Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра: 806 «Прикладная математика и информатика»

Дисциплина: «Базы данных»

Итоговый проект

Тема: «Сервис бронирования авиабилетов»

Выполнил студент группы М8О-308Б-22

Немкова Анастасия Романовна _____
(подпись, дата)

Проверил и принял

Доцент, Киндинова В. В. _____
(подпись, дата)

с оценкой _____

Москва, 2024

Оглавление

Постановка задачи

Общие требования к итоговому проекту:

1. При реализации курсового проекта допускается только использование СУБД PostgreSQL.

2. Необходимо выбрать предметную область для создания базы данных. Выбранная предметная область должна быть уникальной для всего потока, а не только в рамках учебной группы.

3. Необходимо описать модели предметной области и уровня инфраструктуры и их назначение в рамках реализуемого проекта (минимальное количество моделей предметной области и уровня инфраструктуры - 5). Также необходимо выполнить проектирование логической структуры базы данных. Все таблицы, связанные с описанными моделями предметной области, должны находиться в 3NF или выше. База данных должна иметь минимум 7 таблиц.

4. Клиентское приложение должно быть в виде WEB или оконного приложения.

5. Необходимо организовать различные роли пользователей и права доступа к данным (например: администратор,редактор, рядовой пользователь). Клиентское приложение, взаимодействующее с базой данных, должно предоставлять функционал для авторизации пользователя по логину и паролю (хранение непосредственно пароля в базе данных запрещено, надо хранить HASH от этого пароля и проверять его).

6. При разработке функционала базы данных следует организовать логику обработки данных не на стороне клиента (Frontend), а на стороне серверного приложения (Backend). Все обработки «SQL запросов», «работа бизнес-логики должны находится в BACKEND части. FRONT только для отображения.

7. Запросы должны быть асинхронны. То есть, при нажатии на форму она не должна зависать. При нажатии на форму одним пользователем, другой должен иметь возможность свободно пользоваться приложением. То есть действия разных пользователей независимы.

8. Необходимо реализовать возможность создания администратором архивных копий базы данных и восстановления данных из клиентского приложения.

9. Передача параметров в SQL запрос должна происходить только через parameters.

Описание

Проект представляет собой веб-сервис для бронирования авиабилетов, который позволяет пользователям искать доступные рейсы, бронировать места, а также оплачивать бронирования. Пользователи могут зарегистрироваться в системе, авторизоваться и управлять своими бронированиями через понятный интерфейс. Система обеспечивает создание и управление рейсами, проверку доступных мест на самолетах, а также обработку платежей. Администраторы могут управлять базой данных и создавать резервные копии для обеспечения надежности работы.

Вся информация о пользователях, рейсах, бронированиях и платежах хранится в базе данных PostgreSQL. Веб-интерфейс для работы с сервисами реализован с использованием Streamlit. Для взаимодействия с базой данных используется psycopg2.

Проект включает в себя сервисы для авторизации и регистрации пользователей, обработки бронирований, управления платежами, а также сервис для администрирования базы данных, чтобы обеспечить безопасность и надежность работы системы.

Описание и схема моделей предметной области

Модель базы данных сервиса для бронирования авиабилетов представляет собой структуру для хранения информации о рейсах, самолетах, аэропортах, пользователях, бронированиях и платежах. Эта система предназначена для работы с перелетами, позволяя пользователям бронировать билеты, оплачивать их и получать информацию о рейсах. Модель включает шесть основных таблиц, каждая из которых отвечает за хранение соответствующей информации:

1. airports — информация об аэропортах.
2. airplanes — информация о самолетах.
3. users — информация о пользователях системы.
4. flights — информация о рейсах.
5. bookings — информация о бронированиях билетов.
6. payments — информация о платежах за бронирования.

airports (Аэропорты)

Эта таблица хранит информацию о каждом аэропорте, включая его уникальный идентификатор, название, код ИАТА и город расположения. Она используется для указания аэропортов отправления и прибытия в других таблицах.

Основные столбцы:

1. airport_id: Уникальный идентификатор аэропорта.
2. name: Название аэропорта.
3. iata_code: Код ИАТА аэропорта (три символа).
4. city: Город расположения аэропорта.

