Министерство науки и высшего образования Российской Федерации
Фереральное государственное бюджетное образовательное учреждение высшего образования
Российский государственный гидрометеорологический университет (РГГМУ)
Институт информационных систем и геотехнологий
Направление подготовки: 09.03.03 «Прикладная информатика»
Профиль подготовки: «Прикладные информационные системы и геотехнологии»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

На тему: «Разработка приложения интеллектуального ассистента на базе технологий глубокого обучения.»

Научный руководитель,
к.т.н Петров Я.А. _____

Исполнитель студент группы
ПИ-Б20-2-2,
Попов В.Н. _____

«К защите допускаю»,
Заведующий кафедрой,
к.т.н,
Колбина О.Н.
«____» _____ 2024 г. _____

Санкт-Петербург 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4
1 Предпроектный анализ 6
1.1 Анализ предметной области 6
1.2 Сравнительный анализ 7
1.3 Системный анализ 8
1.4 Требования к сервису 9
1.5 Регламентирующие документы 12
1.6 Сроки реализации проекта 13
2 Проектирование информационной системы 14
2.1 Концептуальное проектирование 14
2.2 Диаграмма компонентов 16
2.3 Диаграмма развертывания 17
2.4 Диаграмма последовательности 18
2.5 Схема базы данных 18
2.6 Векторная база данных 19
2.7 Развертывание приложения 20
3 Разработка системы 21
3.1 Выбор средств разработки 21
3.2 Структура приложения 21
3.3 Расчет надежности программного и аппаратного обеспечения 23
3.4 Расчет ожидаемого результата экономической эффективности 26
3.5 Word2vec 27
3.6 Поиск подходящих векторов 30
3.7 Генерация текста при помощи поиска и аугментаций 32
3.8 Распознавание голоса 36
3.9 Распознавание видео 38
3.10 Промт-инженерия 39
3.11 Логирование 41
3.12 Тестирование 42
ЗАКЛЮЧЕНИЕ 46
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47
ПРИЛОЖЕНИЕ А Диаграмма последовательности 48

ПРИЛОЖЕНИЕ Б Конфигурация NeoVim 49
ПРИЛОЖЕНИЕ В Диаграмма Ганта 50
ПРИЛОЖЕНИЕ Г Конфигурация LaTeX 51
ПРИЛОЖЕНИЕ Д Реализация обработки различных данных 52

ВВЕДЕНИЕ

За последние десятилетия произошел огромный скачок в развитии информационных технологий: от создания первой электронной вычислительной машины, до сложный генеративных нейронных сетей (GAN). Сейчас компании проводят множественные исследования для выявления возможностей таких технологий и необходимости дальнейших инвестиций в данную отрасль. Одним из представителей GAN стали большие языковые модели (LLM). В настоящее время лидерами в данной отрасли стали: OpenAI, которые придумали реализовать интерфейс для взаимодействия с нейройнной сетью в виде чата; ПАО Сбербанк, реализовавшие отечественную LLM в условиях изоляции и ограниченных ресурсов; Meta (признана в РФ экстремистской организацией и запрещена), разработавшие малую языковую модель (SLM) LLaMA, ставшая прорывом для энтузиастов, у которых нет таких ресурсов для реализации LLM, как у больших игроков рынка.

В данной выпускной квалификационной работе будет разработана платформа, позволяющая взаимодействовать с ресурсами предприятия, использующая LLM/SLM для генерации релевантных ответов.

Актуальность данной темы обусловлена возможностью оптимизации многих процессов, увеличении производительности сотрудников, повышении лояльности клиентов. В контексте высшего учебного заведения (ВУЗ), использование технологий подобного рода повышает конкуретноспособность ВУЗа, что наряду с предыдущими пунктами является положительной метрикой.

Объект исследования — большие языковые модели (LLM).

Предмет исследования — применение языковых моделей в бизнесе.

Цель работы — проектирование и разработка приложения, которое позволяет взаимодействовать с структурой предприятий посредством интерфейса.

В качестве интерфейса для пользователя была выбрана оболочка в виде чат-бота. Чат-боты давно вошли в жизнь большинства населения. Это подтверждается информацией аналитической компании «eMarketer», согласно которой, чат-ботами пользуются более 1,4 млрд. человек на планете [2].

Для выполнения поставленной цели были поставлены следующие задачи:

– Выполнить анализ предметной области;
– Провести сравнительный анализ информационных систем;
– Рассчитать сроки реализации проекта;
– Смоделировать схему бизнес-процессов;
– Составить описание документов бизнес-процессов;
– Сформировать перечень требований к ИС;
– Исследовать подходы SWOT;
– Описать сценарии вариантов использования;
– Визуализировать описанные сценарии вариантов использования;
– Создать модель диаграммы компонентов;
– Создать модель диаграммы развертывания;
– Реализовать бизнес-логику ассистента и перенести его в интерфейс бота;
– Протестировать приложение;

В работе будет рассматриваться РГГМУ (далее Университет), но применяться бот сможет не только в конкретном учебном заведении, а для любых предприятий.

Во время разработки ассистента использовалась методология Agile. Она позволила работать в удобном темпе и формировать требования во время разработки.

В ходе выполнения практической части выпускной квалификационной работы были использованы: Python, LangChain, FAISS, HuggingFace, Transformers, Docker.

1 Предпроектный анализ

1.1 Анализ предметной области

Индустрия информационных технологий является одной из наиболее динамичных и быстроразвивающихся отраслей, где каждый год появляются новые тенденции и совершенствуются технологии, которые позволяют улучшить пользовательский опыт и приносить большую выгоду бизнесу. Одной из ключевых технологий стала технология трансформер, предложенная в статье “Attention is all you need”. За недолгий промежуток времени технология успела развиться и стать индустриальным стандартом. Увидеть применение технологии трансформер можно по созданным продуктам: GPT-4, BERT, T5, GigaChat.

Одной из ключевых особенностей трансформеров является их способность обрабатывать большие объемы текста без потери информации для выполнения задач таких как машинный перевод, обработка естественного языка, когнитивный анализ текста и генерирование текста. Трансформер состоит из блоков кодировщиков и декодеров, которые обрабатывают входные данные и генерируют выходные данные. Большое количество параметров сети позволяет ей улучшить качество работы по сравнению с другими моделями. В последнее время наблюдается тренд на внедрение больших языковых моделей в различные отрасли бизнеса, например системы автоматических ответов на вопросы, чат-боты, умные помощники.

Преимуществом таких решений является быстрый поиск информации и выдача её в удобоваримом виде, когда без использования таких ассистентов на поиск необходимой информации может потребоваться достаточный промежуток времени.

Не смотря на множество преемуществ данной технологии, все же присутствуют и большие минусы: сложность обучения и масштабирования моделей, большие требования к вычислительным ресурсам являются проблемой для массового пользователя.

В данной дипломной работе предлагается разработать информационную систему-помощника в виде чат-бота который будет представлять собой полезный инструмент как для студентов, так и для сотрудников ВУЗа. Основ-

ная идея информационной системы состоит в том, чтобы получить универсальный инструмент для взаимодействия со всей структурой университета. В рамках чат-бота пользователь сможет получить всю необходимую информацию, например информацию о заселении в общежития, списке необходимых документов для поступления и прочих вопросах, касающихся взаимодействия с университетом.

В общем и целом, интеграция технологии больших языковых моделей является актуальной и перспективной темой для дипломной работы, которая позволит изучить основы построения архитектуры приложения, интеграции технологий в предприятия, основы работы с нейросетями и машинным обучением, а также тестирования решений, где нет очевидных метрик для измерения результата.

1.2 Сравнительный анализ

На данный момент прямых конкурентов у моего решения нет, но я не отрицаю того, что в настоящий момент может разрабатываться схожее решение. Из схожих решений можно отметить следующие решения:

Боты от университетов. Такие решения не имеют возможности масштабирования, имеют ограниченный пул вопрос/ответ и привязанны к какой-то определенной платформе.

Virtual Spirits. Эта зарубежная компания специализируется на создании на создании чат-ботов для различных предприятий. Из преемуществ имеется возможность настройки внешнего вида бота. Сравнение моего приложения и приложений конкурентов приведено на таблице 1.1.

Таблица 1.1 — Сравнительный анализ

Критерий оценки	Мой сервис	Боты от университетов	Virtual Spirits
Масштабируемость	Высокая	Низкая	Высокая
Гибкость настройки	Высокая	Низкая	Высокая
Количество информации	Неограничено	Ограничено	Ограничено
Кроссплатформенность	Высокая	Низкая	Высокая
Стоимость обслуживания	Низкая	Неизвестно	Высокая
Простота интеграции	Высокая	Средняя	Средняя
Ручное обновление информации	Присутствует	Отсутствует	Отсутствует

Продолжение таблицы 1.1

Критерий оценки	Мой сервис	Боты от университетов	Virtual Spirits
Система авторизации	Отсутствует	Присутствует	Присутствует

Опираясь на проведенный анализ можно подвести некоторый итог: в конечной системе не будет системы авторизации, так как мне кажется, что вся информация должна быть в открытом доступе для всех возможных пользователей ИС.

Под удобным сбором информации подразумевается интутивно понятный процесс заполнения базы знаний, который может осуществляться как вручную, так и при помощи API, парсинга или других методов получения информации.

Возможность получать информацию не связанную с обучением мне кажется одним из ключевых преимуществ моей информационной системы: для абитуриентов может быть важно получить информацию как о возможном расписании, так и о поступлении в военный учебный центр (ВУЦ), получении бесконтактной смарты-карты (БСК) или же информации о истории университета.

1.3 Системный анализ

Исследование проектируемой ИС проводилось в виде SWOT анализа.

SWOT-анализ — метод стратегического планирования, суть которого заключается в выявлении факторов внутренней и внешней среды организации и разделении их на четыре категории: Strengths, Weaknesses, Opportunities, Threats. Сильные и слабые стороны представляют факторы внутренней среды объекта анализа. В свою очередь возможность и угрозы представляют собой внешнюю среду объекта анализа.

SWOT анализ находится в таблице 1.2.

Таблица 1.2 — SWOT анализ

	Положительное влияние	Негативное влияние
Внутренняя среда	— Актуальность — Простота использования	— Нейросетевые галюцинации — Необходимость тщательно прорабатывать интеграцию во избежание проблем с безопасностью
Внешняя среда	— Упрощение навигации по ресурсам ВУЗа — Получение поддержки от государства	— Регуляции со стороны государства — Бюрократия с какой-либо стороны

Из положительных аспектов можно выделить простоту конечного использования и улучшение взаимодействие с предприятием.

Из отрицательных факторов стоит выделить следующие аспекты: нейросетевые галюцинации, проблемы с безопасность и бюрократия.

Нейросетевые галюцинации — аномалия, возникающая во время работы нейронной сети, влияющая на вывод результат непредсказуемым образом. Например, спросив о технической оснащенности предприятия нейросеть может начать галлюцинировать и ответить, что у предприятия есть несколько квантовых суперкомпьютеров, хотя это не так. Частично решить эту проблему может грамотный промт-инженеринг, а так же правильное подмешивание контекста в сам промт.

Проблемы с безопасность можно отнести в ту же категорию. Если занести секретные данные в базу знаний, то велик шанс утечки информации и попадание её в руки злоумышленников. Для обхода этой проблемы нужно внимательно относиться к информации, которую вы собираетесь хранить в базе знаний и иметь базовые навыки кибербезопасности.

Бюрократия же не позволит внедрить информационную систему, занести все необходимые данные без множества согласований и утверждений со стороны вышестоящего руководства.

1.4 Требования к сервису

Функциональные и нефункциональные требования должны быть определены до начала реализации ИС, чтобы получить представление о конечном

продукте.

Для классификации требований использовалась модель FURPS. В следующих таблицах были приведены функциональные и нефункциональные требования ИС. Эти требования характеризуют поведение ИС. Функциональные требования приведены в таблице 1.3.

