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 №7

по курсу “Естественно-языковой интерфейс интеллектуальных

систем”

Вариант 33

Выполнили:

Студенты гр. 221701 Каптилович М.А.

Максимков М.С

Слобода Н.С.

Проверил: Крапивин Ю. Б.

Минск

2025

**Цель работы:**

* Освоить на практике основные принципы реализации информационно-поисковых систем и методы оценки качества их работы.

**Задание:**

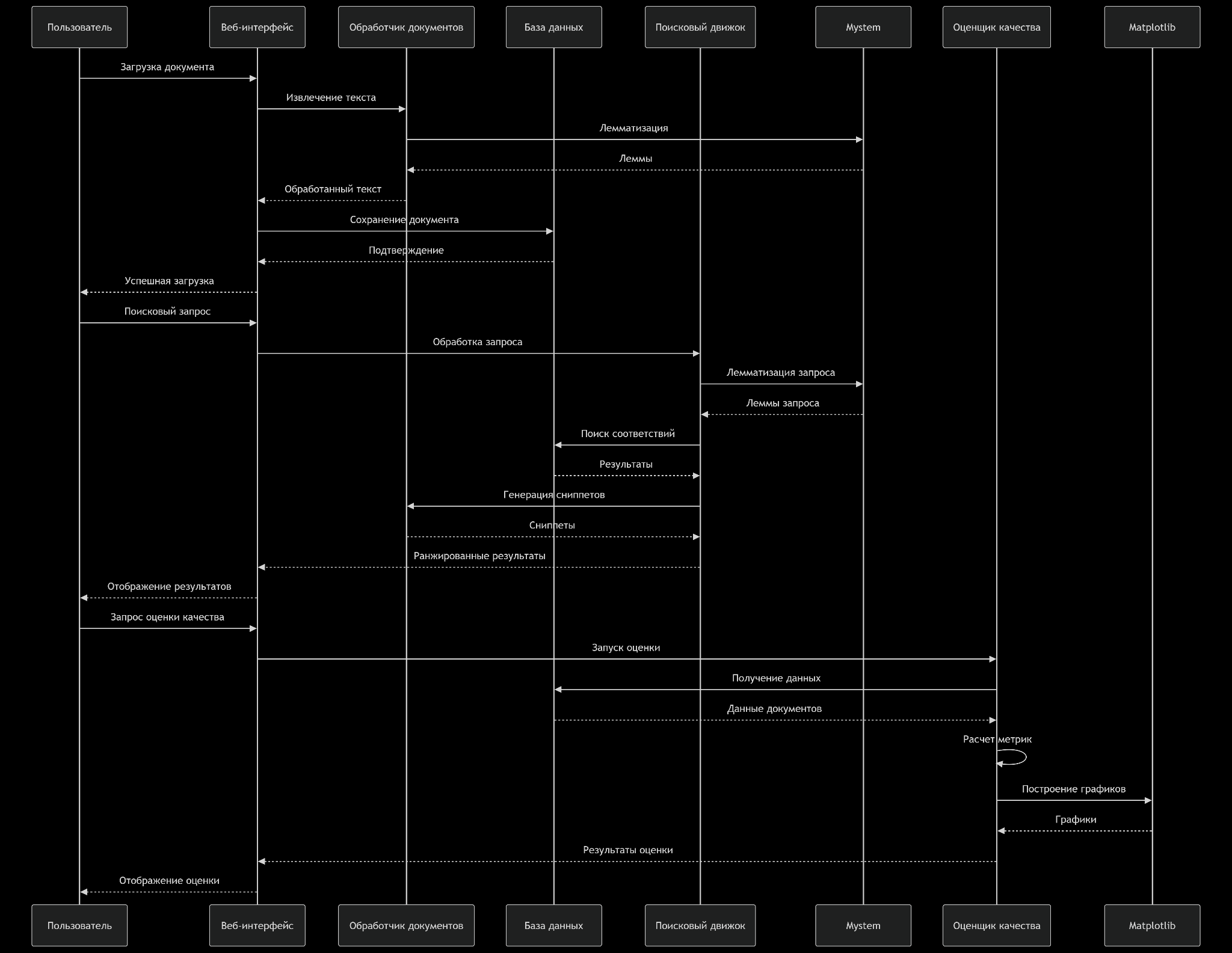
Разработать поисковую систему, удовлетворяющую следующим требованиям:

* Сфера применения - Локальная вычислительная сеть
* Стратегия поиска - векторная
* Язык поиска - английский
* Вход системы - множество естественно-языковых текстов, по которым осуществляется поиск
* Выделение ключевых слов документов осуществляется системой автоматически в соответствие с формулой из методического пособия
* Система должна позволять пользователю формулировать ЕЯ-запрос
* Выход системы - список документов, релевантных запросу пользователя, в соответствие с моделью поиска, согласно варианту
* Результаты поиска должны содержать: активную ссылку на документ, список слов запроса присутствующих в документе
* На основании информации о существующих метриках, наиболее часто использующихся для оценки качества работы систем информационного поиска, требуется дать оценку работы системы
* Вычисление оценок, получаемых на основании метрик, реализовать программно, путем вызова соответствующего подменю, и отображать в виде таблиц и графиков
* Интерфейс системы должен быть предельно простым и доступным для пользователей любого уровня, содержать понятный набор инструментов и средств, а также help-средства

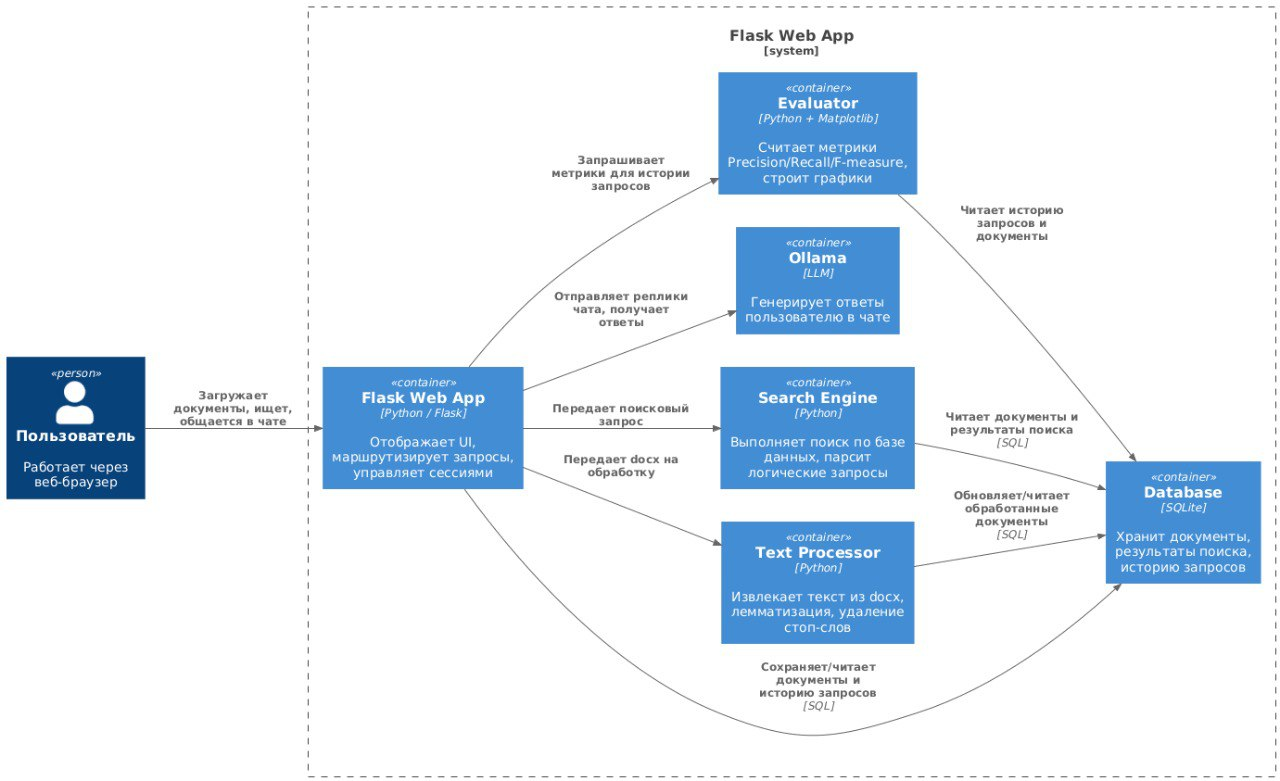
**Используемые библиотеки:**

* Flask (v3.1.2) - микрофреймворк для создания веб-приложений на Python. Использовался для построения backend-части приложения, обеспечивая роутинг, обработку HTTP-запросов, рендеринг HTML-шаблонов, работу с сессиями для хранения истории поиска и функциональность загрузки файлов.
* Werkzeug (v3.1.3) - библиотека утилит WSGI, входящая в состав Flask. Была задействована для безопасной обработки имен файлов и предоставления вспомогательных функций для веб-разработки.
* python-docx (v1.2.0) - библиотека для работы с DOCX-файлами. Использовалась для извлечения текста из загружаемых документов формата .docx.
