МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий   
Кафедра Программной инженерии   
Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем  
Направление специальности 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем (программирование интернет-приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Современные технологии программирования мобильных систем»

Тема «Программный продукт для прослушивания музыки»

Исполнитель

студент 2 курса 7 группы Котович Роман Витальевич

(Ф.И.О.)

Руководитель работы ассистент Панченко О. Л.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой ­

Руководитель Панченко О. Л.

(подпись)

**Содержание**

# **Введение**

Музыка стала неотъемлемой частью нашей жизни. Ежедневно множество людей слушают музыку на работе, дома, во время отдыха или путешествий. По этой причине очень важно иметь под рукой удобное приложение для поиска, прослушивания музыки и создания плейлистов под разные ситуации.

Данный курсовой проект посвящен созданию приложения для прослушивания музыки.

Особое внимание было уделено UI/UX-дизайну интерфейса приложения. В настоящее время пользователь имеет к приложениям со схожим функционалом. Поэтому, чтобы привлечь внимание к программному продукту и выделиться среди аналогов, необходима тщательная проработка дизайна интерфейса.

В процессе работы были поставлены следующие задачи: разработать десктопное приложение с качественным дизайном; добавить возможность прослушивания различной музыки; сделать удобный редактор плейлистов; обеспечить возможность поиска и добавления в свой плейлист музыки.

Целью данной курсовой работы состоит в создании удобного приложения для прослушивания музыки.

В качестве интерфейса приклaдного программирования был выбран API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом.

Для работы с WPF использовался объектно-ориентированный язык программирования С#.

Хранение данных осуществляется в Microsoft SQL Server.

# **1 Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству**

Одним из первых этапов разработки программного продукта является анализ прототипов прототипов и литературных источников. Исходя из анализа формируются требования к проектируемому программному средству. В процессе анализа были рассмотрено веб-приложение: «Spotify».

«Spotify» - стриминговый сервис, позволяющий прослушивать музыкальные композиции. На данный момент это приложение является одним из самых популярных сервисов для прослушивания музыки, поэтому оно и было выбрано как аналог.

На рисунках 1.1, 1.2 и 1.3 представлен интерфейс приложения.

