НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ СИСТЕМ

Кафедра інформаційних технологій

КУРСОВА РОБОТА

на тему «Система замовлення залізничних квитків»

Студента 2 курсу групи

спеціальності 126 – «Інформаційні системи»

Прудкого Дмитра Павловича

Керівник: д. Рудніченко М.Д.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали

Одеса – 2024

**ЗАВДАННЯ**

НА КУРСОВУ РОБОТУ

Прудкий Дмитро Павлович АД-221

1. Тема роботи “ Система замовлення залізничних квитків”

2. Термін здачі студентом закінченої роботи 28.05.2024

3. Початкові дані до проекту (роботи Початкові дані генеруються системою автоматично. Вони включають базу даних з інформацією про поїзди, станції, маршрути, розклад руху поїздів, інформацію про користувачів та квитки.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити): вступ, аналіз специфіки предметної області, порівняння з альтернативами, проектування, опис процесу роботи, висновки, перелік посилань, код програми

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень): Схема бази даних (Database Schema) - відображає структуру бази даних, включаючи таблиці, їх поля та зв’язки між таблицями, Ескізи інтерфейсу користувача (UI Wireframes) - візуальне представлення інтерфейсу основних форм додатка, включаючи розташування елементів

управління, Ганта діаграма (Gantt Chart) - Відображає план-графік виконання проекту з розподілом завдань у часі.

Завдання видано 5.03.2024

Завдання прийнято до виконання 10.03.2024

ЗМІСТ

[ВСТУП 1](#_Toc167750784)

[1. АНАЛІЗ СПЕЦИФІКИ ПРЕДМЕТНОЇ ОБЛАСТІ 3](#_Toc167750785)

[1.1 Опис предметної області 3](#_Toc167750786)

[2. ПРОЕКТУВАННЯ СИСТЕМИ 6](#_Toc167750787)

[2.1 Написання user stories та формування backlog'у продукту 6](#_Toc167750788)

[2.2 Backlog продукту 7](#_Toc167750789)

[2.3 Виявлення класів і зв’язків за допомогою CRC-карток 8](#_Toc167750790)

[2.4 UML – діаграми 9](#_Toc167750791)

[3. ЗАВДАННЯ ДО КУРСОВОЇ РОБОТИ 11](#_Toc167750792)

[Частина 1.ОПИС ІНТЕРФЕЙСУ 13](#_Toc167750793)

[3.1 Форма «Main.cs» 13](#_Toc167750794)

[3.2 Форма «Logon.cs» 14](#_Toc167750795)

[3.3 Форма «AddInformation.cs» 16](#_Toc167750796)

[3.4 Форма «Information.cs» 17](#_Toc167750797)

[4.БАЗА ДАНИХ 19](#_Toc167750798)

[4.1 Опис теми роботи 19](#_Toc167750799)

[4.2 Визначення переліку бізнес-функцій системи. Проектування та створення і заповнення бази даних. 20](#_Toc167750800)

[4.3 Організація пошуку у таблицях бази даних з допомогою SQL запитів 23](#_Toc167750801)

[4.4 Розрахунок підсумків у таблицях бази даних з допомогою агрегатних функцій 24](#_Toc167750802)

[4.5 Пошук даних у складених сутностей з допомогою запитів на з’єднання та об’єднання таблиць 25](#_Toc167750803)

[4.6 Модифікація даних з допомогою SQL запитів 25](#_Toc167750804)

[4.7 Пошук даних у складених сутностей з допомогою підпорядкованих запитів. 26](#_Toc167750805)

[4.8 Використання уявлень і табличних виразів для обробки даних з допомогою. 26](#_Toc167750806)

[4.9 Використання параметрів при обробці даних з допомогою збережених процедур. 27](#_Toc167750807)

[4.10 Використання тригерів при обробці даних 28](#_Toc167750808)

[5.ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ 31](#_Toc167750809)

[5.1 Програмна реалізація форми “Main.cs”. 31](#_Toc167750810)

[5.2 Програмна реалізація форми “Logons.cs”. 33](#_Toc167750811)

[5.3 Програмна реалізація форми “ AddInformation.cs”. 35](#_Toc167750812)

[5.4 Програмна реалізація форми “ Information.cs”. 36](#_Toc167750813)

[Частина 2. СТВОРЕННЯ HTML-СТОРІНОК ДО ДОДАТКУ 39](#_Toc167750814)

[6. РЕАЛІЗАЦІЯ ПРОЕКТУ 40](#_Toc167750815)

[6.1 Сторінка «index.html» 40](#_Toc167750816)

[6.2 Сторінка «registration.html» 42](#_Toc167750817)

[6.3 Сторінка «home.html» 44](#_Toc167750818)

[6.4 Сторінка «seat.html» 47](#_Toc167750819)

[6.5 Сторінка «about.html» 48](#_Toc167750820)

[6.6 Сторінка «data.html» 49](#_Toc167750821)

[6.7 Сторінка «data2.html» 51](#_Toc167750822)

[ВИСНОВОК 53](#_Toc167750823)

[ЛІТЕРАТУРА 54](#_Toc167750824)

[ДОДАТОК 56](#_Toc167750825)

# ВСТУП

У сучасному світі, де інформаційні технології надзвичайно стрімко розвиваються, використання сучасних інструментів для розробки програмного забезпечення стає вирішальним для успіху. Кожен новий курсовий проект стає можливістю не лише поглибити знання у конкретній мові програмування чи платформі, але й підготуватися до викликів сучасного індустріального та бізнес-середовища, де розробка програмного забезпечення відіграє ключову роль у конкурентній боротьбі.

Курсовий проект, розроблений на мові C#, є одним із таких проектів, який відкриває широкі можливості для вивчення і практичного застосування сучасних технологій програмування та розробки ПЗ. Відповідно до завдань, поставлених перед студентом, цей проект може включати в себе вирішення різноманітних завдань, починаючи від опрацювання масивів та роботи з динамічною пам'яттю і закінчуючи розробкою програм з використанням баз даних та мережевою взаємодією.

Мова програмування C# є однією з найбільш важливих та популярних мов в програмному середовищі, особливо в контексті розробки додатків для платформи .NET. Вона поєднує в собі потужність об'єктно-орієнтованого програмування з високим рівнем безпеки та ефективності. Крім того, її інтеграція з іншими продуктами Microsoft, такими як Visual Studio, дозволяє розробникам працювати в зручному та продуктивному середовищі.

Одним із ключових компонентів розробки програм на мові C# є інтегроване середовище розробки (IDE) Visual Studio. Це потужне інструментальне середовище, що надає розробникам широкий спектр можливостей для створення високоякісного програмного забезпечення. Інтелектуальна підтримка, потужні засоби налагодження та широкі можливості для розробки веб-додатків роблять Visual Studio невід'ємною частиною робочого процесу для багатьох розробників у всьому світі.

Додатково, важливо згадати і про мову розмітки HTML. HTML (HyperText Markup Language) є стандартною мовою для створення веб-сторінок. Вона використовується для структурування та відображення вмісту в Інтернеті. HTML дозволяє створювати різноманітні елементи на веб-сторінках, включаючи текст, зображення, посилання та форми. Розуміння HTML допомагає розробникам ефективно створювати інтерфейси користувача та забезпечувати їхню відповідність стандартам веб-розробки.

Тому курсовий проект, розроблений на мові C# в середовищі Visual Studio, не лише дозволить отримати практичні навички в програмуванні, але й допоможе підготуватися до викликів індустріального середовища та засвоїти сучасні підходи до розробки програмного забезпечення.

# 1. АНАЛІЗ СПЕЦИФІКИ ПРЕДМЕТНОЇ ОБЛАСТІ

## Опис предметної області

Залізнична каса є важливою ланкою у системі продажу квитків на залізничний транспорт. Основними функціями залізничної каси є надання пасажирам можливості купівлі квитків на майбутні рейси, а також забезпечення їх актуальною інформацією про наявність місць, розклад руху поїздів, тарифи та інші послуги.

Ключові дані, які необхідні для забезпечення роботи залізничної каси, включають:

* Номер і назва поїзда
* Дата і час відправлення та прибуття
* Номер вагона та місця
* Клас вагона (загальний, купейний, СВ)
* Ціна квитка, що залежить від поїзда, класу вагона, часу купівлі (можливі знижки)
* Дані про пасажира (вік, паспортні дані)
* Номер каси, де було здійснено покупку

Основними вимогами до системи продажу залізничних квитків є зручність використання для пасажирів та касирів, швидкий доступ до інформації, безпека даних та можливість обробки великої кількості запитів одночасно.

* 1. Аналіз аналогів на ринку

Для того, щоб розробити ефективну систему замовлення залізничних квитків, важливо проаналізувати існуючі рішення на ринку. Основними конкурентами та аналогами нашого проекту є:

Booking.com (розділ залізничних квитків):

* Переваги: Зручний інтерфейс, широкий вибір маршрутів, можливість перегляду відгуків та рейтингу, інтеграція з іншими видами транспорту.
* Недоліки: Обмежена інформація про наявність місць в реальному часі, додаткові збори за бронювання.

Rail Europe:

* Переваги: Глобальне покриття, підтримка багатьох мов та валют, можливість бронювання міжнародних маршрутів, інтеграція з готелями.
* Недоліки: Висока комісія за обслуговування, обмежена підтримка локальних маршрутів.

Trainline:

* Переваги: Інтуїтивно зрозумілий інтерфейс, мобільний додаток, реальний час оновлення інформації про наявність квитків, можливість вибору місць.
* Недоліки: Відсутність деяких міжнародних маршрутів, залежність від інтернет-з'єднання.

Укрзалізниця (офіційний сайт):

* Переваги: Актуальна інформація про наявність квитків, прямий доступ до всіх маршрутів Укрзалізниці, підтримка української мови, можливість вибору місць.
* Недоліки: Складний для користувачів інтерфейс, обмежена функціональність мобільного додатку, низька швидкість завантаження сторінок.

Висновки з аналізу аналогів

Аналізуючи аналоги, можна зробити висновок, що на ринку існує багато рішень, які мають свої переваги і недоліки. Наша система повинна поєднати найкращі риси цих рішень, а саме:

* Зручний та інтуїтивний інтерфейс.
* Можливість перегляду інформації в реальному часі.
* Інтеграція з різними платіжними системами.
* Швидка обробка запитів.
* Низькі комісії або їх відсутність.
* Широке покриття як локальних, так і міжнародних маршрутів.

Основний акцент необхідно зробити на зручності для користувача та наданні актуальної інформації про наявність квитків і можливість швидкого бронювання.

# 2. ПРОЕКТУВАННЯ СИСТЕМИ

## 2.1 Написання user stories та формування backlog'у продукту

User Stories:

* Як пасажир, я хочу зареєструватися в системі, щоб мати можливість купувати квитки онлайн.
* Як пасажир, я хочу переглядати розклад поїздів, щоб планувати свої подорожі.
* Як пасажир, я хочу купувати квитки онлайн, щоб уникнути черг у касі.
* Як пасажир, я хочу переглядати історію моїх поїздок і витрат, щоб контролювати свої подорожі та витрати.
* Як касир, я хочу реєструвати продаж квитків у системі, щоб автоматично генерувати чеки та зберігати інформацію про продажі.
* Як касир, я хочу мати доступ до інформації про наявність квитків, щоб надавати актуальні дані пасажирам.

Backlog продукту:

* Реєстрація користувача
* Реєстрація касира
* Перегляд розкладу поїздів
* Онлайн-купівля квитків
* Перегляд історії поїздок і витрат
* Генерація чеків
* Інтеграція з платіжними системами
* Інтеграція з базою даних поїздів
* Збереження даних про продажі
* Підтримка мультимовного інтерфейсу

## 2.2 Backlog продукту

Реєстрація користувача:

* Форма реєстрації з полями для введення облікових даних (ім'я, прізвище, email, пароль).
* Валідація введених даних.
* Збереження даних користувача в базі даних.
* Підтвердження успішної реєстрації.

Реєстрація касира:

* Форма реєстрації касира з полями для введення облікових даних (ім'я, прізвище, ідентифікаційний номер, пароль).
* Валідація введених даних.
* Збереження даних касира в базі даних.
* Підтвердження успішної реєстрації.

Перегляд розкладу поїздів:

* Форма пошуку розкладу поїздів за напрямком та датою.
* Відображення розкладу поїздів з можливістю фільтрації за різними параметрами.
* Інтеграція з базою даних поїздів для отримання актуальної інформації.

