

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (ФГБОУ ВО
«ВГУ»)

Факультет компьютерных наук
Кафедра программирования и информационных технологий

Отчет по курсу «Технологии программирования»

**Каталог музыкальных произведений, записанных в электронной форме.
Рекомендации для конкретного пользователя на основе его вкусов.**

Заведующий кафедрой:

_____ / С. Д. Махортов, д. ф.-м. н., профессор

Заказчики:

_____ / Тарасов В. С.

_____ / Клейменов И. В.

Исполнители:

_____ / В. Н. Яцкова

_____ / А. Г. Сапелкин

Воронеж, 2022 г.

Содержание

Введение	4
1. Анализ предметной области	6
1.1. Терминология	6
1.2. Обзор аналогов	7
1.2.1. Spotify	8
1.2.2. VK Music	9
1.2.3. Deezer	9
1.3. Постановка задачи	10
1.3.1. Функциональные требования	10
1.3.2. Технические требования	11
1.3.3. Требования к интерфейсу	12
1.3.4. Макет интерфейса	12
1.4. Средства реализации	15
2. Реализация	18
2.1. Архитектура приложения	18
2.1.1. База данных	18
2.1.2. Варианты использования	21
2.1.3. Диаграммы классов	21
2.1.4. Диаграммы последовательностей	23
2.1.5. Воронки конверсии	30
2.1.6. IDEF-0 диаграмма	31
2.1.7. Диаграмма развертывания	31

2.1.8. Диаграмма состояний	32
2.2. Серверная часть.....	32
2.2.1. Основные положения	33
2.2.2. Реализация рекомендаций по предпочтениям	33
2.2.3. Развертка.....	34
2.2.4. Документация API	34
2.3. Клиентская часть.....	34
2.3.1. Основные положения	34
Заключение.....	36
Список использованных источников	37

Введение

С развитием компьютерных технологий оцифровке подвергаются многие события и процессы. Музыканты набирают популярность с помощью распространения своих произведений в сети интернет посредством рекламы, а уже известные исполнители получают прибыль продавая авторские права различным сервисам для прослушивания музыкальных треков.

В современном мире большинство музыкантов используют для написания своих треков различные приложения, в которых собраны звуки всевозможных музыкальных инструментов. Это значительно упрощает процесс сочинения, ведь совсем не обязательно иметь под рукой массивные барабанные установки или уметь играть на скрипке, достаточно просто установить подходящее приложение и разобраться с сочетаниями клавиш на клавиатуре.

Советский музыковед и музыкальный социолог Арнольд Наумович Сохор однажды сказал, что музыка, которая, кстати, с греческого переводится как «муза», «отражает действительность и воздействует на человека посредством осмысленных и особым образом организованных по высоте и во времени звуковых последований, состоящих в основном из тонов».

В качестве распорядителей информации в настоящее время все чаще выступают базы данных. Они обеспечивают безопасное хранение структурированных данных и доступ к ним почти в любое время[1]. Любой современный проект, предусматривающий большое количество пользователей, нуждается в базе данных, которая отвечает требованиям проекта, а также удовлетворяет потребности по хранению, управлению и администрированию данных.

С развитием технологий люди получили возможность наслаждаться любимой музыкой в любое удобное время с помощью различных сервисов. Современные системы позволяют не только прослушивать уже известные композиции, но и находить новые треки и новых исполнителей посредством рекомендаций, основанных на плейлисте людей со схожими музыкальными предпочтениями.

Создание единого музыкального пространства позволяет пользователям находить новую для себя музыку и прослушивать уже известные композиции. А для авторов это хороший инструмент визуализации своей музыкальной востребованности, так как, например, по количеству прослушиваний легко можно определить, насколько популярно произведение.

Для сбора и подбора новых треков, авторов, жанров, отслеживания предпочтений пользователей необходима база данных, которая будет содержать эту информацию.

Целью данной работы являлась разработка приложения, предназначенного для подбора и рекомендации новых произведений на основе предпочтений пользователя, которые он также может задать. Также данное приложение может использоваться для самостоятельного поиска треков в случае необходимости. Это позволит пользователям расширять свой музыкальный кругозор и открывать для себя новых исполнителей, новые жанры и новые композиции.

1. Анализ предметной области

1.1. Терминология

Термин	Определение
Интерфейс	Набор инструментов, позволяющий пользователю взаимодействовать с системой.
Чарт	Список самых популярных, на данный момент, песен.
Подкаст	Трансляция музыки или речевого контента в интернете по принципу тематической или жанровой радиостанции.
Авторизация	Предоставление определенному лицу прав на выполнение определенных действий.
Сервер	Обслуживающее устройство в системах автоматической обработки информации.
Модуль	Сложный узел, выполняющий самостоятельную функцию в техническом устройстве, а также вообще отделяемая, относительно самостоятельная часть какой-н. системы, организации.
Валидация	Процесс проверки данных различных типов по критериям корректности и полезности для конкретного применения.

Отладка	Этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.
Рендеринг	Процесс получения изображения по модели с помощью компьютерной программы.
Внешний ключ	Столбец (или группа столбцов), используемый в реляционной базе данных для связи данных между таблицами.
Главный ключ	Минимальный набор атрибутов, по значениям которых можно однозначно выбрать требуемый экземпляр сущности.
Альтернативный ключ	Атрибут (или группа атрибутов), несовпадающий с первичным ключом и уникально идентифицирующий экземпляр сущности.
Составной ключ	Комбинация двух или более столбцов в таблице, которые можно использовать для уникальной идентификации каждой строки в таблице.
Метрика	Инструмент веб-аналитики, который помогает получать наглядные отчеты, записи действий посетителей, отслеживать источники трафика.

1.2. Обзор аналогов

Начиная разработку нового музыкального приложения, стоит ознакомиться с уже существующими и рассмотреть их достоинства и недостатки. На основании полученной информации можно сделать выводы о том, какие недочеты стоит взять на вооружение и не допускать в своей работе, чтобы интерфейс для пользователя был удобен, прост в использовании и управлении.

1.2.1. Spotify

Возможности приложения:

- прослушивание любых выбранных и доступных произведений;
- получение личных рекомендаций на основе избранных или уже прослушанных произведений;
- доступ к скачанным композициям в режиме offline;
- доступ к различным чартам и хит-парадам;
- доступ к радиостанциям;

Достоинства интерфейса:

- простой интерфейс;
- доступ с различных платформ;
- удобно в использовании;
- легко делиться музыкой с друзьями;
- большая база музыкальных произведений.

Недостатки:

- обязательная регистрация для доступа к функционалу приложения;
- ограниченные возможности бесплатной версии;
- отсутствуют тексты песен;
- часто всплывающая реклама.

В итоге, можно сказать, что это удобное и интуитивно понятное приложение с широкими возможностями, которое, однако, требует регистрации для доступа к функционалу, а в бесплатной версии ухудшает впечатления пользователя при использовании приложения.

1.2.2. VK Music

Возможности приложения:

- прослушивание любых доступных композиций;
- просмотр и воспроизведение музыки, которая нравится пользователям в списке друзей;
- просмотр текста некоторых композиций;
- доступ к скачанным трекам в режиме offline;
- доступ к редактированию текста композиции при выявлении ошибок;
- рекомендации на основе сохраненных треков;
- доступ к различным подборкам, чартам, плейлистам, сборникам.

Достоинства интерфейса:

- регулярное добавление новинок и обновление чартов;
- наличие возможности создания плейлистов;
- легко делиться музыкой с друзьями.

Недостатки:

- огромное количество повторяющихся композиций;
- часто всплывающая реклама;
- отсутствует возможность сортировки.

Таким образом, приложение обладает приятным интерфейсом и не менее приятным функционалом, однако композиции не отсортированы, а реклама портит общее впечатление о приложении.

1.2.3. Deezer

Возможности приложения:

- воспроизведение любых находящихся в библиотеке треков;
- воспроизведение сохраненных композиций в режиме offline;
- формирование своего плейлиста;
- получение рекомендаций, на основе избранных композиций;
- доступ к составленным плейлистам других пользователей;
- поиск нужных композиций;

- прослушивание интересующих подкастов;
 - просмотр текстов песен.
- Теперь рассмотрим достоинства интерфейса приложения:
- большая музыкальная библиотека;
 - наличие подкастов;
 - рекомендации не только по трекам, но и по исполнителям, и по плейлистам;
 - расширенный поиск, включающий не только поиск по автору и названию, но и по тексту композиции.

Недостатки:

- серьезные ограничения для бесплатной версии;
- обязательная регистрация/авторизация для использования;
- отсутствие возможности редактирования тестов песен;
- отсутствие чартов и хит-парадов, а также радиостанций.

Делая вывод, можно сказать, что у приложения приятный и интуитивный интерфейс, широкие возможности, в том числе рекомендации по разным критериям, однако все это значительно ограничено в бесплатной версии.

1.3. Постановка задачи

Целью данного проекта является разработка Android приложения «Музкат», которое предназначено для составления индивидуального списка музыкальных предпочтений для каждого пользователя, на основании которого можно получить рекомендации музыкальных произведений, что поможет пользователям этого приложения расширить свой кругозор в музыке, найти новые музыкальные произведения.

