**АНОТАЦІЯ**

**ABSTRAСT**

**ЗМІСТ**

[ВСТУП 7](#_Toc453441982)

[Розділ 1. Огляд програмного забезпечення для моніторингу та планування пасажирських авіаперевезень 8](#_Toc453441983)

[1.1. Опис предметної області 8](#_Toc453441984)

[1.2. Огляд аналогів 8](#_Toc453441985)

[1.3. Висновок 1](#_Toc453441987)3

[Розділ 2. Постановка задачі для розробки програмного забезпечення для моніторингу та планування пасажирських авіаперевезень 15](#_Toc453441988)

[2.1. Загальна постановка задачі 15](#_Toc453441989)

[2.2. Вибір інструментальних засобів 15](#_Toc453441990)

[2.3. Специфікація вимог до програмного продукту 17](#_Toc453441991)

[2.4. Висновок 25](#_Toc453441992)

[Розділ 3. архітектурА і Проектування ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ Для ОПРАЦЮВАННЯ та відображення спортивних подій і букмекерських ставок 26](#_Toc453441993)

[3.1. Архітектура програмного продукту. Опис впровадження архітектури MVP (Model View Presenter) в програмному продукті 26](#_Toc453441994)

[3.2. Проектування компонент та класів 27](#_Toc453441995)

[3.2.1 Проектування компонент та класів MVP (Model View Presenter) 27](#_Toc453441996)

[3.2.2 Проектування компонент та класів для взаємодії з мережею а також опрацювання спортивних подій та букмекерських ставок. 28](#_Toc453441997)

[3.3. Проектування алгоритмів 33](#_Toc453441998)

[3.4. Проектування графічного інтерфейсу користувача 35](#_Toc453441999)

[3.5. Висновок 38](#_Toc453442000)

[Розділ 4. Реалізація та тестування Програмного продукту 39](#_Toc453442001)

[4.1. Реалізація мобільного додатку 39](#_Toc453442002)

[4.2. Опис роботи з програмним продуктом 40](#_Toc453442003)

[4.3. Тестування програмного продукту 44](#_Toc453442004)

[4.4. Висновок 48](#_Toc453442005)

[РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА 49](#_Toc453442006)

[5.1. Економічна характеристика проектного рішення (програмного продукту) 49](#_Toc453442007)

[5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення товару. 50](#_Toc453442008)

[5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ. 51](#_Toc453442009)

[5.4. Формування стратегічних альтернатив 54](#_Toc453442010)

[5.5. Бюджетування 58](#_Toc453442011)

[5.6. Остаточний вибір стратегії 61](#_Toc453442012)

[Висновки 62](#_Toc453442013)

[список Літератури 63](#_Toc453442014)

[Додаток А. Блок-схема алгоритму опрацювання спортивних подій та букмекерських ставок 64](#_Toc453442015)

[Додаток Б. Діаграма варіантів використання 65](#_Toc453442016)

[Додаток В. Діаграма діяльності 66](#_Toc453442017)

[Додаток Д. Звіт по тестуванню 67](#_Toc453442018)

[Додаток Е. Інструкція користувача 71](#_Toc453442019)

[Додаток Ж. Лістинг коду 73](#_Toc453442020)

[Додаток З. Статичний аналіз коду за допомогою Android Studio 82](#_Toc453442021)

[Додаток И. Список багів та зауважень від замовника 83](#_Toc453442022)

[Додаток К. Результати тестування першої версії перед релізом 86](#_Toc453442023)

# ВСТУП

Сьогодні люди мають велику потребу у пересуванні, чи це з особистих причин, чи з професійних. Мільйони людей подорожують кожен день, для цього існує багато видів транспорту, але найшвидший — авіатранспорт.

Пасажирські авіаперевезення – це найшвидший та найкомфортніший транспорт, адже можна подолати лічені години промайнувши кілька тисяч кілометрів. Різноманітні авіакомпанії надають цікаві послуги та сервіси в обслуговуванні клієнтів. Часто буває що клієнтові потрібно переглянути весь список авіарейсів, або переглянути стасус якогось рейсу, або замовити квиток не виходячи з домівки, для цього всього потрібно їхати в аеропорт, тратити свій час та запитувати все у асистентів.

Таким чином з’явилась необхідність у розробці програмного продукту для моніторингу та планування пасажирських авіаперевезень, який суттєво зекономить час клієнтів та дозволить здійснити усі вище згадані послуги не покинувши домівки.

# Розділ 1. Огляд програмного забезпечення для моніторингу та планування пасажирських авіаперевезень

## 1.1. Опис предметної області

Подорож — переміщення якоюсь певною територією з метою її вивчення, а також із загальноосвітньою, пізнавальною, спортивною цілями. Подорожі відіграють велику роль у житті людини, а з появою авіалайнерів вони стали набагато швидшими.

Загалом подорож літаком має такі переваги:

— Найважливішою перевагою подорожей повітряним транспортом є його висока швидкість пересування. Всього за кілька годин можна перетнути тисячі кілометрів і опинитися в потрібному вам місці. Особливо це зручно при обмеженому часі.

— Подорож літаком не здається такою нудною, як, наприклад, на поїзді або автобусі. Адже тут не потрібно декілька днів витрачати на поїздку.

— Літаки відрізняються високим сервісом. Пасажиру під час польоту надається їжа, напої. До того ж тут пропонуються всілякі способи, що дозволяють скоротати час, наприклад, подивитися фільм, ознайомитися з пресою або погортати журнал.

— Під час польоту пасажир відчуває себе максимально комфортно і зручно, що забезпечується зручними сидіннями, що дозволяють чудово відпочити і навіть поспати.

— Дуже вражає вид з ілюмінатора, красиві пейзажі, розташовані внизу або білосніжні хмари, що огортають літак.

— Сьогодні існує унікальна можливість забронювати квиток по телефону або Інтернету і забути про незліченні черги. Причому, доступна така послуга у будь-який час, найбільш зручний для вас.

— Літаком можна відправитися в будь-яку країну, куди не завжди можна дістатися залізничним або автомобільним транспортом.

Дивлячись на всі ці переваги нe дивно, що сьогодні люди в основному подорожують на літаку. Він поєднує в собі комфорт і швидкість, ви доберетеся до місця призначення дуже швидко. Без сумніву, подорож літаком – найзручніше і комфортний засіб пересування. [1]

## 1.2. Огляд аналогів

Переглянувши декілька аналогів мною було вибрано три найяскравіші приклади:

1. WizzAir —  [угорсько-польська бюджетна авіакомпанія, із вільни](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%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)м [веб-застосунок](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)ом для користування послугами компанії. [2]

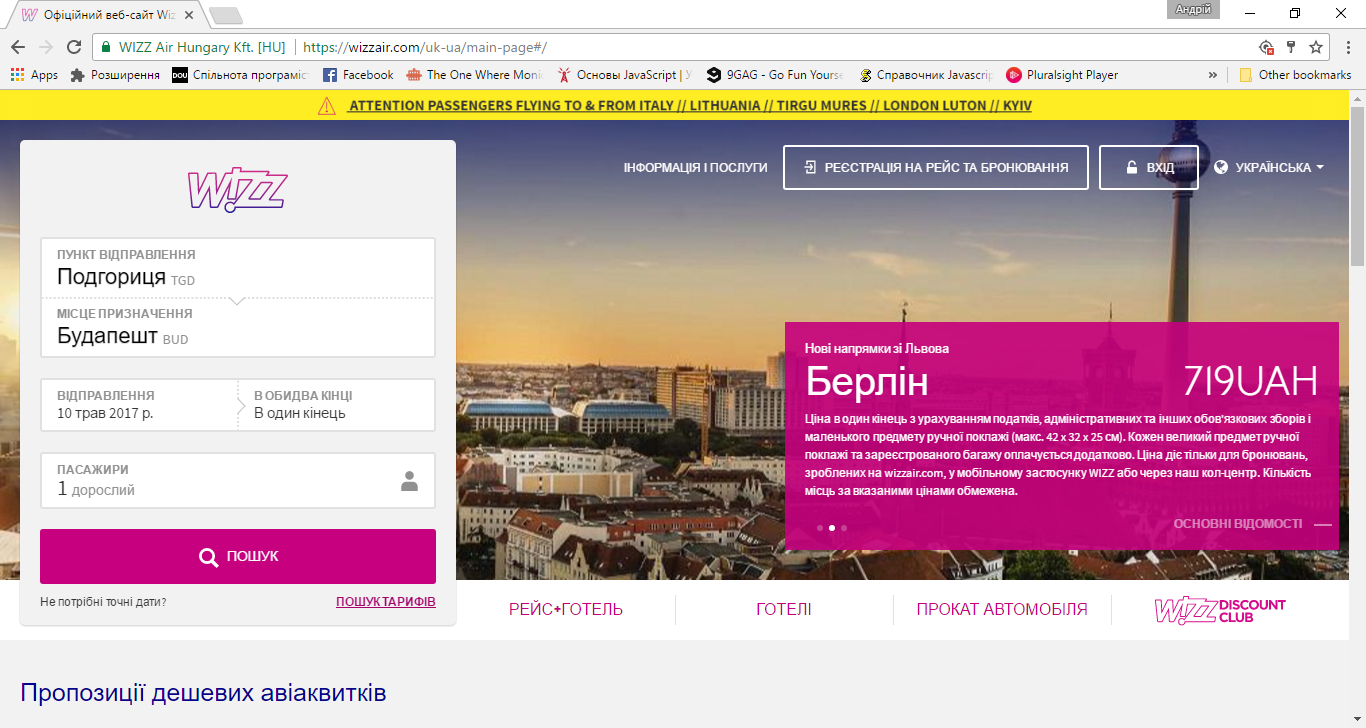


Рис. 1.1 Головна сторінка сервісу WizzAir



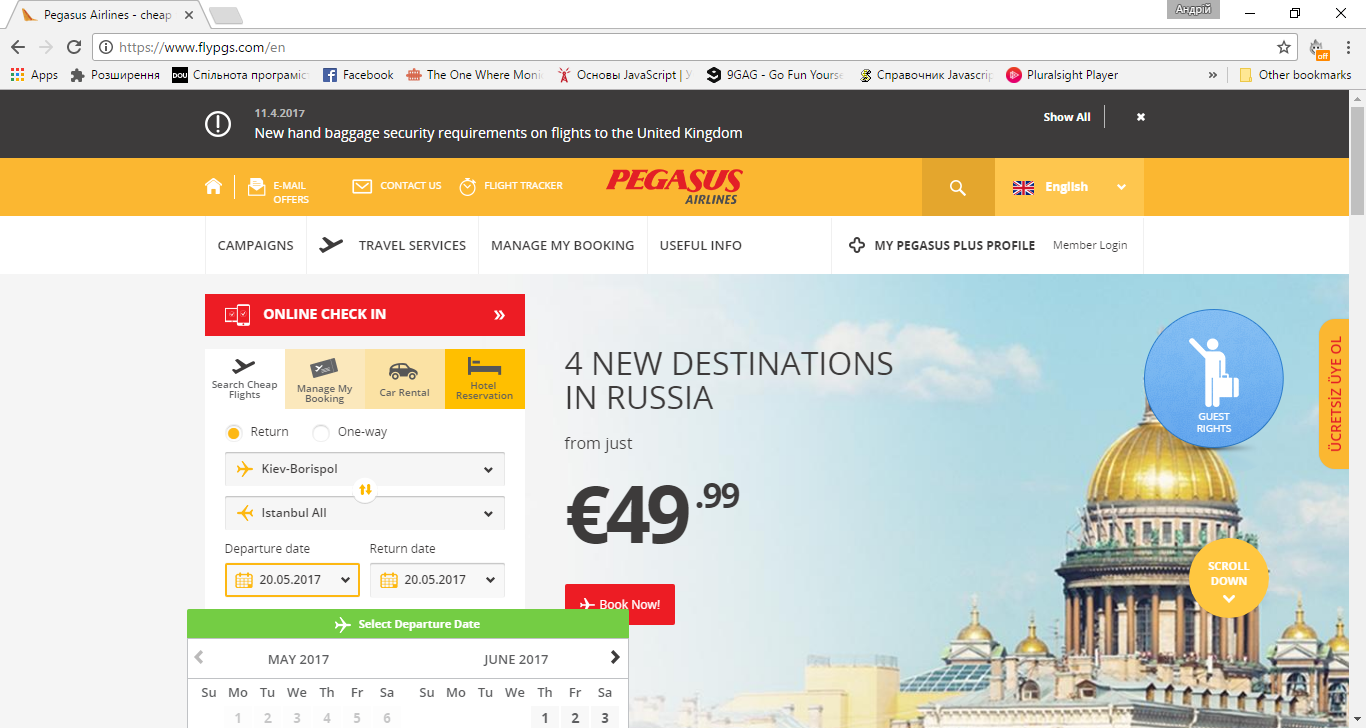
Рис. 1.2. Сторінка результату пошуку рейсів та подальшим замовленням сервісу WizzAir

Переваги:

* Можливість додаткових послуг, таких як: бронювання готелів, виклик таксі та перегляд найблищих закладів харчування.
* Наявна українська локалізація
* Синхронізація із соціальними мережами
* Величезна кількість інформації про компанію та авіаперевезення.

Недоліки:

* Відслідковування статусу рейсу

1. Pegasus Airlines — бюджетна міжнародна авіакомпанія заснована в Туреччині, із вільним у користуванні веб-застосунком.Рис. 1.3. Головна сервісу Pegasus Airlines

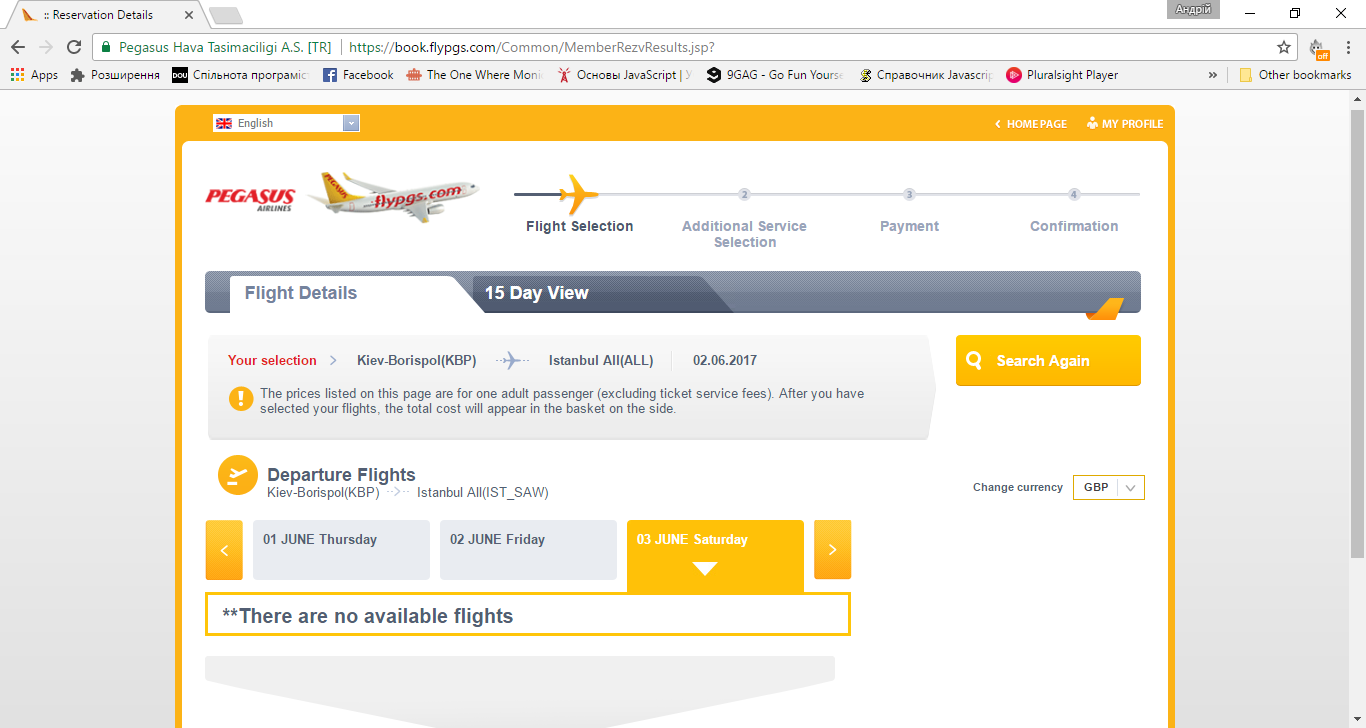


Рис. 1.4. Сторінка результату пошуку рейсів та подальшим замовленням сервісу Pegasus Airlines

Переваги:

* Простота і зручність інтерфейсу
* Пропозиції найпопулярніших рейсів
* Величезна кількість інформації про компанію та авіаперевезення

Недоліки:

* Немає укрїнської локалізації
* Синхронізація із соціальними мережами

British Airways — найбільша авіакомпанія та національний авіаперевізник Великої Британії, одна з найбільших в Європі, [із вільни](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%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)м серверним [веб-застосунок](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)ом для користування послугами компанії. [3]

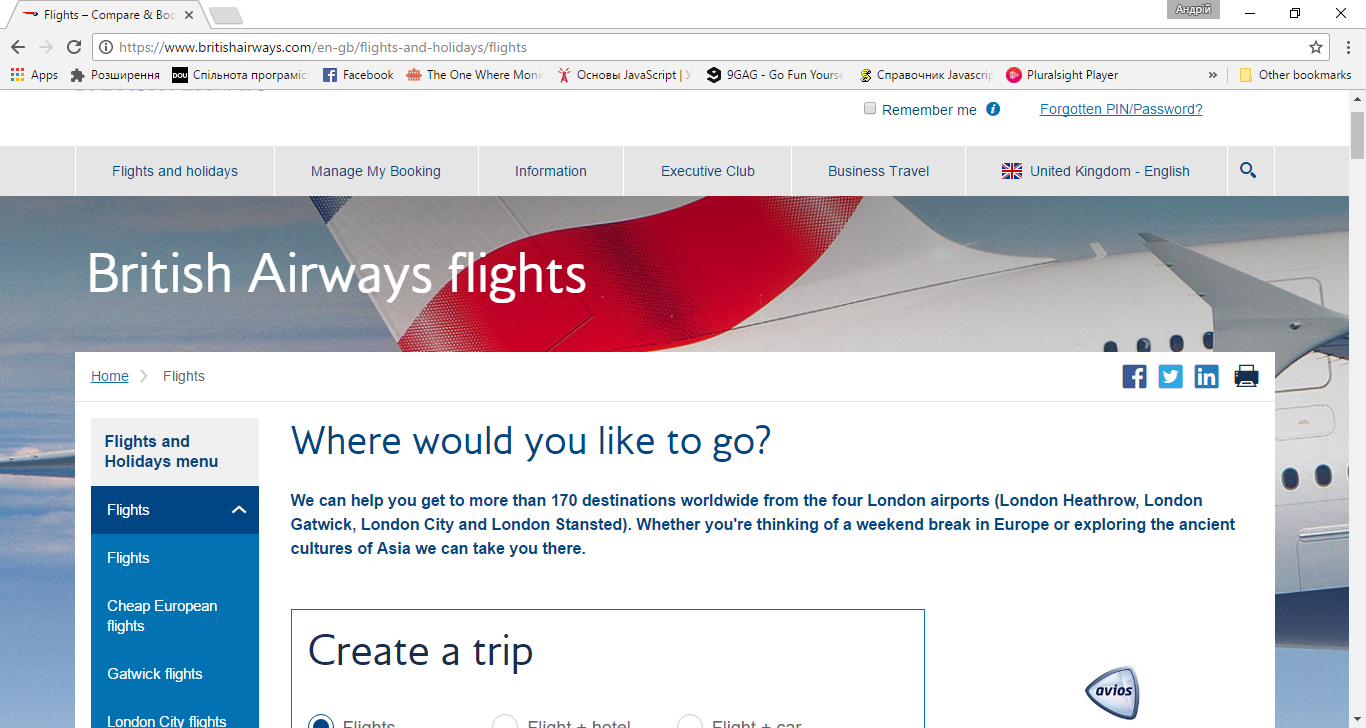


Рис. 1.5. Сторінка пошуку рейсів сервісу British Airways

Переваги:

* Простота і зручність інтерфейсу.
* Пропозиції найдешевших квитків.
* Відслідковування статусу рейса.
* Синхронізація із соціальними мережами
* Величезна кількість інформації про компанію та авіаперевезення.

Недоліки:

* Немає української локалізації

Таблиця 1.1

Порівняння функціоналу

|  |  |  |  |
| --- | --- | --- | --- |
| Функціонал | WizzAir | Momondo | British Airways |
| Відслідковування статусу рейса | - | + | + |
| Наявність інформації про конмпанію та авіаперевезення | + | + | + |
| Українська локалізація | + | + | - |
| Пропозиції додаткових послуг | + | + | + |
| Пропозації найпопулярніших рейсів | - | + | + |
| Присутність реклами | - | + | - |
| Синхронізація із соціальними мережами | + | - | + |

## 1.3. Висновок

В результаті огляду програмного забезпечення моніторингу та планування пасажирських авіаперевезень було проаналізовано предметну область. Описано головні тенденції та потреби. Також проаналізовано декілька продуктів аналогів. В результаті огляду аналогів було виявлено переваги та недоліки, які будуть братись до уваги при розробці програмного забезпечення.

# Розділ 2. Постановка задачі для розробки програмного забезпечення для моніторингу та планування пасажирських авіаперевезень

## 2.1. Загальна постановка задачі

Завданням бакалаврської кваліфікаційної роботи є створення програмного забезпечення для моніторингу та планування авіарейсів для авіакомпанії у вигляді веб сайту.

Цей сайт суттєво зекономить час користувачів, так як їм не прийдеться стояти у величезних чергах для замовлення квитків, а також не потрібно дзвонити у службу підтримки для отримання докладної інформації.

Основними можливостями веб сайту будуть:

* Пошук авіарейсів.
* Перегляд детальної інформації про авіарейс.
* Бронювання або замовлення квитка на вибраний рейс.
* Відслідковування статусу рейсу.
* Реєстрація та авторизація в системі.

Програмне забезпечення складатиметься з front-end(веб-браузер) та back-end(RESTfull API) частин.

Front-end частина буде використовувати основні веб-браузери такі як: Google Chrome, Mozilla Firefox, та наступні технології: HTML5\CSS3, Java Script та фреймворк AngularJS[5].

Back-end частина буде реалізована у вигляді RESTfull API, яке буде написано на об’єктно-орієнтованій мові C#. Для збереження даних слугуватиме MS SQL база даних.

## 2.2. Вибір архітектури

Для написання продукту буде використовуватись архітектура клієнт-сервер[4].

