Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки: 01.03.02 Прикладная математика и информатика

РАБОТА ПРОВЕРЕНА
Рецензент,
_____/Н.В. Плотникова
«____»_____20___г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
профессор
_____/А.А.Замышляева
«____»_____20___ г.

Разработка мобильного приложения с рекомендательной системой для сети ресторанов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–01.03.02.2020.306.476.ПЗ ВКР

Руководитель работы, к.т.н.,
доцент кафедры ПМиП
_____/Т.Ю. Оленчикова
«____»_____2020 г.

Автор работы
Студент группы ЕТ-413
_____/А.И. Лычагов
«____»_____2020 г.
Нормоконтролер,
ст. преподаватель
_____/Н.С. Мидоночева
«____»_____2020 г.

Челябинск
2020

АННОТАЦИЯ

Лычагов А.И. разработка мобильного приложения с рекомендательной системой для сети ресторанов – Челябинск: ЮУрГУ, ЕТ–413, 51 с., 21 ил., 8 табл., библиогр. список – 23 наим., 1 прил.

Выпускная квалификационная работа посвящена разработке мобильного приложения с рекомендательной системой для сети ресторанов.

В первом разделе рассматриваются основные методы построения рекомендательных систем, описываются алгоритмы фильтрации.

Второй раздел посвящен математической модели системы. Описаны формулы нахождения схожих пользователей и ресторанов.

В третьем разделе приведены алгоритмы работы приложения, реализация базы данных и архитектура приложения.

Четвёртый раздел содержит описание и пример работы программы, а также тестирование приложения.

В результате работы создано приложение, отличающееся от других аналогов, представленных на рынке, наличием рекомендательной системы на основе предпочтений пользователя.

Оглавление

ВВЕДЕНИЕ

Выбор куда пойти и где лучше провести своё свободное время всегда остро стоял перед людьми. Среди миллионов рекламных билбордов и вывесок, которые нас окружают, можно легко запутаться, а от разнообразия выбора может закружиться голова.

На сегодняшний день существует огромное количество ресторанов, представляющих кухни всех стран мира. В таком изобилии выбора человеку легко потеряться. Здесь на помощь и приходят рекомендательные системы.

Главное различие от поисковых систем состоит в том, что для рекомендательной системы не нужен чёткий запрос. На основании оценённых пользователем объектов строятся предположения, затем происходит возврат наиболее близких к ним результатов.

Поскольку рекомендательные системы остро востребованы в данный момент, приложение для сети ресторанов будет главным помощником для граждан средневысокого класса в непростом выборе.

Приложение должно обладать следующим функционалом:

– регистрация;

– авторизация пользователя;

– добавление отзыва ресторану;

– выставление оценки ресторану;

– настройка своих предпочтений по критериям.

1 АНАЛИЗ ТРЕБОВАНИЙ К ПРИЛОЖЕНИЮ И МЕТОДОВ ПОСТРОЕНИЯ РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМ

1.1 Постановка задачи

Целью выпускной квалификационной работы является создание мобильного приложения с применением рекомендательной системы.

Тематика приложения – сеть ресторанов. Пользователю приложения будет предоставляться возможность зарегистрировать свой собственный аккаунт, в котором он сможет выставить оценки или выделить необходимые пункты по различным критериям, таким как тип предпочитаемой им кухни, средний чек ресторана, разнообразие меню и т. д. Также пользователь сможет оставить оценку и отзыв об уже посещенных им ресторанах. На основе его предпочтений и отзывах пользователю будут предоставляться рекомендации о других ресторанах, которые могут ему понравиться. Информация о ресторанах будет браться с их официальных сайтов.

Для достижения поставленной цели необходимо решить следующие задачи:

– изучить и проанализировать существующие методы построения рекомендательных систем и платформы для разработки сайтов;

– спроектировать клиент-серверную архитектуру приложения;

– разработать базу данных;

– разработать алгоритм фильтрации;

– разработать алгоритм рекомендательной системы;

– реализовать приложение и проверить его работоспособность.

1.2 Методы рекомендательных систем

Рекомендательные системы – это алгоритмы, предназначенные для предложения пользователям соответствующих элементов (элементов, представляющих фильмы для просмотра, текста для чтения, продуктов для покупки или чего-либо еще, в зависимости от отрасли) [1].

В отличие от поисковых систем, рекомендательная система не требует четкого запроса для получения ответа. Пользователю предлагается оценить некоторые объекты из коллекции, и на основе его оценок и сравнения с оценками других пользователей делаются предположения и возвращаются результаты, наиболее близкие к ним [2].

В каждой рекомендательной системе мы имеем дело с пользователем, которому предоставляется много альтернатив, среди которых ему необходимо осуществить свой выбор. Пользователю может не хватать опыта и знаний для того, чтобы самостоятельно отбросить альтернативы, которые не соответствуют его потребностям. Пользователь в определенной форме, явно или неявно, предоставляет системе информацию о своих предпочтениях, при этом о некоторых альтернативах он может даже и не знать. Таким образом, рекомендательная система представляется как система, которая использует определенный тип фильтрации и существующие сведения о потребностях пользователя, для рекомендации ему набора альтернатив, которые считает наиболее полезными для него [9].

В разработке рекомендательной системы обычно используют 3 основных вида фильтрации:

– фильтрация, основанная на контенте;

– коллаборативная фильтрация;

– гибридная фильтрация.

1.2.3 Рекомендательные системы, основанные на контенте

Рекомендательные системы, основанные на контенте – это предметно-зависимый алгоритм, который делает упор на анализ атрибутов предметов для того, чтобы генерировать прогнозы. Когда необходимо рекомендовать документы, такие как веб-страницы, публикации и новости, метод рекомендации, основанный на контенте является наиболее успешным. В этом методе рекомендация сделана на основе профилей пользователей, использующих функции, извлеченные из содержания элементов, которые

пользователь оценил в прошлом. Предметы, которые в основном относятся к положительно оцененным, рекомендуются пользователю [3]. Рекомендательные системы, основанные на контенте, используют разные типы моделей, чтобы найти сходство между документами, чтобы генерировать значимые рекомендации. Например, можно использовать вектор Космической модели, такой как TF-IDF (от англ. TF — term frequency, IDF – inverse document frequency) или вероятностные модели, такие как классификатор Байеса, деревья решений или нейронные сети для моделирования взаимосвязи между различными документами внутри корпуса. Эти методы дают рекомендации путем изучения базовой модели с помощью любого статистического анализа или техники машинного обучения. Рекомендательные системы, основанные на контенте, не нуждаются в профиле других пользователей, поскольку они не влияют на рекомендацию. Кроме того, если профиль пользователя изменился, система по-прежнему может скорректировать свои рекомендации в течение очень короткого периода времени. Основным недостатком этого метода является необходимость глубокого знания и описания особенностей предметов в профиле [19-20].

1.2.2 Коллаборативные рекомендательные системы

Коллаборативная рекомендательная система – это предметно-независимая технология прогноза для контента, который не может быть легко описан с помощью метаданных, таких как фильмы или музыка [4]. Коллаборативная рекомендательная система работает путем создания базы данных (матрицы пользовательских элементов) предпочтений для элементов пользователями. После сопоставления пользователей с соответствующими интересами и предпочтениями путём расчёта сходств между их профилями, система даёт рекомендацию. Такие пользователи создают группу под названием «соседство» [10]. Пользователь получает рекомендации по тем пунктам, которые он не оценил раньше, но это уже было положительно оценено пользователями в его «соседстве». Рекомендации, которые производит

коллаборативная рекомендательная система могут быть либо прогнозом, либо рекомендацией [21]. Предсказание – это числовое значение $r_{i,j}$, выражающее прогнозируемый балл элемента j для пользователя i, в то время как рекомендация представляет собой список из первых N элементов что больше всего понравится пользователю, как показано на рисунке 1.

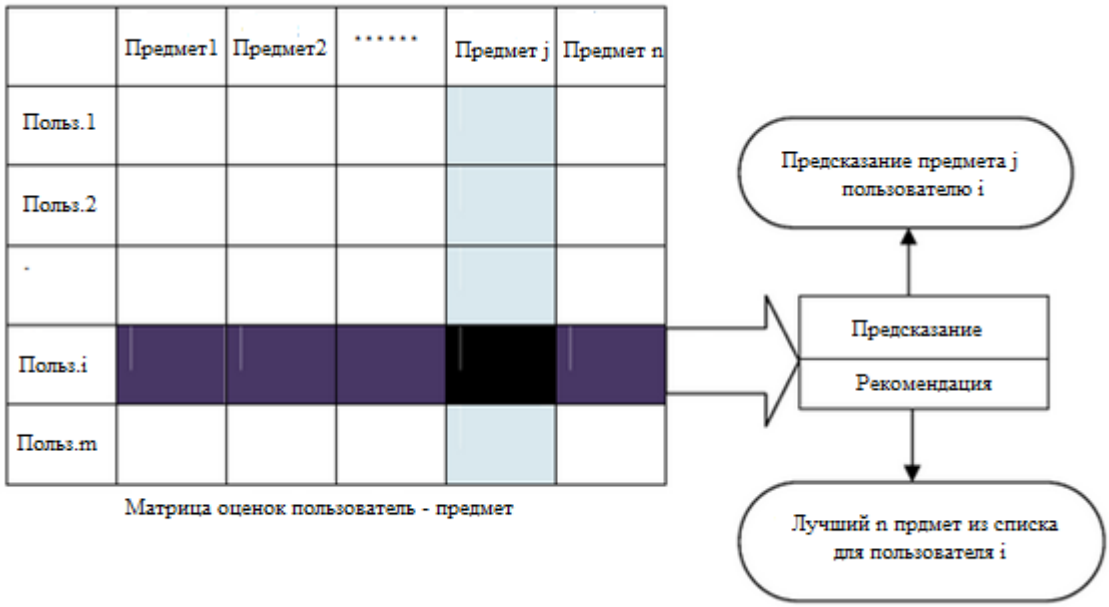

Рисунок 1 – Матрица оценок пользователя

1.2.3 Гибридные рекомендательные системы

Из немногих недостатков коллаборативной фильтрации по сравнению с методами, основанными на описании признаков, можно выделить проблему «холодного старта» [5]. В гибридном подходе используются композиции алгоритмов, основанные на описаниях признаков и результатах коллаборативной фильтрации, что при учёте различной дополнительной может увеличить показатели. В гибридном подходе используются композиции алгоритмов, основанные на результатах коллаборативной фильтрации и описаниях признаков [22].

1.3 Алгоритмы фильтрации

1.3.1 Корреляция Пирсона

Коэффициент корреляции – это мера того, насколько хорошо два набора

данных ложатся на прямую. Формула сложнее, чем для вычисления евклидова расстояния, но она дает лучшие результаты, когда данные плохо нормализованы, а также лучше работает для разреженных данных.

1.3.2 Метод, вычисляющий евклидово расстояние

Объединение или метод древовидной кластеризации используется при формировании кластеров несходства или расстояния между объектами. Эти расстояния могут определяться в одномерном или многомерном пространстве. Например, если вы должны кластеризовать типы еды в кафе, то можете принять во внимание количество содержащихся в ней калорий, цену, субъективную оценку вкуса и т. д. Наиболее прямой путь вычисления расстояний между объектами в многомерном пространстве состоит в вычислении евклидовых расстояний. Если вы имеете двух- или трёхмерное пространство, то эта мера является реальным геометрическим расстоянием между объектами в пространстве (как будто расстояния между объектами измерены рулеткой). Однако алгоритм объединения не «заботится» о том, являются ли «предоставленные» для этого расстояния настоящими или некоторыми другими производными мерами расстояния, что более значимо для исследователя; и задачей исследователей является подобрать правильный метод для специфических применений.

1.3.3 GroupLens алгоритм

GroupLens хранит результаты численной оценки профиля пользователя на сервере. У данного алгоритмы существует два способа оценки. Первый – это оценка пользователями, а второй – оценка по сходству с профилями других пользователей.

