|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ  Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА**  Институт информационных технологий  Кафедра вычислительной техники |

**РАБОТА ДОПУЩЕНА К ЗАЩИТЕ**

Заведующий

кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.В. Платонова

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

по направлению подготовки бакалавров 09.03.04 Программная инженерия

На тему: рекомендательное мобильное приложение выбора товара

Обучающийся Рындык Даниил Андреевич

*подпись Фамилия, имя, отчество*

|  |  |  |
| --- | --- | --- |
| шифр | 18И1257 |  |
| группа | ИКБО-04-18 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Руководитель работы** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *подпись* | к.т.н., доцент, доцент | Унгер А.Ю. |
|  |  |  |  |
| **Консультант по экономическому разделу** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *подпись* | к.ю.н., доцент | Филаткина А.П. |

Москва 2022 г.

Аннотация

Тема выпускной квалификационной работы: рекомендательное мобильное приложение выбора товара

Информация о выпускной квалификационной работе: 67 страниц, из них основного текста 47 страниц, списка использованной литературы 2 страницы, приложений – 15. Таблиц – 6, рисунков – 15, формул - 2.

Цель работы: создание информационной системы, позволяющей формировать список возможно интересных покупателю товаров

Ключевые слова: нейронная сеть, интеллектуальная система, мобильное приложение, клиентская часть, рекомендательная система, основанные на контенте, коллаборативной фильтрации, база данных, архитектура системы.

The final qualifying work «Recommendation mobile application for product selection».

Keywords: neural network, intelligent system, mobile application, client part, content-based recommendation system, collaborative filtering, database, system architecture.

# СОДЕРЖАНИЕ

[АННОТАЦИЯ 2](#_Toc105529492)

[СОДЕРЖАНИЕ 3](#_Toc105529493)

[ВВЕДЕНИЕ 7](#_Toc105529494)

[ЛИТЕРАТУРНЫЙ ОБЗОР 9](#_Toc105529495)

[1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ 10](#_Toc105529496)

[1.1 Анализ предметной области 10](#_Toc105529497)

[1.2 Характеристика объекта исследования 10](#_Toc105529498)

[1.3 Характеристика предмета исследования 11](#_Toc105529499)

[1.4 Постановка задачи 11](#_Toc105529500)

[1.5 Разработка технического задания 12](#_Toc105529501)

[2. АНАЛИТИЧЕСКИЙ РАЗДЕЛ 15](#_Toc105529509)

[2.1 Обзор существующих решений 15](#_Toc105529510)

[2.2 Информационное обеспечение программного продукта 16](#_Toc105529511)

[2.3 Архитектура нейронной сети для формирования рекомендаций 18](#_Toc105529514)

[3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 23](#_Toc105529521)

[3.1 Разработка структуры информационной системы 23](#_Toc105529522)

[3.2 Реализация клиентской части 24](#_Toc105529523)

[3.3 Реализация серверной части 29](#_Toc105529528)

[3.4 Обоснование выбора СУБД 37](#_Toc105529532)

[3.5 Разработка структуры базы данных 38](#_Toc105529533)

[3.6 Тестирование системы 39](#_Toc105529534)

[3.7 Создание руководства пользователя 42](#_Toc105529535)

[4. ЭКОНОМИЧЕСКИЙ РАЗДЕЛ 45](#_Toc105529536)

[4.1 Организация и планирование работ по теме 45](#_Toc105529537)

[4.2 Расчет стоимости проведение работ по теме 47](#_Toc105529540)

[4.3 Вывод 50](#_Toc105529551)

[ЗАКЛЮЧЕНИЕ 51](#_Toc105529552)

[CONCLUSION 52](#_Toc105529553)

[СПИСОК ИСТОЧНИКОВ 53](#_Toc105529554)

[ПРИЛОЖЕНИЯ 55](#_Toc105529555)

# ВВЕДЕНИЕ

Темой данной выпускной квалификационной работы является рекомендательное мобильное приложение выбора товара. Эта тема подразумевает программную реализацию системы, повышающую эффективность выдачи рекомендаций пользователям на основе их предыдущего выбора.

Пользователи обычно оценивают лишь небольшую часть товаров, которые есть в каталоге, и задача системы рекомендаций — обобщить эту информацию и спрогнозировать отношение покупателей к другим неизвестным товарам, а также проинформировать пользователя о продуктах, которые его, вероятно, больше всего интересуют в данный момент.

Создание информационной системы, позволяющей формировать список возможно интересных покупателю товаров, и стало целью этой выпускной квалификационной работы.

Практическая ценность разработанной информационной системы обеспечивается возможностью создать на её основе коммерческий продукт, позволяющий получать прибыль за счет показа товара, основываясь на интересах конкретного покупателя, исключительно от заинтересованных организацией-партнеров за материальную выплату.

В процессе написания выпускной квалификационной работы автор (дипломник) руководствовался следующими нормативными актами:

1. «О защите населения и территории от чрезвычайных ситуаций природного и техногенного характера» от 21.12.1994 № 68-ФЗ.
2. «Об основах охраны здоровья граждан в Российской Федерации» от 21.11.2011 № 323-ФЗ
3. «О гражданской обороне» от 12.02.1998 № 28-ФЗ.
4. Приказ Минздравсоцразвития РФ от 04.05.2012 № 477н «Об утверждении перечня состояний, при которых оказывается первая помощь, и перечня мероприятий по оказанию первой помощи».
5. Трудовой кодекс Российской Федерации от 30.12.2001 № 197-ФЗ
6. СанПин — 2.2.2/542-96 «Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работы»

Литературный обзор

1. Ростовцев В. С. Искусственные нейронные сети. В учебнике приведены основные теоретические и практические сведения по разработке, обучению и применению искусственных нейронных сетей. Автор достаточно просто объясняет теоретические аспекты. Их знание необходимо для понимания принципов функционирования нейронных сетей и написания соответствующих программных инструкций. Расматриваются этапы – подготовки тренировочных данных, тестирование нейросети, улучшение результатов.
2. Чистова Д. В. Проектирование информационных систем. В данном пособии изложены основы проектирования, конструирования и отладки программных средств с использованием технологических и функциональных стандартов, современных моделей и методов оценки качества и надежности. Наряду с теоретическими знаниями по созданию и управлению информационными системами на всех этапах их жизненного цикла, в учебнике предусмотрена практическая часть, представляющая собой многочисленные примеры разработки проектов информационных систем.
3. Подбельский В. В. Программирование. Базовый курс С#. В этой книге изложены основные концепции и механизмы современного программирования. Методика изложения и тщательно отобранные примеры позволяют освоить не только синтаксис и семантику языка C#, но и изучить фундаментальные принципы процедурного, объектного, объектно-ориентированного и обобщенного программирования.
4. Назарова И.А., Вихрова А.С. Экономика предприятия. В учебном пособии изложены теоретические и практические основы деятельности предприятия. Дается характеристика типов собственности предприятия, его организационных форм и структуры. Рассмотрены ключевые экономические понятия, методы расчета амортизации, себестоимости, прибыли и формирования цены предприятия, оптовой цены промышленности и розничных цен.

# 1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

## 1.1 Анализ предметной области

Физиологически человеческий организм вынужден потреблять органическую пищу для обеспечения себя энергией, основным источником продуктов потребления для современного урбанизированного человека является супермаркет, который в изобилии предоставляет самые различные продовольственные товары от большого количество разнообразных конкурирующих между собой компаний. Это в свою очередь в теории должно стимулировать покупателя на потребление, что обеспечивает экономическое развитие. К сожалению, на практике это не всегда реализуемо. Человеческий мозг зачастую склонен к формированию привычек и любое их нарушение, в том числе повторный и, возможно, даже многократный анализ обыденной продуктовой корзины, может вызвать у потребителя отторжение.

Самым банальным и простым решением данной проблемы может выступить ненавязчивая рекомендация покупателю похожих на излюбленные товары повседневного потребления аналогов.

## 1.2 Характеристика объекта исследования

Объектом исследования являются нейронные сети, их проектирование и реализация.

Искусственные нейронные сети (НС) — совокупность моделей биологических нейронных сетей.

Представляют собой сеть элементов — искусственных нейронов — связанных между собой синаптическими соединениями. Сеть обрабатывает входную информацию и в процессе изменения своего состояния во времени формирует совокупность выходных сигналов.

Работа сети состоит в преобразовании входных сигналов во времени, в результате чего меняется внутреннее состояние сети и формируются выходные воздействия. Обычно НС оперирует цифровыми, а не символьными величинами.

Большинство моделей НС требуют обучения. В общем случае, обучение — такой выбор параметров сети, при котором сеть лучше всего справляется с поставленной проблемой. Обучение — это задача многомерной оптимизации, и для ее решения существует множество алгоритмов.

Искусственные нейронные сети — набор математических и алгоритмических методов для решения широкого круга задач.

## 1.3 Характеристика предмета исследования

Предметом данного исследования является использование нейронных сетей в качестве основы для рекомендательной системы.

## 1.4 Постановка задачи

Основными задачами являются:

1. Теоретическое изучение принципов работы нейронных сетей и их возможных реализаций;
2. Анализ и подбор наиболее эффективного вида нейронной сети для нашей задачи;
3. Реализация архитектуры и алгоритмов расчетов нейронной сети;
4. Обучение нейронной сети;
5. Реализация мобильного приложения;

## 1.5 Разработка технического задания

### 1.5.1 Назначение и цели разработки

В ходе данной работы будет реализована мобильная информационная система FoodAmongUs выдачи пользователю рекомендаций по продуктам и его предыдущим отзывам, основанная на базовых принципах работы нейронных сетей.

Разработанная программа помогает пользователю в выборе товаров, которые могли бы его заинтересовать. Это позволит найти более выгодные аналоги продукции и сократить траты на продуктовую корзину семьи, помимо этого, добавлять других пользователей в семейную группу и смотреть их персональные рекомендации. Все вышеописанные алгоритмы будут доступны человеку в любое время, в любом месте при наличии интернета за счет работы системы через мобильное приложение.

### 1.5.2 Основные функции, подлежащие разработке

* авторизация пользователя;
* регистрация пользователя;
* ввод данных клиента веб-клиента;
* отправка данных с веб-клиента в систему;
* отправка данных в веб-клиент из системы;
* работа с базой данных;
* обучение модели на наборе данных;
* вывод списка продуктов;
* вывод информации о продукте;
* вывод отзывов пользователей;
* добавление нового отзыва;
* расчет рекомендаций (изначально в числом виде) на основе предыдущих оценок пользователя, а также иных факторов;
* отображение человеку продуктов, соответствующих интересам, рассчитанные на предыдущем шаге, либо самых востребованных по мнению остальных пользователей товаров.

Выходные данные будут отображаться в виде текстовой и графической информации на экране мобильного устройства пользователя.

Система должна удовлетворять критериям эффективности:

* максимально допустимое время ответа системы — 10 сек;
* максимальный используемый объем оперативной памяти — 4 гб;
* возможность работы системы при низкой скорости интернета.

### 1.5.3 Требования к аппаратно-программной платформе

В состав технических средств должен входить сервер или персональный компьютер с операционной системой Windows 10 и установленным Net.Framework. Минимальные требования к аппаратной части:

* Оперативная память (ОЗУ) — объемом не менее 2 Гб;
* Объём свободного дискового пространства (SSD) — не менее 52 Гб;
* Количество ядер процессора — не менее четырех;
* Минимальная частота процессора — не менее 2 Ггц;
* Бесперебойный доступ в интернет.

