Geekbrains

**Телеграмм бот-помощник по инспектированию площадок в компании SintecGroup**

Программа:

Python - разработчик

Бондаренко Сергей Сергеевич

г. Обнинск

Март 2024 г

Оглавление

[1. Введение 3](#_Toc160990054)

[1.2 Интернет 3](#_Toc160990055)

[1.3 Мессенджеры 3](#_Toc160990056)

[1.4 Актуальность 3](#_Toc160990057)

[1.5 Суть проекта 4](#_Toc160990058)

[1.6 Решение проблем 4](#_Toc160990059)

[1.7 Использование инструментов 4](#_Toc160990060)

[2. Выбранные инструменты для решения задачи 5](#_Toc160990061)

[2.1 Чат-боты 5](#_Toc160990062)

[2.2 Telegram и его особенности 6](#_Toc160990063)

[2.3 Python 8](#_Toc160990064)

[2.3 Webhook 9](#_Toc160990065)

[2.4 Описание задачи и проблем 10](#_Toc160990066)

[2.5 Принятые решения 11](#_Toc160990067)

[3. Разработка Telegram-бота и Bitrix с помощью Python 13](#_Toc160990068)

[3.1 Схема работы Телеграмм бота по Инспекции 13](#_Toc160990069)

[3.3 connector\_bitrix.py 14](#_Toc160990070)

[3.4 Inspection\_bot.py 16](#_Toc160990071)

[3.4.1 Регистрация чат-бота 16](#_Toc160990072)

[3.4.2 Начало работы бота 17](#_Toc160990073)

[3.4.3 Список функций 17](#_Toc160990074)

[3.4.4 Приветствие 19](#_Toc160990075)

[3.4.5 Работа с замечаниями 22](#_Toc160990076)

[3.4.6 Добавление замечаний и завершение 24](#_Toc160990077)

[3.4.7 Функция callback\_update(call) 27](#_Toc160990078)

[3.4.8 Прерывание 29](#_Toc160990079)

[3.4.9 Запуск работы чат бота 29](#_Toc160990080)

[3.4.10 Тесты 31](#_Toc160990081)

[4. Подведение итогов, оценка результатов и заключение 32](#_Toc160990082)

# 1. Введение

## 1.2 Интернет

Интернет в современных условиях — это универсальная среда для общения, развлечений и обучения. В настоящее время в мире существует большое количество средств, форм способов общения, и немалая часть из них так или иначе связана с современными техническими возможностями, которые, в частности, представлены использованием глобальной компьютерной сети. Одним из средств является Мессенджеры

## 1.3 Мессенджеры

Мессенджер – это программа для мгновенного обмена текстовыми сообщениями и мультимедиа между зарегистрированными пользователями через интернет. Использовать мессенджер можно на компьютере, смартфоне или планшете в приложении либо в веб-версии для браузера.

В зависимости от возможностей конкретной программы, с помощью мессенджера можно:

* Общаться с другими пользователями один на один или в групповых беседах
* Отправлять текстовые или голосовые сообщения, а также стикеры, фото, видео и другие файлы
* Звонить и разговаривать и видеосвязи
* Создавать публичные каналы и подписываться на них
* Пользоваться чат-ботами

Обычно мессенджеры не отключаются, а продолжают работать в фоновом режиме и доставляют сообщения мгновенно, если устройство адресата подключено к интернету. Самые распространенные мессенджеры в России – WhatsApp, Telegram

## 1.4 Актуальность

Актуальность дипломной работы обусловлена высокой популярностью мессенджеров и таких средств автоматизации как чат-боты среди пользователей сети интернет. Чат-боты позволяют упростить ежедневные рутинные задачи, такие как получение информации о погоде, пробках, последних новостях и другие. Главным достоинством относительно классических приложений является возможность совмещения всех возможностей и на платформе одного мессенджера.

## 1.5 Суть проекта

Проект дипломной работы представляет из себя чат с автоматическим собеседником, который помогает пользователю выполнять простые автоматизированные операции:

* выбрать доступную площадку исходя из меню кнопок:
* получить информацию по сделанным замечаниям ранее
* добавить замечание

## 1.6 Решение проблем

Чат-бот будет решать проблемы:

* высоких трудозатрат сотрудников
* хранение и консолидирование данных из разных отделов, которые имеют дело с инспекцией площадок такие как блок-контроля, инспекторы, и ответственные за площадки

## 1.7 Использование инструментов

Инструменты, которые я буду использовать:

* **Python -** язык программирования c библиотекой **telebot**.
* **Telegram** мессенджер (чат-бот)
* **Bitrix** хранение и предоставление
* **Webhook Bitrix** будет происходить обмен данными и запись в базу

#### 1.8 Состав команды

Дудукина Анастасия (Заказчик)

Суров Михаил (Проектировщик)

Бондаренко Сергей (Разработчик)

Буриков Екатерина (Администратор Bitrix)

# 2. Выбранные инструменты для решения задачи

## 2.1 Чат-боты

В последнее время в мессенджерах стали популярны такие сервисы как чат-боты. Сам термин «чат-бот» был придуман Майклом Молдина в 1994 г. для описания разговорных программ. Изучив современное состояние использования чат-ботов в мессенджерах, можно прийти к выводу, что чат-боты являются универсальными средствами, способными к решению разнообразных задач – от общения до развлечений, от предоставления медицинской консультации до заказа товаров и услуг посредством специализированных прикладных решений, от распознавания эмоций до решения сложных консалтинговых задач в службах поддержки клиент ориентированных информационных систем. В любом случае, в независимости от платформы, чат-бот – это прикладная программа, которая, получая информацию от пользователя, формирует корректные, логически обоснованные ответы. На сегодняшний момент существует огромное разнообразие чат-ботов.

Некоторые из возможных вариантов представлены ниже:

* игровые чат-боты (квесты, ролевые игры)
* новостные чат-боты
* чат-боты для консультации и поддержке клиентов

В последнее время большие возможности открывают информационные технологии, связанные с чат-ботами. Они настолько прочно вошли в жизнь людей, что применяются во всех сферах человеческой деятельности, и с каждым днем их роль все больше увеличивается. В первую очередь это объясняется тем, что основную часть своего времени люди проводят за смартфоном в email-клиентах и мессенджерах.

Боты – это программы, которые выполняют различные задачи для пользователя, находящегося в мессенджере. Бот выглядит как обычный чат, однако общение происходит не с человеком, а с программой, которая может принять заказ на вызов машины, если это – бот такси, или прислать свежие статьи, если это – новостной бот, или заказать доставку еды на дом, если это – ресторанный бот. Не так давно чат-боты обрели большую популярность, перевоплотившись из развлечения в более серьезную вещь, так как они, в основном, стали использоваться для решения серьезных бизнес-задач. В эпоху информационных технологий – это нормально явление, а тем более – сети Интернет, ведь общество давно перешло на «новое» и деловое, и неделовое общение. Во-первых, чат-боты – это «платформы» для решения бизнес-задач. Во-вторых, чат-бот – это приложение, которое поддерживает диалог с пользователем, выбирая ответы из базы данных: вы спрашиваете, где пообедать и тут же получаете мгновенный ответ. Кроме того, чат-боты выполняют множество полезных функций по исполнению рутинных операций, поиску информации, объединению данных, работе с клиентурой.

Чат-бот как виртуальный собеседник имеет базу знаний, которая представляет собой наборы возможных вопросов пользователя и соответствующих им ответов. Наиболее распространенными вариантами для получения нужного ответа являются ключевые слова, совпадение фразы, совпадение контекста. Всегда существуют какие-то простые и легкие по выполнению дела, на которые не хочется тратить время. Тут на помощь всегда могут прийти чат-боты. Век пишущих машинок, ходьбы по библиотекам и бесконечных очередей в книжные магазины канул в Лету. Теперь для сбора информации можно использовать чат-ботов. Конечно, это делается при помощи общения с людьми, но отнюдь не всегда. Например, в рамках какого-либо мероприятия чат-боты могут сообщать всем участникам новости и предоставлять справочную информацию.

## 2.2 Telegram и его особенности

Telegram – это кроссплатформенное приложение, разработанное на языке программирования С++, позволяющее обменивается сообщениями и файлами большинства форматов. Мессенджер использует специально разработанную северную часть с закрытым кодом, работающим на серверах Германии и США. Telegram имеет ряд преимуществ:

* Приватность – все чаты зашифровываются, а сообщения уничтожаются через обозначаемое время или можно удалить по желанию
* быстрота – скорость доставки сообщений выше, чем у аналогов (например Watsapp)
* распределение – сервера Telegram расположены по всему миру, что повышает отказоустойчивость
* открытость – использование открытого протокола MTProto и API бесплатных для всех
* есть подписка с дополнительными возможностями, но она не влияет на использование ботов

Особенностями данного мессенджера являются секретные чаты и специально разработанный протокол шифрования MTPoto (ссылка на Wikipedia <https://ru.wikipedia.org/wiki/MTProto>).

Чат бот работает по принципу API, (Сообщение – Ответ). Он не может просто так написать сообщение первым. Всегда начинает первым пользователь, например командой /start

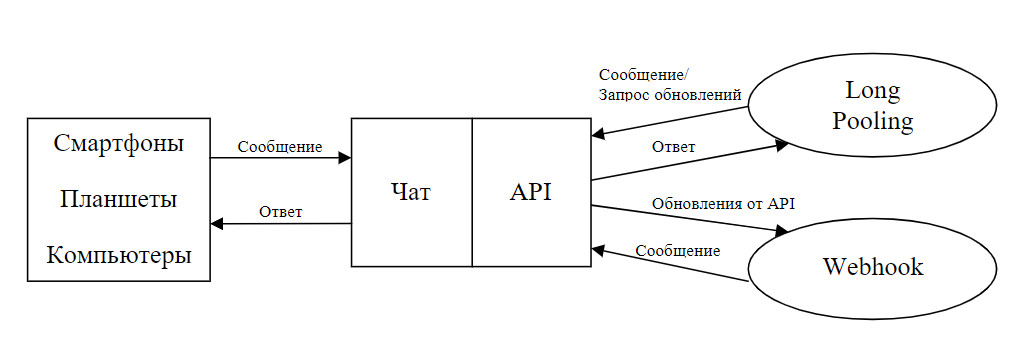


Рисунок 1 – Принцип работы чат-бота на платформе Telegram

Актуальность выбора Telegram.

