МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___» _____ 2025 г.

# Разработка автоматизированной системы поддержки пользователей с применением LLM и RAG

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2025.406 ПЗ ВКР

Руководитель работы,
к. пед. н., доцент каф. ЭВМ
_____Ю.Г. Плаксина
«___» _____2025 г.

Автор работы,
студент группы КЭ-406
_____И.А. Кудряшов
«___» _____2025 г.

Нормоконтролер,
ст. преподаватель каф. ЭВМ
_____ С.В. Сяськов
«___»_____2025 г.

Челябинск-2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___» _____2025 г.

# ЗАДАНИЕ

**на выпускную квалификационную работу бакалавра**

студенту группы КЭ-406
Кудряшову Ивану Алексеевичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

**1. Тема работы: «**Разработка автоматизированной системы поддержки пользователей с применением LLM и RAG» утверждена приказом по университету от 21 апреля 2025 г. № 648-12/12.

**2. Срок сдачи студентом законченной работы:** 1 июня 2025 г.

**3. Исходные данные к работе:**

Требования к функционалу разрабатываемой системы:

- автоматическая генерация ответов с использованием LLM и RAG;
- возможность выбора домена знаний;
- реализация в виде веб-решения;
- реализация с возможностью интегрирования в сторонние ресурсы;
- реализация в формате нативного веб-компонента.

4. **Перечень подлежащих разработке вопросов:**

- анализ предметной области;
- проектирование автоматизированной системы;
- реализация автоматизированной системы;
- тестирование системы на предприятии.

5. **Дата выдачи задания:** 2 декабря 2024 г.

Руководитель работы _____*/Ю.Г. Плаксина/*

Студент _____*/И.А. Кудряшов /*

КАЛЕНДАРНЫЙ ПЛАН

| Этап | Срок сдачи | Подпись руководителя |
|---|---|---|
| Анализ предметной области | 01.03.2025 | |
| Проектирование автоматизированной системы | 25.03.2025 | |
| Реализация автоматизированной системы | 20.04.2025 | |
| Тестирование системы на предприятии | 01.05.2025 | |
| Анализ результатов, оформление отчета и прохождение обязательного нормоконтроля | 23.05.2025 | |
| Подготовка презентации и доклада | 30.05.2025 | |

Руководитель работы _____/*Ю.Г. Плаксина/*

Студент _____/*И.А. Кудряшов/*

Аннотация

И.А. Кудряшов. Разработка автоматизированной системы поддержки пользователей с применением LLM и RAG. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2025, 63 с., 15 ил., библиогр. список – 31 наим.

В рамках выпускной квалификационной работы проводится разработка автоматизированной системы поддержки пользователя с применением технологий LLM и RAG.

Целью работы является разработка автоматизированной системы поддержки пользователей на основе LLM и RAG для улучшения качества поддержки, повышения скорости ответа и значительной разгрузки операторов поддержки, а также обеспечения персонализированного подхода к каждому запросу.

В рамках работы рассматриваются подходы к организации и структурированию знаний для повышения точности и релевантности ответов системы. Особое внимание уделяется удобству взаимодействия с пользователем и возможностям интеграции системы во внутренние ресурсы предприятия, что способствует улучшению качества обслуживания и повышению эффективности работы с информационными ресурсами.

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

В последние годы наблюдается стремительный рост использования технологий искусственного интеллекта (ИИ) в различных областях, включая корпоративное управление, поддержку пользователей и автоматизацию рабочих процессов. Одним из перспективных направлений является применение больших языковых моделей (Large Language Model, LLM) в сочетании с технологией извлечения релевантной информации (Retrieval-Augmented Generation, RAG) для создания систем поддержки пользователей. Эти технологии позволяют не только автоматизировать обработку запросов, но и обеспечивать высокую точность и релевантность ответов за счет интеграции с актуальными базами знаний, что особенно востребовано в условиях цифровизации и роста объемов корпоративной информации.

Современные предприятия, работающие в условиях стремительного развития, сталкиваются с необходимостью оперативного решения множества задач, требующих высокой скорости и точности. Традиционные инструменты поддержки, такие как телефонные линии и электронная почта, зачастую не соответствуют этим требованиям. Данные недостатки особенно критичны в сложных корпоративных средах, где требуется гибкость и постоянная поддержка пользователей. В связи с этим внедрение технологий искусственного интеллекта, таких как большие языковые модели и системы извлечения и генерации информации, открывает перспективы для создания автоматизированных систем поддержки, способных значительно повысить качество поддержки сотрудников. Такие системы обеспечат высокую точность, оперативность и персонализированный подход к каждому запросу пользователя.

Использование автоматизированной системы поддержки становится особенно актуально для предприятий, находящихся в изолированном контуре, где обеспечение конфиденциальности и независимости от внешний сети становится приоритетной задачей. Примером такой организации выступает «Российский федеральный ядерный центр – Всероссийский научно-

исследовательский институт технической физики имени академика Е.И. Забабахина» (далее ВНИИТФ) [1].

В отличие от общедоступных решений, предлагаемая система на основе LLM и RAG обеспечивает автономною работу внутри предприятия, что критически важно для обеспечения безопасности данных. Интеграция такой системы во внутренние ресурсы ВНИИТФ позволит не только сократить временные затраты на поддержку пользователей, но и повысить оперативность обработки запросов, улучшить качество обслуживания и обеспечить соответствие строгим стандартам конфиденциальности. Таким образом, разработка и внедрение интеллектуальных виртуальных ассистентов представляют собой актуальное и практически значимое направление, способствующее оптимизации внутренних процессов и повышению эффективности работы предприятий в условиях современных вызовов.

Целью работы является разработка автоматизированной системы поддержки пользователей на основе LLM и RAG для улучшения качества поддержки, повышения скорости ответа и точности предоставляемой информации. Для достижения данной цели необходимо решить следующие задачи.

1. Провести анализ предметной области.
2. Спроектировать автоматизированную систему.
3. Реализовать автоматизированную систему.
4. Протестировать систему на предприятии.
5. Пройти обязательный нормоконтроль.

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Аналитический обзор современной литературы

В последние годы стремительное развитие искусственного интеллекта привело к появлению больших языковых моделей, которые стали важным шагом на пути к созданию интеллектуальных машин. Основанные на принципах моделирования человеческого мышления, эти технологии заложили основу для генеративного ИИ – системы, способной создавать связный текст, генерировать изображения и писать программный код, приближаясь по уровню к возможностям человека.

Компании по всему миру начали экспериментировать с новой технологией, полагая, что она может преобразовать СМИ, финансы, юриспруденцию и профессиональные услуги, а также государственные услуги. В основе технологии LLM лежит научная разработка, известная как модель-трансформер, созданная исследователями Google в 2017 году. Эта разработка стала прорывом в области обработки естественного языка, обеспечив глубокий анализ и точность генерации текстовой информации.

Модель-трансформер – это архитектура нейронной сети, разработанная для обработки последовательностей данных, таких как текст, изображения или аудио. Впервые она была представлена в статье «Attention is All You Need» и заменила всех своих предшественников, которые имели ограничения в обработке длинных последовательностей из-за проблем с долговременной памятью и вычислительной мощностью [2].

Трансформер используется для извлечения признаков из текстовых данных и создания определенного контекста. Он работает путем многократной обработки последовательности, называемой энкодером, и создания скрытых представлений для каждого элемента последовательности. Трансформер использует механизм внимания (attention mechanism, attention model) для вычисления весов, которые моделируют отношения между элементами

последовательности, и позволяют модели более эффективно использовать контекст. Он состоит из двух основных компонентов – энкодера и декодера (рисунок 1).

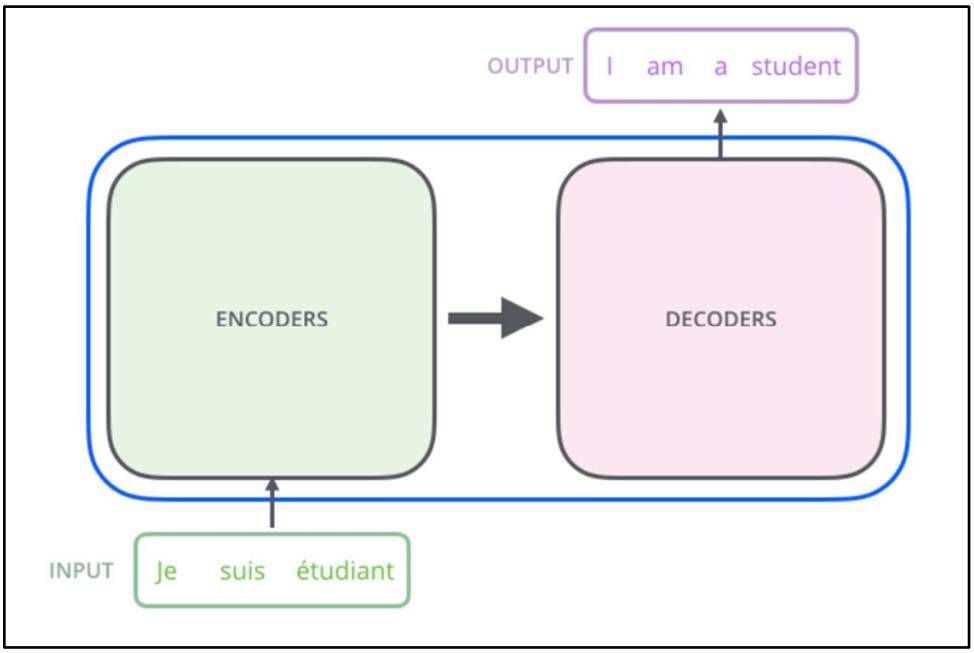


Рисунок 1 – Схема трансформера

Универсальный вычислительный механизм трансформера можно описать следующим образом: он принимает на вход один набор последовательностей (данных) и выдает на выходе тоже набор последовательностей, но уже преобразованный по некоему алгоритму.

Так как текст, картинки и звук можно представить в виде последовательностей чисел, то с помощью трансформера можно решать практически любые задачи. Главная особенность трансформера заключается в его удобстве и гибкости: он состоит из простых модулей-блоков, которые легко масштабировать. Если старые, дотрансформерные языковые модели начинали требовать слишком много ресурсов, когда их пытались заставить обработать быстро и много слов за один раз, то модели-трансформеры справляются с этой задачей гораздо экономнее.

Трансформер состоит из нескольких блоков, каждый из которых имеет свою структуру. Общая структура нейронной сети может быть описана следующим образом.

1. Входной слой, который принимает входные данные и векторизует их.
2. Предобработка, включающая в себя нормализацию и преобразование данных.
3. Энкодер, который разбивает входные данные на несколько последовательностей (например, токенов) и преобразует каждую последовательность в вектор фиксированной длины.
4. Декодер, который генерирует выходные данные на основе векторов, созданные энкодером.
5. Выходной слой, который генерирует финальный ответ.

На рисунке 2 представлены блоки экнодера и декодера, включающие в себя полносвязанные слои, слои само-внимания и многоканальные свертки. Взаимодействуя между собой эти блоки достигают высокой точности в задачах, связанных с обработки текста и непосредственной генерации ответов.

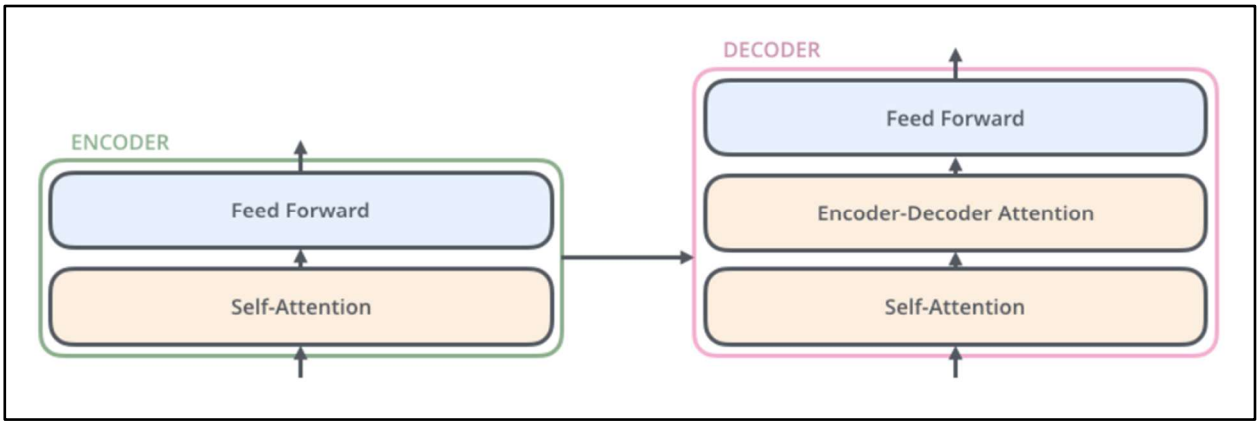


Рисунок 2 – Слои само-внимания в энкодере и декодере

Ключевым преимуществом трансформеры является то, что он обрабатывает сразу всю последовательность – будь то предложение, абзац или целая статья. Он анализирует все ее части, а не только отдельные слова или предложения. Это позволяет программному обеспечению лучше улавливать контекст

и закономерности, а также более точно работать с текстом, генерируя релевантные ответы. Благодаря такому подходу удается ускорить обучение большой языковой модели, что, в свою очередь, повышает ее эффективность и масштабируемость [3].

Большие языковые модели – это системы, использующие искусственный интеллект для обработки и генерации текстовой информации. Модели обучаются на огромных объемах данных и в следствии выполняют широкий спектр задач, связанный с созданием текстов, переводом, анализом и обработкой информации. LLM автоматически генерируют текст, который может быть использован для различных целей, включая создание контента, написание программного кода, а также для анализа и извлечения текстовой информации из больших массивов данных [4].

