|  |
| --- |
|  |
| 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.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc104981099)

[1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ 5](#_Toc104981100)

[1.1 Анализ предметной области 5](#_Toc104981101)

[1.2 Характеристика объекта исследования 5](#_Toc104981102)

[1.3 Характеристика предмета исследования 6](#_Toc104981103)

[1.4 Постановка задачи 6](#_Toc104981104)

[1.5 Разработка технического задания 7](#_Toc104981105)

[2. АНАЛИТИЧЕСКИЙ РАЗДЕЛ 10](#_Toc104981113)

[2.1 Обзор существующих решений 10](#_Toc104981114)

[2.2 Информационное обеспечение программного продукта 11](#_Toc104981115)

[2.3 Архитектура нейронной сети для формирования рекомендаций 14](#_Toc104981116)

[3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 19](#_Toc104981122)

[3.1 Разработка структуры информационной системы 19](#_Toc104981123)

[3.2 Реализация клиентской части 20](#_Toc104981124)

[3.3 Реализация серверной части 25](#_Toc104981129)

[3.4 Обоснование выбора СУБД 33](#_Toc104981133)

[3.5 Разработка структуры базы данных 33](#_Toc104981134)

[3.6 Тестирование системы 33](#_Toc104981135)

[3.7 Создание руководства пользователя 36](#_Toc104981136)

[4. ЭКОНОМИЧЕСКИЙ РАЗДЕЛ 37](#_Toc104981137)

[4.1 Организация и планирование работ по теме 37](#_Toc104981138)

[4.2 Расчет стоимости проведение работ по теме 39](#_Toc104981141)

[4.3 Вывод 42](#_Toc104981152)

[ЗАКЛЮЧЕНИЕ 43](#_Toc104981153)

[СПИСОК ИСТОЧНИКОВ 44](#_Toc104981154)

[ПРИЛОЖЕНИЯ 46](#_Toc104981155)

# 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»

# 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 си-стема будет содержать лучшие практики и подходы к архитектуре рекомен-дательной системы. По аналогии с «Лентой» программа будет показывать пользователю общие трендовые и популярные товары, выборка которых ос-новывается на предыдущих покупках. Из систем «AliExpress», «WildBerries» и «Яндекс-Маркет» будет дополнительно взят принцип рекомендаций на ос-нове предыдущих просмотров, а также по совпадению определенных харак-теристик 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:

* **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.

* **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).
* **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:

* **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.
* **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.
* **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.

### 2.2.3 Выбор системы управления базами данных

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

* **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.
* **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.
* **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.

## 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:

* это 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.



Рисунок 2.1 – Схема искусственного нейрона

Как видно на рисунке 1.1, 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 далее.

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 (далее w – взвешенная сумма):

* 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:

1. Нейронные сети прямого распространения (feed forward neural networks, FF или FFNN)
2. Сети радиально-базисных функций (radial basis function, RBF)
3. Нейронная сеть Хопфилда (Hopfield network, HN)
4. Цепи Маркова (Markov chains, MC или discrete time Markov Chains, DTMC)
5. Машина Больцмана (Boltzmann machine, BM)

