Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по домашнему заданию

Выполнил:

студент группы ИУ5-32Б

Милевич Артём Андреевич

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Подпись и дата:

Москва, 2021 г.

Описание задания:

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 main.py:

```
 from aiogram import Bot, types

from aiogram.dispatcher import Dispatcher
from aiogram.dispatcher.filters.state import StatesGroup,State
from aiogram.dispatcher.filters import Command
from aiogram.dispatcher.storage import FSMContext
from aiogram.utils import executor
from config import TOKEN
from aiogram.types import ReplyKeyboardRemove,ReplyKeyboardMarkup, KeyboardButton,
InlineKeyboardMarkup, InlineKeyboardButton,Message

from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.contrib.middlewares.logging import LoggingMiddleware


bot = Bot(token=TOKEN)
dp = Dispatcher(bot, storage=MemoryStorage())
dp.middleware.setup(LoggingMiddleware())


class Test(StatesGroup):
    Q0 = State()
    Q1 = State()
    Q2 = State()
    Q3 = State()

queen_albums={
    1 : "Sheer Heart Attack",
    2 : "A Night At The Opera",
    3 : "News Of The World"
}
rhcp_albums={
    1 : "Californication",
    2 : "By The Way",
    3 : "Stadium Arcadium"
}
green_day_albums={
    1 : "Dookie",
    2 : "Insomniac",
    3 : "American Idiot"
```

```
}

queen_albums_price={
    "Sheer Heart Attack" : 7000,
    "A Night At The Opera" : 6000,
    "News Of The World" : 6000
}
rhcp_albums_price={
    "Californication" : 4200,
    "By The Way" : 3100,
    "Stadium Arcadium" : 3900
}
green_day_albums_price={
    "Dookie" : 3400,
    "Insomniac" : 3000,
    "American Idiot" : 3500
}

def summary(first_table,first_table_price,second_table,second_table_price,third_tab
le,third_table_price,first_answer,second_answer,third_answer):
    a = first_table_price[first_table[first_answer]]
    b = second_table_price[second_table[second_answer]]
    c = third_table_price[third_table[third_answer]]
    return "Итоговая сумма заказа = " + str(a+b+c)+ "\n"

@dp.message_handler(state="*", commands=['start'])
async def starting_process(message: types.Message):
    await bot.send_message(message.from_user.id,"Привет! У тебя есть нереальная воз
можность купить по 1 альбому " +
    "величайших групп всех времён!\n\n1)Альбом группы Queen\n2)Альбом группы RHCP\n
3)Альбом группы Green Day\n\n" +
    "Чтобы начать формировать заказ напиши /order")
    await Test.Q0.set()

@dp.message_handler(state=Test.Q1, commands=['order'])

@dp.message_handler(state=Test.Q0, commands=['order'])
async def starting_process(message: types.Message,state: FSMContext):
    await bot.send_message(message.from_user.id, "Выбери альбом группы Queen:\n1)Sh
eer Heart Attack - {}\n2)A Night At The Opera - {}\n3)News Of The World - {}".forma
t(queen_albums_price[queen_albums[1]], queen_albums_price[queen_albums[2]], queen_a
lbums_price[queen_albums[3]]))
    await Test.Q1.set()

@dp.message_handler(state=Test.Q1)
async def first_choosing(message: types.Message,state: FSMContext):
    answer = int(message.text)
    if (answer != 1 and answer !=2 and answer !=3):
        return await bot.send_message(message.from_user.id,"Неправильно, попробуй е
щё раз...")
    await state.update_data(q1 = answer)
```

```
    await bot.send_message(message.from_user.id,"Выбери альбом группы RHCP:\n1)Cali
fornication - {}\n2)By The Way - {}\n3)Stadium Arcadium - {}".format(rhcp_albums_pr
ice[rhcp_albums[1]], rhcp_albums_price[rhcp_albums[2]], rhcp_albums_price[rhcp_albu
ms[3]]))
    await Test.Q2.set()

@dp.message_handler(state=Test.Q2)
async def second_choosing(message: types.Message,state: FSMContext):
    answer = int(message.text)
    if (answer != 1 and answer !=2 and answer !=3):
        return await bot.send_message(message.from_user.id,"Неправильно, попробуй е
щё раз...")
    await state.update_data(q2 = answer)
    await bot.send_message(message.from_user.id, "Выбери альбом группы Green Day:\n
1)Dookie - {}\n2)Insomniac - {}\n3)American Idiot - {}".format(green_day_albums_pri
ce[green_day_albums[1]], green_day_albums_price[green_day_albums[2]], green_day_alb
ums_price[green_day_albums[3]]))
    await Test.Q3.set()

@dp.message_handler(state=Test.Q3)
async def third_choosing(message: types.Message,state: FSMContext):
    answer = int(message.text)
    if (answer != 1 and answer !=2 and answer !=3):
        return await bot.send_message(message.from_user.id,"Неправильно, попробуй е
щё раз...")
    await state.update_data(q3 = answer)
    data = await state.get_data()
    sumcheck=summary(queen_albums,queen_albums_price,rhcp_albums,rhcp_albums_price,
green_day_albums,green_day_albums_price,data.get("q1"),data.get("q2"),data.get("q3"
))
    await bot.send_message(message.from_user.id,"Ваш заказ:\n\nАльбом Queen: {}\nАл
ьбом RHCP: {}\nАльбом Green Day: {}".format(queen_albums[data.get("q1")],rhcp_album
s[data.get("q2")],green_day_albums[data.get("q3")]))
    await bot.send_message(message.from_user.id, sumcheck)
    await Test.Q0.set()

async def shutdown(dispatcher: Dispatcher):
    await dispatcher.storage.close()
    await dispatcher.storage.wait_closed()

if __name__ == '__main__':
    executor.start_polling(dp, on_shutdown=shutdown)
```