### 1.5.4 Требования по экономике

Организация, планирование работ, а также расчет стоимости по теме представлены в Экономическом разделе.

### 1.5.5 Режим работы

Запуск программного обеспечения должно осуществляться на выделенном или специализированном информационном устройстве, который выполняет сервисные программы. Взаимодействие с системой выполняется за счет установки специализированного программного обеспечения.

### 1.5.6 Порядок контроля

В процессе разработки должны быть произведены модульное,интеграционное и функциональное тестирование. Все найденные ошибкиинедоработки должны быть зафиксированы и исправлены.

### 1.5.7 Требования к документированию

Требуется разработать руководство пользователя, в котором будет отражены правила эксплуатации программного обеспечения.

# 2. АНАЛИТИЧЕСКИЙ РАЗДЕЛ

## 2.1 Обзор существующих решений

Рекомендательная система — комплекс алгоритмов, программ и сервисов, задача которого предсказать, что может заинтересовать того или иного пользователя. В основе работы лежит информация о профиле человека и иные данные.

Аналогичные разрабатываемой системы можно наблюдать у «Лента», «Яндекс Дзен», «AliExpress», «WildBerries», «Яндекс-Маркет»:

1. В «Ленте» на основе выборов похожей группы покупателей выдаются товары, которые в теории могут заинтересовать пользователя.

2. В «Яндекс Дзене» выводятся статьи, по названию, описанию и типу аналогичные тем, которые часто просматривает пользователь.

3. В «AliExpress» выводятся товары, уже ранее заказанные пользователем, их аналоги, похожие продукты по тематике, стилю, типу, продавцу.

4. В «WildBerries» показываются предметы, уже просмотренные пользователем, и схожие с ними.

5. В «Яндекс-Маркете» выводятся рекомендации на основе просмотров, общей тематики, а также товары, которые по принципу применения подобны заказанным ранее.

Соответственно, разработанная в данной работе информационная система будет содержать лучшие практики и подходы к архитектуре рекомендательной системы. По аналогии с «Лентой» программа будет показывать пользователю общие трендовые и популярные товары, выборка которых основывается на предыдущих покупках. Из систем «AliExpress», «WildBerries» и «Яндекс-Маркет» будет дополнительно взят принцип рекомендаций на основе предыдущих просмотров, а также по совпадению определенных характеристик продукции.

## 2.2 Информационное обеспечение программного продукта

### 2.2.1 Выбор языка программирования

Создание мобильного приложения, как и любой другой системы, подразумевает использование языка программирования. Выбор языка обременен тем фактом, что основная часть программы, используемая для взаимодействия с пользователем, для соответствующего покрытия целевой аудитории должна запускаться на мобильных системах, самой популярной из которых является Android. Сфера реализации информационных продуктов в данной операционной системе в основном заключается в проектировании систем с использованием трех основных языков:

1. **Java** — это объектно-ориентированный язык программирования, который применяется в разработке различных программных продуктов, без четкой специализации в конкретной сфере. Вся структура строится вокруг объектов, классов, экземпляров и прочих формальных сущностей, принятых в сообществе программистов за стандарт разработки. Помимо всего прочего, Java выступает в роли платформы. Код, написанный на этом языке, запускается в виртуальной машине JVM и без проблем инициализируются в любой системе, где поддерживается соответствующая виртуальная машина.
2. **Kotlin** — это язык программирования, созданный в компании JetBrains. Его разработали в 2011 году на замену Java, который в компании считали чересчур многословным. Новый язык получился на 40% компактнее предшественника, что помогло ускорить работу над основным продуктом JetBrains — средой разработки IntelliJ IDEA. При этом Kotlin полностью совместим с Java, потому что запускается на его виртуальной машине (JVM).
3. **C#** — объектно-ориентированный язык программирования. Он был создан в период с 1998 по 2002 год командой инженеров Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота. Язык входит в семью С-подобных языков. Синтаксис приближен к Java и C++. Разработка Microsoft много особенностей унаследовала у Delphi, Smalltalk и Java. При этом создатели нового языка исключили из своего детища многие практики и спецификации, считающиеся «проблемными». Основными преимуществами данного языка можно считать:
   * нетипичные конструкции языка и специфичные конструкции, помогающие максимально органично реализовать намеченные функции;
   * популярность языка;
   * понятный синтаксис C# заметно упрощает не только разработку как таковую, но и другие важные аспекты совместной работы, например, чтение чужого кода. Это упрощает процесс рефакторинга и исправления ошибок;
   * низкий порог вхождения. С# — достаточно простая в освоении технология. Уже через полгода можно поднатореть в разработке и начать делать полноценные программы;
   * кроссплатформенность — огромное количество готовых библиотек позволяет запускать программы на большом количестве операционных систем.

Из представленных вариантов наиболее подходящим под текущие задачи является C#, во многом за счет низкого порога вхождения, кроссплатформенности и понятного синтаксиса.

### 2.2.2 Выбор среды программирования

Из самых популярных средств программирования для языка C# можно выделить:

1. **Visual Studio** — это один из наиболее популярных редакторов кода, разработанный корпорацией Microsoft. Он распространяется в бесплатном доступе и поддерживается всеми актуальными операционными системами: Windows, Linux и MAC OS.
2. **Visual Code** — представляет собой обычный текстовый редактор с возможностью подключения различных плагинов, что дает возможность работать со всевозможными языками программирования для разработки любого IT-продукта.
3. **Project Rider** — среда от JetBrains для работы с платформой .NET. Выпущена в прошлом году, но уже приобрела много поклонников. Позволяет подключить MSBuild и XBuild, работать с CLI-проектами и организовать отладку приложений .NET and Mono. Множество опций для быстрого создания кода улучшает производительность. Вышла совсем недавно, из-за чего часть стартовых ошибок еще не была исправлена, а также не имеет бесплатной версии.

Из всех представленных вариантов, Visual Studio выглядит наиболее подходящими под текущие задачи за счет наличия большого количества дополнительных плагинов и кроссплатформенности.

## 2.3 Архитектура нейронной сети для формирования рекомендаций

Нейронные сети являются одной из областей искусственного интеллекта. Основной их целью является прогнозирование и, как следствие, моделирование различных ситуаций в зависимости от исходных данных. Задачи, решаемые типичной нейронной сетью:

* классификация;
* прогнозирование;
* распознавание.

Нейронные сети часто описываются в виде слоёной структуры, где каждый слой состоит из входных, скрытых или выходных нейронов. Клетки между собой не взаимодействуют в пределах уровня, а связаны только со соседними слоями. Самая простая нейронная сеть прямого распространения имеет два входных нейрона и один выходной.

Замечательным примером биологической нейронной сети является человеческий мозг, представляющий собой многоуровневую биологическую нейронную сеть, которая получает, хранит и обрабатывает информацию от органов чувств.

У нейрона есть n входов xi, у каждого из которого есть вес wi, на который умножается число, проходящий по связи. После этого взвешенные сигналы xi⋅wi складываются и передаются на Рисунке 2.1.



Рисунок 2.1 – Схема искусственного нейрона

Без дополнительных расчетов передавать взвешенную сумму на выход из нейрона достаточно бессмысленно — необходимо информацию обработать и сформировать полноценный выходные данные. Роль преобразователя взвешенной суммы берет на себя функция активации, которая преобразует взвешенную сумму в число, которое и будет являться выходом нейрона. Функция активации обозначается ϕ (взвешенной суммы).

Самые часто используемые функции активации:

* функция единичного скачка, если взвешенная сумма> пороговое значение, возвращаем 1, иначе 0;
* сигмоидальная функция, где показывает степень крутизны функции;
* гиперболический тангенс, где показывает степень крутизны функции;
* исправление линейных единиц

Нейронные сети могут учиться и развиваться самостоятельно, накапливая опыт на основе допущенных ими ошибок.

Как основные можно выделить следующие типы нейронных сетей (Рисунок 2.2):

* нейронные сети прямого распространения;
* сети радиально-базисных функций;
* нейронная сеть Хопфилда;
* цепи Маркова;
* машина Больцмана;
* ограниченная машина Больцмана

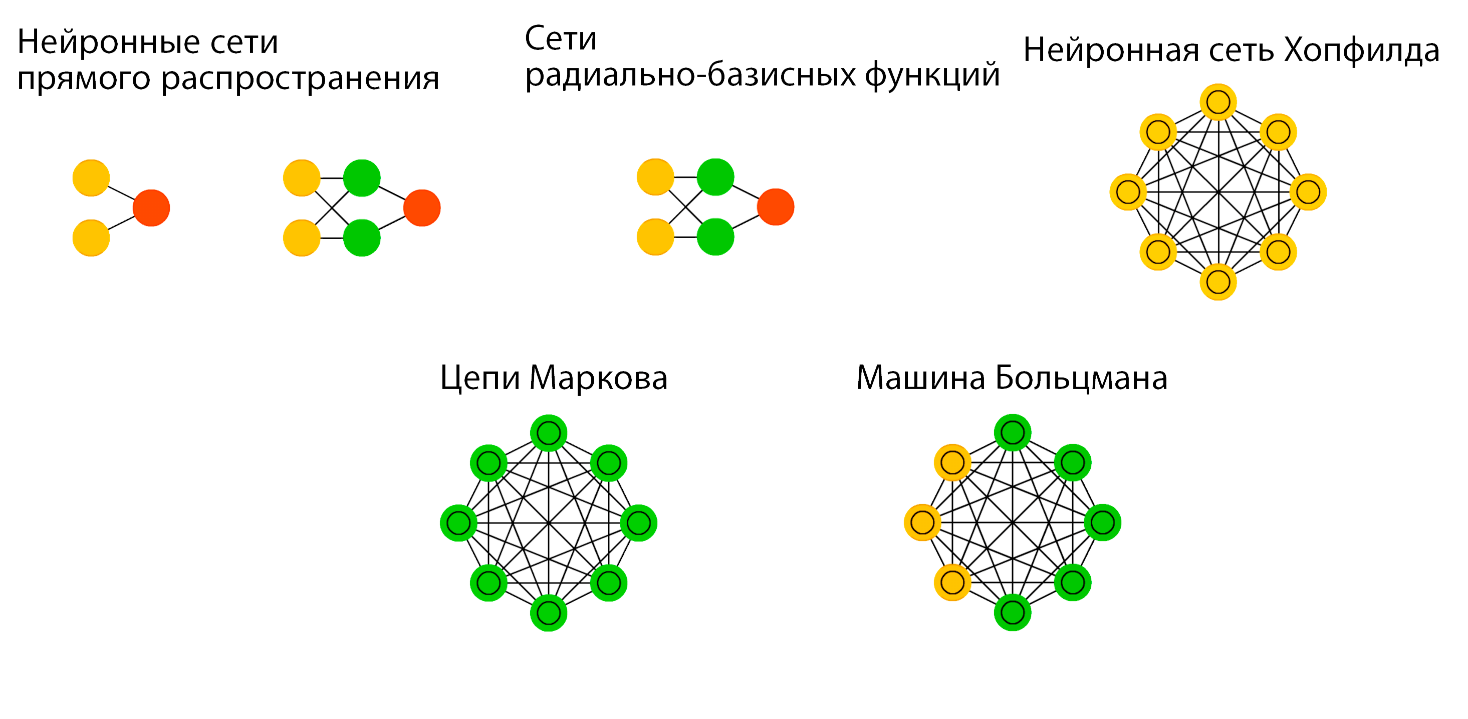


Рисунок 2.2 – Основные виды нейронных сетей

### 2.3.1 Нейронные сети прямого распространения

**Нейронные сети прямого распространения** (feed forward neural networks, FF или FFNN) и **перцептроны** (perceptrons, P) — это тип нейронных сетей, отличающийся прямой передачей информации (с учетом весов). Для обучения нейронной сети обычно используется метод обратного распространения ошибки, в котором для тренировки вносятся входные данные и ожидаемые выходные. Данный формат тренировки называется «обучение с учителем», помимо этого, существует «обучения без учителя», в качестве особенности которого можно выделить самостоятельное получение нейронной сетью выходных данных. Теоретически сеть может смоделировать структуру скрытых слоев так, чтобы получить необходимые выходные данные, на практике сети используются редко, тем не менее их часто комбинируют с другими типами.

