МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Кемеровский государственный университет»**

**Институт фундаментальных наук**

**Кафедра ЮНЕСКО по ИВТ**

**ЛАБОРАТОРНАЯ РАБОТА №2**

**ПО ДИСЦИПЛИНЕ “Автоматизация процессов развертывания и инсталляции информационных систем”**

Направление 09.03.03 – Прикладная информатика в экономике

Студента 2 курса

Башкеева Степана Дмитриевича

Преподаватель:

И.Ю. Степанов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работа защищена:

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022г.

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_

Кемерово 2022

Содержание

[Цель работы 3](#_Toc99423208)

[Задачи 3](#_Toc99423209)

[Описание предметной области 3](#_Toc99423210)

[Практическая часть 3](#_Toc99423211)

[Заключение 7](#_Toc99423212)

[Список литературы 8](#_Toc99423213)

[Приложение 1 9](#_Toc99423214)

# Цель работы

Написать бота для Telegram на python и разместить его на облачной платформе pythonanywhere.

# Задачи

1. Написать телеграм-бота на языке программирования Python.
2. Разместить телеграм-бота на облачной платформе pythonanywhere.

# Описание предметной области

Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

PythonAnywhere — название облачной платформы, предназначенной преимущественно для запуска приложений Python.

Telegram — кроссплатформенная система мгновенного обмена сообщениями с функциями VoIP, позволяющая обмениваться текстовыми, голосовыми и видеосообщениями, стикерами и фотографиями, файлами многих форматов.

# Практическая часть

1. Написать бота, который будет выводить ссылки на популярные сферы IT.

Для вывода ссылок на популярные сайты IT и информацию о них, необходимо создать меню с кнопками, которые будут выводить необходимую информацию. (Рис. 1). [1]

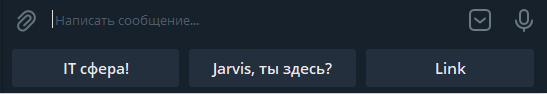


Рис. 1.Меню кнопок.

Также можно добавить подменю, чтобы его открыть необходимо нажать кнопку в главном меню «IT сфера», в которой находятся кнопки, которые будут выводить информацию. (Рис. 2).

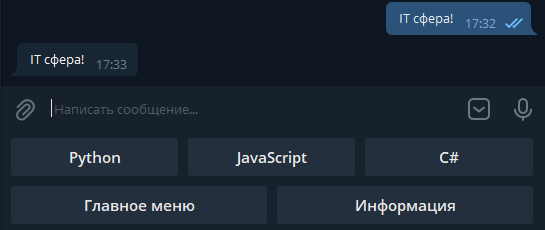


Рис. 2. Новое меню «IT сфера».

Ниже показана работа каждой из кнопок. (Рис. 3-5).

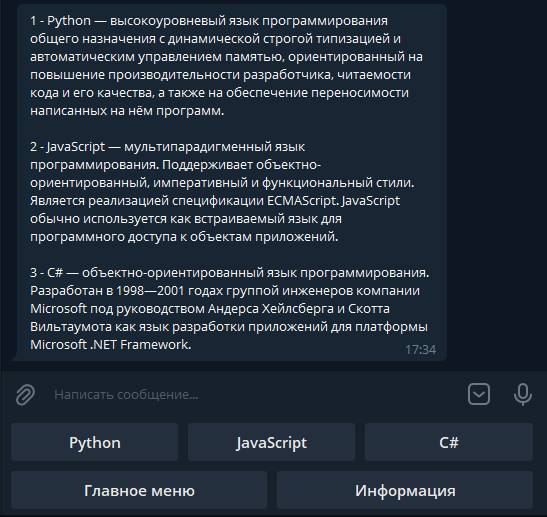


Рис. 3. Результат работы кнопки «Информация».



Рис. 4. Результат работы кнопки “Python”.

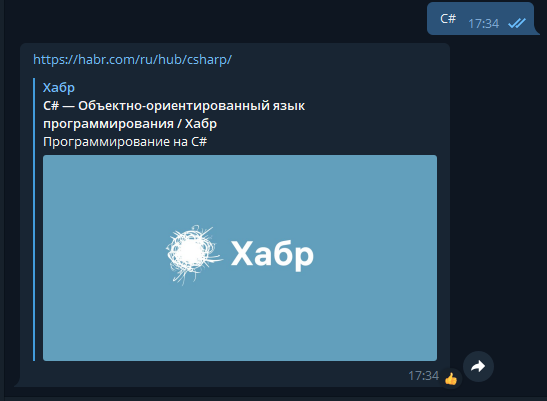


Рис. 5. Результат работы кнопки “C#”.

1. Написать бота, который будет реагировать на голосовые сообщения.

