**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**

**Факультет «ГУИМЦ»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Дисциплина «Базовые компоненты ИТ» ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №5,6

Студент: Смирнов Д.И., группа ИУ5Ц-51Б Преподаватель: Гапанюк Ю.Е.

2021г.

**Описание задания**

1. Разработайте простого бота для Telegram. Бот должен использовать функциональность создания кнопок.
2. Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

**main.py**

from aiogram import Bot, Dispatcher, executor, types

from setting import \*

import test

import requests

from aiogram.types import ReplyKeyboardMarkup, ReplyKeyboardRemove, KeyboardButton

token: str = token\_bot

API = 'https://api.telegram.org/bot'

api\_url\_dog= 'https://random.dog/woof.json'

api\_url\_fox= 'https://randomfox.ca/floof/'

bot = Bot(token)

dp = Dispatcher(bot)

chat\_id=560682705

id=0

#chat\_id=442035403

keyboard: ReplyKeyboardMarkup = ReplyKeyboardMarkup(resize\_keyboard=True)

button\_cat: KeyboardButton = KeyboardButton('Picture of Fox')

button\_dog: KeyboardButton = KeyboardButton('Picture of Dog')

button\_credits: KeyboardButton = KeyboardButton('Credits')

keyboard.add(button\_cat, button\_dog,button\_credits)

def return\_chat\_id():

return id

@dp.message\_handler(commands=['start'])

async def start\_command(message: types.Message):

id= message.from\_user.id

await message.answer(f'Какую картинку хотите получить?', reply\_markup=keyboard)

@dp.message\_handler(commands=['help'])

async def help\_command(message: types.Message):

await message.answer('Данный бот отправляет картинки с животными, нажмите кнопку с нужным животным!')

@dp.message\_handler(text=['Picture of Fox'])

async def fox(message: types.Message):

fox\_response=requests.get(api\_url\_fox)

fox\_link = fox\_response.json()["link"]

if fox\_response.status\_code == 200:

await requests.get(f'{API}{token}/sendPhoto?chat\_id={chat\_id}&photo={fox\_link}')

else:

await message.answer('Error')

@dp.message\_handler(text=['Picture of Dog'])

async def dog(message: types.Message):

# dog\_response=requests.get(api\_url\_dog)

# if dog\_response.status\_code == 200:

# await message.answer(dog\_response.text)

# else:

# await message.answer('Error')

dog\_response = requests.get(api\_url\_dog)

dog\_link = dog\_response.json()["url"]

if dog\_response.status\_code == 200:

await requests.get(f'{API}{token}/sendPhoto?chat\_id={chat\_id}&photo={dog\_link}')

else:

await message.answer('Error')

@dp.message\_handler(text=['Credits'])

async def credits(message: types.Message):

await message.answer('hi,my name is daniil smirnov')

if \_\_name\_\_ == '\_\_main\_\_':

executor.start\_polling(dp, skip\_updates=True)

**test.py**

import math

import unittest

from main import start\_command

from main import credits

from main import id

from main import return\_chat\_id

class TestStart(unittest.TestCase):

# def test\_start(self):

# self.assertEqual(start\_command(message), start\_command(message))

# print()

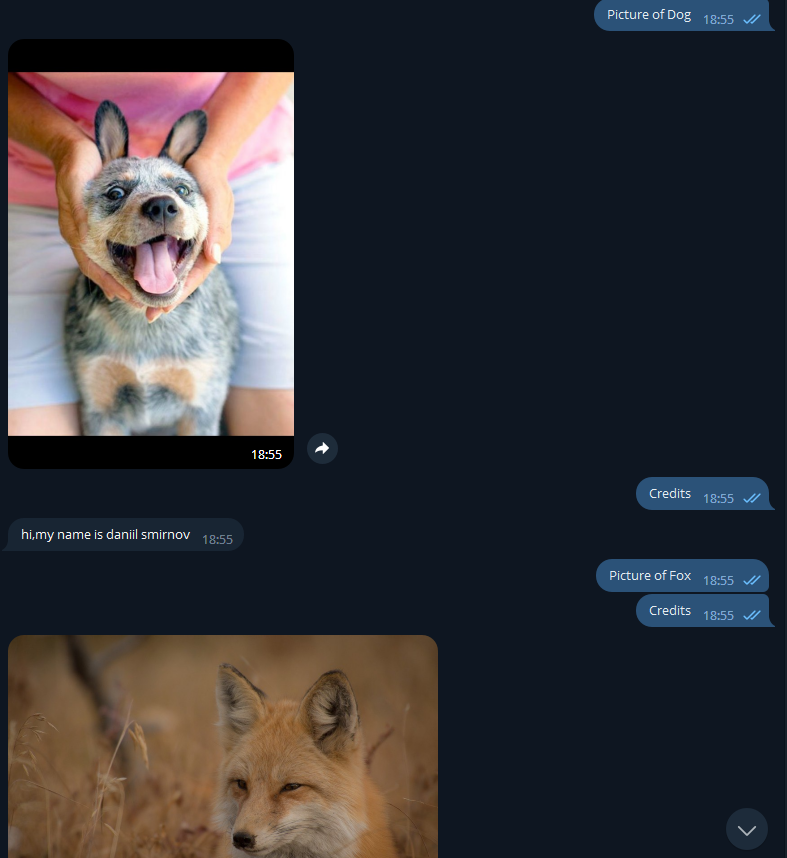
def test\_start(self):

