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»

**Веб-сайт для музыкального сервиса «HARMONIQ»**

Выполнила: Лупанова А.С

Группа: М8О-301Б-22

Преподаватель: А.В Малахов

Москва, 2024

**Описание проекта:**

Проект представляет из себя веб сайт для музыкального сервиса «HARMONIQ» и дает пользователю возможность управлять плейлистами.

**Стек технологий:**

* **Frontend:** Для отображения пользовательского интерфейса используется **Streamlit** — библиотека Python, позволяющая быстро создавать веб-приложения с интуитивно понятным графическим интерфейсом.
* **Backend:** Логика обработки данных и взаимодействия с базой данных реализована на Python с использованием библиотеки **psycopg2** для работы с PostgreSQL.
* **Database:** Для хранения данных используется **PostgreSQL**.

# Основной функционал:

1. **Регистрация пользователей:** Пользователи могут создать новый аккаунт, указав уникальные логин и пароль.
2. **Авторизация пользователей:** Зарегистрированные пользователи могут войти в систему, чтобы получить доступ к своим данным и функционалу.
3. **Действия с плейлистами:** В личном кабинете пользователи могут создавать и удалять плейлисты, добавлять треки в плейлист и удалять их из него.
4. **Выход из профиля:** Пользователи могут завершить сессию и выйти из аккаунта. Для повторного доступа требуется повторная авторизация.
5. **Администрирование:**

* **Просмотр активности пользователей:** Администратор может просматривать информацию о действиях всех зарегистрированных пользователей за определённый период времени.
* **Добавление и удаление треков:** Администратор может загружать треки на платформу. Если исполнитель или жанр отсутствуют в базе данных, они добавляются автоматически. Администратор также может удалять треки из базы данных.
* **Работа с резервными копиями:** Администратор может создавать резервные копии базы данных для обеспечения сохранности данных.

**Описание схемы базы данных музыкального сервиса:**

|  |  |
| --- | --- |
| Таблица | Описание |
| roles | Роли пользователей в системе |
| users | Пользователи и их учетные данные |
| actions | Список возможных действий пользователей |
| log\_actions | Логи действий пользователей |
| artists | Информация об исполнителях |
| genres | Список музыкальных жанров |
| songs | Треки и их характеристики |
| playlists | Пользовательские плейлисты |
| users\_playlists | Связь пользователей и их плейлистов |
| playlists\_songs | Связь плейлистов и треков |
| songs\_listens | Статистика прослушиваний треков |

Таблица 1: Описание схемы базы данных интернет-магазина

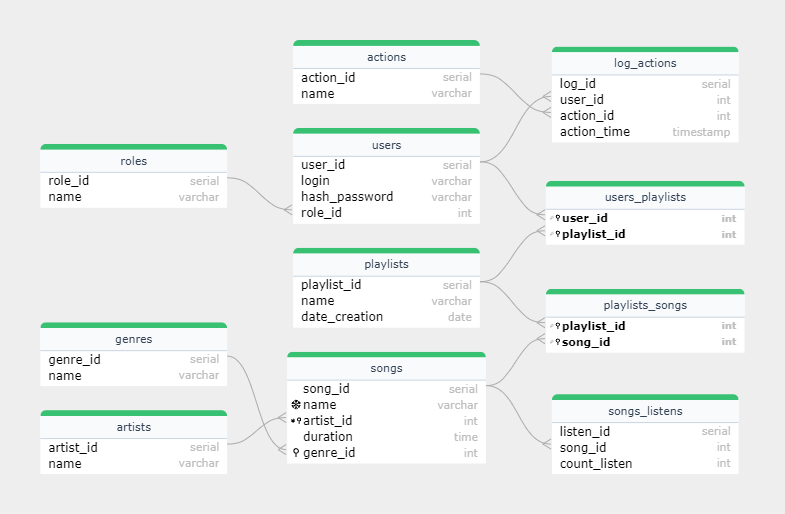


Рис. 1: Схема базы данных

Таблица **roles** содержит информацию о ролях пользователей в системе. Каждая запись включает:

* role\_id — уникальный идентификатор роли;
* name — название роли.

Таблица **users** содержит информацию о зарегистрированных пользователях. Каждая запись включает:

* user\_id — уникальный идентификатор пользователя;
* login — логин пользователя;
* hash\_password — зашифрованный пароль пользователя;
* role\_id — идентификатор роли, связанный с таблицей roles.

Таблица **actions** содержит информацию о возможных действиях пользователей в системе. Каждая запись включает:

* action\_id — уникальный идентификатор действия;
* name — название действия.

Таблица **log\_actions** регистрирует действия пользователей. Каждая запись включает:

* log\_id — уникальный идентификатор лога;
* user\_id — идентификатор пользователя, связанный с таблицей users;
* action\_id — идентификатор действия, связанный с таблицей actions;
* action\_time — время выполнения действия.

