МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА[1]

Ткача Владислава Вікторовича
(прізвище, ім'я, по батькові здобувача)

на тему **Розробка програмного забезпечення бронювання та продажу авіаквитків**

(повна назва теми)

за матеріалами **праць провідних спеціалістів з розробки ПЗ та проектування БД**

(повна назва бази дослідження)

науковий керівник **к.е.н., доцент** _____ **Ткаліченко С.В.**
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від _11.06.2025р._ № _12_
Завідувач кафедри _____
(підпис)

д.т.н., професор Зеленський О.С.
Наук. ступень, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

[1] Назва кваліфікаційної роботи відповідно до ОПП

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
«11» червня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ[2]

1. Тема роботи «Розробка програмного забезпечення бронювання та продажу авіаквитків»

Керівник роботи к.е.н., доцент Ткаліченко С.В.
затверджені наказом закладу вищої освіти від «04» квітня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «09» червня 2025 р.

3. Зміст кваліфікаційної роботи, об'єкт, предмет та мета дослідження:

Розділ 1. Постановка задачі

Розділ 2. Проєктування задачі

Розділ 3. Організація інформаційного забезпечення

Розділ 4. Розробка програмного забезпечення

Об'єкт дослідження: інформаційна система для бронювання авіаквитків

Предмет дослідження: організація процесу бронювання квитків на літаки

Мета кваліфікаційної роботи: *створення програмного забезпечення для бронювання та продажу авіаквитків.*

5. Дата видачі завдання «04» квітня 2025 р.

[2] Назва кваліфікаційної роботи відповідно до ОПП

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МДР	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	04.04.2025-13.04.2025	
2	Підготовка розділу 2	14.04.2025-26.04.2025	
3.	Підготовка розділу 3	27.04.2025-15.05.2025	
4	Підготовка розділу 4	16.05.2025-08.06.2025	
5	Реєстрація завершеної кваліфікаційної роботи	09.06.2025	Реєстраційний №___ «09»червня 2025 р.
6	Отримання відгуку від наукового керівника	10.06.2025	
7	Подання кваліфікаційної роботи на перегляд завідувачу кафедри	11.06.2025	
8	Отримання зовнішньої рецензії	12.06.2025	
9	Попередній захист кваліфікаційної роботи на кафедрі	13.06.2025	
10	Підготовка до захисту в ЕК	16.06.2025-21.06.2025	

Завдання підготував науковий керівник _____ Ткаліченко С.В
(підпис) (прізвище та ініціали)

Завдання одержав _____ Ткач В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

на кваліфікаційну бакалаврську роботу

«Розробка програмного забезпечення бронювання та продажу авіаквитків»

Ткача Владислава Вікторовича

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра зі спеціальності 121 «Інженерія програмного забезпечення» – Державний університет економіки і технологій – Кривий Ріг, 2025.

У бакалаврській дипломній роботі розроблено повнофункціональне програмне забезпечення для автоматизації процесів бронювання та продажу авіаквитків. Система забезпечує зручний інтерфейс користувача, підтримує реєстрацію та авторизацію, пошук авіарейсів за заданими критеріями, бронювання місць із візуалізацією, перегляд детальної інформації про рейси, а також адміністрування польотів. Програмний продукт реалізовано мовою програмування C# із використанням технології WPF, шаблону MVVM, бібліотеки OpenGL для відображення мапи рейсів за заданими координатами та Microsoft SQL Server для організації бази даних.

Ключові слова: бронювання, авіаквитки, програмне забезпечення, база даних, C#, WPF, MVVM, OpenGL.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД	База даних
ПЗ	Програмне забезпечення
СУБД	Система управління базою даних
MVVM	Model-View-ViewModel. Архітектурний патерн, який дозволяє чітко розділити код інтерфейсу та бізнес-логіку, спрощуючи тестування й супровід
XAML	Розмітка інтерфейсу
WPF	Windows Presentation Foundation

ЗМІСТ

ВСТУП

У сучасному світі стрімкого технологічного прогресу та глобалізованого способу життя авіаперельоти стали невід’ємною частиною мобільності людини. Водночас зростання попиту на авіаперевезення створює нові виклики, серед яких особливе значення має забезпечення ефективного та зручного процесу бронювання квитків. Саме системи бронювання авіаквитків відіграють ключову роль у наданні швидкого, доступного та надійного сервісу для організації авіаподорожей.

Створення інформаційної системи для бронювання авіаквитків є важливим і своєчасним завданням, що відповідає запитам як клієнтів, так і авіаперевізників у сучасних ринкових умовах. Запровадження такої системи сприятиме підвищенню конкурентоспроможності авіакомпаній, зробить процес подорожей зручнішим для пасажирів та стимулюватиме розвиток туристичної й авіаційної сфер загалом.

Метою цієї дипломної роботи є створення та дослідження інформаційної системи для бронювання авіаквитків, яка покликана спростити процес планування подорожей і підвищити зручність для користувачів. Об'єктом дослідження виступає інформаційна система бронювання, а предметом — особливості організації процесу бронювання авіаквитків та забезпечення його ефективності й зручності для клієнтів.

Науково-дослідна робота ґрунтується на системному підході із застосуванням методів структурного аналізу, моделювання, вивчення вимог користувачів та проведення експериментальних досліджень. У результаті буде створено ефективну інформаційну систему, що оптимізує процес бронювання авіаквитків та сприятиме підвищенню рівня задоволеності клієнтів авіакомпаній.

Ця робота покликана розв’язати актуальні завдання в сфері авіації та електронного бізнесу, сприяючи створенню ефективної та зручної системи бронювання авіаквитків, яка відповідатиме вимогам сучасного користувача.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ

1.1. Характеристика задачі

Формулювання задачі є одним із ключових етапів у процесі розробки інформаційної системи, оскільки саме на цьому етапі визначається мета створення ПЗ, описуються вимоги до його функціональності, встановлюються критерії успішності та окреслюється загальна структура майбутньої системи [19, с.31]. Від точності та повноти постановки задачі залежить успішність реалізації проєкту, його відповідність очікуванням кінцевих користувачів і замовника, а також можливість подальшого розширення функціоналу.

Розроблювана система призначена для бронювання та продажу авіаквитків через інтуїтивно зрозумілий інтерфейс. Вона має забезпечити повний цикл взаємодії користувача з сервісом — від моменту реєстрації до остаточного оформлення замовлення, включаючи функції пошуку, перегляду, вибору та бронювання квитків. Окрім того, окрема частина функціоналу призначена для адміністраторів, які мають змогу управляти даними про рейси.

Основні вимоги до розроблюваної системи включають:

а) реєстрація та авторизація користувача:

Система повинна забезпечити створення облікових записів для нових користувачів, а також безпечний процес входу до системи для вже зареєстрованих. Передбачено перевірку введених даних на валідність, у тому числі формат електронної пошти, відповідність пароля, перевірку на унікальність користувача. Вхід до системи супроводжується перевіркою облікових даних у базі.

б) пошук рейсів:

Користувач повинен мати змогу швидко та зручно знайти потрібні рейси, задаючи критерії пошуку: місто відправлення, місто призначення, дата подорожі, клас обслуговування. Результати мають відображатися у вигляді карток з

ключовою інформацією: авіакомпанія, номер рейсу, час вильоту та прильоту, вартість квитка, тривалість перельоту.

в) бронювання квитків:

Після вибору рейсу користувач має можливість переглянути доступні місця на борту літака, вибрати потрібне і здійснити бронювання. Система повинна автоматично перевіряти доступність місця на момент вибору та оновлювати дані в режимі реального часу, запобігаючи подвійній броні.

г) адміністрування рейсів:

Для адміністраторів системи передбачено окремий інтерфейс, що дозволяє управляти рейсами. Основні можливості включають додавання нових рейсів із повною інформацією (маршрут, час, літак, кількість місць, ціни), редагування вже наявних рейсів, а також їх видалення у випадку скасування або закінчення дії.

У процесі реалізації було поставлено завдання створити повноцінну інформаційну систему, яка включає зручний графічний інтерфейс, функціонал для бронювання та продажу авіаквитків, взаємодію з базою даних, а також підтримку адміністративного управління рейсами.

1.2. Вхідна інформація

Вхідна інформація є відправною точкою для проєктування та реалізації інформаційної системи. Вона охоплює всі необхідні дані, що надходять на початковому етапі створення програмного забезпечення та служать базою для побудови логічної, інформаційної та функціональної структур системи. Саме ці дані визначають вимоги до структури бази даних, специфіку інтерфейсу користувача, бізнес-логіку обробки запитів, а також обсяг і зміст операцій, які повинна підтримувати система.

У контексті розробки системи бронювання та продажу авіаквитків вхідна інформація охоплює кілька категорій даних, без яких повноцінне функціонування програми є неможливим. Зокрема:

а) дані про рейси;

б) інформація про літаки;

в) дані про пасажирів;

г) дані про бронювання;

д) ціни на квитки;

е) інформація про наявність вільних місць.

На основі зазначених даних формуються ключові функціональні вимоги до системи, що відображаються у вигляді задач, які повинно виконувати програмне забезпечення. У таблиці 1.1 наведено основні задачі системи та їх призначення.

Таблиця 1.1

Задачі програмного забезпечення

№	Найменування задачі	Призначення задачі	Періодичність виконання
1	Пошук рейсів	Пошук відповідних рейсів за заданими параметрами	Постійно
2	Бронювання квитка	Створення нового бронювання на обраний рейс	Постійно
3	Перегляд інформації про бронювання	Надання користувачу деталей його бронювання	Постійно
4	Адміністрування бази даних	Додавання/редагування рейсів, літаків, тарифів	За потреби

Джерела вхідної інформації – це передусім база даних, яка формується у процесі початкового наповнення системи адміністраторами. Цей етап охоплює імпорт розкладу авіакомпаній, конфігурацій літаків, тарифів та інших параметрів. У подальшому, база даних оновлюється автоматично або вручну, залежно від змін у розкладі, цінах або наявності літаків. Також можливе підключення зовнішніх джерел — API авіаперевізників, що дозволяє автоматично синхронізувати дані.

Збереження вхідної інформації здійснюється у структурованому вигляді за допомогою СУБД Microsoft SQL Server, що забезпечує реляційну модель організації даних. Основні таблиці системи включають:

а) Flights — інформація про всі рейси;

б) Seats — інформація про місця;

в) Bookings — інформація про кожне бронювання;

г) Users — облікові записи клієнтів і адміністраторів.

Кожна з таблиць містить відповідні первинні ключі, зовнішні ключі та обмеження цілісності, що дозволяє гарантувати узгодженість та достовірність даних.

Вхідна інформація є основою для наступного етапу — розробки логіки взаємодії користувача з системою, побудови форм введення/виведення та реалізації бізнес-функцій. Саме на підставі аналізу вхідних даних визначаються типові сценарії користувача, оптимальні варіанти побудови інтерфейсу та архітектурні рішення всієї програми [18, с.74].

1.3. Вихідна інформація

Вихідна інформація — це результати обробки запитів користувачів та адміністраторів системи, які формуються після взаємодії з базою даних та внутрішніми модулями програмного забезпечення [24, с.155]. Вона слугує основним засобом зворотного зв’язку, що надає користувачеві змогу приймати рішення, виконувати цільові дії та контролювати стан власних операцій, а адміністратору — отримувати статистичну та поточну інформацію для моніторингу й управління.

У системі бронювання та продажу авіаквитків вихідна інформація представлена у різних формах: табличних звітах, текстових повідомленнях, інформаційних діалогах або оновленнях інтерфейсу в реальному часі. Вона генерується за запитами користувачів, таких як пошук рейсів, перегляд деталей бронювання, редагування особистих даних, підтвердження замовлень тощо.

Для забезпечення ефективної взаємодії з користувачем вихідні дані в системі подаються у зручному та наочному форматі. Інформація відображається у вигляді структурованих таблиць, які містять зрозумілі заголовки, підтримують функції сортування, фільтрації та пошуку. Такий підхід дозволяє користувачам швидко орієнтуватися в даних, оперативно знаходити необхідну інформацію, а також приймати обґрунтовані рішення на основі актуальних відомостей.

Усі повідомлення, сповіщення та дані, що відображаються в інтерфейсі користувача, формуються автоматично — на основі актуального вмісту бази даних. Це гарантує їхню достовірність, точність і своєчасність. Автоматизація процесу оновлення інформації мінімізує ризик помилок, викликаних людським фактором, та забезпечує високий рівень зручності й надійності у повсякденній роботі з системою.

Основні типи вихідної інформації охоплюють такі категорії:

а) Список доступних рейсів який формується внаслідок запиту користувача щодо рейсів між двома містами на конкретну дату. Містить поля: номер рейсу, пункт відправлення та призначення, час вильоту і прильоту, ціна, доступність місць. Ця інформація є критичною на етапі вибору квитка та впливає на рішення користувача.

б) Деталі бронювання які надаються після вибору конкретного рейсу і містять повну інформацію про бронювання: маршрут, дані пасажира, обране місце, вартість. Таке повідомлення відображається у вигляді форми підтвердження.

в) Інформація про пасажира яка формується після автентифікації або в процесі редагування профілю. Містить персональні дані, історію бронювань, налаштування. Це дозволяє здійснювати персоналізацію послуг.

г) Адміністративний звіт: доступний тільки адміністраторам. Містить агреговану інформацію про кількість активних бронювань, завантаженість конкретних рейсів, суму отриманих платежів тощо.

Характеристика вихідних повідомлень наведена у таблиці 1.2.

Таблиця 1.2

Перелік і опис вихідних повідомлень

№ п/п	Назва вихідного повідомлення	Призначення задачі	Форма представлення	Термін і частота надходження
1	Список доступних рейсів	Інформування користувача про рейси, що відповідають заданим критеріям пошуку	Таблична	За запитом користувача
2	Деталі бронювання	Виведення даних про обране бронювання (рейс, місце, ціна)	Таблична / текстова	Після вибору рейсу
3	Інформація про пасажира	Виведення персональних даних клієнта для перевірки чи редагування	Таблична	3
4	Адміністративний звіт	Дані про кількість бронювань, завантаженість рейсів	Таблична	4

Важливим аспектом формування вихідної інформації є її актуальність, достовірність і оперативність. Дані, що відображаються, повинні відповідати реальному стану системи на момент запиту. Для цього забезпечується безперервна синхронізація з базою даних, контроль транзакцій та обмеження доступу до певних повідомлень залежно від ролі користувача.

Таким чином, вихідна інформація є ключовим елементом комунікації між системою та її користувачем. Вона забезпечує не лише інформаційний зворотній зв’язок, а й виконує роль інструменту контролю, аналітики та прийняття рішень. Її правильне структурування та подання безпосередньо впливають на зручність роботи з системою, швидкість виконання дій і загальний рівень задоволеності користувача.

Висновки до розділу

У даному розділі було обґрунтовано необхідність створення спеціалізованої інформаційної системи для автоматизації процесів бронювання та продажу авіаквитків. Актуальність дослідження зумовлена сучасними вимогами до швидкості, зручності та безпеки обслуговування клієнтів у сфері авіаперевезень. Традиційні підходи, що передбачають ручну обробку даних або застарілі інтерфейси, не відповідають очікуванням користувачів та призводять до помилок, затримок і незадоволеності клієнтів. Саме тому виникає потреба у впровадженні інтерактивного програмного рішення, здатного оптимізувати процеси пошуку, бронювання та адміністрування авіарейсів.

У розділі визначено основну мету дослідження — розробити ефективне, надійне та масштабоване програмне забезпечення, що дозволяє здійснювати всі необхідні операції з бронювання авіаквитків у зручному графічному середовищі. Об'єктом дослідження стала система управління бронюванням, а предметом — програмні та інформаційні засоби реалізації цієї системи.

