АНОТАЦІЯ

*Зм.*

*Арк.*

Прізвище

Підпис

Дата

*Арк.*

2

ДП ІС-2306.1448-c.ПЗ

Розроб.

*Григор’єва Ю.В.*

Перевірив.

.

*Халус О.А.*

*Н. кон.*

*Халус О.А.*

*Затв.*

*Халус О.А.*

*Літ.*

*Аркушів*

–

*КПІ ФІОТ*

*кафедра АСОІУ гр. ІС-23*

**Структура та обсяг роботи.** Пояснювальна записка дипломного проекту складається з п’яти розділів, містить 18 рисунків, 12 таблиць, 1 додаток, 7 джерел.

Дипломний проект присвячений розробці комплексу задач з надання рекомендацій мандрівникам України.

В дипломному проекті були розглянуті: алгоритми багатокритеріальної оптимізації, методи визначення подібності користувачів, алгоритми колаборативної фільтрації.

У розділі з інформаційного забезпечення були визначені вхідні та вихідні дані до комплексу задач, була розроблена структура бази даних, яка відповідає поставленим цілям проекту.

У розділі з математичного забезпечення було обґрунтовано створений евристичний підхід. Був запропонований алгоритм пошуку рекомендаційних об’єктів для користувачів системи.

У розділі з програмного забезпечення описані основні засоби розробки комплексу задач, висунуті вимоги до технічного забезпечення, обрано та обґрунтовано архітектуру програмного забезпечення.

У технологічному розділі описана інструкція користувача та проведене тестування комплексу задач.

Розділ охорони праці містить науково теоретичні та практичні результати проблем охорони праці, які виникають при використанні даної системи.

РЕКОМЕНДАЦІНА СИСТЕМА, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, СХОЖІСТЬ, ІНДЕКС ЖАККАРА

ABSTRACT

**Structure and scope of work.** Explanatory memorandum degree project consists of five chapters, contains 18 pictures, 12 tables, 1 appendix, 7 sources.

Дипломний проект присвячений розробці комплексу задач з надання рекомендацій мандрівникам України.

Diplom project dedicated to the developments of the software solution for giving recommendations to the tourists of Ukraine (…)

ЗМІСТ

[Вступ 6](#_Toc452492466)

[1 Загальні положення 8](#_Toc452492467)

[**1.1** **Опис предметного середовища** 8](#_Toc452492468)

[**1.1.1** **Опис процесу діяльності** 9](#_Toc452492469)

[**1.1.2** **Опис функціональної моделі** 11](#_Toc452492470)

[**1.2** **Огляд наявних аналогів** 12](#_Toc452492471)

[**1.3** **Постановка задачі** 16](#_Toc452492472)

[**1.3.1** **Призначення розробки** 16](#_Toc452492473)

[**1.3.2** **Цілі та задачі розробки** 16](#_Toc452492474)

[**Висновок до розділу** 16](#_Toc452492475)

[2 Інформаційне забезпечення 17](#_Toc452492476)

[**2.1** **Вхідні дані** 17](#_Toc452492477)

[**2.2** **Вихідні дані** 18](#_Toc452492478)

[**2.3** **Опис структури бази даних** 18](#_Toc452492479)

[**Висновок до розділу** 19](#_Toc452492480)

[3 Математичне забезпечення 20](#_Toc452492481)

[**3.1** **Змістовна постановка задачі** 20](#_Toc452492482)

[**3.2** **Математична постановка задачі** 20](#_Toc452492483)

[**3.3** **Обґрунтування методу розв’язання** 21](#_Toc452492484)

[**3.4** **Опис методів розв’язання** 22](#_Toc452492485)

[**Висновок до розділу** 24](#_Toc452492486)

[4 Програмне та технічне забезпечення 25](#_Toc452492487)

[**4.1** **Засоби розробки** 25](#_Toc452492488)

[**4.2** **Вимоги до технічного забезпечення** 27](#_Toc452492489)

[**4.3** **Архітектура програмного забезпечення** 27](#_Toc452492490)

[**4.3.1** **Діаграма класів** 27](#_Toc452492491)

[**4.3.2** **Діаграма послідовності** 28](#_Toc452492492)

[**4.3.3** **Діаграма розгортання** 29](#_Toc452492493)

[**4.3.4** **Специфікації функцій** 29](#_Toc452492494)

[**4.4** **Висновок** 33](#_Toc452492495)

[5 Технологічний розділ 34](#_Toc452492496)

[**5.1** **Керівництво користувача** 34](#_Toc452492497)

[**5.2** **Випробування програмного продукту** 41](#_Toc452492498)

[**5.2.1** **Мета випробувань** 41](#_Toc452492499)

[**5.2.2** **Загальні положення** 41](#_Toc452492500)

[**5.2.3** **Результати випробувань** 41](#_Toc452492501)

[**Висновок до розділу** 44](#_Toc452492502)

[ЗАГАЛЬНІ ВИСНОВКИ 45](#_Toc452492503)

[Перелік посилань 46](#_Toc452492504)

Вступ

Інтерактивність - це принцип організації системи, при якому мета досягається інформаційним обміном елементів цієї системи. Елементами інтерактивності є всі елементи взаємодіючої системи, за допомогою яких відбувається взаємодія з іншою системою / людиною (користувачем).

Інтерактивність вже давно стала обов’язковою складовою популярних систем. Більшість великих комерційних і соціальних веб-сайтів використовують рекомендації як один із основних інструментів інтерактивності. Такі системи надають користувачам різноманітні пропозиції, наприклад, товари, фільми, музику тощо. Такі системи змінили способи взаємодії веб-сайтів зі своїми користувачами. Замість скупого надання статистики про найпопулярніші речі, системи змінюють підхід та використовують інтерактивність для покращення досвіду користувачів у межах використання системи.

Як свідчить статистика, внутрішній туризм в Україні значно збільшився. Проте подорожі спрямовані до найвідоміших туристичних центрів: Київ, Карпати, Львів, Одеса. Людям не вистачає інформації про існуючі малі туристичні об’єкти, а тому маловідомі місця так і залишаються маловідвіданими.

Дипломний проект присвячений розробці комплексу задач, що дозволить вирішити проблему за допомогою рекомендаційної системи туристичних місць, яка дасть змогу людям ознайомлюватись з новими об’єктами в Україні та популяризує маловідомі об’єкти.

**Об’єкт** - природно-кліматичний, оздоровчий, історико-культурний, пізнавальний та/або соціально-побутовий ресурс певної території, що задовольняють різноманітні потреби мандрівника.

**Мапа** – повний набір туристичних об’єктів України.

**Рекомендаційна система** – система фільтрації інформації, яка будує рейтинговий перелік об'єктів (фільми, музика, книги, новини), яким користувач може надати перевагу. Для цього використовується інформація з профілю користувача.

Отже, наша система передбачає зв’язок у дві сторони. Перш за все, це зв’язок від системи до користувача. Система з самого початку запуску вже повинна мати певний набір даних, з яких і буде формуватись інформаційне поле мапи, а тому користувач вже має змогу отримати інформацію, що його цікавить. З іншої сторони, користувачу доступна можливість змінювати загальний вигляд мапи. Ця можливість полягає у декількох функціях:

* можливість додавати нові об’єкти;
* можливість доповнювати опис до об’єктів;
* можливість оцінювати об’єкт.

На основі отриманих від користувача даних ми прагнемо отримати повноціннний робочий сервіс, основною функцією якого буде персоналізовані рекомендації об’єктів для кожного користувача.

Розроблюваний комплекс задач призначений для людей, що цікавляться навколишнім світом, хочуть дізнатись більше та відвідати особливі та цікаві місця.

**Практичне значення одержаних результатів.** Розроблено алгоритм пошуку рекомендацій для користувача.

# Загальні положення

## **Опис предметного середовища**

Туристичний сегмент ринку в Україні знаходиться у стадії активного розвитку. Про це свідчить статистика внутрішнього та зовнішнього туризму України. З проведеного аналізу видно, що за останні 2 роки ситуація щодо туризму в Україні значно змінилася. З рисунку 1.1 бачимо, що кількість іноземних туристів зменшилася. Це пов’язано з низькою лояльністю держави до іноземних туристів (довгий процес отримання візи, високий консульський збір) та нестабільною ситуацією в країні.

Рисунок 1.1 – Кількість іноземних туристів в Україні (тис.)

В той же час внутрішній туризм має великий потенціал стрімкого розвитку, так як люди більше цікавляться можливостями подорожей у межах своєї країни. Наразі існує чимало шляхів для розвитку. Один з них – популяризувати туризм в Україні. З проведених опитувань, результати яких відображені на рисунку 1.2 бачимо, що українці недостатньо обізнані про туристичні об’єкти.

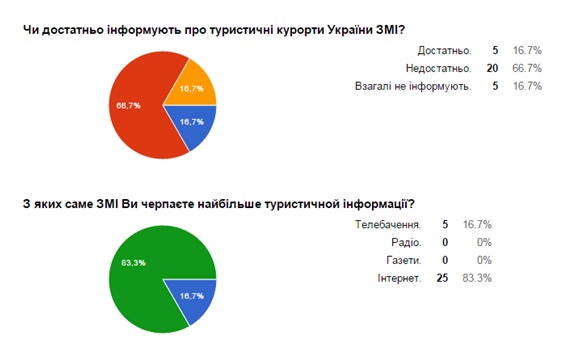


Рисунок 1.2 – Результати соціального опитування щодо туризму в Україні

Робимо висновок, що людям не вистачає інформації про туристичні об’єкти, а туристичному сегменту не вистачає якісної популяризації.

В даній роботі розробляється комплекс задач, основним завданням якого є популяризація туризму в Україні. Для цього створюється загальна мапа України з туристичними об’єктами та рекомендаційна система з персоналізованим підбором місць для користувача.

### **Опис процесу діяльності**

Розглянемо користувача та його дії за відсутності реалізованого комплексу задач.

Отже, середньостатистичний користувач – житель України, що планує дізнатись більше про свою країну. Звісно, він прагне побачити найцікавіше, а тому з готовністю розпитує друзів, знайомих про рекомендації та використовує мережу Інтернет для пошуку цікавої для нього інформації. Мережа Інтернет є надзвичайно великою, а серед кола знайомих можна і не знайти необхідної інформації. Отже, на рисунку 1.3 розглянемо процес діяльності користувача за відсутності нашої системи.

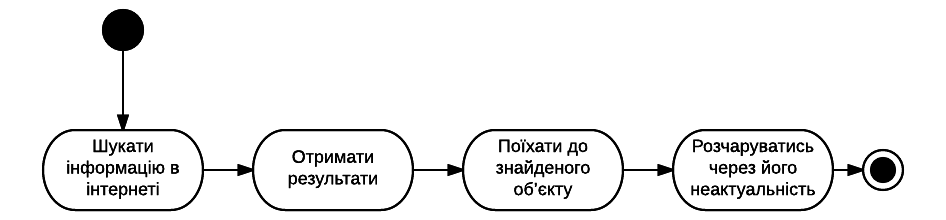


Рисунок 1.3 – Схема структурна діяльності до використання системи

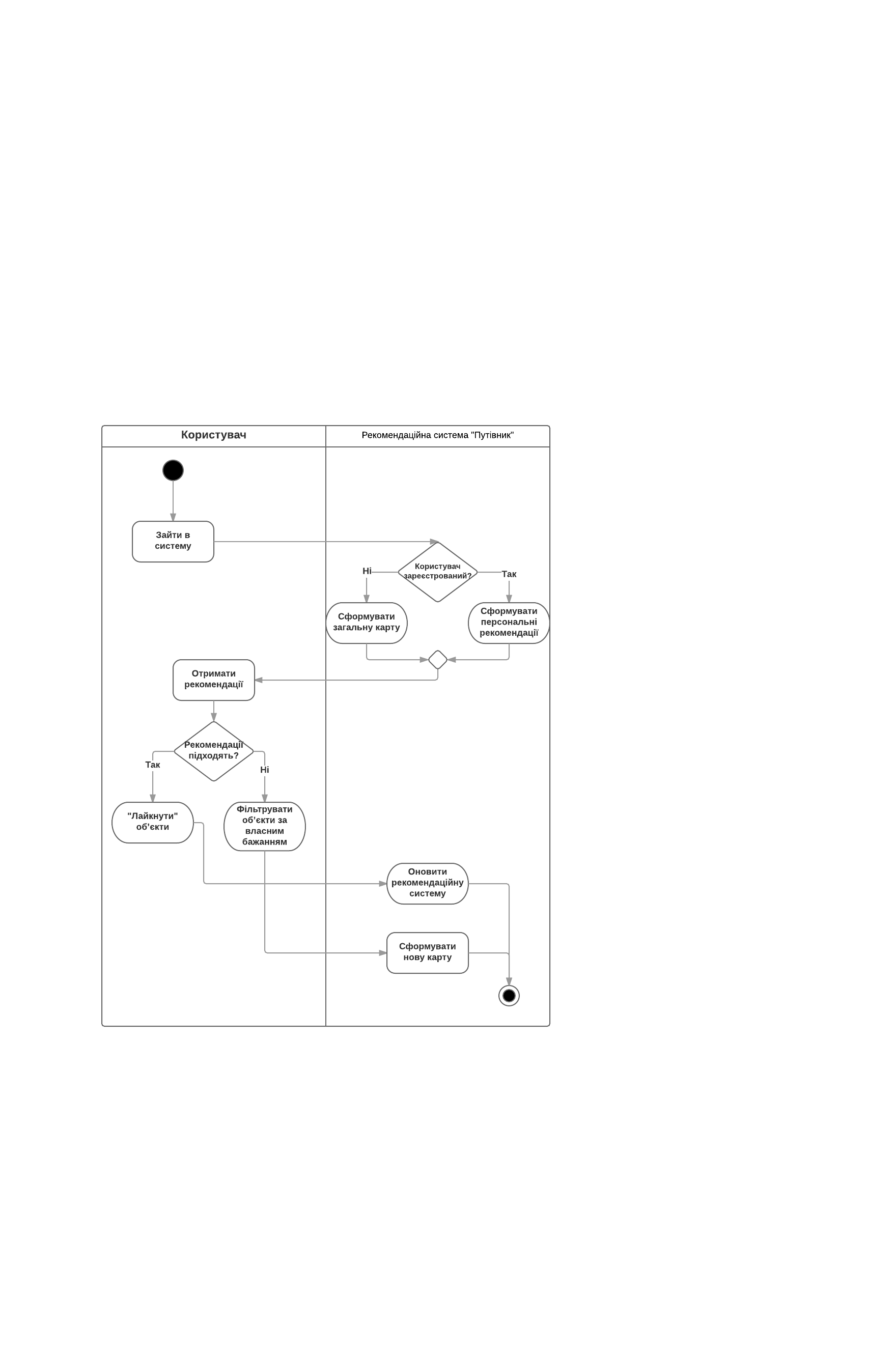
Опис процесу діяльності в умовах функціонування інформаційної системи зображений на рисунку 1.4.