1.3.1. Функциональные требования

Для определения границ проекта и его основных задач были сформулированы следующие функциональные требования:

- возможность входа в аккаунт и регистрации неавторизованного пользователя с использованием логина и пароля;

- возможность выхода из аккаунта авторизованного пользователя;
- возможность просмотра информации о музыкальных записях: название, автор и жанр;
- возможность задания авторизованным пользователем предпочитаемых жанров и авторов;
- возможность авторизованного пользователя пополнять список доступных музыкальных записей, состоящих из названия произведения, автора и жанра.
- возможность авторизованного пользователя получения списка музыкальных произведений, с учетом заданных предпочтений.

1.3.2. Технические требования

Не менее важным также является и определение технических требований:

- все запросы клиента серверу и ответы сервера клиенту осуществляются по протоколу HTTP, если явно не указано иное;
- вход в аккаунт или регистрация осуществляется посредством отправки клиента серверу запроса о регистрации или входе с включенными в тело запроса введенного пользователем логина и пароля, где сервер проверяет и одобряет или отклоняет вход или регистрацию, отправляя соответствующие ответы пользователю;
- для хранения данных должна использоваться реляционная база данных;
- приложение должно быть реализовано под Android версии 6 или старше, с использованием языка Java;
- при добавлении новой информации о музыкальных записях или изменениях предпочтений авторизованного пользователя производится отсылка серверу запроса, содержащего всю необходимую информацию для изменения предпочтений или списка музыки;
- ведение статистики использования приложения в соответствии с описанными в следующей главе воронками конверсии.

1.3.3. Требования к интерфейсу

Для ясности намеченных задач и целей были выделены следующие требования к интерфейсу:

- возможность пропуска входа или регистрации неавторизированного пользователя с дальнейшим переходом к списку информации о музыкальных произведениях;
- возможность просмотреть причину неудачного входа в аккаунт или его регистрацию при, соответственно, попытке входа или регистрации;
- при выходе из аккаунта интерфейс обновляется в соответствии с функциональными требованиями, а список музыкальных произведений сбрасывается на клиентской части с последующим запросом нового списка у сервера.

1.3.4. Макет интерфейса

В качестве предлагаемого для реализации интерфейса приложения был разработан макет экранов приложения, однако реализация может отличаться по усмотрению исполнителя, но должна соответствовать ранее описанным требованиям.

При входе в приложение пользователь видит экран личного кабинета для авторизации, для которой нужно ввести логин и пароль, нажать «Login» или «Lagon» для входа или регистрации соответственно.

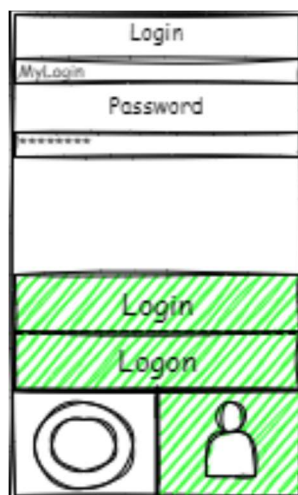


Рисунок 1 — Начальный экран – вход или регистрация

При неудачной попытке входа или регистрации пользователь будет уведомлен об этом с указанной причиной провала.

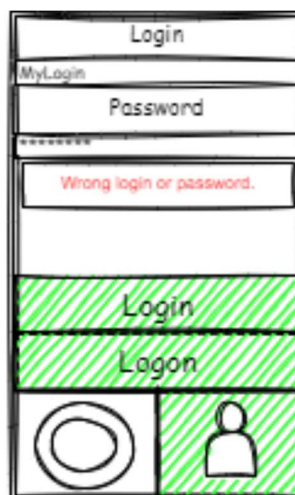


Рисунок 2 — Попытка входа провалена

Не выполнив авторизацию, пользователь может перейти к списку информации о музыкальных произведениях сразу, нажав на левую нижнюю кнопку с изображением диска.

Здесь каждая запись отображает в себе автора, название произведения и жанр.

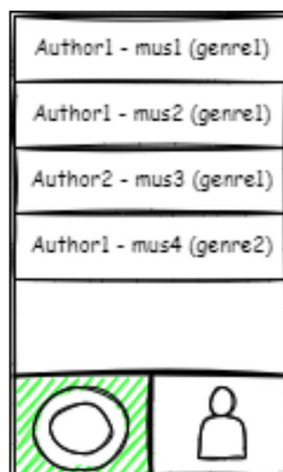


Рисунок 3 — Неавторизованный пользователь пропустил авторизацию и перешел к списку информации о музыкальных произведениях.

Однако, если пользователь авторизовался, экран личного кабинета меняется – появляются возможности выхода из аккаунта (кнопка Logout), добавления (кнопка «+») или удаления (кнопка «-») предпочтений по авторам и жанрам.

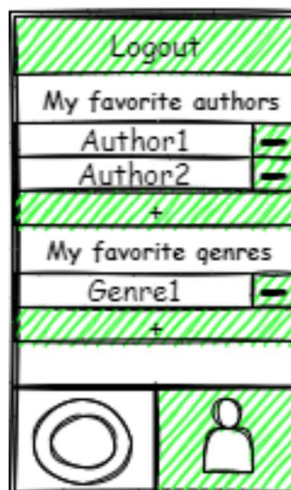


Рисунок 4 — Личный кабинет авторизованного пользователя.

При добавлении новой записи в список своих предпочтений пользователю необходимо нажать «+», ввести предпочтение и нажать кнопку «Ок». Для краткости макета добавление автора и жанра объединены, а название соответствующей функции помечено как [DataTypeName].



Рисунок 5 — Экран добавления новой записи в список предпочтений авторизованного пользователя.

Экран списка информации о музыкальных произведениях при авторизации также меняется: появляется возможность добавления новых записей, а также все записи теперь отсортированы по соответствию предпочтений.

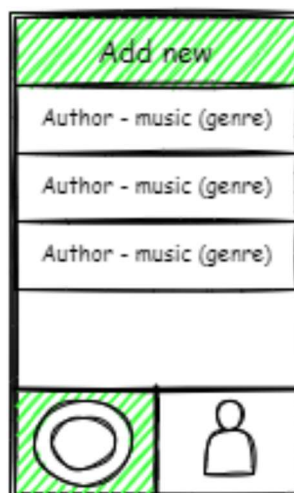


Рисунок 6 — Экран списка музыкальных произведений для авторизованного пользователя.

При добавлении новой записи (кнопка «Add new») авторизованному пользователю нужно ввести название записи, ее автора и жанр, после чего нажать кнопку «Ок».

1.4. Средства реализации

В качестве средств реализации для разработки мобильного приложения использованы:

- Java 11 (клиент и сервер)
- Spring Framework (сервер)
- Spring Boot (сервер)
- Hibernate (сервер)
- Java Persistence API (сервер)
- Lombok (сервер)
- XML при создании экранов приложения и стилей (клиент)
- IntelliJ IDEA Community Edition (сервер)
- Android Studio (клиент)

Перед очевидным аналогом – Kotlin – Java как язык программирования для android-приложений обладает преимуществом в размере сообщества разработчиков и информации по разработке приложений на этом языке, так как является более старым, что позволяет быстрее реализовывать решения и находить проблемы. Одиннадцатая версия была выбрана в связи с тем, что

обладает долгосрочной поддержкой, и, в отличие от 8 или 17 версии, не является слишком старой или слишком новой, что могло бы привести к потере долгосрочной поддержки по мере существования приложения или к ошибкам со стороны языка по причине новизны версии.

Языком для сервера была выбрана Java в связи с использованием Spring Framework[2], который обладает следующими преимуществами:

- Spring Framework может быть задействован на всех архитектурных слоях, применяемых при разработке клиент-серверных приложений;
- использует модель POJO при написании классов;
- позволяет свободно связывать модули и легко их тестировать;
- поддерживает декларативное программирование;
- избавляет от самостоятельного создания фабричных и синглтон-классов;
- поддерживает различные способы конфигурации;
- предоставляет сервис уровня middleware.

Spring Boot[3] обладает следующими преимуществами:

- не требует развертывания WAR-файлов;
- создает автономные приложения;
- помогает напрямую встроить в приложение Tomcat, Jetty или Undertow;
- не требует XML-конфигурации;
- направлен на уменьшение объема исходного кода;
- имеет дополнительную функциональность «из коробки»;
- простота запуска;
- простая настройка и управление.

Java Persistence API[4] обладает следующими преимуществами:

- отделяет SQL логику от бизнес-логики, что позволяет думать более абстрактно и ориентироваться на свои классы, а это, в свою очередь, также облегчает сопровождение;
- нет необходимости контролировать количество «?» в SQL-подобных запросах;

- есть совместимость с библиотеками валидации данных.

Главное преимущество Lombok[5] в том, что он значительно сокращает код, делает его более читаемым при помощи аннотаций, добавляя соответствующие им методы, контролируя уровни доступа. Это снимает рутинную работу с разработчика и ускоряет создание приложения.

Преимущество XML-разметки для создания экранов приложения перед ручным созданием элементов для экранов:

- легкая портируемость и мобильность без особой привязки к какому-либо коду;
- более краткая и удобная настройка элементов за счёт тегов и атрибутов;

IntelliJ IDEA Community Edition[6] была выбрана как среда разработки, так как обладает следующими преимуществами:

- умное автодополнение кода;
- качественная и удобная отладка;
- поддержка системы контроля версий;
- интуитивный интерфейс;
- есть инструменты для безопасного рефакторинга и автогенерации кода.

