

ЗМІСТ

ВСТУП.....	4
1 Загальний розділ.....	6
1.1. Постановка завдання	6
1.2. Опис вхідної та результуючої інформації.....	7
1.3. Формалізований опис задачі.....	8
1.4. Опис існуючих методів та рішень	9
2 Розробка технічного та робочого проекту	11
2.1. Проектування архітектури, структур даних, інтерфейсу ПЗ	11
2.2. Розробка та опис програмної реалізації та тестування ПЗ.....	17
3 Спеціальний розділ	21
3.1. Інструкція з інсталяції розробленого проекту	21
3.2. Інструкція з експлуатації проекту	22
ВИСНОВКИ	29
ЛІТЕРАТУРА	30
ДОДАТКИ	31
Додаток А	31
Додаток Б.....	47
Додаток В	48
Додаток Г	49
Додаток Д	50
Додаток Е	51
Додаток Ж	52

					КП.ОК23.КІ.212.05.000.ВП		
Змін.	Аркуш	№ докум.	Підпис	Дата	Сервіс покупки залізничних квитків “Tickets”. Пояснювальна записка		
Розробив	Данюк Д. О.						
Перевірів	Атаман В. О.						
Н.контр.							
Затвер.							
					Літ.	Аркуш	Аркушів
					н	3	52
					ХПК НУ «ЛП»		

ВСТУП

Залізничний транспорт історично вважається одним із найважливіших засобів масового перевезення, що забезпечує надійність та ефективність у переміщенні пасажирів і вантажів. Він є важливою складовою інфраструктури багатьох країн, забезпечуючи зв'язок між регіонами, містами та селами.

Згідно зі статистикою, потяги залишаються одним з найпопулярніших видів транспорту в Україні. Щорічно мільйони пасажирів користуються послугами залізниці для подорожей між містами та регіонами країни. Це відображається у великій кількості рейсів, що обслуговуються, а також у високій загальній кількості пасажирів, що перевозяться щодня.

Однією з ключових переваг залізничного транспорту є його економічна вигода. У порівнянні з іншими видами транспорту, такими як автомобільний або повітряний, він зазвичай є менш витратним, особливо коли мова йде про великі відстані та масові перевезення. Це дозволяє ефективно використовувати ресурси та зменшує витрати на перевезення. Крім того, залізничний транспорт є менш залежним від погодних умов, що забезпечує його надійність та регулярність. Завдяки цьому користувачі можуть розраховувати на стабільні графіки та мінімальні затримки.

Основний недолік користування залізницею виникає в процесі купівлі квитків, що часто може стати справжнім випробуванням для пасажирів. Традиційні методи придбання квитків, такі як особисте відвідування залізничних кас часто супроводжуються низкою проблем. Стрімкі черги на станціях, обмежена доступність квитків у певні дати чи на популярні маршрути, складність користування деякими веб-платформами - це лише кілька з труднощів, з якими можуть зіткнутися пасажирів.

Автоматизація процесу покупки залізничних квитків надає користувачам можливість зручно та швидко придбати білети без необхідності витрачати час на черги або пошуки оптимальних варіантів. Це значно підвищує зручність, дозволяючи здійснювати покупку квитків у будь-який час і з будь-якого місця,

					КП.ОК23.КІ.212.05.000.ВП	Арк.
Вим.	Арк.	№ докум.	Підпис	Дат		4

де є доступ до Інтернету. Крім того, автоматизація процесу дозволяє зменшити ймовірність допущення помилок, таких як неправильно введені дані або помилки у виборі рейсів, оскільки система автоматично перевіряє та коригує інформацію.

Основною метою роботи було створення застосунка, який надає користувачам зручний та інтуїтивно зрозумілий інтерфейс для покупки білетів. Завдяки цьому додатку, користувачі змогли б швидко знайти потрібний маршрут, вибрати зручний для себе час і місце, та здійснити оплату всього кількома кліками, не виходячи з дому. Крім того, користувачі матимуть змогу отримати всю необхідну інформацію про маршрути, графіки руху поїздів, а також переглянути історію своїх покупок, що значно підвищує рівень зручності та задоволення від використання переваг залізничного транспорту.

Крім того, залізничні компанії можуть використовувати дані, зібрані через додаток, для аналізу популярності маршрутів, виявлення патернів пасажиропотоку та оптимізації розкладів. Це дозволить компанії відповідати на потреби ринку, підвищуючи ефективність роботи та рівень задоволеності пасажирів. Аналіз даних допоможе виявити пікові періоди навантаження на окремих напрямках, що сприятиме кращій організації руху та запобіганню перевантаженням.

Повна автоматизація процесу купівлі квитків через додаток зменшить витрати часу та зусиль пасажирів, а також сприятиме збільшенню обсягів продажів для залізничних перевізників. Це також підвищить загальну ефективність роботи залізничних компаній, дозволяючи їм швидко адаптуватися до змін попиту та оптимізувати свої ресурси. У перспективі, інтеграція додаткових функцій, таких як відстеження рейсів у реальному часі та надання інформації про затримки, ще більше покращить користувацький досвід та зробить подорожі залізницею більш приємними

					КП.ОК23.КІ.212.05.000.ВП	Арк.
Вим.	Арк.	№ докум.	Підпис	Дат		5

1 Загальний розділ

1.1. Постановка завдання

В сьогодення, коли залізничний транспорт є ключовим для переміщення пасажирів, стає важливим автоматизувати процес купівлі залізничних квитків. Існуюча система придбання квитків потребує постійних покращень.

Для досягнення поставленої цілі було сформовано низку завдань, які необхідно вирішити і реалізувати:

- Створення облікового запису користувача: реєстрація нового користувача з обов'язковими полями для особистої інформації, необхідної для придбання квитків. Можливість авторизації для зареєстрованих користувачів, та відновлення паролю в разі необхідності.

- Інтерфейс для пошуку та перегляду квитків: створення зручного та інтуїтивно зрозумілого інтерфейсу. Можливість швидкого пошуку квитків за різними критеріями (дата, час, маршрут тощо). Перегляд доступних квитків з відображенням всієї необхідної інформації. Доступ до фотографій різних типів вагонів та поїздів для ознайомлення користувачів з умовами подорожі.

- Безпека та конфіденційності: забезпечити захист особистих даних користувачів у базі даних, особливо паролів, з використанням хеш-функцій.

- Надання додаткових послуг: можливість отримати придбати їжу, напої, забронювати постільну білизну.

Для створення застосунку було обрано мову програмування C#, яка є високорівневою мовою програмування та широко використовується для розробки програмного забезпечення для операційних систем Windows. Для створення графічного інтерфейсу додатку буде використано технологію Windows Forms (WinForms), яка надає зручний інтерфейс для розробки програм з використанням компонентів GUI.

					КП.ОК23.КІ.212.05.000.ВП	Арк.
						6
Вим.	Арк.	№ докум.	Підпис	Дат		

Збереження та обробка особистих даних користувачів відбувається за допомогою реляційної бази даних MySQL Server, що для наочності встановлена на локальному сервері MAMP.

1.2. Опис вхідної та результуючої інформації

Для створення моделі проекту необхідно докладно розглянути та врахувати різні типи інформації, з якою буде взаємодіяти програма. В результаті отримаємо наступний перелік:

- користувач: прізвище, ім'я, електронна адреса, пароль;
- місто: назва, час прибуття, час відправки;
- вагон: назва, тип вагону, ціна місце;
- потяг: назва, модель, фото, список вагонів;
- маршрут: назва, потяг, міста зупинок.

Аналіз вхідної інформації допомагає виявити потенційні проблеми та недоліки в архітектурі програмного забезпечення та невідповідності у вимогах до системи. Це дозволяє вчасно внести необхідні зміни та уникнути потенційних конфліктів.

Вся вхідна інформація вводиться користувачами через спеціально розроблені форми. Отримані дані будуть зберігатись у базі даних, яка буде інтегрована з програмою. Це забезпечить їх надійність і доступність для подальшого використання, а також спростить роботу та обробку інформації.

Вихідна інформація є результатом обробки вхідної та використовується для подальшого аналізу, прийняття рішень та взаємодії з користувачем. Отримаємо наступний перелік необхідної результуючої інформації:

- список рекомендованих маршрутів;
- список типів місць;
- список вільних та зайнятих місць;
- список білетів кожного користувача;

					КП.ОК23.КІ.212.05.000.ВП	Арк.
Вим.	Арк.	№ докум.	Підпис	Дат		7

Дослідження результуючої інформації дозволяє оцінити ефективність програми, виявити можливі проблеми та недоліки, а також знайти шляхи для їх подальшого вдосконалення та оптимізації. Зокрема, визначити частоту виникнення певних подій або сценаріїв щоб визначити їхню важливість та пріоритетність.

1.3. Формалізований опис задачі

Формалізований опис задачі використовується для чіткого формулювання вимог до програмного продукту перед початком розробки. Він допомагає зрозуміти завдання, яке потрібно вирішити, і визначити обсяг робіт. Цей опис повинен містити деталі про функціональність системи, вхідні та вихідні дані, специфікації її компонентів та взаємозв'язки між ними.

Для створення моделі програмного продукту було використано мову моделювання об'єктів Unified Modeling Language (UML). Вона надає стандартизований набір графічних засобів для опису структури та поведінки системи, що дозволяє візуалізувати всі аспекти програмного продукту.

У моделі UML для проекту використовуються 6 різних діаграм, для детального опису його архітектури та функціональності:

Діаграма випадку використання (Use Case Diagram), що зображена на додатку Б в моделі UML використовується для візуалізації функціональності системи та взаємодії її акторів (користувачів) з цією системою. Кожен випадок використання представляє собою конкретну функцію або функціональний сценарій системи, який може бути виконаний користувачами. Діаграма випадку використання допомагає розуміти, як користувачі будуть взаємодіяти з системою, та служить основою для подальшої розробки моделі продукту.

Діаграма станів (Statechart Diagram) на додатку В моделює різні стани, у яких може перебувати об'єкт або система, а також для відображення переходів між ними в залежності від вхідних подій або умов.

Діаграми активності (Activity Diagram) на додатку Г необхідна для моделювання послідовності дій та процесів в системі. Вони дозволяють візуалізувати, які кроки потрібно виконати для досягнення певної мети або результату та допомагають розібратися в логіці взаємодії між різними елементами системи щоб виявити можливі проблеми або неясності у процесах.

Діаграма послідовності (Sequence Diagram) на додатку Д в моделі UML використовується для візуалізації взаємодії між об'єктами в системі у певний момент часу. Вона показує послідовність повідомлень, які обмінюються об'єктами, та порядок їх виконання.

Діаграма співпраці (Collaboration Diagram) на додатку Е відображає взаємодію об'єктів в системі та їх взаємовідносини в рамках конкретного сценарію виконання

Діаграми класів (Class Diagrams) на додатку Ж використовуються для проектування структури системи шляхом відображення класів, їх атрибутів, методів та взаємозв'язків між класами. Кожен клас відображається у вигляді прямокутника з ім'ям класу та його полями. Зв'язки між класами представлені стрілками, які показують взаємозв'язки, такі як асоціація, наслідування, композиція тощо. Діаграми класів допомагають розуміти структуру програми, визначити класи та їх взаємозв'язки.

Використання UML діаграм забезпечує стандартизований і зрозумілий спосіб візуалізації структури та поведінки системи. Вони допомагають виявити потенційні проблеми на ранніх етапах розробки, покращують планування проекту, забезпечують можливість моделювання складних систем і сприяють створення зручної документації проекту для подальшого обслуговування та розвитку.

1.4. Опис існуючих методів та рішень

Для розробки проекту важливо обрати вдалі інструменти та технології. Приклад аналогічного проекту, сайту www.uz.gov.ua, який має свої переваги та

					КП.ОК23.КІ.212.05.000.ВП	Арк.
Вим.	Арк.	№ докум.	Підпис	Дат		9

недоліки, надав цінний досвід для розробки мого проекту. Зокрема, сайт www.uz.gov.ua пропонує широкий спектр можливостей для користувачів, включаючи зручний інтерфейс для пошуку та купівлі квитків, детальну інформацію про розклад поїздів, а також можливість перевірки наявності місць у вагонах. Однак, існують і певні недоліки, такі як періодичні технічні збої, незручності в процесі реєстрації та авторизації користувачів, а також обмежена функціональність мобільної версії сайту. У своєму проекті я намагався врахувати ці аспекти та забезпечити більш стабільну роботу системи та спрощену процедуру реєстрації та входу. Крім того, мій проект включає додаткові функції, такі як можливість переглядати фото потягів та, що дозволить користувачам заздалегідь оцінити рівень комфорту та вибрати найкращий варіант для своїх потреб і уподобань.

Розглядаючи вже існуючі рішення, можна здійснити аналіз їхніх переваг та недоліків, а також врахувати власні потреби та особливості проекту для вибору найбільш оптимального напрямку розробки.

Мовою програмування було взято C#, яка відома своєю високою продуктивністю, ефективністю та широкими можливостями для розробки різноманітних додатків. Основні переваги:

- Об'єктно-орієнтованість: C# підтримує потужну об'єктно-орієнтовану парадигму, що дозволяє легко організувати програму у вигляді класів і об'єктів, сприяючи чіткій структурі і підтримці розширюваності;
- Широкі можливості розробки: C# має велику кількість стандартних бібліотек і фреймворків, що спрощує розробку різноманітних програм;
- Підтримка мультиплатформеності: За допомогою .NET ви можете розробляти застосунки для різних платформ сімейства Windows та Unix.
- Безпека: C# має вбудовану підтримку для безпеки, включаючи механізми обробки винятків та контроль типів, що допоможе уникнути багатьох потенційних помилок та забезпечить надійність сервісу.

					КП.ОК23.КІ.212.05.000.ВР	Арк.
Вим.	Арк.	№ докум.	Підпис	Дат		10

2 Розробка технічного та робочого проекту

2.1. Проектування архітектури, структур даних, інтерфейсу ПЗ

Щоб розпочати написання коду, потрібно розробити шаблон архітектури та графічного інтерфейсу додатка, який буде визначати структуру, відношення та поведінку всіх компонентів системи. Також слід врахувати особливості взаємодії з користувачем та вимоги до ефективності та безпеки програмного забезпечення.

Проектування архітектури ПЗ передбачає створення класів, що відображатимуть різні аспекти функціоналу програми. Спочатку необхідно спроектувати класи-моделі для представлення даних про користувачів, вагонів, потягів та маршрутів. Кожен клас-модель повинен мати властивості, поля та методи, що відповідають за інформацію про об'єкти цих класів.

