

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:

студент группы ИУ5-
35Б

Ханунов Г.И

(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:

bott.py

```
import asyncio
import logging
import unittest
import stateActualChapter
import manhwaclass
import aiogram_broadcaster
from aiogram import Bot, types
from aiogram.utils import executor
from aiogram.utils.emoji import emoji
from aiogram.dispatcher import Dispatcher
from aiogram.types.message import ContentType
from aiogram.utils.markdown import text, bold, italic, code, pre
from aiogram.types import ParseMode, InputMediaPhoto, InputMediaVideo, ChatActions
from aiogram.types import Message, CallbackQuery
from config import TOKEN, MY_ID
import keyboards
from keyboards import clava, clavaTOP, clavaChangeState, nextchapter, checkSubm, fil, UkazKB
import logging
from aiogram.utils.helper import Helper, HelperMode, ListItem
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.contrib.middlewares.logging import LoggingMiddleware
from manhwaclass import stateManhwa
import dictant
from dictant import Maindict
import os
from mysql.connector import MySQLConnection
from db_map import Base, MediaIds
from aiogram_broadcaster import TextBroadcaster
from aiogram_broadcaster import MessageBroadcaster

from aiogram.dispatcher import FSMContext
S=stateManhwa()
from pathlib import Path
S.txt =[Path('id.txt').read_text().replace("'", "")]
print(S.txt)
storage=MemoryStorage()
bot = Bot(token=TOKEN)
dp = Dispatcher(bot, storage=storage)
channel_id='@runBtsrus'
users=['133886300', '2014374178','557103049', ]
```

```

dp.middleware.setup(LoggingMiddleware())
logging.basicConfig(format=u'%(filename)+13s [ LINE:%(lineno)-4s] %(levelname)-8s [%(asctime)s] %
(message)s',
                    level=logging.INFO)
@dp.message_handler(commands=['broadcast1337'])
async def broadcast_command_handler(msg: Message, state: FSMContext):

    await msg.answer('Введите текст для начала рассылки:')
    await state.set_state('broadcast_text')
async def start_broadcast(msg: Message, state: FSMContext):
    await state.finish()
    storage = state.storage
    print(S.txt)
    users=['133886300', '2014374178','557103049', ]
    await MessageBroadcaster((users), msg).run()
dp.register_message_handler(broadcast_command_handler, commands='broadcast1337')
dp.register_message_handler(start_broadcast, state='broadcast_text',
content_types=types.ContentTypes.ANY)
@dp.message_handler(commands=['start'])
async def process_start_command(message: types.Message):
    if check_sub_channel(await bot.get_chat_member(chat_id=channel_id,
user_id=message.from_user.id)):
        await message.answer(text="start", reply_markup=clava)
    else:
        await bot.send_message(message.from_user.id, 'подписка чек', reply_markup=checkSubm)
    f=open('id.txt', 'r')
    user_id=message.from_user.id
    stroka=str(user_id)
    for line in f:
        if (stroka) in line:

            print("ид записан")
        else:
            f=open('id.txt', 'a')
            f.write(""+ stroka + ", " )

@dp.callback_query_handler(text_contains="returnMenu")
async def process_video_command(call: CallbackQuery):
    await call.message.answer(text="start step", reply_markup=clava)
    S.manhwaSolo=1
    S.manhwaDom=2
    S.manhwaEliced=3
    S.manhwaHerokiller=4
    S.manhwaNanomachines=5
    S.switch=0
    S.buffer=0
    S.buffmas=[]
    S.search=0

@dp.message_handler(commands=['ukaz'])
async def ukazGlav(message:Message):
    await bot.send_message(message.from_user.id, text="ukazat actual glav", reply_markup=UkazKB)