И, наконец, Android Studio[7] обладает такими преимуществами:

- удобный дизайнер пользовательских интерфейсов, позволяющий облегчить визуальное проектирование приложения;
- XML редактор с подсветкой синтаксиса и умным автодополнением;
- поддержка системы контроля версий;
- поддержка эмуляции устройств;
- возможность проводить тестирование и анализ кода;
- высокая скорость сборки приложения;
- поддержка рендера средствами GPU.

2. Реализация

2.1. Архитектура приложения

2.1.1. База данных

В качестве базы данных используется реляционная на основе СУБД [8].

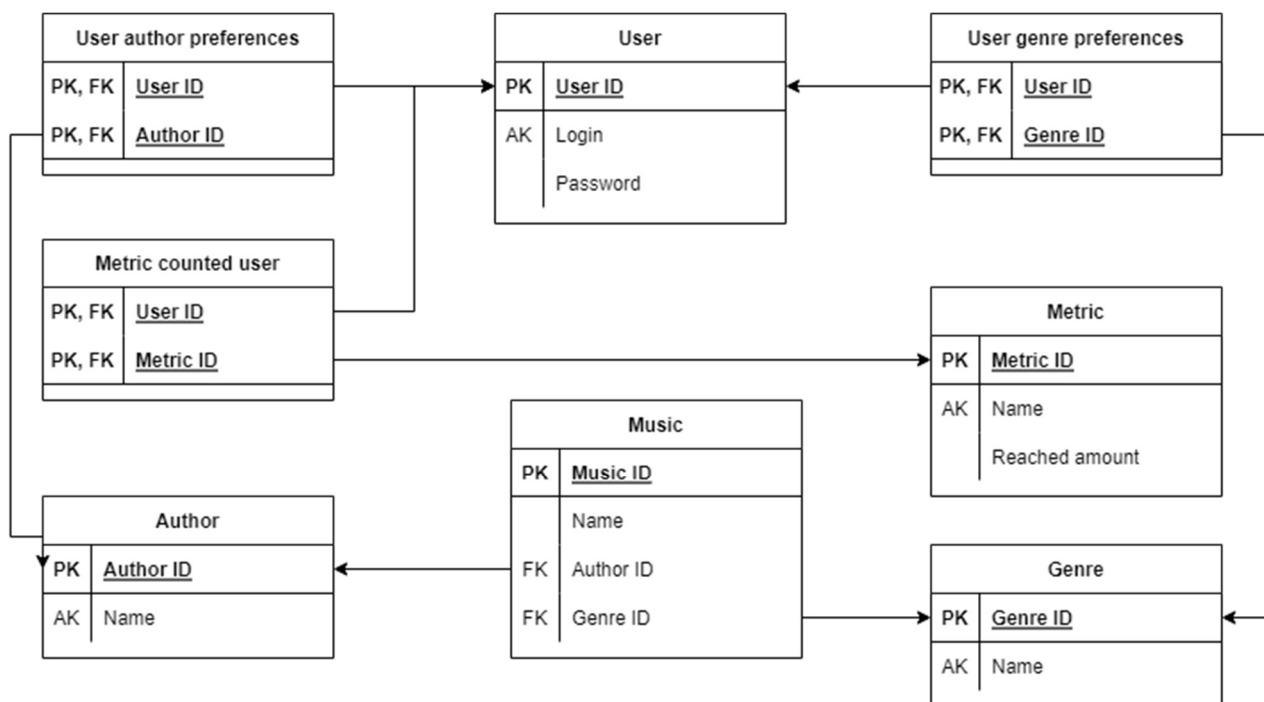


Рисунок 7 — Схема базы данных серверной части приложения.

Таблица *Metric counted user* существует для хранения пользователей, учтенных в какой-либо метрике.

Она обладает следующими атрибутами:

- **User ID**. Внешний и составной в паре с **Metric ID** ключ. Домен: целые числа. Уникальный идентификатор пользователя, учтенный в данной метрике;
- **Metric ID**. Внешний и главный составной в паре с **User ID** ключ. Домен: целые числа. Уникальный идентификатор метрики, в которой учтен данный пользователь.

Таблица *Metric* предназначена для хранения данных для составления воронок конверсии, то есть для подсчета пользователей, соответствующих данной метрике.

Она имеет следующие атрибуты:

- Metric ID. Главный ключ. Домен: целые числа. Уникальный идентификатор метрики. Назначается автоматически при помощи СУБД;
- Name. Альтернативный ключ. Домен: строки. Название метрики;
- Reached amount. Домен: целые числа. Количество пользователей, достигших данного этапа воронки конверсии.

Таблица User предназначена для хранения данных для авторизации пользователей, однажды успешно создавших аккаунт.

Она имеет такие атрибуты как:

- User ID. Главный ключ. Домен: целые числа. Уникальный идентификатор пользователя. Назначается автоматически при помощи СУБД;
- Login. Альтернативный ключ. Домен: строки. Логин пользователя;
- Password. Домен: строки. Пароль пользователя.

Таблица Genre предназначена для хранения информации о всех существующих в приложении жанрах.

Она имеет такие атрибуты как:

- Genre ID. Главный ключ. Домен: целые числа. Уникальный идентификатор жанра. Назначается автоматически при помощи СУБД;
- Name. Альтернативный ключ. Домен: строки. Название жанра.

Таблица Author предназначена для хранения информации о всех существующих в приложении авторах, причем не в буквальном смысле, то есть автором может считаться исполнитель или группа.

Она имеет такие атрибуты как:

- Author ID. Первичный ключ. Домен: целые числа. Уникальный идентификатор автора. Назначается автоматически при помощи СУБД;
- Name. Альтернативный ключ. Домен: строки. Имя автора не в буквальном смысле, то есть здесь может быть фамилия, имя, отчество, псевдоним или их комбинация.

Таблица Music предназначена для хранения информации о всех музыкальных произведениях, существующих в приложении.

Она имеет такие атрибуты как:

- Music ID. Первичный ключ. Домен: целые числа. Уникальный идентификатор музыкального произведения. Назначается автоматически при помощи СУБД;
- Name. Домен: строки. Название музыкального произведения;
- Author ID. Внешний ключ, ссылающийся на Author ID таблицы Author. Домен: целые числа. Указывает на уникальный идентификатор автора данного музыкального произведения;
- Genre ID. Внешний ключ, ссылающийся на Genre ID таблицы Genre. Домен: целые числа. Указывает на уникальный идентификатор жанра данного музыкального произведения.

Таблица User author preferences нужна для хранения информации о предпочитаемых пользователями авторах.

Она имеет такие атрибуты как:

- User ID. Первичный, внешний и составной в паре с Author ID ключ. Домен: целые числа. Уникальный идентификатор пользователя, имеющего в предпочтениях автора с уникальным идентификатором Author ID данной таблицы;
- Author ID. Первичный, внешний и составной в паре с User ID ключ. Домен: целые числа. Уникальный идентификатор автора, которого в качестве одного из предпочитаемых выбрал пользователь с уникальным идентификатором User ID данной таблицы.

Таблица User genre preferences предназначена для хранения информации о предпочитаемых пользователями жанрах.

Она имеет такие атрибуты как:

- User ID. Первичный, внешний и составной в паре с Genre ID ключ. Домен: целые числа. Уникальный идентификатор пользователя, имеющего в предпочтениях жанр с уникальным идентификатором Genre ID данной таблицы;
- Genre ID. Первичный, внешний и составной в паре с User ID ключ. Домен: целые числа. Уникальный идентификатор жанра, которого в качестве одного из

предпочитаемых выбрал пользователь с уникальным идентификатором User ID данной таблицы.

2.1.2. Варианты использования

Рассмотрим сначала диаграмма вариантов использования неавторизованного пользователя.

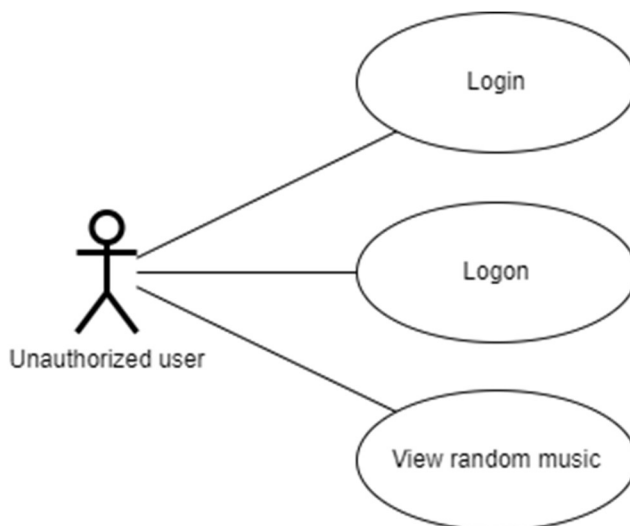


Рисунок 8 — Use-case диаграмма для неавторизованного пользователя

Также рассмотрим диаграмму использования для авторизованного пользователя.