### 2.3.2 Сети радиально-базисных функций

**RBF (**radial basis function) — сети, позволяющие нам перейти к использованию приближений функций. Сети радиальных базисных функций можно представить двумя способами:

в качестве линейной модели, в которой мы вначале произвели извлечение признаков, а признаки стали ядрами радиальных базисных функций;

в виде нейронной сети с одним скрытым слоём и радиальными базисными функциями в качестве функции активации.

### 2.3.3 Нейронная сеть Хопфилда

**Нейронная сеть Хопфилда** (Hopfield network, HN) — это нейронная сеть с симметричной матрицей связей. Во время получения входных данных каждый нейрон является входным, в процессе обучения он становится скрытым, а затем становится выходным. Обучение происходит следующим образом: значения нейронов устанавливаются в соответствии с желаемым шаблоном, вычисляются веса, которые в дальнейшем не будут меняться. После тренировки на одном или нескольких шаблонах, сеть всегда будет сводиться к одному из них. Она стабилизируется в зависимости от общего энергетического состояния всех нейронов. Каждый нейрон может находиться в одном из 2-х состояний, зачастую он может быть «заторможен» (соответствует -1), либо «возбужден» (соответствует 1). Подобные нейронные сети зачастую сравнивают с ассоциативной памятью; как, например, человек, видя половину объекта, может воссоздать вторую часть по памяти, так и эта сеть, получая таблицу, наполовину зашумленную, восстанавливает её до полной.

### 2.3.4 Цепи Маркова

**Цепи Маркова** (Markov chains, MC или discrete time Markov Chains, DTMC) — это предшественники машин Больцмана (BM) и сетей Хопфилда (HN). Общий смысл заключается в следующем: какова вероятность попасть в один из следующих нейронов, если я нахожусь в каком-то конкретном? Каждое следующее состояние зависит только от предыдущего. Хотя на самом деле цепи Маркова не являются нейронной сетью, они весьма похожи. Также цепи Маркова не обязательно полносвязны.

### 2.3.5 Машина Больцмана

**Машина Больцмана** (Boltzmann machine, BM) имеет сходство с сетью Хопфилда, но некоторые её нейроны помечены как входные, а некоторые — как скрытые. Входные нейроны в последствии становятся выходными. Машина Больцмана — это стохастическая сеть. Обучение проходит по методу обратного распространения ошибки или по алгоритму сравнительной расходимости.

### 2.3.6 Ограниченная машина Больцмана

**Ограниченная машина Больцмана (restricted Boltzmann machine, RBM)** имеет аналогичный принцип работы, что и машина Больцмана, следовательно, и основана она на сети Хопфилда. Основополагающей разницей является её ограниченность, проявляющееся в отсутствии связей между нейронами одного типа. Ограниченную машину Больцмана можно обучать методом обратного распространения ошибки, но с небольшим дополнением: вместо прямой передачи данных и обратной передачи ошибки нужно передавать данные сперва в прямом направлении, затем в обратном.

# 3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## 3.1 Разработка структуры информационной системы

Разработанную информационную систему можно условно разбить на четыре основные части (Рисунок 3.1):

1. Библиотека классов — содержит сформированный перечень классов, используемых во всех остальных модулях информационной системы (пользователь, тип продукта, продукт, отзыв).
2. Серверная часть включает в себя модуль работы с базой данных, модуль обработки запросов пользователя, модуль нейронной сети.
3. Клиентская часть включает в себя модуль отрисовки интерфейса, модуль отправки запросов к серверной части, различные модули взаимодействия с пользователем (модуль регистрации, модуль авторизации, модуль формирования рекомендаций)
4. База данных хранит группы товаров, товары, пользователи и их отзывы.

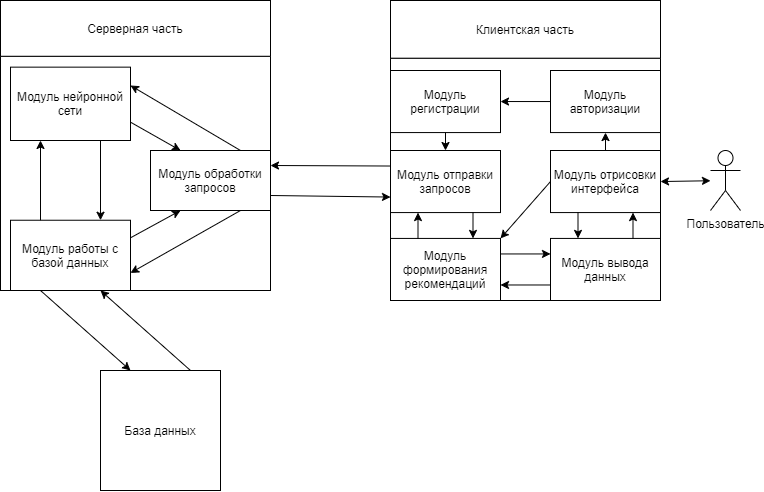


Рисунок 3.1 – Структура информационной системы

## 3.2 Реализация клиентской части

Клиентская часть состоит из следующих структурных модулей:

* модуль авторизации;
* модуль регистрации;
* модуль отправки запросов;
* модуль формирования рекомендаций.

Общая структура страниц клиентской части и их взаимодействия между собой, а также с модулями информационной системы, представлена на Рисунке 3.2.

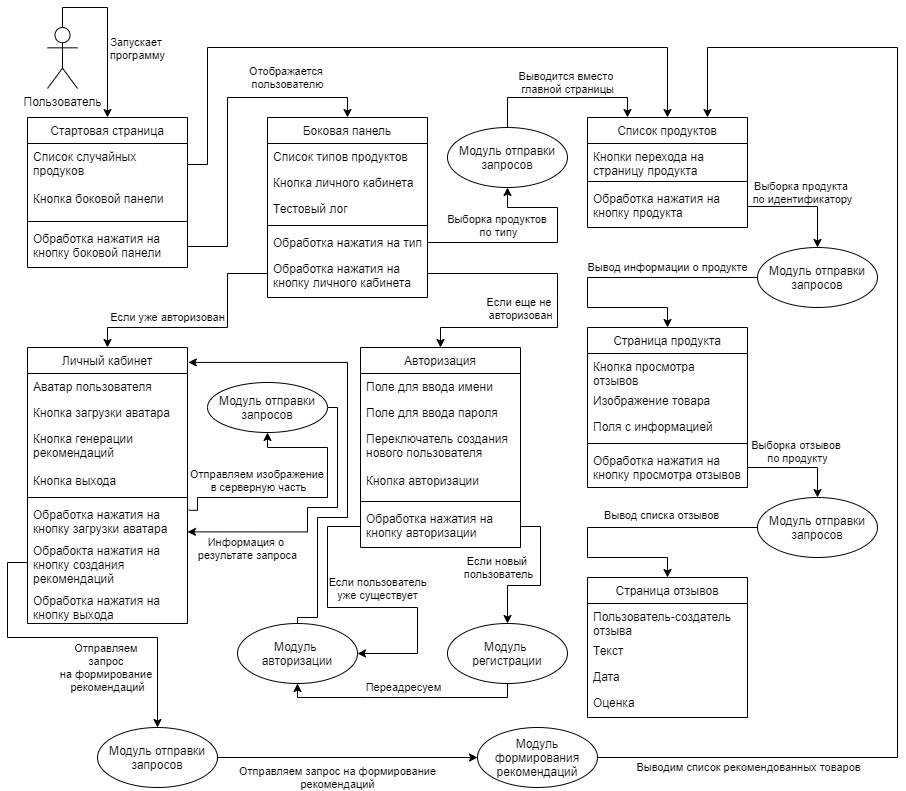


Рисунок 3.2 – Структура и взаимодействие клиентской части информационной системы

Алгоритм взаимодействия пользователя и клиентской части:

1. После запуска программы пользователь попадает на стартовую страницу, которая содержит кнопку для открытия боковой панели и список случайных продуктов.
2. Нажатие на кнопку выдвигает боковую панель со списком возможных типов товаров, помимо этого, содержит кнопку для входа в личный кабинет пользователя.
3. При взаимодействии с кнопкой определенного типа продукта, отправляется запрос на выборку соответствующих продуктов, после чего список названий товаров в виде кнопок пользовательского интерфейса выводится вместо стартовой страницы.
4. Использование любой из кнопок с названием товара вызывает отправку запроса на выборку информации о продукте, после чего выводится на странице.
5. Со страницы продукта можно перейти на страницу просмотра отзывов о соответствующем товаре, который содержит перечисление всех оценок пользователей.
6. При попытке перейти в личный кабинет пользователя впервые, происходит переадресация на страницу авторизации, где, помимо всего прочего, доступна также регистрация. После успешного входа в систему либо регистрации пользователя переносит на страницу личного кабинета.
7. На странице пользователя доступна его информация (изображение и имя), а также есть кнопки для загрузки нового аватара, формирования рекомендаций и выхода.
   * нажатие на кнопку загрузки нового изображения вызывает диалоговое окно для загрузки файла, после чего происходит отправка полученного элемента на сервер с помощью запроса, помимо этого, после формирования результата запроса, аватар в визуальном интерфейсе пользователя обновляется;
   * при взаимодействии с кнопкой формирования рекомендаций отправляется запрос на сервер с идентификатором пользователя, после чего модуль формирования рекомендаций принимает ответ, содержащий список рекомендованных товаров, и переадресует пользователя на стандартную страницу со списком товаров;
   * кнопка выхода позволяет пользователю завершить текущую сессию и авторизироваться под другим аккаунтом, если необходимо.

### 3.2.1 Модуль авторизации

Модуль авторизации получает извне имя пользователя, пароль, отправляет запрос с этими данными в серверную часть, после чего получает результат проверки – существует ли пользователь с введенными данными. Если пользователь существует, происходит переадресация на страницу личного кабинета, в противном случае ничего не происходит (Рисунок 3.3).

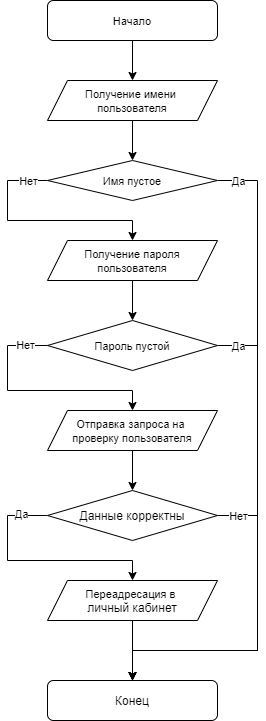


Рисунок 3.3 – Алгоритм работы модуля авторизации

### 3.2.2 Модуль регистрации

Модуль регистрации работает аналогично модулю авторизации, отличие заключается в типе отправляемого запроса. Блок получает извне имя пользователя, пароль, отправляет запрос на создание нового пользователя с этими данными в серверную часть, после чего переадресует пользователя на страницу личного кабинета (Рисунок 3.4).