Архітектура клієнт-сервер являє собою розподілену структуру програми, яка розділяє завдання або робочі навантаження між постачальниками ресурсу або послуг, які називаються серверами і послуг робить запит, званих клієнтів. Часто клієнти і сервери взаємодіють через комп'ютерну мережу на окремому обладнанні, але клієнт і сервер можуть перебувати в одній і тій же системі. Хост-сервер працює один або кілька серверних програм, які діляться своїми ресурсами з клієнтами. Клієнт не поділяє будь-якої зі своїх ресурсів, але вимагає контент або сервіс функції сервера. Таким чином, клієнти ініційовують сеанси зв'язку з серверами, які очікують вхідних запитів. Приклади комп'ютерних програм, які використовують модель клієнт-сервер є електронна пошта, мережевий друк, і World Wide Web.

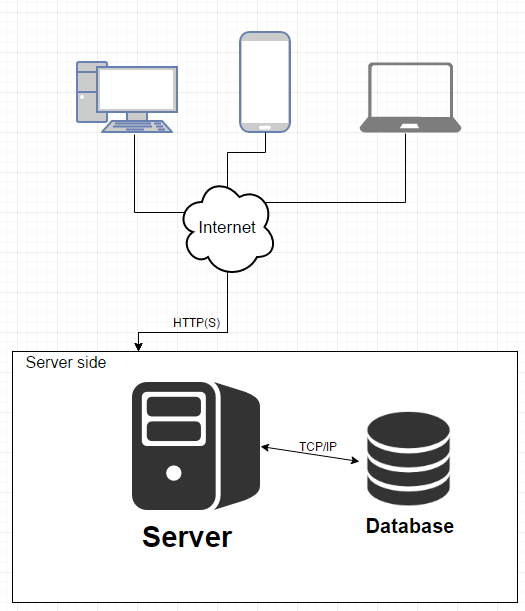


Рис. 2.1. Приклад клієнт серверної архітектури

Клієнти і сервери обмінюються повідомленнями в шаблоні обміну повідомленнями запит-відповідь. Клієнт посилає запит, а сервер повертає відповідь. Цей обмін повідомленнями є прикладом взаємодії між процесами. Для зв'язку, комп'ютери повинні мати спільну мову, і вони повинні слідувати правилам, так, щоб і клієнт, і сервер знали, чого очікувати. Мова і правила спілкування визначені в протоколі зв'язку. Всі протоколи клієнт-сервер працюють на рівні додатків. Протокол прикладного рівня визначає основні закономірності діалогу. Для того, щоб ще більше оформити обмін даних, сервер може реалізувати інтерфейс прикладного програмування (API). API представляє собою рівень абстракції для доступу до служби. Обмежуючи комунікацію до певного вмісту формату, це полегшує розбір. Абстрагуючись доступ, це полегшує обмін даними крос-платформної.

## . Вибір програмних засобів

Для створення програмного продукту на серверній частині було обрано технологію .Net, середовище розробки Microsoft Visual Studio, мова C#[8] та фреймворк ASP.NET Web API2[6]. Також для маніпуляцією з базою даних використовуватиметься Entity Framework[7]. Для збереження даних слугуватиме MS SQL база даних.

На клієнтській було обрано такі технології як: HTML5\CSS3, JavaScript, Bootstrap, фреймворк AngularJS та середовище розробки Web Storm.

 .NET Framework є основою програмного забезпечення, розроблена корпорацією Майкрософт, яка працює в основному на Microsoft Windows. Вона включає в себе велику бібліотеку класів під назвою Framework Class Library (FCL) і забезпечує мовну сумісність (кожна мова може використовувати код, написаний на інших мовах) на кілька мов програмування. Програми, написані для .NET Framework виконуються в програмному середовищі (на відміну від апаратної середовища) під назвою Common Language Runtime (CLR), віртуальна машина додаток, яке надає такі послуги, як безпека, управління пам'яттю і обробки винятків. (Як, наприклад, комп'ютерний код, написаний з використанням .NET Framework називається «керований код».) F і CLR разом становить .NET Framework.

C # — є мовою програмування мульти-парадигма охоплює строгу типізованість, імперативність, декларативність, функціональні, загальні, об'єктно-орієнтовані (на основі класів), компонент-орієнтоване програмування дисциплін. Він був розроблений Microsoft в рамках своєї .NET ініціативи, а потім затверджений як стандарт Ecma (ECMA-334) і ISO (ISO / IEC 23270: 2006). C # є одним з мов програмування, призначених для Common Language Infrastructure.

JavaScript (JS) — це високорівнева, нетипізована і інтерпративна та динамічна мова програмування. Він був стандартизований в специфікації мови ECMAScript. Поряд з HTML і CSS, JavaScript є одним з трьох основних технологій світового виробництва Wide Web контенту; більшість сайтів використовують його, і всі сучасні веб-браузери підтримують його без необхідності плагінів. JavaScript є прототипом основи з функціями першого класу, що робить його мову мульти-парадигму, підтримка об'єктно-орієнтовані, імперативні і функціональні стилі програмування. Він має API для роботи з текстами, масивами, датами і регулярними виразами, але не включає в себе введення / виведення, такі як мережі, зберігання або графічні об'єкти, спираючись на них на приймаючої середовище, в якій він вбудований.

Хоча є сильні зовнішні подібності між JavaScript і Java, в тому числі назви мови, синтаксису і відповідні стандартні бібліотек, це два різні мови і сильно відрізняються за своєю конструкцією.

## 2.4. Специфікація вимог до програмного продукту

1. * 1. **Вступ**
        1. **Призначення, мета**

Призначення даного продукту – дати можливість користувачам моніторити та планувати авіарейси, тобто замовляти або бронювати квитки, а також відслідковувати статус рейсу.

Мета – надати користувачеві інструмент для пошуку та замовлення або бронювання квитків, а також відслідковувати статус рейсу.

* + 1. **Загальний опис**
       1. **Характеристики продукту**

Функції, що будуть реалізовані у програмі:

* Пошук авіарейсів.
* Перегляд детальної інформації про авіарейс.
* Бронювання або замовлення квитка на вибраний рейс.
* Редагування даних в особистому кабінеті.
* Відслідковування статусу рейсу.
* Реєстрація та авторизація в системі.

Діаграма варіантів використання системи наведена в Додатку А.

* + - 1. **Класи користувачів та їх характеристики**

|  |  |
| --- | --- |
| 1. Зареєстрований користувач – це користувач, який є зареєстрованим в системі. | * Пошук рейсів. * Перегляд інформації про рейс. * Замовлення або бронювання квитків. * Відслідковування статусу рейсів. * Зміна персональної інформації. |
| 1. Адміністратор – це користувач, який є зареєстрованим в системі та має найбільші права в системі | * Пошук рейсів. * Перегляд інформації про рейс. * Замовлення або бронювання квитків. * Відслідковування статусу рейсів. * Зміна персональної інформації. * Відслідковування та зміна статусу рейсів. * Редагування рейсів. |
| 1. Незареєстрований користувач – це користувач, який не є зареєстрованим в системі. | * Реєстрація в системі. * Пошук рейсів. * Перегляд інформації про рейс. * Відслідковування статусу рейсів. |

* + - 1. **Середовище функціонування**

Програмний продукт передбачає такі апаратні та програмні вимоги до пристрою:

* Браузери: Google Chrome, Mozilla Firefox;
* Наявність підключення до мережі Internet.
  + 1. **Характеристики системи**
       1. **Пошук рейсів**
          1. Опис і пріоритет

Відображення списку за вибраними параметрами(дата, пункт відправлення, пункт прибуття і т.д.) для подальшого замовлення квитка.

Пріоритет високий.

* + - * 1. Послідовності дія/відгук
  1. Відкриття головної сторінки.
  2. Користувач вводить дані для пошуку рейсів та натискає кнопку «Пошук».
  3. Перехід на сторінку з завантаженим списком усіх можливих рейсів для заданих параметрів.
     + - 1. Функціональні вимоги

REQ-1: Завантаження списку доступних рейсів.

REQ-2: Усі поля повинні бути заповнені.

* + - 1. **Пререгляд детальної інформації про рейс**
         1. Опис і пріоритет

Після пошуку рейсів користувач вибирає конкретний рейс для перегляду додаткової інформації.

Пріоритет середній.

* + - * 1. Послідовності дія/відгук

1. Після пошуку рейсів кориситувач вибирає бажаний рейс клікнувши на посилання «Детальніше».
2. Користувач переглядає детальну інформацію
   * + 1. **Реєстрація в системі**
          1. Опис і пріоритет

Користувач реєструється у системі для отримання додаткової функціональності.

Пріоритет високий.

* + - * 1. Послідовності дія/відгук

1. Перехід на сторінку реєстрації клікнувши на посилання «Реєстрація» у верхній панелі.
2. Користувач заповнює усі поля.
3. Програма перевіряє введені дані.
4. У разі неправильних даних виводиться відповідне повідомлення.
5. Система відображає результат реєстрації.
   * + - 1. Функціональні вимоги

REQ-1: Усі обов’язкові поля повинні бути заповненні.

* + - 1. **Вхід в систему**
         1. Опис і пріоритет

Вхід користувача в систему та надання відповідних прав.

Пріоритет високий.

* + - * 1. Послідовності дія/відгук

1. Перехід на сторінку реєстрації клікнувши на посилання «Вхід» у верхній панелі.
2. Користувач вводить email та пароль.
3. Користувач натискає кнопку «Вхід»
4. Система повідомляє про результат входу.
   * + - 1. Функціональні вимоги

REQ-1: Усі поля повинні бути заповнені.

* + - 1. **Редагування особистого кабінету**
         1. Опис і пріоритет

Користувач має можливість редагувати особисті дані.

Пріоритет низький.

* + - * 1. Послідовності дія/відгук

1. Користувач повинен увійти в систему та перейти в особистий кабінет клікнувши на посилання «Особистий кабінет»
2. На сторінці особистого кабінету натиснути на кнопку «Налаштування».
3. Зявляється додаткове вікно, де користувач змінює дані та натискає на кнопку «Змінити».
4. Зявляється повідомлення про те що дані змінено успішно.
   * + - 1. Функціональні вимоги

REQ-1. Користувач повинен бути авторизований.

REQ-2. Введені дані мають бути конкретного типу.

REQ-3. Можливість скасування операції.

* + - 1. **Замовлення або бронювання квитка**
         1. Опис і пріоритет

Користувач замовляє або бронює квиток.

Пріоритет високий.

* + - * 1. Послідовності дія/відгук

1. Здійснивши пошук та вибравши бажаний рейс користувач натискає на кнопку замовити.
2. Користувач заповнює усі поля.
3. Також вибирає місця натиснувши на кнопку «Обрати місце».
4. Вибравши місце напискає кнопку «Гаразд».
5. Користувач натискає на кнопку «Продовжити».
6. Зявляється повторна інформація про замовлення та в залежності від вибраного методу реєстрації(Онлайн/Аеропорт) зявляється або форма заповнення кредитної картки(Онлайн метод) або кнопка «Забронювати»
7. Користувач заповняє форму з кредитною карткою і натискає кнопку «Замовити»(для онлайн режиму) або натискає кнопку «Забронювати»(для аеропорт режиму).
8. Зявляється повідомлення про успішно виконане замовлення.
   * + - 1. Функціональні вимоги

REQ-1. Користувач повинен бути авторизований.

REQ-2. Введені дані мають бути конкретного типу.

REQ-3. Можливість скасування операції.

* + - 1. **Статус рейсу**
         1. Опис і пріоритет

Користувач має можливість переглянути статус рейсу

Пріоритет високий.

* + - * 1. Послідовності дія/відгук

1. Користувач переходжить на сторінку «Статус рейсу», натиснувши на посилання «Статус рейсу» у верхній панелі
2. Користувач вибирає пункт відправлення/прибуття та дату і натискає на кнопку «Пошук».
3. Користувач відслідковує статус рейсу.
   * + - 1. Функціональні вимоги

REQ-1. Користувач повинен вибрати/заповнити всі поля.

* + - 1. **Список рейсів**
         1. Опис і пріоритет

Користувач може переглядати список усіх рейсів, а також фільтрувати їх.

Пріоритет середній.

* + - * 1. Послідовності дія/відгук

1. Користувач переходжить на сторінку «Список рейсів», натиснувши на посилання «Список рейсів» у верхній панелі.
2. Користувач переглядає список усіх можливих рейсів.
3. Користувач використовує фільтри та пошук для знаходження потрібного рейсу.
   * + - 1. Функціональні вимоги

REQ-1. Користувач повинен вибрати/заповнити всі поля.

* + - 1. **Список замовлень**
         1. Опис і пріоритет

Користувач може переглядати список своїх замовлень, а також фільтрувати їх.

Пріоритет високий.

* + - * 1. Послідовності дія/відгук

1. Користувач переходжить на сторінку «Список замовлень», натиснувши на посилання «Список замовлень» у верхній панелі.
2. Користувач переглядає список своїх замовлень.
3. Користувач використовує фільтри та пошук для знаходження потрібного замовлення.
   * + - 1. Функціональні вимоги

REQ-1. Користувач повинен бути авторизованим у системі.

* + - 1. **Пререгляд детальної інформації про замовлення**
         1. Опис і пріоритет

Після перегляду замовлень користувач вибирає бажане замовлення для перегляду детальної інформації про замовлення.

Пріоритет високий.

* + - * 1. Послідовності дія/відгук

1. Після пошуку рейсів кориситувач вибирає бажаний замовлення клікнувши на посилання «Детальніше».
2. Користувач переглядає детальну інформацію
   * + - 1. Функціональні вимоги

REQ-1. Користувач повинен бути авторизованим у системі.

* + 1. **Вимоги зовнішніх інтерфейсів**
       1. **Користувацькі інтерфейси**

Інтерфейс програми повинен бути простим для розуміння, забезпечувати швидкодію відповідно до вимог та дозволяти зручно виконувати необхідні операції.

* + - 1. **Апаратні інтерфейси**

Апаратні інтерфейси не будуть використовуватися цією програмною системою.

* + - 1. **Програмні інтерфейси**
* Visual Studio
* Entity Framework
* Web Storm
* MS SQL Server
  + - 1. **Комунікаційні інтерфейси**

Клієнтська частина системи буде спілкуватися з серверною за допомогою протоколу HTTP.

* + 1. **Інші нефункціональні вимоги**
       1. **Вимоги продуктивності**

Швидкість завантаження та опрацювання даних буде прямо залежати від швидкості передачі даних в мережі та від кількості даних, які отримуються з мережі.

* + - 1. **Вимоги безпеки**

Програмний продукт немає вимог безпеки.

* + - 1. **Атрибути якості програмного продукту**
* Зручність використання.
* Надійність.
* Супроводжуваність.
  + - 1. **Інші вимоги**

Українська локалізація.

## 2.5. Висновок

В результаті постановки задачі для розробки програмного забезпечення для моніторингу та планування пасажирських авіаперевезень для авіакомпанії було вирішено розробити веб систему яка буде дійсна у всіх сучасних браузерах (таких як Google Chrome та Mozilla Firefox), також вибрано архітектуру(клієнт-сервер), технології та інструментальні засоби для розробки програмного продукту додаток. Для серверної частини використовуватиметься .Net технології, а саме WebApi2 та Entity Framework для надійності та швидкості розробки продукту, середовище розробки Visual Studio, для зберігання даних MS SQL Server. Клієнтська частина буде використовувати стандартні веб-технологіх(HTML5, CSS3, JavaScript), а також фреймворк AngularJS та середовище розробки Web Storm.

Також було сформовано специфікаю вимог для розробки програмного забезпечення моніторингу та планування пасажирських авіаперевезень.

# Розділ 3. архітектурА і Проектування ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ Для моніторингу та планування пасажирських авіаперевезень

## 3.1. Архітектура програмного продукту. Опис впровадження клієнт серверної архітектури, а також MVC шаблону в програмному продукті

Програмний продукт побудований на використанні клієнт серверної архітектури, тобто існує окремо серверна частина(RESTfull API) та клієнтська частина.

Архітектура клієнт – сервер (client-server architecture) – це концепція інформаційної мережі, в якій основна частина її ресурсів зосереджена в серверах, обслуговуючих своїх клієнтів. Розглянута архітектура визначає два типи компонентів: сервери і клієнти.

 Сервер – це об’єкт, що дає сервіс іншим об’єктам мережі за їх запитами. Сервіс – це процес обслуговування клієнтів. Сервер працює за завданнями клієнтів і управляє виконанням їх завдань. Після виконання кожного завдання сервер посилає отримані результати клієнту, який послав це завдання. Сервісна функція в архітектурі клієнт – сервер описується комплексом прикладних програм, відповідно до якого виконуються різноманітні прикладні процеси.

Процес, який викликає сервісну функцію за допомогою певних операцій, називається клієнтом. Ним може бути програма або користувач. Клієнти – це робочі станції, які використовують ресурси сервера і надають зручні інтерфейси користувача. Інтерфейси користувача це процедури взаємодії користувача з системою або мережею.

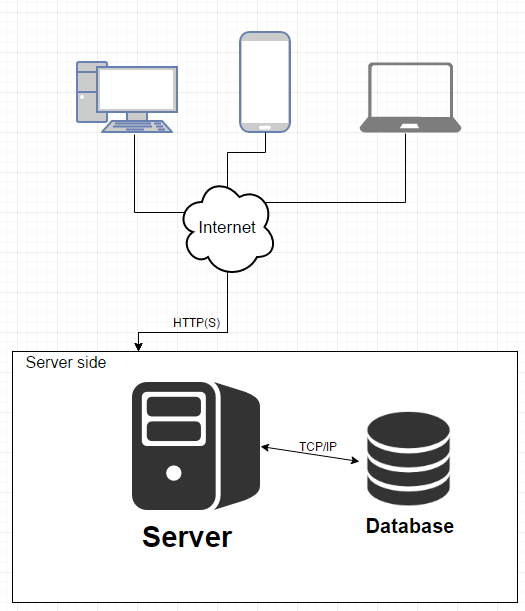


Рис. 3.1 Приклад клієнт серверної архітектури

Северна та клієнтська (фреймворк AngularJS) частини спроектовані за шаблоном MVC (Model-View-Controller).

Model-View-Controller (MVC) шаблон відокремлює [модель даних](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), вигляд ([інтерфейс 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)) та модуль керування, застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Model - представляє дані та реагує на команди контролера, змінюючи свій стан.

View - керує відображенням інформації.

Controller - інтерпретує дії користувача, повідомляючи модель про необхідність змінитись.

Контролер Web Api не працює з відображеннями, замість цього він приймає запити від клієнта та відсилає відповідь через HTTP протокол.

Клієнтська частина буде побудова за SPA(Single Page Application) підходом, тобто весь веб сайт буде поміщено на одній сторінці.

## 3.2. Проектування компонент

При проектуванні серверної частини продукту було вирішено створити три проекти у Visual Studio які взаємодіють між собою:

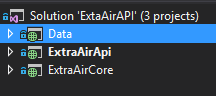


Рис. 3.2 Список пакетів на серверній частині

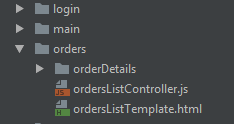
Data – у цьому проекті буде зосереджено роботу на маніпулювання з базою даних, усі запити до бази даних будуть відбуватись у цьому проекті.

ExtraAirCore – цей проект містить усі моделі та інтерфейси за якими відбувається маніпуляції з базою даних.

ExtraAirApi – це основний проект який об’єднює в собі два попередні та обробляє запити від клієнта, містить в собі всі додаткові збірки, бібліотеки та утиліти.

Цей підхід дозволяє зберегти код більш чистішим та зрозумілішим при зростанню основного проекта, також застосовується IoC-контейнер.

На клієнтській частині було вирішено кожен новий модуль створювати як новий пакет який містить: Controller, View та допоміжні файли.

  
Рис. 3.3. Вигляд пакетів на клієнтській частині

## 3.3. Проектування класів



Рис. 3.4. Діаграма класів

В результаті аналізу предметної області, визначено наступні класи:

User – клас в якому міститься інформація основна інформація користувача: імя, прізвище, email, пфроль, дата народження, видалений, адреса та роль.



Рис. 3.5. Клас User

Client – клас який наслідується від User та містить: список замовлень, список відгуків, спосок кредитних карт, спосок історій пошуків.



Рис. 3.6. Клас Client

Address – клас в якому міститься інформація про адресу(користувача та аеропорта) та мітить такі поля: країна, місто, вулиця, номер будинку, поштовий індекс



Рис. 3.7. Клас Address

Airport – клас який містить інформацію про аеропорт, а саме: імя, адресу та посилання на проміжний клас tourToAiprort(зв’язок багато до багатьох).



Рис. 3.8. Клас Airport

TourToAirport – проміжний клас який об’єднює Tour та Airpor та містить: час віправлення, час прибуття, чи це проміжний рейс, та посилання на рейс та аеропорт.



Рис. 3.9. Клас TourToAirport

Tour – клас який містить інформація про рейс, а саме: час віправлення, час прибуття, ціна, літак, поточна кількість пасажирів, список пасажирів, список замовлень та посилання на tourToAirport



Рис. 3.10. клас Tour

Order – клас який містить інформацію про замовлення, а саме: дату замовлення, користувача, список рейсів, список пасажирів, ціна, чи замовлення оплачено, дата відправлення, дата прибуття.



Рис. 3.11. Клас Order

Plane – клас який несе інформацію про літак, назва, максимальна кількість пасажирів, рівні кофморту



Рис. 3.12. Клас Plane

TourDetails – клас який несе детальну інформацію про конкретний рейс для замовлення, та містить такі поля: дату початку та завершення рейсу, поточну кількість пасажирів на рейсі, назва, максимальна кількість пасажирів, рівні кофморту, посилання на рейс та список заброньованих місць.

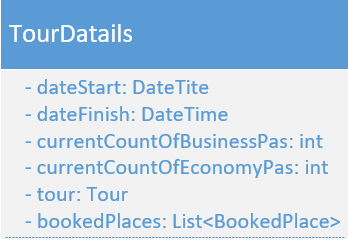


Рис. 3.13. Клас TourDetails

BookedPlace – клас який несе інформацію заброньоване місце та містить координати місця, та тип комфорту(клас рейсу).

