Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему «Розроблення програмного забезпечення для автоматизації роботи приміського вокзалу»

Виконав: студент групи ІПЗ19-2
Спеціальність 121 Інженерія програмного забезпечення
Полеводін Євгеній Петрович
(прізвище та ініціали)
Керівник к.е.н., доц. Яковенко Т. Ю.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
В.о. завідувача кафедри кібербезпеки та інформаційних технологій
(посада)
к.т.н., доц. Прокопович-Ткаченко Д.І.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2023

АНОТАЦІЯ

Полеводін Є.П. Розроблення програмного забезпечення для автоматизації роботи приміського вокзалу.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2023.

Початковим кроком у розробленні програмного забезпечення для автоматизації роботи приміського вокзалу є аналіз вимог та потреб. Це включає вивчення поточних процесів та визначення проблем, які можуть бути вирішені за допомогою автоматизації. Збір вимог відбувається шляхом спілкування зі зацікавленими сторонами, такими як керівники вокзалу, персонал та пасажири. Це допомагає зрозуміти їх потреби та впевнитися, що розробка програмного забезпечення відповідає цим потребам.

Після аналізу вимог розробляється архітектуру системи. Це включає визначення модулів, компонентів та їх взаємодії, а також визначення даних, які будуть використовуватися. Архітектура системи повинна бути гнучкою та масштабованою, щоб забезпечити зручне розширення та підтримку у майбутньому.

Після проектування архітектури системи розпочинається розробка функціональності. Це включає реалізацію основних функцій, таких як онлайн-продаж квитків, інформаційна система, електронні табло та інші. Розробка вимагає створення логіки програми, баз даних, інтерфейсу користувача та інших необхідних компонентів.

Після розробки функціональності необхідно забезпечити інтеграцію з існуючими системами, які вже використовуються на вокзалі.

Ключові слова: бази даних, інтерфейс користувача, впровадження, обладнання, навчання персоналу, підтримка, технічна допомога, мова програмування C#, приміський вокзал, Windows Forms, SQL.

ABSTRACT

Polevodin Ye.P. Development of software for automation of the suburban station.

Qualification work for a bachelor's degree in specialty 121 «Software Engineering». – University of Customs and Finance, Dnipro, 2023.

The initial step in developing software to automate the work of a suburban station is to analyze the requirements and needs. This includes studying current processes and identifying problems that can be solved through automation. Requirements are collected by communicating with stakeholders such as station managers, staff, and passengers. This helps to understand their needs and ensure that the software development meets those needs.

After analyzing the requirements, the system architecture is developed. This includes defining modules, components and their interactions, as well as defining the data that will be used. The system architecture should be flexible and scalable to allow for easy expansion and support in the future.

After designing the system architecture, the development of functionality begins. This includes the implementation of basic functions such as online ticketing, information system, electronic scoreboards, and others. Development requires the creation of program logic, databases, user interface, and other necessary components.

Once the functionality is developed, it is necessary to ensure integration with existing systems already in use at the station.

Keywords: databases, user interface, implementation, equipment, staff training, support, technical assistance, C# programming language, suburban station, Windows Forms, SQL.

ЗМІСТ

ВСТУП

У сучасному світі автоматизація стала необхідною складовою багатьох сфер діяльності, у тому числі і в транспортній індустрії. Однією з ключових галузей є автоматизація роботи приміських вокзалів, яка має на меті поліпшення якості обслуговування пасажирів, оптимізацію робочих процесів та підвищення ефективності роботи вокзалів загалом. Враховуючи ці фактори, розроблення програмного забезпечення для автоматизації роботи приміського вокзалу стає актуальним завданням.

Актуальність роботи з розроблення програмного забезпечення для автоматизації роботи приміського вокзалу є вкрай важливою і обґрунтованою. Ось декілька ключових аспектів, які підкреслюють актуальність цієї роботи:

1) підвищення ефективності. Автоматизація робочих процесів приміського вокзалу дозволяє зменшити ручну працю, усунути часові затримки та помилки, пов'язані з людським фактором. Це сприяє значному зростанню продуктивності та ефективності в роботі вокзалу, а також поліпшує якість обслуговування пасажирів;

2) покращення досвіду пасажирів. Розроблення програмного забезпечення дозволить пасажирам зручно та швидко здійснювати онлайн-покупку квитків, отримувати актуальну інформацію про розклади руху, затримки та зміни маршрутів. Це значно спростить їх планування подорожей та забезпечить більш комфортний досвід;

3) зменшення черг та часу очікування. Автоматизована система може забезпечити оптимальне розподілення пасажирів та керування потоками на вокзалі, що дозволить уникнути довгих черг та скоротити час очікування. Це забезпечить більше задоволення та задоволення пасажирів;

4) легкість управління та аналітика. Розроблення програмного забезпечення дозволить вокзалу легко керувати та контролювати різні аспекти його роботи. Відстеження руху поїздів, продаж квитків, обробка даних та

звітність все це може бути легко автоматизовано та керовано через централізовану систему;

5) розвиток сучасних технологій. Завдяки стрімкому розвитку інформаційних технологій, таких як штучний інтелект, Інтернет речей та хмарні обчислення, впровадження програмного забезпечення для автоматизації роботи вокзалу стає реальним. Використання цих технологій дозволяє створювати розширені функції та забезпечувати зручність та безпеку для користувачів.

Загалом, розроблення програмного забезпечення для автоматизації роботи приміського вокзалу є важливим кроком у напрямку покращення якості та ефективності вокзальних послуг, що відповідає потребам сучасного транспортного сектора та забезпечує комфорт та зручність для пасажирів.

Метою роботи є розроблення програмного забезпечення для автоматизації роботи приміського вокзалу.

Для досягнення поставленої мети розроблення програмного забезпечення для автоматизації роботи приміського вокзалу можуть ставитися наступні задачі:

1) аналіз вимог та потреб. Проведення детального аналізу потреб вокзалу та пасажирів, виявлення основних проблем, які потрібно вирішити через автоматизацію. Це включає збір вимог щодо функціональності системи, інтерфейсу користувача, безпеки, звітності та інших аспектів;

2) проектування архітектури системи. Розроблення високорівневої архітектури програмного забезпечення, яка відповідає вимогам вокзалу та забезпечує ефективну роботу системи. Це включає визначення модулів, компонентів, взаємодій між ними та даних, які будуть використовуватися;

3) розробка функціональності. Реалізація основних функціональних можливостей системи, таких як онлайн-продаж квитків, інформаційна система, електронні табло, анонси тощо. Це включає розробку логіки програми, баз даних, інтерфейсу користувача та інші необхідні компоненти;

4) інтеграція з існуючими системами. У випадку, якщо вокзал вже використовує певні існуючі системи (наприклад, система квиткового обліку), потрібно забезпечити інтеграцію нової системи з цими вже існуючими системами, щоб забезпечити безперебійну роботу та обмін необхідною інформацією;

5) тестування та валідація. Проведення тестування програмного забезпечення для перевірки правильності його роботи, виявлення й виправлення помилок та недоліків. Це може включати функціональне тестування, тестування виконання, тестування безпеки та інші види тестування, які забезпечують якість та надійність системи;

6) впровадження та підтримка. Реалізація процесу впровадження системи на вокзалі, навчання персоналу роботі з новою системою, а також надання підтримки та технічної допомоги після впровадження. Це дозволяє забезпечити успішне використання системи та вирішення потенційних проблем.

Виконання цих задач дозволить досягти мети розроблення програмного забезпечення для автоматизації роботи приміського вокзалу та забезпечити його ефективну та безперебійну роботу, поліпшення якості обслуговування пасажирів та оптимізацію робочих процесів.

Об’єктом дослідження є процеси роботи програмного забезпечення для автоматизації роботи приміського вокзалу.

Предметом дослідження є апаратно-програмне забезпечення для розроблення програмного забезпечення для автоматизації роботи приміського вокзалу.

Практичне значення одержаних результатів полягає у підвищенні якості автоматизації роботи приміського вокзалу.

Результатами роботи є програмне забезпечення для автоматизації роботи приміського вокзалу.

Кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, використаних джерел з 14 найменувань, 2-х додатків.

Обсяг роботи складається з 50 сторінок основного тексту з 69 сторінок кваліфікаційної роботи та 21 рисунка.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд технологій, методів та інструментів для створення програмного забезпечення для автоматизації роботи приміського вокзалу

Огляд використовуваних технологій, методів та інструментів є важливим етапом розробки програмного забезпечення для автоматизації роботи приміського вокзалу. Детальне дослідження доступних технологій та інструментів допомагає вибрати найбільш підходящі для потреб проекту. Нижче наведено огляд деяких потенційних технологій, методів та інструментів, що можуть бути використані у роботі.

Web-розробка:

– Front-end. Для розробки користувацького інтерфейсу можна використовувати такі технології, як HTML, CSS і JavaScript, а також популярні фреймворки і бібліотеки, такі як React, Angular або Vue.js;

– Back-end. Для реалізації серверної частини можна використовувати мови програмування, такі як Python (з фреймворком Django або Flask), Java (з фреймворком Spring), або Node.js (з фреймворком Express).

Бази даних:

– реляційні бази даних. Такі системи, як MySQL, PostgreSQL (рис. 1.1) або Oracle, можуть бути використані для зберігання та керування даними про розклади, білети, пасажирів та іншу відповідну інформацію;

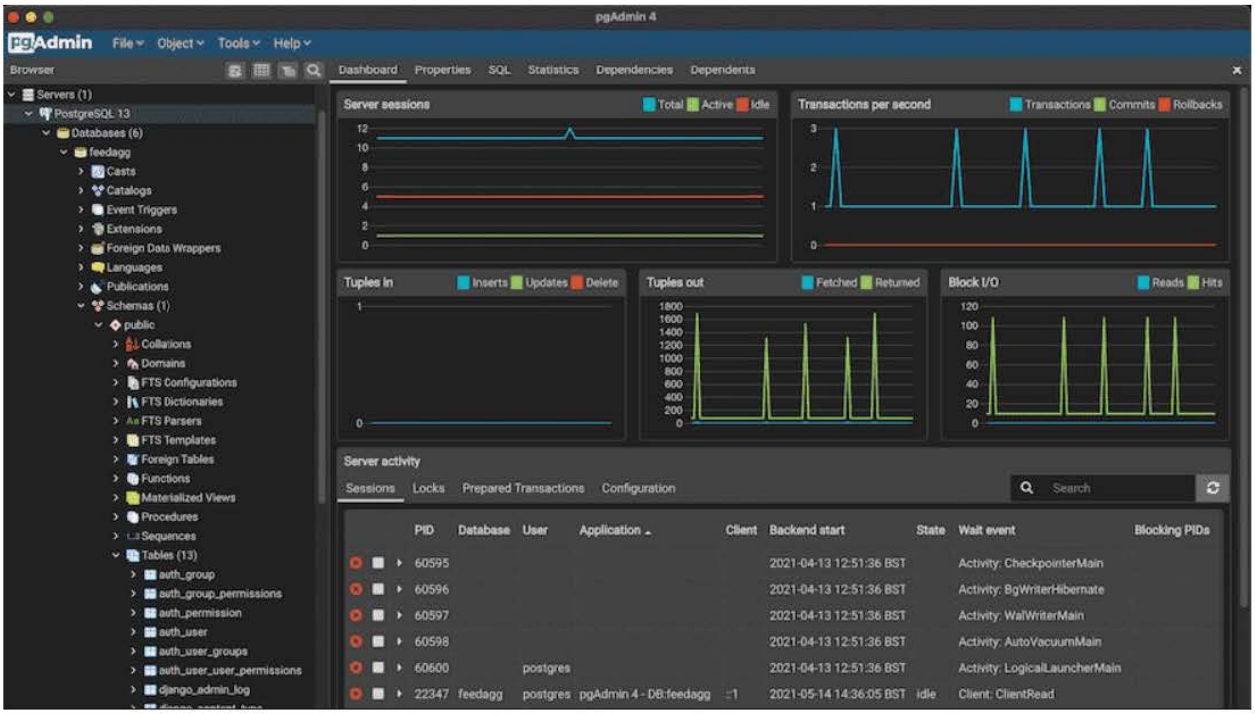

Рисунок 1.1 – Інтерфейс pgAdmin

– NoSQL бази даних. Для зберігання нереляційних даних, таких як журнали подій або великі обсяги даних, можна використовувати системи, такі як MongoDB (рис. 1.2) або Cassandra (рис. 1.3).

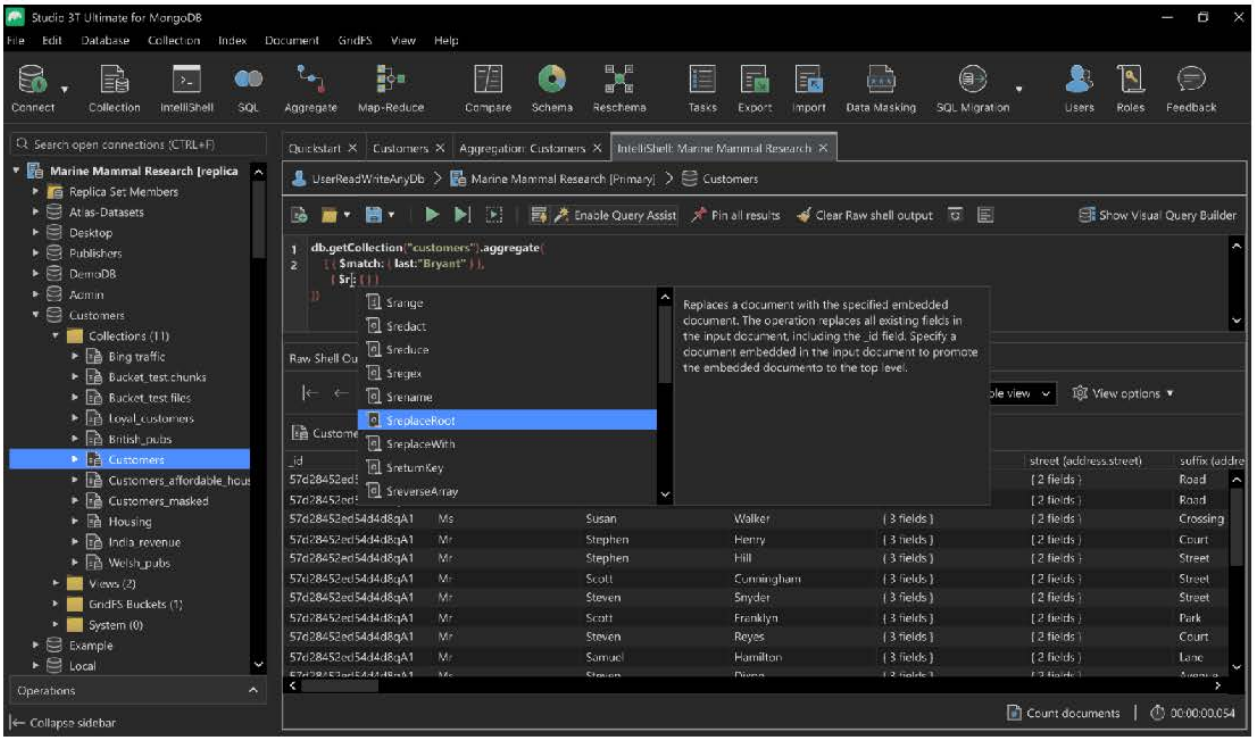

Рисунок 1.2 – Інтерфейс для взаємодії з MongoDB

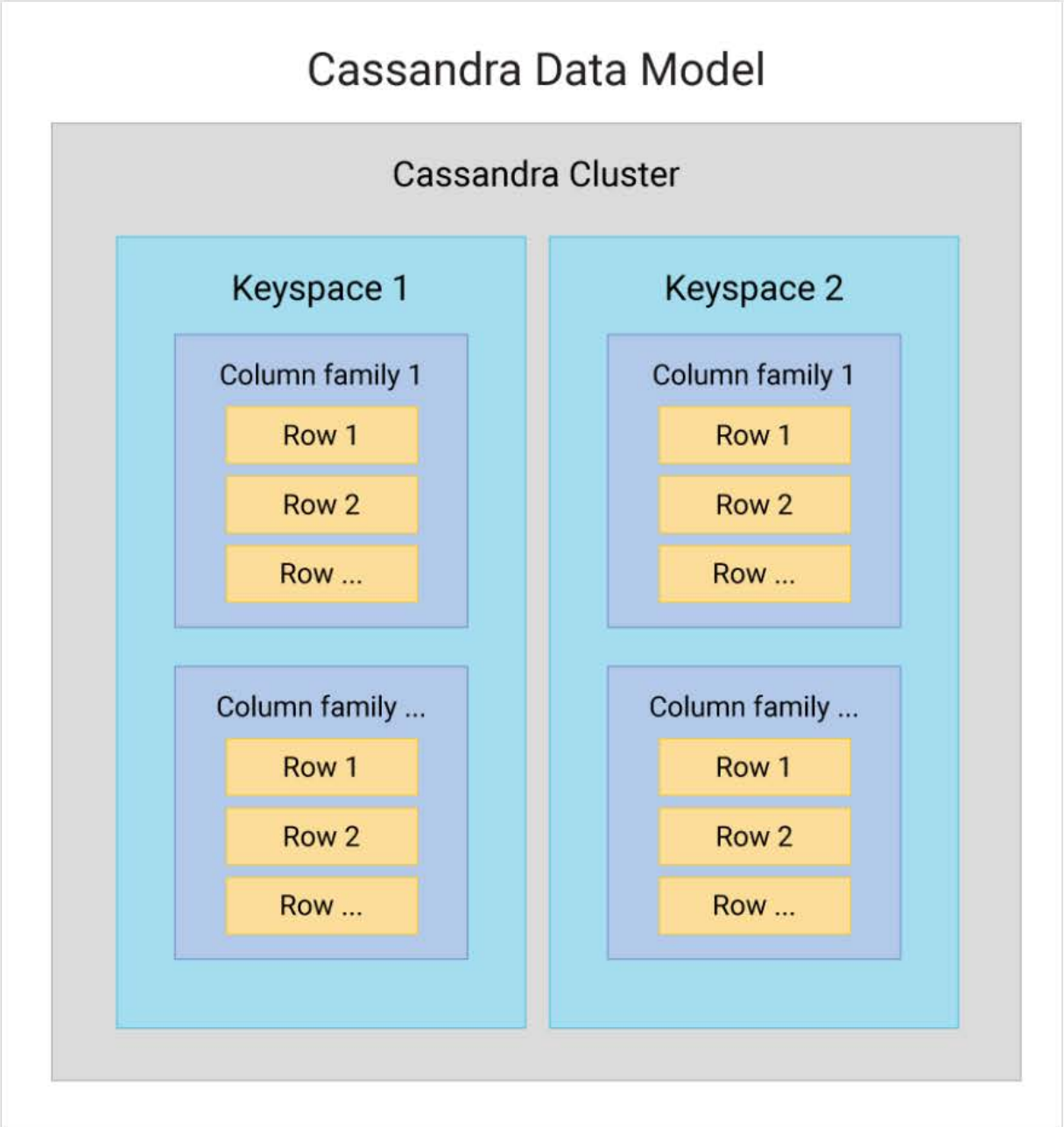

Рисунок 1.3 – Модель Cassandra

Мобільна розробка:

– платформи. Якщо потрібно розробити мобільний додаток, можна використовувати платформи, такі як Android (з використанням Java або Kotlin) або iOS (з використанням Swift або Objective-C);

– реактивні фреймворки. Розробка мобільних додатків може бути спрощена за допомогою реактивних фреймворків, таких як React Native

(рис. 1.4) або Flutter, які дозволяють використовувати один код для розробки додатків для різних платформ.

