**Коммерческое предложение: Юридический AI-ассистент для Telegram**

**1. Описание проекта**

**1.1. Цель проекта**

Создание высокоинтеллектуального Telegram-бота «ЮрПомощник РБ» — AI-ассистента, специализирующегося на законодательстве Республики Беларусь.

Цель бота — предоставлять пользователям быстрые и релевантныe ответы на юридические вопросы, используя продвинутую архитектуру **RAG (Retrieval-Augmented Generation)**. Бот способен самостоятельно пополнять свою базу знаний, анализировать вопросы на релевантность и взаимодействовать с администратором через многофункциональную веб-панель.

**1.2. Используемые технологии**

* **Язык программирования:** Python 3.10
* **Telegram API:** aiogram
* **AI и обработка языка:**
  + **Языковая модель:** OpenAI (gpt-4o-mini)
  + **Векторная база данных:** ChromaDB
  + **ML-классификация:** scikit-learn
* **Обработка данных:** PyMuPDF (для .pdf), python-docx (для .docx), pandas, numpy
* **Веб-панель администратора:** Flask, Flask-SocketIO
* **Развертывание:** Docker, Docker Compose

**1.3. Среда для развертывания**

Проект полностью контейнеризирован с помощью Docker, что обеспечивает его легкое и стабильное развертывание на любом сервере (облачном или физическом), поддерживающем Docker.

* **Сервер:** Облачный сервер (например, DigitalOcean, AWS, Heroku) или локальный сервер.
* **База данных:** Векторная база данных ChromaDB разворачивается локально на сервере вместе с основным приложением.

**2. Основные функции бота**

**2.1. Функциональные требования**

**2.1.1. Основные функции (для пользователя)**

* **Ответы на юридические вопросы:** Бот предоставляет структурированные и обоснованные ответы на вопросы пользователей, касающиеся законодательства РБ.
* **Интеллектуальный конвейер обработки вопроса:**
  1. **ML-фильтрация:** Автоматически определяет, является ли вопрос юридическим, отсеивая офф-топ.
  2. **Поиск по базе знаний:** Осуществляет семантический поиск релевантной информации в своей локальной векторной базе данных.
  3. **Динамический поиск (самообучение):** При отсутствии информации в базе, бот способен выполнять поиск на внешних авторитетных источниках (pravo.by) и **автоматически пополнять свою базу знаний**.
  4. **Генерация ответа:** Формирует финальный ответ с помощью большой языковой модели (LLM), основываясь на найденном контексте.
* **Предоставление справки:** Команда /help выводит подробную инструкцию по использованию бота.

**2.1.2. Административные функции**

* **Управление через Telegram:** Администратор может управлять ботом напрямую из чата с помощью команд:
  + /scrape: Запуск полного скрапинга сайта для пополнения базы знаний.
  + /update: Запуск "умного" инкрементального обновления (парсинг только новых и измененных страниц).
  + /stats, /analytics, /dynamic: Получение статистики по базе знаний, работе ML-фильтра и динамического поиска.
  + /startadmin, /stopadmin: Удаленный запуск и остановка веб-панели администратора.
* **Многофункциональная веб-панель:** Защищенная паролем веб-панель предоставляет полный контроль над системой:
  + **Дашборд:** Мониторинг состояния системы (ЦП, память) в реальном времени.
  + **Управление базой знаний:** Просмотр статистики, запуск скриптов по наполнению и обновлению.
  + **Просмотр логов:** Удобный интерфейс для чтения логов работы бота и админ-панели.
  + **Мониторинг фоновых задач:** Отслеживание статуса длительных процессов (например, скрапинга) через WebSocket.

**2.2. Ключевые технические детали**

**2.2.1. Архитектура**

Проект имеет современную микросервисную архитектуру, оркестрируемую через docker-compose:

* **Сервис** bot**:** Основной Telegram-бот, отвечающий за взаимодействие с пользователями.
* **Сервис** admin\_panel**:** Веб-приложение на Flask, предоставляющее административный интерфейс.
* **Общие** volumes**:** Оба сервиса используют общие тома для данных (/data, /db, /models), что обеспечивает целостность и консистентность базы знаний.