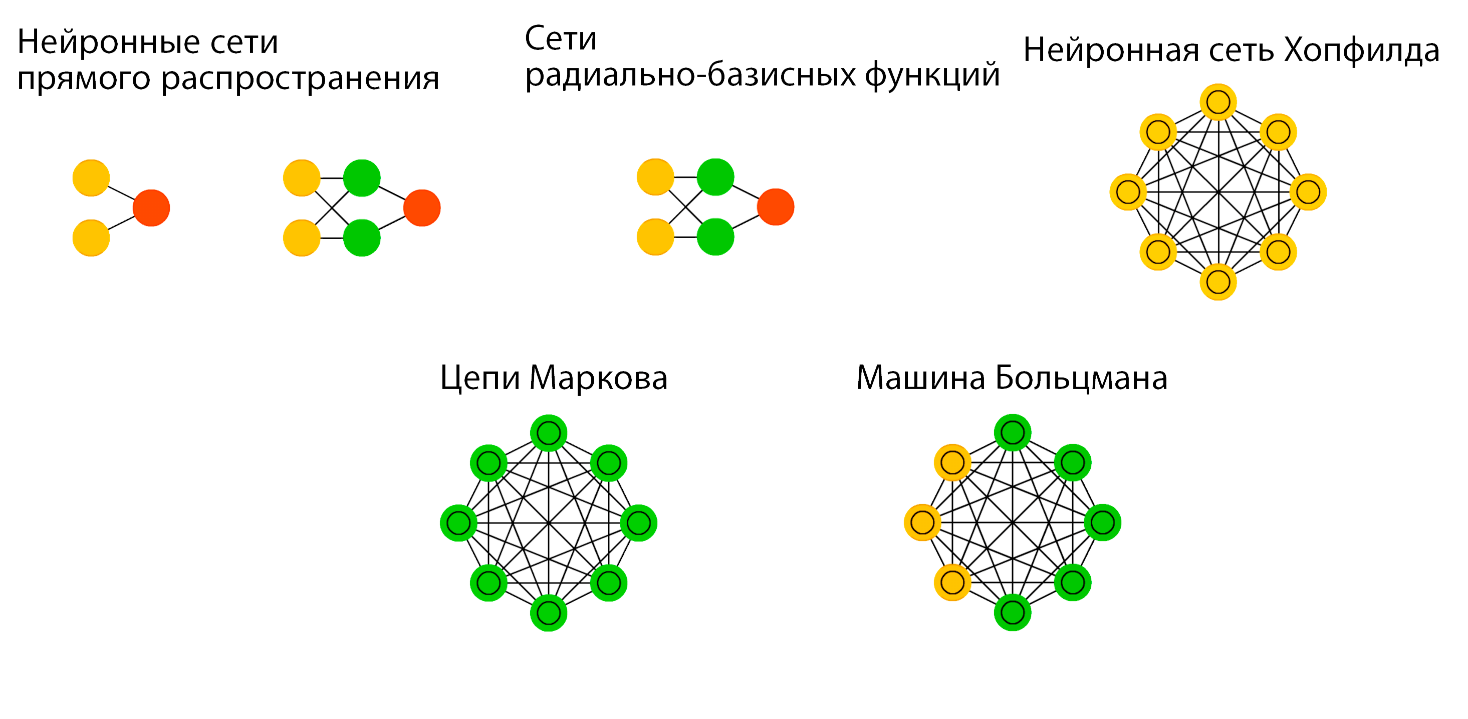


Рисунок 2.2 – Основные виды нейронных сетей

### 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.

# 3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## 3.1 Разработка структуры информационной системы

Разработанную информационную систему можно условно разбить на четыре основные части:

* Библиотека классов – содержит сформированный перечень классов, используемых во всех остальных модулях информационной системы (пользователь, тип продукта, продукт, отзыв);
* Серверная часть включает в себя модуль работы с базой данных, модуль обработки запросов пользователя, модуль нейронной сети;
* 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);
* База данных хранит группы товаров, товары, пользователи и их отзывы;

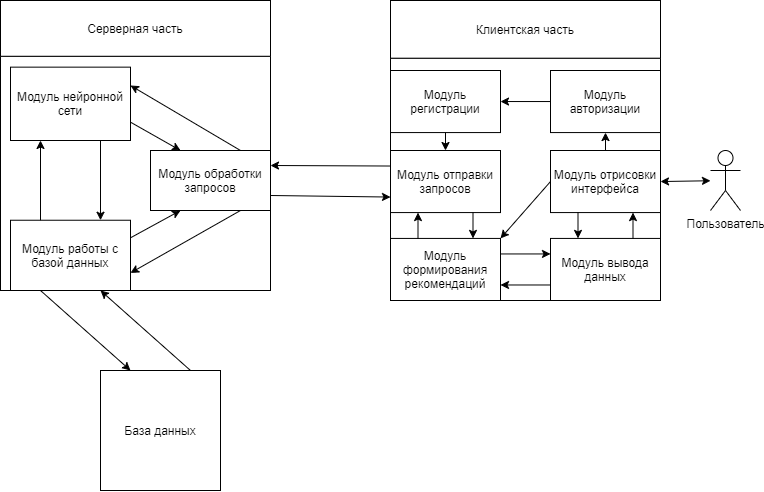


Рисунок 3.1 – Структура информационной системы

## 3.2 Реализация клиентской части

Клиентская часть состоит из следующих структурных модулей:

1. Модуль авторизации;
2. Модуль регистрации;
3. Модуль отправки запросов;
4. Модуль формирования рекомендаций.

