Форма № Н-9.02

Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)
Навчально-науковий інститут інформаційних технологій та робототехніки
(повна назва факультету)
Кафедра комп'ютерних та інформаційних технологій і систем
(повна назва кафедри)

Пояснювальна записка
до дипломного проекту (роботи)
_____магістра_____
(рівеньвищої освіти)
на тему
Інтелектуальний мобільний додаток подорожей Україною

Виконала: студентка 6 курсу, групи 601-ТН
спеціальності
_____122_____Комп'ютерні науки_____
(шифр і назва напряму)
_____Коверда К. М._____
(прізвище та ініціали)
Керівник _____Альошин С. П._____
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Полтава – 2021 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Спеціальність 122 «Комп'ютерні науки»

на тему

«Інтелектуальний мобільний додаток подорожей Україною»

Студентки групи 601-ТН Коверди Каріни Миколаївни

Керівник роботи
кандидат технічних наук,
доцент Альошин С.П.

Завідувач кафедри
кандидат технічних наук,
доцент Головко Г.В.

Полтава – 2021

РЕФЕРАТ

Кваліфікаційна робота магістра: 96с., 34 малюнки, 5 додатків, 34 джерела.

Об’єкт дослідження: діяльність мережі веб-додатку подорожей.

Мета роботи: розроблення веб-ресурсу подорожей Україною за областями.

Методи: розробка та проектування бази даних для функціонування веб-додатку подорожей, програмна реалізація інтерфейсу користувача та розробка інтерфейсу для адміністрування веб-ресурсу.

Ключові слова: веб-додаток, подорожі, база даних, веб-ресурс, програмний код, програмний модуль.

ANNOTATION

Master's qualification work: 96 pages., 34 figures, 5 appendices, 34 sources.

Object of research: travel web application network activity.

Purpose of the work: development of a web resource for travel in Ukraine by region.

Methods: development and design of a database for the operation of a web travel application, software implementation of the user interface and development of an interface for web resource administration**.**

Keywords: web application, travel, database, web resource, program code, program module.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних.

ВР – веб-ресурс.

ВС – веб-сервер.

ІК –інтерфейс користувача.

ІС –Інформаційна система.

ОС – операційна система.

ПК –персональний комп’ютер.

ПЗ – програмне забезпечення.

ТЗ –технічне завдання.

HTML – HyperText Markup Language

JS – Java Script.

VSC – Visual Studio Code

ШІ – штучний інтелект

ВСТУП

Одним з активних видів відпочинку для людей у період відпустки, канікул, свят і просто вільного часу – є подорожі.

Відпочинок дарує почуття свіжості, легкості, морального та естетичного задоволення, гармонії з собою та навколишнім середовищем. Тож досить актуальною є проблема вибору того місця для відпочинку, яке може задовольнити всі потреби людей у повній мірі.

У сучасному світі важко уявити життя людини без жодної допомоги технологій та програмних засобів, а саме широковідомих веб-ресурсів. Загальна кількість різнобічних послуг, у мережі Інтернет, збільшується майже щодня, і як наслідок, витісняє їх з реального та повсякденного нам життя. Саме одною з таких послуг є вибір найбільш зручного маршруту подорожей, коли навіть немає нагальної потреби виходити з дому та можна обрати найкращий варіант, який підходить індивідуально до потреб, дистанційно.

Актуальність створення веб-ресурсу подорожей. У даний період часу,великій кількості людей є актуальним питання вибору найбільш зручного та недорого маршруту мандрівки, за яким є можливість охопити найвизначніші місця країни та при цьому мати найменші витрати. На жаль, малий відсоток людей має змогу звернутися до фахівця, щоби максимально корисно спланувати свою поїздку. Для цього існують спеціальні онлайн-сервіси, які допомагають вирішити ту чи іншу проблему. Але не вдалося віднайти жодного ресурсу, який би зміг надати можливість обирати свої власні маршрути, місця де залишитися харчуватися та спати.

Метою створення веб-додатку подорожей є збільшення кількості якісних веб-ресурсів мандрівок, які користувач має змогу самостійно обирати та планувати найбільш досконалі відпустки для себе, своїх рідних та друзів.

Постановка задачі. Задачею дослідження є проектування та програмнареалізація веб-ресурсу подорожей, проектування бази даних для коректного функціонування системи, а також реалізації адміністративної частини веб-сервісу.

Призначення веб-додатку подорожей Україною. Функціонал веб-ресурсу,який надається користувачу–цеможливість перегляду різноманітних місць країни та її найвизначніших місцевостей, у відповідних категоріях, перегляд діючих закладів харчування та сну, відповідно до обраного міста, та можливість планування власних маршрутів, які можуть складатися з декількох міст. Функціональна частина веб-ресурсу для адміністрування дає можливість додання нових закладів відпочинку чи сну, а також периферій на сайт, відстеження та створення нових локацій та маршрутів на сайті.

Даний веб-ресурс може розміщуватися на будь-якому з веб-хостингів, що надає можливість доступу до нього по всій країні та за її межами. Тобто даний додаток має перспективи розвитку в майбутньому, що, в свою чергу, надає можливість розробки більш якісного функціоналу надалі для його користування.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД РЕСУРСІВ ДЛЯ ПЛАНУВАННЯ ПОДОРОЖЕЙ

1.1 Опис предметної області

У даному періоді життя людства подорожі стали невід'ємною частиною життя великої кількості людей. Мандруючи світом, навіть своїми рідними куточками країни, можна значно покращити свій емоційний стан, набратися енергії для нової роботи, переосмислити значну кількість речей та збагатитися силами.

Оскільки зараз складний час для подорожей закордоном, не варто думати, що більше немає чого дивитися і де помандрувати. Україна, як всім відомо, дуже багата на затишні та мальовничі містечка, які ні чим не відрізняються від закордонних туристичних місць. Головне вміти правильно обирати та знаходити дані місцевості.

Багатьом, я впевнена, може здатися, що на це може піти багато затрат, особливо у даний період часу, але не варто так думати, оскільки основні витрати йдуть на дорогу до того чи іншого міста (а коли подорож за кордон, основні фінансові витрати йдуть на дорогу до іншої країни), але ж ми подорожуємо Україною, і це великий плюс. Також основна частина відомих та популярних місць для туристів абсолютно безкоштовна. Саме для того, щоб дізнатися як вірно обирати шлях мандрівки, і як бонус дешево, створені різноманітні веб-додатки подорожей Україною.

Зараз дуже складно, майже неможливо уявити життя сучасної людини без різноманітнихтехнологій та програмних засобів, зокрема популярних веб-ресурсівЗагальна кількість різнобічних послуг, у мережі Інтернет, збільшується майже щодня, і як наслідок, витісняє їх з реального та повсякденного нам життя. Саме одною з таких послуг є вибір найбільш зручного маршруту подорожей, коли навіть немає нагальної потреби

виходити з дому та можна обрати найкращий варіант, який підходить індивідуально до потреб, дистанційно.

У даний період часу багатьом людямзалишається актуальним питання,щодо вибору максимально зручного маршруту подорожей,на яке витрачається мінімум часу На жаль, лише у малої кількості людей є можливість звернутися до кваліфікованого фахівця, щоби спланувати найбільш корисну мандрівку,яка у повній мірі забезпечить потреби користувача. Також для цього існують спеціальні веб-ресурси, але зараз не вдалося знайти ні одного веб-додатку, який би зміг створювати власні маршрути та обирати саме ті місця для відпочинку, які були б максимально цікаві для шукача.

В Україні існує певна кількість додатків,головною метою яких є показ найвизначніших місць країни.Проаналізувавши уже створені схожі веб-ресурси не було знайдено жодного подібного додаткупланування подорожей,де можна було б одразу обирати та створювати свої власні маршрути подорожей, обираючи їх з даного списку областей, що безумовно пояснює та обґрунтовує актуальність розробки даного виду додатку для України.

Серед доступних ресурсів були проаналізовані:

1. Веб-сайт подорожуй Україною “Етнохата” (рис.1.1, доступ: https://etnoxata.com.ua).

2. Веб-сайт “Мандруй Україною” (рис.1.2, доступ: https://ua.discover.ua).

3. Веб-сайт подорожей Україною “Всі тури” (рис.1.3, доступ: https://vsitury.com.ua).

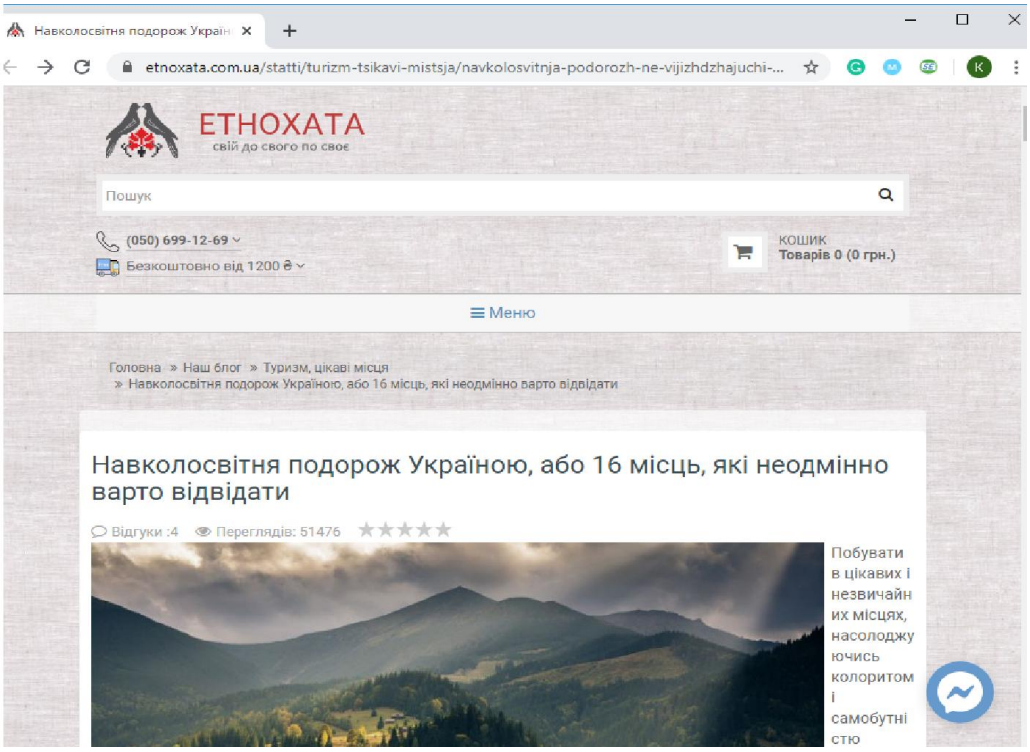

Рисунок 1.1 – Головна сторінка ресурсу подорожуй Україною “Етнохата”

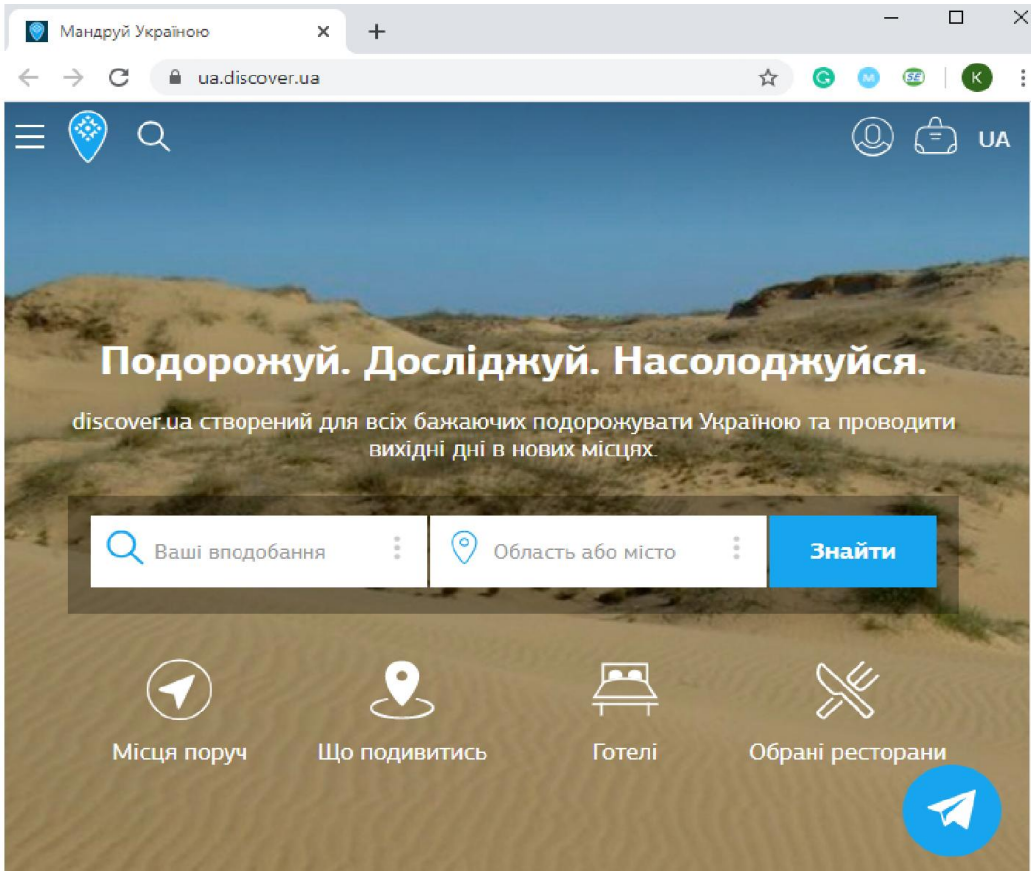

Рисунок 1.2 – Головна сторінка ресурсу “Мандруй Україною”

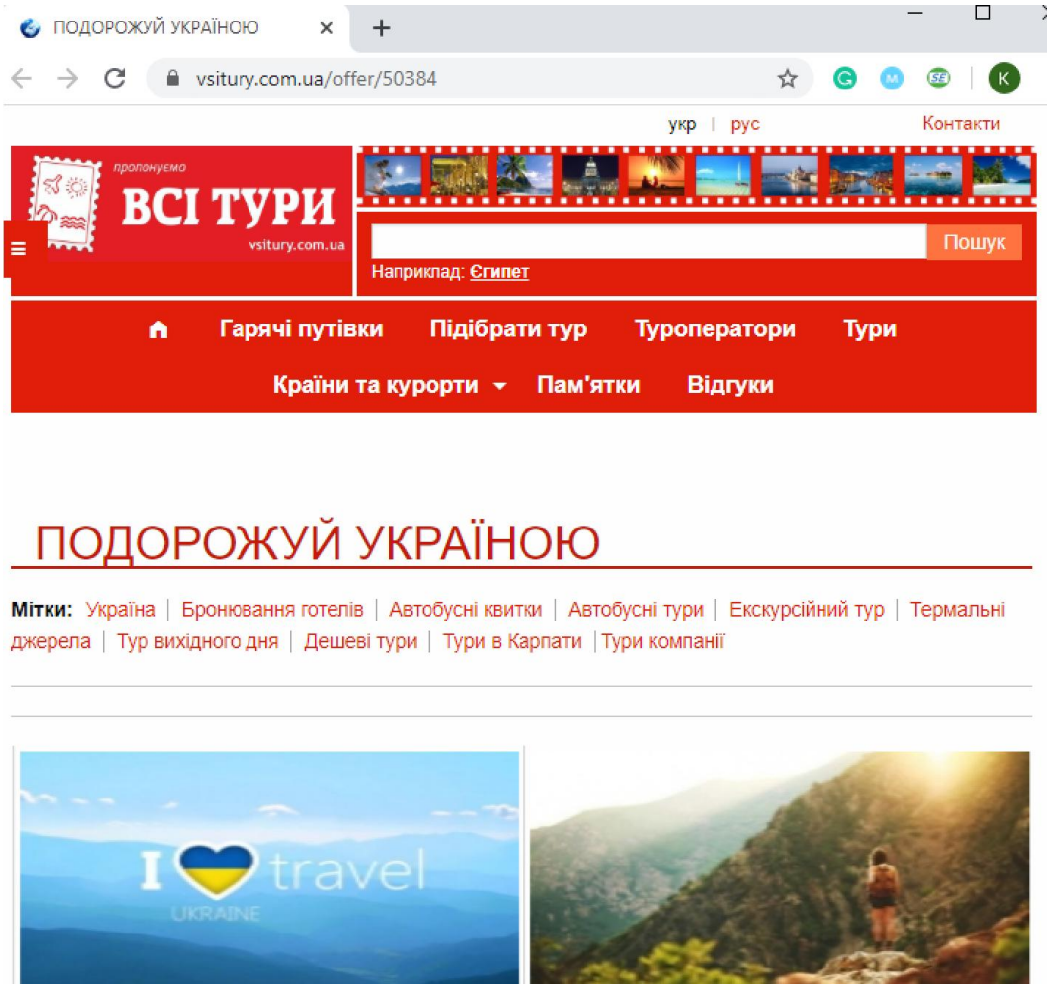

Рисунок 1.3 – Головна сторінка ресурсу “Всі тури”

1.2 Огляд аналогів веб-ресурсів подорожей

Відповідні аналоги розробленого веб-ресурсу мають логічну навігацію, що допомагає користувачам швидко вибиратинеобхідний їм маршрут. Як недолік даних додатків можна віднести відсутність вибору визначного місця по областям, переглядуіснуючих та діючих закладів харчування, інформацію про готелі або хостели, які працюють в даний період часу. Оскільки неповна кількість даних присутня на ресурсах, які розглядалися, можна зробити висновок, що користувачі ознайомлюються з інформацією в неповному обсязі.

Кожен з обраних ресурсів має ненав’язливу кольорову гамму (окрім “Всі тури”, там використано яскравий червоний колір), витриману в більшості випадків, успокійному кольорі, що, в свою чергу, за своїми властивостями не має нав’язливого ефектудля зору людини. Всесвітньовідомий факт, що світлі відтінки насправді заспокоюють користувачів та роблять перебування на веб-сайті найменш травматичним для зору та для психоемоційного стану, що в свою чергу робить їх більш лояльними до вибору маршруту подорожей та збільшує час перебування на даному ресурсі.

1.3 Аналіз функціональних особливостей веб-ресурсів

З технічної сторони реалізації даних проектів було проаналізовано, що ці ресурси не мають зручних дизайнів інтерфейсу користувача саме на мобільних пристроях, що, як наслідок, у час максимального розвитку технологій, стає великим недоліком, адже згідно статистики, відвідуваність будь яких ресурсів відбувається за допомогою мобільних девайсів, тобто телефонів та планшетів, які постійно знаходяться разом з користувачем, аніж зі стаціонарних комп’ютерів чи ноутбуків, оскільки зі зрозумілих причин вони не можуть бути завжди поруч.

Тому особливу увагу потрібно приділяти саме адаптивній розмітці, яка буде перебудовувати відповідні блоки, зменшувати розміри відповідних елементів, перетворювати стандартне меню у мобільну навігацію та гармонійно взаємодіяти з будь-якими розширеннями екранів.

.

1.4 Виділення основних закономірностей веб-ресурсів подорожей

Знову ж таки ознайомившись з існуючими аналогами системи, яка розроблялася в рамках даного дипломного проекту, були виділені основні закономірності, які присутні в кожному з інтернет-ресурсів.

По-перше, кожен з них надає стандартну інформацію про цікаві місцини за областями (а деякі просто висвітлюють найбільш відомі місця, не роблячи акценту за областями). Оскільки аналогів подібного показу інформації багато, тому було прийнято рішення, у своєму мобільному-додатку зробити розподіл інформації про різноманітні маршрути за містами та їх прилеглими частинами. Це значно полегшує пошуки для користувача та впливає на зручний дизайн веб-ресурсу.

По-друге, як тільки користувач відкриває будь-який з наведених сервісів, він бачить велику кількість різноманітної реклами, яка не має жодного безпосередньоговідношення саме до того, що він шукає на даному ресурсі. А це в свою чергу, негативно впливає на економічний фактор, оскільки покупець постійно відволікається від основної інформації на яскраві банери, а це як наслідок може призвести до того, що він може просто забутипро те, що насправді хотів переглянути, або ж нав’язлива реклама змусить користувача перейти до пошуків іншого сервісу, який може здатися більш зручним для нього. Також згідно з проведеними дослідженнями, саме щодо зручності користування веб-ресурсами, більшість респондентів наголосили на тому, що наявність яскравих рекламних оголошень є великим мінусом для веб-додатку, оскільки значна кількість власників безпосередньо даних ресурсів саме через своє бажання максимально збагатитися, забувають про фактор гармонійного та зручного інтерфейсу користувача. Тож беручи до уваги даний фактор, у перспективі розвитку розробленого проекту та перенесенні його на живий сервер, було прийнято рішення, не розміщувати рекламні банери, задля остаточної концентрації уваги клієнта лише на

інформації розмішеної на даному ресурсі, щоб було зручно й легко його використовувати.

