**РЕФЕРАТ**

Обсяг роботи 40 сторінок, 13 ілюстрацій, 19 джерел посилань.

Ключові слова: РОЗРОБКА МОБІЛЬНОГО СЕРВІСУ, СЕРВІС ДЛЯ ПРОВЕДЕННЯ КВЕСТІВ, ТУРИСТИЧНИЙ СЕРВІС, СФЕРА ТУРИЗМУ ТА РОЗВАГ, МОБІЛЬНИЙ ДОДАТОК .

Об’єктом дослідження є туризм і сфера послуг. Предметом дослідження є вивчення потреб споживачів данних галузей і пошук раціональних шляхів для задоволення їх потреб.

Метою данної роботи є створення сервісу призначеного для вирішення проблем туристичної галузі.

Інструменти розроблення: безкоштовне, вільно поширюване середовище розробки Microsoft Visual Studio Community 2017, веб хостинги Azure, Heroku там MLab, мова програмування C#.

Результати роботи: Розроблено мобільний сервіс для проведення користувачами квестів з сайтом для адміністрування та WebAPI.

Програмний продукт, побудований на основі поставленої задачі було опубліковано на Google Play Market в якості «Альфа» версії.

.

## **ЗМІСТ**

|  |  |
| --- | --- |
|  |  |
| СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ | 5 |
| ВСТУП | 6 |
| РОЗДІЛ 1 ОГЛЯД ОСНОВНХ ПРОБЛЕМ І РІШЕНЬ В СФЕРІ ТУРИЗМУ І РОЗВАГ | 11 |
| 1.1. Основні проблеми |  |
| 1.2. Аналіз існуючих рішень | 11 |
| 1.3. Опис основних концепцій власного сервісу | 12 |
| РОЗДІЛ 2 ОГЛЯД ДОСТУПНИХ ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ СЕРВІСУ | 13 |
| 2.1. Фреймворк для мобільного додатку | 13 |
| 2.1.1. Kotline Native | 13 |
| 2.1.2. React Native | 14 |
| 2.1.3. Xamarin Forms | 15 |
| 2.2. Фреймворк для Web API | 17 |
| 2.2.1. ASP.NET Core | 17 |
| 2.2.2. Django | 18 |
| 2.3. СУБД і Хостинги | 21 |
| 2.3.1. Azure | 21 |
| 2.3.2. MLab | 21 |
| 2.3.3. AWS | 22 |
| 2.3.4 Heroku | 23 |
| 2.4 Методи організації процесу розробки | 23 |
| РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА | 27 |
| 3.1 Проектування та розробка API | 27 |
| 3.2 Проектування та розробка мобільного додатку | 29 |
| 3.3 Проектування та розробка сайту для адміністрування | 33 |
| 3.4 Ідеї розвитку проекту | 36 |
| ВИСНОВОК | 38 |
| ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ | 39 |

# **СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ**

MVVM - Model-View-ViewModel патерн програмування;

MVC – Model-View-Controler патерн пограмування;

MVP - minimum viable product, мінімально робочий продукт

## **ВСТУП**

**Оцінка сучасного стану об’єкта дослідження або розробки.**

Всебічне дослідження аспектів туризму є актуальною проблемою, яка обумовлена його багатогранністю та суспільною значущістю. Можна впевнено сказати, що туризм став однією із невід’ємних складових життєдіяльності людей у більшості країн світу, та є одним із найперспективніших напрямків соціально-економічного розвитку у ХХІ сторіччі. Туристична діяльність ввійшла до найбільших експортних галузей світу та стала найрентабельнішою галуззю світової економіки, що визначає актуальність і практичну значимість проблеми та вимагає всебічного її вивчення. Питання розвитку туризму у світі та в Україні набувають все більшої гостроти, відповідно до динамічних умов сьогодення, тому як ніколи актуальним є пошук оптимальних методів для розв`язання нагальних проблем цієї галузі.

До проблем в туризмі та сфері послуг можна віднести:

* проблема вибору;
* проблема проінформованості;
* проблема цікавості;
* проблема популярності;
* проблема “фотографій”.

**Актуальність теми.**

Однією з компаній, що вже 18 років успішно займається пошуками розв`язань цих проблем і пропонує готові варіанти рішеннь, є TripAdvisor [1]. Цей американський веб-сайт дозволяє своїм користувачам спланувати майбутню подорож до будь-якої країни світу. За даними офіційного веб-сайту компанії, TripAdvisor щомісяця обслуговує 315 мільйонів унікальних відвідувачів, більш ніж 70 мільйонів зареєстрованих користувачів, які залишили більше ніж 200 мільйонів відгуків, що свідчить про надзвичайно високу зацікавленість людей в подібних видах послуг.

Хоча попит достатньо великий, проте на ринку представлено невелику кількість програмних продуктів, які б вирішували проблеми туризму та сфери послуг. Дослідження цих проблем є досить перспективним і потрібним на сьогоднішній день, тому розробка сервісу, який би зміг вирішити ці питання, є досить актуальною темою, а знахоження способів для їх розв’язання може створити цілу інфраструктуру для даних галузей.

**Мета й завдання роботи.**

Метою данної роботи є створення сервісу призначеного для вирішення проблем туристичної галузі. Для досягнення поставленої мети виділимо наступні завдання:

1) збір необхідної інформації про програмні засоби в данній сфері, що представлені на ринку (опис недоліків, переваг і особливостей  кожного програмного продукту);

2) аналіз і систематизація зібраної інформації, формування висновків;

3) розробка власного сервісу на основі отриманих даних.

**Об’єкт, методи й засоби дослідження або розроблення.**

**Об’єктом дослідження** є туризм і сфера послуг. **Предметом дослідження** є вивчення потреб споживачів данних галузей і пошук раціональних шляхів для задоволення їх потреб.

В ході роботи буkо застосовано емпіричний метод, який полягає в спостереженні і дослідженні конкретних явищ, а також узагальненні, класифікації та описі результатів дослідження, впровадженні їх у практичну діяльність людини. Цей методи дослідження є визначальними в навчально-дослідній справі, що пов’язана з практикою, та забезпечує накопичення, фіксацію та узагальнення вихідного матеріалу.

**РОЗДІЛ 1 ОГЛЯД ОСНОВНХ ПРОБЛЕМ І РІШЕНЬ В СФЕРІ ТУРИЗМУ І РОЗВАГ**

### **1.1.Основні проблеми**

Всебічне дослідження аспектів туризму і сфери розваг є надзвичайно актуальною проблемою, яка обумовлена його багатогранністю та суспільною значущістю. Можна впевнено сказати, що туризм і сфера розваг стали однією із невід’ємних складових життєдіяльності людей у більшості країн світу, та є одним із найперспективніших напрямків соціально-економічного розвитку у ХХІ сторіччі. Туристична діяльність ввійшла до найбільших експортних галузей світу та стала най рентабельнішою галуззю світової економіки, що визначає актуальність і практичну значимість проблеми та вимагає всебічного її вивчення. Питання розвитку туризму у світі та в Україні набувають все більшої гостроти, відповідно до динамічних умов сьогодення, тому як ніколи актуальним є пошук оптимальних методів для розв`язання нагальних проблем цієї галузі.

До проблем в туризмі та сфері послуг можна віднести такі проблеми:

1. проблема вибору

потрібно вказати, що туризм, або просто відпочинок, зазвичай є коштовним задоволенням, тому проблема вибору між альтернативнми способами відпочинку є пріоритетною для більшості користувачів

1. проблема проінформованості

Більшість людей мало проінформована в якості послуг, які їм будуть надані, і покладається, в основному,на випадок, або на поради близьких людей

1. проблема цікавості

Людині важко оцінити, чи буде для неї дана подорож чи відпочинок цікавим, якщо раніше вона в такому не приймала участі.

1. проблема “фотографій”

На даний час стало дуже популярним, а інколи навіть соціально важливим, робити фото з різних цікавих місць і викладати їх в соціальні мережі

### **1.2. Аналіз існуючих рішень**

**TripAdvisor** [1]— американський веб-сайт, що дозволяє своїм користувачам спланувати майбутню подорож до будь-якої країни світу. Послуги сайту є безкоштовні для користувачів, які надають більшу частину контенту. За даними офіційного веб-сайту компанії, TripAdvisor щомісяця обслуговує 315 мільйонів унікальних відвідувачів , більш ніж 70 мільйонів зареєстрованих користувачів, які залишили більше ніж 200 мільйонів відгуків.[2]

На жаль даний сервіс вирішує лише одну проблему – проінформованість жодним чином не втручається в інші, що обмежує користувачів у

### **1.3. Опис основних концепцій власного сервісу**

**Ідея**

Зробити сервіс, в якому користувачам будуть підбиратися місця для прогулянки і подорожі, де вони зможуть проходити різного роду квести. Кожен квест має набір певних завдань. За проходження квестів надається певна кількість балів, які потім можна буде міняти на реальні кошти припокупці в магазині сервісу.

**Аудиторія**

Це сервіс для активних людей, яким дуже подобається подорожувати, робити фотографії і знайомитися з такими ж цікавими людьми.

