****

**Отдел сервисов электронного взаимодействия**

**Готовое решение чат-бот «ИнфоКомпас»**

авторы: Ермилов В.В.

Файбисович В.А.

Москва 2024

**Оглавление.**

[**Введение.** 4](#_Toc179462192)

[**I. Основные компоненты и функциональность, установка.** 5](#_Toc179462193)

[**1. Функционал бота** 5](#_Toc179462194)

[**2. Основные функции файла app.py** 7](#_Toc179462195)

[**3. Описание функций JavaScript (JS) в файле index.html** 10](#_Toc179462196)

[**II. База данных и её поддержка** 16](#_Toc179462197)

[**1. База данных** 16](#_Toc179462198)

[**2. Поддержка** 17](#_Toc179462199)

[**III. Безопасность и стоимость проекта** 18](#_Toc179462200)

[**1.** **Варианты безопасности проекта** 19](#_Toc179462201)

[**2. Стоимость проекта** 19](#_Toc179462202)

[**Заключение** 20](#_Toc179462203)

**Термины и сокращения.**

|  |  |
| --- | --- |
| ЛК УИО | Личный кабинет участника информационного обмена |
| NLP | Natural Language Processing — обработка естественного языка. |
| РП | Руководство пользователя |
| TF-IDF | метод для оценки важности слова в контексте документа |
| BM25 (Best Matching 25) | улучшенная версия TF-IDF, которая учитывает длину документа и частоту слова в нем |
| Эмбеддинг | векторное представление слов или предложений в многомерном пространстве |
| БД | База данных |
| ВМ | Виртуальная машина |
| УИО | участник информационного обмена |

# **Введение.**

Для лучшего взаимодействия участников финансового рынка с ЛК УИО, а также улучшения пользовательского опыта работы с РП. Нами был создан чат-бот «ИнфоКомпас». Бот функционирует по адресу <https://инфокомпас.рф/> . Также к чат боту прилагается онлайн дашборд основанный на модуле Plotly Python, дашборд доступен по адресу [http:176.109.109.61:8080/dashboard](http://176.109.109.61:8080/dashboard).

Чат-бот выполняет роль виртуального помощника, который может:

* Отвечать на вопросы пользователей.
* Помогать ориентироваться в РП.
* Предоставлять ссылки на внутренние и внешние ресурсы.

Бот создан с использований таких технологий как:

* RuBERT Tiny2 - это уменьшенная версия модели BERT, специально адаптированная для русского языка. Она используется для обработки естественного языка (NLP).
* TF-IDF — это метод для оценки важности слова в контексте документа. Он состоит из двух частей:

1. TF (Term Frequency): Сколько раз слово встречается в документе. Чем чаще слово встречается, тем важнее оно.
2. IDF (Inverse Document Frequency): Насколько редко слово встречается во всех документах. Чем реже слово встречается, тем важнее оно.

* BM25 (Best Matching 25) - улучшенная версия TF-IDF, которая учитывает длину документа и частоту слова в нем. Она помогает лучше определить, насколько релевантен документ для заданного запроса.
* Алгоритм Левенштейна — это метод для измерения разницы между двумя строками (например, словами или предложениями). Он считает количество операций (вставка, удаление, замена символов), необходимых для превращения одной строки в другую.

**I. Основные компоненты и функциональность, установка.**

Установка проекта происходит по средствам клонирования репозитория GitHub с помощью команды:

1. git clone <https://github.com/Vikkingsk8/InfoCompas_2.0.git> (репозиторий приватный)

2. далее прейти в папку проекта:

cd InfoCompas\_2.0

3. установить зависимости:

pip install –r requirements.txt

возможно потребуется какие-то пакеты локально обновить либо установить не достающие модули, в логах будет указано что именно нужно сделать.

Далее только потребуется запустить главный файл app.py:

python app.py – для windows

python3 app.py – для linux

## **1. Функционал бота**

Общий пример работы бота можно увидеть на рисунке 1(рис.1).