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

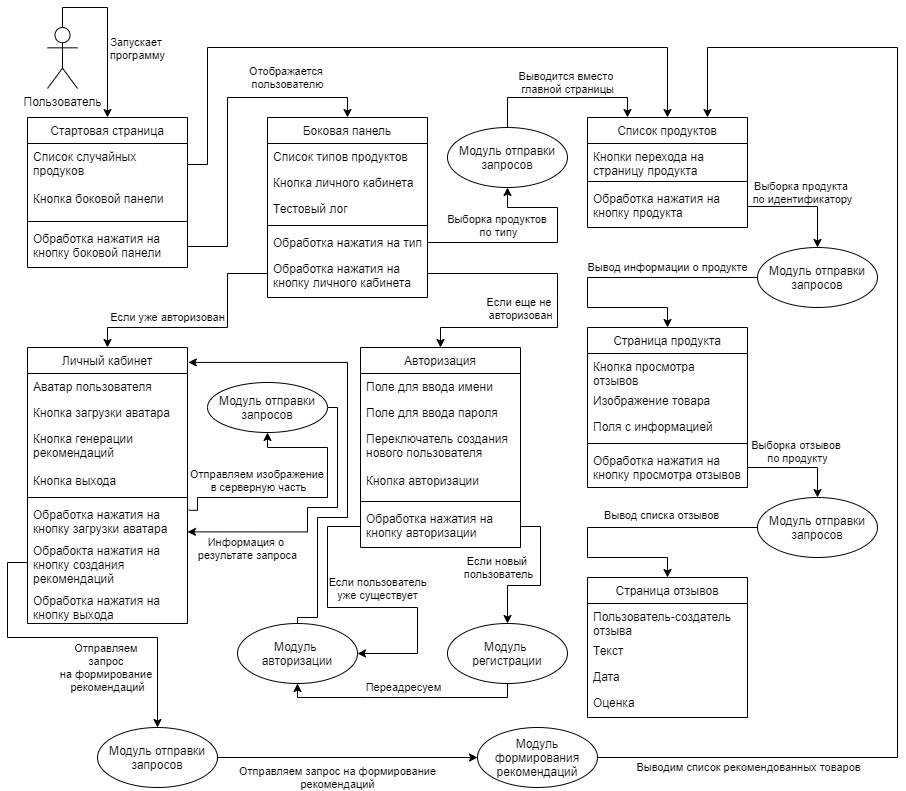


Рисунок 3.2 – Структура и взаимодействие клиентской части информационной системы

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.
   1. 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.
   2. 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.
   3. 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).

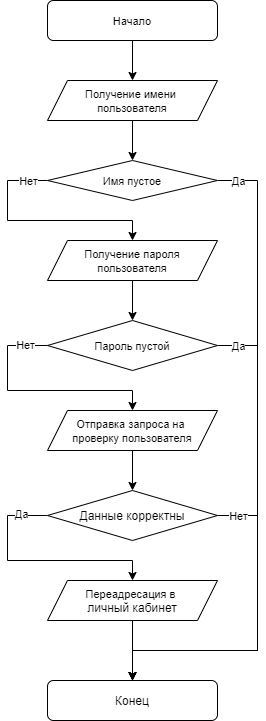


Рисунок 3.3 – Алгоритм работы модуля авторизации

### 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).

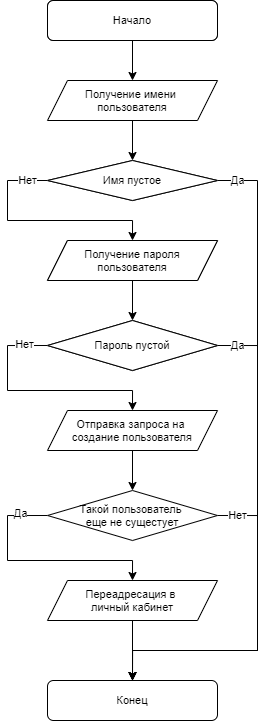


Рисунок 3.4 – Алгоритм работы модуля регистрации

### 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:

1. Получение списка пользователей;
2. Формирование списка существующих типов продуктов;
3. Создание списка всех продуктов определенного типа;
4. Генерация списка рекомендаций по определенному пользователю;
5. Воспроизведение набора отзывов по конкретному товару;
6. Считывание хэш-кода пароля требуемого пользователя;
7. Получение статического изображения по его названию;
8. Формирование списка объектов любого допустимого типа из JSON строки;
9. Загрузка изображения на сервер;
10. Добавление нового пользователя;

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

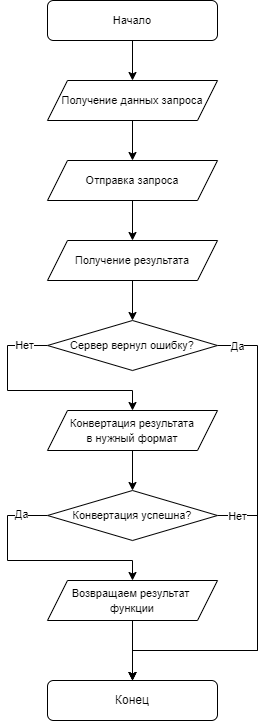


Рисунок 3.5 – Алгоритм работы модуля запросов

### 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).

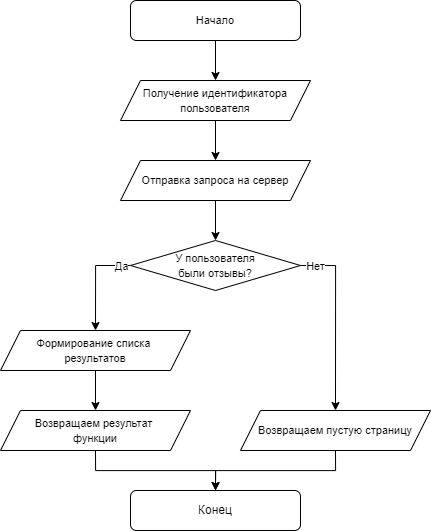


Рисунок 3.5 – Алгоритм работы модуля запросов

## 3.3 Реализация серверной части

Серверная часть состоит из следующих модулей:

1. Модуль работы с базой данных;
2. Модуль обработки запросов пользователя;
3. 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.