Проведено загальну характеристику задачі та інформаційного забезпечення, що передбачає обробку значного обсягу даних, їх збереження у структурованій формі, забезпечення доступу з боку користувачів і адміністраторів, а також дотримання вимог щодо конфіденційності й цілісності інформації. Визначено основні види вхідної, проміжної та вихідної інформації, що циркулює в системі, охарактеризовано джерела даних та описано ключові задачі, які повинні реалізовуватись у програмному середовищі.

Таким чином, проведений аналіз дав змогу визначити вимоги до майбутнього програмного забезпечення, сформулювати технічне завдання та розпочати процес логічного моделювання структури бази даних і розробки програмної логіки. Отримані результати стануть основою для наступних етапів: моделювання, проектування та реалізації системи, що забезпечить ефективну автоматизацію процесів бронювання авіаквитків та підвищення якості обслуговування користувачів.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ЗАДАЧІ

2.1. Розробка алгоритму розв’язання задачі на мові програмування C#

Для реалізації системи бронювання та продажу авіаквитків на мові програмування C# необхідно розробити чіткий алгоритм взаємодії користувача з програмним забезпеченням, що забезпечить коректну обробку запитів, взаємодію з базою даних та зручний інтерфейс [1, с.110], [25, с.422]. Алгоритм має охоплювати як користувацькі, так і адміністративні сценарії використання системи, забезпечуючи відповідну реакцію інтерфейсу та надійний контроль даних на всіх етапах [7, с.178].

Розв’язання поставленої задачі передбачає виконання низки послідовних дій, які охоплюють як початкові етапи запуску програмного забезпечення, так і подальшу взаємодію користувача з системою в межах реалізованих функціональних сценаріїв. На першому етапі здійснюється запуск програми та ініціалізація головного вікна, що включає завантаження необхідних ресурсів, перевірку конфігурацій і встановлення з’єднання з реляційною базою даних Microsoft SQL Server. Наступним кроком є обробка користувацьких дій: вибір напрямку або рейсу, перегляд детальної інформації про маршрут, умови польоту, доступні місця, тарифи тощо. Після цього реалізується процес бронювання квитка, який включає верифікацію введених даних, перевірку наявності місць, формування запису у таблиці бронювань і видачу підтвердження користувачу.

Окремо варто відзначити підтримку адміністративного функціоналу, який включає можливість додавання, редагування та видалення записів про рейси, літаки, тарифи, а також перегляд звітної інформації щодо активності системи, завантаженості рейсів і кількості бронювань. Усі операції відбуваються через головне вікно програми та систему діалогових форм, які дозволяють структурувати роботу користувача та уникати помилкових дій завдяки зрозумілому інтерфейсу та перевірці введених даних [17, с.24].

Програмне забезпечення побудоване з використанням об’єктно-орієнтованого підходу, що забезпечує високу ступінь модульності, повторне використання коду та зручність підтримки й розширення функціональності. Основні класи системи відповідають за роботу з базою даних (виконання SQL-запитів, отримання результатів, оновлення даних), відображення графічного інтерфейсу користувача (форми, кнопки, повідомлення) та реалізацію логіки бізнес-процесів (обробка бронювання, перевірка доступності місць, розрахунок вартості квитків, тощо). Такий підхід дозволяє легко масштабувати систему в майбутньому, інтегруючи нові модулі — наприклад, обробку онлайн-платежів, генерацію електронних квитків або розширення звітності. Важливим етапом є обробка виключень, яка реалізована через конструкції try-catch і дозволяє уникнути аварійного завершення програми при помилках підключення до БД, некоректному введенні даних або логічних помилках користувача. Повідомлення про помилки реалізовані у вигляді інформативних діалогових вікон, що підвищує зручність взаємодії.

Система також підтримує розмежування прав доступу. Залежно від ролі користувача після авторизації відкривається відповідний функціонал:

а) для звичайного користувача — перегляд рейсів, оформлення бронювання, перевірка квитків;

б) для адміністратора — редагування розкладу, управління літаком, перегляд статистики.

Алгоритм роботи програми побудований у вигляді подієво-орієнтованої моделі, де всі дії ініціюються подіями інтерфейсу (натискання кнопки, вибір зі списку, підтвердження форми тощо). Обробники подій взаємодіють із відповідними класами для реалізації логіки програми, що дозволяє досягти високої узгодженості між візуальним інтерфейсом та функціональною частиною програми.

Загальну схему алгоритму роботи програми на мові C# показано на рисунку 2.1. Вона ілюструє основні етапи взаємодії з системою — від входу до

отримання квитка — та взаємозв’язки між модулями інтерфейсу, логіки бізнес процесів та базою даних.

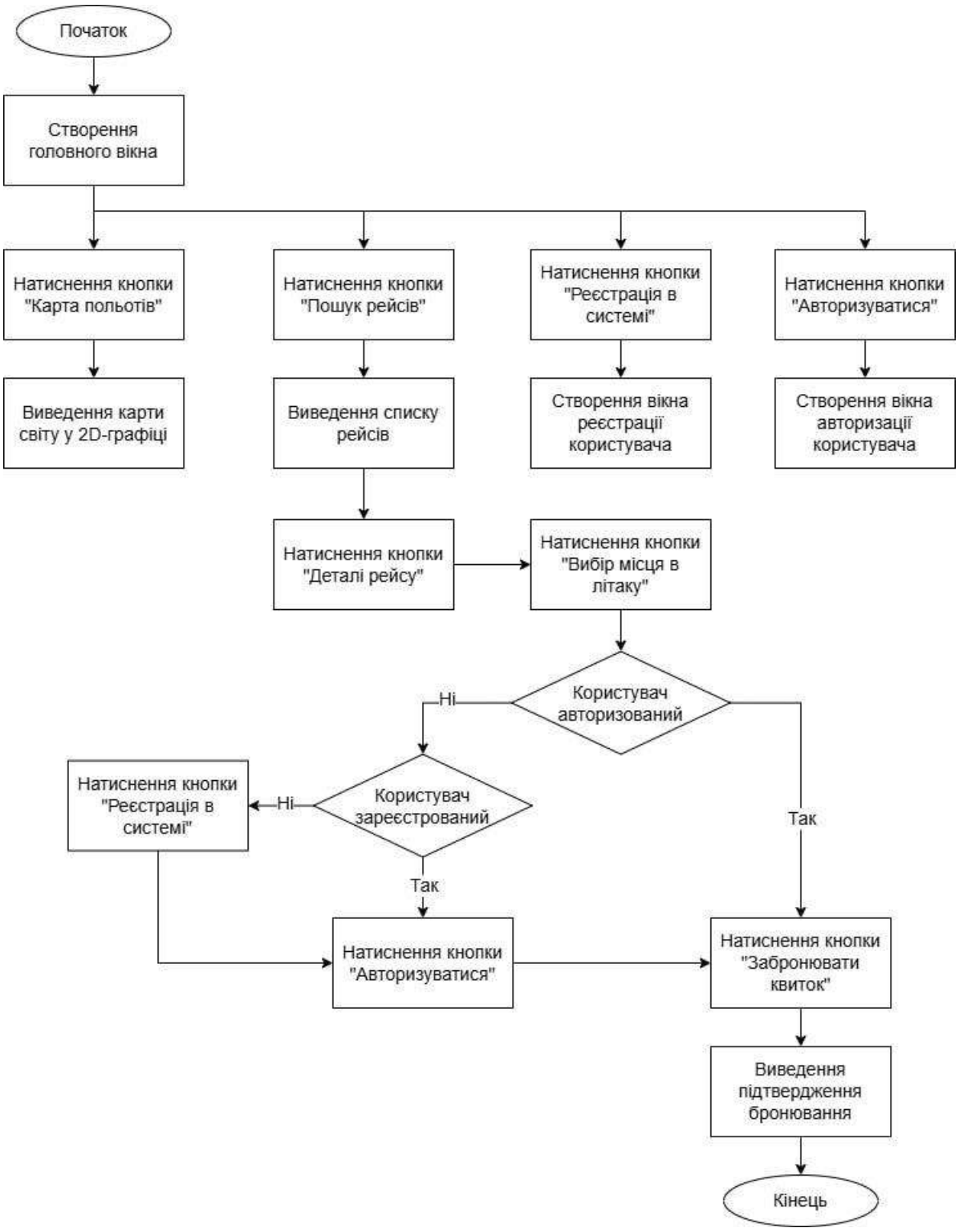

Рис. 2.1. Алгоритм роботи програми

Уся взаємодія користувача з системою відбувається через головне вікно програми та діалогові форми, що відповідають окремим функціональним

модулям: реєстрація та авторизація, перегляд рейсів, оформлення квитків, адміністрування. Кожен модуль має чітку логіку роботи, яка забезпечується внутрішніми методами, подіями та запитами до бази даних.

Інтерфейс побудований із урахуванням принципів юзабіліті та наочності. Кожна дія користувача супроводжується логічно структурованою реакцією системи — завдяки чому забезпечується інтуїтивне керування та мінімальна кількість кроків для досягнення мети.

Таким чином, розроблений алгоритм забезпечує злагоджену роботу всіх компонентів програми, відповідає функціональним вимогам і дозволяє реалізувати повний цикл обслуговування користувача — від пошуку авіаквитків до остаточного оформлення замовлення.

Алгоритм реалізується за допомогою Windows Forms або WPF на мові C# з використанням об’єктно-орієнтованого підходу, патернів MVVM. Зв’язок із базою даних реалізовано за допомогою ORM-фреймворку Entity Framework, що дозволяє працювати з таблицями бази як з об’єктами C# без необхідності прямого написання SQL-запитів [6, с.270], [21, с.550]. Усі запити — на вибірку, вставку, оновлення чи видалення даних — реалізовані через контекст бази та відповідні методи, що забезпечує високу швидкість розробки, меншу кількість помилок і зручне масштабування структури бази.

Таким чином, побудована логіка дозволяє реалізувати повний життєвий цикл взаємодії з користувачем — від первинного входу в систему, через перегляд можливих рейсів, до остаточного бронювання та підтвердження замовлення. Всі елементи системи працюють узгоджено завдяки правильно спроектованим внутрішнім зв’язкам, викликам методів, реакціям на події та запитам до бази даних.

Завдяки цьому алгоритм забезпечує не лише злагоджену роботу всіх компонентів системи, але й високу якість користувацького досвіду, відповідає функціональним і нефункціональним вимогам та демонструє приклад сучасного підходу до побудови інформаційних систем із використанням новітніх технологій Microsoft.

Висновки до розділу

У розділі було проведено проектування задачі з урахуванням вимог до функціональності та зручності використання системи бронювання авіаквитків. Сформовано загальний алгоритм роботи програми, що включає етапи ініціалізації інтерфейсу, підключення до бази даних, обробки дій користувача, формування запитів і виводу інформації.

Також було розглянуто вхідну та вихідну інформацію, необхідну для ефективного функціонування системи. Вхідні дані включають інформацію про рейси, користувачів, замовлення та правила обробки даних, тоді як вихідні – це результати запитів, підтвердження бронювань, звіти та повідомлення для користувачів.

Таким чином, проектування задачі дозволило створити чітку основу для подальшої реалізації програмного забезпечення, забезпечуючи правильну логіку взаємодії між усіма складовими системи та ефективне виконання функціональних задач.

РОЗДІЛ 3
ОРГАНІЗАЦІЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Загальна характеристика інформаційного забезпечення

Інформаційне забезпечення є ключовим компонентом будь-якої інформаційної системи і відіграє вирішальну роль у її функціонуванні. Воно включає сукупність вхідних, проміжних і вихідних даних, необхідних для реалізації основних функцій системи, а також засоби їх збереження, обробки та передачі.

У контексті системи бронювання та продажу авіаквитків інформаційне забезпечення охоплює:

а) вхідну інформацію, до якої належать запити користувачів щодо пошуку рейсів, реєстраційні дані пасажирів, а також адміністраторські дії щодо оновлення рейсів чи розкладу, аеропортів чи зміни цінової політики. До вхідної інформації також належать платіжні реквізити користувачів та ідентифікаційні дані при вході до системи;

б) проміжну інформацію, яка генерується під час взаємодії з базою даних — це результати запитів, перевірка наявності місць, обчислення вартості квитків із врахуванням податків, знижок і комісій, а також формування сесій користувачів, тимчасових даних про замовлення тощо;

в) вихідну інформацію, яка формується в результаті обробки вхідних запитів і включає перелік доступних рейсів за заданими параметрами, підтвердження успішного бронювання, електронні квитки у форматі PDF або QR-коди, фінансові звіти, чеки про оплату, повідомлення на електронну пошту, а також адміністративні звіти про обсяг продажів, завантаженість рейсів тощо.

Однією з ключових вимог до інформаційного забезпечення є актуальність і точність даних. Для цього передбачені механізми оновлення інформації про рейси та валідації введених користувачами даних. Крім того, з метою

забезпечення безперервного доступу до критичних даних, реалізовано надійні засоби збереження та резервного копіювання.

Для збереження і обробки інформації у системі використовується реляційна база даних, створена за допомогою Microsoft SQL Server. Основна структура БД охоплює такі таблиці: Users, Flights, Seats, Bookings. Важливо, щоб база даних відповідала таким вимогам:

а) структурованість;

б) наявність ключових полів;

в) застосування обмежень цілісності;

г) захист даних.

Інформаційне забезпечення системи має бути надійним, адаптивним до змін вимог, забезпечувати швидкий доступ до потрібних даних і підтримувати як користувацький, так і адміністративний функціонал. Його якість безпосередньо впливає на зручність використання програми, швидкість обробки операцій, можливість масштабування системи? загальний рівень задоволеності клієнтів та ефективність обслуговування клієнтів. Особливо важливо, щоб інформація була актуальною, узгодженою і доступною у реальному часі для користувачів, незалежно від обраного пристрою чи локації.

3.2. Моделювання структури бази даних

У цьому розділі розглядається процес побудови логічної структури бази даних, яка лежить в основі функціонування програмного забезпечення для бронювання та продажу авіаквитків. Основною метою даного етапу є створення цілісної, оптимізованої та взаємозв’язаної системи зберігання даних, що дозволяє ефективно обробляти інформацію про рейси, пасажирів, квитки, бронювання, платежі та інші сутності, необхідні для повноцінної роботи системи.

Проєктування структури бази даних є критично важливим етапом у процесі створення інформаційної системи, оскільки саме на цьому етапі

формується логічна організація всієї інформації, яка обробляється в межах програмного забезпечення [14, с.156]. Правильно спроєктована база даних слугує надійним фундаментом для реалізації бізнес-логіки, забезпечує узгодженість і цілісність даних під час виконання множинних транзакцій, дозволяє уникнути дублювання записів, зменшує надлишковість інформації та сприяє економії пам’яті [4, с.115], [12, с.100].

Уся потрібна інформація в базі даних організована у вигляді таблиць, характеристика яких подана в таблиці 3.1. Сама база даних реалізована у форматі Microsoft SQL Server.

Таблиця 3.1

Характеристика таблиць бази даних

№ п/п	Назва таблиці	Опис
1	Flights	Інформація про рейси: • Номер рейсу • Компанія • Місто відправлення • Місто прибуття • Час відправлення • Час прибуття • Ціна • Клас
2	Bookings	Інформація про бронювання: • ПІБ пасажира • Номер паспорту • Пошта • Телефон • Номер сидіння • Дата бронювання
3	Seats	Інформація про сидіння: • Номер сидіння • Доступність
4	Users	Інформація про користувача: • Ім’я користувача • Пароль • Роль користувача

3.3. Моделювання структури таблиць бази даних

Проєктування структури бази даних є одним із ключових та відповідальних етапів створення інформаційної системи, оскільки від правильності її побудови залежить ефективність, стабільність та масштабованість усього програмного продукту. У контексті розробки системи бронювання та продажу авіаквитків, база даних виконує функцію центрального сховища інформації, де зберігаються всі необхідні відомості для забезпечення роботи основних бізнес-процесів.