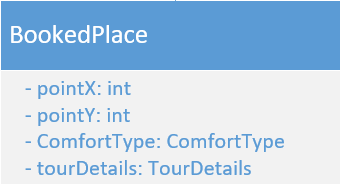


Рис. 3.14. Клас BookedPlace

## 3.4. Проектування бази даних

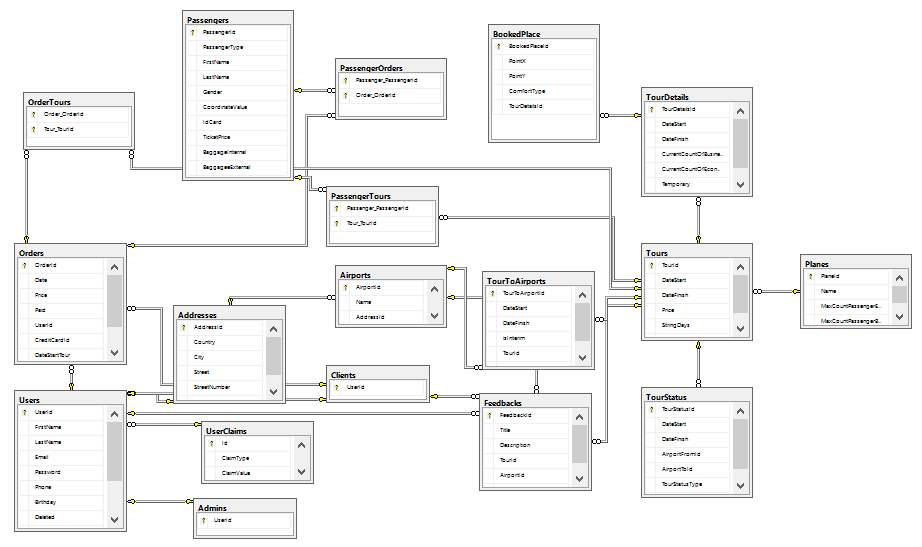


Рис. 3.15. Схема бази даних

В результаті проеткування архітектури бази даних було вирішено використовувати підхід Code First, тобто спочатку спроектувати діаграму класі та на її основі за створити базу даних, тому як можна побачити зі схеми бази даних(рис. 3.15.), вона повмісню ідентична діаграмі класів окрім декількох табличок. Такі сутності як PassengerTours та PassengerOrders, були додані для реалізації звязку багато до багатьох, тому що в базі даних для цього потрібно додати додаткову табличку.

## 3.5. Прототипування інтерфейсу користувача

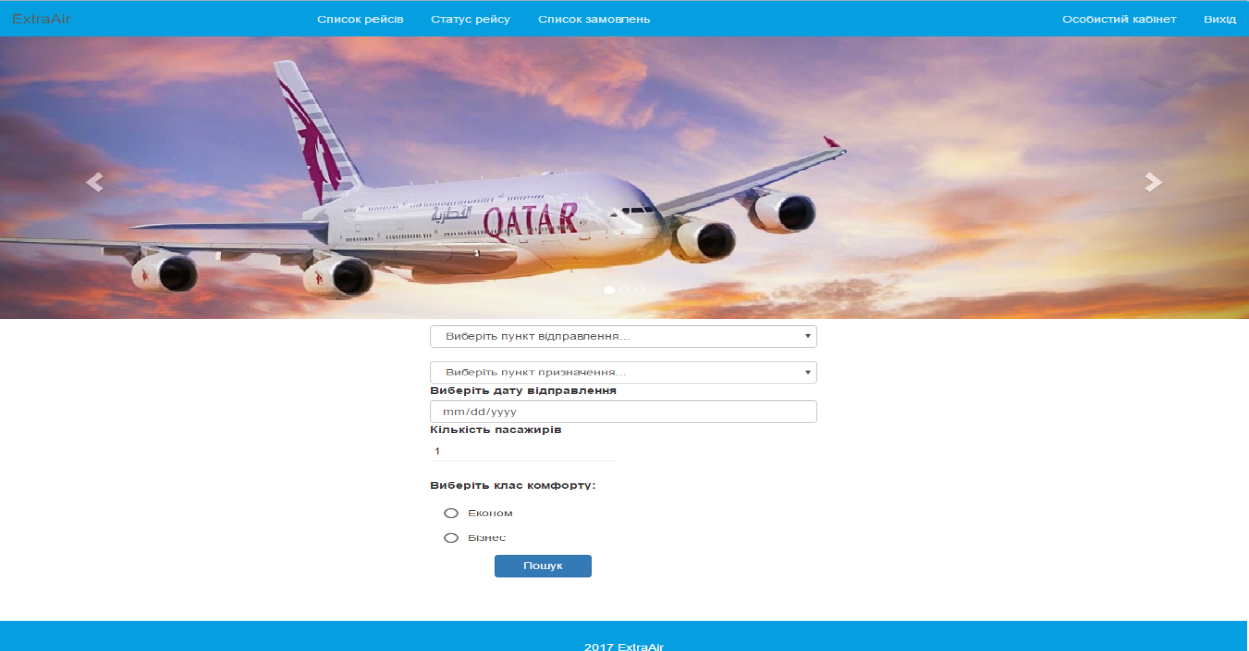


Рис. 3.16. Домашня сторінка додатку

**Опис:** при першому запуску програми відображається головне вікно, на якому користувач може здійснити пошук рейсів, а також перейти на інші вкладки програми.

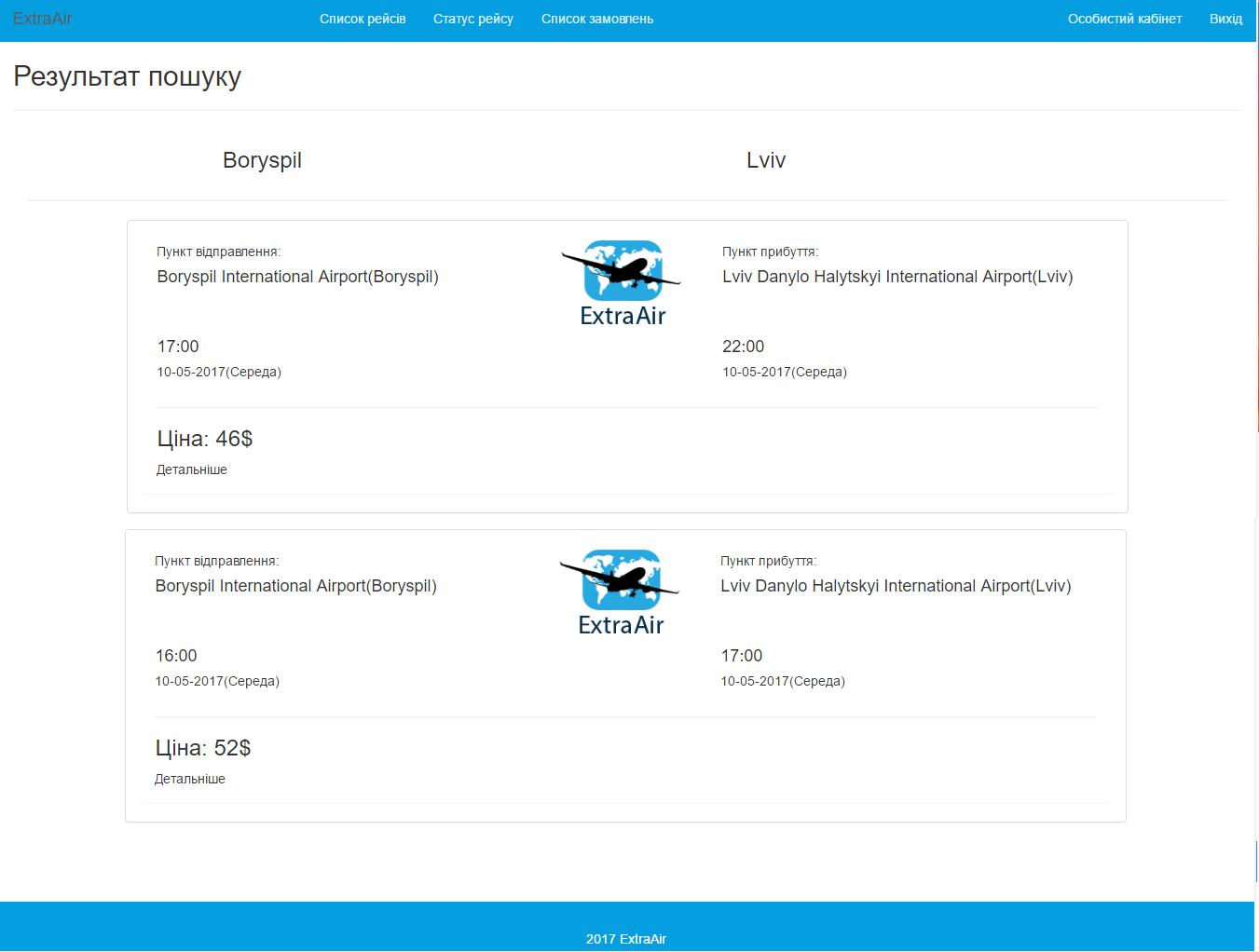
****

Рис. 3.17. Сторінка результату пошуку рейсів

**Опис:** на цій сторінці можна вибрати рейс для детальної інформації та подальшого замовлення.

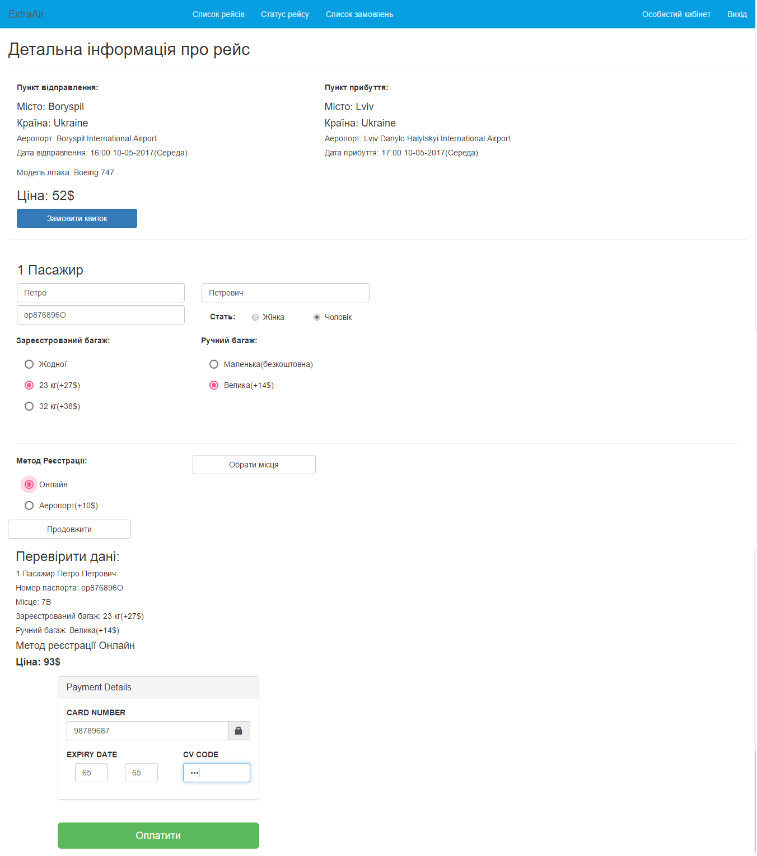


Рис. 3.18 Сторінка замовлення квитка

**Опис:** на сторінці замовлення квитка потрібно заповнити усі поля вводу та натискати відповідні клавіші.

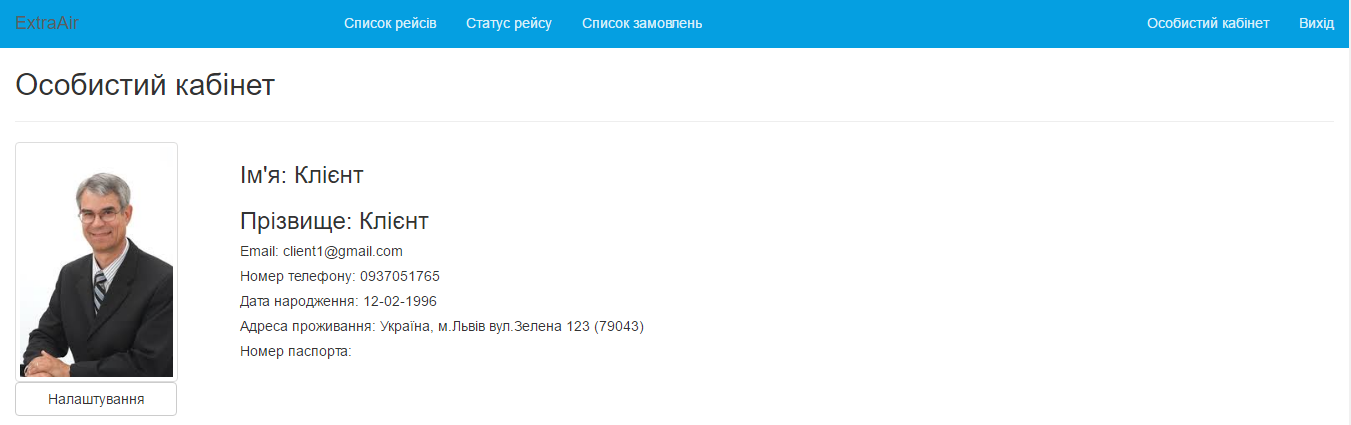


Рис. 3.19 Сторінка особистого кабінету.

**Опис:** на цій сторінці користувач може перегляну особисту інформацію, а також редагувати її.



Рис. 3.20 Модальне вікно редагування особистого кабінету.

**Опис:** на цьому вікні користувач може редагувати особисту інформацію.

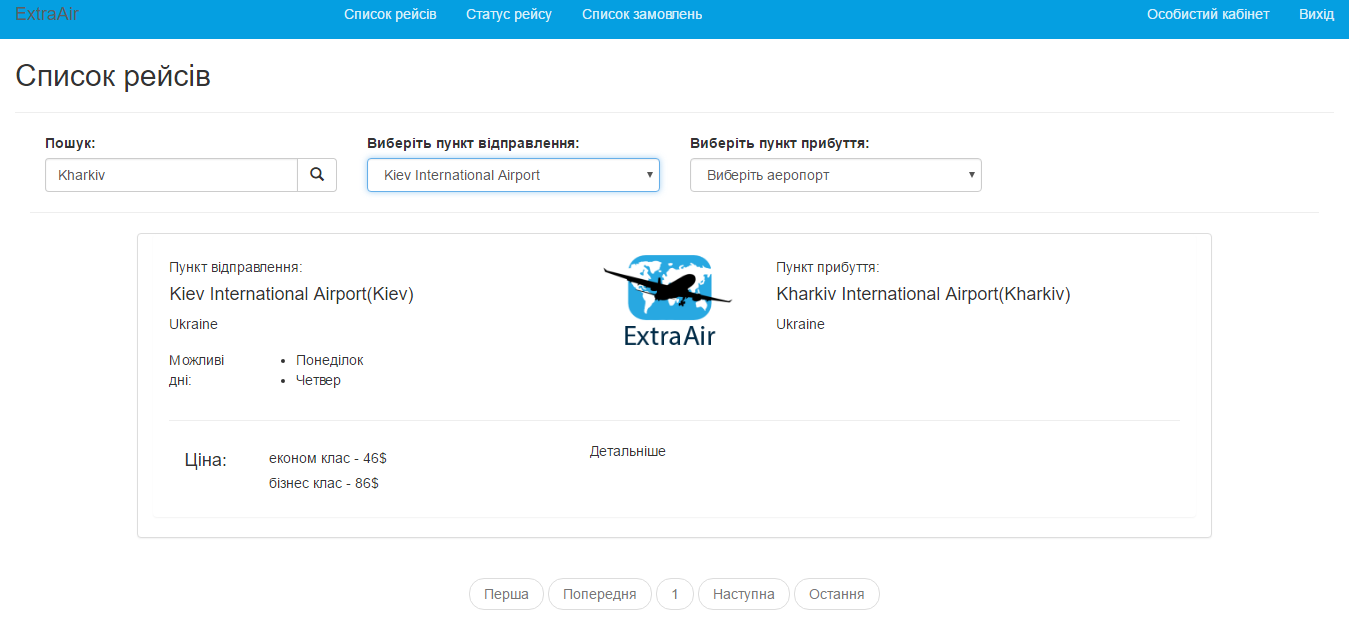


Рис. 3.21. Сторінка «Список рейсів»

**Опис:** на даній сторінці буде можливість переглянути повний список рейсів, а таож їх фільтрація та пошук.

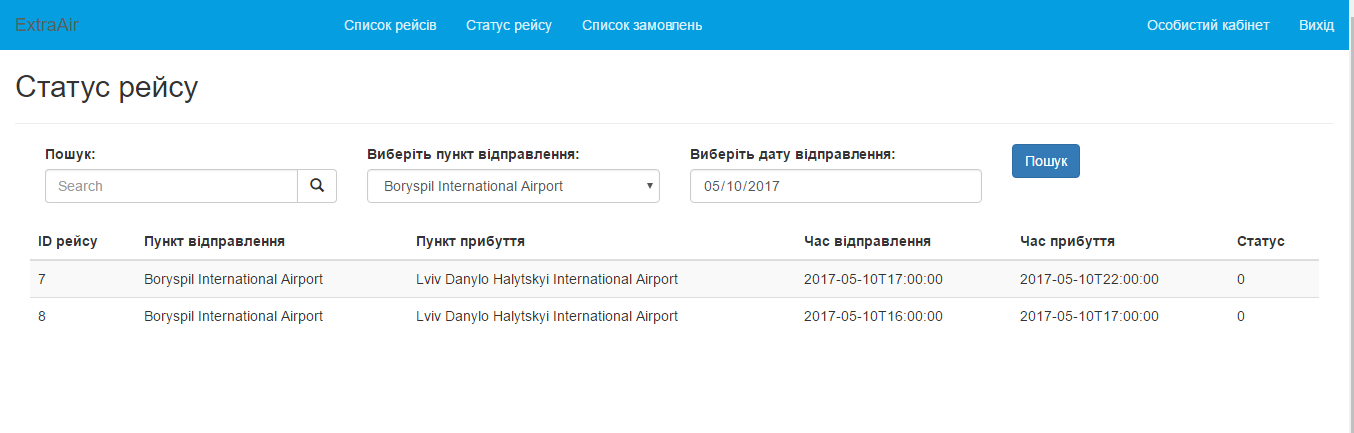


Рис. 3.22. Сторінка «Статус рейсу»

**Опис:** на цій сторінці користувач може відслідковувати старус рейсів

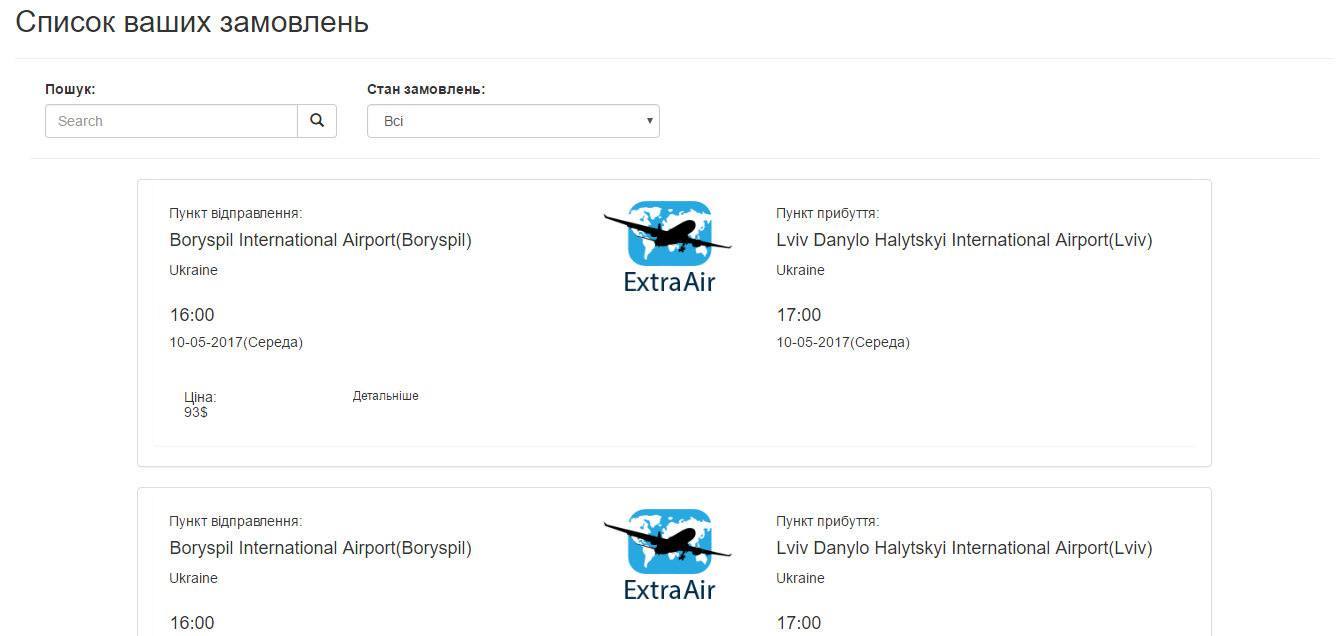


Рис. 3.23. Сторінка «Список ваших замовлень»

**Опис:** користувач буде мати змогу переглядати список усіх своїх замовлень.

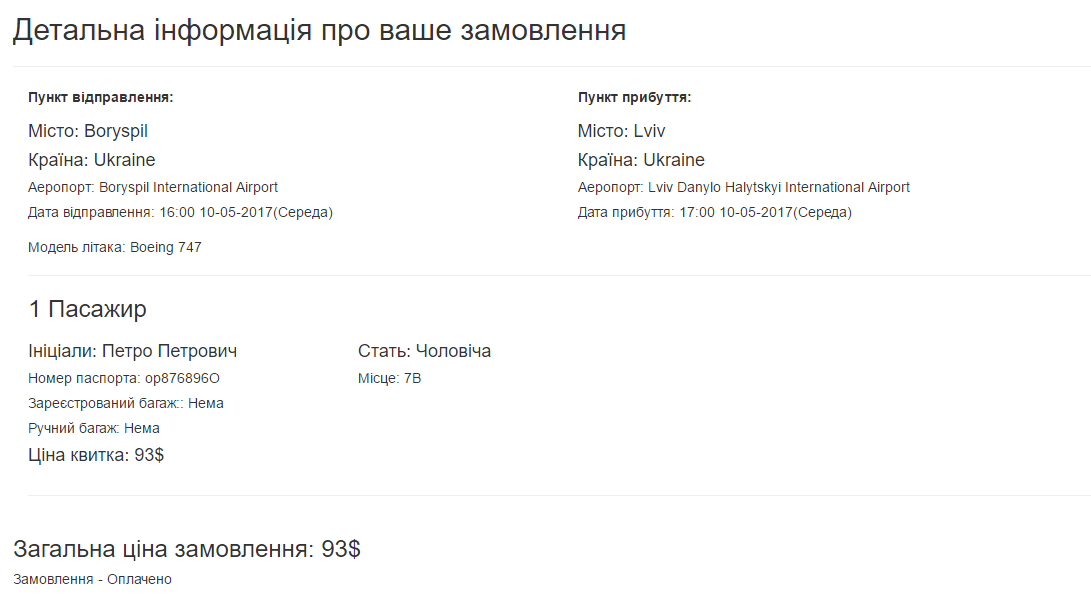


Рис. 3.24. Сторінка детальної інформації про конкретне замовлення

**Опис:** на сторінці користувач буде мати змогу переглянути детальну інформацію про своє замовлення.

## 3.6. Висновок

В результаті виконання 3 розділу бакалаврської кваліфікаційної роботи спроектована архітектура веб додатку. В якості загального принципу роботи продукту вибрано клієнт-серверну архітектуру, а також серверна та клієнтська частини будуть побудовані за шаблоном MVC. Цей підхід набагато спрощує побудову та розуміння проетку для подальшої експлуатації майбутніми розробниками. Також було описано принцип побудови компонент на серверній та клієнтській частинах. Здійснено проектування основних класів системи та бази даних.

