Итоговый отчет по проекту HabrRAG

1. Описание функционала интерфейса

Обзор интерфейса

Пользовательский интерфейс системы HabrRAG реализован на базе **Chainlit** - фреймворка для создания чат-интерфейсов к LLM-приложениям. Интерфейс предоставляет:

- **Веб-чат**: доступен по адресу `http://localhost:8001` с простой авторизацией (любой логин/пароль)
- **Асинхронное взаимодействие**: все запросы обрабатываются асинхронно без блокировки интерфейса
- **Сохранение истории**: все диалоги автоматически сохраняются в PostgreSQL базу данных `chainlit_db`
- **Интерактивные элементы**: кнопки для дополнительных действий (суммаризация источников)

Основной функционал RAG-системы

Процесс работы с вопросами пользователя:

1. **Ввод запроса**: Пользователь вводит вопрос в текстовое поле чата
2. **Обработка запроса**: Система отправляет запрос к FastAPI эндпоинту `/answer`
3. **RAG-пайплайн**:
   - Векторизация запроса через модель embeddings
   - Гибридный поиск по Qdrant (векторный) + BM25 (лексический)
   - Переранжирование найденных документов через Cross-Encoder
   - Генерация ответа через LLM на основе релевантного контекста
4. **Отображение результата**: В чате появляется ответ с указанием использованных источников
5. **Список источников**: Под ответом отображаются кнопки для каждого источника с названиями статей

Формат ответа: - Текстовый ответ от LLM - Список источников (до 5 документов) с заголовками и URL - Интерактивные кнопки для получения суммаризации каждого источника

Функционал суммаризации источников

Функционал суммаризации - это дополнительная возможность интерфейса, позволяющая получить краткое содержание любой статьи из списка источников. Он построен на связке FastAPI + Chainlit и использует LLM для краткого пересказа выбранной статьи.

Пользовательский сценарий

1. Пользователь задает вопрос в чате.
2. UI отправляет запрос в `/answer` и получает ответ + список источников.
3. Для каждого источника создается кнопка с названием статьи.
4. Пользователь нажимает нужную кнопку, чтобы получить краткую суммаризацию выбранной статьи.
5. В чат возвращается отдельное сообщение с суммаризацией.

Итог по интерфейсу

Интерфейс обеспечивает полный цикл взаимодействия пользователя с RAG-системой: от ввода вопроса до получения ответа с релевантными источниками и возможностью их суммаризации.

Ключевые характеристики: - **Хранение данных** - все диалоги и результаты запросов сохраняются в PostgreSQL - **Эффективная обработка** - асинхронная архитектура обеспечивает высокую отзывчивость и масштабируемость системы - **Полный функционал** - пользователь может как получать ответы на вопросы, так и запрашивать суммаризацию источников одним кликом

---

2. Инструкция по запуску

Требования

- Docker и Docker Compose
- API ключ OpenRouter

Шаги по запуску

□□□
Чанкирование и векторизация будут проведены заново Советуем:

1. Предварительно поставить в `.env` параметр `RAG_APP__SPLIT_DATASET=train[:2]`
2. Запустить поднятие docker
3. Подождать несколько минут инициализации приложения
4. Загрузить подготовленный snapshot https://drive.google.com/file/d/1sHARsAjnrDR72dgbSYqxPviEC0Q0YIbj/view?usp=sharing в qdrant прямо через интерфейс
5. Готово, вы великолепны

□□□

1. Получение API ключа

- Перейти на https://openrouter.ai/
- Зарегистрироваться и получить API ключ
- Создать `.env` файл на основе `.env.example`

2. Генерация секрета для Chainlit

- Выполнить команду `chainlit create-secret` и скопировать сгенерированный секрет
- Добавить в `.env` файл: `CHAINLIT_AUTH_SECRET=ваш_секрет`

3. Запуск через Docker

```
docker-compose up --build
```

Сервисы и доступы

Ниже приведены доступные сервисы и их параметры подключения:

FastAPI

- URL: http://localhost:8000/docs

Chainlit (чат-интерфейс)

- URL: http://localhost:8001
- Логин: любой
- Пароль: любой

Qdrant (векторная база данных)

- URL: http://localhost:6333/dashboard#/collections

Langfuse (логирование и оценка)

- URL: http://localhost:3000

PostgreSQL

- Host: `localhost`
- Host name/address: `db`
- Port: `5432`
- Username: `chainlit_user`
- Password: `chainlit_pass`

