**АНОТАЦІЯ**

Основним завданням бакалаврської кваліфікаційної роботи є розробка веб сервісу для моніторингу та планування авіарейсів для авіакомпанії.

Робота складається із програмної реалізації проекту у вигляді веб сайту; пояснювальної записки, обсягом 64 сторінки; графічного матеріалу UML діаграм, звіту по тестуванні та інструкції користувача.

Виконуючи бакалаврську кваліфікаційну роботу було проаналізовано та описав предметну область програмного забезпечення досліджено існуючі програми аналоги, встановлено задачі та цілі для розробки програмного забезпечення, здійсно проектування проекту, виконано тестування програмного забезпечення.

Розроблений веб додаток надає користувачеві можливість переглядати весь список авіарейсів, здійснювати пошук чи фільтрацію конкретного рейсу, переглядати детальну інформацію вибраного рейсу, переглядати статус рейсів, замовляти або бронювати квитки відповідно до вибраного типу квитка, переглядати список замовлень та детальну інформацію про вибране замовлення, реєструватись та авторизуватись в системі, а також змінювати особисту інформацію.

**ABSTRAСT**

The main task of qualifications bachelor work to develop a web service for monitoring and planning passengers air freight for airline.

The work consists of a software project as a web site; explanatory note, volume 64 pages; graphic material UML diagrams, testing report and user guide.

During development a bachelor qualification work was analyzed and described the domain of software programs examined existing analogues, set objectives and goals for development of software, made the design of the project, completed testing software.

A web application allows user to view the entire list of flights, search or filter specific flight, view detailed information of the selected flights, view flight status, book or buy tickets according to the selected type of ticket, view a list of orders and details about selected order, register and authorize into the system and change personal information.

**ЗМІСТ**

[ВСТУП 7](#_Toc483779534)

[Розділ 1. Огляд програмного забезпечення для моніторингу та планування пасажирських авіаперевезень 8](#_Toc483779535)

[1.1. Проблеми компютеризації авіаперевезень 8](#_Toc483779536)

[1.2. Огляд програм аналогів 9](#_Toc483779537)

[1.3. Висновок 13](#_Toc483779538)

[Розділ 2. Постановка задачі для розробки програмного забезпечення для моніторингу та планування пасажирських авіаперевезень 14](#_Toc483779539)

[2.1. Загальна постановка задачі 14](#_Toc483779540)

[2.2. Вибір архітектури 14](#_Toc483779541)

[2.3. Вибір програмних засобів 15](#_Toc483779542)

[2.4. Специфікація вимог до програмного продукту 17](#_Toc483779543)

[2.5. Висновок 23](#_Toc483779544)

[Розділ 3. архітектурА і Проектування ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ Для моніторингу та планування пасажирських авіаперевезень 24](#_Toc483779545)

[3.1. Архітектура програмного продукту 24](#_Toc483779546)

[3.2. Проектування компонент 26](#_Toc483779547)

[3.3. Проектування класів 27](#_Toc483779548)

[3.4. Проектування бази даних 31](#_Toc483779549)

[3.5. Прототипування інтерфейсу користувача 32](#_Toc483779550)

[3.6. Висновок 36](#_Toc483779551)

[Розділ 4. Реалізація та тестування Програмного продукту 37](#_Toc483779552)

[4.1. Реалізація веб сервісу 37](#_Toc483779553)

[4.1.1. Реалізація серверної частини 37](#_Toc483779554)

[4.1.2. Реалізація клієнтської частини 38](#_Toc483779555)

[4.2. Опис роботи з програмним продуктом 39](#_Toc483779556)

[4.3. Тестування програмного продукту 46](#_Toc483779557)

[4.4. Висновок 47](#_Toc483779558)

[РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА 48](#_Toc483779559)

[5.1. Економічна характеристика програмного забезпечення для моніторингу та планування пасажирських авіаперевезень 48](#_Toc483779560)

[5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення проектного рішення. 49](#_Toc483779561)

[5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ. 51](#_Toc483779562)

[5.4. Формування стратегічних альтернатив 54](#_Toc483779563)

[5.5. Бюджетування 57](#_Toc483779564)

[5.6. Остаточний вибір стратегії 61](#_Toc483779565)

[Висновки 63](#_Toc483779566)

[список Літератури 64](#_Toc483779567)

[Додаток А. Діаграма варіантів використання 65](#_Toc483779568)

[Додаток Б. Діаграма діяльності 66](#_Toc483779569)

[Додаток В. Тестові випадки використання програмного забезпечення 67](#_Toc483779570)

[Додаток Д. Інструкція користувача 71](#_Toc483779571)

[Додаток Е. Лістинг коду 72](#_Toc483779572)

# ВСТУП

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

Пасажирські авіаперевезення – це найшвидший та найкомфортніший транспорт, адже можна подолати тисячі кілометрів за лічені години. Різноманітні авіакомпанії надають цікаві послуги та сервіси в обслуговуванні клієнтів. Часто буває, що клієнтові потрібно переглянути весь список авіарейсів, або переглянути статус конкретного рейсу, або замовити квиток не виходячи з домівки, для цього всього потрібно їхати в аеропорт, тратити свій час та запитувати все в 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. Проблеми компютеризації авіаперевезень

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 авіа транспортом є його висока швидкість пересування. Всього за кілька годин можна перетнути тисячі кілометрів і опинитися в потрібному вам місці.

— Подорож літаком є набагато цікавішою ніж поїздом чи автобусом. Адже не потрібно витрачати декілька днів на подорож.

— Літаки мають високий сервіс в обслуговуванні пасажирів. Під час польоту пасажиру надається їжа та напої, а також пропонуються всілякі додаткові послуги, що дозволяють скоротати час, наприклад, подивитися фільм.

— Пасажирські сидіння у літаках відповідають вимогам комфортності та зручності. Отже, пасажир може почувати себе максимально комфортно, має можливість чудово відпочити і навіть поспати.

— Вид з ілюмінатора надзвичайно красивий, можна побачити красиві пейзажі та білосніжні хмари.

— Сьогодні можна забронювати квиток телефоном або Інтернету і забути про незліченні черги. Причому, доступна послуга у будь-який час, найбільш зручний для вас.

— Літаком можна відправитися в будь-яку країну, наприклад перетнути континент.

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 – найзручніший і найкомфортніший засіб пересування. [1]

## 1.2. Огляд програм 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-застосунк](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]

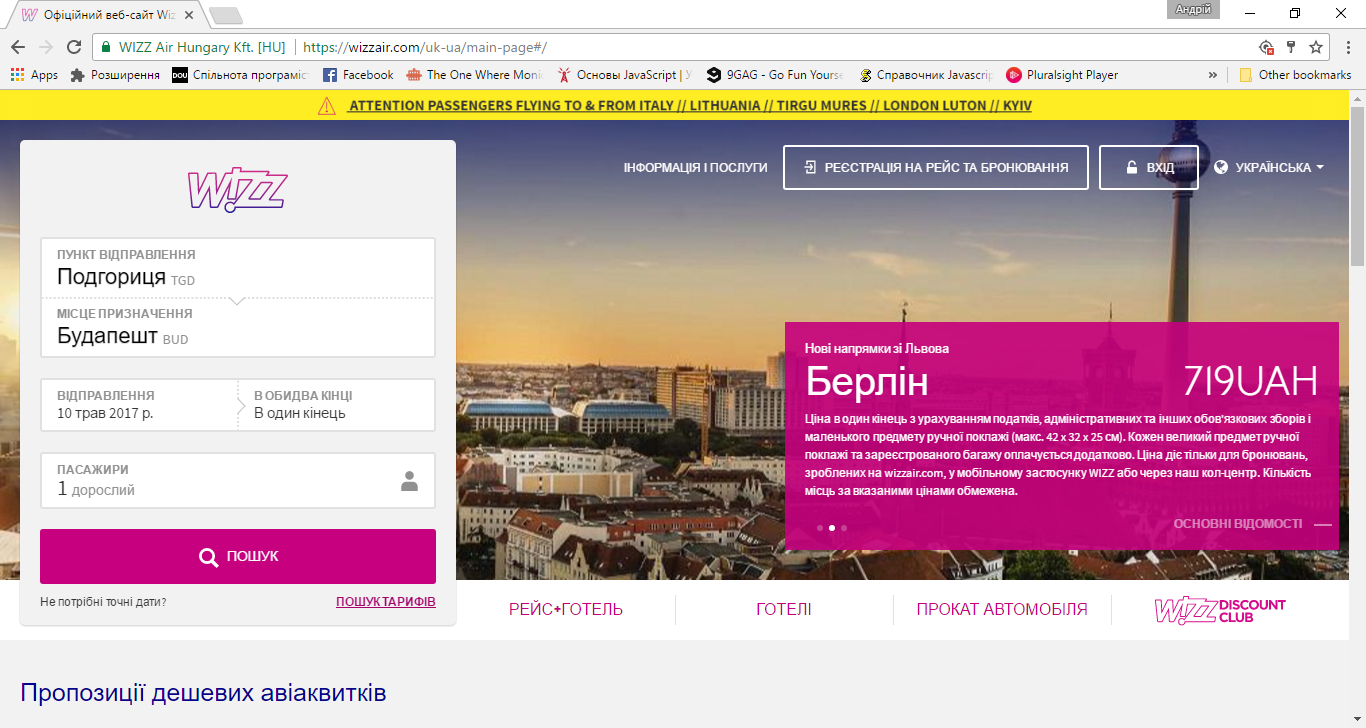


Рис. 1.1 Головна сторінка сервісу WizzAir



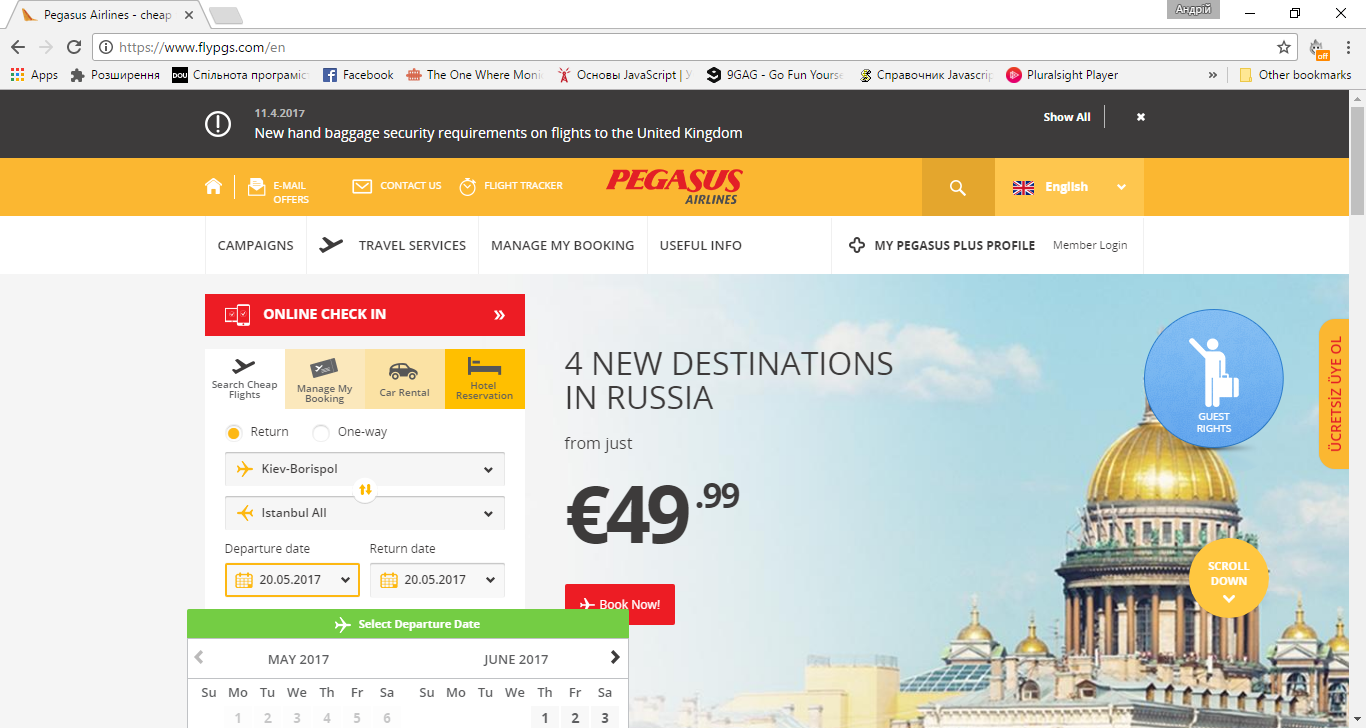
Рис. 1.2. Сторінка результату пошуку рейсів та подальшим замовленням сервісу WizzAir

Переваги:

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

Недоліки:

* Відстежування статусу рейсу

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

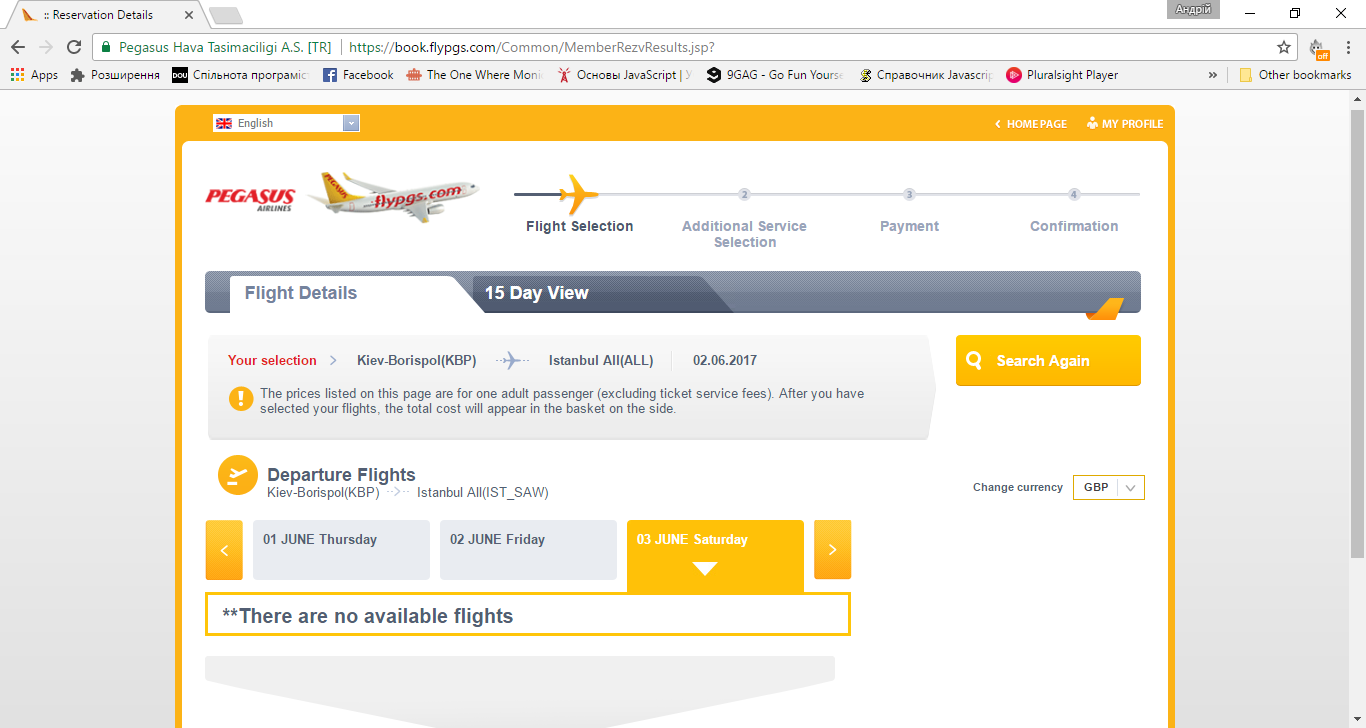


Рис. 1.4. Сторінка результату пошуку рейсів та подальшим замовленням сервісу Pegasus Airlines

Переваги:

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

Недоліки:

* Немає української 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]

