



**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 «ГУИМЦ»

Кафедра ИУ5 «Системы обработки информации и управления»

Дисциплина «Базовые компоненты ИТ»

ОТЧЕТ

ПО ДЗ

Студент: Близнева А.Е., группа ИУ5Ц-51Б

Преподаватель: Гапанюк Ю.Е.

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).

bot_telegram.py

```
from aiogram.utils import executor
from create_bot import dp
from handlers import client, admin
from data_base import sqlite_db

connectionString = 'tour.db'

async def on_startup(_):
    print('Бот в сети')
    sqlite_db.sql_start(connectionString)

client.register_handlers_client(dp)
admin.register_handlers_admin(dp)

executor.start_polling(dp, skip_updates=True, on_startup=on_startup)
```

create_bot.py

```
import os

from aiogram import Bot
from aiogram.dispatcher import Dispatcher
from aiogram.contrib.fsm_storage.memory import MemoryStorage

storage = MemoryStorage()

bot = Bot(token=os.getenv('TOKEN'))
dp = Dispatcher(bot, storage=storage)
```

Папка data_base

__init__.py

```
from data_base import sqlite_db
```

sqlite_db.py

```
import sqlite3 as sq

def sql_start(connectionString):
    global base, cur
    base = sq.connect(connectionString)
    cur = base.cursor()
    if base:
        print('Database is connected')
        base.execute(
            'CREATE TABLE IF NOT EXISTS menu(img TEXT, name TEXT PRIMARY KEY,
            decription TEXT, price TEXT)')
        base.commit()

    return base
```

```

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, bot):
    for ret in cur.execute('SELECT * FROM menu').fetchall():
        await bot.send_photo(message.from_user.id, ret[0],
                              f'{ret[1]}\nОписание: {ret[2]}\nЦена {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]

```

Панка handlers

__init__.py

```

from handlers import client
from handlers import admin

```

admin.py

```

import string

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

from data_base import sqlite_db
from create_bot import bot
from keyboards import adm_kb

ID = None

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

```

```

#Получаем ID текущего модератора
async def make_changes_command(message: types.Message):
    global ID
    ID = message.from_user.id
    await bot.send_message(message.from_user.id, 'Вы зашли в систему как
модератор', reply_markup=adm_kb.button_case_admin)
    await message.delete()

#Начало диалога загрузки нового пункта меню
async def cm_start(message: types.Message):
    if message.from_user.id == ID:
        await FSMAdmin.photo.set()
        await message.reply('Загрузите фото',
reply_markup=ReplyKeyboardRemove())

#Получаем первый ответ и пишем в словарь
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")

#Получаем второй ответ
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 описание")

#Получаем третий ответ
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("Теперь укажите цену",
reply_markup=adm_kb.button_case_admin)

#Получаем последний ответ и используем полученные данные
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 complete(name: string)-> string:
    return f'{name} completed'

'''#Выход из состояний
#dp.message_handler(state="*", commands='отмена')
#dp.message_handler(Text(equals='отмена', ignore_case=True), state="*")
async def cancel_handler(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        current_state = await state.get_state()
        if current_state is None:
            return

```

```

        await state.finish()
        await message.reply('OK') '''

#Регистрируем хендлеры
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_description, state=FSMAdmin.description)
    dp.register_message_handler(load_price, state=FSMAdmin.price)
    dp.register_message_handler(make_changes_command, commands='moderator',
is_chat_admin=True)

```

client.py

```

from aiogram import types, Dispatcher
from create_bot import bot
from keyboards import kb_client
from aiogram.types import ReplyKeyboardRemove
from data_base import sqlite_db
import sqlite3 as sq

async def command_start(message: types.Message):
    try:
        await bot.send_message(message.from_user.id, 'Здравствуйте! Вас
приветствует тур-бот', reply_markup=kb_client)
    except:
        await message.reply('Общение с ботом через ЛС, напишите ему /ссылка
на бота/')

async def tour_open_command(message: types.Message):
    await bot.send_message(message.from_user.id, 'Пн-пт 9:00-20:00')

async def tour_place_command(message: types.Message):
    await bot.send_message(message.from_user.id, 'м.Бауманская')

async def tour_remove_command(message: types.Message):
    await bot.send_message(message.from_user.id, 'Успешно',
reply_markup=ReplyKeyboardRemove())

async def tour_menu_command(message: types.Message):
    await sqlite_db.sql_read(message, bot)

def register_handlers_client(dp: Dispatcher):
    dp.register_message_handler(command_start, commands=['start', 'help'])
    dp.register_message_handler(tour_open_command, commands=['Режим_работы'])
    dp.register_message_handler(tour_place_command,
commands=['Расположение'])
    dp.register_message_handler(tour_remove_command, commands=['Завершить'])
    dp.register_message_handler(tour_menu_command,
commands=['Посмотреть_туры'])
    dp.register_message_handler(tour_menu_command, commands=['Check'])

```

Панка keyboards

__init__.py

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

adm_kb.py

```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

#Кнопки клавиатуры админа
button_load = KeyboardButton('/Загрузить')
'''button_delete = KeyboardButton('/Удалить')'''

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

client_kb.py

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

b1 = KeyboardButton('/Режим_работы')
b2 = KeyboardButton('/Расположение')
b3 = KeyboardButton('/Посмотреть_туры')
b4 = KeyboardButton('/Завершить')

kb_client = ReplyKeyboardMarkup(resize_keyboard=True)

kb_client.row(b1, b2).row(b3, b4)
```

```
===== 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
```