МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ПО ДАННЫМ СОЦИАЛЬНЫХ СЕТЕЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Мкртчян Арташеса Асканазовича

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванов

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2019

ВВЕДЕНИЕ

Практически каждый человек активно использует социальные сети, интернет магазины и сервисы для просмотра фильмов или прослушивания музыки. Во всех таких системах применяются технологии рекомендательных систем для предложения подходящих друзей, новостей, товаров, фильмов и так далее. Без них поиск нужной информации будет занимать значительно больше времени, тем самым ограничивать функционал системы и ее востребованность.

Рекомендательные системы обрели популярность в конце двадцатого века, с появлением первой статьи о совместной фильтрации в середине 90-тых годов. С тех пор они стали важной частью в области поиска новой информации, помогая пользователям идентифицировать данные, продукты и услуги с агрегированием и анализом предложений от других пользователей. Уже несколько десятилетий рекомендательные системы остаются одними из самых известных, легко монетизируемых и тиражируемых применений систем искусственного интеллекта.

Несмотря на популярность, текущие реализации рекомендательных систем не лишены недостатков. Например, социальная сеть «ВКонтакте» позволяет получить в виде рекомендаций самые популярные сообщества в системе, но при этом никак не рассматриваются интересы самого пользователя, что определенно усложняет поиск подходящих групп.

Цель бакалаврской работы — разработать программный продукт, позволяющий рекомендовать персональный сообщества в социальной сети «ВКонтаке».

Поставленная цель определила **следующие задачи**:

- Рассмотреть популярные технологии веб-разработки;
- Проанализировать основные алгоритмы рекомендательных систем;
- Разработать серверное приложение, выполняющее функции рекомендательной системы.

Структура и объем работы. Бакалаврская работа состоит из введения, 3 разделов, заключения, списка использованных источников и 9 приложений. Общий объем работы — 68 страниц, из них 36 страниц — основное содержание, включая 7 рисунков, список использованных источников информации — 21 наименований.

1 Рекомендательные системы

1.1 Постановка задачи

Проблема построения рекомендательных систем на данный момент актуальна для многих областей и является частью таких задач, как предложение товаров в интернет-магазинах, ранжирование результатов выдачи в поисковых системах, поиск подходящего контента в музыкальных, видео сервисах и СМИ. Одной из немаловажных областей применения являются социальные сети, где рекомендации используются для подбора друзей и сообществ, для рекламы товаров и сторонних услуг, для улучшения новостной ленты и так далее. Однако, несмотря на их хорошую интеграцию и точную работу, рекомендательные системы социальных сетей не лишены недостатков. Например, «ВКонтакте» не предоставляет возможность своим пользователям подбирать сообщества по персональным рекомендациям, выводя только наиболее популярные в качестве предложений. Эту проблему и требуется решить.

1.2 Основной принцип работы

Целью любой рекомендательной системы является информирование пользователей о товарах и услугах, которые могут ему показаться интересными в данный момент времени. Говоря более формально, на основе обработки имеющиеся информации система пытается угадать насколько сильно понравится тот или иной продукт потенциальному клиенту. Если рекомендации работают качественно и грамотно, то потенциально пользователь получает нужный товар, а сервис выгоду. При этом не обязательно от прямой продажи услуг, но и за счет популяризации своей площадки, которая в дальнейшем принесет доходы с рекламы.

Также можно ввести классификацию по степени персонализации:

1. Не персональные рекомендации — в таких рекомендациях могут учитываться регион проживания пользователя или время использования, но при этом не рассматриваются персональные предпочтения. Говоря иначе, всем пользователям предлагаются одинаковые товары, на основе их популярности, общего рейтинга и других параметров;
2. Сессионные рекомендации — когда для предложений используется история текущей сессии. Например, какие товары просматривались или что добавлялся в корзину;

3. Персональные рекомендации — берется во внимание всевозможная информация пользователя: возраст, пол, история покупок, список друзей и многое другое. Этот вариант является наиболее сложным, но при этом показывает хорошее качество рекомендаций.

