Московский государственный
технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчёт по лабораторной работе № 6

Выполнил:
студент группы РТ5-31Б
Сафонов Федор
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

2021

Задание

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

Текст программы

utils.py:

```
from aiogram.utils.helper import Helper, HelperMode, ListItem

# класс с состояниями
class TestStates(Helper):
    mode = HelperMode.snake_case

    TEST_STATE_1 = ListItem()
    TEST_STATE_2 = ListItem()
    TEST_STATE_3 = ListItem()

if __name__ == '__main__':
    print(TestStates.all())
```

bot.py

```
from aiogram import Bot, types
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.dispatcher import Dispatcher
from aiogram.utils import executor
from emoji import emojize
from aiogram.contrib.middlewares.logging import LoggingMiddleware
from utils import TestStates
import requests
from googletrans import Translator

def get_cat_fact():
    url = 'http://cat-fact.herokuapp.com/facts/random'
    res = requests.get(url)
    json = res.json()
    return str(json['text'])

def get_dog_fact():
    url = 'http://dog-api.kinduff.com/api/facts'
    res = requests.get(url)
    json = res.json()
    return str(*json['facts'])

def translate_fact(text):
    translator = Translator()
    translation = translator.translate(text, dest='ru')
    return translation.text

def to_db(text):
    f = open('last_trans.txt', 'w')
```

```
    f.write(text)
    f.close()

def get_db():
    f = open('last_trans.txt', 'r')
    s = ""
    for line in f:
        s += line
    f.close()
    return s

mes_cat = 'Cat fact'
mes_dog = 'Dog fact'
mes_trans = 'Translate'

TOKEN = '5086695702:AAFduCBMMovZdojz-NLaKt5Jlyirm3P-QhU'
bot = Bot(token=TOKEN)
dp = Dispatcher(bot, storage=MemoryStorage()) # указываем хранилище состояний
dp.middleware.setup(LoggingMiddleware()) # включаем логирование

@dp.message_handler(commands=['start']) # первое знакомство
async def process_start_command(message: types.Message):
    state = dp.current_state(user=message.from_user.id)
    await state.set_state(TestStates.all()[0]) # устанавливаем начальное состояние
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)  # следующая клава
    buttons = [mes_cat, mes_dog]
    keyboard.add(*buttons)
    await message.reply(emojize('Hi!     '), reply=False, reply_markup=keyboard)

@dp.message_handler(state=TestStates.TEST_STATE_1) # состояние выбора факта
async def first_test_case_fact(message : types.Message):
    state = dp.current_state(user=message.from_user.id)
    await state.set_state(TestStates.all()[1])  # устанавливаем в следующее состояние

    text = message.text
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True) # следующая клава
    buttons = [mes_cat, mes_dog, mes_trans]
    keyboard.add(*buttons)
    if text == mes_cat:
        fact = get_cat_fact()
        await message.answer(fact, reply=False, reply_markup=keyboard)
        to_db(fact)
    elif text == mes_dog:
        fact = get_dog_fact()
        await message.answer(fact, reply=False, reply_markup=keyboard)
        to_db(fact)

@dp.message_handler(state=TestStates.TEST_STATE_2) # возможность перевода
async def second_test_case_trans(message : types.Message):
    state = dp.current_state(user=message.from_user.id)

    text = message.text
    if text == mes_cat:
        fact = get_cat_fact()
        await message.answer(fact, reply=False)
        to_db(fact)
    elif text == mes_dog:
```

```
        fact = get_dog_fact()
        await message.answer(fact, reply=False)
        to_db(fact)
    if text == mes_trans: # переводим текст и переходим в состояние оценки
        await state.set_state(TestStates.all()[2])
        await message.answer(translate_fact(get_db()), reply=False)
        keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
        buttons = list(map(str, [i for i in range(1, 11)]))
        keyboard.add(*buttons)
        await message.answer('Оцените пожалуйста перевод', reply=False,
reply_markup=keyboard) # обновляем клаву

@dp.message_handler(state=TestStates.TEST_STATE_3) # возможность оценки перевода,
переход в начало
async def second_test_case_trans(message : types.Message):
    with open('trans_ratings.txt', 'a') as f:
        f.write(message.text+'\n')
    state = dp.current_state(user=message.from_user.id)
    await state.set_state(TestStates.all()[0]) # переход в начальное состояние автомата
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)  # следующая клава
    buttons = [mes_cat, mes_dog]
    keyboard.add(*buttons)
    await message.reply(emojize('Спасибо за оценку!'), reply=False, reply_markup=keyboard)
    print(message.text)

if __name__ == "__main__":
    executor.start_polling(dp)
```