Таблица 1.3 — Функциональные требования

Номер требования	Описание
FUN_1	При начале общения бот должен представиться и рассказать о своих возможностях
FUN_2	По требованию пользователя бот должен предоставить контактную информацию вышестоящего руководства
FUN_3	По требованию пользователя бот должен предоставить информацию о расписании
FUN_4	По требованию пользователя бот должен предоставить информацию о списке направлений, на которые можно поступить с определенными предметами
FUN_5	По требованию пользователя бот должен предоставить информацию о университете
FUN_6	По требованию пользователя бот должен предоставить актуальную информацию о местах проведения практики
FUN_7	По требованию пользователя бот должен предоставить информацию о проводимых мероприятиях внутри университета

В таблице 1.4 приведены нефункциональные требования, затрагивающие удобство использования приложения.

Таблица 1.4 — Удобство использования

Номер требования	Описание
NFR_1	Пользователю предоставляется информация о сервисе
NFR_2	Использование сервиса не требует от пользователя какого-либо обучения
NFR_3	Бот предоставляет как текстовый, так и голосовой интерфейс для общения
NFR_4	По требованию пользователя бот должен предоставить информацию о списке направлений, на которые можно поступить с определенными предметами
NFR_5	Информация, предоставляемая ботом должна быть избыточной

В таблице 1.5 приведены нефункциональные требования, затрагивающие надежность сервиса.

Таблица 1.5 — Надежность сервиса

Номер требования	Описание
REL_1	Сервиса должен работать 24 часа в сутки 7 дней в неделю, за исключением технических перерывов.
REL_2	Точность предоставляемой информации зависит от предоставляемых организацией данных

В таблице 1.6 приведены нефункциональные требования, затрагивающие производительность сервиса.

Таблица 1.6 — Производительность сервиса

Номер требования	Описание
PER_1	На генерацию ответа у сервиса должно уходить не более 10 секунд
PER_2	Для работы боту необходимо: постоянный выход в сеть интернет, 4ГиБ оперативной памяти. Так же, опционально, чтобы бот был полностью автономен необходима видеокарта уровня RTX 3060 и выше для размещения сервиса на SLM В противном случае необходимо пользоваться посредником в виде LLM от Сбербанк, OpenAI, Mistral и т.п.

В таблице 1.7 приведены нефункциональные требования по поддержке сервиса, описывающие удобство применение конечного приложения для заказчика.

Таблица 1.7 — Поддержка сервиса

Номер требования	Описание
SUP_1	Установка сервиса осуществляется при помощи сценариев
SUP_2	Сервис должен поддерживать как работу с различными типами данных, поступающими от предприятия

В таблице 1.8 приведены нефункциональные требования к безопасности сервиса.

Таблица 1.8 — Требования к безопасности

Номер требования	Описание
SAF_1	Сервис должен использовать протокол HTTPS для обмена информацией

Изложенные требования в полной мере описывают все аспекты приложения и позвляют реализовать корректную работу.

1.5 Регламентирующие документы

Для соблюдения законодательства, соответствия требования заказчика, соответствия стандартам качества и урегулирования многих аспектов работы программного обеспечения используются регламентирующие документы: ГОСТ, пользовательское соглашение, рабочие инструкции и прочие.

Список регламентирующих документов приведен в таблице 1.9.

Таблица 1.9 — Регламентирующие документы

№	Наименование документа	Внутренний документ	Внешний документ
1	Политика приватности	+	-
2	ГОСТ 17657-79 Передача данных.	-	+
3	ГОСТ 34.321-96. Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными	-	+
4	ГОСТ Р 51904-2002. Программное обеспечение встроенных систем	-	+
5	ГОСТ 19.201-78. Техническое задание. требования к содержанию и оформлению	-	+
6	ГОСТ 19.401-78. Текст программы. требования к содержанию и оформлению	-	+
7	ГОСТ 34.201-89. Виды, комплектность и обозначение документов при создании автоматизированных систем	-	+
8	ГОСТ Р 50779.30-95. Приемочный контроль качества	-	+

1.6 Сроки реализации проекта

Для оценки требуемых ресурсов – времени и денег, разработаем график работ. Распишем основные блоки работ – это анализ, проектирование, разработка и документирование. Таким образом получаем, что на разработку системы потребуется 259 дней, что является довольно быстрым сроком. Для оценки стоимости проекта возьмем средние зарплаты специалистов и получим что на оплату труда уйдет 1,38 миллиона рублей. График работ представлен на рисунке 1.1 позволяет наглядно оценить длительность и стоимость работ.

№	Название этапа	Дата начала работ	Длительность	Дата окончания работы	Ответственное лицо	Заработная плата	Итого
1.0	Предпроектный анализ	01.09.2023	37	10.10.2023	Системный аналитик	5000	185000
1.1	Анализ предметной области	01.09.2023	14	15.09.2023	Системный аналитик	5000	70000
1.2	Анализ объекта исследования	16.09.2023	15	01.10.2023	Системный аналитик	5000	75000
1.3	Составление ТЗ	02.10.2023	8	10.10.2023	Системный аналитик	5000	40000
2.0	Проектный анализ	11.10.2023	19	31.10.2023	Системный аналитик	5000	95000
2.1	Анализ функциональных блоков	11.10.2023	10	21.10.2023	Системный аналитик	5000	50000
2.2	Анализ способов реализации	22.10.2023	9	31.10.2023	Системный аналитик	5000	45000
3.0	Проектирование	01.11.2023	40	14.12.2023	Системный аналитик	5000	200000
3.1	UR	01.11.2023	10	11.11.2023	Системный аналитик	5000	50000
3.2	BPMN	12.11.2023	10	22.11.2023	Системный аналитик	5000	50000
3.3	UML	23.11.2023	10	3.12.2023	Системный аналитик	5000	50000
3.4	Концептуальная модель	4.12.2023	10	14.12.2023	Системный аналитик	5000	50000
4.0	Реализация	15.12.2023	91	16.03.2024	Инженер-программист	6000	546000
4.1	Разработка модулей интеграции	15.12.2023	62	15.02.2024	Инженер-программист	6000	372000
4.2	Разработка бота	16.02.2024	29	16.03.2024	Инженер-программист	6000	174000
5.0	Отладка	17.03.2024	10	27.03.2024	Инженер-программист	6000	60000
5.1	Отладка кода модулей	17.03.2024	10	27.03.2024	Инженер-программист	6000	60000
6.0	Тестирование	28.03.2024	32	01.05.2024	Тестировщик	5500	176000
6.1	Альфа-тест	28.03.2024	10	07.04.2024	Тестировщик	5500	55000
6.2	Отчет по итогам альфа-теста	08.04.2024	5	13.04.2024	Тестировщик	5500	27500
6.3	Бета-тест	14.04.2024	11	25.04.2024	Тестировщик	5500	60500
6.4	Отчет по итогам бета-теста	25.04.2024	6	01.05.2024	Тестировщик	5500	33000
7.0	Оформление документации	02.05.2024	30	01.06.2024	Тех.писец	4000	120000
7.1	Заполнение отчетности	02.05.2024	30	01.06.2024	Тех.писец	4000	120000
Примерная стоимость проекта:		01.09.2023	259	01.06.2024			1382000

Рисунок 1.1 — Диаграмма затрат

Так же для наглядности временных затрат была использована «диаграмма Ганта», позволяющая наглядно отследить каждый этап разработки информационной системы. Диаграмма Ганта представленна на рисунке В.1.

2 Проектирование информационной системы

2.1 Концептуальное проектирование

Для представления о работе системы была разработа концептуальная модель итогового приложения, представленная на рисунке 2.1.

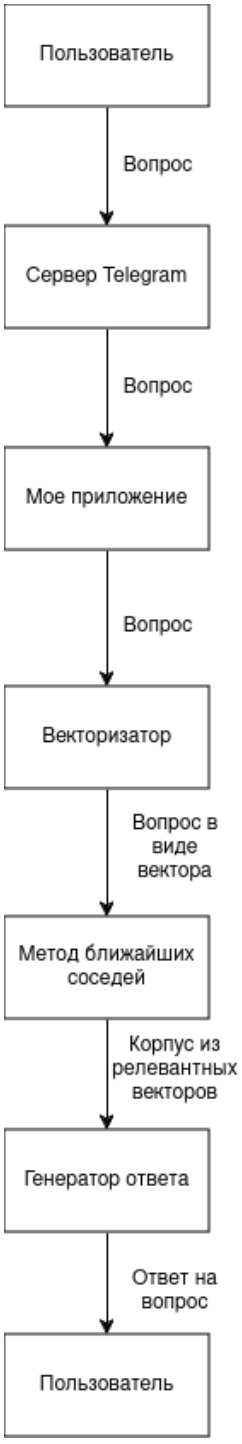

Рисунок 2.1 — Концептуальная модель

Для отображения функциональности системы используется диаграмма вариантов использования, представленная на рисунке 2.2.

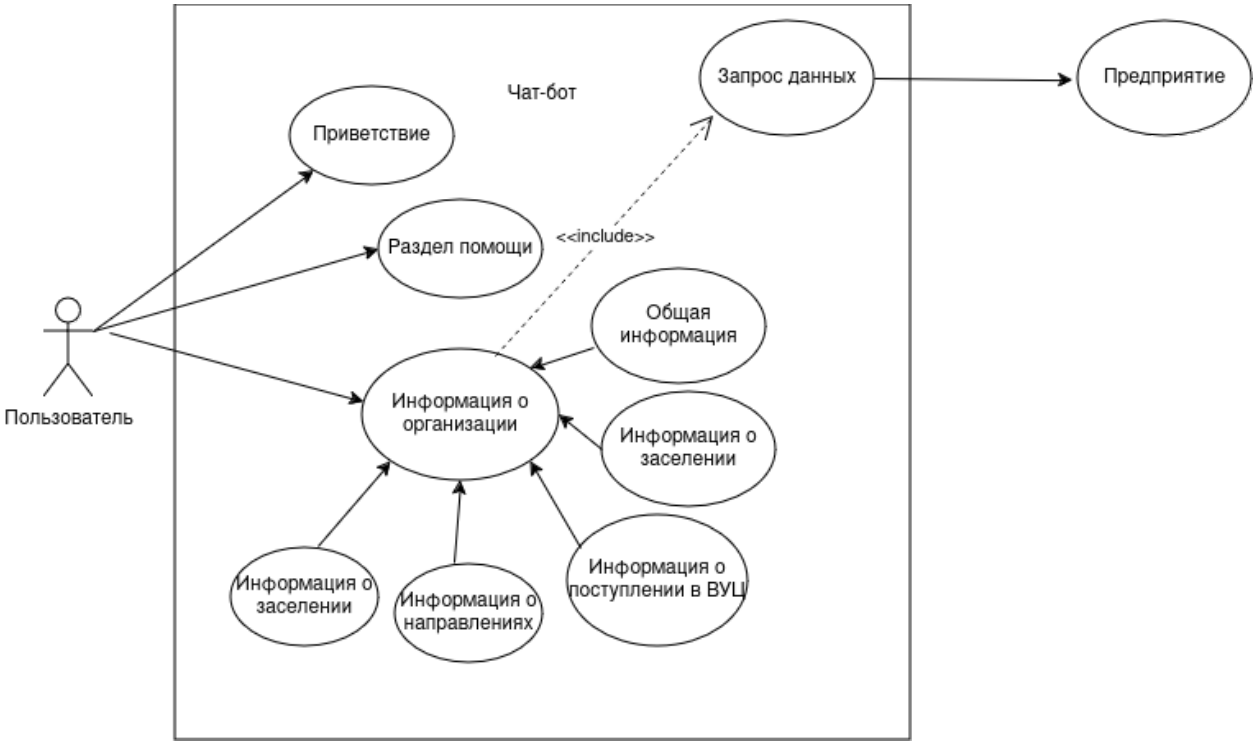

Рисунок 2.2 — Диаграмма вариантов использования

Актор *Пользователь* является обобщением клиентов, которые использует систему. Прецедент Приветствует отражает требование FUN_1. Этот и другие прецеденты описывают субъект Бот, который на диаграмме выделен рамкой. Прецедент Рассказывает о возможностях отражает требование FUN_1. Актор Образовательная организация является источником данных, которые необходимы субъекту. Прецедент *Информация о организации* отражает функциональные требования: FUN_2, FUN_3, FUN_4, FUN_5, FUN_6, FUN_7, и используется для логического объединения других прецедентов, для уменьшения количества связей на диаграмме и облегчить ее восприятие.