```

```

@dp.callback_query_handler(text_contains="last1")
async def funnn(call: CallbackQuery):
    await call.message.answer('введи номер главы с которой ты хочешь продолжить читать')
    @dp.message_handler()
    async def buffer(message: types.Message):
        S.last1=(message.text)

        await call.bot.send_message(call.from_user.id, text="актуальная глава на соло:"+S.last1)
        S.last1=int(S.last1)
@dp.callback_query_handler(text_contains="last2")
async def funnn(call: CallbackQuery):
    await call.message.answer('введи номер главы с которой ты хочешь продолжить читать')
    @dp.message_handler()
    async def buffer(message: types.Message):
        S.last2=(message.text)

        await call.bot.send_message(call.from_user.id, text="актуальная глава на соло:"+S.last2)
        S.last2=int(S.last2)
@dp.callback_query_handler(text_contains="last3")
async def funnn(call: CallbackQuery):
    await call.message.answer('введи номер главы с которой ты хочешь продолжить читать')
    @dp.message_handler()
    async def buffer(message: types.Message):
        S.last3=(message.text)

        await call.bot.send_message(call.from_user.id, text="актуальная глава на соло:"+S.last3)
        S.last3=int(S.last1)
@dp.callback_query_handler(text_contains="last4")
async def funnn(call: CallbackQuery):
    await call.message.answer('введи номер главы с которой ты хочешь продолжить читать')
    @dp.message_handler()
    async def buffer(message: types.Message):
        S.last4=(message.text)

        await call.bot.send_message(call.from_user.id, text="актуальная глава на Hero:"+S.last4)
        S.last4=int(S.last1)
@dp.callback_query_handler(text_contains="last5")
async def funnn(call: CallbackQuery):
    await call.message.answer('введи номер главы с которой ты хочешь продолжить читать')
    @dp.message_handler()
    async def buffer(message: types.Message):
        S.last5=(message.text)

        await call.bot.send_message(call.from_user.id, text="актуальная глава на соло:"+S.last5)
        S.last5=int(S.last5)

def check_sub_channel(chat_member):

```

```
if chat_member['status'] != 'left':
    return True
else:
    return False
```

```
@dp.callback_query_handler(text_contains="ca6")
```

```
async def subfunc(call: CallbackQuery):
```

```
    await call.answer(cache_time=60)
```

```
    callback_data = call.data
```

```
    logging.info(f"callback_data={callback_data}")
```

```
    if check_sub_channel(await bot.get_chat_member(chat_id=channel_id, user_id=call.from_user.id)):
```

```
        await call.message.answer(text="start", reply_markup=clava)
```

```
    else:
```

```
        await call.bot.send_message(call.from_user.id, 'Для просмотра сначала подпишись на канал',
reply_markup=checkSubm)
```

```
@dp.callback_query_handler(text_contains="тон")
```

```
async def process_video_command(call: CallbackQuery):
```

```
    await call.answer(cache_time=60)
```

```
    callback_data = call.data
```

```
    logging.info(f"callback_data={callback_data}")
```

```
    await call.message.answer('рейтинг популярных', reply_markup=clavaTOP)
```

```
@dp.message_handler(commands=['reportAproblem'])
```

```
async def process_start_command(message: types.Message):
```

```
    await message.answer(text="напиши свою проблему", reply_markup=clavaTOP)
```

```
    await bot.send_message(MY_ID, text=message.text)
```

```
@dp.callback_query_handler(text_contains="ани1")
```

```
async def process_video_command(call: CallbackQuery):
```

```
    await call.answer(cache_time=60)
```

```
    callback_data = call.data
```

```
    logging.info(f"callback_data={callback_data}")
```

```
    if check_sub_channel(await bot.get_chat_member(chat_id=channel_id, user_id=call.from_user.id)):
```

```
        await call.message.answer('выбери вариант', reply_markup=clavaChangeState)
```

```
    else:
```

```
        await call.bot.send_message(call.from_user.id, 'Для просмотра сначала подпишись на канал',
reply_markup=checkSubm)
```

```
    S.buffer=1
```

```
    S.switch=0
```

```
@dp.callback_query_handler(text_contains="ани2")
```

```
async def process_video_command(call: CallbackQuery):
```

```
    await call.answer(cache_time=60)
```

```
    callback_data = call.data
```

```
    logging.info(f"callback_data={callback_data}")
```

```
    await call.message.answer('выбери вариант', reply_markup=clavaChangeState)
```

```
    S.buffer=2
```

```
    S.switch=0
```

```

@dp.callback_query_handler(text_contains="ани3")
async def process_video_command(call: CallbackQuery):
    await call.answer(cache_time=60)
    callback_data = call.data
    logging.info(f"callback_data='{callback_data}'")
    await call.message.answer('выбери вариант', reply_markup=clavaChangeState)
    S.buffer=3
    S.switch=0