На сегодняшний день, приложение Telegram является одним из самых востребованных мессенджеров во всем мире. Простота использования, легкость установки, доступность на множестве платформ и операционных систем и бесплатная модель распространения позволили мессенджеру уверенно закрепиться на рынке.

Одной из главных и уникальных функций данного приложения является возможность создания пользователем чат-бота, которого можно спроектировать практически под любые нужды. Все компании активно начинают внедрять в сою работу чат-ботов и использование мессенджеров. Это происходит во всех отраслях.

Все это из-за того, что там сконцентрирована самая большая и живая аудитория. Они же и являются потенциальными потребителями. Это неоднократно подтвердили различные статистические статьи.

Главной причиной активного введения в свой бизнес работу с мессенджерами и непосредственно чат-ботов является то, что у людей на сегодняшний день, в среднем, на мобильном устройстве стоит до 4 различных мессенджеров. Этот факт, как никакой другой, дает понять, что данная сфера популярна и этим фактом обязательно нужно пользоваться в целях продвижения, рекламы и улучшения своего бизнеса в целом. Так же удастся автоматизировать большое множество процессов, которые раньше выполнялись людьми. А вышеупомянутые чат-боты послужат нам средством для достижения этой цели.

## 2.3 Python

Для разработки логики работы бота будем использовать язык программирования Python

Python или же «пайтон» стоит на высоком уровне среди языков программирования. Он является полноценным скриптовым языком программирования. Его уникальность в том, что он универсален и подходит для решения различных сложных задач. Пайтон применяется во многих IT сферах, например: машинное обучение, разработка различных приложений, так же в вебе и т.д. Плюсы Python. Python это интерпретируемый язык. Он не компилируется как обычный язык. Это значит, что пайтон до запуска кода является просто текстом. Следовательно, на нем можно кодить почти везде, на всех платформах. Сам язык отлично спроектирован и логичен.

Примечателен язык также в том, что строчки кода сокращены до минимума. Простота языка — это очень хорошее качество, так как разработка программы ведется гораздо быстрее.

Вот в пример отрывок кода,написанный на языке Java.

Class Main {

Public static void main(String[] args) {

System.out.println(“hello world”);

}

}

В Python для выполнения того же самого кода придется написать всего

Print(“Hello world”)

Именно поэтому и считается, что данный язык программирования хорошо подходит для начинающих IT-специалистов.

Как применяют Python? Как уже было сказано выше, Пайтон универсален, его используют практически везде. Как в web разработке, так и на мобильных программах, а также при разработке ИИ (искусственного интеллекта).

Минусы языка программирования Python. Несмотря на множество плюсов (простота, лаконичность, универсальность), у языка есть и минусы. Скорость слабая сторона языка. Например, язык Swift, который используется, например, для разработки программ на iphone OC, может выполнить работу в 8 раз быстрее, чем конкурент.

Пайтон не очень хороший вариант для выполнения задач, требующих больших объёмов памяти. Такого типа задачи лучше выполнять при помощи языка Си или Си++. Язык слишком сильно зависит от системных библиотек. Из чего следует то, что возникают сложности при переходе на другие системы. Конечно, данное затруднение можно исправить, используя Virtualenv, но у этого способа свои недочеты есть: костыли, повторение системных библиотек, переизбыток полных методов изоляции. Глобальный блок интерпретатора (GIL) не позволяет одновременно выполнить несколько потоков Пайтон в реализации c Пайтон, но Global Interpreter lock можно отключить на время, как например это сделано в пакете NumPy для математики. Вывод: можно уверенно сказать, Пайтон – популярный и универсальный язык программирования. Он востребован во многих сферах. Высокоуровневый язык программирования со своими достоинствами и недостатками

Он отлично подходит для разработки Телеграм-бота т. к. телеграмм бот не требователен к оперативной памяти, и я учился на разработчика Python

## 2.3 Bitrix

Bitrix он же Битрикс24 это набор полезных инструментов, которые помогают бизнесу работать: CRM, Видеозвонки, Документы Онлайн, Задачи и проекты, Контакт-центр, конструктор сайтов и Интернет-магазины.

Битрикс24 помогает руководителю выстроить в своём коллективе эффективную работу.

Мгновенно выходите на связь с любым сотрудником по видеосвязи и в чатах, редактируйте документы, общайтесь с коллегами в новостной ленте и группах.

Календарь поможет спланировать встречи, почта — собрать все письма в одном окне, а диск — хранить файлы внутри компании.

Использование разных методик планирования как задач, так и проектов:

* Сроки,
* Мой план,
* диаграмма Ганта,
* слоты в календаре.

Помогает организовывать работу по правилам Скрам, используя канбан-доску и другие готовые инструменты.

## 2.4 Webhook

Webhook – это автоматически сгенерированный HTTP-запрос, созданный на основе каких-то данных. Он запускается предопределенным событием или действием в исходной системе и передается системе, с которой исходная система пытается установить связь.

Webhook работает быстрее, чем опрос или API. Вместе с этим для разработчиков он является менее трудоемким с точки зрения работы. Применительно к приложениям, Webhook – это не что иное, как SMS-уведомления, которые мы получаем во время использования приложения. Например, при покупке некого товара в Интернете продавец присылает вам уведомление по SMS. Аналогично, каждый раз, когда в исходной системе происходит некоторое событие/действие, система принимающей стороны уведомляется через Webhook.

Webhook используется для связи приложений и быстрого обмена данными между системой-источником и системой-получателем. Это приводит к двусторонней связи между двумя различными сетевыми системами.

Ниже приведен список нескольких сценариев, при который Webhook справится лучше, чем любое другое средство связи приложений:

* Использование Webhook для передачи информации базы данных.
* Требуется мгновенный ответ приложения при выполнении определенного действия.
* Использование Webhook для беспрепятственной синхронизации данных клиентов в приложении.
* Связь должна быть взаимно-однозначной.

## 2.5 Описание задачи

В компании Sintec Group есть Инспектор, он выполняет определенные маршруты обходов по площадкам, на этих площадках выполняются какие-то работы. Инспектор через WhatsApp записывал и передавал в общий чат замечания по этим работам с фотографиями. Далее сотрудник блока контроля консолидировал всю информацию из WhatsApp в общий файл Excel и распределял эту информацию для ответственных лиц, которые эти замечания будут устранять.

Проблемы с таким подходом:

* Трудозатраты сотрудника на перенос данных из WhatsApp в Ексел были высоки
* Затруднялось выявление ответственных лиц
* Увеличивались сроки устранений замечаний (информация по замечаниям доходила не быстро)
* Усложняет подготовку отчета по выполненным работам.
* Хранение информации

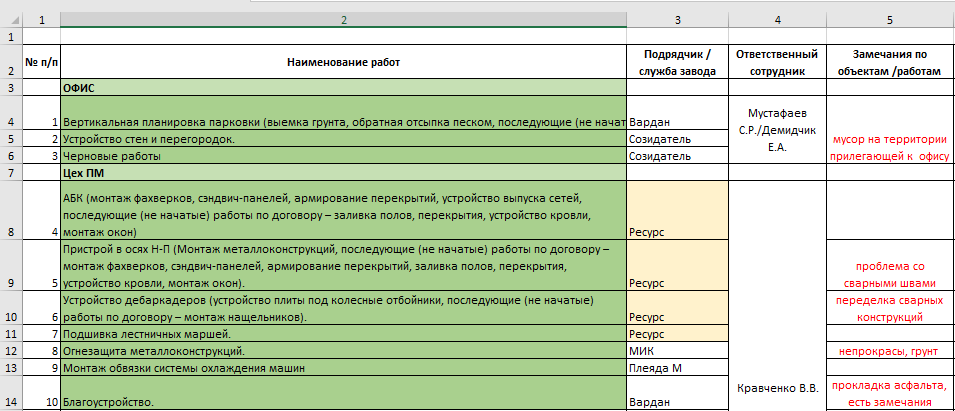


Рисунок 2. Пример заполнение И хранение данных в Excel

## 2.6 Принятые решения

Что бы привести в порядок справочную информацию по маршрутам и работам была создана в базе **bitrix** таблица по маршрутам «**Маршруты обходов**»

Эта таблица выступает в роли справочника по обходам в таблице указаны и записаны такие данные как:

* Название работы
* Площадка
* Главный Инспектор
* Инспектор
* Ответственный за работу
* Блок Контроля
* Статус маршрута – он может быть закрыт или активный

