ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

РАЗРАБОТКА ИНФОРМАЦИОННОЙ БИБЛИОТЕЧНОЙ СИСТЕМЫ ХРАНЕНИЯ ЗВУКОЗАПИСЕЙ УСТНОЙ РЕЧИ

Выпускная квалификационная работа
обучающегося по направлению подготовки
09.03.02 «Информационные системы и технологии»
очной формы обучения, группы 07001408
Нгамби Самсон

Научный руководитель:

Ст. преподаватель Лихолоб П.Г.

БЕЛГОРОД 2018

РЕФЕРАТ

Разработка информационной библиотечной системы хранения звукозаписей устной – Нгамби Самсон, выпускная квалификационная работа бакалавра, Белгород, Белгородский государственный национальный исследовательский университет (НИУ «БелГУ»), количество страниц 45, включая приложения 59, количество рисунков 12, количество использованных источников 26.

КЛЮЧЕВЫЕ СЛОВА: информационная система, база данных, рекомендательные системы, звукозаписей.

ОБЪЕКТ ИССЛЕДОВАНИЯ: информационная библиотечная система хранения звукозаписей.

ПРЕДМЕТ ИССЛЕДОВАНИЯ: методы классификации и сортировки звукозаписей.

ЦЕЛЬ РАБОТЫ: персонализация рекомендации отдельного пользователя музыка, учитывая история прослушивании друзья пользователя.

ЗАДАЧИ ИССЛЕДОВАНИЯ: изучение существующих информационных библиотечных систем; изучение методов классификации и сортировки звукозаписей с использованием методов машинного обучения; анализ требований к разрабатываемой библиотечной системе; разработка алгоритмов группировки и классификации звукозаписей; организация структур данных и разработка базы данных звукозаписей; реализация программного обеспечения; тестирование программного продукта.

МЕТОДЫ ИССЛЕДОВАНИЯ: методы классификации и сортировки звукозаписей.

ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ: в результате работы была спроектирована и реализована полнофункциональное приложение для рекомендации и прослушивании музыки.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С ростом сети за последние десятилетия Интернет стал основным источник получения мультимедийной информации, такой как видео, книги и музыка и т. д.

Люди считают, что музыка является важным аспектом их жизни, и они слушать музыку, деятельность, которую они занимаются часто. Предыдущие исследования также показали, что участники чаще слушали музыку, чем любой другой деятельность (т. е. смотреть телевизор, читать книги и смотреть фильмы) [18].

Музыка, как мощный подход к коммуникации и самовыражения, следовательно, опротестовал множество исследований.

Однако проблема в настоящее время состоит в организации и управлении миллионов названий музыки, производимые общества. Техники Поиск Музыкальной Информации (Music information retrieval - MIR) были разработаны для решения проблем, таких, как классификация жанра, идентификации композитора и распознавания инструмента [5]. С 2005 года ежегодное мероприятие по оценке под названием Музыка Информационный Анализ (Music Information Retrieval Evaluation exchange - MIREX) проводится для облегчения разработка алгоритмов MIR [6].

Объем выпускаемой музыки также растет все больше и больше, при этом многие сервисы, такие как Google Play Music и Deezer, имеют более 30 миллионов песен.

Благодаря таким огромным библиотекам музыки пользователи могут потеряться и бороться за то, чтобы найти новую музыку, которая им нравится. По мере выпуска новых песен пользователи могут не получать информацию, поэтому для них становится проблематичным. Выпускная квалификационная работа посвящен тому, как избавиться от некоторых таких проблем. В этой

работе был реализован приложение, которое рекомендует музыкальные файлы пользователям в соответствии с их предпочтениями.

Рекомендательные системы являются удобной альтернативой традиционным поисковым алгоритмам и активно используются в электронной коммерции [19].

Таким образом, разработка информационной библиотечной системы хранения звукозаписей является актуальной темой.

Объект исследования – информационной библиотечной системы хранения звукозаписей.

Предмет исследования – методы классификации и сортировки звукозаписей.

Цель работы является персонализация рекомендации отдельного пользователя музыка, учитывая история прослушивании друзья пользователя.

Исходя из поставленной цели, выделяют следующие задачи выпускной квалификационной работы:

– изучение существующих информационных библиотечных систем;

– изучение методов классификации и сортировки звукозаписей с использованием методов машинного обучения;

– анализ требований к разрабатываемой библиотечной системе;

– разработка алгоритмов группировки и классификации звукозаписей;

– организация структур данных и разработка базы данных звукозаписей;

– реализация программного обеспечения;

– тестирование программного продукта

Выпускной квалификационной работы состоит из введения, трех глав, заключения, список литературы и приложение.

1 Организация информационных библиотечных систем хранения

1.1 Форматы представления звукозаписей в библиотечных системах

Звукозапись – это звуковые сигналы, полученные в результате звукозаписи и содержащиеся на аналоговом или цифровом носителе, или закодированные в определенном файле [20]. Общее определение звукозаписи следует отличать от определения в законодательстве об авторском праве и смежных правах, где звукозапись — это любая, но исключительно звуковая запись исполнений или иных звуков, то есть, например, звуковая запись на компакт-кассете и звуковая запись, которая является неотъемлемой частью кинофильма, будут охраняться по-разному. Наиболее распространена следующая математическая модель представления звукозаписи [23]:

$$\overset{\rho}{x} = \left(x_1, x_2, .., x_i, .., x_N\right)^T,\ x_i \in [-1, 1], \tag{1.1}$$

где x_i – количество бит в дополнительной информации.

Сегодня управление информацией – это важная часть музыкальных технологий, охватывающая управление публичными и личными коллекциями, строительство крупных редакторов и хранение результатов музыкального анализа [22].

одним из первых шагов к созданию музыкальной информационной системы является классификация музыкальных файлов.

1.2 Методы классификации и сортировки звукозаписей с использованием методов машинного обучения

Классификация музыкального жанра была вдохновляющей работа в области поиска музыкальной информации (MIR). Классификация жанра может быть

полезной для объяснения некоторых фактических интересные проблемы, такие как создание ссылок на песни, поиск связанных с песнями, поиск обществ, которым понравится эта конкретная песня.

Однако для того, чтобы классификация была полезной на практике, необходим надежный алгоритм, который может быть успешно обучен на небольших наборах данных.

Элементы практического машинного обучения приведены в работах [1].

1.2.1 Лес Оптимального Пути (Optimum-Path Forest -OPF)

Классификатор OPF моделирует проблему распознавания образов как задачу разделения графа, в которой предопределенный набор выборок из каждого класса (то есть прототипов) конкурирует за минимальную стоимость пути до остальных образцов. Это приводит к сбору деревьев оптимального пути, внедренных в прототипные узлы, создавая оптимальный путь леса, учитывая все образцы обучения. Образцы тестов классифицируются путем постепенной оценки оптимальных путей от прототипов, как если бы они были частью леса, и назначали метки наиболее сильно связанных корней. Понятие оптимального пути связано с минимизацией функции стоимости пути. Классификатор OPF может быть разработан до тех пор, пока мы используем плавную функцию стоимости пути.

Пусть $\mathcal{D} = \mathcal{D}1 \cup \mathcal{D}2$ - набор данных, разделенный на тренировку ($\mathcal{D}1$) и тестовое ($\mathcal{D}2$) множество. Кроме того, пусть $\mathcal{G} = (\mathcal{V}, \square, w)$ - граф, возникший из $\mathcal{D}1$, такой, что $\mathcal{V} = \mathcal{D}1$ и $\square$ обозначает отношение смежности, которое определяет граф полной связности, то есть график, где каждая пара узлов соединенные друг с другом. Дуги взвешиваются расстоянием между их соответствующими узлами. Каждый узел графа $s \in \mathcal{V}$ моделируется как n-мерный вектор-функция, а w (s, t) - вес (расстояние) между двумя узлами графа xi и xj, используемыми для

взвешивания дуги <s, t> ∈ □. Математически w - это функция, которая берет два узла графа и возвращает расстояние между ними, то есть w → $\mathcal{V}$ × $\mathcal{V}$: □ +.

1.2.2 Наивный байесовский классификатор

Наивный байесовский классификатор — это простой вероятностный классификатор, основанный на применении теоремы Байеса со строгими (наивными) предположениями о независимости.

В зависимости от точной природы вероятностной модели, наивные байесовские классификаторы могут обучаться очень эффективно. Во многих практических приложениях для оценки параметров для наивных байесовых моделей используют метод максимального правдоподобия; другими словами, можно работать с наивной байесовской моделью, не веря в байесовскую вероятность и не используя байесовские методы.

Несмотря на наивный вид и, несомненно, очень упрощенные условия, наивные байесовские классификаторы часто работают намного лучше во многих сложных жизненных ситуациях.

Достоинством наивного байесовского классификатора является малое количество данных для обучения, необходимых для оценки параметров, требуемых для классификации.

Байесовский классификатор оценивает вероятность того, что данная вокализация принадлежит определенному классу. Эта вероятность может быть получена из теоремы Байеса [24]:

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)\,P(\omega_i)}{p(\mathbf{x})},$$

(1.2)

где p (x | ωi) обозначает вероятность наблюдения вектора признаков x, заданного классом ωi, P (ωi) - это предыдущая вероятность класса ωi, а p (x) - вероятность x. Чтобы оценить p (x | ωi), мы предположили, что функция правдоподобия является гауссовой и поэтому может оценивать ее параметры из набора данных.

1.2.3 Многослойный перцептрон Румельхарта

Многослойный перцептрон — это частный случай перцептрона Розенблатта, в котором один алгоритм обратного распространения ошибки обучает все слои. Название по историческим причинам не отражает особенности данного вида перцептрона, то есть не связано с тем, что в нём имеется несколько слоёв (так как несколько слоёв было и у перцептрона Розенблатта). Особенностью является наличие более чем одного обучаемого слоя (как правило — два или три). Необходимость в большом количестве обучаемых слоёв отпадает, так как теоретически единственного скрытого слоя достаточно, чтобы перекодировать входное представление таким образом, чтобы получить линейную разделимость для выходного представления. Существует предположение, что, используя большее число слоёв, можно уменьшить число элементов в них, то есть суммарное число элементов в слоях будет меньше, чем если использовать один скрытый слой. Это предположение успешно используется в технологиях глубокого обучения и имеет обоснование.

Классификатор многослойный перцептрон представляет собой прямую нейронную сеть, состоящую из нескольких нейронных слоев, направленных на решение многоклассических проблем. Ввод каждого слоя представляет собой взвешенную сумму выходного сигнала предыдущего слоя. Число нейронов в первом слое равно числу признаков входа, а число нейронов в последнем слое равно числу классов. Нейронная сеть назначает вектор признаков, извлеченный

из вокализации x в класс ωq, если q-й выходной нейрон имеет самую высокую активацию.

1.2.4 Метод опорных векторов

Метод опорных векторов — это набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа. Принадлежит семейству линейных классификаторов и может также рассматриваться как специальный случай регуляризации по Тихонову. Особым свойством метода опорных векторов является непрерывное уменьшение эмпирической ошибки классификации и увеличение зазора, поэтому метод также известен как метод классификатора с максимальным зазором.

Основная идея метода — перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей. Алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.

Часто в алгоритмах машинного обучения возникает необходимость классифицировать данные. Каждый объект данных представляется как вектор (точка) в p-мерном пространстве (упорядоченный набор p чисел). Каждая из этих точек принадлежит только одному из двух классов. Вопрос состоит в том, можно ли разделить точки гиперплоскостью размерности $(p-1)$. Это — типичный случай линейной разделимости. Искомых гиперплоскостей может быть много, поэтому полагают, что максимизация зазора между классами способствует более уверенной классификации. То есть, можно ли найти такую гиперплоскость, чтобы расстояние от неё до ближайшей точки было максимальным. Это

эквивалентно тому, что сумма расстояний до гиперплоскости от двух ближайших к ней точек, лежащих по разные стороны от нее, максимально. Если такая гиперплоскость существует, она называется оптимальной разделяющей гиперплоскостью, а соответствующий ей линейный классификатор называется оптимально разделяющим классификатором.