* pymystem3 (v0.2.0) - морфологический анализатор для русского языка, представляющий собой Python-обертку над MyStem от Yandex. Применялась для лемматизации текста и приведения слов к их нормальной форме, что является ключевым этапом обработки поисковых запросов и документов.
* nltk (v3.9.1) - комплексная библиотека для обработки естественного языка (NLP). В проекте использовалась для фильтрации стоп-слов русского языка, которые исключаются из обработки для повышения релевантности поиска.
* matplotlib (v3.10.6) - библиотека для построения графиков и визуализации данных. Была применена для генерации графиков, отображающих метрики оценки качества поиска (precision, recall и F-меру), с последующим сохранением результатов в формате PNG.
* sqlite3 - встроенный модуль Python для работы с базами данных SQLite. Обеспечивал хранение документов, выполнение поисковых запросов к индексированным данным.
* re - встроенный модуль Python для работы с регулярными выражениями. Использовался для очистки текста от знаков препинания в процессе предобработки.
* os - встроенный модуль Python для взаимодействия с операционной системой. Применялся для работы с файловой системой: создания папок для загрузок, управления файлами и формирования путей.
* time - встроенный модуль Python для работы со временем. Был задействован для генерации уникальных имен файлов графиков с целью предотвращения их кэширования браузером.
* Ollama - локальная инфраструктура для запуска языковых моделей. Используется для чат-функциональности с документами. Предоставляет API для генерации контекстуальных ответов

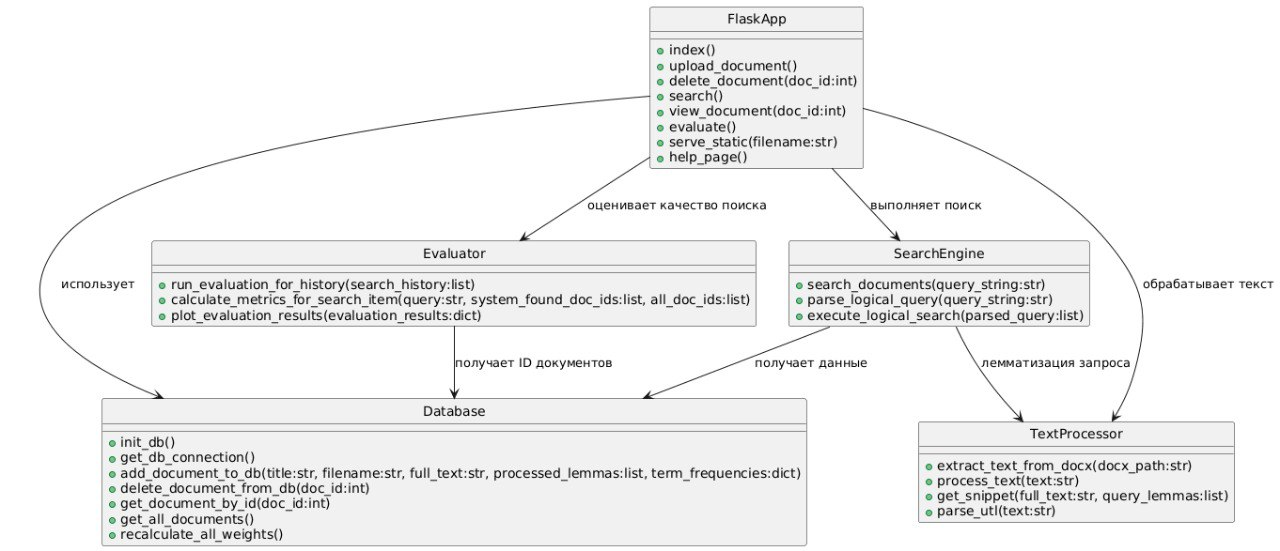
**Структурно-функциональная схема разработанного приложения (диаграмма последовательности)**

****

**Структурно-функциональная схема**

****

**Диаграмма классов системы**

****

**Описание структур данных для хранения информации**

Структуры базы данных (SQLite):

Таблица documents:

* id - уникальный идентификатор документа
* title - название документа
* filename - имя файла (уникальное)
* upload\_date - дата загрузки в формате ISO
* full\_text - полный текст документа

Таблица terms:

* id - уникальный идентификатор термина
* term\_text - текст термина в нормальной форме (уникальный)
* doc\_count - количество документов, содержащих термин (P\_i)

Словари: term\_frequencies, search\_results, search\_history, evaluation\_metrics, evaluation\_results\_by\_query

Списки: processed\_lemmas

**Информация о тестовой коллекции документов:**

Тестовая коллекция документов представляет собой набор текстов жанра сказки, также была использована научная литература. Длина каждого файла составляет от 50 до 3000 слов. Наше приложение загружает и работает только с файлами .docx формата. В качестве одного из примеров использовалась сказка про “Колобка”.

**Алгоритмы обработки данных:**

1. Предобработка текста

* Начало
* Удалить знаки препинания с помощью регулярных выражений
* Привести текст к нижнему регистру
* Выполнить лемматизацию с помощью Mystem:
  + Разбить текст на токены
  + Для каждого токена получить нормальную форму (лемму)
* Отфильтровать результаты:
  + Удалить стоп-слова (используя nltk.corpus.stopwords)
  + Удалить не-буквенные токены
* Подсчитать частоту каждой леммы
* Вернуть список лемм и словарь частот
* Конец

2. Алгоритм добавления документа

* Начало
* Добавить запись о документе в таблицу documents
* Для каждого термина в словаре частот:
  + Если термина нет в таблице terms, добавить его
  + Увеличить счетчик документов для термина (doc\_count)
* Конец

3. Алгоритм парсинга запроса

* Начало
* Разбить запрос на токены
* Инициализировать текущий оператор как "AND"
* Для каждого токена:
  + Если токен - оператор (AND, OR, NOT), установить его как текущий
  + Иначе:
    - Обработать токен (лемматизация, фильтрация)
    - Добавить пару (текущий\_оператор, лемма) в результат
    - Сбросить текущий оператор на "AND"
* Вернуть список пар
* Конец

4. Алгоритм загрузки документа

* Начало
* Валидация URL (только HTTP/HTTPS схемы)
* Предварительный запрос (HEAD или частичный GET) для анализа метаданных
* Проверка Content-Type и Content-Length
* Определение типа файла по магическим числам:
  + DOCX: ZIP-архив с word/document.xml
* Потоковая загрузка с контролем размера
* Вычисление хеша SHA-256 для верификации
* Сохранение во временное место с последующей валидацией
* Финальное перемещение в целевую директорию
* Конец

5. Форматирование текста ассистента

* Начало
* Проверить наличие файла в запросе
* Проверить расширение файла (должно быть .docx)
* Сохранить файл на сервере
* Извлечь текст из файла
* Обработать текст (лемматизация, подсчет частот)
* Добавить документ в базу данных
* Вернуть сообщение о результате
* Конец

6. Алгоритм работы чат-системы

* Начало
* Пользователь открывает документ и нажимает кнопку "Чат"
* Открывается модальное окно с интерфейсом чата
* Пользователь вводит вопрос и отправляет его
* Frontend отправляет AJAX-запрос на endpoint /chat\_with\_doc
* Сервер получает ID документа и сообщение пользователя
* Из базы данных извлекается полный текст документа
* Формируется промпт с контекстом документа
* Отправляется запрос к модели Ollama
* Полученный ответ возвращается клиенту
* Ответ отображается в интерфейсе чата
* Конец

**Интеллектуальность**

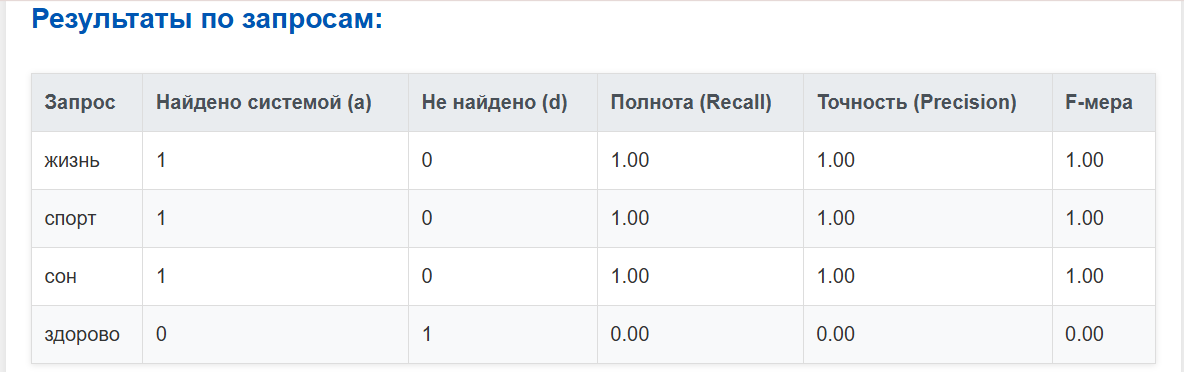
Разработанное приложение интегрировано с большой языковой моделью Ollama, которая реализует чат с пользователем прямо в приложении. Данный функционал добавляет в разработанное приложение интеллектуальность. Для разработки была использована модель saiga\_llama3\_8b\_gguf.

