



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ Международных образовательных программ _____

КАФЕДРА _____ Программное обеспечение ЭВМ и информационные технологии _____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

***Рекомендательная система для музыкального
онлайн-проигрывателя с использованием
коллаборативной фильтрации на основе сходства
пользователей***

Студент ИУ7И-84Б
(Группа)

(Подпись, дата) Ву Минь Куанг
(И.О.Фамилия)

Руководитель ВКР

(Подпись, дата) Быстрицкая А.Ю.
(И.О.Фамилия)

Нормоконтролер

(Подпись, дата) _____
(И.О.Фамилия)

2024 г.

РЕФЕРАТ

Расчетно-пояснительная записка содержит 73 страницы, 4 раздела, 34 рисунка, 11 таблиц, 25 источников, 2 приложение.

Ключевые слова: музыкальные рекомендательные системы, прогноз рейтинга, коллаборативная фильтрация на основе сходства пользователей.

Результаты работы: разработанное веб-приложение музыкального онлайн-проигрывателя с рекомендательной системой.

Цель работы – Разработать веб-приложение музыкального онлайн-проигрывателя с использованием коллаборативной фильтрации на основе сходства пользователей.

В аналитическом разделе работы приведен обзор и сравнение существующих алгоритмов рекомендательной системы для музыкального онлайн-проигрывателя.

В конструкторском разделе представлены проектирование базы данных, хранящей информацию о пользователях, музыкальных треках и выставленных оценках, диаграмма прецедентов, описывающую типы пользователей и их взаимодействие с системой, и алгоритмы фильтрации музыкальных треков и рекомендательной системы.

В технологическом разделе разработано спроектированное программное обеспечение, описано его функционирование.

В исследовательском разделе приведены результаты анализа скорости выполнения при работе алгоритма, и точности используемого алгоритма с другими алгоритмами.

СОДЕРЖАНИЕ

РЕФЕРАТ.....	5
ВВЕДЕНИЕ	8
1. Аналитическая часть.....	9
1.1. Рекомендательная система.....	9
1.2. Формальная постановка задачи поиска рекомендаций	10
1.3. Фильтрация на основе контента	11
1.3.1. Фильтрация на основе памяти	13
1.3.2. Фильтрация по контенту на основе модели.....	15
1.4. Коллаборативная фильтрация	15
1.4.1. Коллаборативная фильтрация на основе памяти	17
1.4.2. Коллаборативная фильтрация на основе моделей	21
1.5. Фильтрация на основе знаний	23
1.6. Гибридные системы.....	25
1.7. Сравнительный анализ алгоритмов	26
2. Конструкторский раздел	29
2.1. Концептуальное проектирование.....	29
2.2. Логическое проектирование	29
2.3. Диаграмма прецедентов	33
2.4. Описание входных и выходных данных	38
2.5. Разработка фильтров.....	41
2.6. Описание алгоритма коллаборативной фильтрации на основе сходства пользователей	42
2.7. Разработка рекомендательной системы	44
3. Технологический раздел	47
3.1. Выбор средств программной реализации	47
3.2. Создание объектов БД.....	48
3.3. Реализация программного обеспечения	49
3.3.1. Авторизованный пользователь.....	49
3.3.2. Администратор.....	56

3.4. Сборка ПО.....	58
4. Исследовательский раздел	61
4.1. Технические характеристики.....	61
4.2. Точность предсказаний	62
4.3. Результаты исследования.....	63
ЗАКЛЮЧЕНИЕ	66
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	67
ПРИЛОЖЕНИЕ А.....	70
ПРИЛОЖЕНИЕ Б.....	73

ВВЕДЕНИЕ

Рекомендательные системы (RS) успешно применяются для прогнозирования предпочтений пользователей. RS используются в электронной коммерции (для онлайн-покупок), в развлечениях (рекомендации музыки/фильмов/видеоклипов...) и в образовании (рекомендации учебных ресурсов). Благодаря своей полезности и популярности, RS становятся интересной темой исследования. Он широко используется для построения интеллектуальных систем, особенно систем поддержки принятия решений.

Музыкальная рекомендательная система (MPC) помогает пользователям музыкальных стриминговых сервисов находить интересующий их музыкальный контент. Разреженность пользовательских оценок – одна из главных проблем использования MPC. Она вызвана тем, что пользователь оценивает лишь малую часть объектов музыкального каталога. В результате MPC часто не обладает достаточным набором данных для составления рекомендаций.

Цель работы – разработать веб-приложение музыкального онлайн-проигрывателя с использованием коллаборативной фильтрации на основе сходства пользователей.

Требуется решить следующие задачи:

1. изучить и провести сравнительный анализ существующих алгоритмов построения рекомендательных систем для разработки веб-приложения;
2. определить параметры оценки музыкальных записей для использования в алгоритме выбора;
3. разработать структуру веб-приложения музыкального онлайн-проигрывателя;
4. реализовать веб-приложение музыкального онлайн-проигрывателя;
5. исследовать разработанное программное обеспечение.

1. Аналитическая часть

1.1. Рекомендательная система

Рекомендательные системы – это программные инструменты и методики, которые предлагают пользователям объекты, наиболее интересные для них. Системы рекомендаций проводят анализ предпочтений посетителей сайта и стараются предсказать, что может понравиться им в будущем. Их алгоритмы часто строятся на основе машинного обучения: искусственный интеллект учится на выборе пользователей и предлагает им все новые возможности взаимодействия.

Целью использования рекомендательных систем является привлечение и удержание многочисленной аудитории и, следовательно, увеличение объемов продаж. Чтобы сформировать рекомендацию, система анализирует разнообразные данные – от личной информации о потребителе из его профиля в социальной сети, совершенных и несовершенных покупок до сделанных им запросов в поисковых системах. В результате на экран выводятся именно те предложения, которые могут быть интересны пользователю.

Рекомендации применяются в различных сферах: в интернет-магазинах предлагается выбрать товары в разделах: «с этим товаром покупают» или «вам может понравиться». При помощи рекомендательных алгоритмов сайты показывают интересные пользователю материалы, а соцсети предлагают добавить в друзья определенных людей. Видео- и музыкальные стриминги также используют рекомендательные системы для сбора персонализированных списков музыкальных произведений или рекомендаций фильмов для конкретного пользователя.

На рисунке 1.1, по методу построения рекомендации системы делятся на четыре основных разновидности рассматриваемых систем выдачи рекомендаций. В каждой из них используются определенные наборы алгоритмов и особенности принятия решений.

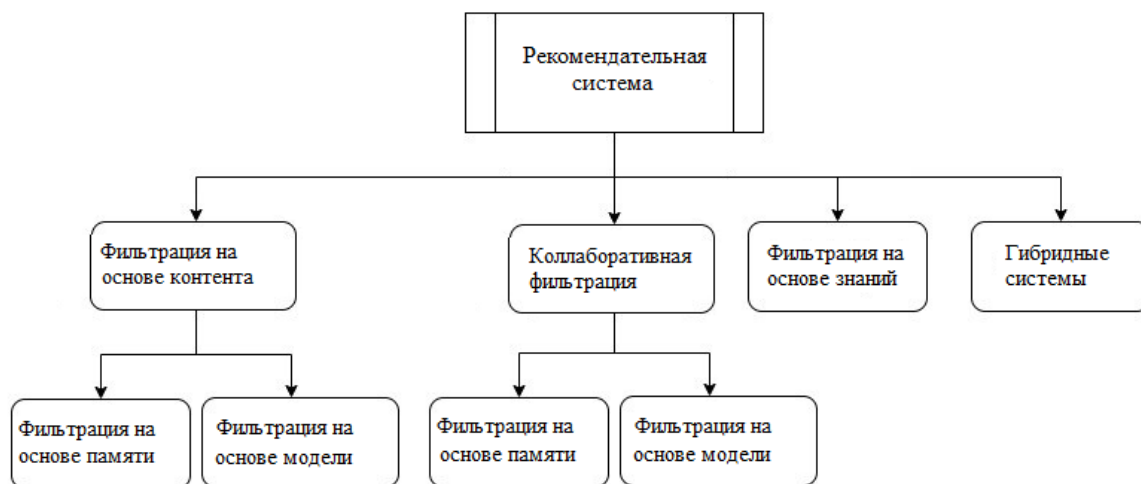


Рисунок 1.1 – Рекомендательная система.

1.2. Формальная постановка задачи поиска рекомендаций

Исходные данные

Пусть U — множество всех пользователей системы; каждый пользователь $u_i \in U$ имеет характеристики $u_i = \{u_{i1}, u_{i2}, \dots, u_{ik}\}$.

Пусть I — множество всех объектов (музыкальных записей); каждый объект $v_j \in I$ имеет характеристики $v_j = \{v_{j1}, v_{j2}, \dots, v_{jx}\}$.

Данные рейтинга $r_{ij} \in R$ — это значение оценки, которую пользователь u_i поставил продукту v_j .

Результаты

Прогнозируемое значение рейтинга r'_{ij} пользователя u_i для объектов v_j , которые еще не взаимодействовали (ранжировались).

Чтобы решить эту проблему, необходимо построить функцию $r(u_i, v_j)$, которая оценивает значение рейтинга пользователя u_i для объекта v_j так, чтобы ошибка между прогнозируемым значением r'_{ij} и значениями рейтинга r_{ij} была наименьшей.

На рисунке 1.2 показана IDEF0-диаграмма нулевого уровня, формализующая поставленную задачу.

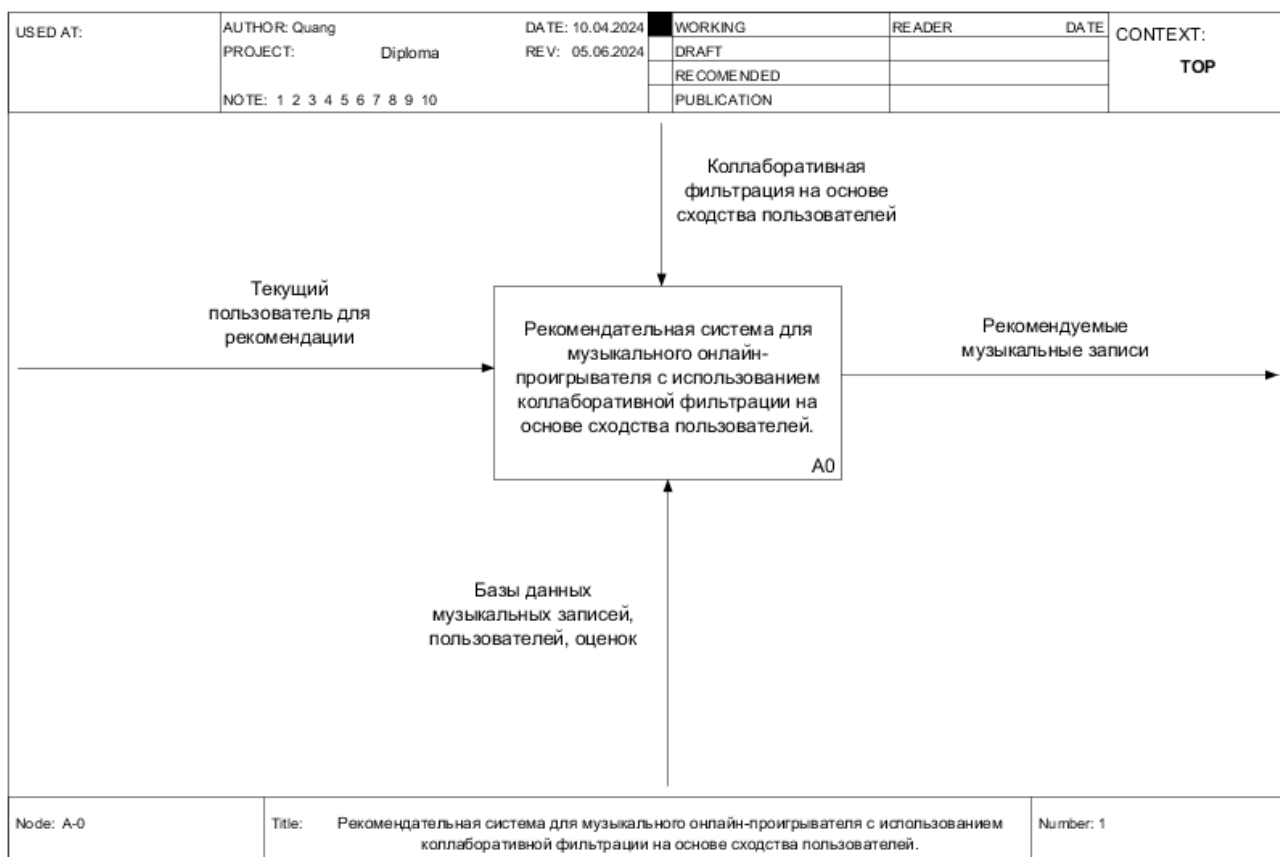


Рисунок 1.2 – IDEF0-диаграмма нулевого уровня.

1.3. Фильтрация на основе контента

Это самый простой способ определения предпочтений и интересов пользователя. Прослушивание трека рэп-исполнителя скорее всего означает, что программа порекомендует еще несколько песен этого же автора, и произведения других авторов в данном музыкальном направлении.

Достоинство такого подхода к фильтрации заключается в том, что он требует минимума ресурсов и очень прост. Недостатком метода является однообразие рекомендаций. При фильтрации на основе контента [1] алгоритмы рекомендуют товары или контент, похожие на те, которые пользователю нравились в прошлом или которые он изучает в настоящее время.

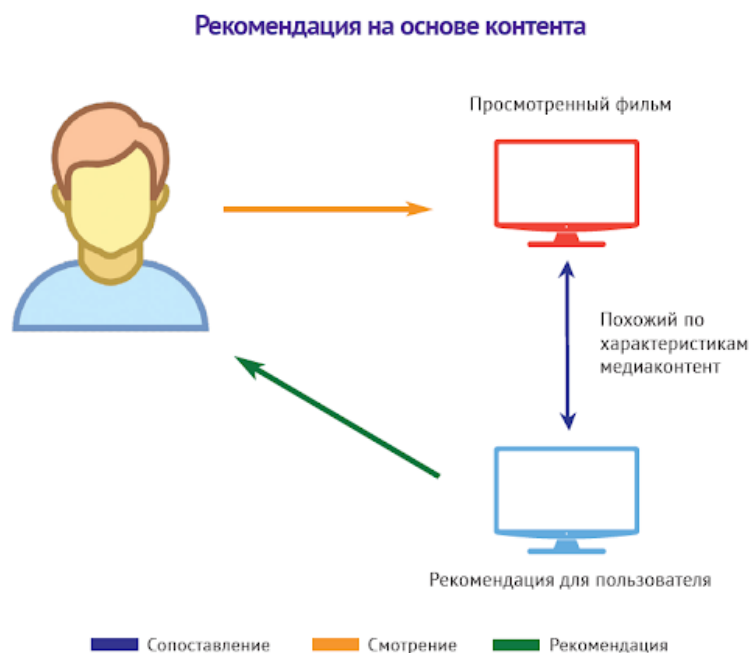


Рисунок 1.3 – Рекомендация на основе контента.

Процесс работы с использованием метода фильтрации на основе контента показан на рисунке 1.4 ниже.

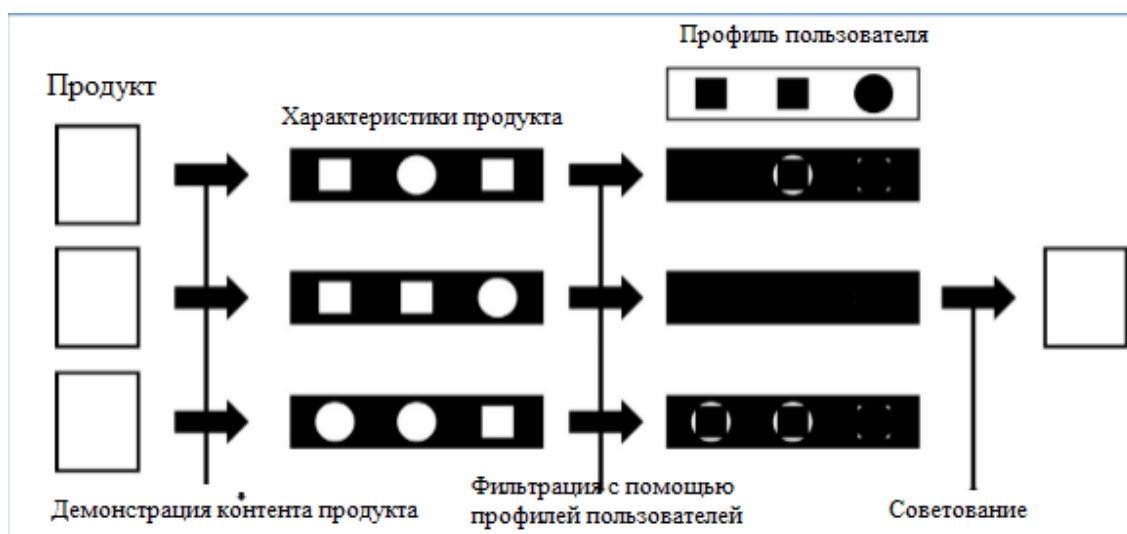


Рисунок 1.4 – Фильтрация на основании контента.

Проблемы при применении метода фильтрации контента

Хотя фильтрация контента успешно применяется во многих приложениях для фильтрации текста, этот метод имеет некоторые проблемы, которые требуют дальнейшего исследования и решения, проблема извлечения признаков и новых пользователей.

Проблема извлечения признаков. Фильтрация на основе контента развивается на основе методов извлечения признаков. Чтобы иметь полный набор функций, содержимое документа должно быть представлено в подходящей форме, чтобы компьютер мог автоматически анализировать и рассчитывать веса функций контента, или это должно быть сделано полуавтоматически. Данный метод будет трудно применять в случаях, когда извлечение контента является сложным, например, при извлечении особенностей контента из объектов мультимедийных данных (изображений, звуков и т. д.).

Проблема новых пользователей. Системы контентной фильтрации работают эффективно только тогда, когда пользователи оценивают или посещают достаточно большое количество продуктов. В случае новых пользователей профиль пользователя представляется как весовой вектор характеристик содержимого продукта, компоненты которого равны 0, поэтому система не сможет выполнять прогнозирование и распределение продуктов.

1.3.1. Фильтрация на основе памяти

Фильтрация на основе памяти использует весь профиль продукта или профиль пользователя для обучения и прогнозирования.

Для информирования пользователей используются новые продукты с наибольшим сходством с профилем пользователя. Система рекомендаций, построенная таким образом, называется системой рекомендаций на основе содержания продукта.

Метод фильтрации контента на основе памяти с использованием всего профиля продукта работает аналогично. Рекомендательные системы, построенные с использованием этого метода, называются рекомендательными системами на основе пользовательского контента.

Частотность термина (TF в уравнении) определяется следующим образом:

$$TF_{i,j} = \frac{f_{i,j}}{\max_z(f_{z,j})}, \quad (1.1)$$

где $f_{i,j}$ - количество появлений контента c_i в продукте p_j , $\max_z(f_{z,j})$ - максимальное количество появлений контента c_z в продукте p_j .