Рисунок 1.4 – Схема структурна діяльності після використання системи

### **Опис функціональної моделі**

Для проектування діаграми використання спочатку необхідно визначити дійових осіб (акторів), а потім визначити, які дії у системі може виконувати кожен з акторів.

У системі виділяються наступні ролі:

* контент-менеджер;

Даній ролі дозволяється модерувати коментарі користувачів, редагувати інформацію про об’єкти, додавати нові об’єкти, виконувати пост-модерацію.

* користувач;

Користувач може коментувати об’єкти, створювати нові місця, що будуть доступними загальному колу користувачів. Для даної ролі доступно переглядати мапу, змінювати та зберігати налаштування, отримувати персоналізовані рекомендації.

* гість;

Дана роль характеризується тим, що вона є неавторизованою, тому вона не може додати нових місць чи коментарів до системи, проте доступ до мапи є вільним.

Дії, які можуть виконувати актори системи зображені на рисунку 1.5.

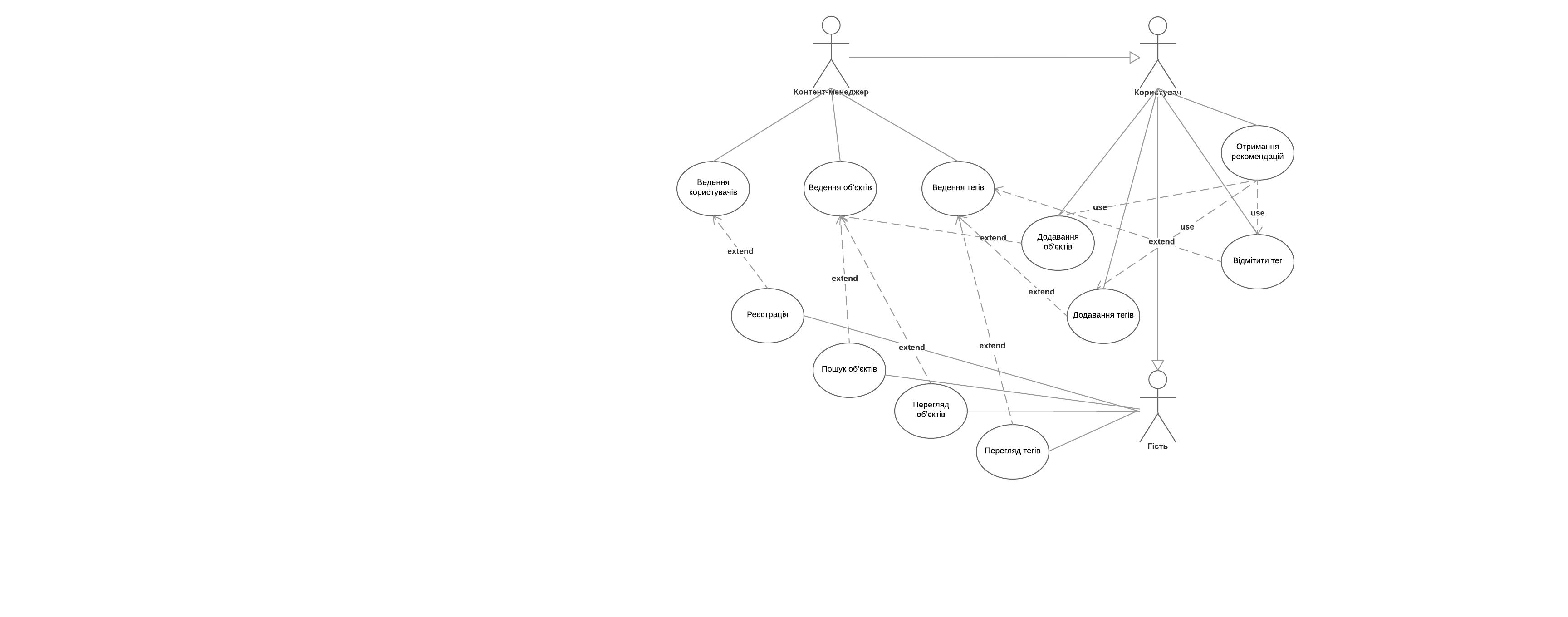


Рисунок 1.5 – Схема структурна варіантів використання

## **Огляд наявних аналогів**

В ході пошуку схожих вирішень було виявлено деякі підходи зі схожою функціональністю:

* блог про замки України <http://www.castles.com.ua> ;
* сайт про подорожі в Україні [www.kardash.kiev.ua/index.htm](http://www.kardash.kiev.ua/index.htm) ;
* Україна Інкогніта <http://ukrainaincognita.com> ;
* Моя країна прекрасна і незалежна <http://kraina.1plus1.ua> .

***Аналіз функціональності існуючих програмних продуктів***

Замки України.

Джерело доступу: <http://www.castles.com.ua/> .

Автор даного проекту, як стверджується в описі сайту, не є професійним істориком. Пише про ті фортеці та храми, які бачила на свої очі, які вдалося відвідати. На сайті є описи місць і чимала кількість зображень різних років. Тут немає наукових розвідок.

Даний сайт – тематичний блог, у якому зібрана інформація про такі туристичні об’єкти в Україні як замки, палаци та храми. Так як він надає користувачам інформацію щодо туристичних об’єктів, ми можемо назвати його в певній мірі аналогом нашого продукту. Проте наша система задовольняє набагато більше запитів користувачів, і як мінімум це те, що інформація надається по усіх туристичних об’єктах, а не лише замках.

Кардаш.

Джерело доступу: <http://www.kardash.kiev.ua/index.htm>.

На сайті є декілька розділів, з яких зручно знаходити необхідну інформацію.

Дана система є більш подібною до нашої. Так, вона охоплює більше туристичних об’єктів, а на головній сторінці є навіть карта, щоправда це просто статичне зображення, а, отже, у нашій системі все ще є сенс.

Україна Інкогніта.

Джерело доступу: <http://ukrainaincognita.com/>.

Сайт «Україна Інкогніта» – неприбутковий проект ентузіастів краєзнавства. Нині він один з найбільш наповнених, а можливо і найбільш наповнений інформацією краєзнавчий портал України. Щоденно кілька тисяч користувачів читають інформацію про Україну; туристи, краєзнавці, автомандрівники розробляють маршрути, послуговуючись даною інформацією. Загалом сайт вже містить понад 100 тисяч сторінок. Даний сайт містить значну кількість об’єктів, які зручно впорядковані, як це показано на рисунку 1.6.

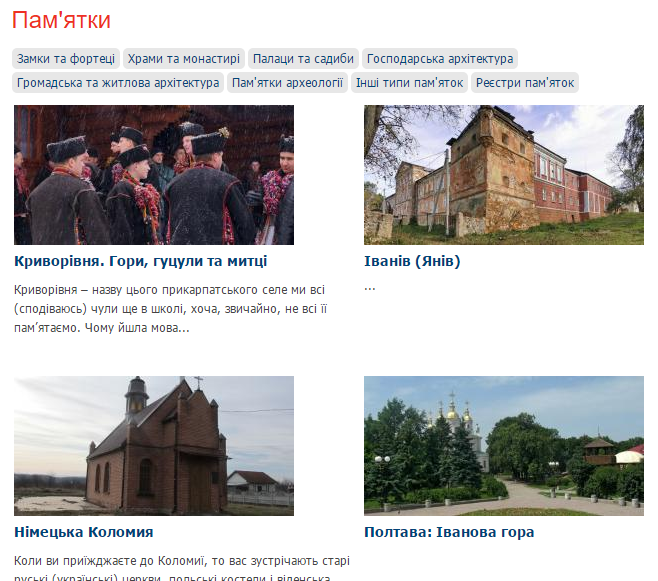


Рисунок 1.6 – Екранна копія системи Україна Інкогніта

У даній системі інформаціє є актуальною, карта - зручна для перегляду.

 Проте є суттєва відмінність, що полягає у взаємодії системи та користувача. Так, в представленому аналозі взаємодія полягає лише у дописі в блог, тоді як наша система надає можливість отримати персоналізовані рекомендації та змінювати загальний вигляд карти, додаючи нові об’єкти на мапу.

Моя країна прекрасна і незалежна 1+1.

Джерело доступу: <http://kraina.1plus1.ua/>.

Дана система є найбільш подібною до нашої. Моя країна прекрасна і незалежна – спільний проект телеканалу 1+1 та першої приватної броварні.

Протягом півроку вони знімають сюжети про те, як виглядає Україна з висоти пташиного польоту. Ці відео можна буде дивитися в ефірі телеканалу 1+1, а також на сайті проекту. Також можна додати на сайт свою історію про цікаве місце України – з фото або відео. Після премодерації історія потрапляє у список для голосування, і, якщо набере більше 100 голосів, додається на карту України. Дана система пропонує перегляд туристичних об’єктів, передбачає взаємодію з користувачем, дозволяє користувачам змінювати загальний вигляд мапи України. Мапа України є зручною та функціональною.

Проте така система є ознайомчою, не передбачає пошуку об’єктів, налаштування мапи, не може надати рекомендації.

В таблиці 1.1 наведено порівняльний опис характерних особливостей існуючих підходів.

Таблиця 1.1 – Порівняльна характеристика існуючих підходів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Можливості системи | Замки України | Кардаш | Україна Інкогніта | Прекрасна і Незалежна | Моя країна |
| Охоплення різних типів туристичних об’єктів | - | + | + | + | + |
| Візуалізація даних | - | - | + | + | + |
| Взаємодія з користувачами | + | - | - | + | + |
| Персоналізовані рекомендації | - | - | - | - | + |

## **Постановка задачі**

### **Призначення розробки**

Призначенням комплексу задач є знаходження персоналізованих рекомендацій для користувачів щодо туристичних місць, які варто відвідати.

### **Цілі та задачі розробки**

Головною метою комплексу задач є спрощення пошуку цікавих туристичних об’єктів в Україні.

Для досягнення поставленої мети мають бути вирішені такі задачі:

* ведення користувача;
* ведення тегів;
* ведення об’єктів;
* надання рекомендацій.

## **Висновок до розділу**

В даному розділі було проаналізовано предметне середовище та доцільність виконання роботи. Досліджено та описано процес діяльності та функціональну модель. Було розглянуто та вивчено існуючі аналоги, виконано порівняння з нашою системою. Виявлено призначення, цілі та задачі розробки.

# Інформаційне забезпечення

## **Вхідні дані**

Вхідні дані комплексу задач з планування подорожей та обміну досвідом для мандрівників України надходять з декількох джерел, а саме від:

* користувачів;
* Інтернет-джерел.

Тепер детально розглянемо, які саме дані надходять з цих джерел.

**Дані, які надходять від користувачів**. При реєстрації користувач надає таку інформацію:

* ім’я;
* електронна пошта.

В процесі користування системою користувач також може надавати наступні дані:

* тег, що відповідає об’єкту;
* оцінка тегу;
* дані про новий об’єкт.

**Дані, які надходять з Інтернет-джерел**. Для системи важливо обрати джерела, що будуть мати достовірні дані та можливість використання інформації (можливо з вказуванням посилання, звідки інформація взята). З відібраних джерел інформація буде агрегуватись у єдину систему з уніфікованим дизайном оформлення даних:

* назва;
* адреса;
* тип;
* опис об’єкту;
* тегування об’єкту.

## **Вихідні дані**

Вихідними даними є список об’єктів, тобто файл який містить інформацію про рекомендовані об’єкти. Список об’єктів складається з ідентифікаційних номерів об’єктів, що дає змогу зменшити кількість даних, що потребують передачі через Інтернет.

## **Опис структури бази даних**

У таблицях 2.1 – 2.4 наведена структура таблиць бази даних.