tdd.py

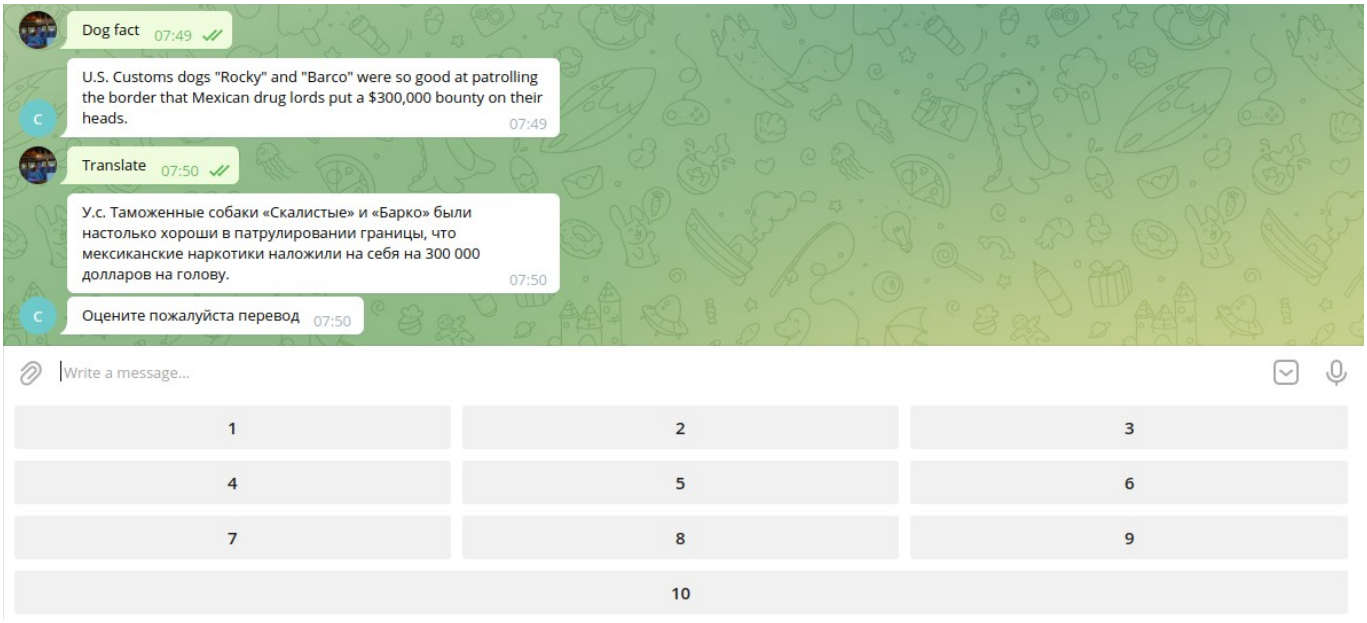
```
import unittest

import requests
from googletrans import Translator

def get_cat_fact():
    url = 'http://cat-fact.herokuapp.com/facts/random'
    res = requests.get(url)
    return res.status_code

def get_dog_fact():
    url = 'http://dog-api.kinduff.com/api/facts'
    res = requests.get(url)
    return res.status_code

def translate_fact(text):
    translator = Translator()
    translation = translator.translate(text, dest='ru')
    return translation.text

class MyTestCase(unittest.TestCase):

    def test_status_code_cat(self): # тест кода состояние отклика сайта
        self.assertEqual(get_cat_fact(), 200)

    def test_status_code_dog(self): # тест кода состояние отклика другого сайта
```

```
        self.assertEqual(get_dog_fact(), 200)

    def test_translate(self): # тест перевода
        alphabet = set('abcdefghijklmnopqrstuvwxyz')
        text = 'I love my mother'
        trans_text = translate_fact(text)
        self.assertSetEqual(alphabet & set(trans_text), set())


if __name__ == '__main__':
    unittest.main()
```

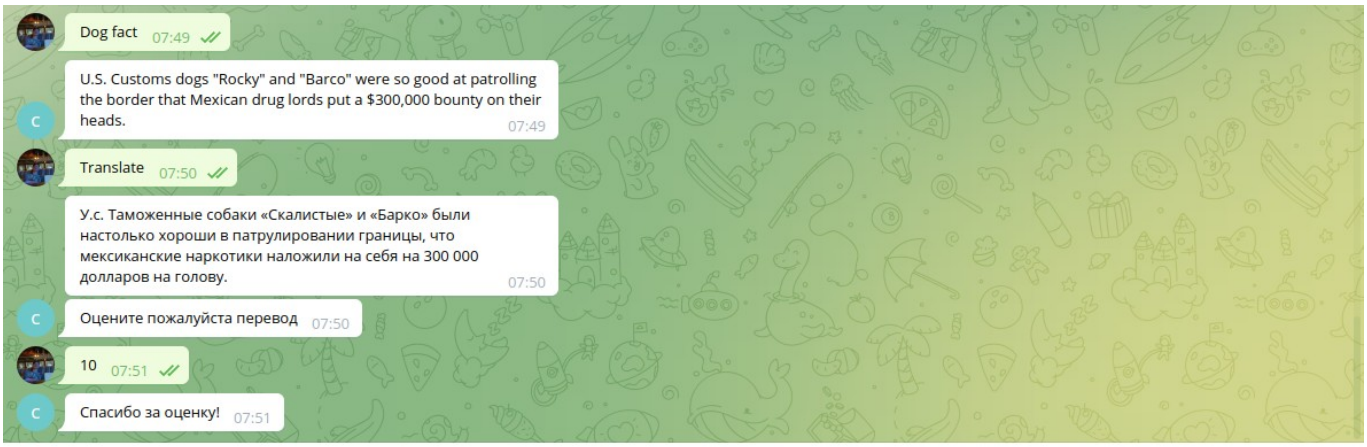
Примеры работы

1 – ое состояние

2 – ое состояние



3 – ое состояние (начальное, когда еще нечего переводить)



Cat fact | Dog fact