СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc167202604)

[1. АНАЛИЗ СУЩЕСТВУЮЩИХ СИСТЕМ, ПРЕДМЕТНОЙ ОБЛАСТИ, ПОСТАНОВКА ЗАДАЧИ 6](#_Toc167202605)

[1.1 АНАЛИЗ СУЩЕСТВУЮЩИХ СИСТЕМ 6](#_Toc167202606)

[1.2 АНАЛИЗ АЛГОРИТМОВ ПОДАЧИ КОНТЕКСТА 7](#_Toc167202607)

[1.3 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9](#_Toc167202608)

[1.4 ПОСТАНОВКА ЗАДАЧИ 10](#_Toc167202609)

[2. ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ СРЕДСТВ 11](#_Toc167202610)

[2.1 СРЕДСТВА РЕАЛИЗАЦИИ 11](#_Toc167202611)

[2.2 ОБЩЕЕ ОПИСАНИЕ СТРУКТУРЫ ПРОГРАММ 14](#_Toc167202612)

[2.3 ПРОЕКТИРОВАНИЕ ПРОГРАММЫ 21](#_Toc167202613)

[3.РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММНЫХ СРЕДСТВ 23](#_Toc167202614)

[3.1 РЕАЛИЗАЦИЯ ПРОГРАММНЫХ СРЕДСТВ 23](#_Toc167202615)

[3.2 НАПОЛНЕНИЕ БАЗЫ ЗНАНИЙ И ТЕСТИРОВАНИЕ ПЛАТФОРМЫ 32](#_Toc167202616)

[4.ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ 36](#_Toc167202617)

[4.1 ИСХОДНЫЕ ДАННЫЕ ДЛЯ РАСЧЕТА ЭКОНОМИЧЕСКОГО ЭФФЕКТА 36](#_Toc167202618)

[4.2 РАСЧЕТ ОБЪЕМА ФУНКЦИЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 37](#_Toc167202619)

[4.3 РАСЧЕТ ПОЛНОЙ СЕБЕСТОИМОСТИ ПРОГРАММНОГО ПРОДУКТА 38](#_Toc167202620)

[4.4 РАСЧЕТ ЦЕНЫ И ПРИБЫЛИ ПО ПРОГРАММНОМУ ПРОДУКТУ 44](#_Toc167202621)

[5. ЭНЕРГОСБЕРЕЖЕНИЕ 46](#_Toc167202622)

[ЗАКЛЮЧЕНИЕ 50](#_Toc167202623)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 51](#_Toc167202624)

[СПИСОК СОКРАЩЕНИЙ 52](#_Toc167202625)

**ВВЕДЕНИЕ**

В современном мире нейронные сети уже не новинка. Они уже используются во многих сферах нашей жизни, их встраивают в системы управления автомобилем, при помощи их прогнозируется погода, в некоторых странах нейронные сети заменяют работников на предприятиях. В настоящее время нейронные сети вошли в практику везде, где это необходимо для решения задач прогнозирования, распознавания образов, классификации или контроля. Эти мощные инструменты машинного обучения позволяют компаниям и организациям решать сложные задачи, которые ранее казались невозможными. Применение нейронных сетей в работе приводит к увеличению производительности, снижению затрат и разработке инновационных продуктов.

Одним из наиболее ярких примеров применения нейронных сетей является обработка текста и анализ естественного языка. Нейросети способны анализировать текстовые данные, извлекать смысл и контекст, классифицировать тексты и даже генерировать текст в стиле человеческого написания. Это пригодно для автоматической обработки текстовых данных, как например, в области клиентского обслуживания, где нейронные сети могут использоваться для анализа отзывов клиентов и определения их настроения, а также для автоматической генерации текстовых ответов.

В данной работе будет рассмотрена разработка программной платформы по созданию цифровых ассистентов (чат-ботов), которые будут основываться на таких генерирующих нейронных сетях.

Для достижения этой цели будет использована библиотека LangChain для языка программирования Python, которая облегчит взаимодействие с нейронными сетями и позволит подгружать контекст диалога для улучшения способностей искусственного интеллекта, в частности его знаний конкретной темы разговора.

Таким образом, данная работа будет представлять собой практическое исследование в области применения нейронных сетей и создания легко расширяемых программных продуктов.

В данной работе главная цель – создание платформы, которая позволит разветвлять области её применения без больших затрат сил по переделыванию методов взаимодействия с нейронными сетями и их пользовательским интерфейсом. Каждая ветка этой платформы будет уметь отвечать на разные темы, которые будут интересовать пользователя. Ответы будут подкреплены учебниками или другими документами в электронном формате.

# АНАЛИЗ СУЩЕСТВУЮЩИХ СИСТЕМ, ПРЕДМЕТНОЙ ОБЛАСТИ, ПОСТАНОВКА ЗАДАЧИ

В этой главе будут введены основные понятия предметной области, проанализируем существующие системы, выделим их плюсы и минусы, рассмотрим различные библиотеки, которые позволят выполнить цель работы.

* 1. **АНАЛИЗ СУЩЕСТВУЮЩИХ СИСТЕМ**

На данный момент существует узкий круг систем, которые выполняют функции схожие с разрабатываемой платформой. На начало разработки их практически не существовало, однако подходя к концу цикла разработки стали появляться схожие системы от больших компаний-разработчиков. Рассмотрим несколько известных аналогов и их характеристики:

* **Coze:**
  + *Плюсы:*
    - Полностью бесплатный хост бота без ограничений по сообщениям.
    - Минималистичный пользовательский интерфейс, нету необходимости в долгом изучении программы перед использованием.
    - Позволяет загружать таблицы в базу знаний бота.
  + *Минусы:*
    - Чат с ботом доступен только на определённых платформах, нельзя разместить его в пользовательском Application Programming Interface (API).
* **Popai:**
  + *Плюсы:*
    - Пользовательский интерфейс красив и лёгок в использовании.
    - Присутствует возможность бесплатного использования.
  + *Минусы:*
    - Использование бесплатно сильно ограничено, платная подписка достаточно дорогая, стоимость которой составляет 33 доллара США.
* **Flowxo:**
  + *Плюсы:*
    - Сервис поддерживает большой спектр моделей нейронных сетей.
    - Имеет возможность бесплатного использования.
  + *Минусы:*
    - Бесплатно пользоваться можно лишь периодом в 14 дней, после чего аккаунт перестаёт работать.
    - Отсутствует возможность вывести общение с ботом через API.

Хотя указанные платформы демонстрируют впечатляющий функционал, применить их для целей данной работы невозможно из-за отсутствия решения, которое позволит разместить бота в виде API для встраивания в пользовательский интерфейс за разумную цену. Для достижения целей работы необходимо создать собственное решение, которое будет отвечать всем необходимым требованиям.

## 1.2 АНАЛИЗ АЛГОРИТМОВ ПОДАЧИ КОНТЕКСТА

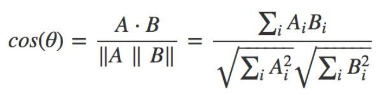
Подача (подгрузка) контекста – важная часть работы цифрового ассистента. Процесс подачи контекста заключается в поиске текстовой информации, которая поможет ассистенту ответить на поставленный человеком вопрос при помощи внешних источников, например научных статей, найденных в интернете, либо электронных документов, загруженных в базу знаний ассистента заранее. Данный процесс принято называть Retrieval Augmented Generation (RAG), однако далее будет написано «подача контекста».

Рассмотрим шаги, которые необходимо выполнить для реализации механизма подачи контекста в цифровых ассистентах:

* **Сборка основы базы знаний:** Собираются необходимые документы, которые будут присутствовать в базе знаний для подкрепления ответов бота. В основном это различные документы формата «.pdf», которые помещаются в отдельную папку компьютера, на котором планируется запуск цифрового ассистента.
* **Разделение текста на части:** Весь тест из каждого документа считывается и обрезается в куски размером от 1300 до 1600 слов, в зависимости от длины последнего предложения.
* **Векторизация текста:** Каждый кусок текста проходит через специальную функцию векторизации. Эта функция векторизации преобразует текст в набор математических векторов, которые отражают семантический смысл предложения (его смысл).
* **Хранение векторов:** Все преобразованные пары текста и вектора хранятся в векторной базе данных, что и является базой знаний цифрового ассистента.
* **Получение запроса от пользователя:** Как только система получает вопрос от пользователя, его вопрос преобразуется по точно такой же функции векторизации и полученные векторы сравниваются с теми, что хранятся в базе знаний. Обычно для этого используется косинусная схожесть между векторами. Самый близкий вектор в базе знаний по сравнению с вопросом и является необходимым нам куском текста.
* **Подача контекста:** Этот найденный в базе знаний кусок текста помещается в контекст для нейронной сети вместе с вопросом пользователя, после чего полученный ответ после генерации ассистентом можно предоставить пользователю. Таким образом, цифровой ассистент видит не только вопрос пользователя, но и предоставленную для ответа на этот вопрос информацию, произошла подача контекста.

Для успешного выполнения описанных ранее шагов, необходимо определиться с используемыми алгоритмами и функциями, в данном случае конкретно с алгоритмом определения схожести векторов и функцией векторизации.

В области нейронных сетей схожесть двух векторов чаще всего определяют путём нахождения косинусной схожести (cosine similarity), так как результатом этого является число в диапазоне от 0 до 1 в каждом случае. Косинусная схожесть определяется следующей формулой:



где:

А – первый вектор.

В – второй вектор.

Аi – очередной компонент вектора А.

Bi – очередной компонент вектора В.

Кроме алгоритма определения схожести векторов, необходимо выбрать функцию векторизации. В сети интернет размещено в открытом доступе огромное количество таких функций. Они представляют из себя преобразующие нейронные сети, которые позволяют захватить смысловую нагрузку последовательных данных, будь это текст или картинка. В случае данной работы необходимо выбрать один из таких преобразователей, который воспринимает на вход текст и выдаёт n-размерный вектор.

Исходя из результатов бенчмарков пользователей интернет-ресурса HuggingFace (известный веб-сайт, посвящённый открытым нейронным сетям), было решено в работе использовать функцию под названием «sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2». Данная функция векторизации преобразует параграфы и предложения в вектор размером 384 и может быть использована для семантического поиска. Выбор обусловлен хорошими результатами бенчмарков (3 место в таблице лидеров HuggingFace), большим количеством скачиваний (отражает популярность модели), а также наличием поддержки русского, белорусского, английского и немецкого языков.