Для роботи з особистими даними користувачів буде використана система управління базами даних (СУБД) MySQL, що надає можливість зберігати та керувати даними, які зберігаються у вигляді таблиць. MySQL використовує мову запитів SQL (Structured Query Language) для взаємодії з базою даних, що дозволяє створювати, змінювати, видаляти та запитувати дані. Таблиця буде містити наступні поля:

- ім'я користувача;
- прізвище користувача;
- електронна адреса;
- пароль;
- ключове поле id

Для збереження пароля користувача буде використовуватись алгоритм хешування SHA-256 (Secure Hash Algorithm 256-bit). В результаті отримаємо унікальний, незворотний рядок (хеш), який може бути збережений у базі безпечно. Коли користувач намагається увійти до системи, його введений пароль також хешується і порівнюється зі збереженим хешем. Якщо вони

співпадають, це означає, що введений користувачем пароль є вірним, і користувачу надається доступ до системи. Використання алгоритму хешування забезпечує безпеку паролів користувачів, оскільки навіть у випадку отримання несанкціонованого доступу до бази даних пароль залишиться засекреченим.

Для графічного відображення бази скористався сервісом phpMyAdmin. Він надає зручний спосіб виконання різноманітних завдань управління базою даних, таких як створення, видалення та редагування таблиць, виконання SQL-запитів, імпорт та експорт даних, керування користувачами та їх привілеями за допомогою простих форм та інтерактивних елементів.

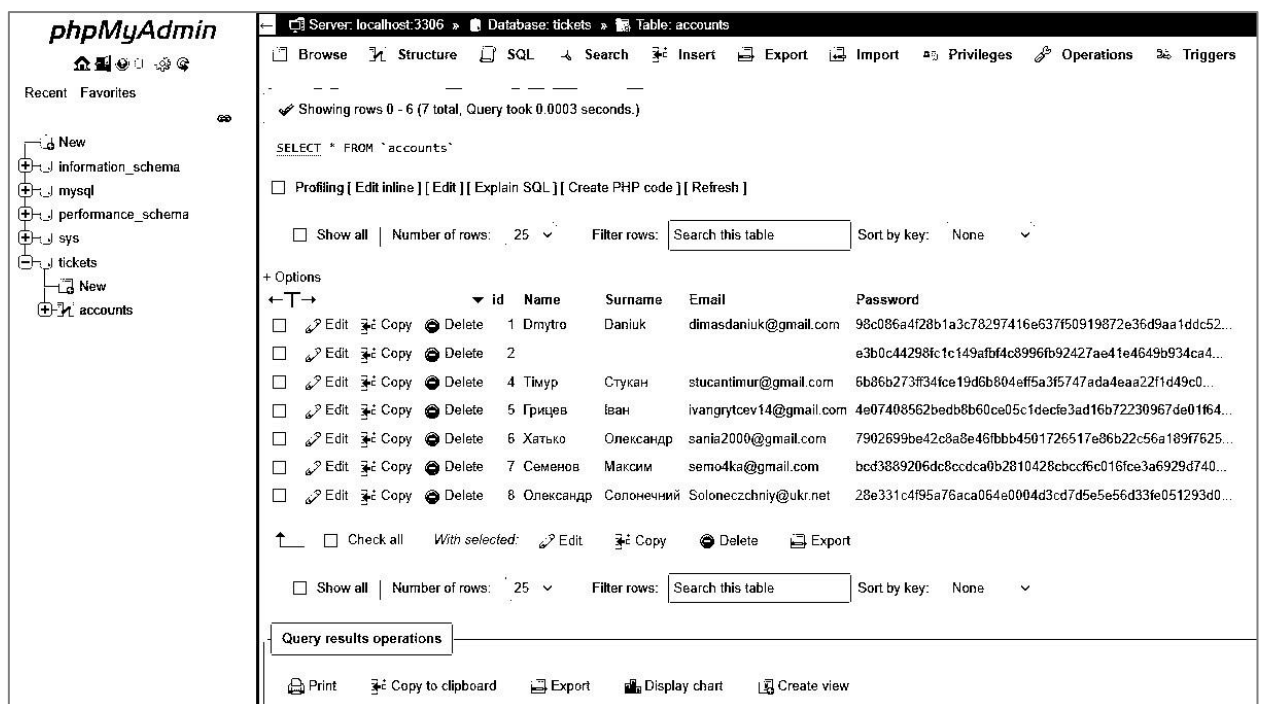


Рисунок 1 – Структура бази даних

Після визначення архітектури, перейшов до наступного кроку, а саме створення макету інтерфейсу. Для цього було вирішено використовувати графічний редактор Figma, оскільки він надає можливості для проектування з урахуванням сучасних дизайн-тенденцій. У макеті було передбачено зручний та інтуїтивно зрозумілий інтерфейс для користувачів, де вони можуть легко знаходити необхідну інформацію і виконувати потрібні дії.

На першій формі будуть розміщені елементи для авторизації користувача, де йому необхідно ввести логін та пароль, а також буде доступна можливість

переходу на форму реєстрації або відновлення паролю. Також на ній буде розміщений логотип. Шаблон розміщений на рисунку 2.

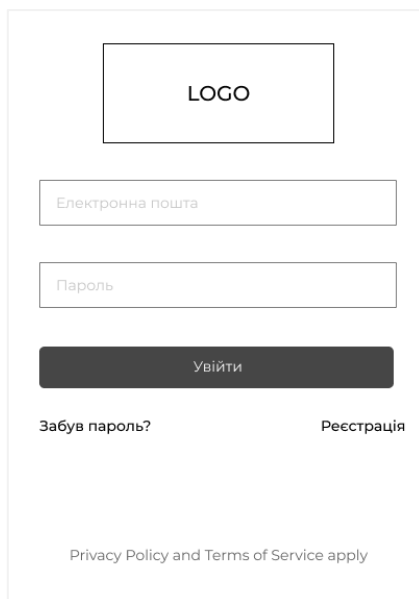


Diagram of a login form template. It features a central box with a 'LOGO' placeholder at the top. Below the logo are two input fields: 'Електронна пошта' (Email) and 'Пароль' (Password). A dark button labeled 'Увійти' (Login) is positioned below the password field. At the bottom of the form, there are two links: 'Забув пароль?' (Forgot password?) on the left and 'Реєстрація' (Registration) on the right. A footer line states 'Privacy Policy and Terms of Service apply'.

Рисунок 2 – Шаблон форми входу

При натисканні користувачем на напис "реєстрація" відкривається відповідна форма, у якій необхідно заповнити необхідну інформацію для створення облікового запису

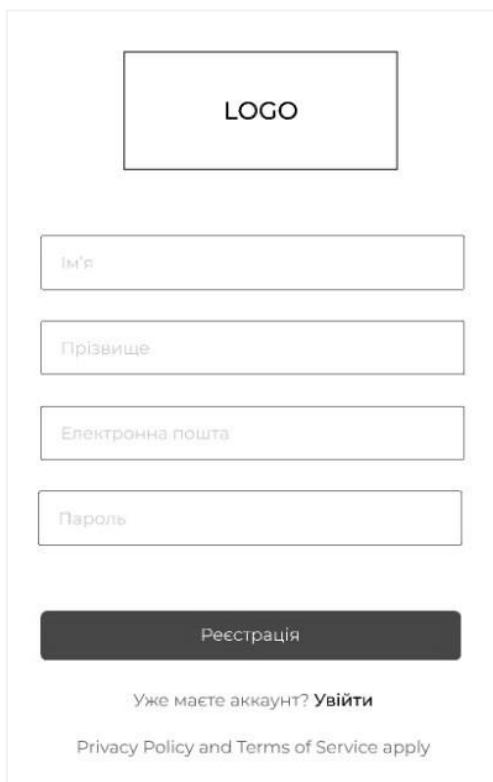


Diagram of a registration form template. It features a central box with a 'LOGO' placeholder at the top. Below the logo are four input fields: 'Ім'я' (Name), 'Прізвище' (Surname), 'Електронна пошта' (Email), and 'Пароль' (Password). A dark button labeled 'Реєстрація' (Registration) is positioned below the password field. Below the button, there is a link 'Уже маєте аккаунт? Увійти' (Already have an account? Login). A footer line states 'Privacy Policy and Terms of Service apply'.

Рисунок 3 – Шаблон форми реєстрації

Форма відновлення пароля містить одне поле, яке приймає логін і перевіряє існує такий користувач у базі даних.

Рисунок 4 – Шаблон форми відновлення пароля

Після успішного входу відкривається головна форма, на якій виводиться список рекомендованих маршрутів, з відображенням початкової та кінцевої станцій, часом їхнього прибуття та тривалістю поїздки.

Рисунок 5 – Шаблон списку рекомендованих маршрутів

Щоб знайти необхідний маршрут, розроблено спеціальний шаблон, який дозволяє користувачам вказати критерії для пошуку квитків. Цей шаблон надає можливість встановлювати такі параметри, як місто відправлення, прибуття та дату.

Рисунок 6 – Шаблон пошуку маршрута

Після натискання кнопки "купити" відкривається спеціальна форма, де користувач може обрати тип вагону та конкретне місце. Відображаються лише доступні варіанти вагонів. Також на цій формі виводиться ціна відповідно до обраного класу вагону.

Рисунок 7 – Шаблон форми вибору вагона та місця

Після вибору місця на квитку наступним кроком є заповнення контактної інформації та даних для оплати. Користувачеві потрібно буде вказати дані платіжної карти для здійснення оплати за квиток. Також на цій формі можна оформити додаткові послуги.

Довідка | Україна | Мій квиток

Звідки: Хмельницький | Куди: Львів | Туди: 13.01.2024 | Назад

Пошук

1. Вибір поїзда | 2. Вибір місць | 3. Вкажіть данні пасажирів

← Вибрати інше місце

Контактна інформація

Ім'я:

Прізвище:

Електронна пошта:

Введіть данні для оплати

Номер карти:

Термін дії: /

Додаткові послуги

☐ Фірмове печиво 40₴

☐ Фірмове печиво 40₴

☐ Авторський чай 30₴

☐ Дріп кава 70₴

☐ Чайний збір 38₴

▼ Маршрут

Всього 244₴

Рисунок 8 – Шаблон форми заповнення контактної інформації

Після успішної оплати користувач може переглянути список придбаних квитків у відповідній формі. У цьому розділі будуть відображені всі деталі куплених квитків.

Вітаємо, ми раді Вас бачити

Львів - Київ 29.03

09:00 8 год. 0 хв. 17:00 Вагон: 1 Місце: 20

Хмельницький - Кривий ріг 31.03

08:40 11 год. 12 хв. 19:52 Вагон: 1 Місце: 26

Рисунок 9 – Шаблон форми перегляду придбаних квитків

На формі довідки розміщена контактна інформація.

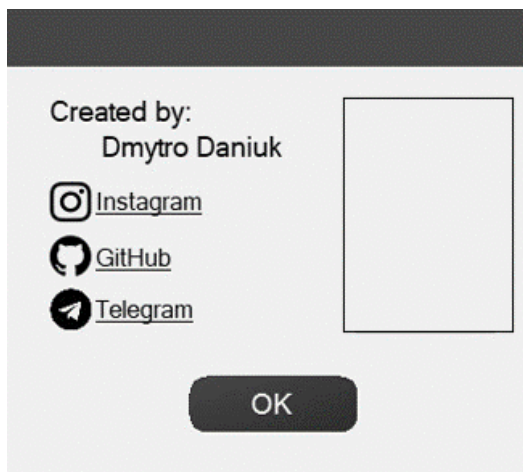


Рисунок 10 – Шаблон форми довідки

Після завершення описаних вище кроків отримаємо пророблену модель проекту програмного забезпечення, готову до фази реалізації. Завдяки цьому проектуванню можна чітко уявити, як має виглядати фінальний додаток, і які функції потрібно реалізувати.

2.2. Розробка та опис програмної реалізації

Після того як було створено додаток, він мав структуру файлів, зображену на рисунку 11.

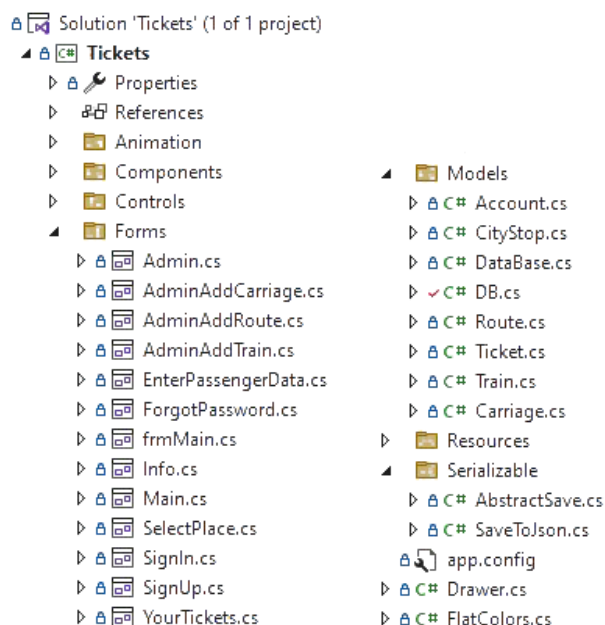


Рисунок 11 – Файлова структура програми

Для збереження даних про маршрути було використано бібліотеку System.Text.Json, яка забезпечить зручний та ефективний спосіб серіалізації та десеріалізації об'єктів у формат JSON. Це дозволить зберігати дані про маршрути у структурованому форматі, що спростить їх подальшу обробку та використання у програмі.

Щоб вести адміністрування потягів, маршрутів, білетів та користувачів були написані класи: Account.cs, Carriage.cs, Train.cs, CityStop.cs, Route.cs, Ticket.cs, кожен з яких має свої поля та методи.

Код класу Account:

```
public class Account
{
    public string Name { get; private set; }
    public string Surname { get; private set; }
    public string Email { get; private set; }
    public string Password { get; private set; }

    public Account(string name, string surname, string email, string password)
    {
        Name = name;
        Surname = surname;
        Email = email;
        Password = password;
    }

    public void ChangePassword(string newPassword)
    {
        Password = newPassword;
    }
    public bool IsPasswordValid(string password)
    {
        return password.Length >= 8;
    }
    public string GetAccountInfo()
    {
        return $"Name: {Name}, Surname: {Surname}, Email: {Email}";
    }
    public bool CheckPassword(string inputPassword)
    {
        return Password == inputPassword;
    }
    public static bool operator ==(Account a, Account b)
    {
        if (ReferenceEquals(a, b))
        {
```



```

        return true;
    }
    if (a is null || b is null)
    {
        return false;
    }
    return a.Name == b.Name && a.Surname == b.Surname && a.Email == b.Email &&
a.Password == b.Password;
}
public static bool operator !=(Account a, Account b)
{
    return !(a == b);
}
}

```

Клас Ticket відповідає за створення, збереження та обробку інформації про квиток на поїздку. Його основна функція полягає у збереженні деталей щодо облікового запису користувача, обраного маршруту, вагона та місця, а також додаткової інформації та ціни квитка.

