**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»**

Лабораторная работа №6

по дисциплине «Базовые компоненты интернет-технологий»

Выполнила:

студентка группы ИУ5-33Б

Шаповалова В.В.

Проверил:

Канев А.И.

2021 г.

Задание:

1. Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

Текст программы

bot.py

import telebot  
import config  
import dbworker  
  
bot = telebot.TeleBot(config.token)  
available\_sign = ["козерог", "водолей", "рыбы", "овен", "телец", "близнецы", "рак", "лев", "дева", "весы", "скорпион", "стрелец"]  
winx =['лейла', 'стела', 'муза', 'флора', 'техна', 'блум', 'не люблю винкс']  
token = '5048990230:AAH2MovxsbqssKT6kMMdbNf01R6S\_B0idRc'  
bot = telebot.TeleBot(token)  
  
  
  
@bot.message\_handler(commands=["cancel"])  
def cmd\_reset(message):  
 bot.send\_message(message.chat.id, "не хочешь развлекух - как хочешь, переходи в /menu")  
 dbworker.set\_state(message.chat.id, config.States. S\_START.value)  
  
  
@bot.message\_handler(commands=["reset"])  
def cmd\_reset(message):  
 bot.send\_message(message.chat.id, "ну что это такое.. минус 10 очков за такое надо...\n представляйся ещё раз")  
 dbworker.set\_state(message.chat.id, config.States.STATE\_FIRST\_NUM.value)  
  
# Начало диалога  
@bot.message\_handler(commands=['love'])  
def cmd\_start(message):  
 bot.send\_message(message.chat.id, "смело! но если захочешь выйти из этого режима - /cancel поможет\nначать заполнять заново - /reset\nкак к тебе обращаться?")  
 dbworker.set\_state(message.chat.id, config.States.STATE\_FIRST\_NUM.value)  
  
# По команде /reset будем сбрасывать состояния, возвращаясь к началу диалога  
  
  
@bot.message\_handler(func=lambda message: dbworker.get\_current\_state(message.chat.id) == config.States.STATE\_FIRST\_NUM.value)  
def user\_entering\_name(message):  
 if not message.text.isalpha():  
 bot.send\_message(message.chat.id, "не дури меня! не очень на имя похоже! давай из буковок")  
 return  
 else:  
 bot.send\_message(message.chat.id, "хорошо, теперь твой знак зодиака...")  
 dbworker.set\_state(message.chat.id, config.States.STATE\_SECOND\_NUM.value)  
  
  
@bot.message\_handler(func=lambda message: dbworker.get\_current\_state(message.chat.id) == config.States.STATE\_SECOND\_NUM.value)  
def user\_entering\_sign(message):  
 # А вот тут сделаем проверку  
 if message.text.lower() not in available\_sign:  
 bot.send\_message(message.chat.id, "ну давай, честно, выбери зз! из всего 12, ты справишься!")  
 return  
 else:  
 if message.text.lower() == 'овен' or message.text.lower() =='лев' or message.text.lower() =='рыбы':  
 bot.send\_message(message.chat.id, "нууу, шансов у нас мало, но глянем, что дальше!")  
 if message.text.lower() == 'рак' or message.text.lower() == 'водолей' or message.text.lower() == 'весы':  
 bot.send\_message(message.chat.id, "уже кайфую")  
 if message.text.lower() == 'козерог':  
 bot.send\_message(message.chat.id, "это мой!!! ладно, видимо твой тоже")  
  
 keyboard = telebot.types.ReplyKeyboardMarkup(True)  
 for fairy in winx:  
 keyboard.add(fairy)  
 bot.send\_message(message.chat.id, "последний шаг! выбери люимую фею винкс", reply\_markup=keyboard)  
 dbworker.set\_state(message.chat.id, config.States.STATE\_THIRD\_NUM.value)  
  
