

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)

Отчет по домашнему заданию по курсу
Базовые компоненты интернет-технологий
«Изучение возможностей создания ботов в Telegram и их
тестирования»

ПРЕПОДАВАТЕЛЬ:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

(подпись)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-34Б

Верин Д.С.

(подпись)

" " _____ 2021 г.

Постановка задачи

1. Модифицируйте код лабораторной работы №6 таким образом, чтобы он был пригоден для модульного тестирования.
2. Используя материалы лабораторной работы №4 создайте модульные тесты с применением TDD - фреймворка (2 теста) и BDD - фреймворка (2 теста).

Текст программы

Файл config.py

```
from enum import Enum

token = 'token'

appid = "appid"

db_file = "db.vdb"
CURRENT_STATE = "CURRENT_STATE"

class States(Enum):
    START_STATE = 0
    LOCATION_STATE = 1
    DAY_STATE = 2
    DETAIL_STATE = 3
    END_STATE = 4
```

Файл dbworker.py

```
from redis import Redis
import config

def get(key):
    with Redis(config.db_file) as db:
        try:
            return db[key].decode()
        except KeyError:
            return config.States.START_STATE.value

def set(key, value):
    with Redis(config.db_file) as db:
        try:
            db[key] = value
            return True
        except:
            return False

def make_key(user_id, current_state):
    return f"{user_id}_{current_state}"
```

Файл functions.py

```
import os

cur_path = os.getcwd()

def picture_path(name):
    return os.path.join(cur_path, f'pictures\\{name}.png')
```

Файл **weather.py**

```
import requests
from config import appid

def weather_forecast(*args):
    try:
        if len(args) == 3:
            res = requests.get("http://api.openweathermap.org/data/2.5/forecast",
                               params={'lon': args[0], 'lat': args[1], 'units':
                                       'metric', 'lang': 'ru', 'APPID': appid})
            day = args[2]

        else:
            res = requests.get("http://api.openweathermap.org/data/2.5/forecast",
                               params={'q' : args[0], 'units': 'metric', 'lang': 'ru',
                                       'APPID': appid})
            day = args[1]

        data = res.json()
        i = 0
        for f in data['list']:
            if i == day:
                return f
            i += 1
    except Exception as e:
        print("Exception:", e)
```

Файл **bot.py**

```
import dbworker
import config
import telebot
from weather import weather_forecast
from functions import picture_path

bot = telebot.TeleBot(config.token)

@bot.message_handler(commands=["start"])
def greetings(message):
    markup = telebot.types.ReplyKeyboardMarkup(row_width=1, resize_keyboard=True)
    markup.add(telebot.types.KeyboardButton('Привет, бот Дима'))
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
                 config.States.START_STATE.value)
    bot.send_message(message.chat.id, 'Привет, меня зовут бот Дима, я могу
    отправлять прогноз погоды!', reply_markup=markup)

@bot.message_handler(func=lambda message:
```

```

int(dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE))) ==
config.States.START_STATE.value)
def start(message):
    markup = telebot.types.ReplyKeyboardMarkup(row_width=1,resize_keyboard=True)
    markup.add(telebot.types.KeyboardButton('Геолокация', request_location=True))
    bot.send_message(message.chat.id, 'Отправь мне свою геолокацию или введи
город!', reply_markup=markup)
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.LOCATION_STATE.value)

@bot.message_handler(content_types=["location"], func= lambda message:
int(dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE))) ==
config.States.LOCATION_STATE.value)
def location_geo(message):
    if message.location is not None:
        dbworker.set(dbworker.make_key(message.chat.id,
config.States.LOCATION_STATE.value), 0)
        dbworker.set(dbworker.make_key(message.chat.id, 'longitude'),
message.location.longitude)
        dbworker.set(dbworker.make_key(message.chat.id, 'latitude'),
message.location.latitude)
        dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.DAY_STATE.value)
        markup = telebot.types.ReplyKeyboardMarkup(row_width=1,
resize_keyboard=True)
        bt1 = telebot.types.KeyboardButton('Сегодня')
        bt2 = telebot.types.KeyboardButton('Завтра')
        bt3 = telebot.types.KeyboardButton('Послезавтра')
        markup.add(bt1, bt2, bt3)
        bot.send_message(message.chat.id, 'Выберите день, на который хотите
посмотреть погоду', reply_markup=markup)

@bot.message_handler(func= lambda message:
int(dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE))) ==
config.States.LOCATION_STATE.value)
def location_city(message):
    if message.text:
        dbworker.set(dbworker.make_key(message.chat.id,
config.States.LOCATION_STATE.value), 1)
        dbworker.set(dbworker.make_key(message.chat.id, 'city'), message.text)
        dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.DAY_STATE.value)
        markup = telebot.types.ReplyKeyboardMarkup(row_width=1,
resize_keyboard=True)
        bt1 = telebot.types.KeyboardButton('Сегодня')
        bt2 = telebot.types.KeyboardButton('Завтра')
        bt3 = telebot.types.KeyboardButton('Послезавтра')
        markup.add(bt1, bt2, bt3)
        bot.send_message(message.chat.id, 'Выберите день, на который хотите
посмотреть погоду', reply_markup=markup)

