Министерство науки и высшего образования Российской Федерации

Федерального государственного бюджетного образовательного учреждения высшего образования

"Сочинский государственный университет"

|  |  |
| --- | --- |
| https://yt3.ggpht.com/a/AGF-l79mnrPjIZT20GpORY7zb_8pkVgzIh6Zu9PsFQ=s900-c-k-c0xffffffff-no-rj-mo | УТВЕРЖДАЮ  Проректор  по учебной работе  и информатизации  \_\_\_\_\_\_\_\_\_\_\_\_ А.В.Ревнивых  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

Регистрационный номер \_\_\_

Факультет «Инженерно-экологический»

Кафедра ИТ «Информационные технологии»

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

**по дисциплине**

**«Операционные системы»**

**Лабораторный практикум**

для направлений (уровень бакалавриата):

09.03.03 «Прикладная информатика», профиль «Прикладная информатика в экономике»

**Автор:**

Драч В.Е.,

доцент к.т.н., доцент кафедры ИТ

[vladimir@drach.](mailto:drach@bmstu.ru)pro

г. Сочи, 2022 г.

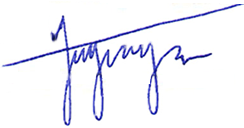
Автор:



Драч Е.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рецензент:

Руководитель направления по работе с ключевыми заказчиками

Siemens Industry Software

Product Lifecycle Management

Н.Г. Тутуев

Утверждена на заседании кафедры «Информационные технологии»

Протокол №6 от «\_\_22\_\_» \_\_02\_\_ 2022 г.

Заведующий кафедрой

А.С.Копырин \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Лабораторная работа №8. Сопряжение систем через API

Цель работы: формирование практических навыков сопряжения микросервисов через API на примере операционной системы **AlmaLinux.**

**Предварительные замечания**

* Предполагается, что перед началом установки VirtualBox уже есть на вашем ПК.
* Потребуется соединение с интернетом
* Потребуется работающий клиент Telegram (хотя бы один на подгруппу)

**Краткие теоретические сведения**

**Telegram** (от др.-греч. τῆλε «далеко» + др.-греч. γράμμα «запись») — кроссплатформенная система мгновенного обмена сообщениями (мессенджер) с функциями VoIP, позволяющая обмениваться текстовыми, голосовыми и видео- сообщениями, стикерами и фотографиями, файлами многих форматов. Также можно совершать видео- и аудио- звонки и трансляции в каналах и группах, организовывать конференции, многопользовательские группы и каналы. Клиентские приложения Telegram доступны для Android, iOS, Windows Phone, Windows, macOS и Linux.

**API** (МФА [ˌeɪ.piˈaɪ]; аббр. от англ. Application Programming Interface — «программный интерфейс приложения») — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо интернет-протокола, программного каркаса (фреймворка) или стандарта вызовов функций операционной системы. Часто реализуется отдельной программной библиотекой или сервисом операционной системы. Используется программистами при написании всевозможных приложений.

**Бот** (англ. bot, сокращение от чеш. robot) — виртуальный робот, который функционирует на основе специальной программы, выполняющий автоматически и/или по заданному расписанию какие-либо действия через интерфейсы, предназначенные для людей. Боты находят также применение в условиях, когда требуется лучшая реакция по сравнению с возможностями человека (например, игровые боты, боты для интернет-аукционов и тому подобное) или, реже, для имитации действий человека (например, боты для чатов и тому подобное). Чат-бот может выдать достаточно адекватный ответ на вопрос, сформулированный на правильном русском языке (или любом другом, работа с которым поддерживается). Такие боты часто применяются для сообщения прогноза погоды, результатов спортивных соревнований, курсов валют, биржевых котировок и тому подобное.

**Чат-бот** — это программа, работающая внутри мессенджера, например Telegram. Такая программа способна отвечать на вопросы, а также самостоятельно задавать их. Чат-боты используются в разных сферах для решения типовых задач.

C точки зрения прикладной информатики (и прикладной информатики в экономике в частности), создание чат-бота — это не только следование последним тенденциям. Прежде всего, речь идёт о сокращении затрат, увеличении конверсии и улучшении качества обслуживания клиентов. Сотни компаний делают всё возможное, чтобы достичь вышеупомянутых целей. И для их достижения необходимы боты как первостепенный инструмент.