Связи:

Связана с таблицей flights через столбцы departure_airport_id и arrival_airport_id, чтобы указать аэропорты отправления и прибытия для рейсов.

```
CREATE TABLE airports (
    airport_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    iata_code VARCHAR(3) NOT NULL,
    city VARCHAR(50) NOT NULL
);
```

airplanes (Самолеты)

Эта таблица содержит информацию о моделях самолетов, которые используются для рейсов, а также количество доступных мест для бронирования.

Основные столбцы:

1. airplane_id: Уникальный идентификатор самолета.
2. model: Модель самолета.
3. available_tickets: Количество доступных мест на борту самолета.

Связи:

Связана с таблицей flights через столбец airplane_id, который указывает, какой самолет используется на определенном рейсе.

```
CREATE TABLE airplanes (
    airplane_id SERIAL PRIMARY KEY,
    model VARCHAR(100) NOT NULL,
    available_tickets INT NOT NULL CHECK (available_tickets >= 0)
);
```

users (Пользователи)

Таблица хранит информацию о пользователях, которые могут просматривать рейсы и бронировать и покупать билеты.

Основные столбцы:

1. user_id: Уникальный идентификатор пользователя.
2. first_name: Имя пользователя.
3. last_name: Фамилия пользователя.
4. email: Электронная почта пользователя (уникальная).
5. password_hash: Хеш пароля пользователя.
6. role: Роль пользователя ('user' или 'admin').

Связи:

Связана с таблицей bookings через столбец user_id, что позволяет отслеживать бронирования, сделанные конкретными пользователями.

```
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    role VARCHAR(50) NOT NULL CHECK (role IN ('user', 'admin'))
);
```

flights (Рейсы)

Таблица содержит информацию о рейсах, такую как аэропорты отправления и прибытия, время и дата рейса, а также цена билета.

Основные столбцы:

1. flight_id: Уникальный идентификатор рейса.

2. departure_airport_id: Идентификатор аэропорта отправления.
3. arrival_airport_id: Идентификатор аэропорта прибытия.
4. airplane_id: Идентификатор самолета, который выполняет рейс.
5. arrival_date, departure_date: Даты прибытия и отправления.
6. arrival_time, departure_time: Время прибытия и отправления.
7. ticket_price: Стоимость билета.

Связи:

1. Связана с таблицей airports через столбцы departure_airport_id и arrival_airport_id, чтобы указать аэропорты отправления и прибытия для каждого рейса.
2. Связана с таблицей airplanes через столбец airplane_id, чтобы указать, какой самолет используется для рейса.
3. Связана с таблицей bookings через столбец flight_id, чтобы отслеживать, какие рейсы были забронированы пользователями.

```
CREATE TABLE flights (
    flight_id SERIAL PRIMARY KEY,
    departure_airport_id INT NOT NULL,
    arrival_airport_id INT NOT NULL,
    airplane_id INT NOT NULL,
    arrival_date DATE NOT NULL,
    departure_date DATE NOT NULL,
    arrival_time TIME NOT NULL,
    departure_time TIME NOT NULL,
    ticket_price DECIMAL(10, 2) NOT NULL CHECK (ticket_price > 0),
    FOREIGN KEY (departure_airport_id) REFERENCES
airports(airport_id),
    FOREIGN KEY (arrival_airport_id) REFERENCES
airports(airport_id),
    FOREIGN KEY (airplane_id) REFERENCES airplanes(airplane_id)
);
```

bookings (Бронирования)

Таблица хранит информацию о бронированиях билетов, включая данные о пользователе, рейсе, номере паспорта и статусе бронирования.

Основные столбцы:

1. booking_id: Уникальный идентификатор бронирования.
2. user_id: Идентификатор пользователя, сделавшего бронирование.
3. flight_id: Идентификатор рейса, на который было сделано бронирование.
4. passport_number, passport_seria: Номер и серия паспорта.
5. booking_date: Дата бронирования.
6. status: Статус бронирования ("подтверждено", "отменено").

Связи:

1. Связана с таблицей users через столбец user_id, чтобы указать, какой пользователь сделал бронирование.
2. Связана с таблицей flights через столбец flight_id, чтобы указать, на какой рейс было сделано бронирование.

```
CREATE TABLE bookings (
    booking_id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    flight_id INT NOT NULL,
    passport_number INT NOT NULL,
    passport_seria INT NOT NULL,
    booking_date DATE NOT NULL DEFAULT CURRENT_DATE,
    status VARCHAR(50) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (flight_id) REFERENCES flights(flight_id)
);
```

payments (Платежи)

Таблица содержит информацию о платежах, которые были произведены пользователями за бронирования. Содержит данные о дате платежа, сумме, статусе платежа и информации о карте.