```

```

@bot.message_handler(func=lambda message:
int(dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE))) ==
config.States.DAY_STATE.value)
def day(message):
    if message.text == 'Сегодня' or message.text == 'Завтра' or message.text ==
'Послезавтра':
        if message.text == 'Сегодня':
            dbworker.set(dbworker.make_key(message.chat.id,
config.States.DAY_STATE.value), 0)
        elif message.text == 'Завтра':
            dbworker.set(dbworker.make_key(message.chat.id,
config.States.DAY_STATE.value), 1)
        else:
            dbworker.set(dbworker.make_key(message.chat.id,
config.States.DAY_STATE.value), 2)
            dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.DETAIL_STATE.value)
            markup = telebot.types.ReplyKeyboardMarkup(row_width=1,
resize_keyboard=True)
            bt1 = telebot.types.KeyboardButton('Коротко')
            bt2 = telebot.types.KeyboardButton('Подробно')
            markup.add(bt1, bt2)
            bot.send_message(message.chat.id, 'Выберите краткий прогноз или полный',
reply_markup=markup)
        else:
            markup = telebot.types.ReplyKeyboardMarkup(row_width=1,
resize_keyboard=True)
            bt1 = telebot.types.KeyboardButton('Сегодня')
            bt2 = telebot.types.KeyboardButton('Завтра')
            bt3 = telebot.types.KeyboardButton('Послезавтра')
            markup.add(bt1, bt2, bt3)
            bot.send_message(message.chat.id, 'Выберите день из списка ниже',
reply_markup=markup)

@bot.message_handler(func=lambda message:
int(dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE))) ==
config.States.DETAIL_STATE.value)
def detail(message):
    if message.text == 'Коротко' or message.text == 'Подробно':
        if int(dbworker.get(dbworker.make_key(message.chat.id,
config.States.LOCATION_STATE.value))):
            forecast =
weather_forecast(dbworker.get(dbworker.make_key(message.chat.id, 'city')),
int(dbworker.get(dbworker.make_key(message.chat.id,
config.States.DAY_STATE.value))))
        else:
            forecast =
weather_forecast(dbworker.get(dbworker.make_key(message.chat.id, 'longitude')),
dbworker.get(dbworker.make_key(message.chat.id, 'latitude')),
int(dbworker.get(dbworker.make_key(message.chat.id,
config.States.DAY_STATE.value))))

```

```

        if message.text == 'Коротко':
            res = {"погода:" : forecast['weather'][0]['description'],
"температура:" : forecast['main']['temp'],
"ощущается:" : forecast['main']['feels_like']}
        else:
            res = {"погода:" : forecast['weather'][0]['description'],
"температура:" : forecast['main']['temp'],
"ощущается:" : forecast['main']['feels_like'], "минимальная
температура:" : forecast['main']['temp_min'],
"максимальная температура:" : forecast['main']['temp_max'],
"атмосферное давление:" : forecast['main']['pressure']}
            path = picture_path(res['погода:'])
            result = ''
            for k, v in res.items():
                result += k + ' ' + str(v) + '\n'
            with open(path, 'rb') as photo:
                bot.send_photo(message.chat.id, photo, result)
            markup =
telebot.types.ReplyKeyboardMarkup(row_width=2,resize_keyboard=True)
            markup.add(telebot.types.KeyboardButton('Да'),
telebot.types.KeyboardButton('Нет'))
            bot.send_message(message.chat.id, 'Желаете ли посмотреть погоду где-
нибудь еще?', reply_markup=markup)

            dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.END_STATE.value)

        else:
            dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.DETAIL_STATE.value)
            markup = telebot.types.ReplyKeyboardMarkup(row_width=1,
resize_keyboard=True)
            bt1 = telebot.types.KeyboardButton('Коротко')
            bt2 = telebot.types.KeyboardButton('Подробно')
            markup.add(bt1, bt2)
            bot.send_message(message.chat.id, 'Ошибка, выберите из списка ниже',
reply_markup=markup)

@bot.message_handler(func=lambda message:
int(dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE))) ==
config.States.END_STATE.value)
def end(message):
    if message.text == 'Да':
        markup =
telebot.types.ReplyKeyboardMarkup(row_width=1,resize_keyboard=True)
        markup.add(telebot.types.KeyboardButton('Геолокация',
request_location=True))
        bot.send_message(message.chat.id, 'Отправь мне свою геолокацию или введи
город!', reply_markup=markup)
        dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.LOCATION_STATE.value)

```

```

elif message.text == 'Нет':
    bot.send_message(message.chat.id, 'Спасибо за использование этого бота!\nЧтобы начать напишите /start',
        reply_markup=telebot.types.ReplyKeyboardRemove())
else:
    markup = telebot.types.ReplyKeyboardMarkup(row_width=2,
        resize_keyboard=True)
    markup.add(telebot.types.KeyboardButton('Да'),
        telebot.types.KeyboardButton('Нет'))
    bot.send_message(message.chat.id, 'Не совсем понимаю, выберите кнопку ниже', reply_markup=markup)

if __name__ == '__main__':
    bot.infinity_polling()