Онлайн-купівля квитків:

* Інтерфейс вибору поїзда, вагона та місця.
* Інтеграція з платіжними системами для оплати квитків.
* Генерація електронного квитка після успішної оплати.
* Відправка підтвердження замовлення на email користувача.

Перегляд історії поїздок і витрат:

* Особистий кабінет користувача з історією поїздок.
* Відображення інформації про попередні поїздки та витрати.
* Можливість експорту даних в зручному форматі (PDF, Excel).

Генерація чеків:

* Автоматичне створення чеку після продажу квитка.
* Відправка чеку на email пасажира.
* Збереження чеку в базі даних.

Інтеграція з платіжними системами:

* Підключення до популярних платіжних систем (Visa, MasterCard, PayPal).
* Забезпечення безпеки платежів.

Інтеграція з базою даних поїздів:

* Синхронізація з актуальною базою даних розкладів та наявності квитків.
* Оновлення даних в режимі реального часу.

Збереження даних про продажі:

* Збереження інформації про всі продажі квитків.
* Забезпечення доступу до даних для звітності.

Підтримка мультимовного інтерфейсу:

* Реалізація інтерфейсу на кількох мовах.
* Можливість перемикання мови користувачем.

## 2.3 Виявлення класів і зв’язків за допомогою CRC-карток

CRC-картки (Class-Responsibility-Collaborator):

Користувач (User)

* Відповідальність: Реєстрація, авторизація, перегляд розкладу, купівля квитків, перегляд історії поїздок.
* Співпраця: Система реєстрації, база даних, платіжна система, інтерфейс перегляду розкладу.

Касир (Cashier)

* Відповідальність: Реєстрація, продаж квитків, генерація чеків.
* Співпраця: Система реєстрації, база даних, інтерфейс продажу квитків.

Квиток (Ticket):

* Відповідальність: Зберігання інформації про поїзд, вагон, місце, клас, ціна.
* Співпраця: База даних, платіжна система.
* Поїзд (Train)
* Відповідальність: Зберігання інформації про номер, назву, розклад.
* Співпраця: База даних, інтерфейс перегляду розкладу.

Платіжна система (Payment System)

* Відповідальність: Обробка платежів, забезпечення безпеки.
* Співпраця: Користувач, касир, квиток.

## 2.4 UML – діаграми

UML-діаграми (Unified Modeling Language):

Діаграма класів:

* Класи: User, Cashier, Ticket, Train, PaymentSystem.
* Зв’язки: User <> Ticket, Cashier <> Ticket, Ticket <> Train, User <> PaymentSystem, Cashier <> PaymentSystem.

Діаграма прецедентів:

* Актори: Користувач, касир.
* Прецеденти: Реєстрація, авторизація, перегляд розкладу, купівля квитка, перегляд історії, продаж квитка, генерація чеку.

Діаграма активностей:

* Процеси: Реєстрація користувача, авторизація користувача, купівля квитка, перегляд розкладу, перегляд історії поїздок, продаж квитка, генерація чеку.

Діаграма послідовностей:

* Сценарії: Купівля квитка користувачем, продаж квитка касиром, генерація чеку, перегляд розкладу поїздів.

# 3. ЗАВДАННЯ ДО КУРСОВОЇ РОБОТИ

1 частина – додаток Windows Forms або Visual C#, що містить інтерфейси завдань, визначених претендентом на вищу освіту в лабораторній роботі № 6. Кількість форм не менше 3-х. Приклад програми – лабораторна робота № 4.

2 частина – HTML-сторінки та таблиця стилів, що містять інтерфейси завдань, визначених здобувачем вищої освіти у лабораторній роботі № 6. Кількість сторінок не менше 6-ти.

Розробити прикладне програмне забезпечення для продажу квитків на майбутні рейси залізничним оператором. Застосунок повинен надавати можливість користувачу самостійно обирати квитки на потяг. Квитки повинні продаватися тільки повнолітнім громадянам за умови пред'явлення паспорта. Реалізувати систему збереження даних про продані квитки, включаючи номер поїзда, дату, клас, інформацію про пасажирів та деталі оплати.

Застосунок повинен мати інтуїтивний і зручний інтерфейс для користувача, щоб він міг швидко і легко знайти потрібні квитки і здійснити оплату. Застосунок повинен мати механізм відображення актуальної інформації про наявність вільних місць на кожному рейсі.

Забезпечити систему авторизації та захисту даних користувачів та транзакцій для забезпечення конфіденційності та безпеки інформації.

Систему замовлення залізничних квитків присвячено автоматизації роботи залізничної каси. Для того щоб точно й оперативно працювати, необхідно автоматизувати роботу. Автоматизувати всю роботу залізничних кас ми не будемо, ми візьмемо лише частину їх, а точніше, відбудеться автоматизація роботи з клієнтами. При автоматизованій роботі людина просто вносить дані, а комп'ютер здійснює їх швидке опрацювання і видає результат.

# Частина 1.ОПИС ІНТЕРФЕЙСУ

Для даного проекту мною було створено 4 головних форми, які відповідають за повноцінну роботу програми.

3.1 Форма «Main.cs»

Ця форма, яка є головною при запуску програми, відіграє важливу роль в системі. Вона слугує входом до системи, де користувачі можуть ознайомитися з важливою інформацією. Основне завдання цієї сторінки полягає в додаванні нового клієнта до системи. Це включає введення всієї необхідної інформації про клієнта, такої як ім’я, контактні дані, адреса тощо.

Крім того, ця сторінка також відповідає за перегляд бази даних клієнтів. Користувачі можуть переглядати список всіх клієнтів, що зареєстровані в системі, а також детальну інформацію про кожного з них. Що стосується зміни інформації про клієнтів, ця сторінка дозволяє користувачам оновлювати дані про клієнтів. Це може бути корисним, наприклад, коли клієнт змінює свою адресу або контактні дані.

У цілому, ця сторінка є важливим інструментом для управління базою даних клієнтів, що дозволяє користувачам легко додавати, переглядати, змінювати та контролювати інформацію про клієнтів. Це, безумовно, сприяє ефективності та продуктивності системи.

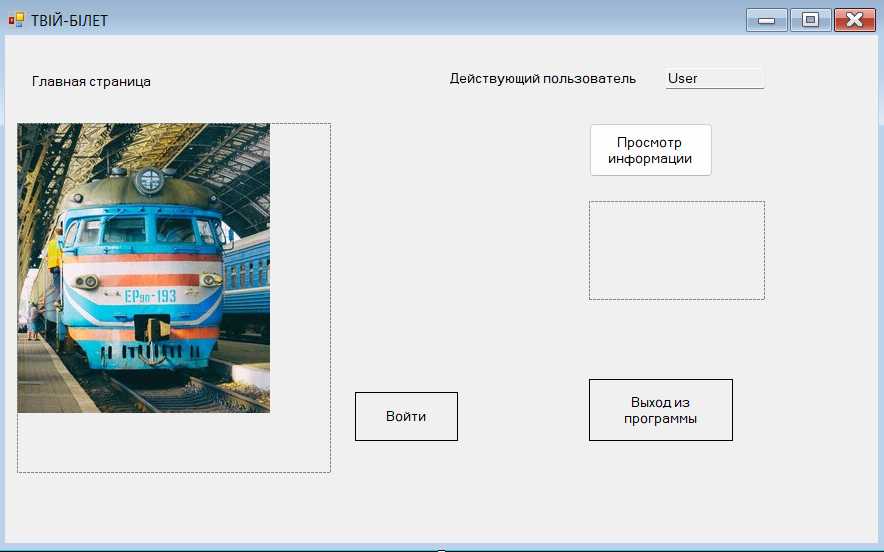


Рисунок 3.1.1 – вікно «Main»

## 3.2 Форма «Logon.cs»

Ця форма, яка була створена для входу в систему, є важливим елементом інтерфейсу користувача. Вона використовується для аутентифікації користувачів, перевіряючи їх ім’я користувача та пароль. У моему випадку, ім’я користувача та пароль для входу в систему є “Admin”.

Коли користувач вводить своє ім’я та пароль, система перевіряє ці облікові дані. Якщо вони вірні, користувач отримує доступ до системи. Якщо вони невірні, система відмовляє в доступі.

Як адміністратор, ми можемо керувати всіма аспектами системи, включаючи керування обліковими записами користувачів, налаштування системи, перегляд та аналіз даних тощо. Це включає можливість додавати, видаляти або змінювати облікові записи користувачів, налаштовувати параметри системи, переглядати звіти та аналізувати дані.

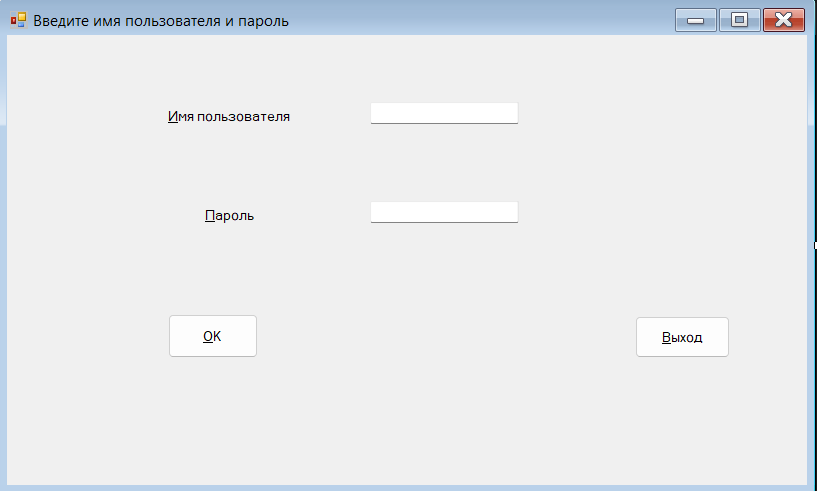


Рисунок 3.2.1 – вікно «Logon.cs»

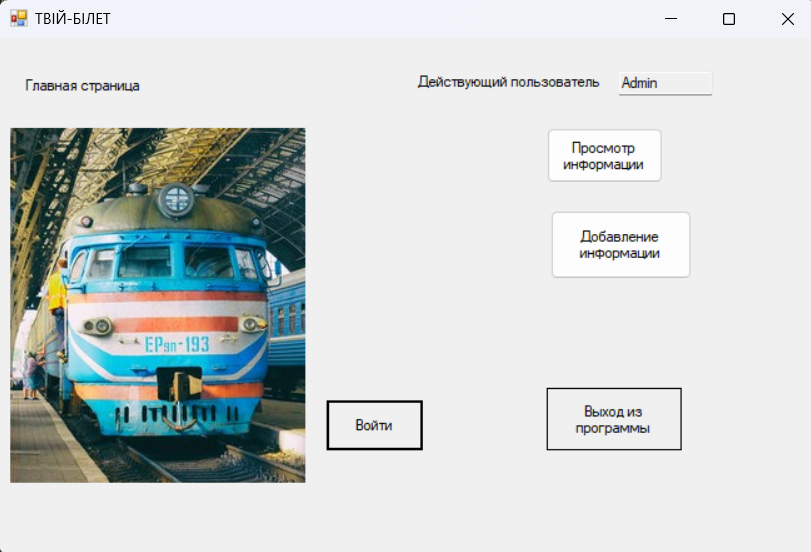


Рисунок 3.2.2 – вікно «Main» після входу користувача

3.3 Форма «AddInformation.cs»

Ця форма, яка була спеціально розроблена для додавання нових рейсів компанії до бази даних, є важливим інструментом управління рейсами. Вона дозволяє адміністратору системи (в цьому випадку, користувачу з ім’ям “Admin”) вводити деталі нових рейсів, які потім зберігаються в базі даних.

Оскільки всі ці деталі важливі для правильного відслідковування та управління рейсами, обов’язковим є заповнення всіх полів форми. Це гарантує, що в базі даних буде повна і точна інформація про кожен рейс.

Основне завдання цієї форми - створення та додавання нового рейсу в базу даних. Коли адміністратор вводить деталі нового рейсу та натискає кнопку “Додати”, інформація про рейс зберігається в базі даних. Це дозволяє компанії легко відслідковувати всі свої рейси та надавати цю інформацію своїм клієнтам.

