

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего
образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

**ИНТЕЛЛЕКТУАЛЬНЫЙ АССИСТЕНТ ДЛЯ
ПРОВЕДЕНИЯ СОБЕСЕДОВАНИЯ**

Студент	 (подпись, дата)	29.05.2016 <u>Духнов Олег Олегович</u> (ФИО полностью)
Руководитель	 (подпись, дата)	29.05.2016 <u>О.В. Юрова</u> (фамилия, инициалы)
Нормоконтролёр	 (подпись, дата)	29.05.2016 <u>А.С. Бычков</u> (фамилия, инициалы)
Рецензент	<u>доцент ПензГТУ</u> (должность, место работы)	 (подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 22.05.26 № 14)

Заведующий кафедрой	 (подпись)	<u>М.А. Митрохин</u> (фамилия, инициалы)
---------------------	--	---

Работа защищена с отметкой отлично (протокол заседания ГЭК от 09.06.2016 № 18)

Секретарь ГЭК	 (подпись)	<u>Е.И. Гурин</u> (фамилия, инициалы)
---------------	--	--

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»



«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

« 20 » 09 20 24 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Духнов Олег Олегович гр. 24ВВИм1 факультета ВТ

2. Тема работы Интеллектуальный ассистент для проведения собеседования

Тема утверждена приказом ПГУ № 920/0 от " 20 " 09 20 24 г.

3. Руководитель магистерской диссертации к.т.н., доцент кафедры ВТ ПГУ Юрова О.В.

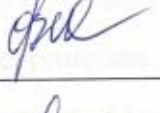
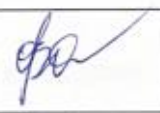
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Выполнить исследование и разработать программный комплекс интеллектуального ассистента для проведения технических собеседований с использованием больших языковых моделей и нейросетевых средств распознавания речи. Создать web-приложение, позволяющее автоматизировать процесс интервью: генерировать технические вопросы, принимать текстовые и голосовые ответы кандидата, анализировать качество ответов, формировать обратную связь и итоговый отчет по результатам прохождения собеседования. Программный комплекс должен состоять из клиентской части, серверной микросервисной архитектуры и AI-сервисов.

5. Перечень подлежащих разработке вопросов

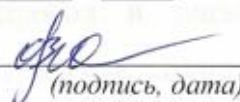
Требуется произвести анализ разработанности предметной области интеллектуальных систем для проведения собеседований и оценки знаний кандидатов. Провести анализ возможностей использования больших языковых моделей для генерации технических вопросов, анализа ответов и формирования обратной связи. Исследовать способы построения микросервисной архитектуры web-приложения, обеспечивающей взаимодействие клиентской части, Java-сервисов и Python AI-сервисов. Провести анализ и реализацию механизма передачи голосовых ответов пользователя, а также последующей передачи распознанного текста. Выполнить тестирование различных больших языковых моделей и моделей распознавания речи, выбрать наиболее подходящие решения для поставленной задачи и провести проверку работоспособности разработанной системы в сценариях обычного AI-собеседования и HR-интервью по публичной ссылке.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	10%	20.09.24-15.11.2024	
2.	Анализ требований к интеллектуальному ассистенту	10%	16.11.2024-20.12.2024	
3.	Изучение литературы и существующих решений	10%	21.12.2024-31.01.2025	
4.	Проектирование архитектуры web-приложения	15%	01.02.2025-15.03.2025	
5.	Реализация интеллектуального ассистента	30%	16.03.2025-31.01.2026	
6.	Тестирование разработанной системы	10%	01.02.2026-28.02.2026	
7.	Анализ полученных результатов исследования и выбор наиболее подходящих моделей	10%	01.03.2026-30.04.2026	
8.	Оформление пояснительной записки	5%	01.05.2026-20.05.2026	

Дата выдачи задания " 20 " 09 20 24 г.

Руководитель магистерской диссертации


(подпись, дата)

Юрова О.В.

Задание к исполнению принял студент


(подпись, дата)

Духнов О.О.

Работу к защите допустить
Декан факультета

 05.06.2020

Фионова Л.Р.

РЕФЕРАТ

Пояснительная записка содержит здесь до приложения 88 лист, 39 рисунков, 1 таблица, 19 источников, 2 приложения.

ИНТЕЛЛЕКТУАЛЬНЫЙ АССИСТЕНТ, ТЕХНИЧЕСКОЕ СОБЕСЕДОВАНИЕ, БОЛЬШАЯ ЯЗЫКОВАЯ МОДЕЛЬ, LLM, МИКРОСЕРВИСНАЯ АРХИТЕКТУРА, РАСПОЗНАВАНИЕ РЕЧИ, ТРАНСКРИБАЦИЯ

Целью работы является разработка интеллектуального ассистента для проведения технических собеседований с использованием больших языковых моделей и нейросетевых средств распознавания речи. Разрабатываемое приложение предназначено для автоматизации процесса интервью: генерации технических вопросов, приема текстовых и голосовых ответов кандидата, анализа качества ответов, формирования обратной связи и итогового отчета.

В результате проведенной работы была реализована микросервисная web-система, серверные Java-микросервисы на Spring, а также Python AI-сервисы для взаимодействия с большой языковой моделью и транскрибации речи.

Особое внимание было уделено исследованию больших языковых моделей, настройке взаимодействия с локальной LLM через Ollama, разработке сервиса генерации вопросов и оценки ответов, а также подключению модуля распознавания голосовых ответов

					ПГУ 1.090401.06.001 ПЗ			
Изм.	Лист	№ докум	Подпись	Дата	Интеллектуальный ассистент для проведения собеседования Пояснительная записка	Лит.	Лист	Листов
Разраб.		Духнов О.О.		29.06.24				
Провер.		Юрова О.В.		29.06.24			2	106
Н. Контр.		Бычков А.С.		29.06.24		ФВТ, каф. ВТ гр. 24ВВИм1		
Утв.		Митрохин М.А.		29.06.24				

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ.....	6
ВВЕДЕНИЕ	7
1 Анализ предметной области.....	9
1.1 Анализ требований.....	9
1.2 Анализ больших языковых моделей.....	11
1.2.1 LLM Qwen3-8B.....	11
1.2.2 LLM Ministral 3 8B Instruct.....	12
1.2.3 LLM Mistral-7B-Instruct-v0.3	13
1.2.4 LLM IBM Granite 3.3 8B Instruct.....	14
1.2.5 LLM DeepSeek-R1-Distill-Qwen-7B.....	14
1.2.6 LLM Phi-4-mini-instruct.....	15
1.2.7 LLM Llama 3.1 8B Instruct	16
1.2.8 LLM Gemma 3 4B-it.....	16
1.2.9 LLM Mistral-Nemo-Instruct-2407	17
1.2.10 Сравнительный анализ моделей	18
1.2.11 Вывод	21
1.3 Модуль транскрибации речи в текст	22
2 Архитектура больших языковых моделей	24
2.1 Задача Encoder	25
2.2 Задача Decoder	28
2.3 Архитектура Decoder-only Transformer	30
3 Анализ аналогов	32
3.1 Приложение HireVue.....	32
3.2 Приложение Interviewer.AI.....	33
3.3 Приложение Willo.....	34
3.4 Приложение Final Round AI	35
4 Архитектура приложения	37
4.1 Клиентская часть приложения	38

4.2 Микросервис API Gateway	39
4.3 Микросервис Auth Service.....	39
4.4 Микросервис Interview Service	40
4.5 Микросервис Assessment Service.....	40
4.6 Микросервис Question Engine Service.....	41
4.7 Микросервис Transcribe Service.....	41
4.8 Микросервис Report Service.....	41
5 Реализация приложения.....	43
5.1 Реализация инструкций	43
5.1.1 Реализация инструкции генерации вопросов.....	43
5.1.2 Реализация инструкции определения параметров.....	44
5.1.3 Реализация инструкции оценки ответа	45
5.1.4 Реализация инструкции оценки ответа в процессе интервью	47
5.1.5 Реализация инструкции остановки интервью	50
5.2 Тестирование больших языковых моделей.....	51
5.2.1 Тестирование Qwen3-8B.....	52
5.2.2 Тестирование Ministral 3 8B Instruct.....	54
5.2.3 Тестирование Mistral-7B-Instruct-v0.3-Q8_0	57
5.2.4 Тестирование IBM Granite 3.3 8B Instruct.....	59
5.2.5 Тестирование DeepSeek-R1-Distill-Qwen-7B.....	62
5.2.6 Тестирование Phi-4-mini	64
5.2.7 Тестирование Llama 3.1 8B Instruct	67
5.2.8 Тестирование Gemma 3 4B	69
5.2.9 Тестирование Mistral-Nemo-Instruct-2407-Q4_K_M.....	71
5.2.10 Вывод.....	73
5.3 Тестирование нейросетевых моделей для транскрипции голоса	74
5.3.1 Тестирование Whisper.....	75
5.3.2 Тестирование Vosk	76
5.3.3 Тестирование NVIDIA NeMo / Parakeet	77
5.3.4 Тестирование Coqui STT	79

5.4 Итоговый результат	80
ЗАКЛЮЧЕНИЕ.....	86
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	87
ПРИЛОЖЕНИЕ А – ЛИСТИНГ ПРОГРАММЫ.....	89
ПРИЛОЖЕНИЕ Б – ПРЕЗЕНТАЦИЯ	101

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей работе применяются следующие сокращения и обозначения.

ИИ – Искусственный интеллект

AI (Artificial Intelligence) – Искусственный интеллект

API (Application Programming Interface) – Программный интерфейс приложения

ASR (Automatic Speech Recognition) – Автоматическое распознавание речи

CORS (Cross-Origin Resource Sharing) – Механизм совместного использования ресурсов между разными источниками

GPU (Graphics Processing Unit) – Графический процессор задач

HR (Human Resources) – Управление персоналом

IT (Information Technology) – Информационные технологии

JSON (JavaScript Object Notation) – Текстовый формат обмена данными

JWT (JSON Web Token) – Токен для передачи данных аутентификации

LLM (Large Language Model) – Большая языковая модель

QKV (Query, Key, Value) – Механизм запросов, ключей и значений в архитектуре внимания

RAG (Retrieval-Augmented Generation) – Генерация ответа с использованием внешних источников данных

REST (Representational State Transfer) – Архитектурный стиль взаимодействия web-сервисов

RNN (Recurrent Neural Network) – Рекуррентная нейронная сеть

RNNT (Recurrent Neural Network Transducer) – Архитектура нейросетевой модели для распознавания речи

STT (Speech-to-Text) – Преобразование речи в текст

URL (Uniform Resource Locator) – Единый указатель ресурса

VRAM (Video Random Access Memory) – Видеопамять графического процессора

ВВЕДЕНИЕ

Двадцатые годы двадцать первого века ознаменовались революцией в области искусственного интеллекта. С каждым годом идет активное увеличение возможностей больших языковых моделей, которые позволяют делегировать повседневные задачи людей на ИИ-ассистентов, а также открывают новые варианты использования программных продуктов путем внедрения этих инструментов в разрабатываемые приложения.

На данный момент искусственный интеллект уже хорошо справляется с такими задачами как: генерация, редактирование, анализ и рерайт текста; написание кода; перевод языков; резюмирование и извлечение данных; ответы на вопросы; генерация и обработка изображений; работа с аудио и видео. С каждым месяцем улучшаются большие языковые модели и соответственно повышается качество выполнения данных задач, что позволяет разработчикам активно внедрять данные технологии в свои программные продукты с целью решения реальных задач бизнеса и закрытие потребностей в тех областях, в которых раньше без человеческого контроля невозможно было решать специфические задачи.

В последнее время на рынке труда IT по данным hh.ru на март 2026 года сформировалась ситуация, когда на одну вакансию приходится 22.9 активных резюме, из-за этого возникает большая нагрузка на специалистов по найму [1]. В связи с этим в сфере рекрутинга HR специалисты стали активно использовать искусственный интеллект для отбора кандидатов. На этом использование этой технологии не ограничивается, также большие языковые модели способны решить задачу проведения вступительного интервью с целью выявления его профессиональных навыков, тем самым упрощая работу специалиста по набору персонала. Поэтому основными задачами магистерской диссертации являются:

- анализ существующих больших языковых моделей, которые позволяют анализировать и генерировать текстовые данные,

- анализ существующих предложений на рынке на тему проведения собеседования с искусственным интеллектом,

- внедрение и тестирование различных моделей для выявления оптимальной,

- анализ результатов.

Целью работы является проектирование, разработка и реализация приложения с использованием большой языковой модели для проведения собеседования.

Работа может быть полезна компаниям, которым необходимо оптимизировать найм сотрудников, а также соискателей, которые хотят потренироваться перед реальным собеседованием.

1 Анализ предметной области

Начальный этап задач, предполагающий работу в рамках информационных технологий, представляет собой теоретический анализ многих аспектов: технологической, предметной и информационной основы задачи, инструментов, с помощью которых можно прийти к наиболее эффективному решению, разделения задачи на последовательные функциональные блоки, реализации из которых приведет к ее решению, возможные архитектурные решения, которые могут быть использованы для получения приложения, отвечающего всем требованиям.

Первым шагом к выполнению необходимого анализа будет исследование общих задач, требуемых к большой языковой модели, а также обоснование выбора класса модели.

1.1 Анализ требований

Для осуществления поставленной цели работы необходимо учесть перед выбором наиболее подходящей модели общие требования, позволяющие отсеять варианты, которые не подходят для использования в рамках реализации приложения. Поэтому были выделены следующие требования:

- возможность бесплатного использования,
- возможность локального запуска модели без использования облачных ресурсов с целью локальной проверки работоспособности модели и приложения,
- поддержка русского языка,
- приемлемое количество параметров модели, которые обеспечивает баланс между качеством модели и производительностью,
- способность к генерации релевантных вопросов, учитывая выбранную специальность и уровень подготовки кандидата,
- способность к анализу и оценке ответов кандидата, включая выявление ошибок и правильных утверждений, а также возможность задавать наводящий вопрос, если у интервьюируемого возникают проблемы с ответом,

- возможность формировать полный отчет о прохождении собеседования кандидатом с указанием недостатков и положительных моментов, а также оценкой знаний кандидата,
- поддержка структурированного вывода в формате JSON,
- способность работы с голосовым вводом, чтобы была возможность у кандидата отвечать на вопросы речью,
- поддержка лицензии, которая допускает использовать модель в коммерческих целях в рамках разрабатываемого приложения.

При определении требований важно отметить, чтобы была возможность запустить локально и проверить модель на видеокарте RTX 3060 с 12 ГБ видеопамяти с целью исследования возможностей тестируемой большой языковой модели.

Отдельным критерием при выборе модели стоит отметить ее лицензию, так как не всегда модели из открытых источников являются свободными для использования в коммерческом плане, поэтому при желании дальше продвигать свой продукт, необходимо провести анализ и в этой области, так как если даже модель запускается локально и не требует облачного подключения, ее веса, архитектура, токенизатор или готовая квантовая версия могут распространяться на условиях конкретной лицензии. Из-за этого следует отдавать предпочтение MIT, Apache 2.0, Llama Community License или Gemma Terms of Use, так как разрешается коммерческое использование и распространение моделей при соблюдении базовых правил.

1.2 Анализ больших языковых моделей

Большие языковые модели представляют собой нейросетевые модели, которые предназначены для различных задач, включая обработку, анализ, генерацию текста на естественном языке. Исходя из рассмотренных требований, необходимо выбрать такие языковые модели, которые будут удовлетворять всем поставленным запросам и подойдут для реализации работы.

В рамках исследования рассматриваются только бесплатные модели, которые будут развертываться локально. Так как присутствуют ограничения по железу, выбраны только те модели, имеющие количество параметров от 4B до 12B, а также с поддержкой квантованных форматов Q4_K_M, Q5_K_M. Использование квантизации позволяет уменьшить разрядность весов модели и снизить объем потребляемой видеопамати при локальном запуске.

Модели с удаленным облачным подключением по API-ключу в рамках исследования не рассматриваются, так как их использование зачастую требует оплаты за обработанные токены. Такая плата связана с высокой стоимостью обслуживания вычислительной инфраструктуры, на которой запускаются большие языковые модели.

Далее будут проанализированы модели, которые потенциально подходят под все требования и в дальнейшем будут исследованы в научно-исследовательской работе.

1.2.1 LLM Qwen3-8B

Qwen3-8B – это большая языковая модель семейства Qwen3, разработанная Alibaba. Модель относится к классу причинных языковых моделей, то есть моделей, которые генерируют текст последовательно, предсказывая следующий токен на основе предыдущих. Такой тип моделей хорошо подходит для диалоговых систем, анализа текста, генерации вопросов и выполнения инструкций [2].

Qwen3-8B содержит 8,2 млрд параметров, из которых 6,95 млрд относятся к основной части модели без учета эмбеддингов. Эмбеддинги – это

числовые представления токенов, с которыми работает нейросеть. Модель имеет 36 слоев и использует плотную архитектуру Transformer-декодера. Плотная архитектура означает, что при обработке запроса задействуется вся модель целиком, в отличие от MoE-моделей, где активируется только часть экспертных блоков. В модели также используется механизм группового внимания к запросам: 32 головы внимания для запросов и 8 голов для ключей и значений. Такой механизм снижает расход памяти и ускоряет генерацию текста. Контекстное окно составляет 32 768 токенов, а при использовании технологии YaRN может быть расширено до 131 072 токенов. Модель распространяется под лицензией Apache 2.0. [3].

Для рассматриваемого приложения Qwen3-8B представляет интерес как одна из наиболее сбалансированных моделей 8B-класса. Она подходит для анализа ответов кандидатов, генерации вопросов, формирования выводов и работы с достаточно длинным контекстом собеседования.

1.2.2 LLM Ministral 3 8B Instruct

Ministral 3 8B Instruct – это модель компании Mistral AI, ориентированная на локальное и периферийное развертывание. Под периферийным развертыванием понимается запуск модели на локальном устройстве или сервере пользователя, а не только в облачной инфраструктуре.

Ministral 3 8B относится к семейству моделей Ministral 3 и имеет размер около 8B параметров. В документации Mistral модель описывается как эффективная модель для локального использования и работы на различном оборудовании. Она поддерживает контекстное окно до 256 000 токенов, что является преимуществом при обработке длинных материалов, например полного транскрипта собеседования. Однако при локальном запуске фактический размер доступного контекста будет зависеть от объема видеопамяти и выбранной квантизации. Архитектурно Ministral 3 8B является плотной Transformer-моделью. В техническом описании указывается, что модель имеет 32 скрытых слоя, размер скрытого состояния 4096, 32 головы

внимания и 8 голов ключей и значений, что указывает на использование группового внимания к запросам. [4]

Модель поддерживает вызов функций и структурированный вывод. Вызов функций означает, что модель может формировать обращение к внешнему инструменту, например к базе вопросов или модулю сохранения результата. Структурированный вывод важен для возврата результата в формате JSON, что необходимо для дальнейшей обработки данных в приложении. Mistral 3 8B также поддерживает текстовые и визуальные задачи, однако в рамках данной работы рассматривается именно текстовый сценарий использования. Модель распространяется под лицензией Apache 2.0.

1.2.3 LLM Mistral-7B-Instruct-v0.3

Mistral-7B-Instruct-v0.3 – это инструкция-ориентированная версия модели Mistral 7B, разработанная компанией Mistral AI. Она относится к классу компактных больших языковых моделей и может использоваться как базовый вариант для сравнения с более новыми моделями. В исследовательской работе такая модель полезна как контрольная точка, поскольку она широко распространена, хорошо изучена и поддерживается многими инструментами локального запуска.

Архитектура Mistral 7B основана на Transformer-декодере. В статье о Mistral 7B указано, что модель содержит около 7 млрд параметров и использует два важных механизма: групповое внимание к запросам и внимание со скользящим окном [5]. Внимание со скользящим окном означает, что модель обрабатывает не весь текст целиком на каждом шаге, а ограниченный фрагмент контекста. Это снижает вычислительные затраты и помогает эффективнее работать с длинными последовательностями.

Версия Mistral-7B-Instruct-v0.3 дополнительно дообучена для выполнения инструкций. В карточке модели указано, что по сравнению с предыдущей версией она получила расширенный словарь токенизатора до 32 768 токенов, поддержку токенизатора версии v3 и поддержку вызова функций [6]. Для ассистента собеседований это полезно, так как модель может не

только генерировать текст, но и использоваться в сценариях взаимодействия с внешними модулями приложения. Модель распространяется под лицензией Apache 2.0.

1.2.4 LLM IBM Granite 3.3 8B Instruct

IBM Granite 3.3 8B Instruct – это инструкция-ориентированная модель семейства Granite, разработанная IBM. Модель предназначена для задач генерации текста, выполнения инструкций, рассуждения, программирования и делового применения. Для ассистента проведения собеседований эта модель интересна тем, что семейство Granite ориентировано на корпоративные сценарии использования.

Granite 3.3 8B Instruct содержит 8 млрд параметров и поддерживает контекстное окно 128K токенов. Модель была дообучена на базе Granite-3.3-8B-Base для улучшения рассуждений и следования инструкциям. Также заявлена поддержка структурированного рассуждения через специальные теги, разделяющие промежуточные рассуждения и финальный ответ. Архитектурно модели Granite 3.3 построены на плотной архитектуре decoder-only dense transformer, то есть на плотном Transformer-декодере [7].

Granite 3.3 8B Instruct поддерживает задачи суммаризации, классификации текста, извлечения информации, ответов на вопросы, RAG-сценарии, программирование, вызов функций, многоязычный диалог и работу с длинным контекстом. RAG-сценарий означает генерацию ответа с использованием внешних источников, например базы вопросов или материалов по специальности. Модель распространяется под лицензией Apache 2.0, что делает ее подходящей для исследования и потенциального коммерческого использования.

1.2.5 LLM DeepSeek-R1-Distill-Qwen-7B

DeepSeek-R1-Distill-Qwen-7B – это модель DeepSeek, ориентированная на задачи рассуждения и созданная на основе модели Qwen. Она была создана путем дистилляции знаний более крупной модели DeepSeek-R1 в модель меньшего размера. Под дистилляцией понимается процесс, при котором

большая модель используется для обучения меньшей, чтобы передать ей часть своих возможностей при меньших вычислительных требованиях.

Модель относится к классу примерно 7B параметров и построена на основе Qwen. Архитектурно DeepSeek-R1-Distill-Qwen-7B наследует основу Qwen. В стороннем описании Benchable уточняется, что данная модель основана на Qwen2.5-Math-7B. Следовательно, ее корректно описывать как плотную языковую модель на базе Transformer-декодера семейства Qwen, дополнительно адаптированную для задач рассуждения [8]. Распространяется под лицензией MIT. MIT является свободной лицензией, допускающей коммерческое использование, модификацию и распространение при сохранении уведомления об авторских правах.

Основная особенность модели заключается в ее ориентации на рассуждение. Семейство R1 развивалось с использованием обучения с подкреплением, то есть метода, при котором модель улучшает ответы на основе сигнала качества. Для ассистента собеседований эта модель может быть полезна как модель-оценщик: она способна анализировать ответ кандидата, выявлять логические ошибки и формировать более обоснованную оценку.

1.2.6 LLM Phi-4-mini-instruct

Phi-4-mini-instruct – это компактная модель Microsoft семейства Phi-4. Она относится к малым языковым моделям и содержит около 3,8 млрд параметров. Несмотря на сравнительно небольшой размер, модель ориентирована на задачи рассуждения, выполнения инструкций и многоязычной генерации.

Phi-4-mini-instruct построена на синтетических данных и отфильтрованных публичных веб-данных с акцентом на качественные данные для рассуждения. Модель поддерживает контекстное окно 128K токенов, что является большим значением для компактной модели [9].

Архитектурно Phi-4-mini-instruct является плотной моделью на основе Transformer-декодера. В описаниях модели также указывается словарь

токенизатора примерно на 200 тысяч токенов, групповое внимание к запросам и общие входные и выходные эмбединги. Данная модель будет рассмотрена в качестве примера, чтобы проверить, на что способны относительно небольшие языковые модели для решения поставленной задачи. Модель распространяется под лицензией MIT.

1.2.7 LLM Llama 3.1 8B Instruct

Llama 3.1 8B Instruct – это инструкция-ориентированная модель семейства Llama 3.1, разработанная Meta. Она относится к текстовым моделям, предназначенным для многоязычных диалоговых сценариев. В документации Meta указано, что Llama 3.1 выпускается в размерах 8B, 70B и 405B, а инструкция-ориентированные версии оптимизированы для многоязычного диалога.

Архитектурно Llama 3.1 является авторегрессионной языковой моделью на основе оптимизированной архитектуры Transformer. Авторегрессионная модель генерирует текст последовательно, предсказывая следующий токен на основе уже обработанного текста. Настроенные версии Llama 3.1 используют обучение на примерах и обучение с учетом человеческих предпочтений, что улучшает полезность и безопасность ответов.

Для Llama 3.1 заявлено контекстное окно до 128K токенов и улучшенная многоязычная поддержка. Для ассистента собеседований это делает модель полезной при работе с длинной историей диалога [10]. Однако модель распространяется по пользовательской лицензии Meta, а не по Apache 2.0 или MIT, поэтому перед использованием в коммерческом приложении необходимо отдельно анализировать условия лицензии.

1.2.8 LLM Gemma 3 4B-it

Gemma 3 4B-it – это инструкция-ориентированная модель семейства Gemma 3 от Google. Она относится к компактным моделям и содержит около 4 млрд параметров. Gemma 3 позиционируется как семейство легких моделей с открытыми весами, созданных на основе технологий, связанных с моделями Gemini.

В официальной карточке указано, что Gemma 3 поддерживает контекстное окно 128K токенов, более 140 языков и подходит для задач генерации текста, ответов на вопросы, суммаризации и рассуждения. Также Gemma 3 является мультимодальной моделью, то есть может принимать на вход не только текст, но и изображения, однако в рамках данной работы рассматривается текстовый сценарий использования.

Архитектурно Gemma 3 4B относится к моделям на основе Transformer-декодера и ориентирована на эффективную работу на разном оборудовании – от облачных систем до локальных устройств. Для ассистента собеседований модель может использоваться как компактный вариант для генерации вопросов, суммаризации ответов и предварительного анализа. При этом из-за меньшего количества параметров она может уступать 7B–8B моделям в сложной оценке знаний кандидата [11]. Лицензия Gemma Terms of Use, коммерческое использование разрешено, только есть некоторые запрещенные виды использования модели, которые включены в соблюдение законов от Google.

1.2.9 LLM Mistral-Nemo-Instruct-2407

Mistral-Nemo-Instruct-2407 – это инструкция-ориентированная модель, разработанная совместно Mistral AI и NVIDIA. Она является более крупной моделью в рассматриваемом наборе и выбрана как представитель класса до 12B параметров, чтобы оценить влияние увеличения размера модели на качество работы ассистента проведения собеседований.

Mistral NeMo содержит 12 млрд параметров и поддерживает контекстное окно до 128K токенов. В официальной публикации Mistral указано, что модель выпущена под лицензией Apache 2.0 и демонстрирует сильные результаты в задачах рассуждения, знаниях о мире и программировании в своем размерном классе [12].

Mistral-Nemo-Instruct-2407 является дообученной на инструкции версией Mistral-Nemo-Base-2407. Она была совместно обучена Mistral AI и NVIDIA и предназначена для многоязычных задач и следования инструкциям.

Модель также использует токенизатор Tekken, который был разработан для более эффективной обработки многоязычного текста. Для ассистента собеседований это полезно, так как ответы кандидатов могут содержать русский язык, английские технические термины и смешанные формулировки.