**Типи завдань**

1)*Просте* (Потрібно зробити фото, в якомусь конкретно заданому місці, потім пройти декілька тестів (на знання історії чи певних особливостей цього місця);

2) *Пошукове*(Користувачу надається набір фактів про якесь місце (подію), потрібно знайти на карті описане місце, приїхати, зробити там фото і пройти тести);

3)*Персональне* (користувач сам має змогу робити завдання для друзів);

4) *Додаткове* (пройшовши завдання, користувачу надається можливість сфотографуватися з іншими користувачами, які нещодавно проходили  це ж завдання, і пройти разом якийсь простий тест).

## **РОЗДІЛ 2 ОГЛЯД ДОСТУПНИХ ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ СЕРВІСУ**

### **2.1. Фреймворк для мобільного додатку**

Для того, щоб зменшити кількість часу на розробку MVP, було прийнято рішення розробити мобільні додатки під дві найпопулярніші платформи(Android та IOS) на красплатформових фреймворках, які дозволять виділити бізнес логіку додатку в окрему бібліотеку

На даний момент існує декілька повноцінних таких фреймворків . Далі буде розглянутий кожен з них

**2.1.1. Kotline Native**

Це фреймворк від компанії JetBrains який з вихідного кода написаного на Kotline генерує байткод для JVM та код на Objective C, що дозволяє виділити в окремий модуль бізнес логіку додатку, в такому випадку спільного коду виходить приблизно 50% [4]

Переваги:

* Лаконічний і продуманий синтакис Kotline
* Повна сумісність з Java

Недоліки:

* Неповна сумісність з Objective C
* Великакілкість помилок (обумовлено «свіжістю» проекту)
* Велика кількість платформозалежного коду

**2.1.2. React Native**

React— [відкрита](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B4%D0%BA%D1%80%D0%B8%D1%82%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) [JavaScript](https://uk.wikipedia.org/wiki/JavaScript) [бібліотека](https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC) для створення [інтерфейсів користувача](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0), яка покликана вирішувати проблеми часткового оновлення вмісту [веб-сторінки](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0), з якими стикаються в розробці [односторінкових застосунків](https://uk.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA). Розробляється [Facebook](https://uk.wikipedia.org/wiki/Facebook), [Instagram](https://uk.wikipedia.org/wiki/Instagram) і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі [веб-застосунки](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA), які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки [користувацький інтерфейс](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0) у застосунках. Це відповідає видові у шаблоні [модель-вид-контролер](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%B2%D0%B8%D0%B4-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80) (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих [фреймворках](https://uk.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) MVC, таких як [AngularJS](https://uk.wikipedia.org/wiki/AngularJS). Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотека інтерфейсу користувача React часто використовується разом з іншими бібліотеками, такими як Redux.[5]

React Native – це адаптація React ідеології на мобільні платформи з використанням нативних елементів. В результаті отримуємо орієнтовно 95% спільного коду.

Переваги :

* Велика кількість спільного коду;
* Простота підключення величезної кількості JS бібліотек.

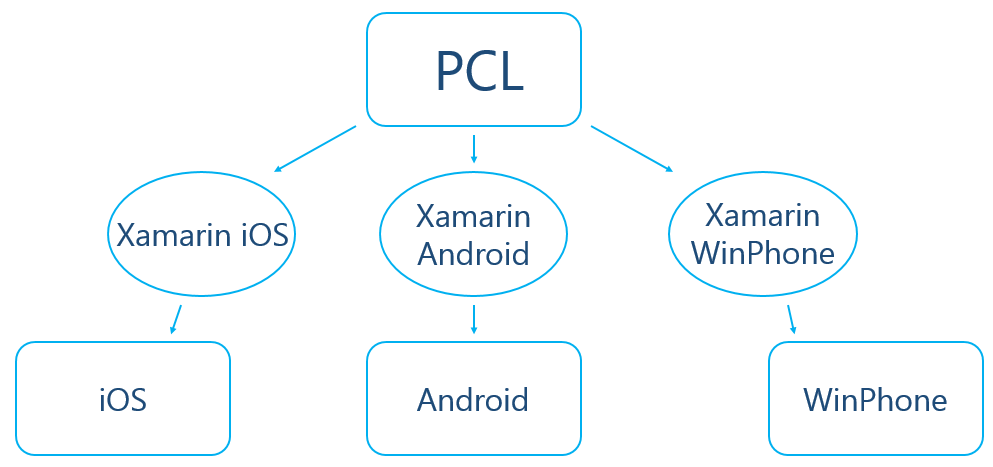
Недоліки:

* Нестабільність;
* Повільність.

**2.1.3. Xamarin Forms**

Xamarin Forms – це фреймворк від компанії Xamarin, для кросплатформової розробки мобільних додатків на базі .NET

Його особливістю є те, що розмітка також є спільною для декількох плаформ але при цьому додаток буде повністю нативним і його модулі можна буде використовувати в інших нативних рішеннях. А також присутня чітка система для зв’язування логіки і інтерфейсу, яка базується на MVVM патерні програмування

Бізнес логіка додатку виділяється в окрему бібліотеку PCL, яка при компіляції інтерпретується у нативний код для кожної платформи

*Рисунок 1. Структура Xamarin Forms*

Переваги:

* Висока частина спільного коду (до 98 %);
* Можливість використання нативних елементів;
* Велика кількість бібліотек з готовими кастомними елементами.

Недоліки:

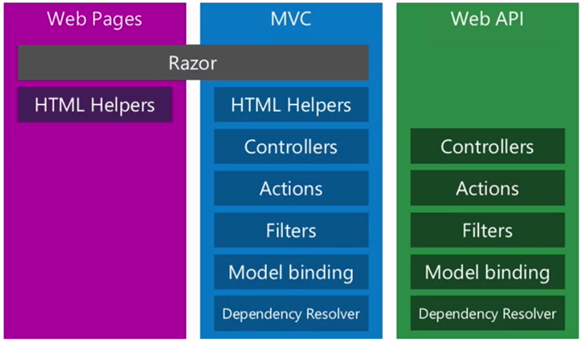
* Складність в підтримці;
* Нестабільність.

Для розробки MVP було вибрано саме даний фреймворк, а стабільність було покращено використанням спеціальної бібліотеки, яка розбиває додаток на модулі, які набагато простіше підтримувати.

**2.2. Фреймворк для Web API**

Для розробки Web API, який би обслуговував мобільний додаток та сайт для адміністрування сервісом, було розглянуто декілька варіантів популярних веб фреймворків.

## **2.2.1. ASP.NET Core**



*Рисунок 2 Структура ASP.NET Core*

ASP.NET — технологія створення [веб-застосунків](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) і [веб-сервісів](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D1%96%D1%81) від компанії [Майкрософт](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B9%D0%BA%D1%80%D0%BE%D1%81%D0%BE%D1%84%D1%82). Вона є складовою частиною платформи [Microsoft.NET](https://uk.wikipedia.org/wiki/Microsoft.NET) і розвитком старішої технології [Microsoft ASP](https://uk.wikipedia.org/wiki/Active_Server_Pages). На цей час останньою версією цієї технології є ASP.NET Core 2.0

ASP.NET зовні багато в чому зберігає схожість із старішою технологією ASP, що дозволяє розробникам відносно легко перейти на ASP.NET. У той же час внутрішній устрій ASP.NET істотно відрізняється від ASP, оскільки вона заснована на платформі. NET і, отже, використовує всі нові можливості, що надаються цією платформою.

Переваги :

* ASP.NET має перевагу у швидкості в порівнянні з іншими технологіями, заснованими на [скриптах](https://uk.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) (PHP, тощо)
* Розширюваний набір елементів управління і бібліотек класів дозволяє швидше розробляти застосунки
* ASP.NET спирається на багатомовні можливості .NET, що дозволяє писати код сторінок на C#, VB, C/C++ та ін.
* Поділ візуальної частини та бізнес-логіки
* Розширювана модель обробки запитів

Також у ASP.NET Core можна використовувати ті ж само моделі даних, що і на клієнтській частині Xamarin Forms, що дуже зручно для підтримки додатку при постійному оновленні WebAPI

Тому даний фреймворк було використано для написання Web API модуля мобільного додатку

**2.2.2. Django**

Django — високорівневий [відкритий](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B4%D0%BA%D1%80%D0%B8%D1%82%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) [Python](https://uk.wikipedia.org/wiki/Python)-[фреймворк](https://uk.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) для розробки веб-систем. Названо його було на честь джазмена [Джанго Рейнхардта](https://uk.wikipedia.org/wiki/%D0%94%D0%B6%D0%B0%D0%BD%D0%B3%D0%BE_%D0%A0%D0%B5%D0%B9%D0%BD%D1%85%D0%B0%D1%80%D0%B4%D1%82) (відповідно до музичних смаків засновника проекту).

Сайт на Django будується з однієї або декількох частин, які рекомендується робити модульними. Це одна з істотних архітектурних відмінностей цього фреймворку від деяких інших (наприклад [Ruby on Rails](https://uk.wikipedia.org/wiki/Ruby_on_Rails)).

