Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и управление»***

КАФЕДРА ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему:

Разработка веб-приложения социальной сети с функциями музыкального стримингового сервиса

по дисциплине *Компьютерные сети*

Студент гр. *ИУК4-72Б* _____ (*Сафронов Н.С.*)
(подпись) (Ф.И.О.)

Руководитель _____ (_____)
(подпись) (Ф.И.О.)

Оценка руководителя _____ баллов _____
30-50 (дата)

Оценка защиты _____ баллов _____
30-50 (дата)

Оценка проекта _____ баллов _____
(оценка по пятибалльной шкале)

Комиссия: _____ (*Белов Ю.С.*)
(подпись) (Ф.И.О.)

_____ (*Гагарин Ю.Е.*)
(подпись) (Ф.И.О.)

_____ (*Красавин Е.В.*)
(подпись) (Ф.И.О.)

Калуга, 2023

Калужский филиал
федерального государственного бюджетного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой *ИУК4*

Гагарин Ю.Е.

«*09*» *сентября* 2023 г.

ЗАДАНИЕ
на выполнение курсового проекта

по дисциплине ***Объектно-ориентированное программирование***

Студент *Сафронов Н.С. ИУК4-72Б*
(фамилия, инициалы, индекс группы)

Руководитель _____
(фамилия, инициалы)

График выполнения проекта: 25% к *4* нед., 50% к *7* нед., 75% к *10* нед., 100% к *14* нед.

1. Тема курсового проекта

Разработка веб-приложения социальной сети с функциями музыкального стримингового сервиса

2. Техническое задание

Разработать веб-приложения социальной сети с функциями музыкального стримингового сервиса

3. Оформление курсового проекта

3.1. Расчетно-пояснительная записка на *N* листах формата А4.

3.2. Перечень графического материала КП (плакаты, схемы, чертежи и т.п.)

1. Структурная схема

2. Демонстрационный чертеж

3. ER-диаграмма

Дата выдачи задания « 03 » *сентября* 2023 г.

Руководитель курсового проекта _____ / _____
(подпись) (Ф.И.О.)

Задание получил _____ / *Сафронов Н.С.* / « 03 » *сентября* 2023 г.
(подпись) (Ф.И.О.)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В наше время музыка по праву занимает своё место одних из крупнейших разделов индустрии развлечений. Она стала неотъемлемой частью досуга множества людей: как музыкантов, так и слушателей. Разработка подобного программного продукта, которым смогут пользоваться как создатели, так и потребители музыкального контента, довольно трудоёмкая задача, требующая проработки многих аспектов.

Одним из основных функциональных блоков такой социальной сети является музыкальный стриминговый сервис. Пользователи могут слушать свои любимые песни и создавать плейлисты. Кроме того, в социальной сети можно общаться с другими пользователями и делиться музыкой.

Объектом курсового проекта являются социальные сети и музыкальный стриминг.

Предметом исследования курсового проекта является реализация социальной сети с функциями стримингового сервиса.

Целью проекта является разработка web-приложения социальной сети с функциями музыкального стриминга.

Для достижения поставленной цели решаются следующие задачи:

1. Выполнить анализ предметной области;
2. Провести сравнительный анализ существующих аналогов;
3. Определить оптимальную структуру системы;
4. Осуществить выбор средств реализации программного продукта, соответствующего выбранной структуре;
5. Реализовать базу данных и программные компоненты системы;
6. Осуществить тестирование компонентов;
7. Разработать сопроводительную документацию.

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1. Введение

1.1.1. Наименование системы

Настоящее Техническое задание определяет требования и порядок создания программного продукта «Веб-приложение социальной сети с функциями музыкального стримингового сервиса».

1.1.2. Основания для разработки

Основанием для разработки социальной сети с функциями музыкального стримигового сервиса является стремление создать уникальную платформу, которая не только будет предлагает широкий доступ к музыкальному контенту, но также создаст сообщество между музыкантами и слушателями.

Такой сервис позволяет музыкантам продвигать свою музыку, делиться своим творчеством и развиваться на глазах у фанатов. В то же время, он предоставляет слушателям возможность быть ближе к артистам, получать эксклюзивные материалы, участвовать в обсуждениях и взаимодействовать с другими поклонниками.

