БПОУ ОО «Сибирский профессиональный колледж»

**КУРСОВОЙ ПРОЕКТ**

по **ПМ 02 Осуществление интеграции программных модулей**

**МДК.02.02 Инструментальные средства разработки программного обеспечения**

Тема: Проектирование и разработка музыкально стримингового сервиса «Протон»

09.02.07 «Информационные системы и программирование»

Выполнил студент

ИП-432 группы 4 курса

Кузнецов Алексей

Юрьевич

Руководитель

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_

Омск 2025

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc191365876)

[1 АНАЛИТИЧЕСКИЙ РАЗДЕЛ 6](#_Toc191365877)

[1.1 Анализ предметной области. 6](#_Toc191365878)

[1.2 **Анализ аналогичных приложений** 6](#_Toc191365879)

[1.3 **Описание средств проектирования и реализации приложения** 13](#_Toc191365880)

[2 ПРОЕКТНЫЙ РАЗДЕЛ 22](#_Toc191365881)

[2.1 Разработка архитектуры приложения 22](#_Toc191365882)

[2.2 Разработка схемы базы данных 23](#_Toc191365883)

[2.3 Проектирование макетов пользовательского интерфейса 26](#_Toc191365884)

[2.4 Описание стратегии тестирования и отладки информационной системы. 29](#_Toc191365885)

[2.5 Разработка тестовых сценариев (Test-cases) для входных и выходных данных. 30](#_Toc191365886)

[2.6 Описание методов и приемов, позволяющие защитить информацию. 31](#_Toc191365887)

[3 ПРАКТИЧЕСКИЙ РАЗДЕЛ 33](#_Toc191365888)

[3.1 Анализ учебно-методических 33](#_Toc191365889)

[ЗАКЛЮЧЕНИЕ 34](#_Toc191365890)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc191365891)

[ПРИЛОЖЕНИЕ А 36](#_Toc191365892)

[ПРИЛОЖЕНИЕ Б 38](#_Toc191365893)

[ПРИЛОЖЕНИЕ В 39](#_Toc191365894)

# ВВЕДЕНИЕ

Аудио развлечения стали одной из самых популярных форм проведения свободного времени в современном обществе. В настоящее время представлено большое разнообразие артистов и музыкальных жанров, что позволяет каждому потребителю выбирать произведения, соответствующие его личным музыкальным предпочтениям. Однако, несмотря на существование множества музыкальных стриминговых сервисов, таких как Spotify, Apple Music, Яндекс. Музыка и других, остаётся актуальным создание универсальной платформы, где каждый пользователь мог бы без труда находить контент, соответствующий его интересам. В условиях цифровой эпохи доступ к миллионам аудиозаписей возможен всего лишь одним нажатием кнопки на смартфоне, компьютере или любом другом устройстве. Современные музыкальные сервисы предлагают обширные каталоги контента, но для повышения комфорта процесса поиска и воспроизведения музыки требуется разработка сервиса, способного учитывать индивидуальные предпочтения пользователей и предоставлять им персонализированные рекомендации. Такой подход позволит сделать использование музыкальной платформы еще более удобным и приятным.

Целью курсового проекта является проектирование и разработка музыкально стримингового сервиса «Протон», который обеспечит пользователям быстрый и удобный поиск необходимых музыкальных произведений. Эта система призвана минимизировать использование традиционных физических носителей информации, упростить процесс подготовки отчетов и прочей документации, связанной с музыкой, а также оптимизировать взаимодействие между пользователями и платформой.

Для достижения поставленной цели определены следующие задачи:

* провести исследование особенностей функционирования существующих музыкальных сервисов, выявить потребности целевой аудитории и выделить ключевые факторы, влияющие на удобство использования аналогичных систем;
* необходимо создать и задокументировать план тестирования и отладки программного продукта, включая его отдельные компоненты. Определить методы тестирования, критерии успешности, возможные сценарии сбоев и стратегии устранения ошибок;
* необходимо разработать и задокументировать архитектуру приложения и базы данных информационной системы. Определить основные компоненты системы, их взаимодействие и принципы интеграции. Создать структурное описание базы данных, включая ключевые сущности, связи и ограничения, обеспечивающие целостность и эффективность хранения данных;
* Необходимо создать и задокументировать стратегию тестирования и отладки информационной системы. В рамках проекта необходимо разработать тестовые сценарии (Test-cases) для проверки корректности обработки входных и выходных данных, а также описать методы и приемы защиты информации, направленные на обеспечение безопасности и целостности данных в системе;
* Разработать программный продукт в соответствии с составленной документацией. В рамках проекта необходимо реализовать функциональные компоненты системы, обеспечивая их соответствие требованиям;

Курсовой проект состоит из следующих разделов: введение, аналитический раздел, проектный раздел, практический раздел и заключение.

* Во введении определяется объект разработки курсового проекта, обосновывается актуальность проекта, формулируются цели и задачи разработки музыкального стримингового сервиса;
* В аналитическом разделе проводится анализ предметной области, рассматриваются существующие решения и их основные функциональные возможности, выявляются ключевые требования к разрабатываемому сервису;
* В проектном разделе определяется архитектура приложения, описывается взаимосвязь между его компонентами, разрабатываются основные механизмы работы сервиса, включая потоковую передачу аудиофайлов, управление плейлистами, авторизацию пользователей и алгоритмы рекомендаций;
* В практическом разделе подробно описывается процесс реализации сервиса, включая программирование функциональных модулей, разработку пользовательского интерфейса, настройку серверной части и интеграцию с базой данных и внешними API;
* В заключении подводятся итоги работы, анализируются достигнутые теоретические и практические результаты, а также оценивается эффективность разработанного решения;

# 1 АНАЛИТИЧЕСКИЙ РАЗДЕЛ

## 1.1 Анализ предметной области.

Стриминговый сервис — это платформа, которая позволяет пользователям просматривать и прослушивать контент в режиме реального времени или по запросу. Развитие интернета и высокоскоростного соединения создало огромный спрос на такие приложения.

Темой курсового проекта было выбрано разработка музыкального **стриминг-сервиса**. Сервис будет и должен обладать функциями воспроизведения, увеличения или понижения уровня громкости, поиска музыкальных исполнителей или песен. Создание плейлистов, некоторого хранящего данные блока, в котором пользователь сам составляет себе музыкальный топ.

Аккаунт, пользователь должен иметь свой аккаунт, благодаря ему он сможет создавать вышеупомянутые плейлисты, будет составляться подборка музыкальных треков и авторов по предпочтениям пользователя, представлены несколько способов авторизации-регистрации, через почту вам на почту будет отправлен код для подтверждения действий, а во второй раз вы получите личный пароль, другим способом будет регистрация через приложение тут вам ничего не будет приходить, а пароль будет создан только вами.

История, здесь будут храниться данные о треках, которые пользователь прослушал.

* 1. Анализ аналогичных приложений

*Yandex.Music*

Yandex.Music — популярный музыкальный стриминговый сервис, предлагающий пользователям широкий спектр возможностей для прослушивания музыки. Как и любой продукт, он имеет свои достоинства и недостатки. Рассмотрим их подробнее (Интерфейс на рисунке №1).

