Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

по направлению: 09.04.03 Прикладная информатика

(код и наименование направления)

профиль: Аналитические информационные системы

на тему: «Разработка алгоритма сортировки для рекомендательной системы подбора музыкальных композиций»

(наименование темы)

ИКСИБ.КИСП.09.04.03.003.ПП

(обозначение документа)

Автор  / А.А. Ручка /

(подпись, дата, расшифровка подписи)

Руководитель / канд. техн. наук., доц. А.Г. Мурлин /

(подпись, дата, расшифровка подписи)

Консультанты:

Раздел информационной

безопасности / канд. техн. наук, доц. В.А. Шарай /

(подпись, дата, расшифровка подписи)

Нормоконтролер  / ст. преп. А.А. Ковтун /

(подпись, дата, расшифровка подписи)

Выпускная квалификационная работа  
допущена к защите

(дата)

Заведующая кафедрой / канд. техн. наук, доц. М.В. Янаева /

(подпись)

Краснодар

2021Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

УТВЕРЖДАЮ

Заведующая кафедрой ИСП

к.т.н, доц. \_\_\_\_\_\_\_\_М.В. Янаева

« » 2021 г.

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

по направлению 09.04.03 Прикладная информатика

(шифр и наименование)

профиль: Аналитические информационные системы

студенту Ручка Артему Алексеевичу

(фамилия, имя, отчество)

Тема выпускной квалификационной работы: Разработка алгоритма сортировки для рекомендательной системы подбора музыкальных композиций

Утверждена приказом ректора университета от 07.04.2021 № 696-Ст

Руководитель канд. техн. наук., доц. А.Г. Мурлин

(должность, фамилия, инициалы)

Консультанты:

Раздел информационной безопасности канд. техн. наук, доц. В.А. Шарай

(должность, фамилия, инициалы)

Нормоконтролер ст. преп. А.А. Ковтун

(должность, фамилия, инициалы)

Срок сдачи выпускной квалификационной работы на кафедру 17.06.2021 г.

**Содержание выпускной квалификационной работы**

Введение

1 Нормативные ссылки

2 Термины

3 Сокращения

4 Анализ существующих рекомендательных систем

5 Исследование и описание алгоритма рекомендации

6 Разработка требований к информационной системе

7 Проектирование информационной системы

8 Разработка информационной системы

9 Описание приложения

10 Информационная безопасность

Заключение

Список использованных источников

Общее количество листов ПЗ \_\_\_

**Объем иллюстративной части**

1. Функциональная схема ИС.

2. Структура приложения.

3. Формы интерфейса приложения.

Общее количество слайдов иллюстративной части 15

**Список основной и рекомендуемой литературы**

1. Методические указания по выполнению выпускной квалификационной работы для студентов всех форм обучения направлений 09.04.03 Прикладная информатика, 09.04.04 Программная инженерия / Сост.: Л.А. Видовский, М.В. Янаева; Кубан. гос. технол. ун-т. Каф. информационных систем и программирования. – Краснодар: Изд. КубГТУ, 2017. – 35 с.

**Календарный план выполнения выпускной квалификационной работы**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Месяц | Числа месяца | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Апрель |  | | | | | | | *Описание требований к ИС и постановка задачи ВКР* | | | | | | | | *Техническое задание на ИС* | | | | | | *Построение модели ИС. Проектирование БД* | | | | | | | | | | |
| Май | *Реализация БД. Реализация ПО* | | | | | | | | | | | | | | | | | | | | | *Информационная безопасность.* | | | | | | | | | |  |
| Июнь | *Оформление пояснительной записки* | | | | | | | | | | *Нормоконтроль, антиплагиат.*  *Сдача ВКР на кафедру* | | | | | | | | *Защита ВКР* | | | | | | |  | | | | | | |

Студент А.А. Ручка Руководитель канд. техн. наук., доцент А.Г. Мурлин

(подпись, дата) (подпись, дата)

**Реферат**

Выпускная квалификационная работа содержит 66 страницы, 19 рисунков, 6 таблиц, 11 формул, 14 источников.

РЕКОМЕНДАТЕЛЬНАЯ ИНФОРМАЦИОННАЯ СИСТЕМА, СОРТИРОВКА, АЛГОРИТМ, ЗВУК, МУЗЫКА, PYTHON, JAVA, MYSQL, БАЗА ДАННЫХ, FLASK, SPRING

Целью выпускной квалификационной работы является создание метода сортировки музыкальных композиций и разработка рекомендательной информационной системы подбора музыкальных композиций.

В результате выполнения выпускной квалификационной работы проведен анализ существующих методов рекомендации музыкальных композиций и создан алгоритм поиска похожих по звуковым характеристикам музыкальных композиций. Также была разработана рекомендательная информационная система для подбора музыкальных композиций, основанная на данном алгоритме.

Программное обеспечение создано для автоматизации подбора музыкальных композиций на основе вкуса пользователя.

Реализованная ИС должна повысить эффективность рекомендательной системы подбора музыкальных композиций.

**Содержание**

[Введение 8](#_Toc74637482)

[1 Нормативные ссылки 10](#_Toc74637483)

[2 Термины 11](#_Toc74637484)

[3 Сокращения 12](#_Toc74637485)

[4 Анализ существующих рекомендательных систем 13](#_Toc74637486)

[4.1 Развитие рекомендательных систем 13](#_Toc74637487)

[4.2 Обзор существующих решений 14](#_Toc74637488)

[4.3 Классификация рекомендательных систем 18](#_Toc74637489)

[4.3.1 Рекомендательные системы на основе коллаборативной фильтрации 18](#_Toc74637490)

[4.3.2 Рекомендательные системы, основанные на контенте 20](#_Toc74637491)

[4.3.3 Рекомендательные системы, основанные на знаниях 21](#_Toc74637492)

[5 Исследование и описание алгоритма рекомендации 22](#_Toc74637493)

[5.1 Набор данных 22](#_Toc74637494)

[5.2 Выбор метода машинного обучения 23](#_Toc74637495)

[5.2.1 Наивный байесовский классификатор 24](#_Toc74637496)

[5.2.2 Метод k-ближайших соседей 25](#_Toc74637497)

[5.2.3. Метод опорных векторов 26](#_Toc74637498)

[5.2.4 Результаты анализа методов машинного обучения 27](#_Toc74637499)

[5.3 Алгоритм рекомендации 27](#_Toc74637500)

[6 Разработка требований к информационной системе 29](#_Toc74637501)

[6.1 Формирование пользовательских требований 29](#_Toc74637502)

[6.2 Формирование системных требований 29](#_Toc74637503)

[7 Проектирование информационной системы 31](#_Toc74637504)

[7.1 Функциональная схема проектируемой информационной системы 31](#_Toc74637505)

[7.2 Объектно-ориентированная модель информационной системы 33](#_Toc74637506)

[7.3 Проектирование БД 35](#_Toc74637507)

[8 Разработка информационной системы 37](#_Toc74637508)

[8.1 Используемые технологии 37](#_Toc74637509)

[8.1.1 Фреймворк Spring 37](#_Toc74637510)

[8.1.2 Шаблон MVC 37](#_Toc74637511)

[8.1.3 Фреймворк Python Flask 38](#_Toc74637512)

[8.2 Структура Java-приложения 38](#_Toc74637513)

[8.3 Структура API 40](#_Toc74637514)

[9 Описание приложения 41](#_Toc74637515)

[9.1 Описание интерфейса пользователя 41](#_Toc74637516)

[10 Информационная безопасность 46](#_Toc74637517)

[10.1 Построение диаграммы потоков данных 46](#_Toc74637518)

[10.2 Схема физического расположения компонентов разработанной ИС 47](#_Toc74637519)

[10.3 Выявление перечня угроз 48](#_Toc74637520)

[10.4 Построение деревьев опасностей 50](#_Toc74637521)

[10.5 Анализ рисков опасности 53](#_Toc74637522)

[10.6 Методы по предотвращению опасностей 55](#_Toc74637523)

[Заключение 56](#_Toc74637524)

[Список использованных источников 57](#_Toc74637525)

[Приложение А – Модуль MainController.java 59](#_Toc74637526)

[Приложение Б – Модуль flaskAPI.py 63](#_Toc74637527)

[Приложение В – Модуль k\_nn.py 65](#_Toc74637528)

Введение

Ежедневно в глобальной сети Интернет генерируются и накапливаются огромные объемы информации. Люди постоянно должны работать с этими данными: обрабатывать, систематизировать, находить новые данные, необходимые им для удовлетворения своих потребностей. Сейчас сложно отбирать интересующую пользователей информацию так как она обычно теряется на фоне того большого объема информации, доступной в современном мире. Поэтому создаются инструменты, которые помогают человеку в поиске релевантной информации и предлагают контент, который интересует пользователя. Такое программное обеспечение называется рекомендательными системами.

Рекомендательные системы – это системы, анализирующие информацию в попытке предсказать, что наиболее вероятно будет интересно для конкретного пользователя в данный момент. Эти системы генерируют рекомендации на основе данных, которые возникают в результате взаимодействия человека с системой, либо на основе данных, указанных пользователем. Рекомендательные системы должны обладать следующими свойствами: программное обеспечение должно быть адаптировано к каждому пользователю системы, так как интересы у каждого человека могут значительно отличаться друг от друга; система должна учитывать текущие интересы пользователя и со временем подстраиваясь под него; рекомендательная система должна постоянно искать новые объекты, которые могут заинтересовать пользователя и давать их в рекомендации. Таки образом, это выгодно и ресурсам, использующим рекомендательные алгоритмы, так как это повышает их популярность и пользователям данных ресурсов, так как система адаптируется под их интересы.

Рекомендательные системы нашли применение во многих областях:

* поиск фильмов и научных статей;
* торговля;
* социальные сети;
* электронная коммерция;
* онлайн-банкинг.

В настоящее время выпущено огромное количество музыкальных композиций, что делает поиск интересных треков долгой и ресурсоемкой задачей. Также существует большое количество музыкальных сервисов. Поэтому задача рекомендации музыки становится все более актуальной: современный пользователь музыкальных сервисов хочет, чтобы сама система предлагала композиции, привлекательные для него.

Целью данной работы является исследование принципов построения рекомендательных систем и их алгоритмов.

В данной работе необходимо провести анализ предметной области и исследовать виды рекомендательных систем и принципы их построения и разработать алгоритм сортировки для рекомендательной системы.

1 Нормативные ссылки

В настоящей ВКР использованы ссылки на следующие руководящие документы:

1. ГОСТ Р 1.12-2004 Стандартизация в Российской Федерации. Термины и определения;
2. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению;
3. ГОСТ Р 7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления;
4. ГОСТ 7.82-2001 СИБИД. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления.

2 Термины

В настоящей выпускной квалификационной работе применяются термины с соответствующими определениями и сокращениями, установленные нормативным документом ГОСТ Р 1.12-2004. Стандартизация в Российской Федерации. Термины и определения:

* 1. **Информационная система (ИС)** – это система, реализующая информационную модель предметной области, чаще всего – какой-либо области человеческой деятельности.
  2. **Контент** – в переводе с английского «содержание». В области веб-разработки контентом является содержимое веб-страницы, такое как текст, аудио- и видеоинформация. Применительно к области музыкальных рекомендательных систем под контентом рассматривают характеристики, присущие объекту.
  3. **Машинное обучение** – обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться.
  4. **Фреймворк** – это программная оболочка, позволяющая упростить и ускорить решение типовых задач, характерных для данного языка программирования.
  5. **Микрофреймворки** – минималистичные каркасы веб-приложений, сознательно предоставляющих лишь самые базовые возможности.
  6. **JSP (JavaServer Pages)** – технология, позволяющая веб-разработчикам создавать содержимое, которое имеет как статические, так и динамические компоненты.
  7. **Холодный старт** – потенциальная проблема компьютерных информационных систем, которая заключается в том, что система не может сделать никаких выводов для пользователей или элементов, о которых она еще не собрала достаточную информацию.

3 Сокращения

В данной выпускной квалификационной работе использованы следующие сокращения:

1. **ИС** – информационная система.
2. **БД** – база данных.
3. **MVC (Model, View, Controller)** – модель, представление, контроллер.
4. **МО (ML)** – машинное обучение.
5. **kNN (k-NN, K-Nearest Neighbour)** – метод k-ближайших соседей.
6. **SVM (Support Vector Machines)** – метод опорных векторов.
7. **DFD (Data Flow Diagram)** –– диаграмма потоков данных.