Согласно текущей статистике чат-ботов за 2020 год, недвижимость является ведущей отраслью по прибылям от чат-ботов (28 %). Затем идут путешествия (16 %), образование (14 %), здравоохранение (10 %) и финансы (5 %).

Чат-боты могут использоваться в службах поддержки, помогая решить простые вопросы, например, такие как смена пароля.

Чат-боты можно использовать для поиска информации. Например, прогноз погоды, афиша мероприятий.

В сфере путешествий стали одними из первых использовать чат-боты. Чат-бот может предложить направления/рейсы/рестораны — на основании поисковых запросов и предпочтений пользователя. После покупки программа обеспечивает клиентскую поддержку, предоставляя ответы на часто задаваемые вопросы.

С помощью ботов функционал приложения Telegram практически не ограничен.

**Практическая часть**

Реализуем бота для Telegram, воспользовавшись мощностями собственного сервера под управлением Linux и языка Python. Воспользуемся API, благо документация исчерпывающая.

Чтобы создать бота, нам нужно дать ему **название**, **адрес** и получить **токен** — цифро-символьную строку, которая будет однозначно идентифицировать бота.

Для выполнения всей ЛР достаточно создать одного бота. Чтобы создать бота в Telegram потребуется (как это ни странно) обратиться к боту: есть удивительный бот BotFather - «прародитель всех ботов».

Необходимо найти бота, нажать кнопку «Запустить» (или написать /start), в ответ BotFather пришлет список доступных команд.

Отправим боту команду /newbot, чтобы создать нового бота. В ответ он попросит ввести имя будущего бота, его можно писать на русском. После ввода имени нужно будет изобрести адрес бота, причем он должен заканчиваться на слово bot. Например, sgubot или sgu\_bot. Если адрес уже кем‑то занят, придётся придумывать новый.

Когда бот будет создан, мы получим сообщение, в котором будет исчерпывающая техническая информация о боте. Обратите внимание, что после фразы Use this token to access the HTTP API будет написана строка из букв и цифр — это токен.

Далее на сервере устанавливаем 3-ю версию Python

dnf update

dnf install epel-release

yum install python3

(если не сработало, то yum install python39 или yum install python)

Проверяем версию:

python3 --version

Внимание! В дальнейшем, скорее всего, обращаться к среде Python надо будет с явным указанием версии, т.е. python3.

~~Как мы знаем из других курсов~~, Python имеет **собственный** менеджер пакетов (это нечто аналогичное yum или dnf) с благозвучным названием **pip**. Это система управления пакетами, которая используется для установки/управления программными пакетами, написанными на Python.

Для взаимодействия с Telegram API средствами языка Python разработаны несколько модулей. Самый простой из них — Telebot. Устанавливаем необходимый пакет:

pip3 install pytelegrambotapi

В редчайших случаях, возможно, понадобится написать просто pip вместо pip3.

## Задача 1. ЭХО

Сначала реализуем простейшего бота – для понимания процесса. Он будет получать от пользователя текстовое сообщение и возвращать его. Создаём текстовый файл с расширением .py и вставляем в него код:

import telebot

# Создаем экземпляр бота

bot = telebot.TeleBot('Сюда вставить токен для идентификации бота')

# Функция, обрабатывающая команду /start

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

def start(m, res=False):

bot.send\_message(m.chat.id, 'К работе готов! Давайте общаться.')

# Получение сообщений от пользователя

@bot.message\_handler(content\_types=["text"])

def handle\_text(message):

bot.send\_message(message.chat.id, 'Вы написали: ' + message.text)

# Запускаем бота

bot.polling(none\_stop=True, interval=0)

Запускать скрипт будем командой:

python3 ./имя\_скрипта.py

Можно запускать скрипт более изящно - сразу уводить его в фон, чтобы консоль оставалась свободной:

python3 ./имя\_скрипта.py &

При таком подходе можно запускать сколько угодно скриптов в фоновом режиме, а узнавать что запущено командой jobs. Избавляемся от лишних выполняющихся скриптов с помощью доброй команды kill, которой передаётся PID процесса.

Если всё запустилось без ошибок, переходим в Telegram – проверять работу бота. Ищем бота по придуманному ранее адресу. Запускаем бота кнопкой «Запустить» (Start) или командой /start. При правильной работе бот возвращает сообщения отправителю, расшатывая нервную систему.