Из-за большего размера Mistral-Nemo-Instruct-2407 может потенциально лучше справляться со сложным анализом ответов и формированием подробных отчетов. Однако по сравнению с моделями 7B–8B она требует больше видеопамяти и может работать медленнее при локальном запуске, особенно при использовании длинного контекста. Поэтому данная модель рассматривается как более мощная, но более ресурсоемкая альтернатива.

1.2.10 Сравнительный анализ моделей

Для проведения сравнительного анализа данных больших языковых моделей были исследованы различные бенчмарки от независимых источников, которые рассматривают по разным параметрам и целям представленные модели, и выставляют оценки в зависимости от качества пройденного теста конкретной моделью [13-17].

Для оценки тестов в бенчмарках были выбраны критерии, которые могут достаточно точно оценить возможности данных моделей:

- MMLU – проверяет общие знания модели по разным областям: математика, история, право, медицина, программирование и др. Чем выше балл, тем лучше общая эрудиция модели,

- MMLU-Pro – более сложная версия MMLU. Она сильнее проверяет рассуждение, а не просто знание фактов,

- Arena Hard – набор сложных пользовательских запросов, где ответы моделей оцениваются автоматически, часто с помощью более сильной модели-судьи. Хорошо отражает качество диалоговых ответов,

- BBH / BigBench Hard – сложные задачи на рассуждение, логику и понимание инструкций,

- GSM8K – школьные математические текстовые задачи. Показывает способность модели к пошаговому решению простых математических задач,

– MATH / MATH-500 – более сложные математические задачи. Хорошо показывает качество рассуждения,

– GPQA Diamond – сложные вопросы уровня экспертов, часто из научных областей. Чем выше балл, тем лучше модель справляется с трудными логическими задачами,

– HumanEval – тест на генерацию программного кода. Модель должна написать функцию, которая проходит скрытые тесты,

– MBPP – еще один тест на программирование, но с более короткими задачами,

– IFEval – проверяет, насколько модель точно следует инструкциям. Для ассистента собеседований это очень важно,

– BFCL – Berkeley Function Calling Leaderboard. Проверяет, насколько хорошо модель умеет вызывать функции и инструменты. Это важно, если приложение будет работать с базой вопросов, модулем отчетов или внешними сервисами. а сами модели строками и выстави баллы. если где-то нет тестов, ставь прочерк. мне не нужно текста в таблице, только сухие цифры, и проверь, действительно ли они такие.

Такой формат представления данных позволяет быстро сопоставить сильные и слабые стороны каждой модели по отдельным группам задач. Особое внимание уделялось показателям, связанным с рассуждением, программированием и следованием инструкциям, поскольку именно эти способности наиболее важны для интеллектуального ассистента при проведении технического собеседования. Полученные результаты использовались не как единственный критерий выбора модели, а как предварительная основа для дальнейшего практического тестирования в разработанном приложении.

В таблице 1 представлен результат исследования, где выставлены баллы за пройденные тесты моделями. Если в таблице под пунктом стоит прочерк, значит тест в бенчмарке не был проведен.

Таблица 1 – Результаты тестирования моделей в различных бенчмарках

Модель	MML U	MML U-Pro	Arena Hard	BB H	GSM8 K	MATH / MATH-500	GPQA Diamond	HumanEval	MB PP	IFEval	BFL
Qwen3-8B	73	56	79	78	84	62	39	72	69	83	60
Minstral 3 8B Instruct	76	–	50	–	–	87	39	–	70	–	–
Mistral-7B-Instruct-v0.3	64	30	–	56	36	10	24	29	51	–	–
IBM Granite 3.3 8B Instruct	65	–	57	69	80	69	–	89	–	74	–
DeepSeek-R1-Distill-Qwen-7B	–	–	10	67	78	92	49	79	–	59	–
Phi-4-mini-instruct	67	52	32	70	88	64	36	74	65	70	70
Llama 3.1 8B Instruct	68	44	25	63	82	47	26	66	69	74	77
Gemma 3 4B-it	58	43	31	72	89	75	30	71	63	90	–
Mistral-Nemo-Instruct-2407	68	–	–	–	65	–	–	38	–	–	–

На рисунке 1 представлена линейная диаграмма, которая построена по принципу взвешенного среднего балла по результатам прохождения больших языковых моделей в бенчмарках.

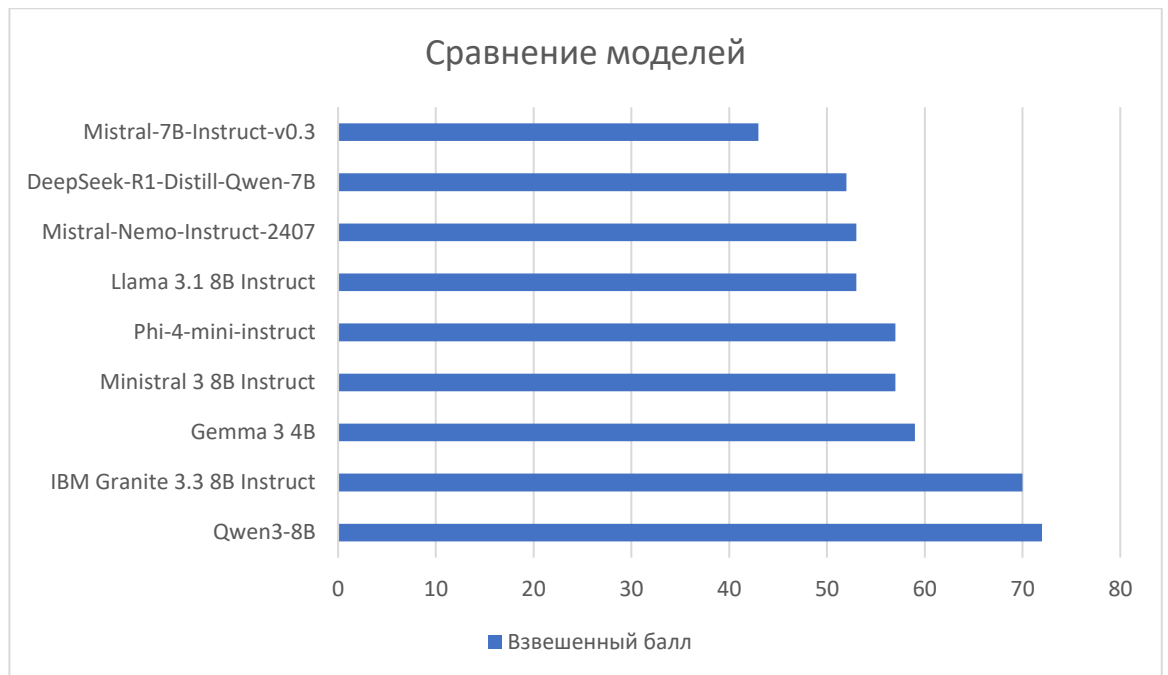


Рисунок 1 – Сравнительная диаграмма моделей

По результатам бенчмарков видно, что модели имеют разную специализацию. Универсальные модели, такие как Qwen3-8B, Llama 3.1 8B Instruct и Ministral 3 8B Instruct, показывают сбалансированные результаты в задачах общего понимания, генерации текста и следования инструкциям. DeepSeek-R1-Distill-Qwen-7B выделяется в задачах рассуждения и математики, поэтому может быть полезна для оценки ответов кандидата. Phi-4-mini-instruct демонстрирует высокие результаты для своего малого размера, но может быть ограничена в задачах, требующих большого объема фактических знаний. Granite 3.3 8B Instruct от IBM хорошо показал себя в задачах генерации кода математических задачах. В целом все модели показали достаточно хороший результат в различных тестах, выделяясь в различных бенчмарках, под которые больше специализированы.

1.2.11 Вывод

Таким образом, все представленные большие языковые модели потенциально соответствуют основным требованиям, предъявляемым к реализации приложения. Они имеют подходящий размер для локального развертывания, могут использоваться без подключения к платным облачным API и обладают возможностями, необходимыми для генерации вопросов,

анализа ответов кандидатов и формирования структурированного вывода, являются бесплатными и с необходимой лицензией для использования. При этом окончательный выбор модели должен осуществляться на основе дальнейшего практического тестирования, включающего оценку качества ответов и корректности обработки русского языка.

1.3 Модуль транскрибации речи в текст

В рамках разрабатываемого приложения ответы кандидата могут поступать не только в текстовом формате, но и в виде голосовой записи. Поэтому для дальнейшей обработки данных необходим отдельный модуль автоматического распознавания речи, который преобразует аудиосигнал в текст. После транскрибации полученный текст может быть передан большой языковой модели для анализа ответа, выявления ошибок, генерации уточняющего вопроса и формирования итогового отчета.

Основными критериями являются: поддержка русского языка, открытый исходный код и бесплатное использование библиотеки. Для реализации такого модуля могут быть рассмотрены следующие бесплатные и локально разворачиваемые решения.

Whisper – система автоматического распознавания речи от OpenAI. Она предназначена для многоязычного распознавания речи, перевода речи и определения языка. Согласно описанию OpenAI, Whisper обучался на 680 000 часах многоязычных и многозадачных данных, что делает его устойчивым к акцентам, фоновому шуму и технической лексике. Open-source реализация Whisper доступна на GitHub и может запускаться локально.

Vosk – офлайн-инструмент для распознавания речи, рассчитанный на работу без подключения к облачным сервисам. Он поддерживает разные платформы, включая компьютеры, серверы, Android, iOS и Raspberry Pi, а также предоставляет API для Python, Java, Node.js, C#, C++, Rust, Go и других языков. Vosk может использоваться для транскрибации лекций, интервью, субтитров и голосовых ассистентов.

NVIDIA NeMo / Parakeet – набор инструментов и моделей для задач Speech AI, включая автоматическое распознавание речи. NeMo является фреймворком для работы с большими языковыми, мультимодальными и речевыми моделями и распространяется под лицензией Apache 2.0. Модели Parakeet построены на базе NeMo, доступны открыто и могут использоваться для распознавания речи; например, Parakeet RNNT 1.1B распространяется под лицензией CC-BY-4.0.

Coqui STT – open-source инструмент для обучения и развертывания моделей speech-to-text. Он поддерживает потоковое распознавание, локальный инференс, работу с предобученными моделями и может использоваться как в исследовательских, так и в прикладных сценариях. При этом данный вариант стоит рассматривать осторожнее, так как проект менее активно развивается по сравнению с Whisper, Vosk или NVIDIA NeMo.

Таким образом, для модуля транскрибации могут быть рассмотрены разные решения: Whisper, Vosk, NVIDIA NeMo/Parakeet и Coqui STT. Они отличаются качеством распознавания, сложностью интеграции, требованиями к оборудованию, поддержкой языков и удобством локального запуска. На следующем этапе данные инструменты могут быть протестированы на одинаковом наборе аудиозаписей собеседований, после чего будет выбран наиболее подходящий вариант для использования в приложении.

2 Архитектура больших языковых моделей

Рассматривая архитектуру данных больших языковых моделей, следует отметить, что они все основаны на Transformer, преимущественно на варианте Transformer-декодера (Decoder-only Transformer). Идея Transformer была предложена в 2017 году компанией Google в статье «Attention is All You Need» и смысл ее заключается в механизме внимания для задач машинного перевода, но впоследствии изучения и развития этой архитектуры ее начали использовать и в других целях [18]. Данная архитектура используется в авторегрессионных языковых моделях, которые генерируют текст последовательно, предсказывая следующий токен на основе предыдущего контекста. При этом большинство выбранных моделей относится к плотным моделям, то есть при обработке запроса задействуются все параметры модели.

Главная идея классической архитектуры трансформера, представленная на рисунке 2, заключается в отказе от рекуррентных и сверточных нейронных сетей и использование только механизма внимания. Такой подход позволяет лучше распараллелить вычисления, быстрее обучать модель и эффективнее учитывать зависимости между разными частями текста, а также позволяет определить, какие токены в последовательности важны друг для друга.

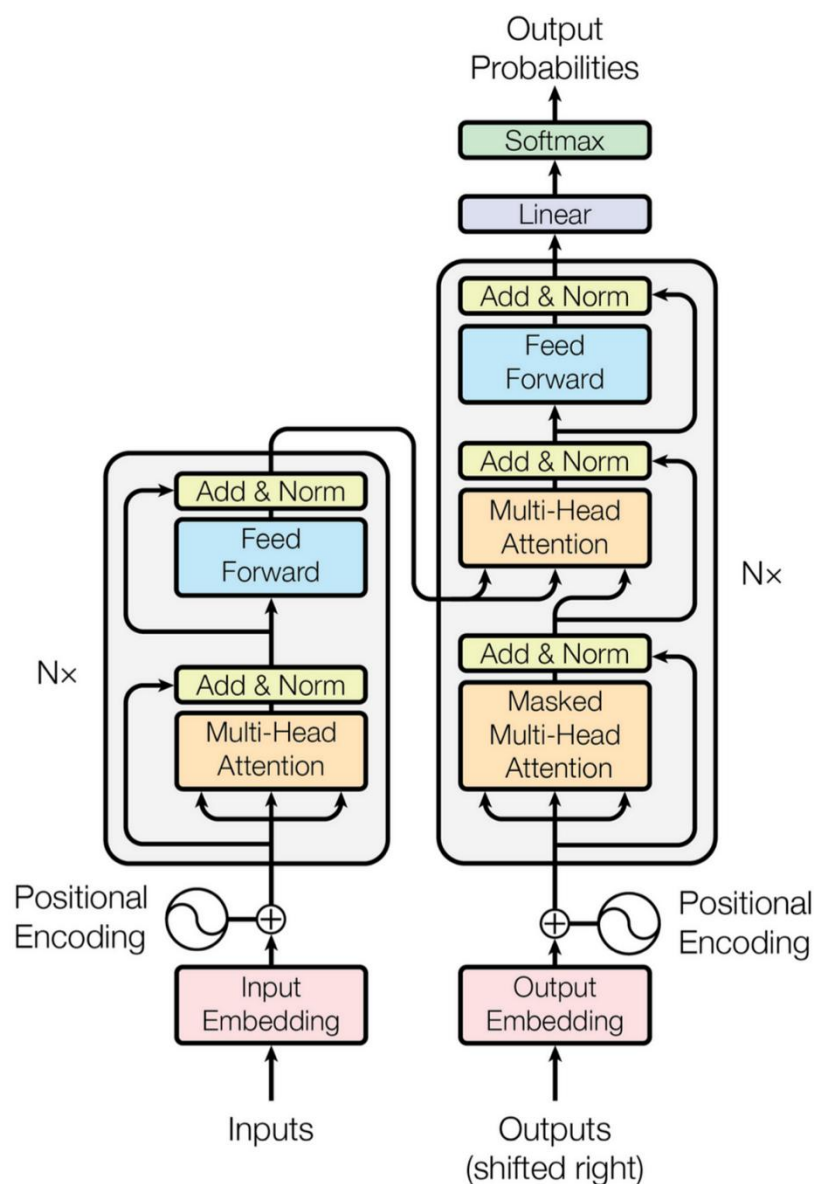


Рисунок 2 – Архитектура Transformer

Изначальная архитектура Transformer состоит из двух основных частей: энкодера и декодера. Энкодер получает входную последовательность токенов и преобразует ее в набор внутренних представлений. Декодер, используя эти представления и уже сгенерированные ранее токены, последовательно формирует выходную последовательность и формирует перевод на целевой язык.

2.1 Задача Encoder

Энкодер предназначен для обработки входной последовательности. Его важное отличие от RNN заключается в том, что он может обрабатывать токены параллельно, а не последовательно. На вход энкодеру поступают токены,

которые сначала преобразуются в эмбединги – числовые векторы, представляющие смысл токенов. После этого они проходят через блоки энкодера.

Основной слой энкодера – это self-attention, или самовнимание. Его задача состоит в том, чтобы обновить эмбединг каждого токена с учетом контекста, то есть всех остальных токенов предложения. Для каждого токена вычисляются веса важности остальных токенов, затем эти веса проходят через функцию softmax, после чего эмбединги агрегируются в новый вектор внимания. Этот вектор добавляется к исходному эмбедингу, благодаря чему токен получает информацию о своем контексте. Но в Transformer используется не один слой внимания, а multi-head attention, то есть многоголовое внимание. Его смысл состоит в том, что разные головы внимания могут анализировать разные связи между токенами (1). Например, одна голова может учитывать грамматические связи, другая – смысловые, третья – связь между местоимением и существительным.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O, \quad (1)$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)W^O. \quad (2)$$

где $head_i$ – отдельная голова внимания;

W_i^Q, W_i^K, W_i^V – обучаемые матрицы для формирования запросов, ключей и значений;

W^O – матрица объединения результатов.

Так как Transformer обрабатывает токены параллельно, сама модель не знает их порядок, поэтому используется позиционное кодирование. Оно добавляет к каждому эмбедингу информацию о позиции токена в последовательности. К примеру, предложения «dog runs horse» и «horse runs dog» состоят из одних и тех же токенов, но имеют разный смысл, поэтому порядок токенов обязательно нужно учитывать, это позволяет модели

указывать, в каком порядке расположены токены. Для этого используются синусоидальные позиционные кодировки, которые позволяют добавлять каждому токenu позицию в тексте (3-4):

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \quad (3)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right). \quad (4)$$

где pos – позиция токена в последовательности;

i – номер измерения вектора;

d_{model} – размерность внутреннего представления модели.

После слоя self-attention используется блок Add & Norm. Add означает сложение исходного эмбединга с результатом слоя внимания, то есть остаточную связь. Она нужна для того, чтобы модель не теряла исходную информацию о токене. Self-attention добавляет к токenu контекстную информацию, но исходное представление токена тоже важно сохранить. Поэтому модель не заменяет старый эмбединг новым, а складывает их.

Часть Norm означает нормализацию, которая стабилизирует значения внутри сети. В Transformer используется Layer Normalization, она приводит значения внутри каждого эмбединга к более стабильному виду. Это помогает избежать нестабильного обучения и делает работу глубоких сетей более устойчивой. LayerNorm нормализует значения каждого эмбединга отдельно и работает одинаково во время обучения и во время использования модели. То есть нормализация сначала приводит эмбединг к более стабильному масштабу, а затем с помощью параметров γ и β позволяет модели самой подобрать удобные значения (5):

$$x_i^e = \frac{x_i^e - \mu_i}{\sigma_i + \epsilon} * \gamma + \beta. \quad (5)$$

где μ_i – среднее значение координат эмбединга;
 σ_i – стандартное отклонение;
 ϵ – маленькое число для защиты от деления на ноль;
 γ и β – обучаемые параметры.

После нормализации в энкодере находится полносвязная часть, или feed-forward layer. Она применяется к каждому эмбедингу независимо и сохраняет размерность входа и выхода. Результат работы блока Encoder – это обновленные эмбединги токенов, причем каждый слой обрабатывает эмбединги параллельно.

2.2 Задача Decoder

Декодер Transformer предназначен для генерации выходной последовательности. Во время обучения в него можно подать сразу все токены целевого предложения, используя подход teacher forcing. После прохождения через декодер выходные эмбединги передаются в полносвязный слой и softmax, после чего получается распределение вероятностей следующего токена. Во время обучения декодеру можно подать сразу все токены target-предложения, а затем считать ошибку через cross-entropy loss. Декодер состоит из трех основных частей: masked self-attention, cross-attention и feed-forward layer.

Во время реального использования модели декодер работает авторегрессионно: сначала получает начальный токен, затем генерирует следующий, добавляет его к входу и повторяет процесс.

Masked self-attention отличается от обычного self-attention тем, что токен не может получать информацию из будущих токенов. Это необходимо, потому что при генерации текста модель не должна заранее знать правильное продолжение. Если во время обучения декодер видит всю целевую последовательность сразу, тогда модель фактически подсматривала бы правильный ответ. В реальном использовании такого будущего токена еще

нет, поэтому обучение было бы некорректным и в декодере запрещаются связи с будущими позициями. На рисунке 3 это изображено красными линиями.

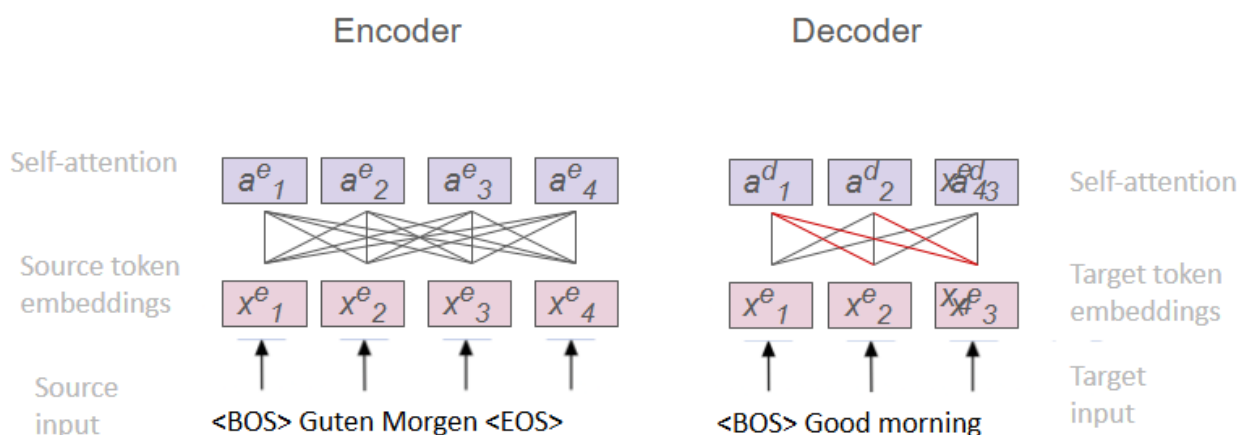


Рисунок 3 - Masked self-attention

То есть первый token может учитывать только себя, второй – первый и второй, третий – первые три и так далее. Благодаря этому декодер обучается в тех же условиях, в которых потом будет работать при генерации текста.

Вторая важная часть декодера – cross-attention, или перекрестное внимание. Это слой, в котором декодер получает информацию от энкодера. Если masked self-attention помогает декодеру учитывать уже сгенерированные токены, то cross-attention помогает ему учитывать входное предложение.

В задаче перевода это означает следующее: декодер генерирует перевод, но на каждом шаге может обращаться к эмбедингам исходного предложения. Например, при генерации английского слова декодер может посмотреть, какие слова во входной последовательности наиболее важны для текущего шага. Работает cross-attention похоже на обычное внимание, но запросы берутся из декодера, а ключи и значения – из энкодера. То есть декодер спрашивает у энкодера, какая часть исходного предложения сейчас важна.

В Transformer используется вариант внимания, который называется QKV-attention (Query Key Value). Запрос показывает, какую информацию ищет текущий token. Ключ показывает, какая информация содержится в другом токене. Значение содержит саму информацию, которую нужно передать дальше, если token оказался важным.

Перед вычислением внимания эмбединги умножаются на обучаемые матрицы W^Q, W^K, W^V . Это нужно для того, чтобы модель могла извлечь из исходных эмбедингов разные типы информации: отдельно для поиска сходства и отдельно для передачи полезного содержания (6):

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \quad (6)$$

где Q – матрица запросов;

K – матрица ключей;

V – матрица значений;

d_k – размерность ключей.

После этого применяется softmax, который превращает оценки в веса внимания. Затем веса умножаются на V , и получается итоговая информация, которую модель передает дальше.

2.3 Архитектура Decoder-only Transformer

Классическая архитектура Transformer изначально была разработана для задач машинного перевода и состояла из двух частей: энкодера и декодера. Однако современные большие языковые модели, предназначенные для генерации текста, чаще всего используют не всю архитектуру целиком, а только ее декодерную часть. Такой подход называется Decoder-only Transformer, то есть Transformer, состоящий только из декодера.

Главное отличие Decoder-only Transformer от классической архитектуры заключается в отсутствии энкодера и слоя cross-attention. В классическом Transformer энкодер обрабатывает входную последовательность, а декодер с помощью cross-attention обращается к результатам энкодера. В Decoder-only архитектуре отдельного энкодера нет, поэтому вся входная информация передается модели напрямую в виде единой текстовой последовательности. Например, в случае ассистента для проведения собеседований в одну

последовательность могут быть объединены системная инструкция, описание вакансии, вопрос, ответ кандидата и требуемый формат результата.

Вместо разделения на входную и выходную последовательность Decoder-only модель работает с одним общим контекстом. Она получает уже имеющиеся токены и предсказывает следующий токен. Поэтому такие модели называются авторегрессионными. После генерации нового токена он добавляется к исходному контексту, и модель снова предсказывает следующий токен, то есть не генерирует весь ответ сразу, а строит его последовательно, опираясь на уже имеющийся контекст.

Основным механизмом внутри Decoder-only Transformer является masked self-attention. Он необходим для того, чтобы модель при предсказании следующего токена не использовала информацию из будущих токенов. Это особенно важно при обучении, когда в модель может подаваться вся последовательность целиком. Если не использовать маску, модель могла бы видеть правильное продолжение и фактически “подсматривать” ответ. В реальном использовании будущих токенов еще не существует, поэтому обучение без маски привело бы к некорректному поведению модели.

Таким образом, классическая архитектура Transformer является основой для современных LLM, однако в большинстве генеративных языковых моделей используется именно ее декодерная разновидность – Decoder-only Transformer. Она сохраняет ключевые механизмы Transformer, такие как self-attention, multi-head attention, позиционное кодирование, нормализацию и полносвязные слои, но не использует энкодер и cross-attention. Это позволяет модели эффективно решать задачи генерации текста и работать с диалоговым контекстом.

3 Анализ аналогов

На рынке представлены решения для автоматизации проведения собеседований и оценки кандидатов. Некоторые системы ориентируются на работодателей и позволяют проводить видеоинтервью, оценивать ответы кандидатов, формировать отчеты и сокращать время первичного отбора. Другая группа предназначена для кандидатов и позволяют готовиться к собеседованиям, проходить пробные интервью и получать рекомендации по улучшению ответов.

Разрабатываемое приложение относится к классу интеллектуальных ассистентов для проведения собеседований. Его основная задача заключается в генерации вопросов, анализе ответов кандидата, формировании уточняющих вопросов и создании итогового отчета. Поэтому в качестве аналогов целесообразно рассмотреть системы, которые частично или полностью решают похожие задачи.

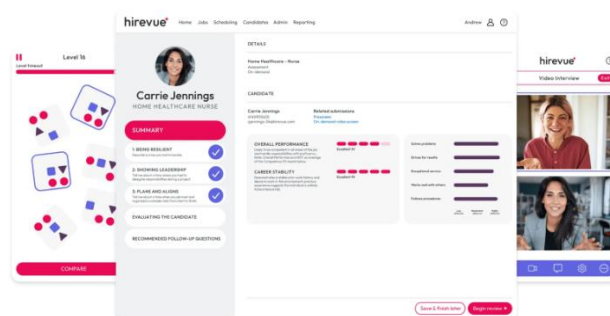
3.1 Приложение HireVue

HireVue (<https://www.hirevue.com/>) – это платформа для видеоинтервью, оценки навыков и автоматизации найма. Сервис ориентирован в первую очередь на компании и HR-отделы, которым необходимо проводить большое количество интервью и быстро сравнивать кандидатов между собой. На официальном сайте HireVue указано, что платформа предоставляет видеоинтервью с использованием искусственного интеллекта, оценку кандидатов и проверку навыков.

HireVue позволяет проводить структурированные видеоинтервью, использовать заранее подготовленные вопросы, а также применять автоматизированные оценки для анализа компетенций кандидатов (рисунок 4). Для технических специальностей платформа также предоставляет инструменты проверки профессиональных навыков, включая оценку технических знаний и гибкие навыки.

