



**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 «Информатика и системы управления»  
Кафедра «Системы обработки информации и управления»**

**Домашнее задание  
по дисциплине «Базовые компоненты интернет технологий»**

**Выполнил:  
студент группы ИУ5-35Б Тазенков И. Д.**

## Задание:

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

### Bot.py

```
from weather import resp_test, Weather
import logging
from aiogram import Bot, Dispatcher, executor, types
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters import Text
from aiogram.dispatcher.filters.state import State, StatesGroup

class Form(StatesGroup):
    country = State()
    city = State()
    send_weather = State()

TOKEN = "5031931528:AAHssOqk4aGjf2Nwj3J8owc8JoVz6ngpRYY"

storage = MemoryStorage()
# Объект бота
bot = Bot(token=TOKEN)
# Диспетчер для бота
dp = Dispatcher(bot, storage=storage)
# Включаем логирование, чтобы не пропустить важные сообщения
logging.basicConfig(level=logging.INFO)

@dp.message_handler(commands='start')
async def start(message: types.Message):
    await message.answer("Привет!\nВ какой стране хотите узнать погоду?")
    await Form.country.set()

@dp.message_handler(state='*', commands='reset')
@dp.message_handler(Text(equals='reset', ignore_case=True), state='*')
async def cancel_handler(message: types.Message, state: FSMContext):
    """
    Allow user to cancel any action
    """
    current_state = await state.get_state()
    if current_state is None:
        return

    logging.info('Cancelling state %r', current_state)
```

```

        # Cancel state and inform user about it
        await state.finish()
        # And remove keyboard (just in case)
        await message.reply('Начнем по новой!\nВ какой стране хотите узнать
погоду?',
                                reply_markup=types.ReplyKeyboardRemove())
        await Form.country.set()

@dp.message_handler(lambda message: not resp_test(message.text),
state=Form.country)
async def process_country_invalid(message: types.Message):
    """
    If country is invalid
    """
    return await message.reply('Такой страны не существует\nПопробуйте ввести
название страны заново')

@dp.message_handler(state=Form.country)
async def entering_country(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['country'] = message.text
    await Form.next()
    await message.answer("В каком городе хотите узнать погоду?")

@dp.message_handler(lambda message: not resp_test(message.text),
state=Form.city)
async def process_city_invalid(message: types.Message):
    """
    If city is invalid
    """
    return await message.reply('Такого города не существует\nПопробуйте
ввести название города заново')

@dp.message_handler(lambda message: resp_test(message.text), state=Form.city)
async def entering_city(message: types.Message, state: FSMContext):
    await Form.next()
    async with state.proxy() as data:
        data['city'] = message.text

    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
    item = types.KeyboardButton('Показать погоду')
    markup.add(item)
    await message.answer('Нажмите, пожалуйста, на кнопку',
reply_markup=markup)

@dp.message_handler(state=Form.send_weather)
async def sending_weather(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        w = Weather()
        w.city = data['city']
        w.country = data['country']
        resp = w.get_weather()
    if not resp:
        await message.answer('Повторите ввести данные заново\nТакого города
или страны не существует',
                                reply_markup=types.ReplyKeyboardRemove())
    else:
        await message.answer(resp, reply_markup=types.ReplyKeyboardRemove())

```

```

    await state.finish()
    await message.answer('Начнем по новой!\nВ какой стране хотите узнать
погоду?',
                        reply_markup=types.ReplyKeyboardRemove())
    await Form.country.set()

if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)

