МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОУ ВПО КЫРГЫЗСКО-РОССИЙСКИЙ СЛАВЯНСКИЙ УНИВЕРСИТЕТ

имени первого Президента Российской Федерации Б.Н.Ельцина

Естественно-технический факультет

Кафедра информационных и вычислительных технологий

**ОТЧЕТ**

**по курсовой работе**

**тема: “** **Музыкальная библиотека”**

Студент: Орозмаматова Жибек

Группы: ЕПИ-2-21

Проверил: Турчанова Татьяна Георгиевна

Бишкек 2024г.

Оглавление

[**1.** **Введение** 3](#_Toc191050878)

[**Цель курсовой работы** 3](#_Toc191050879)

[**Причина выбора данной темы** 3](#_Toc191050880)

[**Основные задачи** 3](#_Toc191050881)

[**2.** **Обзор литературы** 3](#_Toc191050882)

[**Технологии и подходы, которые были изучены перед началом работы.** 3](#_Toc191050883)

[**Обоснование выбора СУБД и языка программирования** 3](#_Toc191050884)

[**3.** **Описание проекта** 4](#_Toc191050885)

[**Архитектура системы** 4](#_Toc191050886)

[**Основные компоненты** 6](#_Toc191050887)

[**Роли пользователей** 7](#_Toc191050888)

[**Функционал платформы** 8](#_Toc191050889)

[**4.** **Краткое руководство пользователя** 9](#_Toc191050890)

[**Как использовать программу** 10](#_Toc191050891)

[**5.** **Заключение** 13](#_Toc191050892)

[**6.** **Литература** 14](#_Toc191050893)

1. **Введение**

**Цель курсовой работы**

Цель данной курсовой работы – разработка много-пользовательского программного приложения "Музыкальная библиотека", позволяющего пользователям управлять своими музыкальными коллекциями, создавать плейлисты, искать и организовывать музыкальные треки.

**Причина выбора данной темы**

Музыка играет важную роль в жизни каждого человека, и современные технологии позволяют хранить, управлять и прослушивать аудиофайлы в удобной форме. Разработка музыкальной библиотеки позволит пользователям систематизировать свои музыкальные предпочтения и легко находить нужные треки.

**Основные задачи**

Основные задачи курсовой работы включают:

1. Разработка базы данных для хранения информации о треках, исполнителях, альбомах и плейлистах пользователей.
2. Реализация пользовательского интерфейса для удобного управления музыкальными коллекциями.
3. Обеспечение функционала поиска по различным критериям.
4. Реализация безопасности данных и предотвращение SQL-инъекций.
5. Проведение тестирования и оптимизация системы.
6. **Обзор литературы**

**Технологии и подходы, которые были изучены перед началом работы.**

**Java и JavaFX:** Основы языка Java и фреймворка JavaFX для создания пользовательского интерфейса.

**База данных:** Работа с базами данных, включая проектирование таблиц, написание SQL-запросов и интеграцию с Java через Hibernate

**MVC-архитектура:** Принципы разделения кода на модели, представления и контроллеры для улучшения читаемости и поддержки кода.

**Apache POI (Poor Obfuscation Implementation)** – это библиотека Java для работы с файлами Microsoft Office (Excel, Word, PowerPoint). Она позволяет читать, записывать и изменять файлы **XLS, XLSX (Excel)**, **DOC, DOCX (Word)** и другие форматы.

**Обоснование выбора СУБД и языка программирования**

Для реализации проекта был выбран язык программирования **Java** из-за его широкого распространения и поддержки библиотек для работы с базами данных. В качестве СУБД использовалась **PostgreSQL**, так как она обладает высокой надежностью и поддерживает сложные запросы, необходимые для работы с музыкальными коллекциями.