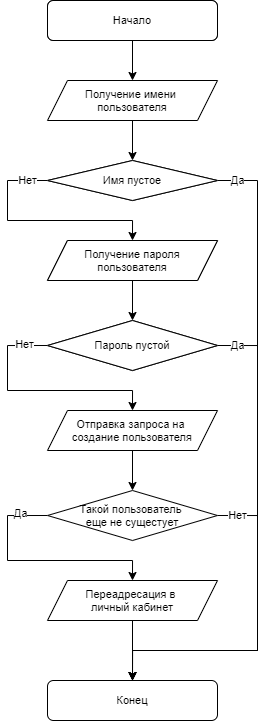


Рисунок 3.4 – Алгоритм работы модуля регистрации

### 3.2.3 Модуль отправки запросов

Модуль отправки запросов содержит большое количество разнообразных методов, необходимых другим блокам кода, отправляет их на сервер и каким-то определенным требуемым образом обрабатывает возвращаемый серверной частью результат. Среди функций данного модуля можно выделить:

* получение списка пользователей;
* формирование списка существующих типов продуктов;
* создание списка всех продуктов определенного типа;
* генерация списка рекомендаций по определенному пользователю;
* воспроизведение набора отзывов по конкретному товару;
* считывание хэш-кода пароля требуемого пользователя;
* получение статического изображения по его названию;
* формирование списка объектов любого допустимого типа из json строки;
* загрузка изображения на сервер;
* добавление нового пользователя.

Общий алгоритм формирования запросов данного модуля можно условно представить в виде следующей диаграммы (Рисунок 3.5).

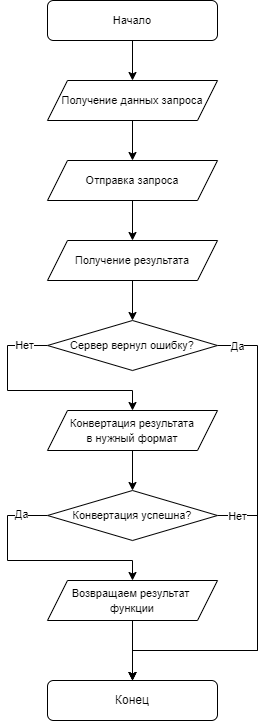


Рисунок 3.5 – Алгоритм работы модуля запросов

### 3.2.4 Модуль формирования рекомендаций

Модуль формирования рекомендаций принимает запрос пользователя (в частности, его уникальный идентификатор) на формирование списка рекомендованных товаров, после чего отправляет запрос на сервер, получает результат и отображает рекомендованный список товаров (Рисунок 3.6).

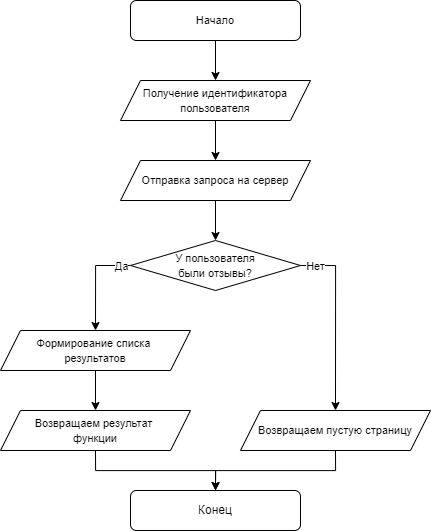


Рисунок 3.5 – Алгоритм работы модуля запросов

## 3.3 Реализация серверной части

Серверная часть состоит из следующих модулей:

* модуль работы с базой данных;
* модуль обработки запросов пользователя;
* модуль нейронной сети.

### 3.3.1 Модуль работы с базой данных

Модуль работы с базой данных отвечает за выполнение всех основных типов запросов на языке структурированных запросов SQL. Поддерживается получение списков всех представленных в базе данных объектов, загрузка изображения для пользователя, а также регистрация нового пользователя.

Получение объектов выполняется следующим образом (Рисунок 3.6):

1. В модуль базы данных приходит активация функции с необходимым типом данных.
2. В базу данных, соединенную с проектом и установленную на удаленном или локальном компьютере, приходит сформированный на основе запроса пользователя SQL запрос.
3. Результат запроса конвертируется специальными методами в список объектов по изначально заготовленным классам.
4. Список объектов, полученный на предыдущем шаге, конвертируется в JSON формат, после чего возвращается пользователю.

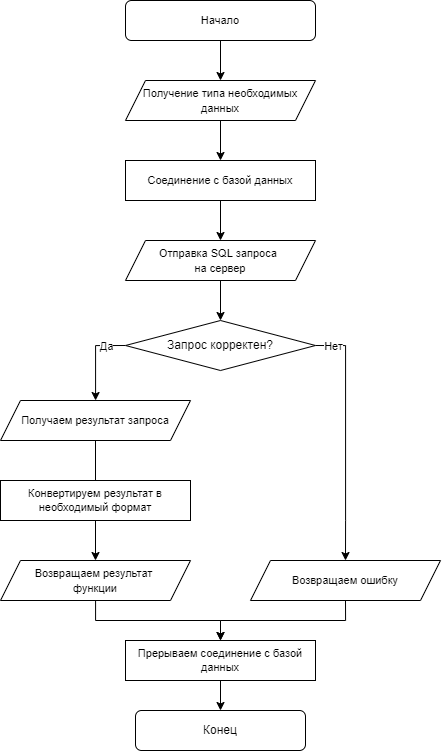


Рисунок 3.6 – Алгоритм работы модуля взаимодействия с базой данных

### 3.3.1 Модуль обработки запросов пользователя

Модуль обработки запросов пользователя принимает заголовки и данные запроса от клиентской части, определяет, какая функция была вызвана, получает из адресной строки аргументы и формирует соответствующий ответ. Общий алгоритм работы данного блока приведен далее (Рисунок 3.7).

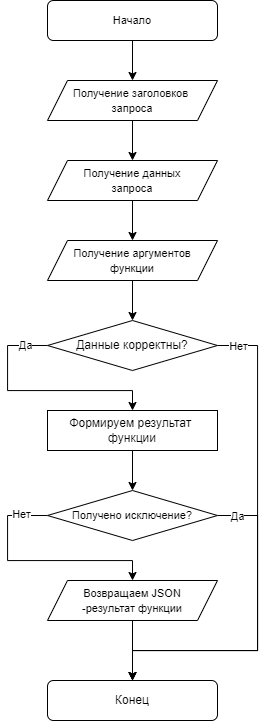


Рисунок 3.7 – Алгоритм работы модуля обработки запросов пользователя

### 3.3.1 Модуль нейронной сети

Модуль нейронной сети можно разбить на следующие основные элементы:

* модуль представления входных данных;
* модуль формирования топологии;
* модуль обучения;
* модуль генерации результата.

Общий процесс взаимодействия происходит следующим образом: серверная часть получает запрос от пользователя на формирование списка рекомендаций, после чего формируется новая нейронная сеть со стандартными весами (каждая связь равна единице), происходит обучение на основе данных товаров по уже написанным пользователем отзывам. Результатом обучения является обученная искусственная нейронная сеть, в которую помещаются все товары из базы данных, те продукты, чей выходной сигнал больше определенного порога, помещаются в список рекомендаций. По окончанию формирования данного списка, он возвращается пользователю как результат функции (Рисунок 3.8).

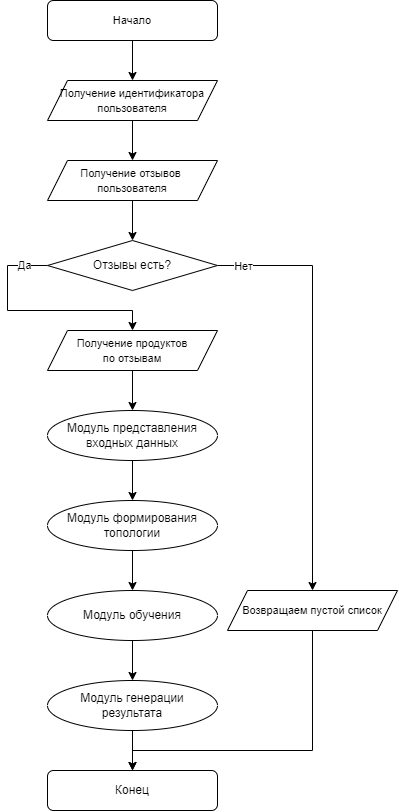


Рисунок 3.8 – Алгоритм работы модуля нейронной сети

#### 3.3.1.1 Модуль представления входных данных

Данный блок кода конвертирует входные данные (характеристики понравившихся пользователю продуктов), и передает их в модуль формирования топологии.

#### 3.3.1.2 Модуль формирования топологии

Модуль формирования топологии принимает входные данные, формирует список слоев нейронной сети (отдельно формируются входные, скрытые и результирующие слои), а также создает взаимосвязанные нейроны внутри слоев (рисунок 3.9).

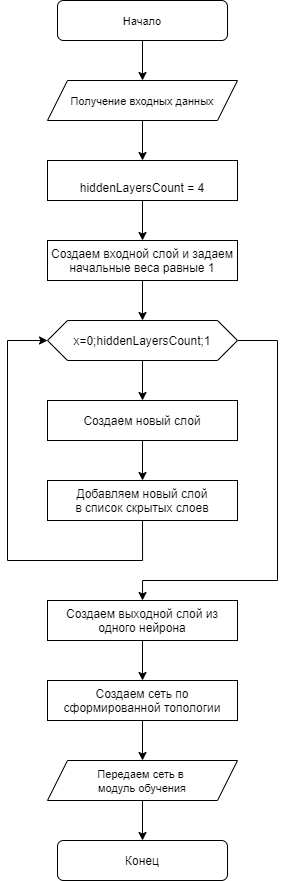


Рисунок 3.9 – Алгоритм работы модуля формирования топологии

#### 3.3.1.3 Модуль обучения

Модуль обучения получает сформированную нейронную сеть, передает в неё понравившиеся пользователю продукты и проводит обучение методом обратного распространения с использованием сигмоидальной функции активации. После обучения нейронная сеть переходит в модуль генерации результата (Рисунок 3.10).

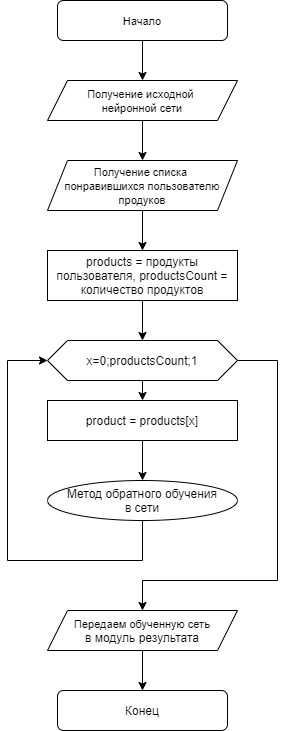


Рисунок 3.10 – Алгоритм работы модуля обучения

#### 3.3.1.4 Модуль генерации результата

Модуль генерации результата получает обученную нейронную сеть, передает в неё все продукты, которые есть в базе данных, после чего просчитывает веса по тому же принципу, что было в обучении. Значение результирующего нейрона сверяется с пороговым значением (в конечной версии оно принято за 2,5), если результат меньше порога, соответствующий товар заносится в список рекомендуемых пользователю. После проверки всех продуктов, список рекомендуемых возвращается в клиентскую часть в JSON формате (Рисунок 3.11).

