

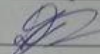
Вінницький національний технічний університет  
Факультет інтелектуальних інформаційних технологій та автоматизації  
Кафедра автоматизації та інтелектуальних інформаційних технологій

**МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**


на тему:

**«Розробка автоматизованої системи автовокзалу»**

Виконав: студент 2-ого курсу, групи ЗАКІТ-22м  
спеціальності 151 – Автоматизація та  
комп'ютерно-інтегровані технології

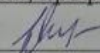
 Дмитро ДІЛИК

Керівник: к.т.н., доцент каф. КСУ

 Олег КОВАЛЮК

« 13 » грудня 2023 р.

Опонент: к.т.н., проф. каф. КН

 Олег КОЛЕСНИЦЬКИЙ

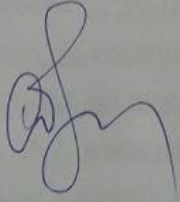
« 13 » грудня 2023 р.

**Допущено до захисту**

Зав. кафедри АІТ

д.т.н., проф. Олег БІСІКАЛО

« 13 » грудня 2023 р.

  
Вінниця ВНТУ - 2023 рік

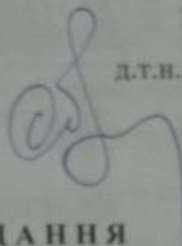
Вінницький національний технічний університет  
Факультет інтелектуальних інформаційних технологій та автоматизації  
Кафедра автоматизації та інтелектуальних інформаційних технологій  
Рівень вищої освіти \_\_\_\_\_ II-й (магістерський)  
Галузь знань – \_\_\_\_\_ 15 Автоматизація та приладобудування  
Спеціальність \_\_\_\_\_ 151 Автоматизація та комп'ютерно-інтегровані технології  
Освітня програма – Інформаційні системи і інтернет речей

**ЗАТВЕРДЖУЮ**

Завідувач кафедри АІТ

д.т.н., проф. Олег БІСІКАЛО

«20» вересня 2023 року



### **ЗАВДАННЯ**

#### **НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**

Дідику Дмитру Максимовичу

- Тема роботи: Розробка автоматизованої системи автовокзалу  
Керівник роботи к.т.н., доцент кафедри КСУ Олег Ковалюк \_\_\_\_\_,  
Затверджені наказом ВНТУ від «18» 09 2023 року № 247
- Термін подання студентом роботи 05.12.2023 р.
- Вихідні дані до роботи архітектура системи; дані про квитки збережені в базі даних; програмне середовище Visual Studio; використання кодування під час передачі даних – так; Windows 10; доступність 24/7; максимальний час обробки запиту – 2 секунди; максимальна кількість одночасних запитів – 10 тис;
- Зміст текстової частини: вступ; аналіз предметної області; проектування автоматизованої системи автовокзалу; розробка автоматизованої системи автовокзалу; висновки; перелік використаних джерел.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):  
 UML-діаграма варіантів використання, UML-діаграма діяльності, UML-діаграма розгортання, UML-діаграма класів, архітектура системи, архітектура системи в Studio, опис структури бази даних, функція фільтру квитків, бронювання квитків, перевірка бронювання квитків у системі, відміна бронювання квитків, перевірка функції відміни бронювання квитків, перевірка роботи чат-боту.

6. Консультанти розділів роботи

Розділ	Ім'я Прізвище	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Ковалюк О.О.	21.09.2023р. <i>[підпис]</i>	08.10.2023р. <i>[підпис]</i>

7. Дата видачі завдання 21.09.2023р.

**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітки
1	Аналіз предметної області	21.09.2023 р. – 1.10.2023 р.	<i>[підпис]</i>
2	Проектування автоматизованої системи автовокзалу	2.10.2023 р. – 18.10.2023 р.	<i>[підпис]</i>
3	Розробка автоматизованої системи автовокзалу	20.10.2023 р. – 15.11.2023 р.	<i>[підпис]</i>
4	Оформлення пояснювальної записки, графічного матеріалу і презентації	16.11.2023 р. – 30.11.2023 р.	<i>[підпис]</i>
5	Попередній захист роботи	21.11.2023 р.	<i>[підпис]</i>
6	Остаточний захист роботи	15.12.2023 р.	<i>[підпис]</i>

Студент

*[підпис]*

Дмитро ДІДИК

Керівник роботи

*[підпис]*

Олег КОВАЛЮК

## АНОТАЦІЯ

УДК 681.3.06.018.1

Дідик Д. М. Розробка автоматизованої системи автовокзалу. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології. Інформаційні системи і інтернет речей. Вінниця: ВНТУ, 2023. 134 с.

На укр. мові. Бібліогр.: 53 назви; рис.: 55.

Магістерська робота присвячена розробці та впровадженню автоматизованої системи для управління та обслуговування автовокзалів. Досліджуючи існуючі системи у сфері транспорту, було виявлено необхідність впровадження інноваційних підходів, які забезпечать оптимізацію рейсів, покращення обслуговування пасажирів та зменшення витрат. У межах даної роботи проведено аналіз вимог та очікувань користувачів, розглянуто різні методи розв'язання задачі автоматизації автовокзалу.

Результатом досліджень є розробка архітектури системи, яка враховує специфіку автовокзалів та включає в себе модулі для обробки бронювань квитків, управління рейсами та взаємодії з пасажирами. Додатково вивчено можливості використання технологій штучного інтелекту для оптимізації процесів.

Розроблена система покликана підвищити ефективність управління автовокзалом, забезпечити зручність для пасажирів та допомогти вирішувати актуальні завдання у сфері транспорту. Ця магістерська робота спрямована на внесення позитивного вкладу в сучасність та підтримку розвитку транспортної інфраструктури.

Ключові слова: автоматизація, автоматизована система, ефективність обслуговування пасажирів, автовокзал.

## ANOTATION

Didyk D. M. Development of an automated bus station system. Master's thesis on the specialty 151 - Automation and computer-integrated technologies. Information systems and the Internet of Things. Vinnytsia: VNTU, 2023. 134 p.

In Ukrainian language. Bibliography: 53 titles; fig.: 55.

The master's thesis is devoted to the development and implementation of an automated system for the management and maintenance of bus stations. Researching the existing systems in the field of transport revealed the need to implement innovative approaches that will ensure flight optimization, passenger service improvement and cost reduction. Within the framework of this work, an analysis of the requirements and expectations of users was carried out, various methods of solving the task of automating the bus station were considered.

The result of the research is the development of the system architecture, which takes into account the specifics of bus stations and includes modules for processing ticket reservations, flight management and interaction with passengers. In addition, the possibility of using artificial intelligence technologies to optimize processes was studied.

The developed system is designed to increase the efficiency of bus station management, provide convenience for passengers and help solve urgent tasks in the field of transport. This master's work is aimed at making a positive contribution to modernity and supporting the development of transport infrastructure.

Keywords: automation, automated system, passenger service efficiency, bus station.

## ЗМІСТ

ВСТУП.....	8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ОБСЛУГОВУВАННЯ ПАСАЖИРІВ.....	11
1.1 Аналіз проблеми розробки автоматизованої системи автовокзалу: автоматизації процесу пошуку та бронювання квитків, застосування модуля, який використовує AI. ....	11
1.2 Дослідження стану розв’язання науково-технічної задачі.....	13
1.2.1 Аналіз існуючих автоматизованих систем у сфері транспорту. Класифікація систем. ....	13
1.2.2 Вибір і обґрунтування сучасних методів розв’язання науково-технічної задачі.....	22
1.2.3 Варіантний аналіз можливих нових методів розв’язання задачі.....	24
1.3 Висновки.....	27
2 ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ АВТОВОКЗАЛУ.....	28
2.1 Аналіз функцій системи.....	28
2.2 Проектування архітектури системи.....	32
2.3 Аналіз засобів створення програмного забезпечення для розробки автоматизованої системи автовокзалу.....	38
2.4 Вибір засобів розробки автоматизованої системи автовокзалу.....	41
2.4.1 Вибір мови програмування.....	41
2.4.2 Вибір системи управління базами даних (СУБД).....	43
2.4.3 Вибір фреймворків.....	45
2.5 Висновки.....	48
3 РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ АВТОВОКЗАЛУ.....	49

3.1 Створення класів доменної області (бізнес-логіки).....	51
3.2 Створення бази даних на основі описаних класів із використанням Entity Framework та підходу Code first.....	56
3.3 Написання логіки фільтрування, відображення квитків на табло та адміністрування квитків .....	59
3.4 Опис модуля який використовує AI. ....	65
3.5 Тестування розробленої системи .....	67
3.6 Висновки.....	81
ВИСНОВКИ .....	83
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ .....	85
ДОДАТКИ .....	90
Додаток А (обов'язковий). Технічне завдання .....	91
Додаток Б (обов'язковий). Ілюстративна частина.....	94
Додаток В (обов'язковий). Лістинг програми .....	102
Додаток Г (обов'язковий). Протокол перевірки МКР.....	134

## ВСТУП

Розробку автоматизованої системи для автовокзалу на сучасному етапі неможливо уявити без згадки про інформаційну та мікропроцесорну революцію, а також про процеси автоматизації, які ці революції спричинили.

Інформаційна революція, яка розпочалася в другій половині 20-го століття, привела до радикальних змін у способах обробки, зберігання та передачі інформації. Введення комп'ютерів, інтернету та інших інформаційних технологій змінило парадигму управління бізнесом та громадськими послугами [1]. В контексті автовокзалів, інформаційна революція розширила можливості для збору та аналізу даних про пасажиропотік, розклади руху, та багато інших параметрів, що впливають на ефективність та комфорт обслуговування.

Мікропроцесорна революція, в свою чергу, призвела до мініатюризації та зниження вартості електроніки, що дозволило впроваджувати автоматизовані системи та пристрої в широкому спектрі сфер, включаючи транспортні вузли. Мікропроцесори стали доступними для використання в сенсорах, контролерах, та інших пристроях, що можуть бути використані для управління автовокзалом. Це відкрило нові можливості для автоматизації процесів, забезпечуючи більшу точність, швидкість та надійність [2].

Автоматизація, в свою чергу, стала ключовим фактором у вдосконаленні управління та обслуговування автовокзалів. Системи автоматизації можуть контролювати рух транспорту, контролювати стан безпеки, регулювати системи кондиціонування, управляти електронними розкладами та інформаційними вивісками, а також забезпечувати пасажирів і клієнтів актуальною інформацією [3][4].

Сучасний світ стрімко розвивається, та відкриває безмежні можливості у використанні інформаційних технологій для поліпшення роботи транспортних вузлів, таких як автовокзали. Актуальність цієї роботи визначається рядом об'єктивних факторів.



По-перше, автовокзали відіграють важливу роль у транспортній інфраструктурі країни, області чи міста, бо вони є важливими вузлами для перевезення пасажирів та вантажів. При цьому, сучасні пасажирів ставлять високі вимоги до комфорту та якості обслуговування. Автоматизована система може допомогти вдосконалити ці процеси.

По-друге, з розвитком технологій і зростанням конкуренції в транспортній галузі, автовокзали повинні ставати більш високоефективними і конкурентоспроможними. Автоматизована система дозволить ефективніше управляти ресурсами, моніторити пасажиропотік, інтегрувати різні послуги та покращити зв'язок з клієнтами.

По-третє, безпека та контроль є ключовими аспектами в роботі автовокзалів. Автоматизована система може допомогти вдосконалити процеси виявлення та запобігання ризикам, забезпечуючи надійну та безпечну роботу вокзалу.

Отже, розробка автоматизованої системи для автовокзалу важлива не тільки для покращення якості обслуговування пасажирів, але й для збільшення ефективності, зниження ризиків та підвищення конкурентоспроможності.

**Метою дослідження** є поліпшення якості обслуговування, підвищення ефективності та забезпечення конкурентоспроможності системи автовокзалу.

**Об'єктом дослідження** є автовокзал, який є транспортним вузлом для перевезення пасажирів і вантажів, а також місцем надання різноманітних послуг і забезпечення комфорту пасажирів та користувачів.

**Предметом дослідження** є процес видачі, пошуку та бронювання квитків для автоматизованої системи автовокзалу.

**Новизна (Інноваційність) роботи** полягає в унікальному підході до вдосконалення автовокзалу за допомогою сучасних технологій та комплексної оптимізації різних аспектів його функціонування, а саме створення програмного забезпечення для цього завдання об'єднує наступні особливості в єдиній системі, що може значно полегшити життя користувачам та оптимізувати процеси управління автовокзалом: широкий функціонал системи, зручний доступ, онлайн бронювання

квитків, можливість легкого інтегрування з іншими системами, мобільний доступ до системи, сучасний дизайн та застосування модуля із AI.

**Практична цінність** полягає в розробці програмного забезпечення та алгоритмів роботи системи автовокзалу.

**Апробація.** Представлені в роботі результати апробовані в результаті участі в конференції «НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ ФАКУЛЬТЕТУ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА АВТОМАТИЗАЦІЇ»:

**Публікації:** Дідик Д. М., Ковалюк О. О. «Розробка автоматизованої системи автовокзалу», «НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ ФАКУЛЬТЕТУ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА АВТОМАТИЗАЦІЇ», 2023. [53]

## **1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ОБСЛУГОВУВАННЯ ПАСАЖИРІВ.**

1.1 Аналіз проблеми розробки автоматизованої системи автовокзалу: автоматизації процесу пошуку та бронювання квитків, застосування модуля, який використовує AI.

Розвиток пошуку інформації є історією технологічних досягнень і прогресу в інформаційній сфері. Спочатку пошук інформації обмежувався ручним пошуком у друкованих джерелах та бібліотечних каталогах. Однак з появою комп'ютерів та Інтернету цей процес різко змінився.

З появою Інтернету прийшла ера інтернет-пошукових систем, таких як Google, які стали незамінними інструментами для мільйонів користувачів щоденно. Вони використовують складні алгоритми для індексації мільярдів веб-сторінок і надають можливість користувачам швидко знаходити необхідну інформацію за допомогою ключових слів [5].

Спостерігається також розвиток семантичного пошуку, коли пошукові системи розуміють не лише окремі ключові слова, але і контекст, співвідношення та значення слів. Це дозволяє користувачам отримувати більш точні та відповідні результати пошуку [6].

Пошук інформації також стає все більше персоналізованим, з урахуванням історії пошукових запитів та інших факторів користувача для надання більш точних рекомендацій та результатів.

Також, в даний момент, відбувається розвиток голосового пошуку, де користувачі можуть шукати інформацію за допомогою голосових команд існуючих асистентів, таких як Siri, Google Assistant та ін. [7].

У майбутньому розвиток пошуку інформації може бути пов'язаний зі штучним інтелектом, машинним навчанням та використанням великих обсягів даних для ще більш точних та персоналізованих результатів.

Розвиток пошуку інформації в контексті розробки автоматизованої системи автовокзалу є надзвичайно важливим аспектом. Автоматизована система автовокзалу має передбачати ефективний інтерфейс для пасажирів та персоналу, а також надавати доступ до інформації про розклади, наявність квитків, стан транспорту та багато іншої важливої інформації.

Завдяки розвитку пошуку інформації, користувачі можуть легко та швидко знаходити необхідну інформацію про рейси, доступність квитків, маршрути та інше. Важливо мати потужну пошукову систему, яка враховує різні параметри, такі як дати, місця, типи квитків та інші, щоб користувачі могли отримувати релевантні результати.

Також може бути важливим розробляти інтерфейс для голосового пошуку, який дозволить користувачам шукати інформацію голосовими командами через мобільні додатки або голосових асистентів.

Усі ці розвитки у галузі пошуку інформації допоможуть покращити функціональність та зручність автоматизованої системи автовокзалу, забезпечуючи швидкий і зручний доступ до важливої інформації для пасажирів та персоналу.

Детальний аналіз проблеми розробки автоматизованої системи автовокзалу передбачає врахування багатьох факторів та викликів, які можуть виникнути в ході цього процесу. Головні проблеми в цьому контексті включають наступне:

- Системна складність: Розробка і впровадження автоматизованої системи автовокзалу є складним і інтегрованим процесом. Це вимагає співпраці різних фахівців і технічних знань в різних галузях, таких як інформаційні технології, транспорт, безпека, обслуговування пасажирів і багато інших.
- Фінансові витрати: Розробка та впровадження автоматизованої системи може бути дорогим процесом. Власники та оператори автовокзалу повинні бути готові інвестувати в цей проект, щоб отримати практичну цінність та вигоди в майбутньому.
- Сумісність з існуючими системами: Якщо на автовокзалі вже існують інші інформаційні системи, важливо забезпечити сумісність та інтеграцію нової системи з

ними. Це може створити складнощі та вимагати додаткових зусиль для успішного впровадження.

- **Безпека і конфіденційність даних:** Збір і обробка чутливої інформації про пасажирів та транспорт може створити проблеми з безпекою і конфіденційністю даних. Важливо вживати заходів для забезпечення захисту цих даних від несанкціонованого доступу.
- **Опозиція персоналу:** Впровадження нових технологій та автоматизації може спричинити опозицію серед персоналу, який може боятися втрати робочих місць або навчання новим навичкам.
- **Пасажирські очікування:** Пасажири можуть мати високі очікування стосовно якості обслуговування та доступності інформації через нову систему, і неспроможність відповісти на ці очікування може викликати негативну реакцію.
- **Технічні проблеми та збої:** Автоматизовані системи можуть стикатися з технічними проблемами та збоями, які можуть впливати на нормальну роботу автовокзалу і обслуговування пасажирів [8].

## 1.2 Дослідження стану розв'язання науково-технічної задачі

1.2.1 Аналіз існуючих автоматизованих систем у сфері транспорту. Класифікація систем.

Ефективне функціонування пасажирського транспорту є надзвичайно важливим аспектом розвитку будь-якого міста. Проблеми перевезення пасажирів виникають в практично кожному місті і стають актуальними з ростом та розвитком населеного пункту. Основна суть цих проблем полягає в необхідності забезпечити громадянам можливість зручного та надійного переміщення на великі відстані в потрібний час.

Розширення міст та збільшення густини населення призводять до надмірного навантаження на громадський транспорт. Це викликає затори, переповнення транспортних мереж, затримки та незручності для пасажирів. Постійний рух та можливість користувачам отримувати транспортні послуги стають ключовими факторами для забезпечення функціональності та життєздатності сучасного міста.

Розвиток та вдосконалення громадського транспорту, а також впровадження інноваційних рішень в управлінні та організації перевезень, є важливими завданнями для влади та громади. Це дозволить зменшити транспортні проблеми, підвищити якість життя мешканців та сприяти сталому розвитку міста [9].

Автоматизовані системи для автовокзалів і транспортних послуг є комплексними інформаційними рішеннями, спрямованими на полегшення та оптимізацію управління різними аспектами функціонування автовокзалів та забезпечення комфорту та ефективності обслуговування пасажирів. Ці системи включають у себе різноманітні модулі та функції, включаючи управління розкладами, бронюванням квитків, контролем доступу, обробкою багажу, веденням документації та багато інших. Вони спроектовані для підвищення продуктивності та забезпечення зручності та безпеки пасажирів і персоналу. Такі системи використовують передові технології, включаючи IoT, штучний інтелект, аналітику даних та мобільні додатки, щоб забезпечити інтегрований та ефективний підхід до управління та обслуговування транспортного вузла.

Існують різні автоматизовані системи та платформи для управління автовокзалами та транспортними послугами. Нижче вказані різні приклади існуючих систем за типами послуг які вони надають своїм користувачам:

1. Розклади та бронювання квитків:

- Amadeus: Amadeus надає програмне забезпечення для автоматизації розкладів, бронювання квитків та управління рейсами для авіакомпаній та автовокзалів. Компанія створює стратегічні рішення, які допомагають авіакомпаніям та аеропортам, готелям та залізницям, пошуковим системам, турагенствам та туроператорам, а також

іншим учасникам туристичного бізнесу проводити свої операції та покращувати якість сервісу у численних поїздках протягом усього року по всьому світу. [10]

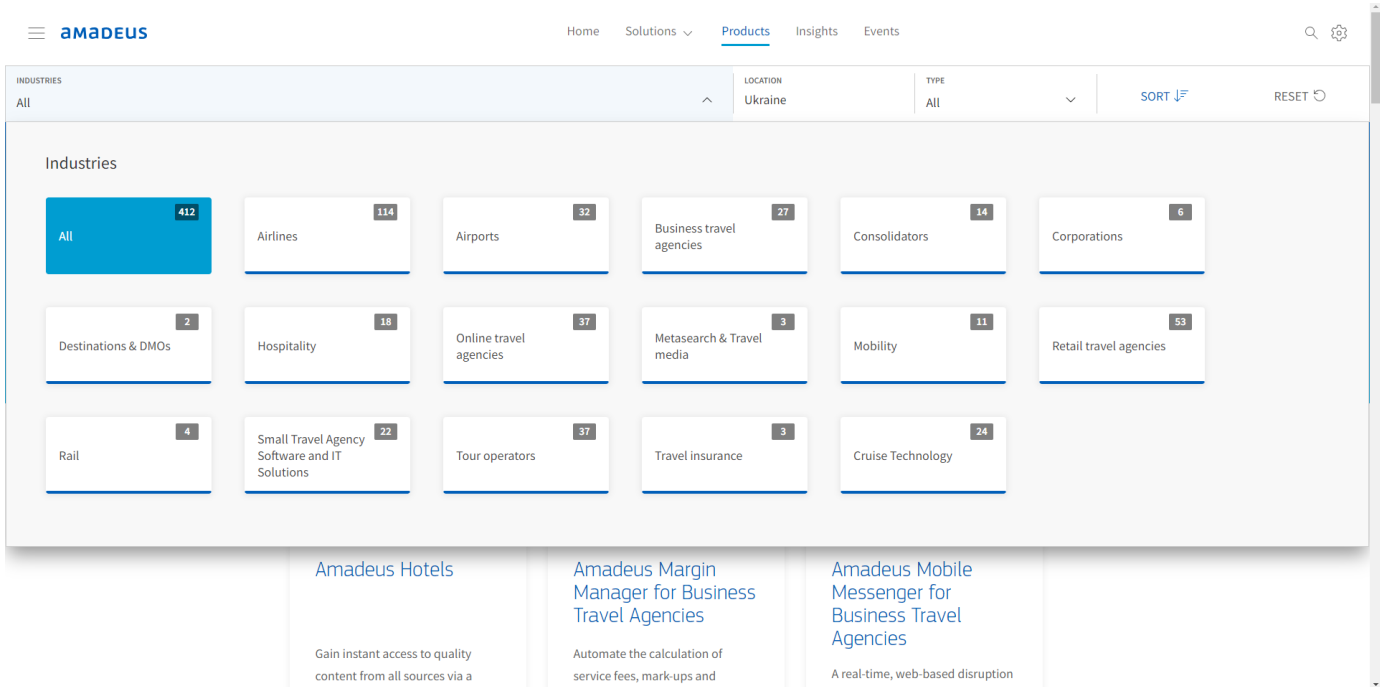


Рисунок 1.1 - Веб-застосування “Amadeus”

- Sabre: Sabre - провідна компанія з розробки програмного забезпечення та технологій, яка підтримує світову туристичну індустрію. Завдяки десятиліттям революційних інновацій команда експертів Sabre запроваджує інновації на всю туристичну екосистему. Sabre співпрацює з авіакомпаніями, готельєрами, агентствами та іншими туристичними партнерами, займаючись роздрібною торгівлею, розповсюдженням та виконанням подорожей. [11]

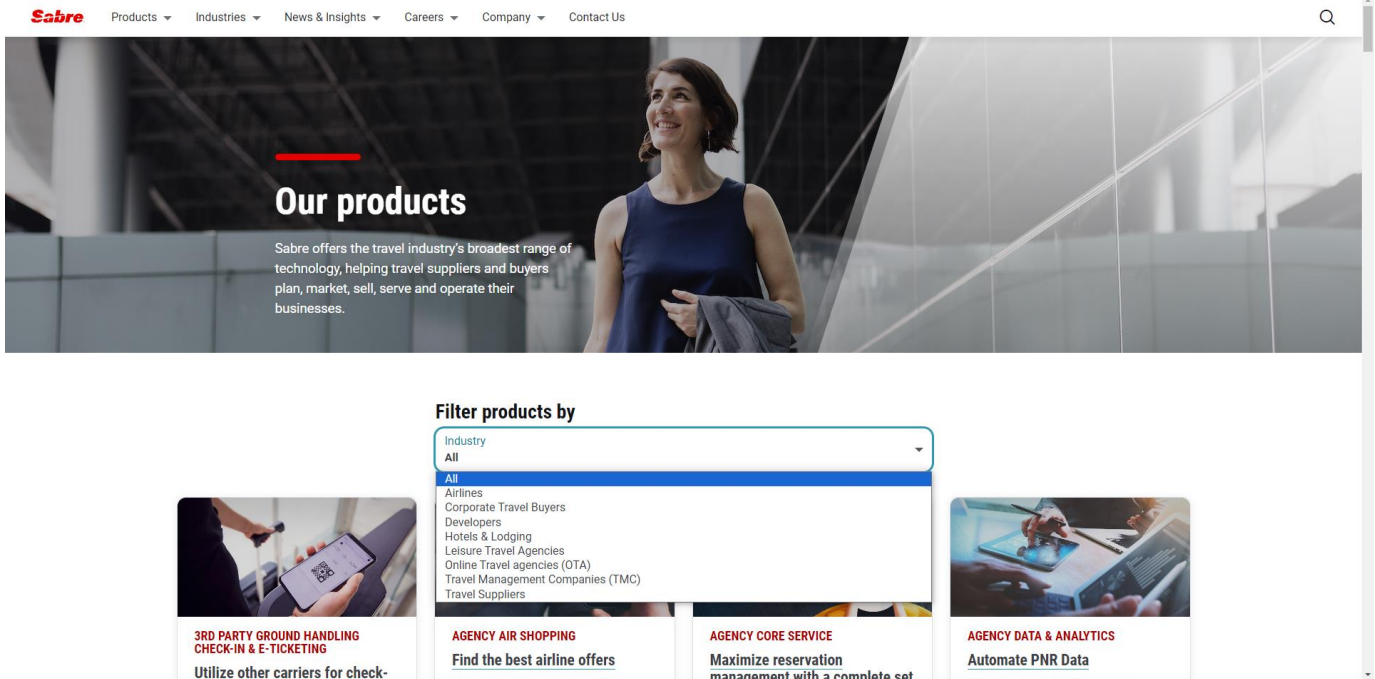


Рисунок 1.2 - Веб-застосування “Sabre”

2. Управління багажем і пасажирським обслуговуванням:

- SITA: SITA надає рішення і послуги для авіакомпаній, аеропортів, операторів наземного обслуговування, які працюють в аеропорту, автоматизації управління багажем, контролю доступу та інших аспектів аеропортового обслуговування. [12]

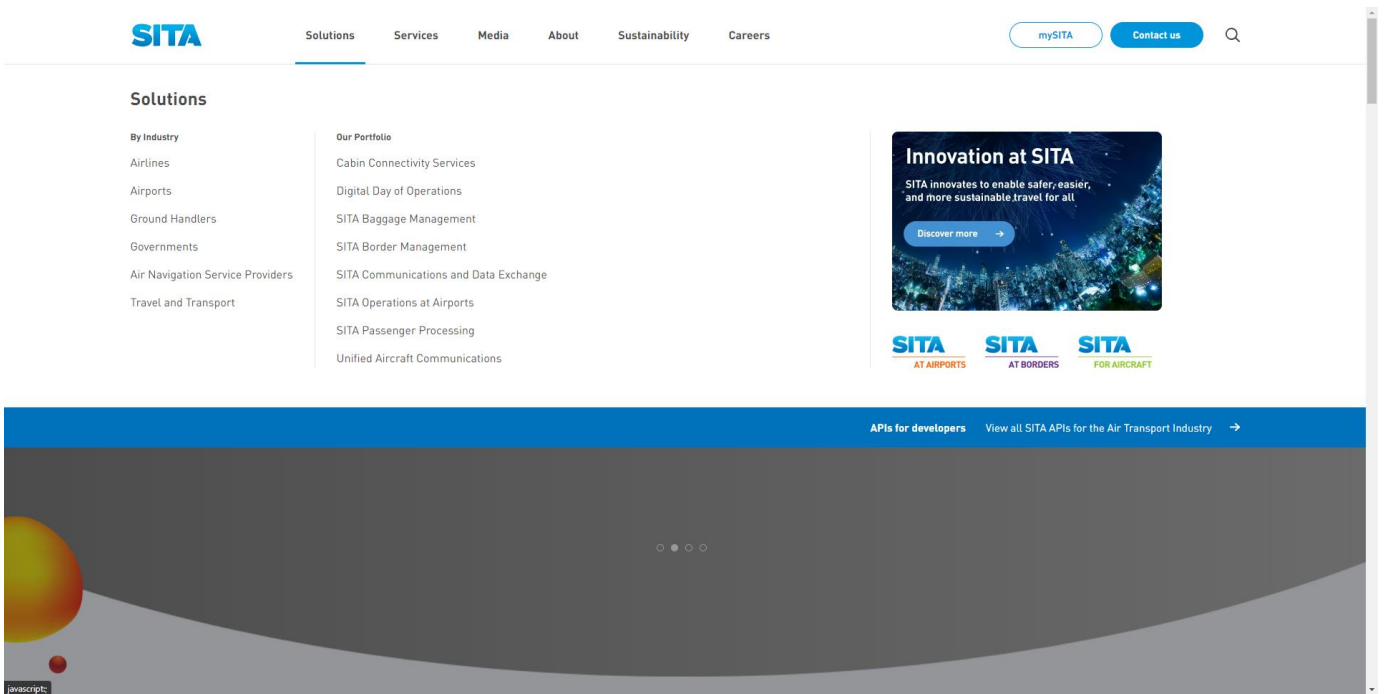


Рисунок 1.3 - Веб-застосування “SITA”



- Passenger Terminal Expo: Ця подія представляє різні технологічні рішення для пасажирського обслуговування, включаючи автоматизовані системи для аеропортів та автовокзалів. [13]

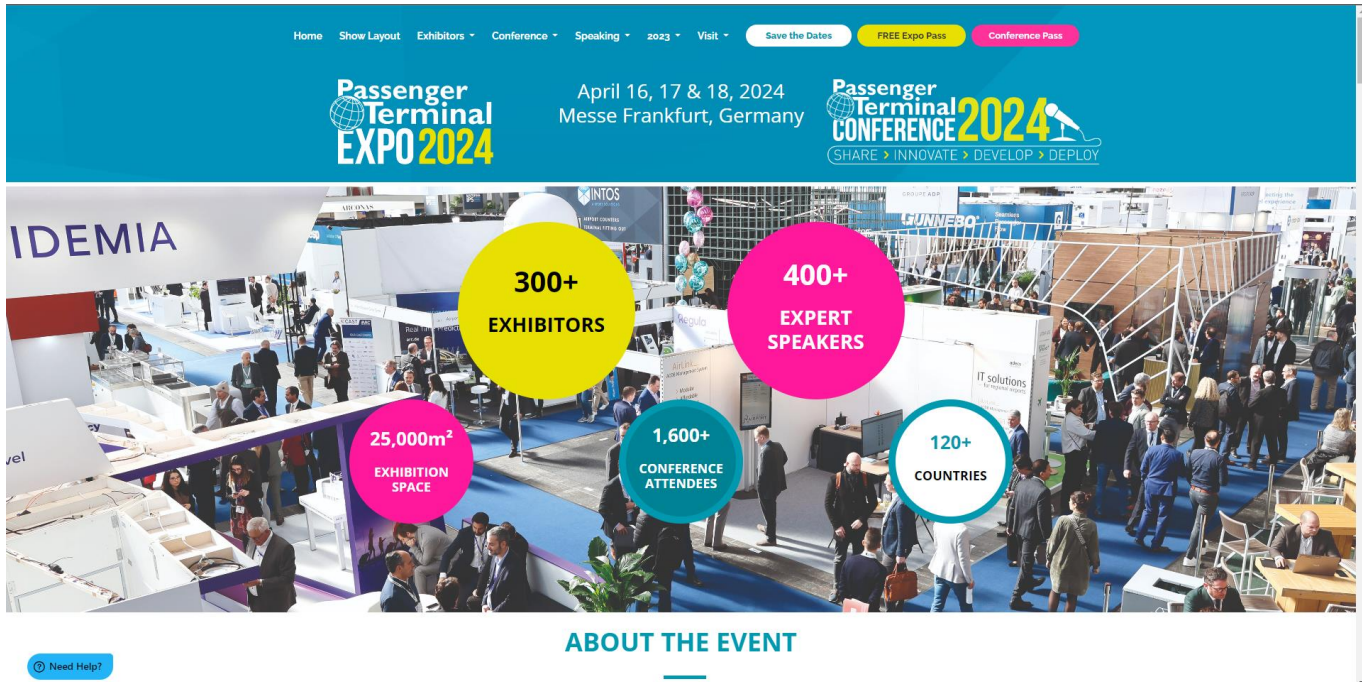


Рисунок 1.4 - Веб-застосування “ Passenger Terminal Expo ”

