Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Факультет «Информационных технологий и анализа больших данных»

Выпускная квалификационная работа

на тему «Разработка приложения для взаимодействия с GPT-моделью»

(наименование темы выпускной квалификационной работы)

Направление подготовки 09.03.03 «Прикладная информатика»

(код и наименование направления подготовки)

Профиль «ИТ-сервисы и технологии обработки данных в экономике и финансах»

(наименование направленности)

Выполнил студент учебной группы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПИ21-1\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(номер учебной группы)

\_\_\_\_\_\_Ишмаев Руслан Ильдарович\_\_\_\_

(фамилия, имя, отчество полностью)

Руководитель к.т.н., доцент \_

(ученая степень и/или звание)

\_\_\_\_Алюнов Александр Николаевич\_\_\_\_

(фамилия, имя, отчество полностью)

**ВКР соответствует предъявляемым**

**требованиям**

должность

ученая степень

Фамилия

(И.О. Фамилия)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_ г.

Москва – 202\_г.

Оглавление

[Введение 3](#_Toc195398659)

[Теоретические основы языковых моделей 3](#_Toc195398660)

[История и эволюция языковых моделей 3](#_Toc195398661)

[Ранние этапы: статистические языковые модели 3](#_Toc195398662)

[Переход к нейронным сетям (1990-2010-е годы) 3](#_Toc195398663)

[Революция трансформеров (2017-по настоящее время) 3](#_Toc195398664)

[Технология GPT: принципы работы и архитектура 3](#_Toc195398665)

[Применение GPT в различных областях 3](#_Toc195398666)

[Популярные реализации GPT: лидеры рынка и их особенности 3](#_Toc195398667)

[Проектирование приложения 3](#_Toc195398668)

[Определения требования к приложению 3](#_Toc195398669)

[Требования к backend-части приложения 3](#_Toc195398670)

[Требования к frontend-части приложения 3](#_Toc195398671)

[Архитектурные решения приложения: теоретическое обоснование и сравнительный анализ 3](#_Toc195398672)

[Теоретические основы архитектурных подходов 3](#_Toc195398673)

[Сравнительный анализ архитектурных подходов 3](#_Toc195398674)

[Обоснование выбора архитектуры 4](#_Toc195398675)

[Теоретическая значимость выбранных решений: 4](#_Toc195398676)

# Введение

Современные технологии искусственного интеллекта стремительно развиваются, предлагая новые возможности для решения задач, которые еще недавно или были полностью неразрешимы методами, предлагаемыми машинным обучением, или решались неэффективно. Одним из наиболее значимых достижений в этой области стало появление GPT-моделей, которые способны обрабатывать и генерировать текст на естественном языке с высокой степенью осмысленности. Такие модели теперь находят применение в самых различных областях: от автоматизации бизнес-процессов до создания адаптируемых под пользователя приложений и цифровых ассистентов.

В последние несколько лет наблюдается значительный рост не только интереса к приложениям, использующих возможности крупных языковых моделей, но и количества реализации самих моделей. Это связано не только с их высокой точностью и адаптивностью, но и с их потенциалом для упрощения различных аспектов жизни человека. Приложения, основанные на GPT, могут выполнять широкий спектр функций: от написания текстов до сложного анализа данных.

Целью данной работы является разработка приложения, которое позволило бы более эффективно взаимодействовать с различными GPT-моделями, представляя пользователю понятный интерфейс и широкий функционал. В рамках работы будут рассмотрены как теоретические аспекты работы языковых моделей, так и современные подходы к реализации языковых моделей, будут рассмотрены современные реализации, пользующиеся наибольшей популярностью. На основе проведенного анализа будет спроектировано и реализовано приложение, которое позволит пользователю выбирать модель из набора доступных для дальнейшего взаимодействия с ней, что продемонстрирует практическую ценность и возможность использования GPT-моделей в различных сценариях.

Для достижения цели были поставлены следующие задачи:

* Изучение теоретических основ языковых моделей и технологии GPT;
* Изучение наиболее популярных реализаций языковых моделей, построенных на технологии GPT;
* Реализация программного интерфейса для взаимодействия с различными GPT-моделями;
* Разработка графического интерфейса;
* Проведение тестирования и апробация готового приложения.

Объектом дипломного исследования является семейство языковых моделей, построенных на GPT-технологии. Предметом – возможность создания информационной системы для взаимодействия с различными генеративными моделями.