В базе данных рейтингов записывается реакция пользователя после просмотра фильма. Результаты показаны на рисунке 2. Пустая ячейка означает, что пользователь еще не смотрел фильм. Знак вопроса говорит о том, что пользователь смотрел фильм, но еще не оценил его. GroupLens получает значение оценки для фильма «Семь», который Саша еще не оценил, с

использованием коэффициента корреляции Пирсона, который находится в диапазоне от –1 до 1. Коэффициенты корреляции Саши с другими людьми составляют: 0,57 с Димой, 0,74 с Машей, 0,12 с Марком и –0,85 с Аней. С этими значениями мы можем получить прогнозируемое значение оценки, которую Саша отметил бы для фильма «Семь». Однако, чтобы оценить ценность каждого пользователя по группам, нам необходимо достаточно данных. Малое количество оценочных данных приведет к плохим прогнозам. Поэтому GroupLens требует от пользователя определенного количества задач.

Таблица 1 – Оценки GroupLens

		Фильмы					
		Титаника	Матрица	Холм	Маска	Хатико	Семь
Пользователи	Дима	5	3	1	4	4	4
	Маша	3	5	2	3	5	3
	Марк	2	4			3	2
	Аня	4	4		1	2	5
	Саша	3	3	2		5	?

1.3.4 Сингулярное разложение матрицы

Пусть M – матрица оценок пользователей размера $n\ x\ m$.

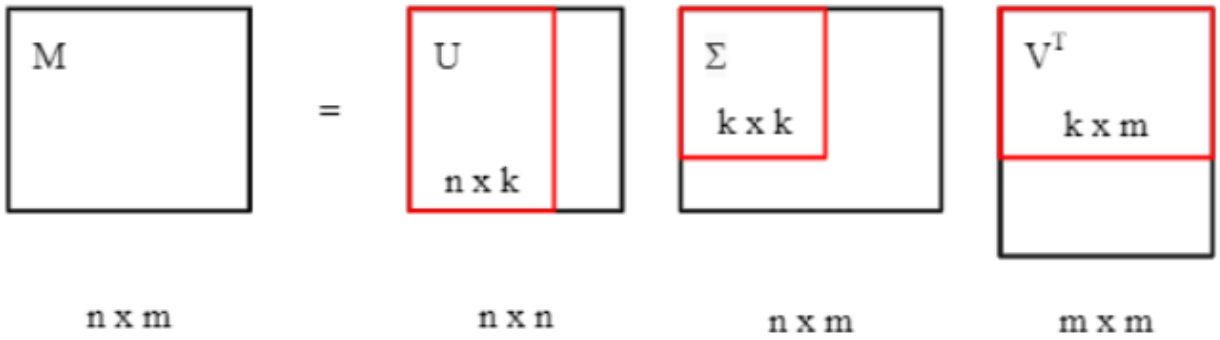

Рисунок 2 – Сингулярное разложение матрицы

Используя сингулярное разложение матрицы оценок пользователей, можно получить две матрицы: матрицу U размера $n\ x\ k$ и матрицу V размера

$m\ x\ k$, где n – число пользователей, m – число объектов, а k – набор факторов. Данные факторы являются характеристикой вкусов и предпочтений (для матрицы U) и характеристикой самих объектов, относительно вкусов и предпочтений пользователей (для матрицы V). Таким образом, имея данные матрицы для получения рейтинга пользователя u к объекту i необходимо лишь вычислить скалярное произведение u-ой строки матрицы U и i-ой строки матрицы V.

1.3.5 Классификатор Байеса

Байесовский классификатор — это основанный на применении теоремы Байеса простой вероятностный классификатор со строгими предположениями о независимости.

Несмотря на очень упрощенные условия, байесовские классификаторы часто работают намного лучше во многих сложных жизненных ситуациях.

Главным достоинством байесовского классификатора является малое количество данных, которые необходимы для оценки параметров, чтобы произвести классификацию [15].

1.4 Интегрированные среды разработки

1.4.1 Программа Eclipse

Это бесплатная среда разработки, созданная компанией Eclipse Foundation. Эта программа является базой, которая регулирует процессы создания приложений.

Преимущества Eclipse:

– полностью переведённый на русский язык интерфейс;

– хорошая совместимость с компьютерами, обладающими низкой производительность;

– дополнительные функции для серверной работы и анализа базы данных;

– возможность подключения к модулям;

– работа в групповом режиме (если проект создают несколько человек одновременно).

Eclipse заняла лидирующие места ещё несколько лет назад и до сих пор их удерживает. Однако, после выхода Android Studio, Google решил перевести сотрудничество с Eclipse на «второй план».

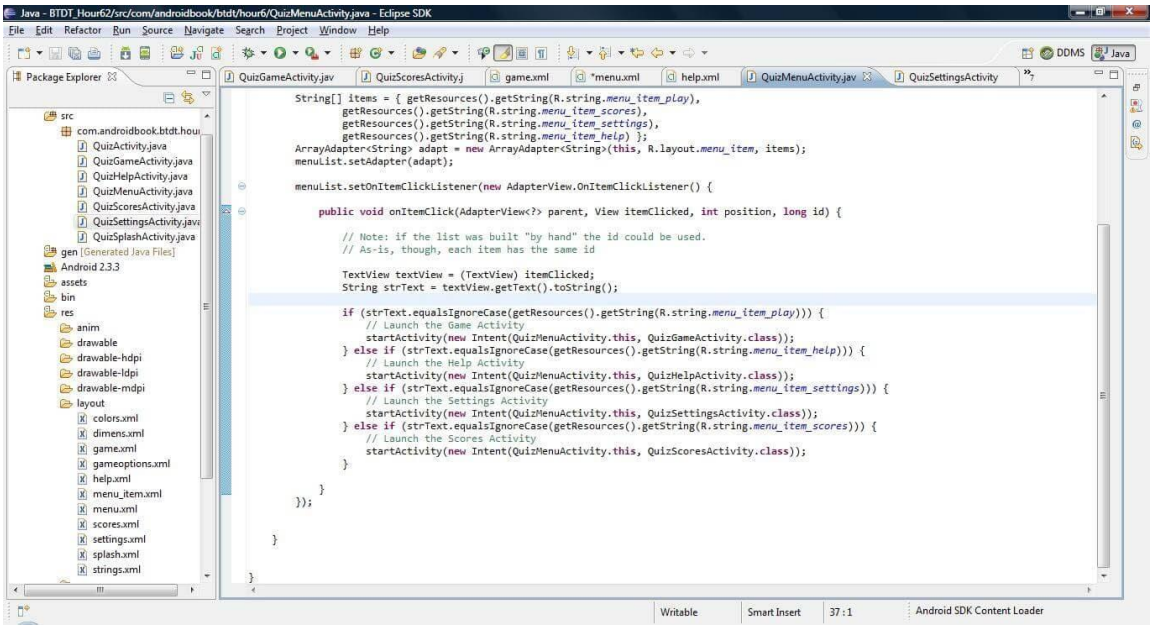

Рисунок 3 – Программа Eclipse

1.4.2 IntelliJ Idea

Данная программа – разработка российской компании JetBrains. Как и Eclipse, данная среда разработки позволяет создавать программы и приложения на нескольких языках программирования. Также IntelliJ Idea не перегружает ПК.

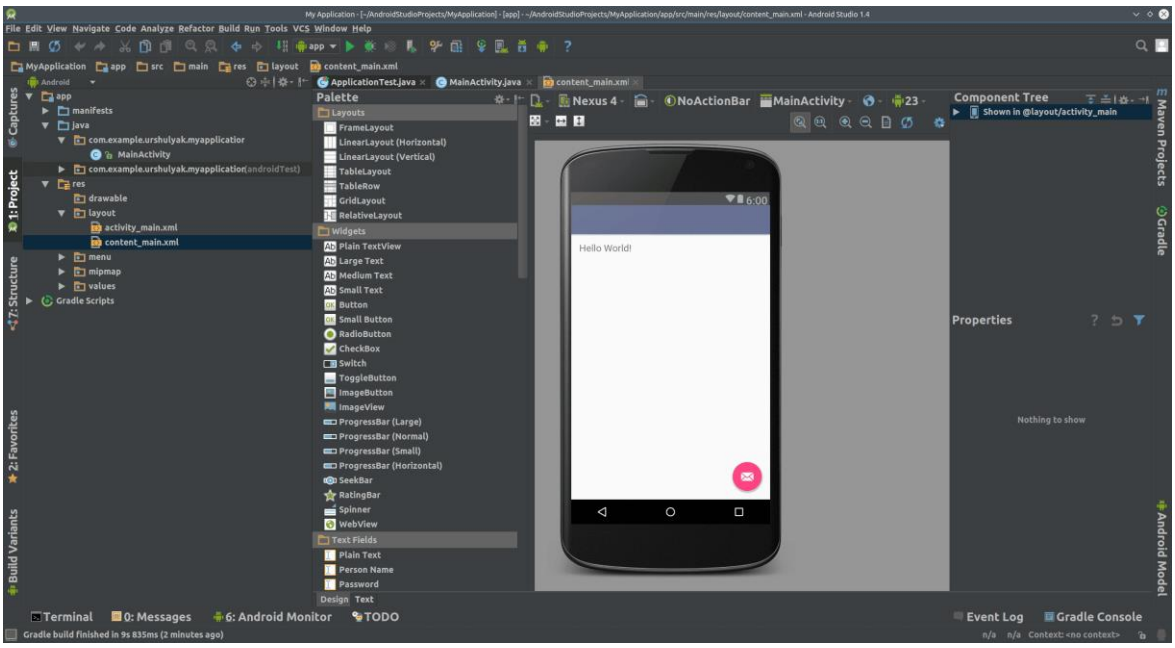

Рисунок 4 – Intellij Idea

Её основные достоинства:

– более оперативная отладка значений;

– присутствует автозаполнитель методов;

– наличие рефакторинга;

– простой, понятный и лаконичный интерфейс;

– отлично подходит для разработчиков и программистов Java.

Главный минус IntelliJ Idea в том, что это платное приложение. Однако человек, понимающий ситуацию на рынке, не будет считать это минусом.

1.4.3 Android Studio

На базе IntelliJ Idea компания Google создала свою собственную интегрированную среду разработки.

При отсутствии принципиальных различий и небольшого числа нововведений, Android Studio удалось стать достойным конкурентом двум предыдущим IDE в кратчайшие сроки. Продукт компании Google вобрал в себя все достоинства уже существующих на рынке решений. Одним из немногих недостатков программы является небольшое количество функций в кодовом редакторе и общих настройках.

