

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук
Кафедра информационных систем и сетевых технологий

Курсовой проект

Разработка веб-приложения «RGAR Music»

Преподаватель_____ В.С. Тарасов

Обучающийся_____ ст. 3 курса оч. отд. *Р.С. Аббасов*

Обучающийся_____ ст. 3 курса оч. отд. *Г.М. Станкеев*

Обучающийся_____ ст. 3 курса оч. отд. *А.Г. Пармонов*

Обучающийся_____ ст. 3 курса оч. отд. *Р.А. Жуков*

Воронеж 2023

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	2
ВВЕДЕНИЕ	4
1 Постановка задачи	5
2 Анализ предметной области	6
2.1 Глоссарий	6
2.2 Анализ существующих решений	7
2.3 Сферы применения	13
3 Диаграммы	14
3.1 Диаграмма прецедентов (Use-case diagram)	14
3.2 Диаграмма активности авторизации	15
3.3 Диаграмма последовательности авторизации	16
3.4 Диаграмма состояний авторизации	16
3.5 Диаграмма активности регистрации	17
3.6 Диаграмма последовательности регистрации	18
3.7 Диаграмма состояний регистрации.....	18
3.8 Диаграмма активности поиска музыки.....	19
3.9 Диаграмма последовательности поиска музыки.....	19
3.10 Диаграмма состояний поиска музыки	20
3.11 Диаграмма активности пользователей.....	20
4 Реализация	21
4.1 Средства реализации	21
4.2 Реализация backend.....	23
4.3 Реализация frontend.....	25
4.4 Навигация по приложению	25
5 Тестирование	32
5.1 UI-тестирование	32
5.2 Дымовое тестирование	35
ЗАКЛЮЧЕНИЕ	37

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	38
--	----

ВВЕДЕНИЕ

В современном мире музыка стала неотъемлемой частью нашей жизни. Мы слушаем ее везде: в дороге, на работе, дома. Интернет и новые технологии открыли перед нами огромные возможности для того, чтобы наслаждаться любимыми треками где бы мы ни находились. Создание собственного музыкального веб-плеера - это один из способов реализовать свои идеи и предложить пользователю удобный способ прослушивания музыки онлайн.

Целью нашего проекта является создание интуитивно понятного и удобного в использовании музыкального плеера, который соответствовал бы потребностям пользователей.

Данный курсовой проект будет посвящен разработке веб-приложения RGAR Music, будет содержать подробное описание каждого этапа процесса разработки сервиса.

1 Постановка задачи

Цель курсовой работы: реализовать веб-приложение музыкального плеера с удобным, простым и минималистичным интерфейсом, который будет выполнять основные задачи:

1. Авторизация
2. Формирование площадки для прослушивания музыки
3. Предоставить возможность поиска среди библиотеки исполнителей, альбомов плейлистов и треков
4. Автоматизация подбора музыкальных рекомендаций для пользователей
5. Предоставить возможность создания собственных плейлистов
6. Предоставить пользователям возможность делиться созданными плейлистами
7. Добавление в избранное понравившегося контента

Для пользователей:

1. Регистрация пользователей.
2. Вход в свой аккаунт.
3. Редактирование профиля.
4. Создание личный плейлистов.
5. Поиск альбомов, исполнителей, плейлистов, пользователей, треков.
6. Фильтрация поиска
7. Play/pause трека
8. Перемотка треков
9. Переключение трека
10. Добавление трека в очередь
11. Удаление трека из очереди
12. Изменение порядка очереди
13. Случайный порядок воспроизведения
14. Возможность смены пароля
15. Добавление трека в плейлист.

16. Удаление треков из плейлиста
17. Изменение порядка треков в плейлисте
18. Добавление треков, плейлистов, исполнителей альбомов в избранное
19. Просмотр статистики
20. Изменение личных данных

Для администратора:

1. Приложение должно поддерживаться администраторами, ознакомленными с правилами работы с приложением. Они будут взаимодействовать с информацией о пользователях, треках, плейлистах, альбомах и исполнителя
2. Через систему администрирования администраторы должны иметь возможность просмотра и удаления пользователей, добавления, редактирования и удаления треков, плейлистов, альбомов и исполнителей

2 Анализ предметной области

2.1 Глоссарий

1. Лайк – условное выражение одобрения материалу.
2. Дизлайк – условное выражение неодобрения материалу.
3. Тег – метка для облегчения процесса поиска.
4. Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью.
5. Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.
6. Django — это высокоуровневый Python веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты.

7. PostgreSQL – объектно-реляционная система управления базами данных.
8. Трек - отдельная цифровая запись музыкального произведения
9. Плейлист - составленный пользователем список музыкальных произведений.
10. API (программный интерфейс приложения) - описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой.
11. REST - архитектурный стиль взаимодействия компонентов распределённого приложения в сети.
12. Бэкенд (backend) — программно-аппаратная часть сервиса. Бэкенд отвечает за осуществление функционирования внутренней части приложения.

2.2 Анализ существующих решений

Сервис Apple Music

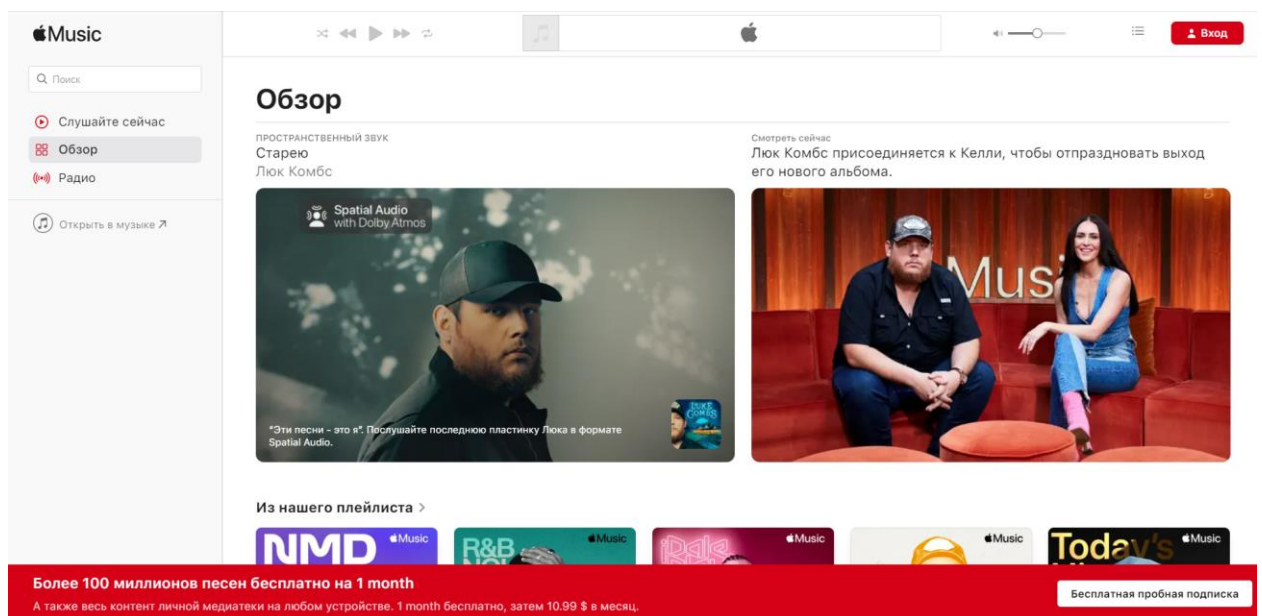


Рисунок 1 - Сервис Apple Music

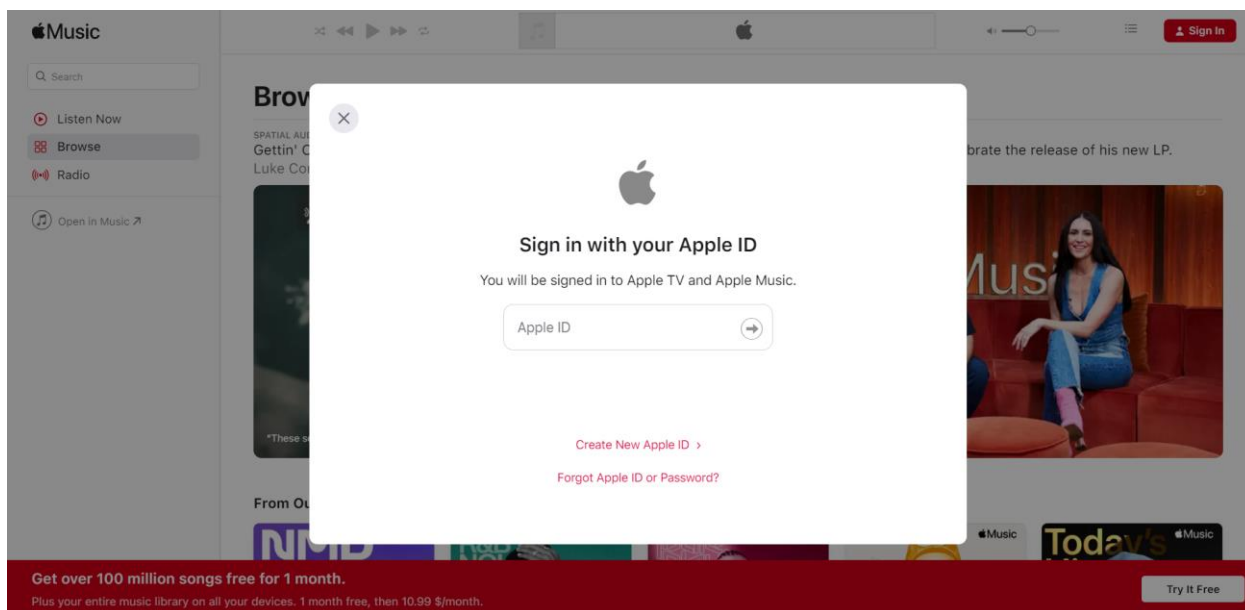


Рисунок 2 - Сервис Apple Music

Плюсы:

1. Большой выбор музыки: Apple Music предлагает доступ к более чем 75 миллионам песен, что позволяет пользователю найти почти любую музыку, которая ему нравится.
2. Высокое качество звука: Apple Music предоставляет доступ к музыке с битрейтом 256 кбит/с, что обеспечивает отличное качество звука.
3. Рекомендации и персонализация: Apple Music использует алгоритмы машинного обучения для предоставления рекомендаций пользователю на основе его предпочтений и прослушиваний.
4. Доступ к эксклюзивному контенту: сервис предлагает эксклюзивные релизы от знаменитых исполнителей, которые нельзя найти на других платформах.
5. Совместимость с другими продуктами Apple: Apple Music интегрируется с другими продуктами Apple, такими как Siri, HomePod, Apple Watch и т.д.

Минусы:

1. Цена: сервис стоит относительно дорого по сравнению с другими музыкальными сервисами.