Архітектура Django подібна до  [MVC](https://uk.wikipedia.org/wiki/MVC). Однак, те що називається «контролером» в класичній моделі MVC, в Django називається «вид» ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) view), а те, що мало б бути «видом», називається «шаблон» ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0)template). Таким чином, MVC розробники Django називають MTV («Модель-Шаблон-Вид»).

Початкова розробка Django, як засобу для роботи новинних ресурсів, досить сильно позначилася на його архітектурі: він надає ряд засобів, які допомагають у швидкій розробці [веб-сайтів](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B0%D0%B9%D1%82) інформаційного характеру. Так, наприклад, розробнику не потрібно створювати контролери та сторінки для адміністративної частини сайту, в Django є вбудований модуль для керування вмістом, який можна включити в будь-який сайт, зроблений на Django, і який може керувати відразу декількома сайтами на одному [сервері](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80). Адміністративний модуль дозволяє створювати, змінювати і вилучати будь-які об'єкти наповнення сайту, протоколюючі всі дії, а також надає [інтерфейс](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) для управління користувачами і групами (з призначенням прав).

У дистрибутив Django також включені програми для системи коментарів, синдикації [RSS](https://uk.wikipedia.org/wiki/RSS) і [Atom](https://uk.wikipedia.org/wiki/Atom), «статичних сторінок»(якими можна управляти без необхідності писати контролери та відображення), перенаправлення URL та інше.

Django був створений для управління сайтами новин [LJWorld.com](http://www.ljworld.com/), [lawrence.com](http://www.lawrence.com/) і [KUsports.com](http://www.kusports.com/) компанії [The World Company](https://uk.wikipedia.org/w/index.php?title=The_World_Company&action=edit&redlink=1) ([Лоренс (Канзас)](https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D1%80%D0%B5%D0%BD%D1%81_(%D0%9A%D0%B0%D0%BD%D0%B7%D0%B0%D1%81)), США), але з моменту початку розповсюдження його у статусі [відкритого програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B4%D0%BA%D1%80%D0%B8%D1%82%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) отримав величезну популярність в усьому світі як платформа до численних систем.

Деякі можливості Django:

* [ORM](https://uk.wikipedia.org/wiki/ORM), [API](https://uk.wikipedia.org/wiki/API) доступу до [БД](https://uk.wikipedia.org/wiki/%D0%91%D0%94) з підтримкою [транзакцій](https://uk.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D1%96%D1%8F)
* вбудований інтерфейс адміністратора з уже наявними перекладами на більшість мов
* диспетчер URL на основі [регулярних виразів](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%96_%D0%B2%D0%B8%D1%80%D0%B0%D0%B7%D0%B8)
* розширювана система шаблонів з тегами та наслідування
* система кешування
* [інтернаціоналізація](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F)
* архітектура застосунків, що підключаються, які можна встановлювати на будь-які Django-сайти
* «generic views» - шаблони функцій контролерів
* авторизація та аутентифікація, підключення зовнішніх модулів аутентифікації: [LDAP](https://uk.wikipedia.org/wiki/LDAP), [OpenID](https://uk.wikipedia.org/wiki/OpenID) та ін.
* система фільтрів («middleware») для побудови додаткових обробників запитів, наприклад включені в дистрибутив фільтри для кешування, стиснення, нормалізації URL і підтримки анонімних сесій
* бібліотека для роботи з формами (наслідування, побудова форм за існуючою моделлю БД)
* вбудована автоматична документація по тегам шаблонів та моделям даних, доступна через адміністративний застосунок

Різні компоненти фреймворку між собою пов'язані слабо, тому достатньо будь-яку частину замінити на аналогічну. Наприклад, замість вбудованих шаблонів можна використовувати [Mako](http://makotemplates.org/) або [Jinja](http://jinja.pocoo.org/2/).

Тому даний фреймворк було обрано для сайта адміністрування

**2.3. СУБД і Хостинги**

**2.3.1. Azure**

Microsoft Azure— це хмарна платформа та інфраструктура корпорації [Microsoft](https://uk.wikipedia.org/wiki/Microsoft), призначена для розробників [застосунків](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) [хмарних обчислень](https://uk.wikipedia.org/wiki/%D0%A5%D0%BC%D0%B0%D1%80%D0%BD%D1%96_%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F) ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) cloud computing) і покликана спростити процес створення онлайнових додатків.

Присутня дуже лояльна програм для стартапів, яка надає безкоштовні сервери на певний час, чим ми й скористалися

**2.3.2. MLab**

Для збереження даних в програмі було використано безкоштовний БД хостинг MLab з системою керування базами даних - MongoDB

**MongoDB** — документо-орієнтована система керування базами даних (СКБД) з відкритим сирцевим кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

Код MongoDB написаний на мові C++ і поширюється в рамках ліцензії AGPLv3.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій. У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин.

**2.3.3. AWS**

**Amazon Web Services** є дочірньою компанією [Amazon.com](https://uk.wikipedia.org/wiki/Amazon.com), що [надає](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F_%D1%8F%D0%BA_%D0%BF%D0%BE%D1%81%D0%BB%D1%83%D0%B3%D0%B0) [платформу хмарних обчислень](https://uk.wikipedia.org/wiki/%D0%A5%D0%BC%D0%B0%D1%80%D0%BD%D1%96_%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F) в оренду приватним особам, компаніям та урядам на основі платної підписки. Технологія дозволяє абонентам мати у своєму розпорядженні повноцінний [віртуальний](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F) [кластер комп'ютерів](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80), який завжди доступний через Інтернет. Віртуальні комп'ютери AWS мають більшість атрибутів реального комп'ютера, включаючи апаратні пристрої (процесор, відеокарту, локальну та оперативну пам'ять, жорсткий диск або SSD-накопичувач); операційну систему на вибір; мережу; і попередньо встановлені прикладні програми, такі як [веб-сервер](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), [база даних](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), [CRM](https://uk.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%B2%D1%96%D0%B4%D0%BD%D0%BE%D1%81%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8_%D0%B7_%D0%BA%D0%BB%D1%96%D1%94%D0%BD%D1%82%D0%B0%D0%BC%D0%B8) і т. д. Кожна система AWS також віртуалізує консольний ввід/вивід (клавіатура, дисплей і миша), що дозволяє користувачам AWS підключитися до своєї системи AWS за допомогою [браузера](https://uk.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80). Браузер виступає як вікно у віртуальний комп'ютер, дозволяючи користувачу входити в систему, налаштовувати та використовувати свої віртуальні системи так само, як справжній, фізичний комп'ютер. Це дозволяє їм налаштувати систему так, щоб надавати інтернет-орієнтовані сервіси та послуги своїм клієнтам.

Даний сервіс було вибрано для зберігання фотографій, які створюються користувачем під час проходження квестів.

**2.3.4 Heroku**

Heroku — [хмарна](https://uk.wikipedia.org/wiki/%D0%A5%D0%BC%D0%B0%D1%80%D0%BD%D1%96_%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F) [PaaS](https://uk.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0_%D1%8F%D0%BA_%D0%BF%D0%BE%D1%81%D0%BB%D1%83%D0%B3%D0%B0)-платформа, що підтримує ряд [мов програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F). Компанією Heroku володіє [Salesforce.com](https://uk.wikipedia.org/wiki/Salesforce.com). Heroku, одна з перших хмарних платформ, з'явилась в червні [2007 року](https://uk.wikipedia.org/wiki/2007) і спочатку підтримувала тільки мову програмування [Ruby](https://uk.wikipedia.org/wiki/Ruby), але на даний момент список підтримуваних мов також включає в себе [Java](https://uk.wikipedia.org/wiki/Java), [Node.js](https://uk.wikipedia.org/wiki/Node.js), [Scala](https://uk.wikipedia.org/wiki/Scala), [Clojure](https://uk.wikipedia.org/wiki/Clojure), [Python](https://uk.wikipedia.org/wiki/Python) і [PHP](https://uk.wikipedia.org/wiki/PHP). На серверах Heroku використовуються операційні системи [Debian](https://uk.wikipedia.org/wiki/Debian) або [Ubuntu](https://uk.wikipedia.org/wiki/Ubuntu) (яка також заснована на Debian.

**2.4 Методи організації процесу розробки**

Для того, щоб ефективно організувати роботу команди над якимось продуктом, необхідно притримуватись якись методологій організації. В умовах постійної невизначеності, в який постійно перебуває даний проект,було прийнять рішення користуватися гнучким підходом організації Scrum.

Scrum — підхід [управління проектами](https://uk.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8) для [гнучкої розробки програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%93%D0%BD%D1%83%D1%87%D0%BA%D0%B0_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F). Скрам чітко робить акцент на якісному контролі процесу розробки. .[3]

Scrum — це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи — ScrumMaster, той хто опікується процесами, веде їх і працює як керівник [проекту](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82_(%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8)), Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників.

Протягом кожного спринту, 15-30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog ([документація](https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D1%96%D1%8F) запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана. Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту.