Файл confing.py:

```
TOKEN = "5034037621:AAEGHuOcgzL6G2Oi-EGd_YWsSoGstQSVWKk"
```

Файл test_bot.py:

```
import unittest
```

```
import sys, os

sys.path.append(os.getcwd())
from main import *

test1 = 1
test2 = 2
test3 = 3

class TestGetRoots(unittest.TestCase):
    def test1_bot(self):
        res = summary(queen_albums,queen_albums_price,rhcp_albums,rhcp_albums_price
,green_day_albums,green_day_albums_price,test1,test2,test3)
        self.assertEqual("Итоговая сумма заказа = 13600\n", res)
    def test2_bot(self):
        res = summary(queen_albums,queen_albums_price,rhcp_albums,rhcp_albums_price
,green_day_albums,green_day_albums_price,test2,test1,test3)
        self.assertEqual("Итоговая сумма заказа = 13700\n", res)

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

Файл bdd_test.py:

```
from behave import Given,When,Then
from main import *

@Given("ordering albums with answers in bot queen album - {a} rhcp album - {b} gree
n day album - {c}")
def given_answers(context,a,b,c):
    context.ans1=int(a)
    context.ans2=int(b)
    context.ans3=int(c)

@When("we form summary of check")
def make_summary(context):
    res = summary(queen_albums,queen_albums_price,rhcp_albums,rhcp_albums_price,gre
en_day_albums,green_day_albums_price,context.ans1,context.ans2,context.ans3)
    context.result=res

@Then("check should be with correct price {out}")
def compare_results(context, out):
    corres = "Итоговая сумма заказа = " + str(out)+ "\n"
    assert(context.result == corres)
```

Файл bdd_test.feature:

```
Feature: Test summary
    Scenario: test summary for making check with 1 2 3
```

```
        Given ordering albums with answers in bot queen album - 1 rhcp album - 2 gr
een day album - 3
        When we form summary of check
        Then check should be with correct price 13600
    Scenario: test summary for making check with 2 1 3
        Given ordering albums with answers in bot queen album - 2 rhcp album - 1 gr
een day album - 3
        When we form summary of check
        Then check should be with correct price 13700
```

Примеры выполнения программы:

TDD:

```
(.venv) PS C:\Microsoft VS Code\python\dz\code> & "c:/Microsoft VS Code/python/dz/code/.venv/Scripts/python.exe" "c:/Microsoft VS Code/python/dz/code/test_bo
t.py"
..
----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
```

BDD:

```
(.venv) PS C:\Microsoft VS Code\python\dz\code> behave
Feature: Test summary # features/bdd_test.feature:1

  Scenario: test summary for making check with 1 2 3                                  # features/bdd_test.feature:2
    Given ordering albums with answers in bot queen album - 1 rhcp album - 2 green day album - 3 # steps/bdd_test.py:4
    When we form summary of check                                                     # steps/bdd_test.py:10
    Then check should be with correct price 13600                                     # steps/bdd_test.py:15

  Scenario: test summary for making check with 2 1 3                                  # features/bdd_test.feature:6
    Given ordering albums with answers in bot queen album - 2 rhcp album - 1 green day album - 3 # steps/bdd_test.py:4
    When we form summary of check                                                     # steps/bdd_test.py:10
    Then check should be with correct price 13700                                     # steps/bdd_test.py:15

1 feature passed, 0 failed, 0 skipped
2 scenarios passed, 0 failed, 0 skipped
6 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.004s
```