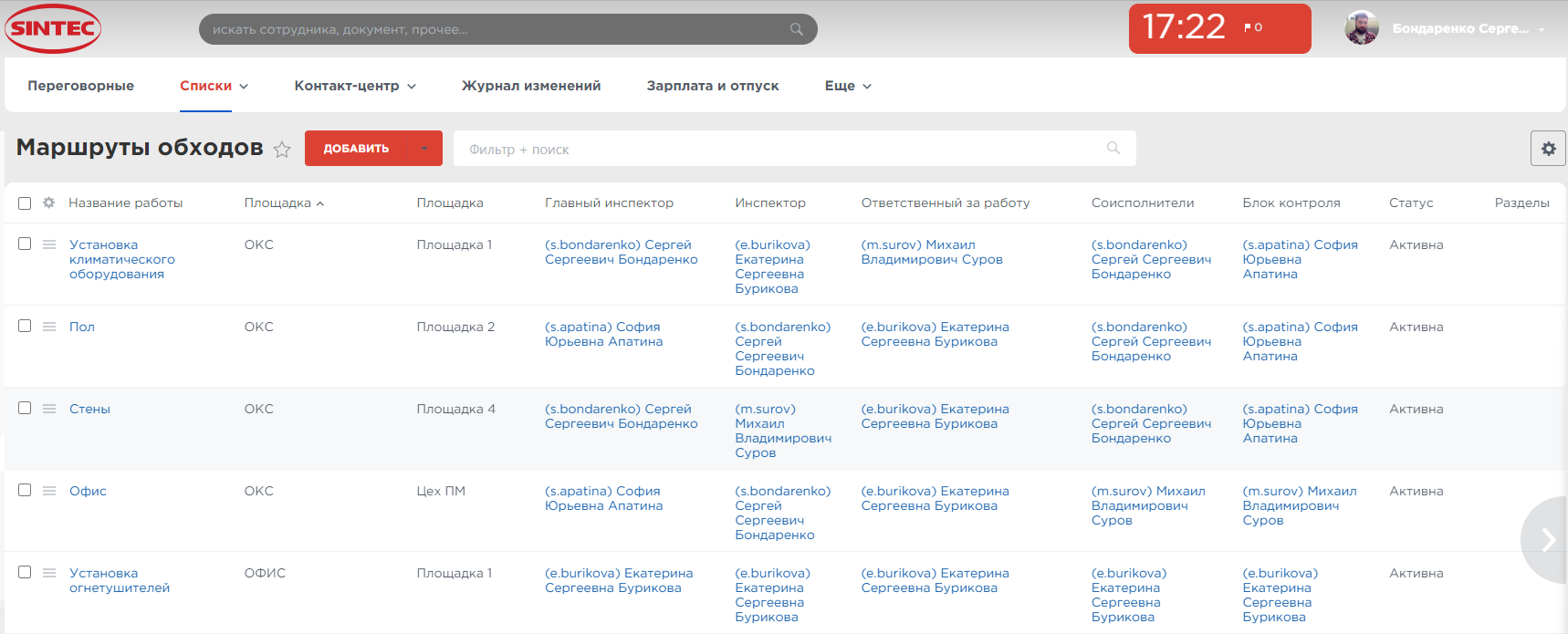


Рисунок 3. Таблица маршрутов в bitrix

Замечания будут записываться в Смарт-процессы.

Смарт-процессы — это автоматизированные рабочие процессы в рамках платформы Битрикс24. Они позволяют автоматически выполнять определённые действия и задачи в соответствии с заданными правилами и условиями.

Смарт-процессы в Битрикс24 помогают:

* упростить и автоматизировать повседневные операции;
* улучшить координацию работы между сотрудниками;
* обеспечить более эффективное использование ресурсов.

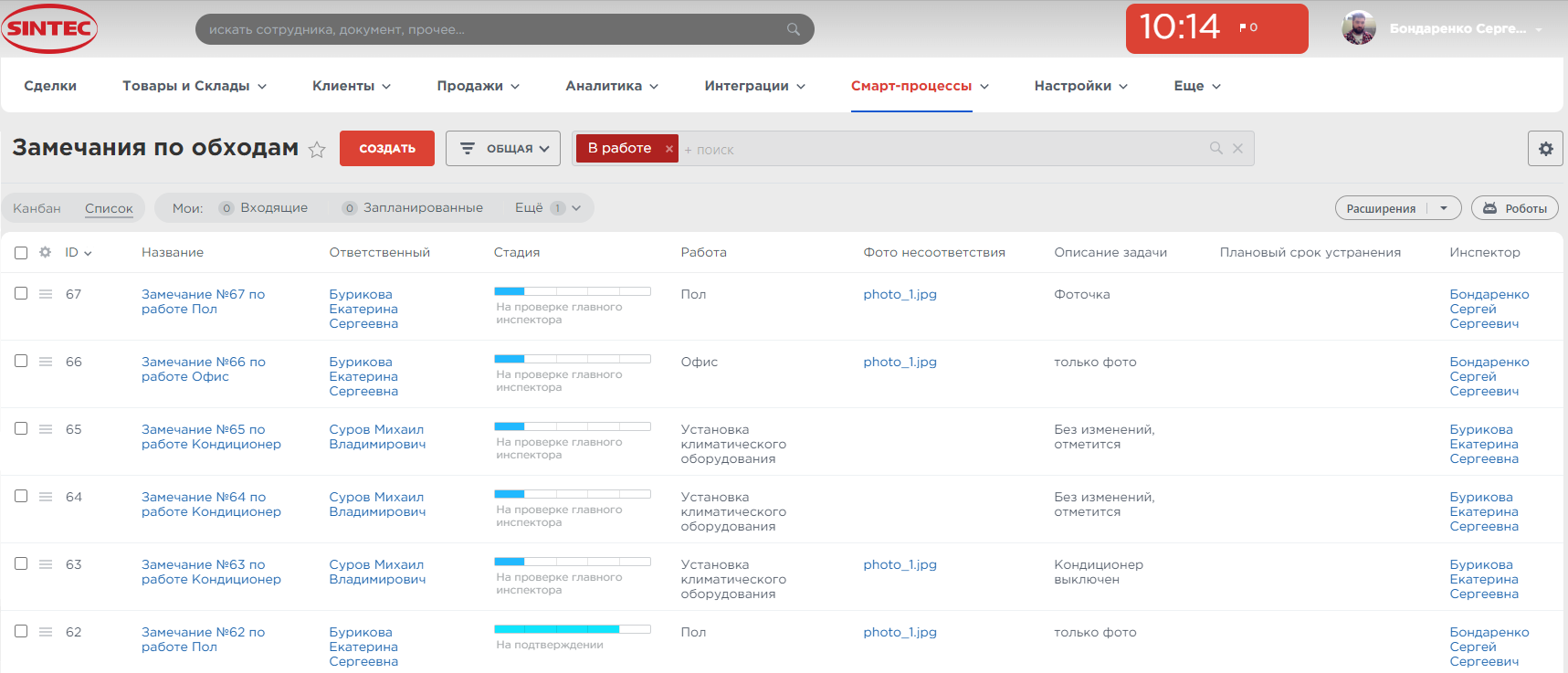


Рисунок 4. Смарт процессы по обходам (Список)

Смарт процессы имеют другой вид. Например «Канбан» он наглядно показывает статусы по замечаниям такие как:

* Новая
* В работе
* На подтверждении
* Завершить

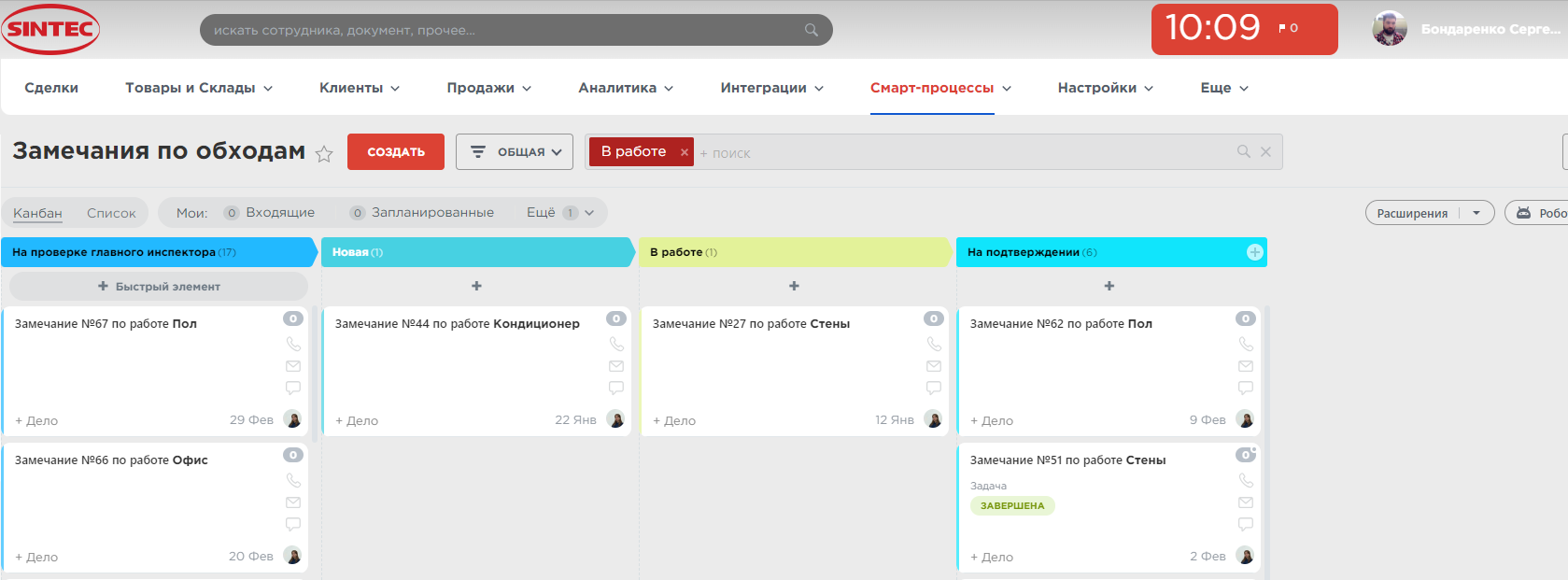
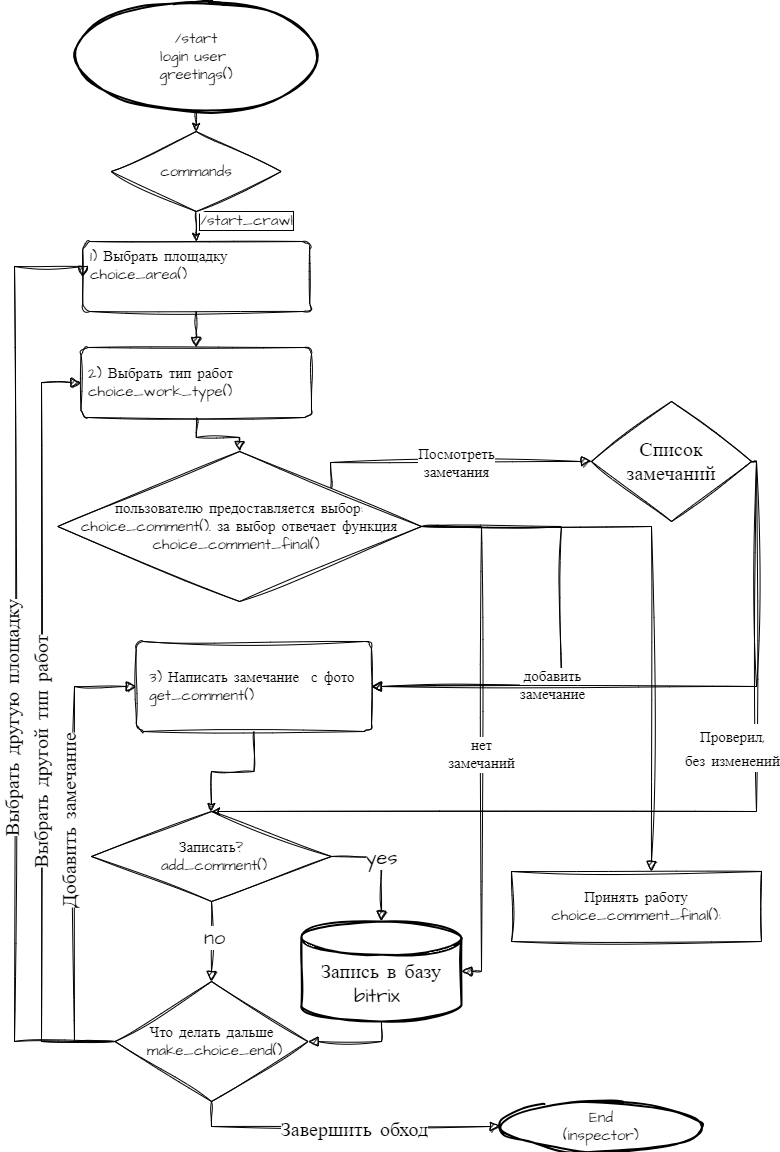