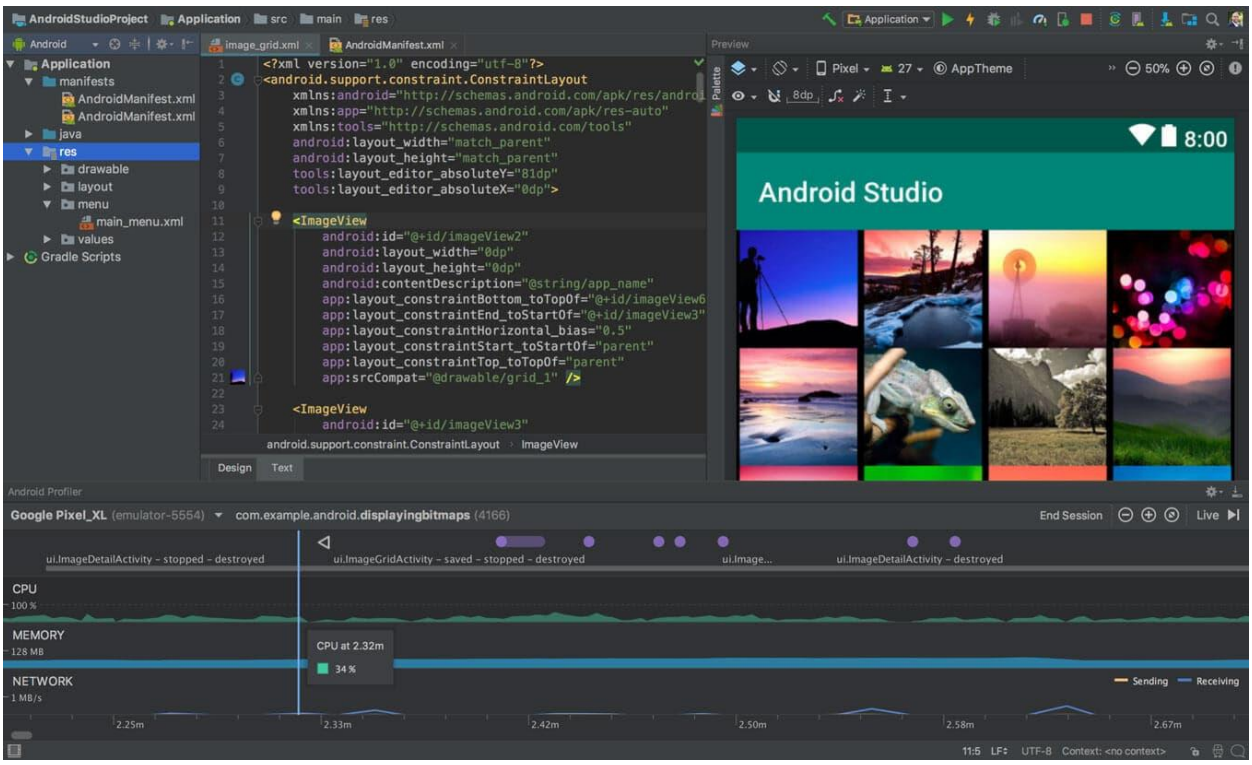

Рисунок 5 – Andoid Studio

1.5 Обоснование выбора среды разработки приложения

В качестве IDE приложения был выбран продукт компании Google. Android studio основана на программном обеспечении IntelliJIDEA, но за счёт её удобного графического интерфейса, средств отладки, а также вёрстки в реальном времени данная среда разработки выигрывает у своих конкурентов. Также доступно множество вариантов разрешений экранов и их размеров. Присутствие раздела справки сильно облегчает работу в среде разработки. Встроены инструменты улучшения качества приложений и монетизации. Имеются инструменты для отслеживания эффективности рекламных объявлений. Добавлено средство взаимодействия с бета-тестерами и многое другое. При этом IDE обладает функцией GoogleCloudMessaging, позволяющей посылать данные с сервера на Android-устройства, используя облако. Данная возможность хорошо подходит для отправки push-уведомлений в приложения. Для оптимизации работы приложения в Android Studio используется инструмент анализа производительности под названием MemoryMonitor, который выдаёт информацию об использовании памяти. Android Studio –

новая и полностью интегрированная среда разработки приложений, выпущенная компанией Google. Данный продукт призван снабдить разработчиков новыми инструментами для создания приложений, а также предоставить альтернативу устаревшей среде разработки Eclipse [14-16].

1.6 Аналоги приложений, реализованных на Android

1.6.1 TripAdvisor

Мобильное приложение TripAdvisor для Android – полезная программа для тех, кто собирается посетить незнакомые города в разных странах мира, а также помощь в планировании поездки с учетом личных предпочтений. Также даёт возможность посмотреть рейтинг ресторанов в любом городе.

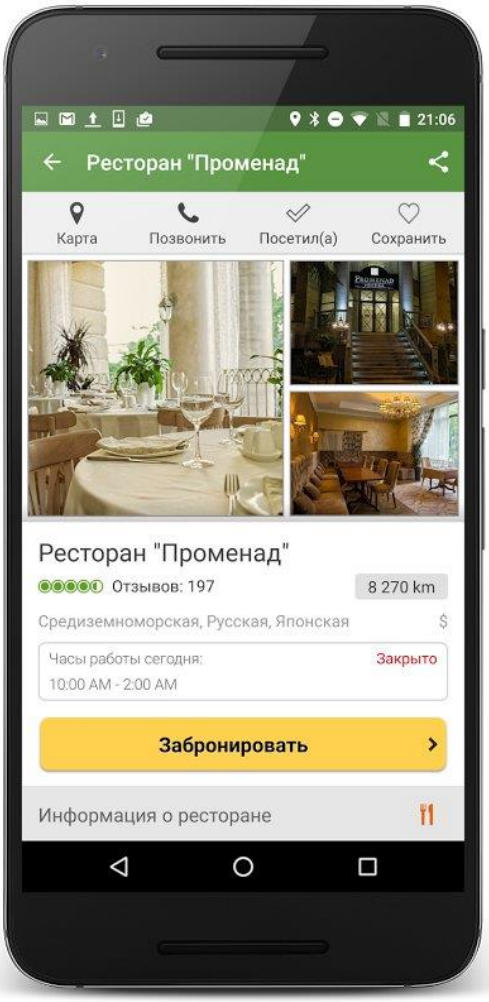

Рисунок 6 – TripAdvisor

1.6.2 Яндекс-карты

Приложение даёт возможность просматривать расположение различных объектов на карте, в том числе ресторанов, просматривать отзывы о них и видеть оценки других пользователей.

1.6.3 Foodspotting

Foodspotting – это полезное социальное приложение, которое поможет вам найти в пределах досягаемости хороший ресторан, столовую или кафе.

Программа Foodspotting – это не рекламная площадка с навязыванием определённых ресторанов и заведений. Все отзывы, фотографии и комментарии вносятся активными пользователями, которые делятся хорошими местами с остальными людьми.

Самое лучшее применение Foodspotting находит во время путешествия за рубежом. В крупных городах и туристических местах отмечены буквально все интересные заведения, при чём в комментариях люди часто приводят фотографии заказанного и сумму чека, что поможет сориентироваться при выборе ресторана.

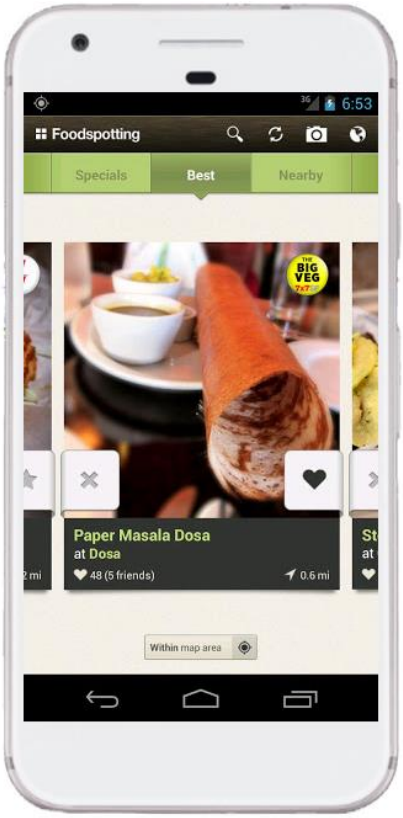

Рисунок 7 – Foodspotting

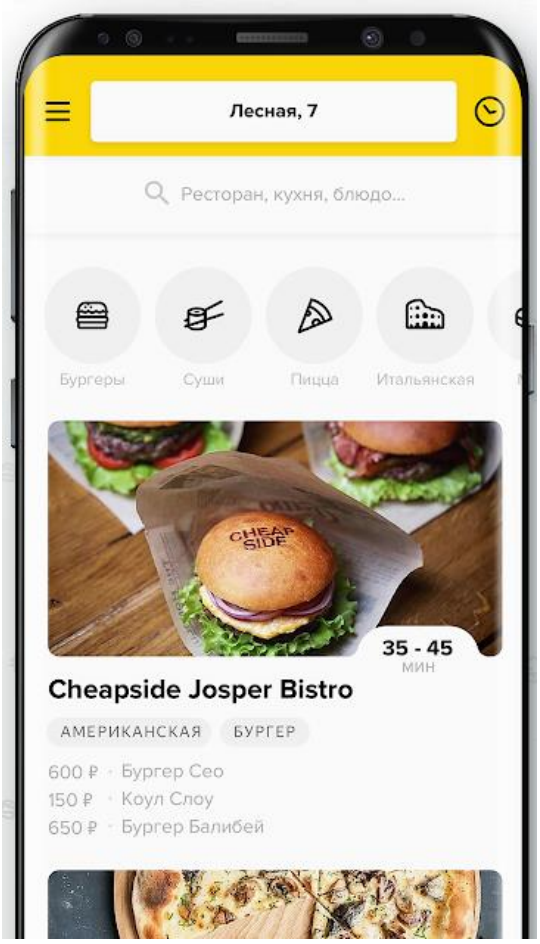

Рисунок 8 – Яндекс-карты

1.7 Система управления базами данных

MySQL представляет собой свободную систему управления базами данных (СУБД). MySQL принадлежит организации SunMicrosystems, которая занимается разработкой и поддержкой приложения. Распространение MySQL осуществляется под лицензией GNU (General Public License), либо же с использованием собственной коммерческой лицензии. Кроме того, лицензионные пользователи могут отправлять заказы, по которым разрабатывается соответствующая функциональность, что обусловило появление механизма репликации уже в ранних версиях MySQL [6].

MySQL наиболее эффективно используется в приложения малого и среднего масштаба. Он входит в стандартные пакеты разработки веб-приложений для таких операционных систем, как Windows (или WAMP) и

Linux (LAMP). В большей части случаев MySQL выполняет роль сервера, к которому обращаются с запросами локальные либо удаленные клиенты, однако MySQL может быть включен и в автономные программы за счет дистрибутива, включенного в библиотеку внутреннего сервера [11].

MySQL способен осуществлять поддержку нескольких механизмов базы данных. Эти механизмы определяют, каким конкретным образом он в конкретный момент времени производит хранение, извлечение и обработку данных. Соответственно, каждый из механизмов имеет свой индивидуальный набор возможностей и преимуществ. Постепенно мощность и скорость имеющихся механизмов возрастает.

1.8 Выводы к первой главе

В данном разделе рассмотрены существующие методы построения рекомендательных систем, а также проведено их сравнение, описаны достоинства [18].

По статистике Android является самой популярной ОС среди мобильных устройств, занимающий 87% всего рынка. Также он имеет большой выбор средств для быстрой разработки, поэтому его выбор является наилучшим.

Доля рынка операционных систем	2016г.	2017г.
Android	84.1%	87.5%
Apple iOS	13.6%	12.1%
Others	2.3%	0.3%
Total	100.0%	100.0%
Total Growth Year-over-Year %	9.5%	6.0%

Source: Strategy Analytics

Рисунок 9 – Доля рынка операционных систем

В настоящее время Java является самым популярным языком программирования для мобильной разработки на Android.

Java является официальным языком для разработки Android и

поддерживается Android Studio. Поэтому в качестве языка программирования был выбран язык Java.

В качестве СУБД был выбран MySQL, т.к. он является свободно распространяемым.

После выполненного анализа был выбран алгоритм коллаборативной рекомендательной системы, исходя из её высокой точности.

Приложение должно обладать следующим функционалом:

– регистрация;

– авторизация пользователя;

– добавление отзыва ресторану;

– выставление оценки ресторану;

– настройка своих предпочтений по критериям.

2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ

2.1 Алгоритм работы системы

Рассмотрим метод item-based в задаче коллаборативной фильтрации. Выборка тут представляет собой тройки вида $(u, i, r_{u,i})$, где u – пользователь, i – специализирующийся на конкретной кухне ресторан, $r_{u,i}$ – оценка, которую пользователь u поставил ресторану i. И будем также считать, что рейтинги нормированы на отрезке [0; 1].

Метод item-based. Пусть есть матрица R, составленная из оценок пользователей. Для нахождения меры схожести пользователей воспользуемся коэффициентом корреляции Пирсона:

$$sim(u,a) = \frac{\sum_i (r_{u,i} - \overline{r}_u)(r_{a,i} - \overline{r}_a)}{\sqrt{\sum_i (r_{u,i} - \overline{r}_u)^2}\sqrt{\sum_i (r_{a,i} - \overline{r}_a)^2}}, \quad (1)$$

где R = $(r_{u,i})$ – матрица оценок пользователя u ресторану i;

$\overline{r_u}$ – средняя оценка, которую ставит пользователь u.

Для прогнозирования рейтинга воспользуемся формулой:

$$\widehat{r_{a,\imath}} = \frac{\sum_u r_{u,i} sim(u,a)}{\sum_u |sim(u,a)|}, \quad (2)$$

где $\widehat{r_{a,\imath}}$ – прогнозируема оценка пользователя a ресторану i.

2.2 Алгоритм работы системы для проблемы холодного старта

Для решения проблемы холодного старта для нового пользователя считаем сходство профилей пользователей u и a $sim(u,a)$ на основании векторов – признаков пользователя:

$$sim(u,a) = \frac{\sum_j p_{u,j}\, p_{a,j}}{\sqrt{\sum_j {p_{u,j}}^2}\sqrt{\sum_j {p_{a,j}}^2}} * \frac{n_{u,a}}{n_p}, \quad (3)$$

где P = $(p_{u,j})$ – матрица профилей пользователей;

$n_{u,a}$ – число параметров, заданных одновременно пользователями и *a;*

n_p – общее число характеристик профиля.