4 Анализ существующих рекомендательных систем

4.1 Развитие рекомендательных систем

Теоретические основы рекомендательных систем были заложены много лет назад в области машинного обучения. В 50-е годы в этой области науки были сформулированы математические подходы и описаны модели самообучающихся алгоритмов, которые до сих пор лежат в основе всех решений. В начале 1990-ых коллаборативная фильтрация стала применяться как решение для борьбы с избыточной информацией в вебе. Tapestry  
(экспериментальный почтовый сервис) стал одной из первых систем,  
использующих данный подход: она позволяла пользователю создавать вручную запросы, основанные на мнениях или действиях других пользователей. Такие манипуляции требовали определенных действий от пользователей, но с другой стороны, это позволяло пользователям определить актуальность переписки для себя, опираясь на реакцию других участников переписки.

В дальнейшем стали появляться системы фильтрации, которые  
автоматически определяли релевантные мнения и обобщали их для  
представления рекомендаций. В программном компоненте GroupLens  
использовался данный метод для нахождения статей в сети Usenet, которые  
могли бы быть интересны конкретному пользователю. Пользователям  
предлагалось оценить статьи, а система, в свою очередь, объединяла их с  
оценками других пользователей для предоставления персонализированных  
результатов.

В то же время рекомендательные системы стали предметом повышенного интереса в вопросах взаимодействия человека и компьютера, а также в области машинного обучения и информационного поиска. Вследствие этого, рекомендательные системы все чаще стали находить применение в областях музыкальных, кино- и множества других рекомендаций. За пределами ИТ-индустрии рекомендательные системы все чаще стали появляться в сфере маркетинга как один из способов увеличения числа продаж.

В конце 1990-ых стали появляться коммерческие рекомендательные  
продукты. Пожалуй, самым известным интегратором подобных технологий того времени является Amazon.com. Основываясь на истории покупок, истории просмотров и текущем просматриваемом товаре, система делала предположения о тех продуктах, которые могли бы быть интересны пользователю. После успеха Amazon многие другие представители сегмента электронной коммерции обратили свое внимание на рекомендательные решения, а некоторые ИТ-компании как NetPerceptions и Strands построили свой бизнес вокруг внедрения систем основанных на рекомендациях в крупные онлайн-магазины. В 2006 году рекомендательные алгоритмы привлекли к себе дополнительное внимание, когда компания Netflix запустила соревнование Netflix Prize, целью которого было создание алгоритма рекомендаций, который смог бы улучшить результат действующего внутреннего алгоритма CineMatch в тестах на 10%. Это вызвало шквал активности, как в академических кругах, так и  
среди любителей. Объявленная победителям премия в размере одного миллиона долларов демонстрирует ценность, которая кроется в изучении  
рекомендательных алгоритмов для крупных компаний. В то же время компания Google представила свою систему персонализации  
новостей Google News на основе истории кликов пользователей. В данном случае новостные статьи рассматриваются как объекты интереса, а клики пользователей как присвоение положительного рейтинга новостной статье[1].

4.2 Обзор существующих решений

Рекомендательные системы уже интегрированы во многие веб-приложения, которые ежедневно широко используются миллионами пользователей. Рассмотрим существующие веб-ресурсы, использующие различные рекомендательные механизмы.

YouTube - это онлайн-платформа для видео, принадлежащая Google. В общей сложности пользователи просматривают более одного миллиарда часов видео на YouTube каждый день, и каждую минуту на серверы YouTube загружаются сотни часов видеоконтента. YouTube предоставляет несколько способов просмотра видео, включая веб-сайт, мобильные приложения и позволяет другим веб-сайтам встраивать их. Доступный контент включает музыкальные клипы, видеоклипы, короткометражные и документальные фильмы, аудиозаписи, трейлеры к фильмам, прямые трансляции и видеоблоги. Большая часть контента создается отдельными людьми, но медиа-корпорации также публикуют видео. Помимо просмотра и загрузки, зарегистрированные пользователи могут комментировать видео, оценивать их, создавать плейлисты и подписываться на других пользователей. Данный сервис использует контент-ориентированный и коллаборативный подход для рекомендации контента для пользователей.

Amazon, один из крупнейших сайтов электронной коммерции, использует рекомендации, основанные на содержании. Когда пользователь выбирает товар для покупки, Amazon рекомендует другие товары, приобретенные другими пользователями на основе этого исходного товара (с помощью матрицы покупки следующего товара на основе сходства с предыдущей покупкой). Amazon запатентовал этот подход, называемый совместной фильтрацией элементов (коллаборативная фильтрация от элемента к элементу).

LinkedIn – это социальная сеть, ориентированная на бизнес. Встроенный механизм рекомендаций предлагает пользователю возможности для привлечения людей, которых он может знать, рабочие места, которые могут привлечь его к сотрудничеству, группы, которые он может привлекать, группы, которые могут быть ему интересны. Специализированная система коллаборативной фильтрации LinkedIn, основана на технологиях Apache Hadoop.

Среди сервисов, основанных на музыкальных рекомендациях, можно выделить следующие:

Яндекс.Музыка – сервис потоковой передачи музыки, разработанный Яндекс. Пользователи выбирают музыкальные композиции, альбомы, коллекции музыкальных треков для потоковой передачи на свое устройство по запросу и получают персональные рекомендации. Сервис также доступен в виде мобильного приложения с версиями, совместимыми с iOS, Android. Услуга доступна в Армении, Азербайджане, Беларуси, Грузии, Израиле, Казахстане, Кыргызстане, Молдове, России, Таджикистане, Туркменистане и Узбекистане. Подписка может быть оплачена только в поддерживаемых странах, указанных выше, но в этом случае услуга будет доступна во всех других странах. По состоянию на октябрь 2017 года в Yandex Music было доступно более 40 миллионов музыкальных треков. Около 20 миллионов человек пользуются сервисом не реже одного раза в месяц. Самая популярная функция Yandex Music – умные плейлисты, которые обновляются ежедневно для каждого пользователя и содержат недавно воспроизведенные треки, музыку, похожую на их любимую, и разнообразные треки, основанные на вкусах пользователя. Yandex.Music использует смешанную систему рекомендаций[2].

Last.fm создает музыкальную "станцию" из рекомендованных песен, отслеживая, какие группы и отдельные треки слушает пользователь на регулярной основе. Last.fm воспроизводит произведения из других библиотек пользователей со схожими музыкальными вкусами. Этот сервис использует поведение пользователя и является примером совместной фильтрации.

Pandora использует более 400 метаданных песен и исполнителей, предоставленных проектом Music Genome Project, чтобы создать «станцию», которая воспроизводит музыку с похожими атрибутами. Кроме того, для уточнения результатов «станции» используется обратная связь от пользователя, которая обесценивает определенные атрибуты, когда пользователю не понравилась определенная песня и увеличивает вклад других атрибутов, когда пользователю нравится песня. В этом сервисе используется контент-ориентированный подход[3].

Spotify – шведский провайдер потокового аудио и мультимедийных услуг, основанный в 2006 году Даниэлем Эком. Это один из крупнейших в мире поставщиков услуг потоковой передачи музыки с более чем 356 миллионами активных пользователей в месяц, включая 158 миллионов платных подписчиков, по состоянию на март 2021 года. Spotify предлагает музыку и подкасты с ограничением авторских прав в цифровом формате, в том числе более 70 миллионов песен, от звукозаписывающих лейблов и медиа-компаний. Пользователи могут искать музыку по исполнителю, альбому или жанру, а также создавать, редактировать и обмениваться плейлистами. Данный сервис использует несколько подходов для рекомендации, в числе которых метод коллаборативной фильтрации и контент-ориентированный подход, объединенные в единую систему[4].

Deezer – французский онлайн-сервис потоковой передачи музыки. Он позволяет пользователям слушать музыкальный контент от звукозаписывающих компаний, в том числе Universal Music Group, Sony Music и Warner Music Group (принадлежит Access Industries , материнской компании Deezer), а также подкасты на различных устройствах онлайн или офлайн. Созданный в Париже, Франция, Deezer в настоящее время имеет 73 миллиона лицензионных треков в своей библиотеке, с более чем 30 000 радиоканалов, 14 миллионами активных пользователей в месяц и 7 миллионами платных подписчиков по состоянию на январь 2019 года. Данный сервис использует метод коллаборативной фильтрации.

4.3 Классификация рекомендательных систем

В зависимости от типов данных можно выделить методы коллаборативной фильтрации и методы, основанные на контенте.

Методы коллаборативной фильтрации, в качестве входных типов данных используют информацию о взаимодействии пользователей с интересующими объектами, а методы, основанные на контенте, используют информацию, которую предоставляют сами пользователи. Например, атрибуты, указанные в профиле, ключевые слова или другую информацию об объектах.

Существует также другой тип рекомендательных систем. Это рекомендательные системы, основанные на знаниях, где рекомендации основаны на четко определенных пользовательских требованиях.

Некоторые рекомендательные системы могут использовать несколько подходов для рекомендации. Такие системы обычно называют гибридными системами. Они объединяют сильные стороны разных подходов для формирования рекомендаций, которые могут работать более эффективно в узкоспециализированных системах.

4.3.1 Рекомендательные системы на основе коллаборативной фильтрации

Коллаборативная фильтрация (англ. Collaborative filtering) генерирует рекомендации на основе модели предыдущего поведения пользователя. Она использует систему пользовательских оценок. Эта модель может быть построена только при использовании предыдущих оценок данного пользователя или, что более эффективно, на основе поведения других пользователей, которые интересовались такими же объектами[5].

В качестве входных данных коллаборативная фильтрация использует множество пользователей и множество объектов интереса. Отношения между пользователями и объектами интереса обычно выражаются в виде оценок, которые пользователи присваивают объектам. Эти данные потом используются для прогнозирования оценок, которые пользователь может присвоить еще неоцененным объектам интереса. Первое, что делает система, это определяет ближайших соседей, а затем экстраполирует рейтинг этого пользователя из оценок пользователей, со схожими интересами[6].

Подход коллаборативной фильтрации к алгоритмам рекомендаций включает сбор «большого количества информации о поведении, действиях или предпочтениях пользователей и прогнозирование того, что пользователям понравится, на основе их сходства с другими пользователями». Ключевым моментом, который следует отметить в отношении этого метода, является то, что сам элемент или его функции, которые рекомендуются, не анализируются. Сбор данных при этом подходе включает как явный сбор данных, например, просьбу пользователя оценить элемент, так и неявный сбор данных, например, ведение записей о том, как часто и как долго пользователь просматривает элемент[7].

Два основных подхода к коллаборативной фильтрации – это пользовательская коллаборативная фильтрация и коллаборативная фильтрация на основе контента. Оба варианты предсказывают, в какой степени пользователь будет заинтересован объектами, которые еще не были им оценены. Пользовательская фильтрация определяет k-ближайших соседей активного пользователя и на основе этих ближайших соседей вычисляет прогноз пользователя для конкретного объекта интереса. В отличие от пользовательской фильтрации, коллаборативная фильтрация на основе контента для текущего объекта ищет соседей, получивших аналогичные оценки. Подход к фильтрации на основе содержимого отличается от подхода совместной фильтрации, поскольку он фильтрует на основе анализа как рекомендуемого элемента, так и пользователя. Фильтрация на основе содержимого внимательно изучает фактический элемент, чтобы определить, какие функции наиболее важны для выработки рекомендаций и как эти функции взаимодействуют с предпочтениями пользователя[8].

К достоинствам такого подхода можно отнести теоретически высокую точность, а к минусам: высокий порог входа – ничего не зная об интересах пользователя, рекомендации практически бесполезны.

4.3.2 Рекомендательные системы, основанные на контенте

Фильтрация на основе контента (англ. Content-based filtering) основана на предположении о том, что интересы пользователя постоянны в течение времени.

Например, у пользователей, интересующихся определенной темой, интересы каждый день не меняются. Поэтому они скорее всего будут интересоваться этой темой в ближайшее время. Такая фильтрация обычно использует множество пользователей и множество категорий (или ключевых слов) в качестве входных данных, с помощью которых помечаются объекты интереса. Задачей систем рекомендаций, основанных на содержании, является вычисление набора объектов, которые будут наиболее близки категориям, которые интересуют текущего пользователя.

Основным подходом к фильтрации на основе содержимого является сравнение уже просмотренных пользователем объектов с новыми объектами, которые потенциально могут быть рекомендованы пользователю. Базовый механизм определения таких сходств заключается в извлечении ключевых параметров непосредственно из контекста, содержащего интересующий объект, или из метаинформации, которой был аннотирован информационный объект[9].

К преимущества такого метода можно отнести:

* возможность давать рекомендации новым пользователям системы;
* возможность рекомендовать объекты, еще не получившие оценок.

К недостаткам данного метода относится меньшая точность рекомендации.