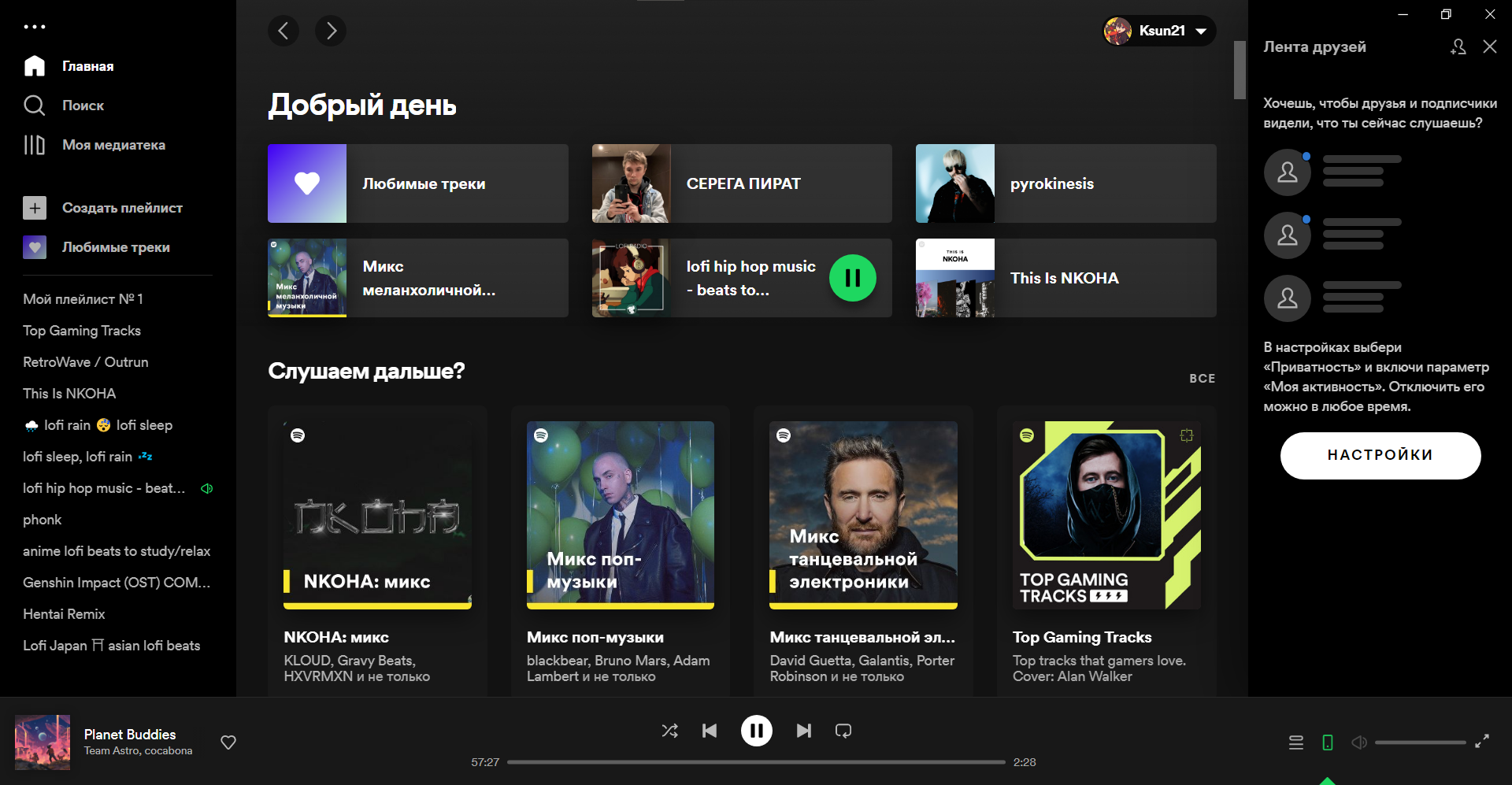


Рисунок 1.1 - Пример интерфейса приложения Spotify

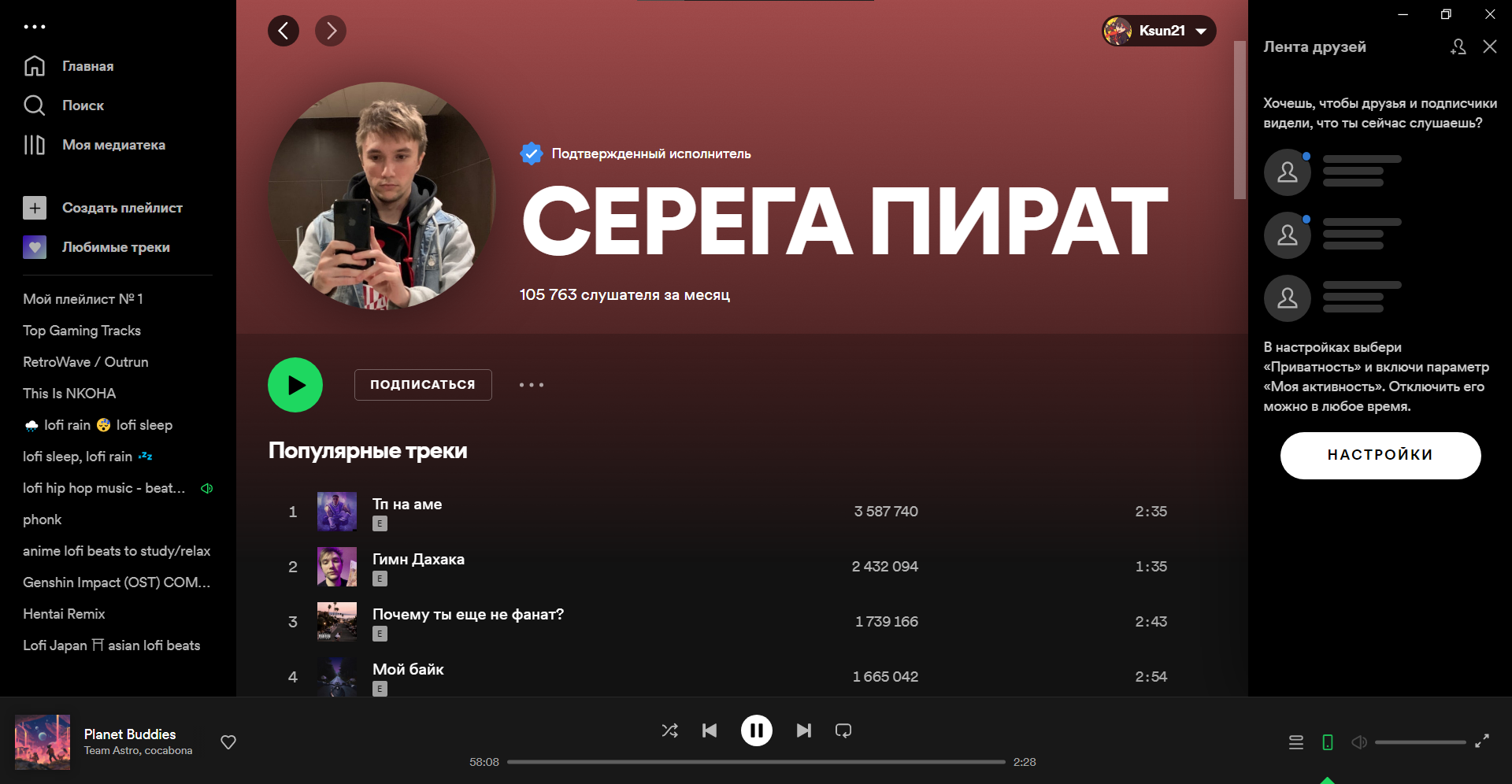


Рисунок 1.2 - Пример интерфейса приложения Spotify

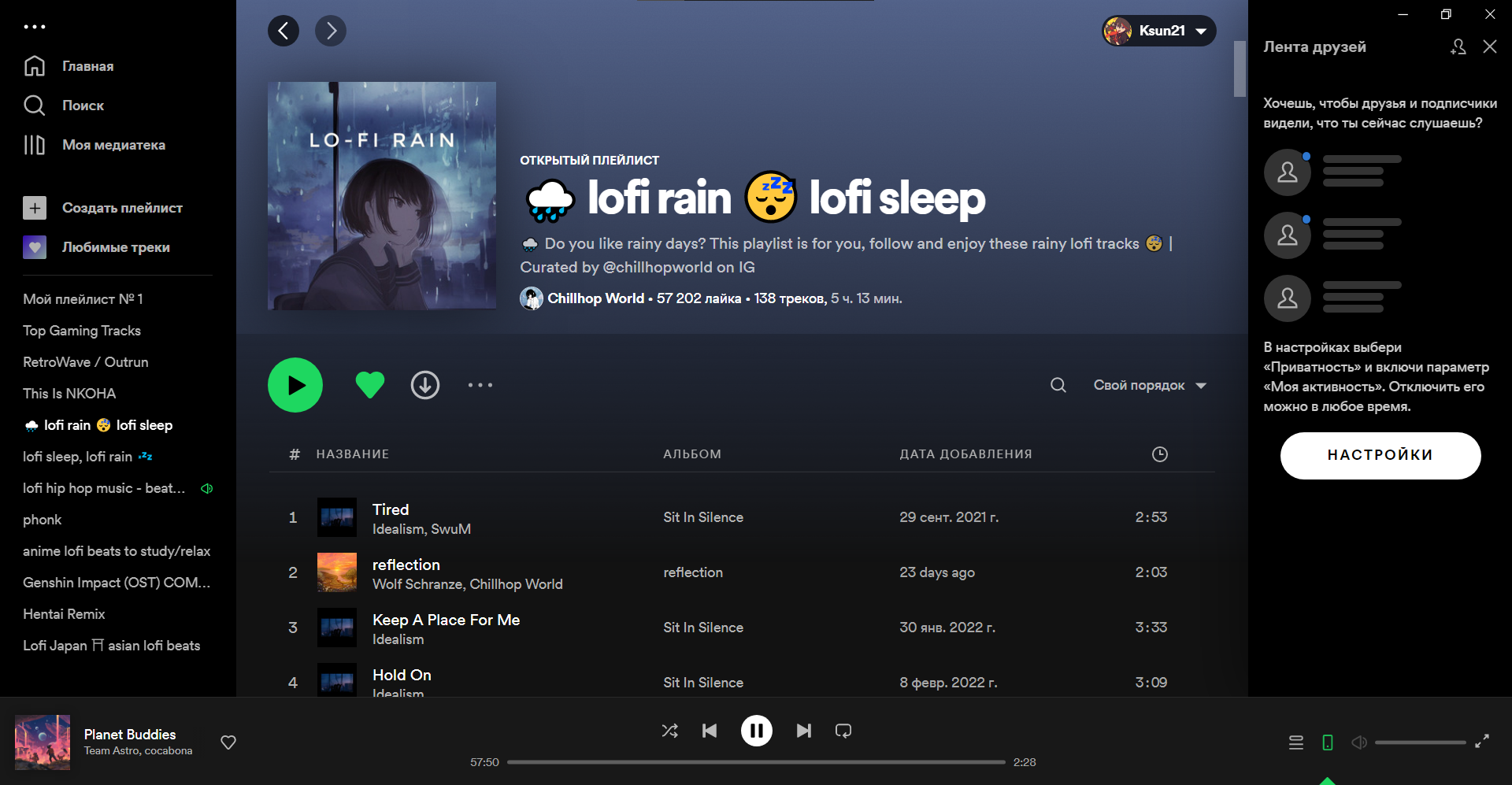


Рисунок 1.3 - Пример интерфейса приложения Spotify

Приложение поддерживает функции:

* Создать свой плейлист;
* Добавление песен в свои плейлисты;
* Поиск песен, артистов, альбомов и плейлистов;
* Добавление собственных песен.

Данное приложение является очень удобным, поэтому его достоинства будут учтены и использованы в разрабатываемом программном продукте.

В разработке приложения были использованы нижеперечисленные технологии.

Windows Presentation Foundation (WPF) — это система следующего поколения для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. C WPF можно создавать широкий спектр как автономных приложений, так и приложений, размещенных в веб-обозревателе. В основе WPF лежит