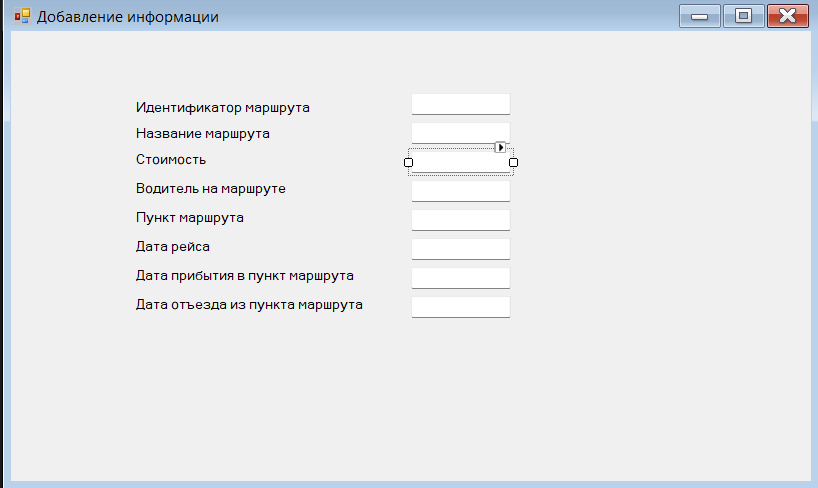


Рисунок 3.3.1 – вікно «AddInformation»

3.4 Форма «Information.cs»

Ця форма, розроблена для відображення інформації про рейси, є ключовим елементом системи. Вона дозволяє користувачам, як адміністраторам, так і звичайним користувачам, переглядати деталі про рейси, які зберігаються в базі даних. Основне завдання цієї форми - надати користувачам доступ до інформації про рейси. Це означає, що користувачі можуть переглядати деталі про рейси, перевіряти їх статус, змінювати свої бронювання та отримувати оновлення про зміни в розкладі рейсів.

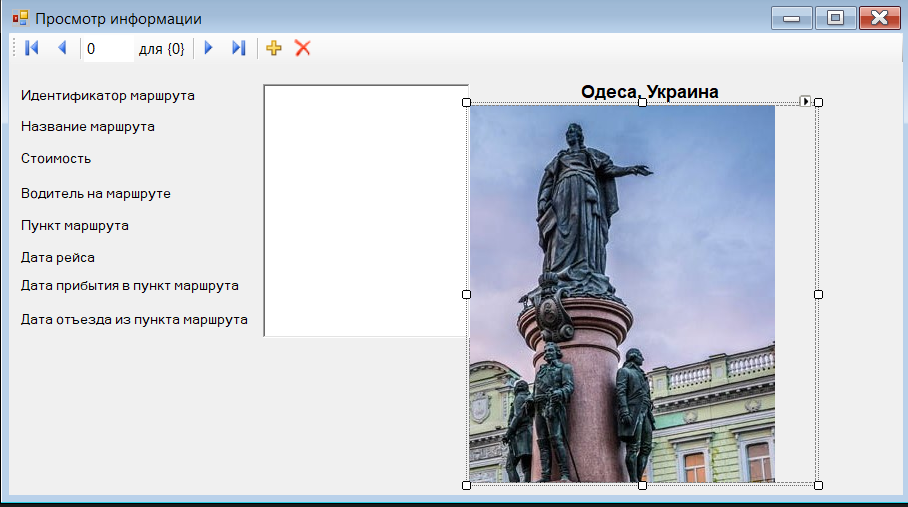


Рисунок 3.4.1 – вікно «Information»

# 4.БАЗА ДАНИХ

## 4.1 Опис теми роботи

Розроблено прикладне програмне забезпечення діяльності залізничної каси. Залізнична каса займається продажем квитків на майбутні рейси. У квитку вказується номер і назва поїзда, а також усі інші необхідні для пасажира дані: дата і час відправлення, прибуття, номер вагона, місця і клас (загальний, купейний, СВ). Ціна квитка залежить від поїзда, класу, а також від часу купівлі квитка - іноді робляться знижки тим, хто купив квиток більш ніж за місяць, або на «гарячі рейси» - все залежить від бажання компанії. Квитки продаються тільки повнолітнім громадянам за умови пред'явлення паспорта. У дороги є кілька кас, розташованих у різних кінцях міста, тому обов'язково необхідно враховувати номер каси, в якій було продано квиток, щоб уникнути непорозумінь під час здачі або обміну квитка.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Поле* | *Тип* | *Размер* | *Описание* |
| 1 | Passage | Числовой | 5 | Номер поезда |
| 2 | Title | Текстовый | 40 | Название поезда |
| 3 | Date | Дата/время | Авто | Дата отправления |
| 4 | Time | Дата/время | Авто | Время отправления |
| 5 | Arrival | Дата/время | Авто | Дата прибытия |
| 6 | ArrivalTime | Дата/время | Авто | Время прибытия |
| 7 | Seats | Логический | 1 | Промежуточные станции (есть/нет) |
| 8 | Places | Текстовый | 50 | Названия промежуточных станций |
| 9 | Passenger | Числовой | 5 | Регистрационный номер пассажира |
| 10 | Name | Текстовый | 40 | ФИО пассажира |
| 11 | Date | Дата/время | Авто | Дата покупки билета |
| 12 | Passport | Текстовый | 20 | Номер паспорта |
| 13 | PassportDate | Дата/время | Авто | Дата выдачи паспорта |
| 14 | Region | Текстовый | 40 | Кем выдан паспорт |
| 15 | Till | Числовой | 5 | Номер билетной кассы |
| 16 | Ticket | Числовой | 10 | Номер билета |
| 17 | Chair | Числовой | 3 | Номер места |
|  | TrainCar | Числовой | 2 | Номер вагона |
| 18 | Class | Числовой | 1 | Класс |
| 19 | Price | Денежный | 6 | Цена билета |
| 20 | Registrar | Текстовый | 15 | Фамилия кассира |
| 21 | Train | Числовой | 10 | Номер состава |
| 22 | TrainName | Текстовый | 15 | Название состава |
| 23 | Year | Числовой | 4 | Год основания маршрута |
| 24 | Picture | Вложение | Авто | Фотография состава |
| 25 | Amount | Числовой | 4 | Количество совершенных рейсов |
| 26 | Admin | Текстовый | 15 | Фамилия начальника поезда |
| 27 | Crew | Числовой | 4 | Номер бригады |
| 28 | Pilot | Текстовый | 15 | Фамилия машиниста |

Таблиця .1- таблиця Бази Даних для крусової роботи

## 4.2 Визначення переліку бізнес-функцій системи. Проектування та створення і заповнення бази даних.

Діаграма розробленої бази даних на рисунку 4.1

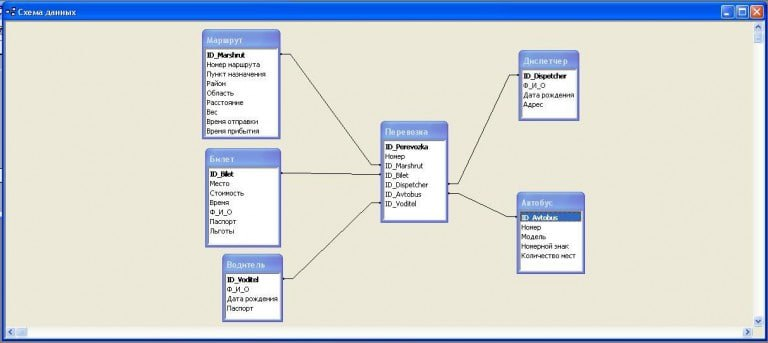


Рисунок 4.1 - Діаграма розробленої бази даних

* Запит на заповнення таблиці[TrainInfo], вказуючи різні параметри, такі як номер проходження, назва, дата, час, місця призначення, час прибуття, кількість місць, місця, номер поїзда, назва поїзда, рік, шлях до зображення, вартість та адміністратор.

Use Testbase

-- Inserting sample data into TrainInfo table

INSERT INTO TrainInfo (Passage, Title, Date, Time, Arrival, ArrivalTime, Seats, Places, Train, TrainName, Year, Picture, Amount, Admin)

VALUES

(1, 'Express 1', '2023-12-15', '08:00:00', '2023-12-15', '12:00:00', 1, 'StationA, StationB, StationC', 101, 'Express-A', 2020, 'path/to/picture1.jpg', 100, 'Smith'),

(2, 'Fast Train', '2023-12-16', '10:30:00', '2023-12-16', '14:30:00', 0, NULL, 102, 'Fast-Train', 2019, 'path/to/picture2.jpg', 80, 'Johnson'),

(3, 'City Connect', '2023-12-17', '15:45:00', '2023-12-17', '19:30:00', 1, 'StationX, StationY', 103, 'City-Connect', 2022, 'path/to/picture3.jpg', 120, 'Williams'),

(4, 'Local Express', '2023-12-18', '12:15:00', '2023-12-18', '16:30:00', 1, 'StationP, StationQ, StationR', 104, 'Local-Express', 2021, 'path/to/picture4.jpg', 90, 'Brown'),

(5, 'Night Rider', '2023-12-19', '21:00:00', '2023-12-20', '05:00:00', 1, 'StationM, StationN', 105, 'Night-Rider', 2018, 'path/to/picture5.jpg', 150, 'Taylor');

* Запит на заповнення таблиці[PassengerInfo], вказуючи різні параметри, такі як номер пасажира, ім'я, дата народження, номер паспорта, дата видачі паспорта та регіон проживання пасажира.

Use Testbase

-- Inserting sample data into PassengerInfo table

INSERT INTO PassengerInfo (Passenger, Name, Date, Passport, PassportDate, Region)

VALUES

(1, 'John Smith', '1990-05-15', 'ABC123456', '2010-05-20', 'CityA'),

(2, 'Jane Doe', '1985-08-22', 'XYZ789012', '2009-10-15', 'CityB'),

(3, 'Robert Johnson', '1978-12-10', 'PQR567890', '2005-03-18', 'CityC'),

(4, 'Emily White', '1993-07-08', 'LMN456789', '2011-02-28', 'CityD'),

(5, 'Michael Brown', '1982-04-25', 'DEF987654', '2008-09-10', 'CityE'),

(6, 'Sophia Miller', '1995-11-30', 'GHI321098', '2012-07-15', 'CityF');

* Запит на заповнення таблиці[TicketInfo], вказуючи різні параметри, такі як номер залу, номер квитка, номер крісла, номер вагону, клас, ціна, реєстратор та пасажир, пов'язаний з квитком.

Use Testbase

-- Inserting sample data into TicketInfo table

INSERT INTO TicketInfo (Till, Ticket, Chair, TrainCar, Class, Price, Registrar, Passenger)

VALUES

(1, 10101, 15, 1, 1, 50.00, 'Johnson', 1),

(2, 10102, 8, 2, 2, 75.50, 'Smith', 2),

(3, 10103, 20, 1, 1, 60.25, 'Brown', 3),

(4, 10104, 10, 2, 1, 55.75, 'Taylor', 4),

(5, 10105, 5, 3, 2, 80.00, 'Williams', 5),

(6, 10106, 12, 1, 1, 65.50, 'Smith', 6);

* Запит на заповнення таблиці[CrewInfo], вказуючи різні параметри, такі як номер екіпажу, пілот, склад екіпажу та пов'язаний поїзд.

Use Testbase

-- Inserting sample data into CrewInfo table

INSERT INTO CrewInfo (Crew, Pilot, CrewNames, Train)

VALUES

(1, 'John Anderson', 'Maria Davis, Richard Harris', 1),

(2, 'Emma White', 'Daniel Robinson, Sarah Johnson', 2),

(3, 'Christopher Martin', 'Olivia Taylor, James Miller', 3),

(4, 'Jessica Smith', 'Michael Brown, Emily White', 4),

(5, 'David Johnson', 'Sophia Miller, Matthew Davis', 5);

## 4.3 Організація пошуку у таблицях бази даних з допомогою SQL запитів

Цей запит вибирає всі рядки з таблиці TrainInfo, де значення у стовпці "Arrival" (дата прибуття) є пізніше (більше) 1 січня 2023 року. Іншими словами, він повертає інформацію про поїзди, які прибули після 1 січня 2023 року.

SELECT \* FROM TrainInfo WHERE Arrival > '2023-01-01';

Цей запит вибирає всі рядки з таблиці PassengerInfo, де значення у стовпці "Region" дорівнює 'CityA'. Це означає, що запит повертає інформацію про пасажирів, які мають регіон проживання, вказаний як 'CityA'.

SELECT \* FROM PassengerInfo WHERE Region = 'CityA';

Цей запит вибирає всі рядки з таблиці TicketInfo, де значення у стовпці "Price" (ціна) є більше 50.00. Таким чином, запит повертає інформацію про квитки, ціна яких перевищує 50.00 одиниць валюти.