4.3.3 Рекомендательные системы, основанные на знаниях

По сравнению с подходами, основанными на коллаборативной и контентной фильтрации, рекомендательные системы, основанные на знаниях, независимы от оценки объектов или их описания с использованием метаданных. Иногда предыдущий подход (основанный на содержании) определяют, как частный случай систем, основанных на знаниях. Дополнительная информация об объектах позволяют рекомендовать их, не полагаясь на «схожесть» чего-либо, а использовать более сложные условия. В рекомендательных системах, основанных на знаниях, в качестве входных данных используют: (а) набор правил (ограничений) или показателей сходства и (б) множество объектов интереса. В зависимости от указанных требований пользователя правила описывают, какие объекты возможно рекомендовать пользователю. Текущий пользователь формулирует свои предпочтения в терминах свойств элемента, которые, в свою очередь, представлены с точки зрения правил (ограничений).

К плюсам данных систем можно отнести:

* возможность исключения из рекомендации объектов, которые уже не актуальны для данного пользователя.

А к недостаткам относится высокая сложность построения правил и сбора данных[10].

5 Исследование и описание алгоритма рекомендации

Так как для коллаборативного подхода необходима большая база пользовательских оценок и предпочтений, для построения рекомендательной системы был выбран контентно-ориентированный подход, который позволяет давать рекомендации сразу же, как только пользователь вошел в систему.

5.1 Набор данных

Для построения рекомендательной системы будем использовать набор данных, опубликованных Spotify[11]. Данный набор включает в себя информацию о более чем 340 тысячах музыкальных композиций и не содержит аудиоданных. Он включает только музыкальные характеристики. Их название и описание приведено в таблице 5.1.

Таблица 5.1 – Музыкальные характеристики

|  |  |
| --- | --- |
| **Характеристика** | **Содержащаяся информация** |
| Темп | Темп композиции. |
| Валентность | Эта величина описывает музыкальную позитивность, передаваемую песней. Песни с высокой валентностью звучат более позитивно (т.е. Они передают счастье, радость или эйфорию), а песни с низкой валентностью звучат негативнее (т.е. Они печальные, депрессивные или гневные). |
| Инструментальность | Прогнозирует, что в треке нет вокала. В этом контексте звуки «оу» и «а-а-а» считаются инструментальными. Чем выше значение инструментальности, тем выше вероятность того, что в треке не содержится голоса. |

*Продолжение таблицы 5.1*

|  |  |
| --- | --- |
| Энергия | Характеризует критерий восприятия «яркости» и «активности» песни. |
| Танцевальность | Описывает пригодность дорожки для танцев на основании таких музыкальных элементов, как темп, стабильность ритма, сила долей и общее постоянство. |
| Живое исполнение | Чем больше значение, тем выше вероятность того, что песня исполнялась вживую. |
| Акустичность | Мера уверенности в том, что композиция является акустической, а не искусственной. |
| Текст | Мера присутствия текста в песне, чем выше, тем больше присутствие текста в песне. |
| Громкость | Усредненная громкость музыкальной композиции. |

5.2 Выбор метода машинного обучения

Для создания рекомендации необходимо выбрать наиболее похожие на заданную пользователем композицию. Эту проблему решает задача классификации музыки. Важным этапом при решении задачи классификации музыки является выбор метода машинного обучения, который будет применяться к векторному представлению музыкальных характеристик композиции, для нахождения наиболее похожих на нее. В настоящее время разработано множество методов МО, которые применяются при решении широкого круга задач. Многие из этих методов МО применяются для решения задач классификации. Рассмотрим основные методы.

5.2.1 Наивный байесовский классификатор

Метод Байеса основан на анализе совместных распределений признаков композиций и категорий. Композиции сопоставляется наиболее вероятная категория по формуле 5.1.

(5.1)

В задаче классификации текстов метод Байеса применяется отдельно для каждой категории и принимается решение, принадлежит композиция категории или нет. Апостериорная вероятность принадлежности музыкальной композиции к классу вычисляется по формуле Байеса (формула 5.2), связывающей априорную вероятность с апостериорной.

(5.2)

Подставляя 5.2 в 5.1, получаем:

(5.3)

Так как знаменатель не зависит от категории, его можно сократить (формула 5.4).

(5.4)

Условные вероятности можно вычислить в предположении условной независимости переменных *x1, x2, …, xn*. В этом случае, формула для определения наиболее вероятной категории будет выглядеть следующим образом (формула 5.5).

(5.5)

Конечно, предположение о независимости переменных *x1, x2, …, xn*, является слишком сильным (поэтому метод Байеса иногда называют «наивным» – naive bayes classifier). На самом деле это предположение практически никогда не выполняется[12].

5.2.2 Метод k-ближайших соседей

Метод k-ближайших соседей (k-nearest neighbours, k-NN), в отличие от других, не требует фазы обучения. Для того чтобы найти данные, релевантные композиции *d*, эта композиция сравнивается со всеми композициями из обучающей выборки. Для каждой композиции *e* из обучающей выборки, находится расстояние – косинус угла между векторами признаков, вычисляемое по формуле 5.6.

(5.6)

Далее из обучающей выборки выбираются *k* композиций, ближайших к *d* (*k* – параметр). Для каждой композиции вычисляется релевантность по формуле 5.7.

(5.7)

Композиции с релевантностью выше некоторого заданного порога считаются соответствующими заданной композиции. Параметр *k* обычно выбирается в интервале от 1 до 100. Данный метод показывает довольно высокую эффективность, но чем больше размер данных, тем больше вычислительных затрат он требует[13].

5.2.3. Метод опорных векторов

Метод опорных векторов (Support Vector Machines, SVM) разработан В. Вапником на основе принципа структурной минимизации риска – одновременного контроля количества ошибок классификации на множестве для обучения и «степени обобщения» обнаруженных зависимостей. В наиболее простом случае метод SVM заключается в нахождении гиперплоскости в пространстве признаков, разделяющей *Rn* на две части: в одной находятся все положительные примеры (композиции, удовлетворяющие условию), а в другой – все отрицательные примеры (композиции, не удовлетворяющие). При этом среди всех таких гиперплоскостей находится та, для которой минимальное расстояние (зазор) до ближайших примеров максимально.

Нахождение оптимальной плоскости методом SVM сводится к решению оптимизационной задачи с линейными ограничениями типа равенств и неравенств по формуле 5.8.

(5.8)

Здесь *K(xi, xj)* – функция ядра SVM, которая в простейшем случае равна евклидову скалярному произведению векторов *xi* и *xj*. Существуют также обобщения метода SVM для случая, когда разделяющей гиперплоскости не существует. В этом случае SVM ищет разделяющую гиперплоскость, одновременно минимизируя количество ошибок и максимизируя зазор между разделяющей гиперплоскостью и ближайшими примерами. Метод SVM работает с абстрактной векторной моделью предметной области. Это позволяет применять SVM для решения различных задач машинного обучения[14].

5.2.4 Результаты анализа методов машинного обучения

После применения данных классификаторов на тестовых данных, лучший результат в задаче классификации музыкальных композиций показал метод k-ближайших соседей, что видно из таблицы 5.2. Поэтому в рекомендательной системе будем использовать именно его. Однако стоит учесть, что при увеличении количества данных, этот классификатор может быть не так эффективен, и для использования в более масштабных информационных системах стоит обратить внимание на другие классификаторы.

Таблица 5.2 – Результаты

|  |  |
| --- | --- |
| Классификатор | Точность |
| Наивный байесовский классификатор | 0,89 |
| kNN | 0,93 |
| SVM | 0,91 |

5.3 Алгоритм рекомендации

Для разрабатываемой рекомендательной системы будем использовать метод k-ближайших соседей, так как он показал наилучший результат классификации на тестовых данных. Так как выбранные с помощью kNN композиции могут иметь одинаковую релевантность, для уточнения результатов классификатора построим рейтинг, используя такие музыкальные характеристики как темп и тональность.

Величина разницы темпов у двух музыкальных композиций будет равна отношению разницы в темпах к значению темпа выбранной композиции. Разница между темпами считается без учета знака и рассчитывается по формуле 5.9.

, (5.9)

где *Tempo –* это относительная разница в темпе между музыкальными композициями *а* и *b. Tempo* принимает значение от 0 до 1, где 0 означает, что у композиций одинаковый темп, а 1 означает, что темпы композиций различаются в два раза.

Далее музыкальные композиции сравниваются по тональностям и рассчитывается по формуле 5.10.

(5.10)

где *V* – относительная разница в тональности композиций.

Итоговая относительная разница между двумя музыкальными композициями по результатам сравнения звуковых характеристик будет считаться следующий образом по формуле 5.11.

, (5.11)

где *Rate –* итоговое значение разницы между двумя музыкальными композициями, принимает положительное значение, где, чем больше число, тем сильнее разница в звучании между композициями, а 0 означает, что композиции идентичны по звучанию.

Итоговое значение разницы *Rate* будет присваиваться для каждого из найденных с помощью метода kNN значений и на его основе будет строиться рейтинг, из которого будут выбираться музыкальные композиции для рекомендации, от наибольшего соответствия к наименьшему.

6 Разработка требований к информационной системе

Определение требований к ИС – ответственный этап программного проекта. Формат проекта должен соответствовать требованиям к ПО, собранным командой разработчиков и аналитиков. Составление требований для реализации программного продукта следует разделить на две части: пользовательские и системные требования. Рассмотрим их далее.

6.1 Формирование пользовательских требований

Для начала необходимо определить пользовательские функции и сценарии, которые могут возникнуть в ходе работы с информационной системой.

Система должна обладать удобным и интуитивно понятным интерфейсом, а именно:

* иметь строку для ввода поискового запроса;
* иметь поля для вывода списка рекомендаций;
* рекомендованный список должен удовлетворять пользовательским вкусам;
* рекомендации должны иметь ссылку на ресурс, с помощью которого можно будет оценить качество рекомендации.

6.2 Формирование системных требований

Системные требования – это детализированное описание системных ограничений и функций. К основным системным требованиям для разрабатываемой информационной системы относятся следующие требования:

* система должна быть размещена на внутренних серверах компании. также необходимо обеспечить отказоустойчивость системы на случай непредвиденных сбоев;
* необходимо обеспечить бесперебойный доступ к информационной системе;
* параметры сервера (таблица 6.1);

Таблица 6.1 – Параметры сервера

|  |  |
| --- | --- |
| Количество серверов | 2 шт. |
| Количество вычислительных потоков процессоров | Не менее 4 |
| Тактовая частота процессора | Не менее 2,5 Гц |
| Оперативная память | Минимум 8 Гб |
| Свободное дисковое пространство | Минимум 10 Гб |
| Пропускная способность сети | Минимум 100 Мбит/сек |

* система должна легко масштабироваться, и иметь возможность доработки и интеграции с другими системами;
* система должна иметь интуитивно понятный интерфейс без избыточного функционала.

7 Проектирование информационной системы

Одной из важных частей при создании любого программного обеспечения является проектирование. Правильно построенная модель приложения позволяет не только показать составные части системы, но и описать бизнес-процессы для команды.

7.1 Функциональная схема проектируемой информационной системы

Для построения функциональной схемы приложения воспользуемся методологией IDEF0.

Исходя из рисунка 7.1 в состав функциональной схемы входят следующие виды компонентов:

1. Входящие – входные данные, которые ставят определенную задачу. К входным данным ИС относится ввод поискового запроса пользователем.
2. Исходящие – результат выполнения бизнес-процессов. В информационной системе исходящим компонентом является получения списка рекомендаций на основе поискового запроса пользователя.
3. Управляющие – механизмы управления. Например, положения, инструкции, постановки и так далее. К управляющим компонентами системы относятся данные и полученные музыкальные характеристики.
4. Механизмы – необходимые для проведения работы ресурсы. К данному пункту относится алгоритм для формирования списка рекомендаций.

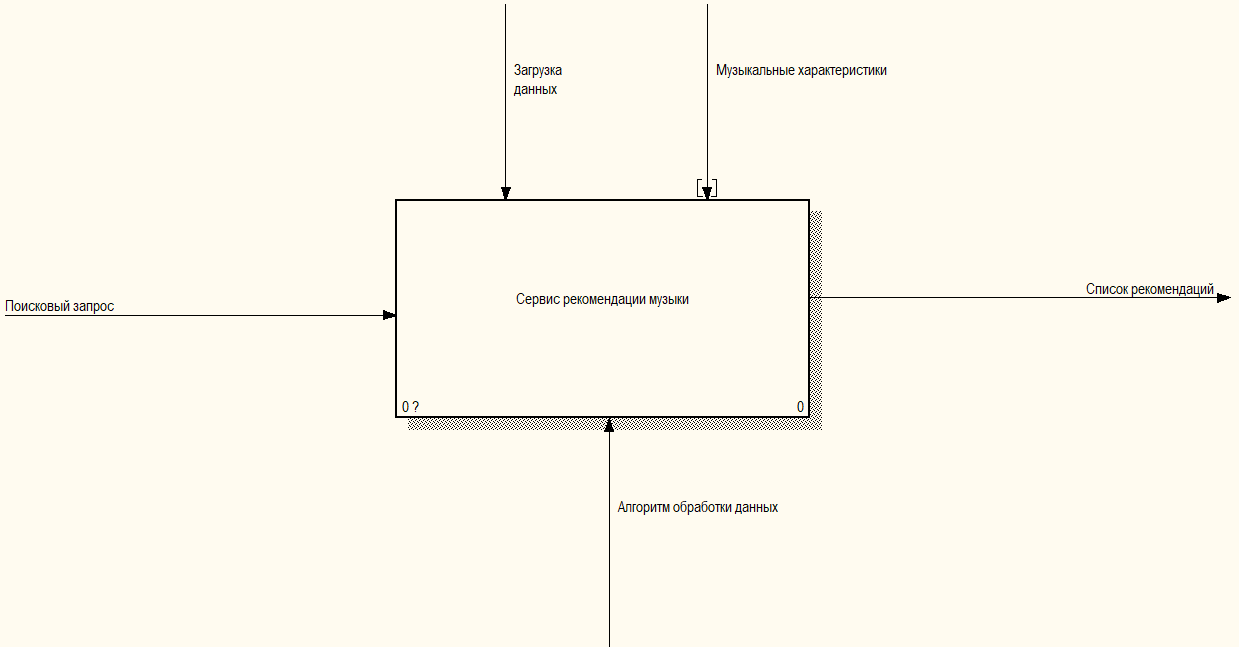


Рисунок 7.1 – Схема IDEF0. Основной блок

Рисунок 7.2 показывает пошаговую работу информационной системы рекомендации музыки.