рис. 1 основная схема работы бота

Когда бот получает вопрос, после проверки на валидность. Бот в БД ищет наиболее схожий вопрос и дает ответ пользователю включая релевантные кнопки (рис.2):

* для внутреннего взаимодействия с ботом (имеют синий цвет)
* ссылки для взаимодействия с РП (имеют темно-синий цвет)
* ссылки для перехода на внешние источники (имеют фиолетовый цвет)

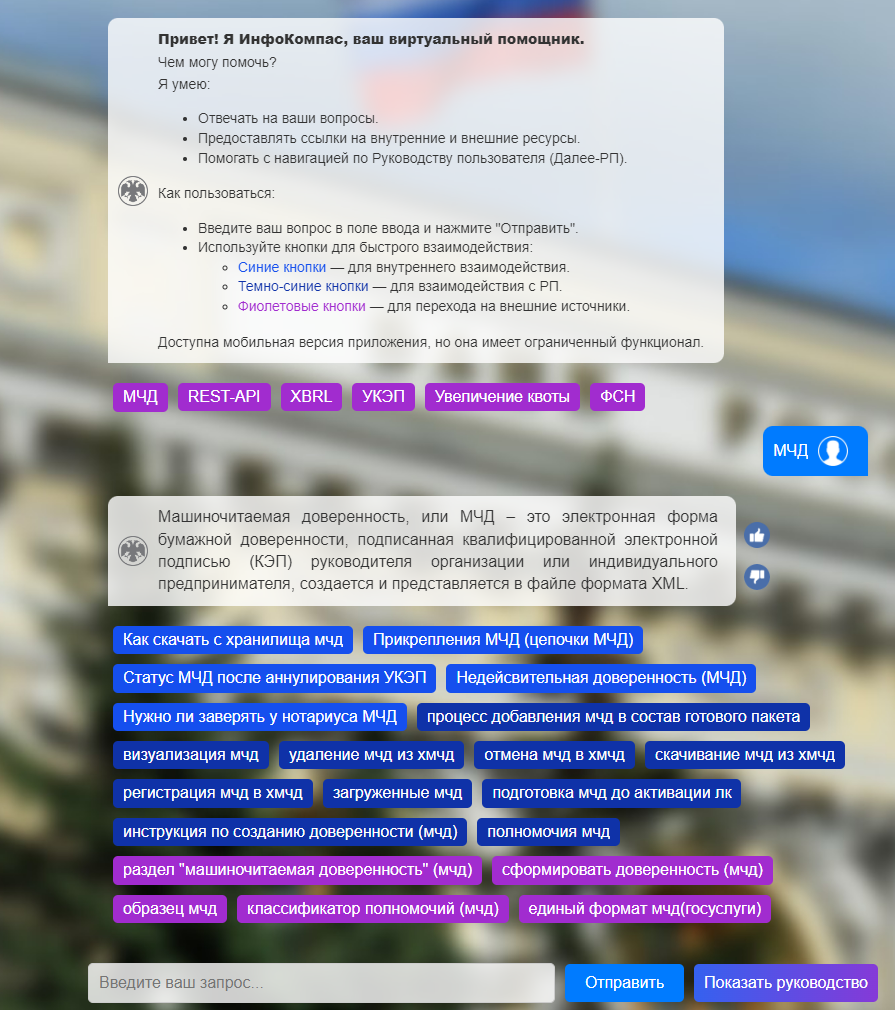


рис. 2 диалог с ботом

## **2. Основные функции файла app.py**

Файл app.py в проекте «ИнфоКомпас» является основным файлом, через который запускается бот. Основной принцип работы файла app.py (рис.3).



рис. 3 принцип работы app.py

Список и описание функций файла app.py:

**1. Загрузка и обработка данных**

**1.1. Загрузка данных из Excel**

Функция load\_excel\_data(path) загружает данные из Excel-файла, который содержит вопросы и ответы. Эти данные используются для поиска ответов на основе введенных пользователем вопросов.

**1.2. Предварительная обработка данных**

Функция preprocess\_excel\_data(df, column\_name) предварительно обрабатывает данные из Excel, разделяя текст вопросов на отдельные вопросы и приводя их к нижнему регистру.

**2. Обработка текста**

**2.1. Получение эмбеддингов**

Функция get\_embedding(text) использует модель AutoModel из библиотеки transformers для получения эмбеддингов текста. Эти эмбеддинги используются для вычисления сходства между введенными пользователем вопросами и текстом в документах.

**2.2. Создание TF-IDF матрицы**

Функция find\_relevant\_links(user\_question, threshold) создает TF-IDF матрицу на основе вопросов из файла `links.xlsx`. Эта матрица используется для вычисления сходства между введенными пользователем вопросами и вопросами в файле `links.xlsx`.

**3. Поиск ответов**

**3.1. Поиск наилучшего ответа**

Функция find\_best\_answer(query\_embedding, df) находит наиболее релевантн-ый ответ на основе введенного пользователем вопроса, используя эмбеддинги и DataFrame с вопросами и ответами.