PgAdmin

- URL: http://localhost:8080
- Логин: `admin@admin.com`
- Пароль: `admin`

API Эндпоинты

GET /health

Проверка статуса приложения

```
curl http://localhost:8000/health
```

POST /answer

Получить ответ от RAG системы. Запросы можно отправлять через Swagger UI по адресу `http://localhost:8000/docs`

Примеры запросов:

```
{ "query": "Who was the original owner of the lot of items being sold?" }

{ "query": "What are some of the skills taught in the Trail Patrol Training course?" }

{ "query": "Who were the two convicted killers that escaped from an upstate New York maximum-security prison?" }
```

Сохранение историй чатов

Истории чатов автоматически сохраняются в PostgreSQL базе данных. Chainlit управляет созданием таблиц и хранением данных пользователей, сессий и сообщений.

- **Чаты в Chainlit** кэшируются в базу данных `chainlit_db`
- **Вопросы и ответы RAG** кэшируются в базу данных `postgres_db` в таблице `answer_cache`

---

3. Финальные метрики качества системы

Параметры запуска

Оценка проводилась на датасете из 511 примеров. Использовались следующие параметры:

- **Модель векторизации**: intfloat/multilingual-e5-large
- **Кросс-энкодер**: cross-encoder/ms-marco-MiniLM-L-6-v2
- **Top K (финальное количество кандидатов)**: 5
- **BM25 K**: 5
- **Qdrant K**: 5
- **Вес BM25 в ансамбле**: `0.4`
- **Вес Qdrant в ансамбле**: `0.6`

Общие результаты

Метрики ретривера

- **Hit@K**: Доля вопросов, где правильный чанк находится в топ-K результатах (Hit@1: `0.2857`, Hit@2: `0.4677`, Hit@3: `0.6204`, Hit@5: `0.8943`).
- **Recall@K**: Доля правильных чанков, найденных в топ-K результатах (Recall@1: `0.2857`, Recall@2: `0.4677`, Recall@3: `0.6204`, Recall@5: `0.8943`).
- **MRR (Mean Reciprocal Rank)**: `0.4892` — Средний обратный ранг первого правильного чанка (чем выше, тем лучше ранжирование).
- **ID-based Context Recall**: `0.8943` — Доля правильных чанков, включенных в предоставленный контекст для генерации ответа.
- **ID-based Context Precision**: `0.1789` — Точность контекста: доля правильных чанков среди всех предоставленных.

Метрики генератора

- **BERTScore**:
  - Precision: `0.8282` — Точность семантического сходства между сгенерированным и эталонным ответом.
  - Recall: `0.872` — Полнота семантического сходства.
  - F1: `0.8494` — Гармоническое среднее точности и полноты семантического сходства.
- **BLEU Score**: `0.0287` — Метрика точности совпадения n-грамм (1-4) между сгенерированным и эталонным текстом; низкие значения типичны для генеративных задач.
- **ROUGE Score (1-gram F-measure)**: `0.1562` — Совпадение униграмм (отдельных слов) между сгенерированным и эталонным ответом.
- **ROUGE Score (L F-measure)**: `0.1494` — Совпадение самых длинных общих подпоследовательностей слов.
- **Similarity Scores**: `0.9023` — Общая схожесть ответов, основанная на векторных представлениях.
- **CHRF Score**: `0.249` — Character-level F-score, измеряющий совпадение на уровне символов.
- **Non-LLM String Similarity**: `0.1074` — Строковая схожесть без использования языковых моделей (Levenshtein distance).

Анализ результатов

Ретривер

- Ретривер показывает хорошую производительность на топ-5 результатах (Hit@5 и Recall@5 около `89%`), что указывает на эффективное извлечение релевантных чанков.
- MRR на уровне `0.4892` свидетельствует о среднем качестве ранжирования.

Генератор

- BERTScore показывает высокие значения (F1 `~0.85`), что говорит о хорошем семантическом сходстве сгенерированных ответов с эталонными.
- Высокий Similarity Score (`0.9023`) подтверждает хорошую релевантность ответов.