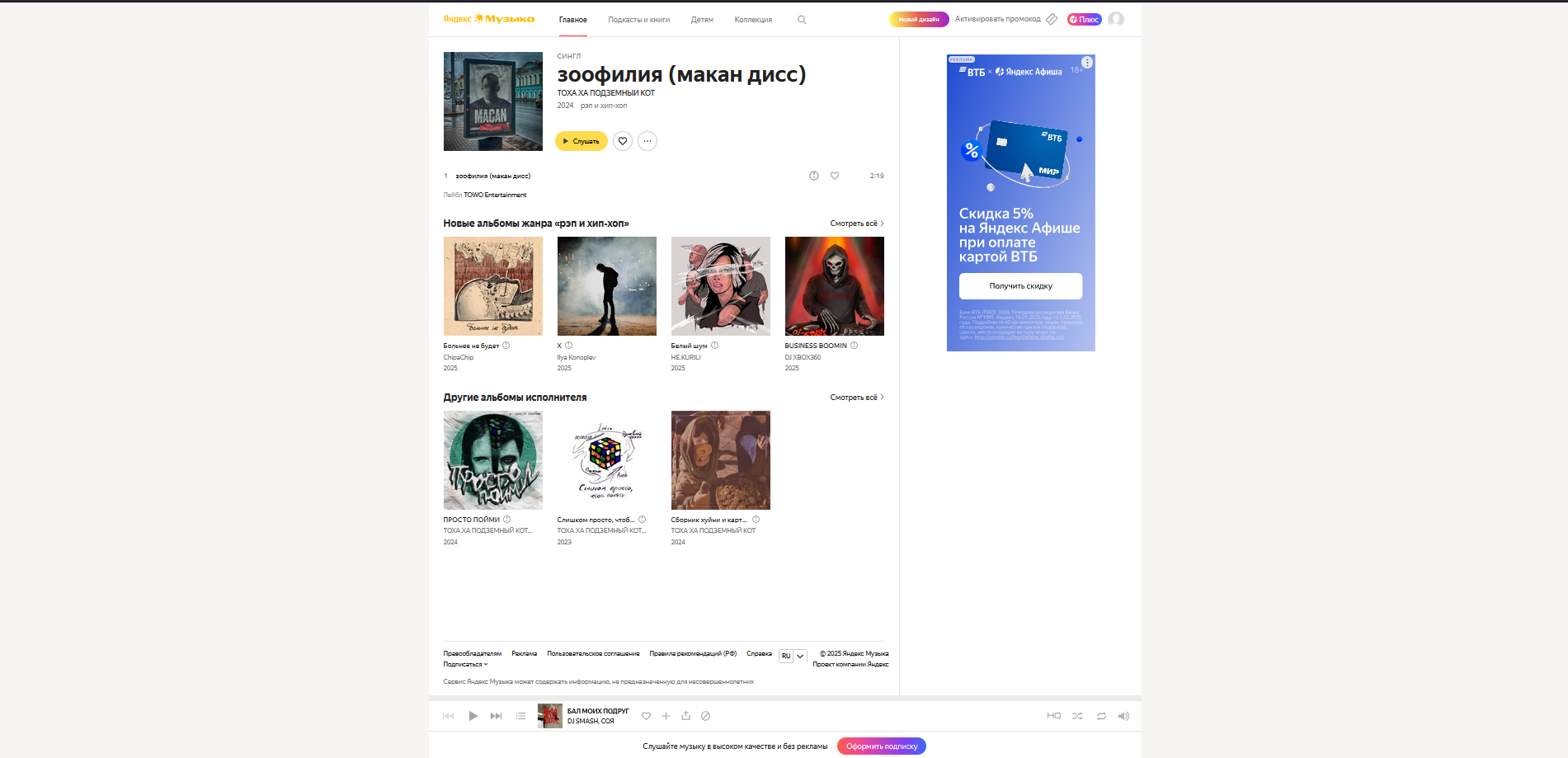


Рисунок 1 – «Интерфейс, яндекс Музыка»

*Достоинства:*

* Обширная библиотека: Yandex.Music предоставляет доступ к большому количеству музыкальных композиций различных жанров и направлений, включая российские и зарубежные исполнители.
* Персонализация:Сервис использует алгоритмы машинного обучения для создания индивидуальных плейлистов и рекомендаций на основе предпочтений пользователя. Это помогает открывать новые треки и исполнителей.
* Интеграция с другими сервисами Яндекса: пользователи могут связывать аккаунт Yandex.Music со своими аккаунтами в других сервисах Яндекса, таких как Яндекс.Почта, Яндекс.Диск и др., что упрощает управление подписками и синхронизацию данных.
* Поддержка русского языка: Интерфейс полностью локализован на русский язык, что делает использование сервиса удобным для русскоязычных пользователей.
* Радио и тематические подборки: помимо стандартных плейлистов, Yandex.Music предлагает радиостанции и тематические подборки, основанные на настроении, времени суток, событиях и т.п.
* Доступность офлайн-прослушивания: подписчики могут скачивать треки для прослушивания без подключения к интернету, что удобно при поездках или отсутствии стабильного соединения.
* Простота интерфейса: интерфейс сервиса интуитивен и удобен для навигации, что облегчает поиск нужных композиций даже новичкам.
* Социальные функции: возможность делиться музыкой с друзьями, создавать общие плейлисты и обсуждать любимые треки добавляет социальной составляющей в использование сервиса.
* Стоимость подписки: стоимость подписки на Yandex.Music относительно невысока по сравнению с аналогичными сервисами, особенно учитывая дополнительные бонусы, такие как скидка на такси и доставка еды.
* Регулярные обновления и улучшения: команда разработчиков постоянно работает над улучшением функционала и добавлением новых возможностей, что обеспечивает актуальность сервиса.

*Недостатки:*

* Ограниченная доступность за пределами СНГ: несмотря на наличие некоторых функций за границей, полноценная работа сервиса возможна преимущественно в странах СНГ. За пределами этого региона доступны ограниченные возможности.
* Реклама в бесплатной версии: бесплатная версия сервиса включает рекламные вставки, что может раздражать пользователей. Также отсутствует возможность прослушивания музыки в высоком качестве и скачивания треков для оффлайн-прослушивания.
* Проблемы с качеством звука: некоторые пользователи отмечают, что качество звучания в сервисе уступает аналогичным предложениям конкурентов, таким как Tidal или Deezer Hi-Fi.
* Отсутствие эксклюзивного контента: в отличие от Spotify или Apple Music, Yandex.Music редко предлагает эксклюзивные релизы или интервью с артистами, что снижает привлекательность для поклонников определенных исполнителей.
* Не всегда точные рекомендации: хотя система рекомендаций в целом работает хорошо, иногда она может предлагать неподходящие треки или повторять одни и те же композиции слишком часто.
* Зависимость от интернета: для полноценного использования сервиса требуется постоянное подключение к интернету, что может быть неудобным в местах с плохим покрытием сети.
* Недостаток интеграции с некоторыми устройствами: поддержка сторонних устройств и приложений может быть ограниченной, что затрудняет использование сервиса на некоторых платформах.
* Политика конфиденциальности: как и многие другие крупные компании, Яндекс собирает данные о своих пользователях, что вызывает вопросы у тех, кто обеспокоен защитой своей приватности.

*Итог:*

Таким образом, Yandex.Music представляет собой качественный и удобный музыкальный сервис с рядом преимуществ, но и с определенными недостатками, которые стоит учитывать при выборе платформы для прослушивания музыки.