```

```

@dp.callback_query_handler(text_contains="ани4")
async def process_video_command(call: CallbackQuery):
    await call.answer(cache_time=60)
    callback_data = call.data
    logging.info(f"callback_data='{callback_data}'")
    await call.message.answer('выбери вариант', reply_markup=clavaChangeState)
    S.buffer=4
    S.switch=0

```

```

@dp.callback_query_handler(text_contains="ани5")
async def process_video_command(call: CallbackQuery):
    await call.answer(cache_time=60)
    callback_data = call.data
    logging.info(f"callback_data='{callback_data}'")
    await call.message.answer('выбери вариант', reply_markup=clavaChangeState)
    S.buffer=5
    S.switch=0

```

```

@dp.callback_query_handler(text_contains="поиск главы")
async def process_video_command(call: CallbackQuery):
    await call.answer(cache_time=60)
    callback_data = call.data
    logging.info(f"callback_data='{callback_data}'")

    await call.message.answer('введи номер главы с которой ты хочешь продолжить читать')
    @dp.message_handler()
    async def buffer(message: types.Message):
        buff=int(message.text)
        print(buff)
        print(S.buffer)
        user_id = message.from_user.id
        S.search=buff
        try:
            await bot.send_document(message.from_user.id, document=Maindict[S.buffer][S.search],
reply_markup=nextchapter)
        except:
            await bot.send_message(message.from_user.id, text='кажется этой главы еще нет :(',
reply_markup=clavaTOP)

```

```

@dp.callback_query_handler(text_contains="начать с начала")
async def process_video_command(call: CallbackQuery):
    await call.answer(cache_time=60)
    callback_data = call.data
    logging.info(f"callback_data='{callback_data}'")
    await call.message.answer('чтение с нулевой главы')
    await call.bot.send_document(call.from_user.id, document=Maindict[S.buffer][1],
reply_markup=nextchapter)

```

```

@dp.callback_query_handler(text_contains="актуальная глава")
async def process_video_command(call: CallbackQuery):
    await call.answer(cache_time=60)
    callback_data = call.data
    logging.info(f"callback_data='{callback_data}'")
    await call.message.answer('присылаю актуальную главу')
    S.lastmain=S.buffer
    await call.bot.send_document(call.from_user.id, document=Maindict[S.buffer][S.lastmain],
reply_markup=nextchapter)

```

```

@dp.callback_query_handler(text_contains="next")
async def nextSERIA(message:types.Message):

    S.search+=1
    try:
        await bot.send_document(message.from_user.id, Maindict[S.buffer][S.search],
reply_markup=nextchapter)
    except:
        await bot.send_message(message.from_user.id, text="кажется эта глава еще не добавлена :(,\n
попробуй чтонибудь другое", reply_markup=clavaTOP)

```

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

```
#fgxnnhgxmhmhg
```

по хорошему нужно будет заменить callback'i на FSM (чтобы не дублировалось одно и то же в каждом callback)