Рисунок 5. Смарт процессы по обходам (Канбан)

# 3. Разработка Telegram-бота и Bitrix с помощью Python

## 3.1 Схема работы Телеграмм бота по Инспекции

Заказчику была представлена и согласована блок схема работы бота то, как пользователь будет общаться с ботом и какие действия будут нужны для выполнения маршрута на площадке и фиксирования замечаний.



Что бы оперировать таблицами и смарт процессами bitrix через Telegram чат-бота, были созданы три модуля в Python:

* connector\_bitrix.py
* Inspection\_bot.py
* Settings.py(Содержит токены бота и webhook и другие контекстные переменные)

## 3.3 connector\_bitrix.py

В Bitrix для webhook есть стандартные методы, которые возвращают данные в формате json.

|  |  |
| --- | --- |
| **Метод Bitrix** | **Jписание** |
| **lists.element.get** | Метод возвращает список элементов или элемент. В случае успеха возвращает данные по элементу(там), иначе пустой массив. |
| **lists.element.update** | Метод обновляет элемент списка. В случае успешного обновления элемента ответ true, иначе Exception |
| **user.get** | Получение фильтрованного списка пользователей. Метод вернет всех пользователей за исключением: ботов, пользователей для e-mail, пользователей для Открытых Линий, пользователей Реплики. |
| **crm.item.list** | вернет массив элементов смарт-процесса с идентификатором entityTypeId |
| **crm.item.add** | метод добавляет элементы в смарт процесс |
| **crm.item.update** | метод обновляет элемент списке |
| **lists.field** | Метод возвращает данные поля. В случае успеха будет возвращен список полей с данными, иначе пустой массив. |

Для более точных запросов и работы методов, написаны функции в модуле **connector\_bitrix.py**: в этом модуле находятся функции, которые взаимодействуют с базой bitrix через вебхук (webhook) и формируют справочные данные или выполняют действия с таблицами по сигналу

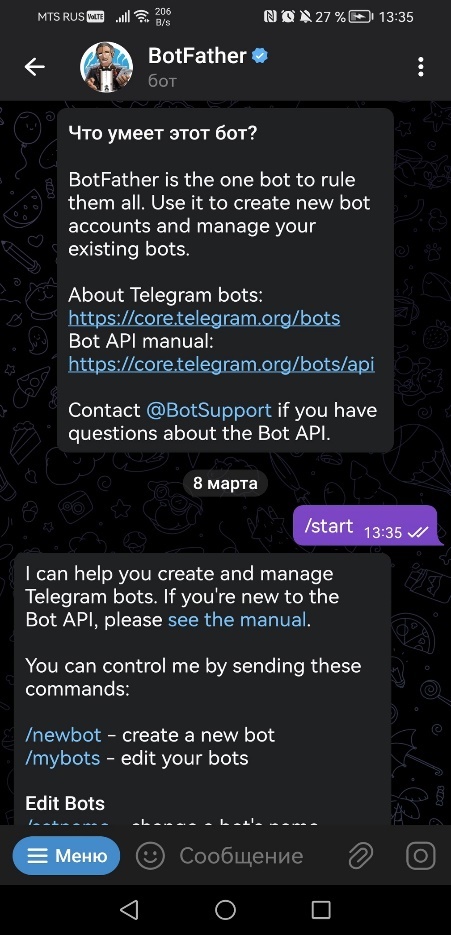
|  |  |
| --- | --- |
| **Функции** | **Описание** |
| **get\_user**(tg\_id\_user: int): | Возвращает лист id пользователя bitrix используя id телеграмм бота и ФИО |
| **get\_val\_of\_list**(name\_field): | возвращает словарь ключ значение поля, которое имеет формат списка |
| **get\_key\_area**(dict\_area:dict, name\_area): | возвращает номер ключа по значению из словаря |
| **replace\_status\_name**(text: str): | функция заменяет статус на более читабельный для пользователя |
| **get\_field\_route\_by\_id**(id, field='NAME'): | Возвращает конкретное поле, которое указано в аргументе по умолчанию поле «NAME» |
| **update\_status\_route**(id\_route): | Меняет статус маршрута на «закрытый» |
| **get\_area\_or\_work**(tg\_name\_inspector, name\_area=0, status=set.ST\_R\_ACTIVE) -> list: | Возвращает список уникальный площадок, если указан дополнительный параметр, name\_area то возвращает список работ на этих площадках, по умолчанию статус маршрутов «активный» |
| **get\_id\_work**(id\_user, name\_area, name\_work): | Возвращает id работы по пользователю площадке и его названию работы |
| **get\_reg\_list**(id\_type\_work): | Возвращает список замечаний на данной работе |
| **get\_comment\_photo\_url**(id\_comment): | Получаем список фотографий, которые были сделаны добавления замечаний |
| **save\_comment**(inspector\_id: int, route\_id: int, assigned\_id: int, comment: str, photo=None): | Функция сохраняет замечание в смарт процессе |
| **confirm\_coment**(id\_coment, confirm=True): | Отклонить или сохранить замечание по умолчанию сохранить если не указывать аргумент «confirm» |

## 3.4 Inspection\_bot.py

### 3.4.1 Регистрация чат-бота

Данный модуль основной, и предназначен для управления ботом, и его логикой.

Для начала нам потребуется зарегистрировать нашего бота в телеграм, то есть нам нужно получить токен бота, он же ID уникальный идентификатор. В этом может помочь специальный бот BotFather и команда /newbot. Далее надо будет ввести название бота. И так мы получаем token и создаем чат бот в телеграмме



Далее будем использовать библиотеку «telebot» и создадим с помощью нее переменную «bot»

import telebot

bot = telebot.TeleBot(set.inspection\_tokken, parse\_mode=None)

Эта библиотека хорошо подходит для небольших чат ботов рассчитанные на небольшое количество единовременных пользователей

Что бы чат бот разделял пользователей, которые с ним работают и не смешивал всех в одну кучу, создадим класс User в котором будет содержатся и записываться основные аргументы и показания пользователя.

В словарь **users** будут хранится пользователи.

users = {}

class User:

def \_\_init\_\_(self, chat\_id, first\_name, id\_tg\_user):

self.chat\_id = chat\_id

self.first\_name = first\_name

self.id\_tg\_user = id\_tg\_user

self.id\_bitrix\_user = None # [Id, ФИО]

self.area = None

self.works\_name = None

self.works\_id = None

self.photo = None

self.comment = None

### 3.4.2 Начало работы бота

Как говорилось ранее принцип работы бота это «Команда пользователя - Ответ» т.е. бот ждет какой-то команды либо текст от пользователя и только потом даёт какой-то ответ, который прописан в скрипте.

Такие команды, на которые бот будет реагировать можно прописать в функции-декораторе @bot.message\_handler(commands=['start'])

В данном случае команда start основная команда бота, которая будет инициализировать пользователя и запускать цепочку действий, которые описаны в функциях.

### 3.4.3 Список функций

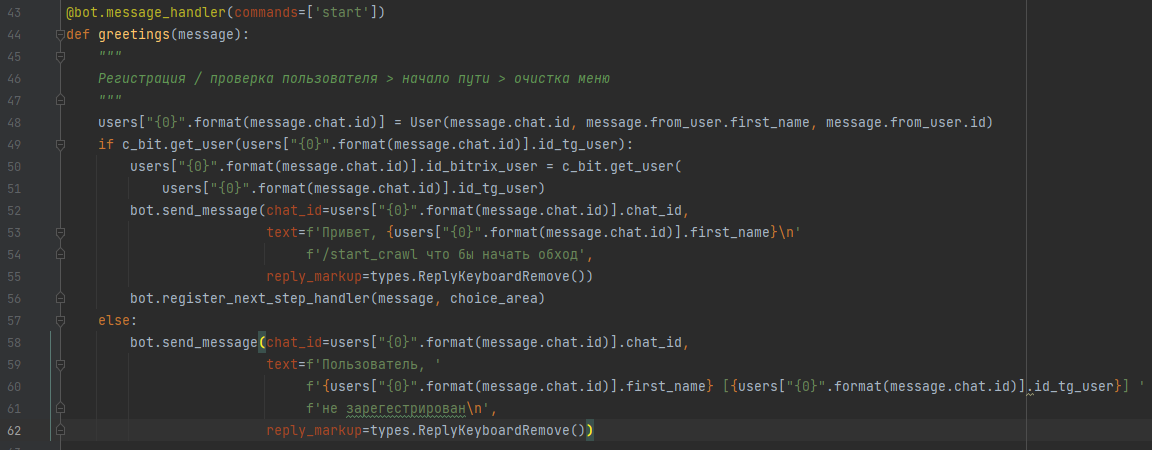
Функции они и создают ту самую цепочку действий. Как правило принимают всего 1 аргумент **message** т.е. ответ от пользователя из предыдущей функции

