера явно чи не явно повинні визначати себе як

кешувальні чи ні, щоб запобігти повторному використанню клієнтами застарілих або неправильних даних у відповідь на подальші запити. Правильно організоване кешування частково або повністю виключає деякі клієнт-серверні взаємодії, ще більш збільшуючи надійність і продуктивність.

4. Єдиний інтерфейс. Чіткі вимоги до єдиного інтерфейсу спрощують і архітектуру, що дозволяє кожній частині розвиватися незалежно.

5. Багаторівневі системи. Клієнт не може точно сказати, з'єднаний він з кінцевим або проміжним сервером. Проміжні сервери можуть покращувати масштабування за допомогою балансування навантаження, надання загального кешу, а також можуть нав'язувати політику безпеки.

6. Код за запитом (необов'язкове обмеження). Сервери можуть тимчасово розширювати або змінювати функціональність клієнта, передаючи йому логіку, яку він може запустити на виконання.

Суть архітектурного стилю RESTful для Web-сервісів можна сформулювати у вигляді принципів побудови REST-інтерфейсу.

Принципи, на яких має будуватися будь-який інтерфейс, вважаються фундаментальними.

1. Ідентифікація ресурсів. Окремі ресурси ідентифікуються в запитах, наприклад використанням URI в REST-системах на основі Інтернету. Ресурси самі по собі концептуально відмінні від уявлень, які повертаються клієнту. Наприклад: сервер не посилає свою базу даних, але шле, можливо, HTML, XML або JSON (фрагмент коду JavaScript), який являє собою записи бази даних, залежно від параметрів запитів і реалізації сервера.

2. Маніпуляція ресурсами через їх подання, коли клієнт володіє поданням ресурсу (включаючи будь-які метаданні) і у нього є достатньо інформації, щоб змінити або видалити ресурс на сервері.

3. Повідомлення, що самоописуються. Кожне повідомлення містить достатньо інформації, яка описує, як обробити повідомлення, наприклад, який парсер запустити. або, медіатипи (раніше відомі як MIME-типи). Якщо

необхідно заглянути в вміст повідомлення, щоб зрозуміти його, то це повідомлення не описує саме себе.

4. Стан додатку моделюється з використанням гіпермедіа. Якщо ймовірно, що клієнт захоче отримати доступ до родинних ресурсів, вони повинні бути ідентифіковані поданням, яке повертається сервером. Наприклад, можна передавати їх URI в достатньому контексті, такому, як гіперпосилання [28].

Структура реалізації Web-сервісу, що містить у собі програмні елементи, їх зовнішні властивості і взаємозв'язки, є важливою при проектуванні Web-сервісів. Це пояснюється тим, що в ній викладаються початкові проектні рішення, вона визначає обмеження реалізації та організаційну структуру, що розробляється, крім того, програмна архітектура дозволяє більш точно планувати ресурси, необхідні для створення програмного проекту. програмна архітектура містить в собі структури, з яких складаються великі програмні системи. Вибір архітектурного рішення суттєво залежить від таких факторів, як:

– зацікавлених в системі осіб, таких як розробники, замовники;

– компанії-розробника з виробленими в ній підходами до реалізації програмних проектів;

– досвіду і звичок архітекторів програмної системи;

– нового покоління, технічної бази, що включає програмні й апаратні ресурси розробки програмного забезпечення.

1.3 Огляд існуючих засобів розробки Web-сервісу

Існує безліч платформ і мов програмування для створення Web-сервісів. Їх використання суттєво залежить, як від обраного архітектурного стилю, так і від наявних інструментів. Розглянемо спочатку мови програмування, що використовуються при розробці Web-сайтів.

HTML – це не мова програмування і не мова оформлення документів. Це, в першу чергу, засіб розмітки тексту. Мова HTML містить достатню кількість елементів, що дозволяють оформити документ [42].

CSS (Cascading Style Sheets) – мова таблиць каскадних стилів. Вона розроблена для того, щоб розширити можливості по оформленню Web-сторінок [16].

CSS використовується Web-розробниками для завдання зовнішнього вигляду (шрифтів, кольорів, відступів, розташування та ін.) Web-сторінок. CSS розроблена для відділення основного вмісту документа (написаного на мові розмітки, наприклад HTML) від оформлення цього вмісту (написаного на CSS). Таке відділення надає Web-розробникам велику гнучкість, спрощує завдання зовнішнього вигляду документів та оформлення повторюваних елементів розмітки.

При використанні HTML і CSS важливо розуміти, наступне:

– HTML-код формує текст логічно, тобто задає структури Web-сторінки: розташування і порядок проходження абзаців, графічних зображень, рядків і осередків в таблиці і особливе значення окремих фрагментів тексту.

– таблиці стилів CSS формують тексти фізично, тобто задають уявлення Web-сторінки: яким шрифтом будуть набрані звичайний текст абзаців, яким кольором виділити заголовки, чи будуть у таблиці рамка тощо.

Правила хорошого тону Web-дизайну вимагають, щоб уявлення Web-сторінки було відокремлено від її структури. Тому професійні Web-дизайнери по можливості виносять визначенн стилів CSS в окремі файли (таблиці стилів).

До того ж, HTML-код, що не захаращений визначеннями стилів, стає більш читабельним. Каскадні таблиці стилів по суті своїй динамічні. Вони дозволяють визначати, як буде виглядати документ при завантаженні і не більше того. Але властивості Web-сторінок, створених за допомогою CSS, можна динамічно змінювати за допомогою мови JavaScript.

Мова програмування Java є однією з найбільш затребуваних в останні кілька років, і ця тенденція зберігається і досі на 2021 рік. Java – типізована об'єктно-орієнтована мова, яка отримала найбільшу популярність під розвитком компанії Oracle. Перший офіційний випуск мови відбувся в 1995 році [43]. Популярність цієї мов обумовлене такими причинами:

– мова постійно удосконалювався і зараз в ній реалізована велика кількість різних просунутих технологій і рішень;

– мова досить проста, її синтаксис є зразком мінімалізму. Це використовується багатьма авторами в навчальних цілях [9].

Python є високорівневою мовою, головне завдання якої спочатку було дати розробнику максимальну продуктивності праці при мінімумі виникаючих проблем і хороша читаність одержуваного коду. Використовуваний в Python синтаксис гранично спрощений та мінімізований, тим не менш, його цілком достатньо для реалізації практично будь-яких завдань. Наявна об'ємна стандартна бібліотека для Python включає в себе масу корисних функцій [9].

Перевагами Python є:

– наявність великої кількості фреймворків, готових рішень і функцій;

– відмінний і зрозумілий синтаксис.

Дана мова дуже добре підходить для реалізації різних математичних обчислень, створення системних інструментів та іншого.

Головне завдання мови C# – створення додатків, оптимізованих під Microsoft. NET Framework. Синтаксис C# дуже близький до C ++ і Java. Мова має масу зручних функцій: анонімні функції, ітератори, події, властивості, винятки, коментарі тощо [25].

Головними її перевагами є:

– висока швидкість розробки;

– широкі можливості оптимізації написаного коду та інші.

JavaScript. На відміну від Java, javascript має більш вузьке застосування, але, тим не менш, він дуже популярний. Ця мова відноситься до ООП, забезпечує мультипарадигму, підтримує роботу в імперативному та функціональному стилі. В основному застосовується у сфері Web-розробок на – для реалізації будь-яких функцій браузерів, скриптів на інтернет сайтах, створення інтерактивних елементів Web-сторінок і тому подібних завдань [17].

При роботі з JavaScript в код Web-сторінки потрібно додати два компоненти: сам скрипт і процедуру, яка буде його запускати.

Популярність мови JavaScript пов'язана з її широкими можливостями по взаємодії з елементами Web-сторінки без її перезавантаження. Це дозволяє ховати і показувати фрагменти дизайну, переміщати їх і міняти оформлення. Шляхом таких дій можна створювати презентаційні ефекти, меню, обробляти дані форм і керувати вмістом.

JavaScript підтримує повноцінну роботу з cookies – невеликими текстовими файлами на локальному комп'ютері, в яких зберігається технічна інформація.

Cookies можна використовувати для збереження дати останнього відвідування читача, паролів, а також будь-якої інформації про дії користувача на сайті. Подібне застосування дозволяє персоналізувати сайт і зробити його більш зручним для відвідувачів.

Перевагами JavaScript є:

– використання в кожному сучасному браузері;

– простота в написанні скриптів;

– постійне вдосконалення мови.

Недоліком JavaScript є знижений рівень безпеки через повсюдний і вільний доступ до вихідного коду популярних скриптів, а також значна кількість можливих помилок при написанні коду.

Мова PHP широко застосовується в сфері Web-технологій. Це скриптова мова, за допомогою якої розробляються різні додатки, в основному для роботи

серверів, динамічних сторінок сайтів. За допомогою PHP досить просто працювати з базами даних, що дуже корисно при обробці великих масивів однотипної інформації. Багато адміністраторів серйозних ресурсів використовують у своїй діяльності в основному саме PHP [9; 11].

Perl – високорівнева та інтерпретована мова, головною перевагою якої є робота з текстовою інформацією. Вона має багато подібностей з C, доповнена багатьма додатковими модулями, які дозволяють використовувати Perl в Web-технологіях, програмуванні ПЗ та ігор, в системному адмініструванні, і навіть при розробці графічних інтерфейсів [25].

Існує три варіанти побудови Web-сервісу:

1. Написати власні програми, що створюють шаблони і реалізують необхідні функції адміністрування.

2. Скористатися допомогою сторонніх розробників для створення сайту "під ключ".

3. Скористатися готовим рішенням.

В Інтернеті існують готові рішення, що дозволяють реалізовувати компромісне рішення між низькою вартістю статичних Web-сервісів і високою гнучкістю динамічних. Завдяки подібним системам різко підвищилася керованість Web-сервісом, і значно знизилися витрати на адміністрування Web-сервіса. Таким варіантами є фреймворки та системи управління контентом [6].

Фреймворк – це каркас для створення додатків певного роду, що викликає функції (методи), написані програмістом і реалізують логіку роботи програми. Відмінність фреймворка від простої бібліотеки функцій полягає в тому, що фреймворк викликає написані розробником функції, а не він його.

Система управління контентом (Content Management System, CMS) – програма для автоматизації управління Web-сайтом [24].

На відміну від фреймворка система управління контентом містить готові модулі для створення типового сайту, але, як правило, є менш гнучкою і зручною для розробки, ніж фреймворк.

Системи управління контентом з особливо хорошим модульним дизайном, розвиненим і гнучким API іноді називають Content Management Framework (CMF), стираючи межу між CMS і фреймворком.

Фактично CMS – це движок, на якому створюється Web-сервіс. Принцип роботи будь-якого движка простий. Користувач системи додає контент на Web-сервіс. Вся інформація, яку ввів користувач, зберігається в базі даних або файлах. Коли відвідувач заходить на Web-сервіс, інформація читається з бази даних і відображається на Web-сервісі. Вид відображення інформації залежить від шаблону [24; 29].

Шаблон Web-сервісу – це заготовка дизайну Web-сервісу, без наповнення її інформацією. Майже у всіх CMS шаблони Web-сервісу легко змінюються [24].

У багатьох движках є система модулів. Тобто, функціонал системи можна розширити, підключаючи додаткові модулі. Наприклад, модуль "Чат" або модуль "Зворотній зв'язок" тощо. Модулі часто називають плагінами, розширеннями або доповненнями.

Найбільш поширеними є безкоштовні CMS. Серед найвідоміших - WordPress, Joomla, Drupal, Blogger, OpenCart та інші. Сфери їх використання найрізноманітніші – від простих односторінкових Web-сервісів, до складних інтернет-магазинів.

Розглянемо основні CMS [21; 25; 26].

Wordpress добре русифікований, для нього написано безліч додаткових модулів і зроблено безліч шаблонів. Звичайно, можливо і самим внести зміни в дизайн.

Мінуси Wordpress типові для популярних CMS – не дуже швидка робота сайту, можливість збоїв при високій відвідуваності і періодичне виявлення тих чи інших дірок в скрипті. Тому деякі користувачі навіть при створенні блогів віддають перевагу все ж більш простим і легким движкам. Також, очевидно, Wordpress навряд чи підійде для складного сайту з великою функціональністю, порталу, інтернет-магазину тощо.

Друпал (Drupal) підійде для створення форумів, блогів (можливо, на багато користувачів), онлайн-енциклопедій, сайтів спільнот. Однак система не є універсальною потребує спеціальних знань і не надає готових простих рішень. Однак, з іншого боку надає користувачу гнучкість та можливість реалізовувати власне бачення під час розробки.

Джумла (Joomla) – досить популярна CMS, яка має широку сферу застосування. Для Joomla розроблена величезна кількість модулів, включаючи форуми, чати, блоги, інтернет-магазини тощо. Joomla є універсальною CMS. Також для Joomla існує величезна кількість шаблонів. Дана система управління контентом, написана на мові PHP і використовує як сховище змісту базу даних MySQL. Joomla є вільним програмним забезпеченням, захищеним ліцензією GPL. Однією з головних особливостей Joomla є відносна простота управління при практично безмежних можливостях і гнучкості при розробці сайтів.

Важливою особливістю системи є мінімальний набір інструментів при початковій установці, який збагачується в міру необхідності. Це знижує захаращення адміністративної панелі непотрібними елементами, а також знижує навантаження на сервер і економить простір на хостингу.

Однак при реалізації більш складних нешаблонних Web-сервісів краще використовувати фреймiорки. Одним з таких є Symfony.

Symfony – вільний PHP фреймворк для швидкої розробки Web-додатків і виконання рутинних завдань Web-програмістів. Розробка і підтримка фреймворку спонсорується французькою компанією Sensio.

Symfony складається з набору не пов'язаних між собою компонентів, які можна використовувати повторно в проектах.

За допомогою Symfony було розроблено безліч великих проектів:

- систем управління контентом: Magento, Drupal, Opencart;
- сервіс соціальних закладок Delicious;
- французький відеохостинг Dailymotion;
- движок форуму phpbb.

Symfony дозволяє встановлювати сторонні пакети, бібліотеки, компоненти і налаштовувати їх за допомогою конфігурації в форматах YAML, XML, PHP, а також .env файлах.

Symfony не забезпечує компонент для роботи з базою даних, але забезпечує тісну інтеграцію з бібліотекою Doctrine.

Symfony надає функцію поштової програми на основі популярної бібліотеки Swift Mailer. Ця поштова програма підтримує відправку повідомлень з ваших власних поштових серверів, а також з використанням популярних поштових провайдерів, таких як Mandrill, SendGrid і Amazon SES.

Механізм інтернаціоналізації дозволяє встановити і здійснити переказ повідомлень Web-додатку на основі вибраної мови або країни.

Symfony пропонує систему логування помилок додатки, а також підключити бібліотеку логування Monolog.

Перевагами даного фреймворку є:

– Потужна екосистема навколо фреймворка, з хорошим співтовариством і безліччю розробників.

– Хороша і постійно оновлювана документація для всіх версій фреймворка.

– Безліч різних не пов'язаних компонентів для повторного використання.

– Механізм функціональних і модульних тестів для знаходження помилок в Web-додатку.

Отже, даний фреймворки підходить для складних і навантажених Web-проектів електронної комерції, в тому числі електронного туризму.

Розглянемо Web-сервери, що застосовуються при реалізації Web-сервісів.

Web-сервер Apache є самим поширеним в світі серед серверів. Причин такої популярності багато. Перш за все, це можливість вільно отримати його як з основного сервера проекту Apache, так і з "дзеркал", розташованих у багатьох країнах світу. . Є докладна документація по налаштуванню і адмініструванню, включаючи FAQ. В рамках даного проекту ведеться докладний облік і виправлення знайдених помилок, чому присвячено кілька сторінок сервера.

Багато розробники модифікують код Apache, вносячи додаткові функції, і пропонують для вільного поширення свої розробки. Зокрема, є версії Apache, в які додані функції по роботі з документами з кирилицею.

Висновки до розділу 1.

Проведений аналіз показав, що в сучасній туристичній галузі досить широко використовуються інформаційно-комунікаційні технології, що суттєво змінило ролі всіх учасників комерційного циклу. З використанням Web-технології споживачі туристичних послуг стають активними співтворцями туристичних продуктів. Це дозволяє персоналізувати надання таких послуг і, отже, підвищити прибуток туристичних компаній.

Одним з напрямків застосування Web-технологій в туристичному бізнесі є створення Web-сервісів, що забезпечує інтерактивну взаємодію учасників комерційного циклу в туристичній галузі.

Вибір архітектури Web-сервісу та засобів його розробки суттєво залежить від розробників та замовників системи і вливає на успішність програмної розробки

РОЗДІЛ 2 ПРОЕКТУВАННЯ WEB-СЕРВІСУ ДЛЯ ПОБУДОВИ ТУРИСТИЧНИХ МАРШРУТІВ

2.1 Розробка вимог до Web-сервісу

На основі проведеного аналізу сформулюємо вимоги до системи та спроектуємо її роботу.

Web-сервіс для формування туристичних маршрутів призначений для надання інформації по туристичних маршрутах і для взаємодії клієнтів сервісу між собою.

Web-сервіс повинен мати такі функціональні можливості:

– реєстрація користувачів;

– перегляд списку маршрутів;

– побудова маршрутів;

– пошук маршруту за довжиною, тривалістю, датою створення.

Web-сервіс буде розташовуватися в загальному доступі для всіх користувачів мережі Інтернет.

При реалізації і використанні web-сервісу повинні бути враховані вимоги до функціональних характеристик, параметрів технічних засобів, інформаційної та програмної сумісності.

При розробці ресурсу для підвищення якості сайту з точки зору юзабіліті потрібно користуватися правилом "Чим простіше, тим краще".

Для дизайну сайту потрібно дотримуватись таких принципів:

– простота і мінімалізм – інформація повинна подаватися в структурованому вигляді і в обсязі, який необхідний на даному етапі. Дизайн сайту повинен відповідати не рясніти графічними ефектами щоб не відволікати користувачів від інформації, що його цікавить.

– доступність – основна інформація і функції сайту повинні бути доступними для користувача на всіх сторінках сайту.