1.2.5 Метод k-ближайших соседей

Метод k-ближайших соседей — метрический алгоритм для автоматической классификации объектов или регрессии.

В случае использования метода для классификации объект присваивается тому классу, который является наиболее распространённым среди k соседей данного элемента, классы которых уже известны.

В случае использования метода для регрессии, объекту присваивается среднее значение по k ближайшим к нему объектам, значения которых уже известны.

Алгоритм может быть применим к выборкам с большим количеством атрибутов (многомерным). Для этого перед применением нужно определить функцию дистанции. Классический вариант определения дистанции — дистанция в евклидовом пространстве.

Однако разные атрибуты могут иметь разный диапазон представленных значений в выборке (например, атрибут А представлен в диапазоне от 0.1 до 0.5, а атрибут Б представлен в диапазоне от 1000 до 5000), то значения дистанции могут сильно зависеть от атрибутов с большими диапазонами. Поэтому данные обычно подлежат нормализации. При кластерном анализе есть два основных способа нормализации данных [25]:

–Мини-макс нормализация.

$$x' = (x - MIN[X])/(MAX[X] - MIN[X]) \qquad (1.3)$$

–Z-нормализация.

$$x' = (x - M[X])/\sigma[X] \qquad (1.4)$$

где σ — стандартное отклонение. В этом случае большинство значений попадет в диапазон (-3;3).

Некоторые значимые атрибуты могут быть важнее остальных, поэтому для каждого атрибута может быть задан в соответствие определенный вес (например, вычисленный с помощью тестовой выборки и оптимизации ошибки отклонения). Таким образом, каждому атрибуту *k* будет задан в соответствие вес *zk*, так что значение атрибута будет попадать в диапазон [0; *zk* max(k)] (для нормализованных значений по методу минимакс). Например, если атрибуту присвоен вес 2.7, то его нормализовано-взвешенное значение будет лежать в диапазоне [0;2.7].

При таком способе во внимание принимается не только количество попавших в область определенных классов, но и их удаленность от нового значения.

Для каждого класса *j* определяется оценка близости [25]:

$$Q_j = \sum_{i=1}^{n} \frac{1}{d(x,a_i)^2} \qquad (1.5)$$

Где $d(x, a)$ – дистанция от нового значения x до объекта a.

У какого класса выше значение близости, тот класс и присваивается новому объекту.

С помощью этого метода можно вычислять значение одного из атрибутов классифицируемого объекта на основании дистанций от попавших в область объектов и соответствующих значений этого же атрибута у объектов.

$$xk = \frac{\sum_{i=1}^{n} k_i d(x,a_i)^2}{\sum_{i=1}^{n} d(x,a_i)^2} \qquad (1.6)$$

Где

a_i – i-ый объект, попавший в область

k_i – это значение атрибута к у заданного объекта a_i

$x_.$ – новый объект

x_k – k-ый атрибут нового объекта

1.2.6 Логистическая регрессия

Логистическая регрессия — это статистическая модель, используемая для предсказания вероятности возникновения некоторого события путём подгонки данных к логистической кривой.

Логистическая регрессия применяется для предсказания вероятности возникновения некоторого события по значениям множества признаков. Для этого вводится так называемая зависимая переменная *y*, принимающая лишь одно из двух значений — как правило, это числа 0 (событие не произошло) и 1 (событие произошло), и множество независимых переменных (также называемых признаками, предикторами или регрессорами) — вещественных $x_1,x_2,...,x_n$, на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной. Как и в случае линейной регрессии, для простоты записи вводится фиктивный признак $x_0 = 1$.

Делается предложение о том, что вероятность наступления события $y = 1$ равна [26]:

$$P\{y = 1|x\} = f(z) \qquad (1.7)$$

Где $z = \theta^T x = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n, x$ и θ – векторы-столбцы значений независимых переменных $1, x_1, \dots, x_n$ и параметров (коэффициентов регрессии)

— вещественных чисел $\theta_1, \dots, \theta$, соответственно, а f(z) — так называемая логистическая функция (иногда также называемая сигмоидом или логит-функцией):

$$f(z) = \frac{1}{1+e^{-z}} \qquad (1.8)$$

Так как y принимает лишь значения 0 и 1, то вероятность первого возможного значения равна:

$$P\{y = 0|x\} = 1 - f(z) = 1 - f(\theta^T x) \qquad (1.9)$$

Для краткости функцию распределения y при заданном x можно записать в таком виде:

$$P\{y|x\} = f(\theta^T x)^y (1 - f(\theta^T x)^{1-y}, y\epsilon\{0,1\} \qquad (1.10)$$

1.3 Анализ требований к разрабатываемой библиотечной системе

Анализ требований — это часть процесса разработки программного обеспечения, включающая в себя сбор требований к программному обеспечению (ПО), их систематизацию, выявление взаимосвязей, а также документирование [21].

1.3.1 Требования к функциональным характеристикам

Система должна обеспечивать следующие функции:

–Разделение пользователей, подключаемых через интерфейс на группы: неавторизованных и администраторов базы данных;

–Для неавторизованных пользователей - возможность поиска (фильтрации) по базе данных информации по разделам библиотечной системы;

−Регистрация пользователей;

−Ввод, вывод, хранение информации о пользователях (администрация): фамилия, имя, отчество, права доступа, логин, пароль;

−Ввод, вывод, хранение информации о неавторизованных пользователях: фамилия, имя, отчество, права доступа, логин, пароль;

−Ввод, вывод, хранение информации о звукозаписях (музыках);

−Возможность играть музыку.

1.3.2 Требования к надежности

Необходимо, чтобы система обладала устойчивостью к отказам оборудования и программных систем, а также электропитания. Для надежной работы электронного ресурса необходимы высоконадежные аппаратные и программные системы.

Необходимо, чтобы система обладала устойчивостью к отказам оборудования и программных систем, а также электропитания. Для надежной работы электронного ресурса необходимы высоконадежные аппаратные и программные системы.

Требования надежности должны быть регламентированы для следующих аварийных ситуаций:

−отсутствие электроэнергии; выход из строя программных средств системы;

−неверные действия пользователей;

−пожар, взрыв;

−попадание вирусов в систему и так далее.

Система должна:

−проводить контроль вводимой информации о музыках и пользователях;

−блокировать некорректные действия пользователя при работе с системой;

−обеспечивать целостность данных;

–разграничение прав доступа;

–хранить данные в отдельной базе.

1.3.3 Требования к лингвистическому обеспечению

К лингвистическому обеспечению предъявляются следующие требования:

–использование русского в интерфейсе информационной системы;

–эффективные интерфейсы должны быть очевидными и внушать своему пользователю чувство контроля. Необходимо, чтобы пользователь мог одним взглядом окинуть весь спектр своих возможностей, понять, как достичь своих целей и выполнить работу;

–интерфейс не должен беспокоить пользователя внутренним взаимодействием с системой. Необходимо бережное и непрерывное сохранение работы, с предоставлением пользователю возможности отменять любые действия в любое время.

1.3.4 Требования к составу и параметрам технических средств

Технические средства должны обладать уровнем надежности, отвечающим современным требованиям; необходимо предусмотреть возможность эффективного наращивания размера системы без осуществления значительных затрат. Сервер должен удовлетворять следующим требованиям:

–процессор Intel Core i5 или i7;

–8 Gb и более оперативной памяти;

–100 Gb –жесткий диск;

–монитор - 17”;

–сетевая карта PCIGeniusGF100TXRRL-8139 3Гб или выше;

–клавиатура;

– манипулятор типа «мышь».

Клиент должен удовлетворять следующим требованиям:

– процессор 1.3 GHz или выше;

– 128 Мб и более оперативной памяти;

– монитор – 17”;

– сетевая карта PCI GeniusGF100TXRRL-8139 10/100Мб;

– клавиатура;

– манипулятор типа «мышь».

1.3.5 Требования к информационной и программной совместимости

Система должна работать под управлением любой ОС. Система управления базы данных PostgreSQL. Другое ПО выбирается по решению разработчика. Основным критерием является низкая стоимость.

1.3.6 Требования к маркировке и упаковке

Готовое программное изделие и документация поставляются на компакт-дисках в стандартной упаковке. Один комплект программной документации должен быть распечатан с помощью лазерного принтера на листах формата А4 и иметь типографский переплет.

1.3.7 Требования к транспортированию и хранению

Требования к транспортированию и хранению программного изделия совпадают с аналогичными требованиями, предъявляемыми к компакт-дискам.

1.3.8 Требования к математическому обеспечению

Математические методы и алгоритмы, используемые для шифрования/дешифрования данных, а также программное обеспечение, реализующее их, сертифицированы уполномоченными организациями для использования в государственных органах Российской Федерации.

1.3.9 Требования к информационному обеспечению

В состав информационного обеспечения программы входит база данных, входная, внутренняя и выходная информация. Структура базы данных поддерживает кодирование хранимой и обрабатываемой информации в соответствии с общероссийскими классификаторами.

В качестве входной информации выступает:

- информация о музыкальных файлах;
- информация об абонентах;
- требования на получение музыкального файла из системы.

В качестве выходной информации служат:

- экранные формы с отображением на них результатов поиска музыкальных файлов;
- экранные формы с отображением списков музыкальных файлов;
- экранные формы с отображением списков рекомендуемых музыкальных файлов.

1.3.10 Требования к организационному обеспечению

Организационное обеспечение системы должно быть достаточным для эффективного выполнения персоналом возложенных на него обязанностей при осуществлении автоматизированных и связанных с ними неавтоматизированных

функций системы. К работе с системой допускаются сотрудники, имеющие навыки работы на персональном компьютере, ознакомленные с правилами эксплуатации и прошедшие обучение при работе с системой.

1.3.11 Требования к программной документации

Программная документация должна содержать следующие документы (см. ГОСТ 19.101-77):

– Программные документы:

а) спецификация (ГОСТ 19.202-78);

б) текст программы (ГОСТ 19.401-78);

в) описание программы (ГОСТ 19.402-78);

г) пояснительная записка (ГОСТ 19.404-79).

– Эксплуатационные документы:

а) ведомость эксплуатационных документов (ГОСТ 19.507-79);

б) формуляр (ГОСТ 19.501-78);

в) описание применения (ГОСТ 19.502-78);

г) руководство системного программиста (ГОСТ 19.503-79);

д) руководство программиста (ГОСТ 19.504-79);

е) руководство оператора (ГОСТ 19.505-79).

Требования к перечисленным документам не отличаются от требований, определенных в единой системе программной документации (ЕСПД).

2 Формализация моделей обработки звукозаписей в информационной библиотечной системе

2.1 Описание обобщенной модели обработки звукозаписей

Для проектирования информационной библиотечной системе были построены диаграммы методология функционального моделирования (IDEF0) с использованием AllFusion Process Modeler r7.

IDEF0 — это методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность (поток работ).

Стандарт IDEF0 представляет организацию как набор модулей, здесь существует правило — наиболее важная функция находится в верхнем левом углу, кроме того, есть правило стороны:

- стрелка входа приходит всегда в левую кромку активности,
- стрелка управления — в верхнюю кромку,
- стрелка механизма — нижняя кромка,
- стрелка выхода — правая кромка.

Описание выглядит как «чёрный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня. Также для того, чтобы быть правильно понятым, существуют словари описания активностей и стрелок. В этих словарях можно дать описания того, какой смысл вы вкладываете в данную активность либо стрелку.

Для того чтобы разработать информационную систему необходимо понимать процессы и функции, происходящие в ней. Таким образом, смоделировав систему

рекомендовать музыку, будут выявлены главные функциональные блоки, которые облегчат программную реализацию.

В данной работе на основе нотации IDEF0 была разработана контекстная диаграмма, изображенная на рисунке 2.1, которая показывает входные и выходные ресурсы.