*YouTube Music:*

YouTube Music – это музыкальный стриминговый сервис от Google, который позволяет пользователям слушать музыку, смотреть музыкальные клипы и создавать плейлисты. Вот основные характеристики сервиса, а также его плюсы и минусы (Интерфейс на рисунке №2):

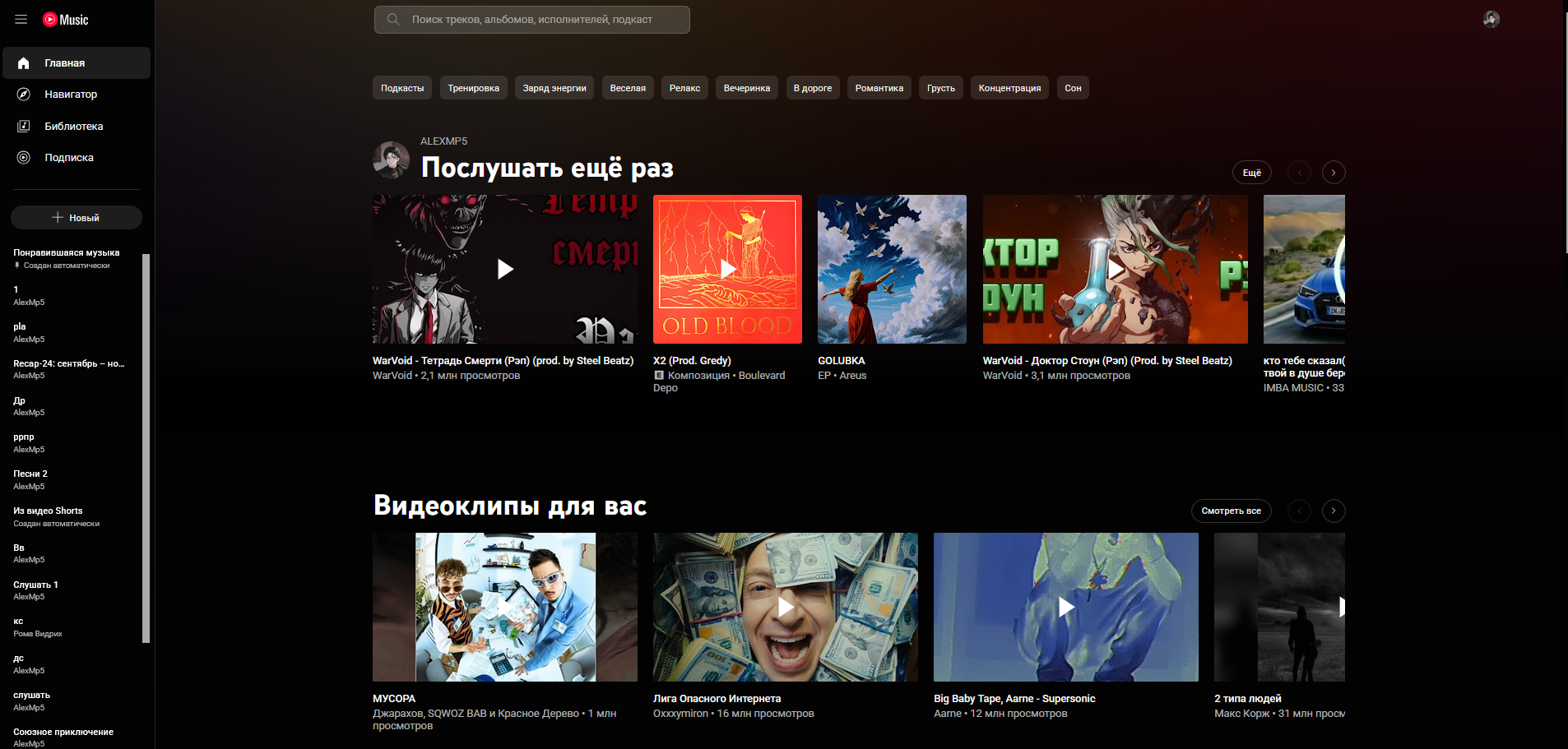


Рисунок 2 – «Интерфейс, YouTube Music»

*Достоинства YouTube Music:*

* Огромная библиотека: Огромное количество музыкальных композиций, включая редкие записи, ремиксы и живые выступления.
* Видеоклипы: Уникальная возможность просмотра музыкальных видео прямо в приложении.
* Умные рекомендации: Хорошие алгоритмы подбора музыки, которые помогают открывать новые треки и исполнителей.
* Простота использования: интерфейс удобен и прост в использовании, даже для новичков.
* Бесплатная версия: есть возможность использовать сервис бесплатно, хотя с ограничениями по функционалу.
* Мобильность: Приложение доступно на большинстве платформ, что делает его удобным для использования в любом месте.
* Качество звука: Высокое качество звучания, особенно в премиум-версии.
* Синхронизация с YouTube Premium: Подписчики YouTube Premium получают доступ ко всем функциям YouTube Music без рекламы.

*Недостатки YouTube Music:*

* Реклама в бесплатной версии: Постоянные рекламные вставки при бесплатном использовании могут раздражать.
* Отсутствие некоторых функций в бесплатной версии: например, отсутствие возможности скачивать треки для офлайн-прослушивания и ограничения на пропуск треков.
* Не всегда точные рекомендации: иногда алгоритмы могут предлагать не совсем подходящие треки.
* Цена подписки: Стоимость премиум-подписки может быть выше, чем у конкурентов, таких как Spotify или Apple Music.
* Некоторые функции доступны только в премиум-версии: Офлайн-режим, отсутствие рекламы и другие полезные функции доступны только за дополнительную плату.
* Нет поддержки Hi-Fi аудио: в отличие от некоторых других сервисов, YouTube Music пока не поддерживает высококачественное аудио формата FLAC или аналогичных форматов.

*Итог:*

YouTube Music – это мощный инструмент для любителей музыки, предлагающий огромный выбор треков, интеграцию с видео платформой YouTube и удобные функции для персонализированного прослушивания. Однако, чтобы получить все преимущества, придется оформить платную подписку, которая стоит дороже, чем у некоторых конкурентов.

*Spotify:*

Spotify — это популярный музыкальный стриминговый сервис с обширной библиотекой песен и удобными функциями рекомендаций, однако он имеет ограничения в бесплатном доступе и региональном контенте. (Интерфейс на рисунке №3)

