Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по домашнему заданию

Выполнил:
студент группы ИУ5-31Б
Зелинский Даниил Михайлович

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич

Подпись и дата:

Подпись и дата:

Москва, 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).

Код.

vzaimoImmporty.py

```
from aiogram import Bot
from aiogram.dispatcher import Dispatcher
import os
from aiogram.contrib.fsm_storage.memory import MemoryStorage
storage = MemoryStorage()

#tok=os.getenv('TOKEN')
#print(tok)

#bot = Bot(token=os.getenv('TOKEN'))
bot=Bot(token="$$$$$$$$$$$$$$$$$$$$$$$")

dp = Dispatcher(bot, storage=storage)
```

steps.py

```
from handlers.admin import message_about_deletion
from handlers.client import message_about_buying
from handlers.other import just_answerer
from behave import given, when, then

@given(u'There is a painting named "{painting_name}"')
def step_impl(context, painting_name:str):
    context.painting_name=painting_name

@when(u'The painting is bought')
def step_impl(context):
    context.result=message_about_buying(context.painting_name)

@when(u'The painting is deleted')
def step_impl(context):
    context.result=message_about_deletion(context.painting_name)

@then(u'Send the message: {result}')
def step_impl(context, result):
    assert context.result==result

@when(u'The message is got: {message}')
def step_impl(context, message):
    context.result=just_answerer(message)
```

bdd.feature

```
Feature: bot testing
Scenario: generating buying message
  Given There is a painting named "Jolyne Cujoh"
  When The painting is bought
  Then Send the message: You have bought "Jolyne Cujoh".
Scenario: generating deleting message
  Given There is a painting named "Jolyne Cujoh"
  When The painting is deleted
```

```
  Then Send the message: "Jolyne Cujoh" has been deleted.
Scenario: chat testing
  When The message is got: Hello!
  Then Send the message: Hello there! How are you?
  When The message is got: I am fine, thanks!
  Then Send the message: What a nice day to see you here!
  When The message is got: You too!
  Then Send the message: :^)
```

test.py

```
import unittest
#from admin import message_about_deletion
from handlers.admin import message_about_deletion
from handlers.client import message_about_buying
from handlers.other import just_answerer

class MyTestCase(unittest.TestCase):
    def setUp(self):
        self.painting_name='Jolyne Cujoh'
    def test_message_about_deletion(self):
        self.assertEqual(message_about_deletion(self.painting_name),
f'"{self.painting_name}" has been deleted.')
        #self.assertEqual(True, True)
    def test_message_about_buying(self):
        self.assertEqual(message_about_buying(self.painting_name), f'You have
bought "{self.painting_name}".')
    def test_simple_answers(self):
        self.assertEqual(just_answerer('Hello!'), 'Hello there! How are
you?')
        self.assertEqual(just_answerer('I am fine, thanks!'), 'What a nice
day to see you here!')
        self.assertEqual(just_answerer('You too!'), ':^)')
if __name__ == '__main__':
    unittest.main()
```

TelBot.py

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

async def on_startup(_):
    print('Bot is online.')
    sqlite_db.sql_start()


from handlers import client, admin, other

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

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

sqlite_db.py

```
import sqlite3 as sq
from vzaimoImporty import bot
def sql_start():
    global base, cur
    base = sq.connect('MyLittle_DataBase.db')
    cur=base.cursor()
    if base:
        print('Data base is connected.')
    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]}\nDescription: {ret[2]}\nPrice ${ret[-1]}')

async def sql_read2():
    return cur.execute('SELECT * FROM menu').fetchall()

async def sql_delete_command(data):
    cur.execute('DELETE FROM menu WHERE name == ?', (data,))
    base.commit()
```

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 vzaimoImporty import dp, bot
from data_base import sqlite_db
from keyboards import admin_kb
from aiogram.types import InlineKeyboardButton, InlineKeyboardMarkup
import string
ID=None

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

#getting moderator's ID
async def make_changes_command(message: types.Message):
    global ID
    ID=message.from_user.id
    await bot.send_message(message.from_user.id, 'At your service, Master',
reply_markup=admin_kb.button_case_admin)
    await message.delete()


#start of the job
async def cm_start(message: types.Message):
    if message.from_user.id==ID:
        await FSMAdmin.photo.set()
        await message.reply('Insert an image')

#stop the states
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:
            await message.reply('WTF BRO')
            return
        await state.finish()
        await message.reply('Ok')
```

```python
#catching the first answer of the user
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('Insert the name')

#catching the second answer of the user
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('Insert the description')

#catching the third answer of the user
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('Insert the price')

#catching the last answer of the user and using the given data
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)
       # async with state.proxy() as data:
       #     await message.reply(str(data))
       await sqlite_db.sql_add_command(state)
       await state.finish()

@dp.callback_query_handler(lambda x: x.data and x.data.startswith('del '))
async def del_callback_run(callback_query: types.CallbackQuery):
    await sqlite_db.sql_delete_command(callback_query.data.replace('del ',
''))
#    await callback_query.answer(text=f'{callback_query.data.replace("del ",
"")} has been deleted.', show_alert=True)
    await
callback_query.answer(text=message_about_deletion(callback_query.data.replace
("del ", "")), show_alert=True)

@dp.message_handler(commands='Delete')
async def delete_item(message: types.Message):
    if message.from_user.id==ID:
        read=await sqlite_db.sql_read2()
        for ret in read:
            await bot.send_photo(message.from_user.id, ret[0], f'{ret[1]}\n
Description: {ret[2]}\n Price: ${ret[-1]}')
            await bot.send_message(message.from_user.id, text='^^^',
reply_markup=InlineKeyboardMarkup().\
                                   add(InlineKeyboardButton(f'Delete
{ret[1]}', callback_data=f'del {ret[1]}')))