Контекстная диаграмма «Рекомендовать музыку» – это диаграмма, расположенная на вершине древовидной структуры диаграмм, представляющая собой самое общее описание системы и ее взаимодействие с внешней средой (как правило, здесь описывается основное назначение моделируемого объекта). Контекстная диаграмма состоит из одного блока, описывающего функцию верхнего уровня, ее входы, выходы, управления, и механизмы, вместе с формулировками цели модели и точки зрения, с которой строится модель.

На контекстной диаграмме представлены:

a) На вход – материал или информация, которые используются или преобразуются работой для получения результата (выхода).

– Профили пользователи;

– Профили музыка.

b) Управление – правила, стратегии, процедуры или стандарты, которыми руководствуется работа.

– Права на музыку;

– Политика конфиденциальности;

c) На выход – материалы или информация, которые производятся работой.

– Рекомендуемая музыка.

d) Механизм – ресурсы, которые выполняют работу.

- Должностные лица.

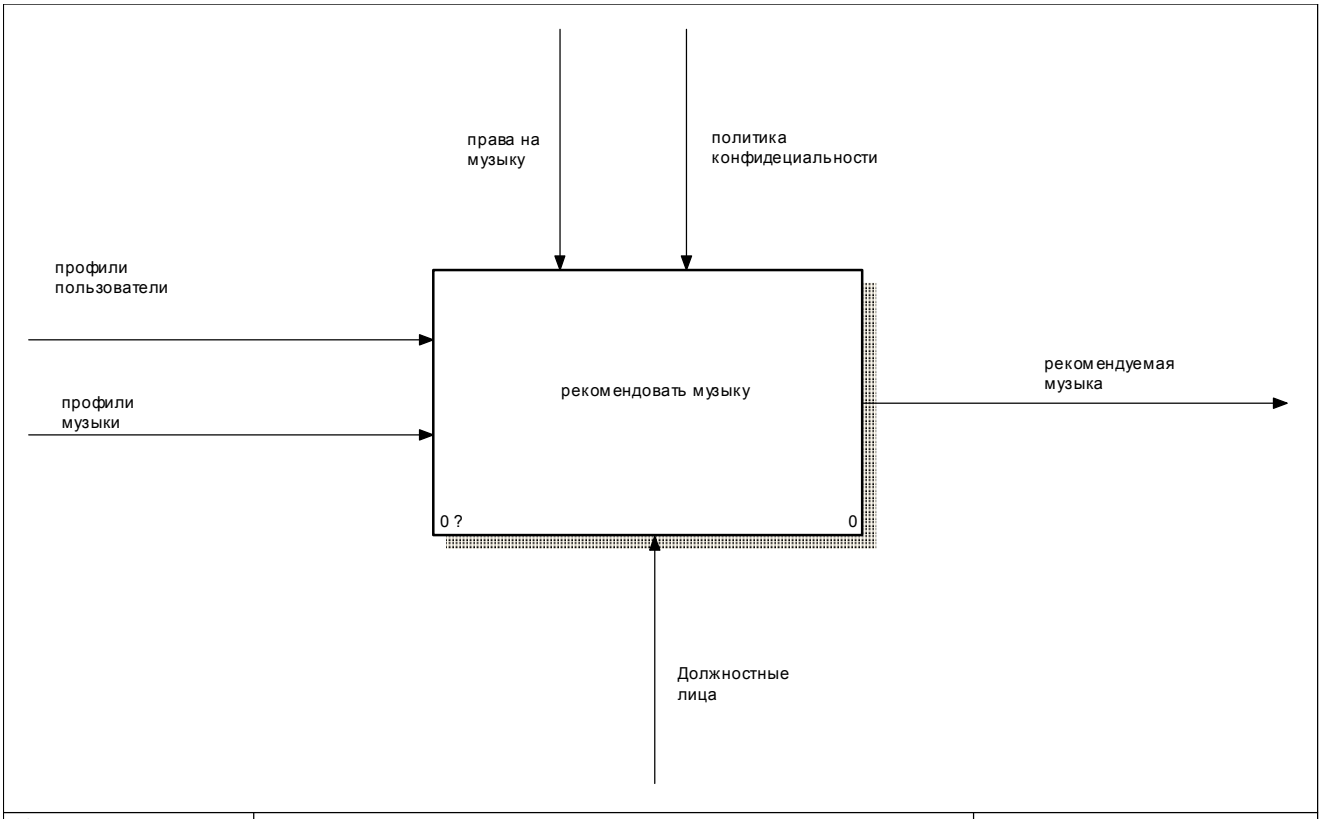

Рисунок 2.1 – Контекстная диаграмма рекомендации музыки

Диаграмма декомпозиции предназначена для детализации функций и получается при разбиении контекстной диаграммы на крупные подсистемы (функциональная декомпозиция) и описывает каждую подсистему и их взаимодействие.

Далее функциональный блок "Рекомендовать музыку" указанной диаграммы был декомпозирован до первого уровня, состоящего из шести функциональных блоков и представленного на рисунке 2.2.

На этой диаграмме было выделено 6 подсистемы: собрать данные, идентифицировать характеристики, выбрать алгоритмы для рекомендации музыкальных файлов, настроить параметры алгоритмов, обучать по собранным данным, обновить изменение.

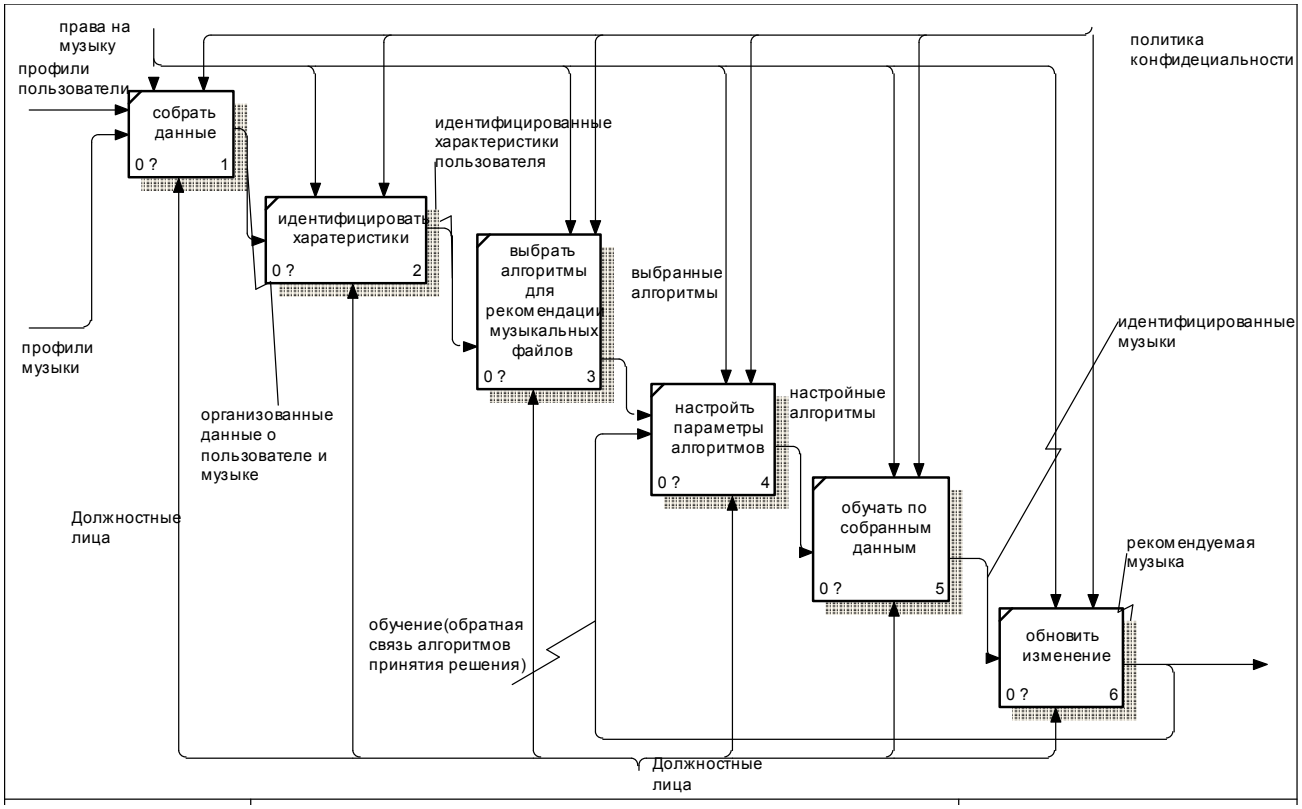

Рисунок 2.2 – Диаграмма декомпозиции функционального блока 0.

2.2 Организация структур данных и разработка базы данных звукозаписей

Для Организация структур данных и разработка базы данных звукозаписей информационной библиотечной системе были построены инфологическая и даталогическая модель с использованием AllFusion ERwin Data Modeler r7.

2.2.1 Инфологическое проектирование базы данных

Инфологическая модель – формализованное описание предметной области, которое будет «легко» читаться не только специалистами по базам данных, но пользователями этой базой данных.

Инфологическое проектирование отражает в себе смысл базы данных и разрабатывается с помощью информационных объектов. Основными элементами инфологических моделей являются сущности и связи между ними.

Сущность моделирует класс однотипных объектов, который имеет уникальное имя, информацию об этих объектах необходимо хранить в базе данных [2].

Для информационной системы выделены следующие сущности:

– Исполнитель

– Трек

– Тректег

– Пользователь

– Пользовательтегтрек

Исходя из приведенных выше сущностей, построена модель схемы отношений с помощью среды ERwin Data Modeler, которая представлена на рисунке 2.3.

AllFusion Erwin Data Modeler (ранее ERwin) – CASE-средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных [3].

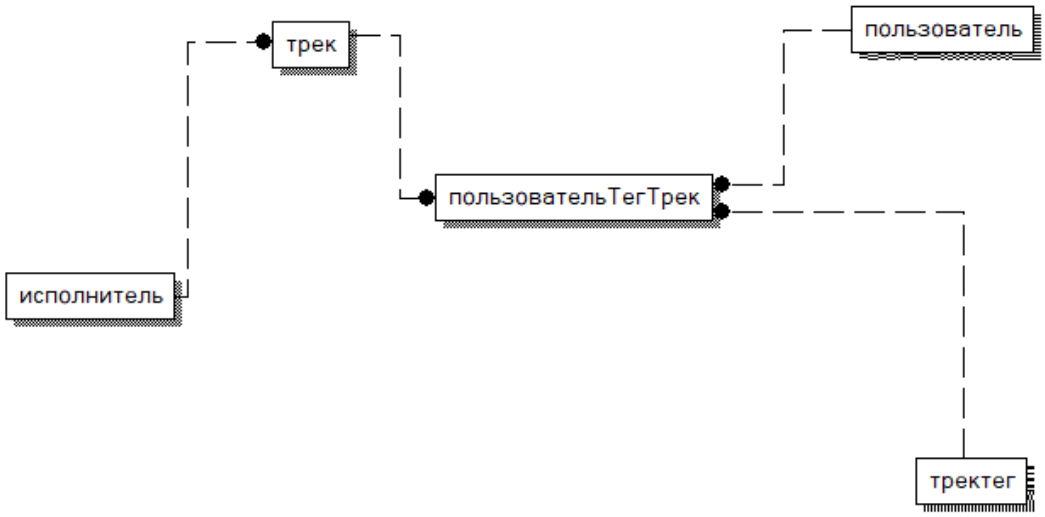

Рисунок 2.3 – Модель схемы отношений.

Связь работает путем сопоставления данных в ключевых столбцах; обычно это столбцы с одним и тем же именем в обеих таблицах. Существует три типа

связей между таблицами. Тип создаваемой связи зависит от того, как определены связанные столбцы.

Связи «один ко многим» (1 – n). Связь «один ко многим» самая распространенная. В этом типе связей у строки таблицы А может быть несколько совпадающих строк таблицы Б, но каждой строке таблицы Б может соответствовать только одна строка из А [4].

Например:

– между таблицами «исполнитель» и «трек» установлена связь «один ко многим»: так как каждый исполнитель имеет несколько треков, но каждый трек принадлежит только одному исполнителю.

