Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и

Радиоэлектроники

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №5, 6

по курсу “Естественно-языковой интерфейс интеллектуальных

систем”

Вариант 1

Выполнили:

Студенты гр. 221701 Абушкевич А. А.

Робилко Т. М.

Проверил: Крапивин Ю. Б.

Минск

2025

**Цель работы:**

* Освоить принципы проектирования и разработки естественно-языковых интерфейсов с пользователем в диалоговых системах.

**Задание:**

* Изучить основы моделирования процесса общения.
* Приобрести навыки проектирования элементов диалоговых систем.
* Изучить основы создания диалоговых систем с поддержкой естественного языка
* Закрепить навыки программирования при решении задач организации диалогового взаимодействия с поддержкой естественного языка.

**Используемые библиотеки:**

Приложение представляет собой клиент-серверный чат-бот с user-friendly интерфейсом под смартфоны с базовыми возможностями мессенджера - ведение диалога, сохранение сообщений, выставление реакций на сообщения, редактирования информации “О себе”.

Клиентная часть приложения реализована на языке C# с использованием фреймворка Avalonia UI. Серверная часть реализована на ЯП Python с использованием фреймворка FastAPI. Выбор таких средств обусловлен популярностью данной связки, преимуществом перед стандартными средствами языка C# и многими аналогами, возможностью использования асинхронности и четкой документацией.

В нашем приложении также использована Chroma DB для хранения векторных эмбеддингов предметной области - предварительно размеченных html файлов, содержащих информацию о кинофильмах (название, жанр, отзывы, оценка, режиссер, год выпуска и др.). Для хранения отладочных логов и сообщений сервера использовали базу данных SQLite в виду ряда особенностей: *простота и автономность* (не требуется установка и настройка отдельного сервера баз данных), *локальное хранение* (все данные хранятся на компьютере пользователя, что удобно для автономных или тестовых приложений), *быстрая настройка* и *низкие системные требования*.

Библиотеки языка Python, использованные в ЛР:

1. langchain - современный open-source фреймворк для создания приложений на основе больших языковых моделей (LLM), таких как GPT, Claude, Llama и другие. В разработке приложения использовался для:
   1. интегрирования языковых моделей с собственными данными;
   2. предоставления готовых абстракций и компонентов для ускорения разработки;
   3. создания цепочек последовательностей с данными и AI-моделями;
2. typing - стандартный модуль Python, предназначенный для поддержки статической типизации и аннотаций типов. Она предоставляет набор инструментов для описания типов переменных, аргументов функций и возвращаемых значений, что помогает улучшить читаемость кода, документацию и позволяет использовать статические анализаторы типов. В данной ЛР использовался компонент List (для аннотирования списков) и Optional (для обозначения принадлежности переменной к определенному типу) ;
3. logging - встроенный модуль Python для ведения логов, который позволяет отслеживать события в приложении, фиксировать ошибки, предупреждения и другую важную информацию во время выполнения программы. Модуль широко используется для отладки, мониторинга и аудита работы приложений. В данной ЛР использовался для формирования логов DEBUG, WARNING, INFO;
4. requests - популярный и удобный модуль для Python, который значительно упрощает работу с HTTP-запросами. Она создана для того, чтобы делать отправку запросов и получение ответов максимально простой и понятной, избавляя разработчика от необходимости вручную формировать URL, кодировать данные или управлять соединениями. Собственно, для этого библиотека и использовалась в ЛР;
5. urllib - стандартная библиотека, предоставляющая инструменты для работы с URL-адресами и выполнения веб-запросов. В ЛР использовалась для открытия и чтения URL адресов, разборку и сборку URL, кодирования параметров запроса и обработку ошибок при работе с URL;
6. sqlite3 - встроенный в стандартную библиотеку Python модуль для работы с базой данных SQLite, которая представляет собой легковесную встраиваемую реляционную СУБД. SQLite не требует отдельного сервера и хранит всю базу данных в одном файле, что делает её удобной для небольших и средних приложений.;
7. datetime - встроенный модуль Python, предоставляющий классы и функции для работы с датами и временем. Использовалась для фиксирования даты и времени и запись в лог программы;
8. json - библиотека, предназначенная для сериализации и десериализации;
9. uvicorn - это высокопроизводительный ASGI-сервер для запуска асинхронных веб-приложений на Python. В приложении использовался для базирования FastAPI сервера;
10. fastapi - высокопроизводительный веб-фреймворк для создания API на Python, появившийся в 2018 году и быстро завоевавший популярность благодаря своей скорости, простоте и удобству использования. В данной лабораторной работе выполнял функции сервера в клиент-серверной архитектуре;
11. uuid - предназначен для генерации универсально уникальных идентификаторов (UUID) длиной 128 бит, которые используются для создания уникальных значений в различных приложениях;
12. shutil - библиотека Python, предоставляющая набор высокоуровневых функций для работы с файлами и каталогами;
13. pydantic - библиотека для валидации и сериализации данных в Python, которая позволяет описывать структуры данных с помощью обычных классов и аннотаций типов, а затем автоматически проверять и преобразовывать входные данные согласно этим описаниям;