**3.2. Поиск релевантных ссылок**

Функция find\_relevant\_links(user\_question, threshold) находит релевантные ссылки на основе введенного пользователем вопроса и порогового значения сходства.

**4. Обработка вопросов и ссылок**

**4.1. Предварительная обработка текста**

Функция preprocess\_user\_question(question) обрабатывает текст вопросов, приводя его к нижнему регистру и удаляя лишние пробелы.

**4.2. Поиск релевантных ссылок**

Функция find\_relevant\_fork\_links(user\_question, threshold) находит релевантн-ые ссылки на основе введенного пользователем вопроса и порогового значения сходства.

**5. Функциональность кнопок ссылок**

**5.1. Отправка вопроса и открытие ссылки**

При нажатии на кнопку ссылки, вопрос, связанный с этой ссылкой, отправляется на сервер для поиска ответа, и ссылка открывается в новом окне. Это позволяет пользователю получить дополнительную информацию, связанную с их запросом.

**6. Обратная связь**

**6.1. Сохранение и загрузка данных о лайках и дизлайках**

Функция save\_feedback(question, answer, feedback) сохраняет данные о лайках и дизлайках в CSV-файл. Функция load\_feedback() загружает эти данные для дальнейшего использования.

**7. Маршруты и эндпоинты**

**7.1. Главная страница**

Маршрут `/` отвечает за отображение главной страницы с интерфейсом для ввода вопросов.

**7.2. Эндпоинт для чата**

Маршрут `/chat` обрабатывает POST-запросы с вопросами от пользователей и возвращает наиболее релевантный ответ.

**7.3. Эндпоинт для загрузки PDF**

Маршрут `/download\_pdf` позволяет пользователям скачать PDF-документ.

**7.4. Эндпоинт для обратной связи**

Маршрут `/feedback` обрабатывает POST-запросы с данными о лайках и дизлайках от пользователей.

**7.5. Эндпоинт для аналитики**

Маршрут `/analytics\_data` возвращает аналитические данные о запросах пользователей.

## **3. Описание функций JavaScript (JS) в файле index.html**

Файл index.html находится в директории templates и отвечает за fronted (визуальная часть) веб приложения совместно с файлом styles.css.

Список и описание функций файла index.html:

**1. showWelcomeMessage()**

Описание:

Функция отображает приветственное сообщение и начальные вопросы при загрузке страницы.

Детали:

Создает приветственное сообщение и добавляет его в чат-бокс.

Создает кнопки с начальными вопросами и добавляет их в чат-бокс.

При нажатии на кнопку начального вопроса, текст вопроса вводится в поле ввода и отправляется.

**2. loadSuggestions()**

Описание:

Функция загружает предложения для автодополнения из сервера.

Детали:

Делает запрос к серверу для получения списка предложений.

Сохраняет полученные предложения в переменную suggestions.

**3. showSuggestion(query)**

Описание:

Функция отображает предложения для автодополнения на основе введенного текста.

Детали:

Ищет совпадения в списке предложений на основе введенного текста.

Отображает найденное предложение в виде кнопки.

При нажатии на кнопку предложения, текст предложения вводится в поле ввода.

**4. sendMessage(message = null)**

Описание:

Функция отправляет сообщение пользователя на сервер и отображает ответ.

Детали:

Получает текст сообщения из поля ввода или из аргумента функции.

Отправляет сообщение на сервер с помощью POST-запроса.

Отображает сообщение пользователя и ответ бота в чат-боксе.

Обрабатывает ссылки и кнопки, возвращаемые сервером.

Добавляет кнопки обратной связи для ответа бота.

**5. displayButtons(buttons)**

Описание:

Функция отображает кнопки в контейнере кнопок.

Детали:

Очищает контейнер кнопок.

Создает и добавляет кнопки в контейнер на основе переданного массива.

**6. clearButtonsContainer()**

Описание:

Функция очищает контейнер кнопок.

Детали:

Удаляет все содержимое контейнера кнопок.

**7. createLinkButton(link)**

Описание:

Функция создает кнопку для ссылки.

Детали:

Создает кнопку с текстом вопроса.

Добавляет классы в зависимости от типа ссылки (форк, PDF-страница, внешняя ссылка).

Обрабатывает нажатие на кнопку.

**8. handleLinkButtonClick(link)**

Описание:

Функция обрабатывает нажатие на кнопку ссылки.

Детали:

Если ссылка является форком, отправляет вопрос на сервер.

