**СОДЕРЖАНИЕ**

[Введение 7](#_Toc193660381)

[1 Обзор литературы 8](#_Toc193660382)

[1.1 Нейронные сети и предобученные модели 8](#_Toc193660383)

[1.2 ONNX и HDF5 9](#_Toc193660384)

[1.3 REST API 9](#_Toc193660385)

[1.4 Серверная часть. Node.js и Express.js 10](#_Toc193660386)

[1.5 Клиентская часть. React 11](#_Toc193660387)

[1.6 JWT для аутентификации 12](#_Toc193660388)

[1.7 WebSocket и двустороннее взаимодействие 12](#_Toc193660389)

[1.8 Управление состоянием с помощью Redux 14](#_Toc193660390)

[1.9 Оптимизация пользовательского интерфейса с Material-UI 14](#_Toc193660391)

[1.10 Обзор существующих платформ чат-ботов 15](#_Toc193660392)

[1.11 Выводы 16](#_Toc193660393)

[2 Системное проектирование 17](#_Toc193660394)

[2.1 Блок API 17](#_Toc193660395)

[2.2 Блок базы данных 17](#_Toc193660396)

[2.3 Блок обработки данных 18](#_Toc193660397)

[2.4 Блок взаимодействия с нейронными моделями 18](#_Toc193660398)

[2.5 Блок авторизации 18](#_Toc193660399)

[2.6 Клиентский интерфейс 18](#_Toc193660400)

[2.7 Блок состояния 19](#_Toc193660401)

[2.8 Блок локального хранилища 19](#_Toc193660402)

[3 Функциональное проектирование 20](#_Toc193660403)

[3.1 Модуль данных 20](#_Toc193660404)

[3.2 Модуль авторизации 22](#_Toc193660405)

[3.3 Модуль работы с моделями 24](#_Toc193660406)

[3.4 Модуль комментариев 26](#_Toc193660407)

[3.5 Модуль управления датасетами 28](#_Toc193660408)

[3.6 Модуль обработки данных для моделей 29](#_Toc193660409)

[3.7 Модуль взаимодействия с моделями 29](#_Toc193660410)

[3.8 Секция представления 30](#_Toc193660411)

[3.9 Секция маршрутизации 32](#_Toc193660412)

[3.10 Секция авторизации 34](#_Toc193660413)

[3.11 Секция выбора темы 35](#_Toc193660414)

[3.12 Секция моделей 35](#_Toc193660415)

[3.13 Секция взаимодействия с нейросетями 37](#_Toc193660416)

[3.14 Секция комментариев 37](#_Toc193660417)

[3.15 Секция датасетов 38](#_Toc193660418)

[7 Технико-экономическое обоснование разработки и реализации на рынке веб-платформы для работы с моделями нейронных сетей 39](#_Toc193660419)

[7.1 Характеристика разработанного по индивидуальному заказу программного средства 39](#_Toc193660420)

[7.2 Расчет затрат на разработку и цена программного средства, созданного](#_Toc193660421)

[по индивидуальному заказу 39](#_Toc193660421)

[7.3 Расчет результата от разработки и использования программного средства, созданного по индивидуальному заказу 43](#_Toc193660422)

[7.4 Расчет показателей экономической эффективности разработки и использования программного средства 44](#_Toc193660423)

[7.5 Вывод об экономической целесообразности реализации проектного решения 45](#_Toc193660424)

[Заключение 46](#_Toc193660425)

[Список использованных источников 47](#_Toc193660426)

[Приложение А 48](#_Toc193660427)

[Приложение Б 49](#_Toc193660428)

[Приложение В 50](#_Toc193660429)

Введение

В последние годы технологии искусственного интеллекта (ИИ) и машинного обучения (МО) стремительно развиваются, находя применение в самых разных областях – от медицины и промышленности до бизнеса и образования. Одним из ключевых инструментов в этой сфере являются нейронные сети, которые позволяют решать такие задачи, как обработка изображений, анализ текста, прогнозирование данных и многое другое. Однако несмотря на их огромный потенциал, обучение работе с нейросетями и их практическое использование остаются сложной задачей для новичков, студентов и даже опытных специалистов.

Сложность работы с нейронными сетями обусловлена несколькими факторами: необходимостью глубоких знаний в области программирования, пониманием архитектуры моделей, а также управлением большими объемами данных. Эти трудности требуют создания удобных и доступных инструментов, которые не только упростят работу с моделями, но и позволят пользователям сосредоточиться на изучении основ и решении прикладных задач.

В ответ на этот вызов возникает необходимость разработки платформы, которая объединит в себе следующие функции: управление моделями нейронных сетей, работа с датасетами, визуализация результатов и аналитика производительности. Такие платформы должны быть интуитивно понятными, функциональными и гибкими, чтобы удовлетворять потребности как начинающих пользователей, так и профессионалов.

Разрабатываемая платформа ставит своей целью создание универсальной среды, где пользователи смогут загружать, тестировать модели, а также обмениваться своими наработками. Поддержка популярных форматов, таких как ONNX и h5, обеспечит совместимость с различными инструментами разработки, а наличие функций анализа и визуализации производительности моделей сделает процесс работы с ними максимально удобным и наглядным.

Для достижения этой цели в рамках проекта необходимо решить следующие задачи:

– реализовать загрузку и хранение моделей в форматах ONNX и h5;

– разработать удобный и интуитивно понятный интерфейс;

– обеспечить масштабируемость платформы для будущего роста.

Таким образом, разработка данной платформы представляет собой шаг вперёд в создании доступных инструментов для работы с ИИ и МО. Она найдёт своё применение не только в образовательной среде, помогая студентам изучать технологии нейросетей, но и в профессиональной деятельности, где высоко ценится удобство и эффективность инструментов для анализа данных и тестирования моделей.

Данный дипломный проект выполнен мной лично, проверен на заимствования, процент оригинальности составляет 85% (отчет о проверке на

заимствования прилагается).

# 1 Обзор литературы

## 1.1 Нейронные сети и предобученные модели

История нейронных сетей начинается с первых моделей в 1950-х годах, таких как перцептрон, разработанный Фрэнком Розенблаттом [1]. Этот ранний алгоритм заложил основу для дальнейших исследований в области искусственных нейронов. В последующие десятилетия нейронные сети прошли через различные этапы развития, включая «зиму ИИ» в 1970-х и 1980-х годах, когда интерес к ним значительно снизился из-за ограничений в вычислительных мощностях и отсутствии значительных результатов.

Ситуация изменилась в 2000-х годах с развитием глубокого обучения, что связано с увеличением объемов данных и вычислительных мощностей. Появление графических процессоров (GPU) открыло новые горизонты для обучения глубоких нейронных сетей, что привело к значительным успехам в различных задачах, таких как распознавание изображений и обработка естественного языка.

В современном мире ИИ и МО становятся неотъемлемой частью многих областей науки и промышленности, включая медицину, финансовый сектор, автомобильную промышленность и образование. Нейронные сети являются ключевыми инструментами в этой сфере, позволяя моделировать сложные зависимости в данных и решать широкий спектр задач, таких как:

1 Распознавание образов.

2 Обработка естественного языка.

3 Предсказательное моделирование.

4 Анализ больших данных.

Нейронные сети представляют собой вычислительные модели, вдохновленные биологическими нейронами. Они состоят из множества взаимосвязанных узлов (нейронов), которые передают сигналы друг другу, применяя весовые коэффициенты и функции активации. Современные архитектуры нейронных сетей, такие как сверточные нейронные сети, рекуррентные нейронные сети и трансформеры, достигают высоких результатов в решении различных задач ИИ.

Одним из значительных достижений в этой области стало развитие предобученных моделей. Эти модели предварительно обучены на больших объемах данных и могут быть использованы в различных приложениях без необходимости повторного обучения с нуля. Это особенно актуально в случаях с ограниченными данными или вычислительными ресурсами.

Использование предобученных моделей позволяет существенно ускорить разработку приложений, повысить их точность и уменьшить затраты на обучение. Кроме того, это предоставляет разработчикам возможность применять передовые алгоритмы машинного обучения без глубоких знаний в данной области. Популярные предобученные модели включают ResNet, BERT, GPT и другие.

Различные предобученные модели имеют свои особенности, которые делают их более подходящими для определённых задач. Например, BERT эффективен для задач обработки естественного языка, тогда как ResNet отлично справляется с задачами компьютерного зрения. Это позволяет разработчикам выбирать наиболее подходящие инструменты для решения конкретных задач, значительно упрощая процесс разработки.

Таким образом, нейронные сети и предобученные модели становятся основополагающими технологиями в области ИИ, открывая новые горизонты для инноваций и улучшая эффективность в различных отраслях.

## 1.2 ONNX и HDF5

Для эффективного использования предобученных моделей важно выбрать подходящий формат хранения. Различные платформы используют разные форматы, наиболее распространёнными из которых являются ONNX и H5.

ONNX (англ. Open Neural Network Exchange) [2] – это открытый формат, разработанный Microsoft и Facebook для обеспечения совместимости между различными фреймворками, такими как TensorFlow, PyTorch и Scikit-learn.

Преимущества ONNX:

1 Поддержка множества фреймворков и библиотек – облегчает переносимость моделей.

2 Оптимизация для различных аппаратных платформ (CPU, GPU, FPGA) повышает производительность.

3 Инструменты для профилирования и оптимизации – позволяют улучшить скорость выполнения моделей.

4 Интеграция с облачными сервисами – упрощает развертывание моделей.

HDF5 (англ. Hierarchical Data Format version 5) [3] – это формат хранения данных, который часто применяется в машинном обучении, особенно при использовании библиотеки Keras. Часто этот формат сокращают и обозначают как H5.

Преимущества HDF5:

1 Гибкая иерархическая структура – позволяет хранить сложные данные в удобном формате.

2 Высокая скорость записи и чтения – оптимизирован для работы с большими объемами данных.

3 Возможность хранения метаданных и параметров модели – облегчает восстановление и повторное использование моделей.

## 1.3 REST API

Для взаимодействия между клиентской и серверной частью веб-платформ широко используется REST API. Этот архитектурный стиль обеспечивает стандартизированный способ обмена данными с использованием HTTP-запросов и является основой для построения масштабируемых веб-сервисов.

Основные принципы REST API:

1 Отделение клиента от сервера – клиент и сервер развиваются независимо друг от друга.

2 Отсутствие сохранения текущего – каждый запрос содержит всю необходимую информацию, и сервер не хранит состояние клиента между запросами.

3 Кеширование – данные могут кешироваться для уменьшения нагрузки на сервер.

4 Единообразие интерфейса – использование стандартных HTTP-методов и URL-структуры.

