|  |  |  |
| --- | --- | --- |
| 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."

14

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы ИУ5Ц-54Б Папин Алексей | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Гапанюк Ю.Е. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

Москва, МГТУ - 2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ ОТЧЕТА**

[1. Цель лабораторной работы 2](#_Toc120625722)

[2. Описание задания. 2](#_Toc120625723)

[3. Листинг программы: 3](#_Toc120625724)

[3.1. config.py 3](#_Toc120625725)

[3.2. calculate\_arifmetic.py 3](#_Toc120625726)

[4.1. calculate\_bot.py 4](#_Toc120625727)

[4.2. json\_function.py 7](#_Toc120625728)

[4.3. work\_with\_calculate.py 7](#_Toc120625729)

[4.4. bmstu.jpg 8](#_Toc120625730)

[5. Результаты работы программы в Telegram 9](#_Toc120625731)

[5.1. Получение справочную информацию 9](#_Toc120625732)

[5.2. Основное меню переключателя 9](#_Toc120625733)

[5.3. Простейший калькулятор (при нажатии на кнопку «Посчитать») 9](#_Toc120625734)

[5.4. Данные хранятся в БД в формате JSON 10](#_Toc120625735)

[5.5. После несколько вычислений 10](#_Toc120625736)

[5.6. Обновленная БД 11](#_Toc120625737)

[5.7. Чтение и просмотр БД в Телеграме (после нажатии на кнопку Посмотреть историю вычисления) 12](#_Toc120625738)

[5.8. Просмотр фотографии (после нажатии на кнопку «Показать фото МГТУ им. Н.Э. Баумана») 13](#_Toc120625739)

# **Цель лабораторной работы**

Изучение разработки ботов в Telegram.

# **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.

# **Листинг программы:**

# **config.py**

token = ''

# **calculate\_arifmetic.py**

1. 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

# **calculate\_bot.py**

1. 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)

# **json\_function.py**

1. 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'

# **work\_with\_calculate.py**

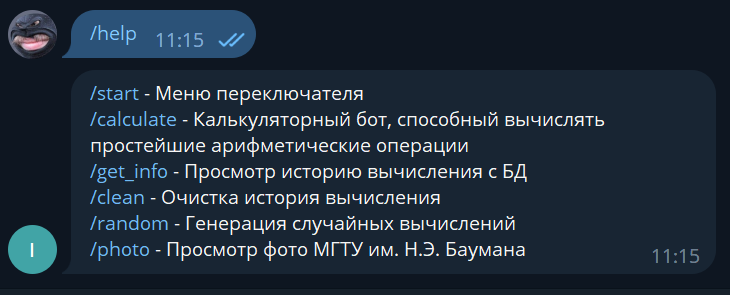
1. import random  
     
   from calculate.json\_function import write\_data, load\_data\_all, merge\_data, load\_data\_for\_id\_user  
   from calculate.calculate\_arifmetic import the\_simplest\_mathematical\_calculator as smc  
     
     
   def generate\_value(id\_user='id\_user'):  
    arifmetic = ['+', '-', '/', '\*']  
     
    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'

# **bmstu.jpg**

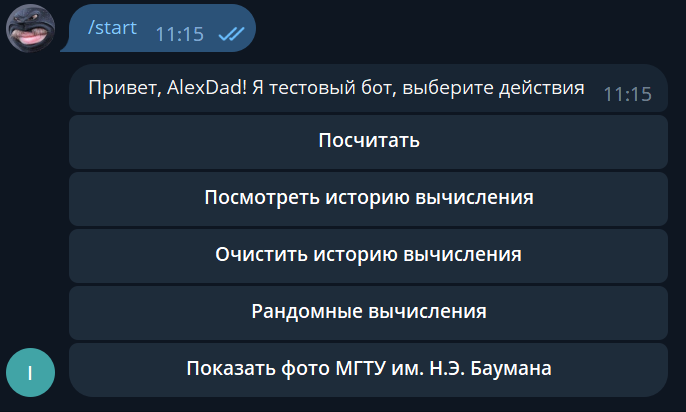


# **Результаты работы программы в Telegram**

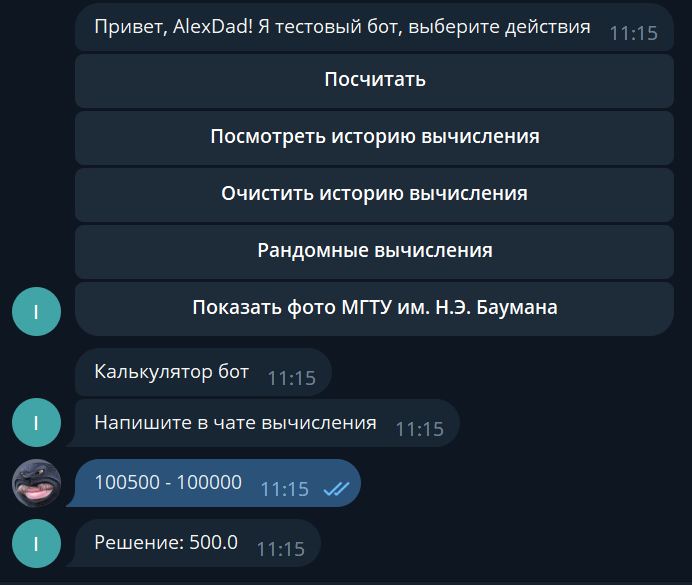
# **Получение справочную информацию**



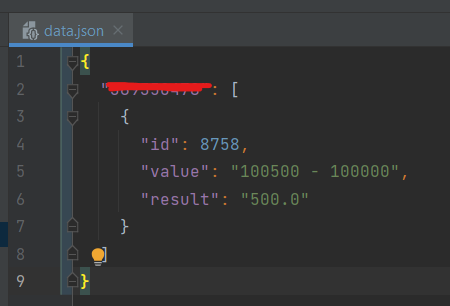
# **Основное меню переключателя**



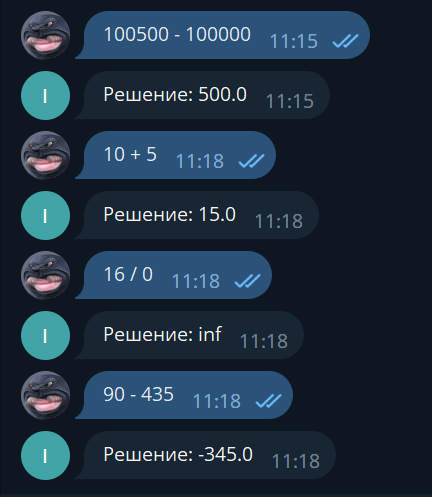
# **Простейший калькулятор (при нажатии на кнопку «Посчитать»)**



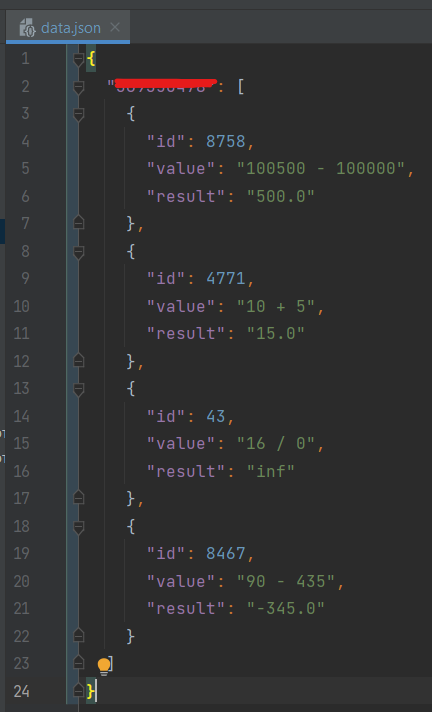
# **Данные хранятся в БД в формате JSON (Строчка кода 2 хранит id пользователя)**



