**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

**ФАКУЛЬТЕТ УПРАВЛІННЯ ФІНАНСАМИ ТА БІЗНЕСУ**

**Кафедра цифрової економіки та бізнес-аналітики**

**КУРСОВА РОБОТА**

**з дисциплін  професійної та практичної підготовки**

на тему:

Інформаційна система для управління рейсами літаків

**спеціальність:** 051 «Економіка»

**спеціалізація:** «Інформаційні технології в бізнесі»

**освітній ступінь:**  бакалавр

|  |  |  |
| --- | --- | --- |
| **Науковий керівник:**  к.ф.-м.н., доц. Депутат Б.Я.  \_\_\_\_\_\_\_\_\_\_\_“\_\_\_” травня 2023 р.  (підпис) |  | **Виконавець:**  Зосімчук М.Б.  \_\_\_\_\_\_\_\_\_\_\_\_ “\_\_\_”травня 2023 р.  (підпис) |

**Загальна кількість балів** \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис, ПІП членів комісії)

**ЛЬВІВ 2023**

ЗМІСТ

ВСТУП

Розділ 1 Аналіз вимог:

1.1. Постановка завдання;

1.2. Розробка моделі варіантів використання веб-сайту;

1.3. Аналіз засобів реалізації (техніко-економічне обґрунтування вибору);

Розділ 2 Розробка бази даних:

2.1 Опис моделі даних;

2.2 Нормалізація відношень;

2.3 Визначення типів даних;

2.4 Обмеження цілісності даних;

2.5 Реалізація SQL-скрипту;

Розділ 3 Розробка веб-додатку:

3.1 Структура веб-сайту;

3.2 Макети сторінок веб-сайту;

3.3 Програмування серверної частини;

3.4 Програмування клієнтської частини;

3.5 Розміщення веб-сайту на локальному віртуальному середовищі або в Інтернеті

ВИСНОВКИ

**ВСТУП**

Актуальність теми дослідження. Тема дослідження про інформаційну систему управління рейсами літаків є надзвичайно актуальною і має велике значення в сучасному світі авіації. Ось декілька аргументів, що підкреслюють актуальність цієї теми:

Зростання обсягу авіаперевезень: З кожним роком кількість пасажирів та вантажів, перевозених повітряним транспортом, зростає. Інформаційна система управління рейсами літаків стає важливою для ефективного планування, координації та моніторингу цього зростаючого обсягу перевезень.

Безпека повітряного руху: Забезпечення безпеки польотів є однією з найважливіших задач у авіаційній галузі. Інформаційна система управління рейсами літаків допомагає уникнути інцидентів і аварій шляхом відстеження положення літаків, контролю маршрутів та оптимізації повітряних просторів.

Економічна ефективність: Оптимізація управління рейсами літаків може принести значні економічні переваги. Зменшення затрат на паливо, скорочення часу польоту, підвищення продуктивності роботи літаків та аеропортів - це лише деякі з аспектів, які можна вдосконалити за допомогою інформаційної системи управління рейсами літаків.

Зменшення навантаження на персонал: Швидкий розвиток авіації створює значне навантаження на персонал, що займається плануванням та управлінням рейсами. Інформаційна система може автоматизувати багато процесів, зменшивши завантаження на персонал та допомагаючи йому зосередитися на вирішенні складних завдань.

Технологічні інновації: Розвиток технологій швидко змінює спосіб, яким управляються рейсами літаків. Впровадження сучасних інформаційних систем, заснованих на штучному інтелекті, аналітиці даних та хмарних технологіях, може принести багато нових можливостей для вдосконалення управління рейсами та підвищення ефективності.

Узагалі, актуальність теми дослідження для інформаційної системи управління рейсами літаків полягає в її потенційній здатності покращити безпеку, ефективність та економічну стійкість авіаційної індустрії.

**Метою та завданням курсової роботи** є створення інформаційної системи для управління рейсами літаків,що буде використовуватись для надання послуг в сфері авіарейсів та дозволить детально ознайомитись з білетами та при необхідності придбати їх.

Для досягнення мети в роботі поставлено й вирішено такі теоретичні та практичні завдання:

* розробка моделі варіантів використання веб-сайту;
* Аналіз засобів реалізації (техніко-економічне обґрунтування вибору);
* Розробити бази даних:описано моделі даних,поясненна нормалізація відношень;проаналізовані типи даних:представлена реалізація SQL-скрипту
* Розробити веб-сайт.

**Обєктом дослідження є** конретна авіалінія,для якої створюється інформаційна система.

**Предметом** є теоретичні, методичні та практичні аспекти розробкипрограмного забезпечення мовою розмітки та створення інтерфейсу: html, css; мовою програмування: JavaScript, мовою запитів MySql для створення фіційного веб-сайту.

**Використане програмне забезпечення**. Для досягнення результу використовувалось середовище розробки Visual Studio Code, та мови програмування: HTML, CSS, JS, MySQL.

**Структура роботи**. Курсова робота складається з трьох розділів

(«Аналіз вимог», «Розробка бази даних» та «Розробка веб-додатку»), висновків, списку використаних джерел.

**Розділ 1 Аналіз вимог**

* 1. **Постановка завдання**

У 21 столітті найпрактичнішим видом транспортного пересування є авіарейси. Така ситуація склалась через доступність в часі (економія часу), збільшення авіасполучень, покращення літальних апаратів, а також через цінову політику компанії. Популяризація цього виду транспортного пересування спонукає відкривати все більше компанії, які надають дані послуги. У світі існує близько 5000–5500 авіакомпаній, оскільки це кількість кодів авіакомпаній ICAO/IATA, які зараз поширюються. Однак, мабуть, більш логічною відповіддю є те, що в усьому світі існує від 770 до 800 комерційних авіакомпаній, які виконують регулярні рейси (офіційно визнані). Їх кількість швидко зростає, особливо в Китаї, де в середньому двічі на тиждень відкривається нова авіакомпанія. Решта складається з виключно чартерних авіакомпаній, приватних операторів (наприклад, заможних осіб), військових операторів, тощо.

Для кожних авіаліній пріоритеті є їхній клієнт. Тож як саме компанії доносять інформацію своїм клієнтам? Вони використовують такий інструмент, як веб сайт.

Існує велика кількість веб-сайтів авіаліній, кожен з яких має свої особливості та функціонал. Нижче наведено загальні головні функції, які часто присутні на таких веб-сайтах:

* Бронювання авіабілетів: Веб-сайти авіаліній надають можливість користувачам здійснювати пошук, порівнювати та бронювати авіабілети на різні напрямки. Користувачі можуть вибрати дату, пункт відправлення та призначення, клас обслуговування тощо.
* Управління бронюванням: Користувачі можуть переглядати, змінювати або скасовувати свої заброньовані авіабілети через веб-сайт авіалінії. Вони також можуть додавати додаткові послуги, такі як багаж, місця у салоні або перевізку тварин.
* Інформація про рейси: Веб-сайти надають інформацію про розклад рейсів, час вильоту та прибуття, термінали, можливі зміни в розкладі, а також статус рейсу в режимі реального часу. Це дозволяє пасажирам відстежувати свої рейси та отримувати оновлення про можливі зміни.
* Інформація про послуги та умови перевезення: На веб-сайтах авіаліній можна знайти інформацію про доступні послуги та умови перевезення, такі як правила багажу, документи, необхідні для подорожі, правила та обмеження щодо провезення різних предметів тощо.
* Програми лояльності та привілеї: Багато авіаліній мають програми лояльності, які надають додаткові переваги та привілеї постійним клієнтам. Ці програми зазвичай дозволяють заробляти бонуси або милі за кожен переліт, які можна обміняти на безкоштовні авіабілети або інші пільги.
* Контактна інформація та підтримка клієнтів: Веб-сайти авіаліній часто містять контактну інформацію та онлайн-форми зворотного зв'язку, щоб користувачі могли звернутися до підтримки клієнтів для отримання допомоги або відповідей на свої запитання.

Враховуючи глобальну природу авіаційної індустрії, кількість веб-сайтів авіаліній є дуже великою. Кожна авіакомпанія має свій власний веб-сайт, що представляє їхні послуги та бронювання авіабілетів. Тому точне число веб-сайтів авіаліній не може бути визначено, оскільки їх кількість постійно змінюється залежно від розмірів та кількості авіакомпаній, які діють у світі.

* 1. **Розробка моделі варіантів використання веб-сайту**

Розробка моделі варіантів використання веб-сайту авіалінії допоможе ідентифікувати ключові сценарії, які користувачі можуть мати при взаємодії з сайтом. Така модель допоможе покращити дизайн і функціональність сайту, забезпечити зручну навігацію та задоволення потреб користувачів. Нижче наведено загальні варіанти використання веб-сайту авіалінії:

Пошук та бронювання авіабілетів: Користувачі можуть використовувати сайт для пошуку та бронювання авіабілетів на різні напрямки. Вони можуть вказати дату, пункт відправлення та призначення, кількість пасажирів, вибрати зручний клас обслуговування та порівняти ціни різних варіантів.

