Національний університет «Полтавська політехніка імені Юрія Кондратюка»

(повне найменування вищого навчального закладу)

\_Навчально-науковий інститут інформаційних технологій та робототехники\_

(повна назва факультету)

\_Кафедра комп’ютерних та інформаційних технологій і систем\_

(повна назва кафедри)

**Пояснювальна записка**

**до дипломного проекту (роботи)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_бакалавра\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(рівень вищої освіти)

на тему

Веб-сервіс пошуку найдешевших квитків на авіарейси \_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Виконав: студент \_4\_ курсу, групи \_402-ТН\_

спеціальності

\_\_ 122 Комп’ютерні науки \_

(шифр і назва напряму)

Сімперович М.М. \_

(прізвище та ініціали)

Керівник Руденко О.А. \_

(прізвище та ініціали)

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Полтава – 2021 року

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНИКИ**

**КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ**

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

**спеціальність 122 «Комп’ютерні науки»**

**на тему**

**«Веб-сервіс пошуку найдешевших квитків на авіарейси»**

**Студента групи 402-ТН Сімперовича Микити Михайловича**

Керівник роботи

кандидат технічних наук,

доцент Руденко О.А.

Завідувач кафедри

кандидат технічних наук,

доцент Головко Г.В.

Полтава – 2021

**РЕФЕРАТ**

**Кваліфікаційна робота бакалавра:** 68 сторінок, 25 малюнків, 5 таблиць, 13 додатків, 28 джерела.

**Об’єкт дослідження**: системи пошуку найдешевших квитків на авіарейси.

**Мета роботи**: розробка веб-сервісу по пошуку найдешевших квитків на авіарейси, що забезпечить фундамент для подальшої розробки та вдосконалення веб сервісу пошуку та придбання найвигіднішого варіанту авіаквитків.

**Методи**: об’єктно-орієнтоване програмування, застосування API віддаленого сервера, розробка інтерфейсу користувача виконана з використанням мови гіпертекстової розмітки HTML та таблиць стилів CSS, для реалізації функціоналу форми було використано мову програмування JavaScript, застосування системи контролю версій GitHub.

**Ключові слова:** веб сервіс, форма, інтерфейс, плагін, бібліотеки JavaScript, Webpack, CSS фреймворк Materializ, запит JSON, API доступ.

**ABSTRACT**

**Bachelor’s qualification work:** 68 pages, 25 drawings, 5 tables, 13 apps, 28 sources.

**Object of research:** search systems for the cheapest flights tickets.

**The purpose of the work**: development of a web service for finding the cheapest tickets for flights, which will provide a foundation for further development and improvement of a web service for finding and buying the most profitable version of tickets.

**Methods:** object-oriented programming, application of remote server API, development of user interface performed using HTML hypertext markup language and CSS style sheets, JavaScript programming language will be used to implement the functionality of the service, used GitHub version control system.

**Keywords:** web service, form, interface, plugin, JavaScript libraries, Webpack, CSS Framework Materializ, JSON request, API access.

**ЗМІСТ**

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ 6](#_Toc74232681)

[ВСТУП 7](#_Toc74232682)

[РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД СИСТЕМ ТА ДОДАТКІВ ПО ПОШУКУ найдешевих АВІАКВИТКІВ 9](#_Toc74232683)

[1.1 Опис предметної області 9](#_Toc74232684)

[1.2 Огляд існуючих програмних рішень 9](#_Toc74232685)

[1.3 Вимоги до системи 18](#_Toc74232686)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ веб сервісу ПОШУКУ НАЙДЕШЕВШИХ КВИТКІВ на авіарейси 19](#_Toc74232687)

[2.1 Функціонал та структура системи 19](#_Toc74232688)

[2.2 Платформа та засоби розробки системи. 22](#_Toc74232689)

[2.2 Архітектура проекту 23](#_Toc74232690)

[РОЗДІЛ 3 Розробка веб сервісу ПОШУКУ НАЙДЕШЕВШИХ КВИТКІВ НА авіарейсИ 25](#_Toc74232691)

[3.1 Підключення та налаштування Webpack 25](#_Toc74232692)

[3.2 Підключення платформи Materialize 26](#_Toc74232693)

[3.3 Розробка дизайну проекту 26](#_Toc74232694)

[3.4 Створення функціоналу для взаємодії з сервером 28](#_Toc74232695)

[3.5 Реалізація функціоналу проекту 32](#_Toc74232696)

[3.6 Вивід авіаквитків на сторінку 36](#_Toc74232697)

[3.7 Реалізація функціоналу додавання квитків в обране 39](#_Toc74232698)

[РОЗДІЛ 4 ТЕСТУВАННЯ 40](#_Toc74232699)

[4.1 Вибір виду тестування. 40](#_Toc74232700)

[4.2 Тест план 41](#_Toc74232701)

[4.3 Введення в експлуатацію 43](#_Toc74232702)

[ВисновкИ 44](#_Toc74232703)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 45](#_Toc74232704)

[ДОДАТОК А PACKAGE.JSON 48](#_Toc74232705)

[ДОДАТОК Б WEBPACK.CONFIG.JS 49](#_Toc74232706)

[ДОДАТОК В style.css 51](#_Toc74232707)

[ДОДАТОК Г materialize.js 52](#_Toc74232708)

[ДОДАТОК Д index.html 53](#_Toc74232709)

[ДОДАТОК Е app.js 56](#_Toc74232710)

[ДОДАТОК Ж appServise.js 58](#_Toc74232711)

[ДОДАТОК З location.js 60](#_Toc74232712)

[ДОДАТОК І form.js 63](#_Toc74232713)

[ДОДАТОК К currency.js 64](#_Toc74232714)

[ДОДАТОК Л tickets.js 65](#_Toc74232715)

[ДОДАТОК М Store/favorites.js 67](#_Toc74232716)

[ДОДАТОК Н Views/favorites.js 68](#_Toc74232717)

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

**GDS** – Global Distribution System

**ОТА** – Online Travel Agency

**UML** – Unified Modeling Language

**IDEF0** – Integrated DEFinition

**HTTP** – HyperText Transfer Protocol

**CSS** – Cascading Style Sheets

**JS** – Java Script

**API** – Application Programming Interface

**NPM** – Node Package Manager

**БД** – база даних

**ПЗ** – програмне забезпечення

**ООП** – об’єктно орієнтоване програмування

# ВСТУП

Для далеких подорожей ідеально підходить авіатранспорт. Він має безліч переваг: швидкість, комфорт, можливість дістатися в будь-які великі міста світу швидко і безпечно. Однак є у нього і істотний недолік – це висока вартість авіаквитків.

Пошук дешевих авіаквитків – це не просте заняття, тут потрібно знати кілька важливих нюансів [1].

Кожна авіакомпанія має свою цінову політику, тому вартість авіаквитків в одному напрямку може дуже відрізнятися. Ті люди, хто часто користуються літаками, знають, що можна істотно заощадити на пересуванні. Але ринок авіабілетів влаштовано так, що на першому місці ланцюжка стоять авіакомпанії – виробники продукту. Для пошуку найвигідніших варіантів використовуються спеціальні системи пошуку та бронювання авіаквитків. Крім можливості вибору напрямку, дати, вартості маршруту, вони також дають шанс взяти участь у спеціальних акціях авіакомпаній, видати пріоритет найдешевшим альтернативам. Саме за допомогою Інтернету покупка недорогих квитків стає простою і швидкою. У великих містах великою популярністю користуються тренінги про те, як користуватися системами пошуку та бронювання авіаквитків [2].

Існують різні системи пошуку авіаквитків. Найпоширеніший і вірний варіант покупки авіаквитків – покупка через Інтернет на сайтах авіакомпаній або агентів. Але вартість на одні і ті ж рейси може бути абсолютно різною.

Спеціалізовані агентства (альянси, авіакомпанії) мають власну базу даних з авіаквитками. Але є сайти – пошуковики (агрегатори або метапошуковики), які порівнюють ціни авіакомпаній. Такі сайти видають інформацію, яку отримують з різних компаній і показують споживачеві всі пропозиції. Клієнт може особисто відфільтрувати список за різними критеріями: ціна (від найдешевших квитків до дорогих), рейтинг (на основі відгуків від клієнтів), критерії (прямі рейси, з пересадкою), час відправлення, спосіб оплати тощо. Як наслідок, вигідніше користуватися системами-сайтами – пошуковиками, в цьому випадку мандрівник в 99 відсотках випадків знайде кращий варіант. Такі пошуковики вигідні звичайному мандрівникові тим, що вони представляють всі компанії, тому клієнт може вибрати і купити те, що йому найбільше підходить. Адже великі авіакомпанії на слуху, а ось про маленькі мало хто знає, при цьому саме у них найвигідніші пропозиції [1].

Тема дипломної роботи присвячена розробці веб сервісу пошуку найдешевших квитків на авіарейси, основною метою якої є пошук найвигіднішого варіанту придбання авіаквитків серед безлічі існуючих на ринку пропозицій авіакомпаній. На сьогодні засоби по пошуку найдешевших варіантів авіаквитків є доволі популярним. Такі системи дозволяють значно скоротити час пошуку та пошук найбільш вигіднішого варіанту.

Так як пошук авіаквитків здійснюється за даними декількох авіакомпаній, які мають власну БД з авіаквитками, то дана система не передбачає розробку власної БД, тому в цій роботі буде використано API запити, які будуть відправлятися на сервер спеціалізованого агентства та приймати результати пошуку.

Завдяки API запитам можна отримувати інформацію з різних джерел, сам проект виконується швидше та ефективніше, використовуються аналітика реального часу і високо інтегровані інструменти.

Відкриті API дозволяють зовнішнім розробникам отримувати доступ до інформації та інтегрувати інформацію, яка передається з одного програмного інструменту в інший. При проектуванні даної роботи буде використано API доступу від Aviasales.

# РОЗДІЛ 1

# АНАЛІТИЧНИЙ ОГЛЯД СИСТЕМ ТА ДОДАТКІВ ПО ПОШУКУ найдешевих АВІАКВИТКІВ

## 1.1 Опис предметної області

Якщо уявити авіаринок, то це:

* сотні авіакомпаній;
* кілька систем дистрибуції (GDS) з різними правилами та опціями;
* консолідатори зі спеціальними умовами від авіакомпаній, тисячі онлайн-агентств подорожей (ОТА);
* десятки різних тарифів;
* опції та правила у кожної авіакомпанії;
* різноманітні обмеження з боку GDS і авіакомпаній для різних ринків і країн;

Пошук по всьому цьому масиву даних – справа масштабна [3].

Тому веб сервіс пошуку найдешевших авіаквитків, що розробляється, повинен передбачати пошук авіаквитків за заданими користувачем параметрами з різних компаній і показувати користувачеві всі пропозиції. Потім клієнт може особисто відфільтрувати список за різними критеріями: ціна (від найдешевших квитків до дорогих), час відправлення, інші критерії (прямі рейси, з пересадкою тощо).

## 1.2 Огляд існуючих програмних рішень

Розвиток сучасних систем по пошуку найдешевших варіантів перельоту з точки А в точку Б неможливий без аналізу існуючих рішень і накопиченого досвіду в цій сфері. Тому дослідження існуючих програмних рішень по пошуку найдешевших авіаквитків є важливим та актуальним в межах реалізації даного проекту.

Почнемо огляд з пошукової системи Aviasales, яка має ряд переваг.

* Зручна навігація інтерфейсу сайту.
* Досить велика база компаній (728). А, як відомо, чим більше пропозицій, тим більше шансів знайти найвигідніший варіант.
* Даний сервіс з пошуку авіаквитків співпрацює з перевіреними компаніями-партнерами, завдяки чому вдалося заробити хорошу репутацію.
* Aviasales представляє як найдешевші квитки, так і акції. Тому шанси на отримання квитків за вигідною ціною збільшуються.
* Видає інформацію не тільки про прості рейси, але і про чартери. Дуже зручно і дешево!
* Є можливість передплати.
* Наявність календаря низьких цін – відчутна перевага сайту.
* Система відстежує акції і лімітовані пропозиції від авіаперевізників.
* Крім покупки квитків на літак, можна через сайт замовити номер у готелі.
* На сервісі зберігається історія пошуку.
* Наявність різних фільтрів, які допоможуть знайти кращий варіант за вказаними критеріями.
* Aviasales надає мандрівникам додаткові можливості, наприклад, оформлення страховки для туриста онлайн, оренда автомобіля.
* Проведення різноманітних конкурсів та акцій (за підтримки авіакомпаній).
* Зручні програми для власників iPod, iPhone, а також для власників телефонів на базі Аndroid.

Але при всіх цих плюсах є кілька мінусів:

* швидкість роботи не найвища, але не критично;
* часті зміни на сайті;
* Відсутня підписка [1].

На рис. 1.1 зображено головну сторінку сервісу Aviasales.

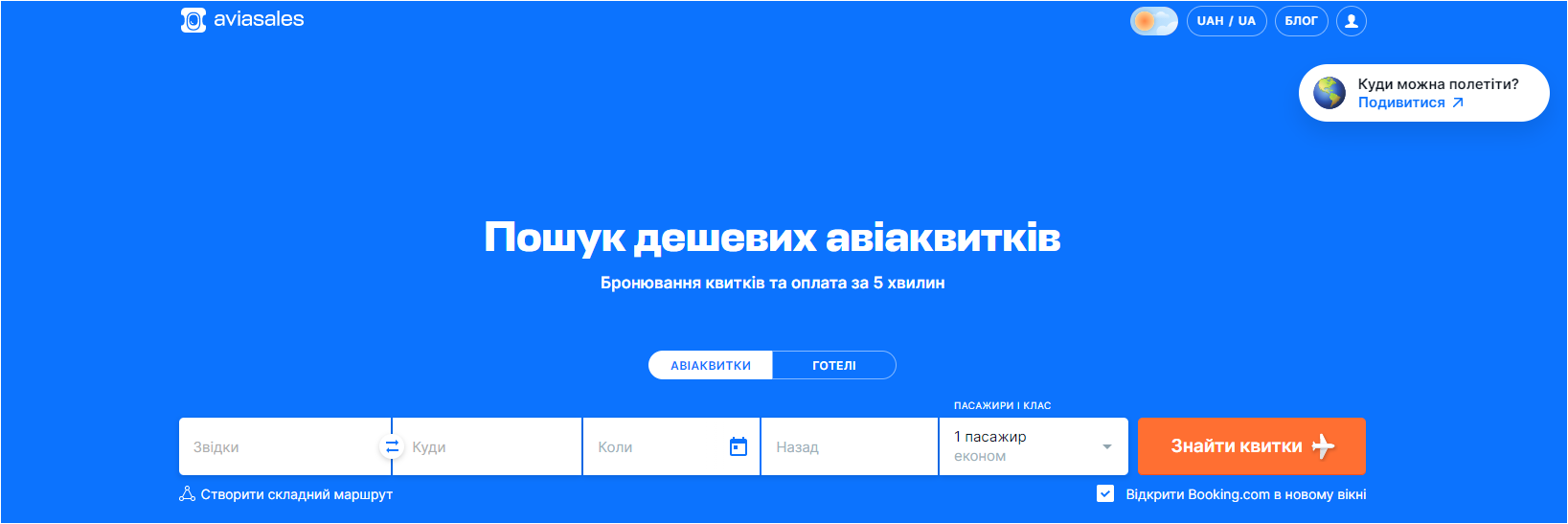


Рисунок 1.1 – Головна сторінка сервісу Aviasales

Пошук найдешевших авіаквитків можна здійснювати за допомогою сервісу Онлайнбілет.

