



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ,

информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему:

Разработка веб-приложения социальной сети с функциями
музыкального стримингового сервиса

по дисциплине Компьютерные сети

Студент гр. ИУК4-72Б _____ (_____ Сафронов Н.С.)
(подпись) (Ф.И.О.)

Руководитель _____ (_____ Белов Ю.С.)
(подпись) (Ф.И.О.)

Оценка руководителя _____ баллов _____
30-50 (дата)

Оценка защиты _____ баллов _____
30-50 (дата)

Оценка работы _____ баллов _____
(оценка по пятибалльной шкале)

Комиссия: _____ (_____ Красавин Е.В.)
(подпись) (Ф.И.О.)

_____ (_____ Белов Ю.С.)
(подпись) (Ф.И.О.)

_____ (_____ Гагарин Ю.Е.)
(подпись) (Ф.И.О.)

Калуга, 2023

Калужский филиал
федерального государственного бюджетного образовательного учреждения высшего образования
**«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой ИУК4

(Гагарин Ю.Е.)

« 08 » сентября 2023 г.

**ЗАДАНИЕ
на выполнение курсового проекта**

по дисциплине Компьютерные сети

Студент Сафронов Н.С. ИУК4-72Б
(фамилия, инициалы, индекс группы)

Руководитель Белов Ю.С.
(фамилия, инициалы)

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 10 нед., 100% к 14 нед.

1. Тема курсового проекта

Разработка веб-приложения социальной сети с функциями музыкального стримингового сервиса

2. Техническое задание

Разработать веб-приложения социальной сети с функциями музыкального стримингового сервиса

3. Оформление курсового проекта

3.1. Расчетно-пояснительная записка на 42 листах формата А4.

3.2. Перечень графического материала КП (плакаты, схемы, чертежи и т.п.)

1. Структурная схема

2. Демонстрационный чертеж

3. ER-диаграмма

Дата выдачи задания « 08 » сентября 2023 г.

Руководитель курсового проекта _____ / Белов Ю.С.
(подпись) (Ф.И.О.)

Задание получил _____ / Сафронов Н.С. / « 08 » сентября 2023 г.
(подпись) (Ф.И.О.)

Примечание:

Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

РЕФЕРАТ

Расчетно-пояснительная записка 42 с., 23 рисунка, 1 таблица, 15 источников.

Разработка веб-приложения социальной сети с функциями музыкального стримингового сервиса.

Объектом курсового проекта являются социальные сети и музыкальный стриминг.

Цель проекта – разработка web-приложения социальной сети с функциями музыкального стриминга.

Поставленные задачи решаются путем разработки социальной сети с функциями музыкального стримингового сервиса.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1. АНАЛИЗ ТРЕБОВАНИЙ И ТЕХНОЛОГИЙ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ	6
1.1. Основные требования к разрабатываемой системе	6
1.2. Анализ аналогов и прототипов	7
1.3. Обоснование выбора инструментов и платформы для разработки	11
2. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ	18
2.1. Разработка структуры системы	18
2.2. Структура базы данных	22
2.3. Описание организации диалога с пользователем	26
3. КОНТРОЛЬ КАЧЕСТВА И ИНТЕГРАЦИЯ КОМПОНЕНТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ	29
3.1. Руководство администратора	29
3.2. Руководство пользователя	29
3.3. Руководство исполнителя	36
ЗАКЛЮЧЕНИЕ	40
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	41

ВВЕДЕНИЕ

Актуальность темы курсового проекта обусловлена тем, что в наше время музыка по праву занимает своё место одних из крупнейших разделов индустрии развлечений. Она стала неотъемлемой частью досуга множества людей: как музыкантов, так и слушателей. Разработка подобного программного продукта, которым смогут пользоваться как создатели, так и потребители музыкального контента, довольно трудоёмкая задача, требующая проработки многих аспектов. Одним из основных функциональных блоков такой социальной сети является музыкальный стриминговый сервис. Пользователи могут слушать свои любимые песни и создавать плейлисты. Кроме того, в социальной сети можно общаться с другими пользователями и делиться музыкой.

Объектом курсового проекта являются социальные сети и музыкальный стриминг.

Предметом исследования курсового проекта является реализация социальной сети с функциями стримингового сервиса.

Целью проекта является разработка web-приложения социальной сети с функциями музыкального стриминга.

Для достижения поставленной цели решаются следующие задачи:

1. Выполнить анализ предметной области;
2. Провести сравнительный анализ существующих аналогов;
3. Определить оптимальную структуру системы;
4. Осуществить выбор средств реализации программного продукта, соответствующего выбранной структуре;
5. Реализовать базу данных и программные компоненты системы;
6. Осуществить тестирование компонентов;
7. Разработать сопроводительную документацию.

1. АНАЛИЗ ТРЕБОВАНИЙ И ТЕХНОЛОГИЙ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

1.1. Основные требования к разрабатываемой системе

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- разрабатываемое приложение должно иметь мобильную и браузерную версии;
- разрабатываемое приложение должно иметь удобный интерфейс для взаимодействия с пользователем, быть клиент-ориентированным и интуитивно понятным в использовании;
- пользователь должен иметь возможность регистрации и авторизации в приложении;
- при вводе некорректных личных данных пользователь должен видеть сообщение об ошибке;
- пользователь должен иметь возможность просматривать и редактировать свой профиль после регистрации и авторизации;
- пользователь может прослушивать доступные для него плейлисты и управлять процессом воспроизведения с помощью интерфейса музыкального плеера;
- пользователь должен иметь возможность создания плейлистов, их редактирования и распространения среди других пользователей;
- исполнитель может выкладывать и распространять на платформе свои композиции;
- правообладатель обладает возможностью пожаловаться на нарушение его авторских прав, которые должен рассматривать администратор платформы;

- исполнитель должен иметь возможность объединять выложенные композиции в альбомы;
- пользователи должны быть доступны типовые функции социальных сетей: поставить отметку «Нравится», подписаться на автора, оставлять комментарии;
- приложение должно обладать рекомендательной системой на основе прослушанных пользователем композиций и тегов, установленных автором;
- пользователю должна быть доступна лента – последние выложенные композиции от отслеживаемых авторов и рекомендуемые для него новые исполнители в виде списка.

Система требует наличия пользователей двух категорий – рядовых пользователей сервиса, которые также могут являться и исполнителями, и администраторов, отвечающих за работу с жалобами пользователей и правообладателей на неправомерно выложенный контент.

Защита информации осуществляется разграничением прав доступа – обычные пользователи не должны иметь возможность вносить изменения, которые могут повлиять на функционирование системы. Также разрабатываемое приложение должно гарантировать пользователю защиту его данных. При указании соответствующих режимов приватности исполнителю также должна быть гарантирована защита объектов его творчества.

1.2. Анализ аналогов и прототипов

В настоящее время существуют аналоги, предоставляющие подобный функционал. Среди аналогов можно выделить следующие программные решения: SoundCloud, Spotify, Bandcamp.

SoundCloud

SoundCloud является уникальным стриминговым сервисом, который предлагает не только множество музыкальных треков, но также функции

социальных сетей. В отличие от других платформ, SoundCloud позволяет не только слушать музыку, но и делиться своими треками с другими пользователями. Это создает живое сообщество музыкантов, диджеев и слушателей, обеспечивая уникальный опыт.

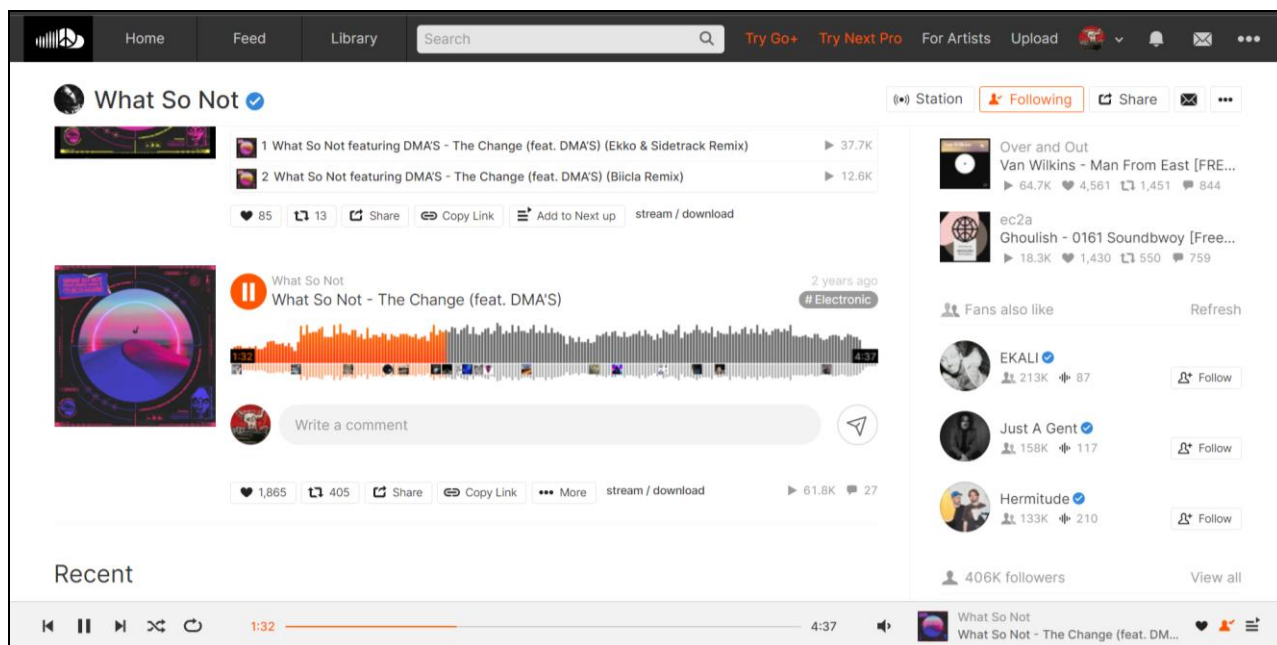


Рисунок 1.1. Интерфейс веб-приложения «SoundCloud»

На SoundCloud пользователи могут стать активными участниками, оставлять комментарии, ставить лайки и репосты, а также подписываться на профили других пользователей. Это позволяет взаимодействовать с артистами, находить новые музыкальные таланты и строить свою аудиторию. Кроме того, SoundCloud предоставляет возможность создавать и прослушивать плейлисты.