База даних забезпечує довготривале збереження, структуровану організацію, швидкий доступ та обробку інформації, пов’язаної з користувачами, авіарейсами, квитками, бронюваннями, а також адміністративними діями. При її розробці враховуються вимоги до цілісності даних, тобто необхідність виключення дублювання, логічних помилок, забезпечення взаємозв’язків між таблицями через зовнішні ключі, а також відповідність нормалізаційним формам, що сприяє ефективній організації даних.

Для побудови бази даних була обрана система керування базами даних Microsoft SQL Server, яка забезпечує високу продуктивність, масштабованість та надійність [26, с.62]. Уся інформація представлена у вигляді взаємопов’язаних таблиць, кожна з яких зберігає певний тип даних, пов'язаний з функціональністю системи.

Структура таблиць спроєктована таким чином, щоб мінімізувати дублювання даних, забезпечити логічну організацію та спростити обробку запитів для користувачів [20, с.430]. Це дозволяє підвищити ефективність зберігання інформації та полегшує масштабування бази даних у разі розширення системи.

У таблиці «Flights» (табл. 3.2) зберігаються основні відомості про рейси, включаючи номери рейсів, часи вильоту та прильоту, пункти призначення і вартість. Таблиця «Bookings» (табл. 3.3) містить інформацію про бронювання, включаючи особисті дані пасажира та пов’язані ідентифікатори рейсу й місця.

Таблиця «Seats» (табл. 3.4) дозволяє вести облік доступності конкретних місць у кожному рейсі, що важливо для точної обробки замовлень у режимі реального часу. Таблиця «Users» (табл. 3.5) забезпечує зберігання облікових записів користувачів з розмежуванням ролей.

У таблицях чітко визначені первинні та зовнішні ключі, що забезпечує підтримку зв’язків між об'єктами бази даних.

Таблиця 3.2

Структура таблиці Flights

№	Назва поля	Тип даних	Призначення
1	Id	int (PK)	Унікальний ідентифікатор рейсу (первинний ключ)
2	FlightNumber	nvarchar(max)	Номер рейсу
3	Airline	nvarchar(max)	Назва авіакомпанії
4	DepartureCity	nvarchar(max)	Місто відправлення
5	ArrivalCity	nvarchar(max)	Місто прибуття
6	DepartureTime	datetime2(7)	Час відправлення
7	ArrivalTime	datetime2(7)	Час прибуття
8	Price	decimal(18,2)	Вартість квитка
9	Class	nvarchar(max)	Клас обслуговування (економ, бізнес тощо)

Таблиця 3.3

Структура таблиці Bookings

№	Назва поля	Тип даних	Призначення
1	Id	int (PK)	Унікальний ідентифікатор бронювання (первинний ключ)
2	FullName	nvarchar(max)	ПІБ пасажира
3	PassportNumber	nvarchar(max)	Номер паспорта пасажира
4	Email	nvarchar(max)	Електронна пошта пасажира
5	Phone	nvarchar(max)	Номер телефону пасажира
6	FlightId	int (FK)	Посилання на рейс, який бронюється (зовнішній ключ до таблиці Flights)
7	SeatId	int (FK)	Посилання на вибране сидіння (зовнішній ключ до таблиці Seats)
8	BookingDate	datetime2(7)	Дата бронювання

Таблиця 3.4

Структура таблиці Seats

№	Назва поля	Тип даних	Призначення
1	Id	int (PK)	Унікальний ідентифікатор сидіння (первинний ключ)
2	SeatNumber	nvarchar(max)	Номер сидіння
3	IsAvailable	bit	Доступність сидіння (1 – доступне, 0 – зайняте)
4	FlightId	int (FK)	Посилання на рейс, до якого належить сидіння (зовнішній ключ до Flights)

Таблиця 3.5

Структура таблиці Users

№	Назва поля	Тип даних	Призначення
1	Id	int (PK)	Унікальний ідентифікатор користувача (первинний ключ)
2	Username	nvarchar(100)	Ім’я користувача (логін)
3	PasswordHash	nvarchar(max)	Хеш пароля користувача для зберігання у безпечному вигляді
4	Role	nvarchar(50)	Роль користувача (наприклад, адміністратор, клієнт тощо)

Ці таблиці формують структуровану та логічно узгоджену модель бази даних, яка дозволяє системі бронювання та продажу авіаквитків ефективно функціонувати на всіх рівнях — від обробки запитів користувача до забезпечення достовірності та збереження інформації. Кожна таблиця виконує свою конкретну роль.

Використання зовнішніх ключів є критично важливим для побудови зв’язків між таблицями. Наприклад, кожне бронювання «Booking» пов’язане з конкретним рейсом «Flight» і користувачем «User», що здійснив бронювання. Завдяки цьому всі дані в системі зберігаються не ізольовано, а як частина єдиної цілісної схеми, що забезпечує узгодженість інформації між модулями.

Таким чином, логічна структура бази даних на основі взаємозв’язаних таблиць забезпечує не лише зручність обробки та зберігання даних, а й гарантує стабільність, безпечність та ефективність роботи всієї інформаційної системи.

Висновки до розділу

У третьому розділі було розглянуто ключові аспекти інформаційного забезпечення системи бронювання та продажу авіаквитків. Надано загальну характеристику інформаційних потоків, зокрема вхідної та вихідної інформації, яка обробляється в процесі функціонування програмного забезпечення.

Особливу увагу приділено побудові та моделюванню структури бази даних. Було детально описано основні таблиці, їхні поля, типи даних і логіку взаємозв’язків між ними. Реалізація бази даних у середовищі Microsoft SQL Server забезпечує необхідну гнучкість, масштабованість і надійність зберігання даних.

Таким чином, інформаційне забезпечення системи є цілісним, структурованим та логічно організованим. Воно повністю відповідає вимогам до розробки сучасних інформаційних систем і створює надійний фундамент для ефективної роботи всіх функціональних модулів програмного забезпечення.

РОЗДІЛ 4
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Інтерфейс користувача системи

На першому етапі розробки особливу увагу було приділено проєктуванню інтерфейсу користувача, адже саме з ним взаємодіє кінцевий користувач. Інтерфейс має бути не лише естетично привабливим, але й інтуїтивно зрозумілим, логічно структурованим та зручним у користуванні. Для реалізації інтерфейсу було обрано технологію WPF, яка надає широкі можливості для створення сучасних настільних додатків із графічно насиченим інтерфейсом [2, с.150], [5, с.92].

Існує кілька основних типів інтерфейсів, які можуть використовуватись у програмному забезпеченні:

а) текстовий інтерфейс (CLI — Command Line Interface) – це взаємодія користувача з додатком відбувається шляхом введення текстових команд у консоль [28, с.155]. Переваги — швидкість, гнучкість, низьке споживання ресурсів. Недоліки — потреба у знаннях, складність для пересічного користувача;

б) графічний інтерфейс користувача (GUI — Graphical User Interface), цей інтерфейс передбачає взаємодію за допомогою вікон, кнопок, меню, форм. Це найпоширеніший тип інтерфейсу для користувацьких додатків, оскільки він інтуїтивно зрозумілий і візуально привабливий;

в) веб-інтерфейс – це додаток який працює через браузер. Зручний для доступу з будь-якого пристрою, але вимагає підключення до мережі та серверної інфраструктури;

г) мобільний інтерфейс. Призначений для пристроїв з сенсорним екраном. Вимагає окремої розробки для Android або iOS;

д) гібридні інтерфейси - поєднують у собі можливості веба та десктопу, але можуть мати обмеження у продуктивності чи складності підтримки.

WPF — це сучасна технологія створення настільних GUI-додатків під Windows. Основними перевагами, які зумовили вибір саме цієї платформи, стали:

а) гнучкий дизайн. Завдяки XAML (розмітка інтерфейсу) легко створювати адаптивні, багатошарові інтерфейси з підтримкою стилів, шаблонів та анімацій;

б) підтримка шаблонів даних і прив’язки (Data Binding) які спрощують взаємодію між інтерфейсом та логікою програми, що ідеально підходить для патерну MVVM;

в) можливість побудови складних візуальних елементів. Наприклад, відображення діаграм, інтерактивних таблиць, календарів, графіків тощо;

г) локальне виконання. Додаток не потребує постійного підключення до Інтернету, що важливо для офлайн-бізнесів;

д) простота інтеграції з базою даних. WPF легко поєднується з ADO.NET або Entity Framework для роботи з SQL Server;

е) розмежування інтерфейсу та логіки. Архітектурний патерн MVVM (Model-View-ViewModel), який підтримується WPF "з коробки", дозволяє чітко розділити код інтерфейсу та бізнес-логіку, спрощуючи тестування й супровід [9].

Розроблене програмне забезпечення є десктопним додатком, реалізованим з використанням мови програмування C#. В якості бази даних використовується Microsoft SQL Server. Архітектура додатку реалізована за шаблоном MVVM [Додаток В], що дозволяє розділити логіку представлення, бізнес-логіку та модель даних [3, с.90].

Основні функціональні можливості системи:

а) реєстрація нового користувача;

б) авторизація користувачів;

в) перегляд доступних рейсів;

г) детальна інформація про обраний рейс;

д) вибір та бронювання місця;

е) введення персональних даних пасажира;

ж) доступ адміністратора до панелі керування рейсами;

з) додавання, редагування та видалення рейсів;

і) візуалізація поточних рейсів на карті.

На рисунку 4.1 представлено головне вікно застосунку «EasyTicket», яке реалізує інтерфейс взаємодії користувача з підсистемою пошуку та бронювання авіаквитків. Основний акцент у дизайні зроблено на простоту, візуальну привабливість та інтуїтивну навігацію, що є критично важливим при створенні програмного забезпечення для широкого кола користувачів [28, с.192]. Вікно складається з чітко структурованих компонентів, що забезпечують зручну роботу з функціональністю системи [13, с.77], [16, с.279].

У верхній частині вікна розміщено логотип EasyTicket, який водночас слугує навігаційною кнопкою — при натисканні користувач повертається до початкового стану вікна. Це дозволяє користувачу швидко повернутись до стартової сторінки незалежно від поточного розділу програми, що підвищує зручність навігації. Поруч з логотипом розташовані дві основні кнопки — «Search of tickets» та «Flight map», які дозволяють здійснити перехід до пошуку квитків або до інтерактивної карти польотів відповідно. У правому куті передбачено кнопки «Login» та «Register», що забезпечують авторизацію та реєстрацію користувачів.

Центральне місце займає блок із популярними напрямками — «Popular destinations», який реалізований у вигляді трьох збільшених карток з зображеннями визначних міст: Парижа, Токіо та Нью-Йорка. Кожна картка оформлена з використанням скруглених кутів, тіней та підписів, що надає інтерфейсу сучасного вигляду й акцентує увагу на візуальному контенті. Зображення підібрані яскраві та високої якості, що стимулює користувача до натискання й перегляду інформації про рейси в обране місто. Додатково реалізована рамка над зображенням, яка знаходиться на передньому плані, що візуально підкреслює межі кожної картки.

У нижній частині розміщено мотиваційне гасло «YOUR ADVENTURE BEGINS WITH EASYTICKET!», виконане великими літерами та з використанням привабливого синього кольору. Це елемент маркетингової

стратегії, що підсилює враження від взаємодії з додатком, формує емоційний зв’язок з брендом та заохочує користувача до здійснення покупки.

Загалом, дизайн вікна відповідає сучасним стандартам UI/UX і орієнтований на забезпечення позитивного користувацького досвіду. Поєднання інформативного змісту, зручного розміщення елементів та емоційно привабливих візуальних рішень сприяє ефективній реалізації цілей програмного забезпечення [29, с.177].

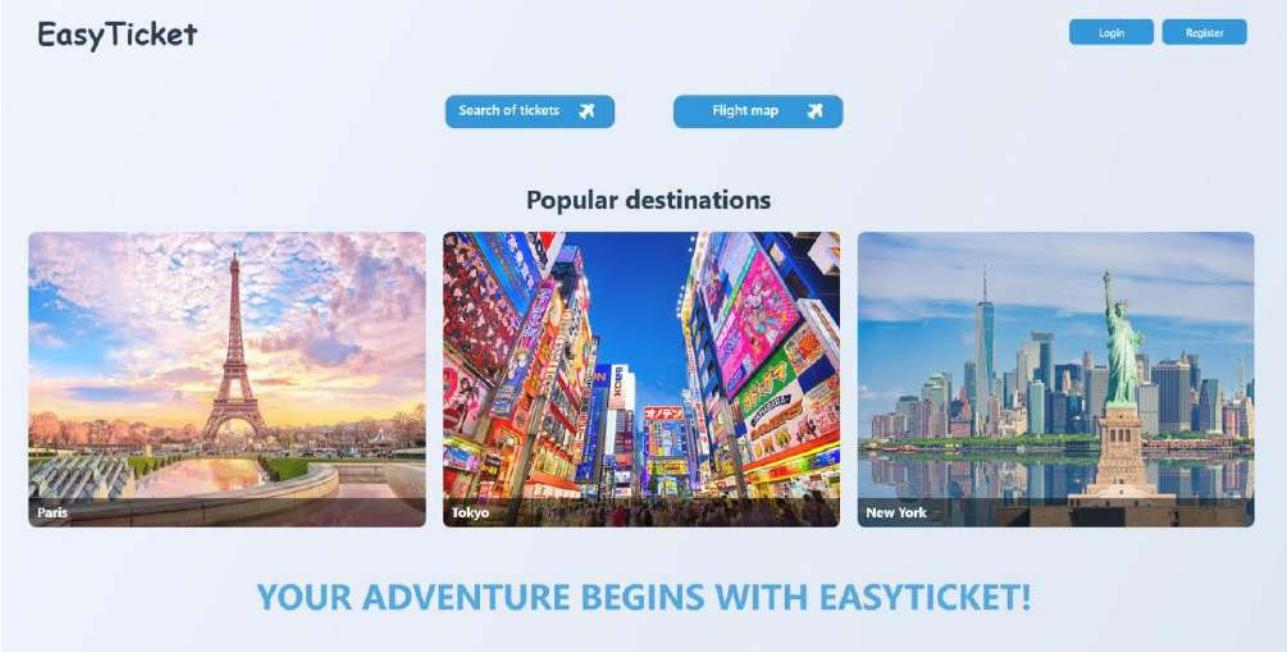

Рис. 4.1. Головне вікно програми

4.2. Модель даних та загальна структура програмного забезпечення

Усі моделі, що використовуються у додатку, знаходяться у папці «Models» [Додаток А]. Кожна модель відповідає сутності з бази даних. Зв’язок із базою даних забезпечується через «Entity Framework Core», а саме через клас «AppDbContext», що міститься у папці «Services» [8].

Розроблена система бронювання та продажу авіаквитків надає користувачам інтуїтивно зрозумілий інтерфейс для виконання ключових дій, пов’язаних із пошуком, переглядом, бронюванням та адмініструванням

авіарейсів. У цьому підрозділі розглянуто основні функціональні можливості системи з детальним описом кожного модуля.

а) реєстрація користувача:

На рисунку 4.2 представлено вікно реєстрації нового користувача, яке є ключовим елементом початкової взаємодії клієнта з системою бронювання та продажу авіаквитків. Реалізація вікна виконана у мінімалістичному стилі, орієнтованому на простоту, інтуїтивну зрозумілість і відповідність сучасним стандартам UX-дизайну. Основна мета цього інтерфейсу — забезпечити швидку, зручну та безпечну процедуру створення облікового запису, що дозволяє користувачу надалі авторизуватись, здійснювати бронювання, переглядати історію замовлень та отримувати персоналізований сервіс.