Hire With **Confidence**. Trusted **Skills**. Trusted **AI**.

Hirevue gives you validated, defensible, and predictive insights, so you gain the clarity and confidence to hire better and grow your business.

[Take a tour](#)[Book a demo](#)

Trusted by the **best**.



Рисунок 4 – Платформа HireVue

Платформа подходит для массового найма, стандартизации интервью и сравнения кандидатов по единым критериям. К слабым сторонам можно отнести закрытость платформы, зависимость от облачной инфраструктуры и отсутствие возможности гибко изменять внутреннюю логику работы модели под конкретную исследовательскую задачу. Для разрабатываемого приложения HireVue является важным аналогом, так как решает похожую задачу автоматизации оценки кандидатов.

3.2 Приложение Interviewer.AI

Interviewer.AI (<https://interviewer.ai/>) – это платформа для проведения структурированных AI-интервью на этапе первичного отбора кандидатов. Сервис задает кандидатам вопросы, связанные с конкретной вакансией, и оценивает их по единым критериям (рисунок 5). На сайте Interviewer.AI указано, что каждый кандидат получает одинаковые вопросы, соответствующие роли, а оценка проводится по согласованным критериям, что помогает быстрее отбирать подходящих кандидатов без ручной проверки резюме.

Платформа поддерживает структурированные интервью и может использоваться для оценки как профессиональных, так и коммуникативных навыков. В описании также указано, что система может использовать преобразование речи в текст, компьютерное зрение, анализ голоса и обработка естественного языка для получения аналитики по ответам кандидатов.

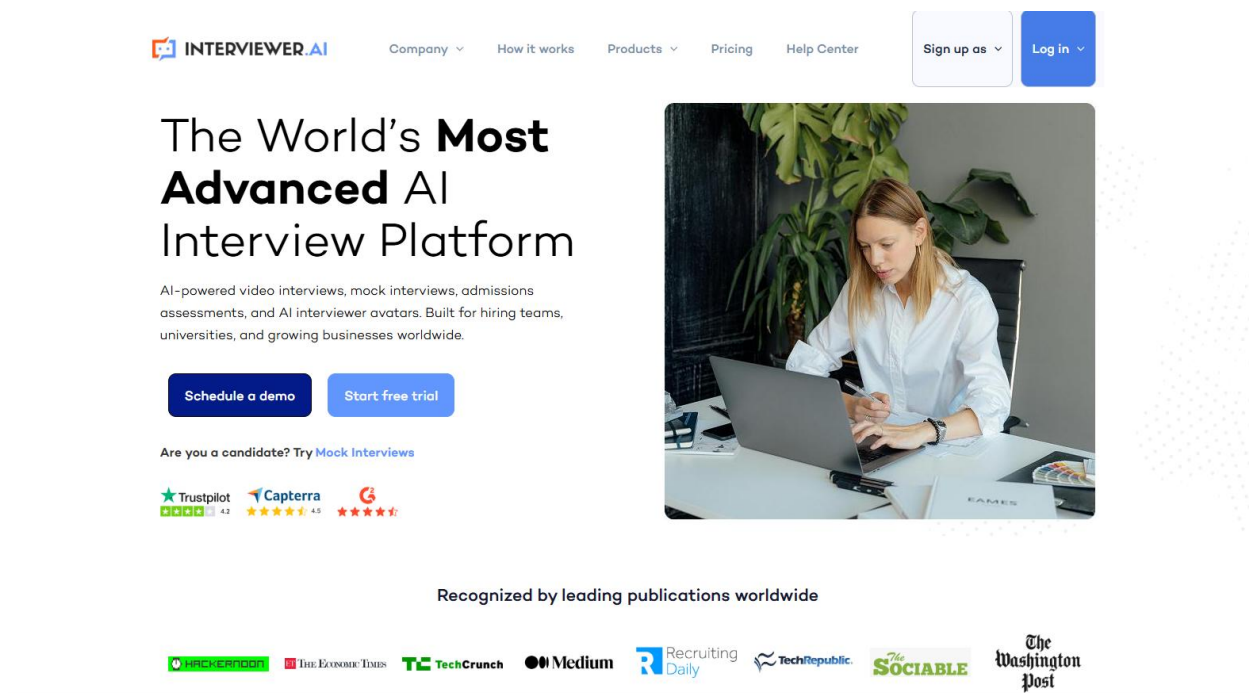


Рисунок 5 – Платформа Interviewer.AI

Преимуществом Interviewer.AI является возможность проводить интервью, системам задает вопросы и оценивает ответы кандидатов. Также важным преимуществом является стандартизация процесса, так как все кандидаты проходят интервью по одинаковой схеме.

3.3 Приложение Willo

Willo (<https://www.willo.video/>) – это платформа для асинхронных видеоинтервью и первичного отбора кандидатов. Работодатель может подготовить набор вопросов, а кандидат отвечает на них в удобное время в формате видео. На сайте Willo указано, что платформа помогает команде по найму быстрее проводить отбор, проводить тщательный отбор, выявлять наиболее подходящего кандидата и принимать решения всей командой (рисунок 6).

Willo подходит для случаев, когда компании нужно быстро собрать ответы кандидатов без необходимости назначать живое интервью. Такой формат удобен при большом количестве откликов и распределенных командах.

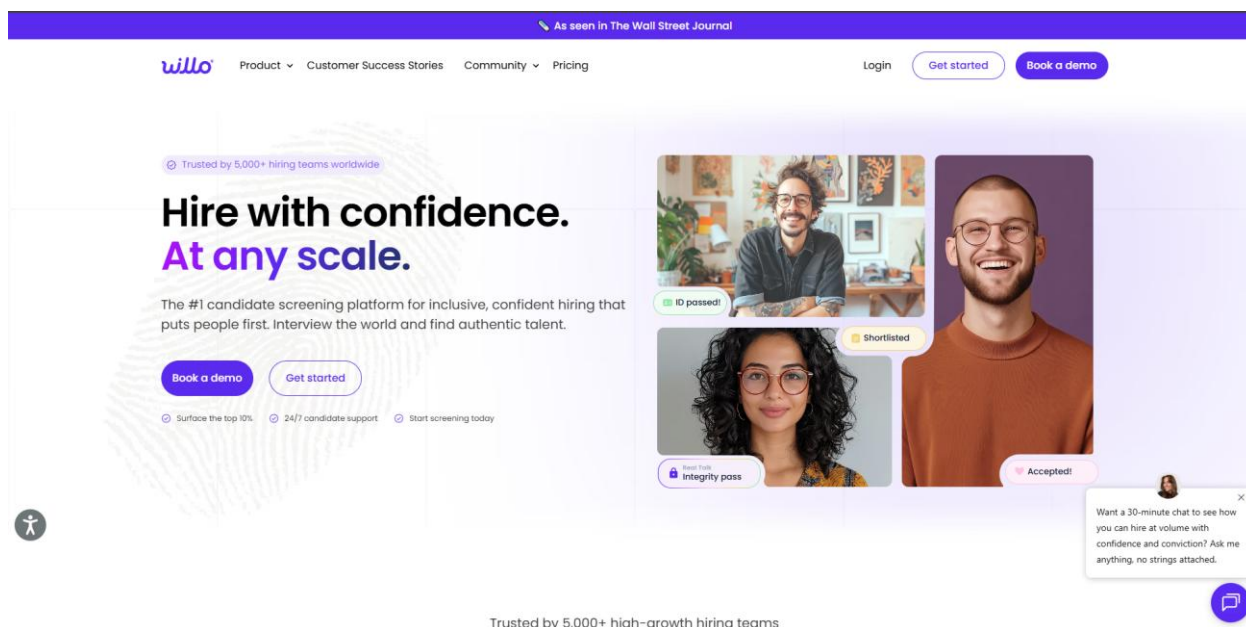


Рисунок 6 – Платформа Willo

Сильной стороной Willo является удобство массового первичного отбора и асинхронный формат интервью. Кандидат может записать ответы в удобное время, а команда позже просматривает и оценивает их. Слабой стороной является меньшая интеграция с искусственным интеллектом, из-за этого необходимо команде по найму самим оценивать ответы кандидатов.

3.4 Приложение Final Round AI

Final Round AI (<https://www.finalroundai.com/>) – это AI-инструмент для подготовки к интервью и сопровождения кандидата во время собеседования. На официальном сайте указано, что Interview Copilot работает в реальном времени на разных платформах для интервью, а ответы генерируются с учетом резюме и описания вакансии пользователя (рисунок 7). Также сервис поддерживает транскрибацию и анализ ответов после интервью.

В отличие от HireVue или Interviewer.AI, Final Round AI ориентирован на кандидата, а не на работодателя. Его задача помочь пользователю лучше

отвечать на вопросы, структурировать мысли и готовиться к следующим этапам интервью.

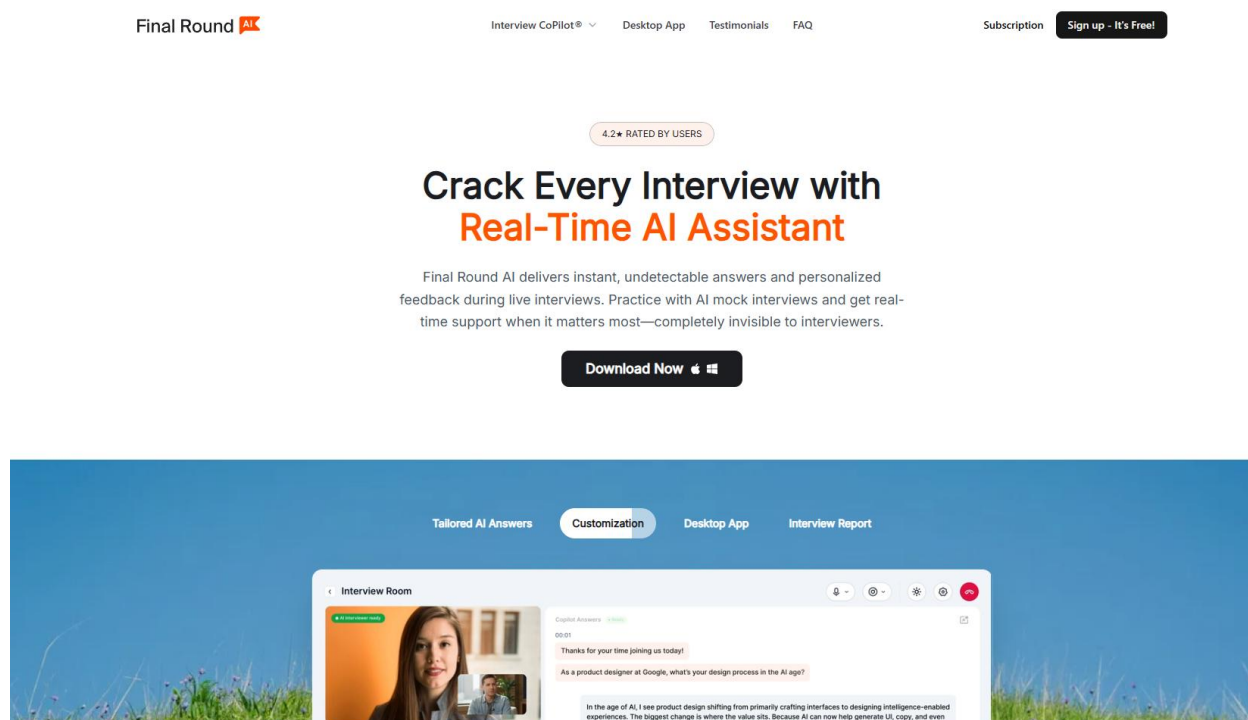


Рисунок 7 – Приложение Final Round AI

Основной целью Final Round AI является персонализация под резюме и вакансию, а также наличие функций подготовки и анализа интервью. Однако этот продукт не предназначен для объективной оценки кандидата работодателем, так как во время интервью есть возможность использовать данный сервис в своих целях, чтобы отвечать на вопросы интервью на реальном собеседовании.

4 Архитектура приложения

Разрабатываемое приложение представляет web-приложение для проведения технического интервью с использованием больших языковых моделей. Приложение поддерживает два основных сценария работы: обычное AI-собеседование, при котором пользователь взаимодействует с ассистентом через чат, и HR-интервью по публичной ссылке, при котором HR-специалист заранее создает набор вопросов, а кандидат проходит интервью без обязательной регистрации в системе.

Для реализации приложения была выбрана микросервисная архитектура, которая представлена на рисунке 8. Такой подход позволяет разделить систему на независимые компоненты, каждый из которых отвечает за свою область: авторизацию, проведение интервью, оценку ответов, транскрибацию речи, хранение отчетов и взаимодействие с большой языковой моделью. Преимуществом микросервисной архитектуры является возможность независимо разрабатывать, тестировать, масштабировать и дорабатывать отдельные части системы [19]. Например, сервис транскрибации речи может быть изменен без изменения сервиса авторизации, а интеллектуальный сервис с AI частью может использовать другую языковую модель без полной переработки backend-части приложения.

Бизнес-логика приложения реализована на Java с использованием Spring, так как данная технология хорошо подходит для построения надежных backend-сервисов, работы с базой данных, авторизацией и REST API. AI-компоненты вынесены в отдельные Python-сервисы, так как Python имеет развитую экосистему библиотек для работы с большими языковыми моделями, транскрибацией речи и машинным обучением.

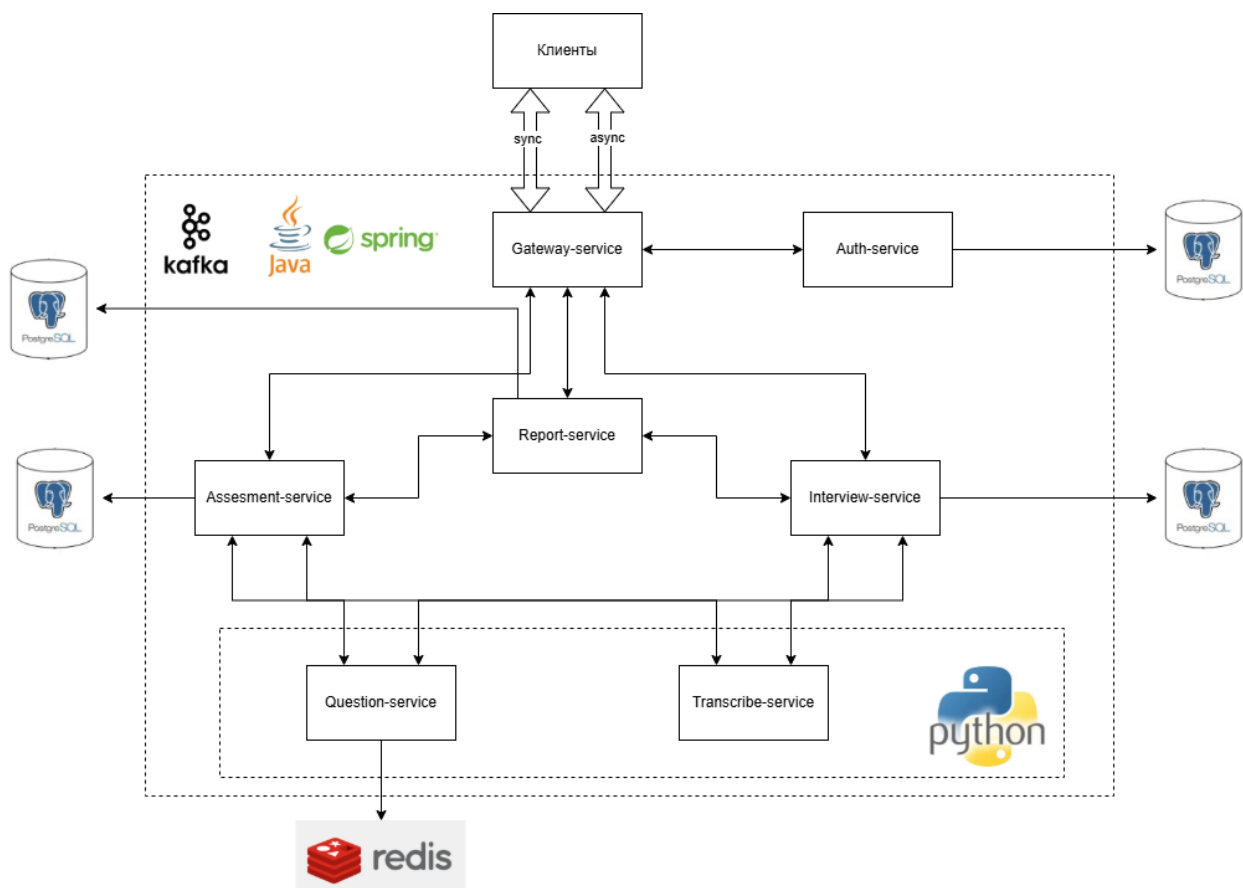


Рисунок 8 - Архитектура приложения

Архитектура приложения состоит из нескольких уровней: клиентского уровня, входного уровня, Java-микросервисов, Python AI-сервисов и уровня хранения данных. Пользователи взаимодействуют только с frontend-частью, а все запросы к backend проходят через API Gateway. Такой подход позволяет централизованно обрабатывать маршрутизацию, авторизацию и доступ к защищенным или публичным маршрутам.

4.1 Клиентская часть приложения

Клиентская часть приложения представляет собой единый frontend, разработанный с использованием React и TypeScript. Frontend отвечает за взаимодействие пользователя с системой и предоставляет интерфейс для всех основных сценариев работы приложения.

Через клиентскую часть пользователь может зарегистрироваться, авторизоваться, пройти обычное AI-собеседование, отправлять текстовые и голосовые ответы, просматривать вопросы, получать обратную связь и итоговый отчет. HR-пользователь может создавать интервью, указывать

список вопросов, задавать требования к прохождению, формировать публичную ссылку для кандидата и просматривать результаты прохождения интервью. Кандидат, в свою очередь, может открыть публичную ссылку, ввести свои данные и пройти интервью без полноценной регистрации в системе.

Frontend не обращается напрямую к каждому микросервису. Все запросы отправляются через API Gateway, который выступает единой точкой входа в backend-систему. Это упрощает структуру клиентской части, так как frontend не должен знать внутреннее расположение всех сервисов и особенности их взаимодействия.

4.2 Микросервис API Gateway

API Gateway является входной точкой во всю backend-систему. Он принимает запросы от frontend-части и перенаправляет их в соответствующие микросервисы. Данный сервис реализован на Java с использованием Spring.

Основная задача API Gateway заключается в маршрутизации запросов. Например, запросы, связанные с регистрацией и входом пользователя, направляются в auth-service, запросы обычного AI-собеседования – в interview-service, а запросы HR-интервью – в assessment-service. Такой подход позволяет скрыть внутреннюю структуру backend от frontend-приложения.

Кроме маршрутизации, API Gateway выполняет функции безопасности. Он обрабатывает CORS, проверяет JWT-токены для защищенных маршрутов и пропускает публичные маршруты без авторизации. Это важно для сценария HR-интервью по ссылке, так как кандидат может проходить интервью без регистрации, но при этом административные функции HR должны оставаться защищенными.

4.3 Микросервис Auth Service

Данный микросервис отвечает за регистрацию пользователей, авторизацию и управление токенами доступа. Сервис реализован на Java с использованием Spring и выделен отдельно, так как безопасность и управление аккаунтами являются самостоятельной зоной ответственности.

Использование отдельного сервиса авторизации позволяет отделить логику безопасности от логики проведения интервью. Например, `interview-service` и `assessment-service` не занимаются регистрацией или проверкой пароля пользователя, а получают уже обработанную информацию через Gateway. Это делает архитектуру более гибкой и упрощает сопровождение приложения.

4.4 Микросервис Interview Service

`Interview Service` отвечает за обычный пользовательский сценарий AI-собеседования. Данный сервис реализован на Java с использованием Spring и является центральным сервисом для режима, в котором пользователь напрямую взаимодействует с AI-ассистентом через чат.

В обычном AI-собеседовании пользователь может указать профессию, уровень подготовки, количество вопросов и другие параметры интервью. После этого `interview-service` создает сессию собеседования и передает необходимые данные в Python-сервис `question-engine-service`, который отвечает за генерацию вопросов и оценку ответов.

Если пользователь отвечает голосом, `frontend` записывает аудио и отправляет его в `backend`. `Interview Service` принимает аудиоданные, передает их в `transcribe-service` и получает распознанный текст. После этого текст ответа отправляется в `question-engine-service` для анализа и оценки. Таким образом, `Interview Service` связывает пользовательский сценарий интервью с AI-компонентами системы.

4.5 Микросервис Assessment Service

`Assessment Service` отвечает за HR-функционал приложения. Он реализован на Java с использованием Spring и предназначен для создания и прохождения интервью по публичной ссылке.

В данном сценарии HR-пользователь авторизуется в системе, создает интервью, задает список вопросов, требования к прохождению и параметры оценки. После создания интервью сервис формирует публичную ссылку, которую можно отправить кандидату. Кандидат открывает ссылку, вводит имя

и email, после чего проходит интервью без необходимости регистрации в системе.

4.6 Микросервис Question Engine Service

Question Engine Service является AI-микросервисом для взаимодействия с моделью приложения. Сервис реализован на Python, так как Python предоставляет удобные библиотеки и инструменты для работы с большими языковыми моделями.

Основная задача данного сервиса – это генерация вопросов и оценка ответов кандидатов. Он используется как в обычном AI-собеседовании, так и в HR-интервью по публичной ссылке. Однако в этих сценариях его роль немного отличается. В обычном AI-собеседовании сервис может управлять логикой генерации вопросов в рамках всей сессии. В HR-интервью вопросы заранее задаются HR-пользователем, поэтому question-engine-service используется преимущественно для оценки отдельных ответов кандидата.

4.7 Микросервис Transcribe Service

Transcribe Service отвечает за преобразование речи в текст. Сервис реализован на Python и выполняет только одну задачу – автоматическую транскрибацию аудиоответов кандидатов.

В процессе работы frontend-чатсть записывает аудиоответ пользователя в формате webm и отправляет его в backend в формате base64. Далее Java-сервис, в зависимости от сценария, передает аудио в transcribe-service. Сервис декодирует audio base64, выполняет распознавание речи с помощью специальной модели для транскрибации голоса в текст и возвращает обычный текст.

4.8 Микросервис Report Service

Report Service отвечает за хранение и выдачу итоговых отчетов. Сервис реализован на Java с использованием Spring и выделен отдельно, так как отчеты являются самостоятельной частью приложения.

Report Service может хранить результаты обычных AI-интервью и HR-интервью, предоставлять историю отчетов пользователю или HR, а также

отдавать публичные результаты по ссылке. При этом сервис не обязан напрямую обращаться к AI-моделям. Оценку ответов и технический анализ выполняют question-engine-service и assessment-service, а report-service отвечает за сохранение и публикацию итоговых данных.

5 Реализация приложения

Для реализации данного приложения, как было указано в предыдущих пунктах работы, необходимо реализовать взаимодействие с большой языковой моделью, которая будет решать задачи связанные с проведением интервью и оценкой знаний кандидата, а также интегрировать нейросетевую модель, которая будет выполнять функцию распознавания речи.

5.1 Реализация инструкций

Чтобы выявить подходящую большую языковую модель, необходимо сначала задать общие инструкции, которые будут описывать требования, задачи и цели запроса к LLM. Эти инструкции не будут отличаться при первичном тестировании с целью уравнивания условий всех моделей для более корректного сравнения, так как каждая из них получает одинаковое описание роли, одинаковые критерии оценки и одинаковый формат ожидаемого ответа.

Использование заранее заданных инструкций также позволяет стандартизировать поведение модели. Это особенно важно, так как разные LLM могут по-разному интерпретировать один и тот же запрос. Поэтому в инструкциях явно указывается роль модели, язык ответа, формат вывода, правила оценки и ограничения, которые модель должна соблюдать.

Важным требованием ко всем инструкциям является возврат результата в формате JSON. Это необходимо для того, чтобы приложение могло автоматически обрабатывать данные, полученные от модели: извлекать оценки, список сильных и слабых сторон, следующий вопрос, итоговый результат и другие параметры.

5.1.1 Реализация инструкции генерации вопросов

Для функции генерации вопросов необходимо конкретно прописать цель, которая преследуется при выполнении данной функции, указать в промте кем является большая языковая модель, ее задачу, которую она должна выполнить и обязательные требования для дальнейшей работы с данными, которые будут получено от модели. Была задана следующая инструкция:

```

async def generate_questions(
    self,
    profession: str,
    level: str,
    num_questions: int,
    focus_areas: list[str] | None = None,
) -> list[str]:
    focus_text = ""
    if focus_areas:
        focus_text = f"Сфокусируйся на аспектах: {'',
'.join(focus_areas)}"

```

```

    prompt = f"""

```

Ты - технический интервьюер.

Сгенерируй {num_questions} технических вопросов для собеседования.

Позиция: {profession}

Уровень: {level}

{focus_text}

Требования:

- все вопросы должны быть только на русском языке;
- вопросы должны быть строго техническими;
- не задавай общие HR-вопросы;
- не начинай с вопроса про общий опыт работы;
- вопросы должны соответствовать уровню {level};
- верни только JSON;
- формат ответа: массив строк.

Пример:

```

[
    "Что такое GIL в Python и как он влияет на многопоточность?",
    "Чем отличаются list, tuple и set в Python?",
    "Как работает async/await в Python?"
]
"""

```

Важно отметить, что требование к JSON-формату используется во всех последующих инструкциях. Это позволяет не только получать текстовый ответ от модели, но и программно обрабатывать его внутри приложения.

5.1.2 Реализация инструкции определения параметров собеседования

Инструкция определения параметров собеседования представляет собой извлечение данных содержания интервью из запроса пользователя, так как пользователь может написать запрос в свободной форме, а именно то, на какую позицию хочет пройти собеседование кандидат, сколько вопросов

необходимо будет сгенерировать и уровень сложности прохождения интервью. Для этого была написана следующая инструкция:

```
async def parse_user_message(self, message: str) ->
ParsedInterviewRequest:
    prompt = f"""
Проанализируй сообщение кандидата и определи параметры
собеседования.

Сообщение: "{message}"

Определи:
1. Профессию/позицию
2. Уровень (junior/middle/senior/lead), если не указан - middle
3. Количество вопросов, если не указано -
{settings.default_questions_count}

Верни только JSON:
{{
    "profession": "строка",
    "level": "junior|middle|senior|lead",
    "num_questions": число
}}
```

Данная инструкция позволяет преобразовать неструктурированный пользовательский запрос в набор параметров, которые далее используются при генерации вопросов и создании интервью.

5.1.3 Реализация инструкции оценки ответа

Одна из ключевых инструкций для работы приложения является оценка ответа кандидата. Чтобы корректно понять, как хорошо ответил кандидат необходимо написать корректную инструкцию. Здесь нужно точно определить, что именно нужно выполнить и сгенерировать большой языковой модели, на что обращать внимания при анализе, чтобы более точно оценить ответ кандидата.

Для усложнения задачи был сформирован список конкретных критериев, по которым будет проводить оценку модель с целью выявления наиболее точной и подходящей LLM для данного приложения. Вместо одной общей оценки модель должна выставить несколько частных оценок: техническую правильность, полноту, логичность, релевантность вопросу, грамотность, уверенность кандидата и скорость ответа. Такой подход

позволяет более детально проанализировать, насколько хорошо конкретная большая языковая модель справляется с оценкой ответа.