5 Поддержка слоев – архитектура может включать промежуточные уровни, такие как балансировщики нагрузки и прокси-серверы.

Основные методы REST API:

1 GET – получение данных.

2 POST – создание нового ресурса.

3 PUT – обновление существующего ресурса.

4 DELETE – удаление ресурса.

На рисунке 1.1 показана архитектура REST API, в которой клиент (например, мобильное или веб-приложение) отправляет запросы (GET, POST, DELETE) к серверу, используя данные в формате JSON. Сервер обрабатывает эти запросы и взаимодействует с базой данных или другими ресурсами для получения необходимых данных.

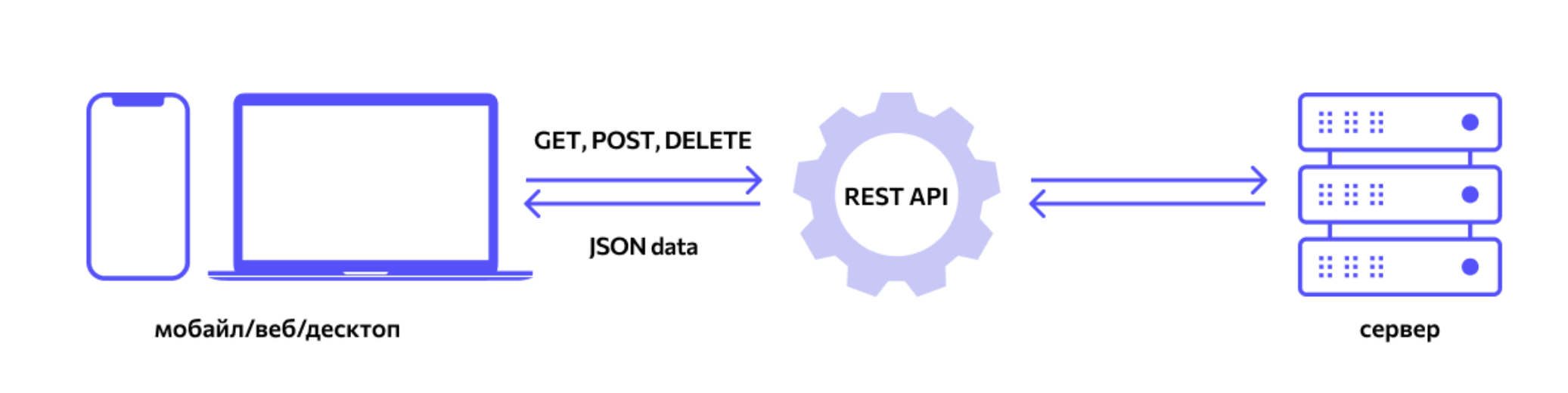


Рисунок 1.1 – Архитектура REST API [4]

REST API является инструментом для создания современных веб-приложений, обеспечивая стандартизированный и эффективный способ взаимодействия между клиентом и сервером. Это способствует масштабируемости и гибкости приложений, что делает их более адаптивными к изменениям и требованиям пользователей.

## 1.4 Серверная часть. Node.js и Express.js

Node.js (Node) [5] – это платформа с открытым исходным кодом для работы с языком JavaScript, построенная на движке Chrome V8. Она позволяет писать серверный код для веб-приложений и динамических веб-страниц, а также программ командной строки. В основе платформы – событийно-управляемая модель с неблокирующими операциями ввода-вывода, что делает ее эффективной и легкой.

Преимущества Node.js:

1 Высокая производительность – благодаря неблокирующему вводу-выводу (I/O) и асинхронной обработке запросов.

2 Кроссплатформенность – поддержка работы на различных операционных системах.

3 Экосистема NPM – огромное количество готовых модулей и библиотек.

Express.js [6] – это минималистичный и гибкий фреймворк для разработки веб-приложений на языке JavaScript с использованием платформы Node.js. Он предоставляет набор инструментов для создания маршрутов, обработки запросов и ответов, управления сессиями, работу с базами данных и многое другое. Express.js обеспечивает высокую производительность и позволяет быстро создавать веб-приложения.

Основные возможности Express.js:

1 Простая настройка маршрутов (GET, POST, PUT, DELETE).

2 Гибкость и расширяемость благодаря подключаемым модулям.

Благодаря сочетанию Node.js и Express.js можно быстро разрабатывать масштабируемые и высокопроизводительные веб-приложения.

## 1.5 Клиентская часть. React

React [7] – это мощная библиотека JavaScript, разработанная для создания пользовательских интерфейсов веб-приложений. Она позволяет разработчикам разбивать приложения на небольшие, повторно используемые компоненты, что значительно упрощает процесс разработки и обновления кода.

React особенно эффективен для создания динамических сайтов, где содержимое обновляется без необходимости перезагрузки страницы. Это делает его идеальным выбором для таких приложений, как социальные сети, онлайн-магазины, панели управления и чаты, где пользовательский опыт является ключевым фактором.

Основные преимущества React включают:

1 Модулярность – разделение приложения на независимые компоненты облегчает их разработку, тестирование и повторное использование.

2 Производительность – React использует виртуальный DOM, что позволяет эффективно обновлять интерфейс и обеспечивает плавную работу приложений.

3 Удобство для разработчиков – React делает код более читаемым и поддерживаемым, что способствует быстрой разработке и исправлению ошибок.

В заключение, React – это один из самых востребованных инструментов для создания современных веб-приложений, который помогает разработчикам создавать удобные и быстрые сайты, а также легко интегрироваться с другими библиотеками и технологиями, что делает процесс разработки более эффективным.

## 1.6 JWT для аутентификации

JWT (JSON Web Token) [8] – это стандарт, используемый для безопасной передачи информации между сторонами в виде JSON-объекта. Он часто применяется для аутентификации и авторизации в веб-приложениях благодаря своей компактности и простоте использования.

Преимущества JWT:

**1 Безопасность** – может быть подписан с использованием алгоритмов, таких как HMAC или RSA, что гарантирует целостность данных. Это позволяет убедиться, что токен не был изменён.

**2 Независимость –** не требует хранения сессий на сервере, что делает его подходящим для распределённых систем и микросервисов. Токен содержит всю необходимую информацию для аутентификации.

**3 Удобство** – JWT имеет компактный формат, что делает его удобным для передачи через HTTP-заголовки или URL-параметры.

**4 Гибкость** – может содержать произвольные данные (например, информацию о пользователе или его ролях), что позволяет легко расширять функциональность.

Принцип работы JWT:

1 Пользователь отправляет свои учетные данные (например, логин и пароль) на сервер.

**2**После успешной аутентификации сервер создаёт JWT, который включает в себя полезную нагрузку (payload) с информацией о пользователе и подписывается секретным ключом.

**3** Сервер возвращает токен клиенту, который может хранить его, например, в локальном хранилище браузера.

4 При последующих запросах клиент отправляет токен в заголовке Authorization. Сервер проверяет токен, и если он действителен, предоставляет доступ к защищённым ресурсам.

JWT – это мощный инструмент для аутентификации в современных веб-приложениях. Его безопасность, независимость от хранения сессий и гибкость делают его идеальным выбором для разработчиков, стремящихся создать масштабируемые и безопасные системы. Использование JWT упрощает процесс аутентификации и позволяет легко управлять доступом к ресурсам, что является важным аспектом в разработке современных приложений.

## 1.7 WebSocket и двустороннее взаимодействие

WebSocket – это протокол, который обеспечивает двустороннее взаимодействие между клиентом и сервером через одно постоянное соединение. В отличие от традиционного HTTP-запроса, который основан на модели «запрос-ответ», WebSocket позволяет обмениваться данными в реальном времени, что делает его идеальным для приложений, требующих мгновенной передачи информации.

Преимущества WebSocket:

1 Двусторонняя связь – обеспечивает постоянное соединение, позволяя как клиенту, так и серверу отправлять сообщения в любое время. Это значительно уменьшает задержку и увеличивает скорость обмена данными.

2 Снижение нагрузки на сервер – постоянное соединение позволяет избежать необходимости постоянно открывать и закрывать соединения, что снижает нагрузку на сервер и экономит ресурсы.

3 Поддержка реального времени – WebSocket идеально подходит для приложений, требующих мгновенной реакции, таких как чаты, игры, финансовые приложения и системы мониторинга.

4 Меньший объем данных: WebSocket использует более легкий протокол по сравнению с HTTP, что снижает объем передаваемых данных и увеличивает скорость обмена.

Принцип работы WebSocket:

1Клиент отправляет HTTP-запрос на сервер с просьбой установить WebSocket-соединение. Если сервер поддерживает WebSocket, он отвечает с кодом 101 (Switching Protocols), и соединение устанавливается.

2 После установки соединения клиент и сервер могут отправлять сообщения друг другу в любое время через это соединение.

3 Когда обмен данными завершён, любая из сторон может закрыть соединение, отправив специальный запрос.

Процесс открытия WebSocket-соединения приведен на рисунке 1.2.

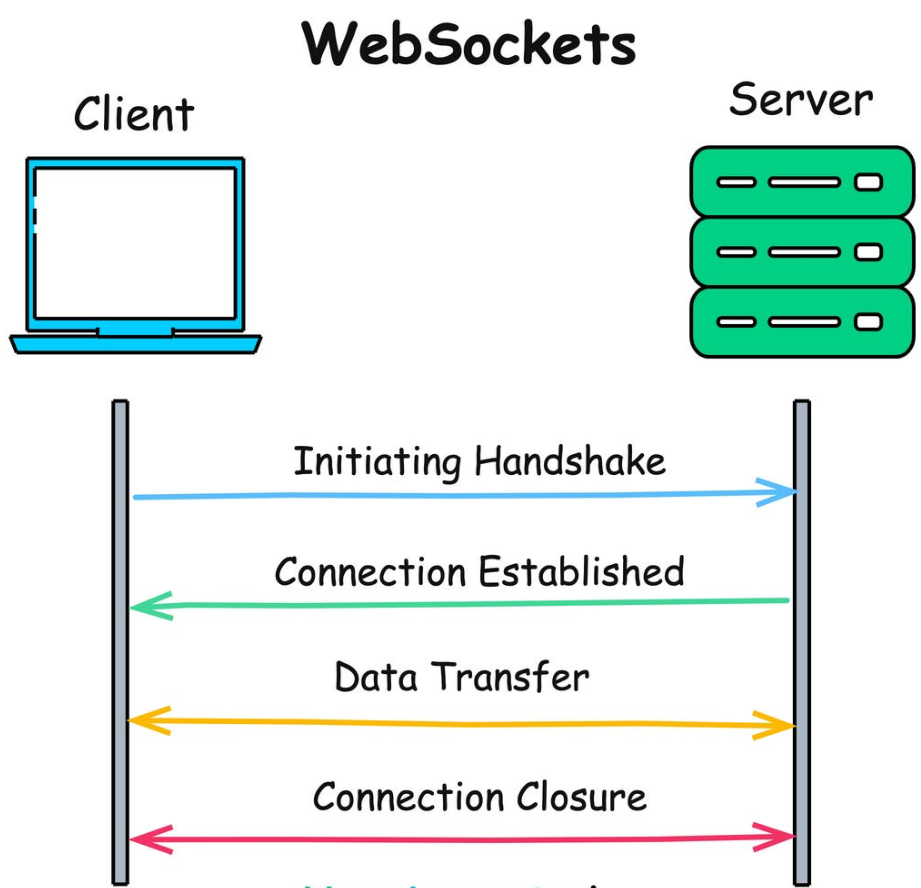
****

Рисунок 1.2 – Процесс открытия WebSocket-соединения [9]

WebSocket служит важным инструментом для создания двустороннего взаимодействия в веб-приложениях. Благодаря своей способности поддерживать постоянное соединение и обеспечивать мгновенный обмен сообщениями, WebSocket становится неотъемлемой частью современных приложений, где важна скорость и эффективность передачи данных. Использование WebSocket позволяет разработчикам создавать интерактивные и отзывчивые интерфейсы, которые значительно улучшают пользовательский опыт.