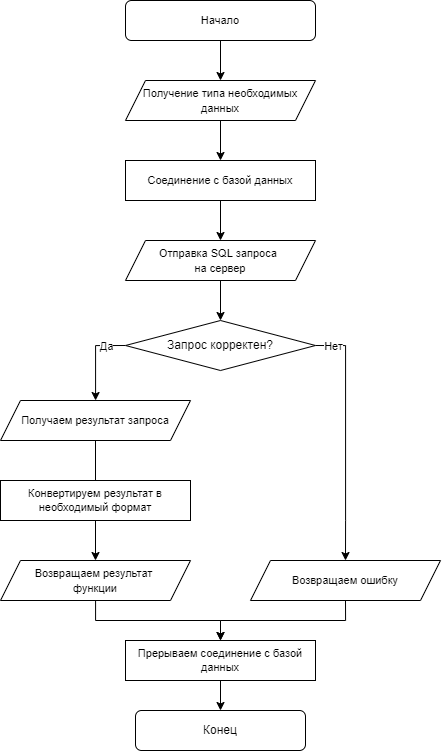


Рисунок 3.6 – Алгоритм работы модуля взаимодействия с базой данных

### 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 на рисунке 3.7.

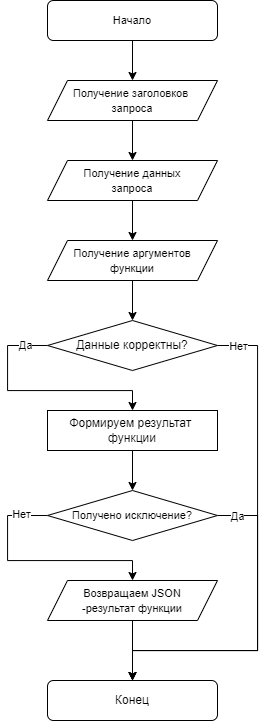


Рисунок 3.7 – Алгоритм работы модуля обработки запросов пользователя

### 3.3.1 Модуль нейронной сети

Модуль нейронной сети можно разбить на следующие основные элементы:

1. Модуль представления входных данных;
2. Модуль формирования топологии;
3. Модуль обучения;
4. 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).

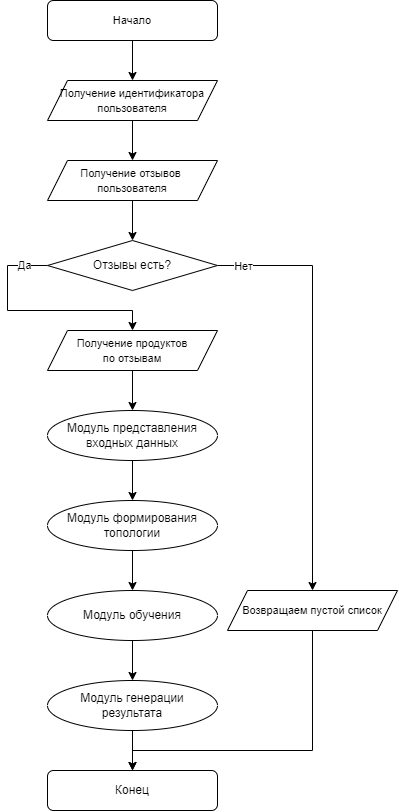


Рисунок 3.8 – Алгоритм работы модуля нейронной сети

#### 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).

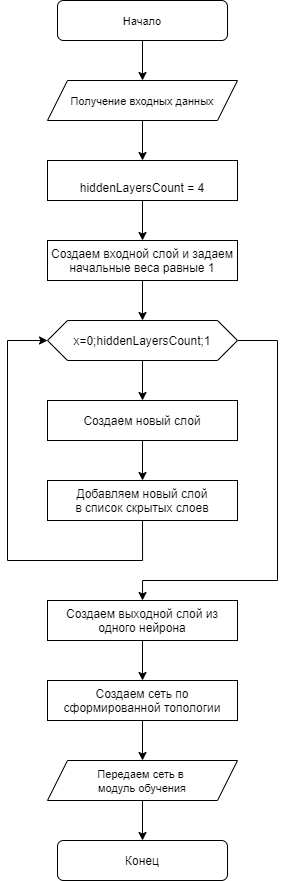


Рисунок 3.9 – Алгоритм работы модуля формирования топологии

#### 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).

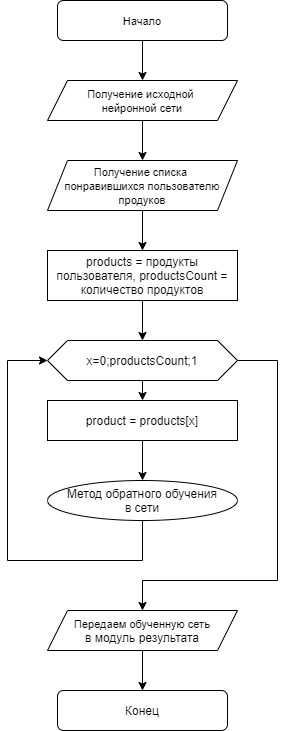


Рисунок 3.10 – Алгоритм работы модуля обучения

#### 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).

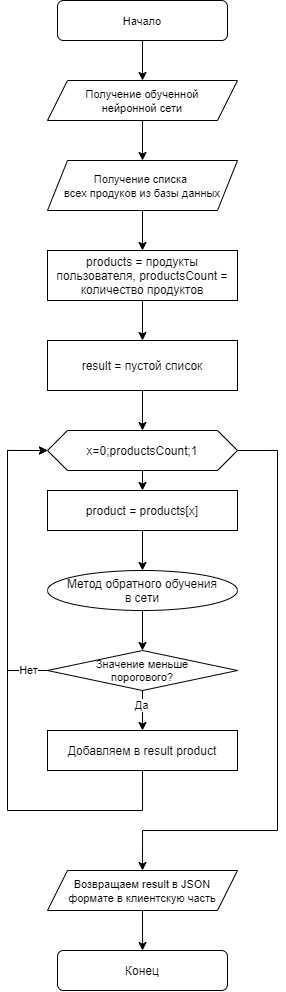


Рисунок 3.11 – Алгоритм работы модуля генерации результата

## 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.

## 3.5 Разработка структуры базы данных

База данных состоит из следующих элементов (рисунок 3.12):