Если ссылка является PDF-страницей, открывает PDF и переходит к указанной странице.

Если ссылка является внешней, открывает ее в новой вкладке.

**9. isValidPageNumber(value)**

Описание:

Функция проверяет, является ли значение числом (номером страницы).

Детали:

Удаляет пробелы в начале и конце строки.

Проверяет, состоит ли строка только из цифр.

**10. createLinksContainer(links, questionText)**

Описание:

Функция создает контейнер с кнопками ссылок.

Детали:

Создает контейнер для кнопок.

Сортирует ссылки по типу (форки, PDF-страницы, внешние ссылки).

Добавляет кнопки в контейнер.

Добавляет кнопку "Показать больше", если есть скрытые ссылки.

**11. clearLinksContainer()**

Описание:

Функция очищает контейнер ссылок.

Детали:

Удаляет все содержимое контейнера ссылок.

**12. addFeedbackButtons(messageElement, question, answer)**

Описание:

Функция добавляет кнопки обратной связи к сообщению бота.

Детали:

Создает контейнер для кнопок обратной связи.

Создает кнопки "Нравится" и "Не нравится".

Добавляет кнопки в контейнер и контейнер в сообщение бота.

**13. createFeedbackButton(feedbackType, initialImage, activeImage)**

Описание:

Функция создает кнопку обратной связи.

Детали:

Создает кнопку с изображением.

Обрабатывает нажатие на кнопку, отправляя обратную связь на сервер.

**14. removeOldFeedbackButtons()**

Описание:

Функция удаляет старые кнопки обратной связи.

Детали:

Проходит по всем сообщениям бота и удаляет кнопки обратной связи, кроме последнего сообщения.

**15. sendLinkQuestion(question, url)**

Описание:

Функция отправляет вопрос, связанный с ссылкой, на сервер.

Детали:

Отправляет вопрос на сервер.

Обрабатывает ответ сервера, отображая его в чат-боксе.

Обрабатывает ссылки и кнопки, возвращаемые сервером.

**16. prependMessage(message, className, icon, images = [], pdfPage = null)**

Описание:

Функция добавляет сообщение в начало чат-бокса.

Детали:

Создает элемент сообщения с указанным классом и иконкой.

Добавляет текст сообщения и ссылку на PDF-страницу, если указана.

Добавляет сообщение в начало чат-бокса.

**17. openPdfAndGoToPage(pageNumber)**

Описание:

Функция открывает PDF и переходит к указанной странице.

Детали:

Открывает PDF-контейнер, если он скрыт.

Создает новый iframe для отображения PDF.

Переходит к указанной странице в PDF.

**18. isGreeting(userMessage)**

Описание:

Функция проверяет, является ли сообщение приветствием.

Детали:

Проверяет, содержит ли сообщение одно из приветственных слов.

**19. scrollToBottom()**

Описание:

Функция прокручивает чат-бокс вниз.

Детали:

Прокручивает чат-бокс до самого нижнего сообщения.

**20. sendFeedback(messageElement, feedbackType)**

Описание:

Функция отправляет обратную связь на сервер.

Детали:

Отправляет обратную связь на сервер с помощью POST-запроса.

Устанавливает атрибут data-feedback-sent для сообщения, чтобы предотвратить повторную отправку.

**21. isMobileDevice()**

Описание:

Функция проверяет, используется ли мобильное устройство.

Детали:

Проверяет строку userAgent на наличие признаков мобильного устройства.

**22. isTelegramInAppBrowser()**

Описание:

Функция проверяет, используется ли браузер Telegram.

Детали:

Проверяет строку userAgent на наличие признаков браузера Telegram.

**23. Обработчики событий**

Описание:

Обработчики событий для кнопок и поля ввода.

Детали:

Обработчик для кнопки "Отправить": вызывает функцию sendMessage().

Обработчик для поля ввода: вызывает функцию sendMessage() при нажатии клавиши Enter.

Обработчик для кнопки "Показать руководство": переключает отображение PDF-контейнера.