Данная инструкция оценивает отдельно один ответ на вопрос, чтобы в дальнейшем полностью оценить все ответы кандидата в процессе интервью.

```
def _build_prompt(self, payload: EvaluateAnswerRequest) ->
str:
    response_time_text = ""
    if payload.response_time_seconds is not None:
        response_time_text = (
            f"Время ответа кандидата:
{payload.response_time_seconds:.2f} секунд.\n"
            "Важно: response_speed_score должен быть оценкой
от 0 до 10, а не количеством секунд.")
    confidence_text = ""
    if payload.confidence_score is not None:
        confidence_text = (
            f"Оценка уверенности из внешнего источника:
{payload.confidence_score} из 10."
        )
    return f"""
Ты - строгий технический интервьюер.
Позиция:
{payload.profession}
Уровень:
{payload.level or "не указан"}

Вопрос:
{payload.question}
Ответ кандидата:
{payload.answer}
Тип ответа:
{payload.input_type or "text"}
Дополнительные данные:
{response_time_text}
{confidence_text}
Оцени ответ кандидата строго по шкале от 0 до 10.
Правила оценки:
- overall_score: общая оценка.
- correctness_score: насколько ответ технически правильный.
- completeness_score: насколько ответ полный.
- clarity_score: насколько ответ логичный и понятный.
- relevance_score: насколько ответ относится к вопросу.
- grammar_score: грамотность русского языка.
- confidence_score: уверенность кандидата по тексту ответа.
- response_speed_score: оценка скорости ответа от 0 до 10.
ВАЖНО:
- Отвечай только на русском языке.
- Верни только JSON.
- Не добавляй markdown.
- Не добавляй пояснения вне JSON.
```

- Если ответ не относится к вопросу, `relevance_score` = 0-1 и `overall_score` не выше 2.
- Если кандидат пишет "не знаю", "не могу", "хз", `overall_score` не выше 2.
- Если ответ содержит шутку, оффтоп или бессмысленный текст, `overall_score` = 0-1.
- Если ответ частично правильный, но неполный, ставь 4-6.
- Если нет примеров там, где они нужны, снижай `completeness_score`.
- `response_speed_score` - это строго оценка от 0 до 10, НЕ секунды.
- Все числовые поля должны быть от 0 до 10.

Формат ответа:

```
{
  "overall_score": 0,
  "correctness_score": 0,
  "completeness_score": 0,
  "clarity_score": 0,
  "relevance_score": 0,
  "grammar_score": 0,
  "confidence_score": 0,
  "response_speed_score": 0,
  "feedback": "строка на русском языке",
  "strengths": ["строка на русском языке"],
  "weaknesses": ["строка на русском языке"],
  "decision": "NEXT",
  "follow_up_question": null
}
```

Такое количество оценок и параметров позволит выявить, насколько хорошо конкретная большая языковая модель способна справляться с анализом данных и генерацией ответа при оценке знаний.

5.1.4 Реализация инструкции оценки ответа в процессе интервью

Инструкция анализа ответа используется в процессе прохождения интервью, когда необходимо не только выставить числовую оценку, но и определить дальнейшее действие системы. В отличие от базовой оценки ответа, здесь модель должна принять решение: перейти к следующему вопросу или задать уточняющий вопрос кандидату.

Данный механизм необходим для приближения работы приложения к реальному собеседованию. В обычном интервью интервьюер не всегда сразу переходит к следующему вопросу. Если ответ кандидата неполный, но содержит частично правильную мысль, интервьюер может задать дополнительный вопрос, чтобы уточнить понимание темы.

Для анализа ответа была реализована следующая инструкция:

```
async def analyze_answer(
    self,
    profession: str,
    question: str,
    answer: str,
    response_time_seconds: float | None = None,
    confidence_score_from_audio: float | None = None,
) -> AnswerAnalysis:
    response_time_text = (
        f"Время ответа кандидата: {response_time_seconds}
секунд."
        if response_time_seconds is not None else
        "Время ответа неизвестно."
    )

    confidence_text = (
        f"Оценка уверенности речи по аудио:
{confidence_score_from_audio} из 10."
        if confidence_score_from_audio is not None else
        "Данных об уверенности речи нет."
    )
```

```
    prompt = f"""
Ты - строгий технический интервьюер на позицию {profession}.
```

```
Вопрос:
{question}
```

```
Ответ кандидата:
{answer}
```

```
Дополнительные данные:
{response_time_text}
{confidence_text}
```

Оцени ответ кандидата по шкале от 0 до 10 и верни только JSON.

Правила:

- overall_score: общая оценка
- correctness_score: насколько ответ правильный
- completeness_score: насколько ответ полный
- clarity_score: насколько ответ логичный и понятный
- relevance_score: насколько ответ соответствует вопросу
- grammar_score: насколько грамотно написан ответ на русском языке
- confidence_score: уверенность кандидата; если нет данных, оцени по тексту косвенно
- response_speed_score - это ИМЕННО оценка по шкале от 0 до 10, где:
 - 0-2: очень медленная реакция
 - 3-4: медленная реакция

- 5-6: средняя скорость
- 7-8: хорошая скорость
- 9-10: очень быстрая и уверенная реакция
- feedback: краткий содержательный фидбек
- strengths: список сильных сторон
- weaknesses: список слабых сторон
- decision: NEXT или FOLLOW_UP
- follow_up_question: если ответ неполный, задай уточняющий вопрос, иначе null

Будь строгим:

- если ответ неполный, снижай балл
- если нет примеров, можно снижать балл
- если есть фактические ошибки, сильно снижай correctness_score
- если русский язык слабый, снижай grammar_score

Оцени ответ кандидата строго по шкале от 0 до 10:

- 0-2: ответ полностью неверный или не относится к вопросу
- 3-4: ответ очень слабый, содержит ошибки
- 5-6: частично правильный, но неполный
- 7-8: хороший ответ, но есть неточности
- 9-10: полный и точный ответ

ВАЖНО:

- если ответ не относится к вопросу → ставь 0-2
- если ответ содержит бред или оффтоп → ставь 0
- не будь вежливым, оцени строго

ВАЖНО:

- Отвечай СТРОГО на русском языке
- ВСЕ текстовые поля (feedback, strengths, weaknesses, follow_up_question) должны быть ТОЛЬКО на русском языке
- Запрещено использовать английский язык в тексте
- Если ответ не на русском - это ошибка

Правила оценки:

- если ответ не относится к вопросу, relevance_score = 0-1 и overall_score не может быть выше 2
- если кандидат пишет "не знаю", overall_score не может быть выше 2
- если ответ содержит оффтоп, шутку вместо ответа или бессмысленный текст, overall_score = 0-1
- если answer correctness_score < 3, feedback не должен содержать фразы "в целом правильный"
- не завышай оценку из вежливости

Формат ответа:

```
{
  "overall_score": число,
  "correctness_score": число,
  "completeness_score": число,
  "clarity_score": число,
```

```

"relevance_score": число,
"grammar_score": число,
"confidence_score": число,
"response_speed_score": число,
"feedback": "строка",
"strengths": ["строка"],
"weaknesses": ["строка"],
"decision": "NEXT или FOLLOW_UP",
"follow_up_question": "строка или null"
}}
"""

```

Данная инструкция позволяет не только получить оценку ответа, но и управлять логикой интервью. Если ответ кандидата является достаточно полным, модель возвращает решение NEXT, после чего система переходит к следующему вопросу. Если ответ частично правильный, но требует уточнения, модель возвращает FOLLOW_UP и формирует дополнительный вопрос.

5.1.5 Реализация инструкции остановки интервью

Во время прохождения интервью пользователь может захотеть завершить собеседование досрочно. Для обработки таких ситуаций была реализована отдельная инструкция определения намерения пользователя. Она необходима для того, чтобы система могла отличить обычный ответ на вопрос от команды остановки интервью.

Например, если кандидат пишет «давай закончим», «стоп», «хватит», система должна распознать это не как ответ на технический вопрос, а как намерение завершить интервью. При этом завершение интервью не должно происходить случайно, поэтому может использоваться дополнительное подтверждение.

Данная инструкция способна выявить возможности большой языковой модели погружения в контекст и способность анализировать дополнительные требования. Для этого было реализовано следующее:

```

async def detect_user_intent(
    self,
    answer: str,
    awaiting_stop_confirmation: bool = False,
) -> UserIntentResult:
    prompt = f"""

```

Определи намерение пользователя по его последнему сообщению.

Сообщение пользователя:

```
"{answer}"
```

Контекст:

- Сейчас идет собеседование.
- `awaiting_stop_confirmation = {awaiting_stop_confirmation}`

Допустимые значения `intent`:

- `ANSWER`: пользователь отвечает на вопрос
- `STOP_INTERVIEW`: пользователь хочет остановить интервью
- `CONFIRM_STOP`: пользователь подтверждает, что хочет завершить интервью
- `CANCEL_STOP`: пользователь отменяет завершение и хочет продолжить интервью

Правила:

- Если пользователь пишет "стоп", "давай закончим", "хватит", "завершим", то это `STOP_INTERVIEW`.
- Если `awaiting_stop_confirmation = true` и пользователь пишет "да", "подтверждаю", "заканчиваем", то это `CONFIRM_STOP`.
- Если `awaiting_stop_confirmation = true` и пользователь пишет "нет", "продолжим", "не надо", то это `CANCEL_STOP`.
- Во всех остальных случаях это `ANSWER`.

Верни только JSON:

```
{{  
  "intent": "ANSWER",  
  "confidence": 0.95  
}}
```

В итоге реализация инструкций позволяет стандартизировать взаимодействие приложения с большой языковой моделью. Каждая инструкция отвечает за отдельную задачу: генерацию вопросов, определение параметров интервью, оценку ответа, анализ ответа в процессе собеседования и распознавание намерения пользователя. Такой подход делает работу приложения более предсказуемой и позволяет сравнивать разные LLM в одинаковых условиях.

5.2 Тестирование больших языковых моделей

Тестирование больших языковых моделей будет проходить для всех по одному сценарию. Сначала будут использоваться те инструкции, которые были заданы в самом начале, если модели будет недостаточно объяснений, то эти инструкции будут корректироваться, чтобы добиться максимально правильного результата от конкретной модели.

В процессе тестирования оценивались следующие параметры: корректность генерации технических вопросов, соответствие вопросов выбранной профессии и уровню кандидата, способность модели анализировать ответы пользователя, корректность выставления оценок, умение задавать уточняющие вопросы, поддержка JSON-формата, реакция на некорректные ответы, возможность завершить интервью по желанию пользователя, а также качество итогового отчета. Отдельное внимание уделялось проблемам заикливания, когда модель продолжала задавать уточняющие вопросы даже после полного ответа кандидата.

По итогам тестирования были выделены модели, наиболее подходящие для реализации приложения ассистента проведения собеседований, которые требуются для реализации приложения данной магистерской работы.

5.2.1 Тестирование Qwen3-8B

Модель Qwen3-8B в целом показала хорошие результаты при генерации вопросов и анализе ответов кандидата. На этапе генерации вопросов были замечены отдельные проблемы с формулировками: некоторые вопросы могли быть составлены не совсем корректно или требовали уточнения. Однако в большинстве случаев модель генерировала релевантные технические вопросы, соответствующие выбранной профессии (рисунок 9).

```
{
  "session_id": "7a73460e5f33",
  "profession": "Java Developer",
  "level": "middle",
  "model_name": "qwen3:8b",
  "questions": [
    {
      "number": 1,
      "text": "Как работает паттерн Singleton в Java и какие проблемы он может вызывать в многопоточном окружении?"
    },
    {
      "number": 2,
      "text": "Чем отличается паттерн Observer от паттерна Observer в Java и как он реализован в библиотеках?"
    },
    {
      "number": 3,
      "text": "Какие механизмы Java используют для синхронизации потоков и как они работают?"
    }
  ]
}
```

Рисунок 9 – Генерация вопросов модели Qwen

Одним из преимуществ Qwen3-8B стала способность достаточно хорошо распознавать некорректные ответы пользователя. Если кандидат давал ответ, не относящийся к вопросу, или демонстрировал незнание темы, модель корректно снижала оценку и давала соответствующую обратную связь, это представлено на рисунке 10. Также модель стабильно возвращала данные в формате JSON, что является важным требованием для дальнейшей программной обработки результатов.

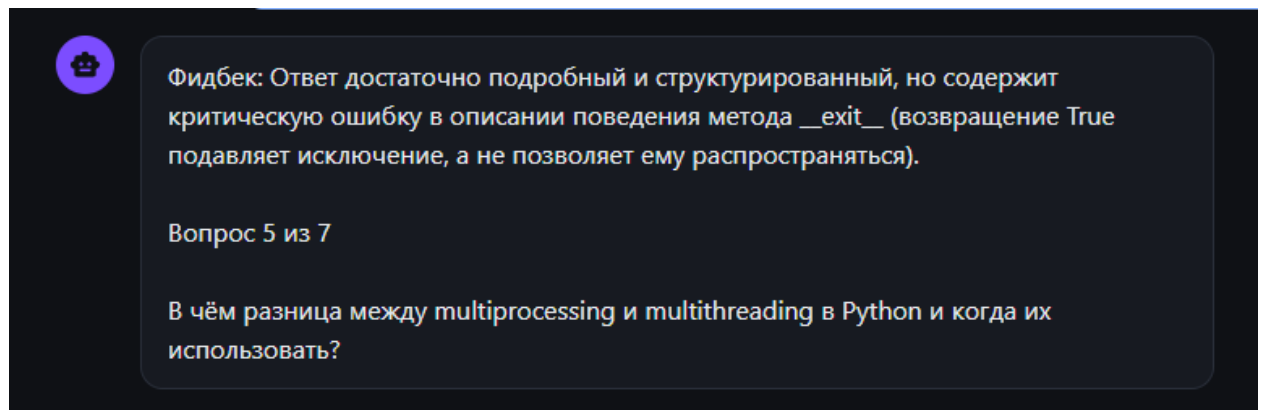


Рисунок 10 – Анализ ответа моделью Qwen

Основная проблема, выявленная на начальном этапе, заключалась в периодическом заикливании. В некоторых случаях модель продолжала задавать уточняющие вопросы, хотя ответ кандидата уже был достаточным для перехода к следующему вопросу. После корректировки инструкции данная проблема была устранена: модель стала корректно определять момент, когда нужно задать уточняющий вопрос, а когда следует перейти дальше. Также неплохо модель оценивает интервью, полный итоговый отчет представлен на рисунке 11.

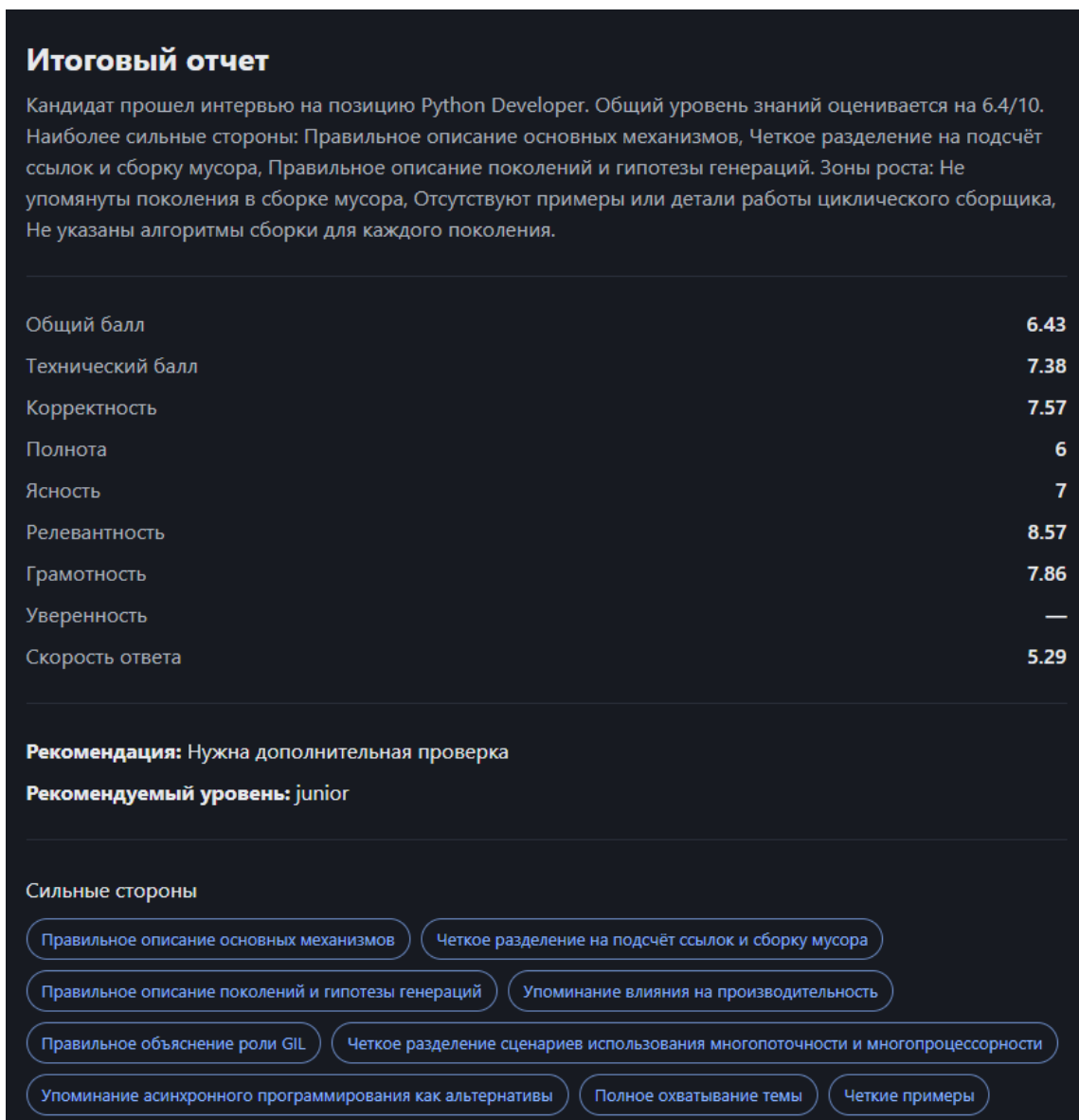


Рисунок 11 – Анализ интервью моделью Qwen

После исправления инструкций Qwen3-8B стала работать значительно стабильнее. Модель корректно задавала вопросы, могла остановить интервью по запросу пользователя, не уходила в заикливание и достаточно точно формировала итоговую оценку. Однако анализ ответов иногда был менее развернутым, чем у лучших моделей, и обратная связь могла быть достаточно краткой. Поэтому Qwen3-8B можно рассматривать как рабочий вариант, но не как наиболее сильную модель среди протестированных.

5.2.2 Тестирование Ministral 3 8B Instruct

Модель Ministral 3 8B Instruct показала хорошее качество генерации технических вопросов. Вопросы, представленные на рисунке 12, в

большинстве случаев были корректными и соответствовали заданной профессии. Однако была замечена особенность: модель иногда немного завышала уровень сложности вопросов. Например, для более низкого уровня кандидата могли появляться вопросы, которые ближе к middle- или senior-уровню.

```
{
  "session_id": "1c20ab41a99f",
  "profession": "Java-разработчик",
  "level": "middle",
  "model_name": "mistral-3:8b",
  "questions": [
    {
      "number": 1,
      "text": "Расскажите о механизмах работы сборщика мусора JVM и основных алгоритмах, таких как G1, ZGC и Shenandoah. В каких сценариях вы бы рекомендовали использовать каждый из них?"
    },
    {
      "number": 2,
      "text": "Опишите, как работают параллельные и конкурентные коллекции в Java, например, ConcurrentHashMap и CopyOnWriteArrayList. Приведите пример использования каждой из них в реальном приложении с объяснением плюсов и минусов."
    },
    {
      "number": 3,
      "text": "Что такое flyweight и facade паттерны? Приведите пример их реализации на Java и объясните, в каких случаях они полезны?"
    },
    {
      "number": 4,
      "text": "Разъясните, как работают аннотации @Transactional и @Cacheable в Spring. Какие параметры вы можете указать в @Transactional, и какие проблемы могут возникнуть при неправильном их использовании?"
    },
    {
      "number": 5,
      "text": "Объясните, как происходит работа JPA/Hibernate при запросе к базе данных. Что происходит с EntityManager и Session в lifecycle объекта сущности? В чем разница между @Entity и @MappedSuperclass?"
    }
  ]
}
```

Рисунок 12 – Генерация вопросов моделью Mistral 8B

На первом этапе тестирования возникла проблема с возвратом данных в формате JSON. Модель вернула некорректно структурированный ответ, из-за чего приложение не смогло обработать результат (рисунок 13). Это является критичной проблемой для разрабатываемой системы, так как backend-сервис ожидает строго заданный формат данных.

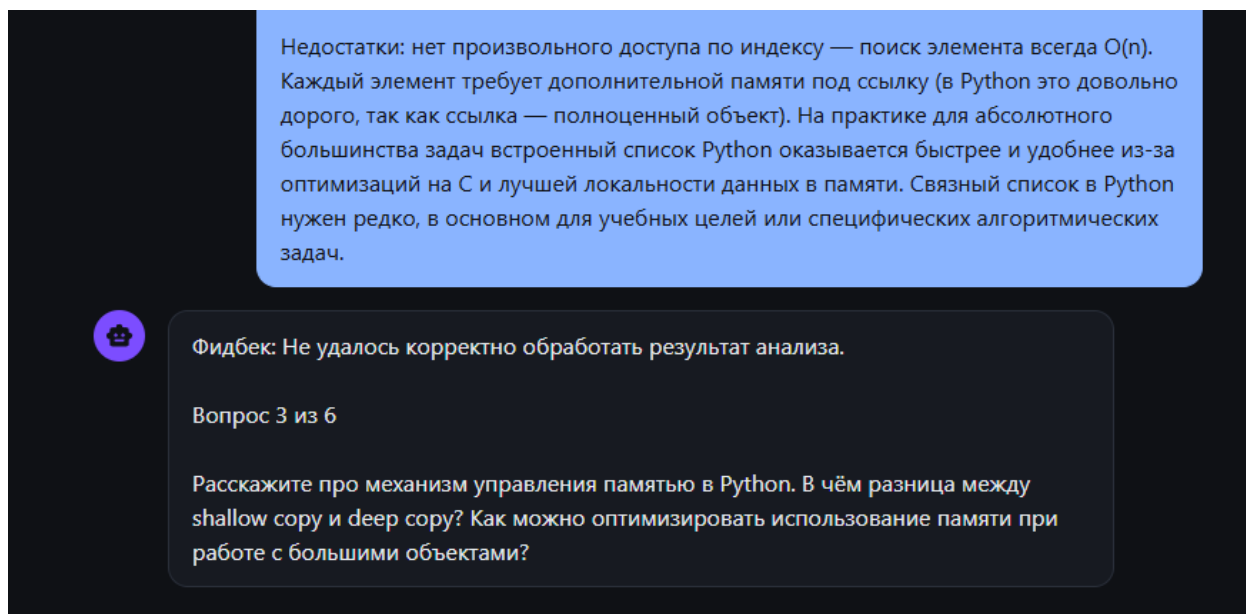


Рисунок 13 – Проблема с возвратом данных модели Mistral 3 8B

После корректировки инструкций модель стала работать значительно лучше. Она начала корректно возвращать JSON, анализировать ответы пользователя, формировать полный фидбек и задавать наводящие вопросы в нужных ситуациях. Также модель корректно реагировала на случаи, когда кандидат не знает ответ: она не за цикливалась на одном вопросе, а переходила к следующему этапу интервью.

Модель корректно анализирует ответы пользователя, а также отлично оценивает полностью интервью (рисунок 14).

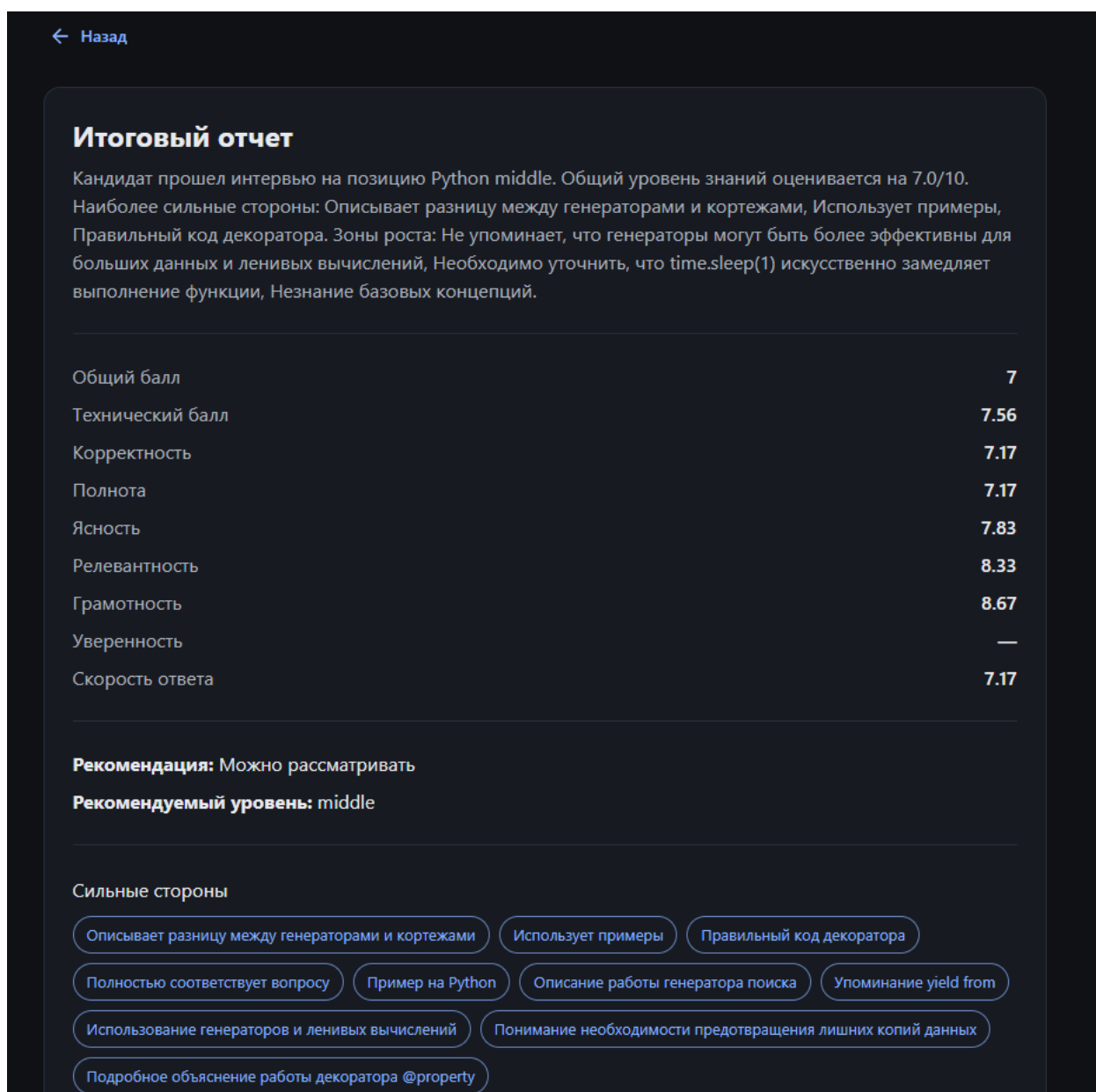


Рисунок 14 – Оценка интервью моделью Mistral 3 8B

В целом Mistral 3 8B Instruct можно считать отличным кандидатом для использования в приложении. Она хорошо справляется с генерацией вопросов, анализом ответов и итоговой оценкой кандидата.