2. Ограничения на использование: Apple Music не позволяет скачивать песни на устройства, которые не являются продуктами Apple, и ограничивает количество устройств, на которых можно использовать аккаунт.
3. Сложность навигации: для новых пользователей интерфейс Apple Music может показаться сложным и запутанным. Он имеет множество вкладок и меню, которые могут быть непривычными для пользователей, которые только начали пользоваться сервисом.
4. Отсутствие бесплатной версии: Apple Music не предоставляет бесплатную версию, и пользователи должны платить за использование сервиса, чтобы получить доступ к музыке.
5. Недоступность для некоторых стран: Apple Music может быть недоступен для некоторых пользователей, которые живут в странах, где сервис не работает.
6. Ограниченность настройки интерфейса: пользователи не могут настроить интерфейс Apple Music таким образом, как им бы хотелось. Например, нельзя изменить порядок воспроизведения песен в рамках одного плейлиста, а также нельзя изменить количество строк, отображаемых в библиотеке.
7. Ошибки в работе алгоритмов рекомендаций: иногда Apple Music может рекомендовать пользователю песни или артистов, которые не соответствуют его вкусам. Это может быть особенно раздражающим для пользователей, которые рассчитывают на индивидуальные рекомендации.

Сервис Spotify

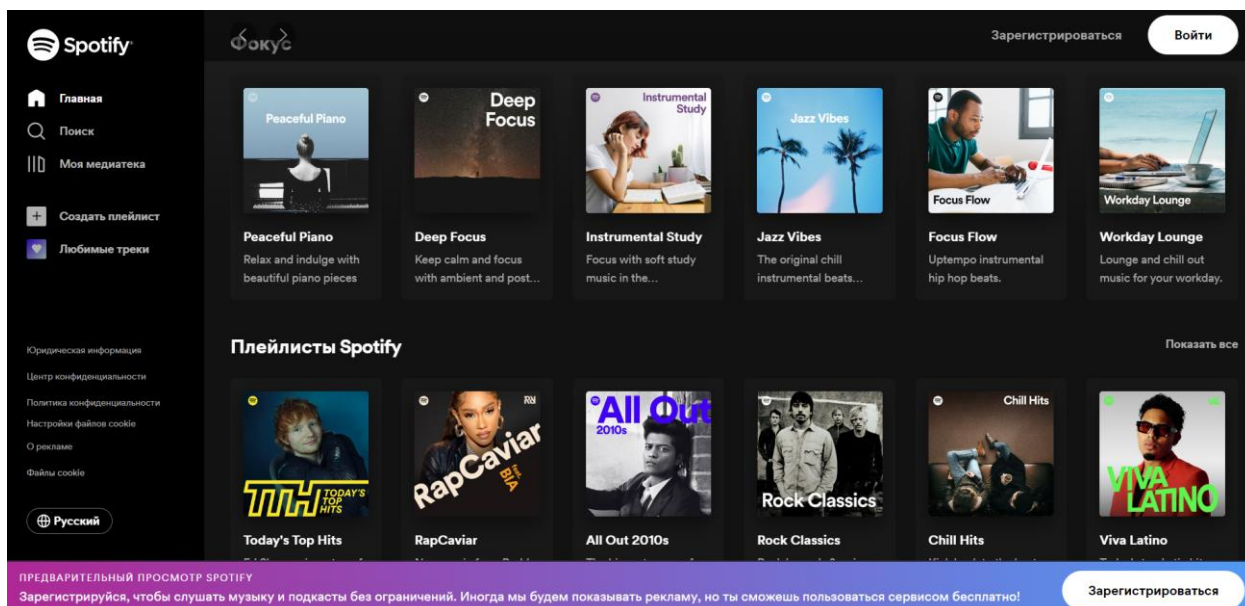


Рисунок 3 - Сервис Spotify

Плюсы:

1. Большой выбор музыки: Spotify предлагает доступ к более чем 70 миллионам песен, что позволяет пользователям находить почти любую музыку, которая им нравится.
2. Можно делиться музыкой с друзьями внутри сервиса
3. Персонализация: Spotify использует алгоритмы машинного обучения для предоставления персонализированных рекомендаций и плейлистов на основе прослушиваний и предпочтений пользователей.
4. Совместимость с другими устройствами: Spotify интегрируется со многими устройствами, включая смартфоны, смарт-часы и смарт-колонки, что делает его удобным для использования в различных ситуациях.
5. Режим оффлайн: пользователи могут скачивать музыку и использовать ее в режиме оффлайн, что позволяет им слушать музыку без доступа к Интернету.
6. Радио и подкасты: Spotify предоставляет доступ к радиостанциям и подкастам на различные темы.

Минусы:

1. Реклама: пользователи бесплатной версии Spotify должны слушать рекламу между песнями.
2. Ограничения на использование: Spotify ограничивает количество устройств, на которых можно использовать аккаунт, и количество песен, которые можно скачать на устройства.
3. Цена: некоторые пользователи могут считать цену подписки на Spotify высокой.
4. Качество звука: в бесплатной версии качество звука ограничено 160 кбит/с, что может не удовлетворять некоторых пользователей.
5. Сложности в поиске: пользователи могут столкнуться с трудностями в поиске песен или альбомов, особенно если они не знают названия или исполнителей. Некоторые пользователи также жалуются на то, что поиск не всегда дает точные результаты.
6. Случайные пропуски и повторы: некоторые пользователи жалуются на случайные пропуски песен или повторы, которые могут происходить при прослушивании музыки на Spotify.

Сервис MTS Music

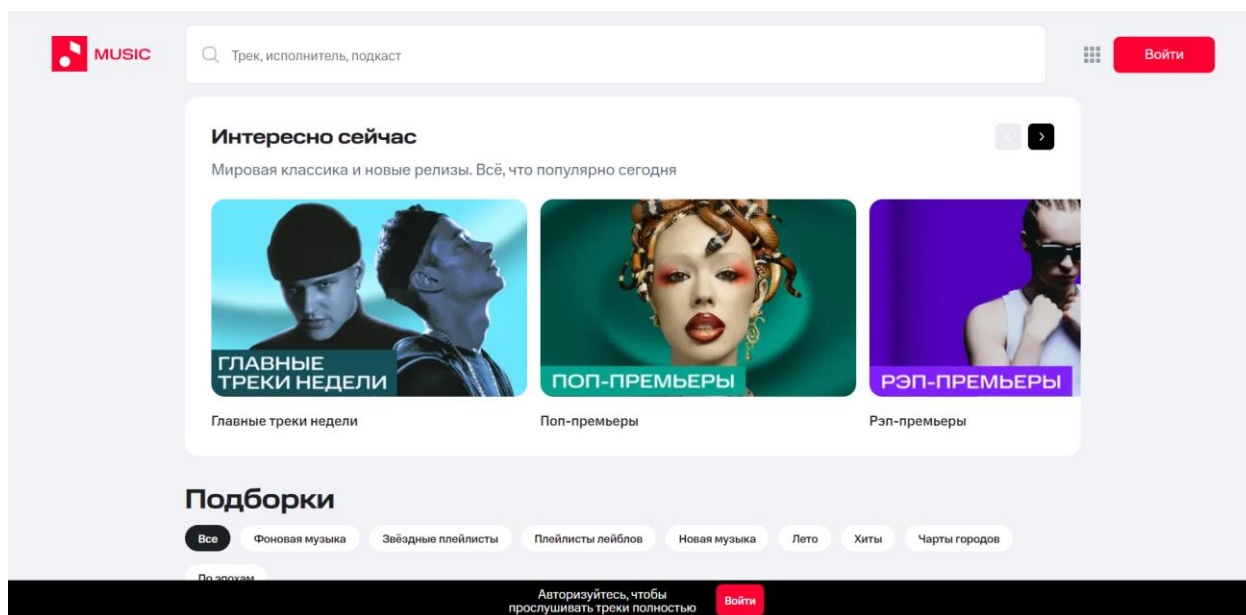


Рисунок 4 - Сервис МТС Музыка

Плюсы:

1. Бесплатная версия: сервис предоставляет бесплатную версию с ограниченными функциями, что позволяет пользователям ознакомиться с сервисом перед покупкой подписки.
2. Выбор плейлистов: MTS Music предоставляет большой выбор плейлистов на различные темы, созданных как экспертами сервиса, так и пользователями.
3. Отсутствие рекламы: сервис не содержит рекламу, даже в бесплатной версии.
4. Возможность прослушивания оффлайн: пользователи могут скачивать музыку и использовать ее в режиме оффлайн, что позволяет им слушать музыку без доступа к Интернету.

Минусы:

1. Ограниченный выбор музыки: в сравнении с другими музыкальными сервисами, такими как Spotify и Apple Music, библиотека MTS Music предлагает меньшее количество треков.
2. Неудобный поиск: поиск музыки на MTS music может быть немного неудобным и трудным для пользователей, особенно если они не знают название или исполнителя песни.
3. Ограниченные возможности в бесплатной версии: в бесплатной версии сервиса доступны ограниченные функции и количество песен, которые можно прослушать.
4. Ограниченность функций: MTS music имеет ограниченные функции, которые не всегда могут удовлетворять пользователей, которые хотят больше контроля над своими плейлистами и настройками воспроизведения.
5. Неэффективное управление плейлистами: пользователи могут столкнуться с трудностями при управлении своими плейлистами на MTS music, так как функции перемещения песен, изменения порядка воспроизведения и редактирования плейлистов не всегда интуитивны и не предоставляют достаточно опций.

6. Ограничения на использование: MTS Music ограничивает количество устройств, на которых можно использовать аккаунт, и количество песен, которые можно скачать на устройства.

2.3 Сферы применения

Музыкальный плеер имеет широкую сферу применения, которая в большинстве случаев является частью ежедневной активности людей. Разберем конкретно:

Персональный использование: Веб-приложения музыкальных плееров могут использоваться для личного использования пользователей, которые хотят воспроизводить музыку в браузере, без необходимости устанавливать дополнительные приложения на компьютер или мобильное устройство.

Музыкальные блоги и веб-сайты: Веб-приложения музыкальных плееров могут использоваться на музыкальных блогах и веб-сайтах для воспроизведения музыкальных треков и создания плейлистов, которые могут помочь привлечь и удержать читателей.

Музыкальные школы и колледжи: Веб-приложения музыкальных плееров могут использоваться в музыкальных школах и колледжах для обучения студентов музыке. Они могут использоваться для воспроизведения учебных материалов, плейлистов и других ресурсов, которые помогают студентам изучать и практиковать музыку.

Фитнес-индустрия: музыкальные плееры могут быть использованы в фитнес-клубах и тренировочных залах для создания музыкального настроения и поддержания мотивации участников занятий.