Для понимания перемещения данных внутри приложения была разрабработана схема перемещения данных, представленная на рисунке 2.3.

Рисунок 2.3 — Схема перемещения данных

2.2 Диаграмма компонентов

Диаграмма компонентов описывает физическое представление системы и является структурной диаграммой языка унифицированного моделирования. Она определяет архитектуру разрабатываемой системы, устанавливая зависимости между программными компонентами. Кроме того, она предоставляет разработчикам и архитекторам общую картину архитектуры си-

стемы, помогает им лучше понимать ее структуру и взаимосвязи, а также является полезным инструментом для коммуникации и документирования архитектурных решений. Разработанная диаграмма представлена на рисунке 2.4.

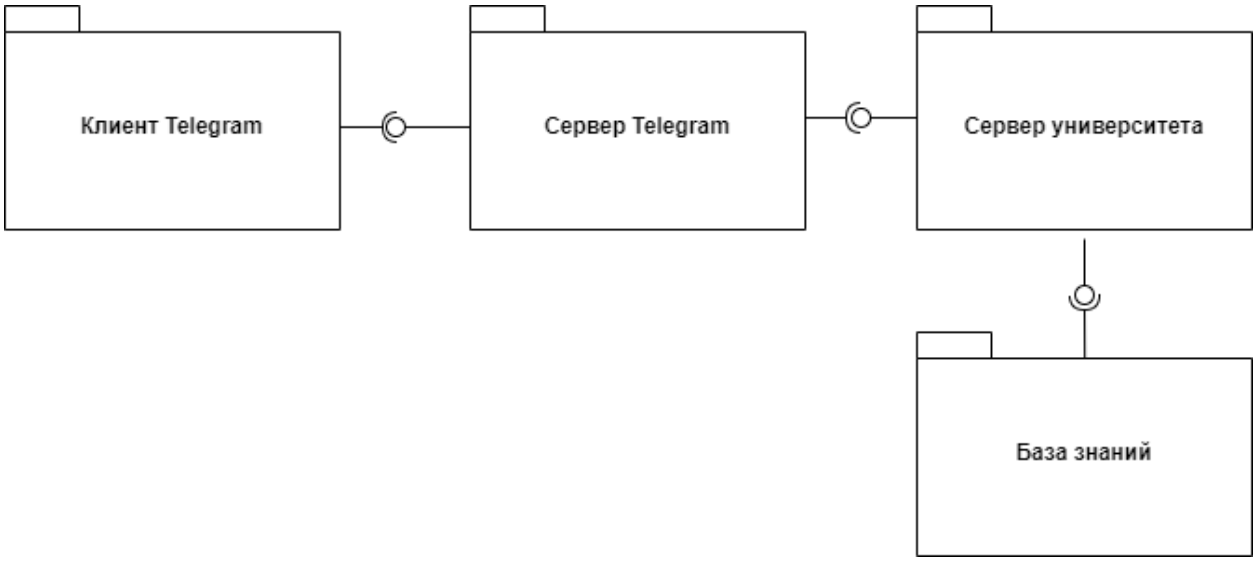

Рисунок 2.4 — Диаграмма компонентов

2.3 Диаграмма развертывания

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их. Для каких задач строят диаграммы развертывания:

1) Визуализация полной структуры исходного кода
2) Визуализация узлов системы для определения слабых мест
3) Обеспечение многократного использования отдельных фрагментов программного кода

Диаграмма разрабатываемой информационной системы содержит несколько узлов: сервер базы знаний внутри университета, сервер приложения, сервер мессенджера, на котором базируется бот, конечный аппарат, при помощи которого будет производиться взаимодействие с информационной системой. Наблюдать разработанную диаграмму развертывания можно на рисунке 2.5.

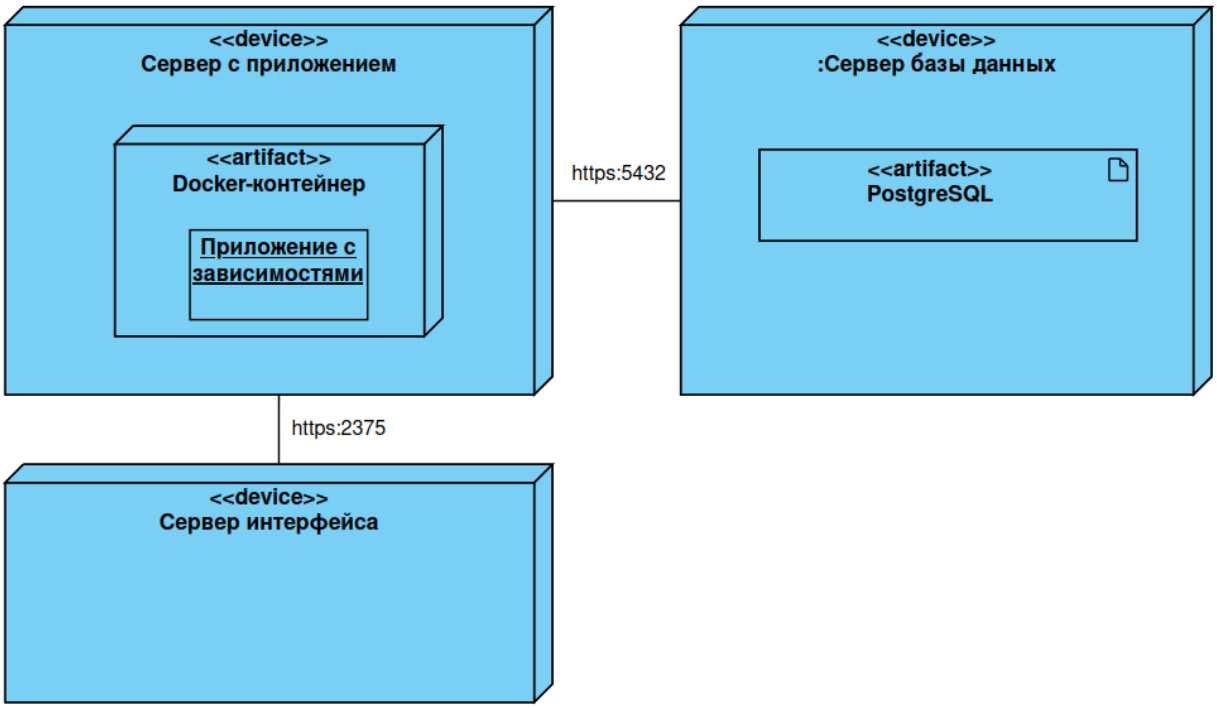

Рисунок 2.5 — Диаграмма развертывания

2.4 Диаграмма последовательности

Пайплайн действий таков: пользователь вводит запрос в текстовом формате или формате аудио, если сообщение передано в аудиоформате, то происходит расшифровка аудиозаписи, после чего полученный текст преобразуется в векторное представление, далее происходит извлечение подходящих документов методом ближайших соседей (KNN) из базы знаний, сформированной из документов в базе данных. Листинг поведения для различных входящих данные приведен в приложении Д.

Далее, наиболее релевантные документы подмешиваются к запросу языковой модели, после чего языковая модель генерирует ответ, опираясь на полученный контекст.

Последовательность работы приложения приведена в приложении А.1.

2.5 Схема базы данных

В качестве базы данных для хранения изначальных документов для построения retirivial использовалась система управления базами данных (СУБД) PostgreSQL. Данное решение обладает рядом преимуществ: высокая произ-

водительность при работе с большими объемами информации; PostgreSQL является свободным программным обеспечением, что позволяет использовать данное ПО для любых нужд на бесплатной основе; удобство использования и возможность удобного переноса данных из других баз данных. Данные пункты стали ключевыми при выборе СУБД для хранения информации.

Такой выбор СУБД позволяет обеспечить высокую отказоустойчивость. Схема хранения данных внутри базы данных указана на рисунке 2.6

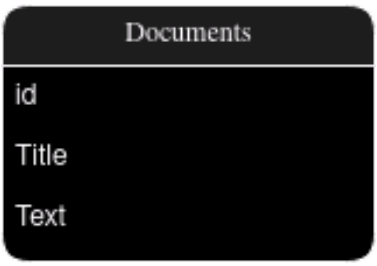

Рисунок 2.6 — Схема хранения документов

2.6 Векторная база данных

Для хранения векторов нужна особая база данных: необходимо быстро находить релевантные вектора и нужно обрабатывать данные любой размерности. Для решение этой задачи было выбрано решение от Meta (признана в РФ экстремистской организацией и запрещена) — FAISS. FAISS является легковесным решением, что идеально кореллирует с моим проектом, в котором не будет огромного количества векторов; не требует большого количества ресурсов; внутри используется хороший поисковый движок; можно использовать ресурсы как GPU, так и CPU, что является плюсом при отсутствии мощного GPU, или для разделения нагрузки при использовании SLM. Схема векторизации документов приведена на рисунке 2.7.

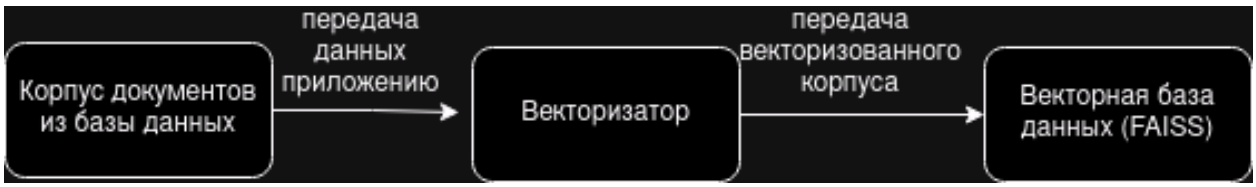

Рисунок 2.7 — Схема векторизации документов

2.7 Развертывание приложения

Планируется, что приложение будет размещаться на мощностях образовательного учреждения. Для упрощения внедрения было принято решение разработать скрипт для контейнеризации приложения.

В качестве базового образа был выбран образ с GNU Linux в качестве операционной системы. GNU Linux является свободным программым обеспечением, что позволяет использовать его для любых нужд. На листинге 2.1 отображена конфигурация Dockerfile.

Листинг 2.1 — Dockerfile

```
1  FROM python:3.12.3-bullseye
2
3  WORKDIR /app
4
5  COPY . .
6
7  RUN pip install -r --no-cashe-dir req.txt
8
9  RUN apt-get update && apt-get install -y nano
10 RUN apt-get update && apt-get install -y wget bzip2
   ↪ libxtst6 libgtk-3-0\
11  libx11-xcb-dev libdbus-glib-1-2 libxt6 libpci-dev
12
13 CMD ["/bin/bash", "-c", "python main.py"]
```

Данный файл позволяет создать контейнер, содержащий все необходимые необходимые для работы зависимости. Использование изолированного окружения, выделенных ресурсов позволяют сервису автономно функционировать на протяжении долгого времени без необходимости обслуживания, что соответствует нефункциональному требованию REL_1. Так же созданная конфигурация для создания изолированного контейнера позволяет запустить приложение и все необходимые зависимости при помощи одной команды, что отражает нефункциональное требование по отношению к поддержке сервиса SUP_1.

3 Разработка системы

3.1 Выбор средств разработки

В качестве языка программирования для разработки системы был выбран Python. Выбор связан с набором ПО, необходимым для функционирования сервиса, который был определен на последнем этапе проектирования. Также Python является наиболее подходящим языком для работы с искусственным интеллектом, для него имеется много библиотек, нацеленных на решение конкретных проблем.

В качестве системы управления версиями был выбран Git. В качестве системы управления репозиториями был выбран GitHub. С недавнего времени GitHub располагает возможностью для создания бесплатных приватных репозиториев — никто, кроме авторизовазанных пользователей не будет иметь доступа к вашему репозиторию.

В качестве системы для написания кода был выбран текстовый редактор NeoVim. NVim ориентирован на разработку без мыши, что позитивно влияет на скорость разработки; можно запустить без графического окружения рабочего стола, что позволяет использовать данный текстовый редактор как на домашнем персональном компьютере, так и на сервере, не имеющим графической оболочки. Конфигурация приведена на листинге Б.1.