Ядром рекомендательных систем является таблица отношения пользователей и товаров, так называемая матрица предпочтений. Как показано на рисунке 1, по одной из осей отложены все клиенты сервиса, а по другой — объекты рекомендации. Если пользователь выражал свою заинтересованность в товаре, то на пересечении матрица будет заполнена соответствующей оценкой. При этом это не обязательно явная шкала оценки, в котором клиент проголосовал, но и возможно покупка товара или вступление в сообщество. Таким образом, оценки можно получить двумя способами:

1. явно (explicit ratings) — клиент явно выражает свое отношение к товару, например, поставив рейтинг или оставив отзыв;
2. неявно (implicit ratings) — пользователь явно свое ничего не выражает, но можно оценить его отношение, анализируя косвенные данные: время просматривание, частота использования какого-то ресурса и так далее.

	Товар 1	Товар 2	Товар 3	Товар 4	Товар 5
Клиент 1		3		5	
Клиент 2	1		1	1	
Клиент 3	2			3	2
Клиент 4		4			5
Клиент 5	5		2	3	4

Рисунок 1 – Матрица предпочтений

В основном эта матрица заполнена частично, так как пользователи оценивают лишь малую часть товаров. Главная задача рекомендательных систем рассмотреть имеющеюся информации и на основе этого предсказать отношение клиента к другим товарам, о которых нет данных. То есть заполнить все пустые ячейки и, выбрав какое-то количество наиболее релевантных, рекомендовать их.

1.3 Не персонализированные рекомендации

Самыми простыми в реализации являются алгоритмы не персонализированных рекомендаций, в которых основной метрикой является средняя оценка товара среди всех пользователей, то есть если нравится всем, то сервис считает, что понравится и новому пользователю. Этот принцип используется в основном для не авторизованных клиентов, когда другие данные собирать не возможно.

Сам же средний рейтинг товара может быть представлен по-разному: в виде звезд с описанием позиции, в виде общей суммы положительных отзывов или их разности с отрицательными. При этом пользователи относятся с большим доверием к таким рекомендациям, мнение большинства считается хорошим источником информации при выборе новых для себя продуктов. Но при этом предложения могут абсолютно не подходить текущему клиенту и их эффективность в среднем будет не очень хорошей.

1.4 Персональные рекомендации

Другой парадигмой рекомендательных систем являются персональные предложения. В этом случае сервис использует максимальное количество данных, которые можно собрать о пользователе, и пытается на основе них предсказать потенциальные позиции для покупки. Например, большую роль может сыграть история покупок. На этой почве появился алгоритм фильтрации основанный на содержимом (content-based filtering) [1]. Целью данного алгоритма является сопоставление интересов пользователя с описанием потенциального продукта. Интересы же можно сформулировать базируясь на предыдущих покупках и отзывах. Тем самым, результатом рекомендации будут товары, которые больше всех соответствуют текущему пользователю. Главным минусом данного подхода является обязательное требование к полным описаниям товаров в каталоге.

1.5 Коллаборативная фильтрация

Третьим вариантом реализации являются рекомендации на основе похожих пользователей. Также эта методология называется коллаборативной фильтрацией, из-за особенностей своей реализации, где рекомендациями считаются данные, полученные на основе «коллаборации» множества клиентов [2]. Основная идея методологии подобрать N наиболее близких пользователей и на

основе их интересов выбрать релевантные товары для предложения. Допустим необходимо рекомендовать пользователю новую книгу, для этого нужно найти ближайшие соседи, используя в качестве метрики, например, количество общих купленных книг. После определить какие из книг соседей еще не куплены текущим пользователем и среди них выбрать наиболее популярную и качественную [3].

1.6 Гибридный подход

Гибридные подходы сочетают коллаборативную и персональную фильтрацию, повышая эффективность и сложность рекомендательных систем. Объединение результатов нескольких подходов потенциально позволяет повысить точность рекомендаций [4]. Кроме того, гибридный подход может решить проблему отсутствия данных в начальный период времени для коллаборативных рекомендаций, получая сначала персональные предложения, а затем смешивая их с рекомендациями на основе похожих пользователей.

1.7 Рекомендации на основе контекста

Одним из путей повышения точности рекомендательной системы является расширение перечня используемой при формировании рекомендаций информацию, в частности использование контекста [5].

Под контекстом будем понимать атрибуты, так или иначе описывающие ситуацию, в которой пользователь оценил объект или получает рекомендации [6]. То есть в рамках систем могут быть учтены два вида контекста:

1. контекст, в котором происходит фиксация предпочтений пользователя;
2. контекст, в котором происходит формирование рекомендаций.

1.8 Проблемы рекомендательных систем

Текущие реалии предоставляют огромные возможности для сбора разнообразных данных, что сильно упрощает применение алгоритмов коллаборативной фильтрации и использования «мудрости толпы» [7]. Но и у большого количества информации есть свои отрицательные стороны, которые увеличивают сложность применения рекомендательных систем. Возьмем в качестве примера различие характеров людей, а именно, поведение пользователей может быть кардинально разным: некоторые вполне стабильные и поддаются моделированию, а другие, наоборот, ведут себя абсолютно непредсказуемо.

Наличие такого большого разброса моделей поведения приводит к ухудшению результатов работы рекомендательных систем и к снижению ее точности. С другой стороны недобросовестные продавцы могут изменить результаты рекомендации путем оставления множества положительных отзывов у своих товаров и, обратное, поставляя плохие оценки продуктам конкурентов. Тем самым увеличивая общий рейтинг для себя и вводя в заблуждения потенциальных покупателей. В идеале качественная рекомендательная система должна учитывать эти факторы и справляться с выше перечисленными проблемами.

2 Практическая реализация системы

Целью данной работы является изучение современных технологий веб-разработки, таких как Spring, VkSDK и Redis, и на основе них разработать собственную рекомендательную систему, которая, анализируя данные пользователя и его друзей, предлагает подходящие сообщества.

Система будет использовать рекомендации на основе похожих пользователей, то есть будут подобраны n наиболее близких друга, а на основе их сообществ выбран наилучший. Для реализации этой задачи буду разработаны логики:

- оптимальной загрузки данных с серверов «ВКонтакте»;
- определяющий схожесть пользователей;
- высчитывающий качество сообщества;
- функции веб-сервера: эндпоинты позволяющие взаимодействовать с сервисом;
- сохраняющих необходимые данные в кэш, для уменьшении времени работы.