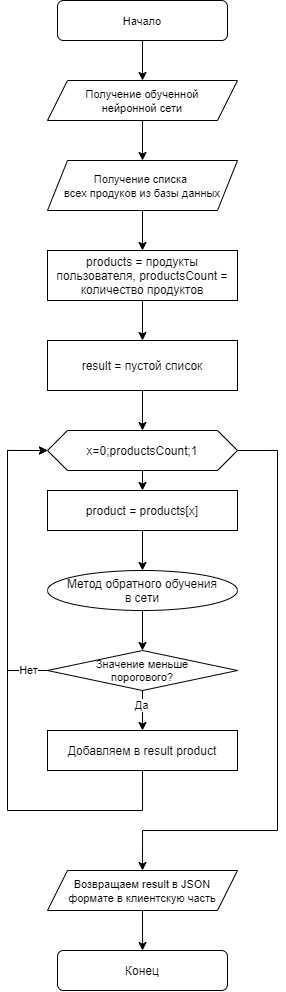


Рисунок 3.11 – Алгоритм работы модуля генерации результата

## 3.4 Обоснование выбора СУБД

Несмотря на то, что все системы управления базами данных дают возможность пользователям создавать, редактировать и получать доступ к информации, хранящейся в базах данных, сам процесс выполнения этой задачи варьируется в широких пределах. Кроме того, функции и возможности каждой СУБД могут существенно отличаться. Различные СУБД документированы по-разному: более или менее тщательно. По-разному предоставляется и техническая поддержка.

При сравнении различных популярных баз данных, следует учитывать, удобна ли для пользователя и масштабируема ли данная конкретная СУБД, а также убедиться, что она будет хорошо интегрироваться с другими продуктами, которые уже используются. Кроме того, во время выбора следует принять во внимание стоимость системы и поддержки, предоставляемой разработчиком.

Из наиболее простых в подключении к Visual Studio и C# СУБД можно выделить:

* + 1. **Microsoft Access** — это программа для создания баз данных, которая позволяет с легкостью управлять и редактировать базы данных. Она подходит буквально для всего, начиная от небольших проектов и заканчивая крупным бизнесом, она очень наглядна. Это делает ее прекрасным помощником для ввода и хранения данных, поскольку пользователю не нужно иметь дело с таблицами и графиками. Несмотря на достоинства, при подключении к классам языка программирования C# возникают трудности.
    2. **Microsoft SQL Server** — это СУБД, которая разработана специалистами из Microsoft. В качестве основного языка запросов используется Transact-SQL (совместная разработка Microsoft и Sybase). При этом Transact-SQL — это реализация стандарта ANSI/ISO по SQL (структурированному языку запросов), но имеющая некоторые расширения. Сегодня СУБД MS SQL широко применяется при работе с базами данных (БД) самых разных размеров, начиная от персональных, заканчивая крупными БД масштаба предприятия. C# имеет встроенные классы для работы с данной СУБД.
    3. **SQLite** — это встраиваемая кроссплатформенная БД, которая поддерживает достаточно полный набор команд SQL и доступна в исходных кодах (на языке C). Не имеет встроенной поддержки со стороны языка программирования.

Исходя из всего вышеописанного, единственным достойным рассмотрения вариантом является использование Microsoft SQL Server.

## 3.5 Разработка структуры базы данных

База данных состоит из следующих элементов (Рисунок 3.12):

* таблица Type содержит название и описание каждого из типов продуктов;
* таблица User описывает имя, пароль и мобильный телефон каждого из пользователей;
* таблица Item представляет разнообразные свойства продуктов, их названия, типы, описание и фото;
* таблица Review показывает все обзоры пользователей, в т.ч. оценки, дату, текст и заголовок.

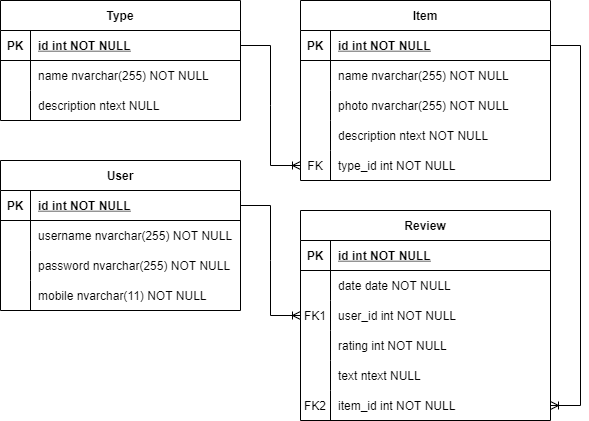


Рисунок 3.12 – Структура базы данных

## 3.6 Тестирование системы

Количество ошибок в программе — это фактор, волнующий каждого программиста. Основополагающим принципом тестирования как методологии является принцип кучкования ошибок, согласно которому обнаружение ошибок в определенной части программы увеличивает вероятность возникновения иных неполадок программного кода в этом же модуле. Безусловно, определить точное количество ошибок в программном обеспечении невозможно, но предугадать вероятность их возникновения и построить модель вполне реально. В данной работе использовалась статическая модель Миллса.

В 1972 г. программист фирмы IBM Харлан Миллс сформировал следующий подход к определению количества программных ошибок в программе: предположим в некоторой программе существует N естественных ошибок. Необходимо добавить в информационную систему дополнительно M искусственных ошибок, после чего провести тестирование программы. Допустим, в ходе тестирования было обнаружено n естественных ошибок и m искусственных, в таком случае выполняется соотношение:

Допущением является тот факт, что вероятность обнаружения естественных и искусственных ошибок одинакова.

Тестируя разработанную систему методом Миллса, было внесено 6 искусственных ошибок, в ходе тестирования было обнаружено 2 искусственных ошибки и 4 естественные ошибки. Получена следующая оценка количества проблемных мест в программе:

Количество необнаруженных ошибок:

Описанный выше способ обладает существенным недостатком: обнаружение 100% искусственных ошибок означает, что и естественных ошибок мы нашли 100%. Но возникает следующая проблема: чем меньше будет внесено искусственных ошибок, тем больше вероятность того, что они все будут найдём. Например, добавление единственной искусственной ошибки и её обнаружение позволяет сделать вердикт, что все естественные ошибки были найдены, что не является правдой. Для решение данного вопроса Миллс добавил вторую часть модели, предназначенную для проверки гипотезы о величине N:

Предположим, что в программе N естественных ошибок, вносится M искусственных ошибок, программа тестируется до тех пор, пока не будут найдены все искусственные ошибки. Допустим, к этому моменту найдено n естественных ошибок. На основании полученных результатов вычислим величину коэффициента C:

Величина C отображает меру доверия к модели и с точки зрения математической модели обозначает вероятность правильного отклонения моделью ложных предположений.

Пример: допустим, естественных ошибок в программе нет (N=0), было внесено 4 искусственные ошибки, проводится тестирование программы до момента, пока не будут обнаружены все ошибки, например, при это мы не обнаружим ни одной естественной ошибки. В данном случае мера доверия нашему предположению будет равна:

Чтобы довести данный показатель до 90%, количество искусственных ошибок придется поднять до 9. Следующие 5% уверенности в отсутствии естественных ошибок обойдутся нам в 10 дополнительных искусственных ошибок (M = 19).

Если мы предположим, что в программе не более 3-х естественных ошибок (N=3), внесем в нее 6 искусственных (M=6), найдем все искусственные и одну, две или три естественных, то мера доверия к модели будет:

Для разработанной в ходе данной выпускной квалификационной работы информационной системы показатель C будет равен:

Модель Миллса достаточно проста, несмотря на основной свои недостаток — предположение о равновероятности нахождения ошибок и требование второй части модели, заключающееся в необходимости отыскать непременно все искусственные ошибки, что может в определенных ситуациях никогда не случиться.

## 3.7 Создание руководства пользователя

Разработанная информационная система предоставляет визуальную и текстовую информацию пользователю о товарах, которые потенциально могут быть ему интересны.

Для обеспечения комфорта пользователя и надежности реализовано:

* разработано мобильное приложение (клиентская часть информационной системы) с удобным пользовательским интерфейсом, что позволит разобраться в программном обеспечении даже человеку, который не владеет в полной мере навыками взаимодействия с компьютерами;
* каждая страница содержит интуитивные элементы управления, маршрутизация между формами выполнена максимально просто, найти необходимую страницу не составляет труда;
* дизайн приложения не оказывает большой нагрузки на смартфон, что позволяет делать работу в системе более комфортной.
* программа отображает список продуктов из базы данных и позволяет переходить между страницами;
* приложение позволяет повторно зайти на любую из доступных страниц;
* информационная система позволяет пользователю авторизироваться и использовать предназначенный исключительно для него функционал;
* использование программы требует наличие интернета, в случае его отсутствия выводится соответствующее окно предупреждения;
* формирование рекомендаций возможно только в том случае, если у пользователя есть какие-либо отзывы.

Руководство пользователя представлено в Таблице 3.1.

Таблица 3.1 – руководство пользователя

|  |  |  |
| --- | --- | --- |
| Операция | Действие пользователя | Действие программы |
| Установка приложения | Установка приложения происходит с помощью запуска соответствующего APK файла | Появится иконка программы на рабочем столе |
| Запуск приложения | Для запуска приложения пользователь активирует ярлык приложения | Приложение открывается в полноэкранном режиме, проверяется наличие соединения, в случае если оно есть, отображается главная страница, иначе показывается окно «нет соединения» |
| Переход на страницу с товарами определенного типа | Пользователь выбирает тип продукта в боковом меню и нажимает на кнопку | Приложение посылает запрос на получение продуктов нужного типа, очищает элементы управления на странице и выводит кнопки с названиями продуктов |

Продолжение таблицы 3.1

|  |  |  |
| --- | --- | --- |
| Операция | Действие пользователя | Действие программы |
| Переход на страницу с информацией товара | Пользователь выбирает определенный товар и нажимает на кнопку | Программа отправляет запрос на выборку информации о товаре и выводит на отдельной странице |
| Переход на страницу с отзывами | Пользователь нажимает на кнопку «Показать отзывы» | Система формирует список отзывов по товару и выводит в заготовленном виде |
| Переход на страницу авторизации | Пользователь нажимает на кнопку «Личный кабинет» | Приложение открывает окно с вводом данных, если пользователь не авторизован, иначе переадресует в кабинет |
| Авторизация | Пользователь вводит свои логин и пароль, после чего жмет на кнопку «Зайти» | Программа проверяет корректность введенных данных, после либо выводит сообщение, либо переадресует в кабинет |
| Загрузка аватара | Пользователь нажимает на кнопку «Загрузить фото» и прикрепляет файл | Система отправляет файл на сервер и отмечает в базе смену аватара |
| Формирование рекомендаций | Пользователь нажимает кнопку «Показать рекомендации» | Приложение отправляет запрос на получение рекомендаций, после чего выводит их списком кнопок на отдельной странице |
| Выход | Пользователь нажимает на кнопку «Выход» | Программа удаляет из оперативной памяти данные пользователя, после чего он считается неавторизованным |

# 4. ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

## 4.1 Организация и планирование работ по теме

В составе работы задействовано 3 человека:

руководитель Унгер Антон Юрьевич, к.т.н. доцент, кафедра ВТ — отвечает за грамотную постановку задачи, контролирует отдельные этапы работы, вносит необходимые коррективы и оценивает выполненную работу в целом;

консультант Филаткина Анна Павловна, доцент, кафедра экономики — отвечает за консультирование экономической части выпускной квалификационной работы;

разработчик Рындык Даниил Андреевич, группа ИКБО-04-18 — реализация всех поставленных задач, в том числе проведение тестирования готового продукта и подготовка проектной документации.

