Цинаридзе Даниил Сергович

Группа ИД 23.2

Тема:

«Телеграм-бот по подбору квартир»

**Введение**Мой проект - это телеграмм бот, который поможет вам быстро и легко найти идеальную квартиру. Никаких бесконечных поисков на сайтах объявлений и сотни звонков агентам недвижимости. Вам просто нужно отправить боту запрос с критериями, которые вам нужны, и он вернет вам все доступные объявления из собственной базы данных, которая автоматически пополняется объявлениями с помощью парсера, который мониторит все крупные площадки (Яндекс Недвижимость, Авито, Циан). Парсер отбирает только объявления от собственников, что избавит вас от переплаты комиссии за услуги агентств. Идея появилась после общения с друзьями риелторами. Думаю, если серьезно заняться этим продуктом, то он может быть полезен людям **Основная часть  
1.1 Список задач**+ Задача выполнена; - Задача не выполнена

* Парсер Авито
  + Получение JSON из мобильной версии сайта
  + Получение всех необходимых данных
  + Сохранение данных в словарь
  + Добавление словаря с данными в базу данных MySQL
  + Автоматизация с помощью бесконечного цикла и time.sleep()
  + Написание нового парсера на selenium после блокировки IP
* Парсер Циан
  + Получение JSON
  + Получение всех необходимых данных
  + Сохранение в словарь
  + Добавление словаря с данными в базу данных MySQL
  + Автоматизация с помощью бесконечного цикла и time.sleep()
* Парсер Яндекс Недвижимость
  + Получение JSON из мобильной версии сайта
  + Получение всех необходимых данных
  + Сохранение данных в словарь
  + Добавление словаря с данными в базу данных MySQL
  + Автоматизация с помощью бесконечного цикла и time.sleep()
* Создание бота в Telegram
  + Создание эхо-бота
  + Создание основной клавиатуры с фильтрами
  + Обработка запроса пользователя
  + Отправка объявлений пользователю
* Создание дополнительной клавиатуры
* Реализация функционала дополнительной клавиатуры
* Подключить прокси