В контексте анализа аналоговых стриминговых сервисов с функциями социальных сетей, SoundCloud является уникальным игроком, который акцентирует внимание на связи и взаимодействии между музыкантами и слушателями. Он предлагает демократичную платформу для продвижения музыкальных талантов и способствует расширению музыкальной культуры. В целом, SoundCloud предоставляет возможность глубокого погружения в мир музыки, где пользователи сами становятся создателями и потребителями, а также находят нового вдохновения и источники музыкального творчества. В

данный момент SoundCloud заблокирован на территории Российской Федерации.

Spotify

Spotify – это один из ведущих стриминговых сервисов мирового уровня, предлагающий широкий доступ к миллионам треков со всего мира.

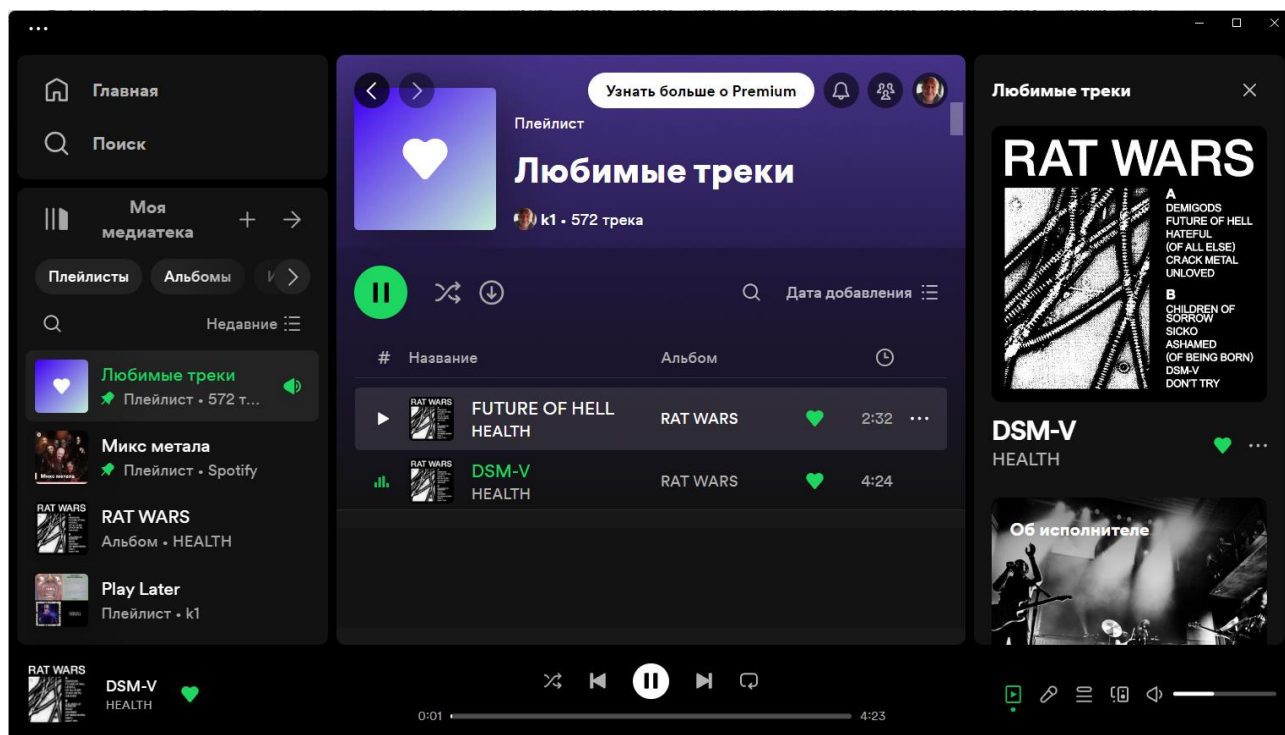


Рисунок 1.2. Интерфейс веб-приложения «Spotify»

На Spotify пользователи имеют возможность создавать собственные профили, настраивать музыкальные предпочтения и выбирать плейлисты, основанные на их вкусах. Пользователи могут следить за плейлистами других пользователей, делиться своими открытиями и даже совместно создавать плейлисты с друзьями. Это создает музыкальное сообщество, которое помогает пользователям находить новую музыку, обмениваться рекомендациями и поддерживать связи в мире музыки.

Несмотря на уход с российского рынка, Spotify остается значимым и востребованным стриминговым сервисом в мировом масштабе, предоставляя пользователям удобство и разнообразие в музыкальном контенте.

Bandcamp

Bandcamp — это уникальная платформа для музыкантов и независимых лейблов. Одной из ключевых особенностей Bandcamp является его поддержка независимых артистов, которые могут продавать свою музыку, мерч и билеты на концерты непосредственно своим поклонникам.

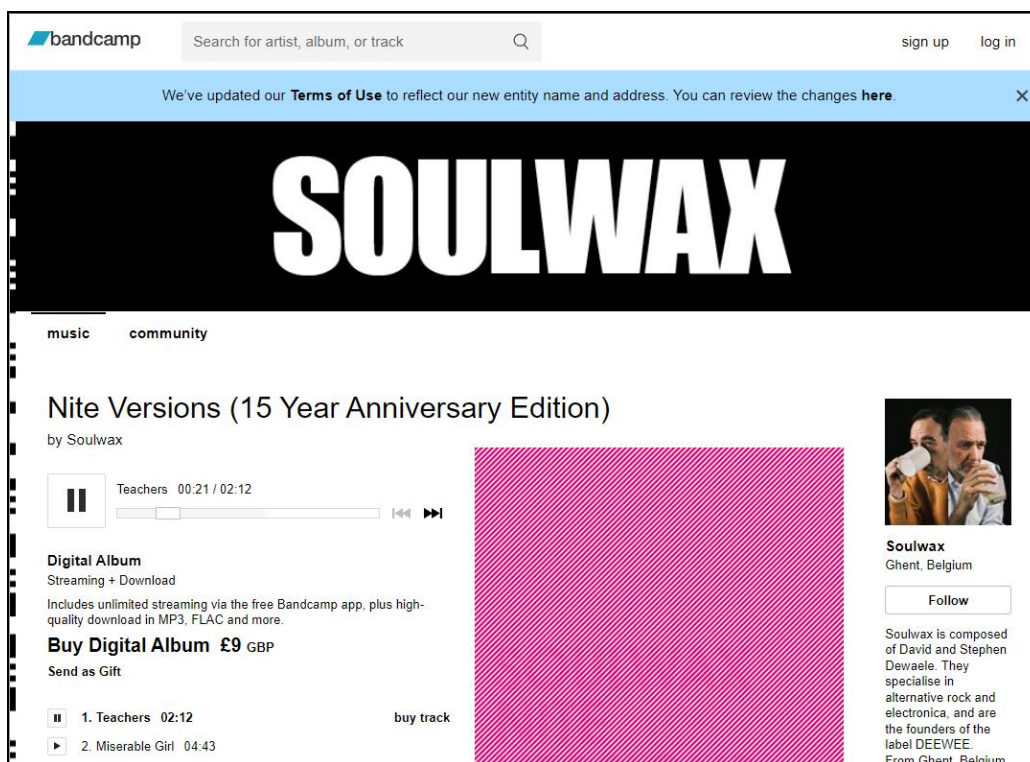


Рисунок 1.3. Интерфейс веб-приложения «Bandcamp»

Пользователи Bandcamp имеют возможность открыть собственные профили и подписываться на музыкантов, чья музыка их интересует. Это позволяет пользователям получать уведомления о новых релизах и мероприятиях, а также обмениваться комментариями, рецензиями и рекомендациями с другими фанатами и музыкантами. Функция социальных сетей Bandcamp стимулирует активное взаимодействие в музыкальном сообществе, способствуя обнаружению новых талантов и созданию тесной связи между музыкантами и их поклонниками.

Однако, Bandcamp не предоставляет полноценный стриминговый сервис, а вместо этого сосредотачивается на продаже и поддержке музыки

независимых артистов. К сожалению, Bandcamp также не доступен на российском рынке.

1.3. Обоснование выбора инструментов и платформы для разработки

Обоснование выбора инструментов для программирования клиентской части приложения

JavaScript

JavaScript — это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах.

JavaScript также имеет множество полезных библиотек, таких как jQuery, React и AngularJS, что делает его еще более привлекательным для разработки веб-приложений. Кроме того, JavaScript используется для создания различных приложений на мобильных устройствах, что значительно расширяет его область применения.

Еще одним преимуществом JavaScript является его простая интеграция с другими технологиями, такими как HTML и CSS. Эта возможность позволяет упростить разработку и использование внешних библиотек.

В целом, выбор JavaScript в качестве языка разработки обусловлен его эффективностью, популярностью, гибкостью, обширной поддержкой и возможностями интеграции с другими технологиями.

React

Для эффективного рендеринга компонентов интерфейса в DOM используется React.

React — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость. В качестве

библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, Redux и GraphQL.

Выбор React.js обоснован его эффективностью, гибкостью и удобством использования, а также его популярностью, большим сообществом разработчиков и наличием многочисленных плагинов и библиотек, которые позволяют расширять его функциональность. Одной из таких библиотек выступила Material Design. Она предоставляет готовые компоненты и стили, позволяющие разработчикам быстро создавать визуально привлекательные пользовательские интерфейсы.

Клиентская часть должна быть не только правильно разработанной, помимо этого для этой части приложения необходимо обеспечить грамотный и приятный глазу дизайн. Данный выбор библиотеки обеспечит качественное представление элементов приложения.

