МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

**КУРСОВА РОБОТА**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Бази даних»

на тему: **«База даних каси автовокзалу»**

студента ІІ курсу групи ВТ-22-1 спеціальності 121 «Інженерія програмного забезпечення»

Середи Івана Романовича

(прізвище, ім’я та по-батькові)

Керівник: ст. викладач кафедри ІПЗ Чижмотря.О.В.

Дата захисту: "\_" \_\_\_\_\_\_\_2024р.

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ольга КОРОТУН 1

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Світлана КРАВЧЕНКО1

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Олексій ЧИЖМОТРЯ 1

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Інна СУГОНЯК 1

(підпис) (прізвище та ініціали)

Житомир – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень: бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри ІПЗ \_\_\_\_\_\_\_\_\_\_\_\_ Т.А. Вакалюк “\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_2023 р.

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ СТУДЕНТУ

Середі Івану Романовичу

1. Тема роботи: **База даних каси автовокзалу**,

керівник роботи: ст. викладач кафедри ІПЗ Чижмотря О.В.

1. Строк подання студентом: “28” березень 2024р.
2. Вихідні дані до роботи: Розробити базу даних для сайту автовокзалу
3. Зміст розрахунково-пояснювальної записки(перелік питань. Які підлягають розробці)
   * Аналіз інформаційних потоків та особливостей предметної області дослідження
   * Проектування бази даних за напрямком курсової роботи
   * Реалізація підсистеми обробки даних за напрямком курсової роботи
   * Адміністрування баз даних

1. Перелік графічного матеріалу(з точним зазначенням обов’язкових креслень) 1. Презентація до КР:

2. Посилання на репозиторій: <https://gitlab.com/sereda_ivan/kursovadb_sereda_ivan_vt-22-1>

1. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання  прийняв |
| 1,2,3 | Чижмотря О.В., к.т.н., доц. | 15.10.2023 | 15.10.2023 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання “ 15 ” жовтня 2023 р.

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсового проекту | Строк виконання етапів проекту | Примітки |
| 1 | Постановка задачі | 10.11.2023 | Виконано |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 20.11.2023 | Виконано |
| 3 | Формулювання технічного завдання | 25.11.2023 | Виконано |
| 4 | Опрацювання літературних джерел | 26.11.2023 | Виконано |
| 5 | Проектування бази даних | 05.01.2024 | Виконано |
| 6 | Написання програмного коду | 20.01.2024 | Виконано |
| 7 | Відлагодження | 20.02.2024 | Виконано |
| 8 | Написання пояснювальної записки | 21.03.2023 | Виконано |
| 9 | Захист |  |  |

**КАЛЕНДАРНИЙ ПЛАН**

**Студент** \_\_ \_\_\_\_\_\_\_\_ \_\_ Іван СЕРЕДА

(підпис) (прізвище та ініціали)

**Керівник роботи**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Олексій ЧИЖМОТРЯ

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Завданням на курсову роботу була розробка веб-додатку бази данних до сайту автовокзалу.

Пояснювальна записка до курсового роботи на тему “ База даних каси автовокзалу ” складається з вступу, чотирьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на 65 сторінках друкованого тексту.

Пояснювальна записка має 88 сторінок додатків. Список використаних джерел містить 10 найменувань і займає 1 сторінку. В роботі наведено 45 рисунків. Загальний обсяг роботи – 153 сторінки.

У першому розділі було обгрунтовано створення сайту на тему “База данних автовокзалу”.

У другому розділі проведено проектування бази данних сайту.

У третьому розділі проведено тестування програмного продукту.

Висновок містить в собі результати виконаної роботи при створені сайту на тему “База данних сайту автовокзалу”.

Ключові слова: SSMS, T-SQL, PYTHON, FLASK, JINJA2, HTML, CSS, BOOTSTRAP, JAVA SCRIPT, ООП, КЛІЄНТ, КОРИСТУВАЧ, АДМІН, АВТОРИЗАЦІЯ, РОЛІ, ДАНІ, БАЗА ДАНИХ, ЗАПИТ.

ЗМІСТ

[ВСТУП 7](#_Toc162542115)

[РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА 9](#_Toc162542116)

[ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ 9](#_Toc162542117)

[1.1 Аналіз інформаційних потреб та визначення предметної області дослідження 9](#_Toc162542118)

[1.2 Архітектура та засоби реалізації бази даних 14](#_Toc162542119)

[Висновки до першого розділу: 18](#_Toc162542120)

[РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ САЙТУ АВТОВОКЗАЛУ 20](#_Toc162542121)

[2.1 Аналіз структури інформаційних процесів адміністраторів 20](#_Toc162542122)

[2.2 Проектування бази даних сайту оголошень 21](#_Toc162542123)

[Висновки до другого розділу 27](#_Toc162542124)

[РОЗДІЛ 3. РЕАЛІЗАЦЯ САЙТУ АВТОВОКЗАЛУ 28](#_Toc162542125)

[3.1. Проектування інтерфейсу системи сайту автовокзалу 28](#_Toc162542126)

[3.2. Реалізації операцій обробки даних в БД 44](#_Toc162542127)

[Висновок до третього розділу 54](#_Toc162542128)

[РОЗДІЛ 4. АДМІНІСТРУВАННЯ БАЗ ДАНИХ 55](#_Toc162542129)

[4.1. Розробка заходів захисту інформації в БД 55](#_Toc162542130)

[4.2. Налаштування параметрів роботи Sql Server. 61](#_Toc162542131)

[Список використаних джерел 65](#_Toc162542132)

[ДОДАТКИ 66](#_Toc162542133)

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ:

ТЗ – Технічне завдання

СУБД – система управління базою даних

CRUD – create, read, update, delete.

ПЗ – Програмне забезпечення

Рис – рисунок

СУБД – система управління базою даних

БД - база данних

AP – адмін панель

ООП – об’єкто орієнтоване програмування

JS – мова програмування Java Script

МП – мова програмування

BT - Bootstrap інструмент для використання шаблонів стилів сторінок та скриптів.

# ВСТУП

**Актуальність теми.** Сучасний світ стрімко розвивається, і разом з ним зростає потреба в зручних та економних рішеннях для повсякденних завдань. Одним із таких рішень є веб-додатки, які полегшують життя користувачам у багатьох сферах, включаючи бронювання квитків на транспорт.

Веб-додаток для купівлі квитків на автобус – це зручний інструмент, який дозволяє:

* Економити час: Не потрібно йти на автовокзал або стояти в черзі, щоб купити квиток.
* Планувати подорож заздалегідь: Можна купити квиток на будь-який рейс, на будь-яку дату, з будь-якого місця.
* Обирати місце в автобусі: Більшість веб-додатків дозволяють заздалегідь вибрати місце, яке вам подобається.
* Оплачувати квиток онлайн: Це безпечний і зручний спосіб оплати, який не потребує готівки.

Веб-додаток для купівлі квитків на автобус – це універсальне рішення, яке підходить людям будь-якого віку. Молодь цінує його за зручність та економию часу, а люди похилого віку – за можливість не виходити з дому, щоб купити квиток.

Я вважаю, що веб-додатки для купівлі квитків на автобус – це майбутнє транспортної системи. Їх зручність, економність та доступність роблять їх все більш популярними, і я впевнений, що в найближчому майбутньому вони стануть основним способом бронювання квитків на автобуси.

**Власна думка:**

Я особисто користуюся веб-додатками для купівлі квитків на автобус вже протягом декількох років. І я можу з упевненістю сказати, що це значно полегшило мені життя. Я більше не витрачаю час на черги, можу заздалегідь спланувати подорож і вибрати місце, яке мені подобається.

Я рекомендую всім, хто ще не користувався веб-додатками для купівлі квитків на автобус, спробувати їх. Ви будете приємно здивовані зручністю та економністю цього способу бронювання квитків.

**Мета розробки.** Метою розробки данного курсового проекту є створення сайту, написаного на таких мовах програмування як Python на фреймворку Flask та JS з використанням об’єктно орієнтованого програмування**,** створення бази данних для збереження інформації про подорожі, квитки та користувачів.

Завданням курсової роботи є розробка веб-додатку та бази данних автовокзалу.

Веб-додаток створюється з метою комфорту замовлення та купівлі квитків на автобуси.

**Об’єктом дослідження** є технологія розробки класів на МП Python які є схожими на таблиці БД на базі T-SQL та СУБД Microsoft SQL Server Management Studio, запис інформації в ці об’єкти та подальше їх використання за правилами ООП. А також використання МП JS для певних дій на самому сайті таких як показ маршруту на Google Map.

# РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА

# ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ

## 1.1 Аналіз інформаційних потреб та визначення предметної області дослідження

В даному курсовому проекті за предметну область взято базу даних, яка буде зберігати усю необхідну інформацію для сайту автовокзалу. Цей сайт надає можливість купівлі квитків онлайн, перегляду певного маршруту поїздки, авторизації користувача та закріплення за ним куплених квитків для подальшого їх перегляду. Також, на сайті є можливість купівлі квитків неавторизованим клієнтам.

Клієнти матимуть можливість купівлі квитків які відповідають певним параметрам, та подальшого їх перегляду. Для збереження історії покупок клієнт може зареєструватись, але й для користувачів є можливість покупки квитків без реєстрації. Для адміністраторів, потрібно зробити можливіть видалення, додавання, оновлення данних. Для цього потрібно створити базу данних в якій буде зберігатись інформація, а для операцій з цими данними розробити адмінпанель на сайті.

База даних має бути зручною для користувачів. Це означає, що інтерфейс користувача (UI) має бути простим, інтуїтивно зрозумілим та навігабельним. Користувачі повинні мати можливість легко знаходити потрібну їм інформацію, а також додавати, видаляти та редагувати дані без зайвих зусиль.

**Швидкість** є однією з найважливіших характеристик бази даних. Користувачі очікують, що запити будуть оброблятися швидко, а дані – видаватися без затримок. Швидкість роботи бази даних залежить від декількох факторів, таких як апаратне забезпечення, програмне забезпечення та структура бази даних.

**Надійність.** База даних повинна бути надійною, щоб дані не втрачалися і не пошкоджувалися. Це означає, що база даних має бути захищена від збоїв живлення, апаратних помилок та несанкціонованого доступу.

**Масштабованість.** База даних повинна бути масштабованою, щоб вона могла рости разом з вашими потребами. Це означає, що база даних має бути здатною обробляти більше даних та користувачів без значного зниження продуктивності.

**Безпека.**База даних повинна бути захищеною від несанкціонованого доступу, використання, розкриття, зміни або знищення. Це означає, що база даних має мати відповідні механізми аутентифікації, авторизації та шифрування.

**Сумісність.**База даних повинна бути сумісною з іншими системами та програмним забезпеченням. Це означає, що база даних має бути здатною обмінюватися даними з іншими системами без проблем.

**Оновлення.** Дані в базі даних повиннi постійно оновлюватися, щоб залишатися актуальними. Це означає, що база даних має мати механізми для імпорту та експорту даних, а також для синхронізації даних з іншими системами.

**Операції з даними.**Для коректного виконання запитів на отримання даних, дані повинні вноситись, видалятись та змінюватись у БД. За це відповідають такі операції:

**Реєстрація користувача**

Додаток повинен розрізняти користувачів та адміністраторів, щоб надати перелік певних функцій.

Перелік дій які доступні користувачу :

* Перегляд інформації
* Купівля квитка
* Перегляд маршруту
* Перегляд куплений квитків

Перелік дій які доступні адміністратору :

* Перегляд інформації
* Купівля квитка
* Перегляд маршруту
* Перегляд куплений квитків
* Доступ до адмінпанелі
* Редагування данних
* Видалення данних

Для цього потрібно створити таблицю з усіма користувачами як сайту так бази данних.

**Вхід користувача**

Авторизація користувачів після реєстрації.

**Роль адміністратор**

Надає можливість користуватись фунціями описаними вище.

**Видалення та додавання маршрутів, автобусів, міст**

Необхідно для того щоб додавати та видаляти інформацію про нові подорожі

**Редагування новини**

Необхідно для зміни інформації про маршрути, автобуси, користувачів і тд. У випадку коли інформація може бути некоректною чи потрібно її замінити на нову.

**Перегляд подорожей**

Щоб купити квиток користувач матиме можливість переглянути вигідних превізників а також маршрутів безпосередньо на картах

**Фільтрація**

Щоб знайти певний маршрут у користувача є можливість відфільтрувати поданий список та віднайти потрібний маршрут.

**Системні повідомлення**

Щоб проінформувати користувача про помилку чи про вдалі події.

**Пошук за номером квитку**

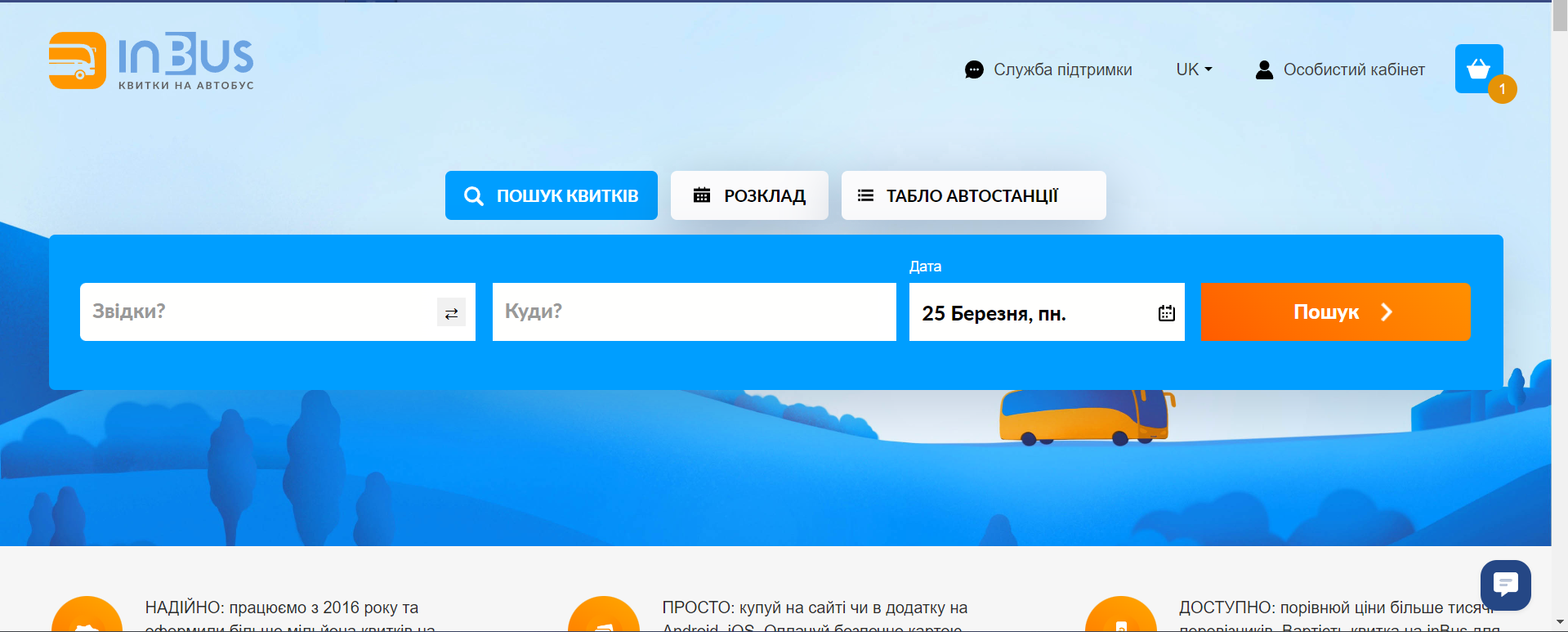
Для комфортності користувача він матиме змогу ввести код квитка та швидко його переглянути.

**Конкуренти:**

Найближчі конкуренти схожі на додаток який розробляється це inBus та LikeBus.

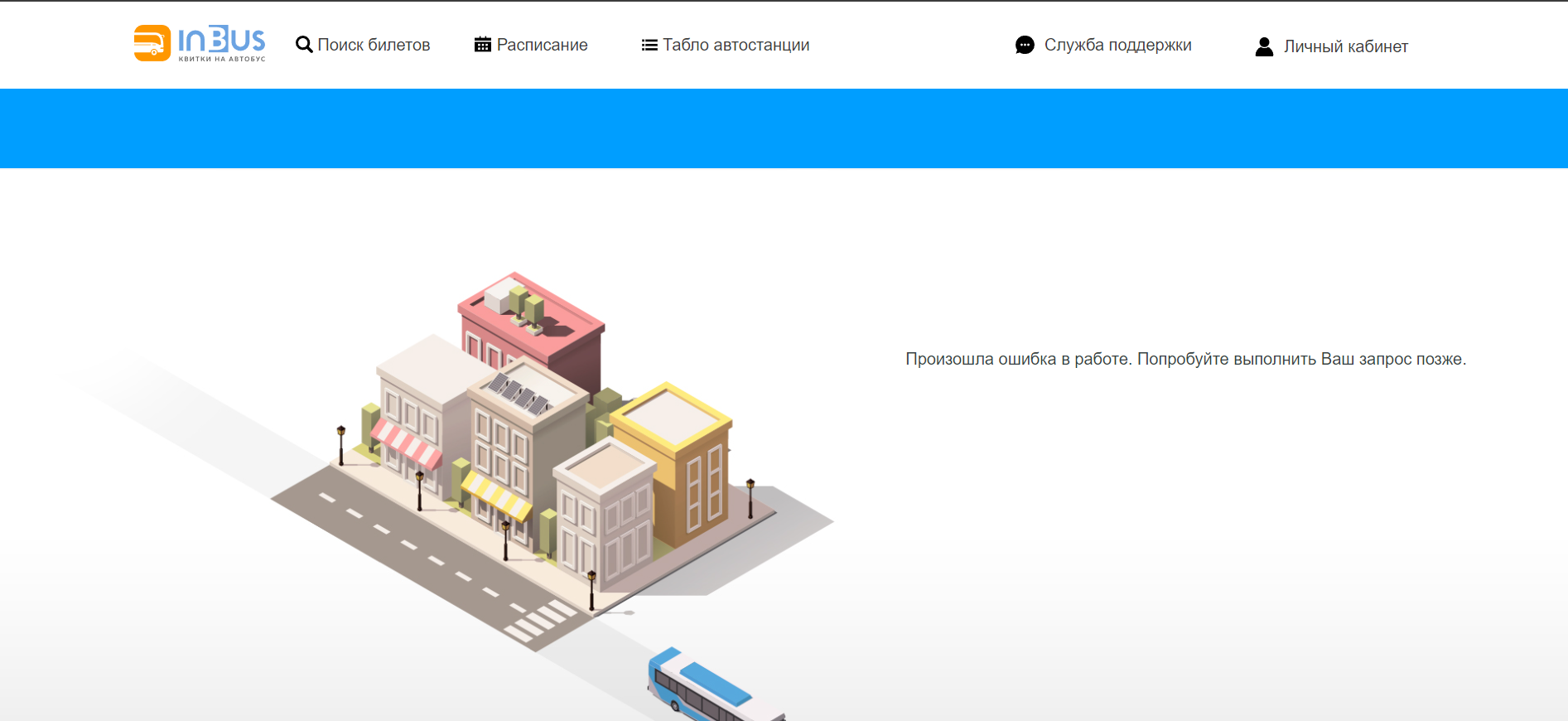
**inBus (**[**https://inbus.ua**](https://inbus.ua)**)**

Великий веб-додаток для пошуку маршрутів автобусів працюючий як в межах України так і надає інформацію про перевезення з України в країни Європи. Сайт має досить привабливий сайт та зручнинй зрозумілий інтерфейс та живий пошук по напрямках.Мають можливість звернутись на інформаційну лінію.

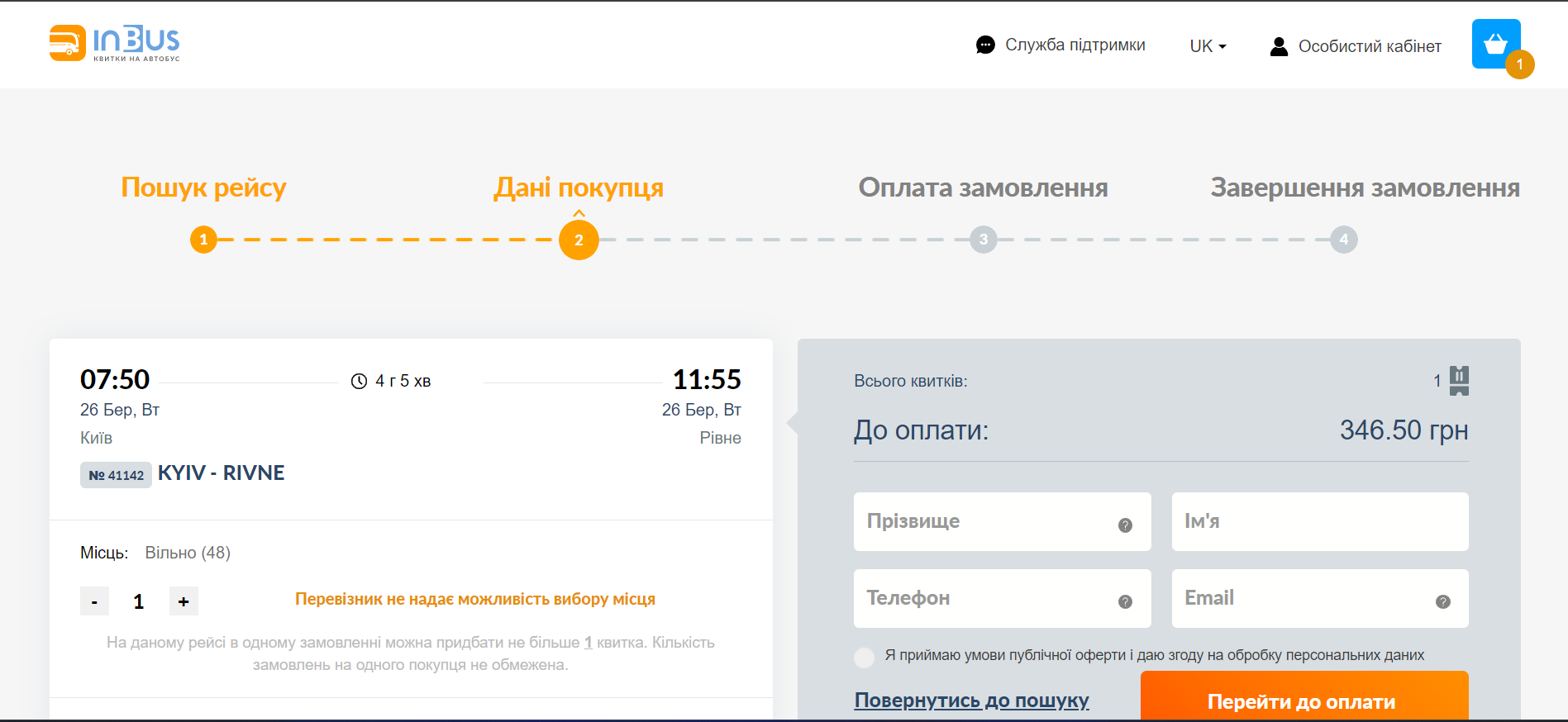


**Рис.1.1.** Вигляд головної сторінки на сайті inBus

З мінусів можу відмітити баги при виборі міста та не можливість вибрати період годин відправлення, а також досить довгий пошук за маршрутами і в залагом довге завантаження інших сторінок.За декілька спроб оформлення квитка були помилки такого типу. Деякі перевізники не надають можливіть вибору місця. Немає фільтрації за обширнішими параметрами.



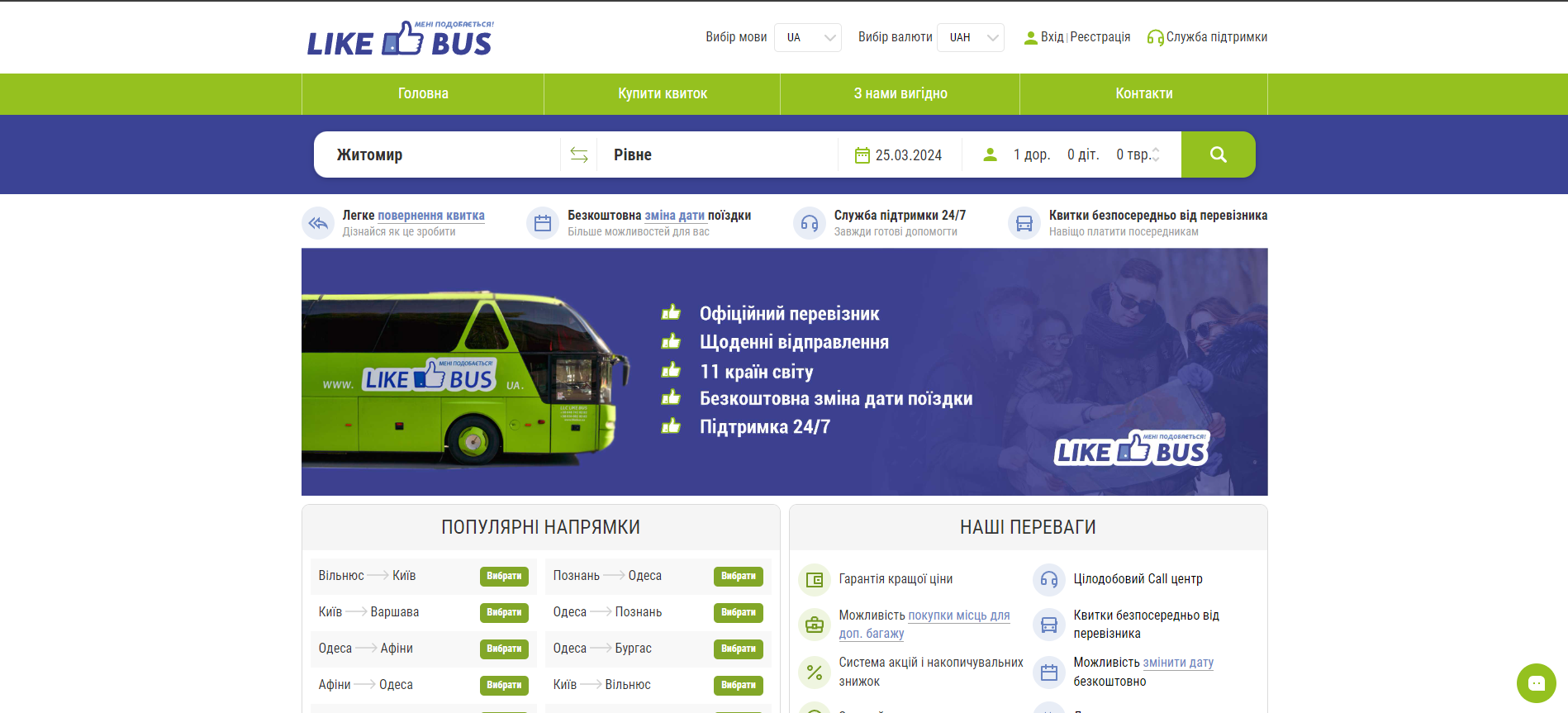
**Рис.1.2.**Помилка при оформленні квитка



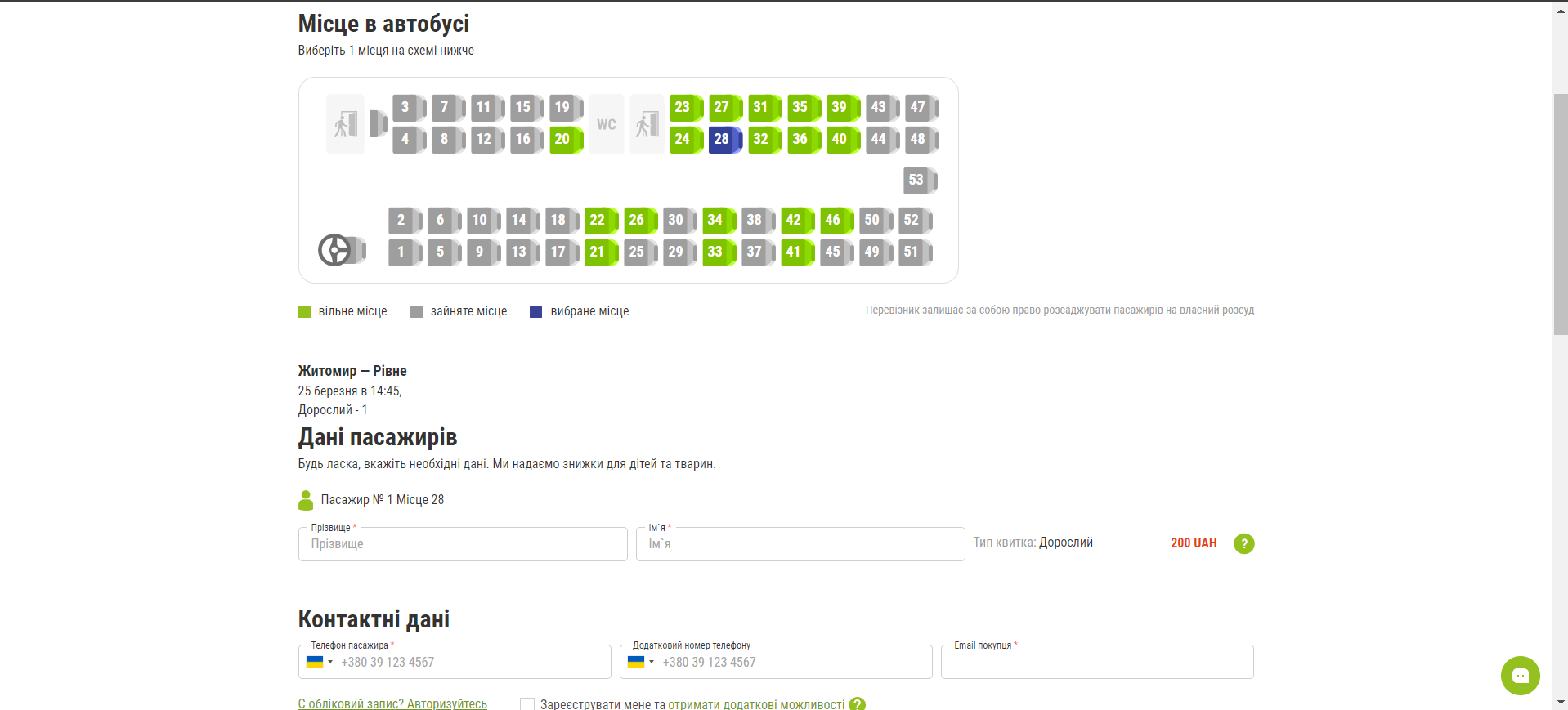
**Рис.1.3.** Оформлення купівлі квитка в inBus

**LikeBus (**[**https://likebus.ua/)**](https://likebus.ua/))

Сайт має презентабельний вид та чудову швидкість роботи, приємний для використання інтерфейс. А також можливіть звязатись в чаті за додатковою інформацією, вибір додаткового багажу.



**Рис.1.4.** Головна сторінка LikeBus



**Рис.1.5.** Оформлення квитка на LikeBus

З мвнусів можу помітити нагромадженість елементами інтерфейсу в яких можна загубитись. Немає фільтрації за обширнішими параметрами.

## 1.2 Архітектура та засоби реалізації бази даних

Для проекту, який очікує велику кількість користувачів, важливо ретельно підібрати СУБД (систему управління базами даних), щоб забезпечити максимально ефективний доступ до даних. Цей документ описує порівняльний аналіз трьох популярних СУБД: Microsoft SQL Server, Oracle та Firebase.

**Таблиця 1.1.**

**Функціональне порівняння СУБД**

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристика** | **Microsoft SQL**  **Server** | **Oracle** | **Firebase** |
| Робота під керівництвом ОС | Задовільно | Добре | Відмінно |
| Складність | Відмінно | Задовільно | Добре |
| Доступ до даних | Відмінно | Добре | Добре |
| Використання у великих проектах | Відмінно | Відмінно | Добре |
| Використання у малих проектах | Добре | Добре | Відмінно |
| Схема даних | Відмінно | Відмінно | Добре |
| Підтримувані мови  програмування | Відмінно | Добре | Задовільно |
| Підключення до  Web | Відмінно | Відмінно | Відмінно |
| Графічні | Добре | Добре | Добре |
| Тригери | Відмінно | Відсутній критерій | Добре |
| Паралелізм | Відмінно | Відмінно | Відмінно |
| Одночасний доступ декількох користувачів | Добре | Добре | Відмінно |
| Обробка даних мультимедіа | Слабо | Відмінно | Відмінно |
| Пошук за регулярними  виразами | Відмінно | Відмінно | Задовільно |
| Підтримка об’єктно-  орієнтованої парадигми | Добре | Відмінно | Задовільно |
| Засоби аналізу | Відмінно | Відмінно | Відмінно |

**Пояснення таблиці:**

Microsoft SQL Server та Oracle це реляційні БД, а Firebase використовує сховище документів

1. Робота під керівництвом ОС:

* Microsoft SQL Server: Windows, Linux.
* Oracle: Windows, Linux, Solaris, HP-UX, OS X.
* Firebase: Не потребує розміщення.

2. Складність:

* Microsoft SQL Server: Легко освоїти, але складні задачі потребують певного досвіду.
* Oracle: Складний для вивчення та потребує знання SQL.
* Firebase: Простий у використанні, але має обмежені можливості.

3. Доступ до даних:

* Microsoft SQL Server: Гнучкий контроль доступу на основі ролей.
* Oracle: Повністю відкритий доступ до даних для користувачів.
* Firebase: Контроль доступу на основі правил Firebase.

4. Використання у великих проектах:

* Microsoft SQL Server: Масштабований, висока продуктивність.
* Oracle: Ідеально підходить для складних систем з високими навантаженнями.
* Firebase: Не рекомендується для проектів з великим об'ємом даних.

5. Використання у малих проектах:

* Microsoft SQL Server: Складний для налаштування.
* Oracle: Зайве громіздкий для малих проектів.
* Firebase: Простий у налаштуванні та підключенні.

6. Схема даних:

* Microsoft SQL Server: Фіксована схема, реляційна модель.
* Oracle: Фіксована та безсхемна схема, реляційна модель.
* Firebase: Не використовує схеми, модель NoSQL.

7. Підтримувані мови програмування:

* Microsoft SQL Server: Широкий спектр мов, C#, Java, Python etc.
* Oracle: C, C#, Go, Java, JavaScript, Python.
* Firebase: C, JS, Objective-C.

8. Графічні інструменти:

* Microsoft SQL Server: SSMS (SQL Server Management Studio).
* Oracle: SQL Developer Data Modeler.
* Firebase: Firebase Console.

9. Тригери:

* Microsoft SQL Server: Повністю підтримуються.
* Oracle: Немає підтримки.
* Firebase: Автоматичне відстеження змін.

10. Паралелізм:

* Microsoft SQL Server: Висока ступінь паралелізму.
* Oracle: Масштабований паралелізм.
* Firebase: Повністю підтримується.

11. Одночасний доступ декількох користувачів:

* Microsoft SQL Server: Масштабований, залежить від ліцензії.
* Oracle: Висока пропускна здатність.
* Firebase: Без обмежень, залежить від тарифу.

12. Обробка даних мультимедіа:

* Microsoft SQL Server: Складне налаштування.
* Oracle: Вбудована підтримка.
* Firebase: Просте зберігання та обробка.

13. Пошук за регулярними виразами:

* Microsoft SQL Server: Повністю підтримується.
* Oracle: Повністю підтримується.
* Firebase: Обмежена підтримка.

14. Засоби аналізу:

* Microsoft SQL Server: SSRS (SQL Server Reporting Services).
* Oracle: Oracle Business Intelligence Suite.
* Firebase: Firebase Analytics (вбудовані).

Висновок:

* Microsoft SQL Server: Ідеально підходить для великих проектів, де потрібна висока масштабованість, доступність, безпека та аналітика.
* Oracle: Рекомендується для складних систем з високими навантаженнями, де потрібна максимальна продуктивність та обробка даних мультимедіа.
* Firebase: Найкращий вибір для швидкої розробки додатків, особливо для малих та середніх проектів. Firebase відрізняється простотою налаштування та використання, але має обмеження щодо масштабованості складних запитів та підтримки об'єктно-орієнтованої парадигми.

## Висновки до першого розділу:

**Конкурентоспроможність**

На ринку проект буде вигравати в конкурентів більшими можливостями за рахунок особливостей яких немає в конкурентів.

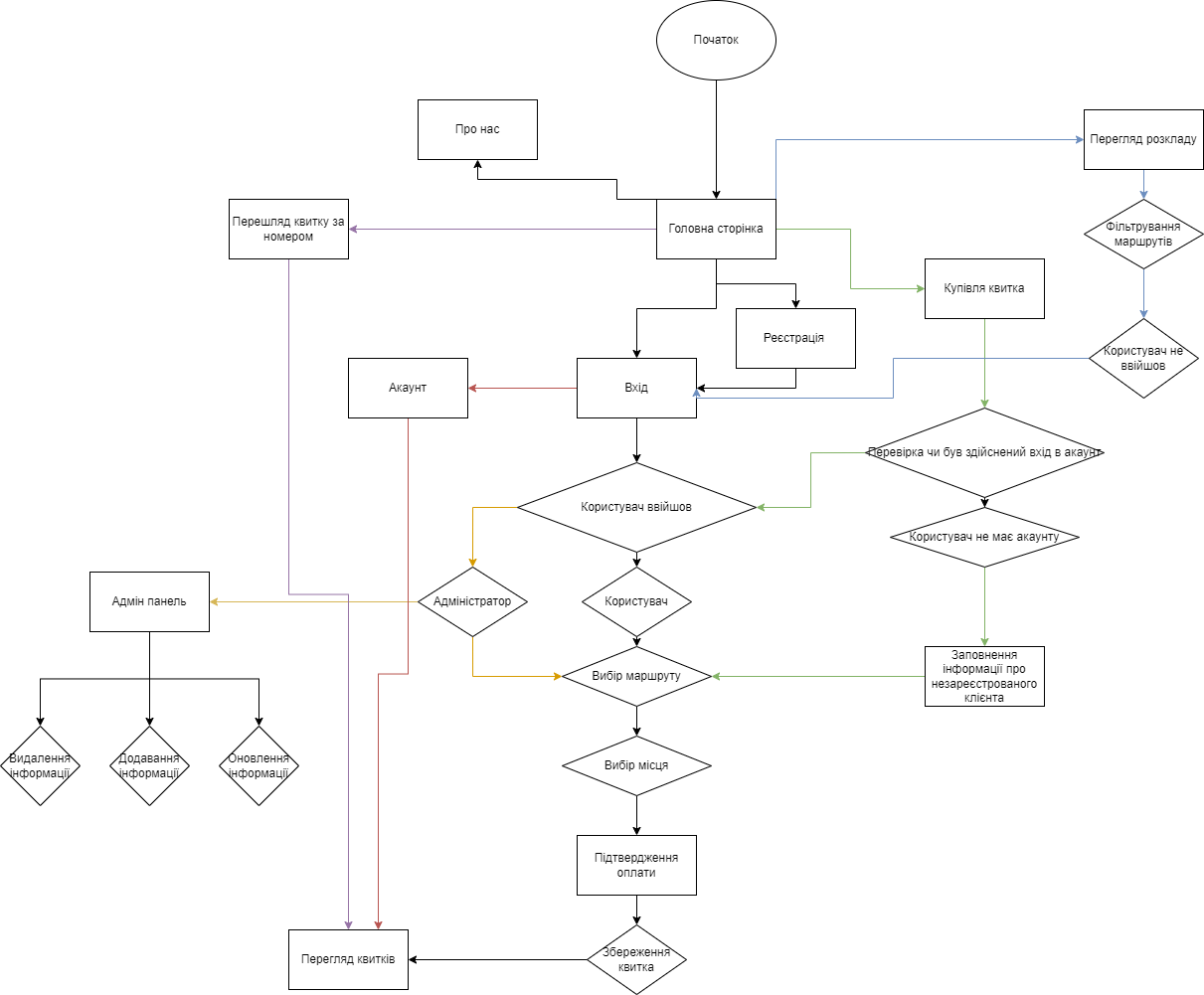
**Архітектура**

Виходячи з потреб конструювання та комфорту як програмування так використання для розробки найбільше підходить БД Microsoft SQL Server та СУБД Server SQL Menegment Studio. Вона безпечна, маштабуєма,високо продуктивна, не працює з медіа, дозволяє надавати ролі для доступу до данних що в рамках веб-додатку автовокзалу практично ідеально підходить.