Таблиця 2.1 – Опис таблиці Users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Код** | **Опис** | **Тип даних** | **Довжина** | **Первинний** |
| objectId | ІД користувача | STRING\_ID | 500 | X |
| name | Ім’я користувача | STRING | 500 |  |
| password | Пароль користувача | STRING | 500 |  |
| email | Електронна пошта | STRING | 500 |  |

Таблиця 2.2 – Опис таблиці Likes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Код** | **Опис** | **Тип даних** | **Довжина** | **Первинний** |
| objectId | ІД запису вподобання | STRING\_ID | 500 | X |
| tag | ІД тегу | Data Relation: Tag |  | X |

Продовження таблиці 2.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| place | ІД об’єкту | Data Relation: Place |  | X |
| user | ІД користувача | Data Relation: Users |  | X |

Таблиця 2.3 – Опис таблиці Place

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| objectId | ІД об’єкту | STRING\_ID | 500 | X |
| name | назва об’єкту | STRING | 500 |  |
| description | опис об’єкту | STRING | 500 |  |
| location | місцезнаходження об’єкту | Geo Relation | 500 |  |

Таблиця 2.4 – Опис таблиці TAG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Код** | **Опис** | **Тип даних** | **Довжина** | **Первинний** |
| objectId | ІД тегу | STRING\_ID | 500 | X |
| name | назва тегу | STRING | 500 | X |

Структурна схема БД наведена у графічному матеріалі, лист <тут вказується номер листа>.

## **Висновок до розділу**

В даному розділі було розглянуто, які дані подаються на вхід, проаналізовано вихідні дані та їх формат. Описано структуру бази даних, наведено описи таблиць бази даних.

# Математичне забезпечення

## **Змістовна постановка задачі**

Задачею системи є персоналізований підбір об’єктів для користувачів. Системі відомо, які інтереси у кожного з користувачів та які місця вони вподобали. Таким чином можна використати алгоритм колаборативної фільтрації. Тобто на основі інтересів користувачів необхідно сформувати групи за інтересами. Припускаємо, що людям зі схожими інтересами подобаються схожі об’єкти і на основі даного припущення формуємо рекомендації.

У нас є система, в якій кожен користувач може знайти або додати місце, у якому він був, та оцінити місце. Оцінка місця в даному випадку бінарна:

* сподобалось (1);
* нема оцінки (0).

Для кожного користувача зберігається історія місць, які йому сподобались.

## **Математична постановка задачі**

Призначенням цієї задачі є пошук найбільш подібного користувача з точки зору інтересів.

Дано:

* – множина користувачів;
* – множина об'єктів.

Від нас вимагається надати k персональних рекомендацій для користувача :

|  |  |
| --- | --- |
|  | (3.1) |

## **Обґрунтування методу розв’язання**

Поставлена задача є задачею надання рекомендацій. Для її вирішення виділяють два основних підходи:

1. сontent-based (фільтрація за змістом):
2. користувачу рекомендуються об’єкти схожі на ті, що даний користувач вже вподобав;
3. схожість оцінюється за ознаками вмісту об’єктів;
4. велика залежність від предметної області, корисність рекомендацій обмежена;
5. сollaborative filtering (колаборативна фільтрація):
6. для рекомендацій використовується історія оцінок як самого користувача, так і групи схожих користувачів;
7. часто дає кращі і корисніші результати;
8. має свої проблеми(холодного старту).

Розглянувши нашу задачу, стає зрозуміло, що для нашої системи краще підходить метод колаборативної фільтрації. Адже даний підхід використовує попередню поведінку користувача або (що більш ефективно) поведінку схожих користувачів. Такі рекомендаційні системи формують рекомендації незалежно для кожного конкретного користувача на основі його попереднього досвіду використання системи, а також на основі поведінки інших користувачів.

Для вирішення задачі пошуку схожих користувачів можна використовувати методи, що полягають у визначенні коефіцієнта подібності.

Подібність двох користувачів визначатимемо за допомогою коефіцієнта Жаккара - бінарна міра подібності, запропонована Полем Жаккаром в 1901 році. Запропонований метод здобув поширення і нині використовується для оцінки подібності скінченних множин, в інформатиці, для пошуку подібних документів, плагіату, тощо.

## **Опис методів розв’язання**

Колаборативна фільтрація складається з двох основних етапів:

* пошук групи схожих користувачів;
* вибір рекомендацій з вподобаних об’єктів серед групи схожих користувачів.

З кожним користувачем ми асоціюємо множину відвіданих місць pi. Саме за цією множиною ми знаходитимемо схожих користувачів. Для цього визначатимемо коефіцієнт Жаккара за формулою 3.2.

|  |  |
| --- | --- |
| , | (3.2) |

- коефіцієнт, що характеризує подібність множин, може набувати значення з проміжку [0, 1], причому:

* 0 – множини абсолютно не подібні;
* 1 – множини повністю подібні.

Обраний користувач порівнюється з рештою користувачів, знаходяться коефіцієнти подібності та визначається топ найбільш схожих користувачів. Тепер виберемо усі улюблені місця подібних користувачів, з них оберемо лише ті, де користувач ще не був, та визначимо рекомендації з отриманої множини місць. Такий підхід називається методом n найближчих сусідів (N-top user-based filtering). Рекомендація місця здійснюватиметься на основі оцінки, що показує наскільки ймовірно, що об’єкт сподобається користувачу. Для цього необхідно просумувати коефіцієнти Жаккара кожного схожого користувача, який вподобав дане місце, та нормалізувати його на число усіх схожих користувачів за формулою 3.3.

|  |  |
| --- | --- |
| , | (3.3) |

де i - кількість схожих користувачів і схожі користувачі, що вподобали місце відповідно.

Рекомендації - це k таких об’єктів, що їм відповідає найбільша оцінка .

Узагальнимо описаний вище метод у вигляді покрокового алгоритму для знаходження рекомендацій користувачу j :

**Крок 1.** Відсортувати усіх користувачів у порядку спадання коефіцієнта Жаккара – схожості з j за формулою (3.2);

**Крок 2.** Для кожного з перших схожих користувачів:

**Крок 2.1.** Додати усі вподобані місця, такі, що j ще не оцінив, до списку рекомендацій;

**Крок 2.2.** Для кожного місця зі списку рекомендацій обчислити за формулою (3.3);

**Крок 3.** Відсортувати список рекомендацій за спаданням ;

**Крок 4.** Рекомендувати користувачу k перших рекомендацій зі списку рекомендацій.

## **Висновок до розділу**

В даному розділі була сформульована змістовна та математична постановки задачі пошуку подібного за інтересами користувача та визначено, що вона належить до класу задач надання рекомендацій. Наведено детальний опис алгоритмів розв’язання задачі методом колаборативної фільтрації.

# Програмне та технічне забезпечення

## **Засоби розробки**

Комплекс задач має клієнт-серверну архітектуру. Тож необхідна розробка як сервера, так і клієнтської частини.

Для створення серверної частини використовується сервіс Backendless.

Backendless пропонує забезпечення найбільш поширених функціональних можливостей на стороні сервера. Надані послуги включають в себе реєстрацію користувачів і аутентифікацю, збереження даних, повідомлення, геолокації, генерування коду і багато іншого.

Backendless використовує базу даних MariaDB.

Для реалізації комплексу задач «Моя країна» були використані наступні сервіси:

* UserService;
* FileService;
* Geo Location;
* Files;
* Data Management;
* Business Logic;
* Registration and Login.

Для вирішення задач рекомендацій були реалізовані модулі сервісу Бізнес Логіки на платформі Node.js. Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript). Платформа Node.js перетворила мову JavaScript, що в основному використовувалась в браузерах, на мову загального використання з великою спільнотою розробників. Для управління модулями використовується пакетний менеджер npm (node package manager).

Для створення клієнтської частини використовується середовище розробки WebStorm Idea та такі технології як HTML/CSS/JavaScript, що вже давно стали стандартом для веб застосунків.

Для розробки клієнтської сторони веб-додатку використовуються технології Angular 2, які дозволяють не лише зменшити швидкість візуалізації елементів інтерфейсу та контенту браузером, а й надають зручні інструменти для розміщення інформації на сторінках веб-додатків.

AngularJS 2 спроектований з переконанням, що декларативне програмування найкраще пасує для побудови інтерфейсів користувача та опису програмних компонентів, в той час як імперативне програмування пасує для опису бізнес-логіки. Фреймворк адаптує та розширює традиційний HTML, щоб забезпечити двосторонню прив'язку даних для динамічного контенту, що дозволяє автоматично синхронізувати модель та вид. У результаті AngularJS 2 зменшує роль DOM-маніпуляцій з метою підвищення продуктивності і спрощення тестування.

Додатки на Angular 2 складаються з компонентів і представляють з себе дерево. Angular 2 підтримує мови програмування ECMASCRIPT 5, ECMASCRIPT 6, TypeScript та Dart. При розробці даного комплексу задач використовувалась мова програмування TypeScript.

Традиційно JavaScript фокусується на функціях та прототипному успадкуванню в якості основного засобу створення багаторазових компонентів, але більш комфортним є об’єктно-орієнтований підхід, де класи успадковують функціонал та об’єкти будуються на основі цих класів. У TypeScript розробникам дозволяється використовувати можливість будувати свої програми використовуючи ООП та компілювати код у JavaScript, який буде працювати на всіх основних браузерах та платформах.

Для збірки клієнтської частини використовується інструмент webpack. Це один із найпотужніших та водночас гнучких інструментів для управління залежностями.

Для відображення мапи використовується Leaflet – це популярна Javascript бібліотека з відкритим кодом для побудови веб-картографічних додатків.

## **Вимоги до технічного забезпечення**

Для коректної роботи даного продукту на стороні сервера мають бути такі технічні характеристики:

* 1. комп’ютер:

1. процесор з частотою - не менше 1 ГГц;
2. об’єм оперативної пам’яті - не менше 4 ГБ;
   1. програмне забезпечення:
3. операційна система Linux;
4. середовище розробки WebStorm Idea;
5. менеджер пакетів npm;
6. компілятор typescript;
7. компонувальник webpack;
8. браузер з підтримкою JavaScript;
   1. комп’ютерна периферія:
9. монітор;
10. мишка;
11. клавіатура.

## **Архітектура програмного забезпечення**

### **Діаграма класів**

На рисунку 4.1 представлена структурна схема класів, які відповідають за виконання таких функцій програми, як відображення блоку інформації про об'єкт, створення нового місця, редагування об'єктів, створення нових тегів, вподобання існуючих тегів, авторизація та реєстрація користувача.

Діаграма містить 17 класів, а саме:

* AllPlaces – клас, що відповідає за відображення групи місць та переходу до створення нового місця;
* PlaceDetail – клас, що відповідає за відображення інформаційного блоку про об’єкт;
* NewPlace – клас, що відповідає за створення нового об’єкту;
* Tags – клас, що відповідає за групу тегів, прив’язаних до конкретного об’єкту;
* AddTag – клас, що відповідає за створення нового тегу;
* TagDisplay – клас, що відповідає за відображення тегу та функцію вподобання тегу;
* FullPlace – клас, що відповідає за відображення сторінки об’єкту, його редагування, видалення;
* Auth – клас, що відповідає за аутентифікацію в програмному забезпеченні;
* Login – клас, що відповідає за авторизацію користувача;
* Register – клас, що відповідає за реєстрацію користувача;
* PlaceService – клас, що відповідає за з’єднання з сервером в роботі з об’єктами та за роботу з об’єктами;
* LikeService – клас, що відповідає за роботу з вподобаннями та тегами та зв’язку з сервером щодо вподобань та тегів;
* DisplayTag – клас, що відображає сутність тегу, прикріпленого за об’єктом;
* Like – клас, що є сутністю вподобання тега;
* Place – клас, що відображає об’єкт;
* Tag – клас, що відображає тег.

