МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет компьютерных наук

Кафедра программирования и информационных систем

Разработка стримингового музыкального веб-приложения «Musicman»

Курсовая работа
09.03.04 Программная инженерия
Информационные системы и технологии

Допущено к защите в ГЭК __.__.2023

Зав. кафедрой _____С.Д. Махортов, д. ф.-м. н., профессор

Обучающийся _____А.Р. Сашина, *3 курс, д/о*

Обучающийся _____З.С. Казмиров, *3 курс, д/о*

Обучающийся _____Е.М. Охрямкин, *3 курс, д/о*

Руководитель _____В.С. Тарасов, старший преподаватель

Воронеж 2023

Содержание

Введение

В современном мире стриминговые музыкальные сервисы являются неотъемлемой частью жизни людей и получают все большую популярность. Они предоставляют пользователю удобный доступ к миллионам аудиозаписей в любое время и в любом месте, что позволяет прослушивать музыку в комфортных условиях.

Цель данной курсовой работы – разработка веб приложения музыкального стримингового сервиса, позволяющего пользователю в удобной форме слушать любимую музыку и делиться песнями с другими пользователями.

В работе будет рассмотрен процесс разработки приложения, включая его проектирование, функциональность, дизайн и архитектуру системы.

Для реализации проекта будут использованы современные технологии и инструменты, необходимые для создания качественного веб приложения. Работа будет состоять из нескольких этапов, включая анализ потребностей пользователей, проектирование интерфейса и архитектуры, разработку функциональности и интеграцию с внешними сервисами, а также тестирование и оптимизацию приложения для улучшения его производительности и удобства использования.

Результатом данной работы будет функциональное и удобное веб приложение музыкального стримингового сервиса, удовлетворяющее потребностям пользователей и соответствующее современным требованиям в области веб разработки.

1 Постановка задачи

1.1 Постановка задачи

Основной задачей курсового проекта является создания стримингового сервиса для прослушивания песен с возможностью настройки приложения под нужды пользователя.

Основными целями создания системы являются:

— Возможность регулировать настройки прослушивания песен;

— Показ песен пользователю на основе его музыкальных предпочтений;

— Возможность рекомендации песни другим пользователям.

1.2 Требования к разрабатываемой системе

1.2.1 Функциональные требования

Для реализации поставленных целей система должна решать следующие задачи:

— Регистрация пользователей;

— Загрузка, удаление и добавление музыки;

— Ведение статистики пользователя по жанрам прослушиваемой музыки;

— Просмотр рекомендуемых песен;

— Возможность рекомендации песни пользователям;

— Возможность добавления в друзья других пользователей;

— Настройка звукового сигнала.

1.2.2 Требования к приложению и программному обеспечению

К разрабатываемому приложению выдвигаются следующие требования:

— Приложение должно корректно отображаться в современных браузерах;

— Должны быть реализованы все цели, стоящие перед проектом;

— Созданное приложение должно соответствовать архитектуре клиент-серверного приложения. Необходимо разделение на серверную клиентскую части, взаимодействующие между собой с помощью REST API.

2 Анализ предметной области

2.1 Терминология (глоссарий)

Администратор сайта – специалист, осуществляющий информационную поддержку сайта, управление контентом.

Веб-браузер (браузер) - клиентская программа, поставляемая третьими сторонами и позволяющая просматривать содержимое веб-страниц.

Контент – совокупность информационного наполнения веб-сайта.

Неавторизованный пользователь (гость) – человек, который может авторизоваться в системе, если был зарегистрирован ранее, или пройти регистрацию.

Пользователь – человек, который зарегистрирован в системе и имеет доступ к личному кабинету и основному функционалу системы.

Система администрирования – закрытая от посетителей часть сайта. Управляется администратором.

Эквалайзер – программа, позволяющая регулировать громкость отдельных зон частотного диапазона и выравнивать амплитудно-частотную характеристику звукового сигнала.

2.2 Анализ рынка

Музыкальные стриминговые сервисы являются одними из наиболее популярных услуг, которые с каждым годом набирают большую аудиторию. С 2017 продажи альбомов на цифровых и физических носителях упали на 17% и 20% соответственно, несмотря на это, объем продаж музыки все равно вырос [1]. Связано это с ростом стриминговых сервисов – количество прослушиваний в них выросло на 60%.

Аналогичная ситуация в 2016 году наблюдалась и в России — наиболее активный прирост дало потоковое вещание. Более миллиона платящих пользователей Apple Music, «Яндекс.Музыки» и других сервисов помогли удвоить доходы этих компаний в данном сегменте. Теперь почти 40% индустрии приходится на стриминг.

Еще 4 года назад лишь четверть россиян готова была заплатить за контент в интернете. Через год эта цифра увеличилась вдвое.

Однако в 2022 году российский рынок стриминга показал спад [2]. Сокращение составило 30-50% год к году до 5-7 млрд рублей. В 2021 аналитики оценивали объем рынка в 9,5 млрд рублей.

При этом российские музыкальные сервисы заявляют о росте аудитории. Например, в «Яндексе» рассказали, что количество подписчиков «Плюса» за год увеличилось на 53%.

Таким образом , можно сделать вывод о том, что рынок стриминговых сервисов в России продолжает развиваться. Сейчас доля пользователей, готовых к покупке контента, достигла свыше 70%. Поэтому существует огромный потенциал развития, возможность завоевать долю этого развивающегося рынка, имея инновационный продукт и правильную стратегию развития проекта.

2.3 Обзор аналогов

Перед разработкой приложения был проведен обзор существующих аналогов на рынке.

2.3.1 Spotify

Spotify – это один из самых популярных музыкальных стриминговых сервисов в мире. Этот сервис имеет огромную библиотеку с миллионами песен и плейлистов, которые можно слушать как онлайн, так и офлайн. Spotify имеет простой и интуитивно понятный интерфейс, а также множество функций, таких как персонализированные рекомендации, функция обнаружения новой музыки, плейлисты, созданные другими пользователями, и многое другое.

Плюсы:

— Большая библиотека с миллионами песен и плейлистов;

— Простой и удобный интерфейс (см. Рисунок 1);

— Персонализированные рекомендации и функция обнаружения новой музыки;

— Доступно на большинстве устройств, включая смартфоны, компьютеры, телевизоры и т.д.;

— Функция офлайн-воспроизведения песен.

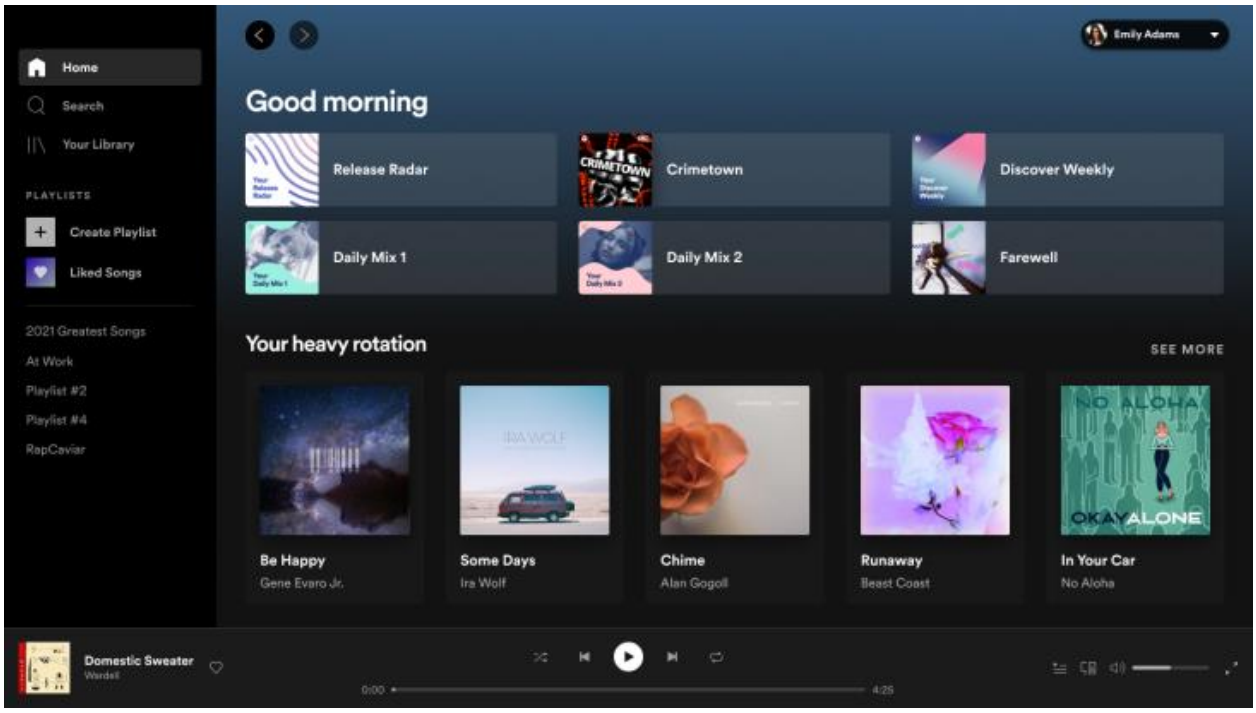

Рисунок 1 - Главная страница Spotify

Минусы:

— Реклама для бесплатной версии;

— Ограниченный доступ к новым альбомам для бесплатной версии;

— Некоторые пользователи могут считать плату за премиум-версию слишком высокой.

2.3.2 Deezer

Deezer – это музыкальный стриминговый сервис, который доступен в более чем 180 странах. Этот сервис также имеет огромную библиотеку с миллионами песен и плейлистов, а также множество функций, таких как подборки песен на основе настроения, автоматическая настройка на любимые исполнители, подкасты и многое другое.

Плюсы:

— Огромная библиотека с миллионами песен и плейлистов;

— Множество функций, таких как подборки песен на основе настроения, автоматическая настройка на любимые исполнители, подкасты и многое другое;

— Доступно на большинстве устройств, включая смартфоны, компьютеры, телевизоры и т.д. (см. Рисунок 2);

— Качество звука выше, чем у конкурентов;

— Функция офлайн-воспроизведения песен.