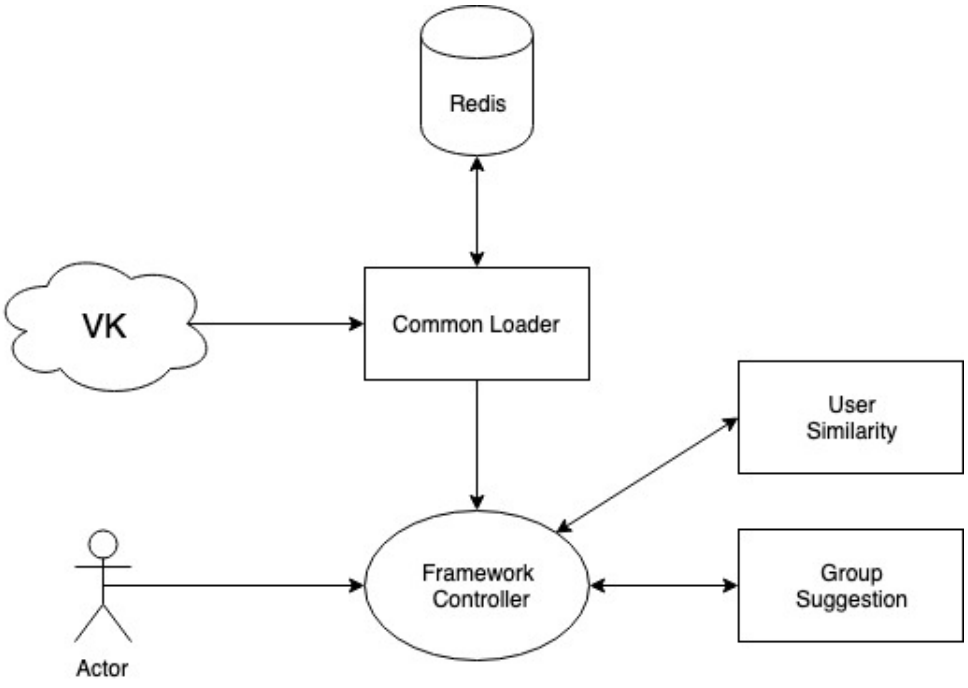

Рисунок 2 – Архитектура системы

Архитектура основных модулей системы приведена на рисунке 2. Пользователь взаимодействует с системой отправляя HTTP запросы и получая ответ на них. Сами запросы обрабатывает модуль `Framework Controller`, это один из основных модулей, объединяющий всю логику приложения. При получе-

нии запроса происходит загрузка данных из модуля Common Loader,после чего с помощью класса User Similarity определяются наиболее похожие друзья. На окончательном шаге с использованием модуля Group Suggestion определяются наилучшие сообщества и отправляются пользователю в виде JSON данных.

Сам же Common Loader загружает данные из кэша, если этот пользователь уже обрабатывался, иначе получает его данные из сервера «ВКонтакте».

2.1 Начальная настройка проекта

Изначально в файле pom.xml необходимо подключить нужные библиотеки и фреймворки к проекту.

Был создан файл конфигурации log4j2.xml для нормального функционирования логгера. Теперь в любом классе проекта можно создать объект логгера с помощью класса LoggerFactory и выводить сообщения в консоль используя метод info. Но так как очень много сообщений записываются в консоль как спрингом, так и VkApi был написан метод printInfo, позволяющий вывести сообщение выделенным от остальных. Этот метод принимает на вход объект логгера и текст, который нужно вывести. Был добавлен конфигурационный класс для Spring AppConfig.java, в дальнейшем здесь будут создаваться бины. Сам по себе класс отличается от других только аннотацией @Configuration. Также был добавлен класс запуска приложения SpringBootStarter.java с единственным методом main который и запускает приложение.

2.2 Регистрация в «ВКонтакте»

В настоящей работе происходит анализ социальной сети «ВКонтакте», использующей регистрацию с системным ключом доступа, то есть для обработки информации о пользователе нет необходимости авторизоваться под его профилем, а лишь достаточно знать системный ключ и чтобы страница была открыта для общего доступа. Этого нам более чем достаточно. Для удобной работы с VkSDK был создан класс VkApiManager содержащий два поля:

— VkApiClient vk — будет хранить объект VKApi, позволяющий вызывать методы получения данных;

— ServiceActor actor — содержит в себе информацию о текущей регистрации.

Зададим секретные параметры регистрации в качестве переменных окружения и внедрим их в класс конфигурации в виде стандартных Java полей, с использованием аннотации `@Value`:

— `appId` — номер приложения;
— `clientSecret` — секретный ключ клиента;
— `serviceCode` — сервисный ключ доступа.

2.3 Модели представления данных

Для удобного представления данных в программе используются модели — классы с нужным набором свойств и функциями их обработки, позволяющими получать, сохранять и изменять данные в удобной форме.

Для хранения данных пользователя и информации о группах используются стандартные модели VK: `UserXtrCounters` и `GroupFull` соответственно. Эти классы предоставляют обширный набор свойств и идеально подходят для хранения полученных из сервера данных.

Но с целью более комфортной работы эти модели были обвернуты в собственные, с добавлением дополнительной информации.

При сохранении данных в кэш возникла необходимость конвертации объектов Java в строку и обратное. С этой целью был разработан класс `ObjMapper` с двумя методами:

1. string2Object — десериализация строки в виде JSON в объект Java;
2. object2string — сериализация объекта в строку JSON.

2.4 Загрузка и кэшрование данных

Для загрузки данных о пользователе был написан класс `UserLoader`, в котором два поля: автоматический внедряемый `apiManager` и `LOADABLE_FIELDS`. В последнем перечислены поля необходимые для загрузки с сервера, например, пол, дата рождения и так далее.

Также были написаны методы для получения данных пользователя, списка его друзей и множества его сообществ: `fetchUser`, `fetchUserFriends` и `fetchUserGroups` соответственно.