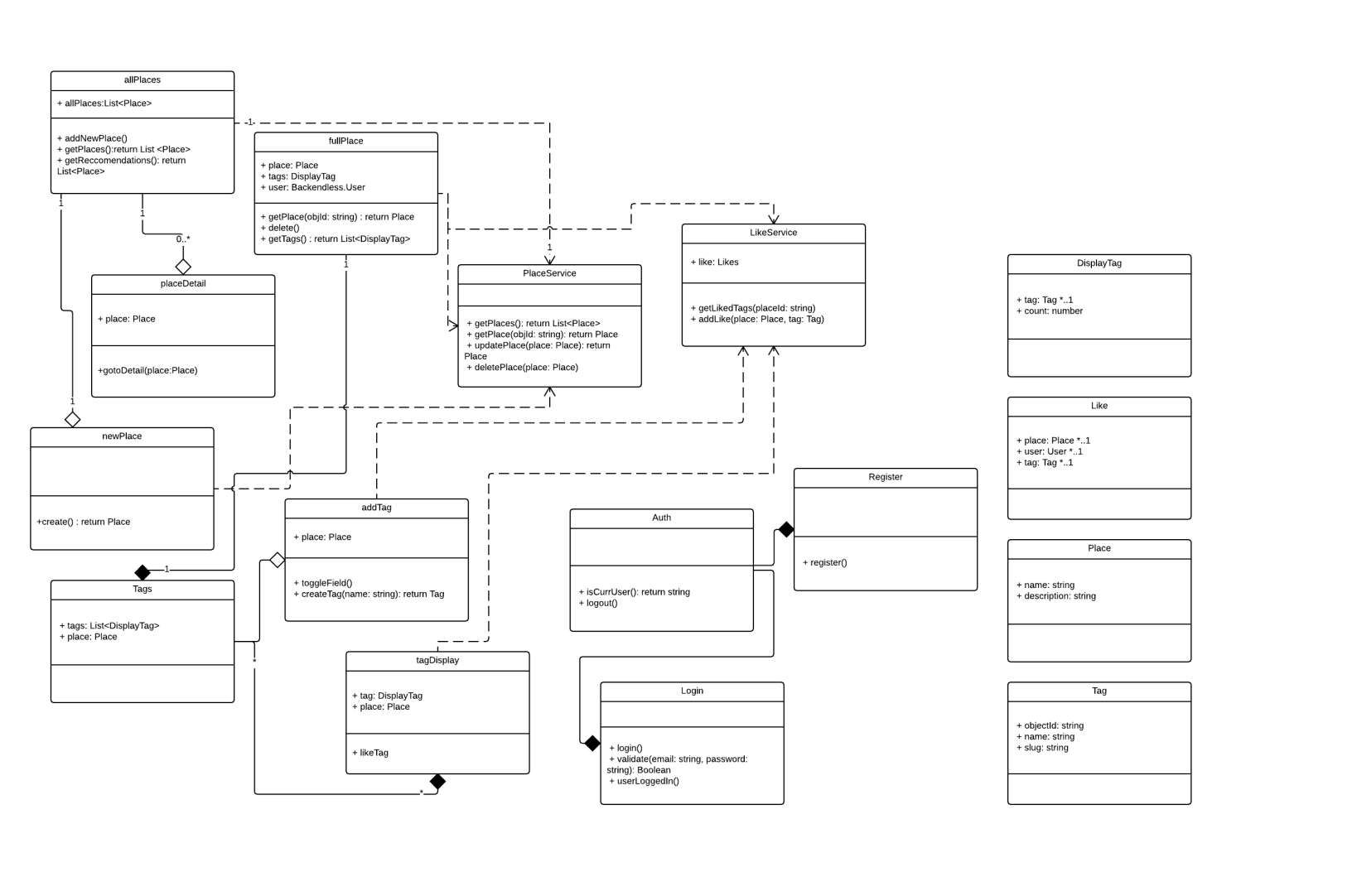


Рис. 4.1 – Схема структурна класів програмного забезпечення

### **Діаграма послідовності**

Нижче представлена типова послідовність дій, які виконуються при наданні рекомендацій для користувача щодо туристичних об’єктів. Для виконання даної задачі клас AllPlaces, що відповідає за відображення рекомендацій, надсилає запит до модулю Backendless.UserService та отримує дані користувача, якому він повинен надати рекомендації. Після цього він звертається до модуля бізнес-логіки Backendless.RecommendSystem, передає даному модулю дані про користувача та запитує рекомендації. Модуль Backendless.RecommendSystem містить алгоритми визначення рекомендацій, він виконує дані алгоритми для заданого користувача та повертає результати класу AllPlaces у вигляді списку об’єктів для рекомендацій. Клас AllPlaces відображає знайдені результати. Виявлена послідовність зображена на рисунку 4.2.

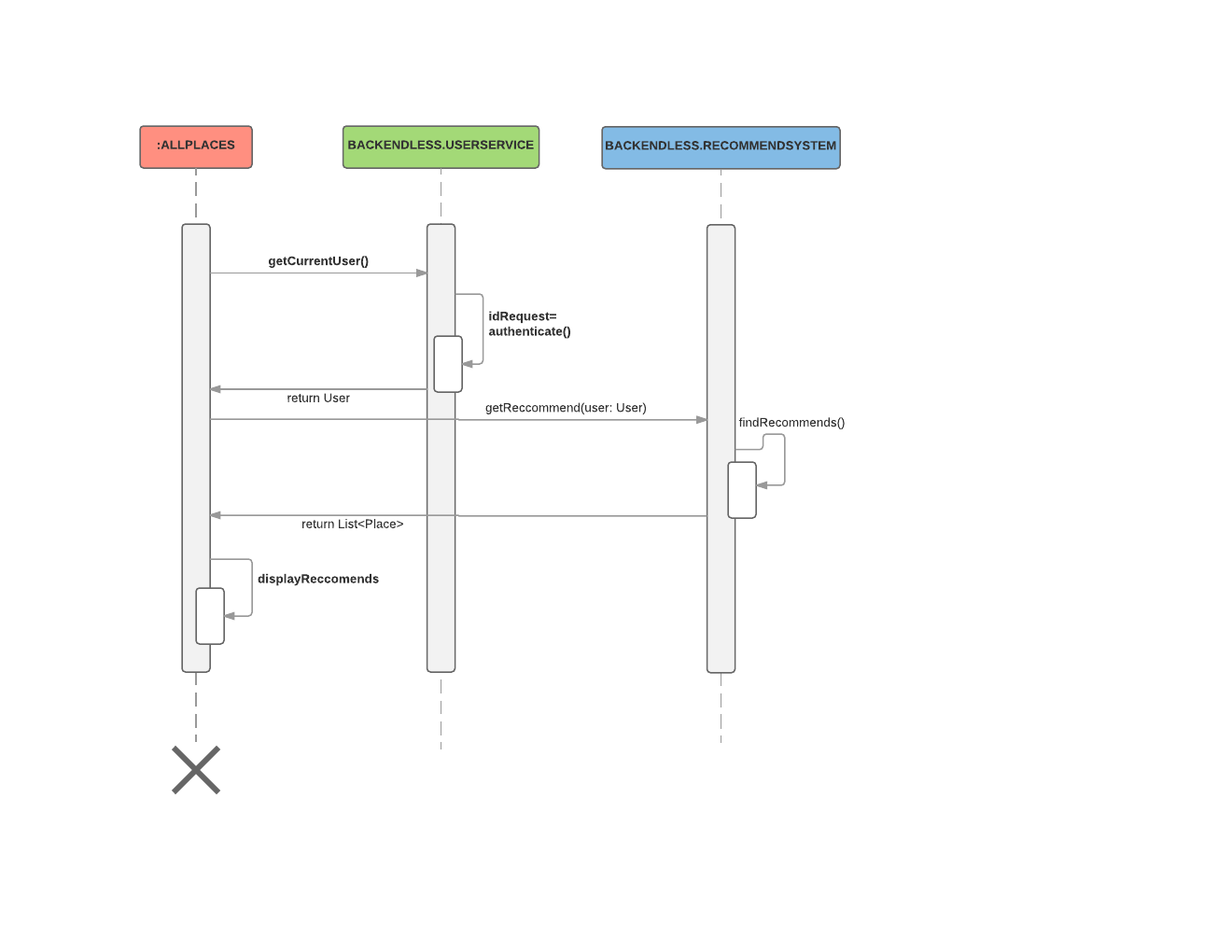


Рисунок 4.2 – Схема структурна послідовності для прецеденту  *Надання рекомендацій для користувача щодо туристичних об’єктів*

### **Діаграма розгортання**

Діаграма розгортання відображає робочі екземпляри компонент. Вона відображає обчислювальні вузли під час роботи програми. Розроблений комплекс задач має клієнт-серверну архітектуру. Тому діаграма розгортання складається з трьох основних блоків. Перший(найнижчий) – це комп’ютер користувача з встановленим веб-браузером. Веб-браузер з’єднується з веб-сервером, що містить презентаційний шар застосунку на основі HTTP. Веб-застосунок з’єднується з сервером на основі TCP/IP. На рисунку 4.3 зображена діаграма розгортання.

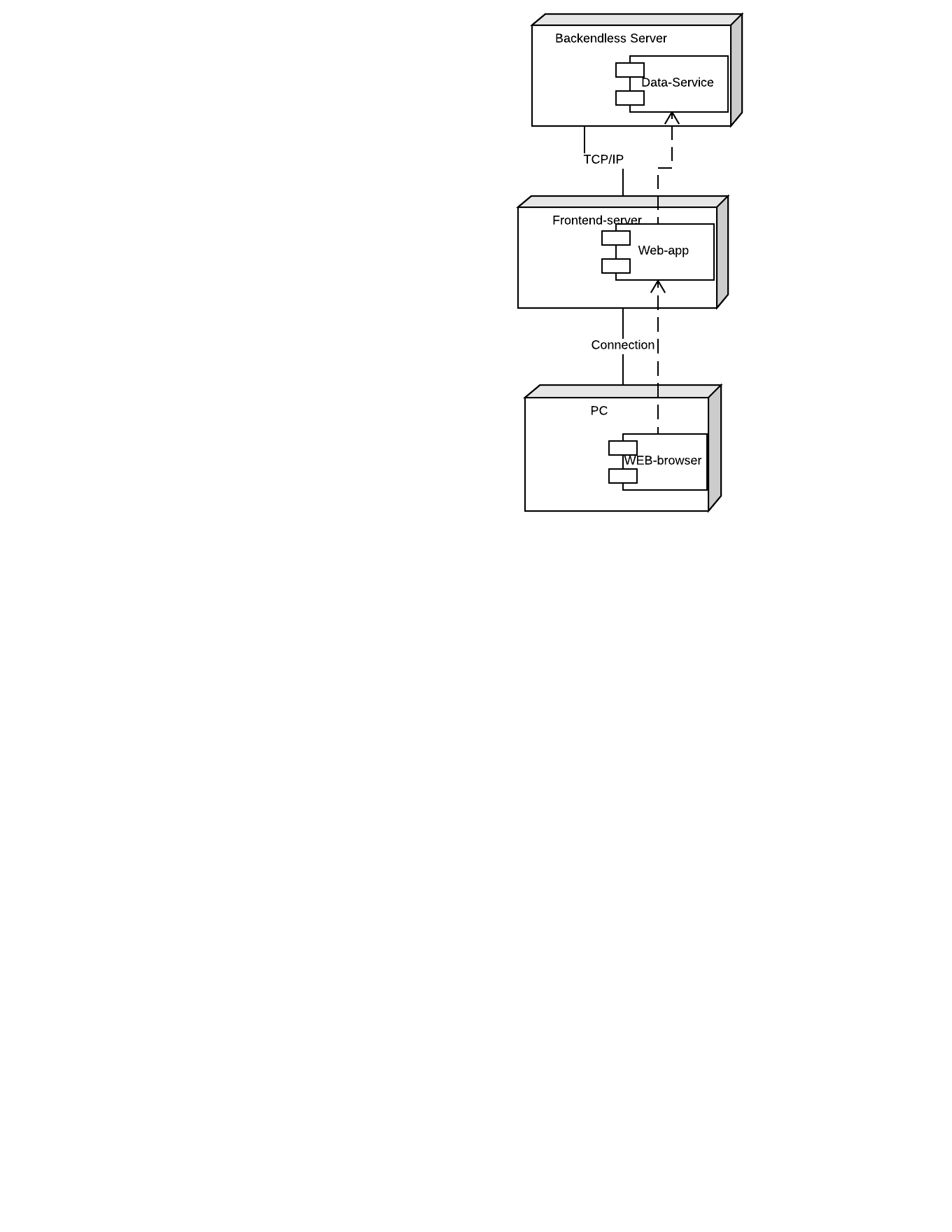


Рис.4.3 – Схема структурна розгортання

### **Специфікації функцій**

В таблиці 4.1 наведена специфікація основних функцій системи.

Таблиця 4.1 – Специфікація функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Клас | Метод | Призначення | Повертає результат | Список  параметрів | Семантика параметрів |
| AllPlaces | getPlaces() | Отримує всі місця з серверу | - | - | - |
| AllPlaces | addNewPlace() | Перенаправляє на створення нового місця | - | - | - |
| Auth | isCurrUser() | Перевіряє, чи користувач авторизований | Bool | - | - |

Продовження таблиці 4.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Auth | logout() | завершення сесії користувача | - | - | - |
| Register | register() | реєстрація користувача | - | - | - |
| Register | valid() | перевіряє, чи введені дані для реєстрації валідні | Bool |  |  |
| Login | login() | авторизація користувача | - | - | - |
| Login | valid() | перевіряє, чи введені дані для авторизації валідні | - | - | - |
| Login | userLoggedIn() | перенаправлення після авторизації | - | - | - |
| FullPlace | getPlace(objId) | дістати місце за ід | Place | objId | ід об'єкту |
| FullPlace | delete() | видалити місце | - | - | - |
| FullPlace | editDescription() | редагувати опис | - | - | - |
| FullPlace | editTitle() | редагувати назву | - | - | - |
| FullPlace | saveChanges() | зберегти зміни | - | - | - |
| FullPlace | toggleEditing() | перемкнути режим редагування | - | - | - |
| FullPlace | getTags() | дістати всі теги для даного місця | - | - | - |
| FullPlace | initMap() | ініціалізувати мапу для даного місця | - | - | - |
| Map | displayMap() | відобразити загальну мапу | - | - | - |

Продовження таблиці 4.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Map | onEachFeature() | для кожної області відслідковувати події | - | - | - |
| Map | createMask() | створити маску для затемнення областей навколо України | - | - | - |
| Map | createDistricts() | ініціалізувати області | - | - | - |
| Map | createInfo() | створити інформаційний тег | - | - | - |
| Map | addMarkers() | додати місця на мапу | - | - | - |
| NewPlace | create() | створити нове місце | - | - | - |
| PlaceDetail | gotoDetail() | перенаправлення на повний опис місця | - | - | - |
| AllTags | toggleField() | ввімкнути режим додання тегів | - | - | - |
| AllTags | saveTag() | зберегти тег | - | - | - |
| AllTags | search() | виконати пошук існуючих тегів | - | - | - |
| TagDisplay | likeTag() | вподобати тег | - | - | - |
| LikeService | getDisplayTag(place:Place, tag: Tag) | дістати з серверу кількість вподобань для тегу на місце | Promise: Like[] | place:Place, tag: Tag | місце, до якого доданий тег;  тег |

Продовження таблиці 4.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LikeService | getLikedTags(placeID:string) | дістати всі вподобання тегів для місця | Promise: Like[] | placeID:string | ід місця |
| LikeService | addLike(place: Place, tag: Tag) | вподобати тег на місце | Promise: Like | place: Place, tag: Tag | місце, до якого доданий тег;  тег |
| LikeService | getTagByName(name: string) | дістати тег за ім’ям | Promise: Tag | name: string | ім’я тегу |
| LikeService | createNewTag(tagName: string) | створити новий тег | Promise: Tag | tagName: string | ім’я нового тегу |
| LikeService | likeNewTag(tagName: string, place:Place) | вподобати новий тег | Promise: Like | tagName: string, place:Place | ім’я нового тегу; місце, до якого він доданий |
| PlaceService | createPlace(place: Place) | зберегти нове місце | Promise: Place | place: Place | місце |
| PlaceService | getPlaces() | дістати усі місця | Promise:Places[] | - | - |
| PlaceService | getPlace(objId: string) | дістати місце за ід | Promise: Place | objId: string | ід місця |
| PlaceService | deletePlace(place:Place) | видалити місце |  | place:Place | місце |

Продовження таблиці 4.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PlaceService | updatePlace(place:Place) | зберегти зміни місця | Promise: Place | place:Place | місце зі змінами |

## **Висновок**

В даному розділі було представлено засоби розробки програмного забезпечення, надано вимоги до технічного забезпечення системи. Було розглянуто архітектуру програмного забезпечення, побудовано та описано діаграму класів, проаналізовано послідовність виконання прецеденту щодо надання рекомендацій для користувача по туристичних об’єктах, розроблено та представлено діаграму компонентів, задокументовано специфікації функцій.