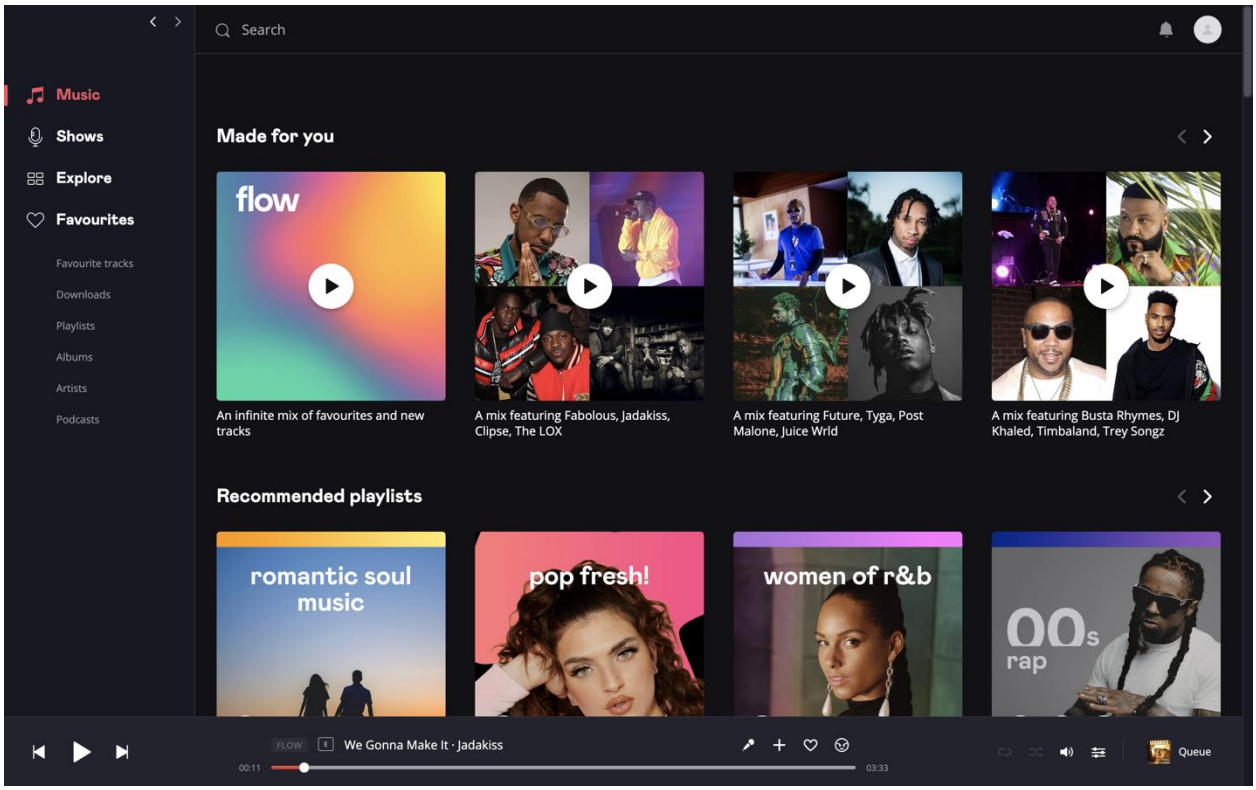

Рисунок 2 - Главная страница Deezer

Минусы:

— Некоторые пользователи могут считать плату за премиум-версию слишком высокой;

— Не все песни доступны в некоторых странах;

— Реклама для бесплатной версии.

2.3.3 Яндекс Музыка

Яндекс Музыка – это музыкальный стриминговый сервис, разработанный Яндексом. Сервис имеет огромную библиотеку с миллионами

песен и плейлистов, а также множество функций, таких как рекомендации на основе настроения, адаптивный плейлист, функция персонализации и многое другое.

Плюсы:

- Большая библиотека с миллионами песен и плейлистов;
- Множество функций, таких как рекомендации на основе настроения, адаптивный плейлист, функция персонализации и многое другое;
- Доступно на большинстве устройств, включая смартфоны, компьютеры, телевизоры и т.д. (см. Рисунок 3);
- Бесплатный доступ к музыке для пользователей Яндекс.Плюс;
- Функция офлайн-воспроизведения песен.

Минусы:

- Некоторые пользователи могут считать интерфейс сложным и запутанным;
- Ограниченный доступ к новым альбомам для бесплатной версии;
- Реклама для бесплатной версии;
- Некоторые пользователи могут испытывать проблемы с качеством звука.

2.3.4 Youtube Music

Youtube Music – это музыкальный стриминговый сервис, разработанный Google. Сервис имеет библиотеку с миллионами песен и плейлистов, а также множество функций, таких как персонализированные рекомендации, офлайн-воспроизведение и многое другое.

Плюсы:

- Большая библиотека с миллионами песен и плейлистов;
- Множество функций, таких как персонализированные рекомендации, офлайн-воспроизведение и многое другое;

— Доступно на большинстве устройств, включая смартфоны, компьютеры, телевизоры и т.д. (см. Рисунок 4);

— Бесплатный доступ к музыке для пользователей Youtube Premium;

— Интеграция с другими сервисами Google, такими как Google Assistant и Google Home.

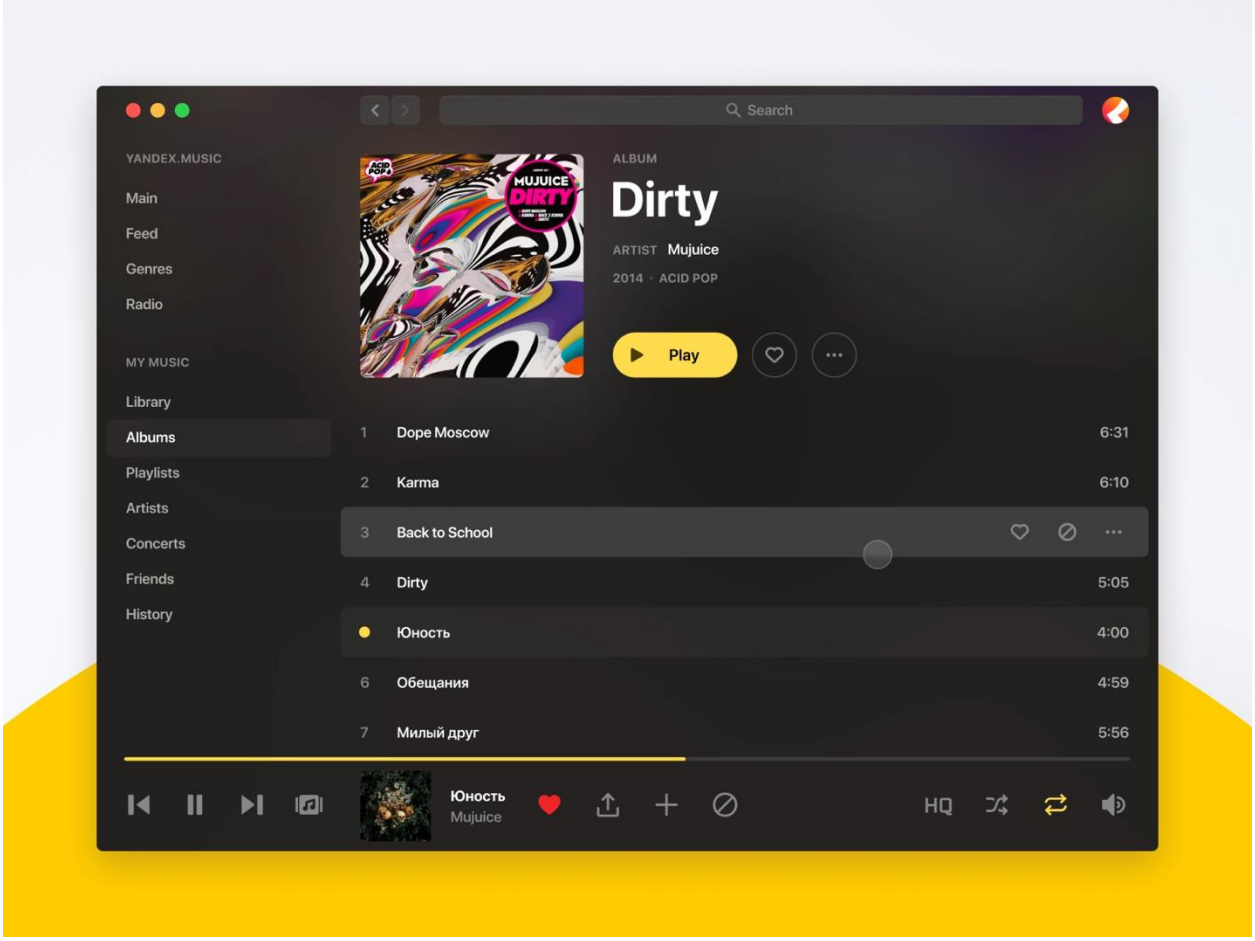

Рисунок 3 - Главная страница Яндекс Музыки

Минусы:

— Некоторые пользователи могут считать интерфейс сложным и запутанным;

— Высокая стоимость премиум-подписки;

— Ограниченный доступ к новым альбомам для бесплатной версии;

— Реклама для бесплатной версии.

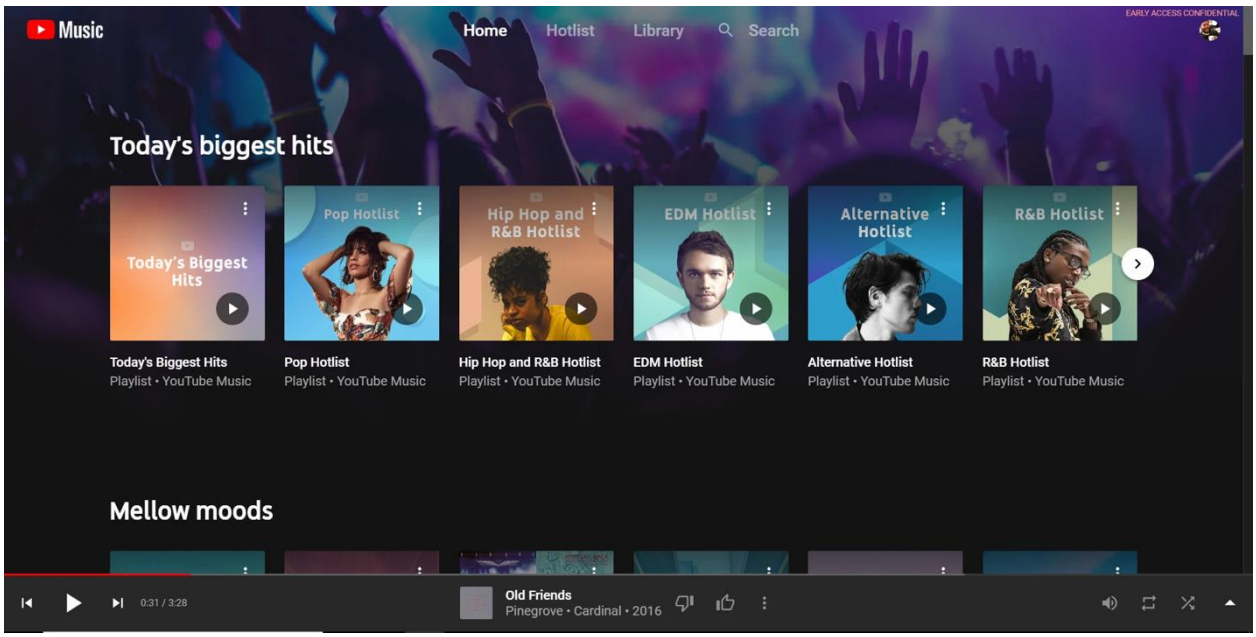

Рисунок 4 - Главная страница Youtube Music

3 Реализация

3.1 Средства реализации

Система должна состоять из сервера приложения, реляционной базы данных, клиентской части.