3. Системи для автоматизації білетного контролю та безпеки:
  - Cognitec: Cognitec спеціалізується на технологіях біометричної ідентифікації, які можуть використовуватися для автоматизованого білетного контролю та безпеки на автовокзалах та аеропортах. [14]

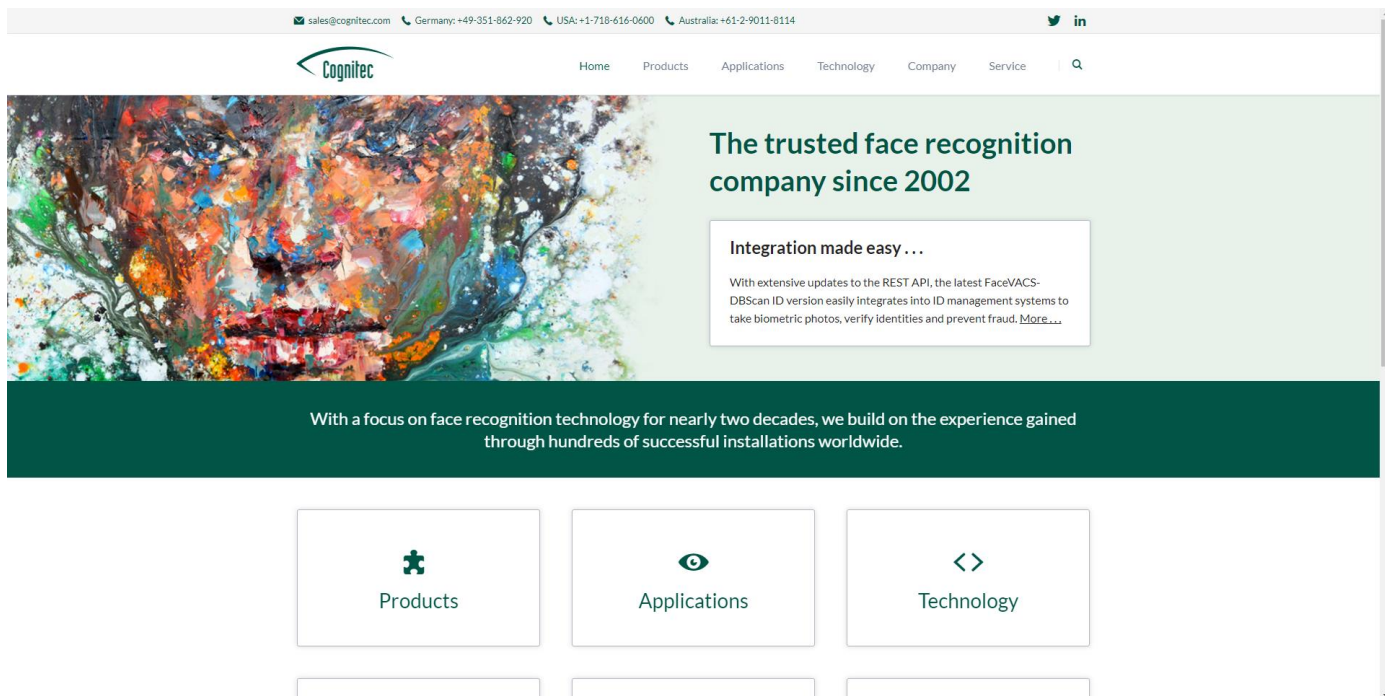


Рисунок 1.5 - Веб-застосування “ Passenger Terminal Expo ”

- **Rapiscan Systems:** Rapiscan Systems постачає рентгенівські системи та інші технології для дослідження багажу та пакунків пасажирів. Як провідний у світі постачальник перевірки безпеки, Rapiscan Systems надає сучасні продукти, рішення та послуги, які відповідають найвибагливішим потребам наших клієнтів у виявленні загроз, одночасно підвищуючи ефективність роботи. [15]

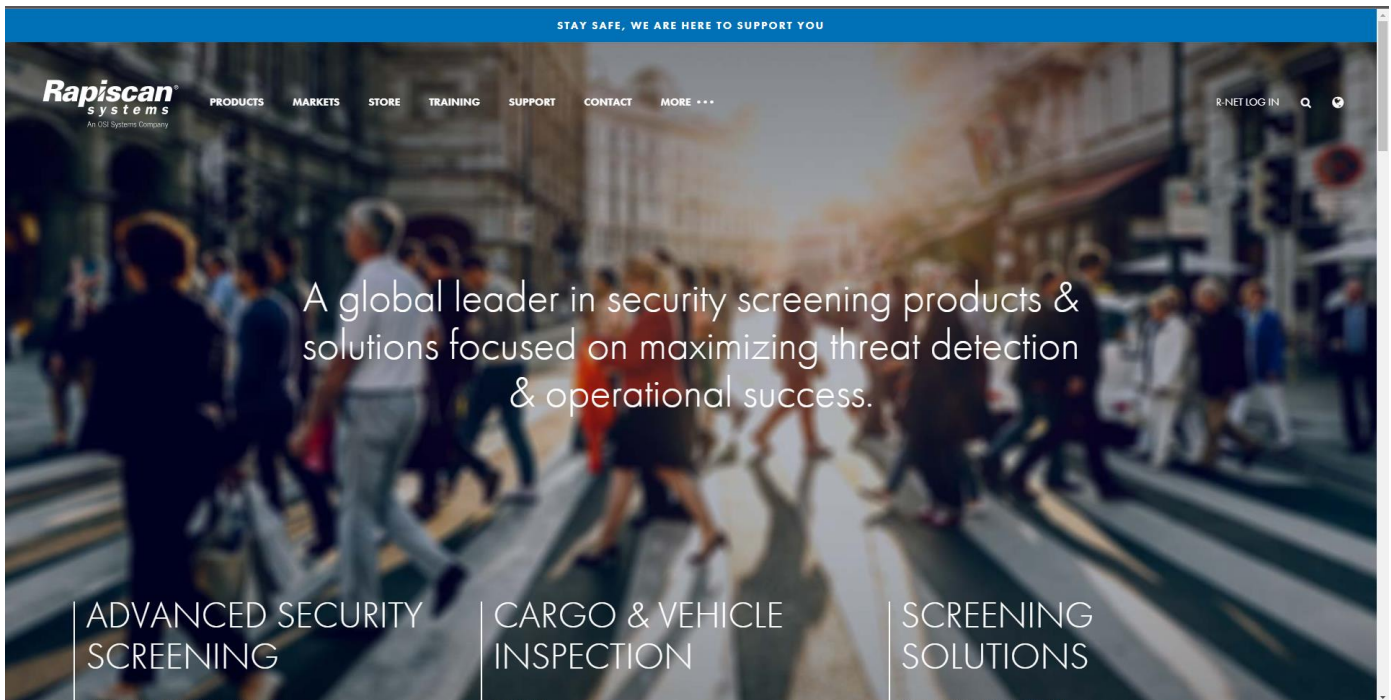


Рисунок 1.6 - Веб-застосування “ Rapiscan Systems”

#### 4. Інтегровані системи для транспортного управління:

- **Trapeze Group:** Trapeze Group розробляє інтегровані системи для управління громадським транспортом, які включають в себе автоматизований розклад та відстеження транспортних засобів. Компанія допомагає постачальникам автобусів, трамваїв, залізниць, таксі і шкільному транспорту керувати всіма аспектами їх діяльності. Рішення Trapeze Group знижують операційні витрати, забезпечуючи найбільш ефективний розподіл доступних ресурсів, дозволяючи транспортним постачальникам робити те, що вони вміють найкраще: забезпечувати доступну, безпечну та надійну мобільність для своїх клієнтів. [16]

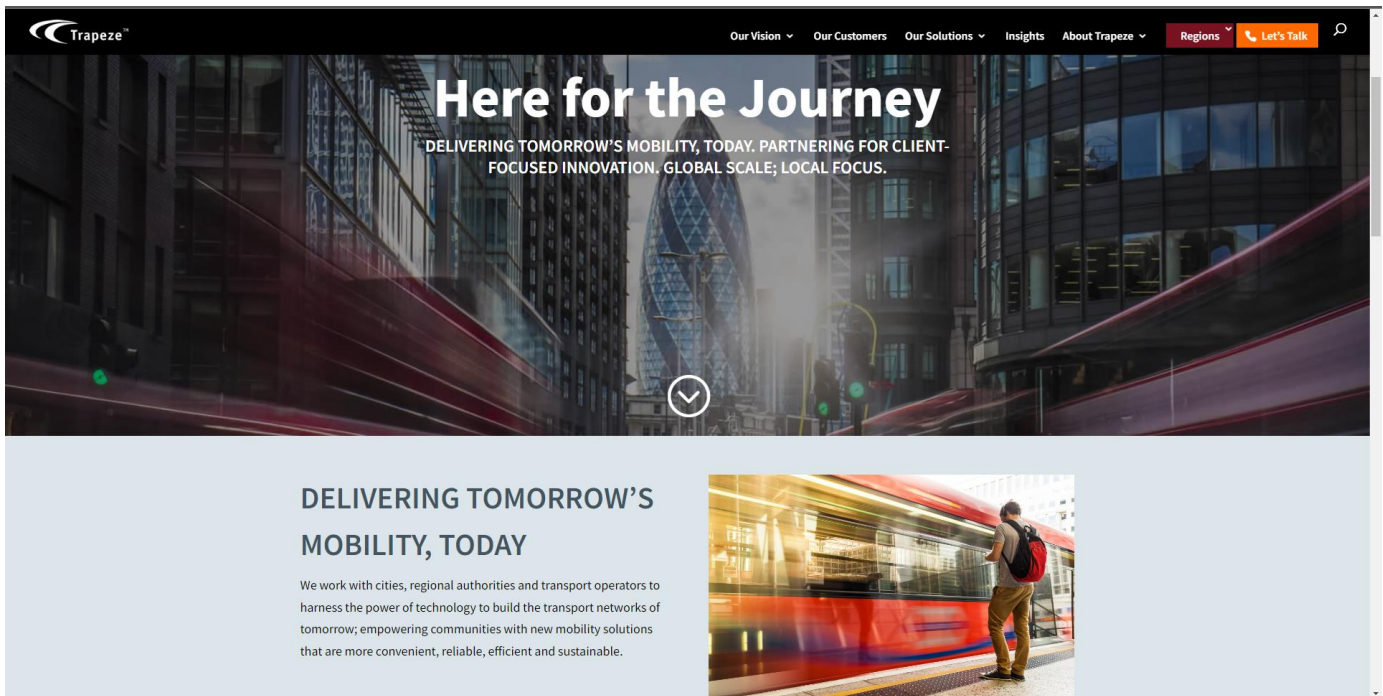


Рисунок 1.7 - Веб-застосування “Trapeze Group”

- **INIT:** INIT спеціалізується на інтегрованих системах для керування громадським транспортом, включаючи білетні системи та системи оплати. Основна конкурентна перевага компанії - INIT є єдиним у світі постачальником, який покриває повний спектр завдань, з якими стикаються компанії громадського транспорту під час надання своїх послуг. Завдяки унікальній інтегрованій системі апаратних і програмних рішень INIT пропонує максимальну допомогу в їх комплексній роботі. [17]

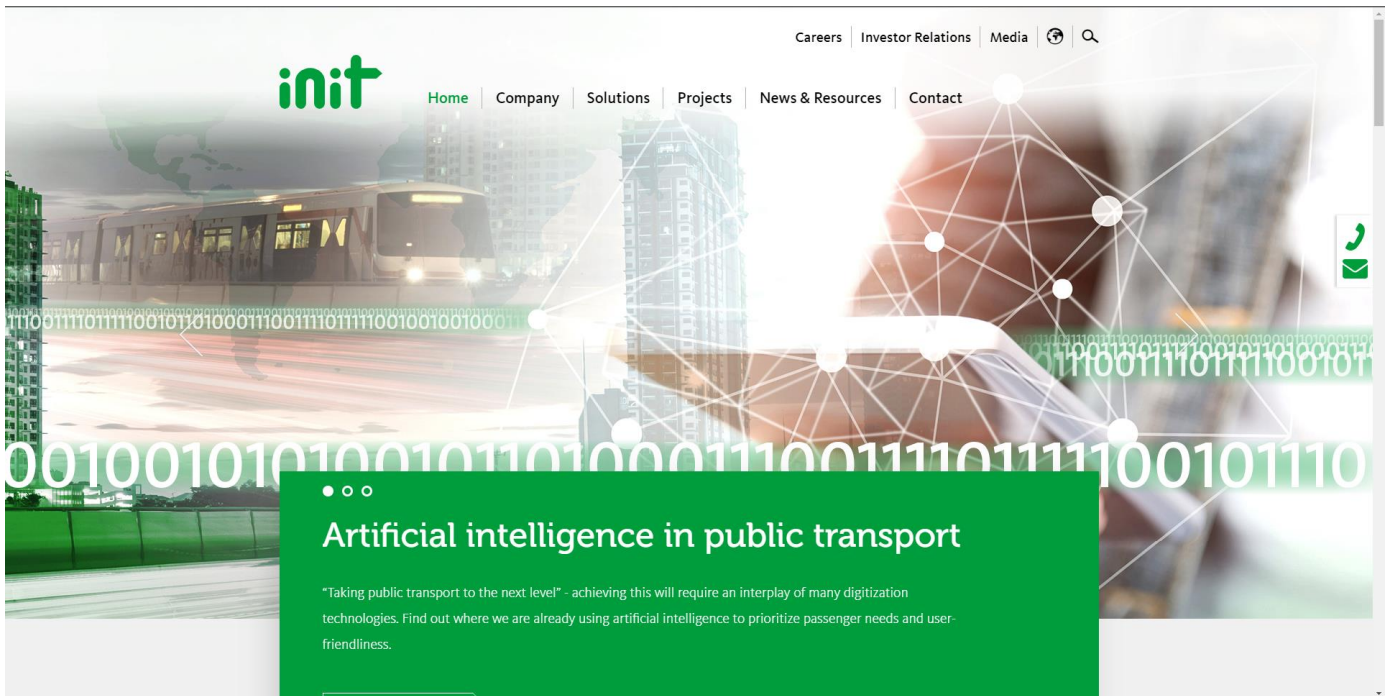


Рисунок 1.8 - Веб-застосування “INIT”

Ці приклади систем для автоматизації та управління транспортними послугами наочно демонструють різноманітність рішень і технологій, що існують на ринку. Однак успішна розробка та впровадження системи для автовокзалу вимагає більш широкого розуміння та урахування низки ключових аспектів:

- **Специфіка вокзалу:** Кожен автовокзал має свою власну специфіку та потреби. Важливо вивчити особливості конкретного вокзалу, такі як розміри, потужність, місце розташування і обсяги пасажиропотоку.
- **Інтеграція з існуючими системами:** Розроблена система повинна інтегруватися з існуючими інфраструктурними та технічними системами вокзалу, такими як квиткові системи, системи безпеки та інші. Це дозволить забезпечити синхронну роботу всіх компонентів.
- **Пасажирський комфорт:** Важливо враховувати потреби та комфорт пасажирів. Зручність інтерфейсу користувача, доступ до інформації та послуг, а також забезпечення безпеки мають високий пріоритет.

- **Складність мережі транспорту:** У великих містах може існувати складна мережа громадського транспорту, включаючи автобуси, метро, трамваї тощо. Система повинна бути здатною координувати та оптимізувати рух транспорту для забезпечення максимальної ефективності.
- **Сучасні технології:** Використання сучасних технологій, таких як штучний інтелект, аналітика даних та IoT, може значно покращити функціональність та ефективність системи.
- **Управління даними:** Важливим аспектом є збір, аналіз та збереження даних про пасажиропотік, рейси та іншу інформацію для подальшого вдосконалення системи.

Урахування цих факторів допомагає забезпечити, що система для автовокзалу відповідає конкретним потребам і стандартам, забезпечуючи ефективне та зручне обслуговування пасажирів та підвищуючи загальний рівень якості транспортних послуг.

### 1.2.2 Вибір і обґрунтування сучасних методів розв'язання науково-технічної задачі

Сучасні методи вирішення завдання автоматизації автовокзалу можуть включати в себе поєднання різних технологій та підходів для досягнення максимальної ефективності та зручності для пасажирів та операторів. Оптимальний вибір методу буде залежати від конкретних потреб і можливостей автовокзалу, наприклад, можливі сучасні методи включають в себе використання систем штучного інтелекту для оптимізації рейсів та обслуговування пасажирів, впровадження безконтактних технологій, таких як NFC для квитків та платежів, використання аналітики даних для планування та прогнозування, а також використання IoT-датчиків для відстеження руху транспорту та стану інфраструктури. Вибір методу повинен базуватися на конкретних вимогах та цілях автовокзалу з метою покращити обслуговування пасажирів, зменшити витрати та покращити екологічну стійкість.

Нижче вказані усі ці сучасні методи та те що вони забезпечують на даний момент:

- Системи Штучного Інтелекту (AI) і Машинного Навчання: Вони використовуються для оптимізації рейсів, розкладів, прогнозування попиту, а також для роботи з клієнтами через віртуальних помічників або чат-ботів. AI допомагає вдосконалити розподіл ресурсів та респондентів на різних етапах обслуговування [18] [19].
- Безконтактні технології, такі як мобільні та web-додатки і NFC: Вони дозволяють пасажиром бронювати та оплачувати квитки, використовуючи свої смартфони або безконтактні картки. Це спрощує процес придбання квитків та прискорює посадку в транспорт [20].
- Аналітика даних: Аналіз даних допомагає в прогнозуванні попиту на рейси, оптимізації роботи автовокзалу, а також в управлінні запасами і ресурсами. Він також допомагає вирішувати проблеми, такі як черги та затримки [21].
- Інтернет речей (IoT): Використання датчиків на транспортних засобах і в інфраструктурі допомагає відстежувати рух транспорту в реальному часі та забезпечує пасажирів актуальною інформацією [22].
- Системи безпеки та розпізнавання обличчя: Вони забезпечують безпеку на автовокзалі та можуть використовуватися для контролю доступу та ідентифікації пасажирів [23].
- Системи автоматизованої обробки багажу: Вони дозволяють швидко та точно обробляти багаж пасажирів і покращити якість обслуговування [24].

Для розв'язання цієї задачі у даній магістерській роботі найбільш перспективним виглядає варіант розробки web-додатку з декількох вагомих причин:

- Створення web-додатку дозволяє реалізувати концепцію автоматизованої системи автовокзалу у вигляді реального програмного продукту. Це надасть конкретні результати та можливість показати практичний внесок роботи.

- Розробка web-додатку є зручною і швидкою завдяки доступності багатьох фреймворків та інструментів для розробки веб-додатків, тестування такого додатку також є досить зручним процесом.
- Створення web-додатку дозволяє продемонструвати навички веб-розробки, баз даних, дизайну і інших аспектів інформаційних технологій.
- Web-додаток доступний для користувачів через браузер на різних пристроях, таких як комп'ютери, планшети та смартфони - це забезпечує зручний доступ до системи для пасажирів, операторів та інших зацікавлених сторін.
- Web-додаток може бути легко розширений та оновлений з часом – до нього можна додавати нові функції та покращувати систему відповідно до потреб та вимог.

### 1.2.3 Варіантний аналіз можливих нових методів розв'язання задачі

Для розв'язання завдання розробки автоматизованої системи для автовокзалу існують різні можливі методи та підходи, які можуть бути новаторськими або вдосконаленими. Нижче наведені варіанти таких методів разом з їх перевагами і недоліками:

#### 1. Використання Штучного Інтелекту (AI):

- Переваги: AI може використовуватися для аналізу даних про пасажиропотік, щоб прогнозувати попит на рейси і оптимізувати розклади. Він може надавати пасажиром індивідуальні рекомендації та інформацію про маршрути, що покращує їхній досвід. Також AI може допомагати виявляти аномалії та покращувати безпеку.

- Недоліки: Розробка та впровадження системи AI можуть бути витратними та складними завданнями. Також потрібно дотримуватися вимог щодо захисту даних та приватності [25].

#### 2. Інтернет речей (IoT):



- Переваги: IoT дозволяє відстежувати розташування та стан транспорту в реальному часі, що допомагає пасажиром точно визначити прибуття автобусів та уникнути затримок. Це також може покращити безпеку та планування обслуговування транспорту.

- Недоліки: Встановлення та обслуговування IoT-систем може вимагати значних інвестицій та технічних навичок. Також потрібно вирішувати питання щодо забезпечення безпеки даних та мережі [26].

### 3. Блокчейн для квитків та платежів:

- Переваги: Блокчейн може забезпечити надійність та безпеку у сфері квитків та платежів. Він робить операції прозорими та недійсними для зловмисників. Крім того, це може спростити облік та відстеження транзакцій.

- Недоліки: Впровадження блокчейну може вимагати змін в існуючих системах та взаємодії з регуляторами. Також може бути важко розв'язати проблеми шкалювання для масового використання [27].

### 4. Мобільні і web-додатки та NFC-технології:

- Переваги: Мобільні додатки надають пасажиром зручний спосіб бронювати квитки, отримувати інформацію про рейси та здійснювати платежі. NFC-технології дозволяють проводити швидкі та безконтактні оплати.

- Недоліки: Потреба в доступі до мобільного Інтернету та сумісних пристроїв може обмежити доступність цих рішень. Також потрібно вирішувати питання безпеки даних користувачів [28].

### 5. Аналітика даних та Big Data:

- Переваги: Аналіз даних дозволяє оптимізувати роботу вокзалу та транспорту, шляхом прогнозування попиту, виявлення тенденцій та виокремлення ефективних рішень. Big Data дозволяє обробляти великі обсяги даних для отримання цінної інформації.

- Недоліки: Збір та аналіз великих обсягів даних може бути складним завданням. Також потрібно забезпечити захист конфіденційної інформації [29].

#### 6. Системи безпеки та розпізнавання обличчя:

- Переваги: Використання систем відеоспостереження та розпізнавання обличчя може значно покращити безпеку на автовокзалі. Це дозволяє вчасно виявляти підозрілу активність та допомагає в розслідуванні подій. Крім того, це може бути використано для контролю доступу та ідентифікації пасажирів.

- Недоліки: Впровадження систем відеоспостереження та розпізнавання обличчя може збільшити витрати на обладнання та підтримку. Також потрібно дотримуватися нормативів щодо захисту приватності та даних [30].

#### 7. Екологічно дружні рішення:

- Переваги: Перехід на автобуси з альтернативними джерелами енергії, такими як електрика або водень, може допомогти зменшити викиди вуглецю та вплив на навколишнє середовище. Це може бути позитивним для репутації автовокзалу та сприяти сталому розвитку.

- Недоліки: Зміна технічних рішень для транспорту може вимагати великих інвестицій і перепланування. Крім того, інфраструктура для зарядки або заправки альтернативних транспортних засобів також потребує інвестицій. Також, автобусні перевезення відбуваються на великі відстані у даній системі автовокзалу і великою проблемою є компактне джерело живлення яке може тримати заряд протягом тривалих поїздок та після цього швидко заряджатися, щоб на наступний день даний вид транспорту знову міг перевозити пасажирів у потрібні їм місця (проблема компактного джерела живлення є великою проблемою у багатьох сферах в сучасному світі) [31].

#### 8. Системи автоматизованої обробки багажу:

- Переваги: Використання роботизованих систем для обробки багажу може покращити швидкість обслуговування пасажирів та зменшити ризик втрати багажу.

Пасажири можуть відстежувати свій багаж в реальному часі, що забезпечує більшу впевненість у безпеці їхніх особистих речей.

- Недоліки: Впровадження роботизованих систем може бути дорогим та потребувати технічного обслуговування. Також може бути необхідно навчати персонал користуватися новою технологією.

Обираючи один або кілька з цих методів, слід враховувати конкретні потреби та можливості проекту. Кожен з цих підходів може мати важливий вплив на функціональність та ефективність автовокзалу та пасажирського обслуговування.

### 1.3 Висновки

У цьому розділі було виконано аналіз проблеми, пов'язаної з розробкою автоматизованої системи для автовокзалу. Було детально розглянуто стан дослідження цієї науково-технічної задачі, спираючись на аналіз існуючих автоматизованих систем у галузі транспорту. Порівняльний аналіз існуючих систем для пошуку квитків на транспорт дав підстави вважати, що створення програмного продукту є цілком доцільним рішенням.

Далі, було розглянуто сучасні методи вирішення цього завдання та обрано оптимальний варіант для подальшого дослідження і розробки. Крім того, був проведений аналіз можливих нових підходів та методів для вирішення цієї задачі, що підсилить виконання цієї роботи та забезпечить її інноваційність.

Загалом, на основі проведеного аналізу, було визначено, що розробка програмного продукту для автоматизації автовокзалу є належним та обґрунтованим напрямком дослідження та розвитку.

## 2 ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ АВТОВОКЗАЛУ

### 2.1 Аналіз функцій системи

Автоматизована система автовокзалу, яка буде розроблена, має включати в себе в себе наступні функції: вхід на сайт, пошук квитків за допомогою фільтру, бронювання та відміна бронювання квитків, застосування модуля який використовує AI. Для візуалізації цих функцій було використано мову моделювання UML.

Мова моделювання UML (Unified Modeling Language) є стандартною мовою для візуального моделювання програмних систем. UML була розроблена для стандартизації та уніфікації процесу моделювання програмних систем і забезпечення чіткого засобу комунікації між розробниками та стейкхолдерами проекту. Основні елементи мови UML включають діаграми, які дозволяють відобразити різні аспекти системи:

- Діаграми класів (Class Diagrams): Ці діаграми використовуються для моделювання класів програми, їх атрибутів та методів. Вони показують структуру системи, включаючи спадкування (наслідування) між класами.
- Діаграми послідовності (Sequence Diagrams): Ці діаграми демонструють послідовність взаємодій між об'єктами системи в часі. Вони особливо корисні для моделювання взаємодій між різними об'єктами під час виконання програми.
- Діаграми активностей (Activity Diagrams): Вони використовуються для моделювання процесів, операцій і алгоритмів у системі. Діаграми активностей дозволяють показати послідовність дій та умовні розвітні пункти.
- Діаграми станів (State Diagrams): Вони дозволяють моделювати стани об'єкта та переходи між ними відповідно до певних подій або умов. Вони корисні для моделювання поведінки системи, яка може знаходитися у різних станах.
- Діаграми співробітництва (Collaboration Diagrams): Вони показують взаємодію об'єктів та відносини між ними. Діаграми співробітництва допомагають

відобразити, як об'єкти співпрацюють один з одним.

- **Діаграми компонентів (Component Diagrams):** Вони моделюють структуру компонентів системи та зв'язки між ними. Це корисний інструмент для проектування розподілених систем.
- **Діаграми розгортання (Deployment Diagrams):** Вони використовуються для моделювання фізичної структури системи, такої як сервери, обладнання та мережі.
- **Діаграми прецедентів (Use Case Diagrams):** Вони допомагають моделювати функціональність системи з точки зору користувача. Вони визначають ролі користувачів та їх взаємодії з системою.

Усі ці діаграми спільно створюють повний огляд програмної системи, дозволяючи розробникам, архітекторам і стейкхолдерам краще розуміти, проектувати та спілкуватися щодо системи. UML надає стандартні символи та правила для створення цих діаграм, що робить їх зрозумілими та стандартизованими для всіх учасників проекту. [32].

У цьому пункті описано існуючі у продукті діаграми варіантів використання, діяльності та розгортання, для більш детального відображення всього процесу взаємодії користувача автовокзалу із системою автовокзалу.

Діаграма варіантів використання (Use Case Diagram) - це графічний інструмент для моделювання функціональних можливостей і взаємодії користувачів (акторів) з системою. Вона дозволяє визначити основні функції системи та сценарії взаємодії між користувачами та системою. Елементи діаграми варіантів використання:

**1. Актори (Actors):** Актори - це зовнішні сутності, які взаємодіють з системою. Вони можуть бути реальними користувачами системи або іншими системами, які взаємодіють з досліджуваною системою. Наприклад, пасажир, оператор вокзалу, адміністратор системи - це приклади акторів.

**2. Варіанти використання (Use Cases):** Варіанти використання - це функціональні можливості системи, які можуть бути виконані акторами. Кожен варіант

використання описує конкретний сценарій взаємодії між акторами та системою. Вони представлені у великих овалах на діаграмі.

**3.** Відносини між акторами та варіантами використання: Відносини показують, які актори беруть участь у виконанні конкретного варіанту використання. Зазвичай використовуються наступні відносини:

- Включає (Includes): Вказує, що один варіант використання може включати інший.
- Розширює (Extends): Вказує, що один варіант використання може розширювати інший. [33]

Мета цієї діаграми полягає у тому, щоб визначити, як користувачі та інші системи (актори) будуть взаємодіяти з проєктованою системою. Ця взаємодія описується через набір сценаріїв, відомих як "варіанти використання." Кожен варіант використання конкретизує, які дії виконує система під час спілкування з акторами.

На рисунку 2.1 наведена діаграма варіантів використання в якій акторами виступають: пасажир, оператор автовокзалу, адміністратор системи та система оплати. За допомогою відношень між цими акторами більш детально описано логіку взаємодії користувача автовокзалу із системою автовокзалу.