5.2.3 Тестирование Mistral-7B-Instruct-v0.3-Q8_0

Модель Mistral-7B-Instruct-v0.3-Q8_0 показала лучший результат среди всех протестированных моделей. На этапе генерации вопросов модель работала стабильно, хотя были замечены отдельные недостатки: некоторые вопросы могли пересекаться между собой, а часть вопросов не всегда точно соответствовала заявленному уровню кандидата, представлено это на рисунке

15. Несмотря на это, общее качество вопросов было достаточным для проведения технического интервью.

Главным преимуществом данной модели стала стабильность работы. В отличие от ряда других моделей, Mistral-7B-Instruct-v0.3-Q8_0 прошла тестирование без необходимости существенной переработки инструкций. Модель корректно возвращала данные в формате JSON, что позволило без ошибок обрабатывать ответы на стороне приложения.

```
{
  "session_id": "2f4d96dd0a08",
  "profession": "Java Middle",
  "level": "middle",
  "model_name": "mistral:7b-instruct-v0.3-q8_0",
  "questions": [
    {
      "number": 1,
      "text": "Как реализована система памяти Heap в Java и какие основные структуры данных там присутствуют?"
    },
    {
      "number": 2,
      "text": "Что такое Java Memory Model и почему она необходима для многопоточного программирования?"
    },
    {
      "number": 3,
      "text": "Как работает JVM и какие основные классы ей предоставляются для работы с памятью?"
    },
    {
      "number": 4,
      "text": "Чем отличается интерфейс (interface) от абстрактного класса в Java?"
    },
    {
      "number": 5,
      "text": "Как реализована сборка мусора в Java и какие стратегии она использует для оптимизации производительности?"
    }
  ]
}
```

Рисунок 15 – Генерация вопросов моделью Mistral 7B

Особенно хорошо модель проявила себя при анализе ответов кандидата. Она корректно понимала смысл ответа, в том числе если пользователь отвечал с использованием программного кода. Модель давала точный и полезный фидбек, правильно определяла сильные и слабые стороны ответа, а также могла задать уточняющий вопрос в подходящий момент (рисунок 16).

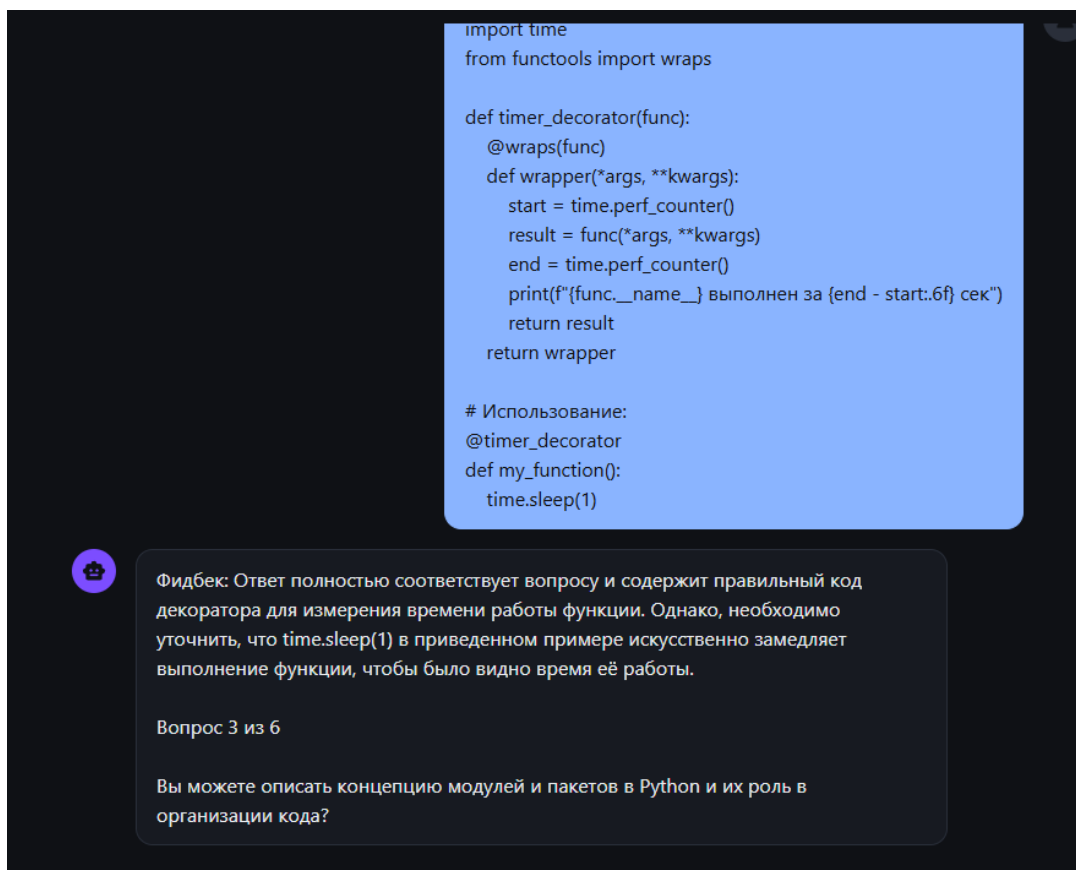


Рисунок 16 – Оценка ответа кандидата моделью Mistral 7B

Если кандидат сообщал, что не знает ответ на вопрос, модель не пыталась принудительно вести его к ответу и не уходила в заикание. В таких случаях она корректно переходила к следующему вопросу. Также модель правильно понимала намерение пользователя завершить интервью и могла остановить сессию в любой момент по запросу кандидата.

Итоговый отчет формировался качественно: модель точно оценивала знания кандидата, корректно выделяла недостатки и давала обоснованную обратную связь. По совокупности факторов – стабильность JSON-вывода, качество анализа, отсутствие заикания и корректная работа с пользовательскими намерениями – данная модель заняла первое место и может рассматриваться как основной кандидат для использования в приложении.

5.2.4 Тестирование IBM Granite 3.3 8B Instruct

Модель Granite 3.3 8B Instruct на начальном этапе потребовала корректировки инструкций, не сразу понимала, какое количество вопросов

необходимо генерировать, это представлено на рисунке 17. После внесения изменений модель смогла корректно сгенерировать необходимое количество вопросов (рисунок 18). Однако качество вопросов на первом этапе было невысоким: они относились к нужной теме, но не всегда соответствовали требуемому уровню собеседования.

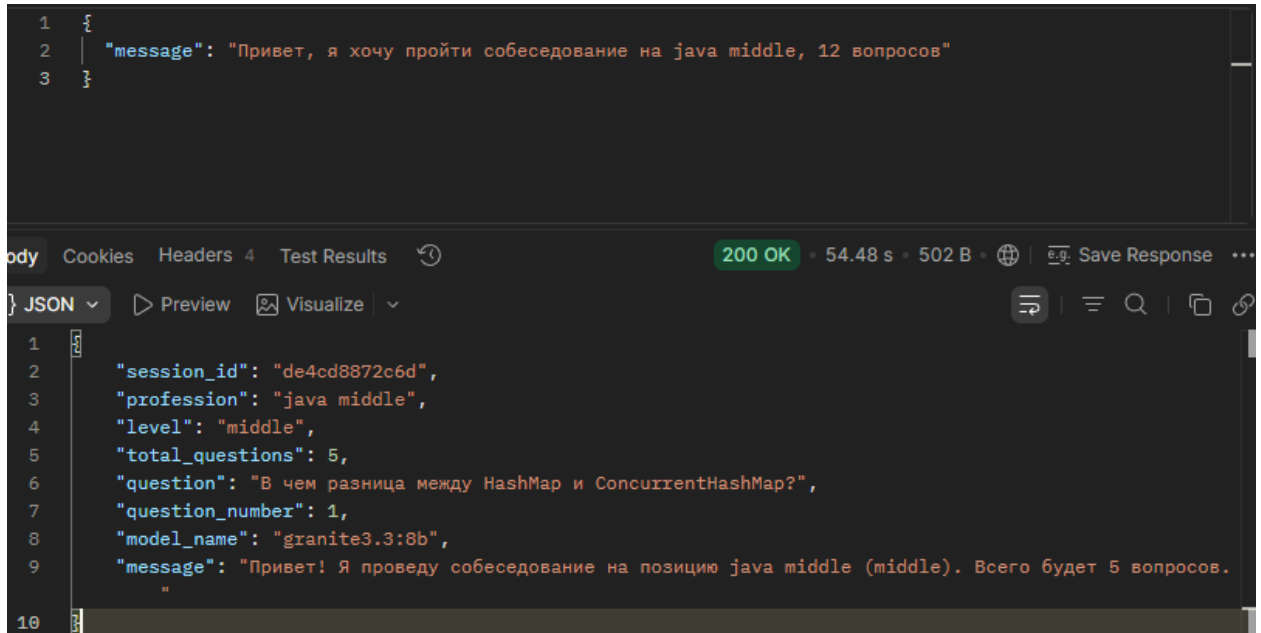


Рисунок 17 – Некорректная генерация вопросов моделью Granite

После корректировки инструкций модель создала правильное количество вопросов.

```

1  {
2    "session_id": "c7696ddb8abf",
3    "profession": "java middle",
4    "level": "middle",
5    "model_name": "granite3.3:8b",
6    "questions": [
7      {
8        "number": 1,
9        "text": "Какие подходы вы знаете для оптимизации производительности кода на Java?"
10     },
11     {
12       "number": 2,
13       "text": "Чем отличаются Interface и Abstract Class в Java?"
14     },
15     {
16       "number": 3,
17       "text": "Как работает сборка в Maven и как можно настроить зависимости?"
18     },
19     {
20       "number": 4,
21       "text": "Как вы реализуете логику обработки исключений в Java?"
22     },
23     {
24       "number": 5,
25       "text": "Чем отличаются HashSet и TreeSet в Java?"
26     },
27     {
28       "number": 6,
29       "text": "Как работает Spring Framework в Java и как его можно интегрировать в проект?"

```

Рисунок 18 – Генерация вопросов моделью Granite

При анализе ответов модель показала сильные стороны. Она хорошо задавала уточняющие вопросы, если ответ кандидата был неполным, и корректно воспринимала ответы, содержащие программный код. Если кандидат не мог ответить на вопрос, модель не продолжала бесконечно требовать ответ, что является важным преимуществом для сценария интервью.

Однако была выявлена проблема заикливания. В некоторых случаях модель просила более развернутый ответ даже тогда, когда пользователь уже дал достаточно полный ответ. Это могло приводить к повторному обсуждению одного и того же вопроса. После дополнительной корректировки инструкций данная проблема была устранена.

После исправлений Granite 3.3 8B Instruct стала работать заметно лучше. Модель корректно анализировала ответы, задавала уточняющие вопросы в подходящих ситуациях, понимала намерение пользователя завершить интервью и формировала качественную обратную связь. Итоговый отчет

также был достаточно точным, а общая оценка кандидата соответствовала качеству его ответов (рисунок 19).

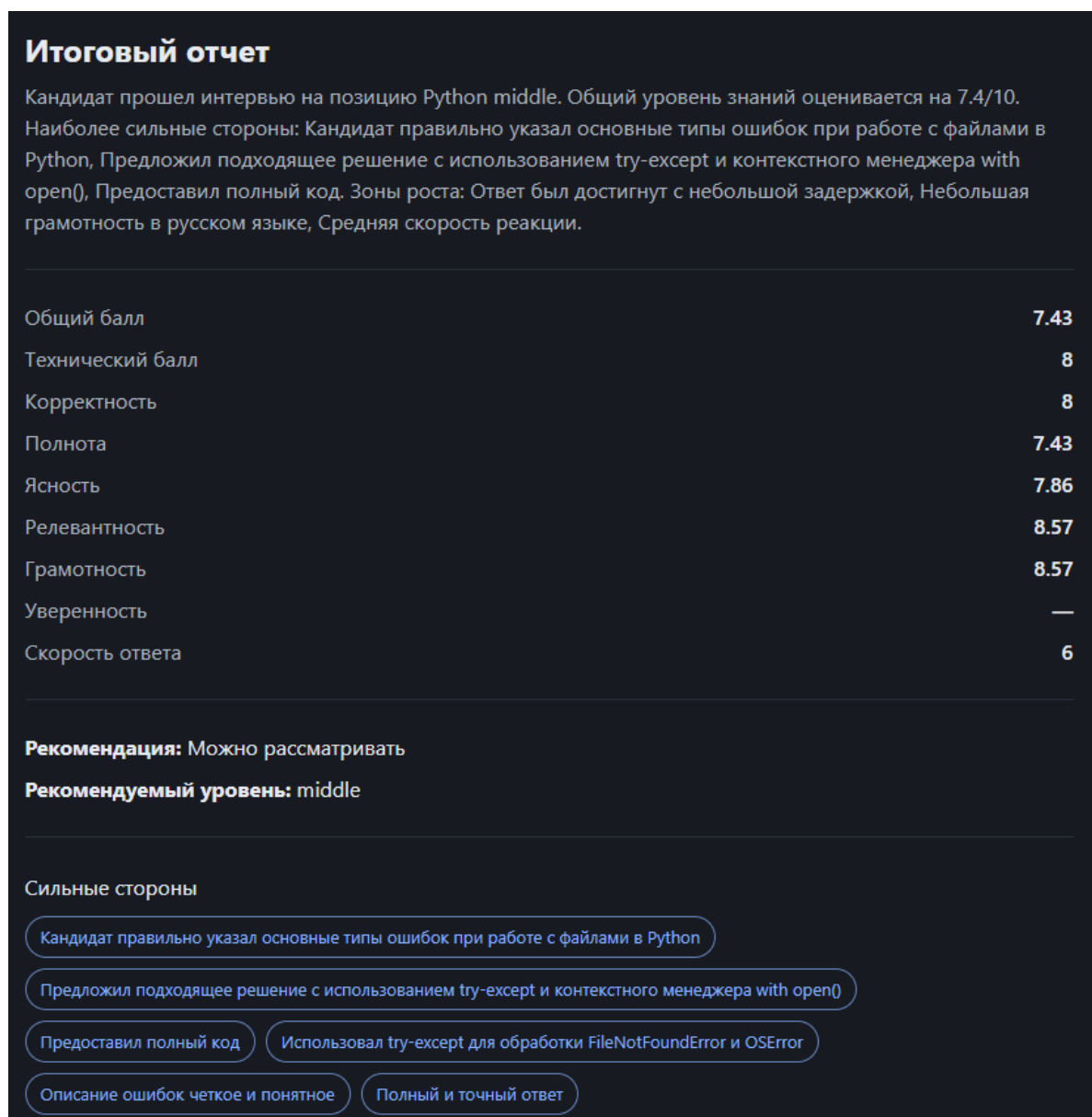


Рисунок 19 – Формирование отчета интервью моделью Granite

Таким образом, Granite 3.3 8B Instruct можно считать пригодной для использования в приложении.

5.2.5 Тестирование DeepSeek-R1-Distill-Qwen-7B

Модель DeepSeek-R1-Distill-Qwen-7B показала неоднозначные результаты. С одной стороны, она ориентирована на рассуждения и способна давать неплохую обратную связь по ответам пользователя. С другой стороны, в рамках задачи проведения русскоязычного технического интервью были выявлены существенные проблемы.

Главная проблема заключалась в генерации вопросов. Несмотря на явное требование использовать только русский язык, модель иногда формировала вопросы с китайскими символами или отдельными фрагментами на английском языке (рисунок 20). Это делает часть вопросов непригодной для использования, так как кандидат может не понять содержание вопроса.

```
{
  "session_id": "14179ca72cbe",
  "profession": "developer",
  "level": "middle",
  "model_name": "deepseek-r1:7b",
  "questions": [
    {
      "number": 1,
      "text": "Что такое Inverted Index в контексте поисковых引擎ов и как он используется?"
    },
    {
      "number": 2,
      "text": "Как эффективно обрабатывать миллионы записей с помощью индексной структуры данных?"
    },
    {
      "number": 3,
      "text": "Как убедиться, что элементы списка уникальны без использования дополнительной памяти?"
    },
    {
      "number": 4,
      "text": "Объясните, как реализовать обмен двумя числами без использования временного переменного. "
    }
  ]
}
```

Рисунок 20 – Генерация вопросов моделью DeepSeek

Также качество вопросов часто не соответствовало поставленному уровню. Некоторые вопросы были плохо сформулированы, не относились к теме или не подходили под заявленную профессию. Для приложения ассистента собеседований это является критичным недостатком, так как качество интервью напрямую зависит от качества сгенерированных вопросов.

На первом этапе также наблюдалось заикливание. Даже если кандидат давал полный ответ, модель могла продолжать задавать уточняющие вопросы. Если пользователь сообщал, что не знает ответ, модель иногда продолжала настаивать на продолжении обсуждения темы или формировала вопрос на другом языке. После изменения инструкций проблема заикливания была частично устранена, но проблема с языком и качеством вопросов осталась (рисунок 21).

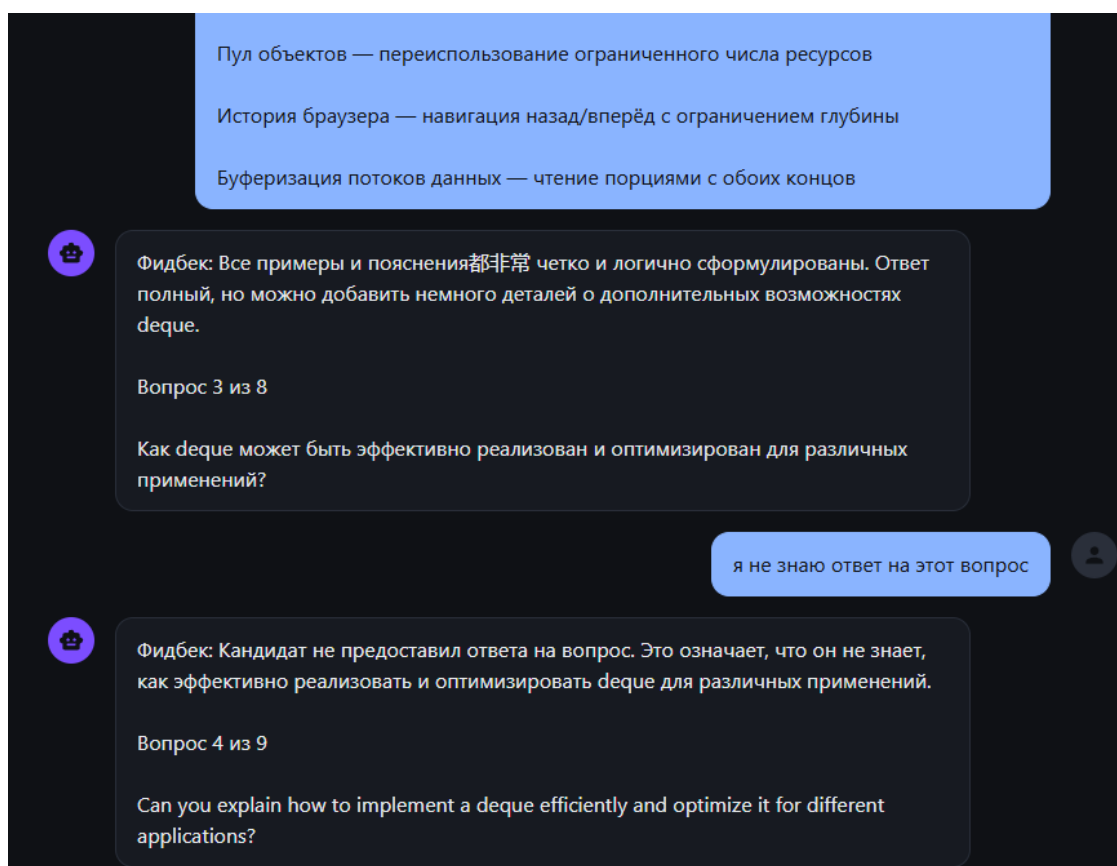


Рисунок 21 – Анализ ответов пользователя моделью DeepSeek

Обратную связь по уже заданным вопросам модель давала достаточно неплохо. Однако из-за низкого качества генерации вопросов итоговая ценность интервью снижалась. Даже если модель хорошо оценивает ответ, это не компенсирует ситуацию, когда сам вопрос сформулирован некорректно или не соответствует теме.

По результатам тестирования DeepSeek-R1-Distill-Qwen-7B не подходит для использования в приложении в качестве основной модели. Ее можно рассматривать только как дополнительный инструмент для важных задач, но не как полноценную модель для генерации и проведения русскоязычного технического интервью.

5.2.6 Тестирование Phi-4-mini

Модель Phi-4-mini показала слабые результаты для задачи проведения интервью. На этапе генерации вопросов модель иногда формировала вопросы, соответствующие требуемому уровню, однако периодически встречались

вопросы не по теме (рисунок 22). Это снижает надежность модели при использовании в автоматизированном сценарии.

```
{
  "session_id": "cd2229553749",
  "profession": "Java Middle",
  "level": "middle",
  "model_name": "phi4-mini:latest",
  "questions": [
    {
      "number": 1,
      "text": "Объясните различия между shallow copy и deep copy в Java."
    },
    {
      "number": 2,
      "text": "Что такое Garbage Collection и какие существуют алгоритмы GC в Java?"
    },
    {
      "number": 3,
      "text": "Опишите механизм синхронизации блоков Lock в Java."
    },
    {
      "number": 4,
      "text": "Как реализовать паттерн Singleton в Java без использования класса static?"
    },
    {
      "number": 5,
      "text": "Раскройте, как работает ThreadLocal в Java и для чего он используется."
    },
    {
      "number": 6,
      "text": "Что такое ConcurrentHashMap и какие преимущества она предоставляет по сравнению с стандартной HashMap?"
    },
    {
      "number": 7,
      "text": "Опишите алгоритм спекулятивного выполнения в JavaScript."
    }
  ]
}
```

Рисунок 22 – Генерация вопросов моделью Phi-4

Основная проблема была связана с управлением диалогом. Даже при полном ответе кандидата модель могла заикливаться и продолжать задавать уточняющие вопросы по уже раскрытой теме. Если пользователь сообщал, что не знает ответ, модель могла ошибочно интерпретировать это как желание завершить интервью. Аналогичная проблема возникала при некорректных или неполных ответах: модель иногда воспринимала их не как ответ на вопрос, а как намерение остановить сессию (рисунок 23).

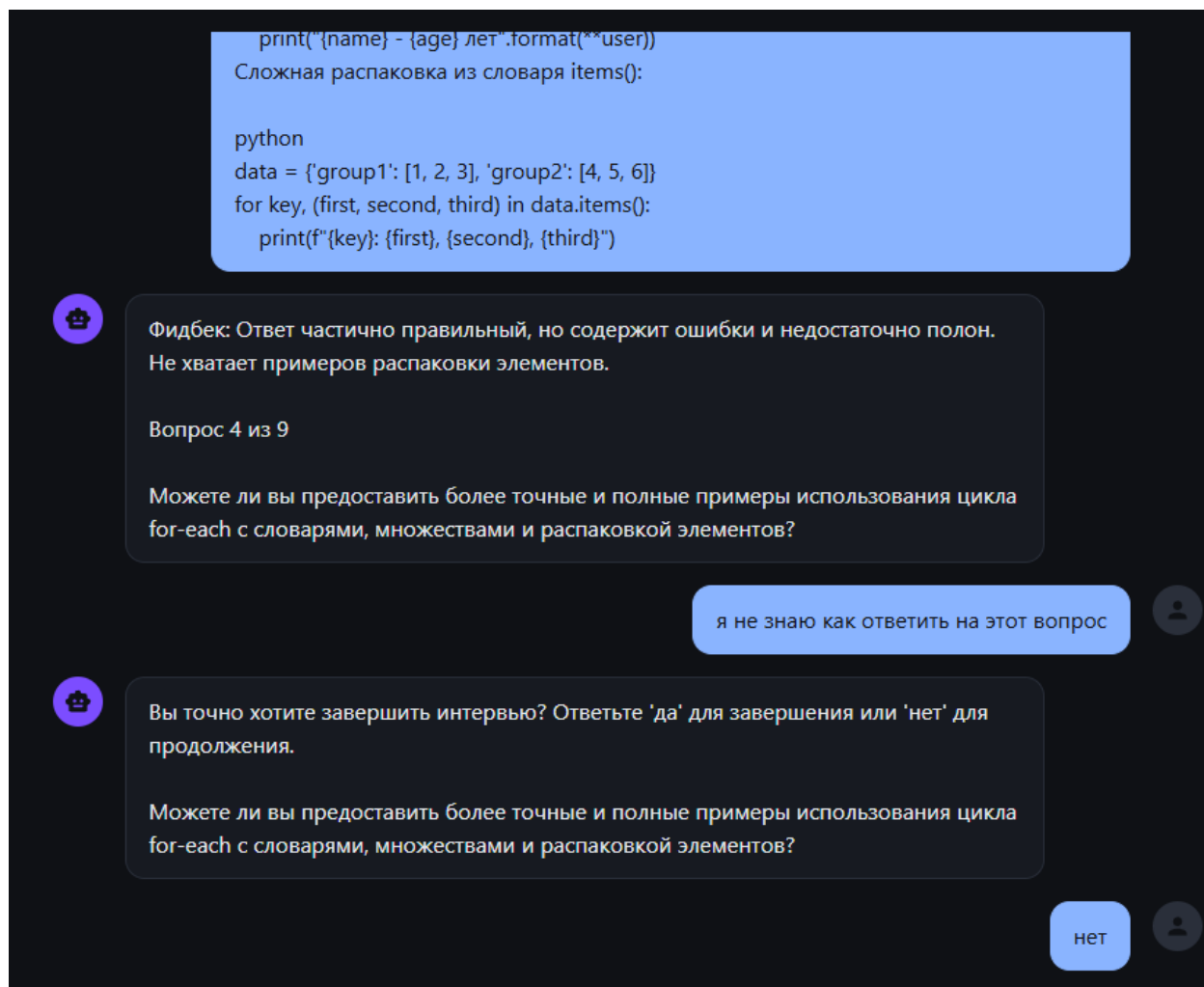


Рисунок 23 – Анализ ответов пользователя моделью Phi-4

После исправления инструкций поведение модели стало немного лучше, но ключевые проблемы полностью устранить не удалось. Модель по-прежнему могла неверно определять намерение пользователя и неправильно реагировать на незнание ответа.

Качество анализа ответов также оказалось недостаточным. Модель давала мало развернутого анализа, иногда некорректно понимала смысл ответа и могла ошибаться при выставлении оценки. Итоговый отчет формировался слабо: встречались случаи как завышения, так и занижения оценки знаний кандидата.

По результатам тестирования Phi-4-mini не подходит для использования в качестве основной модели приложения. Несмотря на небольшой размер и удобство локального запуска, качество анализа и стабильность диалогового поведения оказались недостаточными для поставленной задачи.

5.2.7 Тестирование Llama 3.1 8B Instruct

Модель Llama 3.1 8B Instruct показала хорошие результаты при генерации вопросов (рисунок 24). В большинстве случаев вопросы были релевантными выбранной профессии и подходили для технического собеседования. Иногда встречались вопросы немного выше или ниже заданного уровня, но общая структура интервью сохранялась корректной.

```
{
  "session_id": "75c5c09cd172",
  "profession": "java middle",
  "level": "middle",
  "model_name": "llama3.1:8b-instruct-q8_0",
  "questions": [
    {
      "number": 1,
      "text": "Расскажите о принципах работы Java Virtual Machine (JVM)"
    },
    {
      "number": 2,
      "text": "Как реализуется синхронизация потоков в Java"
    },
    {
      "number": 3,
      "text": "Назовите основные коллекции Java и их применение"
    },
    {
      "number": 4,
      "text": "Опишите работу с базами данных посредством JPA/Hibernate"
    },
    {
      "number": 5,
      "text": "Чем отличается Transactional Manager от Service Layer в Spring Boot"
    },
    {
      "number": 6,
      "text": "Расскажите о различиях между RESTful API и SOAP-услуги"
    },
    {
      "number": 7,
      "text": "Назовите основные методы управления транзакциями в JDBC"
    }
  ]
}
```

Рисунок 24 – Генерация вопросов моделью Llama

Основная проблема модели была связана с возвратом данных в формате JSON. Иногда модель формировала ответ в некорректной структуре, из-за чего приложение не могло сразу обработать результат. Эта проблема встречалась не постоянно, но сохранялась даже после нескольких изменений инструкций. Поэтому полностью стабильного поведения добиться не удалось.

