Московский государственный технический

университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет технологий»

Отчет по лабораторной работе №6

«Разработка бота на основе конечного автомата для Telegram с использованием языка Python»

Выполнил:   
студент группы ИУ5-31Б   
Абуховский Иван

Подпись и дата:

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



06.12.2021

Проверил:  
преподаватель кафедры ИУ5  
Гапанюк Ю. Е.

Подпись и дата:

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

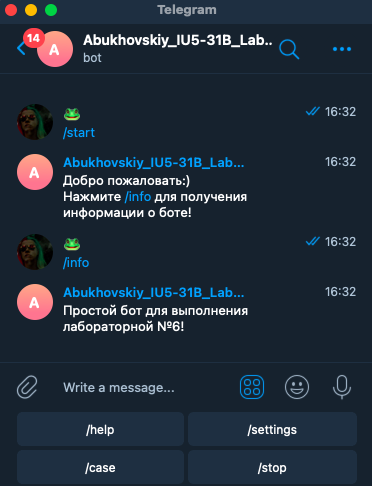
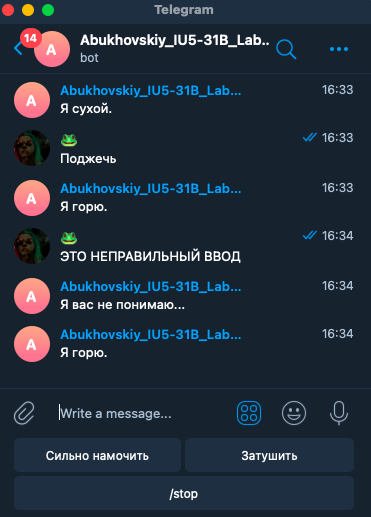
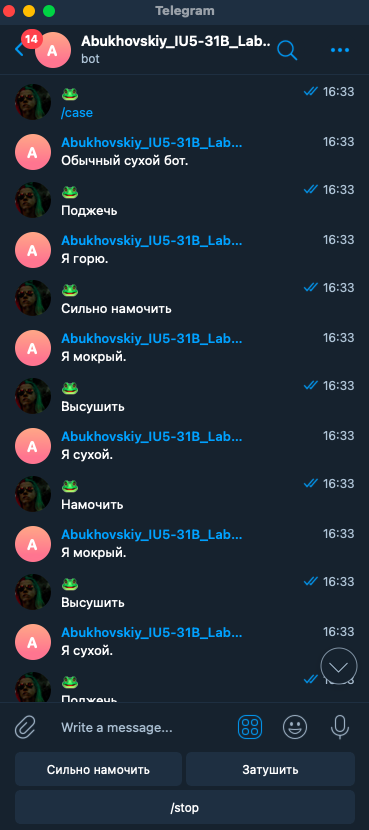
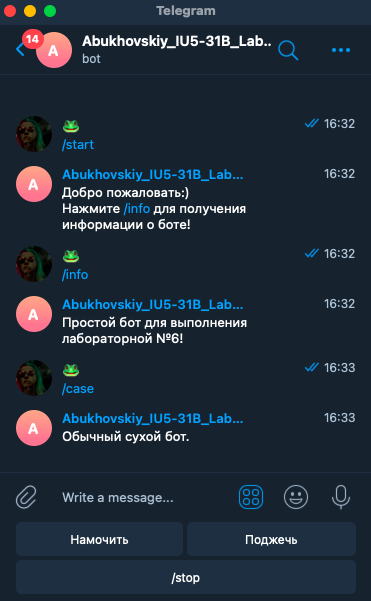
Москва, 2021 г.

**Описание задания:**

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

**Текст программы:**

import telebot  
from telebot import types  
  
# юзать from telebot import \* (импортирует все модули)  
  
###################################################################################################################################  
# Constants #  
Telegram\_API\_key = "5070903838:AAEBlIAsEinNbnQG87bxNpqjg6A-vrSwQXg" # API токен Telegram для обращения к серверам  
bot = telebot.TeleBot(Telegram\_API\_key) # упрощаем обращение через API, чтобы каждый раз не писать всю строчку  
upd = bot.get\_updates() # запрос обновлений у сервера, т.к. мы не имеем SSl сертификации сервера (нашего PC)  
last\_upd = upd[-1] # последнее событие (сообщение-команда)  
message\_from\_user = last\_upd.message # интуитивно понятно  
markup = types.ReplyKeyboardMarkup() # табличка кнопочек под полем ввода сообщения у юзера  
conditions = {"Намочить": "мокрый.",  
 "Сильно намочить": "мокрый.",  
 "Высушить": "сухой.",  
 "Затушить": "сухой.",  
 "Поджечь": "горю.",  
 "Сильно поджечь": "горю.",  
 "last\_condition": "сухой."}  
existing\_commands = "Намочить, Сильно намочить, Высушить, Высушить, Затушить, Поджечь, Сильно поджечь"  
  