Для нового ресторана считаем коэффициенты сходства ресторанов *m* и *k* по формуле (4):

$$sim(m,k) = \frac{\sum_j s_{m,j}\, s_{k,j}}{\sqrt{\sum_j {s_{m,j}}^2}\sqrt{\sum_j {s_{k,j}}^2}} * \frac{n_{m,k}}{n_s}, \quad (4)$$

где S = $(s_{m,j})$ – матрица характеристик ресторанов;

$n_{m,k}$ – число параметров, заданных одновременно ресторанами *m* и *k;*

n_s – общее число характеристик ресторана.

Рекомендация ресторана *i* пользователю *a* считаем по формуле (5):

$$\widehat{r_{a,k}} = \frac{\sum_m r_{a,m}\, sim(m,k)}{\sum_m |sim(m,k)|}, \quad (5)$$

где $\widehat{r_{a,k}}$ – прогнозируемая оценка нового ресторана *k* пользователю *a*.

2.3 Вывод ко второй главе

В данной главе разработана математическая модель рекомендательной системы коллаборативной фильтрации. Описан алгоритм, позволяющий находить схожих пользователей и схожие рестораны, а также рекомендовать рестораны пользователю, которые должны ему понравиться.

Для решения проблемы холодного старта был разработан алгоритм, позволяющий находить сходства пользователей на основе выбранных ими категорий.

3 РАЗРАБОТКА ПРИЛОЖЕНИЯ

3.1 Диаграмма вариантов использования

Диаграмма вариантов использования, также называемая диаграммой прецедентов (use-case diagram), отражает отношения между актерами и прецедентами и дает представление о функциональном поведении системы.

На рисунке 10 представлена диаграмма, описывающая типы пользователей и их взаимодействие с системой.

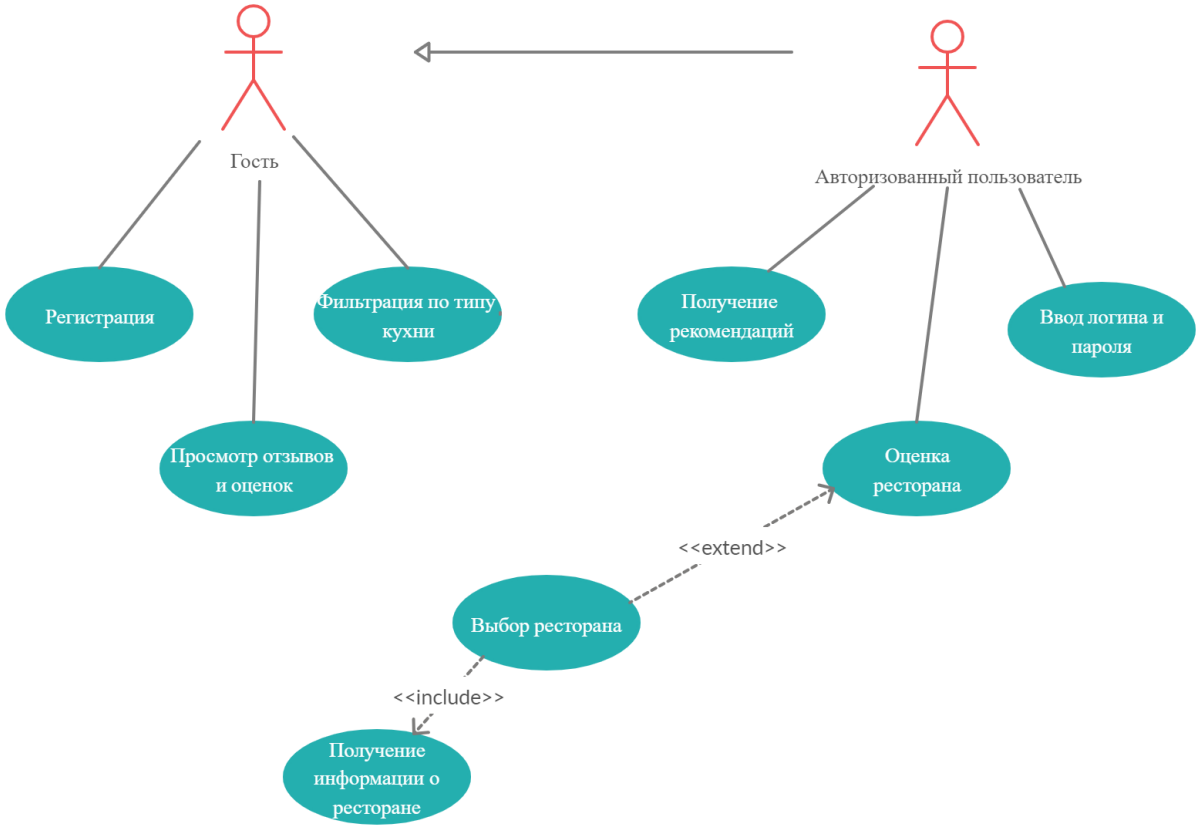

Рисунок 10 – Диаграмма вариантов использования системы

Далее приводится словесное описание (сценарий) для каждого прецедента.

Use-case: «Регистрация»

Для получения доступа к расширенному функционалу приложения пользователь может зарегистрироваться в нем, указав необходимые сведения о себе.

Главная последовательность:

Пользователь вводит логин и пароль и подтверждает ввод. Система переводит пользователя на страницу с информацией о профиле.

Альтернативная последовательность 1:

Ввод уже существующих логин. Система выдает сообщение о том, что логин занят и предлагает повторить ввод.

Альтернативная последовательность 2:

Ввод короткого пароля. Система выдает сообщение о том, что пароль менее шести символов, и предлагает повторить ввод.

Use-case: «Ввод логина и пароля»

Если у посетителя уже есть профиль в приложении, то он может авторизоваться на нем, указав свои логин и пароль.

Главная последовательность:

Пользователь вводит логин и пароль в форме на странице авторизации и подтверждает ввод. Система переводит пользователя на страницу с информацией о профиле.

Альтернативная последовательность:

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

Use-case: «Просмотр отзывов и оценок»

На главной странице приложения представлен каталог всех ресторанов и их оценок, существующих в базе данных.

Use-case: «Фильтрация по типу кухни»

Если у пользователя приложения нет желания просматривать все существующие рестораны, он может выставить необходимые фильтры и просмотреть только те рестораны, которые удовлетворяют им.

Главная последовательность:

На главной странице приложения в специальной вкладке пользователь по желанию выбирает вид или виды интересующих его кухонь и подтверждает ввод. После этого выводятся рестораны, соответствующие выставленным фильтрам.

Альтернативная последовательность:

В базе данных отсутствуют рестораны, удовлетворяющие выставленным фильтрам. Система выводит сообщение о том, что искомые рестораны не существуют.

Use-case: «Выбор ресторана»

При просмотре ресторанов пользователь приложения может выбрать заинтересовавший его ресторан, нажав на его название, и тем самым перейти на посвященную ему страницу.

Use-case: «Получение информации о ресторане»

На странице о ресторане может присутствовать информация о его параметрах. Например, о виде кухни, меню, среднем чеке.

Главная последовательность:

Пользователь выбирает ресторан и автоматически переходит на страницу ресторана, содержащую более подробную информацию.

Альтернативная последовательность:

В базе данных отсутствует информация. Посетитель приложения видит только название и адрес.

Use-case: «Оценка ресторана»

Авторизованным посетителям сайта предоставляется возможность оценивать рестораны на соответствующих им страницах.

Главная последовательность:

Пользователь выставляет оценку ресторану от одного до пяти посредством выбора количества звездочек.

Альтернативная последовательность:

Пользователь не авторизован. Система выводит сообщение о невозможности голосования и предлагает посетителю пройти авторизацию.

Use-case: «Получение рекомендаций»

Авторизованный в приложении пользователь может получить список ресторанов, которые по предположению рекомендательной системы сайта могут быть ему интересны.

Главная последовательность:

Пользователь по желанию выставляет необходимые ему фильтры и нажимает кнопку «Рекомендации» на главной странице приложения. Система выведет рекомендуемые ему рестораны.

Альтернативная последовательность 1:

Пользователь не авторизован. Система выводит сообщение о невозможности получения рекомендаций и предлагает пройти авторизацию.

Альтернативная последовательность 2:

Пользователь еще не оценил ни один ресторан. Система выводит сообщение о невозможности получения рекомендаций и предлагает оценить рестораны.

Альтернативная последовательность 3:

Отсутствуют пользователи, с которыми можно сравнить текущего (то есть текущий пользователь и остальные пользователи выставили оценки разным ресторанам или остальные пользователи не выставили оценки вообще). Система выводит сообщение о невозможности получения рекомендаций и предлагает повторить попытку позже.

Альтернативная последовательность 4:

В базе данных отсутствуют рестораны, удовлетворяющие выставленным фильтрам. Система выводит сообщение о том, что искомые рестораны не существуют.

3.2 Диаграмма классов

При разработке архитектуры приложения был использован шаблон проектирования Model-View-Presenter (MVP). Данный шаблон был разработан для облегчения автоматического модульного тестирования и улучшения разделения ответственности в презентационной логике (отделения логики от отображения):

Модель (Model) – элемент, заключающий в себе всю бизнес-логику, получая при необходимости данные из хранилища (в данном случае – базы

данных).

Представление (View) – реализует отображение данных, обращаясь к Presenter за обновлениями.

Представитель (Presenter) – реализует функциональность посредника между Model и View и отвечает за управление событиями пользовательского интерфейса.

В данном случае функции представления выполняет главным образом клиентское приложение, отображая необходимые данные, а со стороны разрабатываемого приложения роль представления частично играет модуль связи, отвечающий за непосредственную отправку этих данных клиенту.

На рисунке приведена общая диаграмма классов системы, отображающая взаимодействие классов между собой. Далее приведены диаграммы классов, раскрывающие элементы «Представитель» и «Модель» [13].