При этом в части анализа ответов пользователя модель показала себя хорошо. Она корректно понимала ответы, могла выявлять ошибки и давать полезный фидбек. Также модель правильно реагировала на ситуации, когда пользователь не знает ответ: она не заикливалась, а переходила к следующему вопросу (рисунок 25). Намерение завершить интервью модель также распознавала корректно.

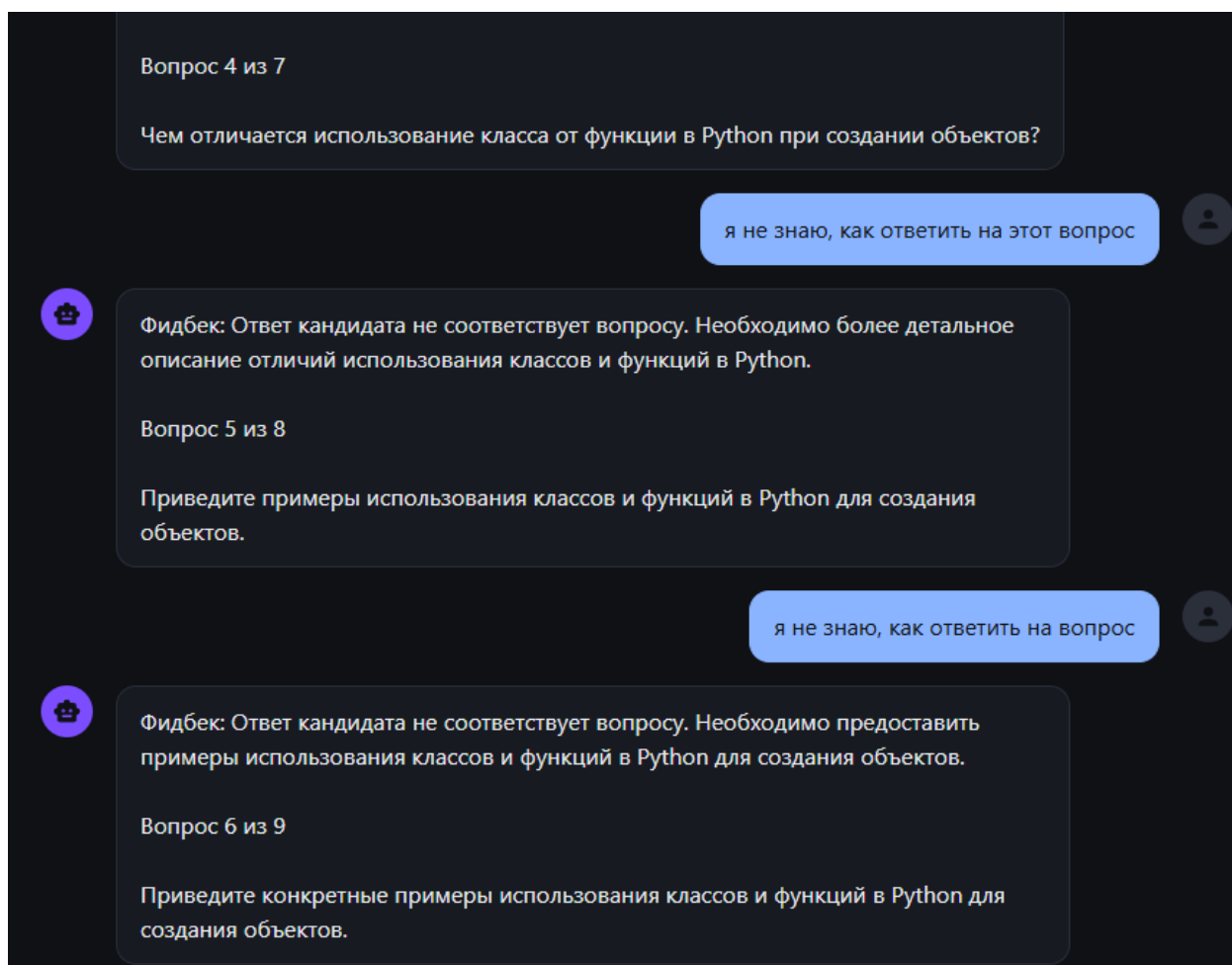


Рисунок 25 – Анализ ответов пользователя моделью Llama

После многочисленных переработок инструкций всё равно не получилось добиться необходимого результата и модель иногда возвращает ответ в неправильном JSON формате. Но это бывает редко, поэтому получалось полностью проходить собеседования и в целом модель показала себя хорошо в плане анализа ответов пользователя.

Итоговый отчет формировался достаточно качественно (рисунок 26). Модель правильно оценивала знания кандидата и давала логичные выводы по

результатам интервью. Однако из-за нестабильности JSON-вывода уступила другим моделям, которые работали более надежно в программной интеграции.

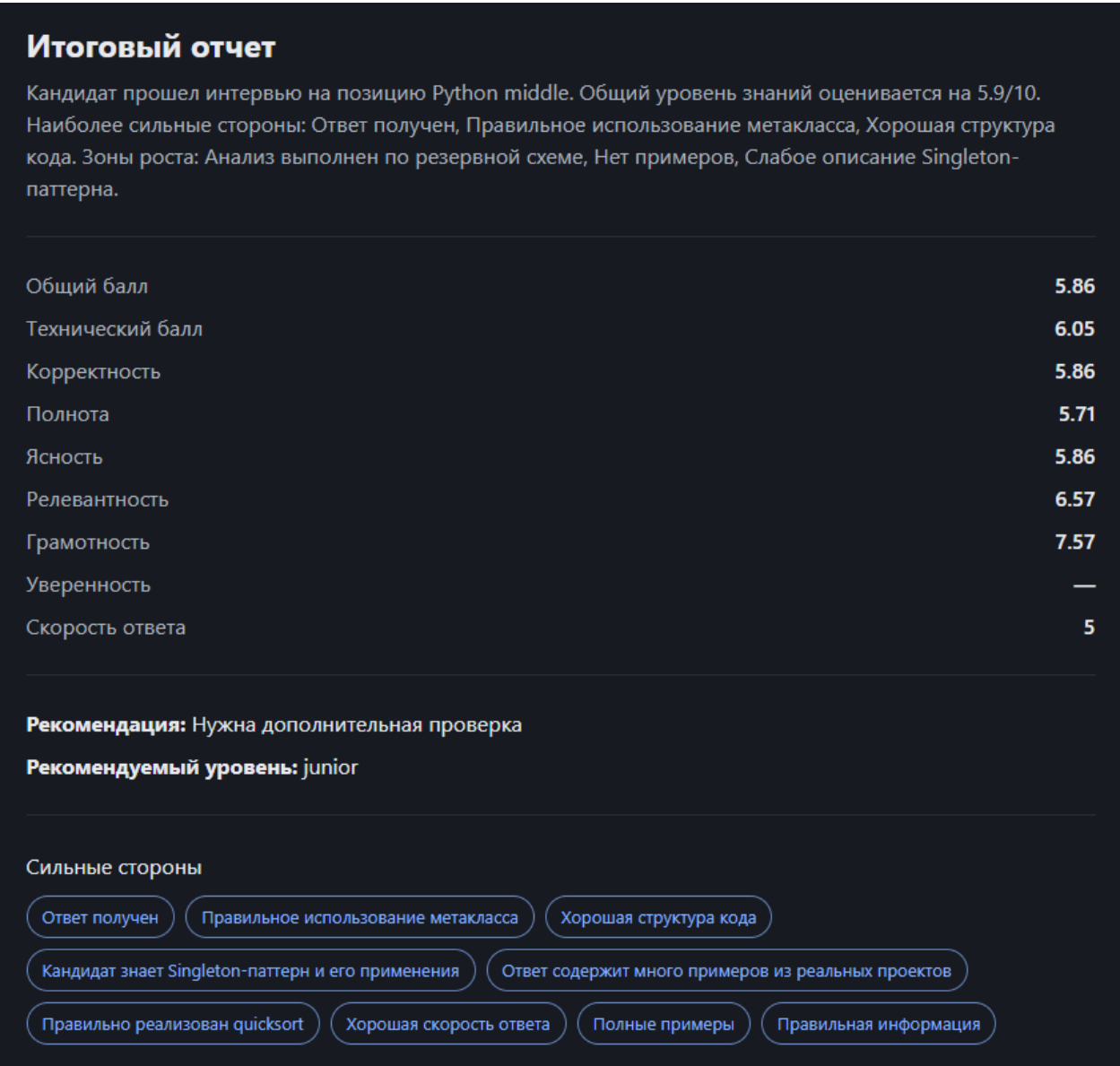


Рисунок 26 – Анализ интервью моделью Llama

Таким образом, модель можно считать пригодной для использования, но с ограничениями. Для практического применения потребуется дополнительная обработка некорректного JSON-ответа или повторный запрос к модели при ошибке формата.

5.2.8 Тестирование Gemma 3 4B

Модель Gemma 3 4B показала очень хороший результат на этапе генерации вопросов (рисунок 27). По данному критерию она оказалась одной из лучших: вопросы были точными, соответствовали выбранной профессии,

уровню кандидата и теме собеседования. Это является сильной стороной модели.

```
{
  "session_id": "1306da5ac0bc",
  "profession": "Java",
  "level": "middle",
  "model_name": "gemma3:4b",
  "questions": [
    {
      "number": 1,
      "text": "Объясните разницу между интерфейсом и абстрактным классом в Java. Приведите примеры, когда вы бы использовали каждый из них."
    },
    {
      "number": 2,
      "text": "Что такое мутабельность и неизменяемость в Java? Приведите примеры объектов, которые мутабельны, и неизменяемы."
    },
    {
      "number": 3,
      "text": "Какие существуют способы реализации Singleton в Java? Опишите преимущества и недостатки каждого способа."
    },
    {
      "number": 4,
      "text": "Что такое Java Reflection и в каких случаях ее можно использовать? Какие риски связаны с ее использованием?"
    },
    {
      "number": 5,
      "text": "Объясните концепцию Dependency Injection (DI) и какие существуют способы ее реализации в Java (например, constructor injection, setter injection, field injection)."
    },
    {
      "number": 6,
      "text": "Что такое Garbage Collection в Java? Как работает процесс сборки мусора? Какие факторы влияют на частоту сборки мусора?"
    }
  ]
}
```

Рисунок 27 – Генерация вопросов моделью Gemma

Однако в процессе самого интервью были выявлены серьезные проблемы. Главная из них – заикливание. Даже если кандидат давал полный ответ, модель продолжала задавать уточняющие вопросы по той же теме. Если пользователь сообщал, что не знает ответ, модель также не переходила дальше, а переформулировала тот же вопрос и продолжала возвращаться к нему.

После изменения инструкций проблема не была устранена. Модель продолжала вести себя некорректно в диалоговом сценарии, что делает ее неудобной для автоматического проведения интервью. Для приложения важно не только сгенерировать качественный список вопросов, но и корректно

управлять ходом диалога, переходить к следующему вопросу и завершать обсуждение темы при необходимости.

Таким образом, несмотря на высокое качество генерации вопросов, Gemma 3 4B не прошла тестирование как полноценная модель для приложения. Ее можно рассматривать как вспомогательную модель для генерации вопросов, но не как основную модель для проведения интервью и анализа ответов.

5.2.9 Тестирование Mistral-Nemo-Instruct-2407-Q4_K_M

Модель Mistral-Nemo-Instruct-2407-Q4_K_M показала один из лучших результатов среди всех протестированных моделей. На этапе генерации вопросов она формировала вопросы, максимально приближенные к реальным вопросам технического собеседования. Вопросы не по теме выявлено не было, а уровень сложности соответствовал заданным параметрам (рисунок 28).

```
{
  "session_id": "0d32e92b6aca",
  "profession": "java",
  "level": "middle",
  "model_name": "mistral-nemo:12b-instruct-2407-q4_K_M",
  "questions": [
    {
      "number": 1,
      "text": "Опишите различия между synchronized и volatile в Java."
    },
    {
      "number": 2,
      "text": "Какие типы коллекций в Java наиболее подходят для хранения больших количеств данных?"
    },
    {
      "number": 3,
      "text": "Как работает механизм кэширования в JVM? Как его можно настроить?"
    },
    {
      "number": 4,
      "text": "Объясните разницу между equals() и hashCode() в Java."
    },
    {
      "number": 5,
      "text": "Каким образом Spring Boot автоматически конфигурирует бины из компонентов, отмеченных аннотацией @Component?"
    },
    {
      "number": 6,
      "text": "Как работать с транзакциями в Spring Framework? Какие аннотации используются для этого?"
    },
    {
      "number": 7,
      "text": "Чем отличается JPA от Hibernate? Как они работают вместе в Spring Boot?"
    }
  ]
}
```

Рисунок 28 – Генерация вопросов моделью Mistral-Nemo

На начальном этапе была обнаружена проблема с заикливанием уточняющих вопросов. Однако после корректировки инструкций модель стала работать корректно. Она начала правильно определять, когда ответ кандидата требует уточнения, а когда необходимо перейти к следующему вопросу.

Сильной стороной модели является качество анализа ответов. Mistral-Nemo-Instruct-2407-Q4_K_M давала подробную и содержательную обратную связь, корректно выделяла ошибки и сильные стороны ответа. Также модель понимала намерение пользователя завершить интервью и могла корректно остановить сессию.

Итоговый отчет формировался достаточно качественно. Модель неплохо оценивала навыки кандидата по разным параметрам и давала логичный общий вывод (рисунок 29). По качеству генерации вопросов и анализа ответов данная модель оказалась одной из сильнейших. Однако из-за большего размера она является более ресурсоемкой по сравнению с 7B–8B моделями.

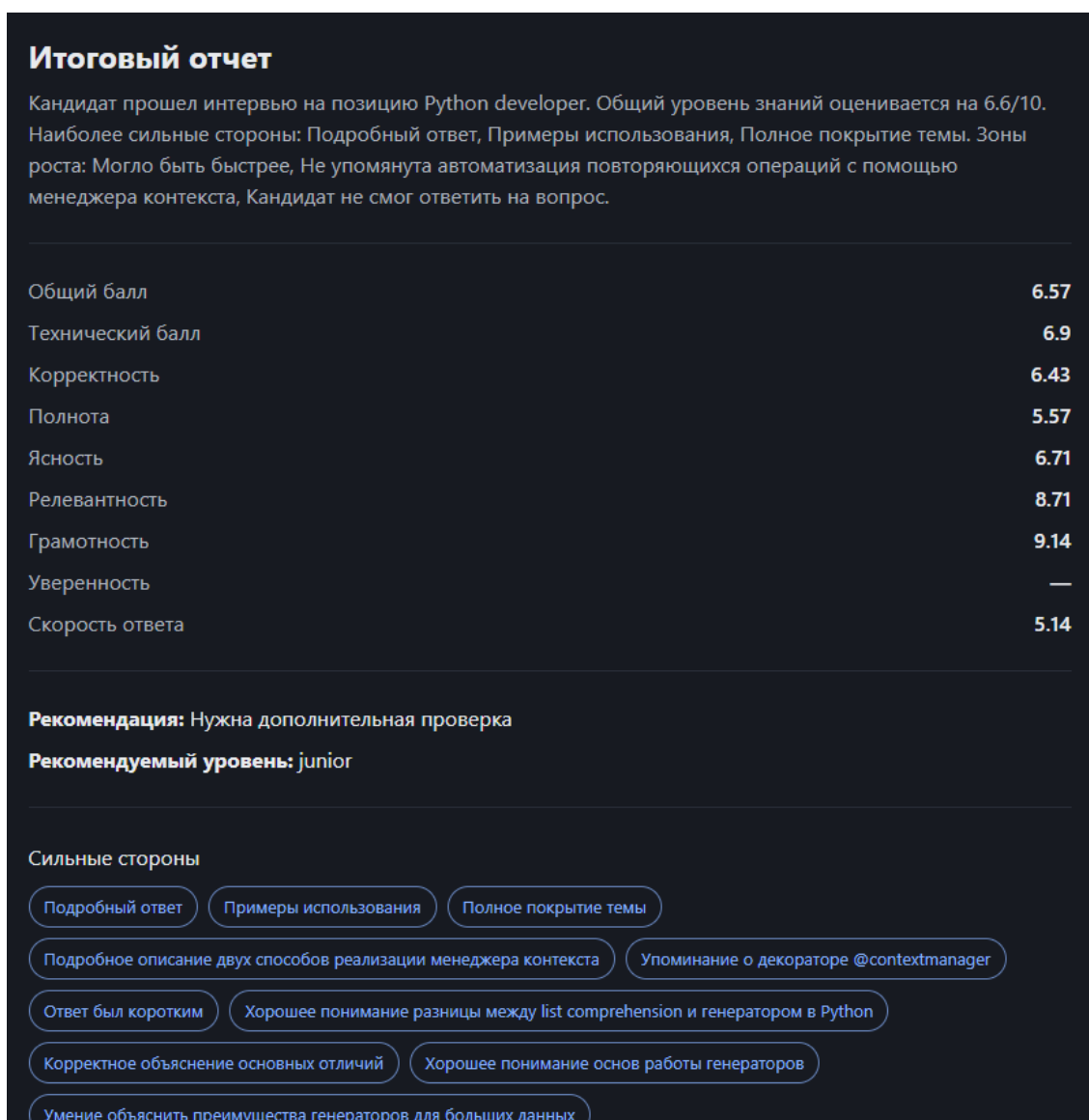


Рисунок 29 – Оценка интервью моделью Mistral-Nemo

По результатам тестирования Mistral-Nemo-Instruct-2407-Q4_K_M может использоваться в приложении, особенно если приоритетом является качество вопросов и анализа, а не минимальная нагрузка на оборудование.

5.2.10 Вывод

По итогам тестирования все модели показали разный уровень пригодности для задачи проведения технического интервью. Основными критериями выбора стали не только качество генерации текста, но и стабильность работы в приложении: корректный JSON-вывод, отсутствие заикливания, понимание намерений пользователя, способность оценивать ответы и формировать итоговый отчет.

Наилучший результат показала модель Mistral 3 8B Instruct. Несмотря на отдельные недостатки при генерации вопросов, она оказалась наиболее стабильной в практическом использовании. Модель корректно возвращала JSON, хорошо анализировала ответы, понимала программный код, правильно реагировала на незнание ответа и позволяла завершить интервью в любой момент. Важным преимуществом стало то, что она прошла тестирование без необходимости существенной переработки инструкций.

Также стоит отметить Mistral-Nemo-Instruct-2407-Q4_K_M. Она показала очень высокое качество генерации вопросов и анализа ответов. Ее вопросы были наиболее похожи на реальные вопросы технического собеседования, а итоговый отчет был достаточно точным. Главным ограничением является более высокая требовательность к вычислительным ресурсам, так как модель относится к 12B-классу.

Модели Qwen3-8B, Ministral 7B-Instruct, Granite 3.3 8B Instruct и Llama 3.1 8B Instruct Q8_0 также можно считать потенциально пригодными для использования, но с дополнительными условиями. Granite 3.3 8B после корректировки инструкций работает хорошо, но изначально имела проблемы с качеством вопросов и заиканием. Llama 3.1 8B хорошо анализирует ответы, но периодически нарушает JSON-формат.

Модели DeepSeek-R1-Distill-Qwen-7B, Phi-4-mini и Gemma 3 4B не подходят в качестве основной модели приложения. DeepSeek имеет проблемы с языком и качеством генерации вопросов, Phi-4-mini некорректно определяет намерения пользователя и слабо формирует итоговый отчет, а Gemma 3 4B, несмотря на хорошую генерацию вопросов, не справляется с управлением диалогом и уходит в заикание.

5.3 Тестирование нейросетевых моделей для транскрипции голоса

Для проверки правильности работы нейросетевых моделей автоматического распознавания речи использовались одинаковые тестовые фразы на русском и английском языках. Такой подход позволяет сравнить

модели в одинаковых условиях и определить, насколько точно каждая из них выполняет преобразование голосового ответа кандидата в текстовый формат.

В рамках тестирования оценивались следующие параметры: точность распознавания русской речи, точность распознавания английских технических терминов, устойчивость к произношению, количество ошибок в итоговой транскрибации, скорость работы и сложность интеграции модели в приложение. Данный этап является важным, так как качество транскрибации напрямую влияет на дальнейшую оценку ответа кандидата большой языковой моделью. Если речь будет распознана некорректно, LLM может неправильно оценить знания кандидата, даже если сам ответ был верным.

5.3.1 Тестирование Whisper

Модель Whisper показала наилучший результат среди протестированных решений для транскрибации. Она хорошо справлялась с распознаванием русского языка, корректно обрабатывала английские технические термины и в большинстве случаев формировала текст, близкий к исходному произнесенному ответу.

В процессе тестирования Whisper показал высокую устойчивость к смешанным фразам, где русский текст сочетался с английскими словами, например названиями технологий, языков программирования или техническими терминами. Это особенно важно для приложения ассистента собеседований, так как в технических интервью кандидаты часто используют смешанную речь. На рисунке 30 показан результат диктовки текста для данной модели.

Жил расшифровывается как Global Interpreter Log или глобальная блокировка интерпретатора. Это мьютекс, который используется в сипетон, основной реализации петона для защиты доступа к объектам интерпретатора. Проще говоря, Jill гарантирует, что в любой момент времени только один поток может выполнять байт-код-петон. Даже если у вас многоядерный процессор и запущенный несколько потоков, одновременно может работать только один из них в рамках одного процессора. Причина существования Jill историческая и практическая. Многие внутренние структуры сипетон, включая счетчики ссылок для управления памяти, не являются потоками безопасными. Добавление тонкой блокировки для каждого отдельного объекта потребовало бы огромных усилий по переписыванию интерпретатора и привело бы к значительному падению производительности в напоточных программах. Jill это компромисс, он делает напоточный код быстрым и упрощает написание си расширений, но ограничивает эффективность многопоточности. Как Jill работает на практике? Каждый поток в петон перед выполнением байт-кода должен схватить Jill. Интерпретатор переключает Jill между потоками через определенную промежуток времени. Обычно каждый из 100 тиков байт-кода или по системному таймеру, что позволяет разным потокам получать скванты времени. Однако одновременно выполняется только один поток. Я не Jill на многопоточность, сильно зависит от типа задач. Без задач, связанных с водом-выводом, таких как чтение файлов, сетевые запросы, работа с базами данных, Jill практически не является проблемой. Когда поток выполняет операцию вод-вывода, он может явно отпустить Jill, потому что эта операция не требует выполнения кода петона. Персонность системы может приостановить поток и Jill освобождается до других потоков. Благодаря этому потоке вод-вывода могут работать правильно и эффективно.

Рисунок 30 – Транскрибация голоса ASR-моделью Whisper

Основным преимуществом Whisper является высокая точность распознавания и хорошее понимание как русского, так и английского языка. По результатам тестирования данная модель является наиболее предпочтительным вариантом для дальнейшего использования в приложении.

5.3.2 Тестирование Vosk

Модель Vosk также в ходе практического тестирования качество распознавания оказалось недостаточным.

Vosk часто допускал ошибки при транскрибации, особенно в случаях, когда в речи присутствовали технические термины, английские слова или сложные формулировки. Из-за этого итоговый текст мог заметно отличаться от исходного ответа кандидата. Для приложения, где дальнейшая оценка

выполняется на основе распознанного текста, такие ошибки являются критичными. На рисунке 31 показан пример транскрипции голоса этой моделью

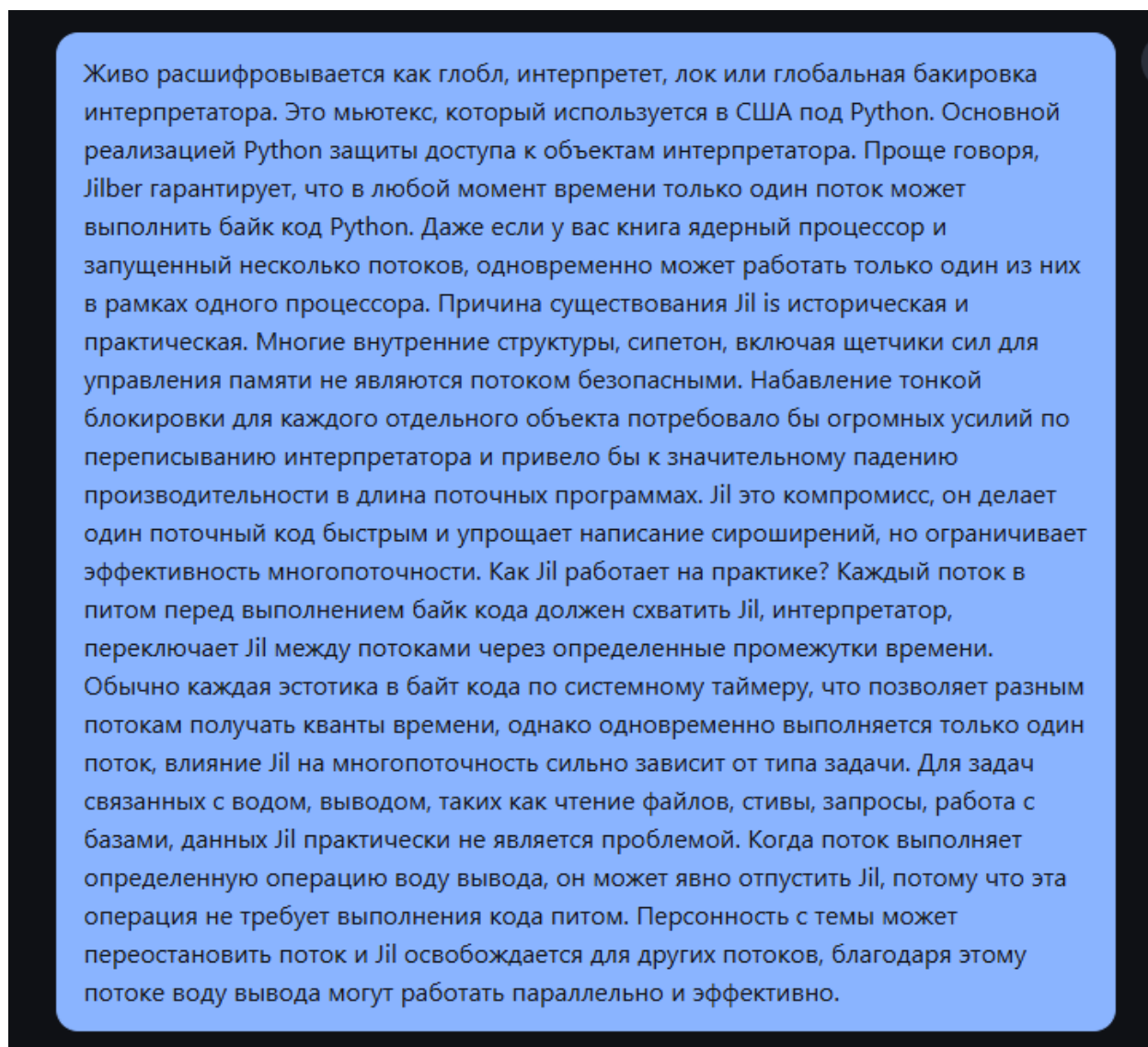


Рисунок 31 – Транскрипция голоса ASR-моделью Vosk

Таким образом, Vosk показал себя плохо, так как имеет много искажений при транскрипции, поэтому не подходит данная модель в качестве инструмента для перевода голоса в текст.