## 1.8 Управление состоянием с помощью Redux

Redux [10] – это библиотека для управления состоянием приложений на JavaScript, которая часто используется вместе с библиотекой React. Основная идея Redux заключается в хранении всего состояния приложения в одном централизованном хранилище, что упрощает управление данными и их обработку, особенно в крупных и сложных приложениях.

Одним из ключевых принципов Redux является единое состояние: всё состояние приложения хранится в одном объекте, что облегчает отслеживание и управление. Изменения состояния происходят только через редюсеры – чистые функции, которые принимают текущее состояние и действие (action), а затем возвращают новое состояние. Это обеспечивает иммутабельность состояния, так как оно не изменяется напрямую, а создаётся новое, что делает приложение более предсказуемым и облегчает отладку.

Когда в приложении происходит определённое событие, пользователь инициирует действие, которое описывает это событие. Это действие отправляется в обработчик состояния, который обновляет состояние на основе текущего состояния и полученного действия. Обновлённое состояние передаётся компонентам интерфейса, которые автоматически обновляются в ответ на изменения.

Использование Redux имеет множество преимуществ. Оно делает поведение приложения более предсказуемым благодаря централизованному хранению состояния и строгим правилам изменения. Кроме того, наличие инструментов разработчика для Redux помогает в отладке и визуализации состояния приложения. Благодаря своей предсказуемости и структурированному подходу, Redux стал важным инструментом для создания масштабируемых и поддерживаемых веб-приложений.

## 1.9 Оптимизация пользовательского интерфейса с Material-UI

Material-UI (MUI) [11] – это популярная библиотека компонентов для React, основанная на принципах Material Design, разработанных Google. Она предоставляет разработчикам набор инструментов для создания адаптивных и эстетически привлекательных пользовательских интерфейсов, что значительно помогает в процессе разработки.

Преимущества использования Material-UI:

1 MUI предлагает широкий ассортимент готовых компонентов, таких как кнопки, формы, карточки и навигационные элементы. Эти компоненты следуют рекомендациям Material Design, что обеспечивает единообразие и гармоничность в дизайне.

2 Библиотека поддерживает адаптивный дизайн, позволяя интерфейсам автоматически подстраиваться под различные размеры экранов и устройства. Это особенно важно в условиях увеличения числа мобильных пользователей.

3 MUI предоставляет мощные возможности для настройки тем, позволяя разработчикам настраивать цвета, шрифты и другие стилистические элементы приложения. Это помогает поддерживать единый стиль и брендирование.

4 Компоненты MUI разработаны с учётом принципов доступности, что позволяет создавать интерфейсы, удобные для всех пользователей, включая людей с ограниченными возможностями.

5 MUI легко интегрируется с другими библиотеками и фреймворками, что делает его универсальным инструментом для разработки. Он поддерживает такие инструменты, как Redux и React Router, обеспечивая более гибкое управление состоянием и маршрутизацией.

6 MUI имеет обширную документацию, которая включает примеры использования и рекомендации по лучшим практикам. Активное сообщество разработчиков способствует быстрому получению помощи и обмену опытом.

Использование Material-UI в разработке веб-приложений позволяет создать красивые и функциональные интерфейсы, что делает его хорошим выбором для современных проектов.

## 1.10 Обзор существующих платформ чат-ботов

С развитием технологий искусственного интеллекта и обработки естественного языка на рынке появилось множество платформ чат-ботов. Рассмотрим две популярных платформы: Poe [12] и Arena от LMSYS [13].

**1.10.1** Poe. Poe – это платформа, разработанная компанией Quora [14], которая предоставляет пользователям доступ к различным предобученным моделям, включая модели для обработки естественного языка. Она предлагает интуитивно понятный интерфейс, позволяющий пользователям легко задавать вопросы и получать ответы от чат-ботов.

Преимущества платформы:

– пользователи могут выбирать из множества предобученных моделей, что позволяет находить оптимальное решение для конкретной задачи;

– платформа предоставляет удобный и простой в использовании интерфейс, что облегчает взаимодействие с чат-ботами.

Недостатки платформы:

– платные подписки могут быть высокими для некоторых пользователей, особенно для небольших компаний или индивидуальных разработчиков.

**1.10.2** Arena от LMSYS. Платформа Arena от LMSYS – это пространство для взаимодействия с различными языковыми моделями, разработанное для упрощения доступа к передовым технологиям обработки естественного языка. Arena предоставляет пользователям возможность взаимодействовать с различными предобученными языковыми моделями, что позволяет экспериментировать с их функциональностью и возможностями.

Преимущества платформы:

– Arena позволяет пользователям обучать и тестировать модели на своих данных, что может быть полезно для адаптации моделей под специфические нужды;

– платформа может иметь активное сообщество, что способствует обмену опытом и знаниями между пользователями.

Недостатки платформы:

– Arena требует стабильного интернет-соединения для работы;

– эффективность моделей может варьироваться в зависимости от языка, что может быть ограничением для пользователей, работающих с менее распространенными языками.

## 1.11 Выводы

На основе обзора литературы принято решение использовать, следующие технологии:

– для разработки серверной части Express.js и Node.js;

– для разработки клиентской части React;

– для управления состоянием приложения Redux;

– для стилизации интерфейса Material-UI;

– JWT для аутентификации.

Такой стек технологий позволяет создать понятную, надежную и эффективную веб-платформу для работы с нейронными моделями, способную автоматизировать процессы обработки данных и облегчить взаимодействие пользователей с моделями.

# 2 Системное проектирование

Программное средство представляет собой веб-платформу для работы с предобученными нейросетевыми моделями. Для корректного функционирования проекта требуется чётко структурированная система, включающая в себя как серверные, так и клиентские компоненты.

Структура проекта основана на клиент-серверной архитектуре, где фронтенд отвечает за взаимодействие с пользователем, а бэкенд – за обработку запросов, управление данными и выполнение вычислений.

Проект включает следующие основные блоки:

– блок API;

– блок базы данных;

– блок обработки данных;

– блок взаимодействия с нейронными моделями;

– блок авторизации;

– клиентский интерфейс;

– блок состояния;

– блок локального хранилища.

Взаимодействие между выделенными блоками представлено на чертеже ГУИР.400201.096 С1.

## 2.1 Блок API

Блок API отвечает за обработку запросов от клиентской части и взаимодействие с серверными компонентами. Он предоставляет унифицированный интерфейс для доступа к данным и выполнению операций, связанных с пользователями, моделями, датасетами и их обработкой. API реализует маршрутизацию запросов, валидацию входных данных, управление сессиями и передачу данных в формате JSON или других стандартизированных форматах.

Блок API играет ключевую роль в обеспечении безопасности приложения, включая контроль доступа, аутентификацию и авторизацию пользователей. Он также обрабатывает ошибки и исключения, предоставляя клиенту соответствующие ответы. Для масштабируемости и отказоустойчивости API может поддерживать кэширование, логирование и балансировку нагрузки.

## 2.2 Блок базы данных

Блок базы данных отвечает за хранение, управление и организацию всех данных проекта, включая информацию о пользователях, моделях, датасетах и их параметрах. Он обеспечивает структурированное хранение информации, позволяя эффективно выполнять поиск, фильтрацию и обработку данных.

Для оптимизации работы система использует механизмы индексации и транзакций, что позволяет ускорять запросы и обеспечивать согласованность данных. Взаимодействие с базой данных осуществляется через ORM или прямые SQL-запросы, в зависимости от потребностей проекта.

## 2.3 Блок обработки данных

Блок обработки данных отвечает за предобработку входных данных перед их передачей в нейронные модели. Он включает нормализацию, очистку, преобразование и аугментацию данных, а также разделение на обучающую, валидационную и тестовую выборки. Блок обработки данных обеспечивает, чтобы данные были в нужном формате для нейронных сетей, а также обрабатывает возможные ошибки во время выполнения. Он может использовать механизмы кэширования для ускорения повторных запросов.

## 2.4 Блок взаимодействия с нейронными моделями

Этот блок отвечает за интеграцию API с предобученными нейросетевыми моделями. Он обрабатывает запросы на выполнение инференса (применение модели к входным данным) и отвечает за передачу данных между клиентом и сервером. Блок взаимодействия с нейронными моделями обеспечивает совместимость с различными фреймворками и форматами (например, ONNX, h5), позволяя использовать модели, обученные в разных средах.

Основная функция этого блока заключается в управлении процессом инференса. Блок принимает обработанные данные от блока обработки данных и возвращает результаты обратно в API.

## 2.5 Блок авторизации

Блок авторизации отвечает за аутентификацию пользователей и контроль их доступа к системе. Он реализует механизмы регистрации, входа, выхода и восстановления пароля, а также управление сессиями. В качестве основного метода аутентификации используется JWT, что позволяет безопасно передавать и проверять учетные данные без необходимости хранения сессионных данных на сервере.

Блок поддерживает разграничение прав доступа на основе ролей, что позволяет разделять пользователей по уровням доступа. Это обеспечивает контроль над функциональностью системы, предоставляя разные возможности администраторам, исследователям и другим категориям пользователей.

## 2.6 Клиентский интерфейс

Клиентский интерфейс отвечает за взаимодействие пользователей с платформой и управление различными ее компонентами. Он предоставляет возможность пользователям изменять личные данные, такие как имя, e-mail и пароль, а также управлять предпочтениями и настройками интерфейса.