Мова програмування вибрана Python так як це доволі сильна, проста, зрозуміла та швидка МП. Для ядра серверу вибрано фреймворк Flask так як він є простим в реалізації але й разом з цим доволі потужним.

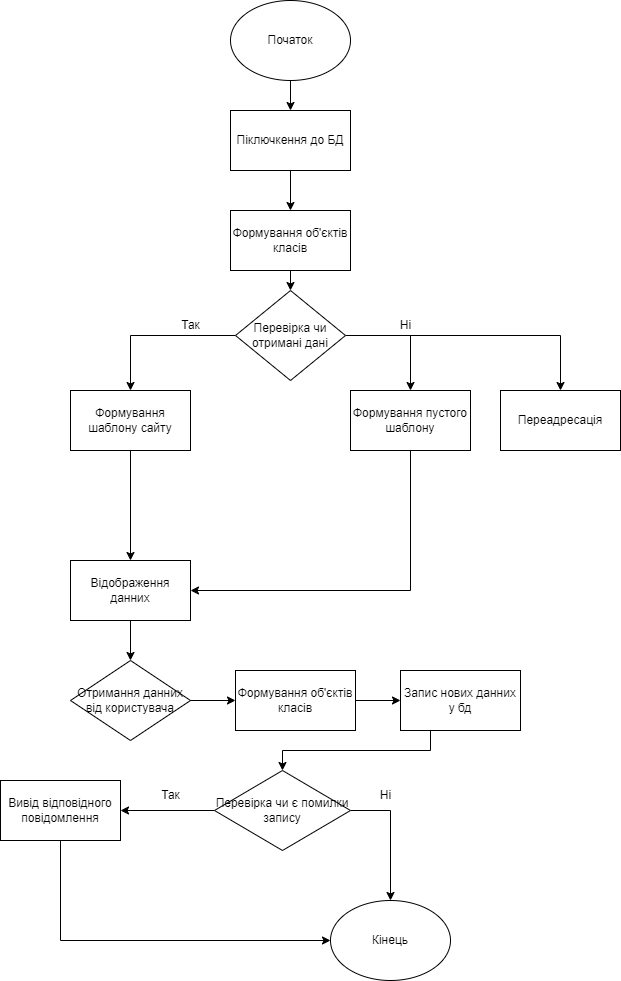
# РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ САЙТУ АВТОВОКЗАЛУ

## 2.1 Аналіз структури інформаційних процесів адміністраторів



**Рис.2.1.** Структура сайту автовокзалу

Користувачі будуть мати змогу працювати з веб-додатком як після аутентифікації, так і без неї. Наприклад переглядати подорожі та їх напрямки, перевірку квитків.

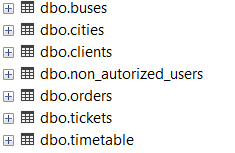


**Рис.2.2.** Структурна схема життєвого циклу запиту

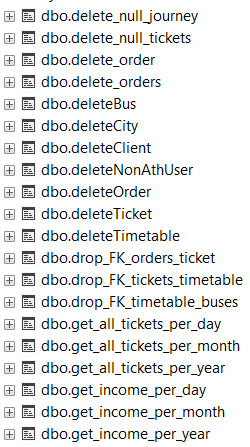
## 2.2 Проектування бази даних сайту оголошень

Після формування функціоналу додатку було створено базу даних. Для проектування було використано SSMS.

Список спроектованих таблиць

****

**Рис.2.3.** Список створених таблиць

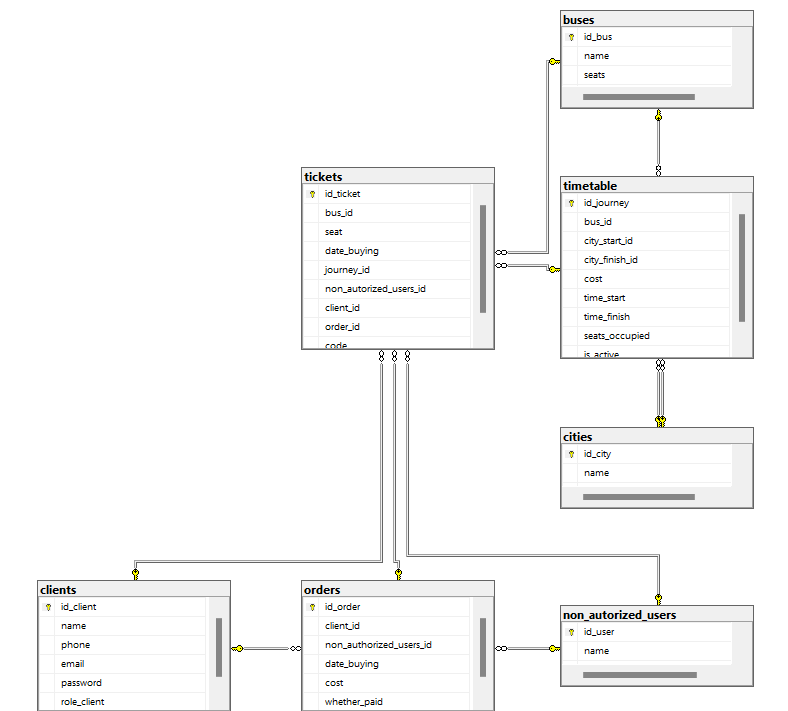


**Рис.2.4.** Список створених процедур

Внаслідок проектування до бази даних включено таблиці:

* buses
* cities
* clients
* non\_autorized\_users
* orders
* tickets
* timetable

Ще була створена діаграма для комфорту представлення зв’язків



**Рис.2.5.** Логічна ERD діаграма

Для збереження інформації про автобус була створена таблиця “buses”. В данну таблицю вводяться всі необхідні дані про автобус такі як назва, кількість місць, номер.

Структура таблиці наведена нижче:

**Таблиця 2.1.** Структура таблиці "buses"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| id\_bus | int | + | - | Id автобуса |
| name | nvarchar | - | - | Назва автобуса |
| seats | int | - | - | Кількість місць |
| bus\_number | nvarchar | - | - | Номер автобуса |

Для збереження інформації про міста була створена таблиця “cities”. В данну таблицю вводяться всі необхідні дані про місто такі як назва та країна.

Структура таблиці наведена нижче:

**Таблиця 2.2.** Структура таблиці "cities"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва поля** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| id\_city | int | + | - | Id міста |
| name | nvarchar | - | - | Назва |
| country | nvarchar | - | - | Країна |

Для збереження інформації про клієнтів була створена таблиця “clients”. В данну таблицю вводяться всі необхідні дані такі як ім’я, номер телефону, email, пароль та роль клієнта яка визначає користувач це чи адмін.

Структура таблиці наведена нижче:

**Таблиця 2.3.** Структура таблиці "clients"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва поля** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| id\_client | int | + | - | Id клієнта |
| name | nvarchar | - | - | Ім'я |
| phone | nvarchar | - | - | Номер телефону |
| email | nvarchar | - | - | Ел.адреса |
| password | nvarchar | - | - | Пароль |
| role\_client | nvarchar | - | - | Роль клієнта |

Для збереження інформації про не зареєстрованих клієнтів була створена таблиця “non\_autorized\_clients”. В данну таблицю вводяться всі необхідні дані такі як ім’я, номер телефону.

Структура таблиці наведена нижче:

**Таблиця 2.4.** Структура таблиці " non\_autorized\_clients"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва поля** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| id\_user | int | + | - | Id користувача |
| name | nvarchar | - | - | Ім'я |
| phone | nvarchar | - | - | Номер телефону |

Для збереження інформації про поїздки була створена таблиця “timetable”. В данну таблицю вводяться всі необхідні дані такі як id автобусу, id мвста відправлення, id міста прибуття, вартість квитка, час відправки та прибуття, кількість зайнятих місць та чи активна дана подорож.

Структура таблиці наведена нижче:

**Таблиця 2.5.** Структура таблиці " timetable "

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва поля** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| id\_journey | int | + | - | Id маршруту |
| bus\_id | int | - | - | Id автобуса |
| city\_start\_id | int | - | + | Id міста відправлення |
| city\_finish\_id | int | - | + | Id міста прибуття |
| cost | float | - | - | Вартість |
| time\_start | datetime | - | - | Час відправлення |
| time\_finish | datetime | - | - | Час прибуття |
| seats\_occupied | int | - | - | Кількість зайнятих місць |
| is\_active | bit | - | - | Чи активний маршрут |

Для збереження інформації про оплату була створена таблиця “orders”. В данну таблицю вводяться всі необхідні дані такі як id клієнта, id не авторизованого клієнта, час купівлі, вартість транзакції, і чи оплачено.

Структура таблиці наведена нижче:

**Таблиця 2.6.** Структура таблиці "orders"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва поля** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| id\_order | int | + | - | Id замовлення |
| client\_id | int | - | + | Id клієнта |
| non\_authorized\_users\_id | int | - | + | Id незареєстрованого користувача |
| date\_buying | datetime | - | - | Дата оплати |
| cost | float | - | - | Вартість |
| whether\_paid | bit | - | - | Оплачено чи ні |

Для збереження інформації про квиток було створено таблицю “tickets”. В данну таблицю вводяться всі необхідні дані такі як id автобуса, вибране місце, час купівлі, id подорожі, id не авторизованого користувача, id клієнта , id оплати, код квитка.

Структура таблиці наведена нижче:

**Таблиця 2.7.** Структура таблиці "tickets"

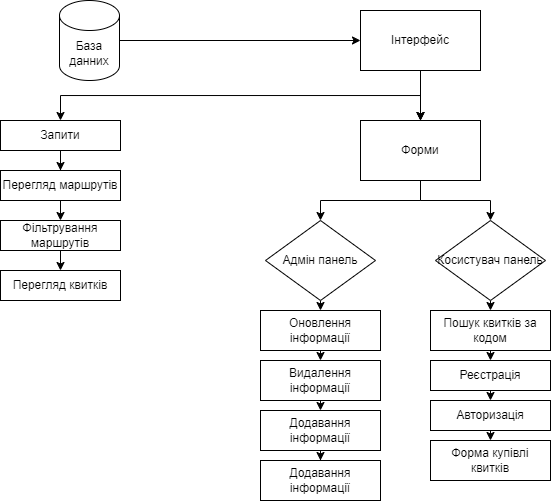
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва поля** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| id\_ticket | int | + | - | Id квитка |
| bus\_id | int | - | + | Id автобуса |
| seat | int | - | - | Номер місця |
| date\_buying | datetime | - | - | Дата покупки |
| journey\_id | int | - | + | Id маршруту |
| non\_autorized\_users\_id | int | - | + | Id незареєстрованого користувача |
| client\_id | int | - | + | Id клієнта |
| order\_id | int | - | + | Id замовлення |
| code | int | - | - | Код квитка |

Висновки до другого розділу **:**

У результаті виконання даного розділу було проаналізовано інформаційні процеси, спроектовано структуру бази даних, описано приблизну логіку функцій для роботи з базою на клієнті, поисано деяку логіку обробки, збереження та генерації даних для бази. Також було описано як буде відбуватись побудова та генерація статистичної інформації про сайт. Крім цього, було описано логіку взаємодії між даними

# РОЗДІЛ 3. РЕАЛІЗАЦЯ САЙТУ АВТОВОКЗАЛУ

## 3.1. Проектування інтерфейсу системи сайту автовокзалу

****

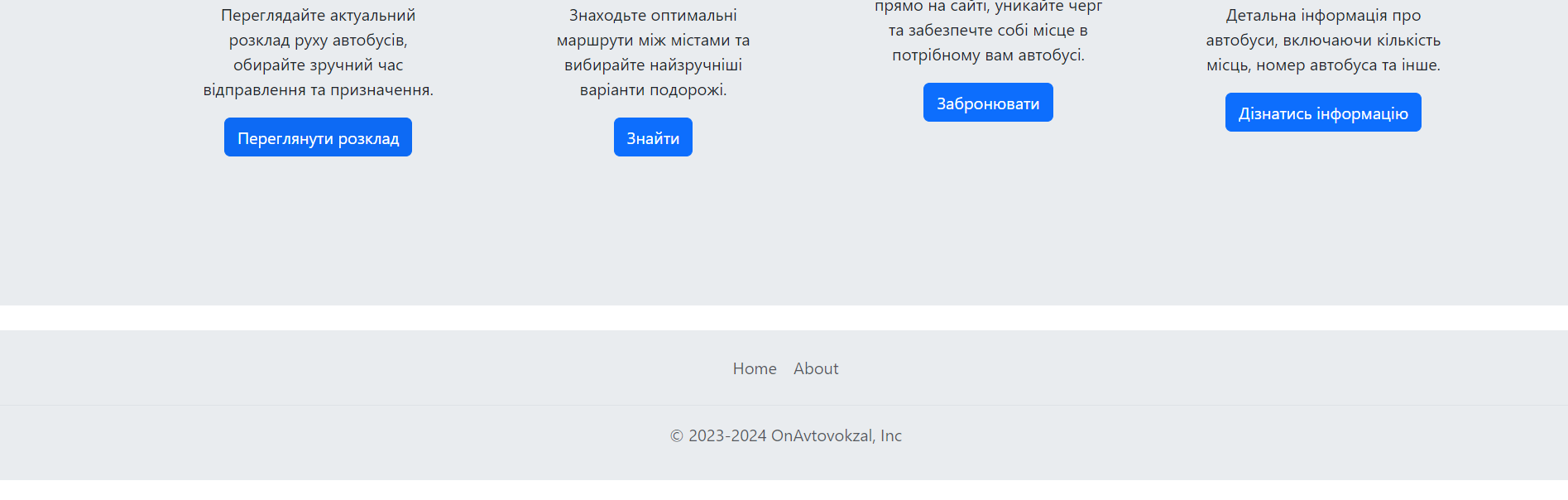
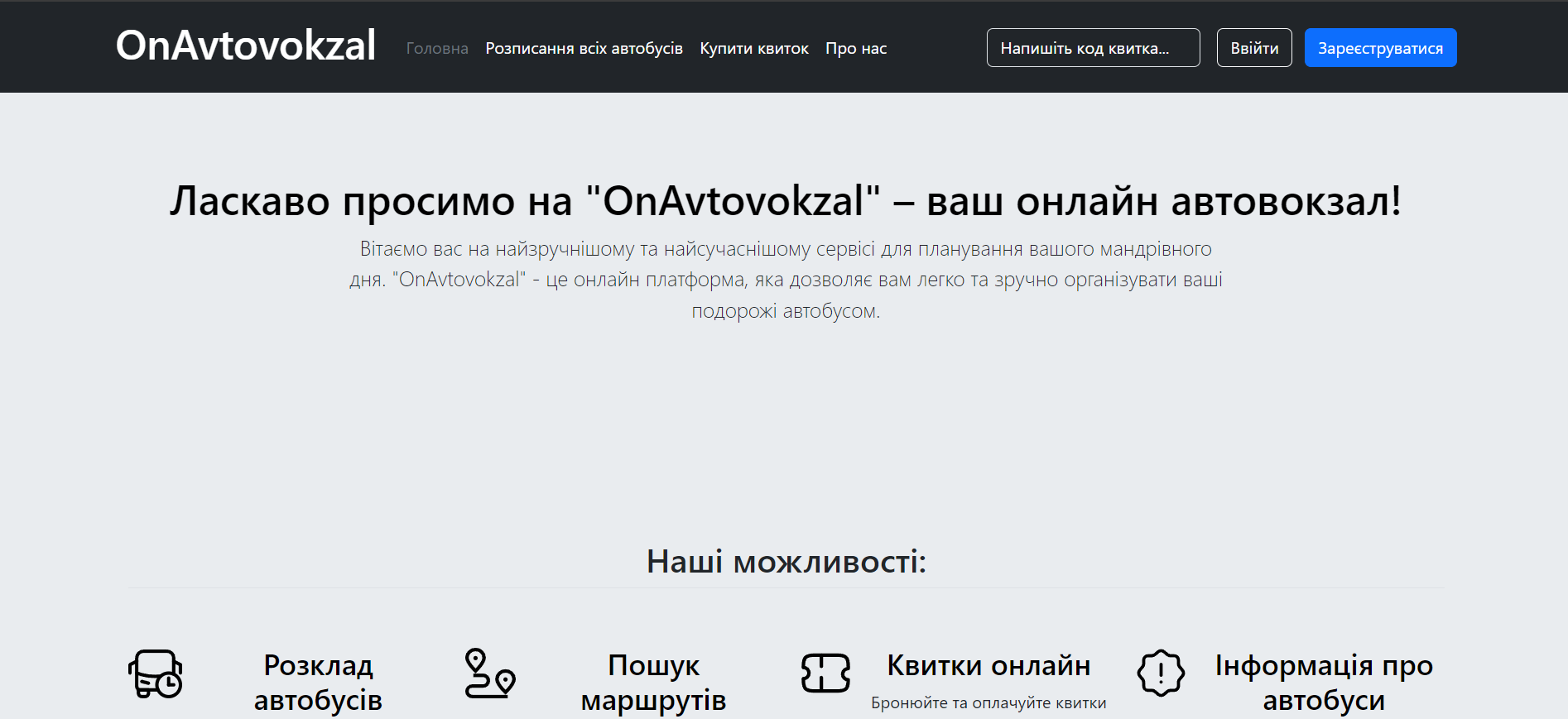
**Рис.3.1.** Структура інтерфейсу

На рисунку 11 зображено загальну схему інтерфейсу сайту автовокзалу. Далі будуть зображені форми та сторінки, що показують фінальний результат виконання проектування інтерфейсу веб-сайту. Форми супроводжуються кодом виконання та запису данних в БД.

Для розробки інтерфейсу було використано мову гіпертекстової розмітки – HTML, стилі – CSS, а також інструмент Bootstrap.

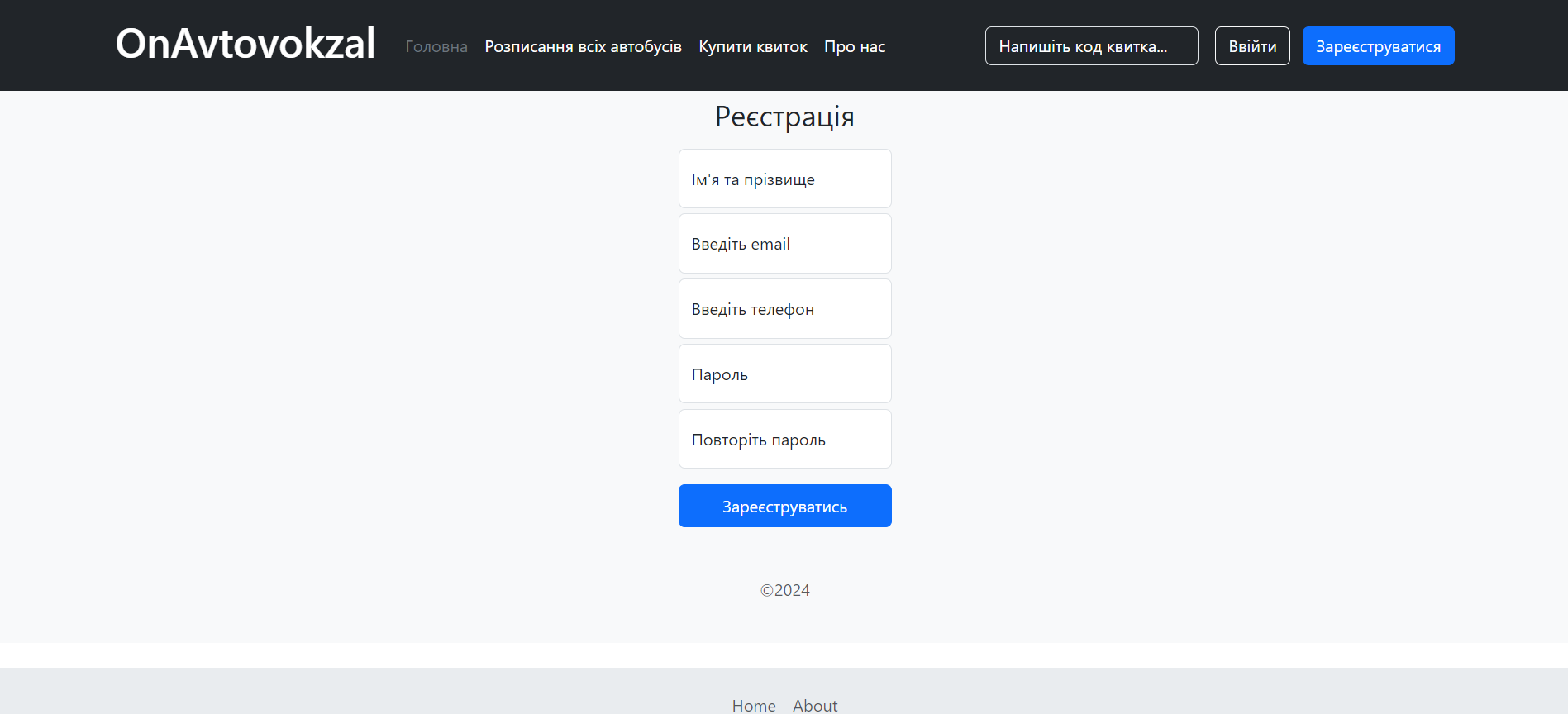
**Форми та сторінки інтерфейсів:**

Відкриваємо наш сайт за адресою: <http://127.0.0.1:5000/home> . При відкритті сайту нас зустрічає головна сторінка. На головній сторінці ми можемо помітити декілька посилань на інші сторінки, такі як “Головна”, “Розписання всіх автобусів”, “Купити квитки”, “Про нас”, форма для пошуку квитків за кодом, та блок з авторизацією. Вміст самої сторінки та посилання на відповідні сторінки.



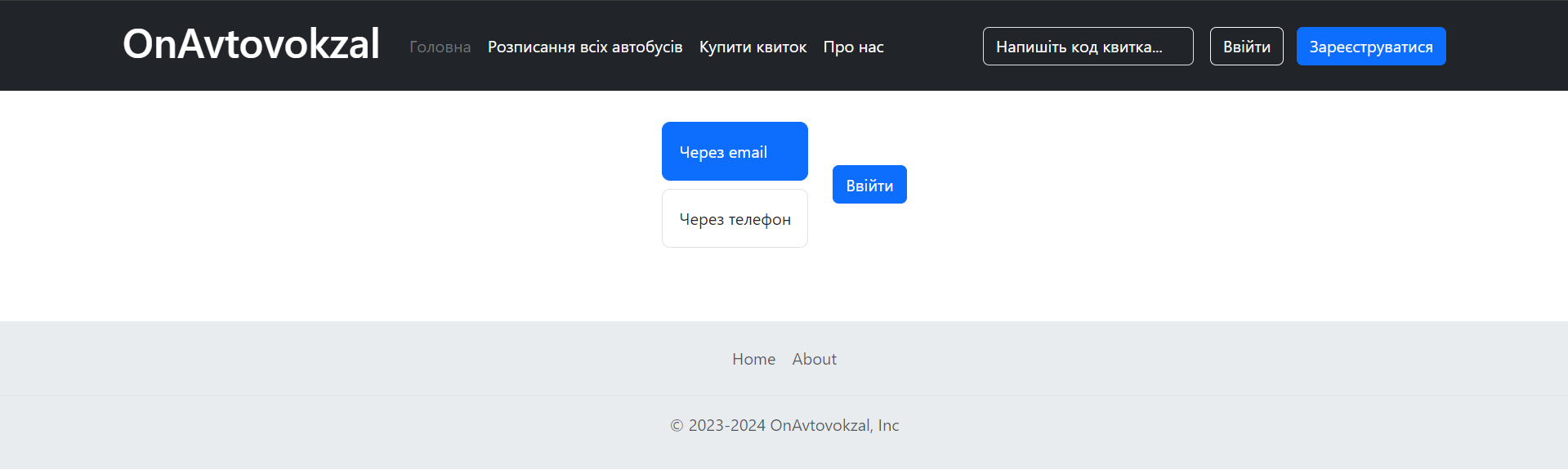
**Рис.3.2.** Головна сторінка сайту

Реєструємось на сайті. Форма реєстрації приймає ім’я та прізвище, пошту, номер телефону, пароль та повторний пароль. Перед відправкою форми на сервер відбувається перевірка того, чи співпадають паролі та чи існує користувач із такою самою поштою, в випадку неспівпадіння, користувача повідомляють про це і попросять змінити дані. Після реєстрації користувач автоматично входить в акаунт.

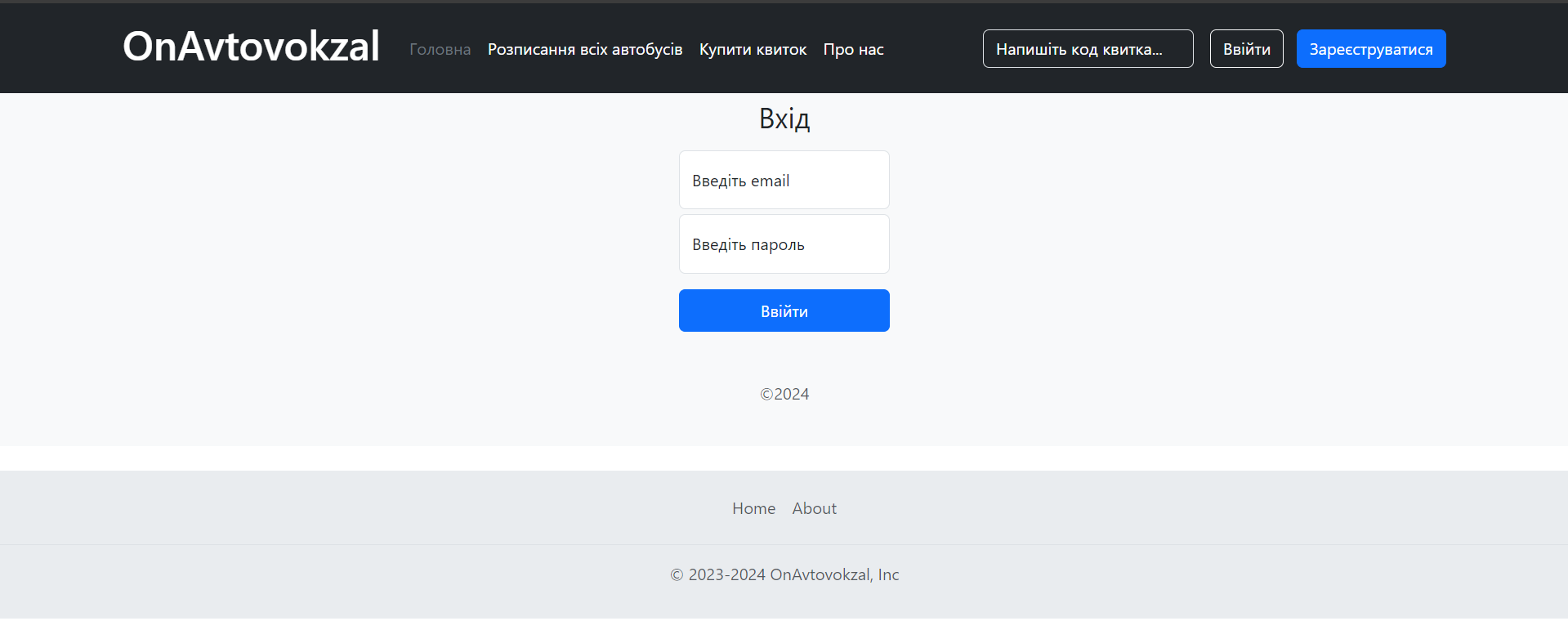


**Рис.3.3.** Форма для реєстрації користувача в систему

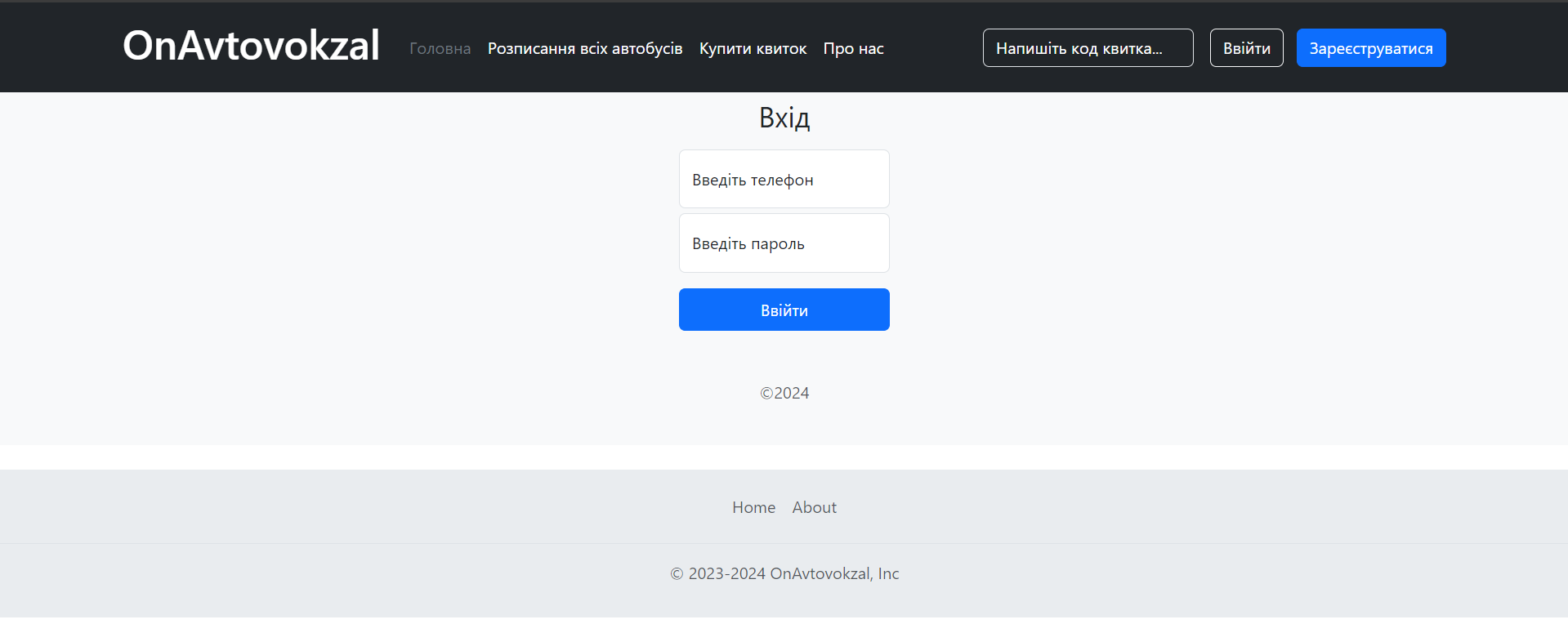
При спробі входу появляється вікно з параметрами входу: телефон, email. Відповідно користувач виберає за чим входити і надає данні. Після успішного входу користувача переадресовують на головну сторінку та змінюється блок авторизації, що показано на рисунку 17. При невдалом вході показується помилка та переадресовується для авторизації заново, представлено на рисунку 18.



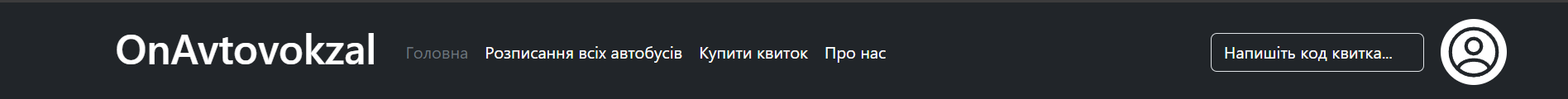
**Рис.3.4.** Вибір за якими параметрими будемо входити в акаунт



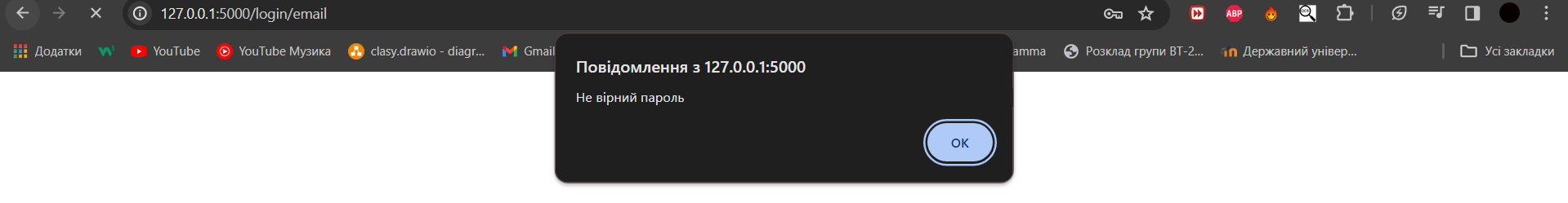
**Рис.3.5.** Сторінка авторизації через email



**Рис.3.6.** Сторінка авторизації через телефон

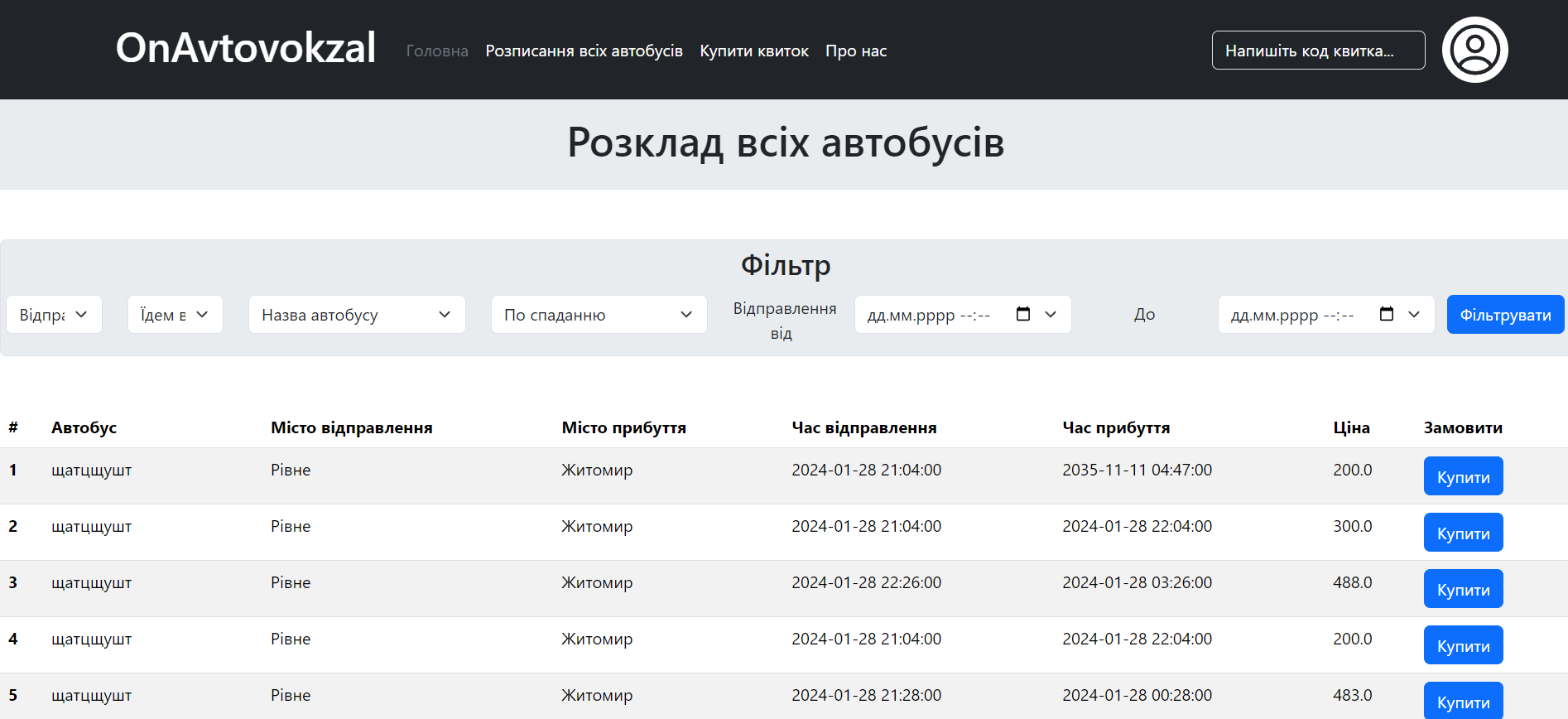


**Рис.3.7.** Успішний вхід

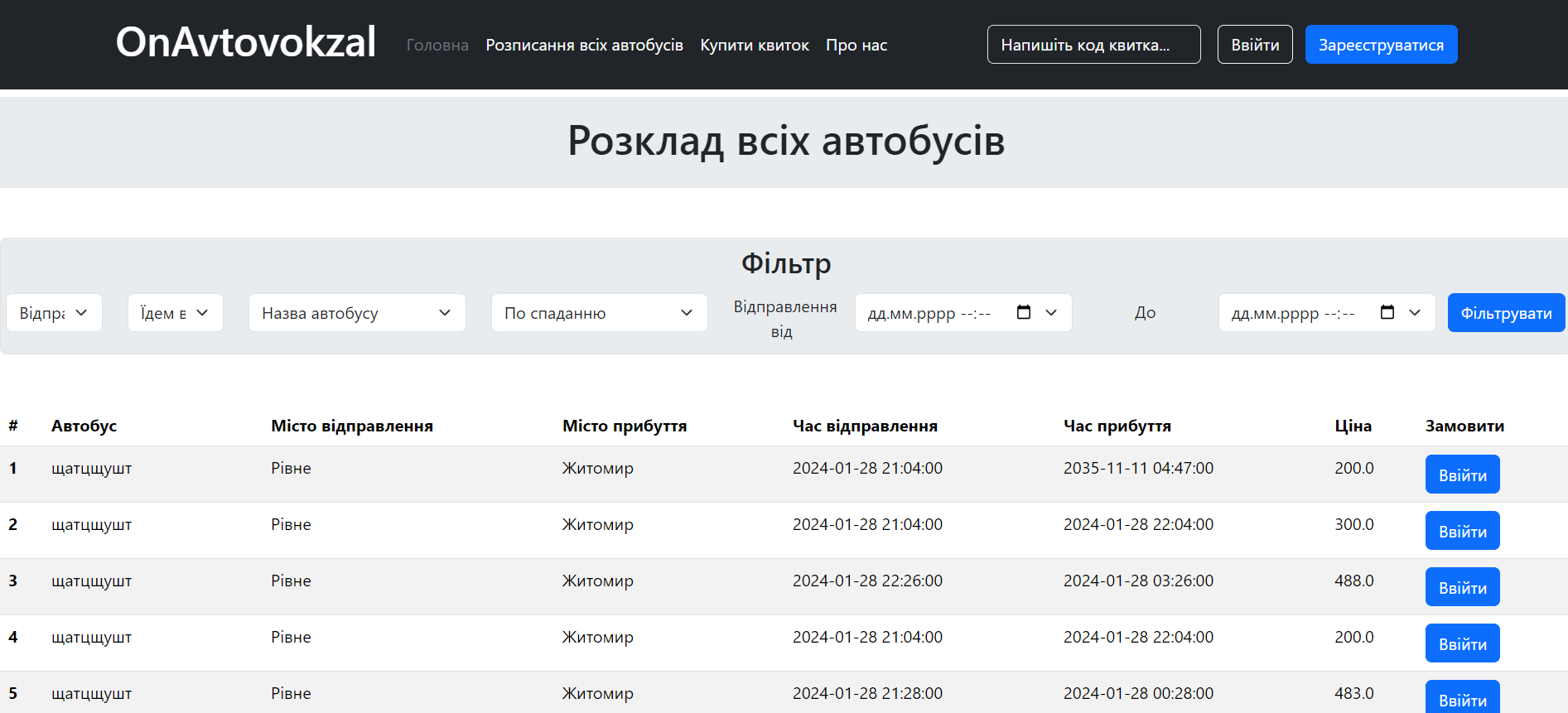
****

**Рис.3.8.** Вхід невдалий

На сторінці “Розписання всіх автобусів” ми можемо переглянути табло всіх маршрутів, яке формується за допомогою запитів до БД та отримання інформації. У випадку коли користувач не авторизований сайт вимагає авторизації, в іншому випадку можна перейти до покупки. Ще є можливість фільтрації маршрутів за місцем відправлення та прибуття, за датою відпралення та прибуття, також присутнє сортування по спаданню та зростанню за такими параметрими : за назвою автобусу в алфавітному порядку, за іменем міста прибуття\відправлення, за часом прибуття\відправлення. Фільтрація відбувається за допомогою select запитів в базу данних.