Redux

Redux — библиотека с простым API, предсказуемое хранилище состояния приложений. Она работает по тому же принципу, что и функция `reduce`, один из концептов функционального программирования. Данная библиотека позволяет более легко масштабировать приложения, избавляет от ошибок, связанных с беспорядком в объекте состояния, повышает производительности и работоспособности программы.

Обоснование выбора инструментов для программирования серверной части приложения

Python

Python — это высокоуровневый язык программирования, который часто используется для программирования веб-приложений на бэкенде. Многие разработчики выбирают Python для разработки бэкенд-части своих приложений по следующим причинам:

1. Простота использования и высокая скорость разработки. Python — это язык программирования с простым синтаксисом и легким для изучения. Он обладает богатыми библиотеками и инструментами, которые позволяют разработчикам разрабатывать более эффективный и продуктивный код, что сокращает время разработки и упрощает поддержку приложения.

2. Большое количество библиотек и фреймворков. В Python существует много отличных фреймворков и библиотек для разработки веб-приложений, таких как Flask, Django, Pyramid и Tornado. Эти фреймворки упрощают разработку и обеспечивают легкость в создании проекта.

3. Широкая поддержка сообщества. Python имеет большое сообщество разработчиков, которые создают новые библиотеки и расширения, а также предоставляют помощь другим разработчикам и содействуют в развитии языка и его инфраструктуры. Благодаря этому поддерживать и развивать приложение на Python можно легко и без проблем.

aiohttp

aiohttp — асинхронный HTTP-сервер для модуля asyncio. Это библиотека языка Python, которая нужна для выполнения клиентских запросов и создания веб-сервера с потоковой выдачей и веб-сокетами. Асинхронность нужна для выполнения нескольких операций, установления соединений, одновременно, без ожидания завершения предыдущих. Использование такого подхода увеличивает скорость работы веб-сервисов в несколько раз, причина низкой производительности которых, обычно, не проведение сложных вычислений, а именно ожидание ввода/вывода.

Flask

Flask — фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2. Относится к категории так называемых микрофреймворков — минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности.

uWSGI

uWSGI — это быстрый скомпилированный серверный пакет с обширной конфигурацией и возможностями, выходящим за рамки базового сервера. Он очень производителен, поскольку является компилируемым. Широкий спектр функциональных возможностей в сочетании с относительной легкостью настройки делают uWSGI отличным вариантом для развертывания многих приложений, особенно в сочетании с Nginx.

SQLAlchemy

SQLAlchemy — это библиотека с открытым исходным кодом для работы с базами данных при помощи языка SQL. Она реализует технологию программирования ORM (Object-Relational Mapping), которая связывает базы данных с концепциями объектно-ориентированных языков программирования. SQLAlchemy позволяет описывать структуры баз данных и способы взаимодействия с ними прямо на языке Python. SQLAlchemy реализована в виде пакета для Python под лицензией MIT, а значит возможно ее использование в проприетарном ПО.

SQLAlchemy была выпущена в феврале 2006 и быстро стала одним из самых распространенных инструментов ORM среди разработчиков на Python. SQLAlchemy обладает несколькими областями применения, которые могут использоваться как вместе, так и по отдельности.

Обоснование выбора СУБД

PostgreSQL

PostgreSQL — это мощная и надежная объектно-реляционная система управления базами данных. Она является открытым исходным кодом и доступна для использования на многих ОС и архитектурах. PostgreSQL предлагает множество функций, широкий спектр поддерживаемых типов данных, а также возможность работы с многими клиентами и библиотеками.

Вот некоторые из основных возможностей PostgreSQL:

1. **Расширяемость.** PostgreSQL может легко быть расширен за счет создания новых типов данных, функций и хранимых процедур. PostgreSQL позволяет разработчикам создавать свои собственные типы данных, определять свои собственные функции и процедуры, что делает его более гибким и удобным для разных приложений.

2. **Транзакции.** PostgreSQL поддерживает транзакции и соответствующую блокировку данных. Он также поддерживает ACID-комплектность, что делает его надежной и безопасной платформой для хранения данных.

3. **Масштабируемость.** PostgreSQL можно использовать в крупных проектах, где требуется обработка больших объемов данных. Он поддерживает многоядерную архитектуру, а его производительность улучшается при использовании RAID-массивов или SSD-накопителей.

4. **Поддержка для сложных запросов и аналитики данных.** PostgreSQL также поддерживает сложные запросы, подзапросы, оконные функции и агрегатные функции. Это делает его подходящим для целого ряда задач, связанных с аналитикой данных.

5. **Открытый исходный код.** PostgreSQL является проектом с открытым исходным кодом и имеет активное сообщество, которое постоянно работает над его улучшением и совершенствованием. Большое количество дополнительных пакетов и модулей доступно для использования, что позволяет настроить PostgreSQL под свои нужды.

В целом, PostgreSQL является надежной, расширяемой и производительной системой управления базами данных с множеством функций и широким сообществом. Он подходит для различных типов приложений и может быть использован как в малых, так и в крупных проектах.

Форма взаимодействия между клиентом и сервером, компонентами системы

В качестве взаимодействия сервера и клиент была выбрана архитектура REST, так что сервер будет представлять собой REST API,

обращающийся к базе данных за информацией и отправляющий ее клиенту по HTTP протоколу. Система считается спроектированной по REST, если:

1. Система имеет явное разделение на клиент и сервер;
2. Сервер хранит какой-либо информации о клиентах. В запросе должна храниться вся необходимая информация для обработки запроса;
3. Используется кеширование данных;
4. Используется единый интерфейс между клиентом и сервером;
5. Система разделена на несколько малосвязанных между собой слоев.

Там, где необходимо непрерывное обновление данных для пользователей (например, лента последних постов), используется библиотека Flask-SocketIO для микрофреймворка Flask, работающая на основе WebSockets и позволяющая в режиме реального времени оповещать пользователей об изменениях данных на сервере.

Приложение будет разделено на микросервисы, взаимодействие которых будет организовываться внутри общей сети, разворачиваемой с помощью docker-compose. Точкой входа извне будет являться веб-сервер Nginx, который также будет находиться в сети docker-compose и будет проксировать запросы непосредственно к модулям приложения.

Выводы

Таким образом, исходя из требований к заявленному приложению был проведен анализ нескольких инструментов для разработки, что послужило к формированию стека следующих технологий:

- для разработки клиентской части приложения были выбраны библиотеки на базе языка программирования высокого уровня — JavaScript: React.js, Redux. Заявленные инструменты обеспечат быстрый и удобный интерфейс для пользователей приложения;
- для разработки серверной части приложения были выбраны следующие инструменты: язык программирования Python, СУБД PostgreSQL и

SQLAlchemy для переноса моделей на код, Flask и aiohttp для разработки серверной части (в зависимости от требований к скорости работы модуля), uWSGI – в качестве самого веб-сервера;

- перед клиентской и серверной частью будет стоять веб-сервер Nginx, а всё приложение будет контейнеризировано и запущено с использованием Docker.

2. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1. Разработка структуры системы

В современной практике обновления приложений широко применяется переход к приложениям, основанным на микросервисной архитектуре. Это включает в себя использование контейнерных технологий, таких как Docker и Kubernetes. Этот подход обеспечивает возможность более гибкого масштабирования, ускорения разработки и сокращения итераций при создании сервисов.

Микросервисная архитектура предполагает разбиение приложения на независимые сервисы, каждый из которых может быть развернут и масштабирован по отдельности через API-интерфейсы. Это позволяет командам быстрее и эффективнее создавать сложные приложения. В отличие от монолитных приложений, подход на основе микросервисов способствует более оперативному внедрению новых функций и изменений без переписывания больших фрагментов кода.

Важные особенности микросервисной архитектуры включают:

- Независимые компоненты-сервисы, обеспечивающие гибкость разработки и развертывания приложения.
- Легкость обслуживания и тестирования, позволяющая экспериментировать с новыми возможностями и быстро возвращаться к предыдущим версиям при необходимости.
- Использование автоматизированных методов инфраструктуры для независимого развертывания и обновления сервисов без прерывания работы других команд или предыдущих версий сервисов.

Общий вид архитектуры представлен на рисунке 2.1:

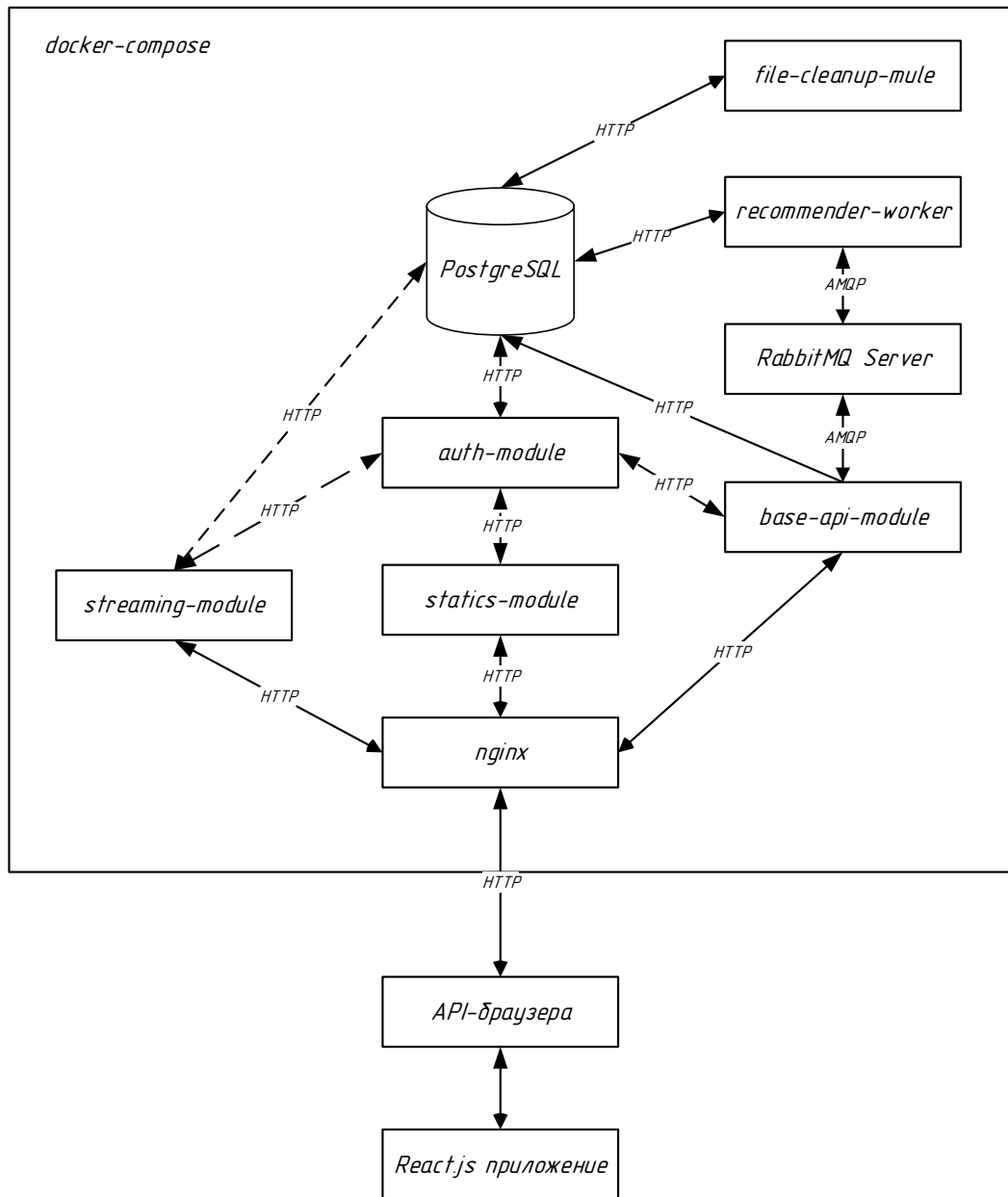


Рисунок 2.1. Структура полученной системы

Система, полученная в результате, представляет собой сложную структуру, включающую следующие ключевые компоненты:

- Приложение React.js, являющееся точкой входа для пользователей.
- Контейнеризированное приложение, состоящее из нескольких модулей:

- Веб-сервера Nginx, который выполняет роль точки входа для всех модулей и направляет запросы в зависимости от адреса.
- Модуль `statics-module` на `aiohttp`, отвечающий за обслуживание статических файлов, включая файлы, необходимые для клиентской части React.js-приложения.
- `streaming-module` на Flask, занимающийся функциями стриминга и управлением доступом к аудиозаписям в файловой системе.
- `base-api-module` на Flask, обеспечивающий основной функционал REST-API для работы с базой данных и непосредственного взаимодействия с пользователем.
- `file-cleanup-mule`, выполняющий очистку временных и неиспользуемых файлов, загружаемых при создании трека или неиспользуемых в базе данных.
- Сервер RabbitMQ, используемый для хранения очереди задач для воркера рекомендательной системы.
- `recommender-worker`, который в фоновом режиме обрабатывает запросы от рекомендательной системы для генерации ответов.
- Модуль `auth-module` на `aiohttp`, обеспечивающий управление сессиями и идентификацию пользователей через внутреннее API для всех модулей системы.

Эта структура объединяет различные функциональные блоки, обеспечивая полноценное взаимодействие между ними для обслуживания пользователей и выполнения различных задач в системе.

Для разрабатываемых модулей было принято использовать трёхуровневую архитектуру, так как преимущества данной модели включают улучшенную масштабируемость, производительность и доступность. В ходе разработки проекта, техническое задание может меняться, что, соответственно, будет влиять на структурную схему приложения; 3-уровневая модель облегчает непрерывное развитие проекта по мере появления новых задач и идей.

Существующие приложения могут быть постоянно или временно сохранены и инкапсулированы в новый уровень, компонентом которого они становятся

Модули имеют типовую структуру, представленную на рисунке 2.2.

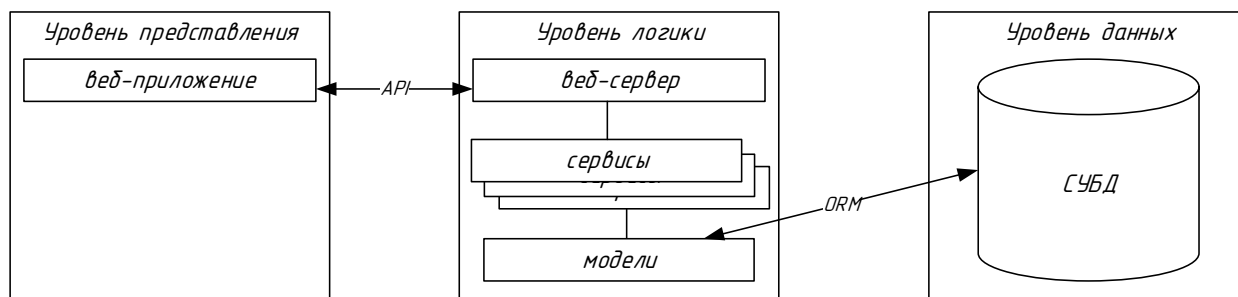


Рисунок 2.2 – Типовая структура модуля

Веб-приложение представляет собой ту часть приложения клиента на React.js, которая непосредственно выполняет запросы к модулю.

Основой модуля является веб-сервер – uWSGI с воркерами Flask или aiohttp. Этот сервер определяет маршруты запросов и передает их соответствующим обработчикам. Обработчики действуют как тонкие клиенты и выполняют методы соответствующих сервисов. Сервисы занимаются всей необходимой обработкой, оперируя моделями, которые абстрагируют строки таблиц базы данных. Важно отметить, что для создания единых точек создания объектов применяется принцип инъекции зависимостей. Это означает, что все объекты, использующие конфигурационные файлы, создаются только инъекторами, обеспечивая единообразие в процессе создания объектов и их зависимостей внутри модуля.

2.2. Структура базы данных

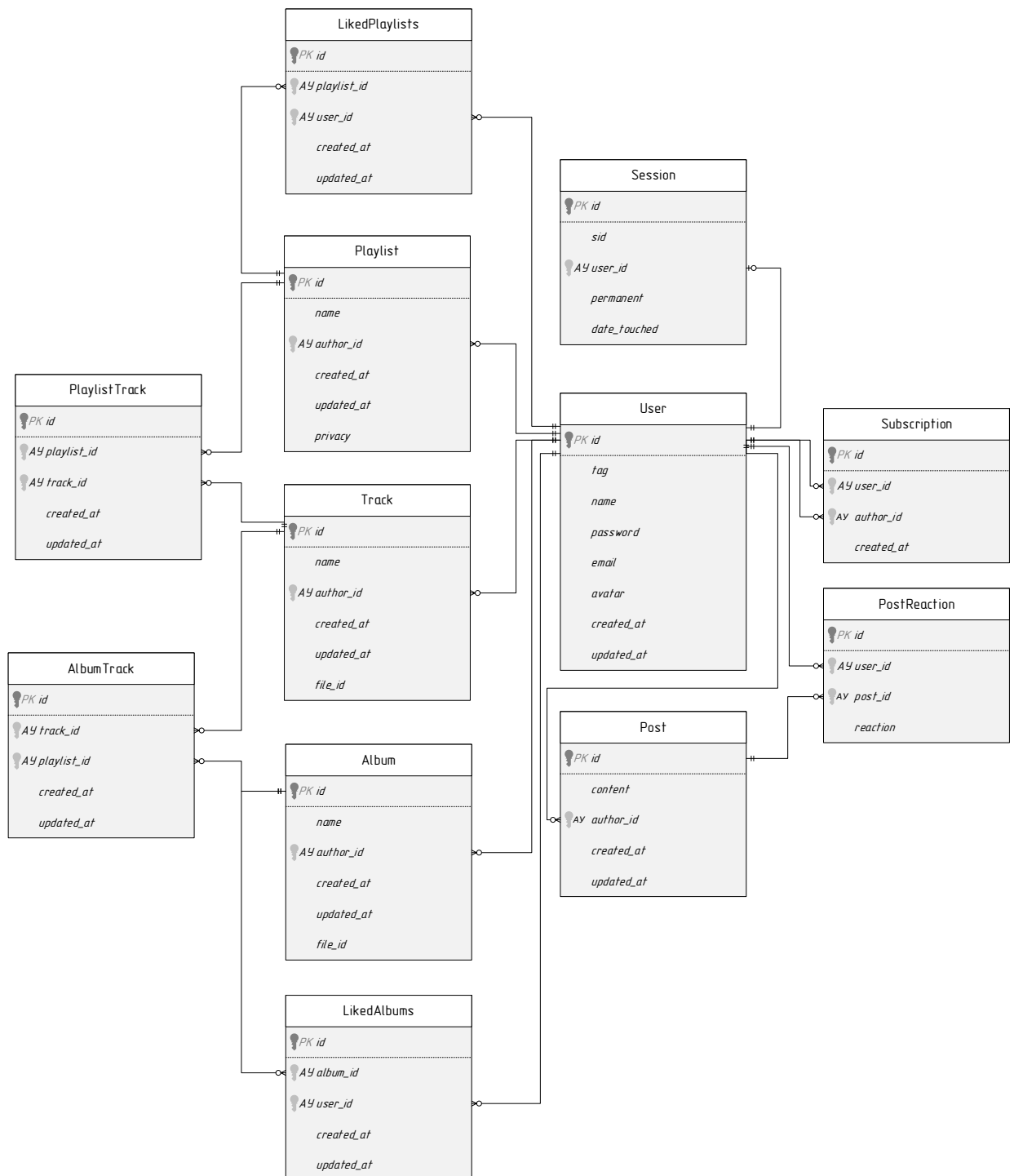


Рисунок 2.3. Структура базы данных

База данных состоит из следующих таблиц (см. рис. 2.3):

- **users** – хранит данные пользователей, используемые для авторизации и управления отображением пользователя;

- tracks – хранит данные о музыкальных композициях, а также идентификатор файла для поиска в файловой системе;
- playlists – хранит данные о плейлистах;
- albums – хранит информацию об альбомах;
- playlist_tracks – таблица реализации связи «многое-ко-многим» между таблицами playlists и tracks;
- album_tracks – таблица реализации связи «многое-ко-многим» между таблицами albums и tracks;
- posts – таблица постов на стене пользователя;
- subscriptions – таблица подписок пользователя на других пользователей;
- post_reactions – таблица реакций на посты;
- sessions – сессий пользователей;
- liked_playlists – понравившиеся пользователю плейлисты;
- liked_albums – понравившиеся пользователю альбомы.