Центральним елементом вікна є заголовок «Register», оформлений жирним шрифтом, що чітко сигналізує про функціональне призначення інтерфейсу. Безпосередньо під ним розміщено три текстові поля: «Username» — для введення унікального імені користувача, «Password» — для створення пароля, та «Confirm Password» — для підтвердження правильності введеного пароля. Така структура є класичною і перевіреною з точки зору користувацького досвіду, оскільки дозволяє уникати помилок у введенні чутливої інформації, таких як пароль, що напряму впливає на безпеку доступу до облікового запису.

Поля мають однакові розміри, вирівняні по вертикалі та забезпечують достатній простір для введення даних. Такий підхід гарантує зручність при взаємодії з інтерфейсом на різних пристроях, включаючи мобільні телефони та планшети. Під формою розміщено синю кнопку «Register», яка візуально виділяється серед білого фону форми та служить основним елементом для підтвердження дій користувача. Кнопка оформлена у сучасному стилі із закругленими краями та тінню, що робить її більш помітною та інтуїтивно зрозумілою.

Нижче присутнє посилання «Back to Login», яке дає можливість швидко повернутися до вікна авторизації у разі, якщо користувач помилково відкрив форму реєстрації або вже має обліковий запис. Така логіка забезпечує зручну

навігацію між вікнами без необхідності повертатися на головну сторінку або повторно вводити дані.

Загалом інтерфейс реалізовано таким чином, щоб забезпечити мінімальний вхідний бар'єр для нових користувачів і сприяти ефективному залученню клієнтів до використання програмного забезпечення. Простий і зручний дизайн, відсутність відволікаючих елементів, інтуїтивне розміщення полів та кнопок сприяє швидкому заповненню форми навіть користувачами з мінімальним технічним досвідом. Крім того, така реалізація полегшує подальшу валідацію введених даних на боці клієнта, а також дозволяє безпечно обробляти інформацію на сервері, що є критично важливим при роботі з персональними даними в межах авіаційного бізнесу.

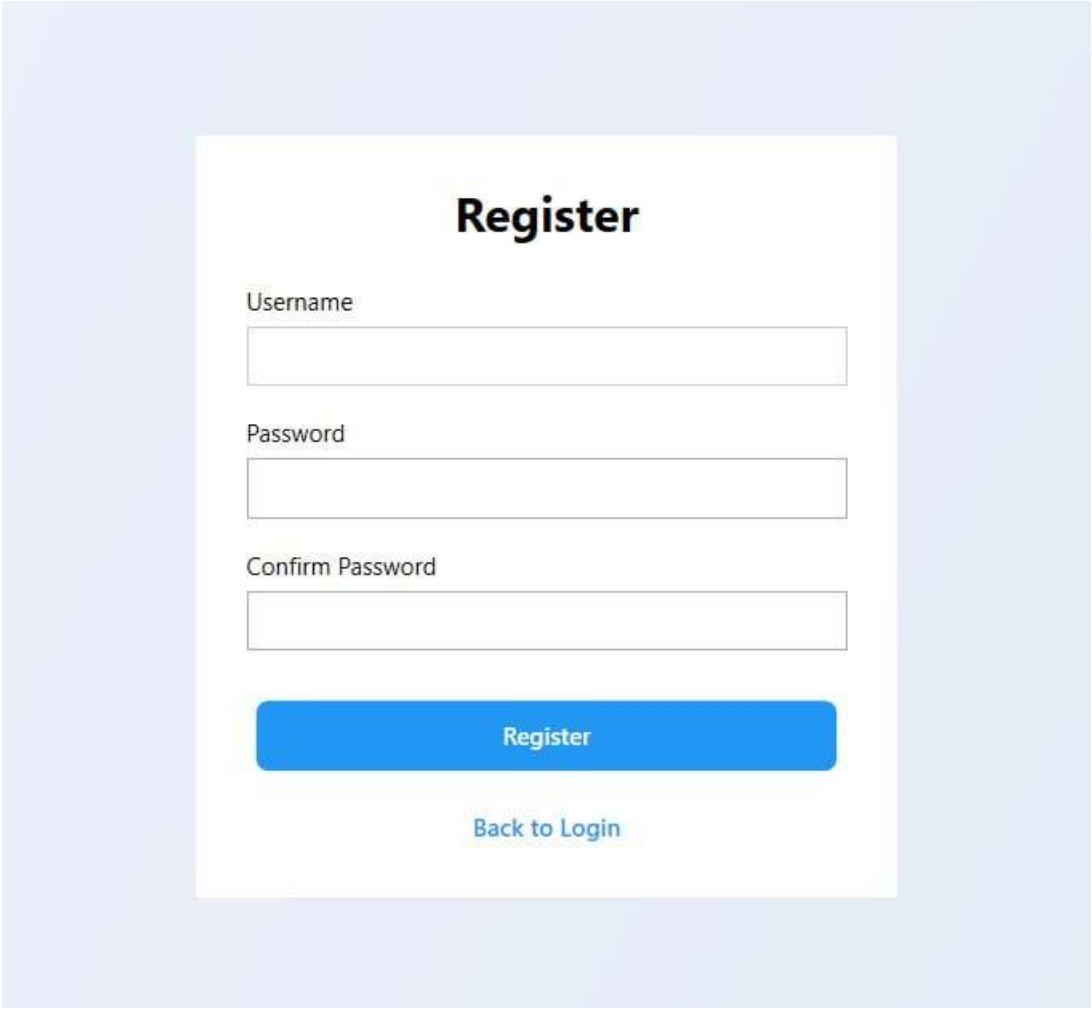

Рис. 4.2. Форма реєстрації

б) авторизація користувача:

На рисунку 4.3 представлена форма авторизації користувача, що є невід'ємним елементом будь-якої інформаційної системи з підтримкою персоніфікованого доступу. Дане вікно реалізоване в рамках програмного забезпечення для бронювання та продажу авіаквитків, і виконує функцію первинної перевірки особи користувача перед наданням доступу до основних функціональних можливостей системи [Додаток Б]. Інтерфейс форми побудовано з урахуванням принципів зручності, безпеки та лаконічності, що дозволяє знизити когнітивне навантаження на користувача та мінімізувати ймовірність помилок при введенні даних.

У центрі форми розміщено заголовок «Login», який виділено жирним шрифтом для кращої візуалізації призначення вікна. Нижче розташовано два текстові поля: «Username» для введення імені користувача та «Password» — для введення пароля. Обидва поля мають чіткі прямокутні межі з достатнім внутрішнім відступом, що забезпечує зручність для користувача на пристроях з різною роздільною здатністю екрана. Поле введення пароля за замовчуванням приховує символи — це стандартний механізм безпеки, що унеможливлює перегляд конфіденційних даних сторонніми особами.

Кнопка «Login» представлена у вигляді яскраво-синього прямокутника з білим написом, що інтуїтивно вказує на основну дію, яку має виконати користувач. Її розміщено безпосередньо під полями введення, що відповідає типовому шаблону поведінки користувача у веб-інтерфейсах [23, с.120]. Важливо також відзначити наявність посилання «Register» нижче кнопки авторизації, яке дає змогу новим користувачам швидко перейти до процедури створення облікового запису. Цей елемент оформлено менш виразно, щоб він не відволікав увагу від основної дії — входу до системи.

Такий дизайн обрано не випадково — він забезпечує максимальну зручність і швидкість взаємодії. Простота й лаконічність форм, відсутність зайвих графічних елементів, наявність лише необхідних функціональних

компонентів — усе це відповідає сучасним вимогам до UX/UI-дизайну для комерційних застосунків, зокрема в авіаційній сфері.

Таким чином, вікно авторизації реалізує один із базових функціональних компонентів системи продажу та бронювання авіаквитків, забезпечуючи безпечний і зручний вхід для зареєстрованих користувачів та закладаючи основу для реалізації ролей, доступів і персоналізованих функцій системи.

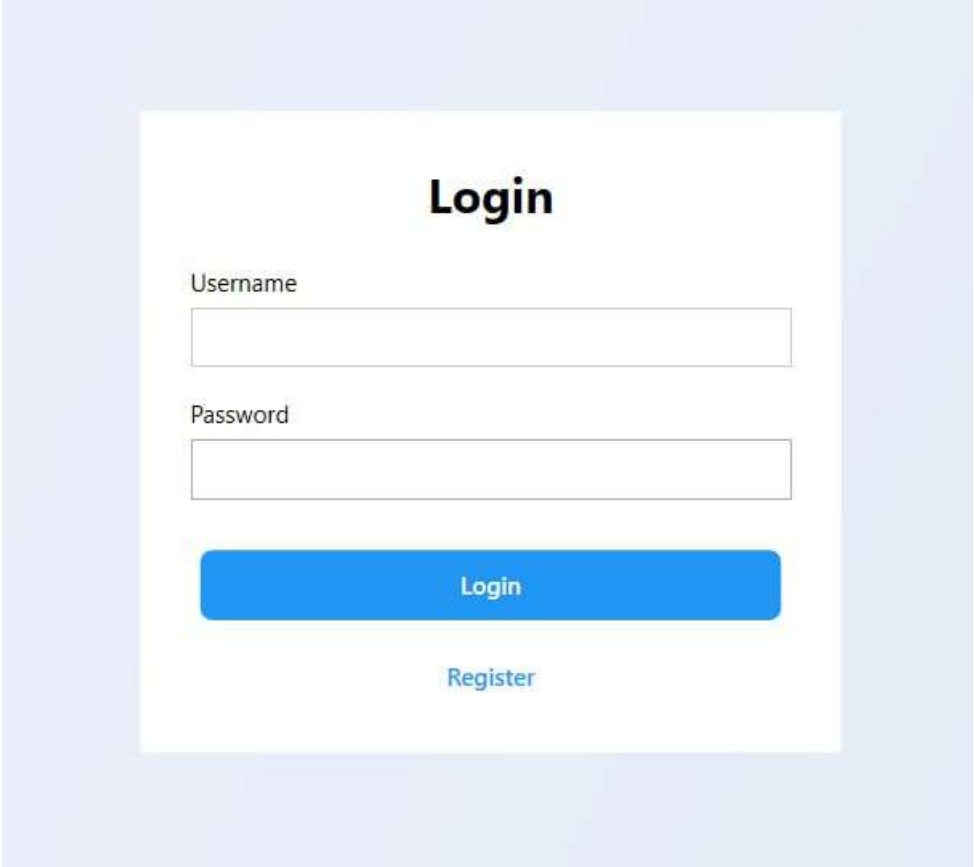

Рис. 4.3. Форма авторизації

в) пошук і перегляд рейсів:

На рисунку 4.4 зображено інтерфейс вікна для зміни або вибору авіарейсу в системі бронювання авіаквитків. Даний елемент користувацького інтерфейсу є частиною функціонального модуля, що відповідає за пошук рейсів за заданими параметрами, такими як пункт відправлення, пункт прибуття та дата подорожі. Вікно реалізовано у сучасному мінімалістичному стилі, що відповідає загальним тенденціям у дизайні веб-застосунків для авіаційної сфери, зокрема через

використання великого простору, простих шрифтів та інтуїтивно зрозумілих кнопок.

Інтерфейс поділено на кілька логічних блоків. У верхній частині розташовано дві основні вкладки: «Search of tickets» та «Flight map», що візуально виокремлені синіми кнопками із піктограмами літака, що підсилюють асоціативне сприйняття користувачем. Активна вкладка дозволяє здійснити пошук квитків за заданими критеріями. Такий підхід спрощує навігацію між різними підмодулями програми та дає змогу швидко перемикатися між пошуком квитків і візуалізацією маршрутів на карті.

Нижче розташовано основну форму пошуку рейсів. Поля «From» та «To» дозволяють користувачу ввести місто відправлення та прибуття відповідно. Кожне поле має прямокутну форму з чіткими межами, що забезпечує добру видимість на світлому фоні інтерфейсу. Праворуч від цих полів розміщено календар для вибору дати рейсу — інструмент, який мінімізує можливість помилки користувача при введенні дати вручну. Кнопка «Search» виконана у вигляді яскраво-синього прямокутника, що контрастує з фоном і привертає увагу, спрямовуючи користувача до логічного завершення операції пошуку.

Обраний дизайн відповідає вимогам до інтерфейсів критично важливих інформаційних систем, оскільки він є інтуїтивним, чистим і не перевантаженим зайвими деталями. Простота форм, чітке групування елементів та логічна послідовність взаємодії забезпечують позитивний користувацький досвід і мінімізують імовірність помилок. Також важливо зазначити, що використання піктограм та кольорових акцентів підвищує доступність інтерфейсу для користувачів з різним рівнем цифрової грамотності.

Таким чином, представлене вікно пошуку рейсів реалізовано згідно з принципами user-centered design, забезпечує зручність для користувача та відіграє ключову роль у процесі взаємодії з програмним забезпеченням бронювання та продажу авіаквитків.

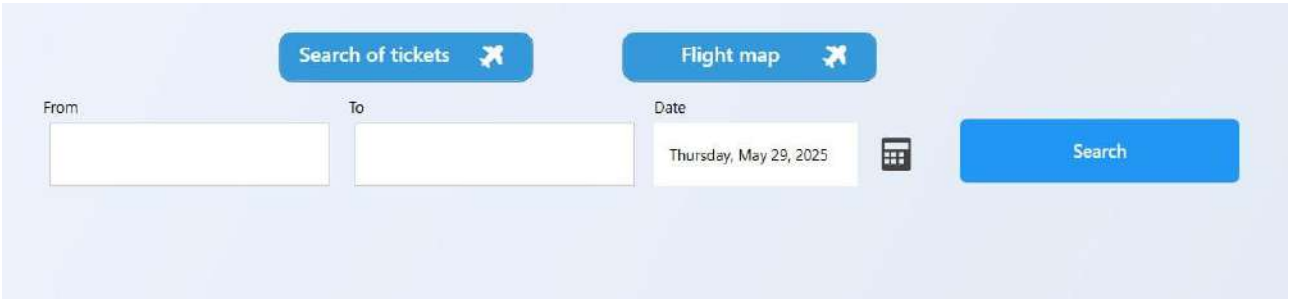

Рис. 4.4. Функція пошуку рейсів за параметрами

На рисунку 4.5 представлено вікно вибору авіарейсу, що виконує функцію пошуку та зміни обраного маршруту в системі онлайн-бронювання квитків. Інтерфейс побудований за принципами зручності, логічної структури та швидкого візуального сприйняття. У верхній частині розташована панель фільтрації, що дозволяє користувачу ввести пункт відправлення, призначення та дату подорожі. Кнопки «Search of tickets» та «Flight map» полегшують навігацію між різними частинами системи, а велика синя кнопка «Search» праворуч від полів введення забезпечує миттєвий запуск пошуку, привертаючи увагу користувача.

Нижче розташована основна частина — результати пошуку у вигляді окремих карток для кожного рейсу. Візуально картки оформлені в стилі мінімалізму: світлий фон, тіні для відділення елементів, гармонійне поєднання шрифтів та кольорів. Кожна картка містить детальну інформацію про авіакомпанію, номер рейсу, маршрут (з вказаними містами), дату та час вильоту й прильоту. Праворуч розміщена ціна рейсу, виділена зеленим кольором для візуального акценту, а також інформація про клас обслуговування. Завершує картку кнопка «More details», яка дозволяє переглянути або змінити більш детальну інформацію про рейс, зокрема місця. Таке розташування та оформлення елементів дозволяє досягти максимальної читаємості й зосередженості користувача на головному — виборі оптимального рейсу.

Такий дизайн дозволяє легко порівнювати доступні варіанти перельоту — кожна картка відокремлена тінню та білим фоном, що створює візуальну ієрархію та забезпечує комфортне сканування очима. Інтерфейс є адаптивним і

підходить як для десктопної версії, так і для мобільних пристроїв. Вся інформація згрупована логічно, без надмірного навантаження — це зменшує ймовірність помилок користувача та підвищує швидкість прийняття рішень. Особливу увагу приділено візуальному балансу між текстом і функціональними елементами: важливі дані (ціна, кнопка дій, маршрут) мають яскраве або контрастне виділення, тоді як допоміжна інформація подається у менш насиченому вигляді.