Актуальность выбранной темы обусловлена кратным ростом использования технологий искусственного интеллекта как в повседневной жизни, так и в рамках различных бизнес-процессов. Согласно статистике, инвестиции по миру в развитие искусственного интеллекта значительно выросли за последнее десятилетие. Инвестиции же в генеративный искусственный интеллект на конец 2023 года составили 25.2 миллиардов долларов США, что примерно в 8 раз больше, чем на конец 2022 года. Данная статистика подтверждает цифрами заинтересованность рынка в искусственном интеллекте, в частности GPT-моделях.

Полученные наработки могут быть полезны для компаний, создающих решения по взаимодействию с искусственными интеллектом, а также людям, часто использующим искусственный интеллект в своей повседневной жизни.

# Теоретические основы языковых моделей

## История и эволюция языковых моделей

История языковых моделей тесно связана с развитием вычислительных технологий, лингвистки и машинного обучения. На протяжении нескольких десятилетий ученые, инженеры и исследователи разрабатывают методы для обработки естественного языка (Natural Language Processing, NLP), которые позволяют создавать современные модели, такие как GPT. Рассмотрим основные этапы эволюции языковых моделей.

### Ранние этапы: статистические языковые модели

В начале XX века зародились первые идеи о моделировании естественного языка при помощи математики, однако только с развитием компьютеров в середине XX века стало возможно реализовать эти идеи на практике.

#### N-граммные модели (1950-1970-е годы)

Одной из первых попыток моделировать языки стали N-граммные модели, также известные как статистически модели. Такие модели анализируют последовательность слов и рассчитывают вероятность появления каждого слова на основе предыдущих. Например, вероятность слова в триграммной модели рассчитывается на основе двух предыдущих слов.

Основные преимущества и недостатки данной модели представлены в таблице 1. Несмотря на то, что модель действительно простая в реализации и дешевая по ресурсам, недостатки, не позволяющие воспринимать контекст больше, чем N слов, а также использование только самых часто-встречаемых слов, оказались главными ограничениями. Из-за этих факторов N-граммные модели использовались, в основном, только в авто-дополнении текста, также известное как Т9.

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| Простота реализации | Ограниченность контекста |
| Скорость работы | Генерация редких последовательностей слов |

Таблица 1

#### Модели на основе скрытых марковских процессов

В 1970–1980-х годах в обработке текста начали использовать скрытые марковские модели, которые рассматривали текст как последовательность состояний. HMM оказались полезными для задач, таких как распознавание речи и разметка текста.

Скрытые марковские модели (Hidden Markov Models, HMM) основаны на статистическом подходе к моделированию последовательностей данных, где наблюдаемые события зависят от скрытых (латентных) состояний, которые эволюционируют согласно марковскому процессу первого порядка. Принцип действия HMM заключается в следующем:

1. Марковское свойство скрытых состояний: Последовательность скрытых состояний . Т.е. удовлетворяет марковскому свойству, где вероятность перехода в следующее состояние зависит только от текущего состояния.
2. Наблюдения зависят от скрытых состояний: Наблюдаемая последовательность генерируется из скрытых состояний с вероятностями, заданными функцией эмиссии. То есть, каждое наблюдение зависит только от текущего скрытого состояния .
3. Параметры модели:
   1. Начальные вероятности определяют распределение первого скрытого состояния.
   2. Матрица переходов задаёт вероятности переходов между скрытыми состояниями.
   3. Матрица эмиссий определяет вероятность наблюдения при нахождении в состоянии .
4. Основные задачи:
   1. Оценка вероятности последовательности: Вычисление , где , с помощью алгоритма вперёд-назад (Forward-Backward).
   2. Декодирование: Поиск наиболее вероятной последовательности скрытых состояний , породившей наблюдения , с использованием алгоритма Витерби.
   3. Обучение модели: Оптимизация параметров по наблюдаемым данным с использованием алгоритма Баум-Велша.

Основные преимущества и недостатки данной модели представлены в таблице 2.

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| Простота и интуитивность | Ограниченность в представлении сложных зависимостей |
| Доказанная эффективность для задач с последовательными данными | Не всегда хорошо работает с большими объемами данных или задачах с высокой сложностью |
| Хорошо изученные алгоритмы |  |

Таблица 2

### Переход к нейронным сетям (1990-2010-е годы)

С развитием вычислительных мощностей и нейросетевых подходов началась новая эра языковых моделей.