1. **Описание проекта**

**Архитектура системы**

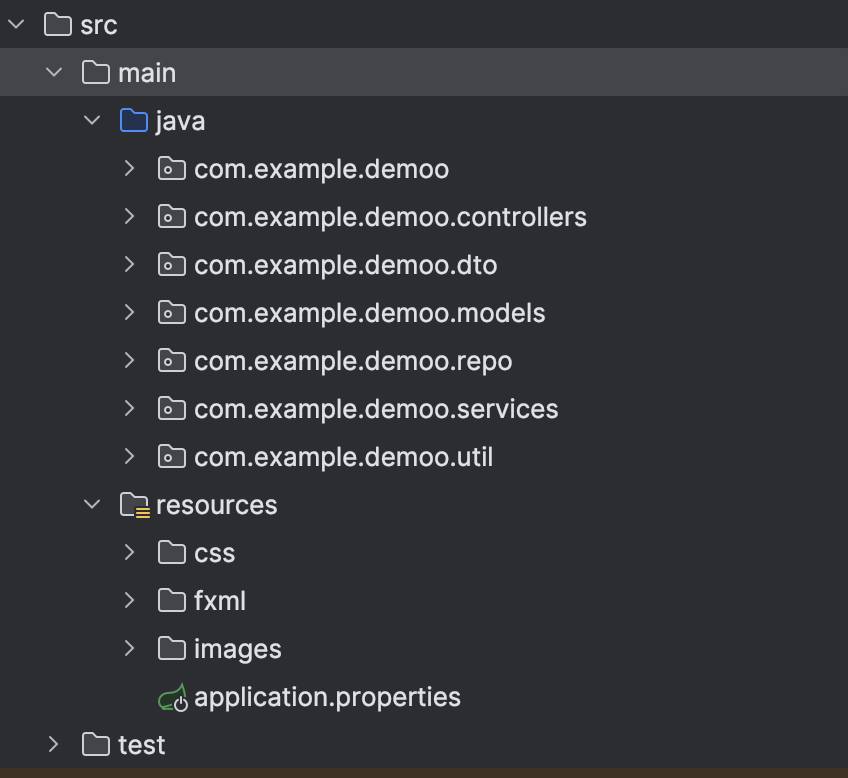


Рисунок 1. Проект

Основная цель проекта — управление объектами системы с помощью графического интерфейса и обеспечения взаимодействия с базой данных через слой сервисов и репозиториев. Проект поддерживает модель **MVC (Model-View-Controller)**, разделяя бизнес-логику, представление и управление данными для повышения гибкости и удобства сопровождения.

Архитектура приложения организована следующим образом:

1. В пакете **controllers** находятся контроллеры, обрабатывающие запросы от пользователя через JavaFX и взаимодействующие с сервисами.
2. В пакете **services**реализована бизнес-логика, выполняющая операции над объектами и взаимодействующая с базой данных через слой репозиториев.
3. В свою очередь, пакет **repo** содержит классы, использующие Spring Data JPA для работы с базой данных.
4. Пакет **models** включает сущности, представляющие структуру таблиц базы данных, а пакет **dto**используется для передачи данных между слоями приложения.
5. Дополнительно присутствует пакет **util**, в котором могут содержаться вспомогательные классы и утилиты для обработки данных.

Для хранения ресурсов используются файлы внутри **resources**, включая файлы конфигурации (**application.properties**), стили (**css**), файлы разметки интерфейса (**fxml**) и изображения (**images**). Это позволяет гибко настраивать поведение приложения без необходимости изменения кода. Подход с использованием Spring Boot упрощает тестирование, развертывание и расширение системы, обеспечивая высокую модульность и поддерживаемость.

## ***Слой представления (View):***

## Реализован на JavaFX с использованием FXML для описания интерфейса.

## Включает формы для регистрации, входа, управления треками, плейлистами, пользователями и другими сущностями.

## Контроллеры (например, LoginController, AddTrackController) обрабатывают действия пользователя и передают данные в сервисный слой.

## ***Бизнес-логика (Controller/Service):***

## Сервисные классы (пакет services) содержат основную логику: валидацию данных, управление транзакциями, взаимодействие с репозиториями.