Задача 2. Бот-всезнайка

Более сложный бот, который будет давать определения для введённых терминов. Черпать информацию будем из Википедии по той причине, что для этого уже разработан прекрасный модуль.

pip3 install wikipedia

В файл сохраним следующий код и запустим его.

import telebot, wikipedia, re

# Создаем экземпляр бота

bot = telebot.TeleBot('Сюда вставить токен для идентификации бота')

# Выбираем русский язык в Wikipedia

wikipedia.set\_lang("ru")

# Чистим текст статьи в Wikipedia и ограничиваем его тысячей символов

def getwiki(s):

try:

ny = wikipedia.page(s)

# Получаем первую тысячу символов

wikitext=ny.content[:1000]

# Разделяем по точкам

wikimas=wikitext.split('.')

# Отбрасываем всё после последней точки

wikimas = wikimas[:-1]

# Создаем пустую переменную для текста

wikitext2 = ''

# Проходимся по строкам, где нет знаков «равно» (то есть все, кроме заголовков)

for x in wikimas:

if not('==' in x):

# Если в строке осталось больше трех символов, добавляем ее к нашей переменной и возвращаем утерянные при разделении строк точки на место

if(len((x.strip()))>3):

wikitext2=wikitext2+x+'.'

else:

break

# Теперь при помощи регулярных выражений убираем разметку

wikitext2=re.sub('\([^()]\*\)', '', wikitext2)

wikitext2=re.sub('\([^()]\*\)', '', wikitext2)

wikitext2=re.sub('\{[^\{\}]\*\}', '', wikitext2)

# Возвращаем текстовую строку

return wikitext2

# Обрабатываем исключение, которое мог вернуть модуль wikipedia при запросе

except Exception as e:

return 'В энциклопедии нет информации об этом'

# Функция, обрабатывающая команду /start

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

def start(m, res=False):

bot.send\_message(m.chat.id, 'Отправьте мне любое слово, и я найду его значение на Wikipedia')

# Получение сообщений от юзера

@bot.message\_handler(content\_types=["text"])

def handle\_text(message):

bot.send\_message(message.chat.id, getwiki(message.text))

# Запускаем бота

bot.polling(none\_stop=True, interval=0)

## Задача 3. БОТ С ДВУМЯ ВИРТУАЛЬНЫМИ КНОПКАМИ

Во многих ботах для выбора каких‑то действий применяются так называемые виртуальные кнопки. Реализуем ветвление. Пользователь будет выбирать, что он хочет прочитать (например, афоризмы или новости).

Потребуется создать два файла: thinks.txt и facts.txt, которые содержат список сегодняшних новостей и афоризмы. Одна строка – одно утверждение. Хотя бы три строки в каждом.

Важно! Если использовать для этого бота тот же токен, что и для предыдущего, то, чтобы увидеть кнопки, потребуется перезапуск командой /start.

import telebot

import random

from telebot import types

# Загружаем список интересных фактов

f = open('data/facts.txt', 'r', encoding='UTF-8')

facts = f.read().split('\n')

f.close()

# Загружаем список поговорок

f = open('data/thinks.txt', 'r', encoding='UTF-8')

thinks = f.read().split('\n')

f.close()

# Создаем бота

bot = telebot.TeleBot('Вставить свой токен')

# Команда start

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

def start(m, res=False):

# Добавляем две кнопки

markup=types.ReplyKeyboardMarkup(resize\_keyboard=True)

item1=types.KeyboardButton("Новость")

item2=types.KeyboardButton("Афоризм")

markup.add(item1)

markup.add(item2)

bot.send\_message(m.chat.id, 'Нажми: \nНОВОСТЬ для получения сегодняшних новостей\nАФОРИЗМ — для получения лучших афоризмов всех времён ', reply\_markup=markup)

# Получение сообщений от клиента

@bot.message\_handler(content\_types=["text"])

def handle\_text(message):

if message.text.strip() == 'Факт' :

answer = random.choice(facts)

elif message.text.strip() == 'Поговорка':

answer = random.choice(thinks)

# Отсылаем сообщение в чат пользователя

bot.send\_message(message.chat.id, answer)

# Запускаем бота

bot.polling(none\_stop=True, interval=0)

## Задача 4. БОТ, ВЕДУЩИЙ КАНАЛ

Разработаем бота, который работает полностью в автоматическом режиме: по таймеру. На вход бот будет получать список анекдотов, которые затем будет отправлять в канал через заданный интервал времени. Потребуется создать свой канал в Telegram, добавить в подписчики канала нашего бота и назначить его администратором канала с правом публиковать сообщения.