1.5 Виділення основних недоліків веб-ресурсів подорожей

Ще одним аспектом, який варто виділити в усіх проаналізованих ресурсах є перевантаженість сторінки різноманітною непотрібною інформацією, тобто її розміщено занадто багато на одній сторінці. Зі сторони зручності та легкості інтерфейсу для користувача, це визиває дискомфортта дратує велику масу людей, оскільки увагу від пошуку найкращого маршруту відволікає широка кольорова гамма та наявність текстів, написаних маленькими шрифтами, який дуже складно розібрати. Зробивши висновки за допомогою попередніх даних, було прийнято рішення для власного проекту зробити зовсім інше розміщення інформації, а саме створити список областей, які будуть відкриватися лише тоді, коли користувач забажає ознайомитися з даною інформацією більш детально, таким чином він обирає саме те, що хоче, а не те, що йому нав’язують.

Різноманітні інтернет-додатки набувають усе більшої популярності серед молоді. Сучасній людині набагато простіше зайти до відповідного веб-сервісу та обрати саме ту інформацію, яка йому необхідна, натиснувши при цьому лише декілька клавіш, аніж їхати до туристичних агентств та витрачати на вибір мандрівкизанадто велику кількість вільного часу, яке у наш час на рахунок з грошима, через велику його швидкоплинність та шалений ритм нашого життя. Тому такі додатки є оптимальним рішенням для розв’язання даної проблеми, але перед представниками, які розробляють згадані вище сервіси насамперед стоїть основна задача створити потужний веб-ресурс, який буде максимально конкурентоспроможним на економічному ринку, також при цьому залучати нових клієнтів та підтримуватиуже діючих, тому потрібно не забувати про сучасність, модернізацію, зручність та саме головне – простоту інтерфейсу.

1.5 Використання інтелектуальних технологій в туризмі

1.5.1. Концепція штучного інтелекту. Штучний інтелект (ШІ) – це нова технологічна наука, яка вивчає та розробляє теорії, методи, технології та застосування для моделювання, розширення та розширення людського інтелекту.

У інформатиці штучний інтелект (ШІ) — це інтелект, який демонструють машини, на відміну від природного інтелекту, який демонструють люди та інші тварини. Комп'ютерні науки визначають дослідження ШІ як дослідження «інтелектуальних агентів»: будь-якого пристрою, який сприймає навколишнє середовище і виконує дії, які максимізують його шанси на успішне досягнення поставлених цілей [1].

Перш ніж визначати ШІ, що було б цікаво спочатку прояснити, що означає інтелект. Інтелект можна визначити як низку здібностей: здатність розуміти навколишнє середовище та явища, що відбуваються, здатність використовувати переваги минулого досвіду та здатність комбінувати наявні знання, щоб належним чином реагувати на новий виклик [2]. Інтелектуальні системи здатні відчувати навколишнє середовище, навчатися та використовувати те, що було навчене, у майбутніх ситуаціях[3].

Штучний інтелект зазвичай визначають як набір технологій, які можуть імітувати інтелект людини в процесі вирішення проблем [4]. У тому ж ключі, що літаки отримують той самий результат (літають), що й птахи, але використовують зовсім інші механізми, ШІ намагається отримати подібні результати (мислення), що й люди, але за допомогою інших механізмів.

Концепція ШІ з часом еволюціонувала [5], від початкових концептуалізацій, в яких ШІ визначався як такий, що володіє певною формою інтелекту, до останніх визначень і концептуалізацій, в яких ШІ визначається як здатність діяти автономно на великих обсягах даних [6], у майбутнє, де ШІ може перевищити людський інтелект, подію, яку назвали технологічною сингулярністю [7]. У зв'язку з цим можливе явище[8], коли

додаток стає настільки популярним, що він перестає вважатися заШІ. Це пов’язано з тенденцією уявляти, що програма насправді не містить ШІ (насправді не думає), а є лише частиною звичайних обчислень. Таким чином, вміст і межі ШІ динамічні в часі.

Описується до чотирьох типів ШІ [9]. Перший тип – реактивний ШІ, який не має пам’яті чи концепції минулого. Deep Blue — найкращий приклад реактивного AI. Другий тип – це штучний інтелект з обмеженою пам’яттю, який має вибірковий/обмежений виклик. Прикладом є те, як самокерований автомобіль ставиться до об’єктів навколо нього. Третій тип - це теорія розуму. Це машини, які можуть представляти інші типи об’єктів та їхні емоції, що дозволило б машинам соціально взаємодіяти. Четвертий тип - ШІ з самосвідомістю або свідомістю. Поточні проблеми з ШІ полягають у розширенні пам’яті штучного інтелекту, покращенні здатності використовувати минулі спогади та досвід для прийняття кращих рішень, а також розвитку здатності обробляти емоції та інтуїцію. Виходить за межі цих чотирьох типів ШІ концепція суперінтелекту [10]. Суперінтелект визначається як машинний інтелект, який перевершує загальний інтелект людини.

Більшість сучасних систем штучного інтелекту є спеціалізованими. Тобто це системи, здатні вирішувати проблеми, пов’язані з конкретними областями та конкретними завданнями, наприклад, фільтрацію спаму, розуміння запитань, які ставлять люди, візуальну навігацію у відомому середовищі або навіть керування автономним автомобілем. Цей тип ШІ був визначений як «слабкий ШІ» [11]. Майбутні розробки створять ШІ загального призначення або «сильний ШІ», тобто ШІ третього та четвертого типів, згаданих у попередньому абзаці (системи, які мають інтелект у більш ніж одній області, з свідомістю та можливостями думати). Нещодавно з’явилася концепція «гібридного ШІ» [12]. Гібридний ШІ розташований між сильним ШІ та слабким ШІ, включаючи ШІ, який перевищує слабкий ШІ, але без усіх можливостей сильного ШІ. Сектор подорожей і туризму потребує

гібридного штучного інтелекту та сильного штучного інтелекту через широкий спектр завдань та елементів, які необхідно інтегрувати, щоб забезпечити найкращий досвід для туристів.

Для роботи систем штучного інтелекту потрібні чотири основні елементи: дані, програми, апаратне забезпечення та взаємозв’язок між різними системами. Програми штучного інтелекту зазвичай вимагають великих апаратних можливостей (обробка та зберігання) для належної роботи, хоча існують певні архітектури апаратного забезпечення, які краще підходять для ШІ. Що стосується сумісності, то однією з ключових особливостей ШІ є його здатність взаємодіяти між машинами [13], таким чином автоматизуючи агрегацію та консолідацію даних з кількох джерел [14].

Штучний інтелект - це галузь інформатики. Для вчених мета вивчення штучного інтелекту полягає в тому, щоб зрозуміти суть інтелекту та створити розумну машину, яка реагує по-новому, подібно до людського інтелекту. Дослідницькі галузі ШІ включають робототехніку, розпізнавання мови, розпізнавання зображень, обробку природної мови та експертні системи. З розвитком цієї технології все більше цінується застосування штучного інтелекту в реальному житті.

До складу штучного інтелекту в основному входять: пристрій введення, арифметичний блок, пам'ять, контролер і пристрій виведення. Існують лише дві частини ШІ, які пов’язані із зовнішнім світом: пристрій введення та пристрій виведення. Більшість пристроїв введення використовують датчики для імітації органів людини, наприклад, інфрачервоні датчики, датчики, щоб імітувати зорову систему людини, тощо. Цей вид датчика може реалізувати сприйняття положення об'єкта, що контролюється. Якщо об’єкт служби з’являється в діапазоні, який може відчути датчик, кожне «око» дасть різні відповіді в системі. Для пристроїв виводу формами є переважно зображення, звуки, тексти, відео тощо. Варто зазначити, що, на відміну від інших

звичайних роботів, існує особлива форма виводу штучного інтелекту — міміка. Якщо силіконовий матеріал з високим ступенем імітації використовувати для імітації людської шкіри, то вираз обличчя штучного інтелекту буде показувати більш людський стан, тому вихідний пристрій також називають «ефекторним органом».Арифметичний блок в основному обробляє та аналізує дані, отримує дані з пам'яті, а потім здійснює необхідні обчислення та обробку після цих даних, результат обробки повертається в пам'ять. Пам'ять використовується для зберігання даних. Має функцію «пам’яті». Після того, як пристрій введення збирає інформацію із зовнішнього світу, інформація буде перетворена в дані, які може бути розпізнано комп’ютером, а потім збережено. Дані потім надсилаються в місце, де їх потрібно обробити, наприклад в арифметичний блок або контролер. Після обробки всіх даних кінцевий результат надсилається на пристрій виведення [15].

IBM вперше висунула концепцію «розумного туризму» у 2008 році. Після цього традиційна туристична індустрія була швидко інтегрована з інтернетом, інтернетом речей, хмарними обчисленнями та технологіями ШІ. Окрім того, у 2011 році Китай свого часу запропонував змусити китайський туризм здійснити інформатизацію та модернізацію. Крім того, з реалізацією плану дій «Інтернет+», запропонованого 3у 2015 році, застосування Інтернету та технології автоматичного керування дозволило привертали все більшу увагу. Характеристики туризму для передачі інформації та популярність розумних мобільних терміналів, таких як смартфони та планшетні комп’ютери, закладають основу для пропозиції розумного туризму. На основі інформаційно-комунікаційних технологій розумний туризм — це систематична та інтенсивна реформа управління для задоволення персоналізованих потреб туристів, надання високоякісних і задовільних послуг, а також для реалізації спільного та ефективного використання туристичних ресурсів і соціальних ресурсів.

Основними технологіями розумного туризму є хмарні обчислення, Інтернет речей, мобільний термінальний зв’язок та штучний інтелект. Чотири основні технології розумного туризму пов’язані та інтегровані за своєю суттю, що формує загальну технічну структуру розумного туризму, в якій штучний інтелект є важливою частиною базової технологічної основи розумного туризму, а також ядром і ланкою інтелектуального туризму. Штучний інтелект є ключовою технологією для ефективного використання та повного використання багатих туристичних інформаційних ресурсів, а також для інтеграції фрагментованої та незалежної прикладної системи туристичної індустрії.

1.5.2. Штучний інтелект та смарт туризм. Індустрія штучного інтелекту швидко розвивається, і багато технологій, таких як розпізнавання мовлення, візуальне розпізнавання, машинний переклад та обробка інформації, знаходяться на провідному рівні у світі. Прориви також були зроблені в області інтелектуальних чіпів і глибокого навчання. Розвиток технологій штучного інтелекту та Інтернет-технологій глибоко змінив наше життя. Це значно підвищить нашу ефективність і підвищить рівень нашого життя в різних сферах. Суперпозиція інтелектуальних фабрик, інтелектуального виробництва, взаємодії людини та комп’ютера, 3D-технологій, машинного розпізнавання, інтернету речей, великих даних, хмарних обчислень та інших технологій поклала початок нової хвилі технологічної революції.

В даний час штучний інтелект широко використовується в промисловому виробництві, інтелектуальному туризмі, лікуванні, освіті та інших аспектах. Розвиток штучного інтелекту приніс можливості та виклики туристичній галузі. Завдяки зрілості та застосуванню Інтернету речей, хмарних обчислень, великих даних, глибокого навчання та інших технологій розумний туризм з’являється в історичний момент. За допомогою технологій штучного інтелекту та продуктів штучного інтелекту туристичні послуги,

туристичний менеджмент, розвиток туризму, туристичний досвід, туристичний маркетинг та інші аспекти загальної модернізації сприяють трансформації та модернізації туризму в напрямку комплексних послуг та персоналізованої настройки. , щоб надати користувачам кращі враження від туризму.

Штучний інтелект особливо актуальний для подорожей і туризму з кількох причин. Туристам необхідно прийняти низку рішень щодо майбутніх поїздок, наприклад, вибір місця призначення, транспорту, проживання та заходів, серед іншого. Ці рішення суттєво вплинуть на задоволеність туристів своєю поїздкою. Однак діапазон напрямків, транспорту, розміщення та доступних на даний момент заходів представляє майже нескінченний набір варіантів, які потребують автоматизації. Туристичні організації та агенти стикаються з подібною проблемою, коли намагаються знайти найкращу відповідність між клієнтами та туристичними пакетами, пристосованими до їхніх потреб. Організації мають майже нескінченну кількість потенційних клієнтів. Таким чином, узгодження попиту з продуктом є надзвичайно складним процесом, який, здається, добре підходить для можливостей ШІ. Потрапивши до місця призначення, туристи повинні орієнтуватися в царстві невідомого, що характеризується різними звичками, мовами, культурними нормами та кухнею, а також багатьма іншими особливостями, які можуть бути їм незнайомими. Знову ж таки, ШІ може допомогти туристам у таких «дивних» середовищах, наприклад, рекомендуючи маршрут подорожі або допомагаючи з мовними та культурними бар’єрами. Крім того, ШІ може допомогти організаціям персоналізувати досвід, щоб адаптувати їх до побажань туристів.

У той час як туристичний сектор був визнаний першим, хто прийняв більшість інновацій, фактичних випадків використання ШІ залишається мало. Більшість існуючої літератури стосується лабораторних сценаріїв та випадків розробки. В даний час ШІ можна знайти вбудованим у системи

обробки даних у реальному середовищі та на стадії виробництва різних установок, наприклад, систем прогнозування, роботів, розмовних систем та систем розпізнавання голосу. Проте найближчим часом ШІ, швидше за все, стане задіяним у всіх сферах індустрії подорожей та туризму.

1.5.3 Системи ШІ та їх використання в туризмі. Системи штучного інтелекту мають кілька застосувань у туризмі. З точки зору споживача, ШІ допомагає користувачам знаходити кращу та більш релевантну інформацію, надає їм більшу мобільність, покращує прийняття рішень і, зрештою, забезпечує кращий досвід туризму [16]. З точки зору бізнесу, ШІ можна використовувати майже в кожному аспекті управління, особливо в просуванні та продуктивності. Очікується, що штучний інтелект також заохочуватиме більш стійкі подорожі, впливаючи на клієнтів, щоб вони мали більш соціальну перспективу.

Системи штучного інтелекту в індустрії туризму можуть бути окремими системами або вбудованими в існуючі програми та системи. Ці системи включають системи рекомендацій, системи та методи персоналізації, розмовні системи (чат-боти та голосові помічники), інструменти прогнозування, автономні агенти, програми мовного перекладу та розумні туристичні напрямки. Хоча ми аналізуємо кожну систему окремо, слід зазначити, що туристи зазвичай взаємодіють з технологіями, які об’єднують декілька з цих систем. Наприклад, гість може взаємодіяти з роботом, який інтегрує розмовну систему, і, залежно від вимог, систему рекомендацій, техніку персоналізації або автономного агента. Діалог з користувачем може здійснюватися за допомогою чат-бота або голосового помічника.

РОЗДІЛ 2
ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЄКТУ

2.1 Мобільні додатки: нативні, веб та гібридні

Компанії, які думають про розробку програми (для своїх клієнтів або внутрішнього користування) можуть вибрати один з трьох варіантів мобільних додатків - нативні, веб або гібридні. Давайте проаналізуємо кожен тип, розглянемо переваги і недоліки, і визначимося, який з них буде кращим вибором в тій чи іншій ситуації і найкраще задовольнить потреби при створенні мобільного додатка.

Нативні додатки. Під нативним ми маємо на увазі мобільний додаток, який створюється для певної платформи і безпосередньо встановлюється на пристрій користувача (займаючи певний обсяг пам'яті). Такі програми користувач завантажує через магазин додатків тієї або іншої платформи, такої як PlayStore для Google і AppleAppStore для iOS.

З нативними додатками компанії можуть виготовити додаток відповідно до індивідуальних запитів, щоб потім користувачеві було зручно ним користуватися, на додаток до веб-сайту або іншому каналу, яким він вже звик користуватися. Ця цілісність і є суттєвою перевагою нативних додатків.

Деякі інші важливі переваги нативних додатків:

1. Позначення геолокації дозволяє компаніям підлаштовувати свої програми лояльності або промоакції. Споживачі можуть отримувати повідомлення, коли вони знаходяться біля фізичних магазинів, або мають можливість отримати регіональну знижку.

2. Дані дій (чи бездіяльності) користувача можуть бути легко зібрані та проаналізовані, таким чином полегшуючи оцінку ефективності всієї програми або її окремих функцій.

3. Нативні додатки, як правило, працюють і «відчуваються» краще. Веб-додатки іноді створюються для імітації нативних, але вони обмежуються швидкістю інтернету і можливостями дизайну.

І можливі недоліки:

1. Нативні додатки часто дорожчі в розробці, особливо для компаній, яким потрібні додатки на кросплатформенних ОС.

2. Нативні додатки повинні бути схвалені кожним магазином додатків, а процес залучення уваги до нього користувачів може бути складним (якщо це не додаток для внутрішнього користування в компанії).

Веб-додатки. Зрозуміло, що ці програми працюють через веб-браузер на пристрої користувача. Ці додатки по суті є індивідуалізованими веб-сайтами, які зроблені таким чином, щоб виглядати і використовуватися як нативні додатки, але насправді вони не знаходяться на пристрої користувача. Їх можна порівняти з хмарним сховищем в порівнянні з даними, які зберігаються на жорсткому диску комп'ютера. При гарної, якісної розробки, яка включає в себе підбір розмірів і прокрутку, веб-додатки часто працюють подібно нативним додатків.

Ось деякі ключові переваги веб-додатків:

1. Додатки на веб-основі легше підтримуються і вони можуть функціонувати на платформі з будь-якої ОС.

2. Розробники можуть пропонувати додатки без необхідності їх затвердження будь-якими магазинами додатків.

3. Більш швидка розробка циклів з використанням CSS, HTML і JavaScript.

І кілька мінусів:

1. У веб-додатків немає доступу до пристрою користувача. Незважаючи на те, що іноді було б зручно, це обмежує багато функцій, які використовуються в нативних додатках для більш персоналізованого використання.

2. Користувачі повинні використовувати їх через мережу, що значно знижує контроль безпеки.

3. Пошук програми може бути складним, тому що не існує магазину додатків з каталогом і функцією пошуку в ньому.

Гібридні додатки. Гібридні додатки є чимось середнім між нативними і веб-додатками. Фактично вони створюються так, щоб виглядати і використовуватися як нативні додатки. Їх також встановлюють на телефон користувача і їх можна знайти в магазинах додатків. Різниця полягає в тому, що вони обов'язково повинні розміщуватися в рамках нативного програми та створені, щоб працювати через WebView, і таким чином вони можуть отримувати доступ до інформації на пристрої користувача для великих можливостей.

Додаткові переваги гібридних додатків:

1. Гібридні додатки мають найбільшу функціональність і персоналізацію для користувача;

2. Розробники не обмежені однією платформою, замість цього вони можуть створити гібридний додаток, який буде працювати з декількома платформами (в разі роботи як нативний додаток);

3. Гібриди – хороша опція для розробників, які створюють візуально насичені програми, наприклад, ігри (які не будуть добре працювати в вигляді веб-додатків).

У будь-якому випадку, є деякі недоліки, про які варто подумати при виборі гібридної програми:

1. Занадто складні додатки найкраще робити нативними;

2. Розробка вимагає додаткових часу і зусиль (в порівнянні з веб-додатками), щоб такий додаток виглядав і відчувався користувачем як нативний;

3. Магазини додатків можуть відхиляти гібридні програми, які працюють недостатньо плавно.

Вибір відповідної моделі мобільного додатка - це дуже важливий етап в його розробці, на який впливають кілька факторів, таких як технічна оцінка розробників; потреба в доступі до інформації на пристрої; вплив швидкості інтернету на додаток; одно- або багатоплатформений додаток[1].

Проаналізувавши всю попередню інформацію, було обрано для даної дипломної роботи обрати гібридний додаток, адже він найкраще підходить для створення даного додатку подорожей.

2.2 AndroidStudio

AndroidStudio– інтегроване середовище розробки (IDE) для платформи Android, представлене 16 травня 2013 року на конференції Google I/O менеджером по продукції корпорації Google– Еллі Паверс (ElliePowers). 8 грудня 2014 року компанія Google випустила перший стабільний реліз AndroidStudio 1.0 [2].

AndroidStudio прийшло на зміну плагіну ADT для платформи Eclipse. Середовище побудоване на базі програмних кодів продукту IntelliJ IDEA CommunityEdition Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки додатків для платформи Android. У тому числі у середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування додатків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо).

Крім можливостей, присутніх в IntelliJIDEA, в AndroidStudio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання програм заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції. Для прискорення розробки програм представлена колекція типових елементів інтерфейсу і візуальний редактор для їхнього

компонування, що надає зручний попередній перегляд різних станів інтерфейсу додатку (наприклад, можна подивитися як інтерфейс буде виглядати для різних версій Android і для різних розмірів екрану).

