**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по домашнему заданию

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-34Б |  |  |
| Свечникова Анна |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021г.

**Постановка задачи**

### 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).

**Текст программы:**

**state\_bot.py**

**import telebot**

from telebot import types

import config

import dbworker

# Создание бота

bot = telebot.TeleBot(config.TOKEN)

# переменная для накопления результата выбора

choice = ''

# надписи на кнопках

mes\_yes = "да"

mes\_no = "нет"

# вопросы бота

mes\_place = "провести встречу на открытом воздухе?"

mes\_is\_active = "предпочитаете активный отдых?"

mes\_quantity = "компания меньше пяти человек?"

#рекомендации бота

recs = {

'000':'экскурсия в музей',

'001':'квест',

'010':'лазертаг',

'011':'батутный центр',

'100':'экскурсия по городу',

'101':'пикник',

'110':'волейбол',

'111':'велопрогулка'

}

# создание кнопок

def make\_btn\_markup():

markup = types.ReplyKeyboardMarkup(row\_width=2)

itembtn1 = types.KeyboardButton(mes\_yes)

itembtn2 = types.KeyboardButton(mes\_no)

markup.add(itembtn1, itembtn2)

return markup

# Начало диалога

@bot.message\_handler(commands=['start'])

def cmd\_start(message):

bot.send\_message(message.chat.id, 'Я подскажу вам идею для встречи с друзьями')

dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE), config.States.STATE\_PLACE.value)

bot.send\_message(message.chat.id, mes\_place)

bot.send\_message(message.chat.id, 'Для ответа нажмите одну из кнопок', reply\_markup=make\_btn\_markup())

# 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, 'Сбрасываем результаты предыдущего ввода.')

dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE), config.States.STATE\_PLACE.value)

bot.send\_message(message.chat.id, mes\_place)

bot.send\_message(message.chat.id, 'Для ответа нажмите одну из кнопок')

# обработка первого вопроса

@bot.message\_handler(func=lambda message: dbworker.get(

dbworker.make\_key(message.chat.id, config.CURRENT\_STATE)) == config.States.STATE\_PLACE.value)

def get\_place(message):

text = message.text

if text == mes\_yes or text==mes\_no:

# Меняем текущее состояние

dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE), config.States.STATE\_IS\_ACTIVE.value)

# Сохраняем выбранный ответ

dbworker.set(dbworker.make\_key(message.chat.id, config.States.STATE\_PLACE.value),

'1' if text == mes\_yes else '0')

bot.send\_message(message.chat.id, mes\_is\_active)

else:

bot.send\_message(message.chat.id, 'Для ответа нажмите одну из кнопок')

# Обработка второго вопроса

@bot.message\_handler(func=lambda message: dbworker.get(

dbworker.make\_key(message.chat.id, config.CURRENT\_STATE)) == config.States.STATE\_IS\_ACTIVE.value)

def get\_is\_active(message):

text = message.text

if text == mes\_yes or text==mes\_no:

# Меняем текущее состояние

dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE), config.States.STATE\_QUANTITY.value)

# Сохраняем выбранный ответ

dbworker.set(dbworker.make\_key(message.chat.id, config.States.STATE\_IS\_ACTIVE.value),

'1' if text == mes\_yes else '0')

bot.send\_message(message.chat.id, mes\_quantity)

else:

bot.send\_message(message.chat.id, 'Пожалуйста, нажмите одну из кнопок')

def get\_resp(str):

return recs[str]

# Обработка третьего вопроса

@bot.message\_handler(func=lambda message: dbworker.get(

dbworker.make\_key(message.chat.id, config.CURRENT\_STATE)) == config.States.STATE\_QUANTITY.value)

def get\_quantity(message):

text = message.text

if text == mes\_yes or text==mes\_no:

# Сохраняем выбранный ответ

dbworker.set(dbworker.make\_key(message.chat.id, config.States.STATE\_QUANTITY.value),

'1' if text == mes\_yes else '0')

bot.send\_message(message.chat.id, "подходящий вариант досуга")

else:

bot.send\_message(message.chat.id, 'Пожалуйста, нажмите одну из кнопок')

# Читаем ответы пользователя из базы данных

k = dbworker.get(dbworker.make\_key(message.chat.id, config.States.STATE\_PLACE.value))

k += dbworker.get(dbworker.make\_key(message.chat.id, config.States.STATE\_IS\_ACTIVE.value))

k += dbworker.get(dbworker.make\_key(message.chat.id, config.States.STATE\_QUANTITY.value))

bot.send\_message(message.chat.id, get\_resp(k))

# Меняем текущее состояние

dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE), config.States.STATE\_PLACE.value)

# Выводим

bot.send\_message(message.chat.id, "Попробуем еще раз!")

bot.send\_message(message.chat.id, mes\_place)

if \_\_name\_\_ == '\_\_main\_\_':

bot.infinity\_polling()

**dbworker.py**

from vedis import Vedis

import config

import state\_bot

# Чтение значения

def get(key):

with Vedis(config.db\_file) as db:

try:

return db[key].decode()

except KeyError:

# в случае ошибки значение по умолчанию - начало диалога

return config.States.STATE\_START.value

# Запись значения

def set(key, value):

with Vedis(config.db\_file) as db:

try:

db[key] = value

return True

except:

bot.send\_message('Произошла критическая ошибка. Пожалуйста, сообщите о ней разработчику')

return False

# Создание ключа для записи и чтения

def make\_key(chatid, keyid):

res = str(chatid) + '\_\_' + str(keyid)

return res

**config.py**

from enum import Enum

#токен бота

TOKEN = '5037268913:AAHDueSGRATmwFeWYNISf9KikpCKJDV1HwY'