векторная система визуализации, не зависящая от разрешения и созданная с расчетом на возможности современного графического оборудования. WPF расширяет базовую систему полным набором функций разработки приложений. Именно использование WPF позволило гибко управлять дизайном интерфейса, также стало возможным подключение различных сторонних пакетов и использование паттернов.

Entity Framework Core представляет собой специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

**2 Анализ требований к программному средству и разработка функциональных требований**

Анализ требований — часть процесса [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), включающая в себя сбор [требований к программному обеспечению (ПО)](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E), их систематизацию, выявление взаимосвязей, а также документирование. Является частью общеинженерной дисциплины «инженерия требований».

В процессе сбора требований важно принимать во внимание возможные противоречия требований различных заинтересованных лиц, таких как заказчики, разработчики или пользователи.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Анализ требований включает три типа деятельности:

* Сбор требований — общение с клиентами и пользователями, чтобы определить, каковы их требования; анализ предметной области.
* Анализ требований — определение, являются ли собранные требования неясными, неполными, неоднозначными или противоречащими; решение этих проблем;
* Документирование требований — требования могут быть задокументированы в различных формах, таких как простое описание, [сценарии использования](https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%B8%D0%B9_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), [пользовательские истории](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B5_%D0%B8%D1%81%D1%82%D0%BE%D1%80%D0%B8%D0%B8).