Однако использование общедоступных LLM моделей, расположенных на сторонних сервисах, невозможно в изолированной корпоративной среде из-за риска утечки информации. Стандартные решения, такие как Perplexity, Deepseek или ChatGPT, требуют доступа к облачным сервисам и обрабатывают данные на внешних серверах, что создает серьезные угрозы для предприятия, работающего с секретной информацией. Кроме того, общедоступные модели не всегда могут быть адаптированы под корпоративные нужны, поскольку не имеют возможности подключения к внутренней базе данных и работы без доступа к сети интернет. Для таких случаев требуется разработка собственных LLM решений, которые могут быть полностью изолированы от внешних сервисов, обеспечивая защиту данных и их локальную обработку. Благодаря этому появляется возможность адаптировать модель под конкретные нужды, обучить ее на внутреннем домене знаний предприятия и задавать контекст для выдачи релевантных ответов.

Учитывая вышеуказанные факторы, было приняло решение о внедрении автоматизированной системы поддержки на основе языковых моделей. В качестве решения рассматриваются как разработка собственной LLM, так и использование существующих открытых моделей, что позволяет выбрать

оптимальный баланс между качеством, безопасностью и затратами на внедрение. Модель будет полностью изолирована от внешних сервисов, что позволит гарантировать защиту корпоративной информации и эффективную обработку данных.

Одной из ключевых сложностей, связанных с большими языковыми моделями, является их склонность к генерации неточной или ложной информации, что приводит к так называемым «галлюцинациям» – созданию текстов, которые кажутся правдоподобными, но не соответствуют реальности. Это происходит из-за того, что LLM обучаются на огромных наборах данных, в которых информация тесно переплетена, и модель иногда выдает неподтверждённые сведения как достоверные.

В работе [5] описаны некоторые проблемы, с которыми могут столкнуться пользователи при использовании LLM. Люди нередко воспринимают браузеры и языковые модели как надежные источники информации, что увеличивает риск дезинформации. Языковая модель может сгенерировать ответ, убеждая собеседника в ее достоверности, даже если она искажает факты или события. Одна из причин выдачи некорректных данных заключается в использовании недостоверной информации, полученной из интернета, которая интерпретируется моделью как достоверная. Более того, модели ограничены объемом и качеством данных, на которых они были обучены, что может привести к поверхностным ответам или недостаточной глубине знаний.

Эти ограничения подчеркивают необходимость ответственного использования больших языковых моделей и требуют внедрения механизмов, повышающих их надежность. Одним из решений может быть внедрение функций отслеживания источников, которая позволяет пользователям проверять происхождение информации, генерируемой моделью. Это не только повышает доверия к результатам, но и помогает пользователям самостоятельно оценивать достоверность сгенерированных данных.

Также важным аспектом является разработка систем, которые могут компенсировать ограничения LLM. Интеграция с актуальными базами знаний,

использование механизмов дополнительной валидации и возможность запрашивать подтверждения или уточнения от модели могут значительно снизить риски дезинформации. Также стоит обратить внимание на обучение моделей умению признавать ограниченность своих знаний и предлагать пользователям дополнительные ресурсы для самостоятельного поиска информации. Таким образом, комбинируя технические улучшения с образовательными мерами, можно существенно повысить качество и безопасность использования LLM.

## 1.2. Механизмы повышающие релевантность информации

Точная настройка модели (fine-tuning) – это процесс, который используется для дальнейшего обучения уже предварительно обученной большой языковой модели на меньшем по размеру, специализированном, размеченном наборе данных. Происходит корректировка некоторых параметров модели, чтобы оптимизировать ее работу для конкретной задачи или набора задач. При полной тонкой настройке обновляются все параметры модели, что делает ее похожей на предварительное обучение, только на размеченном и гораздо меньшем наборе данных. Но такой подход требует огромных вычислительных мощностей и большого объема памяти, поскольку LLM-модели содержат десятки или сотни миллиардов параметров, а также может потребоваться специализированное оборудование [6].

LoRA (Low-Rank Adaptation) – процедура тонкой настройки набора данных заключающаяся в добавлении нескольких адаптивных слоев к базовой предварительно обученной модели, а затем в замораживании весов предварительно обученной модели и обновлении только весов адаптивных слоев. В языковых моделях, основанных на архитектуре трансформера, добавляются эти адаптивные слои к слоям трансформера и настраиваются. При таком подходе возникают некоторые проблемы, связанные с затратами на обучение и задержкой вывода. Кроме того, во время вывода эти дополнительные слои увеличивают время вывода [7].

Retrieval Augmented Generation – подход, избавляющий большие языковые модели от галлюцинаций и недостоверных фактов. Однако у больших языковых моделей есть проблема. В процессе обучения нейросети информация, которая связана со структурой языка, очень сильно перемешивается с фактами, которые она из этого текста выбирает. Нейросеть может что-то выдавать за факты и изложить как некую логическую историю безосновательно. RAG же достаточно жестко задает контекст в виде фрагментов текста, на базе которых LLM должна скомпоновать ответ, то есть нейросеть используют непосредственно в момент генерации. Также RAG использует LLM для извлечения информации из цепочек связанных документов путем интеллектуального анализа, а не разметки страниц [8].

Из всех методов RAG является наиболее подходящим, так как позволяет LLM динамически извлекать актуальную информацию в момент генерации ответа. В отличие от fine-tuning, который требует больших вычислительных ресурсов, и LoRA, увеличивающего задержку вывода, RAG снижает вероятность устаревших или недостоверных данных, обеспечивая точные и обоснованные ответы. Существуют различные модификации RAG, но основные шаги подхода одинаковы (рисунок 3).

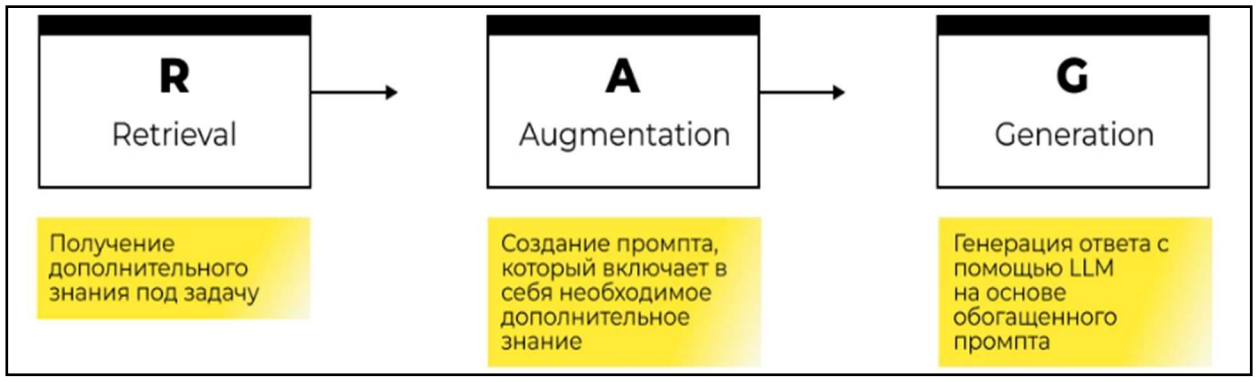


Рисунок 3 – Основные подходы RAG

На основе пользовательского запроса осуществляется обращение к заранее определенным источникам данных. Подход RAG позволяет интегрировать запросы к внутренним базам данных, документам и доменам знаний, чтобы обеспечить генерацию более достоверных и обоснованных ответов. Одним из

ключевых преимуществ RAG является его высокая скорость работы и оперативность. Модель не только использует собственные знания, но и моментально обращается к актуальной информации, что позволяет быстро выдавать релевантные и точные ответы. Результатом этого этапа является получение дополнительных данных, которые обогащают базовую информацию, с которой работает большая языковая модель при формировании ответа на вопрос.

Далее формируется промпт, который включает исходный пользовательский запрос и дополнительную информацию, полученную на предыдущем этапе. Результаты поиска могут быть изменены для создания наиболее точного обогащенного промпта, например, с помощью фильтрации или дополнительного ранжирования. Все результаты должны быть преобразованы в текст для использования в промпте.

На следующем этапе обогащенный промпт передается в большую языковую модель. Ответ может быть дополнен цитатами или ссылками на источники дополнительной информации, использованные в обогащенном промпте. Также возможно осуществление дополнительной проверки ответа модели. На основе результатов этого шага формируется окончательный ответ для пользователя [9].

RAG также сочетает генеративные возможности языковых моделей с дополнительным поиском релевантных данных в базах знаний или других источниках, что позволяет моделям запрашивать и использовать более точную и актуальную информацию для ответа. Поисковая генерация помогает снизить риск дезинформации, дополняя ответы контекстом и деталями из проверенных источников, что особенно важно для предоставления пользователям надежной информации. Кроме того, RAG эффективно решает проблемы обновления данных и специализированного контекста, позволяя моделям обращаться к узкоспециализированным базам данных и доменам знаний. Такой гибридный подход, интегрирующий поиск и генерация, позволяет создавать более точные и полезные приложения для пользователя, минимизируя риск дезинформации.

## 1.3. Подходы к поиску информации

В ходе работы были рассмотрены различные подходы к поиску информации, применяемые в современных системах. Рассмотрим некоторые из них.

**Поиск по ключевым словам**

Наиболее простой метод поиска документов, релевантных запросу пользователя – это поиск по ключевым словам, также известный как полнотекстовый поиск. Он основывается на сопоставлении слов, введённых пользователем, с терминами, содержащимися в тексте документов. Этот подход, несмотря на свою давнюю историю, остаётся актуальным и сегодня. Он особенно эффективен для поиска таких данных, как идентификаторы пользователей, коды продуктов, адреса или любая другая информация, требующая точного соответствия тексту запроса. Общая схема системы, использующей поиск по ключевым словам, представлена на рисунке 4.

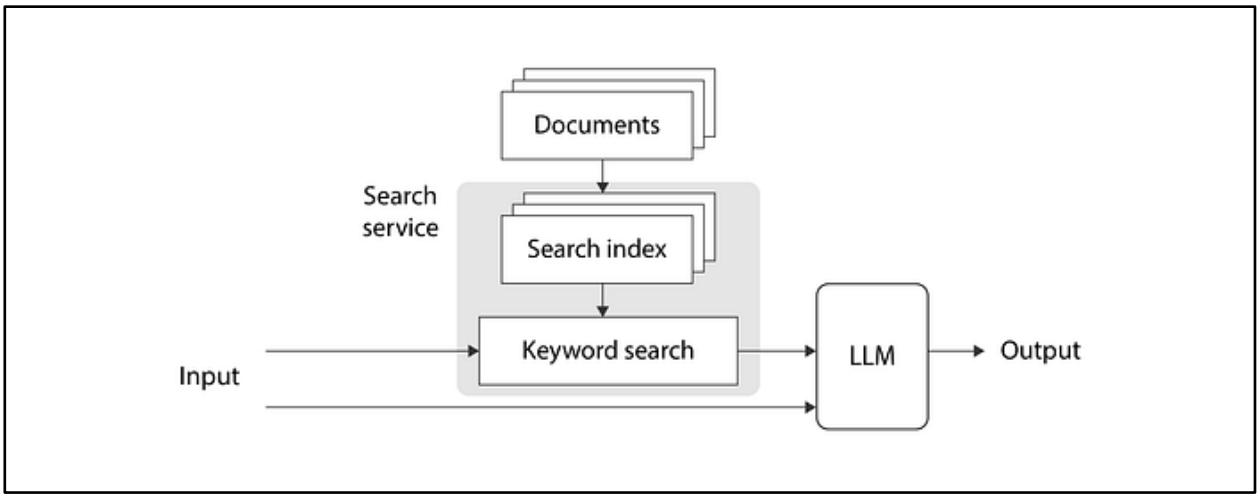


Рисунок 4 – Система с использованием поиска по ключевым словам

При таком подходе поисковый сервис поддерживает работу инвертированного индекса, хранящего сведения о соответствии слов и документов, в которых используются эти слова. Текстовые данные, введенные пользователем, разбирают, извлекая из них ключевые слова. Эти слова анализируют, находя их базовые формы. Затем выполняют поиск по инвертированному индексу, каждому совпадению назначают оценку, а после этого наиболее релевантные

документы, соответствующие запросу, возвращают из поискового сервиса [10].

**Векторный поиск**

Одним из отличительных аспектов является то, что при векторном поиске возможно нахождение соответствия поисковому запросу в документах, в которых нет ключевых слов из запроса, но общий смысл которых близок к смыслу запроса. Векторный поиск лучше всего показывает себя в тех случаях, когда в неструктурированном тексте ищут некие общие идеи, а не точные ключевые слова. Обзор RAG-системы, в которой используется векторный поиск представлен на рисунке 5.