Аналогично был написан класс `CommunityLoader` с одним методом для загрузки информации о сообществе `fetchCommunity`. При этом оба этих класса оснащены аннотацией `@Component`, что означает автоматическое создание бинов для них.

Далее был создан класс `CommonLoader`, состоящий из одного метода `fetchAllNecessaryData`, который позволяет скачать все необходимые данные и вернуть их в виде объекта модели `UsersWithGroups`. Рассмотрим этот метод поподробнее, в начале происходит проверка данных в кэше, если там уже есть нужный пользователь, то мы берем его данные из Redis и возвращаем их.

2.5 Классы для рекомендации

Для определения близких по интересам друзей был разработан класс `UserSimilarity`, с двумя методами:

- `userSimilarity(User first, User second)` — определяет на сколько близки два пользователя;
- `getSimilarUsers(UsersWithGroups usersWithGroups)` — на основе первого метода находит 10 близких пользователей.

Был разработан класс `GroupSuggestion` для определения наиболее качественных сообществ, состоящий из трех методов:

- `getNonRepeatingCorrectGroups` — удаляет группы, которые уже имеются у анализируемого пользователя;
- `groupQuality` — возвращает вещественное значение, характеризующее качество сообщества;
- `suggestGroups` — на основе предыдущих двух методов отбирает наиболее качественные группы.

2.6 Контроллер приложения

Также был разработан класс-контроллер, который и обеспечивает взаимодействие со системой. В самом классе два эндпоинта: проверка состояния системы и сами рекомендации.

Метод рекомендации доступен по пути `/recommendation` и является GET методом. На вход принимает обязательный параметр пути `userId`.

2.7 Настройка окружения приложения

Для полноценного функционирования системы необходимо сначала настроить связь с базой данных Redis, указав параметры окружения `redis.host` и `redis.port`: хост и порт до базы соответственно. Сам же Redis может быть установлен и запущен локально.

Кроме того, как уже было сказано, используется системная регистрация в «ВКонтакте», функционирование которой зависит от трех параметров: `app.id`, `client.secret` и `service.code`. Эти ключи можно получить зарегистрировав собственное приложение на официальном сайте «ВКонтакте»

Теперь можно запустить приложение используя интегрированную среду разработки «Intellij idea», указав параметр `server.port`, который уточняет порт на котором будет запущен сервер.

2.8 Пример использования

После запуска приложения можно сразу отправлять запросы, по умолчанию он слушает порт 8080. Например, проверим эндпоинт «/live» (см. рисунок 3):

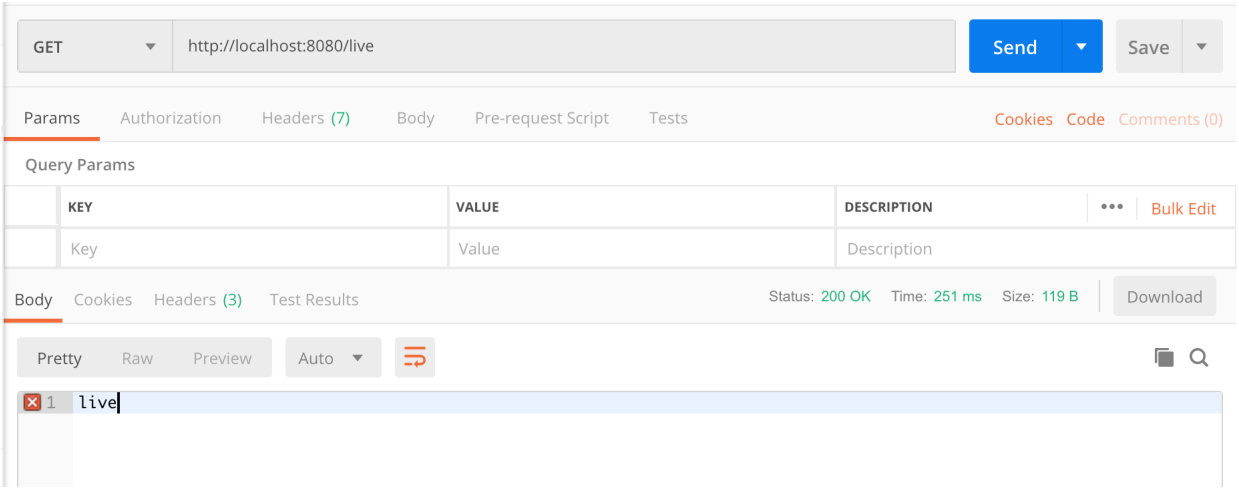

Рисунок 3 – Запрос на /live

Как можно заметить, сервер возвращает текст «live» и статус `200 OK`. После можно запросить рекомендации для пользователя с корректными данными (см. рисунок 4):