Для створення нестандартних інтерфейсів присутній майстер створення власних елементів оформлення, що підтримує використання шаблонів. У середовище вбудовані функції завантаження типових прикладів коду з GitHub. До складу також включені пристосовані під особливості платформи Android розширені інструменти рефакторингу, перевірки сумісності з минулими випусками, виявлення проблем з продуктивністю, моніторингу споживання пам'яті та оцінки зручності використання. У редактор доданий режим швидкого внесення правок. Система підсвічування, статичного аналізу та виявлення помилок розширена підтримкою Android API. Інтегрована підтримка оптимізатора коду ProGuard. Вбудовані засоби генерації цифрових підписів. Надано інтерфейс для управління перекладами на інші мови.

Деякі особливості будуть пізніше розгорнуті для користувачів так як програмне забезпечення розвивається; наразі, передбачені такі функції: Особливості:

1. Живі макети (layout): редактор WYSIWYG — живе кодування — подання (rendering) програми в реальному часі.
2. Консоль розробника: підказки по оптимізації, допомога по перекладу, стеження за напрямком, агітації та акції — метрики Google аналітики.
3. Резерви бета релізів та покрокові релізи.
4. Базування на Gradle.
5. Android-орієнтований рефакторинг та швидкі виправлення.
6. Lint утиліти для охоплення продуктивності, юзабіліті, сумісності версій та інших проблем.
7. Використання можливостей ProGuard та підписів до програм.

8. Шаблони для створення поширених Android дизайнів та компонентів.

9. Багатий редактор макетів (layouts) що дозволяє користувачам перетягнути і покласти (drag-and-drop) компоненти користувацького інтерфейсу, як варіант, переглянути одночасно макети (layouts) на різних конфігураціях екранів[3].

2.3 Вибір мов і засобів програмування

Існує кілька мов програмування розроблених спеціально для створення веб сторінок та сайтів, зокрема PHP, Perl, Python, Ruby, ASP.NET, Java, Groovy. У свою чергу вони поділяються на клієнтські та серверні.

2.3.1 Вибір клієнтської мови програмування.Технології веб-програмування на стороні клієнта включають в себе набір різних засобів і мов програмування. Перш за все це JavaScript, підтримка якого закладена практично в будь-якому браузер. JavaScript використовується частіше, ніж будь-які інші мови для написання скриптів, що працюють на стороні клієнта. Він є простим, його код легко інтегрувати в код HTML-сторінки, в той же час він надає достатньо багато можливостей.

При виборі клієнтської мови доцільно використовувати HTML оскільки вона підтримується усіма відомими браузерами і працює майже безвідмовно. HTML (Мова розмітки гіпертекстових документів) – стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML обробляється браузером та відтворюється на екрані у зручному для людини вигляді.

JavaScript являє собою мову написання сценаріїв на стороні клієнта, яка вносить на Web-сторінки елементи інтерактивності і умовної поведінки. За допомогою JavaScript можна виводити додаткову інформацію про посилання, створювати інтерактивні ефекти при роботі з мишею, змінювати за певних

умов вміст сторінок, випадковим чином відображати вміст сторінки, завантажувати вміст в нові вікна браузера і фреймів.

Сценарії JavaScript зазвичай поміщають безпосередньо в документ HTML. Вони можуть перебувати або в заголовку або в тілі документа. В один документ можна помістити кілька сценаріїв [4] .

2.3.2 Вибір серверної мови програмування. Раціонально буде використовувати JavaScript, а от обґрунтуватице можна тим, що оскільки вона використовується і в клієнтській мові програмування, то для fullstack додатку одному розробнику набагато легше оперувати в схожих середовищах. Веб-сервер було обрано Node.js.

У 2009-му платформа Node.js зробила свої скромні перші кроки в безкрайньому світі розробки бекенд. Це була перша спроба використання JavaScript в серверних додатках. Сьогодні буде вкрай важко знайти веб-розробника, який би не чув про Node. Але не можна сказати, що існування Node було безхмарним. Ця платформа пережила розкол спільноти, була предметом форумних війн і багатьох довела до відчаю.

Перш ніж говорити про те, як виглядає сьогодні серверна платформа Node, згадаємо про те, що це таке. А саме, це середовище виконання JavaScript, побудована на базі JS-движка V8, розробленого Google і застосовуваного в GoogleChrome. Node.js використовує неблокуючих моделей введення-виведення, керовану подіями, яка робить цю платформу простою і ефективною.

На початку цього матеріалу Node показаний як прямо-таки кошмар програміста. Однак, ця платформа не випадково стала дуже популярною. Тут ми не станемо спиратися на голослівні твердження. Краще поглянемо на факти. А саме, свіже дослідження StackOverflow показує, що Node.js - це, на сьогоднішній момент, найпопулярніша серед розробників технологія.

Крім того, JS - це мова, популярність якої за останні п'ять років зростає швидше, ніж у інших мов, при тому, що C # і PHP втрачають позиції.

Поширеність JavaScript, якщо навіть не говорити виключно про Node, йде вгору.

Як можна пояснити те, що JavaScript, в ролі серверного мови, стала настільки швидко і широко прийнятою спільнотою розробників? Простіше кажучи, Node пережив стадію, в якій сприймався як якась забава, і увійшов у фазу стабільності і зрілості. Навколо нього сформувалося потужна спільнота, розмір якої неухильно росте. Екосистема Node також гідна згадки, так як, наприклад, менеджер пакетів Node, npm, зараз представлений найбільшим реєстром ПО в інтернеті.

Node.js не тільки зробив революцію в серверній розробці, але завдяки йому зроблений внесок і в продуктивність клієнтських додатків, так як до розвитку V8 були залучені серйозні сили. Крім того, він грає помітну роль в розширенні всієї екосистеми JavaScript і в удосконаленні сучасних JS-фреймворків, таких, як Angular, ReactабоVue[5].

2.4 Вибір фреймворка

Фреймворк – програмна платформа, яка визначає структуру програмної системи, програмне забезпечення, що полегшує розробку і об’єднання різних компонентів великого програмного проекту. Вживається також слово «каркас», а деякі автори використовують його в якості основного, в тому числі не базуючись взагалі на англомовному аналогу. Можна також говорити про каркасний підхід як про підхід до побудови програм, де будь-яка конфігурація програми будується з двох частин: перша, постійна частина – каркас, незмінний від конфігурації до конфігурації і несе в собі гнізда, в яких розміщується друга, змінна частина – змінні модулі (або точки розширення).

Для даної дипломної роботи було обрано фреймворк Vue.js.

Отже, розберемося з Vue і яким чином він може полегшити нам розробку програми.

Творцем Vue.js є EvanYou, колишній співробітникGoogle і MeteorDevGroup. Почав він розробляти фреймворк в 2013-му, а в лютому 2014 го відбувся перший публічний реліз. Vue широко використовується серед китайських компаній, наприклад: Alibaba, Baidu, Xiaomi, SinaWeibo і ін. Він входить в ядро Laravel і PageKit.

В кінці вересня 2016- го вийшов в реліз Vue.js 2.0, ще крутіше і з упором на продуктивність - тепер використовується віртуальний DOM, підтримується серверний рендеринг, можливість використовувати JSX і т.д. Хоча зараз він підтримується тільки співтовариством, він тримається гідно навіть на рівні продуктів таких гігантів, як Google і Facebook (Angular2 і React 15), і поступово наздоганяє їх по популярності.

Чому саме фреймворк Vue.js, тому що з ним дуже просто почати працювати, навіть якщо ви ніколи не працювали з JavaScript фреймворками. Це ідеальне поєднання зручності і потужності. Розглянемо ще кілька аргументів на його користь:

1. Тепер він ще менше. Runtime збірка Vue.js 2.0 важить всього лише 16kb, а разом з vue-router і vuex - 26kb (приблизно як і ядро першої версії).

2. Він ще швидше. У Vue.js завжди приділялася велика увага швидкодії. Прошарок, що відповідає за рендеринг, переписана на полегшену реалізацію віртуального DOM-а - Форк Snabbdom.

3. Vue 2.0 підтримує серверний рендеринг. Для цього є модуль Vue-server-renderer, що підтримує інші інструменти з екосистеми Vue (vue-router і vuex). Тепер набагато простіше створювати ізоморфні додатки.

Vue пройшов шлях від невеликого аматорського проекту до широко використовуваного проекту з великим співтовариством, тому здається зрозуміли вибір саме даного фреймворку[6].

2.5 Вибір системи управління базами даних

База даних – впорядкований набір логічно взаємопов’язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система керування БД. Головне завдання БД – гарантоване збереження значних обсягів інформації (так звані записи даних) та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином, БД складається з двох частин: збереженої інформації та системи керування нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних).

Для веб-розробки доцільно використовувати MongoDB оскільки це СУБД з відкритим кодом.

MongoDB це крос платформна, документозорієнтована база даних, яка забезпечуєвисоку продуктивністьта легку масштабованість. В основі даної БД лежить концепція колекцій і документів.

База даних. База даних представлена у вигляді фізичного сховища колекцій. Кожна БД має свій власний набір файлів в файлову систему. Зазвичай, один MongoDB сервер має кілька БД.

Колекція. Колекція - це група документів MongoDB. Є еквівалентом простої таблиці в реляційної базі даних. Колекція поміщена всередині однієї БД. Документ в колекції миє мати різні поля. Найчастіше, всі документи в колекції створені для однієї, або відносяться один до одного цілей.

Документ. Документ – це набір пар "ключ - значення". Документ має динамічну схему. Це означає, що документ в одній і тій же колекції не зобов'язаний мати один однаковий набір полів або структуру, а загальні поля в колекції можуть мати різні типи даних[7].

2.6 Вибір середовища програмування

Середовище розробки – це комп’ютерна програма, що допомагає програмістові розробляти нове програмне забезпечення чи модифікувати (удосконалювати) вже існуюче.

Інтегровані середовища розробки зазвичай складаються з редактора сирцевого коду, компілятора чи/або інтерпретатора та засобів автоматизації збірки. Іноді сюди також входять системи контролю версій, засоби для профілювання, а також різноманітні засоби та утиліти для спрощення розробки графічного інтерфейсу користувача. Багато сучасних інтегрованих середовищ розробки також включають оглядач класів, інспектор об’єктів та діаграм ієрархії класів для використання об'єктно-орієнтованого підходу у розробці програмного забезпечення. Сучасні ICP часто підтримують розробку на декількох мовах програмування.

Для зручної розробки веб-проекту знадобляться такі засоби програмування:

1. Локальний сервер.
2. Візуальне середовище програмування.
3. Програма керування БД[8].

2.6.1 Вибір сервера. Для даної роботи було обрано нелокальний хостинг Heroku – хмарна PaaS-платформа, що підтримує ряд мов програмування. Heroku, одна з перших хмарних платформ, з'явилася в червні 2007 року і спочатку підтримувала тільки мову програмування Ruby, але на даний момент список підтримуваних мов також включає в себе Java, Node.js, Scala, Clojure, Python, Go, Ruby і PHP. На серверах Heroku використовуються операційні системи Debian або Ubuntu.

Програми, що працюють на Heroku, використовують також DNS-сервер Heroku. Для кожної програми виділяється кілька незалежних віртуальних процесів, які називаються «dynos». Вони розподілені по спеціальній

віртуальній сітці («dynos grid»), яка складається з декількох серверів. Heroku також має систему контролю версій Git[9] .

2.6.2 Вибір візуального середовища програмування.Візуальне середовище програмування – це інтегроване середовище розробки програмних засобів, яке містить редактор вихідного коду, компілятор чи/або інтерпретатор, засоби автоматизації збірки та засоби для спрощення розробки графічного інтерфейсу користувача. Середовища для візуального програмування також надають змогу конструювати програми шляхом оперування графічними об’єктами. Багато сучасних візуальних середовищ програмування використовуються для реалізації принципів об’єктно-орієнтованого підходу у розробці програмного забезпечення.

Візуальним середовищем програмування було обрано Visual Studio Code (рис. 2.1). Позиціонується як «легкий» редактор коду для кросплатформної розробки веб-і хмарних додатків.

VS Code дозволяє розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ.

У редакторі присутні вбудований відладчик, інструменти для роботи з Git і засоби рефакторинга, навігації по коду, автодоповнення типових конструкцій і контекстної підказки.

Продукт підтримує розробку для платформ ASP.NET і Node.js, і вважається легковажним рішення, яке дозволяє обійтися без повної інтегрованого середовища розробки.Великим плюсом редактора є підтримка великої кількості мов, таких як C ++, C #, Python, PHP, JavaScript та інших [10].

2.6.3 Вибір програми управління БД. MongoDBCompass(рис. 2.2),MongoDB – це потужна, гнучка і розширювана база даних загального призначення. Mongo поєднує в собі вторинні індекси, запити з діапазонами і сортуванням, агрегацією і геопозиційні запити.

MongoDB Compass - це графічний інтерфейс для MongoDB. Компас дозволяє аналізувати та розуміти вміст ваших даних без офіційного знання синтаксису запитів MongoDB. Окрім вивчення даних у візуальному середовищі, ви також можете використовувати компас для оптимізації продуктивності запитів, управління індексами та здійснення перевірки документів [11].

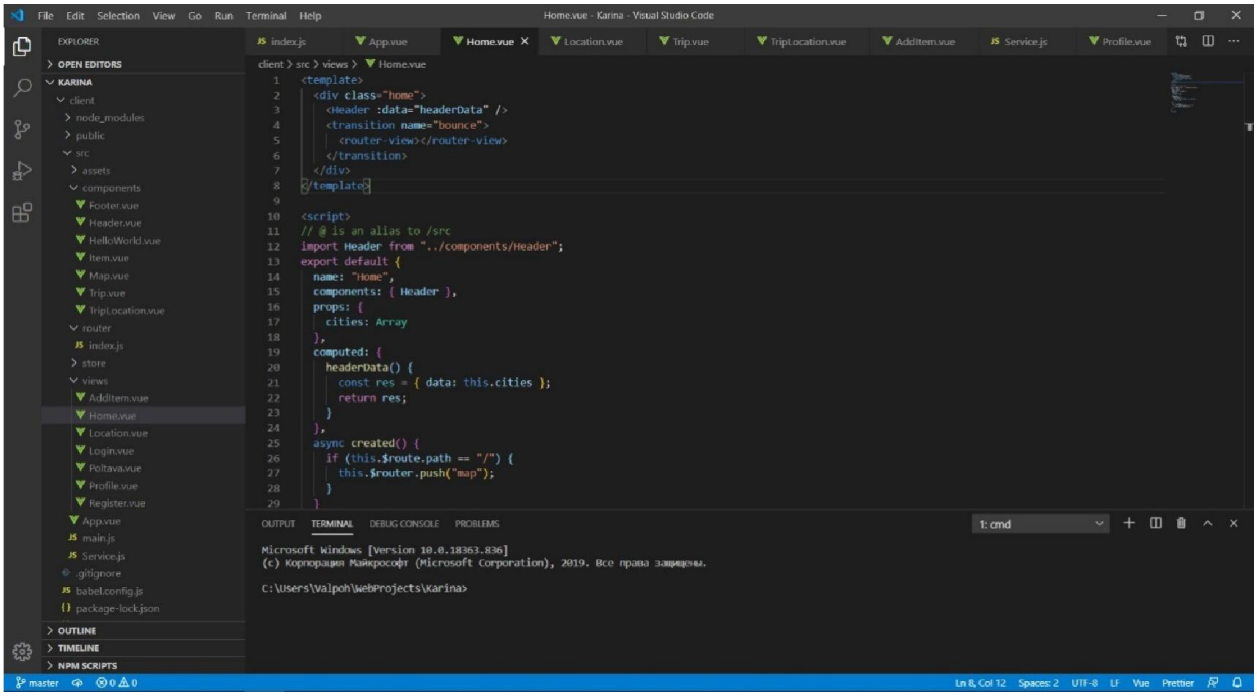

Рисунок 2.1 – Візуальне середовище програмуванняVisual Studio Code.

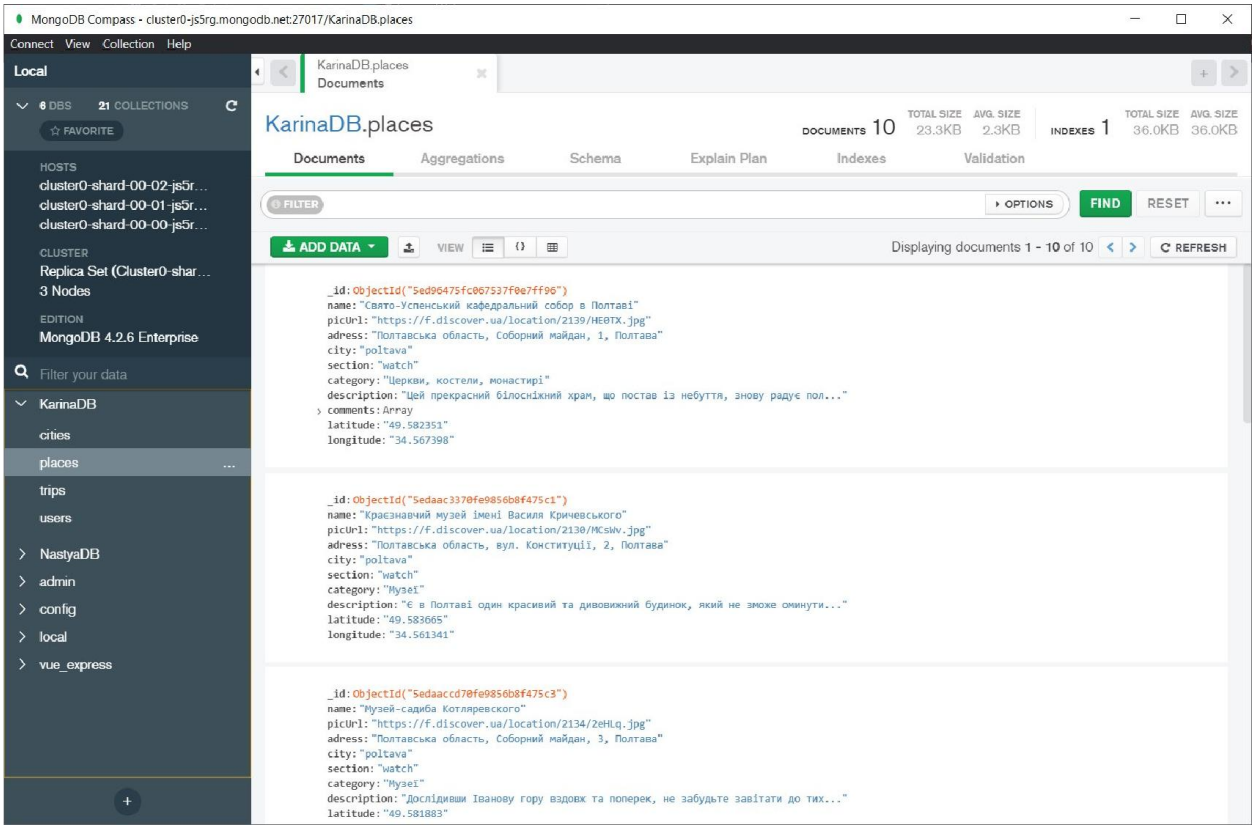

Рисунок 2.2 – Програма керування БД MongoDBCompass

РОЗДІЛ 3
ПРОЕКТНІ РІШЕННЯ

3.1 Розробка структури веб-ресурсу

Створення сайту починається з розробки його логічної структури. Структура сайту – це організація даних, ієрархія матеріалів, необхідна для представлення їх інтернет-аудиторії. Процес створення структури сайту можна розділити на два етапи:

1. Структуризація інформації.
2. Візуальне представлення структури.

Структуризація інформації полягає в тому необхідності класифікації і групування розрізнених матеріалів, об’єднанні їх в категорії або рубрики, присвоюванні їм зрозумілих для користувачів назв. При цьому слід враховувати логіку більшості користувачів. Візуальне представлення структури сайту дозволяє розмістити елементи структури відносно один одного таким чином, щоб користувач першого погляду міг зорієнтуватись на сайті і зрозуміти, де знайти потрібну йому інформацію. При цьому слід враховувати правило Міллера, згідно якого кількість категорій сайту повинно дорівнювати числу 7+2. Це засновано на властивості короткочасної пам’яті людини: одночасно людина може тримати в голові 5-9 слів (назви категорій сайту) [12]. Крім цього слід зробити уніфікований інтерфейс для кожної сторінки, бо, якщо одна сторінка буде відрізнятись від іншої, у користувача може скластись враження, що він перейшов на інший сайт, і він покине його.