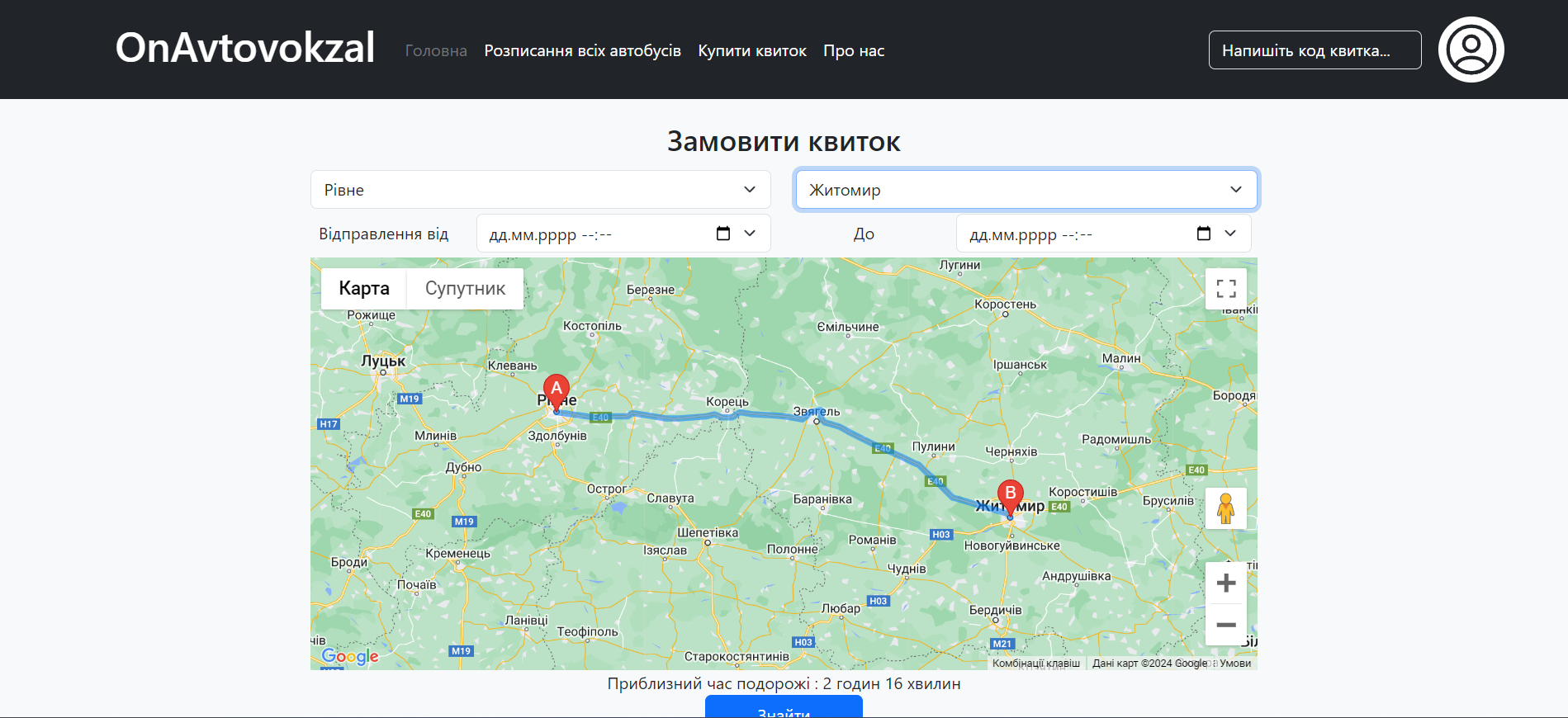
**Рис.3.9.** Сторінка “Розписання автобусів” коли користувач авторизований

****

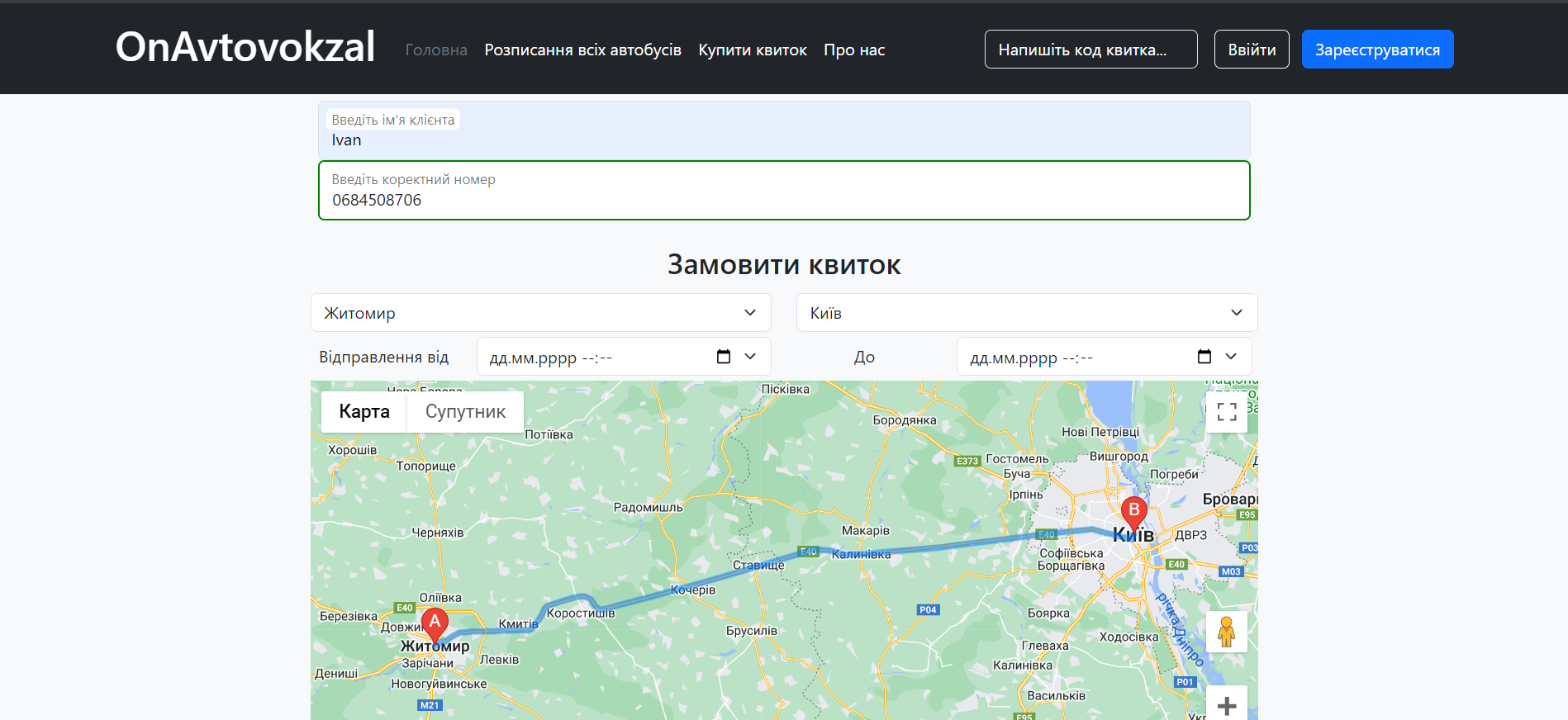
**Рис.3.10.** Сторінка“Розписання автобусів” коли користувач не авторизований

Переглядаємо сторінку “Купити квиток”. Коли користувач авторизований

йому надається можливість вибрати шлях і час поїздки, в іншому випадку йому потрібно ще ввести свої контактні дані. Тут користувач може переглянути маршрут і приблизний час подорожі.

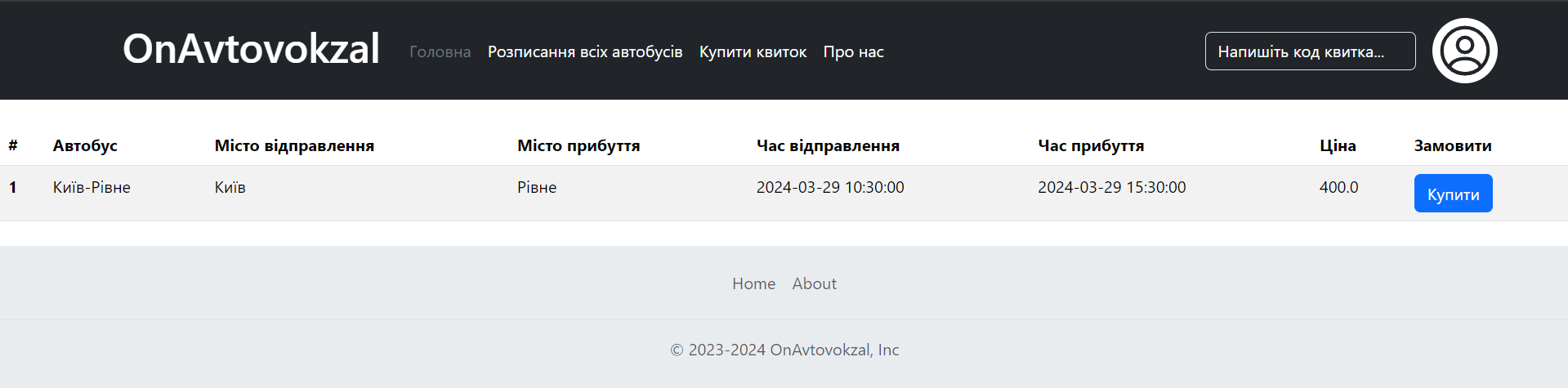


**Рис.3.11.** Сторінка “Купити квиток” коли користувач авторизований

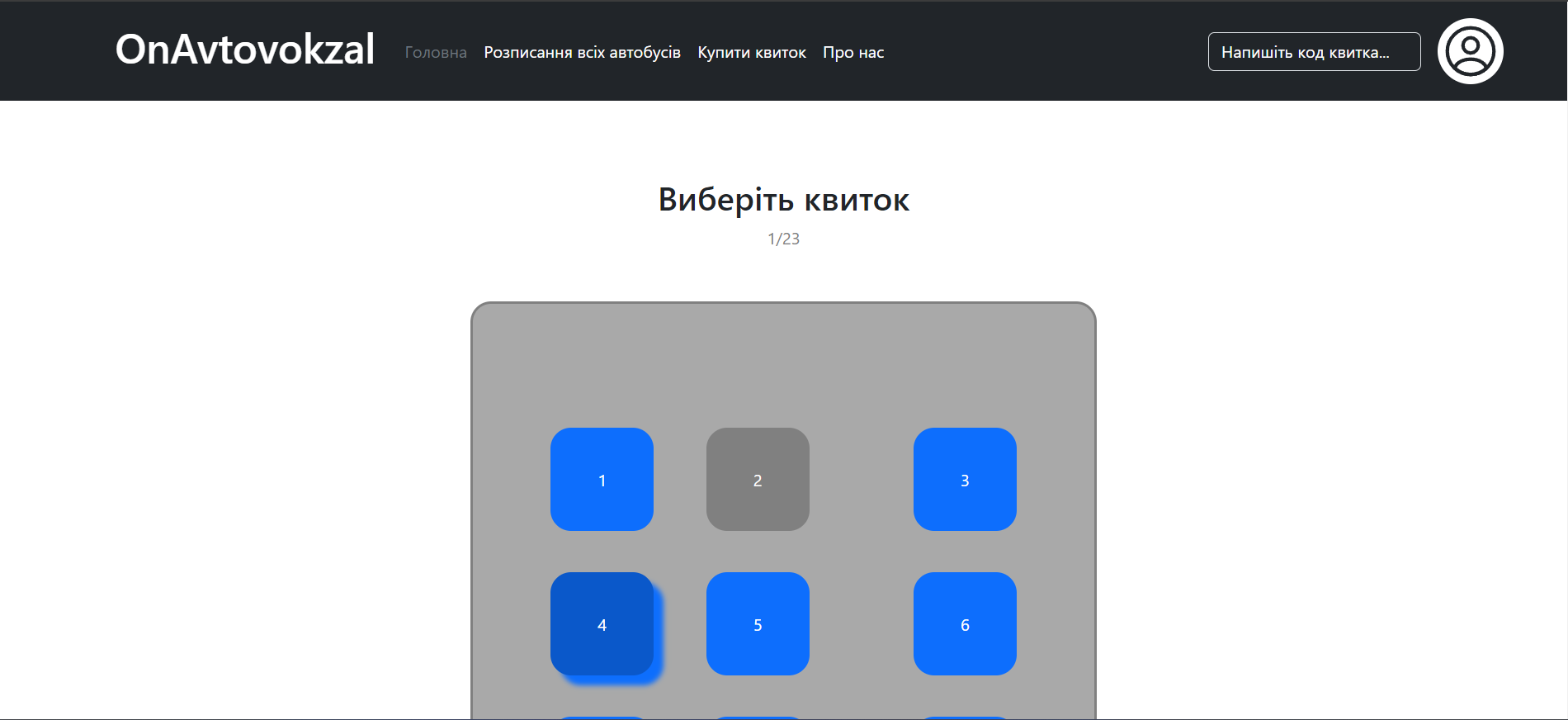


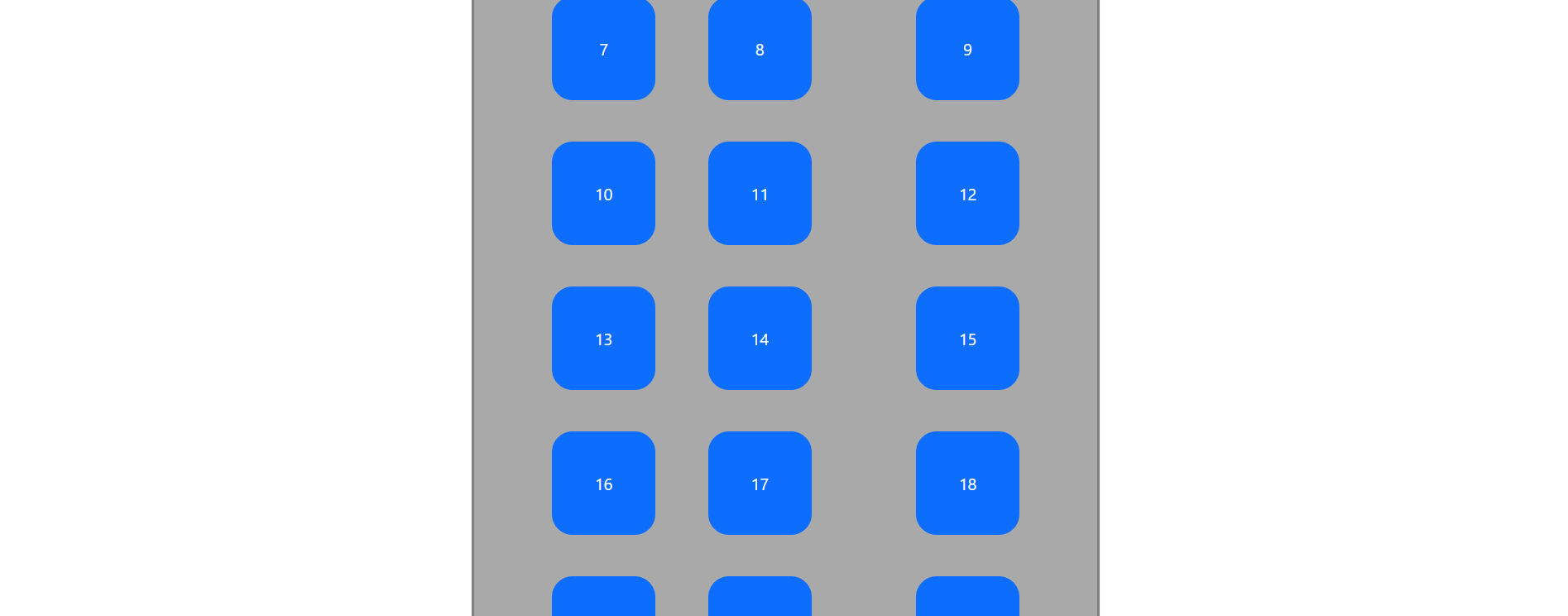
**Рис.3.12.** Сторінка “Купити квиток” коли користувач не авторизований

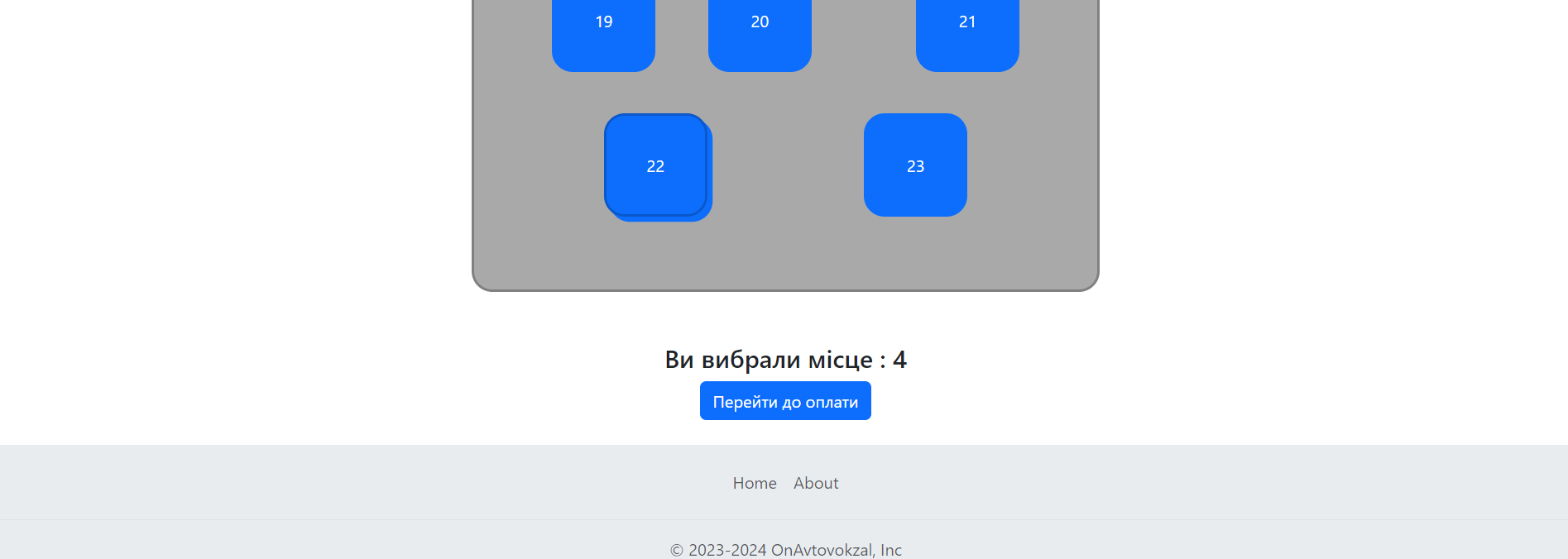
Після вибору шляху та часу робиться get запит з аргументами на сервер, сервер в свою чергу звертається до БД та за допомогою select запитів шукає потрібну інформацію та повертає всі маршрути з відповідними полями.



**Рис.3.13.** Сторінка після вибору напрямку та часу

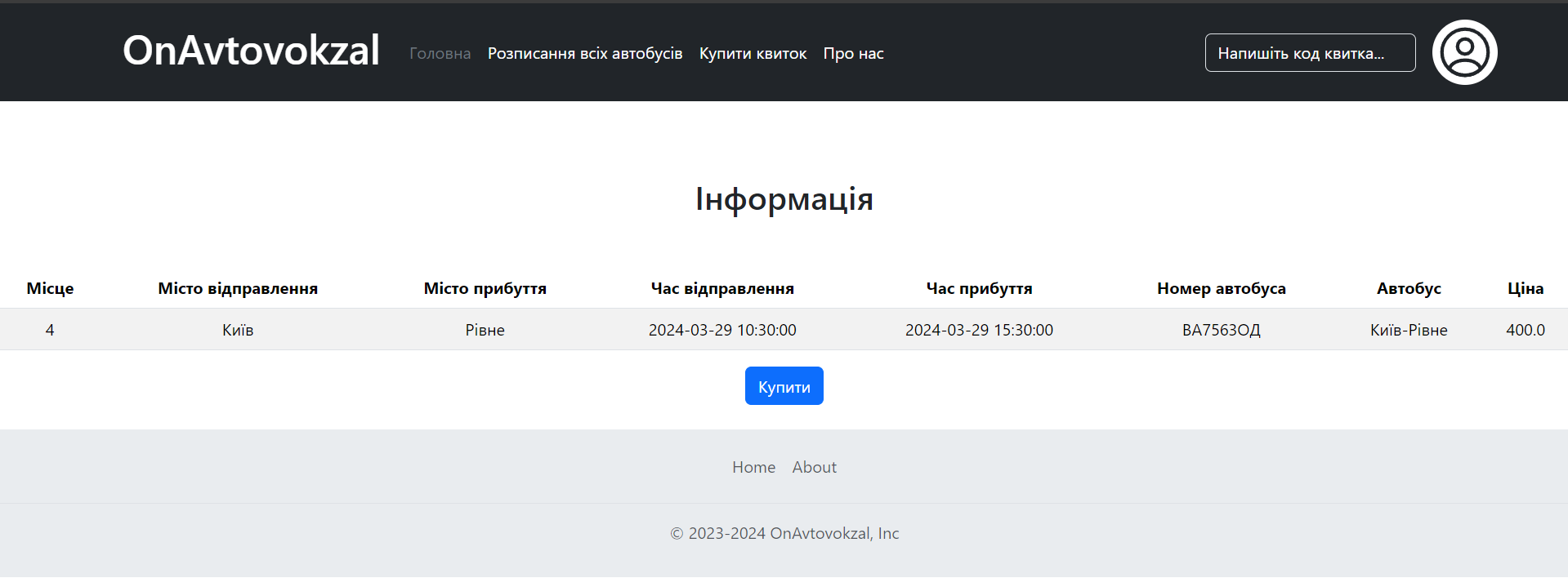






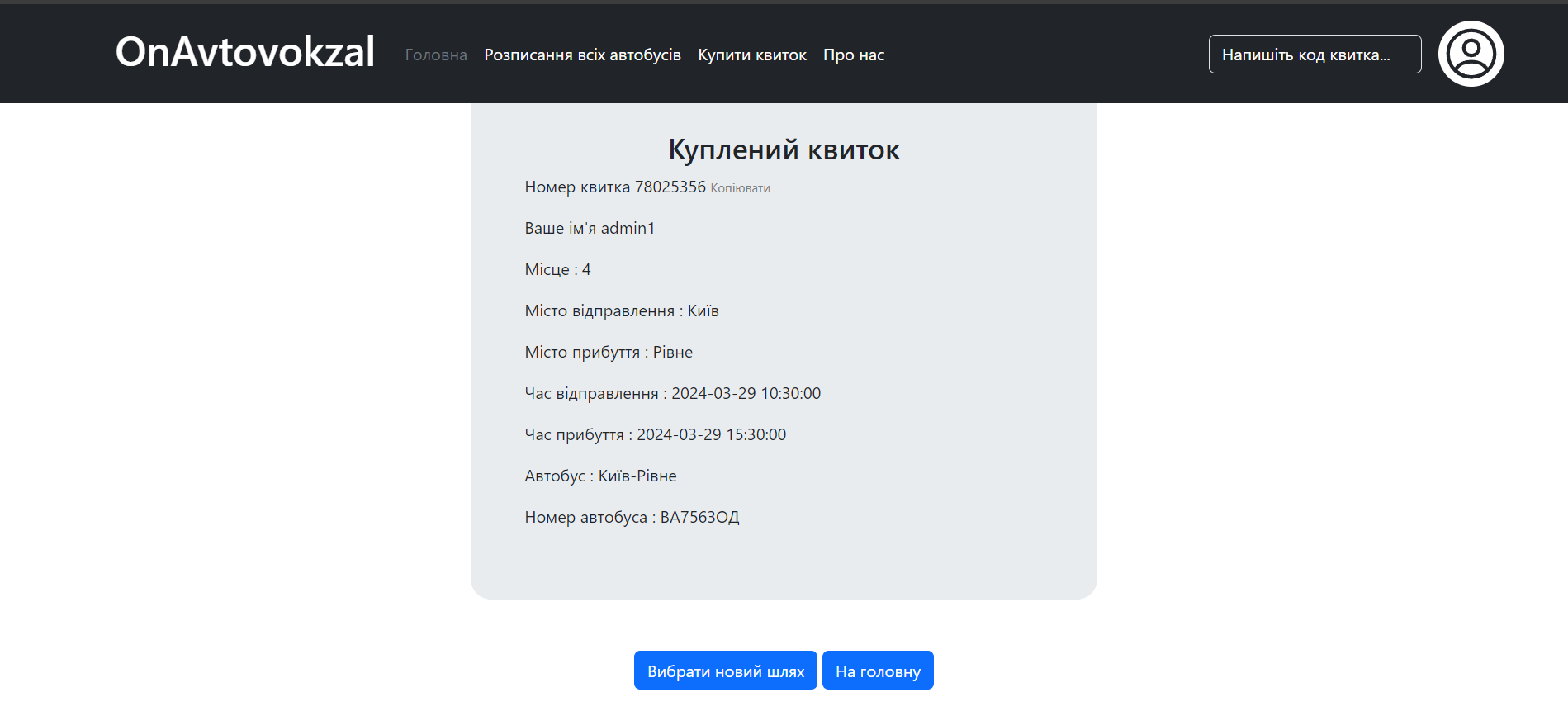
**Рис.3.14** Сторінка вибору місця посадки

На рисунку 24 показано дію після вибору маршруту, ми вибераємо своє сидіння на якому бажаємо провести подорож, зверху ми можемо побачити кількість вільних та зайнятих місць.Сірим колором показані зайняті місця , синім показані вільні , темно-синім ваше вибране місце. В наступному, рисунок 25 ,вікні людина перевіряє інформацію та підтверджує оплату.

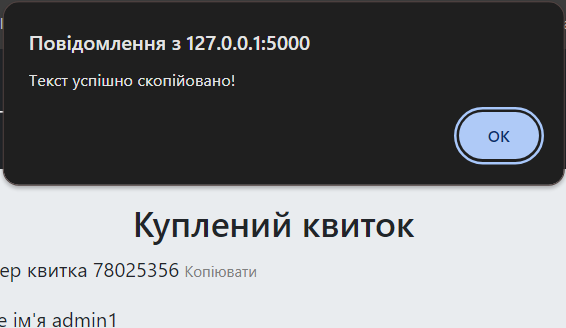


**Рис.3.15.** Підтвердження інформації

Після підтвердення проходить оплата і надається квиток із своїм унікальним кодом, та вся інформація про поїздку.

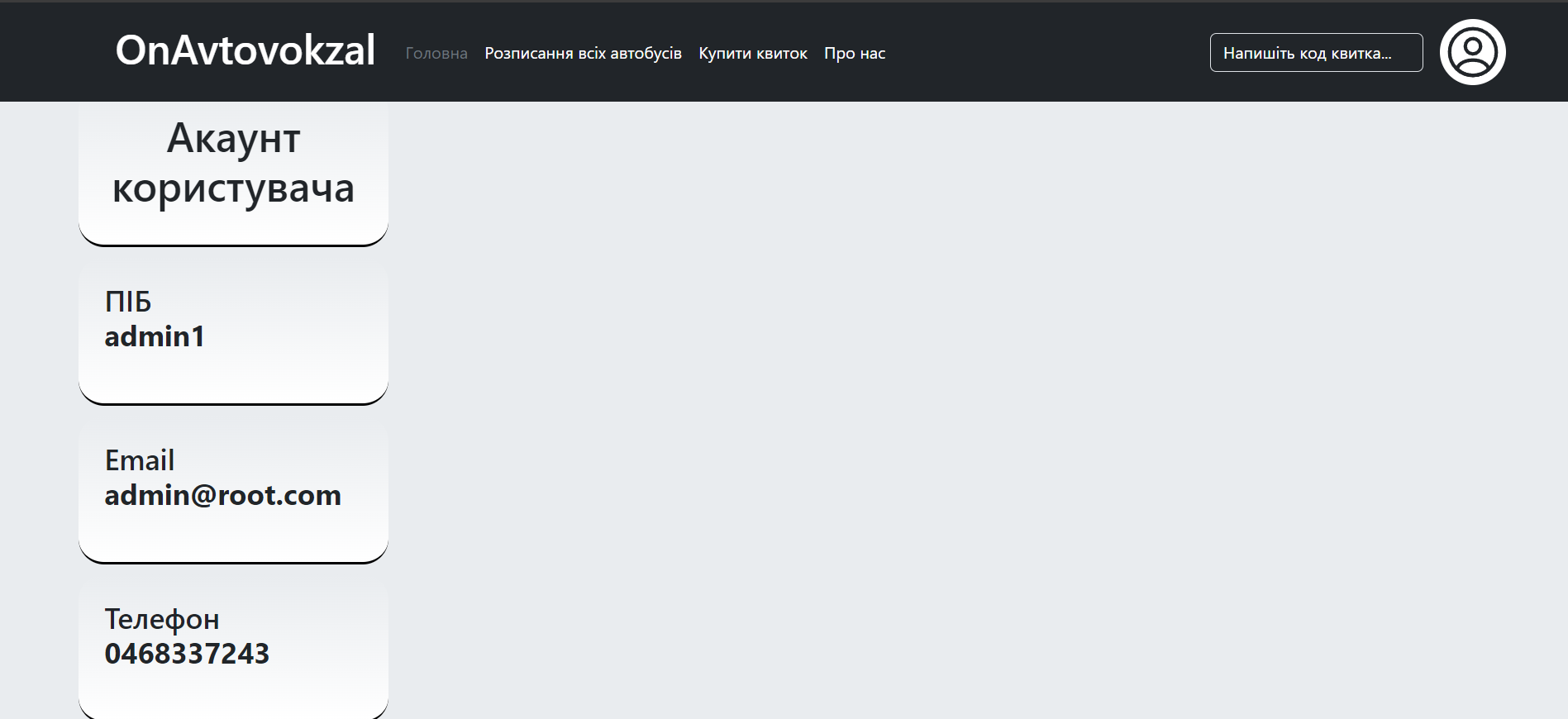


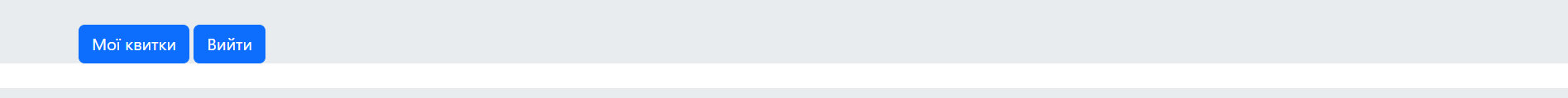
**Рис.3.16.** Отримання квитка та збереження йогов базі данних



**Рис.3.17.**Копіювання коду квитка

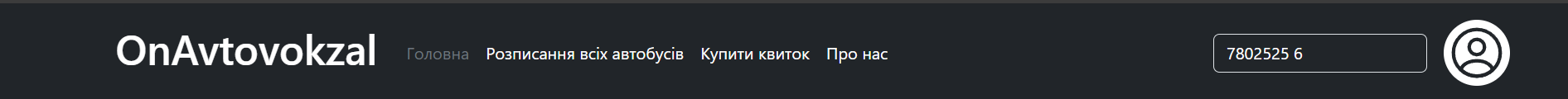
На римсунку 28 показано особистий кабінет користувача тут ми можемо побвчити телефон, email та ПІБ. Також є кнопка вийти яка видаляє користувача з сесії та кнопка мої квитки.





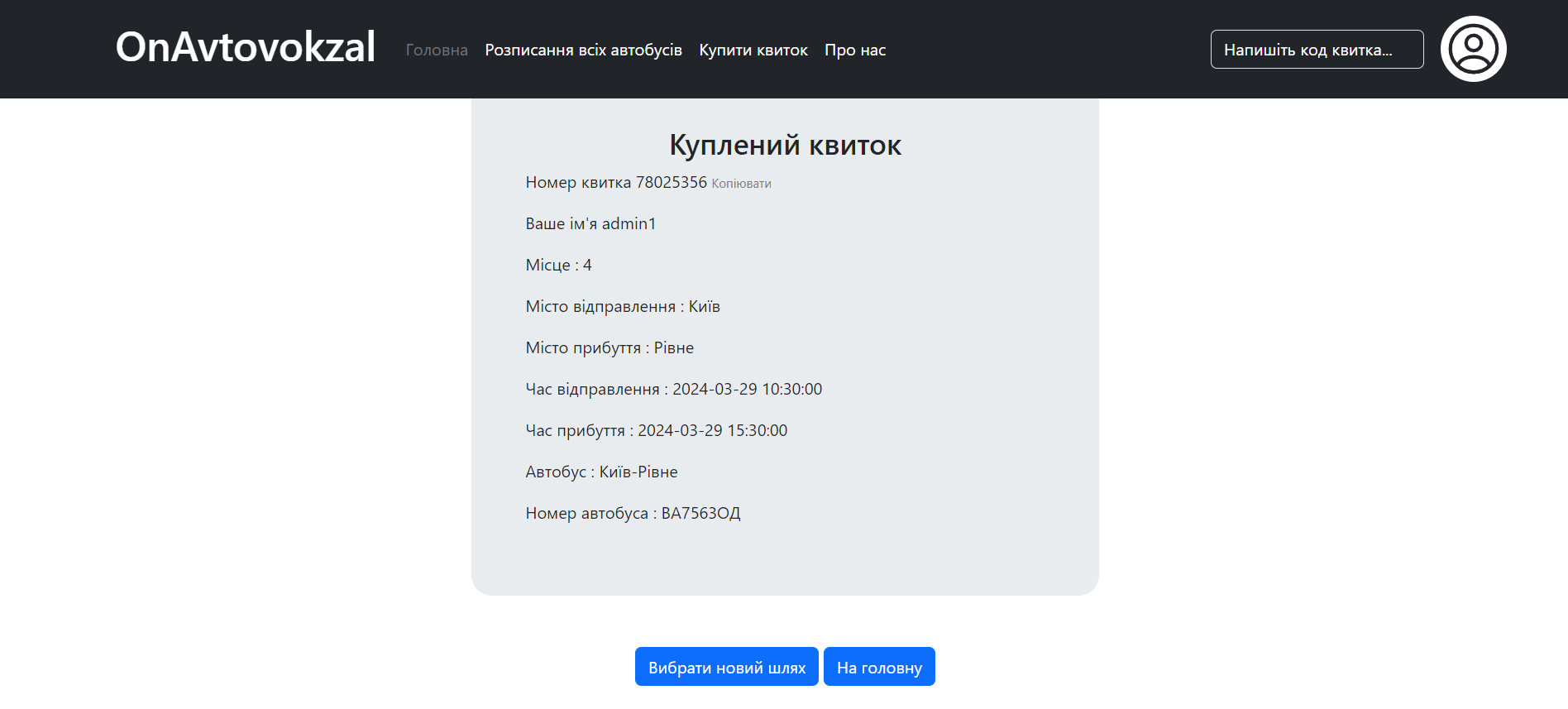
**Рис.3.18.** Переглід інформації акаунту

Тепер ми можемо знайти цей квиток як за його кодом так і за допомогою форми для пошуку квитків за кодом.