Приложения

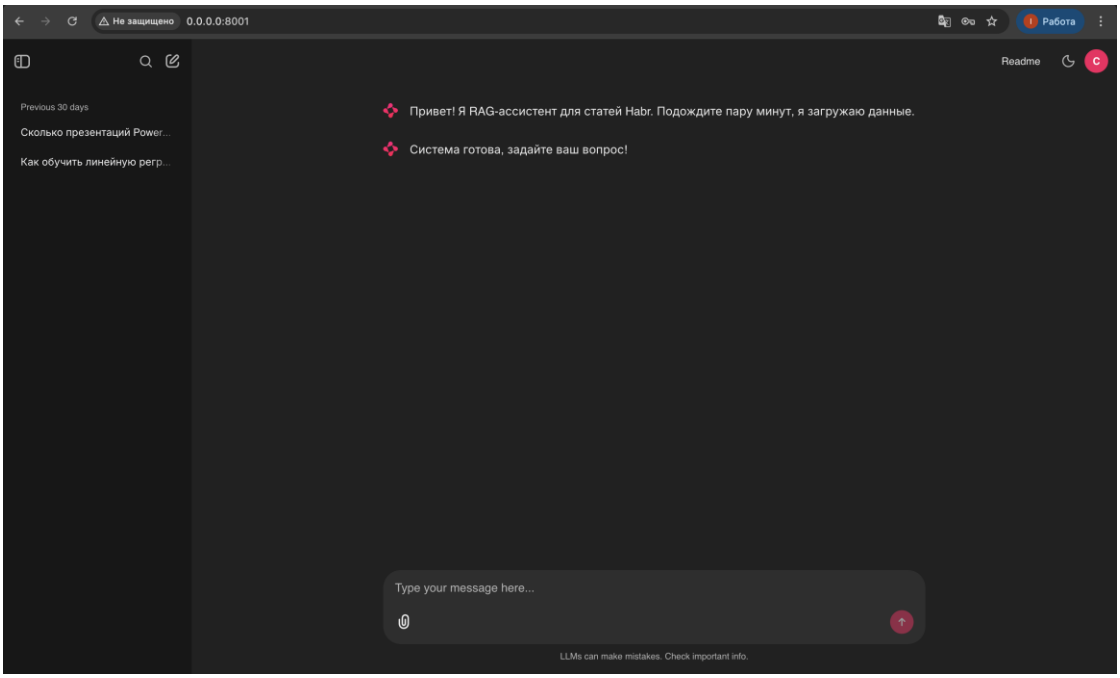

Рисунок 1. Диалоговое окно

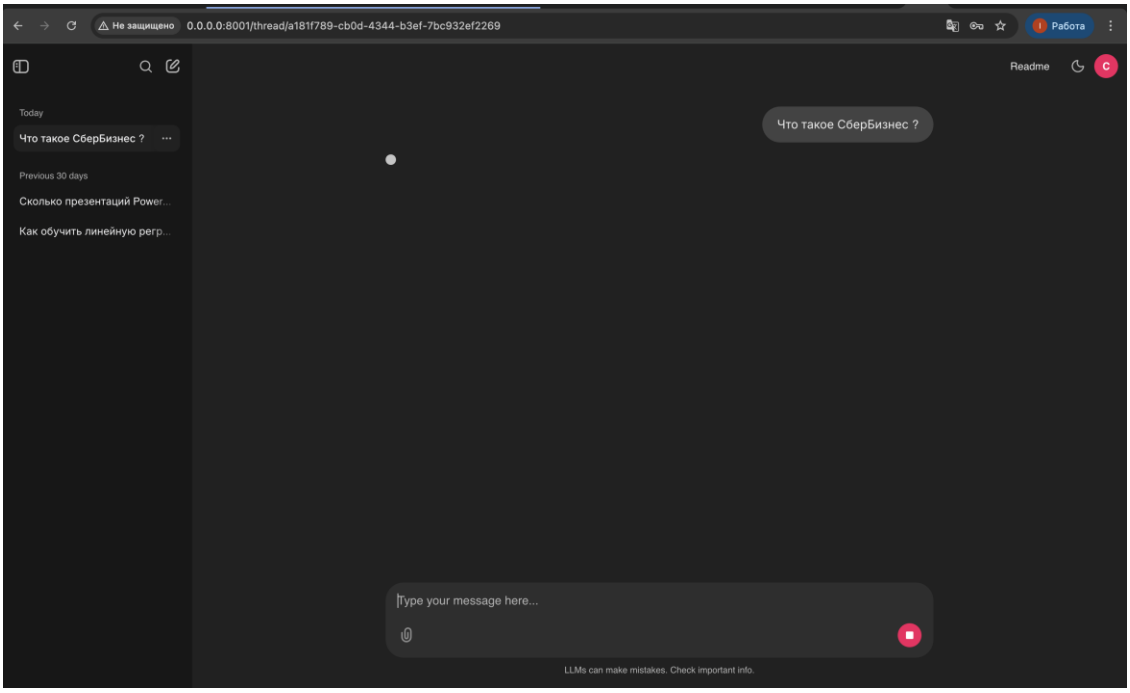