– между таблицами «пользователь» и «пользователь трек тег» установлена связь «один ко многим»: так как каждый пользователь имеет более одного пользователей трек тегов, но каждый пользователь трек тегов принадлежит только одному пользователю.

– между таблицами «тректег» и «пользователь трек тег» установлена связь «один ко многим»: так как каждый трек тега имеет более одного пользователей трек тегов, но каждый пользователь трек тегов принадлежит только одному трек тегу.

– между таблицами «трек» и «пользователь трек тег» установлена связь «один ко многим»: так как каждый трек имеет более одного пользователей трек тегов, но каждый пользователь трек тегов принадлежит только одному треку.

2.2.2 Даталогическая Модель

Даталогическое проектирование заключается в разработке схемы базы данных, то есть совокупности схем отношений, которые адекватно моделируют сущности и семантические связи между ними.

Основой правильности схемы являются функциональные зависимости между атрибутами БД. Некоторые зависимости могут быть нежелательными, от них надо избавляться. Такая схема будет называться корректной.

Процесс разработки корректной схемы базы данных является даталогическим проектированием [4].

В даталогической модели каждой сущности в соответствие ставится отношение, должны быть расставлены первичные и вторичные ключи, все отношения должны быть приведены к нормальной форме.

Все отношения находятся в третьей нормальной форме. Логическая модель представлена на рисунке 2.4.

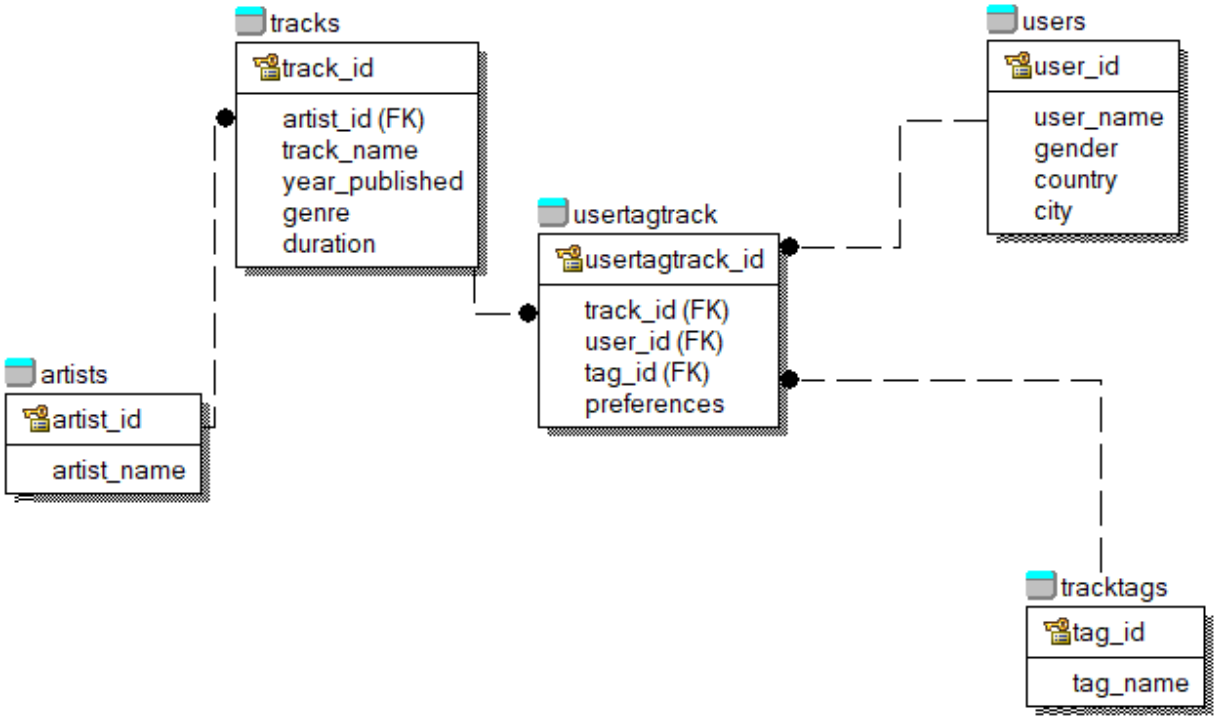

Рисунок 2.4 – Логическая модель

Физические модели баз данных определяют способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. Физическая модель представлена на рисунке 2.5.

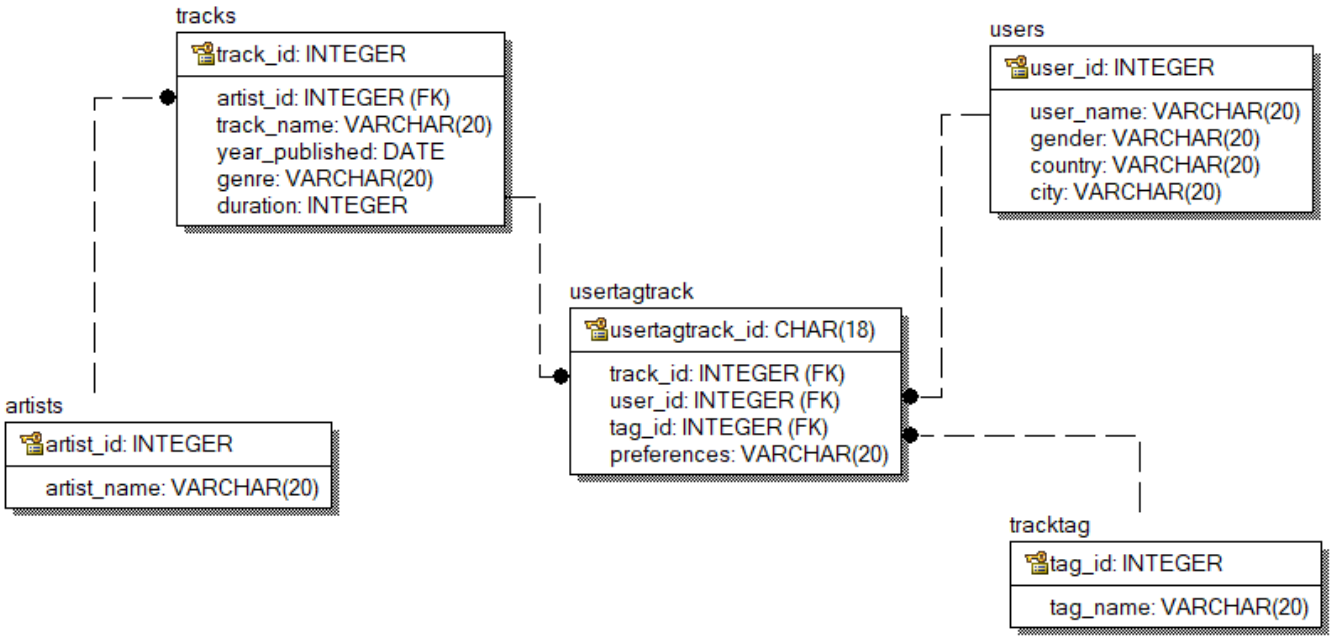

Рисунок 2.5 – Физическая модель.

Для информационной системы выделены следующие сущности, их атрибуты и тип данных:

–Исполнитель «artist» – сущность содержит информацию о исполнителях, которые исполняют разных треков. Таблица содержится следующие атрибуты:

a) artist_id–поле идентификации, которое состоит из значений (1, 2, 3…). Поле является первичным ключом в этой таблице и, соответственно, имеет тип данных INTERGER;

b) artist_name–Обозначает полное имя исполнитель например (Нгамби Самсон). Соответственно, поле имеет тип данных VARCHAR с 20 символами;

–Пользователь «user» – сущность содержит информацию о пользователях, которые слушают разных треков. Таблица содержится следующие атрибуты:

a) user_id–поле идентификации, которое состоит из значений (1, 2, 3…). Поле является первичным ключом в этой таблице и, соответственно, имеет тип данных INTERGER;

b) user_name–Обозначает полное имя пользователя например (Нгамби Самсон). Соответственно, поле имеет тип данных VARCHAR с 20 символами;

c) gender – обозначает пол пользователя например (мужчина). Соответственно, поле имеет тип данных VARCHAR с 20 символами;

d) country – обозначает страна пользователя например (Россия). Соответственно, поле имеет тип данных VARCHAR с 20 символами;

e) city – обозначает город пользователя например (Белгород). Соответственно, поле имеет тип данных VARCHAR с 20 символами;

– Трек тег «tracktag» – сущность содержит информацию о тегах, связанных с треками. Таблица содержится следующие атрибуты:

a) tracktag_id–поле идентификации, которое состоит из значений (1, 2, 3…). Поле является первичным ключом в этой таблице и, соответственно, имеет тип данных INTERGER;

b) tag_name–обозначает имя тега например (рок). Соответственно, поле имеет тип данных VARCHAR с 20 символами;

– Трек «tracks» – сущность содержит информацию о треках. Таблица содержится следующие атрибуты:

a) track_id–поле идентификации, которое состоит из значений (1, 2, 3…). Поле является первичным ключом в этой таблице и, соответственно, имеет тип данных INTERGER;

b) track_name–обозначает имя трека например (я не хочу). Соответственно, поле имеет тип данных VARCHAR с 20 символами;

c) date_published–обозначает дата исполнения трека например (02.06.1996). Соответственно, поле имеет тип данных DATE;

d) genre – обозначает жанра трека например (классика). Соответственно, поле имеет тип данных VARCHAR с 20 символами;

e) duration – обозначает длина продолжительность трека например (4 минут). Соответственно, поле имеет тип данных INTEGER;

– ПользовательТегТрек «usertagtrack» – сущность содержит информацию о отношениях пользователя к трекам. Таблица содержится следующие атрибуты:

a) usertagtrack _id–поле идентификации, которое состоит из значений (1, 2, 3…). Поле является первичным ключом в этой таблице и, соответственно, имеет тип данных INTERGER;

b) track_id– поле связи с таблицей «tracks», которое состоит из значений (1, 2, 3…). Поле является вторичным ключом в этой таблице и, соответственно, имеет тип данных INTERGER;

c) user_id– поле связи с таблицей «users», которое состоит из значений (1, 2, 3…). Поле является вторичным ключом в этой таблице и, соответственно, имеет тип данных INTERGER;

d) tag_id– поле связи с таблицей «tracktag» , которое состоит из значений (1, 2, 3…). Поле является вторичным ключом в этой таблице и, соответственно, имеет тип данных INTERGER;

e) preferences – обозначает предпочтения пользователя. Соответственно, поле имеет тип данных VARCHAR с 20 символами;

2.3 Разработка алгоритмов группировки и классификации звукозаписей

В данной работе был реализованы четыре разных алгоритма построения эффективная система рекомендаций.

2.3.1 Модели на основе популярности

Это самый простой алгоритм. Мы находим популярность каждой песни, заглядывая в тренировочный набор и подсчет числа пользователей, которые слушали эта песня. Затем песни сортируются в порядке убывания их популярность. Для каждого пользователя мы рекомендуем популярные песни, кроме тех, что уже в его профиле. Эта метод не включает персонализацию, и некоторые песни могут никогда не слушаться в будущем.

2.3.2 Коллаборативная фильтрация

Коллаборативная фильтрация включает сбор информации от многих пользователей, а затем прогнозирование, основанное на некоторых мерах сходства между пользователями и между элементами. Это можно разделить на модели, основанные на пользователях и элементах.

В модели, основанной на элементах, предполагается, что песни, которые часто выслушивают некоторые пользователи, имеют тенденцию быть похожими и, скорее всего, будут слушаться вместе в будущем и другим пользователем.

На рисунке 2.6 представлена работа фильтрации на основе элементов.