Блок также обеспечивает доступ к просмотру моделей, их графиков и характеристик, что позволяет пользователям легко ориентироваться в доступных ресурсах. Интерфейс позволяет просматривать комментарии к моделям и заметки, оставленные пользователями, способствуя активному взаимодействию и обмену мнениями. Помимо этого, блок предоставляет возможность просмотра и анализа датасетов, по которым были обучены модели.

Взаимодействие с блоком API и блоком состояния обеспечивает синхронизацию данных между сессиями и устройствами, гарантируя единый пользовательский опыт. Клиентский интерфейс играет ключевую роль в создании интуитивно понятного и удобного взаимодействия с платформой, обеспечивая доступ к различным функциям и данным.

## 2.7 Блок состояния

Блок состояния отвечает за управление данными, используемыми на клиентской стороне приложения. Он хранит текущее состояние пользовательского интерфейса, активные сессии, выбранные модели и датасеты. Взаимодействие с этим блоком позволяет пользователям работать с платформой без необходимости повторной загрузки данных при каждом обновлении страницы.

Этот блок может быть реализован с использованием таких технологий, как Redux или контекстное хранилище в React. Он играет важную роль в обеспечении отзывчивости и удобства работы с интерфейсом, снижая нагрузку на сервер за счет локального кэширования данных.

## 2.8 Блок локального хранилища

Блок локального хранилища отвечает за сохранение токенов и других данных на стороне клиента. Он обеспечивает доступ к информации между сессиями, позволяя пользователям оставаться авторизованными и сохранять свои настройки. Также данный блок может использоваться для кэширования временных данных, что ускоряет работу приложения и уменьшает нагрузку на сервер.

# 3 Функциональное проектирование

Функциональное проектирование является одним из ключевых разделов дипломного проектирования. Этот раздел должен содержать блоки, демонстрирующие взаимосвязь между основными компонентами программы и описывающие, как они взаимодействуют друг с другом для достижения поставленных задач. Диаграмма классов представлена на чертеже ГУИР.400201.096 РР.1.

В рамках проекта можно выделить следующие функциональные блоки:

– модуль данных;

– модуль авторизации;

– модуль управления пользователями;

– модуль работы с моделями;

– модуль комментариев;

– модуль управления датасетами;

– модуль обработки данных для моделей;

– модуль взаимодействия с моделями;

– секция представления;

– секция маршрутизации;

– секция авторизации;

– секция выбора темы;

– секция моделей;

– секция взаимодействия с нейросетями;

– секция комментариев;

– секция датасетов;

## 3.1 Модуль данных

Модуль данных в проекте представляет собой реляционную структуру, хранящую информацию о пользователях, нейросетевых моделях, комментариях, заметках и датасетах. Данные организованы в таблицы, каждая из которых отвечает за хранение конкретных сущностей и их связей. Используется база данных PostgreSQL [14].

Доступ к данным осуществляется через SQL-запросы, обеспечивая удобное управление записями. Для оптимизации работы используются индексы и связи между таблицами, что ускоряет выборки и обеспечивает целостность данных.

**3.1.1** Таблица пользователей. Таблица пользователей служит основой системы управления учетными записями, храня информацию обо всех зарегистрированных участниках платформы.

**Структура таблицы:**

1 id – уникальный идентификатор пользователя.

2 email – адрес электронной почты пользователя.

3 username – имя пользователя.

4 password – хешированный пароль.

5 created\_at – дата и время регистрации.

**3.1.2** Таблица моделей. Хранит информацию о загруженных нейронных моделях на платформу.

Структура таблицы:

1 id – уникальный идентификатор модели.

2 name – название модели.

3 description – описание модели.

4 model\_url – ссылка на файл модели.

5 dataset\_url – ссылка на используемый датасет.

6 parameters – JSON-объект с параметрами модели.

7 created\_at – дата и время загрузки модели.

8 updated\_at – дата последнего обновления модели.

**3.1.3** Таблица комментариев. Хранит комментарии пользователей к моделям.

Структура таблицы:

1 id – уникальный идентификатор комментария.

2 model\_id – ID модели, к которой относится комментарий.

3 user\_id – ID пользователя, оставившего комментарий.

4 content – текст комментария.

5 created\_at – дата и время создания комментария.

6 updated\_at – дата последнего обновления комментария.

**3.1.4** Таблица заметок. Хранит личные заметки пользователей о работе с моделями.

Структура таблицы:

1 id – уникальный идентификатор заметки.

2 user\_id – ID пользователя, создавшего заметку.

3 title – заголовок заметки.

4 content – содержание заметки.

5 created\_at – дата и время создания заметки.

6 updated\_at – дата последнего обновления заметки.

**3.1.5** Таблица датасетов. Таблица датасетов предназначена для хранения информации о загруженных наборах данных, которые используются в процессе обучения, тестирования и валидации моделей.

Структура таблицы:

1 id – уникальный идентификатор датасета.

2 name – название датасета.

3 description – описание датасета.

4 dataset\_url – ссылка на файл датасета.

5 user\_id – ID пользователя, загрузившего датасет.

6 created\_at – дата и время загрузки датасета.

7 updated\_at – дата и время обновления датасета.

## 3.2 Модуль авторизации

Этот блок управляет процессами регистрации, входа и выхода пользователей, обеспечивая контроль доступа и защиту маршрутов с помощью аутентификации. Он позволяет пользователям безопасно взаимодействовать с системой, защищая их данные и обеспечивая возможность работы только авторизованных пользователей.

Функции блока:

– позволяет новым пользователям создавать учетные записи в системе;

– проверяет учетные данные пользователя для предоставления доступа к системе;

– поддерживает работу с токенами доступа и обновления токенов для обеспечения безопасности;

– предоставляет возможность получения информации о пользователях, обновления их данных и удаления учетных записей.

Данный блок включает в себя следующие эндпоинты:

POST /register – регистрация нового пользователя. Принимает данные пользователя и создает новую учетную запись. Если учетная запись успешно создана, возвращает данные пользователя и токены доступа.

Принимает следующие данные для создания нового пользователя:

– username – имя пользователя;

– password – пароль;

– email – электронная почта.

Возвращаемые данные при успешной регистрации:

– user – объект с данными пользователя:

– id – идентификатор пользователя;

– username – имя пользователя;

– email – электронная почта;

– role – роль пользователя;

– created\_at – дата создания учетной записи.

– accessToken – токен доступа для дальнейших запросов;

– refreshToken – токен обновления для получения нового токена доступа.

POST /login – эндпоинт для входа пользователя в систему. Принимает данные пользователя, выполняет проверку их правильности.

Принимает следующие данные пользователя:

– username – имя пользователя;

– password – пароль.

Возвращаемые данные при успешной аутентификации:

– accessToken – токен доступа для дальнейших запросов;

– refreshToken – токен обновления для получения нового токена доступа.

При ошибке возвращает поле error с описанием того, что пошло не так.

POST /logout – выход пользователя. Удаляет токен обновления из базы данных, тем самым разрывая сессию пользователя.

Принимает следующие данные пользователя:

– accessToken – токен доступа для аутентификации пользователя.

Возвращает следующее поле:

– message – строка, подтверждающая успешный выход пользователя, «User logged out successfully».

При ошибке возвращает поле error с описанием того, что пошло не так.

POST /refresh – обновление токена доступа. Принимает токен обновления и, если он действителен, возвращает новый токен доступа.

Принимает следующие данные пользователя:

– refreshToken – токен обновления.

Возвращает следующее поле:

– accessToken – токен доступа.

GET /users – получение списка всех пользователей. Доступен только для пользователей с ролью администратора.

Принимает accessToken декодирует его и вытягивает из него роль пользователя.

Возвращает массив объектов с информацией о всех пользователях в системе, объект пользователя:

– id – идентификатор пользователя;

– username – имя пользователя;

– email – электронная почта;

– role – роль пользователя;

– created\_at – дата создания учетной записи.

GET /user/:id – получение информации о конкретном пользователе по его идентификатору. Доступен только для авторизованных пользователей. Возвращает объект с данными о пользователе.

Принимает данные:

– id – идентификатор пользователя.

Возвращает следующее поля:

– id – идентификатор пользователя;

– username – имя пользователя;

– email – электронная почта;

– role – роль пользователя;

– created\_at – дата создания учетной записи.

PUT /users/:id – обновление информации о пользователе. Позволяет изменять данные пользователя по его идентификатору.

Принимает данные:

– id – идентификатор пользователя.

– username – имя пользователя;

– email – электронная почта;

– password – пароль.

Возвращает сообщение об успешном обновлении:

– message – строка, подтверждающая успешное обновление, «User updated successfully».

При ошибке возвращает поле error с описанием того, что пошло не так.

DELETE /users/:id – удаление пользователя. Удаляет запись о пользователе и все связанные с ним токены обновления.

Принимает данные:

– id – идентификатор пользователя.

Возвращает сообщение об успешном удалении:

– message – строка, подтверждающая успешное удаление, «User deleted successfully».

При ошибке возвращает поле error с описанием того, что пошло не так.

## 3.3 Модуль работы с моделями

Этот модуль управляет предобученными моделями, предоставляя возможность создания, обновления и удаления моделей, а также получения информации о них.

**Функции блока:**

– позволяет пользователям создавать новые модели, загружая соответствующие параметры и метаданные;

– обеспечивает возможность получения списка всех доступных моделей;

– позволяет получать детальную информацию о конкретной модели;

– поддерживает обновление существующих моделей с новыми данными или параметрами;

– предоставляет возможность удаления моделей из системы.

**Данный блок включает в себя следующие эндпоинты:**

POST /models/create – создание новой модели. Принимает данные модели и сохраняет их в базе данных.

Принимает следующие данные:

– name – имя модели;

– description – описание модели;

– modelUrl – URL модели;

– datasetUrl – URL датасета;

– parameters – параметры модели.

При успешном создании модели, метод возвращает следующее поля:

– message – строка, подтверждающая успешное создание модели, «Model created successfully».

– model – объект с данными созданной модели, содержащий:

– id – идентификатор модели;

– name – имя модели;

– description – описание модели;

– modelUrl – URL модели;

– datasetUrl – URL датасета;

– parameters – параметры модели;

– created\_at – дата и время создания;

– updated\_at – дата и время последнего обновления.

При ошибке возвращает поле error с описанием того, что пошло не так.

GET /models/list – получение списка всех моделей. Возвращает массив объектов моделей, отсортированных по дате создания.

Метод не принимает никаких данных.