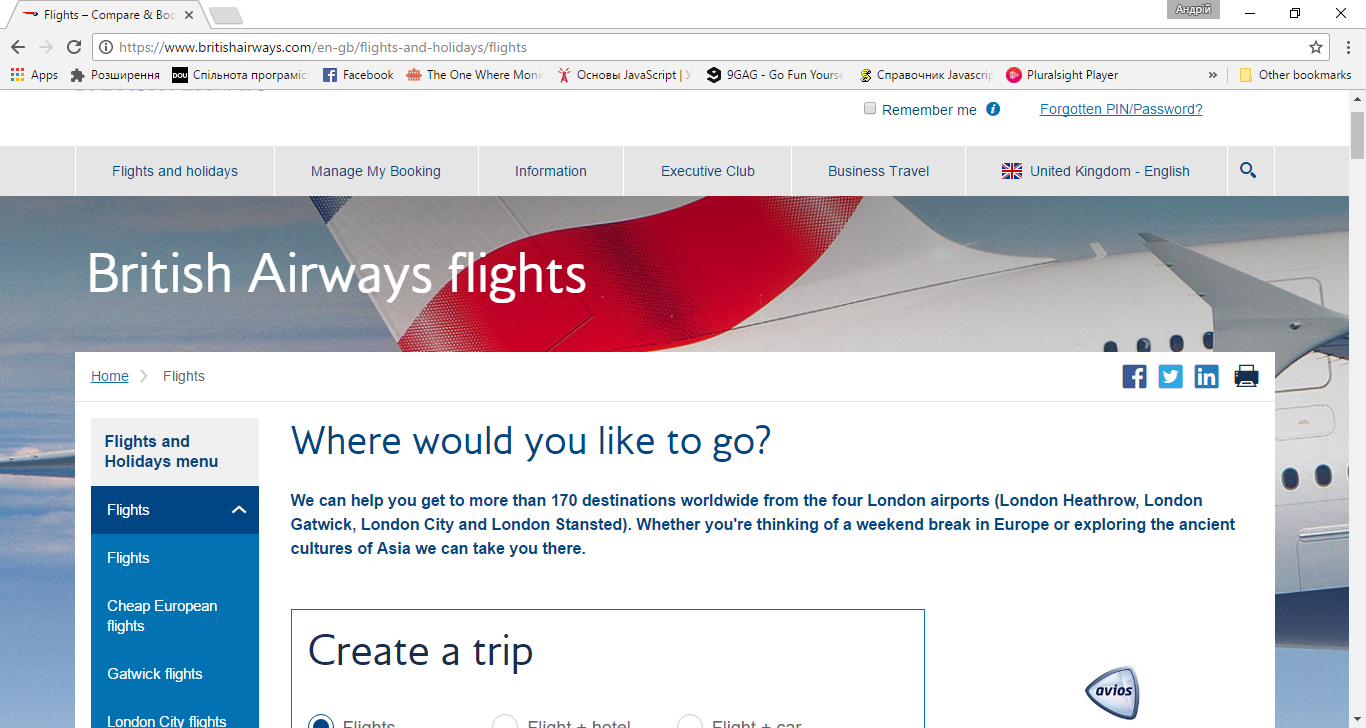


Рис. 1.5. Сторінка пошуку рейсів сервісу British Airways

Переваги:

* Простота і зручність інтерфейсу.
* Пропозиції найдешевших квитків.
* Відстежування 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 |
| Відстежування статусу рейса | - | + | + |
| Наявність інформації про компанію та авіаперевезення | + | + | + |
| Українська локалізація | + | + | - |
| Пропозиції додаткових послуг | + | + | + |
| Пропозиції 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, оскільки 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 квитків на вибраний рейс.
* Відстежування 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 об'єктно-орієнтованою мовою 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 - розподілена структура програми, яка розділяє завдання або робочі навантаження між постачальниками ресурсу або послуг, які називаються серверами та послуг які виконують запити, званих клієнтів. Часто клієнти та 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 існує 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 створюють 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.

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 - 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[9], мова C#[8] та фреймворк ASP.NET Web API[6]. Також для маніпуляції 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[10-13].

 .NET Framework є основою програмного забезпечення, розроблена корпорацією Майкрософт, яка працює в основному на Microsoft Windows. Вона містить 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 об’єктно-орієнтованою мовою програмування яка охоплює строгу типізацію, 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, не типізована 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 містить 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 бібліотеки, це два різні мови і сильно відрізняються за своєю конструкцією.

## 2.4. Специфікація вимог до програмного продукту

* + 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 квитків на вибраний рейс.
* Редагування даних в особистому кабінеті.
* Відстежування статусу рейсу.
* Реєстрація та авторизація в системі.

Діаграму варіантів використання системи та діаграму діяльності наведено в Додатку А та Додатку Б відповідно.

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

Таблиця 2.1

Класи користувачів та їх можливості

|  |  |
| --- | --- |
| 1. Зареєстрований користувач – це користувач, який є зареєстрованим в системі. | * Пошук рейсів. * Перегляд інформації про рейс. * Замовлення або бронювання квитків. * Відстежування 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. * Відстежування статусу рейсів. * Зміна персональної інформації. * Відстежування 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. * Відстежування статусу рейсів. |

* + - 1. **Середовище функціонування**

Програмний продукт потребує 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. **Перегляд 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 користувач вибирає бажаний рейс натиснувши 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 натиснувши 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 натиснувши 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 натиснувши на посилання «Особистий кабінет»
2. На сторінці особистого кабінету натиснути на кнопку «Налаштування».
3. З’являється додаткове вікно, де користувач змінює дані та натискає на кнопку «Змінити».
4. З’являється повідомлення про те, що дані змінено успішно.
   * + - 1. Функціональні вимоги

REQ-1. Користувач повинен бути авторизований.

REQ-2. Введені дані мають бути коректного 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 натискає кнопку «Ок».
5. Користувач натискає на кнопку «Продовжити».
6. З’являється повторна інформація про замовлення та в залежності від вибраного методу реєстрації(Онлайн/Аеропорт) з’являється 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. З’являється 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 переходить 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 переходить 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 переходить 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. **Перегляд 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 користувач вибирає бажане замовлення натиснувши 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 WebApi 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-технології(HTML5, CSS3, JavaScript), а також фреймворк AngularJS та середовище розробки Web Storm.

Також було сформовано специфікацію 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

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.

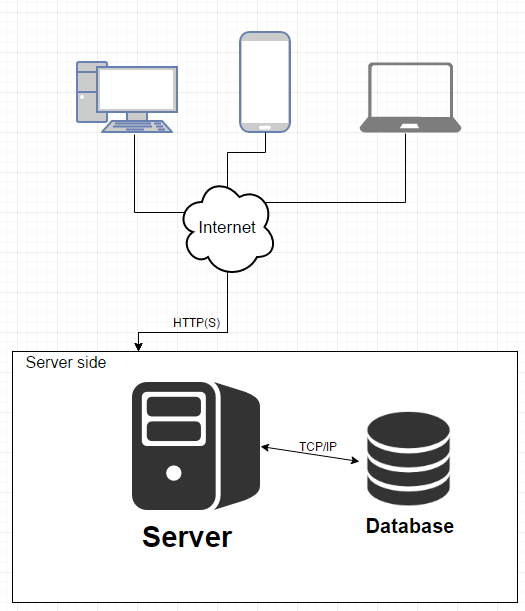


Рис. 3.1 Приклад клієнт серверної архітектури

Серверна 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:

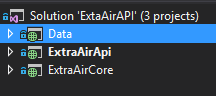


Рис. 3.2 Список пакетів на серверній частині

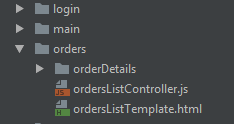
Data – у цьому проекті буде зосереджено роботу на маніпулювання з базою даних, усі запити до бази даних будуть відбуватись у цьому проекті.

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

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

Цей підхід дозволяє зберегти код чистішим та зрозумілішим при зростанню основного проекту, 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: ім’я, прізвище, email, пароль, дата народження, видалений, адреса та роль.



Рис. 3.5. Клас User

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



Рис. 3.6. Клас Client

Address – клас в якому міститься інформація про адресу(користувача та аеропорту) 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: ім’я, адресу та посилання на проміжний клас tourToAiprort(зв’язок багато до багатьох).



Рис. 3.8. Клас Airport

TourToAirport – проміжний клас який об’єднує Tour та Airpor та містить: час відправлення, 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 відправлення, 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 комфорту



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.