SELECT \* FROM TicketInfo WHERE Price > 50.00;

Цей запит вибирає всі рядки з таблиці CrewInfo, де значення у стовпці "Pilot" (пілот) дорівнює 'John Anderson'. Таким чином, запит повертає інформацію про екіпажі, де пілот має ім'я 'John Anderson'.

SELECT \* FROM CrewInfo WHERE Pilot = 'John Anderson';

## 4.4 Розрахунок підсумків у таблицях бази даних з допомогою агрегатних функцій

Цей запит використовує функцію SUM для обчислення загальної суми значень у стовпці "Price" (ціна) в таблиці TicketInfo. Результат цього обчислення повертається як новий стовпець з іменем "TotalMoney" (загальна сума грошей).

SELECT SUM(Price) AS TotalMoney FROM TicketInfo;

Цей запит використовує функцію COUNT для підрахунку унікальних значень у стовпці "Passenger" (пасажир) в таблиці TicketInfo. Використано DISTINCT, щоб врахувати лише унікальні значення пасажирів. Результат цього підрахунку повертається як новий стовпець з іменем "TotalPassengers" (загальна кількість пасажирів).

SELECT COUNT(DISTINCT Passenger) AS TotalPassengers FROM TicketInfo;

Цей запит використовує функцію MAX для знаходження найбільшого значення у стовпці "PassportDate" (дата видачі паспорта) для кожного унікального поєднання значень у стовпцях "Passenger" та "Name" в таблиці PassengerInfo. Результат включає стовпці "Passenger" та "Name", а також новий стовпець з іменем "LastPassportDate", який містить останню дату видачі паспорта для кожного пасажира.

SELECT Passenger, Name, MAX(PassportDate) AS LastPassportDate

FROM PassengerInfo

GROUP BY Passenger, Name;

## 4.5 Пошук даних у складених сутностей з допомогою запитів на з’єднання та об’єднання таблиць

Цей запит використовує операцію JOIN для об'єднання інформації з таблиць TicketInfo (позначена як T) та PassengerInfo (позначена як P) за допомогою спільного стовпця "Passenger". Результат містить вибрані стовпці з обох таблиць, такі як номер залу (T.Till), номер квитка (T.Ticket), номер крісла (T.Chair), номер вагону (T.TrainCar), клас (T.Class), ціна (T.Price), реєстратор (T.Registrar), номер пасажира (T.Passenger), ім'я пасажира (P.Name AS PassengerName) та регіон пасажира (P.Region). Оператор JOIN дозволяє об'єднати відповідні рядки з обох таблиць за значенням у стовпці "Passenger".

SELECT

T.Till,

T.Ticket,

T.Chair,

T.TrainCar,

T.Class,

T.Price,

T.Registrar,

P.Passenger,

P.Name AS PassengerName,

P.Region

FROM TicketInfo AS T

JOIN PassengerInfo AS P ON T.Passenger = P.Passenger;

## 4.6 Модифікація даних з допомогою SQL запитів

Цей запит використовує мову SQL для оновлення запису в таблиці PassengerInfo. У даному випадку, ім'я (стовпець "Name") для пасажира з ідентифікатором 123 буде змінено на 'Valentin'.

UPDATE PassengerInfo

SET Name = 'Valentin'

WHERE Passenger = 123;

Цей запит використовує мову SQL для видалення рядків з таблиці TicketInfo, де значення у стовпці "Passenger" дорівнює 456. Таким чином, всі квитки, пов'язані з пасажиром із номером 456, будуть видалені з таблиці.

DELETE FROM TicketInfo

WHERE Passenger = 456;

## 4.7 Пошук даних у складених сутностей з допомогою підпорядкованих запитів.

Цей запит використовує подзапит, що повертає унікальні значення стовпця "Train" з таблиці TicketInfo, а потім вибирає всі рядки з таблиці TrainInfo, де значення стовпця "Train" присутнє у результаті цього підзапиту.

SELECT \*

FROM TrainInfo

WHERE Train IN (SELECT DISTINCT Train FROM TicketInfo);

Цей запит використовує підзапит для знаходження ідентифікатора пасажира (стовпець "Passenger") для особи з іменем 'Jane Doe' в таблиці PassengerInfo, а потім вибирає всі рядки з таблиці TicketInfo, де значення стовпця "Passenger" співпадає із знайденим значенням у підзапиті.

SELECT \*

FROM TicketInfo

WHERE Passenger = (

SELECT Passenger

FROM PassengerInfo

WHERE Name = 'Jane Doe'

);

## 4.8 Використання уявлень і табличних виразів для обробки даних з допомогою.

Цей SQL-запит створює вид (VIEW) з назвою "TicketInfoView". Вид представляє собою віртуальну таблицю, яка базується на результатах запиту SELECT, який об'єднує дані з таблиць TicketInfo (позначена як T) та PassengerInfo (позначена як P) за допомогою операції JOIN за спільним ключем "Passenger". Цей вид містить стовпці, які вибрані у SELECT-запиті, такі як "Till", "Ticket", "Chair", "TrainCar", "Class", "Price", "Registrar", "Passenger", "PassengerName", та "Region".

CREATE VIEW TicketInfoView AS

SELECT

T.Till,

T.Ticket,

T.Chair,

T.TrainCar,

T.Class,

T.Price,

T.Registrar,

P.Passenger,

P.Name AS PassengerName,

P.Region

FROM TicketInfo T

JOIN PassengerInfo P ON T.Passenger = P.Passenger;

Цей SQL-запит вибирає всі рядки з таблиці TicketInfo, де значення у стовпці "Price" (ціна) більше за середню ціну квитків у цій же таблиці. У підзапиті використовується функція AVG для обчислення середнього значення у стовпці "Price" таблиці TicketInfo, а потім це середнє значення порівнюється зі значеннями цін в основному запиті.

SELECT \*

FROM TicketInfo

WHERE Price > (SELECT AVG(Price) FROM TicketInfo);

## 4.9 Використання параметрів при обробці даних з допомогою збережених процедур.

Цей SQL-запит створює збережену процедуру з назвою "GetTicketsByPassenger", яка приймає параметр @PassengerId типу INT. У тілі процедури виконується запит SELECT, який вибирає всі рядки з таблиці TicketInfo, де значення у стовпці "Passenger" дорівнює переданому параметру @PassengerId.

CREATE PROCEDURE GetTicketsByPassenger

@PassengerId INT

AS

BEGIN

SELECT \*

FROM TicketInfo

WHERE Passenger = @PassengerId;

END;

Цей SQL-запит викликає збережену процедуру "GetTicketsByPassenger" із передачею значення параметра @PassengerId, яке дорівнює 1. Процедура виконує внутрішній SELECT-запит, вибираючи всі рядки з таблиці TicketInfo, де значення у стовпці "Passenger" дорівнює переданому значенню @PassengerId. Це дозволяє отримати інформацію про квитки для конкретного пасажира, чий ідентифікатор передано в якості параметра.

EXEC GetTicketsByPassenger @PassengerId = 1;

Цей SQL-запит створює збережену процедуру з назвою "UpdateTicketPrice". Процедура приймає два параметри: @TicketId типу INT та @NewPrice типу DECIMAL(6,2) OUTPUT. У тілі процедури виконується два SQL-запити. Перший UPDATE-запит оновлює ціну (стовпець "Price") квитка з ідентифікатором @TicketId на нове значення, яке передається як параметр @NewPrice. Другий SET-запит присвоює значенню параметра @NewPrice фактичне значення ціни, яке оновлено в таблиці.

CREATE PROCEDURE UpdateTicketPrice

@TicketId INT,

@NewPrice DECIMAL(6,2) OUTPUT

AS

BEGIN

UPDATE TicketInfo

SET Price = @NewPrice

WHERE Ticket = @TicketId;

SET @NewPrice = (SELECT Price FROM TicketInfo WHERE Ticket = @TicketId);

END;

## 4.10 Використання тригерів при обробці даних

Цей SQL-запит створює тригер з назвою "UpdateTicketPriceTrigger" для таблиці TicketInfo. Тригер викликається після оновлення будь-якого рядка у таблиці TicketInfo і використовує ключове слово AFTER UPDATE. В тілі тригера встановлено SET NOCOUNT ON; для того, щоб вимкнути повідомлення про кількість змінених рядків, що повертається як результат виконання UPDATE-запиту. Умова IF UPDATE(Price) перевіряє, чи був оновлений стовпець "Price". Якщо це так, виконується блок логіки всередині BEGIN...END, який може містити додаткові дії або логіку, що виконується при оновленні ціни квитка. У даному випадку, використовується команда PRINT для виведення повідомлення у консоль або журнал.

CREATE TRIGGER UpdateTicketPriceTrigger

ON TicketInfo

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF UPDATE(Price)

BEGIN

-- Логіка, яка виконається при оновленні ціни квитка

PRINT 'Ціна квитка була оновлена. Виконую додаткові дії...';

END

END;

Цей SQL-запит створює тригер з назвою "CreateCustomerInfoTrigger" для таблиці PassengerInfo. Тригер викликається після вставки нових рядків у таблицю PassengerInfo і використовує ключове слово AFTER INSERT. В тілі тригера встановлено SET NOCOUNT ON; для вимкнення повідомлення про кількість доданих рядків, яке повертається як результат виконання INSERT-запиту. Потім використовується команда INSERT INTO, яка вставляє нові записи в таблицю CustomerInfo. Дані для вставки беруться з таблиці INSERTED, яка містить нові рядки, додані до PassengerInfo під час виконання INSERT-запиту. В даному випадку, для CustomerId береться значення "Passenger" з INSERTED, для CustomerName - "Name", а RegistrationDate встановлюється як поточна дата за допомогою функції GETDATE().

CREATE TRIGGER CreateCustomerInfoTrigger

ON PassengerInfo

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO CustomerInfo (CustomerId, CustomerName, RegistrationDate)

SELECT

i.Passenger,

i.Name,

GETDATE()

FROM

INSERTED i;

END;

## 5.ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ

## 5.1 Програмна реалізація форми “Main.cs”.

Цей лістинг коду представляє клас Main, який є частиною програми, розробленої для курсової роботи. Клас Main є похідним від класу Form, що є частиною бібліотеки System.Windows.Forms в .NET Framework. Це означає, що Main є формою Windows, яка має різні елементи керування та обробники подій.

Ось деякі ключові елементи цього класу:

* userAdminButton: Це кнопка, яка створюється динамічно, коли користувач входить в систему як адміністратор.
* Main(): Це конструктор класу, який викликає метод InitializeComponent() для ініціалізації компонентів форми.
* exitButton\_Click(): Цей обробник подій викликається, коли користувач натискає кнопку виходу. Він закриває програму.
* userAdminButton\_Click(): Цей обробник подій викликається, коли користувач натискає кнопку userAdminButton. Він відкриває нову форму AddInformation.
* autorizeButton\_Click(): Цей обробник подій викликається, коли користувач натискає кнопку авторизації. Він відкриває форму Logon і, якщо користувач входить як адміністратор, додає кнопку userAdminButton до форми.
* tourButton\_Click(): Цей обробник подій викликається, коли користувач натискає кнопку tourButton. Він відкриває нову форму Information.

Форма “Main.cs”:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсова

{

public partial class Main : Form

{

Button userAdminButton;

public Main()

{

InitializeComponent();

}

private void exitButton\_Click(object sender, EventArgs e)

{

System.Environment.Exit(0);

}

private void Main\_Load(object sender, EventArgs e)

{

}

private void userAdminButton\_Click(object sender, EventArgs e)

{

try

{

Form newAddInformation = new AddInformation();

newAddInformation.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error encountered", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

private void exitButton\_MouseEnter(object sender, EventArgs e)

{

exitButton.BackColor = Color.Green;

}

private void exitButton\_MouseLeave(object sender, EventArgs e)

{

exitButton.BackColor = Color.White;

}

private void autorizeButton\_Click(object sender, EventArgs e)

{

Logon logon = new Logon();

logon.ShowDialog(this);

try

{

if (logon.username != "")

{

currentUserTextBox.Text = logon.username;

if (logon.isAdmin)

{

userAdminButton = new Button();

userAdminButton.Name = "UserAdminButton";

userAdminButton.Text = "Добавление информации";

userAdminButton.Width = 113;

userAdminButton.Height = 55;

navigationFlowLayout.Controls.Add(userAdminButton);

userAdminButton.Click += userAdminButton\_Click;

}

}

else

{

System.Environment.Exit(0);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error encountered", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void tourButton\_Click(object sender, EventArgs e)

{

try

{

Form newInformation = new Information();

newInformation.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error encountered", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

}

}

## 5.2 Програмна реалізація форми “Logons.cs”.

Цей лістинг коду представляє клас Logon, який є формою входу в систему. Він є частиною програми, розробленої для курсової роботи. Клас Logon є похідним від класу Form, що є частиною бібліотеки System.Windows.Forms в .NET Framework.