Состав задействованных в работе участников представлен на схеме.

Консультант

Разработчик

Руководитель

### 4.1.1 Организация работ

На разработку отводится 90 рабочих дней.

Этапы разработки представлены в Таблице 4.1.

Таблица 4.1 – Этапы разработки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название этапа | Исполнитель | Трудоемкость,  чел/дни | Продолжительность работ, дни |
| 1 | Разработка и утверждение технического задания | Руководитель | 5 | 5 |
| Разработчик | 5 |
| 2 | Технические предложения | Руководитель | 3 | 3 |
| Консультант | 1 |
| Разработчик | 3 |

Продолжение Таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название этапа | Исполнитель | Трудоемкость,  чел/дни | Продолжительность работ, дни |
| 3 | Эскизный проект: |  |  | 11 |
| 3.1 | Анализ исходных данных и требований | Разработчик | 4 |
| 3.2 | Постановка задачи | Разработчик | 1 |
| 3.3 | Разработка общего описания алгоритма | Руководитель | 2 |
| Разработчик | 4 |
| 4 | Технический проект: |  |  | 15 |
| 4.1 | Определение формы представления входных и выходных данных | Руководитель | 2 |
| Разработчик | 5 |
| 4.2 | Разработка структуры программы | Руководитель | 2 |
| Разработчик | 10 |
| 5 | Рабочий проект: |  |  | 49 |
| 5.1 | Установка и настройка необходимого программного обеспечения | Разработчик | 1 |
| 5.2 | Реализация базы данных | Разработчик | 7 |
| 5.3 | Реализация серверной части | Разработчик | 6 |
| 5.4 | Реализация пользовательского интерфейса | Разработчик | 24 |
| 5.5 | Покупка необходимого для тестирования аппаратного обеспечения | Разработчик | 1 |
| 5.6 | Испытание программы | Разработчик | 2 |
| 5.7 | Корректировка программы по результатам испытаний | Разработчик | 1 |
| 5.8 | Подготовка технической документации на программный продукт | Консультант | 1 |
| Разработчик | 1 |
| 5.9 | Сдача готового продукта и внедрение | Руководитель | 2 | 5 |
| Консультант | 1 |
| Разработчик | 2 |
| Итого | | | | 83 |

### 4.1.2 График проведения работ

Календарный график исполнения работы представлен на Таблице 4.2. Из Таблицы 4.2 так же видно, что общий срок разработки составит 83 дня.

Таблица 4.2 – График исполнения работы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Этапы |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Дни | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 | 65 | 70 | 75 | 80 | 85 | 90 |

## 4.2 Расчет стоимости проведение работ по теме

В выпускной квалификационной работе объем затрат на НИР и ОКР был проведен методом калькулирования.

### 4.2.1 Материалы, покупные изделия и полуфабрикаты

Расходы на материалы представлены в Таблице 4.3.

Таблица 4.3 – Расходы на материалы, покупные изделия и полуфабрикаты

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ пп** | **Наименование**  **материалов** | **Единицы измерения** | **Количество** | **Цена за единицу (руб)** | **Стоимость (руб)** |
| 1 | Смартфон BQ 4030G Nice Mini, золотой | шт | 1 | 3749 | 3749 |
| 2 | USB кабель - Micro USB Exployd EX-K-480 Classic, 2.0м, чёрный | шт | 1 | 95 | 95 |
| 3 | Wi-Fi роутер Mercusys MW301R, белый | шт | 1 | 729 | 729 |
| 4 | Сетевое зарядное устройство Borofone BA19A Nimble, 5 Вт, white | шт | 1 | 230 | 230 |

Продолжение Таблицы 4.3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | Карта памяти Perfeo microSD 4GB (Cl10) | шт | 1 | 285 | 285 |
| 6 | Бумага SvetoCopy A4 Classic 80 г/м², 500 л белая | шт | 1 | 440 | 440 |
| 7 | STAFF Ручка шариковая C-51, 1.0 мм (BP109), BP109 | шт | 10 | 14 | 140 |
| 8 | STAFF Папка-скоросшиватель А4, полипропилен 100/120 мкм | шт | 2 | 27 | 54 |
| 9 | Дырокол BRAUBERG Office Expert 35 листов | шт | 1 | 111 | 111 |
| 10 | Перезаписываемый диск SmartBuy DVD-RW 4,7Gb 4x в бумажном конверте с окном | шт | 1 | 220 | 220 |
| Итого материалов | | | | | 6053 |
| Транспортно-заготовительные расходы | | | | | 907 |
| Итого | | | | | 6960 |

### 4.2.2 Специальное оборудование

Расходы по данному разделу отсутствуют

### 4.2.3 Основная заработная плата

Расчет основной заработанной платы указан в Таблице 4.4.

Таблица 4.4 – Расчет основной заработанной платы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№ пп** | **Наименование этапа** | **Исполнитель (должность)** | **Мес. оклад (руб)** | **Трудоемкость (чел/дни)** | **Оплата за день (руб)** | **Оплата за этап (руб)** |
| 1 | ТЗ | Руководитель | 39006 | 5 | 1773 | 8865 |
| Консультант | 36003 | 1 | 1636,5 | 1636,5 |
| Разработчик | 26004 | 5 | 1182 | 5910 |
| 2 | ТП | Руководитель | 39006 | 3 | 1773 | 5319 |
| Консультант | 36003 | 1 | 1636,5 | 1636,5 |
| Разработчик | 26004 | 3 | 1182 | 3546 |

Продолжение Таблицы 4.4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | Эскизный проект | Руководитель | 39006 | 2 | 1773 | 3546 |
| Консультант | 36003 | 0 | 1636,5 | 0 |
| Разработчик | 26004 | 9 | 1182 | 10638 |
| 4 | Технический проект | Руководитель | 39006 | 4 | 1773 | 7092 |
| Консультант | 36003 | 0 | 1636,5 | 1636,5 |
| Разработчик | 26004 | 15 | 1182 | 17730 |
| 5 | Рабочий проект | Руководитель | 39006 | 2 | 1773 | 3546 |
| Консультант | 36003 | 2 | 1636,5 | 3273 |
| Разработчик | 26004 | 49 | 1182 | 57918 |
| Итого | | | | | | 123427,5 |

### 4.2.4 Дополнительная заработная плата

Дополнительная заработная плата научного и производственного персонала составляет по проекту 24685,5 руб.

### 4.2.5 Страховые отчисления

Сумма страховых отчислений составляет 44433,9 руб.

### 4.2.6 Командировочные расходы

Расходы по данному разделу отсутствуют.

### 4.2.7 Контрагентские услуги

Стоимость использования виртуального хостинга в ООО «Регистратор доменных имён РЕГ.РУ» для установки серверной части и проведения основной работы составляет 256 рублей в месяц. Количество месяцев, в течение которых сервер стоял на хостинге, – 2 месяца. В сумме оплата услуг по использованию хостинга составляет 512 рублей.

В сумме расходы на контрагентские услуги составляет 512 руб.

### 4.2.8 Накладные расходы

Сумма накладных расходов составляет 246855 руб.

### 4.2.9 Прочие расходы

По статье «прочие расходы» затрат нет.

### 4.2.10 Полная себестоимость проекта

Таблица 4.5 – Полная себестоимость проекта

|  |  |  |
| --- | --- | --- |
| **№ пп** | **Номенклатура статей расходов** | **Затраты (руб)** |
| 1 | Материалы, покупные изделия и полуфабрикаты (за вычетом отходов) | 6333 |
| 2 | Специальное оборудование для научных (экспериментальных) работ | - |
| 3 | Основная заработная плата научного и производственного персонала | 123427,5 |
| 4 | Дополнительная заработная плата научного и производственного персонала | 24685,5 |
| 5 | Страховые взносы в социальные фонды | 44433,9 |
| 6 | Расходы на научные и производственные командировки | - |
| 7 | Оплата работ, выполненных сторонними организациями и предприятиями | 512 |
| 8 | Накладные расходы | 246855 |
| 9 | Прочие прямые расходы | - |
| Итого | | 446246,9 |

Полученная информационная система будет в дальнейшем реализована, поэтому цена конечного продукта составляет (с учетом НДС) 669370,35 руб.

## 4.3 Вывод

В этом разделе был осуществлён расчёт всех затрат для реализации проекта информационной системы рекомендательной системы на основе нейронной сети.

# ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе было спроектировано и реализовано рекомендательное мобильное приложение выбора товара. Все требования к системе, указанные в техническом задании, были соблюдены.

Актуальность темы дипломного проекта не вызывает сомнений, поскольку рекомендательная система позволяет повышать прибыль компаний за счет показа пользователю интересующих его товаров, что стимулирует трату денежных средств.

Дипломная работа состоит из четырех частей: исследовательского, аналитического, технологического и экономического разделов.

В первой главе формируется понятие предметной области, объекта и предмета исследования, задачи и технического задания.

Во второй главе сравниваются существующие решения, формируется информационное обеспечение программного продукта, анализируется понятие нейронной сети и её структура.

Третья часть диплома посвящена разработке системы, в том числе его серверной и клиентской частей. Более детально описаны алгоритмы взаимодействия, различные модули программы, обоснование выбора СУБД, реализация структуры базы данных, тестирование, а также сформировано руководство пользователя.

Заключительная четвертая глава содержит обоснование экономической целесообразности проекта, расчеты эффективности проекта и иных показателей.

Разработанная информационная система представляет собой максимально простой и удобный интерфейс для формирования рекомендаций пользователю. Главным преимуществом системы является возможность работы из любой точки мира, стабильность и высокая производительность.

Таким образом, все поставленные задачи выполнены, а цель работы достигнута.

# CONCLUSION

In the final qualifying work, a recommendation mobile product selection application was designed and implemented. All the system requirements specified in the terms of reference have been met.

The relevance of the topic of the diploma project is beyond doubt, since the recommendation system allows companies to increase profits by showing the user the goods they are interested in, which encourages spending money.

The thesis consists of four parts: research, analytical, technological and economic sections.

In the first chapter, the concept of the subject area, the object and subject of research, the task and the technical task is formed.

In the second chapter, existing solutions are compared, the information support of the software product is formed, the concept of a neural network and its structure are analyzed.

The third part of the diploma is devoted to the development of the system, including its server and client parts. Interaction algorithms, various modules of the program, the rationale for choosing a DBMS, the implementation of the database structure, testing are described in more detail, and a user manual is formed.

The final fourth chapter contains the justification of the economic feasibility of the project, calculations of the effectiveness of the project and other indicators.

The developed information system is the simplest and most convenient interface for making recommendations to the user. The main advantage of the system is the ability to work from anywhere in the world, stability and high performance.

Thus, all the tasks have been completed, and the goal of the work has been achieved.