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

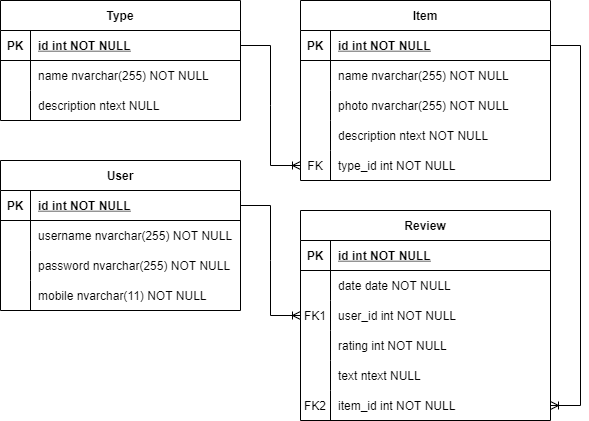


Рисунок 3.12 – Структура базы данных

## 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

# 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 |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название этапа | Исполнитель | Трудоемкость,  чел/дни | Продолжительность работ, дни |
| 5.9 | Сдача готового продукта и внедрение | Руководитель | 2 |  |
| 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 |
| 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 |
| 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

In the process of writing the final qualifying work, the author (graduate student) was guided by the following regulations:

1. "On the protection of the population and territory from natural and man-made emergencies" dated 21.12.1994 No. 68-FZ.

2. On the basics of public health protection in the Russian Federation" dated 21.11.2011 No. 323-FZ

3. "On Civil Defense" dated 12.02.1998 No. 28-FZ.

4. Order of the Ministry of Health and Social Development of the Russian Federation No. 477n dated 04.05.2012 "On approval of the list of conditions in which first aid is provided and the list of first aid measures".

5. Labor Code of the Russian Federation No. 197-FZ dated 30.12.2001

6. SanPiN – 2.2.2/542-96 "Hygienic requirements for video display terminals, personal electronic computers and work organization"

# 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 – Программный код

### «Item.cs»

using System.Collections.Generic;

using System.Linq;

namespace RecommendationSystem.Models

{

//продукт в базе

public class Item : Model

{

#region Private

private string name;

private string photo;

private string description;

private int typeId;

private bool edibility;

private double average\_price;

private bool with\_sugar;

private bool hypoallergenic;

private bool eco\_friendly;

private bool import;

private bool antibacterial;

private bool non\_gmo;

private bool vegan;

private bool lean;

#endregion

#region Properties

//название продукта

public string Name { get => name; }

//фотография продукта

public string Photo { get => photo; }

//описание продукта

public string Description { get => description; }

//тип продукта

public int TypeId { get => typeId; }

private bool IsEdibility { get => edibility; }

private float AveragePrice { get => average\_price; }

private bool IsWithSugar { get => with\_sugar; }

private bool IsHypoallergenic { get => hypoallergenic; }

private bool EcoFriendly { get => eco\_friendly; }

private bool IsImport { get => import; }

private bool IsAntibacterial { get => antibacterial; }

private bool IsNonGMO { get => non\_gmo; }

private bool IsVegan { get => vegan; }

private bool IsLean { get => lean; }

#endregion

public Item()

{

}

//конструктор со всеми основными аргументами

public Item(int id, string name, string photo, string description, int typeId) : base(id)

{

this.name = name;

this.photo = photo;

this.description = description;

this.typeId = typeId;

}

//конструктор в случае, если передают массив объектов

public Item(object[] items)

{

SetProperty(ref Id, items[0]);

SetProperty(ref name, items[1]);

SetProperty(ref photo, items[2]);

SetProperty(ref description, items[3]);

SetProperty(ref typeId, items[4]);

SetProperty(ref edibility, items[5]);

SetProperty(ref average\_price, items[6]);

SetProperty(ref with\_sugar, items[7]);

SetProperty(ref hypoallergenic, items[8]);

SetProperty(ref eco\_friendly, items[9]);

SetProperty(ref import, items[10]);

SetProperty(ref antibacterial, items[11]);

SetProperty(ref non\_gmo, items[12]);

SetProperty(ref vegan, items[13]);

SetProperty(ref lean, items[14]);

}

public Item(dynamic objects)