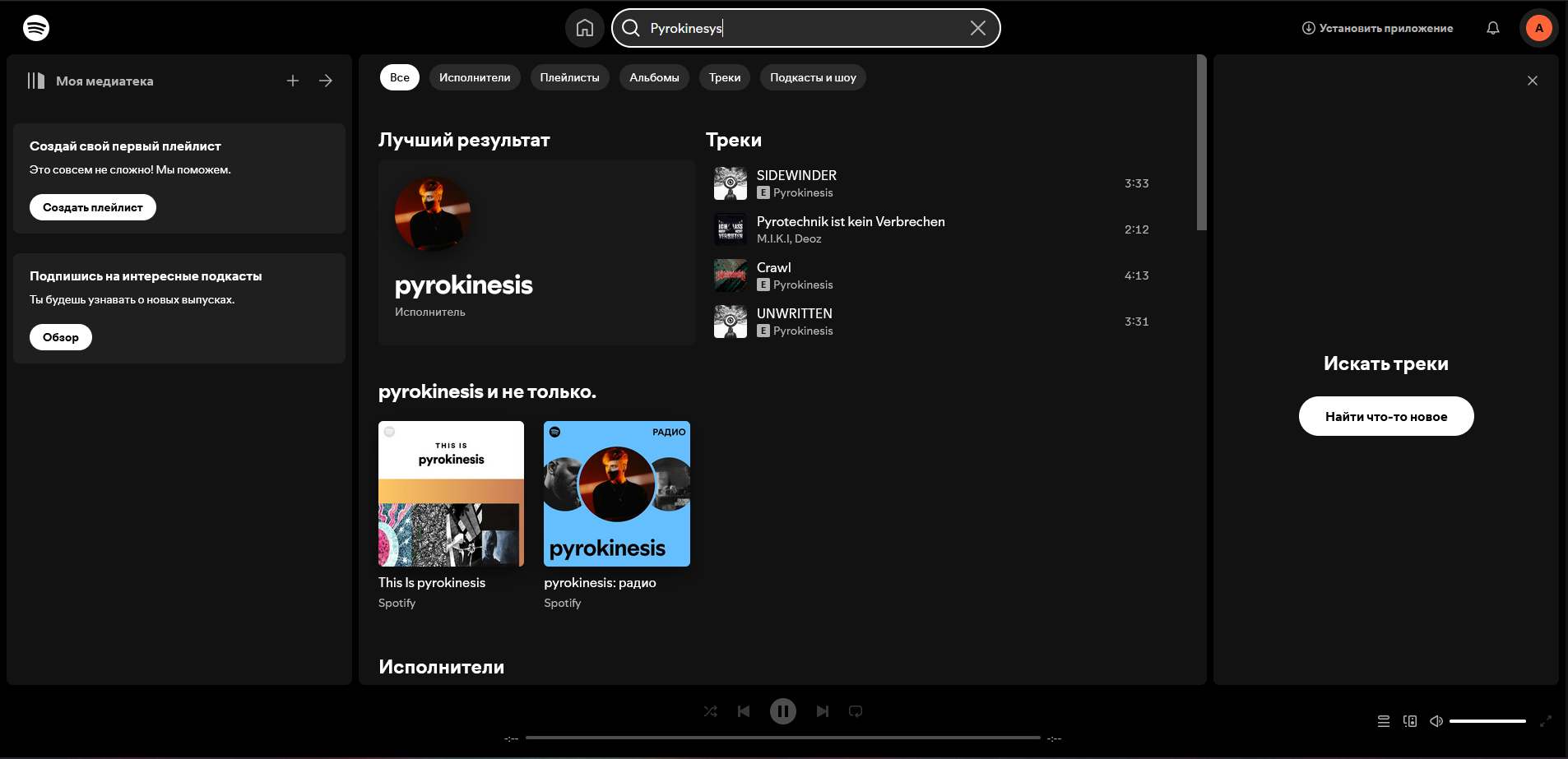


Рисунок 3 – «Интерфейс, Spotify»

*Плюсы Spotify:*

* Широкий ассортимент музыки: Огромная база данных, включающая практически всех популярных исполнителей и жанры.
* Отличные рекомендации: Алгоритмы Spotify хорошо справляются с подбором новой музыки, соответствующей вкусам пользователей.
* Удобство использования: Простой и интуитивно понятный интерфейс, который подходит как новичкам, так и опытным пользователям.
* Доступность на разных платформах: можно пользоваться сервисом на любых устройствах, будь то смартфон, компьютер или телевизор.
* Высокое качество звука: В премиум-версии предлагается отличное качество аудиопотока, которое удовлетворит большинство слушателей.
* Социальные функции: легко делиться музыкой с друзьями и узнавать, что слушают другие люди.
* Разнообразие тарифов: Наличие бесплатных и премиальных версий, а также специальных предложений

*Недостатки Spotify:*

* Ограниченный бесплатный доступ: Бесплатная версия имеет рекламу и ограничивает возможности управления воспроизведением треков.
* Качество звука: хотя Spotify предлагает хорошее качество звука, оно уступает некоторым конкурентам, таким как Tidal или Qobuz, предлагающим Hi-Fi звук.
* Отсутствие некоторых исполнителей и альбомов: Некоторые артисты могут временно или постоянно отсутствовать на платформе из-за лицензионных ограничений.
* Региональные ограничения: Доступность контента может различаться в зависимости от региона, что иногда приводит к тому, что пользователи не могут слушать определенные песни или альбомы.
* Интерфейс и удобство использования: Некоторым пользователям интерфейс кажется перегруженным или сложным для навигации.
* Проблемы с авторскими правами: В прошлом были случаи, когда артисты жаловались на низкую оплату за их музыку со стороны сервиса.

*Итог:*

Spotify является одним из ведущих музыкальных стриминговых сервисов благодаря широкому выбору музыки, отличным рекомендациям, удобству использования и доступности на различных платформах. Однако он имеет свои недостатки, такие как ограниченный бесплатный доступ, региональные ограничения и проблемы с качеством звука по сравнению с некоторыми конкурентами. Кроме того, некоторые пользователи могут столкнуться с неудобствами интерфейса и отсутствием определенных исполнителей или альбомов.

Несмотря на эти минусы, Spotify остается популярным выбором среди любителей музыки благодаря своим сильным сторонам и разнообразию тарифных планов.

* 1. **Описание средств проектирования и реализации приложения**

Для разработки данного проекта будут применены средства разработки MySQL для создания общей базы данных по предметной области, а для разработки пользовательского интерфейса и составления основных скриптов будет использована Visual Studio 2022 — мощный инструмент для разработки программного обеспечения, который предлагает интегрированную среду разработки (IDE) для Windows, Mac и Linux. Также для разработки будет использован PyCharm — интегрированная среда разработки (IDE) для языка программирования Python, созданная компанией JetBrains. Она поддерживает широкий спектр функций, таких как автодополнение кода, отладка, тестирование и работа с системами контроля версий.

О каждом из них подробнее:

* **MySQL** — популярная система управления базами данных (СУБД) с открытым исходным кодом, которая широко используется для хранения и обработки данных. MySQL поддерживает работу с таблицами, индексами, хранимыми процедурами, триггерами и видами, обеспечивая удобные инструменты для администрирования и управления пользователями.
* **Visual Studio 2022** предоставляет разработчикам широкие возможности для создания мощных и профессионально выглядящих приложений. Среда разработки поддерживает использование различных языков программирования (например, C#, Visual Basic), а также различные технологии для разработки приложений для разных платформ.
* **PyCharm** — специализированная среда разработки, ориентированная на работу с языком Python. Она предлагает комплексные инструменты для этого языка, включая анализ кода, отладку и тестирование. Некоторые возможности PyCharm:
  + Поддержка работы с базами данных MySQL, Oracle, PostgreSQL через плагин Database Navigator.
  + Интеграция с Docker, Kubernetes и AWS.
  + Мощная работа с Git, включая выборочную сортировку, перемещение и разрешение конфликтов.
  + Кроссплатформенная совместимость (Windows, macOS, Linux), что делает PyCharm гибким инструментом для разработчиков.

PyCharm особенно подходит для питонистов, занимающихся Data Science, машинным обучением или веб-программированием.