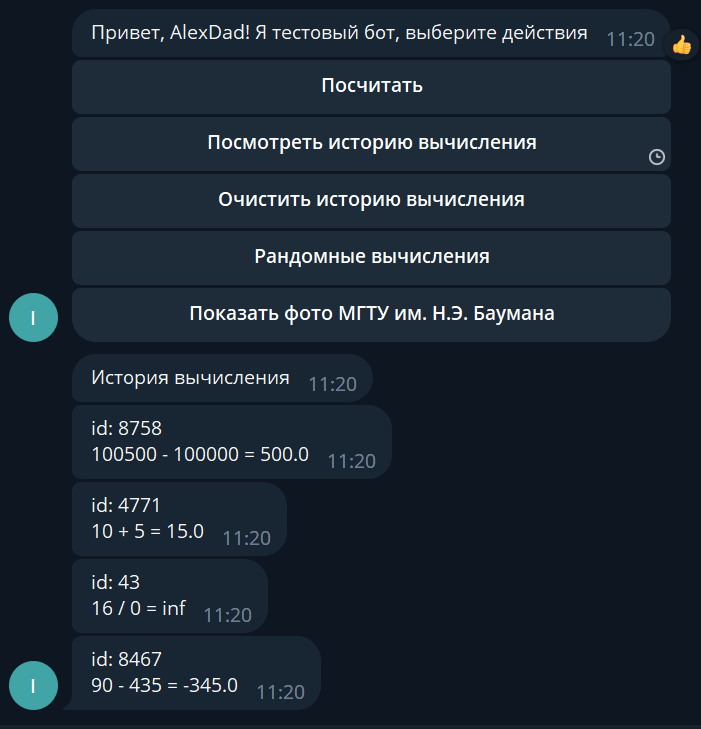
# **После несколько вычислений**



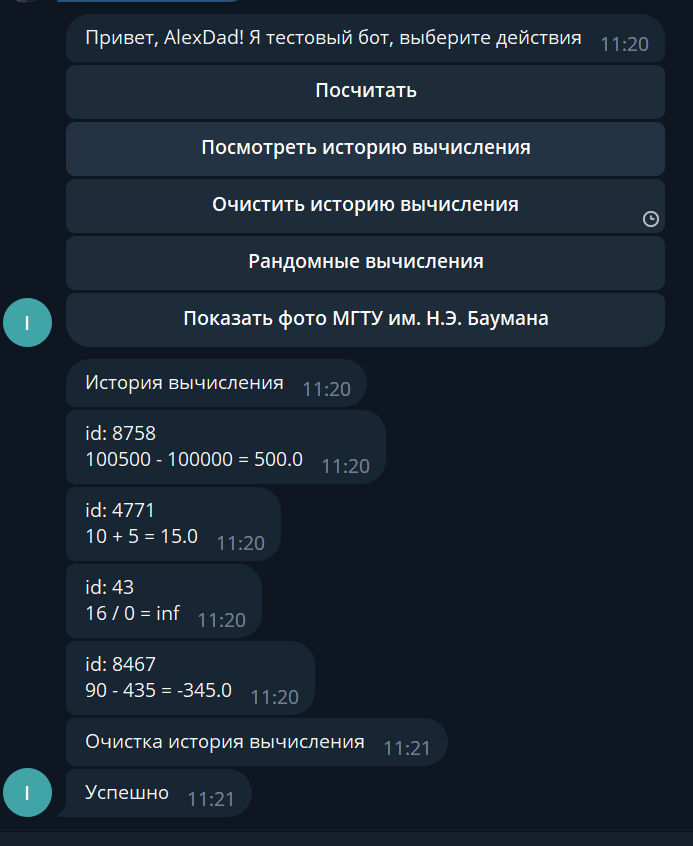
**Обновленная БД**



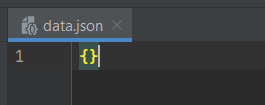
# **Чтение и просмотр БД в Телеграме (после нажатии на кнопку Посмотреть историю вычисления)**