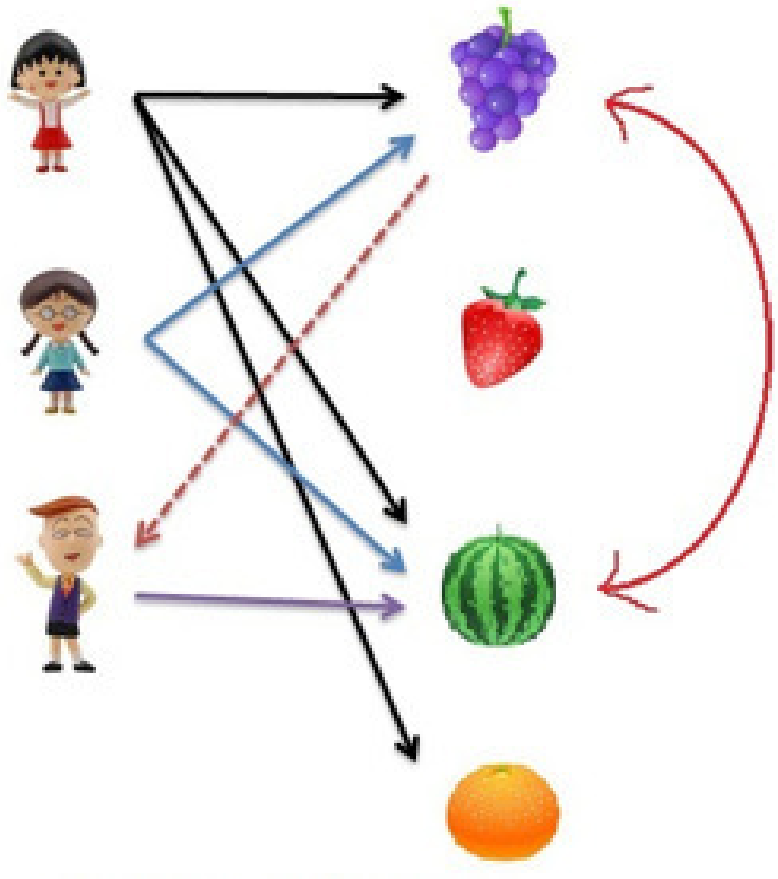

Рисунок 2.6 – Фильтрации на основе элементов.

Согласно модели сходства с пользователями, пользователи, которые имеют похожие истории прослушивания, т. е. прослушали одни и те же песни в

прошлом, имеют сходные интересы и, вероятно, будут слушать те же песни в будущем.

Нам нужна мера сходства для сравнения между двумя песнями или между двумя пользователями. Косинус сходства взвешивает каждого из пользователей одинаково, что обычно не так.

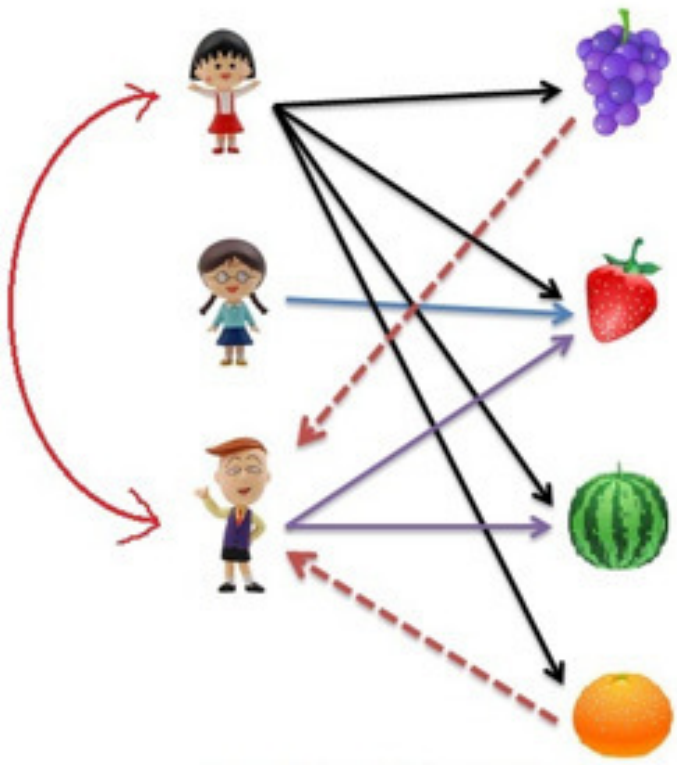

Рисунок 2.7 – Фильтрации на основе пользователей.

Пользователь должен быть взвешен меньше, если он проявил интерес к множеству предметов (он показывает, что либо она не различает песни по их качеству, либо просто любит изучать). Аналогично, пользователь взвешивается больше, если слушает очень ограниченный набор песен. Мера подобия $w_{ij} = P\ (i\ j)$ также имеет недостатки, что некоторые песни, которые больше слушаются пользователями, имеют более высокие значения сходства не потому, что они похожи и слушаются вместе, а потому, что они более популярны.

Мы использовали условно-вероятностную модель сходства между пользователями и между пунктами:

$$W_{u,v} = P\,(u/v)^{\propto} P\,(u/v)^{1-\propto}, \propto \epsilon(0,1) \qquad (2.1)$$

$$W_{uv} = \frac{|I(u) \cap I(v)|}{|I(u)|^{\propto} |I(v)|^{1-\propto}} \qquad (2.2)$$

На рисунке 2.7 представлена работа фильтрации на основе пользователей.

2.3.3 Контентная фильтрация

Фильтрация на основе контента, также называемая когнитивной фильтрацией, рекомендует элементы, основанные на сравнении между содержимым элементов и профилем пользователя. Содержимое каждого элемента представлено в виде набора дескрипторов или терминов, как правило, слов, которые встречаются в документе. Профиль пользователя представлен с теми же терминами и создается путем анализа содержимого элементов, которые были просмотрены пользователем.

При внедрении контентной системы фильтрации необходимо учитывать несколько проблем. Во-первых, термины могут быть назначены автоматически или вручную. Когда термины назначаются автоматически, должен быть выбран метод, который может извлекать эти термины из элементов. Во-вторых, термины должны быть представлены таким образом, чтобы как профиль пользователя, так и элементы могли быть сопоставлены значимым образом. В-третьих, должен быть выбран алгоритм обучения, который может изучить профиль пользователя на основе замеченных элементов и может давать рекомендации на основе этого профиля пользователя.

Различные значения α были протестированы, чтобы, наконец, прийти с хорошей мерой сходства.

Затем для каждого нового пользователя u и песни i пользовательская функция подсчета вычисляется как:

$$h_{u_i}^{U} = \sum_{ve u_i} f(w_{uv}) \quad (2.3)$$

Аналогичная оценка на основе элементов:

$$h_{u_i}^{U} = \sum_{je I_i} f(w_{ij}) \quad (2.4)$$

Местность функции подсчета также необходима, чтобы подчеркнуть элементы, которые более похожи. Для определения локальности мы использовали следующую экспоненциальную функцию.

$$f(W) = w^{q}, q \epsilon N \quad (2.5)$$

Это определяет, как отдельные компоненты подсчета очков влияют на общий выигрыш между двумя элементами. Подобные вещи подчеркиваются больше, в то время как менее похожий вклад падает до нуля.

После вычисления списков на основе пользователей и элементов мы использовали стохастическую агрегацию для их объединения. Это делается путем случайного выбора одного из них в соответствии с их распределением вероятностей, а затем рекомендуя верхние набранные предметы из них. Когда история песен пользователя слишком мала, чтобы использовать алгоритм рекомендаций пользователя, мы можем предложить рекомендации, основанные на сходстве песен, что дает лучшие результаты, когда количество песен меньше, чем у пользователей.

Этот метод не включает персонализацию. Более того, в большинстве песен слишком мало слушателей, поэтому их можно рекомендовать наименее

вероятно. Здесь мы не использовали данные счетчика пьес в конечном результате, так как они не дали хорошего результата, потому что модель сходства предвзята к нескольким песням, воспроизведенным несколько раз, и шум вычисления был порожден несколькими очень популярными песнями.

2.3.4 Модель KNN

В этом методе мы используем доступные метаданные. Мы создаем пространство песен по их характеристикам из метаданных и узнаем окрестности каждой песни. Мы выбираем некоторые из доступных функций (например, громкость, жанр, режим и т. д.), Которые мы считаем наиболее важными для того, чтобы отличить песню от других. Создав пространство функций, чтобы рекомендовать песни пользователям, мы смотрим на каждый профиль пользователей и предлагаем песни, которые являются соседями с песнями, присутствующими в его истории прослушивания. Мы приняли 50 лучших соседей каждой песни. Эта модель полностью персонализирована и использует метаданные. Но так как у нас был 280 ГБ файл метаданных, который занимает огромное количество времени на обработку, мы извлекли функции только 3 ГБ (10 000 песен), что составляет менее 2% от общего числа. Из-за этого у нас были черты лишь небольшого количества песен, что дает нам очень небольшую точность.

3 Концепция программного продукта и оценки работоспособности

3.1 Построение архитектуры программной реализации

Рекомендательная система представляет сравнительно новый класс программного обеспечения, в задачу которого входит предсказание того, какие объекты (фильмы, музыка, книги, новости, веб-сайты) будут интересны пользователю, с помощью анализа его действий и оценок [8]. На рисунке 3.1 представлена общая структура рекомендательной системы.

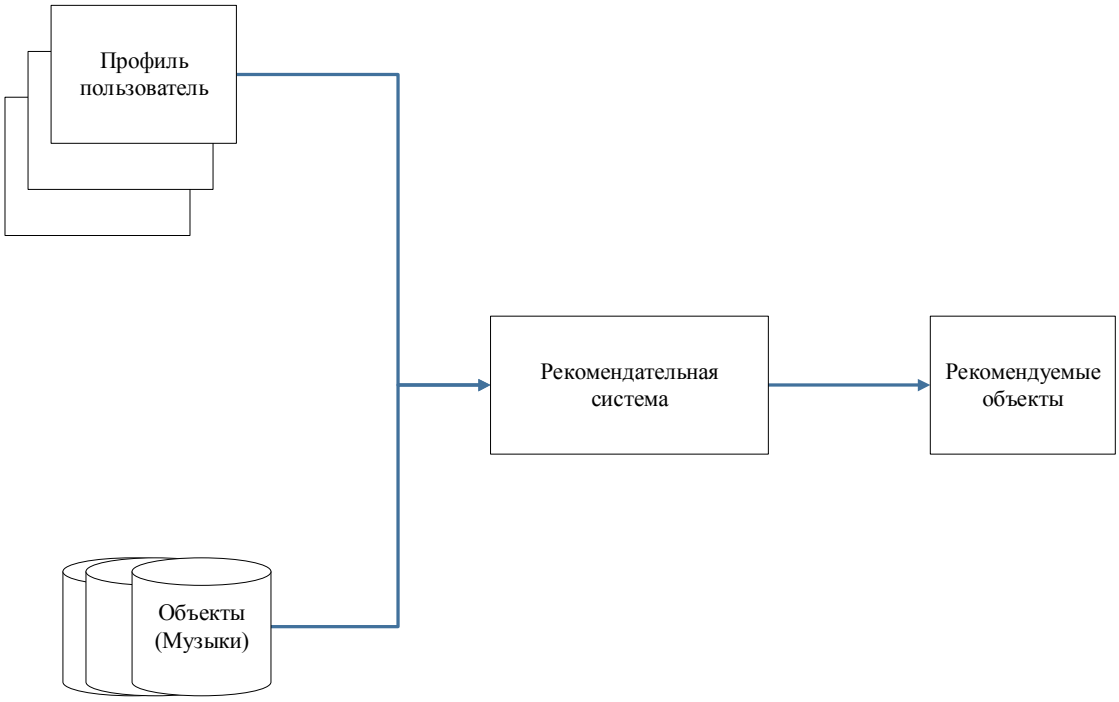

Рисунок 3.1 – Общая структура рекомендательной системы.

Из общей структуры системы рекомендаций в этой работе был построен архитектуру нашего музыкального приложения, используя векторный графический редактор – Microsoft Visio.

Microsoft Visio — векторный графический редактор, редактор диаграмм и блок-схем для Windows [10].

На рисунке 3.2 представлена архитектура разрабатываемое приложение.