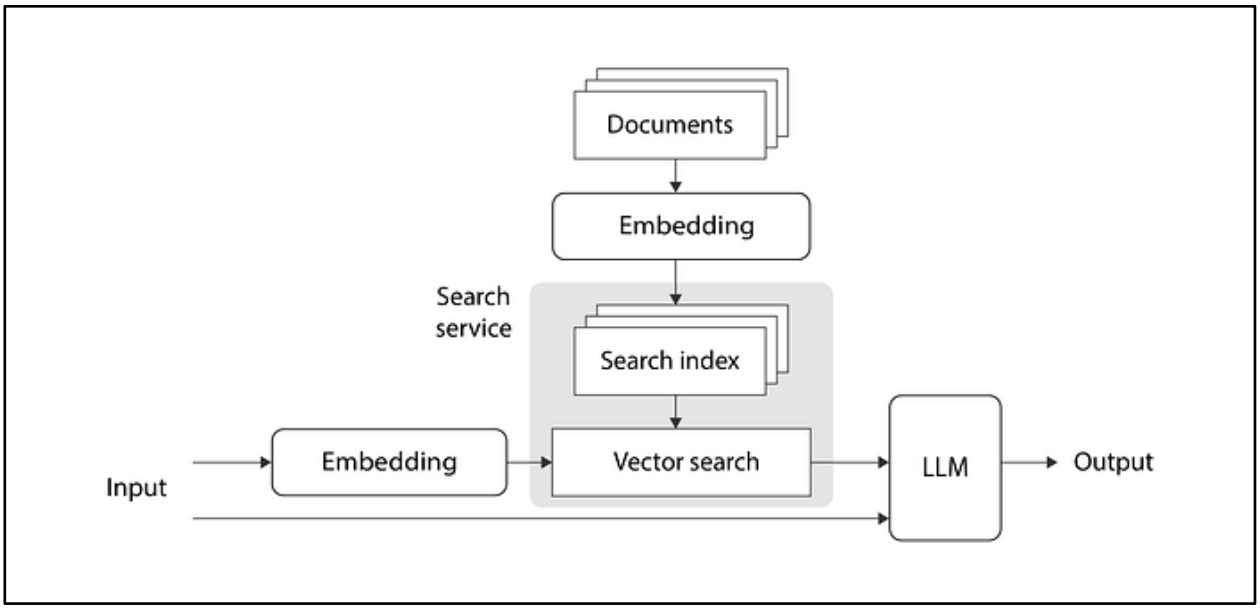


Рисунок 5 – Система с использованием векторного поиска

Чтобы лучше понять, как работает векторный поиск, рассмотрим три запроса пользователя, каждый из которых представлен в виде вектора.

1. Первый запрос: A = (0, 1). Текст запроса: «Посоветуйте просторное жилье, расположенное близко к морю»

2. Второй запрос: B = (0.12, 0.99). Текст запроса: «Дом площадью 150 квадратных метров с видом на океан».

3. Третий запрос: C = (0.96, 0.26). Текст запроса: «Что приготовить на завтрак».

На рисунке 6 представлен график на котором расположены вектора, соответствующие данным запросам.

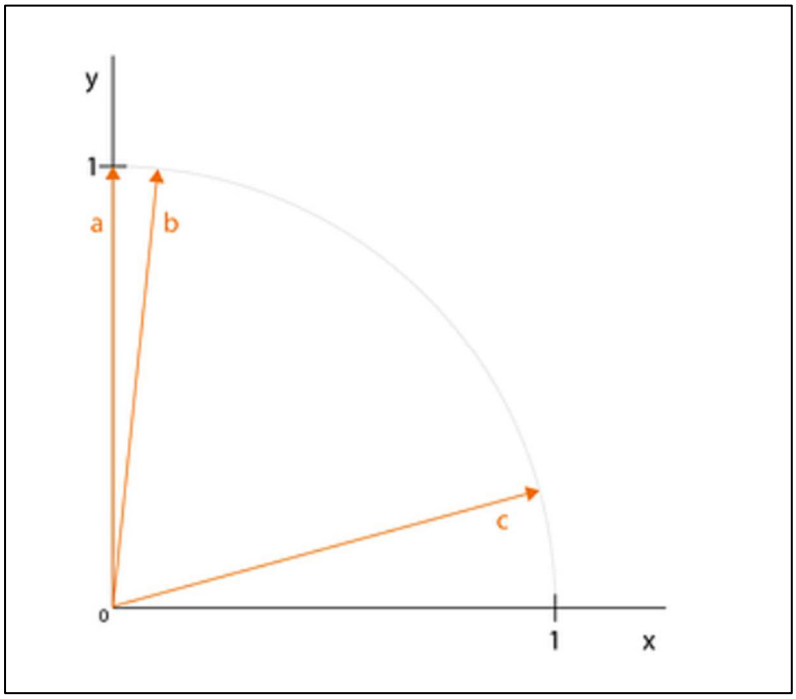


Рисунок 6 – Графическое представление векторов

Вектор C, соответствующий запросу «Что приготовить на завтрак», находится значительно дальше от векторов A и B на данном графике. Это значит, что он имеет совершенно другую тематику и смысл. В то время как векторы A и B сосредоточены на вопросах жилья. Вектор C имеет другой смысл, что делает его не релевантным к запросам пользователя о недвижимости [11].

**Гибридный поиск**

Данный подход сочетает в себе два метода: поиск по ключевым словам и векторный поиск, что позволяет получать более точные и релевантные результаты. Этот подход сначала фильтрует данные по ключевым словам, а затем применяет векторный анализ для глубокого понимания оставшихся данных. Гибридный поиск активно используется в различных сферах, улучшая пользовательский опыт за счет предоставления более качественных результатов. В гибридном поиске результаты от ключевого и векторного методов объединяются в один список. Чтобы правильно понять такие результаты, нужно учитывать, как они оцениваются. Векторный поиск использует специальный

показатель для определения близости, а текстовый поиск – оценку для работы с ключевыми словами. Эти оценки разные и напрямую не сравниваются. Чтобы их можно было сопоставить, оценки из векторного поиска преобразуются в специальный семантический показатель. Его значения лежат в диапазоне от 100 (лучший результат) до 0 (худший результат). Это делает оба подхода совместимыми, что помогает ранжировать результаты точнее и удобнее [12].

В [13] раскрывается важность разработки автоматизированных систем поддержки пользователей, где человеко-компьютерное взаимодействие (ЧКВ) играет важную роль, помогая строить интерфейсы, способные оптимизировать пользовательский опыт, особенно при решении задач технической или информационной поддержки. К ключевым аспектам систем поддержки пользователя относится проектирование пользовательского интерфейса, который должен быть легким в освоении и интуитивно понятным для взаимодействия с автоматизированной системой. Пользовательский интерфейс системы должен быть удобным и отзывчивым, чтобы минимизировать время и усилия пользователя при решении вопросов.

Следующим аспектом является адаптация под пользователя, суть которой заключается в том, что автоматизированные системы поддержки должны учитывать потребности различных пользователей, предоставляя доступ к информации и инструментам, которые соответствуют их уровню подготовки и контексту использования. Это позволяет создать более гибкие и человеко-ориентированные решения.

Автоматизированные системы поддержки выполняют множество функций, направленных на упрощение и улучшение взаимодействия с продуктами компании. Основная задача таких систем – обработка запросов от пользователей и предоставление информации в режиме реального времени. Эффективность поддержки во многом зависит от интеллектуального поиска информации: системы с использованием RAG могут находить и подбирать актуальные данные из базы знаний, что позволяет пользователям оперативно получать

нужные сведения. Кроме того, современные автоматизированные системы используют LLM для персонализации ответов, адаптируя их под стиль общения и предпочтения конкретного пользователя, что делает взаимодействие более естественным. Благодаря применению технологий обработки естественного языка такие системы способны распознавать контекст, обрабатывать синонимы, тональности и вариации запросов, что значительно повышает качество обслуживания и восприятие информации [14].

## 1.4. Обзор аналогов

В процессе исследования были проанализированы несколько современных решений, аналогичных разрабатываемой автоматизированной системе поддержки пользователей с применением LLM и RAG. Рассмотрим некоторые из них.

**Виртуальный ассистент Макс**

«Госуслуги» – это система электронных услуг, предоставляемых государственными органами для граждан и организаций [15]. В рамках этой системы используется чат-поддержка, основанная на LLM, которая помогает пользователям быстро находить нужную информацию о государственных услугах, процессах подачи заявок и другой информации (рисунок 7).

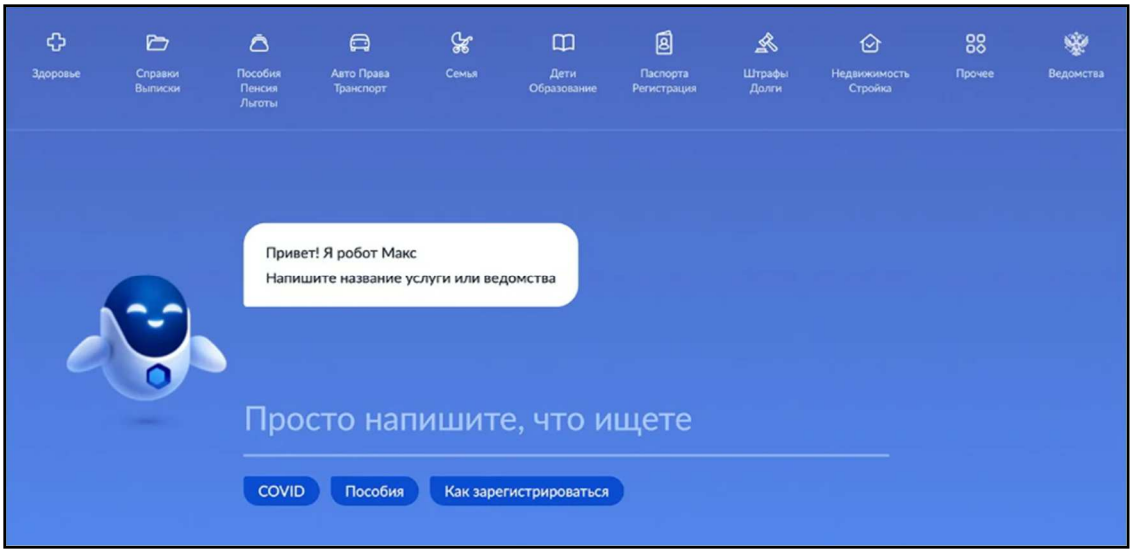


Рисунок 7 – Виртуальный ассистент сайта «Госуслуги»

Основное преимущество чата поддержки в «Госуслугах» – это интеграция с актуальными данными из государственных реестров и баз данных. Система обеспечивает доступ к информации о гражданских и юридических услугах в реальном времени, позволяя быстро получить помощь по вопросам, связанным с налогами, социальной защитой и другими государственными услугами. Ассистент также помогает в процессе подачи заявлений и предоставляет актуальные инструкции.

Из недостатков можно выделить ограниченные возможности для кастомизации и настройки под конкретные запросы пользователя, а также необходимость наличия определенной структуры данных в базе Госуслуг, что ограничивает гибкость в решении нестандартных запросов.

**Виртуальный ассистент Салют**

«Сбербанк» – ведущий российский банк, активно применяющий виртуальных ассистентов на основе больших языковых моделей (LLM) для обслуживания клиентов [16]. Эти ассистенты предоставляют консультации по банковским продуктам, включая кредиты, депозиты, переводы и другие операции. Благодаря LLM ассистенты обеспечивают оперативные и точные ответы, адаптированные к потребностям пользователей. Интерфейс ассистента представлен на рисунке 8.

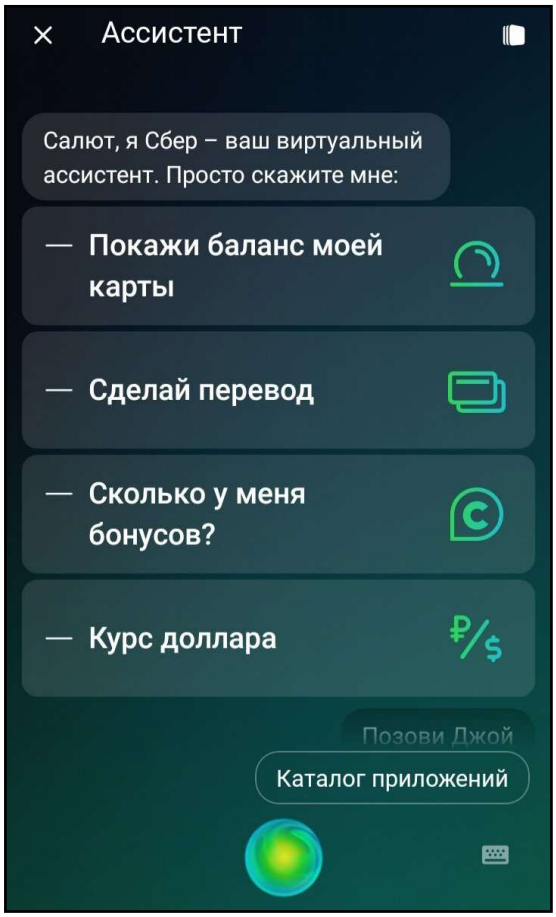


Рисунок 8 – Виртуальный ассистент банка «Сбербанк»

Главное преимущество ассистента в «Сбербанк» заключается в интеграции с внутренними сервисами банка и возможностью работы с личными данными клиента, что позволяет предлагать персональные решения. Также он обрабатывает множество запросов и операций, включая переводы, запросы баланса и блокировку карт. Виртуальный ассистент работает с актуальными данными в реальном времени, что делает его удобным для клиентов банка.

Из недостатков стоит отметить, что настройка и интеграция с корпоративными сервисами требует значительных усилий, а также ограниченную функциональность при более сложных и специфичных запросах, которые требуют вмешательства специалистов.

**Виртуальный ассистент Алиса**

Алиса – это виртуальный ассистент, разработанный компанией Яндекс он также использует LLM и технологию RAG для улучшения качества поиска

информации [17]. Виртуальный ассистент использует алгоритмы обработки естественного языка для предоставления точных и обоснованных ответов на вопросы. С помощью технологии RAG он интегрирует данные из поисковой системы Яндекс, что позволяет предоставлять актуальную информацию, основанную на последних данных из интернета, в том числе новости, статьи и прочее (рисунок 9).