Основной используемый стек технологий:

Back-end (серверная часть):

— Java 17;

— Spring Framework;

— PostgreSQL, Liquibase.

Java [3] является кроссплатформенным языком, т.е. для запуска достаточно иметь Java Virtual Machine. Самым популярным фреймворком для Java является Spring [4]. Основным преимуществом является огромное наличие компонент и внутренних библиотек, которые уже реализованы, а значит позволяет быстро и качественно написать код.

В качестве базы данных была выбрана PostgreSQL [6], так как умеет работать с различными типами данных и позволяет ускорять запросы с помощью индексов. Для управления и применения изменений в базу данных будет использоваться библиотека Liquibase [7]. Основным преимуществом является поддержка написания миграционных файлов в виде yaml или xml файлов.

Front-end (клиентская часть):

— CSS3 + HTML5;

— React.js, TypeScript;

— Effector;

— Material UI.

CSS3 + HTML5 были выбраны потому, что это самый современный стандарт вёрстки и разметки. Он поддерживается большинством браузеров и предоставляет множество новых свойств, упрощающих разработку.

React [5] является популярным фреймворком для разработки клиентской части. Главным его преимуществом является виртуальная объектная модель документа (DOM), занимающая мало места. А значит позволяет быстрее обновлять страницу с изменениями и повышает производительность приложения.

Язык TypeScript был выбран, потому что это компилируемый в JavaScript код, который нивелирует недостатки слабой типизации JavaScript и берёт множество проверок безопасности кода на себя во время трансляции в JavaScript код.

Effector [8] является стейт-менеджером, который хорошо взаимодействует с React-ом и позволяет легко разделять работу с данными по разным хранилищам (декомпозиция). Material UI [9] является библиотекой готовых компонент для React, которые обладают приятным дизайном.

3.2 Анализ поставленной задачи

3.2.1 Диаграмма IDEF0

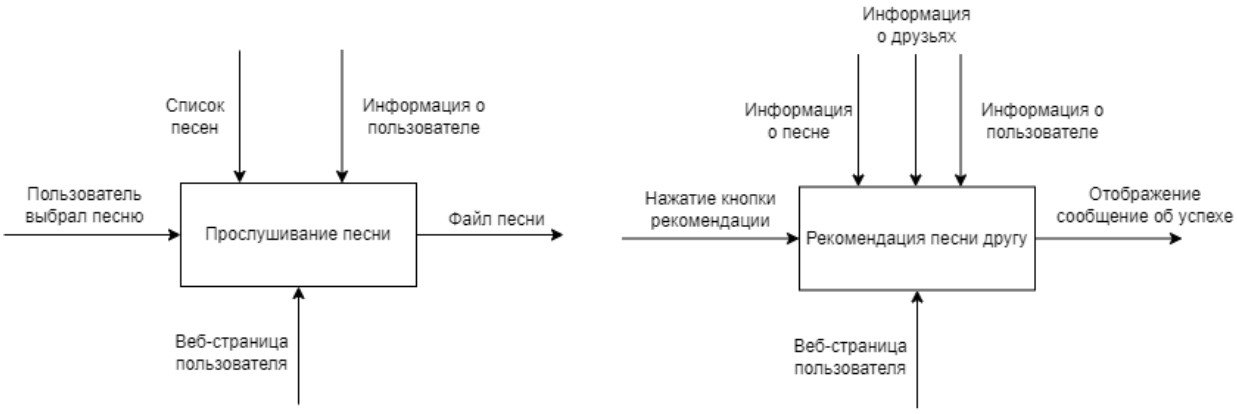

Рисунок 5 - Диаграмма IDEF0

На рисунке 5 представлена IDEF0 диаграмма для описания двух бизнес-процессов: прослушивание песни и рекомендация песни другу.

3.2.2 Диаграмма вариантов использования

Ниже приведено подробное описание диаграммы использования (см. Рисунок 6-7). Система используется следующими группами пользователей:

— Неавторизованный пользователь;

— Авторизованный пользователь;

— Администратор.

Неавторизованный пользователь может:

— Ограниченно просматривать контент (без возможности прослушать композицию);

— Сортировать контент.

Авторизованный пользователь может:

— Сортировать контент;

— Управлять своим аккаунтом. В управлении входит регистрация, вход в систему, редактирование профиля, а также возможность добавлять других зарегистрированных пользователей в друзья или удалять их;

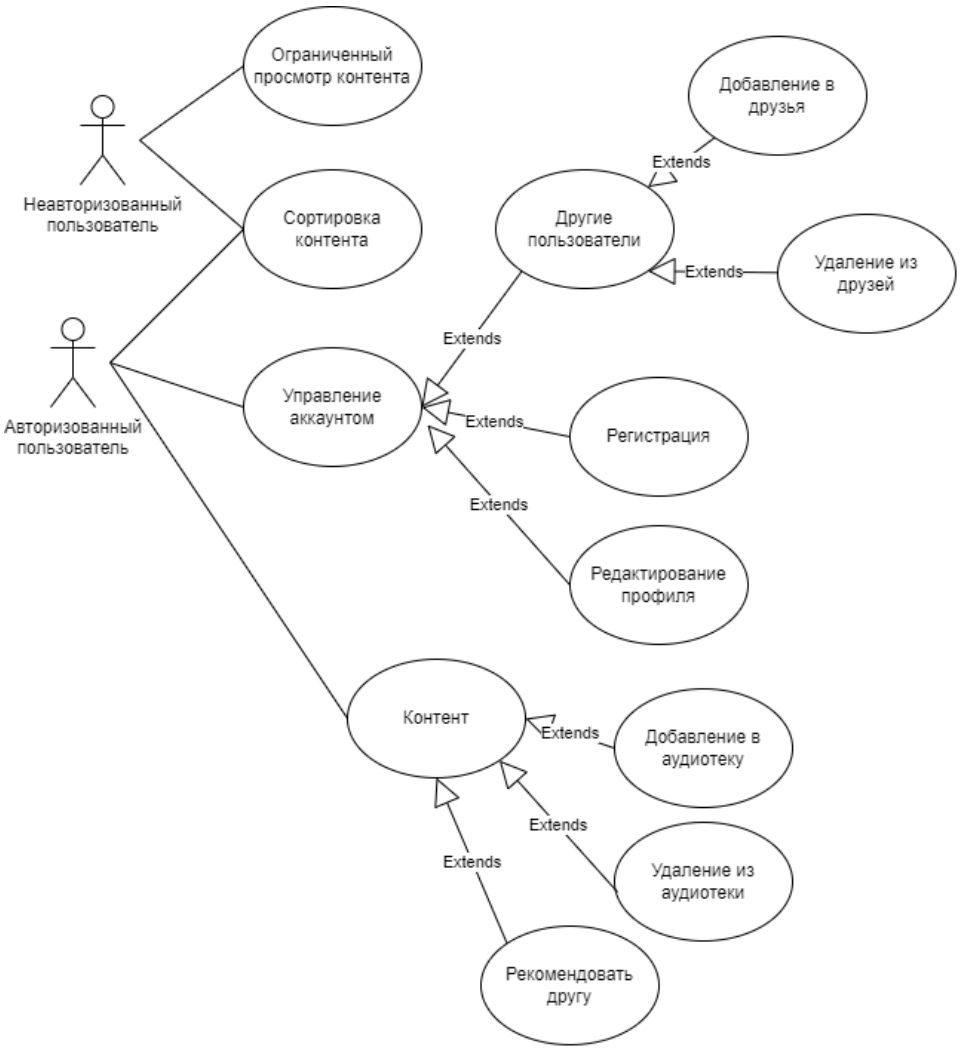

Рисунок 6 - Диаграмма вариантов использования (часть 1)

— Просматривать контент. Также авторизованный пользователь может добавлять в аудиотеку или удалять оттуда композиции, рекомендовать другу композицию.

Администратор может:

— Удалять пользователей из системы в случае нарушения соблюдений правил платформы;

— Управлять контентом: добавлять жанры, исполнителей и композиций.

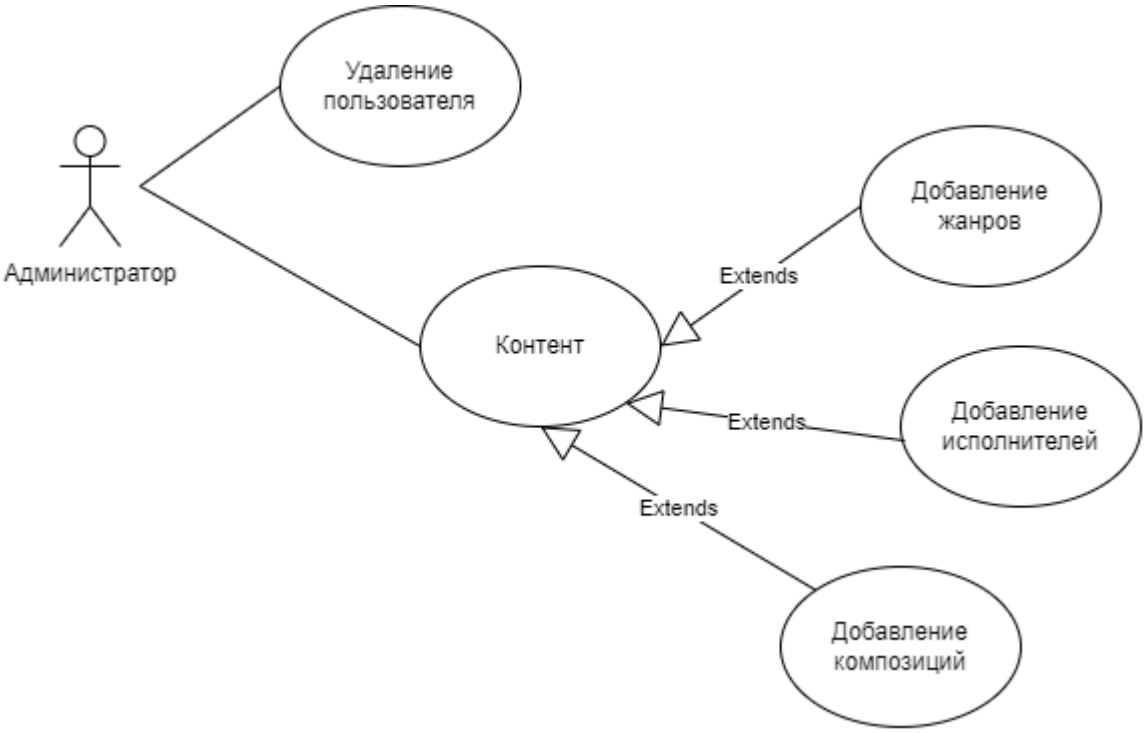

Рисунок 7 - Диаграмма вариантов использования (часть 2)

3.2.3 Диаграмма последовательности

Одной из основной целью приложения является показ пользователю композиций на основе его музыкальных предпочтений. Это достигается не только за счет возможности фильтрации контента по жанрам и исполнителю, но и с помощью учета статистики пользователя. Далее рассматривается диаграмма последовательности при нажатии прослушивания композиции (см. Рисунок 8).