Код класу Ticket:

```

public class Ticket
{
    public Account LoggedAccount { get; private set; }
    public Route SelectedRoute { get; private set; }
    public Carriage SelectedCarriage { get; private set; }
    public int SelectedCarriageSeat { get; private set; }
    public List<CityStop> SelectedCities { get; private set; }
    public List<String> Additional { get; private set; }
    public Double Price { get; private set; }

    [JsonConstructor]
    public Ticket(Account loggedAccount, Route selectedRoute, Carriage selectedCarriage, int
selectedCarriageSeat, List<CityStop> selectedCities, List<string> additional, double price)
    {
        LoggedAccount = loggedAccount;
        SelectedRoute = selectedRoute;
        SelectedCarriage = selectedCarriage;
        SelectedCarriageSeat = selectedCarriageSeat;
        SelectedCities = selectedCities;
        Additional = additional;
        Price = price;
    }

    public Ticket(Account loggedaccount, Route selectedroute, List<CityStop> selectedcities, List<string>
additional)
    {
        LoggedAccount = loggedaccount;
        SelectedRoute = selectedroute;
        SelectedCities = selectedcities;
        Additional = additional;
    }
    public Route GetSelectedRoute()
    {
        return SelectedRoute;
    }
}

```

```

    }
    public void EditAdditionalInfo(string oldInfo, string newInfo)
    {
        if (Additional.Contains(oldInfo))
        {
            int index = Additional.IndexOf(oldInfo);
            Additional[index] = newInfo;
        }
    }
    public void AddPlace(Carriage selectedcarriage, int selectedcarriageseat)
    {
        SelectedCarriage = selectedcarriage;
        SelectedCarriageSeat = selectedcarriageseat;
    }
    public void SetPrice (double price)
    {
        Price = price;
    }

    public void AddAdditionalInfo(string info)
    {
        Additional.Add(info);
    }
    public void RemoveAdditionalInfo(string info)
    {
        Additional.Remove(info);
    }
}

```

Для створення візуальної частини програми розроблено форми: SignIn.cs, SignUp.cs, Main.cs, SelectPlace.cs, EnterPassengerData.cs, YourTickets.cs. Кожна з цих форм відповідає за конкретний етап виконання операцій та надає користувачеві необхідний функціонал для виконання потрібних завдань, від авторизації та реєстрації до вибору місця та перегляду придбаних квитків. Кожна форма має свій власний набір елементів керування та інтерфейсу, що відповідає її функціональності та спрощує взаємодію з програмою для користувача.

Форма SignIn відповідає за процес авторизації користувача в системі. Вона містить необхідні елементи інтерфейсу, які дозволяють вводити логін та пароль. Після введення цих даних користувачем виконується функція, що перевіряє правильність введених даних та виконують процес авторизації. Якщо дані введено вірно, користувач отримує доступ до головного інтерфейсу програми.

3 Спеціальний розділ

3.1. Інструкція з інсталяції розробленого проекту.

Для повноцінної роботи додатку пристрій повинен відповідати рекомендованим системним вимогам:

- оперативна пам'ять 100 Мб.
- процесор із тактовою частотою не нижче 1,8 ГГц;
- 2 ГБ ОЗУ;
- 50 МБ вільного місця на дисковому просторі
- DirectX версії 11.0

3.1. Інструкція з інсталяції розробленого проекту

Щоб інсталювати додаток потрібно завантажити і запустити установочний файл Setup.exe.



Рисунок 11 – Файл для інсталяції додатку

Після завантаження, необхідно запустити файл, щоб розпочати процес інсталяції. Інсталятор буде пропонувати крок за кроком інструкції з установки, де користувач може вибрати шлях для встановлення додатку, та налаштувати параметри інсталяції.

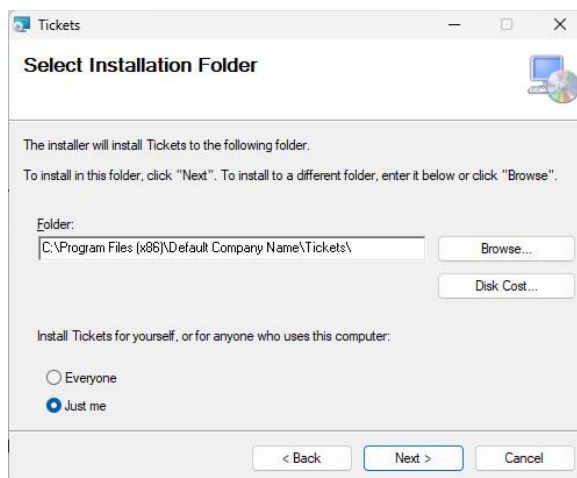


Рисунок 12 – Параметри інсталяції

					КП.ОК23.КІ.212.05.000.ВП	Арк.
Вим.	Арк.	№ докум.	Підпис	Дат		21

При успішній інсталяції додатку на екрані з'явиться вікно з відповідним повідомленням.

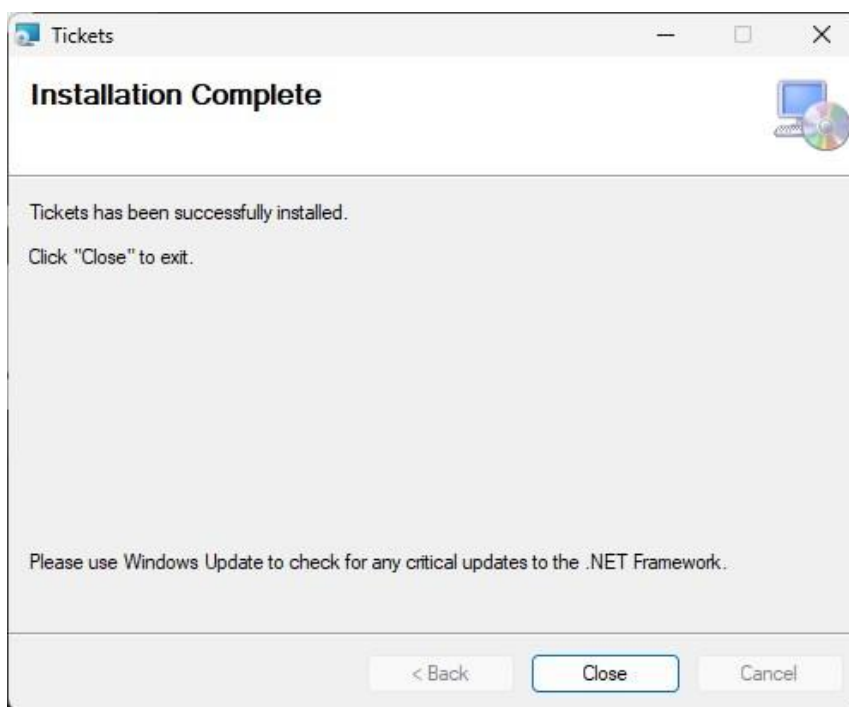


Рисунок 13 – Завершення інсталяції

3.2. Інструкція з експлуатації проекту

Після успішної інсталяції програми, щоб розпочати роботу з нею, користувачу достатньо просто подвійно клацнути на ярлику програми на робочому столі. Це запустить програму і відкриє головне вікно, готове до використання.

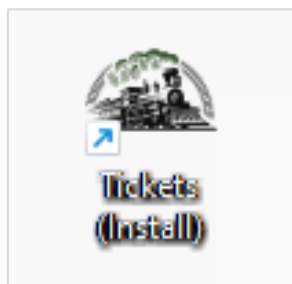


Рисунок 14 – Вигляд ярлика програми

Під час першого запуску програми користувачеві буде запропоновано створити обліковий запис, що відбувається через вікно реєстрації. Тут користувач повинен буде ввести свої особисті дані, такі як ім'я, прізвище,

адресу електронної пошти та пароль. Ці дані використовуються для створення облікового запису користувача та його подальшого використання при вході в систему. Отриманні будуть збережені у базі даних, готові до подальшого використання.

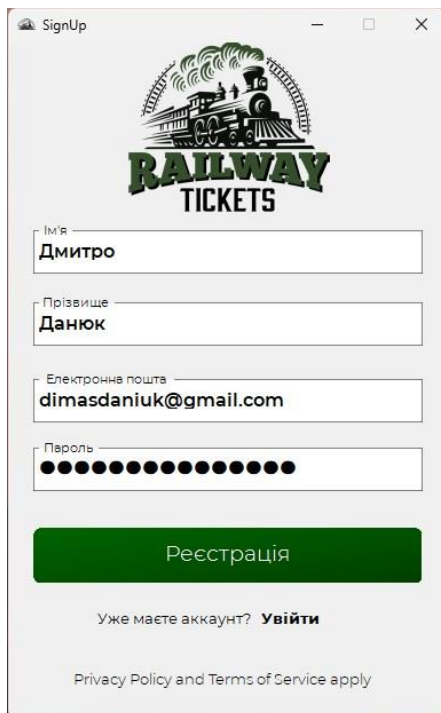


Рисунок 15 – Вікно реєстрації

У випадку, якщо користувач забув свій пароль, він може відновити його за допомогою функції "Відновлення паролю". На відповідній сторінці, користувачу буде запропоновано ввести свою електронну адресу, яку він використовував при реєстрації облікового запису.

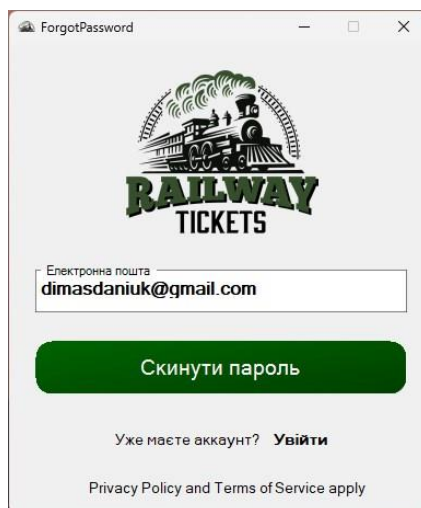


Рисунок 16 – Вікно відновлення пароля

					КП.ОК23.КІ.212.05.000.ВП	Арк.
Вим.	Арк.	№ докум.	Підпис	Дат		23

Після успішної реєстрації користувач може використовувати свої облікові дані - логін та пароль, для входу у свій обліковий запис. Цей процес виконується через вікно авторизації

Рисунок 17 – Вікно авторизації

При успішному вході в систему користувачу автоматично відкривається головна форма додатку, де він може переглянути список рекомендованих маршрутів. На цій сторінці користувачеві надається зручний та інтуїтивно зрозумілий інтерфейс для вибору потрібного маршруту. Кожен маршрут бути відображається з деталями, такими як час відправлення, час прибуття, тривалість подорожі, та інші параметри.

Рисунок 18 – Головна форма

На головній формі додатку реалізовані елементи керування, які дозволяють користувачам швидко та зручно знаходити необхідний маршрут. Використовуючи їх, користувач може здійснювати пошук за різними параметрами, такими як місце відправлення, місце прибуття, дата відправлення та інші.

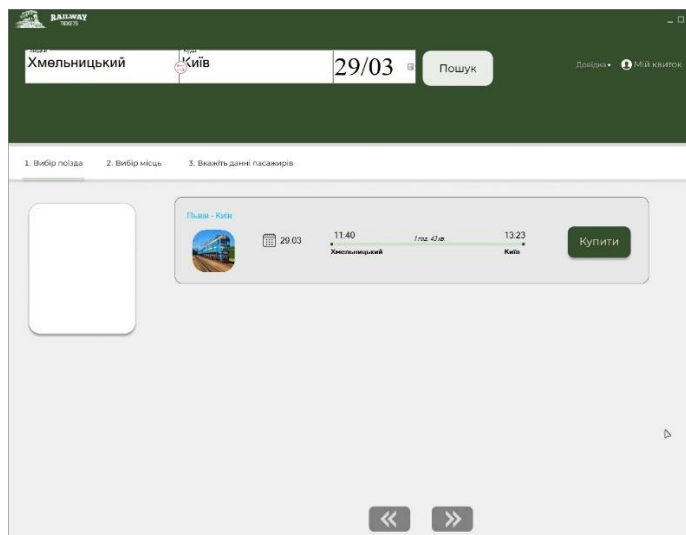


Рисунок 19 – Пошук маршрута

Після натискання кнопки "Купити" користувач переходить на відповідну форму, де йому доступний перегляд вільних місць у вагонах. На цій формі реалізована можливість вибору конкретного вагона та вибір місця з доступних варіантів.

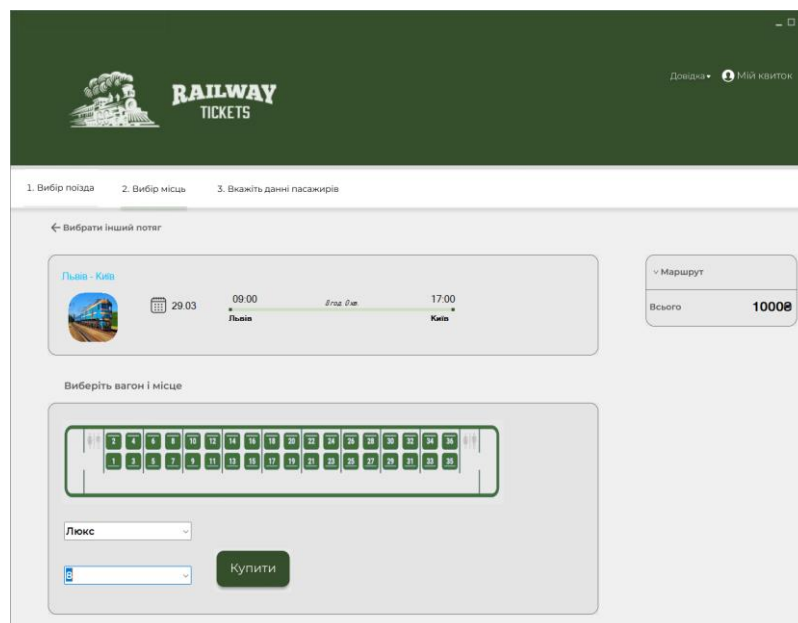


Рисунок 20 – Форма вибору місць

Для оформлення квитка користувачу необхідно ввести дані пасажира, а також надати дані для оплати. Для зручності та максимального комфорту під час подорожі користувач може вибрати додаткові послуги, такі як постіль, їжа та напої, які будуть враховані при оформленні квитка.

Рисунок 21 – Форма оформлення квитка

У відповідній формі користувач може переглянути всі придбані квитки. Це дозволяє зручно, перевіряти інформацію про маршрути, дати відправлення та прибуття, а також переглядати важливі дані про пасажирів та додаткові послуги, які були обрані під час придбання квитків.

Рисунок 22 – Форма перегляду квитків

При успішній авторизації в якості адміністратора, користувач отримує доступ до спеціальної форми для керування вагонами, потягами та маршрутами.