* Пошукова система має три вкладки, завдяки цьому можна знайти квитки для складного маршруту.
* Представлено більше семисот авіакомпаній і десятки систем для бронювання.
* За допомогою цього сайту можна замовити залізничні квитки, оформити туристичну страховку.
* Наявність окремої вкладки «Спецпропозиції» допоможе бути в курсі всіх вигідних акцій.
* Сайт обробляє дані швидко, результат з’являється через пару секунд. Історія пошуку зберігається автоматично.
* Наявність спеціального календаря з вартістю, який допоможе заощадити і дізнатися, в який період найвигідніше летіти [1].

На рис.1.2 зображено головну сторінку сервісу Онлайнбілет.

Рисунок 1.2 – Головна сторінка сервісу Онлайнбілет

YOUROUTE – сервіс, який дозволяє не тільки знайти дешеві авіаквитки, але і допомагає спланувати свою подорож: скористатися готовим маршрутом, побачити реальний фотозвіт, задати питання досвідченому туристу і багато іншого. Відносно сервісу, варто відзначити такі особливості.

* На сайті все працює швидко.
* Найпростіша форма тільки з усіма необхідними основними параметрами.
* Багатофункціональність сервісу: бронювання квитків, готелів, готові маршрути і багато іншого.
* Велика база авіакомпаній і різні системи броні.
* Є необхідні фільтри, що дозволяють знайти оптимальний варіант за ціною, датою вильоту тощо.

YOUROUTE має кілька мінусів.

* Відсутність мобільної версії.
* Сайт не надає інформації про акції і спеціальні пропозиції.
* Немає розсилки.
* У базі небагато партнерів [1].

На рис.1.3 зображена головна сторінка сервісу YOUROUTE.

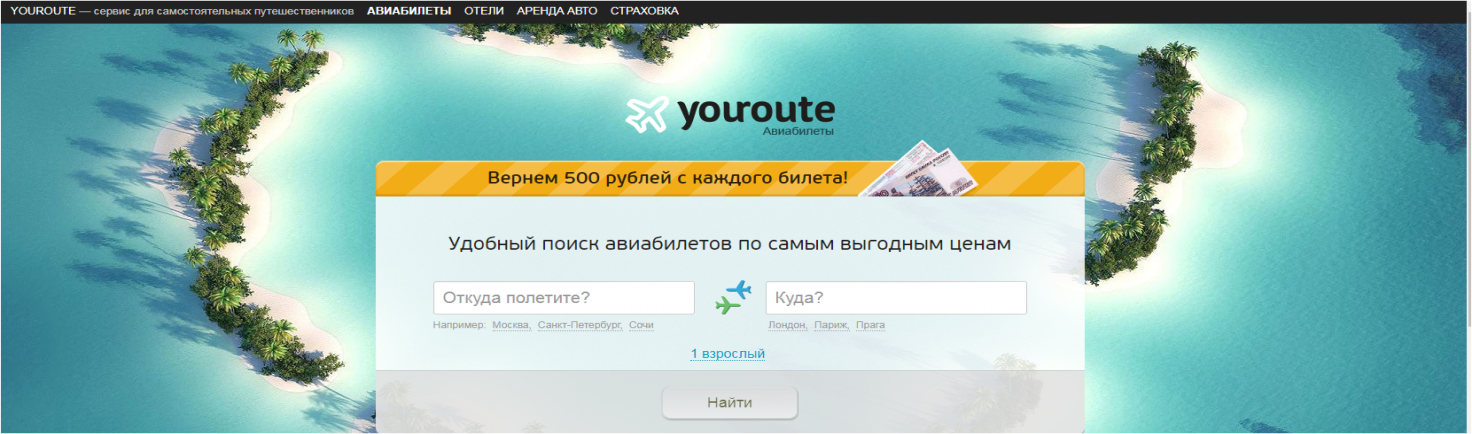


Рисунок 1.3 – Головна сторінка сервісу YOUROUTE

Ще один метапошуковик [Skyscanner](https://clk.tradedoubler.com/click?p=232108&a=2993029&g=21113908&url=https://www.skyscanner.com) з багатьма налаштуваннями, такими як тривалість перельоту, прямі та рейси з пересадкою, дати вильоту. Також сервіс має систему відслідковування напрямів за email підпискою та рекомендує кращі дні для вильоту.

Переваги системи пошуку Skyscanner.

* Має велику базу компаній (близько тисячі).
* Зручний пошук.
* Багато фільтрів.
* Календар цін.
* Є можливість отримувати інформацію про квитки через розсилку.
* Більше можливостей: замовлення готелю, оренда авто через сайт.
* Наявність мобільного застосування і мобільної версії сайту.
* Різні форми пошуку: не тільки за стандартом, але і через вкладку «місто» і подібне.
* Можна обрати перегляд пропозицій на рік і визначитися з датою вильоту (виходячи їх найвигідніших пропозицій).
* На сайті регулярно проводять конкурси з різними призами, акції.
* Робота веб-ресурсу доволі швидка.

Недоліки системи.

* Не досить добре шукає по лукостам [1].

На рис.1.4 зображена головна сторінка сервісу Skyscanner.

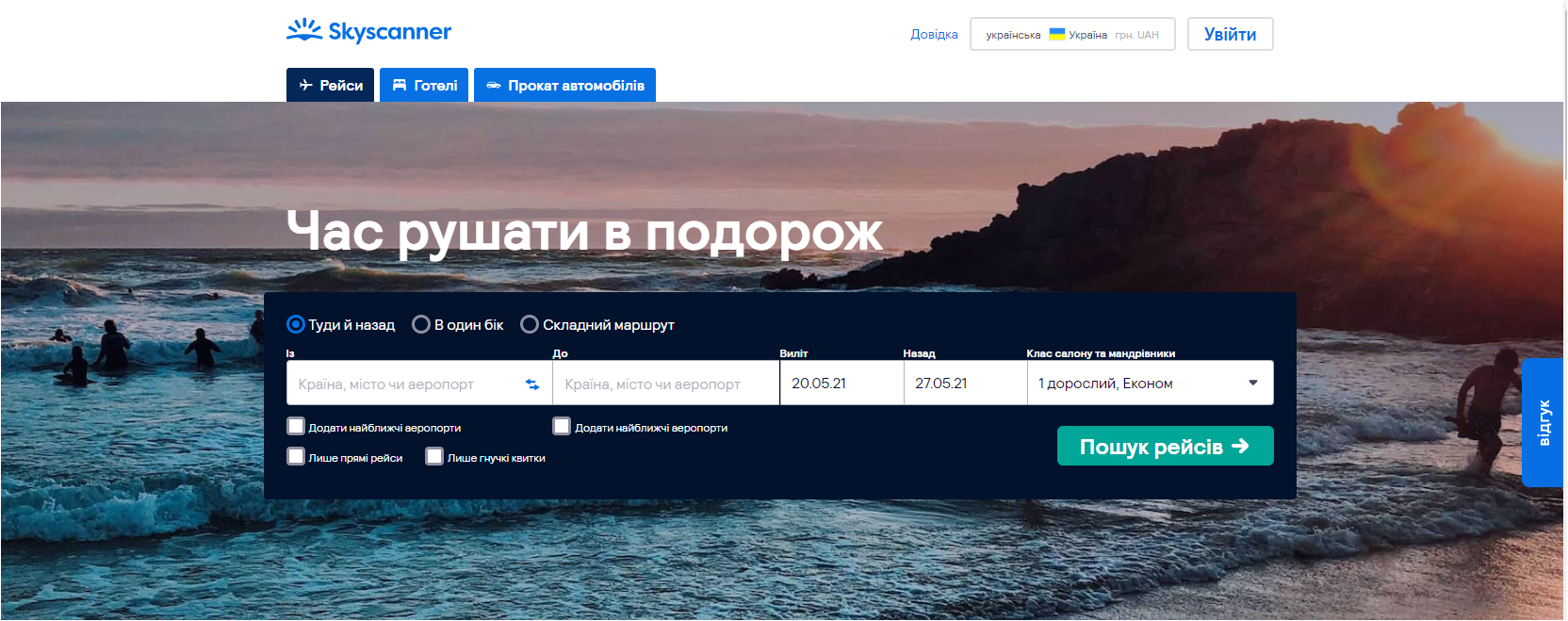


Рисунок 1.4 – Головна сторінка сервісу Skyscanner

Український сервіс [TripMyDream.ua](https://ad.admitad.com/g/hrw43drre76c40ea4461cb4402ea55/?ulp=https%3A%2F%2Favia.tripmydream.com) має досить стандартний набір для пошуку та декілька унікальних переваг.

Одна з них – каталог авіазнижок. В кілька кліків можна подивитися одразу усі дешеві авіа з України. Також є ряд фільтрів, які допоможуть підібрати найкращу пропозицію.

Ще одна перевага сервісу – пошук “Переліт+готель” або сам собі туроператор. Зручно, коли знаєш який тип відпочинку хочеш, але не знаєш куди поїхати. Сервіс підкаже найкращі варіанти під інтереси та бюджет користувача.

Окрім пропозицій, сервіс одразу підтягує інформацію про погоду, умови, середні ціни, що досить зручно при плануванні поїздки.

На рис. 1.5 зображена головна сторінка сервісу [TripMyDream.ua](https://ad.admitad.com/g/hrw43drre76c40ea4461cb4402ea55/?ulp=https%3A%2F%2Favia.tripmydream.com).

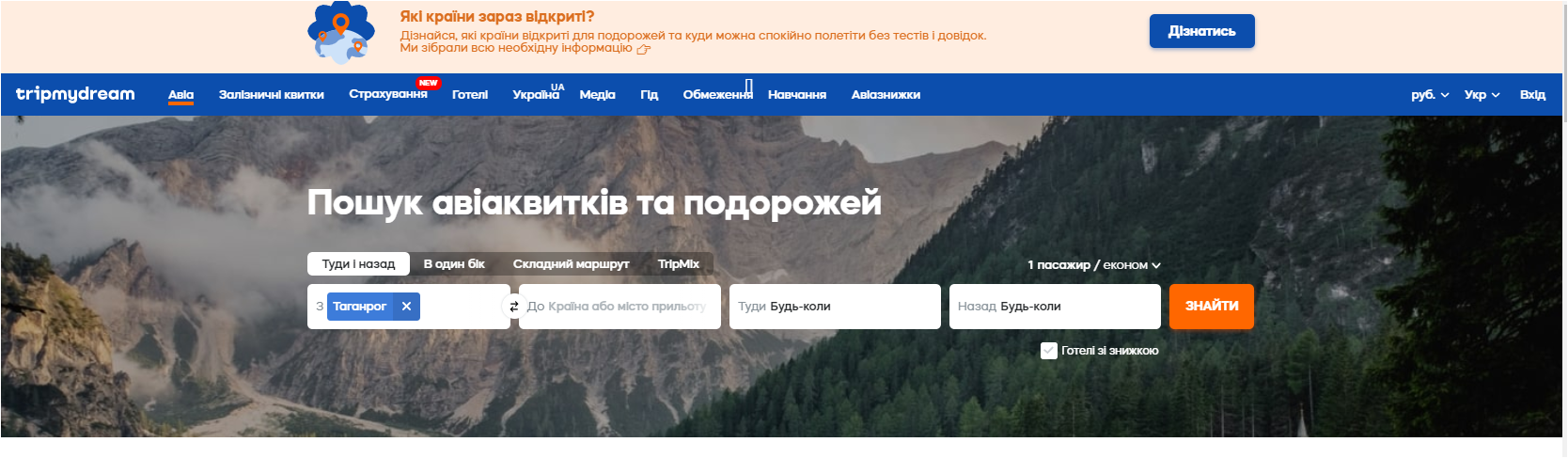


Рисунок 1.5 – Головна сторінка сервісу [TripMyDream.ua](https://ad.admitad.com/g/hrw43drre76c40ea4461cb4402ea55/?ulp=https%3A%2F%2Favia.tripmydream.com)

Ще один український сервіс пошуку дешевих авіаквитків Турист.

* Збирає дані за принципом OTA – безпосередньо у перевізників.
* Надає можливість комбінувати сегменти різних авіакомпаній і інших видів транспорту в одному квитку.
* Комбінувати різні сегменти – це спосіб відмінно заощадити.

«Турист» – планувальник маршруту, який комбінує найдешевші варіанти переміщення в поїздці. Сервіс збирає пропозиції від європейських і азіатських лоукостів, авіакомпаній, залізничних та автобусних перевізників. Система розроблена таким чином, що шукає «ті самі» шматки квитків на шляху з пункту А в пункт Б і формує їх в єдиний маршрут [4].

На рис.1.6 зображена головна сторінка сервісу Турист.