Розроблено прототип інтерфейсу для веб системи, який виконаний у стилі мінімалізму та є інтуїтивно зрозумілим.

# Розділ 4. Реалізація та тестування Програмного продукту

## 4.1. Реалізація веб сервісу

Після визначення вимог та вибору архітектури для програмного продукту є реалізація поставлених цілей. Так як вибрана архітектура є клієнт-сервер то реалізацію можна поділити на клієнтську частину та серверну. Також для зберігання усіх змін у проекті було обрано систему контролю версій Git, та створено репозиторій на GitHub.

### Реалізація серверної частини

Згідно вимог, серверна частина буде використовувати .Net технології, а саме Web Api2, та ORM технологію Entity Framework для маніпуляцією з базою даних та MS SQL Server.

Перш за все було створено моделі класів які відповідають діаграмі класів та використовуються для формування бази даних за підходом Code First(рис 4.1).



Рис. 4.1. Відображення моделів класів

Після цього було зроблено міграцію бази даних за допомогою класу ExtraAirContext, об’єкт якого, за допомогою Linq-методів, здійснює маніпуляцією з базою даних.

public class ExtraAirContext : DbContext

{

public ExtraAirContext(): base (ConfigurationManager.ConnectionStrings["ExtraAirContext"].ConnectionString) { }

public virtual DbSet<Address> Addresses { get; set; }

public virtual DbSet<Airport> Airports { get; set; }

public virtual DbSet<Comfort> Comforts { get; set; }

public virtual DbSet<CreditCard> CreditCards { get; set; }

public virtual DbSet<Feedback> Feedbacks { get; set; }

public virtual DbSet<Order> Orders { get; set; }

public virtual DbSet<Passenger> Passengers { get; set; }

public virtual DbSet<Plane> Planes { get; set; }

public virtual DbSet<Tour> Tours { get; set; }

public virtual DbSet<User> Users { get; set; }

public virtual DbSet<Client> Clients { get; set; }

public virtual DbSet<Dispatcher> Dispatchers { get; set; }

public virtual DbSet<Admin> Admins { get; set; }

public virtual DbSet<UserClaim> UserClaims { get; set; }

public virtual DbSet<TourToAirport> TourToAirports { get; set; }

public virtual DbSet<TourSearchHistory> TourSearchHistories { get; set; }

public virtual DbSet<TourDetails> TourDetailses { get; set; }

public virtual DbSet<BookedPlace> BookedPlaces { get; set; }

public virtual DbSet<TourStatus> TourStatuses { get; set; }

}

Для того щоб ідентифікувати користувача та надати йому права використовується авторизація та аутентифікація. У даному проекті використовується Token base авторизація, а саме jwt-token, який формується на сервері, та наступні рази надсилається у кожному запиті.

Наступним короком є написання контролерів. Контролер являє собою клас який спілкується з клієнтом по HTTP протоколу та обробляє запити(рис 4.2.).

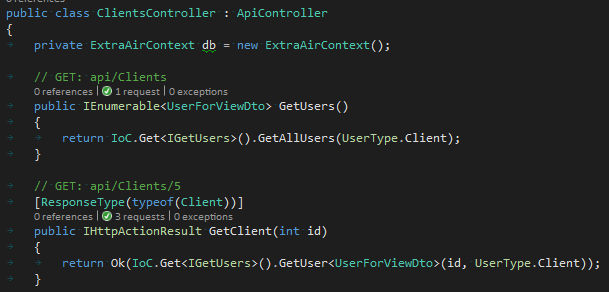


Рис. 4.2. Приклад контролера в Web Api

### 4.1.2. Реалізація клієнтської частини

Як було описаноу вимогах, клієнтська частина буде побудована по SPA(Single Page Application) принципу, використовувати стандартні веб технології(HTML5, CSS3, JavaScript), а також фреймворк AngularJS, який побудований за шаблоном MVC. Отже перш за все було створено модель аплікації, потім файл app.js – це основний файл проекту в якому прописана вся маршрутизація та основні функції проекту, та створено основний файл index.html. Після цього реалізовано логіку окремих компонент в проекті, велику роль в цього відіграє контролер в якому зосереджено основну логіку роботи конкретного модуля та передачу даних на сервер чи відображення в браузері. Після цього створено сервіси та ресурси для доступу до серверної частини.

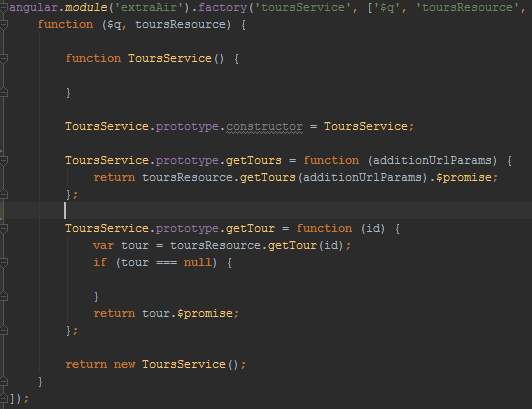


Рис. 4.3. Приклад сервіса в AngularJS

## 4.2. Опис роботи з програмним продуктом

Спочатку користувачу необхідно перейте на веб сторінку. Після цього з’являється головна сторінка(рис. 4.4.) сайту на якій зверху знаходиться навігаційне меню, під ним слайдер з декількома картинками, а під цим - поля для пошуку рейсу(рис. 4.5.).

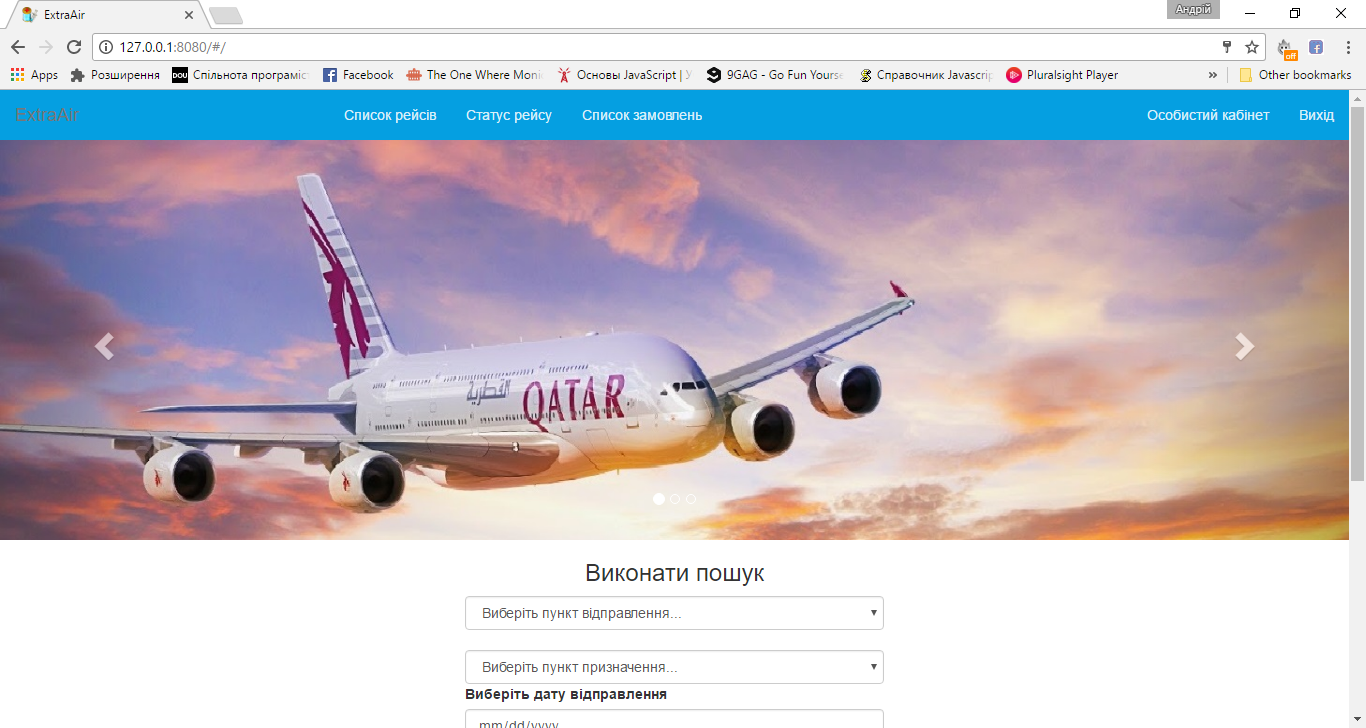


Рис. 4.4.



Рис. 4.5.

Для того щоб отримати всі повноцінні права, користувачу потрібно створити свій аккаунт, та увійти в систему, для цього користувач вибирає посилання «Реєстрація» зправа у навігаційній панелі, перед ним з’являється сторінка реєстрації, на якій користувачеві необхідно заповнити усі поля та натиснути на кнопку «Зареєструватись»(рис. 4.6.). Після цього користувач опиняється на сторінці «Особистий кабінет»(рис. 4.7.) де він може переглянути особисту інформацію, а також змініти її(включаючи нове фото) або поміняти пароль(рис 4.8).

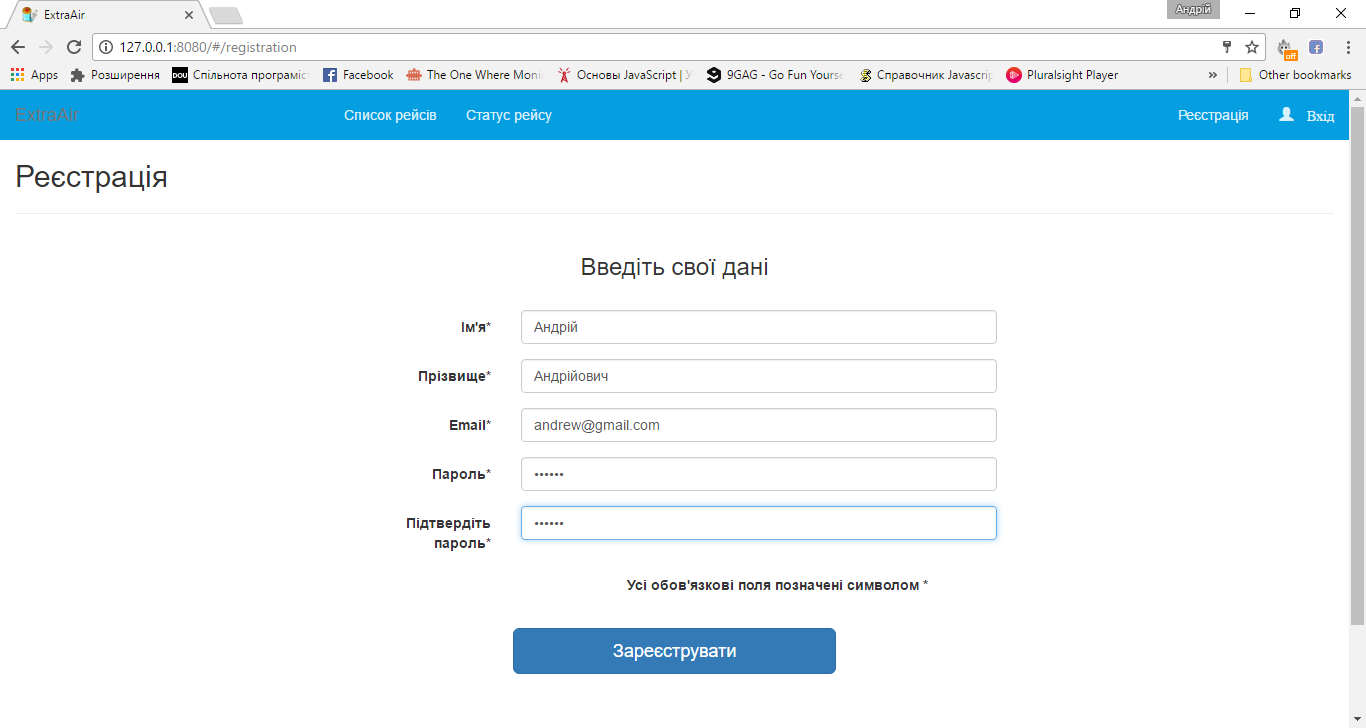


Рис. 4.6.

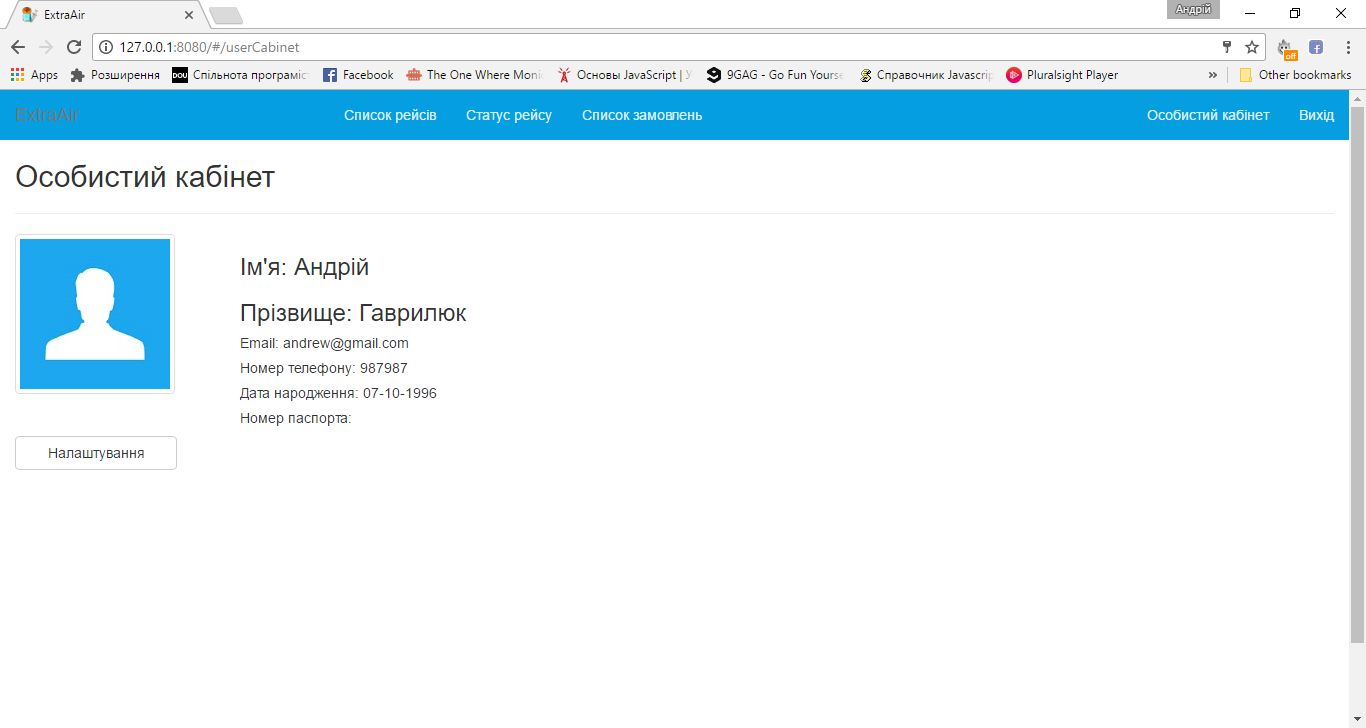


Рис. 4.7.

Отже щоб замовити квиток, користувач повинен увійти в систему перейти на головну сторніку, заповнити усі поля вводу які потрібні для пошуку рейсів, після цього користувач побачить нову сторінку зі списком усіх можливих рейсів за заданими параметрами(рис. 4.9.).

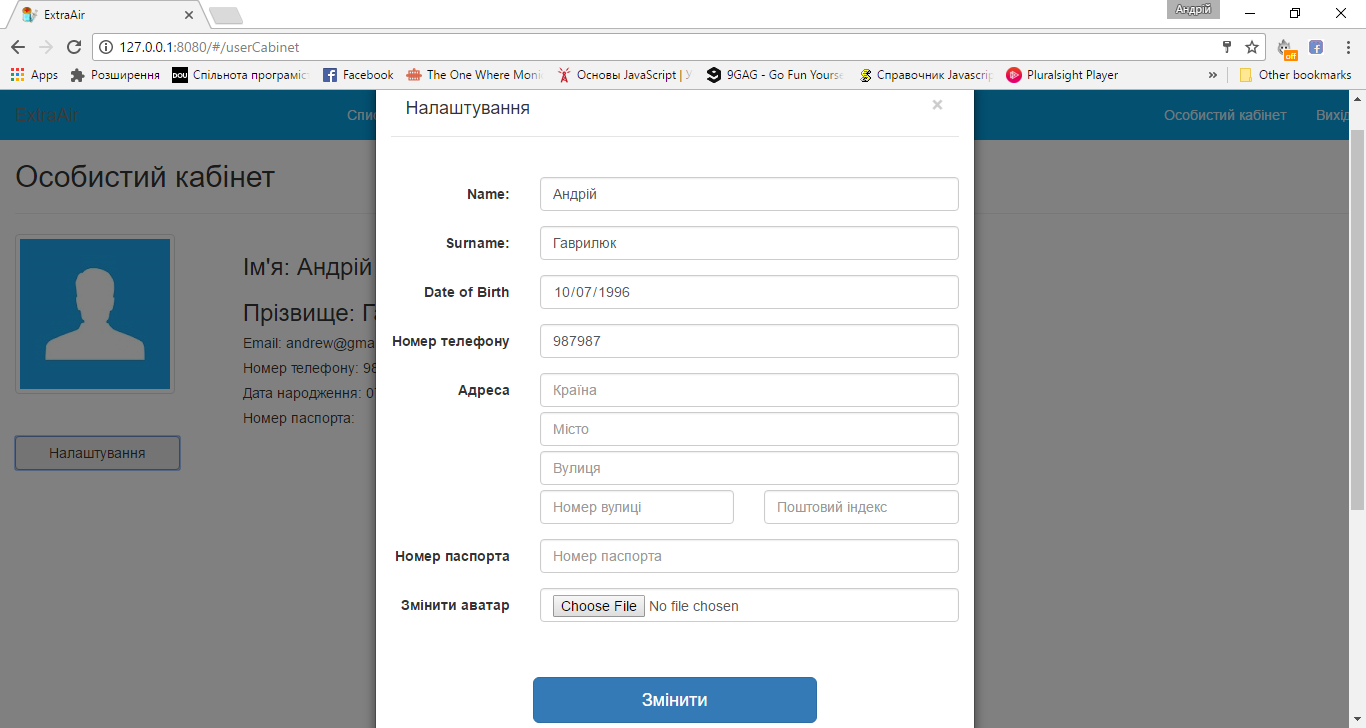


Рис. 4.8.

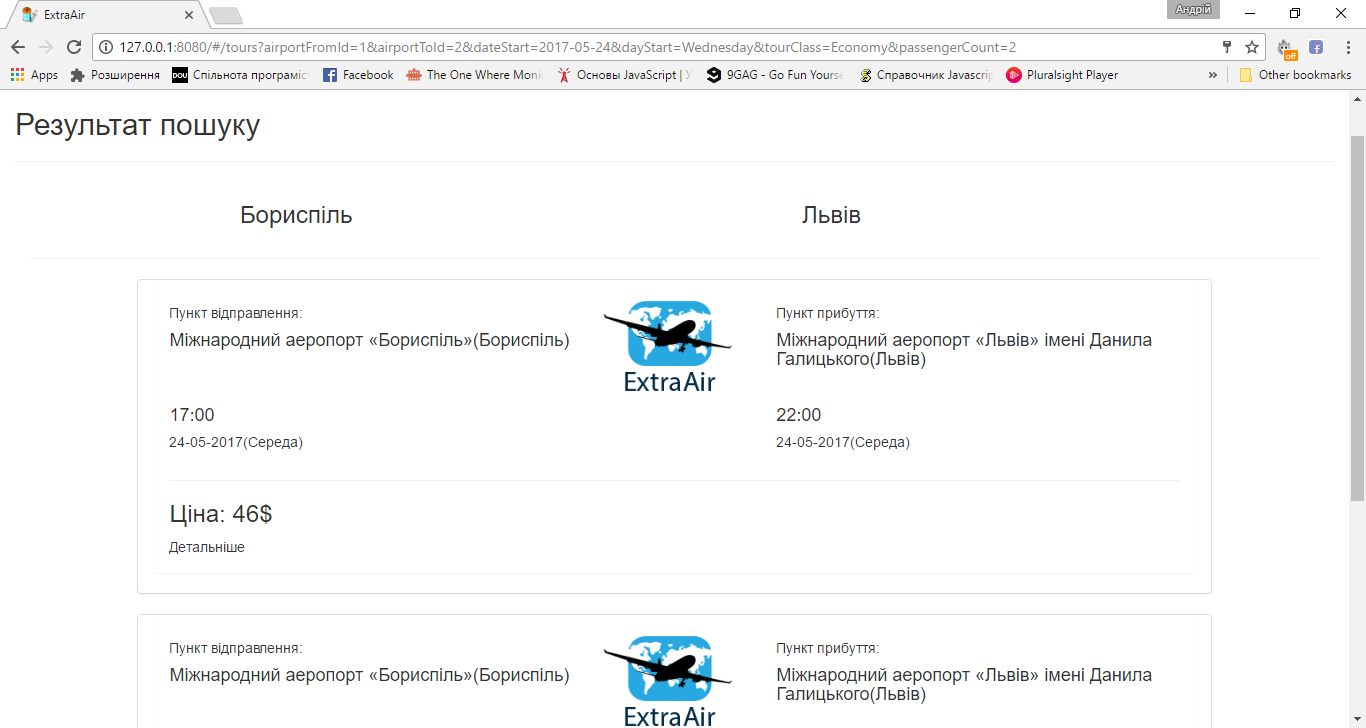


Рис. 4.9.

Вибравши якийсь конкретний рейс та натиснувши на посилання «Детальніше», перед користувачем з’являється сторінка із детальною інформацією про цей рейс та кнопкою «Здійснити замовлення»(рис. 4.10.). Якщо користувачеві підходить цей рейс він нажимає на кнопку «Здійснити замовлення» та перед ним з’являються нові поля(рис. 5.11.) які потрібно заповнити: інформація про пасажирів(ім’я, прізвище, номер паспорту, стать) в залежності від попедньо вибраної кількості, якщо вибрано економ клас, то можливість вибору додаткових послуг(ручний чи зареєстрований багаж), вибір місць(рис. 5.12.) та метод реєстрації на рейс або онлайн або аеропорт.