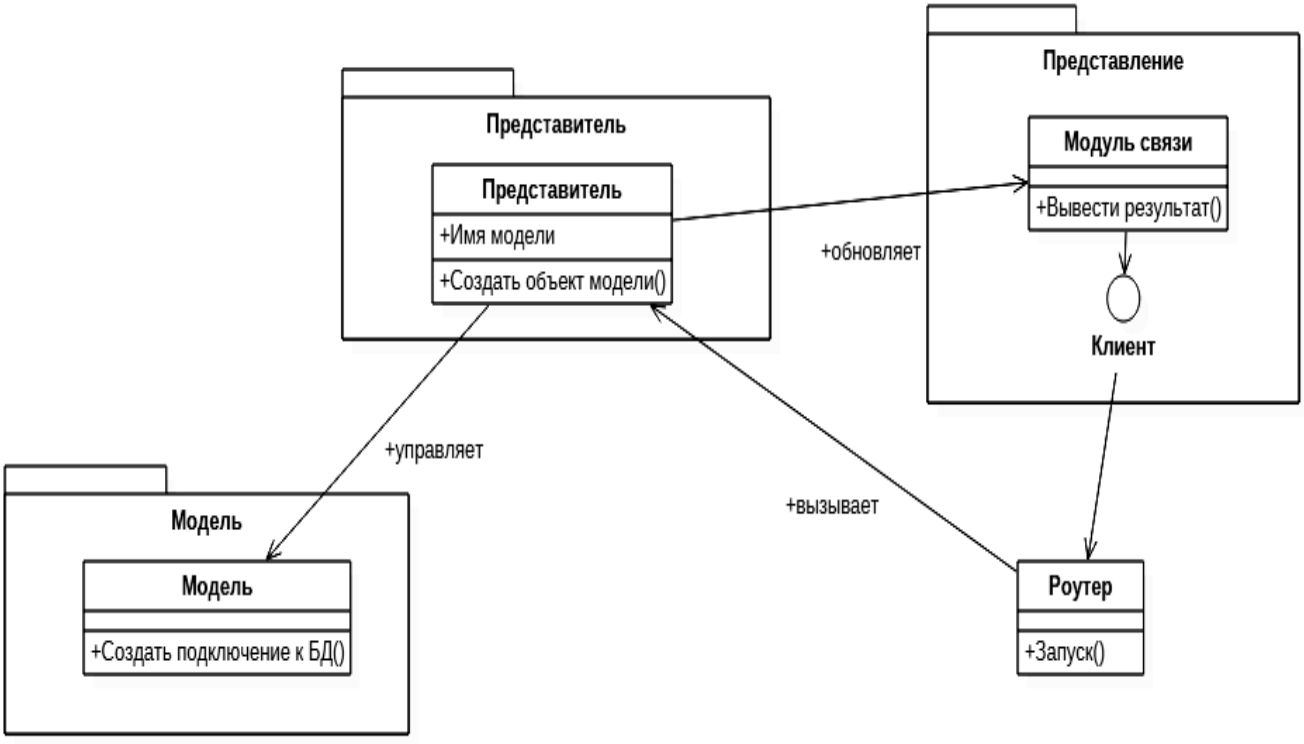

Рисунок 11 – Общая диаграмма классов системы

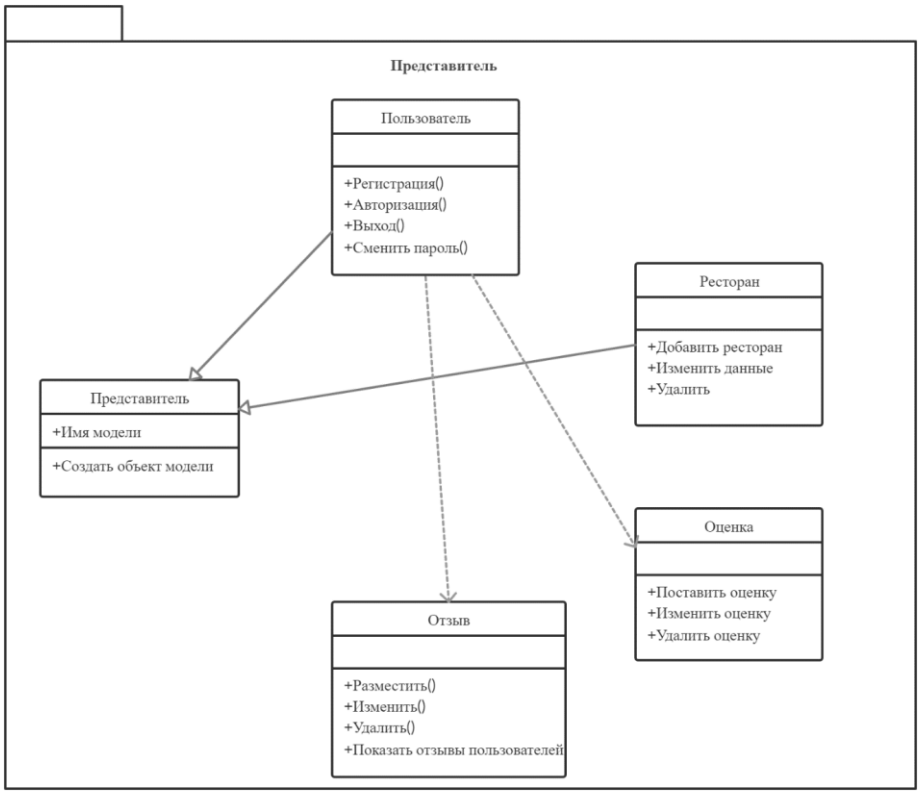

Рисунок 12 –Диаграмма классов элемента «Представитель»

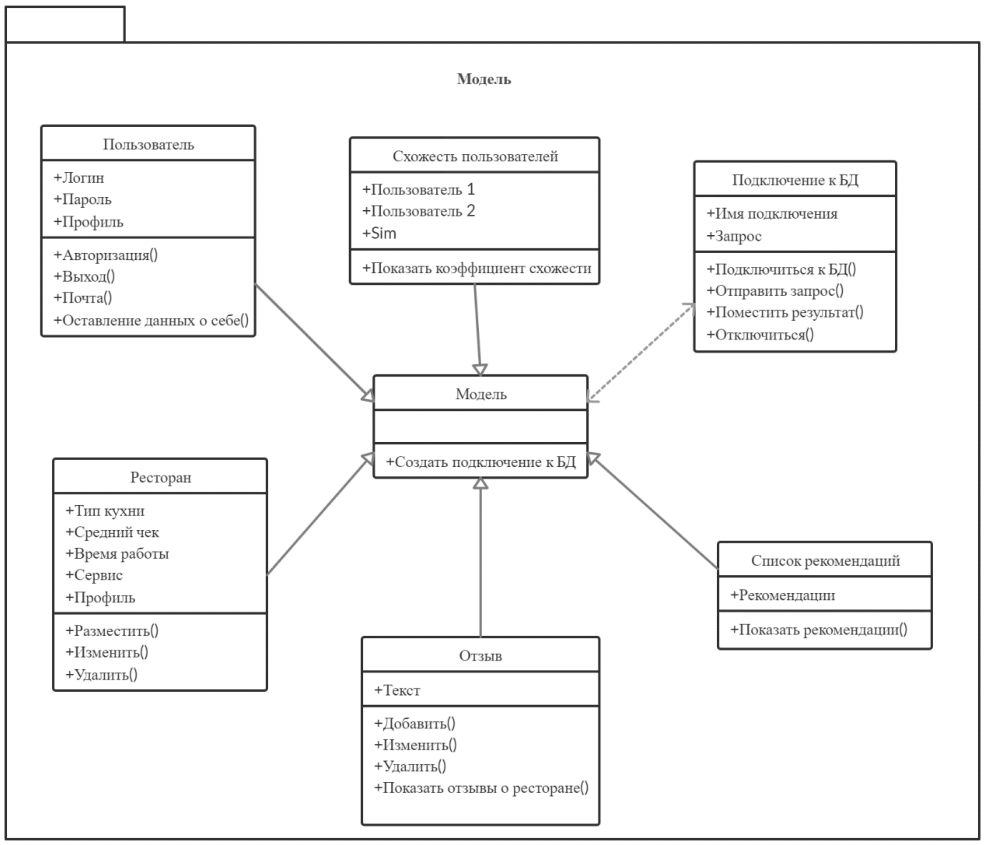

Рисунок 13 – Диаграмма классов элемента «Модель»

3.3 Диаграмма базы данных

На рисунке 14 приведена схема реляционной базы данных для разрабатываемой системы.

База данных предназначена для хранения всей долговременной информации системы: зарегистрированные пользователи, их оценки ресторанам, информация о типе кухне ресторана, его среднем чеке, времени работы [11-12].

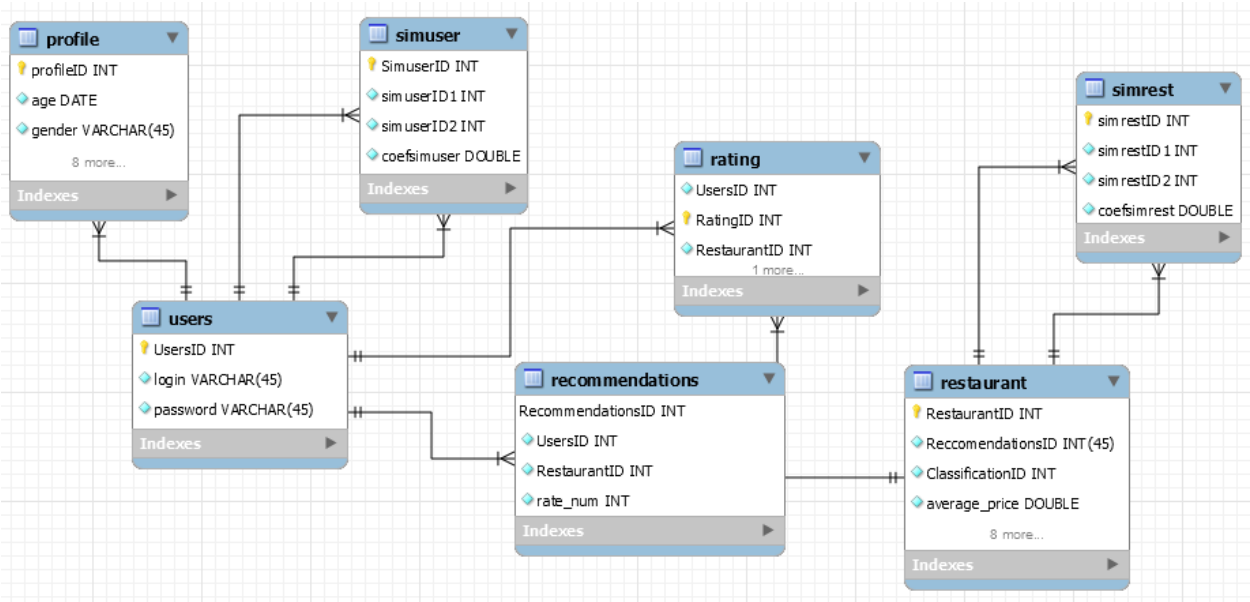

Рисунок 14 – Диаграмма базы данных

Таблица 1 – Пользователи

	Тип данных	Значение по умолчанию	Обязательность	Первичный ключ	Внешний ключ	Ограничения
UserID	int	–	+	+	–	–
Login	varchar(45)	–	+	–	–	уникальный
Password	varchar(45)	–	+	–	–	не менее 6 символов

В данной таблице хранятся данные пользователя – логин (Login) и пароль (Password). Логин должен быть уникальным. В целях повышения безопасности пароль не должен быть короче 6 символов.

Таблица 2 – Профиль

	Тип данных	Значение по умолчанию	Обязательность	Первичный ключ	Внешний ключ	Ограничения
ProfileID	int	–	+	+	–	–
age	date	–	+	–	–	–
gender	varchar(45)	–	+	–	–	–
name	varchar(45)	–	+	–	–	–
UserID	int	–	+	–	+	–
nationality	varchar(45)	–	+	–	–	–
attorus	int	–	+	–	–	{1,2,3,4,5}
attojap	int	–	+	–	–	{1,2,3,4,5}
attochin	int	–	+	–	–	{1,2,3,4,5}
attouzb	int	–	+	–	–	{1,2,3,4,5}
attogeorg	int	–	+	–	–	{1,2,3,4,5}

В данной таблице хранятся все данные о профилях – возраст (age), пол (gender), имя (name) и идентификатор пользователя, создавшего заказ (UserID). Также хранятся оценки пользователя предложенным типам кухонь (attorus) и т. д.

Таблица – Схожие пользователи

	Тип данных	Значение по умолчанию	Обязательность	Первичный ключ	Внешний ключ	Ограничения
SimuserID	int	–	+	+	–	–
Sim User1ID	int	–	+	–	+	уникальный
Sim User2ID	int	–	+	–	+	
Coefsimuser	double	–	+	–	–	–

В данной таблице хранятся данные о схожести пользователей (SimuserID), коэффициент схожести пользователей (Coefsimuser).

Таблица 4 – Ресторан

	Тип данных	Значение по умолчанию	Обязательность	Первичный ключ	Внешний ключ	Ограничения
RestaurantID	int	–	+	+	–	–
type	varchar(45)	–	+	–	–	–
Average_price	double	–	+	–	–	–
place	varchar(45)	–	+	–	–	–
primetime	time	–	+	–	–	–
service	int	–	+	–	–	{1,2,3,4,5}
area	double	–	+	–	–	–
music	varchar(45)	–	+	–	–	–
variety	int	–	+	–	–	{1,2,3,4,5}
entertainment	varchar(45)	–	+	–	–	–

В данной таблице хранится информация о ресторанах. Уникальный номер ресторана (RestaurantID), тип ресторана (type), средний чек ресторана (average_price), расположение (place), время работы (primetime) и тип сервиса (service), наличии музыки (music), разнообразия меню (variety).

Таблица 5 – Схожие рестораны

	Тип данных	Значение по умолчанию	Обязательность	Первичный ключ	Внешний ключ	Ограничения
SimrestID	int	–	+	+	–	–
Sim rest1ID	int	–	+	–	+	уникальный
Sim rest2ID	int	–	+	–	+	
Coefsimrest	double	–	+	–	–	–

В данной таблице хранится информация о схожести ресторанов (SimrestID), коэффициент схожести ресторанов (Coefsimrest).

Таблица 6 – Рекомендации

	Тип данных	Значение по умолчанию	Обязательность	Первичный ключ	Внешний ключ	Ограничения
RecommendationsID	int	–	+	+	–	–
UsersID	int	–	+	–	+	–
RestaurantID	int	–	+	–	+	–
Rate_num	double	–	+	–	–	–

Связывающая таблица, хранящая рекомендацию ресторана пользователю.