Для верстки PDF-документа был использован LaTeX. С использованием пакетов данный инструмент позволяет достичь отличных результатов для написания текстовых документов. Так же данное программное обеспечение является свободным. Частичная конфигурация LaTeXбудет приведена в приложении Г.

3.2 Структура приложения

Созданное приложение имеет модульную структуру. Это позволяет распределить зоны ответственности каждого модуля [4]; к тому же это позволяет повысить отказоустойчивость конечного приложения.

Схема файловой структуры показана на листинге 3.1.

Листинг 3.1 — Файловая структура

```
|-- Dockerfile
|-- documents
|------about.txt
|------checking.txt
|--  common_questions.txt
|--  deductions_recoveries.txt
|-- embeding_store
|------index.faiss
|------index.pkl
|-- LICENSE
|-- main.py
|-- messages.log
|-- README.md
|-- req.txt
|-- retriveval.py
|-- speech2text.py
|-- tex
|-- text_generator.py
|-- videohandler.py
```

Файл *Dockerfile* содержит инструкцию для сборки контейнера и последующего запуска приложения внутри контейнера.

Директория *documents* содержит основные документы в текстовом формате, предназначенные для дальнейшего перевода их в векторное представление. Документы представляют собой файлы в формате .txt, хранящие в себе информацию о предприятии, например, информацию о военном учебном центре, которые в дальнейшем будут использоваться для создания векторного пространства. По мере развития проекта данный список может меняться.

Директория *embeding_store* содержит в себе: созданное векторное пространство, метаданные, необходимые для поиска, индексы векторов, структуры данных, параметры.

Файл *main.py* запускает основное приложение: запуск retrivial, speech2text, запуск логирования, а так же телеграмм клиента.

Файл *retrivial.py* несет в себе следующий смысл: перевод корпуса текстов в векторное представление, извлечение 3х наиболее релевантных документов для составления контекста, а также обогащение запроса пользователя

извлеченной информацией, для составления ответа.

Файл *speech2text.py* представляет собой блок расшифровки в аудио в текстовый формат, для использования блоком *retrivial.py*, если это потребуется.

Файл *videohandler.py* представляет собой блок разделения видео- и звуковых дорожек. Извлеченная звуковая дорожка используется блоком *speech2text.py*.

3.3 Расчет надежности программного и аппаратного обеспечения

Для расчета надежности реализуемого продукта необходимо выявить элементы системы, которые могут выйти из строя. Для себя я поделил их на 3 категории, и составил таблицы 3.1, 3.2 и 3.3. Программная часть, аппаратная часть, независящие от обстоятельств. Данные для столбца 'Коэффициент надежности' брались из открытых источников.

Таблица 3.1 — Аппаратная часть

Номер	Коэффициент надежности	Краткое описание	Что повлечет?
1	0.99	Выход из строя жесткого диска (HDD)	Поломка HDD влечет за собой: потерю данных, простой системы.
2	0.99	Выход из строя процессора (CPU)	Поломка CPU влечет за собой: простой системы; если мы используем сервер, то потерю производительности; большие траты на замену и обслуживание
3	0.87	Выход из строя графического процессора (GPU)	Выход из строя GPU грозит: снижением производительности или полным выходом из строя системы, большие затраты на замену и обслуживание

Продолжение таблицы 3.1

Номер	Коэффициент надежности	Краткое описание	Что повлечет?
4	0.85	Выход из строя оперативной памяти	Выход из строя модуля оперативной памяти не так критичен для системы, модулей памяти несколько и вышел один из них, но если сломались все плашки оперативной памяти, то система не сможет функционировать.
5	0.95	Выход из блока питания	Выход из строя блока питания сулит полный выход строя системы

Таблица 3.2 — Программная часть

Номер	Коэффициент надежности	Краткое описание	Что повлечет?
1	0.93	Выход из строя модуля, ответственного за перевод звука в текст	Выход из строя данного модуля не полечет за собой больших проблем для обычных пользователей, так как будет доступен обычный текстовый ввод, но для людей с ограниченными возможностями перестанет быть возможным использование моей системы

Продолжение таблицы 3.2

Номер	Коэффициент надежности	Краткое описание	Что повлечет?
2	0.84	Выход из строя модуля, ответственного за перевод документов в векторное представление	Поломка данного модуля может делиться на две категории: перестает работать перевод текста в векторное представление и перестает работать извлечение векторов. В первом случае не все так критично, так как часто обновлять список актуальных документов не нужно, ведь большая часть информации не меняется. Если же сломается модуль, ответственный за извлечение векторных представлений, то система в большей своей части перестанет работать, как задумывалось, но все равно будет работать: языковая модель будет выдавать ответы, но не будет иметь контекста о университете, что негативно повлияет на точность ответа.
3	0.77	Выход из строя модуля клиентского приложения	Выход из строя клиентского приложения сулит собой: репутационные риски, недоступность приложения для пользователей

Таблица 3.3 — Стохастичные переменные

Номер	Коэффициент надежности	Краткое описание	Что повлечет?
1	0.99 – 0.97	Массовое отключение интернета	Без подключения к сети интернет система полностью перестает работать. Это можно будет решить, если поднять локальную сеть
2	0.99 – 0.97	Отключение электричество	Без электричества система полностью перестает работать

Расчитывая стохастические переменные я отталкивался от того, что выключить интернет или электричество на сутки могут от 2 до 10 раз

Так как моя система избыточна: множество модулей позволяют работать другим модулям при выходе из строя, то я буду считать отказоустойчивость системы по следующей формуле: $R_{total} = 1 - (1 - R_1)(1 - R_2)\dots(1 - R_n)$, где R_{total} — общая отказоустойчивости системы, R_n — отказоустойчивость компонента системы.

По расчетам надежность аппаратно-программного комплекса $\approx 1 - 2.5116e-14$, что я считаю довольно хорошим показателем для такой системы.

3.4 Расчет ожидаемого результата экономической эффективности

Основными затратами для реализации информационной системы являются:

– покупка необходимого аппаратного обеспечения для обеспечения работы языковой модели

– в случае, если предприятие не хочет или не может позволить аппаратное обеспечение для развертывания системы на своих мощностях, то оформление подписки у предприятий и аренда мощностей для получения доступа к языковой модели

– обслуживание сервера

Предполагается несколько концепций для получения прибыли от ко-

нечного продукта: Продажа подписки коммерческим и некомерческим организациям, по предоставлению конечного продукта. Для некоммерческих организаций подписка будет стоить от 5000 рублей в месяц, до 15000 рублей в месяц. В свою очередь для коммерческих организаций планируется разрабатывать уникальный договор по оформлению подписки на конечный сервис, для получения максимальной выгоды от сделки.

К подписочной модели получения прибыли так же не исключается возможность продажи программного обеспечения. В таком случае цена за конечный продукт так же определяется уникальным договором. В таком случае конечный потребитель не платит за подписку и сам обслуживает всю систему, но имеет над ней полный контроль. Со стороны продавца передается документация к программному обеспечению.

Для расчета возврата инвестиций используется формула 3.1:

$$ROI = \frac{PROFIT - COST}{COST} \cdot 100\% \tag{3.1}$$

Если вести подсчет для реального случая, то в худшем случае получается ситуация, приведенная в формуле 3.2:

$$ROI = \frac{60000 - 15000}{15000} \cdot 100\% \approx 300\% \tag{3.2}$$

что равно 300%. Но в эти цифры не заложена покупка аппаратного комплекса, так как рассматривается ситуация, что я у предприятия есть мощности для размещения малой языковой модели.

3.5 Word2vec

Для хранения представлений документов в понятном для машины виде используется подход word2vec. Подход word2vec имеет две основные реализации: Continuous Bag of Words (CBOW) и skip-gram.

Подход CBOW работает следующим образом: на вход системе подается несколько слов из контекста, где количество слов задается параметром window size; скрытый слой принимает полученные векторные представления слов, складывает их и усредняет; на выходе получается вероятностное распределение для целевого слова. Обучение данной модели заключается в

минимизации функции потерь между предсказанным и истинным словом. Для этого применяется градиентный спуск. Визуализацию работы градиентного спуска можно наблюдать на рисунке 3.1.

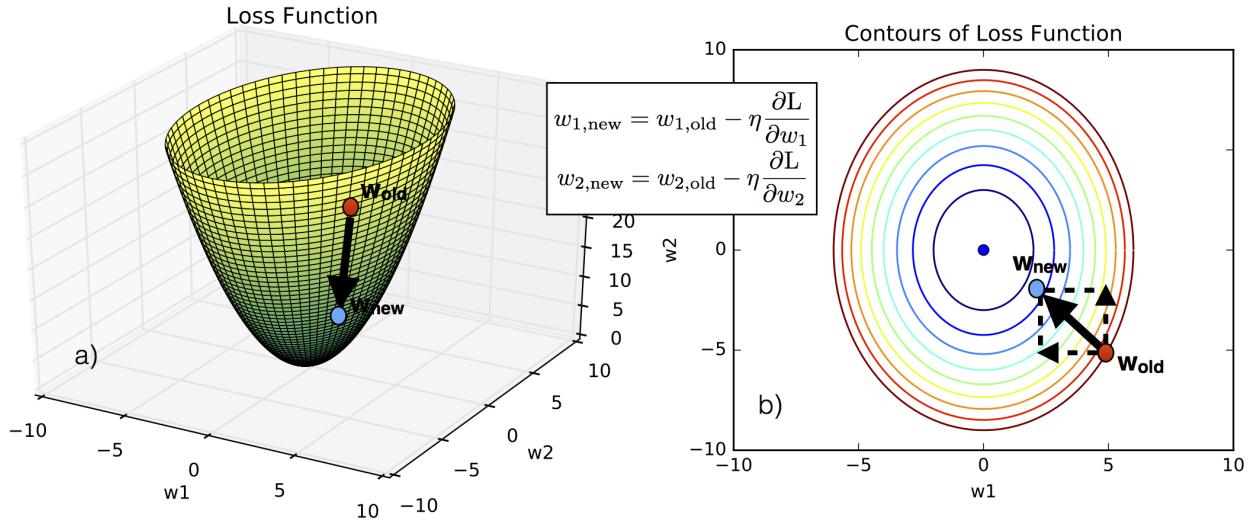

Рисунок 3.1 — Визуализация работы градиентного спуска

Skip-gram работает иначе, чем CBOW[5]: на вход подается единственное слово, после чего скрытый слой преобразует данное слово в вектор; на выходе получаем матрицу, содержащую вероятность появления какого-либо слова рядом с ним. Как и в случае с CBOW обучение представляет собой алгоритм обратного распространения ошибки.

Преобразование текстовых корпусов в вектора работает следующим образом: после передачи на вход программе текста программа создает для каждого слова вектор, после чего происходит процесс обучения. Во время обучения происходит попытка предсказать контекст слова, основываясь на его окружении. Например, если в тексте явно кореллируют слова “упряжка“ и “конь“, то эти векторные представления данных слов будут иметь сходство. Визуализацию данного свойства можно наблюдать на рисунке 3.2

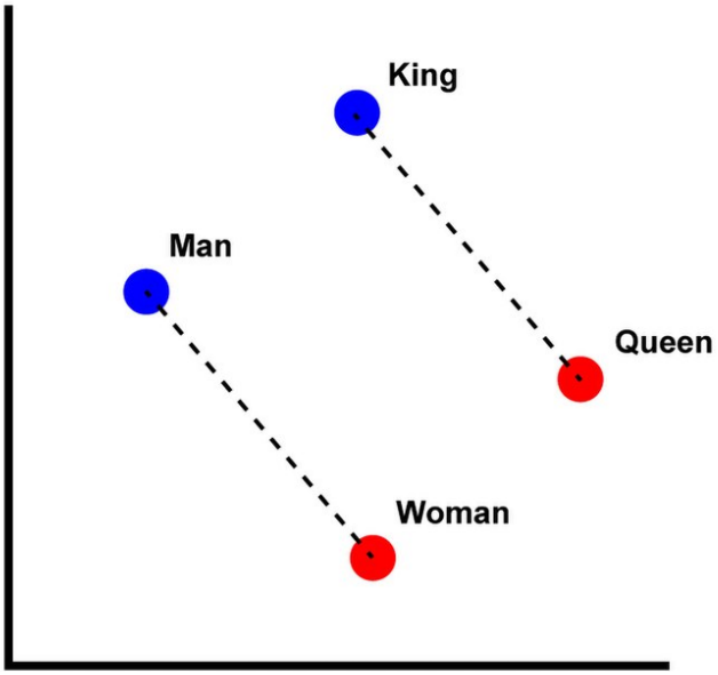

Рисунок 3.2 — Взаимосвязь векторов слов

В конце обучения создается векторное пространство, используемое в дальнейших этапах работы приложения.