Предположим, что в системе есть N продуктов, который необходимо выделить или посоветовать пользователям и которые характеризуют контент c_i , который появляется в n_i продуктах. Обратная частота IDF_i особенностей контента c_i , частота появления которого в продукте p_j равна $TF_{i,j}$, определяется по формуле (1.2), вес контента c_i определяется по формуле (1.3).

$$IDF_i = \log \frac{N}{n_i}, \quad (1.2)$$

$$w_{i,j} = TF_{i,j} \times IDF_i. \quad (1.3)$$

Если в формуле (1.3) взять $n_i \approx N$ (когда характеристики контента c_i появляются в подавляющем большинстве продуктов) вес $w_{i,j} \approx 0$. Это означает, что данные присутствующие в каждом продукте, не содержат большого количества информации, характеризующей продукт. Напротив, если контент появляется только в одном продукте, то $n_i = 1$, следовательно $w_{i,j} = TF_{i,j}$. Таким образом, если контент появляется только в одном типе продуктов и не появляется в других продуктах, этот контент содержит много важной информации для продукта.

Согласно этой оценке, каждый продукт $p_x \in P$ представляется как взвешенный вектор особенностей контента $w_x = \{w_{1x}, w_{2x}, \dots, w_{|C|x}\}$, где $|C|$ — число, характеризующее качественное содержание всего продукта.

Для каждого пользователя $u_i \in U$, $w_i = \{w_{i1}, w_{i2}, \dots, w_{i|C|}\}$ — это весовой вектор контента продукта $c_s \in C$ для каждого пользователя u_i , или профиль пользователя u_i , в котором каждый w_{is} представляет собой вес, отражающий важность контента c_s для пользователя u_i . Вектор w_i рассчитывается с использованием различных методов на основе вектора весов особенностей контента продуктов, которые часто посещались или оценивались пользователями.

В соответствии с этим подходом новые продукты и профили пользователей представляются как весовые векторы контента продукта с одинаковой размерностью и оцениваются одним и тем же методом (в данном случае $TF - IDF$).

Следовательно, пригодность продукта $p_x \in P$ для пользователя $u_i \in U$ определяется на основе степени сходства нового продукта p_x с профилем пользователя u_i . Продукты, наиболее похожие на текущий профиль пользователя, будут использоваться при построении рекомендаций для текущего пользователя.

Распространенный метод оценки сходства между продуктом $p_x \in P$ и профилем пользователя $u_i \in U$ — это косинусная мера близости между двумя весовыми векторами w_i и w_x .

$$sim_{Cosin}(w_i, w_x) = \frac{\vec{w}_i \cdot \vec{w}_x}{|\vec{w}_i|^2 \times |\vec{w}_x|^2} = \frac{\sum_{s=1}^{|C|} w_{s,i} \cdot w_{s,x}}{\sqrt{\sum_{s=1}^{|C|} w_{s,i}^2} \cdot \sqrt{\sum_{s=1}^{|C|} w_{s,x}^2}} \quad (1.4)$$

где $|C|$ — количество характеристик содержания продукта. В формуле (1.4), если $Cosin$ двух векторов близок к 1, то угол, создаваемый этими двумя векторами, мал. Следовательно, сходство между продуктом и профилем пользователя выше, и продукт имеет более высокий уровень совместимости с пользователем. Напротив, если $Cosin$ двух векторов близок к 0, то угол, образованный двумя векторами, велик, и сходство между продуктом и профилем пользователя практически отсутствует, то продукт имеет более низкий уровень релевантности для пользователя.

1.3.2. Фильтрация по контенту на основе модели

Фильтрация контента на основе модели — это метод, который использует весь профиль продукта или профиль пользователя для обучения. Результаты обучения модели будут использоваться при прогнозировании рекомендаций для пользователя.

Pazzani и Billsus [2] используют байесовский классификатор, основанный на бинарных оценках пользователей («нравится» или «не нравится»), для классификации продуктов.

Solombo [3] предлагает модель адаптивной фильтрации, которая фокусируется на учёте релевантности всех продуктов.

1.4. Коллаборативная фильтрация

Коллаборативная фильтрация [1, 4, 5] — это метод использования аспектов, связанных с привычками использования продуктов сообществом пользователей со схожими интересами в прошлом, для прогнозирования новых продуктов, подходящих нынешним пользователям. Существует два способа фильтрации в окрестностях. В одном способе используются схожие пользователи, а в другом — элементы, схожие с теми, которые пользователю понравились.

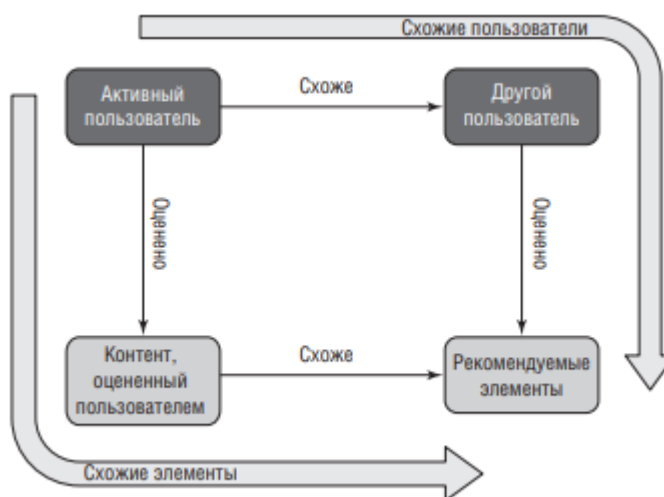


Рисунок 1.5 – Два способа коллаборативной фильтрации.

Таким образом, входной информацией рекомендательной системы, основанной на методе коллаборативной фильтрации, является отзыв пользователя о продуктах в системе, представленный через оценочную матрицу R . Затем происходит процесс построения рекомендательной системы, использующий метод коллаборативной фильтрации, как показано на рисунке 1.6.

Метод коллаборативной фильтрации при построении рекомендательных систем считается одним из лучших подходов к созданию практических рекомендательных систем с рядом преимуществ. К ним относятся простота реализации и возможность обрабатывать все типы информации, что особенно важно для мультимедийной информации, которую не требуется представлять в текстовой форме.

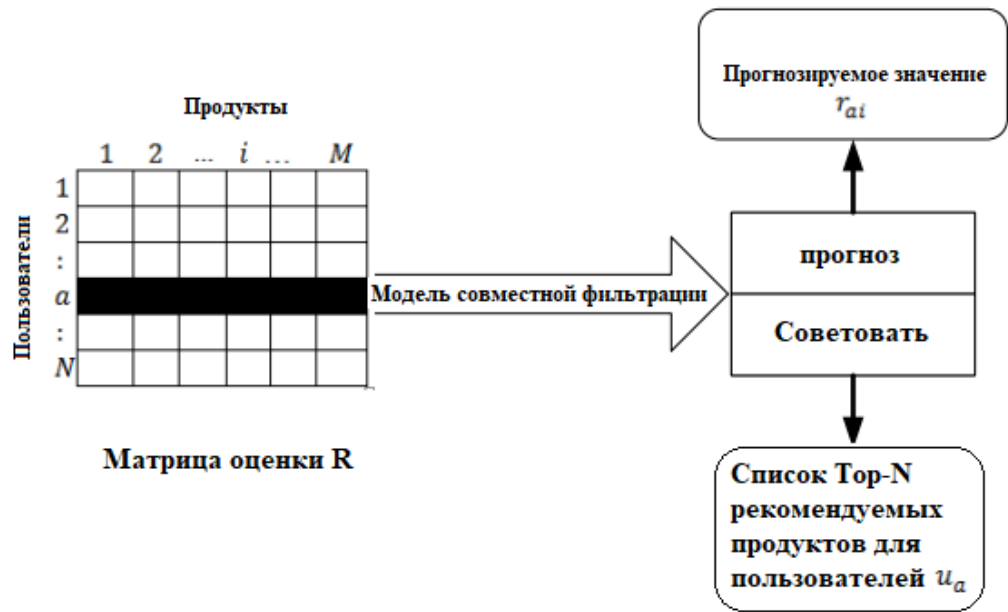


Рисунок 1.6 – Процесс построения рекомендательной системы, использующий метод коллаборативной фильтрации [6].

1.4.1. Коллаборативная фильтрация на основе памяти

Метод коллаборативной фильтрации на основе памяти [7, 8] делится на несколько подходов:

1. совместная фильтрация памяти на основе сходства пользователей;
2. совместная фильтрация в памяти на основе сходства продуктов;
3. взвешивание по сходству;
4. оцените частоту реверса пользователя.

Коллаборативная фильтрация памяти на основе сходства пользователей [7].

Степень сходства между пользователем $u_i \in U$ и текущим пользователем u_a , обозначаемая $sim(u_a, u_i)$, рассматривается на основе набора продуктов, оцененных обоими пользователями.

Корреляция Пирсона между двумя пользователями u_a и u_i :

$$sim_{Pearson}(u_a, u_i) = \frac{\sum_{p_x \in P_{ai}} (r_{ax} - \bar{r}_a)(r_{ix} - \bar{r}_i)}{\sqrt{\sum_{p_x \in P_{ai}} (r_{ax} - \bar{r}_a)^2 \cdot \sum_{p_x \in P_{ai}} (r_{ix} - \bar{r}_i)^2}}, \quad (1.5)$$

где $P_{ai} = \{p_x | r_{ax} \neq 0 \cap r_{ix} \neq 0\}$ — набор всех продуктов, оцененных u_a и u_i ; \bar{r}_a , \bar{r}_i — среднее значение ненулевых оценок u_a и u_i .

Косинусная мера сходства между двумя пользователями u_a и u_i : значение Cosin между двумя векторами u_a и u_i вычисляется по формуле (1.6), в которой они представлены m -мерным вектором ($m = |P_{ai}|$ — количество продуктов, которые оценивают оба пользователя).

$$\text{sim}_{\text{Cosin}}(u_a, u_i) = \frac{\vec{u}_a \cdot \vec{u}_i}{\|\vec{u}_a\| \times \|\vec{u}_i\|} = \frac{\sum_{p_x \in P_{ai}} r_{ax} \cdot r_{ix}}{\sqrt{\sum_{p_x \in P_{ai}} r_{ax}^2 \cdot \sum_{p_x \in P_{ai}} r_{ix}^2}}. \quad (1.6)$$

Пусть \hat{U} — набор пользователей K_1 , наиболее похожих на u_a (набор соседей с u_a). Степень соответствия пользователя u_a новому продукту p_y определяется как функция оценок набора соседей в соответствии с методами, указанными ниже:

$$r_{ay} = \begin{cases} \frac{1}{K_1} \sum_{w \in \hat{U}} r_{u'y} \\ h \sum_{w \in \hat{U}} \text{sim}(u_a, u') \times r_{u'y} \\ \bar{r}_a + h \sum_{w \in \hat{U}} \text{sim}(u_a, u') \times (r_{u'y} - \bar{r}_{u'}) \end{cases}, \quad (1.7)$$

где $h = \frac{1}{\sum_{w \in \hat{U}} \text{sim}(u_a, u')}$ — коэффициент стандартизации, $\bar{r}_a = \frac{1}{|P_a|} \sum_{p_x \in P_a} r_{ax}$ — среднее значение оценок пользователей u_a . ($P_a = \{p_x \in P | r_{ax} \neq 0\}$)

Коллаборативная фильтрация в памяти на основе сходства продуктов.

Суть этого метода заключается в том, что вместо расчета сходства между пользователями в системе и текущим пользователем u_a , рекомендательная система будет рассчитывать сходство между продуктами, которые необходимо оценить для u_a , с продуктами, которые были ранее проверены пользователем u_a . Расчет степени сходства между двумя продуктами рассматривается на основе набора пользователей, оценивающих оба продукта. На этом основании выбирается набор продуктов, которые являются соседями продуктов, которые вам необходимо оценить. Нужно объединить рейтинги u_a с набором соседних продуктов,

чтобы спрогнозировать рейтинги u_a для рассматриваемого продукта. Подобно вычислению уровня сходства между двумя пользователями для совместной фильтрации на основе сходства пользователей, вычисление уровня сходства между двумя продуктами выполняется.

Корреляция Пирсона между двумя продуктами p_x и p_y :

$$sim_{Pearson}(p_x, p_y) = \frac{\sum_{u_i \in U_{xy}} (r_{ix} - \bar{r}_x)(r_{iy} - \bar{r}_y)}{\sqrt{\sum_{u_i \in U_{xy}} (r_{ix} - \bar{r}_x)^2 \cdot \sum_{u_i \in U_{xy}} (r_{iy} - \bar{r}_y)^2}}, \quad (1.8)$$

где $U_{xy} = \{u_i | r_{ix} \neq \emptyset \cap r_{iy} \neq \emptyset\}$ — это совокупность всех пользователей, которые оценили продукт p_x и продукт p_y ; \bar{r}_x, \bar{r}_y — среднее значение ненулевых оценок p_x и p_y .

Косинусная мера сходства между произведениями p_x и p_y : это Косинус угла между двумя векторами p_x и p_y , вычисленный по формуле (1.9), в которой два продукта p_x и p_y рассматриваются как n -мерные векторы ($n = |u_{xy}|$ — количество пользователей, оценивших продукты p_x и p_y).

$$sim_{Cosin}(p_x, p_y) = \frac{\vec{p}_x \cdot \vec{p}_y}{\|\vec{p}_x\|^2 \times \|\vec{p}_y\|^2} = \frac{\sum_{u_i \in U_{xy}} r_{ix} \cdot r_{iy}}{\sqrt{\sum_{u_i \in U_{xy}} r_{ix}^2 \cdot \sum_{u_i \in U_{xy}} r_{iy}^2}}. \quad (1.9)$$

Формирование прогноза рейтинга текущего пользователя u_a для продукта p_y без рейтинга вычисляется путем объединения рейтингов u_a с продуктами в наборе соседей p_y . Пусть P — набор K продуктов, наиболее похожих на p_y . Степень согласия u_a с p_y определяется как функция оценок множества соседей. Наиболее распространенный метод прогнозирования пригодности продукта p_y для пользователя u_a определяется по формуле (1.10).

$$r_{ay} = \frac{\sum_{p' \in P} r_{ap'} \cdot sim(p', p_y)}{\sum_{p' \in P} |sim(p', p_y)|}. \quad (1.10)$$

Взвешивание по сходству

Расчет сходства между пользователями $u_i \in U$ и текущим пользователем u_a рассматривается на основе набора продуктов, которые оценивают оба пользователя. В некоторых случаях количество продуктов, которые u_i и u_a оценивают вместе, очень мало, но значение сходства между u_i и u_a все еще довольно велико, поэтому u_i выбирается в набор соседей u_a , что приводит к качеству прогноза во многих случаях не высока. Эксперименты также показывают, что, если количество продуктов, оцененных двумя пользователями u_i и u_a , велико, сходство между u_i и u_a более стабильно, и поэтому набор соседей а полезен для прогнозирования рейтинга.

На основе таких исследований и экспериментов в некоторых исследованиях были предложены решения по ограничению влияния небольшого количества продуктов, которые оценивают оба пользователя, на точность системы рекомендаций, путем введения параметра (взвешивание значимости) для взвешивания рассчитанных уровней сходства. В частности, если количество продуктов, которые оценили пользователи u_i и u_a (обозначается z), меньше порога θ , то уровень сходства между u_i и u_a , рассчитанный ранее, будет пересчитан путем умножения на $\frac{z}{\theta}$. В случае $z \geq \theta$ уровень сходства между u_i и u_a остается прежним.

$$sim(u_a, u_i) = \begin{cases} sim(u_a, u_i) \cdot \frac{z}{\theta}, & \text{если } z < \theta \\ sim(u_a, u_i), & \text{если } z \geq \theta \end{cases} \quad (1.11)$$

Оцените частоту реверса пользователя

В системе продукты, которые оценены всеми пользователями, часто не так полезны для предоставления рекомендаций, как продукты, которые не оценены всеми пользователями. Поэтому обратная частота появления пользователя $f_i = \log\left(\frac{n}{n_i}\right)$ используется для того, чтобы мы могли уделять больше внимания пользователям, которые не оценивают все продукты системы (где n_i — количество пользователей, поставивших оценку продукту i , n — общее количество пользователей). Соответственно, отзывы пользователей в матрице отзывов нормализуются путем умножения исходного значения отзыва на f_i .

1.4.2. Коллаборативная фильтрация на основе моделей

В отличие от метода совместной фильтрации на основе памяти, метод совместной фильтрации на основе модели [4, 9] использует оценочную матрицу для построения модели прогнозирования, которая генерирует рекомендации для пользователя. Преимущество этого метода заключается в том, что обучающая модель намного меньше по размеру, чем оценочная матрица, и позволяет быстрее делать прогнозы. Модель необходимо обновлять только в случае серьезных изменений и переделывать только этап построения модели.

Модель ассоциативных правил

Поскольку входными данными для совместной фильтрации являются матрица оценки, модель ассоциативных правил [10] применяет алгоритмы интеллектуального анализа ассоциативных правил для извлечения правил, которые прогнозируют появление рекомендуемых продуктов на основе их взаимосвязи с другими продуктами. Закон ассоциации представлен в форме $A \rightarrow B$, где A и B представляют собой два продукта. Считается, что модель ассоциативных правил в некоторых случаях эффективно повышает производительность рекомендательной системы, однако, когда количество пользователей и продуктов велико, поиск ассоциативных правил для системы рекомендаций станет довольно сложным.

Модель кластеризации

Алгоритмы кластеризации [11] пытаются разделить набор данных на части, чтобы обнаружить значимые кластеры или группы, существующие внутри них. Хороший метод кластеризации позволяет создавать кластеры высокого качества, в которых сходство внутри кластера высокое, а сходство между кластерами низкое. После формирования групп среднее значение рейтинга пользователей, принадлежащих к одной и той же группе, применяется к отдельным пользователям. В ситуациях, когда пользователь может принадлежать к нескольким разным кластерам, рекомендации даются на основе среднего уровня участия пользователя в кластерах. Алгоритм k-средних (k-means) и алгоритм самоорга-

низованной карты (SOM) являются двумя наиболее часто используемыми методами кластеризации. Метод k-средних принимает входной параметр, а затем разбивает набор из n элементов на k кластеров. SOM — это метод обучения без учителя, основанный на методе искусственной кластеризации нейронов. Методы кластеризации можно использовать для уменьшения количества кластеров-кандидатов в алгоритмах, основанных на сотрудничестве.

Модель SVM (Support Vector Machine)

Модель машины опорных векторов SVM [12] представляет собой модель классификации двоичных данных. Она может быть расширена до многоклассовой классификации, в которой каждая точка данных x_i представлена как вектор n -мерного пространства. SVM работает по принципу построения гиперплоскости формы $W^T X = C$, которая действует как граница между двумя известными группами точек данных. Гиперплоскость выбирается так, чтобы расстояние от точек данных, принадлежащих двум классам, до нее было как можно большим, это помогает уменьшить ошибку классификации модели SVM. На основе определенной гиперплоскости модель SVM определит класс новой точки данных в зависимости от ее положения выше или ниже гиперплоскости.

Применение модели SVM для коллаборативной фильтрации [12] помогает разделить продукты, которые могут понравиться или не понравиться пользователям. Этот уровень предпочтения прогнозируется на основе положения и расстояния от точки данных, представляющей продукт, до гиперплоскости. Так формируется рекомендация по продукту текущему пользователю. По оценкам, эффективность модели SVM [12] для совместной фильтрации зависит от метода представления точек данных и метода оптимизации параметров при определении гиперплоскости.