– легкість – швидкість завантаження сторінки повинна бути максимальною, але в той же час якість графічних елементів повинна відтворюватись на високому рівні.

– універсальність – сторінки сайту повинні коректно відображатися як на стаціонарних комп'ютерах так і на мобільних пристроях з виходом в інтернет.

Крім того, необхідно забезпечувати стабільно високу швидкість обробки запитів і завантаження сторінки, використовуючи мінімальну кількість ресурсів сервера.

Сервіс повинен мати досить просту навігацію, яка буде інтуїтивно зрозумілою користувачу.

Web-сервіс представляє собою клієнт-сервісний додаток.

Серверна частина Web-сервісу може бути розташований на хостингу з певним доменом. Існують різні тарифи на хостинг з різними цінами. Для того, щоб web-сервіс функціонував повноцінно потрібно вибрати для нього домен, вибрати тариф для хостингу, зареєструвати домен, оплатити замовлення, отримати доступ до вибраного домену. Після чого можна переносити web-сервіс на хостинг.

Web-браузер є клієнтською частиною, за допомогою браузера здійснюється доступ до web-сервісу. Інтерфейс повинен біти достатньо простим та доступним.

Перейдемо до визначення бізнес-вимог до роботи веб-сервісу. Для цього скористаємося об’єктно-орієнтованим підходом.

Візуальне моделювання та представлення проекту розроблюваної системи у вигляді візуальних моделей дає змогу отримати повне відображення можливостей та конфігурації розроблюваної системи. Основою такого підходу є використання UML, при якому система подається у вигляді системи об’єктів, що взаємодіють один з одним [15; 37].

Модель UML подається декількома рівнями. Самий рівень містить концептуальну модель, що дозволяє зрозуміти призначення та основні риси

системи. На цьому рівні розробляється діаграма прецедентів, яка дає змогу описати функціональність системи.

Метою розробки діаграм прецедентів є:

– визначення загальних меж і предметної області;

– формування загальних вимог до функціональної поведінки системи;

– підготовка початкової документації для взаємодії розробників системи з замовниками і користувачами.

Діаграма прецедентів представляє граф спеціального вигляду, за допомогою якого описується взаємодія між компонентами системи. Проектована система подається у вигляді множини сутностей або акторів, що взаємодіють з системою за допомогою варіантів використання. Актором (actor) або дійовою особою виступає будь-яка сутність, що взаємодіє з системою ззовні, а варіант використання описує сервіси, які система надає актору. Діаграма варіантів використання може доповнюватися текстом пояснення, який розкриває сенс або семантику складових її компонентів [37].

Кожен варіант використання визначає певний набір дій, які система здійснює при взаємодії з акторами. Сервіс, який ініціалізується за запитом користувача, є закінченою послідовністю дій. Це означає, що після того як система закінчить обробку запиту користувача, вона повинна повернутися в початковий стан, в якому готова до виконання наступних запитів. Між компонентами діаграми варіантів використання можуть існувати різні відношення, які описують взаємодію екземплярів одних акторів і варіантів використання з екземплярами інших акторів і варіантів. Один актор може взаємодіяти з декількома варіантами використання. В цьому випадку цей актор звертається до кількох сервісів даної системи. У свою чергу, один варіант використання може взаємодіяти з декількома акторами, надаючи для всіх них свій сервіс.

Для розроблювального веб-сервіса діаграма прецедентів наведена на рис.2.1.

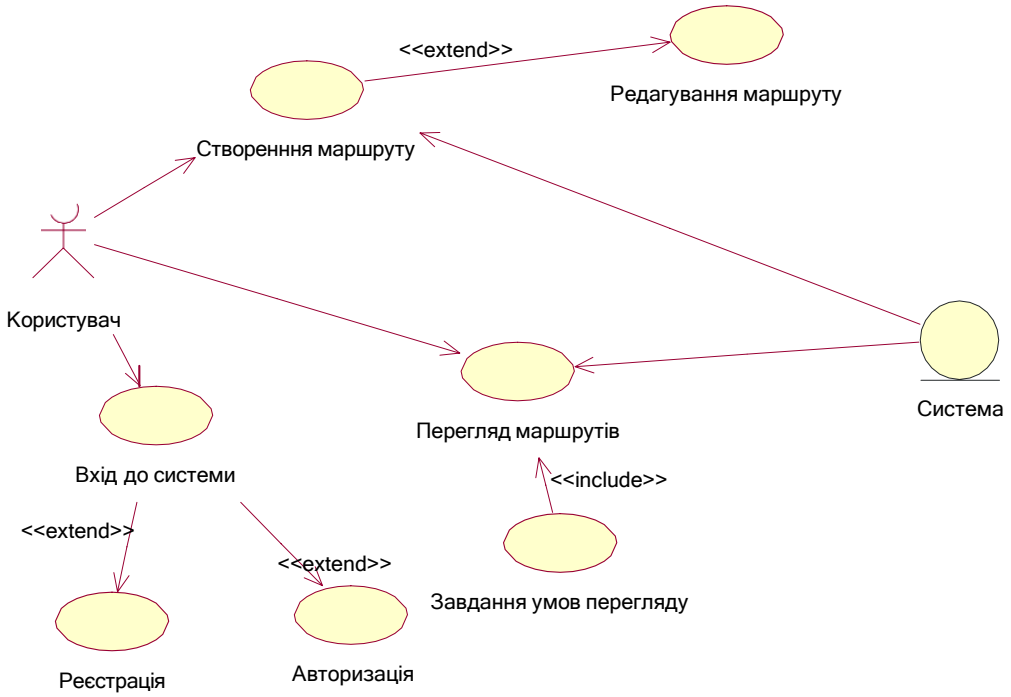

Рисунок 2.1 – Діаграма варіантів використання системи зі створення маршруту

Джерело: розробка автора

Опишемо основні варіанти використання.

Таблиця 2.1 – Опис прецеденту "Вхід в систему"

Найменування	Вхід до системи
Номер	1
Опис	Здійснення входу у веб-сервіс
Актор	Користувач, Система
Головний сценарій	Після переходу за адресою веб-сервіса перед користувачем відкривається форма для введення даних для авторизації. Користувач вводить логін та пароль. Система перевіряє дані та видає користувачу дозвіл на роботу з системою.
Альтернативний сценарій	Користувач з даним логіном та паролем не знайдений. Система відмовляє у вході.

Таблиця 2.2 – Опис варіанту використання "Створення маршруту"

Найменування	Створення маршруту
Номер	2
Опис	Користувачу дозволяється створити маршрут
Актор	Авторизований Користувач
Головний сценарій	Користувач натискає кнопку створення маршруту та виконує всі необхідні дії, задаючи параметри маршруту, і натискає кнопку "Створити"
Альтернативний сценарій	Маршрут не створився, бо немає доступу до карт, немає можливості додати запис в БД

Таблиця 2.3 – Опис варіанту використання "Перегляд маршруту"

Найменування	Перегляд маршруту
Номер	3
Опис	Дозволяє користувачу переглядати маршрут
Актор	Авторизований Користувач
Головний сценарій	Користувач зі списку обирає маршрут та переглядає його детальний опис
Альтернативний сценарій	Маршрут не висвітився, бо немає доступу до карт

Опишемо стандартний алгоритм дій користувача зі створення маршруту (рис.2.2).

1) Користувач в системі переходить в режим створення маршруту.
2) Далі він вводить дані свого маршруту, обираючи його ключові точки.
3) Модуль роботи з картами здійснює прокладання пішого маршруту, використовуючи існуючі дороги.
4) Модуль роботи з картами розраховує довжину та тривалість маршруту
5) Модуль реєструє маршрут, якщо все нормально. В противному випадку користувачу видається повідомлення про відхилення

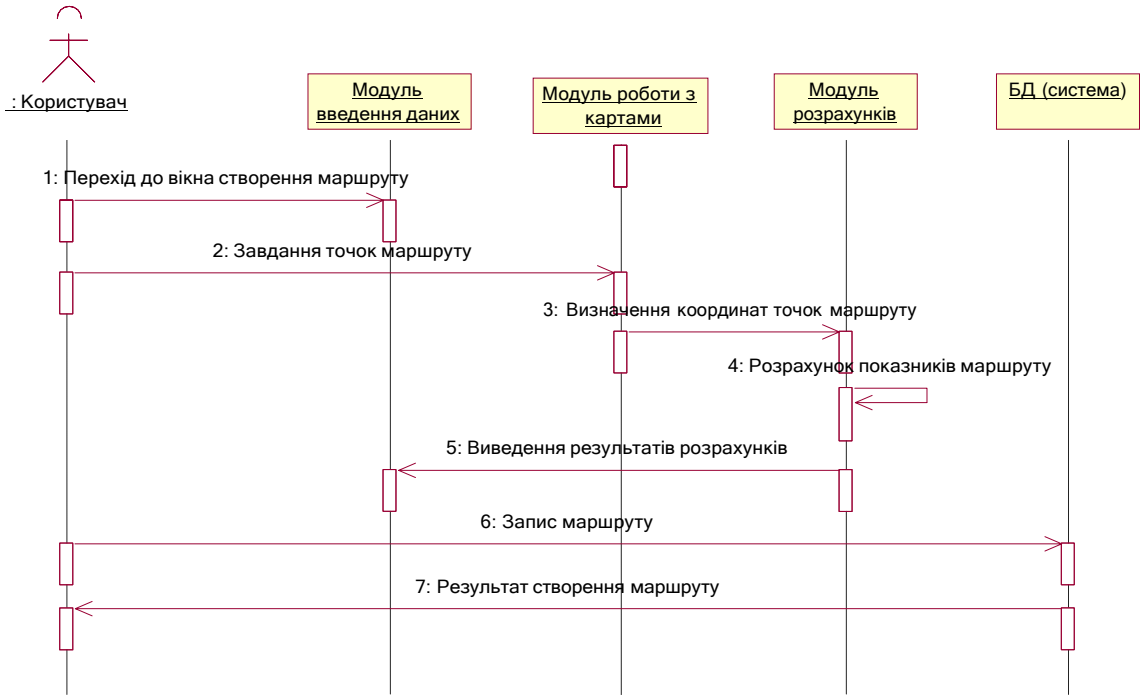

Рисунок 2.2 – Діаграма послідовності процесу зі створення маршруту

Джерело: розробка автора

Діаграма класів моделює відношення між класами, атрибутами та операціями. Класи є абстракцією об’єктів з загальними характеристиками. Між класами встановлюються відношення (зв’язки). Розроблена діаграма класів для веб-сервісу прокладання туристичного маршруту наведена на рис. 2.3.

Рисунок 2.3 – Діаграма класів веб-сервісу

Джерело: розробка автора

Згідно визначених бізнес-вимог до роботи веб-сервісу розробка програмного рішення повинна включати в себе:

– розробка дизайну веб-сайту (клієнтської частини);

– розробка сервісної частини веб-сервісу;

– розробка бази даних.

Весь функціонал даного сервісу можна визначити таким чином

– функціонал для роботи з картами;

– демонстраційний функціонал – для представлення створених за допомогою розробки маршрутів;

– пошуковий функціонал, який передбачає сортування маршрутів за певними критеріями та виведення маршрутів, створених окремими користувачами сервісу.

Веб-сервіс не потребує встановлення особливих вимог до інформаційної безпеки.

2.2 Проектування архітектури Web-сервісу

На основі визначених вимог до веб-сервісу та розробленої функціональної моделі перейдемо до проектування архітектури.

Архітектура Web-додатків – це сукупність рішень, як організувати програму. У неї входять: структурні елементи та інтерфейси, зв'язку між обраними елементами, загальний стиль програми [3; 8].

Існує декілька підходів до організації архітектури веб-сервісів:

– монолітний;

– композитний.

В монолітній архітектурі передбачається, що вся бізнес-логіка зберігається на сервері, а необхідні дані в базі даних. Як правило, подібні програми не відрізняються складністю в розробці і великою вартістю на ранніх етапах. Нова функціональність додається легко і швидко. Однак підтримувати монолітну систему в довгостроковій перспективі дуже складно і дорого. Крім того, в рамках веб-сервісу однією з проблем "моноліту" є масштабованість. Всі складові системи розташовані в одній точці, а тому потужність самої точки повинна бути відповідною для підтримки сервера і бази даних в робочому стані.

Композитний підхід передбачає використання набору програмних систем з різними характеристиками, що виконують певну задачу і розробляються на основі спільного набору базових засобів. Такий підхід дозволяє виділити базові елементи та розбити бізнес-процеси на окремі операції, які можна перенести в набор сервісів, що не буде залежати від інших компонентів системи.

Сервіси є окремі цілком самодостатніми модулями, що можуть володіти власною апаратною базою та власною базою даних. Можливість розробки програмних рішень на основі композитного підходу надає можливість розроблювати сервіси на різних мовах програмування. Для взаємодії цих двох частин необхідно тільки правильно використовувати програмний інтерфейс сервісу в основній частині програми. Розробка сервісів відбувається окремо один

від одного, а тому зміни в коді одного сервісу будуть впливати тільки на нього, якщо тільки не змінилася видача даних в програмному інтерфейсі сервісу, що відбувається вкрай рідко, оскільки з форматом виведення визначаються ще на перших етапах розробки сервісу [33].

Спроектований бізнес-процес побудови маршруту складається з таких задач: аутентифікація користувача, визначення точок маршруту, прокладання маршруту, розрахунок параметрів маршруту, створення повідомлення про успішність/неуспішність прокладання маршруту. Дві з цих задач передбачають роботу з картами, цифрове представлення яких реалізоване сторонніми сервісами. Для реалізації механізму аутентифікації також передбачається використання спеціального сервісу, що надає доступ до захищених ресурсів користувача. Отже, для розроблюваного програмного рішення необхідним є застосування сервісно-орієнтованого типу архітектури. Узагальнена архітектура системи наведена на рис.2.4.

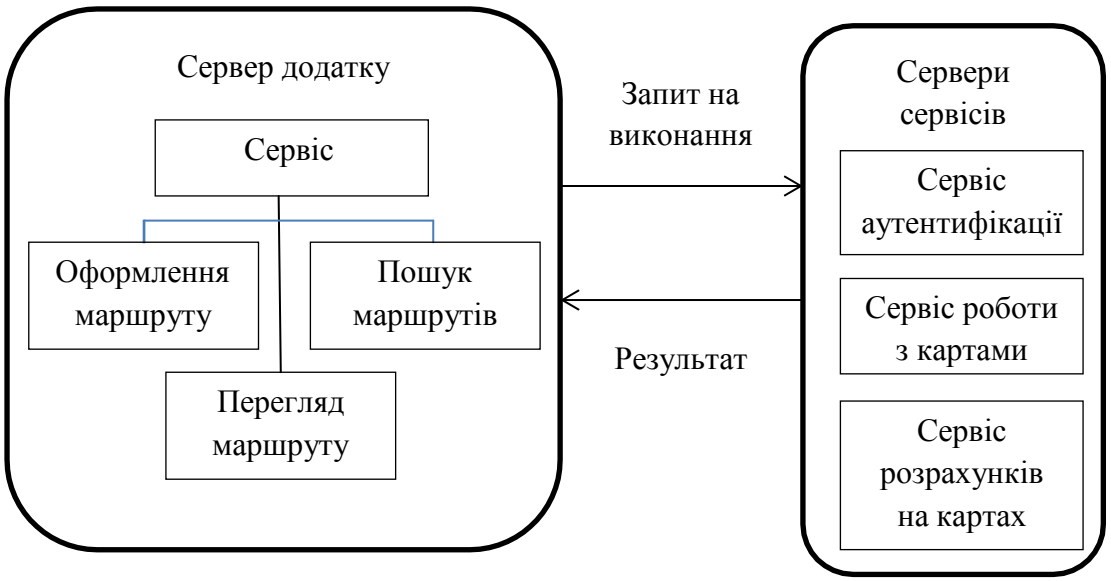

Рисунок 2.4 – Узагальнена схема архітектури розроблювальної системи

Джерело: розробка автора

Структура розроблюваного додатку складається з трьох шарів – шар доступу до даних та даних, шар бізнес-логіки, шар представлення. Оскільки

передбачається використовувати достатньо складну модель для подання географічних даних, то об’єднання моделі предметної області, шару доступу до даних та джерела даних на одному системному рівні дозволить реалізувати єдиний простір рішень, що значно спростить організацію даних всередині веб-сервісу. При прийнятті рішення щодо такої реалізації також було враховано, що програмній розробці не потрібно оперувати з великими масивами даних. Архітектура системи при такому розподілі подана на рис.2.5.

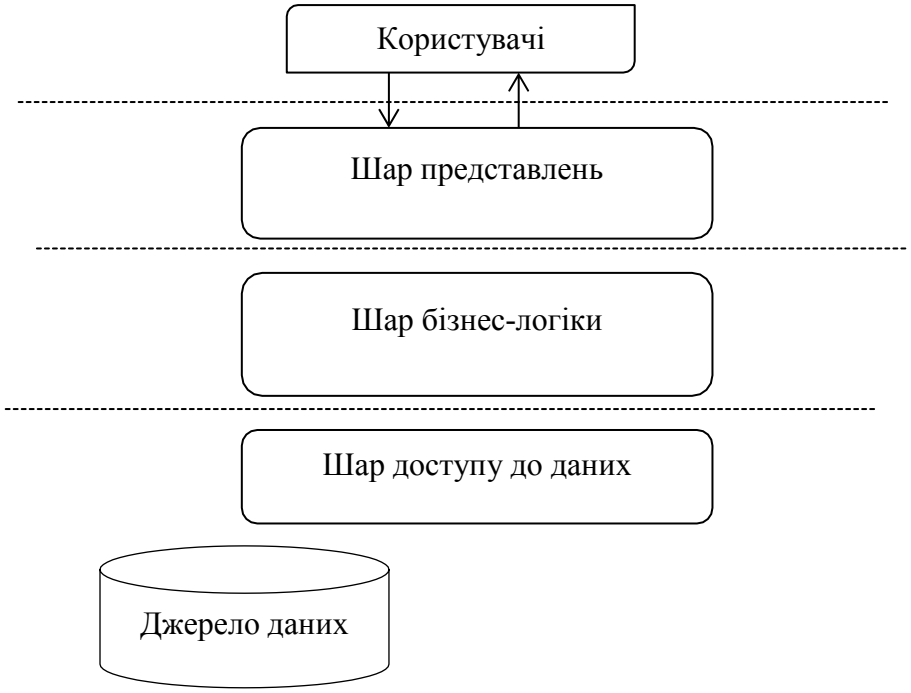

Рисунок 2.5 – Архітектура веб-сервісу

Джерело: розробка автора за матеріалами джерела [33]

Дана архітектура є достатньо стандартним рішенням, яке часто використовують при проектуванні систем роботи з географічними даними.