В базе данных также был создан тип данных:

- playlist_privacy – представляет собой тип доступа к плейлисту:
 - subscribers;
 - user;
 - all.

Таблицы базы данных имеют следующие столбцы (см. табл. 1):

Таблица 1

Описание таблиц базы данных

Таблица	Название	Тип	NN	U	PK	FK	Default
users	id	integer	+	+	+		
	Идентификатор						
	tag	varchar (12)	+	+			
	Тег пользователя, используемый для его идентификации						
	password	text	+				
	Пароль, используемый пользователем при авторизации						
	avatar	text					
	Ссылка на пользовательский аватар						

	name	text	+				
	Имя пользователя						
	email	text	+				
	Электронная почта пользователя						
	created_at	time stamp	+				now()
	Время создания аккаунта						
	updated_at	time stamp					
	Время обновления аккаунта						
tracks	id	integer	+	+	+		
	Идентификатор						
	author_id	integer	+			users.id	
	Пользователь, который опубликовал трек						
	name	text	+				
	Название трека						
	created_at	time stamp	+				now()
	Время создания						
	updated_at	time stamp					
	Время обновления						
	file_id	text	+				
	Идентификатор файла внутри файловой системы						
playlists	id	integer	+	+	+		
	Идентификатор						
	name	text	+				
	Название альбома						
	privacy	playlist_privacy	+				subscribers
	Состояние приватности						
	author_id	integer	+			users.id	
	Создатель альбома						
	created_at	time stamp	+				now()
	Время создания						
	updated_at	time stamp					
	Время обновления						
playlist_tracks	id	integer	+	+	+		
	Идентификатор						
	playlist_id	integer	+			playlists.id	
	Идентификатор плейлиста						
	tracks_id	integer	+			tracks.id	

	Идентификатор трека						
	created_at	time stamp	+				now()
	Время создания						
	updated_at	time stamp					
	Время обновления						
album_tracks	id	integer	+	+	+		
	Идентификатор						
	album_id	integer	+			album.id	
	Идентификатор альбома						
	tracks_id	integer	+			tracks.id	
	Идентификатор трека						
	created_at	time stamp	+				now()
	Время создания						
	updated_at	time stamp					
	Время обновления						
post	id	integer	+	+	+		
	Идентификатор						
	author_id	integer	+			users.id	
	Автор поста						
	content	text	+				
	Текст поста						
	created_at	time stamp	+				now()
	Время создания						
	updated_at	time stamp					
	Время обновления						
post_reactions	id	integer	+	+	+		
	Идентификатор						
	user_id	integer	+			users.id	
	Идентификатор автора реакции						
	post_id	integer	+			posts.id	
	Идентификатор поста						
	reaction	text	+				
	Реакция на пост						
subscriptions	id	integer	+	+	+		
	Идентификатор						
	user_id	integer	+			users.id	

	Идентификатор подписавшегося						
	author_id	integer	+			users.id	
	Идентификатор того, на кого подписался пользователь						
	created_at	time stamp	+				now()
	Время создания						
sessions	id	integer	+	+	+		
	Идентификатор						
	sid	text	+				
	Хэш идентификации сессии						
	pemanent	bool	+				false
	Флаг постоянства сессии						
	date_touched	time stamp	+				now()
liked_albums	Дата создания сессии						
	id	integer	+	+	+		
	Идентификатор						
	user_id	integer	+			users.id	
	Идентификатор пользователя, добавившего альбома в медиатеку						
	album_id	integer	+			almubs.id	
	Идентификатор альбома						
	created_at	time stamp	+				now()
liked_playlists	Время создания						
	id	integer	+	+	+		
	Идентификатор						
	user_id	integer	+			users.id	
	Идентификатор пользователя, добавившего плейлист в медиатеку						
	playlist_id	integer	+			playlists.id	
	Идентификатор плейлиста						
	created_at	time stamp	+				now()
	Время создания						

2.3. Описание организации диалога с пользователем

Весь заявленный функционал приложения будет доступен пользователю посредством интерфейса клиентской части разрабатываемого приложения. Диаграмма вариантов использования отражает работу пользователя с

социальной сетью. На диаграмме действий (см. рис. 2.4) изображаются основные сценарии использования web-приложения.



Рисунок 2.4. Диаграмма действий пользователя

Пользователь может выполнить следующие действия:

- Авторизация;
- Регистрация;
- Просмотр профиля исполнителя;
- Просмотр своего профиля;
- Поиск треков, плейлистов и альбомов;
- Работать со своей медиатекой;
- Работать с записями на главной странице;
- Управлять воспроизведением музыкального плеера, просматривать очередь и текст трека.

На диаграмме описаны не все сценарии взаимодействия из-за их большого числа. Все они доступны через побочные активности и диалоговые окна с интуитивно понятным интерфейсом.

Разработанная диаграмма позволяет создать интуитивно понятный интерфейс с учётом заявленного на начальных этапах функционала создаваемой системы социальной сети.

Выводы

Использованная при разработке микросервисная архитектура обладает множеством преимуществ, что позволяет расширять проект с минимальными затратами по времени, масштабировать в зависимости от наличия ресурсов и обновлять модули независимо друг от друга. Выход из строя одного модуля не повлечёт за собой падение всей системы. Полученная архитектура имеет удобное и безопасное взаимодействие со слоем данных со стороны модулей с использованием технологии ORM. Точкой входа в сеть приложения является веб-сервер Nginx, который с использованием встроенного прокси распределяет запросы в соответствии с поступающим на вход адресом и разграничивает доступ к определённым модулям.

Пользователь непосредственно получает от сервера статичное собранное приложение React.js, которое взаимодействует с системой. Интерфейс клиентского приложения создан в соответствии с диаграммой действий пользователя и является простым и интуитивно понятным.

3. КОНТРОЛЬ КАЧЕСТВА И ИНТЕГРАЦИЯ КОМПОНЕНТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1. Руководство администратора

Для запуска системы необходимо:

1. Наличие ОС Linux (Ubuntu, Arch, CentOS) или Windows с поддержкой WSL2.
2. Установить Docker, Docker Compose.
3. При отсутствии готовых образов модулей необходимо наличие установленной системы сборки make: перейти в директорию каждого из модулей и выполнить его сборку, введя команду `make build`.
4. При наличии собранных модулей в главной директории проекта запустить compose файл: `docker-compose up`. Он также загрузит удалённые образы для Nginx, RabbitMQ и PostgreSQL 15.

После выполнения указанных действий приложение должно запуститься.

Модули имеют возможности конфигурирования, для чего в главной директории системы можно перейти в папку `modules` и, выбрав соответствующий модуль, открыть его `config.yml`, после чего отредактировать необходимое значение. После реконфигурирования необходим перезапуск контейнера модуля командой `docker restart [имя контейнера]`.

Обновление модулей выполняется изменением версий образов и их пересборкой с дальнейшим редактированием `docker-compose.yml` с указанием новой версий для соответствующего образа.

3.2. Руководство пользователя

При входе на сайт пользователю сразу же будет предложено авторизоваться (см. рис. 3.1). Если пользователь не авторизуется в системе, то

дальнейшая работа с приложением окажется невозможной. В форме авторизации пользователь должен указать свои имя пользователя и пароль, указанные при регистрации.

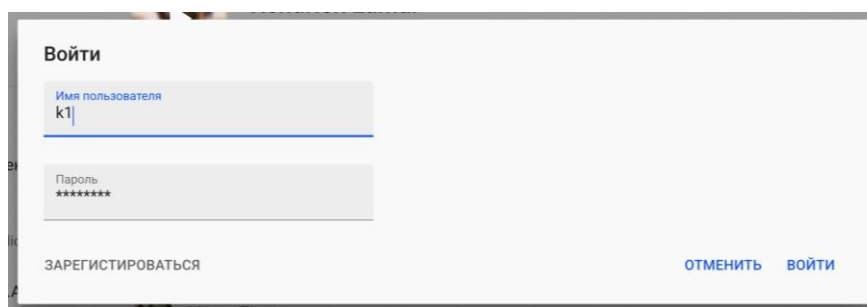
The screenshot shows a login window titled "Войти". It contains two input fields: "Имя пользователя" with the text "k1" and "Пароль" with masked characters "*****". Below the fields are two buttons: "ЗАРЕГИСТРИРОВАТЬСЯ" on the left and "ОТМЕНИТЬ" and "ВОЙТИ" on the right.

Рисунок 3.1. Окно авторизации

При отсутствии учётной записи, пользователь может нажать кнопку «Зарегистрироваться» и перейти к форме регистрации (см. рис. 3.2). В форме необходимо указать имя пользователя, пароль и адрес электронной почты.