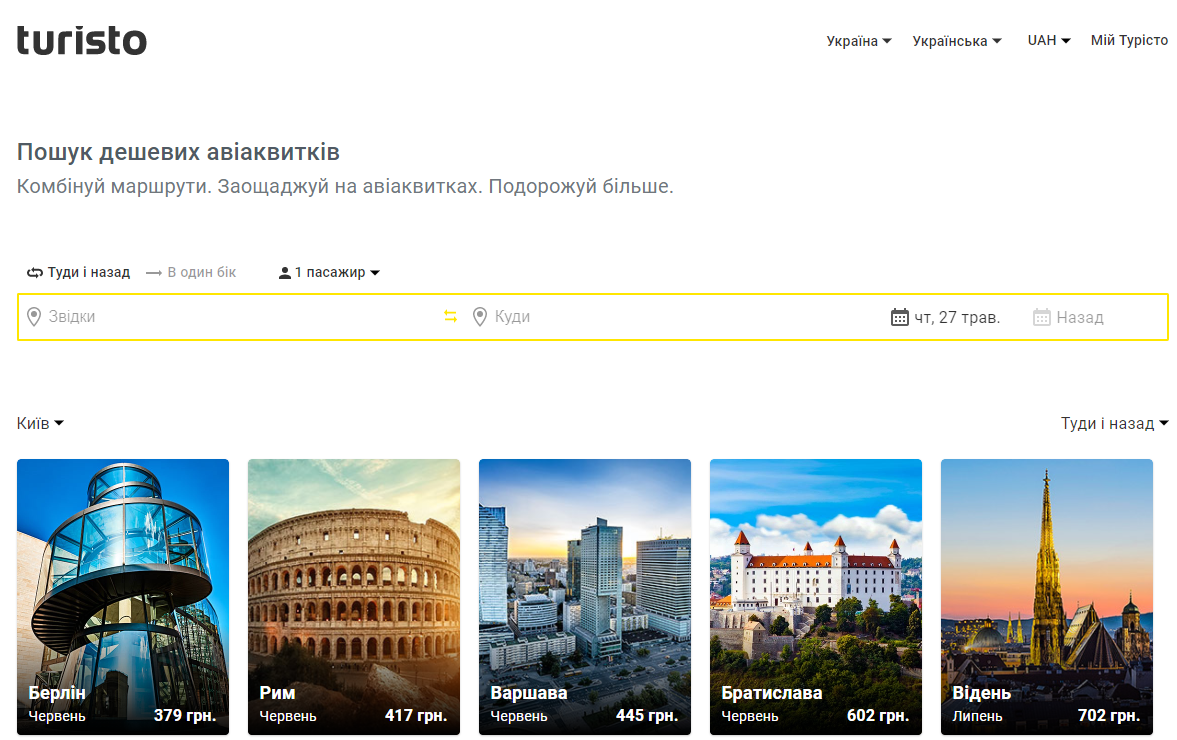


Рисунок 1.6 – Головна сторінка сервісу Турист

Kayak – це закордонний сервіс. Функції та переваги цього сервісу.

* Швидка робота.
* Розширені можливості (замовлення готелю, оренда машини тощо).
* Функціональний додаток для мобільного.
* Можливість підписки на спеціальні пропозиції та акції.
* Наявність зручного і зрозумілого інтерфейсу.
* База з 400 авіакомпаній.
* Функція збереження переглядів [1].

На рис.1.7 зображена головна сторінка сервісу Kayak.

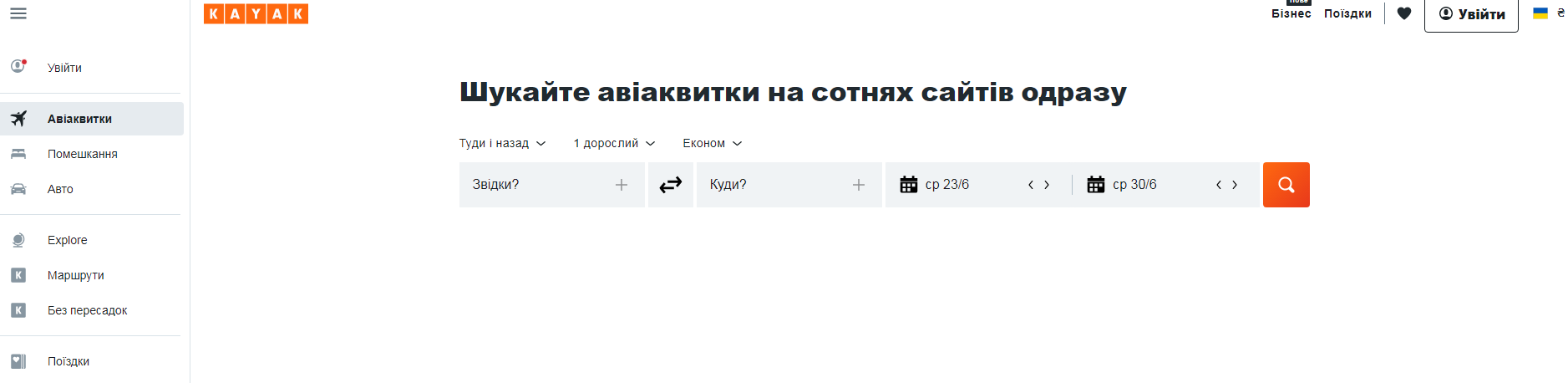


Рисунок 1.7 – Головна сторінка сервісу Kayak

OneTwoTrip – агрегатор, який дозволяє придбати квитки не тільки на літак, але і на поїзди та автобуси. Тут можна замовити екскурсію, забронювати прокат авто, номер в готелі, купити страховку (рис.1.8).

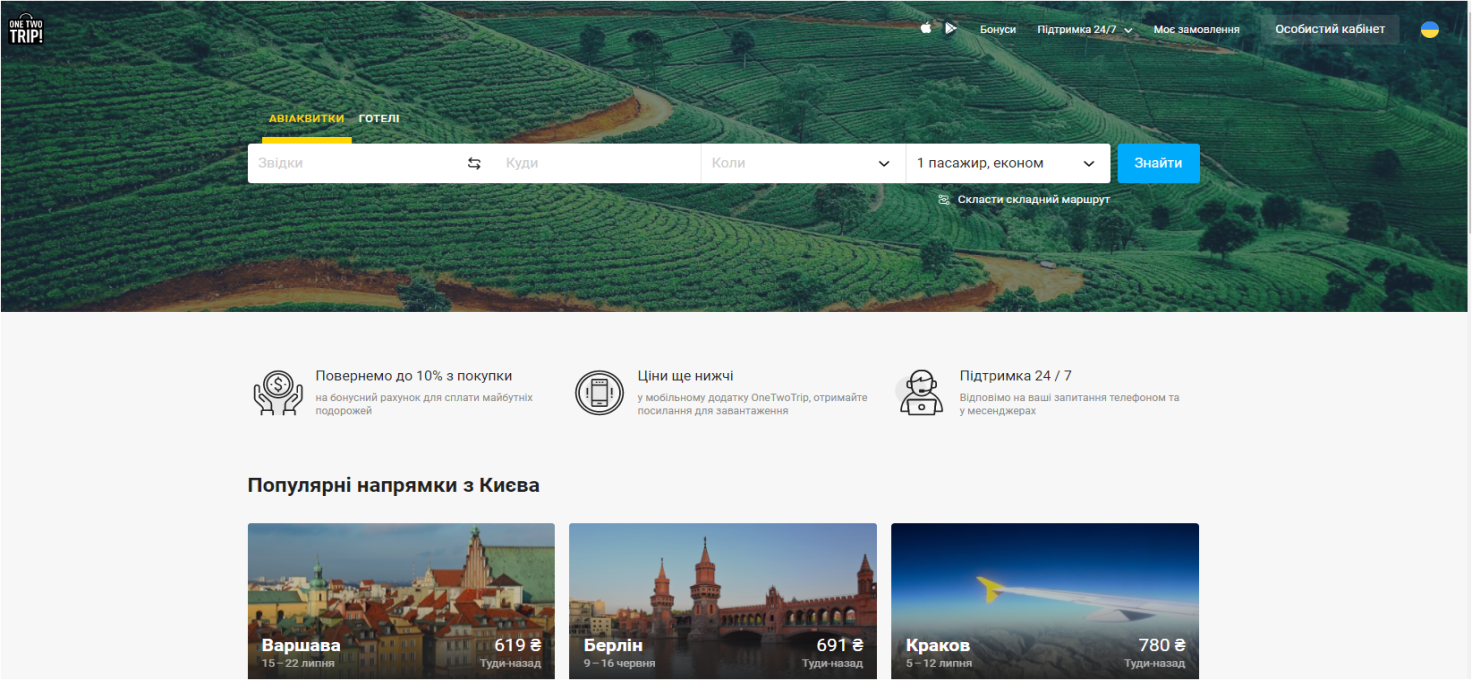


Рисунок 1.8 – Головна сторінка сервісу OneTwoTrip

Основні переваги сервісу.

* База з більш ніж 800 авіакомпаній.
* Календар цін для порівняння вартості в найближчі дати.
* Доступна онлайн-реєстрація на рейс.
* Бонусна система, що дозволяє повернути до 10% від вартості бронювання.
* Дружній інтерфейс вибору та бронювання [5].

OZON.travel – цей ресурс також дозволяє вибрати квитки на літак, потяг, забронювати готель, трансфер і купити страховку (рис.1.9).

Основні функції та переваги сервісу OZON.travel.

* Доступна автореєстрація на посадку.
* В процесі оформлення можна змінювати дати, виправляти помилки в реєстраційних даних.
* Оформлення повернення квитків.
* Показує підсумкову вартість, включаючи всі збори і комісії.
* Є розділ з вигідними пропозиціями на перельоти [5].

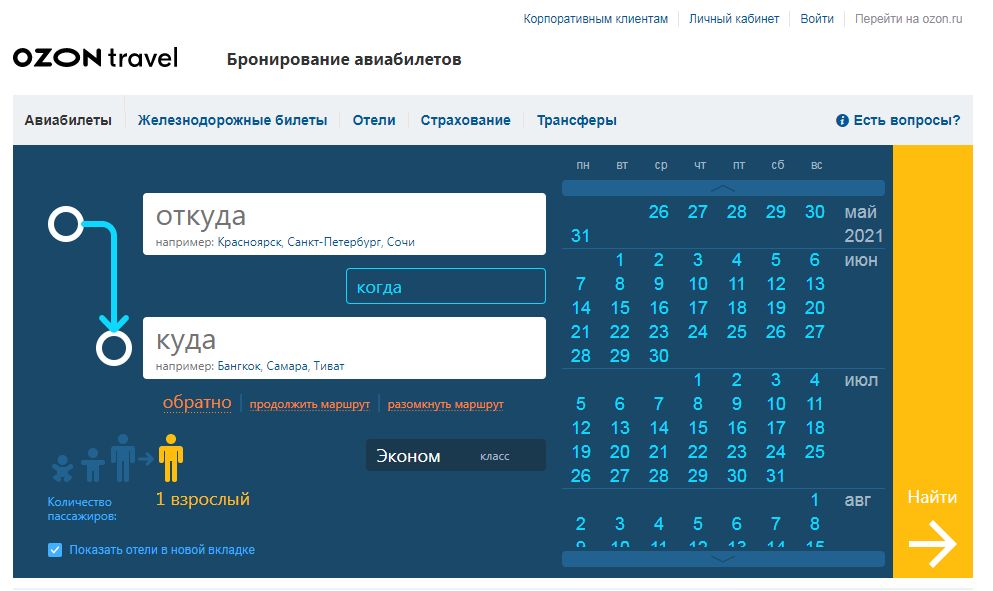


Рисунок 1.9 – Головна сторінка сервісу OZON.travel

Tutu.ru***.*** На сервісі можна придбати квитки на літак, потяг, електричку, автобус, купити тур, забронювати готель, а також знайти цікаву пригоду з походами, екскурсіями, морськими подорожами, фото-турами тощо.

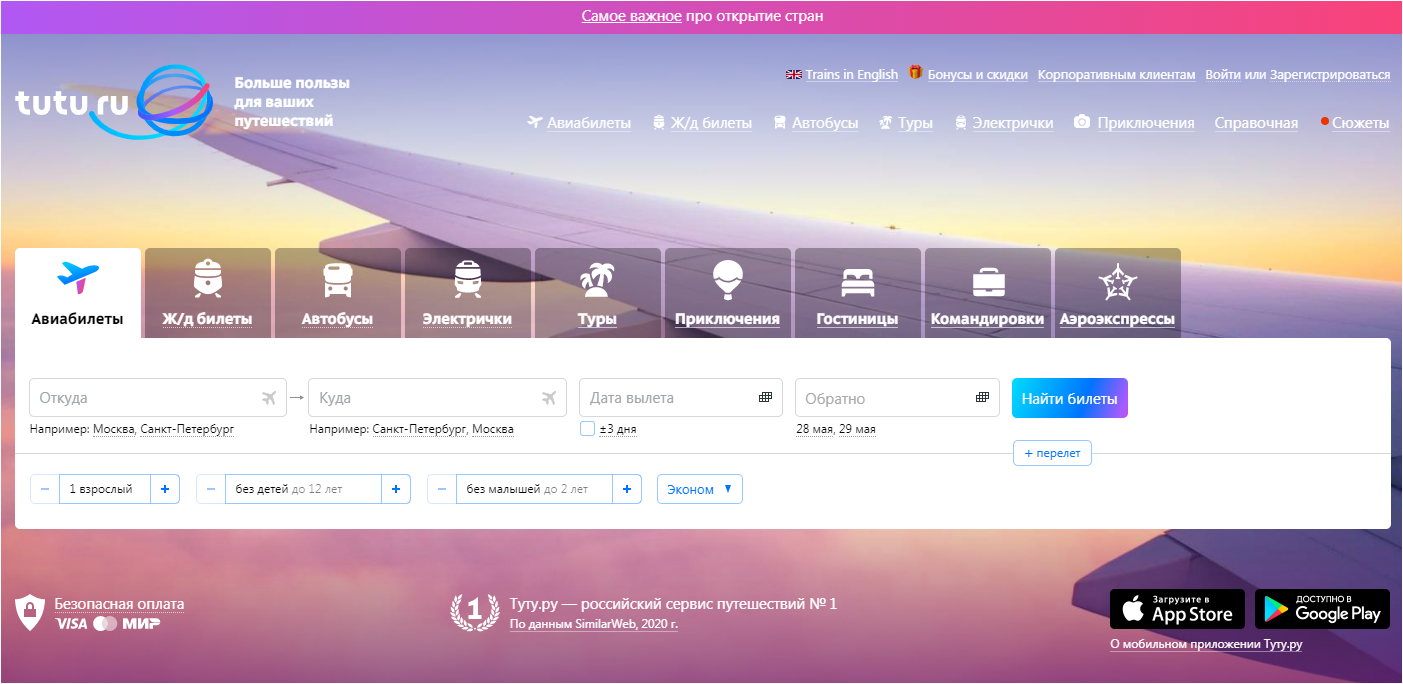


Рисунок 1.10 – Головна сторінка сервісу Tutu.ru

Основні функції та переваги сервісу Tutu.ru:

* можна придбати квиток на літак, потяг, автобус і електричку, а також купити тур та забронювати готель;
* є рейтинг перевізників з оцінками пасажирів;
* доступне онлайн-повернення;
* корисні матеріали для мандрівників;
* акції та розіграші [5].