Файл для кнопок keyboards.py

```
from aiogram.types import ReplyKeyboardRemove, \  
    ReplyKeyboardMarkup, KeyboardButton, \  
    InlineKeyboardMarkup, InlineKeyboardButton  
  
clava = InlineKeyboardMarkup(row_width=2)  
btnreturnmenu=InlineKeyboardButton(text='вернуться в меню', callback_data='returnMenu')  
buy_pear1 = InlineKeyboardButton(text="топ манхв", callback_data="топ")  
clava.insert(buy_pear1)
```



```

UkazKB=InlineKeyboardMarkup(row_width=1)
hz1 = InlineKeyboardButton(text="last1", callback_data="last1")
hz2 = InlineKeyboardButton(text="last2", callback_data="last2")
hz3 = InlineKeyboardButton(text="last3", callback_data="last3")
hz4 = InlineKeyboardButton(text="last4", callback_data="last4")
hz5 = InlineKeyboardButton(text="last5", callback_data="last5")
UkazKB.insert(hz1)
UkazKB.insert(hz2)
UkazKB.insert(hz3)
UkazKB.insert(hz4)
UkazKB.insert(hz5)
#buy_pear = InlineKeyboardButton(text="кнопка отправляет не придумал",
callback_data=vibor_callback.new(ani_name="хз"))
#clava.insert(buy_pear)

clavaTOP = InlineKeyboardMarkup(row_width=1)

buy_pear2 = InlineKeyboardButton(text="поднятие уровня в одиночку", callback_data="ани1")
clavaTOP.insert(buy_pear2)

buy_pear3 = InlineKeyboardButton(text="милый дом", callback_data="ани2")
clavaTOP.insert(buy_pear3)
buy_pear4 = InlineKeyboardButton(text="элисед", callback_data="ани3")
buy_pear11 = InlineKeyboardButton(text="убийца героев", callback_data="ани4")
buy_pear12 = InlineKeyboardButton(text="ппп", callback_data="ани5")
clavaTOP.insert(buy_pear4)
clavaTOP.insert(buy_pear11)
clavaTOP.insert(buy_pear12)
clavaTOP.insert(btnreturnmenu)
but=InlineKeyboardButton(text="ссылка на другое меню, которое будет работать по подписке",
callback_data="саб")
clavaTOP.insert(but)
clavaChangeState=InlineKeyboardMarkup(row_width=1)
buy_pear5 = InlineKeyboardButton(text="читать с начала", callback_data="начать с начала")
buy_pear6 = InlineKeyboardButton(text="читать с определенной главы", callback_data="поиск
главы")
buy_pear9 = InlineKeyboardButton(text="читать актуальную главу", callback_data="актуальная
глава")

clavaChangeState.insert(buy_pear5)
clavaChangeState.insert(buy_pear6)
clavaChangeState.insert(buy_pear9)
clavaChangeState.insert(btnreturnmenu)
nextchapter=InlineKeyboardMarkup(row_width=1)
buy_pear7 = InlineKeyboardButton(text="следующая глава", callback_data="next")
buy_pear8 = InlineKeyboardButton(text="найти другую главу", callback_data="поиск главы")
nextchapter.insert(buy_pear7)

nextchapter.insert(buy_pear8)
nextchapter.insert(btnreturnmenu)
btnurlchannel= InlineKeyboardButton(text='подписаться', url='https://t.me/runBtsrus')

```

```
btndonesub=InlineKeyboardButton(text='я подписан ', callback_data='саб')
checkSubm=InlineKeyboardMarkup(row_width=1)
checkSubm.insert(btnurlchannel)
checkSubm.insert(btndonesub)
```

```
fil=InlineKeyboardMarkup(row_width=1)
but2=InlineKeyboardButton(text="1")
but3=InlineKeyboardButton(text="2")
but1=InlineKeyboardButton(text="3")
fil.insert(but1)
fil.insert(but2)
fil.insert(but3)
fil.insert(btnreturnmenu)
```

Файл config.py

```
TOKEN = 'мой токен'  
MY_ID = 'мой айдишник'
```

Для TDD использовался unittest.

```
import math  
import unittest  
import dictant  
from dictant import Solo, Herokiller  
import bott  
from bott import users  
import load  
from load import dbld, dictantHeroID  
  
class MyTestCase(unittest.TestCase):  
    def test_n1(self):  
        self.assertEqual(dbld, users)  
    def test_n2(self):  
        self.assertEqual(dictantHeroID, Herokiller)  
  
if __name__ == '__main__':  
    unittest.main()
```

Файл test.py

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

Пример работы - TDD тестирование с помощью unittest