#### Рекуррентные нейронные сети (RNN) (1990-е годы)

Рекуррентные нейронные сети (Recurrent Neural Networks, RNN) стали важным этапом в развитии технологий обработки последовательных данных, таких как текст, аудио и временные ряды. Они были разработаны для решения задачи моделирования последовательностей, где текущий элемент данных зависит от предыдущих. Это делает их особенно полезными для задач обработки естественного языка, где порядок слов играет ключевую роль.

Ключевая особенность RNN заключается в том, что они имеют рекуррентные связи, позволяющие сохранять "память" о предыдущих шагах. Это достигается за счет введения скрытого состояния (hidden state), которое обновляется на каждом временном шаге и хранит информацию о предыдущих элементах последовательности.

Преимущества и недостатки RNN-моделей представлены в таблице 3.

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| Обработка последовательностей произвольной длины | Затухающие и взрывающиеся градиенты |
| Память о предыдущих шагах | Краткосрочная память |
| Гибкость – возможность использование в задачах разного типа (классификация, генерация, регрессия, и т.д.) | Медленная скорость обучения |

Таблица 3

#### LSTM и GRU (1997-2010-е годы)

Рекуррентные нейронные сети (RNN) имеют ограничения, такие как проблема затухающих и взрывающихся градиентов, что делает их малопригодными для работы с длинными последовательностями. Для решения этих проблем были разработаны две ключевые модификации RNN — LSTM (Long Short-Term Memory) и GRU (Gated Recurrent Unit).

LSTM была предложена в 1997 году Шмидхубером и Хохрайтером (Hochreiter Schmidhuber). Основная цель LSTM — эффективно сохранять информацию на длительных временных интервалах, предотвращая затухание или взрыв градиентов. Для этого LSTM использует специальные механизмы управления потоком информации через "ячейки памяти" (memory cells) и "гейты" (gates).

Преимущества и недостатки LSTM-моделей представлены в таблице 4.

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| Долгосрочная память | Медленное обучение |
| Гибкость – благодаря гейтам модель может динамически решать, что запоминать, а что забывать. | Высокие вычислительные затраты |
| Широкое применение |  |

Таблица 4

GRU была предложена в 2014 году Чо и его коллегами (Cho et al.) как упрощенная альтернатива LSTM. GRU сохраняет ключевые идеи LSTM (гейты для управления потоком информации), но устраняет некоторые элементы, такие как отдельная ячейка памяти. Это делает GRU проще и быстрее в обучении, при этом она достигает схожих результатов.

Преимущества и недостатки GRU-моделей представлены в таблице 5.

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| Легковесность | Ограниченная память |
| Эффективность | Меньшая гибкость |
| Универсальность |  |

Таблица 5

#### Word2Vec и GloVe (2013-2015-е годы)

С появлением методов Word2Vec и GloVe в 2013–2015 годах обработка естественного языка (Natural Language Processing, NLP) сделала значительный шаг вперед. Эти методы предложили новый способ представления слов в виде векторов, что позволило моделям понимать семантические и синтаксические связи между словами. Такой подход открыл новые возможности для обработки текста и стал важной основой для развития современных языковых моделей.

Word2Vec — это метод, предложенный Томашем Миколовым (Tomas Mikolov) и его командой в 2013 году. Это нейросетевая модель, которая преобразует слова в векторы фиксированной длины, сохраняя их семантические и синтаксические свойства. Векторы, созданные Word2Vec, называются эмбеддингами слов (word embeddings).

Word2Vec основывается на гипотезе распределения: "Слова, которые встречаются в похожих контекстах, имеют схожее значение". Например, слова "кофе" и "чай" часто встречаются в схожих предложениях ("пить кофе", "налить чай"), поэтому их векторные представления должны быть близки друг к другу.

Преимущества и недостатки Word2Vec представлены в таблице 6.

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| Семантическая близость | Не учитывает глобальную структуру текста |
| Эффективность | Не масштабируется для больших текстовых коллекций |

Таблица 6

GloVe (Global Vectors for Word Representation) был предложен в 2014 году исследователями Стэнфордского университета. В отличие от Word2Vec, GloVe использует статистический подход для создания эмбеддингов, комбинируя локальную и глобальную информацию о словах.

GloVe строит эмбеддинги, анализируя совместное появление слов в тексте. Это означает, что модель учитывает, как часто два слова встречаются вместе в одном контексте по сравнению с тем, как часто они встречаются по отдельности.