При отправке голосового сообщения, бот сообщит «Вы отправили голосовое сообщение.». (Рис. 6). [2]

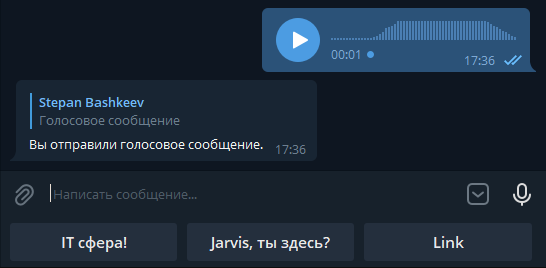


Рис. 6. Реакция бота на голосовое сообщение.

1. Написать бота, который выведет введенное ему сообщение.

Чтобы «поговорить» с ботом необходимо нажать кнопку «Jarvis, ты здесь?», после чего бот отправит сообщение, в котором спросит ваше имя, пользователь должен отправить сообщение со своим именем. Далее бот отправит сообщение с именем пользователя. (Рис. 7).

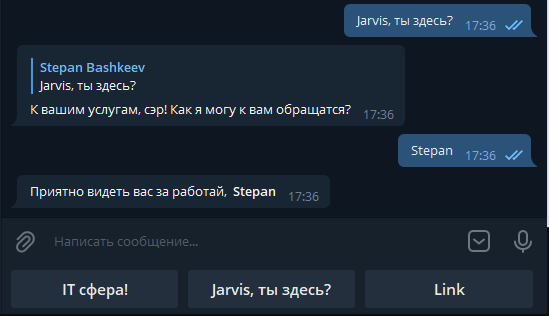


Рис. 7. Вывод введённого сообщения.

1. Написать команду “/close”, которая будет закрывать клавиатуру.

При отправки команды «/close» клавиатура будет закрыта, чтобы её вновь открыть необходимо ввести команду «/start». (Рис. 8).



Рис. 8. Результат работы команды “/close”.

1. Написать бота, который отправит ответ сервера.

Чтобы получить ответ с сервера необходимо нажать на кнопку «Link», после бот отправит сообщение «Введите ссылку», вводим необходимую ссылку и получаем ответ сервера. (Рис. 9).

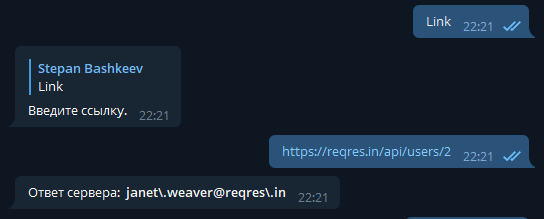


Рис. 9. Результат работы кнопки “Link”.

1. Размещение бота на pythonanywhere.com.
2. Регистрируемся в pythonanywhere.com.
3. Заходим в пункт Files, создаем новую папку (например: bot)

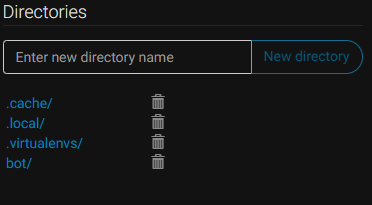


Рис. 10. Создание папки.

1. Загружаем все файлы в ранее созданную папку, связанные с ботом.

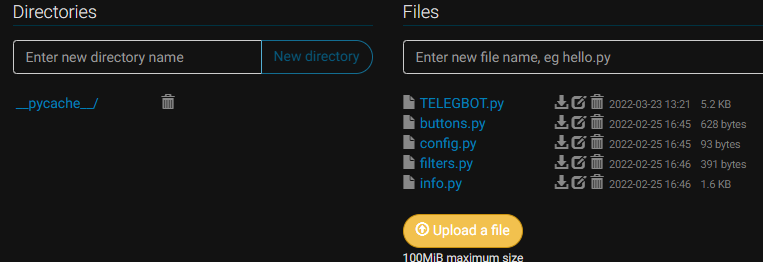


Рис. 11. Загрузка файлов.

1. Переходим в пункт Consoles, открываем “Bash console”.
2. В открывшейся консоли нужно установить все библиотеки, которые были использованы при создании бота (например: pip install aiogram).

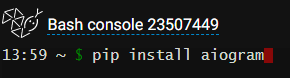


Рис. 12. Bash console.

1. В конце запускаем основной файл через консоль или открываем файл с ботом и запускаем нажав на кнопку “Run”.

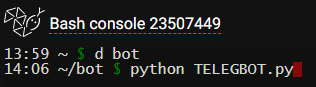


Рис. 13. Запуск бота через консоль.

1. Проверка работоспособности бота.

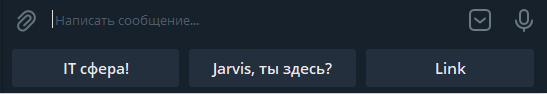


Рис. 14. Проверка бота.

# Заключение

В результате работы была изучена документация библиотеки aiogram. На основе изученной библиотеки был создан бот для Telegram. Изучена облачна платформа pythonanywhere, на данной платформе был размещен раннее созданный бот, получен опыт работы платформой. Также получены навыки работы с python, опыт работы с ботом для Telegram.