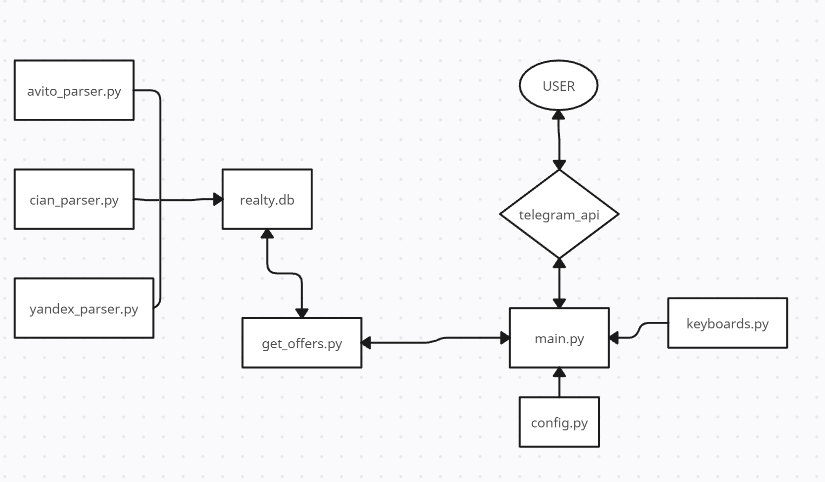
**1.2 Подключаемые модули**

**Requests** – создание HTTP запроса

**aiogram -** асинхронный фреймворк для Telegram Bot API

**selenium –** автоматизация работы браузера

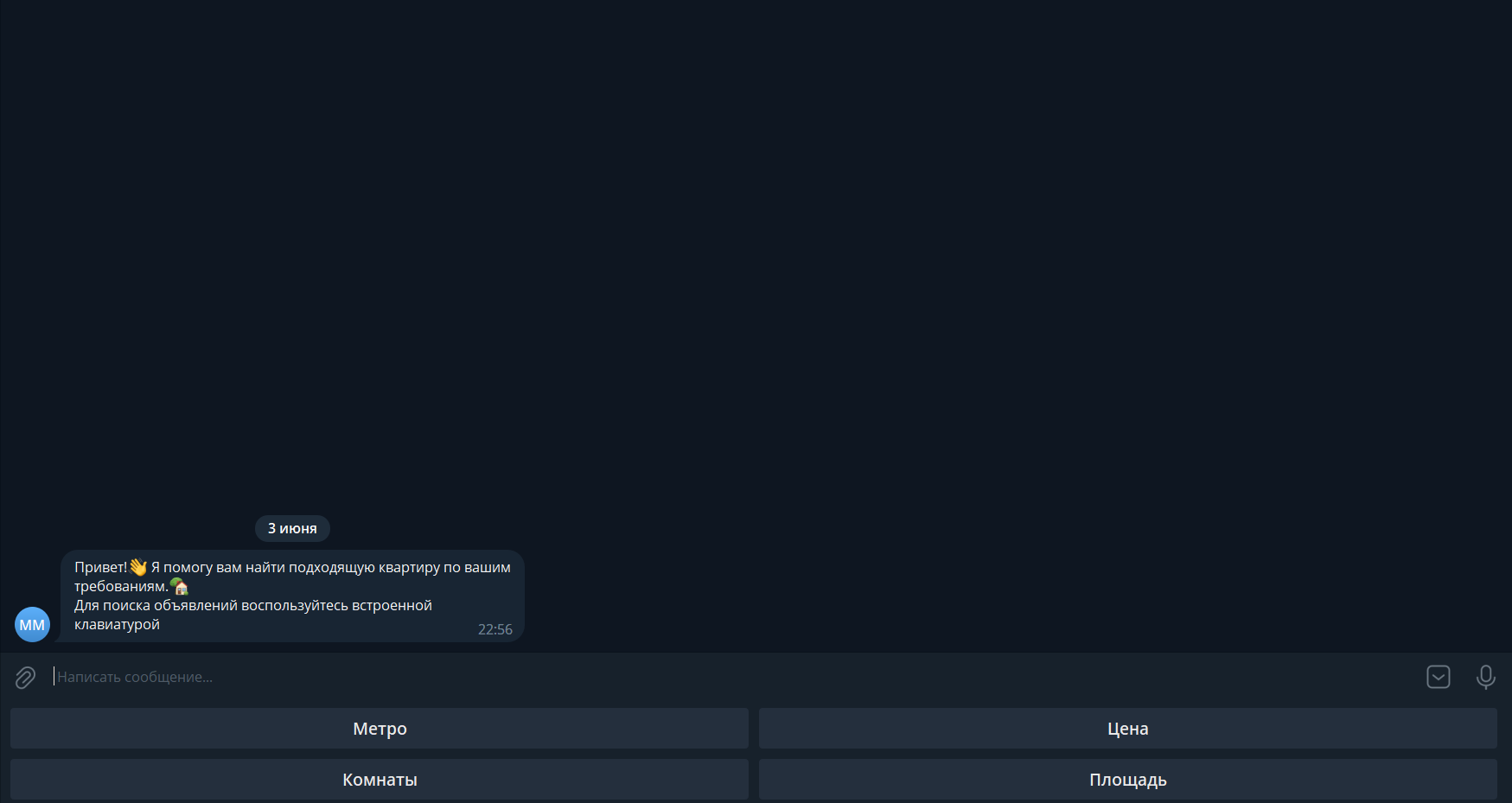
**Реализуемая архитектура**



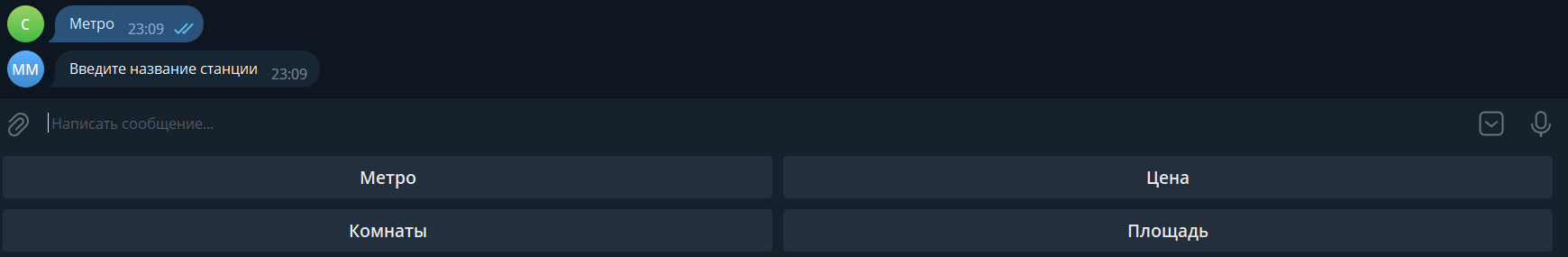
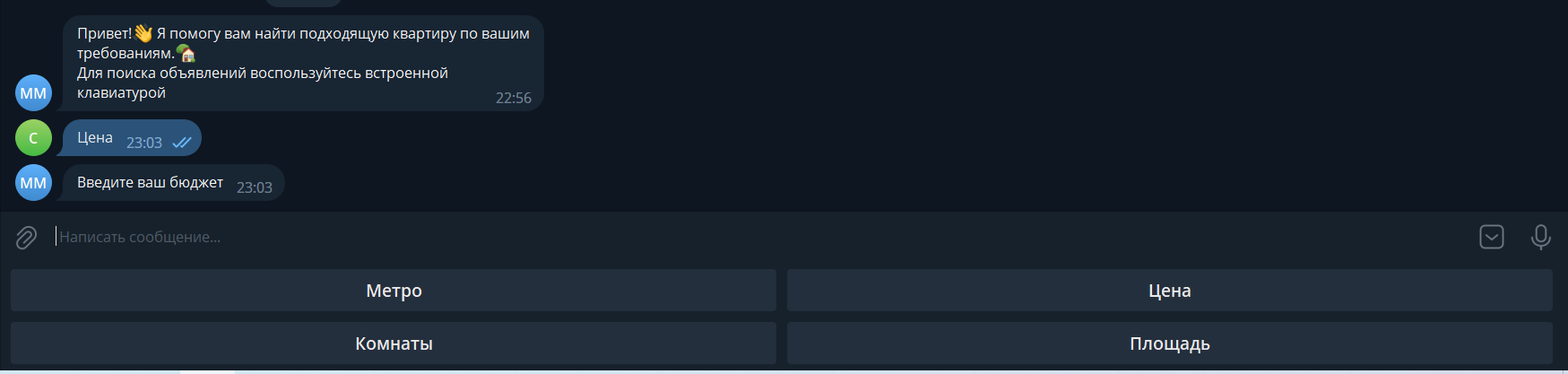
**Описание работы со стороны пользователя**