Сочетание этих трех инструментов позволит проектировать и разрабатывать информационную систему эффективно, а также обеспечит удобство в работе с базой данных и пользовательским интерфейсом. Благодаря возможностям MySQL и Visual Studio 2022, процесс создания и сопровождения проекта будет максимально продуктивным и эффективным.

* 1. Описание стратегии тестирования и отладки проекта
     1. Описание стратегии тестирования

Тестирование в программировании — это процесс проверки программного обеспечения на соответствие требованиям и выявление ошибок. Основная цель тестирования — убедиться, что программа работает корректно и выполняет поставленные задачи.

Основные виды тестирования:

* Функциональное тестирование: Проверка того, что программа выполняет все свои функции в соответствии с требованиями.
* Нефункциональное тестирование: Оценка нефункциональных характеристик программы, таких как производительность и удобство использования.
* Автоматизированное тестирование: Использование специальных инструментов и скриптов для выполнения тестов.
* Ручное тестирование: Выполнение тестов вручную без использования автоматических инструментов.
* Юнит-тестирование: Проверка отдельных модулей или функций программы.
* Интеграционное тестирование: Проверка взаимодействия между различными модулями или компонентами программы.
* Системное тестирование: Проверка всей системы в целом, включая все ее компоненты и взаимодействие с внешней средой.
* Регрессионное тестирование: Проверка того, что изменения в коде не привели к появлению новых ошибок или не нарушили существующую функциональность.
* Нагрузочное тестирование: Проверка способности программы выдерживать высокие нагрузки.
  + 1. Описание стратегии дебагинга

Дебагинг (от англ. debugging) — это процесс поиска и устранения ошибок (багов) в программном коде. Это важная часть разработки программного обеспечения, так как ошибки могут привести к неправильной работе программы, снижению производительности или даже к критическим сбоям.

Основные этапы процесса дебагинга:

* Идентификация проблемы:

Определение симптомов ошибки, например, программа падает, выдает некорректные результаты или работает слишком медленно.

* Репликация проблемы:

Повторение условий, при которых возникает ошибка, чтобы убедиться, что она воспроизводима.

* Локализация ошибки:

Использование отладчиков, логов или других инструментов для определения места в коде, где происходит ошибка.

* Анализ и исправление:

Понимание причины ошибки и внесение изменений в код для ее устранения.

* Проверка исправления:

Повторное тестирование программы, чтобы убедиться, что ошибка устранена и не появились новые проблемы.

*Инструменты для дебагинга:*

Отладчики: специальные программы, которые позволяют пошагово выполнять код, устанавливать точки останова, просматривать значения переменных и отслеживать выполнение программы.

Логирование: запись информации о работе программы в лог-файлы, что помогает отслеживать последовательность событий и выявлять ошибки.

Профилирование: анализ производительности программы, позволяющий выявить узкие места и оптимизировать код.

*Логирование*

Логирование — это процесс записи информации о выполнении программы в специальный файл или консоль. Это помогает отслеживать, что происходит в программе в реальном времени, и выявлять ошибки.

В ключевые места программы добавляются команды, которые записывают информацию о выполнении. Например, можно записывать:

* Входные данные функции.
* Значения переменных.
* Результаты выполнения операций.
* Сообщения об ошибках.

*Преимущества:*

* Позволяет отслеживать выполнение программы без необходимости пошагового выполнения,
* Можно анализировать логи после выполнения программы,
* Помогает выявлять ошибки, которые возникают в сложных или асинхронных процессах;

*Недостатки:*

* Может замедлить выполнение программы, если логирование слишком подробное,
* Требует времени на анализ логов;

*Пошаговое выполнение*

Пошаговое выполнение — это процесс выполнения программы по одной строке кода за раз с помощью отладчика. Это позволяет отслеживать значения переменных и выполнение условий в реальном времени.

Отладчик позволяет устанавливать точки останова (breakpoints) в коде. Когда программа достигает точки останова, выполнение приостанавливается, и вы можете проверить значения переменных, выполнить код пошагово или продолжить выполнение.

*Преимущества:*

* Позволяет детально изучить выполнение программы,
* Помогает выявлять ошибки в сложных алгоритмах,
* Можно быстро проверить значения переменных и результаты выполнения;

*Недостатки:*

* Требует времени на установку точек останова и пошаговое выполнение,
* Не всегда удобно для больших и сложных программ;

*Профилирование*

Профилирование — это процесс анализа производительности программы. Оно помогает выявить узкие места, где программа тратит больше всего времени или ресурсов.

Инструменты профилирования собирают данные о времени выполнения каждой функции, количестве вызовов и использовании памяти. Это позволяет определить, какие части программы требуют оптимизации.

*Преимущества:*

* Помогает оптимизировать программу, выявляя узкие места,
* Позволяет улучшить производительность и эффективность;

*Недостатки:*

* Требует времени на анализ результатов профилирования,
* Может быть сложно интерпретировать данные для больших и сложных программ;

*Точки останова*

Точки останова — это места в коде, где выполнение программы приостанавливается, чтобы вы могли проверить значения переменных и состояние программы.

Вы устанавливаете точки останова в ключевых местах программы, и когда программа достигает этих точек, выполнение приостанавливается. Вы можете проверить значения переменных, продолжить выполнение или выполнить код пошагово.

*Преимущества:*

* Позволяет детально изучить выполнение программы,
* Помогает выявлять ошибки в сложных алгоритмах,
* Можно быстро проверить значения переменных и результаты выполнения;

*Недостатки:*

* Требует времени на установку точек останова и пошаговое выполнение.
* Не всегда удобно для больших и сложных программ.

*Тестовые данные*

Тестовые данные — это специально подготовленные входные данные, которые используются для проверки различных сценариев выполнения программы.

Вы создаете набор тестовых данных, которые охватывают различные возможные сценарии выполнения программы. Затем вы запускаете программу с этими данными и проверяете, что она работает корректно.

*Преимущества:*

* Позволяет проверить различные сценарии выполнения программы,
* Помогает выявлять ошибки, которые могут возникнуть в реальных условиях,
* Можно автоматизировать тестирование с помощью тестовых данных;

*Недостатки:*

* Требует времени на подготовку тестовых данных,
* Не всегда возможно охватить все возможные сценарии;

## 2 ПРОЕКТНЫЙ РАЗДЕЛ

## 2.1 Разработка архитектуры приложения

*диаграмма прецедентов:*

Диаграмма прецедентов (Use Case Diagram) является одним из основных элементов языка моделирования UML (Unified Modeling Language), применяемого для визуализации требований к программному обеспечению или системе. Она отображает взаимодействие между актерами (пользователями или внешними системами) и системой, демонстрируя функции (прецеденты), которые система должна выполнять для удовлетворения потребностей этих актеров.

На рисунке 4 показана диаграмма, объясняющая, как пользователи и система взаимодействуют с сервисом. Основные части диаграммы:

*Участники (акторы):*

* пользователь: человек, использующий систему;

*действия (прецеденты):*

* создать плейлист: пользователь может сделать новый список воспроизведения,
* добавить композиции: пользователь добавляет музыку в плейлист,
* слушать музыку: пользователь воспроизводит плейлисты,
* искать авторов и треки: пользователь ищет конкретные песни или исполнителей,
* посмотреть историю: пользователь проверяет, что слушал раньше,
* вход в систему: пользователь входит в аккаунт;

Эти действия показывают, как пользователь работает с музыкальным плеером: создает плейлисты, добавляет музыку, слушает её и просматривает историю.