При нажатии кнопки прослушивания композиции с вебсайта посылается запрос на сервер, где происходит обработка. В случае если запрос был некорректен (например, запрос отправил неавторизованный пользователь или запрашивалась несуществующая композиция), пользователю демонстрируется экран с ошибкой. В противном случае перед получением файла композиции из хранилища в базу данных записывается информация о жанре композиции и уникальный идентификатор пользователя, а также в отношение «история» заносится информация о песне.

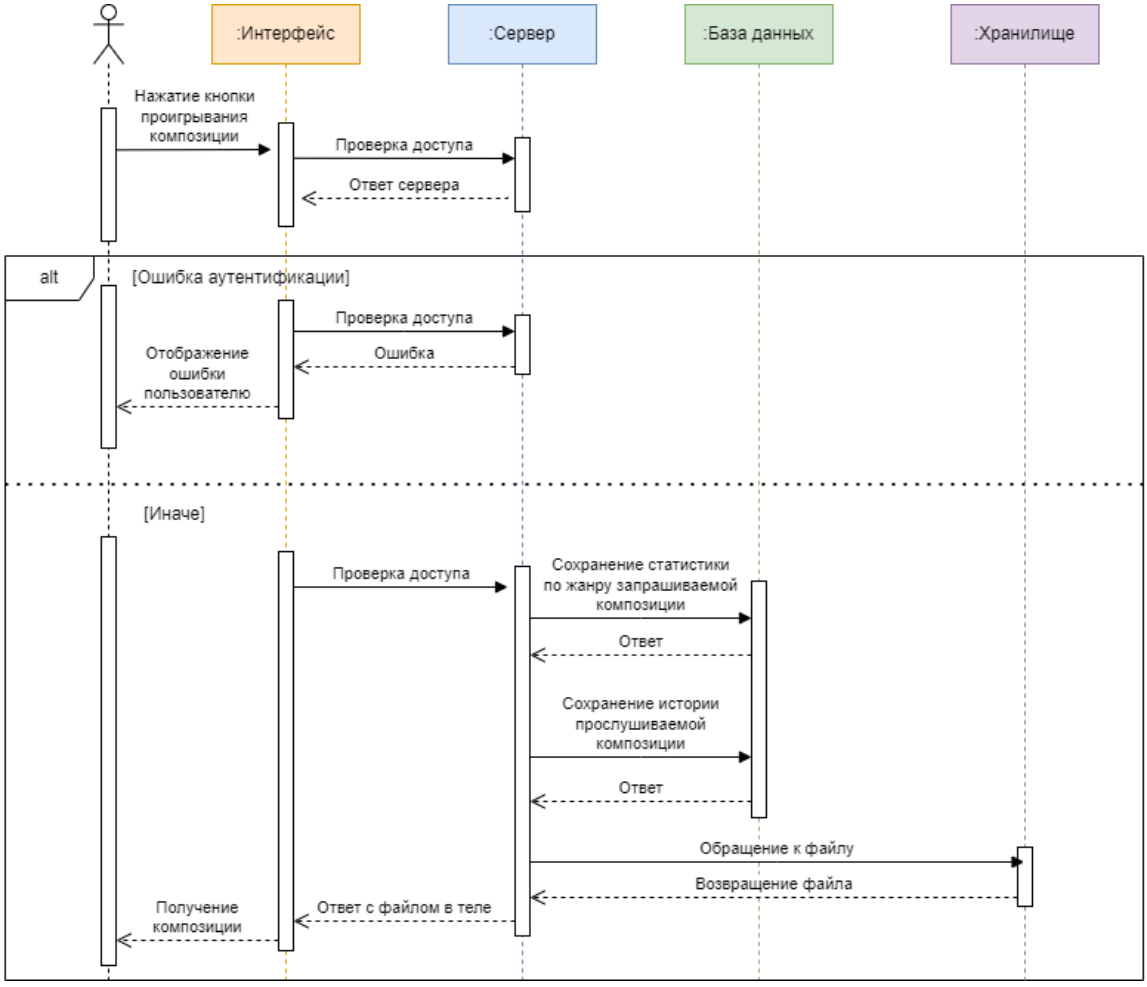

Рисунок 8 - Диаграмма последовательности при нажатии прослушивании композиции

3.2.4 Диаграмма состояний

Ниже на рисунке 9 представлена диаграмма состояний для получения песни для прослушивания пользователем.

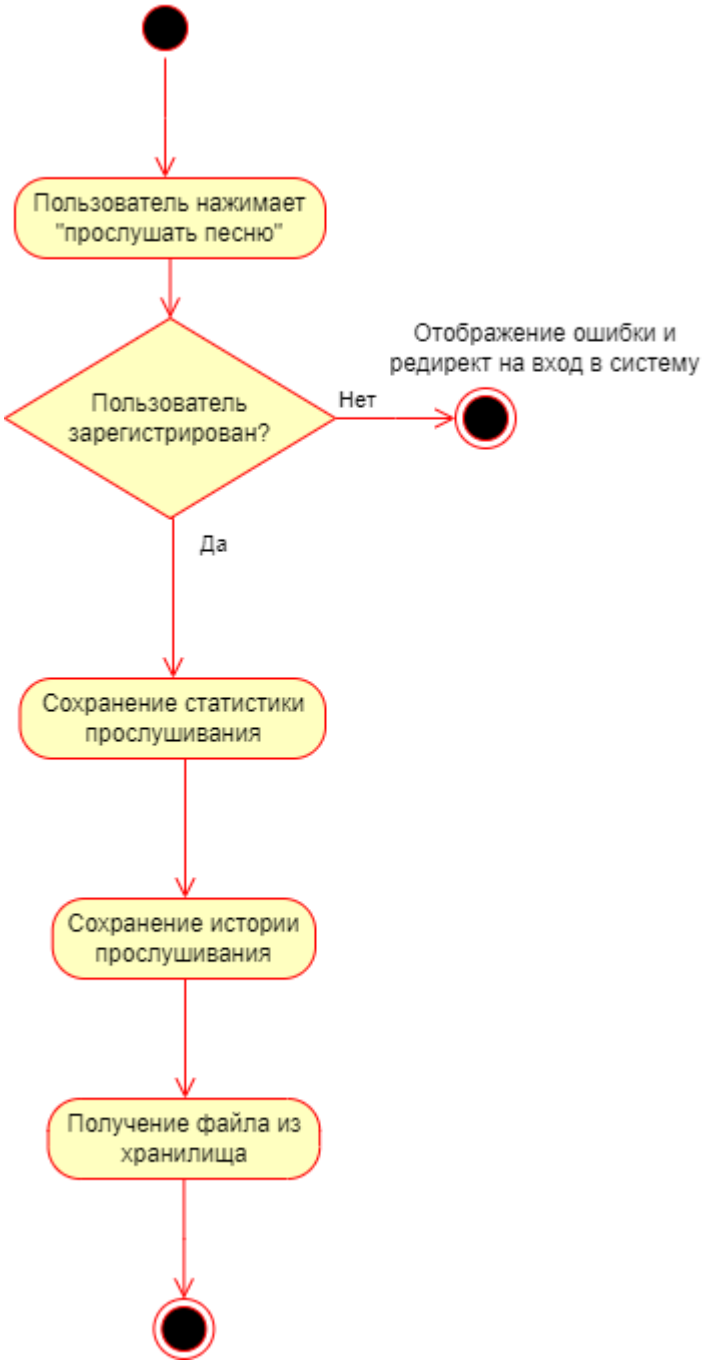

Рисунок 9 - Диаграмма состояний

3.2.5 Диаграмма классов сущностей

Для работы сервера используются основные сущности, изображенные на диаграмме классов (см. Рисунок 10). Далее следует описание классов:

— User – класс пользователя;

— Genre – класс жанра;

— Singer – класс исполнителя;

— RefreshToken – класс токена для обновления access токена, нужный для отправки авторизованных запросов;

— Verification – класс для хранения информации о верификации пользователя;