Scrum — це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи — ScrumMaster, той хто опікується процесами, веде їх і працює як керівник [проекту](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82_(%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8)), Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників. .[3]

Протягом кожного спринту, 15-30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog ([документація](https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D1%96%D1%8F) запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана. Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту.[3]

**Планування спринта (Sprint Planning Meeting)**

Проходить на початку нової ітерації Спринта.

Із Product Backlog обираються задачі, зобов'язання по виконанню яких за спринт приймає на себе команда; На основі обраних задач створюється Sprint Backlog. Кожна задача оцінюється у ідеальних людино-годинах; Рішення задачі не повинно займати більше 12 годин або одного дня. При необхідності задача розбивається на підзадачі; Обговорюється та визначається, яким чином буде реалізовано цей об'єм робіт; Тривалість наради обмежена зверху 4-8 годинами в залежності від тривалості ітерації, досвіду команди тощо; (перша частина наради) Беруть участь Product Owner + Команда: обирають задачі із Product Backlog; (друга частина наради) Бере участь лише команда: обговорюють технічні деталі реалізації, наповнюють Sprint Backlog

Щоденна нарада (Daily Scrum Meeting)

Відбувається кожен день протягом спринта. Є «пульсом» ходу спринта. Нараді властиві наступні обмеження:

починається точно вчасно;

всі можуть спостерігати, але говорять тільки «свині»;

триває не більш ніж 15 хвилин; проводиться в одному і тому ж місці протягом одного спринта. Протягом наради кожен член команди відповідає на 3 запитання:

* Що зроблено з моменту попередньої щоденної наради?
* Що буде зроблено з моменту поточної наради до наступної?
* Які проблеми заважають досягненню цілей спринта? (Над рішенням цих проблем працює ScrumMaster. Зазвичай це рішення проходить за рамками щоденної наради і у складі осіб, що безпосередньо займаються даною перешкодою.)

**Демонстрація (Sprint Review Meeting)**

Проходить у кінці ітерації (спринта). Команда демонструє внесок функціональності до продукту всім зацікавленим особам.

Залучається максимальна кількість глядачів.

Усі члени команди беруть участь у демонстрації (одна людина на демонстрацію або кожен показує, що зробив за спринт).

Обмежена 4-ма годинами в залежності від тривалості ітерації і змін у продукті.

**Ретроспектива (Sprint Retrospective)**

Члени команди висловлюють свою думку про минулий спринт.

Відповідають на два основних запитання:

Що було зроблено добре у минулому спринті?

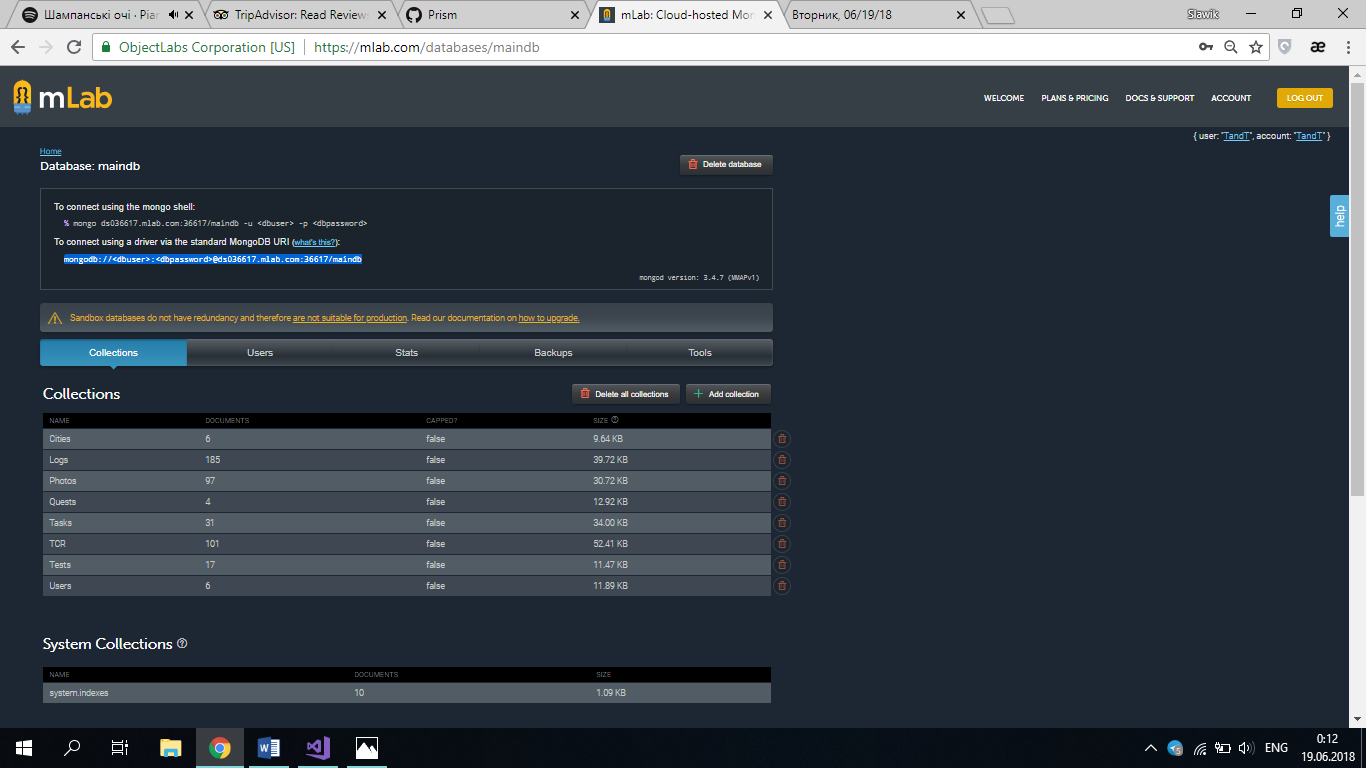
Що потрібно покращити в наступному?

Виконують покращення процесу розробки (вирішують питання та фіксують вдалі рішення).Обмежена 1-3ма годинами.

## **РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА**

**3.1 Проектування та розробка** **API**

У проекті була використана класична трьохрівнева архітектура

**Трирівнева архітектура** - у комп'ютерних технологіях трирівнева архітектура, синонім трьохслойна архітектура, передбачає наявність наступних компонент програми: клієнтський застосунок, або інтерфейс(UI), підключений до сервера застосунків(BLL), який в свою чергу підключений до серверу бази даних (DAL). 