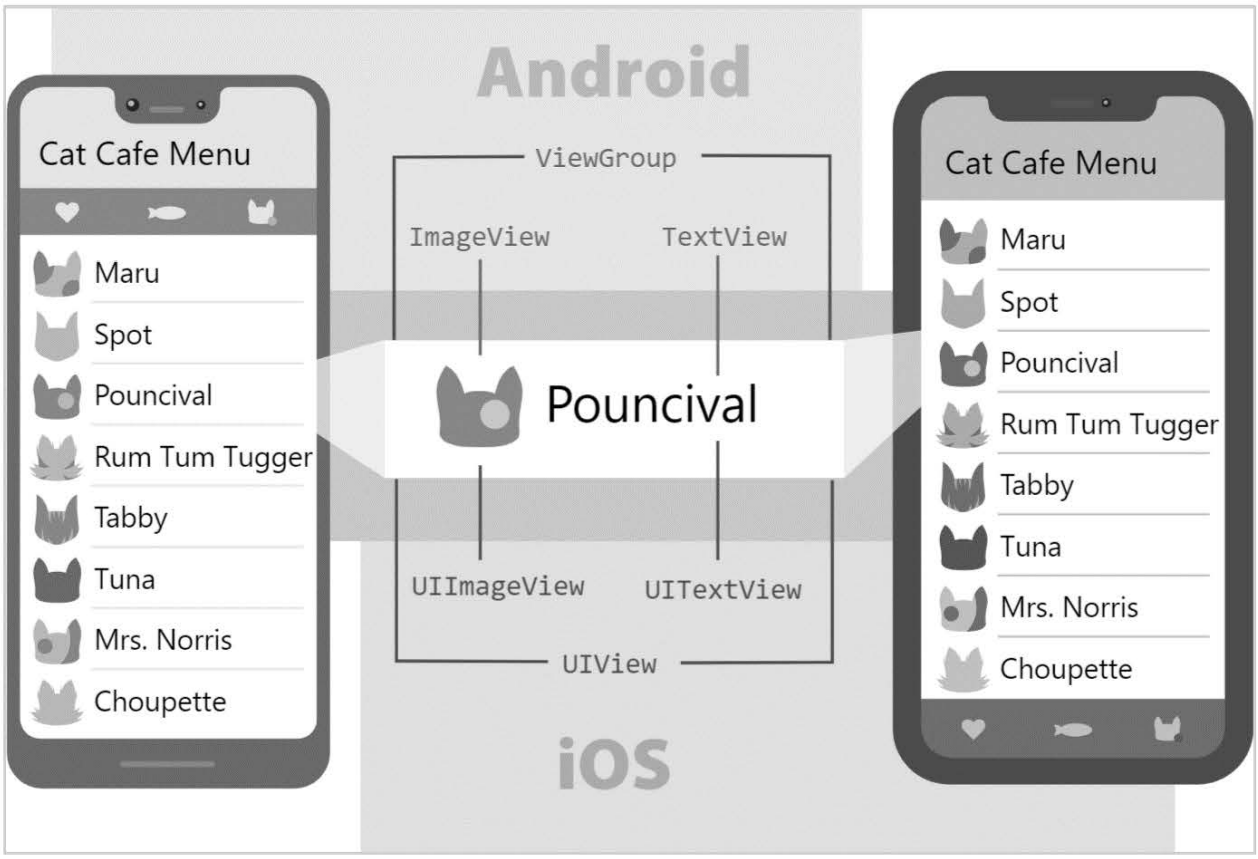

Рисунок 1.4 – Компоненти React Native

Аналіз даних:

– машинне навчання. Використання алгоритмів машинного навчання може допомогти виявити тенденції, прогнозувати попит на квитки, аналізувати використання вокзалу та покращувати процеси прийняття рішень;

– великі дані. Інструменти та технології для обробки та аналізу великих обсягів даних, такі як Apache Hadoop або Apache Spark (рис. 1.5), можуть бути використані для розуміння пасажирського потоку та виявлення патернів.

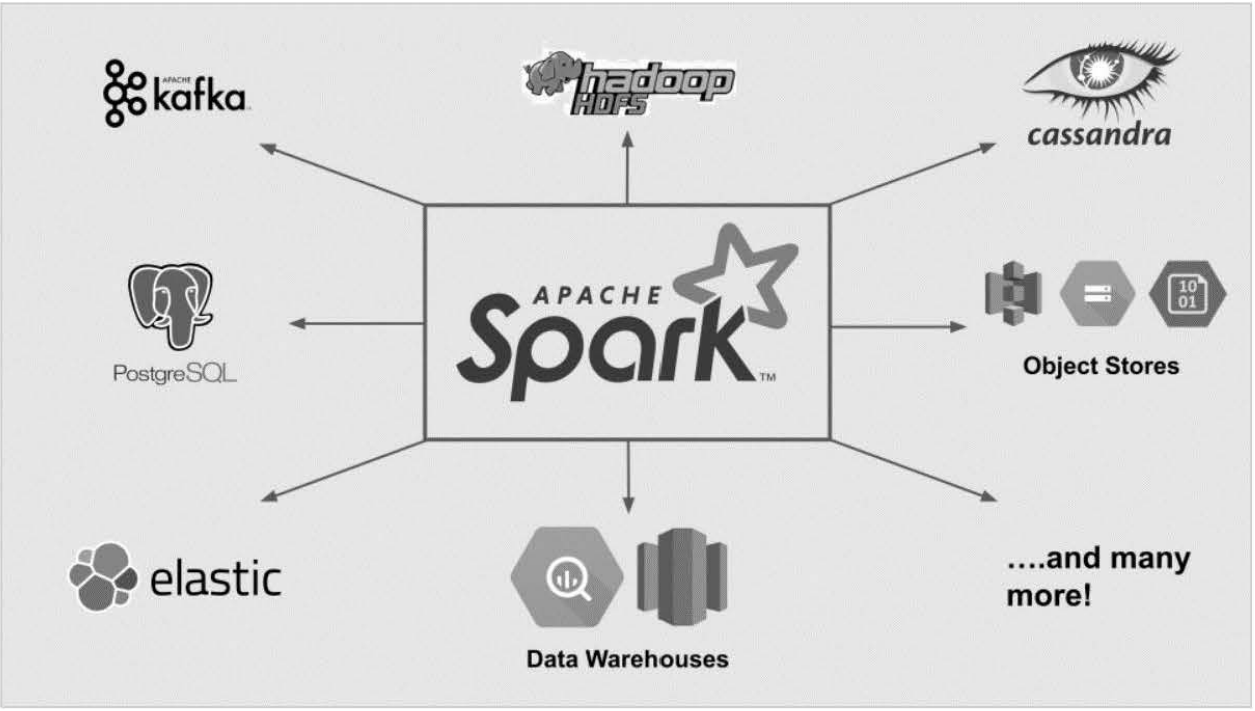

Рисунок 1.5 – Складові Apache Spark

Керування проектом та версіями:

– Git. Система контролю версій Git дозволяє ефективно керувати версіями програмного забезпечення та співпрацювати з іншими розробниками;

– Agile-методології. Методології розробки програмного забезпечення, такі як Scrum або Kanban, можуть бути використані для організації роботи команди та ефективного виконання завдань.

IoT (Internet of Things):

– використання сенсорів та IoT-пристроїв для збору даних про вокзал, наприклад, для моніторингу рівня заповненості поїздів, температури, вологості та інших параметрів;

– використання хмарних платформ для зберігання та аналізу отриманих даних.

RESTful API:

– розробка API (Application Programming Interface) для забезпечення взаємодії між різними системами та додатками;

– використання стандарту REST (Representational State Transfer) для передачі даних та виконання операцій.

Автоматизація тестування: використання фреймворків та інструментів для автоматизованого тестування програмного забезпечення, що дозволяє швидше та більш ефективно виявляти помилки та забезпечувати якість системи.

Системи управління проектами: використання інструментів, таких як JIRA або Trello, для організації роботи команди, відстеження завдань, призначення термінів та звітності про прогрес.

Безпека даних:

– використання шифрування та механізмів аутентифікації для захисту конфіденційності та цілісності даних пасажирів, розкладів та транзакцій;

– впровадження систем моніторингу та виявлення вторгнень для запобігання несанкціонованому доступу до системи.

Це лише кілька додаткових прикладів технологій, методів та інструментів, які можуть бути використані у роботі. При виборі конкретних рішень необхідно враховувати вимоги проекту, технічні можливості та власність до даних.

1.2 Опис функціональних та нефункціональних вимог

Визначення функціональних та нефункціональних вимог до системи є важливим кроком у розробці програмного забезпечення для автоматизації роботи приміського вокзалу. Функціональні вимоги описують, які функції та можливості повинна мати система, а нефункціональні вимоги визначають якості, вимоги до продуктивності та інші характеристики, які повинна мати система. Нижче наведено детальний опис кожного з цих типів вимог.

Функціональні вимоги:

– квиткова система. Система повинна дозволяти пасажирам придбати квитки на потяги або автобуси, переглянути розклади, вибрати місце та сплатити за квиток онлайн;

– інформаційна система. Система повинна надавати користувачам актуальну інформацію про розклади, затримки, маршрути та іншу важливу інформацію;

– електронні табло та анонси. Система повинна забезпечувати відображення інформації про потяги або автобуси на електронних табло в зручний для пасажирів спосіб;

– інтеграція з платіжними системами. Система повинна підтримувати різні способи оплати, такі як кредитні картки, електронні гаманці, банківські перекази тощо.

Нефункціональні вимоги:

– швидкодія. Система повинна бути швидкою та реагувати на запити користувачів майже миттєво, щоб уникнути затримок та незручностей;

– масштабованість. Система повинна бути здатною до масштабування для впорядкування зростаючого обсягу даних та навантаження на вокзалі;

– безпека. Система повинна забезпечувати захист конфіденційної інформації про пасажирів, а також механізми аутентифікації та авторизації;

– надійність. Система повинна бути надійною та стабільною, з мінімальною кількістю відмов та забезпечувати відновлення після збоїв;

– зручність використання. Система повинна мати інтуїтивний та зручний інтерфейс для користувачів, адаптований до різних пристроїв та браузерів;

– сумісність. Система повинна бути сумісною з різними операційними системами, браузерами та пристроями.

У детальному описі функціональних та нефункціональних вимог до системи важливо враховувати потреби та очікування користувачів, правила та регуляції, що стосуються роботи вокзалу, та інші фактори, які впливають на успішну реалізацію проекту.

1.3 Встановлення потреб та пріоритетів вимог

Встановлення потреб користувачів та пріоритетів вимог є важливим етапом у процесі розробки програмного забезпечення для автоматизації роботи приміського вокзалу. Цей етап допомагає зрозуміти, які функції та характеристики системи є найбільш важливими для користувачів та бізнесу. Нижче наведено детальний опис процесу встановлення потреб користувачів та пріоритетів вимог.

Аналіз користувачів:

– ідентифікація різних груп користувачів вокзалу, таких як пасажири, касири, персонал вокзалу, керівництво тощо;

– розуміння потреб та очікувань кожної групи користувачів;

– вивчення робочих процесів та взаємодії користувачів з системою.

Виявлення потреб:

– проведення інтерв'ю, опитувань та фокус-груп з представниками користувачів для визначення їх потреб та проблем;

– аналіз існуючих систем, які використовуються вокзалом, для ідентифікації недоліків та можливостей для поліпшення;

– врахування законодавчих вимог, стандартів та регуляторних вимог, що стосуються роботи вокзалу.

Формулювання вимог:

– визначення функціональних та нефункціональних вимог на основі потреб користувачів;

– приділення пріоритетів вимогам, визначення, які функції та характеристики є найбільш важливими та критичними для успішної роботи системи;

– врахування бізнес-цілей та стратегії вокзалу для встановлення вимог, які допоможуть досягти бажаного результату.

Взаємодія зі зацікавленими сторонами:

– комунікація з представниками вокзалу, керівництвом та іншими зацікавленими сторонами для обговорення вимог та уточнення деталей;

– врахування обмежень, таких як бюджет, ресурси та терміни, при прийнятті рішень щодо вимог.

Документування вимог:

– оформлення вимог у вигляді документації, яка включає опис функцій, інтерфейсів, обмежень та інших характеристик системи;

– використання діаграм, прототипів та інших інструментів для візуалізації вимог та полегшення їх розуміння.

Важливо пам'ятати, що потреби користувачів та пріоритети вимог можуть змінюватися протягом процесу розробки. Тому варто підтримувати відкриту комунікацію з користувачами та стежити за змінами у бізнес-середовищі, щоб адаптувати вимоги відповідно до нових умов і потреб.

1.4 Висновки до першого розділу. Постановка задач дослідження

Згідно з викладеним змістом, висновками до першого розділу є наступне:

1) у розділі «Дослідження предметної області» проведено огляд технологій, методів та інструментів, які використовуються для створення програмного забезпечення для автоматизації роботи приміського вокзалу. Цей огляд дозволив ознайомитись зі сучасними рішеннями та підходами, які можуть бути застосовані в даному проекті;

2) вимоги до системи були розбиті на функціональні та нефункціональні. Функціональні вимоги визначають, які функції та можливості повинна мати система, в той час як нефункціональні вимоги стосуються характеристик, якостей та обмежень системи;

3) встановлення потреб користувачів та пріоритетів вимог було проведено шляхом аналізу потреб різних груп користувачів вокзалу,

інтерв'ювання, опитувань та фокус-груп. Цей процес допоміг зрозуміти, які функції та характеристики є найбільш важливими для користувачів та бізнесу;

4) висновки до першого розділу вказують на важливість проведення дослідження предметної області, постановку задач дослідження та визначення вимог до системи. Це створює основу для подальшого розвитку проекту та розробки програмного забезпечення для автоматизації роботи приміського вокзалу.

Зазначений перший розділ дозволяє зрозуміти загальний контекст проекту, визначити основні напрямки роботи та планувати наступні етапи дослідження та розробки.

Метою роботи є розроблення програмного забезпечення для автоматизації роботи приміського вокзалу.

Для досягнення поставленої мети розроблення програмного забезпечення для автоматизації роботи приміського вокзалу можуть ставитися наступні задачі:

1) аналіз вимог та потреб. Проведення детального аналізу потреб вокзалу та пасажирів, виявлення основних проблем, які потрібно вирішити через автоматизацію. Це включає збір вимог щодо функціональності системи, інтерфейсу користувача, безпеки, звітності та інших аспектів;

2) проектування архітектури системи. Розроблення високорівневої архітектури програмного забезпечення, яка відповідає вимогам вокзалу та забезпечує ефективну роботу системи. Це включає визначення модулів, компонентів, взаємодій між ними та даних, які будуть використовуватися;

3) розробка функціональності. Реалізація основних функціональних можливостей системи, таких як онлайн-продаж квитків, інформаційна система, електронні табло, анонси тощо. Це включає розробку логіки програми, баз даних, інтерфейсу користувача та інші необхідні компоненти;

4) інтеграція з існуючими системами. У випадку, якщо вокзал вже використовує певні існуючі системи (наприклад, система квиткового обліку), потрібно забезпечити інтеграцію нової системи з цими вже існуючими

системами, щоб забезпечити безперебійну роботу та обмін необхідною інформацією;

5) тестування та валідація. Проведення тестування програмного забезпечення для перевірки правильності його роботи, виявлення й виправлення помилок та недоліків. Це може включати функціональне тестування, тестування виконання, тестування безпеки та інші види тестування, які забезпечують якість та надійність системи;

6) впровадження та підтримка. Реалізація процесу впровадження системи на вокзалі, навчання персоналу роботі з новою системою, а також надання підтримки та технічної допомоги після впровадження. Це дозволяє забезпечити успішне використання системи та вирішення потенційних проблем.

РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ПРИМІСЬКОГО ВОКЗАЛУ

2.1 Вибір програмних засобів для розроблення програмного забезпечення для автоматизації роботи приміського вокзалу

Для розробки застосунку з графічним інтерфейсом для приміського вокзалу на платформі Windows були використані наступні технології:

1) Windows Forms. Це технологія розробки графічного інтерфейсу для Windows додатків. Вона дозволяє створювати вікна, кнопки, тексти, списки та інші елементи інтерфейсу за допомогою класів і компонентів з .NET Framework;

2) C#. Це мова програмування, яка використовується для розробки додатків на платформі .NET. C# є сучасною, об'єктно-орієнтованою мовою зі сильною типізацією, яка дозволяє ефективно взаємодіяти з компонентами Windows Forms;

3) Visual Studio. Це інтегроване середовище розробки (IDE) від Microsoft, яке надає зручний набір інструментів для створення програм на платформі .NET. Використовуючи Visual Studio, можна швидко створити, налаштувати та відлагодити застосунок Windows Forms;

4) .NET Framework. Це платформа розробки програмного забезпечення від Microsoft. Вона надає серію бібліотек і компонентів, які використовуються для розробки додатків на платформі Windows, включаючи Windows Forms.