Модель дерева решений

Дерево решений [13] представляет собой структуру принятия решений в виде древовидного графа. Поскольку входные данные представляют собой предварительно помеченный набор обучающих данных, дерево решений проанали-

зирует и предоставит классификационную метку для нового примера. Применение деревьев решений для прогнозирования текущих рейтингов пользователей для неизвестных продуктов оказалось более интуитивно понятным и простым для понимания, чем другие методы классификации, такие как Support Vector Machine (SVM) и Neural Networks [13].

Проблемы при применении метода совместной фильтрации

Коллаборативная фильтрация сталкивается со следующими проблемами:

1. проблема холодного старта для нового пользователя;
2. проблема холодного старта для нового продукта;
3. проблема скудных данных;
4. вопрос изменения предпочтений пользователей со временем.

1.5. Фильтрация на основе знаний

Это самый сложный способ построения рекомендаций. Он предусматривает максимальную детализацию запросов потенциального с целью сбора информации о его предпочтениях. Фильтрация на основе знаний [1] в музыке означает использование информации о предпочтениях, стиле, артистах или других характеристиках, чтобы предложить пользователю персонализированные рекомендации музыкального контента.

Только после получения значений по каждому предпочтению система выдает рекомендации, которые наверняка будут представлять интерес для покупателя. Высокая эффективность этого способа сопровождается сложностью разрабатываемых алгоритмов и необходимостью удерживать внимание пользователя в течение времени, которое требуется для указания значений запрашиваемых параметров.



Рисунок 1.7 – Фильтрация на основе знаний.

По сравнению с подходами, основанными на коллаборативной фильтрации и фильтрации на основе контента, рекомендации, основанные на знаниях, в основном не зависят от оценки объектов или их описания с помощью метаданных, а на более глубоких правилах для выявления объектов интереса. Иногда предыдущий подход (основанный на контенте) определяется как частный случай основанного на знаниях, где в качестве знаний выступает информация об объектах интереса, но из-за большой распространенности систем на основе контента последние обычно выносят в отдельный тип. Дополнительные знания позволяют рекомендовать объекты, не полагаясь на «похожесть» чего-либо, а использовать более сложные условия.

Рекомендации, основанная на знаниях (рисунок 1.7), опираются на следующие входные данные: множество правил (ограничений) или метрик схожести и множество объектов интереса. В зависимости от заданных требований пользователя, правила описывают, какие объекты должны быть рекомендованы. Текущий пользователь формулирует свои предпочтения в терминах свойств элемента, которые, в свою очередь, представляются с точки зрения правил (ограничений).

В качестве плюсов можно отметить возможность исключения рекомендаций уже не актуальных для данного пользователя объектов, минусы - высокая сложность построения и сбора данных.

1.6. Гибридные системы

Каждая из описанных выше рекомендательных систем имеет как ярко выраженные плюсы, так и не менее существенные минусы. Именно поэтому наибольшее распространение получили гибридные наборы алгоритмов [1], представляющие собой комбинацию из разных способов выдачи рекомендаций.

При таком подходе к решению задачи особенно актуальным становится баланс между ними. Помимо трех основных подходов к фильтрации, используется гибридный подход, который объединяет возможности базовых типов. Использование гибридных алгоритмов позволяет достичь более высокой точности.

Существует семь основных методов гибридизации (hybridization techniques).

1. Взвешенные (Weighted): баллы различных компонентов рекомендаций умножаются на весовые коэффициенты и суммируются.
2. Переключение (Switching): система выбирает между предлагаемыми компонентами и применяет выбранную систему.
3. Смешанный (Mixed): Рекомендации разных рекомендателей представлены вместе, чтобы составить рекомендацию.
4. Комбинация функций (Feature Combination): функции, взятые из разных источников знаний, объединяются и представляются в едином алгоритме рекомендаций.
5. Расширение функций (Feature Augmentation): рекомендательный метод используется для вычисления объекта или набора функций, который затем является частью входных данных для следующего метода.
6. Каскад (Cascade): рекомендации строго расставлены по приоритетам, при этом более низкие приоритеты разрывают связь с более высокими.
7. Метауровень (Meta-level): применяется рекомендательный метод и создается некоторый тип модели, которая затем является входными данными, используемыми следующим методом.

1.7. Сравнительный анализ алгоритмов

Три основных подхода к построению рекомендательных систем используют разную базовую входную информацию и имеют различные сильные и слабые стороны, которые приведены в таблице 1.1.

Для реализации коллаборативной фильтрации (CF), а также фильтрации на основе контента (CBF) необходима только базовая информация о предмете, например, его название или другие атрибуты, тогда как решения, основанные на знаниях, требуют более детальной информации о свойствах предмета (а во многих случаях также дополнительных условий и ограничений). CF и CBF более адаптивны в том понимании, что вновь внесенные пользователем оценки автоматически учитываются при будущих запусках алгоритма рекомендации. Напротив, правила в рекомендательных системах, основанных на знаниях необходимо каждый раз адаптировать вручную под вновь внесенные данные.

Свойство интуитивности можно интерпретировать как некую случайность нахождения релевантных объектов, даже когда пользователь не инициировал соответствующего поиска. Такой эффект, в первую очередь, может быть достигнут при использовании подходов коллаборативной фильтрации. В связи с тем, что фильтрация по содержимому не учитывает предпочтений других пользователей, подобное свойство не может быть достигнуто. Подобный эффект для систем, основанных на знаниях, в принципе возможен, однако, его возможность достижения сильно зависит от навыков инженера знаний (который может предвидеть такие свойства при создании рекомендательных правил).

Термин «проблема холодного старта» относится к ситуации, когда возникает необходимость предоставления начальных оценок до того, как алгоритм сможет определить релевантные рекомендации. Такая проблема характерна как для алгоритмов коллаборативной фильтрации, так и для рекомендаций на основе содержимого. Пользователи при коллаборативном подходе должны оценивать набор элементов, прежде чем алгоритм сможет определить ближайших соседей. При использовании рекомендательных алгоритмов на основе контента пользова-

тель также должен указать интересные ему объекты до того момента, когда алгоритм будет способен определять элементы, похожие на уже оцененные пользователем.

Наконец, свойство прозрачности определяет степень того, насколько понятно можно обосновать результат работы рекомендательных алгоритмов для пользователей. Подобные интерпретации результатов в системах коллаборативной фильтрации полагаются исключительно на механизм “ближайших соседей”, то есть, пользователи, которые интересовались объектом X , также интересуются объектом Y . Алгоритмы, основанные на фильтрации содержимого объясняют свои рекомендации в терминах схожести рекомендуемого элемента с объектами, которыми интересовался пользователь: мы рекомендуем Y , поскольку вы интересовались X , который очень похож на Y . В отличие от прошлых методов, подходы, основанные на глубоких знаниях готовы предоставить развернутые пояснения, которые учитывают семантически знания о предметах. Пример такого пояснения - вывод, объясняющий причины того, почему определенный набор требований не позволяют получить конкретный объект в качестве выдачи рекомендательного алгоритма.

Как правило, алгоритмы коллаборативной фильтрации и фильтрации, основанной на содержимом, используются для рекомендации контента низкой степени участия такого как фильмы, книги и новостные статьи. Под низкой степенью участия будем понимать то, что вклад неправильной рекомендации довольно низок, поэтому пользователи прикладывают меньше усилий к оценке предмета. Напротив, системы, основанные на знаниях, обычно используются для рекомендаций высокой степени участия таких как финансовые услуги, автомобили и квартиры. В последнем случае, оценки выставляются с низкой частотой, что делает эти предметные области менее доступными для первых двух подходов. Например, пользовательские предпочтения относительно автомобилей могут значительно измениться в течение нескольких лет, при этом они не будут обнаруженными рекомендательной системой. В то же время, такие сдвиги предпочтений определяются первыми двумя подходами, в связи с тем, что покупки в

их предметных областях происходят чаще и, как следствие, соответствующие рейтинги доступны для вынесения рекомендаций.

Таблица 1.1 — Сравнительный анализ рекомендательных подходов

Подход	Коллаборативная фильтрация	Фильтрация на основе контента	Фильтрация на основе знаний
Быстрое развертывание	Да	Да	Нет
Адаптивность	Да	Да	Нет
Интуитивность	Да	Нет	Нет
Холодный старт	Да	Да	Нет
Прозрачность	Нет	Нет	Да
Высокая степень участия	Нет	Нет	Да

Вывод

В результате исследования можно сделать вывод, что коллаборативная фильтрация в контексте музыкального онлайн-проигрывателя является эффективным методом для предоставления персонализированных рекомендаций пользователям. Анализ взаимодействий между пользователями и музыкальными треками позволит выявить схожие музыкальные предпочтения, что в свою очередь повысит уровень удовлетворенности пользователей от предлагаемого контента.

2. Конструкторский раздел

2.1. Концептуальное проектирование

Концептуальное проектирование базы данных заключается в выявлении всех сущностей и их взаимосвязей в разрабатываемой системе.

Посетители сайта имеют возможность просматривать треки, слушать их, а также, используя необходимые фильтры, искать конкретные треки. У каждого трека есть информация о жанре. Помимо просмотра и прослушивания треков и их поиска авторизованные пользователи могут оценивать треки и на основе выставленных оценок получать рекомендации.

На основе этих данных было выделено 6 сущностей, 6 связей типа «один ко многим» и была построена ER-диаграмма, представленная на рисунке 2.1.

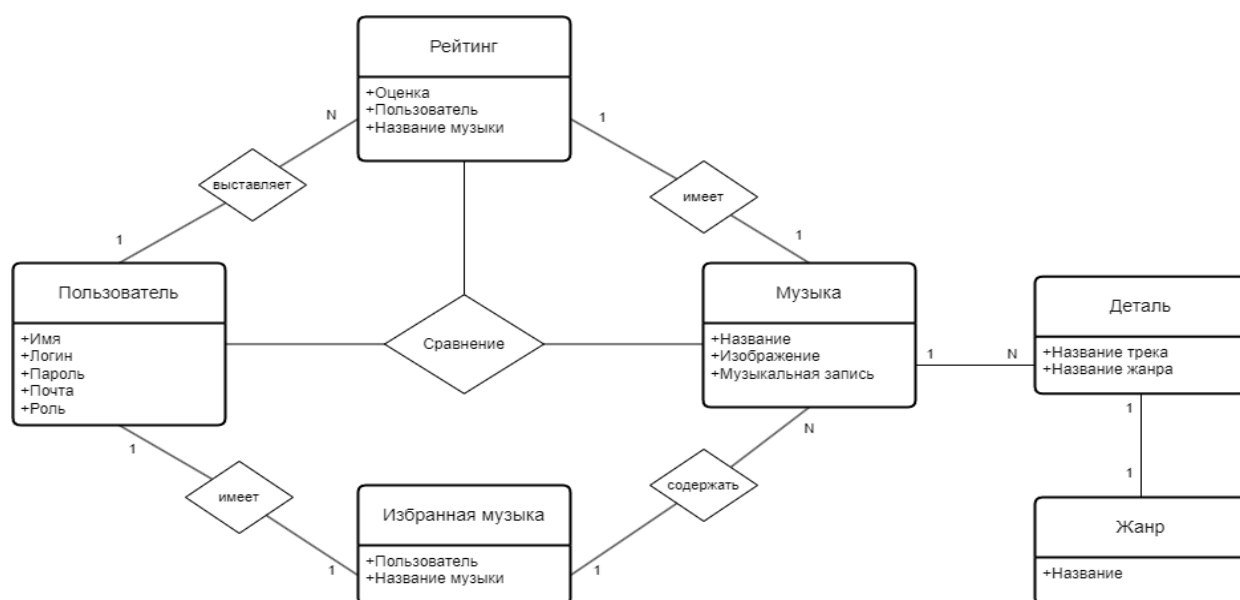


Рисунок 2.1 – ER-диаграмма базы данных.

2.2. Логическое проектирование

На рисунке 2.2 представлена диаграмма реляционной базы данных, разработанной для проектируемой системы.

База данных предназначена для хранения всей долговременной информации системы: зарегистрированные пользователи, список их любимых музыкальных записей, их оценки треков, информация о музыкальных записях, справочники жанров.

Эта база данных является реляционной и реализована при помощи системы управления базами данных SQLite.

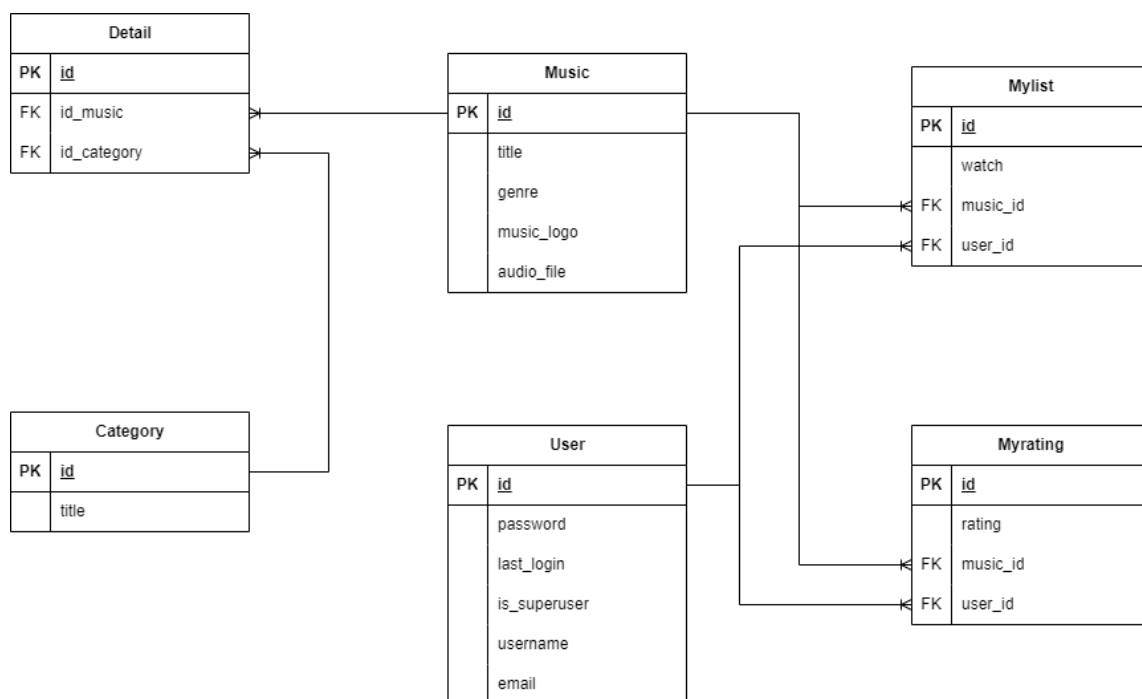


Рисунок 2.2 – Диаграмма базы данных.

Описание назначения и свойств полей базы данных приведено в таблицах 2.1—2.6.

В данной таблице хранятся данные пользователя – логин (username), фамилия (first_name), имя (last_name), почтовый адрес (email), пароль (password), дата последнего логина (last_login), роль (role). Логин должен быть уникальным.

Таблица 2.1 – Пользователи системы

	Тип данных	Значение по умолчанию	Обязательность	Ключ Первичный	Внешний ключ	Ограничения
id	int	-	+	+	-	-
username	varchar (150)	-	+	-	-	уникальный
first_name	varchar (30)	-	+	-	-	-
last_name	varchar (150)	-	-	-	-	-

Продолжение таблицы 2.1

password	varchar (128)	-	+	-	-	-
email	varchar (254)	-	+	-	-	-
last_login	datetime	-	-	-	-	-
role	bool	+	+	-	-	-

В данной таблице хранятся все данные о рейтингах – идентификатор пользователя, поставившего рейтинг (user_id), идентификатор музыкального трека, которому была выставлена рейтинг (music_id) и сам рейтинг (rating), который может иметь натуральное значение от 1 до 5.

Таблица 2.2 – Список рейтинга

	Тип данных	Значение по умолчанию	Обязательность	Ключ Первичный	Внешний ключ	Ограничения
id	int	-	+	+	-	-
rating	int	+	+	-	-	{ 1,2,3,4,5 }
music_id	int	-	+	-	+	-
user_id	int	-	+	-	+	-

Здесь хранятся данные о музыкальных треках – название музыкального трека (title), его жанр (genre), его изображение (music_logo) и музыкальная запись (audio_video).

Таблица 2.3 – Музыкальные треки

	Тип данных	Значение по умолчанию	Обязательность	Ключ Первичный	Внешний ключ	Ограничения
id	int	-	+	+	-	-
title	varchar (200)	-	+	-	-	-
genre	varchar (100)	-	+	-	-	-
music_logo	varchar (100)	-	+	-	-	-
audio_file	varchar (100)	-	+	-	-	-

В таблице 2.4 хранятся все данные о взаимодействии между треком и жанром – идентификатор музыкального трека (id_music), идентификатор жанры (id_category).

Таблица 2.4 – Взаимодействие между треком и жанром.

	Тип данных	Значение по умолчанию	Обязательность	Ключ Первичный	Внешний ключ	Ограничения
id	int	-	+	+	-	-
id_music	int	-	+	-	-	-
id_category	int	-	+	-	-	-

Данная таблица представляет собой справочник, содержащий названия жанров (title).

Таблица 2.5 – Жанры музыкального трека.

	Тип данных	Значение по умолчанию	Обязательность	Ключ Первичный	Внешний ключ	Ограничения
id	int	-	+	+	-	-
title	varchar (200)	-	+	-	-	-

В данной таблице хранятся любимый музыкальный список пользователя. Таблица содержит информации как идентификатор пользователя, добавившего музыкальный трек (user_id), идентификатор музыкального трека, которую пользователь добавил в списке (music_id).

Таблица 2.6 – Любимый музыкальный список пользователя.

	Тип данных	Значение по умолчанию	Обязательность	Ключ Первичный	Внешний ключ	Ограничения
id	int	-	+	+	-	-
watch	bool	-	+	-	-	-
music_id	int	-	+	-	+	-
user_id	int	-	+	-	+	-

2.3. Диаграмма прецедентов

Диаграмма прецедентов, или диаграмма вариантов использования (use-case diagram), отображает взаимосвязи между пользователями системы и

прецедентами (функциональными возможностями системы), предоставляя обзор функционального поведения системы.

На рисунке 2.3 представлена диаграмма, описывающая типы пользователей и их взаимодействие с системой.