* 1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

Предметная область данной работы охватывает применение искусственного интеллекта (ИИ) в качестве цифрового ассистента, который помогает пользователю отвечая на его вопросы, руководствуясь при этом дополнительной информацией из учебников или других источников, подгружающихся в контекст вопроса. Система устроена, используя клиент-серверную архитектуру, где в качестве сервера выступает персональный компьютер, на котором запущен Python сервер, а в качестве клиента выступает пользовательский интерфейс в виде мобильного приложения для операционной системы android. Основные понятия и термины, используемые в этой области, включают в себя:

* **Искусственный интеллект (ИИ):** В данной работе искусственным интеллектом называются генерационные нейронные сети, работающие на удалённых серверах и выдающие текстовый ответ при отправке им вопроса.
* **Цифровой ассистент**: Программная система, выполняющая роль виртуального помощника, который отвечает текстом на вопрос пользователя, руководствуясь при этом различными внешними источниками информации.
* **Контекст:** Часть текстового запроса к цифровому ассистенту, содержащая в себе дополнительную информацию, которая поможет ответить на вопрос пользователя, например, текущая погода в регионе пользователя, либо отрывок из научной статьи, найденной по теме вопроса.
* **Сервер:** Программная среда, в которой запущен код сервера цифрового ассистента и которая принимает запросы пользователя для обработки.
* **Клиент:** Программа, содержащая в себе пользовательский интерфейс и позволяющая взаимодействовать с сервером.
* **Промпт**: Полный конечный текстовый запрос, сформированный при помощи контекста и вопроса пользователя, отправляется целиком на сервер искусственного интеллекта для дальнейшей обработки и получения ответа.
* **Токен**: Малейшая частица общения с искусственным интеллектом, представляет из себя один «символ» алфавита нейронной сети и является целым положительным числом, отражающим его номер в таком алфавите. В русском языке может быть представлен несколькими буквами или словом целиком.
* **Вектор:** Массив чисел с плавающей точкой, отражающее семантический смысл куска текста, к которому он сгенерирован функцией векторизации.
* **Функция векторизации:** Нейронная сеть, принимающая на вход кусок текста ограниченной длины и возвращающая в виде ответа вектор, который отражает смысл этого текста.

Эти основные концепции играют ключевую роль в данной работе и позволяют разработать платформу по созданию цифровых ассистентов, ответы которых будут подкреплены внешними источниками информации.

* 1. **ПОСТАНОВКА ЗАДАЧИ**

В данной работе требуется разработать платформу создания цифровых ассистентов, способных вести диалог с человеком в рамках заданного контекста. Программа должна обеспечивать следующий функционал:

* Создание цифрового ассистента, который сможет вести понятный диалог с пользователем и в это время использовать контекст, установленный этому ассистенту при его создании, с помощью редактора контекста.
* Удобный пользовательский интерфейс в виде мобильного приложения для операционной системы Android.
* Редактирование контекста отдельного цифрового ассистента.
* Редактирование настроек большой языковой модели отдельного цифрового ассистента.

1. **ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ СРЕДСТВ**

В этой главе спроектируем программную систему. Для этого опишем ее общую структуру и в подробностях рассмотрим все подсистемы и выберем средства программирования для реализации программы.

* 1. **СРЕДСТВА РЕАЛИЗАЦИИ**

В процессе разработки серверной части дипломной работы был выбран язык программирования Python. Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Python поддерживает динамическую типизацию, то есть тип переменной определяется только во время исполнения. В Python имеются встроенные типы: булевый, строка, Unicode-строка, целое число произвольной точности, число с плавающей запятой, комплексное число и некоторые другие. Из коллекций в Python встроены: список, кортеж (неизменяемый список), словарь, множество и другие. Все значения являются объектами, в том числе функции, методы, модули, классы.

В качестве интегрированной среды разработки (IDE) был использован Visual Studio (VS) Code. VS Code — текстовый редактор, созданный компанией Microsoft специально для разработки кода. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией. Visual Studio Code был анонсирован 29 апреля 2015 года компанией Microsoft на конференции Build, и вскоре была выпущена бета-версия, вскоре после чего последовал релиз, после чего выпускались множество обновлений, постоянно повышающих качество продукта вдоль до сегодняшнего дня. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов.

VS Code поддерживает:

* Статический анализ кода, подсветка синтаксиса и ошибок.
* Навигация по проекту и исходному коду: отображение файловой структуры проекта, быстрый переход между файлами, классами, методами и использованиями методов.
* Рефакторинг: переименование, извлечение метода, введение переменной, введение константы, подъём и спуск метода и т. д.
* Инструменты для веб-разработки.
* Встроенный отладчик для Python.
* Встроенные инструменты для юнит-тестирования.
* Поддержка систем контроля версий: общий пользовательский интерфейс для Mercurial, Git, Subversion, Perforce и CVS с поддержкой списков изменений и слияния.

Для проведения экспериментов и исследований, а также для демонстрации результатов работы алгоритма использовались Jupyter Notebook. Jupyter Notebook был выбран как основная среда разработки, что обусловлено рядом ключевых преимуществ данного инструмента. Jupyter Notebook предоставляет возможность интерактивного выполнения кода, что особенно ценно при тестировании отдельных фрагментов алгоритмов и наблюдении за их выполнением в реальном времени. Эта особенность существенно упрощает процесс разработки и отладки.

Одним из значимых преимуществ Jupyter Notebook является поддержка визуализации данных через интеграцию с популярными библиотеками, такими как matplotlib и seaborn. Это позволяет наглядно представлять результаты анализа и работы алгоритмов прямо в рабочем документе, что делает процесс исследования более интуитивным и понятным.

Важной характеристикой Jupyter Notebook является возможность документирования исследовательского процесса. Поддержка разметки Markdown и LaTeX позволяет оформлять записные книжки с подробными пояснениями, выводами и математическими формулами, делая их содержание доступным для анализа и представления результатов.

Среда также способствует совместной работе над проектами благодаря удобству распространения записных книжек среди коллег или в публичном доступе. Это облегчает обмен знаниями и коллаборативную разработку.

Jupyter Notebook поддерживает работу с множеством языков программирования, что расширяет возможности для мультиязыкового программирования и исследований. Интеграция с различными инструментами и платформами, включая Google Colab, предоставляет доступ к ресурсам облачных вычислений и упрощает процесс разработки.

Наконец, расширяемость Jupyter Notebook через множество доступных расширений добавляет дополнительные функциональные возможности, такие как версионирование и автоматическое форматирование кода, что делает рабочий процесс более организованным и эффективным.

Для разработки пользовательского интерфейса в виде мобильного приложения для ОС Android была использована Android Studio. Android Studio — интегрированная среда разработки производства Google, с помощью которой разработчикам становятся доступны инструменты для создания приложений на платформе Android OS. Android Studio можно установить на Windows, Mac и Linux. Android Studio создавалась на базе IntelliJ IDEA.

Android Studio можно загрузить и пользоваться бесплатно. В ней присутствуют макеты для создания UI, с чего обычно начинается работа над приложением. В Studio содержатся инструменты для разработки решений для смартфонов и планшетов, а также новые технологические решения для Android TV, Android Wear, Android Auto, Glass и дополнительные контекстуальные модули.

Среда Android Studio предназначена как для небольших команд разработчиков мобильных приложений (даже в количестве одного человека), или же крупных международных организаций с GIT или другими подобными системами управления версиями. Опытные разработчики смогут выбрать инструменты, которые больше подходят для масштабных проектов. Решения для Android разрабатываются в Android Studio с использованием Java или C++. В основе рабочего процесса Android Studio заложен концепт непрерывной интеграции, позволяющий сразу же обнаруживать имеющиеся проблемы. Продолжительная проверка кода обеспечивает возможность эффективной обратной связи с разработчиками. Такая опция позволяет быстрее опубликовать версию мобильного приложения в Google Play App Store. Для этого присутствует также поддержка инструментов LINT, Pro-Guard и App Signing.

С помощью средств оценки производительности определяется состояние файла с пакетом прикладных программ. Визуализация графики дает возможность узнать, соответствует ли приложение ориентиру Google в 16 миллисекунд. С помощью инструмента для визуализации памяти разработчик узнает, когда его приложение будет использовать слишком много оперативной памяти и когда произойдет «сборка мусора». Инструменты для анализа батареи показывают, какая нагрузка приходится на устройство.

Android Studio совместима с платформой Google App Engine для быстрой интеграции в облаке новых API и функций. В среде разработки вы найдете различные API, такие как Google Play, Android Pay и Health. Присутствует поддержка всех платформ Android, начиная с Android 1.6.

Таким образом, выбор средств программирования был обусловлен необходимостью использования современных, надёжных и проверенных инструментов, которые обеспечивают гибкость и масштабируемость при разработке сложных программных систем.

* 1. **ОБЩЕЕ ОПИСАНИЕ СТРУКТУРЫ ПРОГРАММ**

Необходимо спроектировать структуру программы, которая удовлетворяла бы поставленным задачам. Для этого последовательно выдели все основные подсистемы, которые способны решить эти задачи.

Подсистема «Сервер». Данная подсистема является основной в этой работе, она будет принимать запросы пользователя и отправлять созданный большой языковой моделью ответ. Через эту подсистему будет проходить любой ввод и вывод всей платформы.

Подсистема «Пользовательский интерфейс». Для взаимодействия с подсистемой «Сервер», пользователь будет использовать подсистему «Пользовательский интерфейс». Эта подсистема позволяет вводить вопрос с клавиатуры мобильного устройства, либо голосовым вводом и выводить ответ системы на экран.

Подсистема «База знаний». Для генерации своих ответов, большие языковые модели вместе с вопросом пользователя будут получать на вход дополнительную информацию из внешних источников. Эта информация будет содержаться и отправляться именно из подсистемы «База знаний».

Подсистема «Редактор базы знаний». Наполнение текстового содержимого подсистемы «База знаний» будет происходить при помощи подсистемы «Редактор базы знаний». Через него администратор платформы сможет загружать дополнительную информацию в виде электронных документов.

Уточненная схема взаимодействия подсистем представлена на рисунке 2.1.1.

Рисунок 2.1.1 – Схема взаимодействия подсистем «Сервер», «Пользовательский интерфейс», «База знаний» и «Редактор базы знаний»

Далее выделим подсистемы, взаимодействующие с «Редактором базы знаний».

Подсистема «Авторизация». Для получения доступа к подсистеме «Редактор базы знаний», необходимо пройти процесс авторизации, который позволяет выполнить подсистема «Авторизация».

Подсистема «Меню загрузки, удаления и скачивания документов». Данная подсистема позволяет администратору управлять текстовым содержимым подсистемы «База знаний».

Подсистема «Меню взаимодействия с ассистентом». Данная подсистема предоставляет доступ к цифровому ассистенту для тестирования результатов изменения подсистемы «База знаний».

Подсистема «Настройка параметров редактора». Редактор базы знаний позволяет производить одновременную настройку нескольких серверов. Для выбора настраиваемого сервера, администратор использует подсистему «Настройка параметров редактора».