Після структуризації даних про предметну галузь при великій кількості інформації як текстової, так і графічної, її слід розбити на окремі сторінки. В залежності від способу зв’язування сторінок сайт може мати таку внутрішню структуру:

1. Лінійна – якщо матеріал вибудовується у логічний ланцюжок. При цьому перегляд сайту починається з першої сторінки, далі почергово проглядаються інші сторінки.

2. Деревовидна – вміст кожної сторінки крім першої входить у вигляді підрозділу в сторінку більш високого рівня.

3. Гратчаста – заснована на побудові системи навігації сайту, коли присутній зв’язок між вертикальними і горизонтальними елементами сайту (сторінками), тобто є можливість швидкого переходу від однієї до іншої сторінки без відвідування проміжних сторінок.

4. Гібридна – являє собою комбінацію перших двох структур. Оскільки користувачами веб-ресурсу частіше за все можуть бути молоді люди структура має бути простою і універсальною.

Після розробки внутрішньої структури сайту, необхідно визначити місце розташування основних структурних елементів:

1. У верхнійчастині сторінки (шапка, хедер, header) містяться: логотип, заголовок, салоган, вибір мовної версії, меню навігації (по сайту).

2. У середній (основній) частині сторінки: меню навігації по розділах, основна інформація, зображення, банери.

3. У нижній частині сторінки (футер, footer): копірайти, адреси, телефони, лічильники і банери. Враховуючи потреби користувачів було розроблено наступну зовнішню структуру веб-ресурсу(рис 3.1).

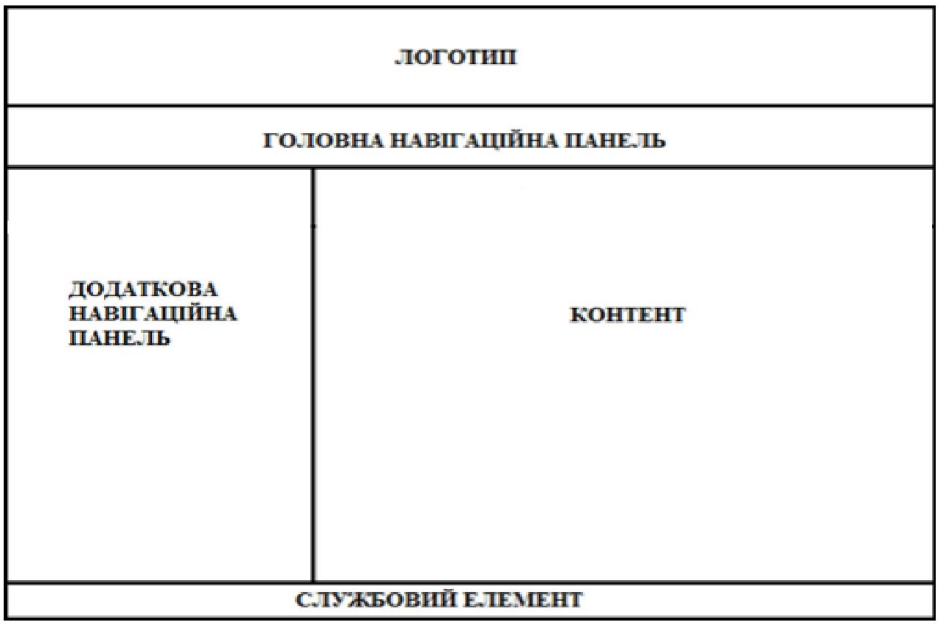

Рисунок 3.1 – Зовнішня структура веб-ресурсу

3.2Розробка бази даних

База даних – впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система керування БД. Головне завдання БД – гарантоване збереження значних обсягів інформації (так звані записи даних) та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином, БД складається з двох частин: збереженої інформації та системи керування нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних).

Для веб-розробки доцільно використовувати MongoDB оскільки це СУБД з відкритим кодом.

MongoDB – це кросплатформна, документозорієнтована база даних, яка забезпечує високу продуктивність та легку масштабованість. В основі даної БД лежить концепція колекцій і документів.

3.3 Розробка модуля авторизації та реєстрації.

Авторизація це: надання повноважень на виконання певних дій в системі обробки даних на віддаленому сервері. Для даної роботи було обрано – JSON Web Token (JWT) - це відкритий стандарт (RFC 7519), який визначає компактний і автономний спосіб безпечної передачі інформації між сторонами як об'єкт JSON. Цю інформацію можна перевірити і довірити, оскільки вона підписана цифровим шляхом. JWT можуть бути підписані за допомогою секретного (за допомогою алгоритму HMAC) або пари відкритого або приватного ключів за допомогою RSA або ECDSA[13].

JWT. JSON Web Token працює схоже зі звичною реалізацією. Але JWT має деякі переваги - він самодостатній, всі необхідні для аутентифікації дані можна зберігати в самому токені.

Структура JWT складається з трьох основних частин: заголовка (header), навантаження (payload) і підписи (signature). Тема і навантаження формуються окремо, а потім на їх основі обчислюється підпис.

Авторизація – це найпоширеніший сценарій використання JWT. Після входу користувача кожен наступний запит буде включати JWT, дозволяючи користувачеві отримувати доступ до маршрутів, служб та ресурсів, дозволених за допомогою цього маркера. Single Sign On - це функція, яка сьогодні широко використовується JWT, через її невеликі накладні витрати та її можливість легко використовуватись у різних областях[14].

3.4 Розробка веб-додатку подорожей

У теоретичній частині були представлені технології реалізації веб-додатків, основною метою використання яких являється зручність, можливості, що надалі продовжуватимуть свій розвиток, а головне економія

часу та ресурсів. Практична частина присвячена на створенні додатку подорожей Україною.

Даний веб-ресурс має повноцінні сторінки, кожна з яких має дизайн та розмітку для пристроїв з усіма можливими розширеннями екранів:профіль, логін, реєстрація, головна сторінка, сторінка з вибором області, сторінка з місцевостями, сторінка з складання маршруту.

Особливу увагу потрібно приділяти головній сторінці сайту, оскільки саме вона знайомить користувача з нашим додатком (рис.3.2).

Рисунок 3.2 – Демонстрація головного екрану додатку

Щоб надалі працювати з додатком максимально зручно, необхідно зареєструвати аккаунт в нашій системі веб-ресурсу і заповнити профіль, саме після цього ми зможемо використовувати всі його функції і проглянути наші дані в меню мобільного додатку (рис. 3.3).

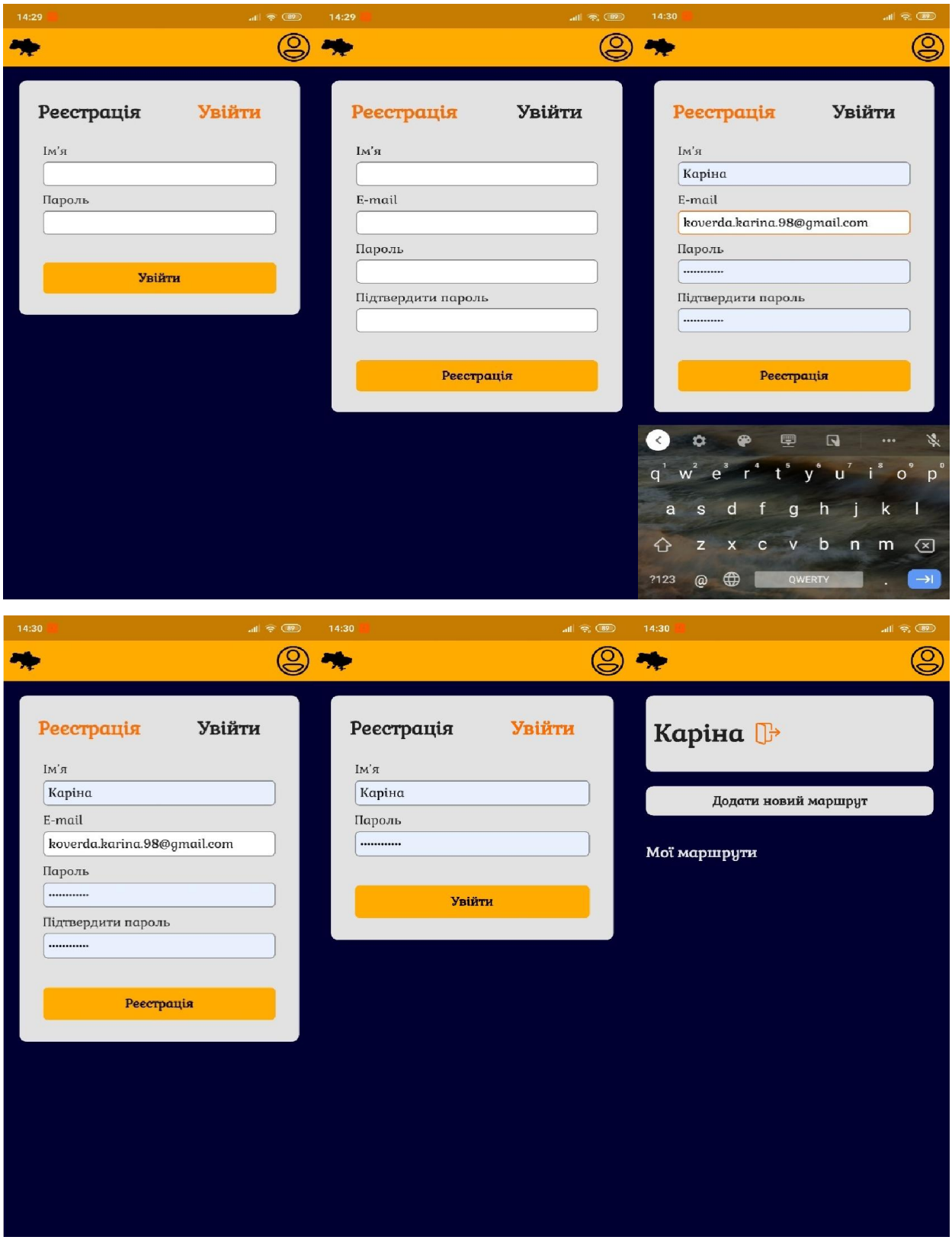

Рисунок 3.3 – Вікно логіну користувача та головне вікно

При натисненні кнопки – «Мої маршрути» – перед вами відкриється список вже створених попередньо маршрутів користувача. Для того щоб додати новий – необхідно обрати «Додати новий маршрут»знизу зліва

У панелі додавання нового маршруту користувачу необхідно ввести назву міста, куди б він хотів відправитися. Після створення він буде відображатися у загальному списку (рис3.4).

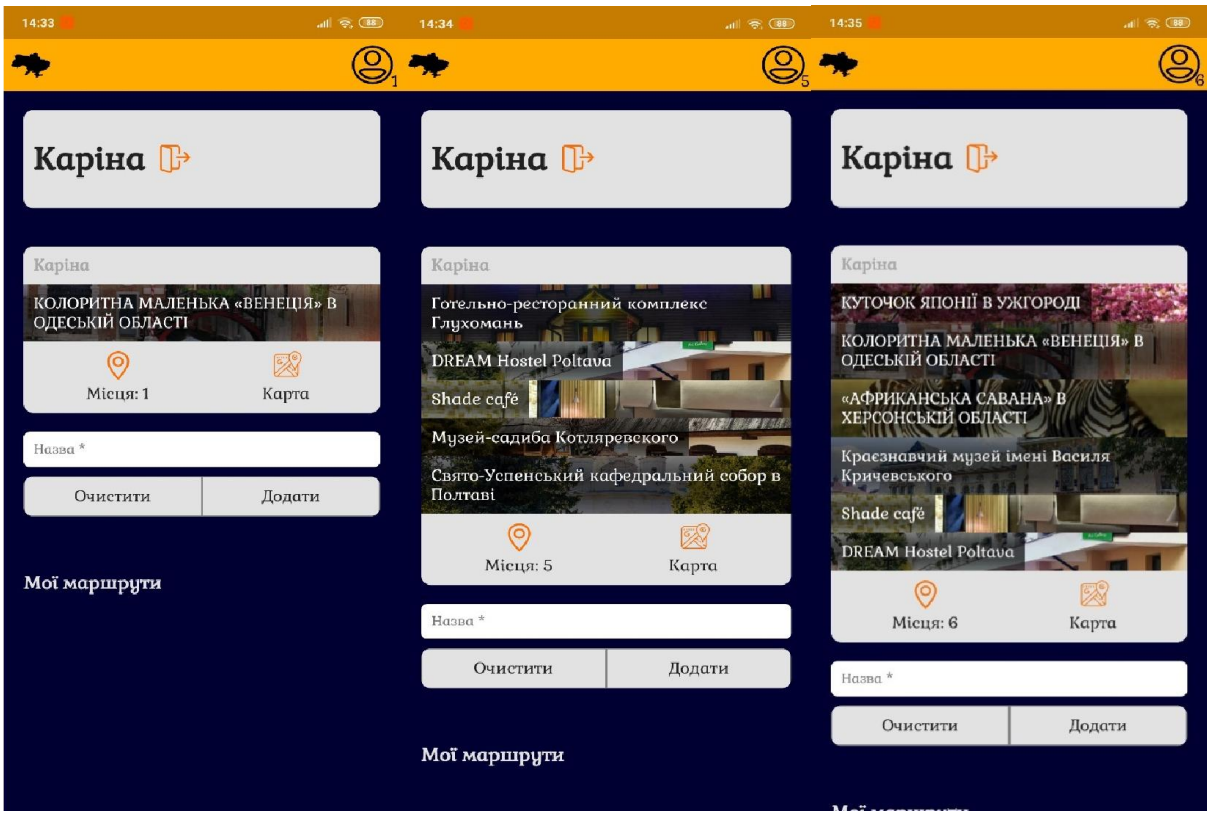

Рисунок 3.4 – Відображення доданих маршрутів

Користувачеві буде одразу складно зорієнтуватися з приводу маршруту (звісно якщо він попередньо не визначився з містом). І саме для того, щоб ознайомитися з містами та їх найвизначнішими місцями, є панель з областями (рис 3.5). Коли місто буде обрано, натиснувши на нього буде доступна інформація стосовно різних категорій, таких як: деможна скуштувати місцеву кухню (категорія їсти), а де залишитися зручно на ночівлю (категорія спати), що відвідати (категорія дивитися), а також загальна інформація стосовно всіг категорій (категорія все) (3.6).

Серед представлених міст України користувач може обрати те місце, яке його найбільше зацікавило та дізнатися повну інформацію по категоріях, які були згадані раніше (рис 3.7).

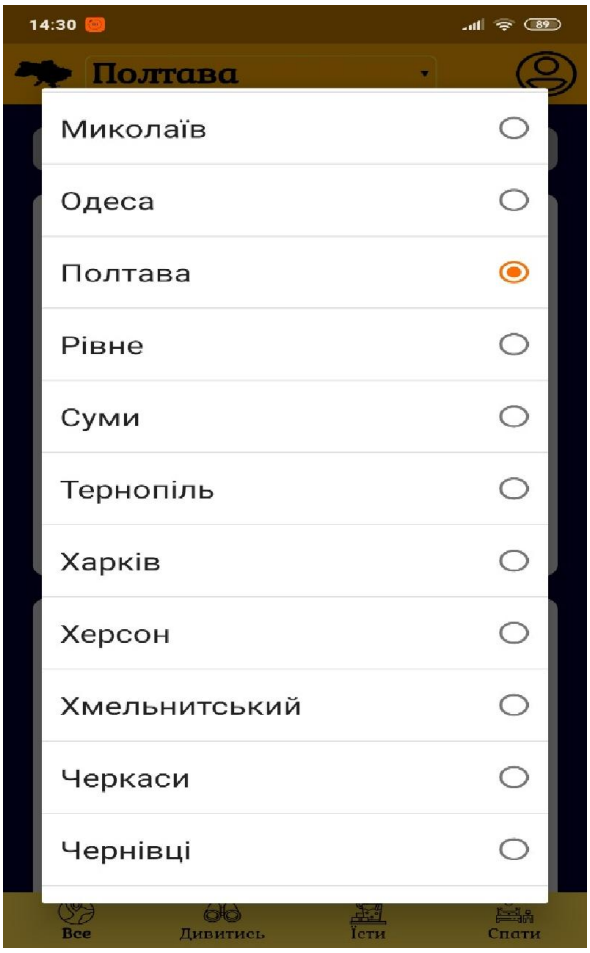

Рисунок 3.5 – Відображення панелі областей

Рисунок 3.6 – Демонстрація категорій

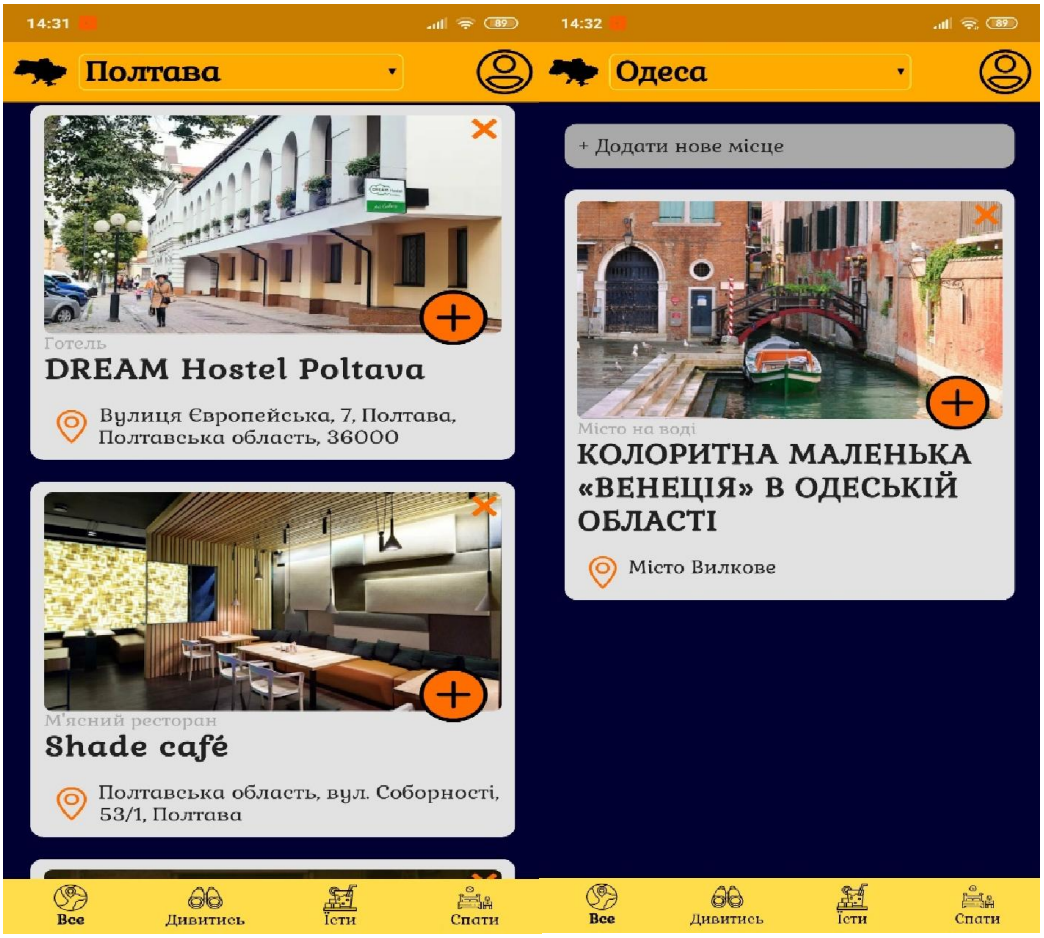

Рисунок 3.7 – Інформація по категоріях

Для максимальної демонстрації функціоналу даного мобільного додатку обрано місто Полтава. На рисунку 3.8, продемонстровано категорію СПАТИ та показанонайрізноманітніші хостели та готелі у даному місті, де можна залишитися на ночівлю, після довгих прогулянок.

Наступною показано інформацію,щоб ознайомитися з категорією ЇСТИ, тобто перегляд тих закладів де можна смачно підкріпитися та спробувати місцеву колоритну кухню у період між виконанням маршрутів (рис 3.9).

