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 Бабаян А.А. ИУ5Ц-52Б _____

(И.О. Фамилия) (Группа) (Подпись, дата)

Преподаватель Гапанюк Ю.Е. _____

(И.О. Фамилия) (Подпись, дата)

2021г.

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).

Файл **main.py:**

```
from aiogram.utils import executor
from create_bot import dp
from data_base import sqlite_db

async def on_startup(_):
    print('Бот вышел в онлайн!')
    sqlite_db.sql_start()


from handlers import client, admin

client.register_handlers_client(dp)
admin.register_handlers_admin(dp)
executor.start_polling(dp, skip_updates=True, on_startup=on_startup)
```

Файл **create_bot.py:**

```
from aiogram import Bot, types
from aiogram.dispatcher import Dispatcher
from aiogram.contrib.fsm_storage.memory import MemoryStorage
import os

storage = MemoryStorage()

bot = Bot('5061919999:AAEoaBVZ99k49gLuYo0WOnlEwwtGzQ77XpQ')
dp = Dispatcher(bot, storage=storage)
```

Папка **keyboards** \ Файл**__init__.py:**

```
from keyboards.client_kb import kb_client
```

Папка **keyboards** \ Файл **admin_kb.py:**

```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton,
ReplyKeyboardRemove

button_load = KeyboardButton('/Загрузить')
#button_delete = KeyboardButton('/Удалить')

button_case_admin =
ReplyKeyboardMarkup(resize_keyboard=True).add(button_load)
```

Папка **keyboards** \ Файл **client_kb.py:**

```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton,
ReplyKeyboardRemove

b1 = KeyboardButton('/Режим_работы')
b2 = KeyboardButton('/Адрес')
b3 = KeyboardButton('/Информация')

kb_client = ReplyKeyboardMarkup(resize_keyboard=True)

kb_client.row(b1, b2, b3)
```

Папка **handlers** \ Файл**__init__.py:**

```
from handlers import client
from handlers import admin
```

Папка **handlers** \ Файл **admin.py:**

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

from create_bot import dp, bot
from data_base import sqlite_db


from keyboards import admin_kb

ID = None

class FSMAdmin(StatesGroup):
    photo = State()
    name = State()
    #air = State()
    description = State()
    price = State()


#Получаем ID текущего модератора
#@dp.message_handler(commands=['moderator'], is_chat_admin=True)
async def make_changes_command(message: types.Message):
    global ID
    ID = message.from_user.id
    await bot.send_message(message.from_user.id, 'Вы вошли в режим
администратора',
                           reply_markup=admin_kb.button_case_admin)
    await message.delete()




#@dp.message_handler(commands='Загрузить', state=None)
async def cm_start(message: types.Message):
    if message.from_user.id == ID:
        await FSMAdmin.photo.set()
        await message.reply('Загрузите фото')

#@dp.message_handler(content_types=['photo'], state=FSMAdmin.photo)
```

```
async def load_photo(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['photo'] = message.photo[0].file_id
        await FSMAdmin.next()
        await message.reply("Введите название самолета")

#@dp.message_handler(state=FSMAdmin.name)
async def load_name(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['name'] = message.text
        await FSMAdmin.next()
        await message.reply("Введите авиакомпанию")

#@dp.message_handler(state=FSMAdmin.air)
#async def load_air(message: types.Message, state: FSMContext):
 #   if message.from_user.id == ID:
  #      async with state.proxy() as data:
   #         data['air'] = message.text
    #    await FSMAdmin.next()
     #    await message.reply("Введите описание самолета")

#@dp.message_handler(state=FSMAdmin.description)
async def load_description(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['description'] = message.text
        await FSMAdmin.next()
        await message.reply("Введите количество самолетов")

#@dp.message_handler(state=FSMAdmin.price)
async def load_price(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['price'] = float(message.text)

        await sqlite_db.sql_add_command(state)
        await state.finish()



def register_handlers_admin(dp : Dispatcher):
    dp.register_message_handler(cm_start, commands=['Загрузить'], state=None)
    dp.register_message_handler(load_photo, content_types=['photo'],
state=FSMAdmin.photo)
    dp.register_message_handler(load_name, state=FSMAdmin.name)
    #dp.register_message_handler(load_air, state=FSMAdmin.air)
    dp.register_message_handler(load_description, state=FSMAdmin.description)
    dp.register_message_handler(load_price, state=FSMAdmin.price)
    dp.register_message_handler(make_changes_command, commands=['admin'],
is_chat_admin=True)
```

Папка **handlers** \ Файл **client.py:**

```
from aiogram import types, Dispatcher
from create_bot import dp, bot
from keyboards import kb_client
from aiogram.types import ReplyKeyboardRemove
from data_base import sqlite_db

#@dp.message_handler(commands=['start', 'help'])
```

```
async def command_start(message : types.Message):
    try:
        await bot.send_message(message.from_user.id, 'Здравствуйте! Вас
приветствует бот-помощник аэропорта Газолин! '
                                                     'Что бы Вы хотели
узнать?', reply_markup=kb_client)
        await message.delete()
    except:
        await message.reply('Общение с ботом через ЛС, напишите ему')