Управління бронюванням: Користувачі можуть увійти до свого облікового запису на сайті та переглядати, змінювати або скасовувати свої заброньовані авіабілети. Вони також можуть додавати додаткові послуги, які необхідні під час подорожі.

Інформація про рейси: Сайт може надавати інформацію про розклад рейсів, статус рейсів у режимі реального часу, термінали відправлення та прибуття, можливі зміни в розкладі та іншу важливу інформацію, пов'язану з рейсами.

Сайт складається з двох частин, а саме інтерфейсу користувача та адміністративної частини. Користувальницький інтерфейс має бути зрозумілим та привабливим для користувачів.

Адміністратор має можливість повністю вести управління сайтом:

* Додавати та видаляти рейси
* Підтверджувати заявки клієнтів
* Відхиляти заявки клієнтів та інше.
  1. **Аналіз засобів реалізації (техніко-економічне обґрунтування вибору)**

Аналіз засобів реалізації та техніко-економічне обґрунтування вибору твідповідних інструментів є важливим етапом при розробці інформаційної системи для сайту продажу авіабілетів.

Для реалізації даної роботи було вибрано наступні технології:

* HTML,CSS(технології для реалізації Front-end частини)
* PHP,Mysql (Для реалізації Back-end частини)

Для Front-end розробки були використані такі інструменти:

* HTML (англ. HyperText Markup Language — мова розмітки гіпертексту) — стандартизована мова розмітки документів для перегляду вебсторінок у браузері. Браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора.
* CSS (англ. Cascading Style Sheets,) — спеціальна мова, яка використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних. Найбільш часто CSS використовують для візуальної презентації сторінок, написаних на HTML та XHTML, але формат CSS може застосовуватись і до інших видів XML-документів.

Для Back-end розробки були обрані такі інструменти:

* PHP,тому що ця мова має такі переваги:
* Простий та інтуїтивно зрозумілий синтаксис.
* Кросплатформність та гнучкість.
* Відмінна масштабованість.
* Вбудовуваність у HTML-документи.
* Активний розвиток та вдосконалення.
* Простий пошук рішень проблем, що виникають.
* Широкі перспективи її подальшого розвитку.
* Sql

Успіх мови SQL принесли наступні його особливості:

* Незалежність від конкретних СУБД;
* Переносимість з однієї обчислювальної системи на іншу;
* Наявність стандартів;
* Підтримка з боку компанії Microsoft (протокол ODBC);
* Реляційна основа;
* Головна міжнародна структура, що нагадує англійську мову;
* Можливість виконання спеціальних інтерактивних запитів;
* Забезпечення програмного доступу до баз даних;
* Повноцінність як мови, призначеного для роботи з базами даних;
* Можливість динамічного визначення даних;
* Підтримка архітектури клієнт / сервер.

**Розділ 2 Розробка бази даних**

**2.1. Опис моделі даних**

База даних — це певний набір даних, які пов'язані між собою спільною ознакою або властивістю, та впорядковані, наприклад, за алфавітом.

Об'єднання великої кількості даних в єдину базу дає змогу для формування безлічі варіації групування інформації — особисті дані клієнта, історія замовлень, каталог товарів та будь-що інше.

Головною перевагою БД є швидкість внесення та використання потрібної інформації. Завдяки спеціальним алгоритмам, які використовуються для баз даних, можна легко знаходити необхідні дані всього за декілька секунд. Також в базі даних існує певний взаємозв'язок інформації: зміна в одному рядку може спричинити зміни в інших рядках — це допомагає працювати з інформацією простіше і швидше.

Бази даних для сайтів дають змогу зберігати інформацію, що виглядає як зв'язані між собою таблиці. Саме в БД зберігаються вся необхідна та корисна інформація для функціонування сайту (клієнтські дані, прайс-лист, список товарів).

Щоб створити запит до бази даних часто використовують Structured Query Language. SQL дає змогу додавати, редагувати та видаляти інформацію, що міститься у таблицях.

Під час програмування сайтів використовують різні системи управління БД. До основних СУБД, відносять:

* об'єктно-реляційна система управління базами даних Oracle Database;
* вільна система управління базами даних PostgreSQL;
* система керування базами даних Microsoft SQL Сервер;
* вільна система управління базами даних MySQL;

Такі системи управління відрізняються централізованою обробкою запитів, забезпечують надійність, доступність та безпеку БД.

Найбільш популярною системою управління є MySQL, вона дає зручний доступ для управління БД та підтримує велику кількість таблиць різних типів.

При побудові таблиць баз даних важливо не суперечити інформації. Зазвичай для цього вводять ключові поля, які забезпечують унікальність кожного запису. База даних до сайту про який йде мова містить три таблиці: users; flight; flight type.

У базах даних використовуються різні типи зв'язків для встановлення взаємозв'язків між таблицями. Основні типи зв'язків в базах даних включають:

* Один-до-одного (One-to-One): В цьому типі зв'язку один запис в одній таблиці відповідає одному запису в іншій таблиці. Наприклад, у таблиці "Користувачі" може бути поле "ID паспорта", яке пов'язується з таблицею "Паспортні дані".
* Один-до-багатьох (One-to-Many): Цей тип зв'язку означає, що один запис в одній таблиці може мати відповідність до багатьох записів в іншій таблиці. Наприклад, в таблиці "Категорії товарів" одна категорія може мати багато товарів у таблиці "Товари".
* Багато-до-багатьох (Many-to-Many): У цьому типі зв'язку багато записів в одній таблиці можуть мати відповідність до багатьох записів в іншій таблиці. Це досягається за допомогою додаткової таблиці-зв'язку, яка містить зв'язки між записами обох таблиць. Наприклад, в таблиці "Студенти" і "Предмети" може бути додаткова таблиця "Реєстрація на курс", що вказує, які студенти зареєстровані на які курси.
* Самоспрямований (Self-Referencing): В цьому типі зв'язку записи в одній таблиці відносяться до інших записів в цій самій таблиці. Наприклад, в таблиці "Співробітники" може бути поле "Керівник", яке посилається на ID іншого співробітника в цій же таблиці.

Ці типи зв'язків допомагають організувати структуру бази даних та встановлювати залежності між таблицями для забезпечення цілісності даних та ефективної обробки інформації.

ER-діаграма (рис 2.1)

Таблиця flight складається з таких стовпців:id(INT),type\_id (INT),user\_id (INT),date\_flight (DATE) , status (INT).

Таблиця users складається з: ID (INT) , login (VARCHAR) , Name (VARCHAR) , Surname (VARCHAR) , Phone (VARCHAR) , Password (VARCHAR).

Таблиця flight\_type складається з: ID (INT) , Name (VARCHAR)

Існує кілька типів моделей даних, які використовуються для представлення структури та взаємозв'язків даних в базах даних. Основні типи моделей даних включають:

* Ієрархічна модель: Ця модель використовує деревоподібну структуру, де дані представлені у вигляді батьківських та дочірніх вузлів. Кожен вузол може мати лише одного батька, але декілька дочірніх вузлів. Ієрархічна модель часто використовується в системах керування базами даних, які були поширені в ранній історії баз даних.
* Мережева модель: Ця модель використовує графову структуру, де дані представлені у вигляді вузлів та зв'язків між ними. Кожен вузол може мати кілька зв'язків з іншими вузлами. Мережева модель надає більш гнучкий спосіб взаємозв'язку даних, але складніше використовувати та підтримувати.
* Реляційна модель: Ця модель є однією з найпоширеніших та найбільш використовуваних моделей даних. Вона базується на табличній структурі, де дані представлені у вигляді таблиць з рядками та стовпцями. Кожна таблиця представляє сутність, а кожен рядок в таблиці представляє окремий запис. Зв'язки між таблицями встановлюються за допомогою ключів. Реляційна модель є основою для систем керування базами даних (СКБД), таких як MySQL, PostgreSQL та Oracle.
* Об'єктно-орієнтована модель: Ця модель базується на концепціях об'єктно-орієнтованого програмування (ООП) і використовує об'єкти, класи та спадкування для представлення даних. Вона дозволяє моделювати складні структури даних з використанням поліморфізму, інкапсуляції та інших принципів ООП. Об'єктно-орієнтована модель часто використовується в системах, що вимагають більш гнучкого та розширюваного підходу до моделювання даних.
* Нереляційна модель (NoSQL): Цей тип моделі даних використовується для нереляційних баз даних, які зберігають дані без використання традиційної табличної структури. Нереляційні бази даних можуть використовувати різні моделі, такі як ключ-значення, документ, стовпчикова та графова модель. Вони зазвичай підходять для розподілених та масштабованих систем, де вимоги до швидкодії та гнучкості переважають над структурованістю даних.
* Ці типи моделей даних надають різні підходи до організації та роботи з даними в базах даних, і кожен з них має свої переваги та обмеження, що варто враховувати при виборі моделі для конкретного проекту.