Крім того, структура карток дозволяє без особливих труднощів масштабувати або динамічно оновлювати дані в залежності від змін попиту, фільтрів. Завдяки такому рішенню цей інтерфейс забезпечує як ефективність з точки зору бізнес-логіки, так і позитивний користувацький досвід.

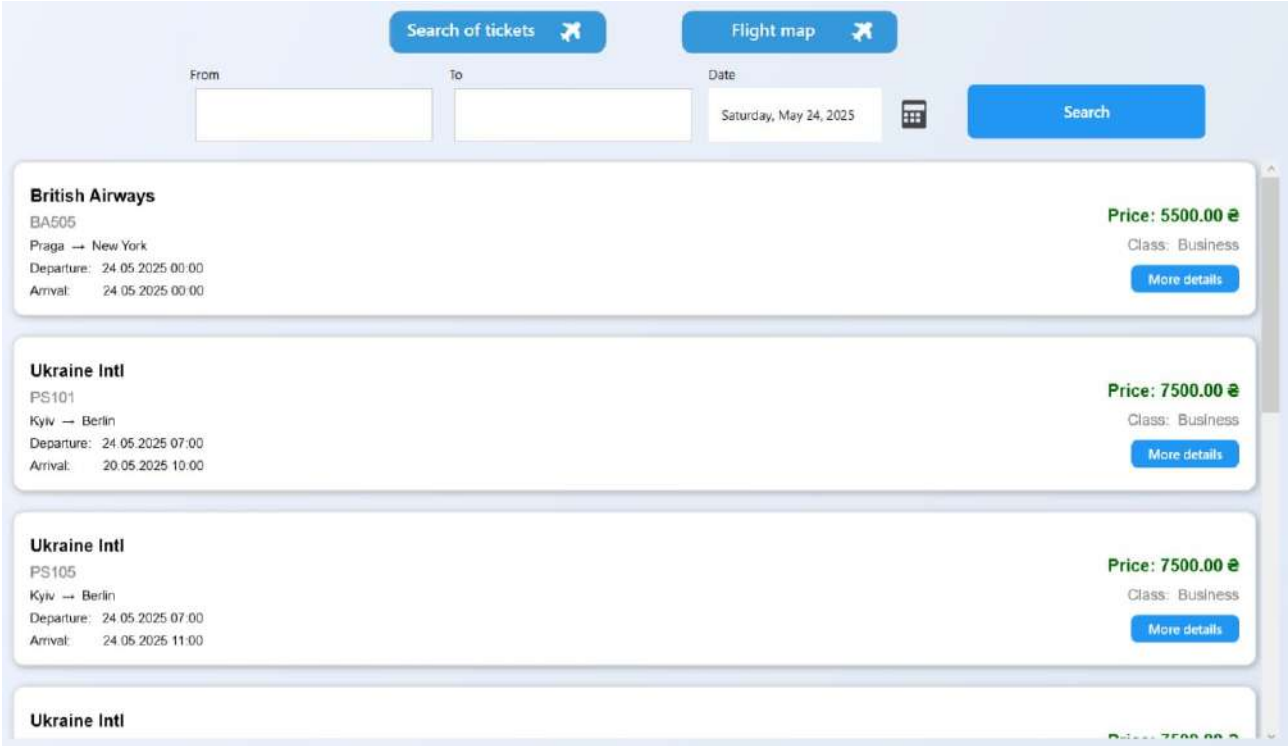

Рис. 4.5. Результат функції пошуку авіарейсів

г) перегляд деталей рейсу:

На рисунку 4.6 представлено вікно вибору місця під час зміни або оформлення авіаквитка, що є важливою частиною процесу бронювання. Інтерфейс побудований за принципами простоти, візуальної ієрархії та зручності у використанні. Основна структура складається з двох логічних блоків: зліва —

детальна інформація про рейс, справа — схема розташування місць у літаку з можливістю вибору.

У верхньому лівому блоці чітко виведені ключові параметри рейсу [Додаток А]:

1. Flight Number;
2. Airline;
3. Route;
4. Departure Time;
5. Arrival Time;
6. Price;
7. Class.

Цей блок дозволяє швидко переглянути всю необхідну інформацію перед вибором місця.

Центральний елемент вікна — візуальна схема салону літака з номерами місць. Вільні місця виділені зеленим кольором, зайняті — сірим, що робить процес вибору інтуїтивно зрозумілим. Таке візуальне кодування дозволяє користувачеві миттєво зорієнтуватися та вибрати бажане місце без додаткових підказок.

Кнопка «Continue» також слугує переходом до фінального етапу оформлення бронювання, що логічно завершує процес вибору місця. Її розміщення та дизайн відповідають сучасним принципам UX-дизайну, спрямованим на мінімізацію зусиль користувача та уникнення плутанини при навігації інтерфейсом.

Завдяки чіткій структурі, візуальній простоті та інтуїтивності, цей інтерфейс дозволяє користувачам легко та швидко завершити критично важливий крок бронювання — обрати комфортне місце для польоту. Крім того, гнучка структура дозволяє легко адаптувати цей інтерфейс під інші класи обслуговування або додаткові послуги.

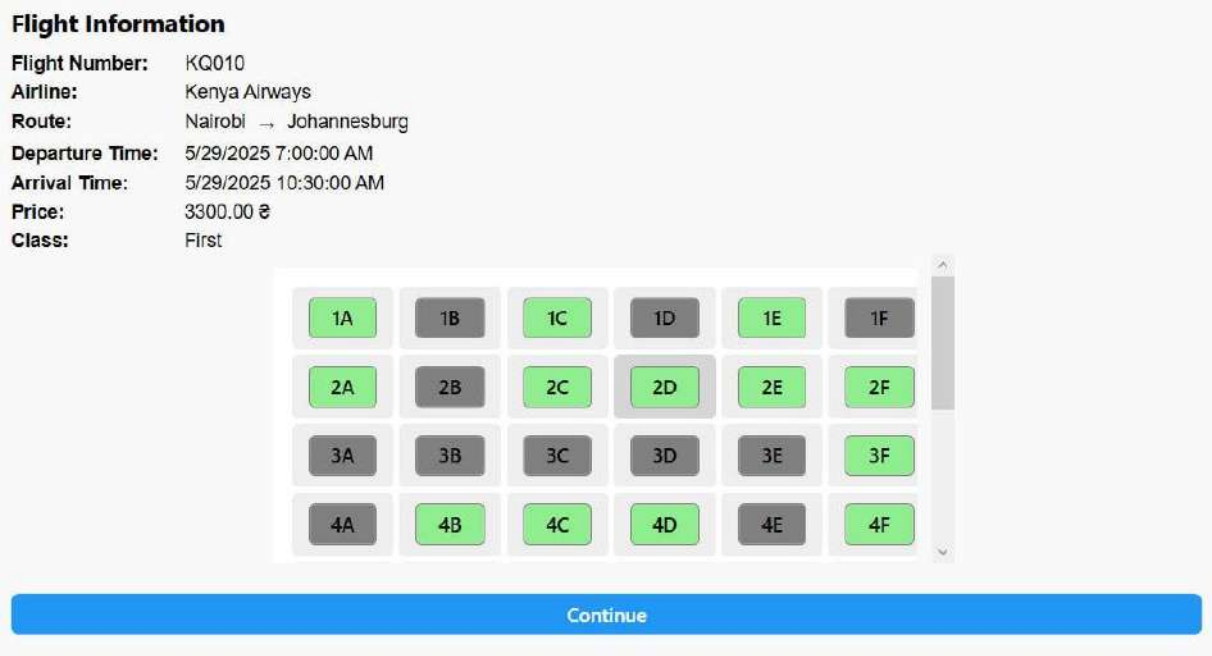

Рис. 4.6. Вікно детальної інформації про рейс та вибір місця

д) бронювання квитка:

На рисунку 4.7 представлено вікно оформлення покупки авіаквитка, яке є ключовим етапом взаємодії користувача з системою бронювання. Дизайн інтерфейсу побудований на принципах простоти, інтуїтивності та зручності. Форма складається з чітко розділених блоків — інформації про рейс, даних пасажира та платіжних реквізитів. Кожне поле супроводжується підписом, що допомагає уникнути помилок при заповненні.

У верхній частині форми відображається детальна інформація про обраний рейс:

1. Flight Number;
2. Airline;
3. Departure;
4. Arrival;
5. Departure Time;
6. Arrival Time;
7. Price;
8. Class.

Далі розміщено блок «Passenger information», де користувач вводить особисті дані [Додаток А]:

1. Full Name
2. Passport
3. Email
4. Phone

Нижче знаходиться блок «Payment Information», який містить обов’язкові для покупки квитка платіжні поля:

1. Card Number
2. Expiry Date (MM/YY)
3. CVV

У нижній частині форми розміщена велика синя кнопка «Buy a ticket», яка виконує роль основного заклику до дії. Вона має яскраве візуальне виділення та розташована окремо від інших елементів форми, що дозволяє уникнути випадкових натискань під час введення даних. Такий дизайн покликаний забезпечити інтуїтивну навігацію та зручність користування навіть для недосвідчених користувачів.

Інтерфейс форми адаптований таким чином, щоб перед підтвердженням покупки користувач міг переглянути всю ключову інформацію про рейс, обране місце, а також без помилок ввести персональні та банківські дані. Це зменшує ймовірність помилок і підвищує довіру до сервісу. Простота та логічність оформлення покупки позитивно впливають на користувацький досвід, знижуючи кількість покинутих транзакцій.

Після заповнення всіх полів та натискання кнопки «Buy a ticket» система автоматично здійснює перевірку введених даних, створює новий запис у таблиці «Bookings», в якому зберігається ідентифікатор відповідного рейсу «FlightId» та місця «SeatId». Користувачу одразу виводиться підтвердження про успішне бронювання, що закріплює відчуття завершеної та вдалої операції.

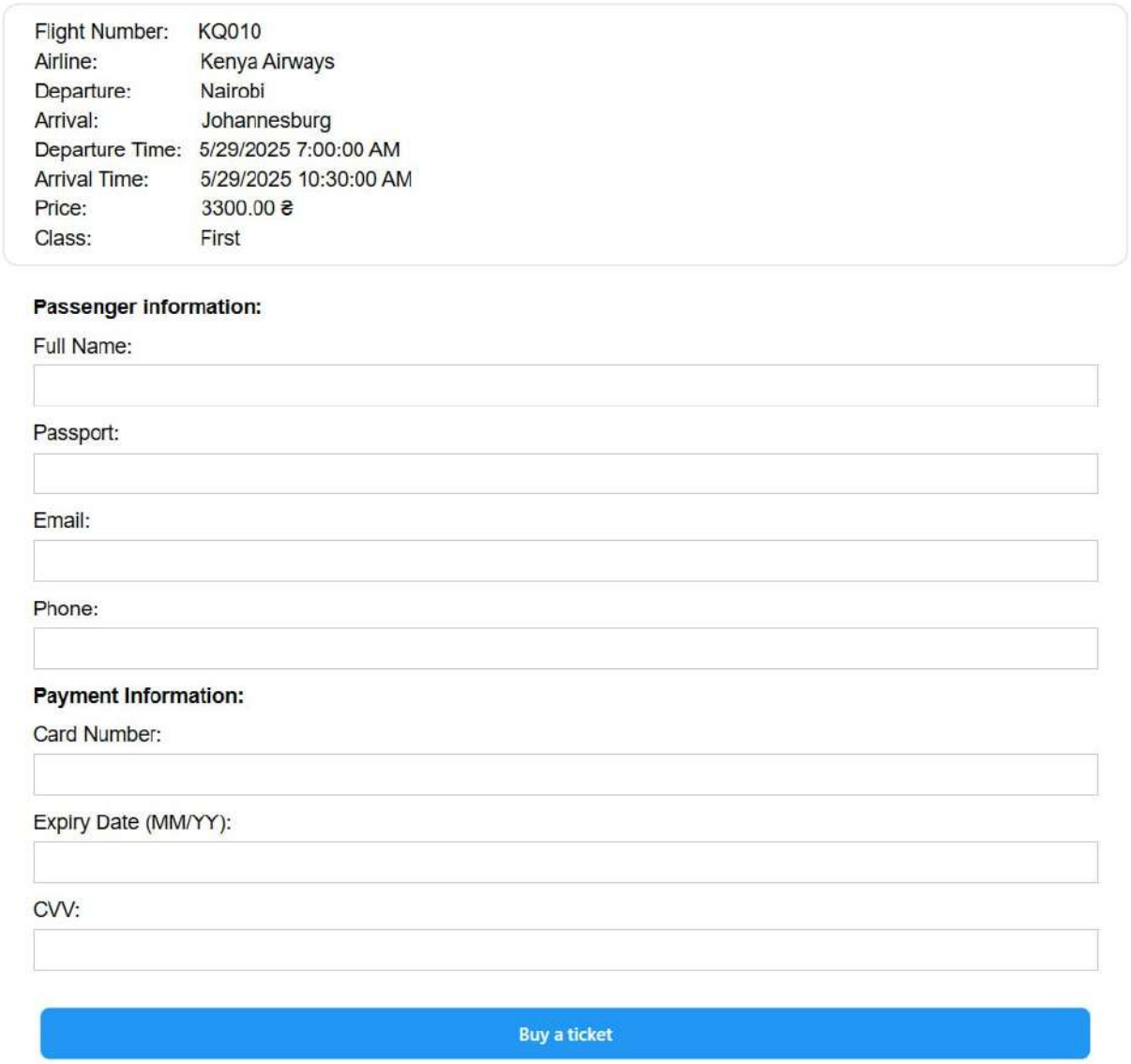

Рис. 4.7. Форма бронювання місця обраного рейсу

е) візуалізація маршрутів на карті:

На рисунку 4.8 показано інтерфейс який відображає світову карту з нанесеними маршрутами авіаперельотів координати яких було задано, що позначені червоними лініями між аеропортами (чорні точки). Над картою розміщено панель із кнопками для пошуку квитків «Search of tickets» та перегляду карти рейсів «Flight map», а також кнопки входу та реєстрації у правому верхньому куті.

Дизайн обрано мінімалістичний і функціональний — основна увага зосереджена на карті, яка відображає зв’язки між обраними пунктами призначення. Завдяки використанню OpenTK, карта реалізована як інтерактивне

2D, що дозволяє візуалізувати великі обсяги маршрутної інформації з гарною продуктивністю [Додаток Г]. Простота стилізації (сірі континенти, блакитне тло) допомагає користувачу не відволікатися від основного — маршрутів і точок призначення. Такий підхід також полегшує адаптацію карти для різних платформ, зберігаючи чіткість та швидкодію.

Це вікно може використовуватись як у процесі бронювання, так і для перегляду вже обраного маршруту користувача, оскільки дозволяє швидко оцінити географічне положення та зв’язки між містами візуально.

Рис. 4.8. Візуалізація рейсів на основі заданих координат

ж) Панель адміністратора:

У разі входу адміністратора у профіль адміністратора додається кнопка (Рис. 4.9) «Admin panel» яка надає можливість адміністратору [Додаток Б]:

1. переглядати всі існуючі рейси;
2. додавати нові рейси;
3. редагувати наявні;
4. видаляти застарілі рейси.

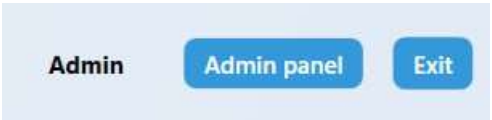

Рис. 4.9. Кнопка входу у панель адміністратора

На рисунку 4.10 представлено головне вікно адміністративної панелі, яке дозволяє адміністратору здійснювати повне управління інформацією про авіарейси, що зберігаються у базі даних системи бронювання авіаквитків.