## Примеры: создание плейлиста, добавление трека, генерация отчетов.

## Контроллеры делегируют выполнение операций сервисам, что обеспечивает разделение ответственности.

## ***Слой доступа к данным (Model/Repository):***

## Используется Spring Data JPA и Hibernate для работы с базой данных PostgreSQL.

## Репозитории (пакет repo) предоставляют CRUD-операции для сущностей: User, Track, Playlist, Genre и др.

## Сущности (пакет models) отображаются на таблицы БД через аннотации JPA (например, @Entity, @Table).

## ***Вспомогательные компоненты (Util):***

## Пакет util содержит утилиты для работы с данными, валидации, конвертации форматов (например, JSON).

***База данных:***

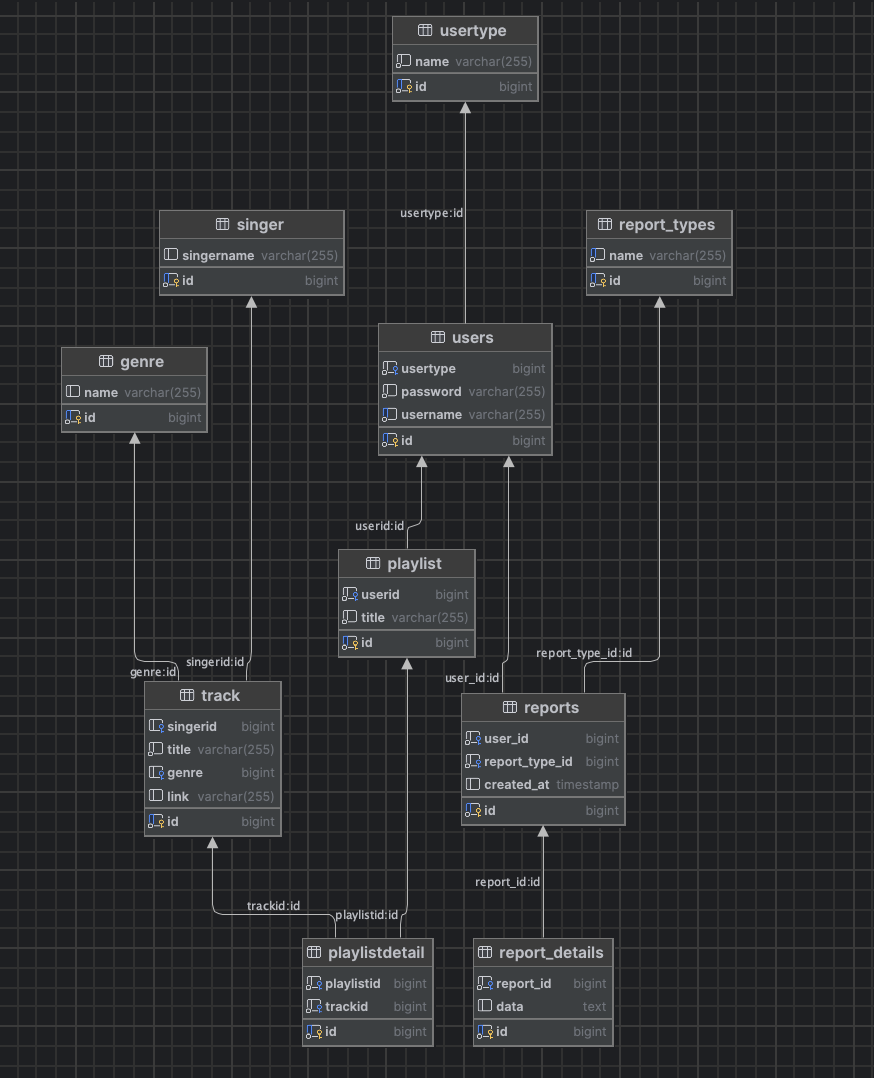


Рисунок 2. Диаграмма базы данных

**Основные компоненты:**

1. **Пользователи:** Таблица для хранения информации о пользователях
2. **Тип пользователей:** Таблица для хранения информации о типах пользователей.
3. **Исполнители:** Таблица для хранения информации об исполнителях.
4. **Треки:** Таблица для хранения информации о треках.
5. **Плейлисты:** Таблица для хранения плейлистов.
6. **Детали плейлистов:** Таблица для хранения деталей плейлистов.
7. **Жанр:** Таблица для хранения информации о жанрах.
8. **Отчеты:** Таблица для хранения отчетов
9. **Детали об отчетах:** Таблица для хранения информации о деталях отчетов.
10. **Тип отчетов:** Таблица для хранения типов отчетов.

## **Основные компоненты**

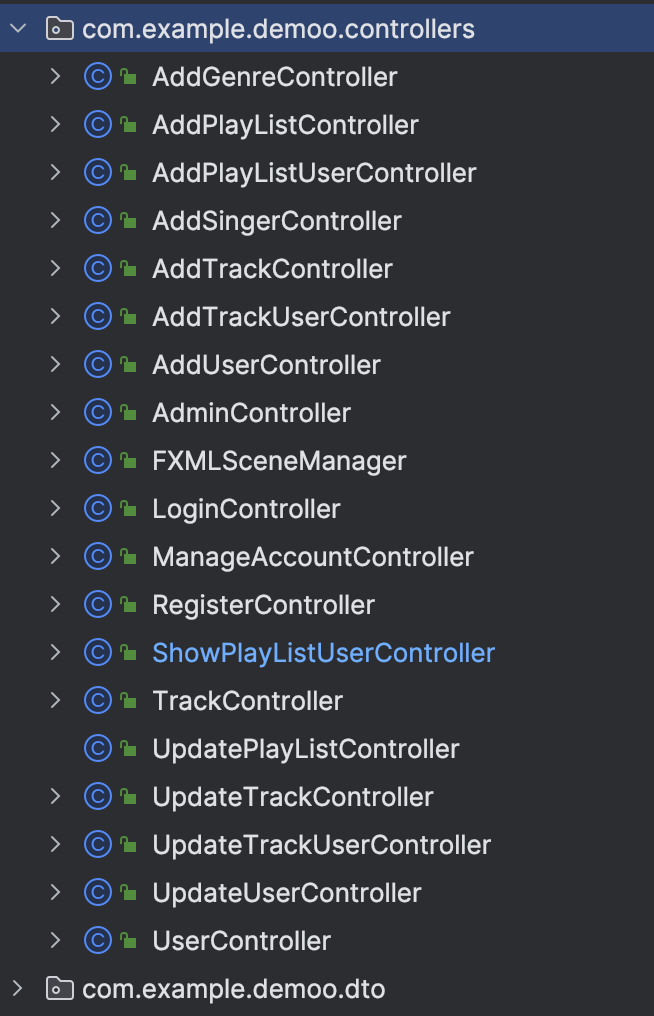


Рисунок 3. Классы-контроллеры

* AddGenreController: Добавление жанров
* AddPlayListController: Создание плейлиста для админа
* LoginController: Аутентификация и вход в систему
* AddPlayListUserController: Создание плейлиста для пользователя
* AddSingerController: Создание новых исполнителей
* AddTrackController: Создание новый треков для админа
* AddTrackUserController: Создание новый треков для пользователя
* RegisterController: Регистрация новых пользователей
* AddUserController: Добавление новых пользователей для админа
* AdminController: Контроллер для главной страницы админа
* FXMLSceneManagerController Управление переходами между сценами
* ManageAccountController: Управление аккаунтами пользователей
* ShowPlayListUserController: Управление плейлистами пользователей
* TrackController: Контроллер для управления треками
* UpdatePlayListController: Обновление существующих плейлистов
* UpdateTrackController: Управление треками для админа
* UpdateTrackUserController: Управление треками для пользователя
* UpdateUserController: Управление аккаунтами пользователей
* UserController: Контроллер для главной страницы пользователя