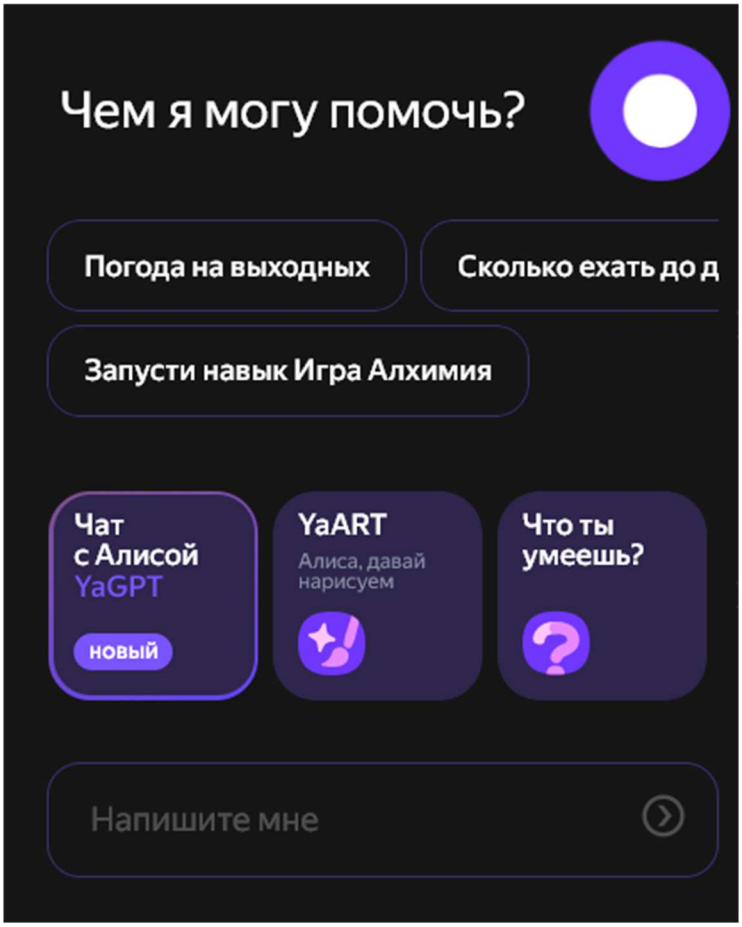


Рисунок 9 – Виртуальный ассистент от «Яндекс»

Основным преимуществом Алисы является возможность работы с актуальной информацией из открытых источников. Однако при таком подходе недостатком является зависимость от качества данных, которые доступны в поисковой системе, что может повлиять на точность ответов.

Результаты сравнения функциональных возможностей виртуальных ассистентов представлены в таблице 1.

Таблица 1 – Сравнение функций виртуальных ассистентов

| Название | Формат данных | Настройка подзадачи | Подключение к базам данных | Поддержка RAG | Основные недостатки |
|---|---|---|---|---|---|
| Виртуальный ассистент Макс | Текст, данные | - | + | Частично | Ограниченная кастомизация, специфическая структура данных |
| Виртуальный ассистент Салют | Текст, банковские данные | + | + | + | Сложность настройки, ограничения на сложные запросы |
| Виртуальный ассистент Алиса | Текст, интернет-данные | + | + | + | Зависимость от качества данных Яндекса |

Обзор этих решений показал, что все перечисленные системы предлагают ряд преимуществ, включая интеграцию с актуальными данными и базами данных, а также поддержку различных функциональных возможностей для обработки текстовых запросов. Однако, несмотря на эти достоинства, каждая из этих систем имеет свои ограничения. Основными недостатками являются зависимость от подключения к внешней сети, что создает проблемы с безопасностью и конфиденциальностью данных, а также невозможность использования доменов знаний и документов с грифом секретности. Поэтому необходимо разработать собственный интерфейс для системы поддержки пользователя, работающий только в рамках предприятия.

## 1.5. Анализ технологий для разработки серверной части

Серверная часть системы поддержки пользователей является ключевым элементом, обеспечивающим обработку запросов, интеграцию с RAG, LLM, управление данными и выполнение асинхронных задач. Ниже подробно описаны выбранные технологии бэкенда, их назначение, преимущества и роль в реализации системы.

**FastAPI**

FastAPI [18] – это современный фреймворк, созданный для Python, который ориентирован на разработку высокопроизводительных API (Application Programming Interface). Фреймворк был создан с учётом возможностей асинхронного программирования для эффективного распределения ресурсов при одновременной обработке множества запросов. Основной упор в архитектуре делается на использовании аннотаций типов Python, что способствует созданию устойчивых и понятных программных решений.

Интеграция FastAPI с современными библиотеками для валидации данных и сериализации позволяет значительно сократить время на написание кода, минимизируя количество ошибок и упрощая сопровождение проектов. Такой подход способствует быстрому переходу от идеи к рабочему прототипу при сохранении высокого уровня масштабируемости разрабатываемых решений.

**WebSocket**

WebSocket [19] – технология, позволяющая обмениваться данными между клиентом и сервером в режиме реального времени. Она создает постоянное соединение для отправки сообщений по одному каналу связи. Основным отличием от протокола HTTP является то, что данные можно отправлять в обе стороны, а не только от клиента к серверу и наоборот.

Первая версия протокола появилась в 2009 году. Уже в 2011 году он получил статус RFC (Request for Comments, рабочее предложение), то есть был признан как стандарт, широко применяемый во всемирной паутине. Сегодня технология поддержки веб-сокетов используется во всех мобильных, десктопных и серверных устройствах. Также существует WSS (WebSockets Secure) – это протокол для обмена данными между веб-сервером и веб-браузером с использованием безопасного соединения, обычно через порт 443 (который также используется для HTTPS). WSS предоставляет безопасное и надежное соединение для двусторонней связи между клиентом и сервером в реальном

времени, что делает его полезным для различных веб-приложений, таких как чаты, онлайн-игры и многие другие.

**RabbitMQ**

RabbitMQ [20] – технология, появившаяся в 2007 году и написанная на языке программирования Erlang, которая представляет собой брокер сообщений. Этот инструмент отлично подходит для асинхронной обработки задач, что помогает серверной части лучше справляться с нагрузками. Одним из преимуществ RabbitMQ является поддержка различных протоколов. Это означает, что он может работать с множеством систем и решать разнообразные задачи.

Одним из ключевых преимуществ RabbitMQ является его надежность. Система использует механизм подтверждения, благодаря которому отправить может быть уверен, что сообщение действительно дошло до получателя. Если сообщение не доходит, то RabbitMQ повторно его отправляет, пока оно не будет успешно доставлено.

Кроме того, RabbitMQ использует пуш-модель для доставки сообщений. Это означает, что сервер может отправлять информацию клиентам, не дожидаясь, пока те сделают запрос. Это особенно удобно, когда нужно быстро уведомить пользователей о важных событиях или изменениях в системе. Таким образом RabbitMQ обеспечивает не только надежную, но и оперативную связь между сервером и клиентом, что делает его ценным инструментом для разработчиков.

**MinIO**

MinIO [21] представляет собой высокопроизводительную систему объектного хранения, разработанную для обеспечения надежного и безопасного управления данными в виде файлов, таких как техническая документация, инструкции или иные материалы. Данная технология реализует механизмы шифрования данных на уровне объекта, что обеспечивает защиту конфиденциальной информации от несанкционированного доступа, а также поддерживает

функционирование в изолированных сетевых средах, что соответствует строгим требованиям безопасности, характерным для корпоративных систем.

В рамках системы MinIO выполняет функцию предоставления доступа к хранимым файлам, которые могут быть извлечены для последующего анализа и использования при формировании ответов на пользовательские запросы. Например, при обращении за технической документацией MinIO обеспечивает оперативный поиск и предоставление соответствующего файла.

## 1.6. Анализ технологий для разработки пользовательской части

Для того, чтобы создать удобный и функциональный интерфейс системы поддержки пользователей, нужно выбрать подходящие технологии, которые обеспечивают гибкость и стабильность. Ниже рассмотрим ключевые веб-технологии, используемые для разработки интерфейса.

HTML5 (HyperText Markup Language) – это язык гипертекстовой разметки, который используется для создания веб-страниц. С его помощью можно формировать структурированные и визуально привлекательные документы и отображать различные элементы, такие как заголовки, абзацы, списки и так далее. HTML задает порядок этих элементов, что помогает пользователям лучше воспринимать информацию. Также HTML включает новые возможности, такие как семантические теги, которые упрощаете структуру документа и его индексацию поисковыми системами. Кроме того поддерживаются мультимедийные элементы без дополнительных плагинов, и предоставляется API для создания сложных веб-приложений, что делает его полезным инструментом для разработчиков.

CSS (Cascading Style Sheets, каскадные таблицы стилей) – язык, который помогает задавать внешний вид HTML-документов. Это одна из основных технологий для стилизации сайтов. CSS работает вместе с HTML и взаимодействует с браузерами, чтобы сделать документы визуально более привлекательными. Последняя версия, CSS3, добавляет новые возможности, такие как

адаптивный дизайн, анимации и различные эффекты. Это позволяет создавать современные интерфейсы, которые могут быть адаптированы для разных устройств и экранов [22].

Одной из важных технологий разработки стали веб-компоненты. Это удобный набор решений, который позволяет создавать универсальные и настраиваемые элементы. Их главное преимущество заключается в том, что они могут работать отдельно от остального кода, что делает их использование в веб-приложениях простым и безопасным.

Далее рассмотрим, из чего состоят веб-компоненты. Они представляют собой три основные технологии, которые можно комбинировать, чтобы разрабатывать удобные и переиспользуемые элементы интерфейса.

Пользовательские элементы – это набор API-интерфейсов JavaScript, который позволяет создавать пользовательские элементы и задавать им уникальное поведение. Эти элементы делают код модульным и упрощают создание сложных приложений. С помощью API разработчики могут создавать новые теги или модифицировать уже существующие, прописывая их структуру, стили и поведение через JavaScript. В сочетании с другими технологиями, такими как Shadow DOM и HTML-шаблоны, пользовательские элементы предлагают возможность создавать изолированные и переиспользуемые компоненты, повышая читабельность кода и облегчая его поддержку.

Shadow DOM – это тоже набор API-интерфейсов JavaScript, которое позволяет создавать «теневое» дерево DOM. Оно отображается отдельно и управляется отдельно от основного документа. Это значит, что можно изолировать функции элементов и прописывать стили, не беспокоясь о конфликте с другими частями документа. Shadow DOM позволяет разрабатывать компоненты, которые не будут конфликтовать с глобальными стилями и скриптами страницы. Это важно, особенно когда нужно убедиться, что внутренние структуры и поведения работают независимо. Кроме того, стили, которые находятся в Shadow DOM, не влияют на элементы за его пределами, делая разработку модульных компонентов легкой задачей.

HTML-шаблоны – это элементы <template> и <slot> которые дают возможность создавать разметку, которая не отображается на странице, пока ее явно не вызовут. Такие шаблоны можно использовать многократно для создания структуры пользовательских элементов, что является весьма удобным инструментом. Элемент <template> хранит HTML-код, который остается невидимым на странице, пока его не инициализируют через JavaScript. Это предоставляет возможность программистам заранее определять структуру компонентов, не вмешиваясь в основную структуру DOM. Элемент <slot> работает вместе с Shadow DOM и создает области, куда можно подставлять содержимое из главного документа. Это делает шаблоны динамичными и гибкими. Все это позволяет легко управлять содержимым и помогает создавать классные интерфейсы.

Веб-компоненты сочетают в себе пользовательские элементы, Shadow DOM и HTML-шаблоны, что предоставляет разработчикам универсальные инструменты для создания модульных и изолированных компонентов, которые можно использовать повторно. Это делает разработку современных веб-приложений проще и улучшает их поддержку и масштабирование [23].

## 1.7. Анализ векторных баз данных

Для определения наиболее подходящей векторной базы данных для интеграции с MinIO рассмотрим и сравним доступные решения.

**ChromaDB**

ChromaDB [24] – это векторная база данных, ориентированная на хранение и быстрый поиск векторных представлений. Она необходима для решения задач в различных приложениях ИИ, таких как семантический поиск и обработка естественного языка, для работы со сложными данными, такими как текст и изображения. ChromaDB позволяет разрабатывать адаптивные приложения, требующие векторного хранения данных. Преимуществом данной базы данных является ее масштабируемость, она способна поддерживает приложения любого размера и обрабатывая обширные наборы данных, необходимые

для приложений на основе ИИ. Также ChromaDB хорошо оптимизирована, что отлично подходит для систем в которых важны быстрый поиск и обработка векторных представлений.

**Weaviate**

Weaviate [25] – это векторная поисковая система с низкой задержкой и встроенной поддержкой различных типов мультимедиа. Она предназначена для семантического поиска, извлечения вопросов и ответов, классификации. Weaviate может хранить как объекты, так и векторы, что позволяет сочетать векторный поиск со структурированной фильтрацией и отказоустойчивостью облачной базы данных. Данная поисковая система позволяет легко использовать самые современные модели искусственного интеллекта, обеспечивая при этом масштабируемость, простоту использования, безопасность и экономичность. Также Weaviate реализована таким образом, что отсутствует необходимость хранения очень большие наборы данных в оперативной памяти. В то же время доступная память может использоваться для повышения скорости запросов. Это позволяет осознанно выбирать компромисс между скоростью и затратами для каждого конкретного случая.