**2.2.2. Структура базы данных**

Используется векторная база данных **ChromaDB**. Данные хранятся в виде "чанков" (небольших фрагментов текста) с обширными метаданными, включающими:

* source: Источник документа (веб-скрапер, загрузка файла).
* url/filename: Ссылка на источник или имя файла.
* title: Заголовок документа.
* added\_date: Дата добавления.
* legal\_score: Оценка релевантности от фильтра контента.

**3. Структура проекта**

legal-bot/

├── data/ # Данные для наполнения базы знаний

│ └── documents/

├── db/ # Локальное хранилище векторной БД

│ └── chroma/

├── logs/ # Файлы логов

├── models/ # Сохраненные ML-модели

├── modules/ # Основные модули приложения

│ ├── \_\_init\_\_.py

│ ├── bot\_handler.py # Ядро бота, обработка команд

│ ├── knowledge\_base.py # Работа с ChromaDB

│ ├── llm\_service.py # Взаимодействие с OpenAI

│ ├── text\_processing.py # Обработка текста, PDF, DOCX

│ ├── web\_scraper.py # Сбор данных с сайтов

│ ├── incremental\_scraper.py # "Умное" обновление данных

│ ├── ml\_question\_filter.py # Фильтр вопросов на основе ML

│ └── ... # Другие вспомогательные модули

├── scripts/ # Вспомогательные скрипты

│ ├── populate\_db.py

│ ├── scrape\_websites.py

│ └── update\_documents.py

├── static/ # Статические файлы для админ-панели (CSS)

│ └── admin.css

├── templates/ # HTML-шаблоны для админ-панели

│ └── admin/

│ ├── index.html

│ └── login.html

├── tests/ # Автоматизированные тесты

│ ├── test\_config.py

│ └── test\_web\_scraper.py

├── .env.example # Пример файла с переменными окружения

├── .gitignore # Файл для исключения из Git

├── admin\_panel.py # Запуск веб-панели администратора

├── config.py # Конфигурация проекта

├── docker-compose.yml # Оркестрация запуска сервисов

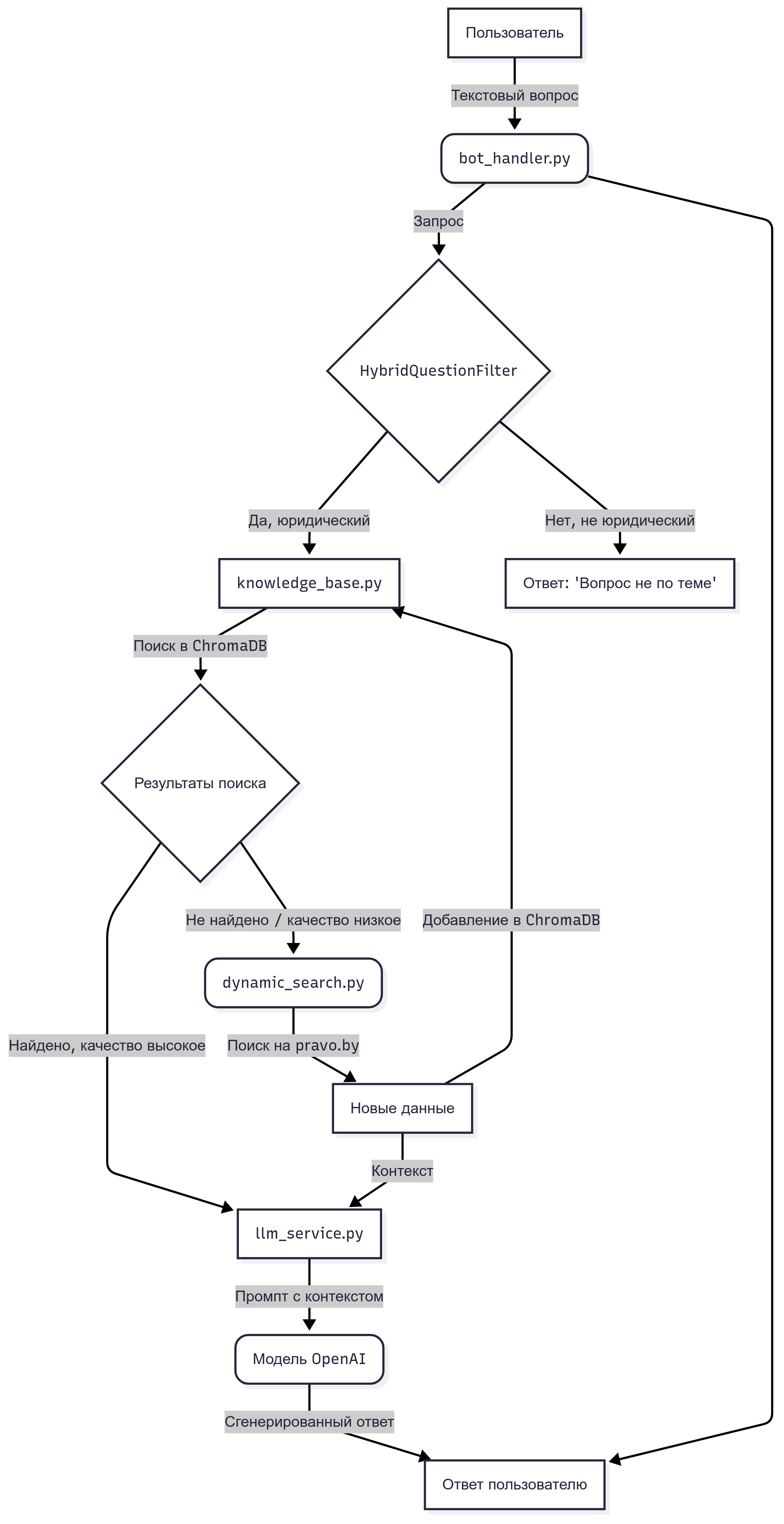
├── Dockerfile # Инструкции по сборке Docker-образа

