МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____Г.И. Радченко
«___»_____2020 г.

Рекомендательная система с настраиваемым блоком анализа данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____И.Л. Надточий
«___»_____2020 г.

Автор работы,
студент группы КЭ-405
_____В.В. Гарманов
«___»_____2020 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____Г.И. Радченко
«___»_____2020 г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра
студенту группы КЭ-405
Гарманову Владимиру Владимировичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Рекомендательная система с настраиваемым блоком анализа данных» утверждена приказом по университету от 24 апреля 2020 г. №627

2. **Срок сдачи студентом законченной работы:** 1 июня 2020 г.

3. **Исходные данные к работе:**

– Рихтер, Дж. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. – 4-е издание. – Санкт-Питербург: Изд-во Питер, 2013. – 896 с.;
– Гудфеллоу, Я. Глубокое обучение / Бенджио И., Гудфеллоу Я., Курвилль А. – Москва: Изд-во ДМК Пресс, 2017. – 652 с.;
– Шилдт, Герберт C# 4.0. Полное руководство / Герберт Шилдт. – Москва: Изд-во Вильямс, 2015. – 670 с..

4. **Перечень подлежащих разработке вопросов:**

– провести анализ существующих подходов к проектированию рекомендательных систем;
– провести анализ существующих рекомендательных систем;

– спроектировать и разработать основную конструкцию рекомендательной системы;
– спроектировать и разработать тестовый алгоритм для спроектированной рекомендательной системы;
– протестировать работоспособность и эффективность разработанного алгоритма.

5. **Дата выдачи задания:** 20 января 2020 г.

Руководитель работы _____/*И.Л. Надточий*/

Студент _____/*В.В. Гарманов*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2020	
Разработка модели, проектирование	15.04.2020	
Реализация интерфейса рекомендательной системы	01.05.2020	
Реализация алгоритма анализа данных	15.05.2020	
Тестирование, отладка, эксперименты	20.05.2020	
Компоновка текста работы и сдача на нормоконтроль	24.05.2020	
Подготовка презентации и доклада	30.05.2020	

Руководитель работы _____/И.Л. Надточий/

Студент _____/В.В. Гарманов/

Аннотация

В.В. Гарманов. Рекомендательная система с настраиваемым блоком анализа данных. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2020, 48 с., 11 ил., библиогр. список – 15 наим.

В рамках выпускной квалификационной работы производится детальный анализ современных рекомендательных систем и основных методов их построения. Производится изучение и анализ требований к программным продуктам для рекомендательных систем. Рассматриваются преимущества и недостатки существующих реализаций рекомендательной системы. Организуется разработка программы рекомендательной системы с настраиваемым блоком анализа данных с использованием ПО Visual Studio 2017. Доказывается способность предлагаемого программного продукта конкурировать с существующими программными продуктами.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире не малый процент экономических ресурсов сосредоточен в сфере Интернет. Это может быть, как непосредственная продажа товаров или услуг на сайтах компаний, так и сбор средств для start-up-ов, продажа рекламы на различных сайтах, ускоренный обмен информацией, который может принести какую-либо прибыль так и многое другое.

В продаже каких-либо товаров или услуг у продавцов встает вопрос о том, как продавать свои товары и услуги более эффективно. Существует множество способов реализовать подобное. Владельцы магазинов могут нанимать консультантов, которые будут помогать в выборе товаров покупателям. Однако данный способ не всегда будут полезен, например для таких интернет-магазинов, как AliExpress или eBuy, где одновременно находится огромное количество продавцов, реализовать консультацию клиентов по всем этим магазинам одновременно невозможно, разве что отдельные магазины могут рекомендовать товары в пределах их магазинов. Так же следует отметить, что наём консультантов так же приносит фирмам дополнительные затраты.

Чтобы реализовать помощь клиентам в выборе товара и при этом убрать или уменьшить влияние вышеописанных проблем были придуманы специальные средства – **рекомендательные системы**.

Рекомендательная система – это программа, имеющая свой логику работы, алгоритм, который осуществляет анализ поступающей на него информации пользователя и на её основании выводящий какой-либо продукт, который предположительно может быть интересен данному пользователю. При этом стоит отметить, что данный продукт не обязательно должен иметь материальную форму, это может быть, например услуга, фильм, игра, музыка, и многое другое. Принцип работы любой рекомендательной системы графически изображен на рисунке 1.

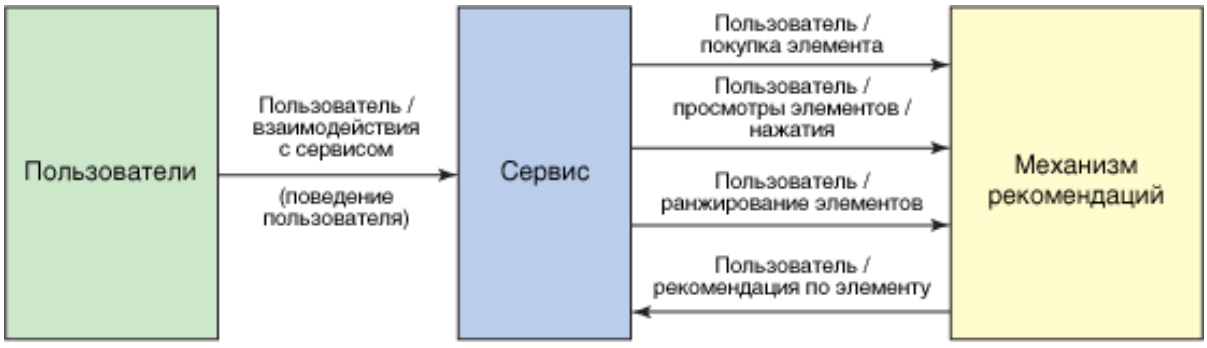

Рисунок 1 – Принцип работы рекомендательных систем

В первую очередь рекомендательные системы используются в интернет-коммерции. В таких фирмах анализируют информацию пользователей для продвижения своих товаров или услуг. Примерами таких фирм являются Amazon, eBuy, AliExpress, iTunes, Steam и другие…

Другое место применения рекомендательных систем заключается в продвижении фильмов, игр, книг и музыки на сайтах, где не осуществляется прямая продажа этих продуктов, а лишь их возможное размещение для просмотра или скачивания. Другими словами применение рекомендательных систем на подобных сайтах не является основным средством повышения прибыли, но может косвенно способствовать этому. Примерами подобных сервисов являются GoodReads, IMDb и другие…

На текущий момент времени существует огромное количество различных вариаций реализации данных систем. Самые простые вариации зачастую в качестве рекомендованных продуктов выставляют самые популярные продукты, возможно с небольшими дополнениями в виде, например для книг предлагать самые популярные книги подобного, просматриваемому жанру. Подобные системы строятся достаточно просто, и любой сайт очень легко может использовать их. Такие системы отличаются относительно низкой эффективностью и обычно в Интернет-торговле не применяются.

Наиболее мощные средства же, уже включают в себя мощные средства оценивания потребностей клиентов. Такие системы обычно очень тесно связаны со многими отраслями искусственного интеллекта. Такие системы тоже не лишены своих недостатков: слишком мощные аналитические системы могут эффективно работать далеко не на всех серверных машинах, что для маленьких сайтов или магазинов так же не позволяет использовать их.

В данной работе преследуется цель разработать программный продукт рекомендательной системы, который позволит владельцам сайтов самостоятельно настраивать эту систему в зависимости от их обстоятельств и таким образом позволить им обойти, хотя бы частично, эти недостатки.

Детальное рассмотрение предметной области содержится в следующих главах выпускной квалификационной работы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Введение в основные методы построения рекомендательных систем

Перед непосредственным этапом разработки программного продукта следует составить общее представления о существующих аналогах и о методах решения поставленной проблемы.

Существует огромное множество методов создания рекомендательных систем, но наиболее часто используемыми являются подходы коллаборативной фильтрации и контентной фильтрации.

Рассмотрим детально каждый из них:

Коллаборативная фильтрация осуществляет выдачу рекомендаций на основании модели предшествующего поведения пользователя. Данная модель может строиться исключительно на поведении данного конкретного пользователя или, что более эффективно, с учетом поведения других пользователей со сходными характеристиками [5].

В случаях, когда коллаборативная фильтрация принимает во внимание не только поведение одного конкретного пользователя, но и поведение других пользователей, она использует **знание о группе** для выработки рекомендаций на основе подобия пользователей.