**Qdrant**

Qdrant [26] – это векторная база данных с открытым кодом, разработанная на языке программирования Rust. Она предоставляет удобный API, который позволяет хранить, искать и управлять векторными представлениями с метаданными, известными как полезная нагрузка. Эти нагрузки помогают повысить точность поиска и предоставлять пользователям релевантную информацию. Qdrant является надежной и быстрой базой данных и позволяет работать с векторными представлениями больших объемов. Она поддерживает работу с такими языками программирования, как Python, TypeScript, JavaScript, Rust и Go, что делает ее универсальным инструментов для разработчиков разного профиля. Также Qdrant использует граф для векторной индексации и предлагает разные метрики расстояний, что позволяет адаптировать систему под различные сценарии поиска.

В таблице 2 представлены преимущества и недостатки рассмотренных баз данных.

Таблица 2 – Сравнение векторных баз данных

| **База данных** | **Преимущества** | **Недостатки** |
|---|---|---|
| ChromaDB | Простота установки, интеграция с Python, фильтрация по метаданным | Отсутствует поддержки кластера, слаборазвитая экосистема |
| Weaviate | Поддержка GraphQL, внешние embedding, фильтрация | Требует инфраструктуры, может быть сложной в настройке |
| Qdrant | Быстрое развертывание, гибкий поиск | Требует настройки, API-ориентирован |

В результате проведения анализа было принято решение использовать базу данных ChromeDB. Она сочетает в себе простоту подключения, возможность хранения метаданных и высокую скорость разработки. Хотя у нее есть некоторые ограничения в масштабировании, ChromeDB вполне отвечает нуждам системы и позволяет реализовать нужный функционал без сложной инфраструктуры.

**Вывод по первой главе**

В первой главе был проведен анализ современной литературы, в ходе которого рассмотрены проблемы использования больших языковых моделей в системах поддержки пользователей, а также методы повышения релевантности информации. Проведен обзор существующих решений, определены их преимущества и недостатки. Особое внимание уделено технологиям для разработки пользовательских интерфейсов и серверной части. Также были проанализированы векторные базы данных, на основе чего выбрана подходящая.

# 2. ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ

ВНИИТФ предъявляет особые требования к внутреннему программному обеспечению. В связи с этим на предприятии было оформлено официальное техническое задание на разработку автоматизированной системы, зарегистрированное под архивным номером 030-14/264 от 07.02.2025 г. Полный текст документа не подлежит публичному раскрытию, поскольку хранится в архиве предприятия. В связи с этим в приложении А представлена разрешенная к публикации часть документа, содержащая ключевые положения, которые служат обязательным руководством для разработки подсистемы.

## 2.1. Функциональные требования

Функциональные и нефункциональные требования используются в процессе разработки программного обеспечения для определения того, каким должно быть окончательное решение. Функциональные требования описывают, что должно делать программное обеспечение, то есть какие функции и возможности оно должно иметь. Нефункциональные требования описывают, как программное обеспечение должно работать, то есть какие должны быть характеристики качества.

В процессе проектирования системы были определены следующие функциональные требования к разрабатываемому решению на основе технического задания (приложение А).

1. Система должна позволять пользователю отправлять текстовые запросы через поле ввода веб-компонента.
2. Система должна отображать ответы сервера в диалоговом окне.
3. Система должна сохранять историю переписки.
4. Система должна открываться и сворачиваться по нажатию кнопки, сохраняя состояние при повторном открытии.
5. Система должна легко интегрироваться в веб-сайты и порталы.

## 2.2. Нефункциональные требования

Помимо функциональных, был согласован список нефункциональных требований.

1. Система должна реализовывать REST API с использованием языка программирования Python и фреймворка FastAPI.
2. Система должна быть разработана на языке программирования JavaScript с применением HTML и CSS.
3. Система должна устанавливать WebSocket-соединение с сервером при инициализации.
4. Система должна отправлять запросы пользователя на сервер в формате JSON.
5. Система должна иметь возможность размещения внутри контейнерной инфраструктуры Docker.
6. Система должна быть нативным веб-компонентом.
7. Система должна размещать домен знаний в векторной базе данных ChromeDB.

## 2.3. Диаграмма вариантов использования

Диаграмма вариантов использования [27] – это тип диаграммы на языке унифицированного моделирования (UML), которая представляет взаимодействие между субъектами (пользователями или внешними системами) и рассматриваемой системой для достижения конкретных целей, иллюстрируя функциональность системы и способы её использования. Она включает следующие элементы: актеры, изображаемые в виде фигурок человечков, которые представляют внешние сущности (например, пользователь или другая система); сценарии использования, обозначенные овалами, описывающие конкретные действия системы (например, «Оформление заказа»); границу системы в виде прямоугольника, отделяющую внутренние компоненты системы от внешних сущностей; а также взаимосвязи, такие как ассоциации (линии между актерами и сценариями), включение (include) и расширение (extend),

показывающие зависимости между сценариями. Кроме того, диаграмма может содержать примечания (notations), обеспечивающие разъяснение дополнительных деталей или ограничений функциональности, а также пакеты (packages), которые позволяют структурировать взаимосвязанные сценарии, повышая читаемость и организацию модели. Данное расширение функционала делает диаграмму вариантов использования эффективным инструментом для систематического анализа требований

UML-диаграмма на рисунке 10 описывает модель взаимодействия «Пользователь» с автоматизированной системой.

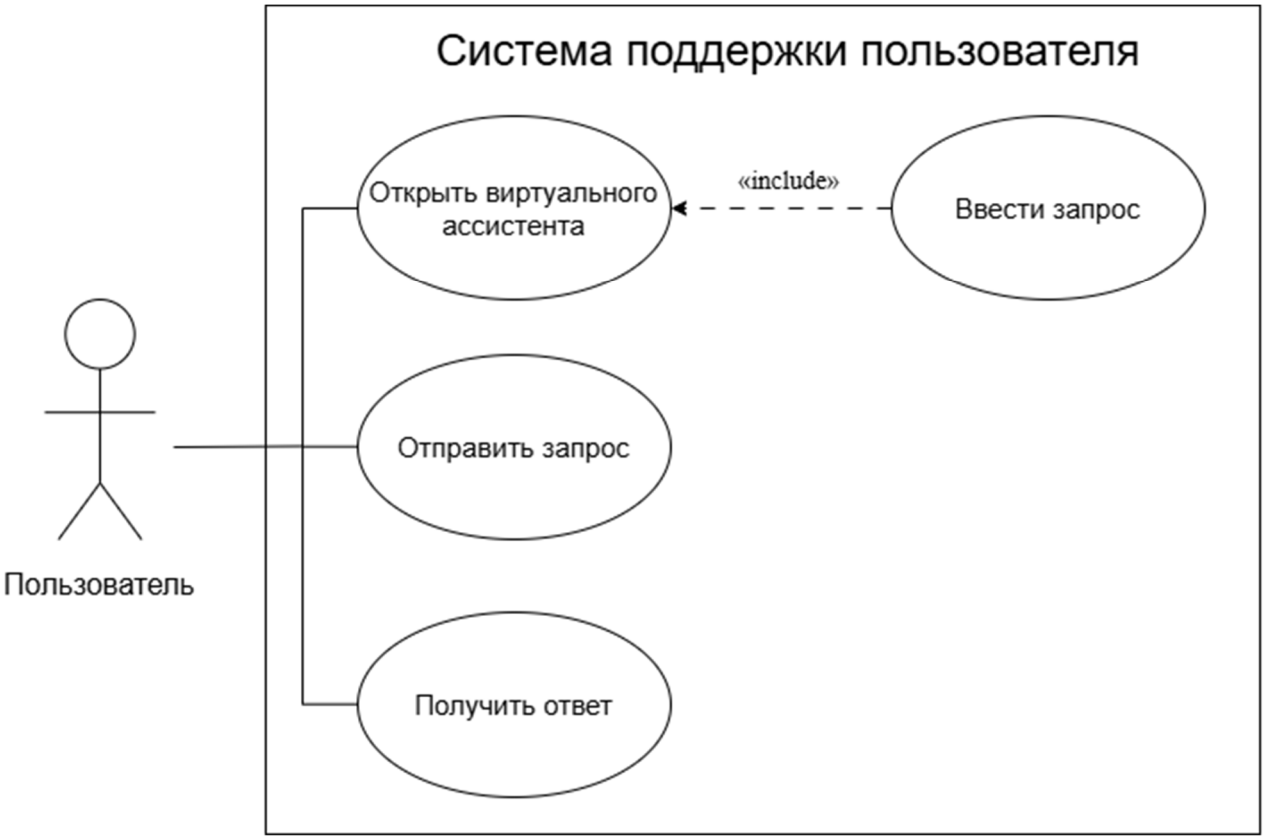


Рисунок 10 – UML-диаграмма вариантов использования

«Пользователь» может инициировать несколько ключевых сценариев взаимодействия с системой. В первую очередь, пользователь открывает виртуального ассистента, который служит основным интерфейсом взаимодействия с интеллектуальной системой. Данный процесс включает в себя ввод запроса, что представлено в диаграмме через связь include с вариантом использования «Ввести запрос».

Далее пользователь имеет возможность отправить запрос, который обрабатывается посредством интегрированной архитектуры, объединяющей LLM и компонент RAG.

Завершающим этапом сценария является получение пользователем ответа, сгенерированного системой поддержки.

## 2.4. Проектирование архитектуры автоматизированной системы

Разработка автоматизированной системы основывается на архитектуре C4, которая позволяет представить систему на разных уровнях абстракции. Модель C4 – это методология для визуализации и документирования архитектуры программного обеспечения, разработанная Саймоном Брауном [28]. Название C4 происходит от слов Context, Containers, Components, Code, которые отражают четыре уровня абстракции, на которых строятся диаграммы для описания системы. Простая и нестрогая нотация позволяет одинаково легко читать и моделировать диаграммы, а разные уровни абстракции подойдут как для бизнес-пользователей, так и для технических специалистов.

**Диаграмма контекста (Context Diagram)**

На этом уровне представляется вся система в ее внешнем контексте. Диаграмма контекста показывает, как система взаимодействует с другими системами и пользователями. Этот уровень не углубляется в детали реализации, а сосредоточен на том, чтобы показать, кто и как использует систему и с какими внешними компонентами она взаимодействует. Здесь важно понять, какие внешние зависимости и интерфейсы существуют, и как система вписывается в более широкую экосистему.


Рисунок 11 – Диаграмма контекста

На рисунке 11 представлена диаграмма контекста, которая иллюстрирует взаимодействие пользователя с автоматизированной системой поддержки пользователей с применением LLM и RAG. В ней представлены два ключевых элемента:

1. Пользователь – взаимодействует с системой, обращаясь к системе с вопросами.

2. Автоматизированная система поддержки – обрабатывает запросы пользователя, ищет контекст во внутренней базе данных и непосредственно генерирует ответ.

**Диаграмма контейнеров (Containers Diagram)**

Диаграмма контейнеров описывает высокоуровневую архитектуру программного обеспечения и то, как оно распределяет обязанности между своими узлами. Диаграмма показывает основные технологические решения и соотношение контейнеров друг с другом. Это простая, высокоуровневая диаграмма, которая должна быть понятна как разработчикам программного обеспечения, так и персоналу службы поддержки и эксплуатации. Диаграмма контейнеров для автоматизированной системы поддержки пользователей представлена на рисунке Б.1 приложения Б. Она иллюстрирует архитектуру системы,

показывая как следующие контейнеры взаимодействуют друг с другом: пользователь, frontend, backend, база данных и облачное хранилище. Они описаны ниже вместе со своими компонентами.

1. Пользотель взаимодействует с системой, отправляя текстовые запросы через веб-интерфейс. Запрос запускает процесс обработкиFrontend реализован с помощью следующих технологий: HTML5, CSS3, JavaScript. Он отвечает за визуальную составляющую интерфейса и устанавливает WebSocket-соединение для обмена данными в реальном времени. Также обеспечивает двустороннюю связь с системой и отображает сгенерированные ответыBackend представляет собой серверную часть систем, которая обрабатывает запросы. Она управляет данными и взаимодействует с внутренними сервисами. Серверная часть включает в себя несколько компонентов:Сервис API (REST API) – обеспечивает взаимодействие между клиентом и сервером через WebSocket. Обрабатывает запросы пользователя и предоставляет доступ к API для дальнейшей генерации ответа с использованием LLM и RAG.

3.2. Брокер сообщений (RabbitMQ) – обеспечивает асинхронную обработку запросов, используя механизм очередей, передавая данные между компонентами системы.

3.3. RAG обращается к базе данных для формирования контекста и передает его для дальнейшей генерации ответа.

3.4. LLM генерирует ответы на основе запроса пользователя и контекста, полученного от RAG.

4. Объектное хранилище (MinIO) –хранит документы и файлы, является внутренним доменом знаний. Передает файлы в базу данных для дальнейшей обработки и векторизации.

5. База данных (ChromeDB) – преобразует в векторные представления файлы, полученные от MinIO и сегментирует их на блоки для последующей обработки системой.

**Диаграмма компонентов (Components Diagram)**

На этом уровне предоставляются подробности о структуре контейнеров. Определенный контейнер разбивается на более мелкие части – компоненты. Это позволяет глубже понять, какие бизнес-логики реализуют эти компоненты и как они взаимодействуют. Диаграмма компонентов помогает увидеть, как конкретные функции и задачи разделяются между различными модулями и компонентами. Пример диаграммы компонентов показан на рисунке Б.2 приложения Б. Диаграмма контейнеров уже описывает все ключевые элементы системы, кроме деталей компонентов внутри сервиса API. Поэтому здесь рассмотрим только его составные части.