# СПИСОК ИСТОЧНИКОВ

1. Гавриленко Т. Ю., Григоренко О. В., Ткаченко Е. К. Методические рекомендации по выполнению организационно-экономической части выпускных квалификационных работ: метод. указания — М.: РТУ МИРЭА, 2019. — 43 с.
2. Назарова И.А., Вихрова А.С. Экономика предприятия: учебно-методическое пособие — М.: РТУ МИРЭА, 2021. — 71 с.
3. Григоренко О.В., Садовничая И.О., Мыльникова А. Экономика предприятия и управление организацией — М.: РУСАЙНС, 2017. — 235 с.
4. Подбельский В. В. Программирование. Базовый курс С# — М.: Издательство Юрайт, 2022. — 369 с. — URL: <https://urait.ru/bcode/475775>.
5. Зыков С. В. Программирование: учебник и практикум для вузов — М.: Издательство Юрайт, 2022. — 320 с. — URL: <https://urait.ru/bcode/489754>.
6. Казанский А. А. Программирование на Visual C#: учебное пособие для вузов — М.: Издательство Юрайт, 2022. — 192 с. — URL: <https://urait.ru/bcode/470261>.
7. Зыков С. В. Программирование. Объектно-ориентированный подход: учебник и практикум для вузов — М.: Издательство Юрайт, 2022. — 155 с. — URL: <https://urait.ru/bcode/490423>.
8. Чистова Д. В. Проектирование информационных систем: учебник и практикум для вузов — М.: Издательство Юрайт, 2022. — 258 с. — URL: <https://urait.ru/bcode/489307>.
9. Грекул В. И. Проектирование информационных систем: учебник и практикум для вузов — М.: Издательство Юрайт, 2022. — 385 с. — URL: <https://urait.ru/bcode/489918>.
10. Гутгарц Р. Д. Проектирование автоматизированных систем обработки информации и управления: учебное пособие для вузов — М.: Издательство Юрайт, 2022. — 304 с. — URL: <https://urait.ru/bcode/494408>.
11. Волкова В. Н. Теория информационных процессов и систем: учебник и практикум для вузов — М.: Издательство Юрайт, 2022. — 432 с. — URL: <https://urait.ru/bcode/489220>.
12. Стружкин Н. П., Годин В. В. Базы данных: проектирование: учебник для вузов — М.: Издательство Юрайт, 2022. — 477 с. — URL: <https://urait.ru/bcode/489099>.
13. Соколова В. В. Вычислительная техника и информационные технологии. Разработка мобильных приложений: учебное пособие для вузов — М.: Издательство Юрайт, 2022. — 175 с. — URL: <https://urait.ru/bcode/490305>.
14. Данилов В. В. Нейронные сети: учебное пособие — Донецк: ДонНУ, 2020. — 158 с. — URL: <https://e.lanbook.com/book/179953>.
15. Ростовцев В. С. Искусственные нейронные сети: учебник для вузов — СПб.: Лань, 2021. — 216 с. — URL: <https://e.lanbook.com/book/160142>.
16. Фальк К. Рекомендательные системы на практике: руководство — М.: ДМК Пресс, 2020. — 448 с. — URL: <https://e.lanbook.com/book/179458>.
17. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение — М.: ДМК Пресс, 2018. — 652 с. — URL: <https://e.lanbook.com/book/107901>.
18. Морозова Ю. В. Тестирование программного обеспечения: учебное пособие – Томск: Эль-Контент — 2019. — 120 с. — URL: <https://znanium.com/catalog/product/1845910>.
19. Савчук, В. П. Управление прибылью и бюджетирование: учебное пособие — М.: Лаборатория знаний, 2020. — 435 с. — URL: <https://znanium.com/catalog/product/1094823>.

# ПРИЛОЖЕНИЯ

Приложение А – Исходный код клиентской системы

Приложение Б – Исходный код серверного модуля

Приложение В – Исходный код нейронной сети

**Приложение А**

Исходный код клиентской системы

*Листинг А.1 — Исходный код клиентской системы*

|  |
| --- |
| using Newtonsoft.Json;  using RecommendationSystem.Models;  using System;  using System.Collections;  using System.Collections.Generic;  using System.IO;  using System.Net;  using System.Text;  using UnityEngine;  using UnityEngine.Networking;  using UnityEngine.UI;  public class Database: MonoBehaviour  {  [SerializeField] private string localHost = @"http://localhost:44381";  [SerializeField] private string internetHost = @"http://u1674941.plsk.regruhosting.ru";  [SerializeField] private bool isLocal = false;  private string host = "";  private void Start()  {  if (isLocal)  {  host = localHost; }  else  {  host = internetHost; } }  [SerializeField] public bool isDebug = true;  [SerializeField] private GameObject noConnectionFrame;  public List<User> Users => GetObjects<User>("users");  public List<RecommendationSystem.Models.Type> Types => GetObjects<RecommendationSystem.Models.Type>("types");  public List<Item> GetItemsByType(int typeId) => GetObjects<Item>($"itemsbytype/?type\_id={typeId}");  public List<Item> GetItemsByUser(int userId) => GetObjects<Item>($"getrecommend?user\_id={userId}");  return text; } |

*Листинг А.2 — Продолжение листинга А.1*

|  |
| --- |
| public List<Review> GetReviewsByItem(Item item) => GetObjects<Review>($"reviewsbyitem/?item\_id={item.Id}");  public User GetUserByReview(int reviewId) => GetObject<User>($"userbyreview/?review\_id={reviewId}");  public string GetUserHash(int user\_id) => GetRequest($"getuserhash?user\_id={user\_id}");  public IEnumerator GetImageByName(string name, Image image)  {  string path = $"{host}/static/{name}";  UnityWebRequest request = UnityWebRequest.Get(path);  DownloadHandlerTexture downloader = new();  request.downloadHandler = downloader;  request.certificateHandler = new ForceAcceptAll();  Log($"Попытка загрузки файла по пути {path}");  request.SendWebRequest();  while (!request.isDone)  yield return null;  var texture = downloader.texture;  image.sprite = Sprite.Create(texture, new Rect(0.0f, 0.0f, texture.width, texture.height), new Vector2(0.5f, 0.5f), 100.0f);  Log($"Картинка {texture.name} успешно загружена"); }  private string GetRequest(string path, string method = "GET") {  try {  string finalPath = $"{host}/api/{path}";  Log($"Отправлен запрос по пути: {finalPath}");  var request = WebRequest.Create(finalPath) as HttpWebRequest;  ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls12;  request.ServerCertificateValidationCallback = delegate { return true; };  request.Method = method;  request.ContentType = @"text/html; charset=windows-1251";  request.UserAgent = "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.5 (KHTML, like Gecko) Chrome/4.0.249.89 Safari/532.5";  request.Timeout = 5000;  HttpWebResponse resp = request.GetResponse() as HttpWebResponse;  string text = "";  using (StreamReader sr = new(resp.GetResponseStream(), Encoding.Default))  text = sr.ReadToEnd();  text = text.Trim(new char[] { '"' });  SwitchFrame(true); |

*Листинг А.3 — Продолжение листинга А.2*

|  |
| --- |
| catch (Exception ex)  {  SwitchFrame(false);  throw new Exception($"Не удалось считать по пути '{path}', причина: '{ex.Message}, {ex.StackTrace}'"); } }  public List<T> GetObjects<T>(string query) where T : Model  {  try  {  string text = GetRequest(query);  var raw = JsonConvert.DeserializeObject<dynamic>(text);  List<T> result = new();  foreach (var data in raw.Children())  {  var parameters = new object[1];  parameters[0] = data;  result.Add((T)Activator.CreateInstance(typeof(T), parameters)); }  return result; }  catch (Exception ex)  {  throw new Exception($"Не удалось считать {typeof(T).Name}, причина {ex.Message}"); } }  public T GetObject<T>(string query) where T : Model {  Try {  string text = GetRequest(query);  var raw = JsonConvert.DeserializeObject<dynamic>(text);  var parameters = new object[1];  parameters[0] = raw.Last;  return (T)Activator.CreateInstance(typeof(T), parameters); }  catch (Exception ex) {  throw new Exception($"Не удалось считать {typeof(T).Name}, причина {ex.Message}"); } }  public void SetImage(string url, Image image) => StartCoroutine(GetImageByName(url, image));  public void UploadTexture(int id) => StartCoroutine(UploadTextureRoutine(id));  public void AddUser(string username, string password)  {  string allPath = $"createuser?username={username}&password={password}"; |

*Листинг А.4 — Продолжение листинга А.3*

|  |
| --- |
| Log($"Отправлен запрос по пути: {allPath}");  GetRequest(allPath); }  private IEnumerator UploadTextureRoutine(int id)  {  Texture2D texture = null;  string filePath = "";  NativeGallery.Permission permission = NativeGallery.GetImageFromGallery((path) =>  {  filePath = path;  if (isDebug)  Debug.Log("Image path: " + path);  if (path != null)  {  texture = NativeGallery.LoadImageAtPath(path, -1, false);  if (texture == null)  {  Debug.Log("Couldn't load texture from " + path);  return; } } });  if (texture != null)  {  var bytes = texture.EncodeToJPG();  List<IMultipartFormSection> form = new();  form.Add(new MultipartFormFileSection("files", bytes, "test.jpeg", "image/jpeg"));  form.Add(new MultipartFormDataSection("id", $"{id}"));  using (var unityWebRequest = UnityWebRequest.Post($"{host}/api/sendavatar", form))  {  var cert = new ForceAcceptAll();  unityWebRequest.certificateHandler = cert;  yield return unityWebRequest.SendWebRequest();  if (unityWebRequest.result != UnityWebRequest.Result.Success)  {  print($"Failed to upload {texture.name}: {unityWebRequest.result} - {unityWebRequest.error}"); }  else  {  print($"Finished Uploading {texture.name}"); } } } } |

*Листинг А.5 — Продолжение листинга А.4*

|  |
| --- |
| private void SwitchFrame(bool isConnected) => noConnectionFrame.SetActive(!isConnected);  private void Log(string text)  {  if (isDebug)  {  Debug.Log($"[LOG|{DateTime.Now:HH:mm:ss}] {text}"); } }  public class ForceAcceptAll : CertificateHandler  {  protected override bool ValidateCertificate(byte[] certificateData)  {  return true; } } } |

**Приложение Б**

Исходный код серверного модуля

*Листинг Б.1 — Исходный код серверного модуля*

|  |
| --- |
| using Microsoft.AspNetCore.Http;  using Microsoft.AspNetCore.Mvc;  using Newtonsoft.Json;  using RecommendationSystem.Core.Helpers;  using RecommendationSystem.Models;  using RecommendationSystem.Neural;  using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Net;  using System.Security.Cryptography;  using System.Text;  namespace RecommendationSystem.Controllers {  public class ApiController : Controller  {  private double neuralOutputLimit = 2.5;  [HttpGet]  public IActionResult Index()  {  Response.StatusCode = (int)HttpStatusCode.OK;  return Json(new { Message = "Its starting page. Use GET/POST requests to use data" }); }  [HttpGet]  public IActionResult Error(string message)  {  Response.StatusCode = (int)HttpStatusCode.BadRequest;  return Json(new { ErrorMessage = message }); }  [HttpGet]  public IActionResult Users() => Ok<User>();  [HttpGet]  public IActionResult UserByReview(string review\_id)  {  var review = Database.GetObject<Review>($"[id] = {review\_id}").FirstOrDefault();  if (review is not null) |