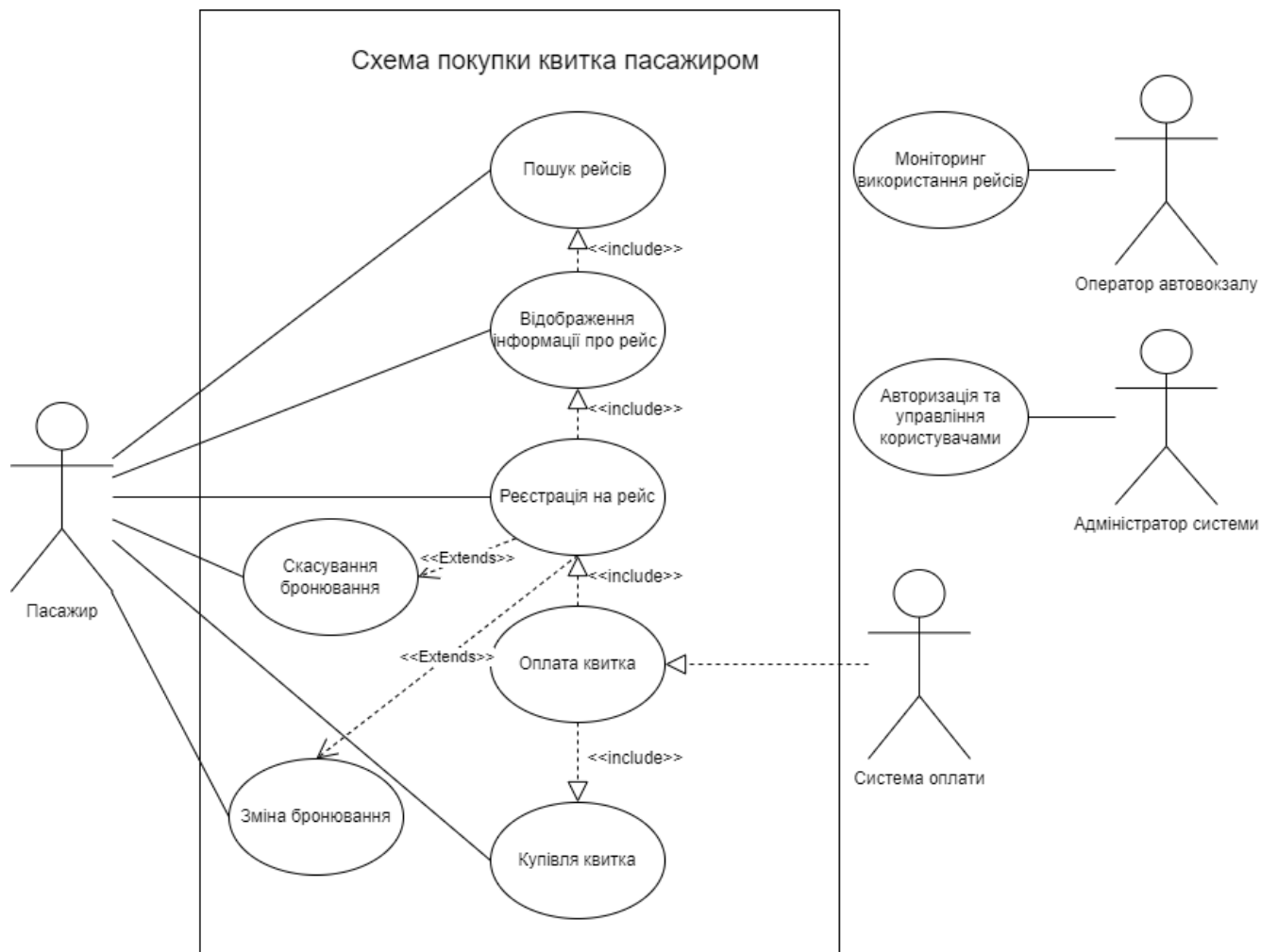


Рисунок 2.1 - UML-діаграма варіантів використання

Опис взаємодії акторів із системою автовокзалу більш детально описано далі: Клієнт (Пасажир) спочатку повинен зареєструватися на сайті, після чого він отримає доступ до функціоналу системи та зможе переглянути інформацію про рейси та доступні квитки, після чого за допомогою фільтру обрати потрібний квиток із сформованого списку та забронювати його, у разі виникнення ситуації коли користувачу більше не знадобиться даний квиток він зможе скасувати бронювання на нього, також у системі буде передбачено можливість оплати квитка, шляхом інтеграції із системою оплати тому в даній схемі вона виступає у вигляді окремого актора. У адміністратора системи є окремий вхід до системи, який дає йому можливість

управляти всією системою, в свою чергу, оператор автовокзалу має можливість моніторити використання рейсів та кількість квитків які залишилися на окремий рейс.

## 2.2 Проєктування архітектури системи

Архітектура системи для автоматизації автовокзалу може бути побудована з урахуванням кількох ключових компонентів та рівнів. Враховуючи обсяг та склад завдань, описаних раніше, розглянемо архітектурний підхід, що включає кілька рівнів бізнес-логіки.

### 1. Клієнтський рівень (Веб-інтерфейс):

- Веб-інтерфейс: Веб-додаток для користувачів, який дозволяє пасажиром бронювати квитки, переглядати розклади рейсів, робити онлайн-платежі та отримувати інформацію про маршрути та послуги. Він є найбільш доступним і зручним способом для пасажирів здійснювати бронювання квитків та отримувати інформацію про рейси.

### 2. Серверний рівень (Серверні застосунки):

- Серверні застосунки: Реалізація бізнес-логіки на сервері для обробки запитів від клієнтського додатку. Це може бути створене з використанням мови програмування C# та фреймворка ASP.NET. Серверні застосунки виконують бізнес-логіку, обробку запитів від клієнтського додатку та взаємодію з базою даних. Вони забезпечують обробку бронювання, планування рейсів тощо.

### 3. Бізнес-логіка:

Рівень бізнес-логіки розділяє функціональність системи на логічні модулі, що полегшує розробку, тестування і супровід системи. Він включає в себе наступні модулі:

- Керування рейсами та маршрутами: Модуль для планування, керування та моніторингу рейсів та маршрутів.
- Бронювання квитків: Обробка запитів щодо бронювання квитків.



- Керування пасажирями: Модуль для обліку та обслуговування пасажирів.

#### 4. Система управління базою даних (СУБД):

- Сховище даних: База даних є центральним сховищем інформації про рейси, квитки, пасажирів та інші дані. Можливе використання СУБД, такої як Microsoft SQL Server, дозволить забезпечити безпеку, цілісність та продуктивність бази даних.

#### 5. Можлива інтеграція з іншими системами (Платіжні системи, Моніторинг транспорту):

- Платіжні системи: Інтеграція з платіжними системами для оплати квитків дозволить обробляти онлайн-платежі в системі.
- Моніторинг транспорту: Моніторинг транспорту забезпечить отримання даних щодо руху транспорту для відстеження рейсів.

#### 6. Забезпечення доступу до системи:

- Авторизація: Авторизація дозволяє контролювати доступ користувачів до різних функцій системи.

#### 7. Інфраструктура:

- Хмарна архітектура: У подальшому як варіант можна розглянути хмарний хостинг для веб-додатку що забезпечить масштабованість та доступність.

Такий рівневий підхід дозволить розділити функціональність системи на логічні компоненти, полегшуючи розробку, супровід та масштабування. Архітектура системи описаної вище зображена на рисунку 2.3.

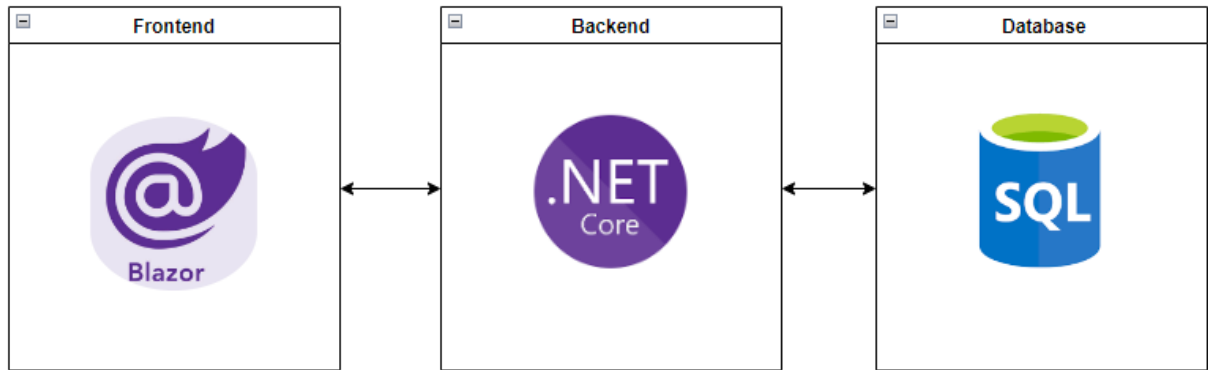


Рисунок 2.3 – Архітектура системи

У графічній формі цю архітектуру описує діаграма розгортання на рисунку 2.4.

Діаграма розгортання (Deployment Diagram) в UML - це графічний інструмент, який використовується для моделювання фізичного розташування компонентів системи та їх взаємозв'язків на обладнанні (апаратурі) або в середовищі виконання (наприклад, серверах або віртуальних машинах). Діаграма розгортання дозволяє візуалізувати архітектурну конфігурацію системи та її фізичну структуру. Основні елементи діаграми розгортання включають в себе:

- **Вузли (Nodes):** Вони представляють фізичні або логічні вузли, де розташовані компоненти системи. Вузли можуть бути обчислювальними серверами, мережевими пристроями, апаратурою, віртуальними машинами тощо.
- **Артефакти (Artifacts):** Артефакти представляють файли, бібліотеки, програми тощо, які зберігаються внутрішньо в вузлах.
- **Зв'язки (Connections):** Зв'язки між вузлами показують, як компоненти системи спілкуються один з одним. Може використовуватися різний тип зв'язків, такий як асоціації, зв'язки залежності, агрегації тощо.
- **Специфікації (Specifications):** Для кожного артефакту можна вказати його специфікацію, тобто деталізований опис, що міститься в артефакті.

- Пакети (Packages): Групи компонентів або артефактів, які можуть бути об'єднані в пакети для кращої структуризації діаграми.

Діаграма розгортання допомагає розуміти, як фізичні компоненти системи розташовані та взаємодіють один з одним, що є важливим при проектуванні та реалізації системи [35].

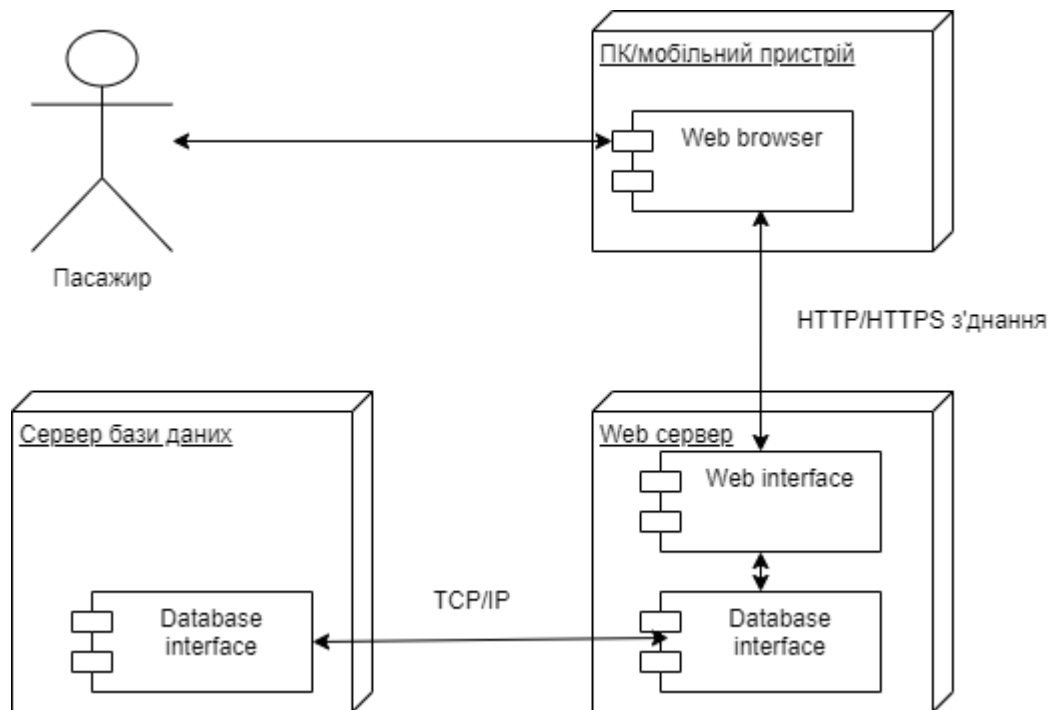


Рисунок 2.4 - UML-діаграма розгортання

Діаграма розгортання показує обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах.

Діаграма діяльності (Activity Diagram) - це графічний інструмент для моделювання послідовності дій і процесів в системі або бізнес-процесі. Вона дозволяє візуалізувати, як окремі дії взаємодіють одна з одною та керують потоком даних. Основні елементи діаграми діяльності включають дії, функціональні блоки, рішення, паралельність та інші компоненти, які допомагають моделювати процеси та послідовності в системі. Діаграми діяльності використовуються для моделювання бізнес-процесів, проектування

програмного забезпечення, аналізу та вдосконалення процесів, а також для документування та спілкування між учасниками проекту. Елементи діаграми діяльності:

- Дія (Action): Дії представлені у вигляді прямокутників і вказують на конкретні дії або обчислення, які відбуваються в системі. Наприклад, "Запит до бази даних" або "Вивід на екран".
- Функціональні блоки (Control Flows): Функціональні блоки - це стрілки, які показують потік виконання між діями. Вони з'єднують дії в послідовний порядок та вказують напрямок потоку виконання.
- Рішення (Decision): Рішення, або відсічки, використовуються для представлення умовних рішень в процесі. Вони визначають, які шляхи будуть обрані в залежності від певних умов.
- Паралельність (Fork/Join): Паралельність використовується для показу паралельних процесів, які відбуваються одночасно. Вона вказує, коли процеси розгалужуються (Fork) і коли об'єднуються (Join).
- Початок і кінець (Start/End): Позначення "Початок" і "Кінець" вказують на початок і завершення виконання діаграми. Вони є обов'язковими елементами [34].

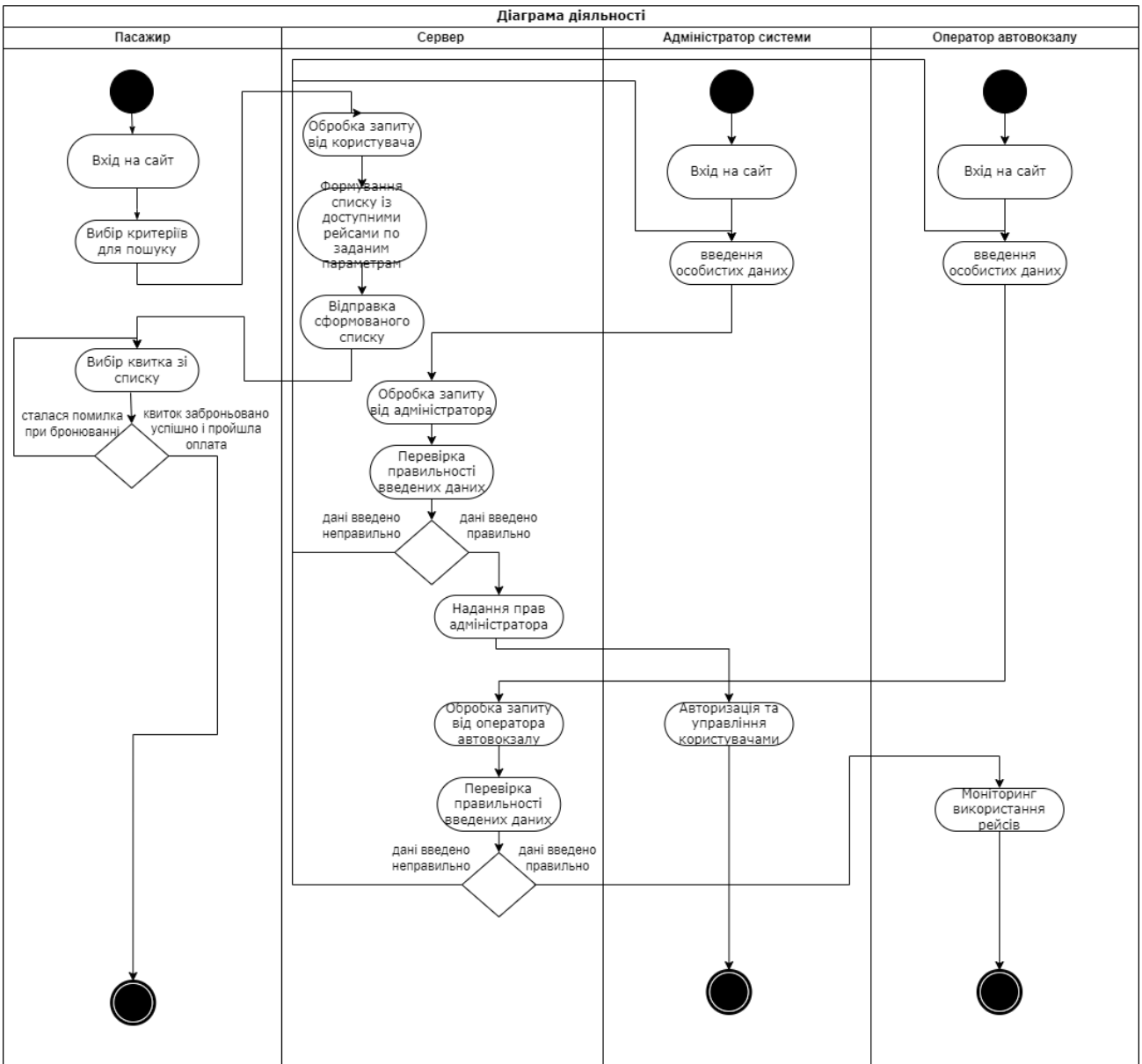


Рисунок 2.2 - UML-діаграма діяльності

Діаграмі діяльності відображена на рисунку 2.3 пояснює роботу процесу пошуку пасажиром потрібного квитка на транспорт. Вона описує взаємодію між користувачем, адміністратором системи, оператором автовокзалу та сервером за допомогою елементів діаграми діяльності.

## 2.3 Аналіз засобів створення програмного забезпечення для розробки автоматизованої системи автовокзалу

У світі спостерігається зростаюча тенденція створення передових автоматизованих інформаційних систем. Ці системи не лише розширюють можливості користувачів і поліпшують їхню взаємодію, але також спрощують процес розробки. На сучасному ринку серверних технологій виділяються дві основні платформи: PHP і ASP.NET. Кожна з цих платформ має свої унікальні особливості і широко використовується в різних проектах завдяки своїм перевагам. У зв'язку з цим для проведення аналізу були обрані PHP5 і ASP.NET, які є одними з найпопулярніших платформ на сьогоднішній день.

PHP (Hypertext Preprocessor) - це високорівнева серверна мова програмування, спеціально призначена для розробки веб-додатків і динамічних веб-сайтів. Основні характеристики PHP включають:

- PHP є інтерпретованою мовою, що означає, що код виконується на сервері і результат відправляється на клієнтський браузер у вигляді HTML, що робить його ідеальним для розробки веб-застосунків.
- PHP підтримує багато систем управління базами даних, таких як MySQL, PostgreSQL, SQLite і багато інших, що дозволяє легко працювати з даними і базами даних.
- Мова має велику і активну спільноту розробників, що призводить до швидкого розвитку та постійного оновлення мови та фреймворків, які її використовують.
- PHP може використовуватися для створення різноманітних веб-застосунків, від простих статичних сторінок до складних корпоративних веб-додатків.
- Ця мова є безкоштовною та має відкритий вихідний код, що дозволяє розробникам використовувати її безкоштовно та модифікувати за потребою.

Загалом, PHP залишається однією з найпопулярніших мов програмування для розробки веб-додатків завдяки своїм простоті, потужності та активній спільноті користувачів [36].

Основними недоліками мови PHP є:

- В мові PHP може бути реалізовано багато різних структур синтаксису, що може призвести до незрозумілого і неоднакового коду між різними розробниками.
- Необережне використання PHP може призвести до вразливостей в безпеці, таких як SQL-ін'єкції і переповнення буфера, які можуть бути використані для атак.
- Хоча PHP досить швидкий, в порівнянні з іншими мовами, він може бути менш ефективним для важких завдань обчислення і обробки даних.
- При роботі над дуже великими проектами PHP може виявитися менш практичним, оскільки він не має вбудованих засобів для організації великих кодових баз і управління залежностями.
- PHP є слабкотипізованою мовою, що може призвести до помилок через неочікувану поведінку змінних.
- Існує багато версій PHP, і деякі функції можуть бути застарілими або видаленими у нових версіях, що може створювати проблеми з сумісністю.
- PHP не завжди ефективно керує пам'яттю, що може призводити до витоків пам'яті та зменшення продуктивності додатка.
- PHP не має вбудованої підтримки багатопотоковості, що робить його менш практичним для додатків, які вимагають одночасної обробки багатьох запитів. [37]

ASP.NET - це високорівнева технологія розробки веб-додатків, створена компанією Microsoft. Вона базується на платформі .NET і надає розробникам потужний інструментарій для створення сучасних веб-застосунків. Основні характеристики технології ASP.NET включають:

- ASP.NET підтримує використання різних мов програмування, таких як C#, VB.NET, F#, і багато інших. Це дає розробникам можливість вибрати мову, з якою вони найкраще орієнтуються.
- ASP.NET надає ефективні інструменти для створення веб-додатків, які можуть масштабуватися від невеликих сайтів до великих корпоративних систем.
- ASP.NET надає різноманітні функції та бібліотеки для роботи з веб-додатками, включаючи управління сеансами, безпеку, автентифікацію та авторизацію, кешування, роботу з базами даних і багато інших.
- За допомогою ASP.NET можна легко розробляти веб-додатки, які працюють на віддалених серверах і можуть бути доступні через Інтернет.
- ASP.NET підтримує відкриті стандарти, такі як HTTP, XML, і SOAP, що робить його сумісним з іншими технологіями та платформами.
- Розробники можуть використовувати інтегроване середовище розробки Visual Studio для створення, налагодження і публікації веб-додатків на платформі ASP.NET.
- ASP.NET підтримує модель розробки MVC (Model-View-Controller), що сприяє розділенню логіки, представлення і обробки запитів.
- Технологія має вбудовані засоби для забезпечення безпеки веб-додатків, включаючи захист від атак і фільтрацію даних [38].

Незважаючи на багато переваг, технологія ASP.NET має деякі недоліки:

- Для використання повних функцій ASP.NET, можливо, доведеться придбати ліцензію на деяке програмне забезпечення Microsoft. Це може бути дорого для деяких проектів.
- ASP.NET розроблено Microsoft і більш сумісне з веб-серверами на базі Windows. Це може обмежити вибір хостинг-постачальників та операційних систем.



- Зазвичай код ASP.NET дещо більший, ніж код, написаний на інших мовах програмування, що може зробити додаток більшим і складнішим для обслуговування.
- Хоча ASP.NET може пропонувати багато інструментів і бібліотек для розробників, розробка додатків на цій платформі може займати більше часу порівняно з іншими швидкими мовами.

Не зважаючи на ці недоліки, важливо враховувати, що ASP.NET є потужною технологією з широкими можливостями, і багато недоліків можна подолати за допомогою правильного досвіду та оптимізації. [39].

Отже, проаналізувавши основні переваги та недоліки обох варіантів для розробки автоматизованої системи автовокзалу, було обрано технологію ASP.NET через її більш високу надійність, масштабованість, продуктивність та можливості які вона надає розробнику.

## 2.4 Вибір засобів розробки автоматизованої системи автовокзалу

### 2.4.1 Вибір мови програмування

Для створення автоматизованої системи автовокзалу було обрано мову програмування C# —сучасну і широко використовувану мову програмування, розробленою компанією Microsoft. C# є об'єктно-орієнтованою мовою програмування, що дозволяє розробляти програми, організовані навколо об'єктів та класів, що сприяє модульності та зручності розробки. Мова сильно типізована, що означає, що кожна змінна та об'єкт мають чітко визначені типи даних, що полегшує виявлення помилок на етапі компіляції.

Впровадження .NET Core та подальший розвиток .NET 5 та .NET 6 розширили підтримку C#, що дозволяє створювати програми для різних операційних систем, таких

як Windows, Linux та macOS. C# має багато можливостей розробки, такі як асинхронне програмування, лямбда-вираження, динамічні типи та інші, що полегшує розробку.

Інтеграція з продуктами Microsoft робить C# чудовим вибором для створення програм для платформ Microsoft, а багата бібліотека класів та фреймворків .NET дозволяє розробникам працювати з різними аспектами розробки, включаючи роботу з базами даних, мережевими програмами, графікою та багатьма іншими.

C# широко використовується в різних галузях розробки програмного забезпечення, від настільних програм до веб-додатків та мобільних додатків. Ця мова програмування користується популярністю завдяки своїм можливостям, зручному середовищу розробки та активній спільноті розробників.

C# був розроблений на основі C++, але він також має елементи, які подібні до функцій мови Java [40]. Таким чином, C# включає три загальні концепції: інкапсуляцію, поліморфізм і успадкування. Більш детально про кожен з цих концепцій:

- Інкапсуляція є одним з основних принципів об'єктно-орієнтованого програмування (ООП) і відіграє важливу роль у створенні структурованих та ефективних програм. Вона полягає в тому, щоб об'єднати дані (змінні) та методи (функції), які працюють з цими даними, в один об'єкт або клас. Основна ідея полягає у тому, щоб приховати внутрішню реалізацію об'єкта від інших частин програми і надати доступ до об'єкта лише через визначені інтерфейси (публічні методи). Інкапсуляція сприяє модульності програми, полегшує її розробку та підтримку, а також зберігає конфіденційність та цілісність даних. Вона є одним із засобів досягнення принципу "розділення відповідальностей" в ООП, що сприяє створенню більш чистого та організованого програмного коду [41].

- Успадкування (наслідування) - це один з основних принципів об'єктно-орієнтованого програмування (ООП), який дозволяє створювати нові класи на основі вже існуючих класів, спадкуючи їх властивості та методи. Успадкування дозволяє створювати ієрархії класів, де класи-потомки спадкують та розширюють функціональність батьківського класу. Цей принцип допомагає впорядковувати код,

сприяє повторному використанню коду та полегшує розробку програм з складною структурою [42].

- Поліморфізм - це один з основних принципів об'єктно-орієнтованого програмування (ООП), який дозволяє об'єктам різних класів виконувати однакові дії, тобто мати спільний інтерфейс, але різну реалізацію цих дій. Поліморфізм робить програми більш гнучкими, зрозумілими та легкими для розширення. Він дозволяє відокремити інтерфейс від реалізації, що сприяє підтримці різних класів та об'єктів у програмі без зміни основного коду [43].

Так як мова програмування C# використовує принципи об'єктно-орієнтованого програмування, має високий рівень абстракції і може легко поєднуватися з технологією ASP.NET, обраною в попередньому пункті - це робить мову C# найбільш оптимальним вибором для виконання поставлених завдань.

#### 2.4.2 Вибір системи управління базами даних (СУБД)

Microsoft SQL Server - це високопродуктивна система управління базами даних (СУБД), розроблена корпорацією Microsoft. Вона призначена для зберігання, керування, опрацювання та забезпечення доступності структурованих даних в різних застосунках і середовищах, від корпоративних систем до веб-додатків. SQL Server володіє великими можливостями щодо обробки даних, безпеки, резервного копіювання і відновлення даних, а також розвитку різноманітних додатків, що забезпечує його популярність серед організацій та розробників програмного забезпечення. Microsoft SQL Server є важливим інструментом для організацій і розробників, які працюють з базами даних і потребують надійного та продуктивного засобу для зберігання та обробки даних [44].

Основні переваги Microsoft SQL Server:

- Надійність: SQL Server відома своєю високою стабільністю і надійністю. Вона добре справляється з великими обсягами даних і має механізми для відновлення в разі аварій.

- **Висока продуктивність:** SQL Server може ефективно опрацьовувати складні запити та обробляти велику кількість транзакцій за короткий час.
- **Широкі можливості:** SQL Server надає розширені можливості для аналізу даних, створення звітів і роботи з даними. Вона підтримує роботу з текстовими, числовими, графічними і аудіо-візуальними даними.
- **Зручний інтерфейс:** Microsoft надає інтерфейси для роботи з SQL Server, включаючи SQL Server Management Studio (SSMS), що робить адміністрування та розробку додатків більш зручними.
- **Інтеграція з іншими продуктами Microsoft:** SQL Server легко інтегрується з іншими продуктами Microsoft, такими як Microsoft Azure, SharePoint, Excel і іншими.
- **Висока безпека:** SQL Server надає різні механізми безпеки, включаючи аутентифікацію, авторизацію, шифрування даних і моніторинг доступу.
- **Спільнота та підтримка:** Існує активна спільнота користувачів і розробників SQL Server, а також офіційна підтримка від Microsoft.
- **Складові для аналізу даних:** SQL Server має вбудовані складові для створення бізнес-аналітики і звітів, такі як SQL Server Analysis Services (SSAS) і SQL Server Reporting Services (SSRS).
- **Резервне копіювання і відновлення:** SQL Server надає засоби для резервного копіювання і відновлення даних, що допомагає забезпечити їхню безпеку.
- **Клауд-інтеграція:** SQL Server може легко інтегруватися з хмарними сервісами Microsoft Azure, що спрощує роботу з хмарними даними та обчисленнями [45].

Незважаючи на багато переваг, Microsoft SQL Server також має свої недоліки і обмеження:

- **Високі витрати:** Ліцензійна вартість Microsoft SQL Server може бути високою, особливо для великих підприємств. Це може бути обмежувальним фактором для менших компаній або стартапів.

- Платформозалежність: SQL Server є платформозалежним продуктом, тобто він в основному призначений для операційних систем Windows. Це може створювати обмеження для розробки крос-платформових додатків.
- Обмеження версій: Різні версії SQL Server мають різні обмеження і функціональність. Деякі більш розширені можливості доступні тільки в дорогих версіях, що може обмежувати вибір для певних застосувань.
- Вимоги до апаратного забезпечення: Для оптимальної продуктивності і обробки великих обсягів даних SQL Server може вимагати потужний апаратний обладнання, що може збільшувати загальні витрати.
- Обмежена безкоштовна версія: Незважаючи на наявність безкоштовної версії SQL Server (Express Edition), вона має обмеження на розмір бази даних і ресурси сервера, що може бути недостатньо для деяких проектів [46].

Врахувавши всі переваги недоліки при розробці автоматизованої системи автовокзалу було обрано СУБД MS SQL Server.

### 2.4.3 Вибір фреймворків

Під час створення автоматизованої системи автовокзалу, крім вибору СУБД також було проведено вибір фреймворків Entity Framework Core та ASP.NET MVC.

Entity Framework Core (EF Core) - це легковагова та швидка технологія об'єктно-реляційного відображення (ORM), розроблена компанією Microsoft. ORM є підходом до роботи з базами даних, який дозволяє розробникам працювати з даними в базі, використовуючи об'єкти та класи, замість написання складних SQL-запитів. Entity Framework Core - це версія Entity Framework, оптимізована для роботи на різних платформах, включаючи .NET Core і .NET 5 і вище [47]. Основні характеристики Entity Framework Core включають:

- Підтримка багатьох провайдерів баз даних: EF Core підтримує різні системи керування базами даних, такі як Microsoft SQL Server, MySQL, PostgreSQL, SQLite, інші. Це дає розробникам можливість використовувати її для різних проектів і баз даних.
- LINQ (Language Integrated Query): Entity Framework Core дозволяє використовувати LINQ для написання запитів до бази даних. Це спрощує взаємодію з даними і робить код більш зрозумілим і читабельним.
- Code-First підхід: Розробники можуть створювати структуру бази даних, визначаючи класи і відносини між ними в коді. EF Core автоматично створює або оновлює схему бази даних відповідно до цього опису.
- Підтримка міграцій: Entity Framework Core надає механізм міграцій, який дозволяє зберігати структуру бази даних під час розвитку додатка. Зміни в структурі бази даних можна легко вносити та зберігати як міграції.
- Вбудована кешування та відстеження змін: EF Core підтримує вбудовані механізми кешування даних та відстеження змін, що дозволяє оптимізувати роботу з базою даних і запитамі.
- Підтримка асинхронного програмування: Реалізація асинхронних запитів дозволяє підвищити продуктивність додатка і уникнути блокування потоків.
- Відкрите джерело: EF Core є проектом з відкритим вихідним кодом, що дозволяє розробникам вносити зміни та вдосконалювати його відповідно до власних потреб [48].

Entity Framework Core став популярним інструментом для роботи з базами даних у світі розробки програмного забезпечення, оскільки він спрощує багато задач, пов'язаних з доступом до даних, і пропонує багато зручних можливостей для розробників.

В свою чергу, ASP.NET MVC (Model-View-Controller) - це фреймворк для розробки веб-додатків, розроблений компанією Microsoft, який базується на патерні

проектування Model-View-Controller, який розділяє додаток на три основні компоненти: модель (Model), представлення (View) і контролер (Controller). Цей розділ дозволяє розробникам розділити логіку додатку на легко керовані і підтримувані частини, що полегшує розробку та підтримку веб-додатків [49].