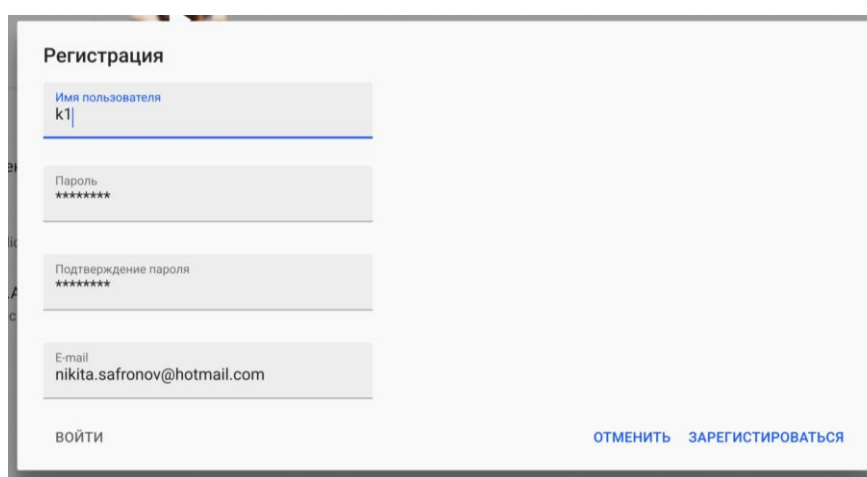
The screenshot shows a registration window titled "Регистрация". It contains four input fields: "Имя пользователя" with "k1", "Пароль" with "*****", "Подтверждение пароля" with "*****", and "E-mail" with "nikita.safronov@hotmail.com". At the bottom left is a "ВОЙТИ" button, and at the bottom right are "ОТМЕНИТЬ" and "ЗАРЕГИСТРИРОВАТЬСЯ" buttons.

Рисунок 3.2. Окно регистрации

После нажатия клавиши «Войти» или «Зарегистрироваться» в соответствующих окнах пользователю откроется доступ к главной странице (см. рис. 3.3).

Ключевым элементом этой страницы является лента записей от исполнителей, на которых подписан пользователь. Записи могут представлять собой как написанные пользователем посты, так и автоматически сгенерированные сообщения о том, что пользователь совершил какое-то действие (например, опубликовал трек, плейлист или альбом). На главной

странице пользователь может как создавать посты, так и оставлять реакции под постами других пользователей.

Другим элементом управления на этой странице является боковая панель, с помощью которой пользователь может переходить к добавленным в медиатеку плейлистам и альбомам, а также к автоматически создаваемому при регистрации плейлисту – «Любимые треки», в который попадают все треки, которым пользователь поставил отметку нравится.

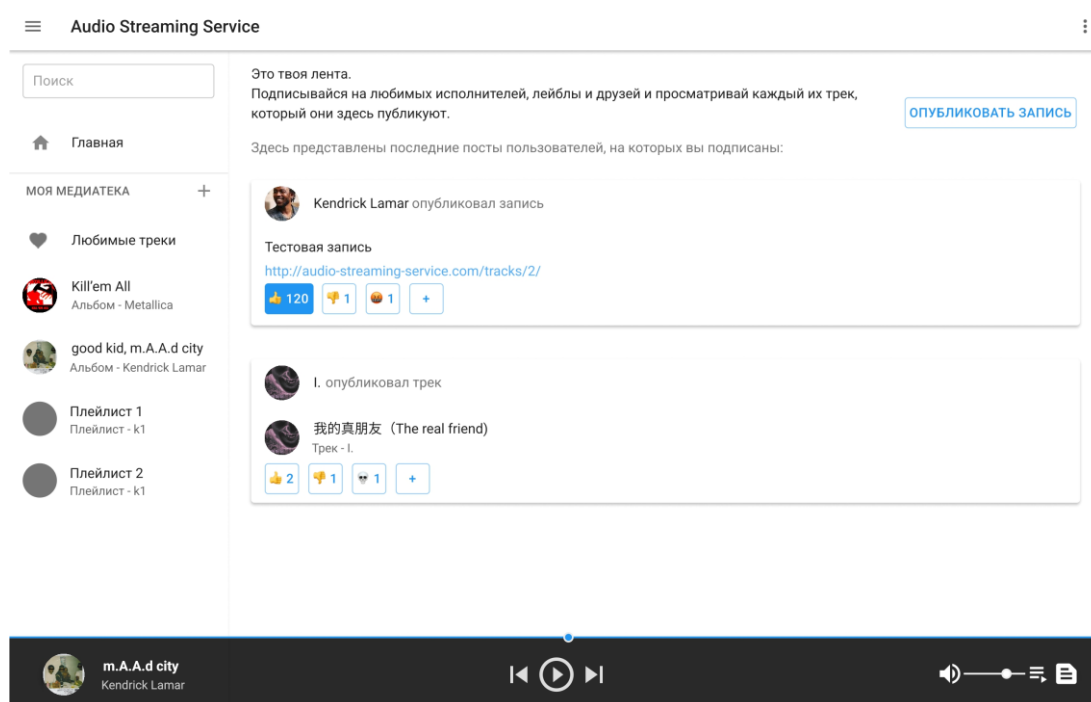


Рисунок 3.3. Главная страница

По нажатию на кнопку «Опубликовать запись» пользователю будет предложено ввести текст создаваемой записи и опубликовать её в соответствующем окне (см. рис. 3.4).

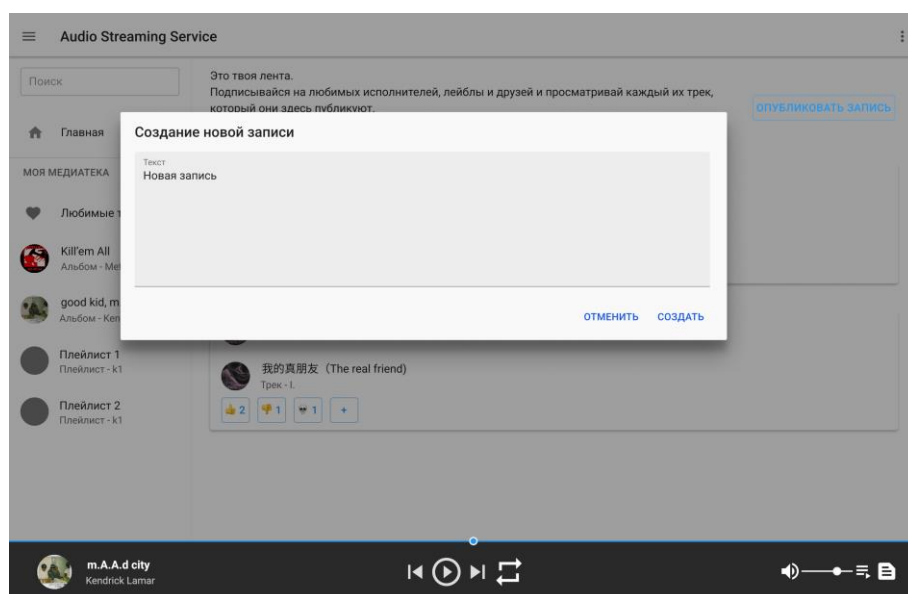


Рисунок 3.4. Окно «Опубликовать запись»

Пользователь может попасть к страницам альбомов и плейлистов либо из поиска, либо из своей медиатеки. Треки, включённые в плейлисты или альбомы, пользователь может добавить в свою медиатеку. Страницы альбома и плейлиста имеют типовой вид (см. рис. 3.5 и 3.6).