{

this.Id = objects["Id"].ToObject<int>();

this.name = objects["Name"].ToObject<string>();

this.photo = objects["Photo"].ToObject<string>();

this.typeId = objects["TypeId"].ToObject<int>();

this.edibility = objects["Edibility"].ToObject<bool>();

this.average\_price = objects["AveragePrice"].ToObject<bool>();

this.with\_sugar = objects["IsWithSugar"].ToObject<bool>();

this.edibility = objects["edibility"].ToObject<bool>();

try

{

this.description = objects["Description"].ToObject<string>();

} catch { }

}

public Type GetType(List<Type> types) => types.Where(x => x.Id == this.TypeId).FirstOrDefault();

}

}

### «Model.cs»

using System;

using System.ComponentModel;

namespace RecommendationSystem.Models

{

// базовый общий класс модели

public class Model

{

// уникальный идентификатор объекта

public int Id;

//пустой конструктор объекта, без аргументов

public Model()

{

}

//конструктор для ситуации, если пользователь передает ID

public Model(int id)

{

Id = id;

}

#region Utils

protected void SetProperty(ref int property, object value)

{

if (value is not DBNull)

{

property = (int)value;

}

}

protected void SetProperty(ref string property, object value)

{

if (value is not DBNull)

{

property = (string)value;

}

}

protected void SetProperty(ref bool property, object value)

{

if (value is not DBNull)

{

property = (bool)value;

}

}

protected void SetProperty(ref float property, object value)

{

if (value is not DBNull)

{

property = (double)value;

}

}

#endregion

}

}

### «Review.cs»

using System.Collections.Generic;

using System.Linq;

namespace RecommendationSystem.Models

{

// отзывы в базе

public class Review : Model

{

#region Private

private string date;

private int authorId;

private int rating;

private string text;

private int itemId;

#endregion

#region Properties

// дата написания

public string Date { get => date; }

// автор отзыва

public int AuthorId { get => authorId; }

// оценка продукта

public int Rating { get => rating; }

// текст отзыва

public string Text { get => text; }

// продукт, к которому относится отзыв

public int ItemId { get => itemId; }

#endregion

// пустой конструктор

public Review()

{

}

// конструктор со всеми основными аргументами

public Review(int Id, string date, int authorId, int rating, string text, int itemId)

{

this.Id = Id;

this.date = date;

this.authorId = authorId;

this.rating = rating;

this.text = text;

this.itemId = itemId;

}

// конструктор в случае, если передают массив объектов

public Review(object[] items)

{

SetProperty(ref Id, items[0]);

SetProperty(ref date, items[1]);

SetProperty(ref authorId, items[2]);

SetProperty(ref rating, items[3]);

SetProperty(ref text, items[4]);

SetProperty(ref itemId, items[5]);

}

// конструктор для динамической конвертации

public Review(dynamic objects)

{

this.Id = objects["Id"].ToObject<int>();

this.date = objects["Date"].ToObject<string>();

this.authorId = objects["AuthorId"].ToObject<int>();

this.rating = objects["Rating"].ToObject<int>();

this.text = objects["Text"].ToObject<string>();

this.itemId = objects["ItemId"].ToObject<int>();

}

#region Utils

// получить предмет по его id

public Item GetItem(List<Item> items) => items.Where(x => x.Id == this.ItemId).FirstOrDefault();

// получить пользователя по его id

public User GetAuthor(List<User> users) => users.Where(x => x.Id == this.AuthorId).FirstOrDefault();

#endregion

}

}

### «Type.cs»

namespace RecommendationSystem.Models

{

// класс типов продуктов

public class Type : Model

{

// скрытое название типа

private string name;

// название типа

public string Name { get => name; }

// пустой конструктор

public Type()

{

}

// конструктор со всеми основными аргументами

public Type(int Id, string name) : base(Id) => this.name = name;

// конструктор в случае, если передают массив объектов

public Type(object[] items)

{

//преобразуем объекты и приравниваем

SetProperty(ref Id, items[0]);

SetProperty(ref name, items[1]);

}

// конструктор для динамического формирования объектов

public Type(dynamic objects)

{

this.Id = objects["Id"].ToObject<int>();

this.name = objects["Name"].ToObject<string>();

}

}

}

### «User.cs»

using System;

using System.ComponentModel;

namespace RecommendationSystem.Models

{

// пользователь в системе

public class User : Model

{

#region Private

private string name;

private string password;

private string mobile;

private string photo;

#endregion

#region Properties

// имя пользователя

public string Name { get => name; }

// пароль пользователя

public string Password { get => password; }

// мобильный телефон пользователя

public string Mobile { get => mobile; }

// путь до аватарки на сервере

public string Photo { get => photo; }

#endregion

// пустой конструктор

public User()

{

}

// конструктор со всеми основными аргументами

public User(int id, string name, string password, string mobile, string photo) : base(id)

{

this.name = name;

this.password = password;

this.mobile = mobile;

this.photo = photo;

}

// конструктор в случае, если передают массив объектов

public User(object[] items)

{

SetProperty(ref Id, items[0]);

SetProperty(ref name, items[1]);

SetProperty(ref password, items[2]);

SetProperty(ref mobile, items[3]);

SetProperty(ref photo, items[4]);

}

// динамический конструктор

public User(dynamic objects)