**2.2. Нормалізація відношень**

Нормалізація є процесом організації даних у базі даних для досягнення ефективності, ефективного використання пам'яті та забезпечення цілісності даних. Вона допомагає уникнути аномалій даних, забезпечити консистентність та поліпшити продуктивність операцій з базою даних. Нормалізація відношень в базах даних зазвичай виконується згідно з правилами нормалізації, такими як нормальні форми.

Основні нормальні форми, які використовуються в процесі нормалізації, включають:

1. Перша нормальна форма (1NF): У цій нормальній формі вимагається, щоб кожна комірка таблиці містила атомарні значення, тобто неможливо розбити значення на більші елементи. Всі дублікати даних повинні бути видалені.
2. Друга нормальна форма (2NF): Ця нормальна форма вимагає, щоб кожний неключовий атрибут таблиці залежав від всього первинного ключа, а не від його частини. Якщо атрибути залежать від частини первинного ключа, їх можна винести в окрему таблицю.
3. Третя нормальна форма (3NF): В цій нормальній формі вимагається, щоб кожен неключовий атрибут таблиці залежав лише від первинного ключа, а не від інших неключових атрибутів. Якщо така залежність існує, атрибути можуть бути винесені в окрему таблицю.
4. Четверта нормальна форма (4NF): Ця нормальна форма стосується залежностей між мультивалентними атрибутами таблиці. Вона вимагає, щоб залежності між мультивалентними атрибутами та первинним ключем були розкриті та збережені.
5. Існують також більш високі нормальні форми, такі як п'ята нормальна форма (5NF) і шоста нормальна форма (6NF), які стосуються більш складних залежностей даних.

Процес нормалізації бази даних включає ідентифікацію залежностей даних, аналіз схеми бази даних та перенесення пов'язаних атрибутів у відповідні таблиці. Цей процес допомагає уникнути проблем зберігання та обробки даних у базі даних, забезпечуючи їх оптимальну організацію та доступність.

**2.3. Визначення типів даних.**

У SQL кожен стовпець (в таблиці) має власний тип даних, який обмежує те, що може зберігатися у стовпці. Наприклад, якщо тип даних стовпця — INT, то в цьому стовпці можуть зберігатися лише цілі числа, такі як 0, 1, -1 тощо.

MySQL має два основні типи даних: текстові, числові,

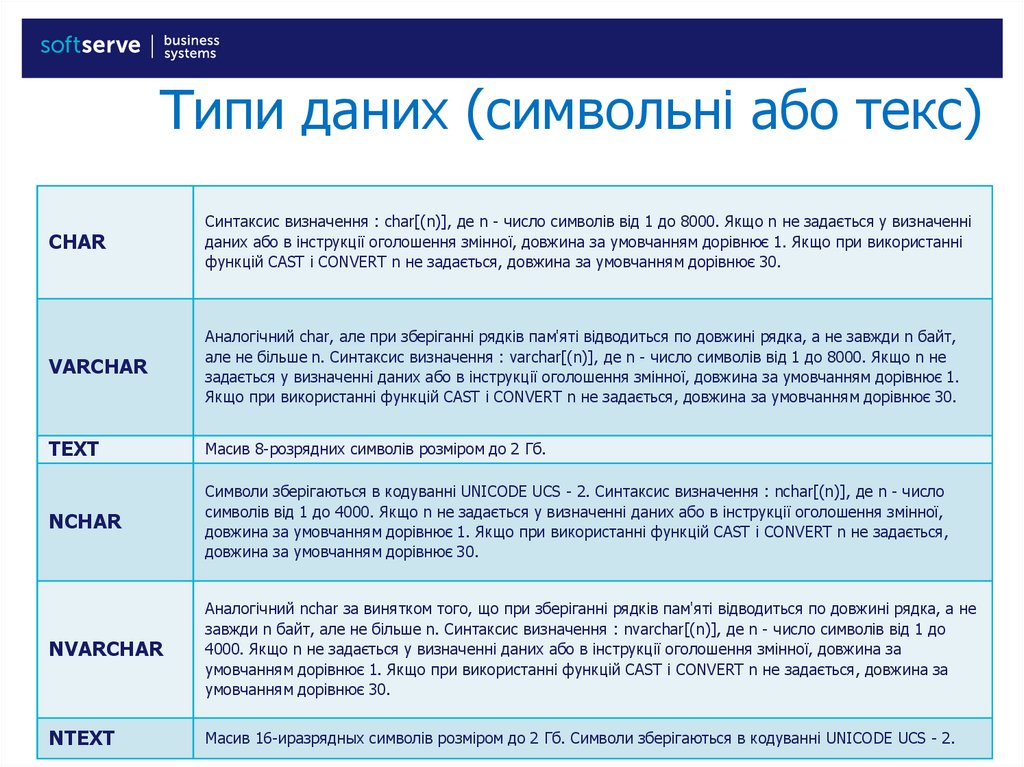


Рис. 1 Типи даних

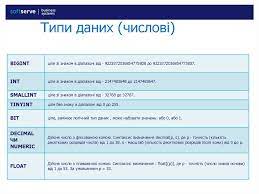


Рис. 2 Типи даних

**2.4 Обмеження цілісності даних.**

Обмеження цілісності даних є важливою частиною проектування бази даних і використовується для забезпечення правильності та цілісності даних в базі даних. Вони накладають правила та обмеження на дані, що вводяться або змінюються, щоб забезпечити консистентність та точність даних. Деякі з найпоширеніших обмежень цілісності даних включають:

* Обмеження первинного ключа (Primary Key Constraint): Вимагає, щоб значення в стовпці, визначеному як первинний ключ, були унікальними та не могли мати значення NULL. Вона гарантує, що кожен рядок в таблиці буде ідентифікований унікальним значенням.
* Обмеження унікальності (Unique Constraint): Вимагає, щоб значення в стовпці або комбінації стовпців були унікальними. Це дозволяє уникнути дублікатів у вказаних стовпцях.
* Обмеження зовнішнього ключа (Foreign Key Constraint): Встановлює зв'язок між двома таблицями, використовуючи значення стовпців. Вона забезпечує цілісність зв'язку даних між таблицями та забороняє вставку або зміну даних, якщо це порушує зв'язок.
* Обмеження перевірки (Check Constraint): Встановлює правила перевірки для значень у стовпці. Вона перевіряє, чи задовольняють значення у стовпці певне умовне виразу, і дозволяє приймати або відхиляти дані на основі цих умов.
* Обмеження нульового значення (Null Constraint): Встановлює, чи можуть стовпці містити значення NULL або чи потрібно задавати значення для стовпців.

Ці обмеження допомагають підтримувати цілісність та правильність даних у базі даних, захищають їх від некоректного вводу або зміни та допомагають забезпечити консистентність даних.

**2.5 Реалізація SQL-скрипту.**

SQL-скрипт є текстовим файлом або набором команд, написаних у мові SQL (Structured Query Language), які використовуються для взаємодії з базою даних. SQL-скрипти використовуються для створення, модифікації та управління базою даних.

SQL-скрипти можуть містити такі типи команд:

* Команди створення таблиць (CREATE TABLE): Використовуються для створення нових таблиць у базі даних. Ці команди визначають структуру таблиць, включаючи назви стовпців, типи даних та обмеження.
* Команди зміни таблиць (ALTER TABLE): Використовуються для модифікації вже існуючих таблиць, наприклад, для додавання, видалення або зміни стовпців, зміни типів даних, додавання обмежень тощо.
* Команди видалення таблиць (DROP TABLE): Використовуються для видалення таблиць з бази даних разом зі всіма даними та структурою.
* Команди вставки даних (INSERT INTO): Використовуються для додавання нових рядків даних до таблиці.
* Команди оновлення даних (UPDATE): Використовуються для зміни вже існуючих даних в таблиці.
* Команди видалення даних (DELETE FROM): Використовуються для видалення рядків даних з таблиці.
* Команди вибірки даних (SELECT): Використовуються для отримання даних з таблиці на основі певних умов.

SQL-скрипти можуть також містити коментарі, які допомагають розуміти та пояснювати роботу скрипту. Коментарі починаються з символу "--" або "/" і закінчуються символом "/".

При виконанні SQL-скрипта база даних виконує команди послідовно, забезпечуючи виконання зазначених дій з базою даних. SQL-скрипти зручні для автоматизації операцій з базою даних та забезпечення консистентності даних.

