

**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»

Домашнее задание

Выполнил:
студент группы ИУ5-34Б
Гордеев Никита

Подпись и дата:

Проверил:
Гапанюк Ю. Е.

Подпись и дата:

Москва, 2021 г.

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

Файлы: bot.py

```
stepsBDD.py  bot.py  X
```

```
import telebot
from telebot import types
from config import get_sen
import config
import dbworker
import os

# Создание бота
bot = telebot.TeleBot(config.TOKEN)

@bot.message_handler(commands=['start'])
def cmd_start(message):
    dbworker.set(dbworker.make_key(message.chat.id, config.SENTENCE), '')
    bot.send_message(message.chat.id, 'Добро пожаловать в игру: "Составь предложение".')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE), config.States.STATE_FIRST_WORD.value)
    bot.send_message(message.chat.id, 'Вам нужно каждый раз выбирать одно слово из трех предложенных.')
    markup = types.ReplyKeyboardMarkup(row_width=2, resize_keyboard=True)
    ibtn1 = types.KeyboardButton('Леонид')
    ibtn2 = types.KeyboardButton('Петр')
    ibtn3 = types.KeyboardButton('Святослав')
    markup.add(ibtn1, ibtn2, ibtn3)
    bot.send_message(message.chat.id, 'Выберете одно из трех слов:', reply_markup=markup)

@bot.message_handler(commands=['reset'])
def cmd_reset(message):
    dbworker.set(dbworker.make_key(message.chat.id, config.SENTENCE), '')
    bot.send_message(message.chat.id, 'Начнем игру сначала!')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE), config.States.STATE_FIRST_WORD.value)
    bot.send_message(message.chat.id, 'Вам нужно каждый раз выбирать одно слово из трех предложенных.')
    markup = types.ReplyKeyboardMarkup(row_width=2, resize_keyboard=True)
    ibtn1 = types.KeyboardButton('Леонид')
    ibtn2 = types.KeyboardButton('Петр')
    ibtn3 = types.KeyboardButton('Святослав')
    markup.add(ibtn1, ibtn2, ibtn3)
    bot.send_message(message.chat.id, 'Выберете одно из трех слов:', reply_markup=markup)

@bot.message_handler(func=lambda message: dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE)) == config.States.STATE_FIRST_WORD.value)
def first_word(message):
    text = message.text
    text_old = dbworker.get(dbworker.make_key(message.chat.id, config.SENTENCE))
    text = text_old + ' ' + text
    dbworker.set(dbworker.make_key(message.chat.id, config.SENTENCE), text)
    bot.send_message(message.chat.id, f'{text}')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE), config.States.STATE_SECOND_WORD.value)
    markup = types.ReplyKeyboardMarkup(row_width=2, resize_keyboard=True)
    ibtn1 = types.KeyboardButton('шел')
    ibtn2 = types.KeyboardButton('прыгал')
    ibtn3 = types.KeyboardButton('бежал')
    markup.add(ibtn1, ibtn2, ibtn3)
```

```

    markup.add(ibtn1, ibtn2, ibtn3)
    bot.send_message(message.chat.id, 'Выберете одно из трех слов:', reply_markup=markup)

# Обработка второго числа
@bot.message_handler(func=lambda message: dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE)) == config.States.STATE_SECOND_WORD.value)
def second_word(message):
    text = message.text
    text_old = dbworker.get(dbworker.make_key(message.chat.id, config.SENTENCE))
    text = text_old + ' ' + text
    dbworker.set(dbworker.make_key(message.chat.id, config.SENTENCE), text)
    bot.send_message(message.chat.id, f'{text}')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE), config.States.STATE_THIRD_WORD.value)