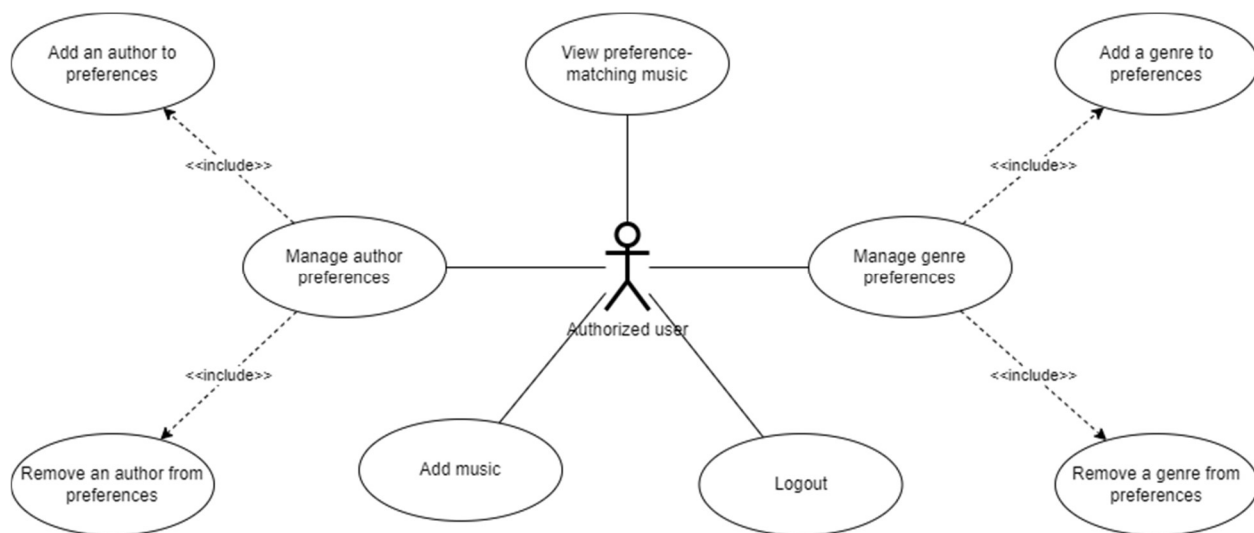


Рисунок 9 — Use-case диаграмма для авторизованного пользователя

2.1.3. Диаграммы классов

В данном пункте будут рассмотрены диаграммы только основных классов клиентской части приложения.

Одними из таких классов являются сущности, в которых хранятся полученные от сервера данные из базы данных.

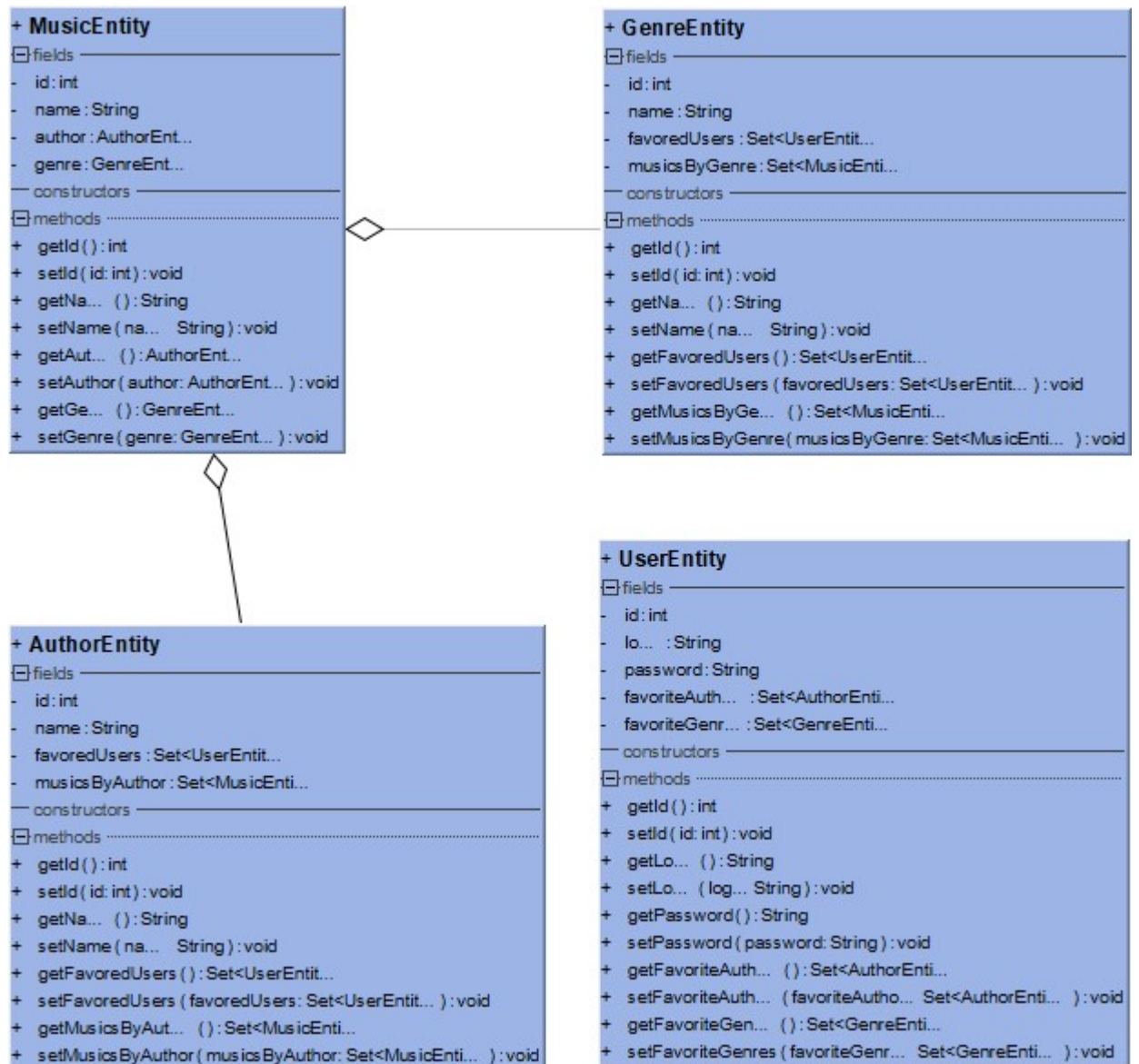


Рисунок 10 — Диаграмма классов сущностей клиентской части приложения.

Также стоит рассмотреть диаграмму классов API, которые играют важную роль в клиент-серверном взаимодействии на стороне клиента.

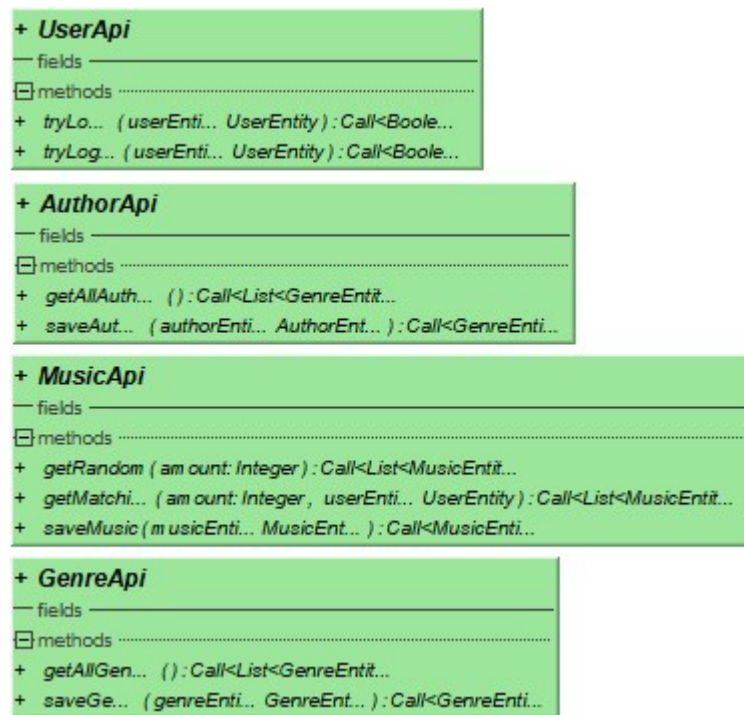


Рисунок 11 — Диаграмма классов API, взаимодействующих с серверной частью приложения.

2.1.4. Диаграммы последовательностей

Часть диаграмм опущена ввиду их тривиальности. Представлены только диаграммы, представляющие наибольший интерес.

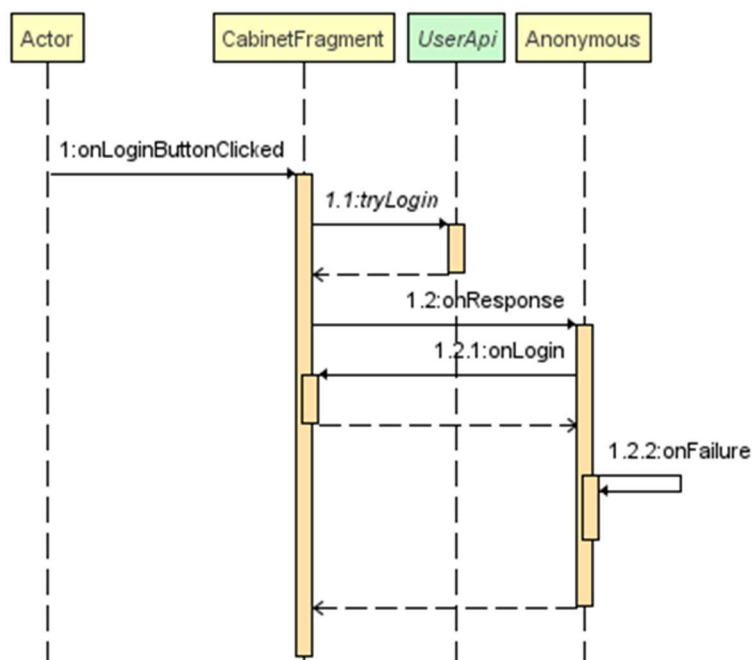


Рисунок 12 — Диаграмма последовательности входа в аккаунт.