  
###################################################################################################################################  
# Strat Commands #  
# далее блоки обработки команд, обернутые в декораторы  
# разберем на 1 примере  
@bot.message\_handler(commands=['start']) # команда /start  
def handle\_start(message): # сама ее функция  
 user\_markup = telebot.types.ReplyKeyboardMarkup(True, False) # вызываем табличку кнопочек  
 ''' Описание парамтров из документации:   
 one\_time\_keyboard (bool, optional) – Requests clients to hide the keyboard as soon as it’s been used.   
 The keyboard will still be available, but clients will automatically display the usual letter-keyboard in the chat -   
 the user can press a special button in the input field to see the custom keyboard again.   
 Defaults to False.  
  
 selective (bool, optional) –  
 Use this parameter if you want to show the keyboard to specific users only. Targets:  
 Users that are @mentioned in the text of the telegram.Message object.  
 If the bot’s message is a reply (has reply\_to\_message\_id), sender of the original message.  
 Defaults to False.  
 '''  
 user\_markup.row("/help", "/settings") # первая строка (раз, два)  
 user\_markup.row("/info", "/stop") # вторая строка (раз, два)  
 user\_markup.row("/case")  
 bot.send\_message(message.chat.id, """Добро пожаловать:)   
Нажмите /info для получения информации о боте!""",  
 reply\_markup=user\_markup) # ну и отправляем пользователю сообщение, чтобы он знал, что все работает  
 # в кноце reply\_markup = user\_markup прикрепляем те самые кнопочки  
  
  
@bot.message\_handler(commands=['info'])  
def handle\_info(message):  
 user\_markup = telebot.types.ReplyKeyboardMarkup(True, True)  
 user\_markup.row("/help", "/settings")  
 user\_markup.row("/case", "/stop")  
 bot.send\_message(message.chat.id, """  
 Простой бот для выполнения лабораторной №6!  
 """, reply\_markup=user\_markup)  
  
  
@bot.message\_handler(commands=['stop'])  
def handle\_stop(message):  
 user\_markup = telebot.types.ReplyKeyboardMarkup(True, True)  
 user\_markup.row("/help", "/settings")  
 user\_markup.row("/info", "/case")  
 bot.send\_message(message.chat.id, "Все-все", reply\_markup=user\_markup)  
  
  
@bot.message\_handler(commands=['help'])  
def handle\_help(message):  
 user\_markup = telebot.types.ReplyKeyboardMarkup(True, True)  
 user\_markup.row("/stop", "/settings")  
 user\_markup.row("/info", "/case")  
 bot.send\_message(message.chat.id, "Чем могу быть полезен?", reply\_markup=user\_markup)  
  
  
@bot.message\_handler(commands=['settings'])  
def handle\_help(message):  
 user\_markup = telebot.types.ReplyKeyboardMarkup(True, True)  
 user\_markup.row("/help", "/case")  
 user\_markup.row("/info", "/stop")  
 bot.send\_message(message.chat.id, "Укажите, что Вы хотели бы изменить в настройках бота.", reply\_markup=user\_markup)  
  
  
@bot.message\_handler(commands=['case'])  
def handle\_case(message):  
 user\_markup = telebot.types.ReplyKeyboardMarkup(True, True)  
 user\_markup.row("Намочить", "Поджечь")  
 user\_markup.row("/stop")  
 bot.send\_message(message.chat.id, "Обычный сухой бот.", reply\_markup=user\_markup)  
  
  
###################################################################################################################################  
# Text Commands #  
  
@bot.message\_handler(content\_types=['text'])  
def handle\_text(message):  
 user\_message = str(message.text)  
 user\_markup = telebot.types.ReplyKeyboardMarkup(True, True)  
 if user\_message in conditions.keys():  
 reply\_message = "Я " + conditions[user\_message]  
 conditions["last\_condition"] = conditions[user\_message]  
 if conditions[user\_message] == "мокрый.":  
 user\_markup.row("Сильно поджечь", "Высушить")  
 user\_markup.row("/stop")  
  
 elif conditions[user\_message] == "сухой.":  
 user\_markup.row("Намочить", "Поджечь")  
 user\_markup.row("/stop")  
  
 elif conditions[user\_message] == "горю.":  
 user\_markup.row("Сильно намочить", "Затушить")  
 user\_markup.row("/stop")  
  
 bot.send\_message(message.chat.id, reply\_message, reply\_markup=user\_markup)  
 else:  
 if conditions["last\_condition"] == "мокрый.":  
 user\_markup.row("Сильно поджечь", "Высушить")  
 user\_markup.row("/stop")  
  
 elif conditions["last\_condition"] == "сухой.":  
 user\_markup.row("Намочить", "Поджечь")  
 user\_markup.row("/stop")  
  
 elif conditions["last\_condition"] == "горю.":  
 user\_markup.row("Сильно намочить", "Затушить")  
 user\_markup.row("/stop")  
 bot.send\_message(message.chat.id, "Я вас не понимаю...")  
 bot.send\_message(message.chat.id, "Я " + conditions["last\_condition"], reply\_markup=user\_markup)  
  
  
'''  
Ну а тут мы просто вызываем polling, чтобы делать запросы на сервер нон-стопом.  
Обработка частых исключений, никак нам не мешающих, чтобы бот не падал просто так.  
'''  
  
try:  
 bot.polling(none\_stop=True, interval=0)  
except IndexError:  
 bot.polling(none\_stop=True, interval=0)  
except TypeError:  
 bot.polling(none\_stop=True, interval=0)

**Экранные формы с примерами выполнения программ**