МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 05 01 Информационные системы и технологии

Специализация

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Выполнил студент

(Ф.И.О.)

Руководитель проекта

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант:

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер:

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2022

**Содержание**

[Введение 3](#_Toc130472960)

[1 Анализ требований к программному средству 4](#_Toc130472961)

[1.1 Аналитический обзор аналогов 4](#_Toc130472962)

[1.2 Разработка функциональных требований, определение вариантов использования 6](#_Toc130472963)

[2.3 Вывод 7](#_Toc130472965)

[2 Разработка архитектуры проекта 9](#_Toc130472966)

[2.1 Обобщенная структура управлением приложения 9](#_Toc130472967)

[2.2 Диаграммы UML, взаимосвязь всех компонентов. 9](#_Toc130472968)

# Введение

Цель данной работы заключается в создании реляционной базы данных для музыкальной платформы, которая обеспечивает пользователя доступом к имеющимся на ней музыкальным композициям. В рамках работы также необходимо разработать соответствующее приложение, которое будет демонстрировать функциональность базы данных и обеспечивать клиентов доступом к музыкальным трекам.

База данных - это организованное собрание данных, которое обычно хранится в электронном виде в компьютерной системе. БД используются для хранения, организации и управления большим объемом структурированных и неструктурированных данных. Реляционная база данных является наиболее распространенной формой организации данных, в которой данные представлены в виде таблиц, состоящих из строк и столбцов, где каждый столбец представляет атрибут, а каждая строка представляет кортеж или запись. В данной работе для управления базой данных была выбрана СУБД Postgres SQL, поскольку эта система обладает высокой надежностью и производительностью, что позволяет обеспечить эффективное хранение, обработку и управление музыкальными данными.

Также необходимо разработать приложение для демонстрации функциональности базы данных и взаимодействия с ней. Приложение было реализовано с использованием языка программирования Node.js и фреймворка React с TypeScript.

Для гарантированной безопасности пользователей приложения в моей курсовой работе применяется метод шифрования паролей перед их сохранением в базу данных. Также для обеспечения функциональности приложения используются мультимедийные форматы данных при сохранении аудио и картинок.

Основные требования к приложению:

* Реализация ролей администратора и пользователя.
* Поиск аудиозаписей по альбому, исполнителю или жанру.
* Загрузка аудиозаписей на платформу пользователем.
* Взаимодействие с базой данных при помощи хранимых процедур и функций.

В пояснительной записке содержится информация о сопоставимых продуктах, структуре и реализации проекта, а также инструкции по использованию приложения.

1. Анализ требований к программному средству

## Аналитический обзор аналогов

Музыка - это одна из самых популярных и распространенных форм искусства, которая имеет давнюю историю и оказывает значительное влияние на культуру и общество в целом. В настоящее время музыкальные платформы являются важной частью музыкальной индустрии, обеспечивая людям доступ к огромной библиотеке музыкальных произведений различных жанров и эпох.

Одним из основных преимуществ музыкальных платформ является возможность слушать музыку в любое время и в любом месте с помощью смартфона, компьютера или другого устройства с доступом в Интернет. Это делает музыкальные платформы очень популярными среди миллионов пользователей по всему миру.

Ниже будут описаны аналоги.

Spotify - это популярная музыкальная платформа, которая позволяет пользователям стримить музыку онлайн, создавать плейлисты и делиться ими с друзьями, слушать радио, а также находить новую музыку, основываясь на своих предпочтениях. Пример интерфейса данного сервиса представлен на рисунке 1.1.

