Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчёт по курсовой работе

Дисциплина: Базы данных

Выполнил студент гр. 3530901/70203 _____ А.А. Ворошилов
(подпись)

Преподаватель _____ А.В. Мяснов
(подпись)

“___”_____ 2021 г.

Санкт-Петербург
2021

Основная информация

В качестве курсового проекта было выбрано реализовать бота в мессенджере Telegram, реализующего способ взаимодействия пользователя с базой данных.

Функции бота (связанные с БД):

1. Добавление агентов, операторов, предметов, личных дел сотрудников;
2. Редактирование личного дела;
3. Удаление предметов, агентов, операторов;
4. Вывод результатов с разными опциями;
5. Вывод выбора пользователя, который обновляется с изменениями в базе данных;

Был выбран язык python и драйвер psycopg2.

Приложение использует модуль config.py для чтения параметров соединения с базой данных. Для чтения данных о соединении, а также для общения программы с ботом, созданным на сервере Telegram, по токену ,используется файл config.ini. Пример файла

config.ini: (токен и пароль частично скрыты)

```
[bot]
API_TOKEN = 1875810409:AAFji5gsMmoj2krmRZly-##############

[postgres]
dbname = cursov_db
user = mermaider
password = qo#######
```

Для работы с Telegram используется API pyTelegramBotAPI v.3.7.7.

Полный код проекта по ссылке:
https://gitlab.icc.spbstu.ru/mermaider/db-spring-2020-spybase/-/tree/master/cursov_db

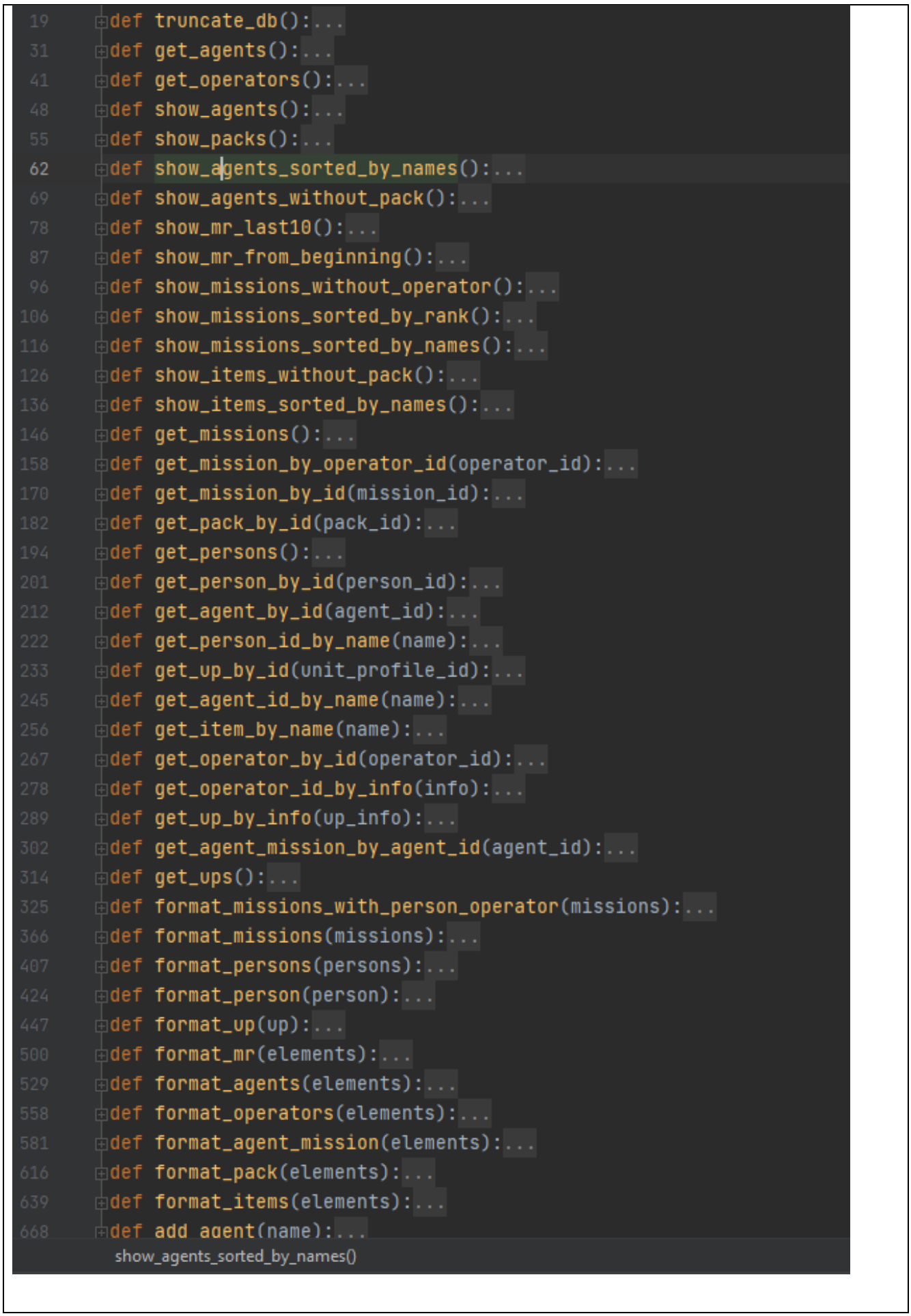
Функциональность бота реализована в двух файлах:

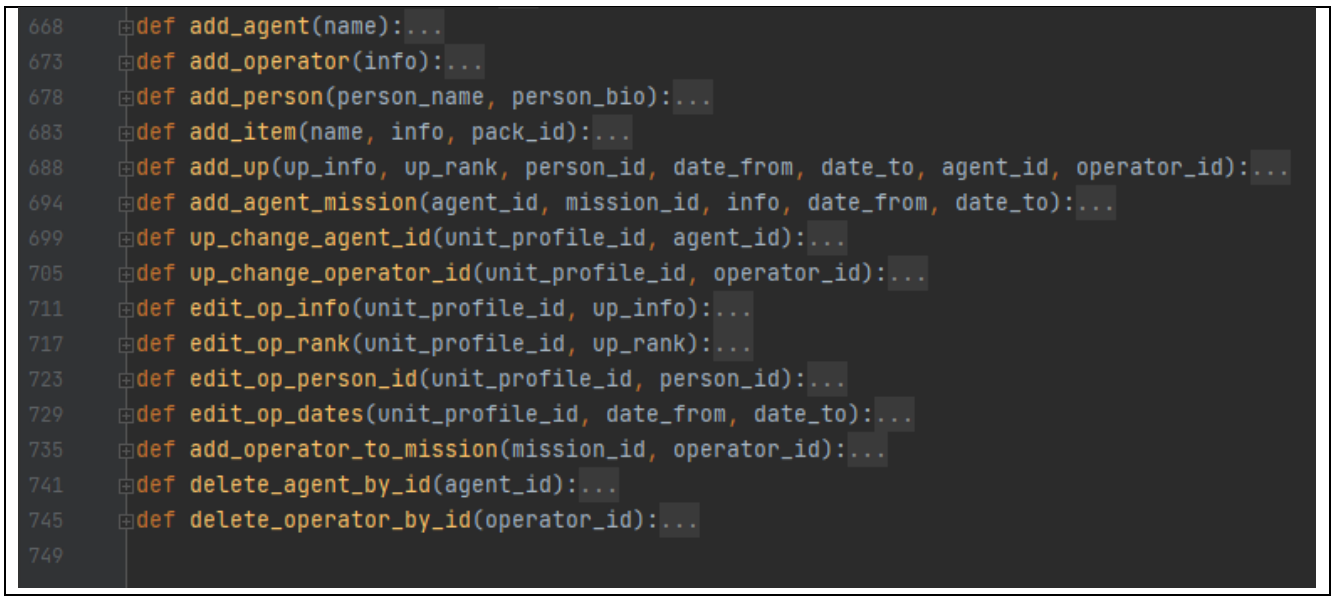
- db_interface.py;
- main.py.

db_interface.py.

Файл, описывающий только интерфейс взаимодействия с базой данных, ничего не знающий о боте, поэтому может быть использован практически в любом приложении, независимо ни от чего. Создание соединения:

Свёрнутые функции, составляющие интерфейс взаимодействия с БД (код слишком велик для полной вставки):


def truncate_db():...
def get_agents():...
def get_operators():...
def show_agents():...
def show_packs():...
def show_agents_sorted_by_names():...
def show_agents_without_pack():...
def show_mr_last10():...
def show_mr_from_beginning():...
def show_missions_without_operator():...
def show_missions_sorted_by_rank():...
def show_missions_sorted_by_names():...
def show_items_without_pack():...
def show_items_sorted_by_names():...
def get_missions():...
def get_mission_by_operator_id(operator_id):...
def get_mission_by_id(mission_id):...
def get_pack_by_id(pack_id):...
def get_persons():...
def get_person_by_id(person_id):...
def get_agent_by_id(agent_id):...
def get_person_id_by_name(name):...
def get_up_by_id(unit_profile_id):...
def get_agent_id_by_name(name):...
def get_item_by_name(name):...
def get_operator_by_id(operator_id):...
def get_operator_id_by_info(info):...
def get_up_by_info(up_info):...
def get_agent_mission_by_agent_id(agent_id):...
def get_ups():...
def format_missions_with_person_operator(missions):...
def format_missions(missions):...
def format_persons(persons):...
def format_person(person):...
def format_up(up):...
def format_mr(elements):...
def format_agents(elements):...
def format_operators(elements):...
def format_agent_mission(elements):...
def format_pack(elements):...
def format_items(elements):...
def add_agent(name):...
show_agents_sorted_by_names()



В этом наборе функций присутствуют как непосредственно обращающиеся к БД для какого-то вывода, так и методы, форматирующие стандартные ответы, полученные с использованием psycopg2в более пригодный для чтения формат.