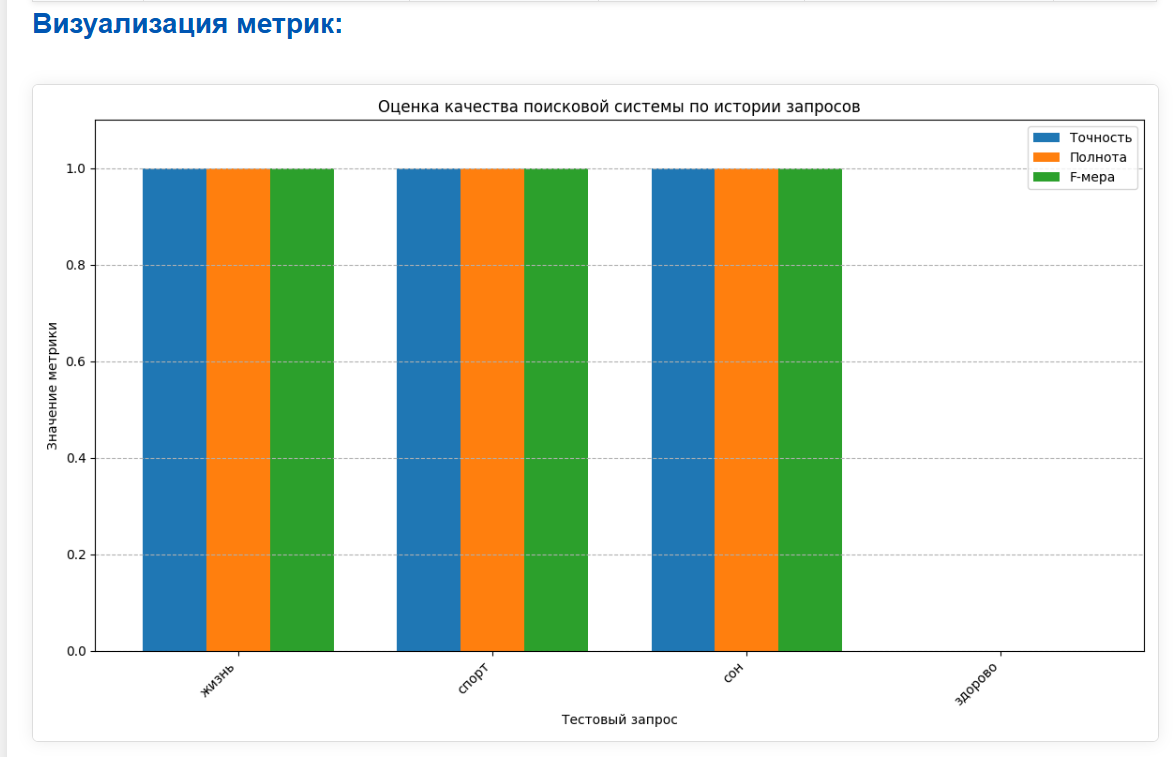
**saiga\_llama3\_8b\_gguf** — это оптимизированная версия русскоязычной чат-модели Saiga3, основанной на архитектуре Meta-Llama-3-8B-Instruct. Модель переведена в формат GGUF, что позволяет эффективно запускать ее на потребительском оборудовании (включая CPU и комбинированные CPU+GPU конфигурации).