Для детальности рассмотрим пример:

Таблица 1.1 – Пример коллаборативной фильтрации

	User1	User2	User3	User4	User5
Книга 1	5	-	4	4	3
Книга 2	3	-	-	-	5
Книга 3	-	3	4	4	5
Книга 4	3	3	2	-	1
Книга 5	5	4	1	-	3

В таблице 1.1 есть 5 пользователей (User1, User2, …, User5) и 5 книг (Книга 1, Книга 2, …, Книга 5), которые могут понравиться или не понравиться данным пользователям. Оценки пользователя могут быть в пределах од 1 до 5, или не читал (-).

Смысл работы данной фильтрации заключается в объединении пользователей с одинаковыми предпочтениями в группы, для определения потребностей других пользователей. На данном примере пользователей можно группировать, как Группа 1 - User1 и User2, а так же Группа 2 - User3, User4 и User5. Группы выделены разными цветами.

На основании данных занесенных в таблицу можно предположить, что User2 предположительно понравиться Книга 1, так как она понравилась User1, а User3 и User4 может заинтересовать Книга 2, так как она понравилась User5. А User4 может не понравиться Книга 4 и Книга 5.

Разумеется, данный пример полностью не отражает эффективность данного метода построения рекомендательных систем из-за малого количества пользователей (малой выборки), но, тем не менее, дает общее представление о принципе работы систем, основанных на коллаборативной фильтрации.

Главный плюс данного подхода состоит в высокой точности, при достаточном размере выборки.

К недостаткам данного подхода можно отнести «холодный старт» - случай, когда товары или услуги, недавно появившиеся, не имеют возможности попасть в рекомендации какому-либо пользователю, а так же высокие затраты программных ресурсов на построение прогнозов, при большом количестве товаров или услуг, т.е. плохо поддаются масштабируемости.

Контентная фильтрация формирует рекомендацию на основе поведения пользователя. Другими словами данный способ подразумевает собой анализ поведения данного конкретного пользователя. Например, этот подход может использовать ретроспективную информацию о просмотрах (какие книги читает пользователь и характеристики этих книг) и на основании всего этого делает прогноз того, какие продукты могут заинтересовать этого пользователя [5].

Данный способ менее требователен к железу серверов, т.к. он не требует столь больших затрат программных ресурсов на построение и анализ модели.

Рассмотрим детальнее данный способ на примере:

Есть 5 фильмов с разными жанрами:

Фильм 1 – фэнтези, комедия, романтика;

Фильм 2 – комедия, фантастика, драма;

Фильм 3 – боевик, ужасы;

Фильм 4 – детектив, комедия;

Фильм 5 – боевик, ужасы, комедия.

Пусть пользователю понравились фильмы 1 и 2. Фильмы 3,4 – не смотрел. Фильм 5 – не понравился.

Для наглядности запишем данные примера в виде таблицы:

Таблица 1.2 – Пример контентнной фильтрации

Фильм 1	**Фильм 2**	**Фильм 3**	**Фильм 4**	**Фильм 5**
+	+	?(-)	?(+)	-

Из Фильма 1 и Фильма 2 можно предположить, что пользователю нравится жанр комедия, т.к. это единственный жанр, который совпадает в обоих фильмах, а следовательно, можно предположить, что ему так же может понравиться Фильм 4.

Из Фильма 5 можно предположить, что пользователю могут, не нравится один или оба из жанров: боевик, ужасы, а, следовательно, возможно ему также не понравиться Фильм 3.

Таким образом, можно сказать, что контентная фильтрация более экономна с точки зрения программных ресурсов, однако обладает меньшей точностью по сравнению с коллаборативной фильтрацией.

Также помимо этих двух наиболее часто используемых методов построения рекомендательных систем широкой популярностью пользуется г**ибридный подход**. Смысл данного подхода заключается в совмещении плюсов как коллаборативной, так и контентной фильтраций для построения рекомендаций.

Для наглядности преимущества и недостатки двух основных подходов запишем в таблицу 1.3.

Таблица 1.3 – Преимущества и недостатки основных подходов построения рекомендательных систем

Коллаборативная фильтрация	**Контентная фильтрация**	**Гибридный подход**
+ высокая точность.	+ низкое потребление программных ресурсов; + быстрота работы; + отсутствие «холодного старта».	+ высокая точность; + отсутствие «холодного старта»; + масштабируемость.
- плохо поддаются масштабируемости; - «холодный старт».	- низкая точность.	

Как видно из таблицы 1.3 общее количество преимуществ у контентной фильтрации больше, а недостатков меньше, чем у коллаборатикной фильтрации. Однако в противовес контентная фильтрация имеет существенный недостаток в виде существенно более низкой точности, что является основным критерием выбора рекомендательных систем для Интернет-магазинов. В то же время гибридный подход подразумевает стремление к избеганию недостатков 2-х основных подходов или повышение точности за счет более глубокого анализа товаров и предпочтений пользователей.

1.2. Анализ существующих реализаций рекомендательных систем

На данный момент основные формы готовых решений рекомендательных систем можно условно разделить на 2 типа:

- сервисный тип рекомендательных систем;
- библиотечный тип рекомендательных систем;

1.2.1. Сервисные рекомендательные системы

К сервисному типу рекомендательных систем можно отнести следующие сервисы: Retail Rocket, Crossss, Rees46 и другие…

К преимуществам данных систем можно отнести более широкий набор функциональных особенностей, например, помимо простого построения рекомендаций, может присутствовать встроенная e-mail /sms / или любая другая рассылка, а также.

К недостаткам можно отнести закрытость данных о рекомендательной системе. Другими словами владельцам сайтов неизвестно, каким образом работает рекомендательная система, а следовательно не может как-либо влиять на принцип работы рекомендательной системы самостоятельно.

Для примера детальнее рассмотрим два наиболее популярных в России сервиса: Retail Rocket и Rees46.

Retail Rocket. Данный сервис является наиболее популярным в России сервисом рекомендаций и используется такими компаниями, как Сбербанк, Auchan, Эльдорадо, Детский мир, Hoff и другие…[6].

Как и многие другие сервисные рекомендательные системы, функционал и другие особенности меняются в зависимости от используемого тарифа. Тарифы и их особенности представлены в таблице 1.4 [7].

Таблица 1.4 – Тарифы Retail Rocket

БЕСПЛАТНЫЙ ТАРИФ	НЕБОЛЬШОЙ МАГАЗИН	СРЕДНИЙ МАГАЗИН	ОГРОМНЫЙ МАГАЗИН
0 руб.	1700 руб/мес	7100 руб/мес	по запросу
До 250 заказов / мес.	До 250 заказов / мес.	До 500 заказов / мес.	От 500 заказов / мес.
До 1 500 товаров в базе	До 10 000 товаров в базе	До 30 000 товаров в базе	От 15 000 товаров в базе
До 10 000 посетителей / мес.	До 25 000 посетителей / мес.	До 50 000 посетителей / мес.	От 50 000 посетителей / мес.
Базовая статистика	Расширенная статистика	Расширенная статистика	Расширенная статистика
Тех. поддержка по email	Приоритетная тех. поддержка	Приоритетная тех. поддержка	Персональный менеджер
16 видов рекомендаций	16 видов рекомендаций	16 видов рекомендаций	Доработки по запросу
«Брошенная корзина»	Триггерные рассылки	Триггерные рассылки	Триггерные рассылки

Рассмотрим преимущества данного сервиса.

Исходя из таблицы 1.4, а также информации найденной на сайте сервиса, к преимуществам сервиса можно отнести:

+ присутствие бесплатного тарифа;
+ триггерные рассылки, в том числе и email;
+ множество видов рекомендаций;
+ персонализированная доработка рекомендаций для тарифа **огромный магазин**;
+ присутствие статистики эффективности работы;
+ присутствие триал-версии на 30 дней для тарифов: **небольшой магазин** и **средний магазин**[7];
+ присутствие платформы для массовых email-рассылок[8];
+ лёгкая интеграция с сайтом через трекинг-коды[8].

К недостаткам можно отнести:

– отсутствие информации об эффективности изначальных алгоритмов рекомендаций;
– отсутствие возможности вносить изменения в существующих алгоритмах;
– зависимость тарифа от размера магазина (количества товаров, заказов и посетителей в месяц).

Таким образом, данный сервис имеет достаточно большой ряд преимуществ, но в то же время имеются недостатки, связанные с размером магазина и закрытостью кода алгоритмов.

Rees46. Как и Retail Rocket у данного сервиса есть несколько тарифов, строящихся на основании количества посещений в месяц:

- Тариф M (До 25.000/мес) – 5000Р;
- Тариф L (От 20К до 50К/мес) – 10000Р;
- Тариф XL (От 50К/мес) – Индивидуально.

В отличие от Retail Rocket дополнительный функционал в зависимости от тарифа не меняется (исключение - тариф XL).

К списку дополнительных возможностей данного сервиса относятся: триггерные цепочки, Email-рассылки, Web/mobile push уведомления, товарная аналитика, динамическая сегментация и другие…[9].

Тарифа XL дополнительно имеет персонального менеджера и приоритетную техподдержку.

К преимуществам данного сервиса можно отнести:

- большое количество дополнительного функционала[9];
- присутствие триал-версии на 14 дней/

К недостаткам можно отнести:

- отсутствие бесплатного тарифа;
- отсутствие информации об эффективности изначальных алгоритмов рекомендаций;
- отсутствие возможности вносить изменения в существующих алгоритмах;
- зависимость тарифа от размера магазина (количества товаров, заказов и посетителей в месяц).

Таким образом, можно сделать вывод, что сервисные рекомендательные системы отличаются легкой интегрируемостью и большим дополнительным функционалом, В противовес имеют и ряд недостатков, а именно: отсутствие

информации об используемых алгоритмах, невозможностью вмешиваться в работу алгоритма рекомендаций.

1.2.2. Библиотечные рекомендательные системы.

Данные системы представляют собой рекомендательные системы в виде библиотеки, зачастую библиотеки классов, и предоставляющие функциональные особенности в виде методов и классов, направленных для написания рекомендательных систем.

В отличие от рекомендательных систем сервисного типа зачастую обладают следующими преимуществами:

- открытостью кода;
- возможностью самостоятельной модификации кода;
- повышенной защищенностью, т.к. все вычисления производятся на стороне сайта;
- отсутствием абонентской платы;
- универсальностью.

Помимо преимуществ существует и определенный ряд недостатков:

- повышенная нагрузка на рабочую машину;
- отсутствие дополнительных функций (например: email-рассылки).

Рассмотрим некоторые из рекомендательных систем данного вида:

1. LensKit. Представляет собой API для создания рекомендательных алгоритмов, включающей инструменты для создания рекомендаций (оценки потребностей пользователей) [10].

Основным типом алгоритмов, используемых в данной библиотеке, является коллаборативная фильтрация. Помимо коллаборативной фильтрации также присутствуют алгоритмы типа: «Топ N», например «10 наиболее продаваемых товаров».

На данный момент существует две версии LensKit, написанные на 2-х разных языках программирования – Java и Python.

В качестве основного достоинства LensKit можно отнести возможность написания собственных алгоритмов. В качестве интерфейсов присутствует возможность написания алгоритмов «Топ N» и алгоритмов коллаборативной фильтрации.

Основным недостатком является отсутствие интерфейса под контентную фильтрацию и под смешанный тип рекомендательных систем.

2. MyMedia Lite [11]. Также как и LensKit представляет собой библиотеку для рекомендательной системы. Также как и LensKit поддерживает возможность модификации алгоритмов. К основному недостатку, также, как и у LensKit отсутствие поддержки алгоритмов, основанных на контентной фильтрации.

К основным особенностям данной рекомендательной системы можно отнести:

- не нуждается в базе данных;
- содержит базовые типы рекомендательных алгоритмов: предсказание рейтинга и товара;
- не содержит сложных функций вроде комплексных рекомендаций, потоков рекомендаций, интеграции с OpenID.

Также MyMedia Lite поддерживает несколько языков программирования, таких, как: C#, Clouri и F#.

Для наглядности запишем описанные ранее различия в виде таблицы:

Таблица 1.5 – Сравнение аналогов

	retailrocket	REES46 TECHNOLOGIES	LensKit	MyMedia Lite
Поддержка модификации	+	-	+	+
Коллаборативная фильтрация	?	?	+	+
Контентная фильтрация	?	?	-	-
Смешанный тип	?	?	-	-
Открытость кода	-	-	+	+
Стоимость	Есть бесплатный тариф	Есть триал-версия	Бесплатно	Бесплатно
Простота	+	+	-	+

1.3. Выбор средства разработки рекомендательной системы

Так как поставленная в данной выпускной квалификационной работе задача представляет собой разработку рекомендательной системы, которая будет поддерживаться для решения максимально большого количества различных задач, а, следовательно, не должна зависеть от среды размещения, то логично

предположить, что в качестве основного средства для разработки необходимо выбрать какой-либо из языков программирования.

Так как разрабатываемый программный продукт будет являться библиотекой, позволяющей на её основе создавать новые вариации алгоритмов рекомендательных систем, то логично предположить, что необходимо выбирать из таких языков программирования, которые поддерживают принципы ООП (объектно-ориентированного программирования), а в частности наиболее интересен такой механизм ООП, как наследование.

Наследование – механизм ООП, позволяющий при описании нового класса наследовать поля и методы класса-родителя.

Таким образом, можно сделать вывод, что основным средством разработки необходимо выбирать язык программирования, поддерживающий ООП. В качестве возможных вариантов рассмотрим следующие популярные языки программирования: Java, C++, C#.

- **Java** – ООП язык программирования, разработанный компанией Sun Microsystems. Первая версия Java вышла 23 мая 1995 года. Ключевой особенностью Java, для решения поставленной задачи, является трансляция кода написанного на Java в специальный байт-код, который затем может быть исполнен на любой машине, где установлена JVM (виртуальная машина Java).

JVM представляет собой основу исполняющей системы Java – JRE (Java Runtime Environment), которая исполняет байт-код Java, созданный на этапе компиляции из кода написанного на ЯП Java.

Преимущество подобного подхода состоит в том, что оно позволяет избавиться от зависимости исполняемой платформы от написанного кода, повышенная защищенность кода от ошибок, например виртуальная машина Java поддерживает автоматическую очистку памяти и тем самым вероятность переполнения памяти, из-за того что разработчик забыл очистить память, значительно снижается.

Принцип работы Java-программ изображен на рисунке 1.1.

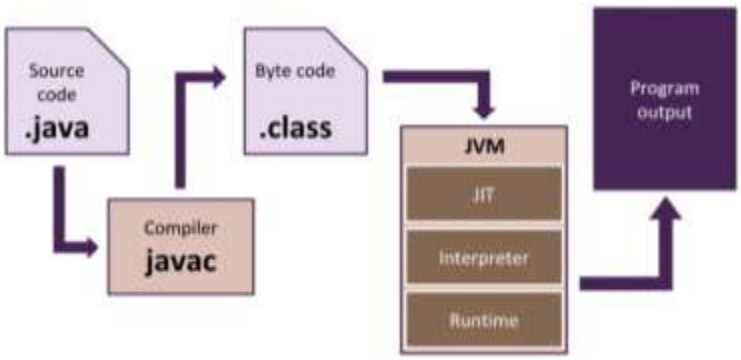

Рисунок 1.1 – Принцип работы Java-программ

Главным недостатком подобного подхода является значительно меньшая скорость работы в сравнении языками программирования, которые компилируют код напрямую в машинные команды.

∘ Далее рассмотрим ЯП **C++**.

В отличие от Java данный язык программирования не имеет среды исполнения кода, как JVM и таким образом исполнение кода происходит непосредственно на машине. Подобный подход отличается повышенной скоростью работы, но меньшей защищенностью и зависимостью итогового кода от среды компиляции. Это означает, что если код, написанный на Java компилируется один раз в байт-код, а затем этот байт-код исполняется в JVM, то код, написанный на C++ для каждой платформы компилируется по разному и исполнение кода происходит непосредственно на процессоре.

∘ В завершение рассмотрим ЯП **C#** и платформу для разработки .NET Core.

Данный язык программирования по принципу исполнения кода очень похож на ЯП Java. Как и на Java, код, написанный на языке программирования C#, сначала компилируется в код ассемблера CIL (Common Intermediate Language), аналог байт-кода Java, который затем исполняется в общеязыковой среде исполнения CLR (Common Language Runtime).

Одной из ключевых особенностей .NET является возможность написания кода на нескольких языках программирования, которые поддерживаются данной платформой.

Процесс компиляции и выполнения кода на платформе .NET представлен на рисунке 1.2.

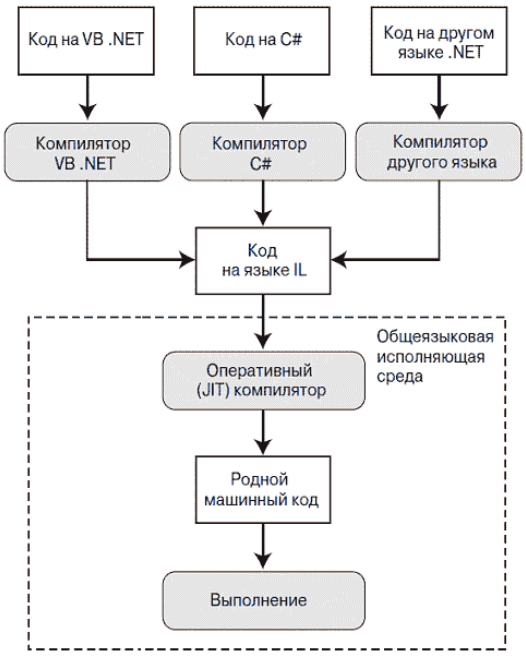

Рисунок 1.2 - Компиляция исходного кода на платформе .NET

В 2016 году вышла первая версия .NET Core, которая позволила исполнять написанный на данной платформе код на таких операционных системах Windows, MacOS и Linux операционных системах, тем самым убирая одно из основных отличий данных средств разработки.

.NET Core, как и Java обладает теми же преимуществами и недостатками, описанными ранее.

Таким образом, исходя из целей поставленной задачи можно сделать вывод, что в качестве ЯП для разработки наиболее подходящими будут являться языки программирования C# и Java, так как, данные языки программирования, в отличие от языка программирования C++ более безопасные в плане написания кода. Так как, последние версии данных языков программирования и их среды для решения поставленной задачи практически идентичны, то выбор отдается, основываясь на удобстве написания кода.

Таким образом, языком программирования для разработки рекомендательной системы с настраиваемым блоком анализа данных был выбран язык программирования C#, а, следовательно .NET Core. Средой разработки будет являться Visual Studio 2017 так, как в ней присутствует весь функционал, который необходим для разрабатываемого программного продукта, а также имеет удобный для пользователя интерфейс.

1.4. Вывод

Исходя из анализа существующих реализаций рекомендательных систем и средств разработки, в качестве основного средства разработки был выбран язык программирования C#, платформа .NET Core и среда разработки Visual Studio. Возможностей данных компонентов достаточно для создания и тестирования разрабатываемого программного продукта.

Для разрабатываемого тестового алгоритма для разрабатываемой рекомендательной системы выберем подход коллаборативной фильтрации так, как он обеспечивает более высокую точность анализа данных.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Для того чтобы перейти к этапу проектирования будущего программного продукта необходимо сформулировать набор требований, которые должны быть реализованы в итоговом программном продукте.

2.1. Функциональные требования

К основным функциональным требованиям разрабатываемой рекомендательной системы относятся:

- рекомендательная система должна быть способна выдавать рекомендации пользователям на основании каких-либо данных и требований;
- рекомендательная система должна поддерживать возможность полной замены используемого алгоритма прогноза рекомендаций;
- рекомендательная система должна поддерживать возможность создания новых алгоритмов рекомендаций без изменения базовой структуры программы;
- рекомендательная система должна обладать возможностью хранить, сохранять и загружать данные, которые были получены в результате анализа данных.

Графически, описанные функциональные требования можно описать в виде диаграммы UML. Диаграмма вариантов использования системы представлена на рисунке 2.1.

Рисунок 2.1 – Диаграмма вариантов использования системы

Из рисунка 2.1 видно, что в данной системе существует 4 категории пользователей, обладающих определенным функционалом:

Во-первых – пользователь, основным функционалом, которого является получение рекомендации;

Во-вторых – разработчик, который может разрабатывать новые варианты рекомендательной системы (алгоритма анализа данных и алгоритма хранения данных);

В-третьих – владелец сайта, который может настраивать рекомендательную систему.

В-четвертых – рекомендательная система, которая может загружать, сохранять и анализировать данные.

2.2. Нефункциональные требования

Основным нефункциональным требованием разрабатываемой рекомендательной системы является:

- рекомендательная система с настраиваемым блоком анализа данных должна быть разработана в виде библиотеки классов;
- должна быть реализована, как минимум одна реализация рекомендательного алгоритма.

3. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

Проектирование – процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части. Результатом проектирования является проект – целостная совокупность моделей, свойств или характеристик, описанных в форме, пригодной для последующей реализации.

3.1. Архитектура предлагаемого решения

Архитектурно рекомендательная система с настраиваемым блоком анализа данных представляет собой совокупность нескольких классов и разделена на 3 основных блока.

К трём основным блокам относятся:

- основной блок программы, содержащий основной функционал РС (входы и выходы данных, установка используемого в блоке анализа данных алгоритма);
- блок анализа данных, представляющий собой логику построения прогнозов потребностей пользователей (представляет собой абстрактный класс);
- блок хранения данных, осуществляющий хранение данных в удобной для использования форме.

Помимо этих 3-х основных классов-блоков присутствует вспомогательные классы «Данные» и «Критерии».

Интерфейс «Данные» позволяет разработчикам алгоритмов отбирать только такие данные, которые необходимые для реализации их алгоритмов.

Интерфейс «Критерии» позволяет разработчикам алгоритмов в некоторых методах (например, метод загрузки данных) передавать необходимые условия критериев выбора данных (например, номер группы, который необходимо получить).

Графическое отображение архитектуры программного продукта отражено на рисунке 3.1.

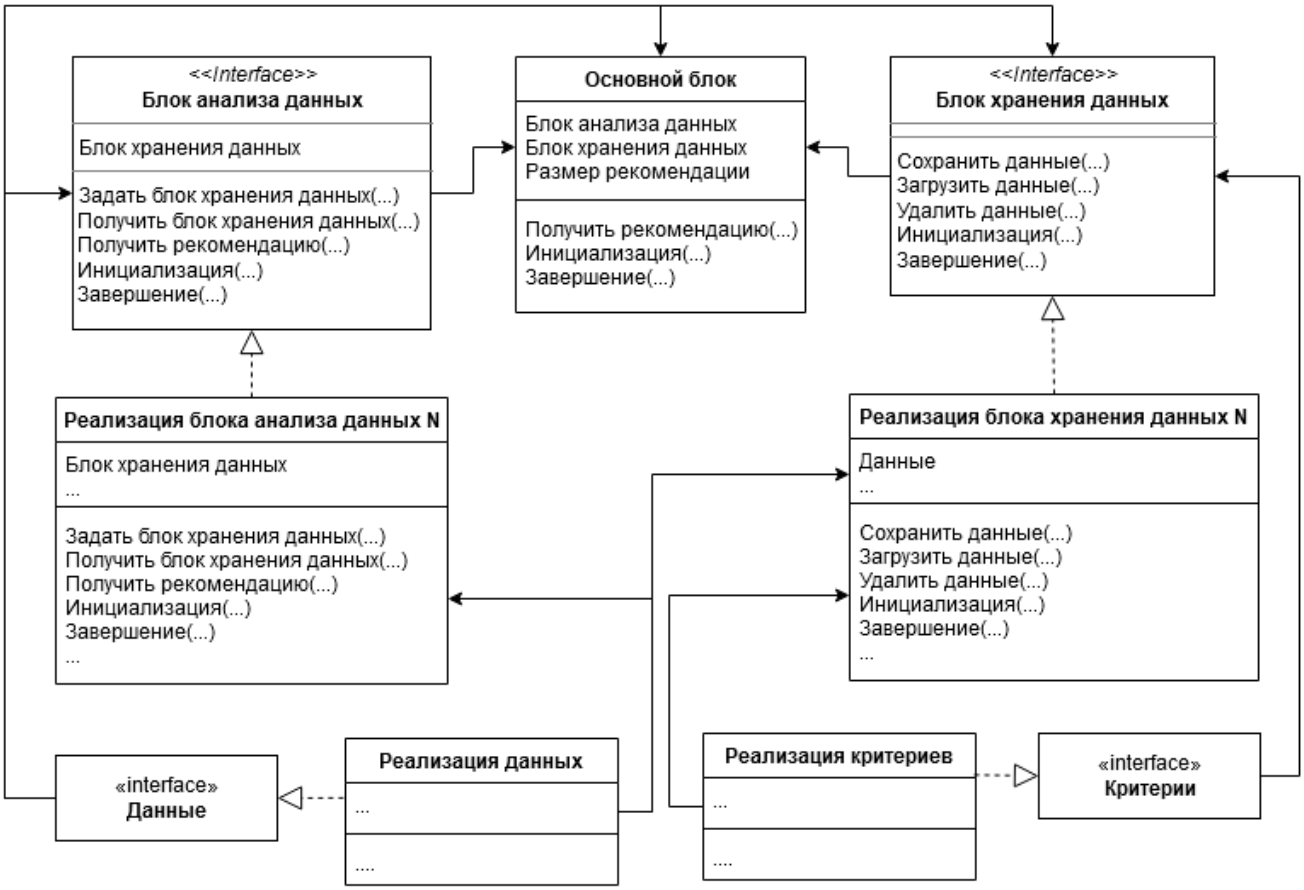

Рисунок 3.1 – Архитектура рекомендательной системы с настраиваемым блоком анализа данных

3.2. Алгоритм блока анализа данных

При реализации алгоритма выдачи рекомендации был разработан алгоритм, основанный на подходе коллаборативной фильтрации.