1. API Controller обрабатывает запросы, поступающие от пользователя через WebSocket. Он управляет тем, как система взаимодействует с пользователем, и пересылает запросы другим компонентам для выполнения бизнес-логики и возврата ответов. API Controller также отвечает за маршрутизацию запросов.

2. Request Handler является обработчиком запросов пользователей. Он анализирует данные и определяет какие действия необходимо выполнить.

3. Error Handler управляет ошибками на уровне API. Обеспечивает обработку исключений и возвращает пользователю информацию о сбоях системы.

Эти компоненты взаимодействуют следующим образом: API Controller принимает запросы от пользователей через WebSocket и передает их Request Handler для обработки. В свою очередь Request Handler взаимодействует с RAG и LLM и обращается к Error Handler для обработки любых ошибок. После обработки запроса результат возвращается пользователю через API Controller.

**Диаграмма кода (Code Diagram)**

Этот уровень является самым детализированным и показывает, как компоненты реализованы на уровне исходного кода. Диаграмма кода может быть представлена в виде классов или других конструкций, отображающих

структуру кода. Это помогает разработчикам и архитекторам понять, как именно реализованы компоненты на уровне программирования.

Для большинства задач проектирования и разработки системы достаточно первых трех уровней архитектурных артефактов – концептуального, логического и физического. Эти уровни обеспечивают полное представление о структуре, взаимодействии компонентов и их развертывании, что удовлетворяет потребности архитекторов, разработчиков и других участников проекта. Диаграмма кода, представляющая самый нижний уровень детализации и отображающая структуру исходного кода, является необязательной. Ее создание часто избыточно, так как код подвержен частым изменениям, а современные инструменты разработки автоматически визуализируют классы и зависимости.

## 2.5. Контейнеризация с использованием Docker

Docker [29] – это платформа для контейнеризации, которая позволяет упаковывать приложения и их зависимости в единый контейнер, обеспечивая их изоляцию, переносимость и единообразие работы в различных средах. Контейнеры представляют собой легковесные виртуальные среды, которые используют ресурсы хост-системы, но при этом изолированы друг от друга на уровне операционной системы. Основное преимущество Docker заключается в том, что он устраняет проблему несоответствия окружений: приложение, запущенное в контейнере, будет работать одинаково на сервере разработчика, в тестовой среде или в эксплуатационной среде, независимо от различий в конфигурации этих систем.

Docker помогает решить ряд важных задач в автоматизированной системе поддержки пользователей. С помощью контейнеризации можно изолировать разные части системы, такие как Frontend, Backend, облачное хранилище и базу данных. Это предотвращает конфликты между зависимостями и позволяет каждой части работать независимо.

Более того Docker упрощает развертывание, так как представляет единый способ развертывания системы на серверах, не требуя ручной настройки окружения. Контейнеры скрывают зависимости от операционной системы, что экономит время на установку и сокращает вероятность ошибок.

Система также легко масштабируется благодаря таким инструментам, как Docker Compose или Kubernetes, что позволяет адаптироваться к увеличивающимся нагрузкам. Безопасность улучшается за счет изоляции контейнеров, что снижает риск доступа к данным. Возможности ограничения ресурсов, такие как CPU и память, также помогаю избежать проблем, связанных с перегрузкой системы.

Кроме того, Docker управляет зависимостями, фиксируя их в контейнерных образах, что исключает необходимость переустановки при переносе приложения на другой сервер. Это снижает риск ошибок, связанных с отсутствующими или несовместимыми библиотеками. Ниже описана структура контейнеров.

1. Frontend-контейнер. Содержит веб-компонент, реализованный с использованием HTML, CSS и JavaScript. Этот контейнер отвечает за отображение интерфейса пользователя и установление WebSocket-соединения с сервером.

2. Backend-контейнер. Включает серверную часть системы, реализованную на Python с использованием FastAPI. Этот контейнер обрабатывает запросы пользователей, взаимодействует с RAG и LLM, а также управляет асинхронной обработкой через RabbitMQ.

3. RabbitMQ-контейнер. Это брокер сообщений, который помогает обрабатывать запросы без задержек. Он работает в изолированной сети и доступен только для Backend.

4. MinIO-контейнер. Это облачное хранилище для документов и файлов, представляющее собой доменом знаний. Образ, который используется в этом контейнере является настраиваемым. В нем можно ограничить доступ,

шифровать данные, что является важной составляющей для обеспечения защиты данных.

5. ChromaDB-контейнер. Этот контейнер хранит векторные данные и контекст, которые использует технология RAG для поиска нужной информации. ChromaDB разбивает и индексирует текстовые данные из документов, для быстрого нахождения векторных представлений и предоставления нужного контекста для формирования ответов.

Все контейнеры работают в одной сети, что защищает их от внешних систем и обеспечивает безопасное взаимодействие. Для управления контейнера используется Docker Compose, который упрощает настройку всех сервисов в одном файле `docker-compose.yam`. Это делает запуск системы проще и гарантирует согласованность окружения.

Использование Docker в проекте дает значительные преимущества, такие как переносимость, изоляция и упрощение управления зависимостями. Однако есть и ограничения: Docker требует дополнительных ресурсов для работы (например, дискового пространства для хранения образов), а также знаний для настройки и управления контейнерами. В данном проекте эти ограничения минимизированы за счет использования легковесных образов и тщательной настройки сети.

## 2.6. Проектирование интерфейса

Для более полного представления о будущем интерфейсе были разработаны макеты интерфейса. На рисунке 12 представлена кнопка с названием «Поддержка», которая располагается в правом нижнем углу страницы.

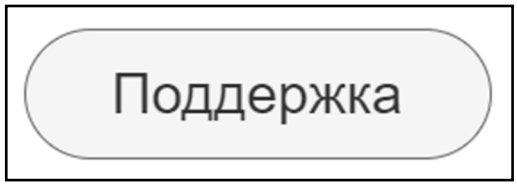


Рисунок 12 – Макет кнопки «Поддержка»

После нажатия на кнопку «Поддержка» открывается диалоговое окно (рисунок 13). Пользователь видит приветственное сообщение, а в нижней части экрана – поле ввода текста и кнопку отправки сообщения. В правом верхнем углу расположена кнопка для закрытия окна. После отправки сообщения пользователем приветственное сообщение «Задайте ваш вопрос…» исчезает, и в поле для отображения сообщений появляется запрос пользователя и ответ чата поддержки.


Рисунок 13 – Макет диалогового окна

**Вывод по второй главе**

В ходе проектирования были определены функциональные и нефункциональные требования к системе. На основе этих требований была построена диаграмма вариантов использования. Была спроектирована архитектура системы, а также описана роль контейнеризации в ней. Для визуального представления были спроектированы макеты интерфейса.

# 3. РЕАЛИЗАЦИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ

Для создания визуального компонента автоматизированной системы поддержки пользователей был разработан веб-компонент, реализующий функционал виртуального ассистента с возможностью обмена сообщениями в реальном времени. Компонент обеспечивает оперативное взаимодействие пользователей с системой через интерфейс, используя протокол WebSocket для связи с сервером. Для обработки и форматирования ответов от бота применена библиотека Markdown-it [30], что позволяет представлять информацию в структурированном и читаемом виде с поддержкой Markdown-разметки. В данном разделе описаны ключевые этапы реализации системы, включая создание веб-компонента, настройку серверной части, интеграцию с технологиями LLM и RAG.

## 3.1. Создание и подключение веб-компонента

Разработка компонента основана на стандарте Web Components, который предоставляет возможности создания самодостаточных и переиспользуемых элементов интерфейса, совместимых с любыми веб-платформами. Основой служит класс `ChatBot`, наследующийся от `HTMLElement`, что позволяет интегрировать его в DOM как пользовательский тег. Для защиты внутренней структуры и стилей от внешнего влияния используется Shadow DOM, инициализируемый в открытом режиме через метод `attachShadow({mode:"open"})`. Этот механизм обеспечивает инкапсуляцию, предотвращая конфликты с глобальными стилями страницы. Во время инициализации компонента подключается шаблон интерфейса, который определяет визуальную часть, а также выполняется проверка на наличие библиотеки Markdown-it. Если библиотека успешно загружена, создается ее экземпляр с опцией поддержки HTML, что расширяет возможности форматирования ответов. При отсутствии библиотеки в консоль выводится сообщение об ошибке, что упрощает диагностику проблем на этапе разработки. Кроме того, в

процессе инициализации регистрируются обработчики событий для ключевых элементов интерфейса – кнопок открытия и закрытия чата, а также поля ввода, что обеспечивает мгновенную реакцию на действия пользователя.

### 3.2. Структура и взаимодействие с интерфейсом

Интерфейс компонента включает кнопку «Поддержка», расположенную в правом нижнем углу страницы, которая служит триггером для открытия или скрытия диалогового окна. Оно представляет собой контейнер с фиксированными размерами, содержащий заголовок (по умолчанию «Техподдержка»), область для отображения сообщений, поле ввода текста и кнопку отправки. При активации окна через метод переключения видимости фокус автоматически устанавливается на поле ввода, что ускоряет процесс взаимодействия. Когда пользователь вводит сообщение и подтверждает его нажатием клавиши Enter или кликом по кнопке отправки, запускается метод обработки ввода. Этот метод включает проверку текста на пустоту, добавление сообщения в массив истории переписки (где каждое сообщение классифицируется как user или assistant с соответствующим содержимым), и отправку данных на сервер через WebSocket. Если соединение с сервером отсутствует или не активно, система фиксирует ошибку в консоли. После отправки сообщение пользователя визуально отображается в чате, а поле ввода очищается. Начальный текст «Задайте ваш вопрос...» скрывается после первого ввода, что сигнализирует о начале активного диалога.

### 3.3. Обработка WebSocket-сообщений

Подключение компонента к DOM инициирует установку WebSocket-соединения с сервером, что обеспечивается с использованием атрибутов компонента: `uuid` (уникальный идентификатор пользователя для маршрутизации), `host` (адрес сервера), `prompt` (системное сообщение, задающее поведение виртуального ассистента) и `temp` (параметр температуры для настройки генерации текста). Инициализируется массив сообщений, начинающийся с системного

сообщения, и объект настроек, включающий максимальное количество токенов и указанную температуру. WebSocket-соединение формируется с динамическим выбором протокола (ws:// или wss://) в зависимости от текущего протокола страницы, что гарантирует совместимость с безопасными соединениями. При успешном открытии соединения в консоль выводится уведомление, а при возникновении ошибок или разрыва связи регистрируются соответствующие сообщения для отладки.

Обработка входящих данных от сервера осуществляется в реальном времени через метод, который реагирует на события `onmessage`. При получении начального фрагмента ответа создается новый элемент для сообщения бота, поле ввода блокируется для предотвращения повторной отправки, и начинается пошаговое отображение текста. По мере поступления данных сообщение обновляется, а при получении маркера `[EOF]`, обозначающего конец ответа, поле ввода становится активным, текст сохраняется в истории переписки как ответ assistant и форматируется через Markdown-it для отображения в чате.

### 3.4. Разметка и стилизация интерфейса

Компонент реализован как HTML-элемент с настраиваемыми атрибутами, что делает его гибким инструментом для интеграции. Интерфейс включает кнопку «Поддержка», фиксированную в правом нижнем углу с зеленым фоном и эффектом затемнения при наведении, контейнер чата с белым фоном, тенью и скругленными углами, область сообщений с поддержкой прокрутки, поле ввода и кнопку отправки.

Структура определяется с помощью элемента `<template>`, который содержит как разметку, так и встроенные стили CSS. Область сообщений различает сообщения пользователя (светло-зеленый фон) и бота (светло-серый фон), обеспечивая визуальную четкость. Поле ввода и кнопка отправки расположены в нижней части, с плавными переходами при наведении. Использование элементов `<slot>` позволяет изменить текст кнопки и заголовка, предоставляя пользователям возможность адаптировать компонент под свои нужды,

например, заменить «Поддержка» на «Помощь» или изменить название чата в зависимости от контекста применения. Визуализация интерфейса представлена на рисунке 14.


Рисунок 14 – Диалоговое окно

### 3.5. Реализация серверной части системы

Серверная часть автоматизированной системы поддержки пользователей реализована на языке Python с использованием фреймворка FastAPI, который обеспечивает высокую производительность и асинхронную поддержку операций. Она отвечает за обработку пользовательских запросов, взаимодейсвие с векторной базой данных ChromeDB, объектным хранилищем MinIO, брокером сообщений RabbitMQ, языковой моделью и технологией RAG. Все компоненты развернуты в контейнерах Docker, что гарантирует изоляцию, переносимость и стабильную работу в изолированной среде.

Пользовательские запросы поступают через WebSocket в FastAPI, который принимает данные в формате JSON они включают в себя текст запроса, историю переписки (массив сообщений с ролями user и system) и параметры

для LLM, такие как температура и максимальное количество токенов (рисунок 15).

