Защищено:
Папин А.В..

"__" _____2022 г.

Демонстрация:
Папин А.В..

"__"_____2022 г.

Отчет по лабораторной работе №6 по курсу базовые компоненты интернет-технологий (БКИТ)

Тема работы: "Разработка бота на основе конечного автомата для Telegram с использованием языка Python."

19
(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-54Б _____
Папин Алексей

(подпись)

Гапанюк Ю.Е.

"__"_____2022 г.

Москва, МГТУ - 2022

СОДЕРЖАНИЕ ОТЧЕТА

1. Цель лабораторной работы

Изучение разработки ботов в Telegram.

2. Описание задания.

Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.

1. Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

3. Листинг программы:

3.1.config.py

```
token = ''
```

3.2.calculate_arifmetic.py

```
4.
   class the_simplest_mathematical_calculator(object):
       '''
       def __init__(self):
           self.math_calculation = ''
           self.math_calculation_list = []
           self.list_enumeration_sign = []
           self.type_error = ''
           self.result = None

       '''
       def __init__(self, math_calculate):
           self.math_calculation = math_calculate
           self.math_calculation_list =
   self.delete_space_into_list(math_calculate)
           self.list_enumeration_sign =
   self.enumeration_sign(self.math_calculation_list)

           self.type_error = None

           for sgin in self.list_enumeration_sign:
               self.arifmetic(sgin, self.math_calculation_list)

           if(self.type_error == None):
               self.result = float(self.math_calculation_list[0])


       # Преобразование строкого типа в list
       def delete_space_into_list(self, str_calculate):
           new_str = []
           str_value = ''
           for i in str_calculate:
               if(i != ' '):
                   str_value += i
               else:
                   new_str.append(str_value)
                   str_value = ''

           new_str.append(str_value)

           return new_str


       # Расстановка приоритета операции
       def enumeration_sign(self, list_str):
           count_list = []
           for i in list_str:
```

```
            if ('*' == i): count_list.append(i)
            if ('/' == i): count_list.append(i)
            if ('+' == i): count_list.append(i)
            if ('-' == i): count_list.append(i)

        count_list = self.prioritet(count_list)

        return count_list

    # Поддержка функции по расстановку приоритета операции
    def prioritet(self, list_str):
        new_list = []
        size = len(list_str)
        count = 0
        while (size != 0):
            if('*' in list_str or '/' in list_str):
                for i in list_str:
                    if(i == '*' or i == '/'):
                        new_list.append(i)
                size -= 1
            if('+' in list_str or '-' in list_str):
                for i in list_str:
                    if(i == '+' or i == '-'):
                        new_list.append(i)
                size -= 1

        return new_list

    # Арифметические операции
    def arifmetic(self, sign, list):
        result = None
        if (sign in list):
            for i in range(1, len(list) - 1):
                try:
                    if(list[i] == sign):
                        if(sign == '*'): result = float(list[i - 1]) *
float(list[i + 1])
                        elif(sign == '/'): result = float(list[i - 1]) /
float(list[i + 1])
                        elif (sign == '+'): result = float(list[i - 1]) +
float(list[i + 1])
                        elif (sign == '-'): result = float(list[i - 1]) -
float(list[i + 1])

                        list[i] = result
                        del list[i - 1: i]
                        del list[i: i + 1]

                # Деление на 0
                except ZeroDivisionError:
                    self.type_error = 'Division by 0'
                    self.result = 'inf'

                # Граница вне диапазона
                except:
```

```
                    return result

    def calculate(self, math_calculate):
        self.math_calculation = math_calculate
        self.math_calculation_list =
self.delete_space_into_list(math_calculate)
        self.list_enumeration_sign =
self.enumeration_sign(self.math_calculation_list)

        self.type_error = None

        for sgin in self.list_enumeration_sign:
            self.arifmetic(sgin, self.math_calculation_list)

        if(self.type_error == None):
            self.result = float(self.math_calculation_list[0])

        return self
```