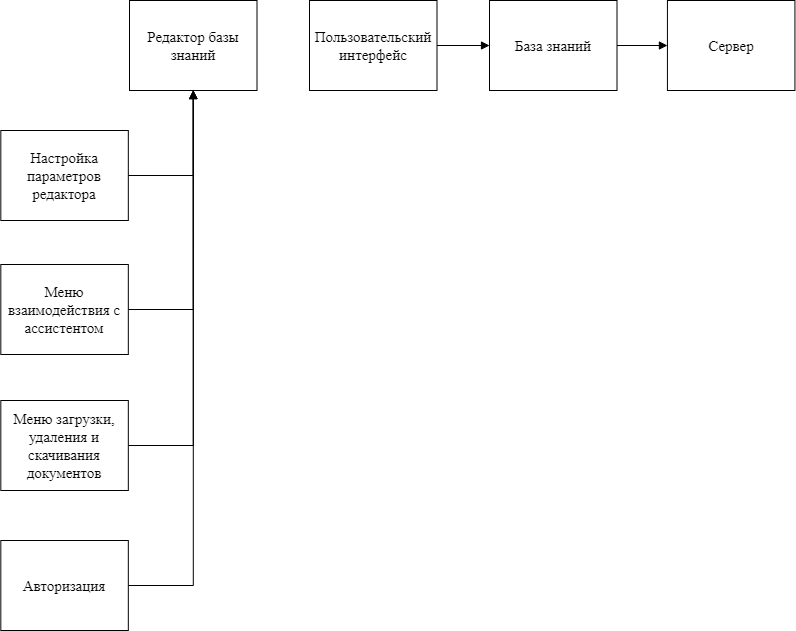
Уточненная схема взаимодействия подсистем представлена на рисунке 2.1.2.

Рисунок 2.1.2 – Схема взаимодействия подсистем, уточненная в соответствии с подсистемами, реализующими взаимодействие с редактором базы знаний

Далее рассмотрим подсистемы «Пользовательский интерфейс».

Подсистема «Главный экран». Используется пользователем для навигации между различными экранами приложения.

Подсистема «База данных». Данная подсистема выполняет сохранение данных пользователя для дальнейшего воспроизведения, такие как диалоги с ассистентом и реакции на сообщения.

Подсистема «Чат с ассистентом». Ключевой частью подсистемы «Пользовательский интерфейс» является общение с ассистентом, которое происходит через подсистему «Чат с ассистентом».

Подсистема «Управление реквизитами». Содержит в себе реквизиты, необходимые для авторизации при общении с ассистентом чтобы предотвратить несанкционированный доступ к серверу.

Подсистема «Автоматическая отправка ошибок». При возникновении критических ошибок в приложении, данная подсистема собирает необходимую информацию и отправляет на адрес электронной почты разработчика.

Подсистема «Экран настроек». Позволяет пользователю настроить внешний вид и функционирование мобильного приложения.

Уточненная схема взаимодействия подсистем представлена на рисунке 2.1.3.

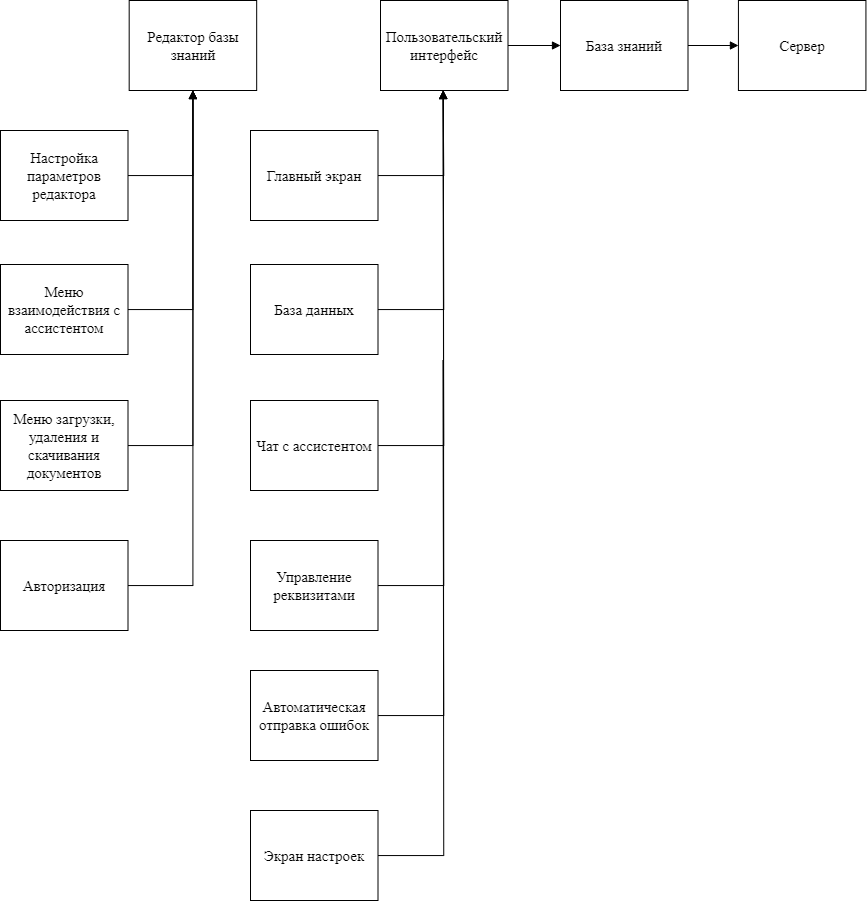


Рисунок 2.1.3 – Схема взаимодействия подсистем, уточненная в соответствии с подсистемами, реализующими пользовательский интерфейс

Подсистемы «База знаний».

Подсистема «Создание коллекций». При подключении к базе знаний необходимо создать коллекцию, в которой будут сохранятся записи одного цифрового ассистента.

Подсистема «Загрузка литературы с диска». Для наполнения базы знаний необходимо загрузить документы в электронном виде в память ЭВМ для дальнейшей обработки.

Подсистема «Функция векторизации». После загрузки документов с диска, для проведения поиска по ним, необходимо сначала их преобразовать. За это ответственна подсистема «Функция векторизации».

Подсистема «Поиск по базе знаний». Является основной функцией подсистемы «База знаний». При получении запроса по наполнению контекста от сервера, происходит поиск информации по базе знаний.

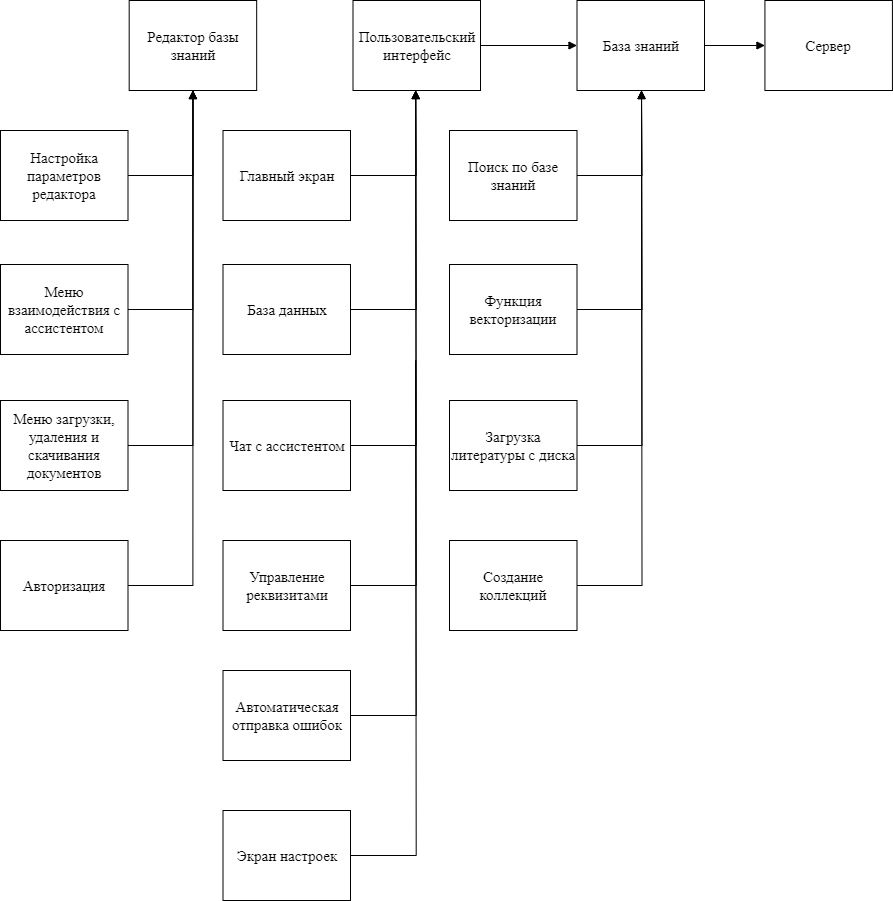
Уточненная схема взаимодействия подсистем представлена на рисунке 2.1.4.

Рисунок 2.1.4 – Схема взаимодействия подсистем, уточненная в соответствии с подсистемами, реализующими базу знаний

Подсистемы «Сервер».

Подсистема «Цикл событий». Сервер постоянно ожидает подключения и обрабатывает их при помощи подсистемы «Цикл событий».

Подсистема «Подключение к базе знаний». Подсистема «База знаний» запускается отдельно от сервера и для полноценной работы цифрового ассистента необходимо установить с ней подключение.

Подсистема «Настройки ключей и адресов». Данная подсистема необходима для настройки сетевых подключений к серверу и к базе знаний, а также настройке доступа к внешним искусственным интеллектам.

Подсистема «Интерфейсы для взаимодействия». Взаимодействие с подсистемой «Сервер» будет происходить при обращении к данной подсистеме. Данные интерфейсы представляют собой конечные точки для подключения, которые будут доступны при запуске подсистемы «Сервер». Для доступа к ним необходимо предоставить реквизиты подключения, которые описаны в подсистеме «Настройки ключей и адресов».

Уточненная схема взаимодействия подсистем представлена на рисунке 2.1.5.