Данный алгоритм можно разбить на несколько шагов:

1. Проверка на наличие содержимого (групп пользователей) в хранилище данных и при необходимости создание новой группы на основе первого пользователя;

2. Расчет «отклонения» содержимого товаров пользователя относительно товаров групп производится с помощью формулы расстояния Хэмминга, которая выглядит следующим образом [12]:

$$d_{ij} = \sum_{k=1}^{p} \left| x_{ik} - x_{jk} \right|$$

где p, q – точки сравнения, в данном случае оценки товаров.

3. Нахождение минимального значения $d(p,q)$ и если это значение меньше определенного числа (в данном случае 50), то создается новая группа пользователей на основе текущего пользователя;

4. Модификация наиболее значимой группы пользователей в соответствии с предпочтениями текущего пользователя;

5. Выдача рекомендации пользователю в виде списка наиболее значимых (с наибольшими значениями переменной *rating*) товаров размера recSize, которых нет у данного пользователя.

Графически данные шаги представлены на рисунке 3.2 в виде блок-схемы.

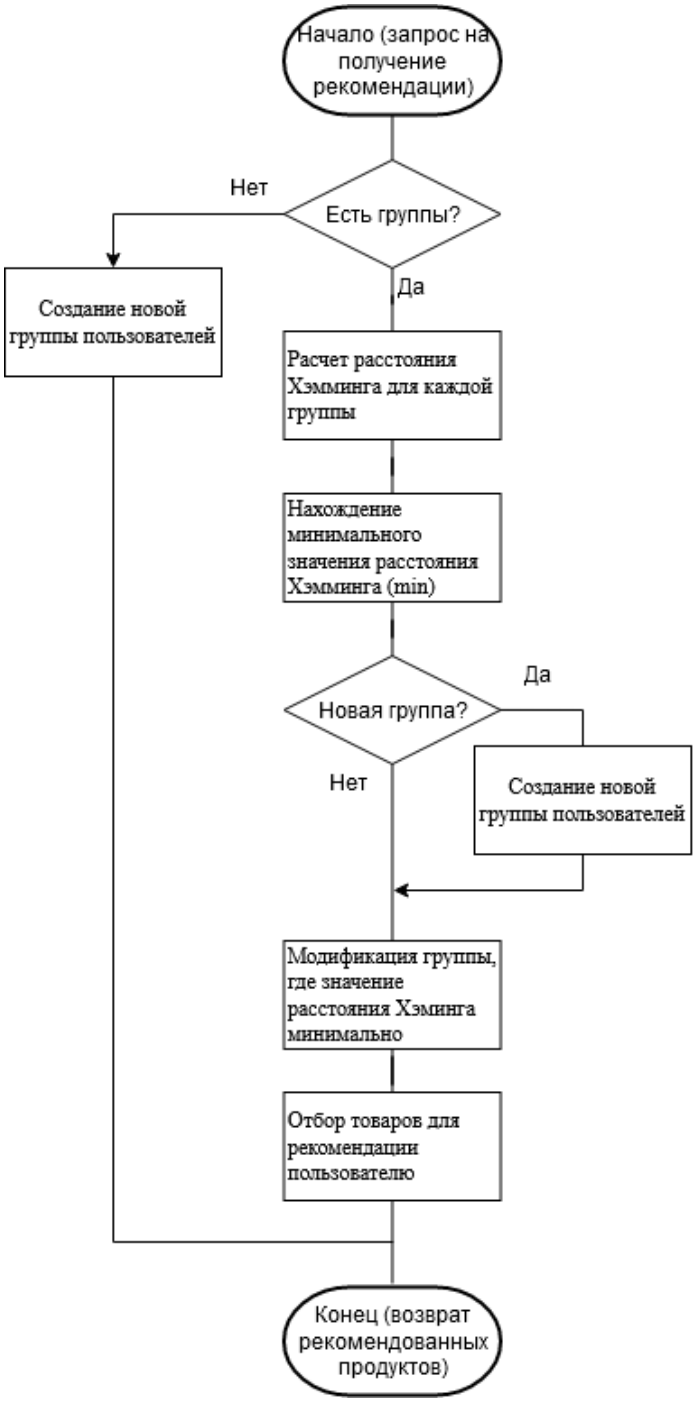

Рисунок 3.2 – Блок-схема алгоритма блока анализа данных

3.3. Реализация алгоритма модификации групп пользователей

Важным аспектом, при правильном проектировании рекомендательной системы является способность рекомендательной системы к обучению.

В данном случае обучение осуществляется путём модификации групп пользователей, которые получаются в результате работы алгоритма.

Для решения данной задачи был реализован простой алгоритм, который решает две основные задачи:

1. Обновляет рейтинг продуктов, которые уже ранее были добавлены в данную группу;
2. Добавляет новые продукты, которых ранее не было в данной группе.

Данный алгоритм осуществляется путем перебора всех продуктов пользователя и сравнения их с продуктами группы по идентификатору.

Если продукт, который есть у пользователя, есть у группы, то обновляется его рейтинг в соответствии оценкой пользователя.

Если при переборе, продукт, который есть у пользователя, не был найден в результате перебора, то его добавляют в группу пользователей.

Схематически данные шаги отображены на рисунке 3.3 в виде блок-схемы.

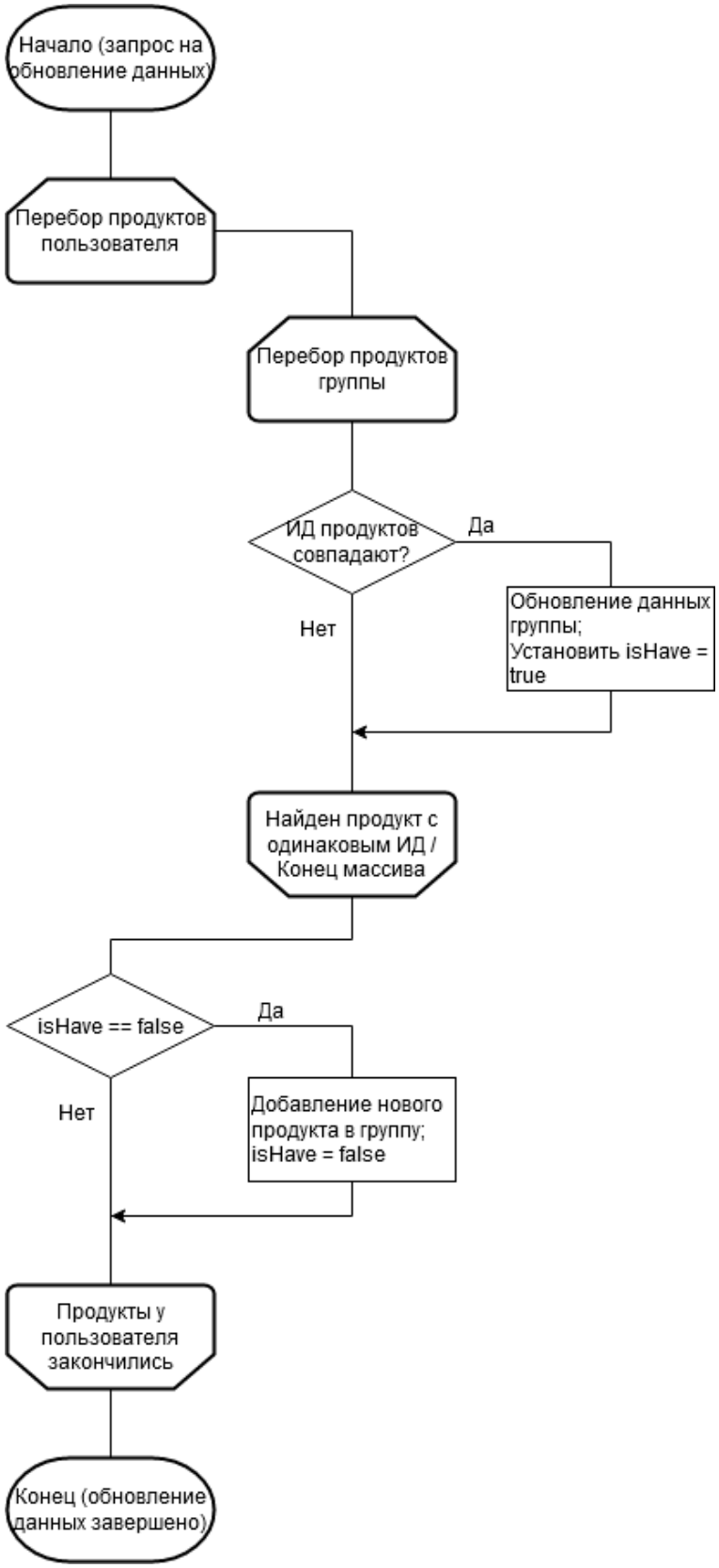

Рисунок 3.3 – Алгоритм добавления обновления данных групп

3.4. Реализация хранилища данных

Так, как рекомендательные системы обычно работают с большим объемом данных, то логично предположить, что при выборе хранилища данных в первую очередь стоит опираться на скорость работы. Для быстрой работы с большим количеством данных лучше всего подходят NoSQL базы данных

NoSQL – это подход к реализации масштабируемого хранилища (базы) информации с гибкой моделью данных, отличающийся от классических реляционных СУБД. В нереляционных базах проблемы масштабируемости (scalability) и доступности (availability), важные для Big Data, решаются за счёт атомарности (atomicity) и согласованности данных (consistency).