Ось деякі ключові елементи цього класу:

* tries, isAdmin, username, password: Це змінні, які використовуються для зберігання даних про спроби входу, статус користувача та облікові дані користувача.
* Logon(): Це конструктор класу, який викликає метод InitializeComponent() для ініціалізації компонентів форми та ініціалізує змінні.
* OK\_Click(): Цей обробник подій викликається, коли користувач натискає кнопку OK. Він перевіряє облікові дані користувача та, якщо вони вірні, закриває форму. Якщо облікові дані невірні, він збільшує кількість спроб входу та показує повідомлення про помилку.
* Cancel\_Click(): Цей обробник подій викликається, коли користувач натискає кнопку Cancel. Він закриває форму.

Форма “Logons.cs”:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсова

{

public partial class Logon : Form

{

int tries;

public bool isAdmin;

public string username;

public string password;

public Logon()

{

InitializeComponent();

tries = 0;

username = "";

password = "";

}

private void OK\_Click(object sender, EventArgs e)

{

username = usernameTextBox.Text;

password = passwordTextBox.Text;

if (username == "Admin" && password == "Admin")

{

isAdmin = true;

this.Close();

}

else

{

usernameTextBox.Text = "";

passwordTextBox.Text = "";

usernameTextBox.Focus();

tries++;

if (tries == 3)

{

OK.Enabled = false;

isAdmin = false;

username = "";

password = "";

MessageBox.Show("You have failed to login three times in a row. The login form will be closed.", "Too many incorrect attempts");

this.Close();

}

else

{

MessageBox.Show("Incorrect username or password", "Login error");

}

}

}

private void Cancel\_Click(object sender, EventArgs e)

{

this.Close();

}

private void Logon\_Load(object sender, EventArgs e)

{

}

private void usernameTextBox\_TextChanged(object sender, EventArgs e)

{

}

}

}

## 5.3 Програмна реалізація форми “ AddInformation.cs”.

Цей лістинг коду представляє клас AddInformation, який є формою для додавання інформації в систему. Він є частиною програми, розробленої для курсової роботи. Клас AddInformation є похідним від класу Form, що є частиною бібліотеки System.Windows.Forms в .NET Framework.

Ось деякі ключові елементи цього класу:

* AddInformation(): Це конструктор класу, який викликає метод InitializeComponent() для ініціалізації компонентів форми.
* AddInformation\_Load(): Цей обробник подій викликається, коли форма завантажується. Він може бути використаний для ініціалізації даних або налаштування елементів форми.
* label1\_Click(): Цей обробник подій викликається, коли користувач натискає на мітку label1. Він може бути використаний для виконання певних дій, коли користувач натискає на мітку.

Форма “AddInformation.cs”:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсова

{

public partial class AddInformation : Form

{

public AddInformation()

{

InitializeComponent();

}

private void AddInformation\_Load(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

}

}

## 5.4 Програмна реалізація форми “ Information.cs”.

Цей лістинг коду представляє клас Information, який є формою для відображення інформації в системі. Він є частиною програми, розробленої для курсової роботи. Клас Information є похідним від класу Form, що є частиною бібліотеки System.Windows.Forms в .NET Framework.

Ось деякі ключові елементи цього класу:

* Information(): Це конструктор класу, який викликає метод InitializeComponent() для ініціалізації компонентів форми.
* Information\_Load(): Цей обробник подій викликається, коли форма завантажується. Він може бути використаний для ініціалізації даних або налаштування елементів форми.
* label3\_Click(), label9\_Click(): Ці обробники подій викликаються, коли користувач натискає на мітки label3 та label9. Вони можуть бути використані для виконання певних дій, коли користувач натискає на мітки.
* bindingNavigator1\_RefreshItems(): Цей обробник подій викликається, коли користувач оновлює елементи навігаційного біндера. Він може бути використаний для оновлення даних або відображення елементів.

Форма “Information.cs”:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсова

{

public partial class Information : Form

{

public Information()

{

InitializeComponent();

}

private void Information\_Load(object sender, EventArgs e)

{

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void label9\_Click(object sender, EventArgs e)

{

}

private void bindingNavigator1\_RefreshItems(object sender, EventArgs e)

{

}

}

}

# Частина 2. СТВОРЕННЯ HTML-СТОРІНОК ДО ДОДАТКУ

У даному розділі створено веб-сторінки для підприємства – ТВІЙ-БІЛЕТ. Сайт орієнтований на людей в руслі подорожі та має бути легким для довгого знаходження на ньому. Саме тому головним кольором для цього проєкту обрано білий, що забезпечує чистий та затишний вигляд.

У процесі розробки веб-сторінок для ТВІЙ-БІЛЕТ, враховано принципи дизайну, що спрямовані на комфортне сприйняття сайту клієнтом. Використано принципи Брусенцевої, які надають можливість користувачеві взаємодіяти з сайтом, не потребуючи додаткової допомоги офісного працівника. Важлива увага приділена когнітивному та несвідомому керуванню на сайті, щоб деякі функції, такі як заповнення форм або вибір послуг, були інтуїтивно зрозумілі.

При проектуванні веб-сайту ТВІЙ-БІЛЕТ використовувалася семантична технологія, що дозволяє створювати сторінки з чіткою структурою та розумінням контенту. Всі елементи створені з урахуванням потреб користувачів, а зайві відволікаючі фактори та зображення були уникнуті. Головною метою було забезпечити можливість користувачам розібратися з керуванням на сайті без додаткової допомоги офісного працівника.

В процесі реалізації даного веб-сайту використовувалися різноманітні технології, зокрема мова розмітки HTML. HTML є однією з основних мов веб-розробки, що використовується для створення структури та розмітки веб-сторінок. Використання семантичної розмітки HTML дозволяє зберігати сторінки чистими та легкими для розуміння як для людей, так і для пошукових систем. Таким чином, HTML відіграв важливу роль у створенні користувацького дружнього та зручного веб-сайту для ТВІЙ-БІЛЕТ.

# 6. РЕАЛІЗАЦІЯ ПРОЕКТУ

У проекті всього 9 сторінок, перехід по яких виконується за допомогою кнопок. Усі сторінки мають підключення до CSS стилів.

## 6.1 Сторінка «index.html»

Сторінка «index.html» відіграє ключову роль у програмі, оскільки вона є головною при запуску. Ця сторінка відповідає за процес авторизації користувача, де він може ввести свої облікові дані для входу в систему.

У процесі авторизації користувача на сторінці «index.html» можуть використовуватися різні методи аутентифікації, наприклад, введення логіна та паролю.

Частковий код програми, пов'язаний з процесом авторизації, можна побачити на рисунку 6.1.1. Цей код може включати в себе форму для введення облікових даних, перевірку їх правильності та перенаправлення користувача на відповідну сторінку після успішної авторизації.

Приклад вигляду сторінки «index.html» можна побачити на рисунку 6.1.2. Цей приклад демонструє зовнішній вигляд форми авторизації, який може включати поля для введення логіна та паролю, кнопку для надсилання облікових даних та інші елементи, необхідні для коректного виконання авторизації.



Рисунок 6.1.1 – код сторінки «index.html»