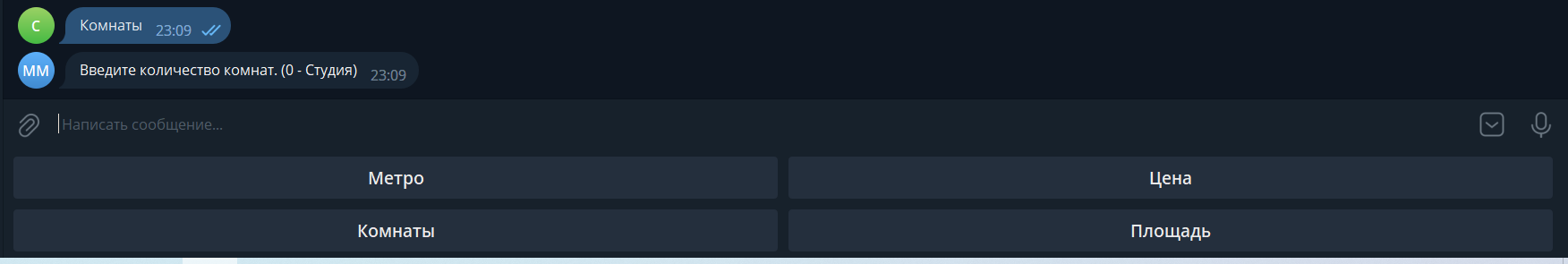
1. **Запуск бота командой /start.**

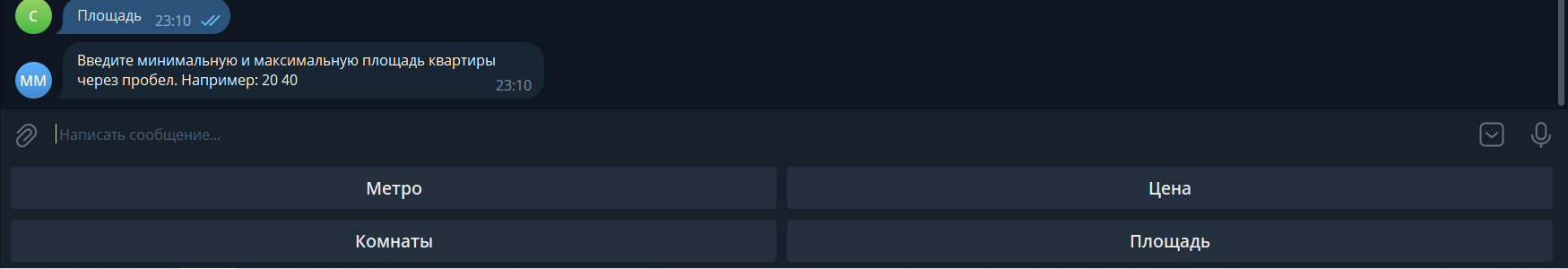
Пользователь отправляет команду боту, после чего бот отправляет приветственное сообщение. Вместе с приветствием появляется клавиатура с фильтрами для отбора объявлений



1. **Выбор пользователем фильтра для отбора объявлений.**

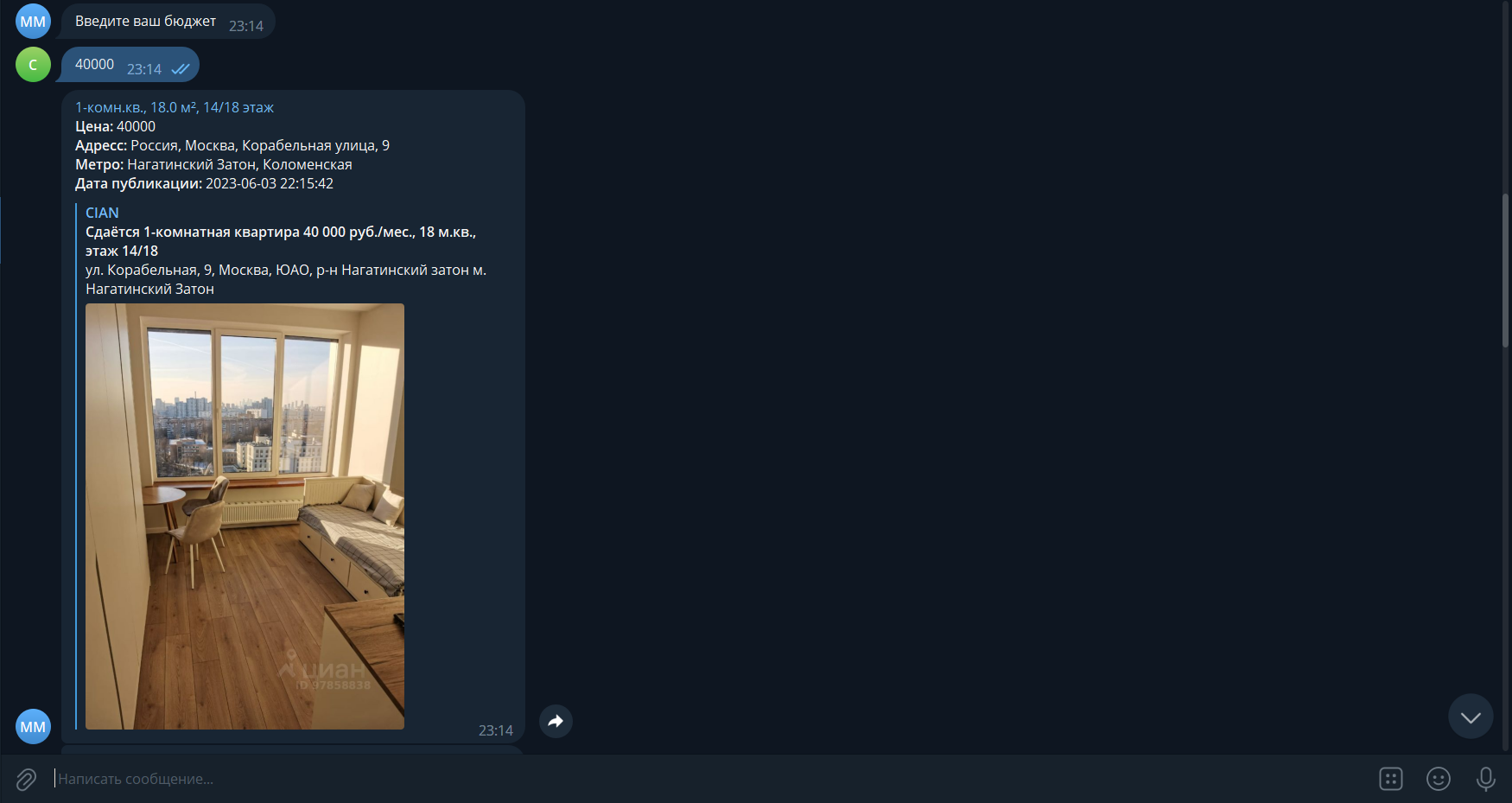
Пользователь выбирает один из четырех фильтров на клавиатуре. После выбора фильтра, бот отправляет сообщение, которое уточняет, как должны выглядеть вводимы данные. 