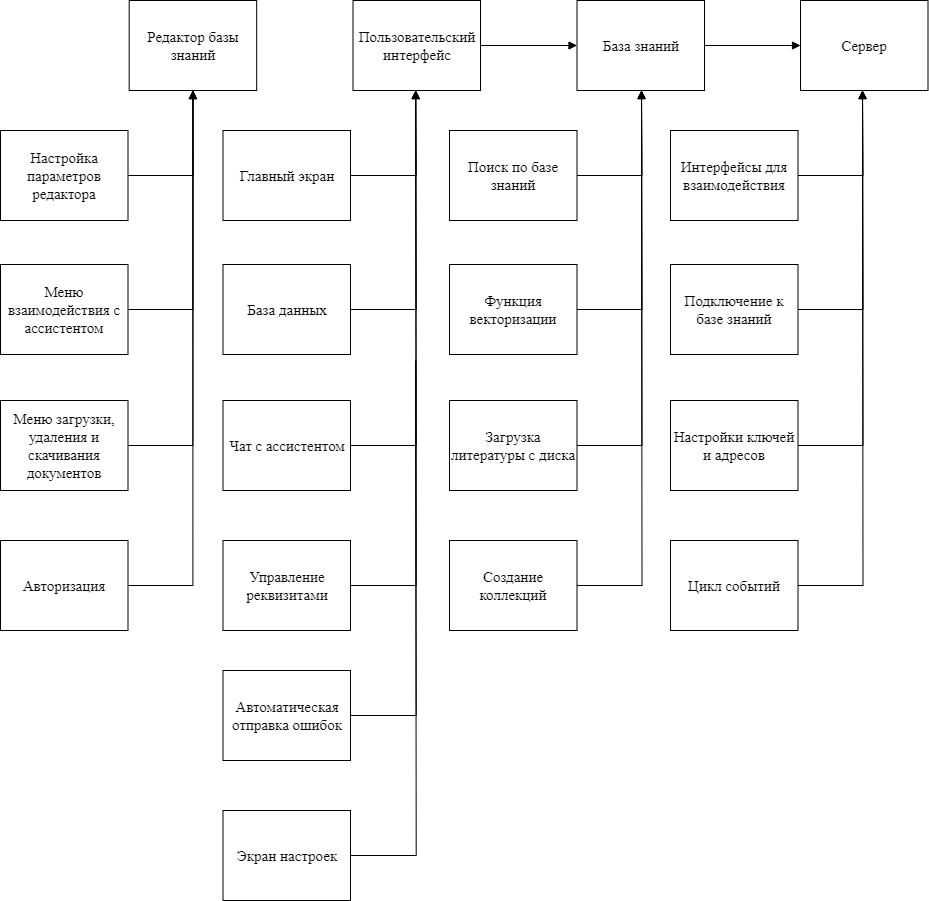


Рисунок 2.1.5 – Схема взаимодействия подсистем, уточненная в соответствии с подсистемами, реализующими сервер

Рисунок 2.1.5 представляет собой итоговую схему взаимодействия различных подсистем, задействованных в работе платформы по созданию цифровых ассистентов, основанных на больших языковых моделях. Схема отражает основные этапы и компоненты процесса создания ассистента, включая настройку базы знаний, получение ввода от пользователя, его обработку и отправку ответа сервером.

Взаимодействие между различными подсистемами обеспечивает эффективную работу всего процесса создания, настройки и использования цифрового ассистента.

* 1. **ПРОЕКТИРОВАНИЕ ПРОГРАММЫ**

При разработке платформы по созданию цифровых ассистентов была использована модульная архитектура, основанная на принципах взаимодействия клиента и сервера. Этот подход позволил разбить систему на отдельные модули, каждый из которых отвечает за определенную функциональность, и организовать их взаимодействие с помощью сетевых объектов, называемых запросами.

**Модульная архитектура:**

Программа была разделена на несколько основных модулей: сервер, пользовательский интерфейс в виде мобильного приложения, база знаний и редактор базы знаний.

Взаимодействие между модулями осуществляется через сетевые объекты, называемых запросами. Отправка запросов происходит по протоколу Hypertext Transfer Protocol (HTTP). На использовании данного протокола построена известная всемирная сеть Интернет. Например, пользовательский интерфейс отправляет запрос на сервер, который содержит в себе реквизиты авторизации и вопрос пользователя. В ответ, сервер отправляет запрос на пользовательский интерфейс, который содержит в себе ответ искусственного интеллекта. Такой подход обеспечивает четкое разделение ответственностей между компонентами и упрощает интеграцию и тестирование отдельных модулей.

В целом, использование модульной архитектуры и клиент-серверного подхода при разработке платформы по созданию цифровых ассистентов обеспечивает модульность, расширяемость, простоту интеграции сторонних библиотек и повышенную тестируемость.

**Клиент-серверная архитектура платформы:**

Каждый модуль платформы выполняет определённую ему часть работ. Это стало возможным благодаря применению клиент-архитектурного подхода к разработке.

Модули, такие как сервер, пользовательский интерфейс, редактор базы знаний и база знаний представляют собой логические единицы кода, отвечающие за определённые задачи и обладающие своей внутренней структурой, различной от других.

**Взаимодействие между модулями:**

Взаимодействие между модулями осуществлялось через HTTP-запросы. Эти запросы отправляются через локальную сеть, либо через всемирную сеть Интернет.

Например, редактор базы знаний может отправить запрос в базу знаний по добавлению очередного документа в её коллекцию. База знаний в это время вернёт ответный запрос, содержащий результат операции.

**Использование сторонних библиотек:**

В программе были использованы сторонние библиотеки, такие как fastapi, kivy и langchain, которые предоставляют готовые классы и функциональность для создания цикла событий сервера, пользовательского интерфейса редактора базы знаний и обращения к искусственному интеллекту.

Эти библиотеки были интегрированы в систему с помощью объектно-ориентированного подхода, создавая экземпляры их классов и взаимодействуя с ними через методы. Руководствуясь принципами объектно-ориентированного программирования (ООП), были созданы объекты этих библиотек, используя методы которых выполнялось взаимодействие с ними.

Применение модульной клиент-серверной архитектуры и объектно-ориентированного подхода при разработке программы принесло следующие преимущества:

* **Модульность**: Разделение системы на отдельные модули упростило разработку, тестирование и сопровождение кода, поскольку каждый модуль мог быть изменен независимо от остальных.
* **Расширяемость**: благодаря инкапсуляции и наследованию, реализованным с помощью ООП, систему можно легко расширять, добавляя новые классы или наследуя функциональность от существующих.
* **Повторное использование кода**: Классы и модули могут быть повторно использованы в других проектах или приложениях, что повышает эффективность разработки.
* **Интеграция сторонних библиотек**: объектно-ориентированный подход упростил интеграцию сторонних библиотек, таких как fastapi, kivy и langchain, в общую систему.
* **Тестируемость**: модульная структура и инкапсуляция облегчили процесс тестирования отдельных компонентов системы, что повысило качество кода и упростило отладку.

Таким образом, применение модульной клиент-серверной архитектуры и принципов объектно-ориентированного программирования при разработке платформы по созданию цифровых ассистентов обеспечило модульность, расширяемость, повторное использование кода, простоту интеграции сторонних библиотек и повышенную тестируемость. Это способствовало повышению качества, поддерживаемости и масштабируемости системы в целом.

1. **РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММНЫХ СРЕДСТВ**

**3.1 РЕАЛИЗАЦИЯ ПРОГРАММНЫХ СРЕДСТВ**

Реализация программных средств была направлена на удобство и скорость создания очередного цифрового ассистента. Основой для этого послужили средства программирования, предоставляющие необходимые инструменты для создания базы знаний и сервера, их настройки и использования через пользовательский интерфейс.

**Создание базы знаний**

База знаний была выполнена с использованием библиотеки chroma, которая предоставляет доступ к векторной базе знаний через код сервера на Python. База знаний создаётся в виде контейнера docker, для этого достаточно его скачать и запустить через консоль:

Docker pull chromadb/chroma

Docker run –p 8000:8000 chromadb/chroma

**Создание сервера и запуск цикла событий**

Чтобы начать принимать пользовательские запросы, необходимо запустить сервер (цикл событий) цифрового ассистента, это можно выполнить при помощи следующего кода в Python, где server\_title, server\_version, server\_description, host\_ip, host\_port – настраиваемые параметры сервера:

App = FastAPI ( title = server\_title , version = server\_version , description = server\_description )

if \_\_name\_\_ == "\_\_main\_\_":

import uvicorn

uvicorn.run(app, host=host\_ip, port=host\_port)

**Подключение к базе знаний и создание коллекций**

Для использования базы знаний в сервере необходимо её подключить и создать базовую коллекцию, используя веб-клиент chroma. Это можно выполнить используя следующий код Python:

client = chromadb.HttpClient(host=chroma\_host, port=chroma\_port, ssl=chroma\_ssl, settings=Settings(allow\_reset=chroma\_allow\_reset))

try:

collection = client.create\_collection(chroma\_collection, embedding\_function=chroma\_embedding\_function)

except Exception as e:

if (str(e).startswith("{\"error\":\"UniqueConstraintError")):

print("Specified collection already exists, recreating...")

client.delete\_collection(chroma\_collection)

collection = client.create\_collection(chroma\_collection, embedding\_function=chroma\_embedding\_function)

print("collection created successfully")

**Загрузка литературы с диска и функция векторизации**

Загрузка литературы с диска происходит на сервере, после чего вся литература отправляется на функцию векторизации базы знаний. Код вышеописанных действий выглядит следующим образом:

print("loading docs from folder")

loader = DirectoryLoader(

literature\_dir,

show\_progress=True,

use\_multithreading=True,

silent\_errors=True

)

documents = loader.load()

if(len(documents)<1):

raise RuntimeError("The literature folder is empty. (/"+literature\_dir+")")

text\_splitter = RecursiveCharacterTextSplitter(

chunk\_size=chunk\_size,

chunk\_overlap=chunk\_overlap,

separators=[" ", ",", "\n", ". "]

)

docs = text\_splitter.split\_documents(documents)

print("Filling collection")

for doc in docs:

collection.add(

ids=[str(uuid.uuid1())], metadatas=doc.metadata, documents=doc.page\_content

)

**Пользовательский интерфейс**

Пользовательским интерфейсом является мобильное приложение для ОС Android. Код этого приложения написан на языке программирования java. Мобильное приложение содержит множество экранов, называемых активностями.

Основной активностью этого приложения является активность чата с цифровым ассистентом. Для общения с цифровым ассистентом пользователь вводит текстовый вопрос, который записывается в переменную question. В последующем, эта переменная передаётся на сервер при помощи создания http-запроса, который содержит в себе ключ доступа к серверу и сам текстовый вопрос. Код этой функции отправки сообщения следующий:

void sendMessage(String question){

service.execute(() -> {

String serverKey = CredentialsManager.getRandomServerKey();

RequestBody body = new MultipartBody.Builder()

.setType(MultipartBody.FORM)

.addFormDataPart("key", serverKey)

.addFormDataPart("question", question)

.build();

Request request = requestBuilder.build();

try (Response response = client.newCall(request).execute()) {

if (response.isSuccessful()) {

JSONObject responseJson = new JSONObject(response.body().string());

addResponse(responseJson.get("output").toString());

} else {

throw new IOException("Запрос к серверу не был успешен: " +

response.code() + " " + response.message());

}

} catch (IOException e) {

Log.d(TAG, "Ошибка подключения: " + e);

} catch (JSONException e) {

throw new RuntimeException(e);

}

});

}

**Обработка пользовательского запроса**

После отправки запроса с клиента на сервер, сервер начинает его обработку. При обработке он получает из запроса пользовательский текст, с которым начинает поиск по базе знаний. База знаний в это время проводит векторизацию вопроса и сравнивает выходной вектор с существующими векторами в базе, после чего находит наиболее подходящий вектор и отправляет на сервер привязанный к нему текст. Этим текстом и является найденный по теме вопроса текст из документа. Код поиска по базе выглядит следующим образом:

def findRelevantDocs(query, query\_type, return\_source, n=2):

relevantDocs = ""

sources = ""

docs = []

if(query\_type==0):

docs = embeddings\_retriever.invoke(input=query)

if(query\_type==1):

docs = bm25\_retriever.invoke(input=query)

if(query\_type==2):

docs = ensemble\_retriever.invoke(input=query)

if(return\_source==True):

k = 0

for doc in docs:

k+=1

if k>documents\_returned\_amount:

break

if(doc.metadata.get("source")):

current\_doc\_source = doc.metadata.get("source").replace("literature/", "").replace("literature\\", "")

if (sources==""): sources += current\_doc\_source

else:

if (current\_doc\_source) not in sources:

sources += "; " + current\_doc\_source

relevantDocs = relevantDocs + doc.page\_content + ". "

return [relevantDocs, sources]

if(return\_source==False):

k = 0

for doc in docs:

k+=1

if k>documents\_returned\_amount:

break

relevantDocs = relevantDocs + doc.page\_content + ". "

return relevantDocs

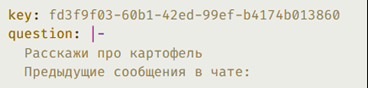
Изображение запроса отправленного пользователем на сервер и запроса после процесса обработки (дополнения контекстом) представлены на рисунках 3.1.1. и 3.1.2. соответственно.