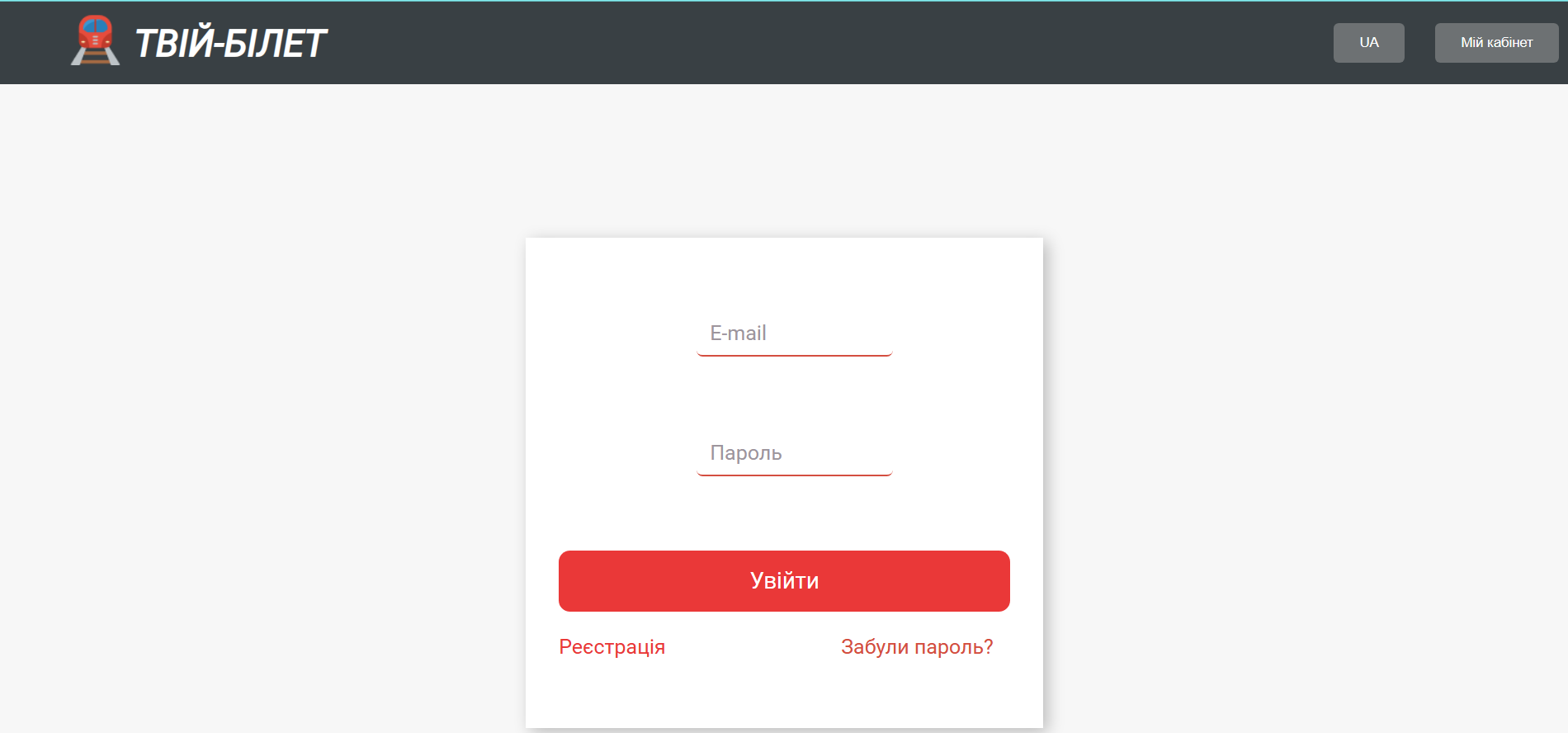


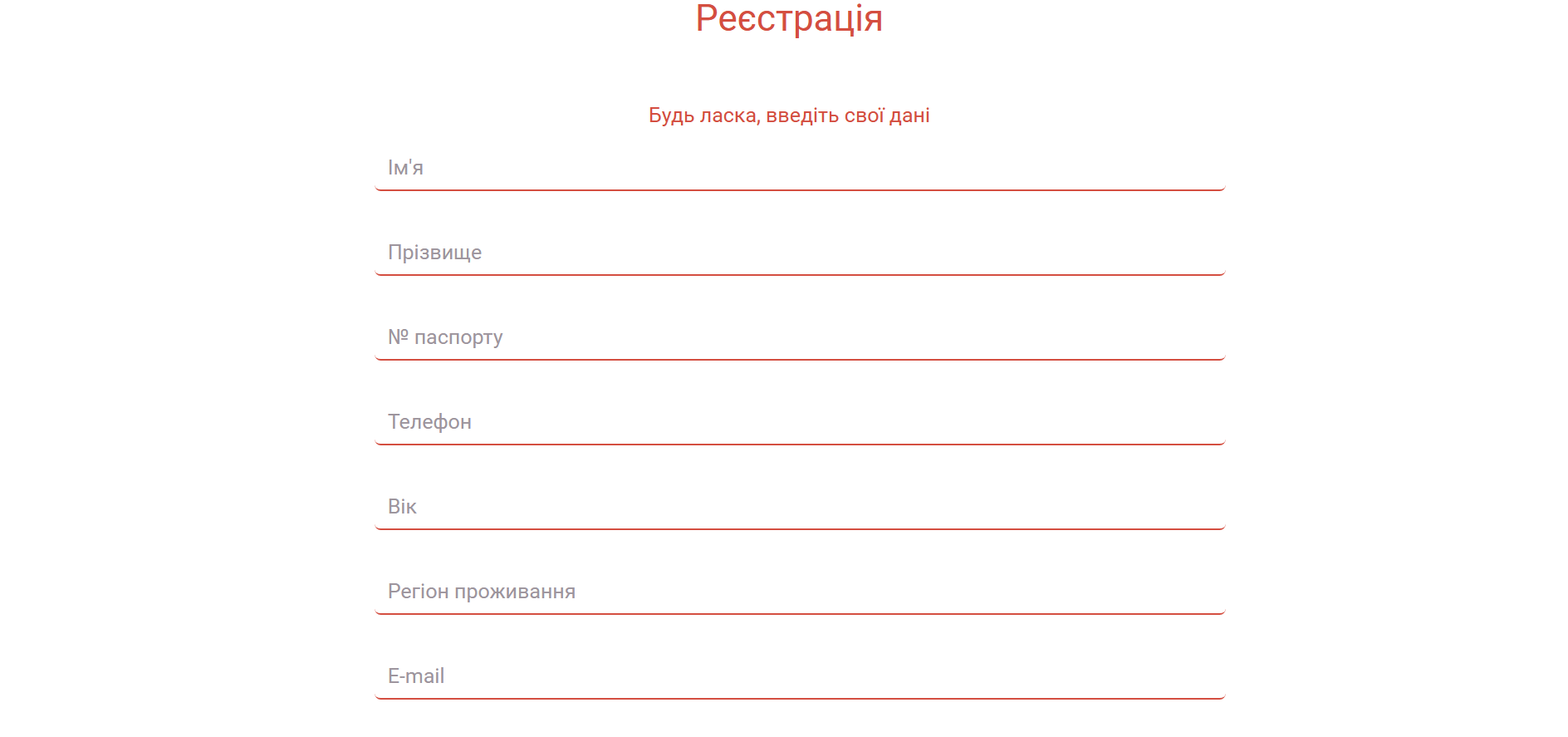
Рисунок 6.1.2 – вигляд сторінки авторизації

## 6.2 Сторінка «registration.html»

Сторінка «registration.html» надає можливість користувачу зареєструватись у додатку, якщо в нього ще нема облікового запису. Ця сторінка містить форму, яка складається з обов'язкових полів для заповнення, таких як ім'я, прізвище, адреса електронної пошти тощо. Якщо користувач не заповнить всі обов'язкові поля, він отримає відповідне зауваження поруч з цими полями, що допоможе йому уникнути помилок при реєстрації.

На сторінці «registration.html» також розміщені кнопки, які дозволяють користувачеві керувати відповідними діями на сторінці, такими як підтвердження реєстрації або повернення до головної сторінки. Ці кнопки забезпечують зручний та інтуїтивно зрозумілий інтерфейс для користувача під час реєстрації.

Для оформлення сторінки «registration.html» використовується підключення до файлу стилів «style.css». Це дозволяє забезпечити однорідний та привабливий зовнішній вигляд форми реєстрації, що покращує користувацький досвід та робить процес реєстрації більш приємним для користувачів.

Рисунок 6.2.1– вигляд сторінки реєстрації

## 6.3 Сторінка «home.html»

Сторінка «home.html» призначена для ознайомлення користувача з найпопулярнішими рейсами та надає можливість обрати час та дату для купівлі квитка на потяг. Ця сторінка має важливе значення для користувачів, оскільки вона надає швидкий доступ до інформації про доступні рейси та дати відправлення.

На сторінці «home.html» користувач може переглянути список найпопулярніших рейсів, що дозволяє йому зробити швидкий вибір та перевірити доступність місць на цих рейсах. Крім того, користувач має можливість обрати зручний час та дату для купівлі квитка, використовуючи відповідні функціональні елементи на сторінці.

Для оформлення сторінки «home.html» застосовується підключення до файлу стилів «style.css». Це дозволяє забезпечити зручний та привабливий зовнішній вигляд сторінки, що покращує користувацький досвід та робить процес пошуку та покупки квитків більш приємним та зручним для користувачів.

Приклад вигляду сторінки «home.html» можна побачити на рисунках 6.3.1 – 6.3.3. Ці приклади демонструють зовнішній вигляд та функціональність сторінки, яка допомагає користувачам швидко та зручно здійснювати пошук та покупку квитків на потяг.