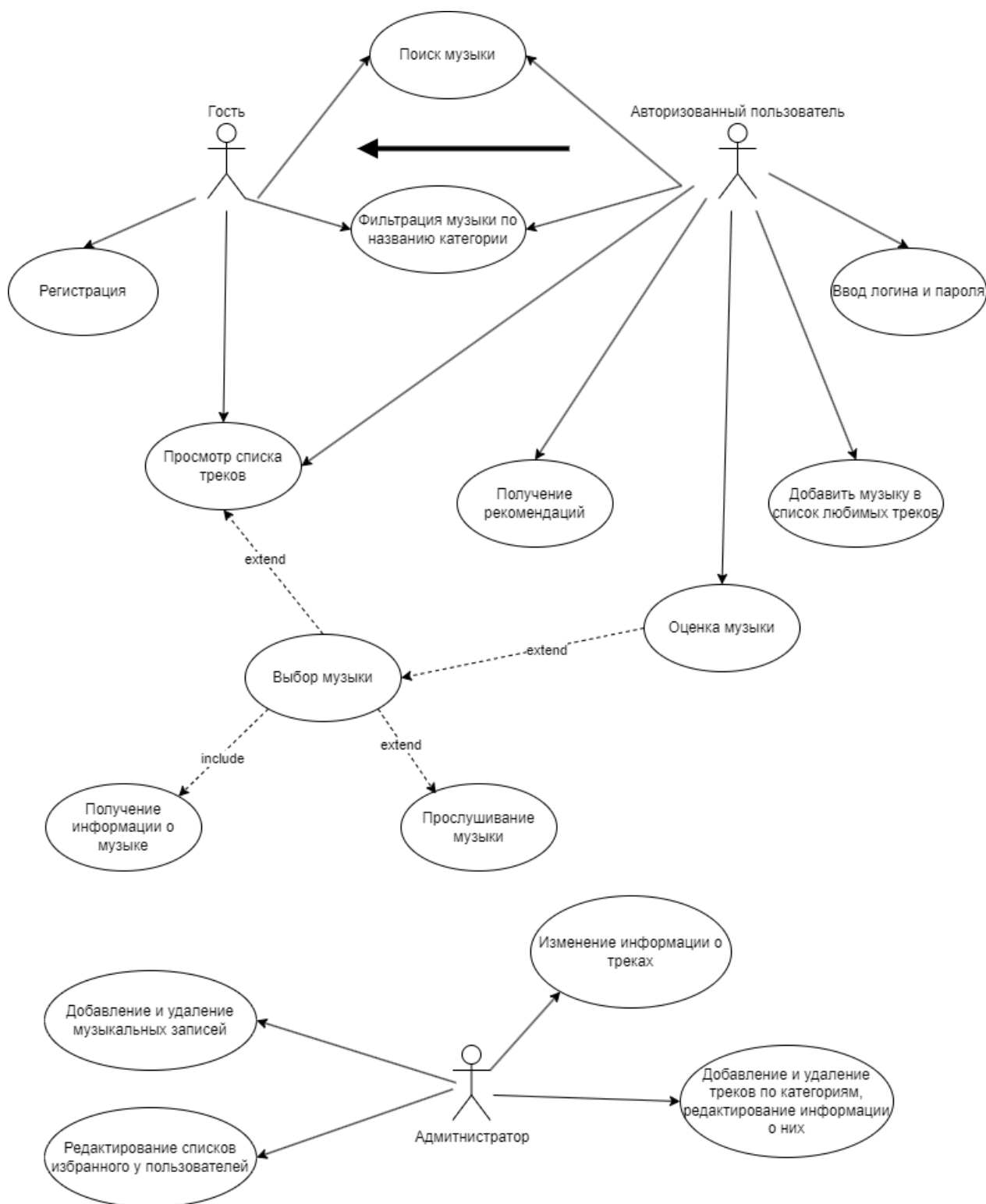


Рисунок 2.3 – Use-Case диаграмма

Для выполнения здесь надо указать какое-то действие необходима авторизация пользователей. Для управления библиотекой введена роль администратора.

Таблица 2.7 – Типы пользователей и доступный им функционал.

Тип пользова- теля	Функционал
Гость	Регистрация; поиск музыкальных записей; фильтр музыкальных треков по жанру; просмотр списка всех музыкальных треков.
Авторизованный	Ввод логина и пароля; поиск музыкальных записей; фильтр музыкальных треков по жанру; просмотр списка всех музыкальных треков; оценка музыкального трека; добавление музыкального трека в список любимой музыки; получение рекомендаций.
Администратор	Ввод логина и пароля; поиск музыкальных записей; фильтр музыкальных треков по жанру; просмотр списка всех музыкальных треков; оценка музыкального трека; добавление музыкального трека в список любимой музыки; получение рекомендаций; удаление, добавление, изменение информации о музыкальном треке; удаление, добавление, изменение музыкального трека по жанру; удаление, добавление, изменение музыкального трека в списке избранных всех пользователей.

Регистрация

Для получения доступа к дополнительным функциям сайта, пользователь может зарегистрироваться, предоставив необходимую информацию о себе.

Главная последовательность: На странице регистрации пользователь вводит своё имя, логин, адрес электронной почты и пароль. Для проверки корректности введенной информации, пользователь повторно вводит адрес электронной почты и пароль в соответствующих полях формы. После этого он подтверждает свой ввод. Система переводит пользователя на главную страницу.

Альтернативная последовательность: При попытке ввода уже существующего логина или адреса электронной почты система выдаёт сообщение о том, что указанный логин или почта уже заняты. Пользователю предлагается повторить ввод с другими учетными данными.

Ввод логина и пароля

Если посетитель уже имеет профиль на сайте, то он может авторизоваться на нем, указав свои логин и пароль.

Главная последовательность: Пользователь вводит логин и пароль в форме на странице авторизации и подтверждает ввод. Система переводит пользователя на главную страницу.

Альтернативная последовательность: Неверный ввод логина или пароля. Система выдает сообщение о том, что логин или пароль введены неверно, и предлагает повторить ввод.

Просмотр музыкальных треков

На главной странице сайта представлен каталог всех музыкальных треков, существующих в базе данных. Пользователь может просматривать их, используя полосу прокрутки.

Фильтрация музыкальных треков

Если у посетителя сайта нет желания просматривать все существующие музыкальные треки, он может выставить необходимые фильтры по жанру и просмотреть только те музыкальные треки, которые удовлетворяют им.

Главная последовательность: На главной странице сайта пользователь по желанию выбирает один жанр и подтверждает ввод. После этого страница перезагружается и выводятся музыкальные треки, соответствующие выставленному фильтру.

Поиск музыкального трека

Если у посетителя сайта нет желания просматривать все существующие музыкальные треки, он может найти музыкальный трек, введя в форму несколько символов или точное название музыкального трека.

Главная последовательность: На главной странице сайта пользователь по желанию вводит несколько символов или точное название музыкального трека и подтверждает ввод. После этого страница перезагружается и выводятся соответствующие музыкальный трек.

Выбор музыкального трека, получение информации об его и прослушивание его.

При просмотре музыкального трека посетитель сайта может выбрать заинтересовавший его музыкальный трек, кликнув на него левой кнопкой мыши, и тем самым перейти на посвященную ему страницу.

На странице о музыкальном треке может присутствовать информация о его жанр и прослушивание о музыкальном записи.

Главная последовательность: Пользователь выбирает музыкальный трек и автоматически переходит на страницу музыкального трека, содержащую более подробную информацию.

Альтернативная последовательность: Если пользователь не авторизован, то только видит обложку музыкального трека, его название и его плеер.

Оценка музыкального трека:

Авторизованным посетителям сайта предоставляется возможность оценивать музыкальный трек на соответствующих им страницах.

Главная последовательность: Пользователь выставляет оценку музыкальному треку от одного до пяти посредством выбора количества звездочек, расположенных внизу страницы музыкального трека.

Альтернативная последовательность: Если пользователь не авторизован, то система выводит сообщение о невозможности голосования и предлагает посетителю пройти авторизацию.

Получение рекомендуемого музыкального трека:

Авторизованный на сайте пользователь может получить музыкальные треки, которые по предположению рекомендательной системы сайта могут быть ему интересны.

Главная последовательность: Пользователь нажимает кнопку «Get Music Recommendation» на главной странице сайта. Страница перезагружается и выводятся рекомендуемые ему музыкальные треки.

Альтернативная последовательность 1: Пользователь не авторизован. Система выводит сообщение о невозможности получения рекомендаций и предлагает пройти авторизацию.

Альтернативная последовательность 2: Если пользователь еще не оценил ни один музыкальный трек, то страница выводится 10 музыкальных треков с самыми высокими средними оценками.

2.4. Описание входных и выходных данных

Входными данными для программы являются список пользователей, список музыкальных записей, пользовательские оценки музыкальной записи.

Список пользователей содержит в себе:

- идентификатор пользователя;
- логин;
- фамилия;
- имя;
- почтовый адрес;
- пароль;
- дата последнего логина;
- роль.

Список музыкальных записей содержит в себе:

- название музыкального трека;
- жанр;
- изображение;
- музыкальный плеер.

Пользовательские оценки музыкальной записи содержит в себе:

- идентификатор пользователя;
- идентификатор музыкального трека;
- рейтинг.

Функциональная модель (IDEF0-диаграмма) разрабатываемой системы представлена на рисунках 2.4 – 2.6.

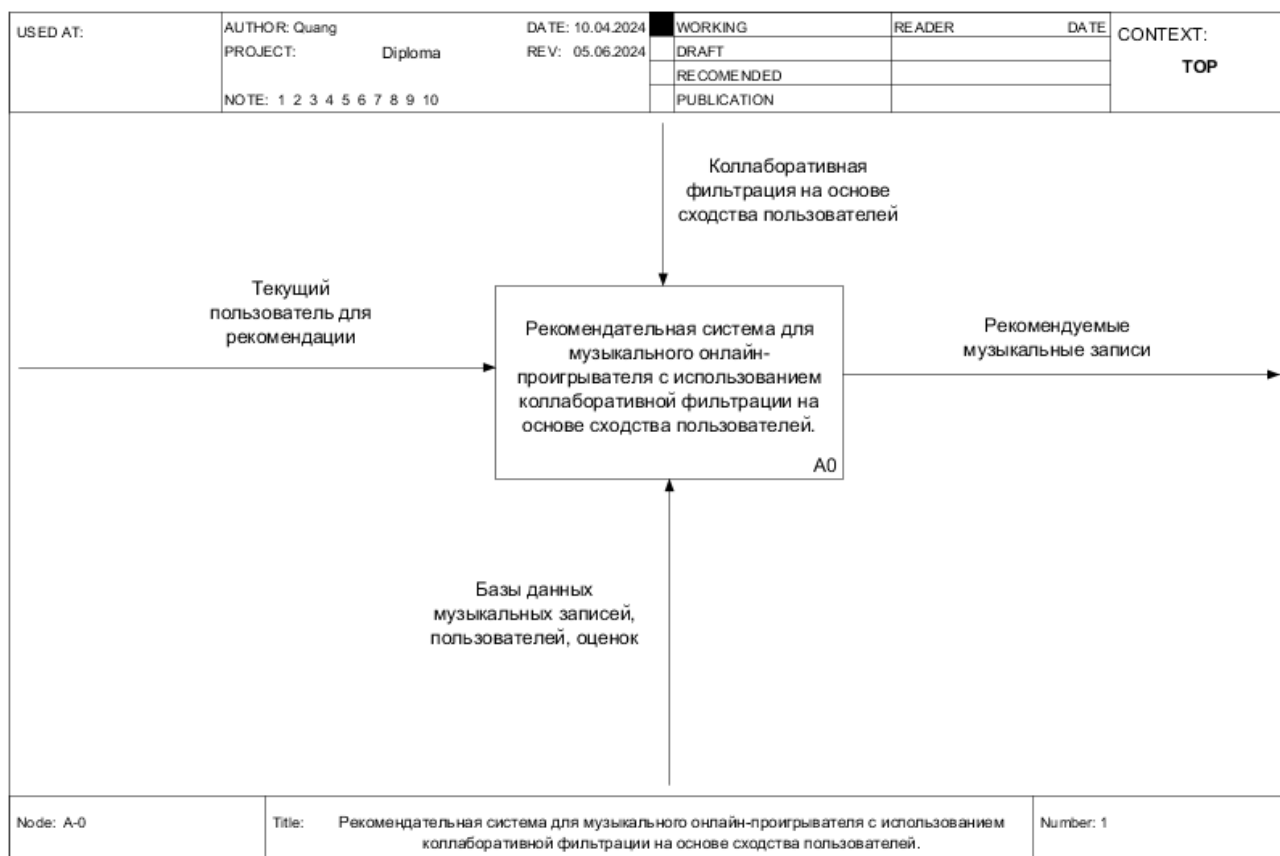


Рисунок 2.4 – Диаграмма системы

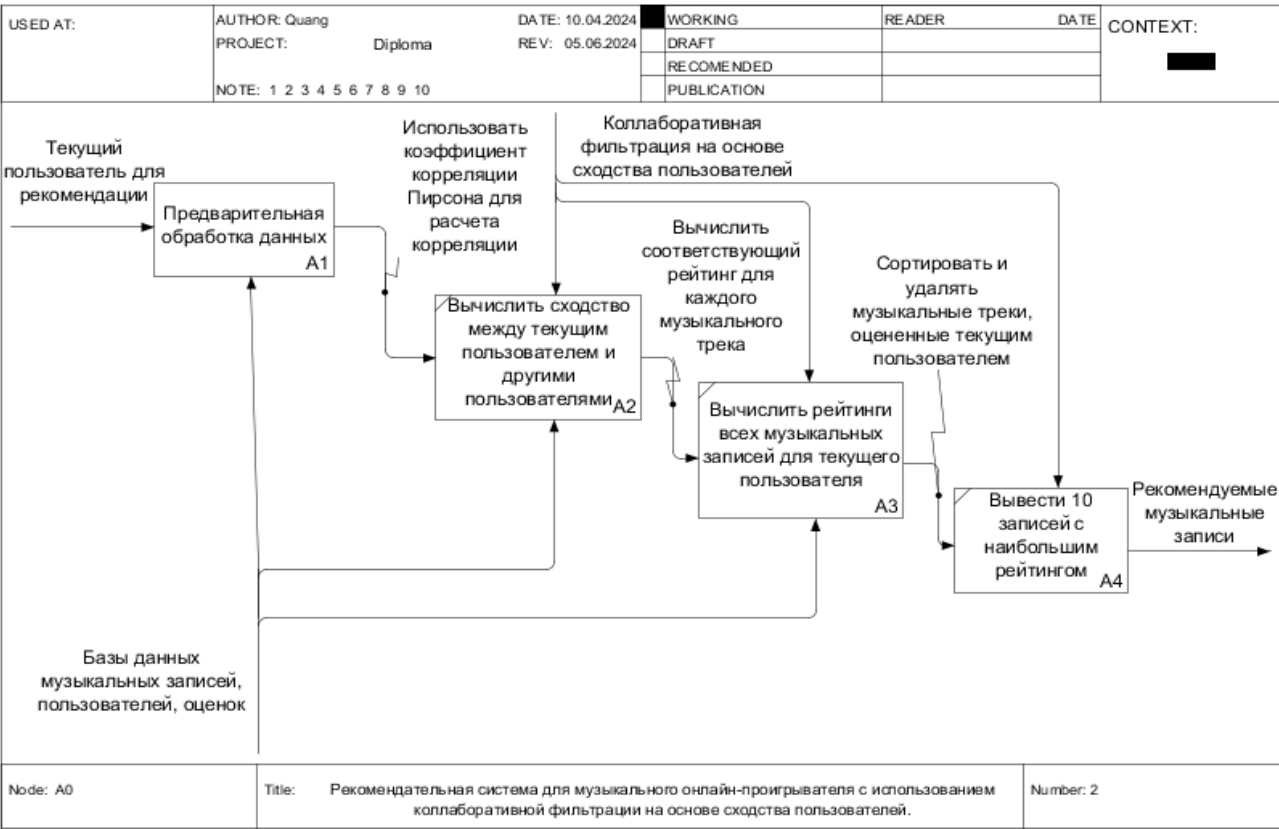


Рисунок 2.5 – Декомпозиция работы «Рекомендательная система для музыкального онлайн-проигрывателя с использованием коллаборативной фильтрации на основе сходства пользователей.»

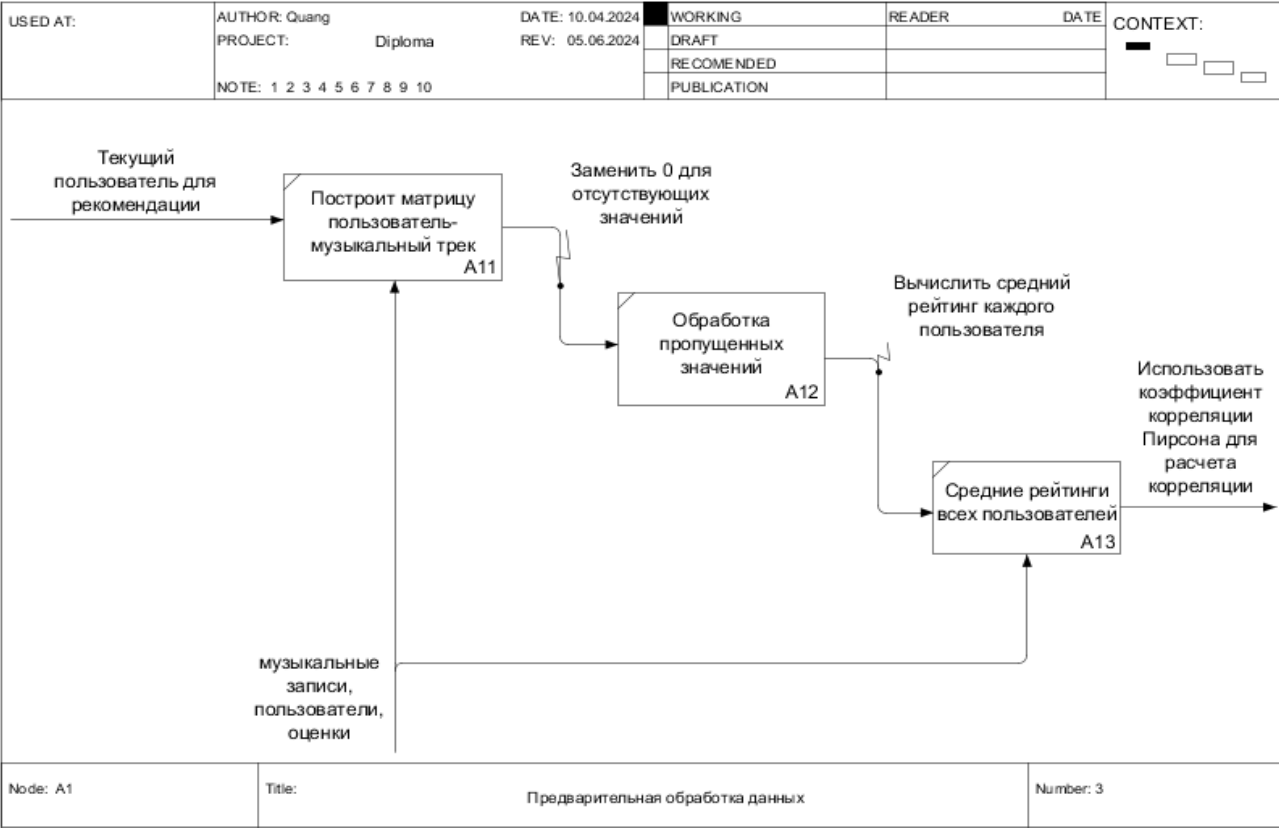


Рисунок 2.6 – Декомпозиция работы «Предварительная обработка данных»

2.5. Разработка фильтров

Для помощи пользователям в ориентировании в большом количестве музыкальных треков были разработаны фильтры, которые позволяют искать музыкальные треки определенного жанра. Алгоритм фильтрации приведен на рисунке 2.7.