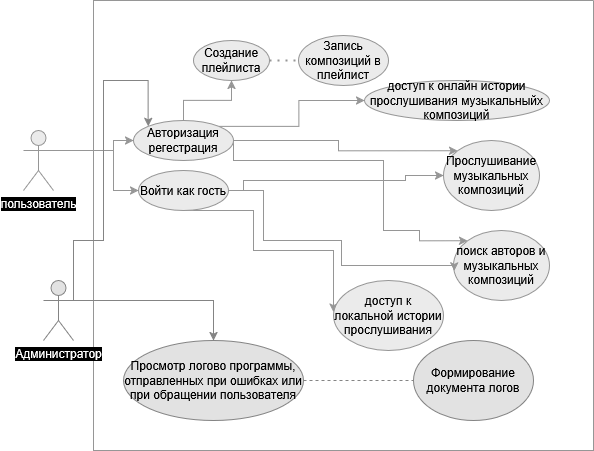


Рисунок 4 –«Диаграмма прецедентов»

## 2.2 Разработка схемы базы данных

База данных (БД) — это организованная совокупность данных, предназначенная для хранения, управления и извлечения информации. Базы данных используются для структурированного хранения больших объемов данных таким образом, чтобы эти данные могли легко обрабатываться, обновляться и извлекаться по запросу.

Для рассматриваемого музыкального сервиса база данных необходима для выполнения следующих задач:

* Хранение данных пользователей: информация о каждом зарегистрированном пользователе (имя, электронный адрес, пароль) должна сохраняться в надежном месте. Это позволяет сервису управлять доступом пользователей к своим аккаунтам и защищать конфиденциальность личных данных,
* Создание и управление плейлистами: пользователи создают плейлисты, добавляют в них музыку и управляют ими. Все эти действия требуют сохранения информации о плейлистах, включая названия, состав треков и связь с пользователями,
* Каталогизация музыки: музыкальные треки должны быть организованы таким образом, чтобы пользователи могли находить нужную песню, узнавать информацию о ней (название, исполнитель), а также воспроизводить её,
* История прослушивания: для улучшения пользовательского опыта важно сохранять историю прослушанной музыки. Это помогает рекомендовать новые треки, показывать часто прослушиваемые композиции и давать пользователю возможность вернуться к ранее услышанным песням,
* Эффективное взаимодействие между компонентами системы: база данных позволяет эффективно связывать различные элементы системы (пользователей, плейлисты, треки) через использование внешних ключей. Например, зная ID пользователя, можно быстро получить список всех его плейлистов, а зная ID плейлиста — все входящие в него треки;

Таким образом, база данных служит основой для работы всего музыкального сервиса, обеспечивая сохранность, доступность и управляемость всех необходимых данных. Структура базы данных (ER-Диaграмма), представлена на рисунке 5.

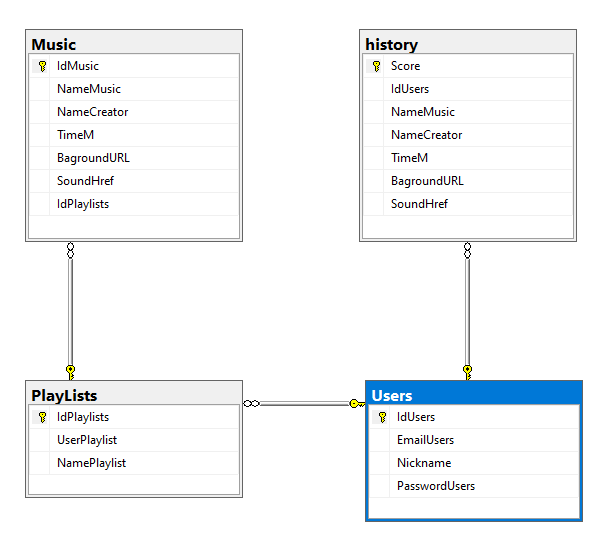


Рисунок 6 – «ER-Диаграмма базы данных»

*Таблица Users имеет поля:*

* IdUsers – порядковый номер пользователя в базе данных,
* EmailUsers – является авторотационным ключом пользователя для входа в программу под своей учетной записью,
* Nickname – это псевдоним, который используется пользователем в интернете, вместо реального имени,
* PasswordUsers – мера безопасности при авторизации пользователя, созданная в данном онлайн сервисе;

*Таблица PlayLists имеет поля:*

* Idplaylists - порядковый номер плейлиста в базе данных,
* UserPlaylist – внешний ключ для обеспечения правильности вывода информации о создателе,
* NamePlaylist – Название плейлиста;

*Таблица Music содержит разделы под названием:*

* IdMusic – Идентификационный код песни,
* NameMusic – Название песни,
* NameCreator – Наименование автора песни,
* TimeM – продолжительность музыкального произведения,
* BackgroundURL – ссылка на изображение музыкального произведения,
* SoundHref – ссылка на файл музыкального произведения, для его воспроизведения в программе,
* IdPlaylists – идентификационный номер плейлиста, в котором находится песня;

*Таблица History содержит разделы:*

* Score – Идентификационный номер песни в истории,
* IdUsers – порядковый номер пользователя,
* NameMusic – Название песни,
* NameCreator – Наименование автора песни,
* TimeM – продолжительность музыкального произведения,
* BackgroundURL – ссылка на изображение музыкального произведения,
* SoundHref – ссылка на файл музыкального произведения, для его воспроизведения в программе;

2.3 Проектирование макетов пользовательского интерфейса

Макет пользовательского интерфейса (UI) — это визуальная схема или прототип, который демонстрирует, как будет выглядеть интерфейс программы или веб-сайта перед началом разработки. Макеты помогают разработчикам и дизайнерам согласовывать основные элементы дизайна, расположение кнопок, меню и других интерактивных элементов до начала программирования.

В контексте вашего музыкального сервиса макет пользовательского интерфейса может включать следующие ключевые элементы

*Основные страницы*

* Страница авторизации,
* Страница регистрации,
* Главная страница,
* Страница администратора;

По порядку, страница авторизации. Страница авторизации — это экран, на котором пользователь вводит свои учетные данные (обычно это электронная почта и пароль) для входа в аккаунт на сайте или в приложении. Эта страница играет важную роль в обеспечении безопасности и идентификации пользователей (Макет страницы авторизации представлен на рисунке 6).