Функции создают условия для ответа пользователя, например какие кнопки вывести и какую информацию показать, что бы пользователь дал на него ответ, как бы ведет пользователя по шагам

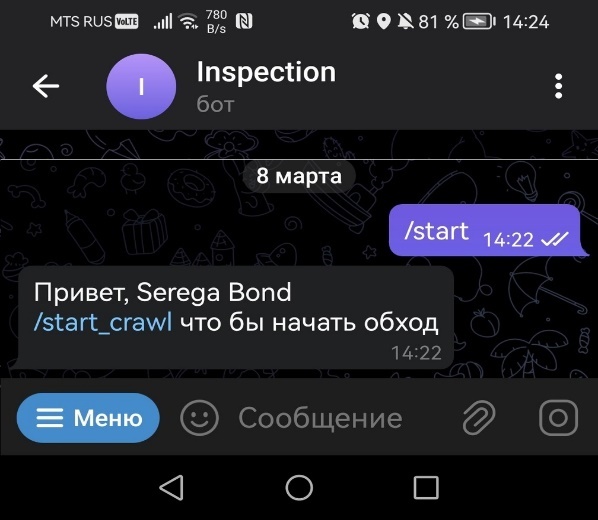
Попытаюсь описать функции и связь между ними в таблице ниже

|  |  |
| --- | --- |
| Функция | Описание |
| greetings(message) | Первая функция, которая запускается командой /start  Определяет пользователя, показывает имеющейся цепочки (на данный момент у нас есть маршрут обхода start\_crawl) |
| choice\_area(message) | Пользователь определен, для него формируется список площадок функцией makeList() |
| choice\_work\_type(message) | message Выбранная площадка из функции choice\_area() + записываем площадку в класс  return: выбираем название работ на основе пользователя и площадки >> формируется список работ для выбора |
| choice\_comment(message) | message: Выбранный тип работ из функции choice\_work\_type() + записываем тип работ в класс  return Проверка на выбор тип работ что бы пользователь не ввел что-то свое, а выбирал только из меню  >> записывает в класс пользователя название работ  >> определяем Id работ  >> проверка есть ли замечания, от этого зависит наполнение меню выбора  >> выводит количество замечаний  >> формирует список для выбора "показать все замечания"  > выводим меню для выбора что делать дальше и как быть с этой работой |
| choice\_comment\_final(message) | message ответ функции choice\_comment  return обработка ответа пользователя |
| get\_comment(message) | Если пользователь в предыдущем ответе choice\_comment\_final выбрал происходить направление на эту функцию |
| add\_comment(message) | message: Ответ после обработки сообщения  return: сохраняем в битрикс >> создаем смарт процесс замечания |
| make\_choice\_end(message) | Завершающая функция |
| callback\_update(call) | Функция, которая обрабатывает нажатие кнопки, которая внутри чата |

### 3.4.4 Приветствие



Пример функции приветствия

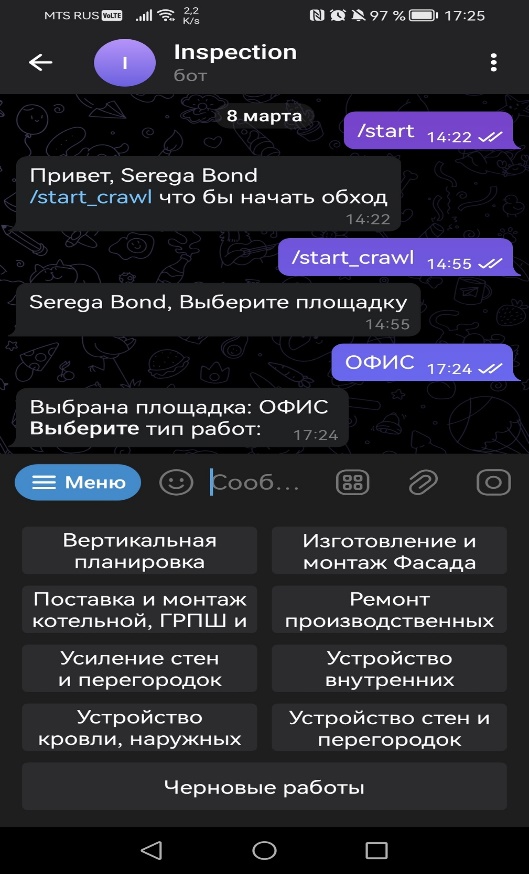


Приветственное сообщение

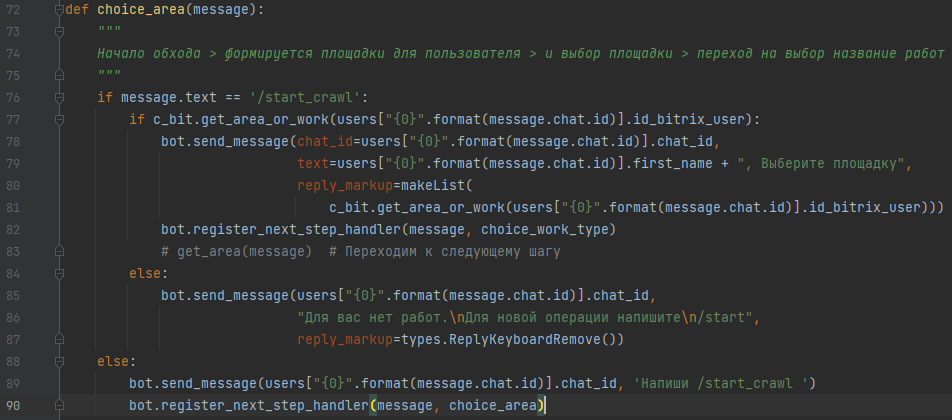
После того как пользователь определен, в зависимости от его роли ему могут быть доступны резные действия. На данном этапе развития бота нам доступен только «обход» (start\_crawl)

Далее начинается диалог с ботом. Инициатива плавно переходит к боту. Бот будет говорить, что надо сделать и будет предлагать пользователю выбрать какой-то вариант ответа, который будет сформирован исходя из ответа пользователя.

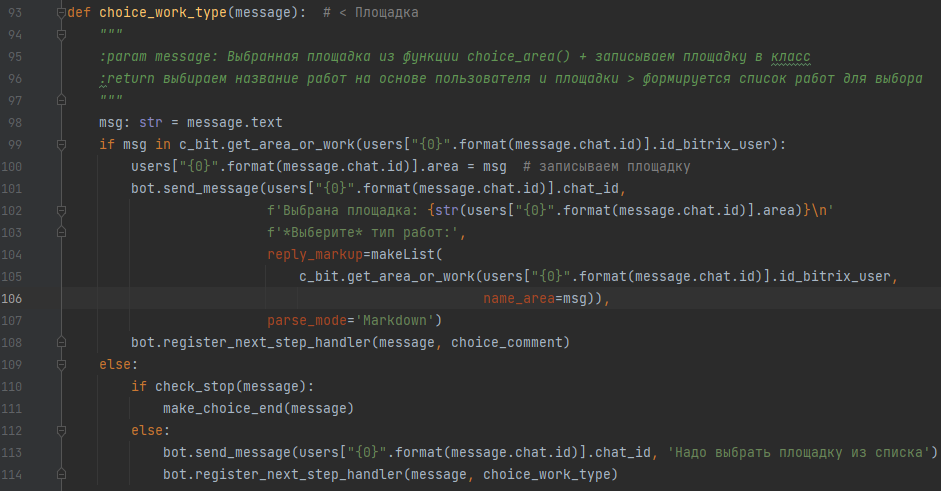
Что бы исключить человеческий фактор ошибок некорректных ответов и неправильных написаний например в названии работ, будет использоваться меню выбора в формате кнопок, которые возможно в Telegram, а в функциях будет проверка на правильность выбора



Выбор пплощадки и далее выбор работ в зависимости от площадки



Пример функции выбора площадки



Пример функции выбора работ

### 3.4.5 Работа с замечаниями

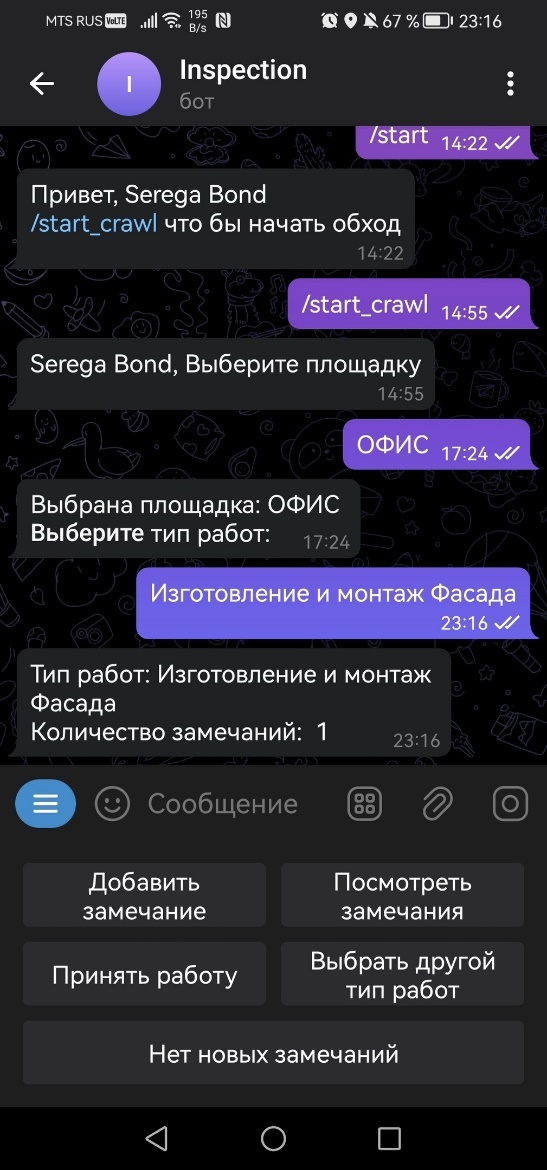
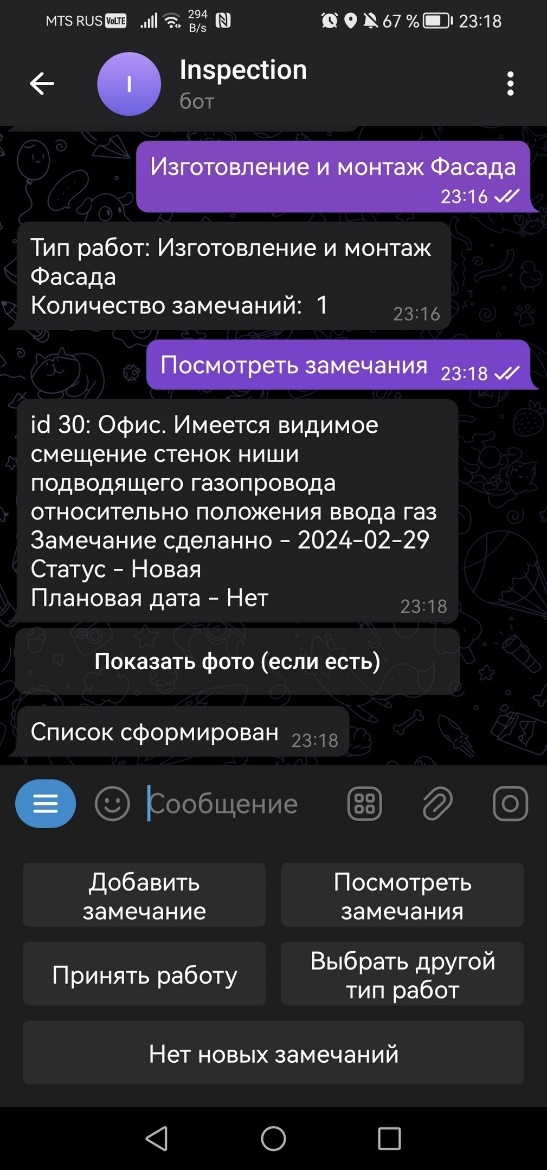
После того как выбрана площадка и работа на ней, мы можем работать с замечаниями.