@bot.message\_handler(func=lambda message: dbworker.get\_current\_state(message.chat.id) == config.States.STATE\_THIRD\_NUM.value)  
def user\_fairy(message):  
 if message.text.lower() not in winx:  
 bot.send\_message(message.chat.id, "знаю, сложно,но феечку надо выбрать")  
 return  
 else:  
 bot.send\_message(message.chat.id, "мммм, интересненько")  
 dbworker.set\_state(message.chat.id, config.States.S\_START.value)  
 bot.send\_message(message.chat.id, "это конец тестика, подведём результаты! хммммммммм")  
 if message.text.lower() == 'не люблю винкс':  
 bot.send\_message(message.chat.id, "сори, но если ты не любишь винкс - то это сразу 0%")  
 else:  
 bot.send\_message(message.chat.id, "ты самый крутой перец, независимо от ответов! переходи в /menu")  
  
  
@bot.message\_handler(commands=['start'])  
def stt\_message(message):  
 keyboard = telebot.types.ReplyKeyboardMarkup(True)  
 buttons = ["/menu"]  
 keyboard.add(\*buttons)  
 bot.send\_message(message.chat.id, 'Привет, молодёжь! Ну или как у вас принято говорить - привет-медвед, молодёжь!', reply\_markup=keyboard)  
  
@bot.message\_handler(commands=['menu'])  
def start\_message(message):  
 keyboard = telebot.types.ReplyKeyboardMarkup(True)  
 buttons = ["анекдот", "пока", "/test", "/love"]  
 keyboard.add(\*buttons)  
 bot.send\_message(message.chat.id, ' Хочешь попрощаться - нажми "пока"\nХочешь интерактива - нажми "/test"\nХочешь пройти тест на крутость - нажми "/love"\nНу и анекдот, думаю, понятно) ', reply\_markup=keyboard)  
  
  
@bot.message\_handler(commands=['test'])  
def start\_message(message):  
 markup = telebot.types.InlineKeyboardMarkup()  
 markup.add(telebot.types.InlineKeyboardButton(text='вероника-голубика', callback\_data=1))  
 markup.add(telebot.types.InlineKeyboardButton(text='вероника-ежевика', callback\_data=2))  
 markup.add(telebot.types.InlineKeyboardButton(text='вероника-облепиха', callback\_data=3))  
 bot.send\_message(message.chat.id, text="ииинтерактивчик! какую Веронику хочешь увидеть?", reply\_markup=markup)  
  
@bot.message\_handler(content\_types=['text'])  
def send\_text(message):  
 if message.text.lower() == 'анекдот':  
 bot.send\_message(message.chat.id, 'Анекдот №186:\nКолобок повесился')  
 elif message.text.lower() == 'пока':  
 bot.send\_message(message.chat.id, 'Досвидули, досвидули, досвидули-дули-дули!')  
 elif message.text.lower() == 'песню!':  
 bot.send\_message(message.chat.id, 'ху ху, ну держи, повайбь')  
 bot.send\_message(message.chat.id, 'https://open.spotify.com/track/7EkWXAI1wn8Ii883ecd9xr?si=51LCrMbFTSyESuTgHBoCGg&utm\_source=copy-link')  
 else:  
 bot.send\_message(message.chat.id, 'не понимаю, о чём ты')  
  
@bot.callback\_query\_handler(func=lambda call: True)  
def query\_handler(call):  
 if call.data == '1':  
 bot.send\_photo(call.message.chat.id, 'https://drive.google.com/file/d/1mxv16SxLnSrX22nboqguxc06M0kBFv\_G/view?usp=drivesdk')  
 elif call.data == '2':  
 bot.send\_photo(call.message.chat.id, 'https://drive.google.com/file/d/1CzNb6C\_7WHeeXkol0QQ5hZzs03c9-Srd/view?usp=drivesdk')  
 elif call.data == '3':  
 bot.send\_photo(call.message.chat.id, 'https://drive.google.com/file/d/1ZQY9LXfqjOpDgGD8pqpoJDOhprPXe2FX/view?usp=drivesdk')  
  
 bot.edit\_message\_reply\_markup(call.message.chat.id, call.message.message\_id)  
  
bot.polling()

dbworker.py

from vedis import Vedis  
import config  
  
# 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()  
 except KeyError:  
 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:  
 return False

config.py

from enum import Enum  
  
token = "1234567:ABCxyz"  
db\_file = "database.vdb"  
  