Рисунок 3.2 – Архитектура рекомендательной системы.

Пользовательский интерфейс приложения позволяет пользователю входить в систему, искать музыку на основе альбома, исполнителя или трека, а также запрашивать рекомендации от системы.

Он также позволяет обрабатывать воспроизведение музыкальных файлов и смешивать музыкальные тэги с пользователем.

Рекомендатель является основой частью системы. Он проходит через несколько этапов для подготовки рекомендаций для пользователя. Некоторые из этапов включают фильтрацию контента [11] и совместную фильтрацию [12] с использованием средств K-NN [14] и чередование матричной факторизации наименьших квадратов [13].

Эти алгоритмы доступны на языке Python из библиотеки Pyspark [15]. В этой работе мы использовали эти уже разработанные алгоритмы для реализации нашего приложения.

Хранилище данных хранит песни с их метаданными, а также данными профиля пользователя. Только песни в этой базе могут быть рекомендуется, играли и оценивали. Механизм поиска музыки периодически собирает новые песни и метаданные из таких услуги, как Last.fm [16]. Каждый рекомендатель может получить доступ ко всем данным, хранящимся в репозитории.

Профиль пользователя содержит личные данные и личные предпочтения. Хранилище данных также собирает пользовательский контекст, например, песни, которые пользователь прослушивал с соответствующими настройками для соответствующего рекомендателя.

Для этой работы мы получили наши музыкальные метаданные и профили пользователей от LAST.FM.

Набор данных LastFM содержит события прослушивания для 992 пользователей и более 100 тысяч исполнителей [17]. Поскольку набор данных чрезвычайно редкие, мы очистили набор художников, оставив только те, для которых мы могли бы получить по меньшей мере три тега LastFM и отбрасывая исполнителей, которых слушали меньше, чем 20 пользователей.

3.2 Разработка программного продукта

Для работы был выбран язык Python, так как на нем реализованы библиотеки NumPy и Sklearn, где все выше описаны алгоритмы реализованы.

При помощи интегрированной среды разработки Visual Studio Code был реализован приложение для рекомендации музыки.

При запуске программа, появляется форма входа в систему для пользователей. На рисунке 3.3 представлена окно входа в систему.

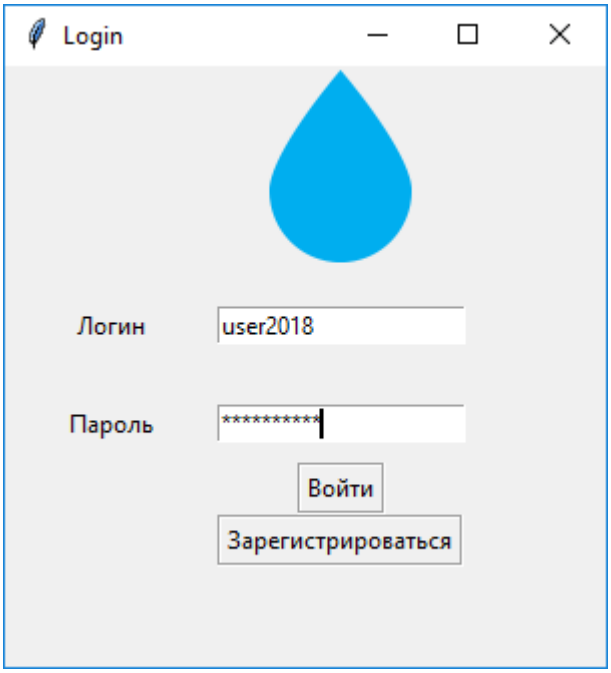

Рисунок 3.3 – Окно для входа в систему.

Требуется вводить «логин» и «пароль», зарегистрированным пользователем. При нажатии кнопки «войти» система проверяет введенных информации в отношении информации в базе данных. Если пользователь зарегистрирован, появляется главная форма.

При нажатии ссылка «зарегистрироваться» откроется форма «Регистрация» для регистрации в систему. На рисунке 3.4 представлена окно для регистрации в систему.

На форме «зарегистрироваться» требуется вводить «фамилию», «имя», «отчество», «пол», «страна», «город», и «электронный адрес».

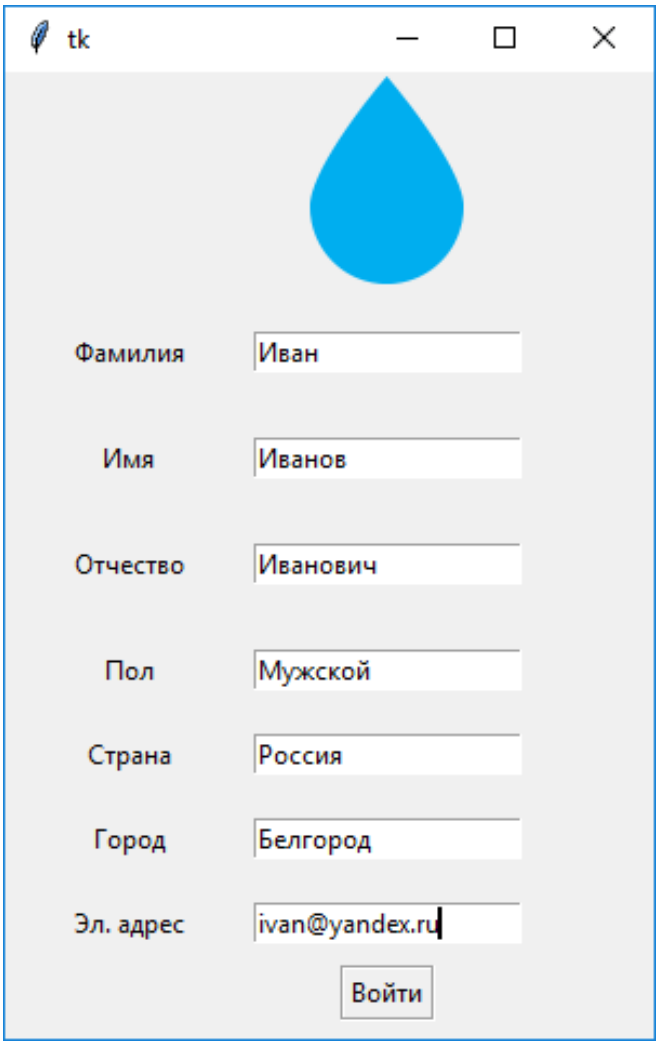

Рисунок 3.4 – Окно регистрации в систему.

При нажатии кнопки «войти» система проверяет введенных информации в отношении информации в базе данных и вводится введенные данные в базе данных и откроется главное окно приложения.

На главном окне можно сделать поиск музыкальных файлов по имени альбома, по имени исполнителя или по имени трека. При нажатии кнопка «получить рекомендации» система вызывает рекомендательная часть приложения и выдаёт список рекомендуемых песни для конкретного пользователя.

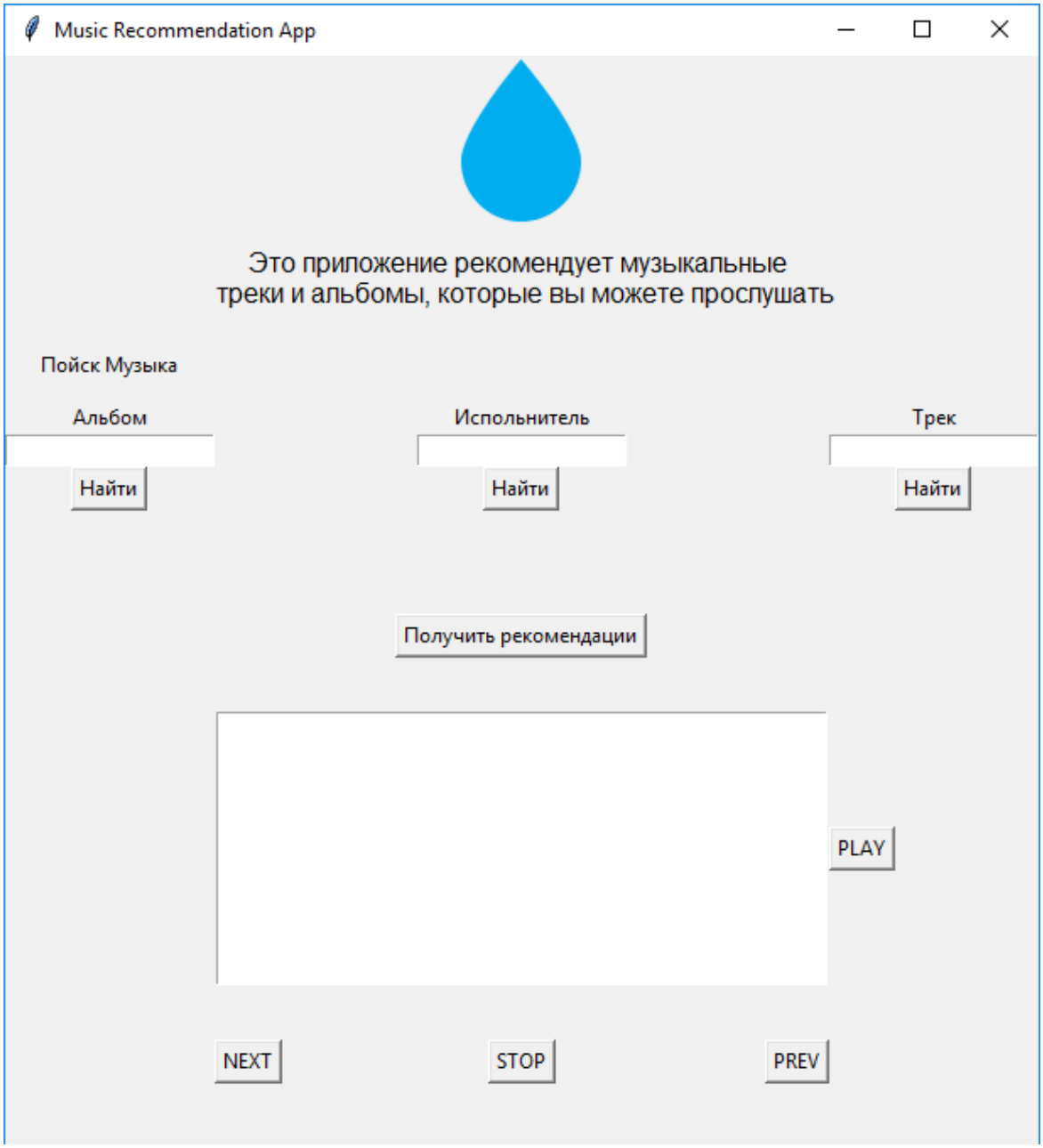

Рисунок 3.5 – Главная форма системы.

На этой форме реализованы музыкальный проигрыватель, который позволяет пользователю играть рекомендованные музыкальные файлы.

Заключение

На сегодняшний день информационные технологии используются во всех областях человеческой деятельности. Рекомендательные системы являются перспективным направлением развития информационных технологий в современном мире.

В ходе выполнения выпускной квалификационной работы был разработан полнофункциональный приложение для рекомендации и прослушивании музыки.

Система рекомендаций по музыке уникальна по сравнению с другими службами потоковой передачи музыки, поскольку она облегчает проблему холодного запуска и предоставляет персональные рекомендации. С постоянно растущим количеством музыки и увеличением числа людей с компьютерами будет возрастать потребность в продвинутых алгоритмах рекомендаций, и это удовлетворяет этот спрос.

Система стала очень сложной, и многие улучшения могут быть сделаны для дальнейшего улучшения рекомендаций. Непосредственной целью является интеграция системы с Facebook и Vkontake, которые предоставит более контекстуальную информацию о пользователе. С улучшенным профилем, созданным вокруг каждого пользователя, алгоритм K-Nearest Neighbor может найти лучшие совпадения, еще больше повышающие точность рекомендаций.

Другие цели включают создание приложения для мобильных устройств, чтобы обеспечить легкий способ для любого пользователя смартфона обнаружить и прослушать новую музыку. Это связано с увеличением числа мобильных устройств сегодня.