├── main.py # Главная точка входа для запуска бота

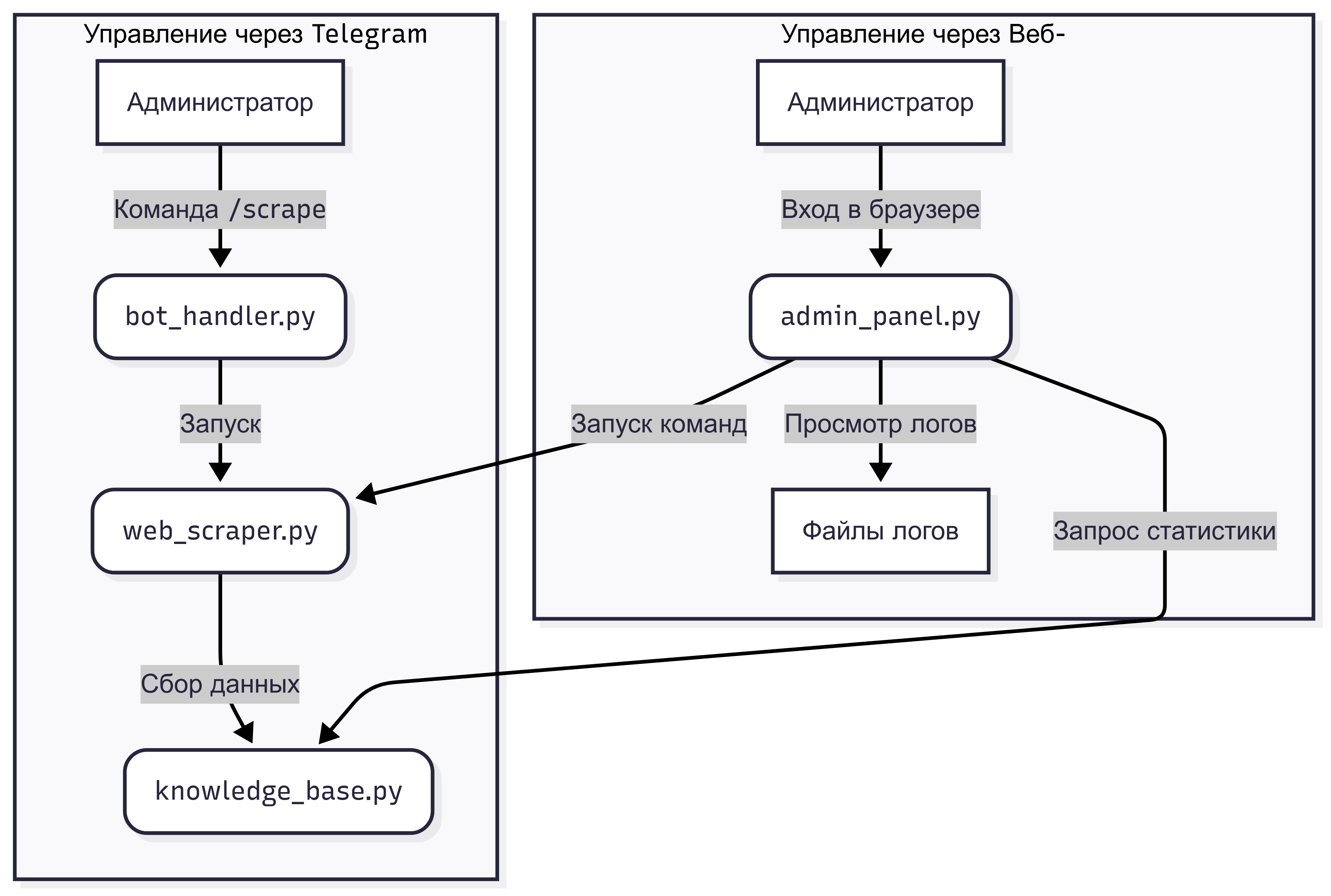
└── requirements.txt # Список зависимостей