Так как большинство решений не оптимизированы для применения на домашних персональных компьютерах (ПК) было принято решение переопределить некоторые методы векторизатора из библиотеки HuggingFace, чтобы улучшить работу на своем ПК. На листинге 3.2 отображены переопределенные методы. Созданое решение отражает нефункциональное требование, относящееся к поддержке сервиса — SUP_2.

Листинг 3.2 — Векторизатор

```
class HuggingFaceE5Embeddings(HuggingFaceEmbeddings):
    def embed_query(self, text: str) -> List[float]:
        text = f"query: {text}"
        return super().embed_query(text)

    def embed_documents(self, texts: List[str]) ->
    ↪ List[List[float]]:
        texts = [f"passage: {text}" for text in texts]
        return super().embed_documents(texts)

    async def aembed_query(self, text: str) ->
    ↪ Coroutine[Any, Any, List[float]]:
        text = f"query: {text}"
        return await super().aembed_query(text)

    async def aembed_documents(
        self, texts: List[str]
    ) -> Coroutine[Any, Any, List[List[float]]]:
        texts = [f"passage: {text}" for text in texts]
        return await super().aembed_documents(texts)
```

3.6 Поиск подходящих векторов

Созданное векторное пространство делится на множество кластеров методом K-средних. Для каждого кластера выставляется центроида для определения сущности каждого корпуса векторов. Это позволяет не перебирать все вектора, а проходиться по принадлежности к какой-то сущности. При большом наборе данных это позволяет быстрее находить необходимый вектор. На рисунке 3.3 схематично отображен алгоритм кластеризации.

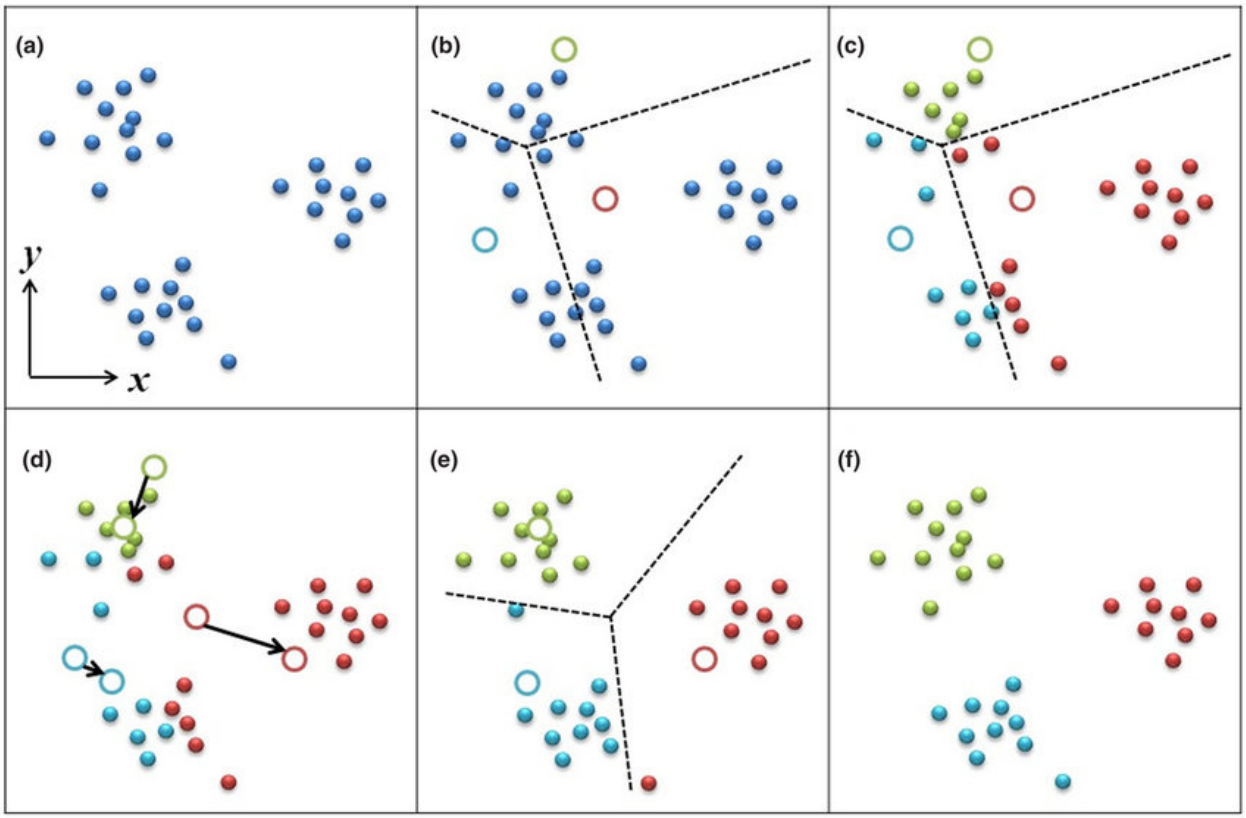

Рисунок 3.3 — Визуализация кластеризации векторов

Наиболее релевантные вектора [3] сопоставляются при помощи метода K-ближайших соседей. Евклидово расстояние между векторами вычисляется по приведенной формуле 3.3:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3.3}$$

Так же для поиска похожих векторов используется косинусное расстояние 3.4:

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{3.4}$$

И хотя данный подход не покрывает все возможные случаи [7], он является наиболее универсальным.

Визуализацию работы поиска релевантных векторов можно на рисунке 3.4

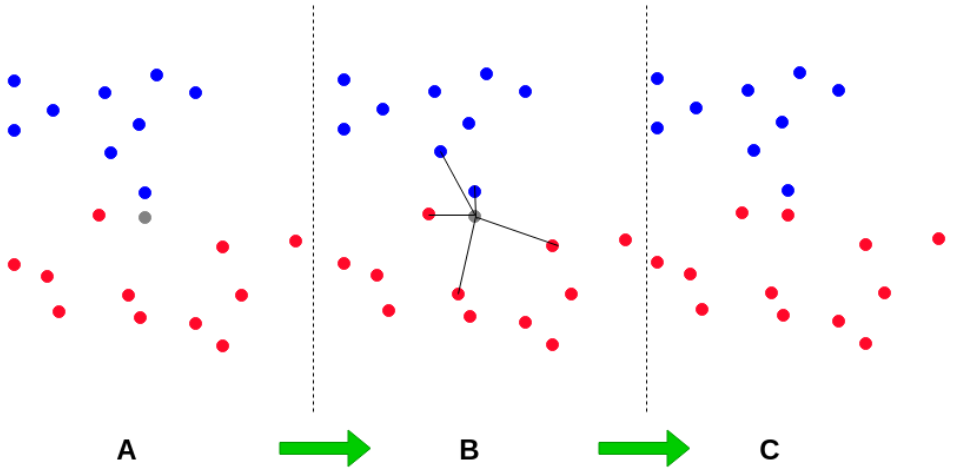

Рисунок 3.4 — Визуализация нахождения релевантных векторов

3.7 Генерация текста при помощи поиска и аугментаций

Для работы системы необходимо подать на вход какой либо текст. В качестве клиента было выбрано клиентское приложение Telegram, а в качестве интерфейса для взаимодействия приложения и пользователя TelegramBotAPI, но в качестве клиента можно использовать и другие платформы: ВК, Яндекс.Алиса, Одноклассники и прочие. За последнее время боты стали очень популярным решением, [6] благодаря удобности и доступности, что позволяет бизнесу удобно использовать подобные интерфейсы в своих интересах. Чтобы пользователю было удобно использовать созданное приложение было принято описать идею приложения, а так же дать ссылку на вызов функции-помощника. Данный аспект отражает нефункциональные требования NFR_1 и NFR_2.

На листинге 3.3 отображена реализация приветственного сообщения от бота.

Листинг 3.3 — Стартовое сообщение

```
    dummy_message = "Ищу ответ..."
    stub = hbold("Бот работает в тестовом режиме - не все ответы могут быть достоверными")
    # Start message
    @dp.message(CommandStart())
    async def command_start_handler(message: Message) -> None:
        """
        This is handler for command `/start`
        """
        await message.answer(
            f"Привет, {hbold('студент')}\n\nПеред тобой бот-асситент ИИСИГТ,\
     подготовленный {hlink('мной', 'https://github.com/yaz0p')} к качестве\
     выпускной квалификационной работы\n\nДанный бот позволяет осуществлять\
     удобную навигацию внутри университета и отвечает на не самые очевидные\
     вопросы. \n\nБот работает на основе большой языковой модели, а\
     информацию о внутренних источниках получает благодаря {hbold('RAG')}.\
     Возможно, я прикреплю сюда git репозиторий с реализацией проекта,\
     чтобы студенты могли изучить работу данного ассистента и/или дополнить\
     его своими идеями.\n\n\
    Пример того, что ты можешь спросить:\n- Адрес какого либо-корпуса\n\
    - Номер приемной комиссии\n- Как поступить в ВУЦ\n- Время приема ректората\
     - Информацию о заселении и медкомиссии\n"
        )
        sleep(2)
        await message.answer(
            f"Для получения инструкции о взаимодействии с\
            ассистентом нажмите /help\n\n\
    {hbold('Внимание, все сообщения логгируются!')}"
        )
```

После получения сообщения сервером Telegram и передачей сообщения приложению по протоколу HTTPS, что отражает нефункциональное требование к безопасности SAF_1, происходит перевод полученного текста в вектор и сопоставление полученного вектора с векторами из созданного векторного пространства базы знаний. После поиска подходящих векторов происходит обратная трансформация из векторного пространства в текстовое представление.

На листинге 3.4 отображен процесс создания контекста и промт для корректной работы языковой модели. Для работы системы выбираются 3 релевантных вектора, так как это позволяет найти необходимые вектора без избыточности и снизить нагрузку на сервис, и к тому же снизить расход токенов для LLM или SLM.

Листинг 3.4 — Функция для поиска векторов и создания контекста

```
1     def augment_prompt(self, query: str,
      ↪  _embedings_storage=_embedings_storage):
2         results =
          ↪  _embedings_storage(self).similarity_search(query,
          ↪  k=3)
3         source_knowledge = "\n".join([x.page_content for x
          ↪  in results])
4         augmented_prompt = f"""Используя предоставленный
          ↪  контекст ответь на следующий вопрос.
5
6         Контекст:
7         {source_knowledge}
8
9         Вопрос: {query}"""
10        return augmented_prompt
```

Далее обогащенный текстовый запрос подается на вход языковой модели, после чего проиходит генерация текстовых токенов декодером [1]. На листинге 3.5 отображен модуль, ответственный за генерацию текста.

Листинг 3.5 — Функция для генерации ответа на заданный вопрос

```
        def answer(self, message: Message) -> None:
            prompt = PromptTemplate(
                template="Ответь на поставленный вопрос: {subject}",
                input_variables=["subject"],
            )

            output = self.chat(
                [
                    SystemMessage(
                        content="""Ты профессиональный ассистент Российского государственного гидрометеорологического университета, который помогает студентам и преподавателям решать их проблемы: навигация внутри университета, ответ на общие вопросы, касающиеся РГГМУ. Если контекст большой, то используй его по максимуму, но не теряй сути. При ответе на вопросы общайся в деловом стиле и пиши только по существу. """
                    ),
                    HumanMessage(
                        content=prompt.format(
                            subject=self.
                            augmentation.
                            augment_prompt(message)
                        )
                    ),
                ]
            )


            return output.content
```

На рисунке 3.5 можно наблюдать работу поиска и аугментации внутри клиента Telegram.