Ці технології дозволяють розробникам створювати ефективні та інтуїтивно зрозумілі застосунки з графічним інтерфейсом для Windows.

Додатково, для розробки застосунку з графічним інтерфейсом для приміського вокзалу, можуть бути використані такі технології та підходи:

– архітектурний підхід. Для розробки застосунку можна використовувати певний архітектурний підхід, наприклад, Model-View-

Controller (MVC) або Model-View-ViewModel (MVVM), що дозволить відокремити логіку додатку від його графічного інтерфейсу і спростити його розширення та підтримку;

– база даних. Для збереження даних про автобуси, квитки та розклад руху можна використовувати базу даних, наприклад, SQL Server, MySQL або SQLite. Використання бази даних дозволяє зберігати, оновлювати та видаляти інформацію про розклади, квитки тощо;

– API або сервіси. Якщо приміський вокзал має зовнішні системи або сервіси, такі як система бронювання або платіжний шлюз, можна використовувати API або сервіси для комунікації та взаємодії з ними;

– інтернаціоналізація та локалізація. Якщо планується, що застосунок буде використовуватися в різних мовах або країнах, слід врахувати можливість локалізації інтерфейсу, щоб забезпечити його придатність до використання в різних культурних середовищах;

– обробка помилок та винятків. Важливо розглянути можливі помилки та виняткові ситуації, що можуть виникнути при роботі з додатком, і забезпечити відповідну обробку та повідомлення про помилки для користувачів.

Зазначені технології та підходи є лише прикладами і можуть бути доповнені або змінені відповідно до конкретних вимог та умов проекту розробки застосунку для приміського вокзалу.

2.2 Windows Forms

Windows Forms (WinForms) є одним з фреймворків розробки графічного інтерфейсу користувача (GUI) для платформи Microsoft Windows. Він надає зручний спосіб створення десктопних додатків з використанням вікон, кнопок, полів вводу, списків, меню і багатьох інших елементів управління (рис. 2.1).

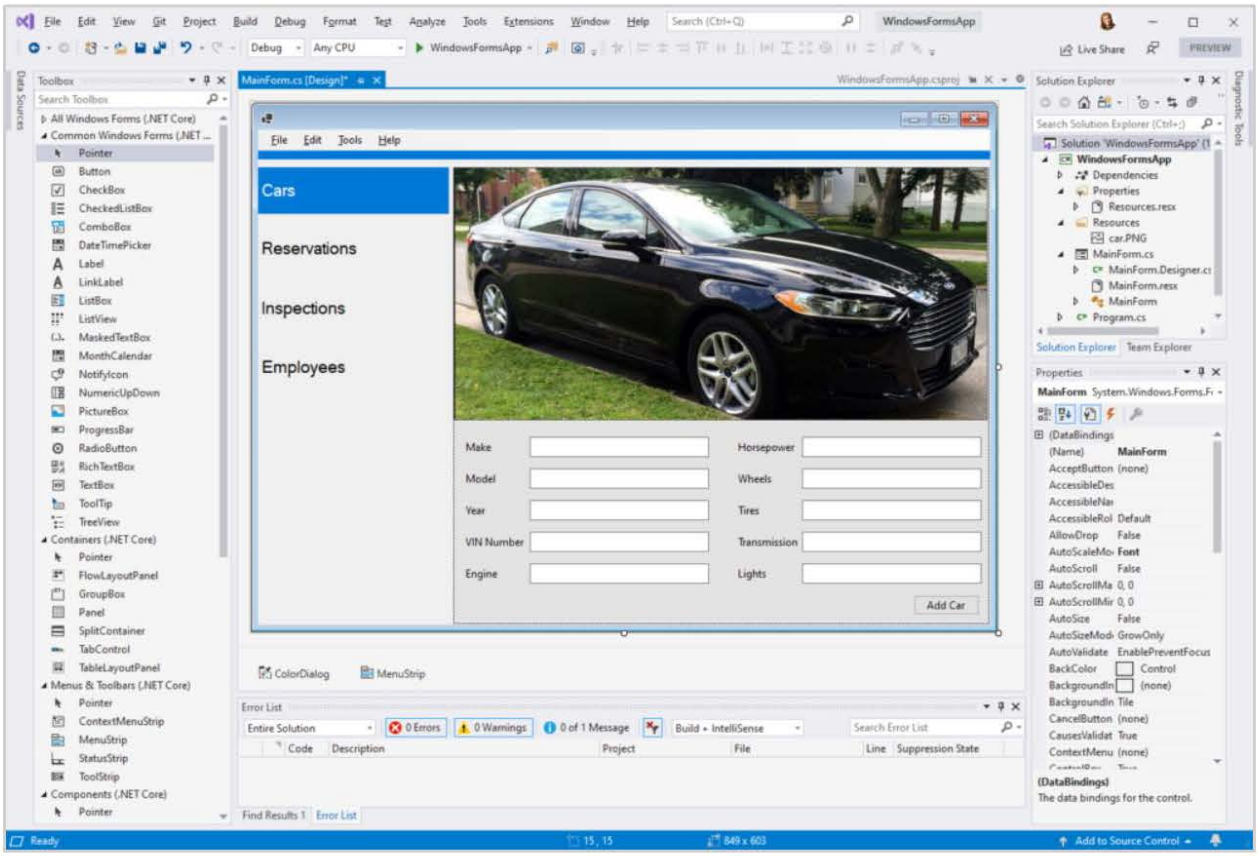

Рисунок 2.1 – Інтерфейс Windows Forms

Особливості Windows Forms:

1) розробка на мові програмування C#. Windows Forms зазвичай використовуються з мовою програмування C#. Це потужна та елегантна мова, яка надає широкий спектр можливостей для розробки додатків;

2) візуальний дизайнер. Visual Studio, основне інтегроване середовище розробки (IDE) для платформи .NET, має вбудований візуальний дизайнер Windows Forms. Він дозволяє створювати і налаштовувати вигляд і поведінку елементів управління шляхом перетягування та встановлення властивостей;

3) елементи управління. Windows Forms надає велику кількість готових елементів управління, таких як кнопки, текстові поля, випадаючі списки, панелі, меню, вкладки і багато інших. Можна використовувати їх для побудови інтерактивного інтерфейсу користувача свого додатка;

4) події та обробники подій. Windows Forms дозволяє зв'язувати обробники подій з елементами управління. Можна реагувати на події, такі як

натискання кнопки або введення тексту, та виконувати певні дії відповідно до цих подій;

5) графічна обробка. Windows Forms має потужні засоби для графічної обробки, такі як малювання фігур, застосування кольорів, відображення зображень та робота з текстом. Можна створювати власні малюнки та реалізовувати різні ефекти;

6) розміщення контролів. Можна використовувати різні стратегії розміщення контролів на формі, такі як таблиця, потоковий макет або абсолютні координати. Це дозволяє створювати гнучкі та привабливі розміщення елементів управління;

7) взаємодія з базами даних. Windows Forms добре інтегрується з базами даних. Можна зчитувати та записувати дані у базі даних, використовуючи різні технології доступу до даних, такі як ADO.NET або Entity Framework;

8) налаштування і міжнародна підтримка. Windows Forms дозволяє налаштовувати різні аспекти додатка, такі як мова, культура, теми оформлення і шрифти. Це дає можливість створювати додатки, які підтримують міжнародну публіку.

За допомогою Windows Forms можна реалізувати різноманітні функціональність, яка полегшує розробку додатків:

1) валідація даних. Windows Forms надає можливість виконувати валідацію введених користувачем даних. Можна перевіряти правильність формату даних, обов'язкові поля, діапазони значень та інші умови. Це допомагає забезпечити правильність та консистентність даних, які вводяться користувачем;

2) доступ до системних ресурсів. Windows Forms дозволяє отримувати доступ до різних системних ресурсів, таких як файлова система, реєстр Windows, мережеві сервіси тощо. Це дозволяє взаємодіяти з іншими додатками та використовувати їх функціонал у додатку;

3) робота зі звуком і мультимедіа. Windows Forms має підтримку відтворення звуку і відео. Можна відтворювати аудіофайли, відеофайли,

створювати звукові ефекти та інше. Це дозволяє створювати більш залучаючі додатки з мультимедійними можливостями;

4) робота зі зовнішніми пристроями. Windows Forms дозволяє взаємодіяти зі зовнішніми пристроями, такими як принтери, сканери, камери, USB-пристрої і багато інших. Можна читати, записувати або керувати цими пристроями з додатку;

5) мультипотоковість. Windows Forms підтримує роботу з мультипотоковістю, що дозволяє виконувати обчислення та довготривалі операції в окремих потоках, не блокуючи головний потік інтерфейсу користувача. Це забезпечує більш відзивчивий та швидкий інтерфейс, а також уникнення «замерзання» програми під час виконання складних операцій;

6) інтеграція з веб-службами. Windows Forms дозволяє взаємодіяти з веб-службами та використовувати їх функціонал у додатку. Можна здійснювати виклики до веб-служб, обмінюватися даними і виконувати різноманітні операції, такі як збереження в базі даних, обробка даних тощо.

Windows Forms є сучасним і потужним інструментом для розробки додатків з графічним інтерфейсом користувача. Він дозволяє швидко створювати функціональні та естетичні додатки, які можуть взаємодіяти з різними ресурсами та сервісами, що доступні у операційній системі Windows.

2.3 Мова програмування C#

C# (C Sharp) – це сучасна, ООП (об'єктно-орієнтована) мова програмування, розроблена компанією Microsoft. Вона використовується для створення різноманітних програмних рішень, включаючи десктопні додатки, веб-додатки, мобільні додатки і багато іншого:

– синтаксис. C# має чистий, лаконічний та легкий для читання синтаксис. Він був сформований з урахуванням зрозумілості для розробників і має схожість з іншими мовами програмування, такими як C++ і Java. Це робить C# легким для вивчення та використання;

– об'єктно-орієнтованість. C# базується на принципах об'єктно-орієнтованого програмування (ООП). Вона підтримує поняття класів, об'єктів, успадкування, поліморфізму, інкапсуляції та абстракції. Це дозволяє створювати модульний, гнучкий і легко зрозумілий код;

– платформа .NET. C# є однією з основних мов програмування, які використовуються в платформі .NET. Це дозволяє розробникам використовувати широкий спектр бібліотек, фреймворків та інструментів, що надаються платформою .NET для розробки різноманітних додатків;

– спрощений керівництво пам'яттю. У C# використовується система автоматичного керування пам'яттю (Garbage Collector), що полегшує роботу з пам'яттю. Вона автоматично видаляє об'єкти, які більше не використовуються, що допомагає уникнути багатьох проблем, пов'язаних з утриманням пам'яті;

– можливості паралельного програмування. C# має вбудовану підтримку паралельного програмування. Можна використовувати паралельні потоки (Parallel Threads) та завдання (Tasks) для одночасного виконання обчислень і підвищення продуктивності додатків;

– розширюваність. Мова C# має підтримку розширення (Extension Methods), що дозволяє додавати нові методи до існуючих типів без необхідності модифікації вихідного коду цих типів. Це дозволяє розширювати функціональність існуючих класів і забезпечує більшу гнучкість в розробці програмного забезпечення;

– інтеграція з іншими мовами. C# може легко інтегруватися з іншими мовами програмування, такими як Visual Basic .NET і C++. Можна використовувати компоненти, написані на різних мовах програмування, у C#-проекті;

– підтримка візуальних інструментів. C# підтримує візуальні інструменти, такі як Visual Studio, що спрощують процес розробки. Вони надають розширену підтримку редактора, відладчика, системи контролю

версій та інших інструментів, що полегшують розробку програмного забезпечення.

– платформа .NET. C# є основною мовою програмування, яка працює на платформі .NET. Ця платформа надає великий набір функцій і бібліотек для розробки програмного забезпечення. Вона включає в себе базові класи, колекції, обробку винятків, роботу з файлами, мережеву взаємодію та багато іншого. Платформа .NET забезпечує переносимість коду між різними платформами, що дає змогу використовувати C# для розробки додатків, які працюють на різних операційних системах, таких як Windows, macOS і Linux;

– широке співробітництво з Microsoft. C# розробляється і підтримується компанією Microsoft, що забезпечує активну спільноту розробників та доступ до багатьох корисних ресурсів. Це включає офіційну документацію, навчальні матеріали, форуми, блоги, вебінари та інші ресурси, які допомагають розробникам у вивченні мови та вирішенні питань під час розробки проектів на C#;

– підтримка розробки мобільних додатків. C# використовується для розробки мобільних додатків на платформах Xamarin та Xamarin.Forms. З використанням C# розробники можуть створювати переносимі додатки для iOS та Android, використовуючи спільний код на C#. Це спрощує розробку мобільних додатків, забезпечує швидкість розробки та забезпечує переносимість між платформами;

– розширені можливості роботи з базами даних. C# надає потужні засоби для роботи з базами даних. Він підтримує об'єктно-орієнтоване взаємодію з базами даних через ADO.NET і Entity Framework. Це дозволяє розробникам легко зчитувати, записувати і змінювати дані в базах даних з використанням стандартних мовних конструкцій C#;

– можливості розробки веб-додатків. За допомогою мови програмування C# можна створювати веб-додатки на платформі ASP.NET. Використовуючи ASP.NET, розробники можуть створювати веб-сторінки, веб-служби та веб-

додатки з використанням мови C# і широкого набору функціональних можливостей, що надаються платформою;

– підтримка мультиплатформеності. Завдяки розвитку .NET Core, C# став більш мультиплатформеним, що дозволяє використовувати його для розробки додатків на різних операційних системах, включаючи Windows, macOS і Linux. Це дає розробникам більшу гнучкість у виборі платформи для розгортання своїх додатків.

Мова програмування C# пропонує багато можливостей для розробки різноманітних додатків на платформі .NET. Вона комбінує лаконічний синтаксис, об'єктно-орієнтованість, підтримку паралельного програмування та широкий вибір інструментів розробки, що робить її привабливим вибором для розробників. Незалежно від того, чи розробляється десктопний додаток, веб-додаток, мобільний додаток або будь-який інший вид програмного забезпечення, C# може бути потужним інструментом для досягнення цілей розробки.

2.4 Visual Studio

Visual Studio – це інтегроване середовище розробки (IDE), розроблене компанією Microsoft, яке надає широкі можливості для розробки різноманітних програмних продуктів (рис. 2.2).

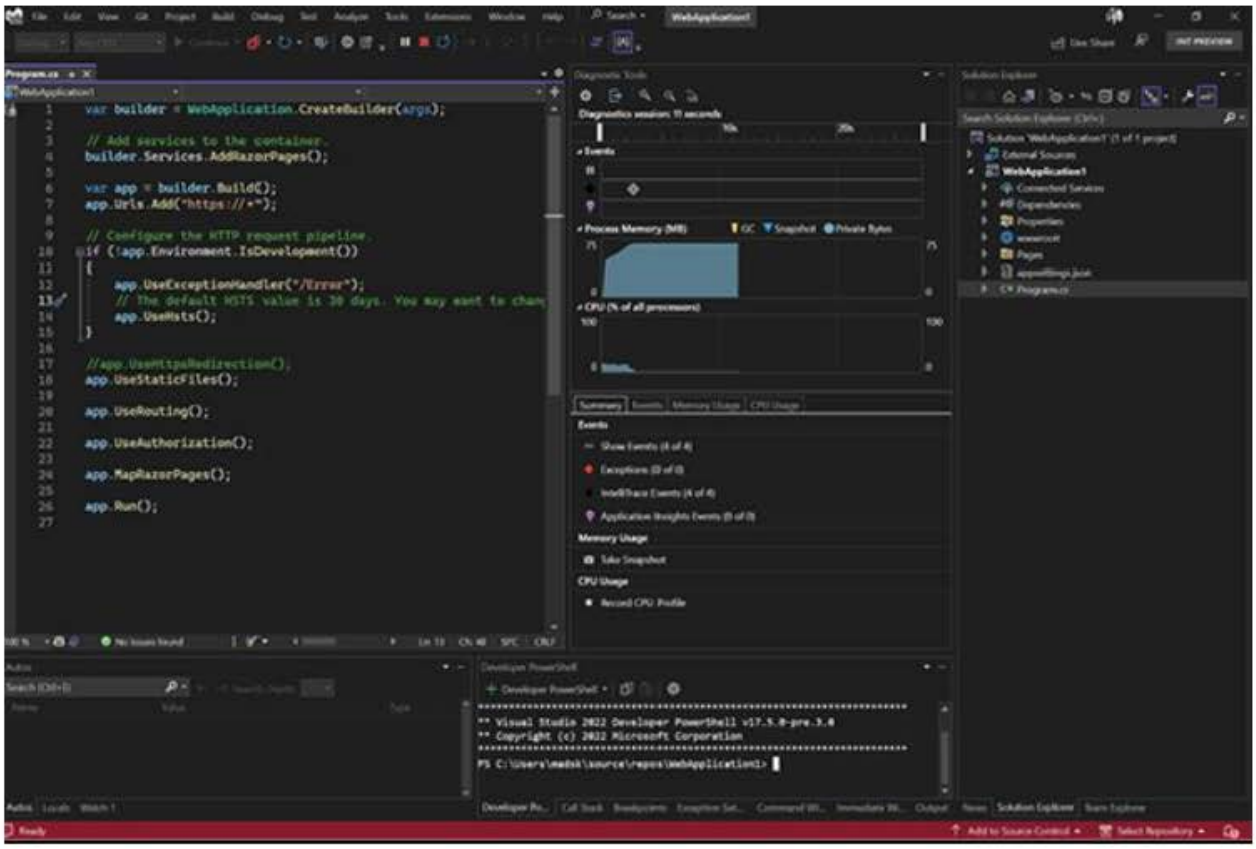

Рисунок 2.2 – Інтерфейс Visual Studio

Ось детальний опис Visual Studio:

– мови програмування та платформи. Visual Studio підтримує широкий набір мов програмування, включаючи C#, Visual Basic .NET, C++, F#, Python, JavaScript і багато інших. Крім того, він підтримує різні платформи, такі як Windows, .NET Framework, .NET Core, Xamarin, ASP.NET, Azure і багато інших;

– редактор коду. Visual Studio надає потужний редактор коду зі вбудованими функціями автодоповнення, перевірки синтаксису, форматування, кольорової підсвітки синтаксису і великої кількості інструментів для полегшення написання коду. Він також підтримує функцію розширення редактора, що дозволяє встановлювати додаткові розширення для розширення можливостей редактора;

– відлагодження. Visual Studio має потужні засоби відлагодження, які допомагають розробникам знайти та виправити помилки в програмному коді.