Преимущества и недостатки GloVe представлены в таблице 7.

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| Глобальная информация | Требует больше памяти |
| Эффективность | Не подходит для потоковых данных |

Таблица 7

### Революция трансформеров (2017-по настоящее время)

Настоящая революция в области языковых моделей произошла с появлением архитектуры трансформеров.

#### Появление трансформеров (2017)

В 2017 году исследователи Google представили архитектуру трансформеров в статье "Attention is All You Need". Трансформеры заменили рекуррентные сети механизмом внимания (attention), который позволяет модели эффективно фокусироваться на ключевых частях текста, независимо от его длины.

Преимущества трансформеров:

* Параллельная обработка данных (ускорение обучения).
* Учет долгосрочного контекста.
* Гибкость для различных задач NLP.

#### BERT (2018)

В 2018 году Google представил модель BERT (Bidirectional Encoder Representations from Transformers), которая впервые использовала двунаправленный контекст. Это означало, что модель могла учитывать не только слова до текущего, но и слова после него. BERT стал стандартом для обработки текста и улучшил результаты во многих NLP-задачах.

BERT основан на энкодере трансформера (Transformer Encoder).

Архитектура BERT, ключевые особенности:

* Маскированное языковое моделирование (Masked Language Modeling, MLM)
* Предсказание следующего предложения (Next Sentence Prediction, NSP)

Преимущества и недостатки BERT представлены в таблице 8.

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| Двунаправленный контекст | Высокая вычислительная сложность |
| Универсальность | Ограниченность в генерации текста |
| Качество |  |

Таблица 8

#### GPT (2018-по настоящее время)

GPT был разработан OpenAI, и его первая версия была выпущена в 2018 году. В отличие от BERT, GPT ориентирован на генерацию текста, что делает его идеальным для задач, связанных с созданием осмысленного и связного текста.

GPT использует однонаправленный контекст, анализируя текст слева направо. Это позволяет модели предсказывать следующее слово в последовательности, основываясь на предыдущих словах.

GPT основан на декодере трансформера (Transformer Decoder).

Архитектура GPT, Ключевые особенности:

* Авторегрессионное обучение
* Обучение на больших корпусах

Преимущества и недостатки GPT-моделей представлены в таблице 9.

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| Генерация текста | Однонаправленный контекст |
| Гибкость | Сложность обучения |
| Масштабируемость | Риск генерации некорректной информации |

Таблица 9

## Технология GPT: принципы работы и архитектура

GPT (Generative Pre-trained Transformer) — это мощная языковая модель, разработанная OpenAI, которая способна генерировать текст, имитирующий человеческую речь. Она основана на архитектуре трансформеров, предложенной в статье "Attention is All You Need" 2017, и представляет собой одно из наиболее значимых достижений в области обработки естественного языка (NLP). GPT отличается своей способностью понимать и генерировать текст, что делает её универсальным инструментом для множества задач.

#### Принципы работы GPT

GPT работает на основе трёх ключевых этапов: предварительное обучение (pre-training), дальнейшая настройка (fine-tuning) и генерация текста (inference). Рассмотрим каждый из них подробнее:

1. Предварительное обучение (Pre-training)  
   На этом этапе модель обучается на огромных объемах текстовых данных (например, книги, статьи, веб-страницы). Основная задача модели — предсказать следующее слово в тексте, основываясь на предыдущих словах. Этот процесс называется авторегрессионным обучением.  
   Обучение через трансформеры:  
   Модель использует механизм внимания (attention), чтобы фокусироваться на наиболее значимых частях текста. Это позволяет учитывать контекст даже на большом расстоянии.
2. Дальнейшая настройка (Fine-tuning)  
   После предварительного обучения модель может быть дополнительно настроена на конкретные задачи или области. Например:
   1. Генерация текстов для определённой тематики.
   2. Ответы на вопросы.
   3. Перевод с одного языка на другой.

Дальнейшая настройка может быть выполнена с использованием небольших специализированных наборов данных, что делает GPT гибкой для множества приложений.

1. Генерация текста (Inference)  
   После обучения модель может использоваться для генерации текста. Пользователь задаёт начальную последовательность слов (промпт), а модель продолжает её, используя вероятностное предсказание следующих слов.