У центрі вікна розміщена таблиця, яка відображає список усіх наявних рейсів. Кожен рядок відповідає одному рейсу й містить такі поля:

а) номер рейсу (Flight);

б) авіакомпанія (Airline);

в) місто відправлення (From);

г) місто прибуття (To);

д) час відправлення (Departure Time);

е) час прибуття (Arrival Time).

У крайньому правому стовпчику для кожного рейсу розміщені кнопки Edit та Delete:

а) Edit — відкриває форму редагування даних рейсу;

б) Delete — видаляє рейс з бази даних після підтвердження.

У верхній частині вікна розташована кнопка «Add flight», яка дозволяє адміністратору додати новий рейс до списку, заповнивши відповідну форму (Рис. 4.12).

Це вікно є ключовим інструментом адміністратора, оскільки забезпечує централізоване управління рейсами без потреби прямого доступу до бази даних. Воно дозволяє оперативно додавати нові рейси, редагувати існуючі записи, а також видаляти застарілу або некоректну інформацію, що сприяє підтримці високої якості та актуальності даних.

Завдяки зручному візуальному інтерфейсу адміністратор отримує наочний контроль над усіма параметрами рейсів — напрямками, датами, часом тощо. Це значно пришвидшує процес адміністрування в порівнянні з ручним редагуванням у СУБД, знижує ризик помилок і підвищує ефективність роботи персоналу.

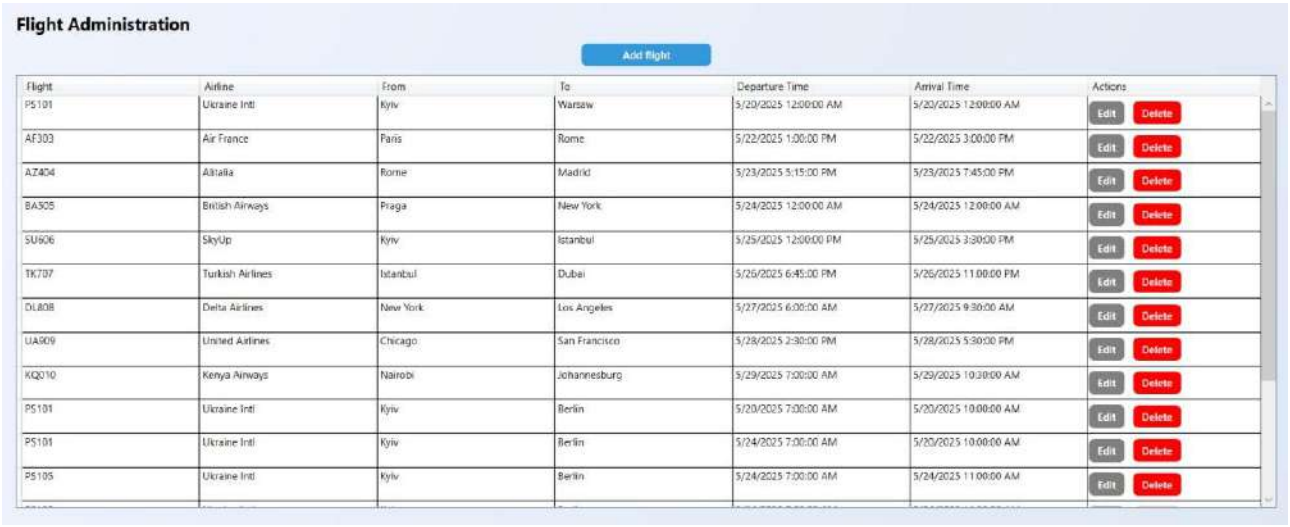

Рис. 4.10. Панель адміністратора

На рисунку 4.11 зображено інтерфейс вікна редагування рейсу, яке відкривається після натискання кнопки «Edit» у таблиці адміністративної панелі. Це вікно дозволяє адміністратору змінити ключові параметри авіарейсу, що вже існує у системі.

Основними елементами інтерфейсу є текстові поля вводу, поля вибору дати та часу та кнопки керування діями:

а) Flight – номер рейсу;

б) Airline – назва авіакомпанії;

в) Departure City – місто відправлення;

г) Arrival City – місто прибуття;

д) Price – вартість квитка;

е) Departure Date – дата та час відправлення;

ж) Arrival Date – дата та час прибуття;

з) Save – зберігає внесені зміни та оновлює запис у базі даних;

і) Cancel – скасовує редагування та повертає до головного вікна без змін.

Цей екран виконує одну з ключових функцій для адміністратора — оновлення вже існуючої інформації про рейси. Наприклад, якщо було змінено розклад, вартість, або потрібно виправити помилку в назві міста — це вікно дозволяє зробити це швидко і зручно.

Інтерфейс зосереджений виключно на потрібних параметрах без зайвих елементів, що дозволяє ефективно управляти авіарейсами та підтримувати дані в актуальному стані.

Рис. 4.11. Вікно змінення обраного рейсу

На рисунку 4.12 показано вікно «Add Flight», яке використовується для створення нового авіарейсу у системі адміністрування. Інтерфейс виконано у мінімалістичному стилі з лаконічною структурою. Усі поля розташовані вертикально в логічній послідовності — від базових характеристик до дат і вартості квитка. Такий підхід дозволяє адміністратору швидко орієнтуватися у формі та не втрачати час на пошук потрібного поля.

Вікно містить такі елементи:

а) Flight – текстове поле для введення унікального номера рейсу.

б) Airline – назва авіакомпанії.

в) Departure City і Arrival City – міста вильоту та прибуття відповідно.

г) Departure Date та Arrival Date – вибір дати та часу за допомогою зручного календаря.

д) Price – вартість квитка.

У нижній частині форми розташовані кнопки «Save» і «Cancel», які дозволяють зберегти новий рейс у базі даних або скасувати дію. Кнопки мають виразний дизайн з відступами, щоб уникнути помилкового натискання.

Інтерфейс розроблений з урахуванням вимог до доступності та зручності користування. Відсутність зайвих елементів дозволяє сконцентруватися на введенні необхідних даних. Крім того, всі поля мають однаковий стиль, що створює відчуття узгодженості інтерфейсу.

Це вікно є ключовим для адміністратора при внесенні нових маршрутів до системи, що дозволяє гнучко оновлювати розклад рейсів, розширюючи можливості пошуку для користувачів.

Рис. 4.12. Вікно додавання нового рейсу

4.3. Розробка інтерфейсу вебсайту бронювання авіаквитків

Під час розробки інформаційної системи бронювання та продажу авіаквитків було створено прототип лендінг-сторінки, що виконує функцію візитівки сервісу та демонструє ключову ідею — простоту взаємодії користувача з системою. Основна мета цього сайту — презентувати можливості платформи, привернути увагу користувача до зручності бронювання, а також підкреслити професійний вигляд сервісу.

Для реалізації серверної частини сайту було використано FastAPI — сучасний асинхронний фреймворк Python, що забезпечує високу продуктивність і зручність розгортання невеликих сайтів та API-сервісів [10], [22]. Він ідеально

підходить для побудови швидких прототипів без потреби у складних налаштуваннях.

Клієнтська частина створена з використанням HTML та CSS, що забезпечує максимальну сумісність з усіма сучасними браузерами. Для зберігання стилів використано зовнішній CSS-файл style.css [11].

Сайт створено в світлій кольоровій гамі, з переважанням блакитних, білих та зелених відтінків, що символізують легкість, довіру та надійність — цінності, що є критичними для авіаперевезень. Головна секція виконана у вигляді гарного зображення на задньому фоні і центральним текстом (Рис. 4.13), що привертає увагу користувача.

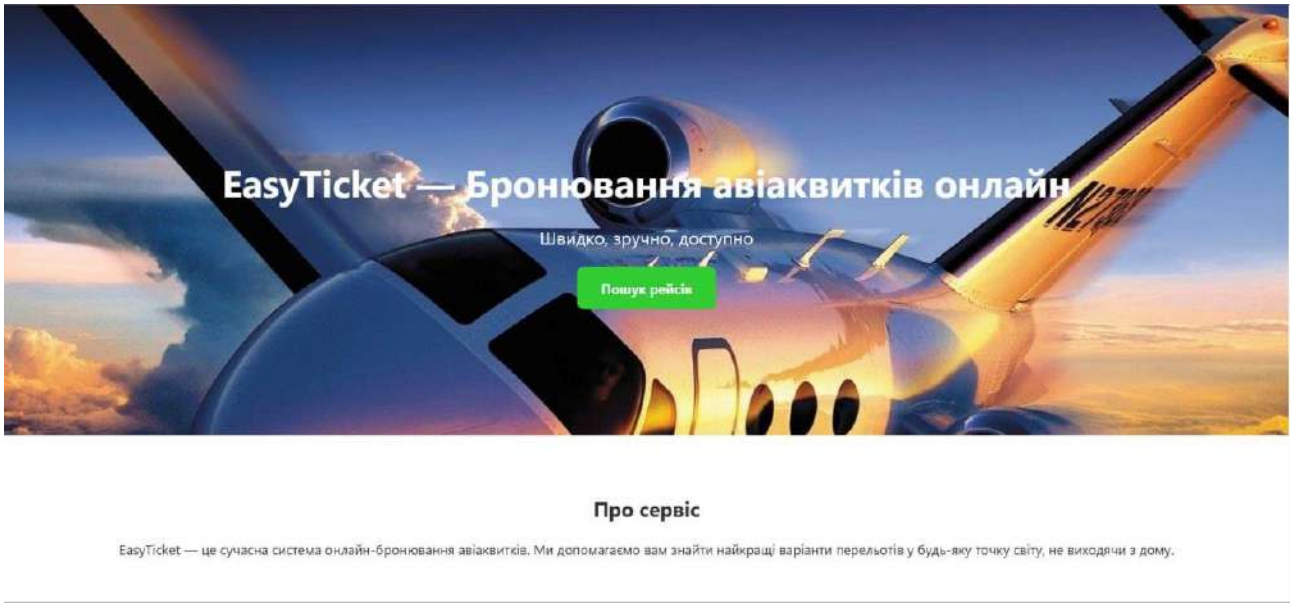

Рис. 4.13. Головна сторінка сайту

а) основні компоненти сторінки:

1. Головна секція з великою назвою, описом та кнопкою дії закликає відвідати систему;
2. блок про сервіс містить короткий опис переваг системи;
3. секція з перевагами оформлена у вигляді карток, кожна з яких підкреслює окрему особливість;
4. секція популярних напрямків імітує список доступних рейсів;

5. блок контактної інформації яка дозволяє користувачеві легко зв’язатися з представниками компанії.

Такий підхід до дизайну дозволяє сформувати інтерфейс, який не потребує додаткових пояснень чи інструкцій для користувача [15, с.450]. Основні елементи, такі як кнопки дій, інформаційні блоки та навігаційні компоненти, мають логічне розташування, зрозумілі написи та візуальне оформлення, яке чітко вказує на їх призначення. Завдяки цьому користувач може одразу орієнтуватися на сторінці, не витрачаючи час на пошук потрібної інформації або розуміння структури сайту. Крім того, використання єдиної кольорової гами та стилістично узгоджених елементів забезпечує комфортне візуальне сприйняття та полегшує навігацію, що загалом зменшує когнітивне навантаження і прискорює процес взаємодії із сайтом.

Окремим елементом головної сторінки є блок, присвячений перевагам сервісу (Рис. 4.14). Його основна мета — швидко та зрозуміло донести до потенційного користувача ключові вигоди від використання саме цієї платформи для пошуку, бронювання та купівлі авіаквитків.

Дизайн блоку виконано в легкому, світлому стилі, що гармонійно поєднується з загальною палітрою сайту. Кожна перевага представлена у вигляді окремої картки, розміщеної на фоні з м’яким відтінком блакитного кольору, який асоціюється з подорожами, небом та комфортом. Така кольорова гама створює довірливу атмосферу та сприяє кращому сприйняттю інформації.

У центрі кожної картки — піктограма, яка візуалізує суть переваги. Іконки виконані в одному стилі, що забезпечує цілісність дизайну. Під кожною піктограмою розміщується короткий заголовок, що стисло описує перевагу, а нижче — невеликий текст із детальнішим поясненням. Такий формат дозволяє користувачу швидко ознайомитися з основними перевагами, не витрачаючи зайвого часу на читання довгих описів.

Інтерактивність блоку реалізована за допомогою візуальних ефектів: при наведенні курсору на картку активується легкий ефект збільшення масштабу або підсвічування, що додає динамічності інтерфейсу. Завдяки цим елементам

дизайн виглядає живим та сучасним, а блок з перевагами — таким, що активно привертає увагу.

Контентно блок містить найбільш релевантну для цільової аудиторії інформацію: простота пошуку рейсів, надійна підтримка, вигідні ціни, зручна система оплати та адаптація до мобільних пристроїв. Ці акценти дозволяють одразу донести до відвідувача основні причини, чому варто скористатися саме цим сервісом.

Завдяки чіткому структурованому викладенню, візуальній привабливості та логічному розташуванню елементів, блок переваг ефективно виконує свою роль — викликає інтерес до сервісу та сприяє формуванню довіри до нього ще до початку активної взаємодії з функціоналом сайту.

Рис. 4.14. Блок з інформацією про переваги

У структурі сайту важливе місце займають картки, в яких представлено ключову інформацію про доступні авіарейси, а також переваги користування сервісом. Дизайн цих елементів продуманий до дрібниць і поєднує в собі як естетичну привабливість, так і функціональність.

Зовнішній вигляд карток оформлений у світлому мінімалістичному стилі з плавними тінями, що створює відчуття легкості та простору. Тінювання є м’яким і неагресивним — воно використовується для візуального відокремлення картки від фону, що робить її помітнішою серед іншого контенту, не порушуючи при цьому загальної гармонії дизайну.

Кожна картка має тонке, але помітне обрамлення блакитного кольору, яке слугує не тільки декоративною деталлю, а й підсилює акценти на важливих

елементах. Блакитний колір був обраний не випадково — він асоціюється з повітрям, небом і подорожами, а також додає свіжості інтерфейсу.

Окрему увагу приділено інтерактивності. При наведенні курсору на картку активується ефект «підняття» — візуальний зсув у просторі, що супроводжується посиленням тіні. Це створює відчуття динаміки та підказує користувачу, що елемент є клікабельним або активним, підвищуючи загальну інтуїтивність взаємодії з інтерфейсом.

Картки мають адаптовану ширину, що дозволяє їм органічно виглядати як на широких екранах комп’ютерів, так і на мобільних пристроях. Завдяки цьому сайт залишається зручним і зрозумілим незалежно від типу пристрою, з якого його переглядає користувач.

Кнопки, які розташовані всередині карток, виділяються насиченим зеленим кольором. Цей відтінок асоціюється з позитивною дією, безпекою та правильним вибором, що психологічно підштовхує користувача до взаємодії. При наведенні або натисканні кнопка змінює свій відтінок — це не тільки додає інтерактивності, а й забезпечує візуальний зворотний зв’язок, що підвищує зручність користування.

Завдяки таким дизайнерським рішенням картки стають центральними інформаційними блоками сторінки. Вони приваблюють увагу, не перевантажуючи її, допомагають швидко орієнтуватися у запропонованих варіантах і сприяють приємному користувацькому досвіду.

Особливу увагу приділено дизайну карток з інформацією про переваги та рейси (Рис. 4.15). Вони мають:

1. м’яке тінювання;
2. обрамлення блакитного кольору;
3. ефект "підняття" при наведенні курсору;
4. адаптовану ширину для гарного відображення на різних пристроях.

Кнопки мають насичений зелений колір, що асоціюється з безпечним вибором, і реагують на дії користувача, змінюючи відтінок.