Основні характеристики ASP.NET MVC включають:

- **Модель (Model):** Модель відповідає за обробку та зберігання даних. Вона взаємодіє з базою даних або іншими джерелами даних та надає контролерам необхідну інформацію для обробки запитів.
- **Представлення (View):** Представлення відповідає за відображення даних для користувача. Воно генерує HTML-код, який відправляється користувачеві через браузер. Використовуючи двигун перегляду (наприклад, Razor), розробники можуть створювати динамічний інтерфейс користувача.
- **Контролер (Controller):** Контролер приймає запити від користувачів і визначає, яка дія повинна бути виконана. Він обробляє запити, взаємодіє з моделлю для отримання необхідних даних та вибирає відповідне представлення для відображення результату.
- **Маршрутизація (Routing):** ASP.NET MVC використовує систему маршрутизації для визначення того, які URL-адреси пов'язуються з контролерами та діями. Це дозволяє створювати зрозумілі та красиві URL-адреси.
- **Підтримка тестування:** Фреймворк сприяє тестуванню коду, оскільки компоненти (контролери, моделі і представлення) можуть бути легко визначені і протестовані окремо.
- **Розширюваність:** ASP.NET MVC дозволяє розробникам розширювати функціональність за допомогою спеціальних бібліотек і плагінів.
- **Підтримка Аях і клієнтської сторони:** Фреймворк забезпечує можливість реалізації Аях-запитів та роботи з клієнтською стороною [50].

## 2.5 Висновки

У даному розділі були створені докладні UML-діаграми, включаючи діаграми варіантів використання, діяльності та розгортання, які надають більш детальний опис системи. Дані діаграми детально висвітлили різні аспекти функціонування системи автоматизації автовокзалу, включаючи процеси взаємодії з користувачами та компонентами системи.

Крім того, був зроблений аналіз та вибір конкретного програмного забезпечення, мови програмування, системи управління базами даних (СУБД) та фреймворків для подальшої розробки автоматизованої системи для автовокзалу. Цей вибір здійснювався з урахуванням усіх вимог та потреб системи з метою забезпечення її ефективності.



### 3 РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ АВТОВОКЗАЛУ

Весь хід розробки програмного забезпечення для такої системи можна умовно поділити на наступні частини:

1. Створення класів доменної області (бізнес-логіки)
2. Створення бази даних на основі описаних класів із використанням Entity Framework та підходу Code first
3. Написання логіки фільтрування, відображення квитків на табло та адміністрування квитків
4. Опис модуля який використовує AI
5. Тестування розробленої системи

Загальна структура проекту зображена на рисунку 3.1 та виглядає наступним чином:

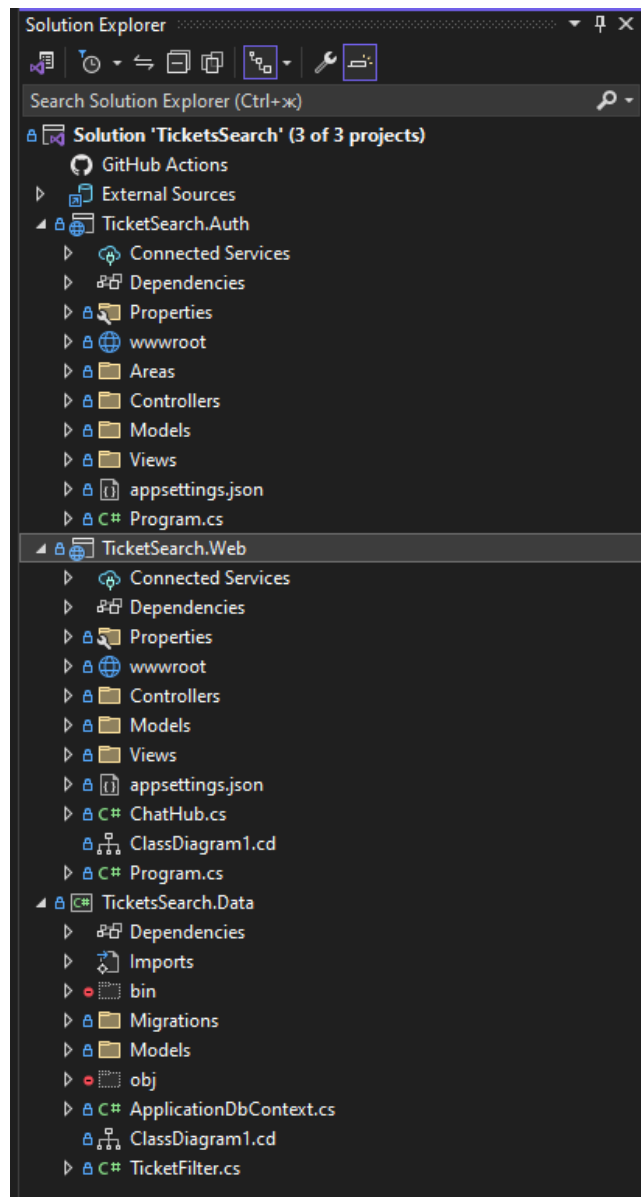


Рисунок 3.1 – Структура проекту

Можна розглянути цю структуру більш детально:

- Connected Services містить підключену базу даних в якій містяться всі дані про квитки (тип квитка, місто відправлення, місто прибуття, дата та час відправлення, дата та час прибуття та ціна для кожного квитка);
- Wwwroot містить css-стилі на підключені бібліотеки які використовуються в проекті;

- Папка `Controllers` містить набір файлів написаних мовою `C#` в яких описані функції які виконують сторінки системи;
- Папка `Models` містить моделі класів;
- Папка `Views` містить `cshtml`-файли які відповідають за візуальне відображення системи;
- Папка `Migrations` містить опис структури бази даних.

Для входу на сайт користувач має попередньо ввести свої дані якщо він є зареєстрованим або зареєструватися якщо це новий користувач даного сайту (також на сайті є підтвердження `email`-адреси), після цього він може перейти до вкладки із таблом яке містить інформацію про квитки та зручний фільтр для пошуку та матиме можливість забронювати квиток. Далі користувач може переглянути на окремій вкладці заброньовані квитки та відмінити бронювання відповідною кнопкою. Все це реалізовано у проекті `TicketSearch.Web`.

### 3.1 Створення класів доменної області (бізнес-логіки)

У даному підрозділі було продумано вимоги доменної області табло автовокзалу і на основі даних вимог створено `C#` класи. Класи було розміщено у відповідній папці `models` (рисунок 3.2), при цьому було використано ООП підхід - всі класи мають модифікатори доступу та властивості - які інкапсулюють дані і дозволяють контролювати доступ до них, за потреби додаючи бізнес-логіку.

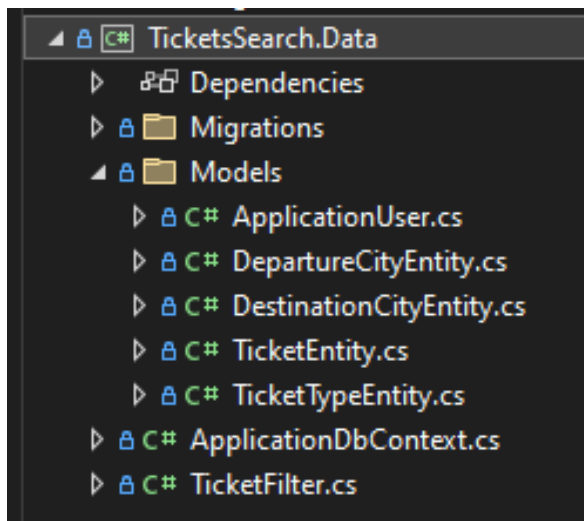


Рисунок 3.2 – Створені класи у папці models

На рисунку 3.3 зображено файл ApplicationUser.cs, який відповідає за реєстрацію та управління користувачами – зберігає логін, пароль, email.

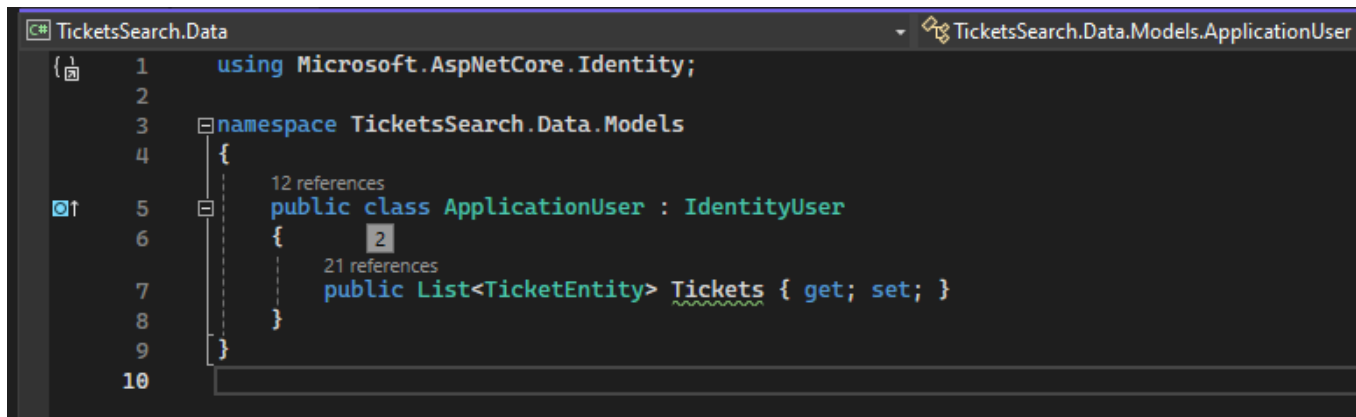


Рисунок 3.3 – Файл ApplicationUser.cs

На рисунку 3.4 зображено файл DepartureCityEntity.cs, який відповідає за місто відправлення і зберігає їх в базі даних.

```

1  using System.ComponentModel.DataAnnotations;
2
3  namespace TicketsSearch.Data.Models
4  {
5      26 references
6      public class DepartureCityEntity
7      {
8          [Key]
9          15 references
10         public Guid Id { get; set; }
11         [Display(Name="Назва")]
12         29 references
13         public string Name { get; set; }
14         [Display(Name = "Квитки")]
15         1 reference
16         public ICollection<TicketEntity> TicketsCollection { get; set; }
17     }
18 }

```

Рисунок 3.4 – Файл DepartureCityEntity.cs

На рисунку 3.5 зображено файл DestinationCityEntity.cs, який відповідає за місто відправлення і зберігає їх в базі даних.

```

1  using System.ComponentModel.DataAnnotations;
2
3  namespace TicketsSearch.Data.Models
4  {
5      26 references
6      public class DestinationCityEntity
7      {
8          [Key]
9          15 references
10         public Guid Id { get; set; }
11         [Display(Name = "Назва")]
12         29 references
13         public string Name { get; set; }
14         [Display(Name = "Квитки")]
15         1 reference
16         public ICollection<TicketEntity> Tickets { get; set; }
17     }
18 }

```

Рисунок 3.5 – Файл DestinationCityEntity.cs

На рисунку 3.6 зображено файл TicketEntity.cs, який відповідає за параметри що має кожен квиток, а саме тип квитка, місто відправлення, місто прибуття, вартість, кількість квитків у базі даних, дата та час відправлення, дата та час прибуття.

```

1  using System.ComponentModel.DataAnnotations;
2  using System.ComponentModel.DataAnnotations.Schema;
3
4  namespace TicketsSearch.Data.Models
5  {
6      public class TicketEntity
7      {
8          [Key]
9          [Display(Name = "Id")]
10         public Guid Id { get; set; }
11         [Display(Name = "Тип")]
12         public int TypeId { get; set; }
13         [Display(Name = "Тип")]
14         public TicketTypeEntity Type { get; set; }
15         [Display(Name = "Місце відправлення")]
16         public Guid DepartureCityId { get; set; }
17         [Display(Name = "Місце відправлення")]
18         public DepartureCityEntity DepartureCity { get; set; }
19         [Display(Name = "Місце прибуття")]
20         public Guid DestinationCityId { get; set; }
21         [Display(Name = "Місце прибуття")]
22         public DestinationCityEntity DestinationCity { get; set; }
23         [Display(Name = "Вартість")]
24         public Decimal Price { get; set; }
25         public IEnumerable<ApplicationUser> Users { get; set; }
26         [Display(Name = "Кількість квитків")]
27         public int TicketsCount { get; set; }
28         [Display(Name = "Дата відправлення")]
29         public DateTime DepartureDate { get; set; }
30         [Display(Name = "Дата прибуття")]
31         public DateTime DestinationDate { get; set; }
32     }
33 }

```

Рисунок 3.6 – Файл TicketEntity.cs

На рисунку 3.7 зображено файл TicketTypeEntity.cs, який відповідає за збереження параметрів типу квитка.

```

TicketsSearch.Data
1  using System.ComponentModel.DataAnnotations;
2
3  namespace TicketsSearch.Data.Models
4  {
5      42 references
6      public class TicketTypeEntity
7      {
8          17 references
9          public int Id { get; set; }
10         [Display(Name = "Назва")]
11         29 references
12         public string Name { get; set; }
13         [Display(Name = "Квитки")]
14         1 reference
15         public ICollection<TicketEntity> Tickets { get; set; }
16     }
17 }

```

Рисунок 3.7 – Файл TicketTypeEntity.cs

Для наглядності відображення класів доменної області (бізнес-логіки) було розроблено UML-діаграму класів, яка зображена на рисунку 3.8.

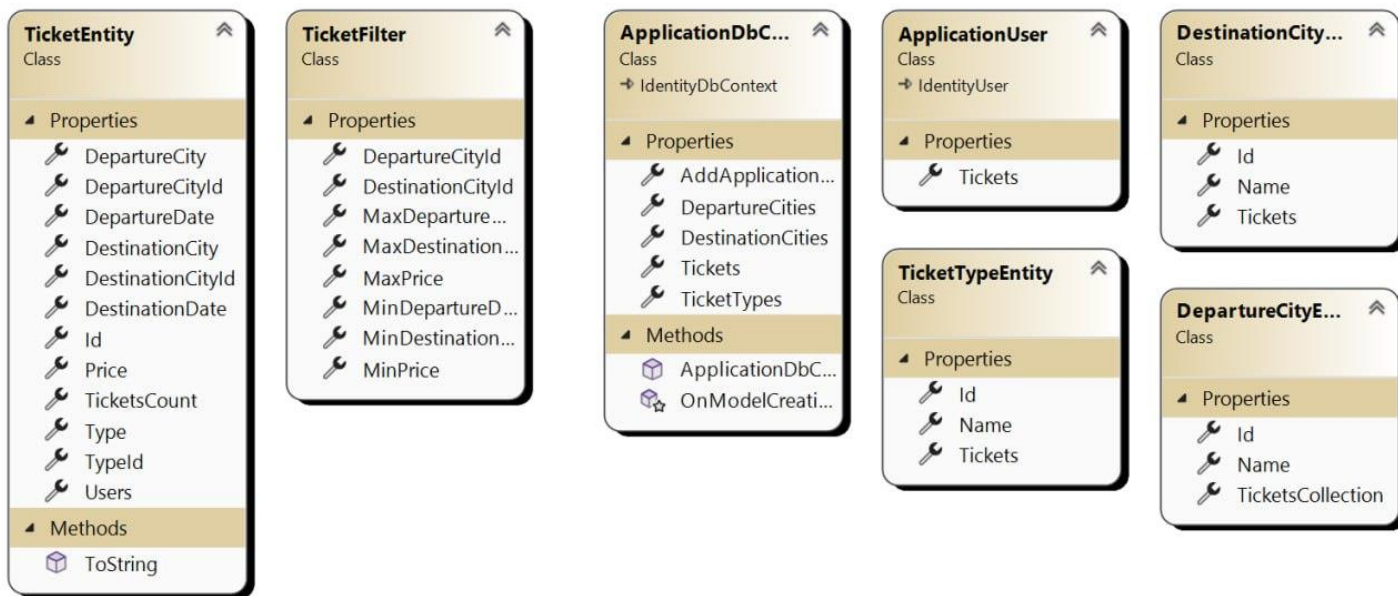


Рисунок 3.8 - UML-діаграма класів

Діаграма відображає основні класи використані в системі, які були відображені на рисунках вище, в якому клас TicketEntity є основною сутністю яка зберігає всі дані про квитки та користувачів, клас TicketFilter виконує функції відображення фільтру та пошуку квитка, клас ApplicationDbContext містить в собі всі сутності, які зберігаються

в базі даних, клас `ApplicationUser` відображає інформацію про користувачів, клас `TicketTypeEntity` відображає інформацію про тип квитка, клас `DestinationCityEntity` містить інформацію про місто прибуття, клас `DepartureCityEntity` містить інформацію про місто відправлення.

### 3.2 Створення бази даних на основі описаних класів із використанням Entity Framework та підходу Code first

У даному підрозділі описується створення та підключення локальної бази даних на основі Microsoft SQL Server до основного проекту. Опис структури бази даних яка зберігає інформацію про квитки, дані зареєстрованого користувача та адміна наведено на рисунку 3.9.

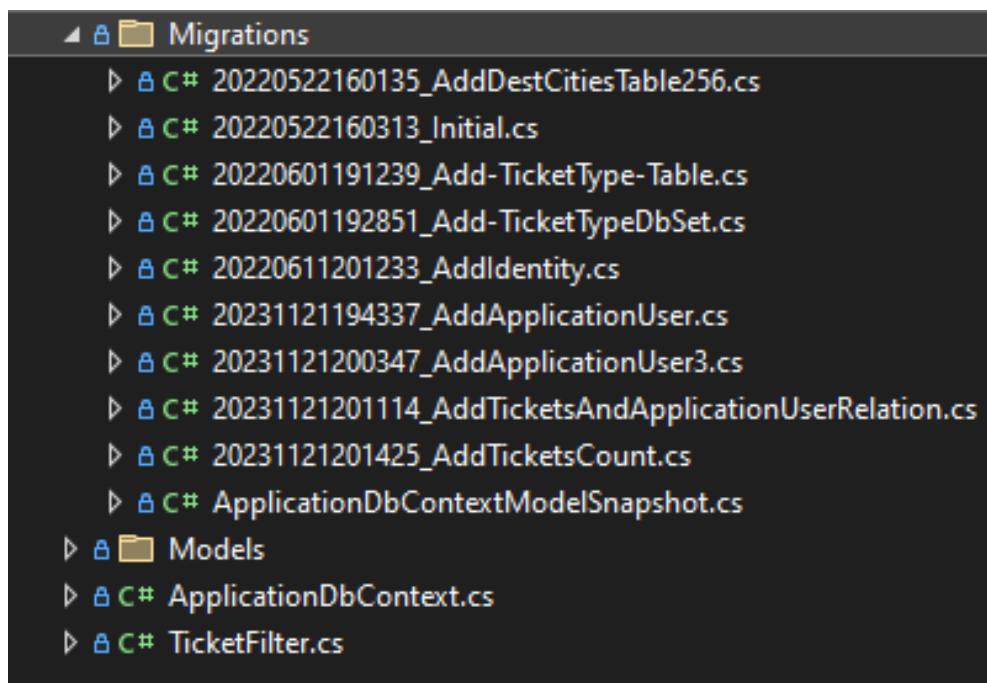


Рисунок 3.9 – Опис структури бази даних

Представлення таблиць в базі даних із використанням Entity Framework Core наведено в фрагменті коду зображеному на рисунку 3.10.



```

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using TicketsSearch.Data.Migrations;
using TicketsSearch.Data.Models;

namespace TicketsSearch.Data
{
    34 references
    public class ApplicationDbContext : IdentityDbContext
    {
        5 references
        public DbSet<ApplicationUser> AddApplicationUsers { get; set; }
        58 references
        public DbSet<TicketEntity> Tickets { get; set; }
        14 references
        public DbSet<TicketTypeEntity> TicketTypes { get; set; }
        30 references
        public DbSet<DepartureCityEntity> DepartureCities { get; set; }
        30 references
        public DbSet<DestinationCityEntity> DestinationCities { get; set; }
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
    }
}

```

Рисунок 3.10 – Представлення таблиць в базі даних

На рисунку 3.11 зображено конфігурацію підключеної до системи бази даних:

SQL Server Instance	
<b>General</b>	
Connection string	Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=master;Integrated Security=True;Connect Timeout=30;E
Default Database Location	C:\Users\nk\AppData\Local\Microsoft\Microsoft SQL Server Local DB\Instances\MSSQLLocalDB
Is Clustered	False
Language	Англійська (США)
Memory	8088 (MB)
Operating System	Microsoft Windows NT 6.3 (19045)
Platform	NT x64
Processors	8
Product	Microsoft SQL Server Express Edition (64-bit)
Server Collation	SQL_Latin1_General_CP1_CI_AS
Version	15
<b>Advanced</b>	
Allow Triggers to Fire Others	True
Blocked Process Threshold	0
Cost Threshold for Parallelism	5
Cursor Threshold	-1
Default Full-Text Language	1033
Filestream Access Level	Disabled
Locks	0
Max Degree of Parallelism	0
Max Text Replication Size	65536
Network Packet Size	4096
Optimize for Adhoc Workloads	False
Query Wait	-1
Remote Login Timeout	10
Scan for Startup Procs	False
Two Digit Year Cutoff	2049
<b>Connections</b>	
ANSI NULL Default Off	False
ANSI NULL Default On	False
Ansi Padding	False
Ansi Warnings	False
AnsiNulls	False
Arithmetic Abort	False

Рисунок 3.11 – Конфігурація підключеної бази даних

Збережені дані про квитки в підключеній до системи базі даних знаходяться у файлі dbo.Tickets та зображені на рисунку 3.12.

Id	Typeld	DepartureCityId	DestinationCityId	Price	DepartureDate	DestinationDate	TicketsCount
c98-1485f3b0544e	2	e7b6645d-8e0a...	8fd710c5-d502-...	800,00	30.11.2023 22:2...	01.12.2023 4:40...	7
28aa97a5-ee41-...	1	5f0d1b94-78c7-...	66cdf845-684f-...	1200,00	28.11.2023 22:2...	29.11.2023 10:3...	41
2909aac5-e88b-...	2	a7950fe8-ef52-...	8c57d4f3-e759-...	348,00	15.12.2023 10:5...	15.12.2023 22:5...	67
bd978d5d-f899-...	1	ff2482d5-6f9a-4...	a098da08-cb2d-...	450,00	12.12.2023 20:0...	12.12.2023 23:4...	165
6e346b56-5e4e-...	1	142d31e3-d7a5-...	a023d94f-7e4d-...	120,00	16.12.2023 2:50...	17.12.2023 4:31...	4
5b30a644-4890-...	1	d244940c-684d-...	8fd710c5-d502-...	255,00	16.11.2023 22:0...	17.11.2023 10:0...	90
e55a1eda-8ff2-...	1	142d31e3-d7a5-...	ba22a0c5-cd54-...	125,00	03.12.2023 20:4...	03.12.2023 22:1...	91
78475711-07e1-...	1	65c20927-38c3-...	f3828196-c492-...	344,00	10.12.2023 19:4...	11.12.2023 1:51...	156
0b6dc4b7-2d04-...	2	d43f9182-b82d-...	8d05930b-a3aa-...	456,00	16.12.2023 2:50...	17.12.2023 4:31...	86
69b8062e-b813-...	2	e7b6645d-8e0a...	d84fea0e-0c5c-...	450,00	22.09.2023 15:1...	23.09.2023 19:1...	13
f51744e7-c59a-...	2	d43f9182-b82d-...	66cdf845-684f-...	1200,00	01.01.2024 7:30...	01.02.2024 13:3...	68
4a9caa26-4b60-...	2	dce3de3b-4cf4-...	a023d94f-7e4d-...	80,00	10.12.2023 4:45...	10.12.2023 6:30...	65
6b1680ce-7ce7-...	1	bbc4bb8e-5786-...	328b8ee5-8998-...	125,00	26.09.2023 20:4...	27.09.2023 4:50...	56
8db0e86f-2812-...	2	e7b6645d-8e0a...	90e43e2b-23e1-...	456,00	11.12.2023 14:1...	12.12.2023 7:40...	144
3e1a3840-0275-...	1	f359183d-9419-...	8d05930b-a3aa-...	175,00	30.11.2023 23:0...	01.12.2023 4:00...	72
9b834277-b69a-...	2	e7b6645d-8e0a...	d84fea0e-0c5c-...	659,00	22.09.2023 10:2...	23.09.2023 4:20...	78
064607fc-09a8-...	2	e7b6645d-8e0a...	d84fea0e-0c5c-...	675,00	22.11.2023 20:1...	23.09.2023 0:10...	47
fbcc19423-91a6-...	1	b8204bfe-6f6e-...	1bb3e3b-5907-...	560,00	10.12.2023 20:0...	11.12.2023 12:0...	78
0959cea0-9f2b-...	2	88401b5f-8709-...	732ed5ff-aa3f-4...	255,00	10.12.2023 15:4...	10.12.2023 20:4...	43
734a8f3a-cb36-...	1	bbc4bb8e-5786-...	a098da08-cb2d-...	243,00	07.12.2023 6:50...	08.12.2023 10:0...	42
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 3.12 – Збережені дані про квитки в файлі dbo.Tickets

Збережені дані про користувачів системи та адміністратора знаходяться у файлі dbo.AspNetUsers та зображені на рисунку 3.13.

Id	UserName	NormalizedUserName	Email	NormalizedEmail	EmailConfirmed	PasswordHash	SecurityStamp	ConcurrencyStamp	PhoneNumber	PhoneNumberConfirmed	TwoFactorEnabled	LockoutEnd	LockoutEnabled
484e66bd-cb4f-...	dimadidyk2017...	DIMADIDYK2017...	dimadidyk2017...	DIMADIDYK2017...	True	AQAAAAEAAAC...	2IPCHRUCUSO...	02d7207f-56eb-...	NULL	False	False	NULL	True
710f449e-e194-...	dimadidyk2017...	DIMADIDYK2017...	dimadidyk2017...	DIMADIDYK2017...	True	AQAAAAEAAAC...	BHH6ZERNFLW...	7c8e1042-340b-...	NULL	False	False	NULL	True
e12-3d28575476e8	dimadidyk2017...	DIMADIDYK2017...	dimadidyk2017...	DIMADIDYK2017...	True	AQAAAAEAAAC...	GBBRSOQEXT...	b60-2340-bc8e-...	NULL	False	False	NULL	True
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 3.13 – Збережені дані про користувачів та адміністратора в файлі dbo.AspNetUsers

Також, доречно додати в даному пункті ER-модель бази даних яка зображена на рисунку 3.14.

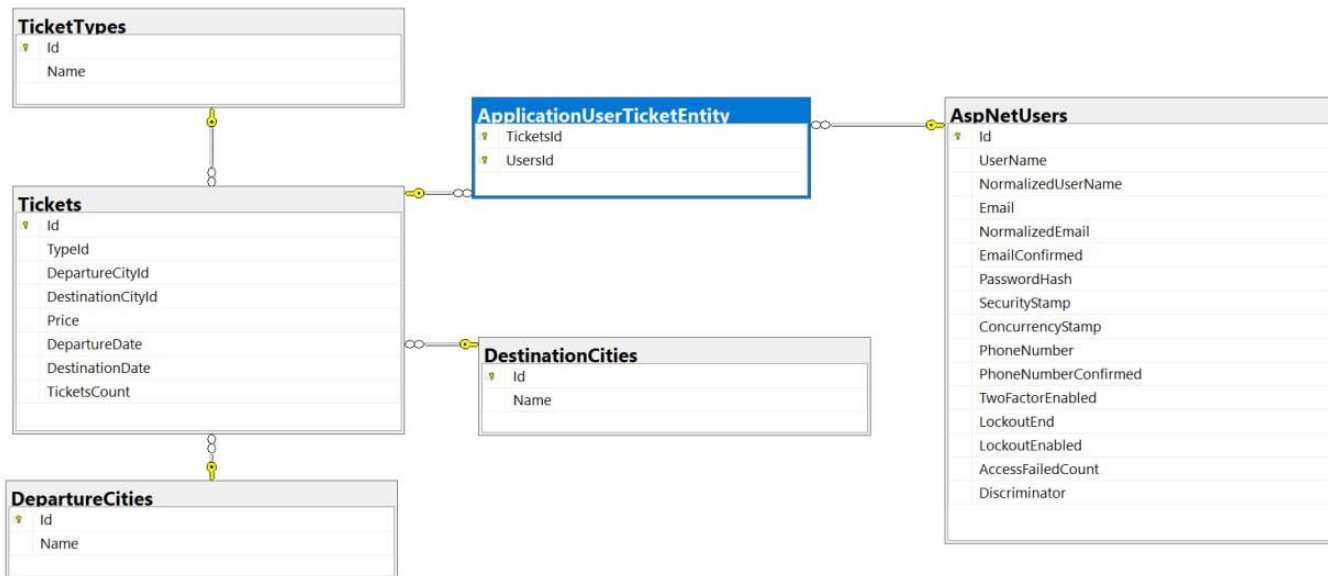


Рисунок 3.14 - ER-модель бази даних

Наведена ER-модель бази даних відображає зв'язок між таблицями, які були описані в базі даних під час розробки системи.

### 3.3 Написання логіки фільтрування, відображення квитків на табло та адміністрування квитків

Опис логіки фільтрування знаходиться в файлі `TicketBoardController.cs` та описана в фрагменті коду яка зображена на рисунку 3.15.

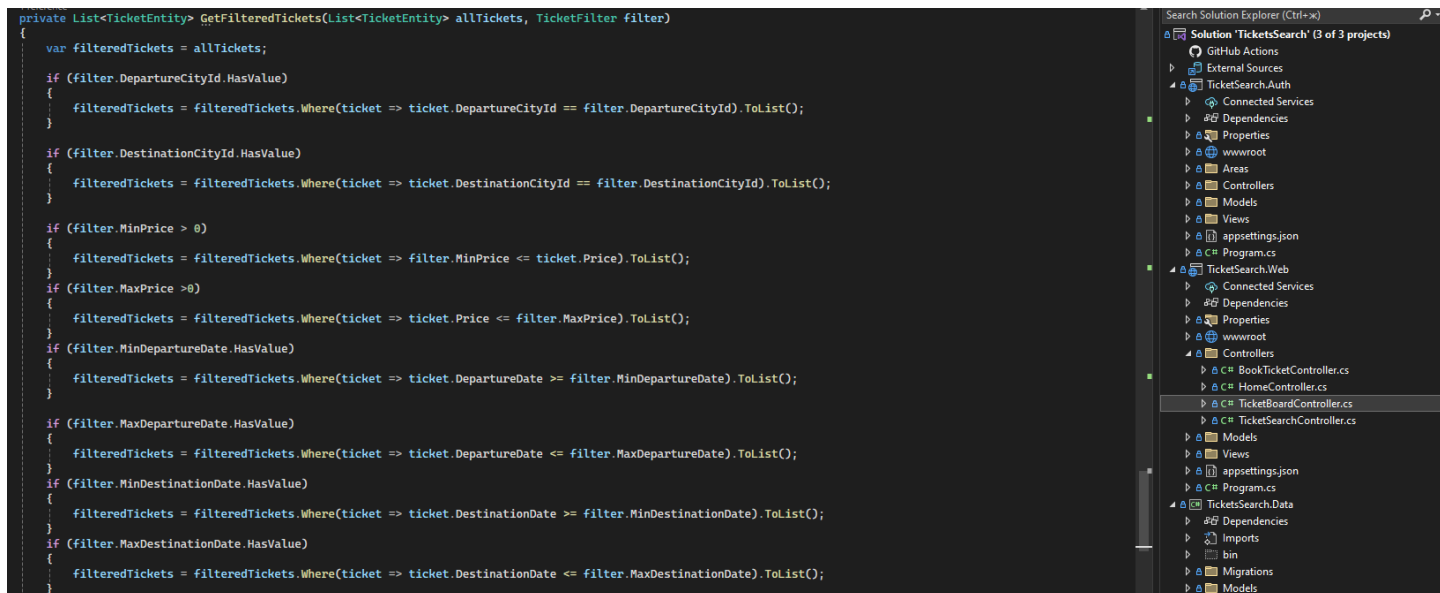


Рисунок 3.15 – Опис логіки фільтрування у файлі TicketBoardController.cs

Цей фрагмент коду представляє метод, який використовується для фільтрації списку квитків з урахуванням заданих у фільтрі умов. Нижче наведено детальний опис роботи цього фрагменту.