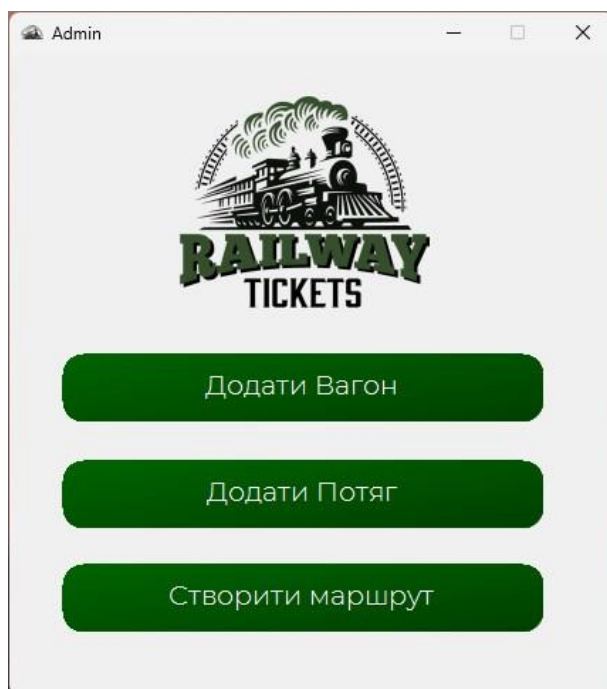


Рисунок 23 – Форма адміністрації

Для додавання нового вагона адміністратор заповнює необхідні поля, такі як Назва, Тип, ціна за місце та кількість місць. Ця інформація дозволяє системі належним чином ідентифікувати та організувати новий вагон у системі.

The image shows a software window titled 'AdminAddCarriage'. It contains four text input fields stacked vertically, each with a placeholder label: 'Назва' (Name), 'Тип вагона' (Carriage type), 'Ціна за місце' (Price per seat), and 'Кількість місць' (Number of seats). Below these fields is a single green button with white text labeled 'Додати вагон' (Add carriage). The window has standard OS controls in the top right corner.

Рисунок 24 – Форма додавання вагону

Щоб додати новий потяг адміністратор повинен вказати назву потягу, його модель та додати фотографію, що допомагає ідентифікувати та рекламувати потяг у системі. Окрім цього, необхідно вибрати список вагонів, які будуть складовою частиною даного потягу.

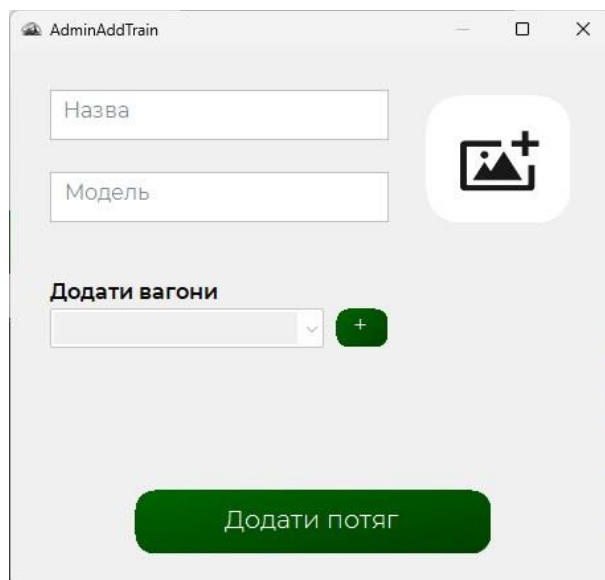


Рисунок 25 – Форма додавання потягу

При створенні нового маршруту адміністратору необхідно вказати назву маршруту, обрати потяг, який буде здійснювати подорож, визначити дату відправлення та вказати список міст, на яких заплановано робити зупинки під час подорожі.

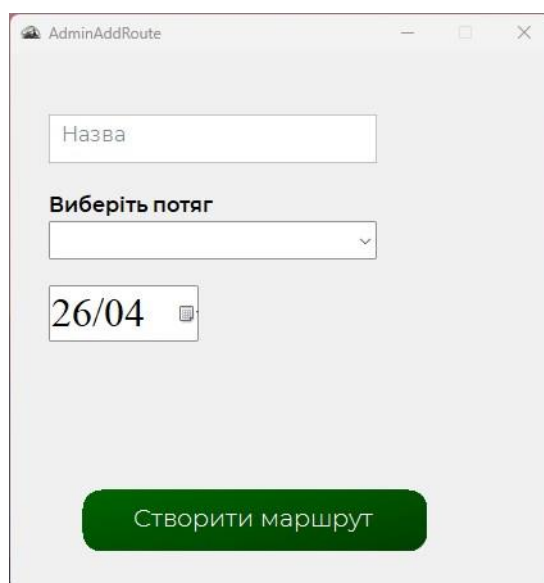


Рисунок 26 – Форма створення маршруту

ВИСНОВКИ

В результаті виконання курсової роботи "Залізничні квитки" було розроблено програмний продукт, що забезпечив користувачам можливість зручного та ефективного бронювання залізничних квитків.

В процесі розробки було створено модель для систематизації програми та визначення необхідних класів програми та їх взаємозв'язків за принципами об'єктно-орієнтованого програмування. Відповідно до візуального макету розроблено графічний інтерфейс, який включав у себе всі необхідні функціональні можливості для зручної взаємодії з користувачем.

Виконуючи роботу я отримав навички у сфері програмування та розробки. Зокрема, цей проєкт допоміг поглибити мої знання з об'єктно-орієнтованого програмування (ООП) та дозволив мені отримати практичний досвід у використанні сучасних технологій та інструментів розробки.

Під час розробки проєкту я вивчав та використовував різні підходи до розробки програмного забезпечення, включаючи створення архітектури програми, реалізацію функціональності та взаємодію з базою даних. Цей досвід допоміг мені розвинути вміння аналізувати задачі, розробляти та впроваджувати ефективні рішення.

Результатом реалізації курсової роботи є програмний продукт, що реалізує автоматизацію покупок залізничних квитків, демонструє використання сучасних підходів до програмної розробки, відповідає вимогам чистого коду та принципам Solid. Проєкт демонструє можливість подальшого розвитку системи та її інтеграції з іншими сервісами для забезпечення більшої зручності функціональності.

					КП.ОК23.КІ.212.05.000.ВП	Арк.
Вим.	Арк.	№ докум.	Підпис	Дат		29

ЛІТЕРАТУРА

1. Albahari J. C# 10 in a Nutshell: The Definitive Reference. O'Reilly Media, Incorporated, 2022.
2. Мартін Р. С. Чистий код. Створення і рефакторинг. Харків: Фабула, 2019. 448 с.
3. Skeet J. C# in depth. 2nd ed. Stamford, CT : Manning, 2011. 554 p.
4. Taher R. Hands-On Object-Oriented Programming with C#: Build Maintainable Software with Reusable Code Using C#. Packt Publishing, Limited, 2019.
5. Bhargava A. Grokking Algorithms: An illustrated guide for programmers and other curious people. Manning Publications, 2016. 256 p.
6. Мартін Р. С. Чиста архітектура. Мистецтво розробки програмного забезпечення. Харків: Фабула, 2022. 368 с.
7. Казимир В. В. Об'єктно-орієнтоване програмування: навчальний посібник для студентів вищих навчальних закладів Київ : Слово, 2008. 192 с.
8. freeCodeCamp.org. C# Tutorial - Full Course for Beginners, 2018. YouTube. URL: <https://www.youtube.com/watch?v=GhQdIIFylQ8>
9. freeCodeCamp.org. Learn C# Programming – Full Course with Mini-Projects, 2024. YouTube. URL: <https://www.youtube.com/watch?v=YrtFtdTTfv0>
10. Traversy Media. MySQL Crash Course | Learn SQL, 2019. YouTube. URL: <https://www.youtube.com/watch?v=9ylj9NR0Lcg>

ДОДАТКИ

Додаток А

Код програми

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

namespace Tickets.Models
{
    public class Account
    {
        public string Name { get; private set; }
        public string Surname { get; private set; }
        public string Email { get; private set; }
        public string Password { get; private set; }

        public Account(string name, string surname, string
email, string password)
        {
            Name = name;
            Surname = surname;
            Email = email;
            Password = password;
        }

        public void ChangePassword(string newPassword)
        {
            Password = newPassword;
        }

        public bool IsPasswordValid(string password)
        {
            return password.Length >= 8;
        }

        public string GetAccountInfo()
        {
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Tickets.Models
{
    public class CityStop
    {
        public string CityName { get; private set; }
        public DateTime ArrivalTime { get; private set; }
        public DateTime DepartureTime { get; private set; }

        public CityStop(string cityName, DateTime
arrivalTime, DateTime departureTime)
        {
            CityName = cityName;
            ArrivalTime = arrivalTime;
            DepartureTime = departureTime;
        }

        private string CalculateTimeDifference(DateTime
time1, DateTime time2)
```

```
        return $"Name: {Name}, Surname: {Surname},
Email: {Email}";
    }

    public bool CheckPassword(string inputPassword)
    {
        return Password == inputPassword;
    }

    public static bool operator ==(Account a, Account b)
    {
        if (ReferenceEquals(a, b))
        {
            return true;
        }

        if (a is null || b is null)
        {
            return false;
        }

        return a.Name == b.Name && a.Surname ==
b.Surname && a.Email == b.Email && a.Password ==
b.Password;
    }

    public static bool operator !=(Account a, Account b)
    {
        return !(a == b);
    }
}

{
    TimeSpan difference = time2 - time1;
    difference = difference.Duration();

    int hoursDifference = difference.Hours;
    int minutesDifference = difference.Minutes;

    string result = $" {hoursDifference} год.
{minutesDifference} хв.";

    return result;
}

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

namespace Tickets.Models
{
```

```

public class DataBase
{
    public static List<Account> accounts = new
List<Account>();
    public static List<Train> trains = new List<Train>();
    public static List<Carriage> carriages = new
List<Carriage>();
    public static List<Route> routes = new
List<Route>();
    public static List<Ticket> tickets = new
List<Ticket>();
}

using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace Tickets.Models
{
    internal class DB
    {
        MySqlConnection connection = new
MySqlConnection("server=localhost;port=3306;username
=root;password=root;database=tickets");

        public void OpenConnection()
        {
            if(connection.State ==
System.Data.ConnectionState.Closed)
            {
                connection.Open();
            }
        }

        public void CloseConnection()
        {
            if (connection.State ==
System.Data.ConnectionState.Open)
            {
                connection.Close();
            }
        }

        public MySqlConnection GetConnection()
        {
            return connection;
        }

        private string HashPassword(string password)
        {
            using (SHA256 sha256Hash = SHA256.Create())
            {
                byte[] bytes =
sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(pas
sword));

                StringBuilder builder = new StringBuilder();
                for (int i = 0; i < bytes.Length; i++)
                {
                    builder.Append(bytes[i].ToString("x2"));
                }
                return builder.ToString();
            }
        }
    }
}

```

```

}
}

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

namespace Tickets.Models
{
    public class Route
    {
        public string RouteName { get; private set; }
        public Train Train { get; private set; }
        public List<CityStop> Stops { get; private set; }
        public DateTime DepartureDate { get; private set; }
        public Route(string routeName, Train
train, List<CityStop> stops, DateTime departureDate)
        {
            RouteName = routeName;
            Train = train;
            Stops = stops;
            DepartureDate = departureDate;
        }

        public void AddStop(string cityName, DateTime
arrivalTime, DateTime departureTime)
        {
            Stops.Add(new CityStop(cityName, arrivalTime,
departureTime));
        }
    }

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace Tickets.Models
{
    public class Ticket
    {
        public Account LoggedAccount { get; private set; }
        public Route SelectedRoute { get; private set; }
        public Carriage SelectedCarriage { get; private set; }
        public int SelectedCarriageSeat { get; private set; }
        public List<CityStop> SelectedCities { get; private
set; }
        public List<String> Additional { get; private set; }
        public Double Price { get; private set; }

        [JsonConstructor]
        public Ticket(Account loggedAccount, Route
selectedRoute, Carriage selectedCarriage, int
selectedCarriageSeat, List<CityStop> selectedCities,
List<string> additional, double price)
        {
            LoggedAccount = loggedAccount;
            SelectedRoute = selectedRoute;
            SelectedCarriage = selectedCarriage;
            SelectedCarriageSeat = selectedCarriageSeat;
        }
    }
}

```

```

        SelectedCities = selectedCities;
        Additional = additional;
        Price = price;
    }

    public Ticket(Account loggedaccount, Route
selectedroute, List<CityStop> selectedcities, List<string>
additional)
    {
        LoggedAccount = loggedaccount;
        SelectedRoute = selectedroute;
        SelectedCities = selectedcities;
        Additional = additional;
    }

    public void AddPlace(Carriage selectedcarriage, int
selectedcarriageseat)
    {
        SelectedCarriage = selectedcarriage;
        SelectedCarriageSeat = selectedcarriageseat;
    }
    public void SetPrice (double price)
    {
        Price = price;
    }

    public void AddAdditionalInfo(string info)
    {
        Additional.Add(info);
    }

    public void RemoveAdditionalInfo(string info)
    {
        Additional.Remove(info);
    }

    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace Tickets.Models
{
    public class Train
    {
        public string Name { get; private set; }
        public string Model { get; private set; }
        public List<Carriage> Carriages { get; private set; }
        public string Photo { get; private set; }

        public Train(string name, string model,
List<Carriage> carriages, string photo)
        {
            Name = name;
            Model = model;
            Carriages = carriages;
            Photo = photo;
        }

        public void AddCarriage(Carriage carriage)
        {
            Carriages.Add(carriage);

```

```

    }

    public void AddPhoto(string photo)
    {
        Photo = photo;
    }

    }

    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Security.Cryptography;
    using System.Text;
    using System.Text.Json.Serialization;
    using System.Threading.Tasks;

    namespace Tickets.Models
    {
        public class Carriage
        {
            public string Name { get; private set; }
            public string Type { get; private set; }
            public double Price { get; private set; }
            public List<int> Seats { get; private set; }

            [JsonConstructor]
            public Carriage(string name, string type,double price,
List<int> seats)
            {
                Name = name;
                Type = type;
                Price = price;
                Seats = seats;
            }

            public void BuySeat(int seatNumber)
            {
                Seats.Remove(seatNumber);
            }
        }
    }

    using MySql.Data.MySqlClient;
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Security.Cryptography;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;
    using Tickets.Forms;
    using Tickets.Models;
    using Tickets.Serializable;
    using static
System.Windows.Forms.VisualStyles.VisualStyleElement
;
    using yt_DesignUI;

    namespace Tickets
    {
        public partial class SignIn : Form
        {
            public SignIn()

```

```

    {
        InitializeComponent();
        egoldsGoogleTextBox2.UseSystemPasswordChar
        = true;
        LoadBase();
    }

    private void yt_Button2_Click(object sender,
    EventArgs e)
    {
        string EmailUser = egoldsGoogleTextBox1.Text;
        string PasswordUser =
        egoldsGoogleTextBox2.Text;

        try
        {
            DB db = new DB();
            DataTable table = new DataTable();
            MySqlDataAdapter adapter = new
            MySqlDataAdapter();