# **II. База данных и её поддержка**

Поскольку бот реализован на технологиях NLP и не обучен на каких-то конкретных документах, а также не имеет возможности генерировать текст. База данных находится в директории data и содержит в себе:

* ответы.xlsx
* links.xlsx
* развилка.xlsx
* instruction.pdf
* feedback.xlsx
* dashboard\_data.db

**1. База данных**

Пример главного файла с вопросами и ответами (рис.4). С помощью такого

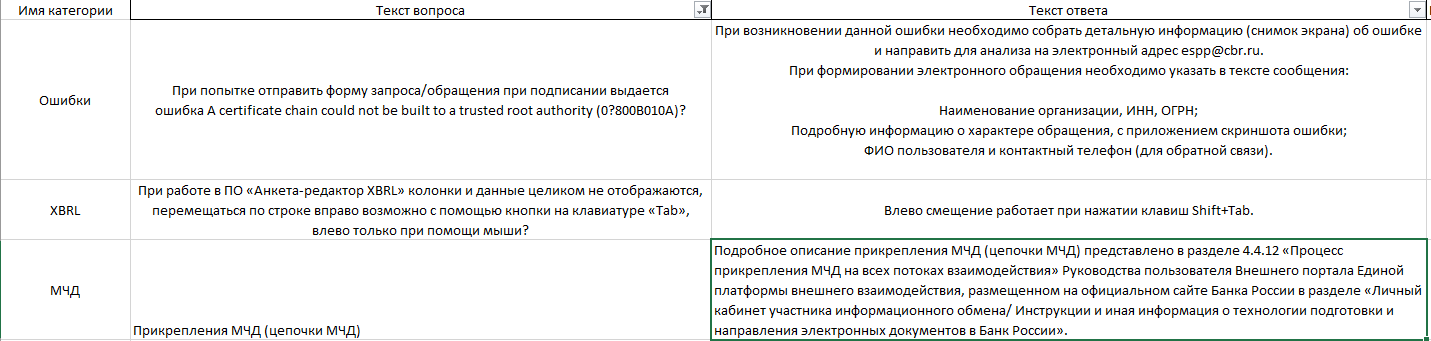
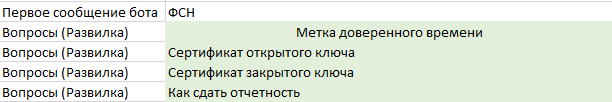
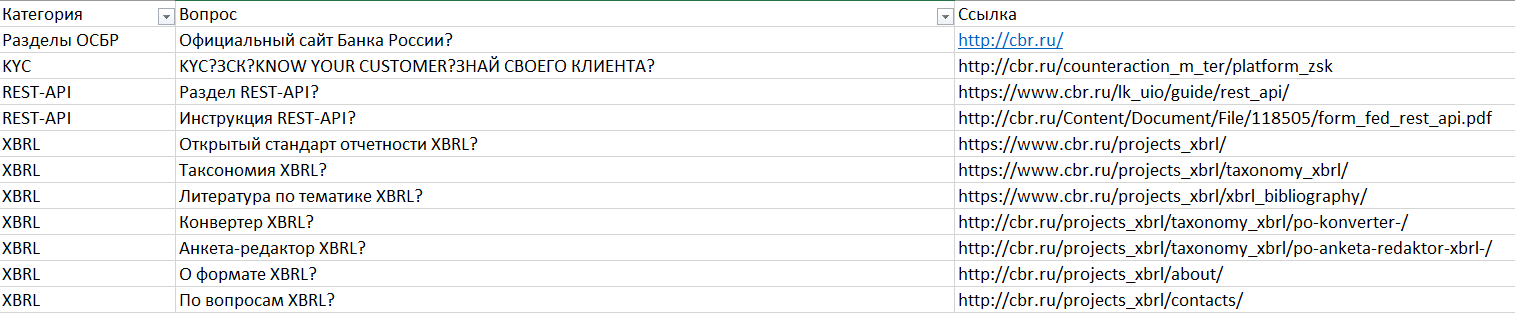


рис. 4 файл ответы.xlsx

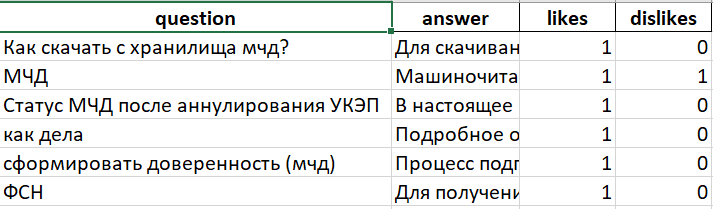
вида бд реализуется механизм поиска ответа бота по эмбеддингам. Бот находит наиболее схожий вопрос по вопросу пользователя и возвращает ответ с развилкой(рис.5) и ссылками, которые в свою очередь делятся на внешние и

 рис. 5 файл развилка.xlsx

на ссылки навигации РП. БД ссылок (рис.6).