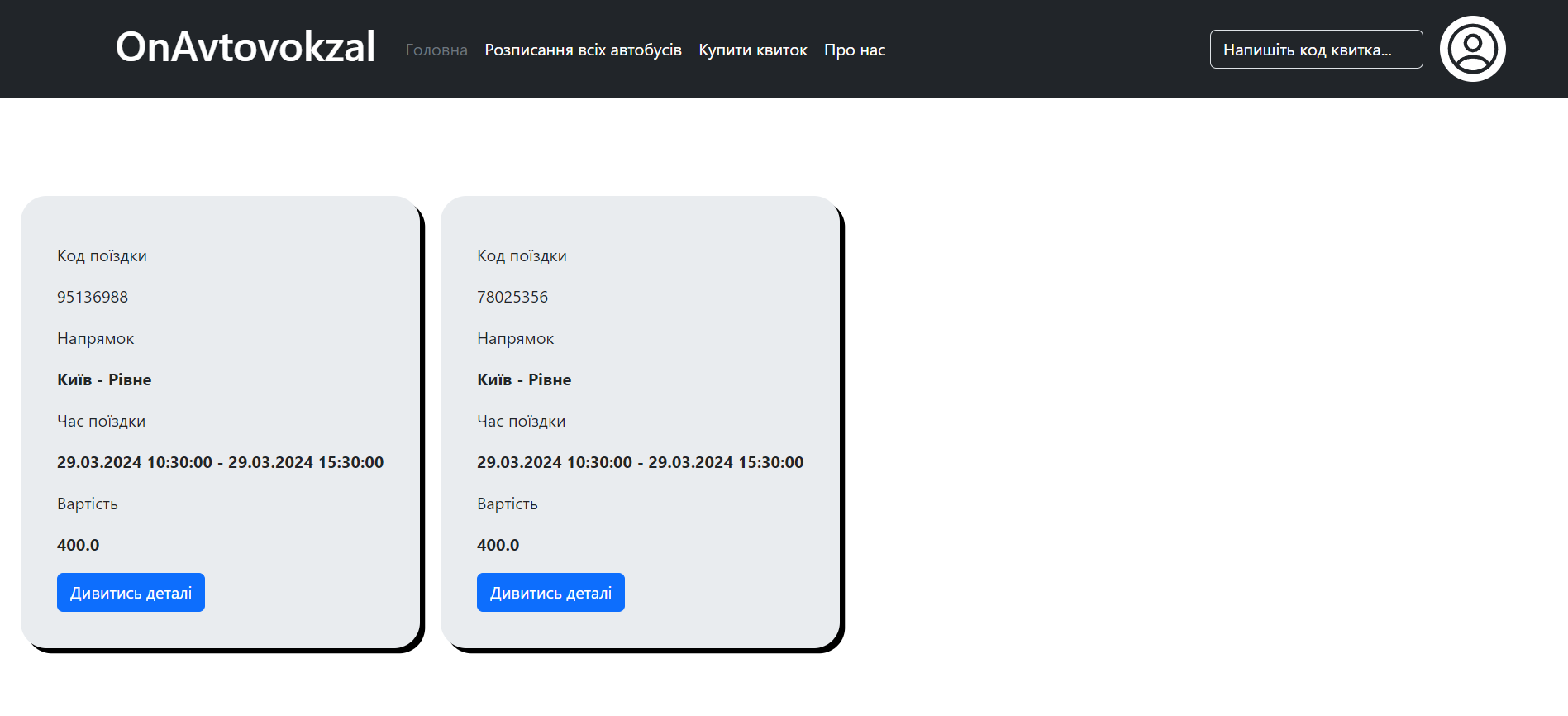
**Рис.3.19.** Пошук за номером квитка

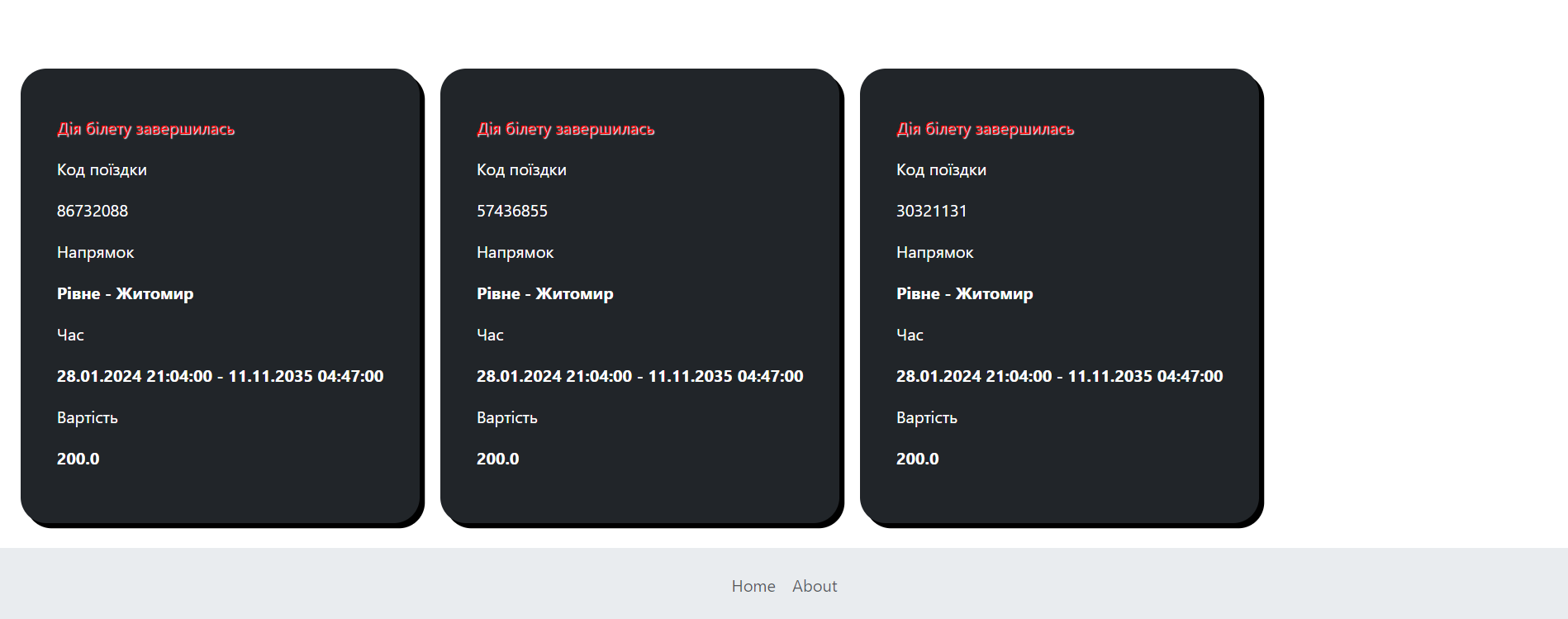
Тільки ввівши всі цифри(без пробілів) квитка він розпочинає швидкий пошук та повертає саме цей квиток який нам потрібен.



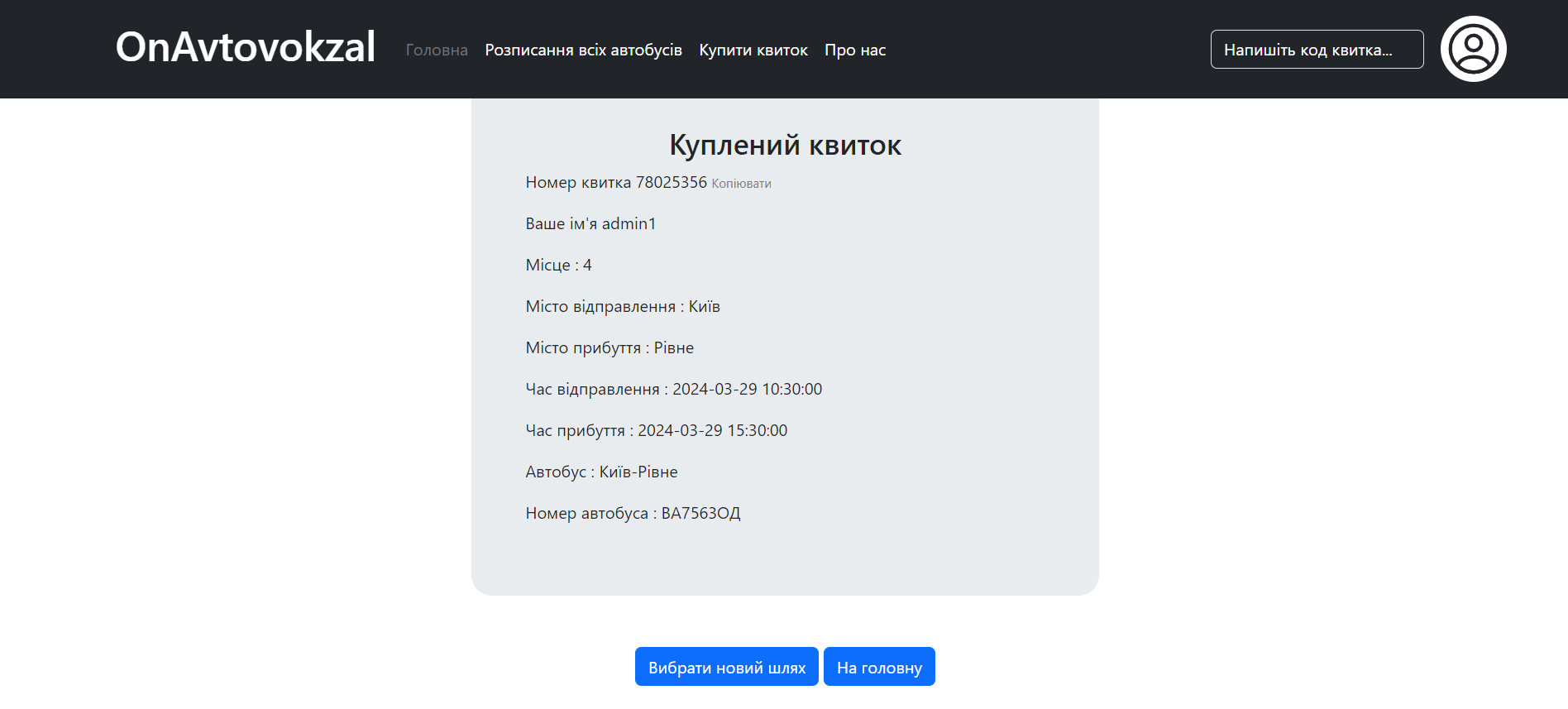
**Рис.3.20.** Результат пошуку

Зайшовши у власний кабінет та вибрати “Мої квитки”, та матимемо можливість переглянути квитки. Дія білетів вважається завершеною, коли подорож починається. Після клікання на кнопку "дивитись далі", користувач отримує можливість переглянути докладнішу інформацію про маршрут. Інформація про квитки доступна завдяки запитам за їхніми ідентифікаторами та ідентифікаторами відповідних даних.



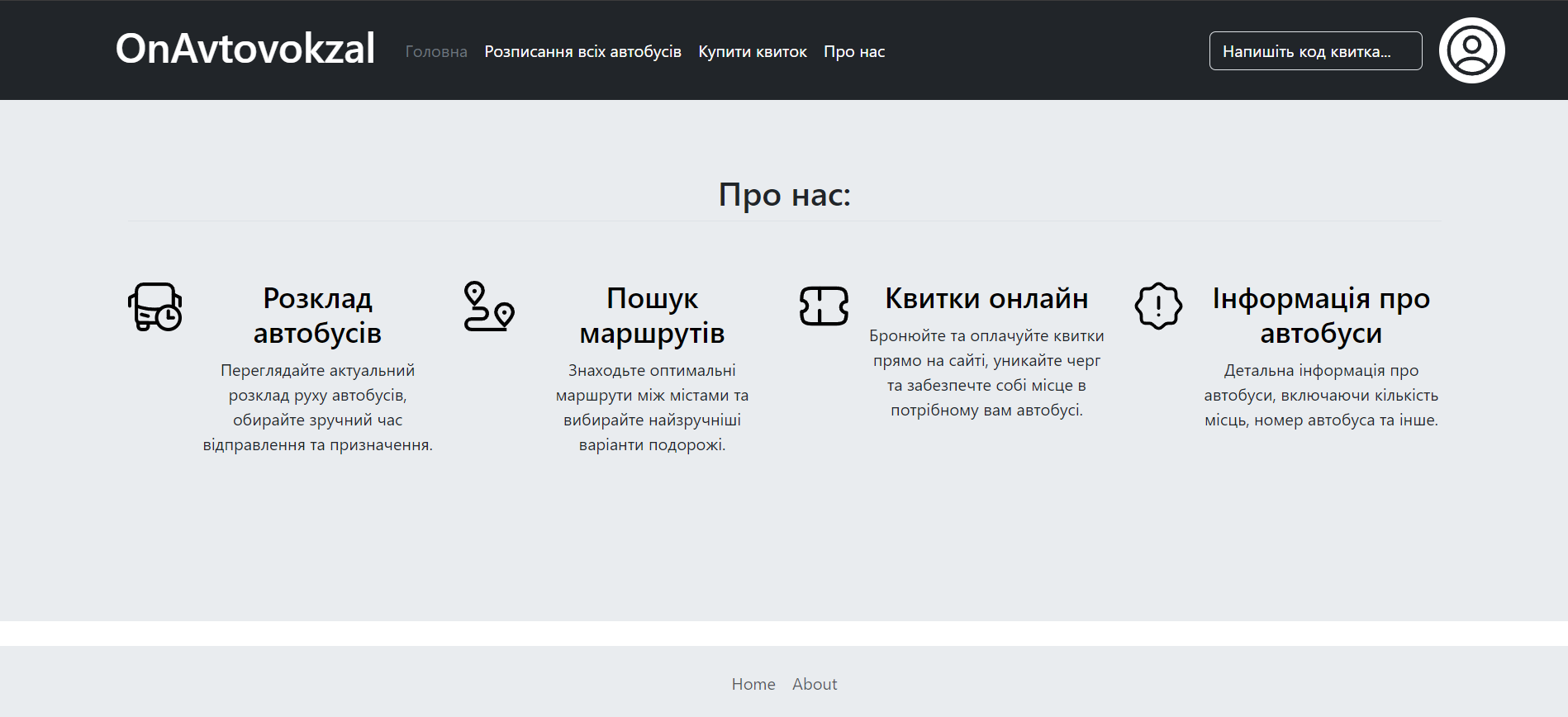


**Рис.3.21.** Сторінка перегляду квитків



**Рис.3.22.** Результат перегляду

На сторінці “Про нас” можна знайти інформацію про компанію та її переваги.



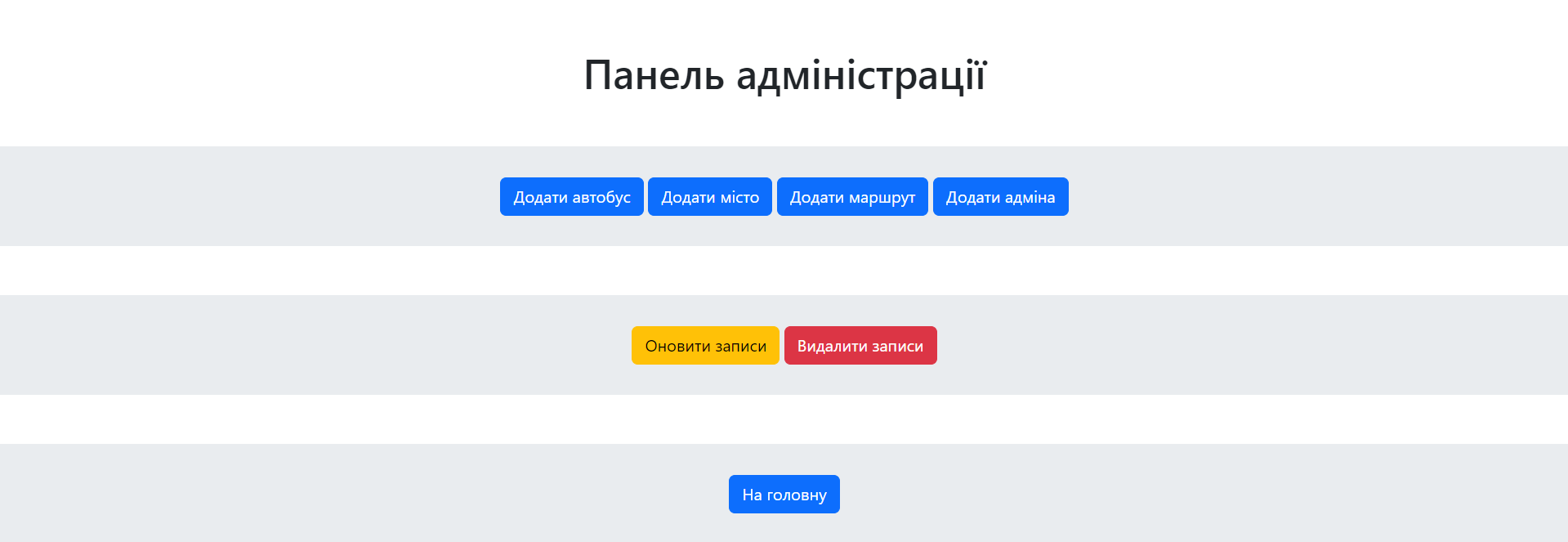
**Рис.3.23.** Сторінка “Про нас”

**Адмінпанель**:

Доступ до панелі адміністратора є тільки в адмінів, кожен користувач має свою роль, на данному етапі є тільки дві ролі такі як user та admin.

Щоб прейти в адмінпанель нам потрібно, авторизуватись з сторінки адміністратора та в полі пошуку прописати <http://127.0.0.1:5000/adminpanel>

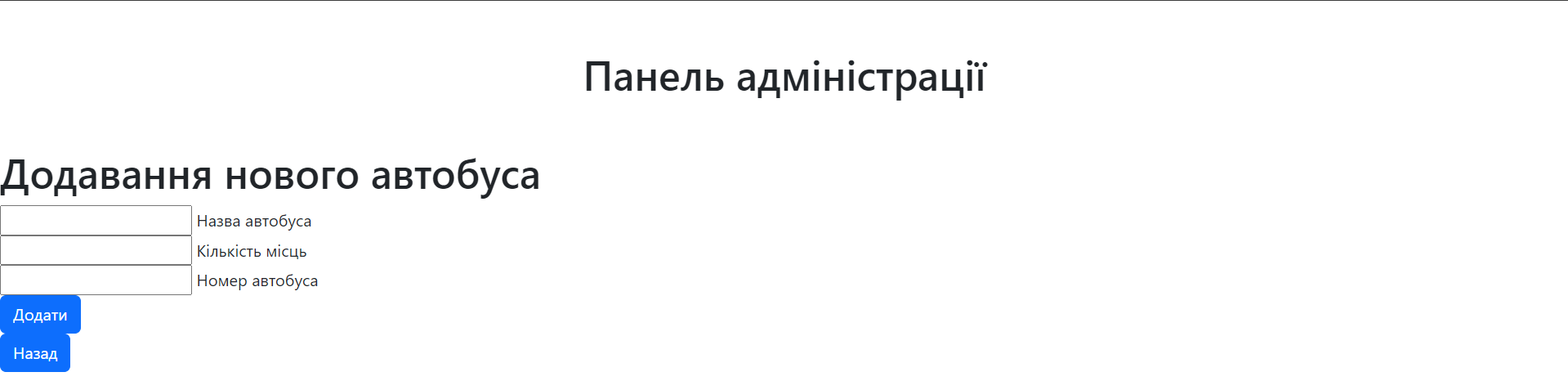
На головній сторінці ми можемо побачити кнопки для додавання інформації, оновлення та видалення, таким чином ця сторінка реалізує посилання в совокупності на всі процеси CRUD – Create, Read, Update, Delete. Завдяки цій панелі ми можемо керувати інформацією комфортно та безпосередньо з сайту без застосування СУБД.



**Рис.3.24.** Головна сторінка адмін панелі

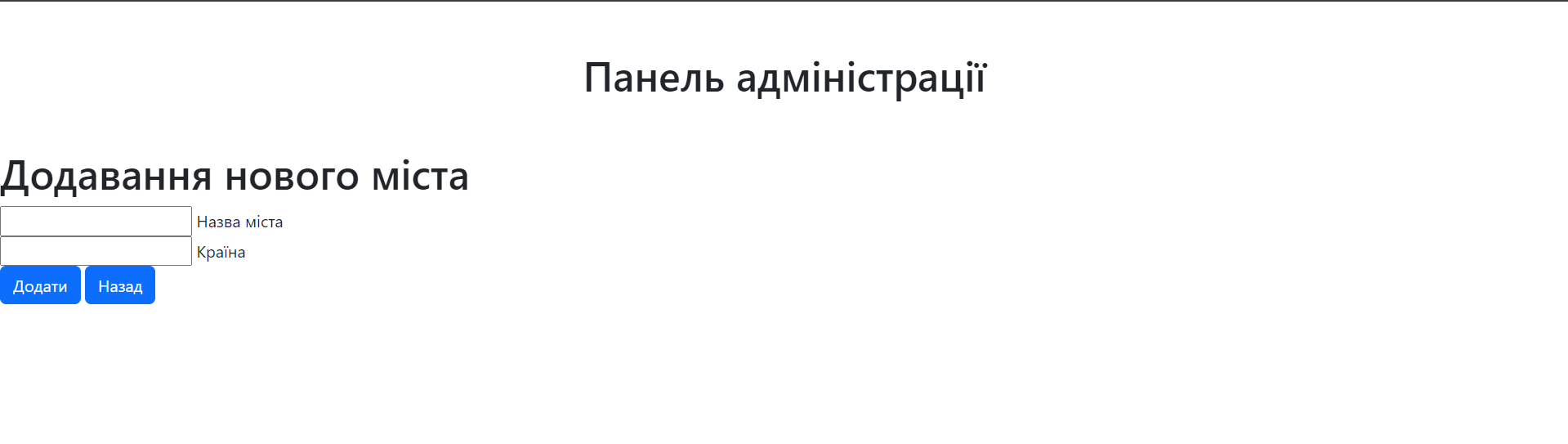
**Операції Create:**

На сторінці “Додати автобус”ми можемо помітитиполя длянадання назви,кількості мість та номеру нового автобуса. У випадку помилки показується аналогове вікно з текстом “Автобус не додано”.



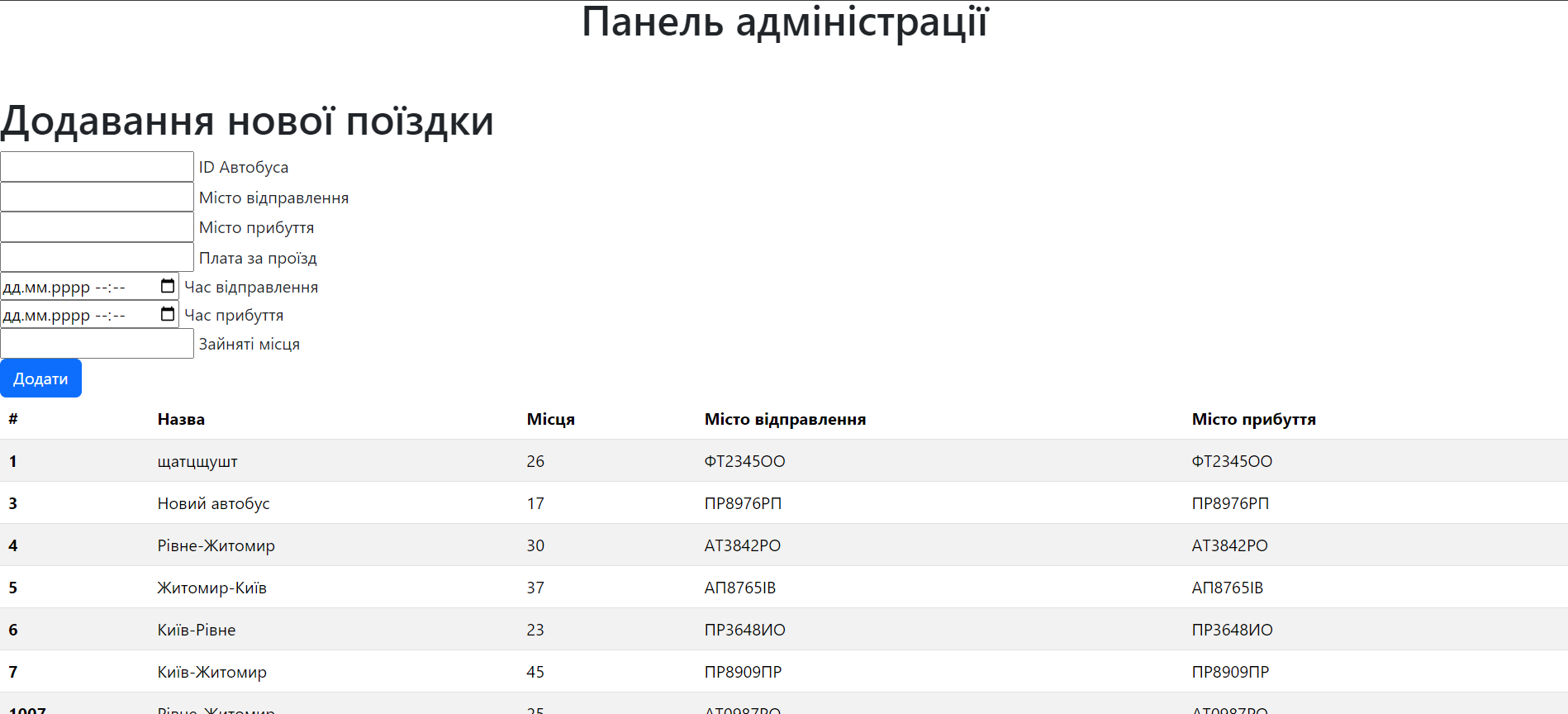
**Рис.3.25.** Сторінка яка реалізує додавання нового автобусу

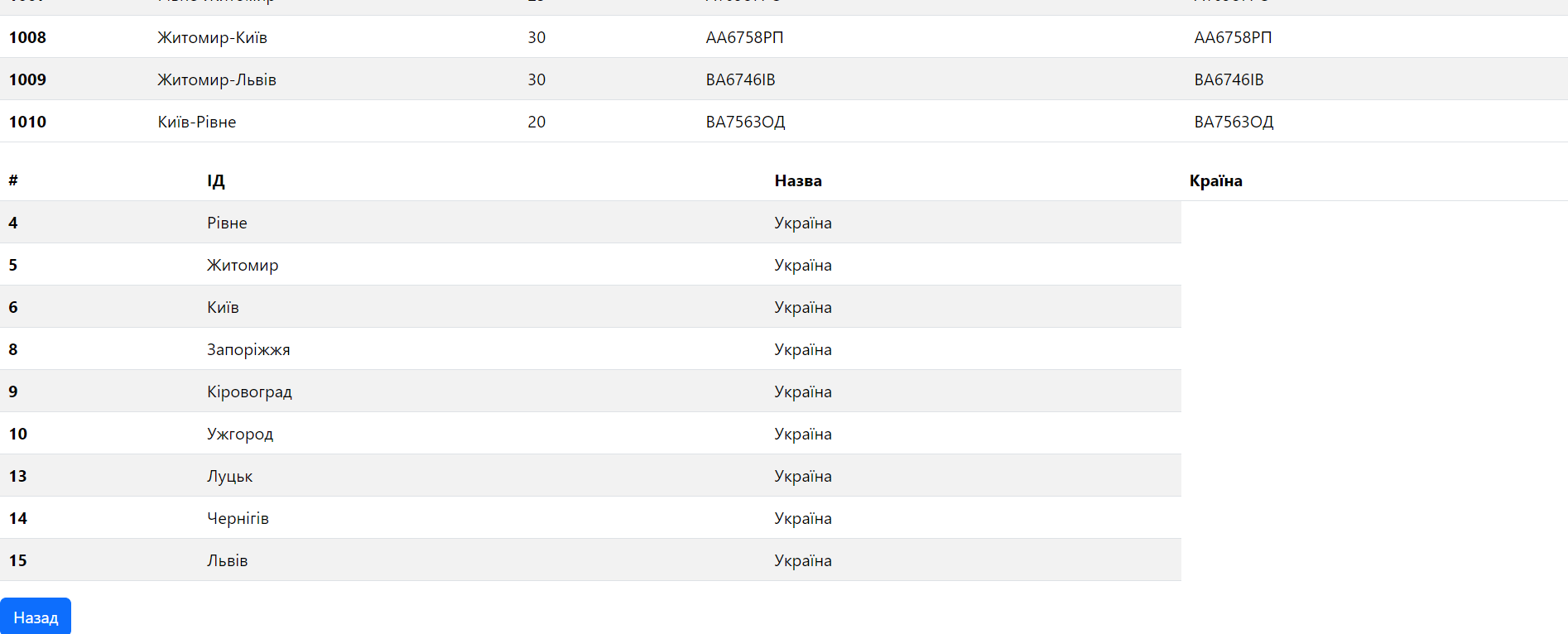
На сторінці “Додати місто”ми можемо помітитиполя длянадання назви та країни. У випадку помилки показується аналогове вікно з текстом “Місто не додано”.



**Рис.3.26.** Сторінка яка реалізує додавання нового міста

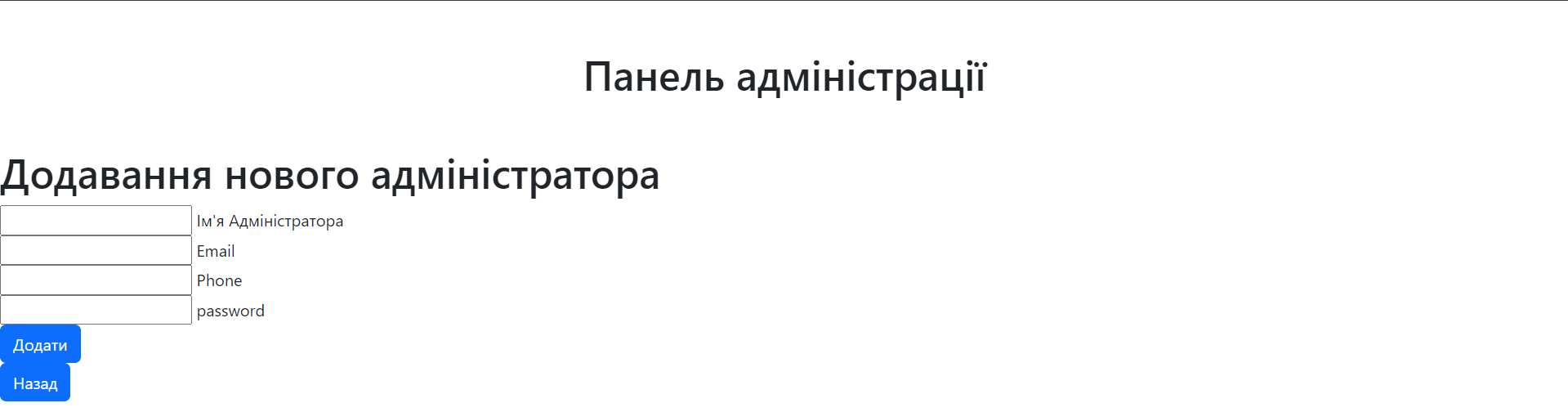
На сторінці “Додати маршрут”ми можемо помітитиполя длянадання id автобуса, міста відправлення та прибуття, плату за проїзд, час відправлення та прибуття і кількість вже зайнятих місць. У випадку помилки показується аналогове вікно з текстом “Маршрут не додано”. На цій сторінці є ще список міст та автобусів з їхніми id для легкого додавання їх ідентифікаторів в новий запис.





**Рис.3.27.** Сторінка яка реалізує додавання нового маршруту

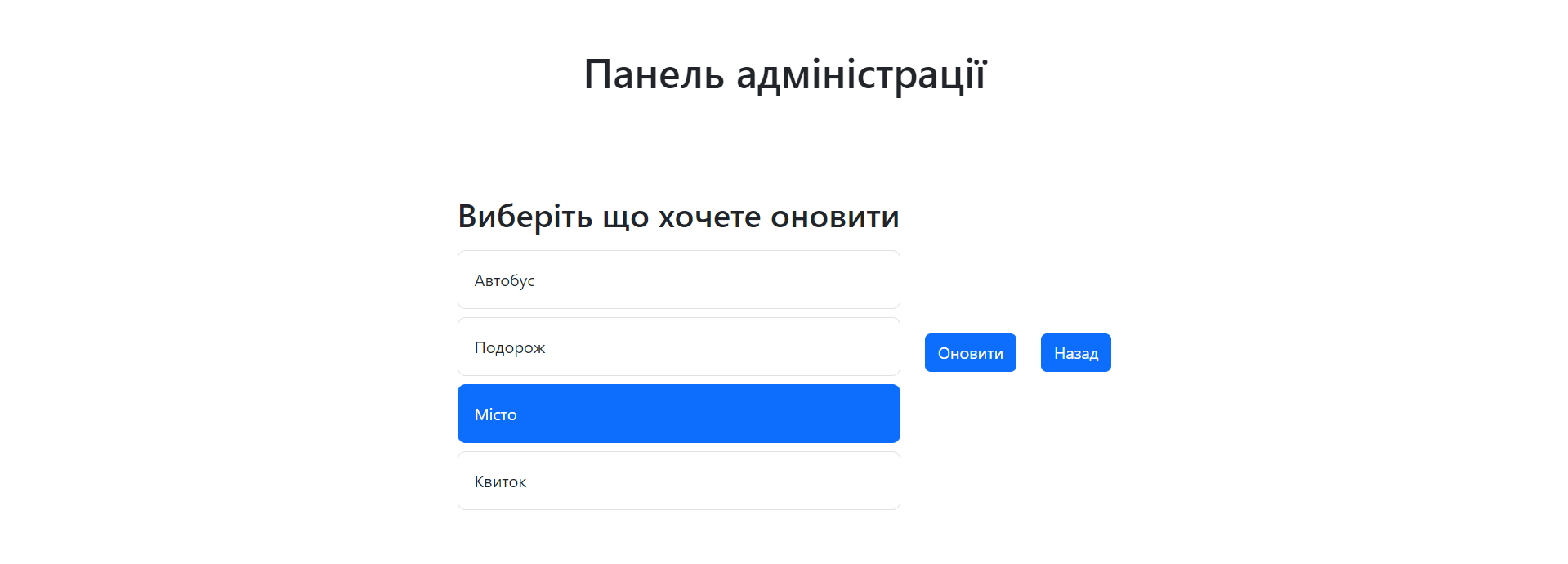
На сторінці “Додатиадміна”ми можемо помітитиполя для надання імені нового адміністратора, email, телефону та паролю . У випадку помилки показується аналогове вікно з текстом “Адміністратора не додано”.



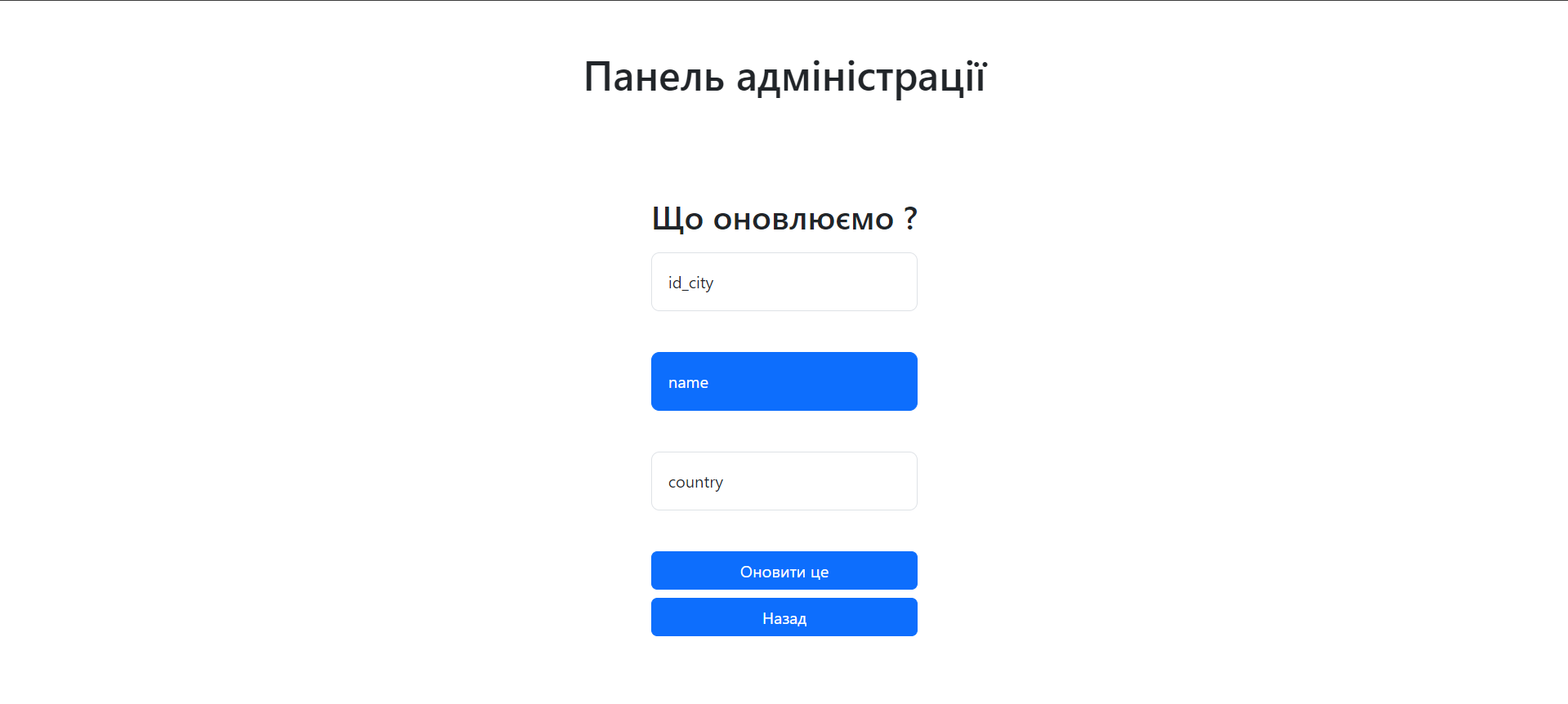
**Рис.3.28.** Сторінка яка реалізує додавання нового адміністратора сайту

**Операції Update :**

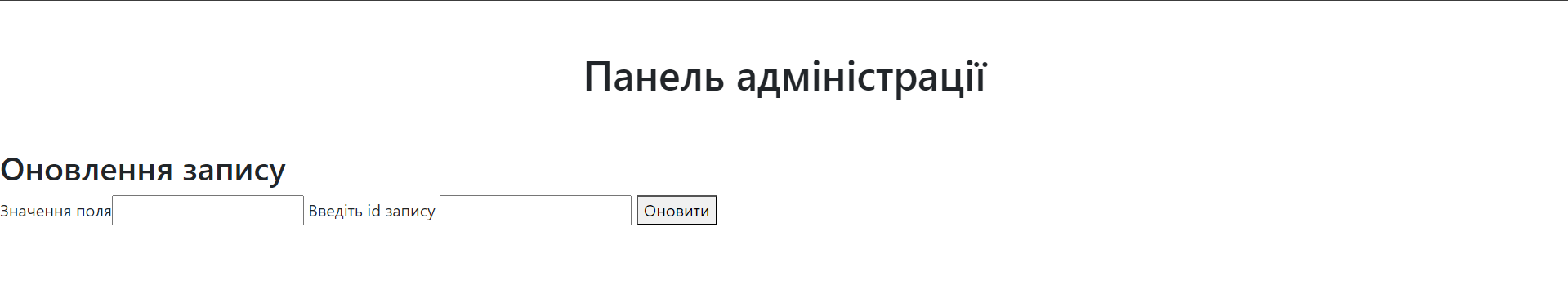
Після кліку по кнопці “Оновити записи” нас посилає на аналогову сторінку з вибором що ми хочемо оновити.