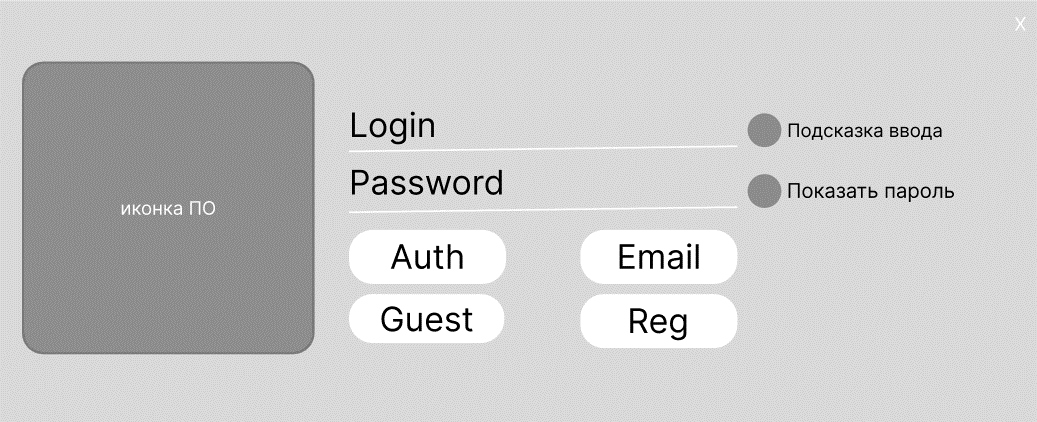


Рисунок 7 –«Макет страницы Авторизации»

Далее приступим к регистрации. Страница регистрации — это экран, на котором новый пользователь создает учетную запись на сайте или в приложении. Этот процесс включает в себя ввод персональных данных, таких как ник, электронная почта, пароль и другие необходимые сведения. Регистрация важна для предоставления персонализированных услуг и защиты конфиденциальности пользователя.

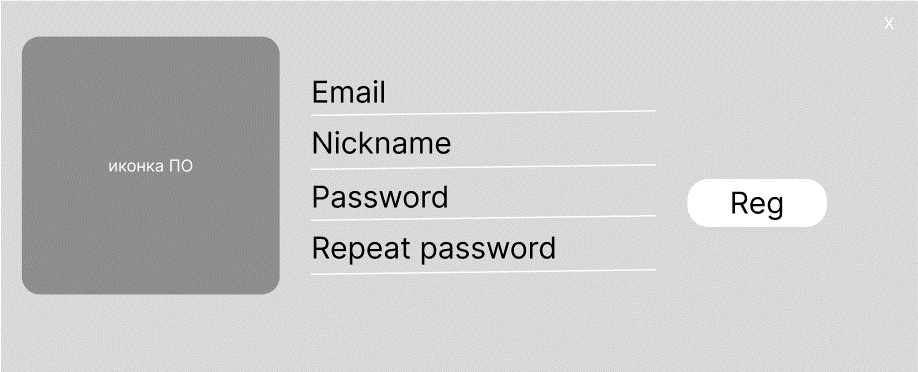


Рисунок 8 –«Макет страницы Регистрации»

Главная страница музыкального сервиса представляет собой каталог различных жанров музыки, от New Age до Классической. Пользователи могут выбирать интересующие их направления и слушать композиции онлайн.



Рисунок 9 -«Макет главной страницы»

Страница просмотра прейлистов, последняя страница в которой будут находится пользовательские плейлисты примерный дизайн продемонстрирован на рисунке 9.

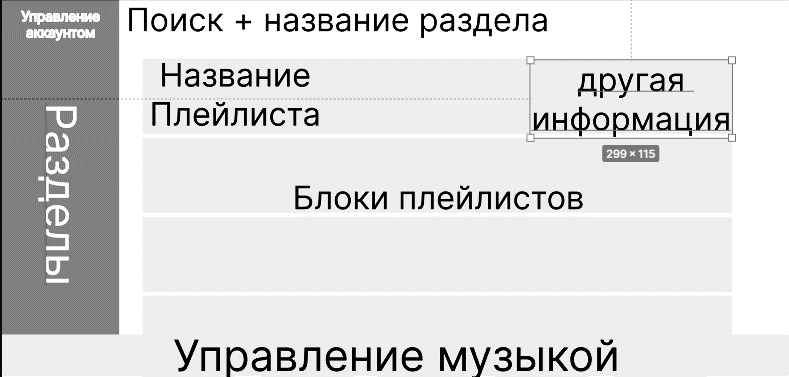


Рисунок 10 –«макет страницы просмотра плейлистов»

## 2.4 Описание стратегии тестирования и отладки информационной системы.

Для обеспечения надежности и стабильности работы музыкального стримингового сервиса будет применяться следующая стратегия тестирования и отладки:

* Модульное тестирование: Проверка отдельных компонентов системы (например, функциональности поиска, воспроизведения и плейлистов), Тестирование запросов к базе данных;
* Интеграционное тестирование: Оценка взаимодействия между различными модулями системы, Проверка корректной работы функций регистрации, авторизации и хранения данных:
* Функциональное тестирование: Проверка соответствия работы сервиса заявленным требованиям, Тестирование воспроизведения музыки, поиска и управления громкостью.
* Тестирование пользовательского интерфейса (UI/UX): Оценка удобства навигации, логики интерфейса и корректного отображения элементов, тестирование на различных устройствах и экранах;
* Регрессионное тестирование: Проверка системы после внесения изменений или исправления ошибок. Тестирование стабильности обновлений.

Эта стратегия позволит минимизировать ошибки, повысить производительность сервиса и обеспечить удобство использования.

## 2.5 Разработка тестовых сценариев (Test-cases) для входных и выходных данных.

Для обеспечения надежности и корректной работы системы необходимо разработать тестовые сценарии (Test Cases), которые будут проверять основные функции сервиса. Ниже представлены примеры тест-кейсов для входных и выходных данных. Примеры тестирований и вводных данных представлены в таблицах 1- 3.

|  |  |
| --- | --- |
| Авторизация пользователя | |
| Описание: | Проверка успешного входа в систему зарегистрированного пользователя. |
| Предусловия: | У пользователя есть учетная запись. |
| Шаги: | Ввести email, ввести пароль, нажать «Войти» |
| Ожидаемый результат: | Пользователь успешно авторизован и перенаправлен в профиль. |

Таблица 1 –«Тестирование Авторизации»

|  |  |
| --- | --- |
| Поиск трека или исполнителя | |
| Описание: | Проверка функциональности поиска. |
| Предусловия: | Пользователь авторизован. |
| Шаги: | Ввести название трека/исполнителя в строку поиска, нажать кнопку «Поиск». |
| Ожидаемый результат: | Отображается список найденных треков или исполнителей. |

Таблица 2 –«Тестирование поиска»

|  |  |
| --- | --- |
| Добавление трека в плейлист | |
| Описание: | Проверка добавления трека в пользовательский плейлист. |
| Предусловия: | У пользователя есть учетная запись, существует плейлист, есть доступные треки. |
| Шаги: | Выбрать трек, нажать «Добавить в плейлист», выбрать нужный плейлист. |
| Ожидаемый результат: | Трек успешно добавлен в выбранный плейлист. |

Таблица 3 –«Добавление трека в плейлист»

* 1. Описание методов и приемов, позволяющие защитить информацию.

При разработке музыкального стримингового сервиса необходимо обеспечить безопасность пользовательских данных, защиту от несанкционированного доступа, а также защиту контента от копирования и пиратства.

* Безопасная аутентификация;
* Использование хэширования паролей (например, алгоритм bcrypt, Argon2 или PBKDF2) для предотвращения их кражи. Защита базы данных. Ограничение доступа к данным. Применение ролевой модели доступа (RBAC) – разные уровни доступа для пользователей, администраторов, модераторов. Минимизация прав доступа к базе данных (например, пользовательский аккаунт базы данных должен иметь только необходимые привилегии). Защита от SQL-инъекций Использование подготовленных запросов (Prepared Statements) и ORM (например, SQLAlchemy, Django ORM). Валидация и экранирование входных данных;

Применение этих методов и приемов обеспечит высокий уровень безопасности музыкального стримингового сервиса, защитит личные данные пользователей и предотвратит нелегальное распространение контента.