 рис. 6 файл links.xlsx

Файл feedback.xlsx (рис. 7) содержит в себе информацию об обратной связи пользователей, а именно о лайках и дизлайках. В файл записывается вопрос, который задал пользователь, ответ, который он получил и реакция на полученный ответ. С помощью этого файла планируется отслеживать актуальность БД.

 рис. 7 файл feedback.xlsx

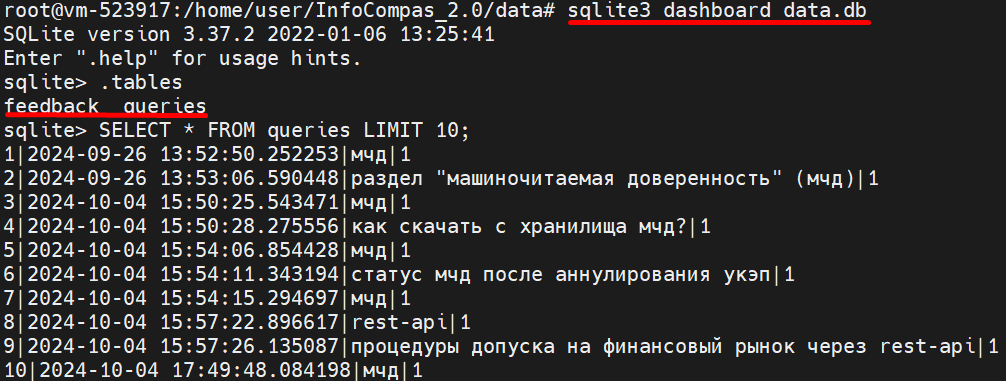
 Файл dashboard\_data.db (рис.8) содержит в себе две таблицы feedback и queries. В таблице feedback содержится абсолютно та же информация что и в файле feedback.xlsx. В таблице queries содержится: дата запроса, вопрос, состояние запросов, был ли он успешно возвращен или нет.

рис. 8 БД dashboard\_data.db

## **2. Поддержка**

Поддержка БД планируется осуществлять с помощью сотрудников отдела сервисов электронного взаимодействия. Особенность обновления будет заключатся в том, что сотруднику нужно будет отслеживать данные в файле feedback.xlsx, на основе этих данных будет происходить определение актуальности данных в файле ответы.xlsx. Также регулярно нужно отслеживать актуальность РП и самостоятельно обновлять pdf файл, скачивая его с официального сайта cbr.ru.

# **III. Безопасность и стоимость проекта**

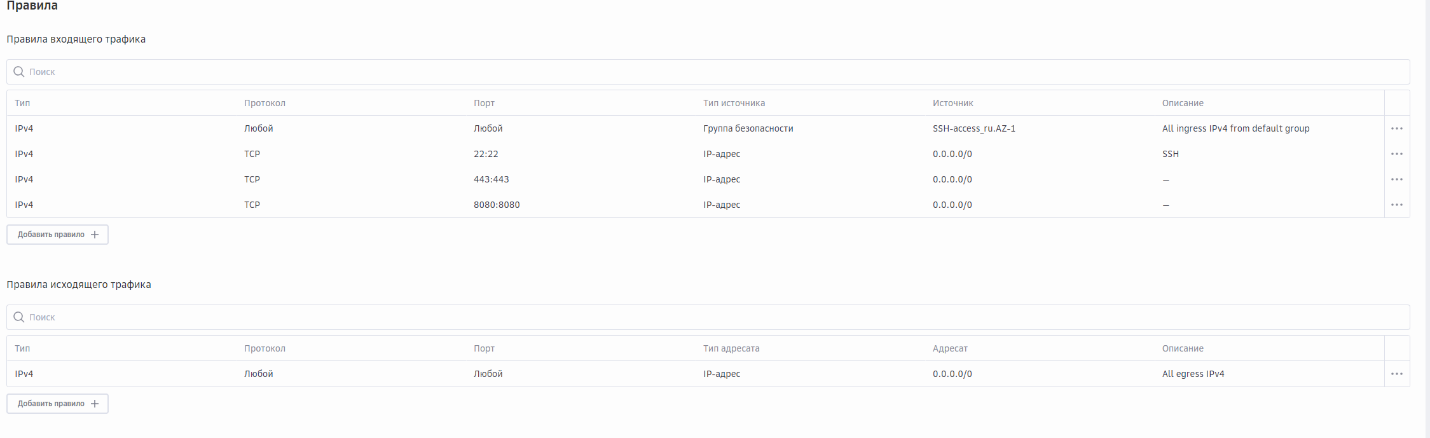
 На данный момент безопасность проекта достигается при помощи групп безопасности (рис. 9) и высокой отказоустойчивости. В группах прописаны только защищенное соединение по порту 443, также есть не защищённое соединение по порту 8080 которое служит для теста.