*Рисунок 3 Список колекцій в MLab, які вкористовуються в проекті*

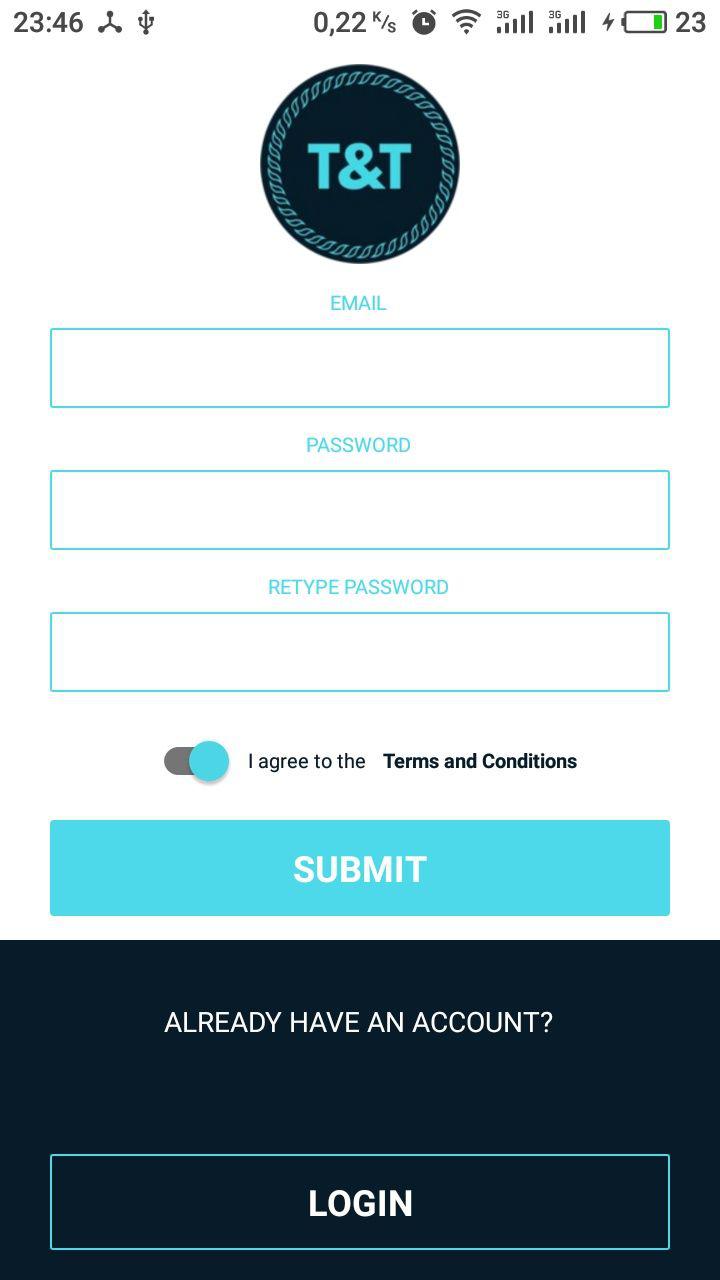
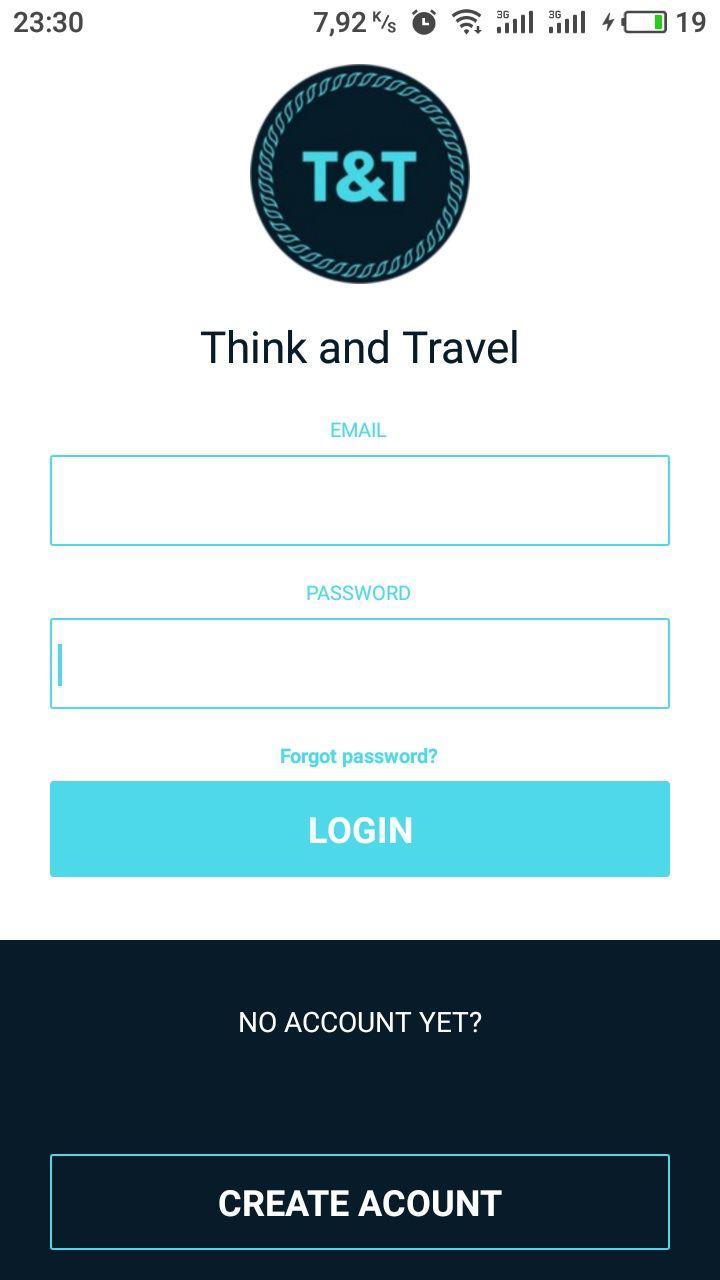
DAL - це слой сервісу, який реалізує взаємодію з NoSQL базами даних, причому:

Клас *DbSet<IBsonModel>* - це  Data access object , який взаємодіє з всіма колекціями даного типу, як з однією.

*IBsonModel* - інтерфейс для моделей даних які зберігаються в БД.

Основним класом є DataManager.cs який ініціалізую всі БД загружає всі DbSet та надає можливість взаємодіяти з ними, реалізує доступ до всіх DbSets.

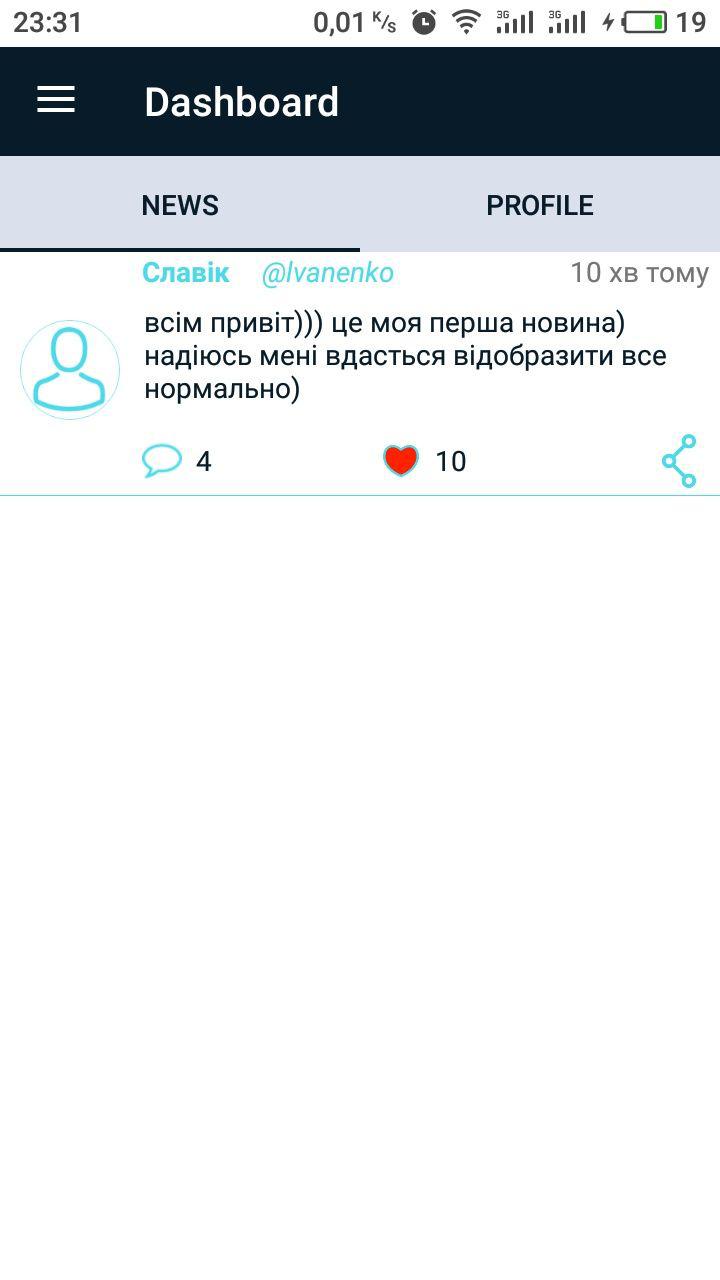
## BLL (Business logic layer) - це слой сервісу, в якому опрацьовується вся бізнес логіка сервісу, а саме підбір квестів, організація плані, реєстрація нових користувачів, ідентифікація користувачів і так далі.



*Рисунок 4 Сторінки ідентифікації та реєстрації користувачів*

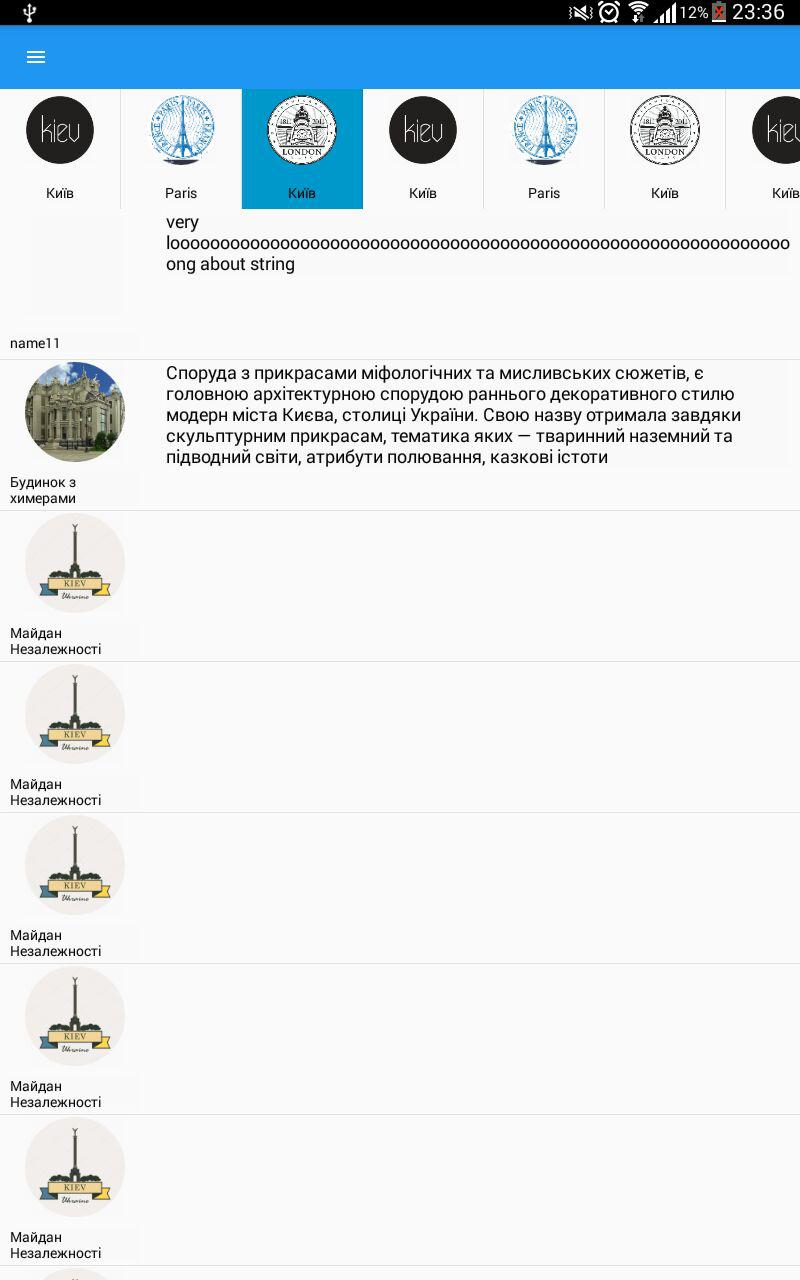
**3.2 Проектування та розробка** **мобільного додатку**