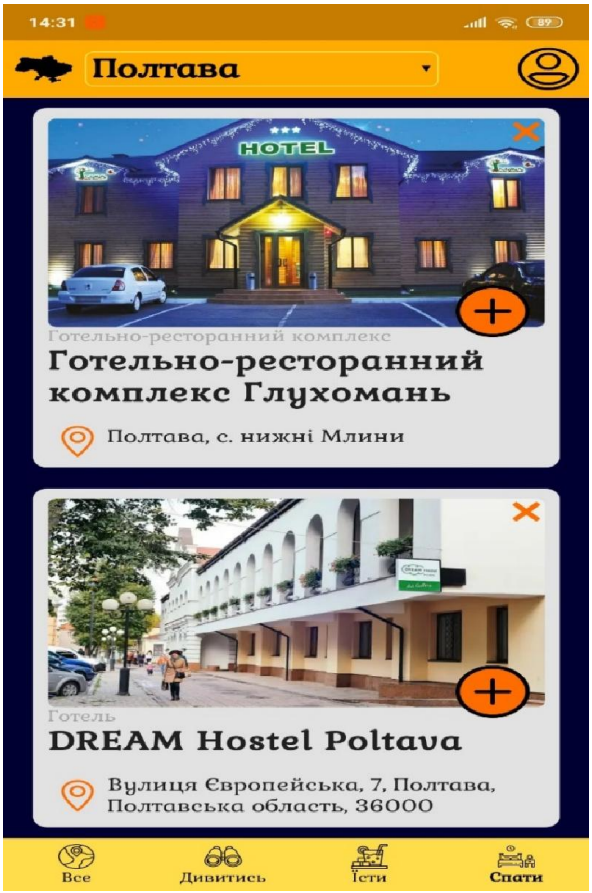

Рисунок 3.8 – Демонстрація категорії Спати

Рисунок 3.9 – Демонстрація категорії Їсти

Даний веб-ресурс зорієнтований на подорожі, тому інформація про найколоритніші місця нашої держави найбільш обширна.

Наше місто Полтава, як відомо, посідає особливе місце в історії української нації та держави. Біля його стін не раз відбувалися події, котрі визначити долю України, впливали на європейські справи.

Засноване слов'янами-сіверянами у IX ст. укріплене першопоселення на Івановій горі поклало початок розвитку давньоруського граду X-XIII ст., поселення XIV, XV віків. Розкопки, проведені в історичному центрі Полтави (Червона площа, вул. Спаська, Першотравневий проспект), виявили ділянки міської забудови, вулиці, житла, господарські і виробничі приміщення давніх полтавців. Ці наукові свідчення стали фундаментом офіційного визнання 1100-літнього віку Полтави[15]. Тому категорія ДИВИТИСЯ розповість користувачеві про ті місцини , які варто обов'язково відвідати(рис3.10).

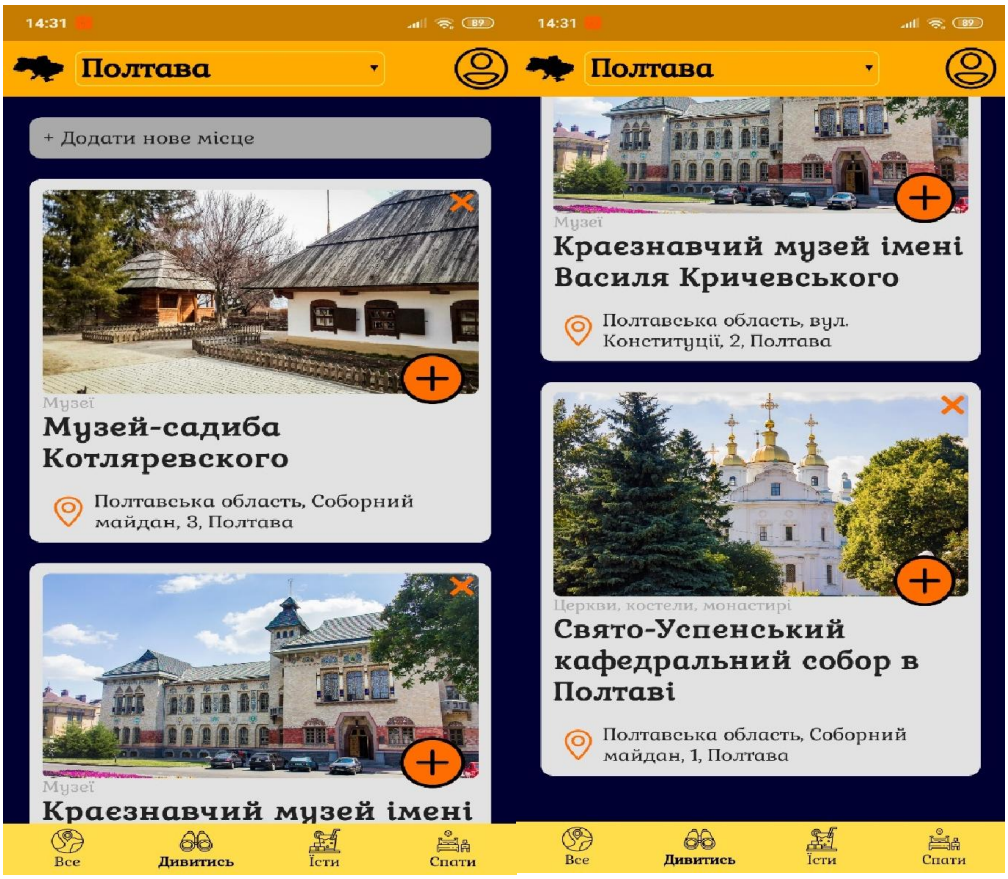

Рисунок 3.10 – Демонстрація категорій Дивитися в місті Полтава

Коли користувач ознайомиться з категоріями та загальною інформацією, то надалі натиснувши на ту місцину, яка максимально зацікавила, він може отримати загальну кількість фактів стосовно неї (рис 3.11).

Також будь-якому відвідувачеві додатку, після реєстрації й ньому,надається можливість залишати свої враження за допомогою коментарів на даному ресурсі (на рисунку 3.12 можна переглянути послідовність залишення коментаря у веб-додатку).

Рисунок 3.11 – Демонстрація розгорнутої інформації місцевості

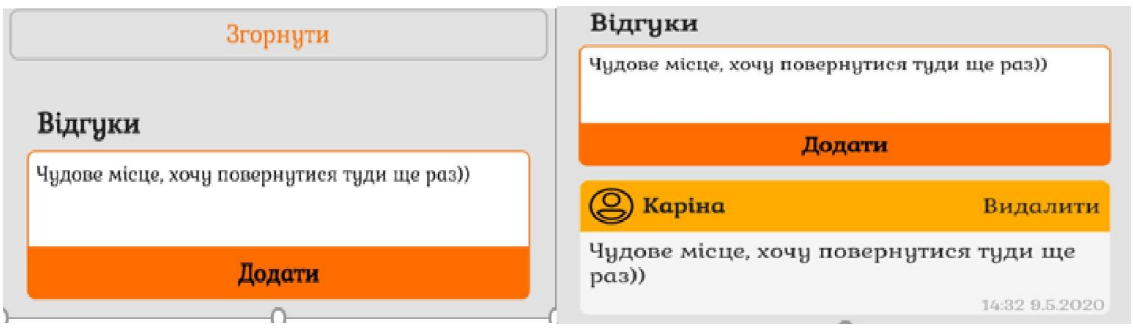

Рисунок 3.12 – Демонстрація залишення відгуків в додатку

Коли користувач повністю ознайомився з інформацією по сайту, з нашими категоріями, відгуками від інших користувачів, то він може обрати власний маршрут для своєї подорожі, навіть декілька, для цього йому необхідно натиснути «+», перейти до акаунту та назвати маршрут і він буде збережений. Якщо користувач обрав не той маршрут, він може його видалити натиснувши «видалити X»(послідовність збереження маршруту указана на рис 3.13).

Коли маршрут буде обрано, то за допомогою карти він може прокласти цей маршрут та дізнатися скільки часу він витратить на цю подорож (рис 3.14).

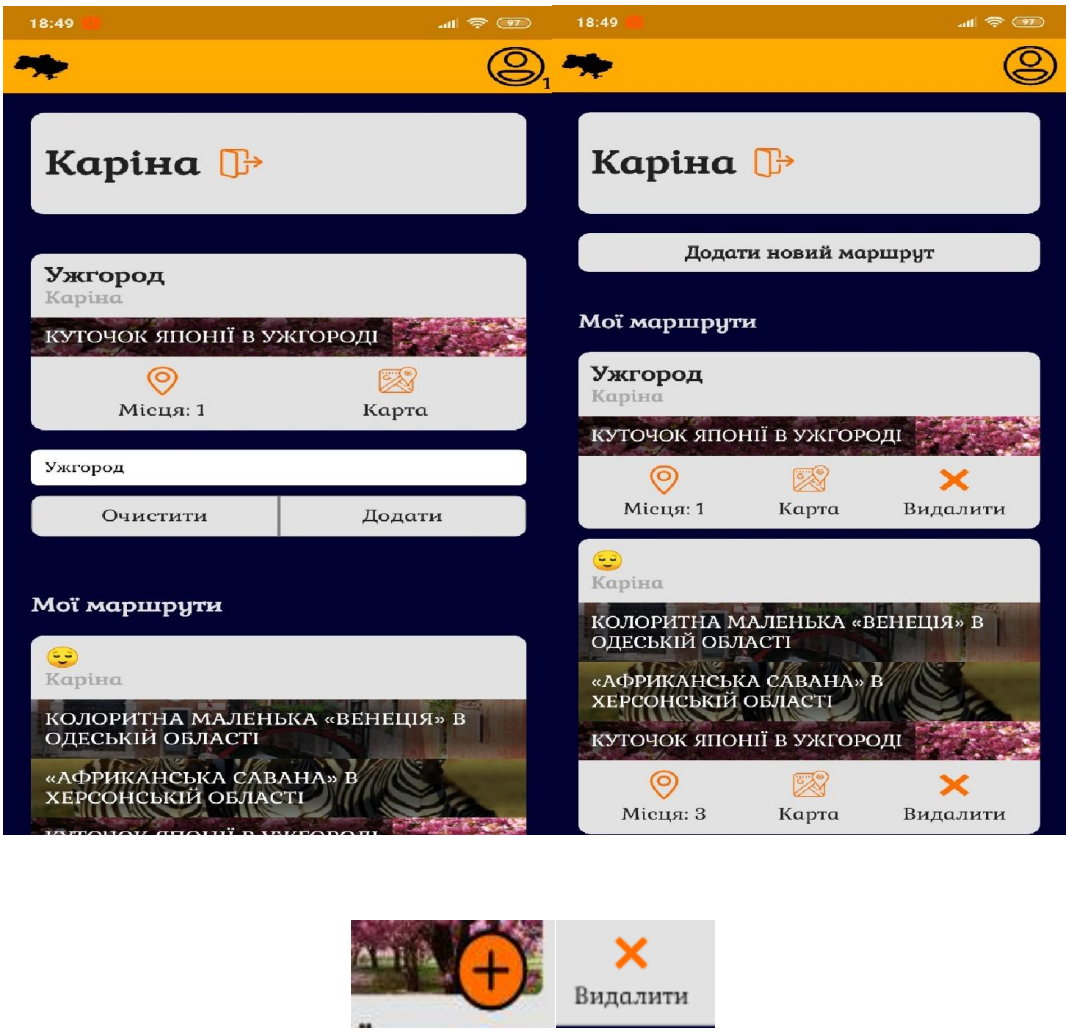

Рисунок 3.13 – Демонстрація збереження маршрутів та їх видалення

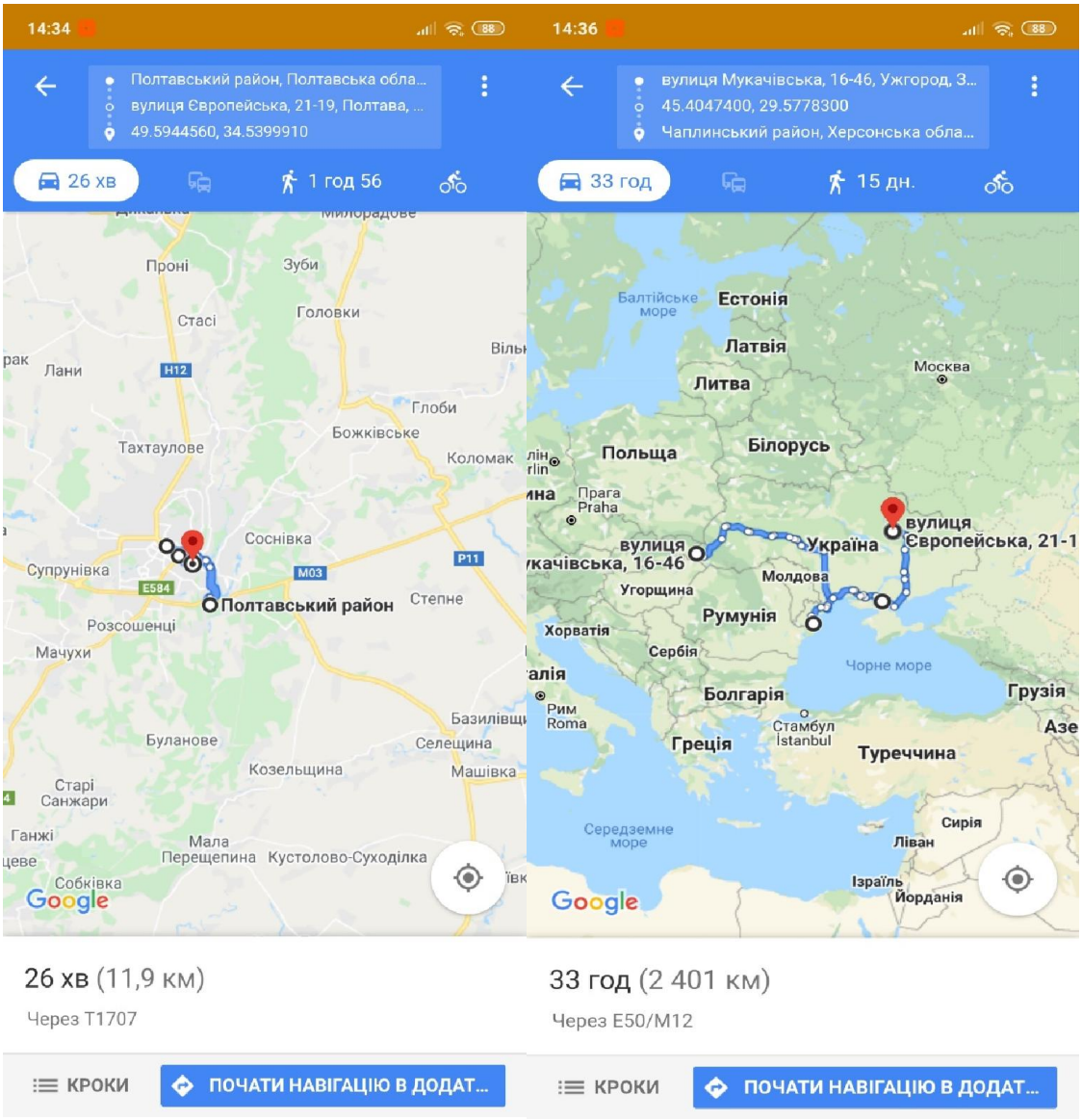

Рисунок 3.14 – Демонстрація прокладеного маршруту

Завдяки використанню сучасних технологій було розроблено мобільний додаток, який допоможе людям набагато швидше обирати маршрути подорожей за мінімальну кількість часу.

3.5 Впровадження інтелектуального модулю

Для реалізації інтелектуальних можливостей додатку було розроблено ряд додаткових особливостей:

Система оцінювання місць.

На екрані перегляду закладів зареєстрованому користувачу дається можливість оцінювання методом «вподобань». При натисканні на відповідну кнопку (рис. 3.15).

Рисунок 3.15 – Вподобання.

Кількість вподобань використовується для сортування закладів від найпопулярнішого до найменш популярного. Заклад з найбільшим рейтингом відображається угорі списку.

Окрім сортування, система рейтингу також використовується для нової можливості прокладання «швидкого маршруту», що є основним елементом інтелектуальної частини розробленого додатку.

Після натискання кнопки «Швидкий маршрут», відкривається нова форма для введення (рис. 3.16).

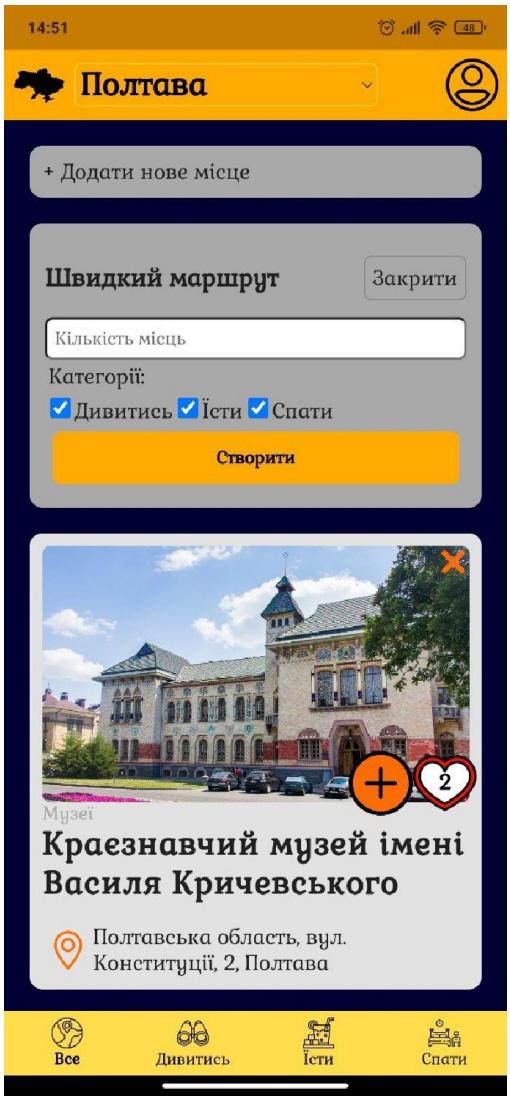

Рисунок 3.16 – Форма генерації швидкого маршруту.

Суть цієї функції полягає в тому, щоб автоматично сформувати маршрут із закладів певного обраного міста таким чином, щоб у нього входила певна кількість найпопулярніших місць обраних категорій. Тобто, це спосіб швидко та автоматично згенерувати маршрут із найкраще оцінених місць міста.

Після введення кількості місць, які б хотів відвідати користувач, а також які категорії повинні бути задіяні у вибірці, відбувається автоматична генерація маршруту. За допомогою множинного вибору виконується перебір та сортування усього масиву доступних закладів за рейтингом, їх фільтрація за категоріями і відсіювання після досягнення бажаного розміру нової

подорожі. По створенню цього маршруту проходить переадресування на профіль користувача, у якому його можна переглянути (рис. 3.17).

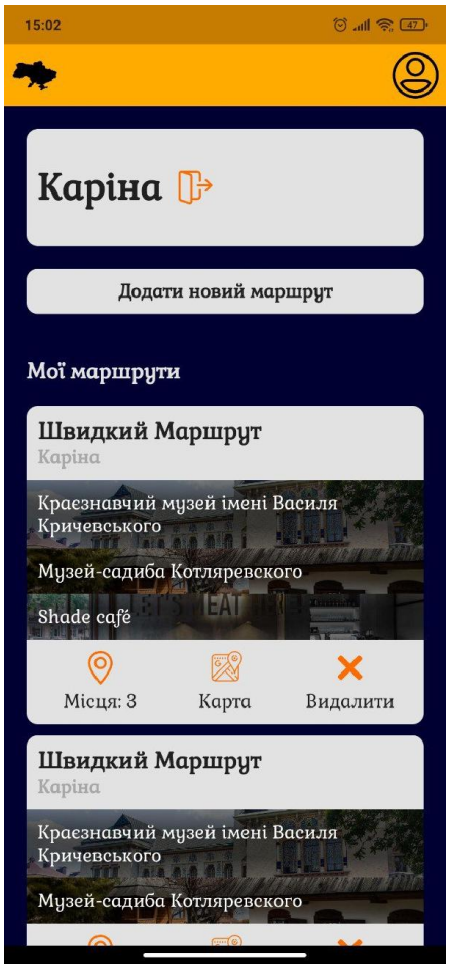

Рисунок 3.17 – Профіль користувача зі створеним маршрутом.

РОЗДІЛ 4

ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Тестування, як завершальний етап розробки веб-сайту, відіграє життєво-важливу роль у процесі створення якісного програмного забезпечення. Чим складніший сайт, тим більше часу потрібно на його перевірки і налагодження. Нажаль, існує безліч прикладів, коли розробники і замовники пропускали етап тестування сайту, що практично завжди призводить до великих фінансових і тимчасових витрат надалі, невдоволення користувачів ресурсу, і, в результаті, необхідність доопрацювання (або навіть повторної розробки) ресурсу. Залежно від специфіки проекту, на тестування може виділятись до 50% загального бюджету та часу.