5.3.3 Тестирование NVIDIA NeMo / Parakeet

Решения NVIDIA NeMo / Parakeet показали средние результаты при транскрипции речи. Модель недостаточно точно распознавала текст, не корректно обрабатывала русскую речь и в целом плохо передавала смысл произнесенного ответа. В результате транскрипции появляется много

бессмысленных фраз, из-за этого большая языковая модель может плохо понять, что сказал пользователь.

Стоит отметить, что интеграция данного решения оказалась сложнее по сравнению с другими нейросетевыми моделями. Для работы требуется установка PyTorch и дополнительных библиотек, а также более внимательная настройка окружения. На рисунке 32 приведен пример перевода из голоса в текст для данной модели.

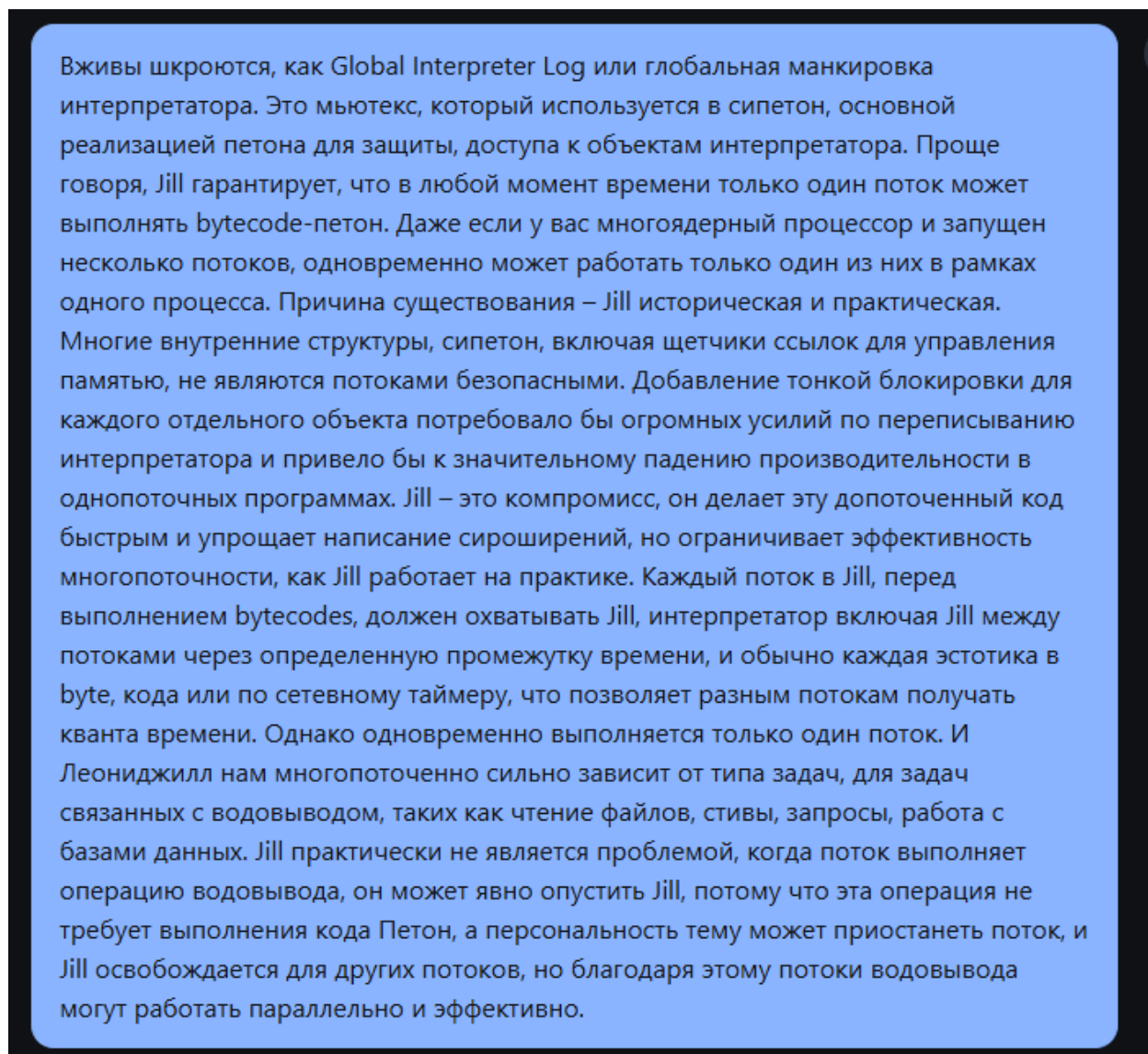


Рисунок 32 – Транскрибация голоса ASR-моделью Nvidia

В целом NVIDIA NeMo / Parakeet показал себя как недостаточно точный инструмент для рассматриваемой задачи. Его можно использовать только в качестве легкой альтернативы, если требования к качеству распознавания невысокие.

5.3.4 Тестирование Coqui STT

Модель Coqui STT в ходе проверки показала приемлемое качество транскрибации и в целом справлялась с распознаванием как русского, так и английского текста. В целом тоже присутствуют ошибки при переводе из голосового формата в текстовый, но большие языковые модели нормально воспринимали такой текст и справлялись с анализом.

На рисунке 33 представлен пример транскрибации голоса данной нейросетевой модели.

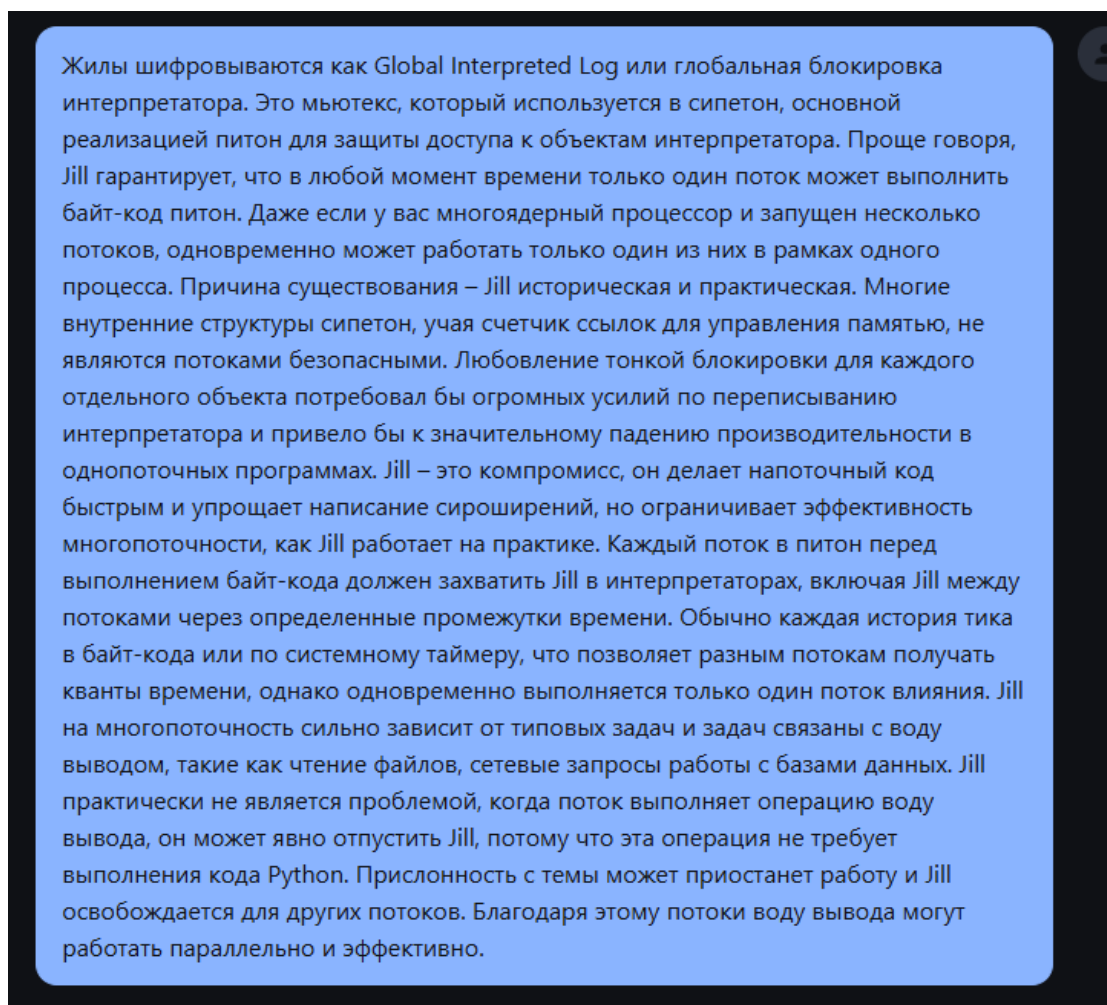


Рисунок 33 – Транскрибация голоса ASR-моделью Coqui

Coqui STT можно рассматривать как рабочий вариант, однако по качеству и стабильности распознавания он уступает Whisper.

Таким образом, для дальнейшей реализации приложения наиболее рационально использовать Whisper в качестве основной модели распознавания речи, а Coqui STT рассматривать как возможную альтернативу для

дальнейших экспериментов. Остальные нейросетевые модели не следует рассматривать в качестве основного инструмента для транскрибации голоса в текст.

5.4 Итоговый результат

В ходе проведенного исследования была выявлена наиболее подходящая большая языковая модель, а именно Mistral 3 8B, которая была выбрана для использования в качестве основной модели в приложении. ASR-модель была выбрана Whisper от компании OpenAI на этапе тестирования с остальными конкурентами.

Итогом магистерской работы «Интеллектуальный ассистент для проведения собеседования» стало приложение, в которой интегрирована большая языковая модель и нейросетевая модель для транскрибации голоса в текст.

На рисунке 34 показана главная страница приложения, на которой происходит собеседование с ИИ на выбранную пользователем тему. Искусственный интеллект сгенерировал вопросы, по которым будет проходить интервью, а пользователь отвечает на эти вопросы, получая необходимую информацию о ходе прохождения. На данном рисунке видно, что у пользователя есть возможность ввода письменного ответа, голосового ответа. Также данный пользователь является HR-агентом, поэтому у него также есть возможность создания собственных интервью и просматривания уже имеющихся.

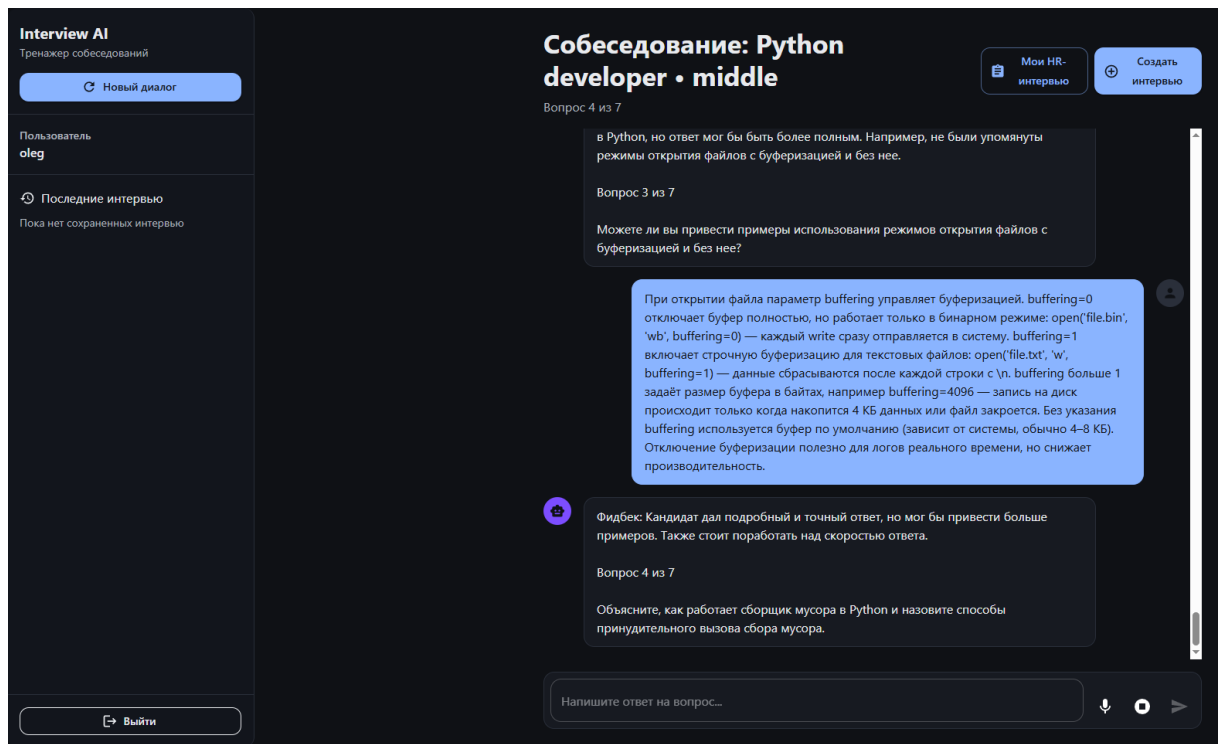


Рисунок 34 – Процесс прохождения интервью

Меню создания интервью представлено на рисунке 35. У HR-агента есть возможность создания собственного интервью, в котором он может указать название, добавить описание и профессию, а также назначить уровень интервью. Присутствуют необязательные настройки требований, предъявляемых к интервью, а именно: обязательный ответ голосом, включение камеры и сохранение записи. И основная часть с созданием собственных вопросов, HR-специалист вводит в нужное поле эти вопросы, которые в последствие большая языковая модель будет задавать кандидату.

Рисунок 35 – Создание интервью

Также у HR-специалиста есть возможность просматривать все созданные ранее интервью, копировать ссылку на созданное интервью, чтобы отправить ее кандидату, и просмотр результатов пройденного интервью. Данный функционал представлен на рисунке 36.

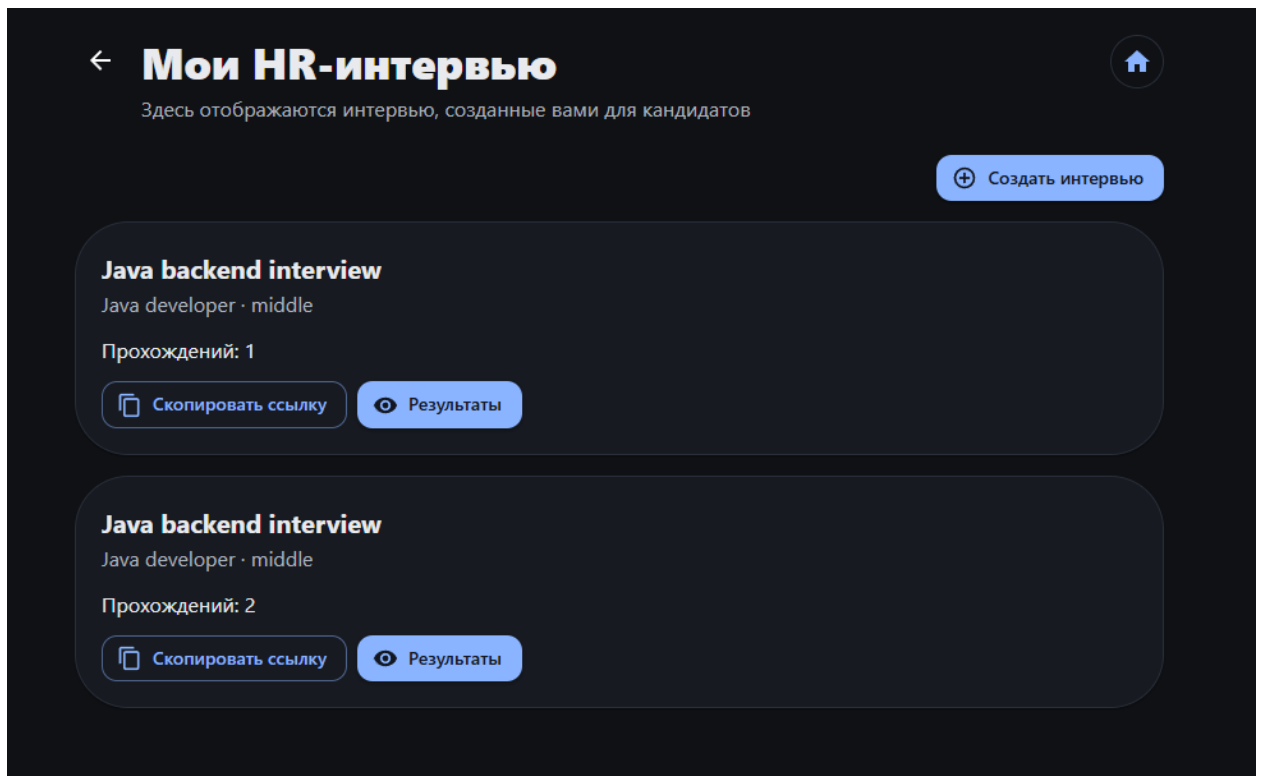


Рисунок 36 – Просмотр созданных интервью

Результат пройденного интервью другими кандидатами представлен на рисунке 37. Здесь указан краткий отчет о пройденном интервью пользователем, балл, на который пользователь прошел интервью, а также рекомендация от большой языковой модели на то, стоит ли брать на данную позицию кандидата. Полный отчет показан на рисунке 38, на котором представлены все ответы кандидата на заданные вопросы, а также обратная связь от модели и выставленные баллы на каждый вопрос.

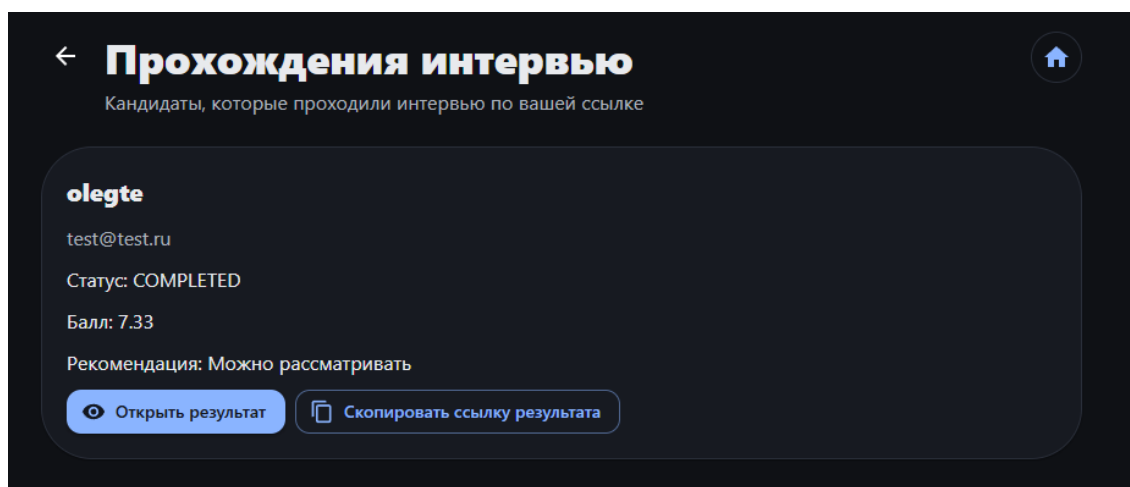


Рисунок 37 – Пройденные пользователями созданного интервью

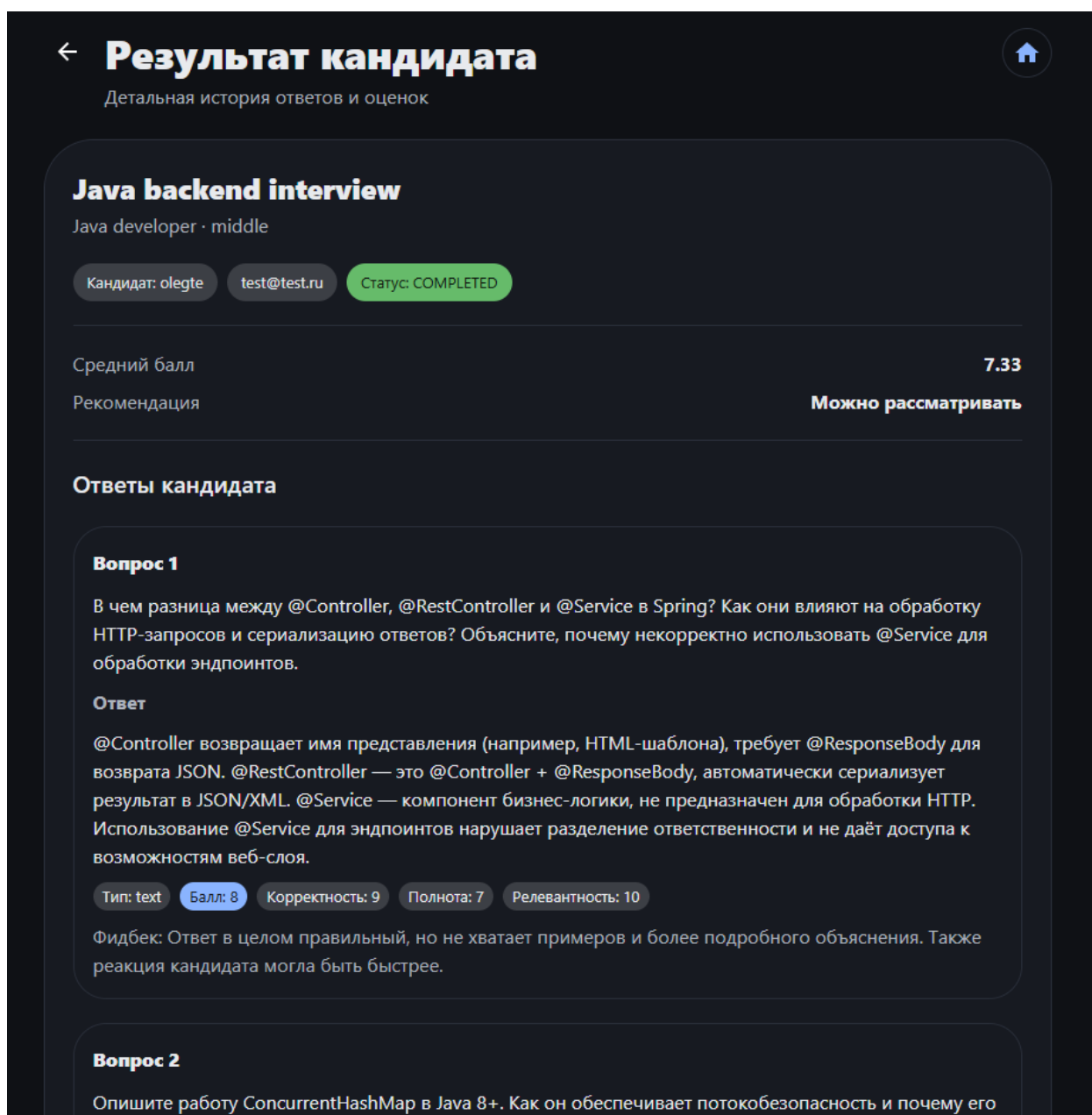


Рисунок 38 – Результат прохождения интервью пользователем

Прохождение интервью, которое было создано HR-агентом, представлено на рисунке 39. Здесь пользователю указываются требования к интервью, а также вопросы, которые были созданы специалистом по рекрутингу.

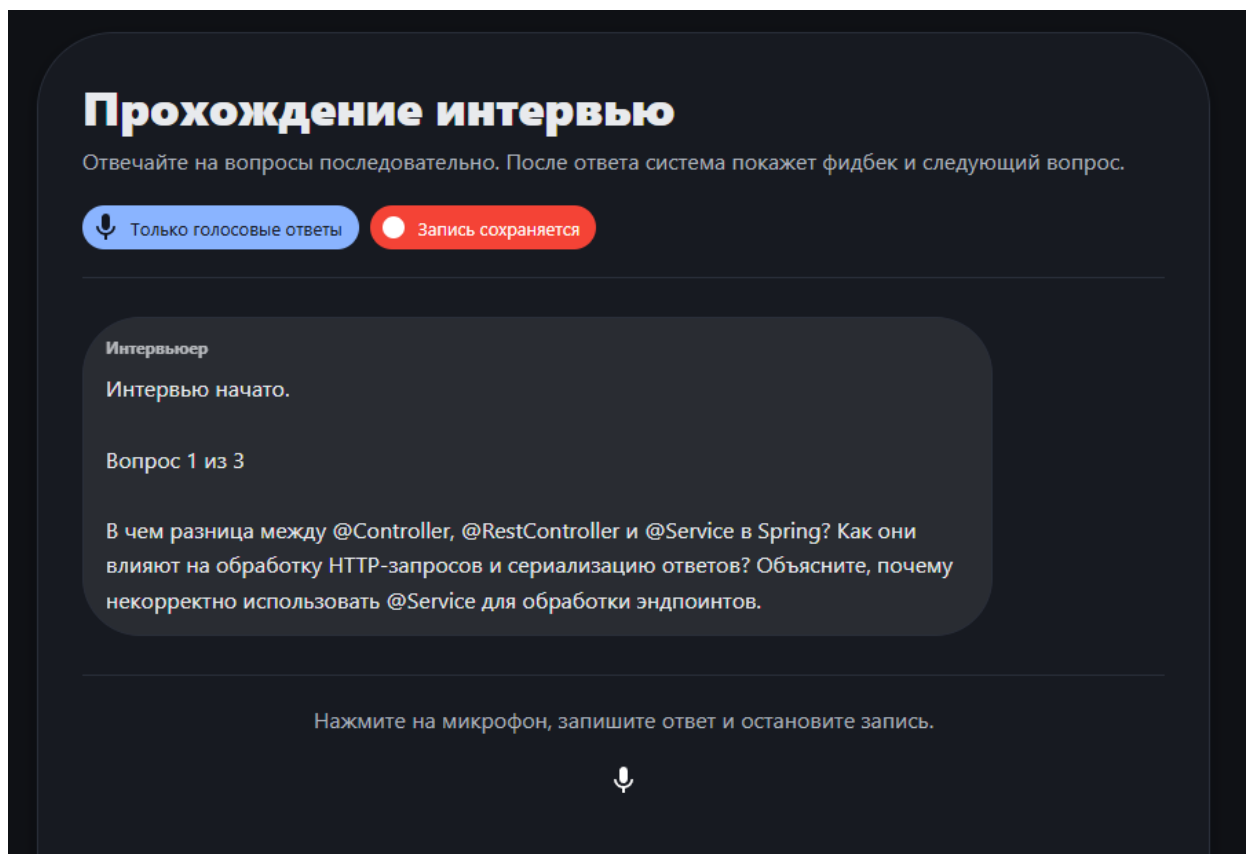


Рисунок 39 – Процесс прохождения созданного интервью

В итоге было разработано приложение с применением искусственного интеллекта, которое выполняет все поставленные требования и является конкурентно способным с другими аналогичными приложениями.