#@dp.message_handler(commands=['Режим_работы'])
async def airport_open_start(message: types.Message):
    await bot.send_message(message.from_user.id, 'Круглосуточно')


#@dp.message_handler(commands=['Адрес_аэропорта'])
async def airport_adress(message: types.Message):
    await bot.send_message(message.from_user.id, 'Аэропорт Газолин,
Грушевский район')


async def airport_information(message: types.Message):
    await bot.send_message(message.from_user.id, '\nИНФОРМАЦИЯ ОБ
АЭРОПОРТЕ:\n'

                                                 '\n - Аэропорт Газолин,
Грушевский район'
                                                 '\n - Дата основания:
12.12.2001 г.'
                                                 '\n - Годовой
пассажиропоток: 55 млн. пассажиров в год'
                                                 '\n - Количество терминалов:
3'

                                                 '\nТЕРМИНАЛЫ:\n'

                                                 '\n 1. Терминал A:\n'
                                                 '\n - Дата основания:
12.12.2001 г.'
                                                 '\n - Годовой
пассажиропоток: 15 млн. пассажиров в год'
                                                 '\n - Количество телетрапов:
9'
                                                 '\n - Количество стоянков:
10'
                                                 '\n - Принимаемые воздушные
суда:'
                                                 '\n a. Airbus A
319/320/321/320 NEO/321 NEO/340/350'
                                                 '\n b. Boeing
737/747/777/787'
                                                 '\n - Для международных
полетов'

                                                 '\n 2. Терминал B:\n'
                                                 '\n - Дата основания:
08.06.2011 г.'
                                                 '\n - Годовой
пассажиропоток: 30 млн. пассажиров в год'
                                                 '\n - Количество телетрапов:
5'
                                                 '\n - Количество стоянков:
8'
```

```
                                                '\n - Принимаемые воздушные
суда:'
                                                '\n a. Airbus A
319/320/321/320 NEO/321 NEO/340/350'
                                                '\n b. Boeing
737/747/777/787'
                                                '\n - Для международных
полетов'

                                                '\n 2. Терминал C:\n'
                                                '\n - Дата основания:
13.06.2021 г.'
                                                '\n - Годовой
пассажиропоток: 10 млн. пассажиров в год'
                                                '\n - Количество телетрапов:
11'
                                                '\n - Количество стоянков:
15'
                                                '\n - Принимаемые воздушные
суда:'
                                                '\n a. Airbus A
319/320/321/320 NEO/321 NEO/340/350'
                                                '\n b. Boeing
737/747/777/787'
                                                '\n - Для внутренних
полетов')

@dp.message_handler(commands=['Самолеты'])
async def info_planes(message : types.Message):
    await sqlite_db.sql_read(message)

def register_handlers_client(dp : Dispatcher):


    dp.register_message_handler(command_start, commands=['start', 'help'])
    dp.register_message_handler(airport_open_start,
commands=['Режим_работы'])
    dp.register_message_handler(airport_adress, commands=['Адрес'])
    dp.register_message_handler(airport_information, commands=['Информация'])
    dp.register_message_handler(info_planes, commands=['Самолеты'])
```

Папка **data_base** \ Файл**__init__.py:**

```
from data_base import sqlite_db
```

Папка **data_base** \ Файл **sqlite_db.py:**

```
import sqlite3 as sq
from create_bot import bot, dp

def sql_start():
    global base, cur
    base = sq.connect('plane.db')
    cur = base.cursor()
    if base:
        print('База Данных активирована')
    base.execute('CREATE TABLE IF NOT EXISTS menu(img TEXT, name TEXT PRIMARY
KEY, description TEXT, price TEXT)')
    base.commit()

async def sql_add_command(state):
```

```
    async with state.proxy() as data:
        cur.execute('INSERT INTO menu VALUES (?, ?, ?, ?)',
tuple(data.values()))
    base.commit()


async def sql_read(message):
    for ret in cur.execute('SELECT * FROM menu').fetchall():
        await bot.send_photo(message.from_user.id, ret[0],
f'{ret[1]}\nАвиакомпания: {ret[2]}\nКоличество {ret[-1]}')
```

sqlite_db_test.py

```
from data_base import sqlite_db
import pytest


def test_throws():
    with pytest.raises(TypeError):
        sqlite_db.sql_start(None)


def test_start_successfully():
    base = sqlite_db.sql_start(':memory:')
    result = base.execute('SELECT count(name) FROM sqlite_master WHERE
type=\'table\' AND name=\'menu\';').fetchone()
    assert 1 == result[0]
```

Скриншоты:

```
============================= test session starts ==============================
collecting ... collected 2 items

sqlite_db_test.py::test_throws PASSED                                    [ 50%]
sqlite_db_test.py::test_start_successfully PASSED                        [100%]Database is connected


============================== 2 passed in 0.01s ===============================

Process finished with exit code 0
```