1.1.3. Краткая характеристика области применения

Разрабатываемая система предназначена для использования как музыкальными исполнителями, так и простыми пользователями.

1.2. Назначение и цель разработки системы

1.2.1. Назначение системы

Разрабатываемая система должна предоставлять возможность пользователям создавать собственный аккаунт, выкладывать музыкальные

композиции для прослушивания другими пользователями, работать с плейлистами (для пользователей) и альбомами (для исполнителей), а также другие типовые функции социальных сетей: добавление композиций в список избранных, редактирование профиля, подписку на избранных авторов, ленту подписок и рекомендаций.

1.2.2. Цели создания системы

Целью создания системы является появление доступа начинающим музыкантам к аудитории и облегчение работы между исполнителями и конечными пользователями.

1.3. Требования к программному продукту

1.3.1. Требования к функциональным характеристикам приложения

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- разрабатываемое приложение должно иметь мобильную и браузерную версии;
- разрабатываемое приложение должно иметь удобный интерфейс для взаимодействия с пользователем, быть клиент-ориентированным и интуитивно понятным в использовании;
- пользователь должен иметь возможность регистрации и авторизации в приложении;
- при вводе некорректных личных данных пользователь должен видеть сообщение об ошибке;
- пользователь должен иметь возможность просматривать и редактировать свои профиль после регистрации и авторизации;

- пользователь может прослушивать доступные для него плейлисты и управлять процессом воспроизведения с помощью интерфейса музыкального плеера;
- пользователь должен иметь возможность создания плейлистов, их редактирования и распространения среди других пользователей;
- исполнитель может выкладывать и распространять на платформе свои композиции;
- правообладатель обладает возможностью пожаловаться на нарушение его авторских прав, которые должен рассматривать администратор платформы;
- исполнитель должен иметь возможность объединять выложенные композиции в альбомы;
- пользователь должны быть доступны типовые функции социальных сетей: поставить отметку «Нравится», подписаться на автора, оставлять комментарии;
- приложение должно обладать рекомендательной системой на основе прослушанных пользователем композиций и тегов, установленных автором;
- пользователю должна быть доступной лента – последние выложенные композиции от отслеживаемых авторов и рекомендуемые для него новые исполнители в виде списка.

1.3.2. Требования к надежности

Требования к обеспечению надежного функционирования

Программный продукт должен обеспечивать надежную защиту данных и устойчиво функционировать при допустимой нагрузке.

Надежное функционирование системы должно быть обеспечено выполнением пользователем совокупности организационно-технических мероприятий, перечень которых приведен ниже:

— организацией бесперебойного питания технических средств;

— использованием лицензионного программного обеспечения;
— регулярным выполнением рекомендаций Минтруда РФ, изложенных в Постановлении от 23 июля 1998 г. №28 «Об утверждении Межотраслевых типовых норм времени на работы по сервисному обслуживанию персональных электронно-вычислительных машин и организационной техники и сопровождению программных средств»;
— регулярным выполнением требований ГОСТ 51188-98 (защита информации, испытание компьютера на наличие компьютерных вирусов);

Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем операционной системы, не должно превышать времени восстановления операционной системы и восстановления работы сети.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

1.3.3. Условия эксплуатации

Климатические условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

Требования к квалификации и численности персонала

Система требует наличия пользователей двух категорий – рядовых пользователей сервиса, которые также могут являться и исполнителями, и администраторов, отвечающих за работу с жалобами пользователей и правообладателей на неправомерно выложенный контент.

1.3.4. Требования к защите информации и программ

Защита информации осуществляется разграничением прав доступа – обычные пользователи не должны иметь возможность вносить изменения, которые могут повлиять на функционирование системы. Также разрабатываемое приложение должно гарантировать пользователю защиту его данных. При указании соответствующих режимов приватности исполнителю также должна быть гарантирована защита объектов его творчества.

1.4. Требования к программной документации

Должны быть разработаны следующие программные документы:

1. Расчетно-пояснительная записка:
   — Техническое задание;
   — Научно-исследовательская часть;
   — Проектная часть;

Производственно-технологическая часть;

— Организационно-экономическая часть;

Раздел охраны труда и экологии;

2. Графическая часть – 3 листов формата А1 включающие в себя:

— Демонстрационные чертежи;

— Структурные схемы;

— ER-диаграммы.