#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\_PLACE = "STATE\_PLACE" # желаемое место

STATE\_IS\_ACTIVE = "STATE\_IS\_ACTIVE" # активный отдых или нет

STATE\_QUANTITY = "STATE\_QUANTITY" # размер компании

**steps.py**

from behave import given, when, then

from state\_bot import get\_resp

@given ("user answers {a}, {b}, {c}")

def parse\_answrs(context, a, b, c):

context.a = a

context.b = b

context.c = c

@when ("bot recieves the messages")

def get\_msgs(context):

context.key = ""

if context.a == "да":

context.key += "1"

else:

context.key += "0"

if context.b == "да":

context.key += "1"

else:

context.key += "0"

if context.c == "да":

context.key += "1"

else:

context.key += "0"

@then("I expect the result to bе {result}")

def expect\_result(context, result):

context.result = get\_resp(context.key)

context.result == result

**bottest.feature**

Feature: generating ideas for parties

Scenario: cycling

Given user answers "да", "да", "да"

When bot recieves the messages

Then I expect the result to bе "велопрогулка"

Scenario: picknick

Given user answers "да", "нет", "да"

When bot recieves the messages

Then I expect the result to bе "пикник"

**tdd.py**

from state\_bot import get\_resp

import unittest

class TddTest(unittest.TestCase):

def test1(self):

self.assertEqual(get\_resp("000"), "экскурсия в музей")

def test2(self):

self.assertEqual(get\_resp("011"), "батутный центр")

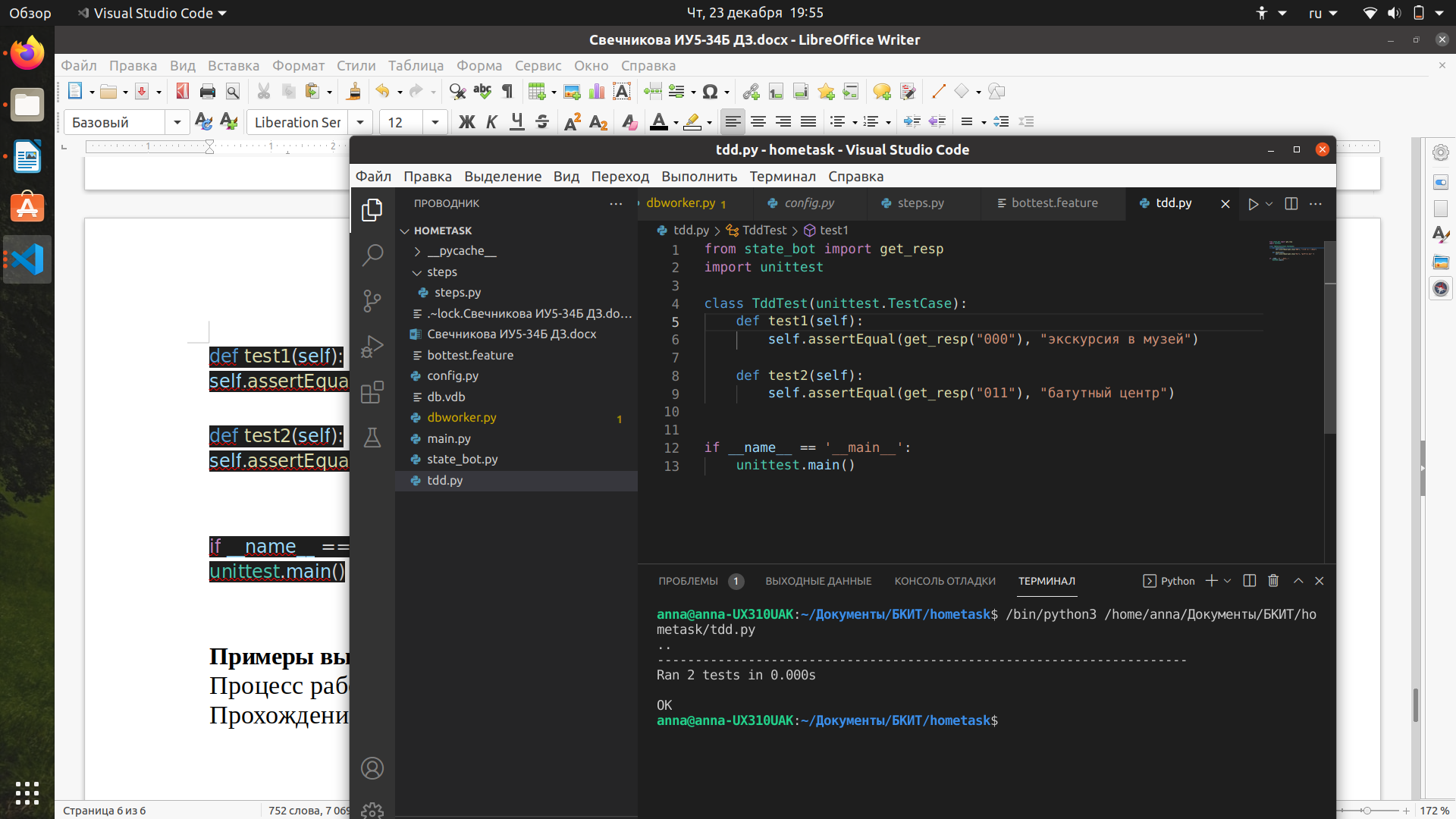
if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

**Примеры выполнения**

Процесс работы бота такой же, как у бота в 6 лабораторной работе.

Прохождение TDD-тестов:



Прохождение BDD-тестов:

