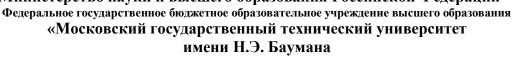
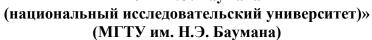
## Министерство науки и высшего образования Российской Федерации





ъ п. 1	
Факультет «Информатика и системы управления»	
дисциплина:	
«БКИТ»	
Домашнее задание	
Студент Распашнов А.А. ИУ5Ц-54Б	
(И.О. Фамилия) (Группа)	(Подпись, дата)
Преподаватель Гапанюк Ю.Е.	
(И.О. Фамилия)	(Подпись, дата)

### Оглавление

Элементы оглавления не найдены.

# Домашнее задание

#### 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).

## ДЗ – код



```
import unittest
from bot import TOKEN
from handlers.food import available burger names, available drink names,
available fries names
class TestBotWork(unittest.TestCase):
    # список еды, напитков и прочего может подгружаться из баз данных,
поэтому важно знать, что список
    # возможных продуктов не пустой
    def test bot food not empty(self):
        self.assertNotEqual(0, len(available burger names))
    def test bot drinks not empty(self):
        self.assertNotEqual(0, len(available drink names))
    def test bot fries not empty(self):
        self.assertNotEqual(0, len(available fries names))
    # важно знать, что токен бот корректен, можно проверять его длину
    def test bot token exist(self):
        self.assertEqual(len(TOKEN), 46)
if name == ' main ':
    unittest.main()
 bot.py
```

```
import asyncio
import logging
import types

from aiogram import Bot
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.types import BotCommand

from handlers.common import *
from handlers.food import *
```

```
class OrderFood(StatesGroup):
    waiting_for_burger_name = State()
    waiting_for_fries_name = State()
    waiting for drink name = State()
async def food start(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize keyboard=True)
    for name in available burger names:
        keyboard.add(name)
    await message.answer("Выберите, с чем вы хотите бургер:",
reply markup=keyboard)
    await OrderFood.waiting for burger name.set()
async def food chosen(message: types.Message, state: FSMContext):
    if message.text.lower() not in available burger names:
        await message.answer("Пожалуйста, выберите мясо в бургере, используя
клавиатуру ниже.")
       return
    await state.update data(chosen burger=message.text.lower())
    keyboard = types.ReplyKeyboardMarkup(resize keyboard=True)
    for name in available fries names:
        keyboard.add(name)
    await OrderFood.next()
    await message.answer("Теперь выберите вид картошки:",
reply markup=keyboard)
async def food fries chosen(message: types.Message, state: FSMContext):
    if message.text.lower() not in available fries names:
        await message.answer("Пожалуйста, выберите вид картошки, используя
клавиатуру ниже.")
        return
    await state.update data(chosen fries=message.text.lower())
    keyboard = types.ReplyKeyboardMarkup(resize keyboard=True)
    for name in available drink names:
        keyboard.add(name)
    await OrderFood.next()
    await message.answer('Теперь выберите напиток:', reply markup=keyboard)
async def food drink chosen (message: types.Message, state: FSMContext):
    if message.text.lower() not in available drink names:
        await message.answer("Пожалуйста, выберите напиток, используя
клавиатуру ниже.")
        return
    user data = await state.get data()
    await message.answer(f"Вы заказали бургер
{user data['chosen burger'][2:]}, \n"
                         f"картофель-{user data['chosen fries']} и напиток -
{message.text.lower()}. \n"
                         f"Спасибо за ваш заказ! ",
reply markup=types.ReplyKeyboardRemove())
    await state.finish()
def register handlers food(dp: Dispatcher):
    dp.register message handler(food start, commands="food", state="*")
    dp.register_message_handler(food_chosen,
state=OrderFood.waiting for burger name)
```

```
dp.register message handler(food fries chosen,
state=OrderFood.waiting_for_fries_name)
dp.register_message_handler(food_drink_chosen,
state=OrderFood.waiting_for_drink_name)
logger = logging.getLogger( name )
# Регистрация команд, отображаемых в интерфейсе Telegram
async def set commands(bot: Bot):
    commands = [
        BotCommand(command="/food", description="Заказать блюда"),
        BotCommand(command="/cancel", description="Отменить текущее
действие")
    ]
    await bot.set my commands(commands)
TOKEN = '5093952542:AAEfnITGgt2MSjOUOlQapYGUYWqiJdY2QJY'
async def main():
    logging.basicConfig(
        level=logging.INFO,
        format="%(asctime)s - %(levelname)s - %(name)s - %(message)s",
    logger.error("Starting bot")
    # Объявление и инициализация объектов бота и диспетчера
    bot = Bot(token=TOKEN)
    dp = Dispatcher(bot, storage=MemoryStorage())
    # Регистрация хэндлеров
    register handlers common(dp)
    register handlers food(dp)
    # Установка команд бота
    await set commands(bot)
    # Запуск поллинга
    await dp.skip updates()
    await dp.start polling()
if __name__ == '__main__':
    asyncio.run(main())
```

## Результат ДЗ:

 $C:\Users\arsec\OneDrive\Desktop\DZ\DZ\venv\Scripts\python.exe $$ "S:\Pyh\PyCharm 2021.3\plugins\python\helpers\pycharm\_jb\_unittest\_runner.py" --path C:\Users\arsec\OneDrive\Desktop\DZ\DZ\test\_bot.py$ 

Testing started at 17:07 ...

Ran 4 tests in 0.010s

OK

Process finished with exit code 0