3 Диаграммы

3.1 Диаграмма прецедентов (Use-case diagram)

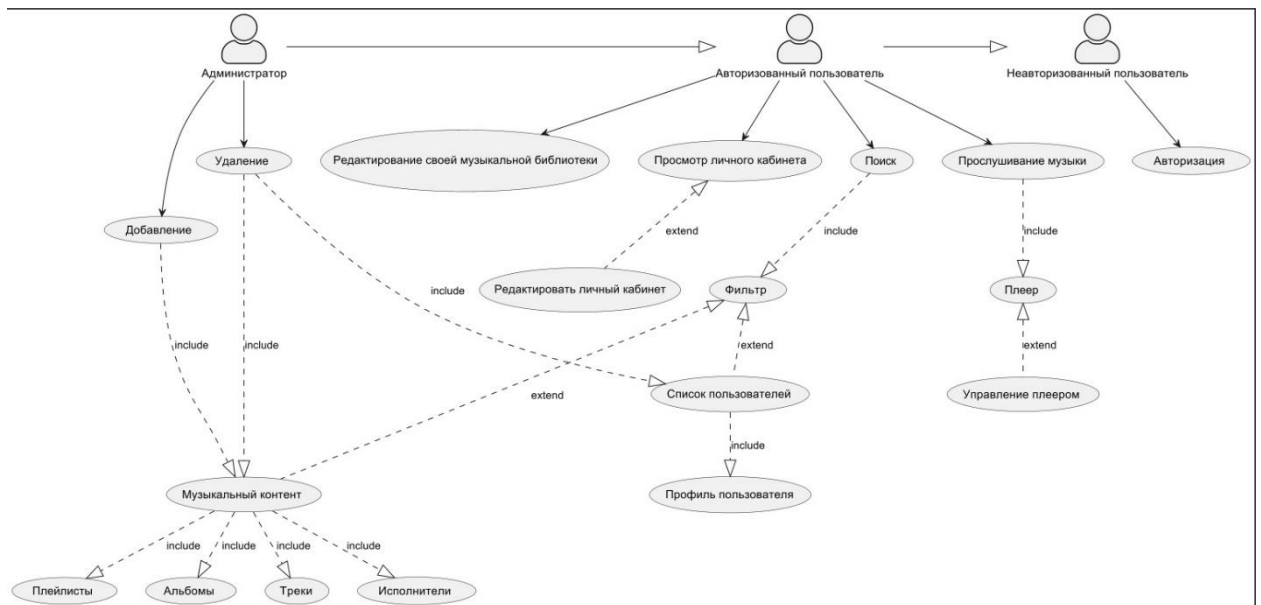


Рисунок 5 - Use-case diagram

Данная диаграмма отражает пользователей системы и их функции.

Неавторизованный пользователь – пользователь сайта, не зашедший в авторизованную зону системы и не имеющий полного доступа к функциям сайта.

Неавторизованный пользователь может:

- Может выполнить действие "Авторизоваться", чтобы войти в систему.

Авторизованный пользователь – пользователь сайт, зарегистрированный или авторизованный в системе и имеющий полный доступ к функциям сайта. Данный пользователь может:

- Просматривать личный кабинет
- Редактировать свой личный кабинет
- Выполнять поиск музыкального контента
- Слушать музыку, что включает управление плеером внутри процесса воспроизведения.

- Управлять своей музыкальной библиотекой (редактирование своей музыкальной библиотеки).
- Просматривать музыкальный контент, такой как плейлисты, альбомы, треки и исполнителей.

Помимо неавторизованного и авторизованного пользователя есть администратор. Он может:

- Выполнять все действия, доступные авторизованным пользователям
- Выполнять дополнительные действия, связанные с управлением музыкальным контентом, включая добавление (Добавление - и удаление музыкального контента
- Выполнять поиск музыкального контента и слушать музыку
- Управлять плеером, если это необходимо
- Просматривать профили других пользователей
- Просматривать список пользователей

3.2 Диаграмма активности авторизации

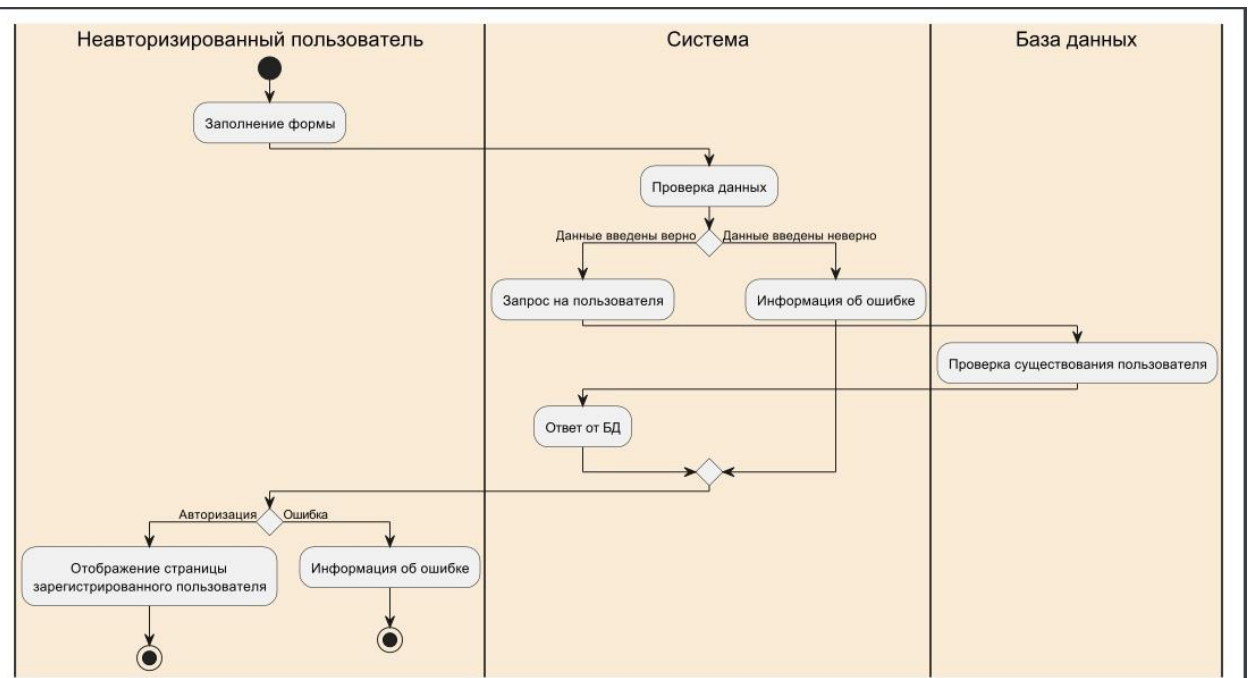


Рисунок 6 - Диаграмма активности авторизации

Диаграмма описывает процесс авторизации неавторизованного пользователя в системе. Пользователь вводит данные, система проверяет их в базе данных. Если данные верны, пользователь авторизуется и видит страницу пользователя; в противном случае, система сообщает об ошибке.

3.3 Диаграмма последовательности авторизации

Данная диаграмма представляет процесс авторизации пользователя в системе. Вначале пользователь вводит данные авторизации, которые затем проверяются в системе. Если данные некорректны, система выдает ошибку. При корректных данных, система отправляет запрос на проверку в базу данных, и, если пользователь найден, происходит авторизация. В противном случае система также сообщает, что пользователь не найден.

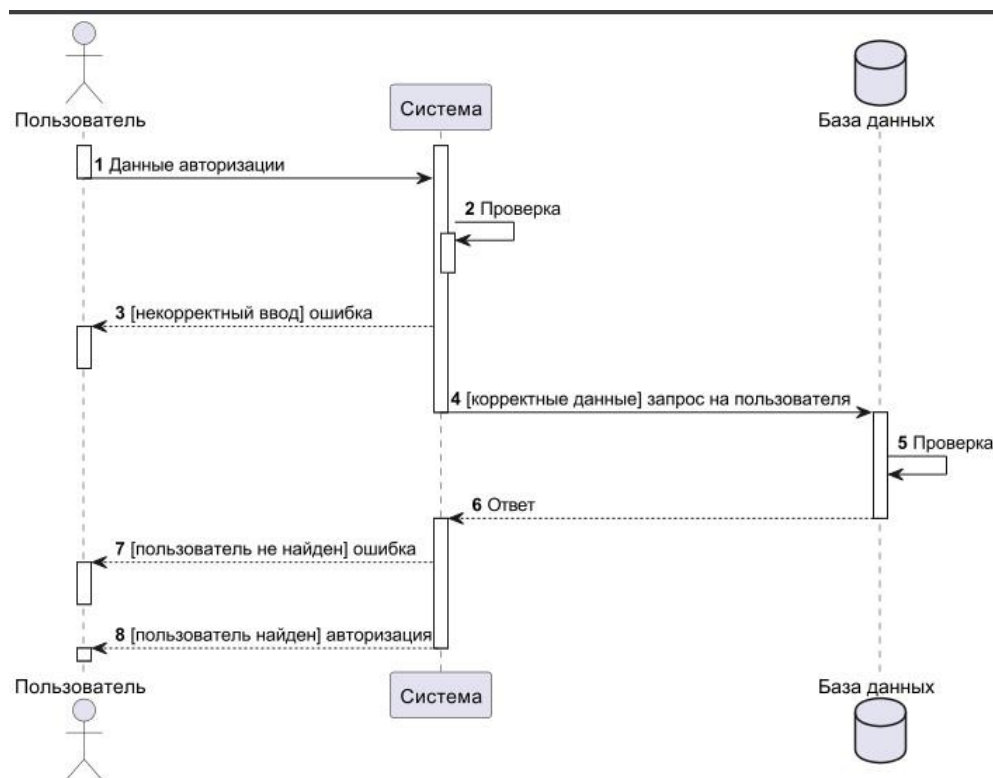


Рисунок 7 - Диаграмма последовательности авторизации

3.4 Диаграмма состояний авторизации

Данная диаграмма представляет процесс авторизации пользователя в системе. Пользователь начинает с запроса на авторизацию и вводит данные. Данные проверяются, и если они верны, происходит авторизация, и пользователь переходит на страницу зарегистрированного пользователя. В

случае неверных данных, система сообщает об ошибке и возвращает пользователя на начальный этап авторизации.



Рисунок 8 - Диаграмма состояний авторизации

3.5 Диаграмма активности регистрации

Диаграмма описывает процесс регистрации нового пользователя в системе. Пользователь вводит данные, система проверяет их уникальность в базе данных. Если данные уникальны, пользователь успешно регистрируется и видит страницу зарегистрированного пользователя; в противном случае система сообщает об ошибке.