{

this.Id = objects["Id"].ToObject<int>();

this.name = objects["Username"].ToObject<string>();

this.password = objects["Password"].ToObject<string>();

this.mobile = objects["Mobile"].ToObject<string>();

this.photo = objects["Photo"].ToObject<string>();

}

}

}

### «Program.cs»

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace RecommendationSystem

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

### «Sturtup.cs»

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.FileProviders;

using Microsoft.Extensions.Hosting;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

namespace RecommendationSystem

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles(new StaticFileOptions

{

FileProvider = new PhysicalFileProvider(

Path.Combine(env.ContentRootPath, "static")),

RequestPath = "/static"

});

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Api}/{action=Index}/{id?}");

endpoints.MapFallback(context => {

context.Response.Redirect("/api/error");

return Task.CompletedTask;

});

});

}

}

}

### «ApiController.cs»

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.Reflection;

using System.Security.Cryptography;

using System.Text;

namespace RecommendationSystem.Controllers

{

// API контроллер

public class ApiController : Controller

{

#region API Methods

[HttpGet] // главная страница

public IActionResult Index()

{

// возвращаем статус ОК

Response.StatusCode = (int)HttpStatusCode.OK;

// выводим стартовое сообщение

return Json(new { Message = "Its starting page. Use GET/POST requests to use data" });

}

[HttpGet] // любой нестандартный путь + ошибки, путь /api/error

public IActionResult Error(string message)

{

// возвращаем статус Плохой запрос

Response.StatusCode = (int)HttpStatusCode.BadRequest;

// сообщаем, что произошла ошибка

return Json(new { ErrorMessage = message });

}

[HttpGet] // получить список пользователей, путь /api/users

public IActionResult Users() => Ok<User>();

[HttpGet] // получить список отзывов по продукту /api/userbyreview/?review\_id=<type\_id>

public IActionResult UserByReview(string review\_id)

{

var review = Database.GetObject<Review>($"[id] = {review\_id}").FirstOrDefault();

if (review is not null)

{

var user = Database.GetJson<User>($"[id] = {review.AuthorId}");

return Ok(user);

}

return Redirect("api/error?message=user doesn't exists");

}

[HttpGet("api/getrecommend")] // получить рекомендацию по пользователю /api/getrecommend/?user\_id=<type\_id>

public IActionResult GetRecommend(string user\_id)

{

var reviews = Database.GetObject<Review>($"[user\_id] = {user\_id}").Where(x => x.Rating == 4 || x.Rating == 5);

foreach (Review review in reviews)

{

Topology topology = new Topology(11, 1, 4);

NeuronNetwork network = new NeuronNetwork(topology);

var item = review.GetItem(Database.GetObject<Item>());

List<double> signals = GetSignals(item);

network.FeedForward(signals);

}

var outputItems = new List<Item>();

var items = Database.GetObject<Item>();

foreach (Item item in items)

{

Topology topology = new Topology(11, 1, 4);

NeuronNetwork network = new NeuronNetwork(topology);

List<double> signals = GetSignals(item);

var neuron = network.FeedForward(signals);

//TODO: откорректировать вес

if (neuron.Output > 0.5)

{

outputItems.Add(item);

}

}

return Ok(outputItems);

}

private List<double> GetSignals(Item item)

{

//TODO: обновить библиотеку и подключить все свойства

return new List<double>

{

item.TypeId

};

}

[HttpGet] // получить список типов продуктов, путь /api/types

public IActionResult Types() => Ok<Models.Type>();

[HttpGet] // получить список продуктов по типу /api/itemsbytype/?type\_id=<type\_id>

public IActionResult ItemsByType(string type\_id) => Ok<Item>($"[type\_id] = {type\_id}");

[HttpGet] // получить список всех продуктов, путь /api/items

public IActionResult Items() => Ok<Item>();

[HttpGet] // получить список всех отзывов, путь /api/reviews

public IActionResult Reviews() => Ok<Review>();

[HttpGet] // получить список отзывов по продукту /api/reviewsbyitem/?item\_id=<type\_id>

public IActionResult ReviewsByItem(string item\_id) => Ok<Review>($"[item\_id] = {item\_id}");

[HttpGet] // получить пользователя по имени /api/getuserbyname?name=<имя>

public IActionResult GetUserByName(string name) => Ok<User>($"[username] = '{name}'");

[HttpGet] // получить хэш пароля текущего пользователя /api/getuserhash?user\_id=<id>

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] // проверка на корректность хэша /api/checkuserhash?user\_id=<id>&password=<password>

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);

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] // загрузка аватарки /api/sendavatar

[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;

return Redirect($"api/error?message={ex.Message}");

}

}

#endregion

#region Utils

// вспомогательный метод для сокращения

private ActionResult Ok<T>(string where = "") where T: Model => Ok(Database.GetJson<T>(where));

// функция формирования хэша

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();

}

#endregion

}

}

### «Database.cs»

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 UnityEditor;

using UnityEngine;

using UnityEngine.Networking;

using UnityEngine.UI;

public class Database: MonoBehaviour

{

// адрес сервера

[SerializeField] private string localHost = @"http://localhost:44381";

[SerializeField] private string internetHost = @"http://u1674941.plsk.regruhosting.ru";