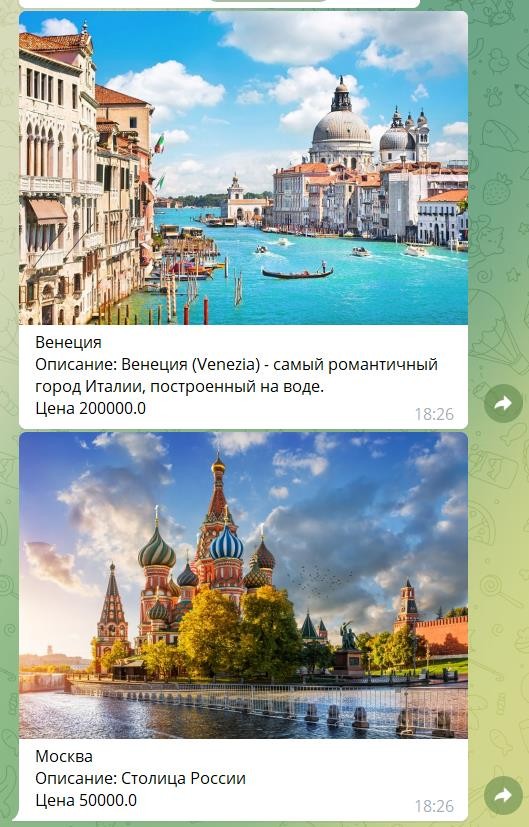
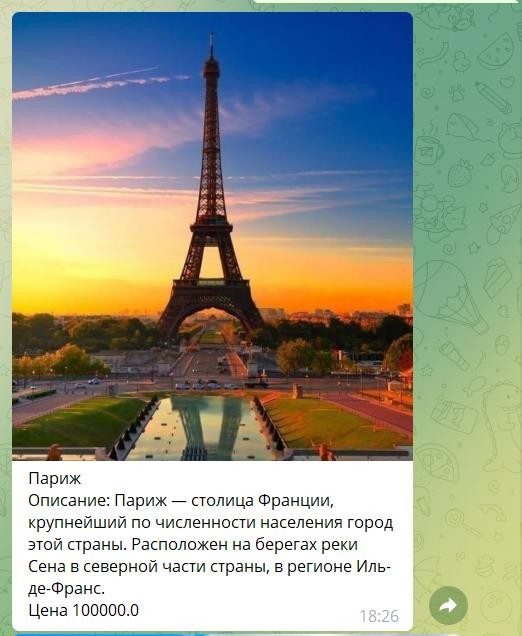
self.assertEqual(return\_chat\_id(),id)

if \_\_name\_\_ == '\_\_main\_\_':

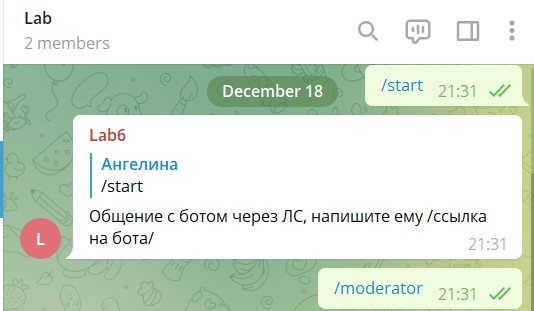
unittest.main()

Бот:

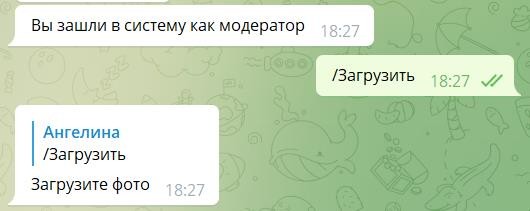
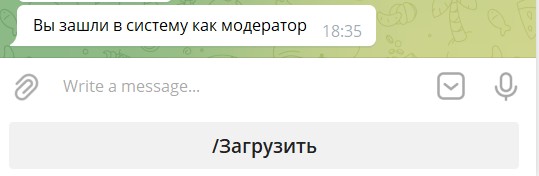


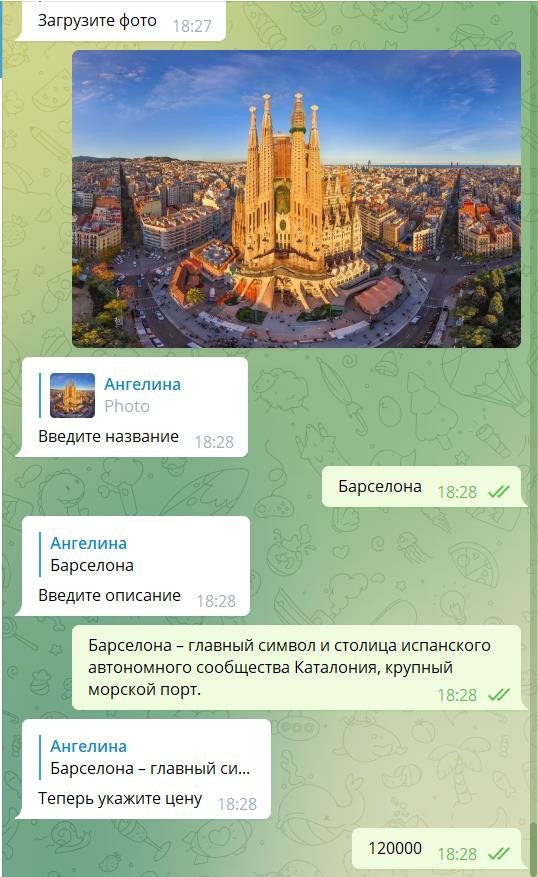


**При нажатии кнопки «Завершить» клавиатура убирается с экрана:**

**Вход в систему как модератор:**

**Загрузка новых данных:**





**Вывод новых данных:**