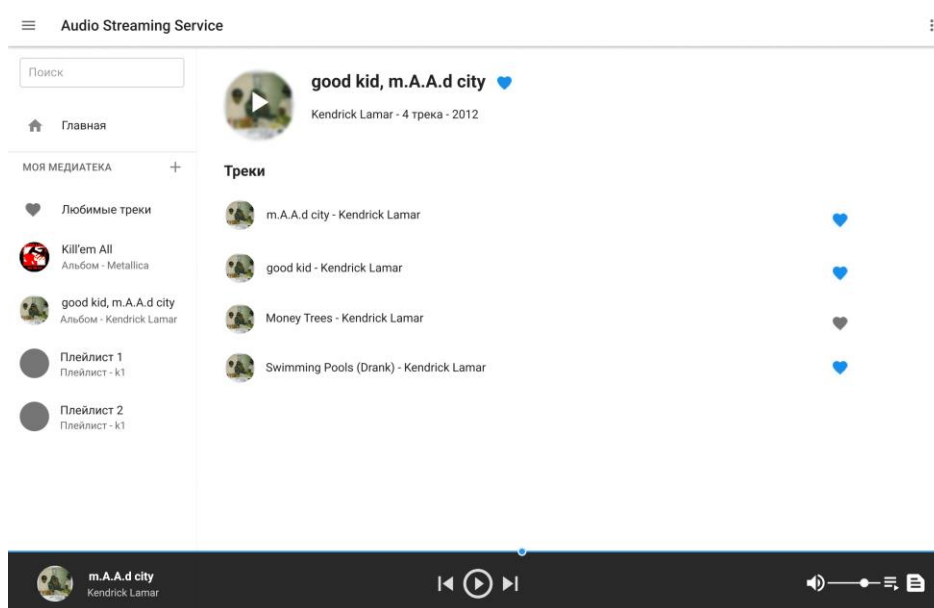


Рисунок 3.5. Страница альбома

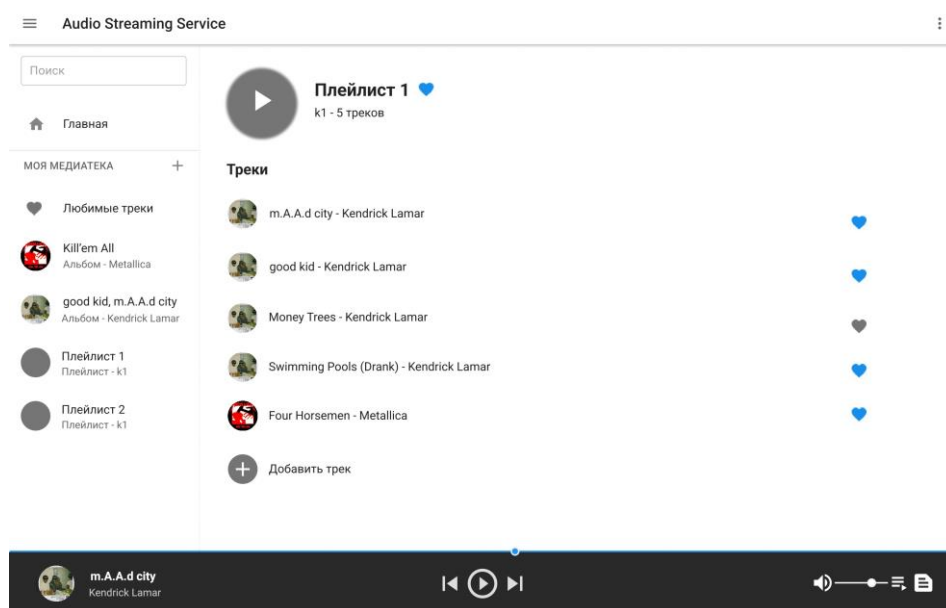


Рисунок 3.6. Страница плейлиста

Отличия этих страниц заключаются в возможности редактировать настройки приватности для плейлиста. Треки, как и сами плейлисты и альбомы, можно добавлять в свою медиатеку, нажав на соответствующую кнопку с иконкой сердца. Иконка также является индикатором того, добавлен ли объект в медиатеку или нет. По нажатию на обложку альбома, плейлиста или трека начнётся проигрывание. Для альбома и плейлиста очередь воспроизведения строится, начиная с первого трека, а для трека – с его позиции и до конца плейлиста. При завершении воспроизведения очереди воспроизведение начнётся с начала плейлиста или альбома.

В нижней панели представлена текущая композиция и исполнитель, а также кнопки для управления воспроизведением: «Предыдущий трек», «Пауза», «Следующий трек»; «ползунок» громкости, «Очередь» и «Текст композиции». В интерфейсе они представлены соответствующими иконками. На верху нижней панели находится «ползунок» управления положением воспроизведения.

По нажатию на кнопку «Очередь» пользователь сможет увидеть следующие композиции в очереди (см. рис. 3.7).

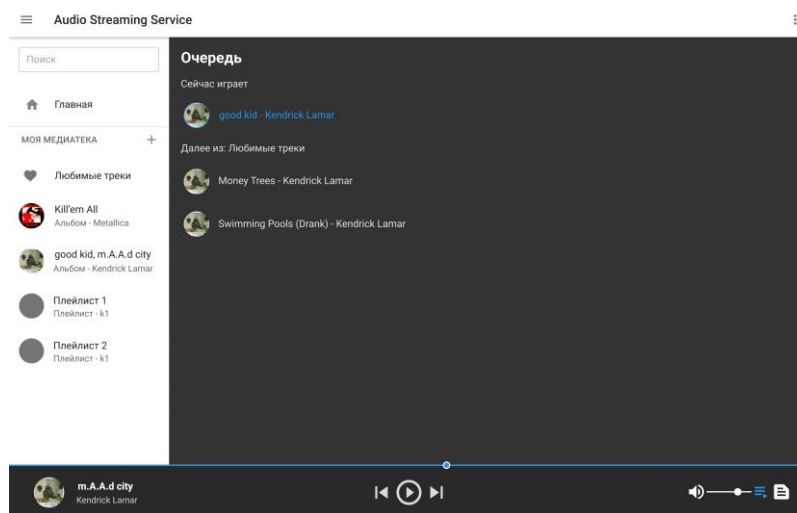


Рисунок 3.7. Очередь воспроизведения

По нажатию на кнопку «Текст трека» пользователь сможет увидеть текст для трека, если он был добавлен (см. рис. 3.8).

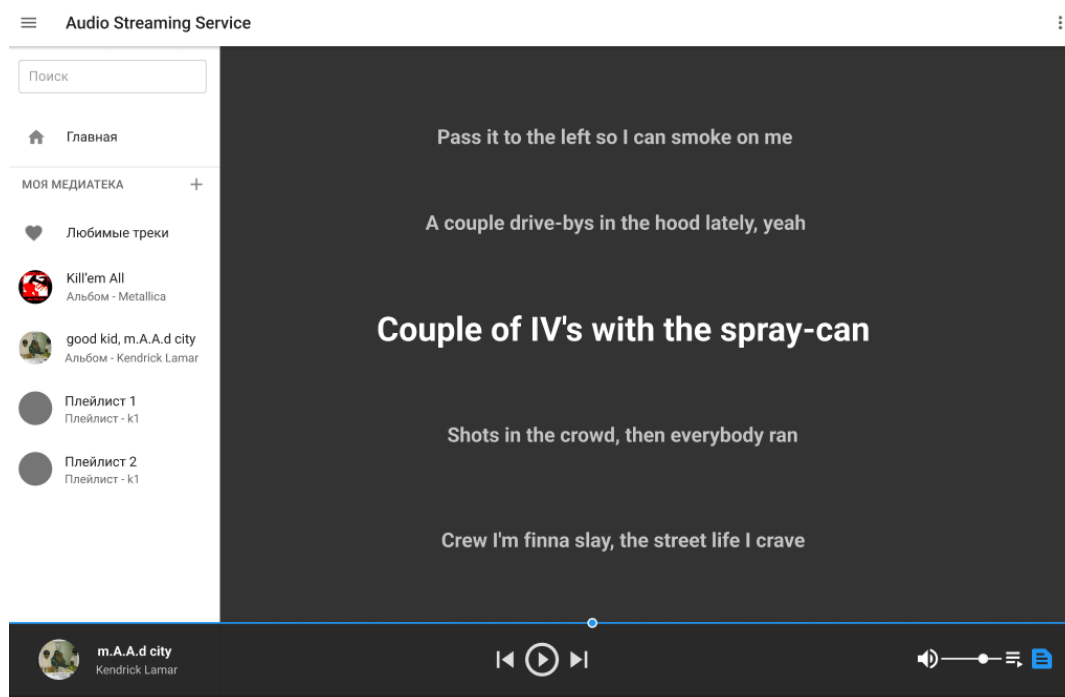


Рисунок 3.8. Просмотр текста трека

С помощью функции поиска пользователь может перейти на страницу конкретного исполнителя (см. рис. 3.9). На этой странице пользователь может выполнить несколько действий: подписаться на исполнителя, просмотреть список популярных треков, ознакомиться с полной дискографией или

посмотреть созданные им плейлисты. Когда пользователь нажимает кнопку «Подписаться», он автоматически подписывается на аккаунт исполнителя, что позволяет получать обновления и посты исполнителя в своей ленте.

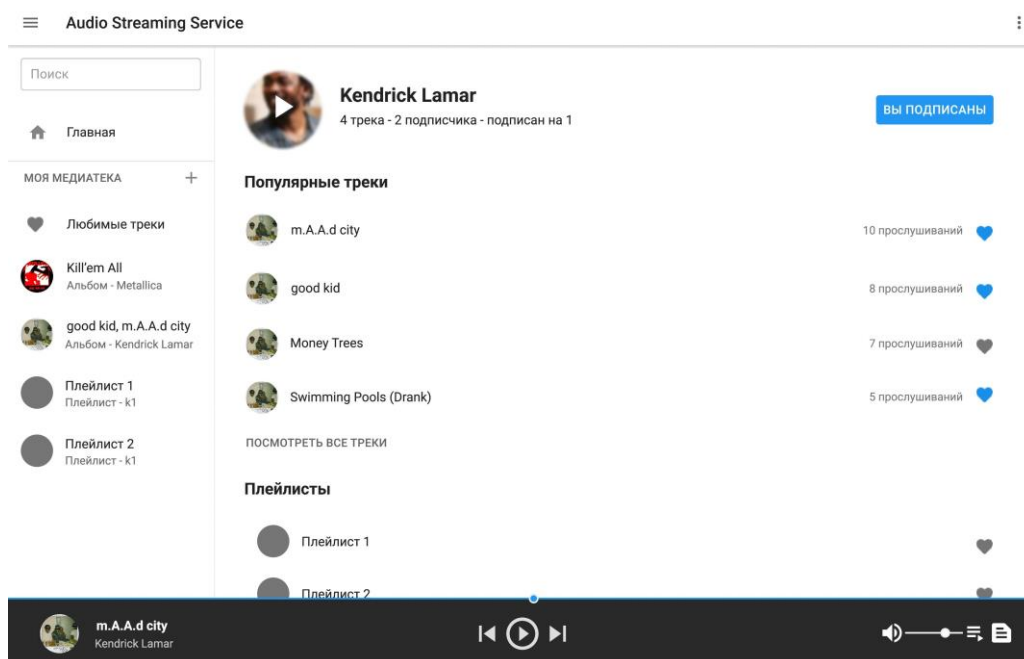


Рисунок 3.9. Страница исполнителя

По нажатию текст с информацией о подписчиках и о том, на кого подписан исполнитель, пользователю откроется соответствующее окно со списком подписок (см. рис. 3.10).

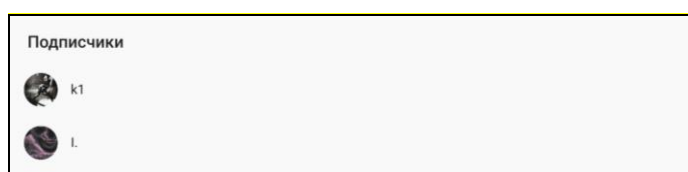


Рисунок 3.10. Просмотр списка подписчиков исполнителя

По нажатию на кнопку «Просмотреть все треки» (или подобные ей кнопки для плейлистов и альбомов) пользователь получает доступ к полному списку треков исполнителя и может добавить понравившиеся в свои избранные композиции (см. рис. 3.11).