# Технологічний розділ

## **Керівництво користувача**

З будь-якої сторінки сайту користувачу доступне навігаційне меню, що знаходиться зверху сторінки. З даного навігаційного меню користувачу доступні кнопки переходу для перегляду інформації про продукт, перегляду усіх місць, перегляду мапи України, авторизації, реєстрації.

На головній сторінці сайту відображена інформація про продукт та розробників.

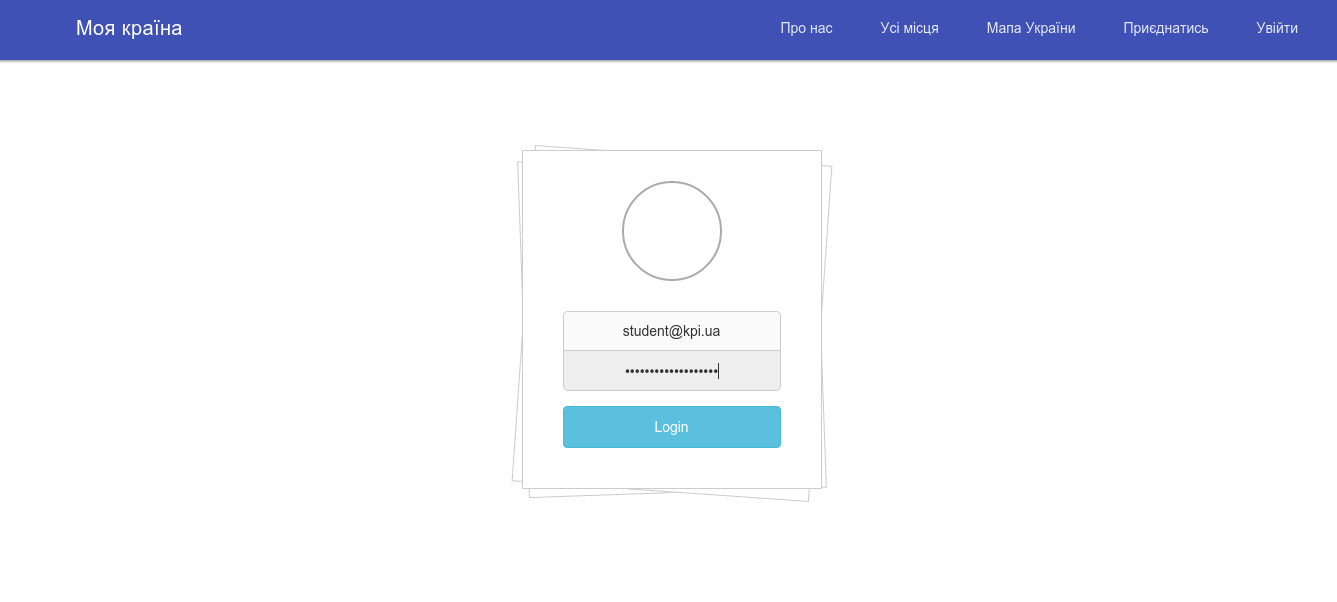
Для авторизації необхідно натиснути кнопку «Авторизація». На сторінці, що з’явилась, необхідно ввести електронну пошту та пароль, на які був зареєстрований акаунт, як показано на рисунку 5.2.

Рисунок 5.2 – Сторінка авторизації користувача

Після введення даних необхідно натиснути кнопку «Авторизуватись». При успішній реєстрації користувача буде перенаправлено на головну сторінку, на навігаційній панелі замість кнопок «Авторизація» та «Реєстрація» будуть доступні кнопки «Профіль» та «Вийти».

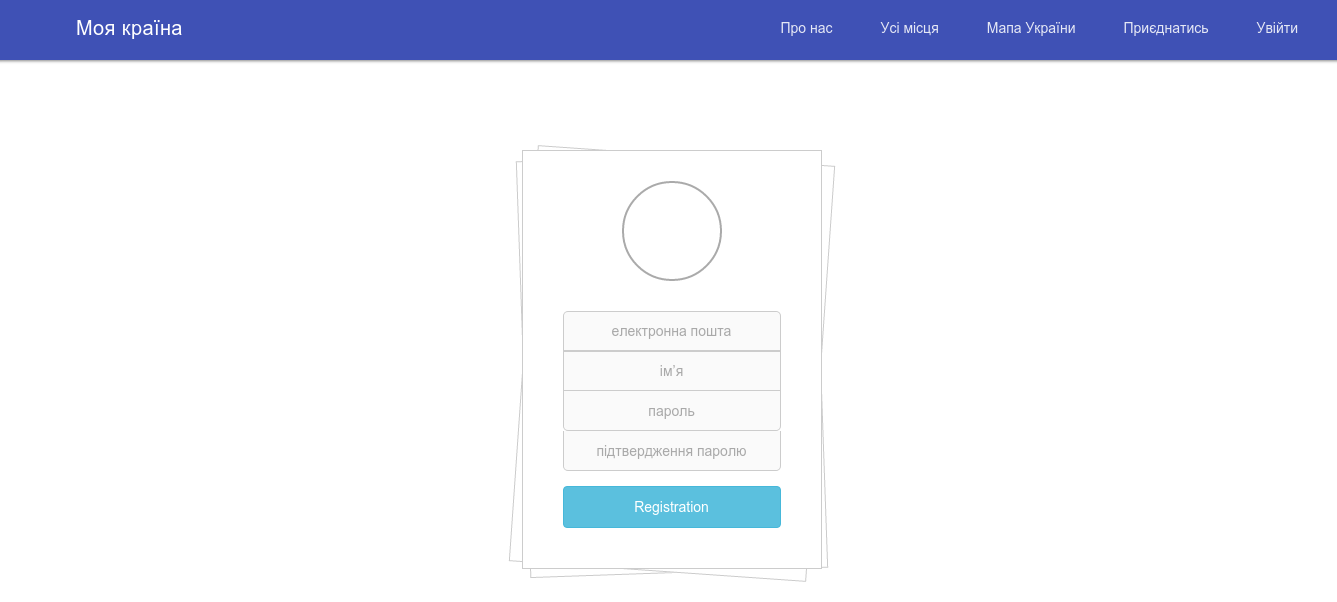
Для реєстрації нового користувача необхідно натиснути кнопку «Реєстрація». На сторінці, що відкрилась, необхідно ввести електронну пошту, ім’я користувача, пароль та підтвердження паролю, як показано на рисунку 5.3.

Рисунок 5.3 – Сторінка реєстрації нового користувача

Для перегляду туристичних об’єктів необхідно натиснути кнопку «Усі місця». На сторінці, що відкрилась, доступно поле для пошуку об’єкту, кнопка для створення нового об’єкту, список рекомендованих об’єктів. У разі, якщо користувач неавторизований, то на сторінці відображаються 12 найпопулярніших об’єктів, інакше – 12 рекомендованих об’єктів. Після блоку об’єктів є кнопка «Дивитись ще», як це зображено на рисунку 5.4.

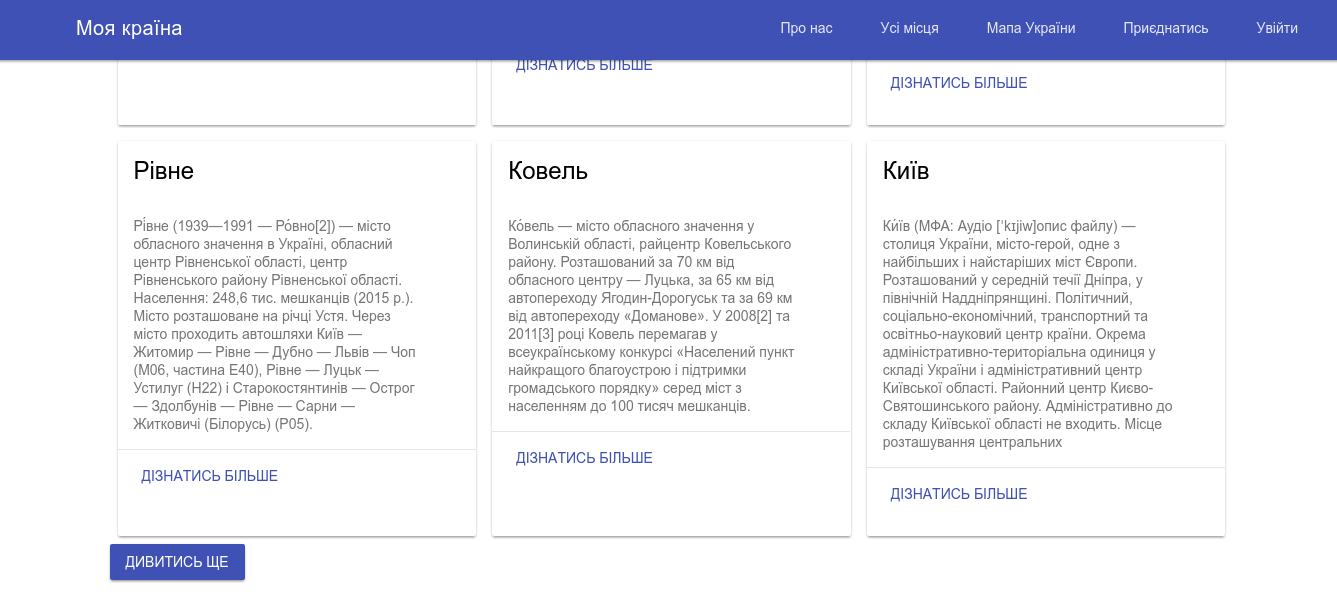


Рисунок 5.4 – Сторінка перегляду усіх місць

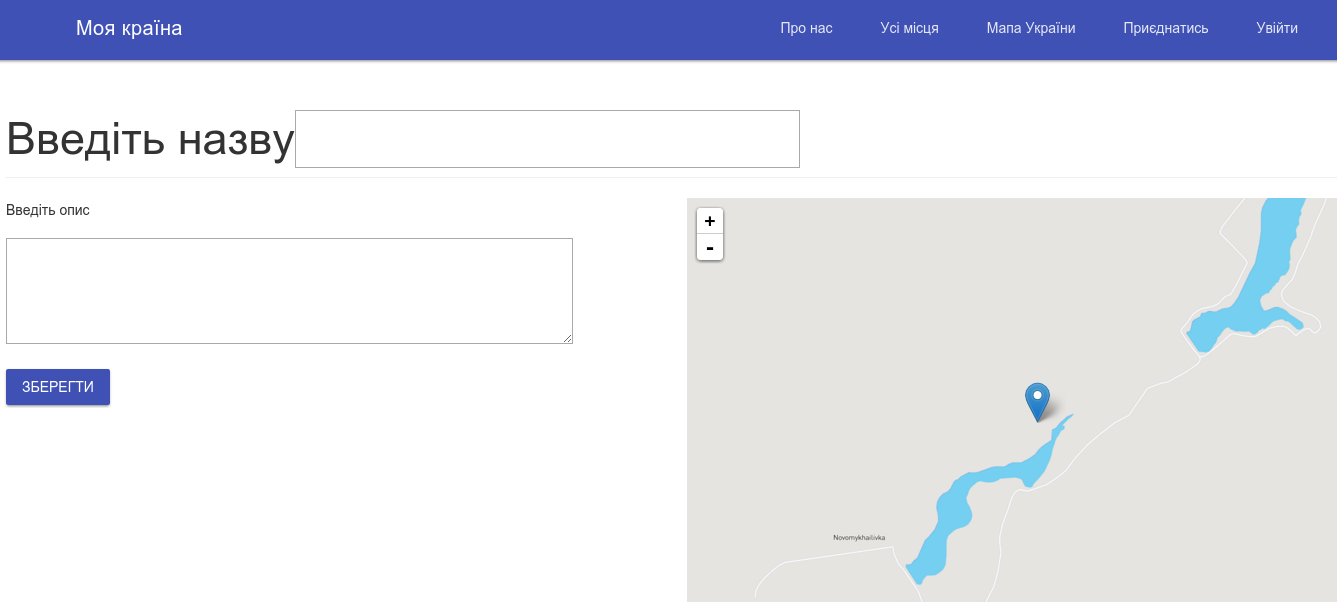
Для створення нового об’єкту необхідно перейти до сторінки перегляду усіх місць та натиснути кнопку «Додати місце». Якщо користувач неавторизований, то відкриється повідомлення з пропозицією авторизуватись або зареєструватись. Якщо ж користувач авторизований, то відкриється сторінка створення нового місця, як зображено на рисунку 5.5.