1.5. Стадии и этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии проектирования программы должен быть выполнен этап выборки программного обеспечения, системы управления базой данных, библиотек для создания, этап проектирования системы в целом, разработка рабочей документации.

На стадии реализации производится разработка и тестирование спроектированной программы.

2. АНАЛИЗ АНАЛОГОВ И ПРОТОТИПОВ

2.1. Анализ существующих аналогов

В настоящее время существуют аналоги, предоставляющие подобный функционал. Среди аналогов можно выделить следующие программные решения: SoundCloud, Spotify, Bandcamp.

SoundCloud

SoundCloud является уникальным стриминговым сервисом, который предлагает не только множество музыкальных треков, но также функции социальных сетей. В отличие от других платформ, SoundCloud позволяет не только слушать музыку, но и делиться своими треками с другими пользователями. Это создает живое сообщество музыкантов, диджеев и слушателей, обеспечивая уникальный опыт.

На SoundCloud пользователи могут стать активными участниками, оставлять комментарии, ставить лайки и репосты, а также подписываться на профили других пользователей. Это позволяет взаимодействовать с артистами, находить новые музыкальные таланты и строить свою аудиторию. Кроме того, SoundCloud предоставляет возможность создавать и прослушивать плейлисты.

В контексте анализа аналоговых стриминговых сервисов с функциями социальных сетей, SoundCloud является уникальным игроком, который акцентирует внимание на связи и взаимодействии между музыкантами и слушателями. Он предлагает демократичную платформу для продвижения музыкальных талантов и способствует расширению музыкальной культуры. В целом, SoundCloud предоставляет возможность глубокого погружения в мир музыки, где пользователи сами становятся создателями и потребителями, а также находят нового вдохновения и источники музыкального творчества. В данный момент SoundCloud заблокирован на территории Российской Федерации.

Spotify

Spotify – это один из ведущих стриминговых сервисов мирового уровня, предлагающий широкий доступ к миллионам треков со всего мира.

На Spotify пользователи имеют возможность создавать собственные профили, настраивать музыкальные предпочтения и выбирать плейлисты, основанные на их вкусах. Пользователи могут следить за плейлистами других пользователей, делиться своими открытиями и даже совместно создавать плейлисты с друзьями. Это создает музыкальное сообщество, которое помогает пользователям находить новую музыку, обмениваться рекомендациями и поддерживать связи в мире музыки.

Несмотря на уход с российского рынка, Spotify остается значимым и востребованным стриминговым сервисом в мировом масштабе, предоставляя пользователям удобство и разнообразие в музыкальном контенте.

Bandcamp

Bandcamp — это уникальная платформа для музыкантов и независимых лейблов. Одной из ключевых особенностей Bandcamp является его поддержка независимых артистов, которые могут продавать свою музыку, мерч и билеты на концерты непосредственно своим поклонникам.

Пользователи Bandcamp имеют возможность открыть собственные профили и подписываться на музыкантов, чья музыка их интересует. Это позволяет пользователям получать уведомления о новых релизах и мероприятиях, а также обмениваться комментариями, рецензиями и рекомендациями с другими фанатами и музыкантами. Функция социальных сетей Bandcamp стимулирует активное взаимодействие в музыкальном сообществе, способствуя обнаружению новых талантов и созданию тесной связи между музыкантами и их поклонниками.

Однако, Bandcamp не предоставляет полноценный стриминговый сервис, а вместо этого сосредотачивается на продаже и поддержке музыки независимых артистов. К сожалению, Bandcamp также не доступен на российском рынке.

2.2. Обоснование выбора инструментов и платформы для разработки клиентской и серверной частей приложения

Обоснование выбора инструментов для программирования серверной части приложения

JavaScript

JavaScript — это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах.

JavaScript также имеет множество полезных библиотек, таких как jQuery, React и AngularJS, что делает его еще более привлекательным для разработки веб-приложений. Кроме того, JavaScript используется для создания различных приложений на мобильных устройствах, что значительно расширяет его область применения.

Еще одним преимуществом JavaScript является его простая интеграция с другими технологиями, такими как HTML и CSS. Эта возможность позволяет упростить разработку и использование внешних библиотек.