Сокращения в названиях:

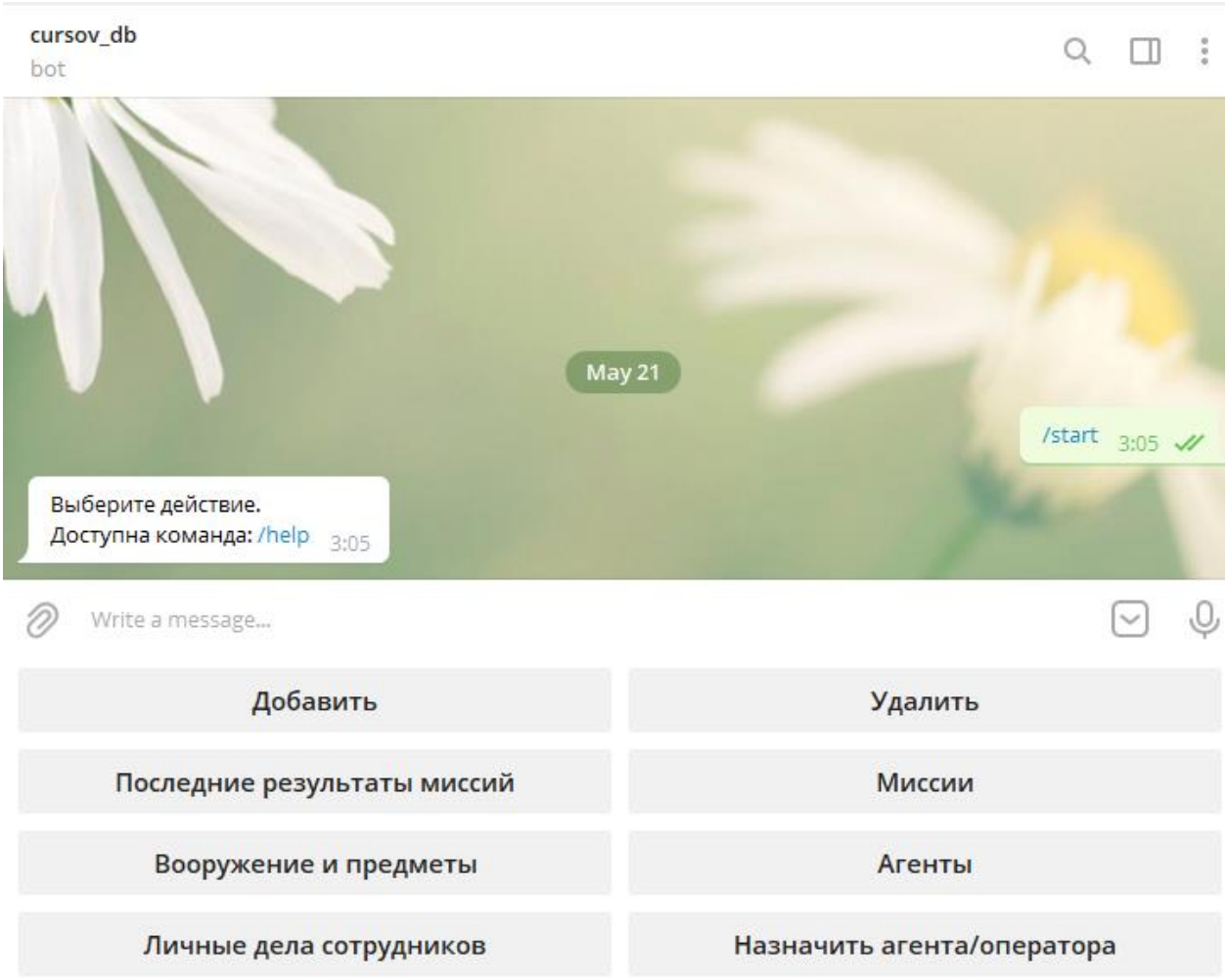
- **mr** -> mission_result (таблица);
- **up/ups** или **op** -> unit_profile (таблица);
- **format** -> функция форматирования пришедшего ответа от базы данных;
- **get** -> в функции происходит получение определённого столбца таблицы БД(запросы типа SELECT);
- **show** -> показ всех значений всех стобцов данной таблицы (запросы типа SELECT);
- **add** -> добавление данных в таблицу (содержит запрос с ключевым словом INSERT);
- **edit/change** -> редактирование строки таблицы (содержит запрос с UPDATE);
- **delete** -> содержит запрос с ключевым словом DELETE.

В остальном, названия функций говорят сами о себе и не нуждаются в дополнительной расшифровке.

Например, функция “**def get_item_by_name(name)”** будет означать SELECT из таблицы item по имени name (с помощью WHERE).

Работа с ботом

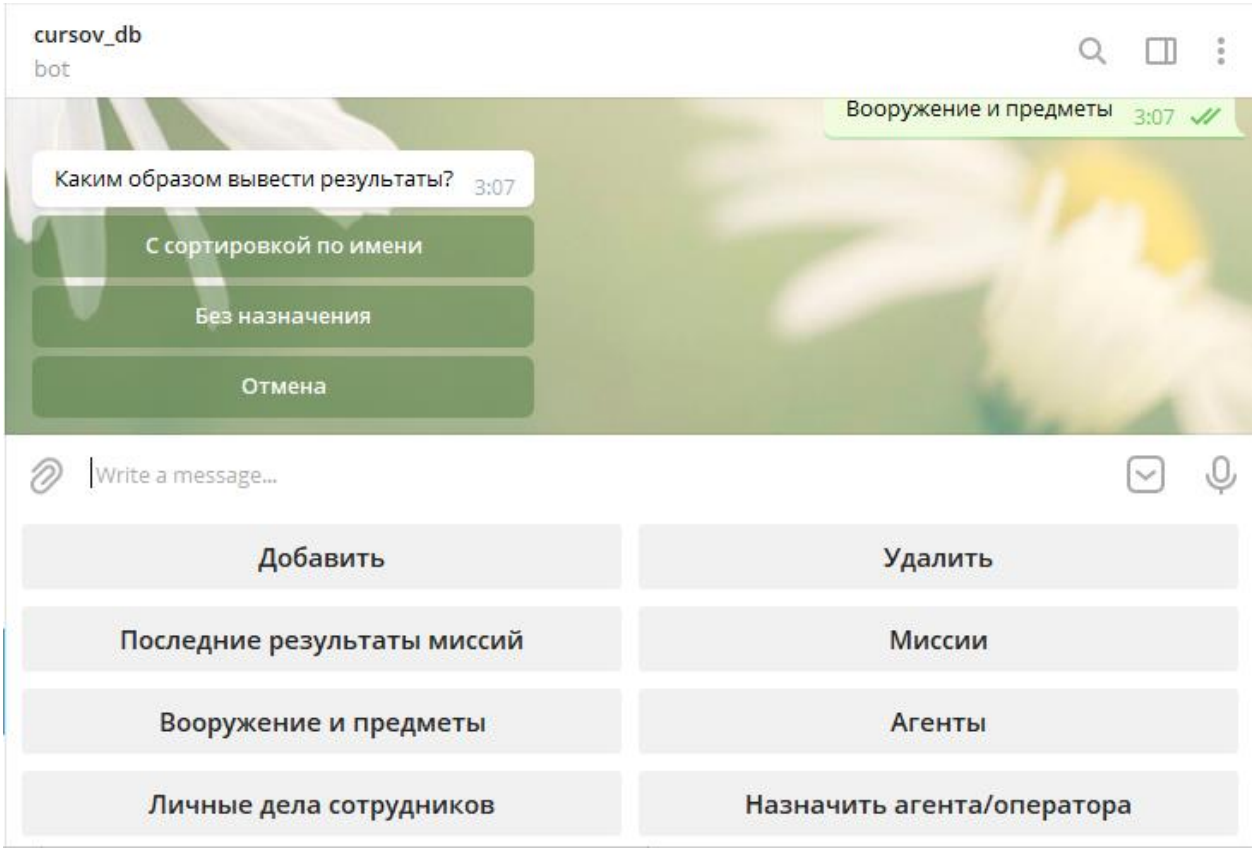
Для просмотра всех возможностей нужно ввести команду /start, и пользователь увидит меню кнопок, благодаря котором возможно взаимодействие с ботом.