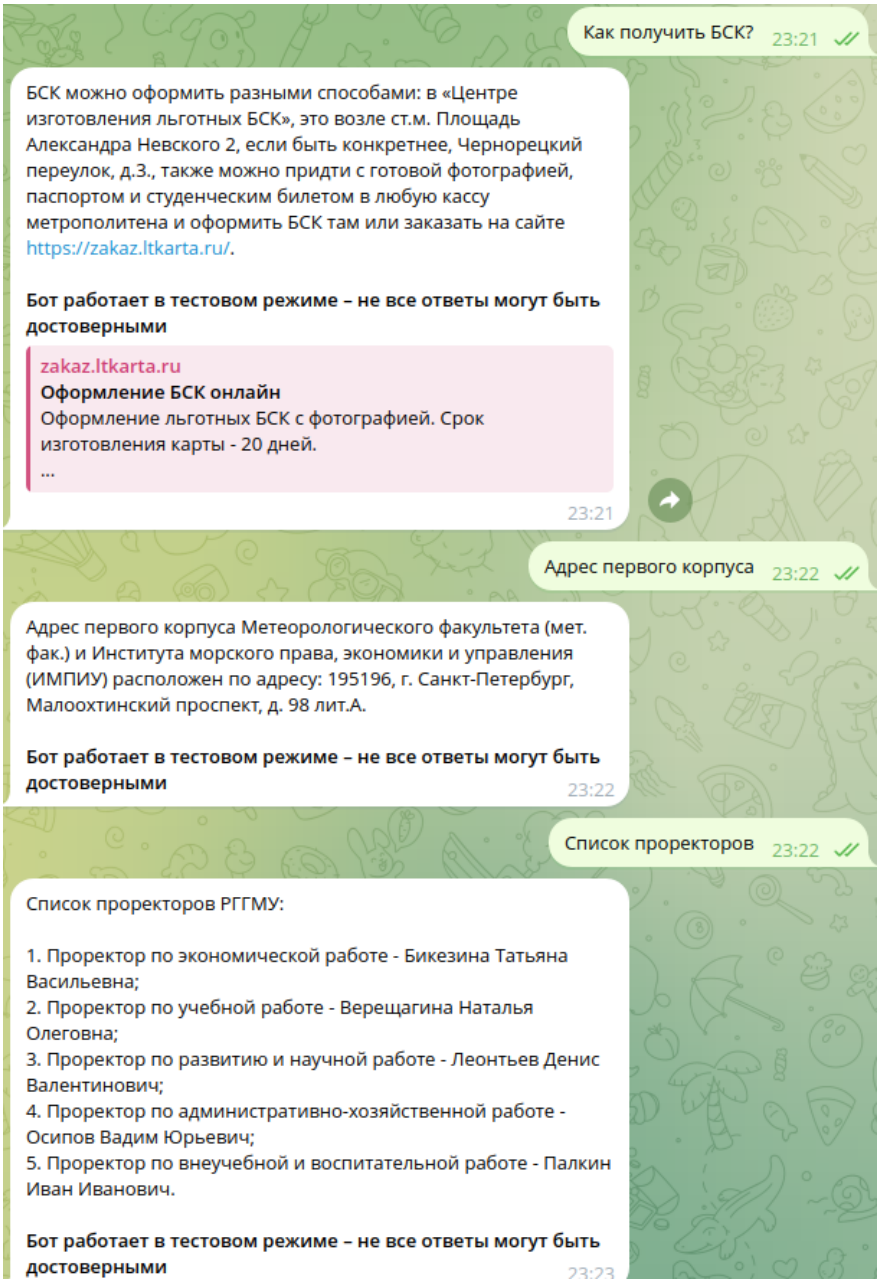

Рисунок 3.5 — Работа приложения

3.8 Распознавание голоса

Для пользователей предусмотренна возможность ввода как текстового сообщения, так и управления ботом при помощи голоса или видео. Данная возможность отражает нефункциональное требование NFR_3. Для реализации данного требования использовалась заранее обученная сверточная нейронная сеть из библиотеки SpeechRecogintion. Для восприятия системой был

написан модуль, сохраняющий текстовое представление звука, после чего текст загружается в модуль, отображенный на листинге 3.5. На листинге 3.6 отображен модуль, ответственный за распознавание речи.

Листинг 3.6 — Функция для распознавания речи

```
import speech_recognition as sr
import soundfile as sf
import os

def recognize_speech(audio_file):
    data, samplerate = sf.read(audio_file)
    sf.write('voice.wav', data, samplerate)

    recognizer = sr.Recognizer()

    with open('voice.wav', 'rb') as audio_file:
        with sr.AudioFile(audio_file) as source:
            audio_data = recognizer.record(source)
            recognizer.adjust_for_ambient_noise(source)

        try:
            text = recognizer.recognize_google(audio_data,
            ↪  language='ru-RU')
            os.remove('voice.wav')
            return text
        except sr.UnknownValueError:
            return "Попробуй ещё раз. Не могу разобрать,
            ↪  что ты говоришь."
        except sr.RequestError:
            return "Ну здесь наши полномочия все :("
```

Данное архитектурное решение позволяет уменьшить количество повторов при написании кода и разделить ответственность между участками приложения.

Работу модуля распознавания голоса и преобразования его в текст можно увидеть на рисунке 3.6.

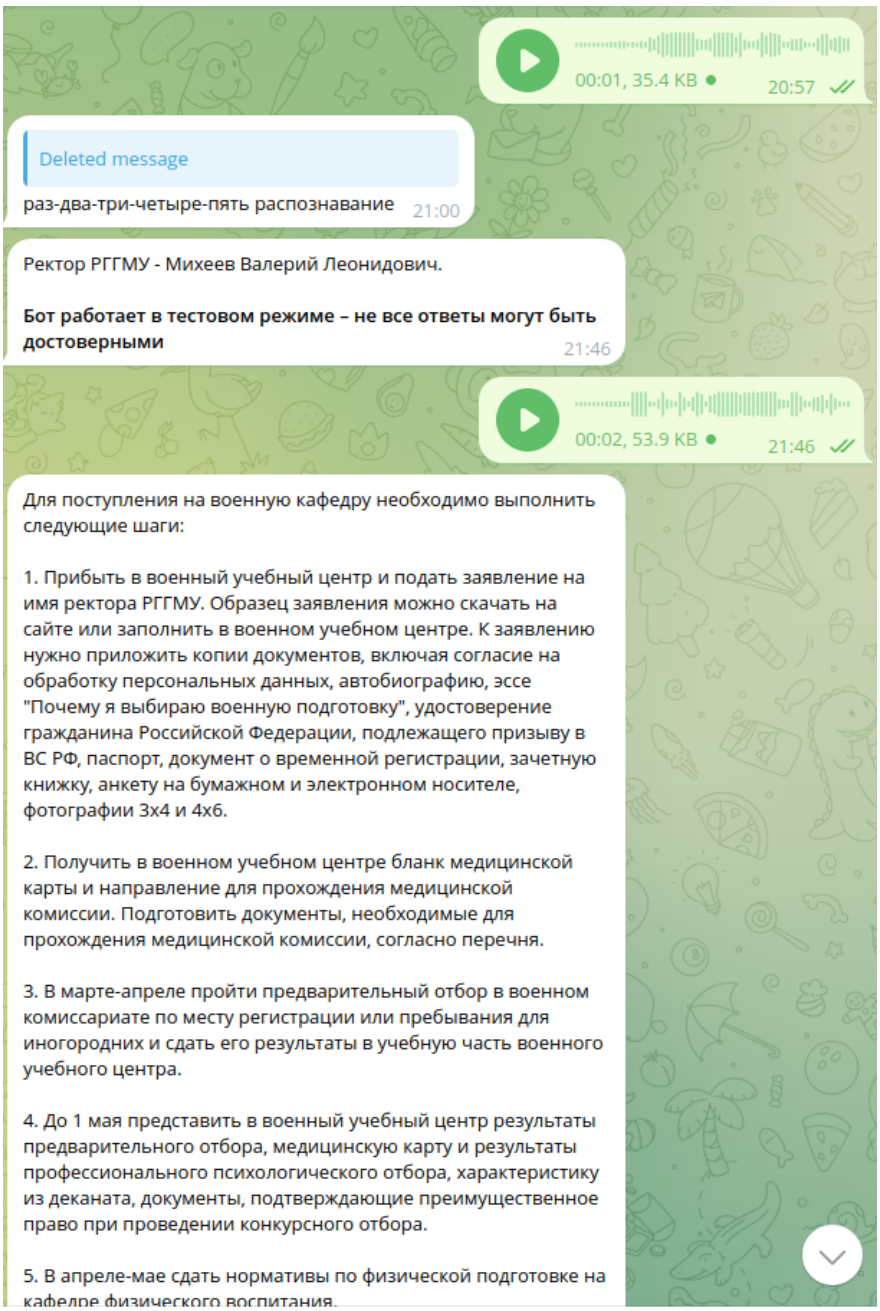

Рисунок 3.6 — Преобразование голосового сообщения в текстовый запрос

3.9 Распознавание видео

Так же для приложения был написан модуль для распознавания звука внутри видеофайлов. Планируется доработать приложения, для поддержки управления глухонемыми людьми. В настоящий момент модуль распознавания видеофайлов работает по аналогии с модулем распознавания аудиофайлов — видео и аудио дороги разделяются, после чего аудиодорожка распознается модулем распознавания голосовых сообщений. Работа данного модуля приведена на изображении 3.7.

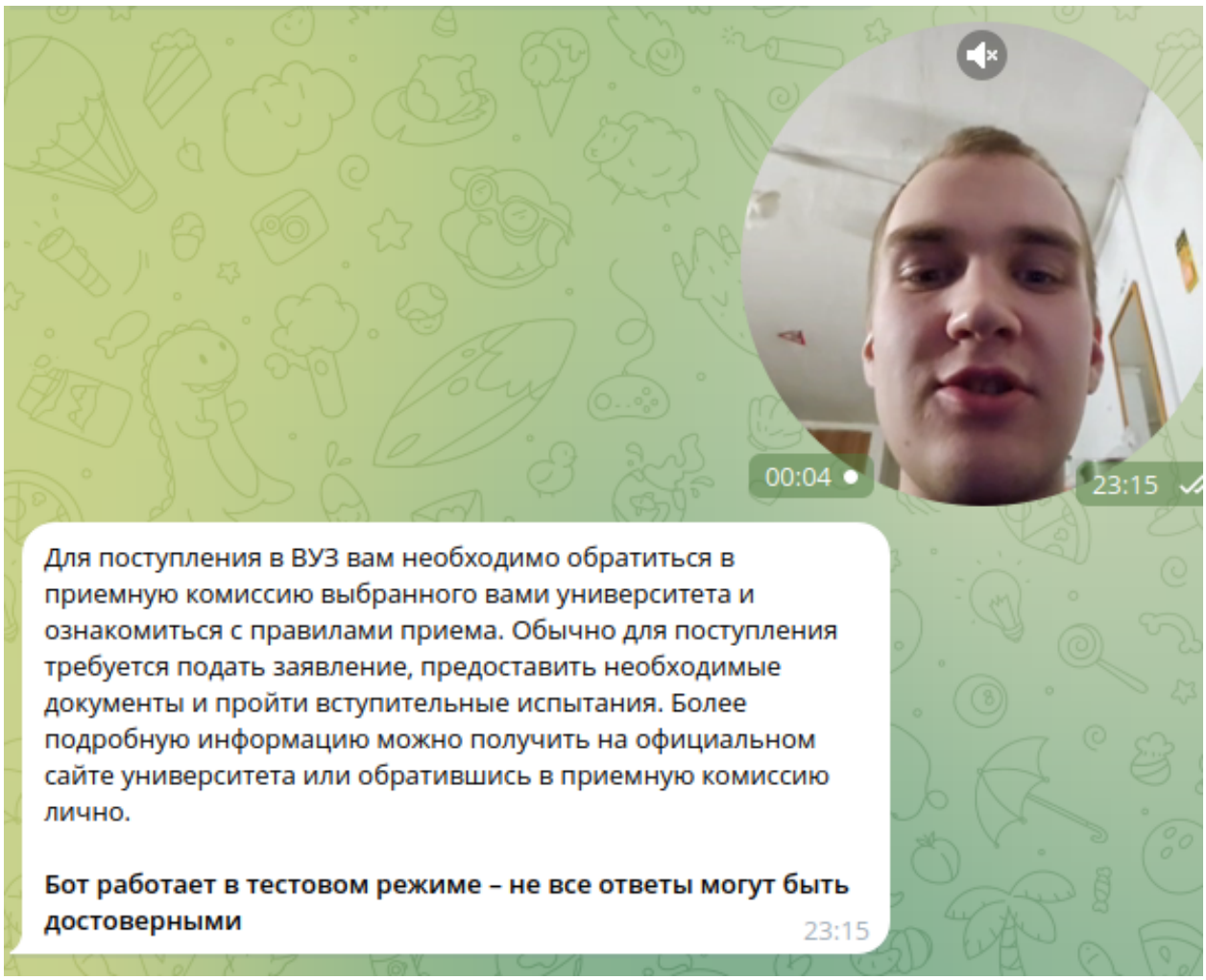

Рисунок 3.7 — Преобразование видеосообщения в текстовый запрос

Отрывок кода, ответственный за разделение видеосообщения на звук и видео приведен на листинге 3.7.