В шарі доступу до даних відбувається підключення до бази даних, а також зберігаються міграції, які використовуються для оновлення структури бази даних. Використання даного патерну забезпечить можливість зміни джерела даних без змін в інших шарах розробки. Репозиторій повинен створюватися для кожної сутності.

В шарі бізнес-логіки зберігаються обробники команд. Вони використовують дані з команди та містять в собі логіку виконання будь-якої дії.

Користувач взаємодіє з шаром представлення, реалізацією якого є веб-сторінка розроблюваного сервісу.

Оскільки розроблюваний проект передбачає використання сторонніх сервісів, то пропонується застосувати технології REST API. Це забезпечить розроблювальній системі доступ та управління веб-ресурсами на основі єдиного наперед визначеного набору операцій. Як вже, відмічалось у першому розділі даної роботи, основними перевагами даної технології є простота єдиного інтерфейсу, масштабованість, можливість модифікування компонентів. Дана технологія використовує клієнт-сервісну архітектуру, що є ще однією перевагою при розробці сервісу прокладання маршрутів.

Загальним форматом надсилання та запиту даних через REST API є JSON (JavaScript Object Notation). Об'єкт JSON виглядає як об'єкт JavaScript [56].

Результуюча архітектурна схема розроблюваного веб-сервісу наведена на рис.2.6.

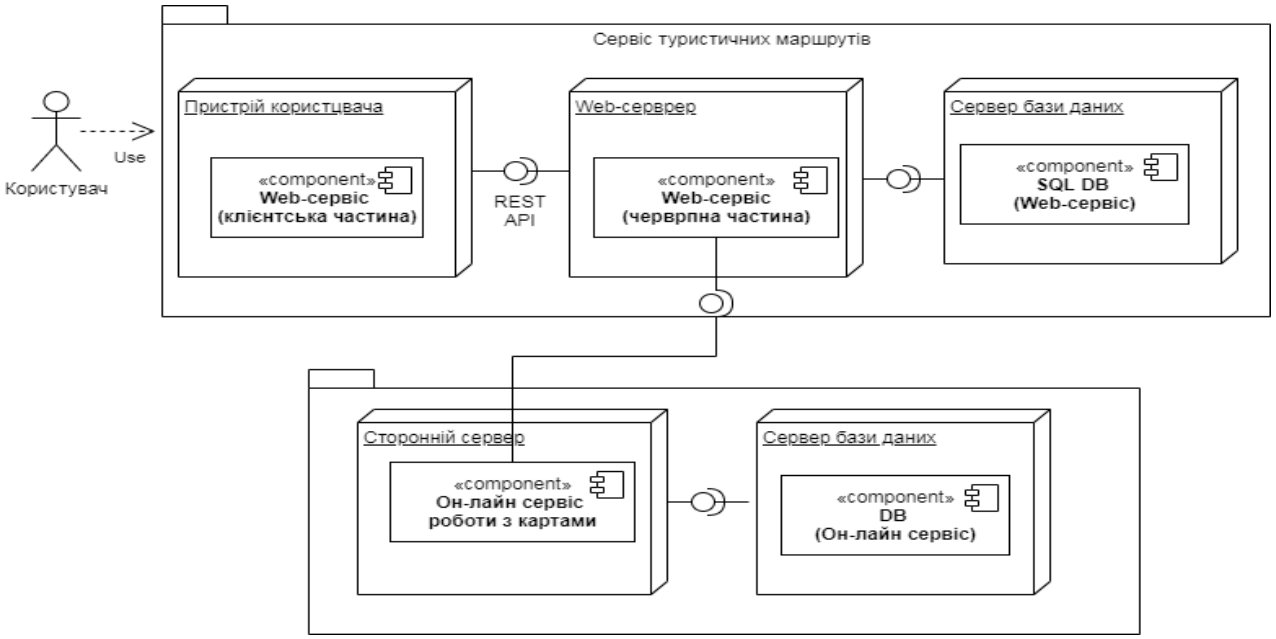

Рисунок 2.6 – Загальна схема клієнт-серверної архітектури програмного рішення

Джерело: розробка автора

Отже, розроблюваний веб-сервіс буде представляти собою трирівневий додаток на основі клієнт-серверної архітектури.

2.3 Проектування БД

База даних – це сукупність взаємопов’язаних даних, що зберігаються разом. Дані мають певну структуру, яка дозволяє використовувати їх оптимальним способом для одного або декількох додатків [9; 49].

Бази даних поділяються на дві великі групи – реляційні та не реляційні, основною відмінністю між якими є спосіб організації збереження інформації, підхід до проектування та типи даних, що зберігаються.

Реляційні бази даних зберігають строго структуровані дані та ґрунтуються на реляційній моделі подання даних і підтримують мову структурованих запитів (SQL).

Реляційна база даних є множиною взаємозв’язаних таблиць, кожна з яких містить інформацію про об’єкти певного виду. Основною характеристиками даної моделі даних є несуперечливість та надійність.

В той же час існують певні типи даних та способи їх зберігання, які або не можуть бути реалізовані в рамках реляційної моделі, або реалізація їх представлення в даній моделі суттєво сповільнює роботу з ними. Тому для уникнення таких недоліків були розроблені також нові підходи до побудови баз даних. Серед них розрізняють:

– сховища у вигляді ключ-значення, де кожному значенню відповідає унікальний ключ;

– документно-орієнтовані бази даних для збереження інформації у вигляді ієрархічної структури даних;

– графові бази даних, що використовують мережеву модель подання даних. Найчастіше їх використовують, коли дані мають природну графову структуру.

– дані big-table, що упорядковані у вигляді великих розріджених таблиць для зберігання великих масивів даних [29].

Розроблюваний додаток оперує з даними, які мають досить чітко визначену структуру, і не висуває підсилених вимог до швидкості обробки, тому пропонується використовувати реляційну модель даних.

Виділимо основні сутності, які представлені в системі, опираючись на проведене проектування.

Сутність Користувач (User) описує користувача веб-сервісу і повинна містити в собі інформацію для авторизації користувача, а саме, ім’я, логін, пароль для входу в систему, дати створення та редагування записів.

Сутність Маршрут (Route) описує створені за допомогою сервісу маршрути та повинна містити назву маршруту, рівень складності, довжину, тривалість, опис маршруту, його кроки, дату створення та редагування та ім’я користувача, що створив даний маршрут.

Модель даних, що використовується в системі подана на схемі 2.7.

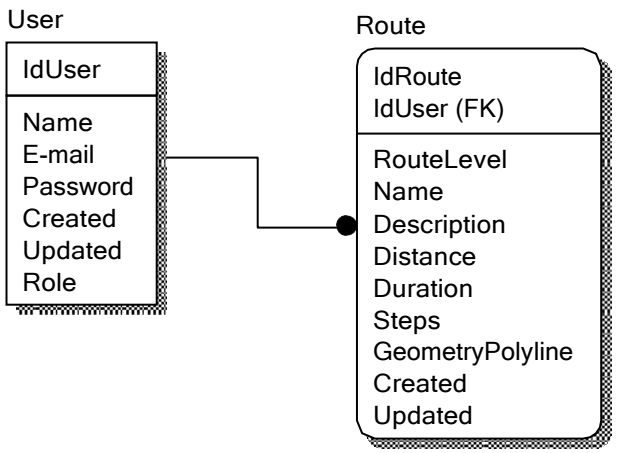

Рисунок 2.7 – Концептуальна схема бази даних розроблювального сервісу

Джерело: розробка автора

Враховуючи умови, що висуваються до реляційних БД, проведемо її нормалізацію. Для цього виділимо окрему реляційну таблицю, що буде містити інформацію щодо різних варіантів складності маршрутів. Таким чином, ми уникнемо зайвої надлишковості.

Остаточна схема спроектованої БД подана на рис 2.8.

Рисунок 2.8 – Остаточна схема бази даних веб-сервісу з розробки туристичних маршрутів

Джерело: розробка автора

Для кодування поліній буде використовуватися алгоритм стиснення з втратами, який дозволяє зберігати ряд координат як єдиний рядок. Координати точок кодуються за допомогою підписаних значень. А сам процес кодування перетворює двійкове значення в ряд символьних кодів для символів ASCII, використовуючи звичну схему кодування base64. Таким чином, використання текстового типу для збереження цих значень у базі даних розроблюваного додатку є виправданим [39].

Висновки до розділу 2.

В результаті проведеного дослідження була розроблена об’єктно-орієнтована модель, що відображає основні аспекти розроблюваної системи, а саме, функціональний аспект та взаємодію об’єктів та повідомлень.

На основі даної моделі була обрана клієнт-серверна композитна архітектура для веб-сервісу. Для реалізації взаємодії між сервером додатку та браузером був запропонований REST підхід з обміном повідомленнями в форматі JSON.

На основі прийнятих проектних рішень була розроблена база даних веб-сервісу.

РОЗДІЛ 3 РОЗРОБКА WEB-СЕРВІСУ ДЛЯ ПОБУДОВИ ТУРИСТИЧНИХ МАРШРУТІВ

3.1 Розробка користувацького інтерфейсу

Проектування інтерфейсу є необхідним етапом процесу розробки будь-якого додатку [5; 10].

Графічний інтерфейс користувача зазвичай поділяють на три види:

– простий: типові екранні форми і стандартні елементи інтерфейсу, що забезпечуються самою підсистемою GUI (graphical user interface);

– істинно-графічний, двовимірний: нестандартні елементи інтерфейсу і оригінальні метафори, реалізовані власними засобами програми або сторонньою бібліотекою;

– тривимірний.

Основними показниками інтерфейсу користувача є ергомічність та юзабіліті [10].

Юзабіліті (англ. Usability – "можливість використання", "корисність") – ступінь ефективності, продуктивності і задоволеності, з якими продукт може бути використаний певними користувачами в певному контексті використання для досягнення певних цілей (пункт 3.1 стандарту ISO 9241-11).

Ергономічність є кількісною характеристикою, описує кількість витрачених фізичних сил для роботи з програмою, а юзабіліті – якісною, яка описує суму розумових зусиль, які вимагаються від користувача для виконання завдань, а також загальний ступінь зручності користування.

Враховуючи сучасні тенденції в області графічного дизайну різних веб-додатків пропонується для Web-сервісу, що розробляється, створювати достатньо мінімалістичний дизайн.

До проектування інтерфейсу можна підходити різними способами. Один з них полягає у створенні макету у вигляді вайрфрейма (від англійського wireframe – "каркасний"). Він представляє собою множину прямокутних блоків, з’єднаних лініями та стрілками. За допомогою цих блоків та стрілок визначається структура програмної розробки та порядок взаємодії користувача з нею [5].

Вайрфрейми фокусують увагу розробників на юзабіліті, тобто взаємодії з програмним продуктом, і в той же час не відволікатися на деталі, що не є суттєвими (наприклад, кольорове оформлення кнопок, тип шрифтів тощо).

Вайрфрейми можна створювати достатньо швидко, і не витрачати на даний процес велику кількість коштів, тобто знизити ризики проекту.

Таким чином, вайрфрейм є інструментом детального документування взаємодії, без додаткових затрат на візуальний дизайн..

При розробці вайрфрейму потрібно показати, до чого і куди призводить взаємодія майбутнього користувача з елементами інтерфейсу. Зв'язавши елементи лініями з іншими екранами, на які потрапить користувач, отримують призначені для користувача сценарії використання додатка, або user flow.

User flow – карта навігації, за якою відслідковується поведінка користувача Web-додатку, наскільки легко він досягає мети і скільки витрачає зусиль. User flow виглядає як логічно пов'язані один з одним прямокутники, акцент в яких зроблений на дії користувача.

Розроблюваний веб-сервіс буде складатися з чотирьох основних екранів: екрану авторизації, головного екрану, екрану створення маршруту та екрану перегляду маршрутів певного користувача.

Процес розробки такого макету – це співпраця технічних та візуальних параметрів майбутнього інтернет-магазину. Це вивчає такі параметри, як розмір елементів веб-сайту, їх розташування з точками змінення зручності пошуку інформації на сайті.

Основними компонентами розроблюваного макету є:

1. Логотип сайту – це зображення з повною або скороченою назвою сайту, або малюнок з різними елементами, які в подальшому використовуються для розробки фірмового стилю та необхідні для ідентифікації сайту.
2. Область для інформації про користувача.
3. Область, в якій розміщений список доступних маршрутів.
4. Область з відображенням карти для створення маршруту та інформацією про маршрут.
5. Область з іконкою активного користувача