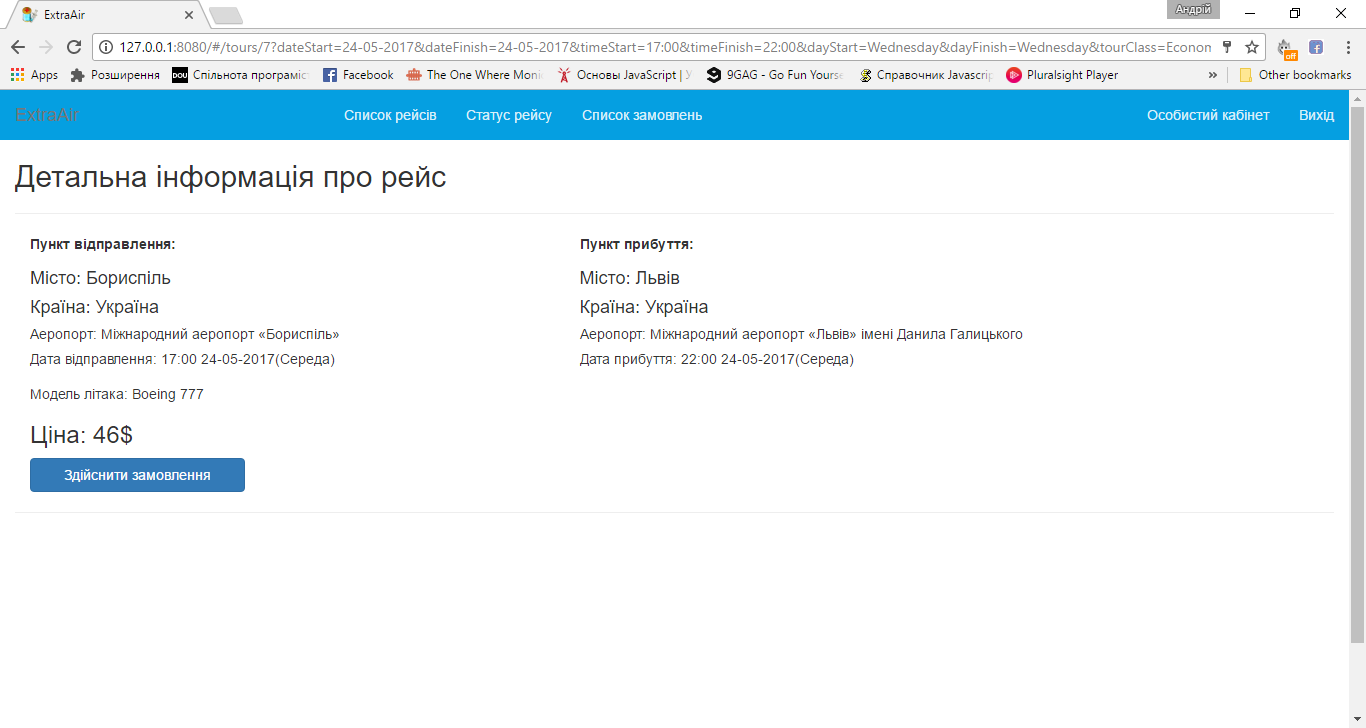


Рис. 4.10.

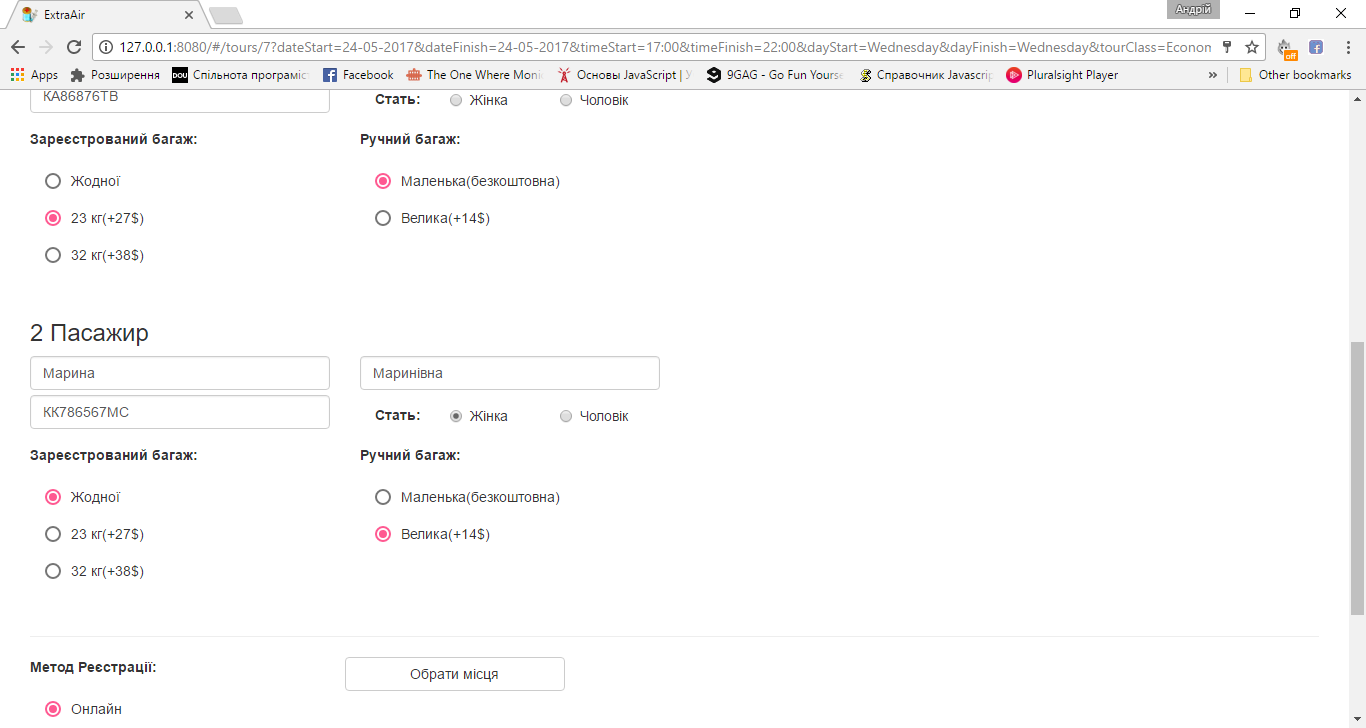


Рис. 4.11.

Після цього з’являється додаткова перевірка інформації та якщо метод реєстрації онлайн з’являється форма для заповнення кредитної картки(рис. 5.13.), якщо метод реєстрації в аеропорті то кнопка «Забронювати». Натиснувши на кнопку «Оплати»/ «Забронювати» з’ятвиться повідомлення про те що операцію здійснено успішно(або не успішно) і ваше замовлення з’явиться у списку замовлень яке можна переглянути на сторнці детальної інформації про замовлення.

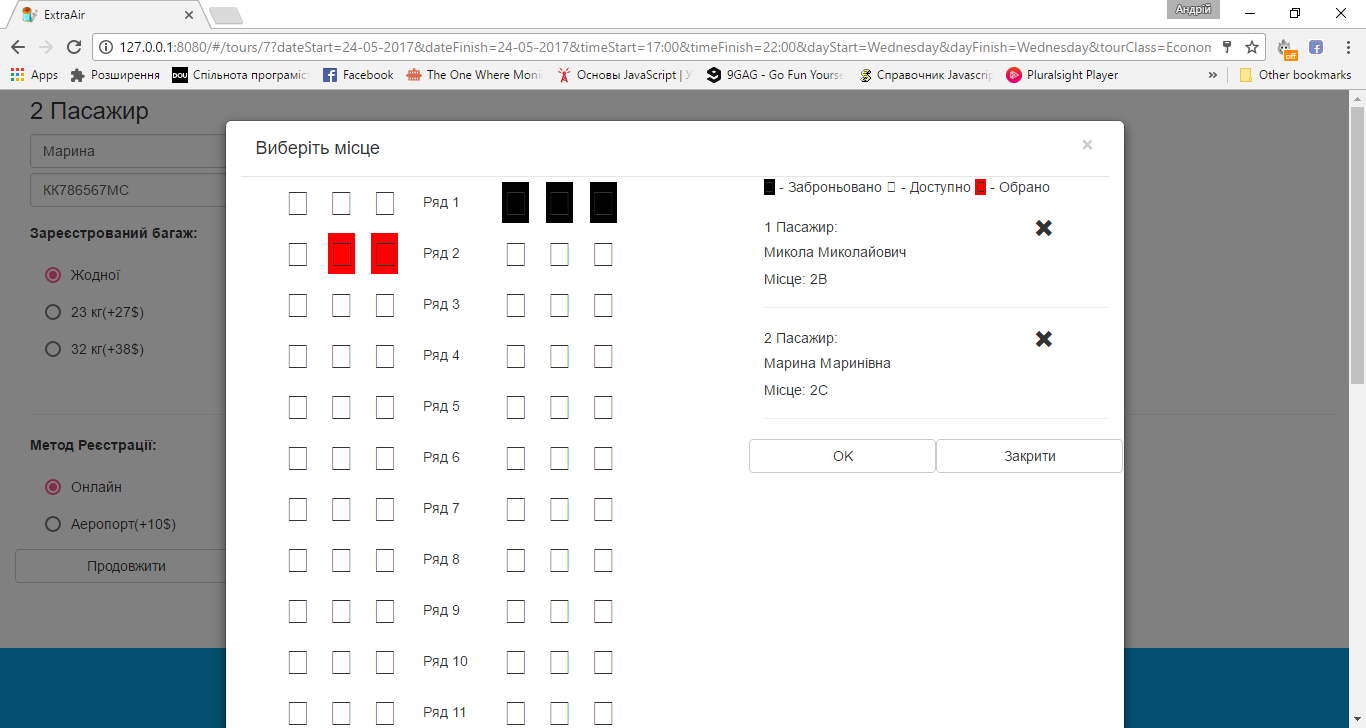


Рис. 4.12.

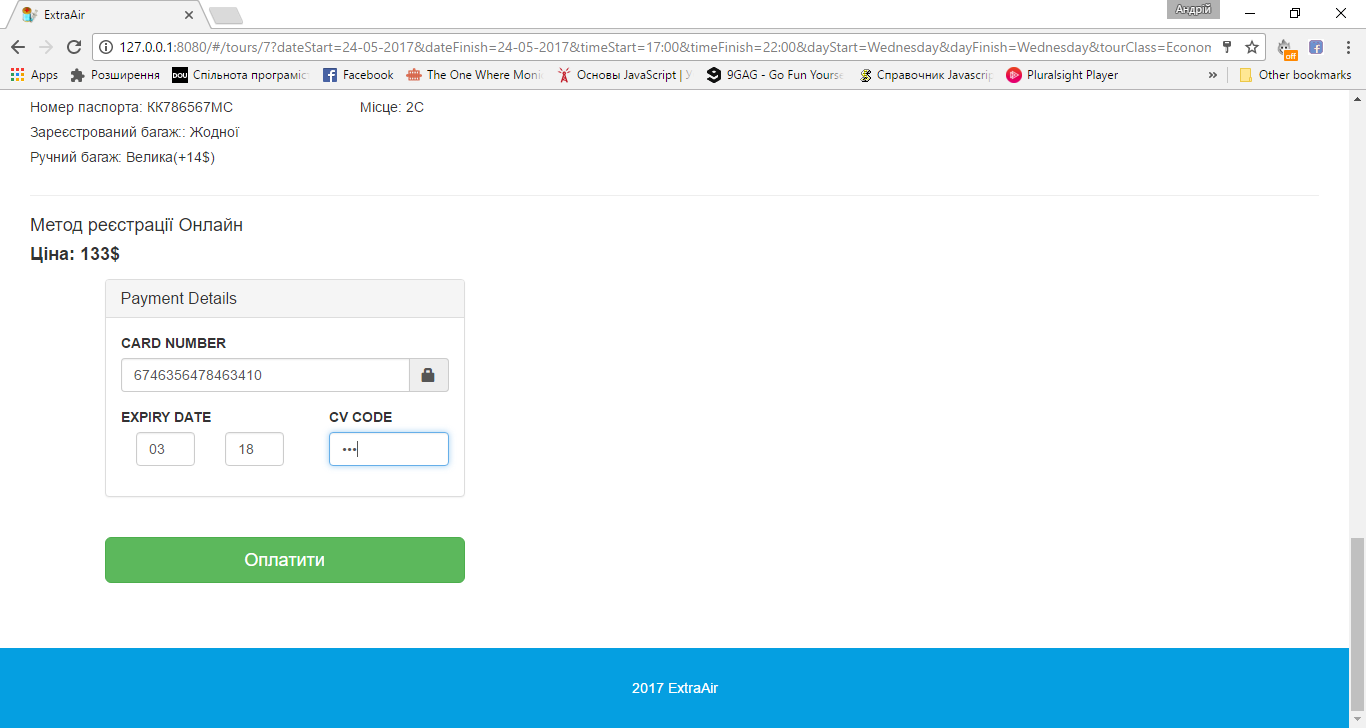


Рис. 4.13.

Список рейсів перейшовши на цю сторінку, натиснувши на відповідне посилання у панелі навігації з’являється список усіх можливих рейсів(рис. 5.14.). У цьому списку присутня pagination, щоб обмежити кількість ітерацій на сторінці, а також можна застосувати фільтри та пошук(рис. 5.15.).

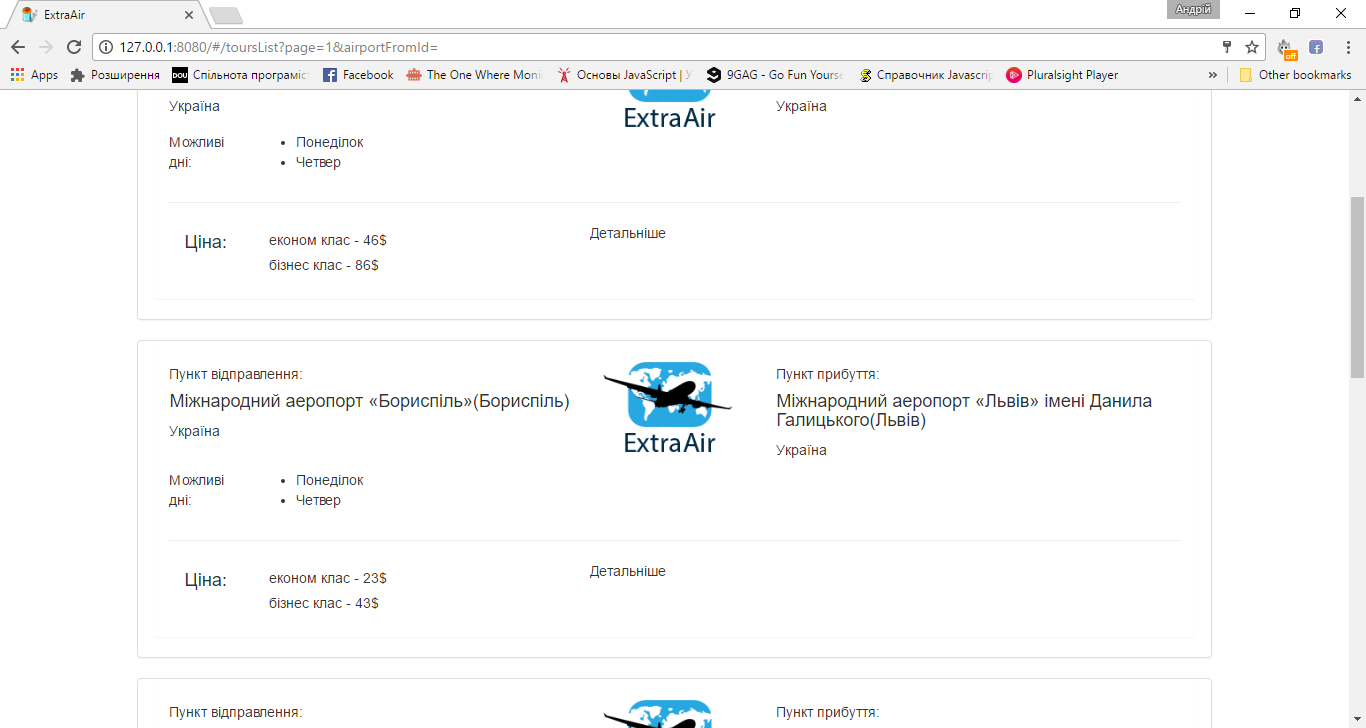


Рис. 4.14.

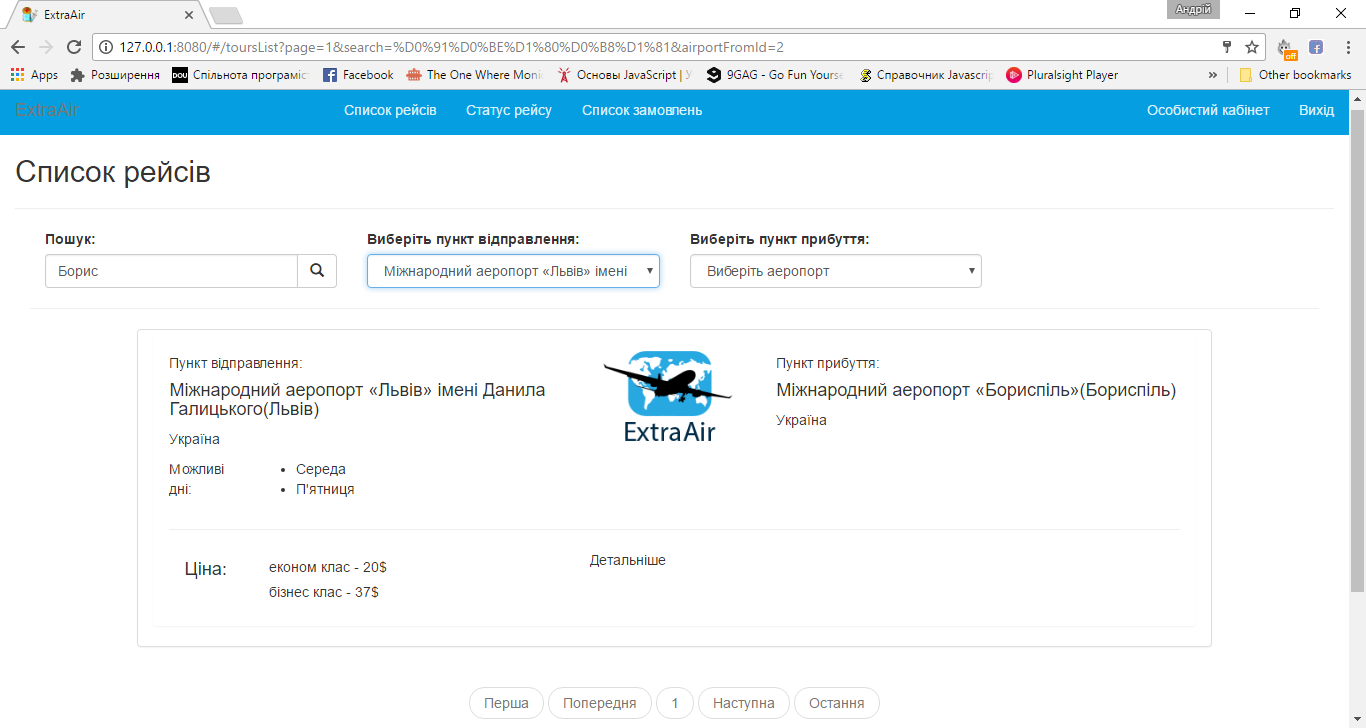


Рис. 4.15.

Можна замітити що після входу в систему зв’явилось нове посилання в верхній навігаційній панелі «Список замовлень»(рис. 5.16.), перейшовши на цю сторінку користувач може переглянути список усіх замовлень які він коли небуть робив. На цій сторінці теж присутня pagination, фільтри та пошук. Натиснувши на посилання «Детальніше» користувач переходить на сторінку з детальною інформацією про вибране замовлення.

На сторінці «Статус рейсу» користувач може переглянути статус рейсу за заданити параметрами.

Окремо потрібно розказати про роль адміністратора, а саме відрізнятиметься сторінка особистого кабінету(рис. 5.17.). Якщо користучав зайшов в систему з правами адміністратора, він має право переглядати та редагувати інформацію про аеропорти, рейси, статус рейсу, користувачів.



Рис. 4.16.

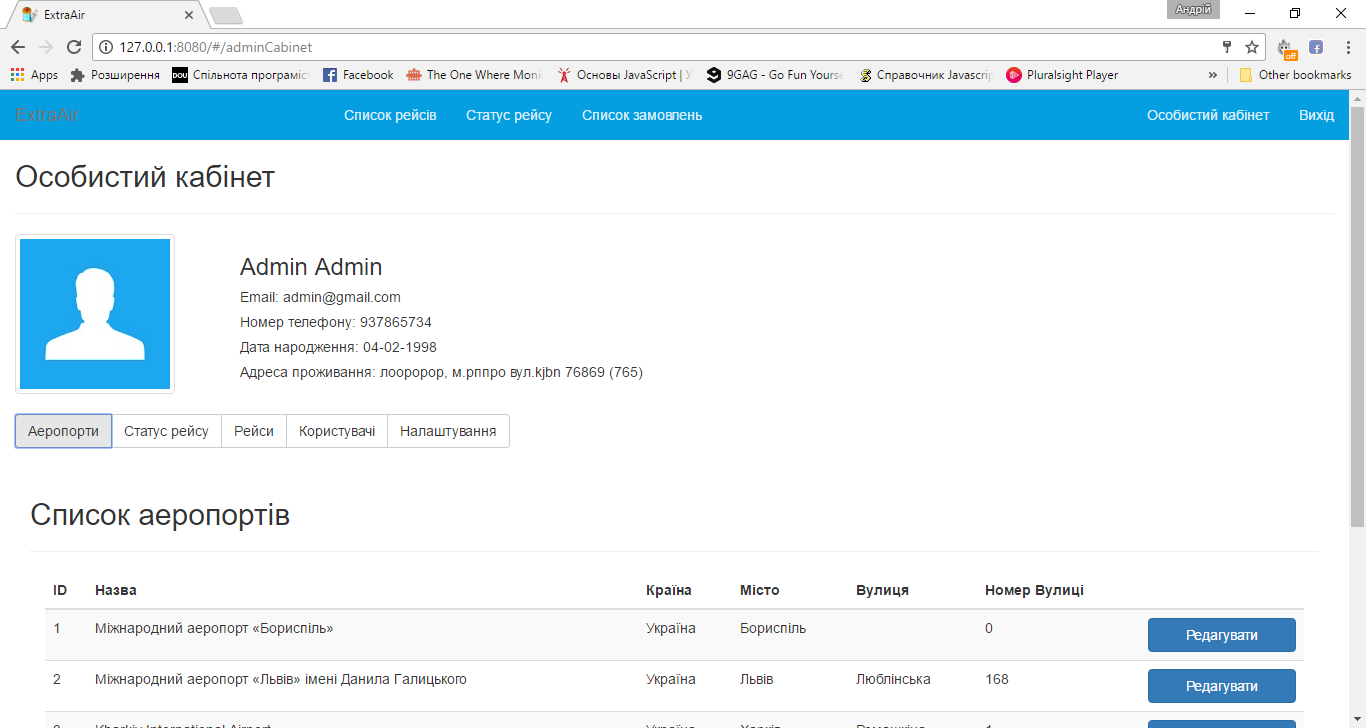


Рис. 4.17.