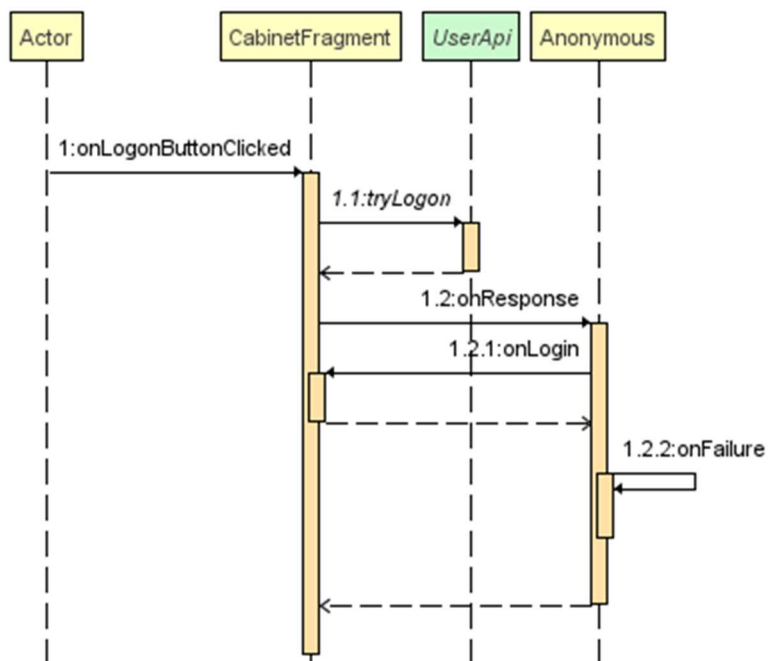


Рисунок 13 — Диаграмма последовательности регистрации.

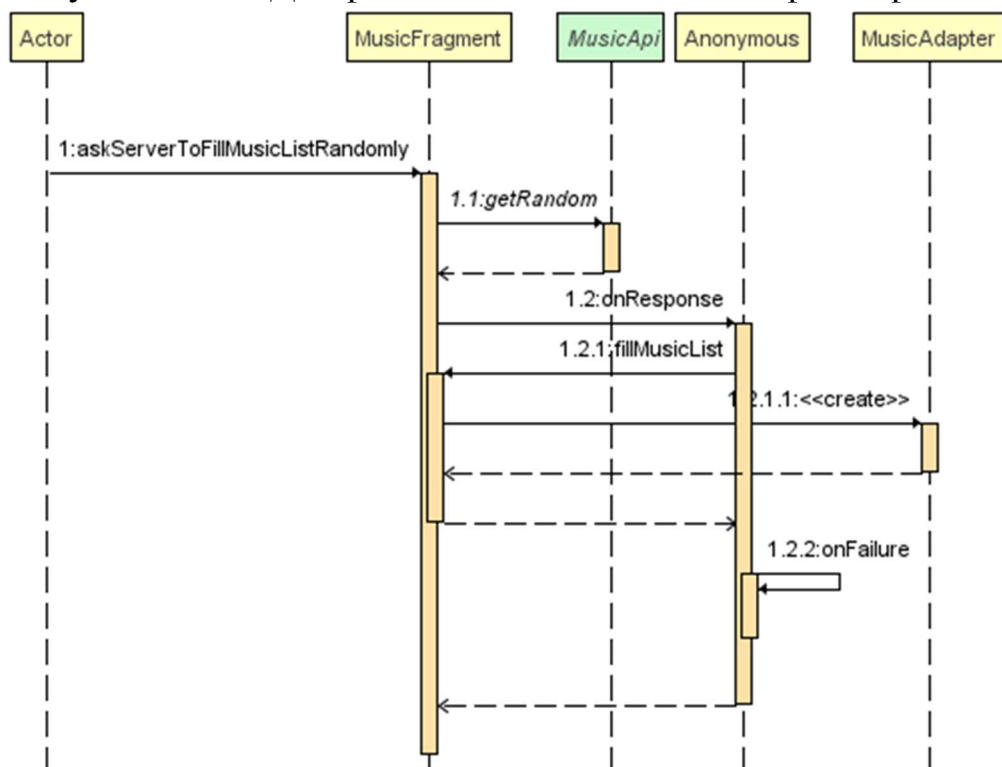


Рисунок 14 — Диаграмма последовательностей получения списка случайной музыки с сервера.

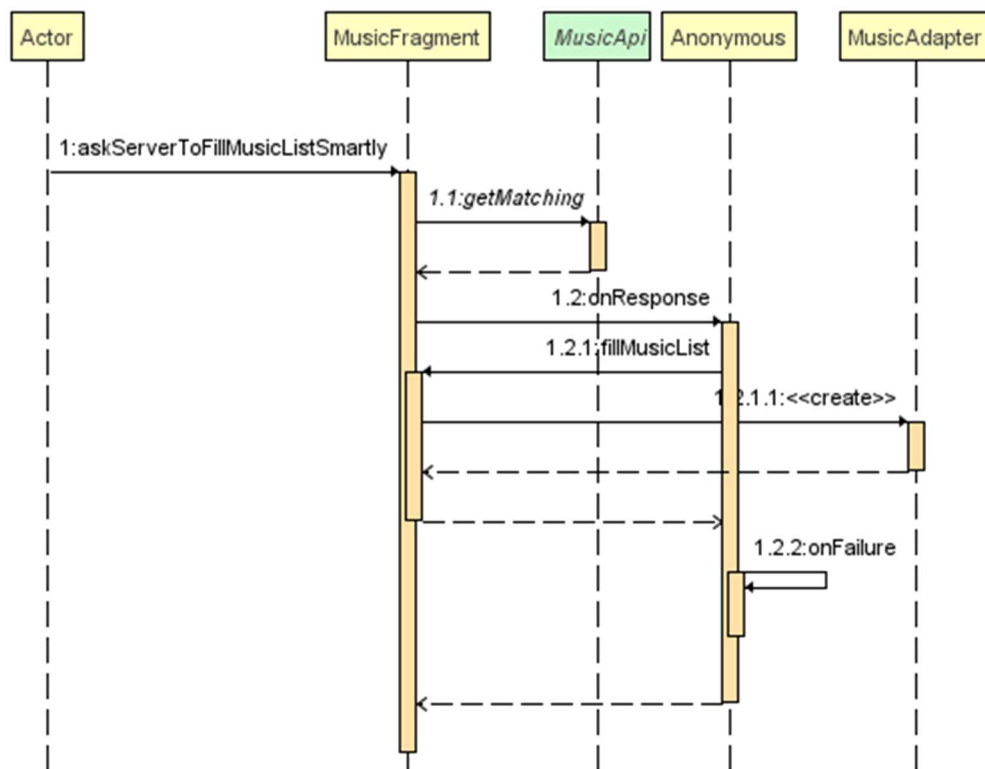


Рисунок 15 — Диаграмма последовательностей получения списка музыки с сервера, в соответствии с предпочтениями пользователей.