Інтерфейс системи повинен передбачати можливість створення маршруту на

Макети вікон Web-додатку мають вигляд, зображений на рис. 3.1-3.4

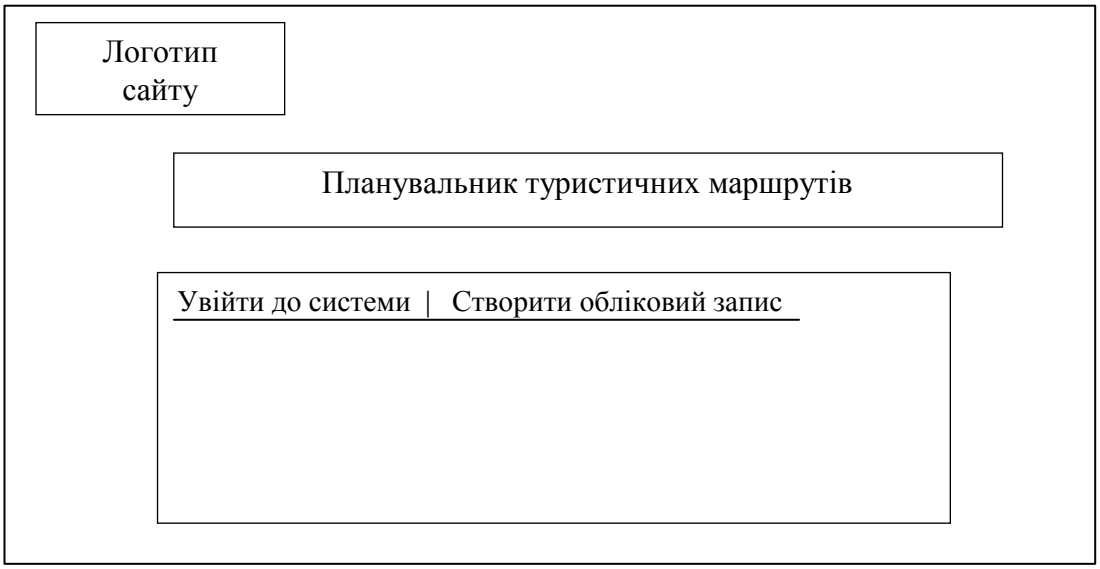

Рисунок 3.1 – Макет вікна авторизації у системі

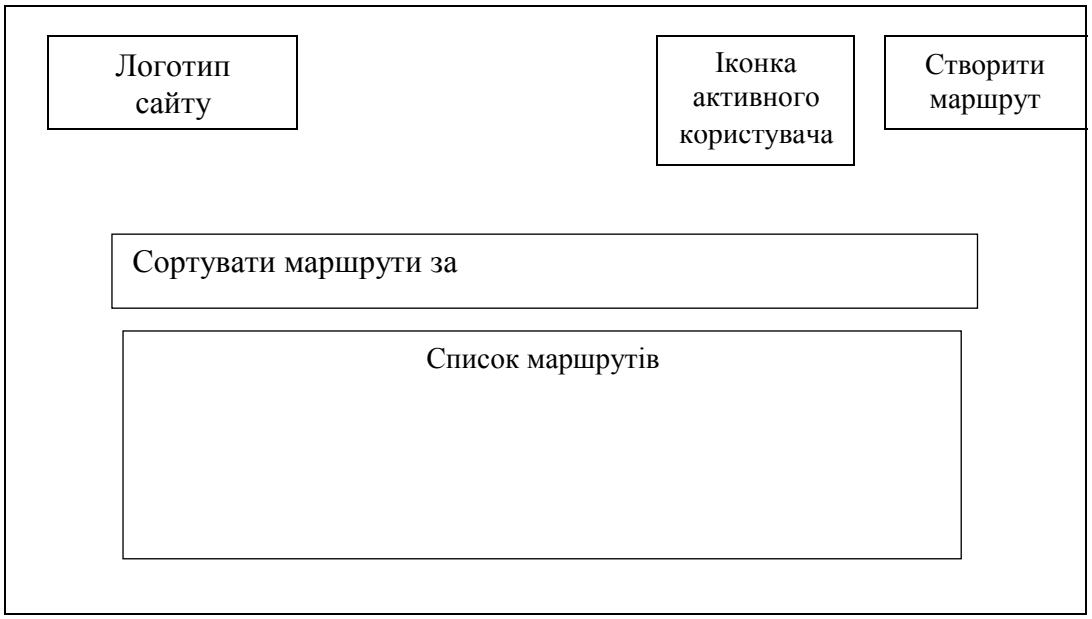

Рисунок 3.2 – Макет головної сторінки системи

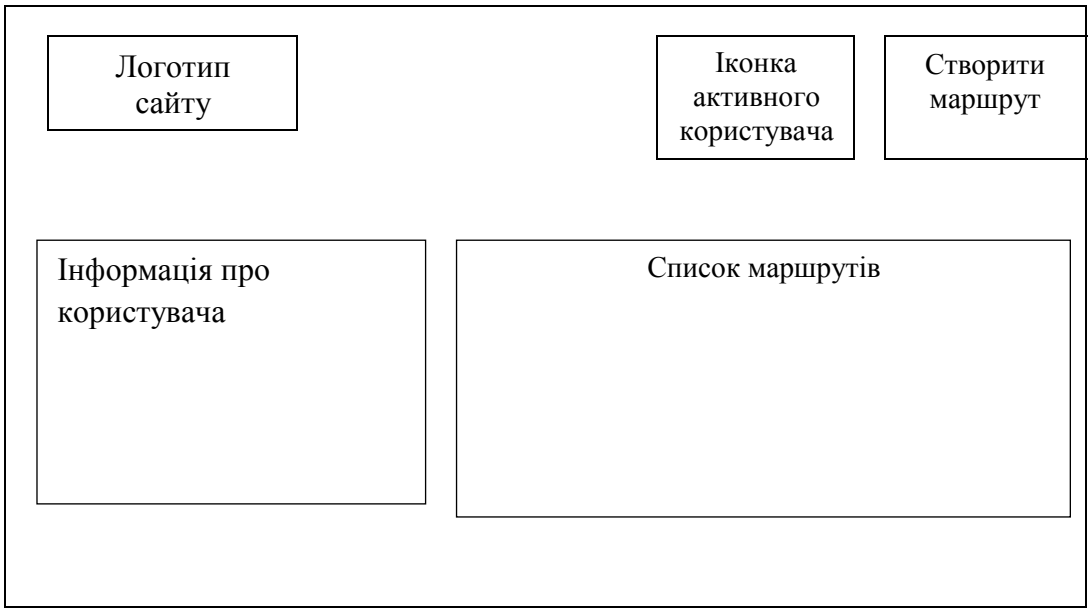

Рисунок 3.3 – Макет сторінки користувача

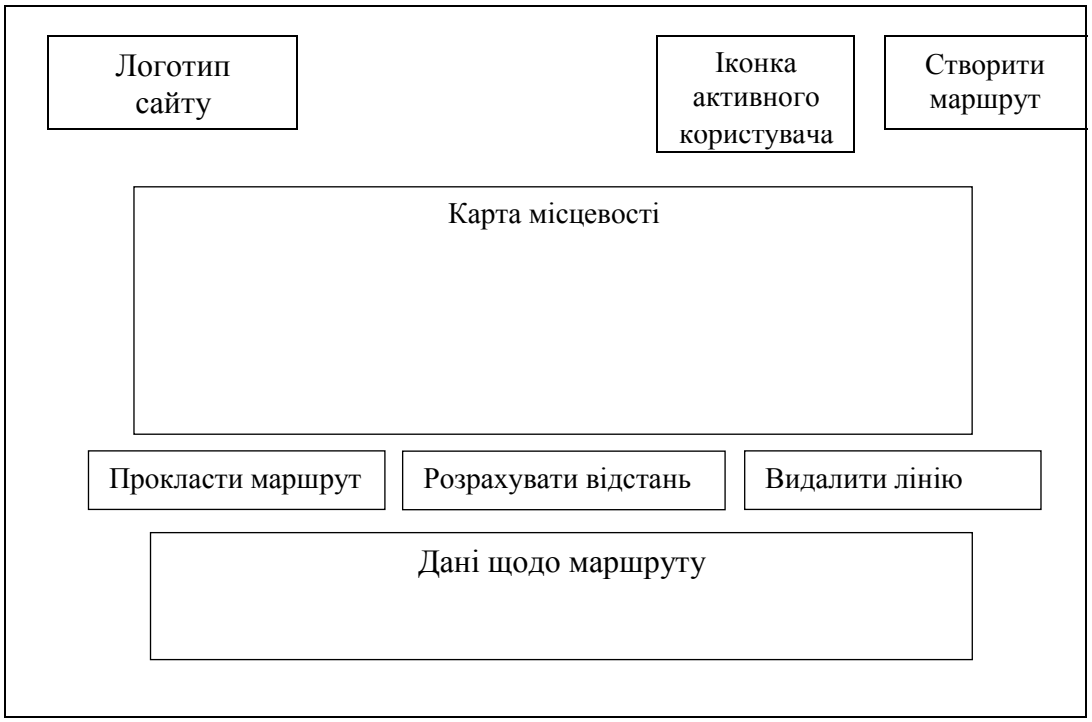

Рисунок 3.4 – Макет сторінки створення маршруту

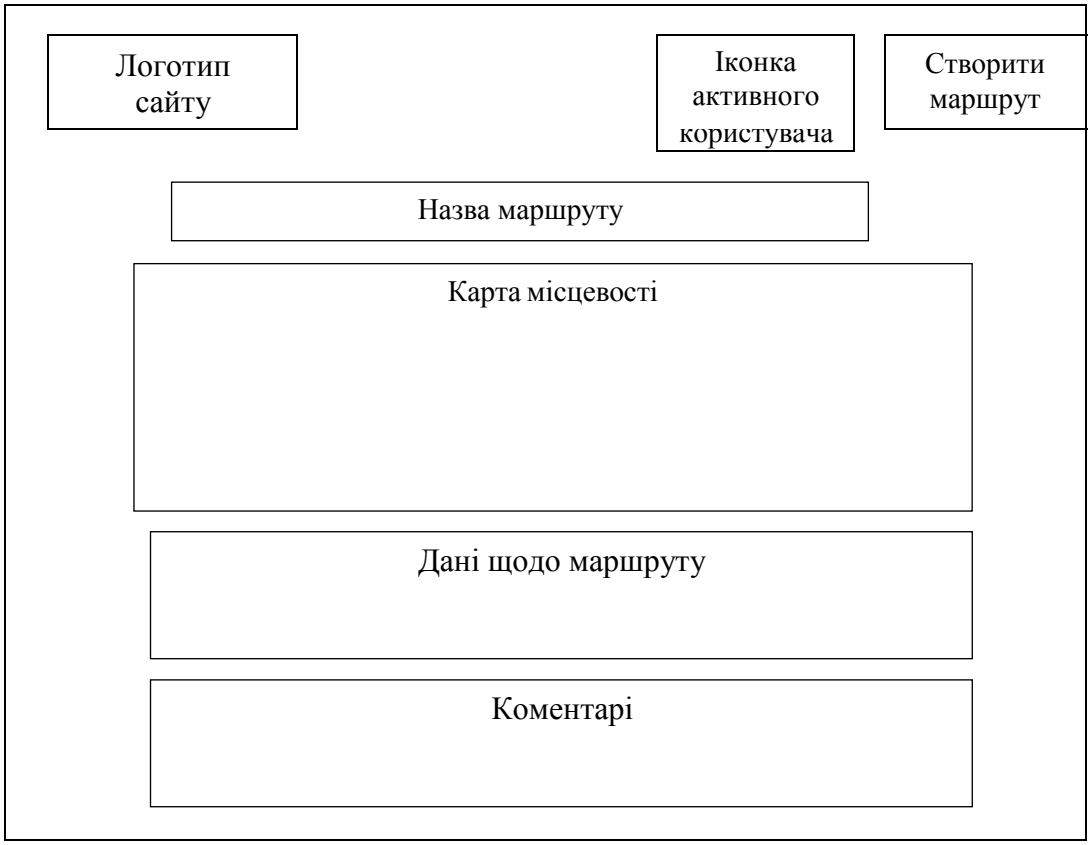

Рисунок 3.5 – Макет сторінки перегляду маршруту

Карта навігації наведена на рис. 3.6.

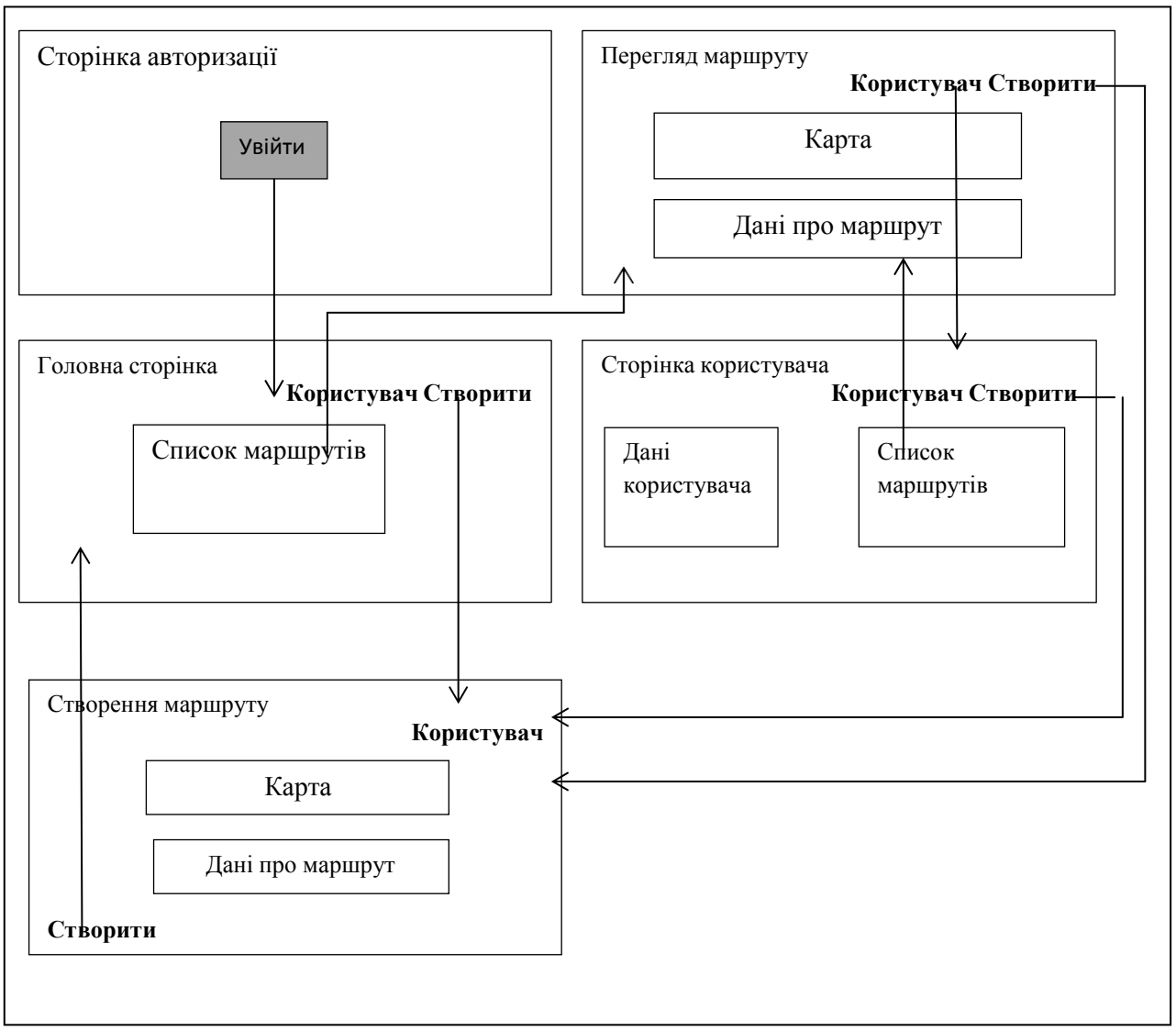

Рис. 3.10 – Карта навігації по сайту Web-сервісу розробки туристичних маршрутів

Спроектований інтерфейс є доволі зручним для користувача, дає змогу з будь-якої сторінки створити маршрут. Користувачу для роботи з спроектованим інтерфейсом не потрібно жодних додаткових навичок, оскільки інтерфейс є інтуїтивно зрозумілим. Також в інтерфейсі буде передбачена можливість навігації по карті при створенні маршруту. Така можливість забезпечується використанням відповідного стороннього он-лайн сервісу. Отже, інтерфейс відповідає вимогам до веб-сервісу.

3.2 Програмування сервісу

Алгоритм роботи розроблюваного сервісу наведений на рис. 3.6.

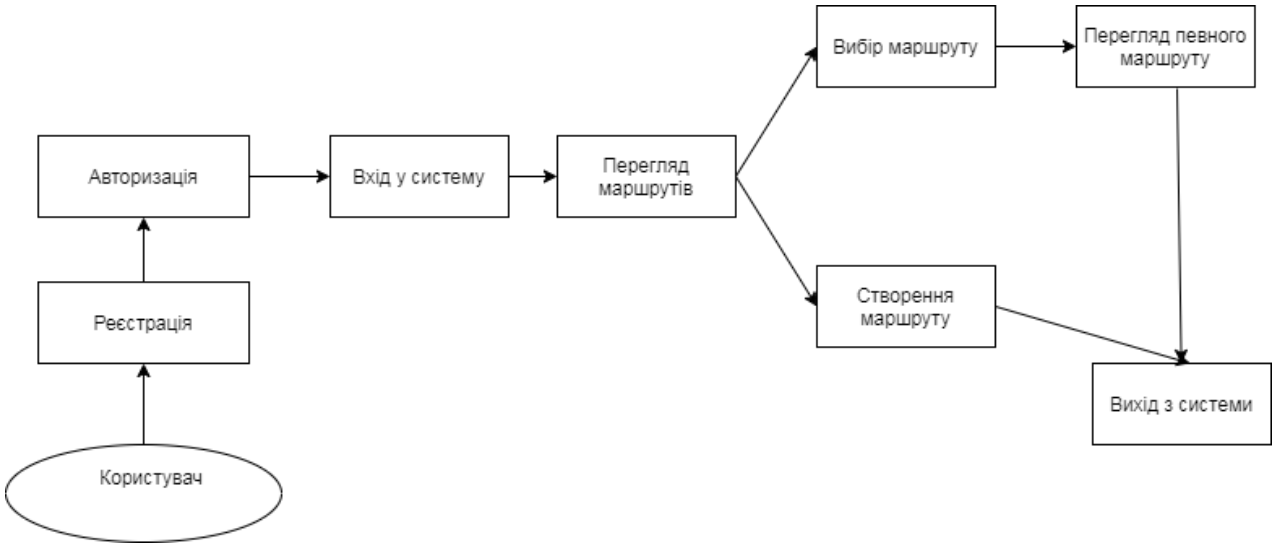

Рисунок 3.6 – Алгоритм роботи веб-сервісу

Згідно розробленої архітектури, розробка веб-сервісу повинна проводитися в трьох напрямках.

Для розробки клієнтської частини використовувався стек технологій: HTML5, CSS3, JavaScript. Сама розробка веб-сторінок здійснювалась за фреймфорка Bootstrap, який є набором інструментів для створення сайтів і Web-додатків, що розповсюджується безкоштовно. Він включає в себе HTML і CSS шаблони оформлення для типографіки, Web-форм, кнопок, міток, блоків навігації та інших компонентів Web-інтерфейсу, включаючи JavaScript-розширення [43].

Bootstrap використовує найсучасніші напрацювання в області CSS і HTML, тому часто необхідно бути уважним при підтримці старих браузерів які не підтримують деякі функції [35].

Основною перевагою Bootstrap є економія часу при розробці, оскільки він дає змогу використовувати шаблони і класи дизайну. Масштабування в

динамічних макетах Bootstrap відбувається для різних пристроїв та роздільних здатностей екрану без будь-яких змін в розмітці. Це скорочує час оптимізації проекту.

Для всіх компонентів використовується єдиний стиль і шаблони за допомогою центральної бібліотеки. Дизайн і макети Web-сторінок узгоджуються один з одним.

Крім того, Bootstrap сумісний з усіма основними браузерами і коректно відображається на останніх версіях Web-браузерів: Mozilla Firefox, Google Chrome, Safari, Internet Explorer, Microsoft Edge і Opera.

Особливістю Twitter Bootstrap є відкритий вихідних код і безкоштовне розповсюдження.

Підключення фреймворка здійснюється таким чином (рис.3.7)

```
<!DOCTYPE html>
<html lang="en" class="h-100">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Всі маршрути | Планувальник туристичних маршрутів</title>
    <link rel="stylesheet" href="/assets/vendor/bootstrap/css/bootstrap.min.css">
</head>
<body class="position-relative" style="min-height: 100%;">
<header class="bg-dark">
    <div class="container">
        <nav class="navbar navbar-expand-lg navbar-dark ">
            <a class="navbar-brand" href="/routes">
                <img src="/assets/img/tourism.svg" width="70" height="70" alt="Лого">
                Туристичні маршрути
            </a>
```

Рисунок 3.7 – Підключення фреймворка Bootstrap

Для розробки використовувався шаблони та елементи, що запропоновані у фреймворку: фіксована панель навігації (рис.3.8), панель Offcanvas, панелі видачі інформації тощо (Додаток А.1).

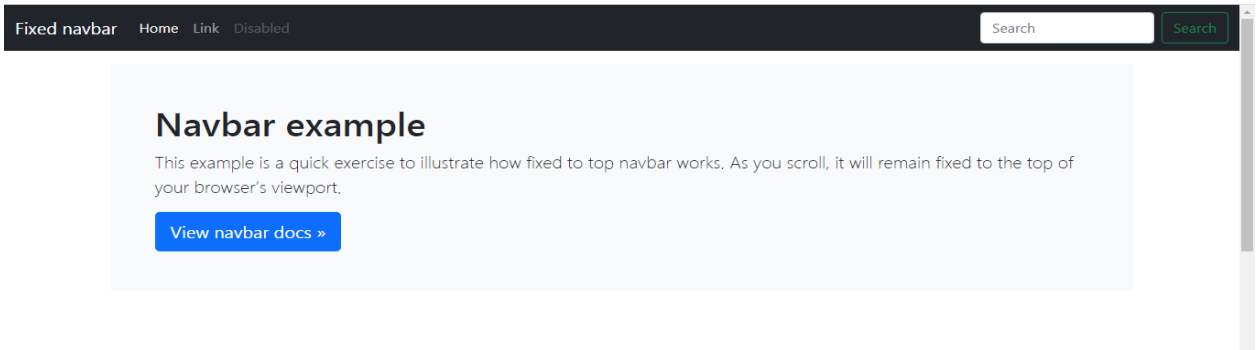

Рисунок 3.8 – Фіксована панель навігації

Для реалізації серверної частини був використаний фреймворк Symfony, написаним на PHP. Фреймворк реалізує паттерн Model-View-Controller і в його архітектурі дуже активно використовуються інші патерни об'єктів-орієнтованого програмування [55].

Symfony має підтримку безлічі баз даних (MySQL, MariaDB, PostgreSQL, SQLite, підходять і інші PDO-сумісні СУБД). Інформація про реляційну базу даних в проекті повинна бути пов'язана з об'єктною моделлю за допомогою ORM інструменту. Базова версія Symfony поставляється з двома ORM: Propel і Doctrine.

