МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра №806 «Вычислительная математика и программирование»

Курсовая работа
по курсу «Базы данных»

Мастер организации конференций

Выполнил: Ахметшин Б.Р.
Группа: 8О-303Б-22
Преподаватель: Малахов А.В.

Москва, 2024

Оглавление

Описание приложения

Мастер организации конференций или Conference Organization Master – это приложение, целью которого является упрощение и частичная автоматизация организации конференций.

Приложение предоставляет возможности для регистрации конференций, докладов а также записи на доклады для интересующихся слушателей. Пользователям также предоставляются возможности регистрации, аутентификации и авторизации действий.

Архитектура и реализация

Приложение разработано на микросервисной архитектуре. Интерфейс пользователя представляет из себя frontend-сервис, разработанный на ЯП Python с помощью библиотеки streamlit. За логику отвечает backend-сервис, разработанный на ЯП Java с помощью фреймворков Spring Boot, Spring Security и библиотеки JDBC. В качества БД используется PostgreSQL.

Развертывается приложение на docker контейнерах, оркестрирование происходит при помощи docker compose.

Backend-сервис разработан в соответствии с Чистой архитектурой и Domain Driven Design.

Аутентификация и авторизация производится с помощью Spring Security. Ролевая модель реализована по модели Role-Based Access Control с ресурсами.

Доступ к БД осуществляется при помощи самостоятельно разработанного класса CustomJdbcTemplate, абстрагирующего пользователя (класса) от работы с соединением, что позволяет сконцентрироваться на SQL запросах. CustomJdbcTemplate использует DataSourceUtils для получения соединения, что позволяет ему поддерживать транзакции (DataSourceUtils отслеживает активные транзакции и возвращает соединение, которое задействовано в транзакции, инициированной предыдущим методом в стеке вызовов).

Схема базы данных

Схема conference

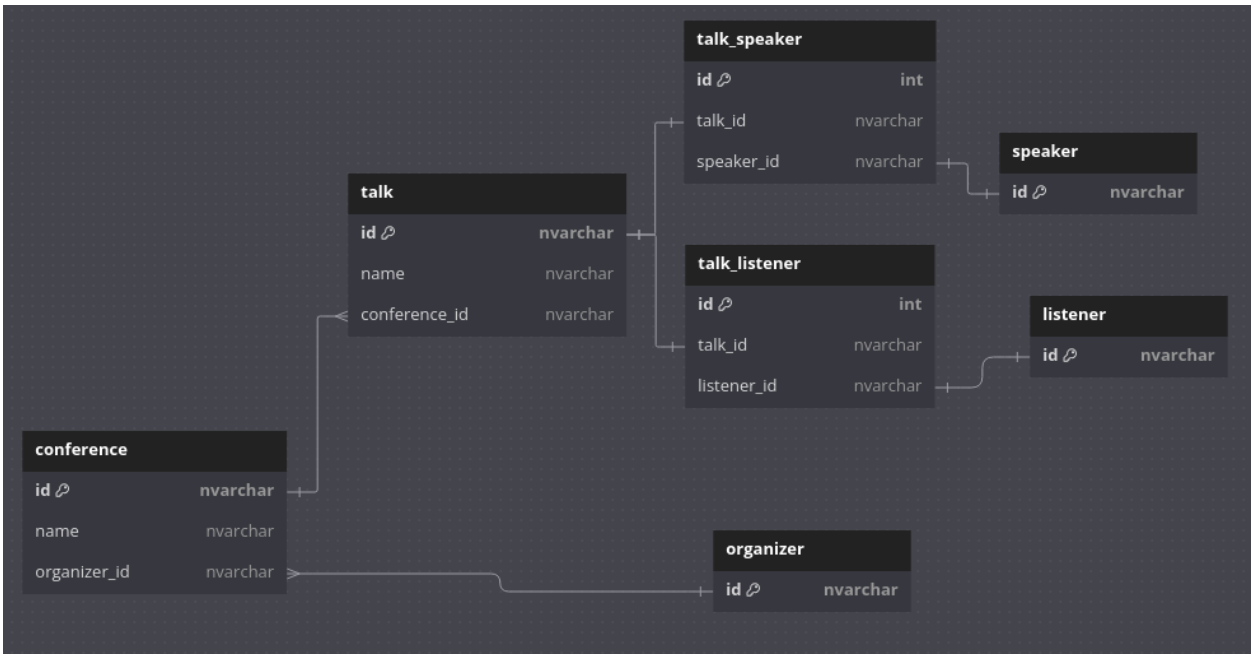
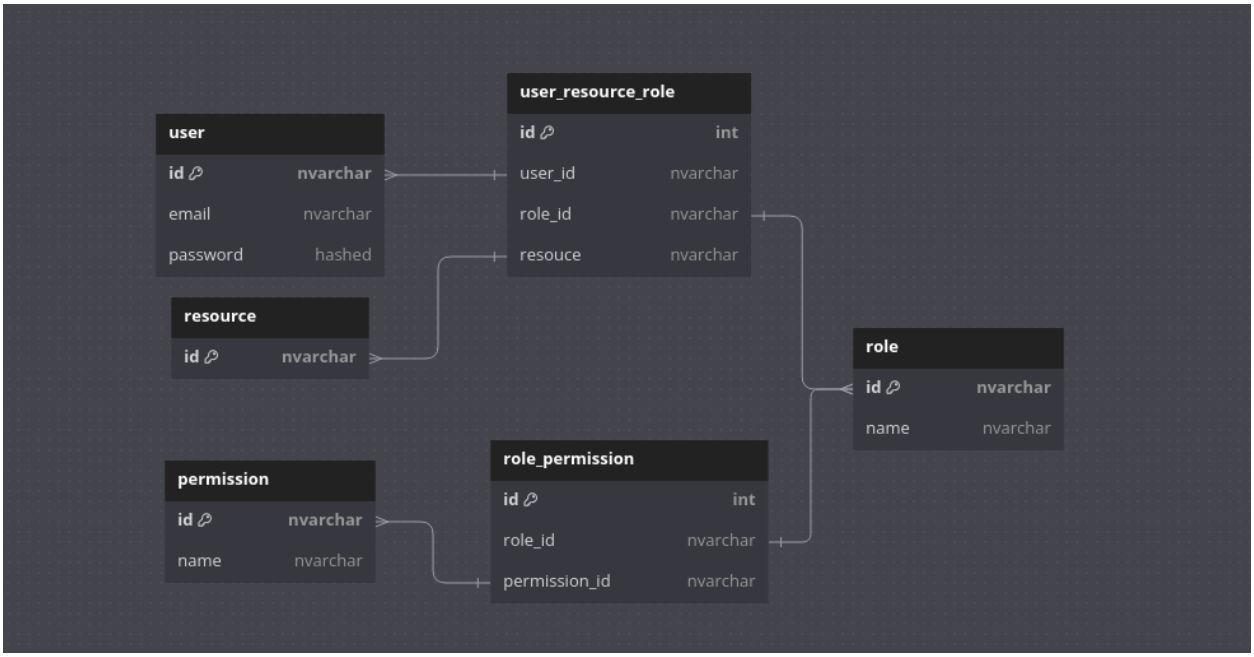