В целом, выбор JavaScript в качестве языка разработки обусловлен его эффективностью, популярностью, гибкостью, обширной поддержкой и возможностями интеграции с другими технологиями.

React

Для эффективного рендеринга компонентов интерфейса в DOM используется React.

React — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, Redux и GraphQL.

Выбор React.js обоснован его эффективностью, гибкостью и удобством использования, а также его популярностью, большим сообществом разработчиков и наличием многочисленных плагинов и библиотек, которые позволяют расширять его функциональность. Одной из таких библиотек выступила Material Design. Она предоставляет готовые компоненты и стили, позволяющие разработчикам быстро создавать визуально привлекательные пользовательские интерфейсы.

Клиентская часть должна быть не только правильно разработанной, помимо этого для этой части приложения необходимо обеспечить грамотный и приятный глазу дизайн. Данный выбор библиотеки обеспечит качественное представление элементов приложения.

Redux

Redux — библиотека с простым API, предсказуемое хранилище состояния приложений. Она работает по тому же принципу, что и функция reduce, один из концептов функционального программирования. Данная библиотека позволяет более легко масштабировать приложения, избавляет от ошибок, связанных с беспорядком в объекте состояния, повышает производительности и работоспособности программы.

Обоснование выбора инструментов для программирования серверной части приложения

Python

Python — это высокоуровневый язык программирования, который часто используется для программирования веб-приложений на бэкенде. Многие разработчики выбирают Python для разработки бэкенд-части своих приложений по следующим причинам:

1. Простота использования и высокая скорость разработки. Python — это язык программирования с простым синтаксисом и легким для изучения. Он обладает богатыми библиотеками и инструментами, которые позволяют

разработчикам разрабатывать более эффективный и продуктивный код, что сокращает время разработки и упрощает поддержку приложения.

2. Большое количество библиотек и фреймворков. В Python существует много отличных фреймворков и библиотек для разработки веб-приложений, таких как Flask, Django, Pyramid и Tornado. Эти фреймворки упрощают разработку и обеспечивают легкость в создании проекта.

3. Широкая поддержка сообщества. Python имеет большое сообщество разработчиков, которые создают новые библиотеки и расширения, а также предоставляют помощь другим разработчикам и содействуют в развитии языка и его инфраструктуры. Благодаря этому поддерживать и развивать приложение на Python можно легко и без проблем.

aiohttp

aiohttp — асинхронный HTTP-сервер для модуля asyncio. Это библиотека языка Python, которая нужна для выполнения клиентских запросов и создания веб-сервера с потоковой выдачей и веб-сокетами. Асинхронность нужна для выполнения нескольких операций, установления соединений, одновременно, без ожидания завершения предыдущих. Использование такого подхода увеличивает скорость работы веб-сервисов в несколько раз, причина низкой производительности которых, обычно, не проведение сложных вычислений, а именно ожидание ввода/вывода.

Flask

Flask — фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2. Относится к категории так называемых микрофреймворков — минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности.

uWSGI

uWSGI — это быстрый скомпилированный серверный пакет с обширной конфигурацией и возможностями, выходящим за рамки базового сервера. Он очень производителен, поскольку является компилируемым. Широкий спектр

функциональных возможностей в сочетании с относительной легкостью настройки делают uWSGI отличным вариантом для развертывания многих приложений, особенно в сочетании с Nginx.

SQLAlchemy

SQLAlchemy — это библиотека с открытым исходным кодом для работы с базами данных при помощи языка SQL. Она реализует технологию программирования ORM (Object-Relational Mapping), которая связывает базы данных с концепциями объектно-ориентированных языков программирования. SQLAlchemy позволяет описывать структуры баз данных и способы взаимодействия с ними прямо на языке Python. SQLAlchemy реализована в виде пакета для Python под лицензией MIT, а значит возможно ее использование в проприетарном ПО.

SQLAlchemy была выпущена в феврале 2006 и быстро стала одним из самых распространенных инструментов ORM среди разработчиков на Python. SQLAlchemy обладает несколькими областями применения, которые могут использоваться как вместе, так и по отдельности.