*Листинг Б.2 — Продолжение листинга Б.1*

|  |
| --- |
| {  var user = Database.GetJson<User>($"[id] = {review.AuthorId}");  return Ok(user); }  return Redirect("api/error?message=user doesn't exists"); }  [HttpGet("api/getrecommend")]  public IActionResult GetRecommend(string user\_id)  {  var reviews = Database.GetObject<Review>($"[user\_id] = {user\_id}").Where(x => x.Rating == 4 || x.Rating == 5);  if (reviews.Any())  {  Topology topology = new(11, 1, 4);  NeuronNetwork network = new(topology);  var items = Database.GetObject<Item>();  var maxAveragePrice = reviews.Select(x => x.GetItem(items)).Average(x => x.AveragePrice);  var maxTypeId = reviews.Select(x => x.GetItem(items)).Average(x => x.TypeId) \* 0.4;  foreach (Review review in reviews)  {  var item = review.GetItem(Database.GetObject<Item>());  List<double> signals = GetSignals(item, maxAveragePrice, maxTypeId);  network.FeedForward(signals); }  var outputItems = new List<Item>();  foreach (Item item in items) {  var networkCopy = network.Clone() as NeuronNetwork;  List<double> signals = GetSignals(item, maxAveragePrice, maxTypeId);  var neuron = networkCopy.FeedForward(signals);  if (neuron.Output < neuralOutputLimit) {  outputItems.Add(item); } }  return Ok(Database.GetJson(outputItems)); }  else {  return Ok(new List<Item>()); } }  [HttpGet]  public IActionResult Types() => Ok<Models.Type>();  [HttpGet]  public IActionResult ItemsByType(string type\_id) => Ok<Item>($"[type\_id] = {type\_id}");  [HttpGet] |

*Листинг Б.3 — Продолжение листинга Б.2*

|  |
| --- |
| public IActionResult Items() => Ok<Item>();  [HttpGet]  public IActionResult Reviews() => Ok<Review>();  [HttpGet]  public IActionResult ReviewsByItem(string item\_id) => Ok<Review>($"[item\_id] = {item\_id}");  [HttpGet]  public IActionResult GetUserByName(string name) => Ok<User>($"[username] = '{name}'");  [HttpGet]  public IActionResult GetUserHash(string user\_id)  {  var users = Database.GetObject<User>();  int userId = int.Parse(user\_id);  var user = users.Where(x => x.Id == userId).FirstOrDefault();  return Ok(user.Password); }  [HttpGet("api/createuser")]  public IActionResult CreateUser(string username, string password)  {  try  {  var users = Database.GetObject<User>();  var names = users.Select(x => x.Name);  if (!names.Contains(username))  {  Database.AddUser(username, Encrypt(password));  return Json(new { Message = "User added" }); }  else  {  return Json(new { Message = "User already exists" }); } }  catch (Exception ex) {  return Redirect($"api/error?message={ex.Message}"); } }  [HttpGet]  public IActionResult CheckUserHash(string user\_id, string password) {  Try {  var users = Database.GetObject<User>();  int userId = int.Parse(user\_id);  var user = users.Where(x => x.Id == userId).FirstOrDefault();  var hash = Encrypt(password); |

*Листинг Б.4 — Продолжение листинга Б.3*

|  |
| --- |
| var result = new List<User>();  if (hash == user.Password)  {  result.Add(user); }  else  {  result.Add(new User()); }  return Ok(JsonConvert.SerializeObject(result)); }  catch (Exception ex)  {  return Redirect($"api/error?message={ex.Message}"); } }  [HttpPost]  [Route("api/sendavatar")]  public IActionResult SendAvatar(IFormFile files, string id)  {  var path = Environment.CurrentDirectory;  try  {  if (files.Length > 0)  {  string staticPath = $"{path}/static";  if (!Directory.Exists(staticPath))  {  Directory.CreateDirectory(staticPath);  }  string filePath = $"{staticPath}/{files.FileName}";  if (System.IO.File.Exists(filePath))  return Ok($"File uplodaded");  using (FileStream fileStream = System.IO.File.Create(filePath)) {  files.CopyTo(fileStream);  fileStream.Flush(); }  Database.UploadAvatar(files.FileName, int.Parse(id));  Response.StatusCode = 200;  return Ok($"File uplodaded"); }  else {  Response.StatusCode = 422;  return Redirect("api/error?message=no one files"); } }  catch (Exception ex) {  Response.StatusCode = 500; |

*Листинг Б.5 — Продолжение листинга Б.4*

|  |
| --- |
| return Redirect($"api/error?message={ex.Message}"); } }  private ActionResult Ok<T>(string where = "") where T: Model => Ok(Database.GetJson<T>(where));  private static int ToInt(bool value)  {  return value switch  {  true => 1,  \_ => 0, }; }  private static List<double> GetSignals(Item item, double averagePrice, double maxTypeId)  {  return new List<double>  {  ToInt(item.IsEdibility),  item.AveragePrice / averagePrice,  ToInt(item.IsWithSugar),  ToInt(item.IsHypoallergenic),  ToInt(item.EcoFriendly),  ToInt(item.IsImport),  ToInt(item.IsAntibacterial),  ToInt(item.IsNonGMO),  ToInt(item.IsVegan),  ToInt(item.IsLean) }; }  public static string Encrypt(string value)  {  var crypt = new SHA256Managed();  var hash = new StringBuilder();  byte[] crypto = crypt.ComputeHash(Encoding.UTF8.GetBytes(value));  foreach (byte theByte in crypto)  {  hash.Append(theByte.ToString("x2")); }  return hash.ToString(); } } } |

**Приложение В**

Исходный код нейронной сети

*Листинг В.1 — Исходный код нейронной сети*

|  |
| --- |
| using System;  using System.Collections.Generic;  namespace RecommendationSystem.Neural {  public class Neuron: ICloneable {  public List<double> Weights { get; }  public NeuronType NeuronType { get; }  public double Output { get; private set; }  public Neuron(int inputCount, NeuronType type = NeuronType.Hidden) {  NeuronType = type;  Weights = new List<double>(inputCount);  for (int i = 0; i < inputCount; i++) {  Weights.Add(0.3); } }  private double Sigma(double x) => 1 / (1 - Math.Pow(Math.E, -x));  public double FeedForward(List<double> inputs) {  double sum = 0;  for (int i = 0; i < inputs.Count; i++)  {  sum += inputs[i] \* Weights[i]; }  if (NeuronType == NeuronType.Input)  {  Output = sum; }  else  {  var sigma = Sigma(sum);  Output = double.IsInfinity(sigma) ? 0 : sigma; }  return Output; }  public void SetWeights(params double[] weights)  {  for (int i = 0; i < weights.Length; i++)  {  Weights[i] = weights[i]; } }  public override string ToString() => $"{Output}";  public object Clone()  {  var neuron = new Neuron(Weights.Count, NeuronType);  neuron.SetWeights(Weights.ToArray());  return neuron; } } } |

*Листинг В.2 — Продолжение листинга В.1*

|  |
| --- |
| using RecommendationSystem.Neural;  using System;  using System.Collections.Generic;  namespace RecommendationSystem.Neural  {  public class Layer: ICloneable  {  public List<Neuron> Neurons { get; }  public int Count => Neurons?.Count ?? 0;  public Layer(List<Neuron> neurons, NeuronType type = NeuronType.Hidden)  {  foreach (Neuron neuron in neurons)  {  if (neuron.NeuronType != type)  {  throw new System.Exception("Попытка добавить нейрон неправильного типа"); } }  Neurons = neurons;  }  public List<double> GetSignals()  {  var result = new List<double>();  foreach (Neuron neuron in Neurons)  {  result.Add(neuron.Output);  }  return result;  }  public object Clone()  {  var neurons = new List<Neuron>();  foreach (Neuron oldNeuron in Neurons)  {  var neuron = oldNeuron.Clone() as Neuron;  neurons.Add(neuron);  }  return new Layer(neurons, neurons[0].NeuronType);  } }  }SwitchFrame(true); |

*Листинг В.3 — Продолжение листинга В.2*

|  |
| --- |
| using RecommendationSystem.Neural;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  namespace RecommendationSystem.Neural {  public class NeuronNetwork: ICloneable {  public Topology Topology { get; }  public List<Layer> Layers { get; set; }  public NeuronNetwork(Topology topology) {  Topology = topology;  Layers = new List<Layer>();  CreateInputLayer();  CreateHiddenLayer();  CreateOutputLayer(); }  public Neuron FeedForward(List<double> inputSignals) {  SendSignalsToInputNeurons(inputSignals);  FeedForwardAllLayers();  if (Topology.OutputCount == 1) {  return Layers.Last().Neurons[0]; }  else {  return Layers.Last().Neurons.OrderByDescending(x => x.Output).First(); } }  private void FeedForwardAllLayers() {  for (int i = 1; i < Layers.Count; i++) {  var previousLayerSignals = Layers[i - 1].GetSignals();  var layer = Layers[i];  foreach (var neuron in layer.Neurons)  {  neuron.FeedForward(previousLayerSignals); } } }  private void SendSignalsToInputNeurons(List<double> inputSignals)  {  for (int i = 0; i < inputSignals.Count; i++)  {  var signal = new List<double> { inputSignals[i] };  var neuron = Layers[0].Neurons[i];  neuron.FeedForward(signal); } }  private void CreateOutputLayer()  {  var neurons = new List<Neuron>(); |

*Листинг В.4 — Продолжение листинга В.3*

|  |
| --- |
| var lastLayer = Layers.Last();  for (int i = 0; i < Topology.InputCount; i++)  {  var neuron = new Neuron(lastLayer.Count, NeuronType.Input);  neurons.Add(neuron); }  var outputLayer = new Layer(neurons, NeuronType.Input);  Layers.Add(outputLayer); }  private void CreateHiddenLayer()  {  for (int j = 0; j < Topology.HiddenLayers.Count; j++)  {  var neurons = new List<Neuron>();  var lastLayer = Layers.Last();  for (int i = 0; i < Topology.HiddenLayers[j]; i++)  {  var neuron = new Neuron(lastLayer.Count);  neurons.Add(neuron); }  var outputLayer = new Layer(neurons);  Layers.Add(outputLayer); } }  private void CreateInputLayer()  {  var neurons = new List<Neuron>();  for (int i = 0; i < Topology.InputCount; i++)  {  var neuron = new Neuron(1, NeuronType.Input);  neurons.Add(neuron); }  var layer = new Layer(neurons, NeuronType.Input);  Layers.Add(layer); }  public object Clone()  {  var topology = Topology.Clone() as Topology;  var network = new NeuronNetwork(topology);  var layers = new List<Layer>();  foreach (Layer layer in Layers)  {  layers.Add(layer.Clone() as Layer);  }  network.Layers = layers;  return network; } } } |

*Листинг В.5 — Продолжение листинга В.4*

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  namespace RecommendationSystem.Neural  {  public class Topology: ICloneable  {  public int InputCount { get; }  public int OutputCount { get; }  public List<int> HiddenLayers { get; }  public Topology(int inputCount, int outputCount, params int[] layers)  {  InputCount = inputCount;  OutputCount = outputCount;  HiddenLayers = new List<int>();  HiddenLayers.AddRange(layers);  }  public object Clone()  {  int[] list = new int[HiddenLayers.Count];  HiddenLayers.CopyTo(list);  return new Topology(InputCount, OutputCount, list);  }  }  } |