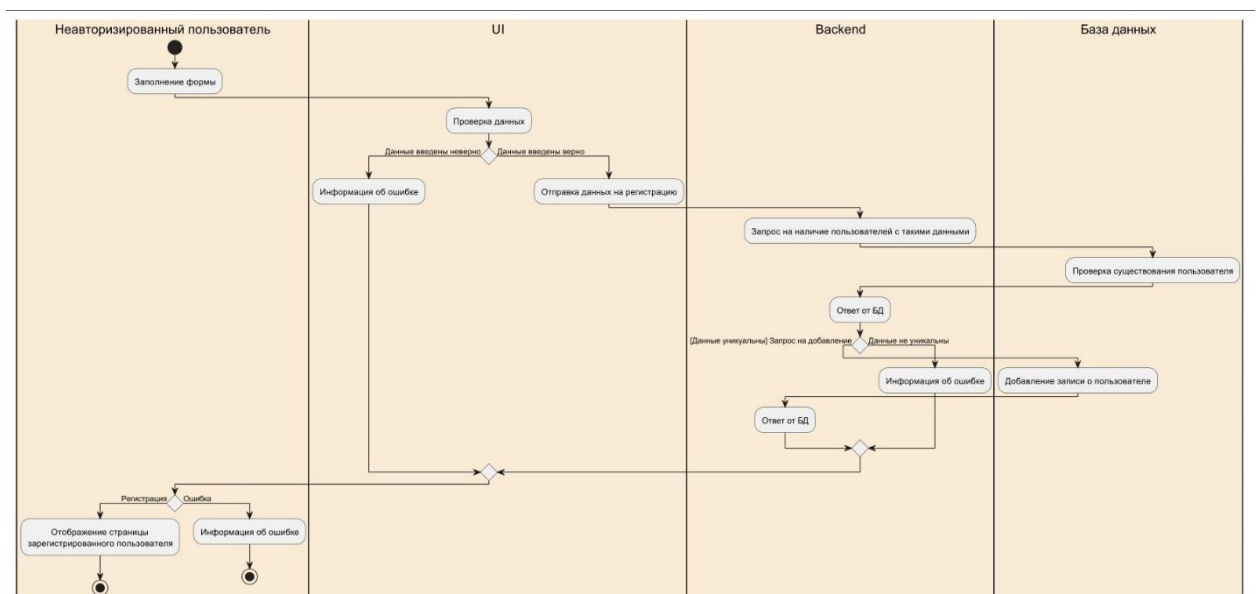


Рисунок 9 - Диаграмма активности регистрации

3.6 Диаграмма последовательности регистрации

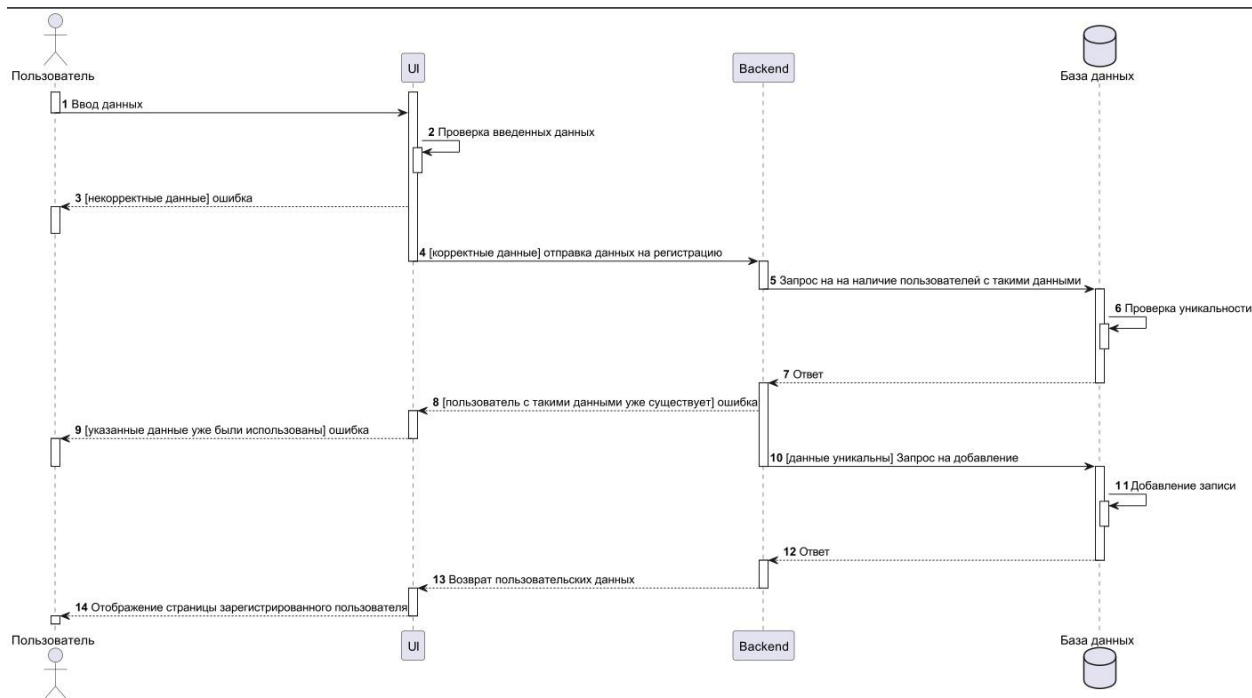


Рисунок 10 - Диаграмма последовательности регистрации

Данная диаграмма иллюстрирует процесс регистрации пользователя в системе. Пользователь вводит данные, которые проверяются на уникальность. Если данные уникальны, пользователь успешно регистрируется и видит страницу зарегистрированного пользователя. В противном случае система сообщает об ошибке, связанной с неправильными данными или уже существующими данными.

3.7 Диаграмма состояний регистрации

Данная диаграмма представляет процесс регистрации пользователя в системе. Пользователь начинает с запроса на регистрацию и вводит данные. Данные проверяются, и если они корректны и уникальны, выполняется регистрация, и пользователь переходит на страницу зарегистрированного пользователя. В случае, если данные уже используются или некорректны, система сообщает об ошибке и возвращает пользователя на начальный этап регистрации.



Рисунок 11 - Диаграмма состояний регистрации

3.8 Диаграмма активности поиска музыки

Данная диаграмма описывает процесс, который выполняется авторизованным пользователем при поиске музыки в системе. Пользователь вводит запрос, система проверяет наличие музыки в базе данных. Если музыка найдена, она выводится пользователю. В противном случае система сообщает, что ничего не найдено.

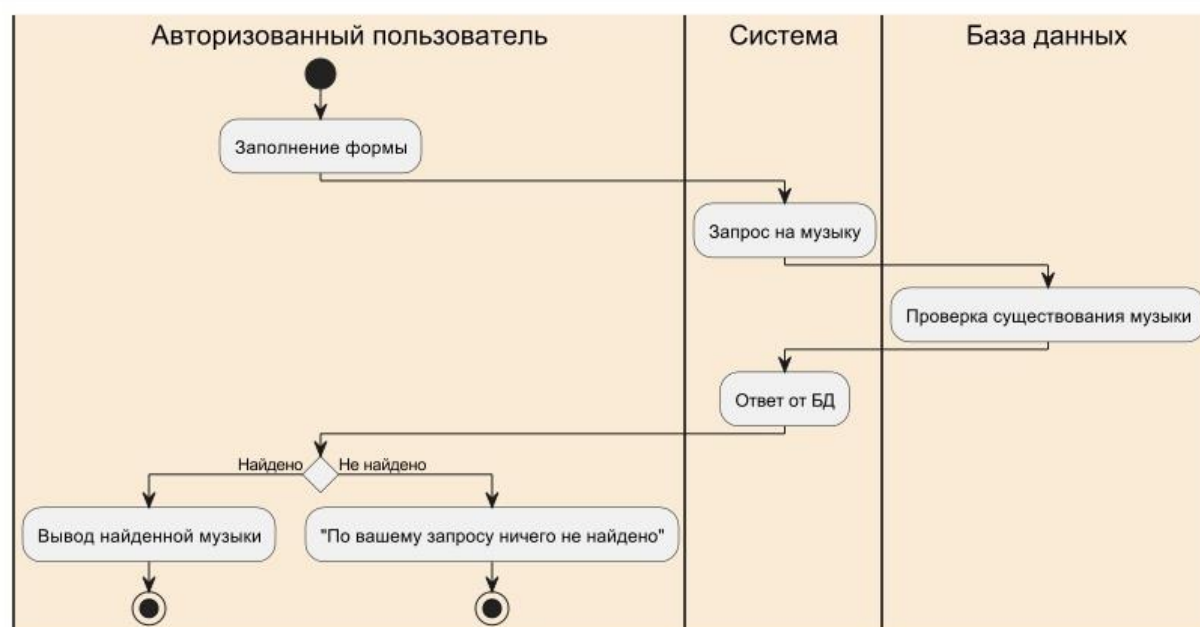


Рисунок 12 - Диаграмма активности поиска музыки

3.9 Диаграмма последовательности поиска музыки

Данная диаграмма представляет процесс поиска музыки в системе. Пользователь вводит метаданные музыки, система проверяет их в базе данных. Если музыка найдена, она выводится пользователю. В противном случае система сообщает, что ничего не найдено.