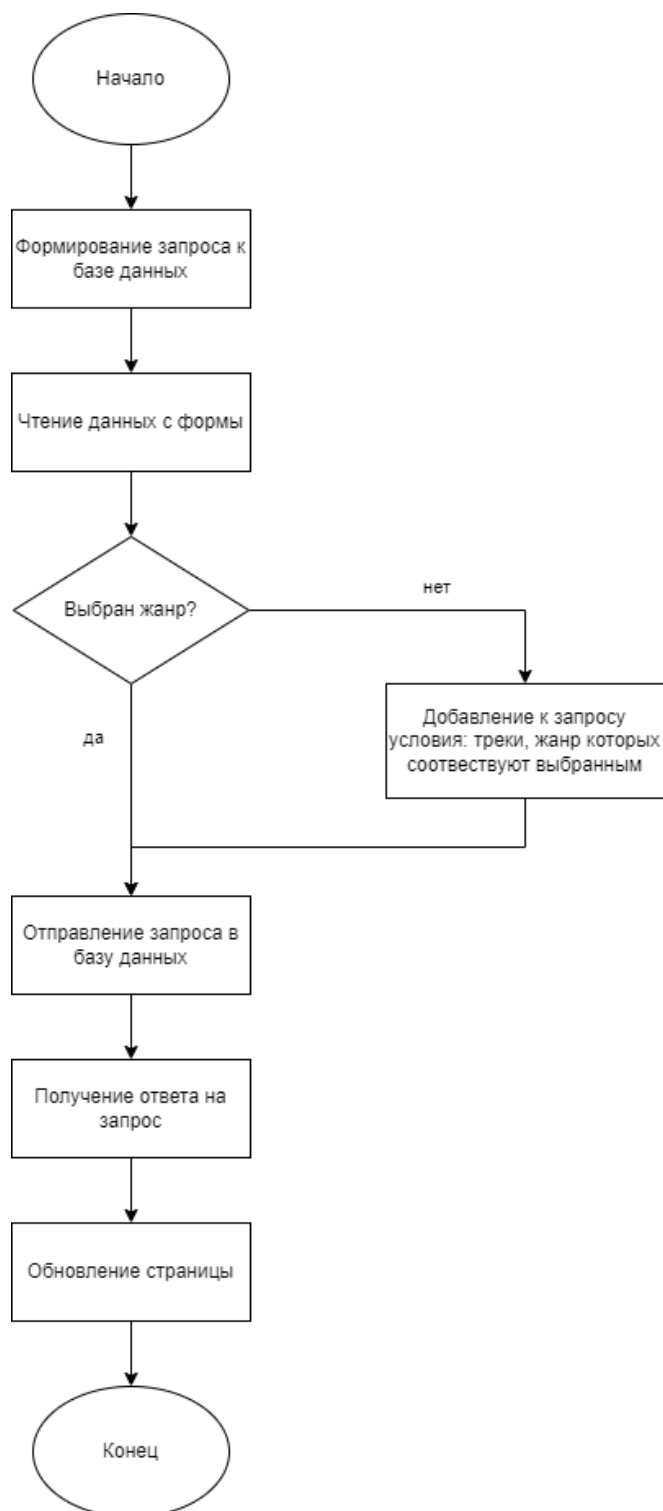


Рисунок 2.7 – Схема алгоритма фильтрации

Алгоритм фильтрации начинается с запроса к базе данных для выборки всех имеющихся музыкальных треков. Затем данные считываются с формы, определяя фильтры, заданные пользователем. Если был выбран один жанр, то к запросу добавляется условие выборки, при котором жанр музыкальных треков должен соответствовать жанру, отмеченному пользователем. После проверки всех условий запрос передается в базу данных, после получения ответа страница обновляется. В результате отображаются музыкальные треки, соответствующие заданным фильтрам пользователя.

2.6. Описание алгоритма коллаборативной фильтрации на основе сходства пользователей

Основная характеристика коллаборативной фильтрации заключается в том, что она часто использует все доступные данные для прогнозирования рейтинга определенного пользователя о новом музыкальном треке. Благодаря тому преимуществу, что он имеет возможность помещать новые данные непосредственно в таблицу данных, он добился довольно большого успеха при применении в практических приложениях. Поэтому эти методы часто делают более точные прогнозы в онлайн-системах, где новые данные всегда обновляются.

Обычно существует два подхода к коллаборативной фильтрации на основе моделей: на основе пользователей (user-based) — прогнозирование на основе сходства между пользователями — и на основе элементов (item-based) — прогнозирование на основе сходства между музыкальными треками. Системы, основанные на пользователях (user-based), определяют сходство между двумя пользователями путем сравнения их оценок одного и того же музыкального трека, а затем прогнозируют рейтинг музыкального трека i пользователем u или средний рейтинг пользователей, похожих на пользователя u . Сходство между пользователем u и пользователем u' можно рассчитать с помощью корреляции Пирсона.

$$sim_{pearson}(u, u') = \frac{\sum_{i \in I_{uu'}} (r_{ui} - \bar{r}_u)(r_{u'i} - \bar{r}_{u'})}{\sqrt{\sum_{i \in I_{uu'}} (r_{ui} - \bar{r}_u)^2 \sum_{i \in I_{uu'}} (r_{u'i} - \bar{r}_{u'})^2}}, \quad (2.1)$$

где:

$I_{uu'}$ – список музыкальных треков, оцениваемых u и u' ;

\bar{r}_u – средняя оценка по всем музыкальным трекам пользователем u ;

$\bar{r}_{u'}$ – средняя оценка по всем музыкальным трекам пользователем u' .

Также можно использовать ранговую корреляцию Спирмена [16] (Spearman's rank correlation) или среднеквадратичную разницу (mean squared difference), но в работе [18] показано, что для систем, основанных на пользователях, вычисление сходства с помощью корреляции Пирсона лучше, чем другие методы.

Предоставление вычисленных прогнозов или предложений — важный шаг в системе рекомендаций по коллаборативной фильтрации. После расчета сходства между пользователями мы можем спрогнозировать рейтинг пользователя u для музыкального трека i по следующей формуле:

$$\widehat{r}_{ui} = \bar{r}_u + \frac{\sum_{u' \in K_u} sim(u, u') \cdot (r_{u'i} - \bar{r}_{u'})}{\sum_{u' \in K_u} |sim(u, u')|}, \quad (2.2)$$

где:

\widehat{r}_{ui} – прогноз для пользователя u по музыкальному треку i ;

$sim(u, u')$ – сходство между пользователем u и u' ;

K_u – список пользователей, близких к пользователю u .

Представим алгоритм коллаборативной фильтрации на основе пользователей, использующей меру сходства Пирсона на псевдоязыке, чтобы предсказать оценку пользователя u для музыкального трека i следующим образом:

Листинг 2.1 – Описание алгоритма

```

1: procedure User-based-CF
2: for u=1 to N do
3:   Вычислить get_user_avg_rating(u)
4: end for

```

Продолжение листинга 2.1

```
5: u = active_user // Идентификатор текущего пользователя
6: for u'=1 to N do
7:   Вычислить  $\text{sim}(u, u')$  по формуле (2.1)
8: end for
9: for i = 1 to M do
10:  Вычислить по формуле (2.2)
11: end for
12: end procedure
```

2.7. Разработка рекомендательной системы

В качестве метода построения рекомендательной системы был выбран алгоритм коллаборативной фильтрации на основе сходства пользователей. Схема алгоритма рекомендательной системы приведена на рисунке 2.7.

Работа алгоритм рекомендательной системы начинается с проверки авторизации пользователя. Если он не авторизован, то он не может получить рекомендации и перейдет на страницу ввода логина. Далее формируются запросы в базу данных на получение информации о пользователях, музыкальных треках и оценках. Полученная информация сохраняется в виде одномерных или двумерных массивов. В массиве с оценками текущего пользователя индекс представляет собой идентификатор музыкального трека, а значение – оценку, выставленную этому музыкальному треку. Аналогичным образом устроен массив для оценок остальных пользователей, но в этом случае он будет двумерным. В нем каждая строка соответствует какому-либо пользователю, каждый столбец соответствует какому-либо музыкальному треку и значение для каждой строки и столбца – это рейтинг музыкального трека, оценённый пользователем. В массиве неопределённое значение заменяется на 0. Затем мы вычисляем средний рейтинг каждого пользователя.

После получения из базы необходимых данных, осуществляется проверка оценок текущего пользователя. Если им не было оценено ни одного трека, то страница выводится 10 музыкальных треков с самыми высокими средними оценками.

На следующем этапе определяется коэффициент сходства между текущим пользователем и другими пользователями на основе их взаимодействия с музыкальными треками. Потом по этому коэффициенту корреляции и средним оценкам пользователей мы можем вычислить подходящий рейтинг каждого музыкального трека для текущего пользователя. Из списка музыкальных треков, отсортированного по соответствующему рейтингу в порядке убывания, мы удаляем музыкальные треки, оцененные текущим пользователем.

После всех вычислений полученные прогнозы заносятся в специальную таблицу, отправляется запрос на получение музыкальных треков, отсортированных в порядке убывания рейтинга. После получения результатов страница обновляется. В итоге пользователь видит рекомендованные ему музыкальные треки.

Вывод

В данном разделе были разработаны база данных для проектируемой системы и алгоритмы фильтрации, предложен алгоритм работы рекомендательной системы и приведены их описания.

Было выполнено концептуальное проектирование, в рамках которого определены сущности и их взаимосвязи в системе, построена ER-диаграмма.

На этапе логического проектирования, была разработана схема реляционной базы данных, включающая описание всех таблиц и атрибутов, присутствующих в ней.

Также была разработана диаграмма прецедентов, рассмотрены сценарии для каждого варианта использования системы.

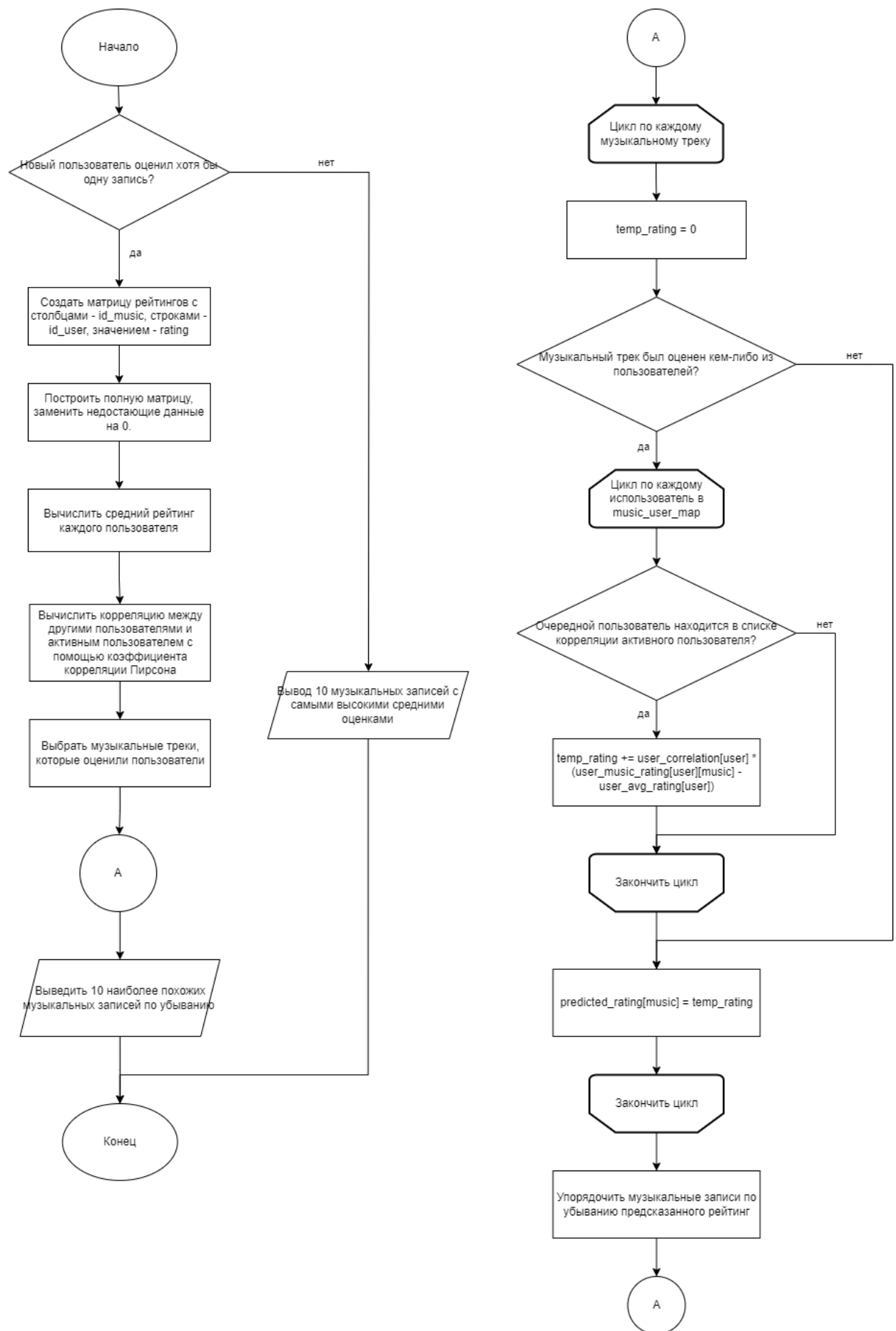


Рисунок 2.8 – Схема алгоритма рекомендательной системы

3. Технологический раздел

3.1. Выбор средств программной реализации

Для реализации рекомендательной системы для онлайн-проигрывателя с использованием коллаборативной фильтрации на основе сходства пользователей в качестве языка программирования выбран Python 3.11 [15]. Он выбран в качестве языка программирования, потому что он имеет следующие преимущества:

- язык программирования высокого уровня;
- простой синтаксис и поддержка библиотек значительно упрощают разработку приложений, использующих машинное обучение;
- существует большое количество библиотек, фреймворков и модулей с открытым исходным кодом;
- легко управляет библиотеками.

В качестве библиотеки машинного обучения выбран Scikit-learn [19]. Для разработки веб-приложения музыкального онлайн-проигрывателя в качестве выбран Django [20]. В нем веб-инфраструктура реализована на основе шаблона проектирования MVT (Model-View-Template). Model — данные из системы баз данных, View — это место, где запросы обрабатываются и возвращаются шаблоны с запрошенными данными, Template — это набор html-файлов, содержащих макет веб-сайта.

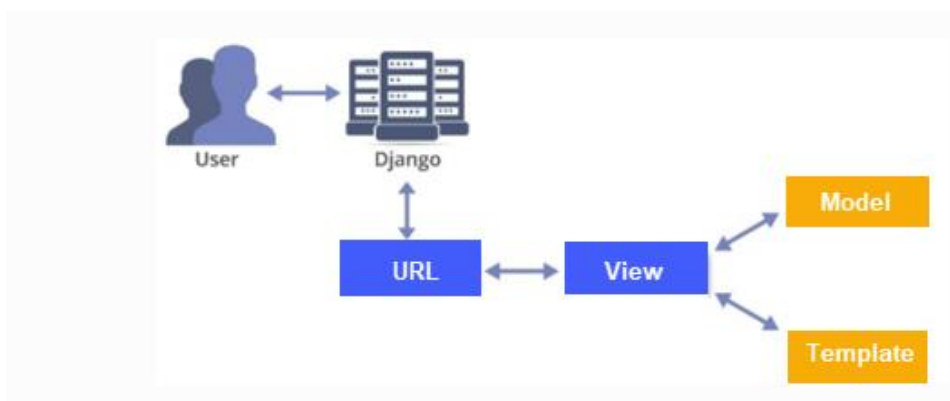


Рисунок 3.1 – Шаблон проектирования MVT

VirtualEnv используется для создания виртуальных окружений для Python программ. Файл «requirements.txt» содержит версии библиотек Python, необходимые для разработки веб-приложения музыкального онлайн-проигрывателя.

В качестве реляционной системы управления базами данных выбрана SQLite [14], которая предоставляет программный интерфейс для работы с таблицами, обеспечивая сохранение информации. SQLite представляет собой встроенную систему управления базами данных, которая может получать доступ к файлам базы данных непосредственно на диске.

3.2. Создание объектов БД

Ниже, на листинге 3.1 – 3.5, представлено создание все таблицы:

Таблица Music содержит информацию о музыкальных треках.

Листинг 3.1 – Создание таблицы Music

```
class Music(models.Model):
    title = models.CharField(max_length=200)
    genre = models.CharField(max_length=100)
    music_logo = models.FileField(upload_to='images')
    audio_file = models.FileField(upload_to='files', null=False, default=None)
    def __str__(self):
        return self.title
```

В таблице Myrating представлены пользовательские оценки музыкальных треков.

Листинг 3.2 – Создание таблицы Myrating

```
class Myrating(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    music = models.ForeignKey(Music, on_delete=models.CASCADE)
    rating = models.IntegerField(default=0, validators=[MaxValueValidator(5), MinValueValidator(0)])
```

Таблица Mylist хранит информацию о любимых музыкальных списках пользователей.

Листинг 3.3 – Создание таблицы Mylist

```
class MyList(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    music = models.ForeignKey(Music, on_delete=models.CASCADE)
    watch = models.BooleanField(default=False)
```

Таблица Category содержит информацию о жанрах.

Листинг 3.4 – Создание таблицы Category

```
class Category(models.Model):
    title = models.CharField(max_length=200)
    def __str__(self):
        return self.title
```

Таблица Detail хранит информацию о взаимодействии между музыкальным треком и жанром.

Листинг 3.5 – Создание таблицы Detail

```
class Detail(models.Model):
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    music = models.ForeignKey(Music, on_delete=models.CASCADE)
```

3.3. Реализация программного обеспечения

3.3.1. Авторизованный пользователь

Главная страница сайта

Главная страница содержит заголовочную область сайта, панель поиска по фильтрам и по конкретному названию музыкального трека и поиска рекомендаций и каталог музыкальных записей (рисунок 3.1).