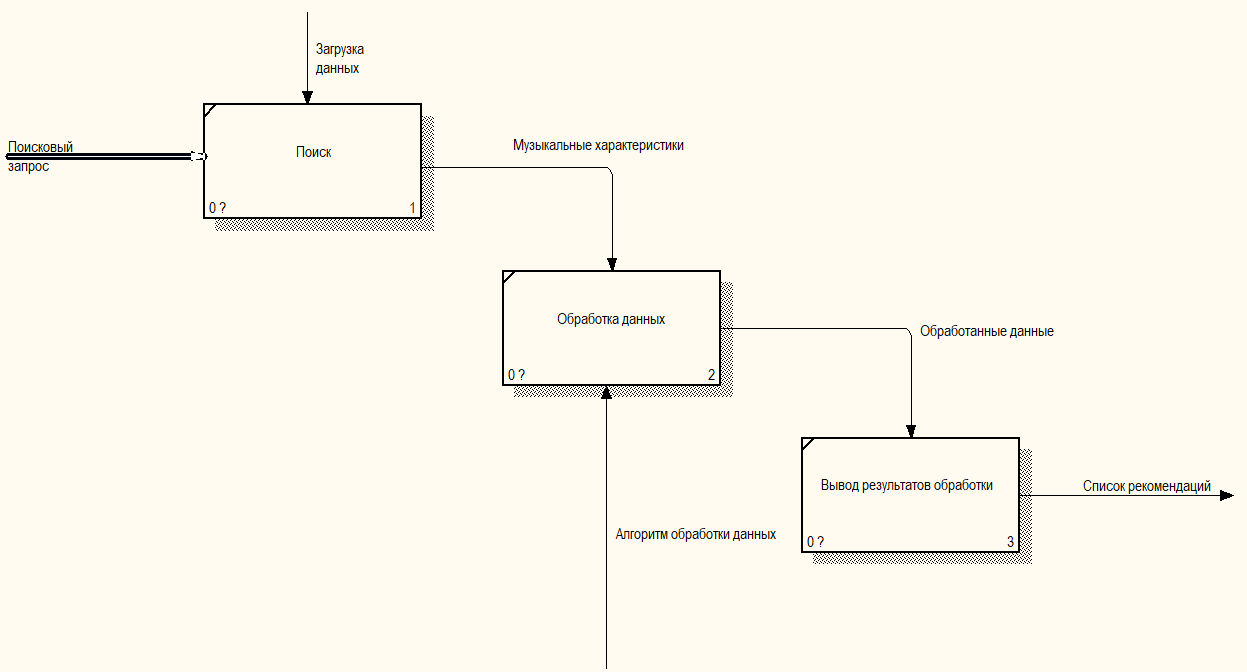


Рисунок 7.2 – Схема декомпозиции IDEF0. Декомпозиция

Схема декомпозиции состоит из трех этапов, связанных между собой:

* поиск – на этом этапе в системе рекомендаций происходит обработка поискового запроса и загружаются музыкальные характеристики для найденного музыкального трека;
* обработка данных – на этом этапе полученные музыкальные характеристики обрабатываются с помощью алгоритма для получения рекомендаций;
* вывод результата обработки – на этом этапе формируется список рекомендаций и выводится на экран для пользователя.

7.2 Объектно-ориентированная модель информационной системы

Понятие объектно-ориентированного моделирования напрямую связано с объектно-ориентированной парадигмой, в основе которой лежат классы и объекты. Для описания взаимодействия классов и объектов используется язык UML. Согласно вышеперечисленным требованиям, опишем нашу модель графически с помощью UML диаграмм. В системе существует одна пользовательская роль:

1. Пользователь – использует информационную систему для получения списка рекомендаций.

Исходя из перечисленных ролей построим диаграмму прецедентов для нашей информационной системы. Полученная диаграмма прецедентов показана на рисунке 7.3.

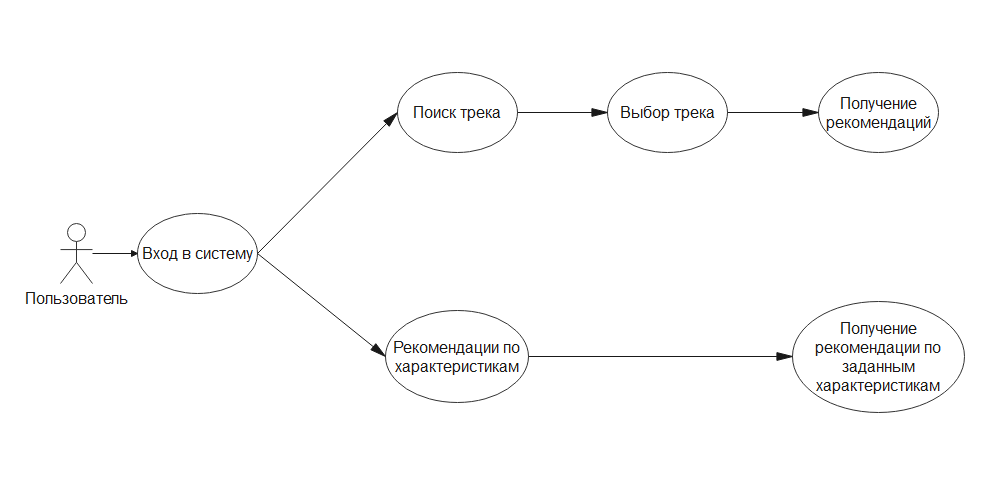


Рисунок 7.3 – Диаграмма прецедентов

В заключение составим диаграмму состояний (рисунок 7.4), на которой отобразим пошаговый алгоритм взаимодействия пользователя с системой рекомендаций. Диаграмма учитывает, как положительные исходы событий, так и отрицательные. Такой подход позволяет проработать возможные сценарии действий пользователя.

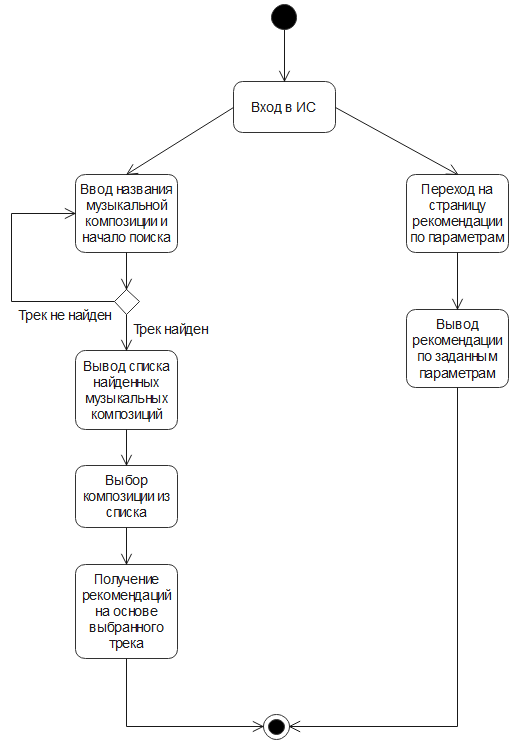


Рисунок 7.4 – Диаграмма состояний ИС

7.3 Проектирование БД

База данных MySQL, спроектированная для информационной системы подбора музыкальных композиций содержит следующую таблицу (таблица 7.1).

Таблица 7.1 – Пользователи

|  |  |  |
| --- | --- | --- |
| **Поле** | **Назначение** | **Тип поля** |
| id | Первичный ключ | int |
| fname | Имя пользователя | varchar(65) |
| lname | Фамилия пользователя | varchar(65) |
| uname | Логин пользователя | varchar(65) |
| pswd | Пароль пользователя | varchar(65) |
| emailid | Электронная почта пользователя | varchar(65) |
| country | Страна пользователя | varchar(65) |

Таблица 7.1 содержит информацию о пользователях информационной системы.

8 Разработка информационной системы

8.1 Используемые технологии

В этом разделе описаны главные принципы, шаблоны и инструменты, используемые при разработке веб-приложения.

Web-приложение разрабатывалось на платформе Spring Framework для Java и следует паттерну проектирования MVC. С помощью фреймворка Python Flask был реализован API, который производит обработку данных и находит рекомендации. Опишем их.

8.1.1 Фреймворк Spring

Среди библиотек, которые присутствуют в языке Java для разработки веб приложений, был выбран Spring Framework, являющейся на данный момент одним из самых популярных и современных инструментов в мире Java.

Spring Framework обеспечивает комплексную модель разработки и конфигурации для современных бизнес-приложений на Java – на любых платформах. Ключевой элемент Spring – поддержка инфраструктуры на уровне приложения.

Этот фреймворк предлагает последовательную модель и делает её применимой к большинству типов приложений, которые уже созданы на основе платформы Java. Считается, что Spring Framework реализует модель разработки, основанную на лучших стандартах индустрии, и делает её доступной во многих областях Java.

8.1.2 Шаблон MVC

Шаблон проектирования MVC–один из фундаментальных шаблонов, применяющихся при разработке веб-приложений. Он предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных независимых компонента – модель, представление и контроллер, так, чтобы изменения каждого компонента могли осуществляться независимо.

Модель – содержит основную логику и данные приложения.

Контроллер – обрабатывает входящий запрос от пользователя и передает его в модель, а затем, если данные изменяются, обновляет представление.

Представление – обеспечивает различные виды представления данных, которые получены из модели, но при этом не может влиять на нее. Полученное из модели представление возвращается пользователю.

8.1.3 Фреймворк Python Flask

Python – относительно простой и достаточно универсальный язык программирования, одним из направлений которого, является сфера веб-программирования.

Сам по себе он мало подходит для веб-разработки, но за счет использования специальных фреймворков можно создавать серверную логику сайта, приложения или игры.

Flask является микрофреймворком для создания веб-сайтов на языке Python. Создатель и основной автор – австрийский программист Армин Ронахер, начал работу над проектом в 2010 году.

8.2 Структура Java-приложения

Структура разработанной ИС представлена на рисунке 8.1.

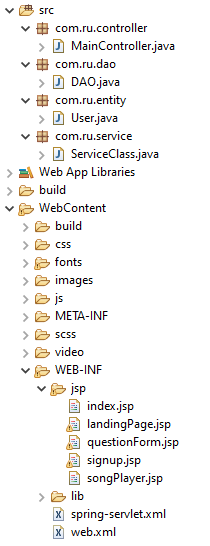


Рисунок 8.1 – Структура проекта

В каталоге com.ru.controller расположен контроллер MainController.java, который отвечает за взаимодействие между пользователем и системой. Он направляет данные от пользователя к системе и наоборот. Контроллер обрабатывает запрос пользователя, создаёт соответствующую модель и передаёт её для отображения в JSP. Код контроллера приведен в приложении А.

Страницы JSP расположены в каталоге WEB-INF/jsp и представляют собой страницы, которые отображаются пользователю. В таблице приведено их описание.

Таблица 8.1 – Описание страниц

|  |  |
| --- | --- |
| **Название** | **Описание** |
| index.jsp | Отображает страницу входа в ИС и содержит переход на страницу регистрации |
| signup.jsp | Отображает страницу регистрации |
| landingPage.jsp | Отображает страницу поиска и содержит переход на страницу ручного ввода музыкальных характеристик |
| questionForm.jsp | Отображает страницу с музыкальными характеристиками |
| songPlayer.jsp | Отображает страницу рекомендаций на основе найденного трека |

8.3 Структура API

Для обработки данных было разработано API с использованием Flask Framework. Данный API отвечает за обработку информации о музыке и формирование рекомендаций, которые отправляются в приложение Java и затем отображаются пользователю.