            MySqlCommand command = new
            MySqlCommand("SELECT * FROM `accounts` WHERE
            `Email` = @uE AND `Password` = @uP",
            db.GetConnection());
            command.Parameters.Add("@uE",
            MySqlDbType.VarChar).Value = EmailUser;
            command.Parameters.Add("@uP",
            MySqlDbType.VarChar).Value =
            HashPassword(PasswordUser);

            adapter.SelectCommand = command;
            adapter.Fill(table);

            if (table.Rows.Count > 0)
            {
                DataRow row = table.Rows[0];
                int index = Convert.ToInt32(row["ID"]);
                string name = row["Name"].ToString();
                string surname = row["Surname"].ToString();
                string email = row["Email"].ToString();
                string password =
                row["Password"].ToString();

                Account loggedInAccount = new
                Account(name, surname, email, password);

                this.Hide();

                if (index == 1)
                {
                    Admin admin = new Admin();
                    admin.Show();
                }
                else
                {
                    Main main = new Main(loggedInAccount);
                    main.Show();
                }
            }
            else
            {
                MessageBox.Show("Невірний логін або
                пароль", "Помилка", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
            }
        }
        catch
    }

```

```

    {
        MessageBox.Show("Помилка бази даних",
        "Помилка", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }

    private string HashPassword(string password)
    {
        using (SHA256 sha256Hash = SHA256.Create())
        {
            byte[] bytes =
            sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(pas
            sword));

            StringBuilder builder = new StringBuilder();
            for (int i = 0; i < bytes.Length; i++)
            {
                builder.Append(bytes[i].ToString("x2"));
            }
            return builder.ToString();
        }
    }

    private void label1_Click_1(object sender, EventArgs
    e)
    {
        this.Hide();
        ForgotPassword forgotPassword = new
        ForgotPassword();
        forgotPassword.Show();
    }

    private void label2_Click(object sender, EventArgs e)
    {
        this.Hide();
        SignUp signUp = new SignUp();
        signUp.Show();
    }

    private void LoadBase()
    {
        SaveToJson load = new SaveToJson();

        DataBase.accounts =
        load.Load<Account>("Accounts.json");
        DataBase.carriages =
        load.Load<Carriage>("Carriages.json");
        DataBase.trains =
        load.Load<Train>("Trains.json");
        DataBase.routes =
        load.Load<Route>("Routes.json");
        DataBase.tickets =
        load.Load<Ticket>("Tickets.json");
    }

    private void Save()
    {
        SaveToJson save = new SaveToJson();

        save.Save("Carriages.json", DataBase.carriages);
        save.Save("Routes.json", DataBase.routes);
        save.Save("Trains.json", DataBase.trains);
    }

    private void SignIn_FormClosing(object sender,
    FormClosingEventArgs e)
    {
        Environment.Exit(0);
    }

```



```

    }
}

using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Tickets;
using Tickets.Models;
using Tickets.Serializable;

namespace yt_DesignUI
{
    public partial class SignUp : Form
    {
        public SignUp()
        {
            InitializeComponent();
        }

        private void label1_Click(object sender, EventArgs e)
        {
            this.Hide();
            SignIn signIn = new SignIn();
            signIn.Show();
        }

        private void yt_Button2_Click(object sender,
            EventArgs e)
        {
            Validation valid = new Validation();
            if (valid.IsString(egoldsGoogleTextBox1.Text)
                && valid.IsString(egoldsGoogleTextBox2.Text)
                &&
                valid.IsValidEmail(egoldsGoogleTextBox3.Text))
            {
                if
                (valid.IsValidPassword(egoldsGoogleTextBox4.Text))
                {
                    DB dB = new DB();
                    MySqlCommand command = new
                    MySqlCommand("INSERT INTO `accounts` (`Name`,
                    `Surname`, `Email`, `Password`) VALUES (@name,
                    @surname, @email, @password);", dB.GetConnection());

                    command.Parameters.Add("@name", MySqlDbType.VarChar).Value = egoldsGoogleTextBox1.Text;
                    command.Parameters.Add("@surname",
                    MySqlDbType.VarChar).Value =
                    egoldsGoogleTextBox2.Text;
                    command.Parameters.Add("@email",
                    MySqlDbType.VarChar).Value =
                    egoldsGoogleTextBox3.Text;
                    command.Parameters.Add("@password",
                    MySqlDbType.VarChar).Value =
                    HashPassword(egoldsGoogleTextBox4.Text);

                    dB.OpenConnection();
                }
            }
        }
    }
}

```

```

        if(command.ExecuteNonQuery() == 1)
        {
            MessageBox.Show("Аккаунт успішно
            створений", "Інформація", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
            this.Hide();
            SignIn signIn = new SignIn();
            signIn.Show();
        }

        dB.CloseConnection();
    }else
    {
        MessageBox.Show("Пароль повинен
        містити літери та символи", "Інформація",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information);
    }
}
else
{
    MessageBox.Show("Неправильно заповнені
    данні. Спробуйте ще раз", "Інформація",
    MessageBoxButtons.OK,
    MessageBoxIcon.Information);
}

private string HashPassword(string password)
{
    using (SHA256 sha256Hash = SHA256.Create())
    {
        byte[] bytes =
        sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(pas
        sword));

        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < bytes.Length; i++)
        {
            builder.Append(bytes[i].ToString("x2"));
        }
        return builder.ToString();
    }
}

private void SignUp_FormClosing(object sender,
    FormClosingEventArgs e)
{
    this.Hide();
    SignIn sign = new SignIn();
    sign.Show();
}

private void Save()
{
    SaveToJson save = new SaveToJson();
    save.Save("Accounts.json", DataBase.accounts);
}

using MySql.Data.MySqlClient;
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;
using Tickets.Models;
using yt_DesignUI;

namespace Tickets.Forms
{
    public partial class ForgotPassword : Form
    {
        public ForgotPassword()
        {
            InitializeComponent();

            private void yt_Button2_Click(object sender,
            EventArgs e)
            {
                string EmailUser = egoldsGoogleTextBox1.Text;

                DB db = new DB();
                DataTable table = new DataTable();
                MySqlDataAdapter adapter = new
                MySqlDataAdapter();

                MySqlCommand command = new
                MySqlCommand("SELECT * FROM `accounts` WHERE
                `Email` = @uE", db.GetConnection());
                command.Parameters.Add("@uE",
                MySqlDbType.VarChar).Value = EmailUser;

                adapter.SelectCommand = command;
                adapter.Fill(table);

                if (table.Rows.Count > 0)
                {
                    MessageBox.Show("Для скидання паролю
                    перейдіть не Вашу пошту і слідуйте інструкціям",
                    "Інформація", MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
                    this.Hide();
                    SignIn signIn = new SignIn();
                    signIn.Show();
                }
                else
                {
                    MessageBox.Show("Користувача не
                    знайдено", "Помилка", MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                }

                private void label1_Click(object sender, EventArgs e)
                {
                    this.Hide();
                    ForgotPassword forgotPassword = new
                    ForgotPassword();
                    forgotPassword.Show();
                }
                private void label2_Click(object sender, EventArgs e)
                {
                    this.Hide();
                    SignIn signIn = new SignIn();
                    signIn.Show();
                }

                private void ForgotPassword_FormClosing(object
                sender, FormClosingEventArgs e)

```

```

{
    this.Hide();
    SignIn signIn = new SignIn();
    signIn.Show();
}
}

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;
using yt_DesignUI.Components;
using Tickets.Forms;
using Tickets.Models;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement
;
using yt_DesignUI;
using System.Drawing.Drawing2D;

namespace Tickets.Forms
{
    public partial class Main : Form
    {
        private Account loggedInAccount;
        private List<Route> routes = new List<Route>();
        private List<CityStop> cityStops = new
        List<CityStop>();
        private int currentIndex;
        public Main(Account account)
        {
            InitializeComponent();
            loggedInAccount = account;
            DoMain();
        }
        private void DoMain()
        {
            egoldsFormStyle1.HeaderColor =
            Color.FromArgb(53, 78, 44);
            egoldsFormStyle1.BackColor =
            Color.FromArgb(53, 78, 44);
            //egoldsFormStyle1.FormStyle =
            EgoldsFormStyle.fStyle.UserStyle;
            SetStyle(this);
            OffPanels();
            routes.Clear();
            cityStops.Clear();
            currentIndex = 0;
            routes.AddRange(DataBase.routes);

            SetcityStops();
            DisplayPanels();
        }

        private void SetcityStops()
        {
            foreach (var route in DataBase.routes)
            {
                cityStops.Add(route.Stops[0]);
            }
        }
    }
}

```

```

        cityStops.Add(route.Stops[route.Stops.Count -
1]);
    }
}

private void SetStyle(Control control)
{
    //label1.BackColor = Color.FromArgb(80, 0xD8,
0xD8, 0xD8);
    button1.BackColor = Color.FromArgb(53, 78, 44);
    button4.BackColor = Color.FromArgb(232, 232,
232);
    button5.BackColor = Color.FromArgb(232, 232,
232);
    button6.BackColor = Color.FromArgb(232, 232,
232);

    foreach (Control ctrl in control.Controls)
    {
        if (ctrl is Label)
        {
            ((Label)ctrl).BackColor =
Color.FromArgb(232, 232, 232);
        }

        if (ctrl is PictureBox)
        {
            ((PictureBox)ctrl).BackColor =
Color.FromArgb(232, 232, 232);
        }

        if (ctrl.HasChildren)
        {
            SetStyle(ctrl);
        }

        if (ctrl is System.Windows.Forms.Button)
        {
            SetButtonsBackColor((System.Windows.Forms.Button)ctr
l);
        }
    }
    pictureBox6.BackColor = SystemColors.Control;
    pictureBox2.BackColor = Color.White;
    pictureBox20.BackColor = Color.FromArgb(53,
78, 44);
    pictureBox21.BackColor = Color.FromArgb(53,
78, 44);

    RoutePictureBox(pictureBox12, 50);
    RoutePictureBox(pictureBox8, 50);
    RoutePictureBox(pictureBox18, 50);
    dateTimePicker1.MinDate = DateTime.Today;
    dateTimePicker1.CustomFormat = " ";
}

private void RoutePictureBox(PictureBox
_pictureBox, int radius)
{
    GraphicsPath path = new GraphicsPath();
    path.AddArc(0, 0, radius, radius, 180, 90);
    path.AddArc(_pictureBox.Width - radius, 0,
radius, radius, 270, 90);
    path.AddArc(_pictureBox.Width - radius,
_pictureBox.Height - radius, radius, radius, 0, 90);
    path.AddArc(0, _pictureBox.Height - radius,
radius, radius, 90, 90);
}

```

```

        path.CloseFigure();
        _pictureBox.Region = new Region(path);
    }
    private void
SetButtonsBackColor(System.Windows.Forms.Button
button)
    {
        button.FlatStyle = FlatStyle.Flat;
        button.FlatAppearance.BorderSize = 0;
    }

    private void pictureBox2_Click(object sender,
EventArgs e)
    {
        string temp = egoldsGoogleTextBox1.Text;
        egoldsGoogleTextBox1.Text =
egoldsGoogleTextBox2.Text;
        egoldsGoogleTextBox2.Text = temp;
    }
    private void PanelVisible(Panel panel, bool status)
    {
        panel.Visible = status;
    }
    private void OffPanels()
    {
        PanelVisible(panel1, false);
        PanelVisible(panel2, false);
        PanelVisible(panel3, false);
    }

    private void DisplayPanels()
    {
        OffPanels();

        for (int i = currentIndex; i <
Math.Min(routes.Count, currentIndex + 3); i++)
        {
            switch (i - currentIndex)
            {
                case 0:
                    Panel1(routes[i], cityStops[i * 2],
cityStops[i * 2 + 1]);
                    break;
                case 1:
                    Panel2(routes[i], cityStops[i * 2],
cityStops[i * 2 + 1]);
                    break;
                case 2:
                    Panel3(routes[i], cityStops[i * 2],
cityStops[i * 2 + 1]);
                    break;
                default:
                    break;
            }
        }

        // Визначення видимості кнопок перемикання
        if (routes.Count > 3)
        {
            button2.Enabled = currentIndex >= 3;
            button3.Enabled = currentIndex + 3 <
routes.Count;
        }
        else
        {
            button2.Enabled = false;
            button3.Enabled = false;
        }
    }
}

```

```

    }
}

private void Panel1(Route route, CityStop city1,
CityStop city2)
{
    PanelVisible(panel1, true);
    label2.Text = route.RouteName;
    label4.Text = city1.CityName;
    label5.Text = city2.CityName;

    label1.Text =
city1.ArrivalTime.ToString("HH:mm");
    label3.Text =
city2.ArrivalTime.ToString("HH:mm");

    label6.Text =
CalculateTimeDifference(city1.ArrivalTime,
city2.ArrivalTime);

    label19.Text =
route.DepartureDate.ToString("dd.MM");
    if (route.Train.Photo != null)
    {
        pictureBox12.Image =
Image.FromFile(route.Train.Photo);
    }
}

private void Panel2(Route route, CityStop city1,
CityStop city2)
{
    PanelVisible(panel2, true);
    label12.Text = route.RouteName;
    label9.Text = city1.CityName;
    label8.Text = city2.CityName;

    label11.Text =
city1.ArrivalTime.ToString("HH:mm");
    label10.Text =
city2.ArrivalTime.ToString("HH:mm");

    label7.Text =
CalculateTimeDifference(city1.ArrivalTime,
city2.ArrivalTime);

    label20.Text =
route.DepartureDate.ToString("dd.MM");
    if (route.Train.Photo != null)
    {
        pictureBox8.Image =
Image.FromFile(route.Train.Photo);
    }
}

private void Panel3(Route route, CityStop city1,
CityStop city2)
{
    PanelVisible(panel3, true);
    label18.Text = route.RouteName;
    label15.Text = city1.CityName;
    label14.Text = city2.CityName;

    label17.Text =
city1.ArrivalTime.ToString("HH:mm");
    label16.Text =
city2.ArrivalTime.ToString("HH:mm");

```

```

    label13.Text =
CalculateTimeDifference(city1.ArrivalTime,
city2.ArrivalTime);

    label21.Text =
route.DepartureDate.ToString("dd.MM");
    if (route.Train.Photo != null)
    {
        pictureBox18.Image =
Image.FromFile(route.Train.Photo);
    }
}

private string CalculateTimeDifference(DateTime
time1, DateTime time2)
{
    TimeSpan difference = time2 - time1;
    difference = difference.Duration();

    int hoursDifference = difference.Hours;
    int minutesDifference = difference.Minutes;

    string result = $"{hoursDifference} год.
{minutesDifference} хв.";

    return result;
}

private void button1_Click(object sender, EventArgs
e)
{
    bool isFromFilled =
string.IsNullOrEmpty(egoldsGoogleTextBox1.Text);
    bool isToFilled =
string.IsNullOrEmpty(egoldsGoogleTextBox2.Text);

    if (isFromFilled || isToFilled)
    {
        MessageBox.Show("Для пошуку заповніть
поля Звідки/Куда", "Інформація",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else if (isFromFilled || isToFilled)
    {
        DoMain();
    }
    else
    {
        Search(egoldsGoogleTextBox1.Text,
egoldsGoogleTextBox2.Text);
    }
}

private void button2_Click(object sender, EventArgs
e)
{
    if (currentIndex >= 3)
    {
        currentIndex -= 3;
        DisplayPanels();
    }
}

private void button3_Click(object sender, EventArgs
e)
{

```

```

        if (currentIndex + 3 < routes.Count)
        {
            currentIndex += 3;
            DisplayPanels();
        }
    }

    private void pictureBox20_Click(object sender,
EventArgs e)
    {
        YourTickets yourTickets = new
YourTickets(loggedInAccount);
        yourTickets.ShowDialog();
    }

    private void pictureBox21_Click(object sender,
EventArgs e)
    {
        Info info = new Info();
        info.ShowDialog();
    }

    private void Search(string city1Name, string
city2Name)
    {
        if (city1Name == "" && city2Name == "")
        {
            DoMain();
        }
        else
        {
            currentIndex = 0;
            routes.Clear();
            cityStops.Clear();

            foreach (Route route in DataBase.routes)
            {
                bool city1Found =
!string.IsNullOrEmpty(city1Name);
                bool city2Found =
!string.IsNullOrEmpty(city2Name);
                bool bothCitiesFound = false; // Флаг, що
позначає, чи були знайдені обидва міста

                // Перевіряємо, чи міста city1Name та
city2Name знаходяться на маршруті
                int city1Index = -1;
                int city2Index = -1;
                foreach (CityStop cityStop in route.Stops)
                {
                    if (city1Found &&
cityStop.CityName.Contains(city1Name))
                    {
                        cityStops.Add(cityStop);
                        city1Index =
route.Stops.IndexOf(cityStop);
                        city1Found = false;
                    }
                    else if (city2Found &&
cityStop.CityName.Contains(city2Name))
                    {
                        cityStops.Add(cityStop);
                        city2Index =
route.Stops.IndexOf(cityStop);
                        city2Found = false;
                    }
                }

                if (city1Index != -1 && city2Index != -1)
                {

```

```

                    bothCitiesFound = true;
                    break;
                }
            }

            if (bothCitiesFound && city1Index <
city2Index)
            {
                routes.Add(route);
            }
        }

        if (dateTimePicker1.CustomFormat != " ")
        {
            DateSearch(dateTimePicker1.Value);
        }

        DisplayPanels();
    }

    private void DateSearch(DateTime _departuredate)
    {
        foreach (var item in routes)
        {
            if (item.DepartureDate.ToString("dd.MM") !=
_departuredade.ToString("dd.MM"))
            {
                DateRemove(routes.IndexOf(item));
                break;
            }
        }
    }

    private void DateRemove(int _indexToRemove)
    {
        routes.RemoveAt(_indexToRemove);
        cityStops.RemoveAt(_indexToRemove);
        cityStops.RemoveAt(_indexToRemove);
        DateSearch(dateTimePicker1.Value);
    }

    private void SelectOlaceSearch(string route, string
city1, string city2)
    {
        List<CityStop> SelectedCities = new
List<CityStop>();

        foreach (CityStop city in cityStops)
        {
            if (city.CityName == city1 || city.CityName ==
city2)
            {
                SelectedCities.Add(city);
            }
        }

        foreach (Route item in routes)
        {
            if (item.RouteName == route)
            {
                Ticket ticket = new Ticket(loggedInAccount,
item, SelectedCities, new List<String>());
                SelectPlace selectPlace = new
SelectPlace(this, ticket);

                selectPlace.Show();
                this.Hide();
                break;
            }
        }
    }

```

```

    }
    }
    private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
    {
        dateTimePicker1.CustomFormat = "dd/MM";
    }

    private void dateTimePicker1_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == Keys.Delete)
        {
            dateTimePicker1.CustomFormat = " ";
        }
    }
    private void button4_Click(object sender, EventArgs e)
    {
        SelectOlaceSearch(label2.Text, label4.Text, label5.Text);
    }

    private void button5_Click(object sender, EventArgs e)
    {
        SelectOlaceSearch(label12.Text, label9.Text, label8.Text);
    }

    private void button6_Click(object sender, EventArgs e)
    {
        SelectOlaceSearch(label18.Text, label15.Text, label14.Text);
    }
    private void Main_FormClosing(object sender, FormClosingEventArgs e)
    {
        this.Hide();
        SignIn sign = new SignIn();
        sign.Show();
    }

}

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;
using Tickets.Forms;

namespace Tickets.Forms
{
    public partial class Admin : Form
    {
        public Admin()
        {
            InitializeComponent();
        }
    }
}

```

```

    private void yt_Button1_Click(object sender, EventArgs e)
    {
        AdminAddCarriage carriage = new AdminAddCarriage();
        carriage.ShowDialog();
    }

    private void yt_Button2_Click(object sender, EventArgs e)
    {
        AdminAddTrain train = new AdminAddTrain();
        train.ShowDialog();
    }

    private void yt_Button3_Click(object sender, EventArgs e)
    {
        AdminAddRoute route = new AdminAddRoute();
        route.ShowDialog();
    }
    private void Admin_FormClosing(object sender, FormClosingEventArgs e)
    {
        this.Hide();
        SignIn sign = new SignIn();
        sign.Show();
    }
}

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;
using Tickets.Models;
using Tickets.Serializable;

namespace Tickets.Forms
{
    public partial class AdminAddCarriage : Form
    {
        public AdminAddCarriage()
        {
            InitializeComponent();
        }

        private void yt_Button2_Click(object sender, EventArgs e)
        {
            Carriage carriage = new Carriage(egoldsGoogleTextBox1.Text, egoldsGoogleTextBox2.Text, Double.Parse(egoldsGoogleTextBox3.Text), Enumerable.Range(1, Int32.Parse(egoldsGoogleTextBox4.Text)).ToList());
            DataBase.carriages.Add(carriage);
            Save();
            this.Close();
        }

        private void Save()
    }
}

```

```

    {
        SaveToJson save = new SaveToJson();
        save.Save("Carriages.json", DataBase.carriages);
    }
}

```

```

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;
using Tickets.Models;
using Tickets.Serializable;

```

```

namespace Tickets.Forms
{
    public partial class AdminAddRoute : Form
    {
        private Route route;
        public AdminAddRoute()
        {
            InitializeComponent();
            SetComboBox();
            dateTimePicker1.MinDate = DateTime.Today;
        }

```

```

        private void yt_Button2_Click(object sender,
            EventArgs e)
        {
            if (route == null)
            {
                Clear();
                route = new
                Route(egoldsGoogleTextBox1.Text,
                DataBase.trains[comboBox1.SelectedIndex], new
                List<CityStop>(), dateTimePicker1.Value);
                yt_Button2.Text = "OK";
                panel1.Visible = false;
                panel2.Visible = true;
            }
            else
            {
                DataBase.routes.Add(route);
                Save();
                this.Close();
            }
        }

```

```

        private void SetComboBox()
        {
            comboBox1.Items.Clear();
            foreach (Train item in DataBase.trains)
            {
                comboBox1.Items.Add(item.Name);
            }
        }

```

```

        private void Save()
        {
            SaveToJson save = new SaveToJson();
            save.Save("Routes.json", DataBase.routes);
        }

```

```

        private void yt_Button1_Click(object sender,
            EventArgs e)
        {
            route.AddStop(egoldsGoogleTextBox2.Text,
            DateTime.Parse(maskedTextBox1.Text),
            DateTime.Parse(maskedTextBox2.Text));

```

```

            egoldsGoogleTextBox2.Text = null;
            maskedTextBox1.Text = null;
            maskedTextBox2.Text = null;
        }

```

```

        private void Clear()
        {
            egoldsGoogleTextBox1.Enabled = false;
            comboBox1.Enabled = false;

```

```

            egoldsGoogleTextBox2.Enabled = true;
            maskedTextBox1.Enabled = true;
            maskedTextBox2.Enabled = true;
            yt_Button1.Enabled = true;
        }

```

```

    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Tickets.Models;
using Tickets.Serializable;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement
;

```

```

namespace Tickets.Forms
{
    public partial class AdminAddTrain : Form
    {
        private Train train;
        Random random = new Random();
        public AdminAddTrain()
        {
            InitializeComponent();
            DoMain();
        }

```

```

        private void DoMain()
        {
            SetComboBox();
            RoutePictureBox(pictureBox12, 50);
        }
        private void RoutePictureBox(PictureBox
        _pictureBox, int radius)
        {
            GraphicsPath path = new GraphicsPath();
            path.AddArc(0, 0, radius, radius, 180, 90);

```

```

        path.AddArc(_pictureBox.Width - radius, 0,
radius, radius, 270, 90);
        path.AddArc(_pictureBox.Width - radius,
_pictureBox.Height - radius, radius, 0, 90);
        path.AddArc(0, _pictureBox.Height - radius,
radius, radius, 90, 90);
        path.CloseFigure();
        _pictureBox.Region = new Region(path);
    }
    private void yt_Button2_Click(object sender,
EventArgs e)
    {
        if (train == null)
        {
            train = new Train(egoldsGoogleTextBox1.Text,
egoldsGoogleTextBox2.Text, new List<Carriage>(), null);
            Clear();
            yt_Button2.Text = "OK";
        } else
        {
            DataBase.trains.Add(train);
            Save();
            this.Close();
        }
    }
    private void yt_Button1_Click(object sender,
EventArgs e)
    {
        train.AddCarriage(DataBase.carriages[comboBox1.Select
edIndex]);
        //yt_Button1.Enabled = false;
    }

    private void
comboBox1_SelectedIndexChanged(object sender,
EventArgs e)
    {
        yt_Button1.Enabled = true;
    }

    private void Clear()
    {
        egoldsGoogleTextBox1.Text = null;
        egoldsGoogleTextBox2.Text = null;
        egoldsGoogleTextBox1.Enabled = false;
        egoldsGoogleTextBox2.Enabled = false;

        comboBox1.Enabled = true;
        yt_Button1.Enabled = true;
    }

    private void SetComboBox()
    {
        comboBox1.Items.Clear();
        foreach (Carriage item in DataBase.carriages)
        {
            comboBox1.Items.Add(item.Name);
        }
    }

    private void Save()
    {
        SaveToJson save = new SaveToJson();
        save.Save("Trains.json", DataBase.trains);
    }

```

```

    private void pictureBox12_Click(object sender,
EventArgs e)
    {
        if (train != null)
        {
            using (OpenFileDialog openFileDialog = new
OpenFileDialog())
            {
                openFileDialog.Filter = "Image files (*.jpg,
*.jpeg, *.png)|*.jpg;*.jpeg;*.png|All files (*.*)|*.*";
                openFileDialog.RestoreDirectory = true;

                if (openFileDialog.ShowDialog() ==
DialogResult.OK)
                {
                    string selectedFilePath =
openFileDialog.FileName;
                    string fileExtension =
Path.GetExtension(selectedFilePath);
                    string folderPath =
Path.Combine(Application.StartupPath, "TrainPhotos");
                    string destinationPath =
Path.Combine(folderPath, train.Name +
$"_{random.Next(10000, 99999)}" + fileExtension);

                    File.Copy(selectedFilePath,
destinationPath, true);

                    pictureBox12.Image =
Image.FromFile(destinationPath);
                    train.AddPhoto(destinationPath);
                }
            }
        }
        else
        {
            MessageBox.Show("Спочатку створіть новий
потяг", "Інформація", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }
    }

}

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;
using Tickets.Models;
using Tickets.Serializable;
using yt_DesignUI;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement
;

namespace Tickets.Forms
{
    public partial class EnterPassengerData : Form
    {
        private Form PreviousForm;
        private Ticket Ticket;
    }
}

```



```

public EnterPassengerData(Form previousform,
Ticket ticket)
{
    InitializeComponent();
    PreviousForm = previousform;
    Ticket = ticket;

    DoMain();
}
private void DoMain()
{
    egoldsFormStyle1.HeaderColor =
Color.FromArgb(53, 78, 44);
    egoldsFormStyle1.BackColor =
Color.FromArgb(53, 78, 44);
    SetBackColors(this);
    SetTextBoxes(Ticket.LoggedAccount);
    SetPrice();
}

private void SetBackColors(Control control)
{
    foreach (Control ctrl in control.Controls)
    {
        if (ctrl is Label)
        {
            ((Label)ctrl).BackColor =
Color.FromArgb(232, 232, 232);
        }

        if (ctrl is PictureBox)
        {
            ((PictureBox)ctrl).BackColor =
Color.FromArgb(232, 232, 232);
        }

        if (ctrl.HasChildren)
        {
            SetBackColors(ctrl);
        }
        if (ctrl is EgoldsToggleSwitch)
        {
            ((EgoldsToggleSwitch)ctrl).BackColorON =
Color.FromArgb(53, 78, 44);
            //((EgoldsToggleSwitch)ctrl).BackColor =
Color.FromArgb(0xC2, 0xD8, 0xBA);
        }
        pictureBox4.BackColor =
SystemColors.Control;
        pictureBox20.BackColor = Color.FromArgb(53,
78, 44);
        pictureBox21.BackColor = Color.FromArgb(53,
78, 44);
        pictureBox12.BackColor = Color.FromArgb(53,
78, 44);
    }
}

private void SetTextBoxes(Account account)
{
    egoldsGoogleTextBox1.Text = account.Name;
    egoldsGoogleTextBox2.Text = account.Surname;
    egoldsGoogleTextBox3.Text = account.Email;
}
private Carriage SearchCariage(Route route)

```

```

{
    foreach (Carriage item in route.Train.Carriages)
    {
        if (item.Type == Ticket.SelectedCarriage.Type)
        {
            return item;
        }
    }
    return null;
}
private void SetPrice()
{
    label7.Text =
$"{SearchCariage(Ticket.SelectedRoute).Price}₾";
}

private void PlusPrice(double Price)
{
    label7.Text =
$"{Double.Parse(label7.Text.Substring(0,
label7.Text.Length - 1)) + Price}₾";
}

private void
egoldsToggleSwitch1_CheckedChanged(object sender)
{
    UpdatePrice(70, egoldsToggleSwitch1.Checked,
"Постільна білизна");
}

private void
egoldsToggleSwitch2_CheckedChanged(object sender)
{
    UpdatePrice(30, egoldsToggleSwitch2.Checked,
"Авторський чай");
}

private void
egoldsToggleSwitch3_CheckedChanged(object sender)
{
    UpdatePrice(30, egoldsToggleSwitch3.Checked,
"Чайний збір");
}

private void
egoldsToggleSwitch5_CheckedChanged(object sender)
{
    UpdatePrice(70, egoldsToggleSwitch5.Checked,
"Дріп кава");
}

private void
egoldsToggleSwitch6_CheckedChanged(object sender)
{
    UpdatePrice(40, egoldsToggleSwitch6.Checked,
"Фірмове печиво");
}

private void UpdatePrice(int priceChange, bool
isChecked, string name)
{
    if (isChecked)
    {
        PlusPrice(priceChange);
        Ticket.AddAdditionalInfo(name);
    }
    else
    {

```