**Основные характеристики:**

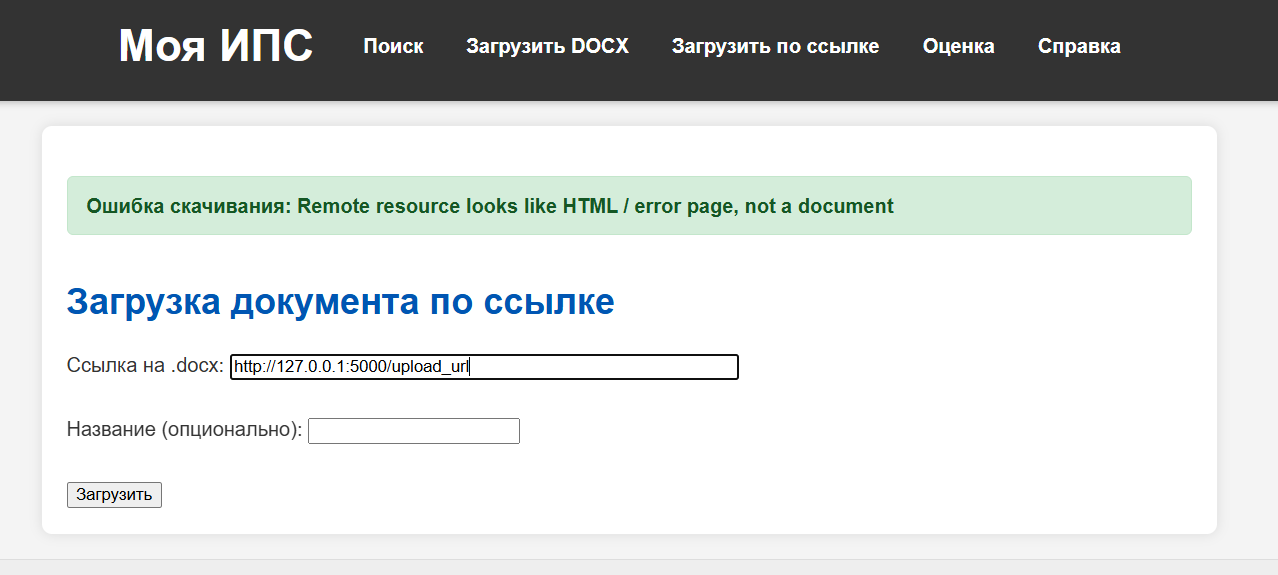
* Базовая архитектура: Meta Llama 3 8B Instruct.
* Язык: В первую очередь русский, с поддержкой английского.
* Тип модели: Чат-ботинструктивная, дообученная для ведения диалогов.
* Формат: GGUF (GPT-Generated Unified Format).
* Размер: Доступна в нескольких квантованных вариантах (например, Q4\_K\_M, Q5\_K\_M), что позволяет выбирать баланс между качеством, скоростью и объемом занимаемой памяти (обычно от 5 до 7 ГБ).

