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)

Отчет по ДЗ по курсу
Базовые компоненты интернет-технологий

ПРЕПОДАВАТЕЛЬ

_____

Гапанюк Ю. Е. (подпись)

ИСПОЛНИТЕЛЬ:

студентка группы ИУ5-35Б _____

Гурова М.Д. (подпись)

"__"_____2021 г.

Москва - 2021

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).

Был использован Aiogram

Текст программы:

Main.py

```
import logging
from aiogram import Bot, types
from aiogram.types.message import  ContentType
from aiogram.utils import executor
from aiogram.dispatcher import Dispatcher
from aiogram.utils.markdown import text, bold, italic
from aiogram.types import ParseMode, user
from aiogram.types import CallbackQuery
from config import MY_ID, TOKEN
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from keyboards import keyboard
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup



import logging

bot = Bot(token=TOKEN)
dp = Dispatcher(bot, storage=MemoryStorage())
#dp.middleware.setup(LoggingMiddleware())
logging.basicConfig(format=u'%(filename)+13s [ LINE:%
(lineno)-4s] %(levelname)-8s [%(asctime)s] %(message)s',
    level=logging.INFO)

# Для машины состояний (ввода информации)
class user():
  name_user = ""
  fname_user = ""
  tel_user = ""
```

```
# Для машины состояний
class reg(StatesGroup):
    name =  State()
    fname = State()
    tel = State()

# запуск бота командой старт
@dp.message_handler(commands=['start'],state = "*")
async def process_start_command(message: types.Message):
        await message.answer(text="Добро пожаловать в МиниБот!",
reply_markup=keyboard)
        await bot.send_message(MY_ID, "start+1")

# нажатие первой кнопки
@dp.callback_query_handler(text_contains="enter", state="*")
async def process_enter_command(call:CallbackQuery,state:
FSMContext):
         await call.bot.send_message(call.from_user.id,'⭐ Вве-
дите ваши данные ⭐')
         await call.message.answer(text='💬 Введите ваше имя
💬')
         await reg.name.set()


# начало работы машины состояний
@dp.message_handler(state=reg.name, content_types=types.Content-
Types.TEXT)
async def fname_step(message: types.Message, state: FSMContext):
    if any(map(str.isdigit, message.text)):
        await message.reply("🖐 Вы ошиблись, попробуйте ввести
имя снова 🖐")
        return
    await state.update_data(name_user=message.text.title())
    user.name_user = message.text
    await message.answer(text='💬 Введите вашу фамилию 💬')
    await reg.fname.set()
```

```
@dp.message_handler(state=reg.fname,
content_types=types.ContentTypes.TEXT)
async def age_step(message: types.Message, state: FSMContext):
    if any(map(str.isdigit, message.text)):
        await message.reply("🖐 Вы ошиблись, попробуйте ввести
фамилию снова 🖐")
        return
    await message.answer(text="☎ Введите свой телефон ☎
(без +)")
    await state.update_data(fname_user=message.text.title())
    user.fname_user=message.text
    await reg.tel.set()
    await user.tel_user.set()


@dp.message_handler(state=reg.tel, content_types=types.Content-
Types.TEXT)
async def res_step(message: types.Message, state: FSMContext):
    if not any(map(str.isdigit, message.text)):
        await message.reply(text="🖐 Вы ошиблись, попробуйте
ввести свой телефон снова 🖐")
        return
    await state.update_data(tel_user=message.text.lower())
    user_data = await state.get_data()
    user.tel_user = message.text
    await state.finish()
    await message.answer(text="⭐ Вы успешно занесли свои данные
⭐")


# нажатие второй кнопки
@dp.callback_query_handler(text_contains="print")
async def process_print_command(call:CallbackQuery,state:
FSMContext):
        await call.bot.send_message(call.from_user.id,'⭐ Рас-
печатка ваших данных              /print⭐')

# запуск печати
```

```
@dp.message_handler(commands = ["print"], )
async def printer(message: types.Message, state: FSMContext):
    await bot.send_message(message.chat.id, text(
                text('Добро пожаловать,', user.name_user),
                text('Тел:', user.tel_user),
                sep='\n',
            ))

# команды бота
@dp.message_handler(commands=['help'], state="*")
async def process_help_command(message: types.Message):
    await message.reply("Список имеющихся команд: \n/start - на-
чало программы, позволяет вам увидеть функционал бота\n/print -
распечатывает данные\n/help - помощь\n")

# обработка не тех сообщений
@dp.message_handler(content_types=ContentType.ANY)
async def unknown_message(msg: types.Message):
    message_text = text(('Я не знаю, что с этим делать :('),
                        italic('\nВведите команду /help'), )
    await msg.reply(message_text, parse_mode=ParseMode.MARKDOWN)

# конец
async def shutdown(dispatcher: Dispatcher):
    await dispatcher.storage.close()
    await dispatcher.storage.wait_closed()

if __name__ == '__main__':
    executor.start_polling(dp,on_shutdown=shutdown)
```

Файл для кнопок keyboards.py

```
from aiogram.types import ReplyKeyboardRemove, \
    ReplyKeyboardMarkup, KeyboardButton, \
    InlineKeyboardMarkup, InlineKeyboardButton
```

```
keyboard = InlineKeyboardMarkup(row_width=1)

btnenter=InlineKeyboardButton(text="Ввести свои данные",
callback_data="enter")
btnprint=InlineKeyboardButton(text="Распечатать свои данные",
callback_data="print")

keyboard.insert(btnenter)
keyboard.insert(btnprint)
```

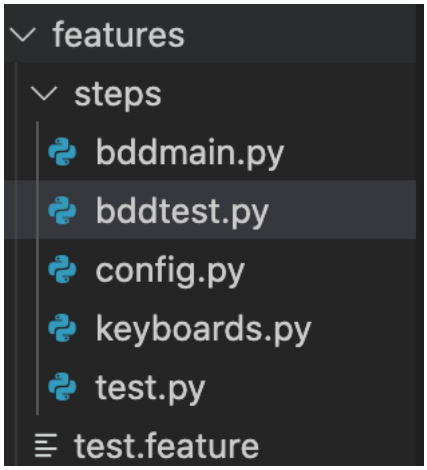
Файл config.py

```
TOKEN = '5069979728:AAFBNizLTwQWzBYeurv1jB7dxc3ecj0G5uk'
MY_ID = 'me'
```

Файл requirments.txt

```
aiogram==2.9.2
```

Для BDD есть папка, где находится bddmain.py идентичный main.py, config.py, keyboards.py также идентичны тем, которые лежат вне папки.



Файл test.py

```
# -*- coding: utf-8 -*-

import unittest
from bddmain import user, reg


class Testing(unittest.TestCase):

    def test_tel(self):

        self.assertEqual(79893003090, 79893003090)
```

```
    def test_name(self):

        self.assertEqual("мария", "мария")

    def test_surname(self):

        self.assertEqual( "гурова", "гурова")


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

Файл test.feature

```
Feature:  Test
    Scenario:  Test bot
    Given data
    When test_name
    When test_tel
    Then Ok
```

Файл bddtest.py

```
from bddmain import user, reg
from behave import *
from test import Testing

@given("data")
def test1(context):
    context.a = Testing()

@when("test_name")
def testing_name(context):
    context.a = Testing()
    context.a.test_name()

@when("test_tel")
def testing_name(context):
    context.a = Testing()
    context.a.test_tel()
```

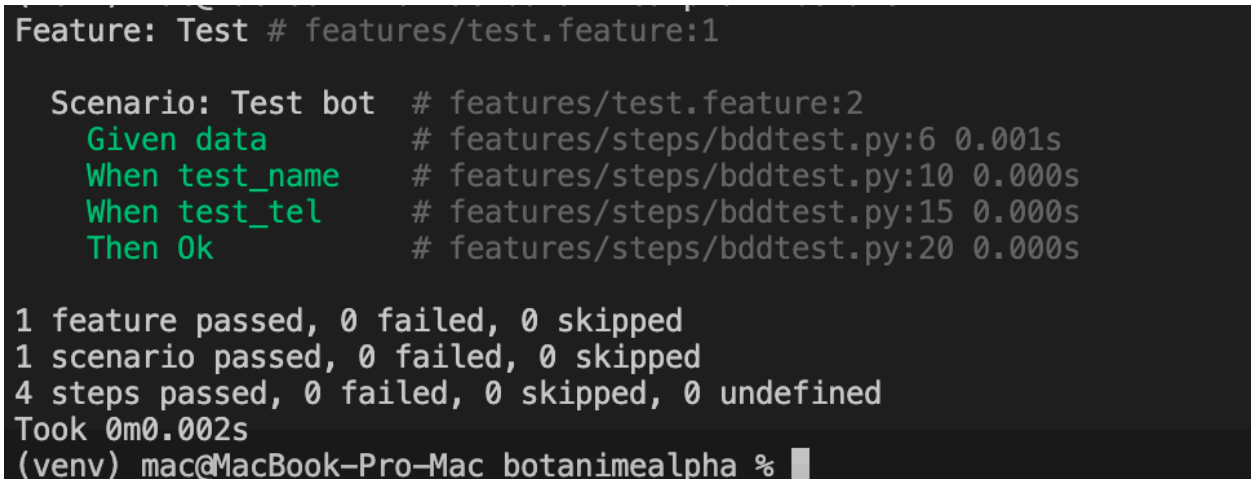
```
@then("Ok")
def result(context):
        pass
```

Для TDD использовался unittest.

Файл test.py

```
# -*- coding: utf-8 -*-

import unittest
from main import user, reg


class Testing(unittest.TestCase):
    def setUp(self):
        self.user = user()
        self.reg = reg()

    def test_tel(self):
        self.user.tel_user = 79893003090
        self.reg.tel.set()
        self.assertEqual(self.user.tel_user, 79893003090)

    def test_name(self):
        self.user.name_user = "мария"
        self.reg.name.set()
        self.assertEqual(self.user.name_user, "мария")

    def test_surname(self):
        self.user.fname_user = "гурова"
        self.reg.fname.set()
        self.assertEqual(self.user.fname_user, "гурова")


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

Пример работы - BDD тестирование с помощью behave:



Пример работы - TDD тестирование с помощью unittest