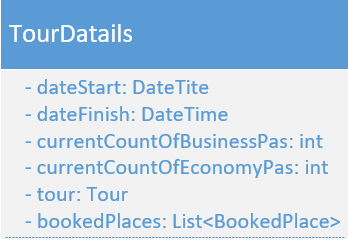


Рис. 3.13. Клас TourDetails

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

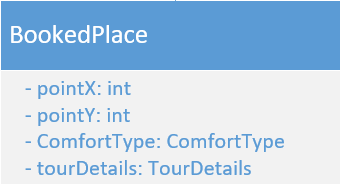


Рис. 3.14. Клас BookedPlace

## 3.4. Проектування бази даних

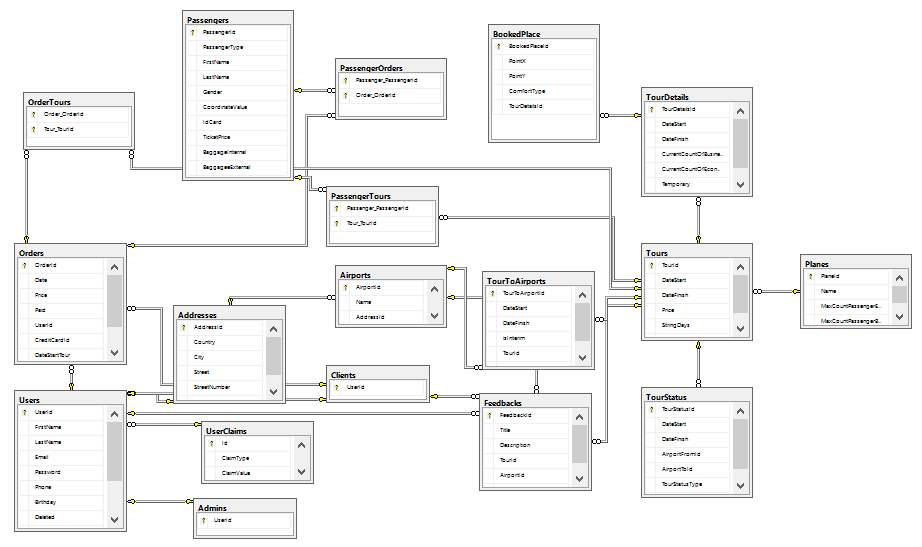


Рис. 3.15. Схема бази даних

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

## 3.5. Прототипування інтерфейсу користувача

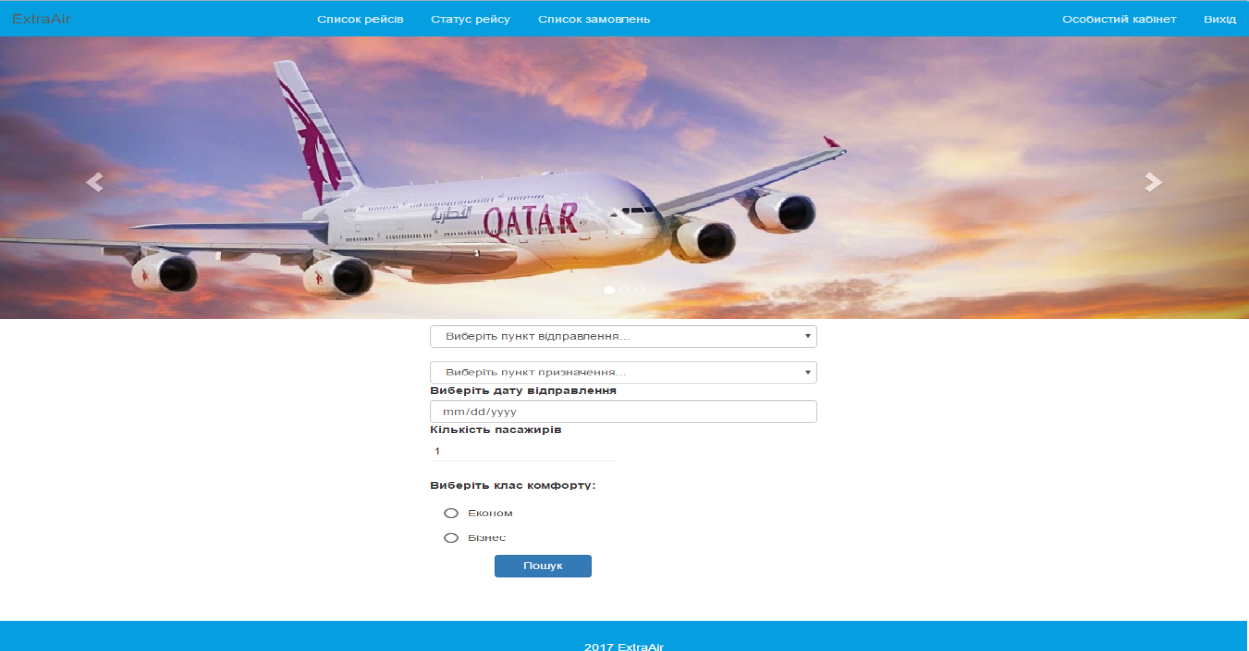


Рис. 3.16. Домашня сторінка додатку

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

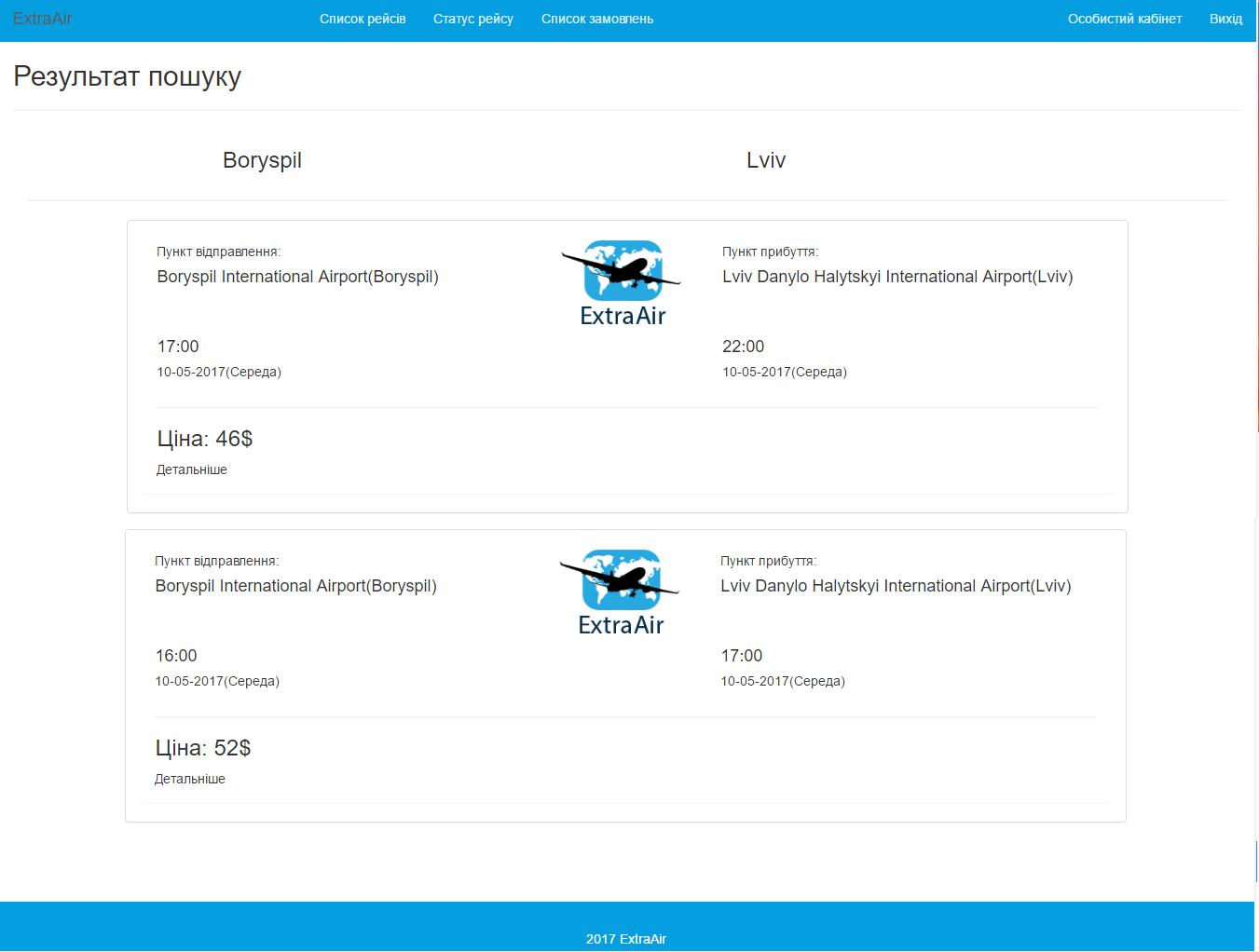
****

Рис. 3.17. Сторінка результату пошуку рейсів

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

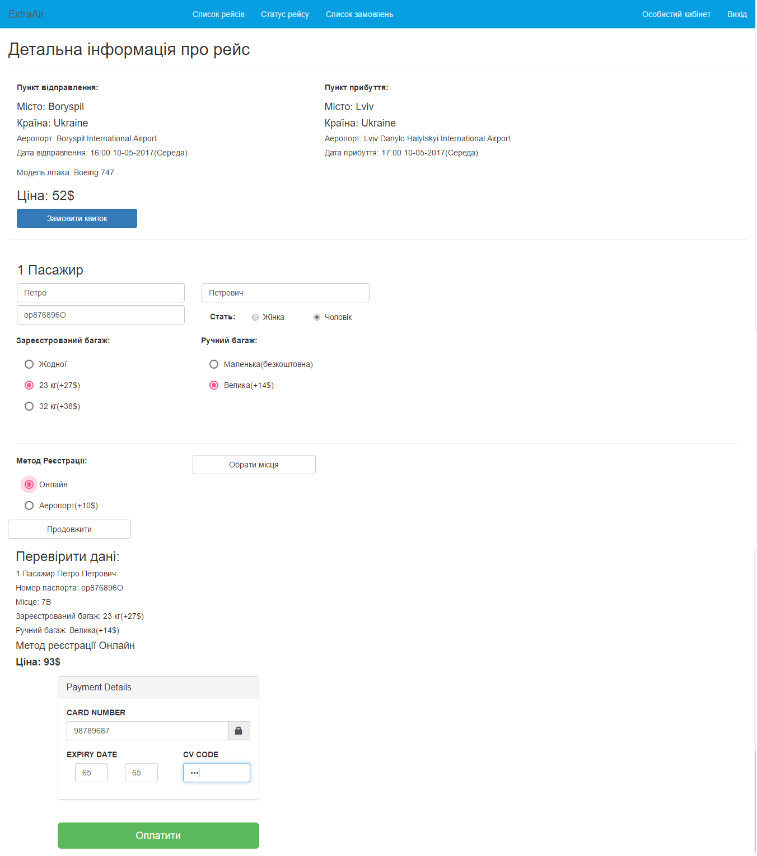


Рис. 3.18 Сторінка замовлення квитка

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

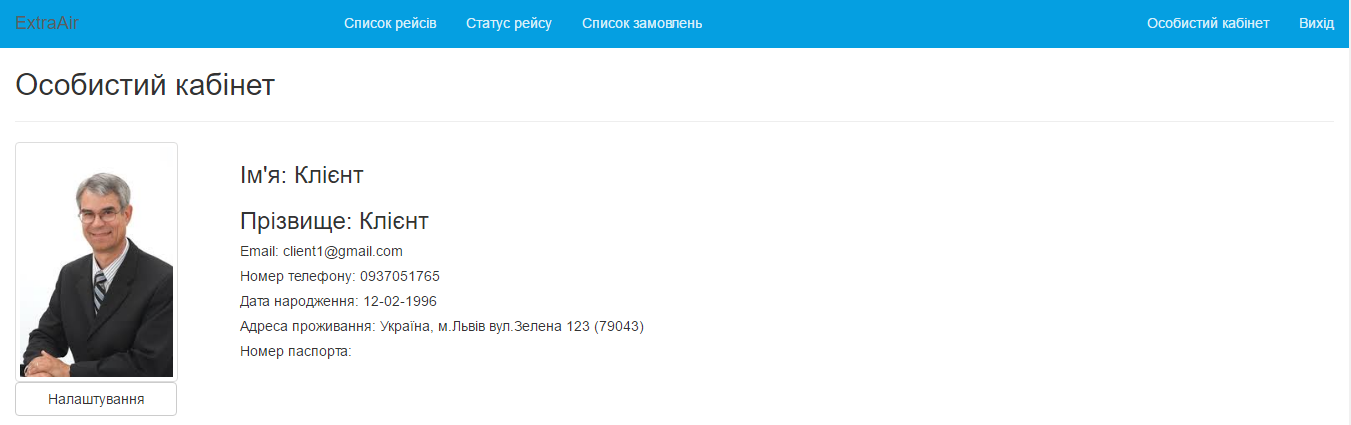


Рис. 3.19 Сторінка особистого кабінету.

**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.

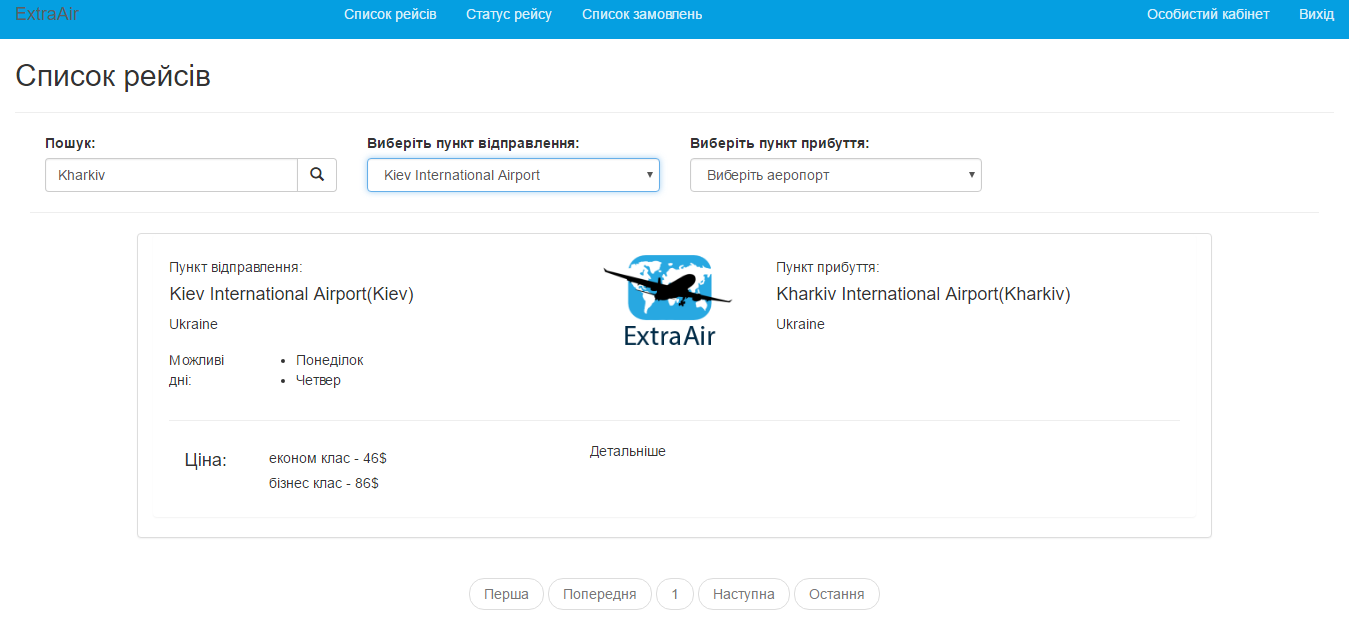


Рис. 3.21. Сторінка «Список рейсів»

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

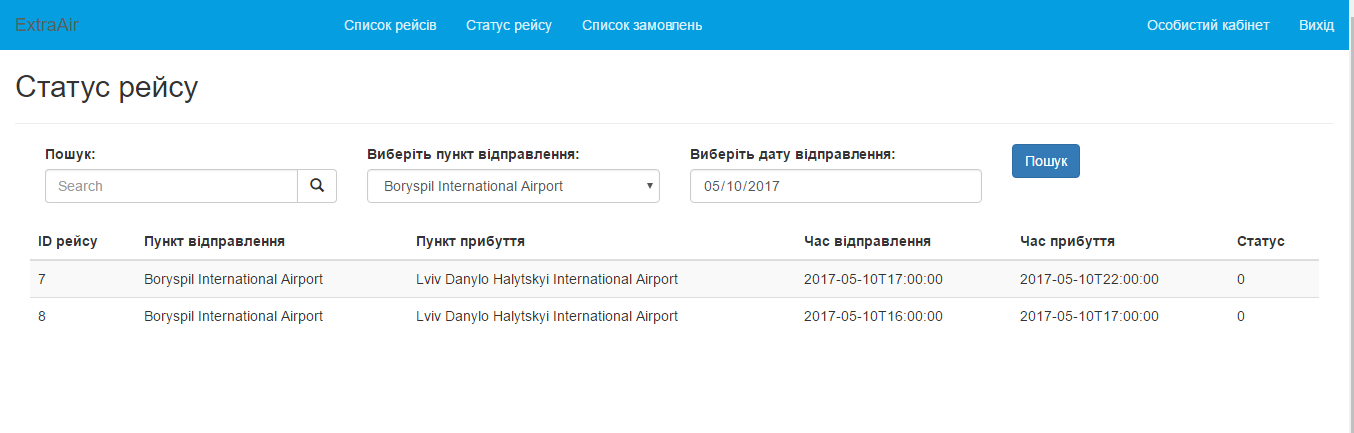


Рис. 3.22. Сторінка «Статус рейсу»

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

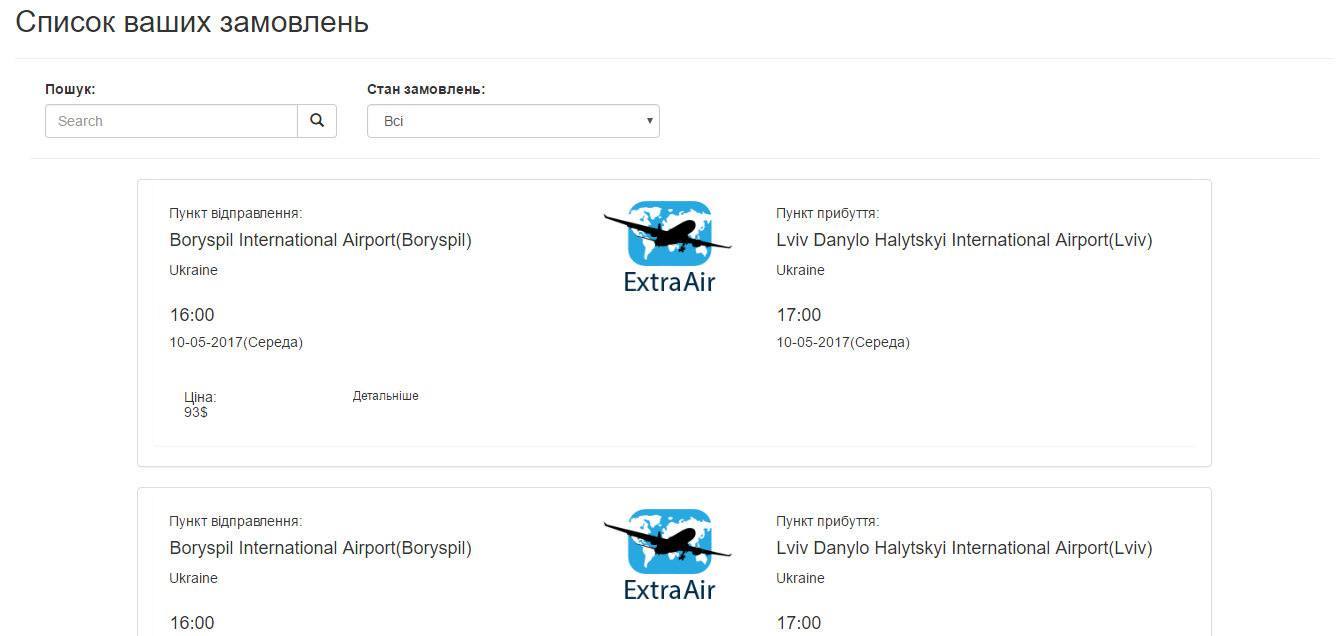


Рис. 3.23. Сторінка «Список ваших замовлень»

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

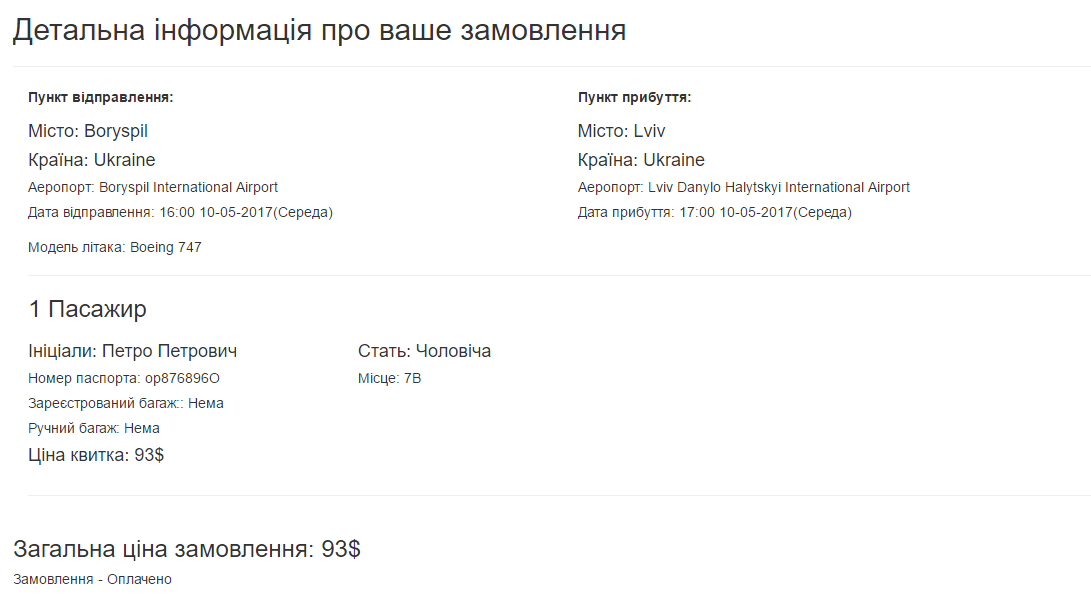


Рис. 3.24. Сторінка детальної інформації про конкретне замовлення

**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 проекту 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. Оскільки 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

Відповідно до вимог, серверна частина буде використовувати .Net технології, а саме Web Api, та ORM технологію Entity Framework для маніпуляції 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 моделей класів

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

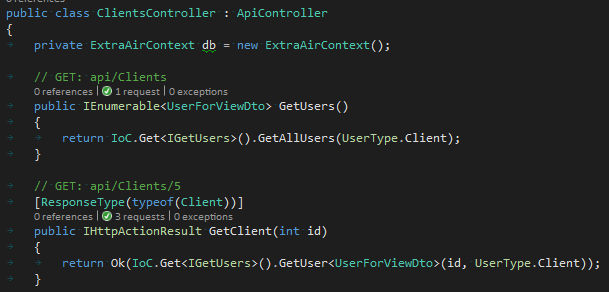


Рис. 4.2. Приклад контролера в Web Api

### 4.1.2. Реалізація клієнтської частини

Як було описано у 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.

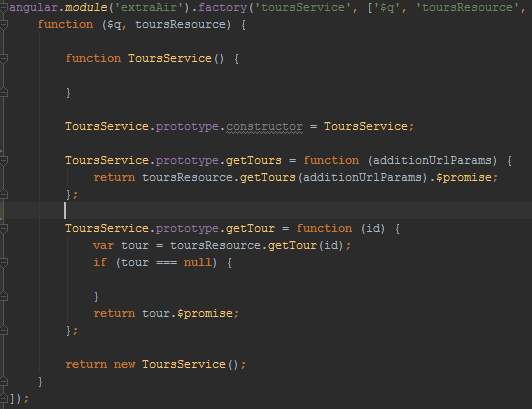


Рис. 4.3. Приклад сервісу в AngularJS

## 4.2. Опис роботи з програмним продуктом

Спочатку користувачу необхідно перейти 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.).