Це включає можливість крокування по коду, спостереження за значеннями змінних, встановлення точок зупинки, аналіз стеку викликів і багато інших корисних функцій;

– управління проектами. Visual Studio дозволяє створювати та управляти проектами різних типів, таких як консольні програми, бібліотеки класів, веб-додатки, мобільні додатки та інші. Він забезпечує шаблони проектів для різних платформ і мов програмування, що спрощує початок роботи з новим проектом;

– керування версіями. Visual Studio інтегрується з системами контролю версій, такими як Git і Team Foundation Server (TFS), що дозволяє розробникам легко керувати версіями свого коду, робити коміти, злиття та інші операції, пов'язані з управлінням версіями;

– веб-розробка. Visual Studio має широкі можливості для веб-розробки, включаючи підтримку HTML, CSS, JavaScript, ASP.NET, Blazor і багато інших технологій. Він надає засоби для швидкої створення веб-сторінок, взаємодії з базами даних, розгортання веб-додатків на серверах і багато іншого;

– тестування. Visual Studio має вбудовані засоби для автоматизованого тестування програмного коду. Він підтримує юніт-тестування, функціональне тестування, тестування производительності і інші типи тестів. Це дозволяє розробникам писати тести для перевірки правильності роботи їх програмного коду;

– розширюваність. Visual Studio дозволяє розробникам створювати власні розширення і додатки, що розширюють функціональність IDE. Це дозволяє налаштовувати середовище розробки під потреби і використовувати додаткові інструменти, які полегшують роботу.

– вбудовані інструменти для створення графічних інтерфейсів. Visual Studio надає розширені можливості для створення графічних інтерфейсів для десктопних, вебта мобільних додатків. Можна використовувати візуальні редактори, такі як Windows Forms Designer, WPF Designer, ASP.NET Designer, Xamarin Designer і інші, щоб швидко створювати і налаштовувати інтерфейс користувача за допомогою перетягування та розміщення елементів керування;

– інтегрована довідка та документація. Visual Studio надає вбудовану довідку і документацію, які допомагають розробникам швидко знайти необхідну інформацію. Можна переглядати документацію API, читати статті, переглядати приклади коду і навіть здійснювати пошук у мережі для отримання додаткових ресурсів і рішень;

– аналіз коду та перевірка якості. Visual Studio має ряд вбудованих інструментів для аналізу коду, виявлення потенційних помилок, недоліків і проблем з безпекою. Можна використовувати статичний аналізатор, кодові метрики, інструменти перевірки стилю коду та інші функції, щоб забезпечити високу якість коду і виявити можливі проблеми ще до запуску програмного продукту;

– інтеграція з хмарними сервісами. Visual Studio підтримує інтеграцію з хмарними сервісами, такими як Azure, що дозволяє розробникам легко взаємодіяти з хмарними послугами, створювати, розгортати та керувати хмарними додатками та сервісами безпосередньо зі свого середовища розробки;

– розширена підтримка тестування. Visual Studio має інструменти для автоматизованого тестування, включаючи модульні тести, тести інтерфейсу користувача, тести навантаження та багато інших. Можна створювати, запускати і аналізувати тести безпосередньо в середовищі Visual Studio;

– спільна робота і збереження проектів. Visual Studio надає засоби для спільної роботи над проектами у команді. Можна використовувати системи контролю версій, такі як Git або TFS, для спільного доступу до коду, вирішення конфліктів злиття, розподілення завдань і спільного ведення проекту в межах команди розробників.

Visual Studio є потужним інструментом для розробки програмного забезпечення на різних платформах і з використанням різних мов програмування. Він надає багато інструментів, що полегшують написання, налагодження і тестування коду, а також допомагають управляти проектами та версіями. Завдяки широкому спектру можливостей Visual Studio,

розробники можуть більш ефективно працювати над своїми проектами і створювати високоякісне програмне забезпечення.

2.5 .NET Framework

.NET Framework (рис. 2.3) є платформою розробки програмного забезпечення, яка надає середовище для створення та виконання додатків різних типів:

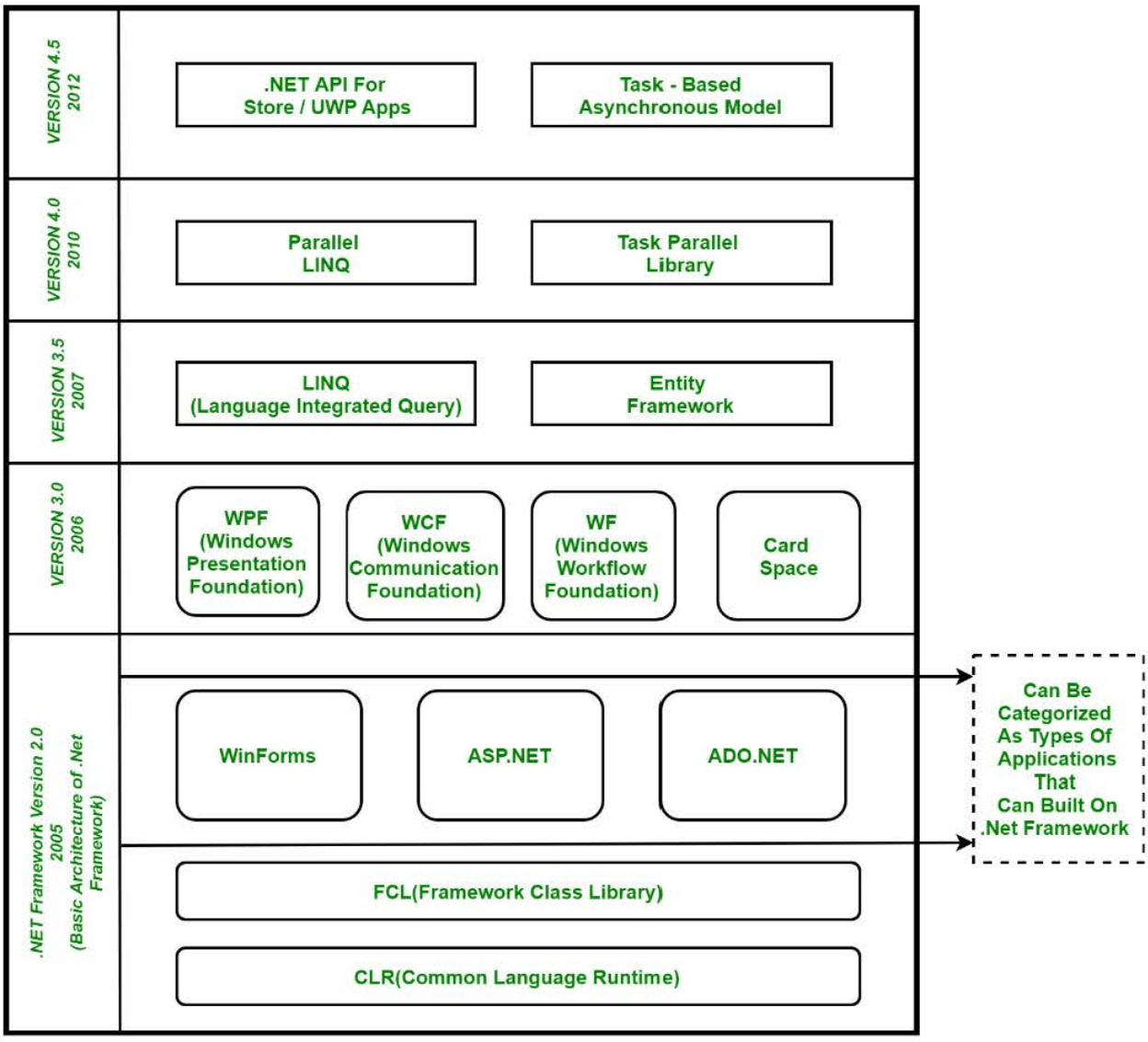

Рисунок 2.3 – Архітектура .NET Framework

– мови програмування. .NET Framework підтримує використання різних мов програмування, таких як C#, Visual Basic.NET, F# та інші. Це дає розробникам можливість вибирати мову, з якою вони найбільш комфортно працюють, і використовувати її для створення програмних рішень;

– класи та бібліотеки. .NET Framework має велику бібліотеку класів, яка включає в себе багато готових компонентів і функцій для різних сфер програмування. Це спрощує розробку додатків, оскільки розробники можуть використовувати готові класи для виконання різних завдань, таких як робота з файлами, мережеве взаємодія, робота з базами даних і багато іншого;

– виконавче середовище. .NET Framework надає виконавче середовище для запуску програм, відоме як Common Language Runtime (CLR). CLR відповідає за виконання коду, керування пам'яттю, обробку виключень, безпеку і інші аспекти виконання програм. Він забезпечує високу продуктивність і надійність додатків, а також незалежність від конкретної апаратної платформи;

– підтримка різних платформ. .NET Framework може бути використаний для розробки додатків для різних платформ, включаючи Windows, Linux і macOS. Завдяки відкритому розширенню .NET Framework, відомому як .NET Core, розробники можуть створювати крос-платформені додатки, які можуть працювати на різних операційних системах;

– інтеграція з Microsoft-технологіями. .NET Framework добре інтегрується з іншими Microsoft-технологіями і продуктами. Наприклад, можна легко використовувати .NET Framework для розробки веб-додатків з використанням ASP.NET, доступу до баз даних з використанням ADO.NET або створення Windows-додатків з використанням Windows Forms або WPF;

– підтримка розробки інструментів. Для розробки додатків на .NET Framework широко використовуються різні інтегровані середовища розробки (IDE), зокрема Microsoft Visual Studio. Visual Studio надає багато функціональних можливостей, таких як візуальний редактор коду,

інструменти налагодження, відладчик, система контролю версій і багато іншого, що полегшує розробку додатків на .NET Framework;

– веб-сервіси і служби хмари. .NET Framework підтримує розробку веб-додатків та інтеграцію з хмарними сервісами, зокрема з Microsoft Azure. Це дозволяє розробникам будувати масштабовані, безпечні та доступні веб-додатки, які можуть використовувати різноманітні хмарні ресурси і сервіси.

– розширені можливості безпеки. .NET Framework має розширені можливості безпеки, що дозволяє розробникам захищати свої додатки від потенційних загроз і атак. Він надає засоби для автентифікації, авторизації, криптографії та керування доступом, що допомагає забезпечити конфіденційність і цілісність даних;

– мультиплатформеність. Однією з ключових особливостей .NET Framework є його можливість роботи на різних платформах, включаючи Windows, Linux і macOS. Це досягнуто завдяки .NET Core відкритій версії .NET Framework, яка надає крос-платформену підтримку. Це дозволяє розробникам створювати додатки, які можуть працювати на різних операційних системах без значних змін в коді;

– підтримка для розробки мобільних додатків. .NET Framework надає інструменти і бібліотеки для розробки мобільних додатків під платформи Android та iOS. З використанням Xamarin фреймворку для розробки мобільних додатків на базі .NET, розробники можуть створювати переносні додатки, що працюють на різних мобільних платформах з використанням загального коду;

– підтримка для IoT. .NET Framework має підтримку для розробки додатків для Інтернету речей (IoT). З використанням платформи .NET та спеціалізованих бібліотек, розробники можуть створювати програмне забезпечення для підключених пристроїв, сенсорів та систем автоматизації;

– масштабованість. .NET Framework надає інструменти для розробки масштабованих додатків. Він підтримує розподілені обчислення, взаємодію з веб-сервісами, роботу з базами даних в режимі реального часу та інші

технології, що дозволяють розробникам будувати потужні та ефективні системи.

.NET Framework є широко використовуваною та надійною платформою для розробки програмного забезпечення. Вона має багато функціональних можливостей, що полегшують розробку додатків, забезпечують безпеку, масштабованість та підтримку різних платформ, що робить її відмінним вибором для розробників у різних сферах програмування.

2.6 Огляд баз даних

Бази даних є важливою складовою будь-якої системи управління даними, включаючи систему автоматизації приміського вокзалу. Використання бази даних дозволяє ефективно зберігати, організовувати і керувати великим обсягом даних, пов'язаних з автобусами, квитками та розкладом руху. Ось деякі деталі про популярні бази даних, такі як SQL Server, MySQL та SQLite:

SQL Server (рис. 2.4):

– SQL Server є реляційною базою даних, розробленою компанією Microsoft. Вона пропонує розширені можливості для управління даними і масштабується від невеликих проектів до підприємственних рішень;

– SQL Server підтримує мову запитів SQL, що дозволяє виконувати складні запити до бази даних і отримувати необхідну інформацію;

– він надає широкий спектр функцій для забезпечення безпеки, резервного копіювання даних, відновлення та керування доступом до бази даних;

– SQL Server може бути розгорнутий на різних платформах, включаючи Windows і Linux, що дає гнучкість вибору оптимального середовища для розробки та експлуатації.MySQL.

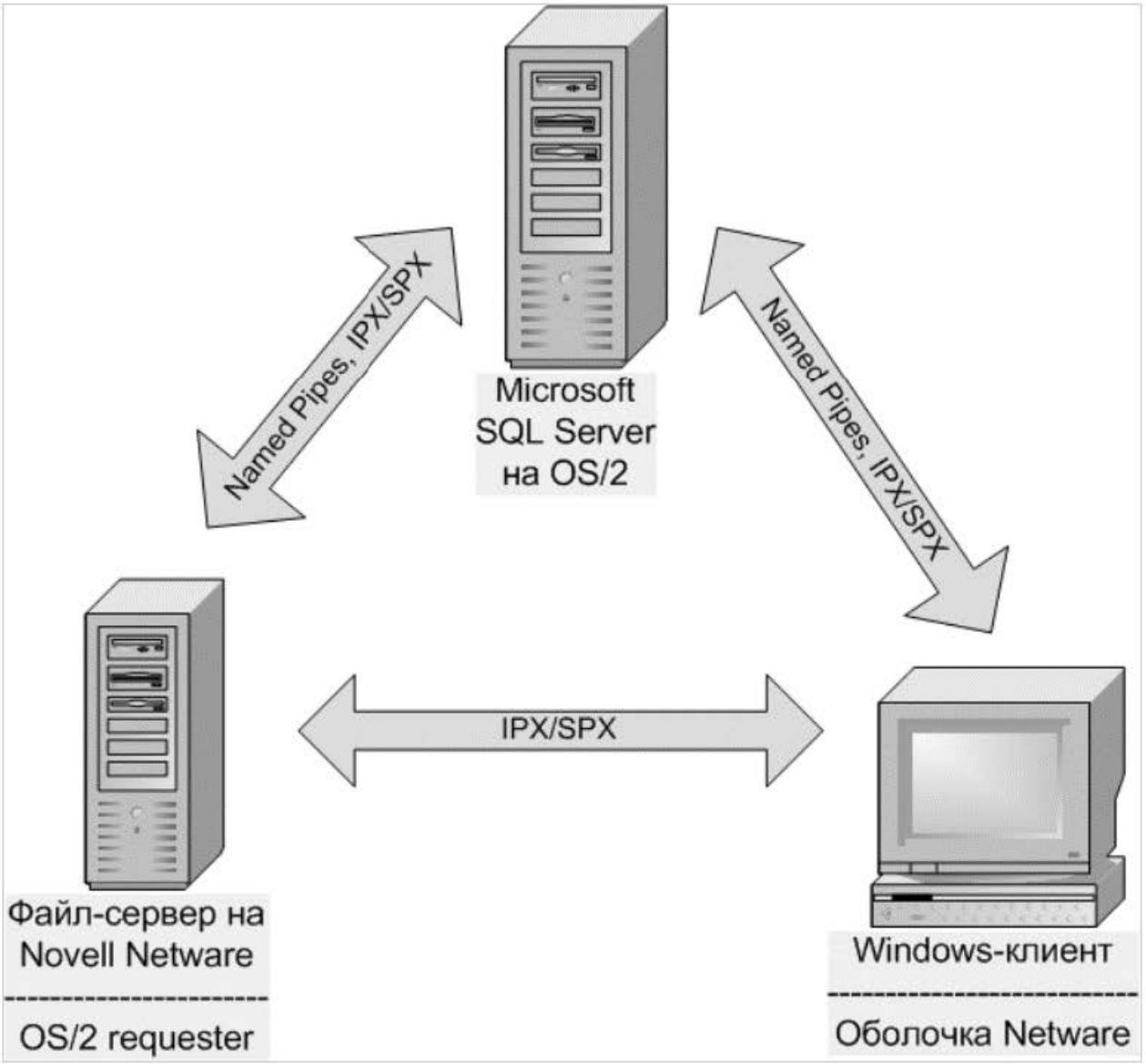

Рисунок 2.4 – Схема роботи SQL Server

– MySQL також є реляційною базою даних, але він має відкритий вихідний код і є популярним вибором для веб-додатків та невеликих проектів (рис. 2.5);