Модуль flaskAPI.py запускает сервер Flask, к которому впоследствии обращается Java-приложение. Код данного модуля приведен в приложении Б.

В модуле k\_nn.py реализован алгоритм k-ближайших соседей. Его код приведен в приложении В.

9 Описание приложения

9.1 Описание интерфейса пользователя

При входе на сайт пользователь попадает на страницу авторизации (рисунок 9.1). На данной странице находятся два поля – «логин» и «пароль», в которые пользователь вводит свои данные для входа в систему. Затем, для авторизации, пользователю необходимо нажать кнопку «Вход».

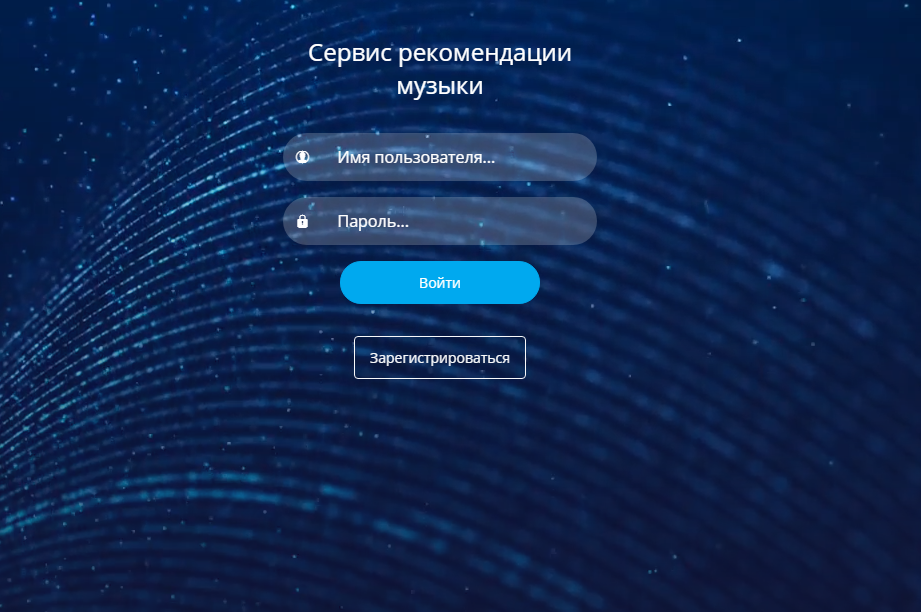


Рисунок 9.1 – Страница авторизации

Если пользователь впервые пользуется данной ИС, то ему необходимо зарегистрироваться. Перейти на страницу регистрации (рисунок 9.2) можно с помощью кнопки «Зарегистрироваться», расположенную на странице авторизации.

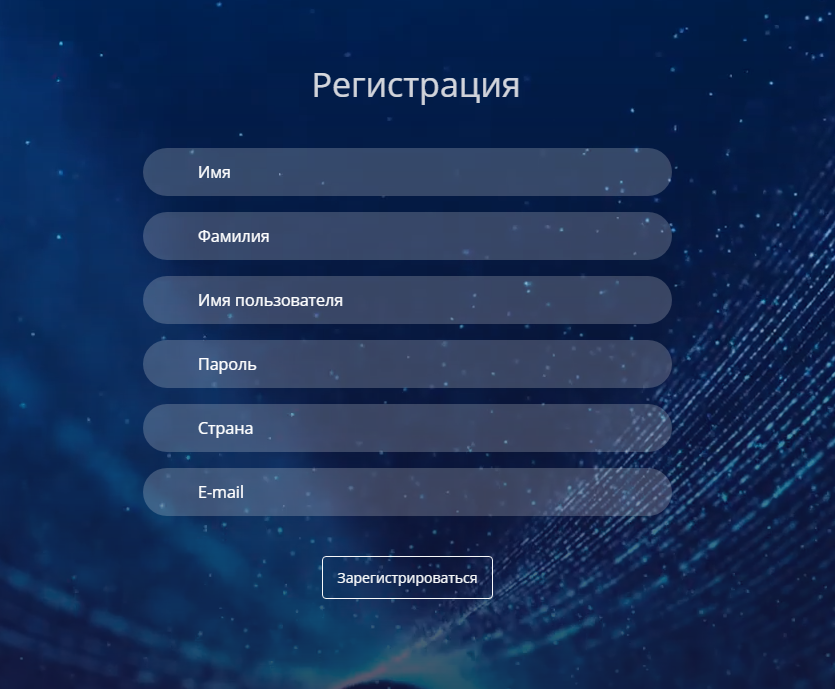


Рисунок 9.2 – Страница регистрации

Для того чтобы зарегистрироваться пользователю необходимо на странице регистрации ввести личные данные и нажать кнопку «Зарегистрироваться». После регистрации происходит перенаправление на страницу авторизации.

После авторизации в приложении открывается страница поиска музыкальных композиций (рисунок 9.3). Поиск музыки можно осуществлять по таким параметрам как «Название трека», «Название исполнителя», «Название альбома».

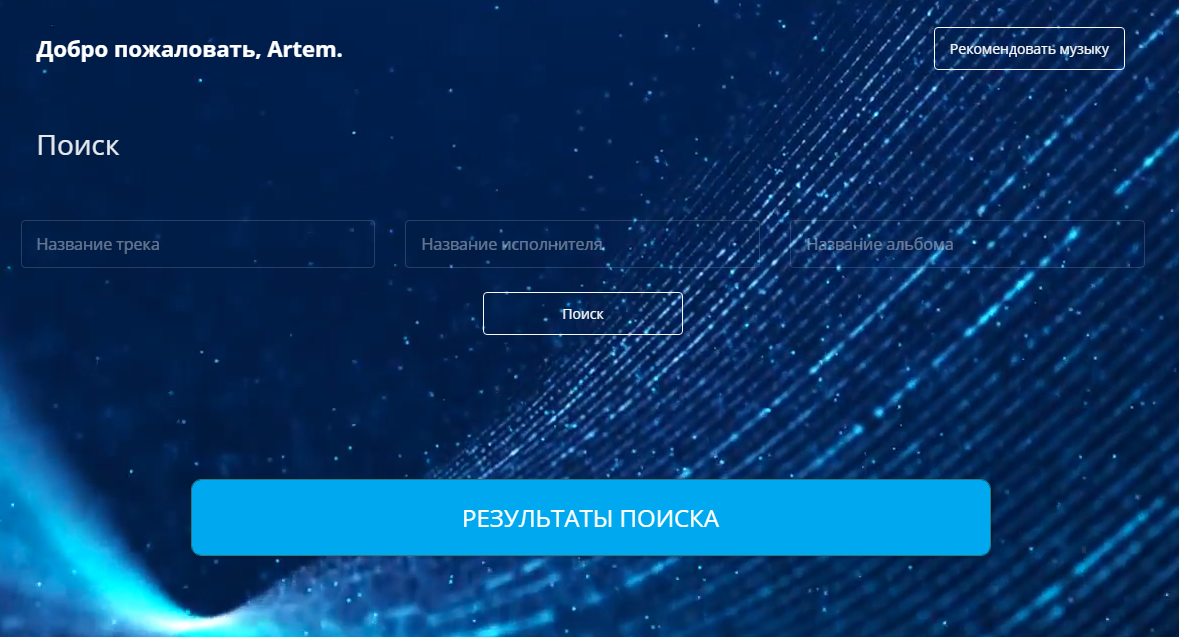


Рисунок 9.3 – Страница поиска

В таблице результаты поиска, выводятся треки, найденные в базе. Результат поискового запроса на песню «The Scientist» показан на рисунке 9.4.

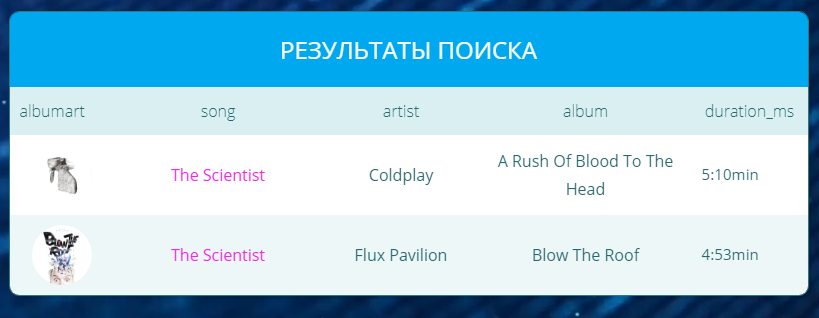


Рисунок 9.4 – Результаты поиска

Теперь пользователь может получить рекомендации на основе найденного трека. Для этого ему необходимо щелкнуть на название найденной песни и тогда откроется страница рекомендаций, показанная на рисунках 9.5 и 9.6. На данной странице расположен плеер, с помощью которого пользователь может прослушать найденный трек, а в нижней части страницы отображаются рекомендации, сделанные на основе найденного музыкального трека. При щелчке на рекомендованную песню ее также можно прослушать и получить рекомендации на ее основе.

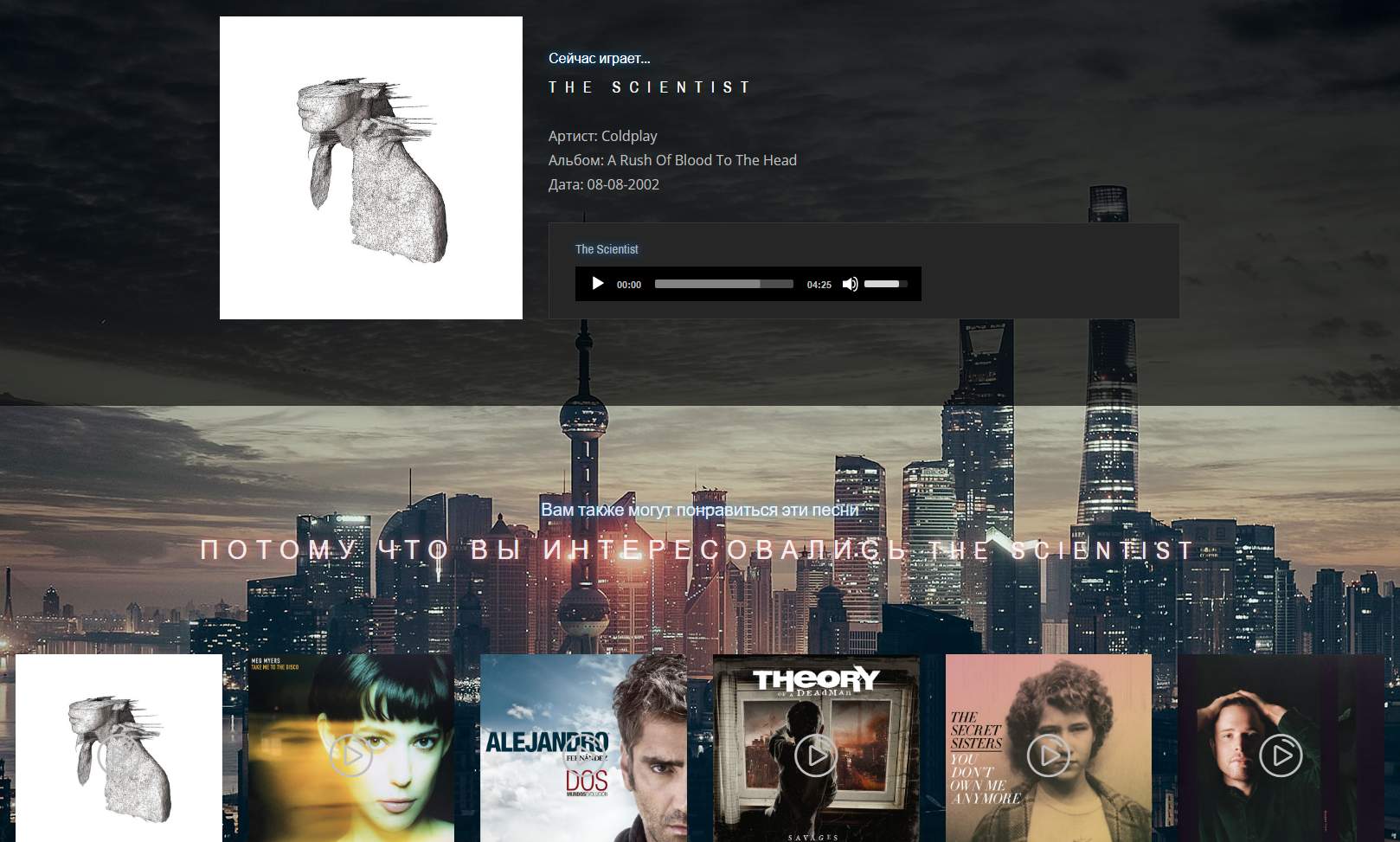


Рисунок 9.5 – Страница рекомендации(часть 1)