Структура проекту наведена на рис.3.9.

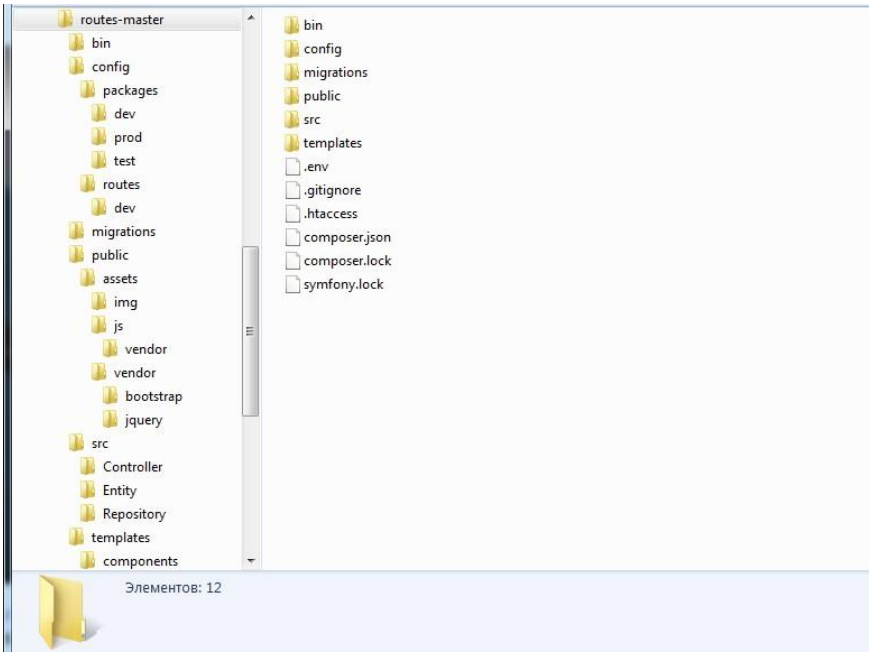

Рисунок 3.9 – Структура розроблюваного веб-сервісу

Архітектура проекту складається з трьох компонентів:

– контролер, що працює з запитами та відповідями;

– сервіс, що отримують аргументи та повертають примітивні типи;

– джерела даних, що містять тільки дані.

Всі запити, які поступають до додатка, спочатку потрапляють на фронт-контролер, який створює екземпляр AppKernel для обробки запиту (Додаток А.2).

Кожна програма Symfony складається з набору бандлів (пакетів), які додають інструменти (сервіси) в проект. Кожен бандл може бути налаштований через файл конфігурації, який за замовчунням розташований в каталозі / config (рис.3.10).

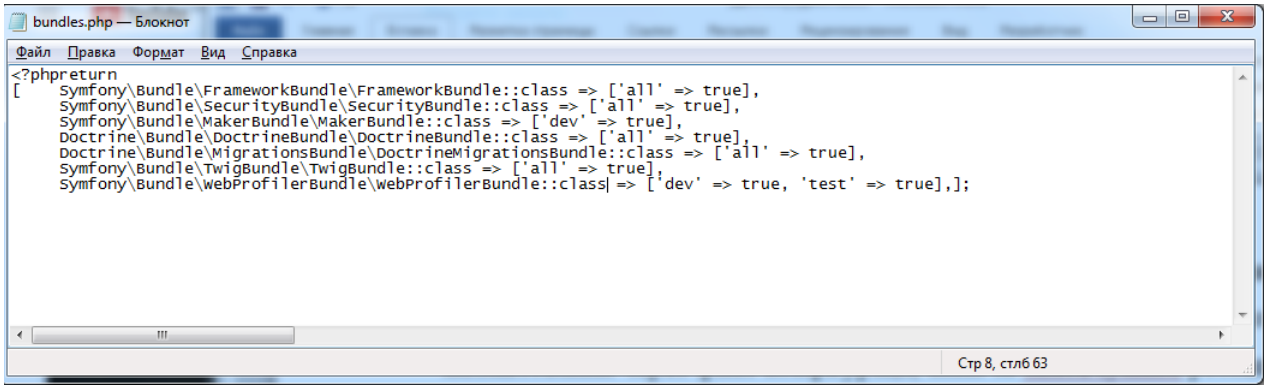

Рисунок 3.10 – Набір бандлів створюваного додатку

Контролер – це розроблена функція PHP, яка зчитує інформацію з об'єкта Request та створює та повертає об'єкт Response. Відповіддю може бути HTML-сторінка, JSON, XML, завантаження файлу, переспрямування, помилка 404 або щось інше. Контролер запускає будь-яку довільну логіку, необхідну програмі для відтворення вмісту сторінки.

Згідно визначеного проекту були розроблені три контролери, що керують сторінкою, користувачем та маршрутом.

Контролер для управління маршрутом може виконувати запити на додавання нового маршруту, редагування вже існуючого маршруту, видалення маршруту. Також в контролері передбачена функція додавання коментарів (Додаток А.3).

Аналогічним чином розробляються контролери для користувача та сторінки.

Абстракція шару доступу до БД в Symfony ґрунтується на Doctrine, об’єктно-орієнтованому проекторі. Для цього Symfony використовує паттерн Repository. Згідно розробленого проекту бази даних було створено 3 сутності (Entity), кожна з яких відповідає конкретній таблиці в базі даних [38].

Для зберігання точок маршруту використовується спеціальний алгоритм кодування поліліній. Цей алгоритм стиснення з втратами дозволяє зберігати ряд координат як єдиний рядок. Координати точок кодуються за допомогою підписаних значень [39].

Процес кодування перетворює двійкове значення в ряд символьних кодів для символів ASCII, використовуючи звичну схему кодування base64: для забезпечення належного відображення цих символів закодовані значення підсумовуються за модулем 63 (символ ASCII '?') перед перетворенням їх в ASCII. Алгоритм також перевіряє наявність додаткових символьних кодів для даної точки, перевіряючи найменш значущий біт кожної групи байтів; якщо для цього біта встановлено значення 1, то точка ще не повністю сформована, і далі ще будуть йти додаткові дані.

Крім того, для економії місця, точки включають лише зміщення від попередньої точки (крім першої точки). Усі точки кодуються як цілі числа зі знаком, оскільки широти та довготи – значення зі знаком. Формат кодування в полілінії повинен представляти дві координати, що представляють широту та довготу з розумною точністю. Враховуючи максимальну довготу +/- 180 градусів з точністю до 5 знаків після коми (від 180,00000 до -180,00000), це призводить до необхідності 32-бітового двійкового цілого числа зі знаком.

Наведемо етапи кодування такого підписаного значення.

Нехай задане початкове підписане значення:

-179,9832104

Його потрібно помножити його на 10^5, округлюючи результат:

-17998321

Перетворимо десяткове значення у двійкове, обчисливши від’ємне значення, використовуючи доповнення до двох, перетворивши двійкове значення та додавши одне до результату:

$$\begin{array}{rl} & 00000001\ 00010010\ 10100001\ 11110001 \\ + & 11111110\ 11101101\ 01011110\ 00001110 \\ \hline & 11111110\ 11101101\ 01011110\ 00001111 \end{array}$$

Далі виконується зсув двійкового значення вліво на один біт:

11111101 11011010 10111100 00011110

Якщо вихідне десяткове значення від’ємне, то необхідно інфертувати це кодування:

00000010 00100101 01000011 11100001

Потім двійкове значення розбивається на 5-бітові фрагменти (починаючи з правого боку):

00001 00010 01010 10000 11111 00001

Отримані 5-бітові фрагменти розміщують у зворотному порядку:

00001 11111 10000 01010 00010 00001

Виконують операцію OR над кожним значенням з 0x20 з переходом у наступний біт:

100001 111111 110000 101010 100010 000001

Після цього необхідно кожне значення перетворити в десяткове:

33 63 48 42 34 1

До кожного отриманого значення додати 63:

96 126 111 105 97 64

Застосування даного алгоритму до точок маршруту наведене на рис.3.11.

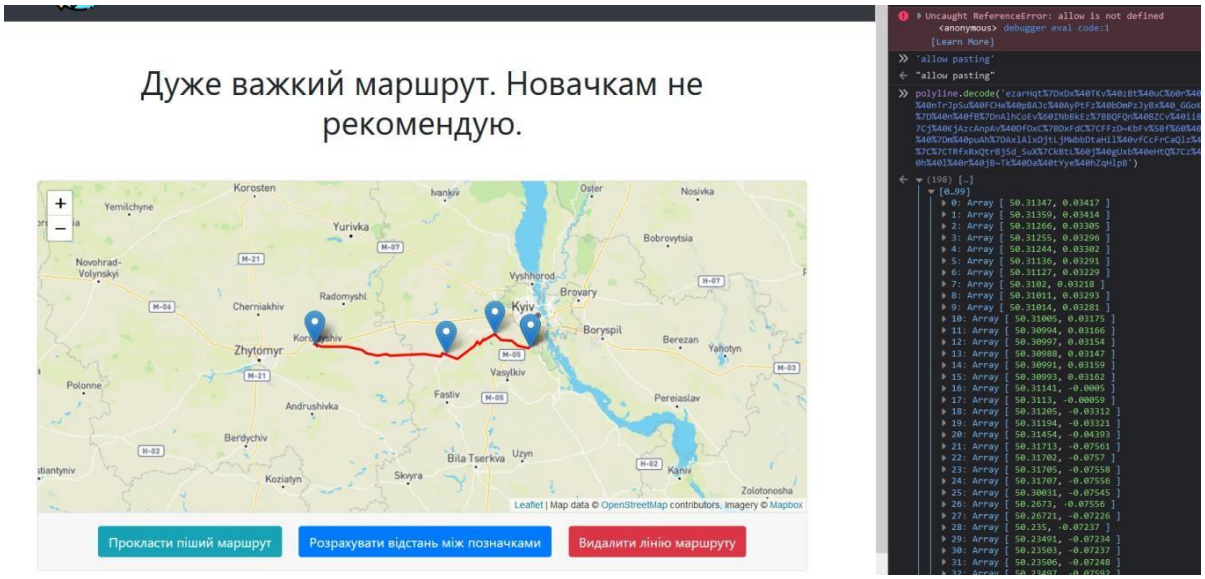

Рисунок 3.11 – Кодування маршруту в розроблюваному веб-сервісі

Для авторизації користувачів використовується сервіс, який базується на протоколі HTTP Basic [34]. Даний алгоритм працює таким чином:

Сервер, при зверненні неавторизованого клієнта до захищеного ресурсу, відсилає HTTP статус "401 Unauthorized" і додає заголовок "WWW-Authenticate" із зазначенням схеми і параметрів аутентифікації.

Браузер, при отриманні такої відповіді, автоматично показує діалог введення username і password. Користувач вводить деталі свого облікового запису.

У всіх наступних запитах до цього веб-сайту браузер автоматично додає HTTP заголовок "Authorization", в якому передаються дані користувача для аутентифікації сервером.

Сервер аутентифікує користувача за даними з цього заголовка. Рішення про надання доступу (авторизація) проводиться окремо на підставі ролі користувача та даних облікового запису.

Схема Basic є найбільш простою. При ній ім’я користувача та пароль передаються у заголовці Authorization в незашифрованому вигляді (кодування 64Base). При використанні протоколу HTTPS дана схема є відносно безпечною.

3.3. Тестування

Проведемо тестування розробленого веб-сервісу планування туристичних маршрутів. Розроблений сервіс знаходиться за адресою https://vr-diplom.site.

Розглянемо алгоритм дії користувача, що вперше потрапив на сайт. Для зареєстрованого користувача цей алгоритм повторюється за винятком перших кроків з реєстрації.

При першому вході на сайт користувач бачить відразу перед собою вікно для реєстрації (або авторизації) (рис.3.12).

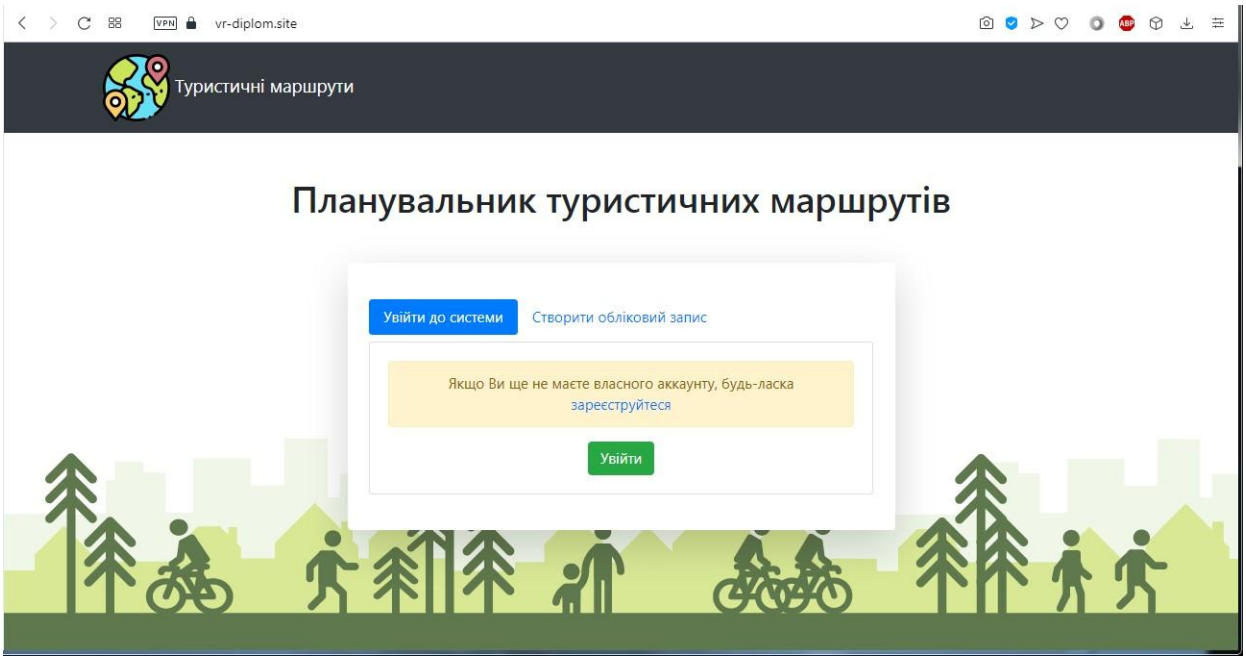

Рисунок 3.12 – Сторінка сервісу при першому вході

Для реєстрації користувачу потрібно натиснути вкладку Створити обліковий запис. У вікні реєстрації (рис.3.13) користувач повинен ввести ім’я (логін), адресу електронної пошти, пароль. Пароль необхідно підтвердити. Якщо паролі не співпадають, то система видає повідомлення про помилку.

Якщо реєстрація пройшла успішно, то система видає повідомлення (рис.3.14).

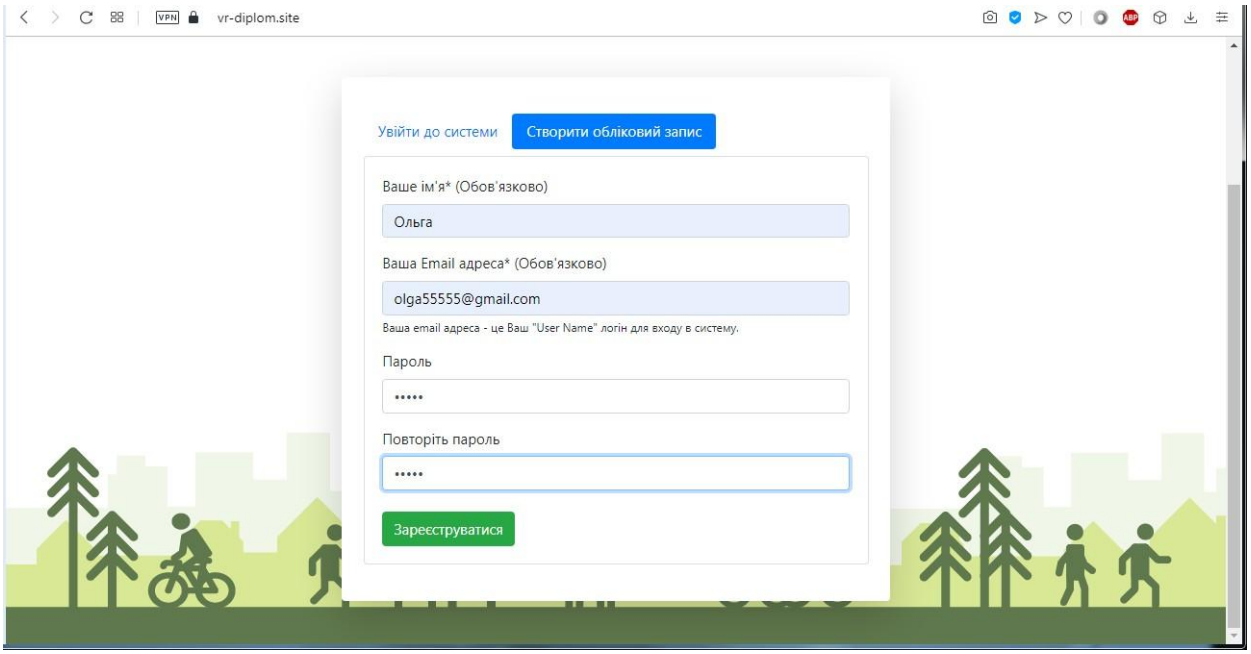

Рисунок 3.13 – Створення облікового запису

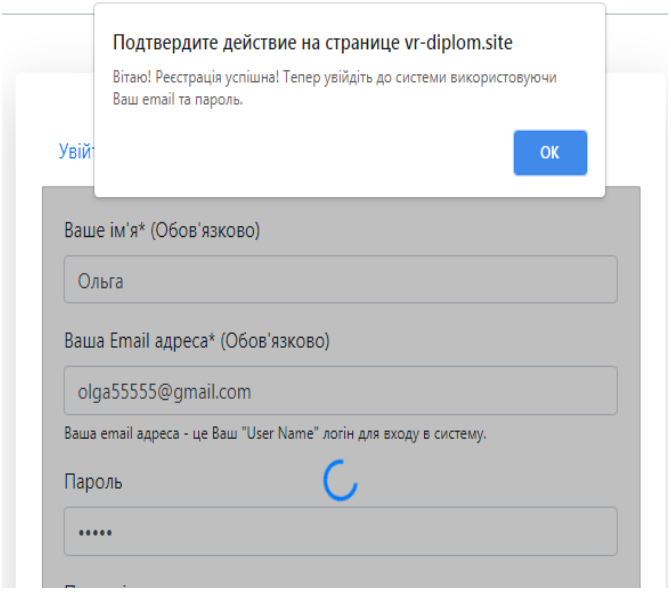

Рисунок 3.14 – Повідомлення про успішну реєстрацію

Після реєстрації користувач може перейти до авторизації, щоби увійти в систему. Для цього потрібно натиснути кнопку Увійти і у вікні, що з’явилось потрібно ввести адресу електронної пошти та пароль та натиснути кнопку Увійти (рис.3.15).