Обоснование выбора СУБД

PostgreSQL

PostgreSQL — это мощная и надежная объектно-реляционная система управления базами данных. Она является открытым исходным кодом и доступна для использования на многих ОС и архитектурах. PostgreSQL предлагает множество функций, широкий спектр поддерживаемых типов данных, а также возможность работы с многими клиентами и библиотеками.

Вот некоторые из основных возможностей PostgreSQL:

1. Расширяемость. PostgreSQL может легко быть расширен за счет создания новых типов данных, функций и хранимых процедур. PostgreSQL позволяет разработчикам создавать свои собственные типы данных, определять

свои собственные функции и процедуры, что делает его более гибким и удобным для разных приложений.

2. Транзакции. PostgreSQL поддерживает транзакции и соответствующую блокировку данных. Он также поддерживает ACID-комплектность, что делает его надежной и безопасной платформой для хранения данных.

3. Масштабируемость. PostgreSQL можно использовать в крупных проектах, где требуется обработка больших объемов данных. Он поддерживает многоядерную архитектуру, а его производительность улучшается при использовании RAID-массивов или SSD-накопителей.

4. Поддержка для сложных запросов и аналитики данных. PostgreSQL также поддерживает сложные запросы, подзапросы, оконные функции и агрегатные функции. Это делает его подходящим для целого ряда задач, связанных с аналитикой данных.

5. Открытый исходный код. PostgreSQL является проектом с открытым исходным кодом и имеет активное сообщество, которое постоянно работает над его улучшением и совершенствованием. Большое количество дополнительных пакетов и модулей доступно для использования, что позволяет настроить PostgreSQL под свои нужды.

В целом, PostgreSQL является надежной, расширяемой и производительной системой управления базами данных с множеством функций и широким сообществом. Он подходит для различных типов приложений и может быть использован как в малых, так и в крупных проектах.

Выводы

Таким образом, исходя из требований к заявленному приложению был проведен анализ нескольких инструментов для разработки, что послужило к формированию стека следующих технологий:

- для разработки клиентской части приложения были выбраны библиотеки на базе языка программирования высокого уровня —

JavaScript: React.js, Redux. Заявленные инструменты обеспечат быстрый и удобный интерфейс для пользователей приложения;

- для разработки серверной части приложения были выбраны следующие инструменты: язык программирования Python, СУБД PostgreSQL и SQLAlchemy для переноса моделей в код сервера, Flask и aiohttp для разработки серверной части, uWSGI – в качестве самого веб-сервера;
- перед клиентской и серверной частью будет стоять веб-сервер nginx, а всё приложение будет контейнеризировано и запущено с использованием docker.

2.3. Разработка структуры приложения

Представленное приложение будет построено по архитектуре клиент-сервер. Трехуровневая архитектура приложений — это модульная клиент-серверная архитектура, которая состоит из уровня представления, уровня приложения и уровня данных.

Для разрабатываемого приложения было принято использовать трёхуровневую архитектуру, так как преимущества данной модели включают улучшенную масштабируемость, производительность и доступность. В ходе разработки проекта, техническое задание может меняться, что, соответственно, будет влиять на структурную схему приложения; 3-уровневая модель облегчает непрерывное развитие проекта по мере появления новых задач и идей. Существующие приложения могут быть постоянно или временно сохранены и инкапсулированы в новый уровень, компонентом которого они становятся.

Общий вид архитектуры представлен на рисунке 1:

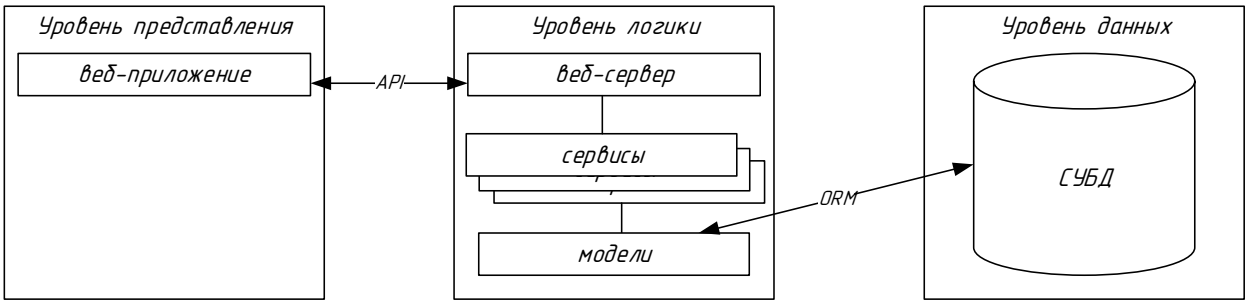

Рисунок 1. Структурная схема разрабатываемой системы

3. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

3.1. Разработка базы данных

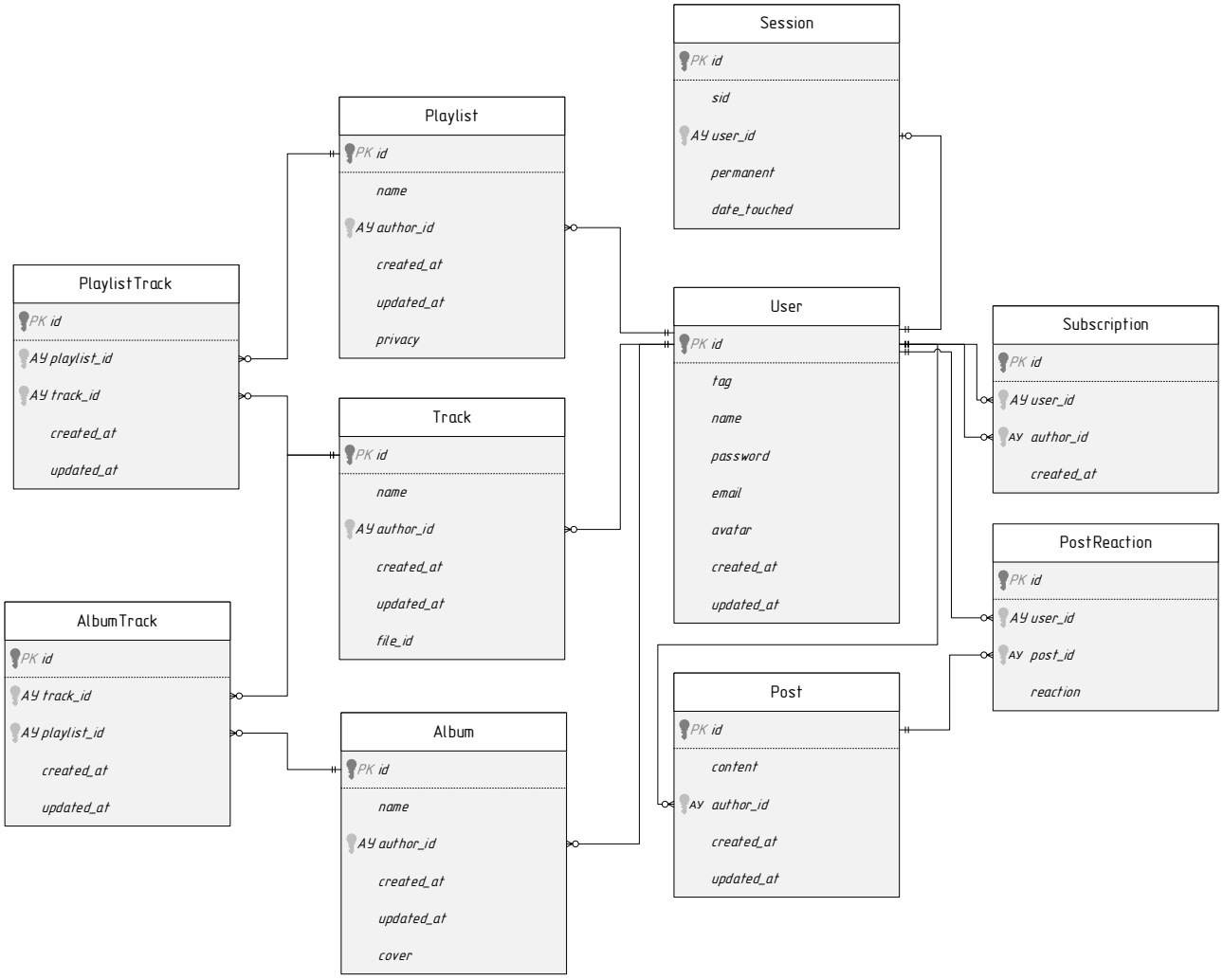

Рисунок 1. База данных

База данных состоит из следующих таблиц (см. рис. 1):