Рисунок 3.1.1 – изображение пользовательского запроса до обработки

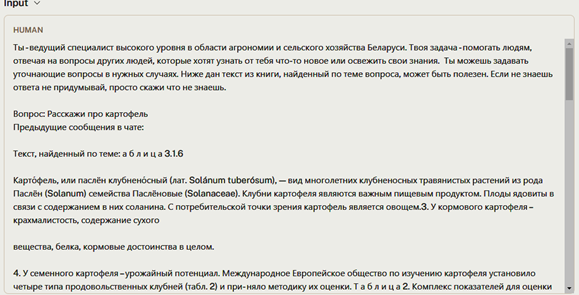


Рисунок 3.1.2 – изображение пользовательского запроса после дополнения контекстом через базу знаний

**Отправка запроса на сервер искусственного интеллекта**

После всех обработок и дополнения контекстом, пользовательский запрос отправляется на сервер искусственного интеллекта, где будет сгенерирован результат в виде текстового ответа, который в последующем отправляется обратно пользователю и отображается в пользовательском интерфейсе. Код получения запроса и отправки ответа будет следующим:

async def invokeModel(key: str = Form(...), question: str = Form(...)):

if not key in server\_keys:

raise HTTPException(status\_code=403, detail="Invalid or missing key.")

try:

print("received question'"+question+"'")

relevantDocs = findRelevantDocs ( question, QUERY\_EMBEDDINGS, return\_source=False)

formattedPrompt = template.format(context=relevantDocs, question=question)

try:

modelResponse = random.choice(chatGPT\_models).invoke(formattedPrompt).content

except Exception as e:

print("SOMETHING WENT WRONG WITH RESPONSE, using gigachat")

modelResponse = random.choice(giga\_models).invoke(formattedPrompt).content

answer = {"output":modelResponse}

print("returning response for '"+question+"'")

return JSONResponse(answer)

except Exception as e:

print(e)

raise HTTPException(status\_code=500, detail=str(e))

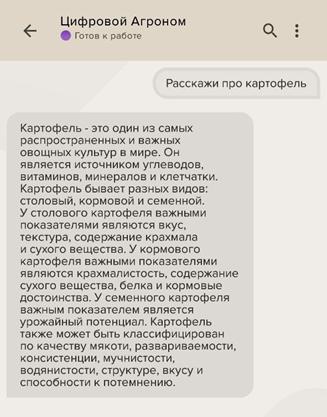
Изображение отправленного пользователем запроса и полученного на него ответа от искусственного интеллекта показано на рисунке 3.1.3.

Рисунок 3.1.3 – Изображение полученного ответа на запрос пользователя в пользовательском интерфейсе

**Управление базой знаний**

База знаний в этой работе используется для наполнения запроса пользователя дополнительной информацией. Этот процесс называется дополнением контекста.

Управление базой знаний происходит через сервер, используя редактор базы знаний как интерфейс администрирования. Редактор манипулирует выделенными для этого интерфейсами сервера, которые используются для загрузки, удаления и сохранения документов в базе знаний. Код на сервере для этих интерфейсов представлен в виде методов uploadLiterature, listLiterature, downloadLiterature, previewLiterature и deleteLiterature. Эти методы отвечают за загрузку, просмотр списка, скачивание, предпросмотр и удаление литературы соответственно.

Каждый метод начинается с проверки отправленного пользователем ключа доступа, если такового не существует на сервере, значит запрос на изменение базы знаний будет отклонён.

@app.post("/literature/list")

async def listLiterature(key: str = Form(...)):

if not key in server\_keys:

raise HTTPException(status\_code=403, detail="Invalid or missing key.")

try:

onlyfiles = [f for f in listdir(literature\_dir) if isfile(join(literature\_dir, f))]

return {"message":onlyfiles}

except Exception as e:

raise HTTPException(status\_code=500, detail="Error: " + str(e))

**Визуализация и мониторинг**

В процессе разработки платформы возникла потребность в отслеживании ошибок мобильного приложения, так как при возникновении критической ошибки на мобильном устройстве, находящемся вне локальной сети компьютера разработчика, такую ошибку будет сложно отследить и выяснить причину её возникновения.

Для облегчения процесса удалённой отладки было принято решение встроить систему автоматического сбора и отправки информации по ошибкам ACRA. Такая система позволяет увидеть разработчику, в какой части кода возникла ошибка, даже если пользователь находится в другом городе.

Код такой системы представлен в следующем виде:

ACRA.init(this, new CoreConfigurationBuilder()

.withBuildConfigClass(BuildConfig.class)

.withReportFormat(StringFormat.KEY\_VALUE\_LIST)

.withReportSendFailureToast("Не удалось отправить сообщение.")

.withReportSendSuccessToast("Сообщение отправлено успешно.")

.withReportContent( ReportField.PHONE\_MODEL,

ReportField.BRAND,

ReportField.ANDROID\_VERSION,

ReportField.TOTAL\_MEM\_SIZE,

ReportField.AVAILABLE\_MEM\_SIZE,

ReportField.STACK\_TRACE,

ReportField.LOGCAT )

.withPluginConfigurations(

new DialogConfigurationBuilder()

.withText("Отправить E-mail разработчикам с информацией об ошибке?")

.withResIcon(R.drawable.baseline\_bug\_report\_24)

.withTitle("Произошла непредвиденная ошибка")

.build(),

new MailSenderConfigurationBuilder()

//required

.withMailTo("digital.agro.development@gmail.com")

.withReportAsFile(true)

.withReportFileName("Crash.txt")

.withSubject("App Crash ACRA")

.withBody("(можете оставить комментарий о произошедшем, убрав этот текст в скобках)\n")

.build() ) );

Эта система автоматически отправляет информацию об ошибке на электронную почту разработчика. Результатом работы такой системы является повышение скорости устранения ошибок кода и улучшение взаимодействия между пользователем и разработчиком.

**Редактор базы знаний**

Для внесения изменений в базу знаний цифрового ассистента был разработан редактор базы знаний. Он взаимодействует с интерфейсами наполнения базы знаний сервера. Для начала работы с редактором базы знаний, необходимо авторизоваться в приложении. Изображение внешнего вида окна авторизации показано на рисунке 3.1.4.

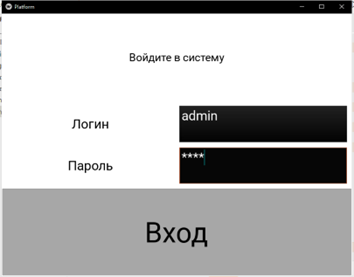
****

Рисунок 3.1.4 – Изображение внешнего вида окна авторизации редактора базы знаний

**Наполнение базы знаний через редактор**

Наполнение базы знаний с помощью редактора происходит через меню взаимодействия с базой знаний. В этом меню можно загрузить, скачать и удалить файл. Также на каждый пункт меню предоставляется краткая справка по нажатию на соответствующую кнопку. Внешний вид окна взаимодействия представлен на рисунке 3.1.5.

****

Рисунок 3.1.5 – Изображение внешнего вида окна взаимодействия с базой знаний

**Взаимодействие с цифровым ассистентом через редактор**

Редактор базы знаний позволяет получить доступ к цифровому ассистенту для тестирования результатов изменения базы знаний. Внешний вид окна взаимодействия с цифровым ассистентом представлен на рисунке 3.1.6.

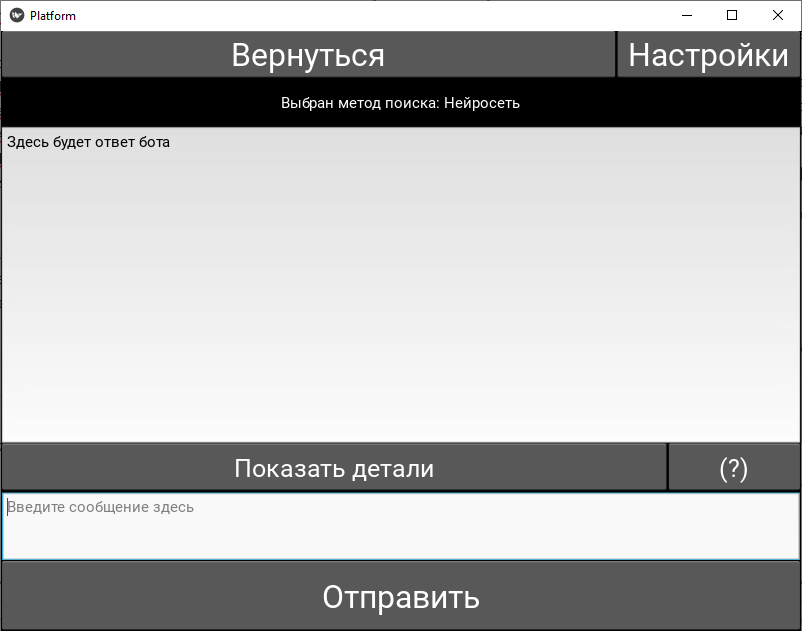
****

Рисунок 3.1.6 – Изображение внешнего вида окна взаимодействия с цифровым ассистентом

Таким образом реализована программная платформа по созданию цифровых ассистентов, основанных на больших языковых моделях. Платформа позволяет создать сервер, базу знаний, настроить их и подключиться к ним через редактор и пользовательский интерфейс в виде мобильного приложения для ОС Android.

**3.2 НАПОЛНЕНИЕ БАЗЫ ЗНАНИЙ И ТЕСТИРОВАНИЕ ПЛАТФОРМЫ**

После выполнения всех мероприятий разработки программного обеспечения, которое позволит функционировать платформе, было принято решение провести тестирование влияния наполненности базы знаний на ответы цифровых ассистентов.