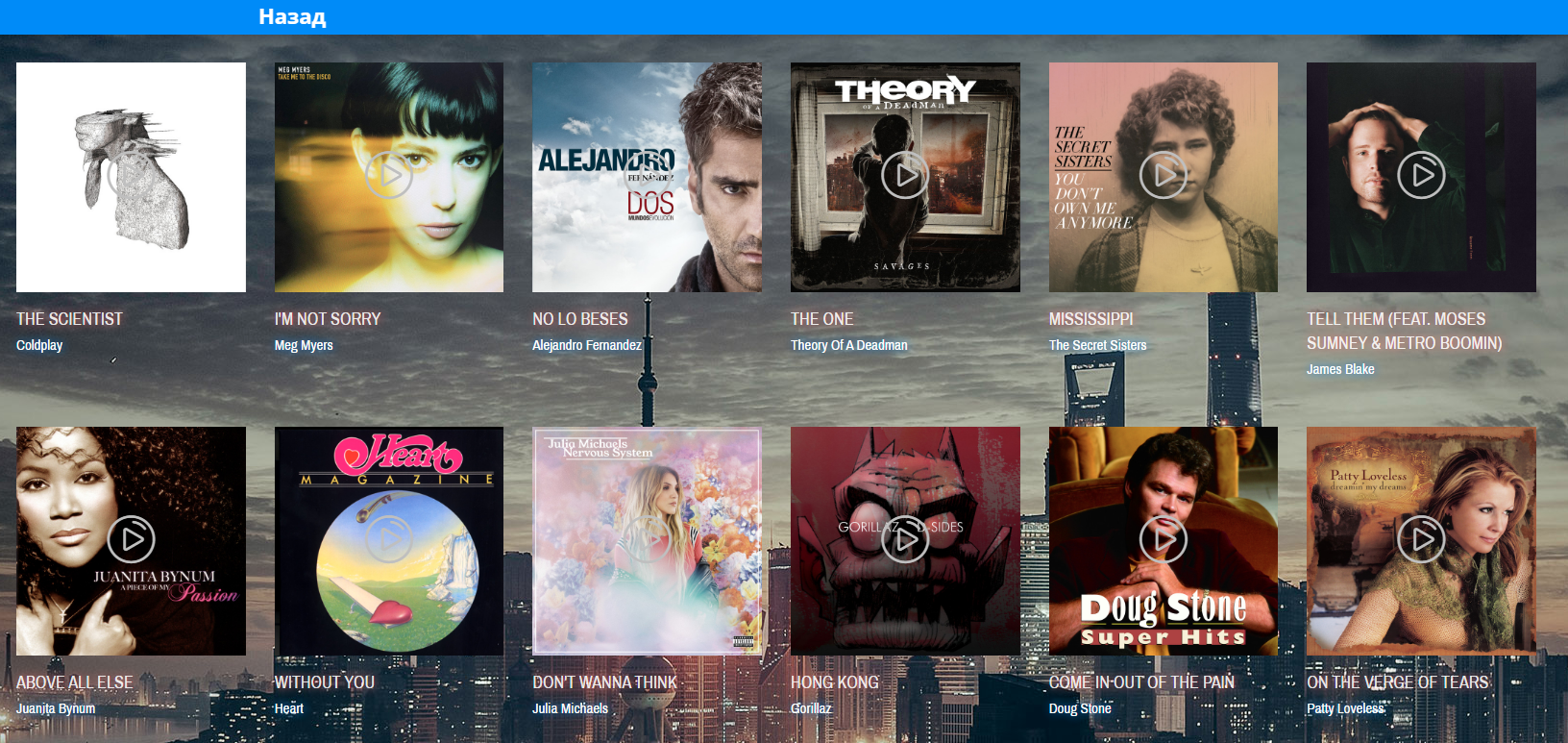


Рисунок 9.6 – Страница рекомендации(часть 2)

На странице поиска (рисунок 9.3) пользователь также может вручную ввести музыкальные характеристики и на их основе получить рекомендации. Для того чтобы это сделать, пользователю необходимо перейти на страницу ручного ввода музыкальных характеристик (рисунок 9.7), нажав на кнопку «Рекомендовать музыку». На данной странице приведены характеристики, описанные в разделе 5.1, а также их описание, которое можно открыть, нажав на характеристику.

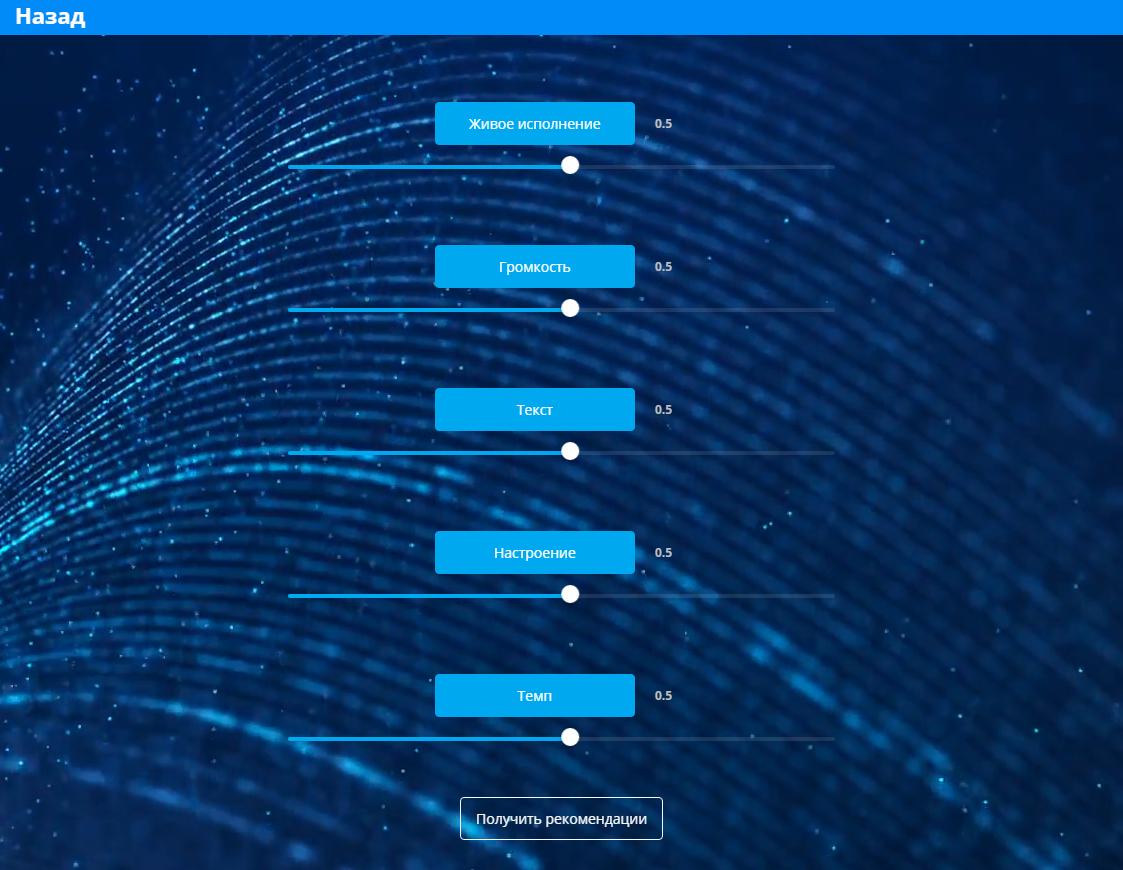


Рисунок 9.7 – Ручной ввод музыкальных характеристик

При нажатии на кнопку «Получить рекомендации», на основе заданных характеристик строится рекомендация, вывод результатов рекомендации аналогичен показанному на рисунке 9.4.

10 Информационная безопасность

10.1 Построение диаграммы потоков данных

В данной ВКР была спроектирована и разработана ИС рекомендательного сервиса музыкальных композиций. Для неё было выполнено моделирование опасностей и построены:

* начальная контекстная диаграмма;
* DFD-диаграмма первого уровня;
* перечень опасностей по методу STRIDE;
* деревья опасностей;
* оценка рисков по методу DREAD.

При моделировании опасностей, DFD диаграммы позволяют провести анализ внешних источников данных, адресатов данных, логических функций, потоков и хранилищ данных, к которым осуществляется доступ.

Используемые диаграммы являются основным средством моделирования опасностей в функциональных требованиях, разработанных для информационной системы. С их помощью эти требования представляются в виде иерархии функциональных компонентов, связанных потоками данных.

Начальная контекстная диаграмма для разработанной информационной системы продемонстрирована на рисунке 10.1. Она отображает минимальный уровень детализации.

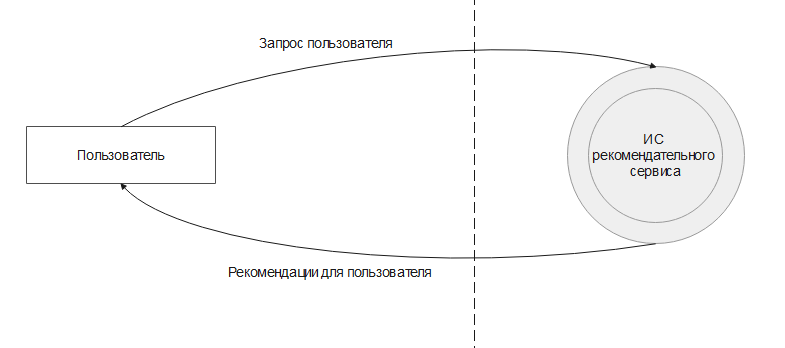


Рисунок 10.1 – Начальная контекстная диаграмма

Как правило, чаще всего подвержена атакам именно серверная часть программного обеспечения. И она является самым уязвимым местом.

DFD диаграмма первого уровня является диаграммой более низкого уровня, чем начальная контекстная диаграмма. Она отображает декомпозицию серверной части и продемонстрирована на рисунке 10.2.

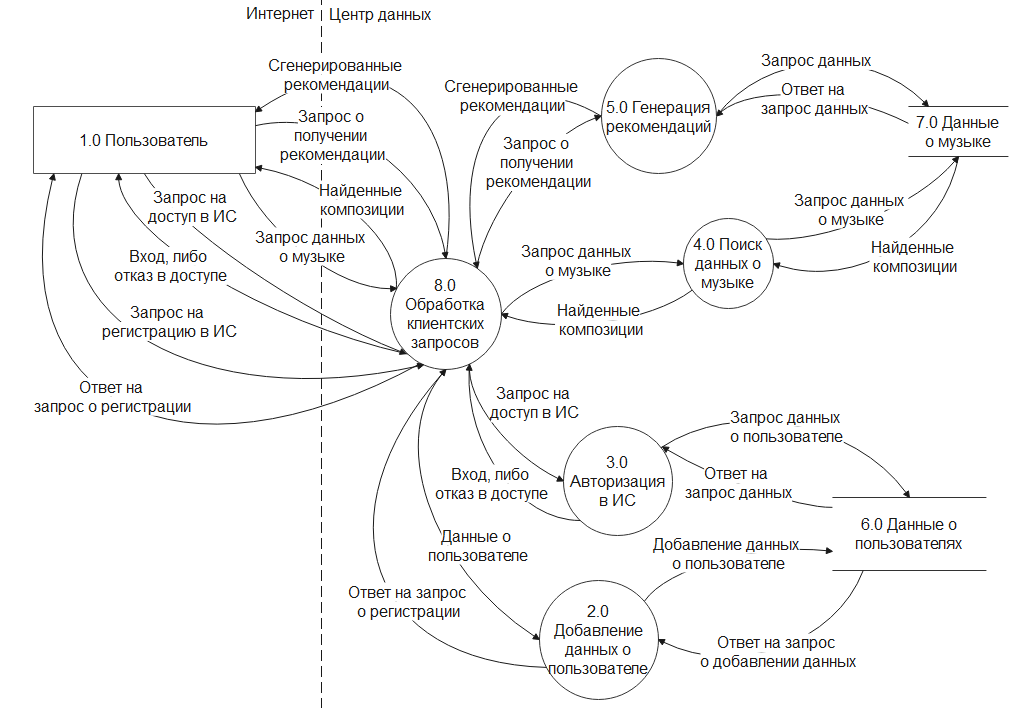
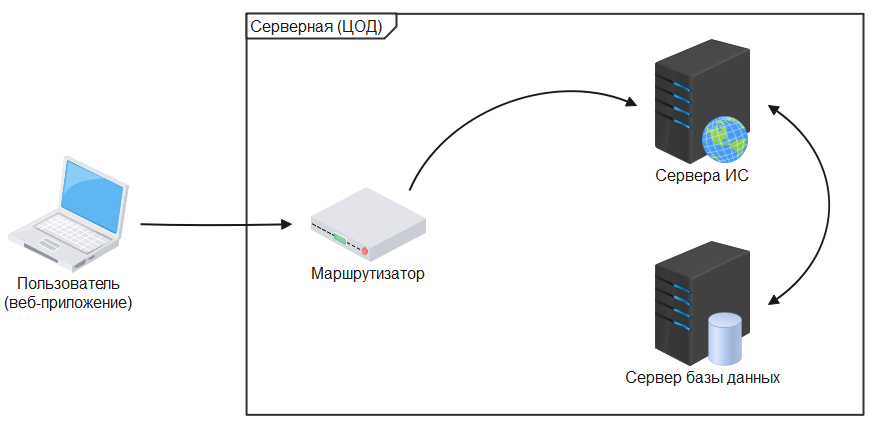


Рисунок 10.2 – DFD-диаграмма первого уровня

10.2 Схема физического расположения компонентов разработанной ИС

Разработанная рекомендательная информационная система музыкальных композиций имеет клиент-серверную архитектуру, это означает, что её компоненты расположены в контролируемой зоне центра обработки данных. Либо допускается размещение серверов в серверной, расположенной в офисе, в закрытом помещении, с системами кондиционирования и газовые установки автоматического пожаротушения (ГАУП). Вход в закрытую зону должен осуществляться с помощью электронных пропусков. Схема расположения компонентов показана на рисунке 10.3.