рис. 9 группы безопасности

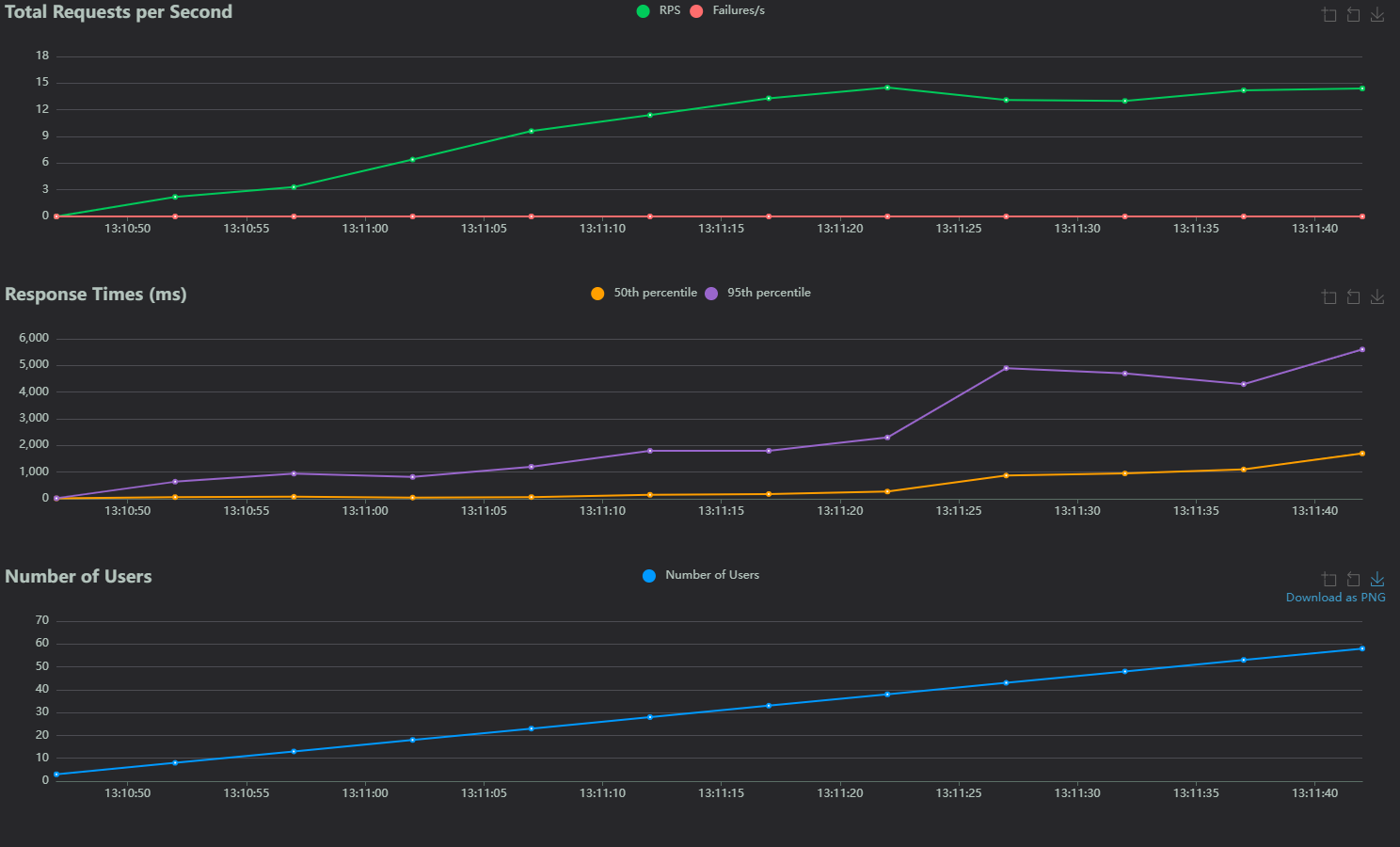
 Отказоустойчивость была достигнута за счет оптимизированного алгоритма поиска ответа. Был проведен тест нагрузки (рис.10) приложения во время которого 100 пользователей одновременно делали запросы. При этом отклик от бота, а именно ответа для 101 пользователя составлял 10 секунд. Тест был проведен с помощью сервера Locust.