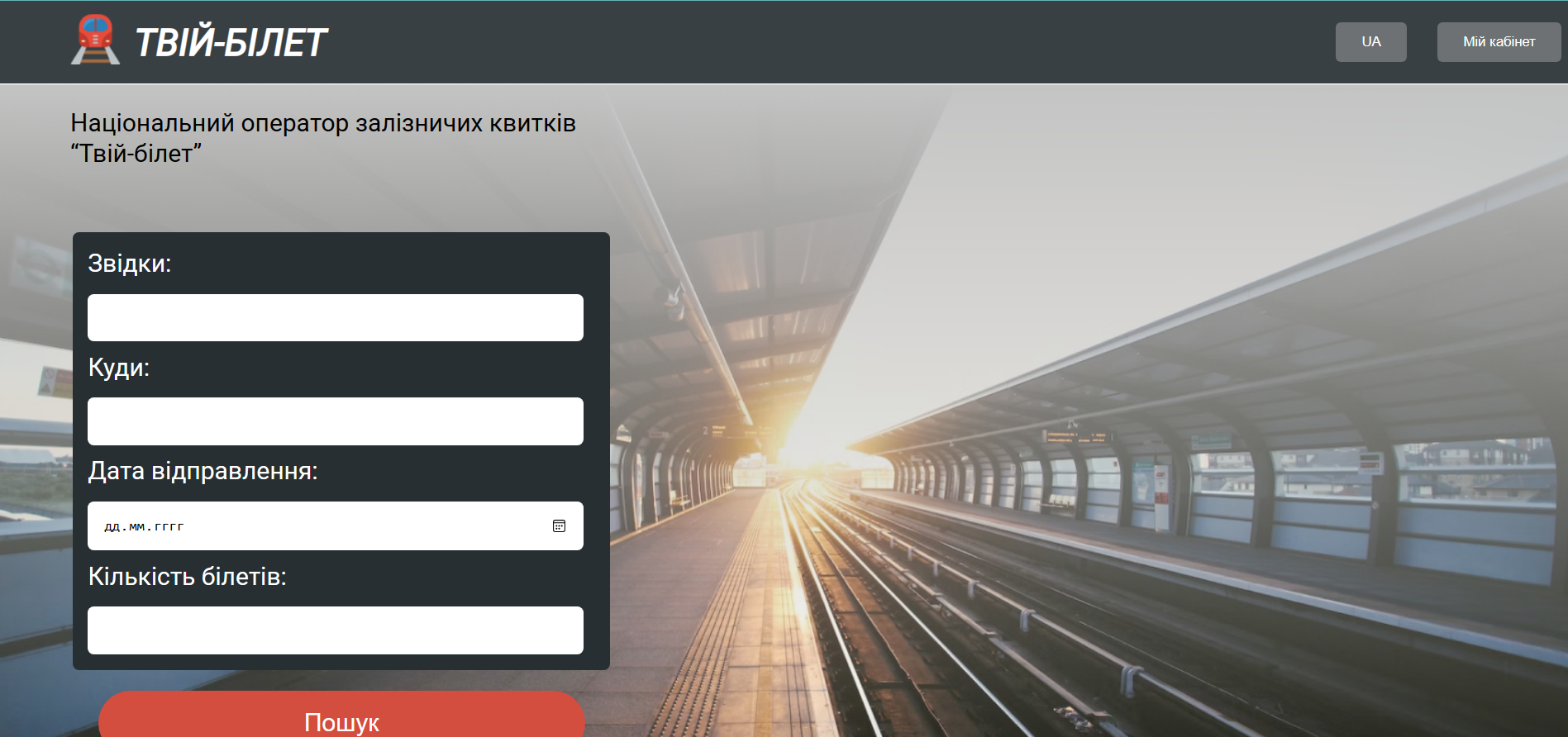


Рисунок 6.3.1 – вигляд головної сторінки сайту

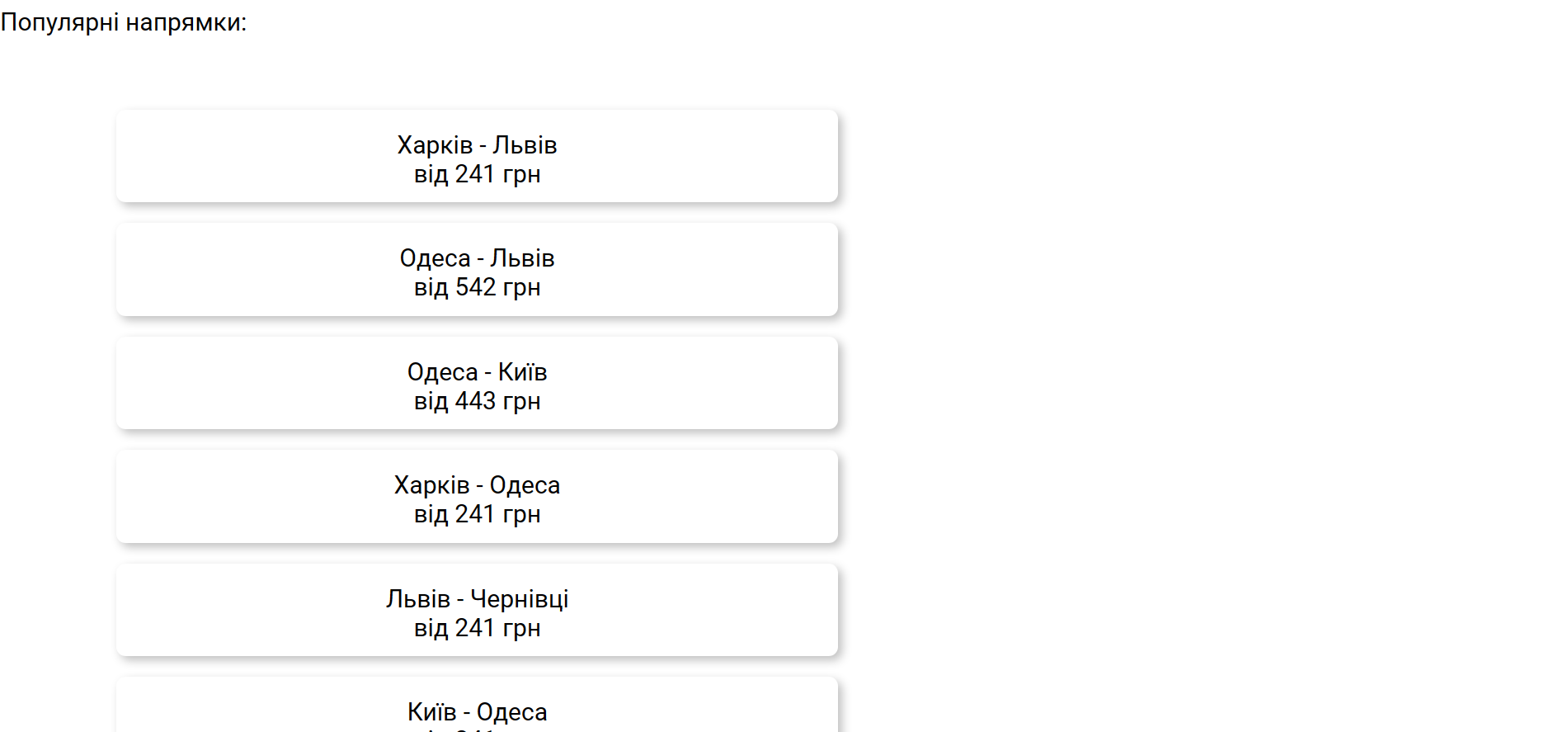


Рисунок 6.3.2. – вигляд головної сторінки сайту

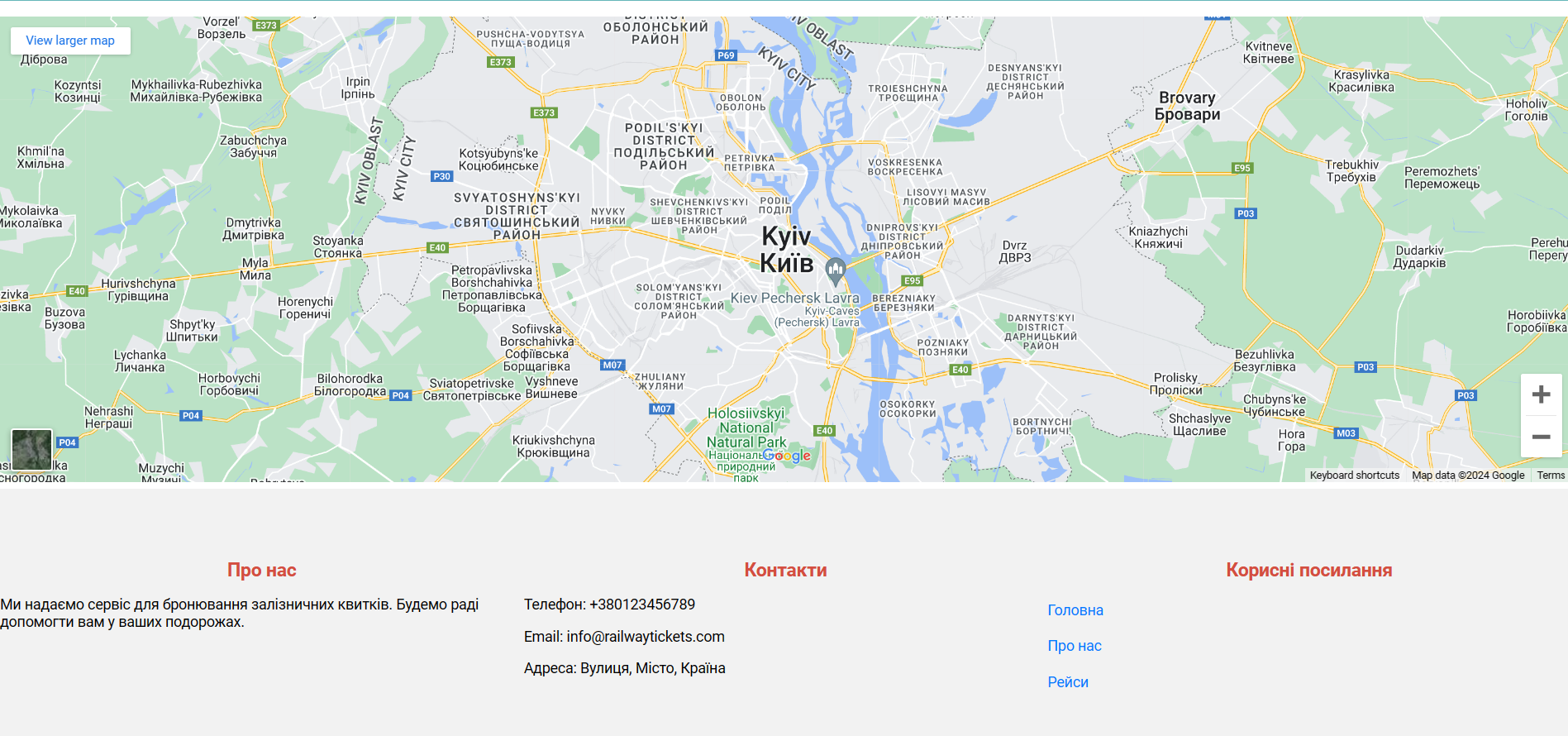


Рисунок 6.3.3 – вигляд головної сторінки сайту

## 6.4 Сторінка «seat.html»

Сторінка «seat.html» відповідає за вибір місць клієнта на сайті та надає можливість забронювати саме ці місця. Ця сторінка є важливою для користувачів, оскільки вона дозволяє їм вибрати бажані місця та забронювати їх для подорожі потягом. Для реалізації цієї функціональності використовується JavaScript, що забезпечує інтерактивність та зручність у виборі місць.

На сторінці «seat.html» клієнт може взаємодіяти з кнопками для вибору місць та їх бронювання. Користувач може вибрати потрібні місця, а після цього вони будуть заброньовані для нього на відповідний рейс. Крім того, на цій сторінці клієнт має можливість залишити свої контактні дані для купівлі квитка, що дозволяє забезпечити зручний та швидкий процес оформлення замовлення.

Приклад вигляду сторінки «seat.html» можна побачити на рисунку 6.4.1. Цей приклад демонструє зовнішній вигляд та функціональність сторінки, яка допомагає користувачам зручно та швидко вибирати та бронювати місця для подорожі потягом.

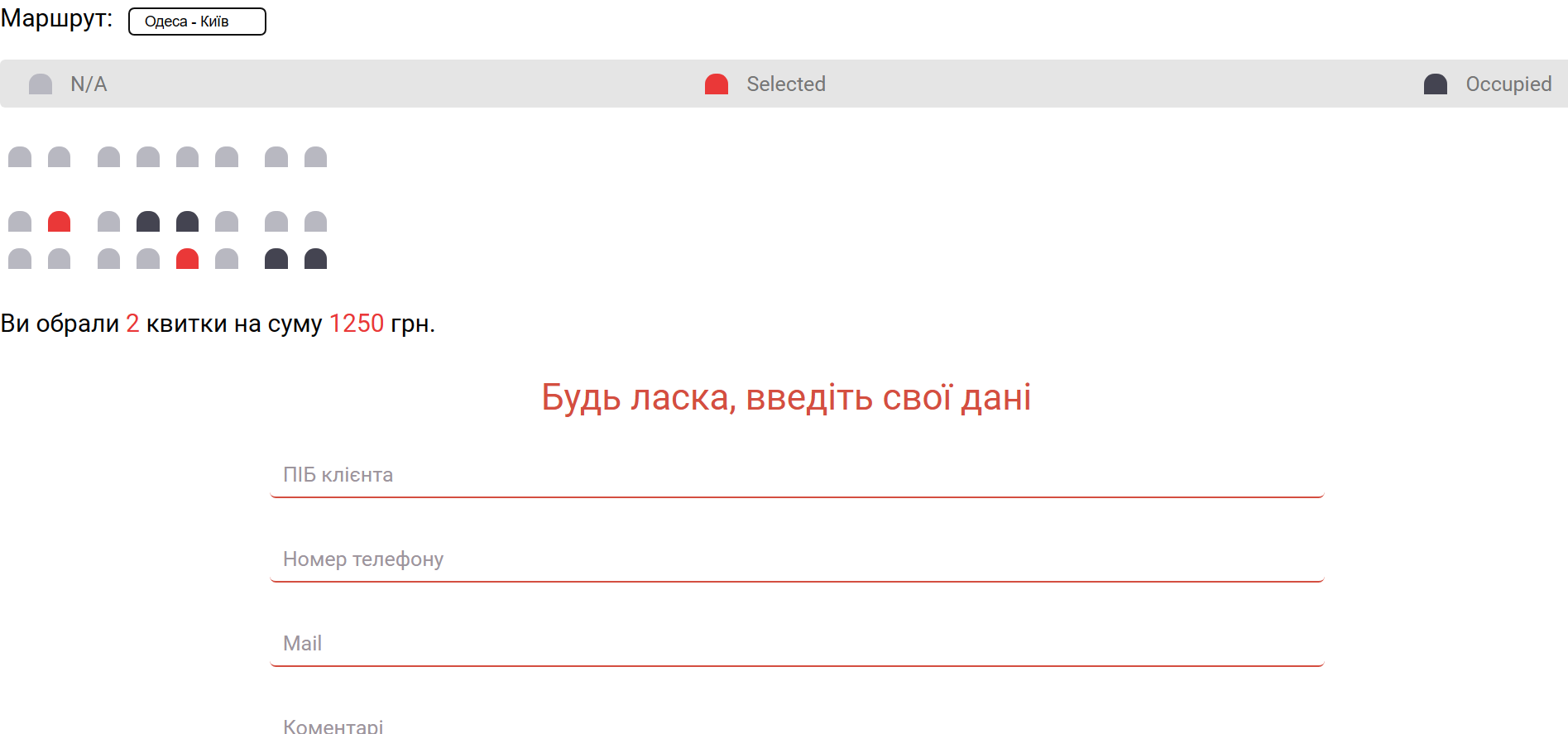


Рисунок 6.4.1 – вигляд сторінки вибору місць

## 6.5 Сторінка «about.html»

Сторінка «about.html» відповідає за унікальну назву компанії та знайомить користувача з основною інформацією про компанію, а також надає контактні дані для зв'язку. Ця сторінка є важливою для будь-якого веб-сайту, оскільки вона дозволяє відвідувачам отримати інформацію про компанію та зв'язатися з нею в разі потреби.

На сторінці «about.html» можна знайти короткий опис компанії, включаючи інформацію про її діяльність, місію, цінності та історію. Крім того, на цій сторінці можуть бути представлені ключові контактні дані компанії, такі як адреса, телефон, електронна пошта тощо, що дозволяє користувачам зв'язатися з компанією для отримання додаткової інформації або рішення питань.

Приклад вигляду сторінки «about.html» можна побачити на рисунку 6.5.1. Цей приклад демонструє зовнішній вигляд та оформлення сторінки, яке допомагає залучити увагу користувачів та надати їм необхідну інформацію про компанію.



Рисунок 6.5.1 – сторінки сайту «Про нас»

## 6.6 Сторінка «data.html»

Сторінка «data.html» відповідає за показ бази усіх рейсів і дозволяє користувачам керувати переглядом цих даних за допомогою кнопок. На цій сторінці можна переглядати інформацію про рейси, які доступні для бронювання, включаючи дату, час, маршрут та інші важливі деталі.

Керування по веб-сторінкам виконується за допомогою кнопок, розміщених на сторінці «data.html». Користувач може використовувати ці кнопки для навігації між різними рейсами, перегляду додаткової інформації та здійснення інших дій.

Приклад вигляду сторінки «data.html» та коду реалізації можна побачити на рисунках 6.6.1 та 6.6.2 відповідно. Ці приклади демонструють зовнішній вигляд сторінки та спосіб реалізації функціональності перегляду даних про рейси на веб-сайті.

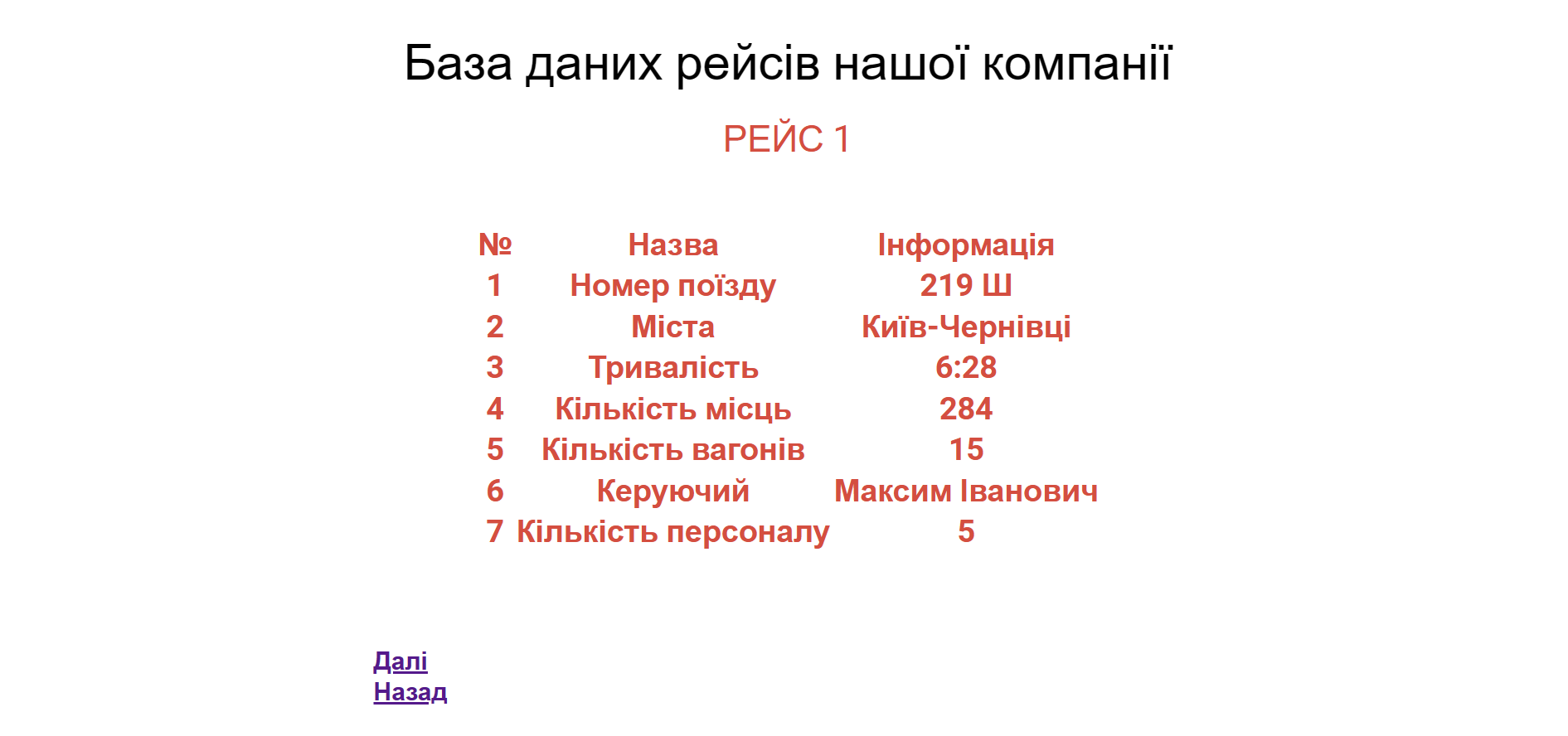


Рисунок 6.6.1 – вигляд сторінки «data.html»