Спочатку створюється змінна `filteredTickets`, яка призначена для зберігання проміжного результату фільтрації. Початково ця змінна встановлюється рівною усім квиткам (`allTickets`).

```
var filteredTickets = allTickets;
```

Потім перевіряється, чи у фільтрі задано `DepartureCityId`, і якщо так, фільтр використовується для обмеження списку квитків лише тими, які відправляються з вказаного міста.

```
if (filter.DepartureCityId.HasValue)
```

```
{
    filteredTickets = filteredTickets.Where(ticket => ticket.DepartureCityId ==
filter.DepartureCityId).ToList();
}
```

Аналогічний підхід використовується для `DestinationCityId`, `MinPrice`, `MaxPrice`, `MinDepartureDate`, `MaxDepartureDate`, `MinDestinationDate` і `MaxDestinationDate`. Кожна з цих умов фільтрує список квитків за відповідними критеріями.

На кожному етапі фільтрації використовується `Where` для обрання елементів, які задовольняють умові, та результат конвертується в список за допомогою `ToList()`. На кінцевому етапі повертається фінальний результат фільтрації.

Цей код ефективно використовує лінійну послідовність умов для фільтрації списку квитків з урахуванням різних критеріїв, заданих у фільтрі.

На сторінці фільтр квитків виглядає наступним чином на рисунку 3.16, а більш наглядний варіант роботи цього фільтру буде описано в пункті тестування системи.

Домашня сторінка ДОМАШНЯ СТОРІНКА МОЇ КВИТКИ ТАБЛО КВИТКІВ ПРИВІТ DIMADIDYK2017@GMAIL.COM! ВІЙТИ

ПОКАЗАТИ ФІЛЬТР

Фільтрування квитків

Мінімальна вартість квитка  
0

Максимальна вартість квитка  
0

Мінімальна дата відправлення  
дд.мм.рррр

Максимальна дата відправлення  
дд.мм.рррр

Мінімальна дата прибуття  
дд.мм.рррр

Максимальна дата прибуття  
дд.мм.рррр

Місце відправлення  
Одеса

Місце прибуття  
Донецьк

ЗАСТОСУВАТИ ФІЛЬТР ОЧИСТИТИ ФІЛЬТР

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття	
Маршрутне таксі	Одеса	Донецьк	659,00	78	22.09.2023 10:20:00	23.09.2023 4:20:00	ДЕТАЛІ ЗАБРОНЮВАТИ

Рисунок 3.16 – Відображення фільтру на сторінці

Для відображення квитків на табло існує спеціальна вкладка Табло квитків, у якій відображений список квитків із можливістю подивитися деталі про квиток, бронювання квитка та відображення фільтру квитків для задання параметрів пошуку, це все відображено на рисунку 3.17.

Домашня сторінка ДОМАШНЯ СТОРІНКА МОЇ КВИТКИ ТАБЛО КВИТКІВ ПРИВІТ DIMADIDYK2017@GMAIL.COM!! ВІЙТИ

ПОКАЗАТИ ФІЛЬТР

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття	
Маршрутне таксі	Одеса	Донецьк	659,00	78	22.09.2023 10:20:00	23.09.2023 4:20:00	ДЕТАЛІ   ЗАБРОНЮВАТИ
Маршрутне таксі	Одеса	Донецьк	450,00	13	22.09.2023 15:10:00	23.09.2023 19:10:00	ДЕТАЛІ   ЗАБРОНЮВАТИ
Автобус	Луцьк	Тернопіль	125,00	56	26.09.2023 20:40:00	27.09.2023 4:50:00	ДЕТАЛІ   ЗАБРОНЮВАТИ
Автобус	Миколаїв	Вінниця	255,00	90	16.11.2023 22:00:00	17.11.2023 10:00:00	ДЕТАЛІ   ЗАБРОНЮВАТИ
Маршрутне таксі	Одеса	Донецьк	675,00	47	22.11.2023 20:10:00	23.09.2023 0:10:00	ДЕТАЛІ   ЗАБРОНЮВАТИ
Автобус	Барселона	Лондон	1200,00	40	28.11.2023 22:25:00	29.11.2023 10:30:00	ДЕТАЛІ   ЗАБРОНЮВАТИ
Маршрутне таксі	Одеса	Вінниця	800,00	6	30.11.2023 22:23:00	01.12.2023 4:40:00	ДЕТАЛІ   ЗАБРОНЮВАТИ
Автобус	Дніпро	Харків	175,00	72	30.11.2023 23:00:00	01.12.2023 4:00:00	ДЕТАЛІ   ЗАБРОНЮВАТИ

Рисунок 3.17 – вигляд вкладки Табло квитків зі сторони користувача

Аналогічна вкладка присутня на сторінці зі сторони адміністратора з деякими відмінностями, а саме - перед таблицею присутні кнопки Створити Новий квиток, що дозволяє адміністратору внести зміни у таблицю додавши новий варіант квитка, Імпортувати Квитки, кнопка що дозволяє імпортувати дані про квитки в базу даних у форматі json, а також в самій таблиці відсутня кнопка забронювати квиток але є кнопки Редагувати обраний варіант квитка що дасть змогу змінити його характеристики та Видалити квиток що дає змогу видалити обраний варіант квитка із бази даних. Все це відображено на рисунку 3.18.

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття	РЕДАГУВАТИ	ДЕТАЛІ	ВИДАЛИТИ
Маршрутне таксі	Одеса	Донецьк	659,00	78	22.09.2023 10:20:00	23.09.2023 4:20:00	РЕДАГУВАТИ	ДЕТАЛІ	ВИДАЛИТИ
Маршрутне таксі	Одеса	Донецьк	450,00	13	22.09.2023 15:10:00	23.09.2023 19:10:00	РЕДАГУВАТИ	ДЕТАЛІ	ВИДАЛИТИ
Автобус	Луцьк	Тернопіль	125,00	56	26.09.2023 20:40:00	27.09.2023 4:50:00	РЕДАГУВАТИ	ДЕТАЛІ	ВИДАЛИТИ
Автобус	Миколаїв	Вінниця	255,00	90	16.11.2023 22:00:00	17.11.2023 10:00:00	РЕДАГУВАТИ	ДЕТАЛІ	ВИДАЛИТИ
Маршрутне таксі	Одеса	Донецьк	675,00	47	22.11.2023 20:10:00	23.09.2023 0:10:00	РЕДАГУВАТИ	ДЕТАЛІ	ВИДАЛИТИ
Автобус	Барселона	Лондон	1200,00	40	28.11.2023 22:25:00	29.11.2023 10:30:00	РЕДАГУВАТИ	ДЕТАЛІ	ВИДАЛИТИ
Маршрутне таксі	Одеса	Вінниця	800,00	6	30.11.2023 22:23:00	01.12.2023 4:40:00	РЕДАГУВАТИ	ДЕТАЛІ	ВИДАЛИТИ

Рисунок 3.18 – Вигляд вкладки Табло квитків зі сторони адміністратора

Також до адміністрування можна віднести вкладки Типи квитків, Місця Прибуття, Місця Відправлення та Пошук.

У вкладці Типи квитків можна переглянути ті типи квитків, які відносяться безпосередньо до роботи автовокзалу, а саме автобусний транспорт та маршрутне таксі (рисунок 3.19).

Назва	РЕДАГУВАТИ	ДЕТАЛІ	ВИДАЛИТИ
Автобус	РЕДАГУВАТИ	ДЕТАЛІ	ВИДАЛИТИ
Маршрутне таксі	РЕДАГУВАТИ	ДЕТАЛІ	ВИДАЛИТИ

Рисунок 3.19 – Вигляд вкладки Типи Квитків

У вкладці Міся Прибуття можна побачити список із міст прибуття, які містяться у базі даних (рисунок 3.20).



Рисунок 3.20 – Вигляд вкладки Міся Прибуття

Аналогічно із вкладкою Міся відправлення тільки в ній збережені дані про міста відправлення, які містяться в базі даних (рисунок 3.21).

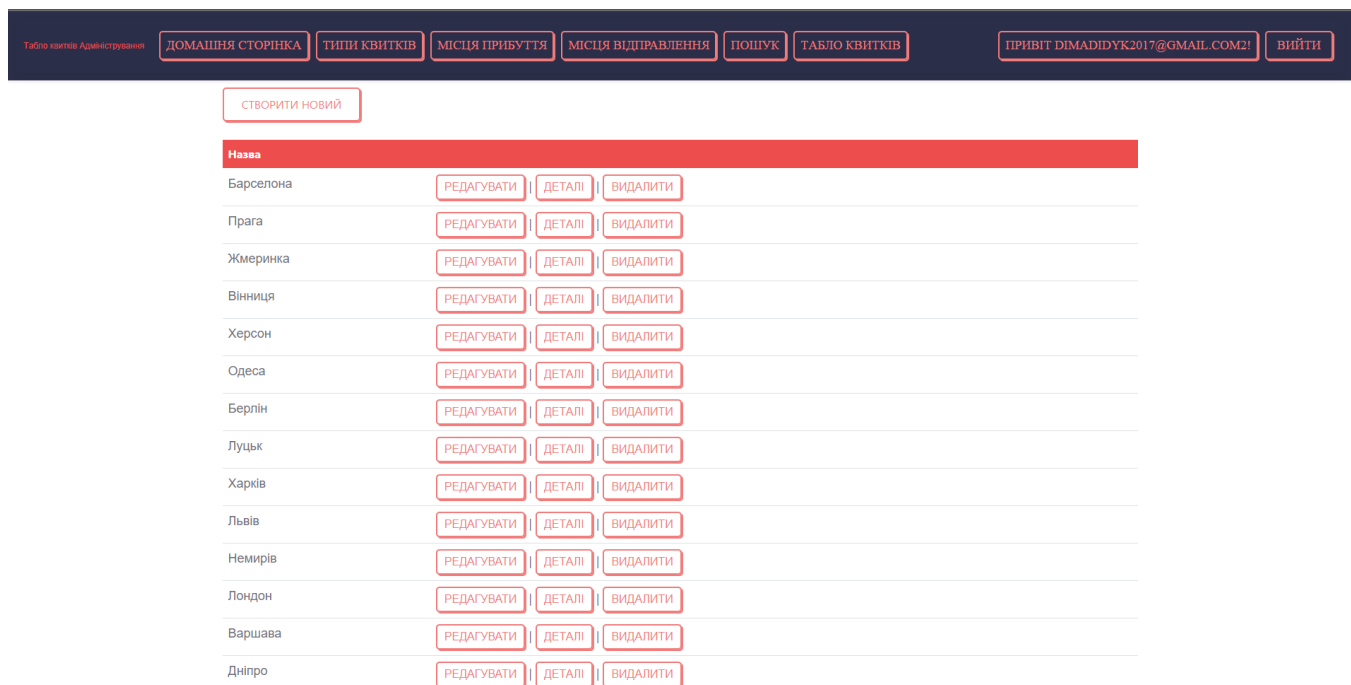


Рисунок 3.21 – Вигляд вкладки Міся Відправлення



Вкладка Пошук зручна тим що можна швидко знайти потрібний квиток за параметрами (кожен з цих параметрів є обов'язковим для введення на відміну від фільтру у вкладці Табло Квитків) та промоніторити їх наявність у базі тому ця вкладка присутня лише зі сторони адміністратора (рисунок 3.22).

Рисунок 3.22 – Вигляд вкладки Пошук

### 3.4 Опис модуля який використовує AI.

Для імплементації модулю чат боту, який дозволяє дізнаватися інформацію про рейси, яка базується на багатьох факторах таких як: місце відправлення та прибуття, час відправлення та прибуття, вартість та наявність квитків на даний рейс, було використано NLP.

Обробка природної мови (NLP) відноситься до галузі штучного інтелекту, яка займається наданням комп'ютерам здатності розуміти текст і слова практично на рівні людей. NLP об'єднує комп'ютерну лінгвістику - моделювання людської мови на основі правил - зі статистичними, машинними та глибокими моделями навчання. Ці технології разом дозволяють комп'ютерам обробляти людську мову у формі тексту або голосових

даних і "розуміти" її повне значення, а також наміри та почуття того, хто говорить чи пише. NLP управляє комп'ютерними програмами, які перекладають текст з однієї мови на іншу, реагують на голосові команди та швидко підсумовують великі обсяги тексту, навіть у режимі реального часу. Прикладами NLP є голосові системи GPS, цифрові помічники, програми для перетворення мови в текст та чат-боти для обслуговування клієнтів. Але NLP також грає все більшу роль у корпоративних рішеннях, що допомагають оптимізувати бізнес-операції, підвищити продуктивність співробітників і спростити критично важливі бізнес-процеси.

У даному випадку метою було забезпечення кінцевих користувачів функцією, яка б дозволяла спростити пошук необхідного квитка, позбавивши їх необхідності переглядати десятки або сотні рейсів, у спробі знайти необхідний квиток. Даний чат бот дозволяє у режимі реального часу, інтерактивно взаємодіяти із системою. Для взаємодії із чат-ботом доступні наступні команди: відправлення, прибуття, вартість.

При виявленні у тексті повідомлення ключових слів, система аналізує запит користувача та намагається знайти співпадіння та закономірності у реченні. У разі успішного знаходження – користувач отримує відповідь. На прикладі команди відправлення, розглянемо роботу системи.

Крок 1. Користувач розпочинає діалог із ботом та отримує відповідь із доступними командами.

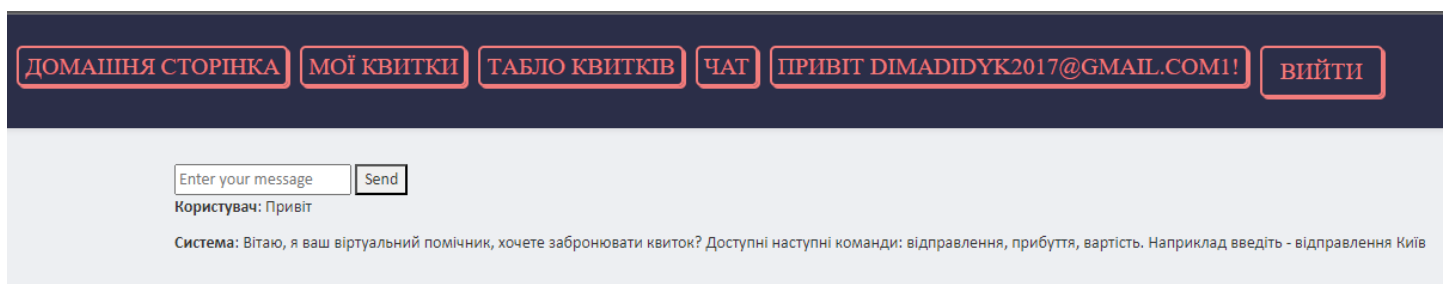


Рисунок 3.23 – початок діалогу користувача із чат-ботом

Крок 2. Користувач вводить команду, система обробляє запит, знаходить ключові слова, у даному випадку команда - “відправлення”, а пошуковий запит – “Вінниця”

(назву міста спеціально написано із маленької літери, дана модель є реєстро незалежною, написана назва міста з маленької літери обробляється відповідним чином і не є проблемою для розуміння) та видає результат, який містить інформацію про наявні рейси.

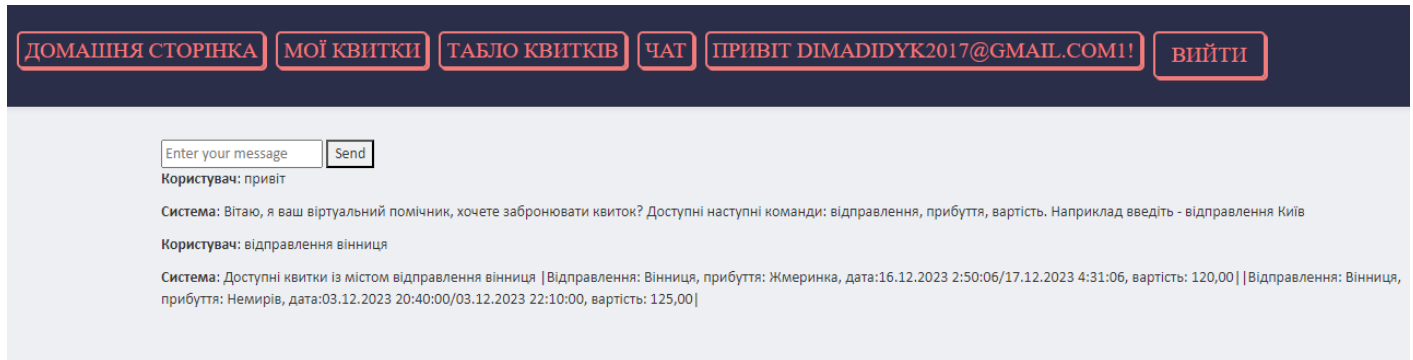


Рисунок 3.24 – відповідь чат-бота на запит користувача

За допомогою цього чат-бота спрощується взаємодія користувача із системою. Більше варіантів взаємодії із ботом буде описано в пункті тестування системи.

### 3.5 Тестування розробленої системи

Тестування програмного забезпечення – це процес перевірки відповідності роботи ПЗ вимогам, специфікаціям та очікуваним результатам. Цей етап розробки виявляє помилки та дефекти, покращує якість продукту, підвищує безпеку та впевненість у його роботі. Тестування включає виконання тест-кейсів, порівняння реальних результатів з очікуваними та виявлення помилок і дефектів.

Тестування є важливим елементом для забезпечення якості програмного забезпечення та впевненості в його надійності. Тестувальники допомагають розробникам виявляти помилки та дефекти, що можуть призвести до збоїв або неправильної роботи ПЗ. Вони також сприяють підвищенню безпеки продукту та поліпшенню його функціональності.

Тестування може бути виконане як вручну, так і з використанням автоматизованих інструментів. Автоматизація тестування – це використання програмних засобів для виконання тестових випадків, оцінки результатів і створення звітів. В залежності від ступеня автоматизації, тестування може бути ручним, напівавтоматичним або повністю автоматичним [51].

Ручне та автоматизоване тестування є важливими на різних рівнях тестування та можуть бути використані в різних типах та видів тестування. Обидва підходи можуть бути включені в різні етапи процесу тестування та сприяють досягненню різних цілей.

Автоматизація тестування має велику перевагу в збереженні часу, зусиль та ресурсів. Один раз створений автоматизований тест можна запускати повторно без значних зусиль.

Хоча мануальне тестування може бути застосовано до практично будь-якого додатка, автоматизоване тестування рекомендується проводити лише для стабільних систем. Автоматизоване тестування часто використовується для регресійного тестування, але не підходить для всіх видів тестування, наприклад, ad-hoc або дослідницького, які краще виконувати вручну.

Мануальне тестування може бути монотонним та повторюваним, але автоматизація допомагає уникнути цього, забезпечуючи ефективність завдяки автоматизованому виконанню завдань.

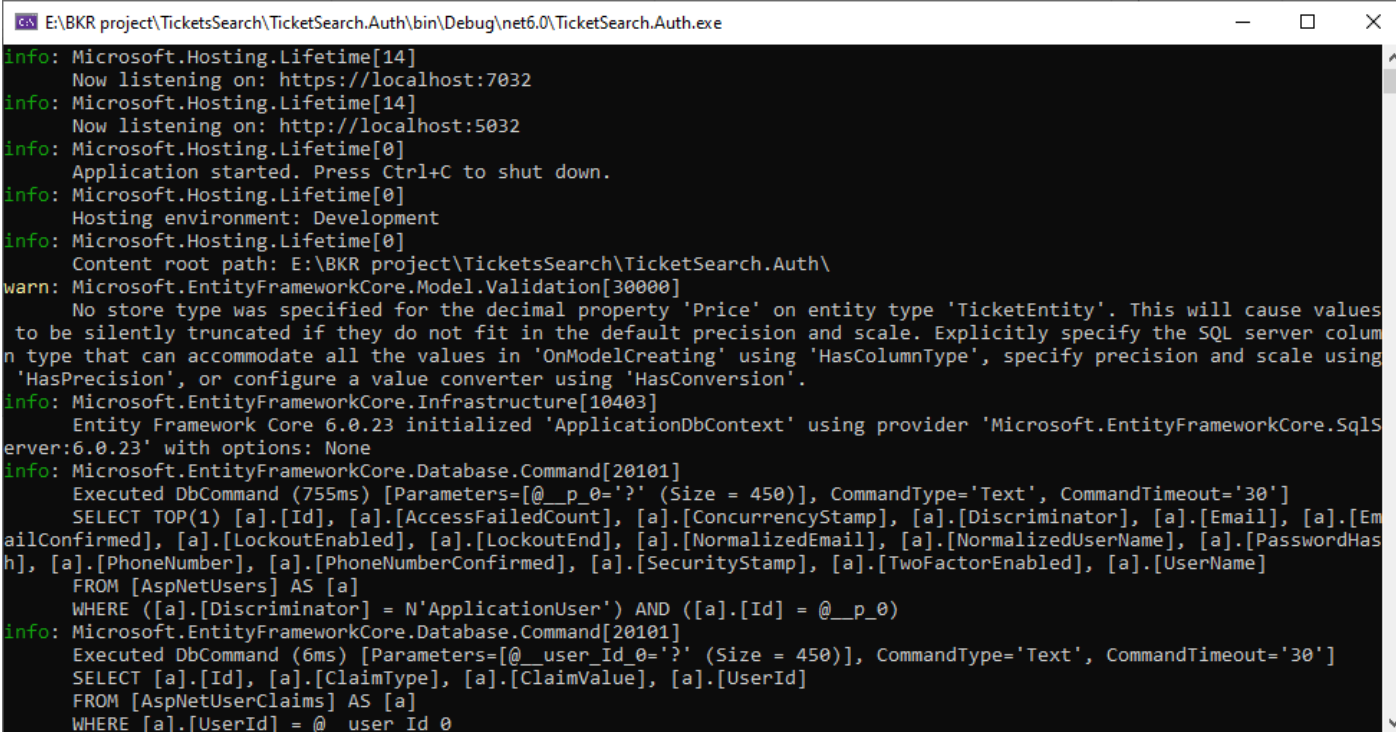
У практиці найчастіше використовується комбінація ручного та автоматизованого тестування, причому ступінь автоматизації залежить від типу проекту та особливостей виробничих процесів у конкретній компанії [52].

Для перевірки правильності роботи розробленої автоматизованої системи автовокзалу було застосовано ручне або мануальне тестування.

Під час проведення мануального тестування, тестувальник виступає в ролі користувача та намагається знайти в програмному забезпеченні помилки та/або неточності в роботі програми.

Вимогами до даної системи є: робочий вхід на сайт як для адміністратора так і для користувача, робочий фільтр квитків та їх коректний пошук із бази даних, робоча можливість бронювання та відміни бронювання користувачем та відображення цих змін у загальному табло квитків як з боку адміністратора сайту так і зі сторони користувача, “user-friendly” інтерфейс.

Для перевірки зазначених вище вимог спочатку потрібно перевірити успішність запуску програми. Це відображено на рисунках 3.25 та 3.26.



```

E:\BKR project\TicketsSearch\TicketSearch.Auth\bin\Debug\net6.0\TicketSearch.Auth.exe
info: Microsoft.Hosting.Lifetime[14]
Now listening on: https://localhost:7032
info: Microsoft.Hosting.Lifetime[14]
Now listening on: http://localhost:5032
info: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
Content root path: E:\BKR project\TicketsSearch\TicketSearch.Auth\
warn: Microsoft.EntityFrameworkCore.Model.Validation[30000]
No store type was specified for the decimal property 'Price' on entity type 'TicketEntity'. This will cause values
to be silently truncated if they do not fit in the default precision and scale. Explicitly specify the SQL server column
n type that can accommodate all the values in 'OnModelCreating' using 'HasColumnType', specify precision and scale using
'HasPrecision', or configure a value converter using 'HasConversion'.
info: Microsoft.EntityFrameworkCore.Infrastructure[10403]
Entity Framework Core 6.0.23 initialized 'ApplicationDbContext' using provider 'Microsoft.EntityFrameworkCore.SqlS
erver:6.0.23' with options: None
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
Executed DbCommand (755ms) [Parameters=@__p_0='?' (Size = 450)], CommandType='Text', CommandTimeout='30'
SELECT TOP(1) [a].[Id], [a].[AccessFailedCount], [a].[ConcurrencyStamp], [a].[Discriminator], [a].[Email], [a].[Em
ailConfirmed], [a].[LockoutEnabled], [a].[LockoutEnd], [a].[NormalizedEmail], [a].[NormalizedUserName], [a].[PasswordHas
h], [a].[PhoneNumber], [a].[PhoneNumberConfirmed], [a].[SecurityStamp], [a].[TwoFactorEnabled], [a].[UserName]
FROM [AspNetUsers] AS [a]
WHERE ([a].[Discriminator] = N'ApplicationUser') AND ([a].[Id] = @__p_0)
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
Executed DbCommand (6ms) [Parameters=@__user_Id_0='?' (Size = 450)], CommandType='Text', CommandTimeout='30'
SELECT [a].[Id], [a].[ClaimType], [a].[ClaimValue], [a].[UserId]
FROM [AspNetUserClaims] AS [a]
WHERE [a].[UserId] = @__user_Id_0
  
```

Рисунок 3.25 – Успішний запуск проекту TicketSearch.Auth

```

E:\BKR project\TicketsSearch\TicketSearch.Web\bin\Debug\net6.0\TicketSearch.Web.exe
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7038
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5285
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: E:\BKR project\TicketsSearch\TicketSearch.Web\
warn: Microsoft.EntityFrameworkCore.Model.Validation[30000]
      No store type was specified for the decimal property 'Price' on entity type 'TicketEntity'. This will cause values
      to be silently truncated if they do not fit in the default precision and scale. Explicitly specify the SQL server colum
      n type that can accommodate all the values in 'OnModelCreating' using 'HasColumnType', specify precision and scale using
      'HasPrecision', or configure a value converter using 'HasConversion'.
info: Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 6.0.5 initialized 'ApplicationDbContext' using provider 'Microsoft.EntityFrameworkCore.SqlSe
      rver:6.0.2' with options: None
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (510ms) [Parameters=@__normalizedUserName_0='?' (Size = 256), CommandType='Text', CommandTimeo
      ut='30']
      SELECT TOP(1) [a].[Id], [a].[AccessFailedCount], [a].[ConcurrencyStamp], [a].[Discriminator], [a].[Email], [a].[Em
      ailConfirmed], [a].[LockoutEnabled], [a].[LockoutEnd], [a].[NormalizedEmail], [a].[NormalizedUserName], [a].[PasswordHas
      h], [a].[PhoneNumber], [a].[PhoneNumberConfirmed], [a].[SecurityStamp], [a].[TwoFactorEnabled], [a].[UserName]
      FROM [AspNetUsers] AS [a]
      WHERE ([a].[Discriminator] = N'ApplicationUser') AND ([a].[NormalizedUserName] = @__normalizedUserName_0)
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (133ms) [Parameters=@__user_Id_0='?' (Size = 450), CommandType='Text', CommandTimeout='30']
      SELECT [a].[Id], [a].[ClaimType], [a].[ClaimValue], [a].[UserId]
      FROM [AspNetUserClaims] AS [a]
  
```

Рисунок 3.26 – Успішний запуск проекту TicketSearch.Web

Після цього запускаються дві окремі сторінки, на одній буде можливість працювати із системою із правами адміністратора а на іншій під виглядом звичайного користувача. Але для того щоб на цих сторінках мати змогу взаємодіяти із різними функціями системи потрібно попередньо увійти до системи або зареєструвати нового користувача. Функція реєстрації зображена на рисунку 3.27, реєстрація нового користувача на обох сторінках має аналогічний вигляд.

The screenshot shows a dark-themed navigation bar at the top with the following items: 'Табло квитків Адміністрування', 'ДОМАШНЯ СТОРІНКА', 'ТИПИ КВИТКІВ', 'МІСЦЯ ПРИБУТТЯ', 'МІСЦЯ ВІДПРАВЛЕННЯ', 'ПОШУК', 'ТАБЛО КВИТКІВ', 'ЗАРЕЄСТРУВАТИСЯ', and 'УВІЙТИ'. The main heading is 'Register'. Underneath, there are two columns. The left column is titled 'Create a new account.' and contains a form with three fields: 'Email' (filled with 'dimadidyk2017@gmail.com3'), 'Password' (filled with '\*\*\*\*\*'), and 'Confirm password'. Below the form, a red error message reads 'The password and confirmation password do not match.' and a red 'REGISTER' button is visible. The right column is titled 'Use another service to register.' and contains a message: 'There are no external authentication services configured. See this article about setting up this ASP.NET application to support logging in via external services.' At the bottom of the page, a dark footer contains the text '© 2023 - TicketSearch.Auth - Privacy'.

Рисунок 3.27 – Функція реєстрації користувача

Після того як користувач натисне на кнопку реєстрації далі йому потрібно буде підтвердити створення нового аккаунту, на рисунку 3.28 зображено це підтвердження яке на даний момент відбувається на сайті, але в подальшому є можливість інтегрування системи яка буде відправляти на пошту повідомлення з підтвердженням.

The screenshot shows the same dark-themed navigation bar as in Figure 3.27. The main heading is 'Register confirmation'. Below the heading, there is a message: 'This app does not currently have a real email sender registered, see these docs for how to configure a real email sender. Normally this would be emailed: Click here to confirm your account'. At the bottom of the page, a dark footer contains the text '© 2023 - TicketSearch.Auth - Privacy'.

Рисунок 3.28 – Функція підтвердження створення аккаунту

Після підтвердження буде виведене повідомлення що електронну адресу успішно підтверджено та новий аккаунт було створено та внесено в базу даних (рисунок 3.29).

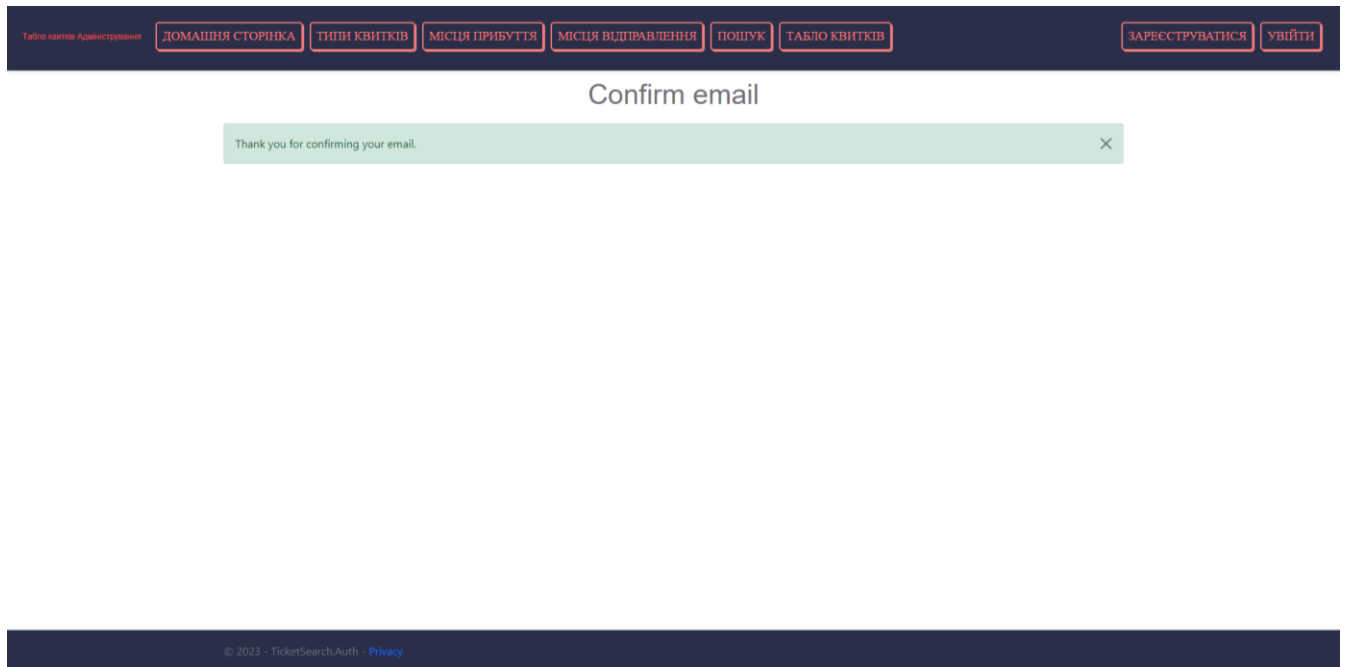


Рисунок 3.29 – Повідомлення про підтвердження даних

Далі з цими даними можна легко зайти на сайт через вкладку Увійти (рисунок 3.30).