#registering the handlers
def register_handlers_admin(dp: Dispatcher):
    dp.register_message_handler(cm_start, commands='Load', state=None)
    dp.register_message_handler(cancel_handler, Text(equals='Cancel',
ignore_case=True), state="*")
```

```
    dp.register_message_handler(make_changes_command,commands=['moderator'],
is_chat_admin=True)
    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(cancel_handler, state="*", commands='Cancel')
    # dp.register_message_handler(del_callback_run, lambda x: x.data and
x.data.startswith('del '))
    # dp.register_message_handler(delete_item, commands='Delete')

def message_about_deletion(painting_name:string)->string:
    return f'"{painting_name}" has been deleted.'
```

client.py

```
from aiogram import types, Dispatcher
from vzaimoImporty import dp,bot
from keyboards import kb_client
from data_base import sqlite_db
from aiogram.types import InlineKeyboardButton, InlineKeyboardMarkup
import string

# @dp.message_handler(commands=['start','help'])
async def command_start(message : types.Message):
    try:
        await bot.send_message(message.from_user.id, 'Have a nice day!',
reply_markup=kb_client)
        await message.delete()
    except:
        await message.reply('Нужно написать боту в ЛС, чтобы он мог ответить:
\nhttps://t.me/telegalegalegabot')

# @dp.message_handler(commands=['Schedule'])
async def information_command(message: types.Message):
    await bot.send_message(message.from_user.id, 'This is the Pixel Buildings
collection by famous artist Voxel Pizovski.')

# @dp.message_handler(commands=['Location'])
async def forecast_command(message: types.Message):
    await bot.send_message(message.from_user.id, 'The next artwork collection
is devoted to JoJo Part 6 event.')

@dp.callback_query_handler(lambda x: x.data and x.data.startswith('buy '))
async def del_callback_run(callback_query: types.CallbackQuery):
    await callback_query.answer(text="Congrats!\n
"+message_about_buying(callback_query.data.replace("buy ", "")),
show_alert=True)
async def menu_command(message: types.Message):
    # await sqlite_db.sql_read(message)
    read = await sqlite_db.sql_read2()
    for ret in read:
        # await bot.send_photo(message.from_user.id, ret[0], f'{ret[1]}\n
Description: {ret[2]}\n Price: ${ret[-1]}')
        await bot.send_photo(message.from_user.id, ret[0])
        await bot.send_message(message.from_user.id, text=f'{ret[1]}\n
Description: {ret[2]}\n Price: ${ret[-1]}',
                               reply_markup=InlineKeyboardMarkup()
                               .add(InlineKeyboardButton(f'Buy {ret[1]}!',
callback_data=f'buy {ret[1]}')))



def register_handlers_client(dp:Dispatcher):
```

```
        dp.register_message_handler(command_start, commands=['start','help'])
        dp.register_message_handler(information_command, commands=['Info'])
        dp.register_message_handler(forecast_command, commands=['Forecast'])
        dp.register_message_handler(menu_command, commands=['Exhibit'])

def message_about_buying(painting_name:string)->string:
    return f'You have bought "{painting_name}".'
```

other.py

```
from aiogram import types, Dispatcher
import json, string
from vzaimoImporty import dp
dontUnderstandYouMessage="I don't understand you. Do you speak English?"

# @dp.message_handler()
async def echo_send(message: types.Message):
    answer=just_answerer(message.text)
    if(answer!=dontUnderstandYouMessage):
        await message.answer(answer)

#    if {i.lower().translate(str.maketrans('', '', string.punctuation))  for
i in message.text.split(' ')}\
#        .intersection(set(json.load(open('cenz.json')))) != set():
#        await message.reply('Speak politely please!')
#        await message.delete()

"""    if message.text == 'Hello!':
        await message.answer('Hello there! How are you?')
    elif message.text == 'I am fine, thanks!':
        await message.answer('What a nice day to see you here!')
    elif message.text == 'test':
        await message.answer('Stop testing me!')
    elif message.text == 'Test':
        await message.answer('No tests are allowed here!')"""
 #  await message.answer(message.text)
 #  await message.reply(message.text)
 #   await bot.send_message(message.from_user.id, message.text)

def register_handlers_other(dp: Dispatcher):
    dp.register_message_handler(echo_send)

def just_answerer(message: string)->string:
    dictionary = {
        'Hello!':'Hello there! How are you?',
        'I am fine, thanks!':'What a nice day to see you here!',
        'You too!':':^)',
        'test':'Stop testing me!',
        'Test':'No tests are allowed here!'
    }
    if message in dictionary:
        return dictionary[message]
    else:
        return dontUnderstandYouMessage
"""    if message == 'Hello!':
        return 'Hello there! How are you?'
    elif message == 'I am fine, thanks!':
        return 'What a nice day to see you here!'
    elif message == 'test':
        return 'Stop testing me!'
    elif message == 'Test':
        return 'No tests are allowed here!'
"""
```

admin_kb.py

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


#admin's keyboard

button_load=KeyboardButton('/Load')
button_delete=KeyboardButton('/Delete')

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

client_kb.py

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

b1=KeyboardButton('/Info')
b2=KeyboardButton('/Forecast')
b3=KeyboardButton('/Exhibit')
# b4=KeyboardButton('Поделиться номером', request_contact=True)
# b5=KeyboardButton('Где я?', request_location=True)
kb_client=ReplyKeyboardMarkup(resize_keyboard=True)
#one_time_keyboard=True

kb_client.add(b1).insert(b2).add(b3)
#.row(b1,b2,b3)
```

Результаты работы программы.

```
1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
12 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.000s
```

```


Ran 1 test in 0.012s

OK
```