**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»**

Домашняя работа

по дисциплине «Базовые компоненты интернет-технологий»

Выполнила:

студентка группы ИУ5-33Б

Румак Д.П.

Проверил:

Канев А.И.

2021 г.

**Текст программы**

**bot.py**

import telebot  
from telebot import types  
import config  
import dbworker  
  
# Создание бота  
bot = telebot.TeleBot(config.TOKEN)  
  
# Начало диалога  
@bot.message\_handler(commands=['start'])  
def start\_message(message):  
 bot.send\_message(message.chat.id, 'Привет!')  
 dbworker.set(message.chat.id, config.States.STATE\_NAME.value)  
 bot.send\_message(message.chat.id, 'Как к тебе можно обращаться?')  
  
 # 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, 'Напомни, пожалуйста...')  
 dbworker.set(message.chat.id, config.States.STATE\_NAME.value)  
 bot.send\_message(message.chat.id, 'Как к тебе можно обращаться?')  
  
# Обработка ввода имени  
@bot.message\_handler(func=lambda message: dbworker.get(message.chat.id) == config.States.STATE\_NAME.value)  
def user\_name(message):  
 text = message.text  
 if not text.isalpha():  
 # Состояние не изменяется, выводится сообщение об ошибке  
 bot.send\_message(message.chat.id, 'Кажется, ты не ввел имя!')  
 return  
 else:  
 bot.send\_message(message.chat.id, f'У тебя красивое имя! Приятно познакомиться, {text})')  
 # Меняем текущее состояние  
 dbworker.set(message.chat.id, config.States.STATE\_NAME.value)  
 # Сохраняем имя  
 dbworker.set((message.chat.id, config.States.STATE\_NAME.value), text)  
 bot.send\_message(message.chat.id, 'Скажи, пожалуйста, свой возраст!')  
 dbworker.set(message.chat.id, config.States.STATE\_AGE.value)  
  
# Обработка вызраста  
@bot.message\_handler(func=lambda message: dbworker.get(message.chat.id) == config.States.STATE\_AGE.value)  
def user\_age(message):  
 text = message.text  
 if not text.isdigit():  
 # Состояние не изменяется, выводится сообщение об ошибке  
 bot.send\_message(message.chat.id, 'Скажи правду!')  
 return  
 elif int(message.text) < 5 or int(message.text) > 90:  
 bot.send\_message(message.chat.id, "Хмм... Какой-то странный возраст. Отвечай честно!")  
 return  
 else:  
 bot.send\_message(message.chat.id, f'Супер, {text}, значит {text}')  
 # Меняем текущее состояние  
 dbworker.set(message.chat.id, config.States.STATE\_OPERATION.value)  
 # Сохраняем число  
 dbworker.set((message.chat.id, config.States.STATE\_AGE.value), text)  
 markup = types.ReplyKeyboardMarkup(row\_width=2)  
 itembtn1 = types.KeyboardButton('Посоветуй книгу')  
 itembtn2 = types.KeyboardButton('Хочу послушать музыку')  
 markup.add(itembtn1, itembtn2)  
 bot.send\_message(message.chat.id, 'Выберите, пожалуйста, чтобы ты хотел сделать', reply\_markup=markup)  
  
# Выбор действия  
@bot.message\_handler(func=lambda message: dbworker.get(message.chat.id) == config.States.STATE\_OPERATION.value)  
def operation(message):  
 # Текущее действие  
 op = message.text  
  
 if op == 'Посоветуй книгу':  
 bot.send\_message(message.chat.id,  
 "https://librebook.me/harry\_potter\_and\_the\_sorcerer\_s\_stone/vol1/1"  
 "Хорошая книга,прочти обязательно!")  
 bot.send\_message(message.chat.id,  
 "Наслаждайся! Если захочешь пообщаться снова - "  
 "отправь команду /start.")  
 elif op == 'Хочу послушать музыку':  
 bot.send\_message(message.chat.id,  
 "Здорово! Попробуй вот это"  
 "https://open.spotify.com/playlist/37i9dQZF1DWU0r6G8OGirN?si=inn5wxqtRHSoNuBML4Yagg"  
 "С тобой очень приятно иметь дело. Если захочешь пообщаться снова - "  
 "отправь команду /start.")  
 dbworker.set(message.chat.id, config.States.STATE\_START.value)  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 bot.infinity\_polling()

**config.py**

from enum import Enum  
  
# Токент бота  
TOKEN = '5025555758:AAGyQBKiYzbYYH3QkeU\_sj9P78sOzn\_Ipcs'  
  
# Файл базы данных Vedis  
db\_file = "db.vdb"  
  
# Ключ записи в БД для текущего состояния  
#CURRENT\_STATE = "CURRENT\_STATE"  
  
# Состояния автомата  
class States(Enum):  
 STATE\_START = "STATE\_START" # Начало нового диалога  
 STATE\_NAME = "STATE\_NAME"  
 STATE\_AGE = "STATE\_AGE"  
 STATE\_OPERATION = "STATE\_OPERATION"