На рис. 1.10 зображена головна сторінка сервісу Tutu.ru.

## 1.3 Вимоги до системи

Для впровадження системи необхідно спершу визначити вимоги до неї і на їх основі зробити вибір необхідних технологій.

Цілі впровадження веб сервісу пошуку найдешевших квитків на авіарейси.

1. Скорочення часу, що витрачається на пошук вигіднішого варіанту квитка.

2. Створення зручного для користувача інтерфейсу.

3. Надання користувачам альтернативного засобу пошуку авіаквитків.

Основними функціональними вимогами до системи пошуку найдешевших авіаквитків є:

* пошук та отримання з різних джерел інформації про розклад, тарифи, рейси перевізників, правила їх застосування та іншої інформації;
* наявність налаштування фільтрації результатів пошуку;
* можливість додавання квитків в обране;
* швидке додавання нової інформації на сайт;
* умова простоти використання (інтуїтивно зрозумілий інтерфейс).

Додатковою функцією може бути демонстрація прогнозу погоди у місці відправлення та місці призначення, що буде зручно при плануванні поїздки.

# РОЗДІЛ 2

# ПРОЕКТУВАННЯ веб сервісу ПОШУКУ НАЙДЕШЕВШИХ КВИТКІВ на авіарейси

## 2.1 Функціонал та структура системи

Для функціонального моделювання системи скористаємося методологією IDEF0.

Загальний процес пошуку найдешевших квитків на авіарейси зобразимо у вигляді контекстної діаграми, яка представлена на рис. 2.1.

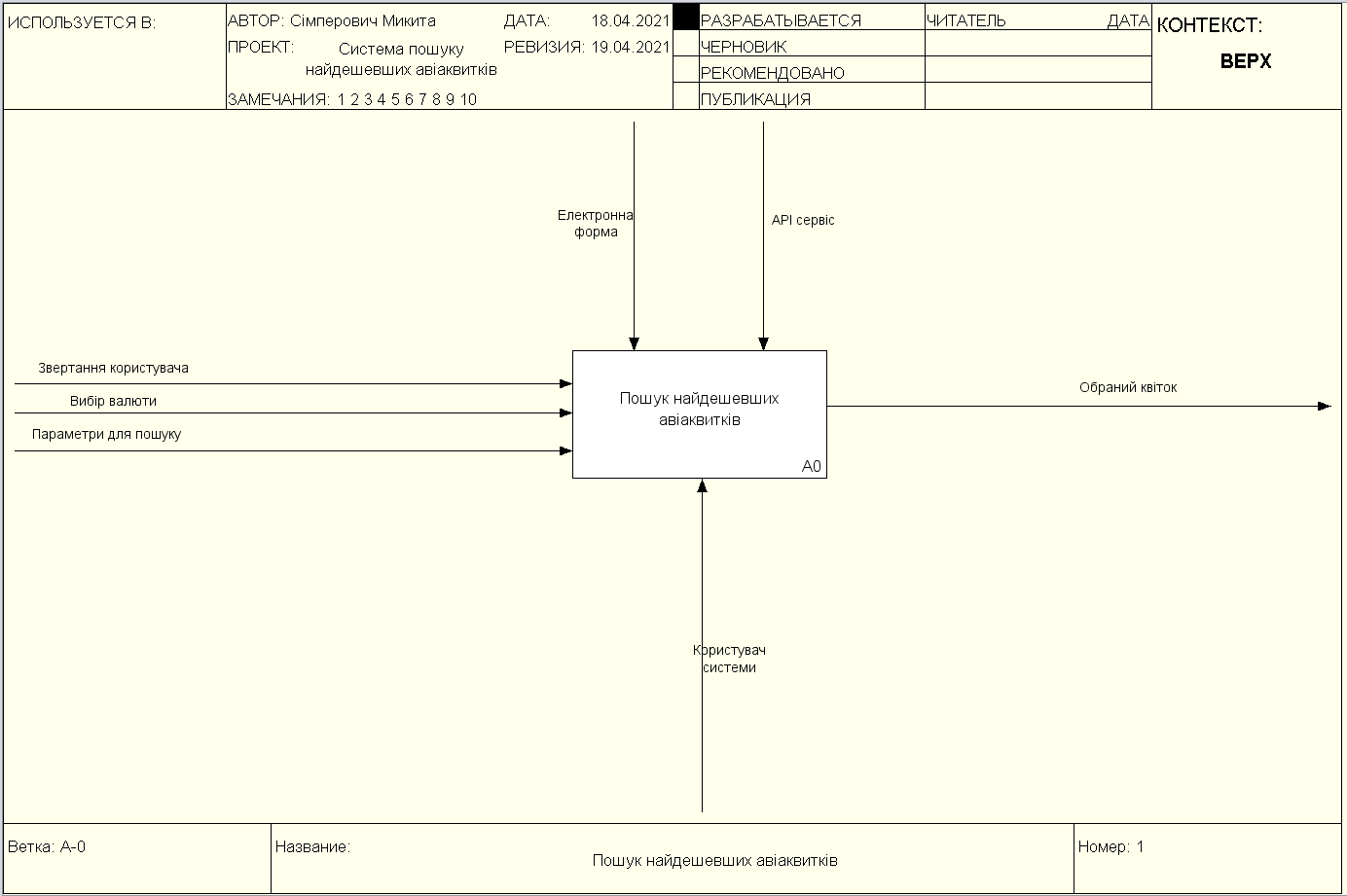


Рисунок 2.1 – Контекстна діаграма системи пошуку найдешевших авіаквитків.

Далі процес пошуку розбивається на три процеси:

* заповнення електронної форми;
* запит до API сервісу;
* вивід результатів пошуку.

Результат проектування представлено на діаграмі декомпозиції (рис.2.2).

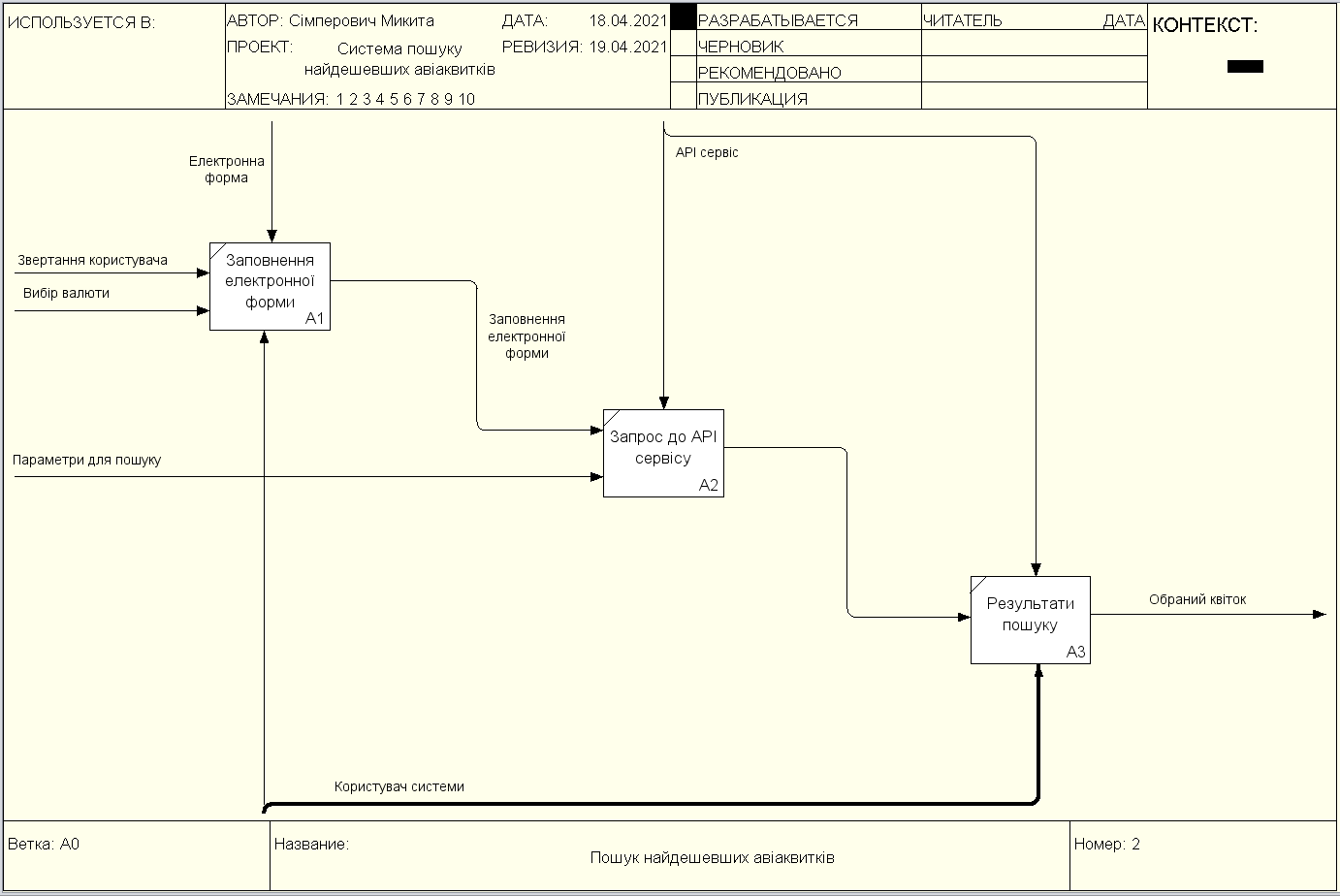


Рисунок 2.2 – Діаграма декомпозиції системи пошуку найдешевших авіаквитків

Функції користувача системи полягають у:

* введенні параметрів (дата вильоту, місце вильоту, місце прибуття, напрямок (в одному напрямку або прямий та зворотній)), для послання запиту на сервер;
* вибір валюти для пошуку;
* отримання та перегляд результатів пошуку;
* сортування отриманого результату за різними параметрами (ціна, дата, кількість пересадок),
* додавання та видалення квитків в/з обраного.

Діаграма варіантів використання показує відносини між акторами і прецедентами, описуючи систему на концептуальному рівні. В даному випадку існує один актор – користувач системи. Також на діаграмі представлені чотири прецеденти:

* введення параметрів для пошуку;
* вибір валюти;
* отримання результатів пошуку;
* додати білет в «Обране».

Діаграма варіантів використання веб сервісу пошуку найдешевших авіаквитків представлена на рис.2.3



Рисунок.2.3 – Діаграма використання веб сервісу пошуку найдешевших авіаквитків

В табл. 2.1 описані варіанти використання веб сервісу пошуку найдешевших квитків на авіарейси.

Таблиця 2.1 – Варіанти використання

|  |  |
| --- | --- |
| Варіанти використання | Опис |
| Вибір валюти | Вибрати валюти для пошуку найдешевших квитків на авіарейси. |

Продовження таблиці 2.1

|  |  |
| --- | --- |
| Занесення параметрів пошуку | Ввести параметри для пошуку:   * місце вильоту; * місце прибуття; * дата вильоту; * дата повернення |
| Отримати результат | Натискаючи на кнопку «Пошук» користувач отримує інформацію про існуючі рейси |
| Додати в обране | Вибрати білет та натиснути кнопку «Додати», зберегти його в блоці «Обране» |

## 2.2 Платформа та засоби розробки системи.

Оскільки система пошуку найдешевших авіаквитків є веб-додатком, то для реалізації дизайну сайту будуть використані: мова гіпертекстової розмітки HTML та каскадні таблиці стилів CSS. Для реалізації функціоналу додатку було вибрано мову програмування JavaScript.

Також використані інші засоби для розробки веб-додатків.

* Webpack – система збирання модулів для сучасних javascript додатків.
* Materialize – CSS фреймворк для створення сайтів, що ґрунтується на принципах material design, тобто це набір інструментів для верстки.
* Axios – JavaScript-бібліотека. Вона являє собою HTTP-клієнт, заснований на Проміс і призначений для браузерів і для Node.js.
* Visual Studio Code – редактор коду, використовується для побудови та налагодження сучасних веб-додатків та хмарних додатків.
* API Service –інтерфейс взаємодії з сервером БД. Використовується для відправки запитів на сервер та отримання відповіді.
* GitHub – система управління проектами і версіями програмного коду.

## 2.2 Архітектура проекту

Архітектура проекту складається з наступних сутностей.

* **Server** для відправки запитів на сервер та отримання даних.
* **API Service** призначений для комунікації з сервером – містить набір методів для отримання даних з серверу до яких буде звертатися клієнтська частина.
* **API.JS** – це клієнтська частина, що взаємодіє з сервісною частиною.
* **Config** – являє собою файл, який містить налаштування, що взаємодіють з API Service та API.JS.
* **Plugins** – набір спеціальних плагінів, які можуть підключатися до API Service та до клієнтської частини.
* **Stora** – сховище даних, яке необхідне для того щоб будь-яка частина додатку мала єдине джерело даних з якого будуть братися дані, наприклад, в нашому випадку це дані про міста, країни, авіакомпанії та інформація про квитки. Кожний компонент додатку матиме доступ до цих даних, та отримуватиме завжди оновлені дані.

Класи **View** для формування розмітки сайту. Їх буде декілька.

* Form – для вводу даних.
* Currency – для роботи з валютою.
* Tickets – для роботи зі списком квитків.
* Favorites – для роботи з обраними квітками.

Схема архітектури проекту зображена на рис. 2.1.