С уникальным подходом к идентификации похожих песен и вынесением рекомендаций, проект может стать рыночным решением, если законы о лицензировании музыки учитываются.

Список используемых источников

1. Dunning, T. Practical Machine Learning: Innovations in Recommendation [Электронный ресурс] / Dunning T., Friedman E. – San Francisco: O'Reilly Media, Inc., 2014. – 48 p. Режим доступа: http://oreilly.com/catalog/errata.csp?isbn=9781491950388 (дата обращения: 05.05.2018 г.)

2. Голицына, О.Л. Базы данных / О.Л. Голицына, Н.В. Максимов, И.И. Попов. - М.: Форум, 2014. - 352 с.

3. Документация на английском по CA ERwin® Data Modeler [Электронный ресурс] Режим доступа:– // http://erwin.com/

4. Маклаков, С. В. Создание информационных систем с AllFusion Modeling Suite / С. В. Маклаков. – М.: Диалог – МИФИ, 2017. – 432 с

5. Burgoyne, John Ashley, Ichiro Fujinaga, and J. S. Downie. Music information retrieval., 2016. – 213 p. [Электронный ресурс] Режим доступа: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118680605.ch15 (дата обращения: 15.04.2018 г.)

6. Downie, J. S. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research., 2008. – 255 p. [Электронный ресурс] Режим доступа: https://www.jstage.jst.go.jp/article/ast/29/4/29_4_247/_article/-char/ja (дата обращения: 15.04.2018 г.)

7. Jayashree, D., S. Goutham Manian, and C. Pranav Srivatsav. Music Recommendation System. Asian Journal of Information Technology, 2016. – 5 p. [Электронный ресурс] Режим доступа: http://docsdrive.com/pdfs/medwelljournals/ajit/2016/4250-4254.pdf (дата обращения: 15.04.2018 г.)

8. Авхадеев, Булат Ринатович, Лилия Ивановн Воронова, and Елена Павловна Охапкина. Разработка рекомендательной системы на основе данных из

профиля социальной сети «Вконтакте». Вестник Нижневартовского государственного университета ,2014. [Электронный ресурс] Режим доступа:http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ssi&paperid=286&option_lang=rus (дата обращения: 03.05.2018 г.)

9. Нефедова Ю.С. Архитектура гибридной рекомендательной системы GEFEST (Generation–Expansion–Filtering–Sorting–Truncation). Системы и средства информатики 2012. –22 p. [Электронный ресурс] Режим доступа:http://www.mathnet.ru/links/de53359b9b773cbff08a77dd19835671/ssi286.pdf (дата обращения: 06.05.2018 г.)

10. Helmers, Scott A. Microsoft Visio 2016 Step by Step. Microsoft Press, 2015.– 560 p. [Электронный ресурс] Режим доступа: https://play.google.com/store/books/details?id=kPA0CwAAQBAJ&rdid=book-kPA0CwAAQBAJ&rdot=1&source=gbs_atb&pcampaignid=books_booksearch_atb (дата обращения: 08.05.2018 г.)

11. Макашова В.Н, Галина Н.Ч. Модернизация ИТ-инфраструктуры образовательных учреждений в целях обеспечения информационной безопасности. Современные информационные технологии и ИТ-образование 10 (2014). [Электронный ресурс] Режим доступа: https://cyberleninka.ru/article/n/modernizatsiya-it-infrastruktury-obrazovatelnyh-uchrezhdeniy-v-tselyah-obespecheniya-informatsionnoy-bezopasnosti (дата обращения: 01.05.2018 г.)

12. Koren, Yehuda, and Robert Bell. Advances in collaborative filtering.Recommender systems handbook. Springer, Boston, MA, 2015. –42 p. [Электронный ресурс] Режим доступа: https://s3.amazonaws.com/academia.edu.documents/36167999/Collaborative-Filtering-_Koren-and-Bell_.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1527584469&Signature=mZDRw3XLvza9t6q%2FtAPogMH%2Fpmk%3D&response-content-

disposition=inline%3B%20filename%3DAdvances_in_Collaborative_Filtering.pdf (дата обращения: 13.05.2018 г.)

13. Li, Zelong, Mengxing Huang, and Yu Zhang. A Collaborative Filtering Algorithm of Calculating Similarity Based on Item Rating and Attributes. Web Information Systems and Applications Conference (WISA), 2017 14th. IEEE, 2017. [Электронный ресурс] Режим доступа: https://link.springer.com/chapter/10.1007/978-1-4899-7637-6_3 (дата обращения: 14.05.2018 г.)

14. Deng, Z., Zhu, X., Cheng, D., Zong, M., & Zhang, S. (2016). Efficient kNN classification algorithm for big data. Neurocomputing, 195, 2016. [Электронный ресурс] Режим доступа: http://keddiyan.com/files/AHCI/week2/3.pdf (дата обращения: 14.05.2018 г.)

15. Mishra, Raju Kumar. PySpark MLlib and Linear Regression. PySpark Recipes. Apress, Berkeley, CA, 2017. –265 p. [Электронный ресурс] Режим доступа: https://link.springer.com/chapter/10.1007/978-1-4842-3141-8_9 (дата обращения: 15.05.2018 г.)

16. Turrin, Roberto, et al. 30 Music Listening and Playlists Dataset. RecSys Posters. 2015. [Электронный ресурс] Режим доступа: http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html (дата обращения: 15.05.2018 г.)

17. Kaminskas, Marius, and Derek Bridge. Measuring surprise in recommender systems. Proceedings of the Workshop on Recommender Systems Evaluation: Dimensions and Design (Workshop Programme of the 8th ACM Conference on Recommender Systems). 2014. [Электронный ресурс] Режим доступа: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.707.7596&rep=rep1&type=pdf (дата обращения: 16.05.2018 г.)

18. Радынова, О. П. Музыкальное воспитание в семье. М.: Просвещение (2014). – 23 р. [Электронный ресурс] Режим доступа:

http://www.orensad5.ru/files/kopilka/muz_vosp_v_semje.pdf (дата обращения: 16.04.2018 г.)

19. Филиппов, С. А., et al. Организация больших объемов данных в рекомендательных системах поддержки жизнеобеспечения, входящих в состав глобальных платформ электронной коммерции. Институт проблем информатики ФИЦ ИУ РАН. Selected Papers of the XVII International Conference on Data Analytics and Management in Data Intensive Domains (DAMDID/RCDL 2015). 2015. –6 р. [Электронный ресурс] Режим доступа: https://pdfs.semanticscholar.org/58c8/8592a2a01fd5b34da036991be3cbdc49291e.pdf (дата обращения: 16.04.2018 г.)

20. Галяшина, Елена Игоревна. Судебная фоноскопическая экспертиза: проблемы диагностики аутентичности фонограмм. Вестник Университета имени ОЕ Кутафина 3 (2014). [Электронный ресурс] Режим доступа: https://cyberleninka.ru/article/n/sudebnaya-fonoskopicheskaya-ekspertiza-problemy-diagnostiki-autentichnosti-fonogramm (дата обращения: 17.04.2018 г.)

21. Аверченков В.Ф., Лозбинев А.Т. Информационные системы в производстве и экономике: учебное пособие. Litres, 2015. [Электронный ресурс] Режим доступа: https://books.google.ru/books?hl=en&lr=&id=xbJlAAAAQBAJ&oi=fnd&pg=PA5&dq=Анализ+требований+—+это+часть+процесса+разработки+программного+обеспечения,+включающая+в+себя+сбор+требований+к+программному+обеспечению&ots=wi-N-iy62n&sig=bn_pmT2s4Nka53uUMhGyUz64--w&redir_esc=y#v=onepage&q&f=false (дата обращения: 19.04.2018 г.)

22. Gasser, Martin, et al. Classical Music on the Web-User Interfaces and Data Representations. ISMIR. 2015. [Электронный ресурс] Режим доступа: http://raimond.me.uk/phd/thesis.pdf (дата обращения: 20.04.2018 г.)

23. Горбунова, И.Б. Информационные технологии в музыке и комплексная модель её семантического пространства. //Научно-технические ведомости

Санкт-Петербургского государственного политехнического университета. Гуманитарные и общественные науки –2014.– [Электронный ресурс] Режим доступа: https://cyberleninka.ru/article/n/informatsionnye-tehnologii-v-muzyke-i-kompleksnaya-model-eyo-semanticheskogo-prostranstva (дата обращения: 02.04.2018 г.)

24. Тунакова Ю. А, Р. И. Файзуллин, В. С. Валиев. Расчет вероятности поступления металлов в организм с потребляемой питьевой водой. Гигиена и санитария 94.5 –2015. –[Электронный ресурс] Режим доступа: hhttps://cyberleninka.ru/article/n/bayesovskie-podhody-k-opredeleniyu-kariesogennyh-streptokokkov-v-zubnoy-blyashke-u-detey-s-distalnoy-okklyuziey-pri-ortodonticheskom (дата обращения: 02.04.2018 г.)

25. Гришанов, К. М., Ю. С. Белов. Метод классификации k-nn и его применение в распознавании символов. В сборнике: Фундаментальные проблемы науки сборник статей Международной научно-практической конференции: Тюмень НИЦ АЭТЕРНА, 2016. – 317 с.

26. Леонов, В. П. Логистическая регрессия в медицине и биологии. –2016. – [Электронный ресурс] Режим доступа: hhttps://cyberleninka.ru/article/n/bayesovskie-podhody-k-opredeleniyu-kariesogennyh-streptokokkov-v-zubnoy-blyashke-u-detey-s-distalnoy-okklyuziey-pri-ortodonticheskom (дата обращения: 19.04.2018 г.)

ПРИЛОЖЕНИЕ А

ЛИСТИНГ ПРОГРАММЫ

menu.py

```
from tkinter import *
import register
import menu
from PIL import ImageTk, Image
root = Tk()
root.title('Login')
root.minsize(300,300)
path = "drop.png"
img = ImageTk.PhotoImage(Image.open(path))
header = Label(root,image=img,padx=20,pady=30)
header.grid(row =0,column=2,columnspan=3)
userNameLabel=Label(root,text='Логин',padx=30,pady=20)
userNameLabel.grid(row=1,column=1)
passwordLabel=Label(root,text='Пароль',padx=30,pady=10)
passwordLabel.grid(row=3,column=1)
userName=Entry(root)
userName.grid(row=1,column=3)
password=Entry(root,show="*")
password.grid(row=3,column=3)
enterButton = Button(root,text ='Войти',relief=GROOVE)
enterButton.grid(row=5,column=3)
enterButton.bind("<Button-1>",menu.mainWindow)
regButton = Button(root,text ='Зарегистрироваться',relief=GROOVE)
regButton.grid(row=6,column=3)
regButton.bind("<Button-1>",register.openRegister)
root.mainloop()
```

register.py

```
from tkinter import *
import menu
```

```
from PIL import ImageTk, Image
def openRegister(event):
    root = Toplevel()
    root.minsize(300,400)
    path = "drop.png"
    img = ImageTk.PhotoImage(Image.open(path))
    header = Label(root,image=img,padx=20,pady=30)
    header.grid(row =0,column=2,columnspan=3)
    name=Label(root,text='Фамилия',padx=30,pady=20)
    name.grid(row=1,column=1)
    entername=Entry(root)
    entername.grid(row=1,column=3)
    lastname=Label(root,text='Имя',padx=30,pady=10)
    lastname.grid(row=3,column=1)
    enterlastname=Entry(root)
    enterlastname.grid(row=3,column=3)
    patromynic=Label(root,text='Отчество',padx=30,pady=20)
    patromynic.grid(row=5,column=1)
    enterpatromynic=Entry(root)
    enterpatromynic.grid(row=5,column=3)
    gender=Label(root,text='Пол',padx=30,pady=10)
    gender.grid(row=7,column=1)
    entergender=Entry(root)
    entergender.grid(row=7,column=3)
    country=Label(root,text='Страна',padx=30,pady=10)
    country.grid(row=9,column=1)
    entercountry=Entry(root)
    entercountry.grid(row=9,column=3)
    city=Label(root,text='Город',padx=30,pady=10)
    city.grid(row=11,column=1)
    entercity=Entry(root)
    entercity.grid(row=11,column=3)
    email=Label(root,text='Эл. адрес',padx=30,pady=10)
    email.grid(row=13,column=1)
```

```
    enteremail=Entry(root)
    enteremail.grid(row=13,column=3)
    enterButton = Button(root,text ='Войти',relief=GROOVE)
    enterButton.grid(row=15,column=3)
    enterButton.bind("<Button-1>",menu.mainWindow)
    root.mainloop()
```