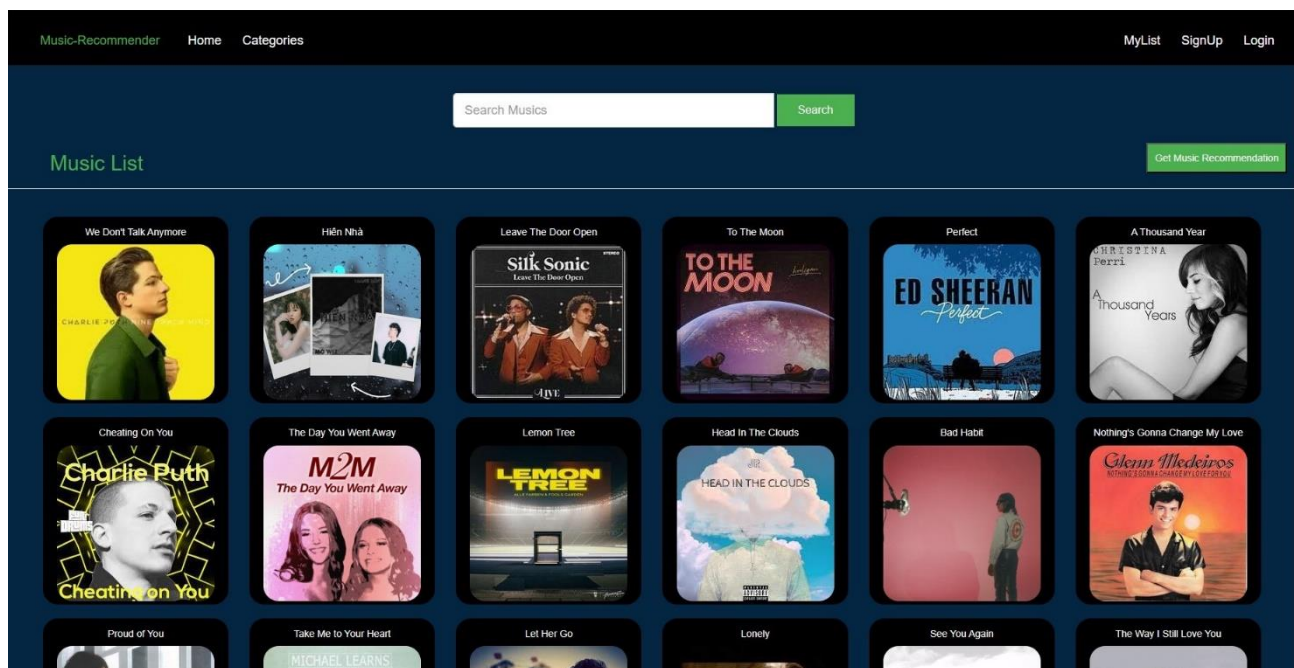


Рисунок 3.2 – Главная страница сайта

Заголовочная область сайта, которая повторяется на всех страницах, содержит логотип и кнопки «Home», «Categories», «MyList», «SignUp» и «Login». При нажатии на кнопку «Home», пользователь перенаправляется на главную страницу, независимо от текущей страницы. Кнопки «Categories», «SignUp» и «Login» переводят пользователя на соответствующие страницы поиска по фильтрам, авторизации или регистрации. При нажатии на кнопку «MyList», неавторизованный пользователь перенаправляется на страницу авторизации.

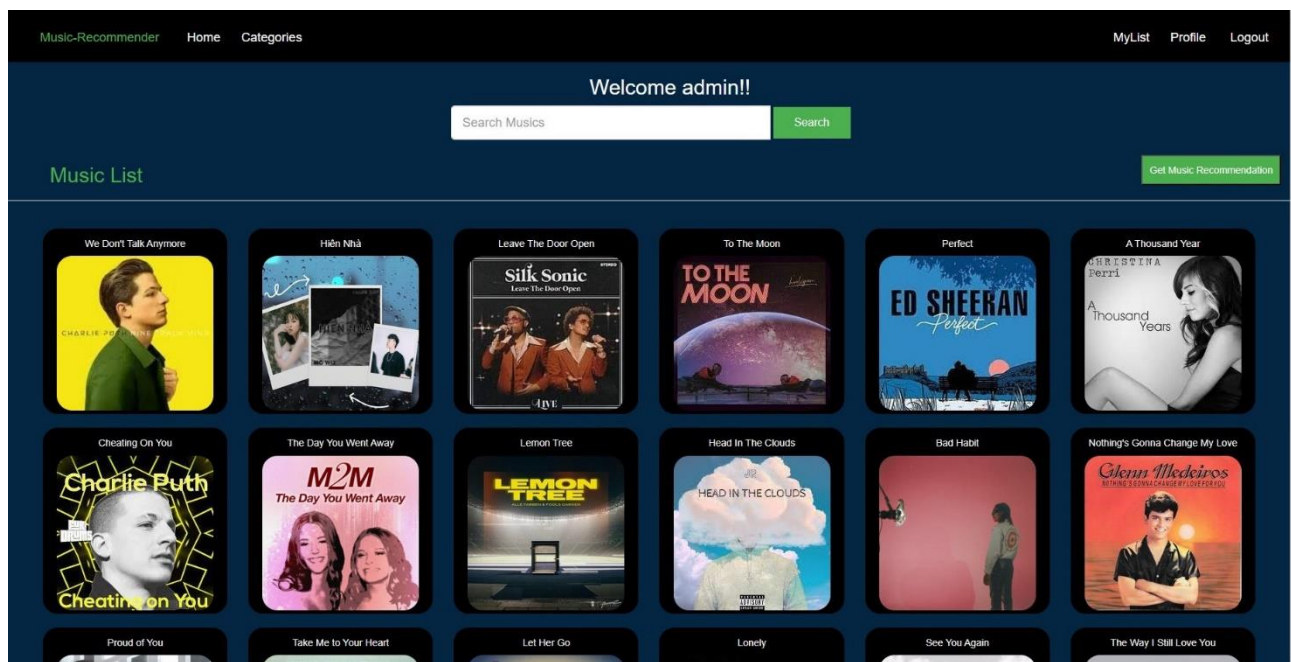


Рисунок 3.3 – Главная страница сайта после авторизации

Панель поиска по фильтрам позволяет пользователю найти музыкальные треки определенных жанров, нажав кнопку «Categories» и выбрав жанр. А панель поиска музыкального трека позволяет пользователю найти конкретное название музыкального трека. В него необходимо ввести точное название музыкальной записи или несколько символов из него и нажать на кнопку «Search». Кнопка «Get Music Recommendation» рекомендует пользователю до 10 музыкальных треков.

На сайте содержится каталог музыкальных треков, в котором представлены названия и обложки музыкальных записей. Музыкальные треки расположены в виде столбцов и строк: по шесть музыкальных треков в строке. Количество строк может быть неограниченным и зависит от количества треков в базе

данных. При нажатии на какой-либо музыкальный трек пользователь переходит на посвященную ему страницу.

Авторизация

Чтобы иметь возможность ставить оценки и получать рекомендации, посетители сайта должны нажать на иконку «Login». После этого им откроется страница авторизации, где будут предложены поля для ввода логина и пароля (рисунок 4.3). После ввода данных для завершения процесса входа на сайт необходимо нажать кнопку «Login». Затем пользователю будет открыта главная страница и доступна кнопка «Logout», при нажатии на которую можно выйти из профиля.

Рисунок 3.4 – Страница авторизации.

Регистрация

Если у посетителя еще нет своего аккаунта, то нужно нажать кнопку «SignUp». После этого появится страница с формой регистрации, где посетителю нужно ввести свои фамилия (First name), имя (Last name), логин (Username), пароль (Password) и почту (Email address) (рисунок 4.4). Для завершения регистрации необходимо нажать кнопку «Submit».

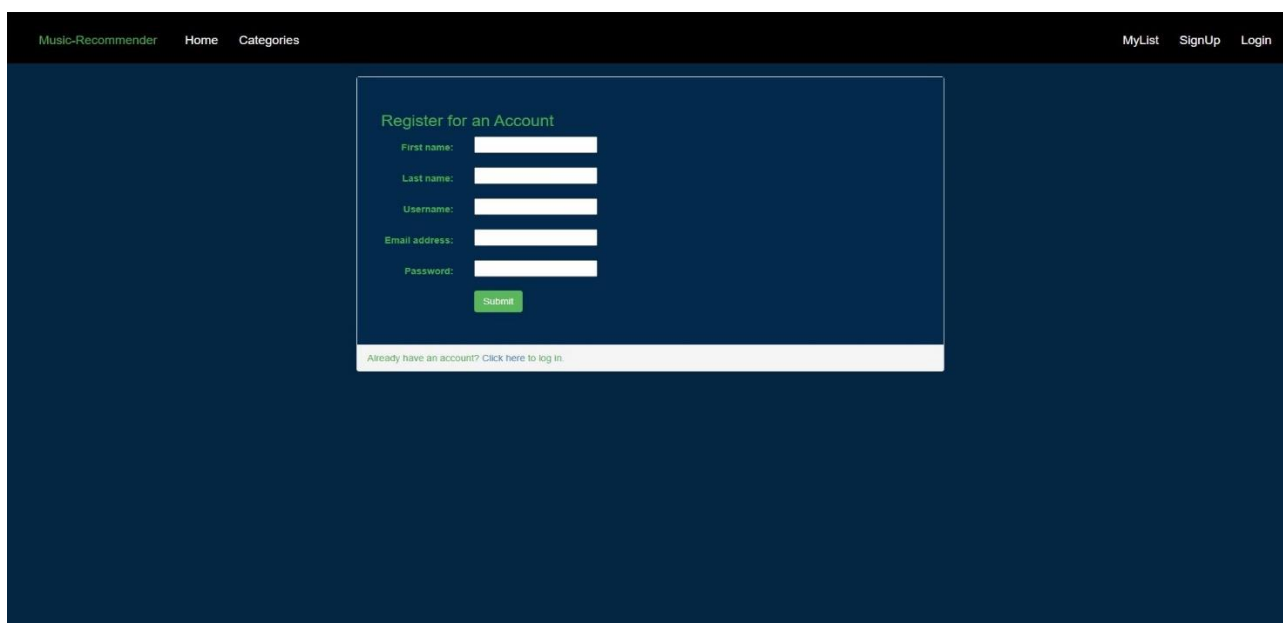


Рисунок 3.5 – Страница регистрации.

Страница музыкального трека

На странице содержится информация о музыкальном треке (рисунок 4.5). Пользователь может посмотреть информацию о жанре музыкальной записи и послушать ее. После прослушивания музыкального трека пользователь может поставить ему оценку, выбрав соответствующее количество звезд, и нажать на кнопку «Submit». А если пользователь хочет добавить музыкальный трек в список любимых музыкальных записей, то он должен поставить галочку и нажать кнопку «Add».

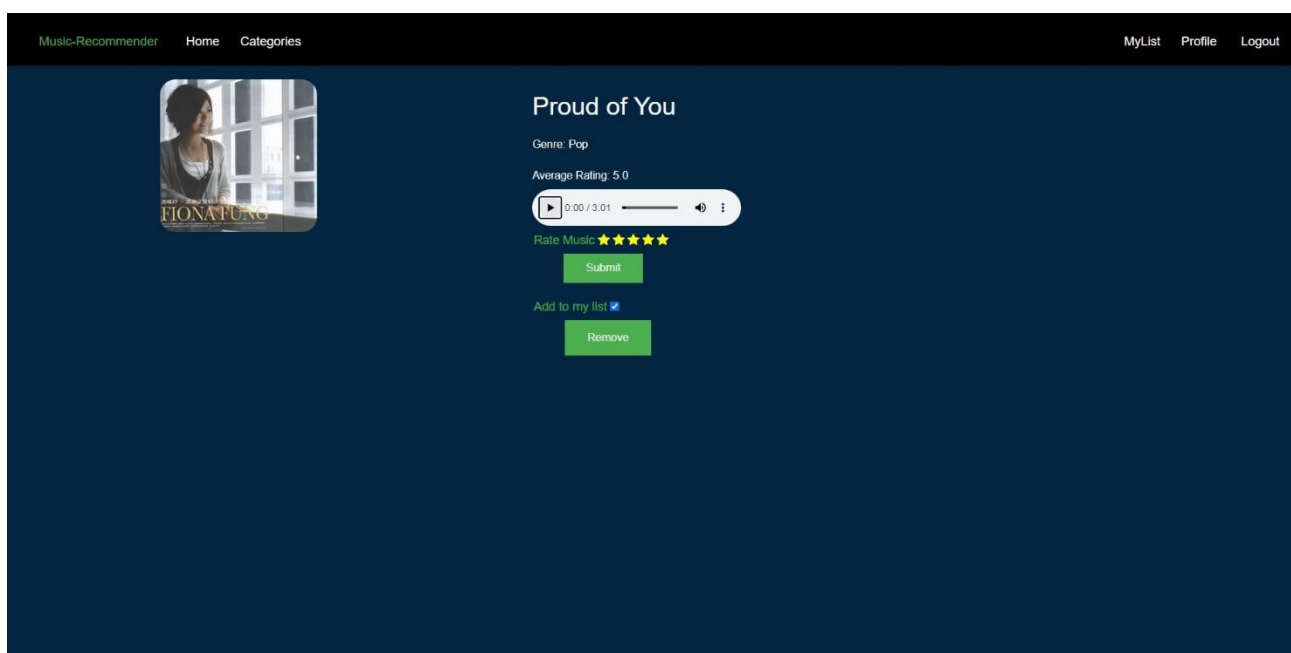


Рисунок 3.6 – Страница музыкального трека.

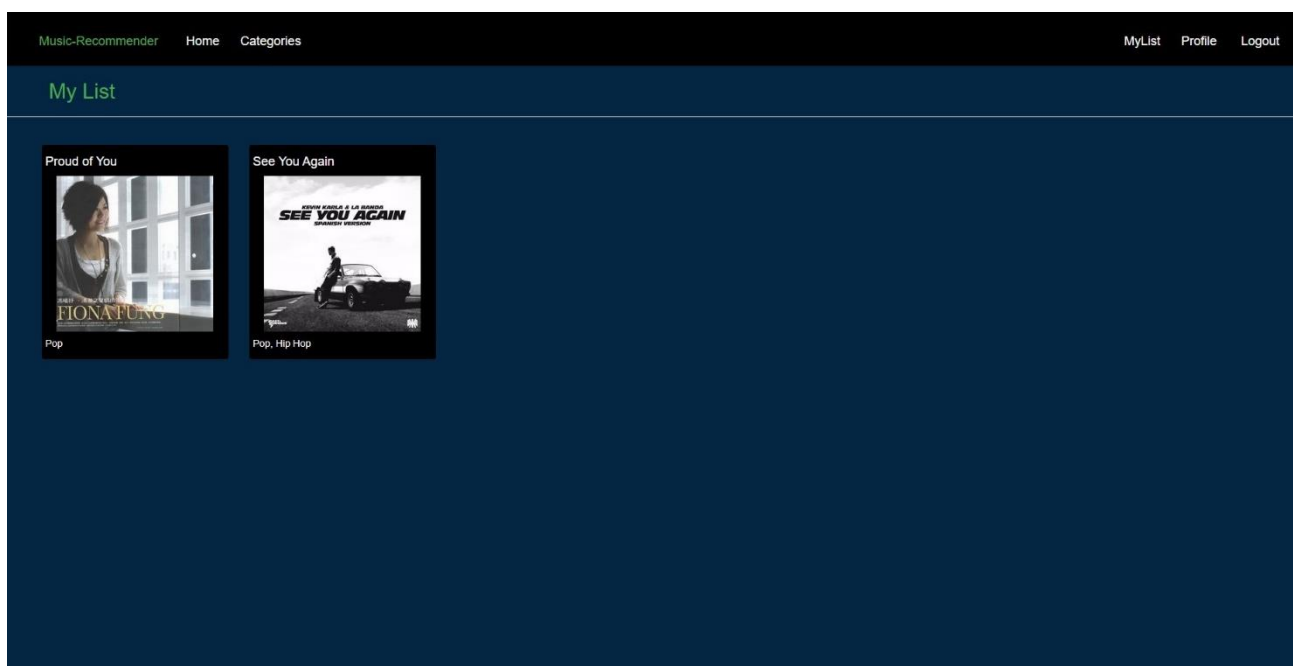


Рисунок 3.7 – Страница избранного списка.

Поиск конкретного музыкального трека и поиск по фильтрам

Если пользователь помнит точное название музыкальной записи, то он может использовать поиск по конкретному названию музыкального трека (рисунок 4.7).

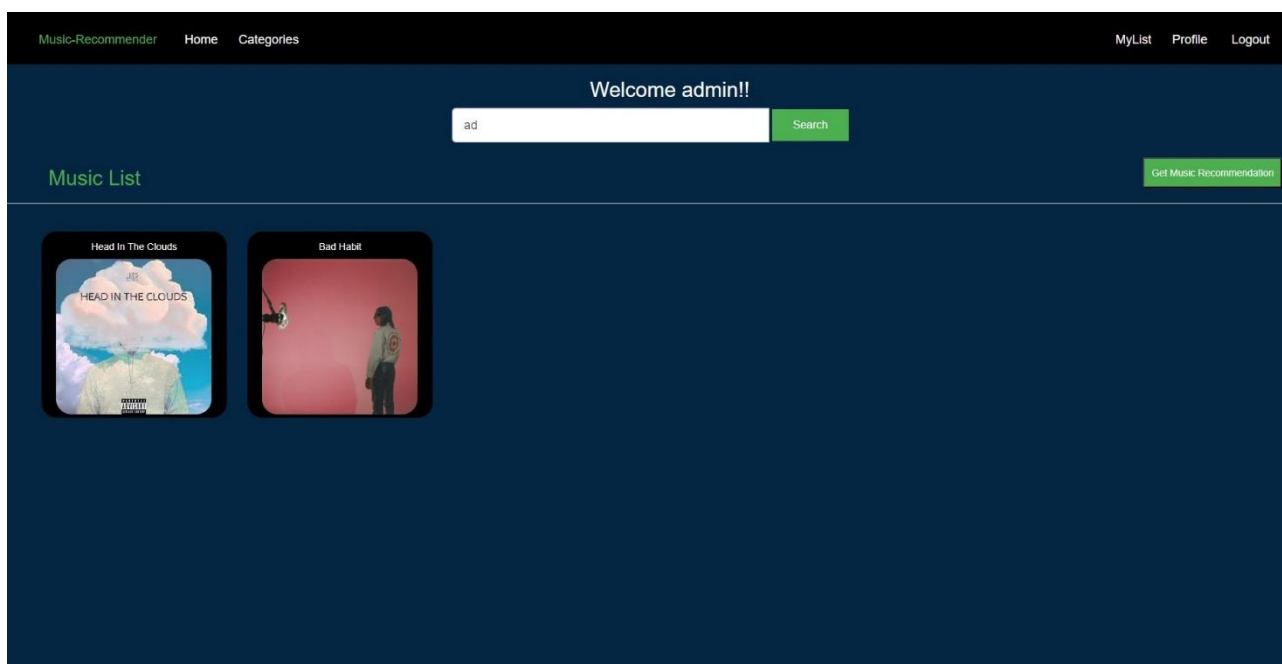


Рисунок 3.8 – Страница поиска конкретного трека.

Если пользователь хочет найти музыкальные треки определенных жанров, то ему надо нажать кнопку «Categories» и выбрать нужный жанр (рисунок 4.8).

Затем он перейдет на страницу, содержащую музыкальные треки выбранного жанра.