Рисунок 2.1 – Архітектура проекту «Веб сервіс пошуку найдешевших квитків на авіарейси»

# РОЗДІЛ 3

# Розробка веб-сервісу ПОШУКУ НАЙДЕШЕВШИХ КВИТКІВ НА авіарейсИ

## 3.1 Підключення та налаштування Webpack

**Webpack** – це система збирання модулів для сучасних JavaScript додатків.

Для підключення та налаштування Webpack спочатку завантажуємо і встановлюємо Node JS версії 10.16.1 LTS. Разом з Node JS встановлюється і NPM, який буде керувати пакетами.

Так само, для написання програмного коду встановимо Visual Studio Code – це редактор, який використовується при розробці хмарних і веб-додатків.

Запускаємо Visual Studio і підключаємо термінал getbash.exe для введення команд і для запуску установки пакетів. Запускаємо команду npm init. При запуску команди створюється Package.json – файл, який містить налаштування для пакетів. Він підтримує скрипти запуску різних команд.

В розділі devDependencies перераховані ті пакети, які потрібні для реалізації проекту. Програмний код файлу Package.json представлено у Додатку А.

Для установки пакетів, які вказані в файлі Package.json використовуємо команду npm i.

При установці пакетів створюється папка node\_modules.

Після установки пакетів необхідно виконати налаштування проекту в файлі webpack.config.js. Програмний код файлу webpack.config.jsпредставлено в Додатку Б.

Запускаємо Webpack (команда npm run build). Після цього створюється папка dist, яка містить файли app.jsтаindex.html.

Запускаємо dev-server командою npm run build dev-server.

## 3.2 Підключення платформи Materialize

Materialize – це CSS-фреймворк, який ґрунтується на принципах матеріального дизайну, розробленого компанією Google.

Ця програмна платформа була обрана через те, що вона має доволі великий набір плагінів, а саме autocomplet – це можливість пошуку по контексту, що буде зручно при пошуку країн та міст, також тут є елементи autocomplet для вибору дати та інші корисні плагіни.

Встановлюємо додатковий плагін mini-css-extract-plugin. Для цього в командному рядку набираємо команду: npm i–save dev mini-css-extract-plugin

Встановлюємо Materialize Sass файли в рамках проекту, команда – npm i--save materialize-css @ next.

Підключаємо його до проекту. Для цього створимо папку **Plugins**, в ній створимо два файли index.js та materialize.js. До materialize.js підключаємо модулі:



Тепер імпортуємо materializeв файл index.js:



Робимо імпорт в app.js:



## 3.3 Розробка дизайну проекту

Для зручності маніпулювання елементами створимо стилі для елементів форми у файлі style.css. Програмний код файлу style.css представлено в Додатку В.

У файлі index.html (Додаток Д) налаштуємо розміщення елементів проекту. В шапці проекту розмістимо.

1. Блок вибору валюти пошуку авіаквитків.
2. Логотип сервісу

В основній частині екрану розмістимо форму, яка міститиме наступні поля та елементи:

* autocomplete для вибору місця відправлення;
* autocomplet для вибору місця прибуття;
* datepickers для вибору дати відправлення;
* datepickers для вибору дати повернення;
* дві кнопки – «Вибір» та «Відміна».

Для цього в файлі index.html створимо клас «from\_section». В цьому класі додамо елементи форми та налаштуємо дизайн форми. Програмний код файлу index.html представлено в Додатку Д.

Для того, щоб працював autocomplete (плагін для автоматичного заповнення полів) його треба ініціалізувати у файлі materialize.js.Також в файліmaterialize.jsтреба зробити ініціалізацію об’єктівdatapickers(елемент для вибору дати),select(вибір валюти).

Програмний код файлу materialize.js представлено у Додатку Г.

На рис. 3.1 представлено зовнішній вигляд вікна веб-сервісу пошуку найдешевших квитків на авіарейси.

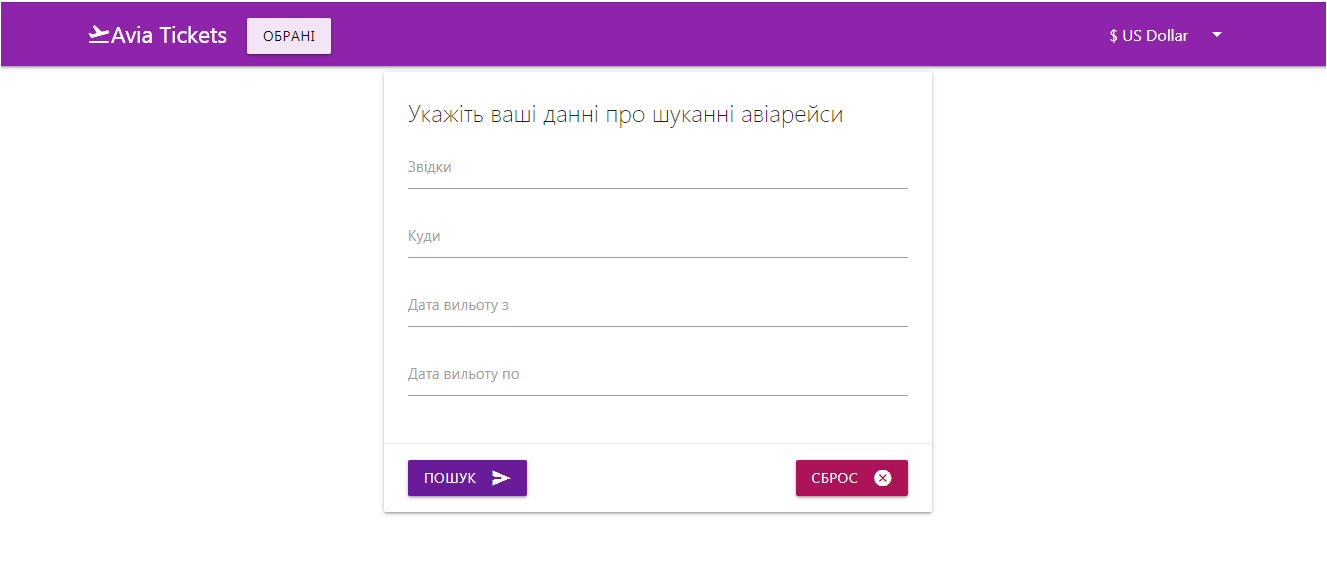
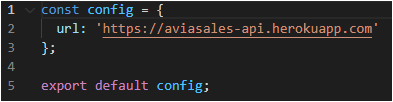


Рисунок 3.1 – Сторінка веб-сервісу пошуку найдешевших квитків на авіарейси

## 3.4 Створення функціоналу для взаємодії з сервером

Для виконання API запитів до серверу встановимо плагін Axios. Створимо папку config та файл appConfig.js, де вкажемо адресу API серверу куди будуть посилатися запити і зробимо експорт конфігурації config, для того щоб ці налаштування були доступні у всіх модулях проекту:



В папці services створимо файл appServise.js з набором методів для взаємодії з сервером. Для цього знадобіться плагін Axios та адреса серверу, тому спочатку в цей клас треба зробити імпорт Axios та імпорт налаштувань із файлу appConfig.js:



Далі створимо клас Api, в якому буде constructor, який буде приймати config з властивістю:



Та створимо екземпляр цього класу:



Далі треба зробити експорт створеного класу Api, щоб він був доступний всім модулям проекту:



Для сервісу пошуку найдешевших квитків на авіарейси потрібна буде наступна інформація: ціни на квітки по визначеним датам, країни та міста (вильоту та прибуття). На сервері є для цього наступні сутності:

* Countries – повертає масив країн;
* Cities – повертає масив місць;
* Prices/cheap – повертає масив рейсів та ціни на квіток;
* Airlines – повертає авіакомпанії.

Для отримання цих даних створимо методи:

* Countries() *–* повертає масив країн;
* Cities() *–* повертає масив міст;
* Airlines() *–* повертає масив авіакомпаній;
* Price() *–* повертає масив з інформацією про наявні квітки та ціни.

Програмний код файлу appServise.js представлено в Додатку Ж.

Метод Countries() повернув масив, який має дані, відображені в табл. 3.1

Таблиця 3.1 – Структура масиву Countries

|  |  |  |
| --- | --- | --- |
| **Код** | **Найменування** | **Ключі** |
| cases | Найменування країни за відмінками |  |
| code | Код країни | Первинний ключ |
| currency | Код валюти |  |
| name | Найменування країни |  |
| name\_translations | Найменування країни на англійській мові |  |

Структура масиву City() представлена в табл. 3.2.

Таблиця 3.2 – Структура масиву City

|  |  |  |
| --- | --- | --- |
| **Код** | **Найменування** | **Ключі** |
| Cases | Найменування міста за відмінками |  |
| Code | Код міста | Первинний ключ |
| coordinates | Координати міста |  |
| countr\_code | Код країни | Зовнішній ключ до масиву Countries |
| Name | Найменування міста |  |
| name\_translations | Найменування міста на англійській мові |  |
| time\_zone | Часова зона |  |

Структура масиву Airlines() представлена в табл. 3.3

Таблиця 3.3 – Структура масиву Airlines

|  |  |  |
| --- | --- | --- |
| **Код** | **Найменування** | **Ключі** |
| Code | Код компанії | Первинний ключ |
| Name | Найменування компанії |  |
| name\_translations | Найменування компанії на англ. мові |  |

Структура масиву Prices() представлена в табл. 3.4

Таблиця 3.4 – Структура масиву Prices

|  |  |  |
| --- | --- | --- |
| **Код** | **Найменування** | **Ключі** |
| Airline | Код авіакомпанії | Первинний ключ |
| departure\_at | Дата та час вильоту с |  |
| destination | Код міста призначення | Зовнішній ключ до масиву Countries |
| expires\_at | Дата та час вильоту по |  |
| flight\_number | Номер рейсу |  |
| Origin | Код місця прибуття | Зовнішній ключ до масиву Countries |
| Price | Ціна |  |
| return\_at | Дата та час прибуття |  |
| transfers | Пересадки |  |

Дані з серверу повертаються у форматі JSON (рис. 3.2-3.5).



Рисунок 3.2 – Фрагмент масиву країн



Рисунок 3.3 – Фрагмент масиву міст

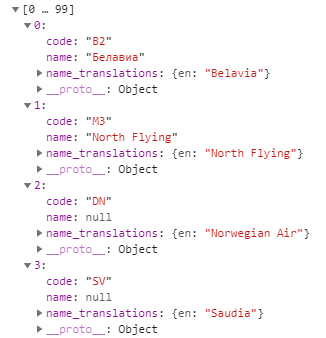


Рисунок 3.4 – Фрагмент масиву авіакомпаній



Рисунок 3.5 – Фрагмент масиву цін на квітки

## 3.5 Реалізація функціоналу проекту

Щоб форма працювала та видавала потрібні дані треба створити сховище, яке призначене для зберігання даних, які були отримані з API серверу. Це потрібно для того, щоб будь-яка частина додатку мала єдине джерело даних, з якого він буде брати поточні дані.

Для реалізації сховища, створимо папку Story, в ній – файл location.js. Сюди імпортуємо appServise, для отримання масиву країн, міст, авіакомпаній та інформації про квитки:



Саме сховище даних представлено у вигляді класу location, який приймає екземпляр класу App та зберігає отримані дані. Також в цьому класі будуть описані властивості countries, cities та інше.

В файлі location.js створимо метод init(), який приймає запити країн, міст та авіакомпанії з appServise.

Для роботи з даними створимо такі уявлення як валюта, міста і країни та список квитків, в яких будуть знаходитись дані у потрібному форматі.

Підготуємо дані, які будемо отримувати з API серверу. Для цього створимо метод getCityByKey(key), який буде формувати фільтр міст та повертати код міста. Метод getCityByСode(code)поверне найменування міста.

Створимо уявлення для країн. Для цього в файлі locations.js створимо функцію serializeCountries(countries), яка буде повертати країні.

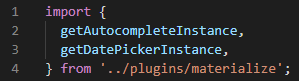
Для міст створимо спочатку метод getCountryNameByCode(code), який буде повертати країну за вказаним користувачем містом, потім створимо функцію serializeCities(cities), яка поверне міста.

Сформуємо список міст та країн вильоту/прибуття для об’єкту autocomplete – функція createShortCities(cities).

Програмний код файлу location.js представлено у Додатку З.

Для того, що б отримані з API серверу дані відображалися у формі в елементах autocomplete, створимо папку View в якій розмістимо файл form.js, в якому створимо клас FormUI. Цей клас буде управляти формую: отримувати саму форму, а також її елементи (місце вильоту, місце прибуття, дата вильоту з по).

Для формування списку міст та країн для об’єктів autocomplete та об’єктів datapickers (вибір дат), треба виконати імпорт функцій із materialize.js та передати в клас FormUI.



Для отримання даних з форми створені наступні методи (Додаток І):

* get form() – отримання самої форми;
* get originValue() – отримання даних про місце вильоту;
* get distinationValue() – отримання даних про місце прибуття;
* get departDateValue() – отримання дати вильоту;
* get returnDateValue() – отримання дати повернення.
* setAutocompleteData(data) – дати приймання.

В файл app.js імпортуємо модулі – FormUI.

Створимо єдине місце, де будемо ініціалізувати додаток, зробимо це на події DOM Content Loaded. Тут буде декілька методів для запуску додатку, які розділимо на методи подій та методи обробки (Додаток Е).

Після внесення відповідних змін в програмний код елементи форми autocompleteвідображають список міст та країн для вибору за вказаним користувачем контекстом (рис. 3.6, 3.7).

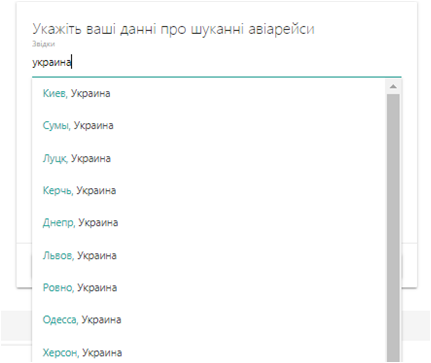


Рисунок 3.6 – Autocompleteпункт вильоту

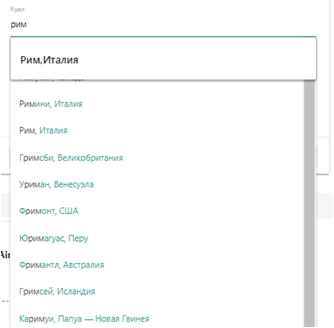


Рисунок 3.7 – Autocompleteпункт прибуття

Елементи форми datapickersвідображають календар для вибору дат (рис. 3.8).