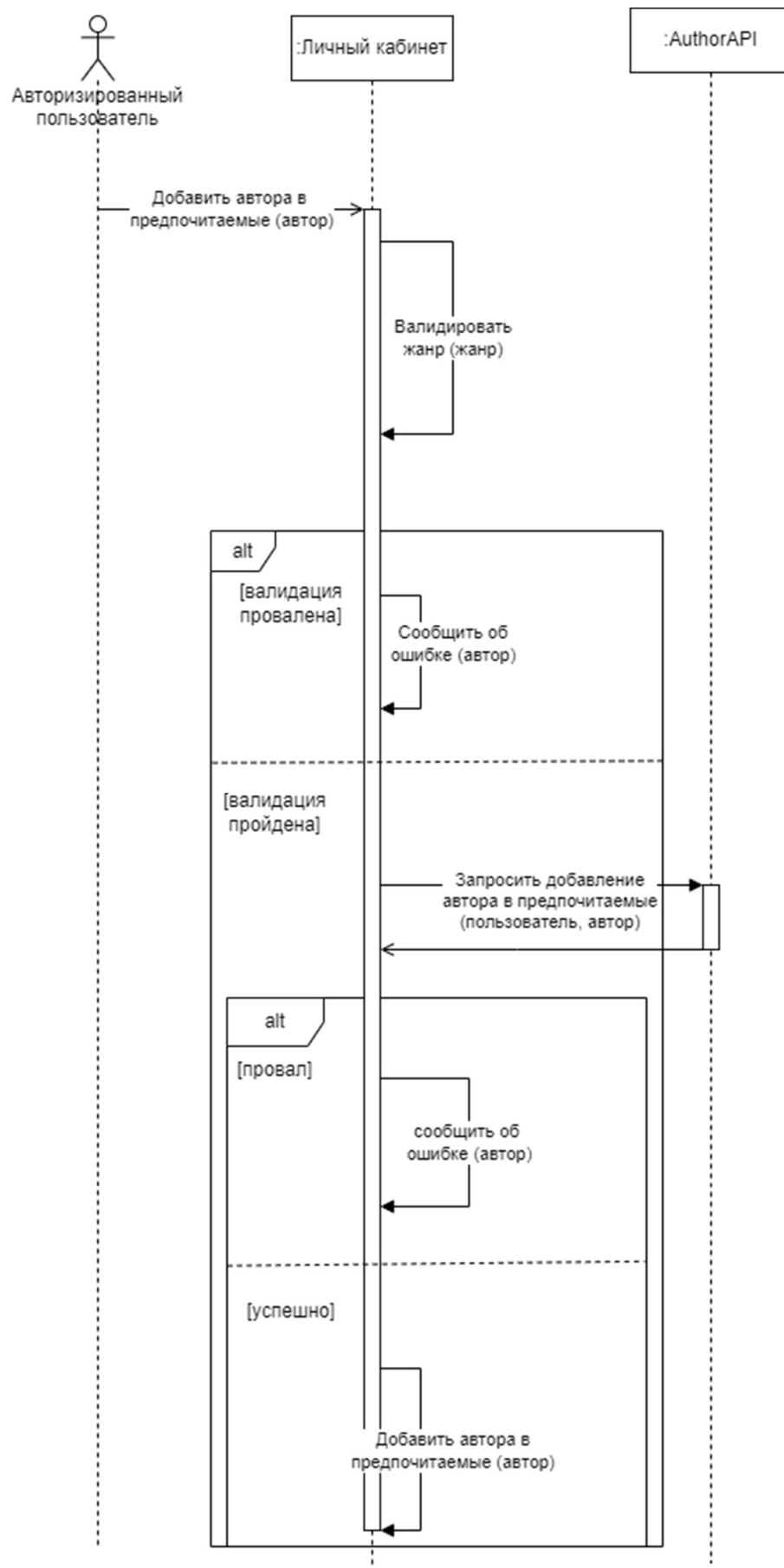


Рисунок 16 — Диаграмма последовательности добавления автора в предпочитаемые

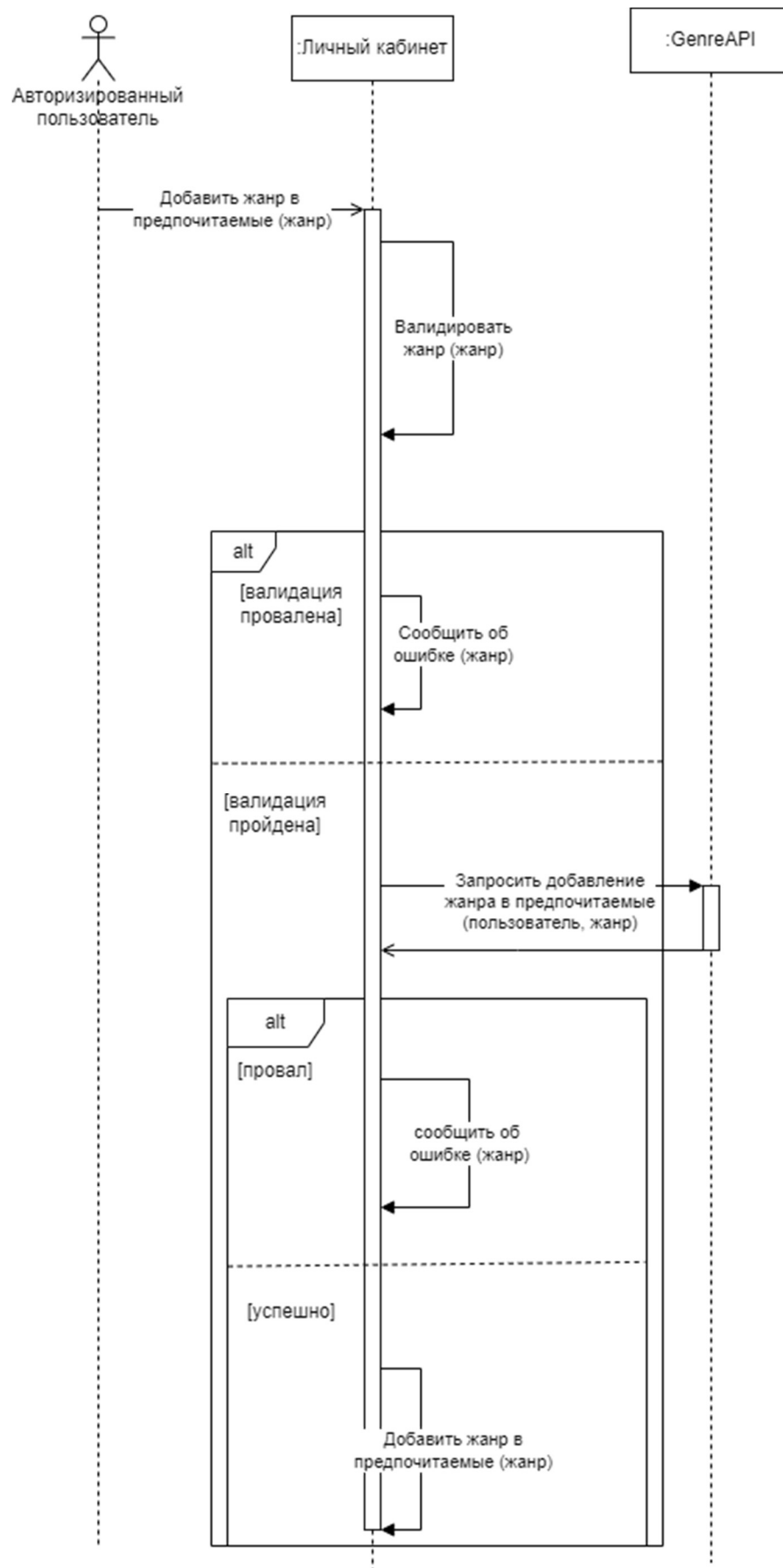


Рисунок 17 — Диаграмма последовательности добавления жанра в предпочитаемые.

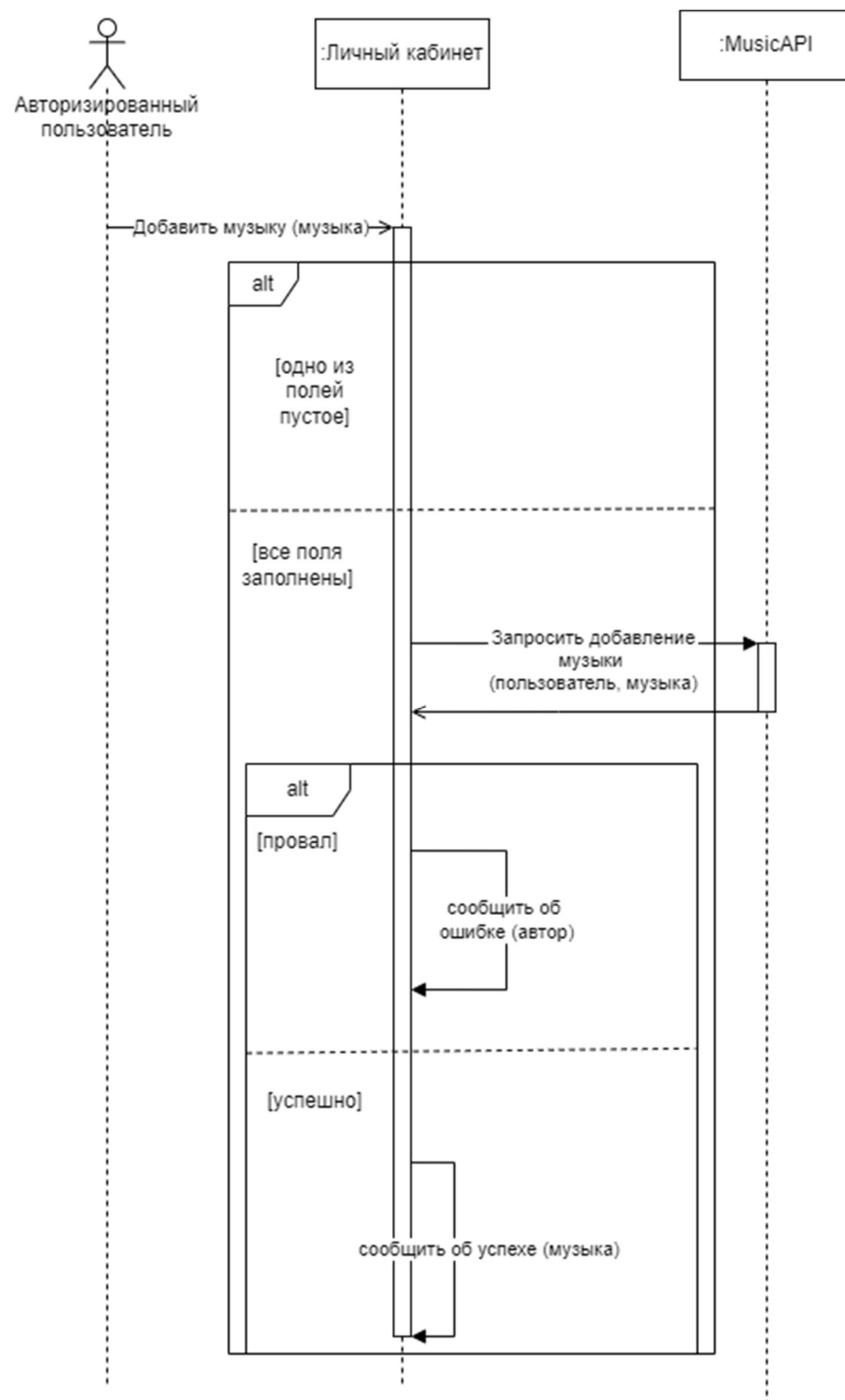


Рисунок 18 — Диаграмма последовательности добавления новой музыки.