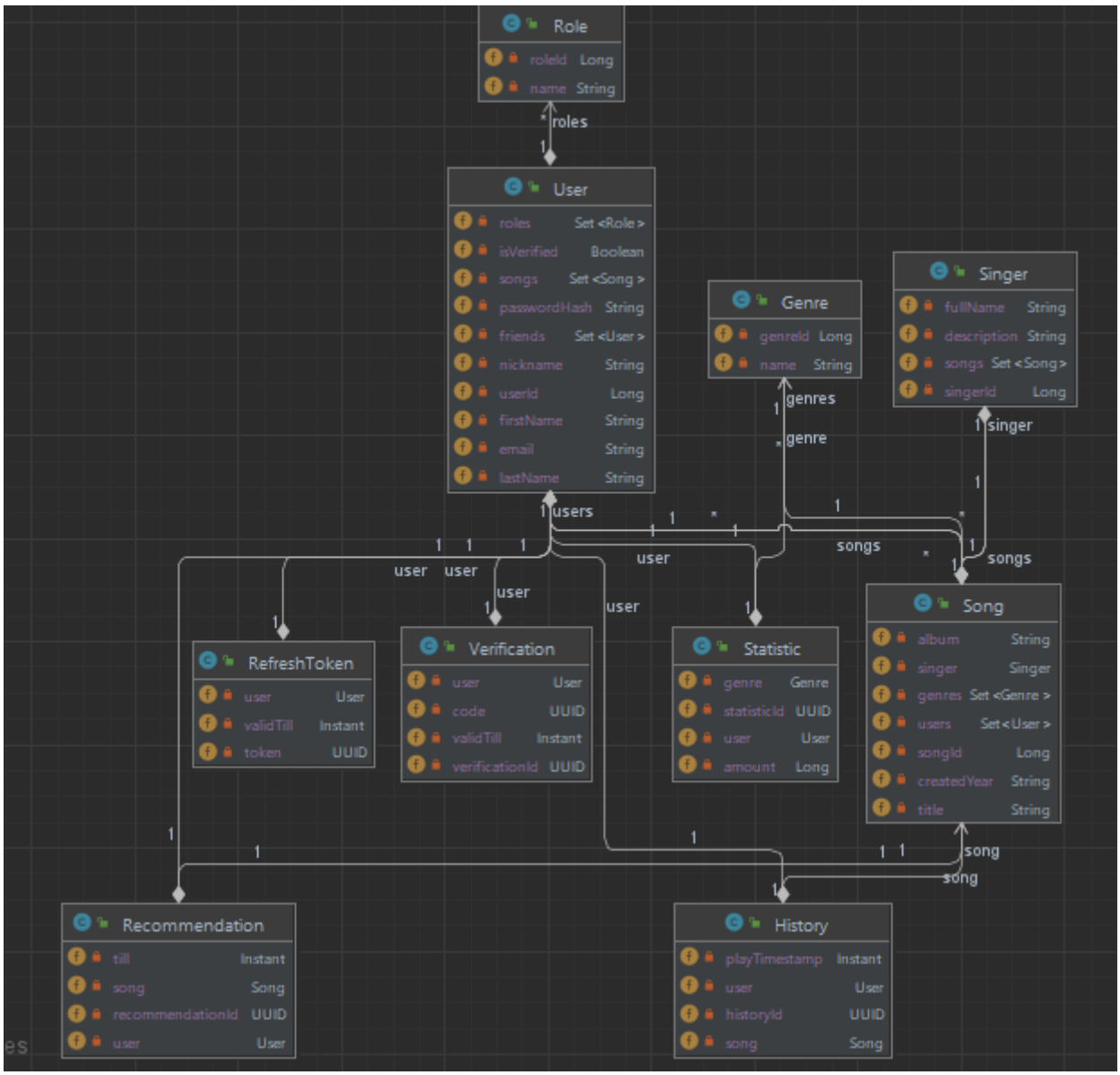

Рисунок 10 - Диаграмма классов сущностей

— Statistic – класс статистики пользователя по прослушиваемым жанрам;

— Song – класс композиции;

— History – класс истории прослушанных композиций;

— Recommendation – класс для хранении информации о рекомендованной песни другим пользователем.

3.2.6 Диаграмма развертывания

На рисунке 11 изображена диаграмма развертывания приложения. Основными компонентами приложения являются сервер, который использует базу данных и компоненты для слежения за метриками бэкенд части, и клиентская часть с веб-браузером. Для работы серверной и клиентской частей необходим выход в интернет.

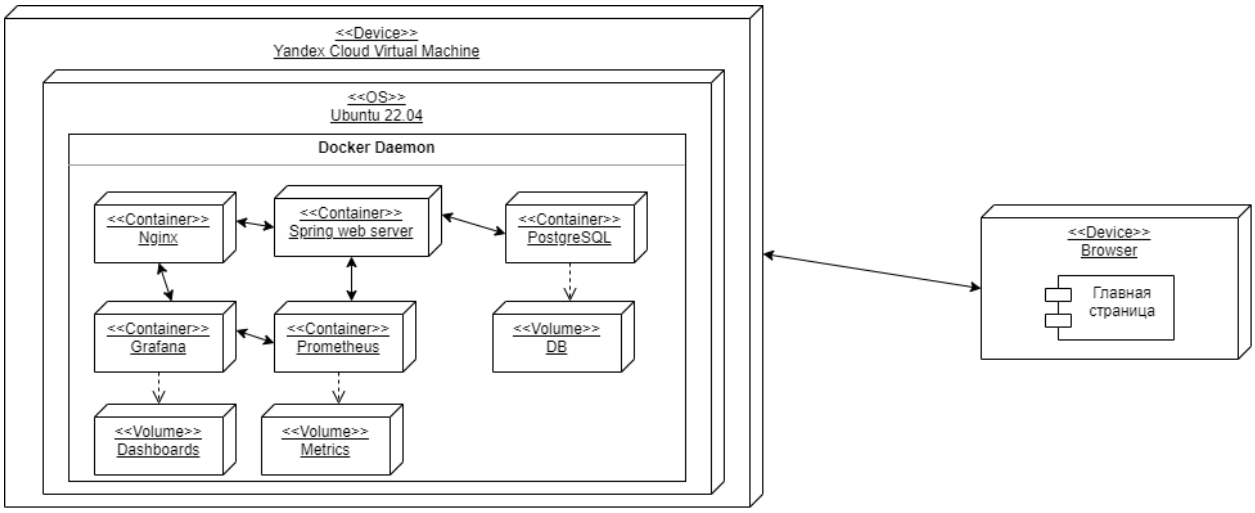

Рисунок 11 - Диаграмма развертывания

3.3 Реализация серверной части приложения

Серверная часть приложения представляет собой монолитную архитектуру. В рамках данного проекта не имело смысла выделять части логики в отдельные микросервисы, так как при неисправности основного сервиса параллельные задачи (например, очистка неактуальных рекомендаций) не должны работать. Организацию кода можно увидеть на Рисунке 12. Каждый класс помещен в соответствующую его предназначению папку, таким образом структурируя код. Папки с классами организованы в соответствии архитектуре MVC (Model-View-Controller). Данная архитектура была выбрана из-за возможности ограничения ответственности между моделью, представлением и контроллером.

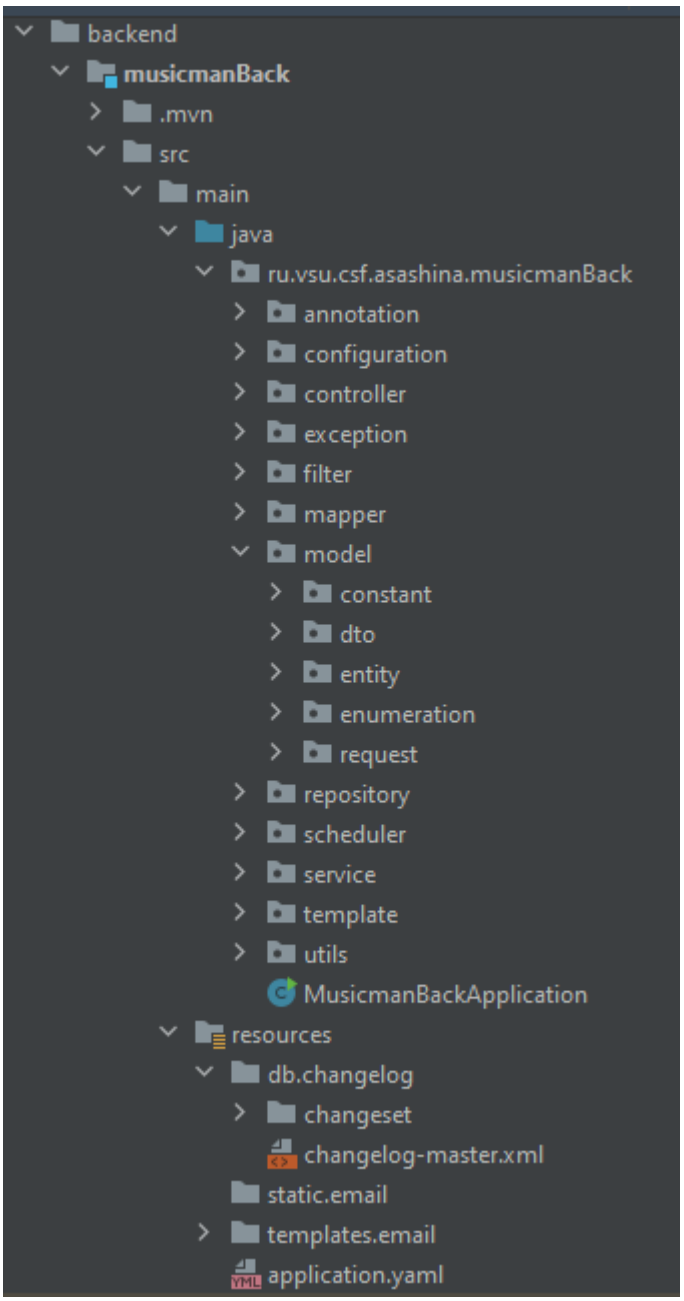

Рисунок 12 - Структура кода серверной части приложения

Для соблюдения принципов SOLID и удобочитаемости [10] разработка классов следовала цепочке model (сущности базы данных и приложения) – repository (класс, отвечающий за взаимодействие с базой данных) – service (класс с бизнес-логикой) – controller (класс, к которому происходит обращение по https запросу извне).

Далее будут подробно рассмотрены содержимое наиболее важных пакетов проекта.

3.3.1 Конфигурации

В конфигурациях хранится информация о создании классов из библиотек для обеспечения работы приложения. Одной из основной конфигурацией является SecurityConfiguration, которая отвечает за фильтрацию https запросов по access токену и определение доступа. Также реализована конфигурация для генерации Swagger документации.

3.3.2 Контроллеры

Всего реализовано 10 контроллеров, включая для обработки ошибок. Они изображены на рисунке 13. Внутри себя контроллеры содержат только сервисы в качестве полей, с которыми и взаимодействуют.