#### Архитектура GPT

GPT построен на основе архитектуры трансформеров, а именно её части — декодера трансформера. Трансформеры были предложены в 2017 году и стали революцией в обработке последовательных данных. Основной компонент трансформера — это механизм внимания (attention), который позволяет модели эффективно работать с длинными последовательностями текста

Компоненты трансформера:

1. Входные эмбеддинги (Input Embeddings)  
   Каждое слово или токен преобразуется в плотный вектор фиксированной длины. Эти эмбеддинги содержат информацию о значении слова.
2. Позиционные эмбеддинги (Positional Embeddings)  
   Поскольку трансформеры обрабатывают текст параллельно, модель не знает порядка слов. Позиционные эмбеддинги добавляют информацию о позиции слова в предложении.
3. Механизм внимания (Attention Mechanism)  
   Механизм самовнимания (Self-Attention) позволяет модели определять важность каждого слова в контексте текущего слова. Это позволяет учитывать как близкие, так и дальние связи между словами.
4. Фидфорвардные слои (Feedforward Layers)  
   После механизма внимания используется многослойная нейронная сеть для обработки информации.
5. Нормализация и остаточные связи (Layer Normalization Residual Connections)  
   Эти компоненты помогают стабилизировать обучение и ускоряют сходимость модели.

GPT использует только декодер трансформера, в отличие от, например, BERT, который использует энкодер. Основная функция декодера — предсказывать следующее слово, основываясь на предыдущих.

Архитектура декодера в GPT:

1. Маскированное внимание (Masked Attention)
2. Иерархия слоёв

## Применение GPT в различных областях

GPT (Generative Pre-trained Transformer) стала одной из наиболее универсальных технологий в области искусственного интеллекта благодаря своей способности понимать, анализировать и генерировать текст. Её применение охватывает множество сфер, от бизнеса и образования до медицины и творчества. Рассмотрим, как GPT используется в различных областях и какие задачи она помогает решать.

1. Бизнес и корпоративные процессы
   1. Чат-боты и виртуальные ассистенты  
      GPT используется для создания умных чат-ботов и виртуальных помощников, которые могут вести диалог с клиентами, отвечать на вопросы и помогать решать проблемы.
   2. Генерация деловых документов  
      GPT может автоматически создавать отчёты, резюме, письма, деловые предложения и другие корпоративные документы.
   3. Анализ данных и прогнозирование  
      GPT применяется для анализа больших объемов текстовых данных, таких как отзывы клиентов, отчёты или рыночные исследования.
2. Образование и наука
   1. Образовательные платформы  
      GPT активно используется для поддержки студентов и преподавателей, предоставляя доступ к информации, объясняя сложные концепции и создавая обучающие материалы.
   2. Научные исследования  
      GPT помогает ученым анализировать большие объемы литературы, генерировать гипотезы и оформлять научные статьи.
3. Творчество и медиа
   1. Создание контента  
      GPT активно используется для генерации текстов, таких как статьи, блоги, сценарии, книги и даже стихи.
   2. Литературное творчество  
      GPT помогает авторам придумывать сюжеты, писать книги или создавать художественные произведения.
4. Программирование и разработка ПО
   1. Автоматизация кода  
      GPT может генерировать код на различных языках программирования, помогая разработчикам экономить время.
   2. Поддержка разработчиков  
      GPT может анализировать ошибки в коде, помогать с документацией и отвечать на вопросы по программированию.

GPT — это универсальная технология, которая находит применение практически во всех сферах жизни. Её способности к анализу и генерации текста открывают новые возможности для автоматизации, повышения эффективности и создания инновационных решений.

### Популярные реализации GPT: лидеры рынка и их особенности

В данной главе представлен обзор пяти наиболее популярных реализаций GPT-моделей, которые выделяются своими характеристиками, функциональностью и уровнем распространения на рынке.

#### OpenAI GPT

Разработчик: OpenAI

Ключевые особенности:

OpenAI является пионером в разработке GPT-моделей и на данный момент предлагает одну из самых мощных и широко используемых версий — GPT-4. Эта модель обладает высокой точностью генерации текста, способностью понимать сложные запросы и поддерживать многоязычное взаимодействие. GPT-4 активно используется в чат-ботах, таких как ChatGPT, а также интегрируется в различные бизнес-приложения через API.

Преимущества:

* Высокая производительность и гибкость.
* Поддержка работы с большими объемами данных.
* Возможность кастомизации для бизнес-задач.

Ограничения:

* Высокая стоимость использования API.
* Закрытый исходный код, что ограничивает исследовательское применение.