## **Описание предметной области “Кинематограф”**

Предметная область данной лабораторной работы – **"Кинематограф"**. Разрабатываемая диалоговая система предназначена для помощи пользователям в поиске кинофильмов, получения отзывов, справочной информации и персональных рекомендаций на основе их запросов, сформулированных на естественном языке. Система позволяет уточнять предпочтения пользователя, задавать уточняющие вопросы и возвращать релевантные результаты.

Ключевая особенность разработанной системы — **использование открытых моделей семейства Gemma 3** от компании Google для обработки естественного языка.

### **Описание языковой модели**

Для выполнения лабораторных работ была выбрана модель Gemma3 1b ввиду своей легковесности и удобства тестирования и развертывания. Gemma 3 — это нейросетевая модель на основе архитектуры Transformer, аналогичной применяемой в GPT-4 и других LLM. Модель выпускается в нескольких вариациях по размеру: на 1 млрд, 4 млрд, 12 млрд и 27 млрд параметров. Даже 27 млрд — это относительно небольшое количество параметров по меркам флагманских ИИ (для сравнения: GPT-3 содержала 175 млрд параметров, а GPT-4, по некоторым данным, может достигать 1 трлн параметров). Но, несмотря на это, Gemma 3 демонстрирует высокую эффективность благодаря оптимизации в обучении.

Модель является мультимодальной, помимо текста на вход она воспринимает изображения и видеофрагменты и способна анализировать их содержимое. В состав модели интегрирован модуль компьютерного зрения. В результате Gemma 3 может обрабатывать задачи, совмещающие текст и визуальную информацию. Например, модель способна распознавать объекты на фото, отвечать на вопросы по картинке, сравнивать два изображения или читать текст, содержащийся внутри изображения (OCR). Такие возможности приближают Gemma 3 к уровню GPT-4, у которого также есть мультимодальная версия, однако в случае Gemma все эти функции доступны открыто для всех пользователей локально — прямо на их устройствах.

Согласно внутренним тестам и независимым рейтингам, Gemma 3 сегодня является одним из лидеров среди открытых компактных моделей. В открытом соревновании LMArena она заняла первое место в своей категории, набрав свыше 1330 баллов и опередив другие модели сопоставимого размера. По совокупности навыков (понимание языка, логика, креативность) Gemma 3 приближается к возможностям более крупных закрытых систем. Хотя напрямую конкурировать с громадными моделями вроде GPT-4 она не может из-за разницы в масштабах, по показателю «качество/размер» Gemma 3 задает фактически новый стандарт в индустрии.