ЗАКЛЮЧЕНИЕ

В результате выполнения магистерской работы были проанализированы большие языковые модели, которые были использованы в качестве AI-ассистента для проведения собеседования, задача которых состояла анализировать навыки кандидата, генерировать вопросы по конкретной специальности и оценивать знания кандидата. Были взяты уже обученные модели от разных компаний для выявления наиболее подходящего продукта под поставленные цели. В итоге были выявлены модели, которые хорошо справились с задачей проведения интервью. И было выявлено, что на данном этапе развития больших языковых моделей, существует очень много различных решений, являющимися бесплатными с возможностью коммерческого использования, которые можно интегрировать для реализации задач в собственных программных продуктах.

Также были проанализированы нейросетевые модели с возможностью транскрипции голоса в текст. Если есть потребность использования данной функции в приложении, то в ходе исследования было выявлено, что на рынке представлено много хороших, качественных, а главное бесплатных продуктов, которые могут решить эту потребность.

Итогом работы стала программа для проведения собеседования с искусственным интеллектом, которая является хорошим вариантом для тренировки перед реальным собеседованием, а также инструментом для сотрудников по найму персонала для разгрузки их обязанностей.

Работа может быть полезна людям, которые хотят проверить или натренировать свои навыки перед собеседованием, а также для компаний, которые хотят уменьшить задачи специалистов в сфере рекрутинга.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. HH Статистика: сервис открытой аналитики рынка труда // hh.ru [САЙТ]. – URL: https://stats.hh.ru/?countrySalaryDynamicChartProfArea=information_technology&hhIndexProfArea=information_technology (дата обращения: 12.03.2026).
2. Qwen3-VL-8B // Baidu Baika [САЙТ]. – URL: <https://baike.baidu.com/ru/item/Qwen3-VL-8B/785925> (дата обращения: 25.04.2025).
3. Qwen3-8B // Hugging Face [САЙТ]. – URL: <https://huggingface.co/Qwen/Qwen3-8B> (дата обращения: 24.04.2025).
4. Ministral 3 8B: Specifications and GPU VRAM Requirements // APXML [САЙТ]. – URL: <https://apxml.com/models/ministral-3-8b> (дата обращения: 24.04.2025).
5. Mistral 7B // Mistral AI [САЙТ]. – URL: <https://mistral.ai/news/announcing-mistral-7b> (дата обращения: 25.04.2025).
6. Mistral-7B-Instruct-v0.3 // Hugging Face [САЙТ]. – URL: <https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3> (дата обращения: 24.04.2025).
7. Granite-3.3-8B-Instruct Model by IBM // NVIDIA NIM [САЙТ]. – URL: https://build.nvidia.com/ibm/granite-3_3-8b-instruct/modelcard (дата обращения: 24.04.2025).
8. DeepSeek-R1-Distill-Qwen-7B // Hugging Face [САЙТ]. – URL: <https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B> (дата обращения: 24.04.2025).
9. Phi-4-mini-instruct // Microsoft Foundry Models [САЙТ]. – URL: <https://ai.azure.com/catalog/models/Phi-4-mini-instruct> (дата обращения: 24.04.2025).
10. Llama-3.1-8B-Instruct Model by Meta // NVIDIA NIM [САЙТ]. – URL: https://build.nvidia.com/meta/llama-3_1-8b-instruct/modelcard (дата обращения: 24.04.2025).

11. Gemma-3-4B-it // Hugging Face [САЙТ]. – URL: <https://huggingface.co/google/gemma-3-4b-it> (дата обращения: 25.05.2026).
12. Mistral NeMo // Mistral AI [САЙТ]. – URL: <https://mistral.ai/news/mistral-nemo> (дата обращения: 24.04.2025).
13. Best Qwen Models Ranked: Which to Run Locally // InsiderLLM [САЙТ]. – URL: <https://insiderllm.com/guides/qwen-models-guide/> (дата обращения: 24.04.2025).
14. Ministral 3 // arXiv [САЙТ]. – URL: <https://arxiv.org/html/2601.08584v1> (дата обращения: 24.04.2025).
15. Mistral-7B-Instruct-v0.3: Benchmarks, Specs, Parameters & Download // DataLearnerAI [САЙТ]. – URL: https://www.datalearner.com/en/ai-models/pretrained-models/Mistral-7B-Instruct-v0_3#opensource-experience (дата обращения: 24.04.2025).
16. Granite 3.3 8B Base vs Granite 3.3 8B Instruct Comparison // LLM Stats [САЙТ]. – URL: <https://llm-stats.com/models/compare/granite-3.3-8b-base-vs-granite-3.3-8b-instruct> (дата обращения: 24.04.2025).
17. Mistral-Nemo-Instruct-2407 // Hugging Face [САЙТ]. – URL: <https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407> (дата обращения: 24.04.2025).
18. Attention Is All You Need // arXiv. – 2017. – URL: <https://arxiv.org/pdf/1706.03762> (дата обращения: 24.04.2025).
19. Ньюмен Сэм. Создание микросервисов. – Санкт-Петербург: Питер, 2023. – 23 с.

ПРИЛОЖЕНИЕ А

(обязательное)

Интеллектуальный ассистент для проведения собеседования

Листинг программы

<https://github.com/DuhnovOleg?tab=repositories>.

Файл assessmentApi.ts

```
import { http } from '../../shared/api/http'
import type {
  AssessmentAttemptDetails,
  AssessmentAttemptListItem,
  AssessmentListItem,
  CreateAssessmentRequest,
  CreateAssessmentResponse,
  PublicAssessment,
  StartPublicAssessmentRequest,
  StartPublicAssessmentResponse,
  SubmitAssessmentAnswerRequest,
  SubmitAssessmentAnswerResponse,
  SubmitAssessmentVoiceAnswerRequest,
} from '../../types/assessment.types'

export async function createAssessment(
  payload: CreateAssessmentRequest,
): Promise<CreateAssessmentResponse> {
  const { data } = await
  http.post<CreateAssessmentResponse>('/api/v1/assessments',
  payload)
  return data
}

export async function getMyAssessments():
Promise<AssessmentListItem[]> {
  const { data } = await
  http.get<AssessmentListItem[]>('/api/v1/assessments/my')
  return data
}

export async function getAssessmentAttempts(
  assessmentId: string,
): Promise<AssessmentAttemptListItem[]> {
  const { data } = await
  http.get<AssessmentAttemptListItem[]>(
    `/api/v1/assessments/${assessmentId}/attempts`,
  )
  return data
}

export async function getAssessmentAttemptDetails(
  attemptId: string,
): Promise<AssessmentAttemptDetails> {
  const { data } = await http.get<AssessmentAttemptDetails>(
    `/api/v1/assessments/attempts/${attemptId}`,
  )
}
```

```

        return data
    }

export async function getPublicAssessment(publicToken: string):
Promise<PublicAssessment> {
    const { data } = await http.get<PublicAssessment>(
        `/api/v1/public/assessments/${publicToken}`,
    )
    return data
}

export async function startPublicAssessment(
    publicToken: string,
    payload: StartPublicAssessmentRequest,
): Promise<StartPublicAssessmentResponse> {
    const { data } = await
http.post<StartPublicAssessmentResponse>(
    `/api/v1/public/assessments/${publicToken}/start`,
    payload,
)
    return data
}

export async function submitPublicAssessmentAnswer(
    attemptId: string,
    payload: SubmitAssessmentAnswerRequest,
): Promise<SubmitAssessmentAnswerResponse> {
    const { data } = await
http.post<SubmitAssessmentAnswerResponse>(
    `/api/v1/public/assessment-
attempts/${attemptId}/answer`,
    payload,
)
    return data
}

export async function submitPublicAssessmentVoiceAnswer(
    attemptId: string,
    payload: SubmitAssessmentVoiceAnswerRequest,
): Promise<SubmitAssessmentAnswerResponse> {
    const { data } = await
http.post<SubmitAssessmentAnswerResponse>(
    `/api/v1/public/assessment-
attempts/${attemptId}/answer/voice`,
    payload,
)
    return data
}

export async function getPublicAssessmentResult(token: string):
Promise<AssessmentAttemptDetails> {
    const { data } = await http.get<AssessmentAttemptDetails>(
        `/api/v1/public/assessment-results/${token}`,
    )

```

```

    )
    return data
}

```

Файл interviews.py

```

from app.dependencies import get_interview_service,
get_session_store
from app.models.evaluation import EvaluateAnswerResponse,
EvaluateAnswerRequest
from app.schemas.interview import StartInterviewRequest,
AnswerRequest, FinalEvaluationRequest
from app.services.interview_service import InterviewService
from app.services.session_store import InMemorySessionStore
from app.core.config import settings
from fastapi import APIRouter, Depends, HTTPException

router = APIRouter(prefix="/interviews", tags=["interviews"])

@router.post("/start")
async def start_interview(
    request: StartInterviewRequest,
    service: InterviewService = Depends(get_interview_service),
):
    session = await service.start_interview(request.message)
    return {
        "session_id": session.session_id,
        "profession": session.profession,
        "level": session.level,
        "total_questions": len(session.questions),
        "question": session.questions[0],
        "question_number": 1,
        "model_name": settings.model_name,
        "message": (
            f"Привет! Я проведу собеседование на позицию "
            f"{session.profession} "
            f"({session.level}). Всего будет "
            f"{len(session.questions)} вопросов."
        ),
    }

@router.post("/answer")
async def submit_answer(
    request: AnswerRequest,
    service: InterviewService = Depends(get_interview_service),
):
    result = await service.submit_answer(
        session_id=request.session_id,
        answer=request.answer,
        input_type=request.input_type,
        confidence_score=request.confidence_score,
        response_time_seconds=request.response_time_seconds,
    )

```

```

    )
    result["model_name"] = settings.model_name
    return result

@router.post("/final_evaluation")
async def final_evaluation(
    request: FinalEvaluationRequest,
    service: InterviewService = Depends(get_interview_service),
):
    result = await
service.generate_final_evaluation(request.session_id)
    result["model_name"] = settings.model_name
    return result

@router.post("/evaluate_answer",
response_model=EvaluateAnswerResponse)
async def evaluate_answer(
    payload: EvaluateAnswerRequest,
    service: InterviewService = Depends(get_interview_service),
):
    analysis = await service.analysis_service.analyze_answer(
        profession=payload.profession,
        question=payload.question,
        answer=payload.answer,
        response_time_seconds=payload.response_time_seconds,
        confidence_score_from_audio=payload.confidence_score,
    )

    return EvaluateAnswerResponse(
        overall_score=analysis.overall_score,
        correctness_score=analysis.correctness_score,
        completeness_score=analysis.completeness_score,
        clarity_score=analysis.clarity_score,
        relevance_score=analysis.relevance_score,
        grammar_score=analysis.grammar_score,
        confidence_score=analysis.confidence_score,
        response_speed_score=analysis.response_speed_score,
        feedback=analysis.feedback,
        strengths=analysis.strengths,
        weaknesses=analysis.weaknesses,
        decision=analysis.decision,
        follow_up_question=analysis.follow_up_question,
    )

@router.get("/{session_id}/generated_questions")
async def generated_questions(
    session_id: str,
    store: InMemorySessionStore = Depends(get_session_store),
):
    session = await store.get(session_id)

```

```

        if not session:
            raise HTTPException(status_code=404, detail="Сессия не
найдена")

    return {
        "session_id": session.session_id,
        "profession": session.profession,
        "level": session.level,
        "model_name": settings.model_name,
        "questions": [
            {"number": index + 1, "text": question}
            for index, question in enumerate(session.questions)
        ],
    }

@router.get("/{session_id}/result")
async def interview_result(
    session_id: str,
    service: InterviewService = Depends(get_interview_service),
):
    result = await service.generate_final_evaluation(session_id)
    result["model_name"] = settings.model_name
    return result

@router.get("/{session_id}")
async def get_session(
    session_id: str,
    store: InMemorySessionStore = Depends(get_session_store),
):
    session = await store.get(session_id)
    if not session:
        raise HTTPException(status_code=404, detail="Сессия не
найдена")

    return {
        "session_id": session.session_id,
        "profession": session.profession,
        "level": session.level,
        "is_complete": session.is_complete,
        "awaiting_stop_confirmation":
session.awaiting_stop_confirmation,
        "current_question": (
            session.current_index + 1
            if not session.is_complete and session.current_index
< len(session.questions)
            else None
        ),
        "total_questions": len(session.questions),
        "answers_given": len(session.answers),
        "progress":
f"{len(session.answers)}/{len(session.questions)}",

```

```

        "model_name": settings.model_name,
    }

```

Файл config_transcribe.py

```

from functools import lru_cache
from typing import Any

from pydantic import field_validator
from pydantic_settings import BaseSettings

class Settings(BaseSettings):
    app_name: str = "transcribe-service"
    app_port: int = 8002
    app_env: str = "dev"

    enable_cors: bool = True
    cors_origins: list[str] = [
        "http://localhost:5173",
        "http://127.0.0.1:5173",
        "http://localhost:8080",
    ]

    max_audio_mb: float = 20

    # whisper | vosk | coqui | nemo | kaldi
    transcribe_engine: str = "whisper"

    # faster-whisper
    whisper_model: str = "small"
    whisper_device: str = "cpu"
    whisper_compute_type: str = "int8"
    whisper_beam_size: int = 5
    whisper_language: str | None = "ru"
    whisper_vad_filter: bool = True

    # Vosk
    vosk_model_path: str = "models/vosk-model-small-ru-0.22"

    # Coqui STT
    coqui_model_path: str = "models/coqui/model.tflite"
    coqui_scorer_path: str | None = None

    # NVIDIA NeMo / Parakeet
    nemo_model_name_or_path: str = "nvidia/parakeet-tdt-0.6b-v2"
    nemo_device: str = "cpu"

    # Kaldi
    kaldi_decode_script: str = "scripts/kaldi_decode.bat"

    @field_validator("cors_origins", mode="before")
    @classmethod
    def parse_cors_origins(cls, value: Any) -> list[str]:

```

```

        if isinstance(value, str):
            return [item.strip() for item in value.split(",") if
item.strip()]
        return value

```

```

class Config:
    env_file = ".env"
    env_file_encoding = "utf-8"
    extra = "ignore"

```

```

@lru_cache
def get_settings() -> Settings:
    return Settings()

```

Файл GatewaySecurityConfig.java

```

package com.example.gateway.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import
org.springframework.security.config.annotation.method.configurat
ion.EnableReactiveMethodSecurity;
import
org.springframework.security.config.annotation.web.reactive.Enab
leWebFluxSecurity;
import
org.springframework.security.config.web.server.ServerHttpSecurit
y;
import
org.springframework.security.web.server.SecurityWebFilterChain;

@Configuration
@EnableWebFluxSecurity
@EnableReactiveMethodSecurity
public class GatewaySecurityConfig {

    @Bean
    public SecurityWebFilterChain
springSecurityFilterChain(ServerHttpSecurity http) {
        return http
            .csrf(ServerHttpSecurity.CsrfSpec::disable)
            .authorizeExchange(exchange -> exchange

.pathMatchers(HttpMethod.OPTIONS, "/*").permitAll()

            .pathMatchers(HttpMethod.POST,
"/api/v1/auth/register").permitAll()
            .pathMatchers(HttpMethod.POST,
"/api/v1/auth/login").permitAll()
            .pathMatchers(HttpMethod.POST,
"/api/v1/auth/refresh").permitAll()

```

```

        .pathMatchers(HttpMethod.GET,
"/api/v1/public/assessments/**").permitAll()
        .pathMatchers(HttpMethod.POST,
"/api/v1/public/assessments/**").permitAll()
        .pathMatchers(HttpMethod.POST,
"/api/v1/public/assessment-attempts/**").permitAll()
        .pathMatchers(HttpMethod.GET,
"/api/v1/public/assessment-results/**").permitAll()

        .pathMatchers(HttpMethod.POST,
"/api/v1/interviews/start").permitAll()
        .pathMatchers(HttpMethod.POST,
"/api/v1/interviews/answer").permitAll()
        .pathMatchers(HttpMethod.POST,
"/api/v1/interviews/answer/voice").permitAll()
        .pathMatchers(HttpMethod.POST,
"/api/v1/interviews/final-evaluation").permitAll()
        .pathMatchers(HttpMethod.POST,
"/api/v1/interviews/stop").permitAll()
        .pathMatchers(HttpMethod.GET,
"/api/v1/interviews/**").permitAll()

        .pathMatchers("/actuator/health").permitAll()
        .pathMatchers("/swagger-
ui.html", "/swagger-ui/**", "/v3/api-docs/**").permitAll()

        .pathMatchers("/api/v1/reports/**").authenticated()

        .anyExchange().authenticated()
    )
    .oauth2ResourceServer(oauth2 -> oauth2.jwt(jwt -
> {
        }))
        .build();
    }
}

```

Файл InterviewController.java

```

package com.example.interviewservice.controller;

import com.example.interviewservice.dto.request.*;
import
com.example.interviewservice.dto.response.FinalEvaluationRespons
e;
import
com.example.interviewservice.dto.response.SessionStatusResponse;
import
com.example.interviewservice.dto.response.StartInterviewResponse
;

```

```

import
com.example.interviewservice.dto.response.SubmitAnswerResponse;
import com.example.interviewservice.service.InterviewService;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Map;

@RestController
@RequestMapping("/api/v1/interviews")
@RequiredArgsConstructor
public class InterviewController {

    private final InterviewService interviewService;

    @PostMapping("/start")
    public StartInterviewResponse startInterview(@Valid
@RequestBody StartInterviewRequest request) {
        return interviewService.startInterview(request);
    }

    @PostMapping("/answer")
    public SubmitAnswerResponse submitAnswer(@Valid @RequestBody
SubmitAnswerRequest request) {
        return interviewService.submitAnswer(request);
    }

    @PostMapping("/answer/voice")
    public SubmitAnswerResponse submitVoiceAnswer(@Valid
@RequestBody SubmitVoiceAnswerRequest request) {
        return interviewService.submitVoiceAnswer(request);
    }

    @PostMapping("/final-evaluation")
    public FinalEvaluationResponse finalEvaluation(@Valid
@RequestBody FinalEvaluationRequest request) {
        return interviewService.finalEvaluation(request);
    }

    @PostMapping("/stop")
    public SubmitAnswerResponse stopInterview(@Valid
@RequestBody StopInterviewRequest request) {
        return interviewService.stopInterview(request);
    }

    @GetMapping("/history")
    public List<Map<String, Object>> history() {
        return interviewService.history();
    }

    @GetMapping("/{sessionId}")
    public SessionStatusResponse getStatus(@PathVariable String

```

```

sessionId) {
    return interviewService.getStatus(sessionId);
}
}

```

Файл ReportService.java

```

package com.example.reportservice.service;

import com.example.reportservice.dto.CreateReportRequest;
import com.example.reportservice.dto.ReportResponse;
import com.example.reportservice.entity.ReportEntity;
import com.example.reportservice.repository.ReportRepository;
import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.web.server.ResponseStatusException;
import org.springframework.http.HttpStatus;

import java.security.SecureRandom;
import java.time.OffsetDateTime;
import java.util.Base64;
import java.util.List;
import java.util.Map;
import java.util.UUID;

@Service
@RequiredArgsConstructor
public class ReportService {
    private final ReportRepository repository;
    private final ObjectMapper objectMapper;
    private final SecureRandom secureRandom = new
SecureRandom();

    @Transactional
    public ReportResponse create(CreateReportRequest request) {
        ReportEntity entity = new ReportEntity();
        entity.setId(UUID.randomUUID());
        entity.setSourceType(request.getSourceType());
        entity.setSourceId(request.getSourceId());
        entity.setOwnerUserId(request.getOwnerUserId());
        entity.setCandidateName(request.getCandidateName());
        entity.setCandidateEmail(request.getCandidateEmail());
        entity.setOverallScore(request.getOverallScore());
        entity.setRecommendation(request.getRecommendation());
        entity.setSummary(request.getSummary());
        entity.setPublicKey(generateToken());
        entity.setCreatedAt(OffsetDateTime.now());
        try {
entity.setReportJson(objectMapper.writeValueAsString(request.get
ReportPayload()));
        } catch (Exception e) {

```

```

        throw new
ResponseStatusException(HttpStatus.BAD_REQUEST, "Invalid report
payload");
    }
    return toResponse(repository.save(entity));
}
public ReportResponse get(UUID id) {
    return toResponse(repository.findById(id)
        .orElseThrow(() -> new
ResponseStatusException(HttpStatus.NOT_FOUND, "Report not
found")));
}
public ReportResponse getPublic(String token) {
    return toResponse(repository.findByPublicToken(token)
        .orElseThrow(() -> new
ResponseStatusException(HttpStatus.NOT_FOUND, "Report not
found")));
}
public List<ReportResponse> byOwner(UUID ownerUserId) {
    return
repository.findByIdOwnerUserIdOrderByCreatedAtDesc(ownerUserId).st
ream().map(this::toResponse).toList();
}
private ReportResponse toResponse(ReportEntity e) {
    ReportResponse r = new ReportResponse();
    r.setId(e.getId());
    r.setSourceType(e.getSourceType());
    r.setSourceId(e.getSourceId());
    r.setOwnerUserId(e.getOwnerUserId());
    r.setCandidateName(e.getCandidateName());
    r.setCandidateEmail(e.getCandidateEmail());
    r.setPublicToken(e.getPublicToken());
    r.setOverallScore(e.getOverallScore());
    r.setRecommendation(e.getRecommendation());
    r.setSummary(e.getSummary());
    r.setCreatedAt(e.getCreatedAt());
    try {
        r.setReportPayload(objectMapper.readValue(e.getReportJson(), new
TypeReference<Map<String, Object>>() {}));
    } catch (Exception ex) {
        r.setReportPayload(Map.of());
    }
    return r;
}

private String generateToken() {
    byte[] bytes = new byte[32];
    secureRandom.nextBytes(bytes);
    return
Base64.getUrlEncoder().withoutPadding().encodeToString(bytes);
}
}

```

ПРИЛОЖЕНИЕ Б

(обязательное)

Интеллектуальный ассистент для проведения собеседования

Презентация

Интеллектуальный ассистент для проведения собеседования

Автор работы:

студент группы 24ВВИМ1 Духнов Олег Олегович

Научный руководитель:

к.т.н., доцент, Юрова Ольга Викторовна

Задачи и цель работы



Исследование области больших языковых моделей



Внедрение моделей в приложение



Выбор наиболее подходящего варианта

Анализ ASR-модели для транскрипции голоса



Тестирование найденных решений



Реализация приложения для проведения собеседования



Актуальность

< 1.9: острый дефицит соискателей
2.0–3.9: дефицит соискателей
4.0–7.9: умеренный уровень конкуренции
8.0–11.9: высокий уровень конкуренции
> 12: крайне высокий уровень конкуренции

Динамика hh.индекса
Информационные технологии



Интерес компаний к внедрению ИИ в свои продукты



Рост конкуренции на рынке труда IT увеличивает нагрузку на HR-специалистов.



Заинтересованность HR-специалистов в инструментах, упрощающих первичный отбор кандидатов.



Исследуемые LLM-модели



Ключевые требования: локальный запуск, использование русского языка, коммерчески допустимая лицензия, генерация вопросов, анализ ответов



Qwen3-8B



Gemma 3 4B



Granite 3.3 8B



Ministral 3 8B
Mistral 7B
Mistral Nemo 12B



Phi 4 mini 4B



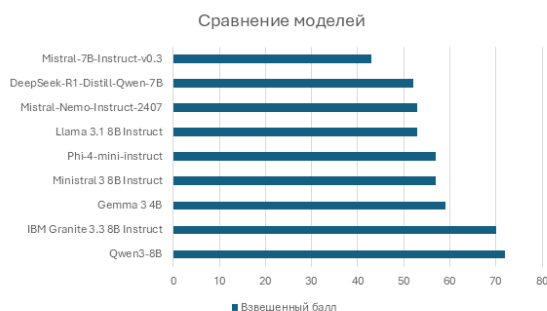
Llama 3.1 8B



DeepSeek R1 Destill Qwen 7B

Тестирование LLM-моделей

Результаты тестирования моделей в бенчмарках, баллы выставлены с помощью механизма взвешивания



Распределение мест при ручном тестировании LLM в приложении:

1. Ministral 3 8B
2. Mistral-Nemo 12B
3. Qwen 3 8B
4. Granite 8B
5. Mistral 7B
6. Llama 3.1 8B

Исследуемые ASR-модели



Ключевые требования: локальный запуск, распознавание русского и английского языков, качество транскрипции голоса



STT

Coqui STT



Vosk



NVIDIA

NVIDIA Nemp / Parakeet

Тестирование ASR-моделей



Тестирование проводилось вручную путем диктовки одинаковых текстов для каждой нейросетевой модели

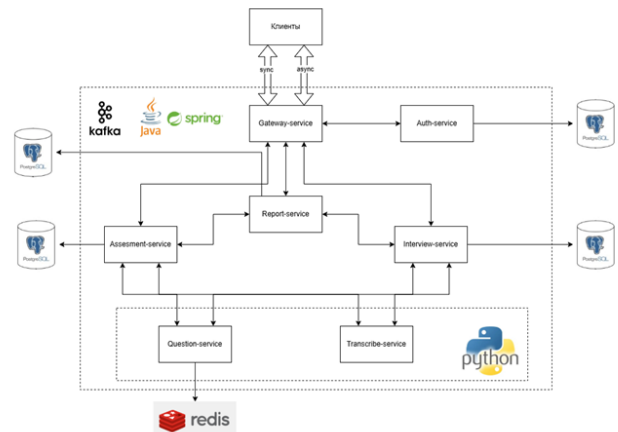
Распределение мест при ручном тестировании моделей для транскрипции голоса в приложении:

1. Whisper
2. Coqui STT
3. Vosk
4. NVIDIA Nemo / Parakeet

Жил расшифровывается как Global Interpreter Log или глобальная блокировка интерпретатора. Это мьютекс, который используется в сипетон, основной реализации петона для защиты доступа к объектам интерпретатора. Проще говоря, JIL гарантирует, что в любой момент времени только один поток может выполнять байт-код-петон. Даже если у вас многоядерный процессор и запущенный несколько потоков, одновременно может работать только один из них в рамках одного процессора. Причина существования JIL историческая и практическая. Многие внутренние структуры сипетон, включая счетчики ссылок для управления памятью, не являются потоками безопасными. Добавление тонкой блокировки для каждого отдельного объекта потребовало бы огромных усилий по переписыванию интерпретатора и привело бы к значительному падению производительности в напотоковых программах. JIL это компромисс, он делает напотоковый код быстрым и упрощает написание си расширений, но ограничивает эффективность многопоточности. Как JIL работает на практике? Каждый поток в петон перед выполнением байт-кода должен схватить JIL. Интерпретатор переключает JIL между потоками через определенную промежуток времени. Обычно каждый из 100 тиков байт-кода или по системному таймеру, что позволяет разным потокам получать скванты времени. Однако одновременно выполняется только один поток. Я не JIL на многопоточность, сильно зависит от типа задач. Без задач, связанных с водом-выводом, таких как чтение файлов, сетевые запросы, работа с базами данных, JIL практически не является проблемой. Когда поток выполняет операцию вод-вывода, он может явно отпустить JIL, потому что эта операция не требует выполнения кода петона. Персональ системы может приостановить поток и JIL освобождается до других потоков. Благодаря этому потоке вод-вывода могут работать правильно и эффективно.

Архитектура приложения

- Frontend взаимодействует только с API Gateway.
- Java-сервисы отвечают за авторизацию, интервью, HR-сценарии и отчеты.
- Python AI-сервисы отвечают за LLM и транскрипцию.





Демонстрация приложения



Заключение



Исследованы большие языковые модели, а также ASR-модели



Проведено тестирование LLM и ASR



Выбраны наиболее подходящие модели, которые были внедрены в приложение



Разработана программа, которая позволяет проводить интервью с искусственным интеллектом