## **Роли пользователей**

В музыкальной библиотеке предусмотрены две основные роли пользователей: администратор (Admin) и обычный пользователь (User). Эти роли имеют различия в правах доступа и функциях.

*Роль админа (Admin):*

Права доступа:

* Имеет полный доступ ко всем данным на платформе.
* Может создавать, изменять, удалять и сохранять треки, плейлисты, а также управлять исполнителями и жанрами.
* Управляет пользователями: может создавать новые учетные записи, назначать роли, редактировать и удалять пользователей.
* Создание отчета для просмотра плейлистов и треков пользователей
* Имеет доступ ко всем функциям платформы без ограничений.

Функции:

* Администрирование контента: добавление и обновление информации о треках, плейлистах, исполнителях и жанрах.
* Управление пользователями: создание и назначение ролей, управление учетными записями пользователей.
* Управление плейлистами: имеет возможность управлять всеми плейлистами пользователей.

*Роль обычного пользователя (User):*

Права доступа:

* Имеет ограниченный доступ, сосредоточенный на функциях просмотра и добавления треков и плейлистов для своего аккаунта
* Может прослушивать треки
* Может создавать и управлять собственными плейлистами, но не имеет доступа к плейлистам других пользователей.

Функции:

* Просмотр и прослушивание треков
* Управление плейлистами: создание, редактирование и удаление своих плейлистов для организации треков по своему усмотрению.

Различия:

Админ имеет полный контроль над всей платформой, включая управление пользователями и модерацию контента.

Обычный пользователь ограничен в своих действиях и сосредоточен на потреблении контента и управлении собственными плейлистами.

## **Функционал платформы**

Функционал музыкальной библиотеки включает в себя несколько ключевых возможностей, которые предоставляются пользователям в зависимости от их роли:

1. Прослушивание треков

Обычные пользователи и админы могут просматривать список фильмов, а также прослушивать треки.

2. Управление плейлистами

Обычные пользователи могут создавать и редактировать собственные плейлисты, добавлять или удалять треки из своих плейлистов.

Админы имеют доступ ко всем плейлистам пользователей и могут их редактировать или удалять по необходимости.

3. Управление треками

Админы могут добавлять, редактировать и удалять треки, а также управлять данными об исполнителях и жанрах.

4. Управление пользователями

Админы могут создавать новые учетные записи пользователей, назначать роли (например, админ или обычный пользователь), редактировать профили пользователей и удалять учетные записи.

Этот функционал обеспечивает пользователям удобное и интуитивное взаимодействие с платформой, предоставляя различные уровни доступа в зависимости от их роли.

**Краткое руководство пользователя**

Программа предназначена для работы музыкальной библиотеки, где пользователи могут прослушивать треки и создавать плейлисты. Платформа поддерживает две роли: администратор и обычный пользователь. Администраторы имеют полный контроль над данными, включая возможность удаления, обновления и сохранения информации. Обычные пользователи могут взаимодействовать с треками, создавая свои персонализированные коллекции треков.

**Пользовательский интерфейс**

Приложение имеет интуитивно понятный интерфейс, состоящий из следующих элементов:

Главное окно: Отображает основные функции системы, такие как управление треками, плейлистами и аккаунтами

Окно управления плейлистами: Позволяет просматривать и редактировать информацию о плейлистах.

Окно управления всеми пользователями системы: Позволяет просматривать и редактировать информацию о всех пользователях системы.

Окно управления треками: Позволяет управлять данными о треках.