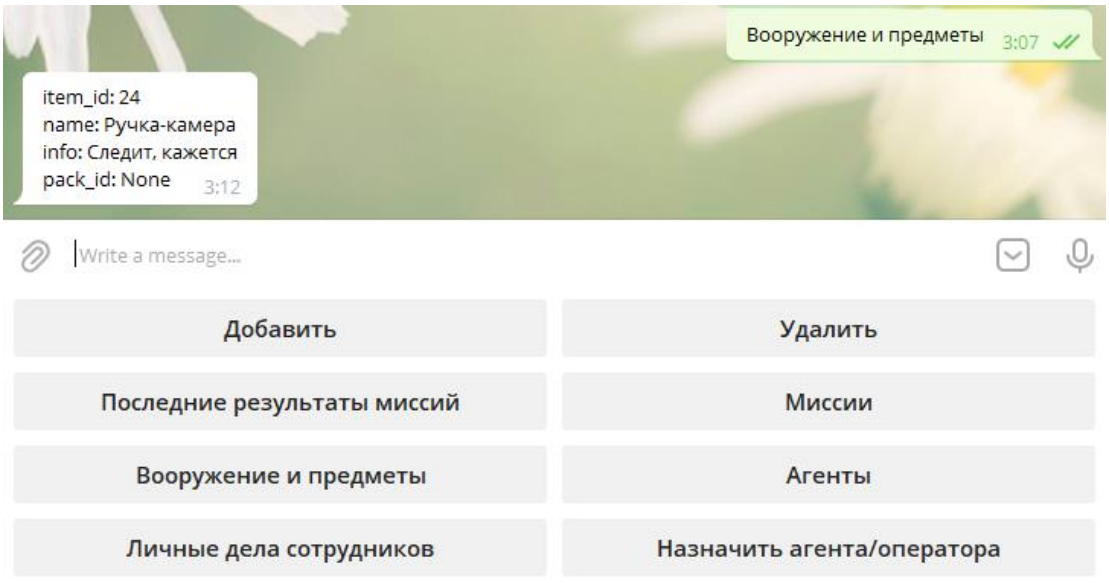
Кнопки на этой клавиатуре отправляют боту сообщение с указанным на кнопке текстом, которое бот примет по регулярному выражению и корректно обработает. В main.py такие обработчики приходящих сообщений называются Handler’ами.

Выбрав кнопку запроса на список “Вооружений и предметов” мы получим ответ с новым видом уже вложенных кнопок. В отличии от типа кнопок, которые снизу и отправляют текст в общий чат, эти вложенные кнопки отправляют боту callback’и, которые обрабатываются немного иначе, но имеют много общего с главными кнопками.

В данной работе реализовано множество callback’ов, поэтому и вложенных кнопок будет встречаться тоже много, потому что они являются удобным способом для взаимодействия пользователя с ботом.

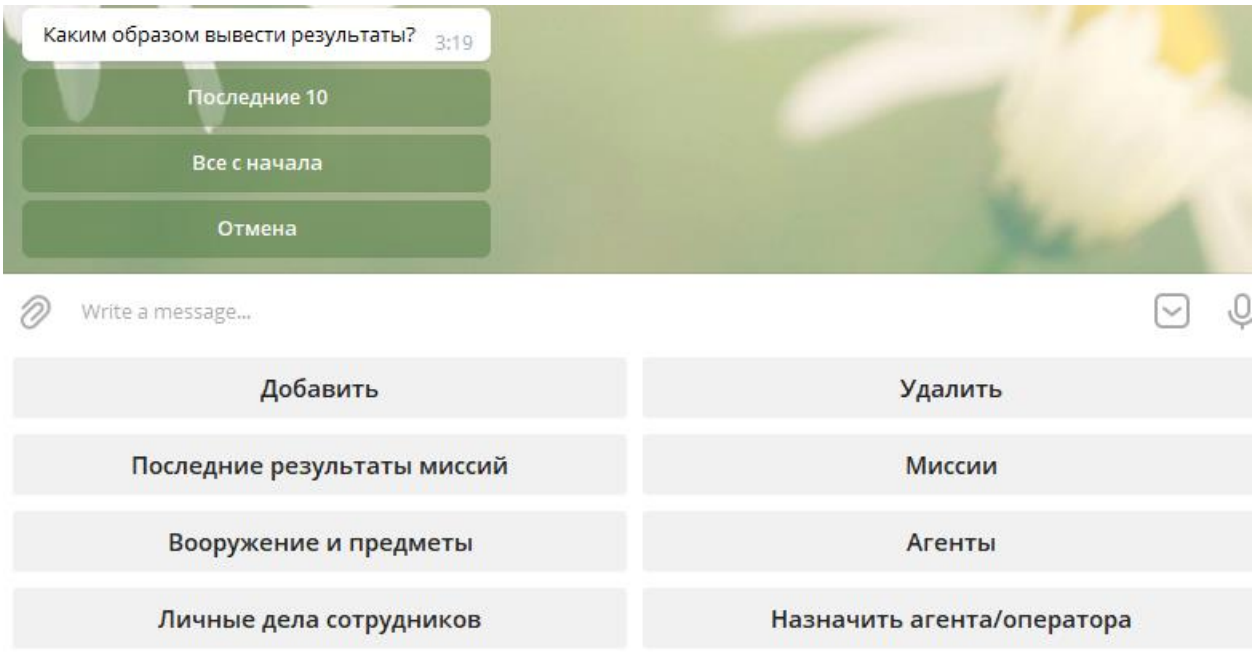


Выберем вложенную кнопку “Без назначения”, тогда соответствующий этой кнопки колбэк будет задействован, а в него вложена строка, которая вызывает из модуля db_interface.py соответствующий метод “def show_items_without_pack()”:

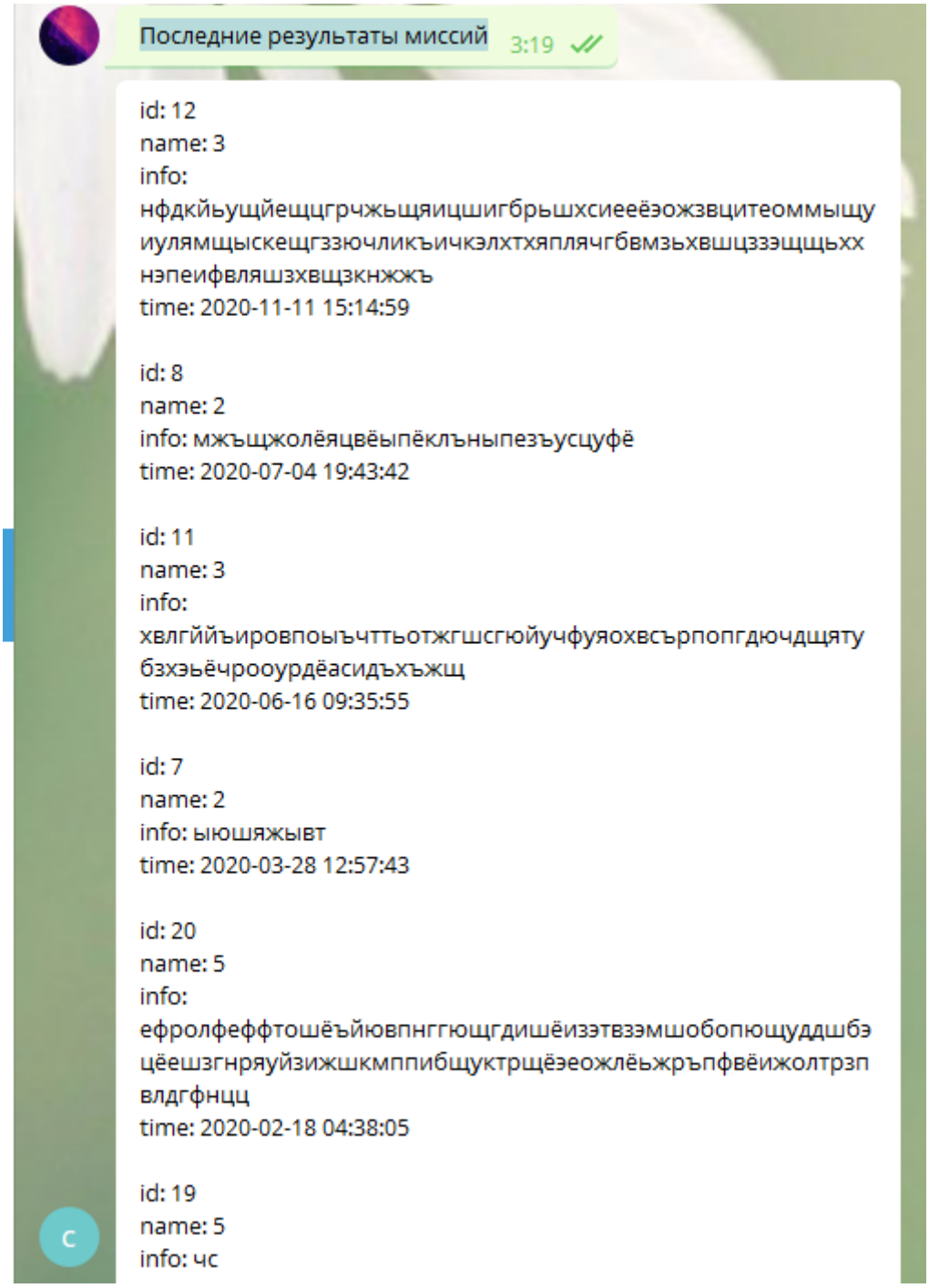


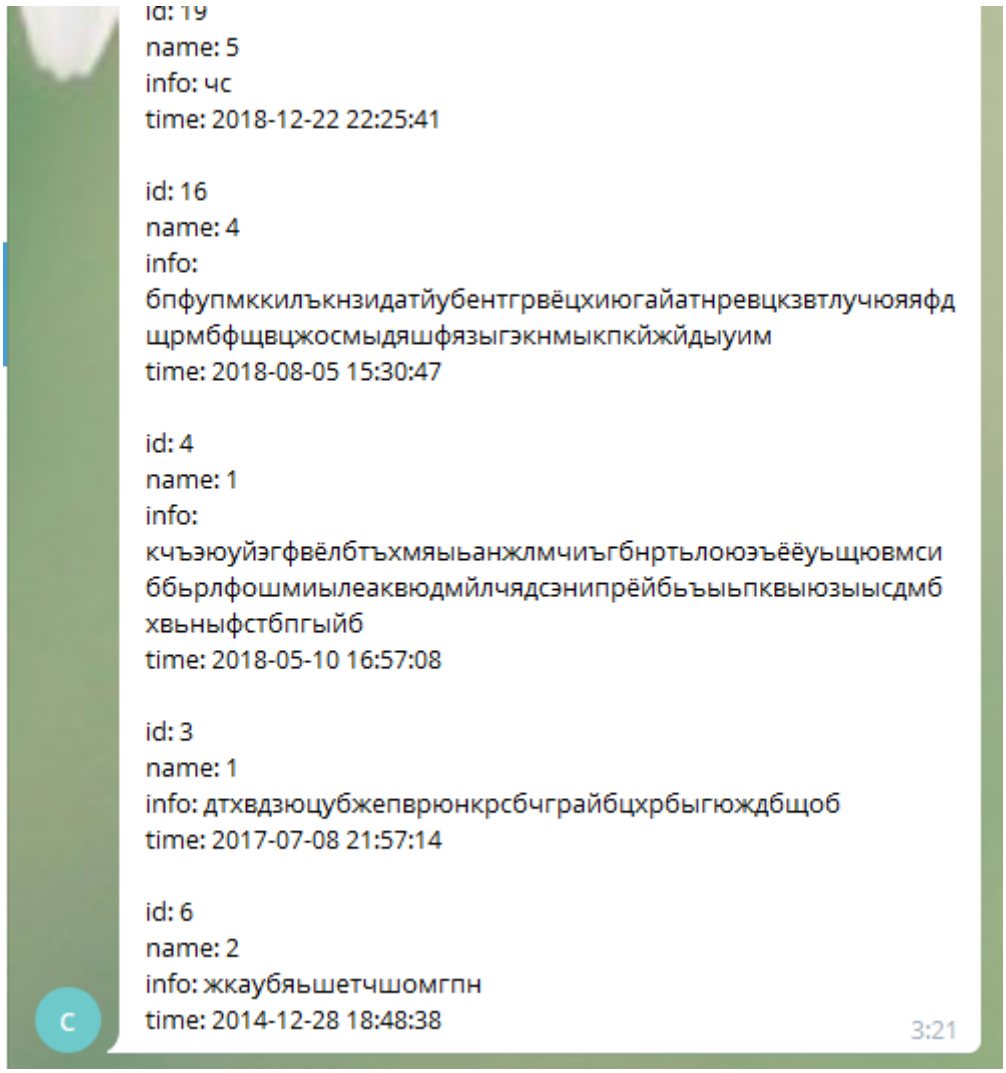
Вложенные кнопки затем исчезают за ненадобностью, как и неактуальное сообщение с предложением опций вывода, оставляя отправленное ботом сообщение с ответом на выбор. Действительно, выведенный ответ является предметом без назначения, так как pack_id = None.

Выбор кнопки “Последние результаты миссий” в очередной раз предложит вложенные опции, которые ведут к соответствующим по смыслам запросам к базе данных. В любом случае оказывается, что любой запрос на вывод - это всё тот же SELECT.



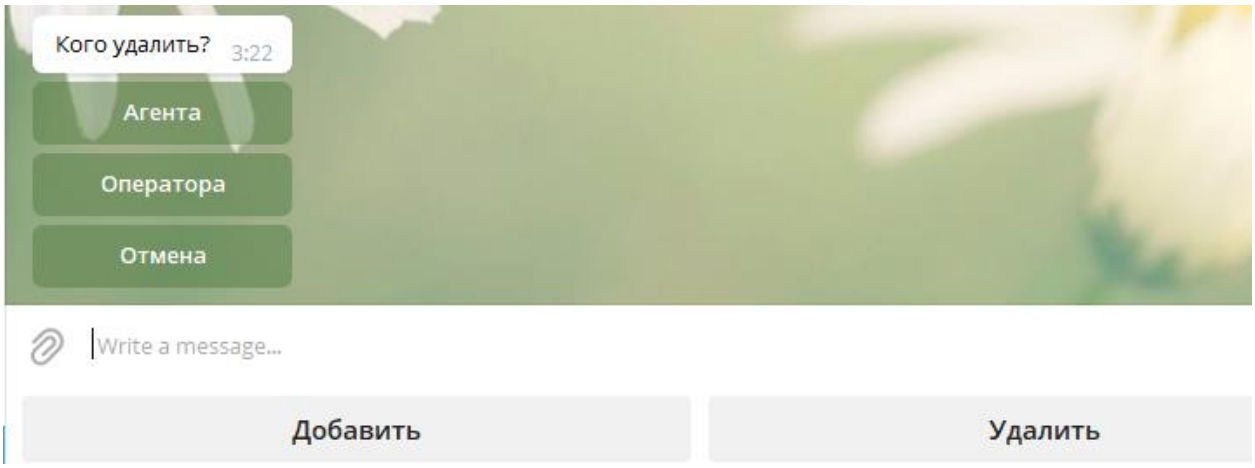
Выберем последние 10:


Последние результаты миссий 3:19
id: 12
name: 3
info:
нфдкйьущйещцгрчжьщяицшигбрьшхсиееёэожзвцитеоммыщу
иулямщыскещгззючликъичкэлхтхяплячгбвмзьхвшцззэщщьхх
нэпеифвляшзхвщзкнжжъ
time: 2020-11-11 15:14:59
id: 8
name: 2
info: мжъщжолёяцвёыпёклъныпезъусцуфё
time: 2020-07-04 19:43:42
id: 11
name: 3
info:
хвлгййъировпоыъчттьотжгшсгюйучфуяохвсърпопгдючдщяту
бзхэьёчрооурдёасидъхъжщ
time: 2020-06-16 09:35:55
id: 7
name: 2
info: ыюшяжывт
time: 2020-03-28 12:57:43
id: 20
name: 5
info:
ефролфеффтошёъйювпнггющгдишёизэтвзэмшобопющуддшбэ
цёешзгнряуйзижшкмппибщуктрщёэеожлёьжръпфвёижолтрзп
влдгфнцц
time: 2020-02-18 04:38:05
id: 19
name: 5
info: чс