Различают три уровня требований к проекту:

* бизнес-требования;
* пользовательские требования;
* функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

* простота и лёгкость интерфейса;
* использование принципов объектно-ориентированного программирования;
* использование архитектурных шаблонов проектирования;
* использование системы управления базами данных (СУБД);

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы. Пользователь данного программного решения должен иметь возможность:

* регистрировать себя в системе;
* входить в приложение, после ввода данных, необходимых для аутентификации;
* производить поиск необходимых песен, альбомов или исполнителей;
* просматривать предложенные популярные альбомы;
* добавлять и удалять песни в свой собственный плейлист;
* добавлять свои собственные альбомы;

Администратор имеет возможность:

* архитектура приложения должна соответствовать шаблону проектирования;
* вся информация должна храниться в базе данных;
* приложение должно производить валидацию вводимых пользователем данных;
* приложение должно корректным образом отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным;
* приложение должно предоставлять возможность поиска треков, альбомов и исполнителей по следующим критериям: по названию и жанру;
* приложение должно предоставлять возможность добавления и удаления треков из плейлиста;
* приложение должно предоставлять возможность добавления своих собственных альбомов;

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

# **3 Проектирование программного средства**

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектировaния является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

3.1 Проектирование архитектуры приложения

Архитектура программного обеспечения — совокупность вaжнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбрaнных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурные шаблоны Repository и Unit of Work.

Паттерн Repository является одним из наиболее часто используемых паттернов при работе с данными. Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программное средство, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой.

Как правило, репозиторий привязан к одной конкретной сущности или модели, данными которой он управляет. Хотя это необязательно – в репозитории мы можем предусмотреть механизм для загрузки связанных данных из других таблиц, которые связаны с основной моделью, и ряд аналогичных операций. Но тем не менее, часто для управления одной сущностью создаётся свой репозиторий.

Паттерн Unit of Work позволяет упростить работу с различными репозиториями и даёт уверенность, что все репозитории будут использовать один и тот же контекст данных.

# **3.2 Проектирование базы данных**

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных.

Проектирование базы данных проводится в два этапа: концептуальное (инфологическое) и логическое (даталогическое) проектирование.

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня aбстракции. В результате этого этапа создаётся ER-модель. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Основными понятиями ER-модели являются: сущность, связь и атрибут

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности − это любая детaль, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

В рамках этого этапа была создана ER-модель, которая включает 5 сущностей:

* альбом;
* жанр;
* плейлист;
* песня;
* пользователь;

Также в ER-модели были определены необходимые связи. Например, между сущностями пользователь и трек была установлена связь один-ко-многим. Для каждой сущности были выделены aтрибуты. Например, для пользователя в качестве атрибутов были выделены такие характеристики, как идентификатор пользователя, логин пользователя, имя пользователя, пароль пользователя, статус пользователя (слушатель и исполнитель) и изображение пользователя.

Логическое (даталогическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели дaнных логическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Логическая модель базы данных представлена на рисунке 3.3

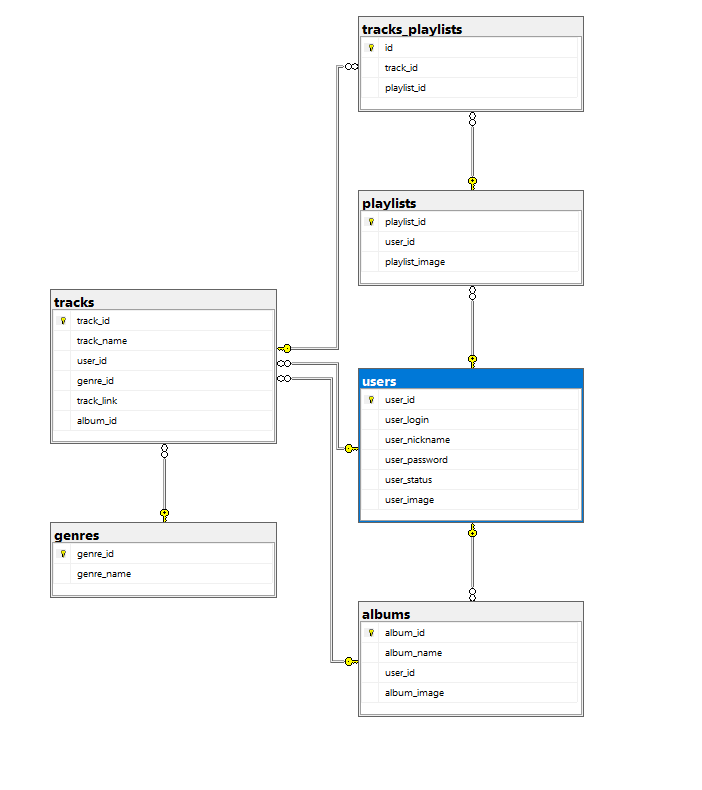


Рисунок 3.1 - Логическая модель базы данных

Всего в базе данных содержится 6 таблиц. В таблице users хранятся все пользователи, зарегистрированные в приложении. Таблица genres хранит все жанры песен. Таблица albums хранит все альбомы, добавленные пользователями в приложение. В таблице tracks хранятся все песни, которые есть в приложении. В таблице playlists хранятся все плейлисты с песнями, которые были добавлены пользователями в них. С помощью таблицы tracks\_playlists организована связь многие-ко-многим между таблицами tracks и playlists.

Таблица users состоит из 6 столбцов:

* user\_id;
* user\_login;
* user\_nickname;
* user\_password;
* user\_status;
* user\_image;

В столбце user\_id хранится идентификатор пользователя, user\_login – логин пользователя, user\_nickname – имя пользователя, user\_password – пароль пользователя, user\_status – статус пользователя, user\_image – изображение пользвателя

Таблица genres состоит из 2 столбцов:

* genre\_id;
* genre\_name.

В столбце genre\_id хранится идентификатор жанра, genre\_name – название жанра.

Таблица albums состоит из 4 столбцов:

* album\_id;
* album\_name;
* album\_image;
* user\_id.

В столбце album\_id хранится идентификатор альбомы, album\_name – название альбома, album\_image – изображение альбома, user\_id – идентификатор пользователя, которому принадлежит альбом.

Таблица playlists состоит из 3 столбцов:

* playlist\_id;
* playlist\_image;
* user\_id.

В столбце playlist\_id хранится идентификатор плейлиста, playlist\_image – изображение плейлиста, user\_id – идентификатор пользователя, которому принадлежит данный плейлист.

Таблица tracks состоит из 6 столбцов:

* track\_id;
* track\_name;
* track\_link;
* user\_id;
* album\_id;
* genre\_id.

В столбце track\_id хранится идентификатор песни, track\_name – название песни, track\_link – ссылка на песню, user\_id – идентификатор пользователя, который добавил песню, album\_id – идентификатор альбома, которому принадлежит песня, genre\_id – идентификатор жанра, которому принадлежит песня.

Таблица track\_playlists состоит из 3 столбцов:

* id;
* track\_id;
* playlist\_id;

В столбце id – идентификатор строки таблицы, track\_id – идентификатор песни, playlist\_id – идентификатор плейлиста.

3.3 Проектирование доступа к базе данных

Для доступа к базе данных используется Entity Framework Core 6. Этот подход предоставляет ряд существенных преимуществ: нам не нужно беспокоиться о коде доступа к данным и не нужно знать деталей работы СУБД SQL Server и синтаксиса языка запросов T-SQL, вместо этого мы работаем с таблицами базы данных как с классами C#, с полями этих таблиц - как со свойствами классов, а синтаксис SQL-запросов заменен на  [LINQ](https://professorweb.ru/my/LINQ/base/level1/info_linq.php). Entity Framework Core берет на себя обязанности по преобразованию кода C# в SQL-инструкции.