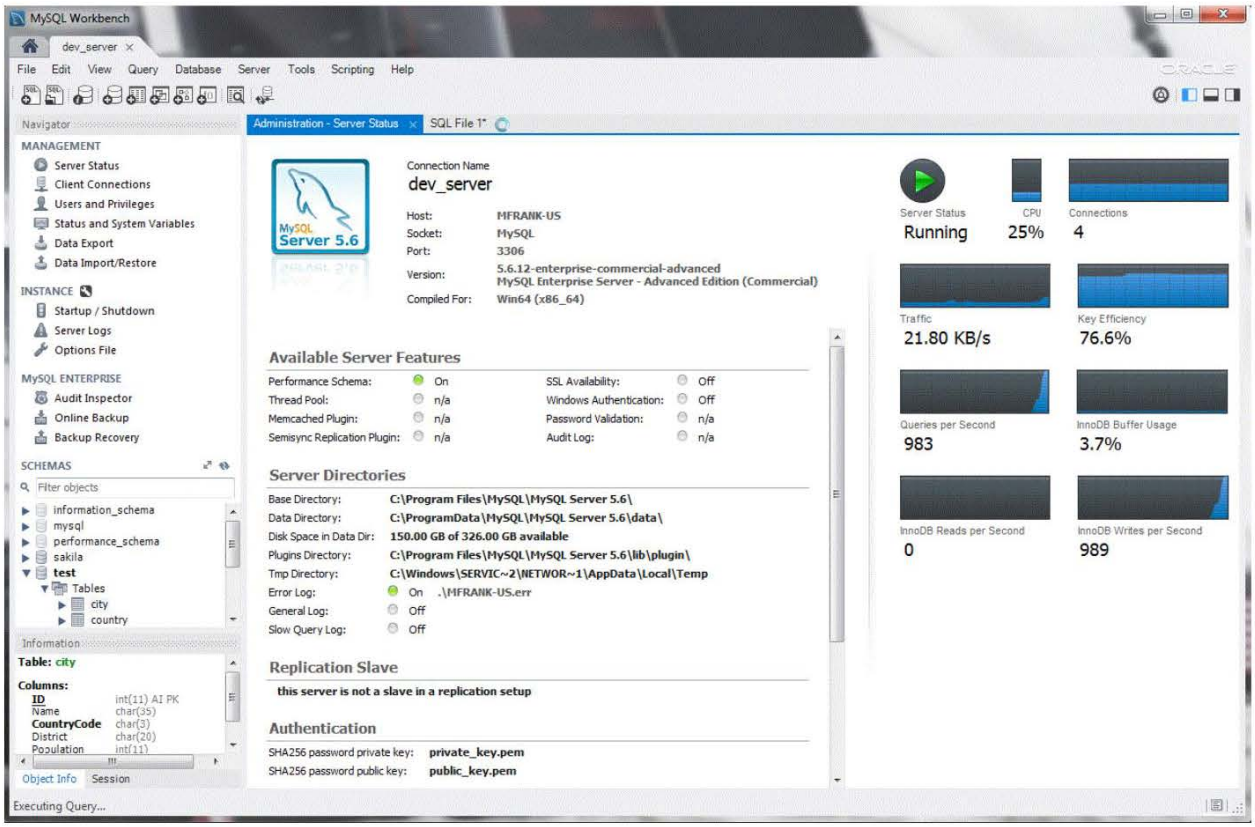

Рисунок 2.5 – Інтерфейс MySQL Workbench

– він має простий у використанні синтаксис SQL і швидко виконує запити до бази даних;

– MySQL підтримує багатопотоковий доступ і може обробляти багато одночасних підключень, що робить його практичним для веб-сайтів з великою кількістю користувачів;

– він також має можливість реплікації, що дозволяє створювати резервні копії даних та забезпечувати високу доступність системи.SQLite:

– SQLite є легковагою вбудованою базою даних, яка не вимагає окремого сервера. Вона зберігає всю інформацію у вигляді одного файлу бази даних;

– SQLite підтримує багато мов програмування, включаючи C#, що робить його зручним для використання в програмному забезпеченні;

– вона проста у використанні і не вимагає значних системних ресурсів, що дозволяє використовувати її на різних платформах, включаючи мобільні пристрої;

– SQLite підтримує транзакції, що дозволяє забезпечити цілісність даних і відновлення в разі помилок.

Ці бази даних можуть використовуватись для зберігання, організації та керування даними про автобуси, квитки та розклад руху у системі автоматизації приміського вокзалу. Вибір конкретної бази даних залежить від потреб проекту, обсягу даних, швидкодії та інших вимог.

2.7 Архітектурний підхід

Архітектурний підхід в розробці програмного забезпечення визначає організацію компонентів системи, спосіб взаємодії між ними та розподіл функціональності для досягнення поставлених цілей. Один з популярних архітектурних підходів, які можна використовувати для розробки додатку, це Model-View-Controller (MVC) і Model-View-ViewModel (MVVM).

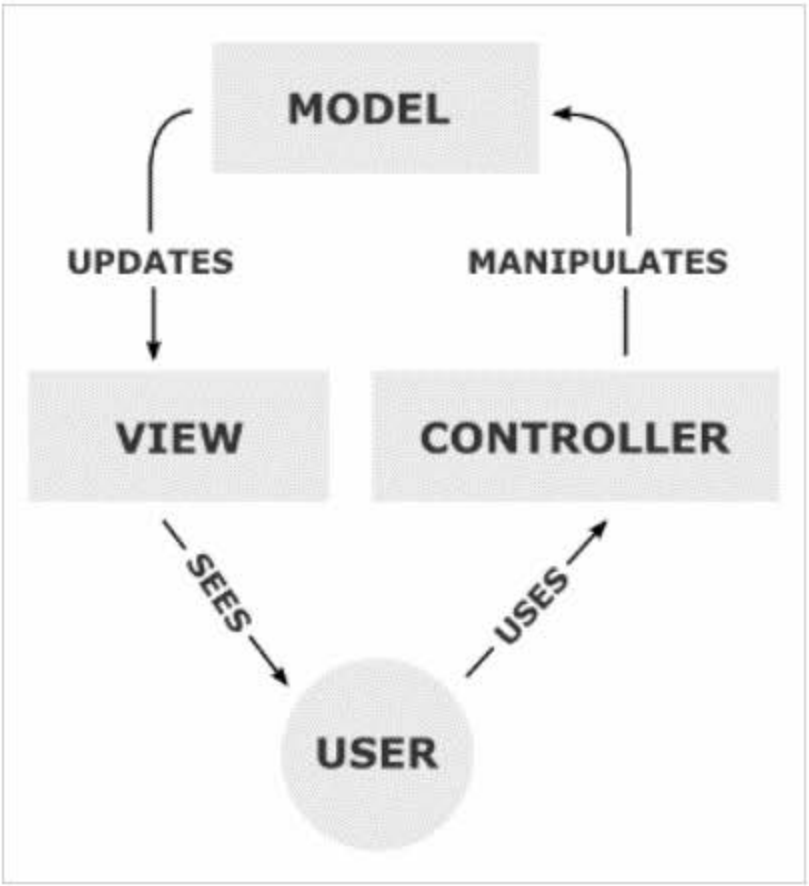

Рисунок 2.6 – Схема роботи Model-View-Controller

Model-View-Controller (MVC):

– MVC є шаблоном проектування, що розділяє компоненти системи на три основні частини. Model (модель), View (вид) і Controller (контролер);

– модель відповідає за управління даними та бізнес-логікою додатку. Вона представляє структуру даних і оперує ними без прив'язки до способу їх відображення;

– вид відповідає за представлення даних користувачеві і інтерфейс взаємодії з користувачем. Він відображає дані, отримані від моделі, і реагує на події користувача;

– контролер діє як посередник між моделлю і видом. Він обробляє вхідні дані від користувача, оновлює модель і забезпечує оновлення вигляду відповідно до змін в моделі.

Model-View-ViewModel (MVVM):

– MVVM є архітектурним підходом, який розширює концепцію MVC. Він включає три основні компоненти. Model (модель), View (вид) і ViewModel (подання моделі);

– модель відповідає за управління даними і бізнес-логікою, аналогічно до MVC;

– вид відображає дані користувачеві і реагує на події, аналогічно до MVC;

– ViewModel діє як посередник між моделлю і видом. Він забезпечує прив'язку даних між моделлю і видом, а також реагує на події користувача. Він дозволяє відокремити логіку відображення від бізнес-логіки та спрощує тестування і розширення додатку.

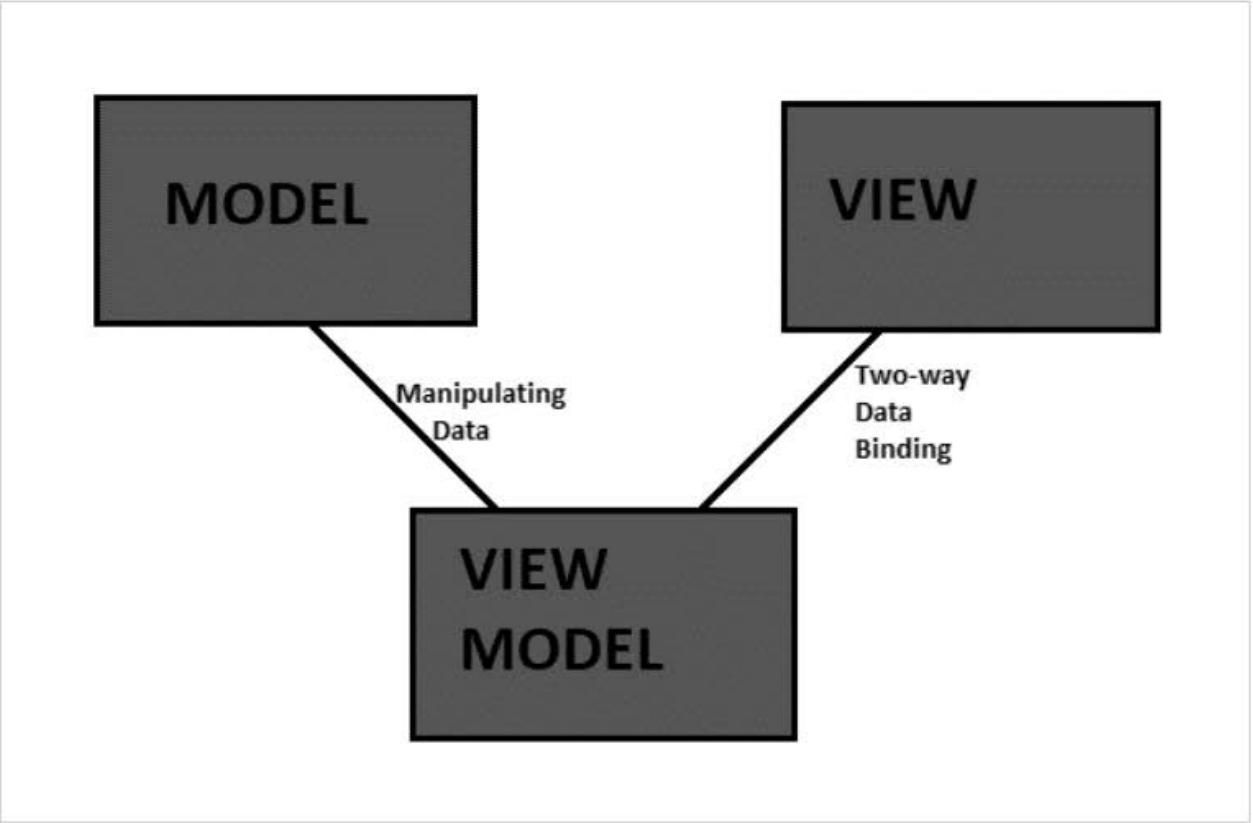

Рисунок 2.7 – Схема роботи Model-View-ViewModel

Якщо використовувати MVC або MVVM для розробки застосунку для автоматизації роботи приміського вокзалу, це дозволить відокремити логіку додатку від його графічного інтерфейсу. Це дасть можливість легко модифікувати і розширювати функціонал, забезпечить кращу перевикористовуваність коду і зробить його більш організованим і зрозумілим для розробників.

2.8 Висновок до другого розділу

У розділі 2 було проведено аналіз засобів розроблення програмного забезпечення для автоматизації роботи приміського вокзалу. Було розглянуто різні аспекти, які мають важливе значення для розробки такого застосунку.

Вибір програмних засобів є критичним етапом в розробці. Було проведено огляд та обгрунтування вибору програмних засобів, таких як Windows Forms, мова програмування C#, Visual Studio і .NET Framework. Ці

засоби є добре встановленими і популярними в середовищі розробки програмного забезпечення та надають потужні інструменти для розробки, тестування і налагодження програм.

Було розглянуто також огляд баз даних і їх роль у збереженні даних про автобуси, квитки та розклад руху. Бази даних, такі як SQL Server, MySQL або SQLite, можуть бути використані для забезпечення надійного збереження даних та ефективного доступу до них.

Також був розглянутий архітектурний підхід для розробки застосунку. Зазначено, що використання певного архітектурного підходу, такого як Model-View-Controller (MVC) або Model-View-ViewModel (MVVM), може допомогти відокремити логіку додатку від його графічного інтерфейсу і спростити розширення та підтримку системи.

Висновок з розділу 2 підкреслює важливість правильного вибору програмних засобів, мови програмування, інтегрованого середовища розробки, а також архітектурного підходу для успішної розробки програмного забезпечення для автоматизації роботи приміського вокзалу. Обрані засоби та підходи повинні відповідати потребам проекту, забезпечувати зручну розробку, масштабованість і підтримку системи.

РОЗДІЛ 3 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ПРИМІСЬКОГО ВОКЗАЛУ

3.1 Принцип роботи програмного забезпечення

Застосунок для приміського вокзалу може мати наступний принцип роботи:

1) графічний інтерфейс. Застосунок має графічний інтерфейс, який надає користувачеві можливість взаємодіяти з різними функціями та операціями. Інтерфейс може бути побудований за допомогою технологій, таких як Windows Forms, WPF або інші, і складатиметься з вікон, кнопок, текстових полів, списків та інших елементів;

2) розклад руху автобусів. Застосунок може містити функціонал для відображення розкладу руху автобусів на приміських маршрутах. Це може включати відображення дат, часів відправлення та прибуття, маршрутів та номерів автобусів. Розклад може бути отриманий з бази даних або зовнішнього джерела, такого як API;

3) пошук рейсів. Користувач може використовувати функцію пошуку, ввівши потрібне місце призначення або інші критерії. Застосунок може шукати в базі даних або звертатися до зовнішнього сервісу, щоб знайти відповідні рейси, які задовольняють критерії пошуку. Результати пошуку можуть бути відображені користувачу;

4) бронювання квитків. Після вибору потрібного рейсу, користувач може здійснити бронювання квитків. Це може включати введення інформації про пасажира (ім'я, контактні дані тощо), вибір місця і оплату. Інформація про бронювання може бути збережена в базі даних для майбутнього використання;

5) видалення квитків. Застосунок може надавати можливість видалення квитків, які були раніше заброньовані. Користувач може ввести номер квитка або інші ідентифікуючі дані, і застосунок видаляє відповідний запис з бази даних;

6) пошук квитків. Застосунок може мати функцію пошуку квитків за певними критеріями, наприклад, за номером квитка, ім'ям пасажира або іншими даними. Це дозволяє користувачу знайти і переглянути інформацію про певні квитки;

7) обробка помилок та повідомлення. Застосунок може містити обробку помилок та виняткових ситуацій, таких як невірний ввід даних або проблеми з підключенням до бази даних. При виникненні помилки або непередбаченої ситуації застосунок може вивести відповідне повідомлення для користувача;

8) інтернаціоналізація та локалізація. Застосунок може мати можливість підтримувати різні мови та культури. Це означає, що інтерфейс може бути локалізований, а текстові повідомлення та інші елементи можуть бути відображені відповідно до обраної мови користувача.

Це лише загальний опис можливого принципу роботи застосунку. Реальна реалізація може залежати від конкретних вимог, архітектурних рішень та використаних технологій.

3.2 Опис архітектури проекту

Детальний опис структури проекту для програми приміського вокзалу:

1) MainForm.cs. Це основний клас форми, який містить головне вікно програми та управляє вкладками та їх вмістом;

2) MainForm.Designer.cs. Цей файл містить автоматично згенерований код для інтерфейсу форми. Він містить описи елементів у вікні, їх розташування та налаштування;

3) MainForm.resx. Цей файл містить ресурси форми, такі як тексти, зображення та інші ресурси, які використовуються в програмі;

4) Program.cs. Це клас Program, який містить метод Main(). Цей метод є входом у програму і створює та запускає головну форму програми;

5) SuburbanStationApp.csproj. Це файл проекту, який містить налаштування та інформацію про проект. Він визначає залежності, включає файли проекту та налаштування компіляції;

6) Properties (папка):

– AssemblyInfo.cs. Цей файл містить інформацію про версію та метадані програми;

– Resources.Designer.cs. Цей файл містить згенерований код для ресурсів програми;

– Resources.resx. Цей файл містить ресурси, такі як тексти, зображення та інші ресурси, які використовуються в програмі;

– Settings.Designer.cs. Цей файл містить згенерований код для налаштувань програми;

– Settings.settings. Цей файл містить налаштування програми.

Це загальна структура проекту для програми приміського вокзалу, створена за допомогою Windows Forms у мові C#. Залежно від вимог, може бути додано додаткові файли, класи та ресурси.

3.3 Опис структури проекту

Детальний опис архітектури проекту для програми приміського вокзалу:

Представлення (Presentation Layer): MainForm. Це головна форма програми, яка відображає графічний інтерфейс користувача. Вона містить вкладки і елементи управління, такі як кнопки, текстові поля, мітки та інші. Взаємодія з користувачем відбувається через цей шар.

Логіка додатку (Application Logic):

– MainForm.cs. Цей клас форми містить логіку відображення вкладок, обробники подій кнопок та інші функції, пов'язані з керуванням інтерфейсом;

– TicketManager.cs. Цей клас відповідає за управління квитками. Він містить функції для створення, видалення, пошуку та покупки квитків.

Взаємодія з базою даних або іншими джерелами даних може здійснюватись через цей клас;

– ScheduleManager.cs. Цей клас відповідає за управління розкладом руху автобусів. Він містить функції для отримання розкладу, оновлення розкладу та пошуку рейсів за заданими критеріями. Взаємодія з базою даних або зовнішніми сервісами може здійснюватись через цей клас.

Дані (Data Layer):

– Ticket.cs. Цей клас представляє модель квитка і містить властивості, такі як номер квитка, номер автобуса, місце та інші деталі квитка;

– Schedule.cs. Цей клас представляє модель розкладу руху автобусів і містить властивості, такі як місце відправлення, місце прибуття, час відправлення та інші деталі рейсу;

– Database.cs. Цей клас відповідає за зберігання та отримання даних з бази даних. Він містить функції для додавання, видалення, оновлення та отримання квитків та розкладу з бази даних. .