Основные столбцы:

1. payment_id: Уникальный идентификатор платежа.
2. booking_id: Идентификатор бронирования, к которому привязан платеж.
3. payment_date: Дата платежа.
4. amount: Сумма платежа.
5. status: Статус платежа.
6. card_number, card_expiry, card_cvc: Информация о карте, с которой был произведен платеж.

Связи:

Связана с таблицей bookings через столбец booking_id, чтобы указать, к какому бронированию относится платеж.

```
CREATE TABLE bookings (
    booking_id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    flight_id INT NOT NULL,
    passport_number INT NOT NULL,
    passport_seria INT NOT NULL,
    booking_date DATE NOT NULL DEFAULT CURRENT_DATE,
    status VARCHAR(50) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (flight_id) REFERENCES flights(flight_id)
);
```

Общая схема моделей предметной области:

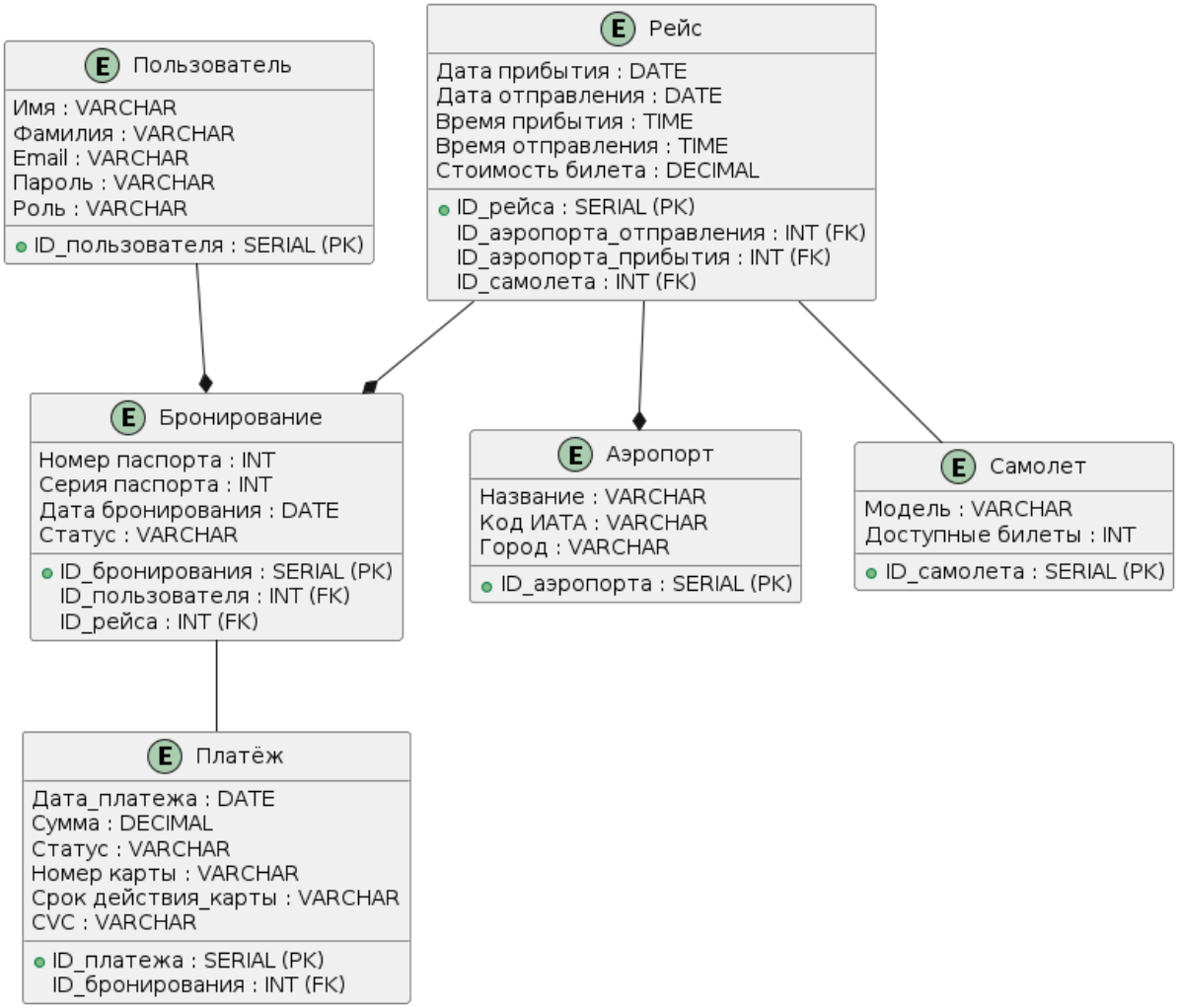

рис.1 “Схема моделей предметной области”