На текущий момент существует огромное множество NoSQL баз данных, таких, как MongoDB, CouchDB, GemFire, LiteDB, и другие…

В качестве подобной базы данных, для данного проекта остановимся на NoSQL базе данных LiteDB [13].

Данная база данных выбрана по нескольким причинам:

1. отсутствует необходимость предустановки внешних баз данных;
2. высокая скорость работы;
3. лёгкая в использовании;
4. малый вес файлов.

Для управления данной базой данных был реализован отдельный класс – **WorkDatabase**, который содержит в себе следующие методы:

Таблица 3.1 – Методы управления базой данных

Название метода	**Описание**
ClearDB	Очищает базу данных.
AddGroup	Добавляет новую группу.
UpdateGroup	Обновляет содержимое определенной группы.
LoadGroupData	Загружает группы данные из базы данных.

Описанный в таблице 3.1 функционал достаточен для корректной работы базы данных и рекомендательной системы.

Содержимое данного класса приведено в листинге Б.4 приложения Б.

3.5. Описание основных классов

В таблице 3.2 приведены классы основной структуры рекомендательной системы с настраиваемым блоком анализа данных.

Таблица 3.2 – Основные классы структуры рекомендательной системы

Название класса	**Тип класс**	**Описание**
RecCore	Класс	Является основным классом, с которым у пользователя происходит взаимодействие для получения рекомендации.
RecommendAlgorithm	Абстрактный класс	Содержит абстрактные методы, которые могут понадобиться для работы с анализом данных и выдачей результатов
DataContainer	Абстрактный класс	Содержит абстрактные методы, которые могут понадобиться для работы с хранилищами данных.
IModifiedData	Пустой интерфейс	Реализации интерфейса используются для установки на вход рекомендательной системы данных любого вида.
ICriteriaData	Пустой интерфейс	Реализации интерфейса используются для задания каких-либо пользовательских условий в некоторых методах.

Классы, содержащиеся в таблице 3.2, расположены в пространстве имен RecSystem, содержимое которого приведено в листингах А.1 – А.5 приложения А. Описание методов, содержащихся в данных классах, отражено в таблице 3.3.

Таблица 3.3 – Описание методов основных классов пространства имен RecSystem

Имя метода	**Класс**	**Описание**
GetRecommendation	RecCore	Возвращает пользователю рекомендацию на основе данных пользователя и ранее проанализированных данных.

Продолжение таблицы 3.3

End	RecCore	Выполняет методы End из реализаций абстрактных классов DataContainer и RecommendAlgorithm.
RecCore	RecCore	Конструктор. Принимает на вход реализации абстрактных классов DataContainer и RecommendAlgorithm и размер требуемой рекомендации.
SetDataContainer	RecommendAlgorithm	Устанавливает значение переменной dataContainer.
GetDataContainer	RecommendAlgorithm	Получает содержимое переменной dataContainer.
GetRecommendation	RecommendAlgorithm	Возвращает рекомендацию пользователю. Выполняет анализ данных
Init	RecommendAlgorithm	Метод инициализации.
End	RecommendAlgorithm	Метод завершения работы. Выполняется через метод End класса RecCore.
SaveData	DataContainer	Выполняет сохранение данных в хранилище данных.
RemoveData	DataContainer	Удаляет данные из хранилища данных на основании каких-либо условий.
LoadData	DataContainer	Возвращает данные из хранилища данных на основании каких-либо условий.
Init	DataContainer	Метод инициализации.
End	DataContainer	Метод завершения работы. Выполняется через метод End класса RecCore.

В качестве вспомогательного инструмента реализован класс Tools. Полное описание методов класса представлено на листинге Б.3 приложения Б.

В таблице 3.4 представлено описание методов, реализованных в данном классе.

Таблица 3.4 – Описание методов класса Tools

Имя метода	**Описание**
EqualCoef	Рассчитывает значимость группы пользователей на конкретного пользователя, с помощью формулы расстояния Хэмминга.
RMSE	Рассчитывает значение RMSE (Root Mean Square Error) для оценки эффективности алгоритма.
GetUserInfo	Осуществляет парсинг тестового документа.
ResultRating	Рассчитывает итоговый рейтинг продуктов в группах.

Метод, рассчитывающий расстояния Хэмминга, приведен на листинге 3.1.

Листинг 3.1 – Реализация метода EqualCoef

```
public static double EqualCoef(GroupInfo group, DataExample user)
{
     int n = 0;
     double score = 0;
     for (int i = 0; i < user.prodID.Length; i++)
         for (int j = 0; j < group.product.Length; j++)
             if (group.product[j].Id == user.prodID[i])
             {
                 score += Math.Abs(user.score[i] - group.product[j].rating);
                 n++;
             }
     score /= (double)n / user.score.Length;
     return score;
}
```

Из листинга 3.1 видно, что метод принимает на вход две переменные классов **GroupInfo** и **DataExample**. В качестве элементов сравнения выступают 2 значения следующих переменных **user.score[i]** и **group.product[j].rating**.

Переменная **user.score[i]** представляет собой значение оценки, которую пользователь с индексом [i] выдал [j]-му продукту из **group.product[j]**.

Переменная **group.product[j].rating** представляет собой значение рейтинга (веса товара в данной группе), который был получен в результате анализа данных ранее.

Для учета процента количества совпавших в рассматриваемой группе и у пользователя продуктов в конце итоговое значение расстояния Хэмминга делится на `(double)n / user.score.Length`, где n – количество совпавших товаров, user.score.Length – общее количество товаров у пользователя.

В качестве проверки работоспособности алгоритма выведем в консоль, какие рекомендации получают пользователи в виде:

ИД_Пользователя | ИД_Группы | Список рекомендаций |

Размер списка рекомендаций примем равным 3.

Результат запроса на выдачу рекомендаций представлен на рисунке 3.1.

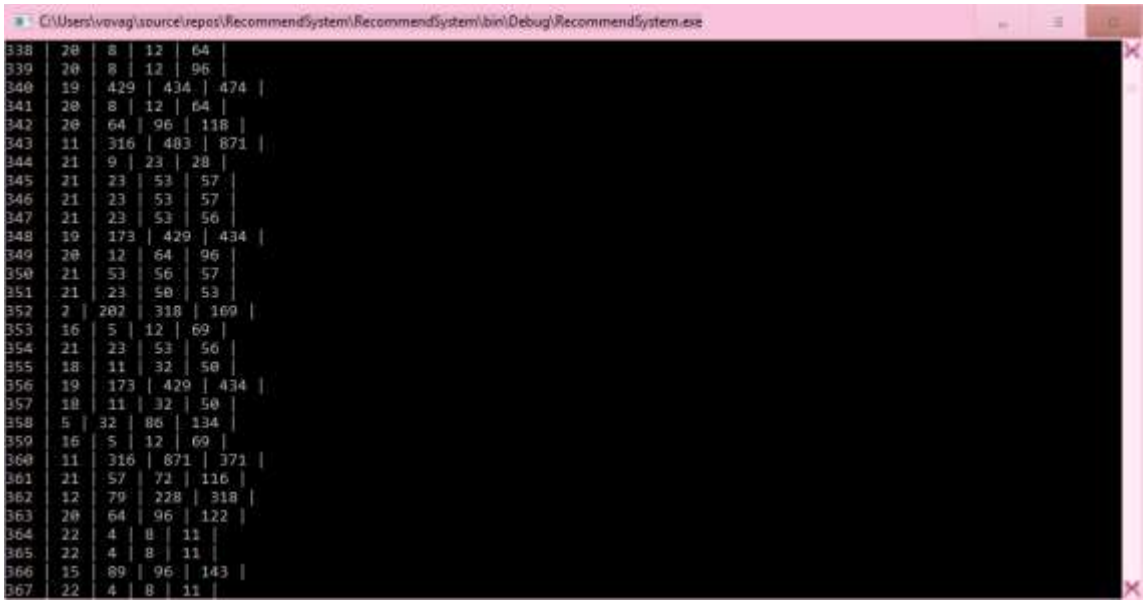

Рисунок 3.1 – Результат выполнения команды на получение рекомендации некоторыми пользователями

Как видно из рисунка пользователей с идентификаторами от 338 до 367 алгоритм рекомендаций приписывает к разным группам и выдает разные рекомендации.