1. **Отправка данных для отбора и вывод подходящих объявлений.**

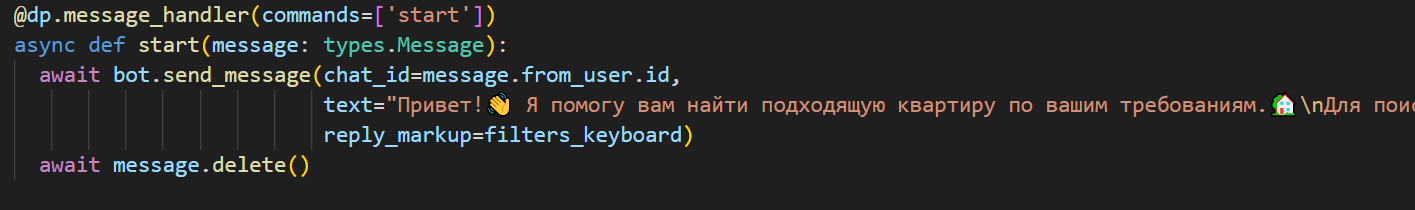
Получив данные для отбора, бот возвращает 5 самых новых объявлений, которые соответствуют критерию отбора.



**Описание работы со стороны бота**

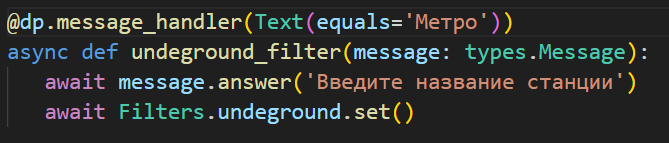
1. **Обработка команды /start.**

После получения команды /start, бот отвечает пользователю приветственным сообщением, к которому привязана Reply клавиатура с фильтрами. Затем бот удаляет сообщение пользователя, во избежание спама