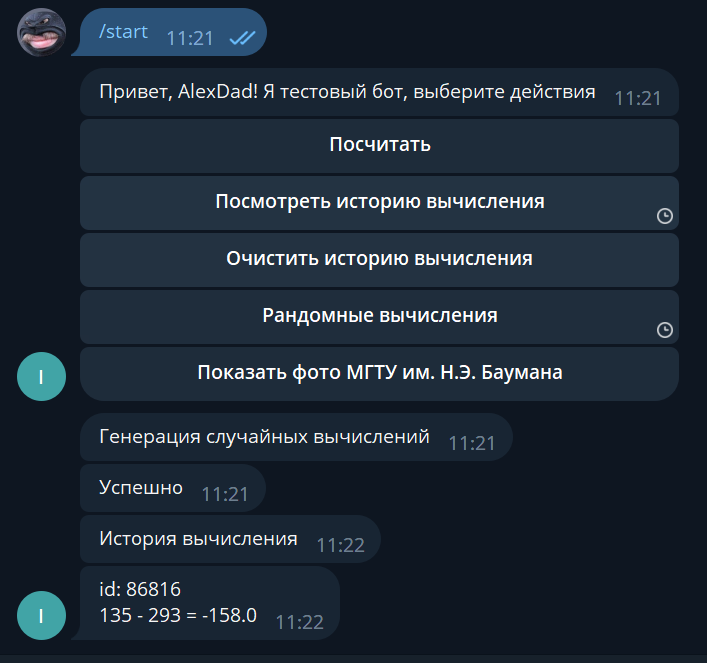
# **Очистка история вычисления БД в Телеграме**

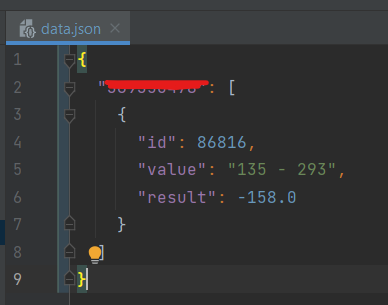


**Обновленная БД**

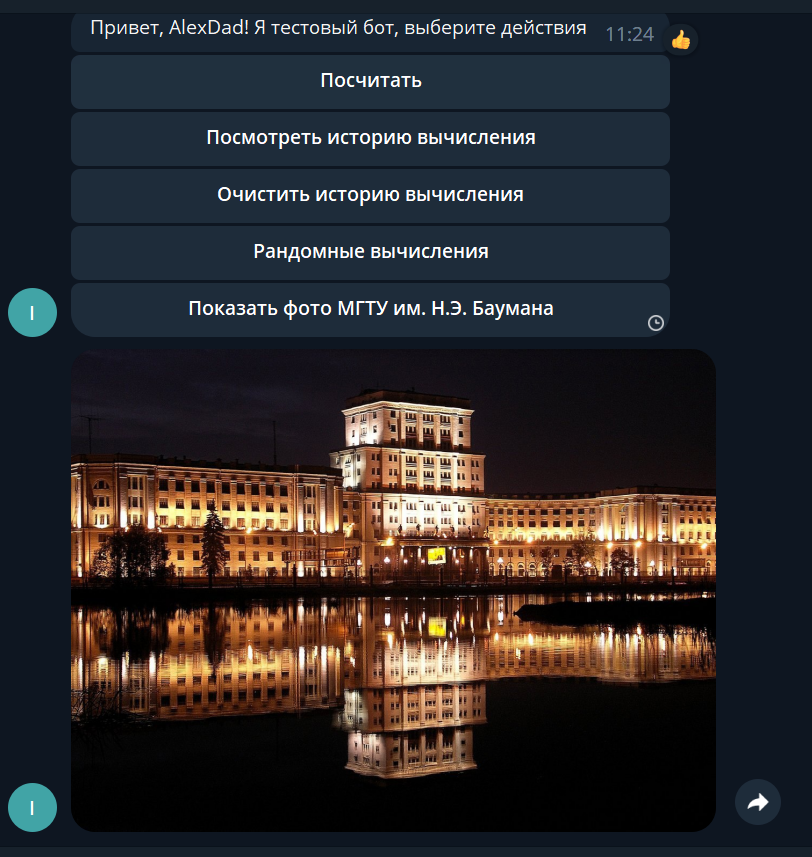
****

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





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

s