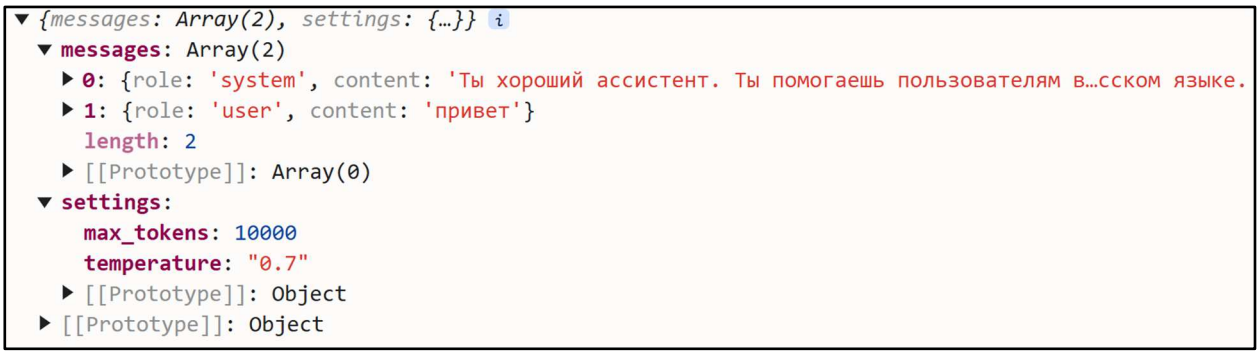


Рисунок 15 – Результат отправки на сервер

FastAPI проверят формат запросов на корректность и направляет в RabbitMQ для асинхронной обработки, что предотвращает перегрузку сервера при выполнении ресурсоемких задач, таких как генерация больших текстов или анализа объемных данных. RabbitMQ в свою очередь распределяет задачи между рабочими процессами, обеспечивая стабильность системы при высоком трафике.

Для поиска релевантного контекста используется ChromeDB, которая хранит векторные представления документов. Исходные документы, такие как техническая документация и инструкции извлекаются из объектного хранилища MinIO, обеспечивающего безопасное хранение. Текст документов сегментирует и преобразуется в векторные эмбеддинги с помощью моделей машинного обучения, что позволяет представить информацию в векторном пространстве. Технология RAG выполняет поиск по этим эмбеддингам, извлекая, извлекая наиболее подходящий контекст, который затем передается в LLM совместно с запросом пользователя для генерации ответа. LLM, настроенная для работы в изолированном контуре, генерирует текст в реальном времени, не создавая при этом задержек. Ответ возвращается через WebSocket и представляется в виде последовательных фрагментов, что позволяет пользователю видеть информацию по мере ее обработки.

Система расположена в контейнерной инфраструктуре Docker, что позволяет каждому компоненту работать в своем собственном контейнере. Это создает внутреннюю сеть, в которой элементы могут безопасно взаимодействовать между собой. Такой подход делает развертывание системы проще и снижает риск возникновения проблем с зависимостями.

Серверная часть системы построена по модульному принципу. Это упрощает ее использование в изолированной среде, поскольку она способна оперативно обрабатывать запросы и предоставлять релевантные ответы. Таким образом, система имеет высокую производительность и масштабируемость, что делает ее подходящей для использования во внутренней экосистеме предприятия.

**Вывод по третьей главе**

В третьей главе описан процесс разработки системы. Рассмотрены этапы создания компонента, а также механизмы взаимодействия с пользователем через кнопки и поле ввода. Описана реализация серверной части, обеспечивающей обработку запросов пользователей и генерацию ответов.

# 4. ТЕСТИРОВАНИЕ СИСТЕМЫ НА ПРЕДПРИЯТИИ

## 4.1. Функциональное тестирование

Функциональное тестирование выполняется с целью проверки соответствия разработанного программного обеспечения изначально заявленной функциональности системы и ее первоначальным функциональным требованиям. Функциональное тестирование также помогает выявить любые потенциальные дефекты или несоответствия в работе программного продукта. Результаты функционального тестирования системы представлены в таблице 3.

Таблица 3 – Тестирование системы

| **Название теста** | **Шаги** | **Ожидаемый результат** | **Тест пройден?** |
|---|---|---|---|
| Открытие диалогового окна | 1. Нажать на кнопку «Поддержка»<br>2. Убедиться, что открывается диалоговое окно с приветственным сообщением, полем ввода и кнопкой отправки сообщения | Диалоговое окно появляется, в нем отображается приветственное сообщение, поле ввода и кнопка отправки | Да |
| Отправка сообщения | 1. Ввести текст в поле ввода<br>2. Нажать на кнопку отправки<br>3. Убедиться, что сообщение пользователя появляется в чате, а ассистент начинает отправку ответа | Сообщение пользователя появляется в чате, ассистент начинает «печатать» ответ | Да |
| Ответ виртуального ассистента | 1. Отправить сообщение пользователем<br>2. Дождаться ответа от бота, который должен отобразиться в чате | Ответ бота динамически появляется в чате | Да |
| Обработка пустого сообщения | 1. Оставить поле ввода пустым<br>2. Нажать на кнопку отправки<br>3. Убедиться, что бот не отправляет пустое сообщение и выводит предупреждение | Бот не отправляет пустое сообщение, выводится сообщение об ошибке | Да |

Продолжение таблицы 3

| Название теста | Шаги | Ожидаемый результат | Тест пройден? |
|---|---|---|---|
| Проверка состояния поля ввода после отправки сообщения | 1. Ввести сообщение и нажить кнопку отправить 2. Нажать на поле ввода сообщений пока бот печатает и убедиться, что поле не активное | Пока бот отвечает на предыдущее сообщение поле для ввода сообщения не активное | Да |
| Проверка сохранения истории переписки | 1. Провести диалог из 3 - 4 сообщений 2. Закрыть и открыть диалоговое окно | Все предыдущие сообщения должны отображаться в том же порядке | да |
| Проверка корректности вывода Markdown | 1. Отправить сообщение с обычным текстом, например: «Привет, как дела?» 2. Убедиться, что бот корректно обрабатывает и форматирует Markdown-разметку | Ответ бота, содержащий Markdown-разметку (например, жирный текст или ссылки), должен быть правильно отформатирован и отображен в чате. Бот должен преобразовывать сообщения в правильный формат | Да |

В результате проведенного функционального тестирования удалось убедиться, что все основные функции веб-компонента работают правильно. Тесты показали, что пользователь может открывать окно поддержки, отправлять сообщения и получать ответы от виртуального помощника без сбоев.

Также были проверены дополнительные моменты, такие как реакция на пустые сообщения, работа поля ввода во время ответа ассистента и корректное отображение текста с разметкой Markdown. Во всех случаях программа вела себя так, как ожидалось.

## 4.2. Пользовательское тестирование

Пользовательское тестирование автоматизированной системы поддержки проведенное на базе ВНИИТФ потвердело ее соответствии заявленным требования. В ходе тестирования респонденты особо отметили продуманный и удобный интерфейс: оптимально подобранные размеры диалогового окна обеспечивают комфортное взаимодействие, плавные анимации открытия и закрытия создают приятное впечатление от работы с системой. Важной

функциональной особенностью, получившая положительные отзывы, стала возможность сохранения истории переписки. Пользователи могут закрывать чат или обновлять страницу без потери контекста диалога. Контрастное и интуитивно понятное оформление элементов значительно упрощает навигацию и сокращает время адаптации к системе. В процессе тестирования, включавшего различные сценарии работы, система продемонстрировала стабильную и надежную работу без каких-либо сбоев или критичных ошибок. По мнению участников система готова к эксплуатации и не требует дополнительных доработок, что свидетельствует о качестве ее реализации и соответствия всем заявленным требованиям.

**Вывод по четвертой главе**

По результатам проверки все тестовые сценарии из подготовленного набора выполнены успешно, что подтверждает корректную работу функционала и пользовательского интерфейса приложения. Кроме того, было проведено пользовательское тестирование, в ходе которого были выявлены недостатки системы, которые были успешно устранены.

# ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана автоматизированная система поддержки пользователей с применением LLM и RAG, обеспечивающая интерактивное взаимодействие с виртуальным ассистентом. В ходе выполнения работы были решены задачи.

1. Проведен анализ предметной области и существующих решений.
2. Спроектирована архитектура автоматизированной системы.
3. Реализована спроектированная автоматизированная система.
4. Пройдено тестирование системы на предприятии.
5. Пройден обязательный нормоконтроль.

Акт о внедрение автоматизированной системы пользователей, подтверждающий успешное прохождение испытаний и ввод в эксплуатацию, представлен на рисунке В.3 приложения В.

Дальнейшее развитие системы планируется с акцентом на расширение функциональности, включая внедрение адаптивных алгоритмов обработки запросов для повышения точности ответов, а также интеграцию дополнительных модулей для анализа пользовательских запросов и формирования статистики. Также система может быть использована на различных ресурсах предприятия для обработки запросов и оптимизации внутренних процессов.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Российский федеральный ядерный центр – ВНИИТФ им. академика Е.И. Забабахина [Электронный ресурс] // URL: https://vniitf.ru/ (дата обращения: 29.10.2024).

2. Блюм, В.С. Семантические принципы разработки текстовых систем искусственного интеллекта / В.С. Блюм, В.М. Космачев [и др.] // Интеллектуальные технологии на транспорте. – 2023. – URL: https://e.lanbook.com/journal/issue/351149 (дата обращения: 16.11.2024).

3. Актуальные проблемы инфотелекоммуникаций в науке и образовании: материалы конф. – СПб.: СПбГУТ им. М.А. Бонч-Бруевича, 2023. – Т. 2. – 997 с. – URL: https://e.lanbook.com/book/426002 (дата обращения: 16.11.2024).

4. Восьмая научно-техническая конференция студентов и аспирантов МИРЭА – РТУ: сб. тр. – М.: РТУ МИРЭА, 2023. – 858 с. – URL: https://e.lanbook.com/book/382673 (дата обращения: 17.11.2024).

5. Современные вопросы педагогики и психологии: теоретико-методологические подходы и практические результаты исследований: монография / А.А. Киселев, А.И. Кугай, Г.И. Авходиев [и др.]; под ред. Ж.В. Мурзиной, А.С. Егоровой. – Чебоксары, 2024. – 172 с. – URL: https://e.lanbook.com/book/412652 (дата обращения: 18.11.2024).

6. What is Fine-Tuning? [Электронный ресурс] // URL: https://www.geeksforgeeks.org/what-is-fine-tuning/ (дата обращения: 05.02.2025).

7. What is Low Rank Adaptation (LoRA) [Электронный ресурс] // URL: https://www.geeksforgeeks.org/what-is-low-rank-adaptation-lora/ (дата обращения: 05.02.2025).

8. What is Retrieval – Augmented Generation (RAG) [Электронный ресурс] // URL: https://www.geeksforgeeks.org/what-is-retrieval-augmented-generation-rag/ (дата обращения: 20.11.2024).

9. Продвинутые методы улучшения качества продуктов с LLM: RAG [Электронный ресурс] // URL: https://gopractice.ru/skills/improving-products-with-llm-rag/ (дата обращения: 20.11.2024).

10. Черенкова, И.А. Использование цифровых технологий в АПК. Компьютерные сети. Информационная безопасность: учеб. пособие / И.А. Черенкова, И.В. Кутликова, М.В. Новиков, В.В. Степанишин. – М.: МГАВМиБ им. К.И. Скрябина, 2022. – 128 с. – URL: https://e.lanbook.com/book/331406 (дата обращения: 20.11.2024).

11. Добавление собственных данных в LLM с помощью RAG [Электронный ресурс] // URL: https://habr.com/ru/companies/wunderfund/articles/779748/ (дата обращения: 22.10.2024).

12. Understand Hybrid Search [Электронный ресурс] // URL: https://docs.oracle.com/en/database/oracle/oracle-database/23/vecse/understand-hybrid-search.html (дата обращения: 23.10.2024).

13. Макшанов, А.В. Технологии интеллектуального анализа данных: учеб. пособие / А.В. Макшанов, А.Е. Журавлев. – 2-е изд., стер. – СПб.: Лань, 2022. – 212 с. URL: https://e.lanbook.com/book/206711 (дата обращения: 26.11.2024).

14. Терещенко, П.В. Проектирование и анализ человеко-компьютерного взаимодействия: учеб. пособие / П.В. Терещенко. – Новосибирск: НГТУ, 2021. – 96 с. – URL: https://e.lanbook.com/book/216326 (дата обращения: 30.11.2024).

15. Кононова, О.В. Технологии извлечения и интеллектуального анализа данных в научных исследованиях: учеб. пособие / О.В. Кононова, Д.Е. Прокудин. – СПб.: НИУ ИТМО, 2021. – 133 с. – URL: https://e.lanbook.com/book/283685 (дата обращения: 01.12.2024).

16. Госуслуги [Электронный ресурс] // URL: https://www.gosuslugi.ru (дата обращения: 10.02.2025).

17. Сбербанк [Электронный ресурс] // URL: https://online.sberbank.ru (дата обращения: 10.02.2025).

18. Чат с Алисой и YaGPT [Электронный ресурс] // URL: https://alice.yandex.ru (дата обращения: 10.02.2025).

19. FastAPI [Электронный ресурс] // URL: https://fastapi.tiangolo.com (дата обращения: 17.02.2025).

20. Хабаров, С.П. Построение распределенных систем на базе WebSocket: учеб. пособие / С.П. Хабаров, М.Л. Шилкина. – 2-е изд., стер. – СПб.: Лань, 2022. – 216 с. – URL: https://e.lanbook.com/book/200510 (дата обращения: 18.02.2025).

21. RabbitMQ [Электронный ресурс] // URL: https://www.rabbitmq.com (дата обращения: 18.02.2025).

22. MinIO [Электронный ресурс] // URL: https://min.io (дата обращения: 20.02.2025).

23. Современный учебник JavaScript [Электронный ресурс] // URL: https://learn.javascript.ru (дата обращения: 20.02.2025).

24. Богданов, М.Р. HTML5. Полный курс: учеб.-метод. пособие / М.Р. Богданов, Л.В. Вахидова, И.Н. Думчикова, Л.В. Миниярова. – Уфа: БГПУ им. М. Акмуллы, 2015. – 168 с. – URL: https://e.lanbook.com/book/72507 (дата обращения: 21.02.2025).