4.1.calculate_bot.py

```
5. import config
import telebot
from telebot import types
import random

from calculate_arifmetic import the_simplest_mathematical_calculator as
smc
from json_function import merge_data, delete_data_for_id_user,
load_data_for_id_user
from work_with_calculate import generate_value

# Создание бота
bot = telebot.TeleBot(config.token)

HELP = '''
/start - Меню переключателя
/calculate - Калькуляторный бот, способный вычислять простейшие
арифметические операции
/get_info - Просмотр историю вычисления с БД
/clean - Очистка история вычисления
/random - Генерация случайных вычислений
/photo - Просмотр фото МГТУ им. Н.Э. Баумана
'''


# Справочник
@bot.message_handler(commands=['help'])
def start(message):
    bot.send_message(message.chat.id, HELP)


@bot.message_handler(commands=['start'])
def start(message):
```

```
    markup = types.InlineKeyboardMarkup(row_width=1)
    btn1 = types.InlineKeyboardButton(text="Посчитать",
callback_data='btn1')
    btn2 = types.InlineKeyboardButton(text="Посмотреть историю
вычисления", callback_data='btn2')
    btn3 = types.InlineKeyboardButton(text="Очистить историю вычисления",
callback_data='btn3')
    btn4 = types.InlineKeyboardButton(text="Рандомные вычисления",
callback_data='btn4')
    btn5 = types.InlineKeyboardButton(text="Показать фото МГТУ им. Н.Э.
Баумана", callback_data='btn5')
    markup.add(btn1, btn2, btn3, btn4, btn5)
    bot.send_message(message.chat.id,
                     text=f"Привет, {message.from_user.first_name}! Я
тестовый бот, выберите действия",
                     reply_markup=markup)


# Функция переключателя
@bot.callback_query_handler(func=lambda callback: callback.data)
def check_callback_data(callback):
    # Пользовательский идентификатор
    user_id = str(callback.from_user.id)

    if (callback.data == "btn1"):
        bot.send_message(callback.message.chat.id, 'Калькулятор бот')
        bot.send_message(callback.message.chat.id, 'Напишите в чате
вычисления')

        # Пользовательский идентификатор
        user_id = str(callback.from_user.id)

        @bot.message_handler(content_types=["text"])
        def echo(message):
            value = smc(message.text)
            bot.send_message(message.chat.id, f'Решение: {value.result}')
            data = {
                user_id: [
                    {"id": random.randint(0, 10000),
                     "value": str(message.text),
                     "result": str(value.result)}
                ]
            }
            merge_data(data, str(message.from_user.id))

    elif(callback.data == "btn2"):
        bot.send_message(callback.message.chat.id, 'История вычисления')

        data = load_data_for_id_user(str(user_id))
        if(data == 'Error! There is no such identifier'):
            bot.send_message(callback.message.chat.id, 'База данных
отсутствует')
        else:
            for j in range(len(data) - 1):
                id = data[j]['id']
```

```
                value = data[j]['value']
                result = data[j]['result']
                print_info = f'id: {id}\n{value} = {result}\n\n'
                bot.send_message(callback.message.chat.id, print_info)

    elif(callback.data == "btn3"):
        bot.send_message(callback.message.chat.id, 'Очистка история
вычисления')

        check_error = delete_data_for_id_user(user_id)

        if(check_error != 'Error! There is no such identifier'):
            bot.send_message(callback.message.chat.id, 'Успешно')
        else:
            bot.send_message(callback.message.chat.id, check_error)

    elif(callback.data == "btn4"):
        bot.send_message(callback.message.chat.id, 'Генерация случайных
вычислений')
        generate_value(user_id)
        bot.send_message(callback.message.chat.id, 'Успешно')

    elif(callback.data == "btn5"):
        img = open('bmstu.jpg', 'rb')
        bot.send_photo(callback.message.chat.id, img)
    else:
        bot.send_message(callback.chat.id, 'Нет такой команды. Введите
/help')

# Вычисления
@bot.message_handler(commands=['calculate'])
def start_calculate(message):
    bot.send_message(message.chat.id, 'Калькулятор бот')
    bot.send_message(message.chat.id, 'Напишите в чате вычисления')

    # Пользовательский идентификатор
    user_id = str(message.from_user.id)

    @bot.message_handler(content_types=["text"])
    def echo(message):
        value = smc(message.text)
        bot.send_message(message.chat.id, f'Решение: {value.result}')
        data = {
            user_id: [
                {"id": random.randint(0, 10000),
                 "value": str(message.text),
                 "result": str(value.result)}
            ]
        }
        merge_data(data, str(message.from_user.id))


# Просмотри история вычисления
@bot.message_handler(commands=['get_info'])
def start_get_info(message):
```

```
    bot.send_message(message.chat.id, 'История вычисления')

    # Пользовательский идентификатор
    user_id = str(message.from_user.id)

    data = load_data_for_id_user(str(user_id))

    if (data == 'Error! There is no such identifier'):
        bot.send_message(message.chat.id, 'База данных отсутствует')
    else:
        for j in range(len(data) - 1):
            id = data[j]['id']
            value = data[j]['value']
            result = data[j]['result']
            print_info = f'id: {id}\n{value} = {result}\n\n'
            bot.send_message(message.chat.id, print_info)


@bot.message_handler(commands=['photo'])
def url(message):
    img = open('bmstu.jpg', 'rb')
    bot.send_photo(message.chat.id, img)

@bot.message_handler(commands=['random'])
def url(message):
    bot.send_message(message.chat.id, 'Генерация случайных вычислений')
    # Пользовательский идентификатор
    user_id = str(message.from_user.id)

    generate_value(user_id)

    bot.send_message(message.chat.id, 'Успешно')
@bot.message_handler(commands=['clean'])
def url(message):
    bot.send_message(message.chat.id, 'Очистка история вычисления')

    # Пользовательский идентификатор
    user_id = str(message.from_user.id)

    check_error = delete_data_for_id_user(user_id)

    if (check_error != 'Error! There is no such identifier'):
        bot.send_message(message.chat.id, 'Успешно')
    else:
        bot.send_message(message.chat.id, check_error)


bot.polling(none_stop=True)
```