Рисунок 5.5 – Сторінка створення нового місця

На сторінці, що з’явилась, необхідно ввести назву місця, опис місця, завантажити зображення та обрати на карті розташування даного місця. Після чого натиснути кнопку «Створити».

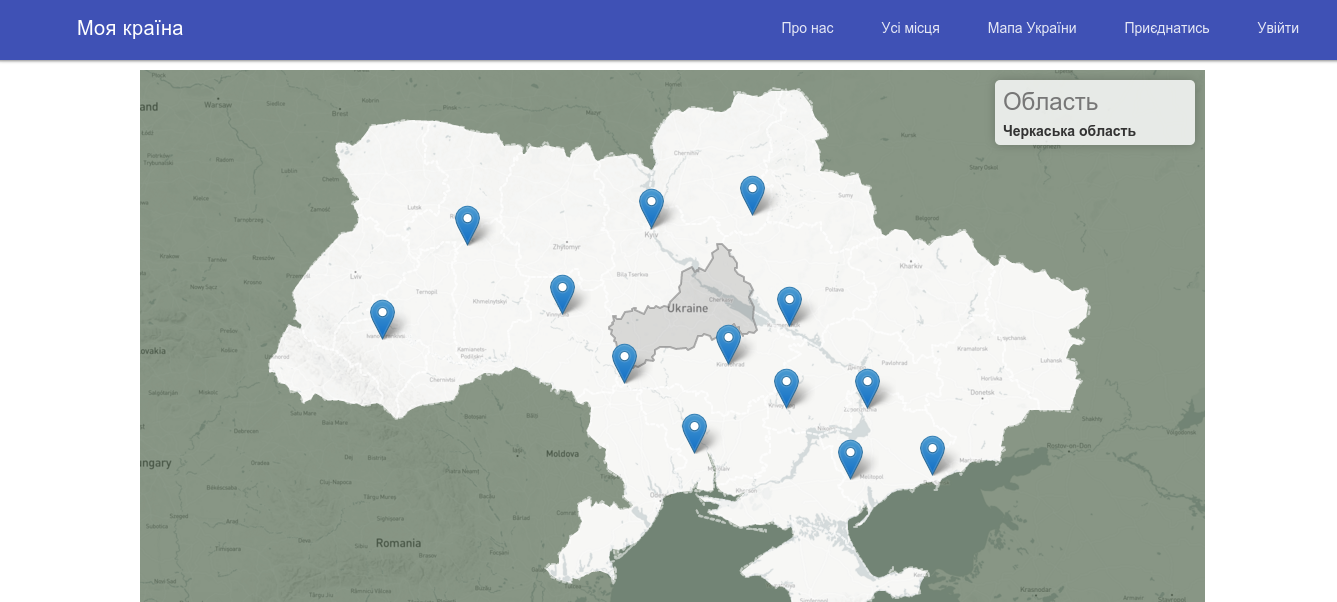
Для перегляду мапи України необхідно натиснути кнопку «Мапа України» з навігаційного меню. Після чого відкриється сторінка з мапою України. На даній карті маркерами позначені рекомендовані для відвідання об’єкти. При наведенні курсора мишки на мапу, відповідна область України підсвічується, у верхньому правому кутку відображається коротка інформація по даній області – назва області і кількість туристичних об’єктів даної області, як це показано на рисунку 5.6.

Рисунок 5.5 – Сторінка перегляду мапи України

При натисненні на область України, карта збільшується та для перегляду стає доступна більша кількість об’єктів саме з даної області. При натисненні на маркер з’являється коротка інформація про даний об’єкт та посилання на повний опис об’єкту. Вибір певного об’єкту зображений на рисунку 5.6.

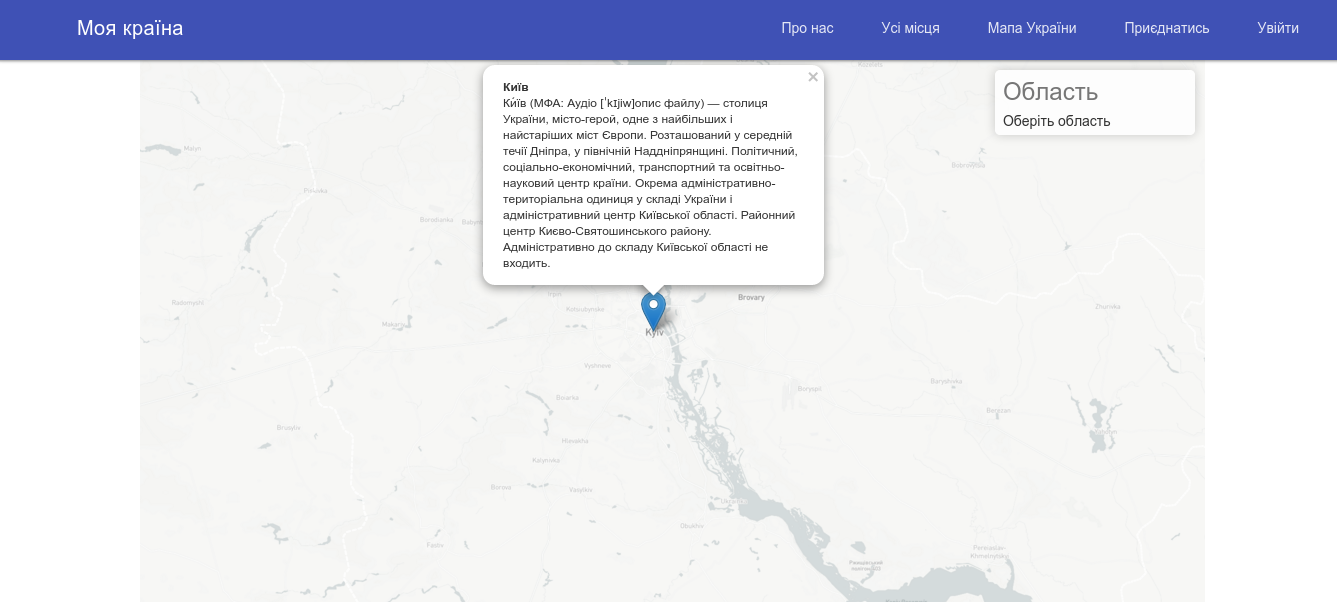


Рисунок 5.6 – Сторінка перегляду мапи України з відображенням об’єкту

На сторінці повного перегляду об’єкту міститься назва, опис, фото об’єкту та мапа, що відображає його розташування. Під описом об’єкту зображаються теги, що характеризують даний об’єкт. Кожен тег містить назву тегу та число, що позначає, скільки користувачів відмітило даний тег, як такий, що підходить для даного об’єкту. Якщо користувач неавторизований, то для нього доступний лише перегляд. Якщо користувач авторизований, то для нього доступна можливість відмітити та створити тег. Для вподобання тегу необхідно натиснути нього, кількість відміток на даному тегу збільшиться на 1. При повторному натисненні поставлена відмітка видаляється.

Для створення тегу необхідно натиснути на кнопку «+», після чого з’явиться поле для вводу нового тегу. Розпочавши вводити тег, для користувача виводяться підкази, які теги вже існують. Створивши новий тег необхідно натиснути кнопку «Enter», після чого тег буде створений та автоматично додасться відмітка, що ви вподобали даний тег. Робота з тегами зображена на рисунку 5.7.

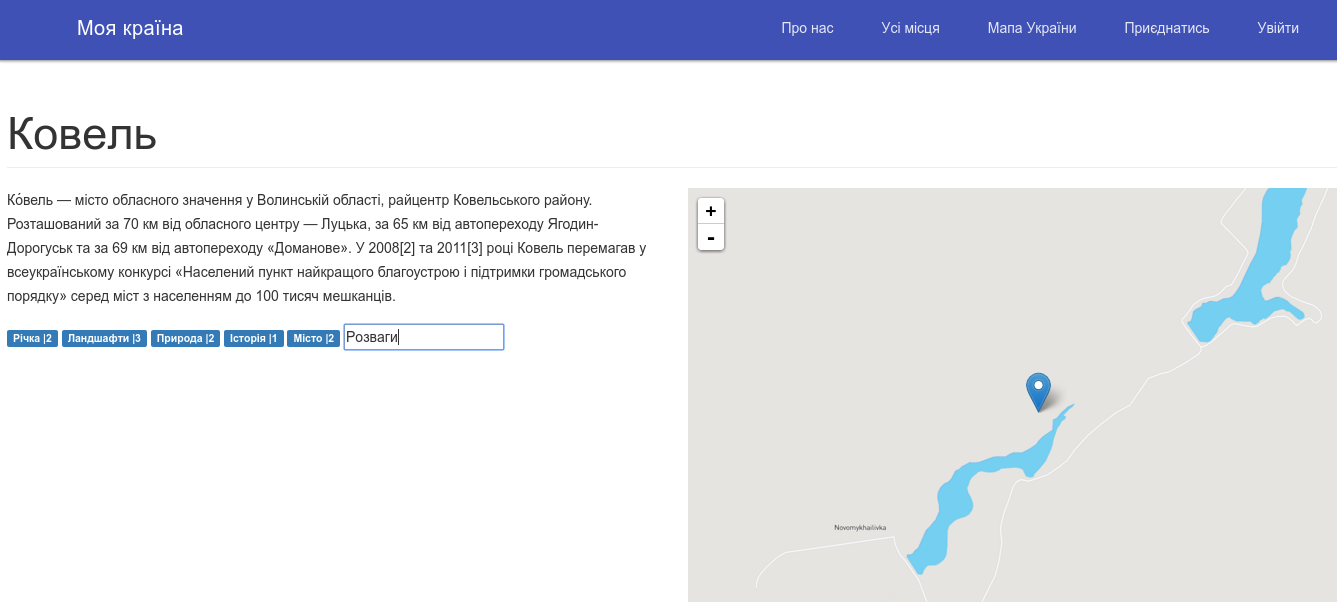


Рисунок 5.7 – Робота з тегами

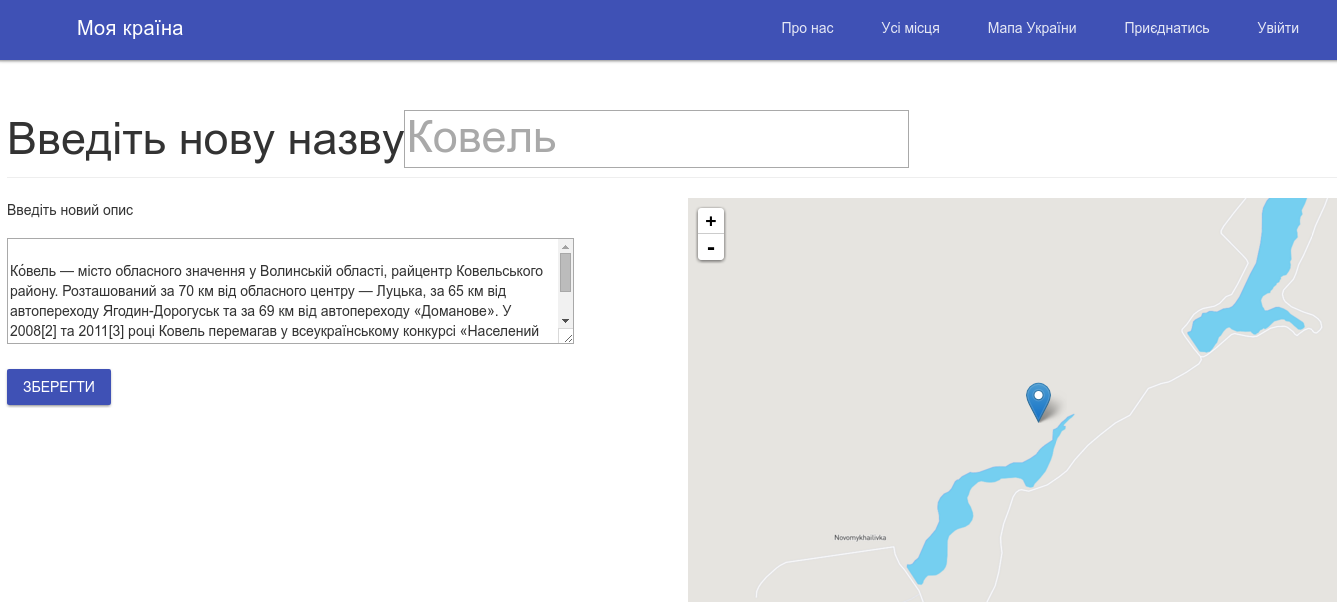
Якщо користувач авторизований і він є власником створеного об’єкту(тобто він додав інформацію про об’єкт на мапу), то для нього доступна функція редагування назви, опису, розташування та зображення об’єкту, як це зображено на рисунку 5.8.

Рисунок 5.8 – Редагування об’єкту

Для видалення об’єкта з мапи України необхідно бути власником даного об’єкту, перейти до його повної сторінки та натисну кнопку «Видалити інформацію».