Інфраструктура (Infrastructure): DatabaseConnection.cs. Цей клас встановлює з'єднання з базою даних та забезпечує функціональність для виконання запитів до бази даних.

Це загальна архітектура проекту, яка розділяє функціональність на різні шари, такі як представлення, логіку додатку, роботу з даними та інфраструктуру.

Це дозволяє підтримувати чистоту коду, полегшує розширення та тестування програми. Залежно від потреб та специфіки проекту, можуть бути додані додаткові класи та модулі для реалізації необхідного функціоналу.

3.4 Проектування бази даних

Детальний опис проектування бази даних для програми приміського вокзалу:

Сутності (Entities):

– квиток (Ticket). Ця сутність представляє квиток на автобусний рейс. Може містити такі атрибути, як унікальний ідентифікатор квитка, номер автобуса, місце, дата та час відправлення, дата та час прибуття, інформація про пасажира і інші відомості, які вважаєте необхідними;

– розклад (Schedule). Ця сутність представляє розклад руху автобусних рейсів. Може містити такі атрибути, як унікальний ідентифікатор розкладу, місце відправлення, місце прибуття, дата та час відправлення, дата та час прибуття, інформація про автобус і інші відомості, які вважаєте необхідними.

Зв'язки (Relationships): квиток може бути пов'язаний з розкладом. Один розклад може мати багато квитків, але кожен квиток може бути пов'язаний тільки з одним розкладом. Це зв'язок один до багатьох між сутностями «Розклад» та «Квиток».

На основі вищеописаних сутностей та зв'язків, можна створити такі таблиці у базі даних:

– Tickets. Таблиця, що містить дані про квитки. Може містити стовпці для унікального ідентифікатора квитка, номеру автобуса, місця, дати та часу відправлення, дати та часу прибуття, інформації про пасажира та інших відомостей, які вважаються за потрібне;

– Schedules. Таблиця, що містить дані про розклади. Може містити стовпці для унікального ідентифікатора розкладу, місця відправлення, місця прибуття, дати та часу відправлення, дати та часу прибуття, інформації про автобус та інших відомостей, які вважаються за потрібне.

Запити (Queries):

У програмі можуть бути різні типи запитів до бази даних, такі як:

– додавання нового квитка до бази даних;

– видалення квитка з бази даних;

– оновлення інформації про квиток;

– отримання списку квитків за певними критеріями (наприклад, за датою відправлення або номером автобуса).

Це загальний опис проектування бази даних для програми приміського вокзалу. Залежно від конкретного проекту та вимог, можуть бути внесені зміни та додані додаткові таблиці та поля в базу даних.

3.5 Тестування програмного забезпечення для автоматизації роботи приміського вокзалу

На рисунку 3.1 зображена головна екранна форма програмного забезпечення приміського вокзалу. Вона містить різні елементи інтерфейсу, такі як кнопки, поля введення та вкладки, які дозволяють користувачеві взаємодіяти з різними функціями програми.

Рисунок 3.2 відображає екранну форму, де користувач може здійснити пошук доступних автобусних рейсів. Форма містить поля введення для вказання місця відправлення та прибуття, дати та інших параметрів, за якими користувач шукає рейси.

На рисунку 3.3 показана екранна форма, де користувач може придбати квиток на обраний рейс. Форма містить поля введення для вказання інформації про пасажира, таку як ім'я, прізвище та інші дані, необхідні для оформлення квитка.

На рисунку 3.4 показана екранна форма, де користувач заповнює необхідні дані для покупки квитка. Вона містить поля введення для вказання інформації про пасажира, такі як ім’я, прізвище, номер телефону та інші деталі.

Рисунок 3.5 показує результат покупки квитка. Він може відображати підтвердження покупки, номер квитка, інформацію про рейс та інші деталі, які стосуються придбаного квитка.

На рисунку 3.6 показана екранна форма, де користувач може переглянути розклад руху автобусних рейсів. Форма містить список доступних рейсів з відповідною інформацією, такою як місце відправлення, місце прибуття, дата та час відправлення та інші деталі.

Рисунок 3.7 показує результат натискання кнопки для перегляду розкладу руху. Він може відображати список доступних рейсів з відповідною інформацією, яка відповідає вказаним критеріям, наприклад, місцю відправлення та прибуття.

Рисунок 3.8 показує екранну форму, де користувач може видалити квиток з бази даних. Вона містить список доступних квитків та кнопки для видалення конкретного квитка.

На рисунку 3.9 показана екранна форма, де користувач може здійснити пошук квитків за різними критеріями, такими як дата відправлення, місце відправлення, місце прибуття тощо. Форма містить поля введення для вказання критеріїв пошуку.

Рисунок 3.10 відображає результати пошуку квитка. Він може містити список знайдених квитків з відповідною інформацією, такою як рейс, дата та час відправлення та інші деталі, що відповідають вказаним критеріям пошуку.

Ці описи допоможуть краще розуміти, що зображено на кожному рисунку і яка функціональність або інформація відображається на відповідних екранних формах.