Сервер возвращает статус `200 OK` и список рекомендуемых сообществ. Если же ID будет не корректным, например профиль закрыт, то сервер вернет статус `406 Not Acceptable`, как показано на рисунке 5.

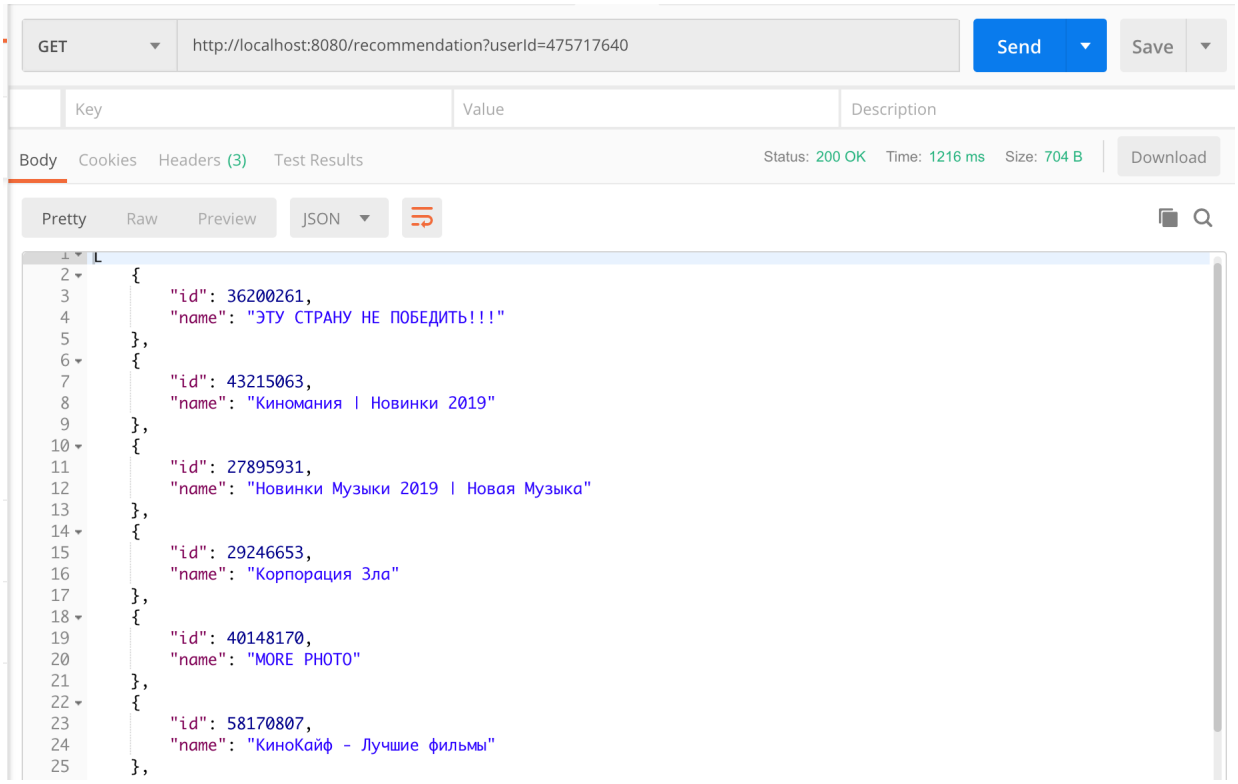

Рисунок 4 – Запрос рекомендаций с корректным пользователем

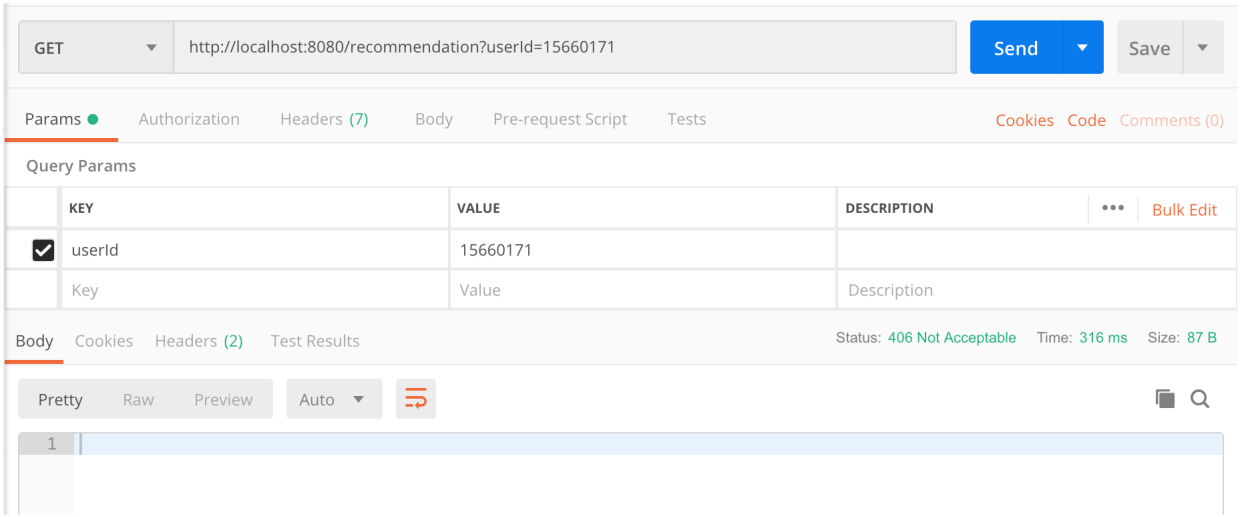

Рисунок 5 – Запрос рекомендаций с закрытым профилем

ЗАКЛЮЧЕНИЕ

Рекомендательные системы занимают сегодня важное место в области поиска новой информации. Почти любая система оснащена логикой предсказывания товаров и услуг, которые могут понравится пользователю.

В настоящей работе было разработано серверное приложение, выполняющее функции рекомендательной системы, для социальной сети «ВКонтакте», которая производит фильтрацию на основе данных пользователя, его друзей и описания сообществ, тем самым пытаясь предложить подходящие группы.

Таким образом, были решены задачи:

- Рассмотрены популярные технологии веб-разработки;
- Проанализированы основные алгоритмы рекомендательных систем;