5.1.json_function.py

```
6. import json


   def write_data(data, title='D:\Python\BKIT\calculate\data'):
       with open(f"{title}.json", "w", encoding="utf-8") as file:
           json.dump(data, file, indent=2, ensure_ascii=False)


   def load_data_all(title="D:\Python\BKIT\calculate\data"):
       with open(f"{title}.json", "r") as file:
           data = json.load(file)
       return data


   def merge_data(data_json, id_user='id_user',
   title="D:\Python\BKIT\calculate\data"):
       # Если файл существует и не пустой
       try:
           with open(f"{title}.json", encoding="utf-8") as file:
               data = json.load(file)
               temp = data[id_user]
               for info_data in data_json[id_user]:
                   y = {
                       'id': info_data['id'],
                       'value': info_data['value'],
                       'result': info_data['result']
                   }
               temp.append(y)
           write_data(data)
       # Если файл не существует
       except:
           write_data(data_json)


   def load_data_for_id_user(id_user, title="D:\Python\BKIT\calculate\data"):
       try:
           with open(f"{title}.json", "r", encoding="utf-8") as file:
               data = json.load(file)
               temp = data[id_user]
               for info_data in data[id_user]:
                   y = {
                       'id': info_data['id'],
                       'value': info_data['value'],
                       'result': info_data['result']
                   }
               temp.append(y)
           return temp
       except:
           return 'Error! There is no such identifier'


   def delete_data_for_id_user(id_user,
   title="D:\Python\BKIT\calculate\data"):
```

```
    try:
        with open(f"{title}.json", encoding="utf-8") as file:
            data = json.load(file)
            new_data = {}
            for id_user_data in data:
                if (id_user != id_user_data):
                    temp = data[id_user_data]
                    new_data = {
                        id_user_data: []
                    }
                    for j in temp:
                        y = {
                            'id': j['id'],
                            'value': j['value'],
                            'result': j['result']
                        }
                        new_data[id_user_data].append(y)
                    temp.append(new_data)
        write_data(new_data)
    except:
        return 'Error! There is no such identifier'
```

6.1.work_with_calculate.py

```
import random

from calculate.json_function import write_data, load_data_all, merge_data,
load_data_for_id_user
from calculate.calculate_arifmetic import
the_simplest_mathematical_calculator as smc


def generate_value(id_user='id_user'):
    arifmetic = ['+', '-', '/', '*']

    af = arifmetic[random.randint(0, 3)]
    gen_id = random.randint(0, 100000)
    v1 = random.randint(0, 1000)
    v2 = random.randint(0, 1000)
    class_calculate = smc(str(v1) + ' ' + str(af) + ' ' + str(v2))

    data = {
        str(id_user): [
            {
                "id": gen_id,
                "value": (str(v1) + ' ' + str(af) + ' ' + str(v2)),
                "result": class_calculate.result
            }
        ]
    }

    merge_data(data, id_user)

def get_info():
```

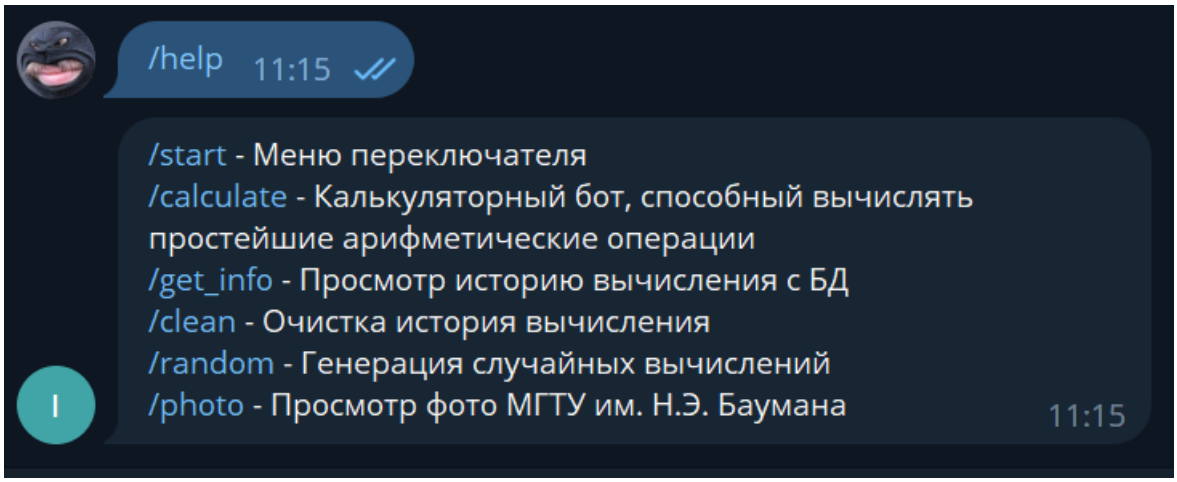
```
    try:
        data = load_data_all()
        return data
    except:
        return 'Файл отсутствует'

def get_info_with_id_user(id_user):
    try:
        data = load_data_for_id_user(id_user)
        return data
    except:
        return 'Файл отсутствует'
```

