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