Кнопка «Посмотреть замечание» выведет список всех незакрытых замечаний. Комментарий выводится не больше 100 символов это сделано для того что бы не выводилось много текста.

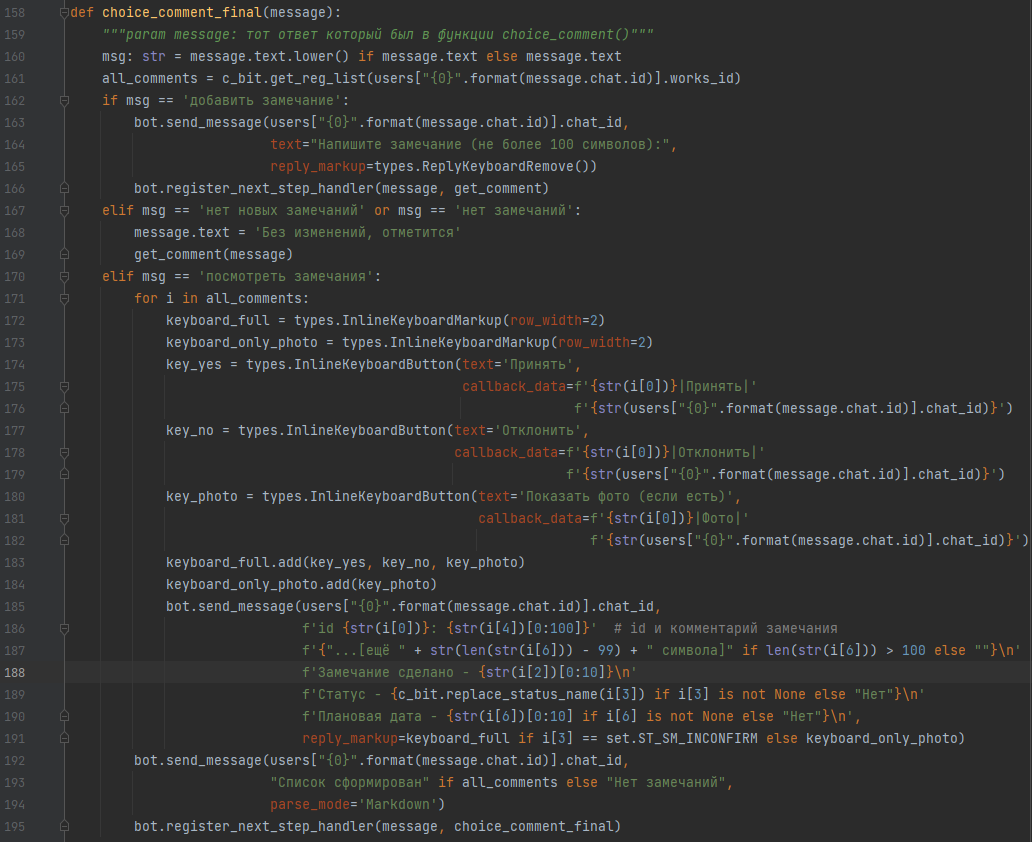
Если статус замечания будет на подтверждении, то появятся дополнительные кнопки «принять» или «отклонить»

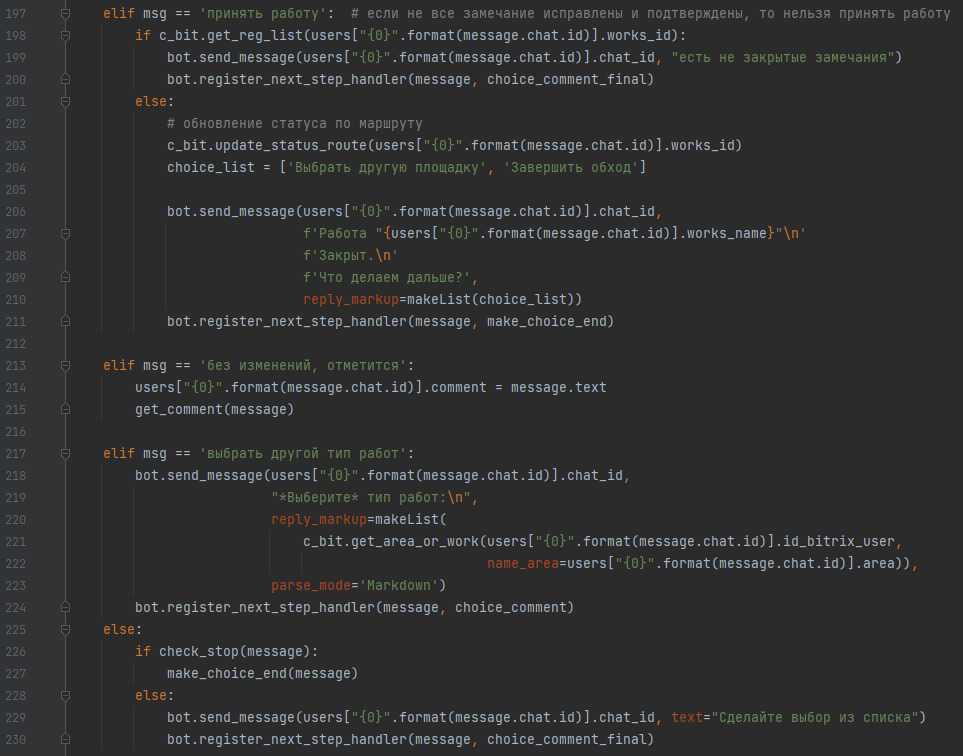
Инспектор может выбрать другую работу тогда его вернет на выбор работ.

Инспектор может принять работу, если ток там нет не закрытых замечаний.

Пример меню замечаний





Пример функции по выбору из меню замечаний

### 3.4.6 Добавление замечаний и завершение

Инспектор может добавить замечаний, или добавить фото и сделать к ней подпись.

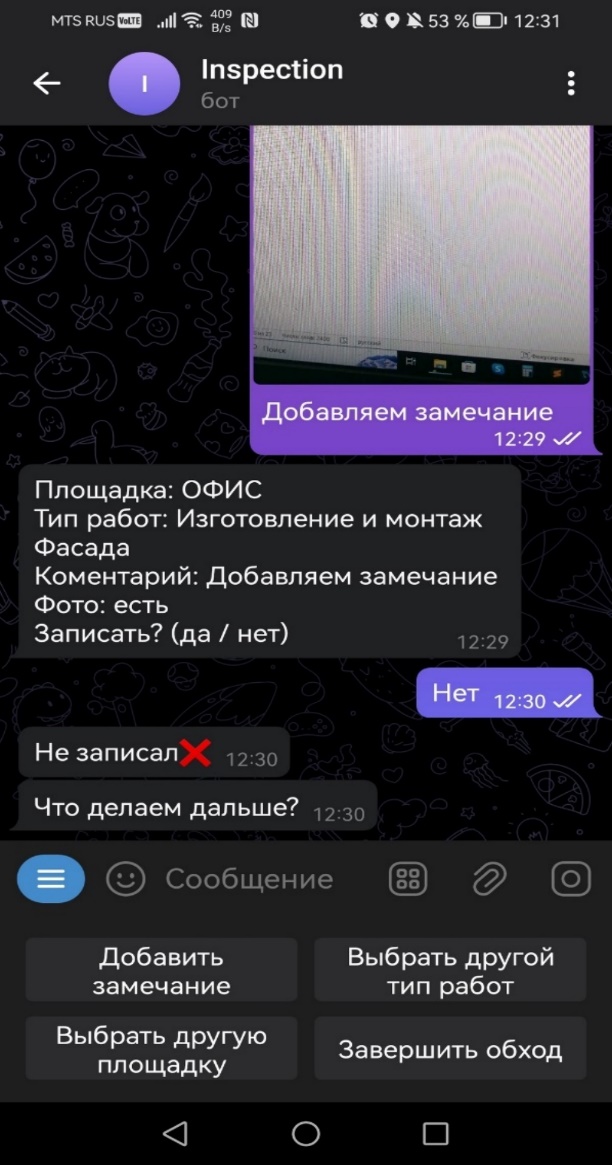
После того как Пользователь отправляет замечание бот отправляет его ранее вводимые ответы что бы подвести итоги того, что будет записано в базу