При помощи организации вопросов сотрудничества со стороны лаборатории робототехники учреждения образования «Брестский государственный технический университет», при партнёрстве с компанией ООО «Технологии земледелия», был собран перечень сельскохозяйственных регламентов и другой литературы, которая позволит создать первого цифрового ассистента, ориентированного на тематику агрономии и сельского хозяйства. Такой ассистент повысит эффективность работы агронома-человека, уменьшив время поиска нужной информации и повысит урожайность сельскохозяйственных предприятий, увеличив количество правильно принятых решений. Такого цифрового ассистента было принято назвать «цифровой агроном».

В базе знаний «цифрового агронома» находится 161 документ, каждый отобран специалистами-агрономами компании «Технологии земледелия».

Тестирование ответов «цифрового агронома» было проведено в сотрудничестве со студентами и преподавателями учреждения образования «Белорусская государственная орденов Октябрьской Революции и Трудового Красного Знамени сельскохозяйственная академия», которая находится в г. Горки. Сотрудники компании «Технологии земледелия» установили договорённость с преподавателями академии на тестирование. Был создан перечень из 19 тестовых вопросов на тему сельского хозяйства, борьбы с вредителями и недугами растений, свойствам почвы и методам обработки почвы.

На каждый тестовый вопрос было получено 18 оценок, 17 из которых получили студенты 2-3 курсов сельскохозяйственной академии, оставшаяся оценка получена на ответ «цифрового агронома». Следует уточнить, что преподаватель, который проводил оценку, не знал какой из ответов написан разработанной системой.

Средний балл ответа системы цифрового ассистента составил 6,56 по десятибалльной шкале, не имея при этом неудовлетворительных оценок.

Лучше всего «цифровой агроном» ответил на вопросы, на которые студенты в среднем набирали меньше всего баллов. Хуже всего «цифровой агроном» отвечал на более лёгкие для студентов вопросы. Это может служить знаком о наличии конфликтующих фактов в собранной базе знаний и уже имеющихся знаниях искусственного интеллекта (для тестов был выбран chatgpt версии 3.5-turbo), которые ИИ предпочитает при ответе на вопросы с большим контекстом. Другой причиной такого явления может являться неправильно собранная база знаний «цифрового агронома», ведь учебники в ней были собраны из открытых источников сети Интернет, которые возможно не подходят для обучения в сельскохозяйственной академии.

Таким образом, ответ «цифрового агронома» на вопрос «Система обработки почвы» оценили в 9 баллов. Ответы студентов на этот же вопрос в среднем заработали 5,76 баллов (вопрос более сложный для студентов, чем остальные). Ответ «цифрового агронома» на вопрос «Предупредительные меры борьбы» оценили в 4 балла. Ответы студентов на этот вопрос заработали в среднем 7,29 баллов, что указывает на лёгкость данного вопроса для студентов.

Ответы «цифрового агронома» с пустой базой знаний не дали результата, который можно оценить положительно, что доказывает полезную работу наполнения базы знаний источниками информации.

В сотрудничестве с компанией «Технологии земледелия» началась разработка нового цифрового ассистента, который будет отвечать на технические вопросы, связанные с сельскохозяйственными машинами, тракторами, навесными орудиями для них. Название такого цифрового ассистента установили «цифровой инженер».

Так как платформа подразумевает удобное создание новых цифровых ассистентов, создание «цифрового инженера» не заняло много времени. Был запущен новый сервер, новая база знаний, собрана новая литература и отредактирован пользовательский интерфейс в другие цвета, вместо зелёного установлен красный цвет как основной. Работа по созданию нового цифрового ассистента заняла около часа, скорость подбора литературы для новой базы знаний не зависит от платформы.

Результаты проведения тестирования «цифрового агронома» в сельскохозяйственной академии занесены в таблицу 3.2.1.

Таблица 3.2.1 – Результаты тестирования «цифрового агронома»

|  |  |  |
| --- | --- | --- |
| **Вопрос** | **Оценка «цифрового агронома»** | **Средний балл студентов академии** |
| Законы научного земледелия, их значение и применение | 7 | 5,941176471 |
| Показатели плодородия почв. | 7 | 6,352941176 |
| Биологические особенности сорных растений | - | 6,705882353 |
| Классификация сорных растений | 5 | 6,705882353 |
| Предупредительные меры борьбы | 4 | 7,294117647 |
| Истребительные меры борьбы | 6 | 7,941176471 |
| Методы учета засоренности посевов | 8 | 7,529411765 |
| Пороги вредоносности сорных растений | 5 | 7,647058824 |
| причины чередования с.-х. культур в севообороте | 6 | 7,058823529 |
| предшественники с.-х. культур и их ценность | 5 | 6,823529412 |
| Промежуточные культуры | 7 | 7,764705882 |
| Полевые севообороты и их Виды | 7 | 6,411764706 |
| Кормовые севообороты и их Виды | 8 | 7,352941176 |
| Способы обработки почвы | 8 | 6,411764706 |
| Приемы обработки почвы | 7 | 5,764705882 |
| Система обработки почвы | 9 | 5,764705882 |
| Основная обработка почвы | 5 | 6,470588235 |
| предпосевная обработка почвы | 7 | 7,764705882 |
| Приемы послепосевной обработки или ухода за посевами | 7 | 7,588235294 |

Таким образом, была проведена объективная оценка ответов цифровых ассистентов, созданных платформой, которая была разработана в ходе выполнения дипломной работы.

1. **ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ**
   1. **ИСХОДНЫЕ ДАННЫЕ ДЛЯ РАСЧЕТА ЭКОНОМИЧЕСКОГО ЭФФЕКТА**

Наименование проекта – «Разработка программной платформы для создания виртуальных ассистентов на основе больших языковых моделей». В рамках данного дипломного проекта создается программное обеспечение, предназначенное для создания и настройки виртуальных ассистентов со своей собственной базой знаний. ПО включает в себя функции для взаимодействия с искусственным интеллектом через API, функции для запуска нейронных сетей – обработчиков функций векторизации, связи с пользовательским интерфейсом, а также редактирования базы знаний цифрового ассистента.

Среда разработки ПО – Visual Studio Code, язык программирования – Python. Проект включает в себя разработку собственного контекста для больших языковых моделей на основе библиотеки langchain, предобработку входных данных, а также получение и обработку http-запросов, используя для этого сервер, написанный при помощи библиотек uvicorn и fastapi на основе языка программирования Python.

Разработка программного продукта предусматривает выполнение всех ключевых этапов проектирования программного обеспечения, от технического задания до рабочего проекта.

Для оценки экономического эффекта от разработки и внедрения программного обеспечения будут выполнены следующие расчеты:

* Определение объема функциональных возможностей программного продукта.
* Расчет полной себестоимости разработки программного обеспечения, включая затраты на исследования, разработку, тестирование и оптимизацию.
* Определение отпускной цены программного продукта и расчет чистой прибыли от его реализации, учитывая потенциальный интерес исследовательских организаций и коммерческих компаний к подобным технологиям.

Проект направлен на создание инновационного продукта, который может найти широкое применение в множестве областей, от сельского хозяйства, до компьютерных наук, предоставляя новые возможности для настройки ИИ под свои нужды, не требуя при этом дополнительных сил по дообучению нейронных сетей.

**4.2 РАСЧЕТ ОБЪЕМА ФУНКЦИЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Общий объем ПО (Vo) определяется исходя из количества и объема функций, реализуемых программой (формула 4.1):

, (4.1)

где Vo– общий объем ПО,

Vi– объем функций ПО,

n – общее число функций.

Расчет общего объема программного обеспечения (количество строк исходного кода) предполагает определение объема по каждой функции. Чаще всего на стадии технико-экономического обоснования проекта невозможно рассчитать объем функций, тогда данный объем может быть получен на основании ориентировочной оценки имеющихся фактических данных по аналогичным проектам, выполненным ранее, или по тем применениям нормативов по каталогу функций.

На основании информации о функциях разрабатываемого ПО по каталогу функций определяется общий объем ПО. В зависимости от организационных и технологических условий, в которых разрабатывается ПО, корректируется объем на основе экспертных оценок.

Уточненный объем ПО (Vу) определяется по формуле 4.2:

, (4.2)

где Vуi – уточненный объем отдельной функции в строках исходного кода

(см. таблицу 4.1).

Таблица 4.1 – Перечень и объем функций программного обеспечения

|  |  |  |  |
| --- | --- | --- | --- |
| **Код функции** | **Наименование (содержание) функции** | **Объем функции строк исходного кода** | |
| **По каталогу (Vi)** | **Уточненный (Vyi)** |
| **1** | **2** | **3** | **4** |
| **1. Ввод, анализ входной информации, генерация кодов и процессор входного языка** | | | |
| 102 | Контроль, предварительная обработка и ввод информации | 280 | 55 |
| **3. Формирование и обработка файлов** | | | |
| 303 | Обработка файлов | 1 050 | 46 |
| 304 | Управление файлами | 5240 | 41 |
| 305 | Формирование файлов | 2130 | 31 |
| **6. Тестирование, проведение тестовых испытаний прикладных программ, вспомогательные программные функции** | | | |
| 601 | Проведение тестовых испытаний прикладных  программ в интерактивном режиме | 3 780 | 36 |
| **7. расчетные задачи, формирование и вывод на внешние носители документов сложной формы и файлом** | | | |
| **1** | **2** | **3** | **4** |
| 707 | Графический вывод результатов | 420 | 33 |
|  | ИТОГО | 12 900 | 242 |

Учитывая информацию, указанную в таблице 4.1, о функциях разрабатываемого программного обеспечения, уточненный объем ПО (Vу) составил 242 строк исходного кода вместо предполагаемого количества строк 6520.

* 1. **РАСЧЕТ ПОЛНОЙ СЕБЕСТОИМОСТИ ПРОГРАММНОГО ПРОДУКТА**

Стоимостная оценка программного средства у разработчика предполагает составление сметы затрат, которая включает следующие статьи расходов:

* Заработную плату (основную – и дополнительную – ).
* Отчисления на социальные нужды ().
* Материалы и комплектующие изделия ().
* Спецоборудование ().
* Машинное время ().
* Расходы на научные командировки ().
* Прочие прямые расходы ().
* Накладные расходы ().

Затраты на освоение и сопровождение программного средства ( и ).

Полная себестоимость () разработки программного продукта рассчитывается как сумма расходов по всем статьям с учётом рыночной стоимости аналогичных продуктов.

Основной статьёй расходов на создание программного продукта является заработная плата проекта (основная и дополнительная) разработчиков (исполнителей) (), в число которых принято включать инженеров-программистов, руководителей проекта, системных архитекторов, дизайнеров, разработчиков баз данных, Web-мастеров и других специалистов, необходимых для решения специальных задач в команде.

Расчёт заработной платы разработчиков программного продукта начинается с определения:

* Продолжительности времени разработки (), которое устанавливается

экспертным путем с учётом сложности, новизны ПП и фактически

затраченного времени. В данном дипломном проекте  –   60 дней.

* Количества разработчиков ПП. В данном дипломном проекте один разработчик.