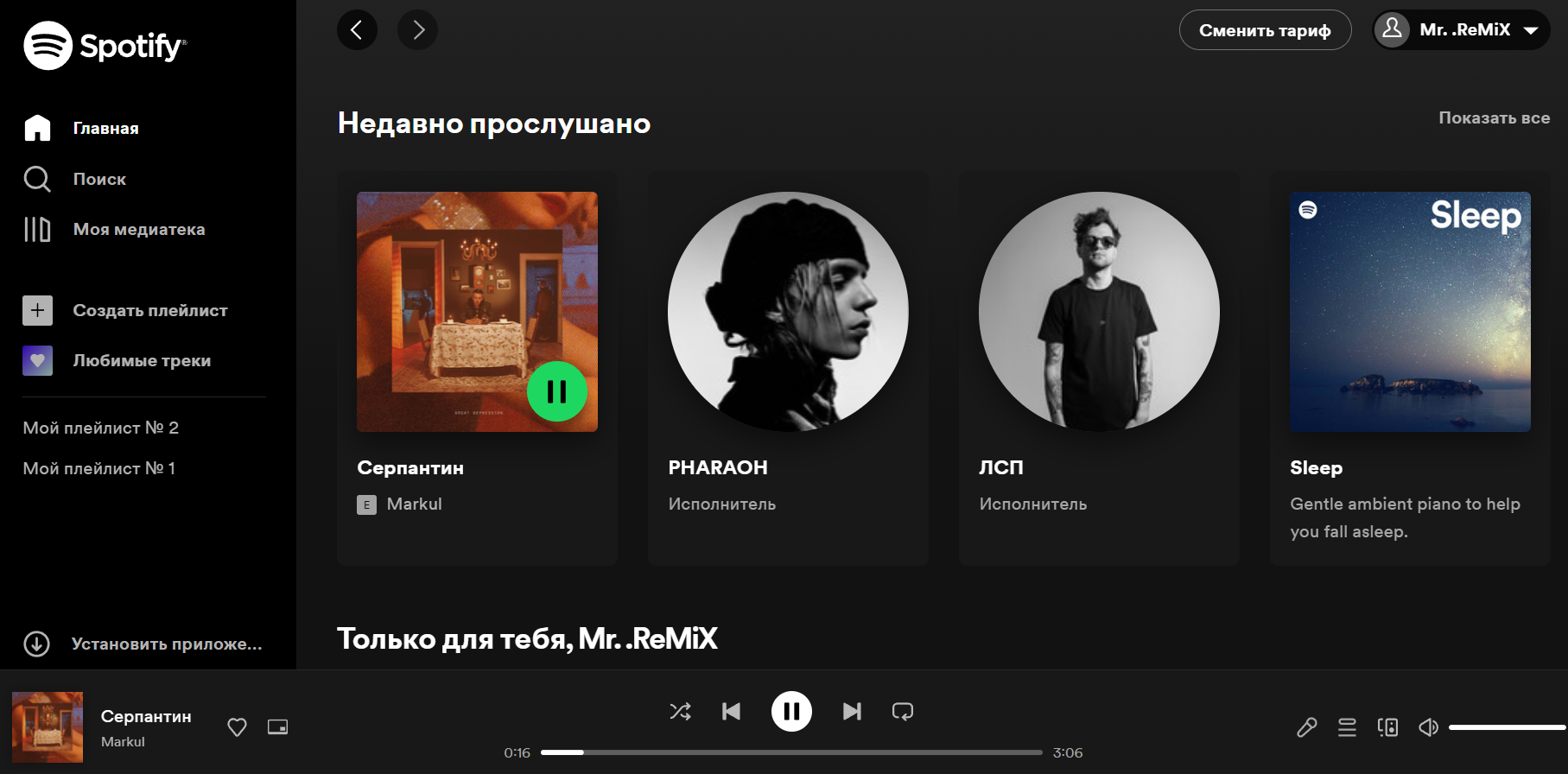


Рисунок 1.1 – Интерфейс сервиса Spotify

Одной из основных функций Spotify является поиск музыки. Пользователи могут искать музыку по альбому, исполнителю, жанру или песне. Кроме того, Spotify предлагает персонализированные рекомендации в соответствии с предпочтениями пользователя, а также плейлисты, созданные другими пользователями и кураторами платформы.

Пользователи могут создавать свои собственные плейлисты и делиться ими с друзьями или сохранять музыку для офлайн-воспроизведения. Spotify также позволяет пользователям слушать радио и подкасты, включая оригинальный контент, созданный самой платформой.