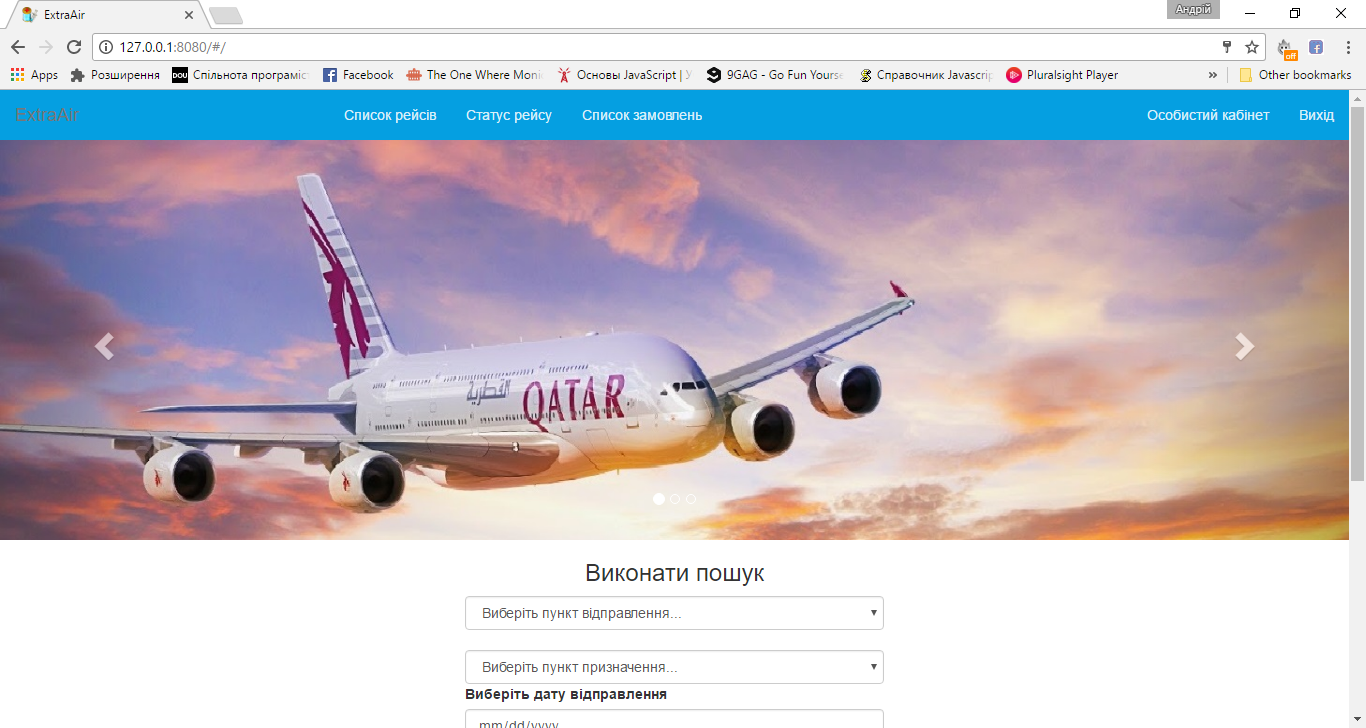


Рис. 4.4. Головна сторінка



Рис. 4.5. Форма для пошуку рейсів

Для того щоб отримати всі повноцінні права, користувачу потрібно створити свій акаунт, та увійти в систему, для цього користувач вибирає посилання «Реєстрація» праворуч 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 змінити її(включаючи нове фото) або поміняти пароль(рис 4.8).

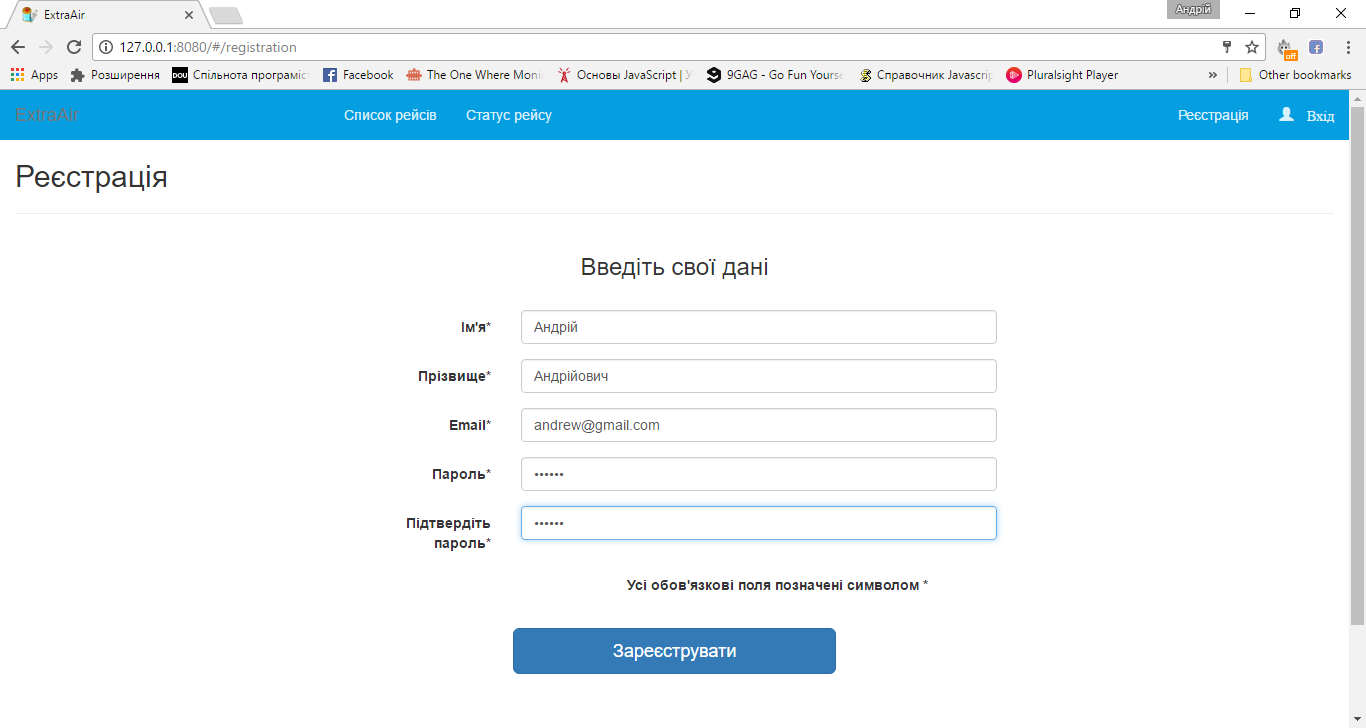


Рис. 4.6. Сторінка реєстрації

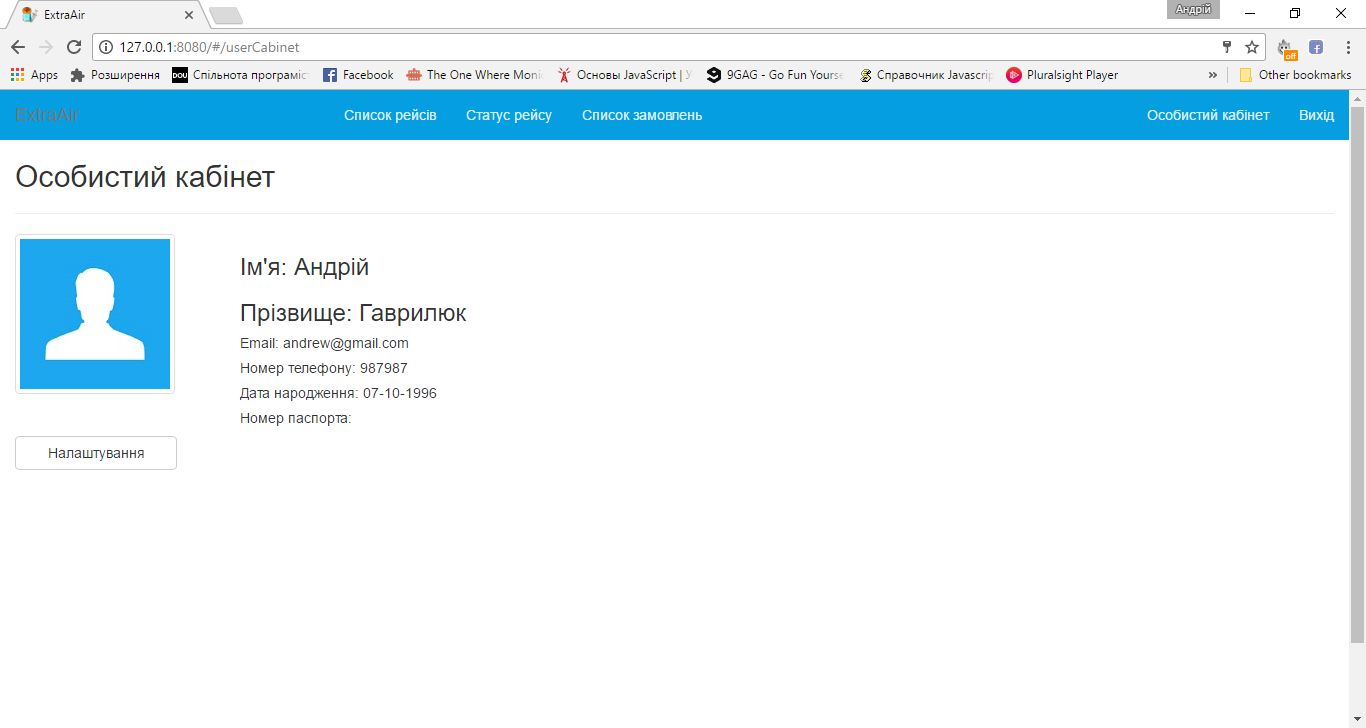


Рис. 4.7. Сторінка особистого кабінету

Отже, щоб замовити квиток, користувач повинен увійти в систему перейти на головну сторінку, 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.).

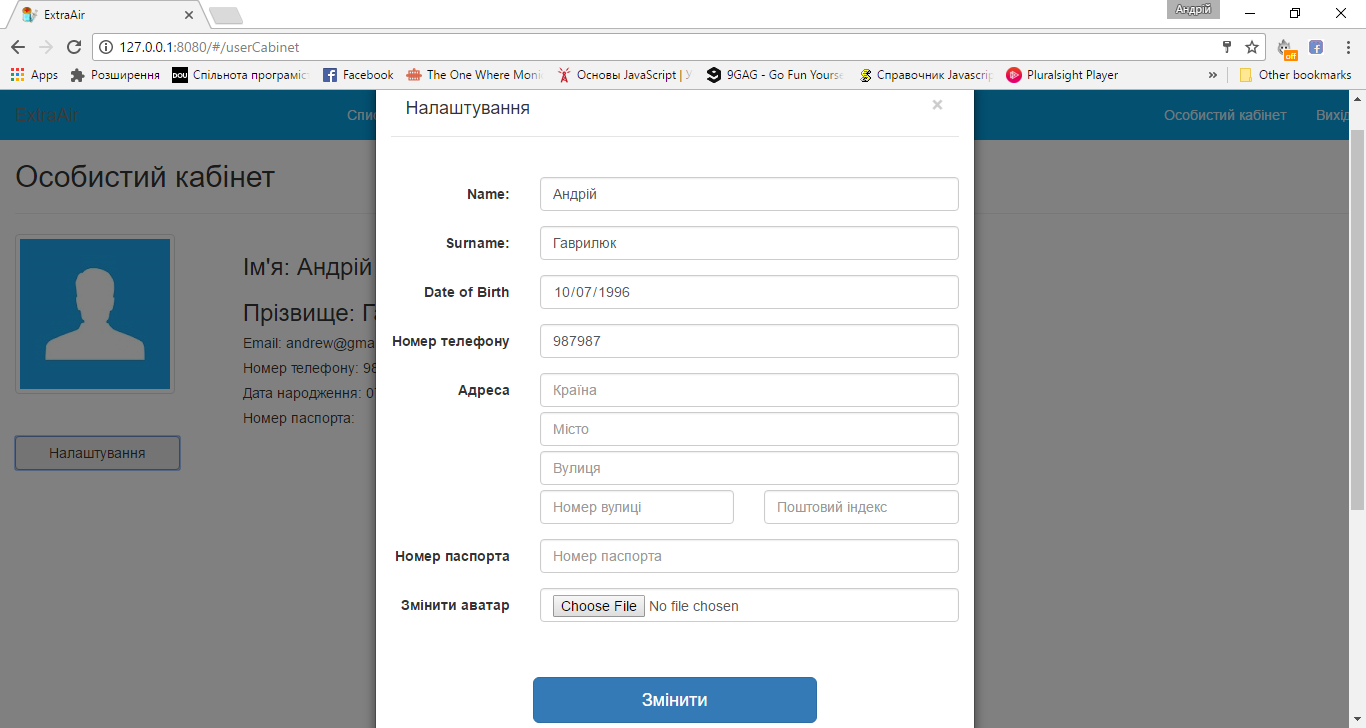


Рис. 4.8. Вікно зміни налаштувань

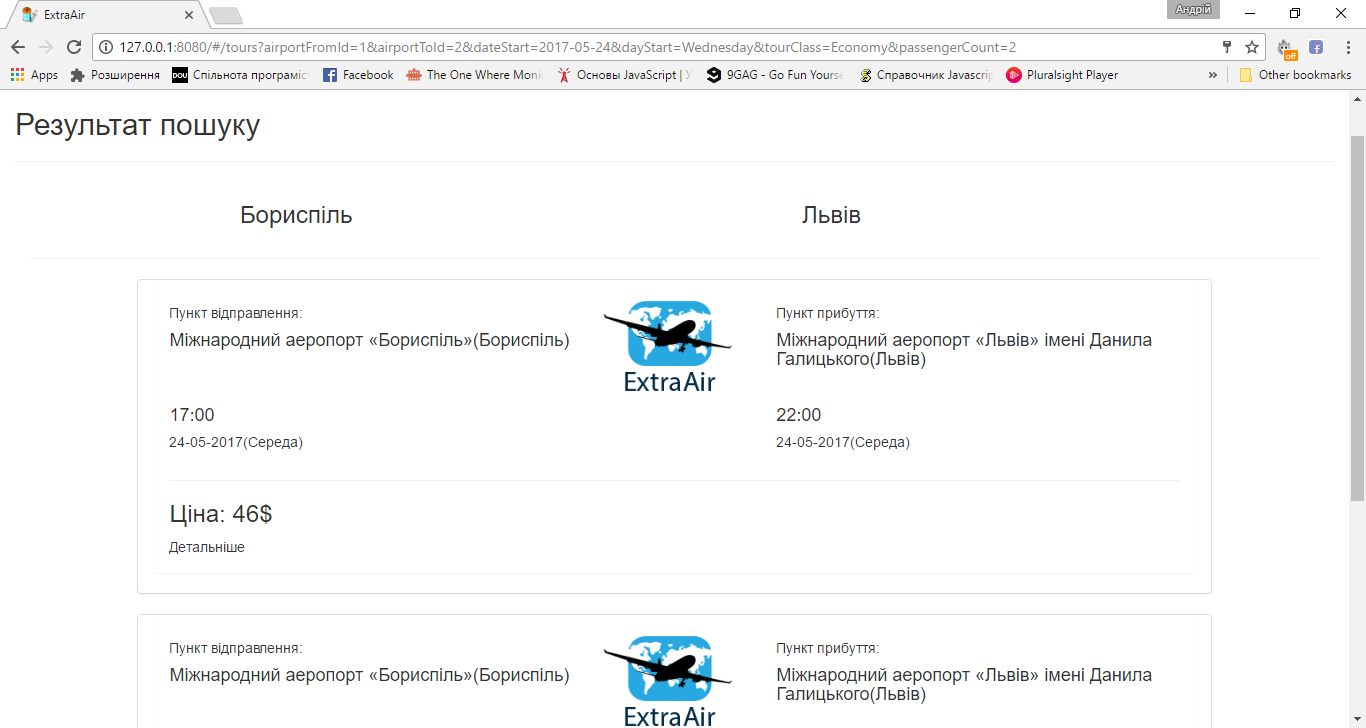


Рис. 4.9. Сторінка результату пошуку

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 натискає 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 попередньо 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.

