**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 «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Домашнее задание №1

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:

студент группы ИУ5-32Б

Панов Г.Д.

Проверил:

Канев А.И.

2021 г.

# **Общее 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).

## **Текст программы**

## bot.py

import telebot  
import config  
import dbworker  
from datetime import datetime  
  
bot = telebot.TeleBot(config.token)  
  
# обработка команды /start  
@bot.message\_handler(commands=['start'])  
def catalog(message):  
 catalogKBoard = telebot.types.ReplyKeyboardMarkup(row\_width=2, resize\_keyboard=True)  
 get\_info = telebot.types.KeyboardButton(text="Добавить нового пользователя")  
 features = telebot.types.KeyboardButton(text="Текущая дата")  
 catalogKBoard.add(get\_info, features)  
 bot.send\_message(message.chat.id, "Выберите Раздел", reply\_markup=catalogKBoard)  
  
  
# Вывести текущую дату  
@bot.message\_handler(func=lambda message: message.text.lower() == 'текущая дата')  
def send\_data(message):  
 current\_datetime = datetime.now().strftime("%d-%m-%Y")  
 bot.send\_message(message.chat.id, current\_datetime)  
  
  
# Начало диалога - Добавление нового пользователя  
@bot.message\_handler(func=lambda message: message.text.lower() == 'добавить нового пользователя')  
def new\_user(message):  
 state = dbworker.get\_current\_state(message.chat.id)  
 if state == config.States.S\_ENTER\_NAME.value:  
 bot.send\_message(message.chat.id, "Кажется, кто-то обещал отправить своё имя, но так и не сделал этого :( Жду...")  
 elif state == config.States.S\_ENTER\_AGE.value:  
 bot.send\_message(message.chat.id, "Кажется, кто-то обещал отправить свой возраст, но так и не сделал этого :( Жду...")  
 else: # Под "остальным" понимаем состояние "0" - начало диалога  
 bot.send\_message(message.chat.id, "Введите имя пользователя.")  
 dbworker.set\_state(message.chat.id, config.States.S\_ENTER\_NAME.value)  
  
  
# По команде /reset будем сбрасывать состояния, возвращаясь к началу диалога  
@bot.message\_handler(commands=["reset"])  
def cmd\_reset(message):  
 bot.send\_message(message.chat.id, "Начнем сначала. Как вас зовут?")  
 dbworker.set\_state(message.chat.id, config.States.S\_ENTER\_NAME.value)  
  
  
# Обработка имени пользователя  
@bot.message\_handler(func=lambda message: dbworker.get\_current\_state(message.chat.id) == config.States.S\_ENTER\_NAME.value)  
def user\_entering\_name(message):  
 # В случае с именем не будем ничего проверять, пусть хоть "25671", хоть Евкакий  
 bot.send\_message(message.chat.id, "Принято! Теперь укажите возраст пользователя.")  
 dbworker.set\_state(message.chat.id, config.States.S\_ENTER\_AGE.value)  
  
  
# Обработка возраста пользователя  
@bot.message\_handler(func=lambda message: dbworker.get\_current\_state(message.chat.id) == config.States.S\_ENTER\_AGE.value)  
def user\_entering\_age(message):  
 # А вот тут сделаем проверку  
 if not message.text.isdigit():  
 # Состояние не меняем, поэтому только выводим сообщение об ошибке и ждём дальше  
 bot.send\_message(message.chat.id, "Что-то не так, попробуйте ещё раз!")  
 return  
 # На данном этапе мы уверены, что message.text можно преобразовать в число, поэтому ничем не рискуем  
 if int(message.text) < 5 or int(message.text) > 100:  
 bot.send\_message(message.chat.id, "Какой-то странный возраст. Не верю! Отвечайте честно.")  
 return  
 else:  
 # Возраст введён корректно, можно идти дальше  
 bot.send\_message(message.chat.id, 'Отлично! Больше ничего не требуется. Если захотите добавить еще пользователя - '  
 'нажмите на кнопку "Добавить нового пользователя".')  
 dbworker.set\_state(message.chat.id, config.States.S\_START.value)  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 bot.polling(none\_stop=True)

dbworker.py

import config  
from vedis import Vedis  
  
# Пытаемся узнать из базы «состояние» пользователя  
def get\_current\_state(user\_id):  
 with Vedis(config.db\_file) as db:  
 try:  
 return db[user\_id].decode() # Если используете Vedis версии ниже, чем 0.7.1, то .decode() НЕ НУЖЕН  
 except KeyError: # Если такого ключа почему-то не оказалось  
 return config.States.S\_START.value # значение по умолчанию - начало диалога  
  
  
  
# Сохраняем текущее «состояние» пользователя в нашу базу  
def set\_state(user\_id, value):  
 with Vedis(config.db\_file) as db:  
 try:  
 db[user\_id] = value  
 return True  
 except:  
 # тут желательно как-то обработать ситуацию  
 return False  
  
# # Добавляем имя пользователя  
# def add\_name(name):  
# with Vedis(config.db\_file\_data) as db:  
# db['Name'] = name  
#  
# # Добавляем возраст пользователя  
# def add\_age(age):  
# with Vedis(config.db\_file\_data) as db:  
# db['Age'] = age  
#  
# # df = pd.DataFrame(total, columns=['desciption', 'price'])  
# # df.index += 1  
# # df.to\_csv('stat.csv', sep='\t')  
#  
# # Извлекаем данные из базы  
# def get\_data(user\_id, value):  
# with Vedis(config.db\_file\_data) as db:

config.py

from enum import Enum  
  
  
  
token = ''  
db\_file = 'database.vdb'  
  
  
class States(Enum):  
  
 S\_START = "0" # Начало нового диалога  
 S\_ENTER\_NAME = "1" # Ввод имени  
 S\_ENTER\_AGE = "2" # Ввод возраста

tests\_tdd.py

import unittest  
from unittest.mock import patch  
from datetime import datetime  
  
class TestReset(unittest.TestCase):  
 @patch('bot.cmd\_reset', return\_value="Начнем сначала. Как вас зовут?")  
 def test\_reset(self, cmd\_reset):  
 self.assertEqual(cmd\_reset(self), "Начнем сначала. Как вас зовут?")  
  
class TestData(unittest.TestCase):  
 @patch('bot.send\_data', return\_value="20-12-2021")  
 def test\_data(self, send\_data):  
 self.assertEqual(send\_data(self), datetime.now().strftime("%d-%m-%Y"))  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

tests\_bdd.py

import pytest  
from pytest\_bdd import scenario, given, when, then  
  
@scenario("catalog.feature", "Creating start menu")  
def test\_start\_bot():  
 pass  
  
@given("The user starts the chat")  
def test\_creation\_dialog():  
 pass  
  
@when("The start command is entered")  
def test\_enter\_command\_start():  
 pass  
  
@then("The start menu is displayed")  
def test\_result\_menu():  
 pass

catalog.feature

Feature: EndMachine  
 Dialog with the bot of the user system.  
  
 Scenario: Creating start menu  
 Given The user starts the chat  
 When The start command is entered  
 Then The start menu is displayed

## **Экранные формы с примерами выполнения программы**