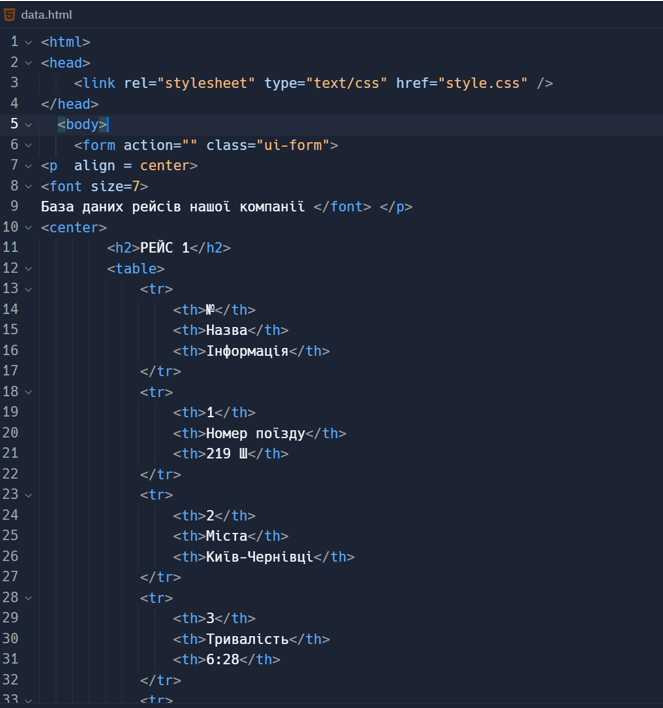


Рисунок 6.6.2 – код сторінки «data.html»

## 6.7 Сторінка «data2.html»

Сторінка «data2.html» також відповідає за рейсів та дозволяє користувачам керувати переглядом цих даних за допомогою кнопок. На цій сторінці можна переглядати інформацію про рейси, доступні для бронювання, включаючи дату, час, маршрут та інші важливі деталі.

Керування по веб-сторінкам виконується за допомогою кнопок, розміщених на сторінці «data2.html». Користувач може використовувати ці кнопки для навігації між різними рейсами, перегляду додаткової інформації та здійснення інших дій.

Для оформлення сторінки «data2.html» використовується підключення до файлу стилів «style.css». Це дозволяє створити зручний та привабливий зовнішній вигляд сторінки, що покращує користувацький досвід та робить процес перегляду даних більш приємним для користувачів.

Приклад вигляду сторінки «data2.html» можна побачити на рисунку 6.7.1. Цей приклад демонструє зовнішній вигляд та оформлення сторінки, яке допомагає залучити увагу користувачів та забезпечити їм зручний доступ до інформації про рейси на веб-сайті.

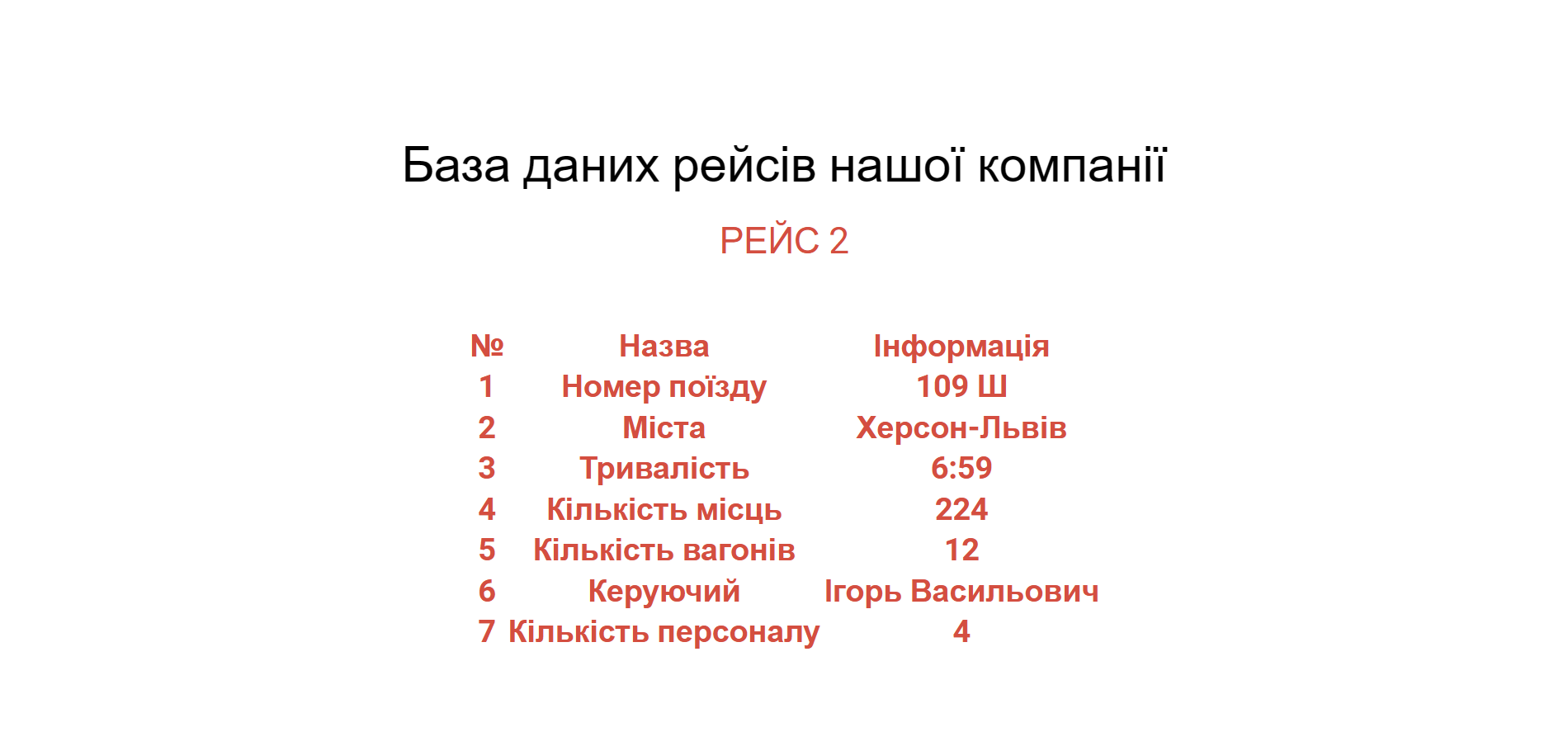


Рисунок 6.7.1 – вигляд сторінки «data2.html»

# ВИСНОВОК

Розвиток інформаційних технологій сприяє зростанню зручності та доступності для користувачів у багатьох сферах, зокрема в сфері транспортних послуг. Система замовлення залізничних квитків через мобільні додатки та онлайн-сервіси стає все більш популярною, оскільки дозволяє швидко та зручно забронювати та оплатити квитки, минувши черги та витрачення часу на поїздку до кас або агентств.

Однією з основних переваг системи замовлення квитків онлайн є її доступність цілодобово та з будь-якого пристрою з Інтернет-підключенням. Крім того, користувачі мають можливість порівнювати ціни та вибирати найвигідніші пропозиції, що робить процес бронювання квитків економічним та зручним.

У процесі розробки цього проекту були використані навички програмування в середовищі Visual Studio, а також концепції юзабіліті та дизайну інтерфейсів, надані в лекціях Брусенцової. Зокрема, були враховані вимоги до дизайну інтерфейсів, орієнтованого на користувача, та використані інструменти прототипування для реалізації зручних та привабливих інтерфейсів.

Під час розробки додатку "Система замовлення залізничних квитків" було проведено тестування, яке не виявило суттєвих помилок, що свідчить про його працездатність та надійність. Основні функції програми були успішно продемонстровані, а дизайн інтерфейсу буде піддаватися подальшій оптимізації для поліпшення зручності користувача.

У майбутньому планується продовження робіт над проектом з метою подальшого вдосконалення та розширення функціональності додатку. Використання сучасних технологій та навичок розробки програм дозволить забезпечити найвищий рівень зручності та задоволення від використання для кінцевих користувачів.

# ЛІТЕРАТУРА

1. Андерсон К. Основи Windows Presentation Foundation:
2. Пров. з англ. - М.: ДМК Прес, 2008. - 428 с.
3. Ватсон Б. C# 4.0 на прикладах. - СПб.: БХВ-Петербург, 2011. - 608 с.
4. Гвоздєва Т.В. Проектування інформаційних систем: навч. посібник/Т.В. Гвоздєва, Т.А. Балод. - Ростов н / Д: Фенікс, 2009. - 508 с.
5. Зіборов В.В. Visual C# 2010 на прикладах. - СПб.: БХВ-Петербург, 2011. - 432 с.
6. Мак-Дональд М. Windows Presentation Foundation у .NET
7. 4.0 з прикладами C# для професіоналів: Пер. з англ. - М.: ТОВ «І.Д. Вільямс», 2010. - 1024 с.
8. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition.
9. Брусенцова, Т. П.Б89 Проектування інтерфейсів користувача: посібник для студентів спеціальності 1-47 01 02 «Дизайн електронних та веб-видань» / Т. П. Брусенцова, Т. В. Кішкурно. - Мінськ: БДТУ, 2019. - 172 с.ISBN 978-985-530-799-1.
10. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. Робин Никсон
11. Вивчаємо HTML, XHTML та CSS 2-ге вид. Книга, Робсон Елізабет та Фрімен Ерік
12. Фримен Эрик, Фримен Элизабет. Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. — 1-е изд.
13. Atkinson P., Vieira R. Beginning Microsoft SQL Server 2012 Programming. - Wiley, 2016, – 864с. ([Електронний ресурс] - Режим доступу: https://www.pdfdrive.com/beginning-microsoft-sql-server-2012-programming-e164606534.html)
14. Petkovic D. Microsoft SQL Server 2012: A Beginner's Guide.— McGraw-Hill, 2012. — 795 с. ([Електронний ресурс] - Режим доступу:  https://www.pdfdrive.com/microsoft-sql-server-2012-a-beginners-guide-e185318281.html).
15. Date C.J. SQL and Relational Theory: How to Write Accurate SQL Code. O’Reilly Media, Inc., 2015. — 584 с. ([Електронний ресурс] - Режим доступу: https://www.pdfdrive.com/sql-and-relational-theory-how-to-write-accurate-sql-code-e176192521.html)
16. Troelsen A.,  Japikse P. Pro C# 7: With .NET and .NET Core. - APress. Inc.,2017.- 1372 c. ([Електронний ресурс] - Режим доступу: https://www.pdfdrive.com/pro-c-7-with-net-and-net-core-d183552783.html)
17. Noyes B. Data Binding with Windows Forms 2.0: Programming Smart Client Data Applications with .NET. - Addison Wesley Professional, 2006. - 675с. ([Електронний ресурс] - Режим доступу: <https://www.pdfdrive.com/data-binding-with-windows-forms-20-programming-smart-client-data-applications-with-net-e185459891.html>)

# ДОДАТОК