При проектуванні мобільного додатку було використано не стандарту, модульну архітектуру, для покращення стабільності, продуктивності та безпеки мобільного додатку.

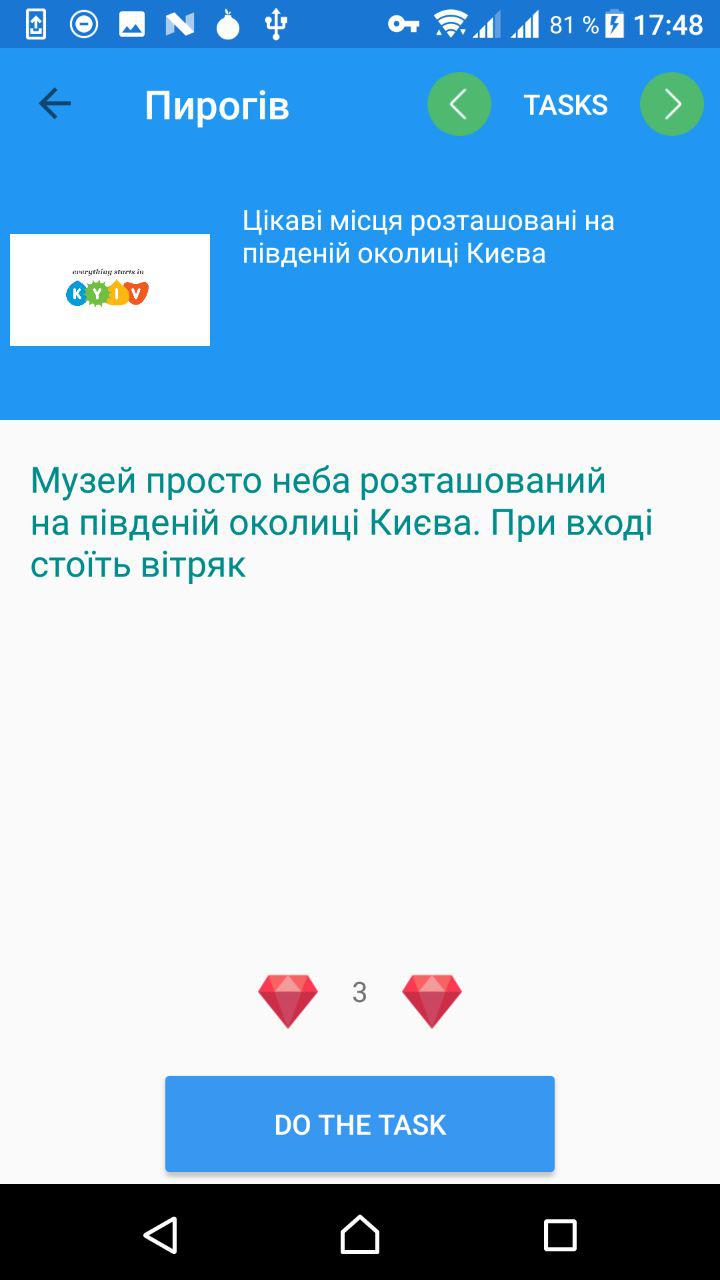


*Рисунок 5 Можливість ділитися своїми новинами з друзями*

Модульність в додатку реалізована через Prism. Це MVVM- фреймворк, особливістю якого є простота застосування, та простота розширення, за допомогою продуманої модульної архітектури.



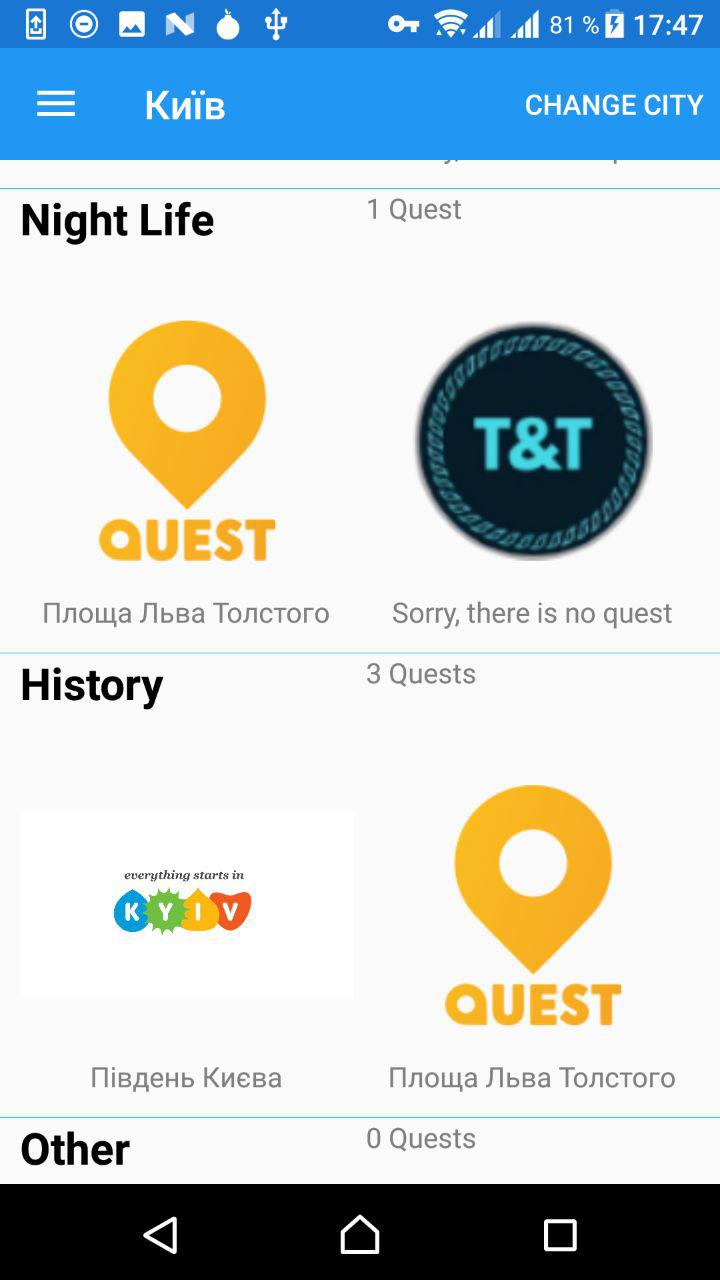
*Рисунок 6 Сторінка вибору квесту з фільтром по місту*