Таблица 7 – Рейтинг

	Тип данных	Значение по умолчанию	Обязательность	Первичный ключ	Внешний ключ	Ограничения
RatingID	int	–	+	+	–	–
UsersID	int	–	+	–	+	–
RestaurantID	int	–	+	–	+	–
rate	int	–	+	–	–	{1,2,3,4,5}

Данная таблица хранит оценку (rate) пользователя (UsersID) ресторану (RestaurantID).

3.4 Клиент-серверное приложение

Веб-сервис – это технология, обеспечивающая взаимодействие между программами на основе веб-стандартов.

Они способны взаимодействовать друг с другом, а также с другими приложениями, обмениваясь сообщениями с помощью сетевых протоколов.

Архитектура серверной части построена на языке программирования Java. Взаимодействие между клиентской и серверной частью осуществляется

посредством протокола HTTP.

Для организации обмена данными с API был выбран шаблон проектирования REST (Representational State Transfer). Он представляет собой набор принципов построения веб-службы, предназначенной для изменения или просмотра информации без обращения к серверу.

Рассмотрим инструмент, позволяющий клиенту считывать данные с серверного приложения – JSON.

JSON (от англ. JavaScript Object Notation) представляет собой основанный на JavaScript текстовый формат обмена данными. Как и большинство других текстовых форматов, JSON удобен для чтения людьми. Данный формат был разработан Дугласом Крокфордом.

Важно отметить, что JSON, будучи производным от JavaScript, тем не менее он классифицируется как независимое средство, не имеющее прямой связи с JavaScript. На сегодня разработано множество средств создания и обработки данных любого типа в формате JSON для очень многих языков программирования.

JSON-текст (в зашифрованном виде) может быть представлен одной или двумя структурами:

– в виде набора пар ключей и соответствующих им значений. Многие языки реализуют это в качестве структуры, объекта, словаря, записи, хэш-таблицы, списка с ключом или ассоциативного массива. Ключом может выступать только строка (зависящая от регистра: если два одинаковых имени имеют буквы из разных регистров, они считаются разными), а значением – любая форма;

– в виде упорядоченного набора значений. Различные языки реализуют это по-разному: вектором, массивом, списком, последовательностью.

Такие структуры данных являются универсальными, поскольку почти все современные языки программирования в той или иной форме поддерживают их. Именно поэтому они были положены в основу JSON, так как он предназначен для обмена данными между разными языками.

Значениями в JSON выступают термы.

Объектом является неупорядоченное множество пар ключей и значений, которое заключено в фигурные скобки «{ }». В качестве описания ключа обязательно выступает строка, между ней и значением располагается символ «:». Пары ключей и значений отделены друг от друга запятыми;

Массив представляет собой упорядоченное множество значений. Он заключен в квадратные скобки «[ ]». Значения внутри массива разделены при помощи запятых.

Литералы логического типа: «true», «false» и «null».

Строка представляет собой упорядоченное множество символов юникоды, количество которых может быть любым, начиная с нулевого. Это множество заключено в двойные кавычки, а входящие в него символы указываются одним из двух способов: либо в качестве escape-последовательностей, которые начинаются обратным слэшем (косой чертой) «\» (поддерживаются варианты \", \\, \/, \t, \n, \r, \f и \b), либо в виде шестнадцатеричного кода (в виде \uFFFF или по кодировке UTF-8).

Следует отметить, что строка в JSON является практически полным аналогом одноименного типа данных в языках программирования C и Java. Это же касается и числа, с тем отличием, что в JSON оно представлено только в десятеричном формате. Использование пробелов допустимо между двумя любыми элементами синтаксиса.

3.5 Общий алгоритм системы

На рисунке 15 приведена схема общего алгоритма системы.

Общий алгоритм системы начинается с инициализации приложения, т. е. с момента его загрузки на сервер. После загрузки оно начинает работать в режиме сервера и отвечать на запросы круглосуточно.

После инициализации приложение ожидает входящие запросы. Если запрос получен, то происходит его обработка, алгоритм которой приведен отдельно.

Если же запроса нет, то проверяются параметры системы. В случае сбоя приложение посылает сообщение администратору и завершает работу. В противном случае оно работает до тех пор, пока сервер не будет отключен, или приложение не будет выгружено с него.

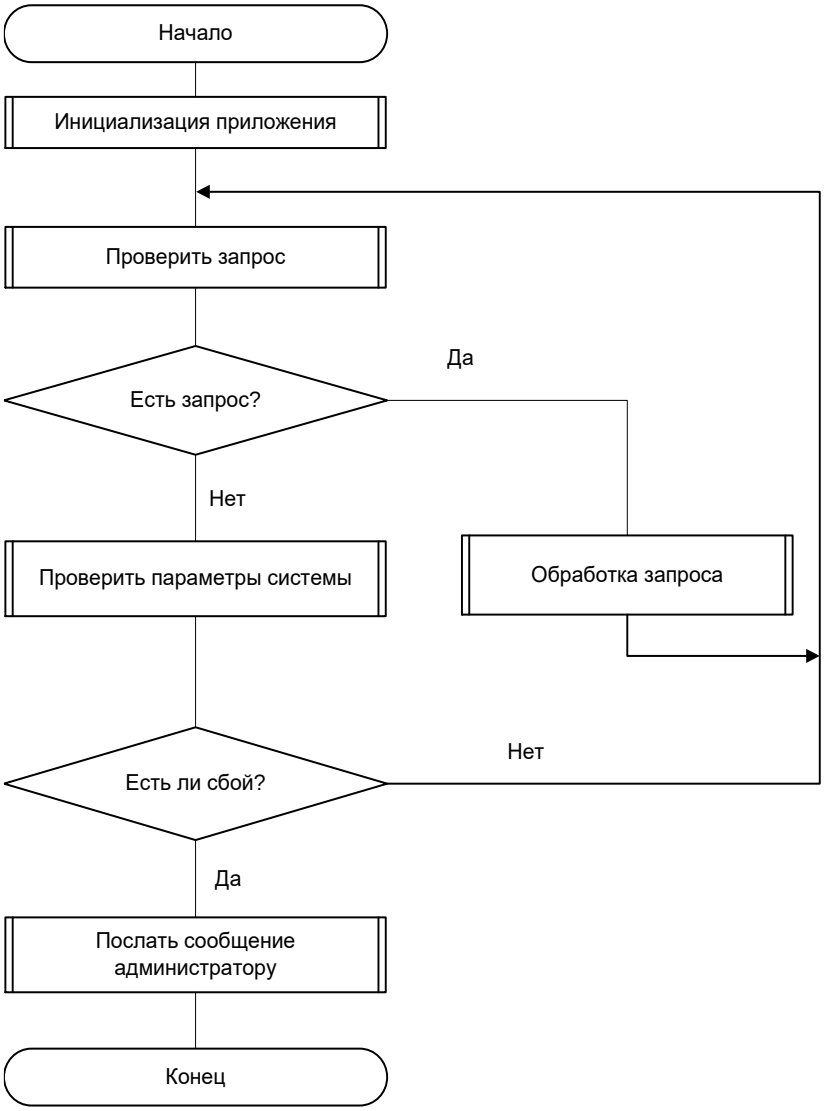

Рисунок 15 –Общий алгоритм системы

3.6 Алгоритм выполнения действия

Сначала маршрутизатор запускает представителя, отвечающего за указанный объект, то есть создает экземпляр соответствующего класса, и вызывает функцию, отвечающую за указанное действие, передавая ему в качестве аргументов все параметры, обнаруженные в запросе.

Представитель первым делом проверяет корректность введенных

данных.

Разные действия требуют разные параметры. Кроме того, многие действия имеют необязательные параметры, т. е. их отсутствие не вызовет ошибки.

Если данные введены корректно, то представитель запускает модель объекта. Модель подключается к базе данных и проверяет данные на ошибки, для обнаружения которых требуется это подключение. Например, при регистрации пользователя логин должен быть уникальным, и для подтверждения этого требуется ответ из базы данных. Если ответ отрицательный, то приложение выводит ошибку типа «Такой логин уже занят». Также при регистрации пароль должен быть не короче шести символов. При введении короткого пароля система выведет ошибку типа «Слишком короткий пароль».

При отсутствии ошибок модель отправляет запрос-действие в базу данных. Например, для той же регистрации требуется добавить в базу данных, в таблицу «Пользователь», новую запись с введенными логином, паролем пользователя.

После того, как запрос осуществлен, ответ модели зависит от действия. Так, при добавлении нового пользователя будет выведено сообщение, свидетельствующее об успешной регистрации.

Все ответы модели отправляются представителю. Все ответы представителя, созданные им или полученные от модели, передаются на модуль связи, после чего этот модуль формирует ответ, либо в числовом виде (если получен код результата) либо в формате JSON (если была получена информация из базы данных), и выводит его. Затем ответ считывается уже клиентским приложением.

3.7 Выводы к третьей главе

В данном разделе разработана архитектура системы, включающая в себя диаграмму вариантов использования и диаграмму классов.

Диаграмма прецедентов включает в себя все возможные варианты использования системы; диаграмма классов описывает структурные компоненты системы.

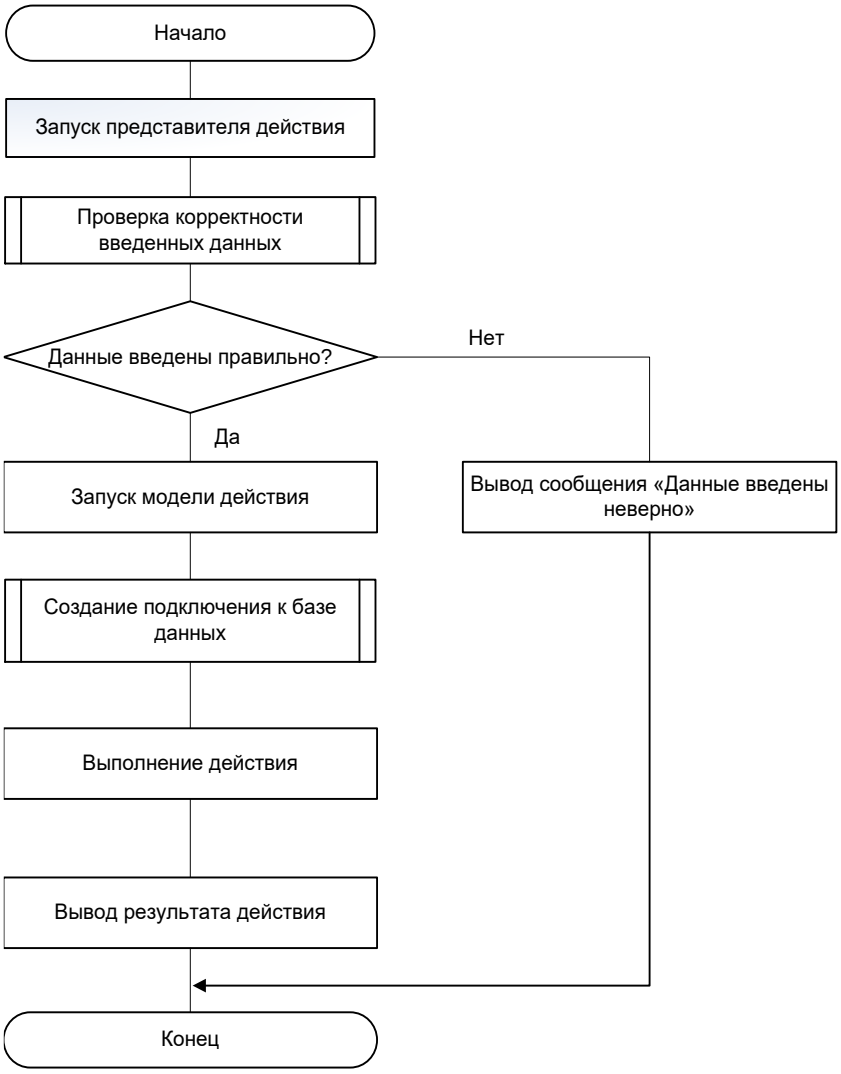

Рисунок 16 – Вспомогательный алгоритм «Выполнение действия»

Диаграммы описаны с помощью языка UML [7]. При разработке системы использован паттерн Model-View-Presenter (MVP).