Листинг 3.7 — Разбиение видео и аудио

```
1    import moviepy.editor as mp
2    import os
3    def video_handler(video_note: str) -> None:
4        clip = mp.VideoFileClip(video_note)
5        clip.audio.write_audiofile("voice.wav")
6        os.remove(video_note)
```

3.10 Промт-инженерия

Для задания корректного поведения работы языковой модели и определения стиля коммуникации с конечным пользователем были разработаны

специальные запросы-подсказки для LLM — промты. Промты отвечают за многие аспекты работы GPT-like моделей:

- Корректность вывода сообщений
- Стиль общения языковой модели
- Точность ответов
- Структура вывода ответа

Промт, написанный для модуля поиска релевантных векторов и обогащения ответа можно наблюдать на листинге 3.8.

Листинг 3.8 — Промт для retrivial

```
1     augmented_prompt = f"""Используя предоставленный
      ↪  контекст ответь на следующий вопрос.
2
3     Контекст:
4     {source_knowledge}
5
6     Вопрос: {query}"""
```

Промт, написанный для задания поведения языковой модели можно наблюдать на листинге 3.9.

Листинг 3.9 — Поведенческий промт

```
        prompt = PromptTemplate(
            template="Ответь на поставленный вопрос:
            ↪ {subject}",
            input_variables=["subject"],
        )

        output = self.chat(
            [
                SystemMessage(
                    content="""Ты профессиональный
                    ↪ ассистент Российского
                    государственного
                    ↪ гидрометеорологического
                    ↪ университета,
                    который помогает студентам и
                    ↪ преподавателям решать их
                    проблемы: навигация внутри
                    ↪ университета, ответ на общие
                    вопросы, касающиеся РГГМУ. Если
                    ↪ контекст большой, то
                    используй его по максимуму, но не теряй
                    ↪ сути.
                    При ответе на вопросы общайся в деловом
                    ↪ стиле и пиши
                    только по существу. """
                ),
                HumanMessage(
                    content=prompt.format(
                        subject=self.
                        augmentation.
                        augment_prompt(message)
```

3.11 Логирование

Для отлавливания ошибок был реализован механизм логирования. Обработанные системой сообщения сохраняются в следующем виде: уникальный идентификатор пользователя; имя пользователя; введенное пользователем сообщение и ответ, полученный от системы. Данный механизм позволяет

проверять корректность работы системы, а при возникновении ошибки или попытке использовать систему не по назначению можно идентифицировать пользователя и связаться с ним. Посмотреть пример сохраненных логов можно на изображении 3.8.

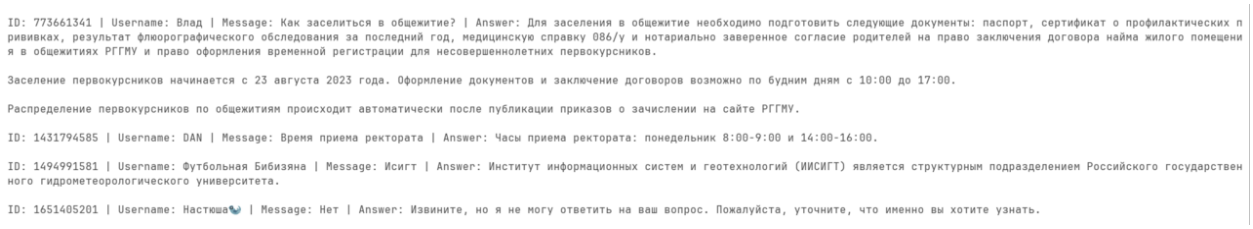

Рисунок 3.8 — Логированные сообщения

3.12 Тестирование

Для проверки корректности приложения было проведено интеграционное тестирование. Результаты тестирования были приведены в таблице 3.4. Работа приложения считается корректной при корректном ответе на более чем 80% вопросов.

Таблица 3.4 — Результаты интеграционного тестирования

№	Вопрос	Ответ
1	Что ты умеешь?	Я могу помочь вам найти информацию о направлениях подготовки в РГГМУ, предоставить примерные размеры стипендий для разных категорий студентови ответить на другие вопросы, связанные с университетом.
2	Что нужно иметь для получения социальной стипендии?	Для получения социальной стипендии необходимо иметь следующие документы: паспорт, СНИЛС, ИНН и справки из соц.защиты. В деканате нужно написать заявление, после чего либо самостоятельно отнести в бухгалтерию на Воронежской 79, либо за вас это делает сам деканат.
3	Как попасть на прием к Палкину	Для записи на прием к Палкину Ивану Ивановичу необходимо зайти на сайт РГГМУ и перейти по ссылке ”https://forms.yandex.ru/u/65323419e010db2119cf9903/”. Там следует выбрать удобное время для посещения и оставить свои контактные данные.
4	Номер телефона студенческого отдела кадров	Номер телефона студенческого отдела кадров РГГМУ - 8(812) 409-13-49.

Продолжение таблицы 3.4

№	Вопрос	Ответ
5	Как оформить бск?	БСК можно оформить разными способами: в «Центре изготовления льготных БСК», это возле ст.м. Площадь Александра Невского 2, если быть конкретнее, Чернорецкий переулок, д.3., также можно придти с готовой фотографией, паспортом и студенческим билетом в любую кассу метрополитена и оформить БСК там или заказать на сайте https://zakaz.ltkarta.ru/.
6	Расписание	Расписание можно найти по данной ссылке: https://www.rshu.ru/university/stud/
7	Направления, связанные с программированием	Направления, связанные с программированием, в РГГМУ: Прикладная информатика, Информационная безопасность телекоммуникационных систем, Бизнес-информатика.
8	Расскажи о РГГМУ	РГГМУ - Российский государственный гидрометеорологический университет, основанный в 1930 году. Университет готовит специалистов в области гидрометеорологии, экологии, океанологии, гидрологии, метеорологии, геофизики, информационных систем и других смежных областях. В университете функционируют различные факультеты, включая метеорологический, гидрологический, экологический, морской права, экономики и управления. РГГМУ активно участвует в научных исследованиях и разработках, сотрудничает с международными организациями и имеет филиалы в различных регионах России.
9	Исигт	Институт информационных систем и геотехнологий (ИИСИГТ) является структурным подразделением Российского государственного гидрометеорологического университета. Он расположен по адресу: 195196, Россия, Санкт-Петербург, проспект Металлистов, дом 3. Директор института - Истомин Евгений Петрович, его электронная почта: dekanat_is@rshu.ru.

Продолжение таблицы 3.4

№	Вопрос	Ответ
10	Как поступить в ВУЦ?	Для поступления в военный учебный центр (ВУЦ) РГГМУ необходимо выполнить следующие шаги: 1. Прибыть в ВУЦ после соответствующего объявления на сайте и подать заявление на имя ректора РГГМУ. Образец заявления можно скачать на сайте или заполнить в ВУЦ. К заявлению нужно приложить копии документов, включая согласие на обработку персональных данных, автобиографию, эссе "Почему я выбираю военную подготовку удостоверение гражданина Российской Федерации, подлежащего призыву в ВС РФ, паспорт, документ о временной регистрации (для иногородних), зачетную книжку, анкету на бумажном и электронном носителе. Также потребуются фотографии 3х4 и 4х6. 2. Получить в ВУЦ бланк медицинской карты и направление для прохождения медицинской комиссии. Подготовить документы, необходимые для прохождения медицинской комиссии, согласно перечня (уточняется самостоятельно в военном комиссариате). 3. В марте-апреле пройти предварительный отбор в военном комиссариате по месту регистрации (или пребывания для иногородних) и сдать его результаты в учебную часть ВУЦ. 4. До 1 мая представить в ВУЦ результаты предварительного отбора, медицинскую карту и результаты профессионального психологического отбора, характеристику из деканата, документы, подтверждающие преимущественное право при проведении конкурсного отбора. 5. В апреле-мае сдать нормативы по физической подготовке на кафедре физического воспитания. 6. Конкурсная комиссия состоится в июле. В случае успешного прохождения по конкурсу до 1 августа заключить договор установленной формы. 7. До 28 августа приобрести специальную форму. 8. 1 сентября 2024 года приступить к занятиям согласно расписанию. Списки студентов, прошедших конкурсный отбор, будут размещены на стенде в ВУЦ.

Продолжение таблицы 3.4

№	Вопрос	Ответ
11	Когда приходит стипендия и что влияет на повышенную стипендию?	Стипендия приходит с 20-25 числа каждого месяца. Повышенная стипендия зависит от успеваемости студента. Она может быть повышенной за хорошую (одна четверка в сессии) или отличную (все "5") успеваемость. Также на нее могут влиять активное участие в жизни университета, научная, спортивная и культурно-творческая деятельность.
12	Где можно забрать аттестат?	Аттестат можно забрать в 3 корпусе РГГМУ - Воронежская 79, кабинет 16/3(первый этаж) - Студенческий отдел кадров.

Тестирование проводилось вручную. Также проводилось нагрузочное тестирование: сервис был опубликован перед студентами, после чего возник массовый наплыв пользователей. Сервис выдержал нагрузку и корректно работал. Пиковая нагрузка — 10 пользователей. Ни один из модулей не отказал. Работа в многопоточном режиме показала себя на достойном уровне.

Так же во время тестирования проверялась скорость поиска и генерации ответа. Максимальное время на генерацию ответа — 12,68 секунд, а минимальное — 4,33. В среднем поиск релевантных векторов, агументация промта и генерация ответа занимают ≈ 6 секунд, что соответствует нефункциональному требованию к производительности сервиса PER_1.

Так же по результатам тестирования можно отметить соотвествие ответов системы нефункциональным требованиям NFR_4 и NFR_5, опираясь на строки 7 и 8 таблицы 3.4.

ЗАКЛЮЧЕНИЕ

В процессе работы были выполнены поставленные задачи:

1) Проведен системный анализ
2) Произведена оценка экономической эффективности
3) Проектирование системы
4) Разработка системы
5) Тестирование системы

В результате выполнения задач мой было реализовано приложение, соответствующее всем функциональным и нефункциональным требованиям.

Созданный бот можно найти по адресу *https://t.me/IISIGT_bot* или по логину *@IISIGT_bot* внутри клиента Telegram.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Attention Is All You Need / A. Vaswani [и др.]. — 2017. — URL: https://arxiv.org/pdf/1706.03762.

2 *Jaekel B.* H&M, Sephora chatbots gain visibility in Kik's new marketplace. — 2015. — URL: https://www.marketingdive.com/ex/mobilemarketer/cms/news/messaging/22588.html.

3 *Jurafsky D.*, *Martin J. H.* Speech and Language Processing. — 2000.