Возвращаемые данные:

– id – идентификатор модели;

– name – имя модели;

– description – описание модели;

– modelUrl – URL модели;

– datasetUrl – URL датасета;

– parameters – параметры модели;

– created\_at – дата и время создания;

– updated\_at – дата и время последнего обновления.

GET /models/:id – получение информации о конкретной модели по её идентификатору.

Принимает следующие данные:

– id – идентификатор модели.

Возвращает детали модели по идентификатору:

– id – идентификатор модели;

– name – имя модели;

– description – описание модели;

– modelUrl – URL модели;

– datasetUrl – URL датасета;

– parameters – параметры модели;

– created\_at – дата и время создания;

– updated\_at – дата и время последнего обновления.

PUT /models/update/:id – обновление информации о модели. Позволяет изменять параметры модели по её идентификатору.

Принимает следующие данные:

– id – идентификатор модели;

– name – имя модели;

– description – описание модели;

– modelUrl – URL модели;

– datasetUrl – URL датасета;

– parameters – параметры модели.

Возвращаемые данные:

– message – строка, подтверждающая успешное обновление модели, «Model updated successfully».

– model – объект с данными созданной модели, содержащий:

– id – идентификатор модели;

– name – имя модели;

– description – описание модели;

– modelUrl – URL модели;

– datasetUrl – URL датасета;

– parameters – параметры модели;

– created\_at – дата и время создания;

– updated\_at – дата и время последнего обновления.

DELETE /models/delete/:id – удаление модели. Удаляет запись о модели из базы данных.

Принимает следующие данные:

– id – идентификатор модели для удаления.

Возвращаемые данные:

– message – строка, подтверждающая успешное удаление модели, «Model deleted successfully».

## 3.4 Модуль комментариев

Этот блок управляет комментариями, оставляемыми пользователями к моделям. Он позволяет пользователям обмениваться мнениями и предложениями по моделям, а также поддерживает взаимодействие внутри платформы.

Функции блока:

– позволяет пользователям добавлять комментарии к моделям;

– обеспечивает возможность получения списка комментариев для конкретной модели;

– поддерживает редактирование и удаление комментариев;

– позволяет пользователям взаимодействовать друг с другом через систему комментариев.

Данный блок включает в себя следующие эндпоинты:

POST /comments/create – добавление нового комментария. Принимает следующие данные для создания комментария:

– model\_id – идентификатор модели;

– user\_id – идентификатор пользователя, который хочет оставить комментарий;

– content – текст комментария.

После успешного запроса, комментарий сохраняется в базу данных, а данный метод возвращает ответ в таком формате:

– message – строка, подтверждающая успешное создание комментария, «Comment added successfully».

– comment – объект нового комментария:

– id – идентификатор комментария;

– model\_id – идентификатор связанной модели;

– user\_id – идентификатор пользователя, который оставил комментарий;

– content – текст комментария;

– created\_at – дата и время создания;

– updated\_at – дата и время последнего обновления.

При ошибке возвращает поле error с описанием того, что пошло не так.

GET /comments/:id – получение комментариев для конкретной модели. Возвращает список комментариев, отсортированных по дате создания.

Принимает следующие данные для получения комментариев:

– model\_id – идентификатор модели.

Возвращает массив объектов комментариев к данной модели, объект комментария:

– id – идентификатор комментария;

– model\_id – идентификатор связанной модели;

– user\_id – идентификатор пользователя, который оставил комментарий;

– content – текст комментария;

– created\_at – дата и время создания;

– updated\_at – дата и время последнего обновления.

PUT /comments/update/:id – обновление существующего комментария. Позволяет изменять содержание комментария по его идентификатору.

Принимает следующие данные:

– model\_id – идентификатор модели;

– content – новое описание комментария.

При успешном обновлении комментария, возвращается следующее поле:

– message – строка, подтверждающая успешное обновление комментария, «Comment updated successfully».

При ошибке возвращает поле error с описанием того, что пошло не так.

DELETE /comments/delete/:id – удаление комментария. Удаляет запись о комментарии из базы данных.

Принимает следующие данные:

– comment\_id – идентификатор комментария.

При успешном удалении комментария, возвращается следующее поле:

– message – строка, подтверждающая успешное удаление комментария, «Comment deleted successfully».

При ошибке возвращает поле error с описанием того, что пошло не так.

## 3.5 Модуль управления датасетами

Этот блок отвечает за управление наборами данных, используемыми для обучения и тестирования моделей. Он позволяет пользователям создавать, обновлять и удалять наборы данных, а также получать информацию о них.

Функции блока:

– позволяет пользователям загружать новые наборы данных;

– обеспечивает возможность получения списка всех доступных наборов данных;

– позволяет получать детальную информацию о конкретном наборе данных;

– поддерживает обновление существующих наборов данных;

– предоставляет возможность удаления наборов данных из системы.

Данный блок включает в себя следующие эндпоинты:

POST /datasets/create – создание нового набора данных. Принимает данные и сохраняет их в базе данных.

Принимает следующие данные для создания нового набора данных:

– name – название набора данных;

– description – описание набора данных;

– dataset\_url – ссылка на набор данных;

– model\_id – id модели, для который данный набор создан.

При успешном создании возвращает объект созданного набора данных. При неудаче, возвращает поле error с описанием того, что пошло не так.

GET /datasets/list – получение списка всех наборов данных.

Не принимает никаких данных. Возвращает массив объектов датасетов, отсортированных по дате создания.

GET /datasets/:id – получение информации о конкретном наборе данных по его идентификатору.

Принимает идентификатор набора данных dataset\_id и возвращает объект набора данных по полученному dataset\_id.

PUT /datasets/update/:id – обновление информации о наборе данных. Позволяет изменять параметры набора по его идентификатору.

Принимает следующие данные:

– id – идентификатор набора данных;

– dataset – новый объект набора данных, который содержит новые значения полей для обновления.

При успешном обновлении возвращает обновленный объект набора данных. При ошибке возвращает поле error с названием ошибки.

DELETE /datasets/delete/:id – удаление набора данных. Удаляет запись о наборе данных из базы данных.

Принимает следующие данные:

– id – идентификатор набора данных.

При успешном удалении возвращает:

– message – строка, подтверждающая успешное удаление набора данных, «Dataset deleted successfully».

При ошибке возвращает поле error с описанием того, что пошло не так.

## 3.6 Модуль обработки данных для моделей

Модуль обработки данных для моделей отвечает за подготовку и предварительную обработку входящих данных, которые будут использоваться в нейронных сетях. Он включает в себя функции для нормализации, преобразования и фильтрации данных различных форматов, таких как текстовые данные и изображения.

## 3.7 Модуль взаимодействия с моделями

Модуль взаимодействия с моделями обеспечивает пользователям возможность тестировать и оценивать эффективность предобученных моделей нейронных сетей в реальном времени. Он включает в себя функции для загрузки моделей, выполнения предсказаний и обработки входных данных. Модуль обеспечивает универсальный подход к обработке данных, позволяя адаптировать их под разные типы нейронных моделей.

**3.7.1** Загрузка моделей. Этот процесс начинается с загрузки предобученных моделей в формате ONNX или h5. Модуль предоставляет API для загрузки моделей, который принимает URL-адреса моделей и сохраняет их в памяти для последующего использования.

Этот модуль имеет эндпоинт POST /load\_models, который принимает поле models, который содержит массив объектов моделей, объект модели имеет следующую структуру:

– id – идентификатор модели;

– model\_url – путь для загрузки модели.

При успешной загрузке эндпоинт возвращает сообщение status, которое содержит «Models loaded».

**3.7.2** Выполнение предсказаний. Модуль предоставляет эндпоинт для выполнения предсказаний на основе входных данных. Пользователь может отправить данные для обработки, и модуль вернет результаты предсказаний. Это позволяет пользователям легко интегрировать модели в свои приложения и получать результаты в реальном времени.

Эндпоинт POST /predict принимает следующие поля:

– id – идентификатор модели;

– input\_data – входные данные.

При успешном предсказании метод возвращает поле result, которое содержит результат выполнения предсказания.

## 3.8 Секция представления

Секция представления отвечает за отображение пользовательского интерфейса и маршрутизацию внутри веб-приложения. Она является основной точкой входа в приложение, обеспечивая рендеринг компонентов, навигацию между страницами, работу с темами и глобальное управление состоянием через Redux.

Приложение построено на React, использует Material UI (MUI) для стилизации и react-router-dom для организации маршрутизации. Главным контейнером является компонент Main.tsx, который управляет основными настройками интерфейса.

**3.8.1** Точка входа. index.tsx.Этот файл отвечает за начальную инициализацию приложения и монтирование главного компонента в DOM.

Основные компоненты:

– ReactDOM.createRoot() – точка монтирования React-приложения.

– <BrowserRouter> – обеспечивает клиентскую маршрутизацию.

– <Main /> – главный компонент, который содержит все ключевые элементы.

Код index.tsx:

import ReactDOM from 'react-dom/client';

import { BrowserRouter } from 'react-router-dom';

import Main from './main.tsx';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<BrowserRouter>

<Main />

</BrowserRouter>

);

**3.8.2** Главный компонент. Main.tsx. Этот компонент управляет глобальными настройками приложения, обеспечивая подключение Redux-хранилища через <Provider store={store} />, что позволяет управлять состоянием на уровне всего приложения. Он также отвечает за обработку смены темы между светлой и темной, используя ThemeProvider из Material UI, что позволяет динамически изменять внешний вид интерфейса.

В зависимости от текущего маршрута компонент определяет, нужно ли отображать Header и Footer. Эти элементы скрываются на страницах авторизации (/login, /register), странице ошибки (/404) и в разделе чата (/chat), чтобы избежать дублирования элементов интерфейса, которые в этих случаях не нужны.

Внутри компонента Main подключается основной маршрутизатор приложения через <Routes />, который управляет переходами между страницами. Дополнительно он включает CustomAlert, который отвечает за отображение всплывающих уведомлений, обеспечивая пользователю обратную связь о важных событиях в системе.

Компонент <Routes /> будет рассмотрен в следующем разделе.

Кроме того, Main.tsx отвечает за инициализацию ключевых глобальных контекстов и провайдеров, обеспечивая их доступность во всех дочерних компонентах. Это включает в себя управление темой, глобальное состояние и уведомления, создавая единое место для конфигурации приложения и улучшая его модульность.

**3.8.3** Глобальное состояние. Redux-хранилище. В приложении используется Redux для управления глобальным состоянием, что позволяет централизованно хранить и изменять данные, необходимые различным компонентам. Redux-хранилище объединяет несколько обработчиков состояния, каждый из которых отвечает за свою область данных.