Таким образом, можно сделать вывод, что реализованный программный продукт работает правильно. Детальная проверка эффективности реализованного алгоритма приведена в разделе 4.

3.6. Вывод

В ходе проектирования программной системы были определены и описаны основные требования к разрабатываемой рекомендательной системе

В данном разделе были рассмотрены основные алгоритмы, используемые для реализации важных для работы рекомендательной системы элементов. Таких, как алгоритм получения рекомендаций, реализованный в блоке анализа данных, и алгоритм модификации групп пользователей, а также описание основных классов и их методов в виде таблиц классов и методов.

Также в данном разделе было выбрано основное средство для хранения данных, которое реализовано в виде блока хранения данных. Им является NoSQL база данных LiteSQL.

В ходе работы по реализации был создан программный продукт, который реализует все основные требования, которые были указаны в разделе 2.

Для проверки корректности и эффективности разработанного программного продукта необходимо произвести его тестирование.

4. ТЕСТИРОВАНИЕ

Для проверки работоспособности программного продукта, а также для проверки эффективности предлагаемого в решении тестового алгоритма произведем два варианта тестирования на тестовых данных.

Тестовые данные взяты из источника: https://grouplens.org/datasets/movielens/100k/, представленного веб-системой рекомендаций фильмов MovieLens.

Для тестов используются файлы uN.base и uN.test, входящие в состав, представленных MovieLens данных.

uN.base – тестовые данные для первичного обучения созданного алгоритма.

uN.test – тестовые данные, используемые для проверки эффективности алгоритма.

Данные в этих файлах представлены в виде таблице со столбцами:

user id | item id | rating | timestamp

Столбец **timestamp** в разработанном алгоритме не учувствует.

В первом тестировании проверим насколько качественно, разработанный алгоритм, выдает рекомендации пользователям.

Для этого воспользуемся методикой RMSE (Root Mean Square Error) – корень среднеквадратической ошибки регрессии, формула которой выглядит следующим образом [14]:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}{n}}$$

где

$y_i - \hat{y}_i$ – остаток регрессии (разности между наблюдаемыми значениями и значениями, предсказанными изучаемой регрессионной моделью);

y_i – наблюдаемое значение;

$\hat{y}_i$ – предсказанное значение;

n – суммарное число тестов (значений).

Тестирование проведем на 5 различных выборках, предварительно перед каждым тестом будем отчищать хранилище данных от старой выборки.

В результате проведённое тестирование на 5 тестовых файлах поочередно выявило значения RMSE, записанные в таблице 4.1. Значение n во всех тестовых файлах равно 20000.

Таблица 4.1 – Значения RMSE для 5 тестов

№ теста	**Значение RMSE**
1	0.94327323
2	0.96181821
3	0.94569236
4	0.96917385
5	0.94019237

Далее проверим, как быстро разработанный алгоритм может выдавать рекомендации.

Для этого поочередно выполни функцию получения рекомендации на 5 файлах «uN.base.txt», где N – номер файла. В данном случае очистка хранилища данных производиться не будет. Каждый файл содержит по 80000 записей.

Для точной оценки времени воспользуемся классом Stopwatch[15].

Результат тестирования представлен на рисунке 4.1 в виде вывода данных времени в консоль в формате «**часы : минуты : секунды . миллисекунды / 10**».

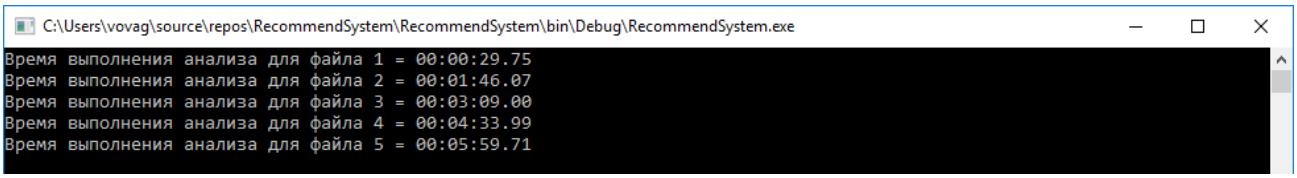

Рисунок 4.1 – Результат тестирования алгоритма на скорость работы

Для наглядности запишем полученные результаты в виде таблицы:

Таблица 4.2 – Затраты времени на проведение тестирования

№ файла	Общее время	Время на файл, сек.	Время на ед. товара, мс.
1	00:00:29.75	29.75	0.371875
2	00:01:46.07	76.32	0.954
3	00:03:09.00	82.93	1.036625
4	00:04:33.99	84.99	1.062375
5	00:05:59.75	85.76	1.072

Для наглядности по третьему столбцу таблицы составим график зависимости числа тестов на время выполнения анализа:

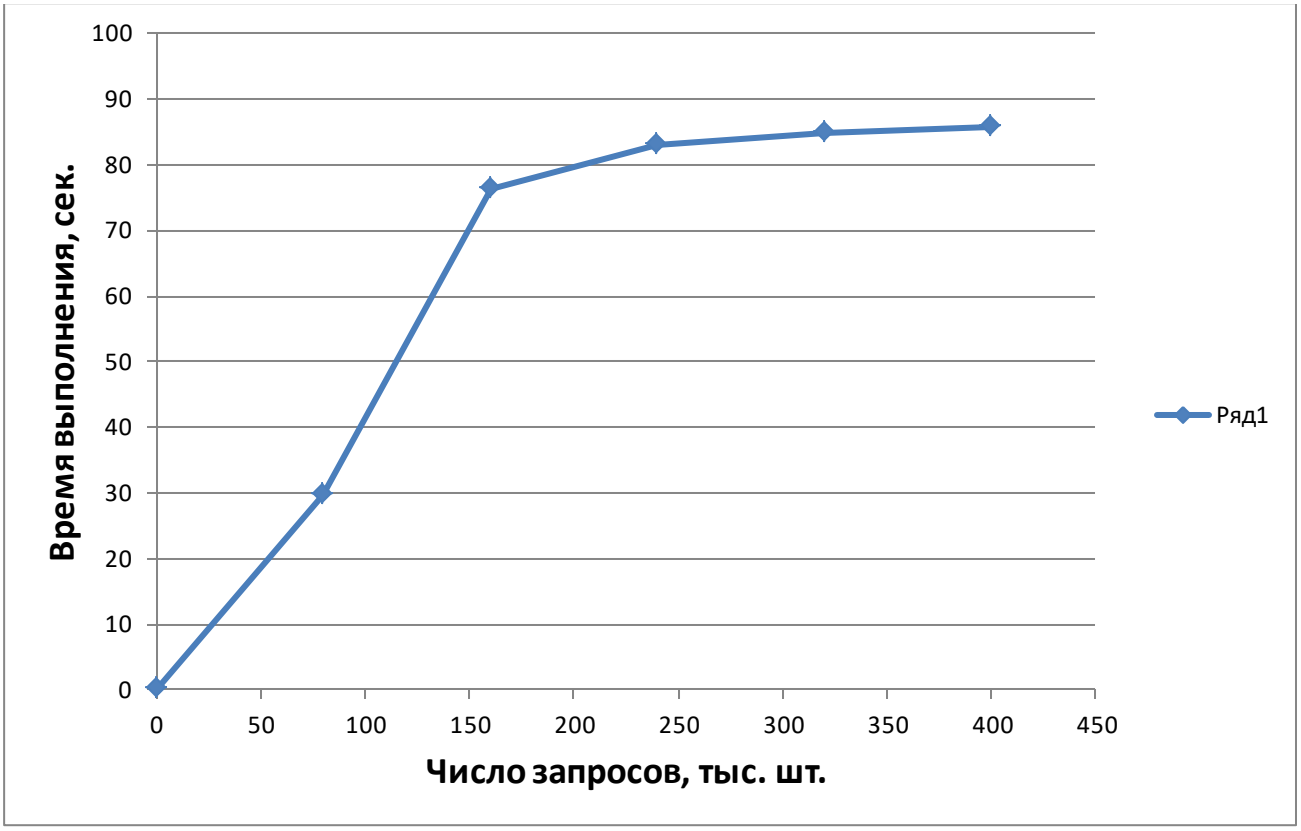

Рисунок 4.2 – Зависимость времени выполнения от общего числа запросов на получение рекомендации

Как видно из рисунка 4.2 наибольший рост времени выполнения запроса на получение рекомендации наблюдается в промежутке от 80 до 160 тыс., второй по времени выполнения скачек наблюдается в промежутке от 0 до 80 тыс. запросов. В противовес, начиная от 160 тыс. до 400 тыс. запросов рост времени затрачиваемого на выполнения анализа становиться незначительным.