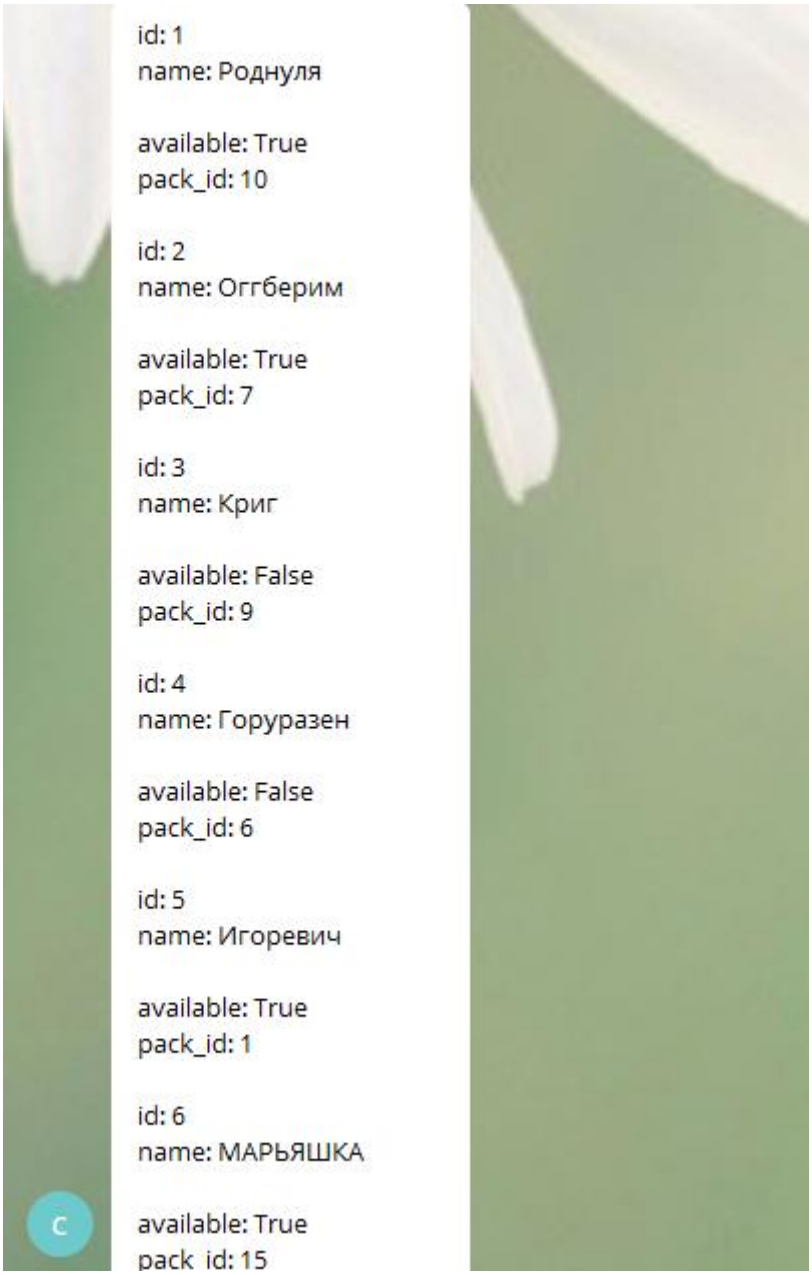


Будут выведены всего 10 строк из mission_result, при этом отсортированные по столбцу time в порядке убывания.

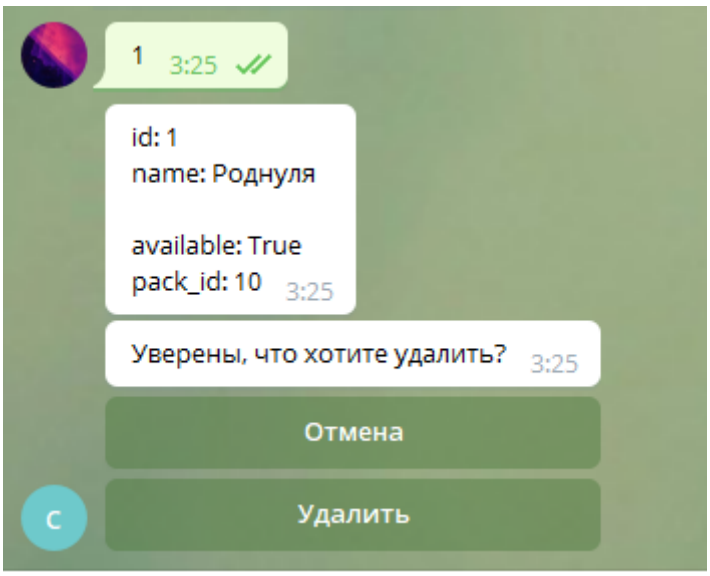
Нажатие главной кнопки удалить так же предложит свои опции:



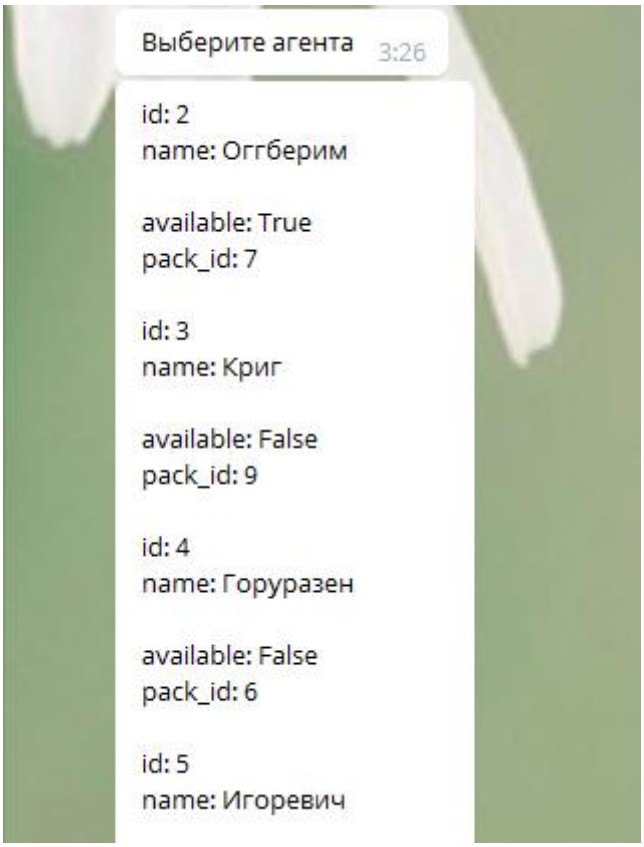
Удалим агента:



В программе предусмотрен вывод опций выбора перед ожиданием принятия решений от пользователя. Удалим агента с id = 1 (Роднуля):

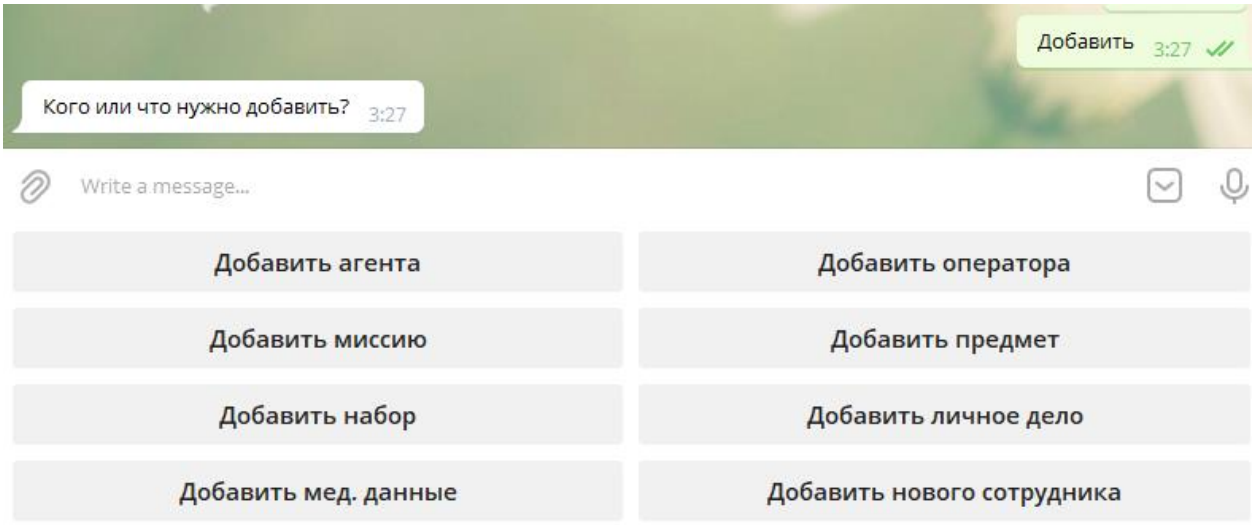


Подтверждаем и убеждаемся, что запрос DELETE в соответствующем callback’е сработал:



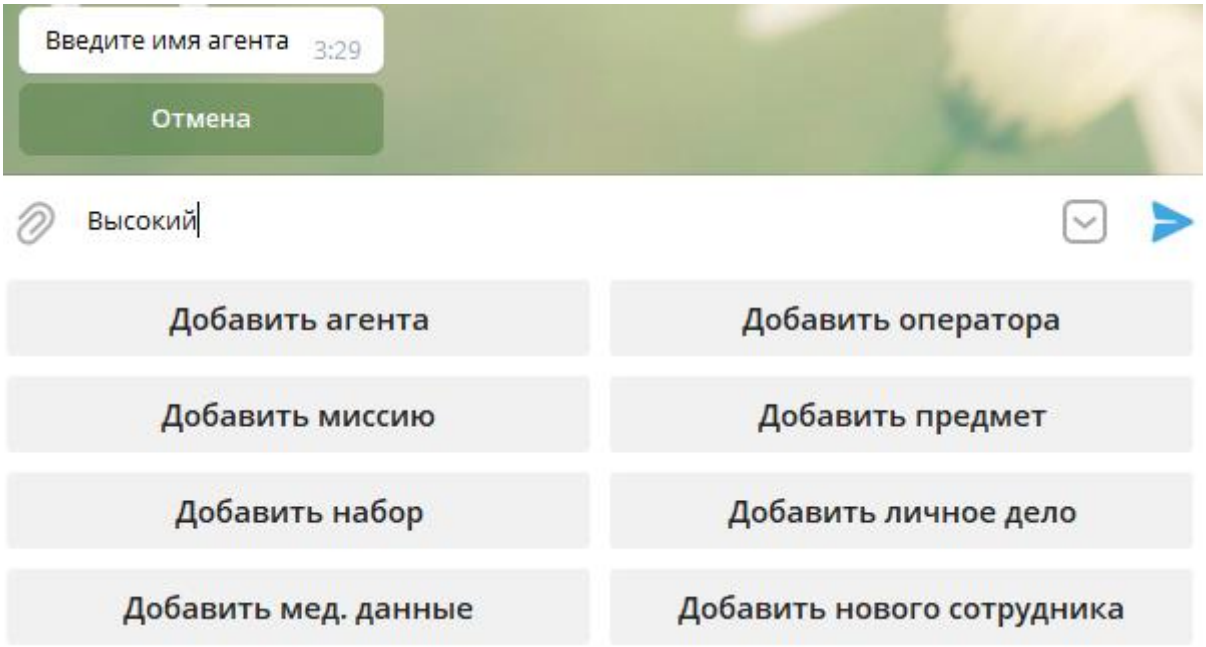
Агента нет.

