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 №4**

**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](#__RefHeading___Toc1742_1493232123)

[Описание предметной области 4](#__RefHeading___Toc1744_1493232123)

[Практическая часть 5](#__RefHeading___Toc1746_1493232123)

[Заключение 8](#__RefHeading___Toc1748_1493232123)

[Список литературы 9](#__RefHeading___Toc1750_1493232123)

[Приложение 1 10](#__RefHeading___Toc1752_1493232123)

# Цель работы

Изучить и научиться работать с облачной платформой Heroku, разместить бота для Telegram на данной платформе.

Задачи

* Изучить работу облачной платформы Heroku.
* Разместить бота на облачной платформе Heroku.

# Описание предметной области

Heroku — облачная PaaS-платформа, поддерживающая ряд языков программирования. С 2010 года является дочерней компанией Salesforce.com. Heroku, одна из первых облачных платформ, появилась в июне 2007 года и изначально поддерживала только язык программирования Ruby, но на данный момент список поддерживаемых языков также включает в себя Java, Node.js, Scala, Clojure, Python, Go, Ruby и PHP. На серверах Heroku используются операционные системы Debian или Ubuntu. [1]

# Практическая часть

1. Регистрируемся на облачной платформе Heroku.

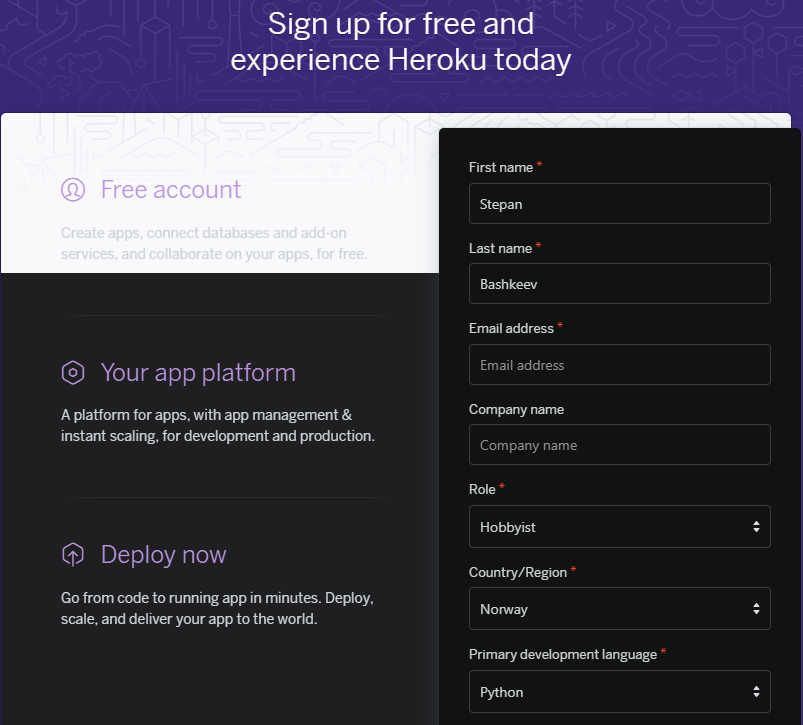


Рис. 1. Регистрация на облачной платформе Heroku. [1].

1. После регистрации скачиваем Heroku CLI.

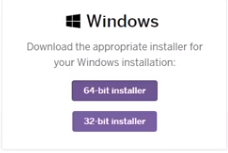


Рис. 2. Скачивание Heroku CLI.

1. После скачивания необходимых компонентов, нужно открыть консоль и вписать команду - heroku login. Это необходимо для авторизации в Heroku.

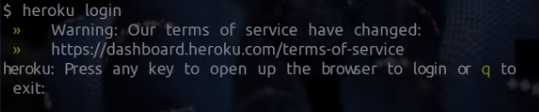


Рис. 3. Команда - heroku login.

1. Далее пройти авторизацию в браузере.

Откроется браузер со страницей авторизации на Heroku, нажимаем кнопку “Log In”.

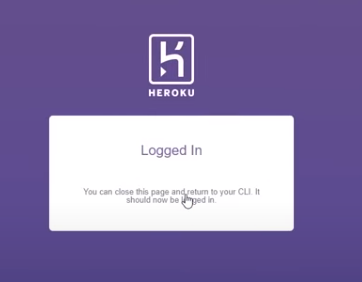


Рис. 4. Авторизация в браузере.

1. Затем необходимо в корневой директории бота создать 3 файла.

Создаем файл “runtime.txt”, в нём указываем версию Python, которую вы используете.

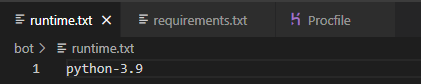


Рис. 5. Файл runtime.txt.

Далее создаем файл “requirements.txt” и указываем библиотеки, которые использовались при создании бота.

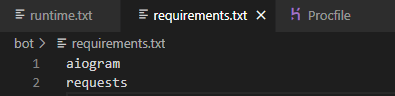


Рис. 6. Файл requirements.txt.

В конце создаем файл “Procfile” без указания расширения, в нем необходимо указать название главного файла бота. Для этого впишем “worker: python TELEGBOT.py”.

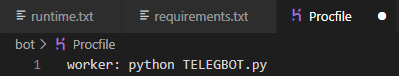


Рис. 7. Файл Procfile.

1. Далее необходимо в консоли вписать команды.

* git init
* git add
* git commit –m “First release”
* git heroku create (название приложения)
* git remote –v
* git heroku master
* heroku ps:scale worker=1

После ввода этих команд бот будет размещён и запущен на Heroku.

1. Заходим в Telegram, открываем бота и убеждаемся, что он работает.

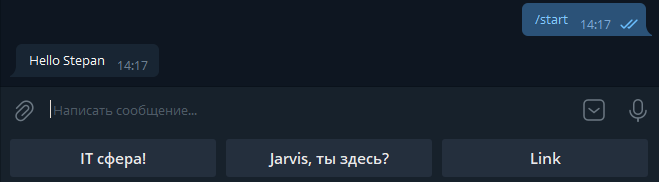


Рис. 8. Проверка работоспособности бота.

# Заключение

В результате работы была изучена облачная платформа Heroku. На данной платформе был размещен и запущен бот для Telegram. Также был получен опыт работы с облачной платформой Heroku.

# Список литературы

1. Облачная платформа Heroku [Электрон. ресурс] \ URL - [https://www.heroku.com](https://www.heroku.com/) .
2. Git [Электрон. ресурс] \ URL - [https://git-scm.com/downloads](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbUs2NFdKcDFXYmliLXJHOHNNRi1lTjFEamwyUXxBQ3Jtc0trUlpNdXVKUzBsenRCYUcxWXNUUW9MRzFrcUZrRnFfazlMU0xnYWxLOFhITU9QZ3FVRC1VTzVGelpfTHVNYzZYQldoOS1JcWxFY0ZxYllwdERLNHNIX01EV0hHSG9VdTQzNkJ3Z01tQ3M4c2h2X3R4QQ&q=https%3A%2F%2Fgit-scm.com%2Fdownloads)
3. SkillFactory Heroku [Электрон. ресурс] \ URL - <https://blog.skillfactory.ru/glossary/heroku/>

# 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)