    markup = types.ReplyKeyboardMarkup(row_width=2, resize_keyboard=True)
    ibtn1 = types.KeyboardButton('по тротуару')
    ibtn2 = types.KeyboardButton('по крышам')
    ibtn3 = types.KeyboardButton('по берегу')
    markup.add(ibtn1, ibtn2, ibtn3)
    bot.send_message(message.chat.id, 'Выберете одно из трех слов:', reply_markup=markup)

@bot.message_handler(func=lambda message: dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE)) == config.States.STATE_THIRD_WORD.value)
def third_word(message):
    text = message.text
    text_old = dbworker.get(dbworker.make_key(message.chat.id, config.SENTENCE))
    text = text_old + ' ' + text
    dbworker.set(dbworker.make_key(message.chat.id, config.SENTENCE), text)
    bot.send_message(message.chat.id, f'{text}')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE), config.States.STATE_FOURTH_WORD.value)
    markup = types.ReplyKeyboardMarkup(row_width=2, resize_keyboard=True)
    ibtn1 = types.KeyboardButton('и сосал')
    ibtn2 = types.KeyboardButton('и грыз')
    ibtn3 = types.KeyboardButton('и раздавал')
    markup.add(ibtn1, ibtn2, ibtn3)
    bot.send_message(message.chat.id, 'Выберете одно из трех слов:', reply_markup=markup)

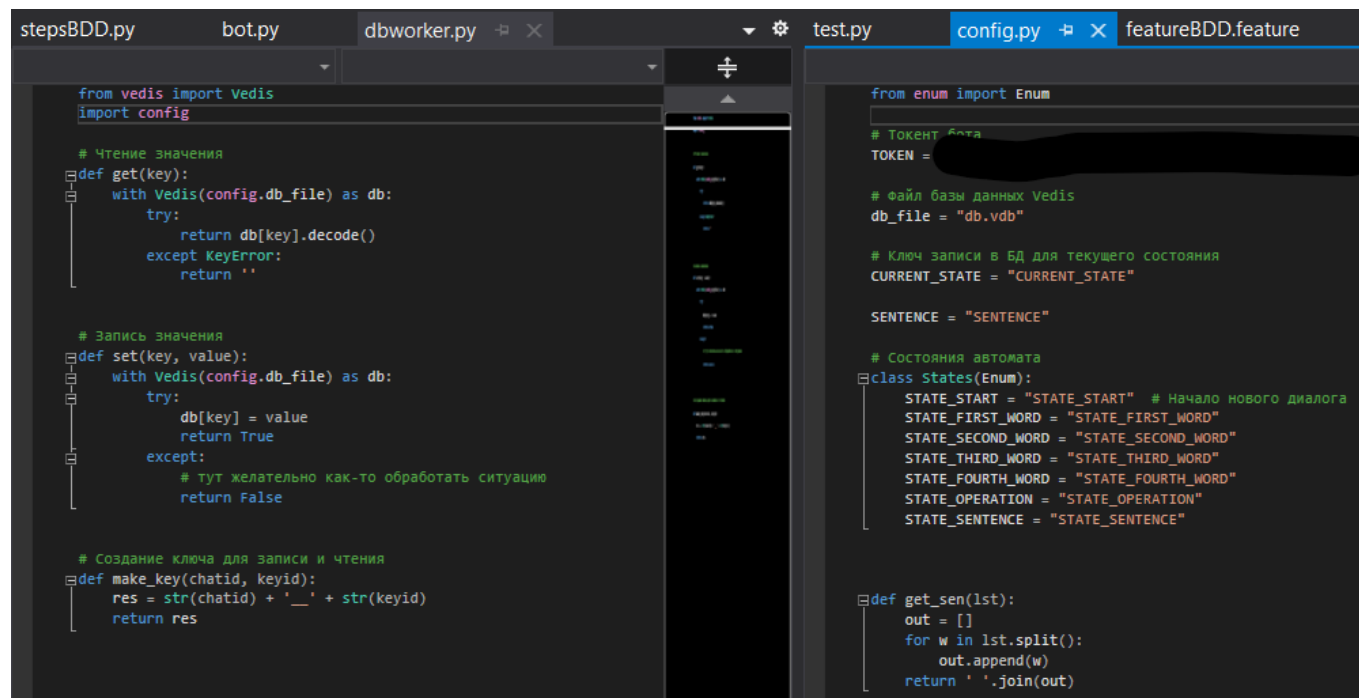
@bot.message_handler(func=lambda message: dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE)) == config.States.STATE_FOURTH_WORD.value)
def fourth_word(message):
    text = message.text
    text_old = dbworker.get(dbworker.make_key(message.chat.id, config.SENTENCE))
    text = text_old + ' ' + text
    dbworker.set(dbworker.make_key(message.chat.id, config.SENTENCE), text)
    bot.send_message(message.chat.id, f'{text}')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE), config.States.STATE_SENTENCE.value)
    markup = types.ReplyKeyboardMarkup(row_width=2, resize_keyboard=True)
    ibtn1 = types.KeyboardButton('чула-чупсы.')
    ibtn2 = types.KeyboardButton('конфеты.')
    ibtn3 = types.KeyboardButton('леденцы.')
    markup.add(ibtn1, ibtn2, ibtn3)
    bot.send_message(message.chat.id, 'Выберете одно из трех слов:', reply_markup=markup)

@bot.message_handler(func=lambda message: dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE)) == config.States.STATE_SENTENCE.value)
def again(message):
    text = message.text
    text_old = dbworker.get(dbworker.make_key(message.chat.id, config.SENTENCE))
    text = text_old + ' ' + text
    text = config.get_sen(text)
    dbworker.set(dbworker.make_key(message.chat.id, config.SENTENCE), text)
    bot.send_message(message.chat.id, f'Получившееся предложение:\n"{text}"')
    bot.send_message(message.chat.id, 'Играем еще!')
    dbworker.set(dbworker.make_key(message.chat.id, config.SENTENCE), '')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE), config.States.STATE_FIRST_WORD.value)
    bot.send_message(message.chat.id, 'Вам нужно каждый раз выбирать одно слово из трех предложенных.')
    markup = types.ReplyKeyboardMarkup(row_width=2, resize_keyboard=True)
    ibtn1 = types.KeyboardButton('Леонид')
    ibtn2 = types.KeyboardButton('Петр')
    ibtn3 = types.KeyboardButton('Святослав')
    markup.add(ibtn1, ibtn2, ibtn3)
    bot.send_message(message.chat.id, 'Выберете одно из трех слов:', reply_markup=markup)

if __name__ == '__main__':
    if os.path.exists('db.vdb'): os.remove('db.vdb')
    bot.infinity_polling()