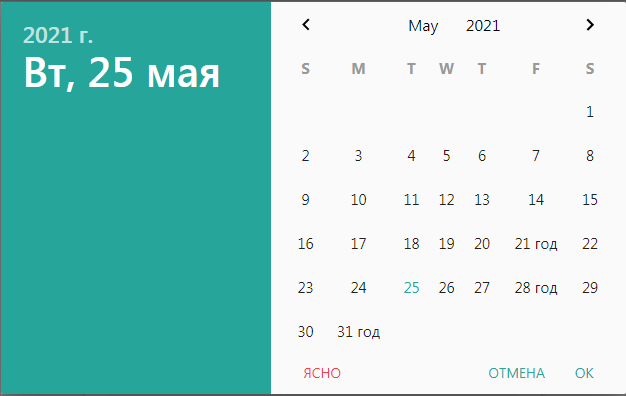


Рисунок 3.8 – Календар для вибору дати вильоту (об’єкт datapickers)

Для вибору валюти в папці View створимо файл currency.js (Додаток К), в якому розмістимо клас CurrencyUI, який буде управляти вибором валюти. Імпортуємо його в app.js(Додаток Е).

На рис. 3.9 відображено вибір валюти.

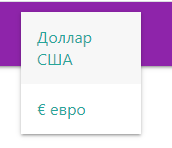


Рисунок 3.9 – Вибір валюти

## 3.6 Вивід авіаквитків на сторінку

Після отримання даних з форми та відправки запиту до API серверу, треба отримати дані в потрібному форматі про наявні на даний час квитки на авіарейси, а саме:

* логотип та назва авіакомпанії;
* місце вильоту;
* місце прибуття;
* дата та час вильоту;
* кількість пересадок;
* номер рейсу;
* ціна.

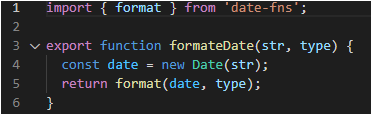
Для отримання інформації про авіакомпанію в файл appServise.js додамо метод airlines(), цей метод ініціалізується в сховище даних (Store). В файлі locatijnce.js є декілька методів для роботи з інформацією про авіакомпанії:

* getAirlineNameByCode() – отримання назви авіакомпанії;
* getAirlineLogoByCode() – отримання логотипу компанії;
* serializeAirlines() – повертає авіакомпанії в потрібному вигляді (викликається в init()).

Для отримання інформації про пункти вильоту/прибуття є методи:

* getCityByKey() – отримання коду міста вильоту/прибуття;
* getCityNameByCode() – отримання найменування міста вильоту/прибуття;
* serializeCities() – повертає місто вильоту/прибуття в потрібному вигляді.

Для перетворення дати в потрібний формат використаємо бібліотеку date-fns, яка містить різні функції для роботи з датами. Після установки плагіну date-fns в папці helpers розмістимо файл date.js, де створимо функцію, яка буде перетворювати дату зі строкового формату у формат date:



Після цього імпортуємо його в файл lacations.js:



Та передамо його в конструктор:



Метод serializeTikets() трансформує отримані дані в форматі потрібному для відображення на сторінці.

Для виведення підготовлених даних на сторінку створимо в папці views файл tickets.js, де розмістимо клас TicketsUI, який буде отримувати контейнер ‘tickets-sections’.

Також в цьому класі створимо методи:

* showEmptyMsg() – виводить повідомлення у випадку відсутності квитків за заданою умовою;
* emptyMsgTemplate() – генерує шаблон повідомлення;
* ticketTemplate(ticket, currency) – генерує шаблон одного квитка;
* renderTickets(tickets) – приймає та виводить масив квитків;
* clearContainer()– очищує контейнер перед новим пошуком.

На виході створимо екземпляр класу TicketsUI.:



Та експортуємо його:



Програмний код файлу tickets.js представлено в Додатку 11.

Щоб викликати створені у файлі tickets.js методи імпортуємо його до app.js:



Та викличемо метод renderTickets():



Результат пошуку квитків за вказаними параметрами відображено на рис.3.10.

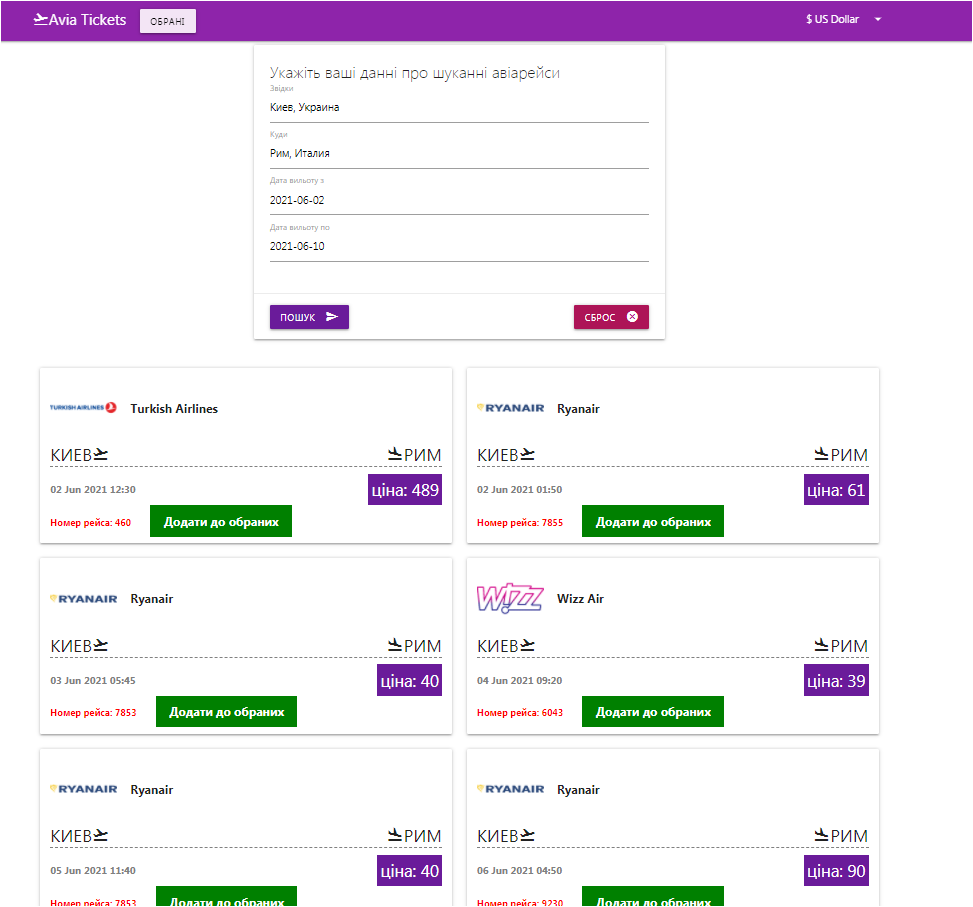


Рисунок 3.10 – Результат пошуку квитків на авіарейси

## 3.7 Реалізація функціоналу додавання квитків в обране

Для реалізації функціоналу додавання квитків в обране додамо кнопку «Обране» до форми виводу квитків та у файлі app.js (Додаток Е) налаштуємо обробку події натискання на кнопку «Додати в обране». В папці Stor створимо файл favorites.jsдля отримання об’єкту з даними квітка по ID (Додаток М). Для відображення даних обраного квітка в «Обране» створимо в папці Views файл favorites.js де розмістимо клас FavoritesUI, який буде отримувати контейнер ‘dropdown1’ та метод renderFavList(items), який виведе інформацію обраного квітка (Додаток Н).

На рис 3.11 зображено блок «Обрані»

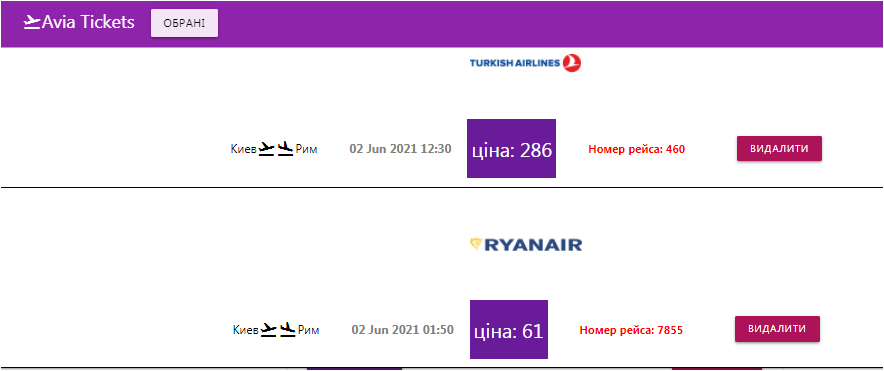


Рисунок 3.11 – Обрані квитки

# РОЗДІЛ 4

# ТЕСТУВАННЯ

## 4.1 Вибір виду тестування

Тестування є заключним етапом розробки програмного продукту, що відіграє важливу роль у процесі створення якісного програмного забезпечення. Техніка тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Оскільки число можливих тестів, навіть для нескладних програмних компонент, практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Тестування ПЗ поділяється на велику кількість видів та класифікується за багатьма ознаками. Перш за все тестування ПЗ буває ручне та автоматизоване. Ручне тестування проводиться безпосередньо людиною, яка натискає на кнопки в додатку. При цьому має місце ймовірність помилки за рахунок людського фактора, наприклад помилки або пропуску кроків в тестовому сценарії. Автоматизовані тести, з іншого боку, виробляються машиною, яка запускає тестовий сценарій, який був написаний заздалегідь. Такий спосіб вважається більш надійним, проте його працездатність залежить від того наскільки скрипт для тестування був добре написаний. Однак в ручному тестуванні все одно є своя цінність [29].

За типами тести бувають – модульні, інтеграційні, функціональні, приймальні, тести продуктивності, димове тестування.

Розроблений проект треба насамперед протестувати на функціональність. В основному для функціонального тестування використовуються ручні тести, адже їх легше адаптувати під потрібні цілі і завдання. До того ж, ручне тестування може виявити дефекти, які не передбачалися, та побачити те, що могло бути не передбачено тестовими сценаріями, тому для тестування розробленого сервісу скористаємося ручним тестуванням, застосовуючи як негативний, так й позитивний підхід.

## 4.2 Тест план

Для тестування веб-сервісу пошуку найдешевших квитків на авіарейси розробимо тест-план та визначимо функції, які повинні бути протестовані.

Функції, що повинні бути протестовані за тест-планом.

* Заповнення та очищення форми пошуку.
* Зміна валюти.
* Результат пошуку квитків на авіарейси.
* Додавання та видалення квитків з обраного.

Розробимо тест-кейси, які необхідно перевірити (Таблиця 4.1).

Таблиця 4.1 – Тест кейси для сторінки веб-сервісу пошуку найдешевших квитків на авіарейси

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості вибору пункту відправки та пункту призначення за контекстом країни або міста | |
| Передумова | | Завантажена сторінка веб-сервісу пошуку найдешевших квитків на авіарейси | |
| № | Дія | | Очікуваний результат |
| 1 | В полі «Пункт відправки» ввести назву країни, наприклад «Україна», або декілька букв з назви країни. | | З’являється список міст за вказаною країною. |
| 2 | В полі «Пункт відправки» ввести декілька букв з назви міста. | | З’являється список міст та країн за вказаним контекстом. |
| 3 | В полі «Пункт призначення» ввести назву країни, наприклад «Італія», або декілька букв з назви країни. | | З’являється список міст за вказаною країною. |
| 4 | В полі «Пункт призначення» ввести декілька букв з назви міста. | | З’являється список міст та країн за вказаним контекстом. |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Опис | | | Перевірка можливості вибору дати вильоту та дати прибуття | | |
| Передумова | | | Завантажена сторінка веб-сервісу пошуку найдешевших квитків на авіарейси | | |
| № | Дія | | | Очікуваний результат | |
| 1 | Тиснути на полі «Дата вильоту» | | | З’являється календар | |
| 2 | У календарі вказати дату та натиснути на кнопку «ОК» | | | Поле «Дата вильоту» заповнено датою у форматі YYYY-MM-DD | |
| 3 | Тиснути на полі «Дата прибуття» | | | З’являється календар | |
| 4 | У календарі вказати дату та натиснути на кнопку «ОК» | | | Поле «Дата прибуття» заповнено датою у форматі YYYY-MM-DD | |
| Опис | | | Перевірка можливості пошуку квитків на авіарейси | | |
| Передумова | | | Завантажена сторінка веб-сервісу пошуку найдешевших квитків на авіарейси | | |
| № | Дія | | | Очікуваний результат | |
| 1 | Заповнені всі поля у формі. Натиснути на кнопку «Пошук» | | | Нижче форми з’являється інформація про наявні квитки по вказаному напрямку | |
| 2 | Заповнені всі поля форми.  Змінити валюту за замовчуванням та натиснути на кнопку «Пошук» | | | У відображеній інформації змінилась ціна | |
| Опис | | Перевірка можливості додавання та видалення квитка в/з обраного | | | |
| Передумова | | Завантажена сторінка веб-сервісу пошуку найдешевших квитків на авіарейси та відображені результати пошуку | | | |
| № | Дія | | | | Очікуваний результат |
| 1 | Натиснути на кнопку «Додати в Обране» | | | | Обраний квіток з’являється у списку «Обране» |
| 2 | У списку «Обране » натиснути на кнопку «Видалити» | | | | Квиток видалений із списку «Обране» |

## 4.3 Введення в експлуатацію

Після того, як веб-сервіс буде готовий, його можна розгорнути, завантаживши файли і структуру каталогів системи на віддалений хостинг.

Є декілька способів розгортання проекту. Для розгортання цього проекту скористаємося системою управління проектами та версіями GitHub.

Посилання на розгорнутий проект – https://nonamev.github.io/avia-Sales/dist/index.html#.