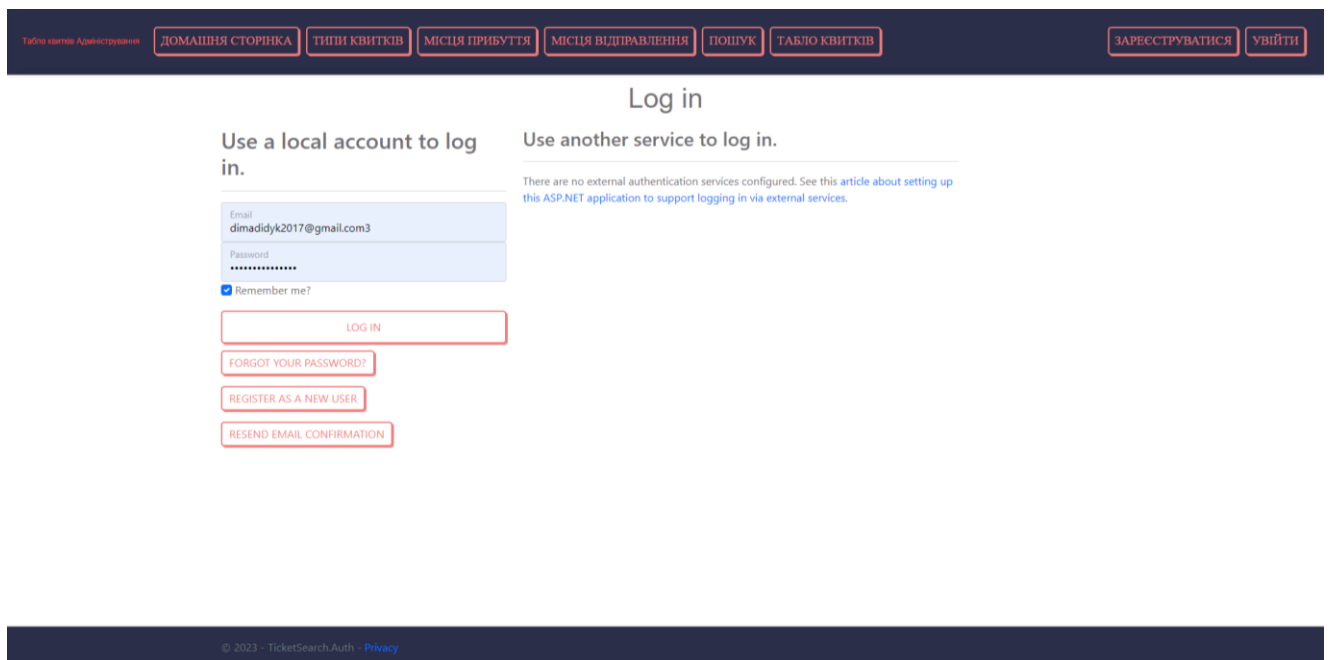


Рисунок 3.30 – Вхід у систему



Увійшовши у систему в правому верхньому куту можна буде пересвідчитися в правильності роботи процесу реєстрації та входу до сайту (рисунок 3.31).

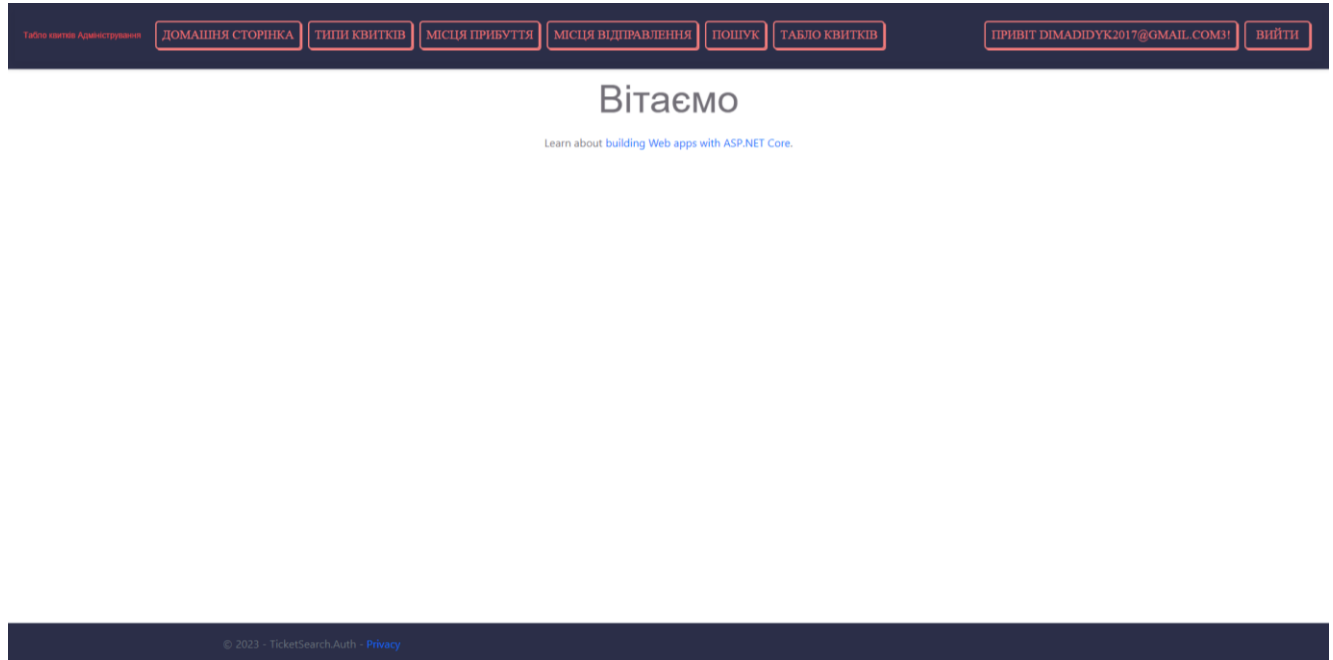


Рисунок 3.31 – Пересвідчення в правильності роботи процесу реєстрації та входу

Аналогічним чином вхід відбувається і зі сторони користувача в проєкті TicketSearch.Web (рисунок 3.32).

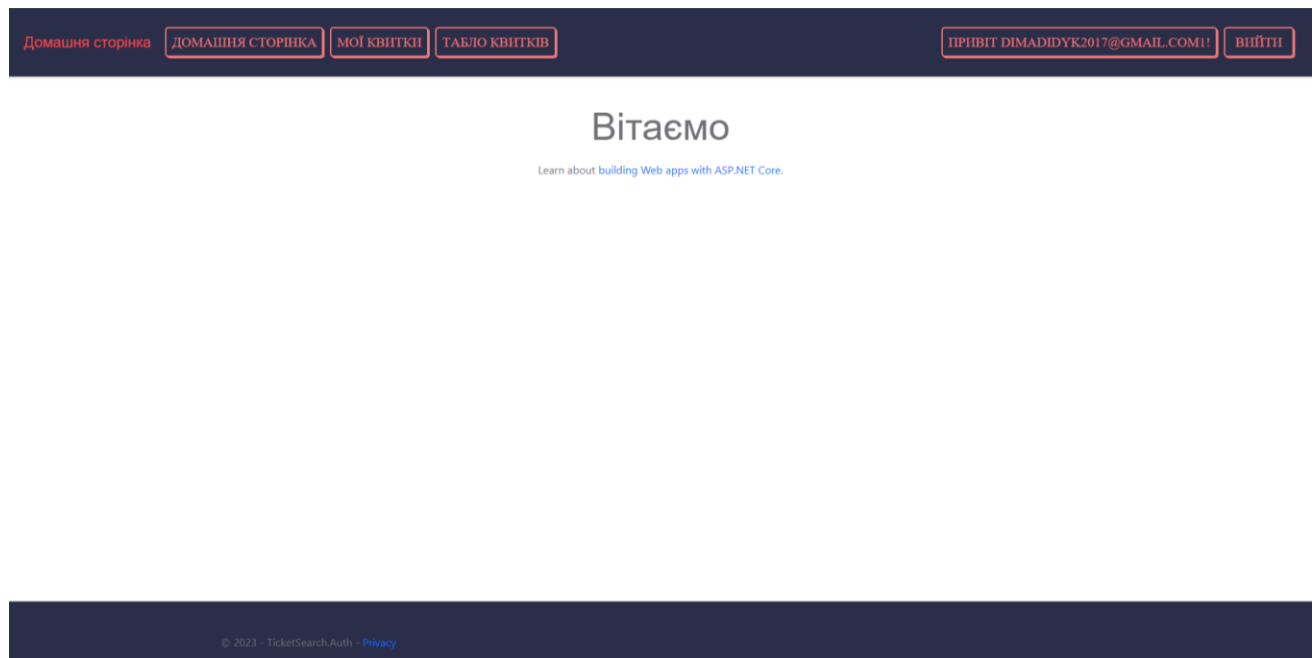


Рисунок 3.32 – Пересвідчення в правильності роботи процесу реєстрації та входу

Далі можна перейти у вкладку Табло квитків та відкрити фільтр для перевірки його роботи та правильності пошуку, наприклад зазначивши містом відправлення Одесу а містом прибуття Донецьк та натиснемо застосувати фільтр (рисунок 3.33 та 3.34).

Домашня сторінка ДОМАШНЯ СТОРІНКА МОЇ КВИТКИ ТАБЛО КВИТКІВ ПРИВІТ DIMADIDYK2017@GMAIL.COM!! ВІЙТИ

ПОКАЗАТИ ФІЛЬТР

Фільтрування квитків

Мінімальна вартість квитка  
0

Максимальна вартість квитка  
0

Мінімальна дата відправлення  
дд.мм.рррр

Максимальна дата відправлення  
дд.мм.рррр

Мінімальна дата прибуття  
дд.мм.рррр

Максимальна дата прибуття  
дд.мм.рррр

Місто відправлення  
Одеса

Місто прибуття  
Донецьк

ЗАСТОСУВАТИ ФІЛЬТР ОЧИСТИТИ ФІЛЬТР

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття	
Маршрутне таксі	Одеса	Донецьк	659,00	78	22.09.2023 10:20:00	23.09.2023 4:20:00	ДЕТАЛІ   ЗАБРОНОВАТИ

Рисунок 3.33 – Введення в фільтр міста відправлення та міста прибуття

Домашня сторінка ДОМАШНЯ СТОРІНКА МОЇ КВИТКИ ТАБЛО КВИТКІВ ПРИВІТ DIMADIDYK2017@GMAIL.COM!! ВІЙТИ

ПОКАЗАТИ ФІЛЬТР

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття	
Маршрутне таксі	Одеса	Донецьк	659,00	78	22.09.2023 10:20:00	23.09.2023 4:20:00	ДЕТАЛІ   ЗАБРОНОВАТИ
Маршрутне таксі	Одеса	Донецьк	450,00	13	22.09.2023 15:10:00	23.09.2023 19:10:00	ДЕТАЛІ   ЗАБРОНОВАТИ
Маршрутне таксі	Одеса	Донецьк	675,00	47	22.11.2023 20:10:00	23.09.2023 0:10:00	ДЕТАЛІ   ЗАБРОНОВАТИ

Рисунок 3.34 – Результат який отримає користувач застосувавши даний фільтр

Щоб переконатися що результат був правильним очистимо фільтр та порівняємо із тим що є в Табло квитків (рисунок 3.35). Як видно з рисунку нижче пошук працює правильно.

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття	
Маршрутне таксі	Одеса	Донецьк	659,00	78	22.09.2023 10:20:00	23.09.2023 4:20:00	ДЕТАЛІ   ЗАБРОНОВАТИ
Маршрутне таксі	Одеса	Донецьк	450,00	13	22.09.2023 15:10:00	23.09.2023 19:10:00	ДЕТАЛІ   ЗАБРОНОВАТИ
Автобус	Луцьк	Тернопіль	125,00	56	26.09.2023 20:40:00	27.09.2023 4:50:00	ДЕТАЛІ   ЗАБРОНОВАТИ
Автобус	Миколаїв	Вінниця	255,00	90	16.11.2023 22:00:00	17.11.2023 10:00:00	ДЕТАЛІ   ЗАБРОНОВАТИ
Маршрутне таксі	Одеса	Донецьк	675,00	47	22.11.2023 20:10:00	23.09.2023 0:10:00	ДЕТАЛІ   ЗАБРОНОВАТИ
Автобус	Барселона	Лондон	1200,00	41	28.11.2023 22:25:00	29.11.2023 10:30:00	ДЕТАЛІ   ЗАБРОНОВАТИ
Маршрутне таксі	Одеса	Вінниця	800,00	7	30.11.2023 22:23:00	01.12.2023 4:40:00	ДЕТАЛІ   ЗАБРОНОВАТИ
Автобус	Дніпро	Харків	175,00	72	30.11.2023 23:00:00	01.12.2023 4:00:00	ДЕТАЛІ   ЗАБРОНОВАТИ

Рисунок 3.35 – Перевірка правильності роботи фільтру та пошуку

Тепер можна в фільтр який вписували назви міст додати максимальну ціну зі значенням 650 (рисунок 3.36)

ПОКАЗАТИ ФІЛЬТР

Фільтрування квитків

Мінімальна вартість квитка  
0

Максимальна вартість квитка  
650

Мінімальна дата відправлення  
дд.мм.рррр

Максимальна дата відправлення  
дд.мм.рррр

Мінімальна дата прибуття  
дд.мм.рррр

Максимальна дата прибуття  
дд.мм.рррр

Місто відправлення  
Одеса

Місто прибуття  
Донецьк

ЗАСТОСУВАТИ ФІЛЬТР    ОЧИСТИТИ ФІЛЬТР

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття	
Маршрутне таксі	Одеса	Донецьк	659,00	78	22.09.2023 10:20:00	23.09.2023 4:20:00	ДЕТАЛІ   ЗАБРОНОВАТИ

Рисунок 3.36 – Додавання нового критерію пошуку

В результаті, як і мало бути, буде знайдено лише 1 квиток що свідчить про те що системи працює правильно (рисунк 3.37).

Домашня сторінка ДОМАШНЯ СТОРІНКА МОЇ КВИТКИ ТАБЛО КВИТКІВ ПРИВІТ DIMADIDYK2017@GMAIL.COM!! ВИЙТИ

ПОКАЗАТИ ФІЛЬТР

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття
Маршрутне таксі	Одеса	Донецьк	450,00	13	22.09.2023 15:10:00	23.09.2023 19:10:00

ДЕТАЛІ | ЗАБРОНЮВАТИ

© 2023 - TicketSearch.Auth - Privacy

Рисунк 3.37 – Перевірка правильності роботи системи із ще одним критерієм пошуку

Для перевірки правильної роботи бронювання квитків, відміни бронювання та відображення цих змін в системі знову потрібно перейти до вкладки Табло квитків та натиснути кнопку забронювати, наприклад, навпроти трьох варіантів квитків, які були знайдені за допомогою фільтру в першому варіанті пошуку після чого користувача перекине у вкладку Мої квитки (рисунк 3.38).

Домашня сторінка [ДОМАШНЯ СТОРІНКА](#) [МОЇ КВИТКИ](#) [ТАБЛО КВИТКІВ](#) [ПРИВІТ DIMADIDYK2017@GMAIL.COM!!](#) [ВИЙТИ](#)

Тип	Місце відправлення	Місце прибуття	Вартість	Дата відправлення	Дата прибуття		
Маршрутне таксі	Одеса	Донецьк	450,00	22.09.2023 15:10:00	23.09.2023 19:10:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ВІДМІНИТИ БРОНУВАННЯ</a>
Маршрутне таксі	Одеса	Донецьк	659,00	22.09.2023 10:20:00	23.09.2023 4:20:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ВІДМІНИТИ БРОНУВАННЯ</a>
Маршрутне таксі	Одеса	Донецьк	675,00	22.11.2023 20:10:00	23.09.2023 0:10:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ВІДМІНИТИ БРОНУВАННЯ</a>

© 2023 - TicketSearch.Auth - Privacy

### Рисунок 3.38 – Заброньовані квитки у вкладці Мої квитки

Далі для перевірки того чи дійсно вони заброньовалися у системі, можна повернутися до вкладки Табло квитків, для зручності застосувати фільтр та побачити що кількість кожного із цих квитків зменшилася на один так як вони заброньовані (рисунок 3.39).

Домашня сторінка [ДОМАШНЯ СТОРІНКА](#) [МОЇ КВИТКИ](#) [ТАБЛО КВИТКІВ](#) [ПРИВІТ DIMADIDYK2017@GMAIL.COM!!](#) [ВИЙТИ](#)

[ПОКАЗАТИ ФІЛЬТР](#)

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття		
Маршрутне таксі	Одеса	Донецьк	659,00	77	22.09.2023 10:20:00	23.09.2023 4:20:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ЗАБРОНУВАТИ</a>
Маршрутне таксі	Одеса	Донецьк	450,00	12	22.09.2023 15:10:00	23.09.2023 19:10:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ЗАБРОНУВАТИ</a>
Маршрутне таксі	Одеса	Донецьк	675,00	46	22.11.2023 20:10:00	23.09.2023 0:10:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ЗАБРОНУВАТИ</a>

© 2023 - TicketSearch.Auth - Privacy

### Рисунок 3.39 – Перевірка змін у системі після бронювання квитків

Також, можна перевірити те ж саме у вкладці Табло квитків але вже з боку адміністратора і побачити такий самий результат (рисунок 3.40)

© 2023 - TicketSearch.Auth - Privacy

Рисунок 3.40 – Перевірка змін у системі після бронювання квитків зі сторони адміністратора

Також можна забронювати декілька однотипних квитків (рисунок 3.41).

© 2023 - TicketSearch.Auth - Privacy

Рисунок 3.41 – Заовлення декількох однотипних квитків

Перевірка бронювання декількох однотипних квитків зображена на рисунку 3.42.

Домашня сторінка ДОМАШНЯ СТОРІНКА МОЇ КВИТКИ ТАБЛО КВИТКІВ ПРИВІТ DIMADIDYK2017@GMAIL.COM!! ВІЙТИ

ПОКАЗАТИ ФІЛЬТР

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття	
Маршрутне таксі	Одеса	Донецьк	659,00	76	22.09.2023 10:20:00	23.09.2023 4:20:00	ДЕТАЛІ   ЗАБРОНЮВАТИ
Маршрутне таксі	Одеса	Донецьк	450,00	12	22.09.2023 15:10:00	23.09.2023 19:10:00	ДЕТАЛІ   ЗАБРОНЮВАТИ
Маршрутне таксі	Одеса	Донецьк	675,00	46	22.11.2023 20:10:00	23.09.2023 0:10:00	ДЕТАЛІ   ЗАБРОНЮВАТИ

© 2023 - TicketSearch.Auth - Privacy

Рисунок 3.42 – Перевірка функції бронювання декількох квитків

Для перевірки відміни бронювання потрібно натиснути на кнопку відмінити бронювання і, наприклад, залишити один квиток (рисунок 3.43), а потім перейти до пункту Табло квитків та перевірити зміни (рисунок 3.44).

Домашня сторінка ДОМАШНЯ СТОРІНКА МОЇ КВИТКИ ТАБЛО КВИТКІВ ПРИВІТ DIMADIDYK2017@GMAIL.COM!! ВІЙТИ

Тип	Місце відправлення	Місце прибуття	Вартість	Дата відправлення	Дата прибуття	
Маршрутне таксі	Одеса	Донецьк	450,00	22.09.2023 15:10:00	23.09.2023 19:10:00	ДЕТАЛІ   ВІДМІНИТИ БРОНЮВАННЯ

© 2023 - TicketSearch.Auth - Privacy

Рисунок 3.43 – Вкладка Мої квитки після відміни бронювання

Як видно з рисунку 3.44 кількість квитків Одеса – Донецьк вартістю 659 та 675 гривень стала такою як і була на початку тестування (тобто збільшилася на один так як бронювання було знято), а кількість квитків вартістю 450 гривень залишилася без змін так як один квиток все ще заброньований даним користувачем.

Домашня сторінка
ПРИВІТ DIMADIDYK2017@GMAIL.COM!!

ПОКАЗАТИ ФІЛЬТР
ДОМАШНЯ СТОРІНКА
МОЇ КВИТКИ
ТАБЛО КВИТКІВ
ВІЙТИ

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття	
Маршрутне таксі	Одеса	Донецьк	659,00	78	22.09.2023 10:20:00	23.09.2023 4:20:00	ДЕТАЛІ   ЗАБРОНЮВАТИ
Маршрутне таксі	Одеса	Донецьк	450,00	12	22.09.2023 15:10:00	23.09.2023 19:10:00	ДЕТАЛІ   ЗАБРОНЮВАТИ
Маршрутне таксі	Одеса	Донецьк	675,00	47	22.11.2023 20:10:00	23.09.2023 0:10:00	ДЕТАЛІ   ЗАБРОНЮВАТИ

© 2023 - TicketSearch.Auth - Privacy

Рисунок 3.44 – Перевірка змін

На рисунку 3.45 проводиться перевірка роботи чат-боту із запитом за командами відправлення Вінниця, прибуття Донецьк, вартість 125 та відправлення Одеса прибуття Донецьк, на що система вивела список квитків із бази даних, які відповідають даним критеріям, а також було введено повідомлення яке не відповідає ключовим словам на що система виводить відповідь “Ваше повідомлення не розпізнано”.





Рисунок 3.45 – Перевірка роботи чат-боту

Відповідно до проведених тестів, описаних у даному підпункті, можна з впевненістю сказати, що тести підтвердили працездатність системи.

### 3.6 Висновки

В цьому розділі було описано розробку автоматизованої системи автовокзалу, а саме описано створення класів доменної області (бізнес-логіки), створення бази даних на основі описаних класів із використанням Entity Framework та підходу Code first, логіку фільтрування, відображення квитків на табло та адміністрування квитків, а також було проведено мануальне тестування для перевірки правильності роботи основних функцій програмного забезпечення. Якщо говорити про перспективи використання розробленого програмного забезпечення то тут можна відмітити наступне: впровадження системи може значно полегшити роботу автовокзалу, зменшити ймовірність помилок та покращити обслуговування пасажирів, а інтеграція з онлайн-

сервісами та мобільними додатками може забезпечити зручний доступ до інформації та послуг для користувачів.

Загалом, розроблена система має великий потенціал для оптимізації роботи автовокзалів, покращення взаємодії з пасажирами та впровадження сучасних технологій у галузь транспортного обслуговування.

## ВИСНОВКИ

У магістерській кваліфікаційній роботі було проведено аналіз проблеми розробки автоматизованої системи автовокзалу та аналіз існуючих автоматизованих систем у сфері транспорту. Після цього було вирішено створити власну систему із відкритим вихідним кодом.

Для розробки було використано наступні технології: .NET 6, C#, Entity Framework Core, Microsoft SQL Server.

У першому розділі було виконано аналіз проблеми, пов'язаної з розробкою автоматизованої системи для автовокзалу. Було детально розглянуто стан дослідження цієї науково-технічної задачі, спираючись на аналіз існуючих автоматизованих систем у галузі транспорту. Порівняльний аналіз існуючих систем для пошуку квитків на транспорт дав підстави вважати, що створення програмного продукту є цілком доцільним рішенням.

Далі, було розглянуто сучасні методи вирішення цього завдання та обрано оптимальний варіант для подальшого дослідження і розробки. Крім того, був проведений аналіз можливих нових підходів та методів для вирішення цієї задачі, що підсилить виконання цієї роботи та забезпечить її інноваційність.

Загалом, на основі проведеного аналізу, було визначено, що розробка програмного продукту для автоматизації автовокзалу є належним та обґрунтованим напрямком дослідження та розвитку.

У другому розділі були створені докладні UML-діаграми, включаючи діаграми варіантів використання, діяльності та розгортання, які надають більш детальний опис системи. Дані діаграми детально висвітлили різні аспекти функціонування системи автоматизації автовокзалу, включаючи процеси взаємодії з користувачами та компонентами системи.

Крім того, був зроблений аналіз та вибір конкретного програмного забезпечення, мови програмування, системи управління базами даних (СУБД) та фреймворків для

подальшої розробки автоматизованої системи для автовокзалу. Цей вибір здійснювався з урахуванням усіх вимог та потреб системи з метою забезпечення її ефективності.

У третьому розділі було описано розробку автоматизованої системи автовокзалу, а саме описано створення класів доменної області (бізнес-логіки), створення бази даних на основі описаних класів із використанням Entity Framework та підходу Code first, логіку фільтрування, відображення квитків на табло та адміністрування квитків, а також було проведено мануальне тестування для перевірки правильності роботи основних функцій програмного забезпечення.

Якщо говорити про перспективи використання розробленого програмного забезпечення то тут можна відмітити наступне: впровадження системи може значно полегшити роботу автовокзалу, зменшити ймовірність помилок та покращити обслуговування пасажирів, а інтеграція з онлайн-сервісами та мобільними додатками може забезпечити зручний доступ до інформації та послуг для користувачів.

Результатом виконання магістерської кваліфікаційної роботи є розроблене програмне забезпечення, яке відповідає поставленим вимогам – система має робочий вхід на сайт як для адміністратора так і для користувача, має робочий фільтр квитків та їх коректний пошук із бази даних, наявна робоча можливість бронювання та відміни бронювання квитків користувачем та відображення цих змін у загальному табло квитків як з боку адміністратора сайту так і зі сторони користувача, а також система має зручний та зрозумілий інтерфейс.

Загалом, розроблена система має великий потенціал для оптимізації роботи автовокзалів, покращення взаємодії з пасажирями та впровадження сучасних технологій у галузь транспортного обслуговування.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. What is the Information Revolution? URL: <https://study.com/learn/lesson/information-revolution-global-economy-effects-importance-examples.html> (дата звернення 21.09.2023)
2. The Microprocessor Revolution URL: <https://clearpurpose.media/the-microprocessor-revolution-633278398e95> (дата звернення 21.09.2023)
3. Bus station management system URL: <http://sdditg.com/wp-content/uploads/2017/11/Bus-Station-Management-System.pdf> (дата звернення 21.09.2023)
4. Automatic Bus Location Announcement System – What You Need to Know? URL: <https://transignllc.com/automatic-bus-location-announcement-system-what-you-need-to-know/> (дата звернення 21.09.2023)
5. Search engine URL: <https://www.bdc.ca/en/articles-tools/entrepreneur-toolkit/templates-business-guides/glossary/search-engine> (дата звернення 22.09.2023)
6. Semantic Search Explained: What Is It And Why Does It Matter? URL: <https://www.bloomreach.com/en/blog/2019/semantic-search-explained-in-5-minutes> (дата звернення 22.09.2023)
7. What is the Voice search technology? AT Internet's definition URL: <https://www.atinternet.com/en/glossary/voice-search/> (дата звернення 22.09.2023)
8. Autonomous Bus Passenger experience URL: <https://www.diva-portal.org/smash/get/diva2:1218637/FULLTEXT01.pdf> (дата звернення 23.09.2023)
9. Підвищення ефективності функціонування транспортно-пересадочних вузлів транспортної системи міста URL: <https://eforum.lntu.edu.ua/index.php/jurnal-mbf/article/view/899/853> (дата звернення 23.09.2023)
10. Amadeus IT Group URL: <https://amadeus.com/en> (дата звернення 23.09.2023)
11. Sabre URL: <https://www.sabre.com/products/> (дата звернення 23.09.2023)
12. Sita URL: <https://www.sita.aero/> (дата звернення 23.09.2023)

13. Passenger terminal expo URL: <https://www.passengerterminal-expo.com/en/show-layout.php> (дата звернення 23.09.2023)
14. Cognitec URL: <https://www.cognitec.com/> (дата звернення 23.09.2023)
15. Rapiscan systems URL: <https://www.rapiscansystems.com/en/> (дата звернення 23.09.2023)
16. Trapeze group URL: <https://www.trapezegroup.eu/> (дата звернення 23.09.2023)
17. INIT URL: <https://www.initse.com/ende/company/about-init/> (дата звернення 23.09.2023)
18. Artificial Intelligence as a factor of public transportations system development URL: <https://www.sciencedirect.com/science/article/pii/S2352146522005312> (дата звернення 24.09.2023)
19. Machine Learning Applied to Public Transportation by Bus: A Systematic Literature Review URL: <https://journals.sagepub.com/doi/full/10.1177/03611981231155189> (дата звернення 24.09.2023)
20. Prototype of Bus Ticketing Application Using NFC on Android Device URL: [https://www.researchgate.net/publication/276386069\\_Prototype\\_of\\_Bus\\_Ticketing\\_Application\\_Using\\_NFC\\_on\\_Android\\_Device](https://www.researchgate.net/publication/276386069_Prototype_of_Bus_Ticketing_Application_Using_NFC_on_Android_Device) (дата звернення 24.09.2023)
21. Use of Big Data Analytics for Public Transport Efficiency URL: <https://www.diva-portal.org/smash/get/diva2:1736700/FULLTEXT01.pdf> (дата звернення 25.09.2023)
22. IoT Connectivity for Public Transportation URL: <https://datablaze.com/resources/articles/iot-connectivity-for-public-transportation/> (дата звернення 25.09.2023)
23. Bus Transportation using Face Recognition System URL: [https://www.researchgate.net/publication/324798015\\_Bus\\_Transportation\\_using\\_Face\\_Recognition\\_System](https://www.researchgate.net/publication/324798015_Bus_Transportation_using_Face_Recognition_System) (дата звернення 25.09.2023)
24. Case Study Of Automated Baggage Handling In Modern Transportations URL: [https://www.researchgate.net/publication/339799822\\_Case\\_Study\\_Of\\_Automated\\_Baggage\\_Handling\\_In\\_Modern\\_Transportations](https://www.researchgate.net/publication/339799822_Case_Study_Of_Automated_Baggage_Handling_In_Modern_Transportations) (дата звернення 26.09.2023)

25. Artificial intelligence in transportation advantages, disadvantages & applications URL: <https://www.online-sciences.com/robotics/artificial-intelligence-in-transportation-advantages-disadvantages-applications/> (дата звернення 27.09.2023)
26. The Advantages and Disadvantages of IoT in Transportation URL: <https://ts2.space/en/the-advantages-and-disadvantages-of-iot-in-transportation/> (дата звернення 27.09.2023)
27. Advantages and disadvantages of blockchain payments URL: <https://www.payretailers.com/en/latam/blockchain-payments/> (дата звернення 28.09.2023)
28. What Are the Benefits of Mobile Ticketing in the Transportation sector URL: <https://www.modeshift.com/what-are-the-benefits-of-mobile-ticketing-in-the-transportation-sector/> (дата звернення 29.09.2023)
29. Big Data analysis: Intelligent Transportation Development Engine URL: [https://www.researchgate.net/publication/292075969\\_Big\\_Data\\_analysis\\_Intelligent\\_Transportation\\_Development\\_Engine](https://www.researchgate.net/publication/292075969_Big_Data_analysis_Intelligent_Transportation_Development_Engine) (дата звернення 30.09.2023)
30. Advantages and Disadvantages of Using Security Cameras URL: <https://www.a1securitycameras.com/blog/advantages-and-disadvantages-of-using-security-cameras/> (дата звернення 01.10.2023)
31. The Pros and Cons of Using Autonomous Vehicles for Delivery URL: <https://www.locate2u.com/route-planning/pros-and-cons-of-using-autonomous-vehicles-for-delivery/> (дата звернення 01.10.2023)
32. Основи UML URL: <https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html> (дата звернення 02.10.2023)
33. Use case diagram URL: <https://www.techtarget.com/whatis/definition/use-case-diagram> (дата звернення 02.10.2023)
34. What's an activity diagram? URL: <https://www.mindmanager.com/en/features/activity-diagram/> (дата звернення 03.10.2023)

35. Deployment diagrams URL: <https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-deployment> (дата звернення 03.10.2023)
36. Що таке PHP? URL: <https://hyperhost.ua/uk/wiki/chto-takoe-php> (дата звернення 04.10.2023)
37. Advantages and Disadvantages of PHP You Should Know When Starting a New Project URL: <https://anywhere.epam.com/business/pros-and-cons-of-php> (дата звернення 04.10.2023)
38. Що таке ASP.NET? Принцип функціонування та моделі розробки URL: <https://highload.today/uk/shho-take-asp-net-printsip-funktsionuvannya-ta-modeli-rozrobki/> (дата звернення 05.10.2023)
39. ASP.NET vs PHP URL: <https://www.orientsoftware.com/technologies/microsoft-net/aspnet-vs-php/> (дата звернення 05.10.2023)
40. Основні відомості про мову програмування C# URL: <https://armedsoft.com/ua/blog/osnovni-vidomosti-pro-movu-programuvannya-c> (дата звернення 06.10.2023)
41. Encapsulation - definition & overview URL: <https://www.sumologic.com/glossary/encapsulation/> (дата звернення 07.10.2023)
42. Inheritance (object-oriented programming) URL: [https://en.wikipedia.org/wiki/Inheritance\\_\(object-oriented\\_programming\)](https://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)) (дата звернення 07.10.2023)
43. OOP Concepts for Beginners: What is Polymorphism URL: <https://stackify.com/oop-concept-polymorphism/> (дата звернення 07.10.2023)
44. Microsoft SQL Server URL: <https://www.techtarget.com/searchdatamanagement/definition/SQL-Server> (дата звернення 10.10.2023)
45. MS SQL SERVER HISTORY AND ADVANTAGES URL: <https://bytescout.com/blog/2014/09/ms-sql-server-history-and-advantages.html> (дата звернення 10.10.2023)



46. The Advantages and Disadvantages of Microsoft SQL Server URL: <https://www.rothrobot.com/the-advantages-and-disadvantages-of-microsoft-sql-server/> (дата звернення 10.10.2023)
47. Entity Framework Core URL: <https://learn.microsoft.com/en-us/ef/core/> (дата звернення 12.10.2023)
48. Introduction to Entity Framework Core URL: <https://www.partech.nl/nl/publicaties/2020/11/introduction-to-entity-framework> (дата звернення 12.10.2023)
49. ASP.NET MVC Pattern URL: <https://dotnet.microsoft.com/en-us/apps/aspnet/mvc> (дата звернення 14.10.2023)
50. What Is MVC With ASP.NET? (With Features and Advantages) URL: <https://in.indeed.com/career-advice/career-development/what-is-mvc-with-asp-net> (дата звернення 18.10.2023)
51. Основи QA - Тестування програмного забезпечення URL: <https://lemon.school/blog/osnovy-qa> (дата звернення 01.11.2023)
52. Ручне та автоматизоване тестування - QALight URL: <https://qalight.ua/baza-znaniy/ruchne-ta-avtomatizovane-testuvannya/> (дата звернення 10.11.2023)
53. Дідик Д. М., Ковалюк О. О. «Розробка автоматизованої системи автовокзалу», «НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ ФАКУЛЬТЕТУ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА АВТОМАТИЗАЦІЇ», 2023. URL: <https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/19683/16324> (дата звернення 13.12.2023)

## ДОДАТКИ

**Додаток А**  
(обов'язковий)  
Технічне завдання

ЗАТВЕРДЖУЮ  
Зав. кафедри АПТ,  
д.т.н., проф.  
\_\_\_\_\_ Олег БІСІКАЛЮ

“ 12 ” жовтня 2023 р.