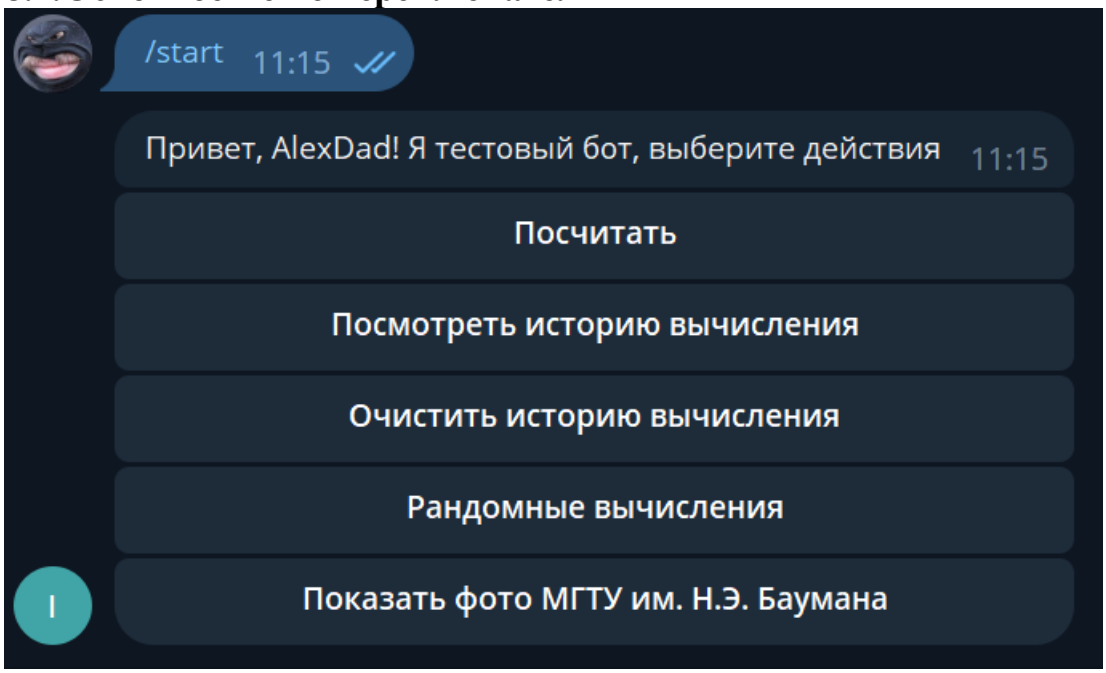
7.1.bmstu.jpg

8. Результаты работы программы в Telegram

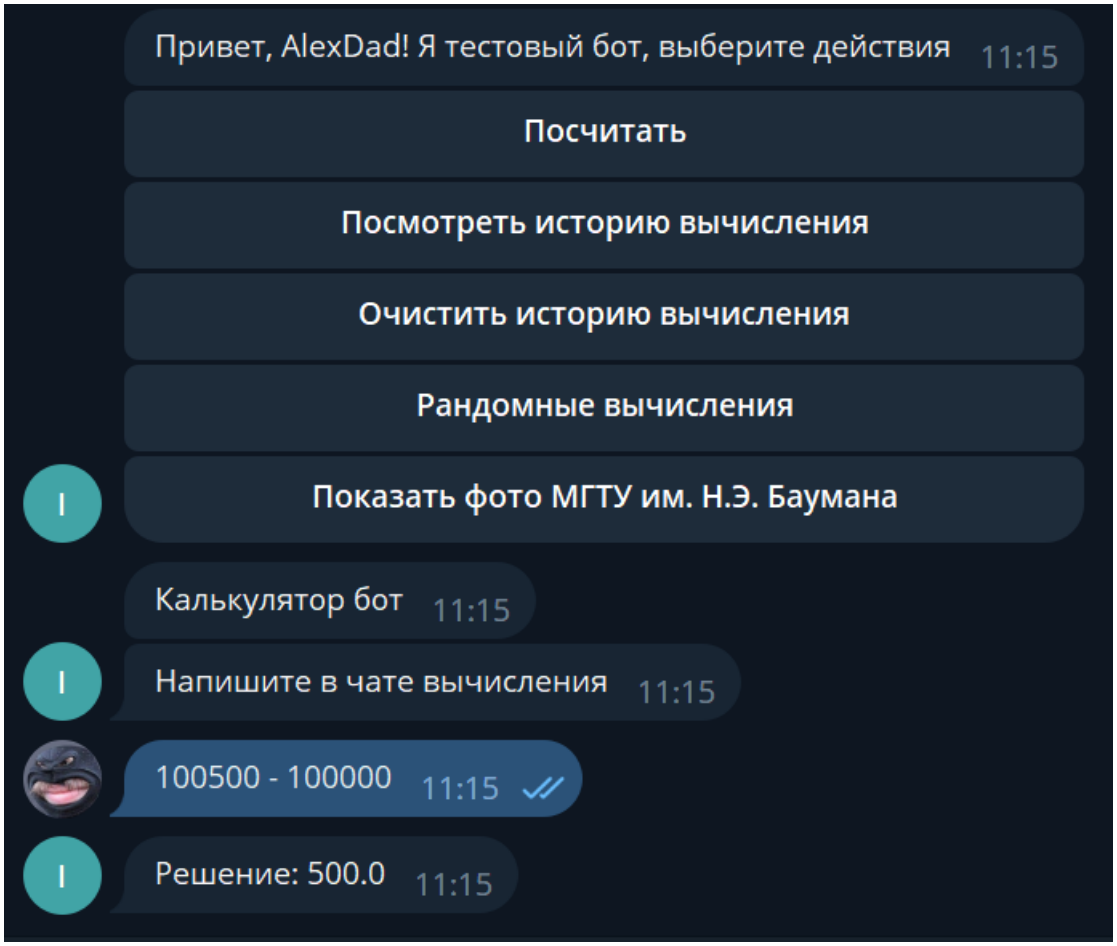
8.1.Получение справочную информацию



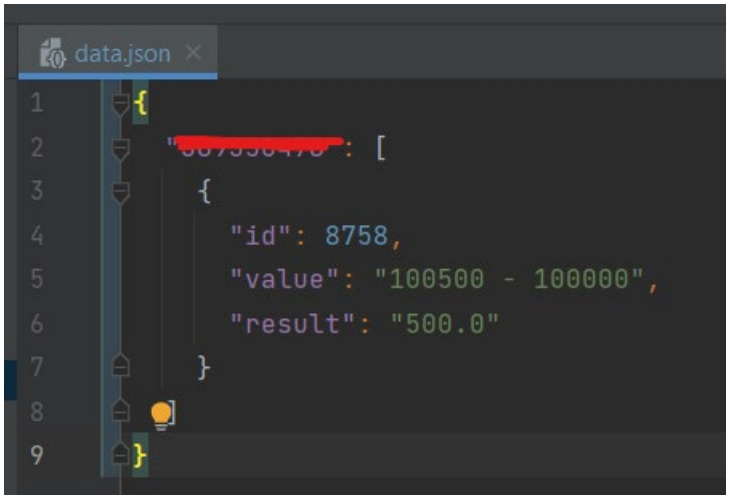
8.2.Основное меню переключателя