```

Файлы: dbworker.py и config.py



The screenshot shows an IDE with two files open: dbworker.py and config.py. dbworker.py contains functions for interacting with a database using the Vedis library. config.py contains configuration variables and an Enum class for states.

```
from vedis import Vedis
import config

# Чтение значения
def get(key):
    with Vedis(config.db_file) as db:
        try:
            return db[key].decode()
        except KeyError:
            return ''

# Запись значения
def set(key, value):
    with Vedis(config.db_file) as db:
        try:
            db[key] = value
            return True
        except:
            # тут желательно как-то обработать ситуацию
            return False

# Создание ключа для записи и чтения
def make_key(chatid, keyid):
    res = str(chatid) + '_' + str(keyid)
    return res
```

```
from enum import Enum

# Токент бота
TOKEN = '...'

# Файл базы данных Vedis
db_file = "db.vdb"

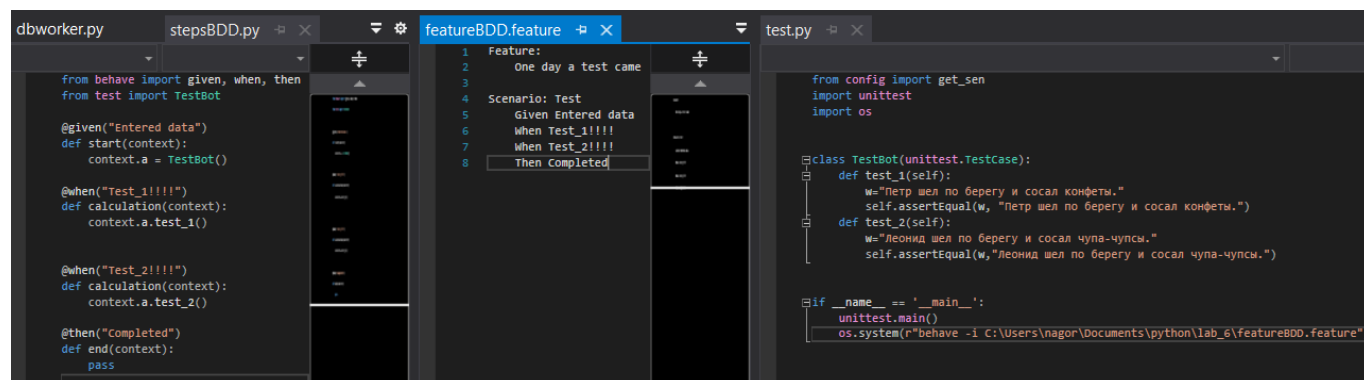
# Ключ записи в БД для текущего состояния
CURRENT_STATE = "CURRENT_STATE"

SENTENCE = "SENTENCE"

# Состояния автомата
class States(Enum):
    STATE_START = "STATE_START" # Начало нового диалога
    STATE_FIRST_WORD = "STATE_FIRST_WORD"
    STATE_SECOND_WORD = "STATE_SECOND_WORD"
    STATE_THIRD_WORD = "STATE_THIRD_WORD"
    STATE_FOURTH_WORD = "STATE_FOURTH_WORD"
    STATE_OPERATION = "STATE_OPERATION"
    STATE_SENTENCE = "STATE_SENTENCE"

def get_sen(lst):
    out = []
    for w in lst.split():
        out.append(w)
    return ' '.join(out)
```