Після завершення основних робіт зі створення програмної частини веб-ресурсу, фахівець контролю якості розробки (тестувальник) отримує всю необхідну документацію та матеріали і розпочинає тестування сайту. Для організації тестування веб-сайту передбачена спеціально розроблена методика, відповідно до якої і здійснюється перевірка. Підходи до тестування веб-ресурсів:

1. Звичайна практика полягає в тому, що після закінчення розробки ПЗ і до передачі продукту замовнику проводиться тестування сайту. Зазвичай вона виражається у вигляді окремої фази тестування в загальному циклі розробки ПЗ.

2. Тестування сайтів починається одночасно зі стартом розробки продукту. Продовжує весь той час поки йде розробка. Даний підхід вимагає більше ресурсів, але якість тестування помітно вище.

Рівні тестування веб-ресурсів:

1. Модульне тестування – тестується мінімально можливі компонент веб-сайту. Клас, функція і т.п.

2. Інтеграційне тестування – шукає проблеми в інтерфейсах (Не користувальницьких уявленнях даних) взаємодії між модулями системи.

3. Функціональне тестування – перевіряється відповідність системи вихідним вимогам до неї. Тестування сайту вирішує кілька основних завдань.

4. Дає впевненість у якості кінцевого продукту, підтверджує що всі заявлені функціональні вимоги реалізовані, веб сайт їм відповідає і не має помилок у програмному коді.

5. Підтверджує, що сайт здатний виконуватися у всіх заявлених режимах і на всіх підтримуваних ОС або Web-браузерах коректно.

6. Гарантує, що зберігаються і обробляються дані надійно захищені від стороннього доступу і злому.

7. Дозволяє переконатися в тому, що користувач може "інтуїтивно" використовувати продукт [16].

4.1 Тестування методом “чорного ящика”

Тестування методом “чорного ящика” – проведення функціонального тестування без відкритого доступу до коду системи. Ґрунтується на знанні тестувальником тільки вимог і специфікацій до продукту, що розробляється, а також світових стандартів. При здійсненні цього типу тестування, не має значення як реалізовано ПЗ, коректність написаного коду та алгоритмів.

Саме тому, тестувальнику не потрібно володіти знаннями в області програмування, він дивиться на продукт з точки зору кінцевого користувача. Простіше кажучи, коли клієнт користується продуктом, він не має знань, як він працює, які процеси відбуваються в залізі в ті моменти, коли він ним користується. Але він точно може сказати, подобається йому, як працює комп’ютер, чи ні. Так само відбувається і з розробленими програмами, іграми, сайтами і т.д. Оскільки це тип тестування, за визначенням він може включати інші його види [17].

Переваги:

1. Тестування проводиться з позиції кінцевого користувача і може допомогти виявити неточності і протиріччя в специфікації.

2. Тестувальнику немає необхідності знати мови програмування і заглиблюватися в особливості реалізації програми.

3. Тестування може проводитися фахівцями, незалежними від відділу розробки, що допомагає уникнути упередженого ставлення; − можна починати писати тест кейси як тільки готова специфікація.

Недоліки:

1. Не помічаючи вихідний код, не розуміючи за яким алгоритмом був розроблений той чи інший функціонал, які моменти не були враховані в розробці, глибина тестів може бути недостатньою, легко втратити моменти, коли може виникнути помилка при роботі з ПЗ.

2. Без чіткої специфікації (а це швидше за реальність на багатьох проектах) досить важко скласти ефективні тест-кейси.

Таблиця 4.1 – Тест-кейси

Дія	**Очікуваний результат**	**Отриманий результат**
Відкриття сторінки «Головна»	− Вікно Головна відкрито; − назва вікна – Головна; − Логотип сайту відображається зверху по центрі; − На формі 2 поля - Логін і Пароль; − Кнопка вхід доступна.	− Вікно Головна відкрито; − назва вікна – Головна; − Логотип сайту відображається зверху по центрі; − На формі 2 поля - Логін і Пароль; − Кнопка вхід доступна
Ввести будь-який рядок в поле Пароль	Введені символи заміняються на *****	Введені символи заміняються на *****
Введення не вірного логіну	Виведення рядку «Логін не вірний»	Виведення рядку «Логін не вірний»
Введення не Вірного паролю	Виведення рядку «Пароль не вірний»	Виведення рядку «Пароль не вірний»
Введені коректні дані	Вхід до профілю	Вхід до профілю

Продовження таблиці 4.1

Дія	Очікуваний результат	Отриманий результат
Натиснення кнопки реєстрація	Перехід на сторінку Реєстрація	Перехід на сторінку Реєстрація
Відправка пустої форми	Виведення помилки про обов'язковість заповнення поля.	Виведення помилки про обов'язковість заповнення поля.

4.2 Перевірка захисту та конфіденційності даних, використаних на веб-додатку подорожей

Захист інформації та даних є основною метою будь-якого сервісу. Кожен рік компанії витрачають велику кількість фінансів для збереження конфіденційності та непублічності інформації [18].

Реалізуючи даний сервіс були підібрані методи захисту інформації з метою збереження її первинного вигляду та недоступності користувачам мережі.

Усі форми, які ініціалізовані на ресурсі для відправки даних, використовують метод для надсилання POST, щоб інформація, яка передається на сервер не показувалась у рядку з посиланням, тобто таким чином ми захищаємо дані користувача, від змін, які може ввести будь-яка людина, яка зафіксує передачу даних.

4.3 Перевірка роботи форм на веб-ресурсі подорожей.

У даному додаткуактивовано такі форми:

1. Форма для реєстрації.
2. Форма введення логіну.
3. Форма додавання локації.
4. Форма створення маршруту.

Усі поля форм були перевірені на можливість:

1. Можливість заповнення усіх полів типу input.
2. Коректність роботи усіх check-box, розміщених на формах.
3. Відпрацювання усіх наявних кнопок, коректна реалізація функцій,які мають виконуватися, при їх натисканні.

Під час проведення даного виду тестування не було виявлено жодних багів, коректність роботи перевірена на різних девайсах, у різних браузерах.

ВИСНОВКИ

У ході виконання дипломної роботи було створено мобільний додаток подорожей Україною.

Відбувся аналізіснуючих аналогів веб-ресурсів, якідопомагають визначитися з маршрутом подорожей, але не вдалося знайти ні одного веб-додатку, який би зміг створювати власні маршрути та обирати саме ті місця для відпочинку, які були б максимально цікаві для шукача.

Розробка даного продукту та його функціональність підтверджує актуальність розробленого ресурсу, оскільки він є новим та єдиним у своєму роді.Даний додаток є гібридним, обрана клієнтська мова для його створення JavaScript, веб-сервер – Node.js.

Для даної дипломної роботи було обрано фреймворк Vue.js.Для веб-розробки використано MongoDB, оскільки це СУБД з відкритим кодом. Створення додатку також відбувалося задопомогою нелокального хостингу Heroku, тобто хмарна PaaS-платформа, що підтримує ряд мов програмування.

Візуальним середовищем програмування було вирішено обрати Visual Studio Code, який позиціонується як «легкий» редактор коду для кросплатформної розробки веб-і хмарних додатків.

Вибір програми управління БД - MongoDB Compass, яка є потужною, гнучкою і розширюваною базою даних загального призначення.

Завдяки проведеним дослідженням було виділено основні недоліки сервісів-аналогів та їх значні переваги, за допомогою яких і створювали функціонал власного веб-ресурсу, тому в програмній реалізації та якості він набагато кращий. Також було проведене тестування аналогів додатку, і як результатбуло отримано слабкі місця веб-ресурсів, що як наслідок надало смогу не повторити дані помилки.І звісно ж опрацьовані переваги даних сервісів,за допомогою яких, були внесені покращення у власному проекті.

Даний веб-ресурс має широкі можливості та нестандартні функції, які мають за мету залучити велику кількість клієнтів та зробити процес подорожей комфортнішим та швидшим.

Веб-сервіс має широкі перспективи розвитку в майбутньому для розширення своєї клієнтської бази та поліпшення інтерфейсу задля забезпечення всіх потреб користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wang L. 2015 Application research of artificial intelligence in electrical automation control. Automation and Instrumentation, (1) pp 113-114.

2. Rudas IJ, Fodor J (2008) Intelligent systems. Int J Comput Commun Control III(Suppl.):132–138.

3. Gretzel U (2011) Intelligent systems in tourism: a social science perspective. Ann Tour Res 38(3):757–779.

4. Lai WC, Hung WH (2018) A framework of cloud and AI based intelligent hotel. In: Proceedings of the 18th international conference on electronic business, ICEB, Guilin, 2–6 Dec, pp 36–43.

5. Buhalis D, Harwood T, Bogicevic V, Viglia G, Beldona S, Hofacker C (2019) Technological disruptions in services: lessons from tourism and hospitality. J Serv Manag 30:484–506.

6. Sterne J (2017) Artificial intelligence for marketing: practical applications. Wiley, Hoboken.

7. Kurzweil R (2005) The singularity is near: when humans transcend biology. Penguin, New York.

8. McCorduck P (2004) Machines who think. A personal inquiry into the history and prospects of artificial intelligence, 2nd edn. A K Peters/CRC Press, Boca Raton.

9. Hintze A (2016) Understanding the four types of AI, from reactive robots to self-aware beings. The conversation.

10. Bostrom N (2016) Superintelligence: paths, dangers, strategies. Oxford University Press, Oxford.

11. Russell SJ, Norvig P (2016) Artificial intelligence: a modern approach. Pearson Education Limited, Harlow.

12. Wirth N (2018) Hello marketing, what can artificial intelligence help you with? Int J Market Res 60(5):435–438.

13. Bowen J, Morosan C (2018) Beware hospitality industry: the robots are coming. Worldwide Hosp Tour Themes 10(6):726–733.

14. Buhalis D, Leung R (2018) Smart hospitality—interconnectivity and interoperability towards an ecosystem. Int J Hosp Manag 71:41–50

15. Tussyadiah I, Miller G (2019) Perceived impacts of artificial intelligence and responses to positive behaviour change intervention. In: Information and communication technologies in tourism 2019. Springer, Cham, pp 359–370

16. Xiao P., Zhong S. 2015 Research on online customer service based on artificial intelligence. Journal of Ppingxiang College, (3) pp 84-85.

17. Мобильные приложения: нативные, веб и гибридные [Електронний ресурс] – Режим доступу до ресурсу: https://agilie.com/ru/blog/mobilnyie-prilozhieniia-nativnyie-vieb-i-ghibridnyie.

18. Фрейн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Бен Фрейн – СПб.: Питер, 2014. – 304 с.

19. Раскин Д. Интерфейс: Новые направления в проектировании компьютерных систем / Д. Раскин. – М.: Символ-Плюс, 2004. – 272 с.

20. Современный учебник JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://learn.javascript.ru/.

21. Node.js и JavaScript для серверной разработки [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/ruvds/blog/345164/.

22. Vue.js для сомневающихся. Все, что нужно знать [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/329452/.

23. MongoDB Compass [Електронний ресурс] – Режим доступу до ресурсу: https://docs.mongodb.com/compass/master/.

24. Зандстра М. А. Объекты, шаблоны и методики программирования, 3-е издание / М. А. Зандстра. – СПб.: Издательский дом «Питер», 2011. – 560 с.

25. Heroku [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Heroku.

26. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://bizzapps.ru/p/visual-studio-code/.

27. Большой туториал по MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@Merrick_krg

28. Романюк О.Н. Веб-дизайн і комп'ютерна графіка. Навчальний посібник/ О.Н. Романюк, Д.І. Катєльніков, О.П. Косовець. – Вінниця: ВНТУ, 2007. – 147с.

29. Аутентификация с помощью JSON Web Token [Електронний ресурс] – Режим доступу до ресурсу: https://codex.so/jwt.

30. Introduction to JSON Web Tokens [Електронний ресурс] – Режим доступу до ресурсу: https://jwt.io/introduction/.

31. Історичний нарис про місто Полтава [Електронний ресурс] – Режим доступу до ресурсу: http://www.rada-poltava.gov.ua/city/historic/.

32. Айзенберг Брайан. Тестирование и оптимизация веб-сайтов: руководство по GoogleWebsiteOptimizer / Брайан Айзенберг, Джон Кварто вон Тивадар, Лайза Т. Дэвис. – изд. Диалектика, 2009. – 336 с. – ISBN 978- 5-8459-1542-9

33. Тестування методом чорного ящика [Електронний ресурс] – Режим доступу до ресурсу: http://ru.qatestlab.com/services/no-documentation/black-box-testing/.

34. Эспозитто Д. Разработка современных веб-приложений: анализ предметных областей и технологий / Дино Эспозитто. – К.: Диалектика-Вильямс, 2017. – 464 с.

ДОДАТОК А

ЛІСТИНГ КОДУ ДЛЯ БЕК-ЕНД РОЗРОБКИ

server/index.js

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');

const app = express();

//Middleware
app.use(bodyParser.json());
app.use(cors());

//Routes
const user = require('./routes/user');
const login = require('./routes/login');
const city = require('./routes/city');
const places = require('./routes/places');
const trips = require('./routes/trips');
app.use('/api/user', user);
app.use('/api/login', login);
app.use('/api/city', city);
app.use('/api/places', places);
app.use('/api/trips', trips);

//Handle production
if (process.env.NODE_ENV === 'production') {
    //Static folder
    app.use(express.static(__dirname + '/public/'));

    //Handle SPA
    app.get(/.*/, (req, res) => res.sendFile(__dirname + '/public/index.html'));
}
const port = process.env.PORT || 5000;

app.listen(port, () => console.log(`Server started on port ${port}`));
```

server/routes/city.js

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

const router = express.Router();

module.exports = router;
router.get('/', async (req, res) => {
    const cities = await loadCities();
    res.send(await cities.find({}).sort({ name: 1 }).toArray());
});

router.get('/:city_name/', async (req, res) => {
    try {
        const cities = await loadCities();
        res.send(await cities.findOne({ unique_name: req.params.city_name }));
    } catch (err) {
        res.status(500).send(err.message);
    }
});

async function loadCities() {
    const client = await mongodb.MongoClient.connect(
        'mongodb+srv://Valpoh:15935750@cluster0-js5rg.mongodb.net/test?retryWrites=true&w=majority',
        { useNewUrlParser: true, useUnifiedTopology: true }
    );
    return client.db('KarinaDB').collection('cities');
}
```

server/routes/login.js

```
const express = require('express');
const mongodb = require('mongodb');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const config = require('../config');
const validator = require('email-validator');

const router = express.Router();

module.exports = router;

router.post('/', async (req, res) => {
    try {
        const userList = await loadUsers();
        const user = await userList.findOne(
            { name: req.body.name },
            { fields: { accessToken: 0 } }
        );
        if (!user) return res.status(404).send(`No user found.`);
        let passwordIsValid = bcrypt.compareSync(req.body.password, user.password);
        if (!passwordIsValid) return res.status(401).send(`Wrong password.`);
        let token = jwt.sign({ id: user._id }, config.secret, { expiresIn: 86400 });
        await userList.update(
            { email: req.body.email },
            {
                $set: {
                    accessToken: token,
                },
            }
        );
        res.status(200).send({ auth: true, token: token, user: user });
    } catch (err) {
        res.status(500).send(err);
    }
});
async function loadUsers() {
    const client = await mongodb.MongoClient.connect(
        'mongodb+srv://Valpoh:15935750@cluster0-js5rg.mongodb.net/test?retryWrites=true&w=majority',
        { useNewUrlParser: true, useUnifiedTopology: true }
    );
    return client.db('KarinaDB').collection('users');
}
```

server/routes/places.js

```
const express = require('express');
const mongodb = require('mongodb');
const { v4: uuidv4 } = require('uuid');

const router = express.Router();

router.post('/', async (req, res) => {
    const places = await loadPlacesCollection();
    await places.insertOne(req.body);
    res.status(201).send();
});

router.get('/city/:city_name', async (req, res) => {
    const places = await loadPlacesCollection();
    res.send(await places.find({ city: req.params.city_name }).toArray());
});

router.get('/:location_id', async (req, res) => {
    try {
        const places = await loadPlacesCollection();
        res.send(
            await places.findOne({
                _id: new mongodb.ObjectID(req.params.location_id),
            })
        );
    } catch (err) {
        res.status(500).send(err.message);
    }
```

```
});

router.put('/:location_id', async (req, res) => {
    try {
        const places = await loadPlacesCollection();

        await places.updateOne(
            {
                _id: new mongodb.ObjectID(req.params.location_id),
            },
            {
                $set: {
                    name: req.body.name,
                    category: req.body.category,
                    description: req.body.description,
                },
            }
        );
    } catch (err) {
        res.status(500).send(err.message);
    }
});

router.delete('/:id', async (req, res) => {
    const places = await loadPlacesCollection();
    await places.deleteOne({ _id: new mongodb.ObjectID(req.params.id) });
    res.status(200).send();
});

router.post('/:id/comment', async (req, res) => {
    const places = await loadPlacesCollection();
    await places.update(
        { _id: new mongodb.ObjectID(req.params.id) },
        {
            $push: {
                comments: {
                    $each: [
                        {
                            text: req.body.comment,
                            user_id: req.body.user._id,
                            username: req.body.user.name,
                            date: new Date(),
                            id: uuidv4(),
                        },
                    ],
                    $position: 0,
                },
            },
        }
    );
    res.status(201).send();
});

router.delete('/:id/comment/:comment', async (req, res) => {
    const places = await loadPlacesCollection();
    await places.updateOne(
        { _id: new mongodb.ObjectID(req.params.id) },
        {
            $pull: {
                comments: {
                    id: req.params.comment,
                },
            },
        }
    );
    res.status(201).send();
});

async function loadPlacesCollection() {
    const client = await mongodb.MongoClient.connect(
        'mongodb+srv://Valpoh:15935750@cluster0-js5rg.mongodb.net/test?retryWrites=true&w=majority',
        { useNewUrlParser: true }
    );
    return client.db('KarinaDB').collection('places');
}
```

```
module.exports = router;
```

server/routes/trips.js

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

const router = express.Router();

router.post('/', async (req, res) => {
    const trips = await loadTripsCollection();
    await trips.insertOne(req.body);
    res.status(201).send();
});

router.get('/', async (req, res) => {
    const trips = await loadTripsCollection();
    res.send(await trips.find({}).toArray());
});

router.get('/:user', async (req, res) => {
    const trips = await loadTripsCollection();
    res.send(await trips.find({ 'user._id': req.params.user }).toArray());
});

router.delete('/:id', async (req, res) => {
    const trips = await loadTripsCollection();
    await trips.deleteOne({ _id: new mongodb.ObjectID(req.params.id) });
    res.status(200).send();
});

async function loadTripsCollection() {
    const client = await mongodb.MongoClient.connect(
        'mongodb+srv://Valpoh:15935750@cluster0-js5rg.mongodb.net/test?retryWrites=true&w=majority',
        { useNewUrlParser: true }
    );
    return client.db('KarinaDB').collection('trips');
}

module.exports = router;
```

server/routes/user.js

```
const express = require('express');
const mongodb = require('mongodb');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const config = require('../config');
const validator = require('email-validator');

const router = express.Router();

module.exports = router;

router.post('/', async (req, res) => {
    try {
        const userList = await loadUsers();
        if (!validator.validate(req.body.email)) {
            return res.status(500).send(`Email failed validation.`);
        }
        await userList.insertOne({
            name: req.body.name,
            email: req.body.email,
            registrationDate: new Date(),
            is_admin: false,
            password: bcrypt.hashSync(req.body.password, 8),
        });
        const user = await userList.findOne(
            { email: req.body.email },
            { fields: { accessToken: 0 } }
        );
        if (!user)
            return res
```