Таблица **artists** содержит информацию об исполнителях. Каждая запись включает:

* artist\_id — уникальный идентификатор исполнителя;
* name — имя исполнителя.

Таблица **genres** содержит информацию о музыкальных жанрах. Каждая запись включает:

* genre\_id — уникальный идентификатор жанра;
* name — название жанра.

Таблица **songs** содержит информацию о треках. Каждая запись включает:

* song\_id — уникальный идентификатор трека;
* name — название трека;
* artist\_id — идентификатор исполнителя, связанный с таблицей artists;
* duration — длительность трека;
* genre\_id — идентификатор жанра, связанный с таблицей genres.

Таблица **playlists** содержит информацию о пользовательских плейлистах. Каждая запись включает:

* playlist\_id — уникальный идентификатор плейлиста;
* name — название плейлиста;
* date\_creation — дата создания плейлиста.

Таблица **users\_playlists** устанавливает связь между пользователями и их плейлистами. Каждая запись включает:

* user\_id — идентификатор пользователя, связанный с таблицей users;
* playlist\_id — идентификатор плейлиста, связанный с таблицей playlists.

Таблица **playlists\_songs** устанавливает связь между плейлистами и треками. Каждая запись включает:

* playlist\_id — идентификатор плейлиста, связанный с таблицей playlists;
* song\_id — идентификатор трека, связанный с таблицей songs.

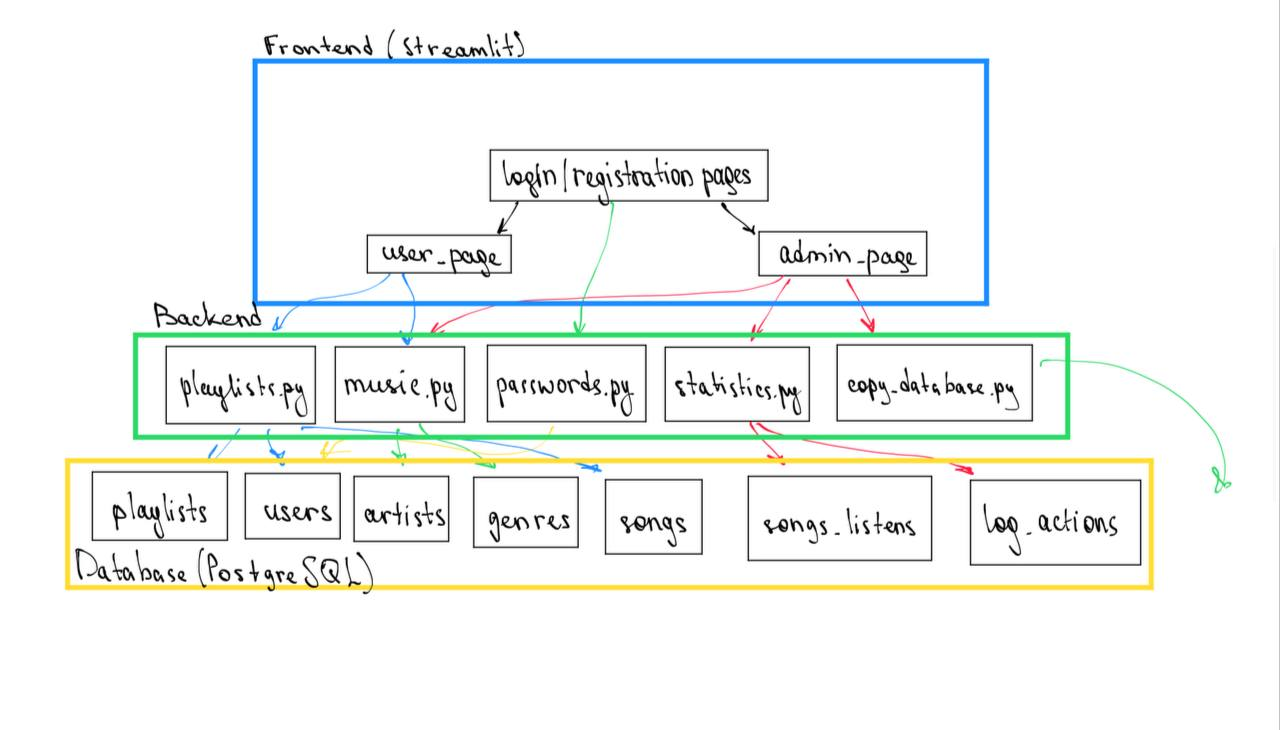
Таблица **songs\_listens** содержит информацию о прослушиваниях треков. Каждая запись включает:

* listen\_id — уникальный идентификатор записи;
* song\_id — идентификатор трека, связанный с таблицей songs;
* count\_listen — количество прослушиваний трека.

# Описание архитектуры приложения:

Архитектура приложения состоит из трёх основных компонентов: **Frontend**, **Backend**

и **Database**.



# QR-код на репозиторий с кодом:

# 