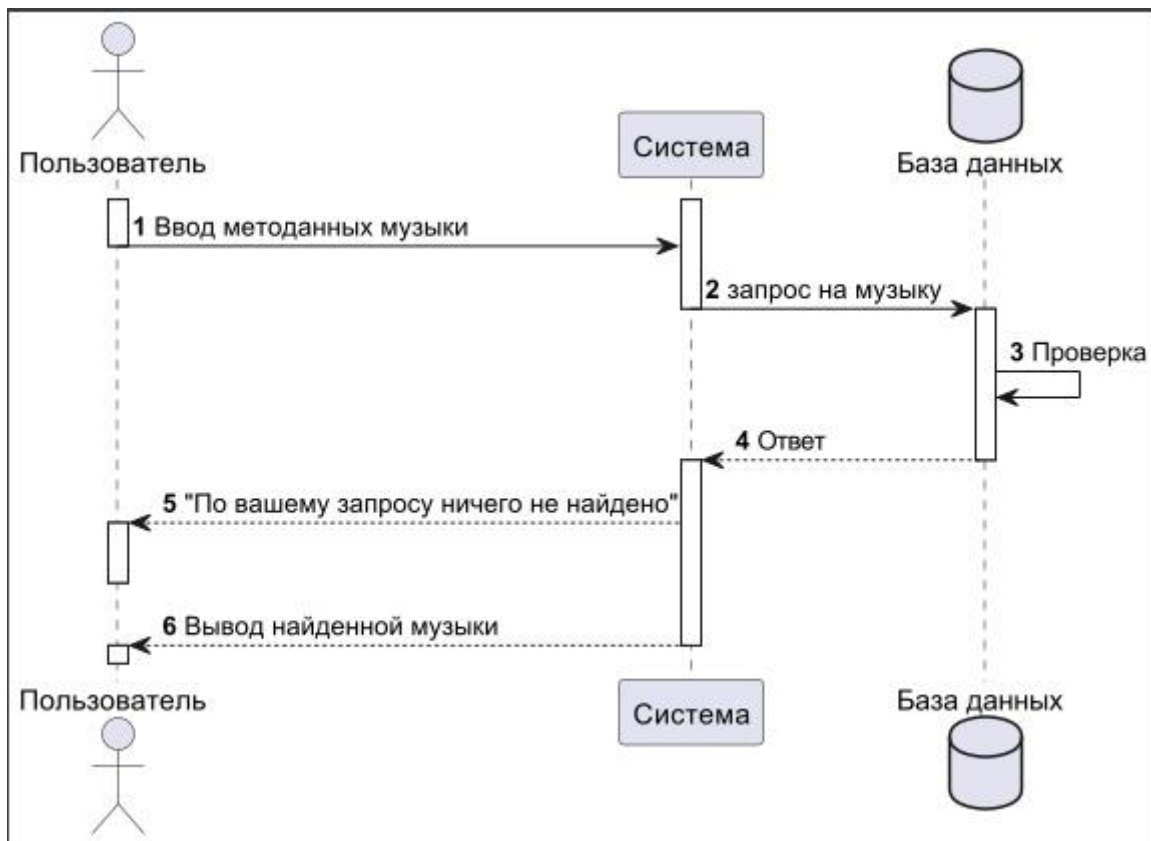


Рисунок 13 - Диаграмма последовательности поиска музыки

3.10 Диаграмма состояний поиска музыки

Данная диаграмма представляет процесс поиска музыки в системе. Пользователь начинает с запроса на музыку, и данные проверяются. Если найдены соответствующие записи, музыка выводится пользователю. В случае отсутствия данных, система сообщает, что ничего не найдено, и пользователь может выполнить новый запрос.



Рисунок 14 - Диаграмма состояний поиска музыки

3.11 Диаграмма активности пользователей

Данная диаграмма иллюстрирует процесс взаимодействия пользователя с системой. Вначале неавторизированный пользователь может либо зарегистрироваться, либо авторизоваться, после чего он переходит на главную

страницу. Затем, в зависимости от выбора, он может выполнять поиск музыки, прослушивать музыку, перейти в свой личный кабинет, где может изменять настройки профиля, просматривать избранное или, если имеет роль администратора, добавлять контент. Пользователь также может выйти из профиля. Если пользователь неавторизован или завершает сеанс, он возвращается на окно авторизации.

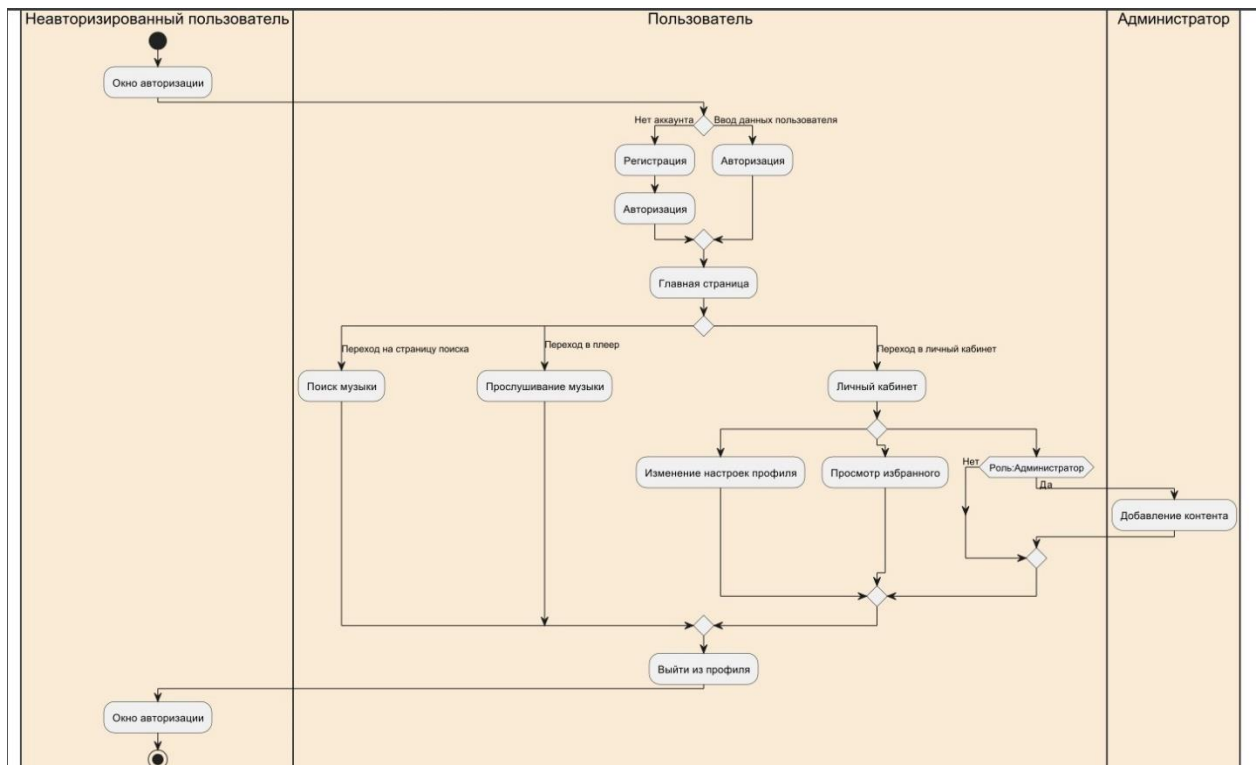


Рисунок 15 - Диаграмма активности пользователей

4 Реализация

4.1 Средства реализации

Для разработки серверной (backend) части приложения был выбран следующий стек технологий:

- Python - это интерпретируемый, высокоуровневый язык программирования, который известен своей простотой и читаемостью кода. Он поддерживает множество парадигм программирования, включая процедурное, объектно-ориентированное и функциональное программирование. Python широко используется для разработки веб-приложений, научных вычислений, искусственного интеллекта, анализа

данных и многих других областей. Python имеет обширную библиотеку сторонних модулей, что делает его мощным инструментом для разработки. Ключевые черты Python включают простоту синтаксиса, динамическую типизацию, автоматическое управление памятью и активное сообщество разработчиков.

- Django - это бесплатный и открытый фреймворк для разработки веб-приложений на языке Python. Он обеспечивает разработчиков инструментами и шаблонами для создания мощных и масштабируемых веб-приложений. Django использует модель-представление-контроллер (MVC) для организации кода и облегчения разработки. Ключевые особенности Django включают в себя встроенную административную панель, автоматическую генерацию адаптивных форм, обработку аутентификации и авторизации, управление базами данных и маршрутизацию URL. Django также предоставляет множество модулей и библиотек для упрощения задач, таких как обработка изображений, работа с электронной почтой и работа с геолокацией.
- Django Rest Framework - это мощный и гибкий инструмент для создания Web API на основе Django. Основными преимуществами являются гибкость и настраиваемость, а также множество встроенных инструментов и функций.
- PostgreSQL - реляционная база данных с открытым кодом, которая является одной из наиболее известных среди всех существующих реляционных баз данных. Ее основными преимуществами являются поддержка различных операционных систем, гибкость и настраиваемость в соответствии с потребностями пользователя, хорошая производительность при обработке большого объема данных
- Swagger - инструмент для документирования и тестирования API. Он позволяет создавать интерактивную документацию для вебсервисов, что упрощает их использование и интеграцию. Swagger автоматически

генерирует документацию на основе аннотаций в коде, что позволяет разработчикам сосредоточиться на написании логики приложения, а не на создании и поддержке документации. Благодаря Swagger, разработчики могут изучить доступные эндпоинты, параметры, модели данных и примеры запросов и ответов

Для разработки клиентской (frontend) части приложения был выбран следующий стек технологий:

- JS — это легковесный, интерпретируемый или JIT-компилируемый, объектно-ориентированный язык с функциями первого класса. Он является прототипно-ориентированным, мультипарадигменным язык с динамической типизацией, который поддерживает объектно-ориентированный, императивный и декларативный (например, функциональное программирование) стили программирования.
- React — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. Обладает широким набором инструментов, эффективностью в работе а также компонентой системой.

4.2 Реализация backend

Для реализации был использован язык программирования Python, фреймворк Django и библиотека Django Rest Framework. В качестве СУБД был использован PostgreSQL Приложение состоит из двух модулей

Первый содержит файл settings.py, определяющий основные настройки приложения, например:

- используемые библиотеки (INSTALLED_APPS)
- настройка подключения к базе данных - задаем пользователя, пароль, адрес базы данных (DATABASES)
- MEDIA_URL для доступа к медиа файлам, хранящимся у нас как статик файлы

- MIDDLEWARE — список всех промежуточных компонентов, используемых в Django для обработки запросов и ответов между сервером и клиентом.
- настройки REST Framework
- настройки авторизации (токены и тд)

Также в основном модуле настраивается автоматическая генерация документирования API Swagger.

Второй модуль содержит основную часть API. В файле `models` определены модели, используемые в приложении. `UserData` переопределяет `django` класс `AbstractBaseUser`. Также заданы модели исполнителя, плейлиста, альбома, трека и тэга. Модели данных — это классы Python, которые определяют структуру базы данных и могут быть использованы для создания или обновления схемы базы данных. Модели могут содержать поля для хранения данных (текстовые, числовые, даты, файлы и др.), а также методы для работы с этими данными;

Сериализаторы находятся в файле `serializers`. Они преобразуют данные от моделей в собственные типы данных Python. Также используются для десериализации.

Файл `views` содержит представления - основную логику приложения.

Тесты (`test.py`) — это файл в приложении, который содержит модульные тесты для этого приложения. Эти тесты используются, чтобы убедиться, что функциональные возможности приложения работают должным образом, и выявить любые ошибки или ошибки до того, как приложение будет развернуто в рабочей среде.

URL-адресация (`urls.py`) — это механизм маршрутизации запросов на определенные представления. Django использует файл `urls.py` для определения соответствующей представлению URL-адресаю

Представления (`views.py`) — это функции Python, которые обрабатывают запросы от клиента и возвращают HTTP-ответы.

Представления включают в себя логику приложения и обработку данных из моделей.

4.3 Реализация frontend

Клиентская часть приложения (frontend) разработана на языке JavaScript с использованием фреймворка React.js.

Одной из особенностей разработки приложения на React является декомпозиция приложения на независимые функциональные компоненты. Каждый компонент отвечает за определенный блок приложения и представляет собой функцию, которая возвращает HTML-код. Это позволяет упростить масштабирование приложения, сэкономить время разработки и избежать дублирования кода.