Основной файл store.ts настраивает хранилище Redux и подключает три обработчика: alertReducer для управления уведомлениями, modelsReducer для работы с моделями и userReducer для хранения информации о пользователе. Это позволяет структурировать данные и упрощает их использование в разных частях приложения.

Обработчик alertSlice.ts отвечает за систему уведомлений. В нем хранится состояние с параметрами isShowAlert (флаг показа уведомления), message (текст сообщения) и type (тип уведомления, например, success, error, info). В initialState уведомления отключены по умолчанию. Два действия, showAlert и hideAlert, позволяют отображать и скрывать уведомления в ответ на события в приложении.

Обработчик modelsSlice.ts управляет списком моделей и их состоянием. В нем определена асинхронная функция fetchModelsList, которая загружает список моделей с сервера. В состоянии modelsState хранятся массив моделей models, флаг loading, индикатор ошибок error и параметр updateComments для отслеживания обновления комментариев к моделям. Через extraReducers обрабатываются три состояния запроса: pending (запрос отправлен), fulfilled (данные успешно получены) и rejected (ошибка загрузки).

Все обработчики состояния объединены в store.ts, формируя единое хранилище для приложения. Тип RootState определяет всю структуру состояния, а AppDispatch позволяет корректно типизировать dispatch(). Это обеспечивает централизованное управление состоянием, улучшает масштабируемость приложения и делает работу с данными более удобной.

**3.8.4** Компонент CustomAlert.tsx. Компонент CustomAlert.tsx отвечает за отображение всплывающих уведомлений в приложении. Он использует Snackbar и Alert из Material UI для удобного и стильного отображения сообщений. Уведомления появляются в нижней части экрана и автоматически исчезают через три секунды.

Компонент получает данные из глобального состояния Redux с помощью useSelector(). Он извлекает информацию о том, нужно ли показывать уведомление, какой у него текст и тип (например, success, error, warning или info). Эта информация управляется в alertSlice.ts.

Закрытие уведомления происходит при нажатии на кнопку закрытия или автоматически после истечения заданного времени. Когда уведомление закрывается, вызывается handleClose(), который отправляет в Redux действие hideAlert(), чтобы скрыть уведомление и очистить его состояние.

Внутри компонента используется Snackbar, который служит контейнером для уведомлений, и Alert, который визуально оформляет их в зависимости от типа. Уведомления располагаются по центру внизу экрана, а их ширина адаптируется под размер экрана.

Компонент CustomAlert добавляется в Main.tsx, чтобы он был доступен на всех страницах приложения. Таким образом, любые события, требующие уведомления пользователя, могут отображать всплывающее сообщение через Redux, и оно автоматически исчезнет через заданное время.

## 3.9 Секция маршрутизации

Секция маршрутизации отвечает за организацию навигации внутри приложения, обеспечивая переходы между страницами и защиту маршрутов. Для этого используется библиотека react-router-dom, позволяющая управлять клиентской маршрутизацией без перезагрузки страницы.

В коде определен специальный компонент ProtectedRoute, который проверяет, авторизован ли пользователь. Если доступ к защищенному маршруту пытается получить неавторизованный пользователь, его автоматически перенаправляет на страницу /login. Это предотвращает доступ к личному кабинету, чату и другим защищенным страницам без входа в систему.

Основная функция Routes отвечает за настройку всех маршрутов приложения. Для их создания используется хук useRoutes, который принимает массив объектов с путями и соответствующими им компонентами. Внутри этого компонента также выполняется проверка авторизации пользователя с помощью useEffect, вызывающего функцию checkAuth. Она проверяет, сохранена ли сессия пользователя, и при необходимости инициирует повторную авторизацию через dispatch(login).

**3.9.1** Открытые маршруты. Эти страницы доступны всем пользователям, даже если они не вошли в систему:

**–  / – Главная страница, компонент <Home.tsx>.** Главная страница приложения, которая содержит общую информацию, описание сервиса и приветственный экран.

**–**/login**– Страница входа, компонент <AuthPage.tsx>.** Форма входа, где пользователи могут ввести учетные данные и авторизоваться в системе.

**–**/register**– Страница регистрации, компонент <AuthPage.tsx>.**Форма регистрации, позволяющая создать новый аккаунт.

**–**/404 **– Страница 404, компонент <NotFound.tsx>.** Отображается в случае, если пользователь пытается перейти на несуществующий маршрут.

**3.9.2** Защищенные маршруты. Для доступа к этим страницам пользователь должен быть авторизован. Если неавторизованный пользователь попытается открыть эти маршруты, он будет автоматически перенаправлен на /login. Защита маршрутов реализована с помощью компонента ProtectedRoute.

**–** /profile **– Профиль пользователя, компонент <Profile.tsx/>.** Личный кабинет пользователя, где он может просматривать и редактировать свои данные.

**–** /models **– Страница моделей, компонент <Models.tsx/>.** Раздел, в котором отображается список доступных нейронных моделей.

**–**/models/:id **– Отдельная страница для конкретной модели, компонент <Model.tsx>.** Страница с подробной информацией о конкретной модели. В :id передается идентификатор выбранной модели.

**–**/chat **– Раздел чатов, компонент <Chat.tsx>.** Интерфейс для пользователь, где они делают запросы к моделям.

**–**/datasets **– Управление датасетами, компонент <Datasets.tsx>.**Раздел, где пользователи могут просматривать и управлять наборами данных, используемыми для обучения моделей.

При открытии приложения пользователь попадает на /. Если он не вошел в систему, он может посетить /login и /register. После успешного входа он получает доступ к защищенным страницам, таким как /profile, /models и /chat.

Защита маршрутов осуществляется с помощью ProtectedRoute, который проверяет isAuth из глобального состояния. Если isAuth === false, пользователя перенаправляют на /login.

Компонент ProtectedRoute:

export const ProtectedRoute = ({ isAuth, children }: { isAuth: boolean, children: JSX.Element }) => {

if (!isAuth) {

return <Navigate to="/login"/>;

}

return children;

};

Система маршрутизации не только организует удобную навигацию по приложению, но и обеспечивает защиту критически важных разделов от неавторизованного доступа. Использование react-router-dom позволяет управлять маршрутизацией на стороне клиента без необходимости перезагрузки страницы, создавая плавный пользовательский опыт.

## 3.10 Секция авторизации

Секция авторизации отвечает за вход и регистрацию пользователей в системе. Она реализована через компонент AuthPage, который предоставляет интерфейс для аутентификации. Компонент принимает параметр type, который определяет, будет ли пользователь входить в систему или регистрироваться.

Внутри компонента используется состояние userFormData, которое хранит данные, введенные пользователем: e-mail, пароль и, если пользователь проходит регистрацию, имя. Для обновления состояния полей формы используется функция handleChange, которая отслеживает изменения в полях ввода и обновляет соответствующие значения.

Основная логика входа и регистрации реализована в функции authHandler. При отправке формы сначала отменяется стандартное поведение браузера, затем выбирается метод API для входа или регистрации. Запрос отправляется на сервер, после чего, если авторизация успешна, происходит декодирование полученного accessToken, а данные о пользователе сохраняются в локальном хранилище (localStorage). Далее в Redux передается действие login, обновляющее состояние пользователя, и отображается уведомление о том, что вход или регистрация прошли успешно.

Затем пользователь перенаправляется в раздел чата. Если же при авторизации возникает ошибка, например, неверный пароль или уже существующий e-mail, она обрабатывается и отображается соответствующее сообщение.

Интерфейс формы стилизован с помощью библиотеки Material UI. Используется компонент Box для разметки, Typography для заголовков и CustomInput для полей ввода. Если на странице находится форма регистрации, отображается дополнительное поле для ввода имени пользователя. Кнопка CustomButton отправляет форму, а внизу страницы есть ссылка (NavLink), которая позволяет переключаться между страницами входа и регистрации.

AuthPage предоставляет удобный интерфейс для работы с аутентификацией, управляет состоянием через Redux, обрабатывает ошибки и уведомляет пользователя о статусе выполнения запроса.

## 3.11 Секция выбора темы

Секция выбора темы отвечает за переключение между светлой и темной темами в приложении. Она использует MUI Theme для создания двух вариантов цветовых схем: lightTheme и darkTheme.

Переключение темы осуществляется с помощью useState. Переменная isDarkMode хранит текущее состояние темы (темная или светлая), а функция toggleTheme меняет его при нажатии на переключатель.

В Header присутствует ThemeSwitch, который визуально отображает текущее состояние темы и позволяет пользователю переключаться между режимами.

Секция выбора темы позволяет пользователю легко переключаться между светлым и темным режимами, улучшая удобство использования приложения в разных условиях освещения.

## 3.12 Секция моделей

Секция моделей является ключевой частью приложения, предоставляя пользователям возможность исследовать доступные модели нейронных сетей и получать детальную информацию о каждой из них. Эта секция включает в себя две основные страницы: страницу со списком всех моделей и отдельную страницу для детального просмотра каждой модели. Обе страницы имеют свои уникальные особенности и функциональность, что делает взаимодействие с приложением более удобным и интуитивно понятным.

**3.12.1** Страница со списком моделей. На странице со списком моделей пользователи могут просматривать все доступные модели в виде карточек. Компонент Models отвечает за загрузку данных о моделях из Redux-хранилища и их отображение. При монтировании компонента используется хук useEffect, который вызывает действие для получения списка моделей. Это позволяет гарантировать, что данные обновляются каждый раз, когда пользователь заходит на страницу.

Во время загрузки данных отображается индикатор загрузки, реализованный с помощью компонента CircularProgress, что позволяет пользователям видеть, что происходит процесс загрузки. Как только данные о моделях загружены, они отображаются в виде сетки с помощью компонента Grid. Каждая модель представлена в виде карточки с помощью компонента ModelCard, который включает название модели, краткое описание и две кнопки для перехода на страницу с подробной информацией или для начала работы с моделью.

Карточки моделей имеют хорошо структурированный дизайн, который помогает пользователям быстро ориентироваться в доступных вариантах. Это создает интуитивно понятный интерфейс, позволяя пользователям легко находить и выбирать интересующие их модели. Кроме того, использование адаптивного дизайна обеспечивает корректное отображение на различных устройствах, что способствует улучшению пользовательского опыта.

**3.12.2** Страница детали модели. Страница детали модели предоставляет пользователям более глубокую информацию о выбранной модели. Компонент ModelPage использует хук useParams для получения идентификатора модели из URL и извлекает соответствующие данные из Redux-хранилища. Это позволяет пользователям легко получать доступ к информации о конкретной модели, просто кликнув на её карточку.