25. ChromaDB [Электронный ресурс] // URL: https://docs.trychroma.com/docs/overview/introduction (дата обращения: 27.02.2025).

26. Weaviate [Электронный ресурс] // URL: https://weaviate.io (дата обращения: 27.02.2025).

27. Qdrant [Электронный ресурс] // URL: https://qdrant.tech (дата обращения: 27.02.2025).

28. Use Case Diagram [Электронный ресурс] // URL: https://www.geeksforgeeks.org/use-case-diagram/ (дата обращения: 15.03.2025).

29. Документация C4 Model [Электронный ресурс] // URL: https://c4model.com/ (дата обращения: 15.03.2025).

30. Docker: Accelerated Container Application Development [Электронный ресурс] // URL: https://www.docker.com (дата обращения: 10.04.2025).

31. Markdown-it 14.1.0 API documentation [Электронный ресурс] // URL: https://markdown-it.github.io/markdown-it/ (дата обращения: 11.04.2025).

# ПРИЛОЖЕНИЕ А
# ОСНОВНАЯ ЧАСТЬ ТЕХНИЧЕСКОГО ЗАДАНИЯ

1. Введение

1.1. Наименование программного изделия

Автоматизированная система поддержки пользователей для ФГУП РФЯЦ-ВНИИТФ.

1.2. Краткая характеристика области применения

Автоматизированная система поддержки пользователей предоставляет интерактивного ассистента для оперативного получения ответов на вопросы, связанные м различными задачами. Программное изделие предназначено для автоматизации поддержки пользователей через виртуального ассистента. Система разрабатывается как нативный веб-компонент, обеспечивающий интеграцию в веб-сайты и порталы.

2. Назначение разработки

Целью создания автоматизированной системы является достижение следующих результатов:

− автоматизация процессов поддержки сотрудников;

− обеспечение оперативного доступа к актуальной информации;

− оптимизация внутренних процессов;

− снижение нагрузки на персонал поддержки.

3. Требования к программе или программному изделию

3.1. Требования к функциональным характеристикам:

− система должна позволять пользователю отправлять текстовые запросы через поле ввода в интерфейсе веб-компонента;

− система должна принимать и отображать текстовые ответы от сервера в диалоговом окне;

− система должна хранить историю переписки для обеспечения логической непрерывности диалога.

– система должна быть скрыта по умолчанию, открываться и сворачиваться по нажатию кнопки, сохраняя состояние при повторном открытии;

– система должна легко интегрироваться в веб-сайты или порталы.

3.2. Требования к надежности:

– система должна иметь механизм обработки ошибок, обеспечивающий возобновление работы после сбоев без потери данных переписки;

– система должна поддерживать автоматическое сохранение контекста переписки, чтобы предотвратить потерю данных при сбоях браузера или сети.

3.3. Требования к составу и параметрам технических средств:

– серверная часть подсистемы должна работать на серверах под управлением Astra Linux (версия не ниже 2.12);

– минимальные требования к серверу: процессор с 4 ядрами (2.5 ГГц), 16 ГБ ОЗУ, 100 ГБ свободного дискового пространства (SSD предпочтительно);

– клиентская часть должна поддерживаться на рабочих станциях с процессором не ниже 2 ГГц, 4 ГБ ОЗУ и разрешением экрана не менее 1280x720;

– подсистема должна быть развернута в контейнерах с использованием Docker.

3.4. Требования к информационной и программной совместимости:

– система должна использовать WebSocket для взаимодействия между клиентом и сервером, с передачей данных в формате JSON;

– данные должны храниться в ChromaDB в формате векторных представлениях, с использованием MinIO для хранения исходных документов;

– клиентская часть подсистемы должна быть реализована с использованием JavaScript (ES6+), HTML5 и CSS3.

– совместимость с внутренними системами предприятия обеспечивается через REST API;

– разработка и поддержка подсистемы выполняются в среде Visual Studio Code (версия не ниже 1.85).

4. Сроки по созданию и стадии разработки

Дата начала работы: 3 февраля 2025 года.

Дата окончания работы: 30 мая 2025 года.

Стадия 1. Анализ требований и разработка пользовательского интерфейса (3 февраля – 2 марта 2025).

Сбор и анализ требований к подсистеме, включая функциональные характеристики и работу в закрытой сети Astra Linux. Разработка интерфейса для автоматизированной системы с использованием JavaScript (ES6+), HTML5 и CSS3 в среде Visual Studio Code (версия не ниже 1.85).

Стадия 2. Проектирование серверной логики и интеграции (3 марта – 11 апреля 2025).

Разработка архитектуры серверной части на Python FastAPI, проектирование WebSocket API, настройка интеграции с ChromaDB, MinIO и RabbitMQ. Определение структуры хранения данных в ChromaDB.

Стадия 3. Реализация, тестирование и внедрение (14 апреля – 30 мая 2025).

Программирование серверной части (FastAPI), настройка ChromaDB, MinIO и RabbitMQ, контейнеризация в Docker, интеграция с веб-компонентом через WebSocket. Настройка LLM для генерации ответов.

# ПРИЛОЖЕНИЕ Б
# ДИАГРАММЫ АРХИТЕКТУРЫ СИСТЕМЫ

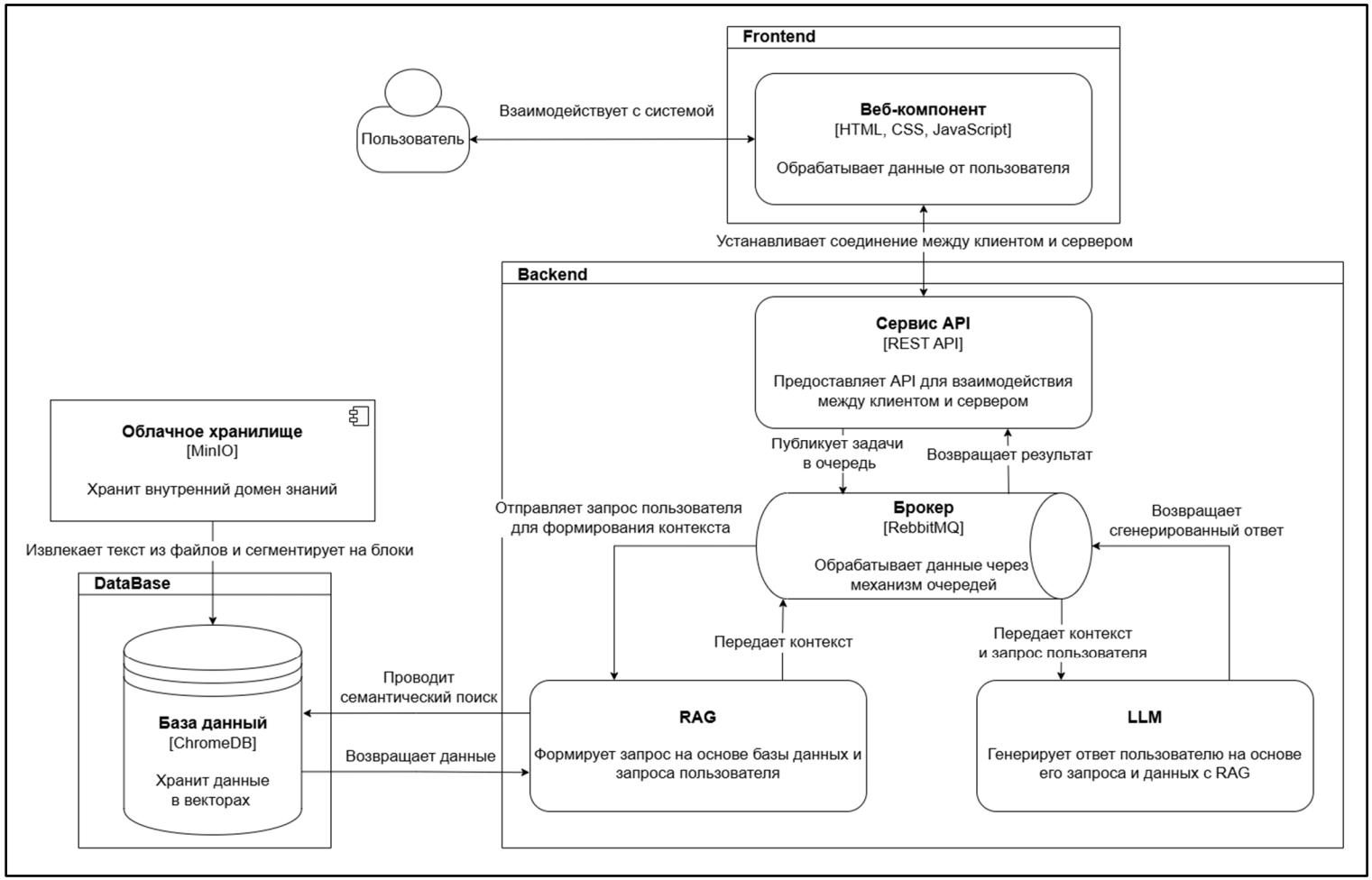


Рисунок Б.1 – Диаграмма контейнеров

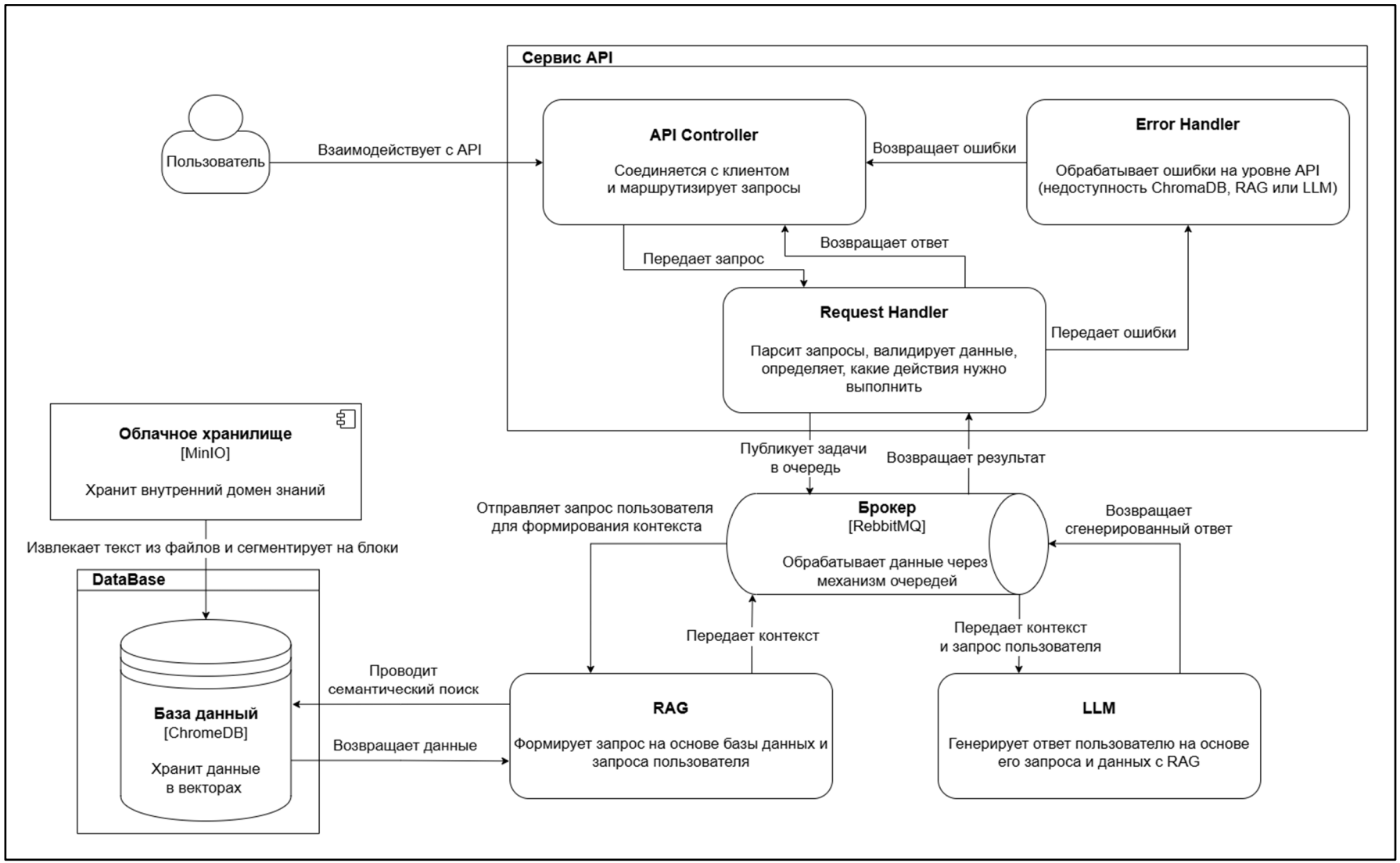


Рисунок Б.2 – Диаграмма компонентов

# ПРИЛОЖЕНИЕ В
# АКТ О ВНЕДРЕНИИ

**АКТ**
**о внедрении научно-технической продукции**

Данный акт удостоверяет, что в ФГУП «РФЯЦ-ВНИИТФ им. академ. Е.И. Забабахина» внедрена в опытную эксплуатацию автоматизированная система поддержки пользователей с применением LLM и RAG, разработанная студентом группы КЭ-406 Кудряшовым Иваном Алексеевичем, научный руководитель – доцент кафедры «Электронные вычислительные машины» ФГАОУ ВО «ЮУрГУ (НИУ)» Плаксина Юлия Геннадьевна.

Акт подписал
Специалист по з.и.
_____ Балыбин Н.С.
20.05.2025 г.
МП

Рисунок В.3 – Акт о внедрении