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 ИУ5 «Системы обработки информации и управления»
Курс «Базовые компоненты интернет-технологий»

Тестирование Telegram бота

Выполнил:
Студент группы ИУ5-34Б
Сергеев Никита
Дата и подпись:

Преподаватель:
Гапанюк Ю.Е.
Дата и 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 = '5096756347:AAEyvUCnXOxvB7E6mTUYAIz3pFVIxIei_XQ'

db_file = 'db.vdb'

CURRENT_STATE = "CURRENT_STATE"

class States(Enum):
    START_STATE = "START_STATE"
    TYPE_STATE = "TYPE_STATE"
    COUNTRY_STATE = "COUNTRY_STATE"
    CONDITION_STATE = "CONDITION_STATE"
    END_STATE = "END_STATE"
```

Файл dbworker.py

```
from vedis import Vedis
import config

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

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

def make_key(message_id, state):
    return "{}_{}".format(message_id, state)
```

Файл functions.py

```
import os

cur_path = os.path.dirname(os.path.abspath(__file__))
```

```
def path_picture(type, country, condition):
    return os.path.join(cur_path, f'Мотоциклы\{type}\{country}\{condition}.jpg')
```

Файл bot.py

```
import telebot
from telebot import types
import config
import dbworker
from functions import path_picture

bot = telebot.TeleBot(config.TOKEN)

@bot.message_handler(commands=['start'])
def welcome(message):
    markup = types.ReplyKeyboardMarkup(row_width=1, resize_keyboard=True)
    markup.add(types.KeyboardButton('Привет, давай начнем'))
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.TYPE_STATE.value)
    bot.send_message(message.chat.id, 'Привет, я помогу подобрать тебе мотоцикл',
reply_markup=markup)

@bot.message_handler(func=lambda message:
dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.TYPE_STATE.value)
def type_car(message):
    if message.text in ['Кроссовый', 'Для путешествий', 'Скутеры']:
        if message.text == 'Кроссовый':
            dbworker.set(dbworker.make_key(message.chat.id, config.States.TYPE_STATE.value),
'Кроссовый')
            dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.COUNTRY_STATE.value)
        elif message.text == 'Для путешествий':
            dbworker.set(dbworker.make_key(message.chat.id, config.States.TYPE_STATE.value),
'Для путешествий')
            dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.COUNTRY_STATE.value)
        elif message.text == 'Скутеры':
            dbworker.set(dbworker.make_key(message.chat.id, config.States.TYPE_STATE.value),
'Скутеры')
            dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.COUNTRY_STATE.value)
        markup1 = types.ReplyKeyboardMarkup(row_width=3, resize_keyboard=True)
        botton1_1 = types.KeyboardButton('Германия')
        botton2_2 = types.KeyboardButton('США')
        botton3_3 = types.KeyboardButton('Япония')
        markup1.add(botton1_1, botton2_2, botton3_3)
        bot.send_message(message.chat.id, 'Выберите страну: ',
reply_markup=markup1)
    else:
        markup = types.ReplyKeyboardMarkup(row_width=3, resize_keyboard=True)
        botton1 = types.KeyboardButton('Кроссовый')
        botton2 = types.KeyboardButton('Для путешествий')
        botton3 = types.KeyboardButton('Скутеры')
```

```
        markup.add(botton1, botton2, botton3)
        bot.send_message(message.chat.id, 'Выберите предназначение мотоцикла:',
reply_markup=markup)