На странице отображается полная информация о модели, включая название, описание, дату создания и последнего обновления. Пользователи могут также видеть ссылки на датасет и саму модель, что позволяет им быстро перейти к необходимым ресурсам. Это особенно полезно для разработчиков и исследователей, которые хотят получить доступ к данным и инструментам, связанным с моделью.

Кроме текстовой информации, на странице предусмотрены графики, которые показывают статистику продуктивности и время обучения модели. Для визуализации данных используются компоненты из библиотеки recharts, что позволяет пользователям наглядно оценивать эффективность модели по различным метрикам. Графики отображают данные о точности модели на разных эпохах и времени обучения, предоставляя пользователям ценную информацию для анализа и принятия решений.

Если модель не найдена, пользователю отображается соответствующее сообщение, уведомляющее его о том, что выбранная модель отсутствует. Это предотвращает путаницу и помогает пользователям понимать, что произошло.

Таким образом, страница деталей модели не только информирует о характеристиках, но и позволяет пользователю анализировать производительность и время обучения, что является важным аспектом для разработчиков и исследователей.

## 3.13 Секция взаимодействия с нейросетями

Секция взаимодействия с нейросетями представляет собой важный компонент приложения, который позволяет пользователям взаимодействовать с моделями нейронных сетей в реальном времени. Эта секция обеспечивает пользователям возможность общения с моделями, задавать вопросы и получать ответы, что делает процесс обучения и тестирования моделей более интерактивным и наглядным.

Компонент Chat служит основным контейнером для этой секции. Он использует параметры из URL для определения, с какой моделью взаимодействует пользователь. В зависимости от наличия идентификатора модели, компонент отображает либо интерфейс для общения с моделью, либо экран, предлагающий пользователю выбрать модель для начала чата.

Внутри компонента Chat находится несколько подкомпонентов. ChatSidebar отображает список доступных моделей, позволяя пользователям быстро переключаться между ними. ChatBody отвечает за отображение текущего чата, включая информацию о выбранной модели, что помогает пользователю понять, с какой моделью он взаимодействует. В этом компоненте также можно добавить функциональность для отображения сообщений, полученных от модели, и вопросов, заданных пользователем.

Компонент ChatBottom предоставляет интерфейс для ввода сообщений. Пользователи могут вводить текстовые сообщения, а также прикреплять изображения, что обогащает взаимодействие. Кнопка отправки сообщения инициирует процесс получения ответа от модели. Этот компонент также может быть дополнен функциональностью для обработки и отображения ответов, полученных от нейросети.

Если пользователь не выбрал модель, отображается компонент ChatLanding, который предлагает ему выбрать модель из списка. Это обеспечивает интуитивно понятный интерфейс и помогает пользователям легко ориентироваться в приложении.

## 3.14 Секция комментариев

Секция комментариев отвечает за отображение и добавление отзывов пользователей к определённой нейронной модели. Она включает в себя три основных компонента: ModelsComments (список комментариев), ModelCommentCard (отдельный комментарий) и CreateModelComment (форма для создания нового комментария).

**3.14.1** Компонент ModelsComments. Компонент ModelsComments получает id модели в качестве параметра и загружает список комментариев при монтировании или при обновлении updateComments из Redux-хранилища. Запрос выполняется через model.comment.getComments({ modelId: id }), а полученные данные сохраняются в useState. Комментарии отображаются в виде сетки.

**3.14.2** Компонент ModelCommentCard. ModelCommentCard отвечает за оформление каждого комментария. Компонент отображает текст, дату и имя пользователя. Текст организован в блоки с интервалами.

**3.14.3** Компонент CreateModelComment. CreateModelComment позволяет пользователю оставить новый комментарий. Он содержит TextField для ввода текста и кнопку отправки. При нажатии на кнопку выполняется POST-запрос model.comment.createComment(), который отправляет данные на сервер (modelId, userId, content). Если запрос успешен, вызывается setUpdateComments() для обновления списка комментариев. При отправке отображается индикатор загрузки, а также предусмотрен вывод ошибки.

Таким образом, секция комментариев даёт пользователям возможность читать отзывы, делиться мнением о модели и взаимодействовать с сообществом.

## 3.15 Секция датасетов

Секция датасетов предназначена для отображения списка доступных наборов данных, которые можно использовать в рамках платформы. Она реализована через компонент Datasets, который при монтировании отправляет запрос к API, получает данные и отображает их в виде карточек. Полученные данные сохраняются в состоянии компонента и передаются в интерфейс для визуализации.

Каждый датасет представлен в виде карточки, содержащей название, описание и кнопку «Подробнее», которая ведёт на отдельную страницу с детальной информацией о наборе данных. Карточки размещаются в сетке, что обеспечивает удобное и адаптивное расположение элементов на разных экранах. Вся разметка оформлена с использованием библиотеки Material UI: контейнеры организованы с помощью Box, заголовки и текстовые элементы оформлены через Typography, карточки структурированы с применением Card и CardContent, а кнопка для перехода к подробной информации расположена внутри CardActions.

Секция обеспечивает удобный просмотр и навигацию по доступным датасетам, а также может быть расширена функционалом фильтрации, поиска и пагинации для более удобного взаимодействия пользователей с данными.

# 7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И РЕАЛИЗАЦИИ НА РЫНКЕ ВЕБ-ПЛАТФОРМЫ ДЛЯ РАБОТЫ С МОДЕЛЯМИ НЕЙРОННЫХ СЕТЕЙ

## 7.1 Характеристика разработанного по индивидуальному заказу программного средства

Создаваемое программное решение представляет собой веб-платформу, предназначенную для студентов и разработчиков, позволяющую создавать, загружать и тестировать предобученные модели нейронных сетей. Платформа предоставит доступ к моделям в форматах ONNX и h5, а также обеспечит управление и хранение датасетов. Ключевые функции будут включать возможность добавления заметок и комментариев, а также сбор и отображение статистики производительности для каждой модели.

Основной целью разработки является создание интуитивно понятного инструмента, который позволит студентам разрабатывать свои модели и загружать их на платформу. Пользователи смогут отправлять запросы к различным моделям, что даст им возможность экспериментировать и обучаться на практике. Это создаст среду для активного обмена знаниями и опытом в области нейронных сетей.

Платформа будет доступна как в локальной сети, так и в облачных средах, обеспечивая гибкость и масштабируемость. В условиях растущего интереса к нейронным сетям и необходимости практического применения знаний данное решение существенно ускорит процесс обучения и улучшит результаты исследований.

Заказчиком платформы является образовательная организация, стремящаяся предоставить студентам современные инструменты для изучения нейросетевых технологий. В настоящее время студенты сталкиваются с недостатком практических возможностей для работы с моделями, что замедляет их обучение и ограничивает возможности для экспериментов. Внедрение данной платформы упростит эти процессы и повысит общую продуктивность обучения.

Актуальность проекта подтверждается увеличением интереса к нейросетевым технологиям и необходимостью предоставления студентам практических инструментов для их изучения. Внедрение платформы устраняет трудности, связанные с доступом к моделям и их тестированием, позволяя пользователям сосредоточиться на практическом применении полученных знаний.

## 7.2 Расчет затрат на разработку и цена программного средства, созданного по индивидуальному заказу

**7.2.1** Расчет основной заработной платы разработчиков. Расчет затрат на основную заработную плату разработчиков основывается на количестве сотрудников, участвующих в создании программного продукта, их месячной зарплате и уровне сложности выполняемых задач. Затраты на основную заработную плату вычисляются по следующей формуле

|  |  |  |
| --- | --- | --- |
|  |  | (7.1) |

где – коэффициент премий и иных стимулирующих выплат;

– категории исполнителей, занятых разработкой программного средства;

– часовая заработная плата исполнителя *i*-й категории, р;

– трудоемкость работ, выполняемых исполнителем *i*-й категории, ч.

Создание современного приложения требует слаженной работы команды специалистов, каждый из которых вносит свой уникальный вклад в общий процесс. В рамках данного проекта будут задействованы два инженера-программиста и один дизайнер, что обеспечит гармоничное сочетание технических и креативных навыков.

Первый инженер-программист отвечает за создание как клиентской, так и серверной части приложения, что предполагает глубокое понимание архитектуры и технологий разработки. Второй программист сосредоточен на интеграции и взаимодействии с нейронными моделями, что требует знаний в области машинного обучения и оптимизации алгоритмов. Дизайнер будет заниматься разработкой визуального оформления приложения, включая графику и интерфейс, что обеспечит удобный и привлекательный пользовательский опыт.

Заработная плата разработчиков взята как медианное значение для разработчиков на территории Республики Беларусь. Месячная заработная плата инженеров-программистов составляет по 700 долларов США[X], месячная зарплата дизайнера составляет 600 долларов США [X].

По состоянию на 9 марта 2025 года курс 1 доллара США составляет 3,2345 белорусских рублей. В пересчете на белорусские рубли, месячный оклад инженеров-программистов составит по 2 264,15 белорусских рублей каждый, а месячный оклад дизайнера – 1 940,70 белорусских рублей.

Часовой оклад исполнителей высчитывается путем деления месячного оклада на количество рабочих часов в месяце. На момент проведения расчета по данным Министерства труда и социальной защиты населения Республики Беларусь количество рабочих часов составляет 167 часов [X].

Для инженеров-программистов принято считать, что для написания полностью работоспособного приложения потребуется 240 часов. Из них первый инженер-программист, отвечающий за клиентскую и серверную части, будет работать 160 часов, а второй инженер-программист, занимающийся интеграцией с нейронными моделями, потратит 80 часов. Для дизайнера количество рабочих часов остается равным 30.

Коэффициент премии и иных стимулирующих выплат приравнивается к единице, так как он уже включен в сумму заработной платы. Расчет затрат на основную заработную плату приведен в таблице 7.1.

Таблица 7.1 – Затраты на основную заработную плату разработчиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячный оклад, р | Часовой оклад, р | Трудоемкость работ, ч | Итого, р |
| Инженер-программист | 2 264,15 | 13,55 | 160 | 2 169,24 |
| Инженер-программист | 2 264,15 | 13,55 | 80 | 1 084,62 |
| Дизайнер | 1940,7 | 11,62 | 30 | 348,62 |
| Итого | | | | 3 602,48 |
| Премия и иные стимулирующие выплаты (0%) | | | | 0 |
| Всего затраты на основную заработную плату разработчиков | | | | 3 602,48 |

**7.2.2**Расчет дополнительной заработной платы разработчиков. Расчёт затрат на дополнительную заработную плату команды разработчиков рассчитывается по формуле

|  |  |  |
| --- | --- | --- |
|  |  | (7.2) |

где – норматив дополнительной заработной платы.