```

Файл test.py

```

import unittest
from functions import picture_path

class Test(unittest.TestCase):
    def test1(self):
        self.assertEqual(picture_path('небольшой снег'),
            'D:\BKIT\dz\pictures\небольшой снег.png')
    def test2(self):
        self.assertEqual(picture_path('облачно с прояснениями'),
            'D:\BKIT\dz\pictures\облачно с прояснениями.png')

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

```

Файл testBDD.feature

Feature: testing my bot

Scenario: Test1

Given I have the conditions пасмурно
 When I find picture_path
 Then I expect the result to be D:\BKIT\dz\pictures\пасмурно.png

Scenario: Test2

Given I have the conditions переменная облачность
 When I find picture_path
 Then I expect the result to be D:\BKIT\dz\pictures\переменная облачность.png

Файл steps.py

```

from behave import given, when, then
from functions import picture_path

@given("I have the conditions {conditions}")
def data(context, conditions):
    context.c = conditions

@when("I find picture_path")

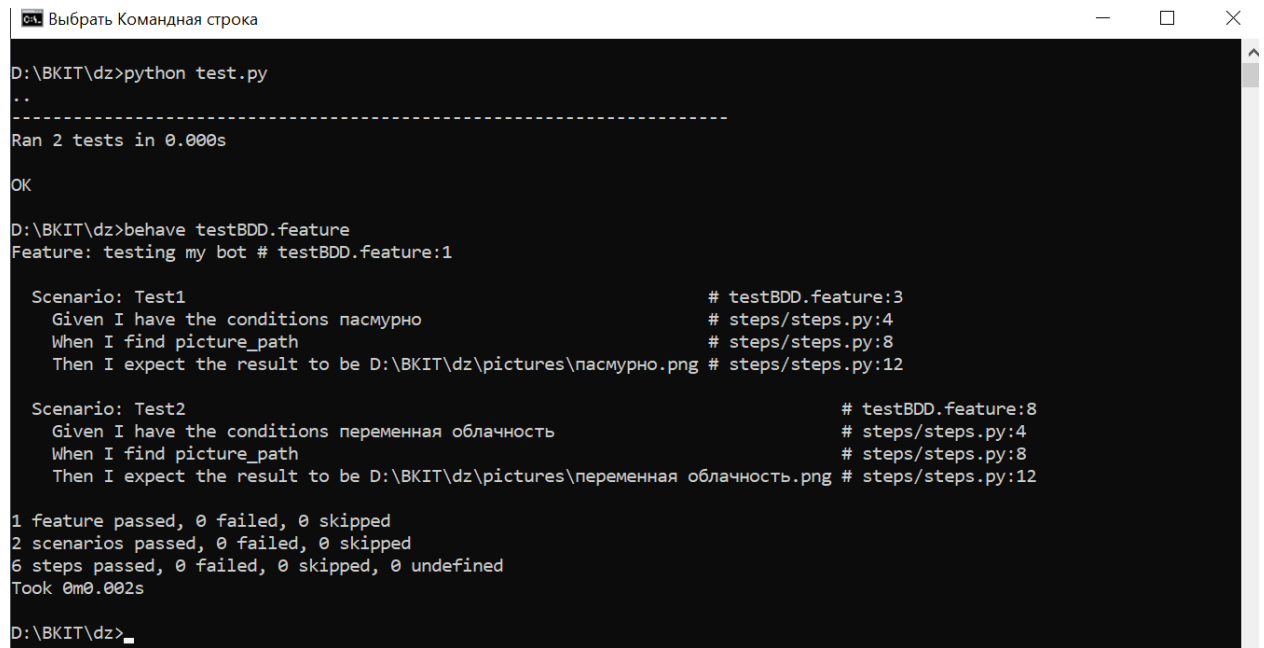
```



```
def perform_func(context):
    context.path = picture_path(context.c)

@then("I expect the result to be {path}")
def expect_result(context, path):
    assert context.path == path
```

Пример выполнения программы



```
Выбрать Командная строка

D:\BKIT\dz>python test.py
..
-----
Ran 2 tests in 0.000s

OK

D:\BKIT\dz>behave testBDD.feature
Feature: testing my bot # testBDD.feature:1

  Scenario: Test1                                     # testBDD.feature:3
    Given I have the conditions пасмурно              # steps/steps.py:4
    When I find picture_path                          # steps/steps.py:8
    Then I expect the result to be D:\BKIT\dz\pictures\пасмурно.png # steps/steps.py:12

  Scenario: Test2                                     # testBDD.feature:8
    Given I have the conditions переменная облачность # steps/steps.py:4
    When I find picture_path                          # steps/steps.py:8
    Then I expect the result to be D:\BKIT\dz\pictures\переменная облачность.png # steps/steps.py:12

1 feature passed, 0 failed, 0 skipped
2 scenarios passed, 0 failed, 0 skipped
6 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.002s

D:\BKIT\dz>
```