```

        PlusPrice(-priceChange);
        Ticket.RemoveAdditionalInfo(name);
    }

    }

    private void pictureBox3_Click(object sender,
EventArgs e)
    {
        this.Close();
        PreviousForm.Show();
    }

    private void pictureBox4_Click(object sender,
EventArgs e)
    {
        this.Close();
        PreviousForm.Show();
    }

    private void button4_Click(object sender, EventArgs
e)
    {
        SearchCariage(Ticket.SelectedRoute).BuySeat(Ticket.Sele
ctedCarriageSeat);

        Ticket.SetPrice(Double.Parse(label7.Text.Substring(0,
label7.Text.Length - 1)));
        DataBase.tickets.Add(Ticket);
        Save();

        this.Hide();

        Main main = new Main(Ticket.LoggedAccount);
        main.Show();
    }

    private void Save()
    {
        SaveToJson save = new SaveToJson();
        save.Save("Routes.json", DataBase.routes);
        save.Save("Tickets.json", DataBase.tickets);
    }

    private void pictureBox20_Click(object sender,
EventArgs e)
    {
        YourTickets yourTickets = new
YourTickets(Ticket.LoggedAccount);
        yourTickets.ShowDialog();
    }

    private void pictureBox21_Click(object sender,
EventArgs e)
    {
        Info info = new Info();
        info.ShowDialog();
    }

    private void maskedTextBox1_TextChanged(object
sender, EventArgs e)
    {
        if (maskedTextBox1.Text.Length == 19 &&
maskedTextBox2.Text.Length == 2 &&
maskedTextBox3.Text.Length == 2 &&
maskedTextBox4.Text.Length == 3)
        {
            button4.Enabled = true;
        }
    }

```

```

        if (maskedTextBox1.Text.Length >= 19)
        {
            maskedTextBox2.Focus();
        }

        if (maskedTextBox2.Text.Length >= 2)
        {
            maskedTextBox3.Focus();
        }

        if (maskedTextBox3.Text.Length >= 2)
        {
            maskedTextBox4.Focus();
        }

    }

    private void
EnterPassengerData_FormClosing(object sender,
FormClosingEventArgs e)
    {
        if (e.CloseReason == CloseReason.UserClosing)
        {
            PreviousForm.Show();
        }
    }

}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;

namespace Tickets.Forms
{
    public partial class SelectPlace : Form
    {
        private Form PreviousForm;
        private Ticket Ticket;

        public SelectPlace(Form previousform, Ticket ticket)
        {
            InitializeComponent();
            PreviousForm = previousform;
            Ticket = ticket;

            DoMain();
            SetBackColors(this);
        }

        private void DoMain()
        {
            egoldsFormStyle1.HeaderColor =
Color.FromArgb(53, 78, 44);
            egoldsFormStyle1.BackColor =
Color.FromArgb(53, 78, 44);
            Panel1(Ticket.SelectedRoute,
Ticket.SelectedCities[0], Ticket.SelectedCities[1]);

            SetComboBox1();
        }

        private void SetBackColors(Control control)

```

```

{
    if (ctrl.HasChildren)
    {
        SetBackColors(ctrl);
    }

    if (ctrl is System.Windows.Forms.Button)
    {
        SetButtonsBackColor((System.Windows.Forms.Button)ctrl);
    }
    pictureBox3.BackColor = SystemColors.Control;
    pictureBox6.BackColor = SystemColors.Control;
    button4.BackColor = Color.FromArgb(232, 232, 232);
    pictureBox20.BackColor = Color.FromArgb(53, 78, 44);
    pictureBox21.BackColor = Color.FromArgb(53, 78, 44);
    pictureBox9.BackColor = Color.FromArgb(53, 78, 44);

    RoutePictureBox(pictureBox12, 50);
}

private void RoutePictureBox(PictureBox _pictureBox, int radius)
{
    GraphicsPath path = new GraphicsPath();
    path.AddArc(0, 0, radius, radius, 180, 90);
    path.AddArc(_pictureBox.Width - radius, 0, radius, radius, 270, 90);
    path.AddArc(_pictureBox.Width - radius, _pictureBox.Height - radius, radius, radius, 0, 90);
    path.AddArc(0, _pictureBox.Height - radius, radius, radius, 90, 90);
    path.CloseFigure();
    _pictureBox.Region = new Region(path);
}

private void SetButtonsBackColor(System.Windows.Forms.Button button)
{
    button.FlatStyle = FlatStyle.Flat;
    button.FlatAppearance.BorderSize = 0;
}

private void PanelVisible(Panel panel, bool status)
{
    panel.Visible = status;
}

private void button4_Click(object sender, EventArgs e)
{
    Ticket.AddPlace(SearchCariage(Ticket.SelectedRoute), Convert.ToInt32(comboBox2.SelectedItem));
    EnterPassengerData enter = new EnterPassengerData(this, Ticket);

    enter.Show();
    this.Hide();
}

```

```

private Carriage SearchCariage(Route route)
{
    foreach (Carriage item in route.Train.Carriages)
    {
        if (item.Type == comboBox1.SelectedItem)
        {
            return item;
        }
    }
    return null;
}

private void Panel1(Route route, CityStop city1, CityStop city2)
{
    PanelVisible(panel1, true);
    label2.Text = route.RouteName;
    label4.Text = city1.CityName;
    label5.Text = city2.CityName;

    label1.Text = city1.ArrivalTime.ToString("HH:mm");
    label3.Text = city2.ArrivalTime.ToString("HH:mm");

    label8.Text = route.DepartureDate.ToString("dd.MM");

    if (route.Train.Photo != null)
    {
        pictureBox12.Image = Image.FromFile(route.Train.Photo);
    }

    private string CalculateTimeDifference(DateTime time1, DateTime time2)
    {
        TimeSpan difference = time2 - time1;
        difference = difference.Duration();

        int hoursDifference = difference.Hours;
        int minutesDifference = difference.Minutes;

        string result = $"{hoursDifference} год. {minutesDifference} хв.";

        return result;
    }

    private void SetPrice(Carriage carriage)
    {
        label7.Text = $"{carriage.Price} ₴";
    }

    private void SetComboBox1()
    {
        comboBox1.Items.Clear();
        foreach (Carriage item in Ticket.SelectedRoute.Train.Carriages)
        {
            comboBox1.Items.Add(item.Type);
        }
    }

    private void SetComboBox2(Carriage carriage)
    {
        comboBox2.Items.Clear();
        comboBox2.Enabled = true;
    }
}

```

```

        if (carriage != null)
        {
            foreach (int item in carriage.Seats)
            {
                comboBox2.Items.Add(item);
            }
        }

        private void
comboBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            foreach (var item in
Ticket.SelectedRoute.Train.Carriages)
            {
                if (item.Type == comboBox1.SelectedItem)
                {
                    SetComboBox2(item);
                    SetPrice(item);
                }
            }
        }

        private void
comboBox2_SelectedIndexChanged(object sender,
EventArgs e)
        {
            button4.Enabled = true;
        }

        private void pictureBox20_Click(object sender,
EventArgs e)

```

```

        {
            YourTickets yourTickets = new
YourTickets(Ticket.LoggedAccount);
            yourTickets.ShowDialog();
        }

        private void pictureBox21_Click(object sender,
EventArgs e)
        {
            Info info = new Info();
            info.ShowDialog();
        }

        private void pictureBox2_Click(object sender,
EventArgs e)
        {
            this.Close();
            PreviousForm.Show();
        }

        private void pictureBox3_Click(object sender,
EventArgs e)
        {
            this.Close();
            PreviousForm.Show();
        }