**Рис.3.29.** Сторінка для вибору оновлення певного об’єкту



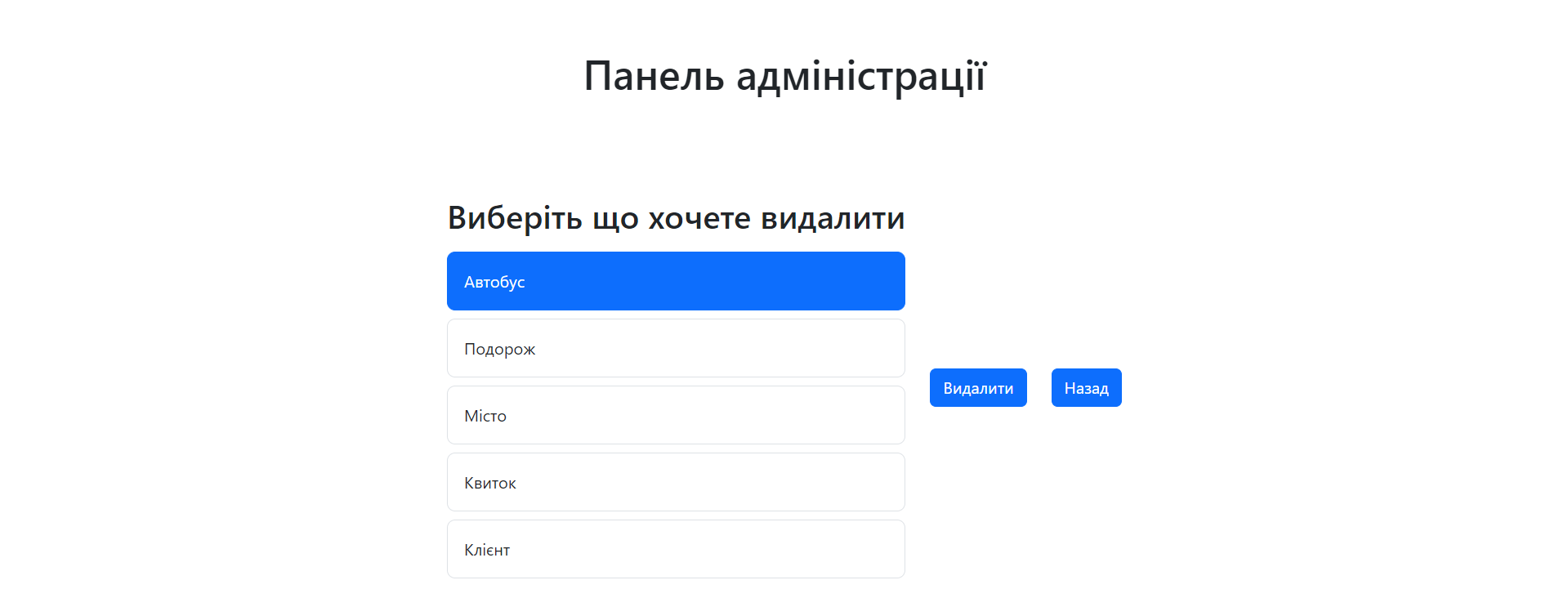
**Рис.3.30.** Сторінка для вибору оновлення певного поля об’єкту



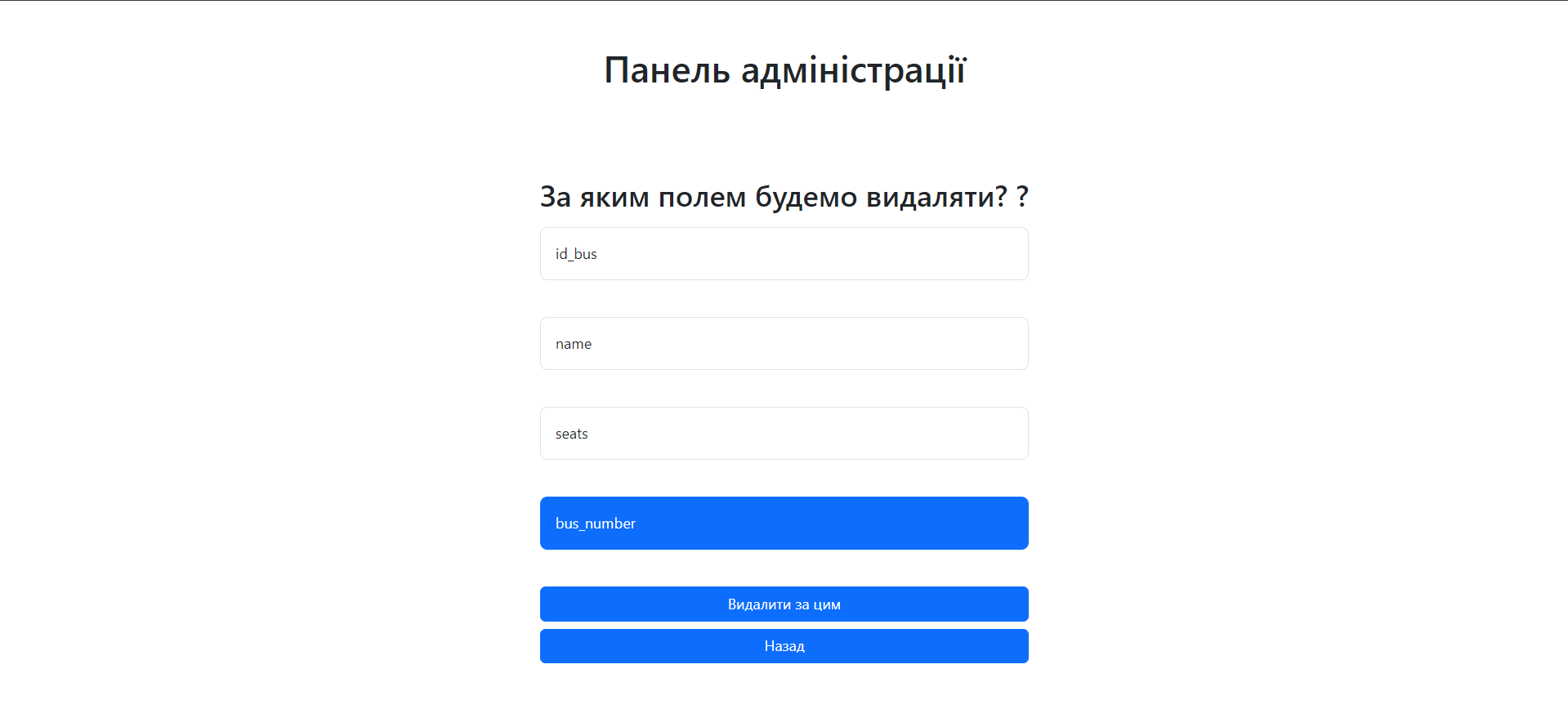
**Рис.3.31.** Інтерфейс для оновлення певного поля об’єкту якого ми вибрали раніше

Після вводу данних надсилається запит оновлення в БД таблиці, поля, відповідного id . Коли виникає проблема сайт відображає повідомлення з повідомленням “Не оновлено”.

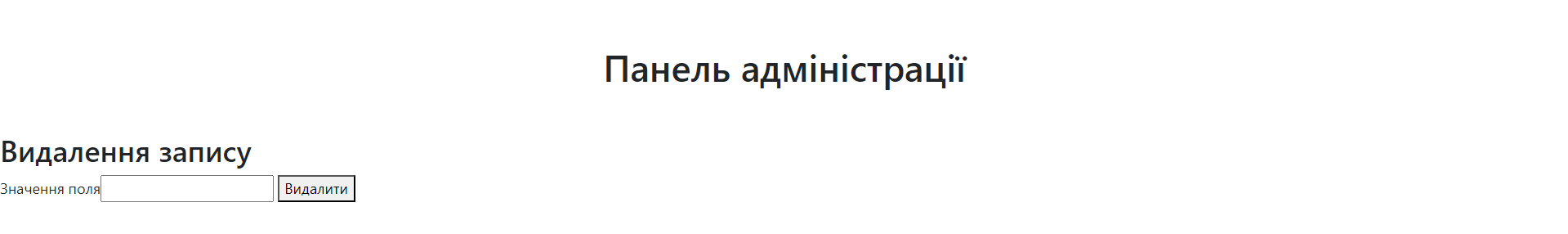
**Операції Delete :**



**Рис.3.32.** Вибір таблиці об’єкта для видалення

****

**Рис.3.33.** Вибір за якийм полем ми будемо видаляти

****

**Рис.3.34.** Вибір за яким значенням поля ми будемо видаляти запис

**Операції Read :**

Операції читання вже реалізовані як на деяких сторінках адмін панелі так і в користувацькому інтерфейсі.

## 3.2. Реалізації операцій обробки даних в БД

Функція перевірки користувача:

def CreateUser(password, item, value):

if db\_context.get\_user(password, item, value) == "Пароль не співпадає":

return "Пароль не співпадає"

elif db\_context.get\_user(password, item, value):

user = db\_context.get\_user(password, item, value)

AddToDict(user)

return True

else:

return False

Ця функція призначена для перевірки користувача в базі даних за допомогою пароля та іншої інформації (наприклад, електронної пошти або номера телефону). Ось її опис:

* Перевіряється, чи співпадає пароль користувача з паролем, що введений.
* Якщо пароль не співпадає, повертається відповідне повідомлення.
* Якщо користувач існує в базі даних і пароль співпадає, користувача додається до сесії або контексту додатка, і повертається True.
* Якщо користувача не знайдено в базі даних, повертається False.

Функція отримання користувача з бази даних:

def get\_user(self, password, item, value):

query = f"SELECT \* FROM clients WHERE {str(item)} = ?"

self.cursor.execute(query, (str(value),))

row = self.cursor.fetchone()

if row:

user = Clients()

attributes = list(user.\_\_dict\_\_.keys())

for attribute, row\_value in zip(attributes, row):

setattr(user, attribute, row\_value if row\_value is not None else "")

print(str(user.id\_client) + " " + user.name + " " + user.phone + " " + user.password + " " + user.email)

if user.password == password:

print("User authorized")

return user

else:

print("Incorrect password")

return "Пароль не співпадає"

print("None account")

return None

Ця функція виконує запит до бази даних для отримання користувача з вказаним паролем та іншим атрибутом (електронна пошта або номер телефону). Ось її опис:

* Формується SQL-запит з використанням формат строки f-string. Запит виглядає як SELECT \* FROM clients WHERE {str(item)} = ?, де {str(item)} замінюється на ім'я поля (електронна пошта або телефон), передане як параметр item.
* Виконується SQL-запит з параметром value, який визначається у вигляді кортежу (str(value),).
* Отриманий результат перевіряється за допомогою методу fetchone(). Якщо рядок існує (тобто користувач з такою електронною поштою або номером телефону існує у базі даних), то відбувається наступне:
* Створюється об'єкт класу Clients.
* Дані з рядка записуються у відповідні атрибути об'єкта.
* Порівнюється пароль, який був переданий функції, з паролем користувача. Якщо вони співпадають, функція повертає об'єкт користувача user.
* Якщо пароль не співпадає, функція повертає рядок "Пароль не співпадає".
* Якщо рядок не знайдено у базі даних, функція повертає None.

Функція додавання об'єкта в базу даних:

def add\_item(self, obj):

db\_context = DbContext()

db\_context.select\_last\_index(obj)

array\_items = list(obj.\_\_dict\_\_.keys())

non\_id\_items = [item for item in array\_items if not item.startswith("id")]

values = ', '.join(

[f"'{obj.\_\_dict\_\_[item]}'" if obj.\_\_dict\_\_[item] is not None else "NULL" for item in non\_id\_items])

query = f"INSERT INTO {obj.\_\_class\_\_.\_\_name\_\_} ({', '.join(non\_id\_items)}) VALUES ({values})"

print(query)

try:

self.cursor.execute(query)

self.conn.commit()

print(f"Item added successfully to {obj.\_\_class\_\_.\_\_name\_\_} table.")

return True

except Exception as e:

print(f"Error adding item to {obj.\_\_class\_\_.\_\_name\_\_} table: {str(e)}")

return False

Ця функція призначена для додавання об'єкта в базу даних, вона додає всі об’єкти які за своєю структурою схожі з таблицями бд. Ось опис цієї функції:

* Викликається метод select\_last\_index, який встановлює останній індекс для нового об'єкта, якщо це потрібно.
* Створюється список non\_id\_items, який містить атрибути об'єкта, за винятком ідентифікатора.
* Формується рядок values, який представляє значення для вставки в SQL-запит. Для кожного атрибута об'єкта перевіряється, чи він не є None, і якщо ні, то він включається у рядок значень.
* Створюється SQL-запит для вставки нового рядка в таблицю бази даних.
* Виконується SQL-запит за допомогою курсора бази даних.
* Зафіксовується зміна у базі даних за допомогою команди commit.
* Виводиться повідомлення про успішне додавання об'єкта до таблиці.
* Якщо сталася помилка під час додавання об'єкта до бази даних, виводиться повідомлення про помилку, і функція повертає False.

Взагалом ця функція додає всі можливі об’єкти які наявні в БД,але розглянемо реєстрацію нового користувача

Додавання нового користувача в базу даних:

if db\_context.add\_item(user):

AddToDict(user)

Ці рядки коду викликають функцію add\_item з екземпляром DbContext для додавання нового користувача user до бази даних. Якщо додавання пройшло успішно (функція add\_item повернула True), то користувач також додається до сесії за допомогою функції AddToDict.

Функція вибірки одного запису по id:

def get\_item\_by\_id(self, other\_class\_instance, item\_id):

array\_items = list(other\_class\_instance.\_\_dict\_\_.keys())

table = other\_class\_instance.\_\_class\_\_.\_\_name\_\_

for item in array\_items:

if item.startswith("id"):

id\_class\_name = item

break

query = f"SELECT {', '.join(array\_items)} FROM {table} WHERE {id\_class\_name} = ?"

self.cursor.execute(query, (item\_id,))

row = self.cursor.fetchone()

instance = None

if row:

instance = other\_class\_instance.\_\_class\_\_()

for idx, item in enumerate(array\_items):

value = row[idx]

setattr(instance, item, value if value is not None else None)

return instance

Ця функція get\_item\_by\_id призначена для отримання об'єкта з бази даних за його унікальним ідентифікатором (ID). Ось її пояснення:

* Отримання назви таблиці та списку атрибутів: Функція отримує назву таблиці та список атрибутів об'єкта, для якого вона шукає об'єкт, через параметр other\_class\_instance.
* Визначення поля ID: Потім визначається поле, яке є ідентифікатором, тобто має назву, що починається з "id".
* Створення SQL-запиту: Формується SQL-запит для отримання об'єкта з бази даних за його ID.
* Виконання запиту та створення об'єкта: Виконується запит до бази даних, і якщо об'єкт знайдено, то на його основі створюється новий об'єкт класу other\_class\_instance.\_\_class\_\_, а дані з бази даних присвоюються відповідним атрибутам цього об'єкта.
* Повернення об'єкта: Повертається знайдений об'єкт з бази даних або None, якщо об'єкт не знайдено.

Ця функція дуже корисна для отримання даних з бази даних за їхнім унікальним ідентифікатором, що дозволяє ефективно працювати з об'єктами в системі. Вона забезпечує можливість отримувати об'єкти з бази даних для подальшої обробки чи відображення.

Ця функція також використовується напротязі всього життя проекту наприклад :

journey = Timetable()

journey = db\_context.get\_item\_by\_id(journey, id\_journey)

Ця строка коду бере з бази данних один запис маршруту і перетворює його в об’єкт Timetable.

Функція вибору останнього індексу з певної таблиці:

def select\_last\_index(self,obj):

array\_items = list(obj.\_\_dict\_\_.keys())

for item in array\_items:

if item.startswith("id"):

id\_class\_name = item

break

query = f"SELECT MAX({str(id\_class\_name)}) FROM {obj.\_\_class\_\_.\_\_name\_\_}"

try:

self.cursor.execute(query)

result = self.cursor.fetchone()[0]

return result if result is not None else 0

except Exception as e:

print(f"Error getting last index from {obj.\_\_class\_\_.\_\_name\_\_}: {str(e)}")

return 0

Ця функція select\_last\_index призначена для отримання останнього індексу (або ID) з таблиці бази даних. Ось її пояснення:

* Отримання списку атрибутів об'єкта: Функція отримує список атрибутів об'єкта через його метод \_\_dict\_\_.
* Визначення поля ID: Потім визначається поле, яке є ідентифікатором, тобто має назву, що починається з "id".
* Створення SQL-запиту: Формується SQL-запит для отримання максимального значення ID з таблиці.
* Виконання запиту та отримання результату: Виконується запит до бази даних, і якщо результат отримано успішно, то повертається максимальне значення ID. Якщо результату немає або сталася помилка, повертається 0.
* Повернення результату: Повертається максимальне значення ID або 0 як значення за замовчуванням.

Ця функція дозволяє отримати останній індекс з таблиці, що може бути корисно для створення нових записів в базі даних з унікальним ідентифікатором. Вона забезпечує можливість отримувати доступ до останнього доступного індексу для подальших операцій з даними.

Отримання елемента шукаючи за іншим значенням відмінним від id:

def get\_item\_by\_other\_value(self, other\_class\_instance, item, value):

array\_items = list(other\_class\_instance.\_\_dict\_\_.keys())

table = other\_class\_instance.\_\_class\_\_.\_\_name\_\_

query = f"SELECT \* FROM {table} WHERE {item} = ?"

self.cursor.execute(query, (value,))

row = self.cursor.fetchone()

if row:

new\_instance = type(other\_class\_instance)()

for idx, attribute in enumerate(array\_items):

setattr(new\_instance, attribute, row[idx] if row[idx] is not None else "")

return new\_instance

return None

Ця функція get\_item\_by\_other\_value призначена для отримання об'єкта з бази даних за значенням певного поля, яке не є ідентифікатором. Ось її пояснення:

* Отримання назви таблиці та списку атрибутів: Функція отримує назву таблиці та список атрибутів об'єкта, для якого вона шукає об'єкт, через параметр other\_class\_instance.
* Створення SQL-запиту: Формується SQL-запит для отримання об'єкта з бази даних за значенням певного поля.
* Виконання запиту та створення об'єкта: Виконується запит до бази даних, і якщо об'єкт знайдено, то на його основі створюється новий об'єкт того ж класу, що і вихідний, а дані з бази даних присвоюються відповідним атрибутам цього об'єкта.
* Повернення об'єкта: Повертається знайдений об'єкт з бази даних або None, якщо об'єкт не знайдено.

Ця функція дозволяє отримувати об'єкти з бази даних за значенням певного поля, що може бути корисно для пошуку об'єктів за різними критеріями. Вона забезпечує можливість отримувати доступ до об'єктів за їхніми властивостями для подальшого використання в програмі.

Отримання списку елементів шукаючи за значенням відмінним від id:

def get\_items\_by\_other\_value(self, other\_class\_instance, item, value):

try:

array\_items = list(other\_class\_instance.\_\_dict\_\_.keys())

table = other\_class\_instance.\_\_class\_\_.\_\_name\_\_

query = f"SELECT \* FROM {table} WHERE {item} = ?"

self.cursor.execute(query, (value,))

rows = self.cursor.fetchall()

result = []

if rows:

for row in rows:

instance = other\_class\_instance.\_\_class\_\_()

for idx, item\_name in enumerate(array\_items):

item\_value = row[idx]

print(item\_value)

setattr(instance, item\_name, item\_value if item\_value is not None else "")

result.append(instance)

print("Items retrieved successfully")

return result

else:

return None

except Exception as e:

print(f"Error executing SQL query: {e}")

return None

Ця функція get\_items\_by\_other\_value призначена для отримання списку об'єктів з бази даних за значенням певного поля, яке не є ідентифікатором. Ось її пояснення:

* Отримання назви таблиці та списку атрибутів: Функція отримує назву таблиці та список атрибутів об'єкта, для якого вона шукає об'єкти, через параметр other\_class\_instance.
* Створення SQL-запиту та виконання: Формується SQL-запит для отримання об'єктів з бази даних за значенням певного поля. Запит виконується із значенням параметра, яке передається через value.
* Обробка результату запиту: Після виконання запиту отримані рядки з бази даних обробляються. Якщо рядки знайдено, для кожного рядка створюється новий об'єкт того ж класу, що і вихідний, а дані з бази даних присвоюються відповідним атрибутам цього об'єкта. Створені об'єкти додаються до списку result.
* Повернення списку об'єктів: Повертається список знайдених об'єктів з бази даних або None, якщо об'єкти не знайдено.

Ця функція дозволяє отримувати списки об'єктів з бази даних за значенням певного поля, що може бути корисно для виконання різноманітних завдань пошуку та фільтрації даних. Вона забезпечує можливість отримувати доступ до групи об'єктів за їхніми властивостями для подальшого використання в програмі.

Функція оновлення запису :

def update\_item(self, id, item, value, obj):

nametable = obj.\_\_class\_\_.\_\_name\_\_

if nametable=="Buses":

name\_id='id\_bus'

elif nametable=="Clients":

name\_id='id\_client'

elif nametable=="Timetable":

name\_id='id\_journey'

elif nametable == "Cities":

name\_id = 'id\_city'

elif nametable == "Non\_autorized\_users":

name\_id = 'id\_user'

elif nametable == "Non\_autorized\_users":

name\_id = 'id\_user'

elif nametable == "Orders":

name\_id = 'id\_order'

elif nametable == "Tickets":

name\_id = 'id\_ticket'

else:

print("Не підтримуємий клас")

print("err on DBContext at 158 line")

return False

query = f"UPDATE {nametable} SET {item}=? WHERE {name\_id}=?"

values = (value, id)

try:

with self.conn:

with self.cursor.execute(query, values):

print(f"Update successful for {item} with value {value} for record with {name\_id}={id} in table {nametable}.")

return True

except Exception as e:

print(f"Error updating record: {e}")

return False

Ця функція update\_item призначена для оновлення запису в базі даних. Ось її пояснення:

* Визначення назви поля для ідентифікатора: Перевіряється назва класу obj, для якого виконується оновлення, і відповідно до цього визначається назва поля ідентифікатора.
* Створення SQL-запиту: Формується SQL-запит для оновлення запису в таблиці. Запит виконується для вказаного поля item та значення value, де умовою є ідентифікатор рядка, що дорівнює id.
* Виконання запиту: Виконується SQL-запит для оновлення запису в таблиці з використанням значень item та id.
* Повернення результату оновлення: Повертається True, якщо оновлення пройшло успішно, або False, якщо сталася помилка під час виконання запиту.

Ця функція дозволяє оновлювати дані у вказаному полі для запису в таблиці з використанням ідентифікатора цього запису. Вона може бути використана для зміни певних даних в базі даних, наприклад, оновлення даних про автобус, клієнта або розклад подорожі.

Функція видалення записів з БД:

def delete\_item(self, obj, item, value):

nametable = obj.\_\_class\_\_.\_\_name\_\_

query = f"DELETE FROM {nametable} WHERE {item} = ?"

values = (value,)

try:

with self.conn:

with self.cursor.execute(query, values):

print(f"Delete successful from {nametable} where {item} = {value}.")

return True

except Exception as e:

print(f"Error deleting record: {e}")

return False

Ця функція delete\_item призначена для видалення записів з бази даних за певною умовою. Ось її пояснення:

* Визначення назви таблиці: Визначається назва таблиці, з якої будуть видалятися записи. Для цього використовується назва класу obj.
* Створення SQL-запиту: Формується SQL-запит на видалення записів з таблиці, де умовою для видалення є співпадіння значення поля item з вказаним значенням value.
* Виконання запиту: Виконується SQL-запит для видалення записів з таблиці, які відповідають умові.
* Повернення результату видалення: Повертається True, якщо видалення пройшло успішно, або False, якщо сталася помилка під час виконання запиту.

Ця функція дозволяє видаляти записи з таблиці бази даних за певною умовою, яка вказується через параметри item та value. Наприклад, вона може бути використана для видалення певного клієнта або автобуса з бази даних за його ідентифікатором або іншими атрибутами.

Клас DbContext реалізує доступ до бази даних та виконує операції з нею за допомогою бібліотеки pyodbc, яка надає засоби для з'єднання з базами даних за допомогою ODBC. Ось коротка інформація про бібліотеку :

* pyodbc - це бібліотека Python для доступу до баз даних за допомогою інтерфейсу ODBC (Open Database Connectivity).
* Вона дозволяє взаємодіяти з різними типами баз даних, такими як SQL Server, MySQL, PostgreSQL тощо, використовуючи стандартний інтерфейс ODBC.

Висновок до третього розділу **:**

У цьому розділі було описано розробку функцій взаємодії з базою даних у веб-додатку за допомогою класу DbContext та бібліотеки pyodbc. pyodbc - це бібліотека для з'єднання з базами даних за допомогою ODBC (Open Database Connectivity) у Python. Вона дозволяє взаємодіяти з різними системами управління базами даних (СУБД), такими як SQL Server, MySQL, PostgreSQL та іншими.

Клас DbContext містить реалізацію функцій для виконання операцій з базою даних, таких як додавання, оновлення, видалення та отримання даних. Ці функції дозволяють взаємодіяти з об'єктами бази даних, такими як користувачі, автобуси, розклади, міста тощо, шляхом виконання SQL-запитів.

У результаті було розроблено повний набір функцій для взаємодії користувача з базою даних у веб-додатку. Ці функції дозволяють додавати, оновлювати, видаляти та отримувати записи з бази даних, що забезпечує повний функціонал для управління даними у додатку.

Також був розроблений комфортний користувацький інтерфейс веб-сайту для, що включає сторінки для реєстрації, авторизації, купівлі квитків, перегляду всіх маршрутів та їх фільтрування, а також перегляд збережених квитків.

# РОЗДІЛ 4. АДМІНІСТРУВАННЯ БАЗ ДАНИХ

## 4.1. Розробка заходів захисту інформації в БД

Отже, розглянемо категорії користувачів веб-сайту автовокзалу “OnAvtovokzal”: адміністратор, авторизовані користувачі, неавторизовані користувачі. Адміністратор має найбільше доступних опцій та функцій, що необхідно для підтримання роботи сайту. Вони можуть редагувати інформацію, більш детально про роль надано ролі " Адміністратор " наведено в табл. 8

**Таблиця 4.1.**

Матриця доступу для ролі "Адміністратор"

|  |  |
| --- | --- |
|  | Адміністратор |
| Створити новий маршрут | + |
| Редагувати інформацію про маршрути | + |
| Створити новий автобус | + |
| Редагувати інформацію про маршрути | + |
| Створити новий маршрут | + |
| Редагувати інформацію про маршрути | + |
| Створити нове місто | + |
| Редагувати інформацію про міста | + |
| Додати адміністратора | + |
| Видалити користувача | + |
| Видалити квиток | + |
| Видалити маршрут | + |
| Видалити автобус | + |
| Видалити місто | + |
| Купити квиток | + |
| Переглянути інформацію про акаунт | + |
| Переглянути інформацію про квитки які звязані з цим клієнтом | + |
| Переглянути розписання маршрутів | + |
| Купівля квитка на сторінці “Розписання всіх автобусів” | + |
| Переглянути квиток за кодом | + |

Основну масу користувачів системи складають клієнти, які реєструються на сайті та мають свої особисті кабінети, тому, що вони можуть переглядати всі квитки в кабінеті, які вони купували на відміну від не авторизованих користувачів. Більш детально про роль "Авторизований користувач" наведено в табл. 9

**Таблиця 4.2.**

Матриця доступу для ролі "Авторизований користувач"

|  |  |
| --- | --- |
|  | Авторизований користувач |
| Створити новий маршрут | - |
| Редагувати інформацію про маршрути | - |
| Створити новий автобус | - |
| Редагувати інформацію про маршрути | - |
| Створити новий маршрут | - |
| Редагувати інформацію про маршрути | - |
| Створити нове місто | - |
| Редагувати інформацію про міста | - |
| Додати адміністратора | - |
| Видалити користувача | - |
| Видалити квиток | - |
| Видалити маршрут | - |
| Видалити автобус | - |
| Видалити місто | - |
| Купити квиток | + |
| Переглянути інформацію про акаунт | + |
| Переглянути інформацію про квитки які звязані з цим клієнтом | + |
| Переглянути розписання маршрутів | + |
| Купівля квитка на сторінці “Розписання всіх автобусів” | + |
| Переглянути квиток за кодом | + |

Також розглянемо функціонал для не авторизованого користувача:

**Таблиця 4.3.**

Матриця доступу для ролі "Не авторизований користувач"

|  |  |
| --- | --- |
|  | Авторизований користувач |
| Створити новий маршрут | - |
| Редагувати інформацію про маршрути | - |
| Створити новий автобус | - |
| Редагувати інформацію про маршрути | - |
| Створити новий маршрут | - |
| Редагувати інформацію про маршрути | - |
| Створити нове місто | - |
| Редагувати інформацію про міста | - |
| Додати адміністратора | - |
| Видалити користувача | - |
| Видалити квиток | - |
| Видалити маршрут | - |
| Видалити автобус | - |
| Видалити місто | - |
| Купити квиток | + |
| Переглянути інформацію про акаунт | - |
| Переглянути інформацію про квитки які звязані з цим клієнтом | - |
| Переглянути розписання маршрутів | + |
| Купівля квитка на сторінці “Розписання всіх автобусів” | - |
| Переглянути квиток за кодом | + |

## 4.2. Налаштування параметрів роботи Sql Server.

Для того, щоб підключити Sql Server в проект потрібно імпортувати та встановити бібліотеку podbc за допомогою команди pip install.

Бібліотека pyodbc - це інструмент для забезпечення зв'язку між Python та базами даних, особливо з реляційними СУБД, такими як Microsoft SQL Server. Вона дозволяє виконувати SQL-запити до бази даних, отримувати результати та взаємодіяти з ними безпосередньо з Python-коду. Ця бібліотека надає доступ до драйверів ODBC (Open Database Connectivity), які є стандартом для з'єднання з різними типами баз даних. Завдяки цьому, вона підтримує роботу з багатьма типами баз даних, включаючи Microsoft SQL Server, MySQL, PostgreSQL та інші.

За допомогою pyodbc можна створювати з'єднання з базою даних, виконувати SQL-запити, вставляти, оновлювати та видаляти дані, а також отримувати результати запитів для подальшого аналізу та використання в програмах на Python. Вона дозволяє розробникам легко і ефективно взаємодіяти з базами даних у своїх проектах.

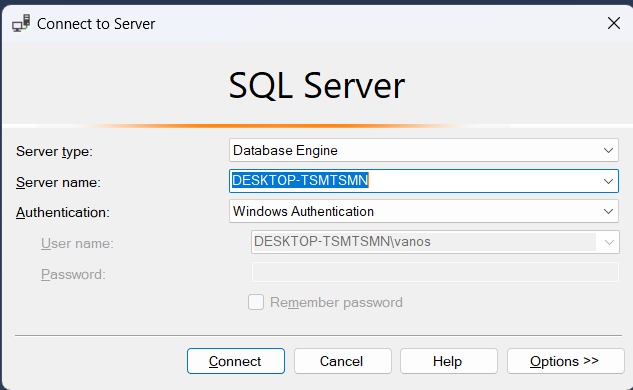
Функціонал цієї бібліотеки було реалізовано в класі DbContext. Цей клас містить реалізацію функціоналу для роботи з базою даних, такий як виконання SQL-запитів, вставка, оновлення та видалення даних, а також отримання результатів для подальшого використання на сервері Flask. Для початку ми під’єднуємо клас до БД за допомогою строки під’єднання, та створрюємо курсор який буде виконувати пошук по БД :

def \_\_init\_\_(self):

self.conn = pyodbc.connect("DRIVER={ODBC Driver 17 for SQL Server};SERVER=DESKTOP-TSMTSMN;DATABASE=avtovokzal;Trusted\_Connection=yes")

self.cursor = self.conn.cursor()

Для адміністрування БД за допомогою СУБД потрібно зайти в програму SSMS та встановлюємо server name у відповідності з назвою комп’ютера, або відповідно з данними які надав хост.



**Рис.4.1.** Підключення до серверу бази данних в СУБД

**Висновки до четвертого розділу**

У ході виконання даного розділу проведено налаштування прав доступу користувачів до бази даних. Було розглянуто різні категорії користувачів веб-сайту автовокзалу "OnAvtovokzal" та встановлено їхні права доступу до функціоналу сайту.

Адміністратор має найбільші можливості управління сайтом, включаючи створення та редагування інформації про маршрути, автобуси, міста, а також можливість додавати та видаляти користувачів та квитки. Авторизований користувач має обмежені можливості порівняно з адміністратором, але все ж може купувати квитки, переглядати інформацію про свій акаунт та квитки, а також переглядати розписання маршрутів. Неавторизований користувач може тільки переглядати розписання маршрутів та купувати квитки.

Для роботи з базою даних SQL Server у проекті використовується бібліотека pyodbc, яка забезпечує зв'язок між Python та базою даних. Бібліотека надає можливість виконувати SQL-запити, вставляти, оновлювати та видаляти дані, а також отримувати результати для подальшого використання. Для адміністрування БД використовується SQL Server Management Studio (SSMS), де налаштовуються параметри роботи сервера баз даних.

У результаті проведених дій забезпечено ефективне управління та захист інформації в БД для забезпечення безпеки та функціональності веб-сайту "OnAvtovokzal".

**ВИСНОВКИ**

У результаті виконання даного проекту було спроектовано та розроблено сайт новинного блогу.

Під час виконання курсового проекту було проаналізовано головні, можливості та особливості різних БД, що допомогло визначити основні плюси та мінуси їх та вибрати ту, за допомогою якої можна якісно та швидко виконати дану роботу.

Після того, як було вибрано та пояснено причину та логіку вибору БД було виконано проектування схеми бази. У результаті виконання цього етапу, було розроблено коректну загальну схему бази, що дозволило отримати розуміння того, що саме має відбуватись в коді та як це потрібно реалізувати.

Після проектування загальної структури бази, було розпочато проектування інтерфейсу додатку. Під час цього етапу було реалізовано підключення до БД, створені різні методи для коректної взаємодії з базою. Наприклад, було написано методи для вставок даних, для оновлен даних, для їх видалення й пошуку.

При виконання загальної структури коду, було виконано налаштування деяких додаткових сервісів для коректної роботи додатку, а саме для збереження медіа-файлів та можливості валідного входу та реєстрації користувача в додатку.

Було проведено тестування у ручному режимі, після виявлення помилок, їх було виправлено та пере-протестовано додаток, що дає можливість визнати, що додаток є стабільним. Таким чином була розроблена програма з використанням мови PHP, SQL, HTML, CSS, JavaScript використаний Bootstrap та phpMyAdmin.

# Список використаних джерел

1. SQL Server Management Studio (SSMS) - TutorialsTeacher.com, [Електронний ресурс] - 2024, Режим доступу: <https://www.tutorialsteacher.com/sqlserver/sql-server-management-studio>
2. SQL Server Management Studio (SSMS) - learn.microsoft.com , [Електронний ресурс] - 2023, Режим доступу: <https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16>s
3. Python Tutorial - W3Schools [Електронний ресурс] -

Режим доступу до ресурсу : <https://www.w3schools.com/python/>

1. Learn Python - Free Interactive Python Tutorial - learnpython.org [електронний ресурс] - 2006. Режим доступу: <https://www.learnpython.org/>
2. Learn Python - Free Interactive Python Tutorial - learnpython.org [електронний ресурс] – 2022. Режим доступу: <https://docs.python.org/3/tutorial/index.html>
3. [Welcome to Flask — Flask Documentation (3.0.x) - flask.palletsprojects.com](https://flask.palletsprojects.com/), Режим доступу : <https://flask.palletsprojects.com/en/3.0.x/>
4. Створення SQl запитів [електронний ресурс] - 2016. Режим доступу: [https://www.politerm.com/zuludoc/spatial\_query\_sql.htm#\_sql\_overview](https://www.politerm.com/zuludoc/spatial_query_sql.htm%23_sql_overview)
5. Форум з питаннями та відповідями [Електронний ресурс] – Режим доступу до ресурсу:

<https://stackoverflow.com/> .

1. SQL [електронний ресурс] - 20 16. Режим доступу: <https://uk.wikibooks.org/wiki/SQL>
2. AJAX Response from Asynchronous JavaScript Call [електронний ресурс] - 2017. Режим доступу: [https://stackify.com/return-ajax-responseasynchronous-javascript-call/](https://stackify.com/return-ajax-responseasynchronous-javascript-call/%20) .

# ДОДАТКИ

Додаток А

Технічне завдання

**1.** Загальне положення

**1.1.** Найменування програмного засобу

Повне найменування програмної системи: “База даних каси автовокзалу”

(надалі "програма"). Коротка назва програмної системи - “OnAvtovokzal”

**1.2.** Призначення розробки та область застосування

Програмна система “База даних автовокзалу” призначена для збору,

обробки та відображення даних про рейси автобусів, квитки та користувачів. В

програмі передбачено використання механізму пошуку рейсів додатку.

Система може бути впроваджена у роботу перевезень з кількістю рейсів

до 100 штук та базою користувачів до 100000 записів.

**1.3.** Мета

Програмна система “База даних автовокзалу” дозволи ть отримати

інформацію про наявні перевезення, купівлю квитків на доступні рейси.

Підвищити прибуток перевезень за рахунок продажу в інтернеті.

**1.4.** Найменування розробника та замовника.

Розробник даного продукту - студент групи ВТ-22-1 Середа Іван Романович (надалі "розробник").

Замовник програмного продукту – кафедра інженерія програмного

забезпечення Житомирського державного технологічного університету в

межах виконання курсової з дисципліни «Бази даних» Чижмотря Олексій

Володимирович, Сугоняк Інна Іванівна (надалі замовник).

**2.** Підстава для розробки

**2.1.** Документ на підставі якого ведеться розробка

Робота ведеться на підставі навчального плану за напрямом

121 «Інженерія програмного забезпечення».

**3.** Вимоги до програми

**3.1.** Вимоги до функціональних характеристик.

**3.1.1.** Загальні вимоги

Веб-додаток має забезпечувати:

* можливість дистанційної роботи з робочих станцій локальної та глобальної мережі підприємства;
* постійний доступ користувачів веб-додатку;
* організацію управління сайтом;
* можливість доступ до бази даних;
* постійний доступ користувачів до БД;
* оптимальне збереження даних (за обсягом та структурою);
* аутентифікацію користувачів та захист інформації від несанкціонованого доступу;
* надійне збереження даних та можливість відновлення даних у випадку
* непередбачуваних збоїв системи;
* можливість модернізації системи через зміну функціональних потреб користувача

**3.1.2.** Склад виконуваних функцій

Розробити базу даних, що використовується у веб-додатку, який

підтримує виконання наступних операцій:

**1.** Реєстрація користувачів

**2.** Вхід в обліковий запис

**3.** Перегляд, редагування, створення, видалення користувачів, квитків,

рейсів, сервісів, автобусів та компаній.

**4.** Редагування прав доступу кожного користувача.

**5.** Купівля квитків на рейси та скасування їх за потреби.

**6.** Резервне копіювання через веб-додаток.

**7.** Відображення всієї необхідної інформації про рейс.

**3.1.3.** Організація введення та виведення даних

Для ефективного введення та виведення даних система буде використовувати інформацію про рейси, квитки, автобуси та компанії. Вхідні та вихідні дані повинні відповідати інформаційній структурі операцій, які виконуються.