Значение норматива дополнительной заработной платы принят за 15 %.

**7.2.3** Расчет отчислений на социальные нужды. Размер отчислений на социальные нужды определяется согласно ставке отчислений, которая на март 2025 г. равняется 35%. Из них 29% отчисляется на пенсионное страхование, а 6% − на социальное страхование. Расчёт отчислений на социальные нужды вычисляется по формуле

|  |  |  |
| --- | --- | --- |
|  |  | (7.3) |

где норматив отчислений в ФСЗН.

**7.2.4** Расчет прочих расходов. Расчёт затрат на прочие расходы определяется при помощи норматива прочих расчётов. По данным компании эта величина имеет значение 30%. Расчёт прочих расходов вычисляется по формуле

|  |  |  |
| --- | --- | --- |
|  |  | (7.4) |

где норматив прочих расходов.

**7.2.5** Расчет общей суммы затрат на разработку. Общая сумма затрат на разработку определяется как сумма ранее вычисленных расходов: на основную заработную плату разработчиков, дополнительную заработную плату разработчиков, отчислений на социальные нужды и прочих расходов. Значение определяется по формуле

|  |  |  |
| --- | --- | --- |
|  |  | (7.5) |

**7.2.6** Расчет плановой прибыли, включаемой в цену программного средства. Расчет плановой прибыли, которая включена в цену программного средства, производится по формуле

|  |  |  |
| --- | --- | --- |
|  |  | (7.6) |

где – затраты на разработку программного средства;

– рентабельность затрат на разработку программного средства.

Исходя из фактических данных компании рентабельность затрат принята за 30%.

**7.2.7** Расчет отпускной цены программного средства. Расчет отпускной цены программного средства вычисляется по формуле

|  |  |  |
| --- | --- | --- |
|  |  | (7.7) |

Формирование отпускной цены программного средства осуществляется в соответствии с методикой, представленной в таблице 7.2.

Таблица 7.2 – Формирование цены программного средства на основе затрат

|  |  |  |
| --- | --- | --- |
| Наименование статьи затрат | Формула/таблица для расчета | Значение, р |
| 1 | 2 | 3 |
| 1 Основная заработная плата разработчиков | Таблица 7.1 | 3 602,48 |
| 2 Дополнительная заработная плата разработчиков |  | 540,37 |
| 3 Отчисления на социальные нужды |  | 1 449,99 |
| 4 Прочие расходы |  | 1 080,74 |
| 5 Общая сумма затрат на разработку |  | 6 673,58 |
| 6 Плановая прибыль, включаемая в цену программного средства |  | 2 002, 07 |

Продолжение таблицы 7.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 7 Отпускная цена программного средства |  | 8 675,65 |

## 7.3 Расчет результата от разработки и использования программного средства, созданного по индивидуальному заказу

**7.3.1** Расчет результата от разработки программного средства для компании-разработчика. Для  компании-разработчика  экономическим эффектом от создания и продажи программного средства является увеличение чистой прибыли, полученной от его разработки и реализации заказчику.

Прирост чистой прибыли может быть рассчитан исходя из того, как была установлена цена программного средства: установлена отпускная цена, сформированная на основе затрат на разработку; цена установлена в процессе переговоров или же цена установлена на основе средних рыночных цен.

Так как программное средство создано под нужды и требования компании-заказчика и является уникальным, цена устанавливается в процессе переговоров между разработчиком и заказчиком. В этом случае экономический эффект для компании-разработчика будет определяться по формуле

|  |  |  |
| --- | --- | --- |
|  |  | (7.8) |

где – цена, установленная на основе переговоров между заказчиком и

компанией-разработчиком (без НДС), р;

– затраты на разработку программного средства, р;

– ставка налога на прибыль.

В случае, если компания-разработчик является резидентом Парка высоких технологий, она освобождена от уплаты налога на прибыль. В данном случае компания не является резидентом Парка высоких технологий и на момент расчетов 9 марта 2025 года ставка налога на прибыль равна 20%. По раннее выполненным расчетам затраты на разработку составили 6673,58 Белорусских рублей. В процессе переговоров была установлена цена программного средства в 10000 Белорусских рублей. Исходя из этих данных, можно вычислить экономический эффект для компании-разработчика:

**7.3.2** Расчет результата от разработки программного средства для компании-заказчика. Экономический эффект компании-заказчика основан на экономии на материальных ресурсах.

Экономия на материальных ресурсах может быть достигнута за счет подписок на специализированные образовательные платформы и онлайн-сервисы, которые студенты используют для обучения и повышения своих навыков. Эти подписки позволяют избежать затрат на традиционные учебные материалы и инструменты, необходимые для выполнения учебных заданий.

Экономический эффект компании-заказчика на экономии на материальных ресурсах рассчитывается по следующей формуле

|  |  |
| --- | --- |
|  | (7.9) |

где – коэффициент транспортных расходов;

– норма расхода материальных ресурсов при выполнении

работ сотрудниками до и после внедрения программного средства, нат. ед.;

– цена за единицу материального ресурса, р;

– плановый объем работ, выполняемых сотрудником с использованием

программного средства;

По данным компании-заказчика, коэффициент транспортных расходов составит 1,1. Норма расхода материальных ресурсов при выполнении работ сотрудниками до внедрения программного средства составляет 50 нат. ед., после – 10 нат. ед. Цена за единицу материального ресурса, согласно данным компании-заказчика, составляет 50 рублей. Планируемый объем работ, выполняемых сотрудником с использованием программного средства, равен 5 единицам.

В итоге экономия на материальных ресурсах за счет снижения их расхода составит:

## 7.4 Расчет показателей экономической эффективности разработки и использования программного средства

Расчет показателей экономической эффективности разработки программного средства. Для компании-разработчика программного средства оценка экономической эффективности разработки осуществляется с помощью расчета простой нормы прибыли (рентабельности инвестиций (затрат) на разработку программного средства) по формуле

|  |  |
| --- | --- |
|  | (7.10) |

где – прирост чистой прибыли, полученной от разработки программного

средства компанией-разработчиком по индивидуальному заказу, р;

– затраты на разработку программного средства организацией разработчиком, р.

Оценка экономической эффективности разработки рассчитана как:

## 7.5 Вывод об экономической целесообразности реализации проектного решения

Проведённые расчёты технико-экономического обоснования подтверждают целесообразность разработки данного программного средства. Общие затраты на его создание и внедрение составили 6 673,58 белорусских рублей, в то время как отпускная цена была установлена на уровне 10 000 белорусских рублей.

При этом прирост чистой прибыли для компании-разработчика составил 2 661,13 белорусских рублей, а рентабельность инвестиций за проект достигла 39,87%. Это указывает на то, что разработка данного программного продукта имеет смысл, и его реализация по установленной цене обоснована.

Однако следует учитывать, что успешность платформы во многом зависит от её фактического использования студентами. Если студенты не начнут активно пользоваться предложенным решением, то затраты на его создание могут оказаться нецелесообразными. Для достижения успеха важно обеспечить вовлеченность пользователей и адаптацию платформы к их потребностям.

В целом, при активном вовлечении студентов и постоянной поддержке проект может стать важным инструментом в образовательном процессе и внести значительный вклад в обучение студентов.

Заключение

В ходе преддипломной практики была проведена работа по изучению и проектированию веб-платформы предназначенной для работы с предобученными нейронными моделями. Основное внимание было уделено изучению технологий, проектированию архитектуры и созданию графических материалов, необходимых для дальнейшей разработки.

В рамках практики были изучены и применены следующие технологии и инструменты:

– клиент-серверная архитектура с использованием React на клиентской части и Node.js/Express на серверной части;

– основы работы с базами данных (PostgreSQL) и принципы организации хранения моделей;

– WebSocket для реализации взаимодействия клиента с сервером в реальном времени;

– использование MUI и Redux для организации пользовательского интерфейса и управления состоянием приложения.

Кроме того, в рамках преддипломной практики были разработаны графические материалы:

– вводный плакат;

– cтруктурная схема приложения;

– диаграмма классов.

Кроме того, было составлено технико-экономическое обоснование разработки и реализации проекта на рынке, в котором была оценена стоимость разработки и потенциальная прибыль от платформы.

Практика позволила определить ключевые требования к системе, спроектировать её архитектуру и подготовить необходимые проектные материалы для дальнейшей реализации. Полученные знания и результаты лягут в основу дипломного проектирования, обеспечивая более эффективную разработку веб-платформы.

Список использованных источников

[1] Всё, что вы хотели знать о перцептронах Розенблатта, но боялись спросить [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/sberdevices/articles/529932/

[2] Почему ONNX так популярен в ML: конвертации, утилиты и инференс [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/757080/

[3] Как использовать HDF5-файлы в Python [Электронный ресурс] – Режим доступа: https://habr.com/ru/companies/otus/articles/416309/

[4] REST API [Электронный ресурс] – Режим доступа: https://blog.skillfactory.ru/glossary/rest-api/

[5] Node.js [Электронный ресурс]. – Режим доступа: https://nodejs.org

[6] Express.js [Электронный ресурс]. – Режим доступа: https://expressjs.com

[7] React [Электронный ресурс]. – Режим доступа: https://reactjs.dev

[8] JWT [Электронный ресурс]. – Режим доступа: https://jwt.io

[9] What are WebSockets and Why are they Used? [Электронный ресурс] – Режим доступа: https://blog.algomaster.io/p/websockets

[10] Redux [Электронный ресурс]. – Режим доступа: https://redux.js.org

[11] Materia-UI [Электронный ресурс]. – Режим доступа: https://mui.com

[12] Poe [Электронный ресурс]. – Режим доступа: https://redux.js.org

[13] Arena от LMSYS [Электронный ресурс]. – Режим доступа: http://arena.lmsys.org

[14] PostgreSQL [Электронный ресурс]. – Режим доступа: https://www.postgresql.org

СТП 01-2024 Дипломные проекты (работы). Общие требования [Электронный ресурс]. – 2024 – Режим доступа: https://www.bsuir.by/m/12\_100229\_1\_185678.pdf

 Вычислительные машины, системы и сети: дипломное проектирование (методическое пособие) [Электронный ресурс]. – 2019 – Режим доступа: https://www.bsuir.by/m/121002291136308.pdf

Приложение А

(обязательное)

**Вводный плакат. Плакат**

Приложение Б

(обязательное)

**Веб-платформа для работы с моделями нейронных сетей.**

**Схема структурная**

Приложение В

(обязательное)

**Веб-платформа для работы с моделями нейронных сетей.**

**Диаграмма классов**

Приложение Г

(обязательное)

**Веб-платформа для работы с моделями нейронных сетей.**

**Диаграмма компонентов**