**dbworker.py**

from vedis import Vedis  
import config  
  
# Чтение значения  
def get(user\_id):  
 with Vedis(config.db\_file) as db:  
 try:  
 return db[user\_id].decode()  
 except KeyError:  
 # в случае ошибки значение по умолчанию - начало диалога  
 return config.States.S\_START.value  
  
  
# Запись значения  
def set(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

**TDD\_test.py**

import unittest  
import sys, os  
from tdd\_test.TDD\_test import \*  
  
sys.path.append(os.getcwd())  
  
class TestBot(unittest.TestCase):  
 def test\_1(self):  
 self.assertEqual('Даша','Даша')  
  
 def test\_2(self):  
 self.assertEqual(19, 19)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

**myfeatures.steps.py**

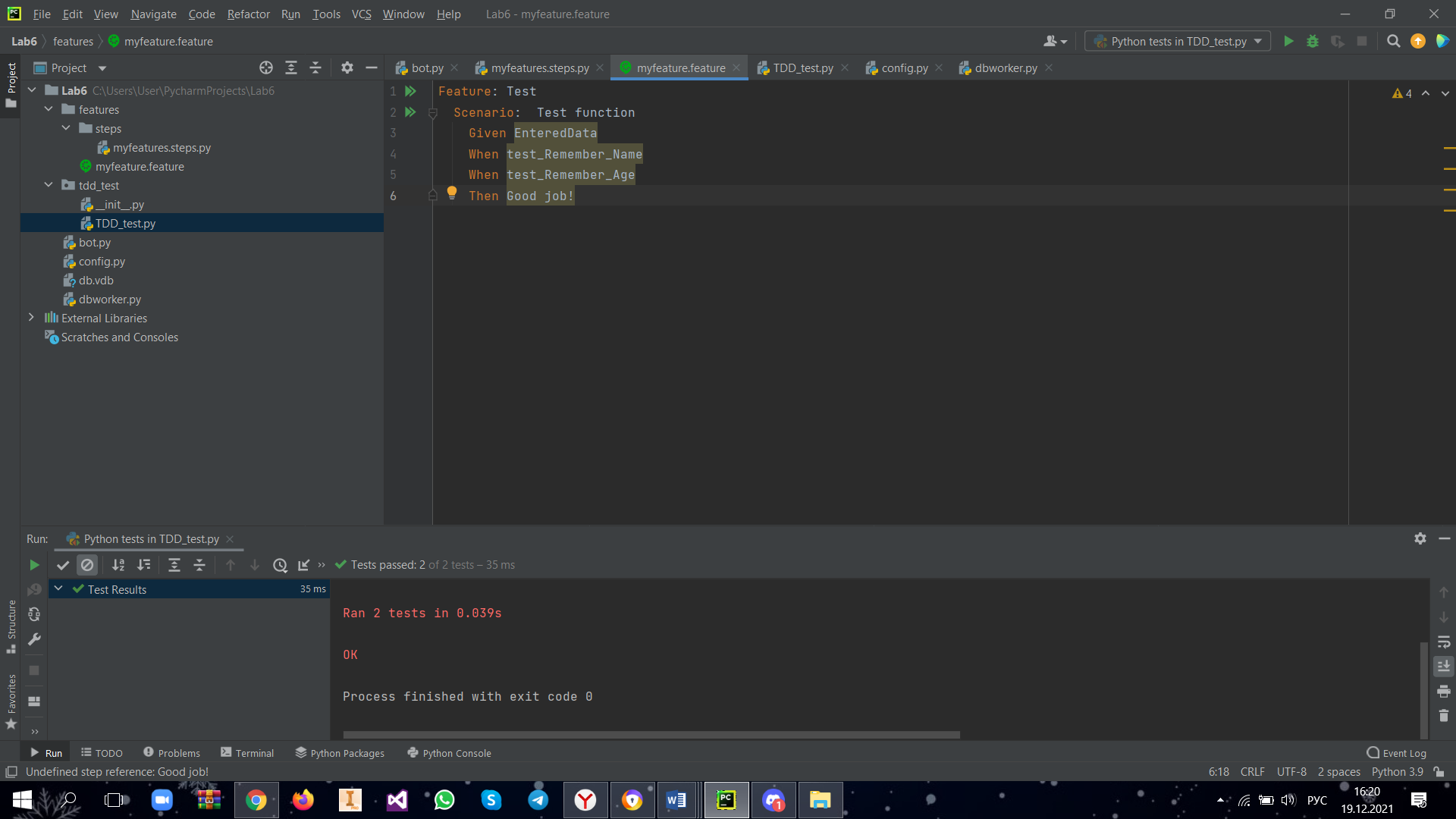
from behave import \*  
from tdd\_test.TDD\_test import \*  
  