*Рисунок 7 Сторінки проходження квестів*

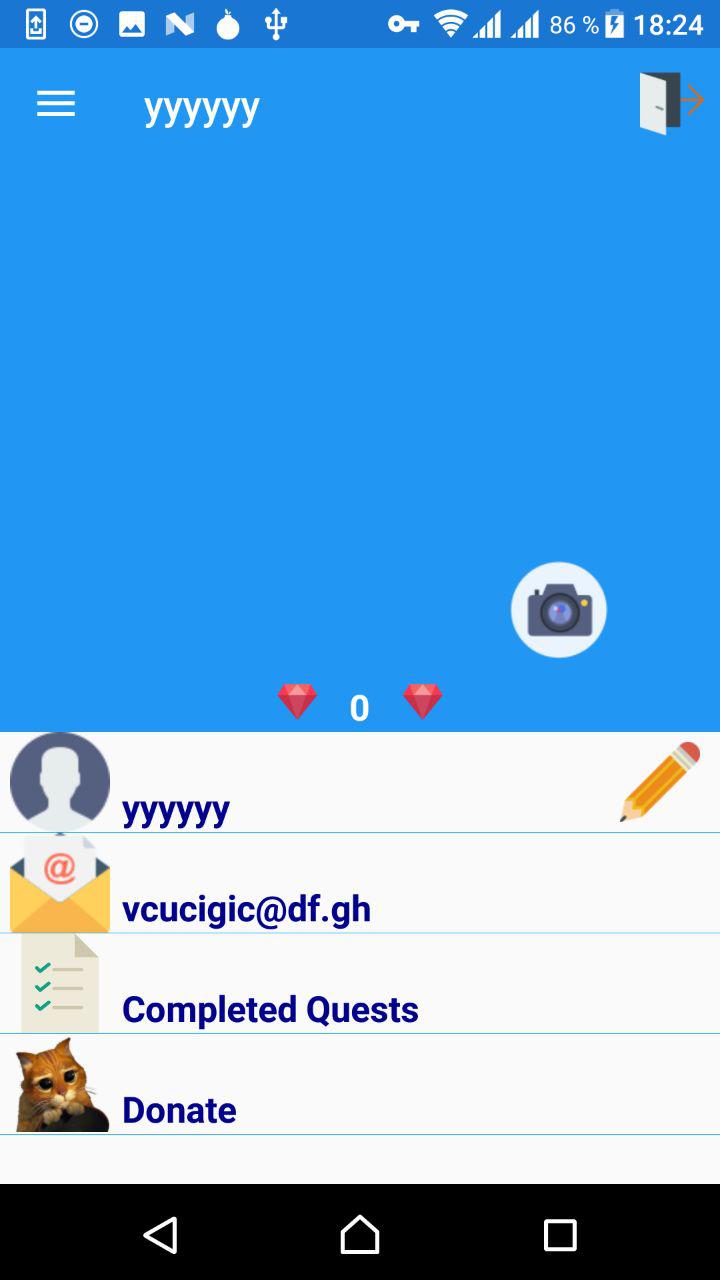
В додатку були виділені наступні модулі:

* Identity (реалізує функціональнвсть та сторінки для ідентифікації користувача);
* CustomControls (реалізує кастомні елементи та їх рендерери);
* Executions (реалізує функціональність та сторінки для виконання різних завдань квестів);
* Popups (реалізує функціональність та розмітку всіх спливаючих вікон додатку);
* Views (реалізує розмітку складних елементів списку);

**Сторінки модуля завантажуються в програму лише тоді, коли було до них звернення. Наприклад, сторінка для авторизації використовується не дуже часто, так як додаток зберігає останнього користувача.

*Рисунок 8 Сортування квестів за категорією для конкретного міста*

Також, всю бізнес логіку додатку було виділено в окрему бібліотеку класів – BLL. Для зручності, всі сервіси BLL було реалізовано, як статичні класи без гнучкого зв’язування. В такому випадку, всі модулі можуть користуватися тими ж сервісами і не потрібно реалізовувати механізм узгодженя різних варіантів сервісів.

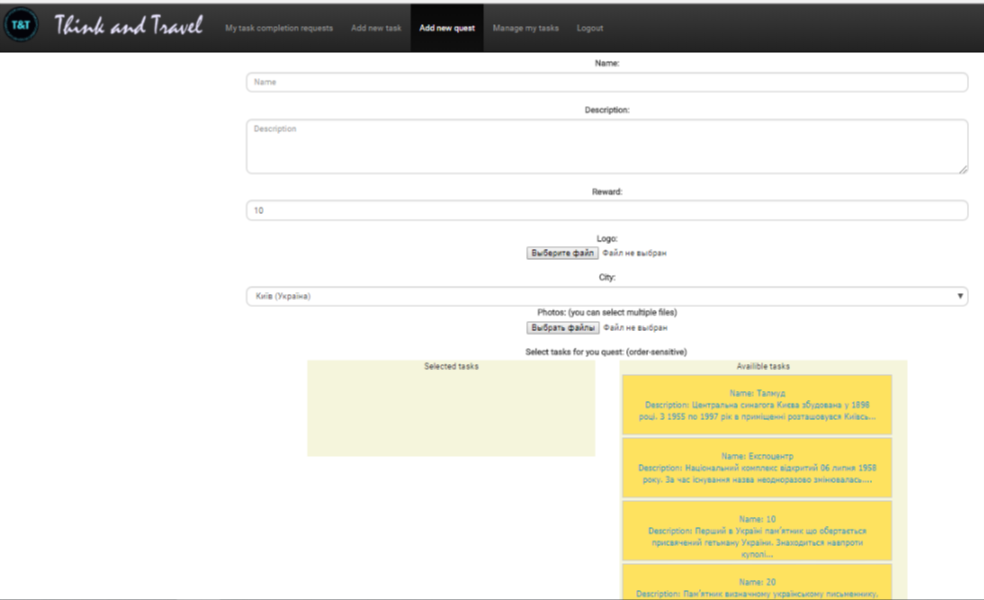
*Рисунок 9 Профіль користувача в додатку*

**3.3 Проектування та розробка сайту для адміністрування**

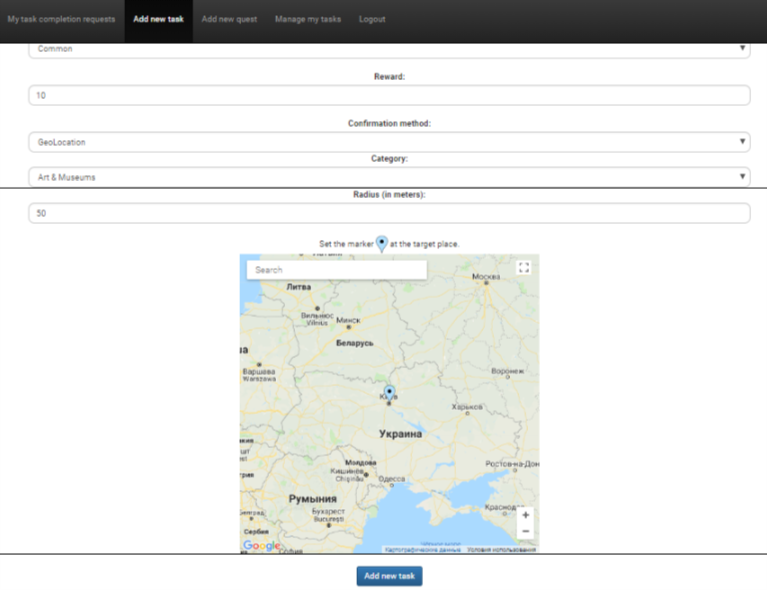
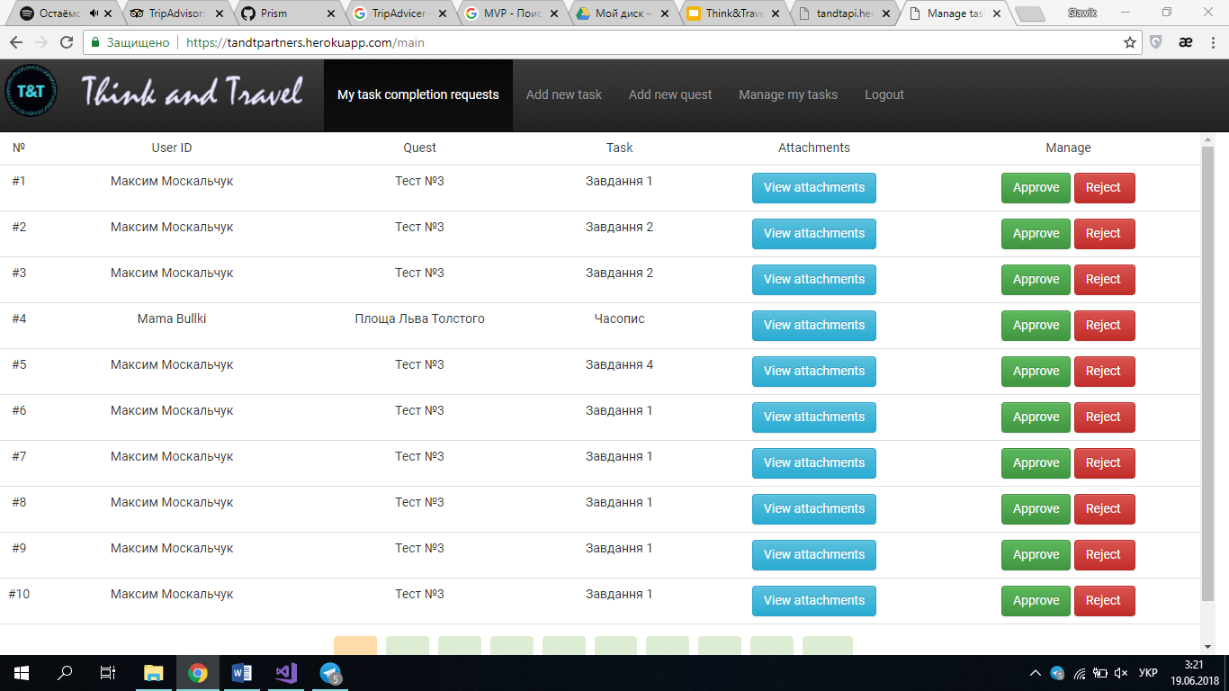
Важливим елементом даного сервісу є постійна підтримка користувачів, розробка і додавання нового контенту перевірка виконання завдань і так далі.