Введення оперативних даних буде здійснюватися через діалогові екранні форми, що побудовані на базі візуальних компонентів. Введення даних буде здійснюватися за допомогою відповідних форм, доступних у додатку.

Це забезпечить зручний та ефективний процес введення та виведення необхідної інформації, що відповідає вимогам операцій та сприятиме оптимізації робочих процесів.

**3.1.4.** Часові характеристики і розмір пам'яті, необхідної для роботи

програми.

Час реакції програми на взаємодію користувача (введення даних) не перевищує 0,25 секунди.

Час виконання команд меню не перевищує 1 секунди.

Відображення масивів даних за запитами не більше 3 хвилин.

Доступність бази даних - 90%, забезпечена цілодобово.

Операції з'єднання з базою даних займають не більше 1 хвилини.

Обсяг оперативної пам'яті, необхідний для роботи програми, складає не менше 1 ГБ.

Дисковий простір, необхідний для збереження програми та файлів даних, не перевищує 300 МБ для робочої станції та 20 ГБ.

Розмір інсталяційного пакету програми, який включає в себе базу даних, не перевищує 100 МБ.

**3.2.** Вимоги до надійності.

**3.2.1.** Вимоги до надійного функціонування

Програма повинна безперебійно функціонувати при стабільній роботі ПК. Доступність бази даних має бути на рівні 90% при одночасному доступі 30 користувачів. У випадку апаратних збоїв відновлення нормальної роботи програми повинно бути наступним чином:

а) При апаратних збоях сервера - виконується перезавантаження операційної системи сервера та запуск сервера бази даних. Якщо передбачено, може бути використано резервний сервер або технології RAID для збереження даних.

б) При апаратних збоях робочої станції - виконується перезавантаження операційної системи ПК та запуск виконуваного файлу програми.

в) База даних повинна бути відновлена до найближчого несуперечного стану, для чого слід передбачити точки відновлення.

У випадку програмних збоїв:

а) Система повинна забезпечувати можливість відновлення даних та фіксацію та "відкат" транзакцій.

б) В системі має бути реалізована коректна обробка виняткових ситуацій.

**3.2.2.**Контроль вхідної і вихідної інформації

Для контролю коректності вхідної інформації та захисту від помилок оператора система передбачає наступні заходи:

* Перевірка відповідності даних доменам інформаційних атрибутів, щоб упевнитися у відповідності введених даних до очікуваних типів та форматів.
* Використання механізму автоматичного заповнення та вибору з переліку для пов'язаних даних, що дозволяє операторам зручно та швидко вводити дані без помилок.
* Захист від помилок оператора, таких як залипання клавіш або випадкового введення символів, шляхом використання відповідних інтерфейсних рішень.

Некоректні дії оператора супроводжуються повідомленнями про помилку та блокуванням операцій оновлення даних. При цьому система запобігає загальному блокуванню для забезпечення неперервності роботи.

Для вихідної інформації система передбачає:

* Відсутність блокування даних через багато користувальницький доступ, забезпечуючи при цьому необхідний рівень доступності даних для всіх користувачів.
* Постійне оновлення даних у відображених на екрані звітних формах для актуальності та точності інформації.
* Повідомлення про використання режиму монопольного доступу супервайзерами бази даних, щоб уникнути конфліктів та забезпечити безперебійну роботу системи.

**3.2.3.** Час відновлення після відмови.

Час відновлення після відмови, не пов'язаною з роботою програми,

повинен складатися із: часу перезапуску операційної системи; часу запуску

сервера БД (підключення до сервера ) запуску виконуваного файлу, часу

повторного введення або зчитування даних.

**3.3.** Умови експлуатації і збереження

Програма призначена для використання у багатокористувацькому середовищі. Регламентовані операції проводяться відповідно до наступного графіка:

* Створення резервних копій даних: проводиться один раз на місяць з метою забезпечення безпеки даних.
* Оновлення системи ідентифікації та аутентифікації користувачів: виконується один раз на місяць, а також за необхідності з урахуванням поточних вимог безпеки. Збереження архівних копій здійснюється на визначених носіях, а резервних копій - на визначеному диску без перезапису.
* Знищення копій проводиться відповідно до регламенту з метою забезпечення конфіденційності та безпеки даних.

**3.4.** Вимоги до інформаційної і програмної сумісності.

**3.4.1.** Вимоги до інформаційних структур на вході і виході

Формат відображення даних має дозволяти виконання операцій з

даними в браузері.

**3.4.2.** Вимоги до методів рішення і мов програмування

Вибір методів рішення здійснюється розробникам без узгодження з

замовником. СУБД обирається у відповідності до характеристик визначених в

п.3. З замовником погоджується вибір варіанту за вартісною ознакою.

**3.4.3.** Вимоги до системи програмних засобів.

Вимоги до програмного забезпечення сервера: ОС – серверна версія ОС

Windows не нижче Windows 2000 Server, IIS (або інший сервер), СУБД

визначається встановлюється та налаштовується розробником (ліцензування

СУБД виконується замовником).

Вимоги до програмного забезпечення робочої станції:

ОС - Windows не нижче Windows XP.

**3.5.** Вимоги до складу і параметрів технічних засобів.

Вимоги до складу технічних засобів:

* Сервер повинен мати базову конфігурацію з підтримкою технології RAID, можливість для замінного зберігання даних та можливість підключення до локальної мережі.
* Робоча станція має бути оснащена процесором Intel або AMD зі стандартним набором комплектації, а також засобами для підключення до локальної мережі. Мінімальні технічні характеристики включають процесор Pentium III з тактовою частотою 750 МГц, оперативну пам'ять об'ємом 128 Мбайт та дисків з пам'яттю 20 Гбайт.

Вимоги до програмної документації:

Програмна документація повинна містити наступні дані:

* "Інструкція по інсталяції програми", яка складається з опису процесу встановлення програми, переліку кроків та послідовності їх виконання, опису встановлених програмних компонентів та їх робочих режимів після встановлення. Ця інструкція зазвичай розміщується у четвертому розділі пояснювальної записки до курсової роботи.
* "Керівництво користувача", яке включає опис процесу запуску програми, основних режимів роботи, опис екранних форм, перелік виняткових ситуацій та рекомендації щодо їх вирішення, а також порядок виконання завдань у системі. Ця інструкція зазвичай розміщується у третьому розділі пояснювальної записки до курсової роботи.
* "Керівництво адміністратора баз даних", яке включає опис структури таблиць бази даних та правил доступу до них, інструкції щодо обслуговування бази даних (наприклад, архівація, створення резервних копій), налаштування сервера та клієнтських додатків. Ця інструкція зазвичай розміщується у другому, третьому та четвертому розділах пояснювальної записки до курсової роботи відповідно до плану.

Додаток Б

Основний клас

**dbcontext.db**

import pyodbc

from ClassLib.clients import Clients

class DbContext:

def \_\_init\_\_(self):

self.conn = pyodbc.connect("DRIVER={ODBC Driver 17 for SQL Server};SERVER=DESKTOP-TSMTSMN;DATABASE=avtovokzal;Trusted\_Connection=yes")

self.cursor = self.conn.cursor()

def get\_items(self, other\_class\_instance, newquery=""):

array\_items = list(other\_class\_instance.\_\_dict\_\_.keys())

table = other\_class\_instance.\_\_class\_\_.\_\_name\_\_

query = f"SELECT {', '.join(array\_items)} FROM {table}"

if newquery != "":

query = newquery

print(query)

try:

self.cursor.execute(query)

rows = self.cursor.fetchall()

result = []

if rows:

for row in rows:

instance = other\_class\_instance.\_\_class\_\_()

for idx, item in enumerate(array\_items):

value = row[idx]

setattr(instance, item, value if value is not None else "")

result.append(instance)

print("Item Get succes")

return result

else:

return None

except Exception as e:

print(f"Error executing SQL query: {e}")

return None

def get\_item\_by\_id(self, other\_class\_instance, item\_id):

array\_items = list(other\_class\_instance.\_\_dict\_\_.keys())

table = other\_class\_instance.\_\_class\_\_.\_\_name\_\_

for item in array\_items:

if item.startswith("id"):

id\_class\_name = item

break

query = f"SELECT {', '.join(array\_items)} FROM {table} WHERE {id\_class\_name} = ?"

self.cursor.execute(query, (item\_id,))

row = self.cursor.fetchone()

instance = None

if row:

instance = other\_class\_instance.\_\_class\_\_()

for idx, item in enumerate(array\_items):

value = row[idx]

setattr(instance, item, value if value is not None else None)

return instance

def add\_item(self, obj):

db\_context = DbContext()

db\_context.select\_last\_index(obj)

array\_items = list(obj.\_\_dict\_\_.keys())

non\_id\_items = [item for item in array\_items if not item.startswith("id")]

values = ', '.join(

[f"'{obj.\_\_dict\_\_[item]}'" if obj.\_\_dict\_\_[item] != None else "NULL" for item in non\_id\_items])

query = f"INSERT INTO {obj.\_\_class\_\_.\_\_name\_\_} ({', '.join(non\_id\_items)}) VALUES ({values})"

print(query)

try:

self.cursor.execute(query)

self.conn.commit()

print(f"Item added successfully to {obj.\_\_class\_\_.\_\_name\_\_} table.")

return True

except Exception as e:

print(f"Error adding item to {obj.\_\_class\_\_.\_\_name\_\_} table: {str(e)}")

return False

def select\_last\_index(self,obj):

array\_items = list(obj.\_\_dict\_\_.keys())

for item in array\_items:

if item.startswith("id"):

id\_class\_name = item

break

query = f"SELECT MAX({str(id\_class\_name)}) FROM {obj.\_\_class\_\_.\_\_name\_\_}"

try:

self.cursor.execute(query)

result = self.cursor.fetchone()[0]

return result if result is not None else 0

except Exception as e:

print(f"Error getting last index from {obj.\_\_class\_\_.\_\_name\_\_}: {str(e)}")

return 0

def get\_item\_by\_other\_value(self, other\_class\_instance, item, value):

array\_items = list(other\_class\_instance.\_\_dict\_\_.keys())

table = other\_class\_instance.\_\_class\_\_.\_\_name\_\_

query = f"SELECT \* FROM {table} WHERE {item} = ?"

self.cursor.execute(query, (value,))

row = self.cursor.fetchone()

if row:

new\_instance = type(other\_class\_instance)()

for idx, attribute in enumerate(array\_items):

setattr(new\_instance, attribute, row[idx] if row[idx] is not None else "")

return new\_instance

return None

def get\_items\_by\_other\_value(self, other\_class\_instance, item, value):

try:

array\_items = list(other\_class\_instance.\_\_dict\_\_.keys())

table = other\_class\_instance.\_\_class\_\_.\_\_name\_\_

query = f"SELECT \* FROM {table} WHERE {item} = ?"

self.cursor.execute(query, (value,))

rows = self.cursor.fetchall()

result = []

if rows:

for row in rows:

instance = other\_class\_instance.\_\_class\_\_()

for idx, item\_name in enumerate(array\_items):

item\_value = row[idx]

print(item\_value)

setattr(instance, item\_name, item\_value if item\_value is not None else "")

result.append(instance)

print("Items retrieved successfully")

return result

else:

return None

except Exception as e:

print(f"Error executing SQL query: {e}")

return None

def get\_user(self, password, item, value):

query = f"SELECT \* FROM clients WHERE {str(item)} = ?"

self.cursor.execute(query, (str(value),))

row = self.cursor.fetchone()

if row:

user = Clients()

attributes = list(user.\_\_dict\_\_.keys())

for attribute, row\_value in zip(attributes, row):

setattr(user, attribute, row\_value if row\_value is not None else "")

print(str(user.id\_client)+" "+user.name+" "+user.phone+" "+user.password+" "+user.email)

if user.password == password:

print("User authorized")

return user

else :

print("incorect password")

return "Пароль не співпадає"

print("None account")

return None

def update\_item(self, id, item, value, obj):

nametable = obj.\_\_class\_\_.\_\_name\_\_

if nametable=="Buses":

name\_id='id\_bus'

elif nametable=="Clients":

name\_id='id\_client'

elif nametable=="Timetable":

name\_id='id\_journey'

elif nametable == "Cities":

name\_id = 'id\_city'

elif nametable == "Non\_autorized\_users":

name\_id = 'id\_user'

elif nametable == "Non\_autorized\_users":

name\_id = 'id\_user'

elif nametable == "Orders":

name\_id = 'id\_order'

elif nametable == "Tickets":

name\_id = 'id\_ticket'

else:

print("Не підтримуємий клас")

print("err on DBContext at 158 line")

return False

query = f"UPDATE {nametable} SET {item}=? WHERE {name\_id}=?"

values = (value, id)

try:

with self.conn:

with self.cursor.execute(query, values):

print(f"Update successful for {item} with value {value} for record with {name\_id}={id} in table {nametable}.")

return True

except Exception as e:

print(f"Error updating record: {e}")

return False

def delete\_item(self, obj, item, value):

nametable = obj.\_\_class\_\_.\_\_name\_\_

query = f"DELETE FROM {nametable} WHERE {item} = ?"

values = (value,)

try:

with self.conn:

with self.cursor.execute(query, values):

print(f"Delete successful from {nametable} where {item} = {value}.")

return True

except Exception as e:

print(f"Error deleting record: {e}")

return False

Вихідний код проекту

**Program.py**

import os

from flask import Flask, render\_template,request,session,redirect,abort,url\_for

from dbcontext import DbContext

from ClassLib.buses import Buses

from ClassLib.cities import Cities

from ClassLib.clients import Clients

from ClassLib.tickets import Tickets

from ClassLib.timetable import Timetable

from ClassLib.orders import Orders

from ClassLib.journeystable import JourneysTable

from ClassLib.non\_autorized\_users import Non\_autorized\_users

from ClassLib.cardticket import CardTicket

from datetime import datetime

import random

import time

app = Flask(\_\_name\_\_)

app.secret\_key = os.urandom(24)

db\_context = DbContext()

@app.route('/buyticket/', methods=['GET','POST'])

def buyticket():

if request.method == "POST":

return redirect("/home")

return redirect("/formbuyticket")

@app.route('/formbuyticket', methods=['GET','POST'])

def formbuyticket():

if request.method=="GET":

if (request.args.get('citystart') and request.args.get('cityfinish')

and request.args.get('starttime') and request.args.get('finishtime')):

query = (

f"SELECT timetable.id\_journey,buses.name AS bus\_name, start\_city.name AS start\_city, finish\_city.name AS finish\_city, timetable.time\_start, timetable.time\_finish ,timetable.cost FROM timetable "

f"JOIN cities AS start\_city ON timetable.city\_start\_id = start\_city.id\_city "

f"JOIN cities AS finish\_city ON timetable.city\_finish\_id = finish\_city.id\_city "

f"JOIN buses ON timetable.bus\_id = buses.id\_bus ")

if request.args.get('citystart') == "Відправляємся з":

pass

elif request.args.get('citystart'):

citystart = request.args.get('citystart')

query += f" WHERE start\_city.name LIKE '{citystart}'"

if request.args.get('cityfinish') == "Їдем в":

pass

elif request.args.get('cityfinish'):

cityfinish = request.args.get('cityfinish')

query += f" AND finish\_city.name LIKE '{cityfinish}'"

if request.args.get('starttime') and request.args.get('finishtime'):

starttime = request.args.get('starttime')

finishtime = request.args.get('finishtime')

starttime = datetime.strptime(starttime, '%Y-%m-%dT%H:%M')

finishtime = datetime.strptime(finishtime, '%Y-%m-%dT%H:%M')

query += f" AND timetable.time\_start BETWEEN '{starttime}' AND '{finishtime}'"

table = db\_context.get\_items(JourneysTable(), newquery=query)

if table is not None:

if request.args.get('name') and request.args.get('phone'):

non\_auth\_user=Non\_autorized\_users()

non\_auth\_user.name=request.args.get('name')

non\_auth\_user.phone=request.args.get('phone')

return render\_template("tablejourneys.html", user=GetFromDict(), table=table,non\_auth\_user=non\_auth\_user)

else:

return render\_template("tablejourneys.html", user=GetFromDict(), table=table,non\_auth\_user=None)

else:

cities = db\_context.get\_items(Cities())

return render\_template("formbuyticket.html", user=GetFromDict(), cities=cities)

cities = db\_context.get\_items(Cities())

return render\_template("formbuyticket.html", user=GetFromDict(), cities=cities)

@app.route("/tablejourneys",methods=['GET','POST'])

def tablejourneys():

pass

@app.route('/selectplace/', methods=['POST'])

def selectplace():

if request.method == "POST":

print('POST')

non\_auth\_user = None

if request.form.get('phone') and request.form.get('name'):

non\_auth\_user = Non\_autorized\_users()

non\_auth\_user.phone = request.form.get('phone')

non\_auth\_user.name = request.form.get('name')

id\_journey = request.form.get('id\_journey')

bus\_name = request.form.get('bus\_name').strip()

start\_city = request.form.get('start\_city')

finish\_city = request.form.get('finish\_city')

print("name" + bus\_name)

selected\_bus = Buses() # Змінилася назва змінної bus на selected\_bus

selected\_bus = db\_context.get\_item\_by\_other\_value(selected\_bus, 'name', bus\_name)

print("seat"+str(selected\_bus.seats))

print("name"+str(selected\_bus.name))

print("bus\_number"+str(selected\_bus.bus\_number))

journey = Timetable()

journey = db\_context.get\_item\_by\_id(journey, id\_journey)

tickets = None

query = f"SELECT \* FROM tickets WHERE tickets.journey\_id Like {id\_journey}"

if db\_context.get\_items(Tickets(), newquery=query) is not None:

tickets = db\_context.get\_items(Tickets(), newquery=query)

countstickets = 0

greyseats = None

if tickets is not None:

countstickets = len(tickets)

greyseats = []

for ticket in tickets:

for seat in range(selected\_bus.seats): # Використовується selected\_bus замість bus

if ticket.seat == seat:

greyseats.append(seat)

if non\_auth\_user is not None:

return render\_template("selectplace.html", user=GetFromDict(), non\_auth\_user=non\_auth\_user,

bus=selected\_bus,

journey=journey,

start\_city=start\_city, finish\_city=finish\_city, tickets=tickets,

countstickets=countstickets, greyseats=greyseats)

else:

return render\_template("selectplace.html", user=GetFromDict(), non\_auth\_user=None, bus=selected\_bus,

journey=journey,

start\_city=start\_city, finish\_city=finish\_city, tickets=tickets,

countstickets=countstickets, greyseats=greyseats)

return redirect('/formbuyticket')

@app.route('/pay/', methods=['POST', 'GET'])

def pay():

if request.method == "POST":

id\_journey = request.form.get('id\_journey')

seat = request.form.get('seat')

non\_auth\_user = None

if request.form.get('phone') and request.form.get('name'):

non\_auth\_user = Non\_autorized\_users()

non\_auth\_user.phone = request.form.get('phone')

non\_auth\_user.name = request.form.get('name')

else:

user = GetFromDict()

journey = Timetable()

journey = db\_context.get\_item\_by\_id(journey, id\_journey)

bus = Buses()

bus = db\_context.get\_item\_by\_id(bus, journey.bus\_id)

start\_city = Cities()

finish\_city = Cities()

start\_city = db\_context.get\_item\_by\_id(start\_city, journey.city\_start\_id)

finish\_city = db\_context.get\_item\_by\_id(finish\_city, journey.city\_finish\_id)

return render\_template('pay.html',user=GetFromDict(), journey=journey, bus=bus, seat=seat, start\_city=start\_city,

finish\_city=finish\_city, non\_auth\_user=non\_auth\_user)

else:

return redirect('/formbuyticket')

@app.route('/succesorder/', methods=['POST'])

def succesorder():

if request.method == 'POST':

non\_auth\_user = None

if request.form.get('phone') and request.form.get('name'):

non\_auth\_user = Non\_autorized\_users()

non\_auth\_user.phone = request.form.get('phone')

non\_auth\_user.name = request.form.get('name')

if db\_context.add\_item(non\_auth\_user):

non\_auth\_user.id\_user=db\_context.select\_last\_index(non\_auth\_user)

id\_journey = request.form.get('id\_journey')

seat = request.form.get('seat')

journey = Timetable()

journey = db\_context.get\_item\_by\_id(journey, id\_journey)

#створення нового ордеру

neworder=Orders()

#заповнення данних

if non\_auth\_user is None:

user = GetFromDict()

user=db\_context.get\_item\_by\_other\_value(user,'email',user.email)

print(user.id\_client)

neworder.client\_id = user.id\_client

else:

neworder.non\_authorized\_users\_id = non\_auth\_user.id\_user

neworder.date\_buying = neworder.date\_buying = datetime.strptime(str(datetime.utcnow().replace(microsecond=0)), '%Y-%m-%d %H:%M:%S')

neworder.cost = journey.cost

db\_context.add\_item(neworder)

#дістаєм id останнього ордеру

indexLastOrder = db\_context.select\_last\_index(neworder)

#створення нового екзепляру ордера та діставання данних з БД

order = Orders()

order = db\_context.get\_item\_by\_id(order, indexLastOrder)

#перевірка ордерів між собою

if (order.client\_id == neworder.client\_id and \

order.non\_authorized\_users\_id == neworder.non\_authorized\_users\_id and \

str(order.date\_buying) == str(neworder.date\_buying) and \

order.cost == neworder.cost):

#зміна статусу оплати

whetherpaid = db\_context.update\_item(order.id\_order, 'whether\_paid', 1, order)

if whetherpaid:

# заповнення данних квитка

newticket = Tickets()

newticket.journey\_id = journey.id\_journey

newticket.bus\_id = journey.bus\_id

newticket.order\_id = order.id\_order

newticket.client\_id = order.client\_id

newticket.date\_buying = datetime.strptime(str(datetime.utcnow().replace(microsecond=0)), '%Y-%m-%d %H:%M:%S')

newticket.code=GenerateRandomNumber()

if non\_auth\_user is None:

newticket.client\_id = user.id\_client

else:

newticket.non\_autorized\_users\_id = non\_auth\_user.id\_user

newticket.seat=seat

iffer = db\_context.add\_item(newticket)

if iffer:

bus = Buses()

bus = db\_context.get\_item\_by\_id(bus, journey.bus\_id)

start\_city = Cities()

finish\_city = Cities()

start\_city = db\_context.get\_item\_by\_id(start\_city, journey.city\_start\_id)

finish\_city = db\_context.get\_item\_by\_id(finish\_city, journey.city\_finish\_id)

if iffer is None:

return render\_template('succesorder.html', title="Succes", user=GetFromDict(), non\_auth\_user=non\_auth\_user,

start\_city=start\_city,finish\_city=finish\_city,journey=journey, bus=bus, seat=seat, ticket=newticket)

else:

return render\_template('succesorder.html', title="Succes",user=GetFromDict(), non\_auth\_user=non\_auth\_user,

start\_city=start\_city,finish\_city=finish\_city,journey=journey, bus=bus, seat=seat, ticket=newticket)

else:

return render\_template('succesorder.html', title=None,user=GetFromDict())

else:

pass

else:

return redirect('/formbuyticket')

return render\_template('succesorder.html', title=None)

@app.route("/getticket",methods=["POST"])

def getticket():

if request.form.get('ticket\_id'):

ticket\_id=request.form.get('ticket\_id')

ticket=Tickets()

if db\_context.get\_item\_by\_other\_value(ticket,'code',ticket\_id):

ticket=db\_context.get\_item\_by\_other\_value(ticket,'code',ticket\_id)

journey=Timetable()

journey=db\_context.get\_item\_by\_id(journey,ticket.journey\_id)

start\_city = Cities()

finish\_city = Cities()

start\_city = db\_context.get\_item\_by\_id(start\_city, journey.city\_start\_id)

finish\_city = db\_context.get\_item\_by\_id(finish\_city, journey.city\_finish\_id)

bus=Buses()

bus=db\_context.get\_item\_by\_id(bus,ticket.bus\_id)

if ticket.client\_id is not None:

user=Clients()

user=db\_context.get\_item\_by\_id(user,ticket.client\_id)

nonuser=None

else:

nonuser = Non\_autorized\_users()

nonuser = db\_context.get\_item\_by\_id(nonuser, ticket.non\_autorized\_users\_id)

user=None

return render\_template('succesorder.html', title="Succes", user=user, non\_auth\_user=nonuser,

start\_city=start\_city, finish\_city=finish\_city, journey=journey, bus=bus, seat=ticket.seat,

ticket=ticket)

else:

return redirect('/home')

@app.route('/journeys', methods=['GET','POST'])

def journeys():

for req in request.args:

print(request.args[req])

print('DA')

try:

if ((request.args.get('item\_sort') and request.args.get('direction') and request.args.get('citystart')) or

(request.args.get('citystart') and request.args.get('cityfinish')) or

(request.args.get('item\_sort') and request.args.get('direction') and request.args.get('citystart')and request.args.get('cityfinish'))):

item\_sort=""

direction=""

if request.args.get('item\_sort'):

item\_sort = request.args.get('item\_sort')

if request.args.get('direction'):

direction = request.args.get('direction')

query = (

f"SELECT timetable.id\_journey,buses.name AS bus\_name, start\_city.name AS start\_city, finish\_city.name AS finish\_city, timetable.time\_start, timetable.time\_finish,timetable.cost FROM timetable "

f"JOIN cities AS start\_city ON timetable.city\_start\_id = start\_city.id\_city "

f"JOIN cities AS finish\_city ON timetable.city\_finish\_id = finish\_city.id\_city "

f"JOIN buses ON timetable.bus\_id = buses.id\_bus ")

if request.args.get('citystart') == "Відправляємся з":

pass

elif request.args.get('citystart'):

citystart = request.args.get('citystart')

query += f" WHERE start\_city.name LIKE '{citystart}'"

if request.args.get('cityfinish')=="Їдем в":

pass

elif request.args.get('cityfinish'):

cityfinish = request.args.get('cityfinish')

query += f" AND finish\_city.name LIKE '{cityfinish}'"

if request.args.get('starttime') and request.args.get('finishtime'):

starttime = request.args.get('starttime')

finishtime = request.args.get('finishtime')

starttime = datetime.strptime(starttime, '%Y-%m-%dT%H:%M')

finishtime = datetime.strptime(finishtime, '%Y-%m-%dT%H:%M')

query += f" AND timetable.time\_start BETWEEN '{starttime}' AND '{finishtime}'"

if item\_sort == "Назва автобусу":

query += " ORDER BY bus\_name"

elif item\_sort == "За іменем міста відправлення":

query += " ORDER BY start\_city"

elif item\_sort == "За іменем міста прибуття":

query += " ORDER BY finish\_city"

elif item\_sort == "За часом відправлення":

query += " ORDER BY time\_start"

elif item\_sort == "За часом прибуття":

query += " ORDER BY time\_finish"

else:

return render\_template('Error', 500)

if direction == "По спаданню":

query += " DESC"

elif direction == "По зростанню":

query += " ASC"

table = db\_context.get\_items(JourneysTable(), newquery=query)

cities = db\_context.get\_items(Cities())

return render\_template("journeys.html", user=GetFromDict(), table=table, cities=cities)

except Exception as e:

print(f"An error occurred: {str(e)}")

buses = db\_context.get\_items(Buses())

journeys = db\_context.get\_items(Timetable())

cities = db\_context.get\_items(Cities())

return render\_template("journeys.html", user=GetFromDict(), buses=buses, journeys=journeys, cities=cities,

table=None)

@app.route('/')

def firstroute():

return render\_template("home.html",user=GetFromDict())

@app.route('/home')

def home():

return render\_template("home.html", user=GetFromDict())

@app.route('/About')

def about():

return render\_template("about.html",user=GetFromDict())

@app.route('/loginselect')

def loginselect():

return render\_template("loginselect.html",user=None)

@app.route('/login/<data>',methods=['GET','POST'])

def login(data):

if request.method == "POST":

item=str(data)

password= request.form['floatingPassword']

if data == "email":

value = request.form['floatingInputEmail']

if CreateUser(password, item, value)=="Пароль не співпадає":

return render\_template('login.html', data=data, user=None, info="Не вірний пароль")

elif CreateUser(password, item, value):

return redirect("/home")

else:

return render\_template('login.html', data=data, user=None, info="")

elif data == "phone":

value = request.form['floatingInputPhone']

if CreateUser(password,item, value):

return redirect("/home")

else:

return render\_template('login.html', data=data, user=None,info="")

if request.method == "GET":

return render\_template('login.html', data=data, user=None,info="")

else:

return redirect("/home")

@app.route('/register' ,methods=['POST','GET'])

def register():

if request.method =="POST":

if GetFromDict() is None:

name = str(request.form['floatingInputName'])

email=str(request.form['floatingInputEmail'])

phone=str(request.form['floatingInputPhone'])

password=str(request.form['floatingInputPassTwo'])

role\_client='user'

user=Clients()

if name:

user.name = name

if email:

user.email = email

if phone:

user.phone = phone

if password:

user.password = password

if role\_client:

user.role\_client=role\_client

if(db\_context.add\_item(user)):

AddToDict(user)

return redirect("/home")

else:

return render\_template("register.html", user=None,info="Account not register")

else:

return redirect("/home")

if request.method == "GET":

return render\_template("register.html",user=None,info=None)

@app.route('/accountview')

def accountview():

user = GetFromDict()

if user:

return render\_template("accountview.html", user=user)

else:

return redirect("/home")

@app.route('/exit')

def exit():

DeleteFromDict()

return redirect('/home');

@app.route('/usertickets')

def usertickets():

user = GetFromDict()

if user:

#потрібно дізнстись id user

user = db\_context.get\_item\_by\_other\_value(user, 'email', user.email)

tickets=None

if db\_context.get\_items\_by\_other\_value(Tickets(), 'client\_id', user.id\_client):

tickets = db\_context.get\_items\_by\_other\_value(Tickets(), 'client\_id', user.id\_client)

if tickets is not None:

for ticket in tickets:

print(ticket.id\_ticket)

oldtickets = []

recenttickets = []

if tickets is not None:

for ticket in tickets:

card=CardTicket()

journeyy = Timetable()

journeyy = db\_context.get\_item\_by\_id(journeyy, ticket.journey\_id)

journeys = []

journeys.append(journeyy)

card.id\_ticket=ticket.id\_ticket

card.code=ticket.code

card.cost=journeyy.cost

time\_start = str(journeyy.time\_start)

time\_finish = str(journeyy.time\_finish)

time\_start\_obj = datetime.strptime(time\_start, "%Y-%m-%d %H:%M:%S")

time\_finish\_obj = datetime.strptime(time\_finish, "%Y-%m-%d %H:%M:%S")

formatted\_time\_start = time\_start\_obj.strftime("%d.%m.%Y %H:%M:%S")

formatted\_time\_finish = time\_finish\_obj.strftime("%d.%m.%Y %H:%M:%S")

str\_time = f"{formatted\_time\_start} - {formatted\_time\_finish}"

card.time = str\_time

for journey in journeys:

city\_start=Cities()

city\_finish=Cities()

city\_start=db\_context.get\_item\_by\_id(city\_start,journey.city\_start\_id)

city\_finish=db\_context.get\_item\_by\_id(city\_finish,journey.city\_finish\_id)

str\_city=str(city\_start.name)+" - " + str(city\_finish.name)

card.name=str\_city

current\_time = datetime.utcnow().replace(microsecond=0)

if ticket.journey\_id==journey.id\_journey:

if journey.time\_start < current\_time:

oldtickets.append(card)

else:

recenttickets.append(card)

return render\_template("usertickets.html", user=user, oldtickets=oldtickets, recenttickets=recenttickets)

else:

return render\_template("usertickets.html", user=user, oldtickets=None, recenttickets=None)

else:

return redirect("/home")

@app.route('/addbus', methods=['POST','GET'])

def addbus():

if request.method=="GET":

if GetFromDict() is not None:

return render\_template("addbus.html", info=None)

else:

return redirect("/home")

if request.method == "POST":

bus=Buses()

bus.name=str(request.form['nameBus'])

bus.seats=str(request.form['seats'])

bus.bus\_number=str(request.form['busNumber'])

if db\_context.add\_item(bus):

return render\_template("addbus.html", info=None)

else:

return render\_template("addbus.html", info="Не додано")

@app.route('/addcity', methods=['POST','GET'])

def addcity():

if request.method=="GET":

if GetFromDict() is not None:

return render\_template("addcity.html", info=None)

else:

return redirect("/home")

if request.method=="POST":

city=Cities()

city.name=str(request.form['nameCity'])

city.country=str(request.form['country'])

if db\_context.add\_item(city):

return render\_template("addcity.html", info=None)

else:

return render\_template("addcity.html", info="Не додано")

@app.route('/addjourney', methods=['POST', 'GET'])

def addjourney():

if request.method == "GET":

if GetFromDict() is not None:

cities = db\_context.get\_items(Cities())

buses = db\_context.get\_items(Buses())

return render\_template("addjourney.html", info=None, buses=buses, cities=cities)

else:

return redirect("/home")

if request.method == "POST":

journey = Timetable()

journey.bus\_id = int(request.form['idbus'])

journey.city\_start\_id = int(request.form['idcitystart'])

journey.city\_finish\_id = int(request.form['idcityfinish'])

journey.cost = float(request.form['cost'])

journey.time\_start = datetime.strptime(request.form['timestart'], '%Y-%m-%dT%H:%M')

journey.time\_finish = datetime.strptime(request.form['timefinish'], '%Y-%m-%dT%H:%M')

journey.seats\_occupied=int(request.form['seats\_occupied'])

cities = db\_context.get\_items(Cities())

buses = db\_context.get\_items(Buses())

if db\_context.add\_item(journey):

return render\_template("addjourney.html", info=None, buses=buses, cities=cities)

else:

return render\_template("addjourney.html", info="Не додано", buses=buses, cities=cities)

@app.route('/addadmin', methods=['POST', 'GET'])

def addadmin():

if request.method == "GET":

if GetFromDict() is not None:

return render\_template("addadmin.html", info=None)

else:

return redirect("/home")

if request.method == "POST":

user = Clients()

user.name=str(request.form['name'])

user.email=str(request.form['email'])

user.phone=str(request.form['phone'])

user.password=str(request.form['password'])

user.role\_client='admin'

if db\_context.add\_item(user):

return render\_template("addadmin.html", info=None)

else:

return render\_template("addadmin.html", info="Не додано")

@app.route('/delete', methods=['POST','GET'])

def delete():

if request.method == "GET":

if GetFromDict():

if GetFromDict().role\_client == "admin":

return render\_template("delete.html")

else:

return redirect("/home")

if request.method == "POST":

selected\_radio = request.form.get('listGroupCheckableRadios')

if selected\_radio == "bus":

bus = Buses()

fieldlist = GetClassFields(bus)

return render\_template("deletewhatrecord.html", fieldlist=fieldlist, clas=bus)

elif selected\_radio == "journey":

journey = Timetable()

fieldlist = GetClassFields(journey)

return render\_template("deletewhatrecord.html", fieldlist=fieldlist, clas=journey)

elif selected\_radio == "city":

city = Cities()

fieldlist = GetClassFields(city)

return render\_template("deletewhatrecord.html", fieldlist=fieldlist, clas=city)

elif selected\_radio == "ticket":

tick = Tickets()

fieldlist = GetClassFields(tick)

return render\_template("deletewhatrecord.html", fieldlist=fieldlist, clas=tick)

elif selected\_radio == "client":

client = Clients()

fieldlist = GetClassFields(client)

return render\_template("deletewhatrecord.html", fieldlist=fieldlist, clas=client)

return redirect("/adminpanel")

@app.route('/deletepost/<selectedfield>/<clas>', methods=['POST', 'GET'])

def deletepost(selectedfield, clas):

if request.method == "GET":

if GetFromDict() is not None:

if GetFromDict().role\_client == "admin":

ss = selectedfield

return render\_template("deletepost.html", selectedfield=ss, clas=clas, info=None)

else:

return redirect("/home")

if request.method == "POST":

item = selectedfield

value = request.form['value']

type=request.form['type']

try:

obj\_class = globals()[clas]

obj = obj\_class()

except KeyError:

return render\_template("deletepost.html", selectedfield=selectedfield, clas=clas,

info="Не вдалося знайти клас")

try:

if type == "datetime":

return render\_template("deletepost.html", selectedfield=selectedfield, clas=clas,

info="За цим полем не можна видалити")

except ValueError:

return render\_template("deletepost.html", selectedfield=selectedfield, clas=clas,

info="Помилка конвертації у формат дати/часу")

try:

if type == "int":

value = int(value)

except ValueError:

return render\_template("deletepost.html", selectedfield=selectedfield, clas=clas,

info="Помилка конвертації у формат int")

if db\_context.delete\_item( obj, item, value):

return render\_template("adminpanel.html")

else:

return render\_template("deletepost.html", selectedfield=selectedfield, clas=clas, info="Не видалено")

@app.route('/update', methods=['POST','GET'])

def update():

if request.method=="GET":

if GetFromDict():

if GetFromDict().role\_client == "admin":

return render\_template("update.html")

else:

return redirect("/home")

if request.method=="POST":

selected\_radio = request.form.get('listGroupCheckableRadios')

if selected\_radio == "bus":

bus=Buses()

fieldlist=GetClassFields(bus)

return render\_template("updatewhatrecord.html", fieldlist=fieldlist,clas=bus)

elif selected\_radio == "journey":

journey=Timetable()

fieldlist = GetClassFields(journey)

return render\_template("updatewhatrecord.html", fieldlist=fieldlist,clas=journey)

elif selected\_radio == "city":

city=Cities()

fieldlist = GetClassFields(city)

return render\_template("updatewhatrecord.html", fieldlist=fieldlist,clas=city)

elif selected\_radio == "ticket":

tick=Tickets()

fieldlist = GetClassFields(tick)

return render\_template("updatewhatrecord.html", fieldlist=fieldlist,clas=tick)

return redirect("/adminpanel")

@app.route('/updatewhatrecord', methods=['POST', 'GET'])

def updatewhatrecord():

if request.method == "GET":

if GetFromDict() is not None:

if GetFromDict().role\_client == "admin":

return render\_template("updatewhatrecord.html")

else:

return redirect("/home")

elif request.method == "POST":

clas = request.args.get('clas')

selected\_radio = request.form.get('listGroupCheckableRadios')

return render\_template("updatepost.html" ,dict=dict)

@app.route('/updatepost/<selectedfield>/<clas>', methods=['POST', 'GET'])

def updatepost(selectedfield, clas):

if request.method == "GET":

if GetFromDict() is not None:

if GetFromDict().role\_client == "admin":

ss = selectedfield

return render\_template("updatepost.html", selectedfield=ss, clas=clas, info=None)

else:

return redirect("/home")

if request.method == "POST":

item = selectedfield

value = request.form['value']

type=request.form['type']

id = request.form['id']

try:

obj\_class = globals()[clas]

obj = obj\_class()

except KeyError:

return render\_template("updatepost.html", selectedfield=selectedfield, clas=clas,

info="Не вдалося знайти клас")

try:

if type == "datetime":

value = datetime.strptime(value, '%Y-%m-%dT%H:%M')

except ValueError:

return render\_template("updatepost.html", selectedfield=selectedfield, clas=clas,

info="Помилка конвертації у формат дати/часу")

try:

if type == "int":

value = int(value)

except ValueError:

return render\_template("updatepost.html", selectedfield=selectedfield, clas=clas,

info="Помилка конвертації у формат int")

if db\_context.update\_item(id, item, value, obj):

return render\_template("adminpanel.html")

else:

return render\_template("updatepost.html", selectedfield=selectedfield, clas=clas, info="Не оновлено")