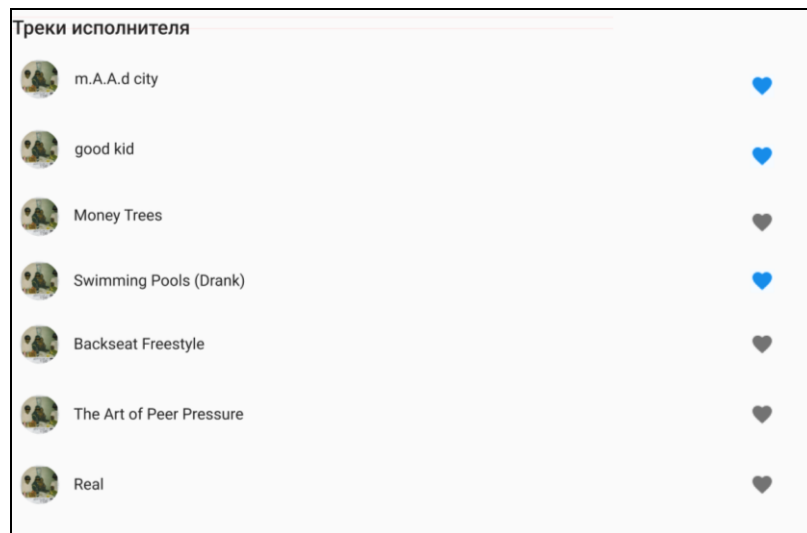


Рисунок 3.11. Просмотр списка треков исполнителя

3.3. Руководство исполнителя

Для выполнения действий от лица исполнителя достаточно перейти в свой профиль (см. рис. 3.12). Чтобы сделать это, нужно нажать на кнопку в правом верхнем углу и выбрать «Профиль» из выпадающего списка.

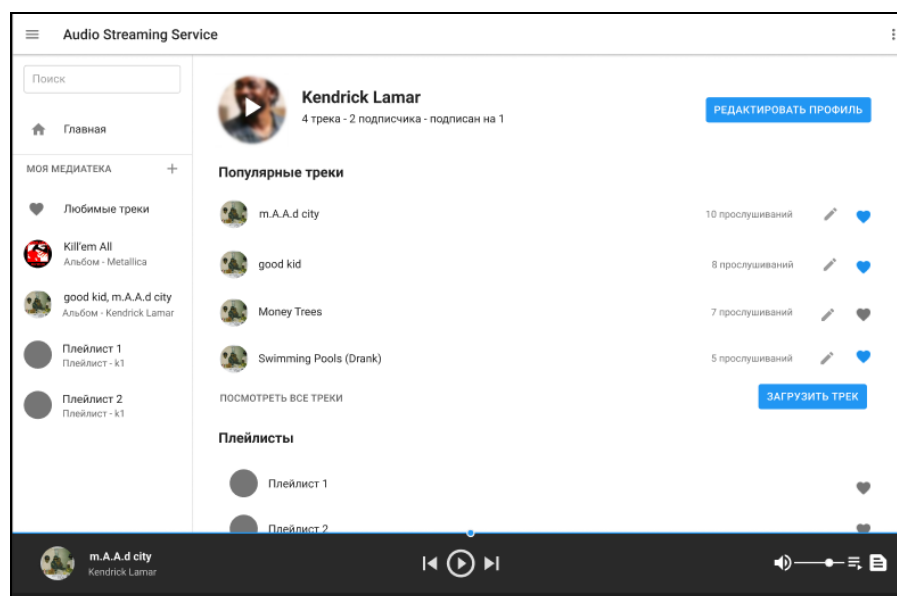


Рисунок 3.13. Страница исполнителя от его лица

Исполнитель может загрузить новый трек, нажав на соответствующую кнопку. В окне «Загрузка нового трека» (см. рис. 3.14) ему будет предложено выбрать название аудиозаписи и альбом, которому она будет принадлежать, а также загрузить непосредственно аудиофайл композиции. Аудиофайлы загружаются во временную директорию сервера и хранятся в ней либо один час, либо пока пользователь не создаст соответствующую запись для трека – тогда файл будет перемещён в постоянное хранилище.



Рисунок 3.14. Загрузка нового трека

Исполнитель может также создать альбом, нажав на соответствующую кнопку. Для создания альбома в соответствующем окне (см. рис. 3.15) необходимо указать название и загрузить обложку. Логика загрузки обложки точно такая же, как и у загрузки аудиофайла.

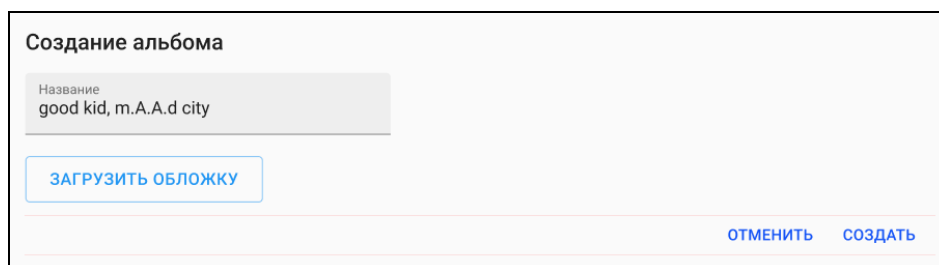


Рисунок 3.15. Создание альбома

Исполнитель может также создать плейлист, нажав на соответствующую кнопку. Для создания плейлиста в соответствующем окне (см. рис. 3.16)

необходимо указать название и выбрать его тип приватности (открытый, закрытый, для подписчиков).



Создание плейлиста

Название
good kid, m.A.A.d city


Открытый ▼

ОТМЕНИТЬ СОЗДАТЬ

Рисунок 3.16. Создание плейлиста

Непосредственно само добавление треков в плейлист или альбом, их редактирование происходит на страницах альбома и плейлиста.

При нажатии на кнопку «Редактирование профиля» пользователю откроется соответствующее окно (см. рис. 3.17). В окне пользователь может отредактировать свои публичные данные: имя пользователя и аватар.



Профиль

Имя
Kendrick Lamar

ЗАГРУЗИТЬ АВАТАР

ОТМЕНИТЬ СОХРАНИТЬ

Рисунок 3.17. Редактирование профиля пользователя

Окна редактирования для альбомов, треков и плейлистов подобны окнам создания, но также включают в себя кнопку «Удалить» выполняющую соответствующее действие после подтверждения пользователя.

Выводы

Первичная настройка и установка системы является довольно простой и выполняется посредством возможностей docker-compose, а также собранных или загруженных образов модулей.

В результате разработки клиентской части приложения был спроектирован удобный и интуитивно понятный пользовательский интерфейс, который позволяет максимально полно взаимодействовать с системой как рядовому пользователю, так и исполнителю.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта был спроектирована и разработана социальная сеть с функциями музыкального стриминга.

Результат соответствует всем заранее определенным требованиям технического задания.

Созданное приложение основано на следующем стеке технологий: для разработки клиентской части: React.js, Redux; для разработки серверной части - СУБД PostgreSQL и SQLAlchemy, Flask и aiohttp; веб-сервер Nginx; Docker для удобства развёртывания.

В работе были подробно описаны развёртывание и настройка приложения.

Разработанное приложение обладает простым и понятным пользовательским интерфейсом, что позволит использовать веб-приложение любому пользователю.

Благодаря удачно спроектированной архитектуре клиентского и серверного приложений на основе микросервисов, при необходимости можно улучшить и расширить разработанную систему путем добавления дополнительного функционала, к примеру:

- Модуль мессенджера;
- Нейросетевой модуль, работающий на очереди RabbitMQ и генерирующие тексты к трекам, для которых они не были добавлены;
- Модуль рекомендательной системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Основная литература

1. Сергеев, А.Н. Основы локальных компьютерных сетей [Электронный ресурс]: учебное пособие / А.Н. Сергеев. — Санкт-Петербург: Лань, 2016. — 184 с. — Режим доступа: URL: <https://e.lanbook.com/book/87591>
2. Топорков, С.С. Компьютерные сети для продвинутых пользователей [Электронный ресурс]: учебное пособие / С.С. Топорков. — Москва: ДМК Пресс, 2009. — 192 с. — Режим доступа: URL: <https://e.lanbook.com/book/1170>

Дополнительная литература

3. Арно Лоре, Проектирование веб-API / Пер. с англ. Д. А. Беликова. — М.: ДМК Пресс, 2020. — 440 с. ISBN 978-5-97060-861-6
4. Ачилов, Р.Н. Построение защищенных корпоративных сетей [Электронный ресурс]: учебное пособие / Р.Н. Ачилов. — Москва: ДМК Пресс, 2013. — 250 с. — Режим доступа: URL: <https://e.lanbook.com/book/66472>
5. Бизли, Д., Джонс Б.К., Python. Книга рецептов. — М.: ДМК Пресс, 2019. — 648 с.: ил.
6. Гамма, З., Хелм, Р., Джонсон, Р., Влиссидес, Дж. Приёмы объектноориентированного проектирования. Паттерны проектирования. — СПб.: Питер, 2016. — 368 с.: ил.
7. Гринберг, М. Разработка веб-приложений с использованием Flask на языке Python. — М.: ДМК Пресс, 2016.
8. Гутман, Г. Н. Объектно-реляционная СУБД PostgreSQL: учебное пособие / Г. Н. Гутман. — Самара: АСИ СамГТУ, 2016.
9. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. — СПб.: Питер, 2018. — 352 с.: ил. — (Серия «Библиотека программиста»). ISBN 978-5-4461-0772-8

10. Мартин Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста. — СПб.: Питер, 2013. — 464 с.
11. Моуэт Э. Использование Docker — Москва : ДМК Пресс, 2017 — 354 с.
12. Ибе, О. Компьютерные сети и службы удаленного доступа [Электронный ресурс]: справочник / О. Ибе. — Москва : ДМК Пресс, 2007. — 336 с. — Режим доступа: URL: <https://e.lanbook.com/book/1169>
13. Персиваль, Г. Паттерны разработки на Python. TDD, DDD и событийноориентированная архитектура. — СПб.: Питер, 2022. — 336 с.: ил.
14. Тиленс, Т.М. React в действии. — М.: ДМК Пресс, 2018.
15. Чиннатамби, К. Изучаем React. — М.: БОМБОРА, 2016.