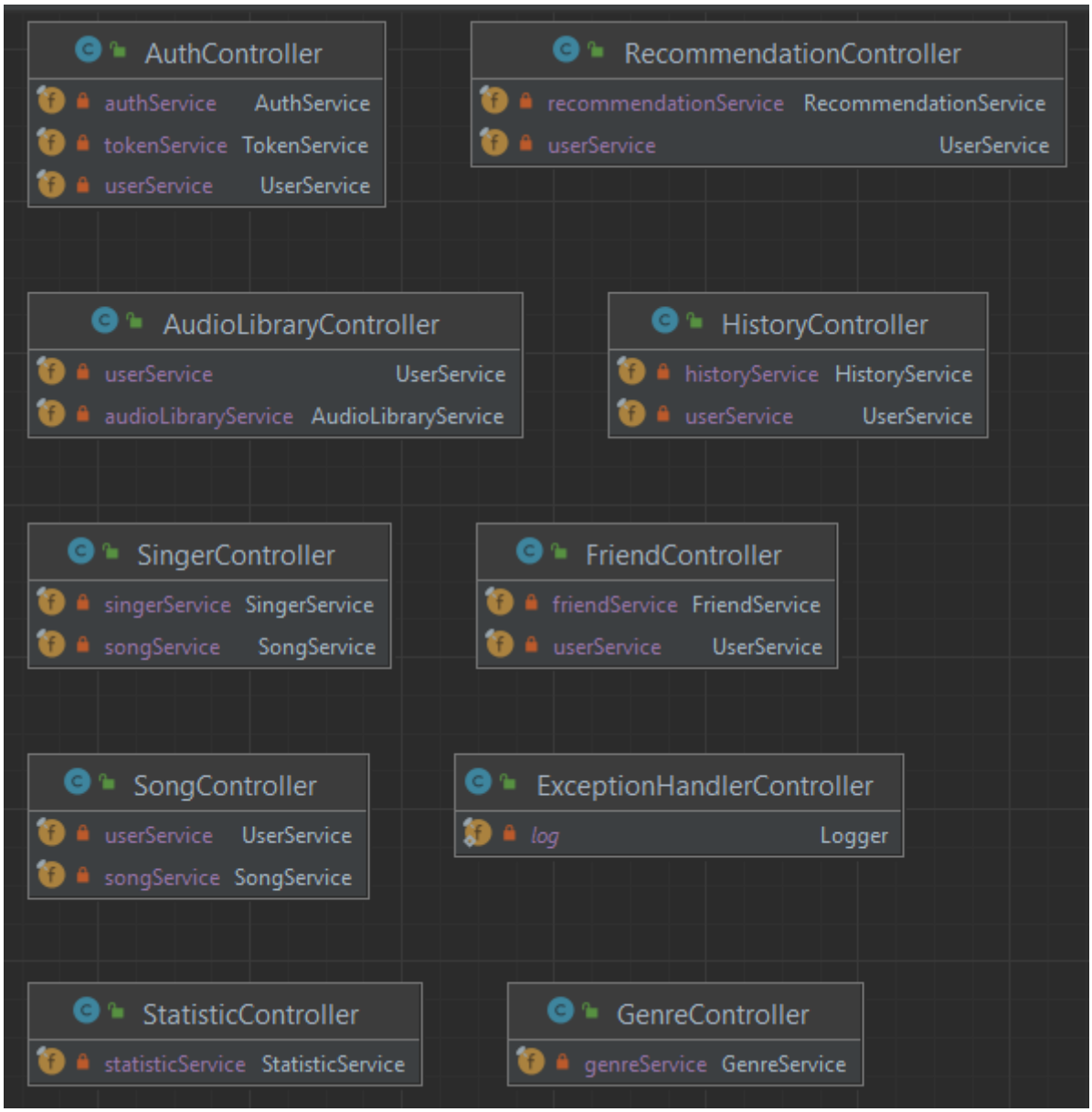

Рисунок 13 - Контроллеры

3.3.3 Мапперы

В серверной части были реализованы мапперы для миграции классов представления в сущности и наоборот. Классы представления являются DTO (data transfer object). Использование DTO необходимо для быстрой работы приложения, чтобы в качестве ответов возвращать не всю сущность с ее дочерними элементами, а лишь некоторую часть информации об объекте. Также мапперы помогают для содержания кода в чистоте, иначе сервисы бизнес-логики были перегружены set() методами.

3.3.4 Модель

Данная папка отвечает за хранение моделей, которые включают в себя DTO, сущности и объекты запросов. Последние являются телами для запросов с методами POST и PUT. Сущности представляют собой отношения из базы данных [13].

3.3.5 Репозитории

Представляют собой классы посредников между базой данных и бизнес-логикой. Все репозитории наследуются от JpaRepository, который уже реализовал простейшие операции, например, получение сущности по ее первичному ключу. Каждый репозиторий соотносится с сущностью.

3.3.6 Планировщик

Планировщик удаления рекомендаций (см. Рисунок 14) представляет собой класс, который асинхронно раз в неделю очищает отношение «рекомендации» в базе данных от неактуальных записей. Использует сервис рекомендаций для обращения к методу удаления старых.

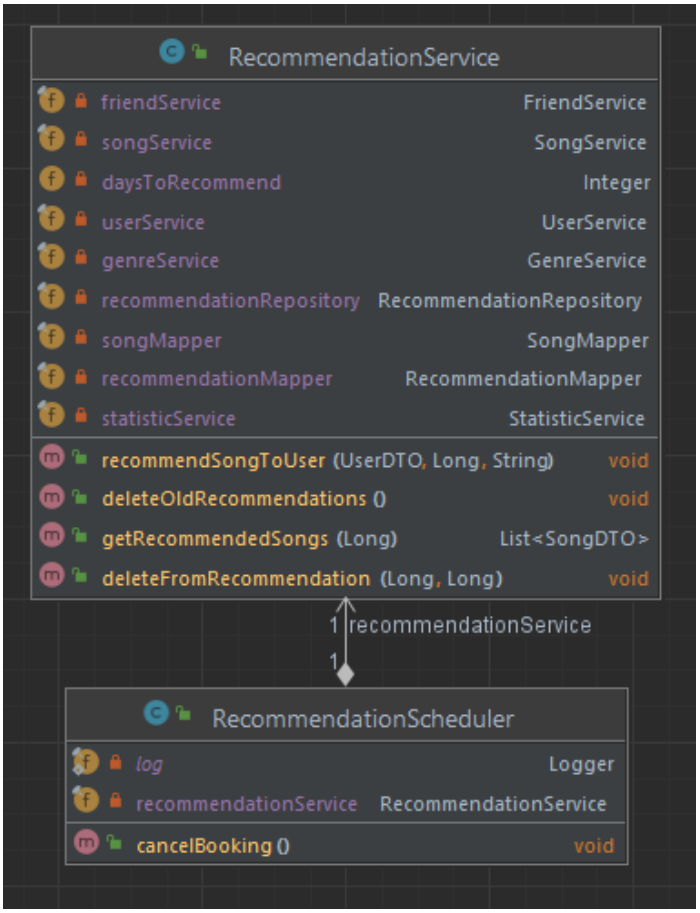

Рисунок 14 - Планировщик удаления рекомендаций

3.3.7 Сервисы

Сервисы являются классами по осуществлению бизнес-логики. Ниже рассмотрены каждые из них.

— AudioLibraryService. Представляет собой сервис для работы с личной аудиотекой пользователя. Работа с базой данной осуществляется через сервисы SongService и RecommendationService;

— AuthService. Класс-фасад, предназначенный для осуществления бизнес-логики, которая связана с профилем пользователя. Здесь реализована регистрация, вход в систему, подтверждения почты. Работа с базой данной происходит через сервисы UserService, TokenService, EmailService, VerificationService;

— EmailService. Сервис для работы с почтой. Здесь создается письмо и с помощью JavaMailSender асинхронно отправляется на указанный

электронный адрес пользователю при регистрации для его верификации;

— FriendService. Данный сервис отвечает за бизнес-логику, связанную с друзьями (удаление и добавление). Работает вместе с UserService;
— GenreService. Отвечает за осуществление логики с жанрами;
— HistoryService. Сервис, предназначенный для работы с историей прослушивания пользователя;
— RecommendationService. Является одним из важнейших сервисов, так как осуществляет логику по рекомендации песен пользователю и друзьям;
— RoleService. Сервис для работы с ролями пользователя;
— SingerService. Сервис, связанный с логикой исполнителя;
— SongService. Данный сервис отвечает за работу с песнями. В этом сервисе происходит и обращение к статистике и истории при получении файла песни;
— StatisticService. Отвечает за сбор статистики пользователя по количеству прослушанных песен по какому-либо жанру;
— TokenService. Данный сервис осуществляет создание и проверку токенов для доступа к эндпоинтам (ручкам) приложения;
— UserService. Центральный сервис серверной части приложения. Отвечает за работу с пользователем и изменение данных о нем. Валидация access токена происходит с использованием данного сервиса;
— VerificationService. Сервис верификации пользователя по электронной почте.

3.4 Реализация клиентской части приложения

Архитектурным решением для клиентской части приложения, является feature-sliced design. Проект клиентского приложения был разделен на следующие каталоги (см. Рисунок 15):

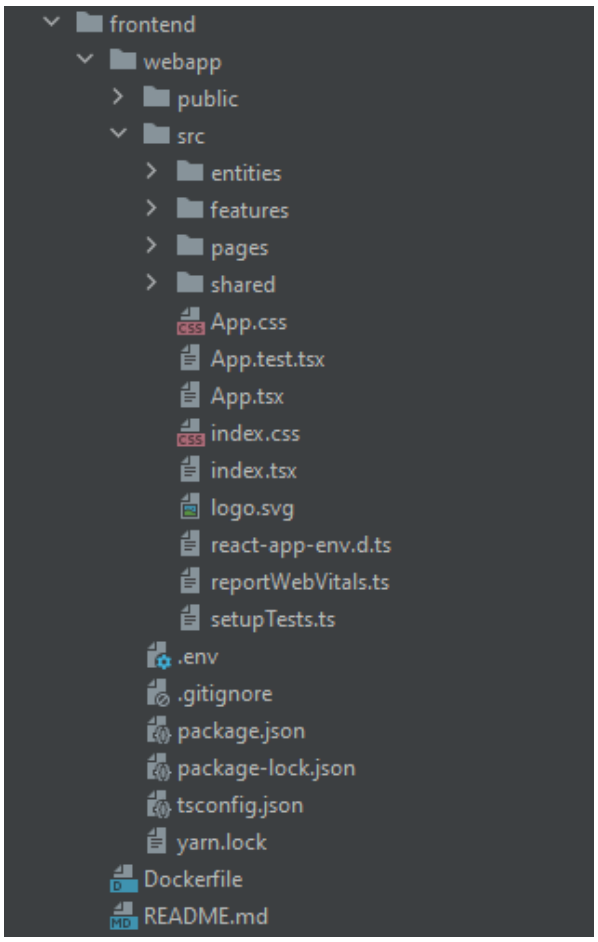

Рисунок 15 - Структура кода клиентской части приложения

— Entities – компоненты, хранящиеся в данном каталоге, являются сущностями без глобального функционала и не взаимодействуют с состоянием приложения;

— Features – компоненты, которые собираются из entity-компонентов, а также могут работать с состоянием приложения;

— Pages – компоненты, которые являются «страницами» приложения. Они собираются из feature-компонентов, имеют доступ к состоянию приложения;

— Shared – компоненты, которые являются побочными, но при этом используются во всем приложении. В данном каталоге, находятся утилиты, функции для взаимодействия с API.