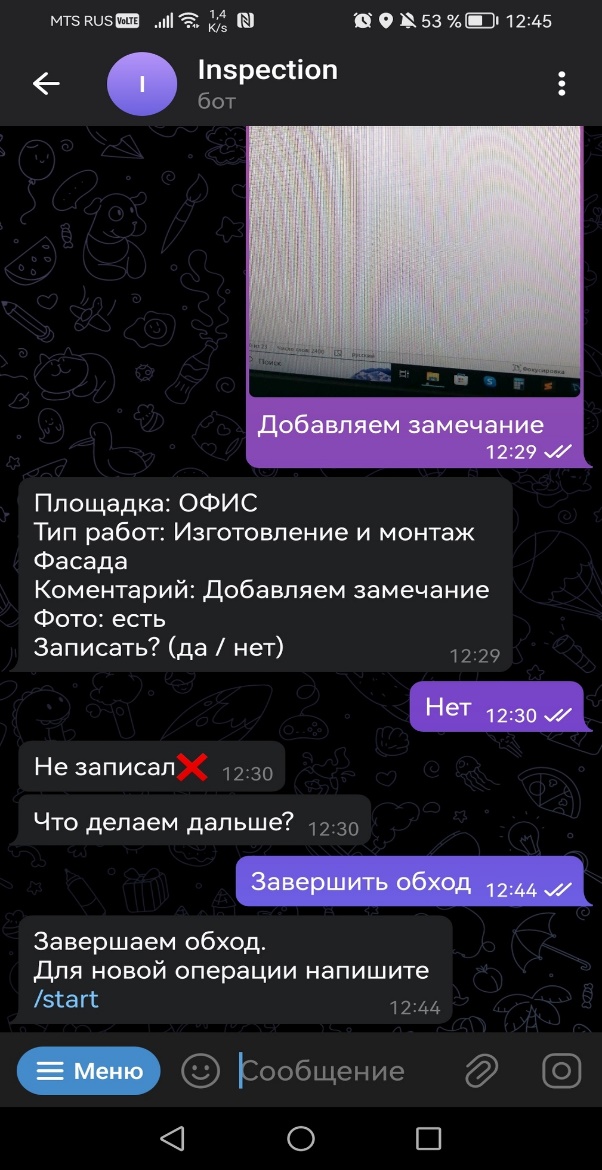
После того как Пользователь отправит или не отправит данные ему будет предложено заключительное меню.

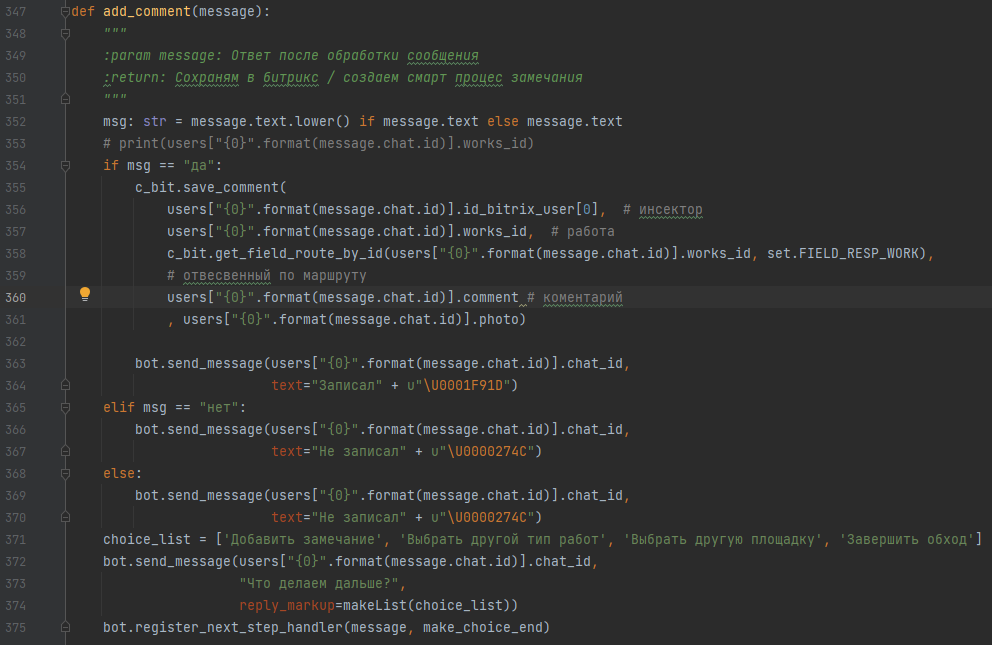
Он может:

* Продолжить добавлять замечания
* Выбрать другую площадку или работу на площадке которой уже находится
* Или завершить цепочку действий

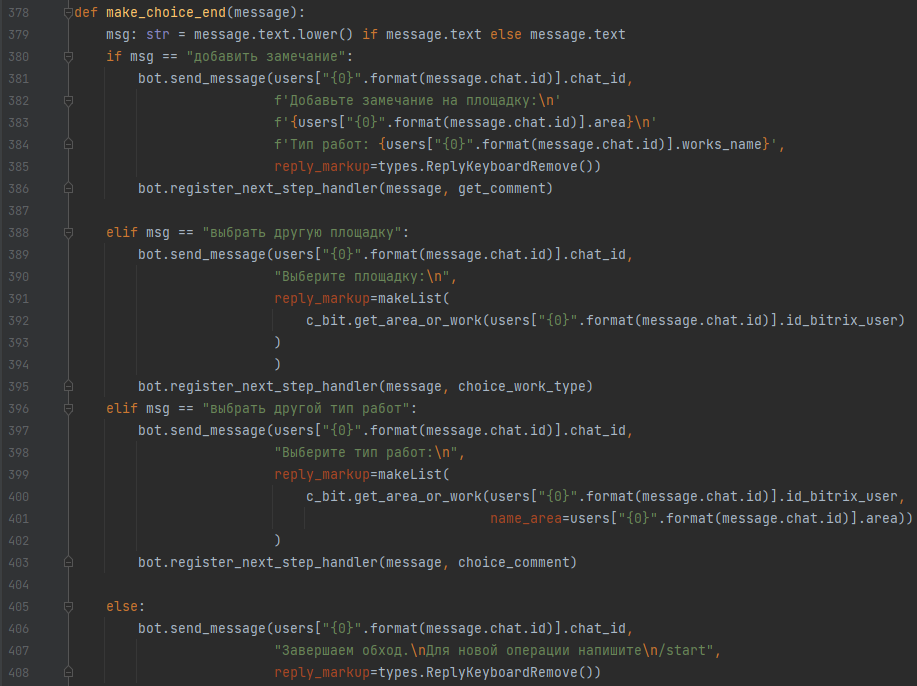
При завершении обхода очищаются / удаляются меню и кнопки



  
Добавление замечаний и завершение обхода



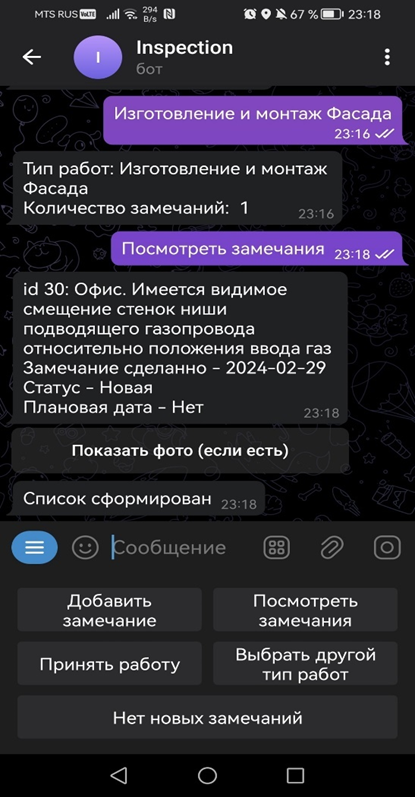
Функция записи данных в базу



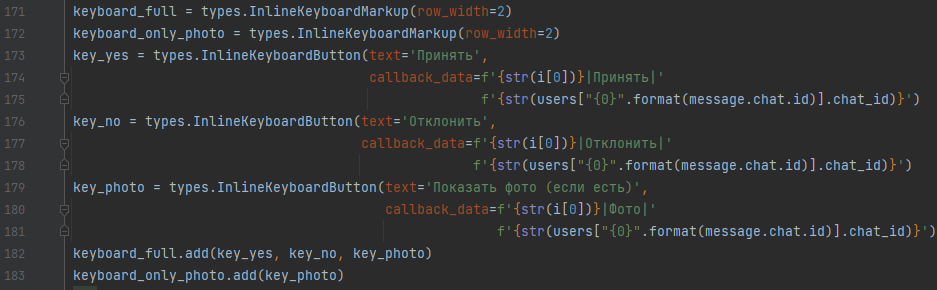
Функция завершения цепочки действий

### 3.4.7 Функция callback\_update(call)

Функция callback\_update принимает аргумент не сообщение, а кнопку, которая находится внутри чата, например кнопки «принять», «отклонить», «Показать фото (если есть)»



Сперва создаем эти кнопки и вкладываем их в сообщения



InLine кнопка состоит из название и код(id) в данном случае id составной состоит из id замечания, Названия и id чата.