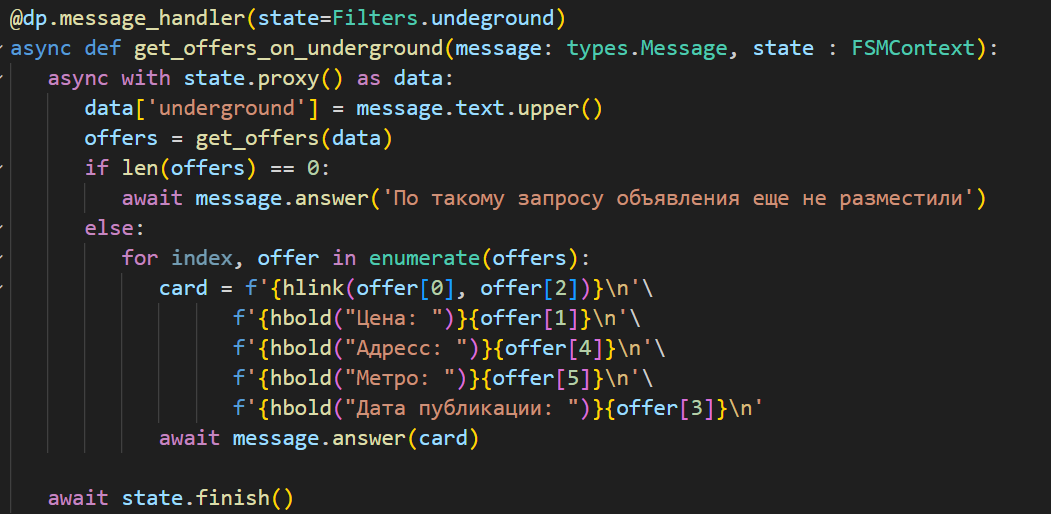


1. **Выбор пользователем фильтра для отбора объявления.**

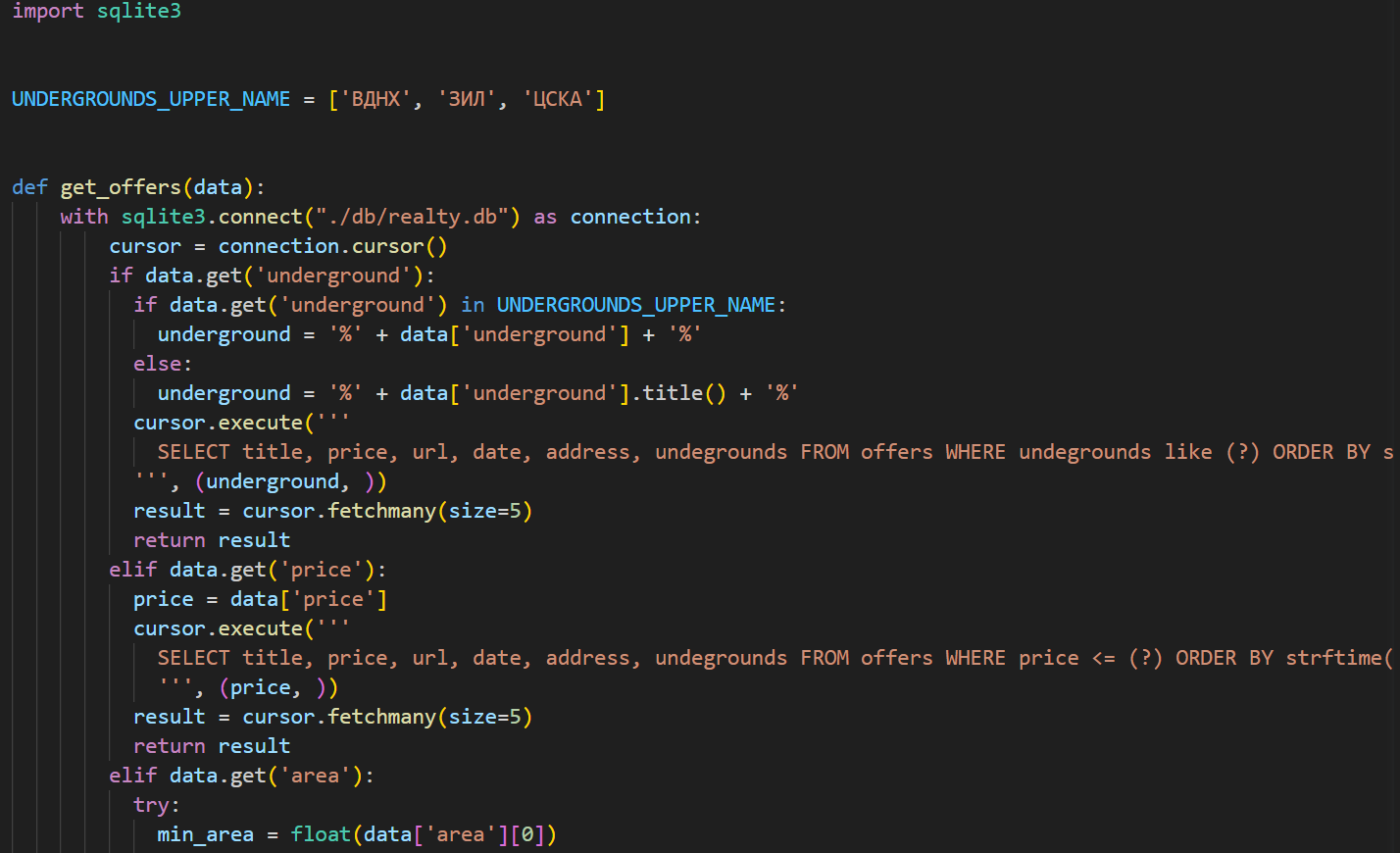
После выбора одного из фильтров на клавиатуре, бот отправляет сообщение, которое уточняет, как должны выглядеть вводимые данные. Затем бот переходит в машину состояний, которая соответствует конкретному фильтру. Благодаря этому, бот находится в ожидании сообщения от пользователя с критериями отбора.



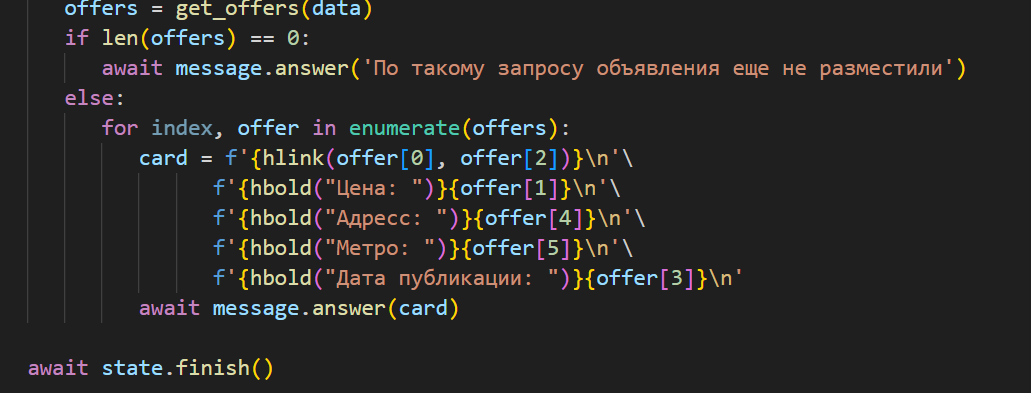
Получив данные для поиска, декоратор «ловит» момент, когда программа находится в машине состояний и запускает функцию, которая соответствует конкретному состоянию. Создается соединение с хранилищем состояний, в котором будут храниться данные. Туда записывает данные, полученные в сообщении пользователя, а ключом является фильтр.