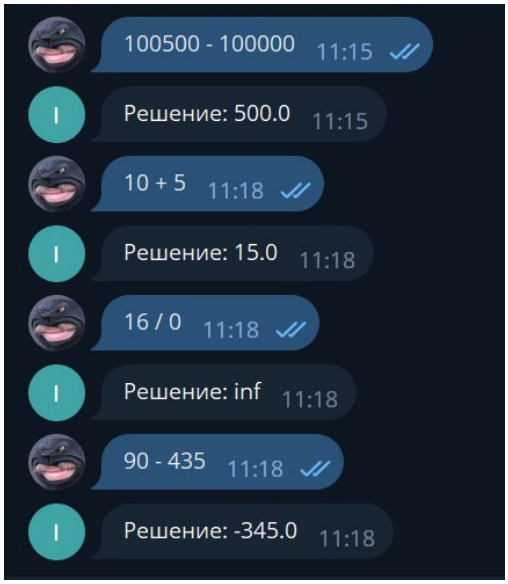
8.3. Простейший калькулятор (при нажатии на кнопку «Посчитать»)



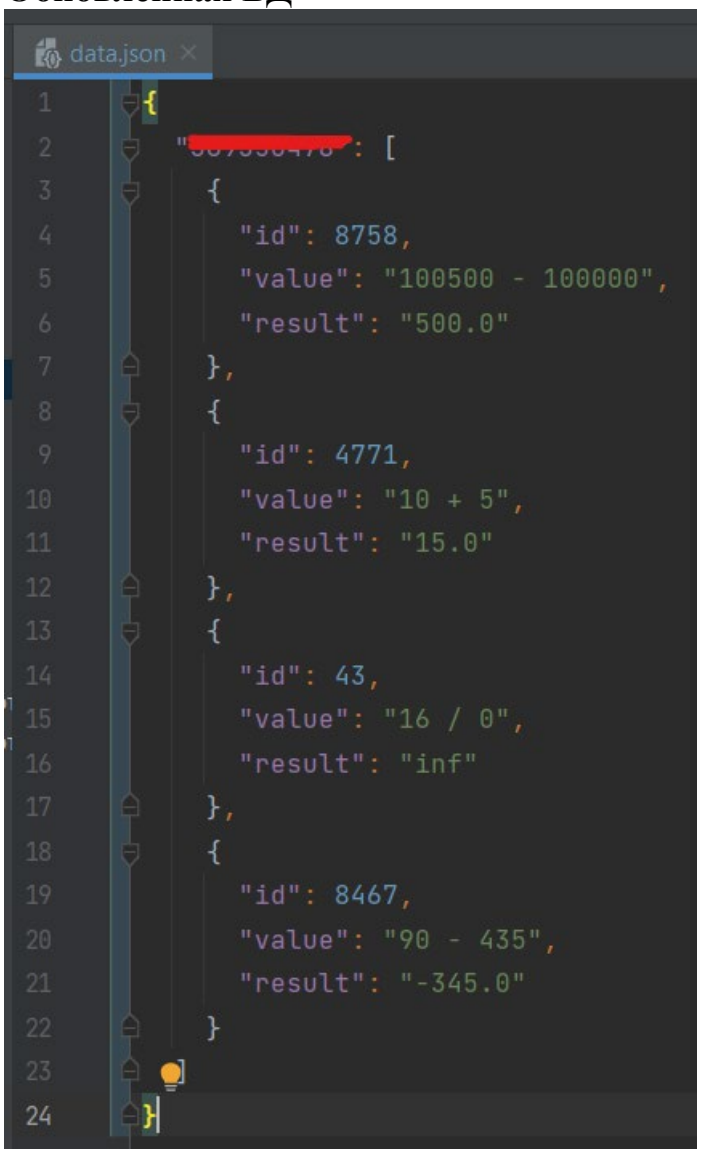
8.4.Данные хранятся в БД в формате JSON (Строчка кода 2 хранит id пользователя)