```
                .status(500)
                .send(`Error: failed to retrieve user information. ${err.message}`);
        let token = jwt.sign({ id: user._id }, config.secret, {
            expiresIn: 86400,
        });
        await userList.update(
            { email: req.body.email },
            {
                $set: {
                    accessToken: token,
                },
            }
        );
        res.status(201).send({ auth: true, token: token, user: user });
    } catch (err) {
        res.status(500).send(err);
    }
});

async function loadUsers() {
    const client = await mongodb.MongoClient.connect(
        'mongodb+srv://Valpoh:15935750@cluster0-js5rg.mongodb.net/test?retryWrites=true&w=majority',
        { useNewUrlParser: true, useUnifiedTopology: true }
    );
    return client.db('KarinaDB').collection('users');
}
```

ДОДАТОК Б

ЛІСТИНГ КОДУ ДЛЯ ФРОНТ-ЕНД РОЗРОБКИ

client/src/main.js

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

Vue.config.productionTip = false

new Vue({
  router,
  store,
  render: h => h(App)
}).$mount('#app')
```

client/src/App.vue

```
<template>
  <div id="app">
    <transition name="fade">
      <router-view :cities="Cities" />
    </transition>
  </div>
</template>

<script>
import { mapActions } from "vuex";
import Service from "./Service";
export default {
  methods: { ...mapActions(["fetchUser", "fetchCities"]) },
  async created() {
    this.fetchUser();
    await this.fetchCities();
    this.Cities = await Service.getCities();
  },
  data() {
    return {
      Cities: []
    };
  }
};
</script>
<style>
* {
  font-family: "Gabriela", serif;
}
#app {
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  color: #212121;
  z-index: -1;
}

.map-wrapper {
  width: 100%;
  height: 100%;
  position: absolute;
  overflow: hidden;
}
textarea {
  resize: none;
}
.map {
  position: relative;
  height: -webkit-fill-available;
  width: 100%;
  fill: url(#MyGradient);
  /* transition: 1s ease-in-out; */
```

```
}

.router-link-exact-active path,
.map path:hover {
  fill: #ff6d00;
  height: 100%;
  transition: 1s ease-in-out;
  cursor: pointer;
}

.map path {
  stroke: #646464;
  stroke-width: 1px;
  stroke-linejoin: round;
}

.corner-bottom-left {
  position: absolute;
  bottom: 1vh;
  left: 0;
  height: 40vh;
  transform: rotate(180deg);
}
.map-profile-button {
  position: absolute;
  display: flex;
  width: 55px;
  height: 55px;
  right: 0;
  z-index: 3;
}
.map-profile-button img {
  height: 75%;
  margin: auto;
}
.corner-top-right {
  position: absolute;
  top: 0;
  right: 0;
  height: 40vh;
}
@media (orientation: portrait) {
  .map {
    transform: rotate(90deg) scale(1.3);
    transform-origin: 50% 50%;
  }
}

@media (orientation: landscape) {
  .corner-bottom-left,
  .corner-top-right {
    height: 75vh;
  }
}

body,
html {
  margin: 0;

  height: 100%;
  width: 100%;
}
a {
  color: inherit;
}
body {
  background: rgb(0, 0, 50);
}

a {
  text-decoration: none;
}

a:visited {
  text-decoration: none;
  color: inherit;
```

```
}

.fade-enter-active,
.fade-leave-active {
  transition: opacity 1s;
}
.fade-enter, .fade-leave-to {
  opacity: 0;
}

.profile-name {
  display: flex;
}
.profile-name a {
  display: flex;
  padding-left: 15px;
}
.profile-name img {
  height: 30px;
  margin: auto;
}

.auth-wrapper,
.profile-wrapper {
  position: absolute;
  height: 100%;
  width: 100%;
  display: flex;
  flex-direction: column;
}
.auth-wrapper form {
  background-color: #e1e2e1;
  width: auto;
  margin: 20px;
  margin-top: 75px;
  border-radius: 10px;
  color: #212121;
  display: flex;
  flex-direction: column;
}
.auth-header {
  display: flex;
  font-weight: 700;
  font-size: 24px;
  padding-top: 30px;
  text-align: center;
}
.auth-header * {
  margin: auto;
  width: 100%;
}

.auth-header a.router-link-exact-active {
  color: #ff6d00;
}
.auth-header a {
  color: inherit;
}

.input-group {
  display: flex;
  flex-direction: column;
  padding: 30px;
}

.input-group .input {
  display: flex;
  flex-direction: column;
  padding-bottom: 10px;
}
.input label {
  text-align: left;
  padding-bottom: 5px;
}

.input input,
```

```
form .send,
.input textarea {
  border-radius: 6px;
  border: 1px solid grey;
  font-size: 16px;
  padding: 6px;
  transition: 0.25s ease-in-out;
}
form .send {
  margin-top: 30px;
  padding: 12px;
  background-color: #ffab00;
  border: none;
  cursor: pointer;
  font-weight: 700;
}
form .send:hover {
  background-color: #ff6d00;
}
.input input:hover,
.input input:focus {
  border: 1px solid #ffab00;
  outline: none;
}
.input input:focus {
  border: 1px solid #ff6d00;
}
form .error {
  color: red;
  text-align: justify;
  border: 1px solid violet;
  border-radius: 6px;
  padding: 10px;
  margin-bottom: -20px;
}
.item-wrapper {
  background-color: #e1e2e1;
  width: auto;
  margin: 20px;
  border-radius: 10px;
  padding: 10px;
  text-align: left;
}
.preview {
  position: relative;
  max-height: 200px;
  min-height: 50px;
}
.preview .preview-picture {
  width: 100%;
  border-radius: 6px;
  height: 200px;
  object-fit: cover;
  object-position: 50% 50%;
}
.preview .delete-btn {
  position: absolute;
  height: 25px;
  right: 0;
  top: 0;
  cursor: pointer;
}
.add-button {
  position: absolute;
  height: 50px;
  right: 10px;
  bottom: -10px;
  cursor: pointer;
}
.item-tag {
  color: darkgrey;
  font-size: 14px;
}
.item-name {
  margin: 0;
  letter-spacing: 1px;
```

```
}
.item-adress img {
  margin: auto;
  height: 30px;
  margin: auto 5px;
}
.item-adress {
  display: flex;
  margin-top: 1em;
}
.bounce-enter-active {
  animation: bounce-in 0.5s;
}
.bounce-leave-active {
  animation: bounce-in 0.5s reverse;
}
@keyframes bounce-in {
  0% {
    transform: scale(0);
  }
  100% {
    transform: scale(1);
  }
}
</style>
```

Компоненти:

client/src/views/Home.vue

```
<template>
  <div class="home">
    <Header :data="headerData" />
    <transition name="bounce">
      <router-view></router-view>
    </transition>
  </div>
</template>

<script>
// @ is an alias to /src
import Header from "../components/Header";
export default {
  name: "Home",
  components: { Header },
  props: {
    cities: Array
  },
  computed: {
    headerData() {
      const res = { data: this.cities };
      return res;
    }
  },
  async created() {
    if (this.$route.path == "/") {
      this.$router.push("map");
    }
  }
};
</script>
```

client/src/views/AddItem.vue

```
<template>
  <div class="item-wrapper add-item-wrapper">
    <div
      class="add-item-header"
      :class="{'img-preview': picUrl}"
      :style="picUrl ? `background-image: radial-
gradient(circle, rgba(0,0,0,0.8) 0%, rgba(50,50,50,0.65) 75%, rgba(100,100,100,0.5) 100%), url(${picUrl});
` : ''"
    >
      <h3>Додати нове місце</h3>
```

```
      <div class="add-item-back" @click="goBack">Закрити</div>
    </div>
    <form v-on:keyup.enter="handleSubmit" @input="error = ''">
      <div class="input-group">
        <div class="input">
          <label>Назва</label>
          <input type="text" v-model="name" required />
        </div>
        <div class="input">
          <label>Посилання на зображення</label>
          <input type="url" v-model="picUrl" required />
        </div>
        <div class="input">
          <label>Місто</label>
          <input v-if="city_name" type="text" required readonly :placeholder="city_name.name" />
        </div>
        <label>Розділ</label>
        <div class="input radio-group">
          <div>
            <input name="section" type="radio" value="watch" v-model="section" />
            <label>Дивитись</label>
          </div>
          <div>
            <input name="section" type="radio" value="eat" v-model="section" />
            <label>Їсти</label>
          </div>
          <div>
            <input name="section" type="radio" value="sleep" v-model="section" />
            <label>Спати</label>
          </div>
        </div>
        <div class="input">
          <label>Категорія</label>
          <input type="text" v-model="category" />
        </div>
        <div class="input">
          <label>Координати</label>
          <div class="add-coordinates">
            <input type="text" v-model="latitude" />
            <input type="text" v-model="longitude" />
          </div>
        </div>
        <div class="input">
          <label>Адреса</label>
          <input type="text" v-model="adress" />
        </div>
        <div class="input">
          <label>Опис</label>
          <textarea type="text" v-model="description" />
        </div>
        <div class="error" v-if="error">{{error}}</div>
        <button class="send" @click="handleSubmit">Додати</button>
      </div>
    </form>
  </div>
</template>

<script>
import { mapGetters } from "vuex";
import Service from "../Service";
import router from "../router";
export default {
  data() {
    return {
      name: "",
      picUrl: "",
      adress: "",
      city: this.$route.params.city_name,
      section: "",
      category: "",
      error: "",
      description: "",
      latitude: "",
      longitude: ""
    };
  },
```

```
  methods: {
    goBack() {
      router.go(-1);
    },
    async handleSubmit(e) {
      e.preventDefault();
      if (
        this.name &&
        this.picUrl &&
        this.adress &&
        this.category &&
        this.section &&
        this.latitude &&
        this.longitude
      ) {
        try {
          await Service.insertPlace({
            name: this.name,
            picUrl: this.picUrl,
            adress: this.adress,
            city: this.city,
            section: this.section,
            category: this.category,
            description: this.description,
            latitude: this.latitude,
            longitude: this.longitude
          }).then(async () => {
            this.$emit("placeAdded");
            this.goBack();
          });
        } catch (err) {
          this.error = err;
        }
      } else {
        this.error = "Заповніть всі поля";
      }
    }
  },
  computed: {
    ...mapGetters(["getCities"]),
    city_name() {
      const res = this.getCities.filter(obj => {
        return obj.unique_name == this.$route.params.city_name;
      });
      return res[0];
    }
  }
};
</script>

<style scoped>
.add-item-wrapper {
  display: flex;
  flex-direction: column;
  padding: 0;
}
.add-item-back {
  cursor: pointer;
  margin: auto;
  margin-right: 0;
  border: 1px solid darkgrey;
  padding: 6px;
  border-radius: 6px;
  height: fit-content;
}
.add-item-header {
  display: flex;
  padding: 30px 30px 0;
  margin-bottom: -30px;
  position: relative;
  border-top-right-radius: inherit;
  border-top-left-radius: inherit;
}

.img-preview {
  background-size: cover;
```

```
  background-position: 50%;
  background-blend-mode: multiply;
  color: white;
}

.radio-group {
  display: flex;
  flex-direction: row;
  margin: auto;
  width: 100%;
}

.add-coordinates {
  display: flex;
}
.add-coordinates * {
  width: 100%;
}
</style>
```

client/src/views/Location.vue

```
<template>
  <div class="location-wrapper">
    <div class="location-image" :style="`background-image: url(${location.picUrl})`">
      <div class="location-image-info">
        <div class="loc-name">{{ location.name }}</div>
        <div class="loc-cat">{{ location.category }}</div>
        <img src="../assets/plus.svg" class="loc-add-button" @click="addToTrip(location)" />
      </div>
    </div>
    <div class="loc-adress">
      <img src="../assets/marker.svg" class="loc-mark" />
      <div>
        {{ location.adress }}
        <span
          style="color: grey; font-size: 10px"
        >{{location.latitude}}, {{location.longitude}}</span>
      </div>
    </div>

    <div class="loc-desc" :style="expanded ? 'max-height: none' : ''">
      <h3>{{ location.name }}</h3>
      {{ location.description }}
    </div>
    <div
      class="expander"
      @click="
        () => {
          this.expanded = !this.expanded;
        }
      "
    >{{ expanded ? `Згорнути` : `Розгорнути` }}</div>

    <div class="comments-wrapper">
      <h3>Відгуки</h3>
      <textarea
        class="add-comment"
        rows="3"
        v-model="newComment"
        placeholder="Додати новий"
        v-if="getUser"
      ></textarea>
      <div class="comment-button" @click="handleSubmit" v-if="getUser">Додати</div>
      <transition-group name="bounce">
        <div class="comment" v-for="comment in location.comments" :key="comment.id">
          <div class="comment-user">
            <img src="../assets/user.svg" />
            <div style="margin: auto 0;">{{ comment.username }}</div>
            <div
              class="comment-delete"
              v-if="
                getUser && (comment.user_id = getUser._id || getUser.is_admin)
              "
              @click="deleteComment(comment.id)"
```

```
            >Видалити</div>
          </div>
          <div class="comment-text">{{ comment.text }}</div>
          <div class="comment-date">{{ getDate(comment.date) }}</div>
        </div>
      </transition-group>
    </div>
  </div>
</template>

<script>
import Service from "../Service";
import { mapGetters, mapMutations } from "vuex";
export default {
  data() {
    return {
      location: "",
      expanded: false,
      newComment: ""
    };
  },
  async created() {
    this.location = await Service.getPlace(this.$route.params.location_id);
  },
  computed: {
    ...mapGetters(["getUser"])
  },
  methods: {
    async updateLocation() {
      await Service.updatePlace(this.location, this.$route.params.location_id);
    },
    async handleSubmit() {
      await Service.insertComment(
        this.newComment,
        this.getUser,
        this.$route.params.location_id
      ).then(async () => {
        this.location = await Service.getPlace(this.$route.params.location_id);
      });
    },
    async deleteComment(id) {
      await Service.deleteComment(id, this.$route.params.location_id).then(
        async () => {
          this.location = await Service.getPlace(
            this.$route.params.location_id
          );
        }
      );
    },
    ...mapMutations(["addToTrip"]),
    getDate(date) {
      const res = new Date(date);
      return (
        res.getHours() +
        ":" +
        res.getMinutes() +
        " " +
        res.getDate() +
        "." +
        res.getMonth() +
        "." +
        res.getFullYear()
      );
    }
  }
};
</script>

<style scoped>
.location-wrapper {
  background-color: #e1e2e1;
  height: 100%;
  width: 100%;
  position: absolute;
  overflow-y: auto;
}
```

```
.location-wrapper textarea:focus {
  outline: none;
}

.location-image {
  width: 100%;
  background-size: cover;
  background-position: center center;
  height: 50%;
}

.location-image-info {
  padding-top: 55px;
  height: calc(100% - 55px);
  width: 100%;
  background-color: rgb(0, 0, 0, 0.4);
  color: white;
  display: flex;
  flex-direction: column;
  text-align: center;
}

.location-image-info textarea {
  color: white;
  text-align: center;
  background: none;
  border: none;
}

.loc-name {
  font-weight: bolder;
  font-size: 24px;
  margin: auto;
  width: 100%;
  margin-bottom: 6px;
}

.loc-cat {
  font-size: 16px;
  margin: auto;
  width: 100%;
  margin-top: 6px;
}

.loc-adress {
  background-color: #f5f5f6;
  padding: 6px;
  display: flex;
  position: relative;
}
.loc-adress .loc-mark {
  height: 30px;
  margin: auto 6px;
}
.loc-add-button {
  height: 50px;
  margin-left: auto;
  padding: 6px;
}
.loc-desc {
  white-space: pre-line;
  padding: 6px;
  max-height: 130px;
  overflow-y: auto;
  transition: 0.5s ease-in-out;
}

.expander {
  color: #ff6d00;
  text-align: center;
  padding: 6px;
  margin: 6px;
  border: 1px solid darkgrey;
  border-radius: 6px;
  cursor: pointer;
```

```
}

.comments-wrapper {
  padding: 6px;
  display: flex;
  flex-direction: column;
}
.comments-wrapper h3 {
  margin-bottom: 6px;
  margin-left: 5%;
}
.add-comment {
  border: 1px solid #ff6d00;
  border-top-left-radius: 6px;
  border-top-right-radius: 6px;
  width: 90%;
  margin: auto;
  padding: 6px;
}

.comment-button {
  width: 90%;
  background-color: #ff6d00;
  width: 90%;
  margin: auto;
  padding: 6px;
  border: 1px solid #ff6d00;
  border-bottom-left-radius: 6px;
  border-bottom-right-radius: 6px;
  color: black;
  text-align: center;
  font-weight: bold;
  margin-bottom: 6px;
}

.comment {
  white-space: pre-line;
  width: calc(90% + 12px);
  margin: 6px auto;
  background-color: #f5f5f6;
  border-radius: 6px;
}

.comment-user {
  display: flex;
  font-weight: bold;
  background-color: #ffab00;
  border-top-left-radius: 6px;
  border-top-right-radius: 6px;
}
.comment-user img {
  height: 30px;
  padding: 6px;
}
.comment-text {
  padding: 6px;
}
.comment-delete {
  margin: auto;
  margin-right: 6px;
  font-weight: normal;
}
.comment-date {
  font-size: 12px;
  color: darkgrey;
  margin-right: 3px;
  text-align: right;
}
</style>
```

client/src/views/Login.vue

```
<template>
    <div class="auth-wrapper">
        <form v-on:keyup.enter="handleSubmit" @input="error = ''">
            <div class="auth-header">
```

```
                <router-link to="register">Реєстрація</router-link>
                <router-link to="login">Увійти</router-link>
            </div>
            <div class="input-group">
                <div class="input">
                    <label>Ім’я</label>
                    <input
                        type="text"
                        name="username"
                        id="name"
                        v-model="name"
                        required
                    />
                </div>
                <div class="input">
                    <label>Пароль</label>
                    <input
                        type="password"
                        name="password"
                        id="password"
                        v-model="password"
                        required
                    />
                </div>
                <button class="send" @click="handleSubmit">Увійти</button>
            </div>
        </form>
    </div>
</template>

<script>
import Service from '../Service';
export default {
    data() {
        return {
            name: '',
            password: '',
            error: '',
        };
    },
    methods: {
        async handleSubmit(e) {
            e.preventDefault();
            if (this.name) {
                if (this.password.length > 0) {
                    try {
                        await Service.logIn(this);
                    } catch (err) {
                        this.error = err;
                        console.log(err);
                    }
                } else {
                    this.error += 'Введіть пароль';
                }
            } else {
                this.error += 'Введіть ім’я';
            }
        },
    },
};
</script>
```

client/src/views/Poltava.vue

```
<template>
  <div class="city-wrapper">
    <div class="city-content">
      <transition name="bounce">
        <router-view @placeAdded="updatePlaces"></router-view>
      </transition>
      <transition name="bounce">
        <router-link
          :to="`/${this.$route.params.city_name}/add`"
          v-if="getUser && getUser.is_admin && this.$route.path != `/${this.$route.params.city_name}/add`"
          class="item-wrapper link-add-new"
```

```
        >+ Додати нове місце</router-link>
      </transition>
      <transition name="bounce" v-for="place in places" :key="place._id">
        <Item
          :data="place"
          v-if="place.section == $route.params.section || $route.params.section == null"
          @itemDeleted="updatePlaces"
        />
      </transition>
    </div>
    <Footer />
  </div>
</template>

<script>
// @ is an alias to /src
import Item from "../components/Item";
import Footer from "../components/Footer";
import Service from "../Service";
import { mapGetters } from "vuex";
export default {
  name: "Poltava",
  data() {
    return {
      places: []
    };
  },
  components: {
    Item,
    Footer
  },
  computed: {
    ...mapGetters(["getUser"])
  },
  methods: {
    async updatePlaces(city) {
      this.places = (await Service.getPlaces(city)).reverse();
    }
  },
  async beforeRouteUpdate(to, from, next) {
    if (this.$route.params.city_name != to.params.city_name) {
      await this.updatePlaces(to.params.city_name);
    }
    next();
  },
  async created() {
    await this.updatePlaces(this.$route.params.city_name);
  }
};
</script>

<style scoped>
.city-wrapper {
  position: absolute;
  height: 100%;
  width: 100%;
  z-index: 1;
  display: flex;
  flex-direction: column;
}
.city-content {
  padding-top: 55px;
  width: 100%;
  height: 100%;
  overflow-y: scroll;
  padding-bottom: 75px;
}

.shrink-enter-active {
  animation: shrink-in 0.5s;
}
.shrink-leave-active {
  animation: shrink-in 0.5s reverse;
}
@keyframes shrink-in {
  0% {
```

```
    height: 0;
  }
  100% {
    height: auto;
  }
}
.link-add-new {
  text-align: center;
  display: flex;
  background-color: darkgrey;
}
</style>
```