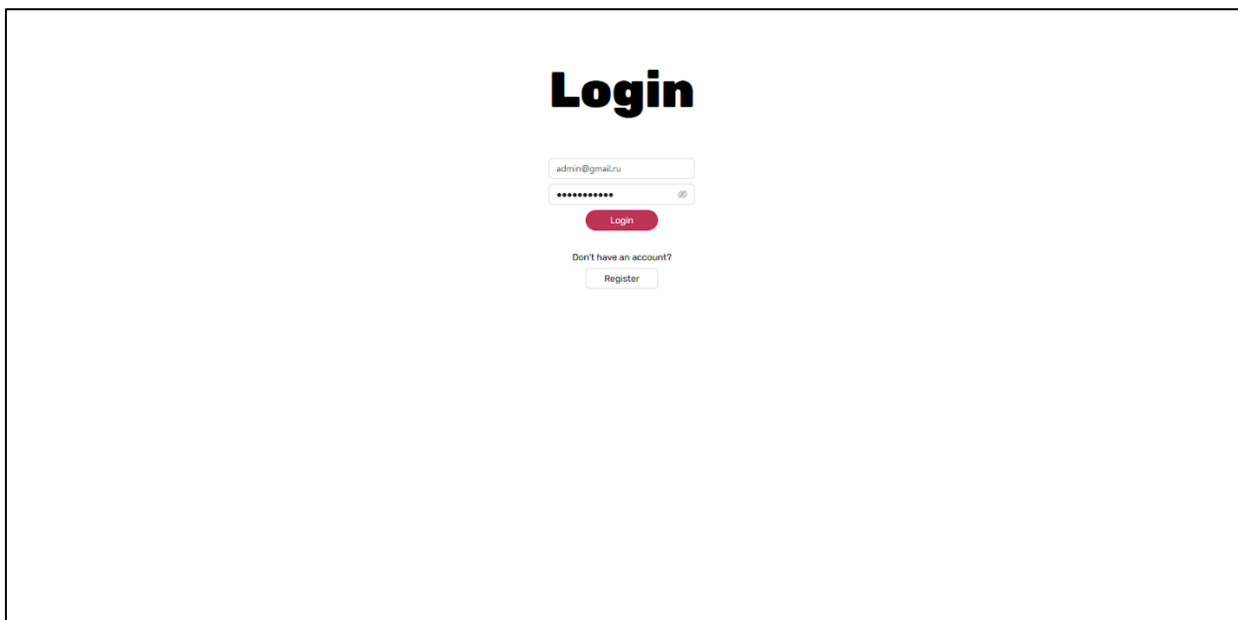
Основной HTML файл называется index.html, именно на него загружаются компоненты, которые будут выведены в браузере. В нём вызывается index.js, в котором находится точка входа React-приложения, и вызывается основной компонент App.js, который будет меняться в зависимости от действий пользователя. Все остальные компоненты вызываются по мере необходимости.

Также при написания клиентской части был использован JavaScript XML (JSX) — расширение синтаксиса JavaScript, которое позволяет использовать HTML-подобный синтаксис для описания структуры интерфейса.

Для сборки приложения был использован менеджер пакетов Yarn, так как лучше прм по скорости сборки и контролю зависимостей.

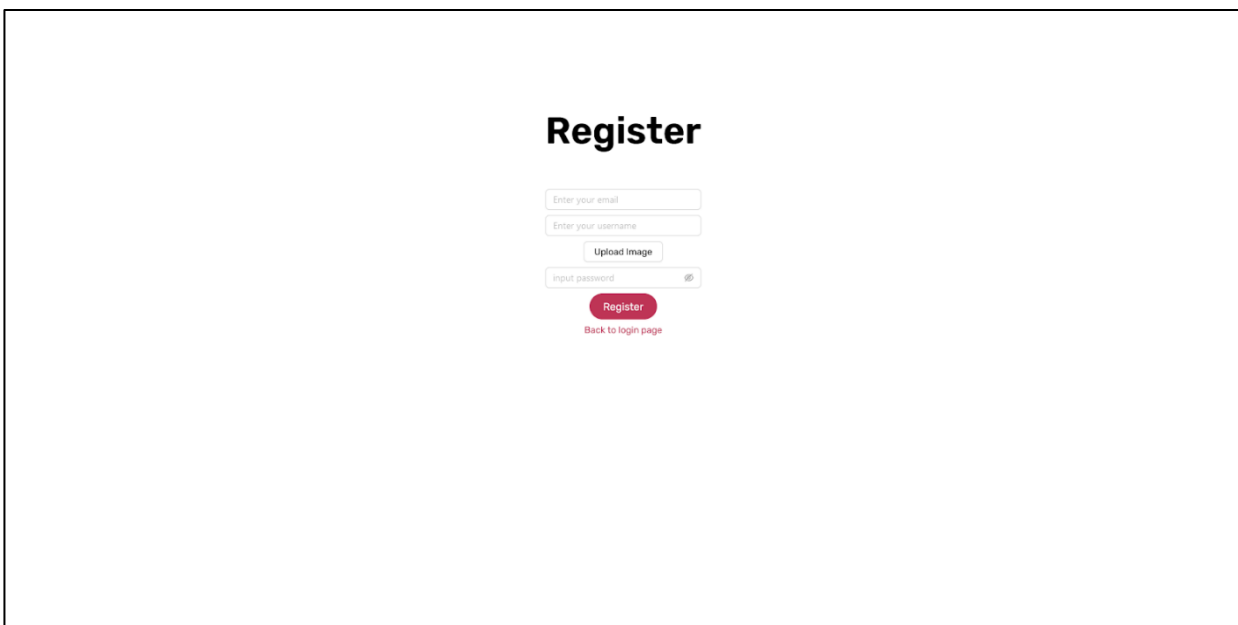
4.4 Навигация по приложению

Первой страницей, что видит пользователь при входе на сайт, является страница логина. У пользователя есть выбор: войти в уже существующий аккаунт или пройти регистрацию.



The image shows a login page with a white background. At the top center is the word "Login" in a large, bold, black font. Below it are two input fields: the first contains the text "admin@gmail.ru" and the second contains a series of dots representing a password. To the right of the password field is a small eye icon. Below the input fields is a red button with the word "Login" in white. Underneath the button is the text "Don't have an account?" and a white button with the word "Register" in black.

Рисунок 16 – Страница Login



The image shows a register page with a white background. At the top center is the word "Register" in a large, bold, black font. Below it are three input fields: the first contains the text "Enter your email", the second contains "Enter your username", and the third contains "Input password". To the right of the password field is a small eye icon. Below the input fields is a white button with the text "Upload image". Below that is a red button with the word "Register" in white. At the bottom is a link that says "Back to login page" in red text.

Рисунок 17 - Страница регистрации

После авторизации пользователь попадает на домашнюю страницу приложения, на которой отображаются популярные, рекомендованные и последние альбомы.

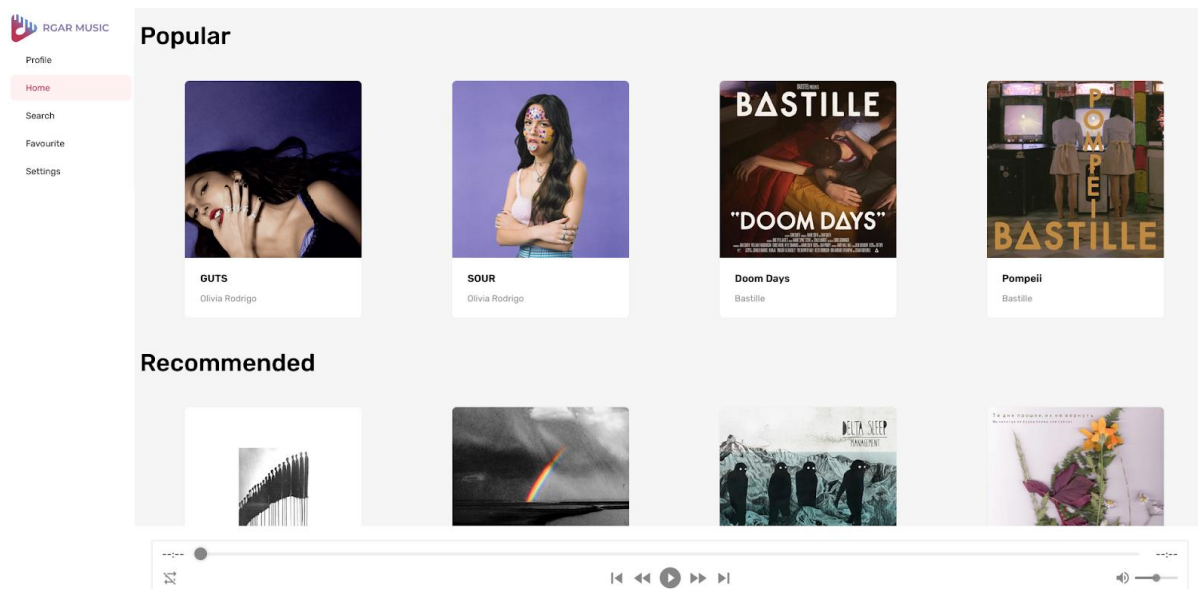


Рисунок 18 - Домашняя страница

Для навигации пользователь может использовать сайдбар. На странице личного профиля находятся созданные плейлисты, а также избранные альбомы и исполнители.

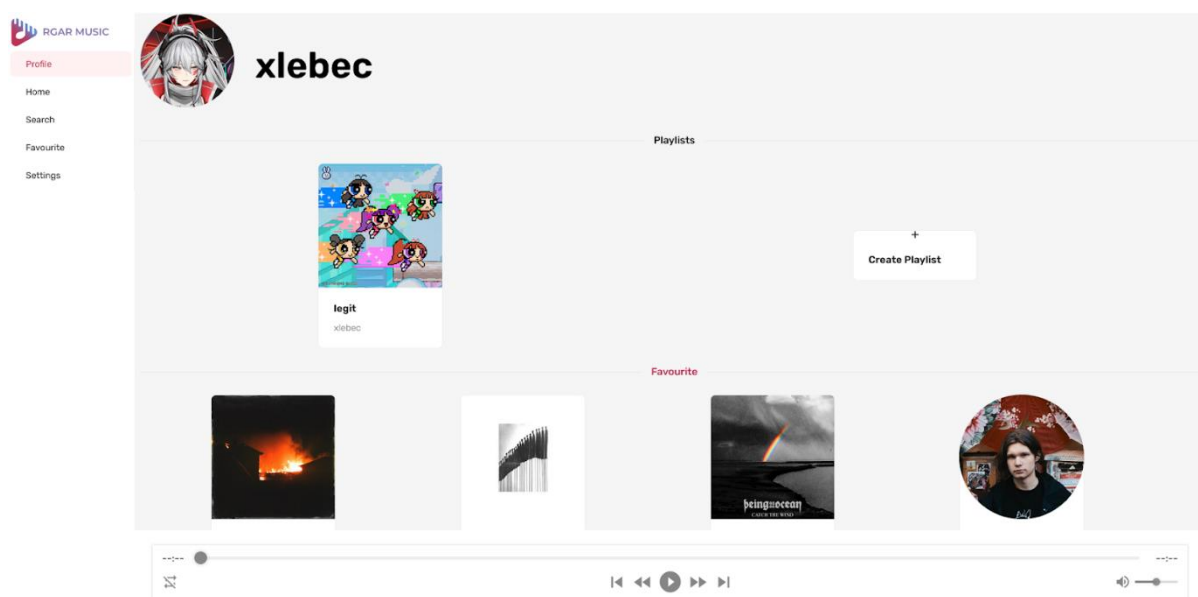


Рисунок 19 - Личный профиль пользователя

На странице поиска пользователь может по запросу найти треки, исполнителей, альбомы, плейлисты и пользователей.