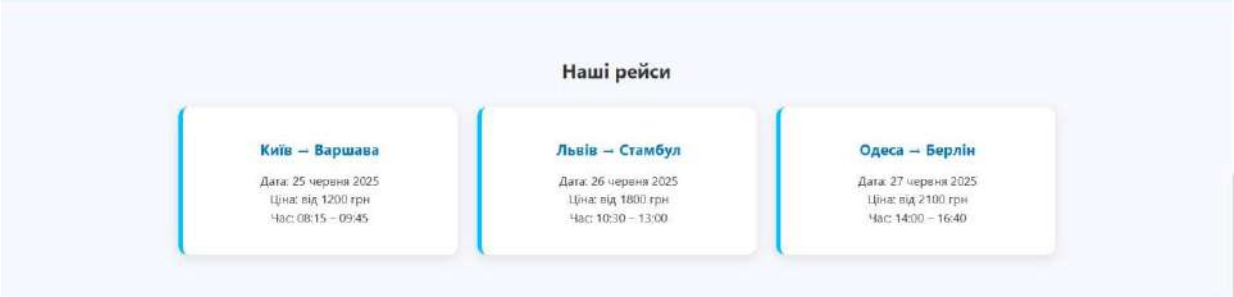

Рис. 4.15. Картки з інформацією про рейси

У нижній частині сторінки, розміщується блок з контактною інформацією (Рис. 4.16), який виконує важливу роль у структурі сайту. Він створений для того, щоб забезпечити зручний і швидкий доступ до каналів зв’язку з компанією, що надає послуги бронювання авіаквитків. У цьому блоці відображено ключові відомості: адресу електронної пошти та контактний номер телефону. Уся інформація структурована компактно та логічно, що дозволяє легко знайти потрібні дані навіть під час швидкого перегляду сторінки.

Візуально блок виконаний у спокійних світло-сірих тонах, які гармонійно поєднуються з загальним стилем сайту та водночас чітко відокремлюють цю частину від основного контенту. Заголовки в блоці виділені напівжирним шрифтом для зручності сприйняття.

Таке оформлення контактної інформації не тільки забезпечує просту навігацію, а й підвищує довіру користувача до сервісу, демонструючи прозорість та відкритість компанії. Крім того, наявність різних способів зв’язку сприяє оперативному реагуванню на запити клієнтів, що позитивно впливає на якість обслуговування.

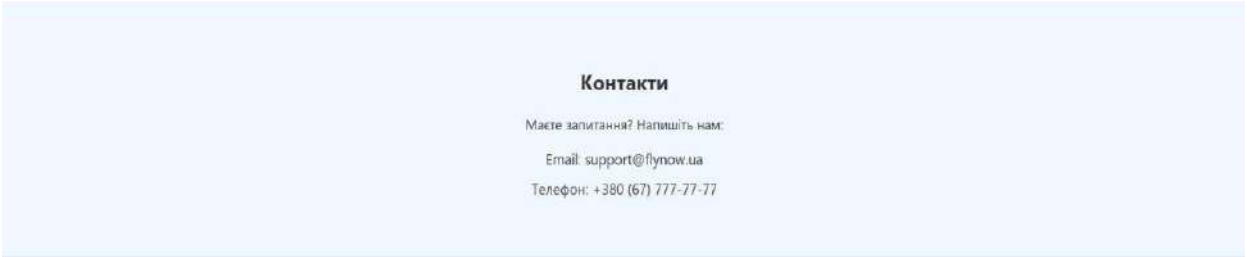

Рис. 4.16. Блок контактної інформації

На відміну від типових шаблонних сторінок, що часто перевантажені візуальними ефектами, великою кількістю кольорів або анімацій, розроблений сайт має мінімалістичний, проте водночас інформативний дизайн, який орієнтований на користувача. Основний акцент зроблено на змісті та простоті навігації, що дозволяє уникнути візуального перевантаження та фокусувати увагу користувача на основних функціях ресурсу — пошуку та бронюванні авіаквитків.

Усі використані візуальні ефекти виконують конкретні прикладні завдання, а не є лише декоративними елементами. Зокрема:

1. Покращення зчитуваності реалізовано шляхом вибору оптимального поєднання кольорів (приглушений фон, темні заголовки, виділені кнопки), контрастного шрифту та достатнього міжрядкового інтервалу. Це дозволяє легко сприймати текстову інформацію навіть на мобільних пристроях або при тривалому перегляді.

2. Виділення головного змісту досягається використанням кольорових акцентів, тіней та відступів. Наприклад, інформаційні блоки з результатами пошуку мають світлий фон і чітку межу, що візуально розділяє окремі картки. Ціна квитка та кнопка дії виділені зеленим кольором — це не лише приваблює погляд, а й підкреслює важливість елемента.

3. Покращення взаємодії з кнопками дій забезпечується за допомогою візуального зворотного зв’язку — при наведенні кнопка змінює колір або накладається легка тінь. Це інтуїтивно сигналізує користувачеві про можливість взаємодії, що підвищує загальний рівень зручності користування.

Таким чином, дизайн поєднує естетику і функціональність, що дозволяє користувачеві швидко орієнтуватися на сайті, легко здійснювати пошук та бронювання, а також отримувати задоволення від взаємодії з сервісом. Завдяки цьому підвищується рівень довіри до ресурсу та залучення потенційних користувачів, які цінують простоту, швидкість та зручність у користуванні онлайн-сервісами.

Висновки до розділу

У даному розділі було детально розглянуто процес розробки програмного забезпечення системи бронювання та продажу авіаквитків. На основі аналізу вимог і попереднього проєктування було реалізовано десктопний застосунок з використанням технологій C#, WPF, Microsoft SQL Server та OpenGL.

Розробка відбувалась згідно з архітектурним підходом MVVM, що забезпечило розділення логіки представлення, бізнес-логіки та доступу до даних, спростивши тестування й масштабування системи. Основні функціональні можливості — реєстрація, авторизація, пошук рейсів, бронювання місць, адміністративне керування рейсами та інтеграція з візуалізацією маршрутів на карті — були реалізовані та протестовані.

Інтерфейс користувача створено з урахуванням зручності й інтуїтивності, що підвищує ефективність взаємодії кінцевого користувача з програмою. Отриманий результат повністю відповідає поставленим вимогам, забезпечуючи стабільну роботу та розширюваність програмного продукту.

ВИСНОВКИ

У ході виконання дипломної роботи було реалізовано повнофункціональну систему бронювання та продажу авіаквитків, що відповідає сучасним вимогам до автоматизації процесів у сфері пасажирських авіаперевезень. В результаті проведеного аналізу предметної області було визначено основні вимоги до системи, розроблено інформаційну модель, спроєктовано структуру бази даних та реалізовано програмне забезпечення на основі технологій C#, WPF, Microsoft SQL Server та OpenGL.

У процесі проєктування було створено зручний користувацький інтерфейс, що забезпечує доступ до основного функціоналу системи: реєстрації та авторизації користувачів, пошуку рейсів, бронювання квитків із вибором місця, перегляду інформації про переліт, а також адміністрування рейсів у відповідному режимі. Додатково реалізовано карту з візуалізацією рейсів по заданих координатах, що підвищує інформативність та інтерактивність застосунку.

Узгодженість логіки роботи системи забезпечено завдяки застосуванню шаблону проєктування MVVM, що дозволило розділити логіку бізнес-процесів, відображення та роботи з даними. Застосування цього підходу покращило структурованість, підтримуваність та масштабованість коду.

Загалом, розроблене програмне забезпечення відповідає поставленим вимогам, демонструє стабільну роботу, має зручний інтерфейс та може бути використане як у навчальних, так і в комерційних цілях після відповідної адаптації.

Окрім десктопного застосунку, також було створено демонстраційний вебсайт у форматі Landing Page за допомогою технології FastAPI та HTML/CSS, що слугує промо-ресурсом для презентації системи. Дизайн сайту виконано у світлих відтінках із використанням плавної анімації та макетування. Він інформує потенційних користувачів про основні можливості програми, переваги використання системи та контактні дані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шилдт Г. — C# 10 і .NET 6. Повний довідник. — М.: Вільямс, 2022. — 1024 с.

2. Петцольд Чарльз. Програмування в Windows з використанням WPF. — Microsoft Press, 2019. — 928 с.

3. Фаулер Мартін. Архітектура корпоративних програм. — К.: Діалектика, 2019. — 560 с.

4. Соммервіл Ієн. Інженерія програмного забезпечення. — К.: Діалектика, 2018. — 832 с.

5. МакДональд Метью. WPF 4.5 у дії. — М.: Видавництво «Вільямс», 2017. — 816 с.

6. Троелсен Ендрю. Pro C# 8.0 and .NET Core 3.0. — Apress, 2020. — 1200 р.

7. Кормен Т. та ін. Алгоритми: побудова та аналіз. — К.: Видавництво «Вільямс», 2021. — 1328 с.

8. Entity Framework Documentation [Електронний ресурс]. — Microsoft Docs. — Режим доступу: https://learn.microsoft.com/en-us/ef/

9. MVVM Design Pattern [Електронний ресурс]. — Microsoft Learn. — https://learn.microsoft.com/en-us/dotnet/desktop/wpf/data/?view=netdesktop-7.0

10. FastAPI Documentation [Електронний ресурс]. — https://fastapi.tiangolo.com

11. HTML Living Standard [Електронний ресурс]. — https://html.spec.whatwg.org/

12. Кузнецов О. І., Мельник Л. М. Бази даних. Проєктування, використання, управління. — Харків: ХНУРЕ, 2020. — 296 с.

13. Барановський В. О. Інформаційні системи та технології в економіці. — К.: КНЕУ, 2019. — 372 с.

14. Шапошніков Д. В. Інженерія програмного забезпечення: курс лекцій. — Х.: ХНЕУ, 2020. — 190 с.

15. Бойко А. І., Кожем’яка О. А. Основи UI/UX-дизайну. — К.: НАУ, 2021. — 220 с.

16. Онищенко В. О. Основи проєктування інформаційних систем. — Київ: КНУТД, 2020. — 320 с.

17. Норт С. Вступ до проектування інтерфейсів. — Львів: Видавництво ЛНУ, 2020. — 128 с.

18. Прокоф'єв С. А. Архітектура програмного забезпечення. — Харків: ХНУРЕ, 2019. — 285 с.

19. Таненбаум А. С. Сучасні операційні системи. — К.: Наука і освіта, 2020. — 1136 с.

20. Лафоре Р. Структури даних і алгоритми в C#. — К.: Діалектика, 2022. — 832 с.

21. Дейтел П., Дейтел Х. Вступ до програмування з C#. — К.: Вільямс, 2020. — 1056 с.

22. Маннінг К. Швидкий старт із FastAPI. — [Електронний ресурс]. — Apress, 2023. — https://www.apress.com/book/9781484294731

23. Бондаренко О. С. Основи веб-розробки. — Київ: КНЕУ, 2021. — 244 с.

24. Курбатов І. М. Системи управління базами даних: теорія та практика. — Харків: ХНУРЕ, 2019. — 310

25. Кнут Дональд. Мистецтво програмування. Том 1: Основні алгоритми. — К.: Діалектика, 2021. — 704 с.

26. Крістіансен Джон. Основи програмування баз даних з Microsoft SQL Server. — К.: Видавництво "Лібра", 2021. — 416 с.

27. Ріхтер Джеффрі. CLR via C#. Програмування на платформі .NET. — М.: Вільямс, 2020. — 896 с.

28. Флойд Ренді. Основи UI/UX-дизайну: створення інтерфейсів, орієнтованих на користувача. — Львів: Видавництво «Альфа», 2022. — 240 с.

29. Швабер Кен. Гнучкі методології розробки програмного забезпечення. — К.: Діалектика, 2017. — 272 с.

ДОДАТКИ

Додаток А

Моделі даних

Файл «User.cs»

```
using System.ComponentModel.DataAnnotations;

namespace AirticketsBookingSystem.Models
{
    public class User
    {
        [Key]
        public int Id { get; set; }

        [Required]
        [MaxLength(100)]
        public string Username { get; set; } = null!;

        [Required]
        public string PasswordHash { get; set; } = null!;

        [MaxLength(50)]
        public string Role { get; set; } = "User";

        public bool IsAdmin => Role == "Admin";
    }
}
```

Файл «FlightModel.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AirticketsBookingSystem.Models
{
    public class FlightModel
    {
        public int Id { get; set; }
```

```
        public string FlightNumber { get; set; } = string.Empty;
        public string Airline { get; set; } = string.Empty;
        public string DepartureCity { get; set; } = string.Empty;
        public string ArrivalCity { get; set; } = string.Empty;
        public DateTime DepartureTime { get; set; }
        public DateTime ArrivalTime { get; set; }
        public decimal Price { get; set; }
        public string Class { get; set; } = "Economy";
        public List<SeatModel> Seats { get; set; } = new();
    }
}
```

Файл «FlightRoute.cs

```
using OpenTK.Mathematics;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AirticketsBookingSystem.Models
{
    public class FlightRoute
    {
        public double StartLatitude { get; set; }
        public double StartLongitude { get; set; }
        public double EndLatitude { get; set; }
        public double EndLongitude { get; set; }
        public float Progress { get; set; }
    }

}
```

Файл «PassengerModel.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;

namespace AirticketsBookingSystem.Models
{
    public class PassengerModel
    {
        public string FirstName { get; set; } = "";
        public string LastName { get; set; } = "";
        public string PassportNumber { get; set; } = "";
    }
}
```

Файл «SeatModel.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Media.Media3D;

namespace AirticketsBookingSystem.Models
{
    public class SeatModel
    {
        public int Id { get; set; }
        public string SeatNumber { get; set; }
        public bool IsAvailable { get; set; }

        public int FlightId { get; set; }
        public FlightModel Flight { get; set; }
    }

}
```

Файл «BookingModel.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
```

Продовження Додатку А

```
using System.Text;
using System.Threading.Tasks;

namespace AirticketsBookingSystem.Models
{
    public class BookingModel
    {
        public int Id { get; set; }
        public string FullName { get; set; } = string.Empty;
        public string PassportNumber { get; set; } = string.Empty;
        public string? Email { get; set; }
        public string? Phone { get; set; }
        public int FlightId { get; set; }
        public int SeatId { get; set; }
        public DateTime BookingDate { get; set; } = DateTime.Now;
    }

}
```

Додаток Б

Контекст бази даних та сервіси

Файл «AppDbContext.cs»

```
using AirticketsBookingSystem.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Media.Media3D;

namespace AirticketsBookingSystem.Services
{
    public class AppDbContext : DbContext
    {
        public DbSet<FlightModel> Flights { get; set; }
        public DbSet<SeatModel> Seats { get; set; }
        public DbSet<BookingModel> Bookings { get; set; }
        public DbSet<User> Users { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {

optionsBuilder.UseSqlServer("Server=MSI;Database=AirticketsDb;Trusted_Connection=True;"
);
        }
    }
}
```

Файл «AuthService.cs»

```
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using AirticketsBookingSystem.Models;

namespace AirticketsBookingSystem.Services
{
```

Продовження Додатку Б

```
    public class AuthService
    {
        private readonly AppDbContext _context;

        public AuthService(AppDbContext context)
        {
            _context = context;
        }

        public bool Register(string username, string password)
        {
            if (_context.Users.Any(u => u.Username == username))
                return false;

            var user = new User
            {
                Username = username,
                PasswordHash = HashPassword(password)
            };

            _context.Users.Add(user);
            _context.SaveChanges();
            return true;
        }

        public User? Login(string username, string password)
        {
            var hash = HashPassword(password);
            return _context.Users.FirstOrDefault(u => u.Username == username &&
u.PasswordHash == hash);
        }

        private string HashPassword(string password)
        {
            using var sha256 = SHA256.Create();
            var bytes = Encoding.UTF8.GetBytes(password);
            var hash = sha256.ComputeHash(bytes);
            return Convert.ToBase64String(hash);
        }
    }
```

```
}
```