Заработная плата разработчиков определятся как сумма основной и дополнительной заработной платы всех исполнителей.

Основная заработная плата каждого исполнителя определяется по формуле (4.3).

, (4.3)

где – тарифная ставка 1-го разряда рабочего (На дату написания дипломного проекта её значение равняется – 250 бел.руб.);

– тарифный коэффициент согласно тарифному разряду исполнителя;

– среднее количество рабочих дней в месяце;

– фонд рабочего времени исполнителя (продолжительность разработки программного модуля, дни);

– коэффициент премии, = 1,5.

Рассчитаем основную заработную плату инженера-программиста и техника-программиста согласно формуле 4.3. Тарифный коэффициент согласно 13 разряду инженера-программиста = 2,17. Продолжительность разработки программного продукта – 60 дней.

*Основная заработная плата инженера-программиста 13 разряда (КПР = 1,5):*

(бел. руб.)

Дополнительная заработная плата каждого исполнителя ( – 20%). Рассчитывается от основной заработной платы по формуле (4.4).

, (4.4)

*Дополнительная заработная плата инженера-программиста 13 разряда:*

(бел. руб.)

Результаты вычислений внесём в таблицу 4.2.

Таблица 4.2 – Расчет заработной платы

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Категория работников | Разряд | Тарифные коэффициент ( | , (дн.) | Коэффициент премии ) | (час.) | Заработная плата, бел. руб. | | |
| Основная | Дополнительная | Всего |
| Инженер-программист | 113 | 2,17 | 44 | 1,5 | 8 | 1484,2 | 296,8 | 1781 |
| Итого | - | - | - | - | - | 1484,2 | 296,8 | 1781 |

Таким образом, как видно из таблицы, заработная плата инженера- программиста составляет 1953 бел. руб.

Отчисления на социальные нужды () определяются по формуле (4.5) в соответствии с действующим законодательством по нормативу (35% – отчисления в ФСЗН):

, (4.5)

(бел. руб.)

Расходы по статье «Спецоборудование» () включает затраты на приобретение технических и программных средств специального назначения, необходимых для разработки методического пособия, включая расходы на проектирование, изготовление, отладку и другое.

В данном дипломном проекте для разработки платформы по созданию цифровых ассистентов приобретение какого-либо спецоборудования не предусматривалось. Так как спецоборудование не было приобретено, данная статья не рассчитывается.

По статье «Материалы и комплектующие изделия» () отражаются расходы на бумагу, картридж и красящие ленты для принтера, необходимые для разработки ПП. Норма расхода материалов в суммарном выражении определяются в расчете на 100 строк исходного кода. В данном дипломном проекте не рассчитывается.

Расходы по статье «Машинное время» () включают оплату машинного времени, необходимого для разработки и отладки программного продукта. Они определяются в машино-часах по нормативам на 100 строк исходного кода машинного времени.  определяется по формуле (4.6).

, (4.6)

где – цена одного машинного часа (1,6 бел. руб.);

– уточнённый общий объём функций строк исходного кода (LOC);

– норматив расхода машинного времени на отладку 100 строк кода, машино-часов. Принимается в размере 0,8.

(бел. руб.)

Расходы по статье «Научные командировки» () берутся либо по смете научных командировок, разрабатываемой на предприятии, либо в процентах от основной заработной платы исполнителей (10-15%). Так как в данном проекте научные командировки не предусмотрены, данная статься не рассчитывается.

Расходы по статье «Прочие затраты» () включают затраты на приобретение специальной научно-технической информации и специальной литературы и определяются по формуле (4.7).

(4.7)

Где:

НПЗ – норматив прочих затрат, НПЗ = 15%.

(бел. руб.)

Затраты по статье «Накладные расходы» () связаны с содержанием вспомогательных хозяйств, а также с расходами на общехозяйственные нужды. Определяется по нормативу в процентах к основной заработной плате по формуле (4.8).

, (4.8)

где – норматив накладных расходов, в данном дипломном проекте норматив накладных расходов равен 40%.

(бел. руб.)

Сумма вышеперечисленных расходов по статьям на программный продукт служит исходной базой для расчёта затрат на освоение и сопровождение программного продукта. Они рассчитываются по формуле (4.9).

, (4.9)

(бел. руб.)

Организация-разработчик участвует в освоении программного продукта и несёт соответствующие затраты, на которые составляется смета, оплачиваемая заказчиком по договору. Затраты на освоение определяются по установленному нормативу от суммы затрат по формуле (4.10).

, (4.10)

где – установленный норматив затрат на освоение. Для данного дипломного проекта принимается равной 5%.

176,74 (бел. руб.)

Организация-разработчик осуществляет сопровождение программного продукта и несёт расходы, которые оплачиваются заказчиком в соответствии с договором и сметой на сопровождение. Эти расходы рассчитываются по формуле (4.11).

, (4.11)

где – установленный норматив затрат на сопровождение программного продукта. Для данного дипломного проекта принимается равной̆ 5%.

(бел. руб.)

Полная себестоимость () разработки программного продукта рассчитывается как сумма расходов по всем статьям. Она определяется по формуле (4.12).

, (4.12)

(бел. руб.)

Результаты вычислений занесём в таблицу 4.3.

Таблица 4.3 – Себестоимость программного продукта

|  |  |  |
| --- | --- | --- |
| **Наименование статей затрат** | **Норматив, %** | **Сумма затрат, бел. руб.** |
| Заработная плата, всего | - | 1 953 |
| Основная заработная плата | - | 1627,5 |
| Дополнительная заработная плата | - | 325,5 |
| Отчисления на специальные нужды | 35% |  |
| Спецоборудование | Не применялось | - |
| Материалы | Не применялись | - |
| Машинное время | - | 3,1 |
| Научные командировки | Не планировались | - |
| Прочие затраты | 15% | 244,13 |
| Накладные расходы | 40% | 651 |
| Сумма затрат | - |  |
| Затраты на освоение | 5% |  |
| Затраты на сопровождение | 5% |  |
| Полная себестоимость | - |  |

В результате всех расчётов полная себестоимость ПП составила бел. руб.

## 4.4 РАСЧЕТ ЦЕНЫ И ПРИБЫЛИ ПО ПРОГРАММНОМУ ПРОДУКТУ

Для определения цены программного продукта необходимо рассчитать плановую прибыль, которая рассчитывается по формуле (4.13).

, (4.13)

где:

– полная себестоимость программного модуля, бел. руб;

R – уровень рентабельности программного модуля. В данном дипломном проекте уровень рентабельности равен 20%.

(бел. руб.)

После расчета прибыли от реализации по формуле (4.14) определяется прогнозируемая цена программного продукта без налогов.

, (4.14)

(бел. руб.)

Отпускная цена (цена реализации) программного продукта включает налог на добавленную стоимость и рассчитывается по формуле (4.15).

, (4.15)

где:

– налог на добавленную стоимость.

Для данного программного продукта рассчитывается по формуле (4.16).

, (4.16)

где:

– прогнозируемая цена, бел. руб.;

НДС – налог на добавленную стоимость. В настоящее время – это 20%.

(бел. руб.)

(бел. руб.)

Прибыль от реализации программного продукта за вычетом налога на прибыль является чистой прибылью (). Чистая прибыль остаётся организации-разработчику и представляет собой экономический эффект от создания нового программного продукта. Она рассчитывается по формуле (4.17).

, (4.17)

где – ставка налога на прибыль. В настоящее время он равен 20%.

(бел. руб.)

Все расчёты цены и прибыли по ПП сведены в таблицу 4.4.

Таблица 4.4 – Расчёт отпускной цены и чистой прибыли программного модуля

|  |  |  |
| --- | --- | --- |
| **Наименование статей затрат** | **Норматив, %** | **Сумма затрат, бел. руб.** |
| Полная себестоимость | - |  |
| Прибыль | 20 |  |
| Цена без НДС | - |  |
| НДС | 20 |  |
| Отпускная цена | - |  |
| Налог на прибыль | 20 | 155,53 |
| Чистая прибыль | - |  |

В ходе произведенных расчетов определены основные экономические показатели:

* Полная себестоимость – бел. руб.
* Прогнозируемая цена – 5599,09 бел. руб.
* Чистая прибыль – 622,12 бел. руб.

Разработанный программный продукт имеет малое количество конкурентов с более высокими ценами на их услуги. Таким образом, рассчитанная отпускная цена на программный продукт, разрабатываемой в рамках данного дипломного проекта, является конкурентоспособной. При расчете цены учтены отчисления в фонд социальной защиты, а также налоги, необходимые к уплате. К конечному итогу получаем окончательную цену продукта, равную 5599,09 белорусских рубля.

# 5. ЭНЕРГОСБЕРЕЖЕНИЕ

Уменьшение количества потребляемой энергии становится одной из важнейших проблем современности. Сегодня на энергосбережение направлено пристальное внимание мировой общественности. С появлением высокоиндустриального общества, опасное вмешательство человека в природу резко усилилось, расширился объём этого вмешательства, оно стало многообразнее и на современном этапе грозит стать глобальной опасностью для человечества. Расход невозобновляемых видов сырья повышается пропорционально росту количества фабрик и заводов. Ежедневно на обеспечение человечества достаточным количеством топлива и энергии направляются максимальные усилия. Ограниченность запасов топливно-энергетических ресурсов заставляет современного человека обратиться к энергосбережению как одному из основных элементов концепции развития мировой энергетики.

Энергосбережение означает рациональное использование энергоресурсов во всех звеньях преобразования энергии - от добычи первичных энергоресурсов до потребления всех видов энергии конечными пользователями.

На развитие хозяйствующих субъектов в нашей стране существенное негативное влияние оказывает высокая доля энергетических затрат в издержках производства, которая на промышленных предприятиях составляет в среднем 8-12% и имеет устойчивую тенденцию к росту в связи с большим моральным и физическим износом основного оборудования и значительными потерями при транспортировке энергетических ресурсов. Одним из определяющих условий снижения издержек на промышленных предприятиях и повышения экономической эффективности производства в целом является рациональное использование энергетических ресурсов.

Энергосберегающий путь развития экономики возможен только при формировании и последующей реализации программ энергосбережения на отдельных предприятиях. Уменьшение потребления энергии позволяет без ущерба для производства повысить удельный вес собственных энергоресурсов в общем количестве потребляемой энергии.

Координацию работ в этом направлении и осуществление надзорных функций осуществляет Комитет по энергоэффективности Республики Беларусь и его территориальные областные управления. Под руководством комитета разработана Республиканская программа по энергосбережению, которая является фундаментом для внедрения методов эффективного и рационального использования энергии во всех сферах деятельности, связанной с производством, передачей, распределением и потреблением энергии.

Энергосбережение (экономия электроэнергии) – реализация правовых, организационных, научных, производственных, технических и экономических мер, направленных на рациональное использование и экономное расходование топливно-энергетических ресурсов и на вовлечение в хозяйственный оборот возобновляемых источников энергии. Энергосбережение – важная задача по сохранению природных ресурсов.