Потрібно створити функцію з'єднання з базою даних, вказати локальний

хост, ім’я та пароль від бази даних та саму назву бази даних.

$servername = '127.0.0.1:3306'; // сервер до якого підключаємось

$username = 'root'; // Логін користувача бази даних

$password = ''; // Пароль користувача бази даних

$dbname = 'avia'; // Назва бази даних, до якої ви хочете підключитися

// Підключення до бази даних

$conn = mysqli\_connect($servername, $username, $password, $dbname);

// Перевірка підключення

if (!$conn) {

die('Connection failed: ' . mysqli\_connect\_error());

}

Для реалізації інформаційної системи магазину одягу було використано

такі види запитів: «SELECT», «DELETE», «UPDATE», «INSERT», «JOIN»

Нижче наведено декілька прикладів таких запитів:

Запит SQL з використанням JOIN для об'єднання таблиць flight і flight\_type

$sql\_flights = "SELECT f.\*, u.phone AS user\_phone, u.name AS user\_name, ft.name AS flight\_type\_name

FROM flight f

JOIN users u ON f.user\_id = u.id

JOIN flight\_type ft ON f.type\_id = ft.id";

$result\_flights = $conn->query($sql\_flights);

if ($result\_flights->num\_rows > 0) {

while ($row\_flights = $result\_flights->fetch\_assoc()) {

if($row\_flights["status"] == '1') {

$status = 'green';

$status\_button = 'Скасувати';

$status\_action = 'cancel';

}else {

$status = 'red';

$status\_button = 'Підтвердити';

$status\_action = 'approve';

Перевірка, чи існує користувач з таким логіном

$check\_user\_query = "SELECT \* FROM users WHERE login = '". $login ."' AND password = '". $password ."'";

$check\_user\_result = $conn->query($check\_user\_query);

if ($check\_user\_result->num\_rows > 0) {

$user\_row = $check\_user\_result->fetch\_assoc();

$user\_id = $user\_row["id"];

$\_SESSION["id"] = $user\_id;

if($login == 'admin'){

header("Location: admin.php?id=". $\_SESSION["id"]);

} else {

header("Location: user.php?id=". $\_SESSION["id"]);

}

} else {

$error\_text = "Помилка входу<br><br>";

}

}

Вставка нового користувача в базу даних

$insert\_user\_query = "INSERT INTO users (login, name, surname, phone, password) VALUES ('". $login ."', '". $name ."', '". $surname ."', '". $phone ."', '". $password ."')";

session\_start();

if ($conn->query($insert\_user\_query) === TRUE) {

$user\_id\_query = "SELECT \* FROM users WHERE login = '". $login ."' AND password = '". $password ."'";

$user\_id\_result = $conn->query($user\_id\_query);

if ($user\_id\_result->num\_rows > 0) {// Отримання ID користувача

$user\_row = $user\_id\_result->fetch\_assoc();

$user\_id = $user\_row["id"];

$\_SESSION["id"] = $user\_id;

header("Location: user.php?id=". $\_SESSION["id"]);

}

} else {

$error\_text = "Помилка реєстрації<br><br>";

}

}

}

Отже,за допомогою цих частин коду ми реалізували певний функціонал для сайту.

**РОЗДІЛ 3 Розробка веб-додатку.**

**3.1 Структура веб-сайту.**

Структура веб-сайту визначає організацію та ієрархію сторінок, що складають сайт, а також способи навігації між ними. Вона грає важливу роль у забезпеченні зручного та логічного користувацького досвіду. Основні компоненти структури веб-сайту включають:

* Головна сторінка (Home Page): Це основна сторінка сайту, на якій зазвичай розміщена загальна інформація про компанію, продукти або послуги, а також посилання на інші ключові сторінки.
* Навігаційне меню: Це список посилань або вкладок, які допомагають користувачам переміщатися по різних сторінках сайту. Навігаційне меню зазвичай розміщується у верхній частині сайту або на бічній панелі.
* Розділи та підрозділи: Сайт може бути організований у вигляді розділів та підрозділів, де кожен розділ представляє певну тематику або категорію, а підрозділи - конкретні підтеми або підкатегорії.
* Контентні сторінки: Це сторінки, що містять інформацію, продукти, послуги, новини, блоги або будь-який інший контент, який цікавий для відвідувачів.
* Форми зворотного зв'язку: На сайті можуть бути розміщені форми, які дозволяють користувачам зв'язатися з власником сайту, заповнити форму замовлення або підписатися на розсилку.
* Статичні та динамічні елементи: Веб-сайт може містити статичні елементи, які залишаються постійними на всіх сторінках, такі як логотип, контактна інформація або посилання на соціальні мережі. Також можуть бути динамічні елементи, які змінюються в залежності від контексту, наприклад, слайдери, галереї, оновлювані блоки тощо.
* Посилання на зовнішні ресурси: Сайт може містити посилання на зовнішні ресурси, такі як соціальні мережі, партнери, документацію або інші корисні ресурси.

При проектуванні структури веб-сайту важливо враховувати потреби цільової аудиторії, забезпечувати легку навігацію, зручний пошук і доступ до важливої інформації.

**3.2 Макети сторінок веб-сайту.**

Макет сторінки веб-сайту (web page layout) визначає спосіб організації та розміщення елементів на сторінці. Макет відображає структуру сторінки, розмір блоків, розташування тексту, зображень, кнопок та інших елементів, що складають веб-сторінку.

Основні елементи макету сторінки веб-сайту включають:

* Заголовок (Header): Часто розміщується у верхній частині сторінки і містить логотип, назву сайту, навігаційне меню або додаткову інформацію, таку як контактні дані.
* Бокова панель (Sidebar): Розташовується збоку від основного вмісту і містить додаткову інформацію, посилання, рекламу, пошукову форму або інші важливі елементи.
* Основний вміст (Main Content): Це центральна частина сторінки, де розміщений основний контент, такий як тексти, зображення, відео, таблиці, форми тощо.
* Підвал (Footer): Розташовується у нижній частині сторінки і містить додаткову інформацію, посилання на важливі сторінки, контактні дані, посилання на соціальні мережі та іншу додаткову інформацію.

Макет сторінки може мати різні варіації, залежно від типу веб-сайту та його цілей. Наприклад, блог може мати макет з однією колонкою, де основний вміст відображається по порядку. Електронний магазин може мати макет з головним банером, боковими панелями для категорій товарів та центральною частиною для списку продуктів. Журнальний сайт може мати макет зі змістом, виокремленими блоками для статей та сайдбаром з додатковою інформацією.

Представлений сайт містить такі сторінки:

1. Сторінка входу та реєстрації

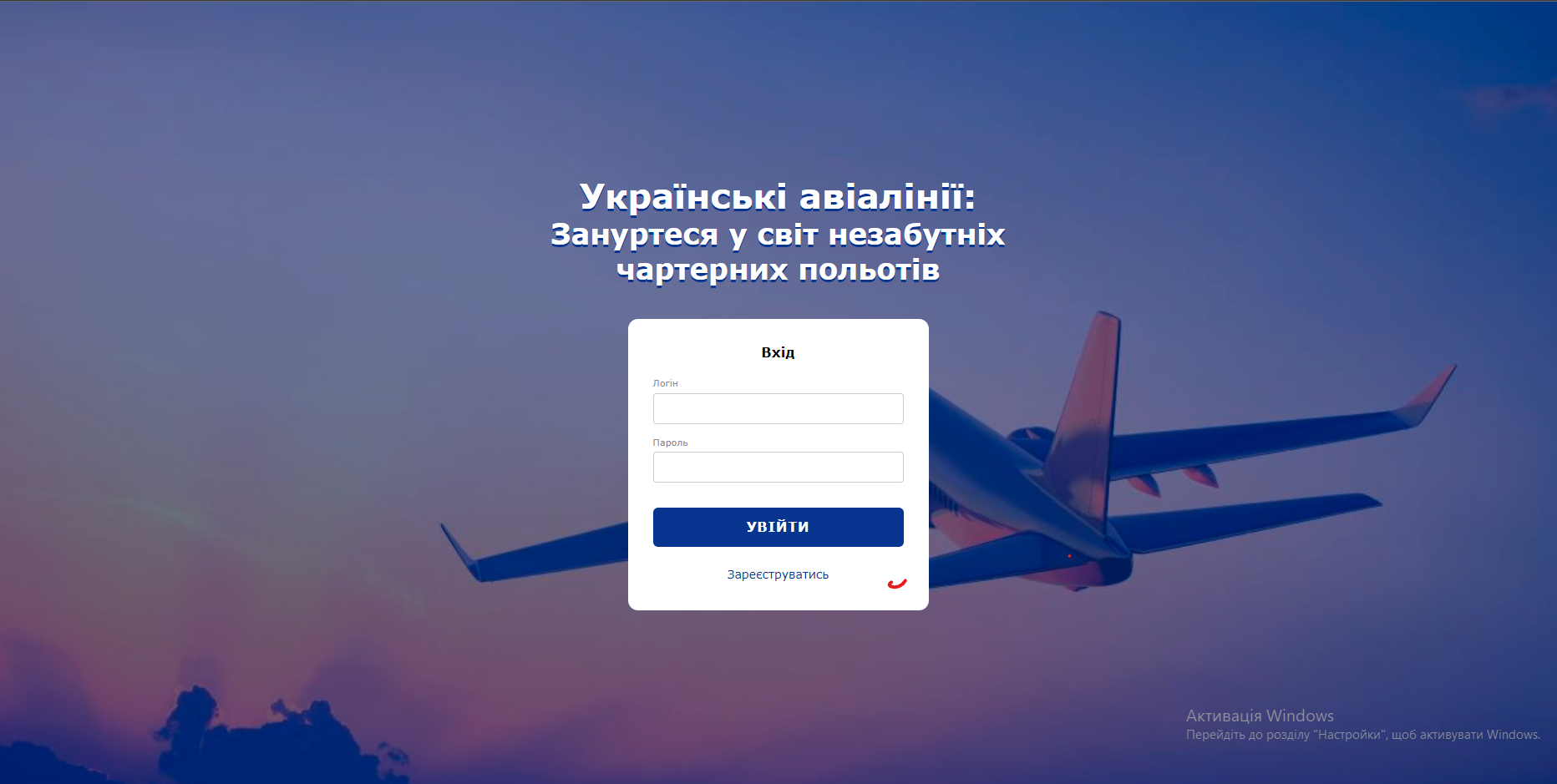


Рис. 3

1. Головна сторінка для користувача

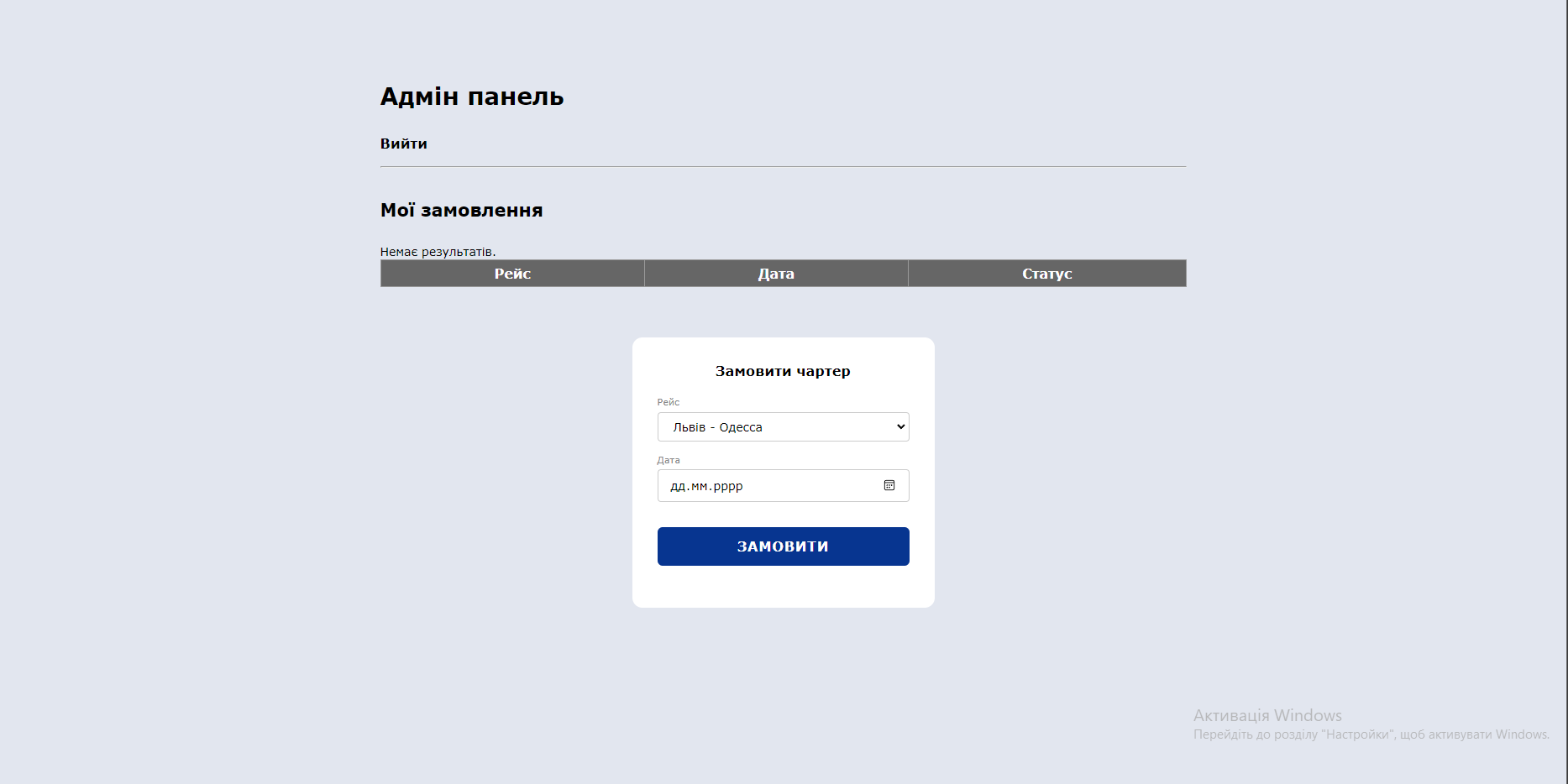


Рис. 4

1. Головна сторінка для адміністратора

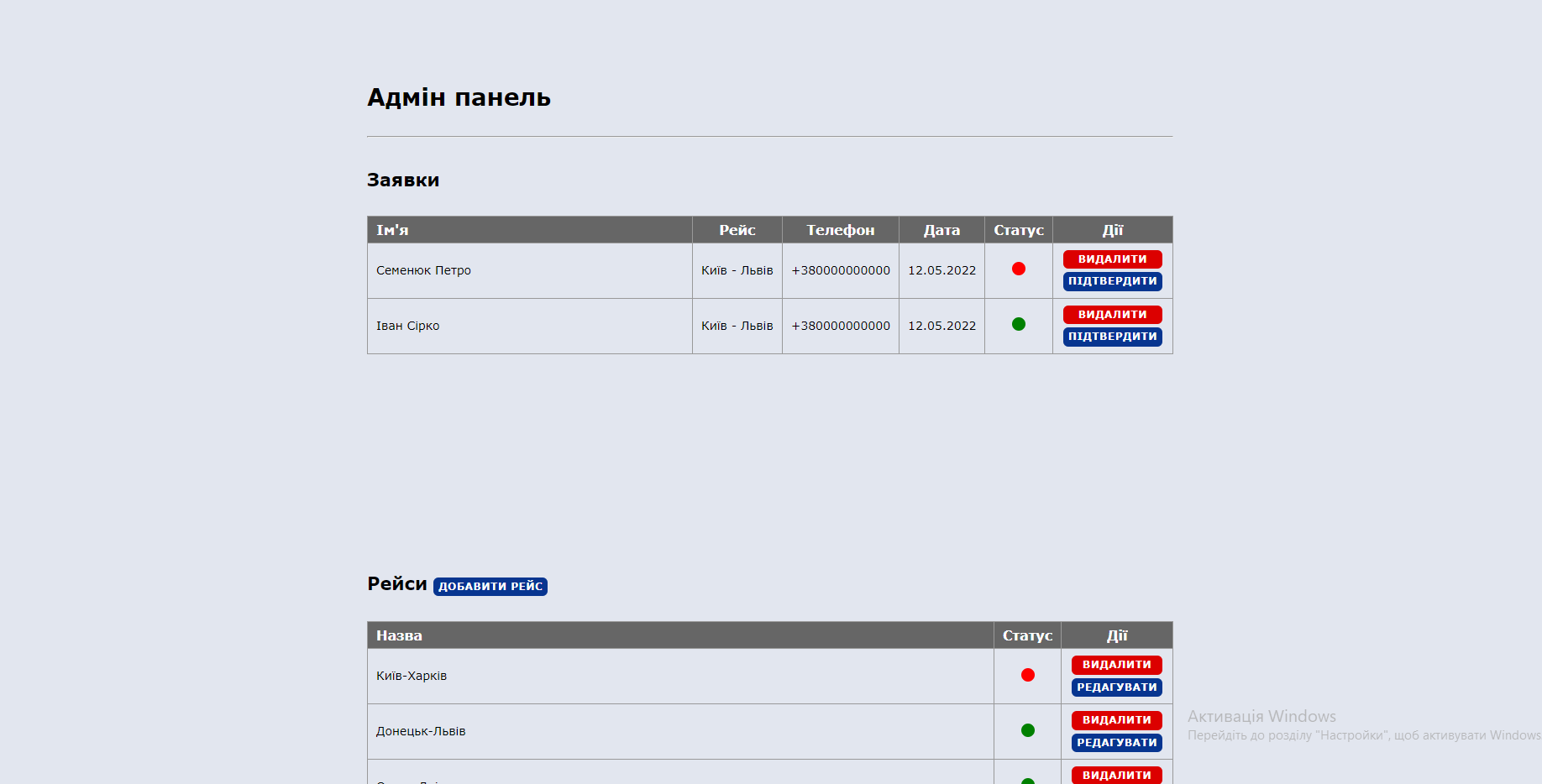


Рис. 5

**3.3 Програмування серверної частини.**

Програмування серверної частини веб-сайту відбувається на стороні сервера і включає розробку логіки, обробки запитів користувача, взаємодії з базою даних та надання необхідних даних та ресурсів клієнтській частині. Основні аспекти програмування серверної частини веб-сайту включають:

* Вибір мови програмування: Існує багато мов програмування, які можна використовувати для розробки серверної частини веб-сайту, такі як PHP, Python, Ruby, Java, C#, Node.js та інші. Вибір мови залежить від ваших потреб, знань та екосистеми інструментів.
* Веб-фреймворки: Використання веб-фреймворків спрощує розробку серверної частини, надаючи готові інструменти, шаблони та структуру проекту. Наприклад, для PHP є фреймворки Laravel, Symfony, для Python - Django, Flask, для Node.js - Express.js, для Ruby - Ruby on Rails та інші.
* Обробка запитів: Серверна частина отримує запити від клієнтської частини і оброблює їх. Це можуть бути HTTP-запити, які містять дані від користувача, запити на отримання або збереження даних у базі даних, запити на виконання певних дій тощо.
* Робота з базою даних: Серверна частина здійснює зв'язок з базою даних, виконує запити до бази даних (наприклад, SQL-запити) для отримання, збереження або оновлення даних.
* Аутентифікація та авторизація: Серверна частина може включати механізми аутентифікації користувачів, перевірку прав доступу та управління сесіями для забезпечення безпеки та обмеження доступу до певних ресурсів.
* Валідація та обробка даних: Серверна частина може проводити валідацію введених даних від користувача, перевірку на правильність та цілісність даних перед збереженням або використанням.
* Розробка API: Якщо ви плануєте надавати програмний інтерфейс (API) для взаємодії з вашим веб-сайтом, серверна частина реалізує логіку та механізми для обробки запитів API, форматування та повернення даних у відповідь.
* Оптимізація та масштабування: При розробці серверної частини важливо розглянути питання ефективності, швидкості відповіді та масштабування, особливо якщо очікується велике навантаження або зростання веб-сайту.

Програмування серверної частини веб-сайту вимагає розуміння логіки роботи, веб-протоколів, мов програмування та фреймворків, а також вміння ефективно працювати з базою даних та забезпечувати безпеку та надійність серверної інфраструктури.

**3.4 Програмування клієнтської частини.**

Програмування клієнтської частини веб-сайту відбувається на боці клієнта, тобто в браузері або на пристрої користувача. Це включає розробку функціональності, взаємодію з користувачем та візуальний дизайн. Основні технології, що використовуються для програмування клієнтської частини, включають:

* HTML (HyperText Markup Language): Використовується для створення структури та маркування контенту веб-сторінки, такого як заголовки, параграфи, списки, таблиці тощо.
* CSS (Cascading Style Sheets): Використовується для оформлення веб-сторінок, задання стилів, кольорів, шрифтів, розміщення елементів та створення візуального дизайну.
* JavaScript: Це скриптова мова програмування, яка дозволяє створювати динамічні ефекти, взаємодію з користувачем, анімацію, валідацію форм, запити до сервера та іншу функціональність на клієнтській стороні.
* Frameworks та бібліотеки JavaScript: Існує багато фреймворків та бібліотек, таких як React, Angular, Vue.js, які надають готові інструменти та шаблони для швидкої розробки клієнтської частини веб-сайту, спрощують роботу зі станом додатка, маршрутизацією та іншими аспектами.
* AJAX (Asynchronous JavaScript and XML): Використовується для взаємодії з сервером без перезавантаження всієї сторінки, що дозволяє оновлювати контент динамічно та асинхронно.
* Responsive Web Design: Техніки та підходи, які дозволяють адаптувати веб-сайт до різних пристроїв та розмірів екранів, забезпечуючи оптимальний користувацький досвід незалежно від пристрою.

Програмування клієнтської частини веб-сайту включає розробку інтерактивних елементів, валідацію форм, обробку подій, асинхронні запити до сервера, анімацію, взаємодію зі сторонніми сервісами та багато іншого. Важливим аспектом є забезпечення зручного та ефективного користувацького інтерфейсу, який би був зрозумілий та зручний для взаємодії користувача з веб-сайтом.

**3.5 Розміщення веб-сайту на локальному віртуальному середовищі або в Інтернеті.**

Розміщення веб-сайту може відбуватись на локальному віртуальному середовищі (локальному сервері) або в Інтернеті. Кожен з цих варіантів має свої особливості:

* Локальне віртуальне середовище (локальний сервер): Ви можете розмістити веб-сайт на своєму комп'ютері або внутрішньому сервері в локальній мережі. Це дає вам можливість розробляти та тестувати веб-сайт локально перед розгортанням в Інтернеті. Для цього можна використовувати локальний веб-сервер, такий як Apache, Nginx або Microsoft IIS, і встановити необхідне програмне забезпечення (наприклад, PHP, MySQL) для роботи веб-сайту. Локальне віртуальне середовище також дає можливість розробляти сайт без доступу до Інтернету або на стадії розробки.
* Розміщення в Інтернеті: Якщо ви бажаєте опублікувати веб-сайт для загального доступу, ви можете розмістити його на веб-сервері, що знаходиться в Інтернеті. Це вимагає придбання доменного імені (наприклад, example.com) та хостингу, який надає простір на сервері для розміщення веб-сайту. Ви можете вибрати спільний хостинг, де ваш веб-сайт буде розміщений на сервері разом з іншими сайтами, або використовувати приватний хостинг, де ви маєте окремий сервер або віртуальний приватний сервер (VPS). Ви також повинні завантажити файли вашого веб-сайту на сервер за допомогою протоколу FTP або іншого методу передачі файлів.
* Хмарні платформи: Існують хмарні платформи, такі як Amazon Web Services (AWS), Microsoft Azure, Google Cloud, які надають послуги хостингу та інфраструктури в Інтернеті. Вони дозволяють легко розгортати та масштабувати веб-сайти, надають готові сервіси для баз даних, кешування, масштабування та інші функції.

При розміщенні веб-сайту в Інтернеті важливо враховувати безпеку, надійність та швидкість роботи. Рекомендується регулярно оновлювати програмне забезпечення, налаштовувати правильні права доступу до файлів та баз даних, робити резервні копії та використовувати захисні механізми, такі як SSL-шифрування та захист від злому.

Вибір між локальним віртуальним середовищем і розміщенням в Інтернеті залежить від ваших потреб, обмежень та масштабу проекту.

ВИСНОВКИ

У процесі виконання було досягнуто всіх поставлених завдань:

* розробка моделі варіантів використання веб-сайту;
* Аналіз засобів реалізації (техніко-економічне обґрунтування вибору);
* Розробити бази даних:описано моделі даних,поясненна нормалізація відношень;проаналізовані типи даних:представлена реалізація SQL-скрипту
* Розробити веб-сайт.

За результатами проведеного дослідження можна зробити наступні висновки:

* З вище опрацьованих джерел використаної літератури та з власного

аналізу інформаційної системи, можна стверджувати про актуальність веб

сайту для авіаліній.

2) Особливість будь-якого веб-сайту полягає в його функціоналі в залежності від кінцевої мети

3) Правильно підібрані технологої для роботи значно полегшують

розробку веб-сайту.

**СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Веб-дизайн, структура сайту [Електронний ресурс]. – Режим доступу: https://webstudio2u.net/ua/design-web/403-structure-models.html .
2. 8.Типи даних MySQL [Електронний ресурс]. – Режим доступу: https://uk.wikibooks.org/wiki/SQL/%D0%A2%D0%B8%D0%BF%D0%B8\_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85\_MySQL.
3. Що таке база даних? Режим доступу: <https://apeps.kpi.ua/shco-take-basa-danykh>
4. Нормалізація. Поняття та необхідність застосування. Приклади. Режим доступу: [https://www.bestprog.net/uk/2020/10/17/databases-normalization- concept-and-necessity-of-application-modification-anomalies-examples-ua/](https://www.bestprog.net/uk/2020/10/17/databases-normalization-%20concept-and-necessity-of-application-modification-anomalies-examples-ua/)
5. PHP URL: <https://uk.wikipedia.org/wiki/PHP>
6. Моделі даних та їх класифікація. URL: <https://wiki.cuspu.edu.ua/index.php/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%96_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85_%D1%82%D0%B0_%D1%97%D1%85_%D0%BA%D0%BB%D0%B0%D1%81%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F>.
7. Фрейн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Б. Фрейн. – СПб.: Питер, 2014. – 304 с.