Вся ця функціональність покладена на сайт для адміністрування сервісу. Він розроблений з акцентом на безпеку, тому жоден адміністратор не зможе зайти на цей сайт, якщо невірно введе код підтвердження, яке приходить йому у вибраний меседжер через чат-бота.

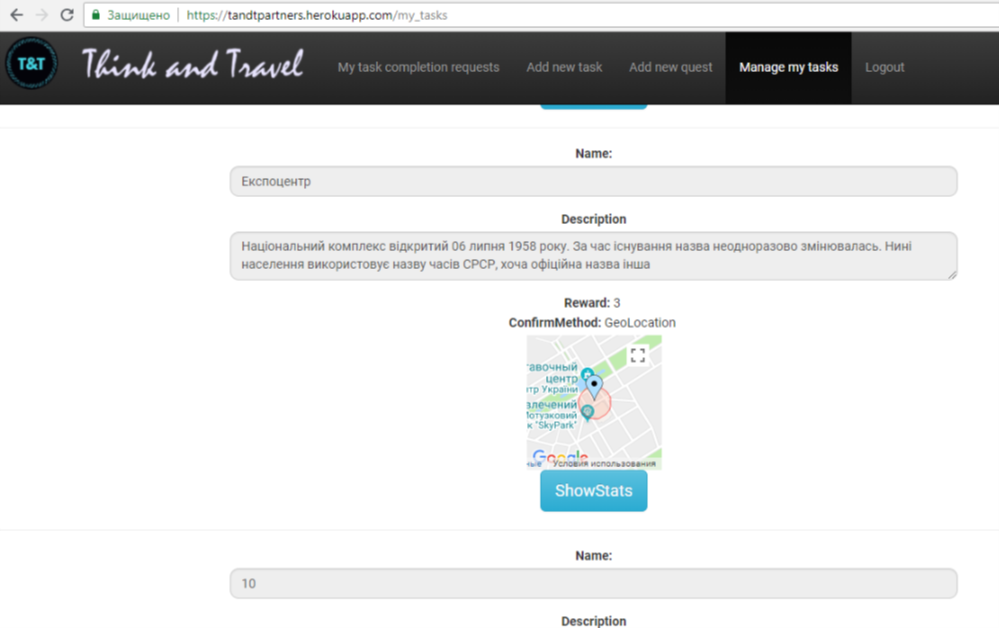
Функціональність сайту для адміністрування:

* Створювати нові квести;
* Сворення нових завдань з вибором способу підтвердження виконання;
* Підтвердження виконання завдань користувачами(«Ручне» підтвердження)
* **Керування створеними поточними завданнями через сайт адміністрування

*Рисунок 10 Створення квестів через сайт адміністрування*

*Рисунок 11 Створення завдань з підтвердженням через геолокацію*

*Рисунок 12«Ручне» підтвердження виконання користувачем завдання через сайт для адміністрування*

*Рисунок 13 Керування поточними завданнями через сайт адміністрування*

**3.4 Ідеї розвитку проекту**

**«Фото сервіс»**

Було розглянуто дуже цікава ідея, фотографувати користувачів під час проходження квестів, для подальшого продажу фотографій користувачам, в додатку. Було розроблено MVP версію додатку не пов’язаного з поточним сервісом. Також було проведено тестування ідеї на реальних користувачах, яке принесло не однозначні результати, так як жоден користувач не наважився на купівлю фотографії.

**AR**

Було розглянуто ідею внести в квести інтерактивність і цікавість, через застосування технології додаткової реальності. За ідеєю користувач міг отримати підказки по знаходженню потрібного місця в світі додаткової реальності, або отримати якусь інформацію від ожившої скульптури. Поки що дана ідея на стадії дослідження.

**Market**

Було розглянуто ідею створити на базі сервісу туристичний магазин де можна буде придбати туристичні путівки, різного роду квитки, приладдя для подорожей і так далі. На даний момент ця ідея на стадії реалізації, так як є хорошим способом монетизації проекту.

**Плани**

Було розглянуто ідею надати можливість користувачам планувати проходження квестів. Для того, щоб програмно можна було повідомити про це друзів, підібрати людей які будуть проходити разом, програмно врахувати погодні умови і завчасно попередити користувачів. Ідея знаходиться на стадії реалізації.

**Чат-боти**

Було розглянуто ідею розробити інтерфейс даного сервісу через чат-ботів, що на даний момент є дуже популярним і по оцінках експертів в майбутньому буде загальною тенденцією. Було розроблено тестового чат-бота, який виконував розсилку новин та повідомлень для користувача. Тестова версія отримала хороші відгуки від потенційних клієнтів сервісу.

## **ВИСНОВКИ**

В ході роботи було розглянуто і проаналізовано основні проблеми таких важливих галузей світового господарства як туризм і сфера послуг. Однією з найбільших проблем виявилось питання вдосконалення сервісної бази цих галузей.

Кінцевим результатом роботи став сервіс, який підбирає кожному користувачу набір найцікавіших локальних квестів, які він може пройти, оцінити, порадити друзям, або запланувати проходження в певній послідовності чи перенести на певний час.

Було зроблено «Альфа» версію додатка в Google Play Developers і поширено для внутрішніх тестувальників, які знаходили різного роду помилки функціональності та користувацького інтерфейсу. Також застосовувалося автоматичне тестування окремих модулів додатку (підключення до WebAPI,  сервіси бізнес логіки, Postman tests).

## **ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. <https://www.tripadvisor.com>
2. <https://uk.wikipedia.org/wiki/TripAdvisor>
3. https://uk.wikipedia.org/wiki/Scrum
4. https://en.wikipedia.org/wiki/Kotlin\_(programming\_language)
5. <https://en.wikipedia.org/wiki/React_(JavaScript_library)>
6. <https://www.xamarin.com/>
7. <https://reactjs.org/>
8. [Jeff Sutherland](https://www.google.com.ua/search?tbo=p&tbm=bks&q=inauthor:%22Jeff+Sutherland%22) [Scrum: The Art of Doing Twice the Work in Half the Time](https://books.google.com/books/about/Scrum.html?id=RoPZCwAAQBAJ&printsec=frontcover&source=kp_read_button&hl=en#v=onepage&q&f=false) 2008
9. https://github.com/aspnet/home
10. Метод Lean Startup для швидкого тестування ідей і вибора бізнес-моделі / Ерік Ріс. — М.: Альпина Паблішер, 2012. — 253 с.
11. <https://ru.wikipedia.org/wiki/TripAdvisor>
12. <https://metanit.com/sharp/xamarin/>
13. [Hans Kung, *Tracing the Way : Spiritual Dimensions of the World Religions*, Continuum International Publishing Group, 2006, page 248](https://books.google.com/books?id=sm0BfUKwct0C&pg=PA248&dq=kairouan+oldest+mosque+north+africa&hl=fr&ei=d1AWTdq0H5Gn8QPq9aCDBw&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCYQ6AEwAA#v=onepage&q=kairouan%20oldest%20mosque%20north%20africa&f=false)
14. Borowiecki, K.J. and C. Castiglione (2014). [Cultural participation and tourism flows: An empirical investigation of Italian provinces](http://ideas.repec.org/p/hhs/sdueko/2012_021.html)
15. Chalmers, William D. (2011). [*On the Origin of the Species Homo Touristicus: The Evolution of Travel from Greek Spas to Space Tourism*](https://books.google.com/books?id=nSXeWMz4jKQC&pg=PA291&dq=%22Centre+for+Tourism+and+Cultural+Change+%28CTCC%29%22&hl=fr&ei=3m3kTsTiHI_U4QSJ0aClBQ&sa=X&oi=book_result&ct=result&resnum=4&ved=0CEUQ6AEwAw#v=onepage&q=%22Centre%20for%20Tourism%20and%20Cultural%20Change%20%28CTCC%29%22&f=false)
16. [*"ASP.NET is part of a great open source .NET community"*](https://www.asp.net/open-source). [*Microsoft*](https://en.wikipedia.org/wiki/Microsoft). May 14, 2013.
17. Jeffrey T. Fritz. ["ASP.NET 5 is dead - Introducing ASP.NET Core 1.0 and .NET Core 1.0"](http://blogs.msdn.com/b/webdev/archive/2016/01/19/asp-net-5-is-dead-introducing-asp-net-core-1-0-and-net-core-1-0.aspx). *.NET Web Development and Tools Blog*. Retrieved 20 January 2016.
18. ["Upcoming Name Change for Windows Azure"](https://azure.microsoft.com/en-us/blog/upcoming-name-change-for-windows-azure/). Microsoft Azure. March 24, 2014. Archived from [the original](https://azure.microsoft.com/) on March 24, 2014. Retrieved August 29, 2016.
19. [*"Mobile Engagement - Microsoft Azure"*](https://azure.microsoft.com/en-us/services/mobile-engagement/). azure.microsoft.com*. Retrieved July 27, 2016*