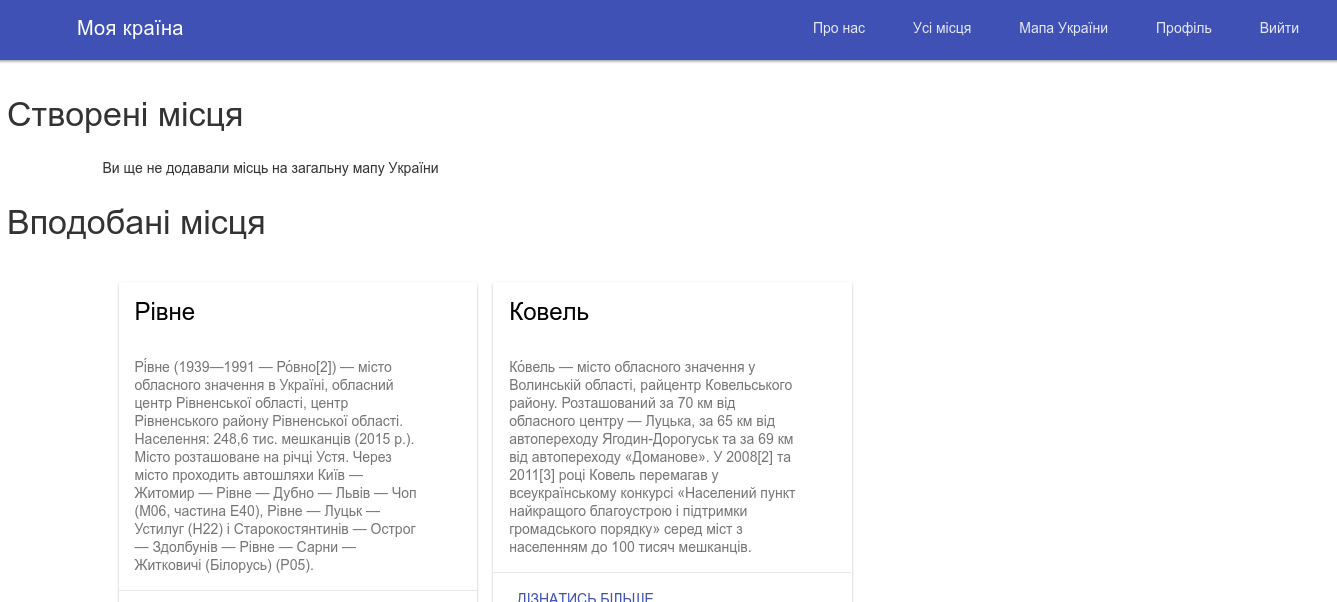
Для перегляду профілю користувача необхідно натиснути на кнопку «Профіль» з навігаційного меню. Після цього відкриється сторінка профілю користувача, що зображена на рисунку 5.9.

Рисунок 5.9 – Профіль користувача

З даної сторінки користувач може перейти до особистої інформації, створених та позначених об’єктів.

Для завершення роботи необхідно натиснути кнопку «Вийти» з навігаційного меню.

## **Випробування програмного продукту**

В цьому підрозділі наведено опис тестів і порядок їх виконання для перевірки відповідності програмного забезпечення комплексу задач функціональним вимогам, представленим у технічному завданні на створення комплексу задач з планування подорожей та обміну досвідом для мандрівників України

### **Мета випробувань**

Метою випробувань являється перевірка відповідності функцій комплексу задач з планування подорожей та обміну досвідом для мандрівників України вимогам технічного завдання.

### **Загальні положення**

Випробування проводяться на основі наступних документів:

* ГОСТ 34.603−92. Інформаційна технологія. Види випробувань автоматизованих систем;
* ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

### **Результати випробувань**

В процесі тестування були перевірена уся функціональність комплексу задач (КЗ). У наступних таблицях наведений перелік випробувань основних функціональних можливостей (табл. 5.1 – 5.9).

Таблиця 5.1 – Реєстрація нового користувача

|  |  |
| --- | --- |
| Мета тесту | Перевірити реєстрацію користувача |

Продовження таблиці 5.1

|  |  |
| --- | --- |
| Початковий стан | Вхідна електронна адреса не використовувалась для реєстрації користувача |
| Вхідні дані | Електронна адреса формату XXX@XX.XX  Ім’я користувача  Пароль користувача |
| Схема проведення тесту | Участник відкриває сторінку реєстрації, вводить дані, натискає кнопку «Зареєструватись» |
| Очікуваний результат | Створення аккаунту користувача |
| Стан після проведення випробувань | Створений новий аккаунт, даний аккаунт авторизований, перенаправлення на головну сторінку |

Таблиця 5.2 – Авторизація нового користувача

|  |  |
| --- | --- |
| Мета тесту | Перевірити авторизацію користувача |
| Початковий стан | Відкрита форма авторизації користувача, в системі створений аккаунт за вхідною електронною поштою |
| Вхідні дані | Електронна пошта користувача  Пароль користувача |
| Схема проведення тесту | Учасник вводить вхідні дані у форму авторизації, натискає кнопку «Увійти» |
| Очікуваний результат | Учасник авторизований |
| Стан після проведення випробувань | Учаснику надана роль користувача, у навігаційному меню з’явились кнопки «Профіль» та «Вийти» |

Таблиця 5.3 – Створення нового місця

|  |  |
| --- | --- |
| Мета тесту | Перевірити створення нового місця користувачем |
| Початковий стан | Користувач авторизований, відкрита сторінка «Усі місця» |
| Вхідні дані | Назва місця, опис місця, геолокації місця |
| Схема проведення тесту | Учасник натискає кнопку «Додати нове місце», вводить вхідні дані у форму створення місця, позначає геолокації місця на мапі, натискає кнопку «Додати місце» |
| Очікуваний результат | Створене нове місце з вхідними даними |

Продовження таблиці 5.3

|  |  |
| --- | --- |
| Стан після проведення випробувань | У профілі користувача в розділі «Створені об’єкти» відображається щойно створене місце |

Таблиця 5.3 – Відображення мапи України

|  |  |
| --- | --- |
| Мета тесту | Перевірити відображення місць на мапі України |
| Початковий стан | Відкрита сторінка «Усі місця» |
| Вхідні дані |  |
| Схема проведення тесту | Учасник натискає кнопку «Мапа України» |
| Очікуваний результат | Відкрита сторінка «Мапа України», області поза кордонами України затемнені, при наведенні на будь-яку область України відповідна область підсвічується, у правому верхньому куті карти відображається інформаційне поле по даній області, на карті відображаються позначки місць |
| Стан після проведення випробувань | Відкрита сторінка відображає мапу України |

Таблиця 5.4 – Відображення сторінки обраного місця

|  |  |
| --- | --- |
| Мета тесту | Перевірити відображення сторінки місця |
| Початковий стан | Відкрита сторінка «Усі місця» |
| Вхідні дані |  |
| Схема проведення тесту | Учасник натискає кнопку «Дізнатись більше», розташованій на блоці інформації щодо певного місця |
| Очікуваний результат | Відкрита сторінка обраного місця. Коректно відображається інформація по даному місцю, на мапі позначено геолокацію об’єкта, під описом об’єкту розташований блок тегів. Якщо в даного місця немає тегів, то відображається лише поле «Додати тег» |
| Стан після проведення випробувань | Відкрита сторінка відображає повну інформацію про об’єкт |

Таблиця 5.5 – Додавання тегу для обраного місця

|  |  |
| --- | --- |
| Мета тесту | Перевірити додавання тегів для обраного місця |
| Початковий стан | Відкрита сторінка об’єкту, тег з вхідною назвою ще не існує в системі, користувач авторизований |
| Вхідні дані | Назва тегу |
| Схема проведення тесту | Учасник натискає на поле «Додати тег», вводить назву тегу, натискає кнопку «Enter» |
| Очікуваний результат | Під описом об’єкту в блоці тегів відображається новий створений тег з відмітко про те, що один користувач його вподобав |
| Стан після проведення випробувань | Створений новий тег прив’язаний до користувача да відповідного об’єкту |

## **Висновок до розділу**

В даному розділі було створено керівництво користувача, наведено опис тестів і порядок їх виконання для перевірки відповідності програмного забезпечення комплексу задач функціональним вимогам.

ЗАГАЛЬНІ ВИСНОВКИ

У ході виконання дипломного проекту були детально розглянуті питання, які виникають в процесі розробки комплексу задач, що стосуються рекомендаційної системи туристичних об’єктів. Були виділені основні ключові етапи, притаманні процесу, та взаємозв’язки між ними.

Був проведений ґрунтовний аналіз предметного середовища, описаний бізнес-процес розробки плану виробництва.

На основі даних, отриманих в процесі аналізу, було сформульовано відповідну математичну задачу та запропоноване рішення, що дасть змогу надавати рекомендації користувачам щодо туристичних об’єктів в Україні.

Використання рекомендаційної системи в застосунку про туристичні об’єкти в Україні дасть змогу користувачам швидше знаходити релевантний контент та збільшить ефективність використання ресурсу.

Для розробки програмного забезпечення серверної частини була використана мова програмування JavaScript на платформі Node.js та сервіс Backendless.

Для розробки клієнтської частини використовувались такі засоби як HTML/CSS/JS, AngularJS2, Leaflet.

Наведена детальна інструкція користувача по експлуатації комплексу задач, описана методика проведення випробувань, яка показує можливість введення програми в експлуатацію.

Перелік посилань

* + 1. Агафонкін В. Interactive Choropleth Map [Електронний ресурс] / Володимир Агафонкін – Режим доступу до ресурсу: <http://leafletjs.com/examples/choropleth.html>.
    2. Головань О. Ю. Розробка веб-додатків за допомогою сучасних технологій на стороні клієнта [Електронний ресурс] / О. Ю. Головань, О. М. Рейда – Режим доступу до ресурсу: <http://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2016/paper/viewFile/762/614>.
    3. Проектирование программного обеспечения [Електронний ресурс] // Хабрахабр. – 2009. – Режим доступу до ресурсу: <https://habrahabr.ru/post/74330/>.
    4. Angular 2: Построенный на TypeScript [Електронний ресурс] // Хабрахабр. – 2015. – Режим доступу до ресурсу: <https://habrahabr.ru/post/252337/>.
    5. Кантор І. Современные возможности ES-2015 [Електронний ресурс] / Ілля Кантор – Режим доступу до ресурсу: <https://learn.javascript.ru/es-modern>.
    6. WEBPACK DOCUMENTATION [Електронний ресурс] – Режим доступу до ресурсу: <http://webpack.github.io/docs/>.
    7. BACKENDLESS DOCUMENTATION [Електронний ресурс] – Режим доступу до ресурсу: <https://backendless.com/products/documentation/>.
    8. Deployment diagram [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Deployment_diagram>.
    9. Recommender system [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Recommender_system>.
    10. Джонс Т. Рекомендательные системы: Часть 1. Введение в подходы и алгоритмы [Електронний ресурс] / Тім Джонс // IBMdeveloperWorks. – 2014. – Режим доступу до ресурсу: <https://www.ibm.com/developerworks/ru/library/os-recommender1/>.
    11. Ройзнер М. Как работают рекомендательные системы. Лекция в Яндексе [Електронний ресурс] / Михайло Ройзнер // Хабрахабр. – 2014. – Режим доступу до ресурсу: <https://habrahabr.ru/company/yandex/blog/241455/>.
    12. Рекомендательные системы: You can (not) advise [Електронний ресурс] // Хабрахабр. – 2013. – Режим доступу до ресурсу: <https://habrahabr.ru/post/176549/>.
    13. Пишем простую систему рекомендаций на примере Хабра [Електронний ресурс] // Хабрахабр. – 2014. – Режим доступу до ресурсу: <https://habrahabr.ru/post/230155/>.
    14. Jaccard index [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Jaccard\_index.

ДОДАТОК А   
Тексти програмного коду

(Найменування програми (документа))

***Тексти програмного коду комплексу задач з надання рекомендацій мандрівникам України***

(Обсяг програми (документа), арк., Кб)

*85 арк, 124 Кб*

(Вид носія даних)

*CD-RW*

*студента групи ІС-23 IV курсу*

*Григор’єва Ю.В.*

Package.json

{

"name": "Places",

"version": "1.0.0",

"description": "recommend system for travellers of Ukraine",

"scripts": {

"typings-install": "node node\_modules/typings/dist/bin.js install",

"postinstall": "npm run typings-install",

"build": "node node\_modules/webpack/bin/webpack.js --inline --colors --progress --display-error-details --display-cached",

"watch": "npm run build -- --watch",

"server": "node node\_modules/webpack-dev-server/bin/webpack-dev-server.js --inline --colors --progress --display-error-details --display-cached --port 3000 --content-base src",

"start": "npm run server"

},

"contributors": [

"Yuliaa Hryhorieva <julia.hrihorieva@gmail.com>"

],

"license": "MIT",

"devDependencies": {

"awesome-typescript-loader": "~0.16.2",

"es6-promise": "3.0.2",

"es6-shim": "0.35.0",

"reflect-metadata": "0.1.2",

"source-map-loader": "^0.1.5",

"typescript": "~1.8.9",

"typings": "^0.7.9",

"webpack": "^1.12.9",

"webpack-dev-server": "^1.14.0",

"webpack-merge": "^0.8.4"

},

"dependencies": {

"@angular2-material/checkbox": "^2.0.0-alpha.4",

"@angular2-material/core": "^2.0.0-alpha.4",

"angular2": "2.0.0-beta.15",

"backendless": ">=3.1.0",

"core-js": "^2.2.0",

"d3": "^3.5.17",

"lodash": "^4.12.0",

"mapbox.js": "^2.4.0",

"moment": "^2.13.0",

"rxjs": "5.0.0-beta.2",

"zone.js": "~0.6.11"

},

"keywords": [

"Angular2",

"angular2-seed",

"official angular 2 seed",

"official angular2 seed",

"bachelours project"

]

}

index.html

<!DOCTYPE html>

<html>

<head>

<meta charset=UTF-8>

<title>Моя країна</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link data-require="bootstrap-css@3.3.6" data-semver="3.3.6" rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.css" />

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

<link rel="stylesheet" href="https://code.getmdl.io/1.1.3/material.indigo-yellow.min.css" />

<script src="http://cdn.leafletjs.com/leaflet/v0.7.7/leaflet.js"></script>

<link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet/v0.7.7/leaflet.css" />

</head>

<body style="height: 100%">

<seed-app>

Зачекайте...