**Додатки**

**Додаток А**

**Лістинг адмінської частини сайту**

<?php

include('db.php');

session\_start();

if($\_SERVER["REQUEST\_METHOD"] == "GET" && isset($\_GET["delete"]) && isset($\_GET["flight\_id"])){

$flight\_id = $\_GET["flight\_id"];

$sql\_delete\_flight = "DELETE FROM flight WHERE id = '". $flight\_id ."'";

$result = mysqli\_query($conn, $sql\_delete\_flight);

if($result) {

header("Location: admin.php?id=". $\_SESSION["id"]);

}

}

if($\_SERVER["REQUEST\_METHOD"] == "GET" && isset($\_GET["approve"]) && isset($\_GET["flight\_id"])){

$flight\_id = $\_GET["flight\_id"];

$sql\_course = "UPDATE `flight` SET `status`='1' WHERE `id`=?";

$stmt = $conn->prepare($sql\_course);

if (!$stmt) {

echo "Помилка підготовки запиту: " . $conn->error;

exit();

}

// Прив'язка параметрів та виконання запиту

$stmt->bind\_param("s", $flight\_id);

if (!$stmt->execute()) {

echo "Помилка виконання запиту: " . $stmt->error;

exit();

}else{

header("Location: admin.php?id=" . $\_SESSION['id']);

}

}

if($\_SERVER["REQUEST\_METHOD"] == "GET" && isset($\_GET["cancel"]) && isset($\_GET["flight\_id"])){

$flight\_id = $\_GET["flight\_id"];

$sql\_course = "UPDATE `flight` SET `status`='0' WHERE `id`=?";

$stmt = $conn->prepare($sql\_course);

if (!$stmt) {

echo "Помилка підготовки запиту: " . $conn->error;

exit();

}

// Прив'язка параметрів та виконання запиту

$stmt->bind\_param("s", $flight\_id);

if (!$stmt->execute()) {

echo "Помилка виконання запиту: " . $stmt->error;

exit();

}else{

header("Location: admin.php?id=" . $\_SESSION['id']);

}

}

if($\_SERVER["REQUEST\_METHOD"] == "GET" && isset($\_GET["delete\_flight"]) && isset($\_GET["flight\_id"])){

$flight\_id = $\_GET["flight\_id"];

$sql\_delete\_flight\_type = "DELETE FROM flight\_type WHERE id = '". $flight\_id ."'";

$result\_flight\_type = mysqli\_query($conn, $sql\_delete\_flight\_type);

$sql\_delete\_flights = "DELETE FROM flight WHERE type\_id = '". $flight\_id ."'";

$result = mysqli\_query($conn, $sql\_delete\_flights);

if($result\_flight\_type) {

header("Location: admin.php?id=". $\_SESSION["id"]);

}

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0, user-scalable=no, user-scalable=0">

<meta name="description" content="Spy mailen">

<title>Особистий кабінет - Українські авіалінії</title>

<link href="./css/style.css" type="text/css" rel="stylesheet">

<script type="text/javascript" src="javascript/jquery-3.6.0.min.js"></script>

<script type="text/javascript" src="javascript/main.js"></script>

</head>

<body>

<div id="root">

<section class="admin-section">

<div class="wrapper admin-section-wrapper">

<h1 class="admin-section-title">Адмін панель</h1>

<h3><?php echo ('<a href="index.php?id='. $\_SESSION["id"] .'&exit=1" class="nav\_\_link">Вийти</a>');?></h3>

<hr>

<div class="admin-section-block">

<h2 class="admin-section-block-title">Заявки</h2>

<table class="main-table">

<thead>

<tr>

<th class="table-name">Ім'я</th>

<th class="table-flight">Рейс</th>

<th class="table-phone">Телефон</th>

<th class="table-date">Дата</th>

<th class="table-status">Статус</th>

<th class="table-action">Дії</th>

</tr>

</thead>

<tbody>

<?php

// Запит SQL з використанням JOIN для об'єднання таблиць flight і flight\_type

$sql\_flights = "SELECT f.\*, u.phone AS user\_phone, u.name AS user\_name, ft.name AS flight\_type\_name

FROM flight f

JOIN users u ON f.user\_id = u.id

JOIN flight\_type ft ON f.type\_id = ft.id";

$result\_flights = $conn->query($sql\_flights);

if ($result\_flights->num\_rows > 0) {

while ($row\_flights = $result\_flights->fetch\_assoc()) {

if($row\_flights["status"] == '1') {

$status = 'green';

$status\_button = 'Скасувати';

$status\_action = 'cancel';

}else {

$status = 'red';

$status\_button = 'Підтвердити';

$status\_action = 'approve';

}

echo('<tr>

<td class="table-name">'. $row\_flights["user\_name"] .'</td>

<td class="table-flight">'. $row\_flights["flight\_type\_name"] .'</td>

<td class="table-phone">'. $row\_flights["user\_phone"] .'</td>

<td class="table-date">'. $row\_flights["date\_flight"] .'</td>

<td class="table-status">

<div class="status status-'. $status .'"></div>

</td>

<td class="table-action">

<div class="table-action-list">

<a href="admin.php?delete=true&flight\_id='. $row\_flights["id"] .'" class="form-btn form-btn-small form-btn-remove table-action-btn">Видалити</a>

<a href="admin.php?'. $status\_action .'=true&flight\_id='. $row\_flights["id"] .'" class="form-btn form-btn-small table-action-btn">'. $status\_button .'</a>

</div>

</td>

</tr>');

}

} else {

echo "Немає результатів.";

}

?>

</tbody>

</table>

</div>

</div>

</section>

<section class="admin-section">

<div class="wrapper admin-section-wrapper">

<div class="admin-section-block">

<h2 class="admin-section-block-title">Рейси <a href="admin-add-flight.php" class="form-btn form-btn-small">Добавити рейс</a></h2>

<table class="main-table">

<thead>

<tr>

<th class="table-name">Назва</th>

<th class="table-action">Дії</th>

</tr>

</thead>

<tbody>

<?php

$sql\_flights\_type = "SELECT \* FROM flight\_type";

$result\_flights\_type = $conn->query($sql\_flights\_type);

if ($result\_flights\_type->num\_rows > 0) {

while ($row\_flights\_type = $result\_flights\_type->fetch\_assoc()) {

echo('<tr>

<td class="table-name">'. $row\_flights\_type["name"] .'</td>

<td class="table-action">

<div class="table-action-list">

<a href="admin.php?delete\_flight=true&flight\_id='. $row\_flights\_type["id"] .'" class="form-btn form-btn-small form-btn-remove table-action-btn">Видалити</a>

<a href="admin-edit-flight.php?flight\_id='. $row\_flights\_type["id"] .'" class="form-btn form-btn-small table-action-btn">Редагувати</a>

</div>

</td>

</tr>');

}

}

?>

</tbody>

</table>

</div>

</div>

</section>

</div>

</body>

</html>

**Додаток Б**

**Лістинг користувацької частини серверу**

<?php

include('db.php');

session\_start();

$user\_id = $\_SESSION["id"];

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST["flight"]) && isset($\_POST["date"])) {

$flight = $\_POST["flight"];

$date = $\_POST["date"];

$insert\_flight\_query = "INSERT INTO flight (type\_id, user\_id, date\_flight) VALUES ('". $flight ."', '". $user\_id ."', '". $date ."')";

if ($conn->query($insert\_flight\_query) === TRUE) {

header("Location: user.php?id=". $\_SESSION["id"]);

} else {

echo "Помилка замовлення: " . $conn->error;

}

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0, user-scalable=no, user-scalable=0">

<meta name="description" content="Spy mailen">

<title>Особистий кабінет - Українські авіалінії</title>

<link href="./css/style.css" type="text/css" rel="stylesheet">

<script type="text/javascript" src="javascript/jquery-3.6.0.min.js"></script>

<script type="text/javascript" src="javascript/main.js"></script>

</head>

<body>

<div id="root">

<section class="admin-section">

<div class="wrapper admin-section-wrapper">

<h1 class="admin-section-title">Адмін панель</h1>

<h3><?php echo ('<a href="index.php?id='. $\_SESSION["id"] .'&exit=1" class="nav\_\_link">Вийти</a>');?></h3>