Затем передает хранилище в функцию get\_offers(), в которой создается соединение с базой данных. Далее проверяет какой ключ есть в словаре data, чтобы определить, по какому фильтру производить отбор. Создается запрос в базу данных с указанием критерия отбора и сортировки по дате. Результат сохраняется в переменную result. (Результат получает функцией .fetchmany(), с параметром size=5, чтобы получить первые 5 объявлений. Это ограничение поставил, чтобы не создавать спам из огромного количества сообщений. Эту проблему хотелось решить созданием дополнительной клавиатуры, на которой была бы кнопка «Показать еще», которая позволила бы отправлять, например по 5 объявлений, после каждого нажатия. Реализовать эту функцию не получилось). Затем функция возвращает полученный список объявлений.

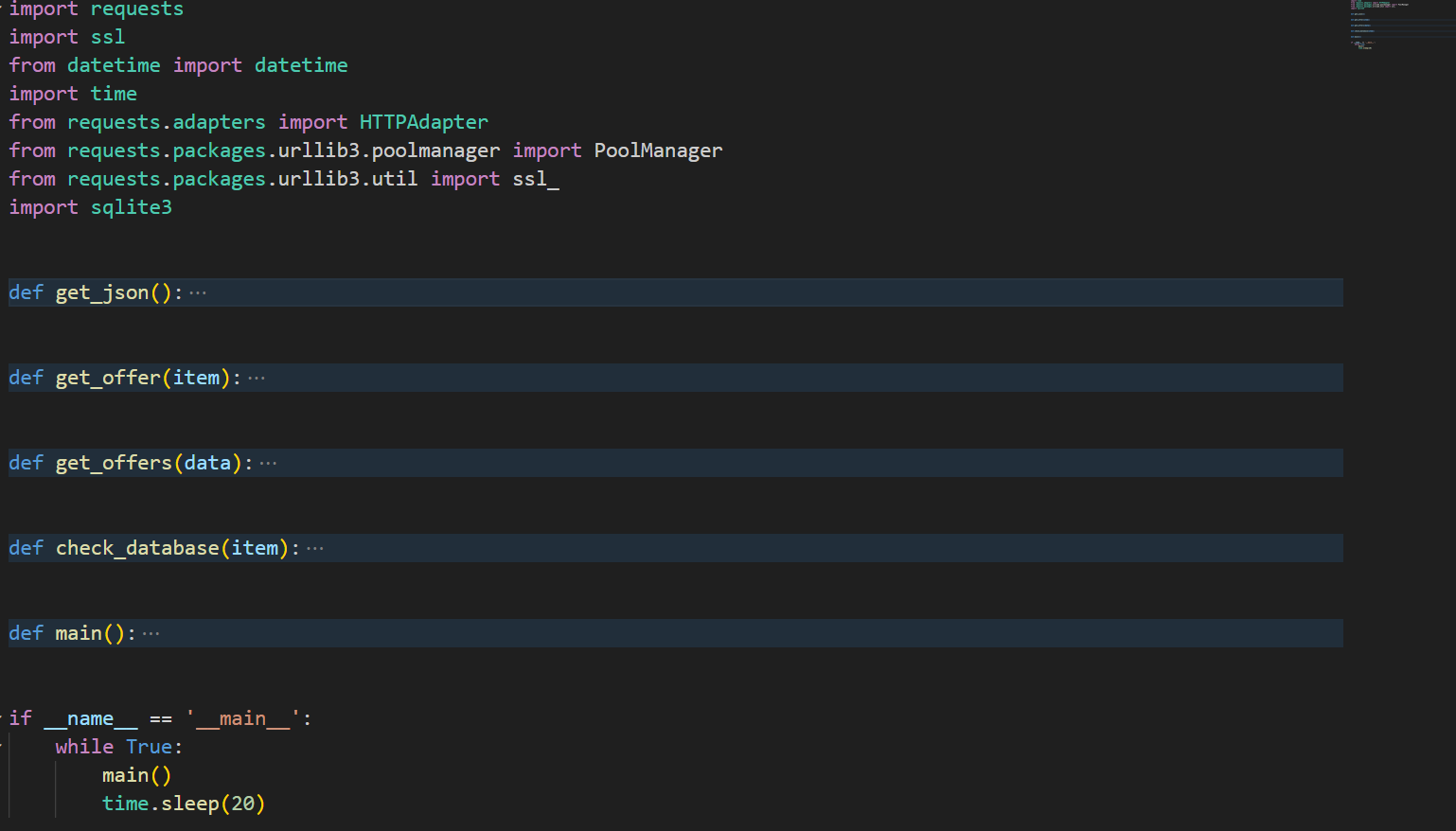


Полученный список записывается в переменную offers. Проверяет, список на наличие объявлений. Если по заданным критериям объявлений не нашлось и список пустой, то отправляется сообщений «По такому запросу объявления еще не разместили». После чего машина состояний завершается. Если же в списке есть объявления, то запускается цикл, в котором перебирается каждое объявление из списка, формируется карточка, которая затем отправляется пользователю в телеграмм. После завершения цикла машина состояний завершается.



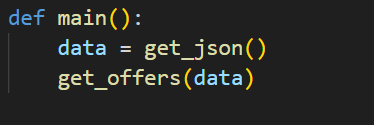
**Работа парсеров**

Общая структура на примере парсера для Циан.