#### Google Bard

Разработчик: Google

Ключевые особенности:

Google Bard — это реализация языковой модели, основанной на архитектуре LaMDA (Language Model for Dialogue Applications). Bard ориентирован на диалоговые взаимодействия и интеграцию с экосистемой Google, включая поиск и приложения Google Workspace. Эта модель выделяется своей способностью к контекстному пониманию и поиску релевантной информации в режиме реального времени.

Преимущества:

* Глубокая интеграция с сервисами Google.
* Возможность работы с актуальными данными через поиск.
* Интуитивный интерфейс для массового пользователя.

Ограничения:

* Ограниченная доступность в некоторых регионах.

#### Anthropic Claude

Разработчик: Anthropic

Ключевые особенности:

Claude, созданный компанией Anthropic, представляет собой безопасную и этически ориентированную альтернативу GPT-моделям. Основное внимание уделяется предотвращению вредоносного использования модели и обеспечению высокого уровня безопасности при генерации текста. Claude активно используется в корпоративной среде для задач, требующих высокой степени конфиденциальности.

Преимущества:

* Усиленные меры безопасности.
* Ориентация на этическое использование.
* Поддержка долгосрочных диалогов.

Ограничения:

* Ограниченная доступность для широкой аудитории.

#### Mistral AI

Разработчик: Mistral AI

Ключевые особенности:

Mistral AI — это относительно новая, но перспективная реализация GPT-моделей, которая делает акцент на открытости и доступности технологий. Модели от Mistral AI ориентированы на высокую производительность и кастомизацию, что делает их привлекательными для исследователей и разработчиков.

Преимущества:

* Открытый исходный код
* Высокая кастомизация и гибкость.
* Поддержка сообществом разработчиков.

Ограничения:

* Ограниченный функционал по сравнению с лидерами рынка.
* Требуется больше времени на обучение и настройку.

#### DeepSeek AI

Разработчик: DeepSeek AI

Ключевые особенности:

DeepSeek AI представляет собой одну из новых разработок в области генеративных языковых моделей, ориентированную на глубокий анализ текстов и извлечение контекстуально значимой информации. Основное внимание в DeepSeek уделяется поисковым и аналитическим задачам, что делает эту модель особенно полезной для обработки больших массивов данных, исследований и бизнес-аналитики. DeepSeek активно используется в приложениях, связанных с интеллектуальным поиском, генерацией отчетов и автоматизацией обработки текстов.

Преимущества:

* Высокая производительность в задачах извлечения информации.
* Оптимизация под аналитические задачи.
* Возможность интеграции в корпоративные системы.

Ограничения:

* Ограниченная поддержка креативных задач, таких как генерация художественного текста.
* Требует значительных вычислительных ресурсов для работы с большими объемами данных.

# Проектирование приложения

## Определения требования к приложению

### Требования к backend-части приложения

#### Функциональные требования

1. Система аутентификации:
   1. Регистрация через email с верификацией выбрана как наиболее универсальный и безопасный метод
   2. JWT-токены обеспечивают stateless-аутентификацию, что важно для масштабируемости
   3. Сброс пароля через временный код соответствует современным стандартам безопасности
2. Управление чатами:
   1. Поддержка множества чатов обусловлена потребностью пользователей в тематическом разделении диалогов
   2. Сохранение истории соответствует принципу непрерывности диалога в AI-системах
3. Работа с AI-моделями:
   1. Разделение модели и промптов позволяет гибко настраивать поведение системы
   2. Архитектура предусматривает простую интеграцию новых моделей
4. Балансовая система:
   1. Токены как учетная единица обеспечивают прозрачный контроль использования ресурсов
   2. Механизм списания учитывает вычислительную сложность запросов

#### Нефункциональные требования

1. Архитектурные решения
   1. Микросервисная архитектура с четким разделением ответственности:
      1. Сервис аутентификации (auth-service)
      2. Сервис взаимодействия с AI (ai-service)
   2. Хранение данных:
      1. Отдельные PostgreSQL-кластеры для каждого сервиса
      2. MinIO-хранилища для файловых данных
   3. API-шлюз на базе nginx
2. Требования к безопасности
   1. Асимметричное шифрование JWT (RS256)
   2. Изоляция сервисов на уровне Docker-сети
   3. Раздельные учетные данные для каждого компонента
   4. Шифрование соединений (HTTPS)
3. Развертывание и эксплуатация
   1. Полная контейнеризация компонентов
   2. Управляемость через .env-конфигурации
   3. Поддержка разных окружений (dev/prod)
   4. Автоматическая настройка MinIO при развертывании
4. Масштабируемость
   1. Горизонтальное масштабирование сервисов
   2. Асинхронная обработка запросов
   3. Выделение отдельных вычислительных ресурсов для каждого компонента