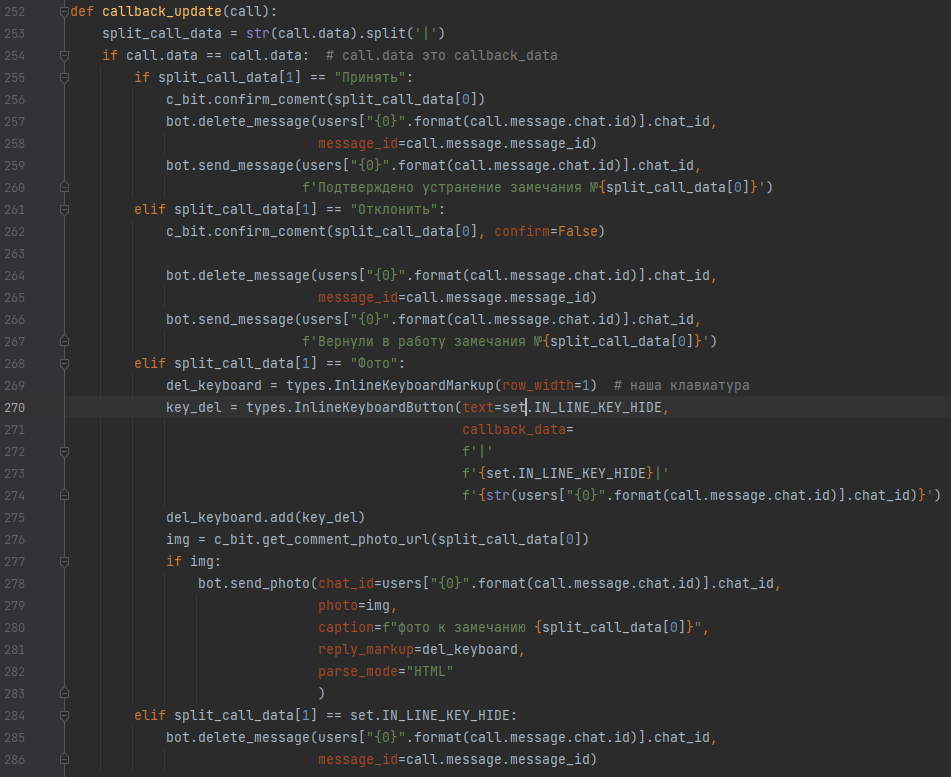
**@bot.callback\_query\_handler** функция декоратор отслеживает нажатие таких кнопок для этого создаем лист с теми кнопками которые возможно у этого пользователя.

Лист нужен для того, чтобы нажатие кнопки было только в зоне чата, в котором была нажата кнопка. Это необходимо для многопользовательского использования.



После того как Пользователь нажмет на такую кнопку будет возвращаться составная строка

Функция callback\_update(call) принимает эту составную строку и обрабатывает



### 3.4.8 Прерывание

После того, как мы выбираем команду start\_crawl мы запускаем цепочку действий как показано на блок схеме из пункта 3.1

Что бы прервать эту цепочку в любой момент, была сделана функция **check\_stop** она присутствует почти в каждой функции

В каждой функции есть дополнительное условие которое проверяет отчет пользователя на сообщение **stop**

def check\_stop(message):

if message.text == '/stop':

return True

else:

return False

### 3.4.9 Запуск работы чат бота

Для того что бы запустить чат бот используем встроенную команду или telebot

bot.polling(none\_stop=True, interval=0, timeout=60) наш бот будет опрашивать сервер телеграмм на какие то новые сообщения. Бывает так что бот вылезает с ошибкой разрыв соединения или ошибка в работе.

Для такого что бы бот сам мог перезапускаться и продолжать работу делаем бесконечный цикл wile и в try exept записываем bot.poolimg.

Чтобы уведомить пользователя о том, что бот перезагрузился или произошла ошибка, была сделана функция декоратор которая принимает любой текст. На любой текст бот будет отвечать, что произошла ошибка, но когда пользователь напишет команду start то начнется процесс начала работы бота.

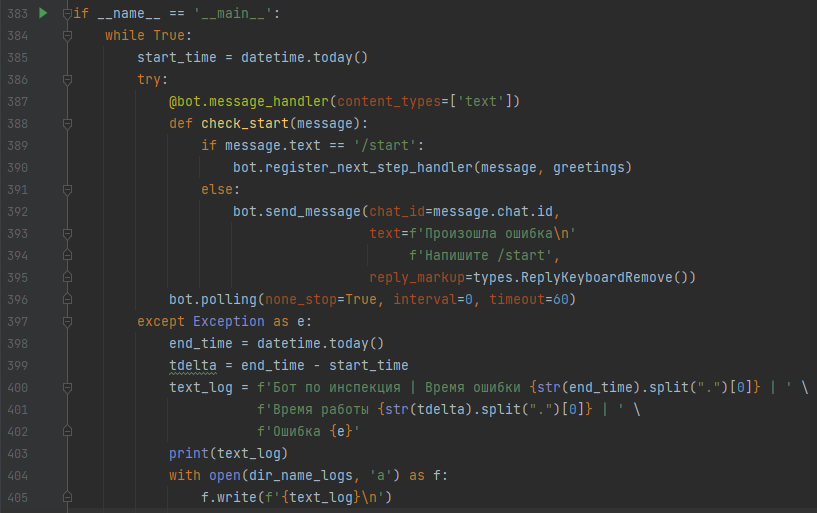
Для того что бы отследить ошибку и в дальнейшем исправить ее было создано логирование.

С помощью библиотеки **from pathlib import Path** создадим переменную, которая будет содержать путь к файлу логирования.

**dir\_name\_logs = str(Path(\_\_file\_\_).parents[1]) + '\logs\logs.txt'**

Логи будут находится на уровне повыше, а не вместе с проектом т.к. планируется иметь и другие проекты с чат ботами. И логирование с разных ботов будет собирается в один файл в одной структуре.

«Название бота | Время ошибки | Время работы | Ошибка»



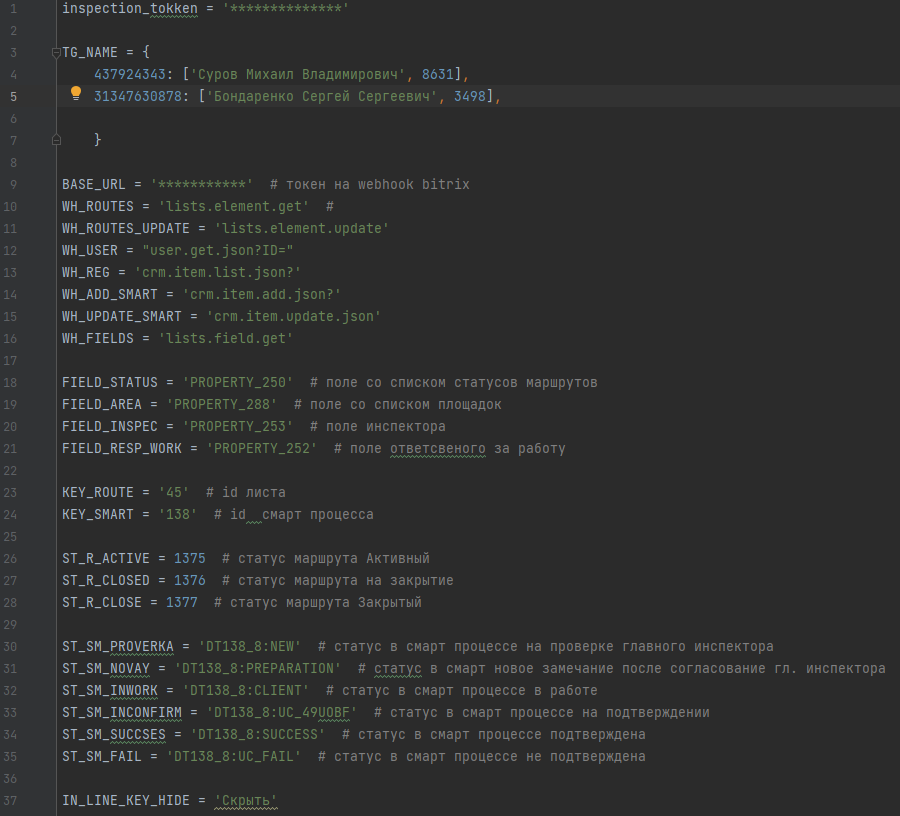
### 3.4.10 Тесты

Для тестирования проекта я зарегистрировал еще одного бота.

На тестовом сервере Bitrix были созданы такие же таблицы, но Id полей стали другие поэтому был создан модуль **Settings.py в** котором будут записаны контекстные переменные.

С помощью такого подхода мы избавляемся от «Хардкода» т.е. не используем какие то значения явно, а держим их в другом модуле и если нам надо изменить какое-то значение, и оно повторяется в коде не один раз, то мы можем изменить его только в файле Settings.

Поэтому мы можем сделать копию модулей и обозначить их как тестовые и вести разработку там, а какие то релизы версий копировать в модули connector\_bitrix.py, Inspection\_bot.py либо GitLab



# 4. Подведение итогов, оценка результатов и заключение

В наше время мессенджеры находятся на своем пике популярности среди всех возрастных категорий людей. Ежедневно физические и юридические лица используют их в своих целях. От отправки смешных картинок до учетных записей крупных складов. Обмен сообщениями на огромном расстоянии далеко не новая технология, но именно мессенджеры прямо подошли к задаче, без излишеств. Это и цепляет людей. Скорость, надежность и приватность — это самое главное для такого вида работы. Чат-боты стали неотъемлемой частью этой системы. Теперь без них нам тяжело жить. Если потребуется скачать видео с интернет-ресурса, то для этого найдется чат-бот. Отпала необходимость скачивания сотни приложений для каких-либо нужд. Достаточно всего лишь скачать мессенджер и начать работу с нужным ботом. В рамках дипломной работы были выполнены поставленные задачи. Был изучен мессенджер Telegram. Было проведено сравнение мессенджеров друг с другом. При сравнивании был выявлен лучший в плане удобства в работе и наличия документации Telegram Bot API. Выявлены их плюсы и минусы, а также сделаны некоторые выводы при разработке «Telegram бот по инспекции». Было проведено сравнение средств разработки, в следствии которого была выбрана платформа Python. Таким образом результатом дипломной работы является реализованный чат-бот для платформы telegram для помощи инспекторам по обходу площадок и фиксировании замечаний, а также помощь в консолидировании информации и подготовке отчетности.

Одним из главных достижений проекта является умение правильно использовать различные инструменты и технологии (Bitrix Telegram) для достижения поставленных целей. Кроме того, были применены современные подходы и методологии в разработке, что позволило получить высокое качество и удобство использования приложения.

Полученные практические знания помогут внести дополнительные улучшения к телеграммам боту, например добавить еще одну ветку по Инспекции по техники безопасности, сделать Меню выбора через Inline кнопки