**4. Графическое представление логики работы**

**4.1. Конвейер ответа на вопрос пользователя**



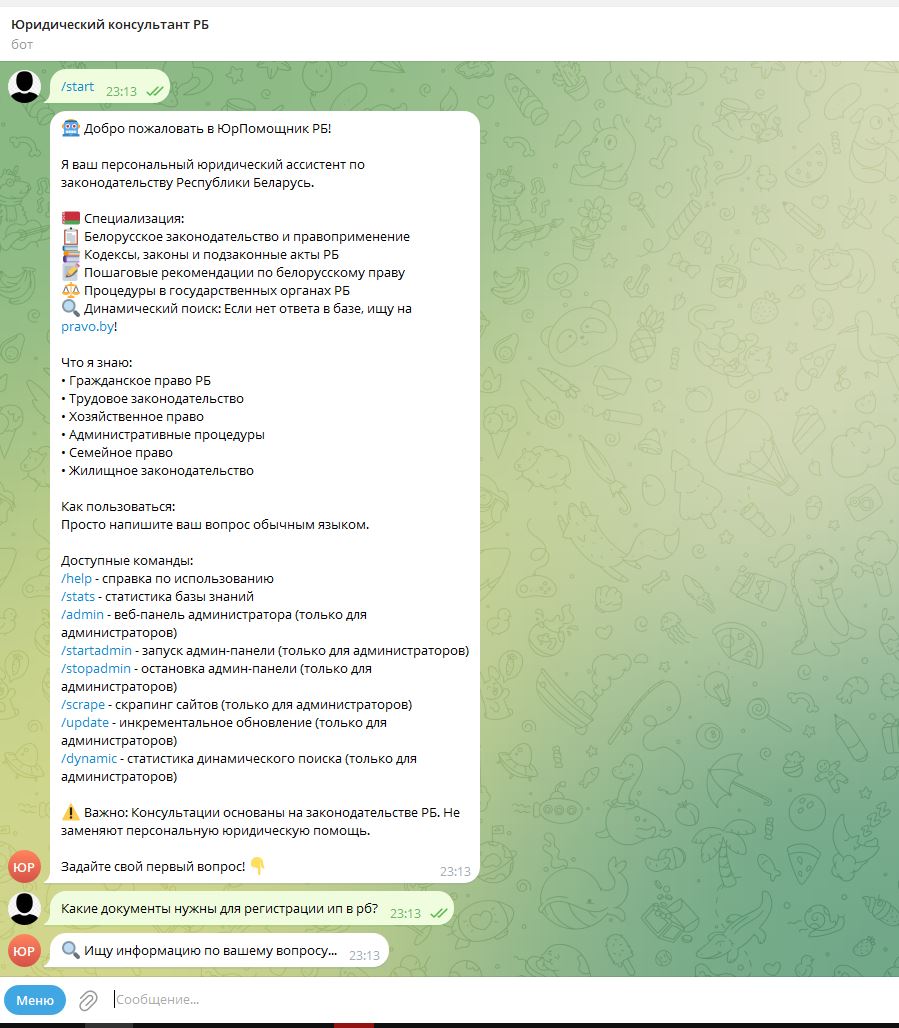
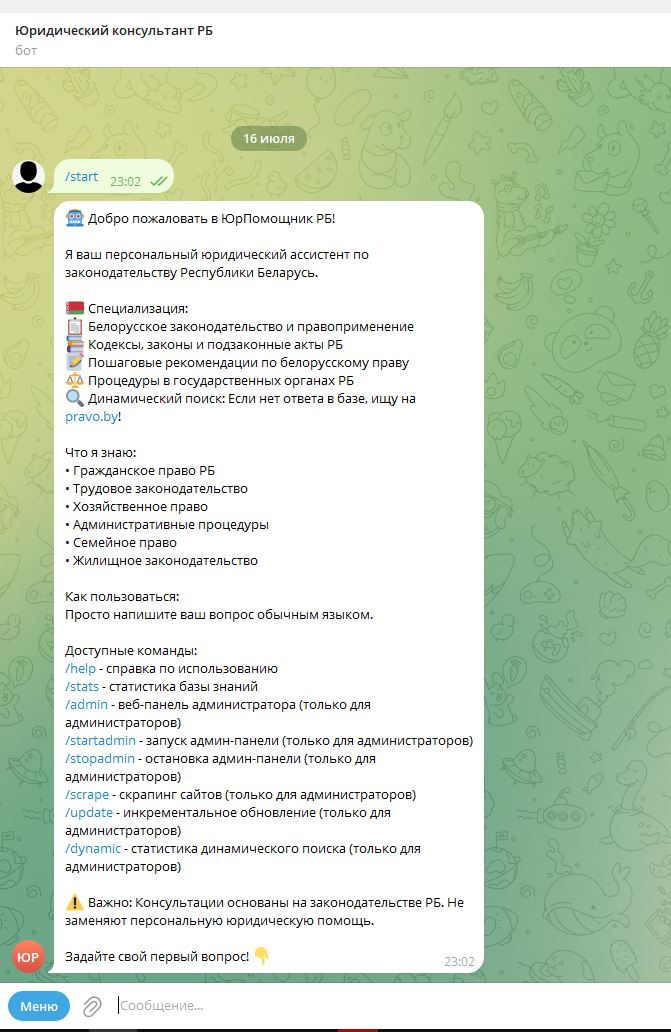
**4.2. Работа администратора**