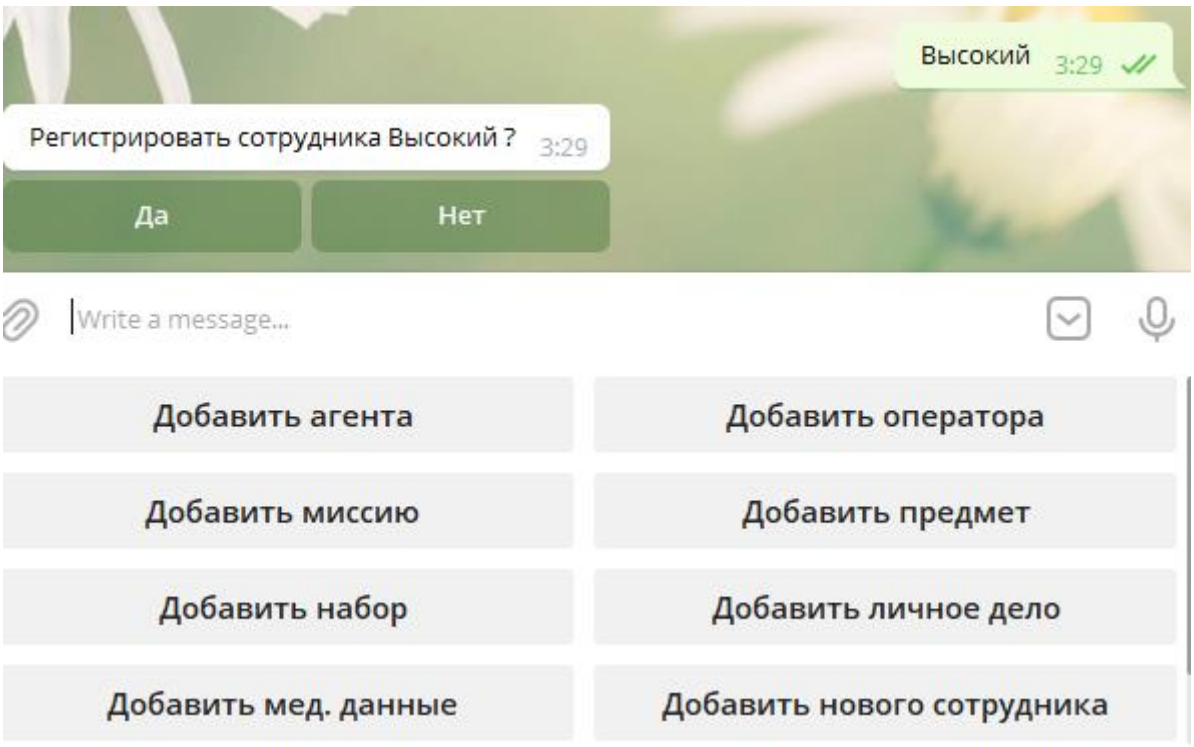
Попробуем добавить агента нажатием кнопки Добавить:



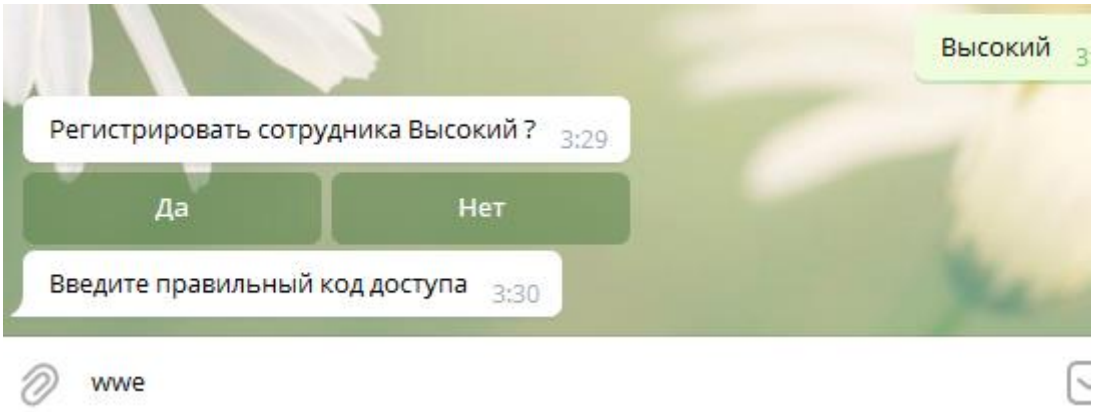
Главные кнопки изменятся. В новом наборе кнопок при пролистывании вниз доступна кнопка возврата обратно к первичному набору кнопок (главное меню).

Вводим имя агента:

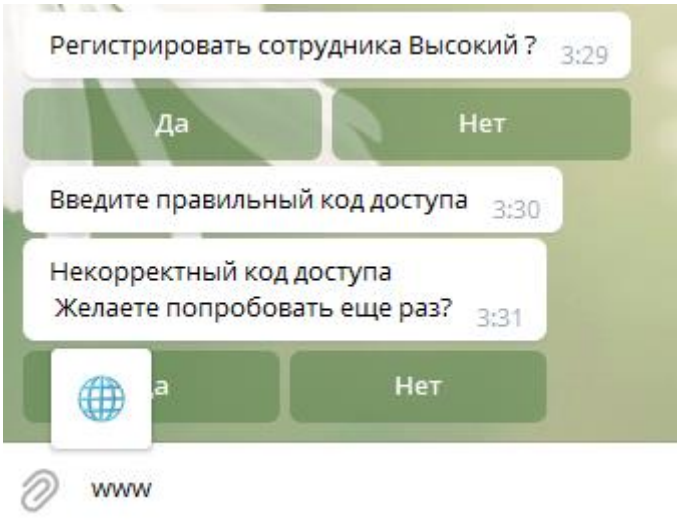


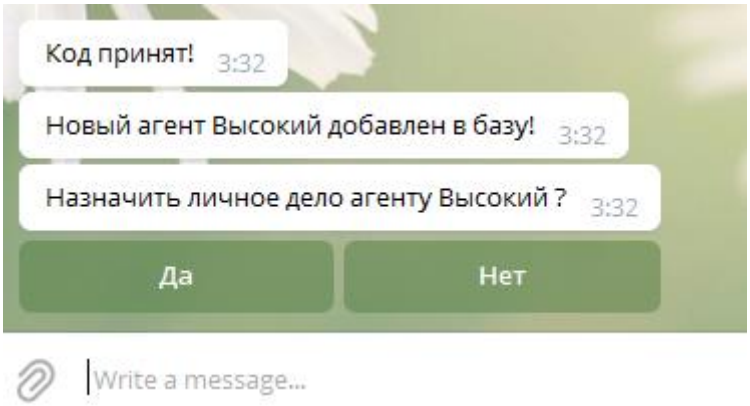


Подтверждаем и вводим пароль:

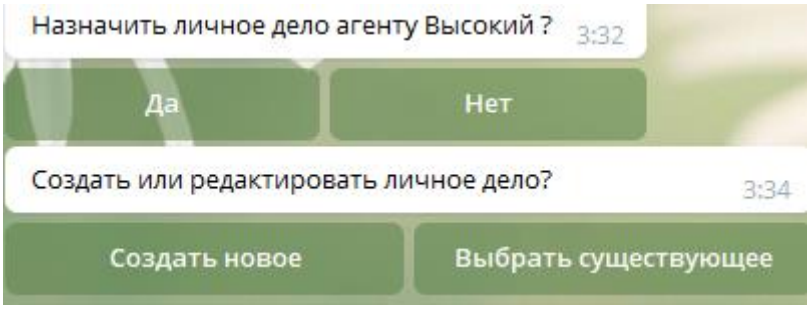


Пароль оказывается неверным, вспоминаем верную комбинацию и колбэк с соответствующим запросом на INSERT к базе данных подтверждается, о чем сообщает сам бот:

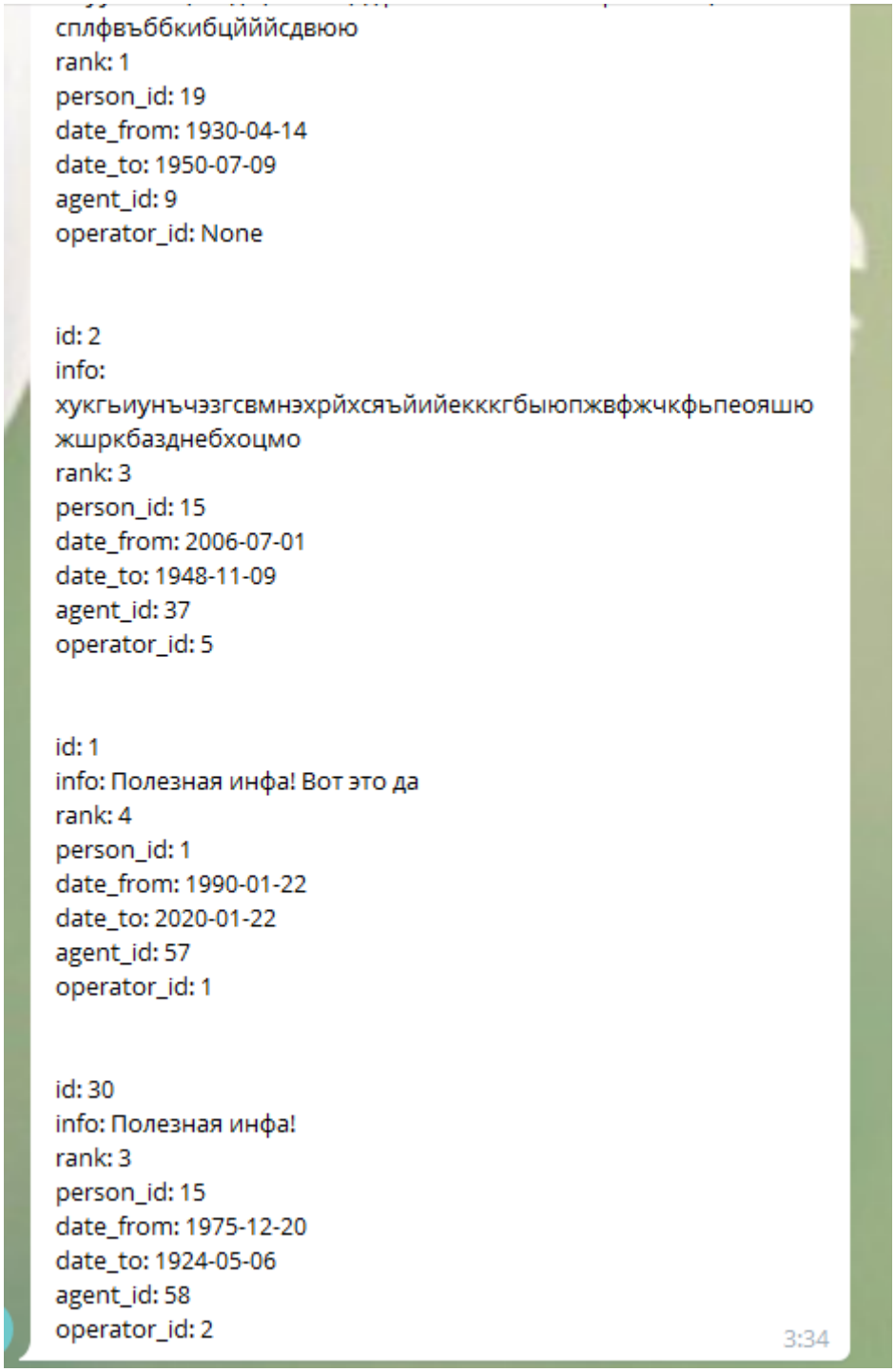




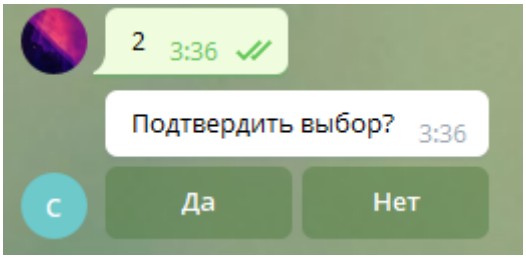
От нового предложения к дальнейшей работе можно отказаться, но в этот раз согласимся проследовать дальше, чтобы либо создать новую строку unit_profile в базе данных, либо редактировать старую, но при выборе существующего личного дела значение в столбце agent_id будет заменено на agent_id нового агента с именем “Высокий”:

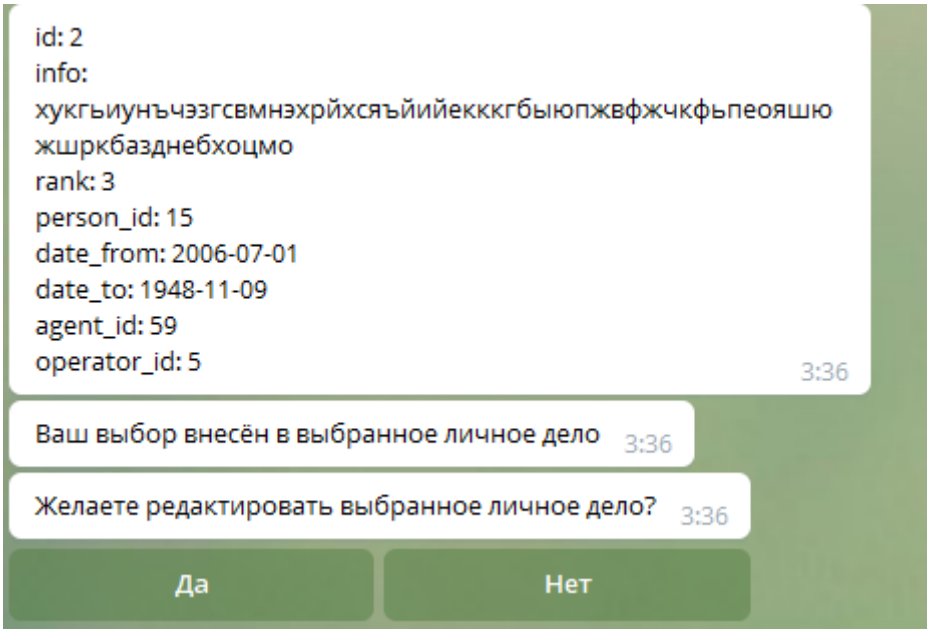


Выбираем существующее дело для демонстрации вида колбэка с UPDATE-запросом.



Выберем дело с ID = 2.
Запомним, что первоначальный agent_id в ней равен 37.

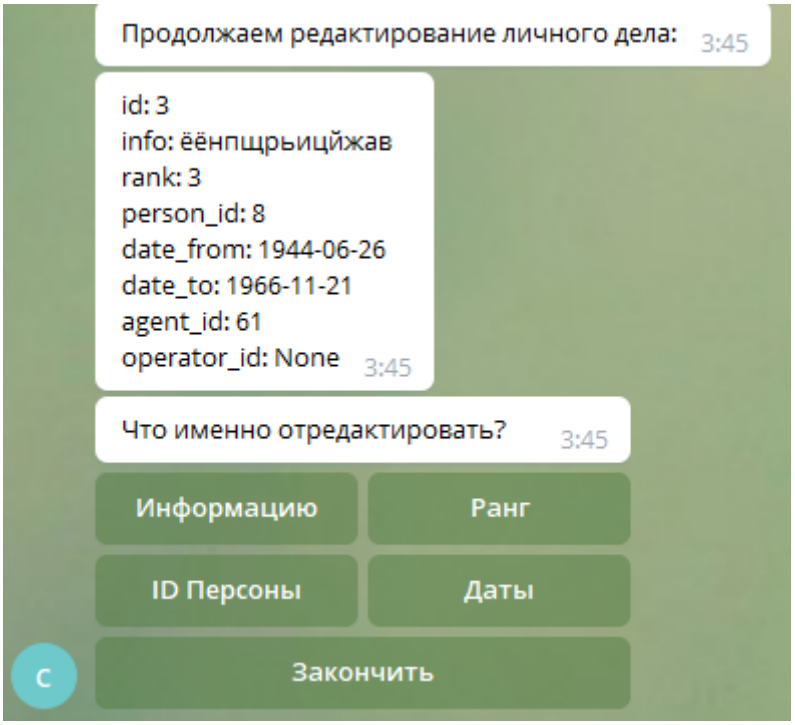




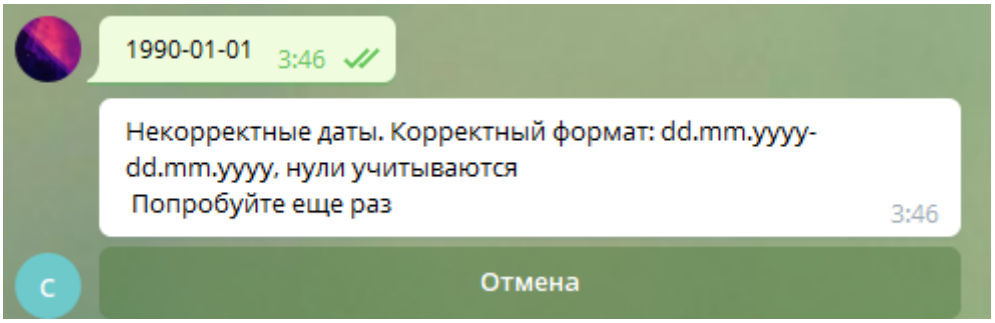
Выбор внесён в базу. Обратим внимание, что новый agent_id у выбранной строки unit_profile стал равен 59, хотя до этого было изначально 37.