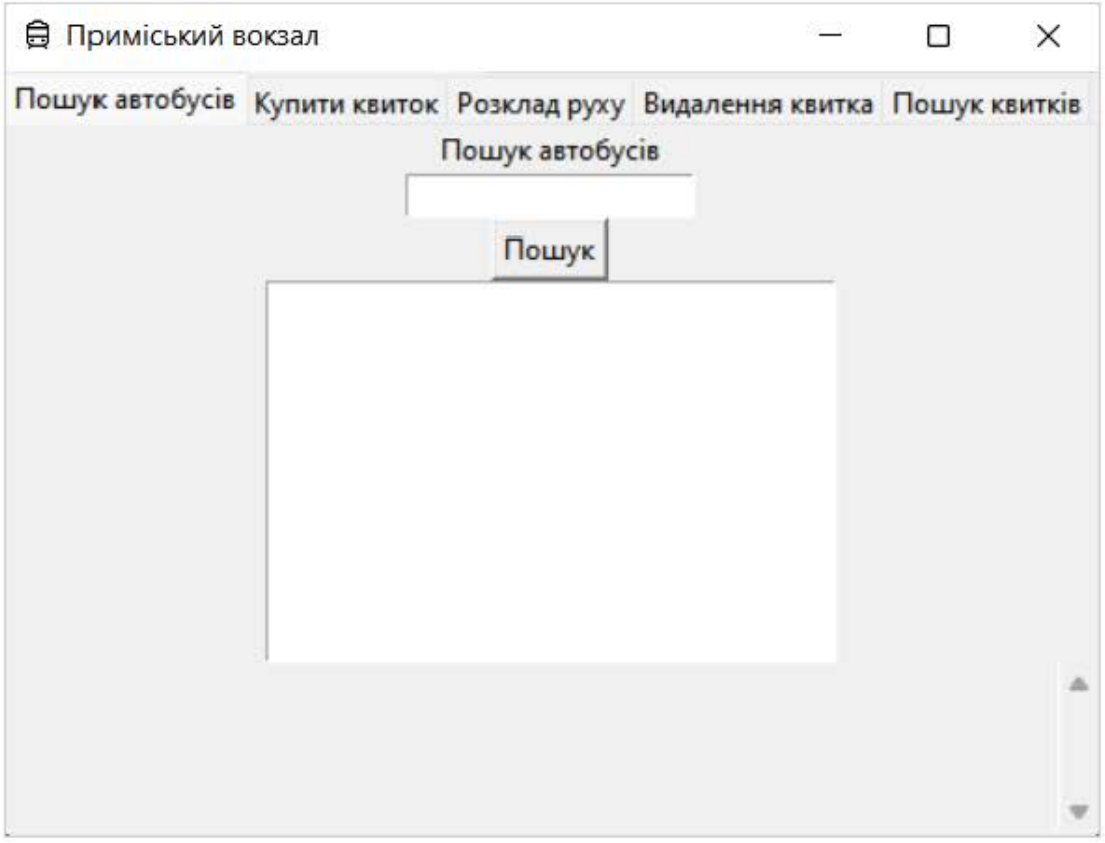

Рисунок 3.1 – Головна екранна форма програмного забезпечення

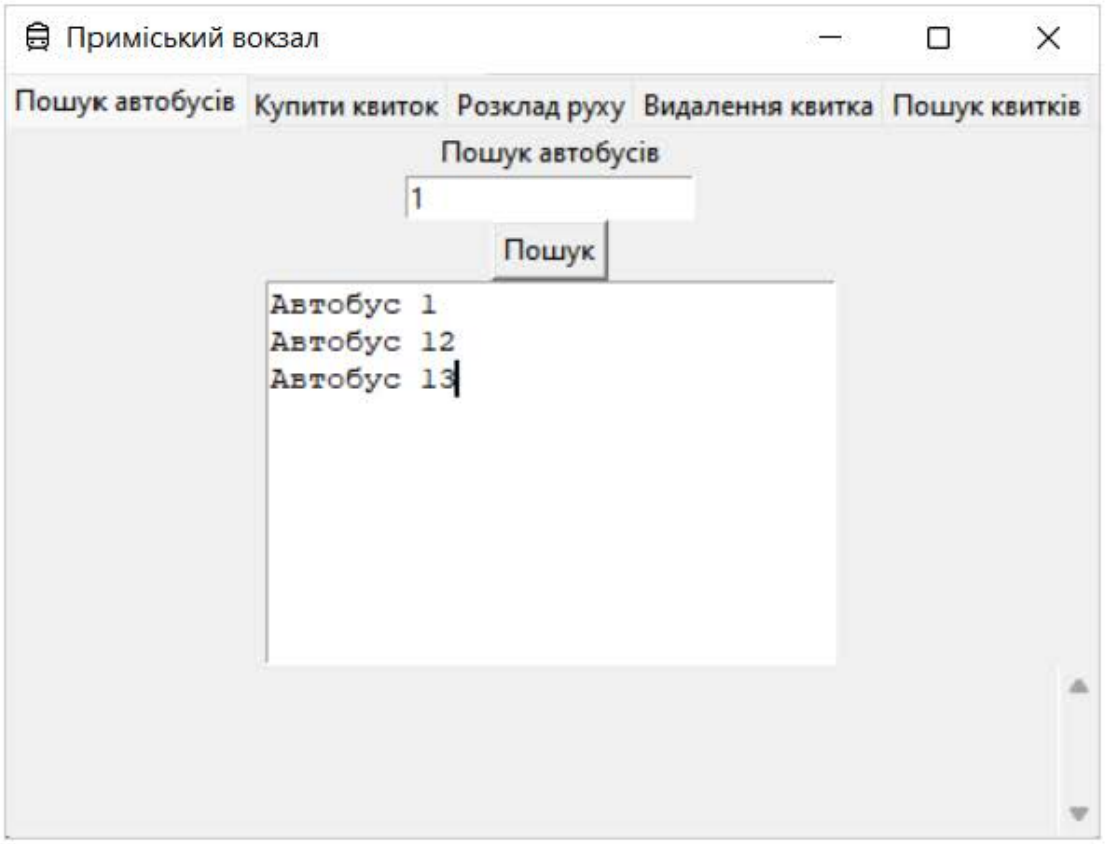

Рисунок 3.2 – Пошук автобусів

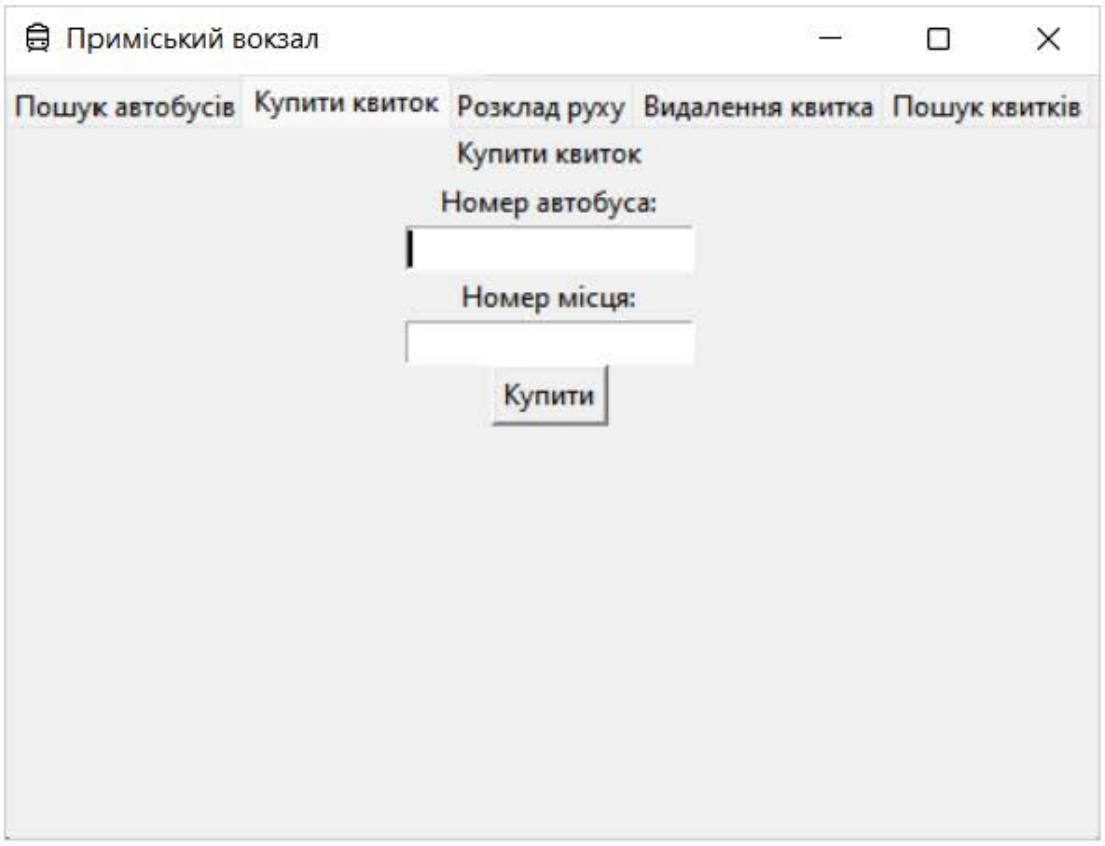

Рисунок 3.3 – Екранна форма покупки квитка

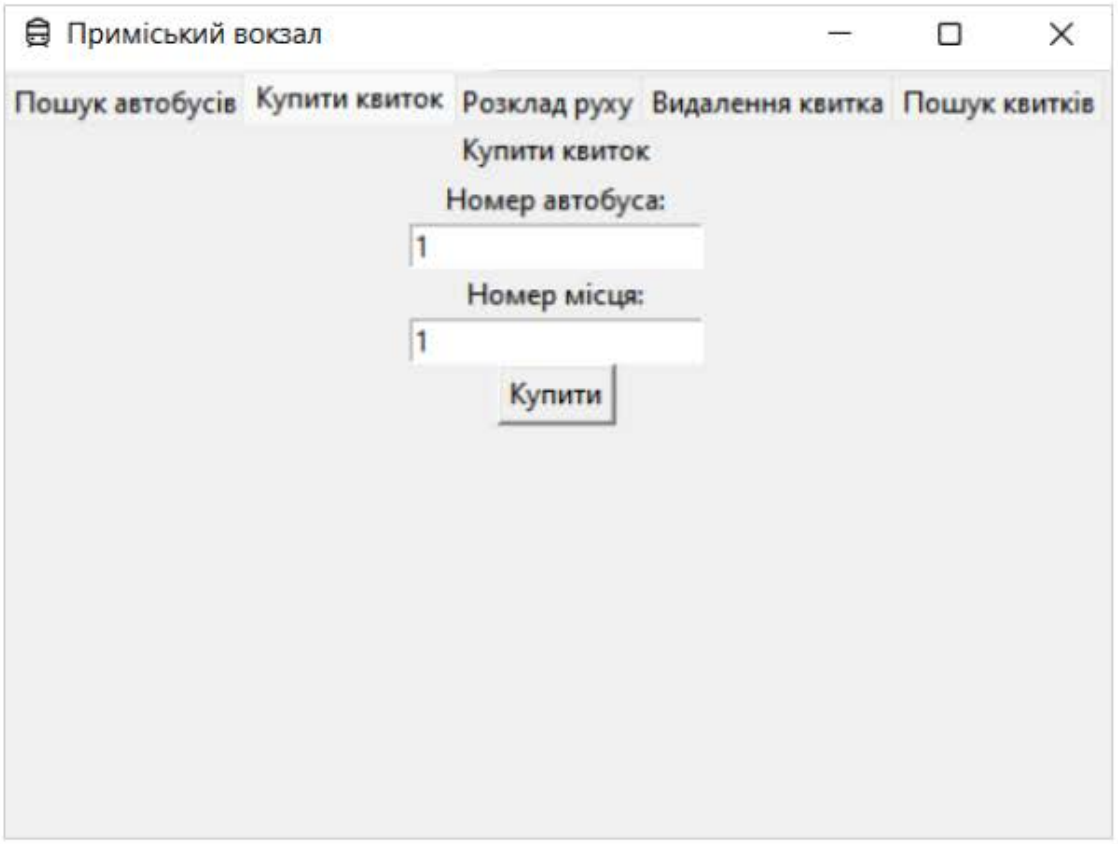

Рисунок 3.4 – Заповнення даних для покупки квитка

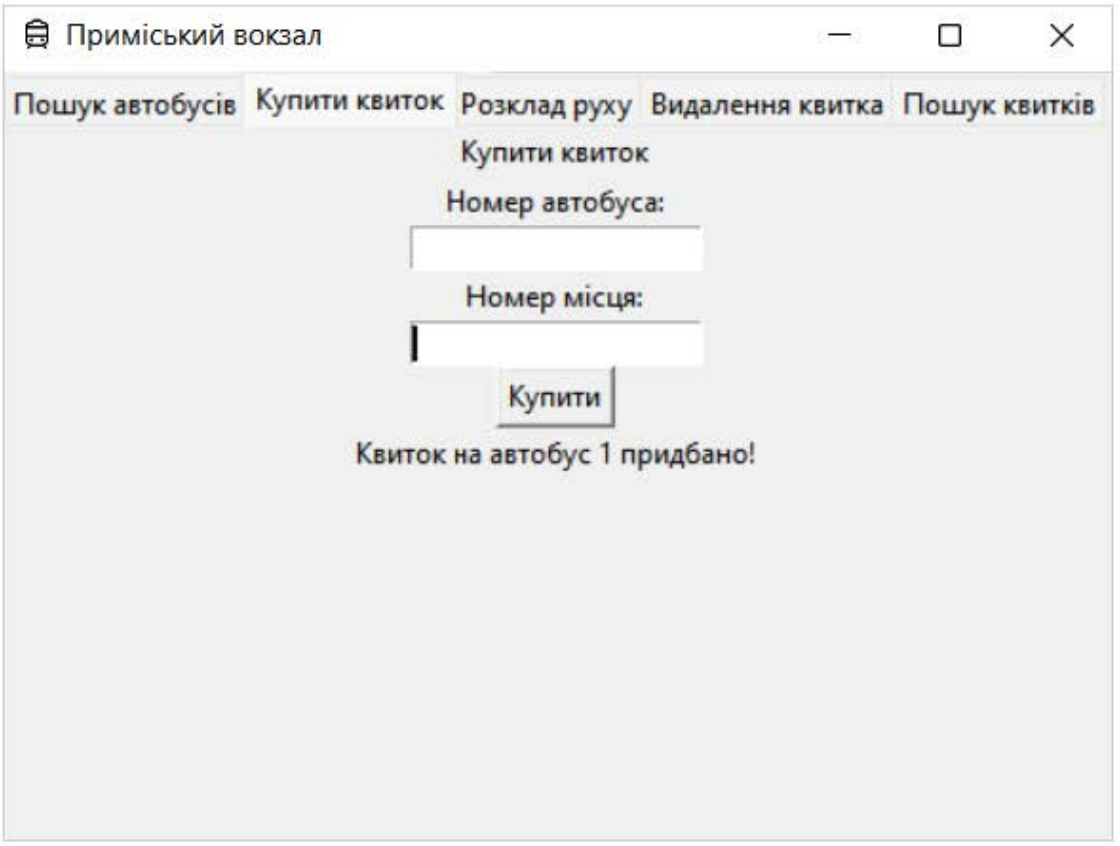

Рисунок 3.5 – Результат покупки квитка

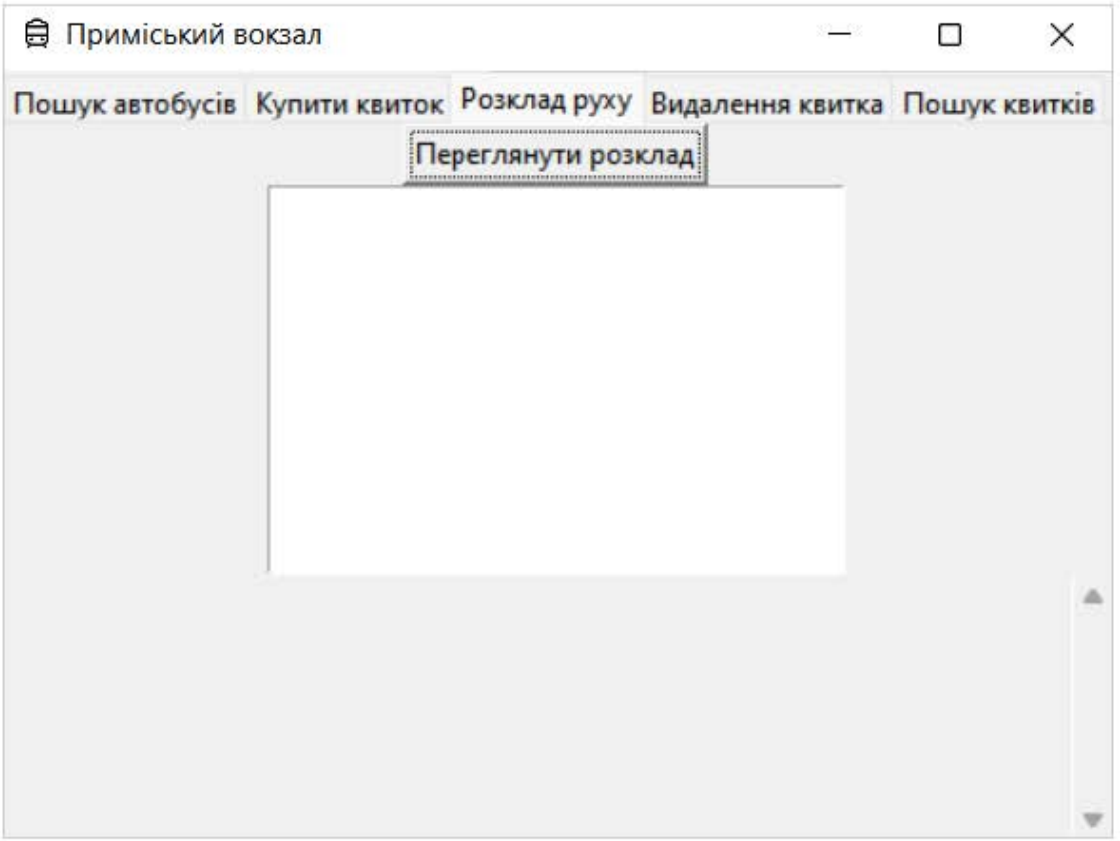

Рисунок 3.6 – Екранна форма розкладу руху

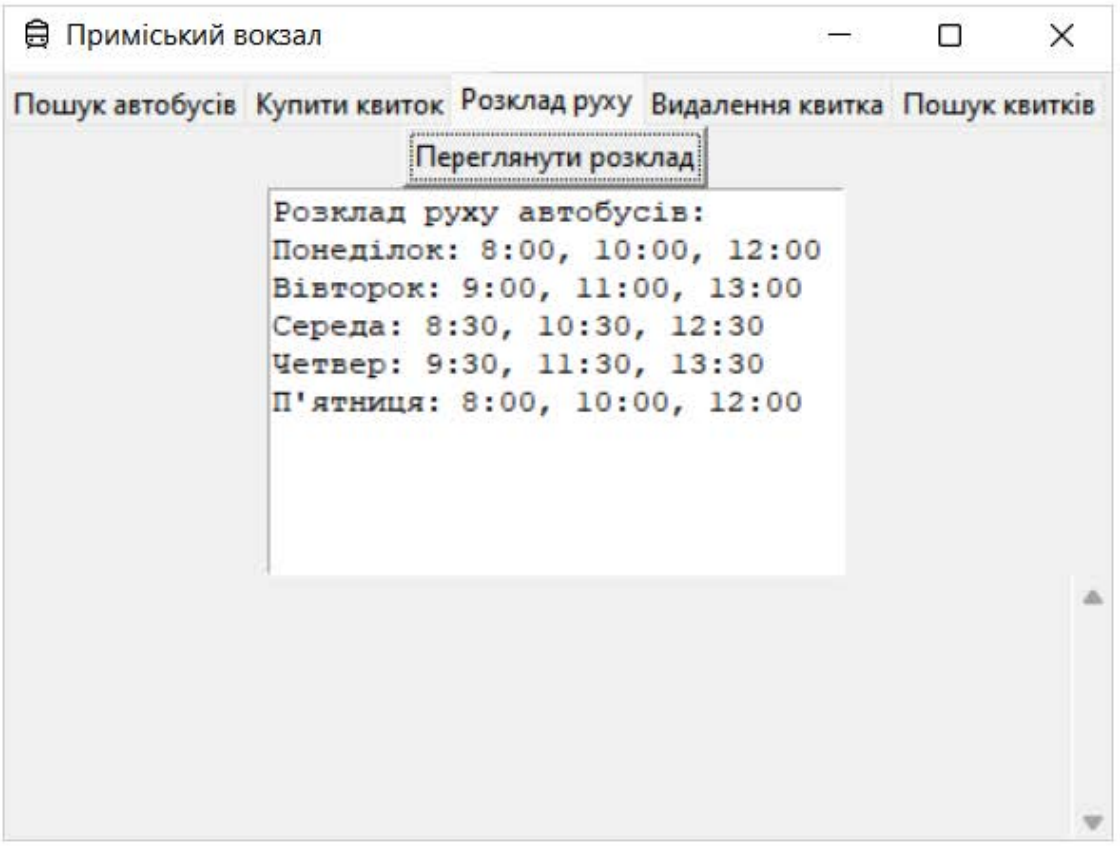

Рисунок 3.7 – Результат натискання кнопки для перегляду розкладу

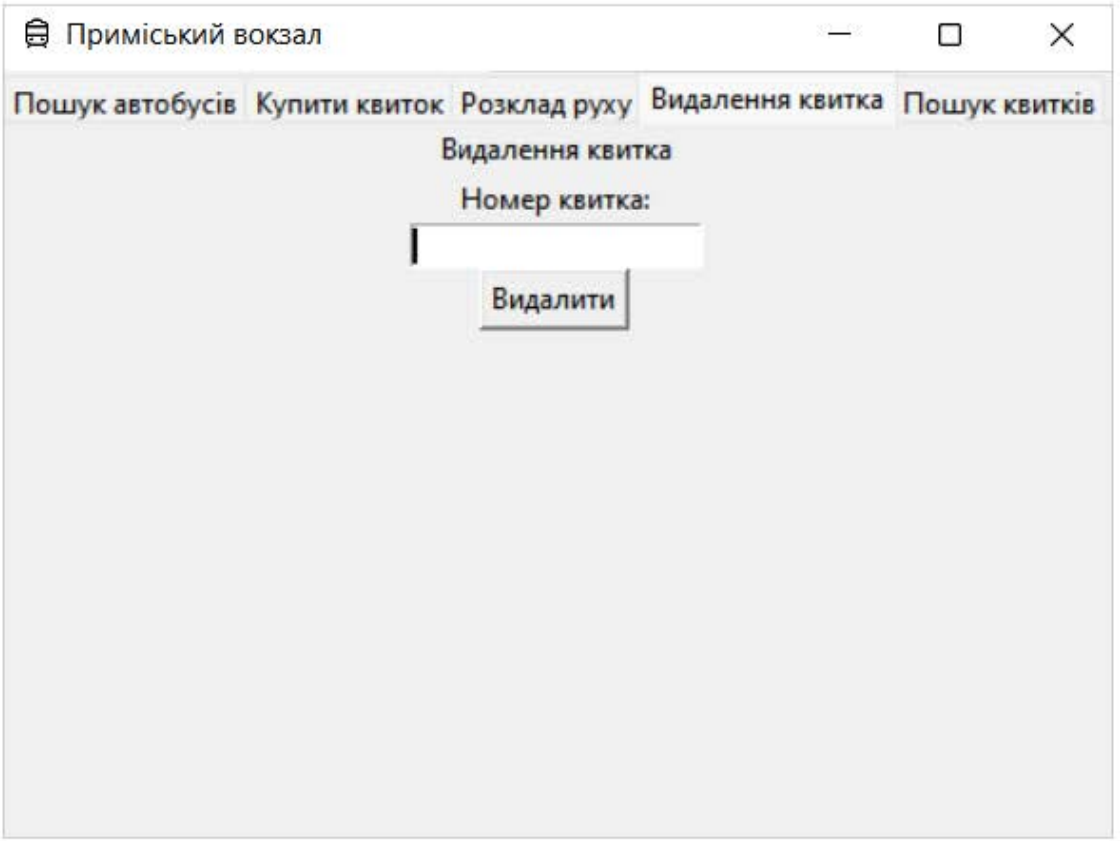

Рисунок 3.8 – Екранна форма видалення квитка

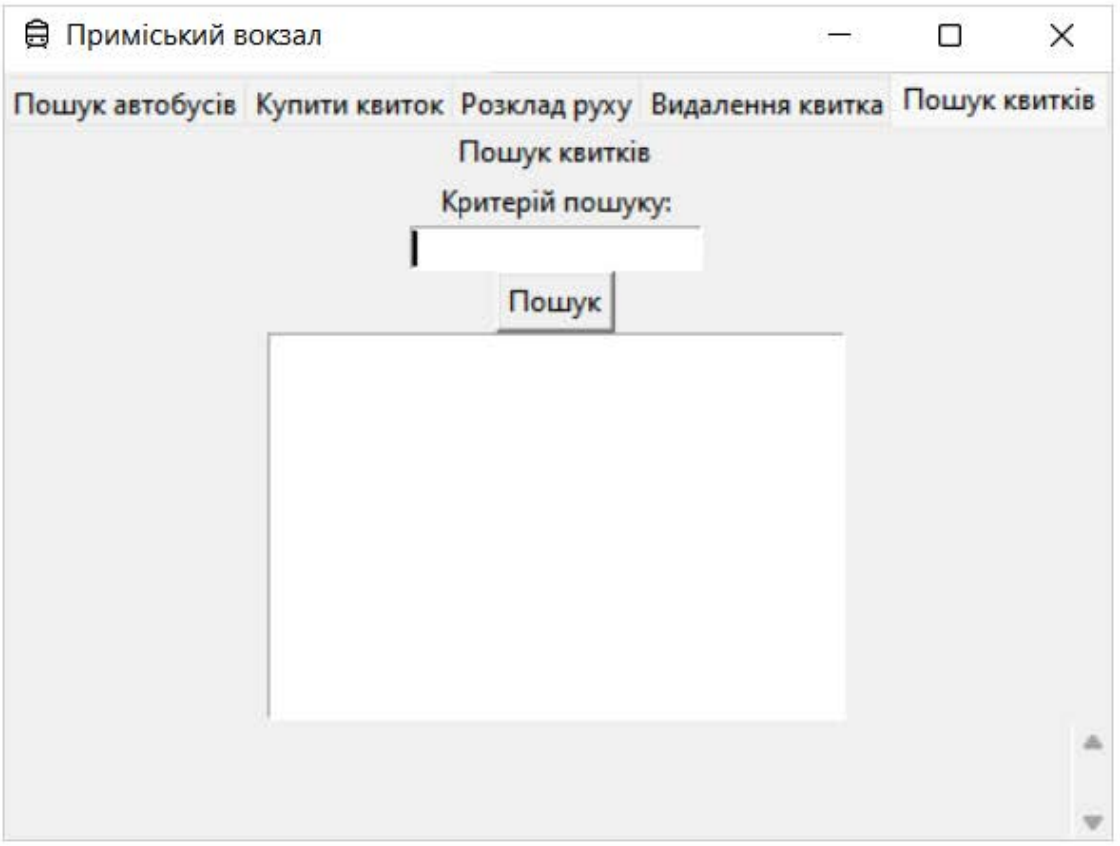

Рисунок 3.9 – Екранна форма пошуку квитків

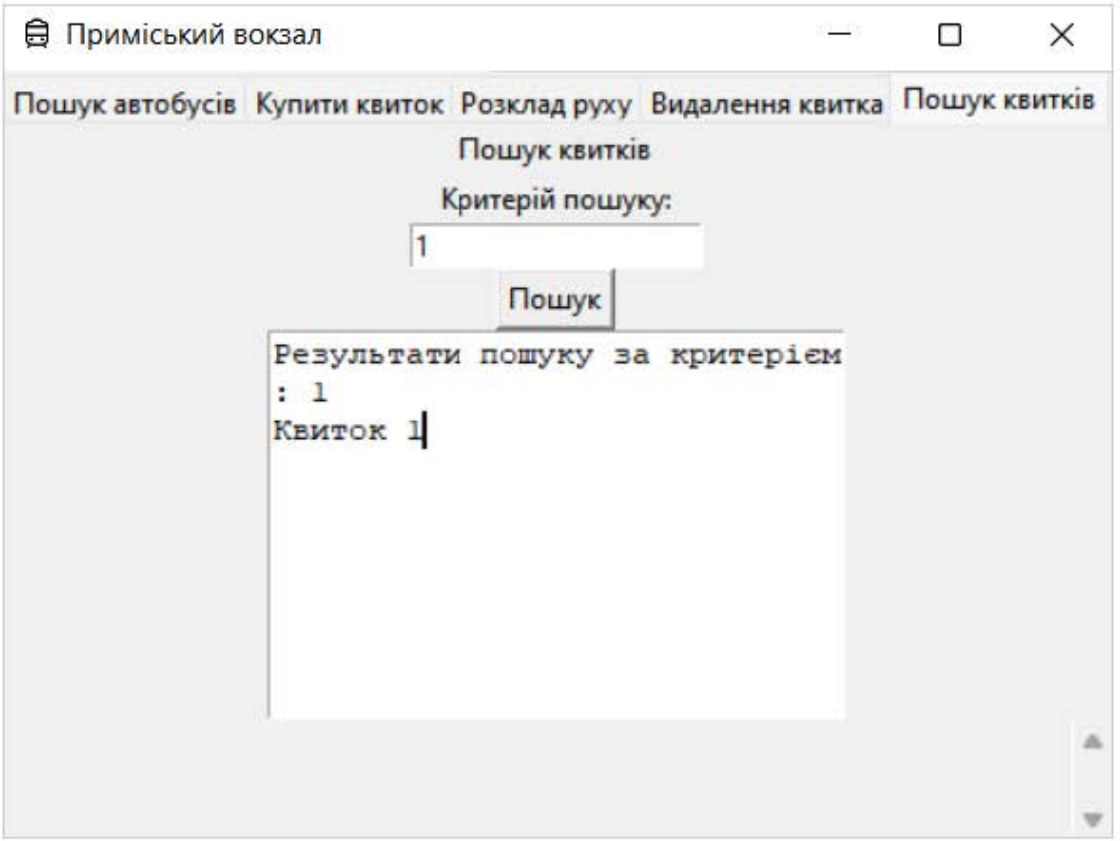

Рисунок 3.10 – Результати пошуку квитка

3.6 Висновки до третього розділу

У третьому розділі було розглянуто розроблення програмного забезпечення для автоматизації роботи приміського вокзалу. Було представлено принцип роботи програмного забезпечення, описано архітектуру проекту та структуру проекту.