Описан общий алгоритм системы и алгоритм выполнения действия.

Также разработана и построена схема реляционной базы данных и приведено описание всех таблиц и атрибутов, находящихся в ней [6].

В качестве шаблона проектирования выбран REST. Для взаимодействия мобильного приложения с веб сервисом в качестве формата данных для обмена между клиентом и сервером выбран JSON из-за его простоты обработки данных на стороне клиента и сервера.

4 ОПИСАНИЕ И ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

4.1 Тестирование системы

В процессе разработки приложения производилось поэтапное тестирование с целью выявления программных ошибок и несоответствий ТЗ. Для этого в среде разработки были созданы эмуляторы смартфона и планшета с разными диагоналями экрана для разных версий Android. Тестируемый программный продукт последовательно запускался на этих эмуляторах, его поведение анализировалось, и при необходимости по результатам анализа вносились изменения в код.

Для проверки работоспособности рекомендательной системой в базу данных было добавлено несколько ресторанов с разными видами кухонь. Также было зарегистрировано несколько пользователей, которые в профиле указали схожую оценку предложенным на выбор кухням мира. После этого все пользователи хорошими оценками оценили один ресторан. Для проверки рекомендательной системы новый пользователь в своём профиле тоже оценил некоторые предложенные кухни и на основе схожести его профиля с другими ему выдалась рекомендация того ресторана, который оценивали предыдущие пользователи.

4.2 Описание пользовательского интерфейса

На рисунке 17 изображено меню авторизации и регистрации, которое содержит кнопки, позволяющие пользователю осуществить вход в приложение или зарегистрироваться в нём. При нажатии на кнопку «Войти» пользователю выведется на экран форма авторизации, в которой ему будет необходимо ввести логин и пароль, указанные при регистрации, а при нажатии на кнопку «Зарегистрироваться» пользователю выведется форма для регистрации.

Рисунок 17 – Меню авторизации и регистрации

Чтобы иметь возможность выставлять оценки и получать рекомендации, посетителю сайта необходимо зарегистрироваться. Меню регистрации содержит поля, которые необходимо указать пользователю. В поле «Логин» пользователь указывает имя своего профиля, а в поле «Пароль» указывает

пароль, которым ему необходимо будет воспользоваться при входе.

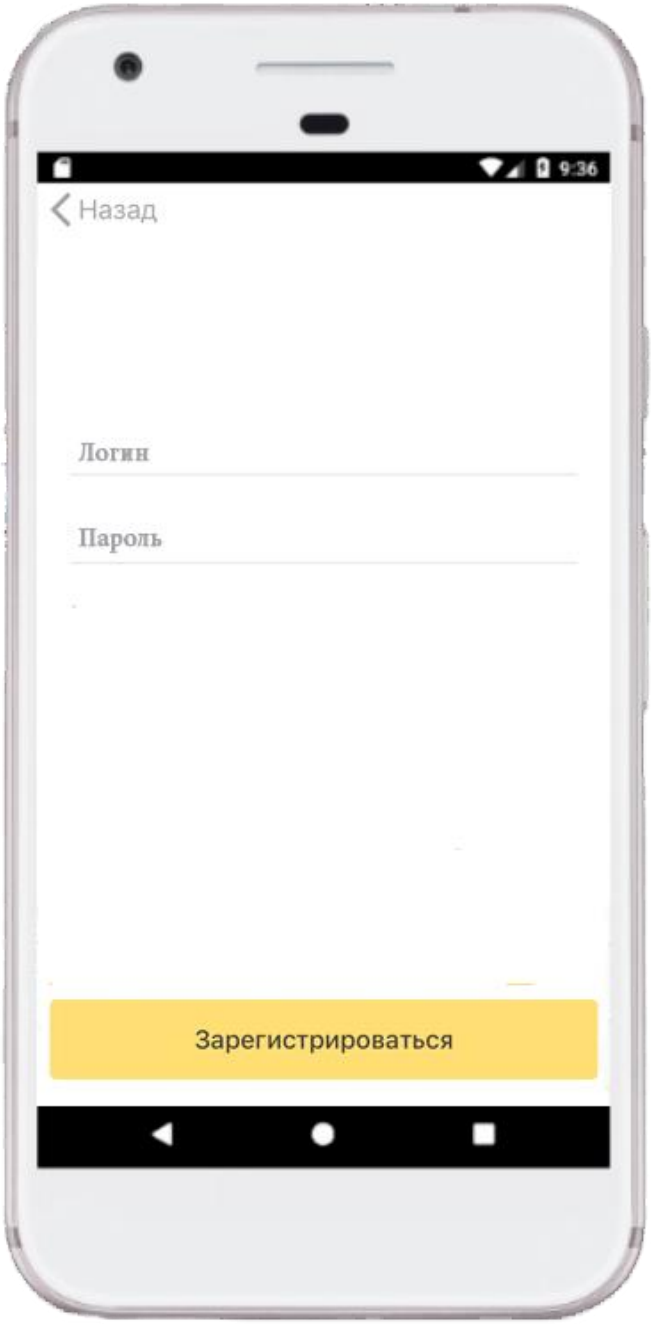

Рисунок 18 – Меню регистрации

Профиль пользователя содержит информацию о пользователе, которую он указал при регистрации. Пользователь может оценить кухни мира, предложенные ему. Таким образом будут формироваться его предпочтения и составляться рекомендации. Также здесь будут отображаться оставленные

отзывы.

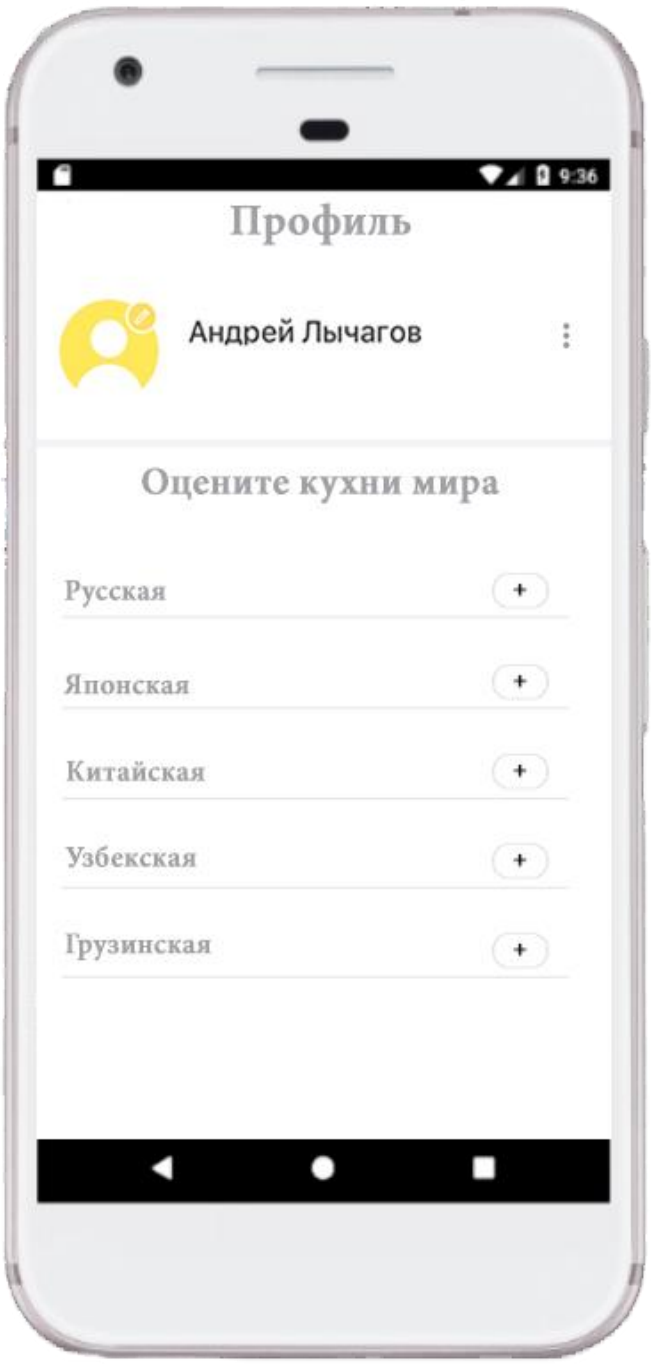

Рисунок 19 – Профиль пользователя

Меню рекомендаций показывает пользователю, какие оценки и отзывы оставляют люди, выставившие такие же критерии и предпочтения у себя в профиле, какие выставил и сам пользователь.

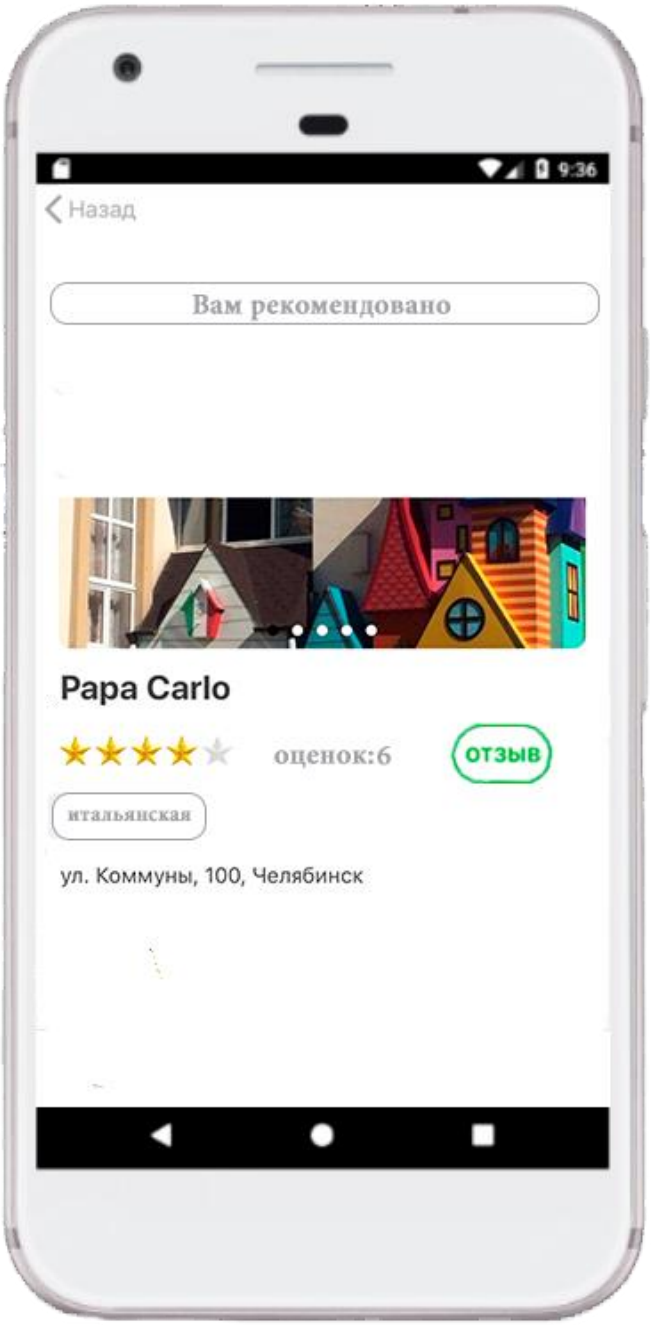

Рисунок 20 – Получение рекомендаций

У зарегистрированного пользователя есть возможность оставить отзыв о ресторане, нажав на соответствующую кнопку на экране. При нажатии откроется новое меню, в котором будет указано название ресторана и поле для написания отзыва. В поле для отзыва пользователь вводит текст, после чего при нажатии кнопки «Опубликовать» отзыв будет опубликован. В этом же

меню пользователь может оценить ресторан от одного до пяти, выставив соответствующее количество звездочек. При нажатии на кнопку «Оценить» его оценка сохранится.

Рисунок 21 – Написание отзыва

4.3 Вывод к четвёртой главе

В данном разделе описан процесс тестирования рекомендательной системы приложения, а также подробно описано меню авторизации и регистрации, профиль пользователя с предложением настройки своих предпочтений пользователем, страница получения рекомендации и добавления отзыва с оценкой.