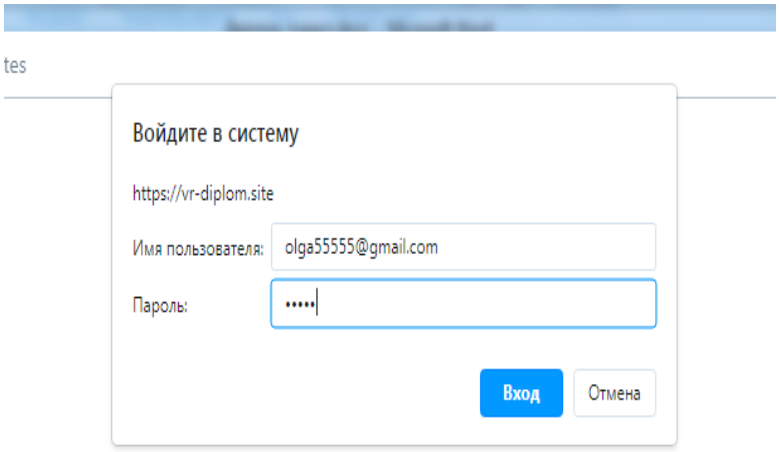

Рисунок 3.15 – Авторизація користувача

Якщо все введено правильно, то завантажується головна сторінка сервісу (3.16). У лівому верхньому куті розташований логотип сервісу, а у правому верхньому куті знаходиться ім’я користувача, що зайшов в систему та кнопка для створення маршруту.

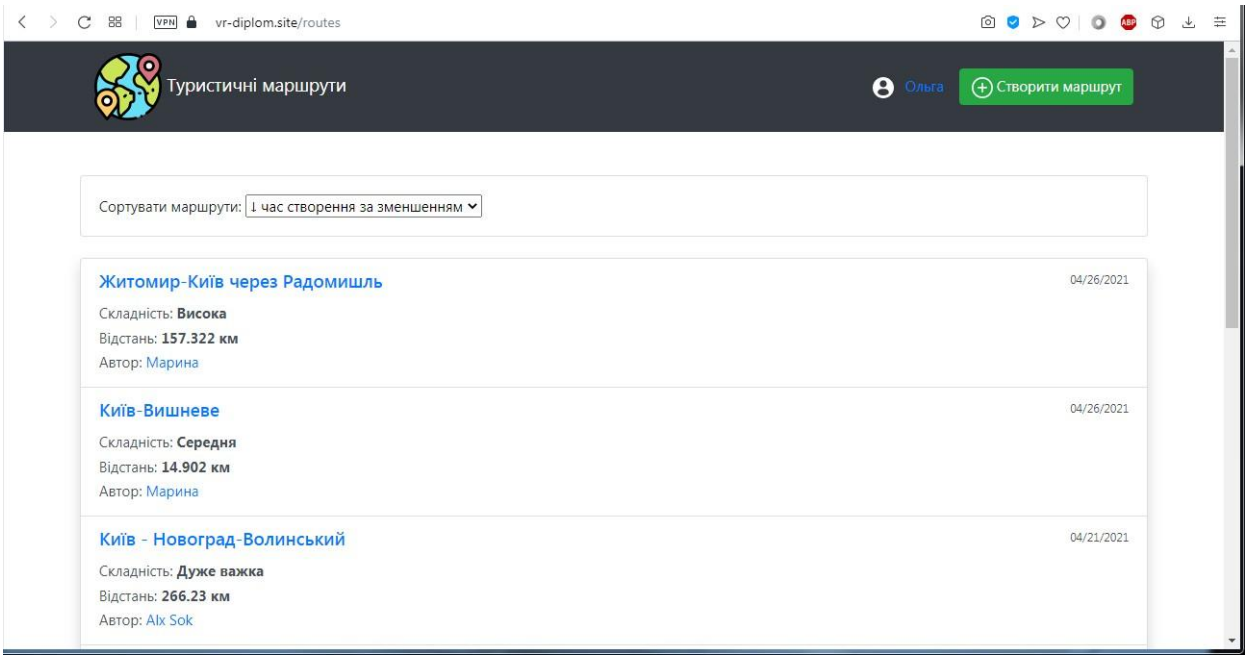

Рисунок 3.16 – Головне вікно сервісу планування туристичних маршрутів

В основній частині вікна знаходиться список маршрутів, що вже створені за допомогою даного сервісу. Для кожного маршруту виводиться назва, дата створення, складність.

За допомогою випадаючого списку з написом сортувати маршрути користувач може здійснити сортування маршрутів за такими параметрами:

– час створення за зменшенням/збільшенням;
– відстань за зменшенням/збільшенням;
– тривалість за зменшенням/збільшенням.

На рисунку 3.17 наведене сортування маршрутів за збільшенням відстані.

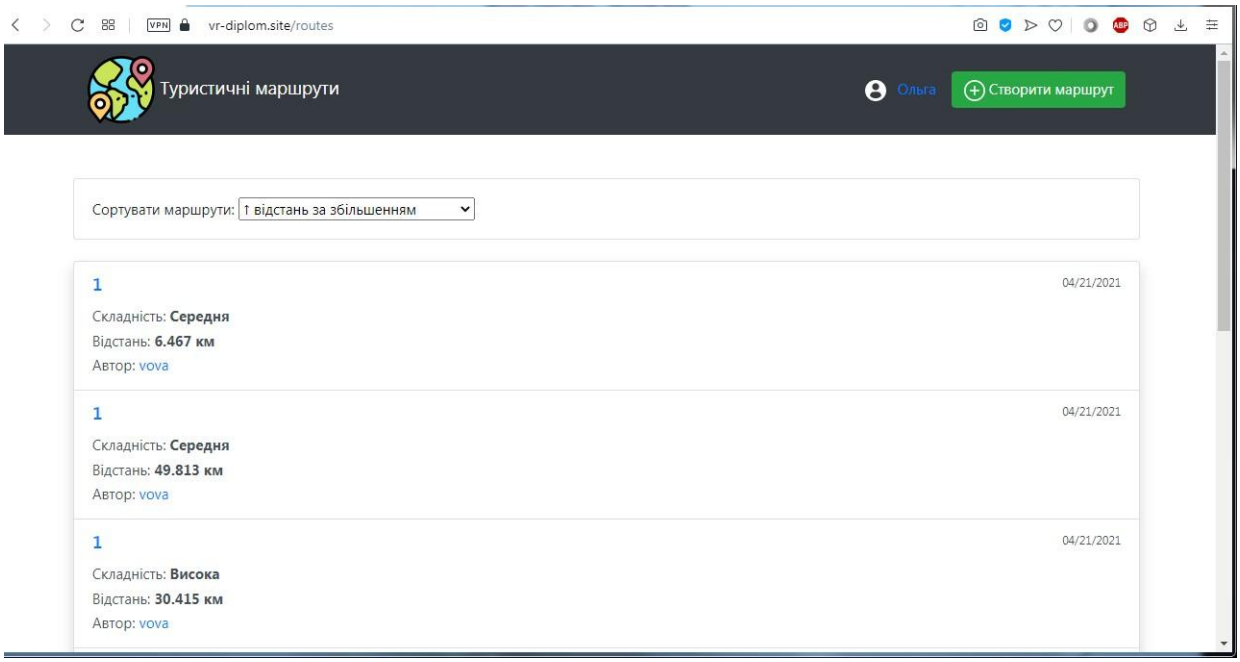

Рисунок 3.17 – Сортування маршруту за збільшенням відстані

Для повернення до головної сторінки з будь-якої іншої користувачу необхідно клацнути по логотипу сервісу в лівому верхньому куті.

Користувач може переглянути певний маршрут, клацнувши по ньому мишею. На екрані виводиться карта даного маршруту та його опис (рис.3.18).

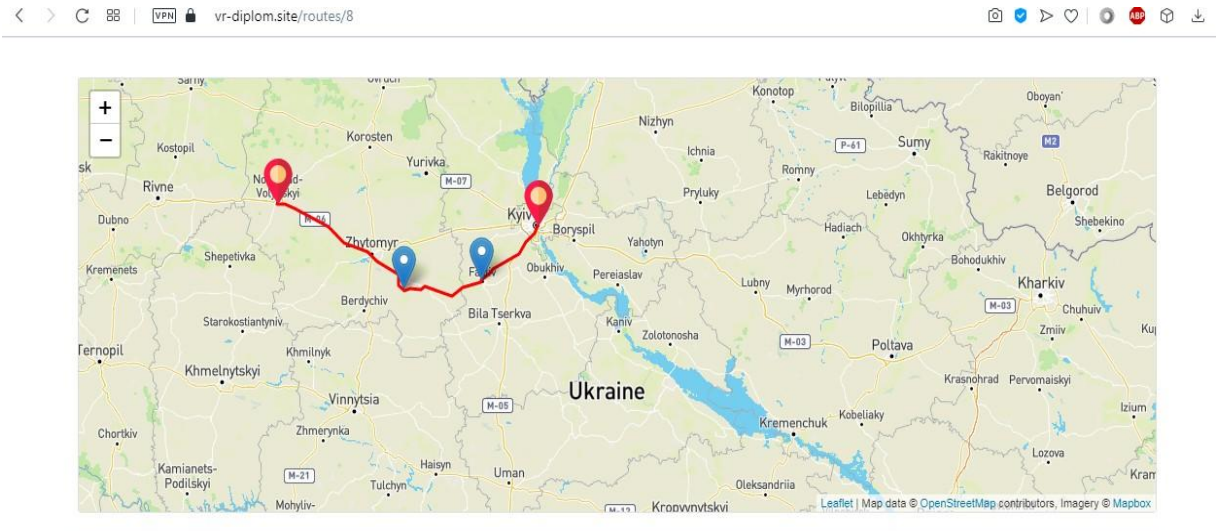

Рисунок 3.18 – Карта маршруту

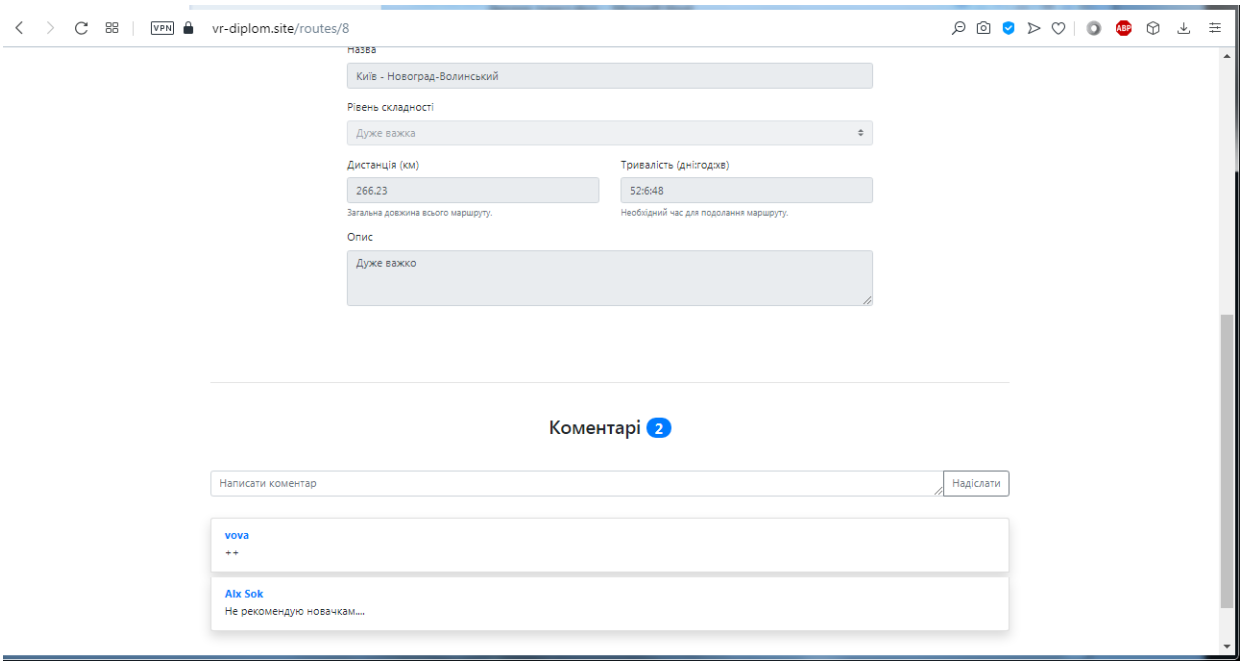

Рисунок 3.19 – Опис обраного маршруту

Для перегляду маршрутів, які буди створені певним користувачем потрібно в списку маршрутів клацнути по імені даного користувача (рис. 3.20). Кожний маршрут можна більш детально переглянути, клацнувши по ньому.

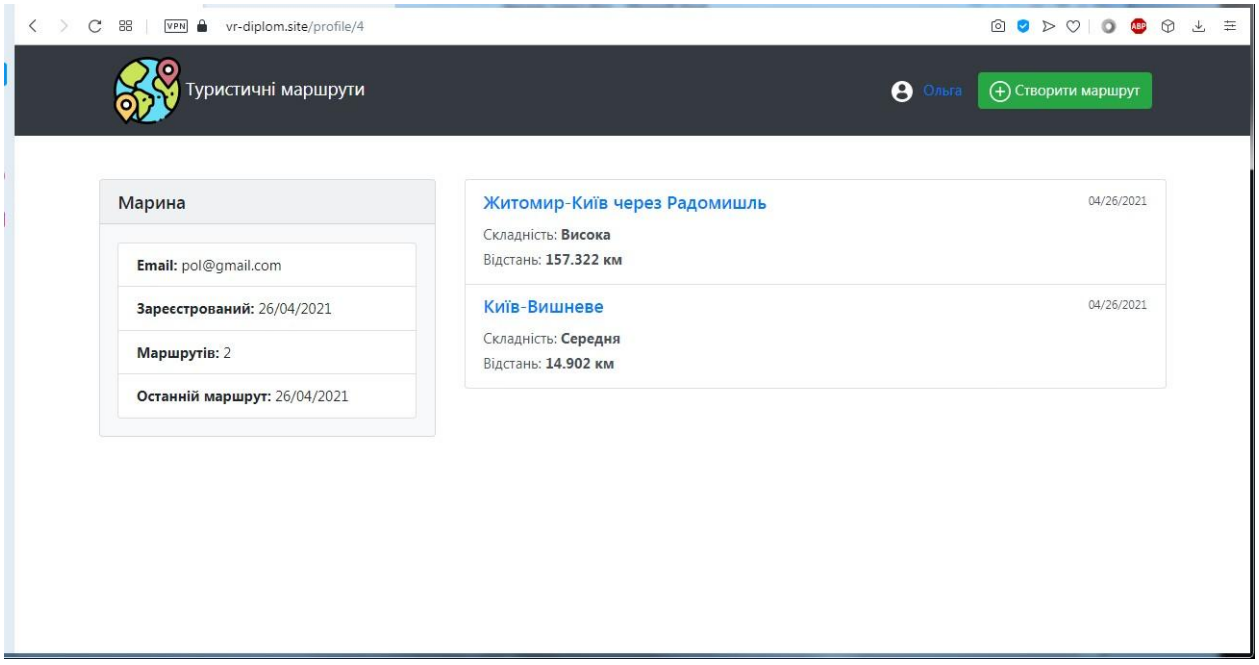

Рисунок 3.20 – Список маршрутів певного користувача

Для створення маршруту користувачу потрібно натиснути кнопку Створити маршрут в правому верхньому куті сторінки. Дана кнопка доступна на

всіх сторінках сервісу. Після цього користувач переходить до сторінки створення маршруту (рис.3.21).

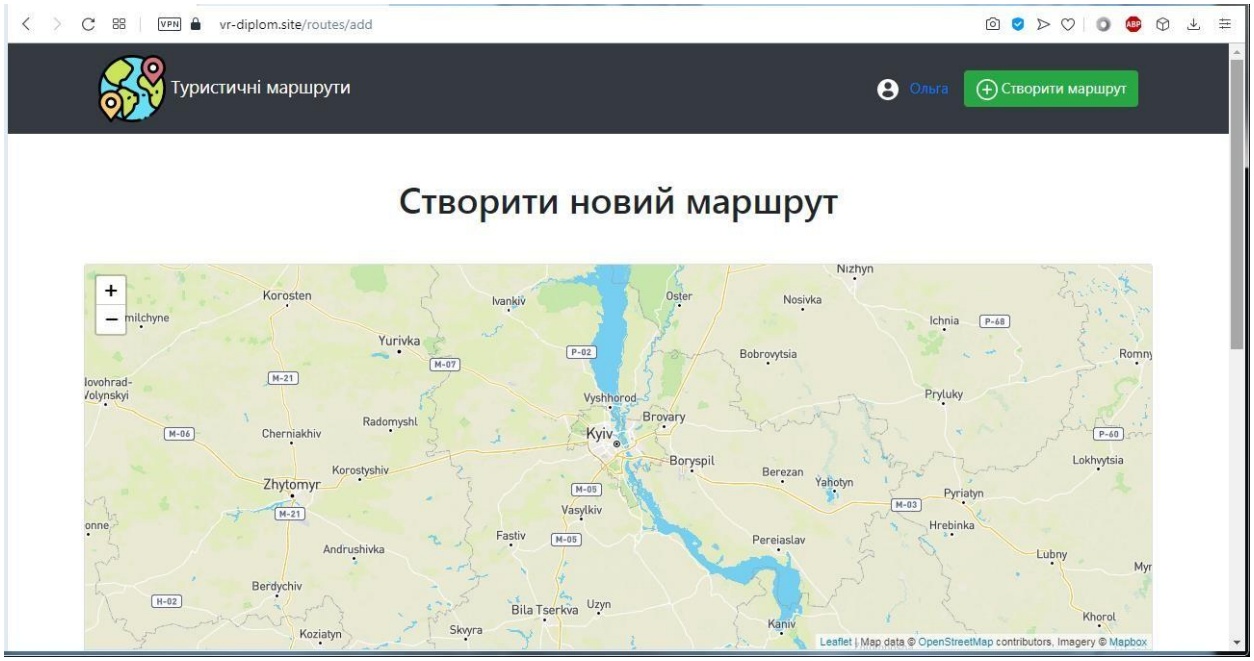

Рисунок 3.21 – Сторінка створення нового маршруту

Користувач на карті визначає точку початку маршруту, проміжні точки та кінцеву точку маршруту і натискає кнопку Прокласти піший маршрут.