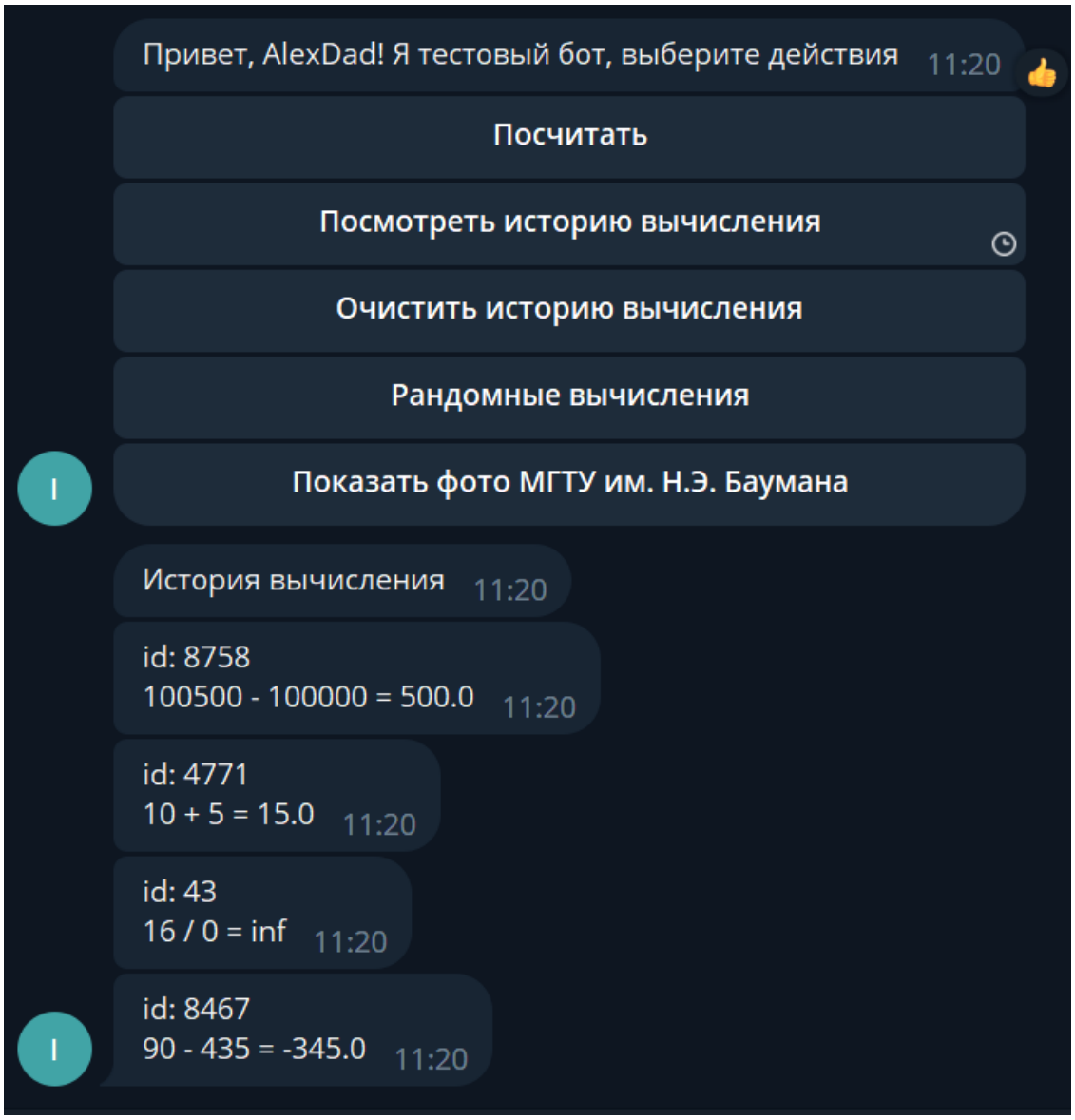
8.5.После несколько вычислений



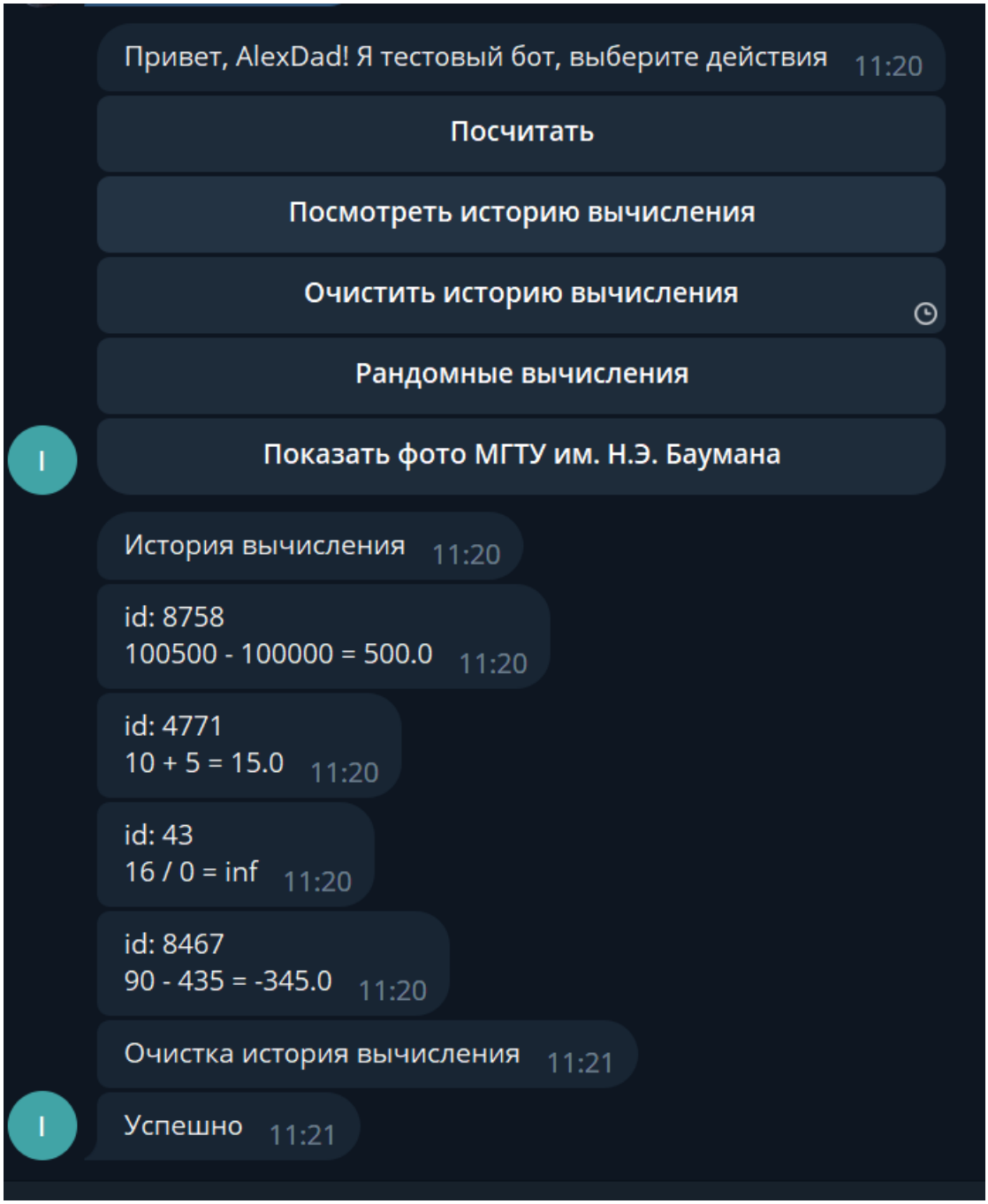
Обновленная БД



8.6. Чтение и просмотр БД в Телеграме (после нажатии на кнопку Посмотреть историю вычисления)