@app.route('/adminpanel')

def adminpanel():

if GetFromDict():

user = GetFromDict()

if user.role\_client == "admin":

return render\_template("adminpanel.html", user=user)

return redirect("/home")

def CreateUser(password,item, value):

if db\_context.get\_user(password, item, value)=="Пароль не співпадає":

return "Пароль не співпадає"

elif db\_context.get\_user(password,item,value):

user = db\_context.get\_user(password,item,value)

AddToDict(user)

return True

else:

return False

def GetClassFields(obj):

return list(obj.\_\_dict\_\_.keys())

def GenerateRandomNumber():

return ''.join(random.choices('0123456789', k=8))

def GetFromDict():

if session.get('user', {}):

user\_dict = session.get('user', {})

user = Clients.from\_dict(user\_dict)

return user

else:

return None

def AddToDict(user=Clients()):

user\_dict = user.to\_dict()

session['user'] = user\_dict

def DeleteFromDict():

if 'user' in session:

del session['user']

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

buses.py

class Buses:

def \_\_init\_\_(self):

self.id\_bus = None

self.name = None

self.seats = None

self.bus\_number = None

cardticket.py

class CardTicket:  
 def \_\_init\_\_(self):  
 self.id\_ticket=None  
 self.code = None  
 self.name=None  
 self.time=None  
 self.cost=None

cities.py

class Cities:  
 def \_\_init\_\_(self):  
 self.id\_city = None  
 self.name = None  
 self.country=None

clients.py

class Clients:  
 def \_\_init\_\_(self):  
 self.id\_client = None  
 self.name = None  
 self.phone = None  
 self.email = None  
 self.password = None  
 self.role\_client = None  
  
 def to\_dict(self):  
 return {  
 'id': self.id\_client,  
 'name': self.name,  
 'email': self.email,  
 'phone': self.phone,  
 'password': self.password,  
 'role\_client': self.role\_client  
 }  
  
 @classmethod  
 def from\_dict(cls, user\_dict):  
 user = cls()  
 user.id\_client = user\_dict.get('id\_client', '')  
 user.name = user\_dict.get('name', '')  
 user.email = user\_dict.get('email', '')  
 user.phone = user\_dict.get('phone', '')  
 user.password = user\_dict.get('password', '')  
 user.role\_client = user\_dict.get('role\_client', '')  
 return user

journeystable.py

class JourneysTable:

def \_\_init\_\_(self):

self.id\_journey=None

self.bus\_name = None

self.start\_city = None

self.finish\_city = None

self.time\_start = None

self.time\_finish = None

self.cost = None

def getJourneyId(self):

return self.id\_journey

non\_autorized\_users.py

class Non\_autorized\_users:  
 def \_\_init\_\_(self):  
 self.id\_user = None  
 self.name = None  
 self.phone = None

orders.py

class Orders:  
 def \_\_init\_\_(self):  
 self.id\_order = None  
 self.client\_id = None  
 self.non\_authorized\_users\_id = None  
 self.date\_buying = None  
 self.cost = None  
 self.whether\_paid = None

tickets.py

class Tickets:  
 def \_\_init\_\_(self):  
 self.id\_ticket = None  
 self.bus\_id = None  
 self.seat=None  
 self.date\_buying = None  
 self.journey\_id = None  
 self.non\_autorized\_users\_id = None  
 self.client\_id = None  
 self.order\_id = None  
 self.code=None

timetable.py

import datetime  
  
class Timetable:  
 def \_\_init\_\_(self):  
 self.id\_journey = None  
 self.bus\_id = None  
 self.city\_start\_id = None  
 self.city\_finish\_id = None  
 self.cost = None  
 self.time\_start = None  
 self.time\_finish = None  
 self.seats\_occupied=None  
 self.is\_active=None  
  
  
 def getBusId(self):  
 return self.id\_journey

**about.html**

{% extends 'base.html' %}

{% block title %}

Про нас

{% endblock %}

{% block body %}

<div class="b-example-divider"></div>

<div class="p-5 text-center bg-body-secondary">

<div class="container px-4 py-5 " id="hanging-icons">

<h2 class="pb-2 border-bottom">Про нас:</h2>

<div class="row g-4 py-5 row-cols-1 row-cols-lg-4">

<div class="col d-flex align-items-start">

<div style="" class="icon-square text-body-emphasis y d-inline-flex align-items-center justify-content-center fs-4 flex-shrink-0 me-3">

<img src="{{ url\_for('static', filename='svgs/bus\_schedule\_icon.svg') }}" alt="Profile SVG" width="52px" height="52px">

</div>

<div>

<h3 class="fs-3 text-body-emphasis">Розклад автобусів</h3>

<p>Переглядайте актуальний розклад руху автобусів, обирайте зручний час відправлення та призначення.</p>

</div>

</div>

<div class="col d-flex align-items-start">

<div class="icon-square text-body-emphasis d-inline-flex align-items-center justify-content-center fs-4 flex-shrink-0 me-3">

<img src="{{ url\_for('static', filename='svgs/distance.svg') }}" alt="Profile SVG" width="52px" height="52px">

</div>

<div>

<h3 class="fs-3 text-body-emphasis">Пошук маршрутів</h3>

<p>Знаходьте оптимальні маршрути між містами та вибирайте найзручніші варіанти подорожі.</p>

</div>

</div>

<div class="col d-flex align-items-start">

<div class="icon-square text-body-emphasis d-inline-flex align-items-center justify-content-center fs-4 flex-shrink-0 me-3">

<img src="{{ url\_for('static', filename='svgs/ticket.svg') }}" alt="Profile SVG" width="52px" height="52px">

</div>

<div>

<h3 class="fs-3 text-body-emphasis">Квитки онлайн</h3>

<p> Бронюйте та оплачуйте квитки прямо на сайті, уникайте черг та забезпечте собі місце в потрібному вам автобусі.</p>

</div>

</div>

<div class="col d-flex align-items-start">

<div class="icon-square text-body-emphasis d-inline-flex align-items-center justify-content-center fs-4 flex-shrink-0 me-3">

<img src="{{ url\_for('static', filename='svgs/info.svg') }}" alt="Profile SVG" width="52px" height="52px">

</div>

<div>

<h3 class="fs-3 text-body-emphasis">Інформація про автобуси</h3>

<p>Детальна інформація про автобуси, включаючи кількість місць, номер автобуса та інше.</p>

</div>

</div>

</div>

</div>

</div>

{% endblock %}

**accountview.html**

{% extends 'base.html' %}

{% block title %}

Аккаунт користувача

{% endblock %}

{% block body %}

<style>

#acc{

padding-left: 5%;

}

#acc>div{

margin-top: 10px;

padding: 25px;

border-bottom: 3px solid black;

border-radius: 25px;

background: linear-gradient(to bottom, #E9ECEF,white);

display: block;

width: 300px;

}

#acc>a{

margin-top: 25px;

}S

span{

font-size: 14px;

}

h3{

margin: 0;

}

</style>

<div id="acc" class="bg-body-secondary " >

<div class="text-center" style="display: inline-block">

<h1>Акаунт користувача</h1>

</div><br>

<div>

<h3>ПІБ</h3>

<h3><strong>{{ user.name }}</strong></h3><br>

</div>

<div>

<h3>Email</h3>

<h3><strong>{{ user.email }} </strong></h3><br>

</div>

<div>

<h3>Телефон</h3>

<h3><strong>{{ user.phone }}</strong></h3><br>

</div>

<a class="btn btn-primary" href="/usertickets">Мої квитки</a>

<a class="btn btn-primary" href="/exit">Вийти</a>

</div>

{% endblock %}

**addadmin.html**

{% extends 'baseadmin.html' %}

{% block title %}

Додати поїздку

{% endblock %}

{% block body %}

{% if info is not none %}

<script>

alert("Адміністратора не додано")

</script>

{% endif %}

<h1>Додавання нового адміністратора</h1>

<form method="post" >

<input type="text" name="name" id="name">

<label for="name">Ім'я Адміністратора</label><br>

<input type="text" name="email" id="email">

<label for="email">Email</label><br>

<input type="text" name="phone" id="phone">

<label for="phone">Phone</label><br>

<input type="text" name="password" id="password">

<label for="password">password</label><br>

<button type="submit" class="btn btn-primary">Додати</button>

</form>

<a href="/adminpanel" class="btn btn-primary">Назад</a>

{% endblock %}

**addbus.html**

{% extends 'baseadmin.html' %}

{% block title %}

Додати автобус

{% endblock %}

{% block body %}

{% if info is not none %}

<script>

alert("Автобус не додано")

</script>

{% endif %}

<h1>Додавання нового автобуса</h1>

<form method="post">

<input type="text" name="nameBus" id="nameBus">

<label for="nameBus">Назва автобуса</label><br>

<input type="text" name="seats" id="seats">

<label for="seats">Кількість місць</label><br>

<input type="text" name="busNumber" id="seats">

<label for="busNumber">Номер автобуса</label><br>

<button type="submit" class="btn btn-primary">Додати</button>

</form>

<a href="/adminpanel" class="btn btn-primary">Назад</a>

{% endblock %}

**addcity.html**

{% extends 'baseadmin.html' %}

{% block title %}

Додати місто

{% endblock %}

{% block body %}

{% if info is not none %}

<script>

alert("Місто не додано")

</script>

{% endif %}

<h1>Додавання нового міста</h1>

<form method="post">

<input type="text" name="nameCity" id="nameCity">

<label for="nameCity">Назва міста</label><br>

<input type="text" name="country" id="country">

<label for="country">Країна</label><br>

<button type="submit" class="btn btn-primary">Додати</button>

<a href="/adminpanel" class="btn btn-primary">Назад</a>

</form>

{% endblock %}

**addjourney.html**

{% extends 'baseadmin.html' %}

{% block title %}

Додати поїздку

{% endblock %}

{% block body %}

{% if info is not none %}

<script>

alert("Поїздку не додано")

</script>

{% endif %}

<h1>Додавання нової поїздки</h1>

<form method="post" >

<input type="text" name="idbus" id="idbus">

<label for="idbus">ID Автобуса</label><br>

<input type="text" name="idcitystart" id="idcitystart">

<label for="idcitystart">Місто відправлення</label><br>

<input type="text" name="idcityfinish" id="idcityfinish">

<label for="idcityfinish">Місто прибуття</label><br>

<input type="text" name="cost" id="cost">

<label for="cost">Плата за проїзд</label><br>

<input type="datetime-local" name="timestart" id="timestart">

<label for="timestart">Час відправлення</label><br>

<input type="datetime-local" name="timefinish" id="timefinish">

<label for="timefinish">Час прибуття</label><br>

<input type="text" name="seats\_occupied" id="seats">

<label for="seats\_occupied">Зайняті місця</label><br>

<button type="submit" class="btn btn-primary">Додати</button>

</form>

<table class="table table-striped">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">Назва</th>

<th scope="col">Місця</th>

<th scope="col">Місто відправлення</th>

<th scope="col">Місто прибуття</th>

</tr>

</thead>

<tbody>

{% for bus in buses %}

<tr>

<th scope="row">{{ bus.id\_bus }}</th>

<td> {{ bus.name }}</td>

<td>{{ bus.seats }}</td>

<td>{{ bus.bus\_number }}</td>

<td>{{ bus.bus\_number }}</td>

</tr>

{% endfor %}

</tbody>

</table>

<table class="table table-striped">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">ІД</th>

<th scope="col">Назва</th>

<th scope="col">Країна</th>

</tr>

</thead>

<tbody>

{% for city in cities %}

<tr>

<th scope="row">{{ city.id\_city }}</th>

<td>{{ city.name }}</td>

<td>{{ city.country }}</td>

</tr>

{% endfor %}

</tbody>

</table>

<a href="/adminpanel" class="btn btn-primary">Назад</a>

{% endblock %}

**adminpanel.html**

{% extends 'baseadmin.html' %}

{% block title %}

Адмінпанель

{% endblock %}

{% block body %}

<div class="my-5 ">

<div class="bg-body-secondary my-5 text-center" style="padding: 30px">

<a href="/addbus" class="btn btn-primary">Додати автобус</a>

<a href="/addcity" class="btn btn-primary">Додати місто</a>

<a href="/addjourney" class="btn btn-primary">Додати маршрут</a>

<a href="/addadmin" class="btn btn-primary">Додати адміна</a>

</div>

<div class="bg-body-secondary my-5 text-center" style="padding: 30px">

<a href="/update" class="btn btn-warning">Оновити записи</a>

<a href="/delete" class="btn btn-danger">Видалити записи</a>

</div>

</div>

<div class="bg-body-secondary my-5 text-center" style="padding: 30px">

<a href="/home" class="btn btn-primary">На головну</a>

</div>

{% endblock %}

**base.html**

<!DOCTYPE html>

<html lang="en">

<script>

function selectDiv(div) {

var seatNumber = div.getAttribute('data-seat');

var seatInput = document.querySelector('input[name="seat"]');

seatInput.value = seatNumber;

var busDivs = document.querySelectorAll('.bus > div');

busDivs.forEach(function (element) {

if (element.getAttribute('data-seat') === seatNumber) {

element.style.boxShadow='10px 10px 5px #0D6EFD'

element.style.backgroundColor = '#0a58ca';

element.style.color = 'white';

let labelseat =document.getElementById('labelseat')

labelseat.innerHTML='Ви вибрали місце : ' + seatInput.value

} else {

element.style.backgroundColor = '#0D6EFD';

element.style.color = 'white';

element.style.boxShadow=''

}

});

}

</script>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="shortcut icon" href="{{ url\_for('static', filename='svgs/bus\_schedule\_icon.svg') }}" type="image/x-icon">

<link rel="stylesheet" href="{{ url\_for('static',filename='css/main.css') }}">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL" crossorigin="anonymous"></script>

<title>{% block title %}{% endblock %}</title>

</head>

<body>

<style>

header {

position: fixed;

top: 0;

left: 0;

width: 100%;

background-color: #343a40;

z-index: 1000;

}

body {

padding-top: 70px;

}

@media (max-width: 1000px) {

body {

padding-top: 150px;

}

}

@media (max-width: 768px) {

body {

padding-top: 200px;

}

}

</style>

<header class="p-3 text-bg-dark" >

<div class="container" >

<div class="d-flex flex-wrap align-items-center justify-content-center justify-content-lg-start" >

<a href="/home" class="d-flex align-items-center mb-2 mb-lg-0 text-white text-decoration-none">

<h1 style="margin-right: 20px">OnAvtovokzal</h1>

</a>

<ul class="nav col-12 col-lg-auto me-lg-auto mb-2 justify-content-center mb-md-0">

<li><a href="/home" class="nav-link px-2 text-secondary">Головна</a></li>

<li><a href="/journeys" class="nav-link px-2 text-white">Розписання всіх автобусів</a></li>

<li><a href="/buyticket/" class="nav-link px-2 text-white">Купити квиток</a></li>

<li><a href="/About" class="nav-link px-2 text-white">Про нас</a></li>

</ul>

<form class="col-12 col-lg-auto mb-3 mb-lg-0 me-lg-3" role="search" method="POST" action="/getticket">

<script>

document.addEventListener('DOMContentLoaded', function() {

document.getElementById('search').addEventListener('focus', function() {

this.value = "";

});

});

function checkTicketId() {

var searchInput = document.getElementById('search');

var ticketId = searchInput.value.trim();

if (ticketId.length === 8 && !isNaN(ticketId)) {

searchInput.closest('form').submit();

}

}

</script>

<input type="search" id="search" name="ticket\_id" class="form-control form-control-dark text-bg-dark" value="Напишіть код квитка..." onkeyup="checkTicketId()">

</form>

{% if user is not none %}

<div class="text-end authorizationcompleted" style="margin-right:-50px ">

<a href="/accountview"> <img src="{{ url\_for('static', filename='svgs/profile.svg') }}" alt="Profile SVG" width="64px" height="64px"></a>

</div>

{% endif %}

{% if user is none %}

<div class="text-end authorizationnotcompleted">

<button type="button" class="btn btn-outline-light me-2" onclick="redirectToLogin()"> Ввійти</button>

<button type="button" class="btn btn-primary" onclick="redirectToRegister()">Зареєструватися</button>

</div>

{% endif %}

<script>

function redirectToLogin() {

window.location.href = "/loginselect";

}

function redirectToRegister() {

window.location.href = "/register";

}

</script>

</div>

</div>

</header>

{% block body %}{% endblock %}

<footer class="py-3 my-4 bg-body-secondary">

<ul class="nav justify-content-center border-bottom pb-3 mb-3">

<li class="nav-item"><a href="/home" class="nav-link px-2 text-body-secondary">Home</a></li>

<li class="nav-item"><a href="/about" class="nav-link px-2 text-body-secondary">About</a></li>

</ul>

<p class="text-center text-body-secondary ">© 2023-2024 OnAvtovokzal, Inc</p>

</footer>

</body>

</html>

**baseadmin.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="{{ url\_for('static',filename='css/main.css') }}">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL" crossorigin="anonymous"></script>

<title>{% block title %}{% endblock %}</title>

</head>

<body>

<div class="text-center my-5">

<h1>Панель адміністрації</h1>

</div>

{% block body %}{% endblock %}

</body>

</html>

**buyticket.html**

{% extends 'base.html' %}

{% block title %}

Купити квитки

{% endblock %}

{% block body %}

{% block form %}

{% endblock %}

{% block table %}

{% endblock %}

{% block pay %}

{% endblock %}

{% endblock %}

**delete.html**

{% extends 'baseadmin.html' %}

{% block title %}

Видалити

{% endblock %}

{% block body %}

<script>

document.addEventListener('DOMContentLoaded', function () {

let radioButtons = document.querySelectorAll('input[name="listGroupCheckableRadios"]');

let labels = document.querySelectorAll('.list-group-item');

for (let i = 0; i < radioButtons.length; i++) {

radioButtons[i].addEventListener('change', function () {

for (let j = 0; j < radioButtons.length; j++) {

if (radioButtons[j].checked) {

labels[j].classList.add("active");

labels[j].classList.remove("non-active");

} else {

labels[j].classList.remove("active");

labels[j].classList.add("non-active");

}

}

});

}

});

</script>

<form method="post">

<div class="d-flex flex-column flex-md-row p-4 gap-4 py-md-5 align-items-center justify-content-center">

<div class="list-group list-group-checkable d-grid gap-2 border-0">

<h2>Виберіть що хочете видалити</h2>

<input style="display: none;" class="list-group-item-check pe-none " type="radio" name="listGroupCheckableRadios" id="listGroupCheckableRadios1" value="bus" checked="">

<label class="list-group-item rounded-3 py-3 active" for="listGroupCheckableRadios1">

Автобус

</label>

<input style="display: none;" class="list-group-item-check pe-none" type="radio" name="listGroupCheckableRadios" id="listGroupCheckableRadios2" value="journey">

<label class="list-group-item rounded-3 py-3" for="listGroupCheckableRadios2">

Подорож

</label>

<input style="display: none;" class="list-group-item-check pe-none" type="radio" name="listGroupCheckableRadios" id="listGroupCheckableRadios3" value="city">

<label class="list-group-item rounded-3 py-3" for="listGroupCheckableRadios3">

Місто

</label>

<input style="display: none;" class="list-group-item-check pe-none" type="radio" name="listGroupCheckableRadios" id="listGroupCheckableRadios4" value="ticket">

<label class="list-group-item rounded-3 py-3" for="listGroupCheckableRadios4">

Квиток

</label>

<input style="display: none;" class="list-group-item-check pe-none" type="radio" name="listGroupCheckableRadios" id="listGroupCheckableRadios5" value="client">

<label class="list-group-item rounded-3 py-3" for="listGroupCheckableRadios5">

Клієнт

</label>

</div>

<button type="submit" class="btn btn-primary">Видалити</button>

<a href="/adminpanel" class="btn btn-primary">Назад</a>

</div>

</form>

{% endblock %}

**deletepost.html**

{% extends 'baseadmin.html' %}

{% block title %}

Видалити запис

{% endblock %}

{% block body %}

<script>

document.addEventListener('DOMContentLoaded', function () {

const fields = {

'id\_bus': 'int',

'name': 'str',

'seats': 'int',

'seats\_occupied': 'int',

'bus\_number': 'str',

'id\_city': 'int',

'country': 'str',

'id\_client': 'int',

'phone': 'str',

'email': 'str',

'password': 'str',

'role\_client': 'str',

'id\_ticket': 'int',

'id\_journey': 'int',

'seat': 'int',

'date\_buying': 'datetime',

'bus\_id': 'int',

'city\_start\_id': 'int',

'city\_finish\_id': 'int',

'cost': 'int',

'time\_start': 'datetime',

'time\_finish': 'datetime'

};

const selectedField = '{{ selectedfield }}';

const form = document.getElementsByTagName('form')[0];

console.log("field"+ selectedField);

let fieldType = fields[selectedField.replace(/["']/g, "")];

let newInput=document.createElement('input');

if (fieldType==='int'){

newInput.type='number';

}

else if (fieldType==='str'){

newInput.type='text';

}

else if (fieldType==='datetime'){

window.location.href="/home";

}

newInput.name = 'value';

newInput.id = 'value';

let newLabel = document.createElement('label');

newLabel.for = 'value';

newLabel.innerHTML = "Значення поля";

let inp=document.createElement('input');

inp.type='hidden';

inp.id='type';

inp.name='type';

inp.value=fieldType;

let h2Element = document.querySelector('h2');

h2Element.insertAdjacentElement('afterend', newInput);

h2Element.insertAdjacentElement('afterend', newLabel);

h2Element.insertAdjacentElement('afterend', inp);

});

function redirectToRegister(url) {

window.location.href = url;

}

</script>

{% if info is none %}

<form method="post">

<h2>Видалення запису</h2>

<button type="submit">Видалити</button>

</form>

{% else %}

<script>

alert('Не оновлено')

</script>

{% endif %}

{% endblock %}

**deletewhatrecord.html**

{% extends 'baseadmin.html' %}

{% block title %}

Видалити запис

{% endblock %}

{% block body %}

<script>

document.addEventListener('DOMContentLoaded', function () {

let radioButtons = document.querySelectorAll('input[name="listGroupCheckableRadios"]');

let labels = document.querySelectorAll('.list-group-item');

for (let i = 0; i < radioButtons.length; i++) {

radioButtons[i].addEventListener('change', function () {

for (let j = 0; j < radioButtons.length; j++) {

if (radioButtons[j].checked) {

labels[j].classList.add("active");

labels[j].classList.remove("non-active");

} else {

labels[j].classList.remove("active");

labels[j].classList.add("non-active");

}

}

});

}

});

function redirectToNext(event) {

event.preventDefault()

let selectedOption = document.querySelector('input[name="listGroupCheckableRadios"]:checked').value;

let clas="{{ clas.\_\_class\_\_.\_\_name\_\_ }}"

console.log(clas)

window.location.href = "/deletepost/" + selectedOption+"/"+clas;

}

</script>

<form method="post">

<div class="d-flex flex-column flex-md-row p-4 gap-4 py-md-5 align-items-center justify-content-center">

<div class="list-group list-group-checkable d-grid gap-2 border-0">

<h2>За яким полем будемо видаляти? ?</h2>

{% for i in fieldlist %}

<input class="list-group-item-check pe-none " type="radio" name="listGroupCheckableRadios" id="{{ i }}" value="{{ i }}" style="display: none;">

<label class="list-group-item rounded-3 py-3 active" for="{{ i }}">{{ i }}</label><br>

{% endfor %}

<button type="submit" class="btn btn-primary" onclick="redirectToNext(event)">Видалити за цим</button>

<a href="/adminpanel" class="btn btn-primary">Назад</a>

</div>

</div>

</form>

{% endblock %}

**formbuyticket.html**

{% extends 'buyticket.html' %}

{% block form %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Form</title>

</head>

<body>

<script>

document.addEventListener('DOMContentLoaded', function () {

document.querySelector('input[name="phone"]').addEventListener('input', function () {

phoneValid()

});

});

document.addEventListener('DOMContentLoaded', function () {

document.querySelector('input[name="finishtime"]').addEventListener('change', function () {

timeValid()

})

})

function timeValid(){

let timestart = document.querySelector('input[name="starttime"]')

let timefinish = document.querySelector('input[name="finishtime"]')

if ( timefinish.value > timestart.value) {

timefinish.style.border = "solid 2px green"

}

else {

timefinish.style.border = "solid 2px red";

}

}

function phoneValid(event=""){

let phone=document.querySelector('input[name="phone"]')

let phoneNumber = phone.value

if (phoneNumber.length === 10) {

phone.style.border = "2px solid green";

//redirectToSelect()

}

else {

phone.style.border = "2px solid red";

let labphone = document.getElementById('labphone');

labphone.innerHTML = "Введіть коректний номер";

event.preventDefault()

}

console.log(phoneNumber)

}

</script>

<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyA9hOcqXCKQ9PCEYOwzO2r-n1\_3aU\_p8Do&libraries=places"></script>

<script>

let citystart;

let cityfinish;

let citylist = [];

{% for city in cities %}

citylist.push({

name: "{{ city.name }}",

country: "{{ city.country }}"

});

{% endfor %}

console.log(citylist);

document.addEventListener('DOMContentLoaded',function (){

document.querySelector('select[name="citystart"]').addEventListener('change', function() {

console.log("strrt")

setCities()

});

document.querySelector('select[name="cityfinish"]').addEventListener('change', function() {

console.log("finish")

setCities()

});

})

function setCities(){

citystart=document.querySelector('select[name="citystart"]').value

cityfinish=document.querySelector('select[name="cityfinish"]').value

if (citystart!="Відправляємся з" && cityfinish!="Їдем в") {

console.log("Обидва міста вибрані:", citystart, cityfinish);

let googlemap = document.querySelector('div[id="googlemap"]')

googlemap.style.display="block"

initMap()

}

}

function initMap() {

let map = new google.maps.Map(document.getElementById('map'), {

center: {lat: 37.7749, lng: -122.4194},

zoom: 13

});

var directionsService = new google.maps.DirectionsService();

var directionsRenderer = new google.maps.DirectionsRenderer({map: map});

calculateAndDisplayRoute(directionsService, directionsRenderer);

}

function calculateAndDisplayRoute(directionsService, directionsRenderer) {

let or = citystart+", "+ citylist[citystart]

let dest = cityfinish+", "+ citylist[cityfinish]

directionsService.route(

{

origin: or,

destination: dest,

travelMode: 'DRIVING'

},

function(response, status) {

if (status === 'OK') {

directionsRenderer.setDirections(response);

var route = response.routes[0];

var durationInSeconds = 0;

for (var i = 0; i < route.legs.length; i++) {

durationInSeconds += route.legs[i].duration.value;

}

var hours = Math.floor(durationInSeconds / 3600);

var minutes = Math.floor((durationInSeconds % 3600) / 60);

var travelTimeInfo = document.getElementById('travel-time');

travelTimeInfo.textContent = 'Приблизний час подорожі : ' + hours + ' годин ' + minutes + ' хвилин';

} else {

window.alert('Не вдалося розрахувати маршрут через: ' + status);

}

}

);

}

</script>

{% if user is none %}

<div class="d-flex align-items-center justify-content-center py-4 bg-body-tertiary">

<form class="text-center" method="GET">

<div class="form-floating">

<input type="text" class="form-control" id="inpname" name="name" required>

<label for="inpname">Введіть ім'я клієнта</label>

</div>

<div class="form-floating">

<input type="text" class="form-control" id="inpphone" name="phone" required>

<label id="labphone" for="inpname">Введіть телефон клієнта</label>

</div>

{% else %}

<div class="d-flex align-items-center justify-content-center py-4 bg-body-tertiary">

<form class="text-center" method="GET">

{% endif %}

<div class="d-flex align-items-center justify-content-center py-4 bg-body-tertiary">

<div class="row text-center">

<h3>Замовити квиток</h3>

<div class="col-md-12 themed-grid-col" style="padding: 5px">

<div class="row justify-content-center">

<div class="col-md-6 themed-grid-col">

<select name="citystart" class="form-select" required>

<option>Відправляємся з</option>

{% for city in cities %}

<option>{{ city.name }}</option>

{% endfor %}

</select>

</div>

<div class="col-md-6 themed-grid-col">

<select name="cityfinish" class="form-select" required>

<option>Їдем в</option>

{% for city in cities %}

<option>{{ city.name }}</option>

{% endfor %}

</select>

</div>

</div>

<div class="col-md-12 themed-grid-col" style="padding: 5px">

<div class="row justify-content-center">

<div class="col-md-2 themed-grid-col" style="margin-top: 6px">Відправлення від</div>

<div class="col-md-4 themed-grid-col">

<input class="form-select" type="datetime-local" name="starttime" id="startinput" required>

</div>

<div class="col-md-2 themed-grid-col" style="margin-top: 6px">До</div>

<div class="col-md-4 themed-grid-col">

<input class="form-select" type="datetime-local" name="finishtime" id="finishinput" required>

</div>

</div>

</div>

<div id="googlemap" class="align-items-cente" style="display: none">

<div id="map" style="height: 400px;"></div>

<div id="travel-time"></div>

</div>

<div class="align-items-cente">

<button type="submit" class="btn btn-primary" style="padding-left: 50px;padding-right: 50px" onclick="phoneValid(event)">Знайти</button>

</div>

</div>

</div>

</div>

</form>

</div>

</body>

</html>

{% endblock %}

**home.html**

{% extends 'base.html' %}

{% block title %}

Головна

{% endblock %}

{% block body %}

<div class="b-example-divider"></div>

<div class="b-example-divider"></div>

<div class="p-5 text-center bg-body-secondary">

<div class="container py-5">

<h1 class="text-body-emphasis">Ласкаво просимо на "OnAvtovokzal" – ваш онлайн автовокзал!</h1>

<p class="col-lg-8 mx-auto lead">

Вітаємо вас на найзручнішому та найсучаснішому сервісі для планування вашого мандрівного дня. "OnAvtovokzal" - це онлайн платформа, яка дозволяє вам легко та зручно організувати ваші подорожі автобусом.

</p>

</div>

</div>

</div>

<div class="b-example-divider"></div>

<div class="p-5 text-center bg-body-secondary">

<div class="container px-4 py-5 " id="hanging-icons">

<h2 class="pb-2 border-bottom">Наші можливості:</h2>

<div class="row g-4 py-5 row-cols-1 row-cols-lg-4">

<div class="col d-flex align-items-start">

<div style="" class="icon-square text-body-emphasis y d-inline-flex align-items-center justify-content-center fs-4 flex-shrink-0 me-3">

<img src="{{ url\_for('static', filename='svgs/bus\_schedule\_icon.svg') }}" alt="Profile SVG" width="52px" height="52px">

</div>

<div>

<h3 class="fs-3 text-body-emphasis">Розклад автобусів</h3>

<p>Переглядайте актуальний розклад руху автобусів, обирайте зручний час відправлення та призначення.</p>

<a href="/journeys" class="btn btn-primary">

Переглянути розклад

</a>

</div>

</div>

<div class="col d-flex align-items-start">

<div class="icon-square text-body-emphasis d-inline-flex align-items-center justify-content-center fs-4 flex-shrink-0 me-3">

<img src="{{ url\_for('static', filename='svgs/distance.svg') }}" alt="Profile SVG" width="52px" height="52px">

</div>

<div>

<h3 class="fs-3 text-body-emphasis">Пошук маршрутів</h3>

<p>Знаходьте оптимальні маршрути між містами та вибирайте найзручніші варіанти подорожі.</p>

<a href="/formbuyticket" class="btn btn-primary">

Знайти

</a>

</div>

</div>

<div class="col d-flex align-items-start">

<div class="icon-square text-body-emphasis d-inline-flex align-items-center justify-content-center fs-4 flex-shrink-0 me-3">

<img src="{{ url\_for('static', filename='svgs/ticket.svg') }}" alt="Profile SVG" width="52px" height="52px">

</div>

<div>

<h3 class="fs-3 text-body-emphasis">Квитки онлайн</h3>

<p> Бронюйте та оплачуйте квитки прямо на сайті, уникайте черг та забезпечте собі місце в потрібному вам автобусі.</p>

<a href="/formbuyticket" class="btn btn-primary">

Забронювати

</a>

</div>

</div>

<div class="col d-flex align-items-start">

<div class="icon-square text-body-emphasis d-inline-flex align-items-center justify-content-center fs-4 flex-shrink-0 me-3">

<img src="{{ url\_for('static', filename='svgs/info.svg') }}" alt="Profile SVG" width="52px" height="52px">

</div>

<div>

<h3 class="fs-3 text-body-emphasis">Інформація про автобуси</h3>

<p>Детальна інформація про автобуси, включаючи кількість місць, номер автобуса та інше.</p>

<a href="/journeys" class="btn btn-primary" style="">

Дізнатись інформацію

</a>

</div>

</div>

</div>

</div>

</div>

{% endblock %}

**index.html**

{% extends 'base.html' %}

{% block title %}

Головна

{% endblock %}

{% block body %}

<h1>Головна сторінка</h1>

{% endblock %}

**journeys.html**

{% extends 'base.html' %}

{% block title %}

Розклад автобусів

{% endblock %}

{% block body %}

<style>

td{

justify-content: center;

}

</style>

<!--<script>

document.addEventListener('DOMContentLoaded', function() {

// Отримуємо всі форми на сторінці

var forms = document.querySelectorAll('form');

// Додаємо обробник подій для кожної форми

forms.forEach(function(form) {

form.addEventListener('submit', function(event) {

event.preventDefault();

var formData = new FormData(form);

fetch('/selectplace/', {

method: 'POST',

body: formData

})

.then(response => {

if (response.ok) {

return response.text();

} else {

console.error('Помилка: ' + response.status);

}

})

.then(html => {

// Оновлюємо вміст поточної сторінки з отриманим HTML

document.body.innerHTML = html;

})

.catch(error => {

console.error('Помилка виконання запиту:', error);

});

});

});

});

</script>-->

<script>

function submitForm(index) {

let nameForm="buyTicketForm"+ index.toString()

//if starttime

document.getElementById('buyTicketForm').submit();

}

async function redirectToSelect() {

const itemSort = document.querySelector('select[name="item\_sort"]').value;

const direction = document.querySelector('select[name="direction"]').value;

const citystart = document.querySelector('select[name="citystart"]').value;

if (document.querySelector('input[name="cityfinish"]').value){

const cityfinish = document.querySelector('select[name="cityfinish"]').value;

}

if (document.querySelector('input[name="starttime"]').value){

const starttime = document.querySelector('select[name="starttime"]').value;

let hiddinpstarttime=document.querySelector('input[name="starttime"]')

hiddinpstarttime.value=starttime

}

if (document.querySelector('input[name="finishtime"]').value){

const finishtime = document.querySelector('select[name="finishtime"]').value;

let hiddinpfinishtime=document.querySelector('input[name="finishtime"]')

hiddinpfinishtime.value=finishtime

}

//alert("POST")

const formData = new FormData();

formData.append('item\_sort', itemSort);

formData.append('direction', direction);

formData.append('citystart', citystart);

if (document.querySelector('input[name="cityfinish"]').value){

formData.append('cityfinish', cityfinish);

}

if (document.querySelector('input[name="starttime"]').value){

formData.append('starttime', starttime);

}

if (document.querySelector('input[name="finishtime"]').value){

formData.append('finishtime', finishtime);

}

try {

const response = await fetch('/journeys', {

method: 'POST',

body: formData

});

if (response.ok) {

window.location.href='/journeys';

window.location.reload();

} else {

// Обробка помилок

console.error('Помилка при відправці POST-запиту');

}

} catch (error) {

console.error('Помилка при виконанні fetch:', error);

}

}

</script>

<div class=" text-center bg-body-secondary" style="margin-top: 25px !important;">

<div class="container py-3">

<h1>Розклад всіх автобусів</h1>

</div>

</div>

<div class="b-example-divider"></div>

<div class="my-5 ">

<form class="form-control bg-body-secondary">

<div class="row text-center">

<h3>Фільтр</h3>

<div class="col-md-11 themed-grid-col" style="padding: 5px">

<div class="row justify-content-center">

<div class="col-md-1 themed-grid-col">

<select name="citystart" class="form-select" required>

<option>Відправляємся з</option>

{% for city in cities %}

<option>{{ city.name }}</option>

{% endfor %}

</select>

</div>

<div class="col-md-1 themed-grid-col">

<select name="cityfinish" class="form-select" required>

<option>Їдем в</option>

{% for city in cities %}

<option>{{ city.name }}</option>

{% endfor %}

</select>

</div>

<div class="col-md-2 themed-grid-col">

<select name="item\_sort" class="form-select" required>

<option>Назва автобусу</option>

<option>За іменем міста відправлення</option>

<option>За іменем міста прибуття</option>

<option>За часом відправлення</option>

<option>За часом прибуття</option>

</select>

</div>

<div class="col-md-2 themed-grid-col">

<select name="direction" class="form-select" required>

<option>По спаданню</option>

<option>По зростанню</option>

</select>

</div>

<div class="col-md-1 themed-grid-col" >Відправлення від</div>

<div class="col-md-2 themed-grid-col">

<input class="form-select" type="datetime-local" name="starttime" id="startinput">

</div>

<div class="col-md-1 themed-grid-col" style="margin-top: 6px">До</div>

<div class="col-md-2 themed-grid-col">

<input class="form-select" type="datetime-local" name="finishtime" id="finishinput">

</div>

</div>

</div>

<div class="col-md-1 themed-grid-col" style="padding: 5px">

<button class="btn btn-primary" type="submit" id="btn\_sort" onclick="redirectToSelect()">Фільтрувати</button>

</div>

</div>

</form>

</div>

{% if table is none %}

<script>

//alert('1')

</script>

<div class="bd-example-snippet bd-code-snippet">

<div class="bd-example m-0 border-0">

<table class="table table-striped">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">Автобус</th>

<th scope="col">Місто відправлення</th>

<th scope="col">Місто прибуття</th>

<th scope="col">Час відправлення</th>

<th scope="col">Час прибуття</th>

<th scope="col">Ціна</th>

<th scope="col">Замовити</th>

</tr>

</thead>

<tbody>

{% for journey in journeys %}

<tr>

<th scope="row">{{ loop.index }}</th>

<td>

{% for bus in buses %}

{% if journey.bus\_id == bus.id\_bus %}

{{ bus.name }}

{% endif %}

{% endfor %}

</td>

<td>

{% for city in cities %}

{% if journey.city\_start\_id == city.id\_city %}

{{ city.name }}

{% endif %}

{% endfor %}

</td>

<td>

{% for city in cities %}

{% if journey.city\_finish\_id == city.id\_city %}

{{ city.name }}

{% endif %}

{% endfor %}

</td>

<td>{{ journey.time\_start }}</td>

<td>{{ journey.time\_finish }}</td>

<td>{{ journey.cost }}</td>

<td>

{% if user is not none %}

<form action="/selectplace/" method="post" >

<input type="hidden" name="id\_journey" value="{{ journey.id\_journey }}">

<input type="hidden" name="bus\_name" value="{% for bus in buses %}

{% if bus.id\_bus == journey.bus\_id %}

{{ bus.name }}

{% endif %}

{% endfor %}">

<input type="hidden" name="citystart" value="{{ citystart }}">

<input type="hidden" name="cityfinish" value="{{ cityfinish }}">

<button type="submit" class="btn btn-primary" >Купити</button>

</form>

{% endif %}

{% if user is none %}

<a class="btn btn-primary" href="/loginselect">Ввійти</a>

{% endif %}

</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

{% endif %}

{% if table is not none %}

<div class="bd-example-snippet bd-code-snippet">

<div class="bd-example m-0 border-0">

<table class="table table-striped">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">Автобус</th>