**Результаты оценки**

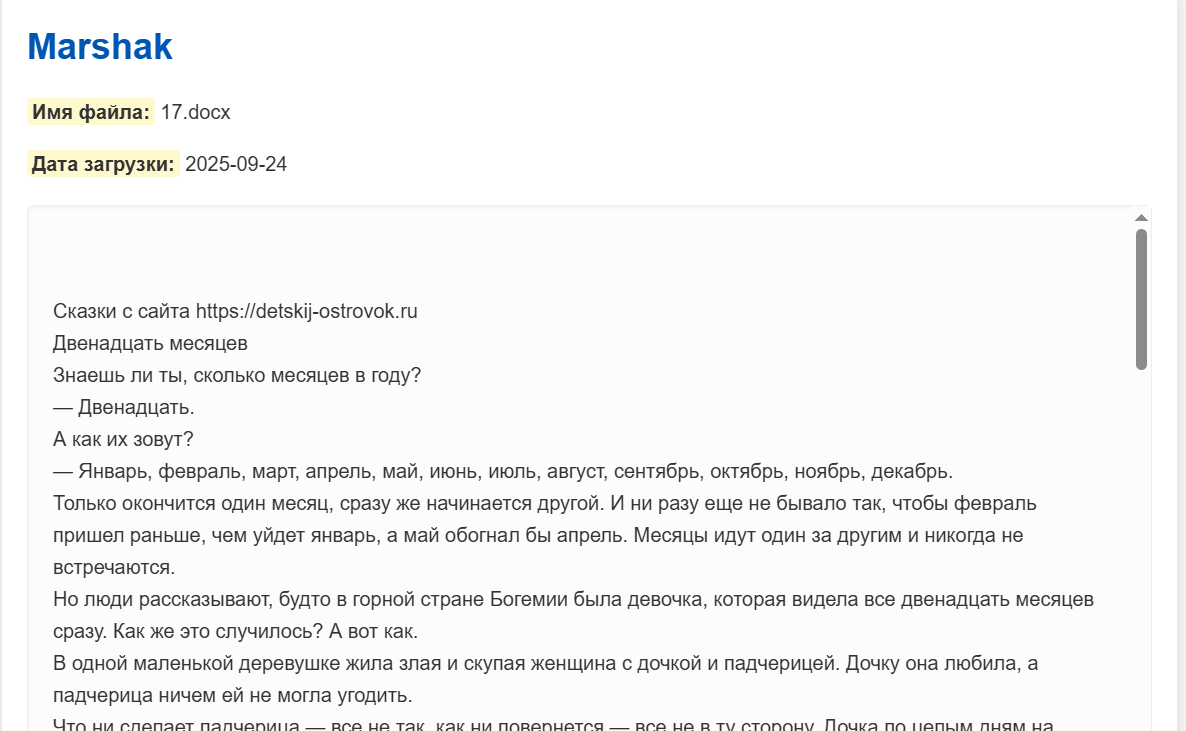




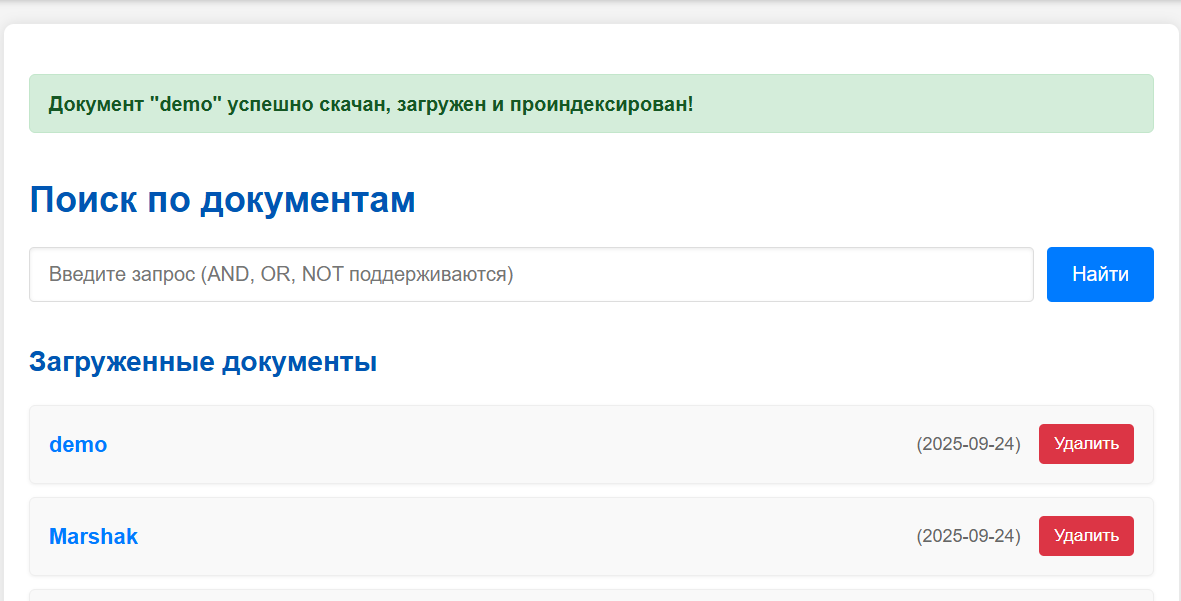
*рис. 2.1 - визуализация метрик*

**Демонстрация программы**

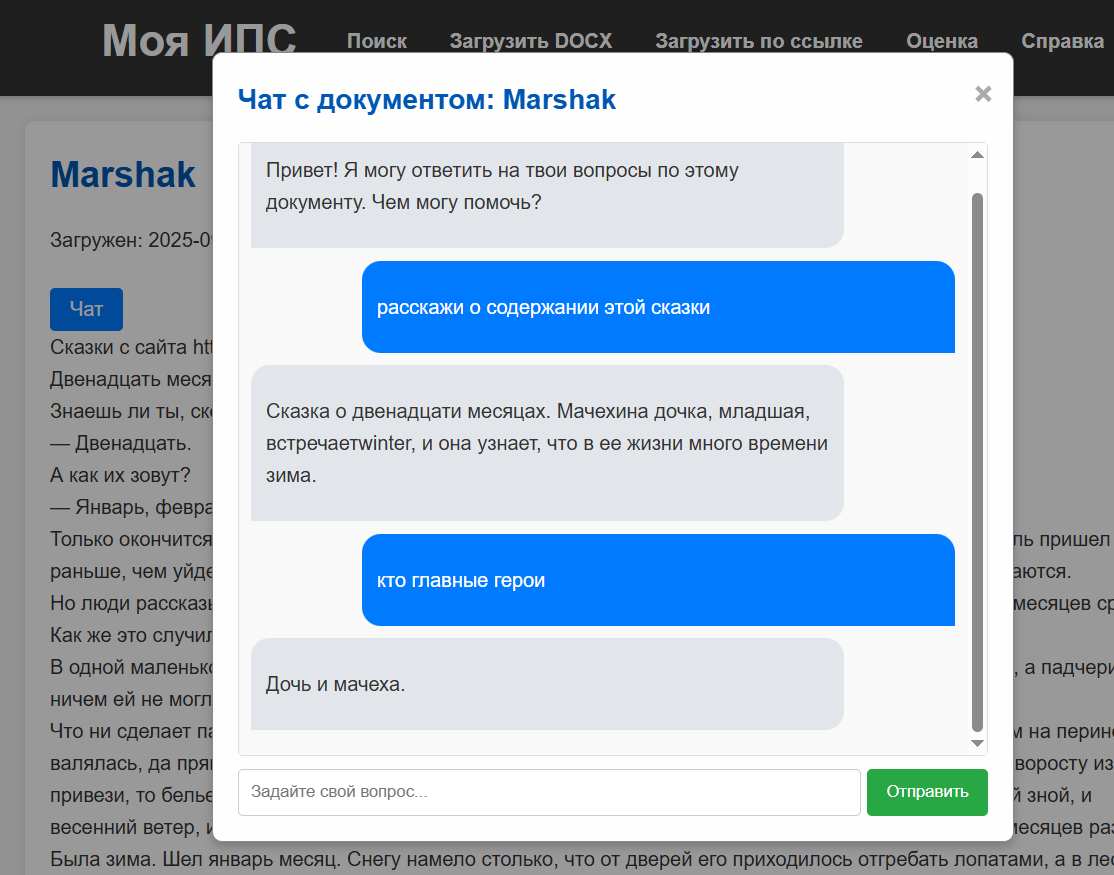
*рис. 3.1 - загрузка документа по ссылке*



*рис. 3.2 - пример загруженного по ссылке текста*



*рис. 3.3 - пример успешного скачивания файла*

**

*рис. 3.4 - окно чата*

**Выводы:**

В ходе выполнения лабораторной работы была успешно разработана и реализована полнофункциональная поисковая система с расширенными возможностями обработки текстовых данных и оценки качества поиска.

Основные достижения:

1. Создана комплексная архитектура поисковой системы, включающая:

* Модуль обработки и индексации документов
* Поисковый движок с поддержкой логических операторов
* Систему оценки качества с визуализацией результатов
* Веб-интерфейс для взаимодействия с пользователем
* Чат-интерфейс с документами на основе Ollama, обеспечивающий контекстно-зависимые ответы без "галлюцинаций"
* Безопасная система загрузки документов из внешних источников с комплексной валидацией

2. Реализована эффективная обработка текстовых данных с применением современных лингвистических инструментов:

* Лемматизация русского текста с помощью pymystem3
* Фильтрация стоп-слов и нормализация терминов

3. Разработана система оценки качества поиска, позволяющая:

* Автоматически рассчитывать метрики precision, recall и F-мера
* Визуализировать историю поисковых запросов
* Анализировать эффективность работы системы в динамике

Технологические преимущества:

Система демонстрирует эффективное использование современных технологий обработки естественного языка и машинного обучения. Применение логического поиска обеспечивает получение корректных результатов. Интеграция с Ollama добавляет элементы искусственного интеллекта в работу с документами.

Практическая значимость:

Разработанная система может быть применена в различных областях, требующих обработки текстовой информации:

* Корпоративные поисковые системы
* Образовательные платформы
* Юридические и научные базы документов
* Библиотечные каталоги и архивные системы

Лабораторная работа подтвердила эффективность выбранных алгоритмов и технологий для построения поисковых систем, а также продемонстрировала возможности расширения базового функционала за счет интеграции с языковыми моделями и системами безопасной загрузки контента.