Рисунок 2. Постановка и отправка вопроса в RAG систему

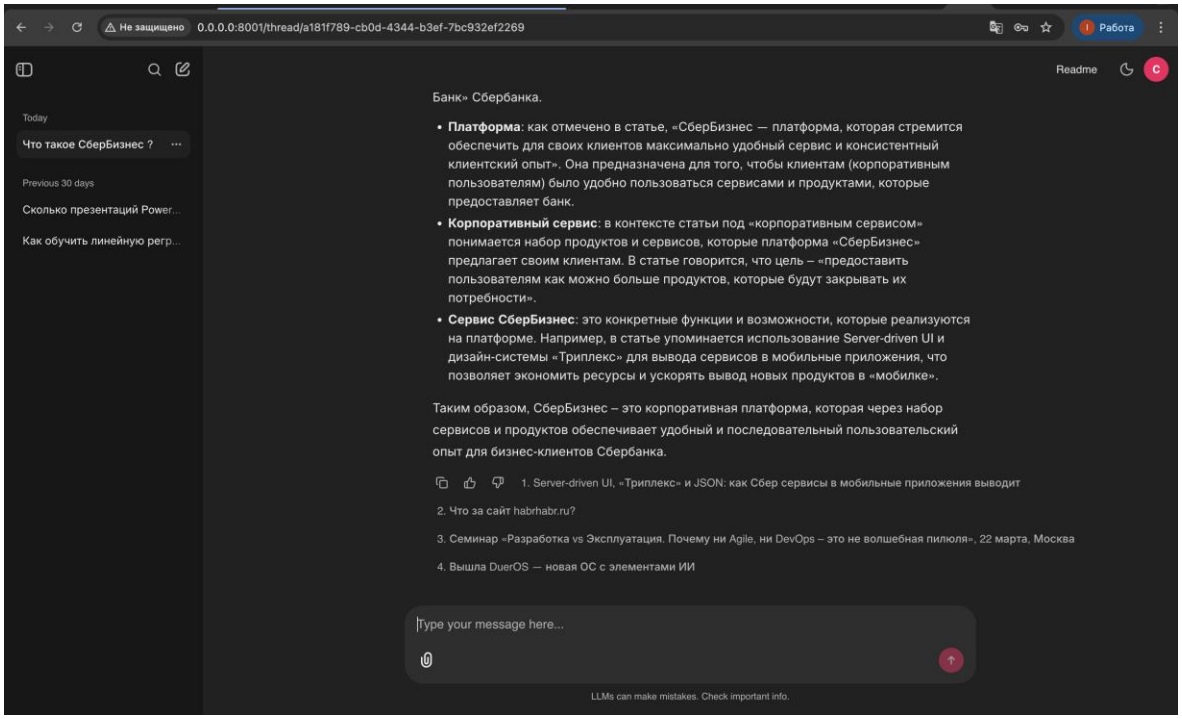

Рисунок 3. Ответ RAG системы на вопрос

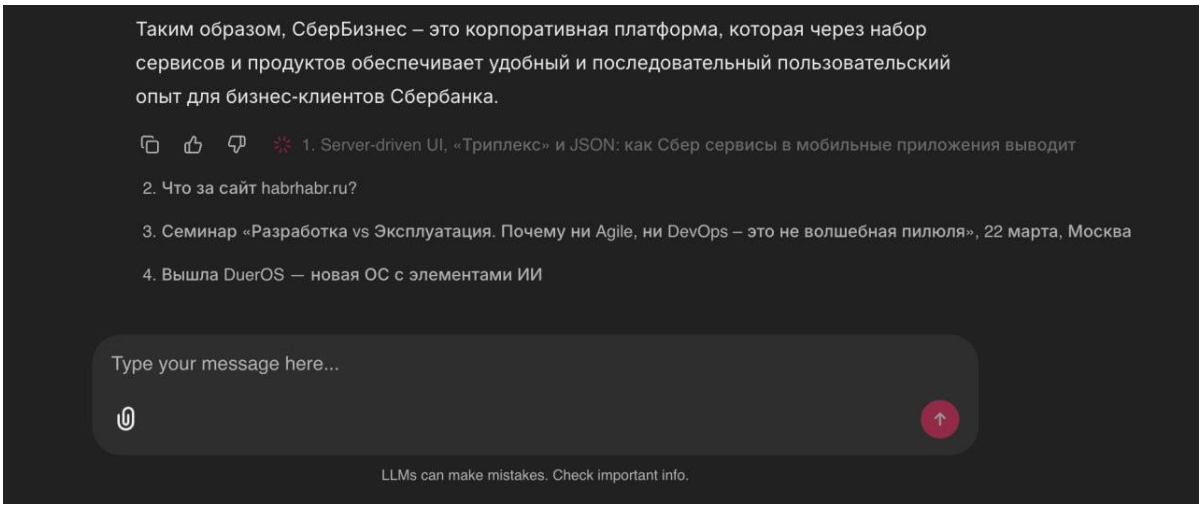