Другая функция Spotify - это возможность использовать платформу как социальную сеть, где пользователи могут подписываться на друг друга, просматривать их плейлисты и рекомендации, а также обмениваться сообщениями.

Spotify также предлагает два варианта подписки: бесплатную и платную. Бесплатная версия содержит рекламу и ограничения в использовании, в то время как платная версия позволяет получить неограниченный доступ к музыке, отсутствие рекламы и другие функции.

Один из аналогов музыкальной платформы, рассмотренной в ходе обзора, - Apple Music. Этот сервис был создан компанией Apple и запущен в 2015 году. Он позволяет пользователям прослушивать музыку на различных устройствах, а также создавать собственные плейлисты и подборки.

Apple Music предлагает пользователю возможность доступа к более чем 75 миллионам песен, а также к подкастам и аудиокнигам. Сервис также предлагает персонализированные рекомендации, основанные на предпочтениях пользователя.

Пользователи могут создавать свои плейлисты и делиться ими с другими пользователями. Apple Music также предлагает радиостанции, созданные известными диджеями и музыкантами, а также живые трансляции концертов и мероприятий.

Интерфейс Apple Music схож с интерфейсом Spotify и предлагает удобный поиск по альбомам, песням и исполнителям. Также сервис позволяет скачивать музыку для офлайн прослушивания. Пример интерфейса данного сервиса представлен на рисунке 1.2.