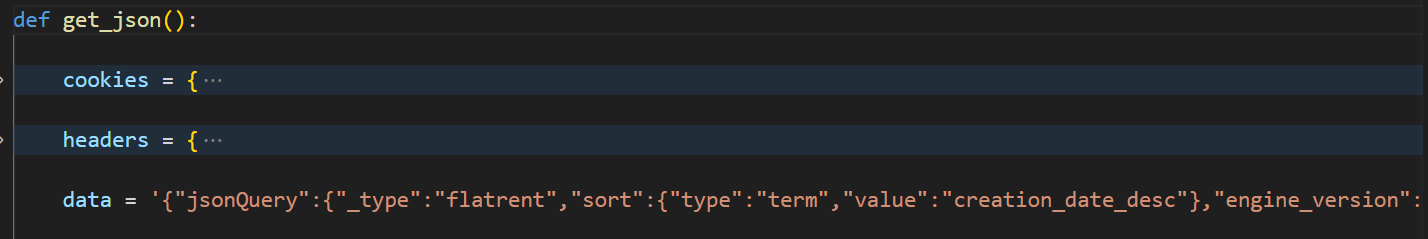
1. **Функция main()**

Вызов функции get\_json() и запись результата в переменную data. Передача json-словаря в фукнкцию get\_offers()



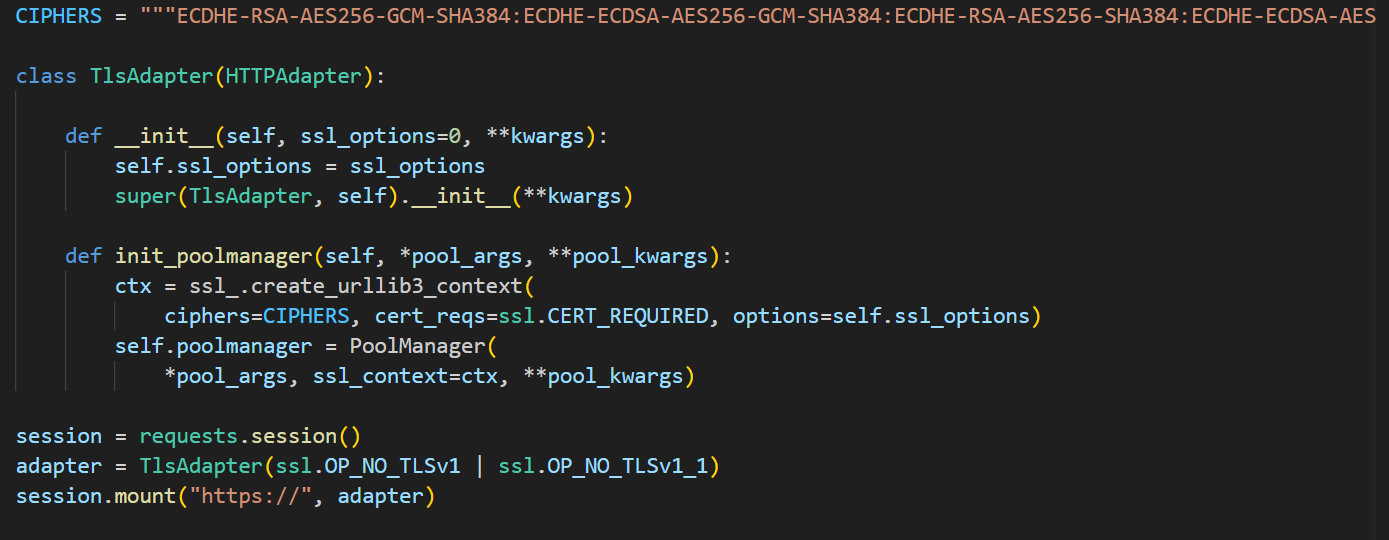
1. **Функция get\_json()**

Создание cookies, headers, data для дальнейшего использования в запросе. Значения получены из curl путем изучения запросов из браузера.

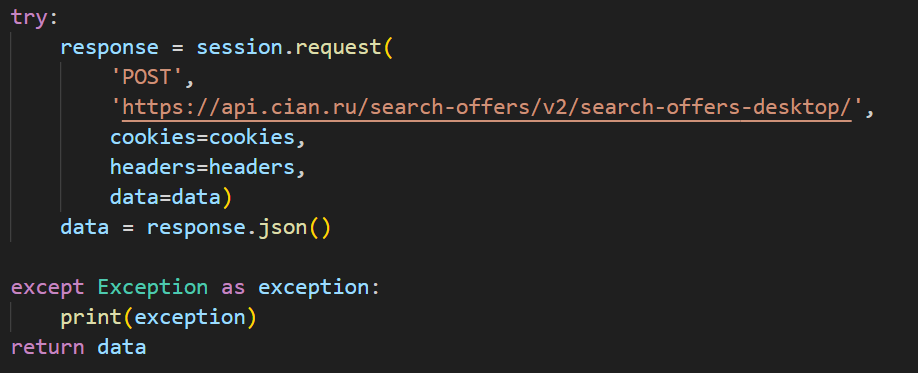


Переменная CIPHERS содержит список допустимых шифров для SSL/TLS-соединения.

