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»

Домашняя работа

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:
студент группы ИУ5-33Б
Некрасов С. А.

Проверил:
Канев А.И.

2021 г.

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:

Файл common.py:

```
from aiogram import Dispatcher, types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters import Text


async def cmd_start(message: types.Message, state: FSMContext):
    await state.finish()
    await message.answer(
        "Купить билет /ticket",
        reply_markup=types.ReplyKeyboardRemove()
    )


async def cmd_cancel(message: types.Message, state: FSMContext):
    await state.finish()
    await message.answer("Действие отменено",
reply_markup=types.ReplyKeyboardRemove())



def register_handlers_common(dp: Dispatcher, admin_id: int):
    dp.register_message_handler(cmd_start, commands="start", state="*")
    dp.register_message_handler(cmd_cancel, commands="cancel", state="*")
    dp.register_message_handler(cmd_cancel, Text(equals="отмена",
ignore_case=True), state="*")
```

Файл ticket.py:

```
from aiogram import Dispatcher, types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup

available_city_name = ["Анапа", "Сочи", "Ялта"]
available_day_quantity = ["5", "7", "9"]
available_trip_method = ["самолёт", "поезд", "яхта"]


class OrderTicket(StatesGroup):
    waiting_for_city_name = State()
    waiting_for_day_quantity = State()
    waiting_for_trip_method = State()


async def ticket_start(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in available_city_name:
        keyboard.add(name)
    await message.answer("Выберите город:", reply_markup=keyboard)
```

```
    await OrderTicket.waiting_for_city_name.set()


async def ticket_chosen(message: types.Message, state: FSMContext):
    if ticket_check(message.text):
        await message.answer("В данный город нет билетов, выберите город,
используя клавиатуру ниже")
        return
    await state.update_data(chosen_ticket=message.text)

    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for quantity in available_day_quantity:
        keyboard.add(quantity)
    await OrderTicket.next()
    await message.answer("Теперь выберите количество дней:",
reply_markup=keyboard)


async def ticket_quantity_chosen(message: types.Message, state: FSMContext):
    if day_check(message.text):
        await message.answer(
            "На такое количество дней нет билетов, выберите количество дней,
используя клавиатуру ниже")
        return
    await state.update_data(chosen_days=message.text)

    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for method in available_trip_method:
        keyboard.add(method)
    await OrderTicket.next()
    await message.answer("Теперь выберите способ поездки:",
reply_markup=keyboard)


async def ticket_method_chosen(message: types.Message, state: FSMContext):
    if trip_check(message.text):
        await message.answer("Такого способа поездки нет, выберите способ
поездки, используя клавиатуру ниже")
        return
    user_data = await state.get_data()
    await message.answer(f"Вы отдохнёте {user_data['chosen_days']} дней в
городе {user_data['chosen_ticket']}. "
                         f"Ваш способ поездки: {message.text}\n",
                         reply_markup=types.ReplyKeyboardRemove())
    await state.finish()


def register_handlers_ticket(dp: Dispatcher):
    dp.register_message_handler(ticket_start, commands="ticket", state="*")
    dp.register_message_handler(ticket_chosen,
state=OrderTicket.waiting_for_city_name)
    dp.register_message_handler(ticket_quantity_chosen,
state=OrderTicket.waiting_for_day_quantity)
    dp.register_message_handler(ticket_method_chosen,
state=OrderTicket.waiting_for_trip_method)


def ticket_check(text):
    if text not in available_city_name:
        return 1
    else:
        return 0
```

```
def day_check(text):
    if text not in available_day_quantity:
        return 1
    else:
        return 0


def trip_check(text):
    if text not in available_trip_method:
        return 1
    else:
        return 0
```

Файл config.py:

```
import configparser
from dataclasses import dataclass


@dataclass
class TgBot:
    token: str
    admin_id: int


@dataclass
class Config:
    tg_bot: TgBot


def load_config(path: str):
    config = configparser.ConfigParser()
    config.read(path)

    tg_bot = config["tg_bot"]

    return Config(
        tg_bot=TgBot(
            token=tg_bot["token"],
            admin_id=int(tg_bot["admin_id"])
        )
    )
```

Файл bot.ini:

```
[tg_bot]
token = 5003047874:AAHmH7rfUJmXdg5vyrDnaddK1fUNSxBNbac
admin_id = 987654321
```

Файл bot.py:

```
import asyncio
import logging

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

from app.config import load_config
from app.handlers.ticket import register_handlers_ticket
```

```
from app.handlers.common import register_handlers_common

logger = logging.getLogger(__name__)


async def set_commands(bot: Bot):
    commands = [
        BotCommand(command="/ticket", description="Купить билет"),
        BotCommand(command="/cancel", description="Отменить текущее
действие")
    ]
    await bot.set_my_commands(commands)


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

    # Парсинг файла конфигурации
    config = load_config("config/bot.ini")

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

    # Регистрация хэндлеров
    register_handlers_common(dp, config.tg_bot.admin_id)
    register_handlers_ticket(dp)

    # Установка команд бота
    await set_commands(bot)

    # Запуск поллинга
    # await dp.skip_updates()  # пропуск накопившихся апдейтов
(необязательно)
    await dp.start_polling()


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

requirements

```
aiogram==2.17.1
```

test_TDD

```
import unittest  # автоматизация тестов
import sys, os  # предоставляет системе особые параметры и функции

from app.handlers.ticket import *

sys.path.append(os.getcwd())  # добавить путь поиска модулей


class TestBot(unittest.TestCase):
    def test_ticket(self):
```

```
        self.assertEqual(ticket_check("Сочи"), 0)
        self.assertEqual(ticket_check("Липецк"), 1)

    def test_day(self):
        self.assertEqual(day_check("7"), 0)
        self.assertEqual(day_check("4"), 1)

    def test_trip(self):
        self.assertEqual(trip_check("поезд"), 0)
        self.assertEqual(trip_check("пешком"), 1)
```

my_feature.feature

```
Feature: Test
  Scenario: Test bot
    Given bot
    When ticket_check return OK
    And day_check return OK
    And trip_check return OK
    Then Successfully
```

test_BDD

```
from behave import *

from test_tdd.test_TDD import *


@given('bot')
def first_step(context):
    context.a = TestBot()


@when('ticket_check return OK')
def test_ticket(context):
    context.a.test_ticket()


@when('day_check return OK')
def test_day(context):
    context.a.test_day()


@when('trip_check return OK')
def test_trip(context):
    context.a.test_trip()


@then('Successfully')
def last_step(context):
    pass
```

Экранные формы с примерами выполнения программы:

test_TDD

```
C:\Users\79508\PycharmProjects\Lab6\venv\Scripts\python.exe "C:\Program Fi
Testing started at 23:15 ...
Launching unittests with arguments python -m unittest test_TDD.TestBot in



Ran 3 tests in 0.002s

OK

Process finished with exit code 0
```

test_BDD

```
PS C:\Users\79508\PycharmProjects\Lab6> behave features\my_feature.feature
Feature: Test # features/my_feature.feature:1

  Scenario: Test bot             # features/my_feature.feature:2
    Given bot                    # features/steps/test_BDD.py:6
    When ticket_check return OK  # features/steps/test_BDD.py:11
    And day_check return OK      # features/steps/test_BDD.py:16
    And trip_check return OK     # features/steps/test_BDD.py:21
    Then Successfully            # features/steps/test_BDD.py:26

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
5 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.000s
PS C:\Users\79508\PycharmProjects\Lab6>
```