



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»

ДИСЦИПЛИНА:

«БКИТ»

Домашнее задание

Студент Распашнов А.А. ИУ5Ц-54Б

(И.О. Фамилия) (Группа)

(Подпись, дата)

Преподаватель Гапанюк Ю.Е.

(И.О. Фамилия)

(Подпись, дата)

Оглавление

Элементы оглавления не найдены.

Домашнее задание

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).

ДЗ – код



test_bot.py

```
import unittest
from bot import TOKEN
from handlers.food import available_burger_names, available_drink_names,
available_fries_names

class TestBotWork(unittest.TestCase):
    # список еды, напитков и прочего может подгружаться из баз данных,
    # поэтому важно знать, что список
    # возможных продуктов не пустой
    def test_bot_food_not_empty(self):
        self.assertNotEqual(0, len(available_burger_names))

    def test_bot_drinks_not_empty(self):
        self.assertNotEqual(0, len(available_drink_names))

    def test_bot_fries_not_empty(self):
        self.assertNotEqual(0, len(available_fries_names))

    # важно знать, что токен бота корректен, можно проверять его длину
    def test_bot_token_exist(self):
        self.assertEqual(len(TOKEN), 46)

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



bot.py

```
import asyncio
import logging
import types

from aiogram import Bot
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.types import BotCommand

from handlers.common import *
from handlers.food import *
```

```

class OrderFood(StatesGroup):
    waiting_for_burger_name = State()
    waiting_for_fries_name = State()
    waiting_for_drink_name = State()

    async def food_start(message: types.Message):
        keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
        for name in available_burger_names:
            keyboard.add(name)
        await message.answer("Выберите, с чем вы хотите бургер:",
            reply_markup=keyboard)
        await OrderFood.waiting_for_burger_name.set()

    async def food_chosen(message: types.Message, state: FSMContext):
        if message.text.lower() not in available_burger_names:
            await message.answer("Пожалуйста, выберите мясо в бургере, используя
клавиатуру ниже.")
            return
        await state.update_data(chosen_burger=message.text.lower())

        keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
        for name in available_fries_names:
            keyboard.add(name)
        await OrderFood.next()
        await message.answer("Теперь выберите вид картошки:",
            reply_markup=keyboard)

    async def food_fries_chosen(message: types.Message, state: FSMContext):
        if message.text.lower() not in available_fries_names:
            await message.answer("Пожалуйста, выберите вид картошки, используя
клавиатуру ниже.")
            return
        await state.update_data(chosen_fries=message.text.lower())

        keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
        for name in available_drink_names:
            keyboard.add(name)
        await OrderFood.next()
        await message.answer('Теперь выберите напиток:', reply_markup=keyboard)

    async def food_drink_chosen(message: types.Message, state: FSMContext):
        if message.text.lower() not in available_drink_names:
            await message.answer("Пожалуйста, выберите напиток, используя
клавиатуру ниже.")
            return
        user_data = await state.get_data()
        await message.answer(f"Вы заказали бургер
{user_data['chosen_burger'] [2:]}, \n"
            f"картофель-{user_data['chosen_fries']} и напиток -
{message.text.lower()}. \n"
            f"Спасибо за ваш заказ! ",
            reply_markup=types.ReplyKeyboardRemove())
        await state.finish()

    def register_handlers_food(dp: Dispatcher):
        dp.register_message_handler(food_start, commands="food", state="*")
        dp.register_message_handler(food_chosen,
            state=OrderFood.waiting_for_burger_name)

```

```

        dp.register_message_handler(food_fries_chosen,
state=OrderFood.waiting_for_fries_name)
        dp.register_message_handler(food_drink_chosen,
state=OrderFood.waiting_for_drink_name)

logger = logging.getLogger(__name__)

# Регистрация команд, отображаемых в интерфейсе Telegram
async def set_commands(bot: Bot):
    commands = [
        BotCommand(command="/food", description="Заказать блюда"),
        BotCommand(command="/cancel", description="Отменить текущее
действие")
    ]
    await bot.set_my_commands(commands)

TOKEN = '5093952542:AAEfnITGgt2MSjOUOlQapYGUYWqiJdY2QJY'

async def main():
    logging.basicConfig(
        level=logging.INFO,
        format="%(asctime)s - %(levelname)s - %(name)s - %(message)s",
    )
    logger.error("Starting bot")

    # Объявление и инициализация объектов бота и диспетчера
    bot = Bot(token=TOKEN)
    dp = Dispatcher(bot, storage=MemoryStorage())

    # Регистрация хэндлеров
    register_handlers_common(dp)

    register_handlers_food(dp)

    # Установка команд бота
    await set_commands(bot)

    # Запуск поллинга
    await dp.skip_updates()
    await dp.start_polling()

if __name__ == '__main__':
    asyncio.run(main())

```

Результат ДЗ:

```
C:\Users\arsec\OneDrive\Desktop\DZ\DZ\venv\Scripts\python.exe  
"S:\Pyh\PyCharm 2021.3\plugins\python\helpers\pycharm\_jb_unittest_runner.py"  
--path C:/Users/arsec/OneDrive/Desktop/DZ/DZ/test_bot.py
```

Testing started at 17:07 ...

```
Launching unittests with arguments python -m unittest  
C:/Users/arsec/OneDrive/Desktop/DZ/DZ/test_bot.py in  
C:\Users\arsec\OneDrive\Desktop\DZ\DZ
```

Ran 4 tests in 0.010s

OK

Process finished with exit code 0