Описание моделей уровня инфраструктуры

Сервис авторизации/регистрации

Сервис авторизации и регистрации пользователей предоставляет функциональность для создания новых аккаунтов и проверки учетных данных для входа в систему. Он отвечает за безопасное хранение паролей и управление пользовательскими данными.

Основные функции:

- Регистрация новых пользователей.
- Авторизация пользователей (проверка учетных данных).
- Обработка и хранение паролей с использованием алгоритмов хеширования.
- Изменение пароля, восстановление доступа.
- Управление ролями пользователей ("user" и "admin").

Сервис бронирования

Сервис бронирования обрабатывает все операции, связанные с поиском рейсов, бронированием мест на рейсах и управлением статусами бронирований. Он отвечает создание рейсов, управление доступностью билетов и бронированиями, а также создание возвратных рейсов.

Основные функции:

- Поиск рейсов по различным параметрам (дата, аэропорты отправления и прибытия).
- Бронирование мест на рейсах.

- Управление статусами бронирования ("подтверждено", "отменено").
- Создание возвратных рейсов и управление их доступностью.
- Обновление количества доступных мест на рейсах после бронирования.

Сервис платежей

Сервис платежей занимается созданием и обновлением статусов платежей, а также отменой платежей.

Основные функции:

- Обработка платежей за бронирования.
- Обновление статуса платежей ("Оплачено", "Не оплачено").
- Хранение информации о платежах, включая данные карт (номер карты, срок действия, CVC).
- Отмена платежей и возврат средств.

Сервис администрирования БД

Сервис администрирования БД управляет процессом создания и восстановления резервных копий базы данных PostgreSQL.

Основные функции:

- Создание резервных копий базы данных.
- Восстановление данных из резервных копий.

Use Case диаграмма для описания функционала сервиса:

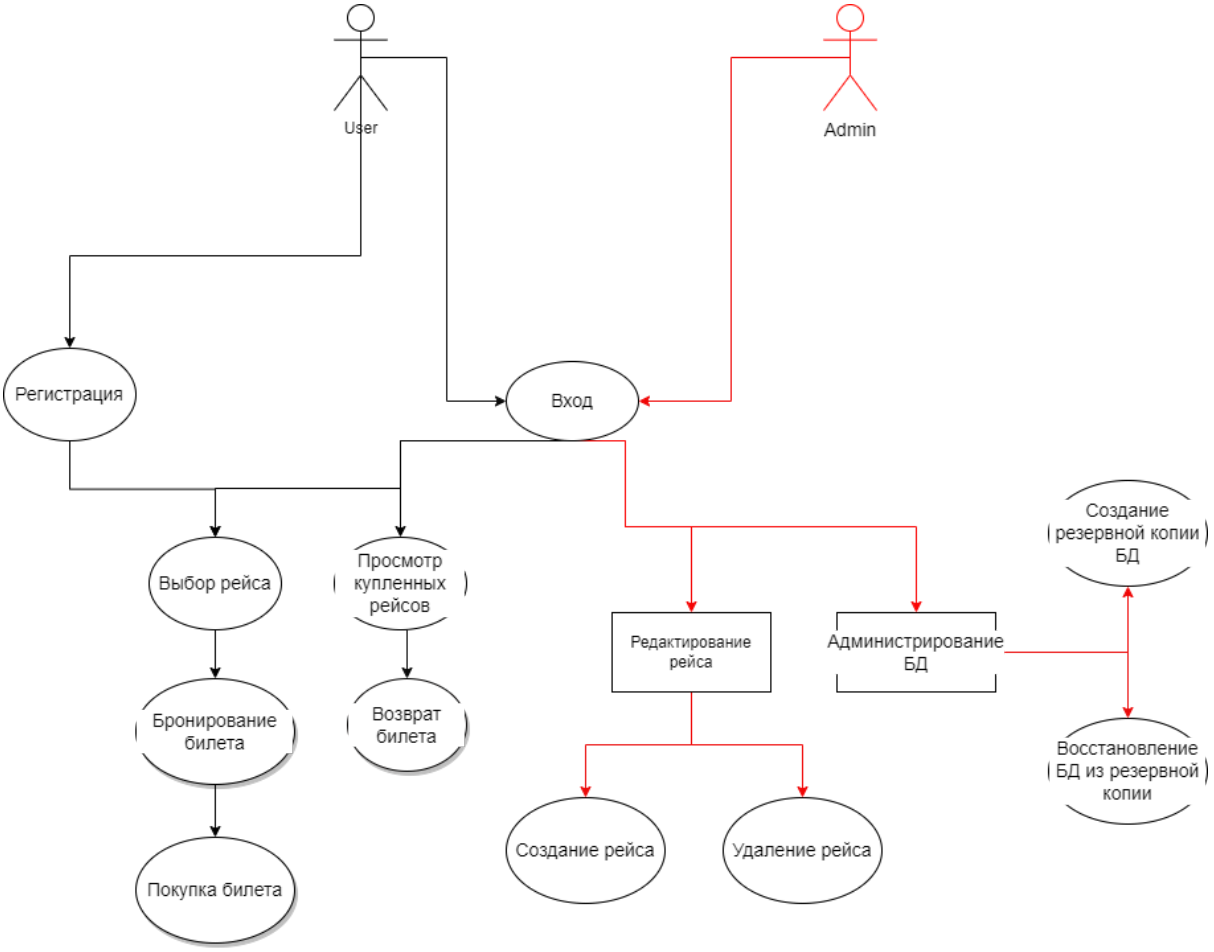

рис. 2 “Use Case диаграмма”

Разработка запросов к базе данных

Основные запросы к базе данных, используемые для реализации сервиса бронирования авиабилетов.

Добавление нового пользователя

```
INSERT INTO users (first_name, last_name, email, password_hash,
role)
VALUES (%(first_name)s, %(last_name)s, %(email)s,
%(password_hash)s, %(role)s);
```

Запрос добавляет нового пользователя в таблицу users, записывая его имя, фамилию, email, хеш пароля и роль. По умолчанию роль устанавливается как "user".

Получение пользователя по email

```
SELECT *
FROM users
WHERE email = %(email)s;
```

Этот запрос используется для получения информации о пользователе из таблицы users по его адресу электронной почты для проверки пароля при входе.

Получение рейсов

```
SELECT f.flight_id, f.departure_airport_id, f.arrival_airport_id,
f.airplane_id, f.departure_date, f.arrival_date, f.departure_time,
f.arrival_time, a1.city AS departure_city, a2.city AS
arrival_city,a1.name AS departure_airport, a2.name AS
arrival_airport,air.model AS airplane_model, f.ticket_price AS
price, air.available_tickets AS available_tickets
FROM flights f
JOIN airports a1 ON f.departure_airport_id = a1.airport_id
```

```
JOIN airports a2 ON f.arrival_airport_id = a2.airport_id
JOIN airplanes air ON f.airplane_id = air.airplane_id
WHERE a1.city = %(departure_city)s AND
      a2.city = %(arrival_city)s AND
      f.departure_date = %(departure_date)s AND
      air.available_tickets > 0
```

Этот запрос возвращает информацию о доступных рейсах между двумя городами на заданную дату.

Получение доступных самолетов

```
WITH flight_counts AS (
SELECT flights.airplane_id, COUNT(*) AS flight_count
FROM flights
WHERE flights.departure_date = %(departure_date)s
GROUP BY flights.airplane_id
)

SELECT airplanes.airplane_id, airplanes.model
FROM airplanes
LEFT JOIN flights ON airplanes.airplane_id = flights.airplane_id
LEFT JOIN flight_counts ON airplanes.airplane_id =
flight_counts.airplane_id
WHERE (
      lights.airplane_id IS NULL OR
      (flights.departure_date != %(departure_date)s) OR
          (flights.departure_date = %(departure_date)s AND
          (flights.departure_time + INTERVAL '10 hour' <=
%(departure_time)s OR
          flights.departure_time - INTERVAL '10 hour' >=
%(departure_time)s)
           )
        ) AND
       (flight_counts.flight_count IS NULL OR
flight_counts.flight_count != 2)
```

Запрос находит все доступные самолеты для рейсов на определенную дату и время, учитывая, что самолеты не должны быть заняты другими рейсами.