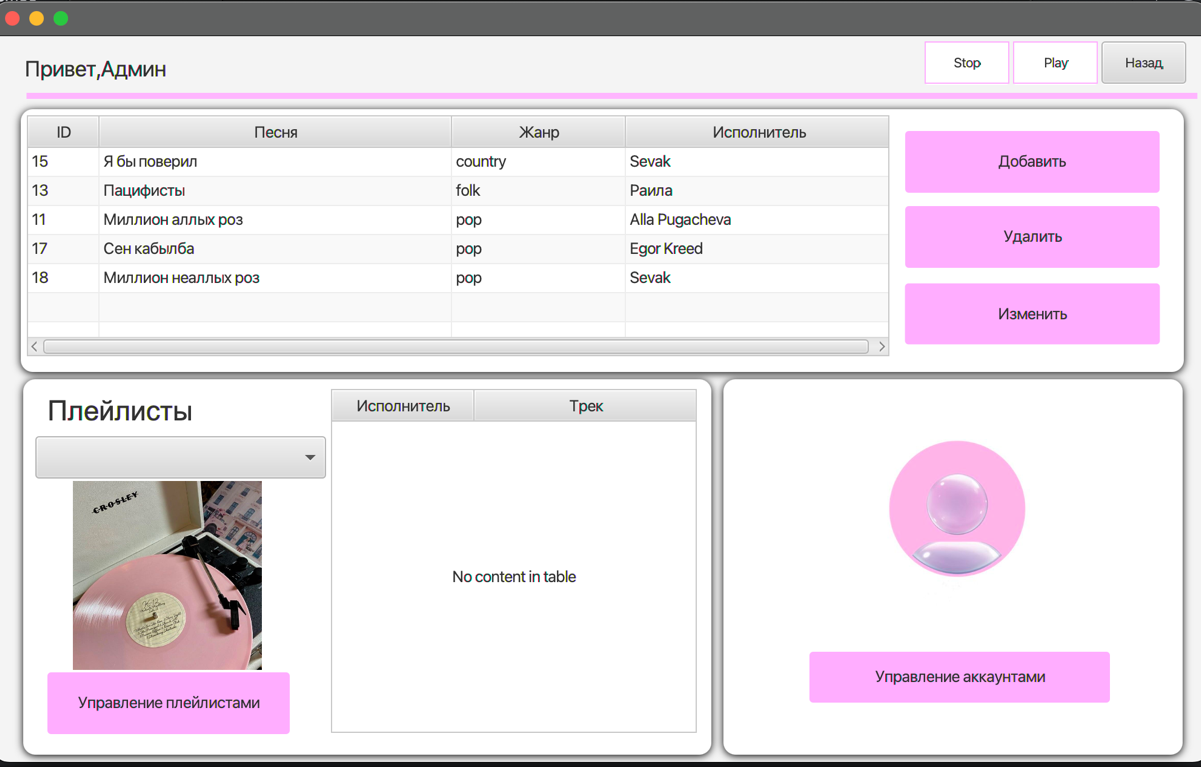


Рисунок 4. Главное окно админа

**Как использовать программу**

*Регистрация и вход в систему:* Новый пользователь должен зарегистрироваться, указав свои данные, после чего ему будет доступен вход в систему.

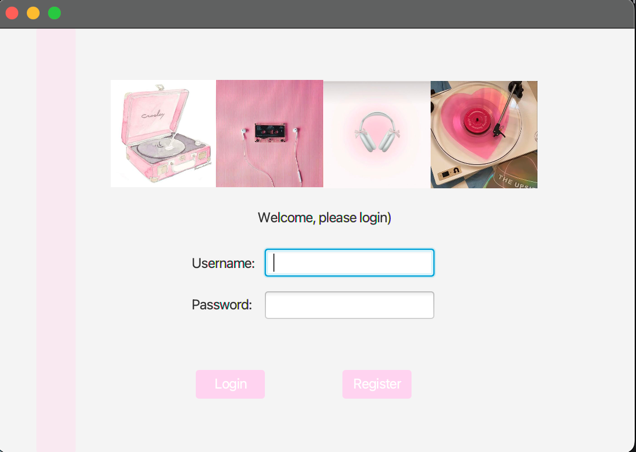
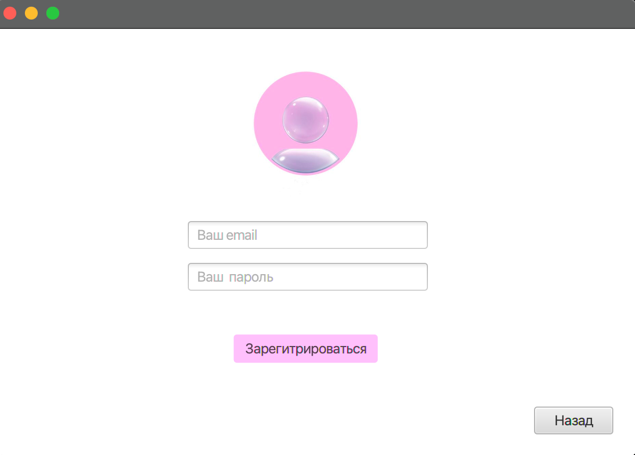
 

Рисунок 5. Окно входа в систему Рисунок 6. Окно регистрации нового пользователя в систему

*Создание плейлиста:* На вкладке "Управление плейлистами" пользователь может создать новый плейлист, затем искать треки и добавлять их в плейлист, удалять плейлисты и треки