Файлы: StepsBDD.py, featureBDD.feature и test.py



The screenshot shows an IDE with three files open: StepsBDD.py, featureBDD.feature, and test.py. StepsBDD.py contains BDD steps for a test scenario. featureBDD.feature contains a Gherkin scenario. test.py contains a unittest class for testing the bot.

```
from behave import given, when, then
from test import TestBot

@given("Entered data")
def start(context):
    context.a = TestBot()

@when("Test_1!!!!")
def calculation(context):
    context.a.test_1()

@when("Test_2!!!!")
def calculation(context):
    context.a.test_2()

@then("Completed")
def end(context):
    pass
```

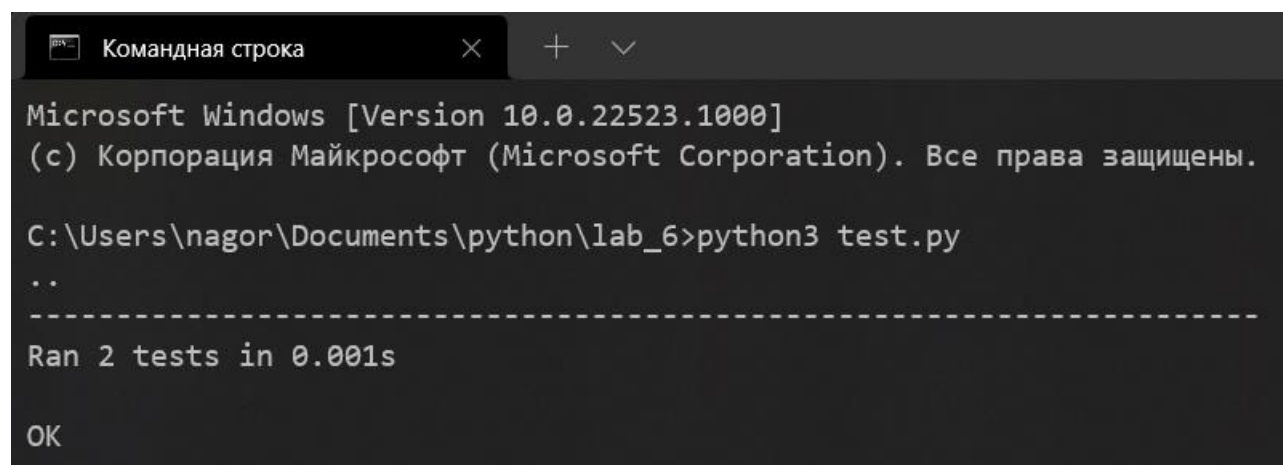
```
1 Feature:
2   One day a test came
3
4 Scenario: Test
5   Given Entered data
6   When Test_1!!!!
7   When Test_2!!!!
8   Then Completed
```

```
from config import get_sen
import unittest
import os

class TestBot(unittest.TestCase):
    def test_1(self):
        w="Петр шел по берегу и сосал конфеты."
        self.assertEqual(w, "Петр шел по берегу и сосал конфеты.")
    def test_2(self):
        w="Леонид шел по берегу и сосал чупа-чупсы."
        self.assertEqual(w, "Леонид шел по берегу и сосал чупа-чупсы.")

if __name__ == '__main__':
    unittest.main()
    os.system(r"behave -i C:\Users\nagor\Documents\python\lab_6\featureBDD.feature")
```

Результат выполнения тестов



The screenshot shows a Windows command prompt window titled "Командная строка". It displays the output of running the test script.

```
Microsoft Windows [Version 10.0.22523.1000]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\nagor\Documents\python\lab_6>python3 test.py
..
-----
Ran 2 tests in 0.001s

OK
```

```
C:\Users\nagor\Documents\python\lab_6>behave -i featureBDD.feature
Feature: # featureBDD.feature:1
  One day a test came
    Scenario: Test # featureBDD.feature:4
      Given Entered data # steps/stepsBDD.py:4
      When Test_1!!!! # steps/stepsBDD.py:8
      When Test_2!!!! # steps/stepsBDD.py:13
      Then Completed # steps/stepsBDD.py:17

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
4 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.001s
```