Существуют 3 подхода по проектированию базы данных:

* Database-First;
* Model-First;
* Code-First.

В своей работе я использовал подход Database-First. Данный подход подходит для тех случаев, когда уже создана база данных. Поэтому перед созданием программного продукта сначала была создана база данных для разрабатываемого продукта.

3.4 Проектирование вариантов использования

На рисунке 3.2 представлена диаграмма использования приложения для пользователя и администратора.

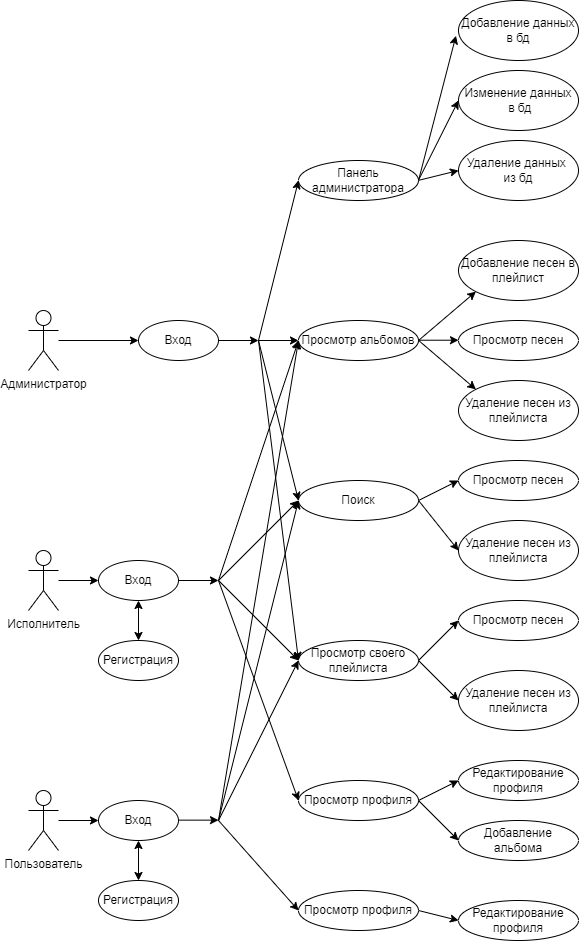


Рисунок 3.2 - Диаграмма вариантов использования для пользователя и администратора

На диаграмме видно, что пользователь при запуске будет приложения будет попадать на страницу входа, на которой чтобы перейти дальше необходимо ввести соответствующие данные или перейти на страницу регистрации и ввести данные для регистрации.

После входа в приложение пользователю откроется главное окно приложения с навигационной панелью, с помощью которой можно переходить по всем страницам приложения. Первой страницей открывается главная страница, где находятся популярные альбомы. Пользователь может их просмотреть на содержимое и узнать все песни, которые находятся в данных альбомах. Также пользователь может добавить или удалить песню из своего плейлиста и прослушать любую песню из плейлиста.

Пользователь может переходить на другие страницы. На странице поиска есть возможность найти любую песню или любой альбом по названию, а также фильтровать песни по жанру. На странице своего плейлиста у пользователя есть возможность просмотра песен, которые пользователь добавлял из альбомов, а также удалять и прослушивать данные песни. На странице своего профиля у пользователя есть возможность отредактировать свой профиль. В случае изменение своего статута на «Исполнитель», у пользователя добавляется функция добавления, изменения и удаления своих собственных альбомов.

У администратора при входе появляется ещё одна страница с панелью администратора. В ней администратор может просматривать все данные базы данных и редактировать их. В редактирование входят возможности добавления, изменение и удаление любых данных.