</seed-app>

<script src="polyfills.bundle.js"></script>

<script src="vendor.bundle.js"></script>

<script src="app.bundle.js"></script>

</body>

</html>

app.ts

import {bootstrap} from 'angular2/platform/browser';

import {provide} from 'angular2/core';

import {HTTP\_PROVIDERS} from 'angular2/http';

import {ROUTER\_PROVIDERS, LocationStrategy, HashLocationStrategy} from 'angular2/router';

import {SeedApp} from './app/seed-app';

const Backendless = require('backendless');

var APP\_ID:string = '7B2AF945-909B-6DB6-FF64-C772BC00BD00';

var APP\_KEY:string = 'ED9F0C34-3A5E-CB75-FF86-0E4AE53E1400';

var APP\_VER:string = 'v1';

Backendless.initApp(APP\_ID, APP\_KEY, APP\_VER);

Backendless.enablePromises();

bootstrap(SeedApp, [

HTTP\_PROVIDERS,

ROUTER\_PROVIDERS,

provide(LocationStrategy, {useClass: HashLocationStrategy})

])

.catch(err => console.error(err));

seed-app.html

<!-- Always shows a header, even in smaller screens. -->

<div class="mdl-layout mdl-js-layout mdl-layout--fixed-header page-wrap">

<header class="mdl-layout\_\_header">

<div class="mdl-layout\_\_header-row">

<!-- Title -->

<span class="mdl-layout-title">Моя країна</span>

<!-- Add spacer, to align navigation to the right -->

<div class="mdl-layout-spacer"></div>

<nav class="mdl-navigation">

<a class="mdl-navigation\_\_link" [routerLink]=" ['/About'] ">Про нас</a>

<a class="mdl-navigation\_\_link" [routerLink]=" ['/Places'] ">Усі місця</a>

<a class="mdl-navigation\_\_link" [routerLink]="['/Map']">Мапа України</a>

<auth></auth>

</nav>

</div>

</header>

<main class="mdl-layout\_\_content">

<div class="page-content">

<router-outlet></router-outlet>

</div>

</main>

</div>

seed-app.ts

import {Component} from "angular2/core";

import {RouteConfig, ROUTER\_DIRECTIVES} from "angular2/router";

import {About} from "./components/about/about";

import {Places} from "./components/allPlaces/places";

import {FullPlace} from "./components/fullPlace/fullPlace";

import {NewPlace} from "./components/newPlace/newPlace";

import {PlaceService} from "./services/placeService";

import {LikeService} from "./services/likeService";

import {MapService} from "./services/mapService";

import {Register} from "./components/auth/register/register";

import {Login} from "./components/auth/login/login";

import {Auth} from "./components/auth/auth";

//import {Logout} from './components/auth/logout/logout';

import {Map} from './components/map/map';

@Component({

selector: 'seed-app',

providers: [

PlaceService, LikeService, MapService

],

pipes: [],

directives: [ROUTER\_DIRECTIVES, Auth, Map],

templateUrl: 'app/seed-app.html',

styleUrls: ['app/seed-app.css']

})

@RouteConfig([

{ path: '/about', component: About, name: 'About' },

{ path: '/places', component: Places, name: 'Places', useAsDefault: true },

{ path: '/place/:objId', component: FullPlace, name: 'FullPlace' },

{ path: 'places/add', component: NewPlace, name: 'NewPlace' },

{ path: 'register', component: Register, name: 'Register' },

{ path: 'login', component: Login, name: 'Login' },

{ path: 'map', component: Map, name: 'Map'}

])

export class SeedApp {

constructor() {}

}

likeService.ts

import {Injectable} from "angular2/core";

import {Likes} from "../components/entities/like";

import {DisplayTag} from '../components/entities/DisplayTag';

import \* as \_ from "lodash";

import Dictionary = \_.Dictionary;

import User = \_\_Backendless.User;

import {Tag} from "../components/entities/tag";

import {Place} from "../components/entities/place";

import {TagDisplay} from "../components/tagsComponent/tagDisplay/tagDisplay";

const Backendless = require('backendless');

@Injectable()

export class LikeService {

like:Likes;

getDisplayTag(place:Place, tag: Tag) {

let query = new Backendless.DataQuery();

query.condition = "place.objectId='" + place.objectId + "' and tag.objectId='" + tag.objectId + "'";

return Backendless.Persistence.of(Likes).find(query);

}

getLikedTags(placeID:string) {

let query = new Backendless.DataQuery();

query.condition = "place.objectId='" + placeID + "'";

query.options = { relations: ["tag"] };

return Backendless.Persistence.of(Likes).find(query).

then(res => res.data.map((like) => like.tag))

.then(tags => {

let displayTags = [];

let tagsCount = \_.groupBy(tags, "objectId");

for (let key in tagsCount) {

displayTags.push(new DisplayTag(tagsCount[key][0], tagsCount[key].length))

}

return displayTags;

});

}

addLike(place: Place, tag: Tag) {

console.log("tag in addLike:");

console.log(tag);

return Backendless.UserService.getCurrentUser()

.then(user=>{

let like = new Likes(place, user, tag);

return Backendless.Persistence.of(Likes).save(like);

});

}

getTagByName(name: string): Promise<Tag> {

let query = new Backendless.DataQuery();

query.condition = "name='" + name + "'";

return Backendless.Persistence.of(Tag).find(query)

.then(res=>{

if(!res.data.length) throw new Error('Tag with such name isn\'t existed');

return res;

});

}

createNewTag(tagName: string): Promise<Tag> {

return this.getTagByName(tagName)

.then(

res=>{

console.log(res);

return res.data[0];

},

err=>{

let tag = new Tag();

tag.name = tagName;

return Backendless.Persistence.of(Tag).save(tag);}

);

}

likeNewTag(tagName: string, place:Place) {

return this.createNewTag(tagName)

.then( tag=> {

console.log("created tag:");

console.log(tag);

this.addLike(place, tag)});

}

}

placeService.ts

import {Injectable} from "angular2/core";

import {Http, Response, Headers, RequestOptions} from "angular2/http";

import {Observable} from "rxjs/Observable";

import "rxjs/Rx";

import {Place} from "../components/entities/place";

const Backendless = require('backendless');

@Injectable()

export class PlaceService {

constructor(private \_http : Http){}

extractData(res: Response){

if (res.status < 200 || res.status >= 300) {

throw new Error('Bad response status: ' + res.status);

}

let body = res.json();

return body.data || body || {};

}

private handleError (error: any) {

// In a real world app, we might send the error to remote logging infrastructure

let errMsg = error.message || 'Server error';

console.error(errMsg); // log to console instead

return Observable.throw(errMsg);

}

createPlace(place: Place) : Promise<Place>{

return Backendless.Data.of(Place).save(place);

}

getPlaces() {

return Backendless.Data.of(Place).find();

}

getPlace(objId: string) : Promise<Place> {

var query = new Backendless.DataQuery();

query.options = { relations: ["location"] };

// query.options = {relationsDepth:1};

query.condition="objectId='" + objId + "'" ;

return Backendless.Data.of(Place).findById(objId, 1);

}

deletePlace(place:Place) {

return Backendless.Data.of(Place).remove(place);

}

updatePlace(place:Place) {

return Backendless.Data.of(Place).save(place);

}

}

allPlaces.html

<h3>Усі місця</h3>

<div class="container">

<div class="mdl-grid">

<div class="demo-card-wide mdl-card mdl-shadow--2dp mdl-cell mdl-cell--4-col">

<div class="mdl-card\_\_title">

</div>

<div class="mdl-card\_\_actions mdl-card--border">

<a class="mdl-button mdl-button--colored mdl-js-ripple-effect" (click)="addNewPlace()">

Додати нове місце

</a>

</div>

</div>

<place-detail \*ngFor="#place of places" [place]="place" class="demo-card-wide mdl-card mdl-shadow--2dp mdl-cell mdl-cell--4-col"></place-detail>

<button class="mdl-button mdl-js-button mdl-button--raised mdl-button--colored">

ДИВИТИСЬ ЩЕ

</button>

</div>

</div>

allPlaces.ts

import {Component, OnInit} from "angular2/core";

import {Router} from "angular2/router";

import {PlaceDetail} from "../placeDetail/placeDetail";

import {PlaceService} from "../../services/placeService";

import {Place} from "../entities/place";

const Backendless = require('backendless');

@Component({

selector: 'places',

templateUrl: 'app/components/allPlaces/allPlaces.html',

directives: [PlaceDetail],

styleUrls: ['app/components/allPlaces/allPlaces.css']

})

export class Places implements OnInit{

places : Place[];

constructor(

private \_router: Router,

private \_placeService: PlaceService

){}

ngOnInit(){

//this.places = this.\_placeService.getPlaces();

this.getPlaces();

}

getPlaces(){

this.\_placeService.getPlaces().then( res => this.places = res.data );

}

addNewPlace(){

let link = ['NewPlace'];

this.\_router.navigate(link);

}

}

fullPlace.html

<div class="row">

<div class="col-lg-12">

<h2 class="page-header">{{place?.name}}</h2>

</div>

<div class="col-md-6">

<p>{{place?.description}}</p>

<all-tags [tags]="tags" [place]="place"></all-tags>

</div>

<div class="col-md-6">

<div id="little\_map" style="width: 700px; height: 450px;"></div>

</div>

</div>

fullPlace.ts

import {Component, OnInit} from "angular2/core";

import {RouteParams, Router} from "angular2/router";

import {PlaceService} from "../../services/placeService";

import {LikeService} from "../../services/likeService";

import {Place} from "../entities/place";

import {DisplayTag} from "../entities/DisplayTag";

import {AllTags} from "../tagsComponent/allTags/allTags";

const Backendless = require('backendless');

const L = require('leaflet');

// import GeoPoint = \_\_Backendless.GeoPoint;

@Component({

selector: 'full-place',

templateUrl: 'app/components/full' +

'Place/fullPlace.html',

styleUrls: ['app/components/fullPlace/fullPlace.css'],

directives: [AllTags]

})

export class FullPlace implements OnInit {

place:Place;

count:number;

tags:DisplayTag[];

user:Backendless.User;

map;

constructor(private \_router:Router,

private \_routeParams:RouteParams,

private \_placeService:PlaceService,

private \_likeService:LikeService) {

}

ngOnInit() {

let objId = this.\_routeParams.get('objId');

this.getPlace(objId)

.then(()=> this.initMap());

this.getTags(objId);

// this.initMap();

}

getPlace(objId) {

return this.\_placeService.getPlace(objId)

.then(place => this.place = place);

}

delete() {

this.\_placeService.deletePlace(this.place).then(

()=> {

let link = ['Places'];

this.\_router.navigate(link);

}

);

}

editDescription() {

console.log('description will be changed');

let field = <HTMLInputElement>document.getElementById('description');

let button = document.getElementById('editDescription');

this.toggleEditing(field, button);

}

editTitle(event) {

console.log('title will be changed');

let field = <HTMLInputElement>document.getElementById('title');

let button = document.getElementById('editTitle');

this.toggleEditing(field, button);

}

toggleEditing(field:HTMLInputElement, button:HTMLElement) {

if (field.readOnly) {

field.readOnly = false;

field.focus();

} else field.readOnly = true;

button.classList.toggle('edited');

button.classList.toggle('saved');

}

getTags(placeId : string) {

return this.\_likeService

.getLikedTags(placeId)

.then(tagsCount => this.tags = tagsCount);

}

initMap(){

// var bounds = new L.latLngBounds([52.5, 21], [44, 41]);

var lat = this.place.location? this.place.location.latitude : 48.46;

var long = this.place.location? this.place.location.longitude : 30.87;

this.map = L.map('little\_map', {

center: new L.LatLng(lat, long),

zoom: 14

});

L.tileLayer(

'https://api.mapbox.com/styles/v1/kastrulya/ciot1llnt001ydaniiuhzd4sh/tiles/{z}/{x}/{y}?access\_token=pk.eyJ1Ijoia2FzdHJ1bHlhIiwiYSI6ImNpb3Bsdm92dTAwMDJ2bG0xenEwZmJlYm4ifQ.nsPNZQ726nMQtszDGhDX3w',

{

attribution: 'Map data &copy; <a href="http://openstreetmap.org">OpenStreetMap</a> contributors, <a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, Imagery © <a href="http://mapbox.com">Mapbox</a>',

accessToken: 'pk.eyJ1Ijoia2FzdHJ1bHlhIiwiYSI6ImNpb3Bsdm92dTAwMDJ2bG0xenEwZmJlYm4ifQ.nsPNZQ726nMQtszDGhDX3w'

}).addTo(this.map);

var marker = L.marker([lat, long]).addTo(this.map);

marker.bindPopup("<b>Title</b><br>I am a popup.");

}

edit(newTitle, newDescription){

// editDescription.value();

this.place.name = newTitle;

this.place.description = newDescription;

this.place.location = new Backendless.GeoPoint();

this.place.location.latitude = 55.332;

this.place.location.longitude = 30.453;

this.\_placeService.updatePlace(this.place);

}

}