@bot.message_handler(func=lambda message:
dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.COUNTRY_STATE.value)
def car_price(message):
    if message.text in ['Германия', 'США', 'Япония']:
        if message.text == 'Германия':
            dbworker.set(dbworker.make_key(message.chat.id,
config.States.COUNTRY_STATE.value), 'Германия')
            dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.CONDITION_STATE.value)
        elif message.text == 'США':
            dbworker.set(dbworker.make_key(message.chat.id,
config.States.COUNTRY_STATE.value), 'США')
            dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.CONDITION_STATE.value)
        elif message.text == 'Япония':
            dbworker.set(dbworker.make_key(message.chat.id,
config.States.COUNTRY_STATE.value), 'Япония')
            dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.CONDITION_STATE.value)
        markup1 = types.ReplyKeyboardMarkup(row_width=3, resize_keyboard=True)
        botton1_1 = types.KeyboardButton('новый')
        botton2_2 = types.KeyboardButton('бу')
        markup1.add(botton1_1, botton2_2)
        bot.send_message(message.chat.id, 'Выберите состояние:', reply_markup=markup1)
    else:
        markup = types.ReplyKeyboardMarkup(row_width=3, resize_keyboard=True)
        botton1 = types.KeyboardButton('Японский')
        botton2 = types.KeyboardButton('Американский')
        botton3 = types.KeyboardButton('Немецкий')
        markup.add(botton1, botton2, botton3)
        bot.send_message(message.chat.id, 'Выберите страну-производителя:',
reply_markup=markup)


@bot.message_handler(func=lambda message:
dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.CONDITION_STATE.value)
def car_price(message):
    if message.text in ['новый', 'бу']:
        path = path_picture(dbworker.get(dbworker.make_key(message.chat.id,
config.States.TYPE_STATE.value)),
        dbworker.get(dbworker.make_key(message.chat.id, config.States.COUNTRY_STATE.value)),
        message.text)
        with open(path, 'rb') as img:
            bot.send_photo(message.chat.id, img)
        dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.END_STATE.value)
```

```
        bot.send_message(message.chat.id, 'Чтобы начать еще раз введите /start',
reply_markup=types.ReplyKeyboardRemove())
    else:
        markup1 = types.ReplyKeyboardMarkup(row_width=3, resize_keyboard=True)
        botton1_1 = types.KeyboardButton('новый')
        botton2_2 = types.KeyboardButton('бу')
        markup1.add(botton1_1, botton2_2)
        bot.send_message(message.chat.id, 'Повторите выбор:', reply_markup=markup1)


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

Файл testTDD.py

```
import unittest
from functions import path_picture

class TestTDD(unittest.TestCase):
    def test1(self):
        self.assertEqual(path_picture('Для путешествий', 'США', 'бу'),
        r'D:\лабы\5-6\Мотоциклы\Для путешествий\США\бу.jpg')
    def test2(self):
        self.assertEqual(path_picture('Кроссовый', 'Япония', 'новый'),
        r'D:\лабы\5-6\Мотоциклы\Кроссовый\Япония\новый.jpg')

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

Файл steps.py

```
from behave import given, when, then
from functions import path_picture

@given("I have the data {type}, {price} and {country}")
def have_numbers(context, type, price, country):
    context.type = type
    context.price = price
    context.country = country

@when("I find the path")
def calculate_roots(context):
    context.path = path_picture(context.type, context.price, context.country)

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

Файл testBDD.feature

```
Feature: BDD tests

    Scenario: Test test1
        Given I have the data Скутеры, Германия and новый
        When I find the path
```

```
        Then I expect the result to be D:\лабы\5-6\Мотоциклы\Скутеры\Германия\новый.jpg

    Scenario: Test test2
        Given I have the data Кроссовый, США and бу
        When I find the path
        Then I expect the result to be D:\лабы\5-6\Мотоциклы\Кроссовый\США\бу.jpg
```

Результат выполнения

```
PS D:\лабы\5-6> python testTDD.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
PS D:\лабы\5-6> behave testBDD.feature
Feature: BDD tests # testBDD.feature:1

  Scenario: Test test1                                                          # testBDD.feature:3
    Given I have the data Скутеры, Германия and новый                           # steps/steps.py:4
    When I find the path                                                        # steps/steps.py:10
    Then I expect the result to be D:\лабы\5-6\Мотоциклы\Скутеры\Германия\новый.jpg # steps/steps.py:14

  Scenario: Test test2                                                      # testBDD.feature:8
    Given I have the data Кроссовый, США and бу                             # steps/steps.py:4
    When I find the path                                                    # steps/steps.py:10
    Then I expect the result to be D:\лабы\5-6\Мотоциклы\Кроссовый\США\бу.jpg # steps/steps.py:14

1 feature passed, 0 failed, 0 skipped
2 scenarios passed, 0 failed, 0 skipped
6 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.008s
PS D:\лабы\5-6>
```