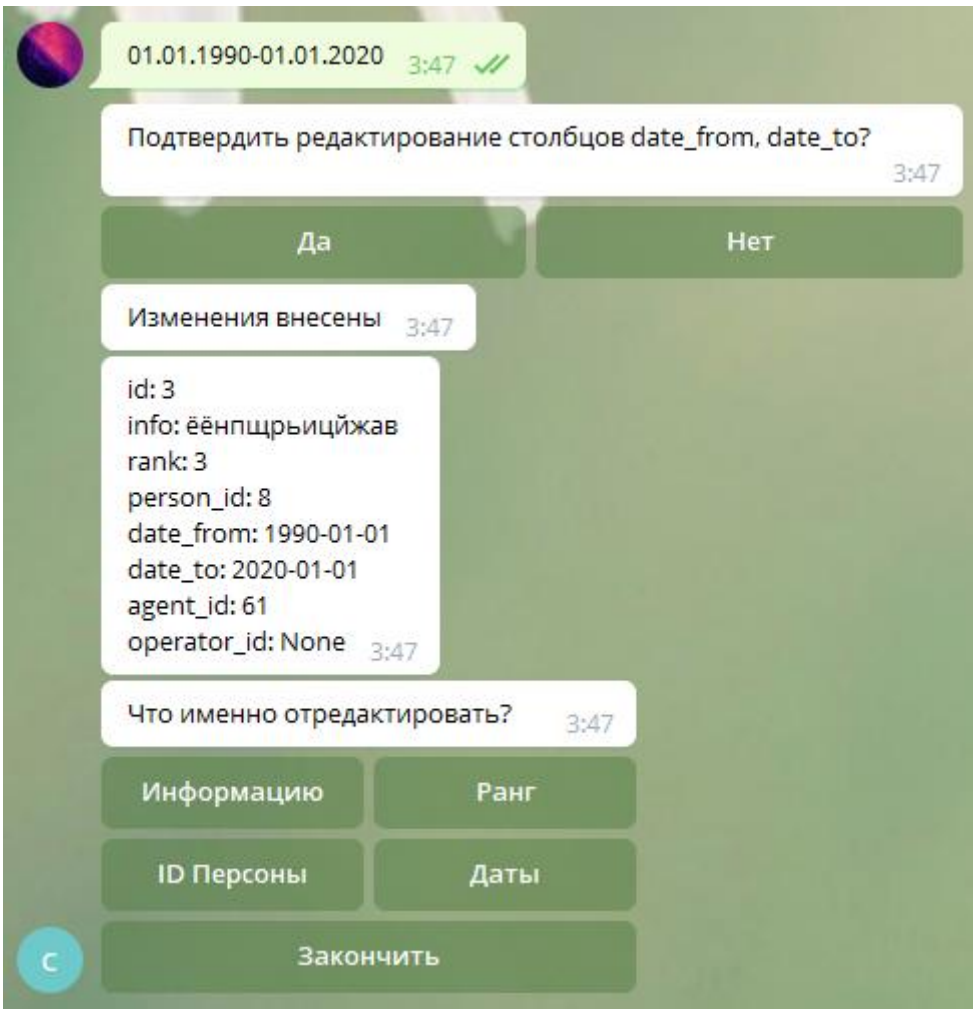
Когда мы вводили имя агента, программа запомнила ввод пользователя в глобальной переменной. Затем, после успешного добавления агента в базу, программа вызвала функцию в db_interface.py. под названием get_agent_id_by_name(name). После этого программа знала не только введенное нами имя Высокий, но и полученный ID к этому агенту, равный 59.

Лично дело также можно редактировать после этого. Создадим нового агента и выберем дело с ID = 3

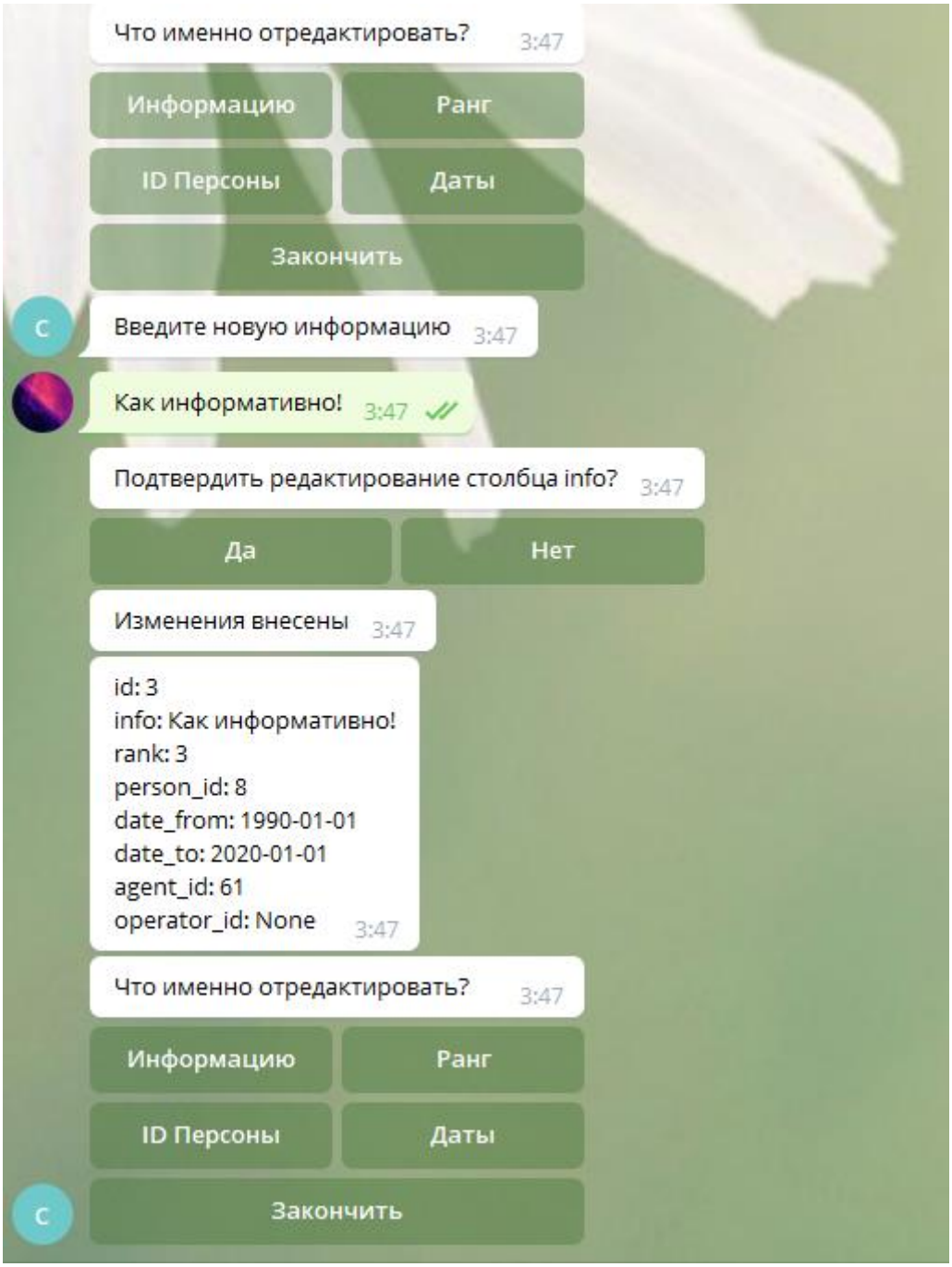


Менять можно любое из доступных полей. Выберем даты, назначим их с 1990-01-01 по 2020-01-01:





Также изменим информацию:



Все вложенные кнопки, которые предлагают любое редактирование задействуют свои определённые колбэки, которые в свою очередь вызывают приписанные им функции из db_interface.py с запросами типа UPDATE.

Остальной функционал похож на указанные ранее операции и является объёмным.

Итог работы

Изучен процесс регистрации и настройки Telegram-бота. Рассмотрена API pyTelegramBotAPI v 3.7.7.

Написано приложение в виде бота для мессенджера Telegram, реализующее надстройку над реализованным интерфейсом взаимодействия с БД, который может быть использован и в любом другом приложении, которое сможет брать любые методы из него.

Приложение реализует широкий ряд запросов к базе данных и обеспечивает при этом удобное взаимодействие пользователя с этой БД.