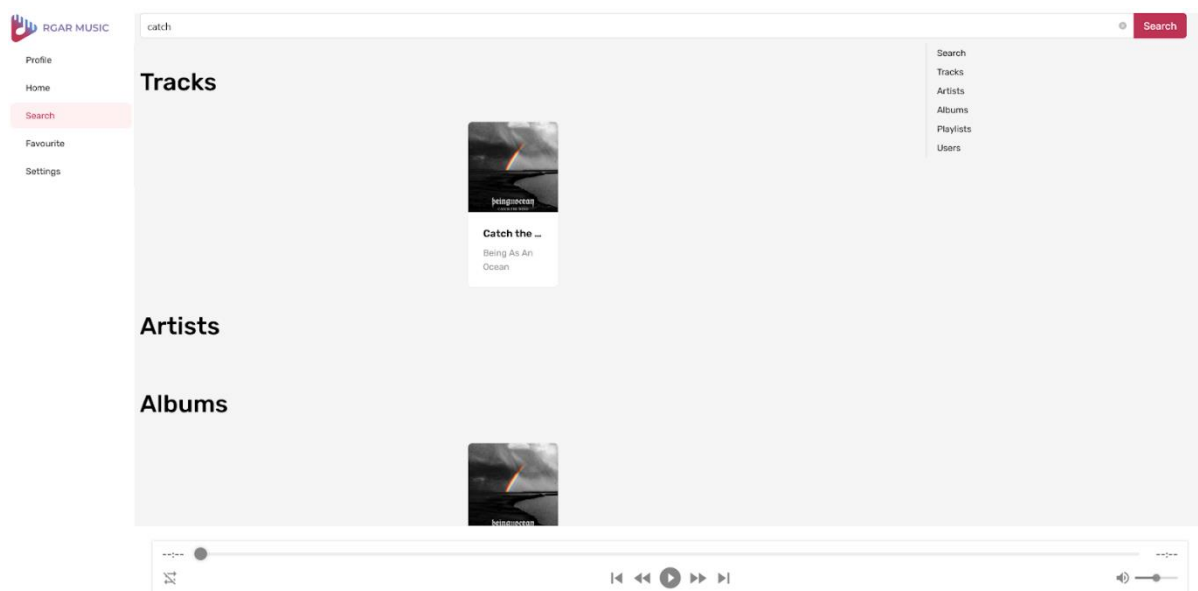


Рисунок 20 – Поиск

На странице избранного находятся лайкнутые альбомы, плейлисты, исполнители и пользователи

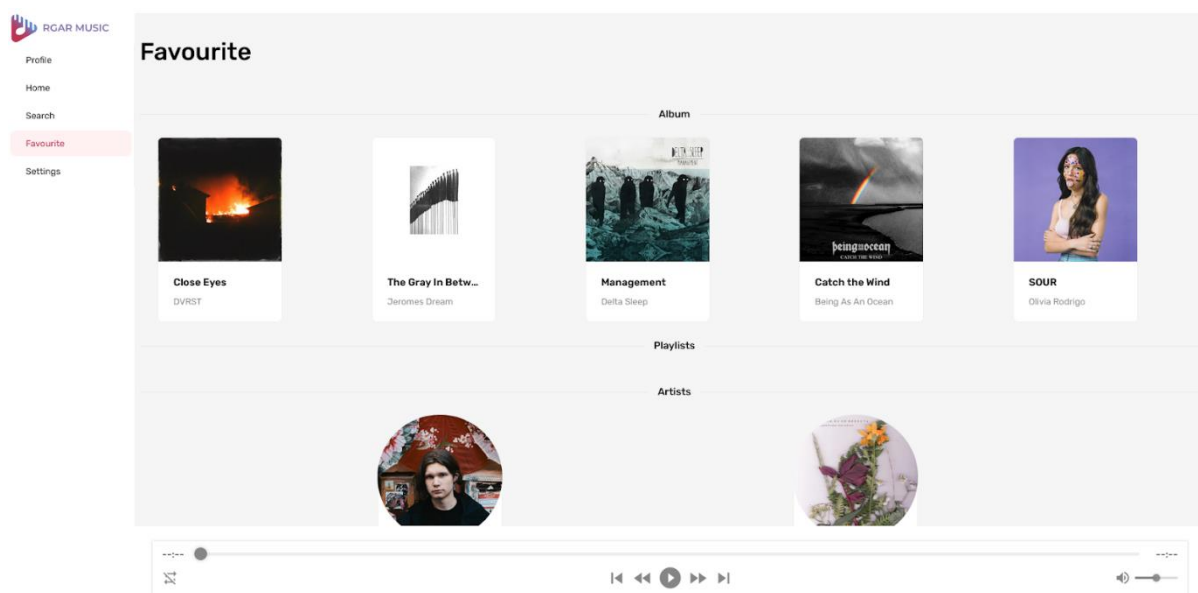


Рисунок 21 – Избранное

На экране настроек можно изменить почту, юзернейм, аватар и пароль, а также избранные теги. Для администратора доступны страницы добавления тегов, исполнителей и альбомов.

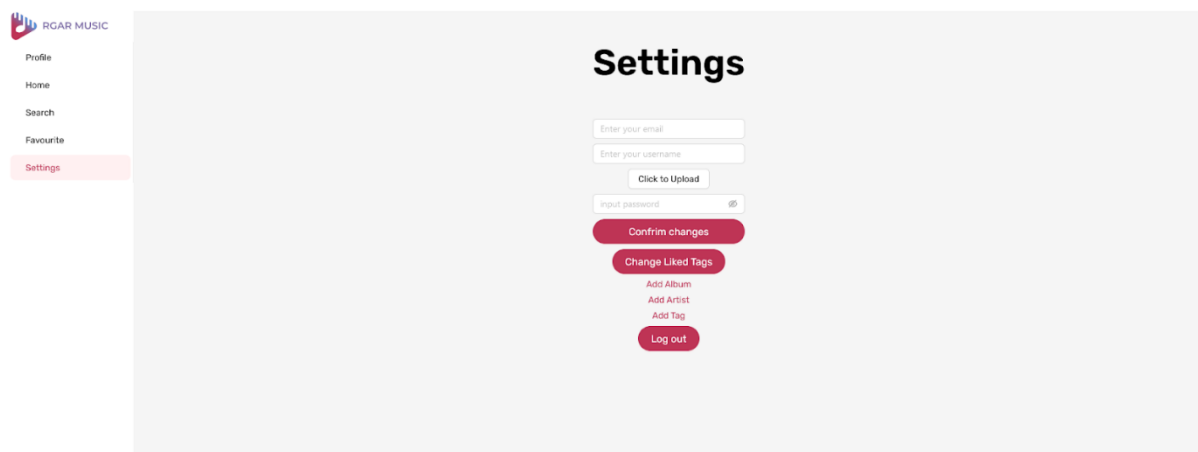


Рисунок 22 - Настройки

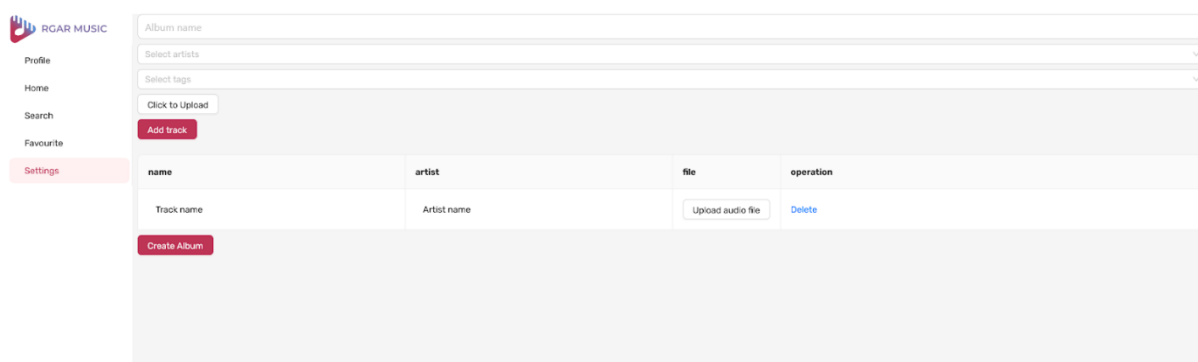


Рисунок 23 – Добавление альбома

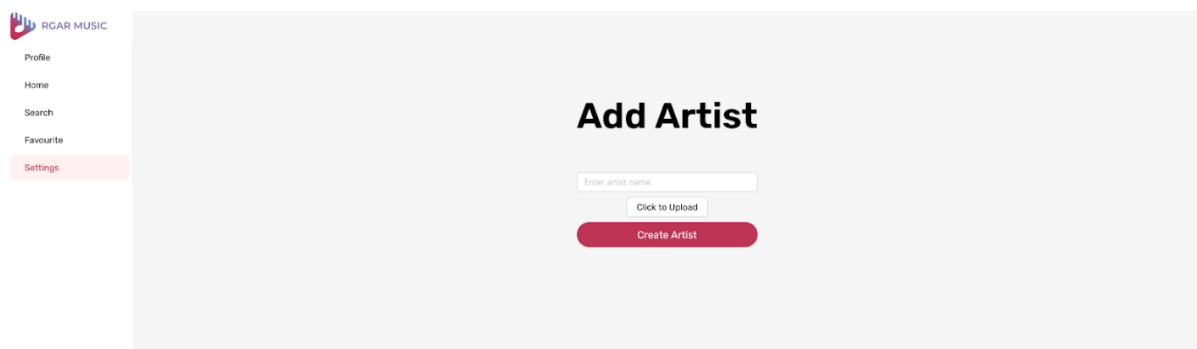


Рисунок 24 - Страница добавления исполнителя

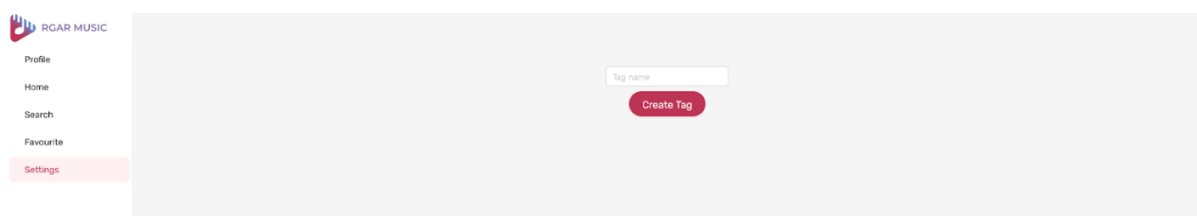


Рисунок 25 - Добавление тега

На страницах альбомов, исполнителей и плейлистов располагается список треков, а также кнопки для добавления альбома в избранное, в очередь воспроизведения, копирования ссылки на альбом. На странице исполнителя также находится список его полной дискографии.