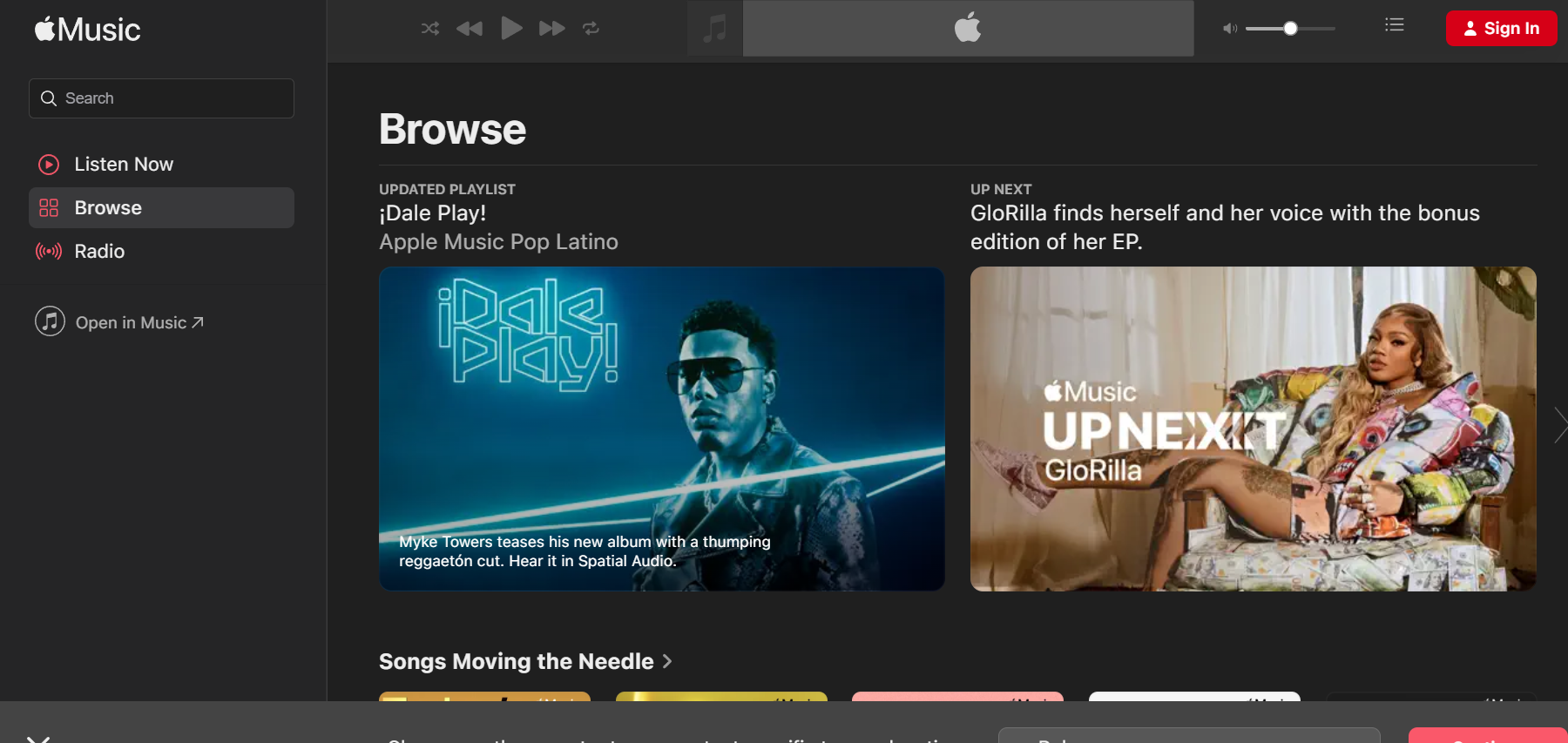


Рисунок 1.2 – Интерфейс сервиса Apple Music

В данном разделе были проведены аналитические обзоры двух популярных музыкальных сервисов - Spotify и Apple Music. Оба сервиса предоставляют пользователям возможность прослушивать музыку онлайн, создавать персональные плейлисты и получать рекомендации на основе своих предпочтений.

## 1.2 Разработка функциональных требований, определение вариантов использования

Функциональные требования базы данных определяют, как база данных должна обрабатывать данные и предоставлять пользовательскому интерфейсу необходимую функциональность. Это может включать в себя описание того, как данные должны храниться и организовываться, как происходит поиск и выборка данных, каким образом обновляются данные и какие механизмы используются для защиты данных. Кроме того, функциональные требования могут определять интеграцию базы данных с другими системами и программами. Например, для музыкальной площадки функциональные требования могут включать в себя функции для хранения информации о музыкальных треках и пользователях, поиска музыки по категориям и критериям, создания и управления плейлистами, а также функции для оценки и прослушивания музыки.

Помимо функциональных требований, важно также определить роли пользователей и их варианты использования системы. Варианты использования описывают, как пользователи будут взаимодействовать с системой в зависимости от своих ролей. Это помогает определить, какие функции должны быть доступны для каждой роли, какие данные должны быть доступны для каждой роли, а также как должна быть организована навигация в системе. Варианты использования обычно представляются в виде UML диаграмм, которые позволяют наглядно отобразить взаимодействие между пользователями и системой.

Роли пользователя — это набор прав, которые пользователь может получить в системе. В зависимости от роли пользователя, он может иметь доступ к различным функциям системы. В данном проекте роли пользователей будут следующими:

* Гость.
* Пользователь.
* Администратор.

На основе предоставленного списка ролей необходимо построить варианты использование. Варианты использование изображена на рисунке 1.3.