Для того щоб відредагувати інформацію про аеропорт потрібно вибрати вкладку «Аеропорти», вибрати якийсь конкретний аеропорт, натиснути кнопку «Редагувати», та у відповідні поля ввести нову інформацію. Аналогічно для рейсів та користувачів. Тільки для статусу рейсу потрібно спочатку знайти потрібний рейс.

## 4.3. Тестування програмного продукту

**4.3.1. Вступ.**

В цьому розділі наведені результати тестування мобільного додатку. Також показано оцінку якості. Було здійснено функціональне тестування готового мобільного додатку.

**4.3.2. Розробка тестів.**

Під час фази тестування було розроблено тести функціонального тестування, а також проведено статистичний аналіз коду за допомогою Android Studio.

**4.3.3. Функціональне тестування**

4.3.3.1. Результати тестування

Під час функціонального тестування провірялись усі реалізовані функціональні можливості додатку.

Звіт по функціональному тестуванню наведено в додатку Д.

4.3.3.2. Підсумки тестування

Мобільний додаток розроблявся ітераційно, тобто від замовника постійно надходили нові побажання та зауваження. Тому процес тестування та відлагодження продовжувався протягом усього проекту. Часто були були зауваження, пов’язані інтерфейсом програми, наприклад, «Підправити відображення прапорів країн», поміняти графічне відображення спортивних подій в списку. Також були проблеми із пам’яттю, програма аварійно завершувалась після тривалого користування програмою. Тому мені прийшлось оптимізувати програмний код та алгоритми, для економії ресурсів. Також деякі дані сервер генерував некоректно. Наприклад, JSON даних про бейсбол мав іншу структуру. Це стало причиною того, що програма аварійно завершувалась після вибору цього виду спорту. Для вирішення цієї проблеми я модифікував алгоритм парсингу спеціально для цих даних, після чого проблема вирішилась. Дані завантажувались не в тому порядку, в якому потрібно. Для цього прийшлось розробляти алгоритми для їх впорядкування.

Крім того, замовник завжди вносив свої поправки. Добавлялись нові види спорту, які треба було інтегрувати в додаток. Також появлялись нові букмекерські контори. В процесі, змінювались посилання на них, логотип, назви. Це все виправлялось. В JSON появлялись нові поля, наприклад, посилання на «Watch Live» для букмекерів. Які я добавляв в додаток. Всі помилки були виправлені і всі зауваження замовника були задоволені. Більш детальна інформація про список багів та зауважень від замовника знаходиться в додатку З.

Перед релізом першої версії. Я здійснив тестування функціоналу системи. В результаті програма вже працювала стабільніше, ніж раніше, але все-таки появились декілька багів. Не завжди коректно завантажувався список подій, хоча мав би. Це було причиною поганої обробки виняткових ситуацій. Коефіцієнти в списку букмекерів некоректно відображались, якщо кількість цифр в числі була надто великою. Іноді список матчів в додатку не відповідав списку матчів на сайті. Також було декілька дефектів в інтерфейсі додатку. В результаті, я виправив всі помилки, провів повторне тестування, яке не виявило цих помилок. Згенерував підписаний apk файл, який згодом був опублікований на Google Play. Більш детальна інформація про список багів та зауважень від замовника знаходиться в додатку К.

Після редезайну та розробки нової версії продукту я провів функціональне тестування, результати якого наведені в таблиці 4.1.

Таблиця 4.1

Розподіл варіантів використання для функціонального тестування

| **Варіанти використання** | **Тестові дані** |
| --- | --- |
| Вибір виду спорту | 28 |
| Додавання команд (учасників) у «вибране» | 10 |
| Додавання ліг у «вибране» | 5 |
| Фільтрування «активних» подій | 3 |
| Фільтрування подій з прогнозами | 3 |
| Користувацьке оновлення даних | 5 |
| Фільтрування «вибраних» подій в Explore | 4 |
| Вибір дати для завантаження за допомогою календаря | 5 |
| Пролистування подій за датою | 4 |
| Пошук подій за пошуковою фразою | 10 |
| Перевірка деталей події на відповідність із сайтом | 10 |
| Перехід на сайт букмекера | 4 |
| Користувацьке оновлення деталей події | 3 |
| Поділитись подією з друзями в соціальних мережах | 4 |
| Отримання даних турнірної таблиці та перевірка її на відповідність із сайтом | 5 |
| Отримання даних «вибраних подій» | 10 |
| Перевірка роботи сповіщень | 3 |
| Зміна інтервалу оновлень | 2 |
| Очищення даних користувача | 3 |
| **Загалом** | **121** |

Всі тестові випадки пройшли успішно, тому функціональне тестування вважається успішним.

**4.3.4. Статичний аналіз коду в Android Studio**

Для визначення та усунення потенційно небезпечних ділянок в коді програми був проведений статистичний аналіз коду, результати якого наведені в Додатку З. Він допоміг знайти такі критичні місця програми як непроініціалізовані змінні, що можуть бути причиною небезпечних помилок, чи непотрібні ресурси, які погіршують продуктивність програми за рахунок надмірного використання оперативної пам’яті. Також аналіз виявив хардкод в коді програми, а також посилання, які можуть спричинити переповнення пам’яті. Після усунення цих критичних місць в програмі, якість коду та ресурсів значно покращилася.

**4.3.5. Критерій успіх/провал проекту.** Тут наведені умови, виконання яких свідчить про успішність проведення тестування та якість продукту:

Розробка тестів:

* Розроблені тести для кожного варіанту використання системи;
* Тести покривають усі варіанти використання;
* Тести покривають усі програмні вимоги.

Виконання тестів:

* Всі розроблені тести виконано;
* Програма успішно пройшла всі тести;
* Здійснено статистичний аналіз проекту засобами Android Studio.

Наведені умови задоволено, проект вважається успішним.

## 4.4. Висновок

В результаті реалізації та тестування програмного продукту розроблений мобільний додаток для ОС Android. Реалізовано всі класи та алгоритми необхідні для роботи всіх функцій програми. Реалізовано алгоритми завантаження та парсингу даних з мережі, алгоритм опрацювання даних спортивних подій. Реалізовано графічний інтерфейс. Виконане тестування програмного продукту, яке відбувалось на протязі всього процесу розробки. При цьому враховувались побажання та зауваження замовника. Також здійснено функціональне тестування, під час якого виконано перевірку всіх варіантів використання продукту. Для перевірки правильності написання коду був здійснений статичний аналіз коду за допомогою Androi Studio. Всі баги виправлено.

# РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА

## 5.1. Економічна характеристика програмного забезпечення для моніторингу та планування пасажирських авіаперевезень

Метою даної бакалаврської кваліфікаційної дипломної роботи є розроблення веб орієнтованого сервісу для моніторингу та планування пасажирських авіаперевезень з метою полегшення планування та здійснення подорожей. Ціль даного сервісу – полегшити процес пошуку авіаквитків та планування подорожей. Програма повинна видавати результати пошуку авіа білетів по зацікавлених напрямках, а також дає можливість відслідковувати стан певного авіарейсу.

Дана програма призначена як для людей, які планують власну подорож за допомогою авіаперевезень, так і для людей, яким необхідно знати статус певних рейсів для можливих зустрічей рейсів, планування поїздок тощо. Додаток пропонує велику кількість міських з’єднань через авіа шляхи за різними категоріями із відповідними статусами польотів та усією необхідною інформацією щодо рейсу.

Розробка даного веб сервісу є доцільною, з економічної точки зору. Є програми аналоги, які мають можливість пошуку та підбору авіаквитків інколи навіть з можливістю онлайн покупки та майбутнього відправлення квитка на скриньку електронної пошти, проте цей процес є часто досить складним та незрозумілим для користувача, який користується таким сервісом вперше, що заставляє надавати перевагу авіакасам та туристичним агентствам. Даний сервіс містить інформацію про понад 50 популярних авіа напрямків по всьому світу, з наступною інформацією про вартість квитка, тривалість польоту, дати відправлення та прибуття, статус рейсу та ін., крім того використання сучасних веб технологій програмування зумовлює надзвичайно швидку роботу сервісу, а простий дизайн створює процес пошуку авіарейсу швидким та інтуїтивним.

Потенційним економічний результат даного є сервісу є розміщення реклами різноманітних туристичних агентств, співпраця із авіаперевізниками, що зумовить надходження грошових коштів у вигляді доходу.

## 5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення проектного рішення.

Подорож — переміщення якоюсь певною територією з метою її вивчення, а також із загальноосвітньою, пізнавальною, спортивною цілями. Подорожі відіграють велику роль у житті людини, а з появою авіалайнерів вони стали набагато швидшими.

Загалом подорож літаком має такі переваги:

— Найважливішою перевагою подорожей повітряним транспортом є його висока швидкість пересування. Всього за кілька годин можна перетнути тисячі кілометрів і опинитися в потрібному вам місці. Особливо це зручно при обмеженому часі.

— Літаки відрізняються високим сервісом. Пасажиру під час польоту надається їжа, напої. До того ж тут пропонуються всілякі способи, що дозволяють скоротати час, наприклад, подивитися фільм, ознайомитися з пресою або погортати журнал.

— Під час польоту пасажир відчуває себе максимально комфортно і зручно, що забезпечується зручними сидіннями, що дозволяють чудово відпочити і навіть поспати.

Проте процес пошуку та придбання авіаквитків, часте запізнення авіалайнерів зумовлює дискомфорт для споживачів, що у свою чергу є загрозою для відтоку потенційних або існуючих споживачів авіа перевезень.

На сьогодні розроблення сервісу, який допоможе з легкістю знайти авіаквитки у потрібне місце є дуже необхідним, у зв’язку із усуненням формальних кордонів між державами, глобалізацією та появою так званих «лоукостів», які пропонують авіа білети за помірними цінами для кожного. Для ретельного збору інформації перед розробкою продукту, необхідно дослідити існуючих конкурентів та їхню пропозицію на ринку:

1. WizzAir —  [угорсько-польська бюджетна авіакомпанія, із вільни](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%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)м серверним [веб-застосунок](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)ом для користування послугами компанії.

Переваги:

* Можливість додаткових послуг, таких як: бронювання готелів, виклик таксі та перегляд найблищих закладів харчування.
* Наявна українська локалізація
* Синхронізація із соціальними мережами
* Величезна кількість інформації про компанію та авіаперевезення.

Недоліки:

* Складний інтерфейс.
* Потребує багато кроків для купівлі або бронювання квитка.

2. Pegasus Airlines — бюджетна міжнародна авіакомпанія заснована в Туреччині.

Переваги:

* Простота і зручність інтерфейсу
* Пропозиції найпопулярніших рейсів.
* Величезна кількість інформації про компанію та авіаперевезення.

Недоліки:

* Відслідковування статусу рейса.
* Присутня реклама

3. British Airways — найбільша авіакомпанія та національний авіаперевізник Великої Британії, одна з найбільших в Європі, [із вільни](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%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)м серверним [веб-застосунок](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)ом для користування послугами компанії.

Переваги:

* Простота і зручність інтерфейсу.
* Пропозиції найдешевших квитків.
* Відслідковування статусу рейса.
* Величезна кількість інформації про компанію та авіаперевезення.

Недоліки:

* Немає укрїнської локалізації

В результаті огляду програмного забезпеченнядля моніторингу та планування пасажирських авіаперевезень було проаналізовано предметну область, а саме розглянуто специфіку авіаперевезень,основні повітряні з’єднання між країнами, популярні туристичні напрямки, а також проаналізовані продукти – аналоги, а саме веб орієнтовані застосунки. Визначені їх переваги та недоліки. В результаті було виявлено, що хоча є достатня кількість сервісів, які дають можливість відслідковувати авіаквитки, бронювати їх та моніторити статус рейсів, проте більшість із них мають ряд обмежень, а саме: відсутність української мови, складність пошуку, ускладнений функціонал непотрібними фільтрами пошуку, надмірна кількість рекламних оголошень. Як результат огляду продуктів-аналогів, необхідно взяти до уваги всі переваги та недоліки перед розробкою програмного забезпечення.

## 5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ.

Фактори зовнішнього середовища оцінюються за шкалою у проміжку від -5 до 5 балів. Оцінка -5 означає найбільш негативний вплив даного фактора, а найбільш позитивний вплив фактора – оцінка 5. Якщо фактор не має впливу на проектне рішення, то його оцінка становить 0 балів.

Фактори внутрішні оцінюються за шкалою від 0 до 5 балів, при цьому 0 означає відсутній розвиток, чи надзвичайно низький рівень функціонування даного фактора внутрішнього середовища, оцінка 5 свідчить про високий рівень розвитку фактора.

Сумою усіх факторних ваг є одиниця. Отож середня вагомість фактору – є коефіцієнтом. За допомогою даного коефіцієнта можна отримати рівень впливу факторів, помноживши його на оцінку експертів. Результати оцінок впливу факторів зовнішнього та внутрішнього середовищ на організацію та проектне рішення наведено у табл. 5.1.

**Результати експертного оцінювання впливу факторів зовнішнього та внутрішнього середовищ**

*Таблиця 5.1*

|  |  |  |  |
| --- | --- | --- | --- |
| **Фактори** | **Середня експертна оцінка, бали** | **Середня вагомість факторів** | **Зважений рівень впливу, бали** |
| 1 | 2 | 3 | 4 |
| *Фактори зовнішнього середовища* |  | | |
| Споживачі | 5 | 0,11 | 0,55 |
| Постачальники | 0 | 0,1 | 0 |
| Конкуренти | -3 | 0,1 | -0,3 |
| Державні органи влади | 0 | 0,05 | 0 |
| Інфраструктура | 2 | 0,06 | 0,12 |
| Законодавчі акти | -1 | 0,1 | -0,1 |
| Профспілки, партії та інші громадські організації | 0 | 0,05 | 0 |
| Система економічних відносин в державі | 2 | 0,06 | 0,12 |
| Організації-сусіди | 1 | 0,01 | 0,01 |
| Міжнародні події | -1 | 0,01 | -0,01 |
| Міжнародне оточення | 2 | 0,03 | 0,06 |
| Науково-технічний прогрес | 4 | 0,07 | 0,28 |
| Політичні обставини | -1 | 0,06 | -0,06 |
| Соціально-культурні обставини | 1 | 0,05 | 0,05 |
| Продовження табл. 5.1 | | | |
| Рівень техніки та технологій | 4 | 0,04 | 0,16 |
| Особливості міжнародних економічних відносин | -1 | 0,02 | -0,02 |
| Стан економіки | -2 | 0,08 | -0,08 |
| **Загальна сума** | **-** | **1** | **0,78** |
| *Фактори внутрішнього середовища* |  | | |
| Цілі | 4 | 0,11 | 0,44 |
| Структура | 1 | 0,16 | 0,16 |
| Завдання | 3 | 0,07 | 0,21 |
| Технологія | 5 | 0,2 | 1 |
| Працівники | 3 | 0,21 | 0,63 |
| Ресурси | 1 | 0,25 | 0,25 |
| **Загальна сума** | **-** | **1** | **2,69** |

Згідно із результатами експертного оцінювання впливу факторів зовнішнього та внутрішнього середовищ, найпозитивніший вплив на проектне рішення програмного продукту мають наступні фактори зовнішнього середовища: *споживачі*, оскільки саме кількість відвідувачів застосунку та цілодобовий трафік користувачів нададуть можливість до отримання прибутку та розширення функціоналу сервісу; *науково-технічний прогрес та рівень техніки та технологій*, саме ці ключові фактори відіграють вирішальну роль під час процесу розробки програмного продукту, забезпечують швидкодію програмного продукту та використання численних підходів та методик до проектного дизайну та гнучкої розробки у короткий період із високою якістю продукту. Серед факторів внутрішнього середовища найбільш вагомі – технології та цілі.

Негативний впливають наступні наведені фактори зовнішнього середовища: *конкуренти*; *особливості міжнародних економічних відносин та стан економіки*, оскільки економічний стан країни та її міжнародні відносини визначають напрямок руху галузей економіки, інвестиційну привабливість та ін..

## 5.4. Формування стратегічних альтернати

**5.4.1 Перша група стратегічних альтернатив**

Критеріями поділу першої групи альтернативних стратегій розвитку є вже існуючий продукт (програмне забезпечення) та новий, а також супутні їм послуги.



Рис. 5.1 Стратегічні альтернативи група №1

**Стратегія розробки повністю нового продукту (проектного рішення**)передбачає планування, проектування та розробку нового програмного забезпечення із подальшим виходом на ринок.

**Стратегія розвитку уже існуючого програмного продукту**, характеризується удосконаленням, оновленням, модернізацією існуючого продукту для підвищення його якісних властивостей, зростаючих потреб споживачів.

**Стратегія розвитку наявного на ринку продукту (проектного рішення) з супутніми послугами** характеризується додаванням до існуючого продукту супутніх послуг для потенційного збільшення прибутку, випередження конкурентів: служба підтримки, встановлення продукту, гарантійне обслуговування тощо.

**Стратегія розвитку нового продукту з супутніми послугами** характеризується проектуванням та розробкою нового програмного забезпечення поряд із включеними супутніми послугами, які дають можливість збільшити привабливість продукту на ринку, надають конкурентну перевагу, але є досить ресурсоємкими.

**5.4.2 Стратегічні альтернативи, група №2**

Альтернативні стратегії другої групи поділяються за такими критеріями, як новий або існуючий продукт (чи проектне рішення), а також відсутній чи наявний ринок.

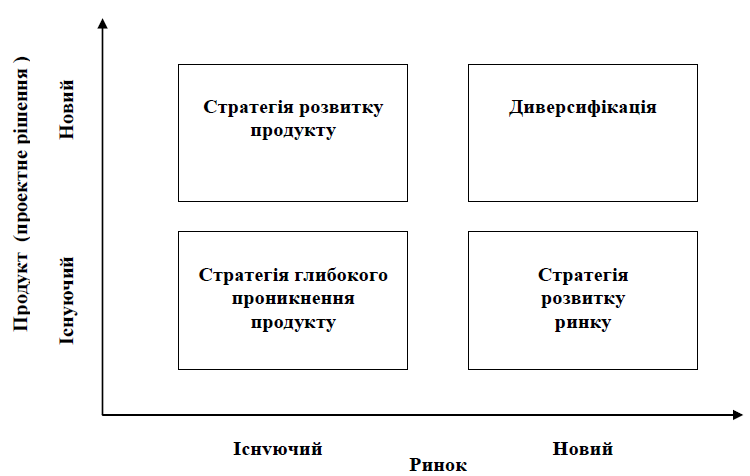


Рис. 5.2 Стратегічні альтернативи група №2

**Стратегія глибокого проникнення на ринок (концентрація)** – полягає глибокому аналізі концентрації пропозиції продукту на існуючому ринку, що зможе збільшити рівень збуту. Досить агресивна маркетингова стратегія забезпечує необхідний ефект зростання. Ця стратегія є ефективною, коли ринок відносно новий та темпи розширення його є помірними або низькими. Організація намагається збільшити інтенсивність руху товарів, ефективність рекламної кампанії і використовує найбільш конкурентоспроможні ціни, таким чином розширює ринок для збуту товарів, здобуваючи конкурентну перевагу.

**Щодо стратегії розвитку ринку**, то вона ефективна при виході компанії на нові ринки збути із уже готовим продуктом для продажу. Дана стратегія передбачає орієнтацію на нові ніші ринку, охоплення нових груп цільових споживачів або орієнтацію на нові іноземні ринки збуту.

**Стратегія розвитку товару** передбачає удосконалення та модернізацію вже існуючого товару чи групи товарів для забезпечення зростання продажів, охоплення нових потенційних споживачів. Вимагає значних часових та фінансових затрат та є ризикованою, оскільки кінцева модернізацію продукту може викликати хибну реакцію в уже існуючих користувачів та споживачів продукту.

**Стратегія диверсифікації** – передбачає розробку цілком іншого типу продукції для іншого сегменту економіки, означає повне або часткове перепрофілювання організації. Рівень ризику в неї найвищий.

**5.4.3 Вибір стратегії**

Після детального вивчення та аналізу запропонованих стратегічних альтернатив розробки продукту та виходу на ринок наступні висновки були зроблені:

Відповідно до групи альтернативних стратегій розвитку, критеріями поділу яких є існуючий продукт, а також супутні додаткові послуги, було обрано необхідну стратегію розробки нового продукту, що зумовлено новизною ідей, які будуть імплементовані у новому застосунку, що відсутні у продуктах конкурентів та усунуть недоліки уже існуючих сервісів, що надасть конкурентну перевагу організації.

Відповідно до групи альтернативних стратегій розвитку, критеріями поділу яких є існуючий ринок та продукт або цілком новий ринок та продукт, була обрана стратегія глибокого проникнення існуючого продукту, оскільки ефективна та агресивна маркетингова стратегія необхідна для розповсюдження та збільшення кількості користувачів, залученні нових партнерів, укладення співпраць, що зумовить необхідних ріст частки ринку.

## 5.5. Бюджетування

Бюджетування є важливою та комплексно обґрунтованою системою розрахунку витрат на проектування, виготовлення та реалізацію продукту, що дає можливість здійснити повний аналіз витрат за категоріями та розробити заходи щодо підвищення рентабельності. Важливим є визначення собівартості продукту, який перебуває у розробці та економічно обґрунтувати доцільність вибору однієї із стратегій.

Також зроблено обрахунок амортизації матеріалів та комплектуючих виробів, ціна за одиницю товару яких вища 6000 грн.

Для визначення амортизації використає наступну формулу:

**А = (ВПК / Т) / 365 ∗ КРД**, де

* ВПК-первісна вартість одного комп’ютера
* КРД-кількість робочих днів працівника.
* Т-термін експлуатації комп’ютера, який становить 2 роки.

А1 = (16000/2)/365\*35 = 8000 грн.

А2 = (16000/2)/365\*10 = 219,17 грн.

Загальна сума амортизації: 8219,17 грн.