- Profile
- Home
- Search
- Favourite
- Settings

The Gray In Between

Jeromes Dream

10 tracks

Add to queue
 Share

N	Name	Artist	Action
1	Conversations: In Time, On Mute	Jeromes Dream	Add to Playlist Add to Queue
2	Stretched invisible From London	Jeromes Dream	Add to Playlist Add to Queue
3	South By Isolation	Jeromes Dream	Add to Playlist Add to Queue
4	Pines On The Hill (With Guests)	Jeromes Dream	Add to Playlist Add to Queue
5	Cosmos In Season	Jeromes Dream	Add to Playlist Add to Queue

Glow

Being As An Ocean

00:00 05:14

Рисунок 26 - Страница альбома

- Profile
- Home
- Search
- Favourite
- Settings

Delta Sleep

Add to queue
 Share

Popular Tracks

N	Name	Action
1	16:40 AM	Add to Playlist Add to Queue
2	Camp Adventure	Add to Playlist Add to Queue

Glow

Being As An Ocean

00:00 05:14

Рисунок 27 - Страница исполнителя

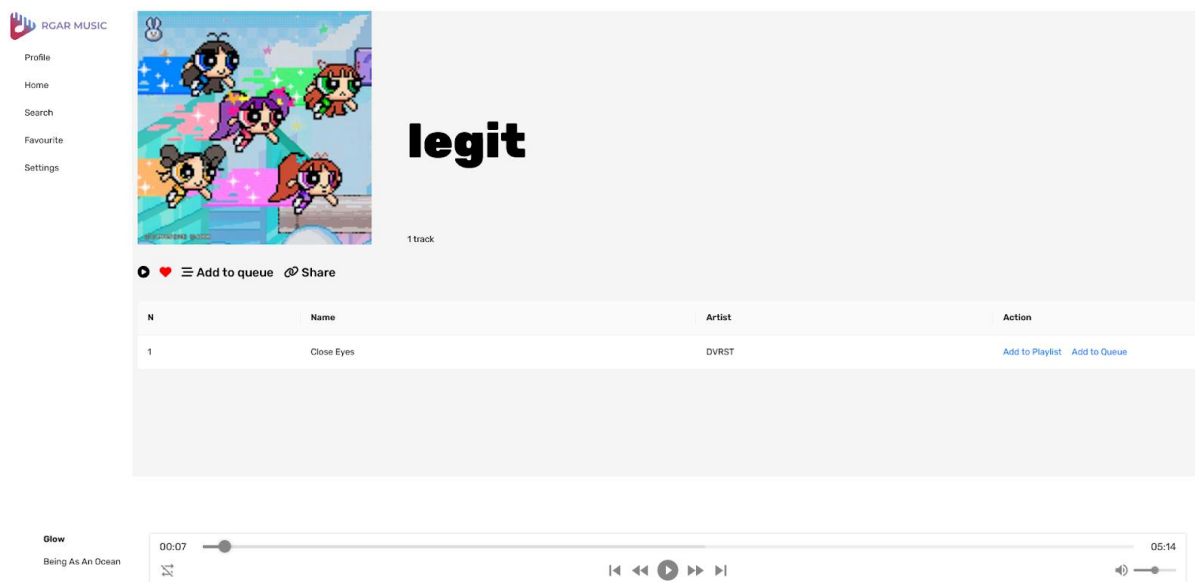


Рисунок 28 - Страница плейлиста

5 Тестирование

Работоспособность разработанного приложения проверим с помощью проведения тестирования системы с использованием следующего типа тестов:

- UI-тестирование (тестирование пользовательского интерфейса)
- Smoke testing (дымовое тестирование)

5.1 UI-тестирование

UI-тестирование (User Interface testing) - это вид функционального тестирования программного обеспечения, который сосредотачивается на проверке пользовательского интерфейса (UI) приложения или веб-сайта. Основная цель UI-тестирования - убедиться, что пользовательский интерфейс работает корректно, выглядит правильно и взаимодействует с пользователем так, как это задумано.

Таблица 1 - Результаты UI-тестирования для авторизованного пользователя

Тестовый сценарий	Ожидаемый результат	Статус теста
Нажатие кнопку «Home» для перехода в раздел	Переход на страницу «Home»	Пройден

Нажатие на кнопку «Profile»	Переход в личный аккаунт	Пройден
Нажатие на кнопку «Search»	Переход на страницу поиска	Пройден
Нажатие на кнопку «Favourite»	Переход на страницу с избранным	Пройден
Нажатие на кнопку «Settings»	Переход на страницу с настройками	Пройден
Нажатие на карточку трека со страницы «Home»	Переход на страницу с альбомом	Пройден
Нажатие на карточку альбома со страницы «Home»	Переход на страницу с альбомом	Пройден
Нажатие на карточку плейлиста со страницы «Profile»	Переход на страницу с плейлистом	Пройден
Нажатие на карточку трека со страницы «Profile»	Переход на страницу с альбомом	Пройден
Нажатие на карточку альбома со страницы «Profile»	Переход на страницу с альбомом	Пройден
Нажатие на кнопку поиска и выдача поиска по запросу со страницы «Search»	Выдача результатов согласно запросу пользователя	Пройден
Нажатие на карточку трека со страницы «Search»	Переход на страницу с альбомом	Пройден
Нажатие на карточку альбома со страницы «Search»	Переход на страницу с альбомом	Пройден
Нажатие на карточку плейлиста со страницы «Search»	Переход на страницу с альбомом	Пройден
Нажатие на карточку трека со страницы «Favourite»	Переход на страницу с альбомом	Пройден

Нажатие на карточку альбома со страницы «Favourite»	Переход на страницу с альбомом	Пройден
Нажатие на карточку плейлиста со страницы «Favourite»	Переход на страницу с альбомом	Пройден
Нажатие кнопки воспроизведения трека на странице альбома	Воспроизведение трека	Пройден
Нажатие кнопки воспроизведения трека на странице исполнителя	Воспроизведение трека	Пройден
Нажатие кнопки воспроизведения трека на странице плейлиста	Воспроизведение трека	Пройден
Нажатие круглой кнопки «Play/Pause» в плеере	Поставить на паузу воспроизведение трека. Продолжить воспроизведение трека	Пройден
Нажатие на левую кнопку с двумя треугольниками в плеере	Перемотка воспроизведения трека на 10 секунд назад	Пройден
Нажатие на правую кнопку с двумя треугольниками в плеере	Перемотка воспроизведения трека на 10 секунд вперед	Пройден
Нажатие на левую кнопку с треугольником	Вернуться на предыдущий трек	Пройден
Нажатие на правую кнопку с треугольником	Перемотать на следующий трек	Пройден

Перевод ползунка громкости в плеере	Уменьшение или увеличение громкости воспроизведения	Пройден
Нажатие на кнопку перемешивания треков в плеере	Изменения порядка очереди воспроизведения треков	Пройден

5.2 Дымовое тестирование

Дымовое тестирование (smoke testing), также известное как "smoke test," - это вид первичного функционального тестирования, который выполняется с целью быстро проверить, работает ли основной функционал приложения или системы после внесения каких-либо изменений, обновлений, или новых версий. Главная цель дымового тестирования - убедиться, что основные функции приложения или системы работают как минимум на базовом уровне, и нет критических ошибок, которые могли бы предотвратить использование приложения или системы.

Таблица 2 - Результаты дымового тестирования для пользователя

Тестовый сценарий	Результат теста
Регистрация	Пройден
Авторизация	Пройден
Просмотр страницы «Home»	Пройден
Просмотр страницы «Profile»	Пройден
Просмотр страницы «Search»	Пройден
Просмотр страницы «Favourite»	Пройден
Просмотр страницы «Settings»	Пройден
Воспроизведение трека	Пройден
Воспроизведение альбома	Пройден
Воспроизведение плейлиста	Пройден

Поиск треков, плейлистов и альбомов	Пройден
Добавление в избранное треков, плейлистов и альбомов	Пройден
Изменение информации пользователя	Пройден
Добавление альбома	Пройден
Добавление плейлиста	Пройден
Добавление артиста	Пройден

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы были выполнены все поставленные задачи. Было разработано веб-приложение музыкального плеера. Благодаря нему пользователи могут:

- Слушать музыку
- Искать среди библиотеки исполнителей, альбомов плейлистов и треков
- Слушать музыки согласно рекомендациям
- Создавать собственные плейлисты
- Создавать собственные альбомы
- Добавлять музыку

Таким образом, итоги разработки, проверенные в ходе тестирования, позволяют достигнуть поставленных заказчиком целей и решают сформулированные в начале разработки задачи.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Apple Music [Электронный ресурс]. — Режим доступа: <https://music.apple.com/us/browse> — Заглавие с экрана. — (Дата обращения: 11.07.2023).
2. Spotify [Электронный ресурс]. — Режим доступа: <https://open.spotify.com/?> — Заглавие с экрана. — (Дата обращения: 11.07.2023).
3. MTC Music [Электронный ресурс]. — Режим доступа: <https://music.mts.ru/>. — Заглавие с экрана. — (Дата обращения: 11.07.2023).
4. Backend разработка [Электронный ресурс]. — Режим доступа: <https://mobile-erp.ru/backend-razrabotka/>. — Заглавие с экрана. — (Дата обращения: 16.07.2023).
5. Postgresql [Электронный ресурс]. — Режим доступа: <https://www.postgresql.org/>. — Заглавие с экрана. — (Дата обращения: 22.07.2023).
6. Фронтенд [Электронный ресурс]. — Режим доступа: <https://www.gorkilib.ru/events/frontend>. — Заглавие с экрана. — (Дата обращения: 18.07.2023).
7. Как проводить UI-тестирование мобильных и верстки + сравнений инструментов автоматизации [Электронный ресурс]. — Режим доступа: <https://ux-journal.ru/kak-provodit-ui-testirovanie-sravnenieinstrumentov.html#testcase-example>. — Заглавие с экрана. — (Дата обращения: 25.08.2023).
8. Smoke-тестирование [Электронный ресурс]. — Режим доступа: <https://blog.skillfactory.ru/glossary/smoke-test/>. — Заглавие с экрана. — (Дата обращения: 25.08.2023).