 Рисунок 10.3 – Схема физического расположения

10.3 Выявление перечня угроз

Также, при анализе уязвимостей используют метод STRIDE. С помощью него определяют уровень опасности на возможность уязвимости системы.

Для его использования составляется список опасностей и далее они классифицируются по типам. В методе STRIDE существует 6 пунктов, каждый из которых соответствует определенной букве в слове STRIDE:

* S – подмена сетевых объектов (spoofing identity);
* T – модификация данных (tampering with data);
* R – отказ от авторства (repudiation);
* I – разглашение информации (information disclosure);
* D – отказ в обслуживании (denial of service);
* E – повышение привилегий (elevation of privilege).

Перечень уязвимостей, составленный по методу STRIDE, описан в таблице 10.1.

Таблица 10.1 – Перечень опасностей STRIDE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Элемент *DFD* | Подмена (*S*) | Несанкционированный доступ (*T*) | Отказ от факта  получения или  отправки сообщения (*R*) | Раскрытие  информации (*I*) | Отказ  в обслуживании (*D*) | Повышение  привилегий (*E*) |
| Внешние сущности | | | | | | |
| 1 Пользователь | х |  | х |  |  |  |
| Хранилища данных | | | | | | |
| 1 База данных *MySQL* | х | х | х | х | х |  |
| Потоки данных | | | | | | |
| 1 Запрос на регистрацию в ИС |  | х | х | х | х |  |
| 2 Запрос на доступ в ИС |  | х | х | х | х |  |
| 3 Запрос данных о музыке |  | х | х | х | х |  |
| 4 Запрос о получении рекомендации |  | х | х | х | х |  |
| 5 Запрос данных |  | х | х | х | х |  |
| 6 Данные о пользователе |  | х | х | х | х |  |
| 7 Добавление данных о пользователе |  | х | х | х | х |  |
| 8 Запрос на доступ в ИС |  | х | х | х | х |  |
| 9 Вход, либо отказ в доступе |  | х | х | х | х |  |
| 10 Найденные композиции |  | х | х | х | х |  |
| 12 Сгенерированные рекомендации |  | х | х | х | х |  |
| 13 Ответ на запрос о регистрации |  | х | х | х | х |  |
| 14 Ответ на запрос о добавлении данных |  | х | х | х | х |  |
| Процессы | | | | | | |
| 1 Обработка клиентских запросов | х | х |  | х | х | х |
| 2 Авторизация в ИС | х | х |  | х | х | х |
| 3 Поиск данных о музыке | х | х |  | х | х | х |
| 4 Добавление данных пользователе | х | х |  | х | х | х |
| 5 Генерация рекомендаций | х | х |  | х | х | х |

Далее происходит анализ всех найденных уязвимостей системы и составляются деревья опасностей. На рисунках 10.4-10.7 изображены деревья опасностей, определенные по методу STRIDE.

Такое исследование информационной сети позволяет обнаружить максимальное количество опасностей.

10.4 Построение деревьев опасностей

Следующим шагом при анализе угроз необходимо выполнить построение деревьев развития опасностей для самых максимальных угроз из таблицы 10.1. На рисунке 10.4 представлено дерево опасности при несанкционированном доступе к базе данных.

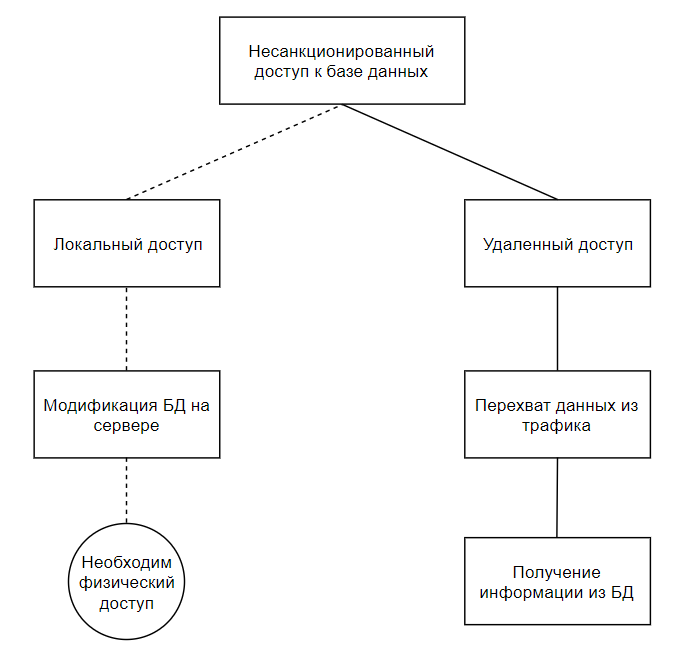


Рисунок 10.4 – Дерево опасности при несанкционированном доступе к базе данных

На рисунке 10.5 представлено дерево опасности при подмене сервера базы данных.

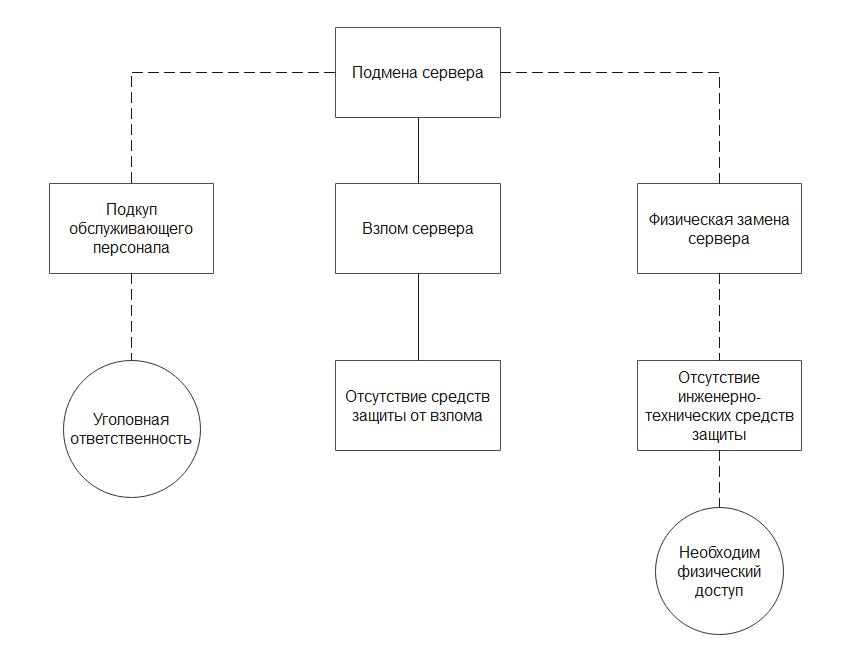


Рисунок 10.5 – Дерево опасностей при подмене сервера

На рисунке 10.6 представлено дерево опасности при несанкционированном доступе к Аутентификации в ИС.

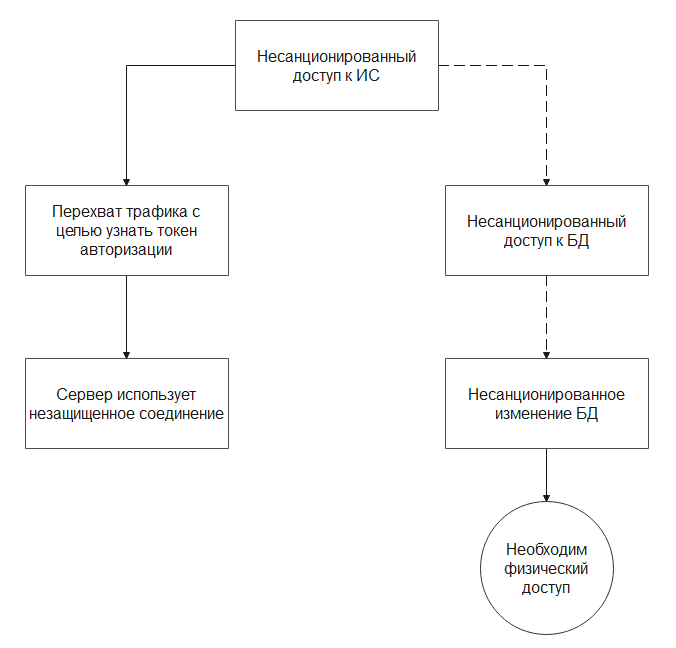


Рисунок 10.6 – Дерево опасностей при несанкционированном использовании системы с повышенными привилегиями

На рисунке 10.7 представлено дерево опасности при отказе от обслуживания.

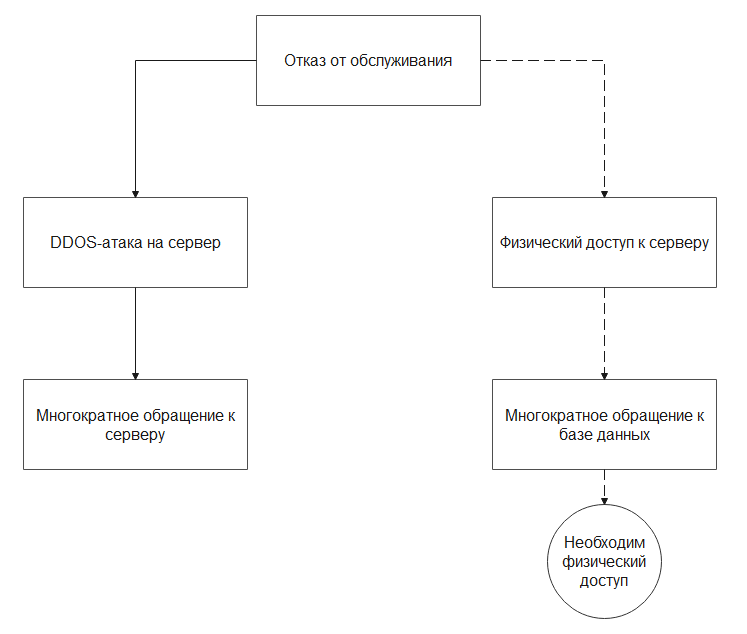


Рисунок 10.7 – Дерево опасностей при отказе от обслуживания

10.5 Анализ рисков опасности

Для анализа рисков опасности уязвимостей следует произвести их оценку по методу DREAD. Были выбраны ранее описанные опасности.

Используемый метод DREAD делится на пять пунктов, каждый из которых соответствует букве в названии методики. Метод DREAD расшифровывается следующим образом:

* D – потенциальный ущерб (damage potential);
* R – повторяемость (reproducibility);
* E – подверженность взлому (exploitability);
* A – охваченные пользователи (affected users);
* D – вероятность обнаружения (discoverability).

Расчеты по методу DREAD отображены в таблице 10.2.