@given("EnteredData")  
def Function(context):  
 context.a = TestBot()  
  
  
@when("test\_Remember\_Name")  
def Function\_1(context):  
 context.a.test\_1()  
  
@when("test\_Remember\_Age")  
def Function\_2(context):  
 context.a.test\_2()  
  
@then("Good job!")  
def check\_result(context):  
 pass

**myfeature.feature**

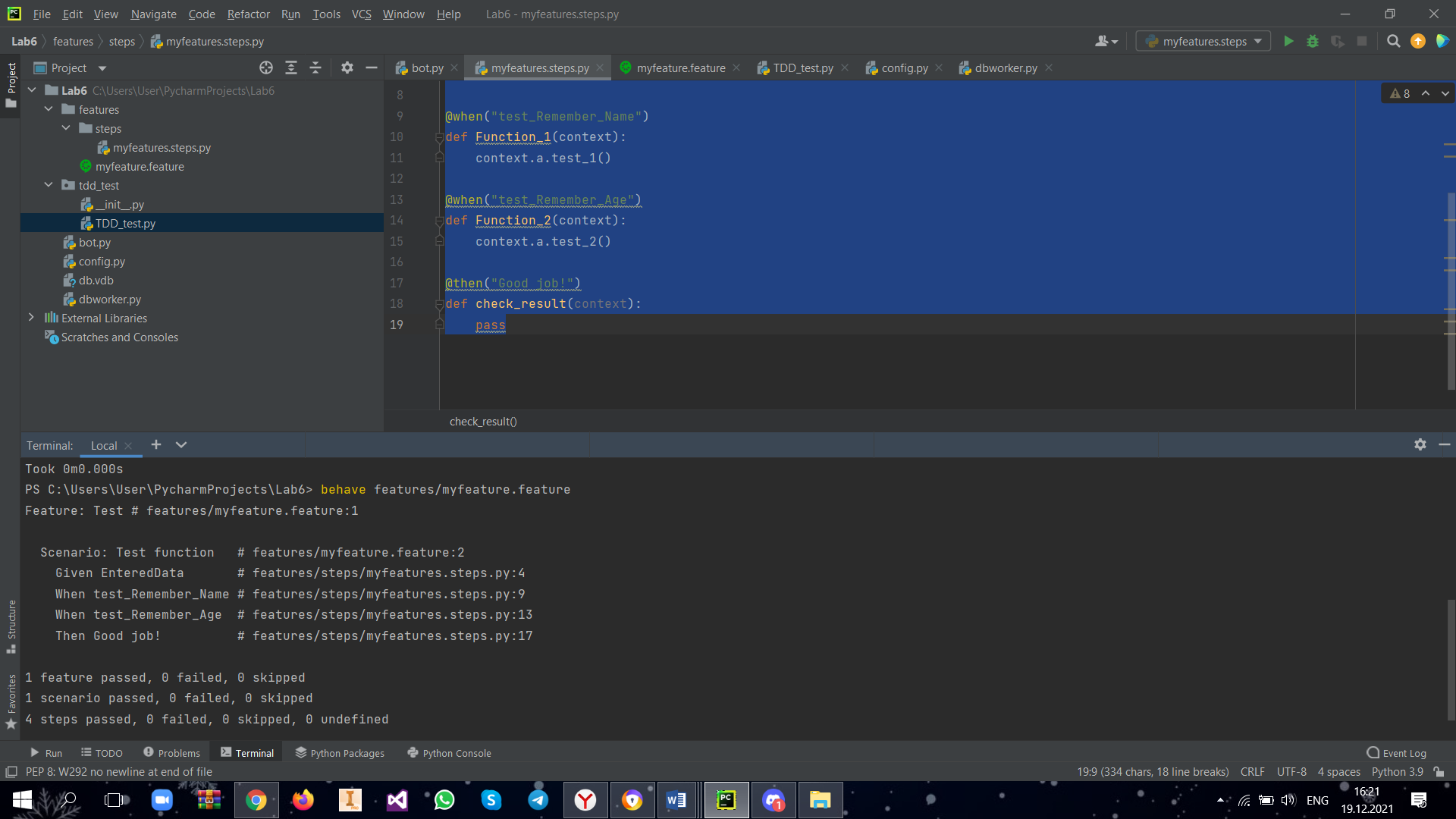
Feature: Test  
 Scenario: Test function  
 Given EnteredData  
 When test\_Remember\_Name  
 When test\_Remember\_Age  
 Then Good job!

**Результаты выполнения программы:**

**TDD\_test.py**

****

**myfeatures.steps.py**

****