Файл «SessionManager.cs»

```
using AirticketsBookingSystem.Models;

namespace AirticketsBookingSystem.Services
{
    public static class SessionManager
    {
        public static bool IsLoggedIn { get; set; }
        public static User? CurrentUser { get; private set; }
        public static string? Role { get; set; }
        public static string? CurrentUsername => CurrentUser?.Username;
        public static void Login(User user)
        {
            IsLoggedIn = true;
            CurrentUser = user;
            Role = user.Role;
        }

        public static void Logout()
        {
            IsLoggedIn = false;
            CurrentUser = null;
            Role = null;
        }
    }
}
```

Файл «RelayCommand.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;
```

Продовження Додатку Б

```
namespace AirticketsBookingSystem.Services
{
    public class RelayCommand : ICommand
    {
        private readonly Action<object?> _execute;
        private readonly Predicate<object?>? _canExecute;

        public RelayCommand(Action<object?> execute, Predicate<object?>? canExecute =
null)
        {
            _execute = execute ?? throw new ArgumentNullException(nameof(execute));
            _canExecute = canExecute;
        }

        public bool CanExecute(object? parameter) => _canExecute == null ||
_canExecute(parameter);

        public void Execute(object? parameter) => _execute(parameter);

        public event EventHandler? CanExecuteChanged
        {
            add { CommandManager.RequerySuggested += value; }
            remove { CommandManager.RequerySuggested -= value; }
        }
    }

}
```

Додаток В

ViewModel-шар (MVVM логіка)

Файл «BaseViewModel.cs»

```
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace AirticketsBookingSystem
{
    public abstract class BaseViewModel : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler? PropertyChanged;

        protected void OnPropertyChanged([CallerMemberName] string? propertyName = null)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

Файл «BookingViewModel.cs»

```
using AirticketsBookingSystem.Services;
using System.Windows.Input;
using System.Windows;

namespace AirticketsBookingSystem.Models
{
    public class BookingViewModel : BaseViewModel
    {
        public FlightModel Flight { get; }
        public SeatModel Seat { get; }
        public BookingModel Bookings { get; }

        public int Id { get; set; }
        public string FullName { get; set; } = string.Empty;
        public string PassportNumber { get; set; } = string.Empty;
        public string? Email { get; set; }
        public string? Phone { get; set; }
        public int FlightId { get; set; }
```

Продовження Додатку В

```
public int SeatId { get; set; }
public DateTime BookingDate { get; set; } = DateTime.Now;

public string InfoText =>
    $"Flight Number: {Flight.FlightNumber}\n" +
    $"Seat Number: {Seat.SeatNumber}\n" +
    $"Departure City: {Flight.DepartureCity} → {Flight.ArrivalCity}\n" +
    $"Departure Time: {Flight.DepartureTime}\n" +
    $"Price: {Flight.Price} ₴";

public ICommand BuyCommand { get; }

public BookingViewModel(FlightModel flight, SeatModel seat)
{
    Flight = flight;
    Seat = seat;
    BuyCommand = new RelayCommand(ExecuteBuy, CanBuy);
}

private bool CanBuy(object? parameter) =>
    !string.IsNullOrWhiteSpace(FullName) &&
    !string.IsNullOrWhiteSpace(Email) &&
    !string.IsNullOrWhiteSpace(Phone) &&
    !string.IsNullOrWhiteSpace(PassportNumber);

private void ExecuteBuy(object? parameter)
{
    Seat.IsAvailable = false;

    System.Windows.MessageBox.Show(
        $"Ticket is successfully booked!\n\n" +
        $"Passenger: {FullName}\n" +
        $"Passport: {PassportNumber}\n" +
        $"Flight Number: {Flight.FlightNumber}, Seat Number: {Seat.SeatNumber}",
        "Success", MessageBoxButton.OK, MessageBoxImage.Information);

    if (parameter is Window w)
        w.Close();
}
```

```
    }
}
```

Файл «SearchFlightsViewModel.cs»

```
using AirticketsBookingSystem;
using AirticketsBookingSystem.Models;
using AirticketsBookingSystem.Services;
using System.Collections.ObjectModel;

namespace AirlineApp.ViewModels
{
    public class SearchFlightsViewModel : BaseViewModel
    {
        public string From { get; set; } = "";
        public string To { get; set; } = "";
        public DateTime Date { get; set; } = DateTime.Today;

        public ObservableCollection<FlightModel> Flights { get; } = new();

        public void SearchFlights()
        {
            using (var context = new AppDbContext())
            {
                IQueryable<FlightModel> query = context.Flights;

                // Фільтрація за датою (тільки дата, без часу)
                query = query.Where(f => f.DepartureTime.Date == Date.Date);

                // Умови за From та To
                bool fromFilled = !string.IsNullOrWhiteSpace(From);
                bool toFilled = !string.IsNullOrWhiteSpace(To);

                if (fromFilled && !toFilled)
                {
                    query    =    query.Where(f    =>    f.DepartureCity.ToLower()    ==
From.ToLower());
                }
                else if (!fromFilled && toFilled)
                {
```

```
                query = query.Where(f => f.ArrivalCity.ToLower() == To.ToLower());
            }
            else if (!fromFilled && !toFilled)
            {
                // нічого додатково не фільтруєм – вже по даті
            }
            else
            {
                // якщо оба поля заповнені – можна фільтрувати по відправленню, та
по прибуттю
                query = query.Where(f => f.DepartureCity.ToLower() == From.ToLower()
                                      && f.ArrivalCity.ToLower() == To.ToLower());
            }

            var results = query.ToList();

            Flights.Clear();
            foreach (var flight in results)
            {
                Flights.Add(flight);
            }
        }
    }
}
```

Файл «UserViewModel.cs»

```
using AirticketsBookingSystem.Models;
using System.ComponentModel;

namespace AirticketsBookingSystem.ViewModels
{
    public class UserViewModel : INotifyPropertyChanged
    {
        private User? _currentUser;
        public User? CurrentUser
        {
            get => _currentUser;
            set
```

```
            {
                if (_currentUser != value)
                {
                    _currentUser = value;
                    OnPropertyChanged(nameof(CurrentUser));
                    OnPropertyChanged(nameof(IsAdmin));
                }
            }
        }

        public bool IsAdmin => CurrentUser?.IsAdmin ?? false;

        public event PropertyChangedEventHandler? PropertyChanged;
        protected void OnPropertyChanged(string name) =>
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
    }
}
```

Файл «FlightDetailsViewModel.cs»

```
using System.Collections.ObjectModel;
using System.Windows.Input;
using AirticketsBookingSystem.Services;
using AirticketsBookingSystem.Views;
using Microsoft.EntityFrameworkCore;

namespace AirticketsBookingSystem.Models
{
    public class FlightDetailsViewModel : BaseViewModel
    {
        public FlightModel FlightInfo { get; }
        public ObservableCollection<SeatModel> Seats { get; set; } = new();

        private SeatModel? _selectedSeat;
        public SeatModel? SelectedSeat
        {
            get => _selectedSeat;
            set
            {
                _selectedSeat = value;
```

Продовження Додатку В

```
            OnPropertyChanged();
            }
        }

        public ICommand NextCommand { get; }
        public FlightDetailsViewModel(FlightModel flight)
        {
            FlightInfo = flight ?? throw new ArgumentNullException(nameof(flight));
            LoadSeatsAsync(flight.Id);
            NextCommand = new RelayCommand(OnNext, CanExecuteNext);
        }

        private async void LoadSeatsAsync(int flightId)
        {
            using (var db = new AppDbContext())
            {
                var seats = await db.Seats
                    .Where(s => s.FlightId == flightId)
                    .ToListAsync();

                Seats = new ObservableCollection<SeatModel>(seats);
                OnPropertyChanged(nameof(Seats));
            }
        }
        private bool CanExecuteNext(object? parameter) => SelectedSeat != null &&
SelectedSeat.IsAvailable;

        private void OnNext(object? parameter)
        {
            if (SelectedSeat == null || !SelectedSeat.IsAvailable)
            {
                System.Windows.MessageBox.Show("Please choose an available seat.");
                return;
            }

            var bookingWindow = new BookingWindow(FlightInfo, SelectedSeat);
            bookingWindow.ShowDialog();
        }
    }
}
```

Додаток Г

Рендеринг карти та конвертери

Файл «FlightMapRenderer.cs»

```
using AirticketsBookingSystem.Models;
using OpenTK.Graphics.OpenGL;
using OpenTK.Mathematics;
using System.Drawing.Imaging;
using System.Drawing;
using System.IO;

namespace FlightMap.Renderer
{
    public class FlightMapRenderer
    {
        private List<FlightRoute> Routes = new();
        private float AnimationSpeed = 0.004f;
        private int mapTextureId;
        private int planeTextureId;

        public FlightMapRenderer()
        {
            InitializeRoutes();
        }

        private void InitializeRoutes()
        {
            Routes.Add(new FlightRoute
            {
                StartLatitude = 60.7700628702647,
                StartLongitude = -90.3095821765758,
                EndLatitude = 67.832815990150465,
                EndLongitude = -9.3692052885631343
            });

            Routes.Add(new FlightRoute
            {
                StartLatitude = 71.5074,
                StartLongitude = -0.1278,
```

Продовження Додатку Г

```
            EndLatitude = 13.7563,
            EndLongitude = 101.5018
        });

        Routes.Add(new FlightRoute
        {
            StartLatitude = 61.8566,
            StartLongitude = 2.3522,
            EndLatitude = 37.5665,
            EndLongitude = 122.9780
        });
    }

    private Vector2 GeoToMap(double latitude, double longitude)
    {
        float x = (float)((longitude + 180.0) / 360.0);
        float y = (float)(1.0 - (Math.Log(Math.Tan(Math.PI / 4 + (latitude * Math.PI
/ 180.0) / 2)) / Math.PI)) / 2.0f;
        return new Vector2(x, y);
    }

    public void Load()
    {
        GL.ClearColor(0.53f, 0.81f, 0.92f, 1.0f);

        mapTextureId = LoadTexture("Assets/world_map.png");
        planeTextureId = LoadTexture("Assets/plane.png");

        GL.Enable(EnableCap.Texture2D);
        GL.Enable(EnableCap.Blend);
        GL.BlendFunc(BlendingFactor.SrcAlpha, BlendingFactor.OneMinusSrcAlpha);
    }

    private static int LoadTexture(string relativePath)
    {
        string fullPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
relativePath);

        if (!File.Exists(fullPath))
            throw new FileNotFoundException($"Texture not found: {relativePath}");
```

```
            using var bitmap = new Bitmap(fullPath);
            int tex = GL.GenTexture();
            GL.BindTexture(TextureTarget.Texture2D, tex);

            BitmapData data = bitmap.LockBits(
                new Rectangle(0, 0, bitmap.Width, bitmap.Height),
                ImageLockMode.ReadOnly,
                System.Drawing.Imaging.PixelFormat.Format32bppArgb);

            GL.TexImage2D(TextureTarget.Texture2D, 0, PixelInternalFormat.Rgba,
                data.Width, data.Height, 0,
                OpenTK.Graphics.OpenGL.PixelFormat.Bgra,
                PixelType.UnsignedByte, data.Scan0);

            bitmap.UnlockBits(data);

            GL.TexParameter(TextureTarget.Texture2D,
TextureParameterName.TextureMinFilter, (int)TextureMinFilter.Linear);
            GL.TexParameter(TextureTarget.Texture2D,
TextureParameterName.TextureMagFilter, (int)TextureMagFilter.Linear);
            GL.TexParameter(TextureTarget.Texture2D,  TextureParameterName.TextureWrapS,
(int)TextureWrapMode.ClampToEdge);
            GL.TexParameter(TextureTarget.Texture2D,  TextureParameterName.TextureWrapT,
(int)TextureWrapMode.ClampToEdge);

            return tex;
        }

        public void Update()
        {
            foreach (var route in Routes)
            {
                route.Progress += AnimationSpeed;
                if (route.Progress > 1.0f)
                    route.Progress = 0.0f;
            }
        }

        private void DrawLine(Vector2 start, Vector2 end, int width, int height)
        {
```

```
    float x1 = start.X * width;
    float y1 = start.Y * height;
    float x2 = end.X * width;
    float y2 = end.Y * height;
    GL.Color3(1.0f, 0.0f, 0.0f);
    GL.Begin(PrimitiveType.Lines);
    GL.Vertex2(x1, y1);
    GL.Vertex2(x2, y2);
    GL.End();
}

public void Render(int width, int height)
{
    GL.Viewport(0, 0, width, height);
    GL.Clear(ClearBufferMask.ColorBufferBit);
    GL.MatrixMode(MatrixMode.Projection);
    GL.LoadIdentity();
    GL.Ortho(0, width, height, 0, -1, 1);
    GL.MatrixMode(MatrixMode.Modelview);
    GL.LoadIdentity();

    // Відображення карти
    DrawTexture(mapTextureId, 0, 0, width, height);

    foreach (var route in Routes)
    {
        Vector2 start = GeoToMap(route.StartLatitude, route.StartLongitude);
        Vector2 end = GeoToMap(route.EndLatitude, route.EndLongitude);
        GL.Disable(EnableCap.Texture2D);
        GL.LineWidth(2.0f);
        DrawLine(start, end, width, height);
        GL.Enable(EnableCap.Texture2D);
    }

    // Відображення літаків
    foreach (var route in Routes)
    {
        Vector2 start = GeoToMap(route.StartLatitude, route.StartLongitude);
        Vector2 end = GeoToMap(route.EndLatitude, route.EndLongitude);
        Vector2 pos = Vector2.Lerp(start, end, route.Progress);
```

Продовження Додатку Г

```
                float x = pos.X * width;
                float y = pos.Y * height;

                Vector2 dir = start - end;
                float angleRad = MathF.Atan2(-dir.Y, dir.X) - MathF.PI / 2f;
                float angleDeg = angleRad * (180f / MathF.PI);

                GL.PushMatrix();
                GL.Translate(x, y, 0);
                GL.Rotate(angleDeg, 0, 0, 1);
                GL.Translate(-10.5f, -10.5f, 0);
                DrawTexture(planeTextureId, 0, 0, 21, 21);
                GL.PopMatrix();
            }
        }
        private void DrawTexture(int textureId, int x, int y, int width, int height)
        {
            GL.BindTexture(TextureTarget.Texture2D, textureId);
            GL.Color3(1.0f, 1.0f, 1.0f);
            GL.Begin(PrimitiveType.Quads);
            GL.TexCoord2(0, 0); GL.Vertex2(x, y);
            GL.TexCoord2(1, 0); GL.Vertex2(x + width, y);
            GL.TexCoord2(1, 1); GL.Vertex2(x + width, y + height);
            GL.TexCoord2(0, 1); GL.Vertex2(x, y + height);
            GL.End();
        }
    }
}
```

Файл «BoolToVisibilityConverter.cs»

```
using System.Globalization;
using System.Windows;
using System.Windows.Data;
namespace AirticketsBookingSystem.Converters
{
    public class BoolToVisibilityConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
```

```
            {
                if (value is bool boolValue)
                    return boolValue ? Visibility.Visible : Visibility.Collapsed;
                return Visibility.Collapsed;
            }

            public object ConvertBack(object value, Type targetType, object parameter,
    CultureInfo culture)
            {
                if (value is Visibility visibility)
                    return visibility == Visibility.Visible;
                return false;
            }
        }
}
```

Файл «SeatAvailabilityConverter.cs»

```
using System.Globalization;
using System.Windows.Data;
using System.Windows.Media;

namespace AirticketsBookingSystem.Converters
{
    public class SeatAvailabilityConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            bool isAvailable = (bool)value;
            return isAvailable ? new SolidColorBrush(Colors.LightGreen) : new
SolidColorBrush(Colors.Gray);
        }
        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}
```