- users – хранит данные пользователей, используемые для авторизации и управления отображением пользователя;
- tracks – хранит данные о музыкальных композициях, а также идентификатор файла для поиска в файловой системе;
- playlists – хранит данные о плейлистах;
- albums – хранит информацию об альбомах, EP, LP;

- playlist_tracks – таблица реализации связи «многое-ко-многим» между таблицами playlists и tracks;
- album_tracks – таблица реализации связи «многое-ко-многим» между таблицами albums и tracks;
- posts – таблица постов на стене пользователя;
- subscriptions – таблица подписок пользователя на других пользователей;
- post_reactions – таблица реакций на посты;
- sessions – сессий пользователей.

В базе данных также был создан тип данных:

- playlist_privacy – представляет собой тип доступа к плейлисту:
  - subscribers;
  - user;
  - all.

3.2. Описание таблиц базы данных

Таблицы базы данных имеют следующие столбцы (см. табл. 1):

Таблица 1

Описание таблиц базы данных

Таблица	Название	Тип	NN	U	PK	FK	Default
users	id	integer	+	+	+		
	Идентификатор						
	tag	varchar (12)	+	+			
	Тег пользователя, используемый для его идентификации						
	password	text	+				
	Пароль, используемый пользователем при авторизации						
	avatar	text					
	Ссылка на пользовательский аватар						
	name	text	+				
	Имя пользователя						
	email	text	+				
	Электронная почта пользователя						

	created_at	time stamp	+				now()
	Время создания аккаунта						
	updated_at	time stamp					
	Время обновления аккаунта						
tracks	id	integer	+	+	+		
	Идентификатор						
	author_id	integer	+			users.id	
	Пользователь, который опубликовал трек						
	name	text	+				
	Название трека						
	created_at	time stamp	+				now()
	Время создания						
	updated_at	time stamp					
	Время обновления						
	file_id	text	+				
	Идентификатор файла внутри файловой системы						
playlists	id	integer	+	+	+		
	Идентификатор						
	name	text	+				
	Название альбома						
	privacy	playlist_privacy	+				subscribers
	Состояние приватности						
	author_id	integer	+			users.id	
	Создатель альбома						
	created_at	time stamp	+				now()
	Время создания						
	updated_at	time stamp					
	Время обновления						
playlist_tracks	id	integer	+	+	+		
	Идентификатор						
	playlist_id	integer	+			playlists.id	
	Идентификатор плейлиста						
	tracks_id	integer	+			tracks.id	
	Идентификатор трека						
	created_at	time stamp	+				now()
	Время создания						

	updated_at	time stamp					
	Время обновления						
album_tracks	id	integer	+	+	+		
	Идентификатор						
	album_id	integer	+			album.id	
	Идентификатор альбома						
	tracks_id	integer	+			tracks.id	
	Идентификатор трека						
	created_at	time stamp	+				now()
	Время создания						
	updated_at	time stamp					
	Время обновления						
post	id	integer	+	+	+		
	Идентификатор						
	author_id	integer	+			users.id	
	Автор поста						
	content	text	+				
	Текст поста						
	created_at	time stamp	+				now()
	Время создания						
	updated_at	time stamp					
	Время обновления						
post_reactions	id	integer	+	+	+		
	Идентификатор						
	user_id	integer	+			users.id	
	Идентификатор автора реакции						
	post_id	integer	+			posts.id	
	Идентификатор поста						
	reaction	text	+				
	Реакция на пост						
subscriptions	id	integer	+	+	+		
	Идентификатор						
	user_id	integer	+			users.id	
	Идентификатор подписавшегося						
	author_id	integer	+			users.id	
	Идентификатор того, на кого подписался пользователь						
	created_at	time	+				now()

		stamp					
	Время создания						
sessions	id	integer	+	+	+		
	Идентификатор						
	sid	text	+				
	Хэш идентификации сессии						
	pemanent	bool	+				false
	Флаг постоянства сессии						
	date_touched	time stamp	+				now()
	Дата создания сессии						