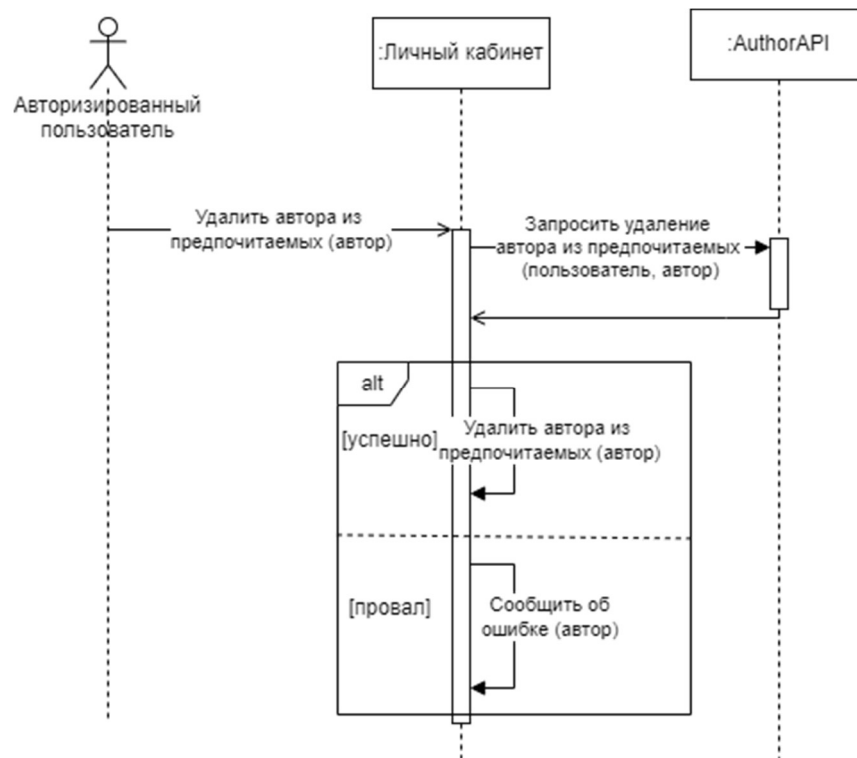


Рисунок 19 — Диаграмма последовательности удаления автора из предпочитаемых.

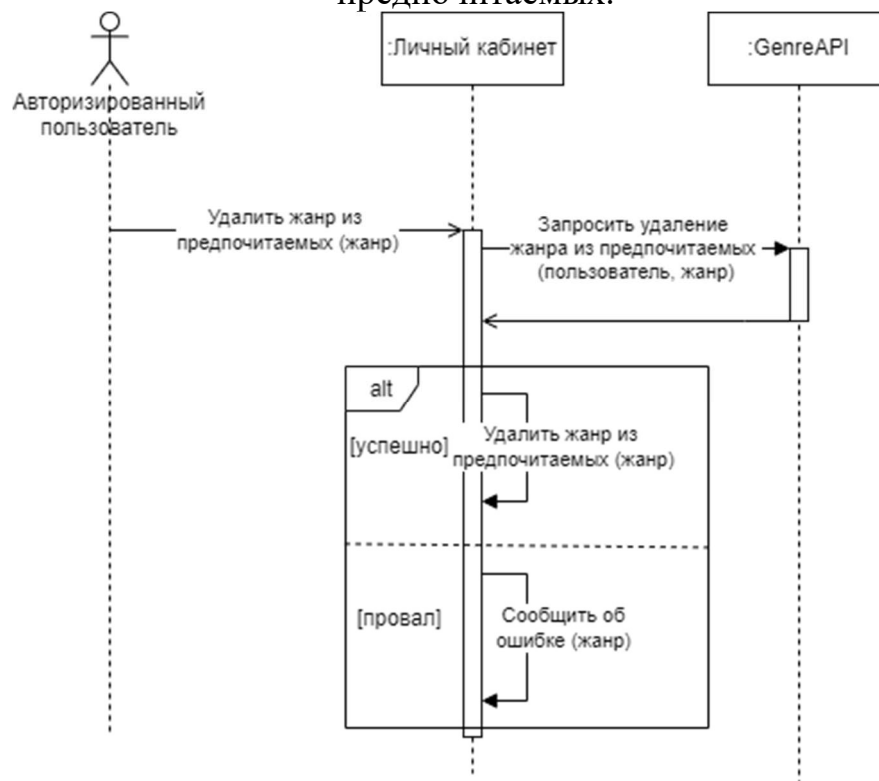


Рисунок 20 — Диаграмма последовательности удаления жанра из предпочитаемых.

2.1.5. Воронки конверсии



Рисунок 21 — Воронка конверсии пополнения списка музыкальных произведений.



Рисунок 22 — Воронка конверсии использования предпочтений.



Рисунок 23 — Воронка конверсии использования поиска музыкальных рекомендаций.

2.1.6. IDEF-0 диаграмма

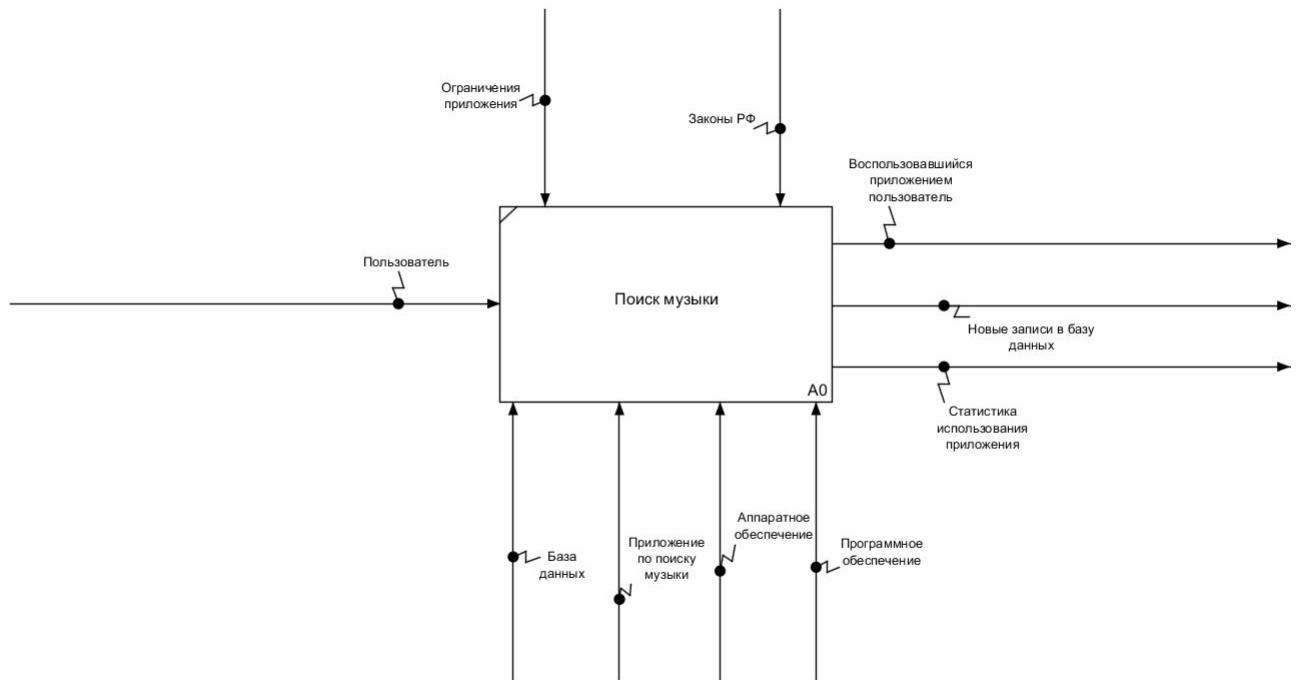


Рисунок 24 — IDEF-0 диаграмма приложения.