Обновление доступных мест на рейсе

```
UPDATE airplanes
SET available_tickets = available_tickets - 1
WHERE airplane_id = (SELECT airplane_id FROM flights WHERE
flight_id = %s);
```

Запрос обновляет количество доступных мест на самолете после того, как пользователь забронировал билет на рейс.

Получение списка всех рейсов

```
SELECT f.flight_id, a1.city AS departure_city,
       a2.city AS arrival_city,
       f.departure_date,
       f.departure_time,
       f.arrival_date,
       f.arrival_time,
       air.model AS airplane_model
FROM flights f
JOIN airports a1 ON f.departure_airport_id = a1.airport_id
JOIN airports a2 ON f.arrival_airport_id = a2.airport_id
JOIN airplanes air ON f.airplane_id = air.airplane_id;
```

Запрос извлекает список всех рейсов с информацией о городах вылета и прилета, времени вылета и прилета, а также модели самолетов.

Добавление нового рейса

```
INSERT INTO flights (departure_airport_id, arrival_airport_id,
airplane_id, departure_date, arrival_date, departure_time,
arrival_time, ticket_price) VALUES (%s, %s, %s, %s, %s, %s, %s,
%s);
```

Запрос добавляет новый рейс в таблицу flights. Для этого необходимо указать аэропорты вылета и прилета, самолет, дату и время вылета и прилета, а также цену билета.

Получение списка городов

```
SELECT DISTINCT city FROM airports;
```

Запрос извлекает уникальные города из таблицы airports, чтобы предоставить список всех доступных городов для выбора пользователю.

Получение аэропортов

```
SELECT airport_id, name, city
FROM airports
WHERE city = ANY(%(cities)s);
```

Запрос извлекает все аэропорты из списка городов, что позволяет администратору выбрать аэропорты для вылета и прилета из определенных городов.

Удаление рейса

```
DELETE FROM flights
WHERE flight_id = %(flight_id)s;
```

Запрос удаляет рейс из базы данных по его flight_id

Создание нового платежа

```
INSERT INTO payments (booking_id, payment_date, amount, status,
card_number, card_expiry, card_cvc)
VALUES (%s, %s, %s, %s, %s, %s, %s)
RETURNING payment_id;
```

Запрос вставляет новый платеж в таблицу payments. Он принимает информацию о бронировании, дате платежа, сумме, статусе (по умолчанию "Не оплачено"), номере карты, сроке ее действия и CVC. После успешного выполнения запроса возвращается payment_id нового платежа, который затем используется для дальнейших операций с этим платежом.

Вывод

В рамках проекта "Сервис бронирования авиабилетов" был разработан функционал для управления бронированиями и оплатой авиабилетов.

Система поддерживает две роли: пользователь и админ, что позволяет различать права доступа и функционал для каждого типа пользователя.

Для обеспечения безопасности, все пароли пользователей хешируются, что исключает возможность их хранения в открытом виде.

Платежи обрабатываются с возможностью изменения статуса (например, "Оплачено" или "Отменено"), а при отмене бронирования происходит автоматическое восстановление доступных билетов для других клиентов.

Для резервного хранения данных предусмотрена возможность создания резервной копии базы данных, что позволяет восстановить информацию в случае сбоев или потери данных.

Модели уровня инфраструктуры обеспечивают взаимодействие с базой данных через различные запросы, включая вставку данных, обновление информации и выборку данных по определённым критериям. Система использует транзакции для обеспечения атомарности операций, например, при создании платежей или отмене бронирований. Все данные хранятся в структурированных таблицах, что обеспечивает их целостность и безопасность.

Администраторы имеют доступ к управлению пользователями, бронированиями и платежами, что позволяет эффективно контролировать процесс бронирования и оплату авиабилетов.

Веб-сервис представляет собой надёжную и масштабируемую систему, способную поддерживать высокую нагрузку и обеспечивать удобный интерфейс для пользователей и администраторов.

Список использованных источников

1. https://docs.streamlit.io/develop
2. https://www.postgresql.org/docs/17/index.html
3. https://www.psycopg.org/docs/usage.html
4. GitHub - apftf777/sample_course_work