# Список литературы

1. Документация библиотеки aiogram [Электрон. ресурс] / URL- <https://docs.aiogram.dev/en/latest/>
2. Aiogram\_bot [Электрон. ресурс] / URL - <https://github.com/aiogram/bot#readme>
3. Echo bot [Электрон. ресурс] / URL - <https://docs.aiogram.dev/en/latest/ex>

# Приложение 1

Листинг 1. Код бота

from aiogram.dispatcher import FSMContext  
from aiogram.dispatcher.filters import Text  
from aiogram.contrib.fsm\_storage.memory import MemoryStorage  
from aiogram.types import ParseMode  
from aiogram.utils import executor  
  
import config  
import logging  
import buttons as btn  
import aiogram.utils.markdown as md  
import requests as req  
import aiohttp  
import asyncio  
  
from aiogram import Bot, Dispatcher, executor, types  
  
from aiogram.dispatcher.filters.state import State, StatesGroup  
  
from filters import IsAdminFilter  
#import infomations text  
from info import info\_lang, py\_link, js\_link, c\_link  
  
#log level  
logging.basicConfig(level=logging.INFO)  
  
#bot init  
bot = Bot(token=config.TOKEN)  
dp = Dispatcher(bot, storage=MemoryStorage())  
storage = MemoryStorage()  
  
#activate filters  
dp.filters\_factory.bind(IsAdminFilter)  
  
#class State  
class Mudialog(StatesGroup):  
 res\_user = State()  
  
#start bot message  
@dp.message\_handler(commands=['start'])  
async def command\_start(message: types.Message):  
 await bot.send\_message(message.from\_user.id, 'Hello {0.first\_name}'.format(message.from\_user), reply\_markup=btn.mainMenu)  
  
#remove keyboard  
@dp.message\_handler(commands=['close'])  
async def command\_start(message: types.Message):  
 await bot.send\_message(message.from\_user.id, 'Клавиатура закрыта!', reply\_markup=types.ReplyKeyboardRemove())  
  
  
class Form(StatesGroup):  
 name = State()  
 link = State()  
  
  
  
#buttons  
@dp.message\_handler()  
async def bot\_message(message: types.Message):  
 if message.text == 'IT сфера!':  
 await bot.send\_message(message.from\_user.id, 'IT сфера!', reply\_markup=btn.otherMenu)  
  
 elif message.text == 'Главное меню':  
 await bot.send\_message(message.from\_user.id, 'Главное меню', reply\_markup=btn.mainMenu)  
  
 elif message.text == 'Python':  
 await bot.send\_message(message.from\_user.id, py\_link)  
  
 elif message.text == 'JavaScript':  
 await bot.send\_message(message.from\_user.id, js\_link)  
  
 elif message.text == 'C#':  
 await bot.send\_message(message.from\_user.id, c\_link)  
  
 elif message.text == 'Информация':  
 await bot.send\_message(message.from\_user.id, info\_lang)  
  
 elif message.text == 'Jarvis, ты здесь?':  
 await Form.name.set()  
 await message.reply("К вашим услугам, сэр! Как я могу к вам обращатся?")  
  
 elif message.text == 'Link':  
 await Form.link.set()  
 await message.reply('Введите ссылку.')  
  
  
@dp.message\_handler(state='\*', commands='cancel')  
@dp.message\_handler(Text(equals='отмена', ignore\_case=True), state='\*')  
async def cancel\_handler(message: types.Message, state: FSMContext):  
 current\_state = await state.get\_state()  
 if current\_state is None:  
 return  
  
 await state.finish()  
 await message.reply('ОК')  
  
#input name  
@dp.message\_handler(state=Form.name)  
async def process\_name(message: types.Message, state: FSMContext):  
 async with state.proxy() as data:  
 data['name'] = message.text  
  
 async with state.proxy() as data:  
  
 await bot.send\_message(  
 message.chat.id,  
 md.text(  
 md.text('Приятно видеть вас за работай, ', md.bold(data['name'])),  
 sep='\n',  
 ),  
 parse\_mode=ParseMode.MARKDOWN,  
 )  
  
 await state.finish()  
  
#outlink  
@dp.message\_handler(state=Form.link)  
async def process\_link(message: types.Message, state: FSMContext):  
 async with state.proxy() as data:  
 try:  
 lk = req.get(message.text)  
 data['link'] = lk.json()['data']['email']  
 except:  
 data['link'] = 'Ошибка'  
  
 async with state.proxy() as data:  
  
 await bot.send\_message(  
 message.chat.id,  
 md.text(  
 md.text('Ответ сервера: ', md.bold(data['link'])),  
 sep='\n',  
 ),  
 parse\_mode=ParseMode.MARKDOWN,  
 )  
  
 await state.finish()  
#voice message  
@dp.message\_handler(content\_types=['voice'])  
async def message\_voice\_handler(message: types.Message):  
 await message.reply("Вы отправили голосовое сообщение.")  
  
#run long-polling  
if \_\_name\_\_ == "\_\_main\_\_":  
 executor.start\_polling(dp, skip\_updates=True)