## 3 ПРАКТИЧЕСКИЙ РАЗДЕЛ

## 3.1 Реализация приложения в соответствии с техническим заданием;

Проект разрабатывается в соответствии с установленными требованиями и включает несколько ключевых модулей, обеспечивающих его функциональность, масштабируемость и удобство использования. Каждый из модулей выполняет свою роль в общей архитектуре приложения, что способствует разделению ответственности и повышению надежности системы.

Основные модули системы:

* UI (Пользовательский интерфейс):

Этот модуль отвечает за взаимодействие с пользователем и представляет собой набор экранов и элементов управления. Основные компоненты пользовательского интерфейса включают:

* Форма авторизации – предоставляет возможность входа в систему;
* Главная страница – отображает основную информацию и предлагает пользователю навигацию по возможностям сервиса;
* Поиск – функционал, позволяющий находить музыкальные композиции, исполнителей и плейлисты по заданным критериям;
* Плейлисты – возможность создания, редактирования, удаления и просмотра персонализированных плейлистов.

Для удобства взаимодействия с системой UI будет разрабатываться с учетом современных стандартов UX/UI-дизайна, обеспечивая интуитивно понятный интерфейс и адаптивность под различные устройства.

* BLL (Бизнес-логика)

Бизнес-логика является центральной частью приложения и отвечает за обработку данных, обеспечивая корректное выполнение всех операций. Основные функции модуля BLL:

* Регистрация пользователей – обработка данных при создании нового аккаунта, валидация введенной информации, передача данных в модуль DAL.
* Авторизация – проверка учетных данных пользователя, управление сеансами и безопасностью.
* Работа с плейлистами – добавление, удаление, редактирование и управление музыкальными коллекциями пользователя.
* Формирование истории прослушивания – отслеживание активности пользователя и сохранение информации о воспроизведенных треках.

BLL взаимодействует с другими модулями, обеспечивая обработку запросов и управление данными в соответствии с бизнес-правилами приложения.

* DAL (Доступ к данным)

Модуль DAL отвечает за хранение и управление данными в базе данных, предоставляя интерфейс для безопасного доступа к информации. Основные задачи модуля включают:

* Хранение пользователей – управление учетными записями, безопасное хранение паролей, обновление пользовательских данных.
* Работа с плейлистами – сохранение данных о созданных пользователями плейлистах, их содержимом и настройках.
* Фиксация истории прослушиваний – запись информации о воспроизведенных пользователем треках, что позволяет персонализировать рекомендации.

Для обеспечения надежности и быстродействия будет использоваться современная система управления базами данных (СУБД), обеспечивающая защиту данных и высокую производительность при обработке запросов.

*Итог:*

Реализация приложения предполагает четкое разделение модулей, что обеспечит гибкость, легкость в доработке и масштабируемость системы. Взаимодействие между модулями будет осуществляться с использованием современных технологий, что позволит создать надежное, удобное и функциональное приложение, соответствующее требованиям технического задания.

# ЗАКЛЮЧЕНИЕ

БЛА бла бла бла бла

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ДОЛЖНО БЫТЬ НЕ МЕНЕЕ 30 ШТУК
2. https://proglib.io/p/samouchitel-po-python-dlya-nachinayushchih-chast-17-osnovy-skrapinga-i-parsinga-2023-03-1 (Самоучитель по Python для начинающих);
3. https://code.mu/ru/php/book/prime/basis/encoding-problem/(php и проблема с кодировками);
4. https://code.mu/ru/php/book/prime/forms/intro/ (Основы работы с формами в PHP);
5. https://code.mu/ru/php/book/prime/forms/form-submission-method/ (Метод отправки формы в PHP);
6. https://code.mu/ru/php/book/prime/forms/one-file-handling/ (Обработка формы в одном файле PHP)
7. https://code.mu/ru/php/book/prime/database/intro/ (Введение в базы данных SQL в PHP для новичков)
8. https://code.mu/ru/php/book/prime/database/preparatory-manipulations/ (Подготовительные манипуляции для работы с SQL в PHP)
9. https://code.mu/ru/php/book/prime/database/sending-queries/ (Отправка запросов к базе данных)
10. https://code.mu/ru/php/book/prime/database/encoding-problems/ (Проблемы с кодировками при работе с SQL в PHP)

# ПРИЛОЖЕНИЕ А

**Техническое задание**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. ВВЕДЕНИЕ

В разделе указывают наименование, краткую характеристику области применения программы или программного продукта или объекта, в котором предполагается их использовать. Основное назначение введения – продемонстрировать актуальность данной разработки и показать, какое место данная разработка занимает в ряду подобных.

2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

В разделе должны быть указаны: наименование документа, на основании которых ведется разработка; организация, утвердившая этот документ, и дата его утверждения; наименование и/или условное обозначение темы разработки.

Таким документом может служить план, приказ, договор и т.п.

3. НАЗНАЧЕНИЕ РАЗРАБОТКИ

В разделе должно быть указано функциональное и эксплуатационное назначение программного продукта с указанием категории пользователей.

4. ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ

Раздел должен содержать следующие подразделы:

4.1 *Требования к функциональным характеристикам*

В подразделе должны быть перечислены выполняемые функции и описаны состав, характеристики и формы представления исходных данных и результатов и указаны требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т.п.

4.2 *Требования к надежности*

В подразделе должны быть указаны требования к обеспечению надежного функционирования (обеспечение устойчивости, контроль входной и выходной информации, время восстановления после отказа и т.п.).

4.3 *Условия эксплуатации*

В подразделе должны быть указаны условия эксплуатации: температура окружающего воздуха, относительная влажность и т.п. Подобные требования формулируют, если разрабатываемая система будет эксплуатироваться в нестандартных условиях или использовать специальные внешние устройства, например, для хранения информации. Здесь же указывают вид обслуживания, необходимое количество и квалификация персонала. В противном случае допускается указывать, что требования не предъявляются.

4.4 *Требования к составу и параметрам технических средств*

В подразделе указывают необходимый состав технических средств с указанием их основных технических характеристик: тип микропроцессора, объем памяти, наличие внешних устройств и т.д. При этом часто указывают два варианта конфигурации: минимальный и рекомендуемый.

4.5 *Требования к информационной и программной совместимости*

В подразделе должны быть указаны методы решения, определены язык и среда программирования для разработки, а также операционная система и другие системные и пользовательские программные средства, с которыми должен взаимодействовать программный продукт. При необходимости указывают, какую степень защиты информации необходимо предусмотреть.

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

В разделе должны быть указаны: необходимость руководства программиста, руководства пользователя, руководства системного программиста, пояснительной записки.

6. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

В разделе устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а также, как правило, сроки разработки, и определяют исполнителей.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Этап** | **Результаты** | **Срок** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |

7. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

В разделе должны быть указаны виды испытаний и общие требования к приемке работы.

Техническое задание принято к исполнению «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ФИО

(подпись)

Руководитель КП: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ФИО

(подпись)

Руководитель ДП: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ФИО

(подпись)

# ПРИЛОЖЕНИЕ Б

Вапвапвап

# ПРИЛОЖЕНИЕ В

ЩВАЩПШВЩАШПЩЗВ