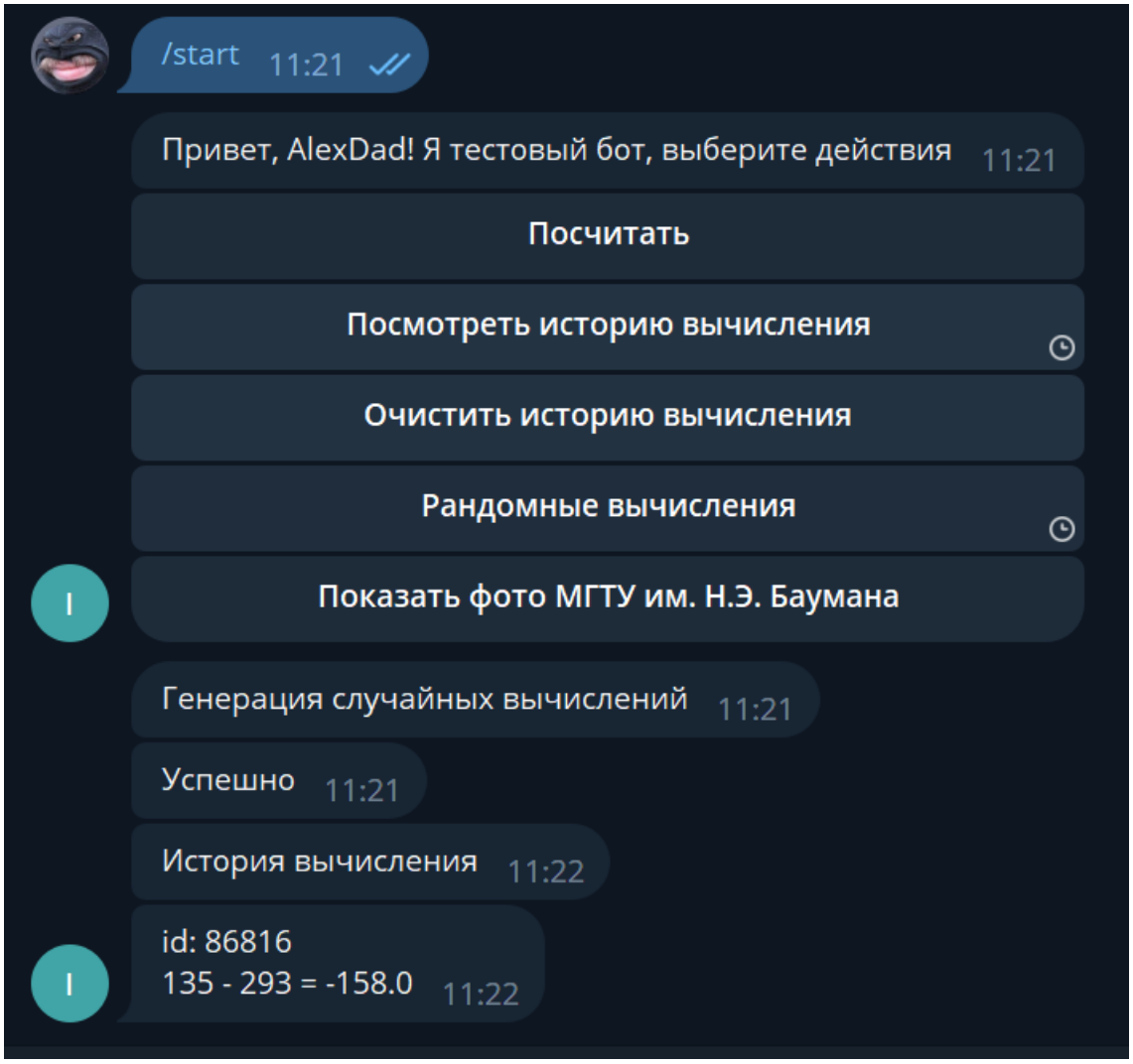


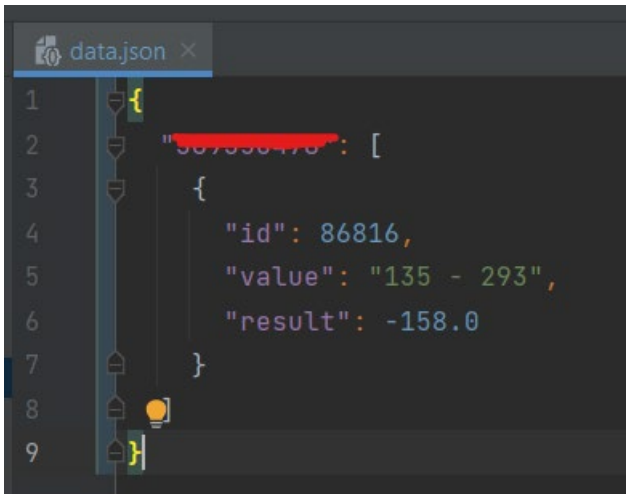
8.7.Очистка история вычисления БД в Телеграме



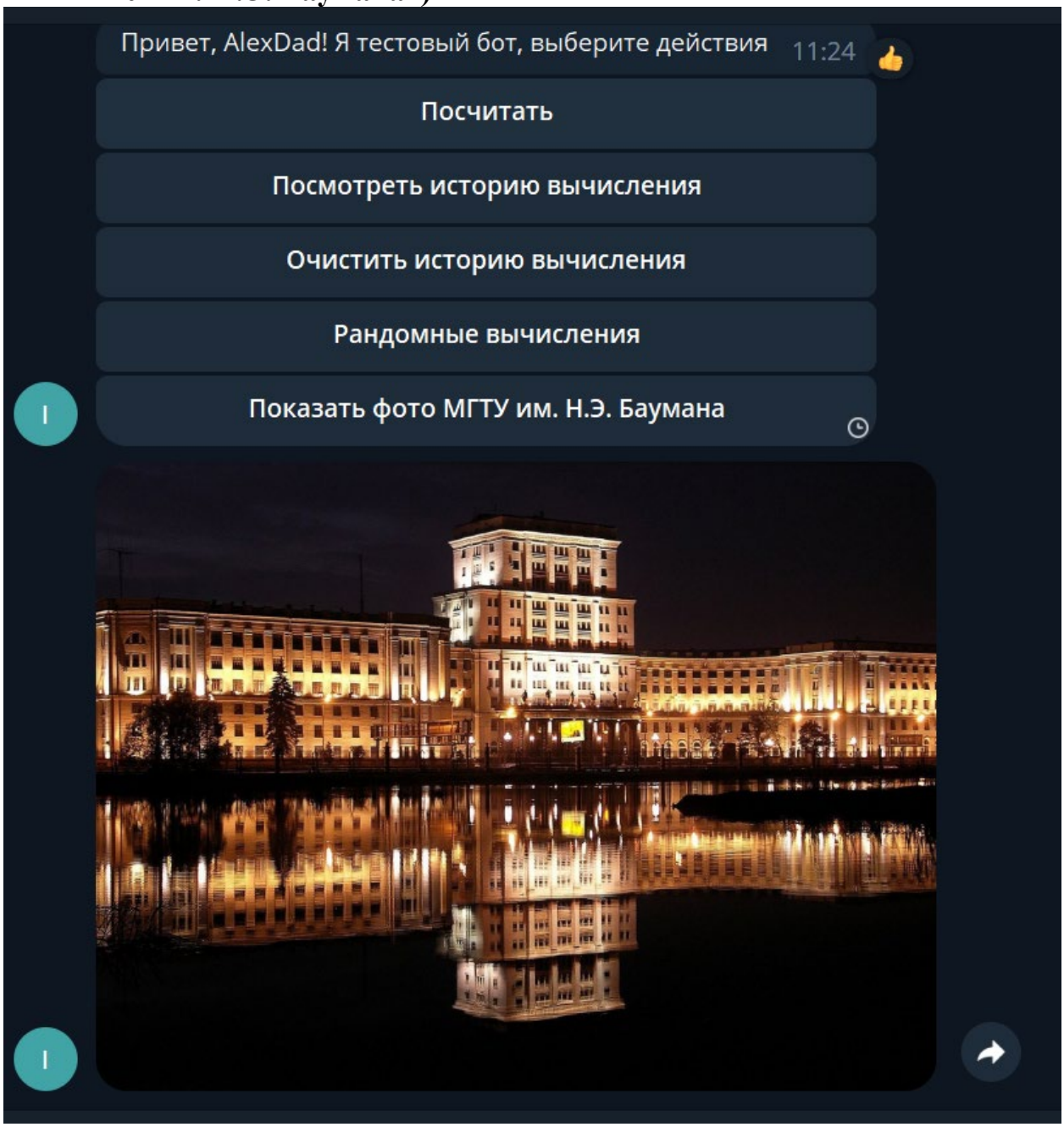
Обновленная БД

8.8.Генерация случайных вычислений и просмотр вычислений (после нажатии на кнопку «Показать фото МГТУ им. Н.Э. Баумана»)





8.9. Просмотр фотографии (после нажатии на кнопку «Показать фото МГТУ им. Н.Э. Баумана»)



s