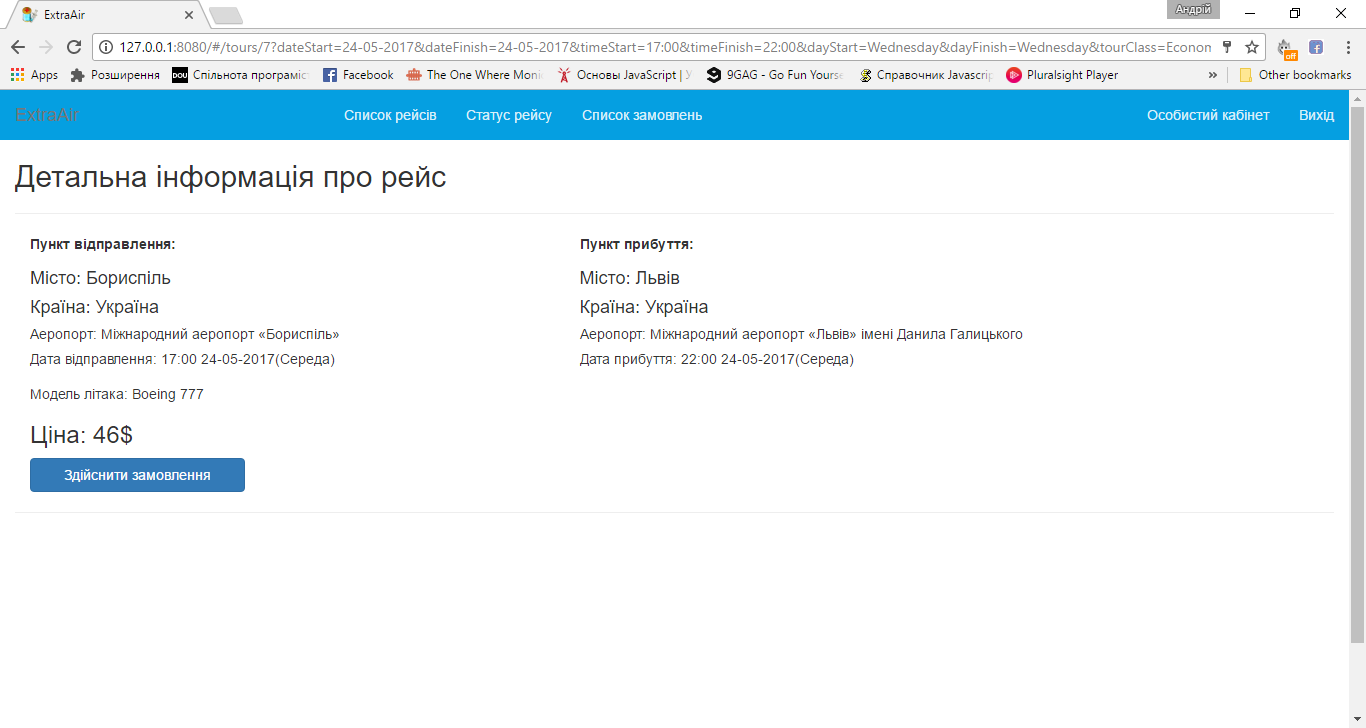


Рис. 4.10. Сторінка детальної інформації про рейс

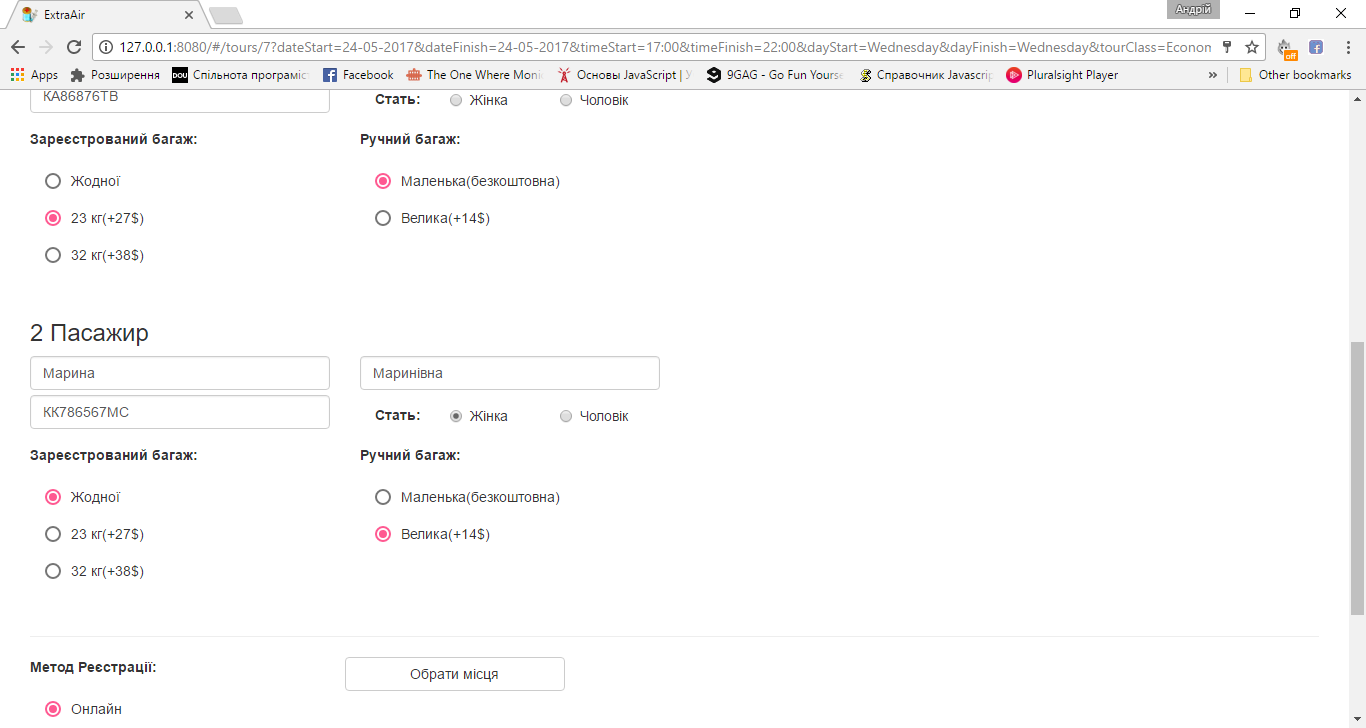


Рис. 4.11. Процес замовлення квитків

Після цього з’являється додаткова перевірка інформації та якщо метод реєстрації онлайн з’являється форма для заповнення кредитної картки(рис. 4.13.), якщо метод реєстрації в аеропорті то кнопка «Забронювати». Натиснувши на кнопку «Оплати»/ «Забронювати» з’явиться повідомлення про те що операцію здійснено успішно/не успішно і ваше замовлення з’явиться у списку замовлень яке можна переглянути на сторінці детальної інформації про замовлення.

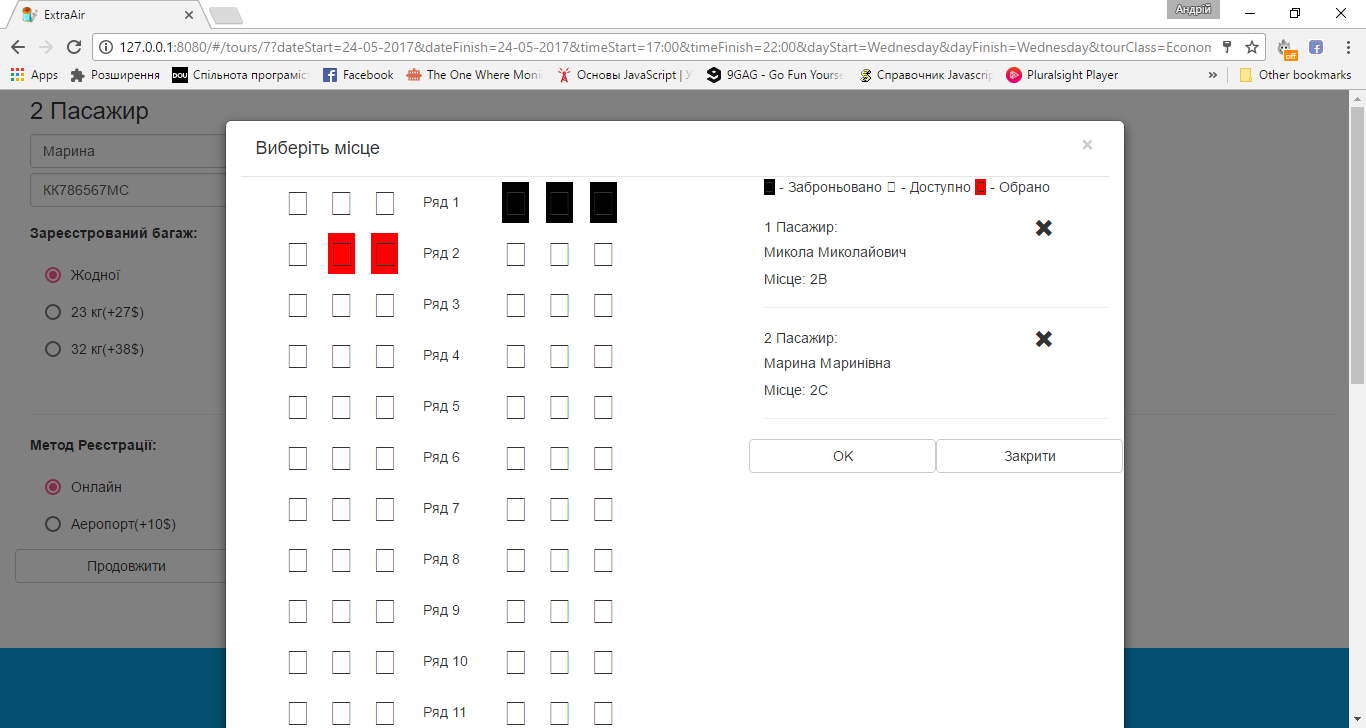


Рис. 4.12. Вікно для вибору місць на рейс

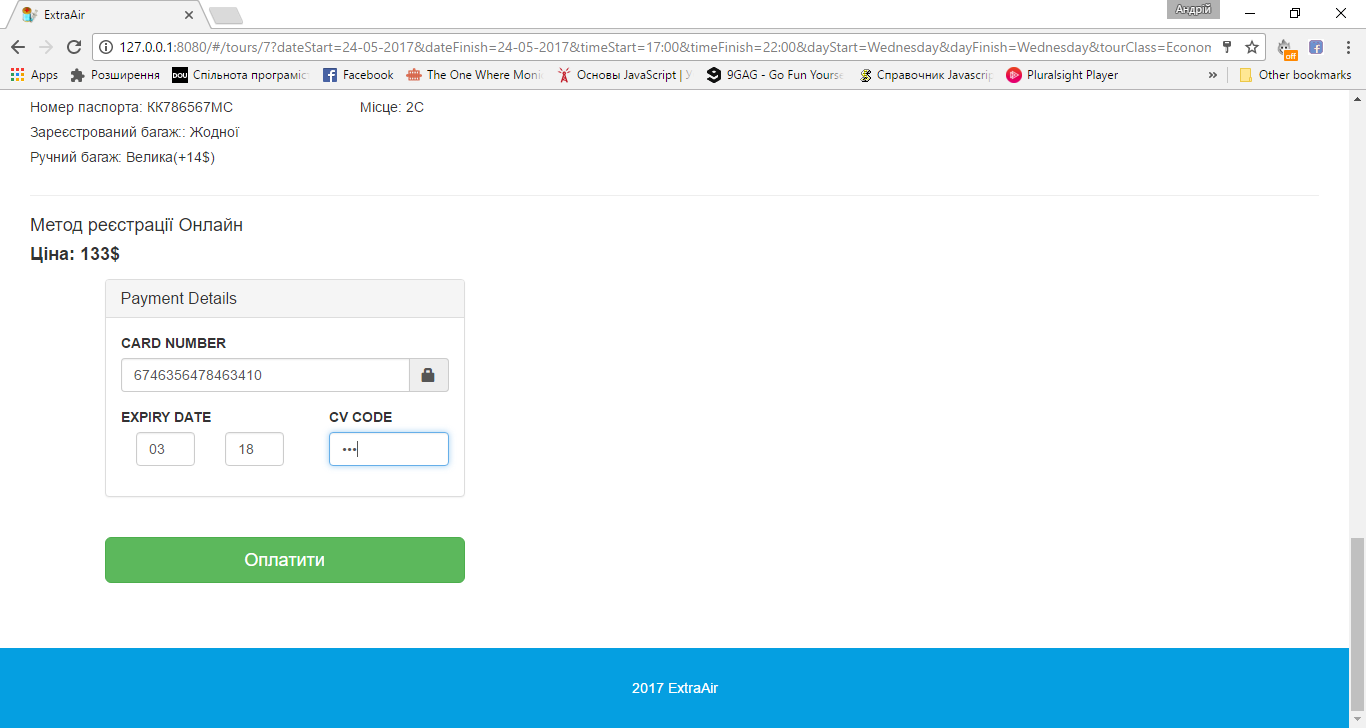


Рис. 4.13. Форма для заповнення кредитної картки

Список рейсів перейшовши на цю сторінку, натиснувши на відповідне посилання у панелі навігації з’являється список усіх можливих рейсів(рис. 4.14.). У цьому списку присутня pagination, щоб обмежити кількість ітерацій на сторінці, а також можна застосувати фільтри та пошук(рис. 4.15.).