- Разработано серверное приложение, выполняющее функции рекомендательной системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Burke, R.* Knowledge-based recommender systems / R. Burke // *Encyclopedia of Library and Information Science*. — 2000. — Pp. 180–200.

2 *Koren, Y.* Advances in collaborative filtering / Y. Koren, R. Bell // *Springer*. — 2011. — P. 2.

3 *Resnick, P.* Recommender systems / P. Resnick, R. Varian // *Communications of the ACM*. — 1996. — Pp. 56 – 58.

4 *Гомзин, А.* Системы рекомендаций: обзор современных подходов / А. Гомзин, А. Коршунов // *Москва*. — 2012. — С. 20.

5 Contextualizing Useful Recommendations [Электронный ресурс]. — URL: https://www.inf.unibz.it/~ricci/Slides/Context-UMAP-2012-Ricci.pdf (Дата обращения 24.05.2019). Загл. с экр. Яз. анг.

6 *Пономарев, А.* Обзор методов учета контекста в системах коллаборативной фильтрации / А. Пономарев // *Труды СПИИРАН*. — 2013. — С. 169–188.

7 Принципы работы рекомендательных механизмов Интернета [Электронный ресурс]. — URL: https://www.ibm.com/developerworks/ru/library/os-recommender1/index.html (Дата обращения 24.05.2019). Загл. с экр. Яз. рус.