Определяется класс TlsAdapter, который наследуется от HTTPAdapter. В этом классе определены методы для настройки адаптера HTTP и создания пула соединений с заданными SSL-опциями. В методе \_\_init\_\_ класса TlsAdapter инициализируется объект адаптера с параметрами SSL-опций.В методе init\_poolmanager класса TlsAdapter создается объект контекста SSL с заданными опциями и шифрами, а затем создается пул соединений PoolManager с этим контекстом. Создается объект сессии session класса requests.session(). Создается объект адаптера adapter класса TlsAdapter, в котором указываются опции безопасности для SSL/TLS-соединения. В данном случае указаны опции ssl.OP\_NO\_TLSv1 и ssl.OP\_NO\_TLSv1\_1, которые отключают использование протоколов TLS 1.0 и TLS 1.1. Объект адаптера adapter монтируется в сессию session для обработки запросов через протокол HTTPS, используя метод session.mount("https://", adapter).

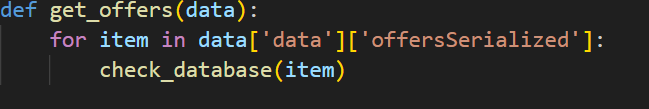


Пробуем отправить post запрос по полученной ссылке из curl, полученной путем изучения запросов из браузера. Получив положительный ответ, сохраняем его в переменную data в формате json.



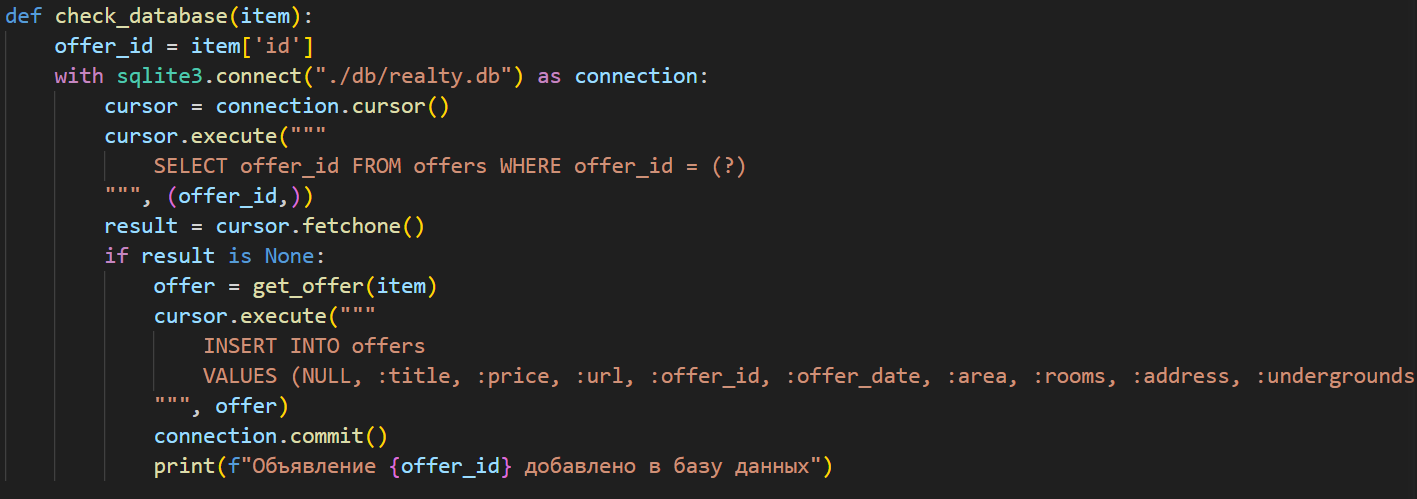
1. **Функция get\_offers()**

Полученный json отпраляет в функцию get\_offers(). Внутри функции циклом перебираем список, в котором хранятся объявления. Полученное объявление отправляем на проверку в функцию check\_database()



1. **Функция check\_database()**

В переменную offer\_id записываем id объявления из полученного объявления. Создаем подключение к базе данных, объект курсора. После, создаем запрос для проверки наличия объявления с таким id в базе данных. Результат сохраняем в переменную result. Проверяем результат, если result = None, то создаем переменную offer в которую поместим словарь, полученный из функции get\_offer(). Затем добавляем запись с этим объявлением в базу данных. Сохраняем изменение, отправляем сообщение о добавлении в терминал.



1. **Функция get\_offer()**

Создает словарь, в котором будут храниться данные. Ключи – элементы объявления:   
offer\_id – ID объявления на площадке

price – цена за месяц

rooms – кол-во комнат

title – заголовок объявления

address – адрес квартиры

undergrounds – ближайшее метро

url – ссылка на объявление

offer\_date – дата публикации объявления

area – общая площадь квартиры

floor – этаж квартиры



**Выводы**

В ходе работы над курсовой, удалось познакомиться с новыми полезными библиотеками, которые в будущем будут полезны. Самым полезным для себя отмечу опыт работы над реальным проектом и решением ситуаций, когда код в определенный момент просто перестает работать, а информации в интернете по этой проблеме просто нет.

**Библиография**

* [**https://www.youtube.com/@PythonToday**](https://www.youtube.com/@PythonToday) – все, что связано с парсингом
* [**https://surik00.gitbooks.io/aiogram-lessons/content/chapter1.html**](https://surik00.gitbooks.io/aiogram-lessons/content/chapter1.html)**,** [**https://docs.aiogram.dev/en/latest/**](https://docs.aiogram.dev/en/latest/) **-** aiogram
* [**https://habr.com/ru/articles/114772/**](https://habr.com/ru/articles/114772/)**,** [**https://habr.com/ru/companies/otus/articles/533354/**](https://habr.com/ru/companies/otus/articles/533354/) **-** изучение xpath