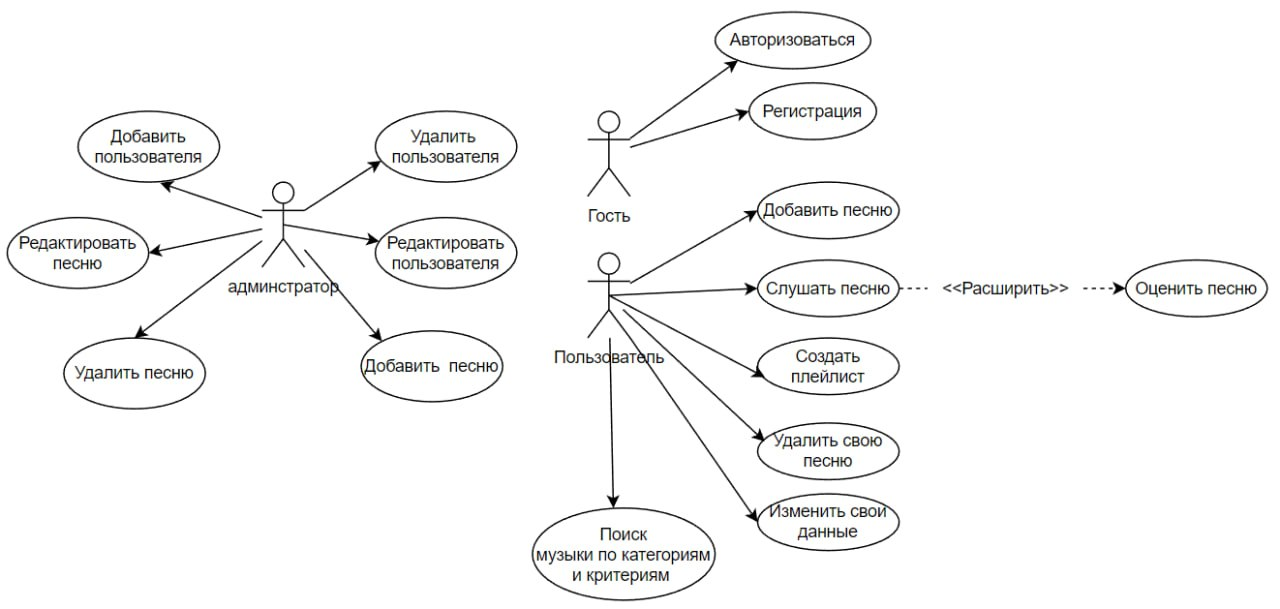


Рисунок 1.3 – UML диаграмма вариантов использования

В начале работы с приложением пользователь является гостем. Ему будет доступна только просмотр всей информации без возможности слушать музыку. После регистрации пользователь становится пользователем (User).

После регистрации пользователь (User) получает возможность слушать музыку, добавлять понравившиеся треки в свой персональный плейлист, оценивать музыкальные треки и осуществлять поиск музыки по категориям и критериям.

Роль администратора (Admin) заключается в управлении площадкой и мониторинге действий пользователей. Администратор имеет доступ к просмотру информации о всех пользователях, в том числе их действиях на площадке. При необходимости администратор может редактировать информацию о треках и пользователях, например, изменять категории и теги у треков, а также изменять личную информацию о пользователях.

## В данном разделе были определены роли пользователей и разработаны варианты использования системы в зависимости от этих ролей. Гость может только просматривать информацию, пользователь получает доступ к функциям системы, таким как прослушивание музыки, добавление треков в плейлист и поиск музыки, а администратор имеет права на управление площадкой и редактирование информации о пользователях и треках. Варианты использования системы изображены на UML диаграмме.

## 2.3 Вывод

Итого, был проведен аналитический обзор аналогов музыкальных платформ и сервисов, которые уже существуют на рынке. Этот обзор позволил определить основные характеристики и функциональные возможности, которые необходимо предусмотреть в разрабатываемой системе. Также были определены функциональные требования базы данных, а также роли пользователей и варианты использования системы в зависимости от этих ролей. Была разработана UML-диаграмма, на которой отображены основные функции, которые доступны для каждой из ролей пользователей.