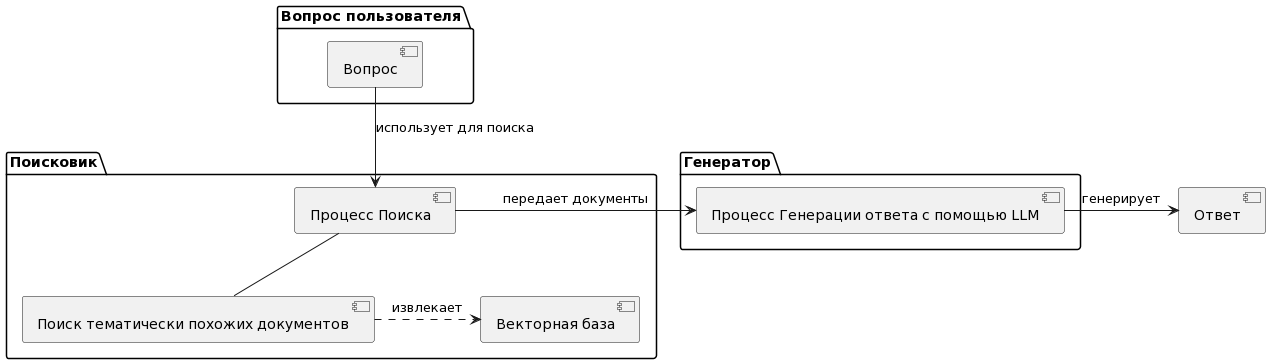
**Использование LLM в системе**

Gemma3: 1b применяется в следующих задачах:

1. **Понимание пользовательского запроса**Языковая модель анализирует текстовые запросы пользователей, выделяет основные **намерения**, **жанры или другие характеристики** и подготавливает данные для поиска.
2. **Генерация ответов**Модель формирует ответы на естественном языке, включая приветствия, уточняющие вопросы, сообщения с результатами (могут включать ссылки или изображения) и прощания. Это делает взаимодействие с системой более понятным и дружелюбным.
3. **Поддержание контекста диалога**При каждом новом запросе передаётся история предыдущих сообщений. Это позволяет системе **вести связный диалог**, учитывать уже полученную информацию и не задавать повторные вопросы.

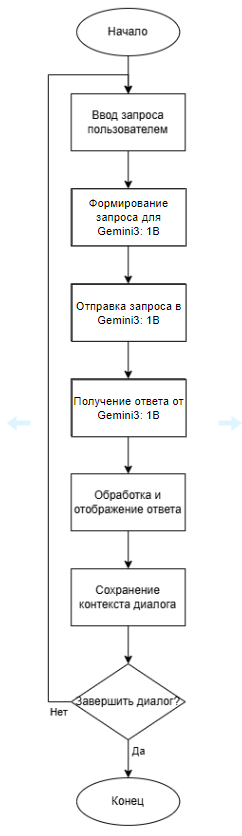
Тем не менее, для целей лабораторной работы и тестирования базовой функциональности диалоговой системы этих возможностей достаточно.

**Структурно-функциональная схема разработанного приложения**

****

*рис. 1.1 - структурно-функциональная схема*

**Логическая структура сценария диалога**



**Структуры данных**

В приложении определены модели для “общения” клиента и сервера, что по сути и является структурой данных приложений. Модели:

* DocumentInfo - класс, предназначенный для передачи и хранения документа (html-страницы):
  + id - идентификатор файла;
  + filename - имя файла;
  + timestamp - метка загрузки файла;
  + User - класс для хранения пользователя и информации о нем:
    - * name - имя пользователя;
      * about - информация “О себе”;
* MessageContent - класс для представления содержимого сообщения в чате:
  + images - список путей к изображениям;
  + links - список путей к ссылкам;
  + text - текст сообщения;
  + MessageMetadata - класс для представления метаданных о сообщении:
    - * + sent - дата отправки;
        + user - объект класса пользователя;
* MessageReactions - класс для реализации функционала выставления реакций на сообщения:
  + rating - рейтинг сообщения;
* Message - класс, предназначенный представления сообщения:
  + content - объект класса MessageContent;
  + metadata - объект класса MessageMetadata;
  + reactions - объект класса MessageReactions;
* RateMessageRequest - класс для представления сообщения отправки реакции:
  + message - объект класса Message;
  + rating - отметка;
  + ChatRequest - класс для представления отправки сообщения:
    - * session\_id - идентификатор сессии;
      * message - объект класса Message;
* ChatResponse - класс для представления получения сообщения:
  + session\_id - идентификатор сессии;
  + message - объект класса Message;