menu.py

```
import os
from tkinter.filedialog import askdirectory
import pygame
from mutagen.id3 import ID3
from tkinter import *
from PIL import ImageTk, Image
def mainWindow(event):
    root = Toplevel()
    root.minsize(500,400)
    path ="drop.png"
    img = ImageTk.PhotoImage(Image.open(path))
    path1 ="play.png"
    play = ImageTk.PhotoImage(Image.open(path1))
    path2 ="next.png"
    next = ImageTk.PhotoImage(Image.open(path2))
    path3 ="previous.png"
    previous = ImageTk.PhotoImage(Image.open(path3))
    path4 ="stop.png"
    stop = ImageTk.PhotoImage(Image.open(path4))
    listofsongs = []
    realnames = []
    v = StringVar()
    songlabel = Label(root,textvariable=v,width=35)
    index = 0
    def directorychooser():
        directory = askdirectory()
        os.chdir(directory)
```

```
    for files in os.listdir(directory):
        if files.endswith(".mp3"):
            realdir = os.path.realpath(files)
            audio = ID3(realdir)
            #if 'TIT2' in audio:
            song =audio.getall('TIT2')
            realnames.append(song)
            listofsongs.append(files)
    pygame.mixer.init()
    pygame.mixer.music.load(listofsongs[0])
    #pygame.mixer.music.play()
directorychooser()
def updatelabel():
    global index
    global songname
    v.set(realnames[index])
    #return songname
def playsong(event):
    pygame.mixer.music.play()
    updatelabel()
def nextsong(event):
    global index
    index += 1
    pygame.mixer.music.load(listofsongs[index])
    pygame.mixer.music.play()
    updatelabel()
def prevsong(event):
    global index
    index -= 1
    pygame.mixer.music.load(listofsongs[index])
    pygame.mixer.music.play()
    updatelabel()
def stopsong(event):
    pygame.mixer.music.stop()
```

```
        v.set("")
        #return songname
    def searchAlbum(event):
        print('It is working')
    def searchArtist(event):
        print('It is working')
    def searchTrack(event):
        print('It is working')
    header = Label(root,image=img,padx=20,pady=30)
    header.grid(row =0,column=1,columnspan=3)
    subheader = Label(root,text='Это приложение рекомендует музыкальные \n треки и альбомы, которые вы можете прослушать',font=20)
    subheader.grid(row =2,column=1,columnspan=3,padx =10, pady =10)
    searchTitle = Label(root,text='Пойск Музыка')
    searchTitle.grid(row=3,column=1,padx =10, pady =10)
    album = Label(root,text='Альбом')
    album.grid(row=4,column=1)
    artist = Label(root,text='Испольнитель')
    artist.grid(row=4,column=2)
    track=Label(root,text='Трек')
    track.grid(row=4,column=3)
    albumEntry=Entry(root)
    albumEntry.grid(row=5,column=1)
    artistEntry=Entry(root)
    artistEntry.grid(row=5,column=2)
    trackEntry=Entry(root)
    trackEntry.grid(row=5,column=3)
    enterAlbum = Button(root,text = 'Найти')
    enterAlbum.grid(row=6,column=1)

    enterArtist = Button(root,text = 'Найти')
    enterArtist.grid(row=6,column=2)
    enterTrack = Button(root,text='Найти')
    enterTrack.grid(row=6,column=3)
```

```
enterAlbum.bind("<Button-1>",searchAlbum)
enterArtist.bind("<Button-1>",searchArtist)
enterTrack.bind("<Button-1>",searchTrack)
blank =Label(root,text='')
blank.grid(row=7,column=2,pady=20,padx=20)
recommedButton = Button(root,text = 'Получить рекомендации')
recommedButton.grid(row=8,column=2)
blank1 =Label(root,text='')
blank1.grid(row=9,column=2,pady=5)
listbox = Listbox(root,width=60)
listbox.grid(row=10,column=2)
listbox.insert(1,"First Song")
listbox.insert(2,"First Song")
listbox.insert(3,"First Song")
listbox.insert(4,"First Song")
listbox.insert(5,"First Song")
listbox.insert(6,"First Song")
listbox.insert(7,"First Song")
listbox.insert(8,"First Song")
listbox.insert(9,"First Song")
listbox.insert(10,"First Song")
#listofsongs.reverse()
realnames.reverse()
for items in realnames:
    listbox.insert(0,items)
realnames.reverse()
#listofsongs.reverse()
blank2 =Label(root,text='')
blank2.grid(row=11,column=2,pady=5)

playbutton=Button(root,text='PLAY')
playbutton.grid(row=10,column=3,sticky=W)
nextbutton = Button(root,text='NEXT')
nextbutton.grid(row=12,column=2,sticky=W)
```

```
    previousbutton = Button(root,text='PREV')
    previousbutton.grid(row=12,column=2,sticky=E)
    stopbutton = Button(root,text='STOP')
    stopbutton.grid(row=12,column=2,sticky=S)
    blank3 =Label(root,text='')
    blank3.grid(row=13,column=2,pady=5)
    playbutton.bind("<Button-1>",playsong)
    nextbutton.bind("<Button-1>",nextsong)
    previousbutton.bind("<Button-1>",prevsong)
    stopbutton.bind("<Button-1>",stopsong)
    songlabel.grid(row=14,column=2)
    root.mainloop()
```

main.py

```
from pyspark.mllib.recommendation import *
import random
from operator import *
from pyspark import SparkContext
sc = SparkContext("local", "Music Recommendation App")
#Loading data into RDD
artistData = sc.textFile("artist_data_small.txt")
artistAlias = sc.textFile("artist_alias_small.txt")
userArtistData = sc.textFile("user_artist_data_small.txt")
alias_data = artistAlias.collect()
user_data = userArtistData.collect()
artist_canonical_dict = {}
user_list = []
for line in alias_data:
    artist_record = line.split("\t")
    artist_canonical_dict[artist_record[0]] = artist_record[1]
#Function to get canonical artist names
def canonicalArtistID(line):
    line = line.split(" ")
```

```
        if line[1] in artist_canonical_dict:
            return (int(line[0]),int(artist_canonical_dict[line[1]]),int(line[2]))
        else:
            return (int(line[0]),int(line[1]),int(line[2]))
        #Getting canonical artist names
    userArtistData = userArtistData.map(canonicalArtistID)

    #Creating allArtists dataset to be used later during model evaluation process
    allArtists = userArtistData.map(lambda x:(x[1])).collect()
    allArtists = list(set(allArtists))
    artist_data = artistAlias.collect()
      user_play_count = {}
    user_count_number = {}
    for line in user_data:
        user_record = line.split()
        if user_record[0] in user_play_count:
            user_play_count[str(user_record[0])] = user_play_count[user_record[0]] +
int(user_record[2])
            user_count_number[str(user_record[0])] = user_count_number[user_record[0]] + 1
        else:
            user_play_count[str(user_record[0])] = int(user_record[2])
            user_count_number[str(user_record[0])] = 1
    top = 0
    maximum = 2

    for word, count in sorted(user_play_count.items(),key = lambda kv: (-kv[1], kv[0]), reverse =
True):
        if top > maximum:
          break
        print( 'User ' + str(word) + ' has a total play count of ' + str(count) + ' and a mean play count
of ' + str(count/user_count_number[word]))
        top += 1

    #Splitting the data into train, test and cross validation
```

```
trainData, validationData, testData = userArtistData.randomSplit([4, 4, 2], 13)

print (trainData.take(3))
print (validationData.take(3))
print (testData.take(3))
print (trainData.count())
print (validationData.count())
print (testData.count())

#Caching and creating ratings object
trainData = trainData.map(lambda l: Rating(*l)).cache()
validationData = validationData.map(lambda l: Rating(*l)).cache()
testData = testData.map(lambda l: Rating(*l)).cache()

from pyspark.mllib.recommendation import ALS, MatrixFactorizationModel, Rating
from collections import defaultdict
#model evaluation function
def modelEval(model, dataset):
    global trainData
    global allArtists
        #Getting nonTrainArtists for each user
    userArtists = defaultdict(list)

    for data in trainData.collect():
        userArtists[data[0]].append(data[1])

    cvList = []

    for key in userArtists.keys():
        userArtists[key] = list(set(allArtists) - set(userArtists[key]))
        for artist in userArtists[key]:
            cvList.append((key, artist))

    #Creating user,nonTrainArtists RDD
```

```
        cvData = sc.parallelize(cvList)

        userOriginal = dataset.map(lambda x:(x.user, (x.product,
x.rating))).groupByKey().collect()

        #prediction on the user, nonTrainArtists RDD
        predictions = model.predictAll(cvData)
        userPredictions = predictions.map(lambda x:(x.user, (x.product,
x.rating))).groupByKey().collect()
        original = {}
        predictions = {}

        #Getting top X artists for each user
        for line in userOriginal:
            original[line[0]] = sorted(line[1], key=lambda x:x[1], reverse = True)

        for line in userPredictions:
            predictions[line[0]] = sorted(line[1], key=lambda x:x[1], reverse = True)

        similarity = []

        for key in userOriginal:
            similar = 0.0

            pred = predictions[key[0]]
            org = original[key[0]]

            for value in org:
                for item in pred[0:len(org)]:
                    if (value[0] == item[0]):
                        similar += 1
                        break

            #Similarity calculation
```

```
            similarity.append(float(similar/len(org)))

        string = "The model score for rank " + str(rank) + " is " +
str(float(sum(similarity)/len(similarity)))
        print (string)
        #Model evaluation through different rank parameters
    rank_list = [2, 10, 20]

    for rank in rank_list:
        model = ALS.trainImplicit(trainData, rank, seed=345)
        modelEval(model,validationData)

    bestModel = ALS.trainImplicit(trainData, rank=10, seed=345)
    modelEval(bestModel, testData)
    ratings = bestModel.recommendProducts(1059637, 5)

    import re
    artist_data = artistData.collect()
    artist_names_dict = {}

    for line in artist_data:
        pattern = re.match( r'(\d+)(\s+)(.*)', line)
        artist_names_dict[str(pattern.group(1))] = pattern.group(3)

    for i in range(0,5):
        if str(ratings[i].product) in artist_canonical_dict:
            artist_id = artist_canonical_dict[str(ratings[i].product)]
            print  ("Artist " + str(i) + ": "  + str(artist_names_dict[str(artist_id)]))
        else:
            print ("Artist " + str(i) + ": " + str(artist_names_dict[str(ratings[i].product)]))
```

Обозначение	Наименование	Дополнительные сведения
	Текстовые документы	
1. 11070026.09.03.02.427.ПЗВКР	Пояснительная записка	59 с.
	Графические документы	
2. 11070026.09.03.02.427.ДМВКР	Демонстрационные материалы (презентация)	11 сл.
	Демонстрационные материалы (пл. ф. А4)	11 лист.*5 экз
	Другие документы	
3.	Документы на компакт-диске	1 CD

Изм.	Лист.	Номер докум.	Подп.	Дата	11070026.09.03.02.427.ПЗВКР		
Разработал		Нгамби С.			Тема ВКР Разработка информационной библиотечной системы хранения звукозаписей устной речи	Лит.	Лист 58 / Листов 59
Проверил		Лихолоб П.Г.				У	
Н.контр.		Нестерова Е.В.				НИУ «БелГУ» гр.07001408	
Утвердил		Иващук О.А.			Ведомость ВКР		

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*