<hr>

<div class="admin-section-block">

<h2 class="admin-section-block-title">Мої замовлення</h2>

<table class="main-table">

<thead>

<tr>

<th class="table-flight">Рейс</th>

<th class="table-date">Дата</th>

<th class="table-status">Статус</th>

</tr>

</thead>

<tbody>

<?php

// Запит SQL з використанням JOIN для об'єднання таблиць flight і flight\_type

$sql\_my\_flight = "SELECT f.\*, (SELECT name FROM flight\_type WHERE id = f.type\_id) AS flight\_type\_name

FROM flight f

WHERE f.user\_id = '".$user\_id."'";

$result\_my\_flight = $conn->query($sql\_my\_flight);

if ($result\_my\_flight->num\_rows > 0) {

while ($row\_my\_flight = $result\_my\_flight->fetch\_assoc()) {

if($row\_my\_flight["status"] == 1){

$status = 'green';

}else {

$status = 'red';

}

echo('<tr>

<td class="table-flight">'. $row\_my\_flight["flight\_type\_name"] .'</td>

<td class="table-date">'. $row\_my\_flight["date\_flight"] .'</td>

<td class="table-status">

<div class="status status-'. $status .'"></div>

</td>

</tr>');

}

} else {

echo "Немає результатів.";

}

?>

</tbody>

</table>

</div>

<div class="admin-section-block">

<form action="user.php?id=<?php echo($\_SESSION["id"]); ?>" method="POST" class="main-section-form">

<h3 class="main-form-title">Замовити чартер</h3>

<label class="form-label">

<p>Рейс</p>

<select name="flight" class="text-el">

<?php

$sql\_select = 'SELECT \* FROM flight\_type';

$result\_select = mysqli\_query($conn, $sql\_select);

// Обробка результату запиту

if (mysqli\_num\_rows($result\_select) > 0) {

// Виведення результату запиту

while($row\_select = mysqli\_fetch\_assoc($result\_select)) {

echo '<option value="'. $row\_select["id"] .'">'. $row\_select["name"] .'</option>';

}

} else {

echo "Результатів не знайдено";

}

?>

</select>

</label>

<label class="form-label">

<p>Дата</p>

<input type="date" name="date" class="text-el" required value="">

</label>

<button type="submit" class="form-btn form-btn-el">Замовити</button>

</form>

</div>

</div>

</section>

</div>

</body>

</html>

**Додаток С**

**Лістинг сторінки реєстрації**

<?php include('db.php');

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$login = $\_POST["login"];

$name = $\_POST["name"];

$surname = $\_POST["surname"];

$password = $\_POST["password"];

$phone = $\_POST["phone"];

// Логіка реєстрації користувача

// Перевірка, чи існує користувач з таким логіном

$check\_user\_query = "SELECT \* FROM users WHERE login = '". $login ."'";

$check\_user\_result = $conn->query($check\_user\_query);

if ($check\_user\_result->num\_rows > 0) {

echo "Користувач з таким логіном вже існує!";

} else {

// Вставка нового користувача в базу даних

$insert\_user\_query = "INSERT INTO users (login, name, surname, phone, password) VALUES ('". $login ."', '". $name ."', '". $surname ."', '". $phone ."', '". $password ."')";

session\_start();

if ($conn->query($insert\_user\_query) === TRUE) {

$user\_id\_query = "SELECT \* FROM users WHERE login = '". $login ."' AND password = '". $password ."'";

$user\_id\_result = $conn->query($user\_id\_query);

if ($user\_id\_result->num\_rows > 0) {// Отримання ID користувача

$user\_row = $user\_id\_result->fetch\_assoc();

$user\_id = $user\_row["id"];

$\_SESSION["id"] = $user\_id;

header("Location: user.php?id=". $\_SESSION["id"]);

}

} else {

$error\_text = "Помилка реєстрації<br><br>";

}

}

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0, user-scalable=no, user-scalable=0">

<meta name="description" content="Spy mailen">

<title>Реєстрація - Українські авіалінії</title>

<link href="./css/style.css" type="text/css" rel="stylesheet">

<script type="text/javascript" src="javascript/jquery-3.6.0.min.js"></script>

<script type="text/javascript" src="javascript/main.js"></script>

</head>

<body>

<div id="root">

<section class="main-section">

<div class="wrapper main-section-wrapper">

<h1 class="main-section-title"><span>Українські авіалінії:</span><br>Зануртеся у світ незабутніх чартерних польотів</h1>

<form action="#" method="POST" class="main-section-form">

<h3 class="main-form-title">Реєстрація</h3>

<?php echo $error\_text; ?>

<label class="form-label">

<p>Логін</p>

<input type="text" name="login" class="text-el" required value="">

</label>

<label class="form-label">

<p>Пароль</p>

<input type="password" name="password" class="text-el" required value="">

</label>

<label class="form-label">

<p>Ім'я</p>

<input type="text" name="name" class="text-el" required value="">

</label>

<label class="form-label">

<p>Прізвище</p>

<input type="text" name="surname" class="text-el" required value="">

</label>

<label class="form-label">

<p>Телефон для підтвердження заявки</p>

<input type="number" name="phone" class="text-el" required value="">

</label>

<button type="submit" class="form-btn form-btn-el">Зареєструватись</button>

<footer class="form-footer">

<a href="index.php">Вхід</a>

</footer>

</form>

</div>

</section>

</div>

</body>

</html>

**Додаток Д**

**Лістинг створення та заповнення бази даних**

Структура таблиці `flight`

--

CREATE TABLE `flight` (

`id` int NOT NULL,

`type\_id` int NOT NULL,

`user\_id` int NOT NULL,

`date\_flight` date NOT NULL,

`status` int NOT NULL DEFAULT '0'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

--

-- Дамп даних таблиці `flight`

--

INSERT INTO `flight` (`id`, `type\_id`, `user\_id`, `date\_flight`, `status`) VALUES

(4, 2, 2, '2023-05-12', 1),

(5, 11, 3, '2023-05-21', 0),

(6, 7, 3, '2023-06-11', 0),

(7, 2, 4, '2023-06-02', 1),

(8, 14, 4, '2023-06-11', 0),

(9, 12, 5, '2023-06-08', 0),

(10, 7, 5, '2023-06-07', 1);

-- --------------------------------------------------------

--

-- Структура таблиці `flight\_type`

--

CREATE TABLE `flight\_type` (

`id` int NOT NULL,

`name` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

--

-- Дамп даних таблиці `flight\_type`

--

INSERT INTO `flight\_type` (`id`, `name`) VALUES

(2, 'Львів - Одесса'),

(5, 'Харків - Одесса'),

(7, 'Київ - Дніпро'),

(8, 'Харків - Одесса'),

(10, 'Дніпро - Львів'),

(11, 'Дніпро - Харків'),

(12, 'Київ - Запоріжжя'),

(13, 'Львів - Запоріжжя'),

(14, 'Харків - Запоріжжя');

-- --------------------------------------------------------

--

-- Структура таблиці `users`

--

CREATE TABLE `users` (

`id` int NOT NULL,

`login` varchar(50) NOT NULL,

`name` varchar(50) NOT NULL,

`surname` varchar(50) NOT NULL,

`phone` varchar(15) NOT NULL,

`password` varchar(30) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

--

-- Дамп даних таблиці `users`

--

INSERT INTO `users` (`id`, `login`, `name`, `surname`, `phone`, `password`) VALUES

(1, 'admin', 'admin', 'admin', 'none', '1234'),

(2, 'Bohdan', 'Bohdan', 'Boiko', '+380983737885', '1234'),

(3, 'vasyl', 'Василь', 'Бондаренко', '+380953232655', '1234'),

(4, 'olena', 'Olena', 'Vakula', '+380985252568', '1234'),

(5, 'maria', 'Марія', 'Гончаренко', '+380505478354', '1234');

--

-- Індекси збережених таблиць

--

--

-- Індекси таблиці `flight`

--

ALTER TABLE `flight`

ADD PRIMARY KEY (`id`);

--

-- Індекси таблиці `flight\_type`

--

ALTER TABLE `flight\_type`

ADD PRIMARY KEY (`id`);

--

-- Індекси таблиці `users`

--

ALTER TABLE `users`

ADD PRIMARY KEY (`id`);

--

-- AUTO\_INCREMENT для збережених таблиць

--

--

-- AUTO\_INCREMENT для таблиці `flight`

--

ALTER TABLE `flight`

MODIFY `id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=11;

--

-- AUTO\_INCREMENT для таблиці `flight\_type`

--

ALTER TABLE `flight\_type`

MODIFY `id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=15;

--

-- AUTO\_INCREMENT для таблиці `users`

--

ALTER TABLE `users`

MODIFY `id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=7;

COMMIT;