*Таблиця 5.2*

Бюджет витрат матеріалів та комплектуючих виробів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва матеріалів та комплектуючих** | **Марка, тип, модель** | **Фактична кількість, шт.** | **Ціна за одиницю, грн.** | **Разом, грн.** |
| Ноутбук | HP ProBook | 2 | (16000, 16000) | 8219 |
| Wifi роутер | TP-LINK TL-WR720N | 1 | 400 | 400 |
| Разом: | - | - | - | 8619 |

*Таблиця 5.3*

Бюджет витрат на оплату праці

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Посада,**  **спеціальність** | **Кількість працівників, осіб** | **Час роботи, дні** | **Денна заробітна плата працівників, грн.** | **Сума витрат на оплату праці, грн.** |
| Основна заробітна плата | | | | |
| Дизайнер | 1 | 5 | 500 | 2500 |
| Програміст | 2 | 35 | 500 | 35000 |
| Тестер | 1 | 10 | 350 | 3500 |
| Разом | 3 | 85 | 1250 | 41000 |

*Таблиця 5.4*

Бюджет загальновиробничих витрат

|  |  |
| --- | --- |
| **Статтi витрат** | **Сума, грн.** |
| *Змiннi загальновиробничi витрати, у т.ч.:* |  |
| - заробітна плата допоміжного персоналу; | 3200 |
| - витрати на МШП; | 150 |
| - витрати на електроенергію | 500 |
| - інші змінні витрати; | 100 |
| Разом змiнних витрат: | 3950 |
| *Постiйнi загальновиробничi витрати, у т.ч.:* |  |
| - комунальнi послуги; | 950 |
| - витрати на оренду; | 5000 |
| - витрати на ремонт; | 500 |
| - iншi постiйнi витрати; | 200 |
| Разом постiйних витрат: | 5700 |
| *Разом загальновиробничих витрат:* | 11850 |

*Таблиця 5.5*

**Бюджет адміністративних витрат та витрат на збут**

|  |  |
| --- | --- |
| **Статтi витрат** | **Сума, грн.** |
| 1 | 2 |
| *Адмiнiстративнi витрати, у т.ч.:* |  |
| - заробiтна плата адмiнiстративного персоналу; | 3200 |
| - витрати на МШП; | 150 |
| - витрати на сплату податкiв i зборiв; | 1800 |
| - знос адмiнiстративного обладнання; | 350 |
| Продовження табл. 5.5 | |
| Разом адмiнiстративних витрат: | 5500 |
| *Витрати на збут, у т.ч.:* |  |
| - заробiтна плата менеджерiв зi збуту; | 7000 |
| - витрати на гарантiйний ремонт; | 0 |
| - витрати на гарантiйне обслуговування; | 0 |
| - витрати на налагодження i експлуатацiю; | 1500 |
| - витрати на паливо-мастильнi матерiали; | 0 |
| - витрати на рекламу; | 9000 |
| Разом витрат на збут: | 17500 |

*Таблиця 5.6*

**Зведений кошторис витрат на розробку проектного рiшення (продукту)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Статтi витрат** | **Одиницi вимiру** | **Фактична кiлькiсть, шт.** | **Цiна одиницi, грн.** | **Разом, грн.** |
| Сировина i матерiали | шт | 3 | (16000,16000,400) | 8619 |
| Паливо та електроенергiя на технологiчнi цiлi | кВт | 4545 | 0.99 | 450 |
| Основна заробiтна плата | грн | 3 | (2500,35000, 3500) | 41000 |
| Додаткова заробiтна плата | грн | - | - | - |
| Вiдрахування на соцiальне страхування | грн | 3 | (915,12810,915) | 14640 |
| Продовження табл. 5.6 | | | | |
| Загальновиробничi витрати, у т.ч.: |  | | | |
| - змiннi; | грн | - | - | 3950 |
| - постiйнi; | грн | - | - | 5700 |
| *Разом виробничих витрат:* | грн | - | - | **74359** |
| Адмiнiстративнi витрати | грн | - | - | 5500 |
| Витрати на збут | грн | - | - | 17500 |
| Iншi операцiйнi витрати | грн | - | - | 0 |
| *Разом виробничих i операцiйних витрат:* | грн | - | - | **97359** |

Для того, щоб визначити фінансові результати, необхідно розрахувати вартість програмного продукту, який розробляється. Ціна визначається на основі суми операційних та виробничих витрат з врахуванням рентабельності виробництва 26%. Ціна = 97359 \* 0.26 + 97359 = 122672,3 грн.

Ціна з ПДВ = 122672,3 \* 1,2 = 147206,8 грн.

Отже, вартість програмного забезпечення становить 147206,8 грн.

*Таблиця 5.7*

**Бюджет фінансових результатів**

|  |  |
| --- | --- |
| **Показники** | **Сума, грн.** |
| 1 | 2 |
| Дохід від реалізації продукції (1 шт) | 147206,8 |
| Податок на додану вартість (20%) | 24534,5 |
| Чистий дохід від реалізації продукції | 122672,3 |
| Собівартість реалізованої продукції | 97359 |
| Валовий прибуток | 25313,3 |
| *Операційні витрати:* | |

|  |
| --- |
| Продовження табл. 5.7 |

|  |  |
| --- | --- |
| - адміністративні витрати | 5500 |
| - витрати на збут | 17500 |
| - інші операційні витрати; | 0 |
| Фінансовий результат від операційної діяльності | 2313,3 |
| Податок на прибуток (18%) | 416,39 |
| **Чистий прибуток (збиток)** | **1896,91** |

## 5.6. Остаточний вибір стратегії

Провівши економічну оцінку доцільності проектування та розроблення програмного продукту, а саме веб сервісу для пошуку, моніторингу авіаквитків, відслідковування їхнього поточного статусу було визначено, що чистий прибуток від реалізації однієї продукції становитиме **1896,91** грн.

Проаналізувавши предметну область та порівнявши з конкурентами, було обрано стратегію входу на ринок із новим розробленим продуктом, через ряд недоліків у товарах конкурентах, які буде усунено даним програмним забезпеченням. Отже, ефективною стратегію для даного продукту буде стратегія глибокого проникнення на ринок за рахунок агресивних маркетингових дій стосовно конкурентів.

Після проведення розрахунку витрат, пов’язаних з виготовленням та реалізацією продукту, було визначено собівартість програмного продукту у розмірі **97359** грн., основною статтею витрат становить заробітна плата інженерного технічного персоналу, оскільки розробка інформаційного продукту потребує значних інтелектуальних затрат у програміста, що відповідно винагороджується.

Опираючись на результати економічного аналізу для розроблюваного продукту, випливає, що продукт має позитивне економічне підґрунтя для реалізації та впровадження на ринку веб систем.

# Висновки

В результаті виконання бакалаврської кваліфікаційної роботи я проаналізував існуючі додатки – аналоги, вивчив та оцінив різні алгоритми для отримання опрацювання даних з мережі, зібрав та проаналізував вимоги до програмного продукту, в результаті чого сформовано специфікацію вимог до системи. Також я вивчив архітектури MVP та використав цей шаблон під час проектування системи, розробив мобільний додаток та здійснив його тестування.

Проаналізувавши продукти конкуренти я виявив, що жоден з продуктів аналогів не володіє достатньою функціональністю. Є додатки які отримують інформацію тільки про один вид спорту, звужуючи при цьому аудиторію користувачів. Є такі, які отримують інформацію про спортивні події, але в них відсутня букмекерська інформація та прогнози. Тому постає необхідність в користуванні декількома мобільними додатками, кожний з яких окремо надає певний функціонал, або ж в реалізації нового додатку, що й було здійснено.

Реалізовано функції отримання та парсингу даних з мережі, опрацювання даних спортивних подій, «живий» пошук спортивної події за пошуковою фразою, отримання даних турнірних таблиць, різноманітні фільтри, можливість підписуватись на улюблені ліги та команди, автоматичні оновлення даних та сповіщення.

Виконано тестування та відлагодження мобільного додатку. Для цього я виконав GUI тестування, модульне тестування основних модулів системи. Крім того я здійснив статичний аналіз коду за допомогою Android Studio. В результаті було виправлено помилки.

В майбутньому я планую вдосконалювати мобільний додаток, додаючи в нього нові можливості, такі як підтримку нових видів спорту, отримання інформації від нових букмекерських контор, оптимізація алгоритму опрацювання даних спортивних подій, покращення або заміна API сервера для економії інтернет трафіку.

# список Літератури

1. Архітектура клієнт-сервер [Електронний ресурс] : – Режим доступу : <http://svitmandr.com.ua>
2. Сайт аналог авіакомпанії WizzAir [Електронний ресурс] : – Режим доступу : <https://wizzair.com/uk-ua/main-page#/>
3. Сайт аналог авіакомпанії British Airways [Електронний ресурс] : – Режим доступу : <https://www.britishairways.com/travel/home/public/en_ua>
4. Архітектура клієнт-сервер [Електронний ресурс] : – Режим доступу : <http://www.4stud.info/networking/lecture5.html>
5. AngularJS [Електронний ресурс] : – Режим доступу: <https://docs.angularjs.org/api>
6. WebApi2 [Електронний ресурс] : – Режим доступу: <https://www.asp.net/web-api>
7. Entity Framework [Електронний ресурс] : – Режим доступу : <https://docs.microsoft.com/en-us/aspnet/core/data/ef-mvc/intro>
8. Документація по мові C# [Електронний ресурс] : – <https://docs.microsoft.com/en-us/dotnet/articles/csharp/csharp>
9. Документація по середовишу розробки Visual Studio [Електронний ресурс] : – <https://www.visualstudio.com/en-us/docs/vs/overview>
10. Документація по JavaScript [Електронний ресурс] : – Режим доступу : <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>
11. Документація по Bootstrap [Електронний ресурс] : – Режим доступу : <https://www.w3schools.com/bootstrap>
12. Документація по HTML5 [Електронний ресурс] : – Режим доступу : <https://www.w3schools.com/html/default.asp>
13. Документація по середовишу розробки Web Storm [Електронний ресурс] : – Режим доступу : <https://www.jetbrains.com/webstorm/documentation/>
14. Документація по CSS3 [Електронний ресурс] : – Режим доступу : <https://www.w3schools.com/css/default.asp>

# Додаток А. Діаграма варіантів використання

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Виконав | Гаврилюк А.М. | Підпис |  | Діаграма варіантів використання | Сторінка |
| Перевірив | Коротеєва Т.О. | Підпис |  | 1 з 1 |
| ІКНІ кафедра ПЗ ПІ-43 | | | |

|  |  |  |
| --- | --- | --- |
| ІКНІ кафедра ПЗ ПІ-43 |  |  |

# Додаток Д. Тестові випадки використання програмного забезпечення

*ID RM-1 : Пошук авіарейсів*

*Таблиця 1*

|  |  |
| --- | --- |
| *Summary:* Перевіряється чи зможе користувач знайти бажані авіарейси | |
|  | |
| *Steps:* | *Expected Results:* |
| 1. Відкрити головну сторінку сайту. 2. Заповнити форму для пошуку рейсів(пункт відправлення, пункт прибуття, дату відправлення та клас літака). 3. Натиснути на кнопку пошук | 1. З’являється головна сторінка. 2. Дані заповнені коректно. 3. З’являється сторінка зі списком усіх можливих рейсів за заданими параметрами. |

Резолюція тестового випадку *–* **Passed**

*ID RM-2 : Перегляд детальної інформації про рейс*

*Таблиця 2*

|  |  |
| --- | --- |
| *Summary:* Перевіряється чи зможе користувач переглянути детальну інформацію про вибраний авіарейс | |
| *Pre-condition:* Виконано тестовий випадок RM-1. | |
| *Steps:* | *Expected Results:* |
| 1. Виконавши пошук рейсів, вибрати бажаний рейс зі списку можливих. 2. Натиснути на посилання «Детальніше». | 1. Вибрано рейс зі списку можливих. 2. З’являється сторінка із детальною інформацією про вибраний рейс. |

Резолюція тестового випадку *–* **Passed**

*ID RM-3 : Реєстрація в системі*

*Таблиця 3*

|  |  |  |  |
| --- | --- | --- | --- |
| *Summary:* Перевіряється чи користувач буде доданий у системі | | | |
| *Steps:* | | *Expected Results:* | |
| Продовження табл. 3 | |
| 1. Перейти на сторінку реєстрації, натиснувши на посилання «Реєстрація» у верхньому праву куті. 2. Заповнити усі поля вводу. 3. Натиснути на кнопку «Зареєструватись». | | 1. Відкривається сторінка реєстрації. 2. Поля вводу заповнені коректно. 3. Користувач успішно зареєстрований у системі та залогінений. | |

Резолюція тестового випадку *–* **Passed**

*ID RM-4 : Перегляд списку авіарейсів, пошук та фільтрування.*

*Таблиця 4*

|  |  |
| --- | --- |
| *Summary:* Перевіряємо чи коректно відобразиться список рейсів після фільтрування та пошуку. | |
| *Steps:* | *Expected Results:* |
| 1. Відкриваємо сторінку зі списком рейсів, натиснувши на посилання у верхній панелі навігації «Список рейсів». 2. Вибираємо пукнт відправлення та пункт прибуття. 3. Очищаємо фільтри та вводимо в поле пошуку назву пункту відправлення/прибуття. | 1. Сторінку зі списком рейсів відкрито. 2. Список рейсів змінився відповідно до вибраного пункту відправлення та прибуття. 3. Список рейсів змінився відповідно до введеного слова в полі пошуку. |

Резолюція тестового випадку *–* **Passed**

*ID RM-5 : Замовлення квитків*

*Таблиця 5*

|  |  |  |
| --- | --- | --- |
| *Summary:* Перевіряємо проходять сповіщення після настання подій, таких як гол забитий на матчі, який відзначено як «Вибране» | | |
| *Pre-condition:* Виконано тестовий випадок RM-2 та користувач увійшов в систему. | | |
| *Steps:* | *Expected Results:* | |
| Продовження табл. 5 | | |
| 1. Після перегляду детальної інформації про рейс. Натискаємо на кнопку «Здійснити замовлення». 2. Заповняємо усі необхідні поля вводу. 3. Вибираємо місця натиснувши на кнопку «Обрати місця». 4. Після вибору місць натиснути на кнопку «Ок». 5. Натискаємо кнопку продовжити. 6. Натискаємо на кнопку Оплатити/Забронювати. | 1. З’являються додаткові поля для замовлення квитків. 2. Поля заповнені коректно. 3. З’являється діалогове вікно для вибору місць, після вибору місць. 4. Діалогове вікно зникає. 5. Відповідно до вибраного методу реєстрації з’являється або форма для оплати, або кнопка «Забронювати» 6. З’являється повідомлення про успішно виконану операцію. | |

Резолюція тестового випадку *–* **Passed**

*ID RM-6 : Перевірка сторінки статусу рейсу*

*Таблиця 6*

|  |  |
| --- | --- |
| *Summary:* Перевірка на коректність виведення інформації про статус рейсу | |
| *Steps:* | *Expected Results:* |
| 1. Відкриваємо сторінку для пошуку статусу рейсу, натиснувши на посилання у верхній панелі навігації «Статус рейсу». 2. Вибирає дату. 3. Вибираємо пункт відправлення 4. Натискаємо на кнопку «Пошук» | 1. Сторінку зі статусом рейсу відкрито. 2. Дата початку рейсу вибрана. 3. Пункт відправлення вибрано. 4. З’являються рейси за датою та пунктом відправлення. |

Резолюція тестового випадку *–* **Passed**

*ID RM-7 : Перегляд списку замовлень та детальної інформації про замовлення.*

*Таблиця 7*

|  |  |  |
| --- | --- | --- |
| *Summary:* Переглядаємо список з усіма здійсненними замовленнями, а також детальної інформації про вибране замовлення. | | |
| *Pre-condition: К*ористувач увійшов в систему. | | |
| Продовження табл. 7 | | |
| *Steps:* | *Expected Results:* | |
| 1. Відкриваємо сторінку для перегляду списку замовлень, натиснувши на посилання у верхній панелі навігації «Список замовлень». 2. Вибираємо замовлення та нитискаємо на посилання «Детальніше». | 1. Сторінку зі списком замовлень відкрито. 2. З’являється інформація про вибране замовлення. | |

Резолюція тестового випадку *–* **Passed**

*ID RM-8 : Редагування даних в особистому кабінеті*

*Таблиця 8*

|  |  |
| --- | --- |
| *Summary:* Перевіряємо чи дані будуть змінені в остобистому кабінеті після їх редагування. | |
| *Pre-condition: К*ористувач увійшов в систему. | |
| *Steps:* | *Expected Results:* |
| 1. Відкриваємо сторінку особистого кабінету, натиснувши на посилання у верхньому куті панелі навігації «Особистий кабінет». 2. Натискаємо на «Налаштування». 3. Змінюємо ім’я та особисте фото. 4. Натискаємо на кнопку «Змінити». 5. Змінюємо пароль. 6. Натискаємо на копку «Змінити пароль» | 1. Сторінка особистого кабінету відкрито. 2. З’являється діалогове вікно з можливістю зміни даних. 3. Нове ім’я введено та фото вибране. 4. Виведено повідомлення про успішно виконані зміни. Та після перезавантеження сторінки видно нові дані. 5. Виведено повідомлення про успішну зміну пароля. 6. При повторному входу в систему потрібно ввести новий пароль. |

Резолюція тестового випадку *–* **Passed**

# Додаток Е. Інструкція користувача

**Е.1. Компоненти програмного забезпечення**

Мобільний додаток написаний на мові Java. Для цього використовувалось середовище програмування Android Studio 2.1. Дана програма може працювати під управлінням ОС Android 4.0.4 і вище.

Для роботи програми потрібний доступ до інтернет мережі, частота процесора не нижче 1 GHz, оперативна пам’ять понад 512 Mb та мінімум 20 Mb вільного простору на диску.

Таблиця Е.1

Файли, що необхідні для коректної роботи мобільного додатку

|  |  |  |  |
| --- | --- | --- | --- |
| № | Файл | Призначення | Належить проекту |
| 1 | Live2Sport.apk | Виконавчий файл | Мобільний додаток для опрацювання та відображення спортивних подій і букмекерських ставок |

**E.2. Встановлення програмного забезпечення**

Для для установлення мобільного додатку потрібно скопіювати файл Livi2Sport.apk в постійну пам’ять смартфону, наприклад, на sd-карту або в пам’ять телефону. Після цього слід установити галочку «Дозволяти установку сторонніх додатків» в системних налаштуваннях смартфону. Зайти в директорію з файлом додатку та натиснути на нього. Система може спитати про дозвіл встановлення програми із стороннього джерела та дозвіл на використання інтернету. В обох випадках потрібно погодитись.

**Е.3. Базові функції ПЗ**

Мобільний додаток дозволяє виконувати такі базові функції як вибрати вид спорту за замовчуванням, отримати інформацію про спортивні події вибраного виду спорту, отримати детальну інформацію про спортивну подію із списком букмекерських коефіцієнтів та переходом на сайти букмекерських контор. Також програма отримує турнірні таблиці спортивних змагань, наприклад, футбольних ліг. Фільтрувати спортивні події за такими критеріями як активні спортивні події, спортивні події із прогнозами, «вибрані події». Наявний пошук. Також можна підписатись на «вибрані» ліги, команди (учасники спортивних змагань) та отримувати інформацію про спортивні події, які з ними пов’язані. Є можливість отримувати сповіщення. Є можливість налаштувати додаток.

Для того, щоб додати команду чи лігу у «вибране» та отримувати інформацію про них потрібно:

1. В зайти в вікно «Explore» знайти потрібну лігу чи команду. Для цього можна скористатись пошуком, ввівши потрібну пошукову фразу.
2. Відзначити потрібну команду чи лігу зірочкою.
3. При потребі можна налаштувати додаток (зайти в налаштування, ввімкнути автоматичні оновлення, сповіщення, вибрати частоту оновлення).
4. Зайти в вікно «Feed» де буде відображатись «вибрані» події.
5. Події можна відфільтрувати, наприклад, відображати тільки ті, які йдуть в даний момент.
6. Для перегляду турнірної таблиці ліги натиснути на неї.
7. Для перегляду букмекерської інформації про спортивну події натиснути на цю подію в «Feed». Відобразиться детальна інформація про цю подію.

**Е.4. Аналіз помилок та можливих проблем**

В разі неправильної роботи програми в першу чергу потрібно перевірити наявність та якість інтернет з’єднання. Якщо проблема не вирішена то перезавантажити додаток, очистити кеш та відключити оновлення та сповіщення. Якщо й це не допомогло то переустановити додаток.

# Додаток Ж. Лістинг коду

***Проектування алгоритму завантаження та парсингу даних з мережі***

TvEventsFetchr – клас, що відповідає за завантаження та парсинг даних спортивних подій і букмекерської інформації, що приходить із сервера.

Public public class TvEventsFetchr {

Нижче наведений алгоритм парсингу даних спортивних подій із json.

Void parseMatches(List<Match> items, String jsonString) {

try {

//отримуємо головний json обєкт, в якому містяться інші обєкти

JSONObject mainJsonObject = (JSONObject) new JSONTokener(jsonString.toString())

.nextValue();

//отримуємо назву виду спорту

String sportTypeKey = mainJsonObject.names().get(0).toString();

if (sportTypeKey != null) {

//формуємо об’єкт який містить масив спортивних подій

JSONArray array = mainJsonObject.getJSONArray(sportTypeKey);//sport type array

//із json масиву витягуємо окремо кожну подію

for (int i = 0; i < array.length(); i++) {

JSONObject jsonObject = array.getJSONObject(i);

добавляємо її список подій – моделі даних

items.add(new Match(jsonObject));

//Log.i(TAG, «Json object : « + jsonObject.toString());

}

}

}catch (Exception e) { Log.d(TAG, e.getMessage()); } }

Нижче наведений алгоритм парсингу букмекерських даних із json.

//на вхід отримуємо текстові дані в форматі json

List<Bookmaker> parseBookmakers(String jsonString) throws JSONException{

List<Bookmaker> bookmakers = new ArrayList<>();//сюди будуть записуватись списку букмекерів

JSONArray jsonObjectArr = new JSONArray(jsonString);

//отримуємо посилання на оголовний обєкт json

JSONObject jsonObject = new JSONObject(jsonObjectArr.get(0).toString());

//отримуємо список отрибутів json в них містяться назви букмекерів

JSONArray jsonKeys = jsonObject.names();

for (int i = 0; i < jsonKeys.length(); i++) {

String key = (jsonKeys.get(i).toString());//берем поточний атрибут

if (!(key.endsWith(«\_1\_link») ||key.endsWith(«\_2\_link») ||key.endsWith(«\_x\_link») || key.equals(«event\_id»))){//берем тільки ті, що відносяться до букмекерів, фільтруєм решту атрибутів

String bookmakerName = key;//поточний атрибут – назва букмекера

//значення атрибуту – букмекерські коефіцієнти в форматі W:D:L (виграш, нічия, поразка)

String koefsStr = jsonObject.get(bookmakerName).toString();

if (koefsStr == null || koefsStr.equals(«»))

continue;

String[] koefs = koefsStr.split(«:»);//отримуємо значення коефіцієнтів

if (bookmakerName.equals(«will»))//костиль – проблема не в мене а на сервері

bookmakerName = «whill»;

bookmakers.add(new Bookmaker(bookmakerName, koefs));//додаємо букмекера

}

}

return bookmakers; }

Отримуємо json обєкт зі спортинвою подією в коструктор моделі спортивної події. Після цього модель заповнюється.