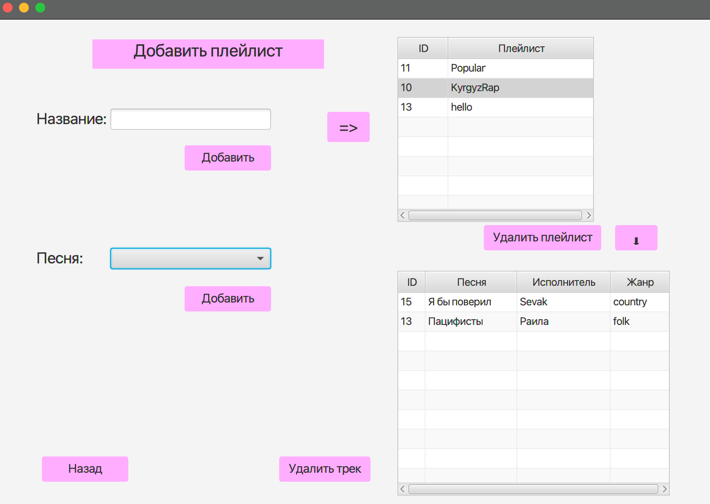


Рисунок 7. Окно управления плейлистами

*Прослушивание и управление треками:* В главном окне админа можно выбрать трек для прослушивания трека. Здесь также можно найти информацию о треке: название песни, жанр, исполнителя. Во вкладке “Play” можно послушать трек. Здесь же доступны кнопки “Добавить” ,“Изменить” – они предназначены для открытия новых окон добавления и изменения треков соответственно. Также доступна кнопка “Удалить”– она предназначена для удаления трека.

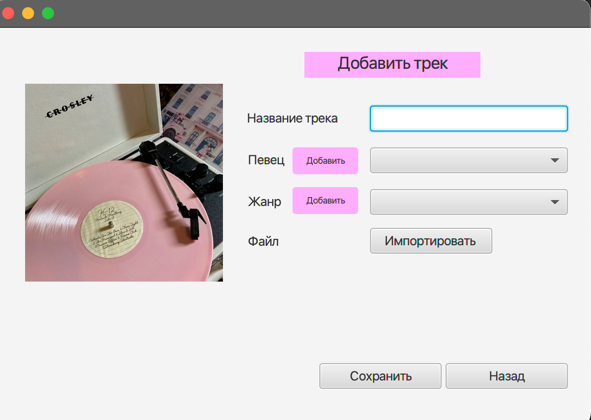
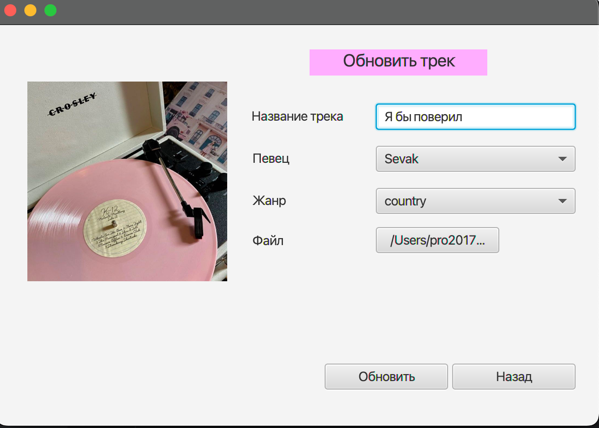
 

Рисунок 8. Окно добавления трека Рисунок 9. Окно обновления трека

В окне добавления трека можно также добавить новых исполнителей и жанров.

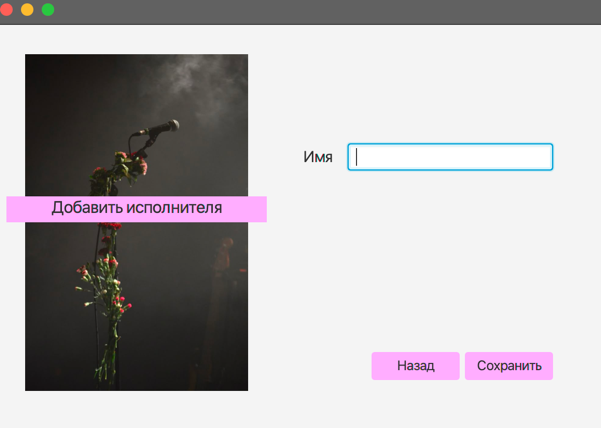
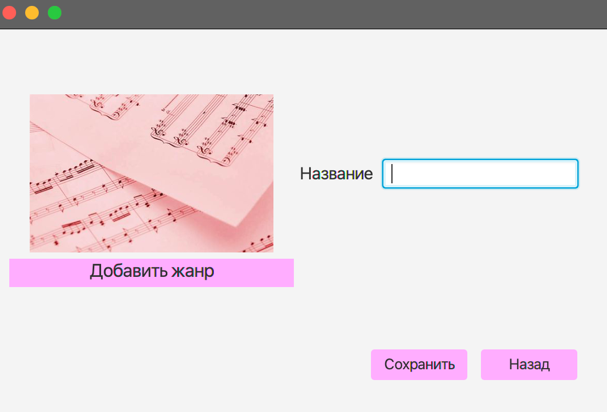
 

Рисунок 10. Окно добавления исполнителя Рисунок 11. Окно добавления жанра

*Управление пользователями (для администраторов):* через панель администратора можно добавлять, удалять или редактировать информацию о пользователях. К примеру, так выглядит окно редактирования информации о пользователях:

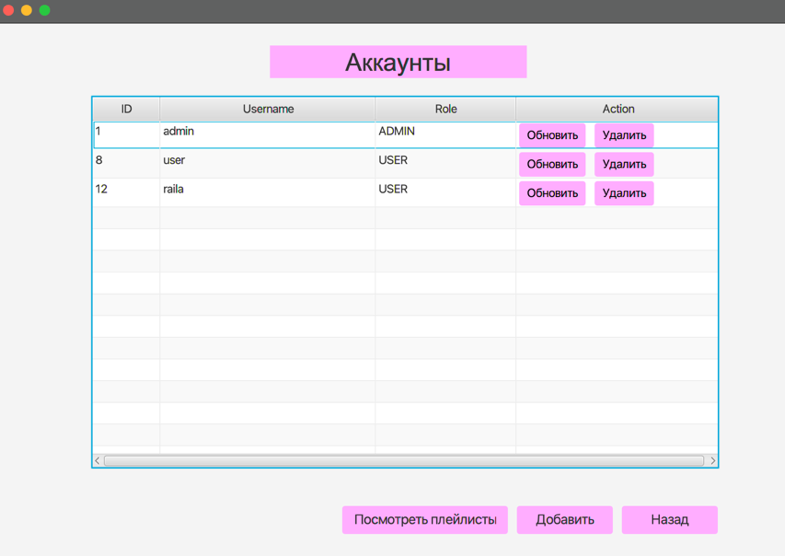


Рисунок 12. Окно управления всеми пользователями системы для админа

В окне управления пользователями можно также добавить новых исполнителей, изменить данные существующих пользователей, просмотреть плейлисты и сгенерировать отчет о существующих плейлистах и треках в них конкретного пользователя. Отчеты сформулированы с помощью технологии **Apache POI (Poor Obfuscation Implementation)**, библиотека которая реализует формат хранения данных Microsoft Office, предоставляя удобные API для работы с таблицами, текстами, графиками и форматированием.

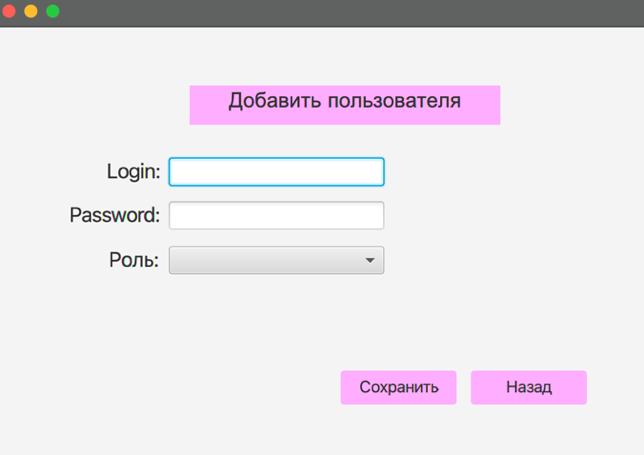
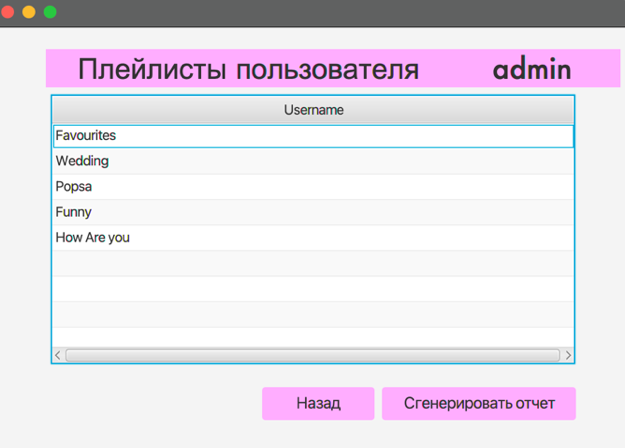
 

Рисунок 13. Окно добавления нового пользователя Рисунок 14. Окно отображения плейлистов конкретного пользователя

*Интерфейс и функционал для роли пользователь:* В главном окне пользователя можно увидеть такие вкладки как “Треки”, “Артисты” , “Плейлисты” , “Выйти”. По нажатию в каждую из них отображаются данные из соответствующих таблиц, а также включен функционал управления то есть : добавления , удаления и изменения. Во вкладках “Треки” и “Артисты” добавлен поисковик для поиска по конкретному критерию.