рис. 10 тест отказоустойчивости

1. **Варианты безопасности проекта**

Для обеспечения оптимальной безопасности предлагается арендовать еще одну ВМ и использовать ее как прокси сервер балансировщик, при помощи утилиты linux squid.

Второй вариант сделать whitelist прописать туда айпи адреса только УИО и сотрудников которые будут работать с ботом. Это обеспечит 100% безопасность.

## **2. Стоимость проекта**

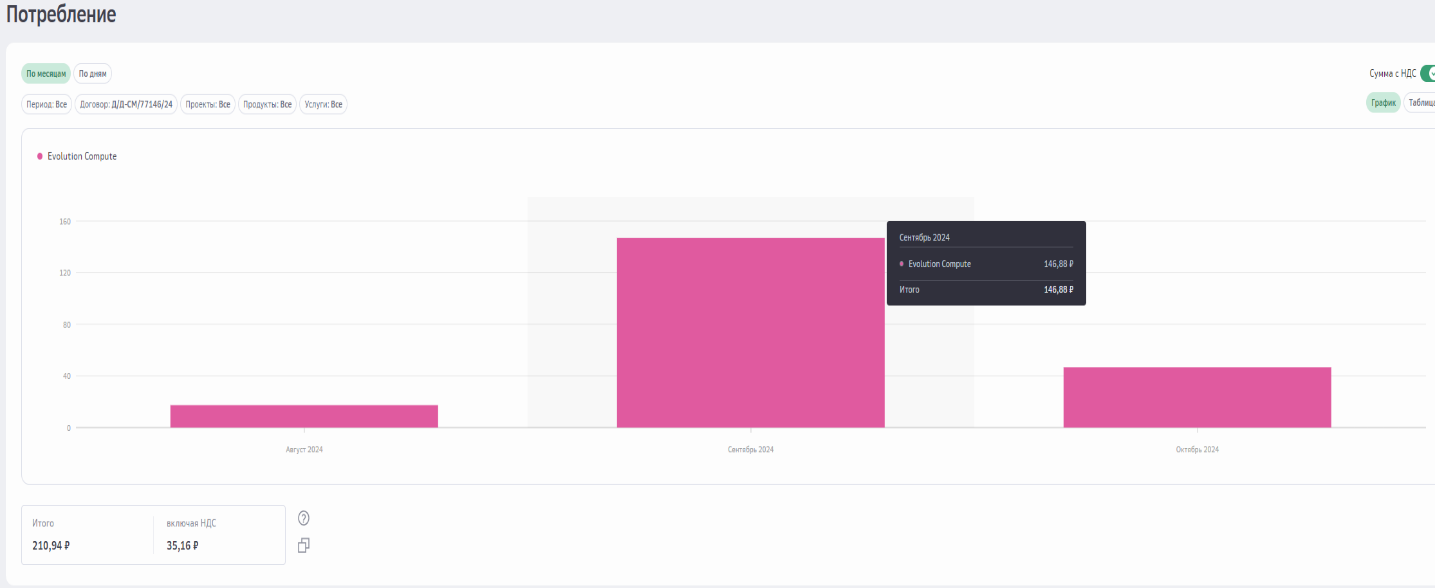
Помимо того, что бот максимально легковесный, развернутый проект на данный момент занимает всего 12 гигабайт. Он относительно не дорогой. На данный момент ВМ на Cloud.ru в месяц стоит 150 рублей (рис.11) + домен «инфокомпас.рф» с хостингом 276 рублей на три месяца. В месяц 242 рубля. Такая стоимость достигается оптимизированными алгоритмами бота, поэтому достаточно самой дешевой ВМ в которую включены 

рис. 11 стоимость ВМ

# **Заключение**

На данный момент проект необходимо протестировать чтобы получить обратную связь от пользователей по функционалу. Также в планах на будущее есть идея сделать бота не только для УИО, а для всего Банка России, чтобы любой сотрудник мог загрузить свой нормативный документ и бот помог ему в нем быстро разобраться. Реализовать это можно с помощью RAG подхода и нескольких предобученных ИИ моделей.

Уже сейчас прослеживается потенциал «конструктора», то есть любой отдел, который хочет себе такого бота. Может составить БД вопрос-ответ, поменять развилку на их вопросы, и составить новый файл links.xlsx с ссылками, которые относятся к отделу. И в итоге выйдет такой же бот только уже непосредственно под нужды других отделов.