Схема user



DDL на SQL:

```
CREATE SCHEMA conference;

CREATE TABLE IF NOT EXISTS conference.organizer (
  id VARCHAR(255) PRIMARY KEY NOT NULL
);

CREATE TABLE IF NOT EXISTS conference.conference (
  id VARCHAR(255) PRIMARY KEY NOT NULL,
  name VARCHAR(255) NOT NULL,
  organizer_id VARCHAR(255) NOT NULL,
  FOREIGN KEY (organizer_id) REFERENCES
conference.organizer (id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS conference.talk (
  id VARCHAR(255) PRIMARY KEY NOT NULL,
  name VARCHAR(255) NOT NULL,
  conference_id VARCHAR(255) NOT NULL,
  FOREIGN KEY (conference_id) REFERENCES
conference.conference (id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS conference.speaker (
  id VARCHAR(255) PRIMARY KEY NOT NULL
);

CREATE TABLE IF NOT EXISTS conference.listener (
  id VARCHAR(255) PRIMARY KEY NOT NULL
);

CREATE TABLE IF NOT EXISTS conference.talk_speaker (
  talk_id VARCHAR(255) NOT NULL,
  speaker_id VARCHAR(255) NOT NULL,
  PRIMARY KEY (talk_id, speaker_id),
  FOREIGN KEY (talk_id) REFERENCES conference.talk
(id) ON DELETE CASCADE,
  FOREIGN KEY (speaker_id) REFERENCES
conference.speaker (id) ON DELETE CASCADE
```

```
);

CREATE TABLE IF NOT EXISTS conference.talk_listener (
  talk_id VARCHAR(255) NOT NULL,
  listener_id VARCHAR(255) NOT NULL,
  PRIMARY KEY (talk_id, listener_id),
  FOREIGN KEY (talk_id) REFERENCES conference.talk
(id) ON DELETE CASCADE,
  FOREIGN KEY (listener_id) REFERENCES
conference.listener (id) ON DELETE CASCADE
);

CREATE SCHEMA user;

CREATE TABLE IF NOT EXISTS user.user (
    id NVARCHAR(50) PRIMARY KEY,
    email NVARCHAR(255) NOT NULL UNIQUE,
    password NVARCHAR(MAX) NOT NULL
);

CREATE TABLE IF NOT EXISTS user.resource (
    id NVARCHAR(50) PRIMARY KEY
);

CREATE TABLE IF NOT EXISTS user.role (
    id NVARCHAR(50) PRIMARY KEY,
    name NVARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE IF NOT EXISTS user.permission (
    id NVARCHAR(50) PRIMARY KEY,
    name NVARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE IF NOT EXISTS user.role_permission (
    id INT IDENTITY PRIMARY KEY,
    role_id NVARCHAR(50) NOT NULL,
    permission_id NVARCHAR(50) NOT NULL,
    FOREIGN KEY (role_id) REFERENCES user.role(id) ON
```

```
DELETE CASCADE,
    FOREIGN KEY (permission_id) REFERENCES
user.permission(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS user.user_resource_role (
    id INT IDENTITY PRIMARY KEY,
    user_id NVARCHAR(50) NOT NULL,
    role_id NVARCHAR(50) NOT NULL,
    resource NVARCHAR(50) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES user.user(id) ON
DELETE CASCADE,
    FOREIGN KEY (role_id) REFERENCES user.role(id) ON
DELETE CASCADE,
    FOREIGN KEY (resource) REFERENCES
user.resource(id) ON DELETE CASCADE
);
```

Репозиторий с проектом