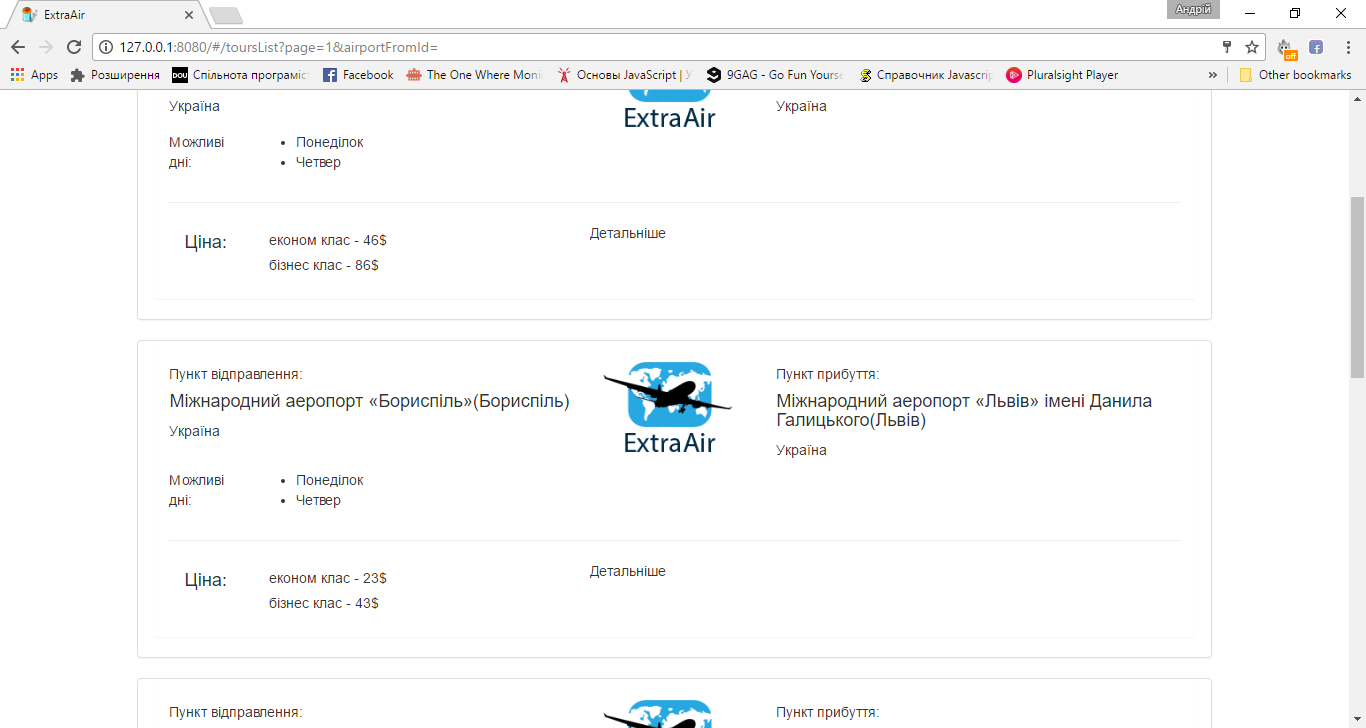


Рис. 4.14. Сторінка списку усіх можливих рейсів

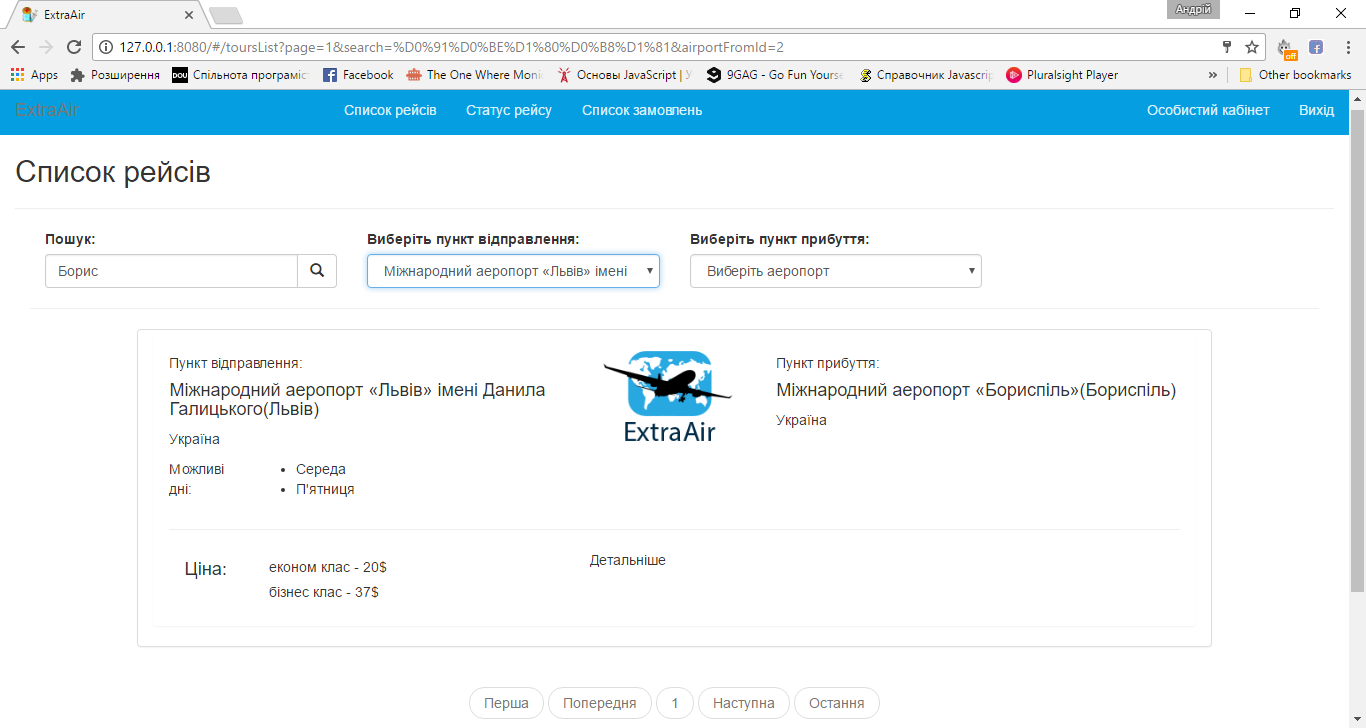


Рис. 4.15. Список усіх можливих рейсів із застосуванням фільтрів та пошуку

Можна замітити що після входу в систему з’явилось нове посилання в верхній навігаційній панелі «Список замовлень»(рис. 4.16.), перейшовши на цю сторінку користувач може переглянути список усіх замовлень які він 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 заданими параметрами.

Окремо потрібно розказати про роль адміністратора, а саме відрізнятиметься сторінка особистого кабінету(рис. 4.17.). Якщо користувач зайшов в систему з правами адміністратора, він має право переглядати та редагувати інформацію про аеропорти, рейси, статус рейсу, користувачів.



Рис. 4.16. Список замовлень

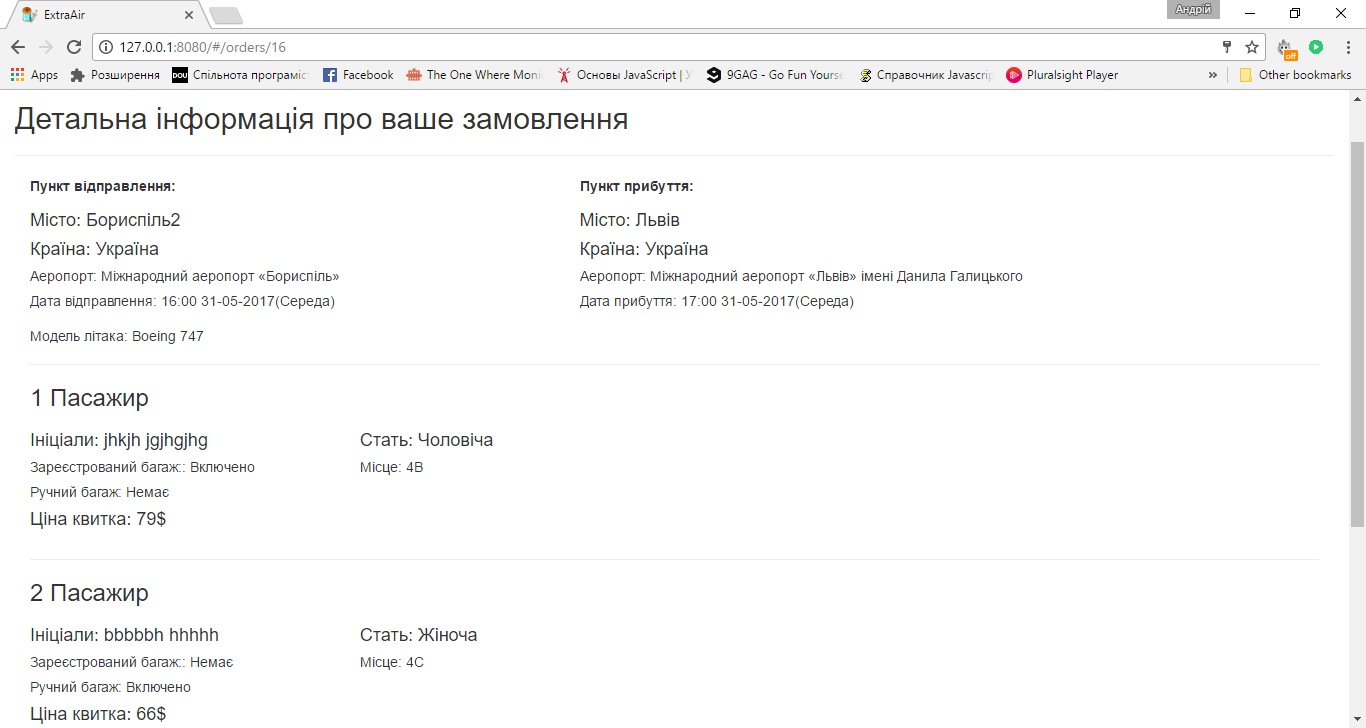


Рис. 4.17. Детальна інформація про замовлення

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.**

Тестування програмного забезпечення – це перевірка роботи програмного продукту за вказаними вимогами, є важливим етапом у розробці програмного забезпечення. Отже, в цьому розділі проведено тестування програмного забезпечення та наведені результати тестування веб сервісу для моніторингу та планування пасажирських авіарейсів для авіакомпанії. Проведено функціональне тестування веб сервісу.

**4.3.2. Функціональне тестування**

Функціональне тестування проводиться для тестування функціональності та поведінки програмного забезпечення відповідно до поставлених вимог.

Під час функціонального тестування було реалізовано 8 тестових випадків, які покривають увесь функціонал проекту.

Список тестових випадків:

* Пошук авіарейсів
* Перегляд детальної інформації про рейс
* Реєстрація в системі
* Замовлення квитків
* Перегляд списку авіарейсів, пошук та фільтрування
* Статус рейсу за вибраними параметрами
* Перегляд списку замовлень та детальної інформації про замовлення
* Редагування даних в особистому кабінеті

Звіт по функціональному тестуванню наведено в додатку Б.

Таблиця 4.1

Розподіл варіантів використання для функціонального тестування

|  |  |
| --- | --- |
| **Варіанти використання** | **Тестові дані** |
| Пошук авіарейсів | 4 |
| Перегляд детальної інформації про рейс | 4 |
| Реєстрація в системі | 5 |
| Замовлення квитків | 12 |
| Перегляд списку авіарейсів, пошук та фільтрування | 3 |
| Статус рейсу за вибраними параметрами | 2 |
| Перегляд списку замовлень та детальної інформації про замовлення | 2 |
| Редагування даних в особистому кабінеті | 8 |
| **Загалом** | **40** |

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

## 4.4. Висновок

У цьому розділі було описано реалізацію та тестування програмного забезпечення для моніторингу та планування пасажирських авіаперевезень. Було детально описано розробку проекту на клієнтській та серверній частих, із наведеними прикладами коду та графічними знімками. Здійснено опис роботи з програмним продуктом, та проведено функціональне тестування, яке було успішно пройдене.

# 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 для конкретної авіакомпанії 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 програмного забезпечення є доцільною, з економічної точки зору. Є програми аналоги (авіакомпанії), 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 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 авіа транспортом є його висока швидкість пересування. Всього за кілька годин можна перетнути тисячі кілометрів і опинитися в потрібному вам місці.

— Подорож літаком є набагато цікавішою ніж поїздом чи автобусом. Адже не потрібно витрачати декілька днів на подорож.

— Літаки мають високий сервіс в обслуговуванні пасажирів. Під час польоту пасажиру надається їжа та напої, а також пропонуються всілякі додаткові послуги, що дозволяють скоротити час, наприклад, подивитися фільм.

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 найближчих 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:

* Відстежування 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.
* Відстежування статусу рейса.
* Величезна кількість інформації про компанію та авіаперевезення.

Недоліки:

* Немає української локалізації

В результаті огляду програмного забезпеченнядля моніторингу та планування пасажирських авіаперевезень для конкретної авіакомпанії 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 |
| Рівень техніки та технологій | 4 | 0,04 | 0,16 |
| Продовження табл. 5.1 | | | |
| 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 альтернатив

**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.

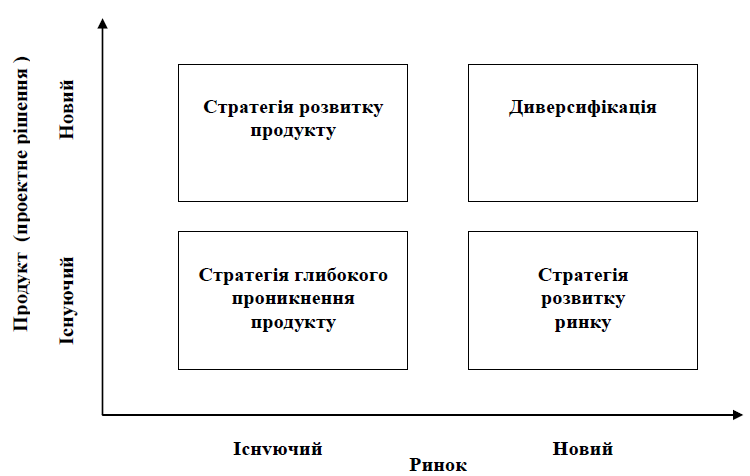


Рис. 5.2 Стратегічні альтернативи група №2

**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 = 767,12 грн.

А2 = (16000/2)/365\*15 = 328,76 грн.

А3 = (16000/2)/365\*35 = 767,12 грн.

Загальна сума амортизації: 1863 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 | 3 | (16000, 16000, 16000) | 1863 |
| Продовження табл. 5.2 | | | | |
| Wifi роутер | TP-LINK TL-WR720N | 1 | 400 | 400 |
| Разом: | - | - | - | 2263 |

*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

|  |  |
| --- | --- |
| **Статті витрат** | **Сума, грн.** |
| *Змінні загальновиробничі витрати, у т.ч.:* |  |
| - заробітна плата допоміжного персоналу; | 3200 |
| - витрати на МШП; | 150 |
| - витрати на електроенергію | 500 |
| - інші змінні витрати; | 100 |
| Разом змінних витрат: | 3950 |
| Продовження табл. 5.4 | |
| *Постійнi загальновиробничі витрати, у т.ч.:* |  |
| - комунальні послуги; | 950 |
| - витрати на оренду; | 5000 |
| - витрати на ремонт; | 500 |
| - інші постiйні витрати; | 200 |
| Разом постiйних витрат: | 5700 |
| *Разом загальновиробничих витрат:* | 11850 |

*Таблиця 5.5*

**Бюджет адміністративних витрат та витрат на збут**

|  |  |
| --- | --- |
| **Статті витрат** | **Сума, грн.** |
| 1 | 2 |
| *Адміністративні витрати, у т.ч.:* |  |
| - заробітна плата адміністративного персоналу; | 3200 |
| - витрати на МШП; | 150 |
| - витрати на сплату податків i зборів; | 1800 |
| - знос адміністративного обладнання; | 350 |
| Разом адміністративних витрат: | 5500 |
| *Витрати на збут, у т.ч.:* |  |
| - заробітна плата менеджерів зі збуту; | 7000 |
| - витрати на гарантійний ремонт; | 0 |
| - витрати на гарантійне обслуговування; | 0 |
| - витрати на налагодження i експлуатацію; | 1500 |
| - витрати на паливо-мастильні матеріали; | 0 |
| - витрати на рекламу; | 9000 |
| Разом витрат на збут: | 17500 |

*Таблиця 5.6*

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Статті витрат** | **Одиниці виміру** | **Фактична кількість, шт.** | **Ціна одиниці, грн.** | **Разом, грн.** |
| Сировина i матеріали | шт | 4 | (16000,16000 16000,400) | 2263 |
| Паливо та електроенергія на технологічні цілі | кВт | 4545 | 0.99 | 450 |
| Основна заробітна плата | грн | 3 | (2500,35000, 3500) | 41000 |
| Додаткова заробiтна плата | грн | - | - | - |
| Відрахування на соцiальне страхування | грн | - | - | 9020 |
| Загальновиробничі витрати, у т.ч.: |  | | | |
| - змінні; | грн | - | - | 3950 |
| - постійні; | грн | - | - | 5700 |
| *Разом виробничих витрат:* | грн | - | - | **62383** |
| Адміністративні витрати | грн | - | - | 5500 |
| Витрати на збут | грн | - | - | 17500 |
| Інші операційні витрати | грн | - | - | 0 |
| *Разом виробничих i операційних витрат:* | грн | - | - | **85383** |

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 = 85383 \* 0.26 + 85383 = 107582,52 грн.

Ціна з ПДВ = 107582,52 \* 1,2 = 129099,09 грн.

Отже, вартість програмного забезпечення становить 129099,09 грн.

*Таблиця 5.7*

**Бюджет фінансових результатів**

|  |  |
| --- | --- |
| **Показники** | **Сума, грн.** |
| 1 | 2 |
| Дохід від реалізації продукції (1 шт) | 129099,09 |
| Податок на додану вартість (20%) | 21516,57 |
| Чистий дохід від реалізації продукції | 107582,52 |
| Собівартість реалізованої продукції | 62383 |
| Валовий прибуток | 45199,52 |
| *Операційні витрати:* | |
| - адміністративні витрати | 5500 |
| - витрати на збут | 17500 |
| - інші операційні витрати; | 0 |
| Фінансовий результат від операційної діяльності | 22199,52 |
| Податок на прибуток (18%) | 3995,91 |
| **Чистий прибуток (збиток)** | **18203,61** |

## 5.6. Остаточний вибір стратегії

Схарактеризувавши програмне забезпечення для моніторингу та планування пасажирських авіаперевезень для конкретної компанії, потрібно провести економічну оцінку розробки.

Оцінивши та проаналізувавши фактори можна зробити висновок що основними зовнішніми факторами є споживачі, науково-технічний прогрес та рівень техніки та технологій. Основними внутрішніми факторами постають технології та цілі.

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, найефективнішою стратегію для даного продукту буде стратегія глибокого проникнення на ринок за рахунок агресивних маркетингових дій стосовно конкурентів.

Провівши розрахунок витрат, пов’язаних з виготовленням та реалізацією продукту, було визначено собівартість програмного продукту у розмірі **85383** грн., основною статтею витрат становить заробітна плата інженерного технічного персоналу, оскільки розробка інформаційного продукту потребує значних інтелектуальних затрат у програміста, що відповідно винагороджується.

Провівши економічну оцінку доцільності проектування та розроблення програмного продукту, а саме веб сервісу для пошуку, моніторингу авіаквитків, відстежування їхнього поточного статусу було визначено, що чистий прибуток від реалізації однієї продукції становитиме **18203,61** грн.

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 та описав предметну область програмного забезпечення дослідив існуючі програми аналоги, встановив задачі та цілі для розробки програмного забезпечення, здійснив проектування проекту, розробив та здійснив тестування програмного продукту для моніторингу та планування авіарейсів для авіакомпанії.