Такая структура позволяет разделить функциональность на маленькие и независимые куски кода, обеспечивая повторное использование компонентов и легкость сопровождения.

3.4.1 Описание страниц веб-приложения

Ниже представлены изображения клиентской части и их описания.

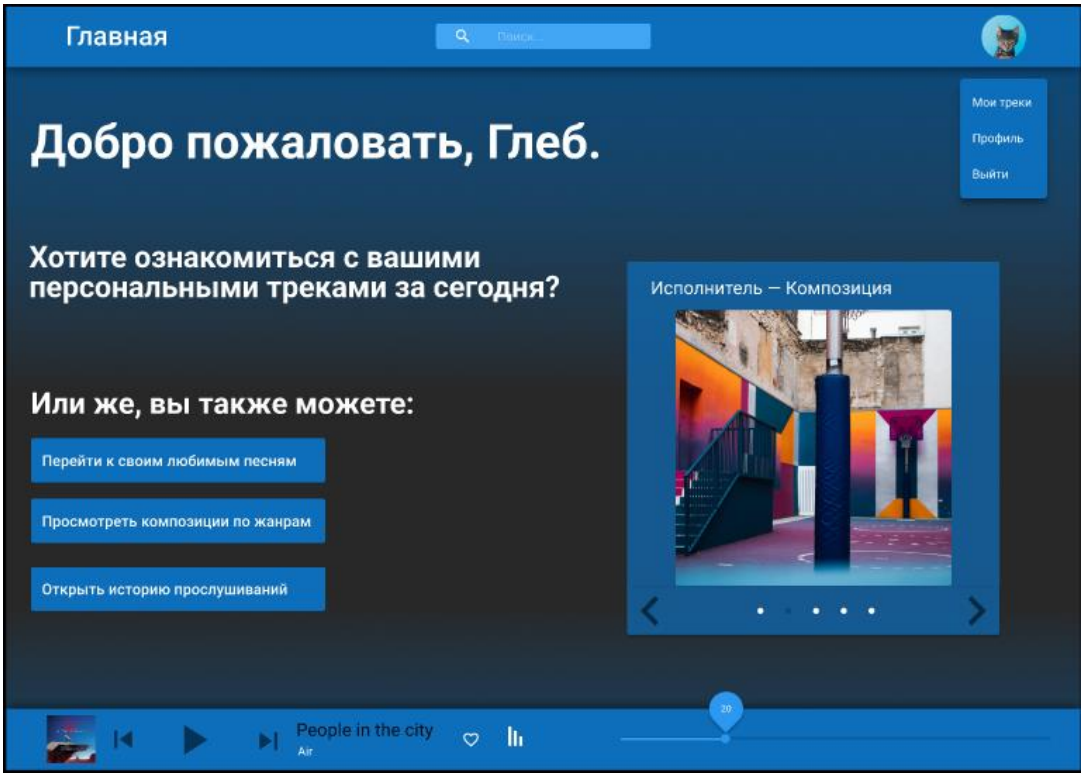

Рисунок 16 - Главная страница

На рисунке 16 изображена главная страница приложения. С нее пользователь может перейти к эквалайзеру (см. Рисунок 17), к своим песням (см. Рисунок 18), просмотреть композиции по жанру, а также открыть историю прослушивания. Если пользователь уже прослушивал песню, то она будет проигрываться в нижней панели интерфейса. При нажатии на иконку профиля пользователь может просмотреть свои треки, профиль или выйти из системы.

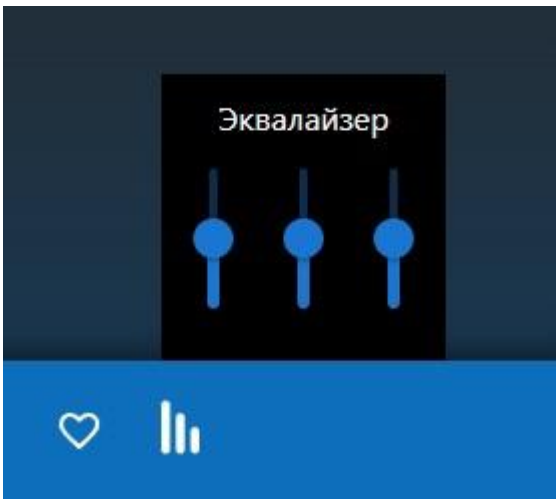

Рисунок 17 - Эквалайзер

Эквалайзер выглядит следующим образом (см. Рисунок 17). Левая колонка отвечает за низкие частоты, средняя за средние и правая за высокие.

Верхняя и нижняя панели одинаковы на всех страницах (кроме регистрации и входа). Страница с треками пользователя показывает список добавленных песен. При нажатии на конкретную песню по ней выводится информация. В троеточии можно выбрать рекомендовать песню другу или удалить ее.

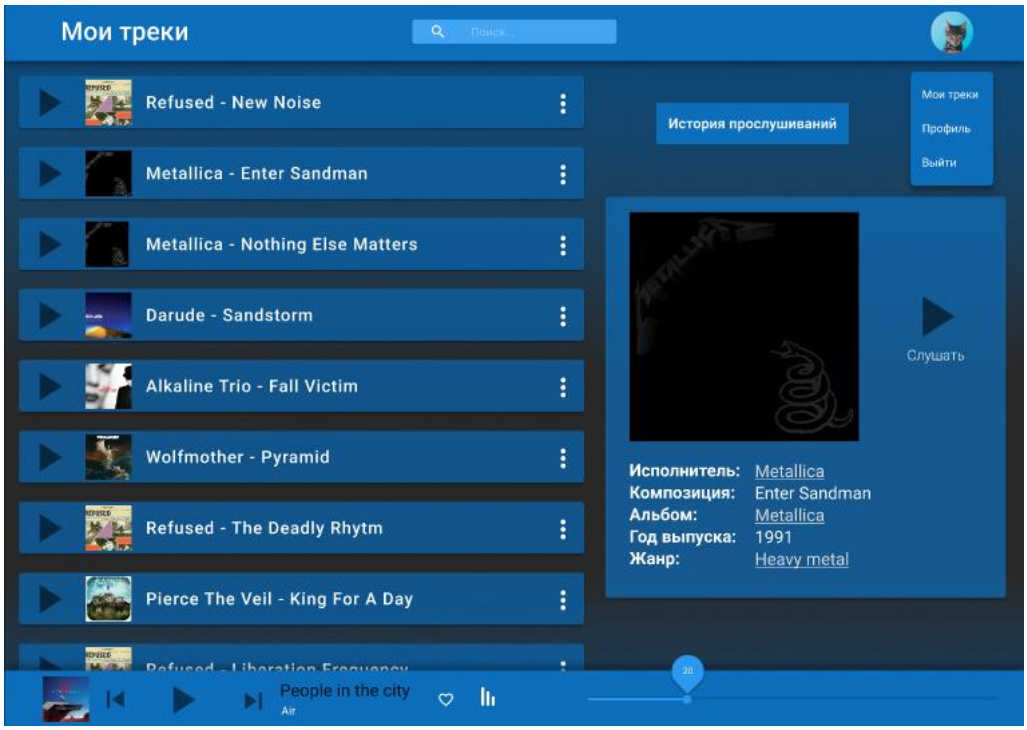

Рисунок 18 - Страница с треками пользователя

Профиль пользователя показывает информацию о нем. Кнопка «список друзей» ведет на страницу с друзьями (см. Рисунок 20). На странице с друзьями пользователь может их добавлять или удалять.