Это связано в первую очередь с тем, что в промежутке от 0 до 160 тыс. запросов был перебран почти полный состав фильмов.

Таким образом, можно сделать вывод, что, как и описывалось в первой главе, основная нагрузка от алгоритмов, основанных на коллаборативной фильтрации, связана с размером различных данных, в данном случае фильмов.

В завершении проверим, какие фильмы будут рекомендованы на основании выборки фильмов, составленной испытуемым. Идентификаторы фильмов, их названия и оценка, составлены испытуемым и отображены в таблице В.1 приложения В.

В результате выполнения запроса на получение рекомендации система выдала три фильма, изображенных на рисунке 4.3.

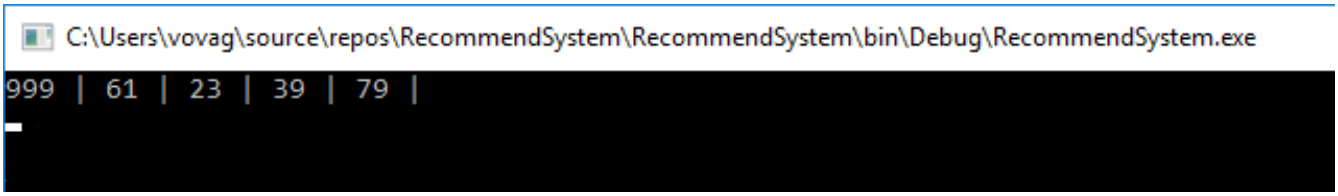

Рисунок 4.3 – Результат работы рекомендательной системы

Или, если соотнести названия фильмов и их идентификаторы, то:

1. 23 - Таксист (1976);
2. 39 - Странные дни (1995);
3. 79 - Беглец (1993).

По результату просмотра данных фильмов можно выставить следующие оценки: 4, 4, 5. Таким образом, можно заключить, что в случае с испытуемым рекомендательная система сработало в большей степени исправно.

На основании произведенных тестов можно сделать ряд определённых выводов:

1. качество прогнозов у реализованного алгоритма по методике RMSE находится в районе 0.94 – 0.96;
2. скорость работы рекомендательной системы напрямую зависит от количества товаров (в данном случае в роли товаров выступают фильмы);
3. результат тестирования на испытуемом прошёл успешно и оценки, выставленные испытуемым скорее положительны.

Таким образом, тестирование реализованного алгоритма разработанной рекомендательной системы было выполнено успешно. Далее необходима отладка кода для повышения качества прогнозов и скорости работы алгоритма.

ЗАКЛЮЧЕНИЕ

Таким образом, данная выпускная квалификационная работа по разработке рекомендательной системы с настраиваемым блоком анализа данных завершена успешно.

В ходе данной работы был проведен анализ основных подходов к реализации рекомендательных систем, а также существующие решения рекомендательных систем. По результату анализа были выделены основные преимущества и недостатки, на основании которых была доказана актуальность рассматриваемой темы.

По результату анализа были выделены, основные требования, которые должны быть реализованы в результирующем программном продукте и на основании которых была спроектирована архитектура рекомендательной системы с настраиваемым блоком анализа данных и затем реализована, как программный продукт.

Рекомендательная система успешно прошла тестирование по нескольким критериям. В результате тестирования была проанализирована эффективность разработанного алгоритма по методике RMSE, была оценена скорость работы, из которой следует, что скорость работы в первую очередь зависит от количества рассматриваемых товаров, в случае тестов – фильмов.

Помимо этого был проведен дополнительный тест над испытуемым – разработчиком рекомендательной системы, который позволил на практике оценить качество рекомендаций.

К дальнейшим планам по развитию программного продукта относится:

- доработка используемого алгоритма для повышения эффективности;
- разработка дополнительных вариантов алгоритмов, предоставляемых вместе с основным блоком;
- расширение функционала.

В дальнейшем данный программный продукт можно использовать, как инструмент для создания рекомендательных систем на различных сайтах, так и, как основу для дальнейших программных продуктов, например: сервисной рекомендательной системы или плагина для браузера.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Рихтер, Дж. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. – 4-е издание. – Санкт-Питербург: Изд-во Питер, 2013. – 896 с.;
2. Гудфеллоу, Я. Глубокое обучение / Бенджио И., Гудфеллоу Я., Курвилль А. – Москва: Изд-во ДМК Пресс, 2017. – 652 с.;
3. Шилдт, Герберт C# 4.0. Полное руководство / Герберт Шилдт. – Москва: Изд-во Вильямс, 2015. – 670 с.;
4. Статья об рекомендательных системах компании Яндекс [Электронный ресурс] URL:https://habr.com/ru/company/yandex/blog/241455/ (дата обращения: 28.02.2020);
5. Статья об рекомендательных системах на сайте IBM [Электронный ресурс] URL:https://www.ibm.com/developerworks/ru/library/os-recommender1/index.html; (дата обращения: 28.02.2020);
6. Официальный сайт рекомендательной системы RetailRocket [Электронный ресурс] URL:https://retailrocket.ru/; (дата обращения: 28.02.2020);
7. Официальный сайт рекомендательной системы RetailRocket / Тарифы [Электронный ресурс] URL:https://retailrocket.ru/pricing/; (дата обращения: 01.03.2020);
8. Интервью генерального директора Retail Rocket Николая Хлебинского агентству EmailMatrix [Электронный ресурс] URL:https://emailmatrix.ru/blog/retailrocket-platform/ (дата обращения: 01.03.2020);
9. Официальный сайт рекомендательной системы Rees46 [Электронный ресурс] URL:https://rees46.com/ru/price; (дата обращения: 03.03.2020);
10. Официальный сайт документации по рекомендательной системе LensKit, раздел алгоритмов [Электронный ресурс] URL: https://lkpy.readthedocs.io/en/stable/algorithms.html; (дата обращения: 05.03.2020);
11. Официальный сайт рекомендательной системы MyMedia Lite [Электронный ресурс] URL: http://www.mymedialite.net/index.html; (дата обращения: 06.03.2020);
12. Блейхут, Р.Э. Теория и практика кодов, контролирующих ошибки / Р.Э Блейхут. – Москва: Изд-во Книга по Требованию, 2013. – 576 с.;
13. Официальный сайт NoSQL базы данных LiteDB [Электронный ресурс] URL: http://www.litedb.org/ (дата обращения: 21.03.2020);

14. Hyndman, Rob J. Another look at measures of forecast accuracy / Rob J Hyndman, Anne B Koehler. – Amsterdam: Elsevier, 2006. – 679 с.;
15. Stopwatch Класс (System.Diagnostics) | Microsoft Dock [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/dotnet/api/system.diagnostics.stopwatch?view=netcore-3.1 (дата обращения: 21.03.2020).

ПРИЛОЖЕНИЕ В

Таблица Б.1 – Список фильмов с оценками, составленный испытуемым

ИД	Название	Оценка
1	История игрушек (1995)	4
50	Звёздные Войны (1977)	5
144	Крепкий орешек (1988)	5
226	Крепкий орешек 2 (1990)	5
550	Крепкий орешек 3 (1995)	4
195	Терминатор (1984)	5
96	Терминатор 2: Судный день (1991)	4
993	Геркулес (1997)	4
210	Индиана Джонс и последний крестовый поход (1989)	4
94	Один дома (1990)	5
894	Один дома 3 (1997)	3
271	Звёздный десант (1997)	3
82	Парк юрского периода (1993)	4
123	Страшилы (1996)	3
820	Космический джем (1996)	4
472	Сердце дракона (1996)	4
596	Горбун из Нотр-Дама (1996)	4
257	Люди в чёрном (1997)	5
22	Храброе сердце (1995)	4
1240	Призрак в доспехах (1995)	5
29	Бэтмен навсегда (1995)	3
231	Бэтмен возвращается (1992)	5
254	Бэтмен и Робин (1997)	2

403	Бэтмен (1989)	5
72	Маска (1994)	3
71	Король лев (1994)	4
217	Дракула (1992)	3
588	Красавица и чудовище (1991)	4
1274	Робокоп 3 (1993)	3
1078	Оливер и компания (1988)	4
95	Аладдин (1992)	4