[SerializeField] private bool isLocal = true;

private string host = "";

private void Start()

{

if (isLocal)

{

host = localHost;

}

else

{

host = internetHost;

}

}

// подключен ли клиент

//[SerializeField] private bool isConnected = true;

// включена ли отладка

[SerializeField] private bool isDebug = true;

// страница, выводимая при отсутствии подключения

[SerializeField] private GameObject noConnectionFrame;

// список пользователей

public List<User> Users => GetObjects<User>("users");

// список типов продуктов

public List<RecommendationSystem.Models.Type> Types => GetObjects<RecommendationSystem.Models.Type>("types");

// список продуктов по типу

public List<Item> GetItemsByType(int typeId) => GetObjects<Item>($"itemsbytype/?type\_id={typeId}");

// список отзывов по продукту

public List<Review> GetReviewsByItem(Item item) => GetObjects<Review>($"reviewsbyitem/?item\_id={item.Id}");

// получение автора отзыва

public User GetUserByReview(int reviewId) => GetObject<User>($"userbyreview/?review\_id={reviewId}");

// получение хэша пароля пользователя по его id

public string GetUserHash(int user\_id) => GetRequest($"getuserhash?user\_id={user\_id}");

#region Utils

// получение картинки с сервера по названию

public IEnumerator GetImageByName(string name, Image image)

{

// путь до статичных файлов

string path = $"{host}/static/{name}";

// отправляем запрос на скачивание файла

UnityWebRequest request = UnityWebRequest.Get(path);

DownloadHandlerTexture downloader = new();

request.downloadHandler = downloader;

request.certificateHandler = new ForceAcceptAll();

// если отладка включена

Log($"Попытка загрузки файла по пути {path}");

request.SendWebRequest();

// ждем загрузки картинки

while (!request.isDone)

yield return null;

// получаем загруженную картинку

var texture = downloader.texture;

// передаем её в интерфейс

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");

}

// получение json как ответ с сервера

private string GetRequest(string path, string method = "GET")

{

try

{

string finalPath = $"{host}/api/{path}";

Log($"Отправлен запрос по пути: {finalPath}");

// кидаем запрос

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;

// получаем ответ

HttpWebResponse resp = request.GetResponse() as HttpWebResponse;

string text = "";

using (StreamReader sr = new(resp.GetResponseStream(), Encoding.Default))

text = sr.ReadToEnd();

text = text.Trim(new char[] { '"' });

// отмечаем, что соединение есть

//isConnected = true;

// отключаем панель отсутствия соединения

SwitchFrame(true);

// вовзращаем json

return text;

}

catch (Exception ex)

{

// включаем панель отсутствия соединения

SwitchFrame(false);

// кидаем ошибку

throw new Exception($"Не удалось считать по пути '{path}', причина: '{ex.Message}'");

}

}

// функция конвертации списка объектов из json

public List<T> GetObjects<T>(string query) where T : Model

{

try

{

// кидаем запрос на сервер и получаем ответ

string text = GetRequest(query);

// конвертируем json в динамический словарь узлов

var raw = JsonConvert.DeserializeObject<dynamic>(text);

// создаем пустой-список результат

List<T> result = new();

// проходимся по каждому узлу json

foreach (var data in raw.Children())

{

var parameters = new object[1];

parameters[0] = data;

// создаем экземпляр объекта

result.Add((T)Activator.CreateInstance(typeof(T), parameters));

}

// возвращаем результат

return result;

}

catch (Exception ex)

{

throw new Exception($"Не удалось считать {typeof(T).Name}, причина {ex.Message}");

}

}

// функция получения одного объекта из json

public T GetObject<T>(string query) where T : Model

{

try

{

// кидаем запрос

string text = GetRequest(query);

// конвертируем json в динамический словарь узлов

var raw = JsonConvert.DeserializeObject<dynamic>(text);

var parameters = new object[1];

parameters[0] = raw.Last;

// создаем экземпляр объекта

return (T)Activator.CreateInstance(typeof(T), parameters);

}

catch (Exception ex)

{

throw new Exception($"Не удалось считать {typeof(T).Name}, причина {ex.Message}");

}

}

// функция загрузки картинки

public void SetImage(string url, Image image) => StartCoroutine(GetImageByName(url, image));

// функция отправки изображения

public void UploadTexture(int id) => StartCoroutine(UploadTextureRoutine(id));

// функция создания пользователя

public void AddUser(string username, string password)

{

string allPath = $"createuser?username={username}&password={password}";

Log($"Отправлен запрос по пути: {allPath}");

GetRequest(allPath);

}

// корунтина для отправки изображения

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)

{

// Create Texture from selected image

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}");

}

}

}

}

// функция переключения состояния активности панели отсутствия подключения

private void SwitchFrame(bool isConnected) => noConnectionFrame.SetActive(!isConnected);

// функция логирования

private void Log(string text)

{

if (isDebug)

{

Debug.Log($"[LOG|{DateTime.Now:HH:mm:ss}] {text}");

}

}

#endregion

public class ForceAcceptAll : CertificateHandler

{

protected override bool ValidateCertificate(byte[] certificateData)

{

return true;

}

}

}