#### Заключение

Сформулированные требования отражают современный подход к разработке AI-приложений:

* Обеспечивают безопасность пользовательских данных
* Позволяют гибко масштабировать систему
* Поддерживают различные сценарии использования
* Соответствуют принципам DevOps при развертывании

Все требования согласованы между собой и направлены на создание надежной, безопасной и удобной системы для работы с AI-моделями. Особое внимание уделено аспектам, важным для конечных пользователей - скорости работы, простоте интерфейса и защите личных данных.

### Требования к frontend-части приложения

#### Функциональные требования

1. Система аутентификации и авторизации
   1. Реализована регистрация новых пользователей с валидацией данных на стороне клиента
   2. Предусмотрена процедура подтверждения email через одноразовый код
   3. Обеспечен вход в систему по логину/email и паролю с проверкой учетных данных
   4. Реализовано автоматическое обновление JWT-токена при его истечении
   5. Реализован механизм восстановления пароля через подтверждение по email
   6. Ограничен доступ к защищенным маршрутам для неавторизованных пользователей
2. Управление чатами
   1. Реализовано создание новых чатов с указанием языка общения
   2. Обеспечено отображение истории сообщений в каждом чате
   3. Реализована отправка и прием текстовых сообщений в реальном времени
   4. Предусмотрена обработка и отображение системных сообщений
   5. Обеспечено отображение статуса доставки сообщения
3. Пользовательский интерфейс
   1. Реализованы основные страницы приложения:
   2. Страница авторизации
   3. Страница регистрации
   4. Страница списка чатов
   5. Страница переписки
   6. Обеспечена адаптивная верстка для различных устройств
   7. Реализована смена цветовых тем (темная/светлая)

#### Нефункциональные требования

1. Требования к производительности
   1. Время загрузки страницы не должно превышать 2 секунд
   2. Обеспечена плавная работа интерфейса при 60 FPS
   3. Оптимизирована работа с памятью для длительных сессий
   4. Реализована ленивая загрузка компонентов
2. Требования к безопасности
   1. Применено HTTPS соединение для всех API-запросов
   2. Реализована защита от XSS атак
   3. Обеспечено безопасное хранение учетных данных
   4. Реализована проверка прав доступа на уровне интерфейса
   5. Осуществляется мониторинг подозрительной активности
3. Требования к удобству использования
   1. Обеспечена понятная навигация между разделами
   2. Реализованы понятные сообщения об ошибках
   3. Обеспечена консистентность интерфейса
   4. Предусмотрены подсказки для сложных элементов интерфейса
4. Технические требования
   1. Используется React 18 в качестве основной библиотеки
   2. Применен React Router v6 для маршрутизации
   3. Использована библиотека компонентов Chakra UI
   4. Реализована работа с API через Axios
   5. Использован React Query для управления состоянием
   6. Обеспечена поддержка современных браузеров

#### Заключение

Заключение по требованиям к фронтенд-части приложения

Сформулированные требования отражают современные подходы к разработке пользовательских интерфейсов веб-приложений:

* Обеспечивают интуитивно понятное взаимодействие с системой за счет продуманного UX/UI
* Поддерживают высокую производительность при работе с динамическим контентом
* Гарантируют безопасность передачи и хранения пользовательских данных
* Позволяют адаптировать интерфейс под различные устройства и платформы
* Обеспечивают стабильную работу при высоких нагрузках

Все требования взаимосвязаны и направлены на создание:

* Удобного для пользователей интерфейса
* Надежной системы с защитой от сбоев
* Масштабируемого решения для будущего развития

Особый акцент сделан на аспектах, критически важных для пользовательского опыта:

* Быстрота отклика интерфейса
* Стабильность работы при длительном использовании
* Консистентность визуального оформления
* Простота навигации между разделами

Требования учитывают как текущие потребности пользователей, так и перспективы развития системы, что позволяет создавать конкурентноспособный продукт с длительным жизненным циклом.