4 *Mentor O.* SRP: The Single Responsibility Principle [Электронный ресурс]. — Архивировано 2 февраля 2015. — URL: https://web.archive.org/web/20150202200348/http://www.objectmentor.com/resources/articles/srp.pdf.

5 *Rong X.* word2vec Parameter Learning Explained. — 2016. — URL: https://arxiv.org/abs/1411.2738.

6 Исследование VK Мессенджера: три четверти россиян активно используют чат-боты. — 2023. — URL: https://vk.com/press/messenger-bots-research.

7 *Левенштейн В. И.* Двоичные коды с исправлением выпадений, вставок и замещений символов. Доклады Академий Наук СССР. — 1965. — URL: https://www.mathnet.ru/links/575633ee4c33f4301f89032ac23cc9e1/dan31411.pdf.

ПРИЛОЖЕНИЕ А
Диаграмма последовательности

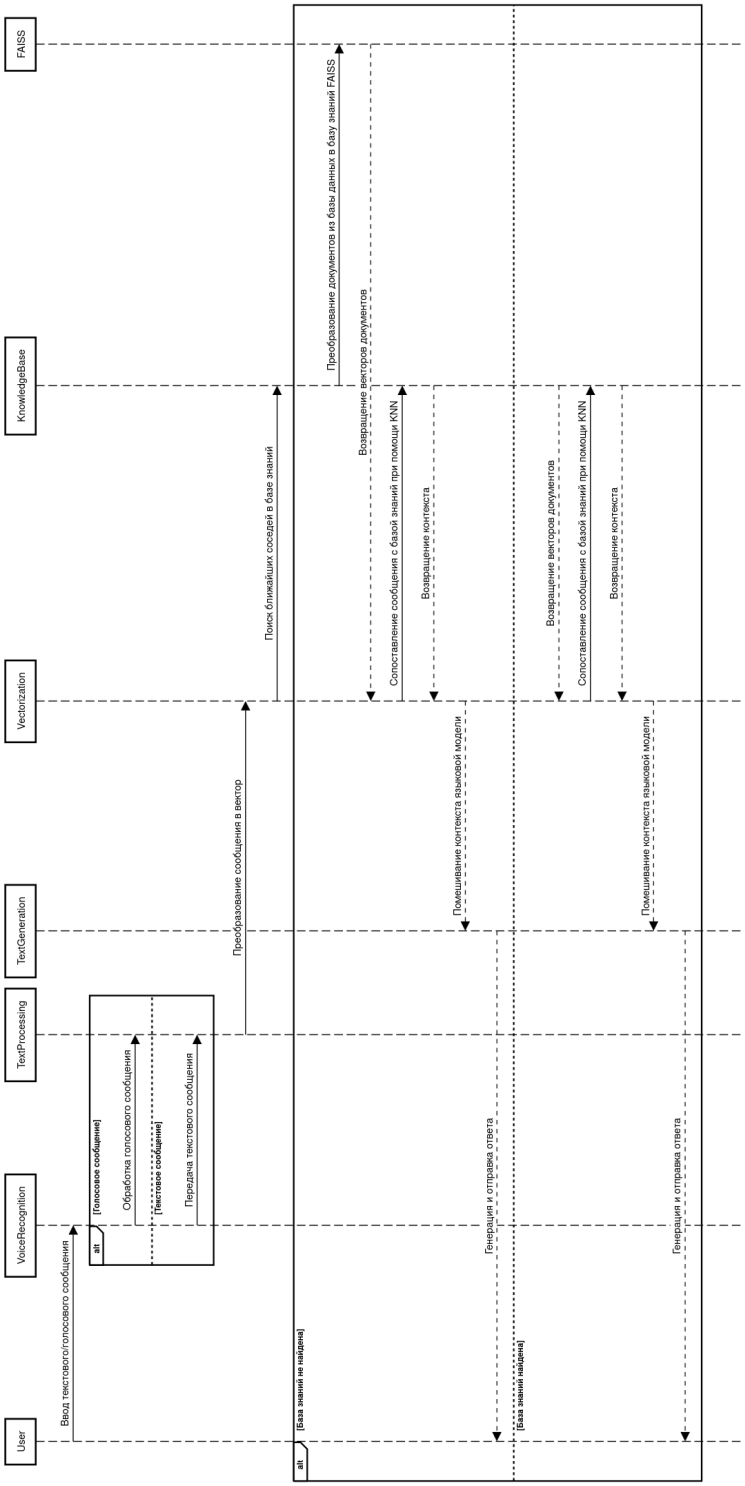

Рисунок А.1 — Диаграмма последовательности

ПРИЛОЖЕНИЕ Б
Конфигурация NeoVim

Листинг Б.1 — Отрывок конфигурации языкового сервера для работы NeoVim

```
local lspconfig = require 'lspconfig'

local capabilities =
 ↪  require('cmp_nvim_lsp').default_capabilities()

lspconfig.pylsp.setup {
  capabilities = capabilities,
  settings = { pylsp = { plugins =
   ↪  require('project.config').pylsp_plugins } },
}
lspconfig.tsserver.setup {
  capabilities = capabilities,
}
lspconfig.ccls.setup {
  capabilities = capabilities,
}
lspconfig.gopls.setup {
  capabilities = capabilities
}

-- `:help vim.diagnostic.*`
vim.keymap.set('n', '<space>e', vim.diagnostic.open_float)
```

ПРИЛОЖЕНИЕ В
Диаграмма Ганта

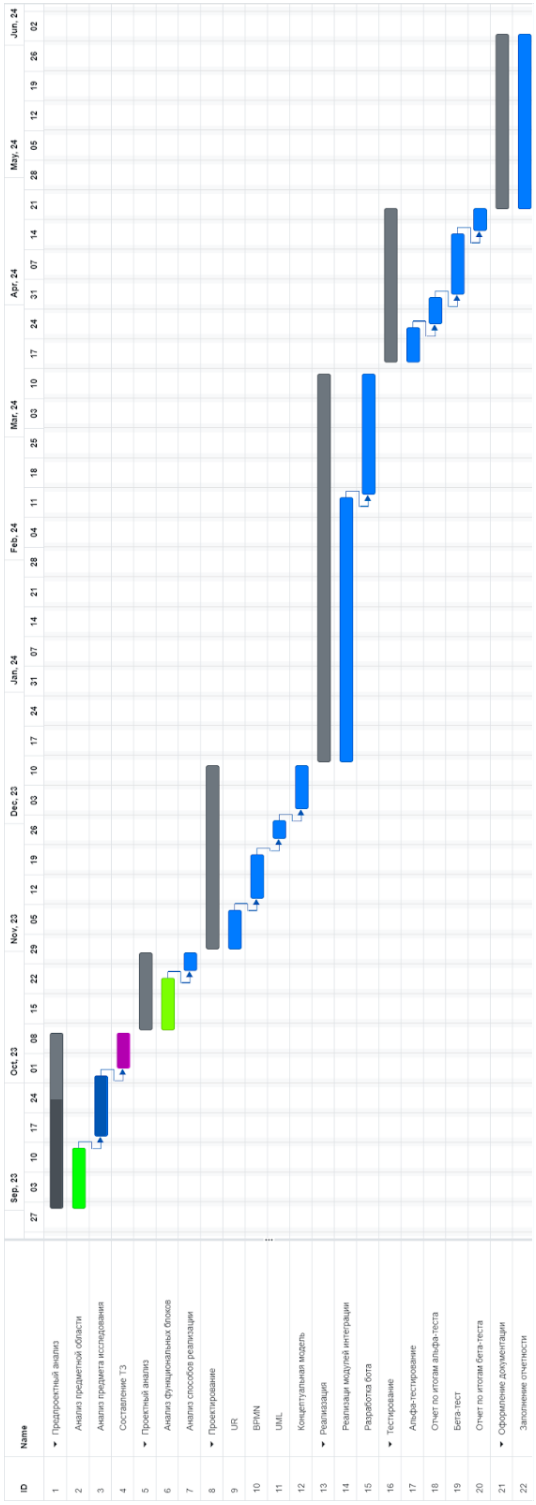

Рисунок В.1 — Диаграмма Ганта

ПРИЛОЖЕНИЕ Г
Конфигурация LaTeX

Листинг Г.1 — Конфигурация LaTeXдля титульного листа по ГОСТ 7.32

```
\begin{titlepage}
    \singlespacing
    \setlength{\parindent}{0pt}
    \begin{center}
        Министерство науки и высшего образования Российской
        ↪ Федерации\\
        Фереральное государственное бюджетное
        ↪ образовательное учреждение
высшего образования\\
        Российский государственный гидрометеорологический
        ↪ университет\\
        (РГГМУ)\\
        Институт информационных систем и геотехнологий\\
        Направление подготовки: 09.03.03 <<Прикладная
        ↪ информатика>>\\
        Профиль подготовки: <<Прикладные информационные
        ↪ системы\
        и геотехнологии>>

    \end{center}
    \vspace{\oneinterv}
    \begin{center}
    ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА\\
    (БАКАЛАВРСКАЯ РАБОТА)
    \end{center}
    \vspace{\oneinterv}
    На тему: <<Разработка приложения интеллектуального
    ↪ ассистента на базе
    технологий глубокого обучения.>>
    \vspace{\twointerv}

    \vfill

    \begin{tabular}{N{70mm}N{80mm}}
        Научный руководитель,\\
```

ПРИЛОЖЕНИЕ Д
Реализация обработки различных данных

Листинг Д.1 — Обработка текста

```
    @dp.message(F.text)
    async def chat_handler(message: Message) -> None:
        try:
            msg = await message.answer(dummy_message)
            talkbox_msg = talkbox.answer(message.text)
            await
             ↪  message.answer(f"{talkbox_msg}\n\n{stub}")
            await msg.delete()
            with open(logs, "a") as log:
                log_msg = f"ID: {message.from_user.id} |
                 ↪  Username: {message.from_user.full_name}
                 ↪  | Message: {message.text} | Answer:
                 ↪  {talkbox_msg}\n\n"
                log.write(log_msg)
                print(log_msg)

        except TypeError:
            await message.answer(
                "Что-то пошло не так! Сообщил разработчку о
                 ↪  произошедшей неполадке!"
            )
            with open(logs, "a") as log:
                log.write(
                    f"ID: {message.from_user.id} |
                     ↪  Username:
                     ↪  {message.from_user.full_name} |
                     ↪  ErrorMessage\n\n"
                )


```

Листинг Д.2 — Обработка звука

```
@dp.message(F.voice)
async def audio_handler(message: Message) -> None:
    voice_message = await
    ↪ bot.get_file(message.voice.file_id)
    voice_message = voice_message.file_path
    await bot.download_file(voice_message, "voice.wav")
    with open("voice.wav", "rb") as voice_message:
        text = recognize_speech(voice_message)
        msg = await message.answer(dummy_message)
        talkbox_msg = talkbox.answer(text)
        await message.answer(f"{talkbox.
            answer(text)}\n\n{stub}")
        await msg.delete()
        with open(logs, "a") as log:
            log_msg = f"ID: {message.from_user.id} |
            ↪ Username: {message.from_user.full_name} |
            ↪ Message: {text} | Answer:
            ↪ {talkbox_msg}\n\n"
            log.write(log_msg)
```

Листинг Д.3 — Обработка видео

```
@dp.message(F.video_note)
async def videomessage_handler(message: Message) -> None:
    voice_message = await
    ↪  bot.get_file(message.video_note.file_id)
    voice_message = voice_message.file_path
    await bot.download_file(voice_message, video_name)
    video_handler(video_name)
    with open("voice.wav", "rb") as voice_message:
        text = recognize_speech(voice_message)
        msg = await message.answer(dummy_message)
        talkbox_msg = talkbox.answer(text)
        await message.answer(f"{talkbox_msg}\n\n{stub}")
        await msg.delete()
        with open(logs, "a") as log:
            log_msg = f"ID: {message.from_user.id} |
            ↪  Username: {message.from_user.full_name} |
            ↪  Message: {text} | Answer:
            ↪  {talkbox_msg}\n\n"
            log.write(log_msg)

```