Файл с анекдотами должен лежать в папке data рядом со скриптом бота.

import telebot

import time

# Токен, который выдает @botfather

bot = telebot.TeleBot('Вставить токен, полученный от @botfather')

# Адрес телеграм-канала, начинается с @

CHANNEL\_NAME = '@адрес\_твоего\_канала'

# Загружаем список шуток

f = open('data/fun.txt', 'r', encoding='UTF-8')

jokes = f.read().split('\n')

f.close()

# Пока не закончатся шутки, посылаем их в канал

for joke in jokes:

bot.send\_message(CHANNEL\_NAME, joke)

# Делаем паузу в шесть минут

time.sleep(360)

bot.send\_message(CHANNEL\_NAME, "Анекдоты закончились")

## Задача 5. ЧАТ-БОТ

Разработаем чат‑бота, который будет поддерживать диалог с пользователем. Для этого мы подготовим файл boltun.txt, содержащий строки с вопросами и ответами на них в следующей строке. Вопрос будем помечать специальным образом в начале строки - **u:**.

u: Привет

Привет!

u: как зовут

Федя меня зовут!

u: как дела

Лучше всех!

Файл boltun.txt поместим в папку data рядом со скриптом бота. Для поиска похожих вопросов используем модуль fuzzywuzzy, который позволяет сравнивать, насколько похожи между собой две строки. Естественно, этот модуль нужно установить:

pip3 install fuzzywuzzy

pip3 install python-Levenshtein

Ниже приведен исходный код бота. После его запуска инициируем диалог с ботом словом «Привет». Естественно, это не искусственный интеллект, и набор его ответов ограничен фразами из фай­ла boltun.txt. Однако, чем больше входной файл, тем больше переписка с ботом будет напоминать осмысленный диалог.

import telebot

import os

from fuzzywuzzy import fuzz

# Создаем бота, пишем свой токен

bot = telebot.TeleBot('Вставить токен')

# Загружаем список фраз и ответов в массив

mas=[]

if os.path.exists('data/boltun.txt'):

f=open('data/boltun.txt', 'r', encoding='UTF-8')

for x in f:

if(len(x.strip()) > 2):

mas.append(x.strip().lower())

f.close()

# С помощью fuzzywuzzy определяем наиболее похожую фразу и выдаем в качестве ответа следующий элемент списка

def answer(text):

try:

text=text.lower().strip()

if os.path.exists('data/boltun.txt'):

a = 0

n = 0

nn = 0

for q in mas:

if('u: ' in q):

# Изучаем, насколько похожи две строки

aa=(fuzz.token\_sort\_ratio(q.replace('u: ',''), text))

if(aa > a and aa!= a):

a = aa

nn = n

n = n + 1

s = mas[nn + 1]

return s

else:

return 'Не смог'

except:

return 'Ошибка'

# Команда «Старт»

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

def start(m, res=False):

bot.send\_message(m.chat.id, 'Я на связи. Напиши мне Привет!')

# Получение сообщений от клиента

@bot.message\_handler(content\_types=["text"])

def handle\_text(message):

# Запись логов

f=open('data/' + str(message.chat.id) + '\_log.txt', 'a', encoding='UTF-8')

s=answer(message.text)

f.write('u: ' + message.text + '\n' + s +'\n')

f.close()

# Отправка ответа

bot.send\_message(message.chat.id, s)

# Запускаем бота

bot.polling(none\_stop=True, interval=0)

## Ход выполнения работы

1. В Telegram создать группу и бота.
2. На сервере под управлением Linux реализовать 5 вышеописанных ботов (back-end): модифицировать скрипты из примеров (вставить уникальный токен) и запустить выполнение
3. Подготовить уникальные входные текстовые файлы самостоятельно.
4. Проверить работоспособность ботов в Telegram

## Содержание отчёта

* Титульный лист
* Цель
* Описание проделанных действий
* Содержимое входных текстовых файлов
* Скриншоты, показывающие работоспособность ботов
* Выводы о достижении цели и полученных результатах

## ВЫВОДЫ

Разработаны и запущены пять простых ботов, на примере которых научились принимать и отправлять сообщения, реализовывать ветвление и понимать неточные запросы.