## Архитектурные решения приложения: теоретическое обоснование и сравнительный анализ

### Теоретические основы архитектурных подходов

Фронтенд-архитектуры

1. Монолитная архитектура (Monolithic Frontend):
   1. Теоретическая основа: классическая модель единого кодового блока
   2. Принципы: централизованное состояние, единая точка входа
   3. Преимущества: простота начальной настройки, низкие накладные расходы
   4. Недостатки: высокое связывание компонентов, сложность масштабирования
2. Микросервисный фронтенд (Micro Frontends):
   1. Теоретическая основа: принципы domain-driven design
   2. Принципы: независимые модули с собственным жизненным циклом
   3. Преимущества: изолированность ошибок, независимое развертывание
   4. Недостатки: сложность синхронизации состояния, дублирование зависимостей
3. Модульный монолит (Modular Monolith):
   1. Теоретическая основа: принцип разделения ответственности
   2. Принципы: логическая изоляция модулей при единой сборке
   3. Преимущества: баланс между структурой и производительностью

Бэкенд-архитектуры

1. Многослойная архитектура (Layered Architecture):
   1. Теоретическая основа: принцип разделения уровней абстракции
   2. Уровни: представление, бизнес-логика, доступ к данным
   3. Преимущества: четкое разделение ответственности
2. Микросервисная архитектура:
   1. Теоретическая основа: теория ограниченных контекстов
   2. Принципы: слабая связанность, сильная когерентность
   3. Преимущества: горизонтальное масштабирование
3. Гибридная архитектура:
   1. Теоретическая основа: принцип постепенной эволюции
   2. Подход: сочетание монолита с элементами микросервисов

### Сравнительный анализ архитектурных подходов

Критерии сравнения:

1. Сложность внедрения (Implementation Complexity):
   1. Монолит: низкая (1-2)
   2. Микросервисы: высокая (4-5)
   3. Гибридная: средняя (3)
2. Коэффициент связности (Coupling Coefficient):
   1. Монолит: 0.8-1.0
   2. Микросервисы: 0.1-0.3
   3. Гибридная: 0.4-0.6
3. Показатель масштабируемости (Scalability Index):
   1. Монолит: 2/10
   2. Микросервисы: 9/10
   3. Гибридная: 6/10
4. Скорость разработки (Development Velocity):
   1. Монолит: высокая
   2. Микросервисы: низкая
   3. Гибридная: средняя

### Обоснование выбора архитектуры

Для фронтенд-части была выбрана модульная монолитная архитектура по следующим причинам:

1. Теоретические предпосылки:
   1. Соответствует принципам bounded contexts для средних проектов
   2. Позволяет реализовать концепцию vertical slicing
   3. Обеспечивает оптимальный уровень связности (0.4-0.6)
2. Практические преимущества:
   1. Сохраняет преимущества монолита при разработке
   2. Позволяет постепенно эволюционировать в микросервисную архитектуру
   3. Упрощает тестирование за счет четких границ модулей
3. Архитектурные характеристики:
   1. Удовлетворяет требованиям по масштабируемости (6/10)
   2. Обеспечивает приемлемый уровень сложности внедрения (3/5)
   3. Поддерживает необходимую скорость разработки

Для бэкенд-части была выбрана гибридная архитектура, сочетающая:

1. Многослойную структуру:
   1. Четкое разделение на контроллеры, сервисы и репозитории
   2. Соблюдение принципа единственной ответственности
   3. Упрощенное тестирование отдельных компонентов
2. Элементы микросервисов:
   1. Возможность выделения отдельных модулей в сервисы
   2. Использование шаблона API Gateway для агрегации запросов

### Теоретическая значимость выбранных решений:

Выбранные архитектурные подходы демонстрируют:

1. Применение современных парадигм:
   1. Компромисс между микросервисами и монолитом (MASA - Microservices and Monoliths Are Similar Actually)
   2. Реализация принципов адаптивного дизайна
   3. Соблюдение законов Conway при проектировании
2. Научную обоснованность:
   1. Соответствие теории модульных систем
   2. Применение принципов инкрементального проектирования
   3. Учет ограничений CAP-теоремы для распределенных систем
3. Практическую ценность:
   1. Оптимальное соотношение качества и сложности
   2. Поддержка эволюционной архитектуры
   3. Возможность постепенной трансформации системы

Выбор архитектурных решений основан на комплексном анализе:

* Теоретических моделей проектирования
* Практических ограничений проекта
* Перспектив развития системы
* Требований к качеству и надежности