CURRENT\_STATE = "CURRENT\_STATE"  
  
class States(Enum):  
 *"""  
 Мы используем БД Vedis, в которой хранимые значения всегда строки,  
 поэтому и тут будем использовать тоже строки (str)  
 """* S\_START = "0" # Начало нового диалога  
 STATE\_FIRST\_NUM = "1"  
 STATE\_SECOND\_NUM = "2"  
 STATE\_THIRD\_NUM = "3"

bdd

from behave import \*  
import telebot  
token = '5048990230:AAH2MovxsbqssKT6kMMdbNf01R6S\_B0idRc'  
bot = telebot.TeleBot(token)  
  
  
@given("Entered data")  
def DataEnt(context):  
 context.a = bot()  
  
@when("Fix name")  
def Fix(context):  
 context.a.test\_1()  
  
@when("Fix winx")  
def Fix2(context):  
 context.a.test\_2()  
  
@then("Completed")  
def check\_result(context):  
 pass

m.feature

Feature: Test  
 Scenario: Test function  
 Given Entered data  
 When Fix name  
 When Fix winx  
 Then Completed

test\_tdd

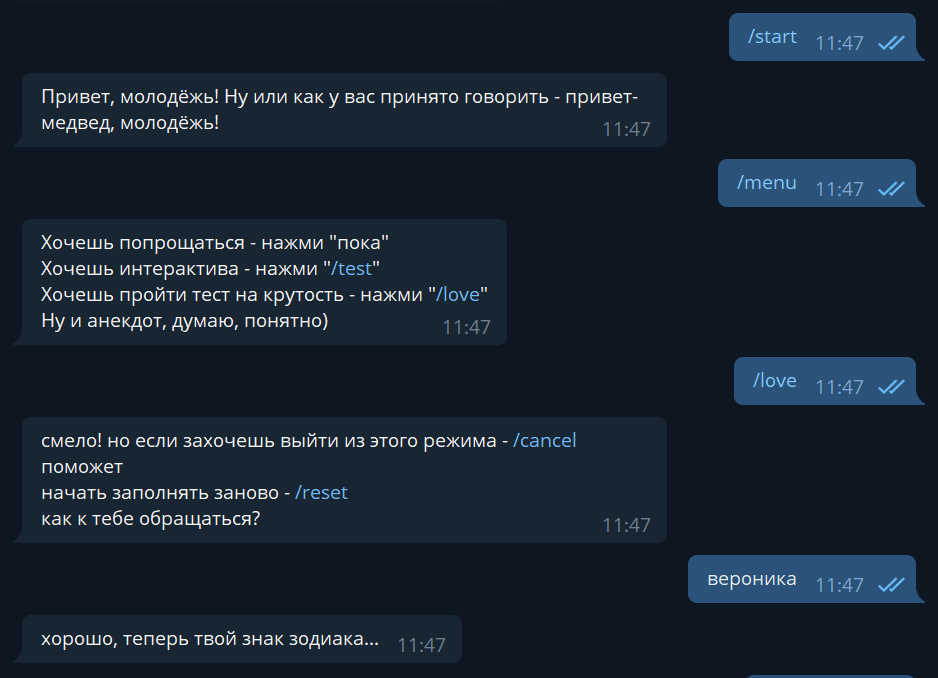
import unittest  
import sys, os  
sys.path.append(os.getcwd())  
text = 'не хочешь развлекух - как хочешь, переходи в /menu'  
winx = ['лейла', 'стела', 'муза', 'флора', 'техна', 'блум', 'не люблю винкс']  
  
class TestBot(unittest.TestCase):  
 def test\_1(self):  
 reset = text  
 self.assertEqual(reset, 'не хочешь развлекух - как хочешь, переходи в /menu')  
  
 def test\_2(self):  
 cmd = winx  
 self.assertEqual(cmd, ['лейла', 'стела', 'муза', 'флора', 'техна', 'блум', 'не люблю винкс'])  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

Экранные формы

Изображение выглядит как текст

Автоматически созданное описание

Скрины бота



Изображение выглядит как текст

Автоматически созданное описание