# ВисновкИ

Мета дипломної роботи полягала у розробці веб-сервісу по пошуку найдешевших квитків на авіарейси та скорочення часу на пошук найвигіднішого варіанту, що забезпечує фундамент для подальшої розробки та вдосконалення веб-сервісу пошуку та придбання авіаквитків.

В даній роботі проведено огляд та аналіз існуючих сервісів по пошуку квитків на авіарейси, перелічені їх характеристики, переваги та недоліки. Виконано аналіз вимог, проектування та реалізація програмних засобів, які надають користувачеві можливість реалізувати всі функції програми. В результаті виконання роботи було розроблено веб-сервіс по пошуку найдешевших квитків на авіарейси, який автоматизує процес пошуку авіаквитків для пересування з точки А в точку Б серед множини пропонованих варіантів різних компаній та агентств, що значно скорочує час на пошук найвигіднішого варіанту.

Програма надає користувачеві наступні основні функції:

* введення параметрів для пошуку;
* вибір валюти для пошуку;
* отримання та перегляд результатів пошуку;
* сортування отриманого результату за різними параметрами (ціна, дата, кількість пересадок);
* додавання та видалення квитків в/з обраного.

В ході тестування розробленого інтерфейсу було продемонстровано, що система виконує всі поставлені задачі відповідно з сформованими вимогами та демонструє коректну роботу системи.

Програмний код проекту написано з застосування техніки ООП, що дозволяє легко змінювати програмний код для подальшої модифікації продукту та внесення змін. В майбутньому планується удосконалення та розвиток розробленого сервісу та розробка мобільної версії додатку.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Огляд пошукових систем авіаквитків. [Електронний ресурс]. Режим доступу: <https://euroroaming.ru/lajfhaki-dlya-puteshestvennikov/obzor-poiskovyh-sistem-aviabiletov/>
2. Euroavia. Популярні системи бронювання авіаквитків. [Електронний ресурс]. Режим доступу: <https://www.euroavia.ru/blog/3356-populjarnye-sistemy-bronirovanija-aviabiletov-osobennosti>.
3. Що таке метапошук авіаквитків. [Електронний ресурс]. Режим доступу: https://vc.ru/flood/32372-chto-takoe-metapoisk-aviabiletov-otvet-komandy-aviasales-na-razbor-poletov-ot-chitatelya.
4. Як працюють сервіси по пошуку дешевих авіаквитків. [Електронний ресурс]. Режим доступу: <https://ain.ua/2019/02/13/kak-rabotayut-servisy-dlya-poiska-deshevyx-aviabiletov>.
5. ТОП – 10 сайтів на яких можна шукати найдешевші авіаквитки. [Електронний ресурс]. Режим доступу: <https://investbro.ru/luchshie-sajjty-po-prodazhe-deshevykh-aviabiletov>.
6. Skyscanner. [Електронний ресурс]. Режим доступу: https://sites.google.com/site/stuchintel/najbils-posireni-riznovidi-botiv/skyscanner
7. [Tripmydream](https://intuit.ru/studies/courses/690/546/info). [Електронний ресурс]. Режим доступу: <https://tripmydream.ua/flights>
8. Онлайнбілет. [Електронний ресурс]. Режим доступу: <https://aviabilet.guru>.
9. youroute. [Електронний ресурс]. Режим доступу: <http://avia.youroute.ru>
10. OneTwoTrip. [Електронний ресурс]. Режим доступу: http://onetwotrip.com/ru
11. OZON.travel. [Електронний ресурс]. Режим доступу: <http://ozon.travel>.
12. Tutu. [Електронний ресурс]. Режим доступу: http://tutu.ru.
13. Петюшкин, А. HTML экспресс-курс / А. Петюшкин. – М.: СПб: БХВ-Петербург, 2016. - 250 c.
14. Дебольт. HTML и CSS. Совместное использование / Дебольт, Вирджиния. - М.: НТ Пресс, 2017. - 512 c.
15. Прохоренок, Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н.А. Прохоренок, В.А. Дронов. – Москва: СПб. [и др.]: Питер, 2015. - 768 c.
16. Фримен Эрик. Изучаем программирование на JavaScript / Фримен Эрик, Робсон Элизабет - ,2015. – 656 c.
17. Дєвид Флєнаган. JavaScript полное руководство/ Дєвид Флєнаган. – Диалектика – Киев, 2021. – 722с.
18. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированный анализ и проектирование с примерами приложений 3-е приложение. Вильямс, 2020. – 720 с.
19. Зыков, С. В. Программирование. Объектно-ориентированный подход : учебник и практикум для академического бакалавриата / С. В. Зыков. – М. : Издательство Юрайт, 2019. — 155 с..
20. Сысолетин, Е. Г. Разработка интернет-приложений : учеб. пособие для вузов / Е. Г. Сысолетин, С. Д. Ростунцев ; под науч. ред. Л. Г. Доросинского. — М. : Издательство Юрайт, 2019. — 90 с.
21. [Електронний підручний по HTML, CSS, JS](https://intuit.ru/studies/courses/690/546/info) . [Електронний ресурс]. Режим доступу: https://www.w3schools.com/css/default.asp
22. Самоучитель HTML, CSS [Електронний ресурс]. Режим доступу: http://htmlbook.ru/
23. Електронний підручник по JavaScript. [Електронний ресурс]. Режим доступу: <https://learn.javascript.ru/>.
24. WebForMySelf. Все о создании сайтов. [Електронний ресурс]. Режим доступу: <https://learn.javascript.ru/>
25. Современный интерфейсный фреймворк на основе Material Design. [Електронний ресурс]. Режим доступу: https://materializecss.com/
26. Создай свой собственный Aviasales с блэкджеком и фичами. [Електронний ресурс]. Режим доступу: <https://www.aviasales.ru/API>
27. Веб-технологии для разработчиков. [Електронний ресурс]. Режим доступу: <https://developer.mozilla.org/ru/docs/Web>.
28. Способи тестування програмного забезпечення. [Електронний ресурс]. Режим доступу: https://habr.com/ru/company/otus/blog/443418/

# ДОДАТОК А PACKAGE.JSON

{

"name": "lesson-54-intro-webpack",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"dev": "webpack-dev-server",

"build": "webpack"

},

"keywords": [],

"author": "",

"license": "ISC",

"devDependencies": {

"@babel/core": "^7.5.5",

"@babel/preset-env": "^7.5.5",

"autoprefixer": "^9.6.1",

"babel-core": "^6.26.3",

"babel-loader": "^8.0.6",

"babel-polyfill": "^6.26.0",

"css-loader": "^3.1.0",

"file-loader": "^4.1.0",

"html-webpack-plugin": "^3.2.0",

"postcss-loader": "^3.0.0",

"precss": "^4.0.0",

"style-loader": "^0.23.1",

"webpack": "^4.39.1",

"webpack-cli": "^3.3.6",

"webpack-dev-server": "^3.7.2"

},

"dependencies": {}

}

# ДОДАТОК Б WEBPACK.CONFIG.JS

const path = require(‘path’);

const autoprefixer = require(‘autoprefixer’);

const precss = require(‘precss’);

const HtmlWebpackPlugin = require(‘html-webpack-plugin’);

const MiniCssExtractPlugin = require(‘mini-css-extract-plugin’);

module.exports = {

entry: {

polyfill: ‘babel-polyfill’,

app: ‘./js/app.js’,

},

context: path.resolve(\_\_dirname, ‘src’),

devServer: {

publicPath: ‘/’,

port: 9000,

contentBase: path.join(process.cwd(), ‘dist’),

host: ‘localhost’,

historyApiFallback: true,

noInfo: false,

stats: ‘minimal’,

hot: true,

},

module: {

rules: [

{

use: {

loader: ‘babel-loader’,

options: {

presets: [‘@babel/preset-env’],

},

},

test: /\.js$/,

},

{

test: /\.css$/,

use: [

{

loader: MiniCssExtractPlugin.loader,

},

{

loader: ‘css-loader’,

options: {

importLoaders: 1,

sourceMap: true,

},

},

{

loader: ‘postcss-loader’,

options: {

plugins: () => [precss, autoprefixer],

},

},

],

},

{

test: /\.(png|jpe?g|gif)$/,

use: [

{

loader: ‘file-loader’,

options: {

name: ‘[path][name].[ext]’,

},

},

],

},

],

},

plugins: [

new MiniCssExtractPlugin({ filename: ‘./style.css’ }),

new HtmlWebpackPlugin({

template: ‘index.html’,

}),

],

output: {

path: path.resolve(\_\_dirname, ‘dist’),

filename: ‘[name].[hash].js’,

},

mode: ‘development’,

};

# ДОДАТОК В

# style.css

form { margin-top: 70px;}

.d-flex { display: flex;}

.hide-on-med-and-down { display: block !important;}

.align-items-center { align-items: center;}

.justify-items-center { justify-items: center;}

.justify-items-between { justify-content: space-between;}

.ml-auto { margin-left: auto;}

.mr-auto { margin-right: auto;}

.mx-auto { margin-left: auto; margin-right: auto;}

.header .brand-logo { position: static;}

.header .currency-select input.select-dropdown { margin: 0; color: #fff; border-bottom: none;}

.header .currency-select input.select-dropdown:focus { border-bottom: none;}

.header .currency-select { width: 120px;}

.header .currency-select .caret { fill: #fff;}

.header .brand-logo { font-size: 1.5rem;}

.ticket-card { padding: 10px 15px;}

.ticket-airline-name { font-weight: bold; font-size: 18px; text-transform: capitalize;}

.ticket-airline-img { max-width: 100px; margin-right: 20px;}

.ticket-destination { margin-bottom: 10px; border-bottom: 1px dashed gray;}

.ticket-destination .ticket-city { font-size: 26px; text-transform: uppercase;}

.ticket-destination i.material-icons { font-size: 26px;}

.ticket-time-departure { font-weight: bold; color: gray;}

.ticket-price { font-size: 25px; padding: 5px; background: #6a1b9a; color: #fff;}

.ticket-additional-info { font-size: 14px; font-weight: bold;}

.tickets-empty-res-msg {padding: 10px 15px; text-align: center; background: whitesmoke; font-weight: bold; font-size: 18px;}

#dropdown1 { opacity: 1; display: none; margin-top: 65px; width: 100%; max-height: 400px; overflow-y: scroll;}

.favorite-item {align-items: center; justify-content: space-around; color: #000; flex-direction: column; border-bottom: 2px solid black; padding-bottom: 10px;}

.favorite-item img { width: 140px;}

.favorite-item-info { align-items: center; width: 100%; justify-content: center;}

.favorite-item-info > \* { margin-left: 20px; margin-right: 20px;}

.ticket-time-departure { margin-right: 20px;}

.ticket-flight-number { color: red; padding-right: 25px;}

.dropdown-trigger { margin-left: 20px;}

.add-to-fav { padding: 10px 20px; background-color: green; display: inline-block; color: #fff; font-size: 18px; cursor: pointer;}

.add-to-fav { transition: 0.4s;}

.add-to-fav:hover { background-color: #6a1b9a;}

nav { position: fixed; z-index: 5;}

# ДОДАТОК Г

# materialize.js

import ‘materialize-css/dist/css/materialize.min.css’;

import ‘materialize-css/dist/js/materialize.min.js’;

*//ініціалізація селекта*

const select = document.querySelector(‘select’);

M.FormSelect.init(select);

*//експортуємо функцію яка отримувмує вибраний елемент в селекти*

export function getInstanseSelect(el){

return M.FormSelect.getInstanse(el);

}

*//ініціалізація полей з автопідбором*

const autocompletes = document.querySelectorAll(‘.autocomplete’);

M.Autocomplete.init(autocompletes, {

data: {

"Apple": null,

"Microsoft": null,

"Google": ‘https://placehold.it/250x250’

},

});

*//експортуємо функцію яка отримувмує введенні данні з полей із автопідбором*

export function getAutocompleteInstanse(el){

return M.Autocomplete.init(el);

}

*//ініціалізація календарів*

const datePickers = document.querySelectorAll(‘.datepicker’);

M.Datepicker.init(datePickers,{

showClearBtn: true,

format: ‘yyyy-mm-dd’

});

*//експортуємо функцію яка отримувмує введенні дати з календарів*

export function getDatepickerInstanse(el){

return M.Datepicker.init(el, {

showClearBtn: true,

format: ‘yyyy-mm-dd’

});

}

# ДОДАТОК Д

# index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<meta http-equiv="X-UA-Compatible" content="ie=edge" />

<link

href="https://fonts.googleapis.com/icon?family=Material+Icons"

rel="stylesheet"

/>

<title>Document</title>

</head>

<body>

<nav class="header purple darken-1">

<div class="nav-wrapper">

<div class="container d-flex align-items-center">

<div class="logo d-flex align-items-center">

<i class="large material-icons">flight\_takeoff</i>

<a href="#" class="brand-logo">Avia Tickets</a>

</div>

<div class="favorites">

<!-- Dropdown Trigger -->

<a

class="dropdown-trigger btn purple lighten-5 black-text"

href="#"

data-target="dropdown1"

>Обрані</a

>

<!-- Dropdown Structure -->

<div id="dropdown1" class="dropdown-content">

</div>

</div>

<ul id="nav-mobile" class="right hide-on-med-and-down ml-auto">

<li>

<div class="input-field currency-select">

<select id="currency">

<option value="USD">$ US Dollar</option>

<option value="EUR">€ Euro</option>

</select>

</div>

</li>

</ul>

</div>

</div>

</nav>

<div class="form-section">

<div class="container">

<div class="row d-flex">

<div class="col m6 mx-auto">

<div class="card">

<form name="locationControls">

<div class="card-content">

<span class="card-title">Укажіть ваші данні про шуканні авіарейси</span>

<div class="input-field">

<input

type="text"

id="autocomplete-origin"

class="autocomplete"

/>

<label for="autocomplete-origin">Звідки</label>

</div>

<div class="input-field">

<input

type="text"

id="autocomplete-destination"

class="autocomplete"

/>

<label for="autocomplete-destination">Куди</label>

</div>

<div class="input-field">

<input

type="text"

class="datepicker"

id="datepicker-depart"

/>

<label for="datepicker-depart">Дата вильоту з</label>

</div>

<div class="input-field">

<input

type="text"

class="datepicker"

id="datepicker-return"