1. Разработка архитектуры проекта

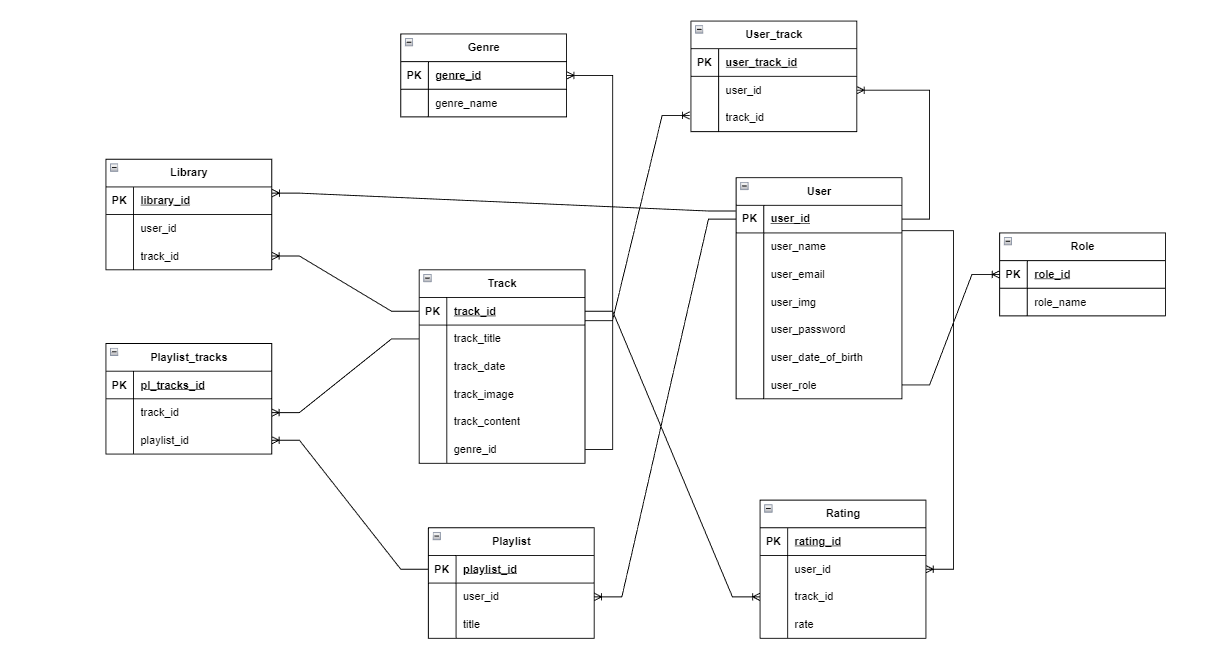
## Обобщенная структура управлением приложения

Для обеспечения управления приложением с использованием базы данных необходимо разработать удобный и интуитивно понятный интерфейс, который позволит пользователю взаимодействовать с базой данных и эффективно управлять данными. Это может включать в себя разработку оптимизированных запросов для вставки, обновления и удаления данных, а также разработку механизмов для извлечения и обработки информации из базы данных.

В функциональность приложения для прослушивания музыки может входить возможность быстрого и удобного поиска музыкальных композиций, исполнителей и альбомов по различным параметрам, таким как жанр, год выпуска, рейтинг и т.д. Также пользователи должны иметь возможность проигрывать музыкальные композиции на сайте без необходимости скачивания или установки специального программного обеспечения. Для удобства пользователей также должна быть предусмотрена возможность создания собственных плейлистов с возможностью настройки порядка воспроизведения.

## 2.2 Диаграммы UML, взаимосвязь всех компонентов.

Диаграмма базы данных таблиц (Database Table Diagram) - это визуальное представление структуры базы данных и отношений между таблицами, которые хранятся в этой базе данных. Диаграмма базы данных будет представлена на рисунке 2.1.



Таким образом, диаграмма показывает связи между таблицами и полями, а также отношения между ними, такие как связи "один-ко-многим", "многие-ко-многим" и "один-к-одному". Например, таблица Users связана с таблицами Artist, Rating, Playlist и Library\_user через внешние ключи user\_id. Также видно, что таблицы Track, Rating, Playlist\_tracks и Library\_user связаны с таблицей Users, а таблицы User\_track, Track и Genre связаны друг с другом через внешние ключи.

**2.3 Описание информационных объектов**

Для реализации базы данных было разработано 8 таблиц. В структуру схемы базы данных для проекта входят следующие таблицы: Role, Users, Genre, Artist, Track, Rating, Playlist, Playlist\_tracks, Library\_user. Ниже мы разберем каждую из них более подробно.

Таблица Role представляет собой перечень ролей, которые могут иметь пользователи. Имеющиеся поля: role\_id (идентификатор роли, типа SERIAL, первичный ключ), role\_name (текстовое название роли, тип VARCHAR(255)).

Таблица Users содержит информацию о пользователях. Имеющиеся поля: user\_id (идентификатор пользователя, типа SERIAL, первичный ключ), user\_name (имя пользователя, тип VARCHAR(255)), user\_img (изображение пользователя, тип BYTEA), user\_email (адрес электронной почты пользователя, тип VARCHAR(255)), user\_password (пароль пользователя, тип VARCHAR(255)), user\_date\_of\_birth (дата рождения пользователя, тип DATE), user\_role\_id (идентификатор роли пользователя, тип INTEGER, внешний ключ ссылающийся на role\_id в таблице Role).

Таблица Genre содержит информацию о жанрах музыки. Имеющиеся поля: genre\_id (идентификатор жанра, типа SERIAL, первичный ключ), genre\_name (название жанра, тип VARCHAR(255)).

Таблица Artist содержит информацию об исполнителях. Имеющиеся поля: artist\_id (идентификатор исполнителя, типа SERIAL, первичный ключ), user\_id (идентификатор пользователя-владельца, тип INTEGER, внешний ключ ссылающийся на user\_id в таблице Users).

Таблица Track содержит информацию о треках. Имеющиеся поля: track\_id (идентификатор трека, типа SERIAL, первичный ключ), track\_title (название трека, тип VARCHAR(255)), track\_date (дата создания трека, тип DATE), artist\_id (идентификатор исполнителя, тип INTEGER, внешний ключ ссылающийся на artist\_id в таблице Artist), track\_image (изображение трека, тип BYTEA), track\_content (содержание трека, тип BYTEA), genre\_id (идентификатор жанра, тип INTEGER, внешний ключ ссылающийся на genre\_id в таблице Genre).

Таблица Rating содержит информацию о рейтинге пользователей для треков. Имеющиеся поля: rating\_id (идентификатор рейтинга, типа SERIAL, первичный ключ), user\_id (идентификатор пользователя, который оценил трек, тип INTEGER, внешний ключ ссылающийся на user\_id в таблице Users), track\_id (идентификатор трека, который был оценен, тип INTEGER, внешний ключ ссылающийся на track\_id в таблице Track), rate (оценка, которую поставил пользователь, тип INTEGER).

Таблица Playlist содержит информацию о плейлистах пользователей. Имеющиеся поля: playlist\_id (идентификатор плейлиста, типа SERIAL, первичный ключ), user\_id (идентификикатор пользователя, создавшего плейлист, тип INTEGER, внешний ключ ссылающийся на user\_id в таблице Users) и title (название плейлиста, тип VARCHAR(255) NOT NULL).

Таблица Playlist\_tracks связывает плейлисты и треки. Имеющиеся поля: id (идентификатор записи, типа SERIAL, первичный ключ), track\_id (идентификатор трека, тип INTEGER, внешний ключ ссылающийся на track\_id в таблице Track) и playlist\_id (идентификатор плейлиста, тип INTEGER, внешний ключ ссылающийся на playlist\_id в таблице Playlist).

Таблица Library\_user содержит информацию о треках, которые добавлены в библиотеку пользователей. Имеющиеся поля: id (идентификатор записи, типа SERIAL, первичный ключ), user\_id (идентификатор пользователя, который добавил трек в свою библиотеку, тип INTEGER, внешний ключ ссылающийся на user\_id в таблице Users) и track\_id (идентификатор трека, тип INTEGER, внешний ключ ссылающийся на track\_id в таблице Track).

**2.4 Вывод**

Разработка архитектуры проекта необходима для определения структуры и функциональности приложения. Обобщенная структура управления приложения позволяет определить, какие компоненты необходимы для реализации приложения и как они должны взаимодействовать между собой.

Описание информационных объектов является важной частью архитектуры проекта, так как это помогает понять, какие данные будут использоваться в приложении, и как они будут храниться и обрабатываться. В данном проекте были описаны объекты, такие как пользователи, роли, жанры, артисты, треки, рейтинги, плейлисты и библиотека пользователя.

В целом, разработка архитектуры проекта является важным шагом в процессе разработки приложения, так как это позволяет определить необходимые компоненты и информационные объекты, которые помогут создать функциональное и эффективное приложение.