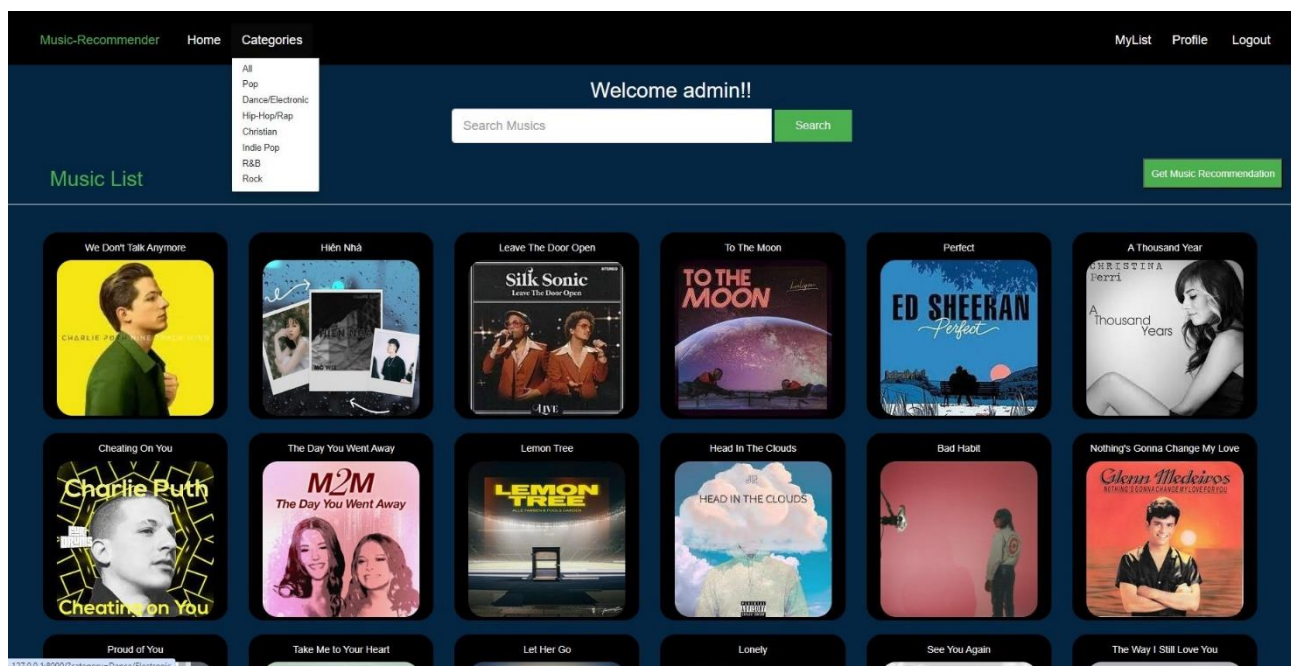


Рисунок 3.9 – Список жанра.

На рисунке 4.9 представлен пример фильтрации треков по жанре «Hip-Hop/Rap».

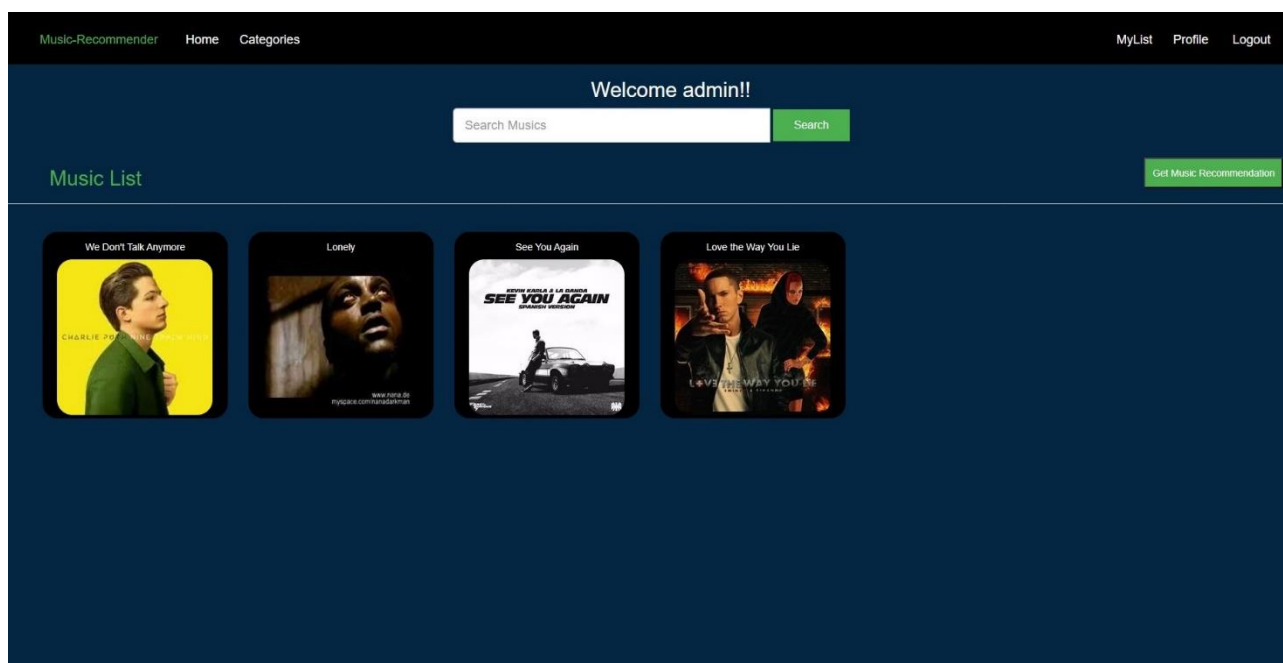


Рисунок 3.10 – Страница поиска по фильтрам.

Получение рекомендаций

Если пользователь авторизован, он может использовать рекомендательную систему. Для этого необходимо нажать кнопку «Get Music Recommendation» на главной странице. После этого произойдет переход на страницу, содержащую рекомендованные музыкальные треки, которые пользователь еще не оценил.

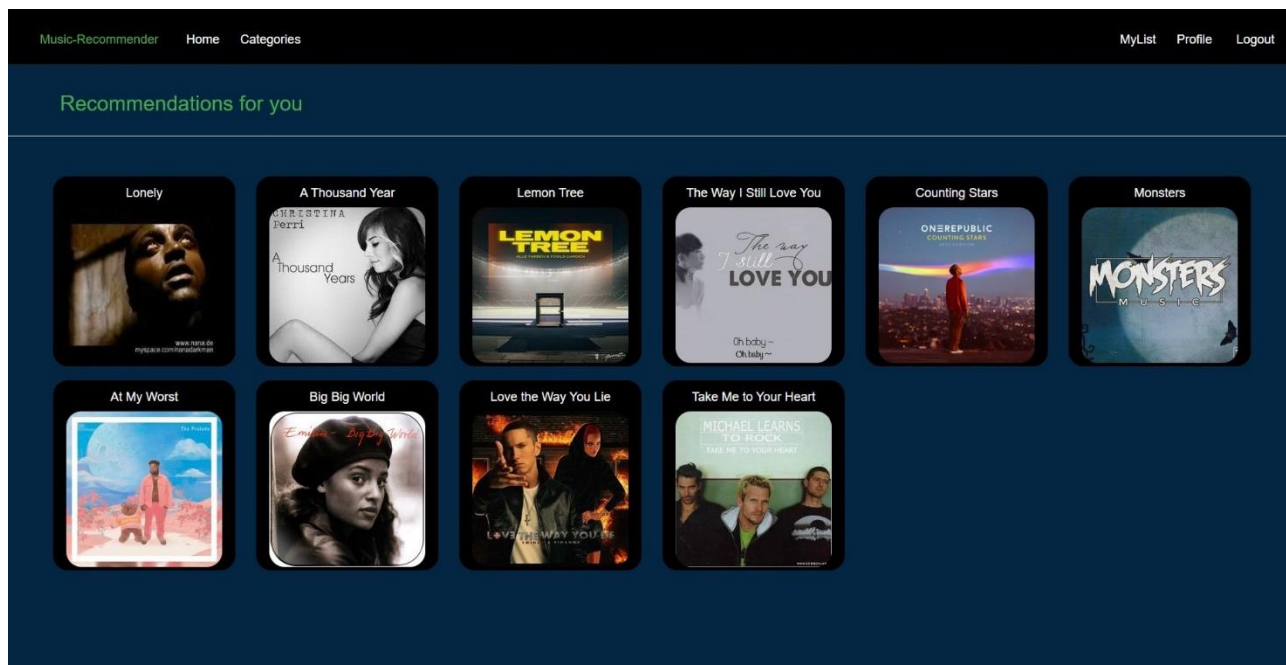


Рисунок 3.11 – Страница получения рекомендаций.

Профиль пользователя

Если авторизованный пользователь хочет посмотреть или изменить свой профиль, то он должен нажать кнопку «Profile». Чтобы обновить свой профиль, пользователю нужно ввести необходимую информацию и нажать кнопку «Update».

Music-Recommender Home Categories MyList Profile Logout

This is the profile of admin
You are Admin

First Name:
Vu

Last Name:
Quang

Email Address:
admin@gmail.com

Update

Рисунок 3.12 – Страница профиля пользователя.

3.3.2. Администратор

При переходе по ссылке «127.0.0.1:8000/admin», пользователь становится администратором.

Управление пользователями

Если администратор хочет добавить пользователя, то он нажимает кнопку «Add user» и вводит необходимую информацию о пользователе.

Если администратор хочет удалить пользователя, то он должен выбрать пользователя и нажать кнопку «Go». Аккаунт пользователя будет удален.

Если администратор хочет обновить информацию о пользователе, то он выбирает пользователя и редактирует соответствующую информацию.

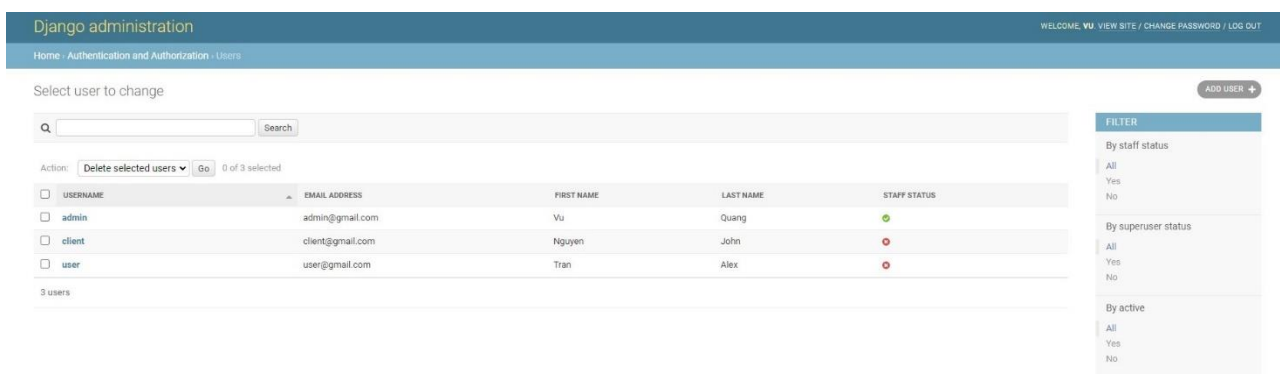


Рисунок 3.13 – Страница управления пользователями.

Управление музыкальными треками и управление категориями

Управление музыкальными треками и управление категориями аналогично управлению пользователями.

Страница управления музыкальными треками представлена на рисунке 4.13, а страница управления категориями на рисунке 4.14.

Если администратор хочет добавить музыкальный трек, то он нажимает кнопку «Add music» и вводит необходимую информацию о музыкальном треке.

Если администратор хочет удалить музыкальный трек, то он должен выбрать музыкальный трек и нажать кнопку «Go». Этот музыкальный трек будет удален.

Если администратор хочет обновить информацию о музыкальном треке, то он выбирает музыкальный трек и редактирует соответствующую информацию.

Если администратор хочет добавить категорию, то он нажимает кнопку «Add category» и вводит необходимую информацию о категории.

Если администратор хочет удалить категорию, то он должен выбрать категорию и нажать кнопку «Go». Эта категория будет удалена.

Если администратор хочет обновить информацию о категории, то он выбирает категорию и редактирует соответствующую информацию.



Рисунок 3.14 – Страница управления музыкальными треками.

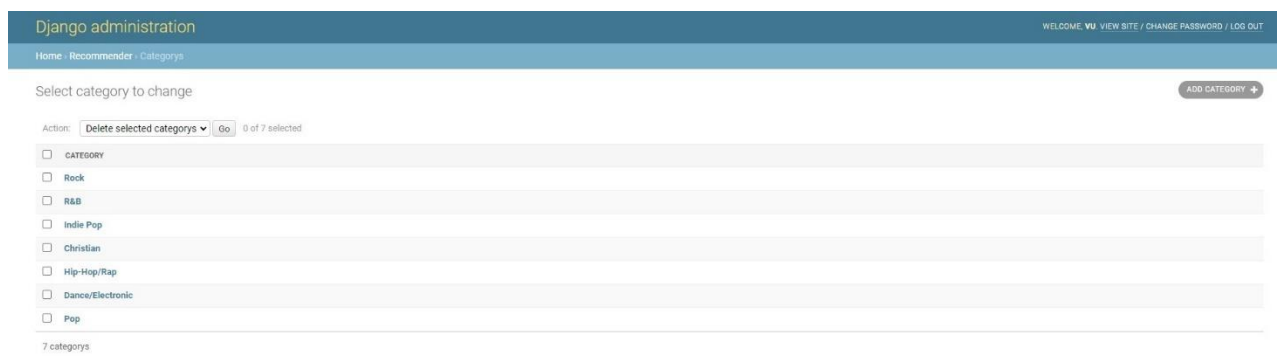


Рисунок 3.15 – Страница управления категориями.

3.4. Сборка ПО

Для реализовать работу веб-приложения, сначала нам нужно создать виртуальную среду и активировать эту виртуальную среду.

После установки `virtualenv`, мы установим все другие пакеты, в том числе и `django` в изолированные окружения. Файл «`requirements.txt`» содержит версии библиотек Python, необходимые для разработки веб-приложения музыкального

онлайн-проигрывателя. На листинге 3.16 описаны библиотеки, необходимые для программы.

Листинг 3.6 – Виртуальная среда

```
virtualenv .  
cd Scripts  
.\activate
```

Листинг 3.7 – Установить файл «requirements.txt».

```
pip install -r requirements.txt
```

Листинг 3.8 – Файл «requirements.txt».

```
asgiref==3.2.7  
boto3==1.14.1  
botocore==1.17.1  
Django==3.0.6  
django-storages==1.9.1  
docutils==0.15.2  
gunicorn==20.0.4  
idna==2.9  
jmespath==0.10.0  
numpy  
pandas  
Pillow  
python-dateutil==2.8.2  
pytz==2020.1  
recommend==0.2.2  
requests==2.23.0  
s3transfer==0.3.3  
scipy
```

Продолжение листинга 3.8

```
six==1.15.0  
sqlparse==0.3.1  
urllib3==1.25.9  
whitenoise==5.1.0
```

В Django встроен простой виртуальный веб-сервер. Не надо устанавливать никаких других программ на локальной машине. Чтобы его проверить, запустите в терминале команду:

Листинг 3.9 – Запустить сервер.

```
python manage.py runserver
```

Затем переходить по ссылке «<http://127.0.0.1:8000>», чтобы протестировать программу.

Вывод

В данном разделе был представлен интерфейс сайта и его функциональные возможности. Описана каждая страница сайта и ее предназначение. Продемонстрирована работа рекомендательной системы и режима фильтрации музыкальных треков.

4. Исследовательский раздел

4.1. Технические характеристики

Colaboratory

Colaboratory (Colab) [21] — это продукт, бесплатно выпущенный исследовательским отделом Google для пользователей, позволяющих запускать программы, написанные на языке Python, через веб-браузер.

Большим преимуществом сервера Colab является то, что на нем предустановлены популярные библиотеки для задач глубокого обучения, такие как Tensorflow, Keras, sklearn, numpy..., что значительно упрощает настройку исходной среды. Кроме того, Colab позволяет легко подключаться к таким ресурсам, как Github (предоставляет общие настройки исходного кода), Google Drive (предоставляет пространство для хранения данных, совместимое с Colab), что упрощает работу. Удобно для исследователей быстро устанавливать и тестировать модели.

В Colab использует технические характеристики устройства, на котором выполнялось исследование:

- операционная система: Ubuntu 22.04.1 Linux x86_64 [24];
- оперативная память: 8 Гбайт;
- процессор: 11th Gen Intel® Core™ i5-1135G7 @ 4.20ГГц [25].

Locust [22] — это инструмент нагрузочного тестирования с открытым исходным кодом, который использует HTTP и несколько других протоколов. Он выбран в качестве тестирования, потому что он имеет следующие преимущества:

- Простое использование и высокая мотивация
- Хорошая масштабируемость
- Можно комбинировать с другими инструментами
- Предоставляет подробную статистику

Набор данных Movielens

Movielens [23] была создана в 1997 году для сбора данных обзора фильмов в исследовательских целях.

Набор данных ml-100k представляет собой набор данных из 100 000 оценок от 943 пользователей для 1682 фильмов, выпущенных в апреле 1998 года.

Набор данных ml-1M представляет собой набор данных из 1 000 209 оценок от 6 040 пользователей для 3 900 фильмов, выпущенных в феврале 2003 года.

4.2. Точность предсказаний

Данный тип метрик позволяет оценить разницу между предсказанной и реальной оценкой, причем одной из самых популярных метрик этого типа является средняя абсолютная ошибка (MAE). Кроме того, используются другие связанные метрики, такие как среднеквадратичная ошибка (MSE), корень из среднеквадратичной ошибки (RMSE) или нормализованная средняя абсолютная ошибка [17].

Средняя абсолютная ошибка (MAE)

Средняя абсолютная ошибка представляет собой среднее значение абсолютной разницы между предсказанием алгоритма и реальными оценками. Это одна из самых популярных метрик при оценке рекомендательных систем. Она измеряет среднее значение остатков в наборе данных. Среднюю абсолютную ошибку можно вычислить по следующей формуле:

$$MAE = \frac{\sum_{i=1}^N |y_i - \hat{y}|}{N}, \quad (4.1)$$

где N — количество точек данных; y_i — реальные оценки; \hat{y} — предсказание алгоритма.

Корень из среднеквадратичной ошибки (RMSE)

Среднеквадратичная ошибка (RMSE) — это квадратный корень из среднего квадрата ошибок. RMSE, рассчитанная с использованием следующей формулы, измеряет стандартное отклонение остатков (ошибку прогноза):

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2}, \quad (4.2)$$

где N — количество точек данных; y_i — реальные оценки; \hat{y} — предсказание алгоритма.

Основная причина использования этой метрики заключается в том, что ошибки, выявленные с помощью этой метрики, могут оказать более сильное влияние на решение пользователя.

Чем меньше значение MAE и RMSE, тем точнее результаты рекомендательной системы.

4.3. Результаты исследования

Чтобы избежать случайности алгоритма, каждый метод построения рекомендаций запускался 10 раз и вычисляется средняя ошибка за 10 прогонов.