**ТЕХНІЧНЕ ЗАВДАННЯ**  
на магістерську кваліфікаційну роботу  
«РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ АВТОВОКЗАЛУ»  
08-31.МКР.005.02.000 ТЗ

Керівник роботи:  
к.т.н., доц. каф. КСУ  
Олег КОВАЛЮК  
«12» жовтня 2023 р.

Виконавець:  
Ст. гр. 3Акіт-22м  
Дмитро ДІДИК  
«12» жовтня 2023 р.

## 1. Назва та галузь застосування

1.1. Назва – Розробка автоматизованої системи автовокзалу.

1.2. Галузь застосування – Пасажирські перевезення.

## 2. Підстава для проведення розробки.

Підставою для виконання роботи є наказ №247 по ВНТУ від «18» 09 2023 р. та індивідуальне завдання на МКР, затверджене протоколом №1 засідання кафедри АІТ від «30» 08 2023 р.

## 3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є поліпшення якості обслуговування, підвищення ефективності та забезпечення конкурентоспроможності системи автовокзалу.

## 4. Джерела розробки.

1. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальностей 126 – «Інформаційні системи та технології», 151 – «Автоматизація та комп'ютерно-інтегровані техно-логії» / Уклад. Р. Н. Кветний, О. М. Бевз, О. В. Бісікало., Р. В. Маслій – Вінниця : ВНТУ, 2020.

2. Development of a computer system at the bus station URL:

<http://eir.zntu.edu.ua/handle/123456789/7589> (дата звернення 13.10.2023)

3. Розробка інформаційно-керуючої системи для автовокзалу URL:

<https://cardfile.ontu.edu.ua/server/api/core/bitstreams/09420ce8-2238-4cf1-b9f6-25b75a9d1e90/content> (дата звернення 20.10.2023)

## 5. Показники призначення

5.1. Перелік головних функцій:

- вхід на сайт як для адміністратора так і для користувача;
- фільтр квитків та їх коректний пошук із бази даних;

- можливість бронювання та відміни бронювання користувачем та відображення цих змін у загальному табло квитків як з боку адміністратора сайту так і зі сторони користувача;

- керування базою даних;

## 5.2. Основні технічні вимоги до розробки.

### 5.2.1. Вимоги до програмної платформи:

- WINDOWS 10;
- Microsoft sql server;

### 5.2.2. Умови експлуатації системи:

- Доступність 24/7;
- Максимальний час обробки запиту – 3 секунди;
- Максимальна кількість одночасних запитів – 10 тис.

## 6. Стадії та етапи розробки.

### 6.1 Пояснювальна записка:

1. Розділ 1 «Аналіз предметної області обслуговування пасажирів» «10» жовтня 2023 р.
2. Розділ 2 «Проектування автоматизованої системи автовокзалу» «24» жовтня 2023 р.
3. Розділ 3 «Розробка автоматизованої системи автовокзалу» «22» листопада 2023 р.

### 6.2 Графічні матеріали:

1. Розробка UML-діаграм системи «14» жовтня 2023 р.
2. Розробка моделі бази даних системи «30» жовтня 2023 р.
3. Тестування програмного забезпечення «20» листопада 2023 р.

## 7. Порядок контролю і приймання.

- 7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «1» листопада 2023 р.
- 7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «21» листопада 2023 р.
- 7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «15» листопада 2023 р.

**Додаток Б**  
(обов'язковий)

**ІЛЮСТРАТИВНА ЧАСТИНА**

Розробка автоматизованої системи автовокзалу

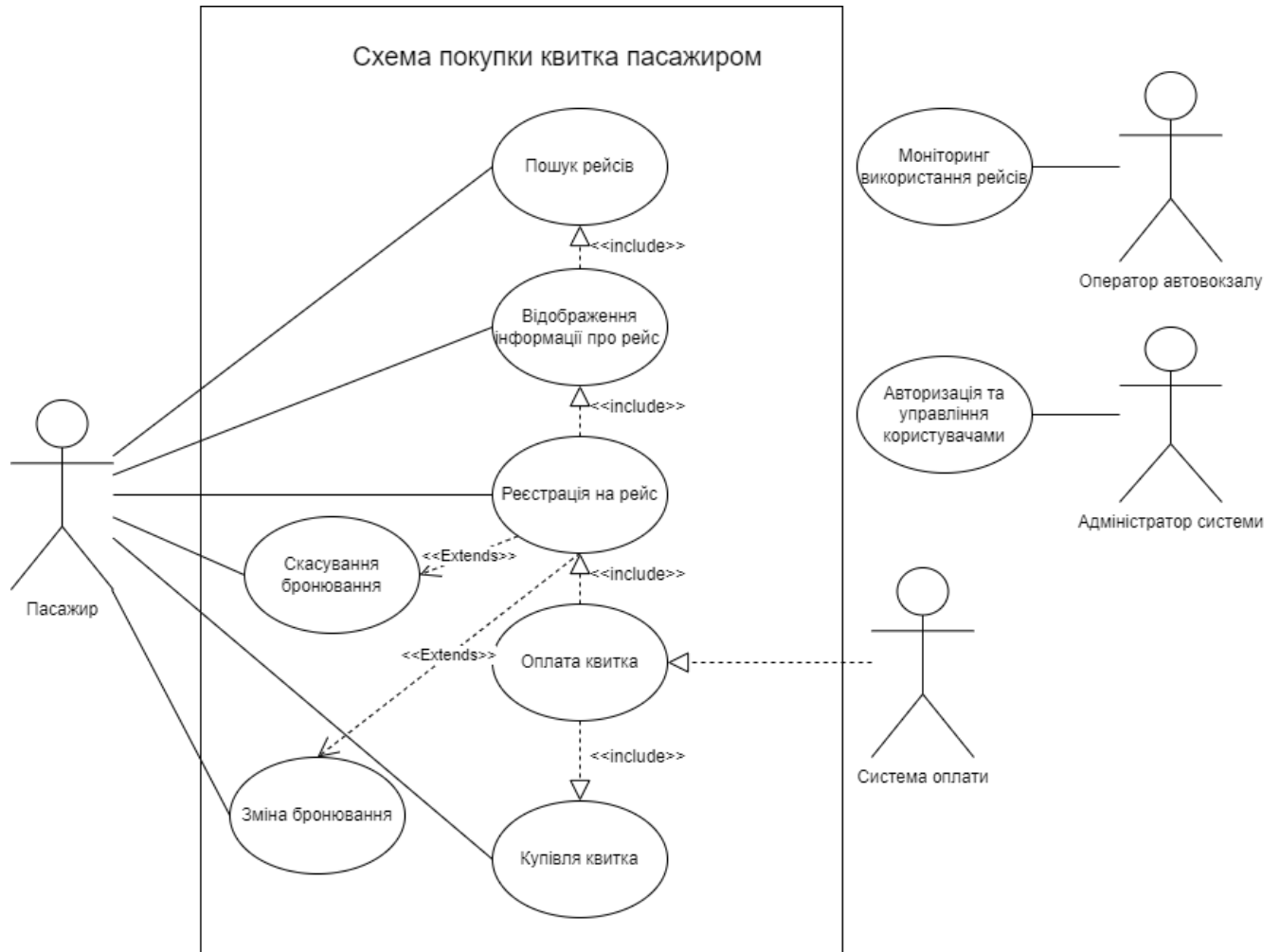


Рисунок Б.1 – UML- діаграма варіантів використання

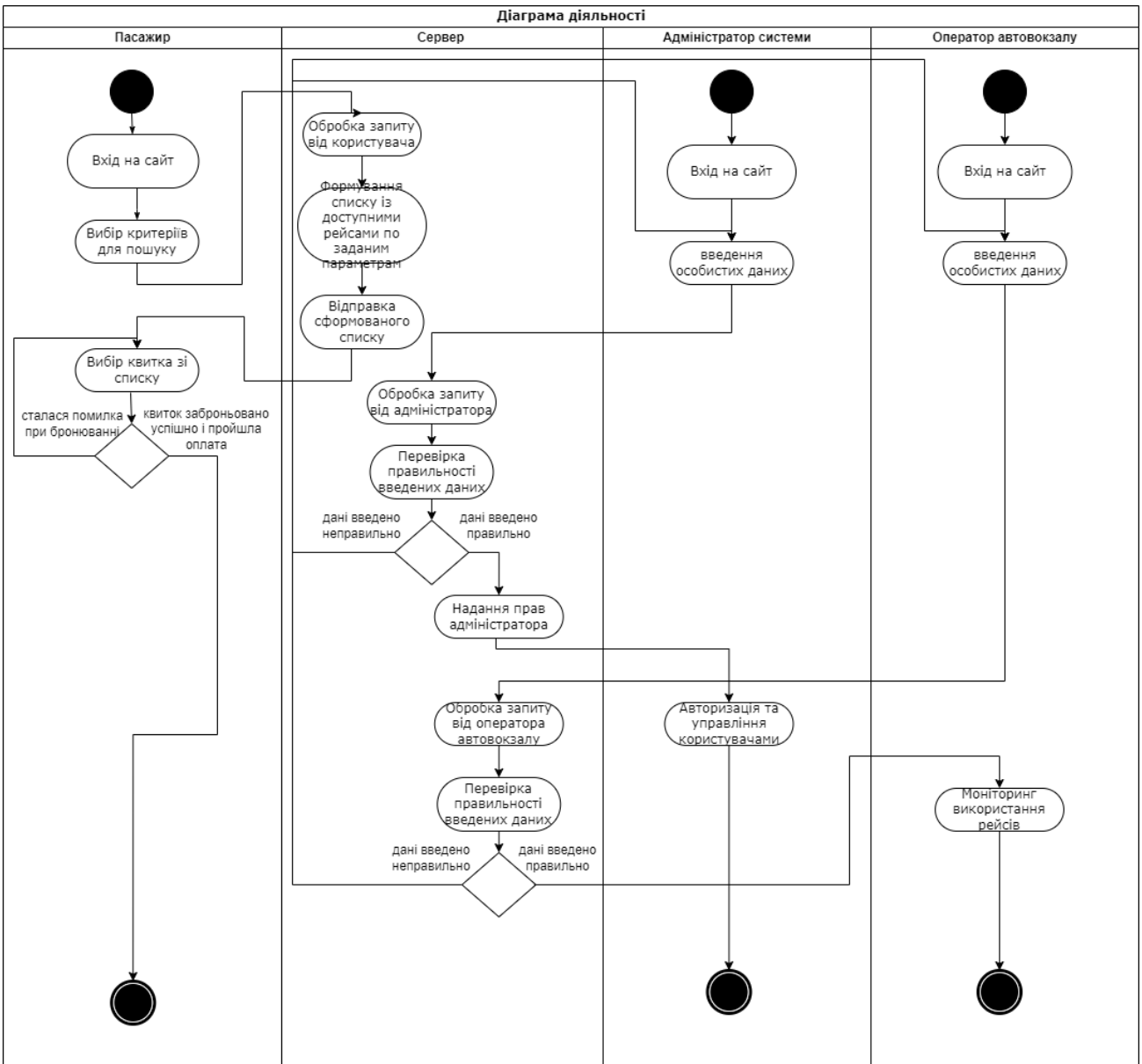


Рисунок Б.2 – UML- діаграма діяльності

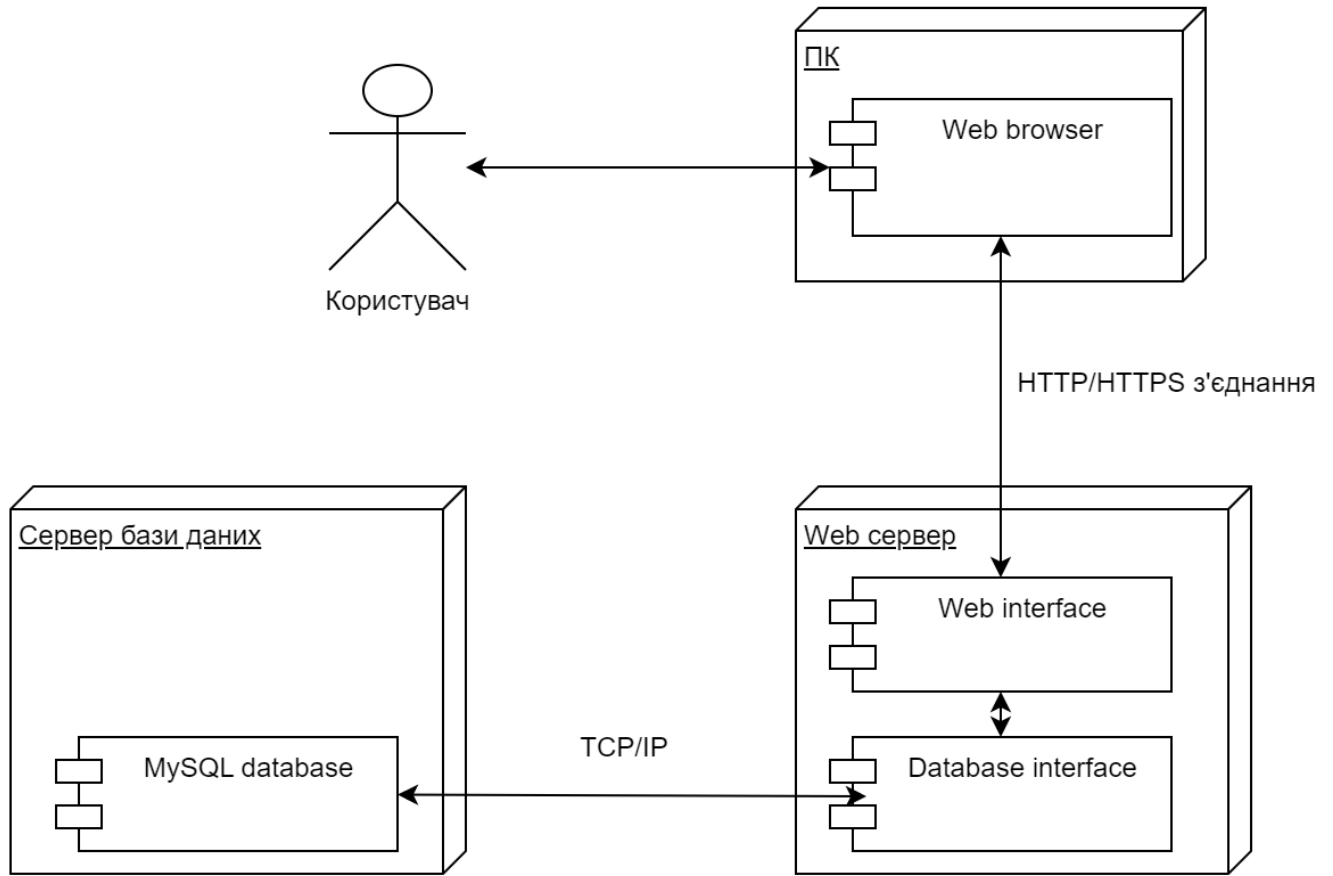


Рисунок Б.3 – UML- діаграма розгортання

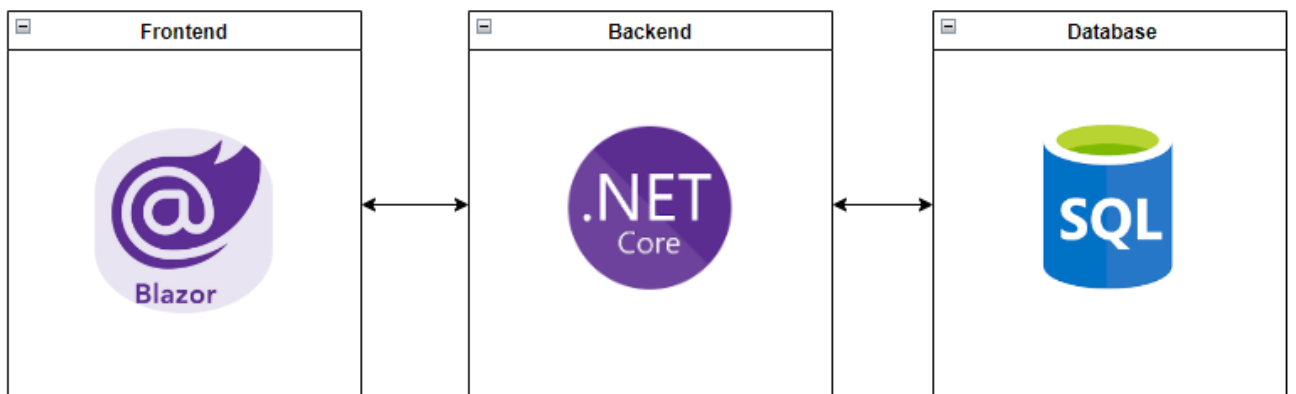


Рисунок Б.4 – Архітектура системи



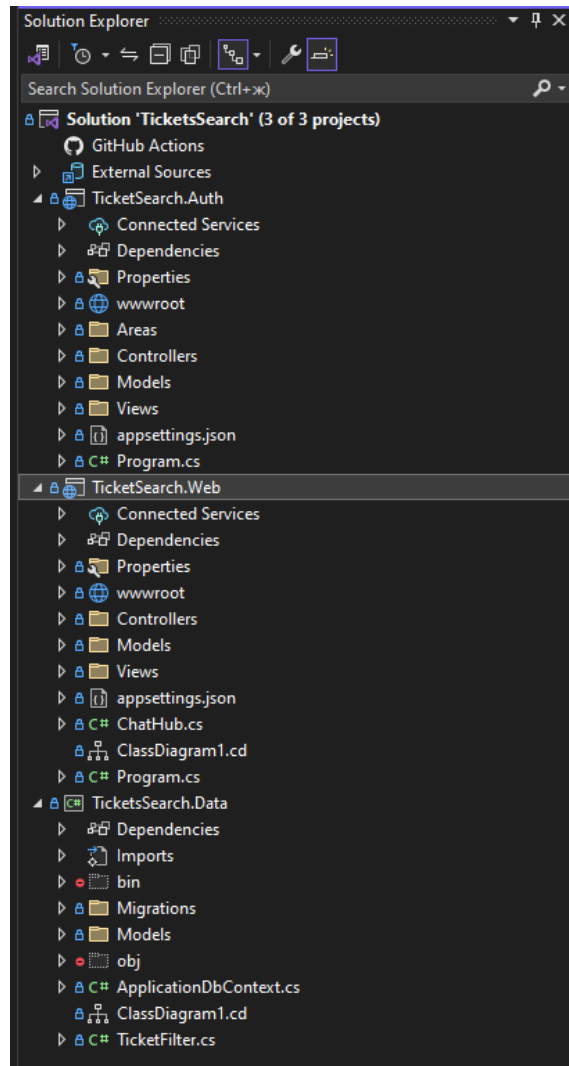


Рисунок Б.5 – Архітектура системи в Visual Studio

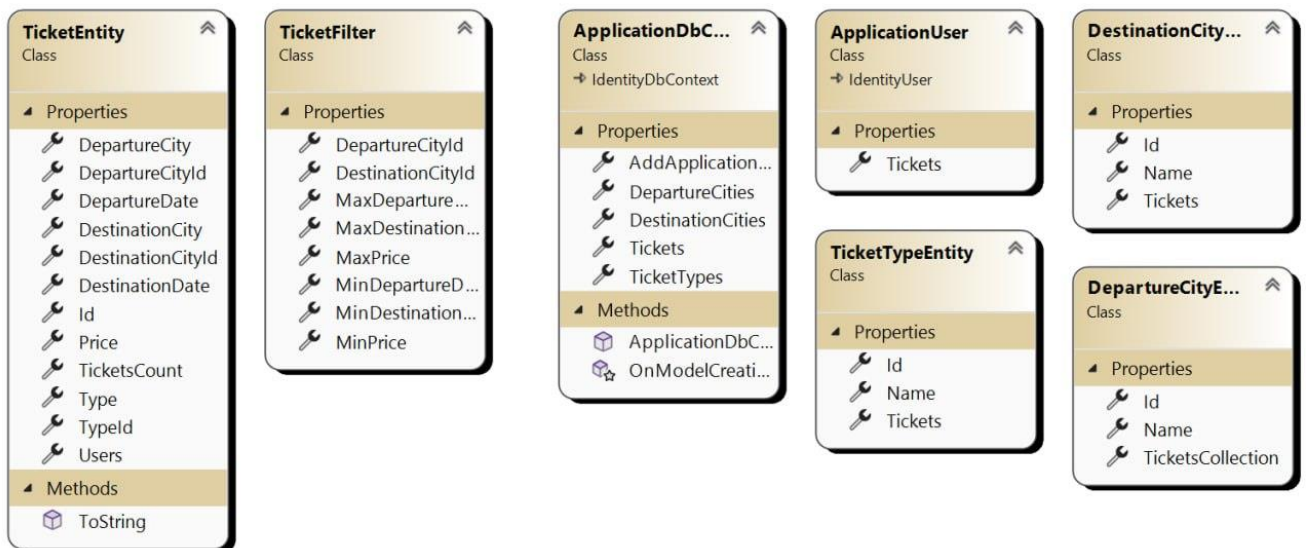


Рисунок Б.6 - UML- діаграма класів

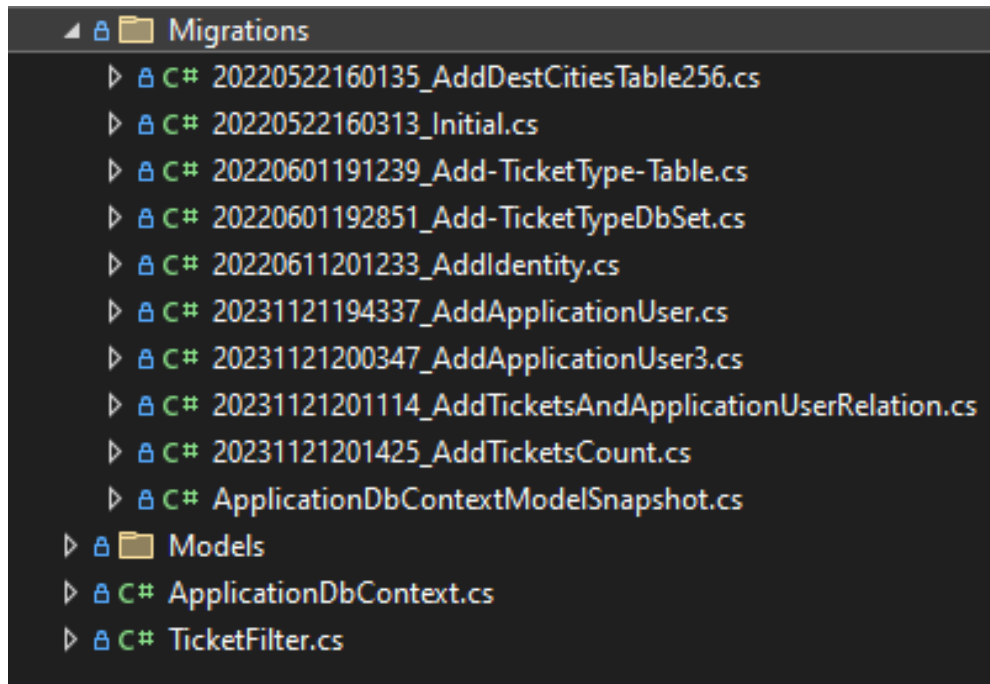


Рисунок Б.7 – Опис структури бази даних

Домашня сторінка [ДОМАШНЯ СТОРІНКА](#) [МОЇ КВИТКИ](#) [ТАБЛО КВИТКІВ](#) [ПРИВІТ DIMADIDYK2017@GMAIL.COM!!](#) [ВІЙТИ](#)

[ПОКАЗАТИ ФІЛЬТР](#)

**Фільтрування квитків**

Мінімальна вартість квитка

Максимальна вартість квитка

Мінімальна дата відправлення

Максимальна дата відправлення

Мінімальна дата прибуття

Максимальна дата прибуття

Місто відправлення

Місто прибуття

[ЗАСТОСУВАТИ ФІЛЬТР](#) [ОЧИСТИТИ ФІЛЬТР](#)

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття	
Маршрутне таксі	Одеса	Донецьк	659,00	78	22.09.2023 10:20:00	23.09.2023 4:20:00	<a href="#">ДЕТАЛІ</a>   <a href="#">ЗАБРОНІЮВАТИ</a>

Рисунок Б.8 – Функція фільтру квитків

Тип	Місце відправлення	Місце прибуття	Вартість	Дата відправлення	Дата прибуття		
Маршрутне таксі	Одеса	Донецьк	450,00	22.09.2023 15:10:00	23.09.2023 19:10:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ВІДМІНИТИ БРОНЮВАННЯ</a>
Маршрутне таксі	Одеса	Донецьк	659,00	22.09.2023 10:20:00	23.09.2023 4:20:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ВІДМІНИТИ БРОНЮВАННЯ</a>
Маршрутне таксі	Одеса	Донецьк	675,00	22.11.2023 20:10:00	23.09.2023 0:10:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ВІДМІНИТИ БРОНЮВАННЯ</a>
Маршрутне таксі	Одеса	Донецьк	659,00	22.09.2023 10:20:00	23.09.2023 4:20:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ВІДМІНИТИ БРОНЮВАННЯ</a>

## Рисунок Б.9 – Бронювання квитків

[ПОКАЗАТИ ФІЛЬТР](#)

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття		
Маршрутне таксі	Одеса	Донецьк	659,00	76	22.09.2023 10:20:00	23.09.2023 4:20:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ЗАБРОНЮВАТИ</a>
Маршрутне таксі	Одеса	Донецьк	450,00	12	22.09.2023 15:10:00	23.09.2023 19:10:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ЗАБРОНЮВАТИ</a>
Маршрутне таксі	Одеса	Донецьк	675,00	46	22.11.2023 20:10:00	23.09.2023 0:10:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ЗАБРОНЮВАТИ</a>

## Рисунок Б.10 – Перевірка бронювання квитків у системі

Тип	Місце відправлення	Місце прибуття	Вартість	Дата відправлення	Дата прибуття		
Маршрутне таксі	Одеса	Донецьк	450,00	22.09.2023 15:10:00	23.09.2023 19:10:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ВІДМІНИТИ БРОНЮВАННЯ</a>

## Рисунок Б.11 – Відміна бронювання квитків

[ПОКАЗАТИ ФІЛЬТР](#)

Тип	Місце відправлення	Місце прибуття	Вартість	Кількість квитків	Дата відправлення	Дата прибуття		
Маршрутне таксі	Одеса	Донецьк	659,00	78	22.09.2023 10:20:00	23.09.2023 4:20:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ЗАБРОНЮВАТИ</a>
Маршрутне таксі	Одеса	Донецьк	450,00	12	22.09.2023 15:10:00	23.09.2023 19:10:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ЗАБРОНЮВАТИ</a>
Маршрутне таксі	Одеса	Донецьк	675,00	47	22.11.2023 20:10:00	23.09.2023 0:10:00	<a href="#">ДЕТАЛІ</a>	<a href="#">ЗАБРОНЮВАТИ</a>

## Рисунок Б.12 – Перевірка функції відміни бронювання квитків

[СТОРІНКА](#)
[МОЇ КВИТКИ](#)
[ТАБЛО КВИТКІВ](#)
[ЧАТ](#)
[ПРИВІТ DIMADIDYK2017@GMAIL.COM!!](#)
[ВИЙТИ](#)

**Користувач:** Привіт

**Система:** Вітаю, я ваш віртуальний помічник, хочете забронювати квиток? Доступні наступні команди: відправлення, прибуття, вартість. Наприклад введіть - відправлення Київ

**Користувач:** відправлення вінниця

**Система:** Доступні квитки із містом відправлення вінниця | Відправлення: Вінниця, прибуття: Жмеринка, дата:16.12.2023 2:50:06/17.12.2023 4:31:06, вартість: 120,00 | Відправлення: Вінниця, прибуття: Немирів, дата:03.12.2023 20:40:00/03.12.2023 22:10:00, вартість: 125,00 |