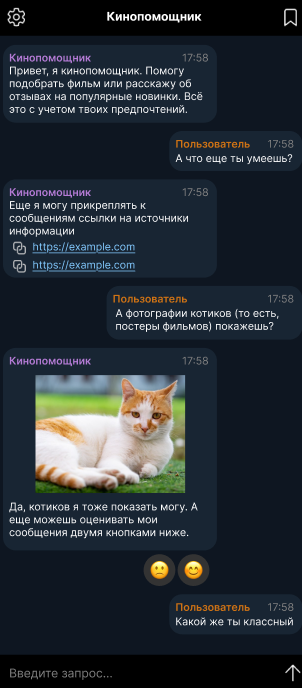
**Входная информация** (запрос пользователя) представляет собой текстовое сообщение, введенное пользователем в интерфейсе чата (тип данных String).

**Выходная информация (ответ пользователю)** представляет собой структурированную модель сообщения, которая отображается пользователю в окне чата.

**Результаты проектирования внешнего интерфейса с пользователем,**

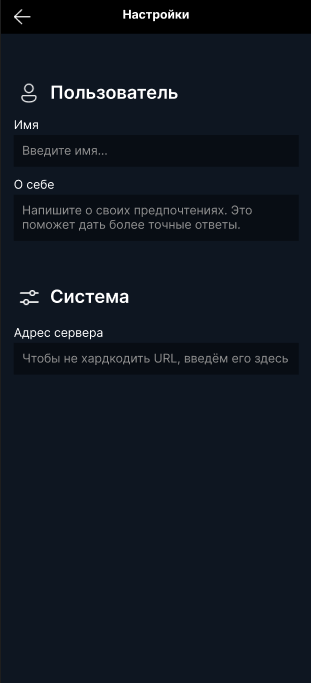
**удобного для организации и поддержания естественно-языкового**

**интерфейса взаимодействия система-пользователь**



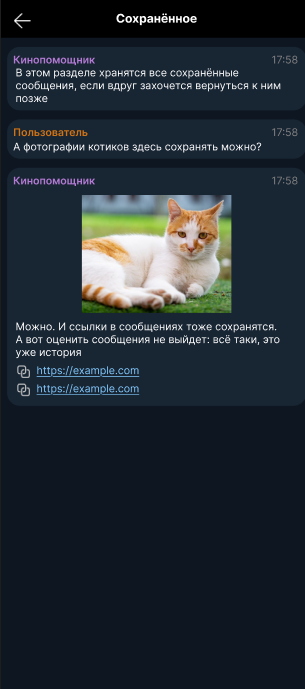
*рис. 2.1 - чат приложения*

На рис. 2.1 представлена вкладка чата приложения, которое по сути является также и главным экраном. Доступны кнопки настройки, а также переход в сохраненные сообщения.