Таблица 4.1 – MAE и RMSE между методами при наборе данных ml-100k

	SVD	PMF	NMF	User-based	Item-based
MAE	0.7384	0.7473	0.7727	0.7693	0.7696
RMSE	0.9369	0.9495	0.979	0.9736	0.9741

Таблица 4.2 – MAE и RMSE между методами при наборе данных ml-1m

	SVD	PMF	NMF	User-based	Item-based
MAE	0.6856	0.6876	0.7275	0.7198	0.7197
RMSE	0.8737	0.8743	0.9228	0.9145	0.9144

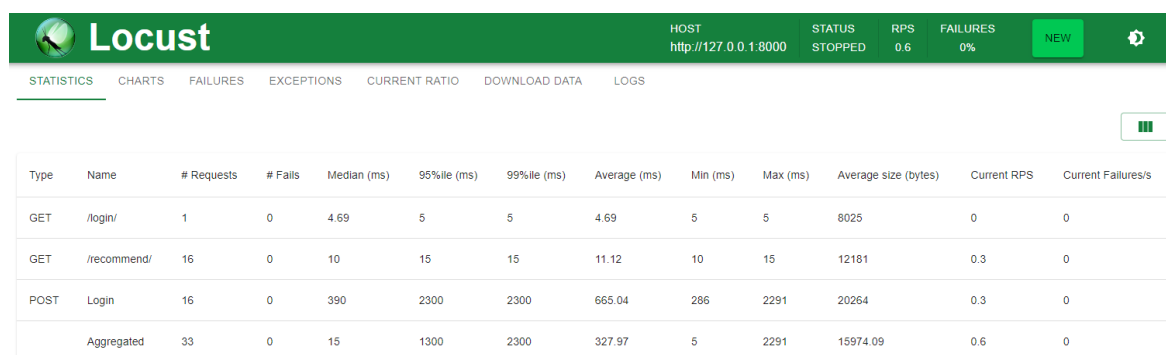
При наборе данных ml-100k мы сравниваем различные меры сходства, используя коллаборативную фильтрацию:

Таблица 4.3 – Сравнительный анализ различных мер сходства

	Корреляции Пирсона	Косинусное сходство	Среднеквадратическая разница
MAE	0.8032	0.8040	0.7732
RMSE	1.0124	1.0167	0.9789

Исследование веб-приложения

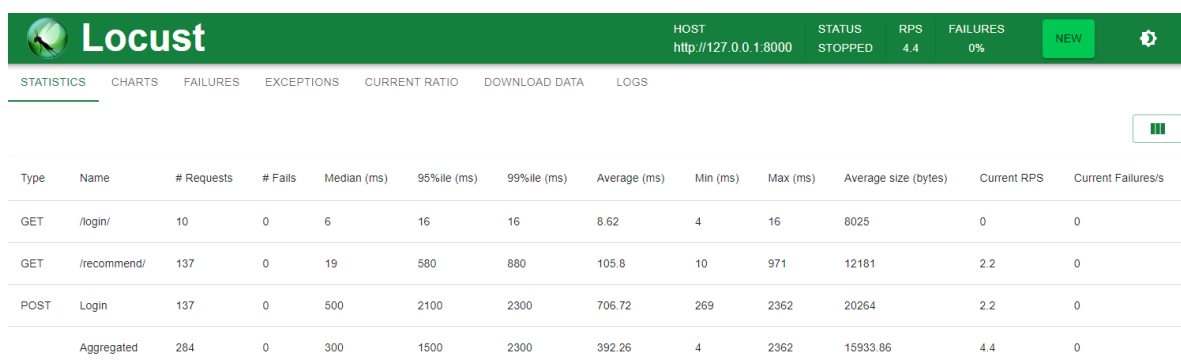
Для одного пользователя среднее время ответа на запрос рекомендательной системы составляет 11,12 мс. А при одновременном запросе 10 пользователей среднее время ответа составляет 105,8 мс.



The screenshot shows the Locust web interface with the following data:

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/login/	1	0	4.69	5	5	4.69	5	5	8025	0	0
GET	/recommend/	16	0	10	15	15	11.12	10	15	12181	0.3	0
POST	Login	16	0	390	2300	2300	665.04	286	2291	20264	0.3	0
Aggregated		33	0	15	1300	2300	327.97	5	2291	15974.09	0.6	0

Рисунок 4.1 – Запрос пользователя.

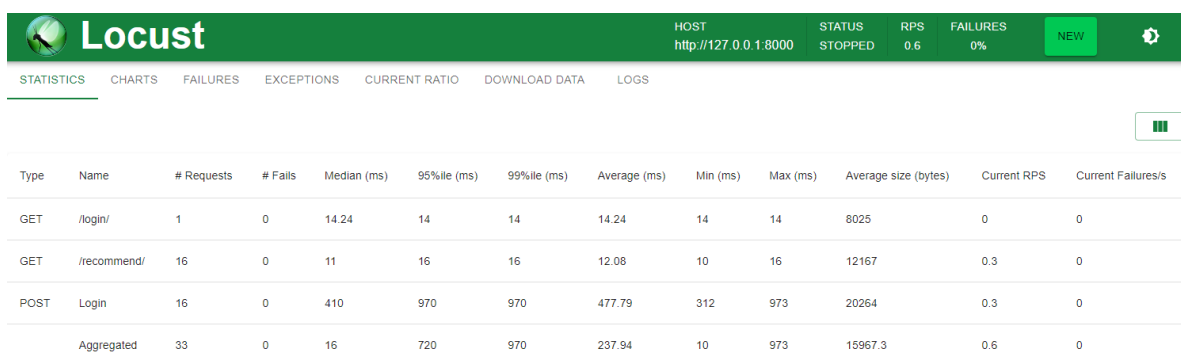


The screenshot shows the Locust web interface with the following data:

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/login/	10	0	6	16	16	8.62	4	16	8025	0	0
GET	/recommend/	137	0	19	580	880	105.8	10	971	12181	2.2	0
POST	Login	137	0	500	2100	2300	706.72	269	2362	20264	2.2	0
Aggregated		284	0	300	1500	2300	392.26	4	2362	15933.86	4.4	0

Рисунок 4.2 – Одновременный запрос 10 пользователей.

Когда мы добавляем 1 рейтинг пользователя, для 1 пользователя среднее время ответа на запрос рекомендательной системы увеличивается от 11,12 мс до 12,08 мс.



The screenshot shows the Locust web interface with the following data:

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/login/	1	0	14.24	14	14	14.24	14	14	8025	0	0
GET	/recommend/	16	0	11	16	16	12.08	10	16	12167	0.3	0
POST	Login	16	0	410	970	970	477.79	312	973	20264	0.3	0
Aggregated		33	0	16	720	970	237.94	10	973	15967.3	0.6	0

Рисунок 4.3 – Запрос пользователя при добавлении рейтинга.

На рисунке 4.4 показана зависимость времени ответа на запрос рекомендательной системы от количества пользователей:

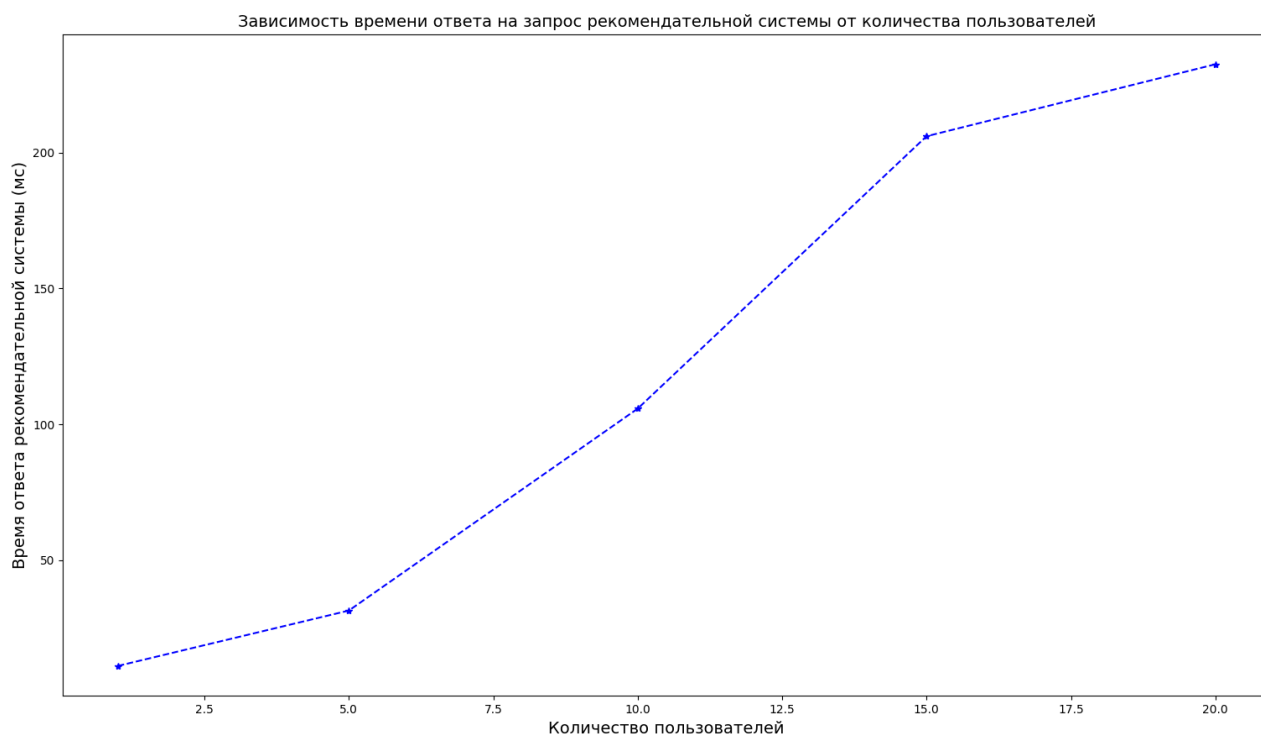


Рисунок 4.4 – График зависимости времени ответа на запрос рекомендательной системы от количества пользователей.

Вывод

На таблицах 4.1 и 4.2, мы видим, что алгоритм SVD является лучшим, но остальные алгоритмы на двух наборах данных с набором данных Movielens 100K и набором данных Movielens 1M все имеют результаты с ошибками RMSE и MAE меньше 1. Таким образом, мы видим, что погрешности небольшие, в пределах допустимого и становятся лучше, когда данных больше.

В таблице 4.3 мы сравниваем результаты, полученные по всем мерам сходства на основе разных метрик. Результат показывает, что сходство, основанное на среднеквадратичной разнице, является одним из лучших показателей расстояния, используемых в коллаборативной фильтрации на основе сходства пользователей с использованием общедоступного набора данных MovieLens.

Из рисунка 4.4, мы видим, что чем больше данных, тем дольше время обработки, что приводит к большему времени ответа на запрос рекомендательной системы пользователю. Кроме того, если много пользователей запрашивают одновременно, системе необходимо одновременно обрабатывать много запросов, и время ответа увеличивается.

ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной квалификационной работы были получены следующие результаты:

- проведен сравнительный анализ алгоритмов рекомендательных систем;
- выполнено концептуальное и логическое проектирование базы данных для музыкального онлайн-проигрывателя;
- реализован алгоритм коллаборативной фильтрации на основе сходства пользователей;
- разработано веб-приложение музыкального онлайн-проигрывателя, включающее в себя рекомендательную систему на основе коллаборативной фильтрации по сходству пользователей;
- выполнено исследование выбранного алгоритма построения рекомендаций и эффективности разработанного приложения.

В результате исследования показано, что хотя коллаборативная фильтрация на основе сходства пользователей не превосходит по эффективности другие алгоритмы, но разница между ними незначительна, и значения всех метрик качества рекомендаций соответствуют общепринятым требованиям. Разработанная рекомендательная система для музыкального онлайн-проигрывателя с использованием коллаборативной фильтрации на основе сходства пользователей является простой и эффективной. Она позволит пользователям легче находить интересные для них музыкальные записи.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рекомендательные системы: [Электронный ресурс]. – Режим доступа URL: https://rdc.grfc.ru/2023/05/recommendation_services_methods_and_regulation/ (дата обращения: 30.04.2024)
2. Pazzani M., Billsus D., Learning and Revising User Profiles: The Identification of Interesting Web Sites, // Mach. Learn. - Spec. issue multistrategy Learn., T. 27, № 3, с. 313–331, 1997.
3. Somlo G. L., Howe A. E., Somlo G. L., Adaptive Lightweight Text Filtering Adaptive Lightweight Text Filtering, // in IDA 2001: Advances in Intelligent Data Analysis, 2001, T. 2189, с. 319–329.
4. Su X., Khoshgoftaar T. M., A Survey of Collaborative Filtering Techniques, // Adv. Artif. Intell., T. 2009, 2009.
5. Isinkaye F. O., Folajimi Y. O., Ojokoh B. A., Recommendation systems: Principles, methods and evaluation, // Egypt. Informatics J., T. 16, № 3, с. 261–273, 2015.
6. Sarwar B., Item-Based Collaborative Filtering Recommendation Algorithms, // in Proceedings of the 10th international conference on World Wide Web, 2001, с. 285–295.
7. Sindhvani P. M. V., Recommender systems, // Commun. ACM, с. 1–21, 2010.
8. Herlocker J. L., Konstan J. A., Terveen L. G., Evaluating Collaborative Filtering Recommender Systems, // ACM Trans. Inf. Syst., T. 22, № 1, с. 5–53, 2004.
9. Portugal I., Alencar P., Cowan D., The use of machine learning algorithms in recommender systems: A systematic review, // Expert Syst. Appl., T. 97, с. 205–227, 2018.
10. Mobasher B., Jin X., Zhou Y., Semantically enhanced collaborative filtering on the web, // Lect. Notes Comput. Sci., T. 3209, № 49, с. 57–76, 2004.
11. Han J., Kamber M., Data Mining: Concepts and Techniques, // Data Mining Concepts Tech., с. 3–26, 2000.

12. Ren L., Wang W., An SVM-Based Collaborative Filtering Approach for Top-N Web Services Recommendation, // Futur. Gener. Comput. Syst., 2017.
13. Caruana R., Niculescu-Mizil A., An empirical comparison of supervised learning algorithms, // Proc. 23rd Int. Conf. Mach. Learn., Т. С, № 1, с. 161–168, 2006.
14. SQLite Documentation: [Электронный ресурс]. – Режим доступа URL: <https://www.sqlite.org/docs.html> (дата обращения: 30.04.2024)
15. Welcome to Python: [Электронный ресурс]. – Режим доступа URL: <https://www.python.org/> (дата обращения: 30.04.2024)
16. Коэффициент корреляции Спирмена: [Электронный ресурс]. – Режим доступа URL: <https://www.simplilearn.com/tutorials/statistics-tutorial/spearmans-rank-correlation> (дата обращения: 30.04.2024)
17. Метрики качества линейных регрессионных моделей: [Электронный ресурс]. – Режим доступа URL: <https://loginom.ru/blog/quality-metrics> (дата обращения: 30.04.2024)
18. An algorithmic Framework for Performing Collaborative Filtering: [Электронный ресурс]. – Режим доступа URL: <https://dl.acm.org/doi/10.1145/312624.312682> (дата обращения: 30.04.2024)
19. Scikit-learn: [Электронный ресурс]. – Режим доступа URL: <https://scikit-learn.org/stable/> (дата обращения: 30.04.2024)
20. Django: [Электронный ресурс]. – Режим доступа URL: <https://www.djangoproject.com/> (дата обращения: 30.04.2024)
21. Google Colab: [Электронный ресурс]. – Режим доступа URL: <https://colab.google/> (дата обращения: 30.04.2024)
22. Locust: [Электронный ресурс]. – Режим доступа URL: <https://locust.io/> (дата обращения: 30.04.2024)
23. Movielens: [Электронный ресурс]. – Режим доступа URL: <https://movielens.org/> (дата обращения: 30.04.2024)

24. Ubuntu 22.04.1 Linux x86_64: [Электронный ресурс]. – Режим доступа URL: <https://old-releases.ubuntu.com/releases/22.04.1/> (дата обращения: 30.04.2024)
25. Процессор Intel® Core™ i5-1135G7: [Электронный ресурс]. – Режим доступа URL: <https://ark.intel.com/content/www/ru/ru/ark/products/208658/intelcorei51135g7-processor-8m-cache-up-to-4-20-ghz.html> (дата обращения: 30.04.2024)

ПРИЛОЖЕНИЕ А

Листинг А.1 – views.py

```
# Recommendation Algorithm
def recommend(request):
    if not request.user.is_authenticated:
        return redirect("login")
    if not request.user.is_active:
        raise Http404

    music_rating = pd.DataFrame(list(Myrating.objects.all().values()))

    new_user = music_rating.user_id.unique().shape[0]
    current_user_id= request.user.id
    # if new user not rated any music
    if current_user_id > new_user:
        # 10 bai hat co luot rating trung binh cao nhat
        core = music_rating.groupby('music_id').mean()
        core_rating = core['rating'].sort_values(ascending=False)[:10]
        begin_id = core_rating.index.tolist()
        #print(core_rating)
        #print(begin_id)
        new_list = list(Music.objects.filter(id__in = begin_id))
        #print(new_list)
        context_0 = {'music_list': new_list}
        return render(request, 'recommend/recommend.html', context_0)

    music_titles_map = {}
    user_music_rating = {}
    music_user_map = {}
    user_avg_rating = {}
    cache = {}

    list_music = list(Music.objects.all().values())
    for parts in list_music:
        music_id = int(parts['id'])
        music_title = parts['title']
```

Продолжение листинга А.1

```
music_titles_map[music_id] = music_title

list_rating = list(Myrating.objects.all().values())
for parts in list_rating:
    music_title_id = int(parts['music_id'])
    user_id = int(parts['user_id'])
    rating = int(parts['rating'])
    if user_id not in user_music_rating:
        user_music_rating[user_id] = {}
    user_music_rating[user_id][music_title_id] = rating

    if music_title_id not in music_user_map:
        music_user_map[music_title_id] = []
    music_user_map[music_title_id].append(user_id)
# get_user_avg_rating
for user in user_music_rating:
    sum = 0
    i = 0
    for music in user_music_rating[user]:
        sum = sum + float(user_music_rating[user][music])
        i = i + 1
    avg = sum/i
    user_avg_rating[user] = avg
active_user = request.user.id;
# Pearson correlation coefficient
user_correlation = {}
for user in user_music_rating:
    if user != active_user:
        nominator = 0
        sum_vaj_diff = 0
        sum_vij_diff = 0
        for music in user_music_rating[active_user]:
            if music in user_music_rating[user]:
                nominator += (user_music_rating[active_user][music] - user_avg_rating[active_user]) *
                (user_music_rating[user][music] - user_avg_rating[user])
```

Продолжение листинга А.1

```
        sum_vaj_diff += np.power(user_music_rating[active_user][music] - user_avg_rating[active_user], 2)

        sum_vij_diff += np.power(user_music_rating[user][music] - user_avg_rating[user], 2)
        denominator = np.sqrt(sum_vaj_diff * sum_vij_diff)
        if denominator != 0:
            user_correlation[user] = nominator/denominator
    if active_user not in cache:
        predicted_rating = {}
        for music in music_titles_map:
            temp_rating = 0
            if music in music_user_map:
                for user in music_user_map[music]:
                    if user in user_correlation:
                        temp_rating += user_correlation[user] * (user_music_rating[user][music] - user_avg_rating[user])
            predicted_rating[music] = temp_rating

        predicted_rating = sorted(predicted_rating.items(), key=lambda kv: kv[1], reverse=True)
        cache[active_user] = predicted_rating
    else:
        predicted_rating = cache[active_user]
    musics_id = []
    for i in predicted_rating:
        musics_id.append(i[0])

    user = pd.DataFrame(list(Myrating.objects.filter(user=request.user).values())).drop(['user_id', 'id'], axis=1)
    user_filtered = [tuple(x) for x in user.values] # doi type sang dang tuple
    music_id_watched = [each[0] for each in user_filtered]
    musics_id_recommend = [each for each in musics_id if each not in music_id_watched]
    preserved = Case(*[When(pk=pk, then=pos) for pos, pk in enumerate(musics_id_recommend)])
    music_list = list(Music.objects.filter(id__in = musics_id_recommend).order_by(preserved)[:10])
    print(musics_id_recommend)

    context = {'music_list': music_list}
    return render(request, 'recommend/recommend.html', context)
```


ПРИЛОЖЕНИЕ Б

(презентация)