**5. Презентация проекта (Скриншоты)**

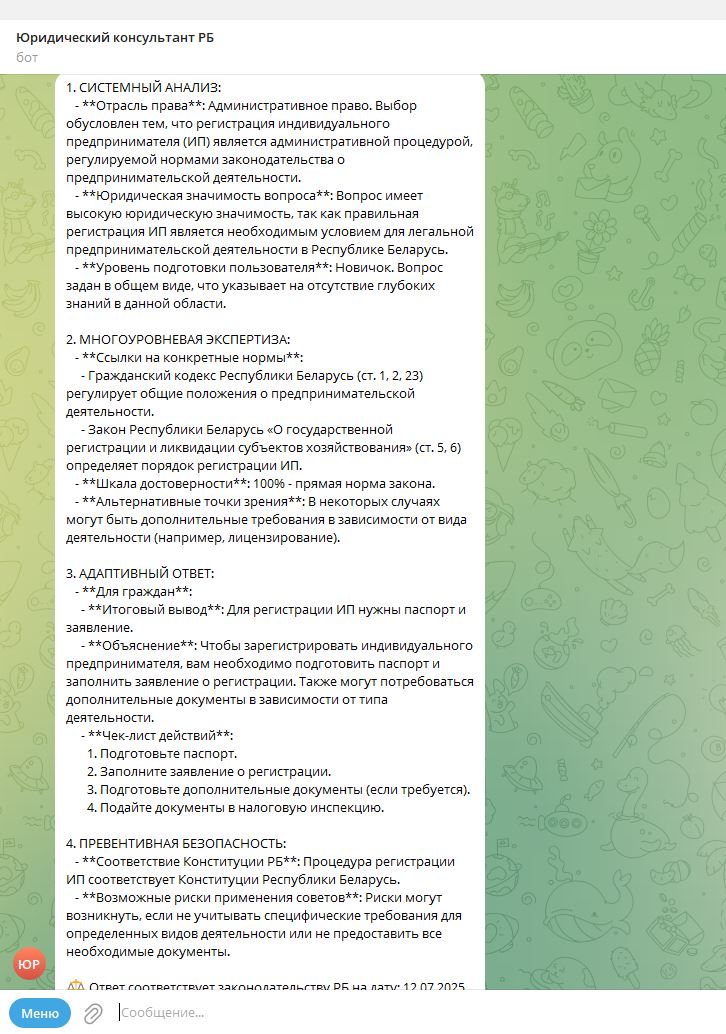
**5.1. Взаимодействие с пользователем**

1. **Начало работы (**/start**):** Пользователь запускает бота и получает подробное приветственное сообщение с описанием возможностей.
2. **Запрос пользователя:** Пользователь задает вопрос в свободной форме, например: "Какие документы нужны для регистрации ип в рб?".

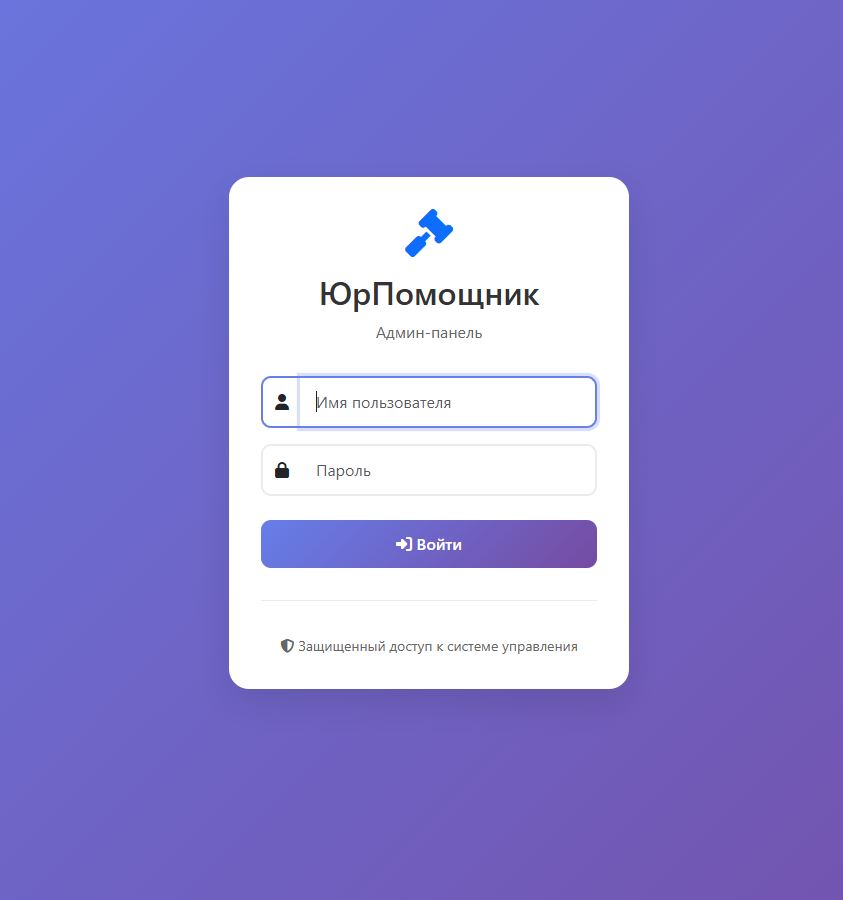


1. **Ответ бота:** Бот, следуя своей методологии, предоставляет структурированный ответ, включающий ссылки на нормы закона, оценку достоверности и обязательный дисклеймер.