Public Match(JSONObject json) throws JSONException{

try{//записуємо в полі моделі відповіддні значення із json обєктів

eventId = json.getString(JSON\_EVENT\_ID);

teamHome = json.getString(JSON\_TEAM\_HOME);

teamAway = json.getString(JSON\_TEAM\_AWAY);

scores1 = json.getString(JSON\_SCORES1).split(« «)[0];//without penalties

scores = json.getString(JSON\_SCORES);

league = json.getString(JSON\_LEAGUE);

time = json.getString(JSON\_TIME);

coeff = json.getInt(JSON\_COEFF);

updateScore = json.getString(JSON\_UPDATE\_SCORE);

eventPrediction = json.getString(JSON\_EVENT\_PREDICTION);

liveScore = json.optString(JSON\_LIVE\_SCORE, null);

extraInfo = parseExtraInfo(json.getString(JSON\_EXTRA\_INFO));

}catch (JSONException e){

Log.e(TAG, «Error parsing item match»);

e.printStackTrace();

throw e;

}

//парсинг букмекерської інформації

JSONArray jsonKeys = json.names();

final String EVENT\_LINK = «events\_link\_»;

//тут отримуємо список букмекерів, які мають відеотрансляції спортивної події

for (int i = 0; i < jsonKeys.length(); i++) {

String key = (jsonKeys.get(i).toString());

if (key.startsWith(EVENT\_LINK)){

String bookmakerName = key.replace(EVENT\_LINK, «»);

String isWatchLive = json.get(key).toString();

if (isWatchLive != null && isWatchLive.equals(«1»)){

if (bookmakerName.equals(«1xbet»))

bookmakerName = «marathon»;

mWatchLiveBookmakers.add(new Bookmaker(bookmakerName, true));

}

}

} }

Уривок коду із коструктора головної моделі спортивної події

public Match(JSONObject json) throws JSONException{

try{//записуємо в полі моделі відповіддні значення із json обєктів

eventId = json.getString(JSON\_EVENT\_ID);

teamHome = json.getString(JSON\_TEAM\_HOME);

teamAway = json.getString(JSON\_TEAM\_AWAY);

scores1 = json.getString(JSON\_SCORES1).split(« «)[0];//without penalties

scores = json.getString(JSON\_SCORES);

league = json.getString(JSON\_LEAGUE);

time = json.getString(JSON\_TIME);

coeff = json.getInt(JSON\_COEFF);

updateScore = json.getString(JSON\_UPDATE\_SCORE);

eventPrediction = json.getString(JSON\_EVENT\_PREDICTION);

liveScore = json.optString(JSON\_LIVE\_SCORE, null);

extraInfo = parseExtraInfo(json.getString(JSON\_EXTRA\_INFO));

}catch (JSONException e){

Log.e(TAG, «Error parsing item match»);

e.printStackTrace();

throw e;

}

//парсинг букмекерської інформації

JSONArray jsonKeys = json.names();

final String EVENT\_LINK = «events\_link\_»;

//тут отримуємо список букмекерів, які мають відеотрансляції спортивної події

for (int i = 0; i < jsonKeys.length(); i++) {

String key = (jsonKeys.get(i).toString());

if (key.startsWith(EVENT\_LINK)){

String bookmakerName = key.replace(EVENT\_LINK, «»);

String isWatchLive = json.get(key).toString();

if (isWatchLive != null && isWatchLive.equals(«1»)){

if (bookmakerName.equals(«1xbet»))

bookmakerName = «marathon»;

mWatchLiveBookmakers.add(new Bookmaker(bookmakerName, true));

}

}

} }

***Реалізація алгоритму опрацювання спортивних подій та букмекерських ставок***

Даний алгоритм здійснює основну обробку списку спортивних подій. На вхід отримуємо модель даних із неопрацьованими спортивними подіями із попреднього кроку. Алгоритм їх фільтрує, розбиває по групах (якщо це матчі, то здійснюється розподіл по лігах). Впорядковує групи та події всередині груп.

Public class MatchLab {

public static List<League> groupMatchesByLeagues(Context context, List<Match> matches){

//перевірка опції «завантажити тільки спортивні події»

if (ResourcesSolver.loadIsActiveMatches(context)){

List<Match> tempMatches = matches;

matches = new ArrayList<>();

//перевіряється конжна подія

for (Match match : tempMatches){

if (match.isActive())//чи подія активна (відбувається в даний момент)

matches.add(match);//вставляємо в список

}

}

//перевірка опції «завантажити тільки події з прогнозами»

if (ResourcesSolver.loadIsEventPredictionMatches(context)){

List<Match> tempMatches = matches;

matches = new ArrayList<>();

for (Match match : tempMatches){

if (match.getEventPrediction() != null)

//аналогічно здійснюється перевірка чи подія має прогноз

matches.add(match);

}

}

//на цьому кроці список подій вже пофільтрований

//створюємо обєкт який вміє порівнювати події за коефіцієнтами

//потім буде використаний при сортуванні

Comparator<Match> matchComparator = new Comparator<Match>() {

@Override

public int compare(Match lhs, Match rhs) {

return lhs.getCoeff() < rhs.getCoeff()? 1: -1;

}

};

//групуємо події за лігами

List<League> groupedMatches = new ArrayList<>();

Set<String> leagues = new TreeSet<>();//формуємо множину назв ліг

for (Match match : matches){//здіснюємо прохід по всіх подіях

String league = match.getLeague();//отримуємо лігу

if (!leagues.contains(league)){

leagues.add(league);//вставляємо лігу в множину

}

}//на виході отримуємо множину із назвами ліг

for (String league : leagues){//формуємо список ліг

List<Match> matchesInLeague = new ArrayList<>();

double maxCoeff = 0;

for (Match match : matches){

//формуємо список подій для конкретної ліги

//для цього порівнюємо назву його ліги з поточною

String matchLeague = match.getLeague();

if (league.equals(matchLeague)){

matchesInLeague.add(match);

}

}

//сортуємо подій всередині ліги по спаданню коефіцієнтів

Collections.sort(matchesInLeague, matchComparator);

//визначаємо максимальний коефіцієнт подій всередині ліги. Це буде загальний коефіцієнт ліги

maxCoeff = (double)matchesInLeague.get(0).getCoeff();

String countryFlagPath = CountryFlagsLab.get(context).getCountryPath(league);

//формуємо обєкт ліги

groupedMatches.add(new League(league, matchesInLeague, countryFlagPath, maxCoeff));

}

//здійснюємо сортування списку ліг за їхнім коефіцієнтом

Collections.sort(groupedMatches, new Comparator<League>() {

@Override

public int compare(League lhs, League rhs) {

if (lhs.getCoeff() == rhs.getCoeff()){

return lhs.getName().compareTo(rhs.getName());

}

return lhs.getCoeff() < rhs.getCoeff()? 1: -1;

}

});

//для кожної з ліг здійснюємо сортування списку подій

//таким чином, що спочатку йтимуть активні події, потім подіії, які ще не розпочались

//останніми будуть завершені події

// кожна з цих трьох груп буде впорядкована за спаданням коефіцієнтів події

for (League league : groupedMatches){

Collections.sort(league.getMatches());

}

/\*тут наведений перевантажений метод класу Match

public int compareTo(@NonNull Object another) {

Match match = (Match)another;

int thisKoef = 0, anotherKoef = 0;

if (isActive())//визначається ця подія є активна

thisKoef = 1;

else if (isEnded())//чи подія завершена

thisKoef = -1;

if (match.isActive())//чи інша подія активна

anotherKoef = 1;

else if (match.isEnded())//чи інша подія завершена

anotherKoef = -1;

if (thisKoef > anotherKoef){//порівнюються коефіцієнти подій

return -1;

}else if (thisKoef < anotherKoef){

return 1;

}\*/

return groupedMatches;

}}

Інші важливі класи та методи наведені в Додатку Ж.

Клас, який відповідає за автоматичні сповіщення, які приходять із сервера.  
public class NotificationService extends IntentService {  
   
 @Override  
 protected void onHandleIntent(Intent intent) {  
 Date currentDate = new Date();  
 String sportName = ResourcesSolver.loadSportType(this);  
 modelImp.loadMatches(currentDate, sportName, new EventsLoadingListener() {  
 @Override  
 public void onSuccess(List<League> leagues) {  
 if (leagues != null) {  
 generateNotifications(leagues);  
 } }  
  
 @Override  
 public void onFailure(String message) {  
 //hide all errors  
 }  
 });  
 }  
  
 private void generateNotifications(List<League> leagues) {  
  
 for (League league : leagues) {  
 List<Match> matches = league.getMatches();  
 for (Match match : matches) {  
 if (match.isActive()) {  
 generateInfoForMatch(match);  
 }  
 } }  
 }  
 private void generateInfoForMatch(Match match) {  
 String teamHome = match.getTeamHome();  
 String teamAway = match.getTeamAway();  
 String event = match.getUpdateScore();  
 String notificationInfo = null;  
 String notificationMessage = String.format(«%s - %s», teamHome, teamAway);  
 switch (event){  
 case «1»:  
 notificationInfo = String.format(«%s scored goal», teamHome);  
 break;  
 case «2»:  
 notificationInfo = String.format(«%s scored goal», teamAway);  
 break;  
 case "3":  
 notificationInfo = String.format("Both team scoed goal");  
 break;  
 default: break;  
 }  
 if (notificationInfo != null){  
 showNotification(notificationMessage, notificationInfo);  
 } }  
 private void showNotification(String notificationMessage, String notifInfo) {  
 NotificationCompat.Builder builder =  
 new NotificationCompat.Builder(this)  
 .setSmallIcon(R.drawable.logo\_l2s)  
 .setContentTitle(getString(R.string.Live2Sport))  
 .setContentInfo(notifInfo)  
 .setContentText(notificationMessage);  
 Intent intent = new Intent(this, MainDrawerActivity.class);  
 //intent.putExtra(Main.IS\_LOAD\_FROM\_NOTIFICATION, true);  
 PendingIntent pi = PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG\_UPDATE\_CURRENT);  
 builder.setContentIntent(pi);  
 NotificationManager mNotificationManager =  
 (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);  
 mNotificationManager.notify(NOTIFICATION\_ID, builder.build());  
 }  
  
 public static void setServiceAlarm(Context context, boolean isOn){  
 Intent i = new Intent(context, NotificationService.class);  
 PendingIntent pi = PendingIntent.getService(context, 0, i, 0);  
 AlarmManager alarmManager = (AlarmManager)context.getSystemService(Context.ALARM\_SERVICE);  
 alarmManager.cancel(pi);  
 if (isOn){  
 alarmManager.setRepeating(AlarmManager.RTC, System.currentTimeMillis(),  
 PrefsHelper.getAutomaticRefreshInterval()\*1000,//convert to miliseconds  
 pi);  
 }else {  
 alarmManager.cancel(pi);  
 pi.cancel();  
 }  
 }}

Клас який відповідає за асинхронне завантаження та опрацювання даних спортивних подій  
public class FetchItemsTask extends AsyncTask<Object,Void,List<League>> {  
 public static final String TAG = "FetchItemsTask";  
  
 private Context context;  
 private ItemFetcherListener listener;  
 private String errorMessage = null;  
  
 public FetchItemsTask(Context context){  
 this.context = context;  
 }  
  
 @Override  
 protected void onPreExecute() {  
 super.onPreExecute();  
 errorMessage = null;  
 }  
  
 @Override  
 protected void onPostExecute(List<League> leagues) {  
 if (errorMessage == null){  
 if (listener != null)  
 listener.onItemsLoaded(leagues);  
 }else{  
 if (listener!= null)  
 listener.onFailure(errorMessage);  
 }  
 errorMessage = null;  
 }  
  
 @Override  
 protected List<League> doInBackground(Object... params) {  
 errorMessage = null;  
 Date date = (Date)params[0];  
 String sportType = (String)params[1];  
 @SuppressWarnings("deprecation")  
 String sdate = new SimpleDateFormat("yyyy-MM-dd").format(date);  
 List<Match> matchItems = null;  
 try {  
 matchItems = new TvEventsFetchr().fetchMatches(context, sdate, sportType);  
 List<League> leagues = MatchLab.groupMatchesByLeagues(context, matchItems);  
 return (leagues != null ? leagues : new ArrayList<League>());  
 }catch (Exception ex){  
 errorMessage = ex.getMessage();  
 Log.d(TAG, ex.getMessage());  
 }  
 return null;  
 }  
  
 public void setListener(ItemFetcherListener listener) {  
 this.listener = listener;  
 }  
  
 public interface ItemFetcherListener {  
 void onItemsLoaded(List<League> leagues);  
 void onFailure(String message);  
 }  
}

Клас моделі MVP, який предаставляє роботу головного списку спортивних подій. Він отримує команди від презентера, здійснює обробку інформації. В ньому присутні методи, які виконують завантаження даних спортивних подій, відзначення команд та ліг як вибрані, здійснення пошуку.

public class MatchesListModelImp {  
 private Context context;  
 private FetchItemsTask fetchItemsTask;  
  
 private boolean isExplore;  
  
 public MatchesListModelImp(Context context, boolean isExplore){  
 this.context = context;  
 this.isExplore = isExplore;  
 }  
  
 public void loadMatches(Date date, String sportType, final EventsLoadingListener listener){  
 fetchItemsTask = new FetchItemsTask(context);  
 fetchItemsTask.setListener(new FetchItemsTask.ItemFetcherListener() {  
  
 @Override  
 public void onItemsLoaded(List<League> leagues) {  
 if (isExplore) {  
 if (PrefsHelper.getExploreStarred())  
 leagues = getChosenLeagues(leagues);//filter  
 leagues = starredChosenEvents(leagues);  
 }  
 if (!isExplore)  
 leagues = getChosenLeagues(leagues);//filter  
 if (listener != null)  
 listener.onSuccess(leagues);  
 }  
  
 @Override  
 public void onFailure(String message) {  
 if (listener != null)  
 listener.onFailure(message);  
 }  
 });  
 fetchItemsTask.execute(date, sportType);  
 }  
  
 private List<League> starredChosenEvents(List<League> leagues) {  
 if (leagues == null)  
 return leagues;  
 ChosenSports chosenSports = ChosenService.get().getChosenSports();  
 String sportType = ResourcesSolver.loadSportType(context);  
 ChosenSportLeagues sportLeagues = chosenSports.getSportLeagues(sportType);  
 if (sportLeagues == null)  
 return leagues;//empty  
 List<ChosenLeague> leagueList = sportLeagues.getLeagueList();  
 if (leagueList == null)  
 return leagues;//empty  
 for (int i = 0; i < leagues.size(); i++){  
 League league = leagues.get(i);  
 ChosenLeague currentLeague = sportLeagues.getleague(league.getName());  
 if (currentLeague == null){  
 continue;  
 }else if (!currentLeague.isChosen()&&  
 (currentLeague.getTeamsSet() == null ||  
 currentLeague.getTeamsSet().size() == 0)) {  
 continue;  
 }else {  
 league.setIsStarred(currentLeague.isChosen());  
 for (int j = 0; j < league.getMatches().size(); j++) {  
 Match match = league.getMatches().get(j);  
 boolean isTeamHomeStarred = currentLeague.isTeamStarred(match.getTeamHome());  
 boolean isTeamAwayStarred = currentLeague.isTeamStarred(match.getTeamAway());  
 match.setIsTeamHomeStarred(isTeamHomeStarred);  
 match.setIsTeamAwayStarred(isTeamAwayStarred);  
 }  
 }  
 }  
 return leagues;  
 }  
  
 private List<League> getChosenLeagues(List<League> leagues){  
 if (leagues == null)  
 return null;  
 ChosenSports chosenSports = ChosenService.get().getChosenSports();  
 String sportType = ResourcesSolver.loadSportType(context);  
 ChosenSportLeagues sportLeagues = chosenSports.getSportLeagues(sportType);  
 if (sportLeagues == null)  
 return new ArrayList<>();//empty  
 List<ChosenLeague> leagueList = sportLeagues.getLeagueList();  
 if (leagueList == null)  
 return new ArrayList<>();//empty  
 for (int i = 0; i < leagues.size(); ){  
 League league = leagues.get(i);  
 ChosenLeague currentLeague = sportLeagues.getleague(league.getName());  
 if (currentLeague == null){  
 leagues.remove(i);  
 continue;  
 }else if (!currentLeague.isChosen()&&  
 (currentLeague.getTeamsSet() == null ||  
 currentLeague.getTeamsSet().size() == 0)) {  
 leagues.remove(i);  
 continue;  
 }else {  
 if (!currentLeague.isChosen()) {  
 for (int j = 0; j < league.getMatches().size(); ) {  
 Match match = league.getMatches().get(j);  
 boolean isExists = currentLeague.getTeamsSet().contains(match.getTeamHome())  
 || currentLeague.getTeamsSet().contains(match.getTeamAway());  
 if (!isExists) {  
 league.getMatches().remove(j);  
 continue;  
 }  
 j++;  
 }  
 if (league.getMatches().size() == 0) {  
 leagues.remove(i);  
 continue;  
 }  
 }  
 }  
 i++;  
 }  
 return leagues;  
 }  
  
 public void cancellLoading(){  
 fetchItemsTask.setListener(null);  
 }  
  
 public void processSearch(List<League> originalLeagues, String query, final EventsLoadingListener listener) {  
 if (query == null) {  
 if (listener != null)  
 listener.onSuccess(originalLeagues);  
 return;  
 }  
 List<League> leagues = new ArrayList<>();  
 for (League league : originalLeagues){  
 if (league.getName().toLowerCase().contains(query.toLowerCase())){  
 leagues.add(league);  
 }else{  
 List<Match> matches = new ArrayList<>();  
 for (Match match : league.getMatches()){  
 if (match.getTeamAway().toLowerCase().contains(query.toLowerCase()) ||  
 match.getTeamHome().toLowerCase().contains(query.toLowerCase()) ){  
 matches.add(match);  
 }  
 }  
 if (matches.size() > 0){  
 leagues.add(new League(league.getName(), matches, league.getCountryFlagPath(), league.getCoeff()));  
 }  
 }  
 }  
 if (listener != null)  
 listener.onSuccess(leagues);  
 }  
}

# Додаток З. Статичний аналіз коду за допомогою Android Studio

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Виконав | Шліхутка Б. Т. | Підпис |  | Статичний аналіз вихідного коду за допомогою Android Studio | Сторінка |
| Перевірив | Сердюк П. В. | Підпис |  | 1 з 1 |
| ІКНІ кафедра ПЗ ПІ-43 | | | |

# Додаток И. Список багів та зауважень від замовника

|  |  |  |  |
| --- | --- | --- | --- |
| № | Дата | Опис | Bug resolution |
| 1 | 8 червня 2015 | Некілкабельна дата в списку спортивних подій | FIXED |
| 2 | 8 червня 2015 | Зробити кнопку оновити як на LiveScore, яка б підтягувала нові дані | FIXED |
| 3 | 8 червня 2015 | Підправити відображення прапорів країн | FIXED |
| 4 | 9 червня 2015 | Немає риски між спортивними подіями списку | FIXED |
| 5 | 9 червня 2015 | Замінити N/A на “? : ?” при відображенні незіграних матчів. | FIXED |
| 6 | 10 червня 2015 | Програма повинна завантажувати дані більшої кількості попередніх днів | FIXED |
| 7 | 23 червня 2015 | Відображення матчів, які не почались. змінити «? -?» на знаки питання один під одним | FIXED |
| 8 | 23 червня 2015 | Для матчу, який розпочався показувати не знаки питання, а поточний рахунок та хвилину | FIXED |
| 9 | 2 липня 2015 | Програма аварійно завершується при завантаженні даних бейсболу | FIXED |
| 10 | 6 липня 2015 | Порядок відображення спортивних подій неправильний. Спочатку повинні йти ті що не почалися і ті, які ще йдуть. | FIXED |
| 11 | 23 липня 2015 | Не відображається посилання на watch live, для подій, що йдуть в деталях. | FIXED |

|  |  |  |  |
| --- | --- | --- | --- |
| 12 | 17 липня 2015 | Всі ліги зразу відкриті в списку | NOT A BUG |
| 13 | 17 липня 2015 | Додати американський футбол | FIXED |
| 14 | 17 липня 2015 | В детальній інформації про матч повинен відображатись детальний рахунок, як такий 4:0(1:0 3:0) | FIXED |
| 15 | 17 липня 2015 | Не оновлюються деталі спортивної події | FIXED |
| 16 | 21 липня 2015 | Програма аварійно завершується, в зв’язку з переповненням пам’яті | FIXED |
| 17 | 10 серпня 2015 | Американський футбол у списку видів спорту останній - він має бути на тому місці що й на сайті | FIXED |
| 18 | 10 серпня 2015 | Хокей в не тому місці і прапорці на хокеї Ісландії стоять | FIXED |
| 19 | 12 вересня 2015 | Деталі матчу - немає пробілу немає пробілу між закриваючою дужкою і вертикальною рискою. | FIXED |
| 20 | 12 вересня 2015 | Немає в списку MMA, eSports | FIXED |
| 21 | 20 листопада 2015 | Якщо є кінцевий результат – значить матч закінчився. А це поля scores - його і треба виводити в детальніше про матч – так і має бути по вигляду. А не писати в стовпчик результати по таймах. | FIXED |
| 22 | 14 січня 2016 | Замінити логотип для контори Марафон. | FIXED |

|  |  |  |  |
| --- | --- | --- | --- |
| 23 | 14 січня 2016 | Змінити посилання для контори Марафон | FIXED |
| 24 | 14 січня 2016 | Не відображається посилання на трансляцію контори Марафон | FIXED |

# Додаток К. Результати тестування першої версії перед релізом

|  |  |  |  |
| --- | --- | --- | --- |
| № | Дата | Опис | Bug resolution |
| 1 | 31 грудня 2015 | Список подій не завжди завантажується, хоча підключення до інтернету присутнє. | FIXED |
| 2 | 31 грудня 2015 | Час матчів в додатку та на сайті відрізняються на одну годину. | NOT A BUG |
| 3 | 31 грудня 2015 | Різна кількість ставок на сайті та в додатку. | NOT A BUG |
| 4 | 31 грудня 2015 | Деталі матчу – список букмекерів. Цифри ставок з’їжджають вниз, замале поле для відображення. | FIXED |
| 5 | 31 грудня 2015 | Різні дані на сайті і в додатку, деякі матчі відсутні. | FIXED |
| 6 | 31 грудня 2015 | Немає можливості вибрати дату, матчів, потрібно гортати весь час | FIXED |
| 7 | 31 грудня 2015 | Немає іконок на всі види спорту | NOT A BUG |
| 8 | 31 грудня 2015 | Рекомендація на ставку повинна бути після завершення матчу? | NOT A BUG |
| 9 | 31 грудня 2015 | В видах спорту в яких немає рахунку, так і мають бути знаки питання? | FIXED |