```

## weather.py

```

import requests

app_id = "0d6eb8362cfe0622d5bc1b40279fb037"

def resp_test(name) -> bool:
    try:
        res = requests.get("http://api.openweathermap.org/data/2.5/find",
                           params={'q': name, 'units': 'metric', 'lang':
'ru', 'APPID': app_id})
        data = res.json()['list'][0]
    except:
        return False
    return True if data is not None else False

class Weather:
    def __init__(self):
        self._city = ""
        self._country = ""
        self._app_id = app_id

    @property
    def city(self):
        return self._city

    @city.setter
    def city(self, city):
        self._city = city

    @property
    def country(self):
        return self._country

    @country.setter
    def country(self, country):
        self._country = country

    def get_weather(self):
        place = ''.join([self.city, ', ', self.country])

        try:
            res = requests.get("http://api.openweathermap.org/data/2.5/find",
                               params={'q': place, 'units': 'metric', 'lang':
'ru', 'APPID': app_id})
            data = res.json()['list'][0]
            city = data['name'] + ' ' + str(data['sys']['country'])
            cond = "Условия:" + ' ' + data['weather'][0]['description']
            temp = "Температура:" + ' ' + str(data['main']['temp'])
            temp_min = "Минимальная температура:" + ' ' +

```

```

str(data['main']['temp_min'])
    temp_max = "Максимальная температура:" + ' ' +
str(data['main']['temp_max'])
    feels_like = "Ощущается как:" + ' ' +
str(data['main']['feels_like'])
    response = ''.join(
        [city + '\n', cond + '\n', temp + '\n', feels_like + '\n',
temp_min + '\n', temp_max + '\n', ])
    return response
except Exception as e:
    #print("Exception (weather):", e)
    pass

if __name__ == '__main__':
    w = Weather()
    w.city = 'dsfghdsgh'
    w.country = 'dsfhfkjtgk'
    print(w.get_weather())

```

## test.py

```

import unittest
from weather import Weather

class TestBot(unittest.TestCase):
    def test_1(self):
        w = Weather()
        w.city = 'Тула'
        w.country = 'Россия'
        self.assertEqual(w.get_weather().split('\n')[0], 'Tula RU')

    def test_2(self):
        w = Weather()
        w.city = 'asdklfjh'
        w.country = 'hkjagsdf'
        self.assertEqual(w.get_weather(), None)

    def test_3(self):
        w = Weather()
        w.city = 'Стамбул'
        w.country = 'Турция'
        self.assertEqual(w.get_weather().split('\n')[0], 'Istanbul TR')

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

```

## test.feature

```

Feature: Test
    Scenario: Test function
        Given Entered data
        When Check Tula Russia
        When Check None
        When Check Istanbul Turkey
        Then Completed

```

## Steps/steps.py

```

from test import TestBot
from behave import given, when, then

```

```

@given("Entered data")
def start_up(context):
    context.a = TestBot()

@when("Check Tula Russia")
def test1(context):
    context.a.test_1()

@when("Check None")
def test2(context):
    context.a.test_2()

@when("Check Istanbul Turkey")
def test3(context):
    context.a.test_3()

@then("Completed")
def check_result(context):
    pass

```

## Результаты

### Bdd:

```

puzzzik@MacBook-Air dz % behave
Feature: Test # test.feature:1

  Scenario: Test function # test.feature:2
    Given Entered data # steps/steps.py:5 0.001s
    When Check Tula Russia # steps/steps.py:10 0.199s
    When Check None # steps/steps.py:15 0.185s
    When Check Istanbul Turkey # steps/steps.py:20 0.188s
    Then Completed # steps/steps.py:25 0.000s

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
5 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.573s

```

### Tdd:

Run: Unittest weather x

▶

✓

⊗

↓½

↓¼

≡

≡

↑

↓

🔍

»

✓ Tests passed: 3 of 3 tests – 577 ms

▼ ✓ Test Results 577 ms

/usr/local/bin/python3 "/Users/puzzzik/Library/Application Support/Python/Python39-64/Scripts/python.exe" /Users/puzzzik/Library/Application Support/Python/Python39-64/Scripts/python.exe  
Testing started at 1:19 PM ...  
Launching unittests with arguments python -m unittest /Users/puzzzik/Library/Application Support/Python/Python39-64/Scripts/python.exe  
  
Ran 3 tests in 0.581s  
  
OK  
  
Process finished with exit code 0