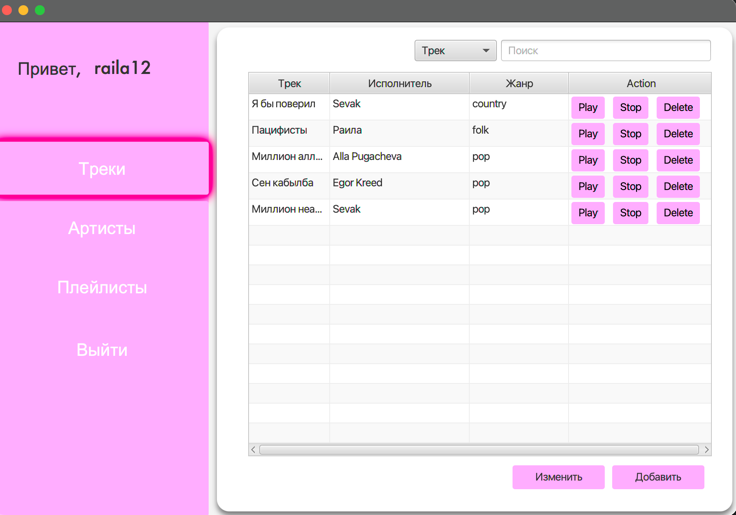
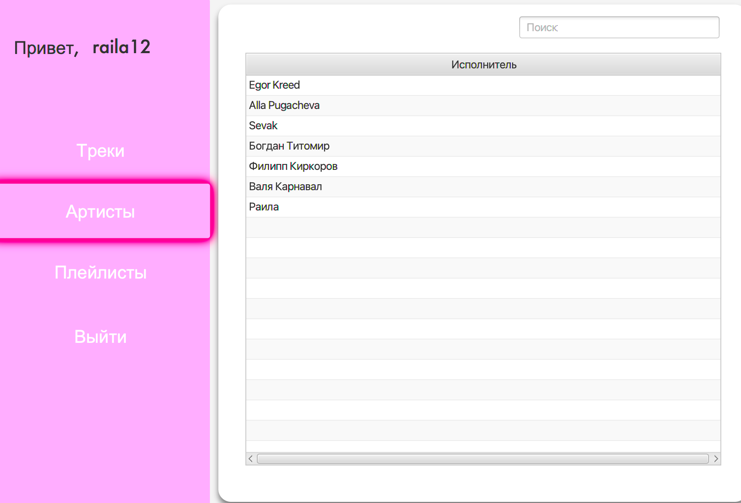
 

Рисунок 15. Окно отображения всех доступных треков Рисунок 16. Окно отображения всех доступных исполнителей

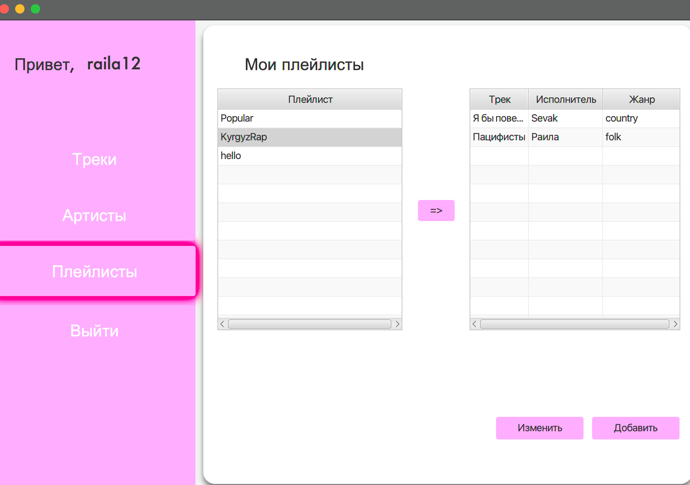


Рисунок 17. Окно отображение плейлистов и треков в них

1. **Заключение**

Разработанное приложение «Музыкальная библиотека» успешно решает поставленные задачи по организации музыкальных коллекций, управлению плейлистами и обеспечению безопасности данных. В ходе работы была создана многослойная архитектура на основе паттерна MVC, что позволило четко разделить логику, представление и данные. Использование JavaFX и FXML обеспечило интуитивно понятный интерфейс, а интеграция с PostgreSQL через Hibernate и Spring Data JPA гарантировала надежное хранение и обработку информации. А внедрение Apache POI (Poor Obfuscation Implementation) позволило читать, записывать, редактировать файлы Word (DOC, DOCX.

Ключевые достижения:

* Реализована система ролей (администратор и пользователь) с разграничением прав доступа.
* Внедрены функции поиска треков по жанрам, исполнителям и названиям для всех типов пользователей.
* Обеспечена защита от SQL-инъекций за счет параметризованных запросов и ORM-подхода.
* Создан механизм генерации отчетов, позволяющий администраторам анализировать активность пользователей.

Приложение демонстрирует потенциал Java-экосистемы для создания сложных мультимедийных систем и может служить основой для дальнейших исследований в области управления цифровым контентом.

1. **Литература**
2. JavaFX Documentation – <https://openjfx.io> - официальная документация по JavaFX: руководство по созданию интерфейсов, работе с FXML и анимацией.
3. Spring Boot Reference Guide – <https://spring.io/projects/spring-boot> - описание настройки Spring Boot, работы с JPA, REST API и security.
4. PostgreSQL Documentation – <https://www.postgresql.org/docs> - подробное руководство по SQL-синтаксису, оптимизации запросов и администрированию БД.
5. MVC Pattern in Java: <https://www.baeldung.com/java-mvc-pattern> - статья о реализации паттерна MVC в Java-приложениях с примерами кода.
6. JavaFX GUI Development: <https://www.tutorialspoint.com/javafx/index.htm> - учебные материалы по разработке графических интерфейсов на JavaFX для начинающих.