ЗАКЛЮЧЕНИЕ

Данная работа посвящена разработке рекомендательной системы для мобильного приложения. В результате работы:

– проведен анализ методов построения рекомендательных систем;

– разработан математический алгоритм нахождения схожести пользователей и ресторанов;

– предложена оценка схожести двух векторов профилей пользователей в условиях отсутствия полной информации по параметрам;

– спроектирована база данных;

– спроектирована архитектура клиент-серверного приложения;

– разработано мобильное приложение и проведено тестирование системы;

Все поставленные задачи выполнены и все цели достигнуты.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Глибовец, Н.Н. Создание рекомендационной системы учебного типа с использованием фреймворка / Н.Н. Глибовец, М.О. Сидоренко // Проблемы интеллектуализации компьютера: сб. ст. / Институт кибернетики им. В.М. Глушкова НАН Украины. – Киев, 2012. – С. 176–181.

2. Глибовец, Н.Н. Создание рекомендационной системы учебного типа с использованием фреймворка / Н.Н. Глибовец, М.О. Сидоренко // Проблемы интеллектуализации компьютера : сб. ст. / Институт кибернетики им. В.М. Глушкова НАН Украины. – Киев, 2012. – С. 176–181.

3. Джонс, М. Рекомендательные системы: Часть 1. Введение в подходы и алгоритмы. – Дата обновления: 29.04.2014. URL: http://www.ibm.com/developerworks/ru/library/os-recommender1.html (дата обращения: 01.04.2020).

4. Джонс, М. Рекомендательные системы: Часть 2. Механизмы с открытым исходным кодом. URL: http://www.ibm.com/developerworks/ru/library/os-recommender2.html (дата обращения: 03.04.2020).

5. Жернакова, О. Системы рекомендаций и поиска видеоконтента. – Дата обновления: 01.02.2012. URL: http://www.telemultimedia.ru/art.php?id=464.html (дата обращения : 02.03.201).

6. Бейли, Л. Изучаем PHP и MySQL / Л. Бейли, М. Моррисон; пер. с англ. ЧП «Айдиономикс». – М.: Эксмо, 2010. – 800 с.

7. Буч, Г. Язык UML. Руководство пользователя. [Электронный ресурс] / Г. Буч, Д. Рамбо, И. Якобсон. – Электрон. дан. – М.: ДМК Пресс, 2008. – 496 с. – URL:http://e.lanbook.com/book/1246 (дата обращения: 15.04.2020).

8. Горнаков, С.Г. Осваиваем популярные системы управления сайтом (CMS) / С.Г. Горнаков. – М.: ДМК Пресс, 2009. – 336 с.

9. Гомзин, А.Г. Обзор рекомендательных систем и возможностей учета контекста при формировании индивидуальных рекомендаций / А.Г Гомзин // Academy – 2016 – 6(9) – С.20-22

10. Пономарев, А. В. Обзор методов учета контекста в системах коллаборативной фильтрации / А.В. Пономарев //Труды СПИИРАН. – 2013. – Т. 7. – №. 30. – С. 169-188.

11. Колисниченко, Д.Н. PHP и MySQL. Разработка веб-приложений / Д.Н. Колисниченко. – СПб. : БХВ-Петербург, 2015. – 592 с.

12. Новиков, Ф.А. Учебно-методическое пособие по дисциплине «Анализ и проектирование на UML». [Электронный ресурс] – Электрон. дан. – СПб.: НИУ ИТМО, 2007. – 286 с. – Режим доступа: http://e.lanbook.com/book/43540 – Загл. с экрана.

13. Орлов, В.В. Технологии разработки программных продуктов / В.В. Орлов. – СПб.: Питер, 2003. – 437 с.

14. Мельник, К.В. Применение аппарата Байесовых сетей при обработке данных из медицинских карточек / К.В. Мельник, В.Н. Глушко // Science and Education a New Dimension: Natural and Technical Sciences. – I(2), Issue:15, 2013.– С.126-129

15. Прохоренок, Н.А. HTML, JavaScript, PHP и MySQL. Джентельменский набор Web-мастера / Н.А. Прохоренок. – СПб.: БХВ-Петербург, 2010. – 912 с.

16. Ромашов, В. CMS Drupal: система управления содержимым сайта (+CD с видеокурсом) / В. Ромашов. – Спб. : Питер, 2010. – 256 с.

17. Сегеран, Т. Программируем коллективный разум / Т. Сегеран; пер. с англ. А. Слинкина. – Спб.: Символ-Плюс, 2008. – 368 с.

18. Recommender systems: an introduction / D. Jannach, M. Zanker, A. Felferning [et al.]. – New-York: Cambridge University Press, 2011. – 352 p.

19. Recommender Systems Handbook / F. Ricci, L. Rokach, B. Shapira [et al.]. – New-York : Springer Science+Bussiness Media, 2011. – 842 p.

20. Hybrid Recommender systems: survey and experiments. / R. Burke, 2002. – 331 p.

21. Recommender systems survey. Knowl-Based Systyms / J. Bobadilla, F. Ortega, A. Hernando, A Gutierrez. 2013. – P. 46–109.

ПРИЛОЖЕНИЕ

```
//Авторизация

LoginActivity.java

package com.example.recrest.activities.join;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import com.example.recrest.R;
import com.example.recrest.task.UserInfoTask;
import com.example.recrest.task.UserLoginTask;
import com.example.recrest.utils.ViewUtils;
import com.example.recrest.service.UserSessionService;
import com.example.recrest.web.response.ApplicationResponse;
import
   com.example.recrest.web.response.ApplicationResponse.Response
   Status;
import com.example.recrest.web.response.UserModelResponse;
import androidx.appcompat.app.AppCompatActivity;
import static
   com.example.recrest.utils.ViewUtils.showSnackBarWithButton;
import static
   com.example.recrest.web.response.ApplicationResponse.Response
   Status.SUCCESS;
import static
   com.example.recrest.web.utils.UserDataSecurityUtils.isPasswor
   dValid;
public class LoginActivity extends AppCompatActivity {
   private UserLoginTask authTask = null;
   private UserInfoTask userinfotask = null;
   private EditText passwordView;
   private View progressView;
   private View loginFormView;
   private View loginMainLayout;

   @Override

   protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_login);
   passwordView = findViewById(R.id.password);
   loginFormView = findViewById(R.id.login_form);
   progressView = findViewById(R.id.login_progress);
   loginMainLayout = findViewById(R.id.login_layout);

   View toRegistrationForm =
   findViewById(R.id.textRegistrationButton);
   toRegistrationForm.setOnClickListener(v -> {
```

```
    Intent intent = new Intent(LoginActivity.this,
    RegistrationActivity.class);
startActivity(intent);
    });
    }
    private void attemptLogin() {
    if (authTask != null) {
    return;
    }
    passwordView.setError(null);
    String password = passwordView.getText().toString();

    if (!TextUtils.isEmpty(password)
    && !isPasswordValid(password)) {

    passwordView.setError(getString(R.string.error_invalid_passwo
    rd));
    passwordView.requestFocus();
    return;
    }
    ViewUtils.showProgress(this, loginFormView, progressView,
    true);

    authTask.setOnTaskCanceledListener(this::userLoginTaskCancele
    d);

    authTask.setOnTaskFinishedListener(this::userLoginTaskFinishe
    d);
    authTask.execute((Void) null);
    }

    private void userLoginTaskFinished(ApplicationResponse<?>
    response) {
    if (isFinishing() || isDestroyed()) {
    return;
    }
    authTask = null;
    passwordView.setText("");
    ResponseStatus responseStatus = response.getResponseStatus();
    if (responseStatus == SUCCESS) {
    String token = (String) response.getData();
    UserSessionService.saveUserToken(this, token);
    userInfoTask = new UserInfoTask(token);
    userInfoTask.setOnTaskFinishedListener(this::userGet
    InfoTaskFinished);
    userInfoTask.execute((Void) null);
    } else {
    ViewUtils.showProgress(this, loginFormView, progressView,
    false);
    showSnackBarWithButton(this, loginMainLayout,
    R.string.login_error, R.string.caption_done);
    }
    }
```

```
    private void userLoginTaskCanceled() {
    if (isFinishing() || isDestroyed()) {
    return;
    }
    authTask = null;
    passwordView.setText("");
    ViewUtils.showProgress(this, loginFormView, progressView,
    false);
    }
    private void userGetInfoTaskFinished(ApplicationResponse<?>
    response) {
    if (isFinishing() || isDestroyed()) {
    return;
    }
    userInfoTask = null;
    UserModelResponse userModelResponse = (UserModelResponse)
    response.getData();
    UserSessionService.saveUserInfo(this, userModelResponse);
    finish();
    }
    }

// Регистрация

RegistrationActivity.java

package com.example.recrest.activities.join;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import com.example.recrest.R;
import com.example.recrest.model.UserRegistrationData;
import com.example.recrest.task.UserRegistrationTask;
import com.example.recrest.utils.ViewUtils;
import com.example.recrest.web.response.ApplicationResponse;
import
    com.example.recrest.web.response.ApplicationResponse.Respons-
    eStatus;
import com.example.recrest.web.utils.ResponseUtils;
import com.google.android.material.snackbar.Snackbar;
import androidx.appcompat.app.AppCompatActivity;
import static
    com.example.recrest.utils.ViewUtils.showSnackBarWithButton;
import static
    com.example.recrest.web.response.ApplicationResponse.Response
    Status.SUCCESS;
import static
    com.example.recrest.web.utils.UserDataSecurityUtils.isPasswor
    dValid;
```

```
import static
   com.google.android.material.snackbar.BaseTransientBot-
   tomBar.LENGTH_INDEFINITE;
public class RegistrationActivity extends AppCompatActivity {
private UserRegistrationTask registrationTask = null;
private EditText loginnameView;
private EditText passwordView;
private View progressView;
private View registrationFormView;
private View registrationMainLayout;

   @Override

   protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_registration);
   loginnameView = findViewById(R.id.registration_login_name);
   passwordView = findViewById(R.id.registration_password);
   registrationFormView = findViewById(R.id.registration_form);
   progressView = findViewById(R.id.registration_progress);
   registrationMainLayout =
   findViewById(R.id.registration_layout);
   View toLoginForm = findViewById(R.id.textLoginButton);
toLoginForm.setOnClickListener(v -> {
   Intent intent = new Intent(RegistrationActivity.this,
   LoginActivity.class);
   startActivity(intent);
   });
   Button registrationButton.setOnClickListener(view ->
   registration());
}
   private void registration() {
   if (registrationTask != null) {
   return;
   }

   loginnameView.setError(null);
   passwordView.setError(null);
   String loginname = loginnameView.getText().toString();
   if (TextUtils.isEmpty(loginname)) {

   loginnameView.setError(getString(R.string.error_field_require
   d));
   loginnameView.requestFocus();
   return;
   }
   String password = passwordView.getText().toString();
   if (!isPasswordValid(password)) {
passwordView.setError(getString(R.string.error_invalid_password)
   ;
   passwordView.requestFocus();
   return;
   }
```

```
    ViewUtils.showProgress(this, registrationFormView,
    progressView, true);
    UserRegistrationData userRegistrationData = new
    UserRegistrationData(loginname, password);
registrationTask = new
    UserRegistrationTask(userRegistrationData);
    registrationTask.setOnTaskFinishedListener(this::onRegistrati
    onTaskFinished);

    registrationTask.setOnTaskCanceledListener(this::onRegistrati
    onTaskCanceled);
    registrationTask.execute((Void) null);
}
    private void onRegistrationTaskCanceled() {
    if (isFinishing() || isDestroyed()) {
    return;
    }
    registrationTask = null;
    ViewUtils.showProgress(this, registrationFormView,
    progressView, false);
    }
    private void onRegistrationTaskFinished (ApplicationResponse<?>
    response) {
    if (isFinishing() || isDestroyed()) {
    return;
    }
    registrationTask = null;
    ViewUtils.showProgress(this, registrationFormView, pro-
    gressView, false);
    passwordView.setText("");
    passwordRetryView.setText("");
    ResponseStatus status = response.getResponseStatus();
    if (status == SUCCESS) {
    showSnackBarWithButton(this, registrationMainLayout,
    R.string.registration_success, R.string.caption_done);
    } else {
    Snackbar.make(registrationMainLayout,
    R.string.registration_error, LENGTH_INDEFINITE).show();
    if (codes.containsKey("password")) {
    passwordView.setError(codes.get("password"));
    } else if (codes.containsKey("loginname")) {
    loginnameView.setError(codes.get("loginname"));

    }
```