Принцип роботи програмного забезпечення полягає у забезпеченні функціональності для пошуку рейсів, покупки квитків, перегляду розкладу руху та видалення квитків.

Програма надає зручний інтерфейс для користувача, що дозволяє легко взаємодіяти з функціями.

Структура проекту включає різні компоненти, такі як класи для взаємодії з базою даних, логіку бізнес-процесів, а також реалізацію графічного інтерфейсу.

Код проекту організований з урахуванням принципів модульності та читабельності, що сприяє його підтримці та розширенню в майбутньому.

Проектування бази даних передбачає використання таблиць для зберігання інформації про розклади, квитки та інші деталі. Запити до бази даних дозволяють виконувати додавання, видалення, оновлення та пошук даних залежно від потреб користувача.

Тестування програмного забезпечення було проведено для перевірки правильності роботи функціоналу. Застосовувалися різні тестові сценарії та вхідні дані для оцінки працездатності та надійності програми.

Загальними висновками до третього розділу можна зазначити, що розроблене програмне забезпечення забезпечує зручну і ефективну автоматизацію роботи приміського вокзалу.

Воно дозволяє користувачам здійснювати пошук рейсів, покупку квитків, перегляд розкладу та видалення квитків з легкістю та швидкістю.

Крім того, проектування бази даних враховує потреби програми та забезпечує ефективне зберігання та обробку інформації.

ВИСНОВКИ

Метою роботи було розроблення програмного забезпечення для автоматизації роботи приміського вокзалу.

Для досягнення поставленої мети розроблення програмного забезпечення для автоматизації роботи приміського вокзалу ставилися наступні задачі:

1) аналіз вимог та потреб. Проведення детального аналізу потреб вокзалу та пасажирів, виявлення основних проблем, які потрібно вирішити через автоматизацію. Це включає збір вимог щодо функціональності системи, інтерфейсу користувача, безпеки, звітності та інших аспектів;

2) проектування архітектури системи. Розроблення високорівневої архітектури програмного забезпечення, яка відповідає вимогам вокзалу та забезпечує ефективну роботу системи. Це включає визначення модулів, компонентів, взаємодій між ними та даних, які будуть використовуватися;

3) розробка функціональності. Реалізація основних функціональних можливостей системи, таких як онлайн-продаж квитків, інформаційна система, електронні табло, анонси тощо. Це включає розробку логіки програми, баз даних, інтерфейсу користувача та інші необхідні компоненти;

4) інтеграція з існуючими системами. У випадку, якщо вокзал вже використовує певні існуючі системи (наприклад, система квиткового обліку), потрібно забезпечити інтеграцію нової системи з цими вже існуючими системами, щоб забезпечити безперебійну роботу та обмін необхідною інформацією;

5) тестування та валідація. Проведення тестування програмного забезпечення для перевірки правильності його роботи, виявлення й виправлення помилок та недоліків. Це може включати функціональне тестування, тестування виконання, тестування безпеки та інші види тестування, які забезпечують якість та надійність системи;

6) впровадження та підтримка. Реалізація процесу впровадження системи на вокзалі, навчання персоналу роботі з новою системою, а також надання підтримки та технічної допомоги після впровадження. Це дозволяє забезпечити успішне використання системи та вирішення потенційних проблем.

Виконання цих задач дозволило досягти мети розроблення програмного забезпечення для автоматизації роботи приміського вокзалу та забезпечити його ефективну та безперебійну роботу, поліпшення якості обслуговування пасажирів та оптимізацію робочих процесів.

Об'єктом дослідження були процеси роботи програмного забезпечення для автоматизації роботи приміського вокзалу.

Предметом дослідження було апаратно-програмне забезпечення для розроблення програмного забезпечення для автоматизації роботи приміського вокзалу.

Практичне значення одержаних результатів полягає у підвищенні якості автоматизації роботи приміського вокзалу.

Результатами роботи є програмне забезпечення для автоматизації роботи приміського вокзалу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yuen S. Mastering Windows Presentation Foundation: Build responsive UIs for desktop applications with WPF, 2nd Edition, 2020.

2. Petzold C. Programming Windows: Writing Windows 8 Apps With C# and XAML, 2013.

3. Liberty J., MacDonald M. Learning C# by Developing Games with Unity: Create your first 2D and 3D games for web, desktop, and mobile, 2019.

4. Richter J. CLR via C#: Applied Microsoft .NET Framework, 4th Edition, 2012.

5. Freeman A., Robson S. Pro C# 7: With .NET and .NET Core, 8th Edition, 2017.

6. Документація Windows Forms. URL: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/

7. Freeman A., Robson S. Pro WPF in C# 2008: Windows Presentation Foundation with .NET 3.5, 2nd Edition, 2008.

8. Sells C., Weinhardt C. Programming Windows Presentation Foundation, 2005.

9. MacDonald M. Pro .NET 2.0 Windows Forms and Custom Controls in C#, 2005.

10. Petzold C. Applications = Code + Markup: A Guide to the Microsoft Windows Presentation Foundation, 2006.

11. Rector M., Sells C., Weinhardt C. Essential Windows Presentation Foundation (WPF), 2007.

12. Balena F. Programming Microsoft Windows Forms, 2005.

13. Troelsen A. Pro C# 2008 and the .NET 3.5 Platform, 4th Edition, 2007.

14. Naucke S. Building Windows Forms Applications: Using C#, 2003.

ДОДАТОК А

ЛІСТИНГ КОДУ ДЛЯ ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

```
namespace SuburbanStationApp
{
   partial class MainForm
   {
      private System.ComponentModel.IContainer components = null;

      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      private void InitializeComponent()
      {
         this.tabControl = new System.Windows.Forms.TabControl();
         this.searchTabPage = new System.Windows.Forms.TabPage();
         this.searchButton = new System.Windows.Forms.Button();
         this.destinationLabel = new System.Windows.Forms.Label();
         this.destinationTextBox = new System.Windows.Forms.TextBox();
         this.resultTextBox = new System.Windows.Forms.TextBox();
         this.buyTicketTabPage = new System.Windows.Forms.TabPage();
         this.confirmationLabel = new System.Windows.Forms.Label();
```

```
this.buyButton = new System.Windows.Forms.Button();
this.seatNumberTextBox = new System.Windows.Forms.TextBox();
this.seatNumberLabel = new System.Windows.Forms.Label();
this.busNumberTextBox = new System.Windows.Forms.TextBox();
this.busNumberLabel = new System.Windows.Forms.Label();
this.scheduleTabPage = new System.Windows.Forms.TabPage();
this.scheduleTextBox = new System.Windows.Forms.TextBox();
this.scheduleButton = new System.Windows.Forms.Button();
this.deleteTicketTabPage = new System.Windows.Forms.TabPage();
this.deleteButton = new System.Windows.Forms.Button();
this.ticketNumberTextBox = new System.Windows.Forms.TextBox();
this.ticketNumberLabel = new System.Windows.Forms.Label();
this.searchTicketsTabPage = new System.Windows.Forms.TabPage();
this.searchTicketsButton = new System.Windows.Forms.Button();
this.searchCriteriaTextBox = new System.Windows.Forms.TextBox();
this.searchCriteriaLabel = new System.Windows.Forms.Label();
this.tabControl.SuspendLayout();
this.searchTabPage.SuspendLayout();
this.buyTicketTabPage.SuspendLayout();
this.scheduleTabPage.SuspendLayout();
this.deleteTicketTabPage.SuspendLayout();
this.searchTicketsTabPage.SuspendLayout();
this.SuspendLayout();

// Код для налаштування елементів інтерфейсу тут...

this.tabControl.Controls.Add(this.searchTabPage);
this.tabControl.Controls.Add(this.buyTicketTabPage);
this.tabControl.Controls.Add(this.scheduleTabPage);
this.tabControl.Controls.Add(this.deleteTicketTabPage);
```

```
    this.tabControl.Controls.Add(this.searchTicketsTabPage);
    this.tabControl.Location = new System.Drawing.Point(12, 12);
    this.tabControl.Name = "tabControl";
    this.tabControl.SelectedIndex = 0;
    this.tabControl.Size = new System.Drawing.Size(400, 300);
    this.tabControl.TabIndex = 0;

    // Додавання вкладок та елементів у вікно

    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(424, 331);
    this.Controls.Add(this.tabControl);
    this.Name = "MainForm";
    this.Text = "Приміський вокзал";
    this.tabControl.ResumeLayout(false);
    this.searchTabPage.ResumeLayout(false);
    this.searchTabPage.PerformLayout();
    this.buyTicketTabPage.ResumeLayout(false);
    this.buyTicketTabPage.PerformLayout();
    this.scheduleTabPage.ResumeLayout(false);
    this.scheduleTabPage.PerformLayout();
    this.deleteTicketTabPage.ResumeLayout(false);
    this.deleteTicketTabPage.PerformLayout();
    this.searchTicketsTabPage.ResumeLayout(false);
    this.searchTicketsTabPage.PerformLayout();
    this.ResumeLayout(false);
}

private System.Windows.Forms.TabControl tabControl;
```

```
private System.Windows.Forms.TabPage searchTabPage;
private System.Windows.Forms.TabPage buyTicketTabPage;
private System.Windows.Forms.TabPage scheduleTabPage;
private System.Windows.Forms.TabPage deleteTicketTabPage;
private System.Windows.Forms.TabPage searchTicketsTabPage;
private System.Windows.Forms.Button searchButton;
private System.Windows.Forms.Label destinationLabel;
private System.Windows.Forms.TextBox destinationTextBox;
private System.Windows.Forms.TextBox resultTextBox;
private System.Windows.Forms.Button buyButton;
private System.Windows.Forms.TextBox seatNumberTextBox;
private System.Windows.Forms.Label seatNumberLabel;
private System.Windows.Forms.TextBox busNumberTextBox;
private System.Windows.Forms.Label busNumberLabel;
private System.Windows.Forms.Label confirmationLabel;
private System.Windows.Forms.Button scheduleButton;
private System.Windows.Forms.TextBox scheduleTextBox;
private System.Windows.Forms.Button deleteButton;
private System.Windows.Forms.TextBox ticketNumberTextBox;
private System.Windows.Forms.Label ticketNumberLabel;
private System.Windows.Forms.Button searchTicketsButton;
private System.Windows.Forms.TextBox searchCriteriaTextBox;
private System.Windows.Forms.Label searchCriteriaLabel;
  this.tabControl = new System.Windows.Forms.TabControl();
    this.searchTabPage = new System.Windows.Forms.TabPage();
    this.searchButton = new System.Windows.Forms.Button();
    this.destinationLabel = new System.Windows.Forms.Label();
    this.destinationTextBox = new System.Windows.Forms.TextBox();
    this.resultTextBox = new System.Windows.Forms.TextBox();
    this.buyTicketTabPage = new System.Windows.Forms.TabPage();
```

```
this.confirmationLabel = new System.Windows.Forms.Label();
this.buyButton = new System.Windows.Forms.Button();
this.seatNumberTextBox = new System.Windows.Forms.TextBox();
this.seatNumberLabel = new System.Windows.Forms.Label();
this.busNumberTextBox = new System.Windows.Forms.TextBox();
this.busNumberLabel = new System.Windows.Forms.Label();
this.scheduleTabPage = new System.Windows.Forms.TabPage();
this.scheduleTextBox = new System.Windows.Forms.TextBox();
this.scheduleButton = new System.Windows.Forms.Button();
this.deleteTicketTabPage = new System.Windows.Forms.TabPage();
this.deleteButton = new System.Windows.Forms.Button();
this.ticketNumberTextBox = new System.Windows.Forms.TextBox();
this.ticketNumberLabel = new System.Windows.Forms.Label();
this.searchTicketsTabPage = new System.Windows.Forms.TabPage();
this.searchTicketsButton = new System.Windows.Forms.Button();
this.searchCriteriaTextBox = new System.Windows.Forms.TextBox();
this.searchCriteriaLabel = new System.Windows.Forms.Label();
this.tabControl.SuspendLayout();
this.searchTabPage.SuspendLayout();
this.buyTicketTabPage.SuspendLayout();
this.scheduleTabPage.SuspendLayout();
this.deleteTicketTabPage.SuspendLayout();
this.searchTicketsTabPage.SuspendLayout();
this.SuspendLayout();

// Код для налаштування елементів інтерфейсу тут...

this.tabControl.Controls.Add(this.searchTabPage);
this.tabControl.Controls.Add(this.buyTicketTabPage);
this.tabControl.Controls.Add(this.scheduleTabPage);
```

```
    this.tabControl.Controls.Add(this.deleteTicketTabPage);
    this.tabControl.Controls.Add(this.searchTicketsTabPage);
    this.tabControl.Location = new System.Drawing.Point(12, 12);
    this.tabControl.Name = "tabControl";
    this.tabControl.SelectedIndex = 0;
    this.tabControl.Size = new System.Drawing.Size(400, 300);
    this.tabControl.TabIndex = 0;

    // Додавання вкладок та елементів у вікно

    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(424, 331);
    this.Controls.Add(this.tabControl);
    this.Name = "MainForm";
    this.Text = "Приміський вокзал";
    this.tabControl.ResumeLayout(false);
    this.searchTabPage.ResumeLayout(false);
    this.searchTabPage.PerformLayout();
    this.buyTicketTabPage.ResumeLayout(false);
    this.buyTicketTabPage.PerformLayout();
    this.scheduleTabPage.ResumeLayout(false);
    this.scheduleTabPage.PerformLayout();
    this.deleteTicketTabPage.ResumeLayout(false);
    this.deleteTicketTabPage.PerformLayout();
    this.searchTicketsTabPage.ResumeLayout(false);
    this.searchTicketsTabPage.PerformLayout();
    this.ResumeLayout(false);
}
```

```
        private System.Windows.Forms.TabControl tabControl;
        private System.Windows.Forms.TabPage searchTabPage;
        private System.Windows.Forms.TabPage buyTicketTabPage;
        private System.Windows.Forms.TabPage scheduleTabPage;
        private System.Windows.Forms.TabPage deleteTicketTabPage;
        private System.Windows.Forms.TabPage searchTicketsTabPage;
        private System.Windows.Forms.Button searchButton;
        private System.Windows.Forms.Label destinationLabel;
        private System.Windows.Forms.TextBox destinationTextBox;
        private System.Windows.Forms.TextBox resultTextBox;
        private System.Windows.Forms.Button buyButton;
        private System.Windows.Forms.TextBox seatNumberTextBox;
        private System.Windows.Forms.Label seatNumberLabel;
        private System.Windows.Forms.TextBox busNumberTextBox;
        private System.Windows.Forms.Label busNumberLabel;
        private System.Windows.Forms.Label confirmationLabel;
        private System.Windows.Forms.Button scheduleButton;
        private System.Windows.Forms.TextBox scheduleTextBox;
        private System.Windows.Forms.Button deleteButton;
        private System.Windows.Forms.TextBox ticketNumberTextBox;
        private System.Windows.Forms.Label ticketNumberLabel;
        private System.Windows.Forms.Button searchTicketsButton;
        private System.Windows.Forms.TextBox searchCriteriaTextBox;
        private System.Windows.Forms.Label searchCriteriaLabel;
    }
}
```

ДОДАТОК Б

ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
using System;
using System.Windows.Forms;

namespace SuburbanStationApp
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        // Обробник події кнопки "Пошук"
        private void searchButton_Click(object sender, EventArgs e)
        {
            string destination = destinationTextBox.Text;

            // Код для пошуку автобусів за заданим місцем призначення

            // Видалення попередніх результатів
            resultTextBox.Clear();

            // Вивід результатів пошуку
            resultTextBox.AppendText("Результати пошуку для: " + destination +
"\n");
            resultTextBox.AppendText("Автобус 1\n");
```

```
        resultTextBox.AppendText("Автобус 2\n");
        resultTextBox.AppendText("Автобус 3\n");
    }

    // Обробник події кнопки "Купити квиток"
    private void buyButton_Click(object sender, EventArgs e)
    {
        string busNumber = busNumberTextBox.Text;
        string seatNumber = seatNumberTextBox.Text;

        // Код для обробки купівлі квитка

        // Очищення полів введення
        busNumberTextBox.Clear();
        seatNumberTextBox.Clear();

        // Виведення підтвердження
        confirmationLabel.Text = "Квиток на автобус " + busNumber + "
придбано!";
    }

    // Обробник події кнопки "Переглянути розклад"
    private void scheduleButton_Click(object sender, EventArgs e)
    {
        // Код для отримання розкладу руху автобусів

        // Видалення попередніх результатів
        scheduleTextBox.Clear();

        // Вивід розкладу руху
```

```
    scheduleTextBox.AppendText("Розклад руху автобусів:\n");
    scheduleTextBox.AppendText("Понеділок: 8:00, 10:00, 12:00\n");
    scheduleTextBox.AppendText("Вівторок: 9:00, 11:00, 13:00\n");
    scheduleTextBox.AppendText("Середа: 8:30, 10:30, 12:30\n");
    scheduleTextBox.AppendText("Четвер: 9:30, 11:30, 13:30\n");
    scheduleTextBox.AppendText("П'ятниця: 8:00, 10:00, 12:00\n");
}

// Обробник події кнопки "Видалити квиток"
private void deleteButton_Click(object sender, EventArgs e)
{
    string ticketNumber = ticketNumberTextBox.Text;

    // Код для видалення квитка з бази даних

    // Очищення поля введення
    ticketNumberTextBox.Clear();

    // Виведення підтвердження
    confirmationLabel.Text = "Квиток №" + ticketNumber + " видалено!";
}

// Обробник події кнопки "Пошук квитків"
private void searchTicketsButton_Click(object sender, EventArgs e)
{
    string searchCriteria = searchCriteriaTextBox.Text;

    // Код для пошуку квитків за заданим критерієм

    // Видалення попередніх результатів
```

```
            resultTextBox.Clear();

            // Вивід результатів пошуку
            resultTextBox.AppendText("Результати пошуку за критерієм: " +
searchCriteria + "\n");
            resultTextBox.AppendText("Квитка 1\n");
            resultTextBox.AppendText("Квитка 2\n");
            resultTextBox.AppendText("Квитка 3\n");
        }
    }
}
```