*рис. 2.2 - настройки*

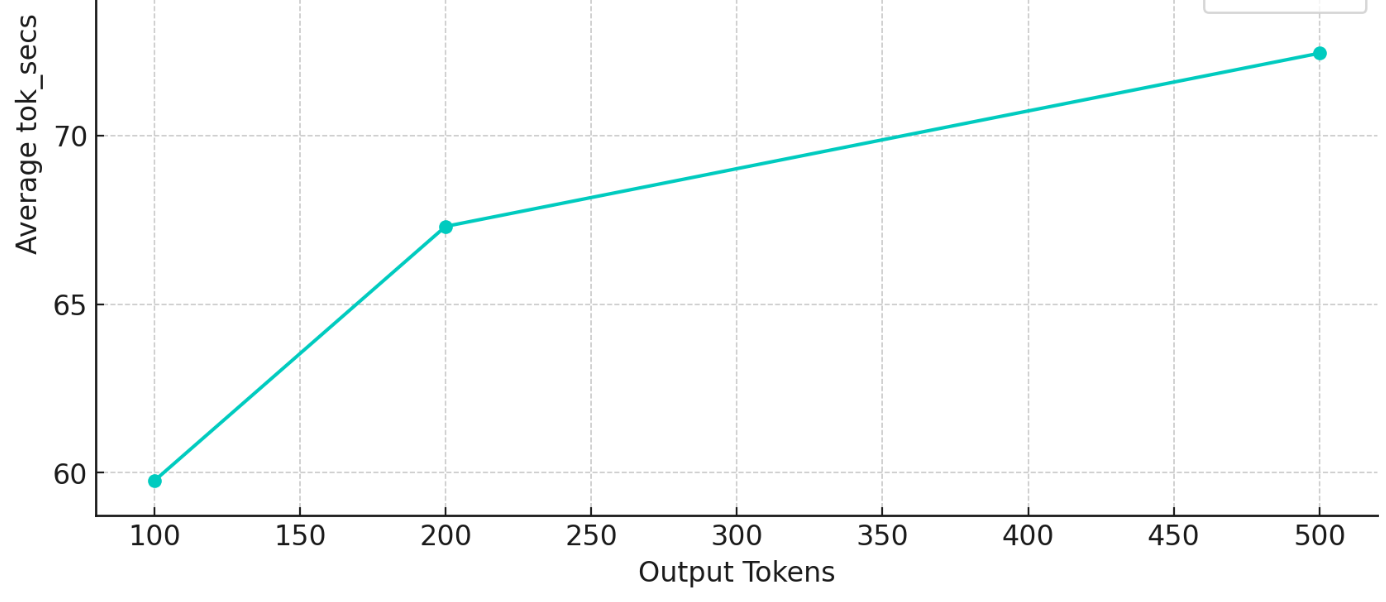
На рис. 2.2 представлена вкладка настроек. Пользователь может ввести имя, информацию о себе. Оба этих параметра будут учитываться моделью при составлении ответов. Доступна функция ввода адреса локального сервера, на котором собственно находится база с моделью.



*рис. 2.3 - сохраненные сообщения*

На рис. 2.3 представлена вкладка с сохраненными сообщениями. Здесь представлены те сообщения, которые пользователь в процессе диалога сохранил себе для дальнейших действий. Сообщения могут как добавляться, так и удаляться.

**Оценка быстродействия:**

****

Тестирование проводилось на процессоре Intel Core I7 8700K, видеокарта GTX 1050Ti. Вот несколько возможных причин, почему данные выглядят именно так:

* Сетевые задержки и загрузка модели;
* Вариативность модели;
* Нелинейность генерации текста;
* Особенности алгоритмов генерации;

**Выводы:**

В ходе лабораторной работы было реализовано десктопное клиент-серверное приложение, взаимодействующая с RAG системой. Основное внимание уделялось построению диалога между пользователем и ИИ, хранению истории сообщений и правильной обработке входных и выходных данных.

Были изучены и применены:

* современные подходы к хранению диалогов с использованием реляционной базы данных и разметке документа, а также его последующего хранения в виде эмбеддинга в векторной БД (ChromaDB, SQLite);
* современные подходы по проектированию сложных систем с использованием разных языков программирования;
* обработка и отображение ответа модели с сохранением форматирования;
* базовые элементы веб-разработки на FastAPI совместно с Avalonia UI (роутинг, шаблоны, сессии).

**Перспективы использования приложения:**

Разработанное приложение обладает хорошей основой для дальнейшего развития и может быть применено в следующих направлениях:

* Сервисы по рекомендации и поиску кинофильмов на подобие Кинопоиск.ру, [IVI.ru](http://ivi.ru);
* Сервисы для просмотра кинофильмов;
* Интеграция непосредственно с этими сервисами в качестве компонента мобильного, веб или десктоп приложения;
* Перевод API на Telegram API;