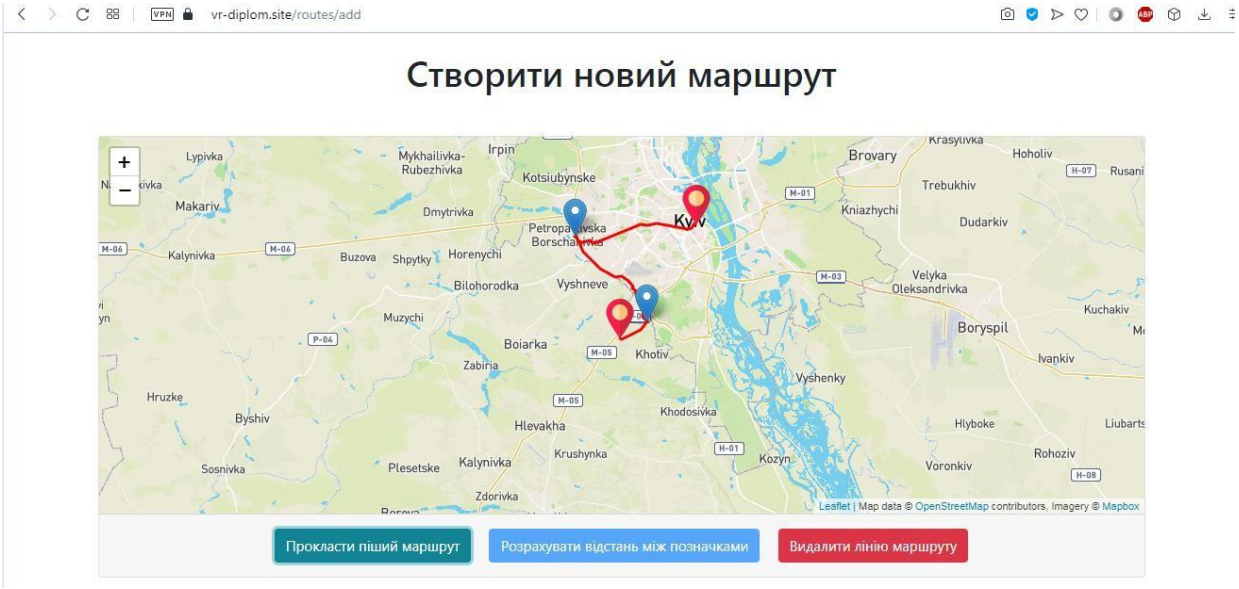

Рисунок 3.21 – Прокладання маршруту

Після цього користувачу потрібно ввести назву маршруту, його складність, опис (рис.3.22).

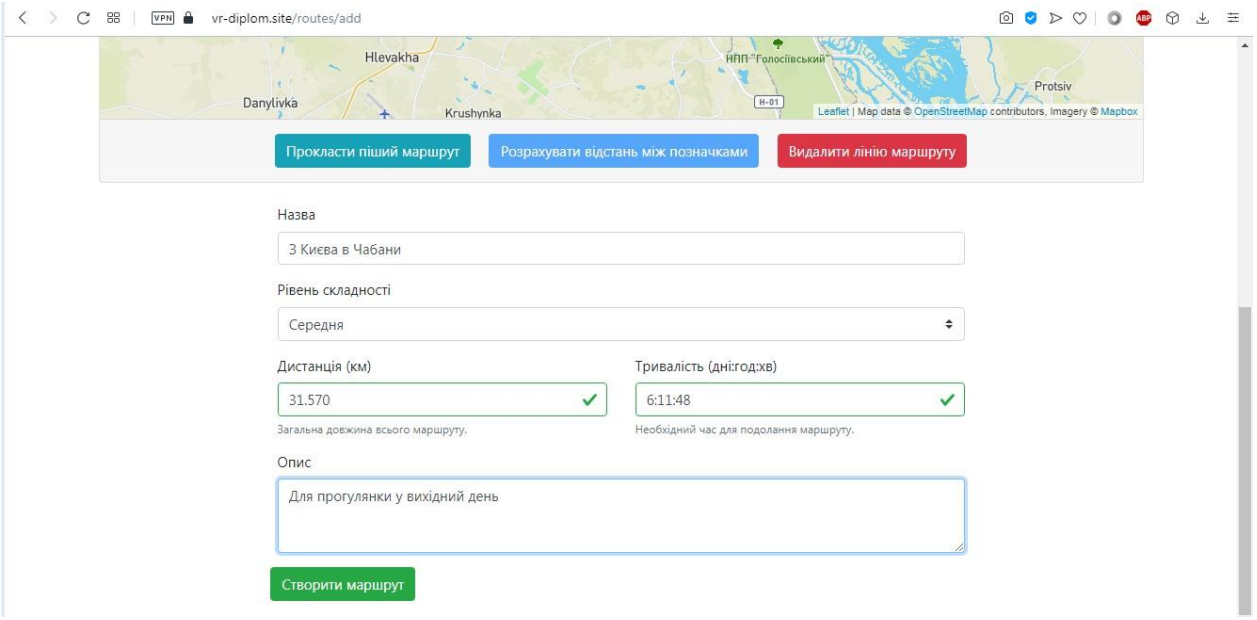

Рисунок 3.22 – Введення інформації щодо маршруту, який створюється.

Після натискання кнопки Створити маршрут користувач повертається на головну сторінку сервісу, а створений маршрут додається до списку (рис.3.23).

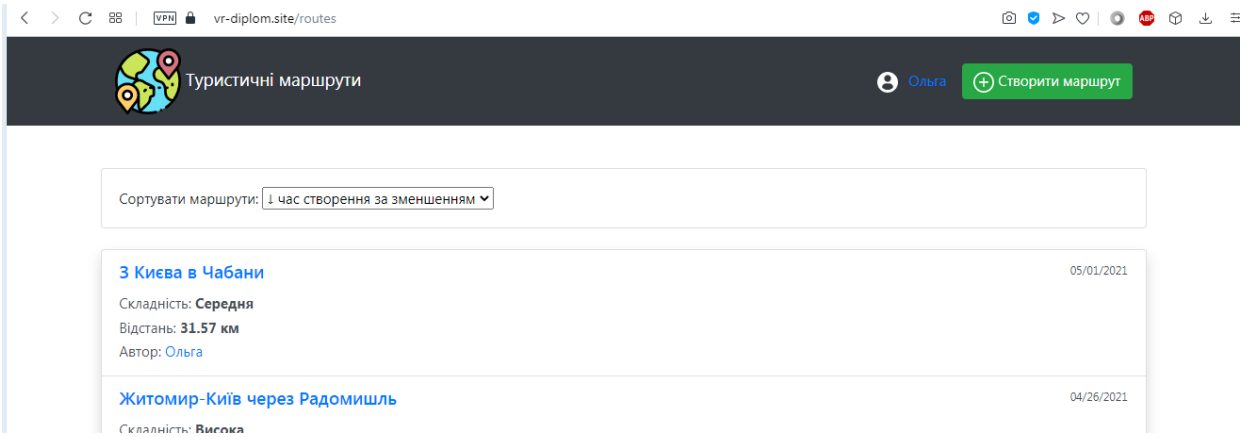

Рисунок 3.23 – Додавання нового маршруту у список

Користувач може редагувати власний маршрут, клацнувши по ньому. В режимі редагування користувач може змінити сам маршрут, відредагувати інформацію про маршрут (рис.3.24).

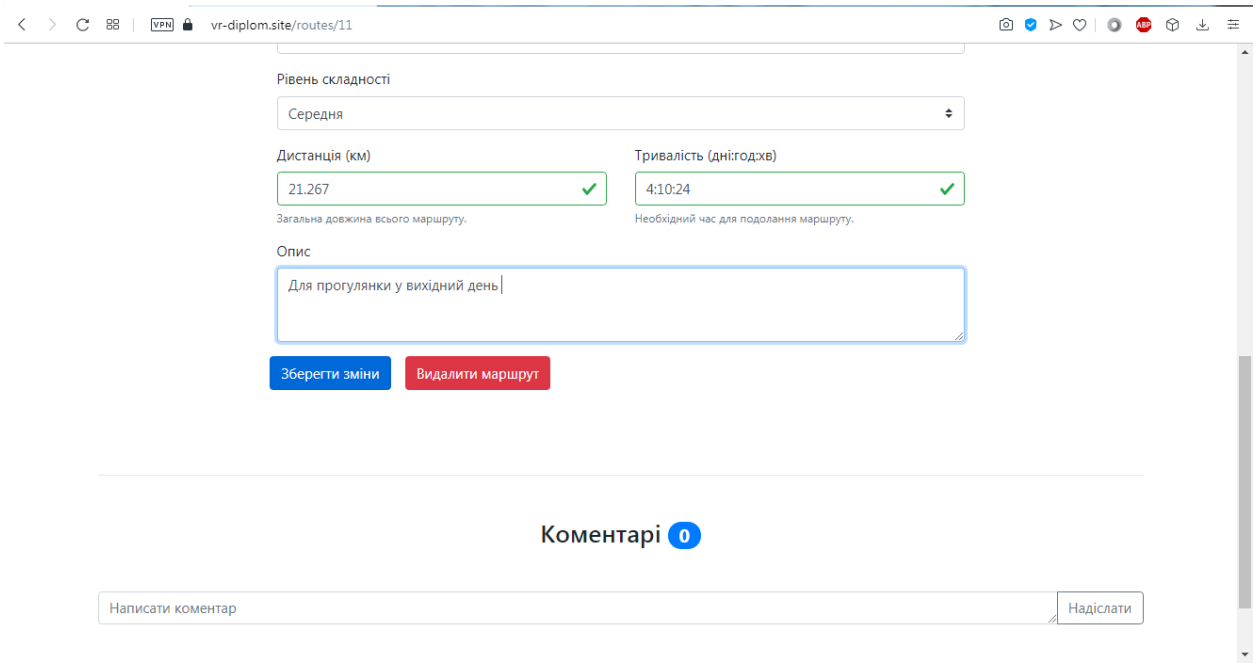

Рисунок 3.24 – Редагування маршруту

Проведене тестування показало, що сервіс відповідає визначеним функціональним вимогам, має достатньо простий, інтуїтивно зрозумілий інтерфейс, що дозволяє рекомендувати його для використання користувачам, які мають бажання спланувати туристичні прогулянки Україною.

Висновки до розділу 3.

На основі визначених вимог було проведене проектування інтерфейсу користувача розроблювальної системи з використанням вайрфреймів.

Використовуючи розроблену архітектури системи були визначені інструменти для програмування сервісу з планування туристичних маршрутів.

Проведене тестування розробленого додатка показало його відповідність визначеним специфікаціям.

ВИСНОВКИ

В даному дослідженні була розв’язана задача побудови Web-сервісу для планування туристичних маршрутів.

Використання веб-сервісів є одним з найбільш перспективних напрямків туристичної діяльності. Їх інтерактивність, орієнтованість на сучасні Інтернет-технології дасть можливість з одного боку збільшити цільову аудиторію, з іншого боку залучити клієнтів до активної участі в розробці нових послуг. Недостатня кількість таких ресурсів в Україні зумовлює підвищений інтерес до розробки та впровадження нових програмних розробок.

Для досягнення поставленої мети дослідження були виконані такі задачі:

1. Проведений аналіз впливу сучасних Інтернет-технологій на туристичну галузь. Було показано, що простота та розширюваність веб-технологій забезпечує зміну в бізнес-процесах, які відбуваються у туристичній галузі, внаслідок чого в формуванні туристичних продуктів активно включаються користувачі.

2. Був проведений аналіз сучасних підходів до створення веб-сервісів, виявлені їх недоліки та переваги. Було показано, що одним з перспективних підходів є архітектурний стиль REST, який орієнтований на використання протоколів HTTPS.

3. На основі проведеного аналізу були сформовані функціональні та нефункціональні вимоги до побудови програмної системи з планування маршрутів.

4. Був розроблений логічна модель веб-сервісу з планування маршрутів на основі об’єктно-орієнтованого підходу з використанням UML.

5. Було здійснено проектування архітектури розроблювальної системи. Враховуючи вимоги до сервісу та особливості його функціонування, був обраний стиль REST та використана трирівнева клієнт-серверна архітектура проекту.

6. На основі логічної моделі веб-сервісу було проведене проектування бази даних.

7. Був реалізований веб-сервіс з використанням сучасних інструментальних засобів розробки та проведене його тестування, результати якого свідчать про відповідність сервісу заданим вимогам.

Використання сучасних архітектурних підходів та алгоритмів реалізації роботи зі складними структурами даних при розробці туристичного веб-сервісу є підґрунтям для теоретичної цінності роботи.

Практична цінність роботи полягає в можливості використання даного продукту при плануванні подорожей Україною.

В подальшому передбачається збільшити кількість параметрів, які можна використовувати при плануванні маршруту та підключенні додаткових сервісів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Артеменко О.І., Пасічник В.В., Єгорова В.В. Інформаційні технології в галузі туризму. Аналіз застосувань та результати досліджень. URL: http://science.lpnu.ua/sites/default/files/journal-paper/2017/may/2615/814ism2015min3-22_0.pdf (дата звернення 25.04.2021)

2. Архітектури та технології WEB-служб. Конспект лекцій для студентів спеціальності 121 Інженерія програмного забезпечення усіх форм навчання. URL: http://eir.zntu.edu.ua/bitstream/123456789/2852/1/Stepanenko_Summary_of_lectures.pdf (дата звернення 20.04.2021)

3. Басс Л., Клементс П., Кацман Р. Архітектура програмного забезпечення на практиці. С-Петербург: Питер, 2006. 576 с.

4. Бібо Б., Кац І. jQuery. Докладне керівництво по просунутому JavaScript. С-Петербург : Пітер. 2011. 258 с.

5. Брусенцова, Т. П. Проектування інтерфейсів користувача: посібник для студентів спеціальності 1-47 01 02 «Дизайн електронних і Web-видань». Минск : БГТУ, 2019. 172 с.

6. Веб-сервіси як засіб інтеграції додатків у WWW. URL: http://www.4stud.info/networking/web-services.html (дата звернення 23.4.2021).

7. Глєбова А. О. Інноваційні технології в туризмі. *Економіка. Управління. Інновації*. 2012. Випуск 2 (8).

8. Загальні архітектури веб-додатків. Документація Microsoft. URL: https://docs.microsoft.com/ru-ru/dotnet/standard/modern-web-apps-azure-architecture/common-web-application-architectures (дата звернення 23.4.2021).

9. Засоби створення Web-сайтів. Введенння в розробку Web-додатків URL: http://5fan.ru/wievjob.php?id=51474 (дата звернення 17.04.2021).

10. Коба С., Калмиков П., Вайленко І. Веб-технології та веб-дизайн. Харків : Нац. аерокосм, ун-т ім. М. Є. Жуковського «Харк. авіаційн. ін-т», 2016. 44 с.

11. Котеров Д., Костарев А. PHP 5. 2-е вид., перероб. і доп. С-Петербург : БХВ-Петербург, 2008.

12. Кукліна Т. С. Використання інформаційних технологій в діяльності туристичних підприємств. *Причорноморські економічні студії. Математичні методи, моделі та інформаційні технології в економіці*. 2017. Випуск 13-2. С.217-221.

13. Кучеренко К. Розвиток інформаційних технологій та їх запровадження у діяльність підприємств туристичної сфери. *Вісник Київського національного університету. Економіка*. 2014. Вип. 10(163). С.31-35.

14. Майер Е. CSS-каскадні таблиці стилей. Детальне керівництво. 3е вид. С-Петербург: Символ-Плюс, 2008. 576 с.

15. Маклафлін Б. Об'єктно-орієнтований аналіз і проектування. Бостон .: O'Reilly. 258 с.

16. Макфарланд Д. Велика книга CSS3 - CSS3. С-Петербург: Пітер, 2016. 251 с.

17. Макфарланд Д. С. Розробка сайтів на JavaScript і jQuery. Вичерпне керівництво. Бостон : O'Reilly, 2012. 587 с.

18. Мінухін С. В. Методи і моделі проектування на основі сучасних CASE–засобів. Харків: Вид. ХНЕУ, 2008. 272 с

19. Одиночкина С. В. Розробка WEB-сервісів в інфокомунікаційних системах 2013. 124 с.

20. Олейніков І. Кращі ідеї застосування ГІС. Харків. ГІС-форум. URL: http://gisforum. org.ua/files/2016 /contest/oleynikov.pdf (дата звернення 1.4.2021)

21. Програмні засоби для розробки web-сайта. URL: http://zdamsam.ru/a19366.html (дата звернення 18.04.2021).

22. Рябов В. А., Несвижский А. И. Сучасні web-технології. Москва: БИНОМ. Лаборатория знаний. 2006. 432 с.

23. Синица С. Г. Веб-програмування і веб-сервіси. Краснодар: Кубанский гос. ун-т, 2013. 158 с.

24. Сичев А. В. Web-технології. URL: http://www.intuit.ru/department/internet/webtechno/ (дата звернення 21.04.2021).