Дослідивши продукти конкуренти було виявлено декілька недоліків, які було враховано та вирішено при розробці програмного продукту.

Провівши постановку задачі та проектування архітектури, було вирішено розробляти проект у вигляді веб сайту, та як це актуально на сьогодні і для цього потрібен лише веб браузер та з’єднання з інтернетом. Отже, було вирішено використати архітекруту клієнт-сервер. Серверна частина використовує .Net технології, а саме мову C# та фреймворк Web Api. Клієнтська частина написана використовуючи такі технології як: HTML5, CSS3, JavaScript та фреймворк AngularJS.

Реалізовано функції пошуку авіарейсів, перегляд детальної інформації про авіарейс, бронювання або замовлення квитків на вибраний рейс, відстежування статусу рейсу, реєстрація та авторизація в системі.

Проведено функціональне тестування веб додатку. Для цього було здійснено GUI тестування.

# список Літератури

1. Плюси та мінуси подорожі на літаку [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. WebApi [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/>

# 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 | | | |

# Додаток Б. Діаграма діяльності

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

# Додаток В. Тестові випадки використання програмного забезпечення

*ID RM-1 : Пошук авіарейсів*

*Таблиця В.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 В.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 В.3*

|  |  |  |  |
| --- | --- | --- | --- |
| *Summary:* Перевіряється чи користувач буде доданий у системі | | | |
| *Steps:* | | *Expected Results:* | |
| Продовження табл. В.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 В.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 пункт 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 В.5*

|  |  |  |
| --- | --- | --- |
| *Summary:* Перевіряємо чи відбудеться замовлення квитків | | |
| *Pre-condition:* Виконано тестовий випадок RM-2 та користувач увійшов в систему. | | |
| *Steps:* | *Expected Results:* | |
| Продовження табл. В.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 : Статус рейсу за вибраними параметрами*

*Таблиця В.6*

|  |  |
| --- | --- |
| *Summary:* Перевірка на коректність виведення інформації про статус рейсу, за вибраними параметрами. | |
| *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 В.7*

|  |  |  |
| --- | --- | --- |
| *Summary:* Переглядаємо список з усіма здійсненними замовленнями, а також детальної інформації про вибране замовлення. | | |
| Продовження табл. В.7 | | |
| *Pre-condition: К*ористувач увійшов в систему. | | |
| *Steps:* | *Expected Results:* | |
| 1. Відкриваємо сторінку для перегляду списку замовлень, натиснувши на посилання у верхній панелі навігації «Список замовлень». 2. Вибираємо замовлення та натискаємо 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 В.8*

|  |  |
| --- | --- |
| *Summary:* Перевіряємо чи дані будуть змінені в особистому 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 перезавантаження 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 Д. Інструкція користувача

**Д.1. Компоненти програмного забезпечення**

Веб сервіс складається з двох частин: серверної та клієнтської. Серверна частина становить RESTfull API яке написане на мові C# та використовую MSSQL Server для зберігання даних. Клієнтська є повністю SPA(Single Page Application) та використовує такі технології як: HTML5, CSS3, JavaScript та фреймворк AngularJS.

Для роботи додатку потрібний доступ до інтернету та веб браузер.

**Д.2. Доступ до додатку**

Для використання програмного забезпечення користувачу необхідно ввести відповідну URL адресу в браузері. Для входження в систему з правами адміністратора потрібно використовувати:

* email: admin@gmail.com
* пароль: 123456

**Д.3. Базові функції ПЗ**

Основні можливостями веб сайту:

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

**Д.4. Налаштування ПЗ**

Програмне забезпечення не потребує додаткового налаштування.

**Д.5. Аналіз помилок та можливих проблем**

В разі некоректної роботи програми спочатку потрібно перевірити наявність інтернет з’єднання. Якщо з’єднання наявне то перезавантажити веб додаток, ще раз перейшовши по відповідній URL адресі. Якщо це не допомогло то почистити кеш та кукі.

# Додаток Е. Лістинг коду

***Виведення інформації про список усіх рейсів***

Серверна частина. API контролер

[RoutePrefix("api/tours")]

public class ToursController : ApiController

{

private ExtraAirContext db = new ExtraAirContext();

// GET: api/Tours

[Route("")]

public object GetTours([FromUri]int page = 1, [FromUri]int itemsPerPage = 15, [FromUri]string search = null, [FromUri]int? airportFromId = null, [FromUri]int? airportToId = null,

[FromUri]string day = null)

{

var list = IoC.Get<IGetTours>().GetAllTours();

return IoC.Get<IGetTours>().GetToursWithPaginFiltering(new PaginFilteringHelper

{

Day = day,

AirportToId = airportToId,

AirportFromId = airportFromId,

ItemsPerPage = itemsPerPage,

Page = page,

Search = search

}, list);

}

[HttpGet]

[Route("alltours")]

public object GetAllTours()

{

return IoC.Get<IGetTours>().GetAllTours();

}

[HttpGet]

[Route("bysearch")]

public object GetToursBySearch([FromUri]int airportFromId, [FromUri]int airportToId, [FromUri]string dayStart)

{

return IoC.Get<IGetTours>().GetToursBySearch(new TourSearchHelperDto

{

AirportFormId = airportFromId,

AirportToId = airportToId,

DayStart = dayStart

});

}

// GET: api/Tours/5

[ResponseType(typeof(Tour))]

public IHttpActionResult GetTour(int id)

{

var tour = IoC.Get<IGetTours>().GetTourById(id);

return Ok(tour);

}

}

Одержання інформації з бази даних.

using System.Collections.Generic;

using System.Linq;

using ExtraAirCore.API\_DTOs;

using ExtraAirCore.API\_DTOs.Helper\_DTOs;

using ExtraAirCore.Command.Tour;

using ExtraAirCore.Models.EFContex;

using ExtraAirCore.Models.EFModels;

namespace Data.Api.Tours

{

public class GetTours : IGetTours

{

public IEnumerable<TourDto> GetAllTours()

{

using (var dbContext = new ExtraAirContext())

{

return dbContext.Tours.Select(TourMapHelder).ToList();

}

}

public IEnumerable<TourDto> GetToursBySearch(TourSearchHelperDto searchHelper)

{

using (var dbContext = new ExtraAirContext())

{

return dbContext.Tours.Select(TourMapHelder).ToList().Where(a => (a.AirportFrom != null

&& a.AirportFrom.AirportId == searchHelper.AirportFormId)

&& (a.AirportTo != null && a.AirportTo.AirportId == searchHelper.AirportToId)

&& (a.PossibleDays.Contains(searchHelper.DayStart))).ToList();

}

}

public object GetToursWithPaginFiltering(PaginFilteringHelper pfHelper, IEnumerable<TourDto> list)

{

list = list.OrderBy(x => x.TourId).ToList();

if (pfHelper.AirportFromId != null)

{

list = list.Where(x => x.AirportFrom.AirportId == pfHelper.AirportFromId).ToList();

}

if (pfHelper.AirportToId != null)

{

list = list.Where(x => x.AirportTo.AirportId == pfHelper.AirportToId).ToList();

}

if (pfHelper.Day != null)

{

list = list.Where(x => x.PossibleDays.Contains(pfHelper.Day)).ToList();

}

if (pfHelper.Search != null)

{

pfHelper.Search = pfHelper.Search.ToLower();

list = list.Where(x =>

(x.AirportFrom.City + x.AirportFrom.Country + x.AirportFrom.Name +

x.AirportTo.City + x.AirportTo.Country + x.AirportTo.Name).ToLower().Contains(pfHelper.Search.Replace(" ", "")));

}

var listPaged = list.Skip((pfHelper.Page - 1) \* pfHelper.ItemsPerPage).Take(pfHelper.ItemsPerPage).ToList();

var json = new

{

count = list.Count(),

list = listPaged

};

return json;

}

public static TourDto TourMapHelder(Tour tour)

{

var airports = tour.TourToAirports.ToList();

Airport airportFrom = null;

Airport airportTo = null;

if (airports.Any())

{

airportFrom = airports.Where(x => x.DateStart != null).OrderBy(x => x.DateStart).FirstOrDefault().Airport;

airportTo = airports.Where(x => x.DateFinish != null).OrderByDescending(x => x.DateFinish).FirstOrDefault().Airport;

}

var itnerimAirports = airports.Skip(1).Take(airports.Count - 2);

return new TourDto

{

TourId = tour.TourId,

DateStart = tour.DateStart,

DateFinish = tour.DateFinish,

Price = tour.Price,

Plane = new PlaneDto

{

Name = tour.Plane.Name,

PlaneId = tour.Plane.PlaneId,

MaxCountPassenger = new CountPassengerDto

{

CountOfBusinessPassenger = tour.Plane.MaxCountPassengerBusiness,

CountOfEconomyPassenger = tour.Plane.MaxCountPassengerEconomy

}

},

AirportFrom = airportFrom != null ? new AirportDto

{

AirportId = airportFrom.AirportId,

Name = airportFrom.Name,

City = airportFrom.Address.City,

Country = airportFrom.Address.Country

} : null,

AirportTo = airportTo != null ? new AirportDto

{

AirportId = airportTo.AirportId,

Name = airportTo.Name,

City = airportTo.Address.City,

Country = airportTo.Address.Country

} : null,

ItnerimAirports = itnerimAirports.Any() ? itnerimAirports.Select(x => new InterimTourDto()

{

Airport = new AirportDto()

{

AirportId = x.AirportId,

Name = x.Airport.Name,

Country = x.Airport.Address.Country,

City = x.Airport.Address.City

},

DateStart = x.DateStart,

DateFinish = x.DateFinish

}).ToList() : null,

PossibleDays = !string.IsNullOrEmpty(tour.StringDays) ? tour.StringDays.Split(' ').ToList() : new List<string>()

};

}

public TourDto GetTourById(int id)

{

using (var dbContext = new ExtraAirContext())

{

return dbContext.Tours.Where(x => x.TourId == id).Select(TourMapHelder).FirstOrDefault();

}

}

}

}

Клієнтська частина. Виведення списку рейсів. Контролер.

'use strict';

var app = angular.module('extraAir');

app.controller('toursListController', function ($rootScope, $scope, $location, $window, $filter, getService, paginationArrayService, paginationService, airportsService) {

$scope.isLoading = true;

initPaginationParams();

airportsService.getAirports().then(function(data){

$scope.airports = data;

});

var utils = new CommonUtils();

$scope.$on('$locationChangeStart', function(e) {

$scope.isLoading = true;

initPaginationParams();

// paginationService.ChangeURL($scope.loadList, $scope.tours, $rootScope.preArray, '/toursList', $rootScope.pagingInfo);

});

$scope.tours = [];

$scope.loadList = function(){

$scope.isLoading = true;

var URL = "api/tours?page=" + $rootScope.pagingInfo.page + "&itemsPerPage=" + $rootScope.pagingInfo.itemsPerPage +

"&search=" + $rootScope.pagingInfo.search + "&airportFromId=" + $rootScope.pagingInfo.airportFromId +

"&airportToId=" + $rootScope.pagingInfo.airportToId + "&day=" + $rootScope.pagingInfo.day;

getService.GetObjects(URL).then(function (data) {

$scope.tours = data.data.list;

$scope.AnyElementOfList = data.data.count == 0;

$rootScope.pagingInfo.totalItems = data.data.count;

$scope.pages = Math.ceil(data.data.count / $rootScope.pagingInfo.itemsPerPage);

$scope.paginArray = paginationArrayService.Array($scope.pages, $rootScope.pagingInfo.page);

correctData();

$scope.paginationScope = {

data: !!$rootScope.pagingInfo ? {} : $rootScope.pagingInfo,

list: $scope.paginArray

};

if (!$scope.$$phase) {

$scope.$apply();

}

$scope.isLoading = false

}, function (error) { }).finally(function(){

$scope.isLoading = false;

});

$location.search('search', !!$rootScope.pagingInfo.search ? $rootScope.pagingInfo.search : null);

$location.search('page', $rootScope.pagingInfo.page);

$location.search('airportFromId', $rootScope.pagingInfo.airportFromId);

$location.search('airportToId', $rootScope.pagingInfo.airportToId);

$location.search('day', !!$rootScope.pagingInfo.day ? $rootScope.pagingInfo.day : null);

$rootScope.preArray = $scope.tours;

};

$scope.loadList();

function initPaginationParams(){

$rootScope.pagingInfo = {

page: $location.search().page !== undefined ? $location.search().page : 1,

itemsPerPage: 5,

search: $location.search().search !== undefined ? $location.search().search : '',

airportFromId: $location.search().airportFromId !== undefined ? $location.search().airportFromId : null,

airportToId: $location.search().airportToId !== undefined ? $location.search().airportToId : null,

day: $location.search().day !== undefined ? $location.search().day : ''

};

}

function correctData() {

for (var i = 0; i < $scope.tours.length; i++) {

$scope.tours[i].timeStart = $filter('date')($scope.tours[i].DateStart, 'HH:mm');

$scope.tours[i].timeFinish = $filter('date')($scope.tours[i].DateFinish, 'HH:mm');

$scope.tours[i].economyPrice = ($scope.tours[i].Price \* Constants.PRICE\_COEF.ECONOMY).toFixed(0);

$scope.tours[i].businessPrice = ($scope.tours[i].Price \* Constants.PRICE\_COEF.BUSSINESS).toFixed(0);

for (var j = 0; j < $scope.tours[i].PossibleDays.length; j++)

{

$scope.tours[i].PossibleDays[j] = utils.translateDays($scope.tours[i].PossibleDays[j]);

}

}

}

});

Список рейсів html сторінка.

<div class="container-fluid">

<h2>Список рейсів</h2>

<hr class="colorgraph">

<div class="container-fluid" >

<div class="col-md-3 col-sm-4">

<label>Пошук:</label>

<div class="input-group col-md-12 col-sm-12">

<input type="text" class="form-control" placeholder="Search" ng-change="search(loadList)"

ng-model="pagingInfo.search"

ng-model-options="{updateOn: 'default blur', debounce: {default: 200, blur: 0} }" name="q">

<div class="input-group-btn">

<button class="btn btn-default" ng-click="">

<i class="glyphicon glyphicon-search"></i>

</button>

</div>

</div>

</div>

<div class="col-md-3 col-sm-4" id="departmentDiv" >

<label for="airportSelectFrom">Виберіть пункт відправлення:</label>

<select class="form-control" id="airportSelectFrom" ng-model="pagingInfo.airportFromId" ng-change="search(loadList)">

<option value="" selected>Виберіть аеропорт</option>

<option ng-repeat="airport in airports" value="{{airport.AirportId}}">{{airport.Name}}</option>

</select>

</div>

<div class="col-md-3 col-sm-4" >

<label for="airportSelectTo">Виберіть пункт прибуття:</label>

<select class="form-control" id="airportSelectTo" ng-model="pagingInfo.airportToId" ng-change="search(loadList)">

<option value="" selected>Виберіть аеропорт</option>

<option ng-repeat="airport in airports" value="{{airport.AirportId}}">{{airport.Name}}</option>

</select>

</div>

</div>

<div class="form-group" ng-if="isLoading">

<br><br><br>

<div class="">

<div style="position: absolute; right: 50%; left: 46%">

<img src="content/images/loading.gif">

</div>

</div>

</div>

<div class="container-fluid mycontent-center" ng-if="!isLoading">

<hr>

<div class="panel panel-default col-sm-10 col-sm-offset-1" ng-repeat="res in tours">

<div class="panel panel-body">

<div class="row">

<div class=" col-sm-5 col-xs-6 col-md-5">

<h5>

Пункт відправлення:

</h5>

<h4>

{{res.AirportFrom.Name }}({{res.AirportFrom.City }})

</h4>

<p>{{res.AirportFrom.Country}}</p>

</div>

<div class="col-md-2" style="">

<img src="../content/images/logo.png" style="width: 130px; height: 100px;">

</div>

<div class="col-md-5 col-sm-5 col-xs-5">

<h5>

Пункт прибуття:

</h5>

<h4>

{{res.AirportTo.Name }}({{res.AirportTo.City }})

</h4>

<p>{{res.AirportTo.Country}}</p>

</div>

</div>

<div class="row">

<div class="col-md-1 col-sm-1 col-xs-1">

<p>

Можливі дні:

</p>

</div>

<div class="col-md-4 col-sm-4 col-xs-4">

<ul>

<li ng-repeat="day in res.PossibleDays">

{{day}}

</li>

</ul>

</div>

</div>

<hr>

<h3>

<div class="col-md-1 col-sm-1 col-xs-1">

<h4>

Ціна:

</h4>

</div>

<div class="col-md-4 col-sm-4 col-xs-4">

<h5>економ клас - {{res.economyPrice}}$</h5>

<h5>бізнес клас - {{res.businessPrice}}$</h5>

</div>

</h3>

<span ng-click="goToDetailPage(res)">Детальніше</span>

</div>

</div>

<div class="container-fluid" ng-show="tours.length === 0 && !isLoading">

<br><br>

<br><br>

<h3 class="text-muted text-center" ng-hide="pagingInfo.search != undefined">This list is empty</h3>

<h3 class="text-muted text-center" ng-show="pagingInfo.search != undefined">There is no doctor with such

name!</h3>

<br><br>

<br><br>

</div>

<div custompagination ng-hide="tours.length === 0"></div>

</div>

</div>

***Замовлення або бронювання квитків***

Серверна частина. API контролер

public class OrdersController : ApiController

{

private ExtraAirContext db = new ExtraAirContext();

// GET: api/Orders

public object GetOrders([FromUri] int userId, [FromUri] string type = "All", [FromUri]int page = 1, [FromUri]int itemsPerPage = 15, [FromUri]string search = null)

{

switch (type)

{

case "All":

{

var list = IoC.Get<IGetOrders>().GetAllOrders(userId);

return IoC.Get<IGetOrders>().GetOrdersWithPagination(new PaginFilteringHelper()

{

ItemsPerPage = itemsPerPage,

Page = page,

Search = search,

Type = type

}, list);

}

case "Future":

{

var list = IoC.Get<IGetOrders>().GetFutureOrders(userId);

return IoC.Get<IGetOrders>().GetOrdersWithPagination(new PaginFilteringHelper()

{

ItemsPerPage = itemsPerPage,

Page = page,

Search = search,

Type = type

}, list);

}

case "Last":

{

var list = IoC.Get<IGetOrders>().GetLastOrders(userId);

return IoC.Get<IGetOrders>().GetOrdersWithPagination(new PaginFilteringHelper()

{

ItemsPerPage = itemsPerPage,

Page = page,

Search = search,

Type = type

}, list);

}

default:

break;

}

return null;

}

// GET: api/Orders/5

[ResponseType(typeof(Order))]

public IHttpActionResult GetOrder(int id)

{

return Ok(IoC.Get<IGetOrders>().GetOrder(id));

}

// POST: api/Orders

[ResponseType(typeof(Order))]

public IHttpActionResult PostOrder(Order order)

{

var list = db.Tours.ToList();

order.Tours = list.Where(l => order.Tours.Any(x => x.TourId == l.TourId)).ToList();

db.Orders.Add(order);

db.SaveChanges();

return Ok();

}

}

Одержання інформації з бази даних.

public class GetOrders : IGetOrders

{

public IEnumerable<OrderDto> GetAllOrders(int userId)

{

using (var dbContext = new ExtraAirContext())

{

var l = dbContext.Orders.Where(x => x.UserId == userId && (x.DateStartTour != null && x.DateFinishTour != null)).ToList();

return l.Select(MapOrderHellper).Where(x => x.tour != null).ToList();

}

}

public IEnumerable<OrderDto> GetFutureOrders(int userId)

{

using (var dbContext = new ExtraAirContext())

{

var l = dbContext.Orders.Where(x => x.UserId == userId && (x.DateStartTour != null

&& x.DateStartTour > DateTime.Now)).ToList();

return l.Select(MapOrderHellper).Where(x => x.tour != null).ToList();

}

}

public IEnumerable<OrderDto> GetLastOrders(int userId)

{

using (var dbContext = new ExtraAirContext())

{

var l = dbContext.Orders.Where(x => x.UserId == userId && (x.DateStartTour != null

&& x.DateStartTour < DateTime.Now)).ToList();

return l.Select(MapOrderHellper).Where(x => x.tour != null).ToList();

}

}

public object GetOrdersWithPagination(PaginFilteringHelper pfHelper, IEnumerable<OrderDto> list)

{

list = list.OrderByDescending(x => x.OrderId).ToList();

if (pfHelper.Search != null)

{

pfHelper.Search = pfHelper.Search.ToLower();

list = list.Where(x =>

(x.tour.AirportFrom.City + x.tour.AirportFrom.Country + x.tour.AirportFrom.Name +

x.tour.AirportTo.City + x.tour.AirportTo.Country + x.tour.AirportTo.Name).ToLower().Contains(pfHelper.Search.Replace(" ", "")));

}

var listPaged = list.Skip((pfHelper.Page - 1) \* pfHelper.ItemsPerPage).Take(pfHelper.ItemsPerPage).ToList();

var json = new

{

count = list.Count(),

list = listPaged

};

return json;

}

}

Клієнтська частина. Замовлення квитків. Контролер.

$http.post(Constants.REST\_URL + "api/BookedPlaces", BookedPlaces, {

headers: {

'Content-type': 'application/json'

}

}).then(function () {

console.log("fine");

}, function () {

console.log("error")

});

}, function (data) {

console.log("fail");

});

$scope.showCheckStep = true;

};

$scope.orderTickets = function () {

var data = prepareDataForOrder();

$http.post(Constants.REST\_URL + "api/Orders", data, {

headers: {

'Content-type': 'application/json'

}

}).then(function () {

alert("Квиток замовлено!")

}, function () {

console.log("error")

});

};

function prepareDataForOrder() {

var Passengers = [];

$scope.allPassengers.forEach(function (pas) {

Passengers.push({

PassengerType: 0,

FirstName: pas.name,

LastName: pas.surname,

Gender: pas.gender,

CoordinateValue: pas.coordinateValue,

IdCard: pas.idCard,

TicketPrice: parseInt(pas.baggage.external) + parseInt(pas.baggage.inner) + parseInt($scope.tour.Price),

BaggageInternal: pas.baggage.inner,

BaggageeExternal: pas.baggage.external

});

});

var Tours = [];

Tours.push({

TourId: $scope.tour.TourId

});

return {

Date: new Date(),

Price: $scope.finishTourResult.Price,

Paid: $scope.showCreditCard,

UserId: jwtHelper.decodeToken(localStorage.getItem('token')).id,

DateStartTour: getGeneralDateFormat($scope.tourSearchInfo.dateStartR, $scope.tourSearchInfo.timeStart),

DateFinishTour: getGeneralDateFormat($scope.tourSearchInfo.dateFinishR, $scope.tourSearchInfo.timeFinish),

Passengers: Passengers,

Tours: Tours

};

}

function validationLastStep() {

if ($scope.activePlacesList.length !== $scope.allPassengers.length) {

alert("Виберіть місця!");

return false;

}

for (var i = 0; i < $scope.allPassengers.length; i++) {

var pas = $scope.allPassengers[i];

if (!pas.name || !pas.surname || !pas.idCard || pas.gender === null) {

alert("Заповніть дані про пасажирів!");

return false;

}

}

return true;

}

function setTourDetails() {

$scope.allPlaces = $scope.allPlaces !== undefined ? $scope.allPlaces : getAllPlaces();

var additionUrl = {

comfort: $scope.tourSearchInfo.tourClass === 'Economy' ? 0 : 1,

dateStart: getGeneralDateFormat($scope.tourSearchInfo.dateStartR, $scope.tourSearchInfo.timeStart),

dateFinish: getGeneralDateFormat($scope.tourSearchInfo.dateFinishR, $scope.tourSearchInfo.timeFinish)

};

tourDetailsService.getTourDetails($routeParams.id, additionUrl).then(function (data) {

setCheckedPlaces(data.BookedPoints);

console.log(data);

}, function (data) {

console.log(data);

})

}

function getAllPassenger() {

var list = [];

for (var i = 0; i < $scope.tourSearchInfo.passengerCount; i++) {

list.push({

name: '',

surname: '',

gender: null,

idCard: '',

baggage: {

inner: 0,

external: 0

},

coordinate: null,

coordinateValue: null

});

}

return list;

}

function setCheckedPlaces(bookedPoints) {

if (bookedPoints === undefined) {

return;

}

$scope.allPlaces.forEach(function (place) {

place.forEach(function (point) {

for (var i = 0; i < bookedPoints.length; i++) {

if (point.coordinate.x === bookedPoints[i].X && point.coordinate.y === bookedPoints[i].Y) {

point.booked = true;

break;

}

}

});

});

}

function getAllPlaces() {

var tmp = $scope.tourSearchInfo.tourClass === 'Economy' ? 'CountOfEconomyPassenger' : 'CountOfBusinessPassenger';

var list = [];

var res = [];

var literals = ['A', 'B', 'C', 'D', 'E', 'F'];

for (var i = 1, x = 0, y = 0; i <= $scope.tour.Plane.MaxCountPassenger[tmp]; i++, x++) {

list.push({

value: (y + 1) + literals[x],

coordinate: {

x: x,

y: y

},

booked: false,

active: false

});

if (i % 6 === 0) {

res.push(list);

list = [];

x = -1;

y++;

}

}

return res;

}

function getGeneralDateFormat(date, time) {

var from = date.split("-");

return from[2] + '-' + from[1] + '-' + from[0] + 'T' + time + ':00';

}

function setupURL() {

$location.search('dateStart', $scope.tourSearchInfo.dateStartR);

$location.search('dateFinish', $scope.tourSearchInfo.dateFinishR);

$location.search('timeStart', $scope.tourSearchInfo.timeStart);

$location.search('timeFinish', $scope.tourSearchInfo.timeFinish);

$location.search('dayStart', $scope.tourSearchInfo.dayStartO);

$location.search('dayFinish', $scope.tourSearchInfo.dayFinishO);

$location.search('tourClass', $scope.tourSearchInfo.tourClass);

$location.search('passengerCount', $scope.tourSearchInfo.passengerCount);

}

function getSearchInfoURL() {

return {

dateStartR: $location.search().dateStart,

dateFinishR: $location.search().dateFinish,

timeStart: $location.search().timeStart,

timeFinish: $location.search().timeFinish,

dayStartO: $location.search().dayStart,

dayFinishO: $location.search().dayFinish,

tourClass: $location.search().tourClass,

passengerCount: $location.search().passengerCount

}

}

function correctData(coef) {

$scope.tour.timeStart = $filter('date')($scope.tour.DateStart, 'HH:mm');

$scope.tour.timeFinish = $filter('date')($scope.tour.DateFinish, 'HH:mm');

$scope.tour.dateStartR = $scope.tourSearchInfo.dateStartR;

$scope.tour.dateFinishR = $scope.tourSearchInfo.dateFinishR;

$scope.tour.Price = ($scope.tour.Price \* coef).toFixed(0);

$scope.tour.dayStart = utils.translateDays($scope.tourSearchInfo.dayStartO);

$scope.tour.dayFinish = utils.translateDays($scope.tourSearchInfo.dayFinishO);

}

});

Список рейсів html сторінка.

<div class="container-fluid">

<h2>Детальна інформація про рейс</h2>

<hr class="colorgraph">

<div class="container-fluid">

<div class="row">

<div class=" col-sm-4 col-xs-4">

<label>Пункт відправлення:</label>

<h4><span>Місто: </span>{{tour.AirportFrom.City}}</h4>

<h4><span>Країна: </span>{{tour.AirportFrom.Country}}</h4>

<h5><span>Аеропорт: </span>{{tour.AirportFrom.Name}}</h5>

<h5><span>Дата відправлення: </span>{{tour.timeStart }} {{tour.dateStartR }}({{tour.dayStart}})</h5>

</div>

<div class=" col-sm-1 col-xs-1">

</div>

<div class=" col-sm-6 col-xs-6">

<label>Пункт прибуття:</label>

<h4><span>Місто: </span>{{tour.AirportTo.City}}</h4>

<h4><span>Країна: </span>{{tour.AirportTo.Country}}</h4>

<h5><span>Аеропорт: </span>{{tour.AirportTo.Name}}</h5>

<h5><span>Дата прибуття: </span>{{tour.timeFinish }} {{tour.dateFinishR }}({{tour.dayFinish}})</h5>

</div>

</div>

<div><h5>Модель літака: {{tour.Plane.Name}}</h5></div>

<div><h3>Ціна: {{tour.Price}}$</h3></div>

<button type="button" class="btn btn-primary col-md-2"

ng-click="openOrderingForm()">Здійснити замовлення

</button>

</div>

<hr class="colorgraph">

</div>

</div>

<div class="container-fluid" ng-if="ordering && isAuthorized">

<div class="container-fluid">

<div ng-repeat="passenger in allPassengers">

<h3>{{($index + 1)}} Пасажир</h3>

<div class="row">

<div class="col-md-3 col-sm-3">

<input type="text" placeholder="Ім'я" ng-model="passenger.name" class="form-control" required

maxlength="30">

</div>

<div class="col-md-3 col-sm-3">

<input type="text" placeholder="Прізвище" ng-model="passenger.surname" class="form-control" required

maxlength="30">

</div>

</div>

<div class="row" style="padding-top: 5px;">

<div class="col-md-3 col-sm-3">

<input type="text" placeholder="Номер паспорта" ng-model="passenger.idCard" class="form-control"

required

maxlength="30">

</div>

<div class="col-md-3 col-sm-3">

<div class="form-group" style="padding-top: 10px;">

<label class="control-label col-sm-3">Стать:</label>

<div class="row">

<div class="col-sm-4">

<label class="radio-inline">

<input type="radio" name="gender" id="femaleRadio" value="1"

ng-model="passenger.gender">Жінка

</label>

</div>

<div class="col-sm-4">

<label class="radio-inline">

<input type="radio" name="gender" id="maleRadio" ng-model="passenger.gender"

value="0">Чоловік

</label>

</div>

</div>

</div>

</div>

</div>

<div class="row" ng-if="tourSearchInfo.tourClass === 'Economy'">

<div class="col-md-3 col-sm-3">

<label>Зареєстрований багаж:</label>

<md-radio-group ng-model="passenger.baggage.external">

<md-radio-button value="0">Жодної</md-radio-button>

<md-radio-button value="27">23 кг(+27$)</md-radio-button>

<md-radio-button value="38">32 кг(+38$)</md-radio-button>

</md-radio-group>

</div>

<div class="col-md-3 col-sm-3">

<label>Ручний багаж:</label>

<md-radio-group ng-model="passenger.baggage.inner">

<md-radio-button value="0">Маленька(безкоштовна)</md-radio-button>

<md-radio-button value="14">Велика(+14$)</md-radio-button>

</md-radio-group>

</div>

</div>

<br>

</div>

<hr class="colorgraph">

<div class="row">

<div class="col-md-3 col-sm-3">

<label>Метод Реєстрації:</label>

<md-radio-group ng-model="methodRegistration.value">

<md-radio-button value="0">Онлайн</md-radio-button>

<md-radio-button value="10">Аеропорт(+10$)</md-radio-button>

</md-radio-group>

</div>

<button type="button" class="btn btn-default col-md-2" data-toggle="modal"

data-target="#plaseChooseModalView"

ng-click="getTourDetails()">Обрати місця

</button>

</div>

<div class="row">

<button type="button" class="btn btn-default col-md-2" ng-click="lastStep()">Продовжити</button>

</div>

<div ng-if="showCheckStep">

<h3>Перевірити дані:</h3>

<div ng-repeat="pas in allPassengers">

<h3>{{($index + 1)}} Пасажир</h3>

<div class="row">

<div class="col-md-3">

<h4>Ініціали: {{pas.name}} {{pas.surname}}</h4>

<h5>Номер паспорта: {{pas.idCard}}</h5>

<h5>Зареєстрований багаж:: {{pas.baggage.externalValue}}</h5>

<h5>Ручний багаж: {{pas.baggage.innerValue}}</h5>

</div>

<div class="col-md-3">

<h4>Стать: {{pas.gender === 0 ? 'Чоловік' : 'Жінка' }}</h4>

<h5>Місце: {{pas.coordinateValue}}</h5>

</div>

</div>

<hr class="colorgraph">

</div>

<h4>Метод реєстрації {{methodRegisterValue}}</h4>

<h4><label>Ціна: {{finishTourResult.Price}}$</label></h4>

</div>

<div ng-if="showCreditCard">

<credit-card></credit-card>

</div>

<div ng-if="!showCreditCard && showCheckStep">

<button type="button" class="btn btn-default col-md-2" ng-click="orderTickets()">Забронювати</button>

</div>

</div>

</div>

<choose-place></choose-place>