<th scope="col">Місто відправлення</th>

<th scope="col">Місто прибуття</th>

<th scope="col">Час відправлення</th>

<th scope="col">Час прибуття</th>

<th scope="col">Ціна</th>

<th scope="col">Замовити</th>

</tr>

</thead>

<tbody>

{% for tab in table %}

<tr>

<th scope="row">{{ loop.index }}</th>

<td>

{{ tab.bus\_name }}

</td>

<td>

{{ tab.start\_city }}

</td>

<td>

{{ tab.finish\_city }}

</td>

<td>{{ tab.time\_start }}</td>

<td>{{ tab.time\_finish }}</td>

<td>{{ tab.cost }}</td>

<td>

{% if user is not none %}

<form action="/selectplace/" method="post">

<input type="hidden" name="id\_journey" value="{{ tab.id\_journey }}">

<input type="hidden" name="bus\_name" value="{{ tab.bus\_name }}">

<input type="hidden" name="start\_city" value="{{ tab.start\_city }}">

<input type="hidden" name="finish\_city" value="{{ tab.finish\_city }}">

<button type="submit" class="btn btn-primary" >Купити</button>

</form>

{% endif %}

{% if user is none %}

<a class="btn btn-primary" href="/loginselect">Ввійти</a>

{% endif %}

</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

{% endif %}

{% endblock %}

**login.html**

{% extends 'base.html' %}

{% block title %}

Ввійти в акаунт

{% endblock %}

{% block body %}

{% if info=="Не вірний пароль" %}

<script>

alert("Не вірний пароль")

</script>

{% endif %}

<style>

div.form-floating>input{

margin-bottom: 5px;

}

form.my>button {

margin-top: 10px;

}

</style>

<div class="d-flex align-items-center justify-content-center py-4 bg-body-tertiary">

<form class="text-center my" method="post">

<h1 class="h3 mb-3 fw-normal">Вхід</h1>

{% if data == 'email' %}

<div class="form-floating">

<input type="email" class="form-control" id="floatingInputEmail" name="floatingInputEmail" placeholder="name@example.com">

<label for="floatingInput">Введіть email</label>

</div>

{% endif %}

{% if data == 'phone' %}

<div class="form-floating">

<input type="text" class="form-control" id="floatingInputPhone" name="floatingInputPhone" placeholder="Phone">

<label for="floatingInputPhone">Введіть телефон</label>

</div>

{% endif %}

<div class="form-floating">

<input type="password" class="form-control" id="floatingPassword" name="floatingPassword" placeholder="Password">

<label id="passwordlabel" for="floatingPassword">Введіть пароль</label>

</div>

<button class="btn btn-primary w-100 py-2" type="submit">Ввійти</button>

<p class="mt-5 mb-3 text-body-secondary">©2024</p>

</form>

</div>

{% endblock %}

**loginselect.html**

{% extends 'base.html' %}

{% block title %}

Виберіть спосіб входу

{% endblock %}

{% block body %}

<script>

function redirectToLogin() {

let selectedOption = document.querySelector('input[name="listGroupCheckableRadios"]:checked').value;

window.location.href = "/login/" + selectedOption;

}

document.addEventListener('DOMContentLoaded', function () {

let radioButtons = document.querySelectorAll('input[name="listGroupCheckableRadios"]');

let labels = document.querySelectorAll('.list-group-item');

for (let i = 0; i < radioButtons.length; i++) {

radioButtons[i].addEventListener('change', function () {

for (let j = 0; j < radioButtons.length; j++) {

if (radioButtons[j].checked) {

labels[j].classList.add("active");

labels[j].classList.remove("non-active");

} else {

labels[j].classList.remove("active");

labels[j].classList.add("non-active");

}

}

});

}

});

</script>

<div class="d-flex flex-column flex-md-row p-4 gap-4 py-md-5 align-items-center justify-content-center">

<div class="list-group list-group-checkable d-grid gap-2 border-0">

<input style="display: none;" class="list-group-item-check pe-none " type="radio" name="listGroupCheckableRadios" id="listGroupCheckableRadios1" value="email" checked="">

<label class="list-group-item rounded-3 py-3 active" for="listGroupCheckableRadios1">

Через email

</label>

<input style="display: none;" class="list-group-item-check pe-none" type="radio" name="listGroupCheckableRadios" id="listGroupCheckableRadios2" value="phone">

<label class="list-group-item rounded-3 py-3" for="listGroupCheckableRadios2">

Через телефон

</label>

</div>

<button type="button" class="btn btn-primary" onclick="redirectToLogin()">Ввійти</button>

</div>

{% endblock %}

**pay.html**

{% extends "base.html" %}

{% block title %}

Оплата

{% endblock %}

{% block body %}

<style>

div.main{

display: inline-block;

width: 100%;

}

</style>

<div class="main text-center" style="margin-top: 100px">

<h2>Інформація</h2>

<div class="bd-example-snippet bd-code-snippet">

<div class="bd-example m-0 border-0" style="margin-top: 50px !important;">

<table class="table table-striped">

<thead>

<tr>

<th scope="col">Місце</th>

<th scope="col">Місто відправлення</th>

<th scope="col">Місто прибуття</th>

<th scope="col">Час відправлення</th>

<th scope="col">Час прибуття</th>

<th scope="col">Номер автобуса</th>

<th scope="col">Автобус</th>

<th scope="col">Ціна</th>

</tr>

</thead>

<tbody>

<tr>

<td>{{ seat }}</td>

<td>{{ start\_city.name }}</td>

<td>{{ finish\_city.name }}</td>

<td>{{ journey.time\_start }}</td>

<td>{{ journey.time\_finish }}</td>

<td>{{ bus.bus\_number }}</td>

<td>{{ bus.name }}</td>

<td>{{ journey.cost }}</td>

</tr>

</tbody>

</table>

<form method="post" action="/succesorder/">

{% if non\_auth\_user is not none %}

<input type="hidden" name="phone" value="{{ non\_auth\_user.phone }}">

<input type="hidden" name="name" value="{{ non\_auth\_user.name }}">

{% endif %}

<input type="hidden" name="id\_journey" value="{{ journey.id\_journey}}">

<input type="hidden" name="seat" value="{{ seat }}">

<button type="submit" class="btn btn-primary"> Купити </button>

</form>

</div>

</div>

</div>

{% endblock %}

**register.html**

{% extends 'base.html' %}

{% block title %}

Зареєструватись

{% endblock %}

{% block body %}

{% if info is not none %}

<script>

let info = {{ info|tojson|safe }}

alert(info)

</script>

{% endif %}

<style>

div.form-floating>input{

margin-bottom: 5px;

}

form.my>button {

margin-top: 10px;

}

</style>

<div class="d-flex align-items-center justify-content-center py-4 bg-body-tertiary">

<form class="text-center my" method="post">

<h1 class="h3 mb-3 fw-normal">Реєстрація</h1>

<div class="form-floating">

<input type="text" class="form-control" id="floatingInputName" name="floatingInputName" placeholder="Ім'я" required>

<label for="floatingInputPhone">Ім'я та прізвище</label>

</div>

<div class="form-floating">

<input type="email" class="form-control" id="floatingInputEmail" name="floatingInputEmail" placeholder="name@example.com" required>

<label for="floatingInput">Введіть email</label>

</div>

<div class="form-floating">

<input type="text" class="form-control" id="floatingInputPhone" name="floatingInputPhone" placeholder="Phone" required>

<label id ="labphone"for="floatingInputPhone">Введіть телефон</label>

</div>

<div class="form-floating">

<input type="password" class="form-control" id="floatingInputPassOne" name="floatingInputPassOne" placeholder="Password" required>

<label for="floatingPassword">Пароль</label>

</div>

<div class="form-floating">

<input type="password" class="form-control" id="floatingInputPassTwo" name="floatingInputPassTwo" placeholder="Password" required>

<label id="labtwo" for="floatingPassword">Повторіть пароль</label>

</div>

<script>

let username = "{{ user.name }}";

console.log("NAME "+username)

document.addEventListener('DOMContentLoaded', function () {

document.getElementById('floatingInputPhone').addEventListener('blur', function () {

let labphone = document.getElementById('labphone');

let phoneNumber = this.value.replace(/\D/g, '');

if (phoneNumber.length === 10) {

this.style.border = "2px solid green";

labphone.innerHTML = "Номер телефону вірний";

//this.value = "+" + phoneNumber;

} else {

this.style.border = "2px solid red";

labphone.innerHTML = "Введіть вірний номер телефону";

}

});

});

document.addEventListener('DOMContentLoaded', function () {

document.getElementById('floatingInputPhone').addEventListener('change', function () {

if (this.value.length<=11||this.value.match(/[a-zA-ZА-Яа-я]/)) {

let labphone = document.getElementById('labphone');

this.style.border = "2px solid red";

labphone.innerHTML = "Введіть вірний номер";

}

});

});

function intersting(event) {

let pass2 = document.getElementById('floatingInputPassTwo');

let pass1 = document.getElementById('floatingInputPassOne');

let labeltwo = document.getElementById('labtwo');

if (pass1.value !== pass2.value && pass1.value !== "" && pass2.value !== "") {

pass2.style.border = "2px solid red";

labeltwo.innerHTML = "Введіть схожий пароль";

event.preventDefault();

}

else{

if (pass1.value !== "" && pass2.value !== "") {

}

}

}

</script>

<button class="btn btn-primary w-100 py-2" type="submit" id="buttonsumb" onclick="intersting(event)">Зареєструватись</button>

<p class="mt-5 mb-3 text-body-secondary">©2024</p>

</form>

</div>

{% endblock %}

**selectplace.html**

{% extends "base.html" %}

{% block title %}

Вибрати місце

{% endblock %}

{% block body %}

<style>

.maindiv {

display: inline-block;

margin-top: 100px;

width: 100%;

background-color: white;

}

.bus {

margin-top: 50px;

margin-bottom: 50px;

margin-left: 30%;

margin-right: 30%;

background-color: darkgray;

border: 3px solid grey;

border-radius: 20px;

padding: 50px;

padding-top: 100px;

display: flex;

flex-wrap: wrap;

justify-content: space-around;

}

.bus>div {

margin: 20px;

width: 100px;

height: 100px;

background-color: #0D6EFD;

color: white;

padding: 20px;

display: flex;

justify-content: center;

align-items: center;

border-radius: 20px;

cursor: pointer;

}

.bus>div:nth-child(3n) {

margin-left: 70px;

}

.bus>div:hover {

border: 3px solid #0a58ca;

box-shadow: 5px 5px #0D6EFD;

background-color: #0a58ca;

}

.doesnt-select {

background-color: grey !important;

color: white !important;

}

</style>

<script>

function validation(event) {

var seatInput = document.querySelector('input[name="seat"]');

var seatValue = seatInput.value.trim();

if (seatValue === "") {

event.preventDefault();

}

}

</script>

<div class="maindiv text-center">

<h2>Виберіть квиток</h2>

<div><span style="color: grey">{{ countstickets }}/{{ bus.seats }}</span></div>

<div class="bus">

{% set indexticket = 0 %}

{% for seat in range(bus.seats) %}

{% set index = loop.index %}

{% if greyseats is not none %}

{% if index in greyseats %}

<div class="doesnt-select" data-seat="{{ index }}">{{ index }}</div>

{% set indexticket = indexticket + 1 %}

{% else %}

<div class="selectable-seat" data-seat="{{ index }}" onclick="selectDiv(this)">{{ index }}</div>

{% endif %}

{% else %}

<div class="selectable-seat" data-seat="{{ index }}" onclick="selectDiv(this)">{{ index }}</div>

{% endif %}

{% endfor %}

</div>

<form method="POST" action="/pay/">

{% if non\_auth\_user is not none %}

<input type="hidden" name="phone" value="{{ non\_auth\_user.phone }}">

<input type="hidden" name="name" value="{{ non\_auth\_user.name }}">

{% endif %}

<input type="hidden" name="id\_journey" value="{{ journey.id\_journey }}">

<input type="hidden" name="seat" value="">

<h4 id="labelseat">Ви вибрали місце : </h4>

<button class="btn btn-primary" name="butsub" onclick="validation(event)">Перейти до оплати</button>

</form>

</div>

{% endblock %}

**succesorder.html**

{% extends "base.html" %}

{% block title %}

{% if title is not none %}

Succes

{% else %}

Failed

{% endif %}

{% endblock %}

{% block body %}

<style>

#divcent {

margin-top: -50px;

margin-bottom: 50px;

margin-left: 30%;

margin-right: 30%;

background-color: #E9ECEF;

border: 3px solid #E9ECEF;

border-radius: 20px;

padding: 50px;

padding-top: 100px;

display: flex;

flex-wrap: wrap;

justify-content: space-around;

}

#divcent>p{

width: 100%;

}

</style>

<script>

function copyText() {

var text = document.getElementById("textToCopy").innerText;

navigator.clipboard.writeText(text)

.then(function() {

alert("Текст успішно скопійовано!");

})

.catch(function(error) {

console.error("Не вдалося скопіювати текст: ", error);

});

}

</script>

{% if title is not none %}

<div id="divcent">

<div class="text-center" ><h3>Куплений квиток</h3></div>

<p >Номер квитка <span id="textToCopy" >{{ ticket.code }}</span> <a onclick="copyText()"><span style="font-size: 12px; color: grey;cursor:pointer ; ">Копіювати</span></a></p>

<p>Ваше ім'я

{% if non\_auth\_user is none %}

{{ user.name }}

{% else %}

{{ non\_auth\_user.name }}

{% endif %}

</p>

<p>Місце : {{ seat }}</p>

<p>Місто відправлення : {{ start\_city.name }}</p>

<p>Місто прибуття : {{ finish\_city.name }}</p>

<p>Час відправлення : {{ journey.time\_start }}</p>

<p>Час прибуття : {{ journey.time\_finish }}</p>

<p>Автобус : {{ bus.name }}</p>

<p>Номер автобуса : {{ bus.bus\_number }}</p>

</div>

{% else %}

<div class="text-center" style="margin-top: 100px">

<h2>Failed payment</h2>

{% endif %}

<div class="text-center">

<a class="btn btn-primary" href="/formbuyticket">Вибрати новий шлях</a>

<a class="btn btn-primary" href="/home">На головну</a>

</div>

</div>

{% endblock %}

**tablejourneys.html**

{% extends 'buyticket.html' %}

{% block form %}

<body>

<script>

document.addEventListener('DOMContentLoaded', function() {

var forms = document.querySelectorAll('form');

forms.forEach(function(form) {

form.addEventListener('submit', function(event) {

event.preventDefault();

var formData = new FormData(form);

fetch('/selectplace/', {

method: 'POST',

body: formData

})

.then(response => {

if (response.ok) {

return response.text();

} else {

console.error('Помилка: ' + response.status);

}

})

.then(html => {

document.body.innerHTML = html;

})

.catch(error => {

console.error('Помилка виконання запиту:', error);

});

});

});

});

</script>

{% if table is not none %}

<div class="bd-example-snippet bd-code-snippet">

<div class="bd-example m-0 border-0" style="margin-top: 50px !important;">

<table class="table table-striped">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">Автобус</th>

<th scope="col">Місто відправлення</th>

<th scope="col">Місто прибуття</th>

<th scope="col">Час відправлення</th>

<th scope="col">Час прибуття</th>

<th scope="col">Ціна</th>

<th scope="col">Замовити</th>

</tr>

</thead>

<tbody>

{% for row in table %}

<tr>

<form method="post">

<th scope="row">{{ loop.index }}</th>

<td>

{{ row.bus\_name }}

</td>

<td>

{{ row.start\_city }}

</td>

<td>

{{ row.finish\_city }}

</td>

<td>

{{ row.time\_start }}

</td>

<td>

{{ row.time\_finish }}

</td>

<td>

{{ row.cost }}

</td>

<td>

{% if non\_auth\_user is not none %}

<input type="hidden" name="phone" value="{{ non\_auth\_user.phone }}">

<input type="hidden" name="name" value="{{ non\_auth\_user.name }}">

{% else %}

<input type="hidden" name="phone" value="">

<input type="hidden" name="name" value="">

{% endif %}

<input type="hidden" name="id\_journey" value="{{ row.id\_journey }}">

<input type="hidden" name="bus\_name" value="{{ row.bus\_name }}">

<input type="hidden" name="start\_city" value="{{ row.start\_city }}">

<input type="hidden" name="finish\_city" value="{{ row.finish\_city }}">

<input type="hidden" name="time\_start" value="{{ row.time\_start }}">

<input type="hidden" name="time\_finish" value="{{ row.time\_finish }}">

<input type="hidden" name="cost" value="{{ row.cost }}">

<button type="submit" class="btn btn-primary" >Купити</button>

</td>

</form>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

{% endif %}

{% endblock %}

**update.html**

{% extends 'baseadmin.html' %}

{% block title %}

Update

{% endblock %}

{% block body %}

<script>

document.addEventListener('DOMContentLoaded', function () {

let radioButtons = document.querySelectorAll('input[name="listGroupCheckableRadios"]');

let labels = document.querySelectorAll('.list-group-item');

for (let i = 0; i < radioButtons.length; i++) {

radioButtons[i].addEventListener('change', function () {

for (let j = 0; j < radioButtons.length; j++) {

if (radioButtons[j].checked) {

labels[j].classList.add("active");

labels[j].classList.remove("non-active");

} else {

labels[j].classList.remove("active");

labels[j].classList.add("non-active");

}

}

});

}

});

</script>

<form method="post">

<div class="d-flex flex-column flex-md-row p-4 gap-4 py-md-5 align-items-center justify-content-center">

<div class="list-group list-group-checkable d-grid gap-2 border-0">

<h2>Виберіть що хочете оновити</h2>

<input style="display: none;" class="list-group-item-check pe-none " type="radio" name="listGroupCheckableRadios" id="listGroupCheckableRadios1" value="bus" checked="">

<label class="list-group-item rounded-3 py-3 active" for="listGroupCheckableRadios1">

Автобус

</label>

<input style="display: none;" class="list-group-item-check pe-none" type="radio" name="listGroupCheckableRadios" id="listGroupCheckableRadios2" value="journey">

<label class="list-group-item rounded-3 py-3" for="listGroupCheckableRadios2">

Подорож

</label>

<input style="display: none;" class="list-group-item-check pe-none" type="radio" name="listGroupCheckableRadios" id="listGroupCheckableRadios3" value="city">

<label class="list-group-item rounded-3 py-3" for="listGroupCheckableRadios3">

Місто

</label>

<input style="display: none;" class="list-group-item-check pe-none" type="radio" name="listGroupCheckableRadios" id="listGroupCheckableRadios4" value="ticket">

<label class="list-group-item rounded-3 py-3" for="listGroupCheckableRadios4">

Квиток

</label>

</div>

<button type="submit" class="btn btn-primary">Оновити</button>

<a href="/adminpanel" class="btn btn-primary">Назад</a>

</div>

</form>

{% endblock %}

**updatepost.html**

{% extends 'baseadmin.html' %}

{% block title %}

Оновити запис

{% endblock %}

{% block body %}

<script>

document.addEventListener('DOMContentLoaded', function () {

const fields = {

'id\_bus': 'int',

'name': 'str',

'seats': 'int',

'seats\_occupied': 'int',

'bus\_number': 'str',

'id\_city': 'int',

'country': 'str',

'id\_client': 'int',

'phone': 'str',

'email': 'str',

'password': 'str',

'role\_client': 'str',

'id\_ticket': 'int',

'id\_journey': 'int',

'seat': 'int',

'date\_buying': 'datetime',

'bus\_id': 'int',

'city\_start\_id': 'int',

'city\_finish\_id': 'int',

'cost': 'int',

'time\_start': 'datetime',

'time\_finish': 'datetime'

};

const selectedField = '{{ selectedfield }}';

const form = document.getElementsByTagName('form')[0];

console.log("field"+ selectedField);

let fieldType = fields[selectedField.replace(/["']/g, "")];

let newInput=document.createElement('input');

if (fieldType==='int'){

newInput.type='number';

}

else if (fieldType==='str'){

newInput.type='text';

}

else if (fieldType==='datetime'){

newInput.type='datetime-local';

}

newInput.name = 'value';

newInput.id = 'value';

let newLabel = document.createElement('label');

newLabel.for = 'value';

newLabel.innerHTML = "Значення поля";

let inp=document.createElement('input');

inp.type='hidden';

inp.id='type';

inp.name='type';

inp.value=fieldType;

let h2Element = document.querySelector('h2');

h2Element.insertAdjacentElement('afterend', newInput);

h2Element.insertAdjacentElement('afterend', newLabel);

h2Element.insertAdjacentElement('afterend', inp);

});

function redirectToRegister(url) {

window.location.href = url;

}

</script>

{% if info is none %}

<form method="post">

<h2>Оновлення запису</h2>

<label for="id">Введіть id запису</label>

<input type="text" name="id" id="id" required>

<button type="submit">Оновити</button>

</form>

{% else %}

<script>

alert('Не оновлено')

</script>

{% endif %}

{% endblock %}

**updatewhatrecord.html**

{% extends 'baseadmin.html' %}

{% block title %}

Оновити запис

{% endblock %}

{% block body %}

<script>

document.addEventListener('DOMContentLoaded', function () {

let radioButtons = document.querySelectorAll('input[name="listGroupCheckableRadios"]');

let labels = document.querySelectorAll('.list-group-item');

for (let i = 0; i < radioButtons.length; i++) {

radioButtons[i].addEventListener('change', function () {

for (let j = 0; j < radioButtons.length; j++) {

if (radioButtons[j].checked) {

labels[j].classList.add("active");

labels[j].classList.remove("non-active");

} else {

labels[j].classList.remove("active");

labels[j].classList.add("non-active");

}

}

});

}

});

function redirectToNext(event) {

event.preventDefault()

let selectedOption = document.querySelector('input[name="listGroupCheckableRadios"]:checked').value;

let clas="{{ clas.\_\_class\_\_.\_\_name\_\_ }}"

console.log(clas)

window.location.href = "/updatepost/" + selectedOption+"/"+clas;

}

</script>

<form method="post">

<div class="d-flex flex-column flex-md-row p-4 gap-4 py-md-5 align-items-center justify-content-center">

<div class="list-group list-group-checkable d-grid gap-2 border-0">

<h2>Що оновлюємо ?</h2>

{% for i in fieldlist %}

<input class="list-group-item-check pe-none " type="radio" name="listGroupCheckableRadios" id="{{ i }}" value="{{ i }}" style="display: none;">

<label class="list-group-item rounded-3 py-3 active" for="{{ i }}">{{ i }}</label><br>

{% endfor %}

<button type="submit" class="btn btn-primary" onclick="redirectToNext(event)">Оновити це</button>

<a href="/adminpanel" class="btn btn-primary">Назад</a>

</div>

</div>

</form>

{% endblock %}

**usertickets.html**

{% extends 'base.html' %}

{% block title %}

Мої квитки

{% endblock %}

{% block body %}

<style>

#flexmain {

margin-top: 100px;

display: flex;

flex-wrap: wrap;

justify-content: flex-start;

}

#flexmain>div {

background-color: #E9ECEF;

border-radius: 25px;

box-shadow: 5px 5px black;

padding: 35px;

margin-left: 20px;

margin-top: 20px;

}

#flexmain>div:hover {

background-color: #cfd0d1;

border-radius: 25px;

box-shadow: 6px 5px #0D6EFD;

}

#flexmain>div>\* {

margin-top: 10px;

}

</style>

{% if recenttickets is not none %}

<div id="flexmain">

{% for cardticket in recenttickets %}

<div>

<p>Код поїздки</p><p>{{ cardticket.code }}</p>

<p>Напрямок</p><p><strong>{{ cardticket.name }}</strong></p>

<p>Час поїздки</p><p><strong>{{ cardticket.time }}</strong></p>

<p>Вартість</p><p><strong>{{ cardticket.cost }}</strong></p>

<form action="/getticket" method="post">

<input type="hidden" name="ticket\_id" value="{{ cardticket.code }}">

<button type="submit" class="btn btn-primary">Дивитись деталі</button>

</form>

</div>

{% endfor %}

</div>

{% endif %}

{% if oldtickets is not none %}

<div id="flexmain">

{% for cardticket in oldtickets %}

<div style="background-color: #212529;color: white">

<p style="color: red;text-shadow: 1px 1px 1px white;">Дія білету завершилась</p>

<p>Код поїздки</p><p>{{ cardticket.code }}</p>

<p>Напрямок</p><p><strong>{{ cardticket.name }}</strong></p>

<p>Час</p><p><strong>{{ cardticket.time }}</strong></p>

<p>Вартість</p><p><strong>{{ cardticket.cost }}</strong></p>

</div>

{% endfor %}

</div>

{% else %}

<div style="margin-top: 100px" class="text-center">

<h3>Квитків не знайдено</h3><br>

</div>

{% endif %}

{% endblock %}

**db\_avtovokzal\_create\_script.sql**

--create database avtovokzalf;

use avtovokzalf;

create table buses (

id\_bus int not null identity(1,1) PRIMARY KEY,

name nvarchar(50) not null,

seats int not null,

bus\_number nvarchar(8) check (bus\_number like '[А-Я][А-Я][0-9][0-9][0-9][0-9][А-Я][А-Я]') unique

);

create table cities (

id\_city int not null identity(1,1) PRIMARY KEY,

name nvarchar(50) not null,

country nvarchar(50) null

);

create table clients (

id\_client int not null identity(1,1) PRIMARY KEY,

name nvarchar(50) not null,

phone nvarchar(11) check (phone like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]') not null unique default '',

email nvarchar(50) not null unique default '' check (email like '%@%.%'),

password nvarchar(50) not null,

role\_client nvarchar(50) not null default 'user'

);

CREATE TABLE non\_autorized\_users(

id\_user int PRIMARY KEY not null,

name nvarchar(20) not null,

phone nvarchar(11) not null unique check (phone like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]') default ''

);

CREATE TABLE timetable (

id\_journey int not null identity(1,1) PRIMARY KEY,

bus\_id int null,

city\_start\_id int not null,

city\_finish\_id int not null,

cost float not null,

time\_start datetime default GETDATE() not null,

time\_finish datetime default GETDATE() not null,

seats\_occupied int not null default 0,

is\_active bit null default 1,

FOREIGN KEY (bus\_id) REFERENCES buses(id\_bus),

FOREIGN KEY (city\_start\_id) REFERENCES cities(id\_city),

FOREIGN KEY (city\_finish\_id) REFERENCES cities(id\_city)

);

create table orders (

id\_order INT PRIMARY KEY IDENTITY(1,1),

client\_id INT NULL,

non\_authorized\_users\_id INT NULL,

date\_buying DATETIME DEFAULT GETDATE(),

cost float null,

whether\_paid bit null,

FOREIGN KEY (client\_id) REFERENCES clients(id\_client),

FOREIGN KEY (non\_authorized\_users\_id) REFERENCES non\_autorized\_users(id\_user)

);

create table tickets(

id\_ticket int not null identity(1,1) PRIMARY KEY,

bus\_id int not null,

seat int not null,

date\_buying datetime default GETDATE() null,

journey\_id int null,

non\_autorized\_users\_id int null,

client\_id int not null,

order\_id int not null,

code int null unique,

FOREIGN KEY (bus\_id) REFERENCES buses(id\_bus),

FOREIGN KEY (client\_id) REFERENCES clients(id\_client),

FOREIGN KEY (journey\_id) REFERENCES timetable(id\_journey),

FOREIGN KEY (order\_id) REFERENCES orders(id\_order),

FOREIGN KEY (non\_autorized\_users\_id) REFERENCES non\_autorized\_users(id\_user)

);

**Procedures.sql**

create procedure deleteClient

@id int

AS

BEGIN

delete from clients

where id\_client=@id

END;

create procedure deleteBus

@id int

AS

BEGIN

delete from buses

where id\_bus=@id

END;

create procedure deleteCity

@id int

AS

BEGIN

delete from cities

where id\_city=@id

END

create procedure deleteNonAthUser

@id int

AS

BEGIN

delete from non\_autorized\_users

where id\_user=@id

END

create procedure deleteOrder

@id int

AS

BEGIN

delete from orders

where id\_order=@id

END

create procedure deleteTicket

@id int

AS

BEGIN

delete from tickets

where id\_ticket=@id

END

create procedure deleteTimetable

@id int

AS

BEGIN

delete from timetable

where id\_journey=@id

END

--DROP PROCEDURE IF EXISTS get\_income\_per\_day

create procedure get\_income\_per\_day

as

begin

DECLARE @time\_now datetime,

@time\_start\_day datetime,

@time\_finish\_day datetime;

SET @time\_now = GETDATE();

SET @time\_start\_day = CONVERT(datetime, CONVERT(date, @time\_now));

SET @time\_finish\_day = DATEADD(second, -1, DATEADD(day, 1, CONVERT(datetime, CONVERT(date, @time\_now))));

SELECT @time\_now AS time\_now, @time\_start\_day AS time\_start\_day, @time\_finish\_day AS time\_finish\_day;

select Sum(orders.cost) as income\_per\_day from orders where date\_buying between @time\_start\_day and @time\_finish\_day and orders.whether\_paid=1

end

exec get\_income\_per\_day

--DROP PROCEDURE IF EXISTS get\_income\_per\_month

CREATE PROCEDURE get\_income\_per\_month

AS

BEGIN

DECLARE @time\_now datetime,

@time\_start\_month datetime,

@time\_finish\_month datetime;

SET @time\_now = GETDATE();

SET @time\_start\_month = DATEADD(month, DATEDIFF(month, 0, @time\_now), 0);

SET @time\_finish\_month = DATEADD(second, -1, DATEADD(month, 1, @time\_start\_month));

SELECT @time\_now AS time\_now, @time\_start\_month AS time\_start\_month, @time\_finish\_month AS time\_finish\_month;

SELECT SUM(orders.cost) as income\_per\_month

FROM orders

WHERE date\_buying BETWEEN @time\_start\_month AND @time\_finish\_month

AND orders.whether\_paid = 1;

END;

exec get\_income\_per\_month

--DROP PROCEDURE IF EXISTS get\_income\_per\_year

CREATE PROCEDURE get\_income\_per\_year

AS

BEGIN

DECLARE @time\_now datetime,

@time\_start\_year datetime,

@time\_finish\_year datetime;

SET @time\_now = GETDATE();

SET @time\_start\_year = DATEFROMPARTS(YEAR(@time\_now), 1, 1); -- Початок поточного року

SET @time\_finish\_year = DATEADD(second, -1, DATEADD(year, 1, @time\_start\_year)); -- Кінець поточного року

SELECT @time\_now AS time\_now, @time\_start\_year AS time\_start\_year, @time\_finish\_year AS time\_finish\_year;

SELECT SUM(orders.cost) as income\_per\_year

FROM orders

WHERE date\_buying BETWEEN @time\_start\_year AND @time\_finish\_year

AND orders.whether\_paid = 1;

END;

exec get\_income\_per\_year

--DROP PROCEDURE IF EXISTS get\_all\_tickets\_per\_day

CREATE PROCEDURE get\_all\_tickets\_per\_day

AS

BEGIN

DECLARE @time\_now datetime,

@time\_start\_day datetime,

@time\_finish\_day datetime;

SET @time\_now = GETDATE();

SET @time\_start\_day = CONVERT(datetime, CONVERT(date, @time\_now));

SET @time\_finish\_day = DATEADD(second, -1, DATEADD(day, 1, CONVERT(datetime, CONVERT(date, @time\_now))));

SELECT @time\_now AS time\_now, @time\_start\_day AS time\_start\_day, @time\_finish\_day AS time\_finish\_day;

SELECT COUNT(id\_ticket) as total\_tickets

FROM tickets

WHERE order\_id IS NOT NULL

AND order\_id <> '';

END;

exec get\_all\_tickets\_per\_day

--DROP PROCEDURE IF EXISTS get\_all\_tickets\_per\_month;

CREATE PROCEDURE get\_all\_tickets\_per\_month

AS

BEGIN

DECLARE @time\_now datetime,

@time\_start\_month datetime,

@time\_finish\_month datetime;

SET @time\_now = GETDATE();

SET @time\_start\_month = DATEFROMPARTS(YEAR(@time\_now), MONTH(@time\_now), 1); -- Початок поточного місяця

SET @time\_finish\_month = DATEADD(second, -1, DATEADD(month, 1, @time\_start\_month)); -- Кінець поточного місяця

SELECT @time\_now AS time\_now, @time\_start\_month AS time\_start\_month, @time\_finish\_month AS time\_finish\_month;

SELECT COUNT(id\_ticket) as total\_tickets

FROM tickets

WHERE order\_id IS NOT NULL

AND order\_id <> ''

AND date\_buying BETWEEN @time\_start\_month AND @time\_finish\_month;

END;

exec get\_all\_tickets\_per\_month

--DROP PROCEDURE IF EXISTS get\_all\_tickets\_per\_year;

CREATE PROCEDURE get\_all\_tickets\_per\_year

AS

BEGIN

DECLARE @time\_now datetime,

@time\_start\_year datetime,

@time\_finish\_year datetime;

SET @time\_now = GETDATE();

SET @time\_start\_year = DATEFROMPARTS(YEAR(@time\_now), 1, 1); -- Початок поточного року

SET @time\_finish\_year = DATEADD(second, -1, DATEADD(year, 1, @time\_start\_year)); -- Кінець поточного року

SELECT @time\_now AS time\_now, @time\_start\_year AS time\_start\_year, @time\_finish\_year AS time\_finish\_year;

SELECT COUNT(id\_ticket) as total\_tickets

FROM tickets

WHERE order\_id IS NOT NULL

AND order\_id <> ''

AND date\_buying BETWEEN @time\_start\_year AND @time\_finish\_year;

END;

exec get\_all\_tickets\_per\_year

--DROP PROCEDURE IF EXISTS delete\_all\_non\_active\_journeys;

CREATE PROCEDURE get\_all\_tickets\_per\_year

AS

BEGIN

DECLARE @time\_now datetime,

@time\_start\_year datetime,

@time\_finish\_year datetime;

SET @time\_now = GETDATE();

SET @time\_start\_year = DATEFROMPARTS(YEAR(@time\_now), 1, 1); -- Початок поточного року

SET @time\_finish\_year = DATEADD(second, -1, DATEADD(year, 1, @time\_start\_year)); -- Кінець поточного року

SELECT @time\_now AS time\_now, @time\_start\_year AS time\_start\_year, @time\_finish\_year AS time\_finish\_year;

SELECT COUNT(id\_ticket) as total\_tickets

FROM tickets

WHERE order\_id IS NOT NULL

AND order\_id <> ''

AND date\_buying BETWEEN @time\_start\_year AND @time\_finish\_year;

END;

exec get\_all\_tickets\_per\_year

**when\_delete\_bus.sql**

USE [avtovokzal]

GO

drop trigger[when\_delete\_bus]

go

create TRIGGER when\_delete\_bus

ON buses

INSTEAD OF DELETE

AS

BEGIN

PRINT 'Trigger fired';

DECLARE @id\_bus INT;

SELECT @id\_bus = id\_bus FROM deleted;

EXEC drop\_FK\_timetable\_buses @id\_bus;

delete from buses

where id\_bus=@id\_bus

END;

--0->>

create PROCEDURE drop\_FK\_timetable\_buses

@id\_bus INT

AS

BEGIN

BEGIN TRANSACTION;

BEGIN TRY

UPDATE timetable

SET bus\_id = NULL

WHERE bus\_id = @id\_bus;

print ('EXEC delete\_null\_journey start')

EXEC delete\_null\_journey;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

print ('rollback')

ROLLBACK TRANSACTION;

END CATCH;

END;

--0->>

create PROCEDURE delete\_null\_journey

AS

BEGIN

DELETE FROM timetable

WHERE bus\_id IS NULL;

print ('EXEC delete\_null\_journey end')

END;

**when\_delete\_journey.sql**

USE [avtovokzal]

GO

IF EXISTS (SELECT \* FROM sys.triggers WHERE name = 'when\_delete\_journey')

BEGIN

DROP TRIGGER when\_delete\_journey;

END

GO

CREATE TRIGGER when\_delete\_journey

ON timetable

INSTEAD OF DELETE

AS

BEGIN

PRINT 'Trigger fired';

DECLARE @id\_journey INT;

SELECT @id\_journey = id\_journey FROM deleted;

EXEC drop\_FK\_tickets\_timetable @id\_journey;

DELETE FROM timetable

WHERE id\_journey = @id\_journey;

END;

GO

IF EXISTS (SELECT \* FROM sys.procedures WHERE name = 'drop\_FK\_tickets\_timetable')

BEGIN

DROP PROCEDURE drop\_FK\_tickets\_timetable;

END

GO

CREATE PROCEDURE drop\_FK\_tickets\_timetable

@id\_journey INT

AS

BEGIN

BEGIN TRANSACTION;

BEGIN TRY

UPDATE tickets

SET journey\_id = NULL

WHERE journey\_id = @id\_journey;

PRINT 'EXEC delete\_null\_journey start';

EXEC delete\_null\_tickets;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

PRINT 'Rollback';

ROLLBACK TRANSACTION;

END CATCH;

END;

GO

IF EXISTS (SELECT \* FROM sys.procedures WHERE name = 'delete\_null\_tickets')

BEGIN

DROP PROCEDURE delete\_null\_tickets;

END

GO

CREATE PROCEDURE delete\_null\_tickets

AS

BEGIN

DELETE FROM tickets

WHERE journey\_id IS NULL;

PRINT 'EXEC delete\_null\_journey end';

END;

GO

**when\_delete\_ticket.sql**

USE [avtovokzal]

GO

IF EXISTS (SELECT \* FROM sys.triggers WHERE name = 'when\_delete\_ticket')

BEGIN

DROP TRIGGER when\_delete\_ticket;

END

GO

create TRIGGER when\_delete\_ticket

ON tickets

INSTEAD OF DELETE

AS

BEGIN

PRINT 'Trigger fired';

DECLARE @id\_ticket INT,@id\_order int;

SELECT @id\_ticket = id\_ticket FROM deleted;

SELECT @id\_order = order\_id FROM deleted;

DELETE FROM tickets

WHERE id\_ticket = @id\_ticket;

exec delete\_order @id\_order

END;

GO

IF EXISTS (SELECT \* FROM sys.procedures WHERE name = 'delete\_order')

BEGIN

DROP PROCEDURE delete\_order;

END

GO

create procedure delete\_order

@id\_order int

as

begin

DELETE FROM orders

WHERE id\_order = @id\_order;

end