25. Сичев А. В. Перспективні технології та мови веб-розробки. URL: http://www.intuit.ru/department/internet/atlwebdev/ (дата звернення 7.04.2021)

26. Створення сайта, просування сайта, ррозробка сайтів. URL: http://vismech.ru/google/indeksy-google/ (дата звернення 08.04.2021).

27. Столлігнс В. Комп'ютерні мережі, протоколи і технології Інтернета. Москва: BHV, 2005. 817 с.

28. Технологія REST API URL: http://fb.ru/article/338879/rest-api---chto-eto-rest-perevod-representational-state-transfer (дата звернення 05.04.2021).

29. Фастовський Е.Г. Сервіс-орієнтовані технології інтеграції інформації. Лекції. Харків: НТУХПИ, 2011. 50 с.

30. Фрімен Е., Робсон Е. Head First JavaScript Programming. Бостон : O'Reilly 2010. 267 с.

31. Шилдт Г. Java 8. The Complete Reference, 9th Edition. Москва: Вильямс. 2015

32. Що таке веб-сервер? VPS Хостінг. URL: https://vmlab.ru/Articles/What_is_web_server (дата звернення 23.4.2021).

33. Arsanjani A. Поради з програмування Web-сервісів: Сервіс-орієнтоване моделювання і архітектура. URL: http://www.ibm.com/developerworks/ru/library/ws-soa-design1/6 (дата звернення 2.04.2021).

34. Authorization. URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Headers/Authorization (дата звернення 18.04.2021).

35. Bootstrap. The world's most popular mobile-first and responsive front-end framework. URL: https://getbootstrap (дата звернення 06.04.2021).

36. Damianos G., Charalampos K., Mastakas K., Grammati P. Mobile recommender systems in tourism. *Journal of Network and Computer Applications.* 2014. Volume 39. P. 319–333.

37. Dobing B., Parson J. How UML is Used. *CACM.* 2006. Volume 49, #5.

38. Doctrine. URL: https://www.doctrine-project.org/ (дата звернення 18.04.2021).

39. Encoded Polyline Algorithm Format. URL: https://developers.google.com/maps/documentation/utilities/polylinealgorithm (дата звернення 18.04.2021).

40. Erickson J., Siau K. Can UML Be Simplified? Practitioner Use of UML in Separate Domains. Нью-Йорк: 2007.

41. Gruman G. SOA's Technology Underpinnings URL: https://www.cio.com/article/2446812/service-oriented-architecture/soa-s-technology-underpinnings.html (дата звернення 05.04.2021).

42. HTML 5. URL: https://ru.wikipedia.org/wiki/HTML5 (дата звернення 10.04.2021)

43. JavaScript, HTML, DOM. URL: https://www.w3schools.com/js/js_htmldom.asp (дата звернення 10.04.2021).

44. JavaScript. URL: https://learn.javascript.ru/ (дата звернення 06.04.2021).

45. jQuery API Documentation. URL: https://api.jquery.com/ (дата звернення 11.04.2021).

46. Khan A. Automatic tour guide system *International Journal of Scientific & Engineering Research.* 2013. Volume 4, Issue 6.

47. Khan Y.H., Hossain A. The effect of ICT Application on the Toirism and Hospitality Industries in London. *SocioEconomic Challenges*. 2018. Volume 2, Issue 4. DOI:10.21272/sec.4(2).60-68.2018

48. Minić N., Njeguš A., Ceballos J.T. The impact of WEB 3.0 technologies on tourism information systems. DOI: 10.15308/SInteZa-2014-781-787

49. MySQL. URL: https://www.mysql.com/ (дата звернення 10.04.2021)

50. Recommendations of the National Institute of Standards and Technology. URL: http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf (дата звернення 2.04.2021)

51. SOA Platforms / Hailstone R., Illsley R., Jones T., Kellet A. Butler Group, 2007. 243 p.

52. SOAP Version 1.2 Part 0: Primer (Second Edition). URL: http://www.w3.org/TR/2007/REC-soap12-part0-20070427/ (дата звернення 17.03.2021).

53. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). URL: http://www.w3.org/TR/2007/REC-soap12-part1-20070427/ (дата звернення 17.03.2021).

54. Standard W3C: Selectors Level 3. W3C Recommendation 29 September 2011. W3C. URL: http://www.w3.org/TR/css3-selectors/ (дата звернення 11.04.2021).

55. Symfony Documentation. URL: https://symfony.com/ doc/current/index.html (дата звернення 18.04.2021).

56. Tagliaferri L. An Introduction to JSON. URL: https://www.digitalocean.com/community/tutorials/an-introduction-to-json (дата звернення 22.04.2021)

57. UML Diagram. URL: https://www.smartdraw.com/uml-diagram (дата звернення 15.04.2021).

58. Wrycza S., Marcinkowski B. Towards a Light Version of UML2.X: Appraisal and Model, 2007.

ДОДАТКИ

Додаток А.1

Лістинг основної сторінки веб-сервісу

```
<html lang="en" class="h-100">

<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Всі маршрути | Планувальник туристичних маршрутів</title>
   <link rel="stylesheet" href="/assets/vendor/bootstrap/css/bootstrap.min.css">
</head>
<body class="position-relative" style="min-height: 100%;">
<header class="bg-dark">
   <div class="container">
      <nav class="navbar navbar-expand-lg navbar-dark ">
         <a class="navbar-brand" href="/routes">
            <img src="/assets/img/tourism.svg" width="70" height="70" alt="Лого">
            Туристичні маршрути
         </a>

         <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
            <span class="navbar-toggler-icon"></span>
         </button>

         <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav mr-auto">
                        </ul>

            <div class="">
```

```
                <a class="btn btn-link" href="/profile/4">
                    <img src="/assets/img/profile-user.svg" class="mr-2" width="22" height="22"
alt="Турист">
                    Марина
                </a>
                <a href="/routes/add" class="btn btn-success">
                    <img src="/assets/img/add.svg" width="22" height="22" alt="Турист">
                    Створити маршрут
                </a>
            </div>
        </div>
    </nav>
  </div>
</header>

<div class="container py-5 position-relative" style="z-index: 99">
  <div class="card mb-4">
    <div class="card-body">
      <div>
        <span>Сортувати маршрути: </span>
        <select id="sort">


                        <option value="created_desc" selected>
                    &darr; час створення за зменшенням
                </option>
                        <option value="created_asc" >
                    &uarr; час створення за збільшенням
                </option>
                        <option value="distance_desc" >
                    &darr; відстань за зменшенням
                </option>
                        <option value="distance_asc" >
                    &uarr; відстань за збільшенням
                </option>
```

```
                        <option value="duration_desc" >
                    &darr; тривалість за зменшенням
                </option>
                        <option value="duration_asc" >
                    &uarr; тривалість за збільшенням
                </option>

            </select>
        </div>
    </div>
</div>

<div class="list-group shadow rounded bg-white">
            <div class="list-group-item list-group-item-action">
<div class="mb-1 d-flex w-100 justify-content-between">
    <a href="/routes/10" class="text-primary h5">
        Житомир-Київ через Радомишль
    </a>
    <small>04/26/2021</small>
</div>

<p class="mb-1">
    Складність:
    <span class="font-weight-bold">
        Висока
    </span>
</p>

<p class="mb-1">
    Відстань:
    <span class="font-weight-bold">
        157.322 км
    </span>
</p>
```

```
        <p class="mb-1">
        Автор:
        <a href="/profile/4">
            Марина
        </a>
    </p>
</div>
            <div class="list-group-item list-group-item-action">
<div class="mb-1 d-flex w-100 justify-content-between">
    <a href="/routes/9" class="text-primary h5">
        Київ-Вишневе
    </a>
    <small>04/26/2021</small>
</div>

<p class="mb-1">
    Складність:
    <span class="font-weight-bold">
        Середня
    </span>
</p>

<p class="mb-1">
    Відстань:
    <span class="font-weight-bold">
        14.902 км
    </span>
</p>


        <p class="mb-1">
        Автор:
        <a href="/profile/4">
            Марина
```

```
        </a>
    </p>
</div>
            <div class="list-group-item list-group-item-action">
<div class="mb-1 d-flex w-100 justify-content-between">
    <a href="/routes/8" class="text-primary h5">
        Київ - Новоград-Волинський
    </a>
    <small>04/21/2021</small>
</div>

<p class="mb-1">
    Складність:
    <span class="font-weight-bold">
        Дуже важка
    </span>
</p>

<p class="mb-1">
    Відстань:
    <span class="font-weight-bold">
        266.23 км
    </span>
</p>



        <p class="mb-1">
        Автор:
        <a href="/profile/1">
            Alx Sok
        </a>
    </p>
</div>
            <div class="list-group-item list-group-item-action">
<div class="mb-1 d-flex w-100 justify-content-between">
```

```
    <a href="/routes/7" class="text-primary h5">
        Біла Церква - Узин
    </a>
    <small>04/21/2021</small>
</div>

<p class="mb-1">
    Складність:
    <span class="font-weight-bold">
        Середня
    </span>
</p>

<p class="mb-1">
    Відстань:
    <span class="font-weight-bold">
        25.758 км
    </span>
</p>


        <p class="mb-1">
        Автор:
        <a href="/profile/3">
            Катя
        </a>
    </p>
</div>
            <div class="list-group-item list-group-item-action">
<div class="mb-1 d-flex w-100 justify-content-between">
    <a href="/routes/6" class="text-primary h5">
        4
    </a>
    <small>04/21/2021</small>
</div>
```

```
<p class="mb-1">
   Складність:
   <span class="font-weight-bold">
      Дуже важка
   </span>
</p>

<p class="mb-1">
   Відстань:
   <span class="font-weight-bold">
      67.629 км
   </span>
</p>



      <p class="mb-1">
      Автор:
      <a href="/profile/2">
         vova
      </a>
   </p>
</div>
            <div class="list-group-item list-group-item-action">
<div class="mb-1 d-flex w-100 justify-content-between">
   <a href="/routes/4" class="text-primary h5">
      1
   </a>
   <small>04/21/2021</small>
</div>

<p class="mb-1">
   Складність:
   <span class="font-weight-bold">
      Висока
```

```
    </span>
</p>

<p class="mb-1">
    Відстань:
    <span class="font-weight-bold">
        30.415 км
    </span>
</p>



        <p class="mb-1">
        Автор:
        <a href="/profile/2">
            vova
        </a>
    </p>
</div>
            <div class="list-group-item list-group-item-action">
<div class="mb-1 d-flex w-100 justify-content-between">
    <a href="/routes/3" class="text-primary h5">
        1
    </a>
    <small>04/21/2021</small>
</div>

<p class="mb-1">
    Складність:
    <span class="font-weight-bold">
        Середня
    </span>
</p>

<p class="mb-1">
    Відстань:
```

```
    <span class="font-weight-bold">
      49.813 км
    </span>
</p>


      <p class="mb-1">
      Автор:
      <a href="/profile/2">
        vova
      </a>
    </p>
</div>
            <div class="list-group-item list-group-item-action">
<div class="mb-1 d-flex w-100 justify-content-between">
    <a href="/routes/2" class="text-primary h5">
      1
    </a>
    <small>04/21/2021</small>
</div>

<p class="mb-1">
    Складність:
    <span class="font-weight-bold">
      Середня
    </span>
</p>

<p class="mb-1">
    Відстань:
    <span class="font-weight-bold">
      6.467 км
    </span>
</p>
```

```
            <p class="mb-1">
            Автор:
            <a href="/profile/2">
               vova
            </a>
          </p>
    </div>
            </div>
</div>


<div style="
    z-index: 1;
    position: absolute;
    bottom: 0; left: 0; right: 0;
    width: 100%;
    height: 226px;
    opacity: 0.9;
    background-size: cover;
    background-image: url(/assets/img/trail-artwork-greentheme2.png)"></div>

<script>
    var sortSelect = document.getElementById('sort');
    sortSelect.addEventListener('change', function (e) {
        var sortType = this.options[this.selectedIndex].value;
        window.location.href = '?sort=' + sortType;
    });
</script>

    <script src="/assets/vendor/jquery/jquery.min.js"></script>
    <script src="/assets/vendor/bootstrap/js/bootstrap.min.js"></script>
</body>
</html>
```

Додаток А.2

Лістинг файла Kernel.php

```
<?php
namespace App;
use Symfony\Bundle\FrameworkBundle\Kernel\MicroKernelTrait;

use Symfony\Component\DependencyInjection\Loader\Configurator\ContainerConfigurator;

use Symfony\Component\HttpKernel\Kernel as BaseKernel;
use Symfony\Component\Routing\Loader\Configurator\RoutingConfigurator;
class Kernel extends BaseKernel
{
    use MicroKernelTrait;
     protected function configureContainer(ContainerConfigurator $container): void
         {
  $container->import('../config/{packages}/*.yaml');
           $container->import('../config/{packages}/'.$this->environment.'/*.yaml');
           if (is_file(\dirname(__DIR__).'/config/services.yaml'))
              {
   $container->import('../config/services.yaml');
           $container->import('../config/{services}_'.$this->environment.'.yaml');
                }
             elseif (is_file($path = \dirname(__DIR__).'/config/services.php'))
                {
    (require $path)($container->withPath($path), $this);
                 }
           }
      protected function configureRoutes(RoutingConfigurator $routes): void
       {
    $routes->import('../config/{routes}/'.$this->environment.'/*.yaml');
           $routes->import('../config/{routes}/*.yaml');
       if (is_file(\dirname(__DIR__).'/config/routes.yaml'))
                {
  $routes->import('../config/routes.yaml');
                 }
```

```
            elseif (is_file($path = \dirname(__DIR__).'/config/routes.php'))
                {
    (require $path)($routes->withPath($path), $this);
                }
        }
}
```

Додаток А.3

Лістинг файла Контроллер RoutesController.php

```
<?php
namespace App\Controller;
use App\Entity\Comment;
use App\Entity\Route;
use App\Entity\RouteLevel;
use App\Repository\RouteLevelRepository;
use App\Repository\RouteRepository;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Exception\BadRequestException;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpKernel\Exception\BadRequestHttpException;
use Symfony\Component\Routing\Exception\InvalidParameterException;
class RoutesController extends AbstractController
{
   public function addRoute(RouteRepository $routeRepository, RouteLevelRepository
$routeLevelRepository)
    {
       $user = $this->isAuthorized();
           $req = Request::createFromGlobals();
           $route = $this->composeRouteEntity(new
Route(),$req,$routeRepository,$routeLevelRepository);
       $route->setUser($user);
       $routeRepository->add($route);
       return new Response($route->getId());
   }

public function editRoute(RouteRepository $routeRepository, RouteLevelRepository
$routeLevelRepository)
   {
       $user = $this->isAuthorized();
           $req = Request::createFromGlobals();
```

```
        $id = $req->request->get('id');
        if (!isset($id)) throw new InvalidParameterException('Відсутнє id маршруту.');
    $route = $routeRepository->find($id);
    if (!isset($route))
        {           throw new InvalidParameterException('Маршрут не знайдено.');    }
    if ($route->getUser()->getId() !== $user->getId())
        {          throw $this->createAccessDeniedException();       }
    $route = $this->composeRouteEntity($route,$req,$routeRepository,$routeLevelRepository);
       $route->setUpdated();
    $routeRepository->flush();
    return new Response('ok');
  }

  public function deleteRoute(RouteRepository $routeRepository)
  {
    $user = $this->isAuthorized();
    $req = Request::createFromGlobals();
    $id = $req->request->get('id');
    if (!isset($id)) throw new InvalidParameterException('Відсутнє id маршруту.');
    $ok = $routeRepository->remove($id);
    if (!$ok) throw new BadRequestHttpException();
        return new Response('ok');
    }
  public function addComment(RouteRepository $routeRepository)
  {
    $user = $this->isAuthorized();
    $req = Request::createFromGlobals();
    $message = $req->request->get('message');
    $routeId = $req->request->get('route-id');
        $route = $routeRepository->find($routeId);
        if (!isset($route) || !isset($message))
        { throw new BadRequestHttpException('Помилка в параметрах запиту. Перевірте
route-id, message');     }
        $comment = new Comment();
        $comment->setUser($user);
```

```
        $comment->setRoute($route);
            $comment->setMessage($message);
            $route->addComment($comment);
        $routeRepository->flush();
            return new JsonResponse([
                    "user_id" => $comment->getUser()->getId(),
                    "user_name" => $comment->getUser()->getName(),
                    "message" => $comment->getMessage()
        ]);
      }
     private function isAuthorized()
      {
       $user = $this->getUser();
        if (!isset($user) || $user->getId() === null)
              {          throw $this->createAccessDeniedException();       }
          return $user;
    }
    private function composeRoute(EntityRoute $route, Request $req,  RouteRepository
$routeRepository,RouteLevelRepository $routeLevelRepository)
      {
       $routeLevel = $routeLevelRepository->findOneBy(['id' => $req->request->get('difficulty')]);
            $route->setName($req->request->get('name'));
            $route->setDescription($req->request->get('description'));
             $route->setLevel($routeLevel);
            $route->setSteps($req->request->get('steps'));
            $route->setDistance($req->request->get('distance'));
            $route->setDuration($req->request->get('duration'));
            $route->setGeoPolyline($req->request->get('geo-polyline'));
            return $route;
       }
}
```

А.4. Лістинг файла RouteRepository.php

```
<?php
namespace App\Repository;
```

```
use App\Entity\Route;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;
/**
 * @method Route|null find($id, $lockMode = null, $lockVersion = null)
*     @method Route|null findOneBy(array $criteria, array $orderBy = null)
*     @method Route[]    findAll()
*     @method Route[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
*/
class RouteRepository extends ServiceEntityRepository
{
    private $manager;
    public function__construct(ManagerRegistry $registry)
  {  parent::_construct($registry, Route::class);
$this->manager = $this->_em;
   }
public function add(Route $route): Route
   {
      $this->manager->persist($route);
          $this->manager->flush();
      return $route;
   }
public function flush()
   {
      $this->manager->flush();
     }
public function persist(Route $entity)
   {
      $this->manager->persist($entity);
   }
public function remove(int $id)
  {
      $route = $this->find($id);
         if (!isset($route)) return null;
      $this->manager->remove($route);
```

```
        $this->manager->flush();
        return true;
    }
}
```