Энергоэффективность – эффективное использование энергетических ресурсов – достижение экономически оправданной эффективности использования технико-экономических ресурсов с модернизацией существующей техники и технологии и при соблюдении требований к охране окружающей среды.

Более 30 миллиардов киловатт-часов энергии ежегодно тратятся впустую только из-за того, что люди ленятся вовремя выключить свой компьютер. Если бы к данной проблеме относились серьезнее, то, кроме всего прочего, это помогло бы сэкономить более 3 млрд долларов. Выбросы СО2 от сжигания топлива для генерации электроэнергии для питания всего 15 среднестатистических компьютеров равны таковым от одного автомобиля.

Из одного барреля нефти (160 литров) вырабатывается около 556 кВт•ч электроэнергии. Мощность, потребляемая современным системным блоком, в среднем достигает 200 Вт. Добавьте к этому еще 80 Вт, если используется ЭЛТ-монитор (значение для жидкокристаллических моделей примерно в два раза меньше). Таким образом, среднестатистический ПК нуждается примерно в 1 кВт каждые 4 часа. Если оставить компьютер работать в режиме 24/7 (а ведь именно так многие пользователи и поступают), он будет «съедать» один баррель нефти каждые 90 дней. При оптимизации уровня его энергопотребления можно увеличить длительность данного периода до 6 месяцев и более. Каждый использованный на питание ПК киловатт электроэнергии условно равен попаданию в атмосферу порядка 0,68 кг CO2, что за день составит примерно 8 кг, за год 1,49 тонны CO2, и это всего от одного компьютера!

Из-за удорожания электроэнергии и ужесточения экологических стандартов на энергетическое и радиационное загрязнение среды на самых больших рынках развитых стран, сегодня все производители переходят на технологии уменьшения потребления электроэнергии и ПК, и периферией, особенно в моменты простоя и ожидания. Это касается и мониторов, и принтеров, и всяческой бытовой техники.

**Энергосбережение в центрах обработки данных (ЦОД) или дата-центрах**

По подсчетам экспертов, на строительство и дальнейшую эксплуатацию крупного дата-центра в течение пяти лет может уходить до 15 и более миллионов долларов. Расходы на оплату электроэнергии обычно занимают вторую строку. По разным оценкам, это 20-25%.

Существует ряд мер, которые помогают уменьшит энергозатраты:

* Применение эффективных систем кондиционирования и охлаждения ЦОД, например, систем свободного охлаждения. Они ощутимо дороже стандартных решений и имеют большие габариты, но за счет «холодных» месяцев (до полугода) в разы снижают потребление электроэнергии, а потому довольно быстро окупаются.
* Оптимизация количества единиц техники. Это снижает потребляемое электричество и уменьшает количество выделяемого тепла, что напрямую связано с инженерной инфраструктурой, как наиболее энергоемкой и критичной системой для обеспечения бесперебойной работы всего оборудования в ЦОД.
* Перенос ЦОД из мегаполисов в регионы.
* Применение технологии виртуализации. Данная технология позволяет значительно снижать затраты на оборудование и ПО ЦОД, повышать коэффициент загрузки серверов, получать большую степень доступности, а также существенно уменьшает энергопотребление.
* Использование солнечных батарей в качестве дополнительного и экологически чистого источника энергии.

**Энергосбережение на серверах**

Blade-серверы – это ближайшее будущее традиционных серверных технологий. С одной стороны, blade-серверы - альтернатива решениям на базе разрозненных серверов начального и среднего уровня, с другой - дорогостоящим многопроцессорным серверам с симметричной архитектурой. Как правило, blade-серверы отличаются высокой эффективностью энергосбережения.

В новых системах используются новейшие микропроцессоры AMD и Intel с низким напряжением питания, а также реализована инновационная технология хранения для blade-серверов на базе устройств флэш-памяти с низким энергопотреблением.

Практически все Blade-решения имеют собственные интеллектуальные системы управления охлаждением и питанием, что позволяет оптимизировать затраты на них в зависимости от потребностей. Помимо значительной экономии электроэнергии (до 20%), ограничения тепловыделения и потребности в системе охлаждения (т.е. сокращения долгосрочных расходов на эксплуатацию), данный подход существенно экономит время IT-персонала и, таким образом, снижает совокупную стоимость владения.

В последние годы такие производители как HP, IBM, Sun, Intel, AMD и другие интенсивно вкладывают деньги в использование энергосберегающих технологий, что вместе с технологией виртуализации серверов дает ощутимую экономию электроэнергии.

Виртуализация серверов – возможность одному компьютеру выполнять работу нескольких. Выигрыш достигается за счет более эффективного использования вычислительных ресурсов; уменьшения количества оборудования при увеличении производительности; снижения расходов на поддержку ИТ; снижение энергозатрат.

По данным IBM, при стандартном классическом использовании серверов, эффективность использования процессоров у мэйнфрэйм (mainframe) составляет около 55%, Unix ~ 25%, x86 ~ 10%. В случае x86, при увеличении числа процессоров, доля времени использования каждого процессора становится еще меньше. Виртуализация и новые аппаратные решения позволяют повысить загрузку x86 серверов до 50-70%.

**Энергосбережение ПК**

Один из наиболее простых и эффективных способов уменьшить энергопотребление ПК – модернизировать его отдельные компоненты или систему в целом. Конечно, процесс затратен в финансовом плане, но бесспорное преимущество для пользователя – зачастую получение высшего уровня быстродействия. Так, процессоры Intel Core 2 Duo гораздо экономичнее, чем памятные Pentium D 8xx/9xx, которые с успехом могли заменить небольшие печки. При этом экономичность современных моделей на порядок выше. Еще один хороший пример – графические адаптеры. Ни для кого не секрет, что некогда топовый Radeon HD 2900 XT отличался хорошим «аппетитом» в плане энергопотребления. Современные видеокарты (к примеру, HD 4850), чипы которых выполнены по более тонкому техпроцессу, наряду с увеличением производительности позволяют сэкономить пару десятков ватт при работе системы.

Мощная современная видеокарта под полной нагрузкой требует столько же энергии, сколько остальные комплектующие ПК вместе взятые: от 110 до 270 Вт. Поэтому производители приступили к выпуску интеллектуальных видеокарт с управлением потребления электроэнергии в зависимости от нагрузки

Все вышесказанное касается и мониторов. Так, потребление устаревших ЭЛТ-моделей находится на отметке 70–100 Вт и более в зависимости от диагонали экрана. Жидкокристаллические обходят их на порядок в плане экономичности – им для работы достаточно 30–50 Вт. Логично: с увеличением диагонали дисплея растет и его энергопотребление. И если переход от 17-дюймового дисплея к 22-дюймовому почти незаметен (30 Вт против 40–45 у ЖК-версий), то уже 30-дюймовые экземпляры потребляют свыше 130 Вт электроэнергии.

Еще один вариант экономии – отказ от слишком мощных блоков питания, которые при обеспечении слабых и средних систем работают с низким КПД. Чем ближе показатели реальной потребляемой мощности от заявленной, тем коэффициент полезного действия выше.

Соблюдая несложные правила, можно значительно сократить энергопотребление своего ПК.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения дипломной работы была успешно разработана программная платформа для создания цифровых ассистентов на основе генерирующих нейронных сетей. Была использована библиотека LangChain для Python, что существенно облегчило взаимодействие с нейронными сетями и позволило подгружать контекст диалога для улучшения способностей искусственного интеллекта.

В результате работы над проектом были достигнуты все поставленные цели. Созданная платформа позволяет разветвлять области и тематики её применения без больших затрат сил по разработке новых интерфейсов и программных решений. Каждый созданный цифровой ассистент способен отвечать на разные темы, которые интересуют пользователя, а ответы подкреплены учебниками и другими документами в электронном формате, загруженными в его базу знаний.

Результат работы подтверждает, что нейронные сети могут быть успешно использованы для создания цифровых ассистентов, способных анализировать текстовые данные, созданная платформа может дополнять контекст пользователя и помогать искусственному интеллекту генерировать текст в тематике вопроса. Это открывает новые возможности для автоматической обработки текстовых данных, включая область клиентского обслуживания.

Таким образом, данная работа подтвердила возможность применения нейронных сетей в разработке программных продуктов и продемонстрировало их эффективность. В дальнейшем, эта работа может служить основой для создания новых и более сложных систем на основе нейронных сетей.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. ГОСТ 7.1-2003. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – Минск, 2004. – 48 с.
2. ГОСТ ЕСПД 19.103-77. Обозначение программ и программных документов.
3. ГОСТ ЕСПД 19.105-78. Общие требования к программным документам.
4. ГОСТ ЕСПД 19.301-2000. Программа и методика испытаний.
5. ГОСТ ЕСПД 19.401-78. Текст программы.
6. ГОСТ ЕСПД 19.402-78. Описание программы.
7. ГОСТ ЕСПД 19.502-78. Описание применения.
8. ГОСТ ЕСПД 19.504-79. Руководство программиста.
9. ГОСТ ЕСПД 19.505-79. Руководство оператора.
10. ГОСТ ЕСПД 19.508-79. Руководство по техническому обслуживанию.
11. ГОСТ ЕСПД 19.701-90. Схемы алгоритмов, программ, данных и систем.
12. LangChain // LangChain [Электронный ресурс]. – Режим доступа: https://python.langchain.com/v0.2/docs/introduction. –Дата доступа: 02.04.2024.
13. Visual Studio Code - Code Editing. Redefined // VS Code [Электронный ресурс]. – Режим доступа: https://code.visualstudio.com. – Дата доступа: 26.04.2024.
14. FastAPI // Tiangolo [Электронный ресурс]. – Режим доступа: https://fastapi.tiangolo.com. – Дата доступа: 09.05.2024.
15. What is Retrieval-Augmented Generation aka RAG // NVIDIA Blogs [Электронный ресурс]. – Режим доступа: https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation. – Дата доступа: 28.05.2024.
16. RAG (Retrieval Augmented Generation) — простое и понятное объяснение // Habr [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/779526. – Дата доступа: 02.06.2024.
17. MTEB Leaderboard // a Hugging Face Space by mteb [Электронный ресурс]. – Режим доступа: https://huggingface.co/spaces/mteb/leaderboard. – Дата доступа: 04.06.2024.
18. Sentence Transformers documentation // Sentence Transformers documentation [Электронный ресурс]. – Режим доступа: https://sbert.net. – Дата доступа: 05.06.2024.

**СПИСОК СОКРАЩЕНИЙ**

API – Application programming interface.

RAG – Retrieval augmented generation.

PDF – Portable document format.

VS – Visual studio.

CVS – Concurrent versions system.

OS – Operating system.

IDEA – Integrated development environment advanced.

TV – Television.

HTTP – Hypertext transfer protocol.

ACRA – Application crash report android.

GPT – Generative pre-training transformer.

ИИ – Искусственный интеллект.

ООП – Объектно-ориентированное программирование.

ПП – Программный продукт.