**Користувач:** прибуття донецьк

**Система:** Доступні квитки із містом прибуття донецьк | Відправлення: Одеса, прибуття: Донецьк, дата:22.09.2023 15:10:00/23.09.2023 19:10:00, вартість: 450,00 | Відправлення: Одеса, прибуття: Донецьк, дата:22.09.2023 10:20:00/23.09.2023 4:20:00, вартість: 659,00 | Відправлення: Одеса, прибуття: Донецьк, дата:22.11.2023 20:10:00/23.09.2023 0:10:00, вартість: 675,00 |

**Користувач:** вартість 125

**Система:** Доступні квитки із вартістю 125 | Відправлення: Вінниця, прибуття: Немирів, дата:03.12.2023 20:40:00/03.12.2023 22:10:00, вартість: 125,00 | Відправлення: Луцьк, прибуття: Тернопіль, дата:26.09.2023 20:40:00/27.09.2023 4:50:00, вартість: 125,00 |

**Користувач:** відправлення одеса прибуття донецьк

**Система:** Доступні квитки із містом відправлення одеса та містом прибуття донецьк: | Відправлення: Одеса, прибуття: Донецьк, дата:22.09.2023 15:10:00/23.09.2023 19:10:00, вартість: 450,00 | Відправлення: Одеса, прибуття: Донецьк, дата:22.09.2023 10:20:00/23.09.2023 4:20:00, вартість: 659,00 | Відправлення: Одеса, прибуття: Донецьк, дата:22.11.2023 20:10:00/23.09.2023 0:10:00, вартість: 675,00 |

**Користувач:** хочу знижку

**Система:** Ваше повідомлення не розпізнано. Доступні наступні команди: відправлення, прибуття, вартість. Наприклад введіть - відправлення Київ

## Б.13 – Перевірка роботи чат-боту

**Додаток В**  
(обов'язковий)  
**Лістинг програми**

Лістинг програми TicketController.cs для проекту TicketSearch.Auth

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using TicketsSearch.Data;
using TicketsSearch.Data.Models;

namespace TicketsSearchApp.Controllers
{
    [Authorize]

    public class TicketController : Controller
    {
        private readonly ApplicationDbContext _context;

        public TicketController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Ticket
        public async Task<IActionResult> Index()
        {
            var applicationDbContext = _context.Tickets.Include(t => t.DepartureCity).Include(t
=> t.DestinationCity).Include(t => t.Type);
            return View(await applicationDbContext.ToListAsync());
        }

        // GET: Ticket/Details/5
```

```

public async Task<IActionResult> Details(Guid? id)
{
    if (id == null || _context.Tickets == null)
    {
        return NotFound();
    }

    var ticketEntity = await _context.Tickets
        .Include(t => t.DepartureCity)
        .Include(t => t.DestinationCity)
        .Include(t => t.Type)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (ticketEntity == null)
    {
        return NotFound();
    }

    return View(ticketEntity);
}

// GET: Ticket/Create
public IActionResult Create()
{
    ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
"Name");
    ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
"Name");
    ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
"Name");
    return View();
}

// POST: Ticket/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult>
Create([Bind("Id,TypeId,DepartureCityId,DestinationCityId,Price,TicketsCount,DepartureDa
te,DestinationDate")] TicketEntity ticketEntity)
{

```

```

//if (ModelState.IsValid)
{
    ticketEntity.Id = Guid.NewGuid();
    _context.Add(ticketEntity);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
"Name", ticketEntity.DepartureCityId);
ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
"Name", ticketEntity.DestinationCityId);
ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
"Name", ticketEntity.TypeId);
return View(ticketEntity);
}

// GET: Ticket/Edit/5
public async Task<IActionResult> Edit(Guid? id)
{
    if (id == null || _context.Tickets == null)
    {
        return NotFound();
    }

    var ticketEntity = await _context.Tickets.FindAsync(id);
    if (ticketEntity == null)
    {
        return NotFound();
    }
    ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
"Name", ticketEntity.DepartureCityId);
    ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
"Name", ticketEntity.DestinationCityId);
    ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
"Name", ticketEntity.TypeId);
    return View(ticketEntity);
}

// POST: Ticket/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

```



```

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(Guid id,
[Bind("Id,TypeId,DepartureCityId,DestinationCityId,Price,TicketsCount,DepartureDate,Des
tinationDate")] TicketEntity ticketEntity)
{
    if (id != ticketEntity.Id)
    {
        return NotFound();
    }

    // if (ModelState.IsValid)
    {
        try
        {
            _context.Update(ticketEntity);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!TicketEntityExists(ticketEntity.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
"Name", ticketEntity.DepartureCityId);
    ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
"Name", ticketEntity.DestinationCityId);
    ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
"Name", ticketEntity.TypeId);
    return View(ticketEntity);
}

// GET: Ticket/Delete/5

```

```

public async Task<IActionResult> Delete(Guid? id)
{
    if (id == null || _context.Tickets == null)
    {
        return NotFound();
    }

    var ticketEntity = await _context.Tickets
        .Include(t => t.DepartureCity)
        .Include(t => t.DestinationCity)
        .Include(t => t.Type)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (ticketEntity == null)
    {
        return NotFound();
    }

    return View(ticketEntity);
}

// POST: Ticket/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(Guid id)
{
    if (_context.Tickets == null)
    {
        return Problem("Entity set 'ApplicationDbContext.Tickets' is null.");
    }
    var ticketEntity = await _context.Tickets.FindAsync(id);
    if (ticketEntity != null)
    {
        _context.Tickets.Remove(ticketEntity);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool TicketEntityExists(Guid id)
{

```

```

        return (_context.Tickets?.Any(e => e.Id == id)).GetValueOrDefault();
    }
}
}

```

Лістинг програми TicketSearchController.cs для проекту TicketSearch.Auth

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using TicketsSearch.Data;
using TicketsSearch.Data.Models;
using TicketsSearchApp.Models;

namespace TicketsSearchApp.Controllers
{
    public class TicketSearchController : Controller
    {
        private readonly ApplicationDbContext _context;

        public TicketSearchController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: TicketSearch
        public async Task<IActionResult> Index(IEnumerable<TicketEntity> tickets = null)
        {
            if (tickets.Any())
            {
                return View(tickets);
            }

            var applicationDbContext = _context.Tickets.Include(t => t.DepartureCity).Include(t
=> t.DestinationCity).Include(t => t.Type);

```

```

        return View(await applicationDbContext.ToListAsync());
    }
    // GET: TicketSearch
    [HttpGet]
    public async Task<IActionResult> Search()
    {
        ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
"Name");
        ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
"Name");
        ViewData["TypeId"] = new SelectList(_context.TicketTypes, "Id", "Name");
        return View();
    }
    // GET: TicketSearch
    [HttpPost]
    public async Task<IActionResult>
Search([Bind("TypeId,DepartureCityId,DestinationCityId,DepartureDate,DestinationDate,De
partureDate,DestinationDate")] FilterModel filter)
    {
        var applicationDbContext = _context.Tickets
            .Include(t => t.DepartureCity)
            .Where(t => t.DepartureCityId == filter.DepartureCityId)

            .Include(t => t.DestinationCity)
            .Where(t => t.DestinationCityId == filter.DestinationCityId)

            .Include(t => t.Type)
            .Where(t => t.TypeId == filter.TypeId)

            .Where(t => t.DepartureDate.Date == filter.DepartureDate.Date)
            .Where(t => t.DestinationDate.Date == filter.DestinationDate.Date);

        return View(nameof(Index), await applicationDbContext.ToListAsync());
    }

    // GET: TicketSearch/Details/5
    public async Task<IActionResult> Details(Guid? id)
    {
        if (id == null || _context.Tickets == null)

```

```

    {
        return NotFound();
    }

    var ticketEntity = await _context.Tickets
        .Include(t => t.DepartureCity)
        .Include(t => t.DestinationCity)
        .Include(t => t.Type)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (ticketEntity == null)
    {
        return NotFound();
    }

    return View(ticketEntity);
}
}
}

```

Лістинг програми TicketSearchController.cs для проекту TicketSearch.Web

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using TicketsSearch.Data;
using TicketsSearch.Data.Models;
using TicketsSearchApp.Models;

namespace TicketsSearchApp.Controllers
{
    public class TicketSearchController : Controller
    {
        private readonly ApplicationDbContext _context;
    }
}

```

```

public TicketSearchController(ApplicationDbContext context)
{
    _context = context;
}

// GET: TicketSearch
public async Task<IActionResult> Index(IEnumerable<TicketEntity> tickets = null)
{
    if (tickets.Any())
    {
        return View(tickets);
    }

    var applicationDbContext = _context.Tickets.Include(t => t.DepartureCity).Include(t
=> t.DestinationCity).Include(t => t.Type);
    return View(await applicationDbContext.ToListAsync());

}

// GET: TicketSearch
[HttpGet]
public async Task<IActionResult> Search()
{
    ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
"Name");
    ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
"Name");
    ViewData["TypeId"] = new SelectList(_context.TicketTypes, "Id", "Name");
    return View();
}

// GET: TicketSearch
[HttpPost]
public async Task<IActionResult>
Search([Bind("TypeId,DepartureCityId,DestinationCityId,DepartureDate,DestinationDate,De
partureDate,DestinationDate")] FilterModel filter)
{
    var applicationDbContext = _context.Tickets
        .Include(t => t.DepartureCity)
        .Where(t => t.DepartureCityId == filter.DepartureCityId)

        .Include(t => t.DestinationCity)

```

```

        .Where( t=> t.DestinationCityId == filter.DestinationCityId)

        .Include(t => t.Type)
        .Where(t=> t.TypeId == filter.TypeId)

        .Where(t => t.DepartureDate.Date == filter.DepartureDate.Date)
        .Where(t => t.DestinationDate.Date == filter.DestinationDate.Date);

        return View(nameof(Index), await applicationDbContext.ToListAsync());
    }

    // GET: TicketSearch/Details/5
    public async Task<IActionResult> Details(Guid? id)
    {
        if (id == null || _context.Tickets == null)
        {
            return NotFound();
        }

        var ticketEntity = await _context.Tickets
            .Include(t => t.DepartureCity)
            .Include(t => t.DestinationCity)
            .Include(t => t.Type)
            .FirstOrDefaultAsync(m => m.Id == id);
        if (ticketEntity == null)
        {
            return NotFound();
        }

        return View(ticketEntity);
    }
}
}
}

```

Лістинг програми TicketBoardController.cs для проекту TicketSearch.Auth

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Net.Sockets;

```

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using TicketsSearch.Data;
using TicketsSearch.Data.Models;

namespace TicketsSearchApp.Controllers
{
    [Authorize]

    public class TicketBoardController : Controller
    {
        private readonly ApplicationDbContext _context;

        public TicketBoardController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Ticket
        public async Task<IActionResult> Index()
        {
            var applicationDbContext = _context.Tickets
                .Include(t => t.DepartureCity)
                .Include(t => t.DestinationCity)
                .Include(t => t.Users)
                .Include(t => t.Type);

            var tickets = await applicationDbContext.ToListAsync();

            tickets.ForEach(x =>
            {
                x.TicketsCount -= x.Users?.Count() ?? 0;
            });
            var departureCities = tickets.Select(x => x.DepartureCity);
            var destinationCities = tickets.Select(x => x.DestinationCity);
        }
    }
}
```



```

ViewData["DepartureCityId"] = new SelectList(departureCities, "Id", "Name");
ViewData["DestinationCityId"] = new SelectList(destinationCities, "Id", "Name");

tickets.Sort((x, y) => x.DepartureDate.CompareTo(y.DepartureDate));
return View(tickets);
}

// GET: Ticket
[HttpPost]
public async Task<IActionResult> Index(TicketFilter filter = null)
{
    var applicationDbContext = _context.Tickets
        .Include(t => t.DepartureCity)
        .Include(t => t.DestinationCity)
        .Include(t => t.Users)
        .Include(t => t.Type);

    ViewBag.Filter = filter;

    var tickets = await applicationDbContext.ToListAsync();

    tickets.ForEach(x =>
    {
        x.TicketsCount -= x.Users?.Count() ?? 0;
    });
    //filtering
    if (filter != null)
    {
        tickets = GetFilteredTickets(tickets, filter);
    }

    var departureCities = tickets.Select(x => x.DepartureCity);
    var destinationCities = tickets.Select(x => x.DestinationCity);

    ViewData["DepartureCityId"] = new SelectList(departureCities, "Id", "Name");
    ViewData["DestinationCityId"] = new SelectList(destinationCities, "Id", "Name");

    //TODO: possibly need to show only recent races

    tickets.Sort((x, y) => x.DepartureDate.CompareTo(y.DepartureDate));
    return View(tickets);
}

```

```

    }

    // GET: Ticket/Details/5
    public async Task<IActionResult> Details(Guid? id)
    {
        if (id == null || _context.Tickets == null)
        {
            return NotFound();
        }

        var ticketEntity = await _context.Tickets
            .Include(t => t.DepartureCity)
            .Include(t => t.DestinationCity)
            .Include(t => t.Type)
            .FirstOrDefaultAsync(m => m.Id == id);
        if (ticketEntity == null)
        {
            return NotFound();
        }

        return View(ticketEntity);
    }

    // GET: Ticket/Create
    public IActionResult Create()
    {
        ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
"Name");
        ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
"Name");
        ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
"Name");
        return View();
    }

    // POST: Ticket/Create
    // To protect from overposting attacks, enable the specific properties you want to bind to.
    // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]

```

```

    public async Task<IActionResult>
Create([Bind("Id,TypeId,DepartureCityId,DestinationCityId,Price,TicketsCount,DepartureDate,DestinationDate")] TicketEntity ticketEntity)
    {
        //if (ModelState.IsValid)
        {
            ticketEntity.Id = Guid.NewGuid();
            _context.Add(ticketEntity);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
"Name", ticketEntity.DepartureCityId);
        ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
"Name", ticketEntity.DestinationCityId);
        ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
"Name", ticketEntity.TypeId);
        return View(ticketEntity);
    }

// GET: Ticket/Edit/5
public async Task<IActionResult> Edit(Guid? id)
    {
        if (id == null || _context.Tickets == null)
        {
            return NotFound();
        }

        var ticketEntity = await _context.Tickets.FindAsync(id);
        if (ticketEntity == null)
        {
            return NotFound();
        }
        ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
"Name", ticketEntity.DepartureCityId);
        ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
"Name", ticketEntity.DestinationCityId);
        ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
"Name", ticketEntity.TypeId);
        return View(ticketEntity);
    }

```

```

// POST: Ticket/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(Guid id,
[Bind("Id,TypeId,DepartureCityId,DestinationCityId,Price,TicketsCount,DepartureDate,DestinationDate")] TicketEntity ticketEntity)
{
    if (id != ticketEntity.Id)
    {
        return NotFound();
    }

    // if (ModelState.IsValid)
    {
        try
        {
            _context.Update(ticketEntity);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!TicketEntityExists(ticketEntity.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
    "Name", ticketEntity.DepartureCityId);
    ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
    "Name", ticketEntity.DestinationCityId);
    ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
    "Name", ticketEntity.TypeId);

```

```

    return View(ticketEntity);
}

// GET: Ticket/Delete/5
public async Task<IActionResult> Delete(Guid? id)
{
    if (id == null || _context.Tickets == null)
    {
        return NotFound();
    }

    var ticketEntity = await _context.Tickets
        .Include(t => t.DepartureCity)
        .Include(t => t.DestinationCity)
        .Include(t => t.Type)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (ticketEntity == null)
    {
        return NotFound();
    }

    return View(ticketEntity);
}

[HttpGet]
public async Task<IActionResult> Import()
{
    return View();
}

[HttpPost, ActionName("Import")]
public async Task<IActionResult> ImportConfirmed()
{
    return View();
}

// POST: Ticket/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(Guid id)

```

```

{
    if (_context.Tickets == null)
    {
        return Problem("Entity set 'ApplicationDbContext.Tickets' is null.");
    }
    var ticketEntity = await _context.Tickets.FindAsync(id);
    if (ticketEntity != null)
    {
        _context.Tickets.Remove(ticketEntity);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool TicketEntityExists(Guid id)
{
    return (_context.Tickets?.Any(e => e.Id == id)).GetValueOrDefault();
}

private List<TicketEntity> GetFilteredTickets(List<TicketEntity> allTickets,
TicketFilter filter)
{
    var filteredTickets = allTickets;

    if (filter.DepartureCityId.HasValue)
    {
        filteredTickets = filteredTickets.Where(ticket => ticket.DepartureCityId ==
filter.DepartureCityId).ToList();
    }

    if (filter.DestinationCityId.HasValue)
    {
        filteredTickets = filteredTickets.Where(ticket => ticket.DestinationCityId ==
filter.DestinationCityId).ToList();
    }

    if (filter.MinPrice > 0)
    {
        filteredTickets = filteredTickets.Where(ticket => filter.MinPrice <=
ticket.Price).ToList();
    }
}

```

```

        if (filter.MaxPrice >0)
        {
            filteredTickets = filteredTickets.Where(ticket => ticket.Price <=
filter.MaxPrice).ToList();
        }
        if (filter.MinDepartureDate.HasValue)
        {
            filteredTickets = filteredTickets.Where(ticket => ticket.DepartureDate >=
filter.MinDepartureDate).ToList();
        }

        if (filter.MaxDepartureDate.HasValue)
        {
            filteredTickets = filteredTickets.Where(ticket => ticket.DepartureDate <=
filter.MaxDepartureDate).ToList();
        }
        if (filter.MinDestinationDate.HasValue)
        {
            filteredTickets = filteredTickets.Where(ticket => ticket.DestinationDate >=
filter.MinDestinationDate).ToList();
        }
        if (filter.MaxDestinationDate.HasValue)
        {
            filteredTickets = filteredTickets.Where(ticket => ticket.DestinationDate <=
filter.MaxDestinationDate).ToList();
        }

        return filteredTickets;
    }
}
}

```

Лістинг програми TicketBoardController.cs для проекту TicketSearch.Web

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Net.Sockets;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;

```

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using TicketsSearch.Data;
using TicketsSearch.Data.Models;

namespace TicketsSearchApp.Controllers
{
    [Authorize]

    public class TicketBoardController : Controller
    {
        private readonly ApplicationDbContext _context;

        public TicketBoardController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Ticket
        public async Task<IActionResult> Index()
        {
            var applicationDbContext = _context.Tickets
                .Include(t => t.DepartureCity)
                .Include(t => t.DestinationCity)
                .Include(t => t.Users)
                .Include(t => t.Type);

            var tickets = await applicationDbContext.ToListAsync();

            tickets.ForEach(x =>
            {
                x.TicketsCount -= x.Users?.Count() ?? 0;
            });
            var departureCities = tickets.Select(x => x.DepartureCity);
            var destinationCities = tickets.Select(x => x.DestinationCity);

            ViewData["DepartureCityId"] = new SelectList(departureCities, "Id", "Name");
            ViewData["DestinationCityId"] = new SelectList(destinationCities, "Id", "Name");
        }
    }
}

```



```

tickets.Sort((x, y) => x.DepartureDate.CompareTo(y.DepartureDate));
return View(tickets);
}

// GET: Ticket
[HttpPost]
public async Task<IActionResult> Index(TicketFilter filter = null)
{
    var applicationDbContext = _context.Tickets
        .Include(t => t.DepartureCity)
        .Include(t => t.DestinationCity)
        .Include(t => t.Users)
        .Include(t => t.Type);

    ViewBag.Filter = filter;

    var tickets = await applicationDbContext.ToListAsync();

    tickets.ForEach(x =>
    {
        x.TicketsCount -= x.Users?.Count() ?? 0;
    });
    //filtering
    if (filter != null)
    {
        tickets = GetFilteredTickets(tickets, filter);
    }

    var departureCities = tickets.Select(x => x.DepartureCity);
    var destinationCities = tickets.Select(x => x.DestinationCity);

    ViewData["DepartureCityId"] = new SelectList(departureCities, "Id", "Name");
    ViewData["DestinationCityId"] = new SelectList(destinationCities, "Id", "Name");

    //TODO: possibly need to show only recent races

    tickets.Sort((x, y) => x.DepartureDate.CompareTo(y.DepartureDate));
    return View(tickets);
}

```

```

// GET: Ticket/Details/5
public async Task<IActionResult> Details(Guid? id)
{
    if (id == null || _context.Tickets == null)
    {
        return NotFound();
    }

    var ticketEntity = await _context.Tickets
        .Include(t => t.DepartureCity)
        .Include(t => t.DestinationCity)
        .Include(t => t.Type)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (ticketEntity == null)
    {
        return NotFound();
    }

    return View(ticketEntity);
}

// GET: Ticket/Create
public IActionResult Create()
{
    ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
"Name");
    ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
"Name");
    ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
"Name");
    return View();
}

// POST: Ticket/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult>
Create([Bind("Id,TypeId,DepartureCityId,DestinationCityId,Price,TicketsCount,DepartureDa
te,DestinationDate")] TicketEntity ticketEntity)

```

```

{
    //if (ModelState.IsValid)
    {
        ticketEntity.Id = Guid.NewGuid();
        _context.Add(ticketEntity);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
    "Name", ticketEntity.DepartureCityId);
    ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
    "Name", ticketEntity.DestinationCityId);
    ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
    "Name", ticketEntity.TypeId);
    return View(ticketEntity);
}

// GET: Ticket/Edit/5
public async Task<IActionResult> Edit(Guid? id)
{
    if (id == null || _context.Tickets == null)
    {
        return NotFound();
    }

    var ticketEntity = await _context.Tickets.FindAsync(id);
    if (ticketEntity == null)
    {
        return NotFound();
    }
    ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
    "Name", ticketEntity.DepartureCityId);
    ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
    "Name", ticketEntity.DestinationCityId);
    ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
    "Name", ticketEntity.TypeId);
    return View(ticketEntity);
}

// POST: Ticket/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.

```

```

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(Guid id,
[Bind("Id,TypeId,DepartureCityId,DestinationCityId,Price,TicketsCount,DepartureDate,DestinationDate")] TicketEntity ticketEntity)
{
    if (id != ticketEntity.Id)
    {
        return NotFound();
    }

    // if (ModelState.IsValid)
    {
        try
        {
            _context.Update(ticketEntity);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!TicketEntityExists(ticketEntity.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["DepartureCityId"] = new SelectList(_context.DepartureCities, "Id",
    "Name", ticketEntity.DepartureCityId);
    ViewData["DestinationCityId"] = new SelectList(_context.DestinationCities, "Id",
    "Name", ticketEntity.DestinationCityId);
    ViewData["TypeId"] = new SelectList(_context.Set<TicketTypeEntity>(), "Id",
    "Name", ticketEntity.TypeId);
    return View(ticketEntity);
}

```

```

// GET: Ticket/Delete/5
public async Task<IActionResult> Delete(Guid? id)
{
    if (id == null || _context.Tickets == null)
    {
        return NotFound();
    }

    var ticketEntity = await _context.Tickets
        .Include(t => t.DepartureCity)
        .Include(t => t.DestinationCity)
        .Include(t => t.Type)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (ticketEntity == null)
    {
        return NotFound();
    }

    return View(ticketEntity);
}

// POST: Ticket/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(Guid id)
{
    if (_context.Tickets == null)
    {
        return Problem("Entity set 'ApplicationDbContext.Tickets' is null.");
    }
    var ticketEntity = await _context.Tickets.FindAsync(id);
    if (ticketEntity != null)
    {
        _context.Tickets.Remove(ticketEntity);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool TicketEntityExists(Guid id)

```

```
{
    return (_context.Tickets?.Any(e => e.Id == id)).GetValueOrDefault();
}
private List<TicketEntity> GetFilteredTickets(List<TicketEntity> allTickets,
TicketFilter filter)
{
    var filteredTickets = allTickets;

    if (filter.DepartureCityId.HasValue)
    {
        filteredTickets = filteredTickets.Where(ticket => ticket.DepartureCityId ==
filter.DepartureCityId).ToList();
    }

    if (filter.DestinationCityId.HasValue)
    {
        filteredTickets = filteredTickets.Where(ticket => ticket.DestinationCityId ==
filter.DestinationCityId).ToList();
    }

    if (filter.MinPrice > 0)
    {
        filteredTickets = filteredTickets.Where(ticket => filter.MinPrice <=
ticket.Price).ToList();
    }
    if (filter.MaxPrice >0)
    {
        filteredTickets = filteredTickets.Where(ticket => ticket.Price <=
filter.MaxPrice).ToList();
    }
    if (filter.MinDepartureDate.HasValue)
    {
        filteredTickets = filteredTickets.Where(ticket => ticket.DepartureDate >=
filter.MinDepartureDate).ToList();
    }

    if (filter.MaxDepartureDate.HasValue)
    {
        filteredTickets = filteredTickets.Where(ticket => ticket.DepartureDate <=
filter.MaxDepartureDate).ToList();
    }
}
```

```

        if (filter.MinDestinationDate.HasValue)
        {
            filteredTickets = filteredTickets.Where(ticket => ticket.DestinationDate >=
filter.MinDestinationDate).ToList();
        }
        if (filter.MaxDestinationDate.HasValue)
        {
            filteredTickets = filteredTickets.Where(ticket => ticket.DestinationDate <=
filter.MaxDestinationDate).ToList();
        }

        return filteredTickets;
    }
}
}

```

Лістинг програми `site.css` яка містить загальні стилі для проекту `TicketSearch.Auth` (більшість з яких аналогічні і для проекту `TicketSearch.Web`)

```

html {
    font-size: 14px;
}

body {
    color: #726f77;
}

a {
    text-decoration: none;
}

a:hover {
    color: Navy;
}

header {
    background: #2b2e48;
    padding: 10px;
    box-shadow: 0 3px 3px rgba(0, 0, 0, 0.05);
}

```

```
}
```

```
.navbar-light .navbar-brand {
  color: #ee4d4d;
  float: left;
  font-family: "Dosis", arial, tahoma, verdana;
  font-size: 12px;
  font-weight: 400;
}
```

```
nav.navbar.navbar-expand-sm.navbar-togglerable-sm.navbar-light.bg-white.border-
bottom.box-shadow.mb-3 {
  background-color: #2b2e48 !important;
  margin-bottom: 0px !important;
  border-bottom: 1px solid #2b2e48 !important;
}
```

```
div.container-fluid {
  background: #2b2e48;
}
```

```
a.nav-link.text-dark {
  padding: 5px 15px;
  background: transparent;
  border: 2px solid #f27c7c;
  color: #f27c7c !important;
  position: relative;
  text-transform: uppercase;
  border-radius: 5px;
  transition: background 0.3s ease;
  box-shadow: 2px 2px 0 #f27c7c;
  font-family: "Dosis";
  font-size: 18px;
  margin: 10px 5px;
}
```

```
button.nav-link.btn.btn-link.text-dark {
  padding: 5px 15px;
  background: transparent;
  border: 2px solid #f27c7c;
  color: #f27c7c !important;
```



```
    position: relative;
    text-transform: uppercase;
    border-radius: 5px;
    transition: background 0.3s ease;
    box-shadow: 2px 2px 0 #f27c7c;
    font-family: "Dosis";
    font-size: 18px;
    margin: 10px 5px;
}

a.nav-link.text-dark:hover {
    box-shadow: none;
    top: 2px;
    left: 2px;
    box-shadow: 2px 2px 0 transparent;
}

button.nav-link.btn.btn-link.text-dark:hover {
    box-shadow: none;
    top: 2px;
    left: 2px;
    box-shadow: 2px 2px 0 transparent;
}

h1 {
    text-align: center;
    padding: 10px 0;
    font-family: "Dosis", arial, tahoma, verdana;
    font-weight: 500;
    color: #726f77;
}

h4 {
    margin-top: 15px;
}

#searchButton, button.btn.btn-primary, a.btn.btn-default {
    margin-top: 10px;
    margin-bottom: 10px;
    padding: 10px 25px;
    background: transparent;
```

```

border: 2px solid #f27c7c;
color: #f27c7c;
display: inline-block;
position: relative;
text-transform: uppercase;
font-size: 16px;
border-radius: 5px;
transition: background 0.3s ease;
box-shadow: 2px 2px 0 #f27c7c
}

#searchButton:hover, button.btn.btn-primary:hover, a.btn.btn-default:hover {
  box-shadow: none;
  top: 2px;
  left: 2px;
  box-shadow: 2px 2px 0 transparent;
}

thead {
  background-color: #ee4d4d !important;
}

th {
  padding: 15px;
  color: #fff;
}

tbody {
  box-shadow: 0 3px 0 rgba(0, 0, 0, 0.1);
}

a[href*="TicketBoard"], a[href*="BookTicket"], a[href*="TicketType"],
a[href*="DestinationCity"], a[href*="DepartureCity"], a[href*="TicketSearch"],
a[href*="Edit"], a[href*="Details"], a[href*="Delete"], a[href*="ForgotPassword"],
a[href*="ResendEmail"] {
  padding: 4px 10px;
  background: transparent;
  border: 2px solid #f27c7c;
  color: #f27c7c;
  display: inline-block;

```

```

position: relative;
text-transform: uppercase;
font-size: 16px;
border-radius: 5px;
transition: background 0.3s ease;
box-shadow: 2px 2px 0 #f27c7c
}

a[href*="TicketBoard"]:hover, a[href*="BookTicket"]:hover, a[href*="TicketType"]:hover,
a[href*="DestinationCity"]:hover, a[href*="DepartureCity"]:hover,
a[href*="TicketSearch"]:hover, a[href*="Edit"]:hover, a[href*="Details"]:hover,
a[href*="Delete"]:hover, a[href*="ForgotPassword"]:hover, a[href*="ResendEmail"]:hover {
    box-shadow: none;
    top: 2px;
    left: 2px;
    box-shadow: 2px 2px 0 transparent;
}

input.btn.btn-primary {
    padding: 4px 10px;
    background: transparent;
    border: 2px solid #f27c7c;
    color: #f27c7c;
    display: inline-block;
    position: relative;
    text-transform: uppercase;
    font-size: 16px;
    border-radius: 5px;
    transition: background 0.3s ease;
    box-shadow: 2px 2px 0 #f27c7c;
    margin-top: 15px;
    margin-bottom: 15px;
}

input.btn.btn-primary:hover {
    box-shadow: none;
    top: 2px;
    left: 2px;
    box-shadow: 2px 2px 0 transparent;
}

```

```
td {
  color: #726f77;
  font-family: helvetica, arial, tahoma, verdana;
  font-size: 18px;
}
```

```
.footer[b-2oyr0ceqo0] {
  position: absolute;
  bottom: 0;
  width: 100%;
  white-space: nowrap;
  line-height: 60px;
  background: #2b2e48;
}
```

```
#Search_button, #Back_to_all_tickets_button, #Create_New_button {
  margin-top: 10px;
  margin-bottom: 10px;
  padding: 10px 25px;
  background: transparent;
  border: 2px solid #f27c7c;
  color: #f27c7c;
  display: inline-block;
  position: relative;
  text-transform: uppercase;
  font-size: 16px;
  border-radius: 5px;
  transition: background 0.3s ease;
  box-shadow: 2px 2px 0 #f27c7c
}
```

```
#Search_button:hover, #Back_to_all_tickets_button:hover, #Create_New_button:hover {
  box-shadow: none;
  top: 2px;
  left: 2px;
  box-shadow: 2px 2px 0 transparent;
}
```

```
@media (min-width: 768px) {
  html {
```

```
    font-size: 16px;  
  }  
}
```

```
html {  
  position: relative;  
  min-height: 100%;  
}
```

```
body {  
  margin-bottom: 60px;  
}
```

**Додаток Г**  
(обов'язковий)

**ПРОТОКОЛ  
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА  
НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ**

Назва роботи: «Розробка автоматизованої системи автовокзалу»

Тип роботи: Магістерська кваліфікаційна робота

Підрозділ АПТ, ФІТА

**Показники звіту подібності Unicheck**

Оригінальність 96.6% Схожість 3.4%

Аналіз звіту подібності (відмітити потрібне)

Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Роман МАСЛІЙ

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Дмитро ДІДИК

Керівник роботи Олег КОВАЛЮК