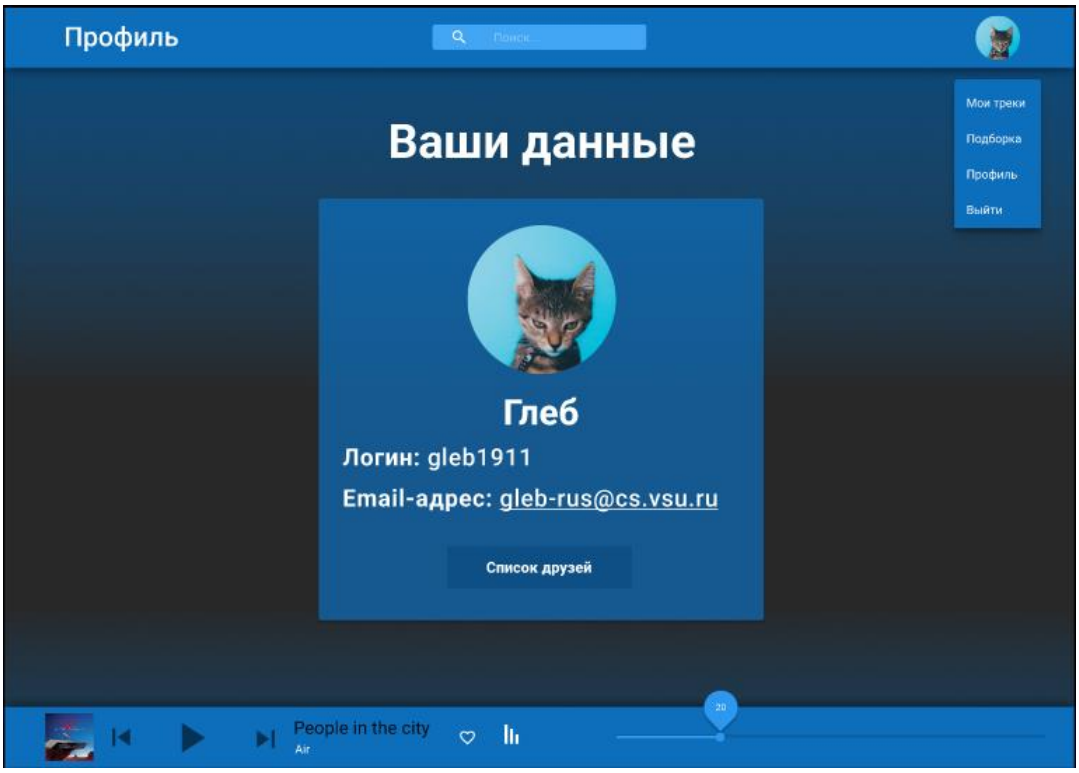

Рисунок 19 - Профиль пользователя

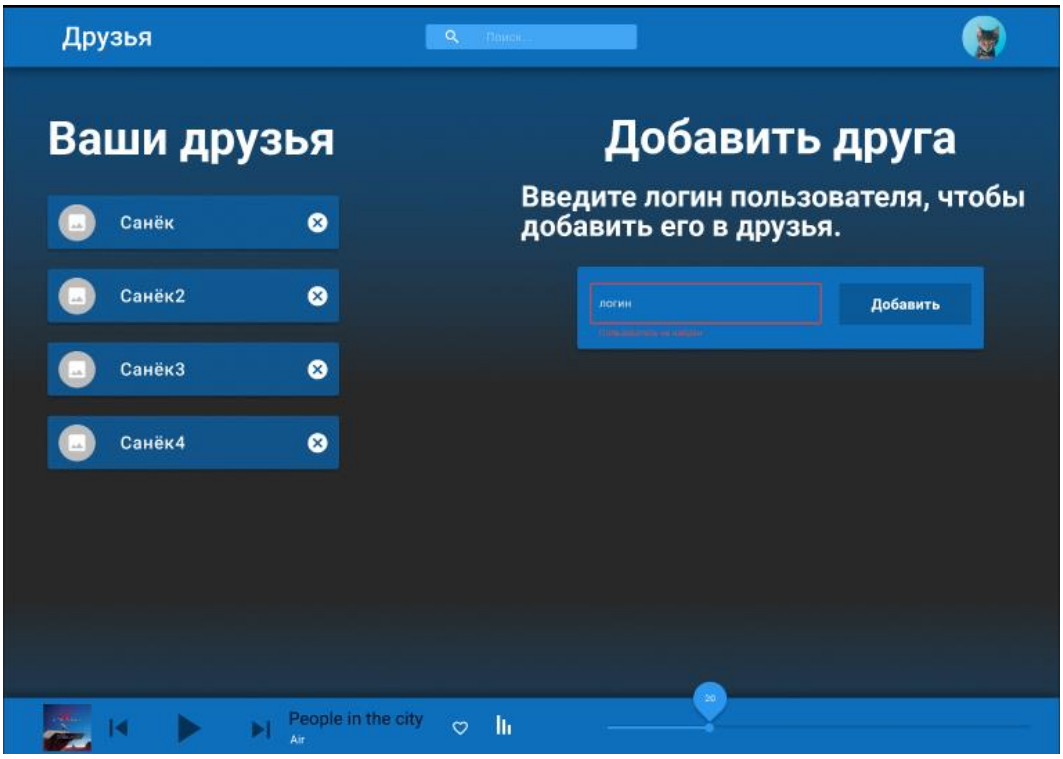

Рисунок 20 - Страница с друзьями и их менеджментом

Также выполнена страница (см. Рисунок 21) с поиском песен по жанрам и по названию. Слева отображаются жанры, а справа найденные песни.

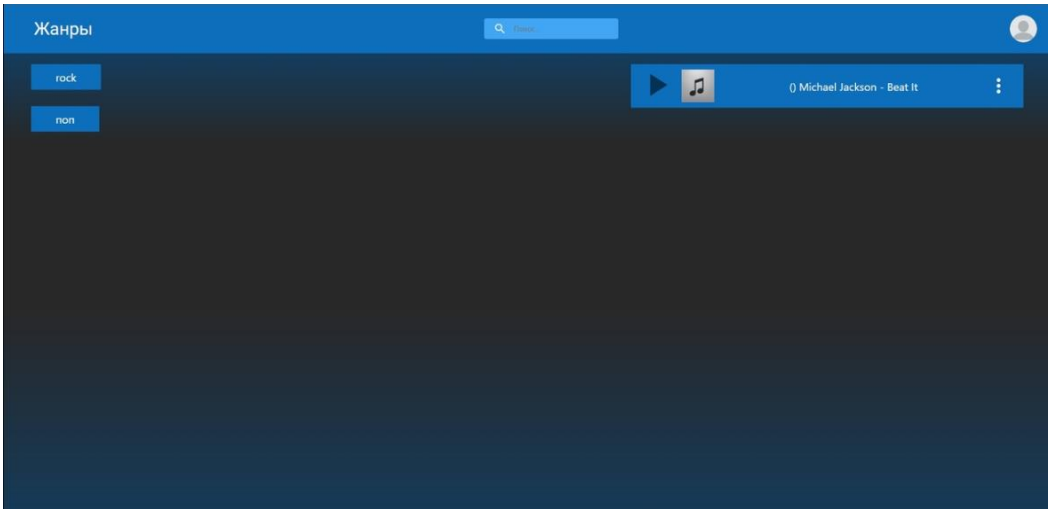

Рисунок 21 - Страница с поиском песен по жанрам и названию

Страница администратора похожа на страницу пользователя, однако в отличие от обычных посетителей сайта имеет доступ к панели администрирования (см. Рисунок 22). Администратор может добавить жанр, исполнителя и песню.

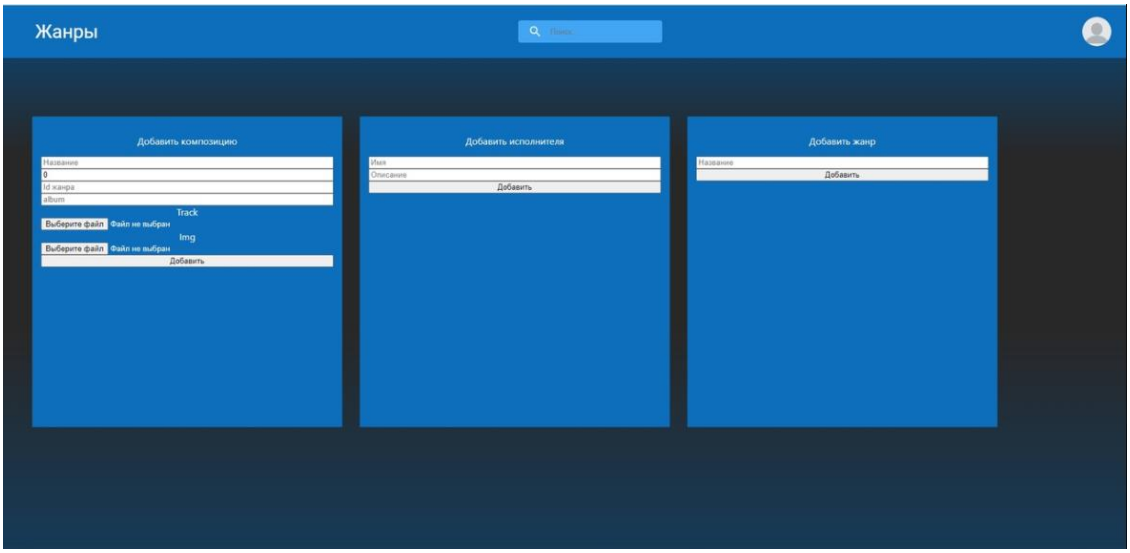

Рисунок 22 - Панель администратора

4 Метрики

Помимо ведения аналитики за пользователем по количеству прослушанных песен в том или ином жанре, был реализован доступ к просмотру панелей с метриками Java Virtual Machine (см. Рисунок 23). Данные метрики были получены с помощью системы мониторинга Prometheus [11] и изображены с помощью интерфейса Grafana [12].

Рисунок 23 - Метрики Java Virtual Machine

Одной из целью заведения метрик являлась возможность слежением за расходом памяти и скоростью выполнения запросов. Данную информацию необходимо учитывать для дальнейшего улучшения системы и ее логики особенно в тех местах, где выполняется выгрузка большого объема данных (например, выгрузка песен в аудиотеке).

5 Тестирование

Тестирование было проведено в ручном формате. Для всех реализованных эндпоинтов были написаны тест-кейсы. С помощью Swagger документации и запросов через программу Postman были протестированы все важные сценарии.

На серверной части приложения был реализован тест с поднятием контекста фреймворка Spring. Данный тест может стать основой CI проекта (continuous integration) и помогает удостовериться, что изменения на бэкенде не повлияли на работоспособность сервера.

Заключение

В ходе выполненной работы было реализовано приложение Musicman для прослушивания музыки. Веб-приложение обладает визуальным интерактивным интерфейсом, который необходим для взаимодействия пользователя с приложением и позволяет выводить на экран всю нужную информацию. Были выполнены следующие задачи:

- Разработана серверная часть приложения, развернутая на виртуальном машине;
- Разработана клиентская часть, необходимая для взаимодействия с логикой приложения;
- С помощью API была реализована связь между клиентской и серверной частями;
- Приложение учитывает статистику по композициям для рекомендации, осуществляет логику по возможно рекомендации песни и другу и использует эквалайзер.

Список использованных источников

1. Миннебаева, Р.А. Анализ рынка стриминговых сервисов / Р.А. Миннебаева, Р.А. Миннебаев // Форум молодых ученых. – Саратов, 2018. - №5(21). – С. 666-670.
2. Российский рынок музыкального стриминга в 2022 году сократился на 30-50% до 5-7 млрд рублей – исследование [Электронный ресурс] : статья на сайте новостей про информационные технологии «vc.ru». – Режим доступа: https://vc.ru/services/600606-rossiyskiy-rynok-muzykalnogo-striminga-v-2022-godu-sokratilsya-na-30-50-do-5-7-mlrd-rubley-issledovanie
3. Блох Д. Java: Эффективное программирование / Д. Блох – М. : Вильямс, 2018 – 464с.
4. Spring [Электронный ресурс] : официальный сайт фреймворка Spring. – Режим доступа: https://spring.io/
5. React.js [Электронный ресурс] : документация фреймворка React. – Режим доступа: https://react.dev/
6. PostgreSQL [Электронный ресурс] : официальный сайт. – Режим доступа: https://www.postgresql.org/
7. Liquibase [Электронный ресурс] : официальный сайт. – Режим доступа: https://www.liquibase.org/
8. Effector [Электронный ресурс] : документация. – Режим доступа: https://effector.dev/
9. Material UI [Электронный ресурс] : официальный сайт. – Режим доступа: https://mui.com/
10. Мартин Р. Чистый код: создание, анализ и рефакторинг / Р. Мартин – СПб. : Питер, 2022 – 464с.
11. Prometheus [Электронный ресурс] : официальная документация. – Режим доступа: https://prometheus.io/docs/introduction/overview/

12. Grafana [Электронный ресурс] : официальный сайт. – Режим доступа: https://grafana.com/
13. Стриминговый сервис «Musicman» [Электронный ресурс] : репозиторий на веб-хостинге «GitHub». – Режим доступа: https://github.com/TP-6-1-3/StreamingService