/>

<label for="datepicker-return">Дата вильоту по</label>

</div>

</div>

<div class="card-action d-flex">

<button

class="btn waves-effect waves-light purple darken-3"

type="submit"

name="action"

>

Пошук

<i class="material-icons right">send</i>

</button>

<button

class="btn waves-effect waves-light pink darken-3 ml-auto"

type="reset"

name="action"

>

Сброс

<i class="material-icons right">cancel</i>

</button>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

<div class="tickets-sections">

<div class="container">

<div class="row">

</div>

</div>

</div>

</body>

</html>

# ДОДАТОК Е

# app.js

import ‘./plugins’;

import ‘../css/style.css’;

import locations from ‘./store/locations’;

import formUI from ‘./views/form’;

import ticketsUI from ‘./views/tickets’

import currencyUI from ‘./views/currency’;

import favorites from ‘./store/favorites’

import favoritesUI from ‘./views/favorites’

document.addEventListener(‘DOMContentLoaded’, () => { *//обробчик події загрузки сторінки*

const form = formUI.form;

initApp();

//Події

document.addEventListener(‘click’, ({target})=>{

if(target.innerHTML === ‘Обрані’ || target.innerText === ‘ВИДАЛИТИ’){

document.querySelector(‘#dropdown1’).style.display = ‘block’;

}else{

document.querySelector(‘#dropdown1’).style.display = ‘none’;

}

})

document.addEventListener(‘click’, (e) => { *//обробчик події нажатої кнопки додавання білету з вибраних*

if(e.target.innerHTML == ‘Додати до обраних’){

const ticketFavObj = {

logo: e.target.parentElement.parentElement.querySelector(‘.ticket-airline-img’).currentSrc,

from: e.target.parentElement.parentElement.querySelectorAll(‘.ticket-city’)[0].innerHTML,

to:e.target.parentElement.parentElement.querySelectorAll(‘.ticket-city’)[1].innerHTML,

date: e.target.parentElement.parentElement.querySelector(‘.ticket-time-departure’).innerHTML,

price: e.target.parentElement.parentElement.querySelector(‘.ticket-price’).innerHTML,

number: e.target.parentElement.parentElement.querySelector(‘.ticket-flight-number’).innerHTML

}

console.log(e.target.parentElement.parentElement.querySelectorAll(‘.ticket-price’));

favorites.addNewItem(ticketFavObj);

favoritesUI.renderFavList(favorites.items);

}

})

document.addEventListener(‘click’, (e) => { *//обробчик події нажатої кнопки видалення білету з вибраних*

if(e.target.innerText == ‘ВИДАЛИТИ’){

e.target.parentElement.parentElement.style.display = ‘none’;

}

})

form.addEventListener(‘submit’, (e) => { *//обробчик події відправки форми*

e.preventDefault();

onFormSubmit();

})

async function initApp(){

await locations.init();

formUI.setAutocompleteData(locations.shortCities);

}

async function onFormSubmit(){

*//функція яка збтраї введенні в форму данні на відправляє на сервер*

const origin = locations.getCityCodeByKey(formUI.originValue); //отримуємо код міста для зручної відправки на сервер

const destination = locations.getCityCodeByKey(formUI.destinationValue);

const depart\_date = formUI.departValue.slice(0, -3);

const return\_date = formUI.returnValue.slice(0, -3) ? formUI.returnValue.slice(0, -3) : ‘‘;

const currency = currencyUI.CurrencyValue;

await locations.fetchTickets({

origin,

destination,

depart\_date,

return\_date,

currency

});

ticketsUI.renderTickets(locations.lastSearch)

}

})

# ДОДАТОК Ж

# appServise.js

import config from ‘../config/apiConfig’

import axios from ‘axios’ // плагин полегшує реалізацію запитів на сервер

*//класс з набором методыв для взаємодії с сервером*

*// Ендпоінти сервера*

*// /countries -- массив країн*

*// /cities -- массив міст*

*// /prices/cheap -- массив с доступними авіарейсами*

class Api{

constructor(config){

this.url = config.url;

}

async countries() {

*// ця асинхронна функція виконує запит на сервер для отримання міст*

try {

const response = await axios.get(`${this.url}/countries`);

return response.data;

} catch(err){

console.log(err);

return Promise.reject(err);

}

}

async cities(){

*// ця асинхронна функція виконує запит на сервер для отримання країн*

try {

const response = await axios.get(`${this.url}/cities`);

return response.data;

} catch(err){

console.log(err);

return Promise.reject(err);

}

}

async prices(params){

*// ця асинхронна функція виконує запит на сервер для отримання білетів*

*try {*

const response = await axios.get(`${this.url}/prices/cheap`, {

params

});

return response.data;

} catch(err){

alert(‘Введіть всі данні’)

return Promise.reject(err);

}

}

async airlines(params){

*// ця асинхронна функція виконує запит на сервер для отримання інформації про авіарейси*

try {

const response = await axios.get(`${this.url}/airlines`, {

params

});

return response.data;

} catch(err){

console.log(err);

return Promise.reject(err);

}

}

}

const api = new Api(config);

export default api

# ДОДАТОК З

# location.js

import api from ‘../services/apiService’;

import {formatDate} from ‘../helpers/date’;

class Locations {

constructor(api, helpers){

this.api = api;

this.countries = null;

this.cities = null;

this.airlines = null;

this.lastSearch = null;

this.formatDate = helpers.formatDate;

}

async init(){

*//метод одразу запрашуэ міста та країни у apiService*

const res = await Promise.all([

this.api.countries(),

this.api.cities(),

this.api.airlines()

])

const [countries, cities, airlines] = res; *//деструктуруємо повернений массив на два массива країн та міст*

this.countries = this.transformCountries(countries);

this.cities = this.transfromCities(cities);

this.airlines = this.transformAirlines(airlines);

this.shortCities = this.createShorCitiesList(this.cities);

return res;

}

getCitiesByCountryCode(code){

// отримання міст по коду країни

return this.cities.filter(city => city.country\_code === code)

}

transformCountries(countries){

*//функція яка трансформовує повернений сервером массив даних в зручний вигляд для відправки кода країни на сервер та отримання об’єкта цієї країни як відповідь від сервера*

// {‘Код країни’: {... об’єкт країни}}

return countries.reduce((acc, country)=>{

acc[country.code] = country;

return acc;

}, {})

}

transfromCities(cities){

*// функція яка трансформує массив с містами повернений сервером у зручний вигляд для автокомпліта*

*// так буде легше отмитати код країни з автокомпліта для відправки на сервер*

*// {‘Назва місто, Назвакраїни’: {... об’єкт міста}}*

return cities.reduce((acc,city) => {

const countryName = this.getCountryNameByCode(city.country\_code);

city.name ? city.name : city.name.translation.en;

const fullName = `${city.name}, ${countryName}`;

acc[city.code] = {

...city,

countryName,

fullName

};

return acc;

}, {})

}

transformAirlines(airlines){

*//функція що трансформує масив об’єктів з авілініями в об’єкт з об’єктами*

return airlines.reduce((acc, airline) => {

airline.logo = `http://pics.avs.io/200/200/${airline.code}.png`; // http://pics.avs.io -- сервіс по пошуку зображень логотипів авіакомпаній

airline.name = airline.name ? airline.name : airline.name\_translations.en;

acc[airline.code] = airline;

return acc;

}, {})

}

getCountryNameByCode(code){

*// функція що повертає назву країни по коду країни*

return this.countries[code].name || this.countries[code].name\_translations.en;

}

createShorCitiesList(cities){

*//функція яка створює список для автозаповнення*

return Object.entries(cities).reduce((acc, [, city]) => {

acc[city.fullName] = null;

return acc

}, {})

}

getCityCodeByKey(key){

*//функція отримує код міста по ключу*

const city = Object.values(this.cities).find((item) => item.fullName === key)

return city.code;

}

getCityNameByCode(code){

*//отримання назви міста по коду*

return this.cities[code].name;

}

getAirlineNameByCode(code){

*//функція отримує назву авіакомпанії по коду авіакомпанії*

return this.airlines[code].name ? this.airlines[code].name : ‘‘;

}

getAirlineLogoByCode(code){

*//функція отримує логотип авіакомпанії по коду авіакомпанії*

return this.airlines[code].name ? this.airlines[code].logo : ‘‘;

}

async fetchTickets(params){

const response = await this.api.prices(params);

this.lastSearch = this.transformTickets(response.data);

}

transformTickets(tickets){

*// функція преобразовує данні білетів до зручного предоставлення їх користувачу*

return Object.values(tickets).map(ticket => {

let obj = {

...ticket,

origin\_name: this.getCityNameByCode(ticket.origin),

destination\_name: this.getCityNameByCode(ticket.destination),

airline\_logo: this.getAirlineLogoByCode(ticket.airline),

airline\_name: this.getAirlineNameByCode(ticket.airline),

departure\_at: this.formatDate(ticket.departure\_at, ‘dd MMM yyyy hh:mm’),

return\_at: this.formatDate(ticket.return\_at, ‘dd MMM yyyy hh:mm’),

}

return obj;

})

}

}

const locations = new Locations(api, {formatDate});

export default locations;

# ДОДАТОК І

# form.js

import {getDatepickerInstanse, getAutocompleteInstanse} from ‘../plugins/materialize’

class FormUI{

constructor(DatepickerInstanse, AutocompleteInstanse){

//отримання та ініціалізація елементів форми

this.\_form = document.forms[‘locationControls’];

this.origin = document.getElementById(‘autocomplete-origin’);

this.destination = document.getElementById(‘autocomplete-destination’);

this.depart = document.getElementById(‘datepicker-depart’);

this.return = document.getElementById(‘datepicker-return’);

this.originAutocomplete = AutocompleteInstanse(this.origin);

this.destinationAutocomplete = AutocompleteInstanse(this.destination);

this.departDatepicker = DatepickerInstanse(this.depart);

this.returnDatepicker = DatepickerInstanse(this.return);

}

get form(){

*//гетер для форми*

return this.\_form;

}

get originValue(){

return this.origin.value;

}

get destinationValue(){

return this.destination.value;

}

get departValue(){

return this.departDatepicker.toString();

}

get returnValue(){

return this.returnDatepicker.toString();

}

setAutocompleteData(data){

*// змінна даних в полі автозаповнення*

this.originAutocomplete.updateData(data);

this.destinationAutocomplete.updateData(data);

}

}

const formUI = new FormUI(getDatepickerInstanse, getAutocompleteInstanse);

export default formUI;

# ДОДАТОК К

# currency.js

class CurrencyUI{

constructor(){

this.currency = document.getElementById(‘currency’);

}

get CurrencyValue(){

return this.currency.value;

}

}

const currencyUI = new CurrencyUI();

export default currencyUI;

# ДОДАТОК Л

# tickets.js

class TicketsUI{

constructor(){

this.container = document.querySelector(‘.tickets-sections .row’)

}

renderTickets(tickets){

this.clearContainer();

if(!tickets.length){

this.showEmptyMessage()

}else{

tickets.forEach(e => {

this.container.innerHTML += TicketsUI.ticketTemplate(e);

})

}

}

clearContainer(){

this.container.innerHTML = ‘‘;

}

showEmptyMessage(){

this.container.innerHTML = ‘<div class="tickets-empty-res-msg">По вашему запиту квитків не знайдено.</div>‘;

}

static ticketTemplate({airline\_logo, airline\_name, destination\_name, origin\_name, departure\_at, price, flight\_number, current\_id}){

return `<div class="col m12 l6">

<div class="card ticket-card" id=‘${current\_id}’>

<div class="ticket-airline d-flex align-items-center">

<img src="${airline\_logo}" class="ticket-airline-img">

<span class="ticket-airline-name">${airline\_name}</span>

</div>

<div class="ticket-destination d-flex align-items-center">

<div class="d-flex align-items-center mr-auto">

<span class="ticket-city">${origin\_name} </span>

<i class="medium material-icons">flight\_takeoff</i>

</div>

<div class="d-flex align-items-center">

<i class="medium material-icons">flight\_land</i>

<span class="ticket-city">${destination\_name}</span>

</div>

</div>

<div class="ticket-time-price d-flex align-items-center">

<span class="ticket-time-departure">${departure\_at}</span>

<span class="ticket-price ml-auto">ціна: ${price}</span>

</div>

<div class="ticket-additional-info">

<span class="ticket-flight-number">Номер рейса: ${flight\_number}</span>

<span class=‘add-to-fav’>Додати до обраних</span>

</div>

</div>

</div>`

}

}

const ticketsUI = new TicketsUI();

export default ticketsUI;

# ДОДАТОК М

# Store/favorites.js

import shortid from ‘shortid’; // пакет для генерації ID

class Favorites{

constructor(){

this.items = {};

}

addNewItem(ticket){

this.items[shortid.generate()] = ticket;

}

get favoritesItems(){

return this.items;

}

}

const favorites = new Favorites();

export default favorites;

# ДОДАТОК Н

# Views/favorites.js

class FavoritesUI{

constructor(){

this.container = document.querySelector(‘#dropdown1’);

}

renderFavList(items){

this.container.innerHTML = ‘‘;

Object.values(items).forEach(e => {

this.container.innerHTML += FavoritesUI.favItemTemplate(e);

})

}

static favItemTemplate(ticket){

return `<div class="favorite-item d-flex align-items-start">

<img

src="${ticket.logo}"

class="favorite-item-airline-img"

/>

<div class="favorite-item-info d-flex flex-column">

<div

class="favorite-item-destination d-flex align-items-center"

>

<div class="d-flex align-items-center mr-auto">

<span class="favorite-item-city">${ticket.from} </span>

<i class="medium material-icons">flight\_takeoff</i>

</div>

<div class="d-flex align-items-center">

<i class="medium material-icons">flight\_land</i>

<span class="favorite-item-city">${ticket.to}</span>

</div>

</div>

<div class="ticket-time-price d-flex align-items-center">

<span class="ticket-time-departure">${ticket.date}</span>

<span class="ticket-price ml-auto">${ticket.price}</span>

</div>

<div class="ticket-additional-info">

<span class="ticket-flight-number">${ticket.number}</span>

</div>

<a

class="waves-effect waves-light btn-small pink darken-3 delete-favorite ml-auto"

>Видалити</a

>

</div>

</div>`

}

}

const favoritesUI = new FavoritesUI();

export default favoritesUI;