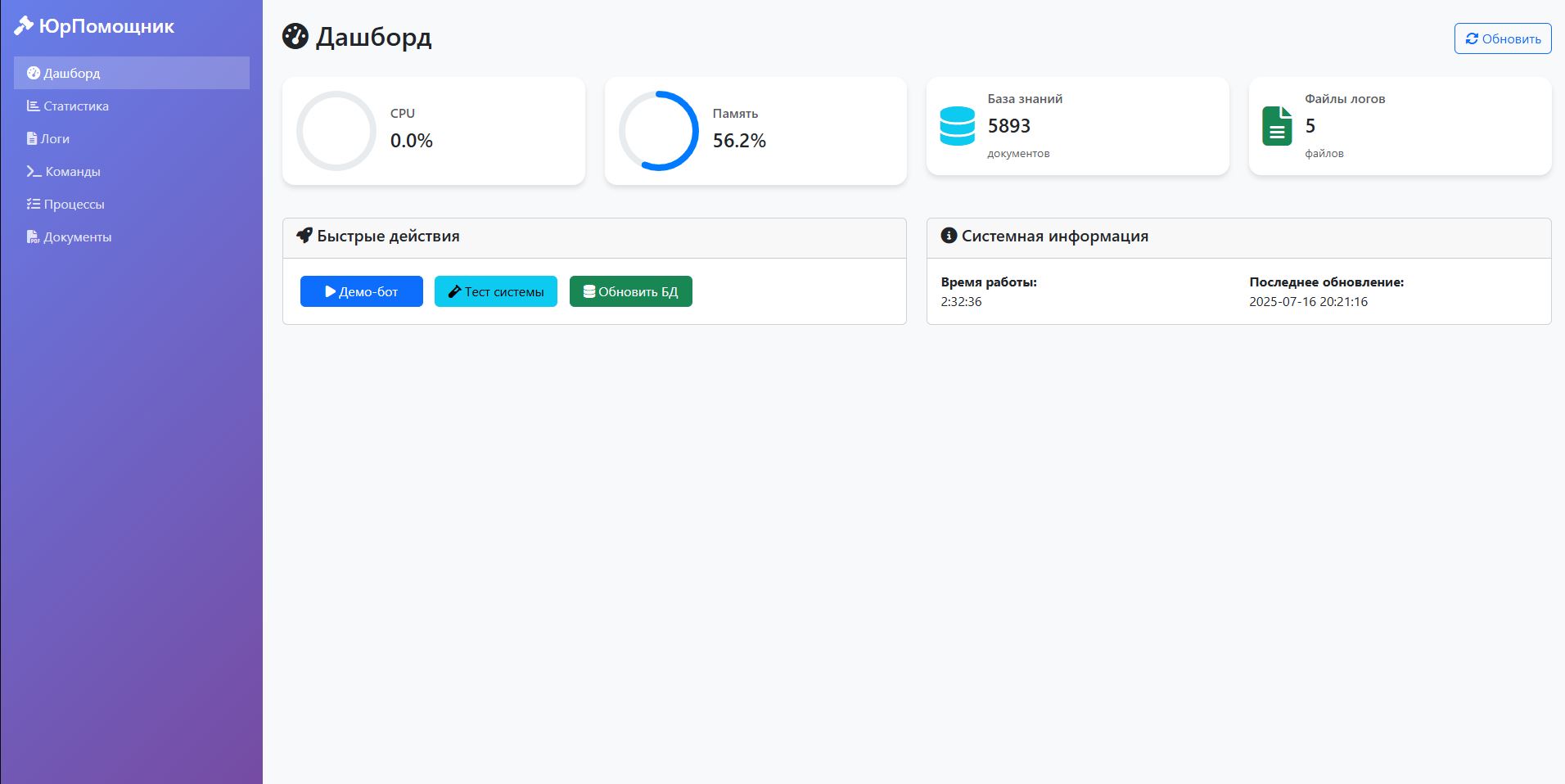
**5.2. Панель администратора**



1. **Вход в админ-панель:** Администратор переходит по ссылке, полученной от бота по команде /admin, и вводит логин/пароль.



1. **Главный дашборд:** После входа администратор видит главный экран с системной статистикой, состоянием базы знаний и запущенными процессами.



1. **Управление командами:** Администратор может запускать длительные задачи (например, скрапинг сайта) прямо из веб-интерфейса.