Рисунок 4. Кнопки получения суммаризации статей

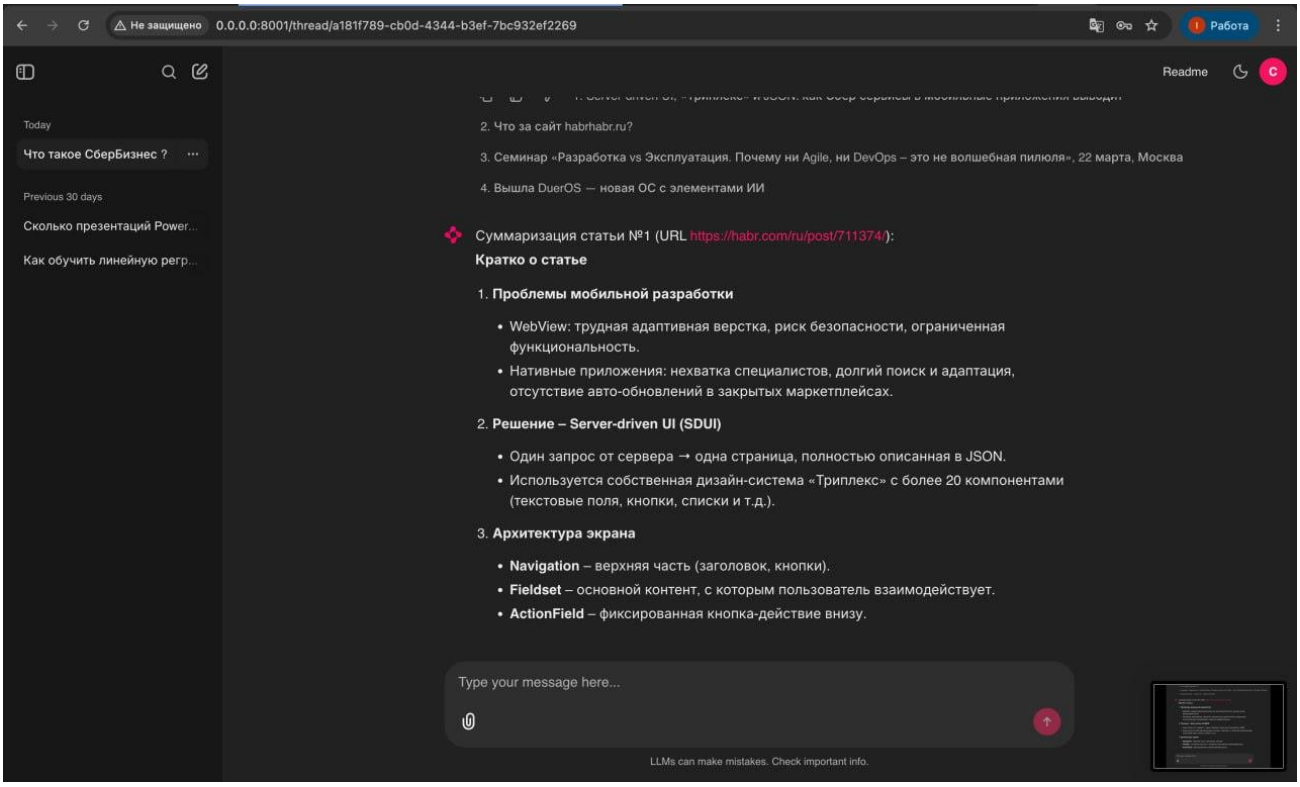

Рисунок 5. Саммари выбранной статьи