

Московский государственный технический университет им. Н.Э. Баумана

Факультет “Радиотехнический”
Кафедра ИУ5 “Системы обработки информации и управления”

Отчет по проделанному ДЗ
по курсу

Базовые компоненты интернет технологий

Вариант 14

Подготовил:

Студент группы РТ5-31Б

Мамаев Т.Э.

Проверил:

Доцент кафедры ИУ5

Гапанюк Ю.Е.

23 Декабря 2021г

Задание:

Модифицируйте код лабораторной работы No6 таким образом, чтобы он был пригоден для модульного тестирования.

Используя материалы лабораторной работы No4 создайте модульные тесты с применением TDD - фреймворка (2 теста) и BDD - фреймворка (2 теста).

Листинг

```
import logging
from aiogram import *
from aiogram.types import ReplyKeyboardRemove,
ReplyKeyboardMarkup, KeyboardButton,
InlineKeyboardMarkup, InlineKeyboardButton
from aiogram.dispatcher.filters import Text
from aiogram.dispatcher import *
from aiogram.dispatcher.filters.state import *
from aiogram.types.bot_command import *
from logging import *
import asyncio
from aiogram.contrib.fsm_storage.memory import *
import configparser

goods_name = ["видеокарта", "оперативная память",
"флешка"] vid_size = ["2Гб", "4Гб", "8Гб"]
dost = ["Москва", "Санкт-Петербург", "Казань"]
user_data=[]

a_1 = [] b_1 = [] c_1 = []

class Order(StatesGroup): waiting_goods = State() waiting_size
= State() waiting_dost = State()

async def Start(message: types.Message):
keyboard =
```

```
types.ReplyKeyboardMarkup(resize_keyboard=True) for name
in goods_name:
```

```
keyboard.add(name)
```

```
await message.answer("Выберите товар, который хотите
приобрести:",
```

```
reply_markup=keyboard)
```

```
await Order.waiting_goods.set()
```

```
async def Chose(message: types.Message, state: FSMContext):
if message.text.lower() not in goods_name:
```

```
await message.answer(f"К сожалению, данного товара нет в
наличии. \n" f"Выберете товар из списка ниже")
```

```
return
```

```
await state.update_data(chosen_good=message.text.lower())
```

```
keyboard =
```

```
types.ReplyKeyboardMarkup(resize_keyboard=True) for size in
vid_size:
```

```
keyboard.add(size)
```

```
await Order.next()
```

```
await message.answer("Выберите размер памяти:",
```

```
reply_markup=keyboard) await Order.waiting_size.set()
```

```
a = state.update_data(chosen_good=message.text.lower())
```

```
return a
```

```
def chose_good():
```

```
a = "Видеокарта" return a
```

```
def chose_size(): b = "4Гб"
```

```
return b
```

```
def chose_dost():
```

```
c = "москва" return c
```

```
async def Size_chose(message: types.Message, state:
FSMContext): if message.text.lower() not in vid_size:
```

```
await message.answer(f"К сожалению, данного объема нет в
наличие. \n" f"Выберите объем из списка ниже")
```

```
return
```

```
await state.update_data(chosen_size=message.text.lower())
```

```
keyboard =
```

```
types.ReplyKeyboardMarkup(resize_keyboard=True) for d in
dost:
```

```
keyboard.add(d)
```

```
await Order.next()
```

```
await message.answer("Выберите город доставки:",
reply_markup=keyboard) b = message.text.lower
return message.text.lower()
```

```
async def Dost_chose(message: types.Message, state:
FSMContext): user_data = await state.get_data()
```

```
await message.answer(f"Вы заказали
{user_data['chosen_good']} на {user_data['chosen_size']} в
город {message.text.lower()}! \n"
```

```
f"Спасибо за заказ!") await state.finish()
```

```
return message.text.lower() def a(a_1):
```

```
return a_1 def b(b_1):
```

```
return b_1 def c(c_1):
```

```
return c_1 def print():
```

```

global user_data
global a_1, b_1, c_1
A=a
B=b
C=c
return "Вы заказали {} на {} в город {}".format(A,B,C)

def register_handlers_food(dp: Dispatcher):
    dp.register_message_handler(Start, commands="go", state="*")
    dp.register_message_handler(Chose,
    state=Order.waiting_goods)
    dp.register_message_handler(Size_chose,
    state=Order.waiting_size)
    dp.register_message_handler(Dost_chose,
    state=Order.waiting_dost)

async def bot_start(message: types.Message, state:
FSMContext): await state.finish()
await message.answer(

'Это магазин комплектующих, нажмите кнопку "/go" чтобы
перейти к товарам',

reply_markup=types.ReplyKeyboardRemove() )

def register_handlers_common(dp: Dispatcher):
    dp.register_message_handler(bot_start, commands="start",
    state="*")

logger = logging.getLogger(__name__)

```

TDD_test.py

```

import unittest import main_DZ

class Test(unittest.TestCase): def TestBot1(self):

```

```
self.assertEqual(main_DZ.chose_good(), "видеокарта")
self.assertEqual(main_DZ.chose_dost(), "москва")
self.assertEqual(main_DZ.chose_size(), "4гб")
```

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

BDD.py

```
from behave import given, when import main_DZ
```

```
@given('I send bot message {start}') def step_impl(context,
start: str):
```

```
main_DZ.cmd_start(start)
```

```
@when('I send bot first message {go}') def step_imp2(context,
go: str):
```

```
context.firstNum = main_DZ.cmd_start(go)
```

BDD.features

Feature: testing bot Scenario: start bot

Given I send bot message /start When I send bot first message /
go