Таблица 10.2 – Оценка рисков по методу DREAD

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Опасность | D | R | E | A | D | Итоговая оценка |
| 1 Подмена пользователя | 5 | 5 | 8 | 4 | 7 | 5,8 |
| 2 Подмена базы данных | 9 | 6 | 5 | 7 | 5 | 6,4 |
| 3 Несанкционированный доступ к базе данных | 9 | 10 | 5 | 6 | 10 | 8 |
| 4 Раскрытие информации из базы данных | 6 | 5 | 7 | 5 | 3 | 5,2 |
| 5 Отказ в обслуживании базы данных | 5 | 5 | 8 | 4 | 7 | 5,8 |
| 6 Несанкционированный доступ к данным о пользователях ИС | 6 | 7 | 8 | 4 | 5 | 6 |
| 7 Раскрытие информации данных о пользователям ИС | 6 | 7 | 8 | 4 | 5 | 6 |
| 8 Отказ в обслуживании ИС | 5 | 5 | 6 | 1 | 9 | 5,2 |
| 9 Несанкционированный доступ к данным о музыке | 6 | 7 | 8 | 4 | 5 | 6 |
| 10 Раскрытие информации запроса данных о музыке | 6 | 5 | 4 | 5 | 2 | 4,4 |
| 11 Отказ в обслуживании на запрос данных о музыке | 7 | 4 | 4 | 6 | 2 | 4,6 |
| 12 Несанкционированный доступ к авторизации | 10 | 9 | 7 | 10 | 10 | 9,2 |
| 12 Несанкционированный доступ к добавлению данных о пользователе | 7 | 3 | 4 | 5 | 8 | 5,4 |
| 13 Отказ в обслуживании при добавлении данных о пользователе в БД | 7 | 3 | 4 | 5 | 8 | 5,4 |
| 14 Подмена процесса авторизации в ИС |  |  |  |  |  | 4,4 |
| 15 Раскрытие информации авторизации в ИС | 6 | 7 | 8 | 4 | 5 | 6 |
| 16 Раскрытие информации обработчика клиентских запросов | 6 | 7 | 8 | 4 | 5 | 6 |
| 17 Отказ в обслуживании обработчика клиентских запросов | 5 | 5 | 6 | 1 | 9 | 5,2 |
| 18 Попытка взлома пароля | 7 | 10 | 4 | 8 | 10 | 7,8 |

Из таблицы 10.2 видно, что для разработанной в выпускной квалификационной работе рекомендательной системе наибольшую опасность представляет угроза, связанная с несанкционированным доступом к авторизации в ИС.

10.6 Методы по предотвращению опасностей

В ходе анализа информационной системы на возможные угрозы была выявлена угроза несанкционированного доступа к авторизации в ИС. В ИС необходимо защитить все возможные уязвимые места, найденные для этой угрозы по методу STRIDE. Они были продемонстрированы на дереве опасностей (рисунке 10.6).

Были найдены следующие уязвимости, с помощью которых можно осуществить описываемую угрозу:

* перехват трафика с целью узнать токен, логин или пароль в незашифрованном сетевом трафике;
* попытка подобрать комбинации логин/пароль методом перебора.

Уязвимость перехвата трафика, с целью узнать токен, является самой незащищенной частью этой угрозы. Для ее предотвращения необходимо:

* использовать защищённые соединения;
* шифровать передаваемые по сети данные;
* использовать принцип сессий;
* внедрить в программную и серверную часть ИС шифрованную передачу данных, либо использовать для взаимодействия между частями информационной системы VPN тоннели;
* внедрение авторизации по типу OAuth2.

Для защиты от попытки подобрать комбинации логин/пароль методом перебора необходимо:

* использовать проверку, при попытке множественного ввода некорректных аутентификационных данных.

В случае использования всех предложенных мер защиты от уязвимостей можно будет считать, что разработанная в рамках выпускной квалификационной работы, информационная система защищена от угрозы несанкционированного доступа к информационной системе.

Заключение

В ходе данной работы были исследованы виды рекомендательных систем и принципы их построения, было проведено исследование существующих алгоритмов рекомендации.

В результате анализа рекомендательных систем подбора музыкальных композиций был выбран контентно-ориентированный подход для построения рекомендательной системы. Данный подход не требует большой базы пользовательских оценок для построения рекомендаций и поэтому позволяет решить проблему холодного старта. Также был разработан алгоритм поиска и сортировки музыкальных композиций по степени сходства их музыкальных характеристик.

В результате выполнения данной работы была построена информационная система для рекомендации музыки на основе созданного алгоритма сортировки.

Разработанный алгоритм можно использовать для совершенствования существующих музыкальных сервисов, использующих рекомендательные системы с фильтрацией по контенту.

Список использованных источников

1. Ricci, Francesco. Recommender Systems Handbook. Second edition / Francesco Ricci, Lior Rokach, Bracha Shapira. – Switzerland AG : Springer, 2015. – 1003 p.
2. Как это работает? Рекомендации в Яндекс.Музыке [Электронный ресурс]. URL: https://yandex.ru/blog/company/92883 (дата обращения 15.04.2021).
3. Парошина И. Как музыкальный сервис Pandora узнаёт и предсказывает предпочтения слушателей [Электронный ресурс]. URL: http://www.computerra.ru/87479/pandora-bigdata/ (дата обращения 19.04.2021).
4. Pandora vs. Spotify – Satisfaction Study. [Электронный ресурс]. URL: http://www.bridgeratings.com/pandoravs-spotify (дата обращения 03.05.2021).
5. D. Goldberg, D. Nichols, B.M. Oki, D. Terry Using Collaborative Filtering to Weave an Information Tapestry Comm. ACM. – 2017. – 70p.
6. J.S. Breese, D. Heckerman, C. Kadie Empirical Analysis of Predictive Algorithms for Collaborative Filtering Proc. 14th Conf. Uncertainty in Artificial Intelligence. – 2016 – 10p.
7. C.C. Aggarwal, J.L. Wolf, K-L. Wu, P.S. Yu Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering Proc. Fifth ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining. – 2015 – 12p.
8. P. Resnick, N. Iakovou, M. Sushak, P. Bergstrom, J. Riedl GroupLens: An Open Architecture for Collaborative Filtering of Netnews Proc. 2014 Computer Supported Cooperative Work Conf. – 2014 – 18p.
9. Kaitila, Juuso. (2017). A content-based music recommender system. [Электронный ресурс]. URL: https://trepo.tuni.fi/bitstream/handle/10024 /101198/GRADU1495623946.pdf (дата обращения 25.04.2021).
10. Shuyang, Zhao. (2014). A personalized hybrid music recommender based on empirical estimation of user-timbre preference. [Электронный ресурс]. URL: https://pdfs.semanticscholar.org/42b7/4109074d6a8665c001dbee3410e86 98eff50.pdf (дата обращения 03.05.2021).
11. Есть ли что-то общее у разных песен-хитов? [Электронный ресурс]. URL: https://m.habr.com/en/post/460313/ (дата обращения 04.05.2021).
12. Ройзнер М. Как работают рекомендательные системы. [Электронный ресурс]. URL: https://habrahabr.ru/company/yandex/blog/241455/ (дата обращения 19.04.2021).
13. Королева Д., Филиппов М. Анализ алгоритмов обучения рекомендательных систем [Электронный ресурс]. URL: http://engjournal.ru/articles/816/816.pdf (дата обращения 01.05.2021).
14. U. Shardanand Social Information Filtering: Algorithms for Automating “Word of Mouth” / U. Shardanand, P. Maes // Proc. Conf. Human Factors in Computing Systems. – 2015 – 124p.

Приложение А

**Модуль MainController.java**

@Controller

public class MainController {

@Autowired

private ServiceClass sclass;

@RequestMapping(value="/signup.htm",method=RequestMethod.GET)

public ModelAndView renderSignUp(){

ModelAndView modelAndView = new ModelAndView("signup.jsp");

return modelAndView;

}

@RequestMapping(value="/savedata.htm", method=RequestMethod.POST)

public ModelAndView getData(HttpServletRequest request,HttpServletResponse response){

User user = new User();

user.setFname(request.getParameter("firstname"));

user.setLname(request.getParameter("lastname"));

user.setCountry(request.getParameter("country"));

user.setEmailId(request.getParameter("email"));

user.setUname(request.getParameter("uname"));

user.setPswd(request.getParameter("password"));

String message = sclass.submitDataToDB(user);

ModelAndView modelAndView = new ModelAndView("index.jsp","responseMessage",message);

return modelAndView;

}

@RequestMapping(value="/validate.htm",method=RequestMethod.POST)

public ModelAndView recognizeUser(HttpServletRequest request,HttpServletResponse response){

Object responseObject = sclass.validateUser(request.getParameter("un"), request.getParameter("pwd"));

ModelAndView modelAndView;

if(responseObject.getClass()==User.class){

User user = (User) responseObject;

modelAndView = new ModelAndView("landingPage.jsp","userinfo",user);

}else{

modelAndView = new ModelAndView("index.jsp","loginstatus",responseObject);

}

return modelAndView;

}

@RequestMapping(value="/validat.htm", method=RequestMethod.POST)

public ModelAndView search(HttpServletRequest request, HttpServletResponse response){

HttpSession session = request.getSession();

User user = (User) session.getAttribute("sessionUser");

return new ModelAndView("landingPage.jsp","userinfo",user);

}

@RequestMapping(value="/questionset.htm", method=RequestMethod.GET)

public ModelAndView renderQuestionPage(HttpServletRequest request, HttpServletResponse response){

HttpSession session = request.getSession();

User user = (User) session.getAttribute("sessionUser");

return new ModelAndView("questionForm.jsp","userinfo",user);

}

@RequestMapping(value="/getplayer.htm", method=RequestMethod.GET)

public ModelAndView renderMusicPlayer(HttpServletRequest request, HttpServletResponse response){

ArrayList<String> ls = new ArrayList<String>();

String baseURL = "http://0.0.0.0:5000/";

String endPoint = "getmp3url?";

String artist = request.getParameter("artist").replace(" ","%20");

String userInput = request.getParameter("songName").replace(" ","%20");

String songId = request.getParameter("songid");

try{

URL url = new URL(baseURL + endPoint + "input=" + userInput + "&songid=" + songId + "&artist=" + artist);

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("GET");

if (conn.getResponseCode() != 200) {

throw new RuntimeException("Failed : HTTP Error code : "

+ conn.getResponseCode());

}

BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));

String inputLine;

StringBuffer sb = new StringBuffer();

while ((inputLine = in.readLine()) != null) {

sb.append(inputLine);

}

in.close();

JSONObject myResponse = new JSONObject(sb.toString());

JSONObject tempjs = myResponse.getJSONObject("data");

ls.add(myResponse.getString("url"));

ls.add(tempjs.getString("song"));

ls.add(tempjs.getString("date"));

ls.add(tempjs.getString("artist"));

ls.add(tempjs.getString("albumart"));

ls.add(tempjs.getString("album"));

ls.add(tempjs.getString("artist\_image"));

ls.add(tempjs.getString("id"));

conn.disconnect();

}catch(Exception e){

System.out.println(e.getMessage());

}

return new ModelAndView("songPlayer.jsp","fetchedData",ls);

}

}

Приложение Б

**Модуль flaskAPI.py**

from flask import Flask,request

from flask\_cors import CORS

import json

import helperFunctionsAPI as hp

app = Flask(\_\_name\_\_)

CORS(app)

@app.route('/search', methods=['GET'])

def searchResult():

song\_input = request.args.get('song\_name')

artist\_input = request.args.get('artist\_name')

album\_input = request.args.get('album\_name')

response = hp.getDataFromQuery(song\_input, artist\_input, album\_input)

return json.dumps(response)

@app.route('/recommend', methods=['GET'])

def generateRecommend():

featureVector = json.loads(request.args.get("query"))["featureVector"]

print(featureVector)

recommendations = hp.getRecommendations(featureVector,"fromProfile",50,"")

return json.dumps(recommendations)

@app.route('/getmp3url', methods=['GET'])

def getMP3URL():

response = {}

artist = request.args.get('artist')

songInput = request.args.get('input')

songId = request.args.get('songid')

response["url"] = hp.getMP3URL(songInput, artist)

response["data"] = hp.getInformationFromID(songId)

return json.dumps(response)

@app.route('/recommendByTrack', methods=['GET'])

def recommendByTrack():

songId = request.args.get("input")

response = hp.getRecommendations([], "fromTrackID", 18, songId)

return json.dumps(response)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0')

Приложение В

**Модуль k\_nn.py**

from scipy import spatial

from itertools import islice

def take(n, iterable):

return dict(islice(iterable, n))

def getProfileFromTrackID(item\_profile, trackID):

query\_vector = item\_profile.loc[item\_profile["id"]==trackID]

for index,row in query\_vector.iterrows():

query\_vector = row.to\_list()

return query\_vector

def getKNN(item\_profile, way, K, trackID, query\_vector):

if(way=="fromTrackID"):

query\_vector = getProfileFromTrackID(item\_profile, trackID)

each\_item\_vector = item\_profile.iloc[:,1:11].values

print(each\_item\_vector)

each\_item\_id = item\_profile.iloc[:,0].values

print(query\_vector)

cosine\_distances = {}

i = 0

while (i != len(each\_item\_vector)):

cosine\_distance = spatial.distance.cosine(query\_vector[1:11],each\_item\_vector[i])

cosine\_distances[each\_item\_id[i]] = (1-cosine\_distance)

i = i + 1

cosine\_distances = {k: v for k, v in sorted(cosine\_distances.items(), key=lambda item: item[1], reverse=True)}

k\_items = take(K, cosine\_distances.items())