        private void SelectPlace_FormClosing(object sender,
FormClosingEventArgs e)
        {
            if (e.CloseReason == CloseReason.UserClosing)
            {
                PreviousForm.Show();
            }
        }

```

Додаток Б
Діаграма Use Case

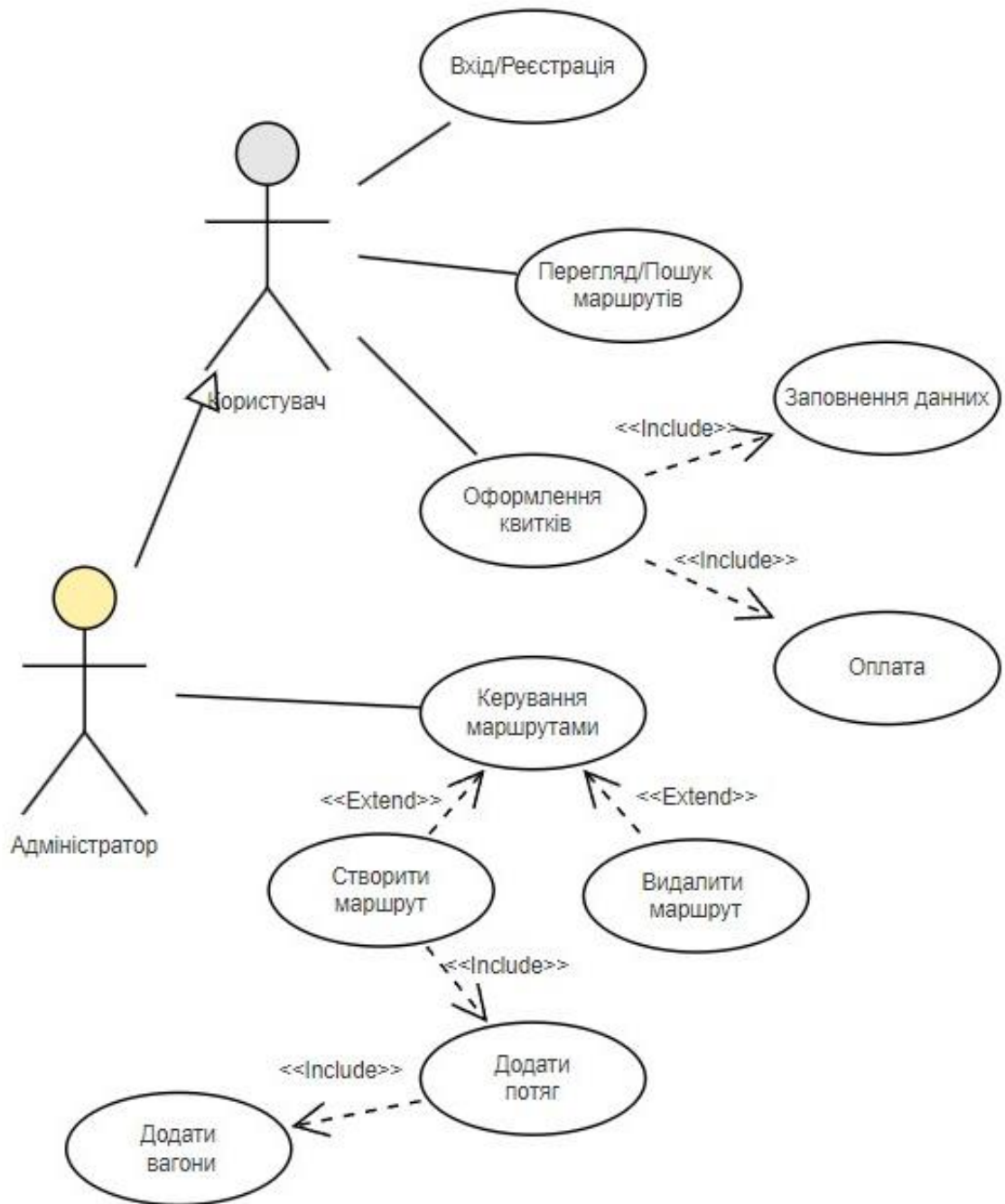


Рисунок Б.1 – Діаграма Use Case

Додаток В

Діаграма State Chart

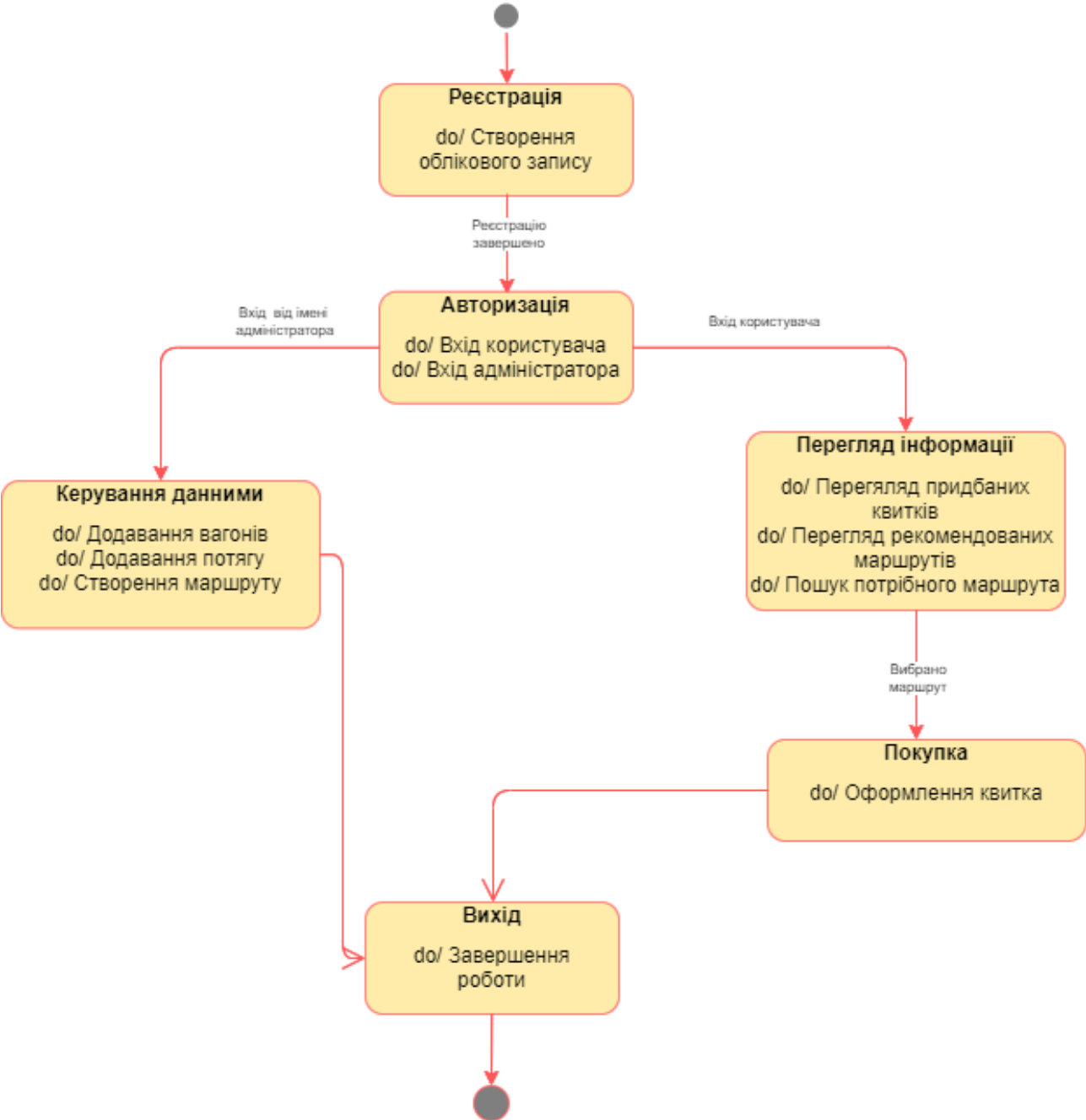


Рисунок В.1 – Діаграма State Chart

Додаток Г

Діаграма Activity

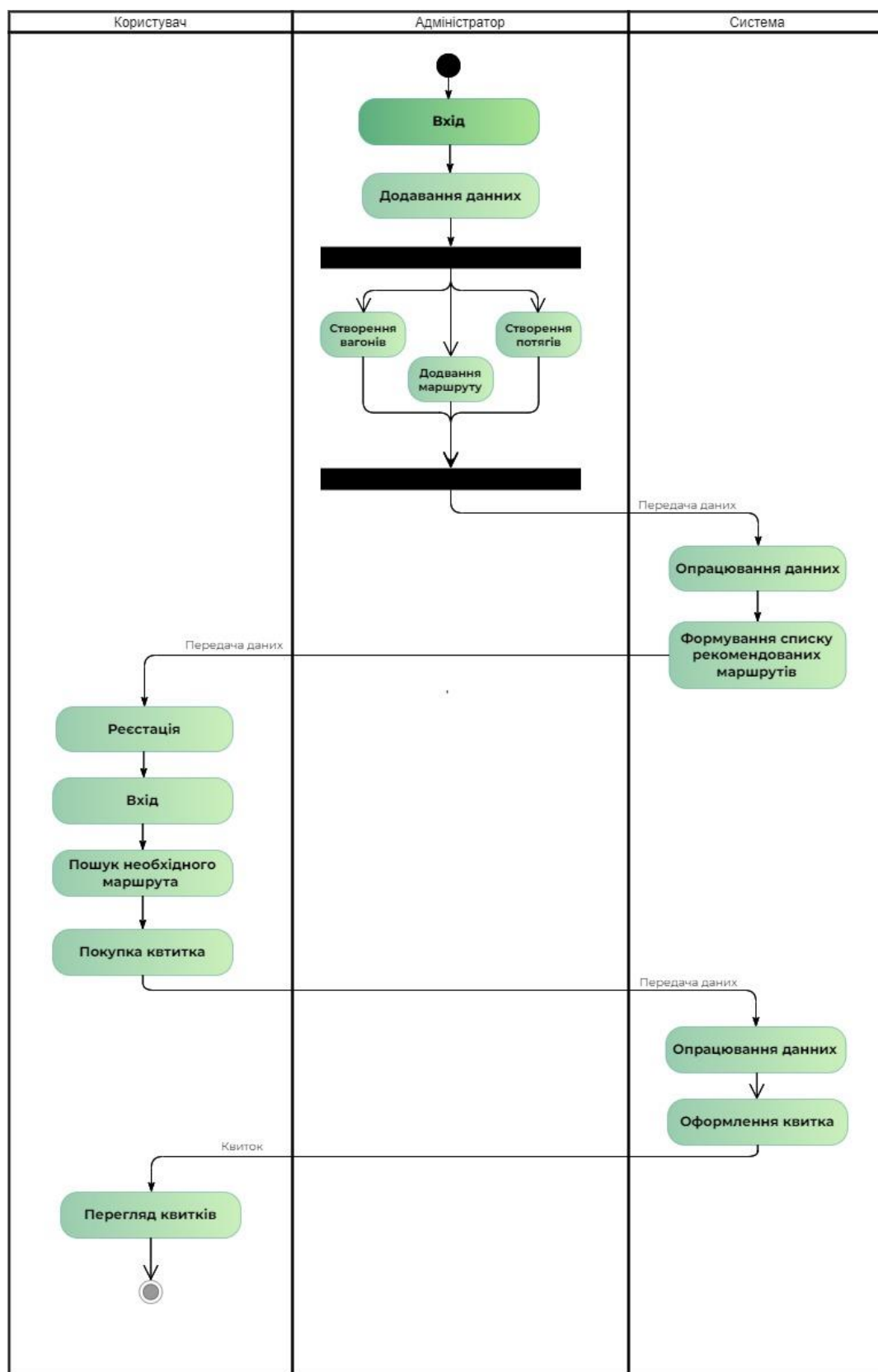


Рисунок Г.1 – Діаграма Activity

Додаток Д

Діаграма Sequence

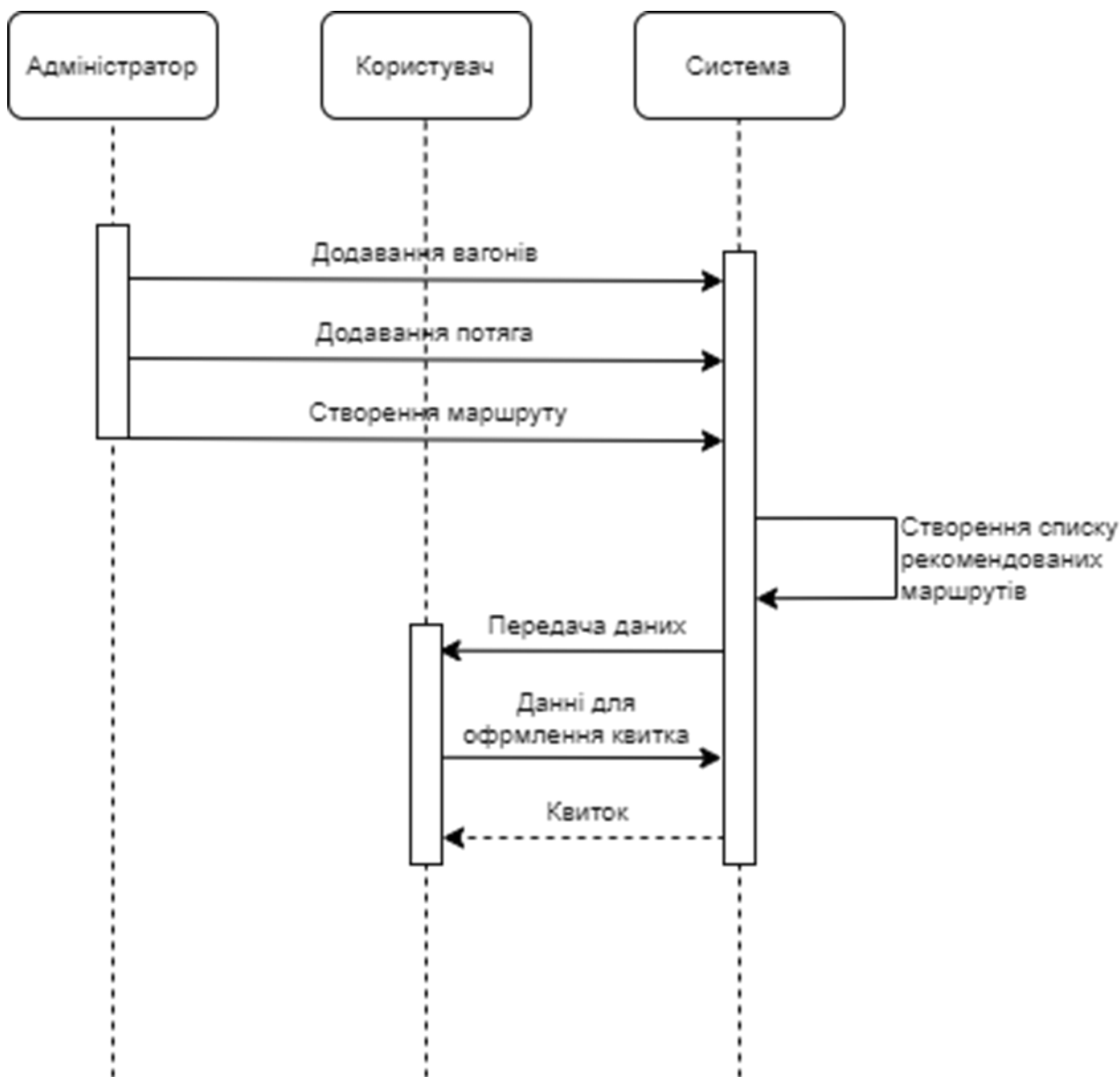


Рисунок Д.1 – Діаграма Sequence

Додаток Е Діаграма Collaboration

4. Створення списку
рекомендованих маршрутів

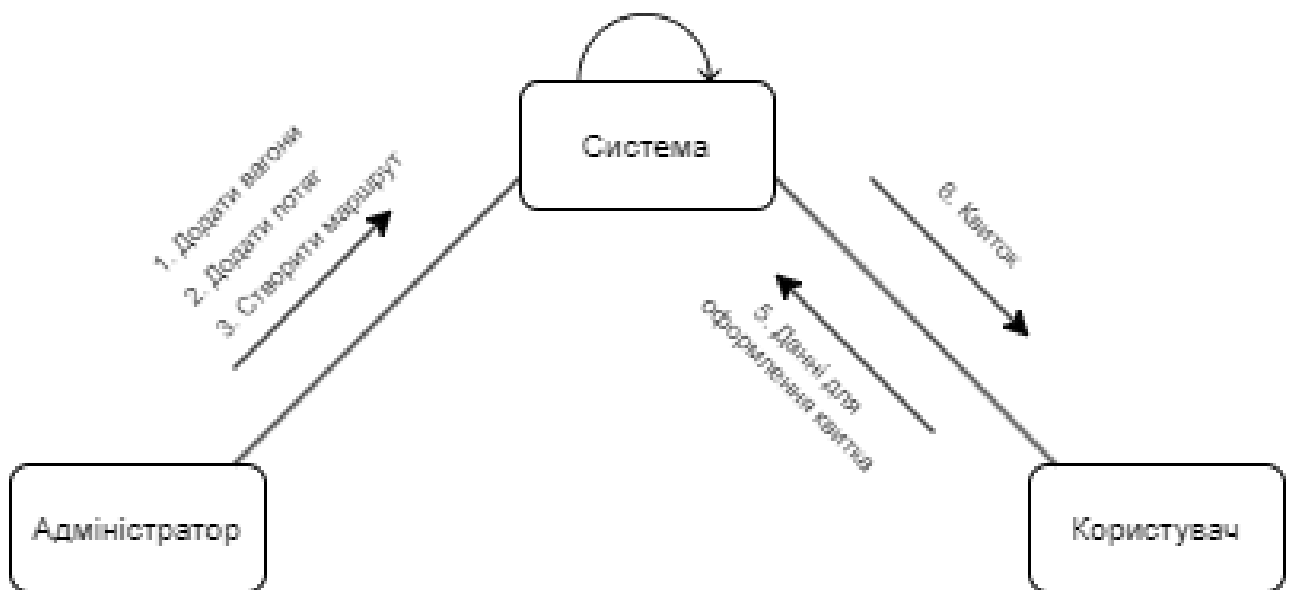


Рисунок Е.1 – Діаграма Collaboration

Додаток Ж

Діаграма Class

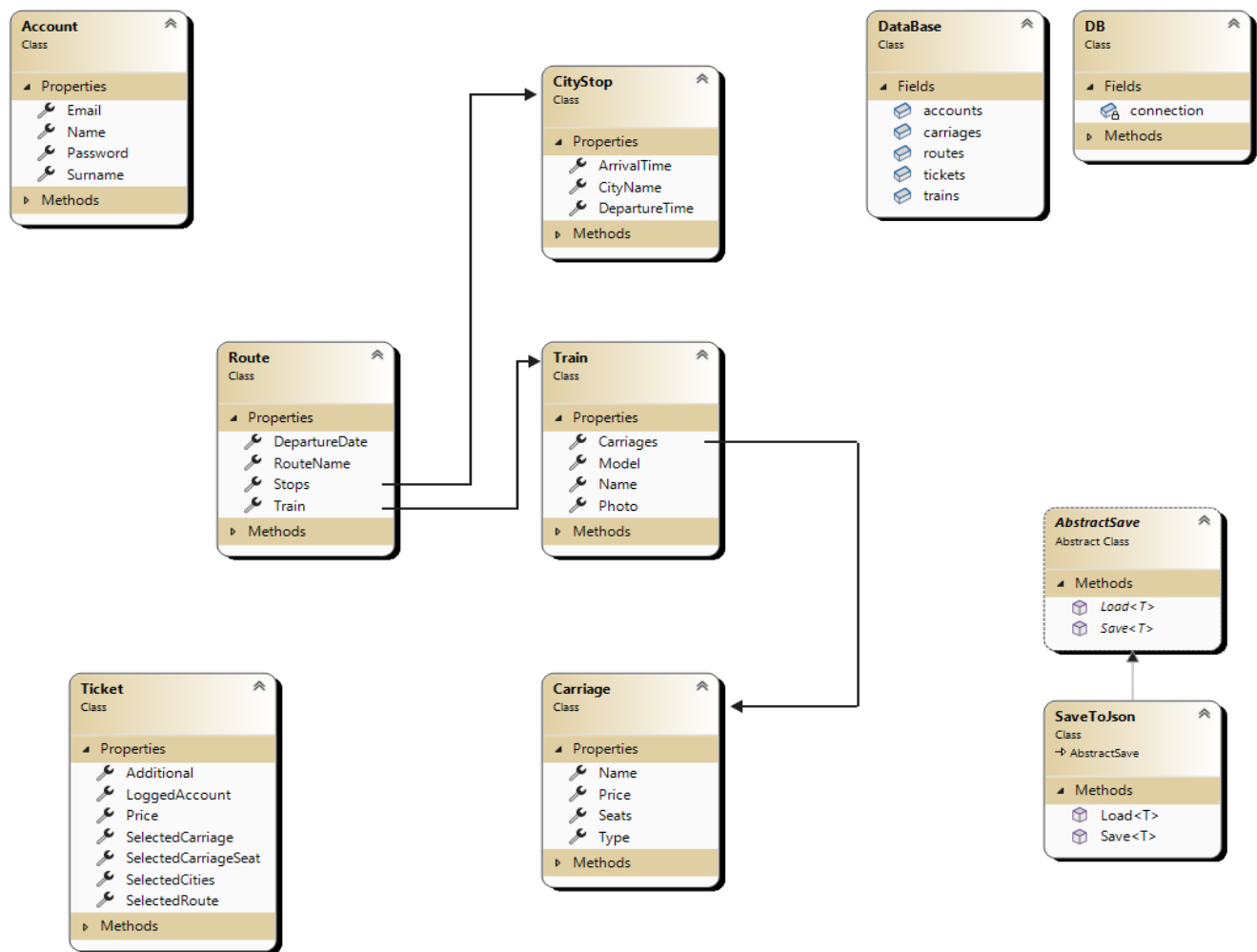


Рисунок Д.1 – Діаграма Class