client/src/views/Profile.vue

```
<template>
  <div class="profile-wrapper">
    <div class="profile-card">
      <div class="profile-name">
        <h1>{{ getUser.name }}</h1>
        <a href="javascript:void(0)" @click="logOut">
          <img src="../assets/logout.svg" alt="Вийти" />
        </a>
      </div>
    </div>
    <transition name="bounce" v-if="getTrip.length > 0">
      <div class="profile-trip-wrapper">
        <Trip :data="tripData" />
        <input
          type="text"
          v-model="tripName"
          style="margin: 10px 0; border-radius: 6px; border: none; padding: 10px; outline: none"
          :placeholder="placeholder"
        />
        <div class="profile-new-trip-buttons-wrapper">
          <div class="profile-new-trip-button" @click="resetTrip">Очистити</div>
          <div class="profile-new-trip-button" @click="handleSubmit">Додати</div>
        </div>
      </div>
    </transition>
    <router-link to="/map" v-else class="profile-add-button">Додати новий маршрут</router-link>
    <div class="profile-trip-wrapper">
      <h3 style="color: #e1e2e1">Мої маршрути</h3>
      <transition-group name="bounce">
        <Trip v-for="trip in Trips" :key="trip._id" :data="trip" @tripDeleted="onTripDeletion" />
      </transition-group>
    </div>
  </div>
</template>
<script>
import { mapGetters, mapMutations } from "vuex";
import Service from "../Service";
import Trip from "../components/Trip";
export default {
  computed: {
    ...mapGetters(["getUser", "getTrip"]),
    tripData() {
      return {
        locations: this.getTrip,
        name: this.tripName,
        user: { name: this.getUser.name, _id: this.getUser._id }
      };
    }
  },
  data() {
    return {
      tripName: "",
      placeholder: "Назва *",
      Trips: []
    };
  },
  methods: {
    ...mapMutations(["resetTrip"]),
    logOut() {
      Service.logOut(this);
    },
```

```
    async handleSubmit(e) {
      e.preventDefault();
      if (this.tripName) {
        await Service.insertTrip(this.tripData).then(async () => {
          this.resetTrip();
          await this.updateTrips();
        });
      } else {
        this.placeholder = "Необхідно вказати ім’я маршруту";
      }
    },
    async onTripDeletion() {
      await this.updateTrips();
    },
    async updateTrips() {
      this.Trips = (await Service.getUserTrips(this.getUser._id)).reverse();
    }
  },
  components: {
    Trip
  },
  async created() {
    await this.updateTrips();
  }
};
</script>
<style scoped>
.profile-card {
  background-color: #e1e2e1;
  width: auto;
  margin: 20px;
  margin-top: 75px;
  border-radius: 10px;
  padding: 10px;
}
.profile-trip-wrapper {
  padding: 20px;
  display: flex;
  flex-direction: column;
}

.profile-add-button {
  background-color: #e1e2e1;
  border-radius: 10px;
  padding: 10px;
  margin: 0 20px;
  font-weight: bold;
  text-align: center;
}

.profile-new-trip-buttons-wrapper {
  display: flex;
  background: #e1e2e1;
  border-radius: 10px;
  overflow: hidden;
}
.profile-new-trip-button {
  width: 100%;
  text-align: center;
  padding: 10px;
  border: 1px solid grey;
}
</style>
```

client/src/views/Register.vue

```
<template>
  <div class="auth-wrapper">
    <form v-on:keyup.enter="handleSubmit" @input="error = ''">
      <div class="auth-header">
        <router-link to="register">Реєстрація</router-link>
        <router-link to="login">Увійти</router-link>
      </div>
      <div class="input-group">
        <div class="input">
```

```
          <label>Ім’я</label>
          <input type="text" name="username" id="name" v-model="name" required />
        </div>
        <div class="input">
          <label>E-mail</label>
          <input type="email" name="email" id="email" v-model="email" required />
        </div>
        <div class="input">
          <label>Пароль</label>
          <input type="password" name="password" id="password" v-model="password" required />
        </div>
        <div class="input">
          <label>Підтвердити пароль</label>
          <input id="password-confirm" type="password" v-model="password_confirmation" required />
        </div>
        <div class="error" v-if="error">{{error}}</div>
        <button class="send" @click="handleSubmit">Реєстрація</button>
      </div>
    </form>
  </div>
</template>

<script>
import Service from "../Service";
export default {
  data() {
    return {
      name: "",
      email: "",
      password: "",
      password_confirmation: "",
      error: ""
    };
  },
  methods: {
    async handleSubmit(e) {
      e.preventDefault();
      if (
        this.password === this.password_confirmation &&
        this.password.length > 0
      )
        try {
          await Service.registerUser(this);
        } catch (err) {
          this.error = err;
          this.password = "";
          this.password_confirmation = "";
        }
      else {
        if (this.password != this.password_confirmation)
          this.error = "Неправильне підтвердження паролю";
        if (this.password.length <= 0) this.error = "Введіть пароль";
      }
    }
  }
};
</script>
```

client/src/components/Footer.vue

```
<template>
  <div class="footer-wrapper">
    <router-link
      :to="`/${this.$route.params.city_name}`"
      class="footer-item"
      :style="!$route.params.section ? 'font-weight: bolder' : ''"
    >
      <img src="../assets/location.svg" />
      <div class="footer-item-desription">Все</div>
    </router-link>
    <router-link
      :to="`/${this.$route.params.city_name}/watch`"
      class="footer-item"
      :style="$route.params.section == 'watch' ? 'font-weight: bolder' : ''"
    >
      <img src="../assets/find.svg" />
```

```
      <div class="footer-item-desription">Дивитись</div>
    </router-link>
    <router-link
      :to="`/${this.$route.params.city_name}/eat`"
      class="footer-item"
      :style="$route.params.section == 'eat' ? 'font-weight: bolder' : ''"
    >
      <img src="../assets/alcohol.svg" />
      <div class="footer-item-desription">Їсти</div>
    </router-link>
    <router-link
      :to="`/${this.$route.params.city_name}/sleep`"
      class="footer-item"
      :style="$route.params.section == 'sleep' ? 'font-weight: bolder' : ''"
    >
      <img src="../assets/hotel.svg" />
      <div class="footer-item-desription">Спати</div>
    </router-link>
  </div>
</template>

<style scoped>
.footer-wrapper {
  display: flex;
  text-align: center;
  background-color: #ffdd4b;
  color: black;
  overflow: hidden;
  width: 100%;
  position: absolute;
  bottom: 0;
  max-height: 75px;
  z-index: 2;
}
.footer-item {
  display: flex;
  flex-direction: column;
  margin: auto;
  height: 100%;
  width: 100%;
  padding: 5px;
}
.footer-item * {
  margin: auto;
}

.footer-item img {
  width: 25px;
}

.footer-item-desription {
  font-size: 12px;
}

@media only screen and (min-width: 700px) {
  .footer-item {
    flex-direction: row;
    margin: 10px 0;
  }
  .footer-item img {
    margin: auto 5px auto auto;
  }
  .footer-item-desription {
    margin: auto auto auto 5px;
  }
}
</style>
```

client/src/components/Header.vue

```
<template>
  <div
    class="header-wrapper"
    :style="$route.params.location_id ? `background: linear-
gradient(180deg, rgba(255,255,255,0.6) 0%, rgba(255,255,255,0.3) 70%, rgba(255,255,255,0.0) 100%);` : ``"
  >
```

```
    <router-link to="/map" class="header-button">
      <img src="../assets/ukraine-map.svg" />
    </router-link>
    <select @change="redirect" class="city-name" v-if="$route.params.city_name">
      <option
        v-for="city in data.data"
        :key="city._id"
        :selected="true ? $route.params.city_name == city.unique_name : false"
        :value="`/${city.unique_name}`"
      >{{city.name}}</option>
    </select>
    <router-link to="/profile" class="header-button profile">
      <img src="../assets/user.svg" />
    </router-link>
    <div class="header-trip-number">{{getTrip.length != 0 ? getTrip.length : ''}}</div>
  </div>
</template>

<script>
import { mapGetters } from "vuex";
export default {
  props: {
    data: Object
  },
  created() {},
  methods: {
    redirect(e) {
      this.$router.push(e.target.value);
    }
  },
  computed: {
    tripValue() {
      return JSON.parse(localStorage.getItem("trip")).length;
    },
    ...mapGetters(["getTrip"])
  }
};
</script>

<style scoped>
.header-wrapper {
  background-color: #ffab00;
  color: black;
  display: flex;
  position: absolute;
  width: 100%;
  height: 55px;
  vertical-align: middle;
  z-index: 2;
  transition: 0.5s ease-in-out;
}
.city-name {
  font-weight: 700;
  font-size: 24px;
  display: flex;
  margin: auto 0;
  background-color: inherit;
  border-radius: 6px;
  border: 1px solid #ffdd4b;
}
.city-name:focus {
  outline: none;
}
.header-button {
  display: flex;
  width: 55px;
}
.header-button img {
  height: 75%;
  margin: auto;
}
.header-button.profile {
  margin-left: auto;
}
.header-button.profile img {
  height: 75%;
```

```
  min-width: 0;
  margin: auto;
}
.header-trip-number {
  position: absolute;
  padding-right: 2px;
  right: 0;
  bottom: 0;
}
</style>
```

client/src/components/Item.vue

```
<template>
  <div class="item-wrapper">
    <div class="preview">
      <router-link :to="`/location/${data._id}`">
        <img :src="data.picUrl" alt class="preview-picture" />
      </router-link>
      <img
        class="delete-btn"
        v-if="getUser && getUser.is_admin"
        src="../assets/delete.svg"
        @click="deleteItem"
      />
      <img src="../assets/plus.svg" class="add-button" @click="addToTrip(data)" />
    </div>
    <div class="item-tag">{{ data.category }}</div>
    <router-link :to="`/location/${data._id}`">
      <h2 class="item-name">{{ data.name }}</h2>
    </router-link>
    <div class="item-adress">
      <img src="../assets/marker.svg" />
      <div class="item-adress-text">{{ data.adress }}</div>
    </div>
  </div>
</template>

<script>
import { mapGetters, mapMutations } from "vuex";
import Service from "../Service";
export default {
  props: {
    data: Object
  },
  computed: {
    ...mapGetters(["getUser"])
  },
  methods: {
    async deleteItem() {
      await Service.deleteItem(this.data._id).then(() =>
        this.$emit("itemDeleted")
      );
    },
    ...mapMutations(["addToTrip"])
  }
};
</script>

<style scoped></style>
```

client/src/components/Trip.vue

```
<template>
  <div class="trip-wrapper">
    <div class="trip-header">
      <div style="font-weight: bolder; font-size: 20px">{{data.name}}</div>
      <div style="color: darkgrey">{{data.user.name}}</div>
    </div>
    <TripLocation v-for="location in data.locations" :key="location._id" :data="location"></TripLocation>
    <div class="trip-footer">
      <div>
        <img src="../assets/marker.svg" />
        <div>Місця: {{data.locations.length}}</div>
      </div>
```

```
      <a :href="mapLink">
        <img src="../assets/icons8-google-maps-old.svg" />
        <div>Карта</div>
      </a>
      <div
        v-if="getUser && data._id && (getUser._id == data.user._id || getUser.is_admin)"
        @click="deleteTrip"
      >
        <img src="../assets/delete.svg" />
        <div>Видалити</div>
      </div>
    </div>
  </div>
</template>

<script>
import { mapGetters } from "vuex";
import Service from "../Service";
import TripLocation from "./TripLocation";
export default {
  props: {
    data: Object
  },
  computed: {
    ...mapGetters(["getUser"]),
    mapLink() {
      var link = "https://www.google.com/maps/dir/?api=1";
      this.data.locations.forEach(element => {
        if (element.latitude && element.longitude) {
          if (this.data.locations.indexOf(element) == 0) {
            link += "&origin=";
          } else if (
            this.data.locations.indexOf(element) ==
            this.data.locations.length - 1
          ) {
            link += "&destination=";
          } else if (this.data.locations.indexOf(element) == 1) {
            link += "&waypoints=";
          } else {
            link += "%7C";
          }
          link += element.latitude + "," + element.longitude;
        }
      });
      return link;
    }
  },
  methods: {
    async deleteTrip() {
      await Service.deleteTrip(this.data._id).then(() =>
        this.$emit("tripDeleted")
      );
    }
  },
  components: {
    TripLocation
  }
};
</script>

<style scoped>
.trip-wrapper {
  width: 100%;
  display: flex;
  border-radius: 10px;
  background-color: #e1e2e1;
  flex-direction: column;
  overflow: hidden;
  margin-bottom: 10px;
}

.trip-footer {
  display: flex;
  padding: 10px;
}
```

```
.trip-footer * {
  width: 100%;
  margin: auto;
  text-align: center;
}
.trip-footer img {
  height: 30px;
}

.trip-header {
  padding: 10px;
  font-weight: bolder;
}
</style>
```

client/src/components/TripLocation.vue

```
<template>
  <router-link
    :to="`/location/${data._id}`"
    class="trip-location-wrapper"
    :style="`background-image: url(${data.picUrl})`"
  >
    <div class="trip-location-text">{{data.name}}</div>
  </router-link>
</template>

<script>
export default {
  props: {
    data: Object
  }
};
</script>

<style  scoped>
.trip-location-wrapper {
  display: flex;
  height: fit-content;
  color: white;
  background-size: cover;
  background-position: center center;
}
.trip-location-text {
  padding: 10px;
  background: linear-gradient(
    rgb(0, 0, 0, 0.5) 0%,
    rgb(0, 0, 0, 0.7) 30%,
    rgb(0, 0, 0, 0.7) 70%,
    rgb(0, 0, 0, 0.5) 100%
  );
}
</style>
```

ДОДАТОКВ Vue Router

client/src/router/index.js

```
import Vue from 'vue';
import VueRouter from 'vue-router';
import Home from '../views/Home.vue';

Vue.use(VueRouter);
const SITE_TITLE = 'Travel Ukraine';

const routes = [
{
path: '/map',
name: 'Map',
component: () =>
import(/* webpackChunkName: "about" */ '../components/Map.vue'),
},
{
path: '/',
name: 'Home',
component: Home,
children: [
{
path: 'profile',
component: () => import('../views/Profile.vue'),
meta: { requiresAuth: true },
},
{
path: '/register',
component: () => import('../views/Register.vue'),
meta: { guest: true },
},
{
path: '/login',
component: () => import('../views/Login.vue'),
meta: { guest: true },
},
{
path: '/location/:location_id',
component: () => import('../views/Location.vue'),
},
{
path: '/:city_name',
component: () => import('../views/Poltava.vue'),
children: [
{
path: '/:city_name/add',
component: () => import('../views/AddItem.vue'),
meta: { requiresAuth: true, requiresAdmin: true },
},
{
path: '/:city_name/:section',
},
],
},
],
},
];

const router = new VueRouter({
routes,
});

router.beforeEach((to, from, next) => {
if (to.matched.some((record) => record.meta.requiresAuth)) {
if (localStorage.getItem('jwt') === null) {
next({
path: '/login',
params: { nextUrl: to.fullPath },
});
document.title = to.meta.title + ' - ' + SITE_TITLE;
} else {
let user = JSON.parse(localStorage.getItem('user'));
```

```
if (to.matched.some((record) => record.meta.requiresAdmin)) {
if (user.is_admin === true) {
next();
document.title = to.meta.title + ' - ' + SITE_TITLE;
} else {
next(false);
document.title = to.meta.title + ' - ' + SITE_TITLE;
}
} else {
next();
document.title = to.meta.title + ' - ' + SITE_TITLE;
}
}
} else if (to.matched.some((record) => record.meta.guest)) {
if (localStorage.getItem('jwt') == null) {
next();
document.title = to.meta.title + ' - ' + SITE_TITLE;
} else {
next(false);
document.title = to.meta.title + ' - ' + SITE_TITLE;
}
} else {
next();
document.title = to.meta.title + ' - ' + SITE_TITLE;
}
});

export default router;
```

ДОДАТОКГ

Vuex

client/src/store/index.js

```
import Vue from 'vue';
import Vuex from 'vuex';
import User from './modules/user';
import Cities from './modules/cities';
import Trip from './modules/trip';
Vue.use(Vuex);

export default new Vuex.Store({
state: {},
mutations: {},
actions: {},
modules: { User, Cities, Trip },
});
```

client/src/store/modules/cities.js

```
import Service from '../../Service';

const state = {
cities: [],
};

const getters = {
getCities: (state) => state.cities,
};

const actions = {
async fetchCities({ commit }) {
commit('setCities', await Service.getCities());
},
};

const mutations = {
setCities: (state, cities) => {
state.cities = cities;
},
};

export default {
state,
getters,
actions,
mutations,
};
```

client/src/store/modules/trip.js

```
const state = {
trip: [],
};

const getters = {
getTrip: (state) => state.trip,
};

const mutations = {
addToTrip: (state, location) => {
if (state.trip.indexOf(location) === -1) {
state.trip.push(location);
}
},
resetTrip: (state) => {
state.trip = [];
},
};
```

```
export default {
state,
getters,
mutations,
};
```

client/src/store/modules/user.js

```
const state = {
user: {},
};

const getters = {
getUser: (state) => state.user,
};

const actions = {
fetchUser({ commit }) {
if (localStorage.getItem('jwt') != null) {
const response = JSON.parse(localStorage.getItem('user'));
commit('setUser', response);
} else {
commit('setUser', null);
}
},
clearUser({ commit }) {
commit('setUser', null);
},
setUser({ commit }, user) {
commit('setUser', user);
},
};

const mutations = {
setUser: (state, user) => {
state.user = user;
},
};

export default {
state,
getters,
actions,
mutations,
};
```

ДОДАТОК Д

ЛІСТИНГ КОДУ ДЛЯ ПЕРЕДАЧІ ДАНИХ МІЖ ФРОНТ-ЕНДОМ ТА БЕК-ЕНДОМ

client/src/Service.vue

```
import axios from 'axios';
import Store from './store/index.js';
const userUrl = '/api/user/';
const loginUrl = '/api/login/';
const cityUrl = '/api/city/';
const placesUrl = '/api/places/';
const tripsUrl = '/api/trips/';

class PostService {
static registerUser(arg) {
return new Promise((resolve, reject) => {
axios
.post(userUrl, arg.$data)
.then((response) => {
localStorage.setItem('user', JSON.stringify(response.data.user));
localStorage.setItem('jwt', response.data.token);
Store.dispatch('setUser', response.data.user);
if (localStorage.getItem('jwt') != null) {
if (arg.$route.params.nextUrl != null) {
arg.$router.push(arg.$route.params.nextUrl);
} else {
arg.$router.push('/profile');
}
}
})
.catch((error) => {
reject(error.response.data);
});
});
}

static logOut(arg) {
localStorage.removeItem('jwt');
localStorage.removeItem('user');
Store.dispatch('clearUser');
if (localStorage.getItem('jwt') == null) {
if (arg.$route.meta.requiresAuth) {
arg.$router.push('login');
}
}
}

static logIn(arg) {
return new Promise((resolve, reject) => {
axios
.post(loginUrl, arg.$data)
.then((response) => {
localStorage.setItem('user', JSON.stringify(response.data.user));
localStorage.setItem('jwt', response.data.token);
if (localStorage.getItem('jwt') != null) {
Store.dispatch('setUser', response.data.user);
if (arg.$route.params.nextUrl != null) {
arg.$router.push(arg.$route.params.nextUrl);
} else {
arg.$router.push('/profile');
}
}
})
.catch((error) => {
reject(error.response.data);
});
});
}

static getCity(name) {
```

```
return new Promise((resolve, reject) => {
axios
.get(`${cityUrl}${name}/`)
.then((res) => {
const data = res.data;
resolve(data);
})
.catch((err) => {
reject(err);
});
});
}

static getCities() {
return new Promise((resolve, reject) => {
axios
.get(cityUrl)
.then((res) => {
const data = res.data;
resolve(
data.map((post) => ({
...post,
}))
);
})
.catch((err) => {
reject(err);
});
});
}

static insertPlace(arg) {
return axios.post(placesUrl, arg);
}

static updatePlace(arg, id) {
return axios.put(`${placesUrl}${id}/`, arg);
}

static getPlaces(city_name) {
return new Promise((resolve, reject) => {
axios
.get(`${placesUrl}city/${city_name}/`)
.then((res) => {
const data = res.data;
resolve(
data.map((post) => ({
...post,
}))
);
})
.catch((err) => {
reject(err);
});
});
}

static getPlace(id) {
return new Promise((resolve, reject) => {
axios
.get(`${placesUrl}${id}/`)
.then((res) => {
const data = res.data;
resolve(data);
})
.catch((err) => {
reject(err);
});
});
}

static deleteItem(id) {
return axios.delete(`${placesUrl}${id}`);
}

static insertComment(comment, user, id) {
```

```
return axios.post(`${placesUrl}${id}/comment/`, {
comment: comment,
user: user,
});
}

static deleteComment(comment_id, location_id) {
return axios.delete(`${placesUrl}${location_id}/comment/${comment_id}`);
}

static addToTrip(id) {
var locStore = JSON.parse(localStorage.getItem('trip'));
if (locStore) {
if (locStore.indexOf(id) === -1) {
locStore.push(id);
localStorage.setItem('trip', JSON.stringify(locStore));
}
} else {
localStorage.setItem('trip', JSON.stringify([id]));
}
}

static insertTrip(data) {
return axios.post(`${tripsUrl}`, data);
}

static getTrips() {
return new Promise((resolve, reject) => {
axios
.get(`${tripsUrl}`)
.then((res) => {
const data = res.data;
resolve(
data.map((post) => ({
...post,
}))
);
})
.catch((err) => {
reject(err);
});
});
}

static getUserTrips(user) {
return new Promise((resolve, reject) => {
axios
.get(`${tripsUrl}/${user}`)
.then((res) => {
const data = res.data;
resolve(
data.map((post) => ({
...post,
}))
);
})
.catch((err) => {
reject(err);
});
});
}

static deleteTrip(id) {
return axios.delete(`${tripsUrl}${id}`);
}
}

export default PostService;
```