2.1.7. Диаграмма развертывания

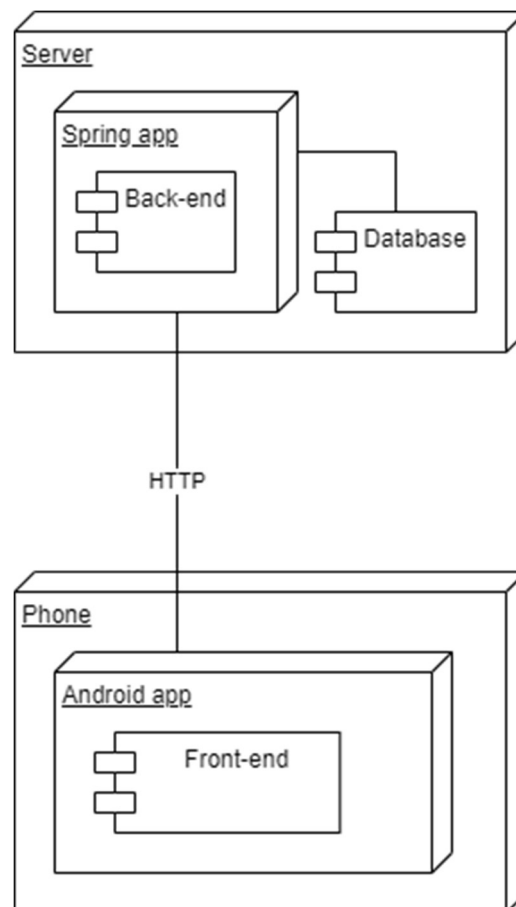


Рисунок 25 — Диаграмма развертывания приложения

2.1.8. Диаграмма состояний

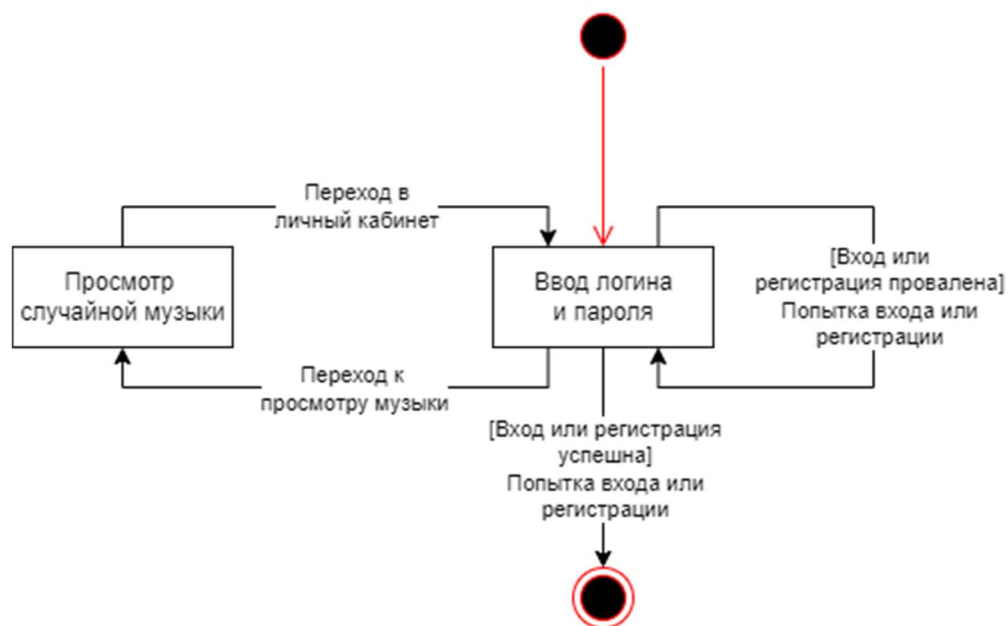


Рисунок 26 — Диаграмма состояний неавторизованного пользователя.



Рисунок 27 — Диаграмма состояний авторизованного пользователя.

2.2. Серверная часть

2.2.1. Основные положения

Основной библиотекой сервера является Java Spring Framework.

Запросы разных типов поступают на соответствующие ранее упомянутой базе данных контроллеры, которые передают обработку бизнес логики сервисам. Сервисы же, в свою очередь, взаимодействуют с репозиториями и сущностями. По окончании работы метода сервиса результат возвращается в контроллер, а далее – в тело-ответ сервера.

2.2.2. Реализация рекомендаций по предпочтениям

Одной из основных задач данного курсового проекта была реализация создания рекомендаций музыкальных произведений, основанная на предпочтениях пользователей по автору и жанру.

Рекомендации по собственным предпочтениям пользователя по автору и жанру были реализованы в одном SQL-запроса к базе данных:

```
SELECT *  
FROM music  
WHERE genre_id IN (:aids) OR author_id IN (:gids)  
ORDER BY (CASE  
    WHEN author_id IN (:aids)  
        AND genre_id IN (:gids) THEN 1  
    ELSE 2  
END)  
LIMIT :amt
```

В места «:aids» и «:gids» будут подставлены соответственно главные ключи предпочитаемых пользователем авторов и жанров. В «:amt» будет передано максимальное количество музыкальных произведений, которое может вернуть этот запрос серверу.

Данный запрос одновременно совершает выборку соответствующих предпочтениям музыкальных произведений, совершает их сортировку в зависимости от степени соответствия и ограничивает количество выдаваемых серверу результатов.

Реализация же рекомендаций по предпочтениям пользователей с похожими предпочтениями более сложна и объемна для представления ее в данном параграфе.

Ознакомиться с полным алгоритмом подбора рекомендованной музыки, а также документации, можно, перейдя по ссылке с соответствующей меткой [9].

2.2.3. Развертка

При развертке серверной части приложения использовалась облачная платформа Heroku, включая используемую базу данных.

2.2.4. Документация API

Серверная часть имеет несколько точек взаимодействия с клиентами, представленными различными запросами на различные URL.

Для подробного описания таких точек использовалась ранее упомянутая библиотека Swagger.

Документация же развернута на облачной платформе Heroku и доступна в используемых источниках с соответствующей меткой [10].

2.3. Клиентская часть

2.3.1. Основные положения

В качестве клиента выступает запущенное приложение на мобильном устройстве с операционной системой Android шестой версии или старше. Запросы отправляются на сервер при помощи ранее описанной библиотеки Retrofit.

Основу интерфейса составляют два фрагмента – личный кабинет и список музыки. Остальные экраны, такие как экран добавления музыки, предпочитаемого автора или жанра, представлены отдельной активностью, то есть Activity.

Списки музыки, предпочитаемых авторов и жанров представлены списком RecyclerView, дающим возможность описать каждый элемент списка со своей логикой. В списке музыки, в частности, присутствует плавающая кнопка FloatingActionButton, по нажатию которой открывается активность добавления

новой музыкальной записи. Для предпочитаемых авторов и жанров же используется обычная кнопка Button после соответствующих списков.

Приложение способно запоминать логин и пароль пользователей при входе в аккаунт. При перезапуске приложения, сохраненные в SharedPreferences логин и пароль, будут загружены и применены при запуске приложения автоматически.

Заключение

В ходе разработки данного мобильного приложения в рамках курсового проекта, и, в частности, в ходе анализа приложений по рекомендации музыки, была выявлена потребность в создании нового приложения, лишенного недостатков проанализированных аналогов.

Для этого были поставлены задачи, в соответствии с которыми было создано клиент-серверное приложение «Музкат», предназначенное для рекомендации музыкальных произведений пользователям, причем, для зарегистрированных пользователей, рекомендации основываются на заданных ими предпочтениях по автору или жанру, включая рекомендации по жанрам и авторам пользователей, имеющих похожие предпочтения.

Список использованных источников

1. Безопасность баз данных под управлением Oracle [Электронный ресурс]. Режим доступа: URL: https://www.aladdin-rd.ru/company/pressroom/articles/bezopasnost_baz_dannyh_pod_upravleniem_oracle. – Заглавие с экрана. – (Дата обращения 26.05.2022).
2. Java Spring Framework [Электронный ресурс]. Режим доступа: URL: <https://ask-dev.ru/info/6467/what-are-the-pros-and-cons-of-the-assorted-java-web-frameworks>. – Заглавие с экрана. – (Дата обращения 26.05.2022).
3. Преимущества и недостатки использования Spring Boot [Электронный ресурс]. Режим доступа: URL: <https://javarush.ru/groups/posts/3380-kofe-breyk-75-preimujshestva-i-nedostatki-ispoljhzovanija-spring-boot-funkcii-dlja-strok-v-java>. – Заглавие с экрана. – (Дата обращения 26.05.2022).
4. JPA и гибернация [Электронный ресурс]. Режим доступа: URL: <https://javascopes.com/introducing-jpa-and-hibernate-4hgc-00778adc/>. – Заглавие с экрана. – (Дата обращения 26.05.2022).
5. Lombok преимущества и недостатки [Электронный ресурс]. Режим доступа: URL: <https://www.codetd.com/ru/article/6990413>. – Заглавие с экрана. – (Дата обращения 26.05.2022).
6. IntelliJ IDEA [Электронный ресурс]. Режим доступа: URL: <https://habr.com/ru/post/112749/>. – Заглавие с экрана. – (Дата обращения 26.05.2022).
7. Анализ среды разработки мобильных приложений android studio. [Электронный ресурс]. Режим доступа: URL: <https://ilyarm.ru/analiz-sredy-razrabotki-mobilnyh-prilozhenii-android-studio-razrabotka.html>. – Заглавие с экрана. – (Дата обращения 26.05.2022).
8. Чем PostgreSQL лучше других SQL баз данных. [Электронный ресурс]. Режим доступа: URL: <https://habr.com/ru/post/282764/>. – Заглавие с экрана. – (Дата обращения 26.05.2022).

9. GitHub [Электронный ресурс]. Режим доступа: URL: <https://github.com/Lizurt/Muzkat>. – Заглавие с экрана. – (Дата обращения 06.06.2022).
10. Heroku [Электронный ресурс]. Режим доступа: URL: <https://muzkat-server.herokuapp.com/swagger-ui/index.html>. – Заглавие с экрана. – (Дата обращения 06.06.2022).