**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

**Отчет по практической работе №3**

по дисциплине «Введение в информационные технологии» на тему:

Разработка простого Telegram-бота с помощью библиотеки telebot.

Выполнил: студент группы БВТ2104

Кравченко Николай Алексеевич

Проверил:

Мкртчян Грач Маратович

Москва

2021

**1 Постановка задачи**

Создать простого Telegram-бота на языке программирования Python, с использованием фреймворка Flask.

**Задание**

* Создать обработку трех любых сообщений
* Создать обработку трех любых команд
* Подробно расписать команду /help

2 Результат выполненной работы

Код bot.py:

import telebot

from config import \*

import db

import threading

from opencv import demotivator

bot = telebot.TeleBot(token)

@debug

def process\_image(message):

file\_info = bot.get\_file(message.photo[len(message.photo) - 1].file\_id)

downloaded\_file = bot.download\_file(file\_info.file\_path)

src = '/tmp/' + message.photo[0].file\_id + '.jpg'

with open(src, 'wb') as new\_file:

new\_file.write(downloaded\_file)

demotivator(src, db.get(message.chat.id).text)

with open(src, 'rb') as file:

bot.send\_photo(message.chat.id, file)

def state(value):

def checker(message):

return db.get(message.chat.id).state == value

return checker

def state\_change(f):

def ret\_fun(message):

x = f(message)

if x:

logging.info(f"Set state of {message.chat.id} to {x}")

db.get(message.chat.id).state = x

db.session.commit()

return ret\_fun

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

@bot.message\_handler(func=state(States.S\_START))

@state\_change

@log

def cmd\_start(message):

bot.send\_message(message.chat.id, "Привет! Как я могу к тебе обращаться?")

return States.S\_ENTER\_NAME

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

@log

def cmd\_start(message):

bot.send\_message(message.chat.id, "Я могу создать демотиватор.\nДля этого тебе необходимо представится, ввести текст и отправить изображение")

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

@state\_change

@log

def cmd\_reset(message):

bot.send\_message(message.chat.id, "Что ж, начнём по-новой. Как тебя зовут?")

return States.S\_ENTER\_NAME

@bot.message\_handler(func=state(States.S\_ENTER\_NAME))

@state\_change

@log

def user\_entering\_name(message):

db.get(message.chat.id).name = message.text.strip()

bot.send\_message(message.chat.id, "Отличное имя, запомню!\nТеперь ты можешь пользоваться сервисом")

bot.send\_message(message.chat.id, enter\_text)

return States.S\_ENTER\_TEXT

@bot.message\_handler(func=state(States.S\_ENTER\_TEXT))

@state\_change

@log

def user\_entering\_text(message):

text = message.text.strip()

if not text:

bot.send\_message(message.chat.id, "Что-то не так, введи текст ещё раз!")

return

if len(text) > 20:

bot.send\_message(message.chat.id, "Слишком длинный текст, попробуй еще раз...")

return

else:

db.get(message.chat.id).text = text

bot.send\_message(message.chat.id, "Теперь отправь мне фотографию.")

return States.S\_SEND\_PIC

@bot.message\_handler(func=state(States.S\_SEND\_PIC), content\_types=["photo"])

@state\_change

@log

def recive\_image(message):

threading.Thread(target=process\_image, args=(message,)).start()

bot.send\_message(message.chat.id, "Фото обрабатывается")

bot.send\_message(message.chat.id, "Если понадобится сделать новый демотиватор, введи текст")

return States.S\_ENTER\_TEXT

@bot.message\_handler()

@log

def recive\_image(message):

bot.send\_message(message.chat.id, "Что-то не так, попробуй ещё раз!")

bot.send\_message(message.chat.id, "Я жду что ты " + States.waiting\_for(db.get(message.chat.id).state))

if \_\_name\_\_ == '\_\_main\_\_':

bot.infinity\_polling()

Код config.py:

from os import environ

import logging

from functools import partial

if environ["DEBUG"] == "YES":

logging.basicConfig(level=logging.DEBUG,

format="%(asctime)s - %(levelname)s : %(message)s")

else:

logging.basicConfig(level=logging.INFO,

format="%(asctime)s - %(levelname)s : %(message)s")

def debug(f, logger\_normal=logging.debug):

def dec(\*args, \*\*kwargs):

s = f"Run {f.\_\_name\_\_} with args( {args}{' | ' + str(kwargs) if kwargs else ''} )"

try:

logger\_normal(s)

return f(\*args, \*\*kwargs)

except Exception as e:

logging.error(f"Error while {s} with message {e}")

return dec

log = partial(debug, logger\_normal=logging.info)

token = environ["TELEGRAM\_BOT\_TOKEN"]

wait\_list=["начнешь беседу","введешь свое имя","введешь текст для демотиватора","отправишь мне картинку"]

class States:

S\_START = 0 # Начало нового диалога

S\_ENTER\_NAME = 1

S\_ENTER\_TEXT = 2

S\_SEND\_PIC = 3

def waiting\_for(*state*):

return wait\_list[*state*]

enter\_text = "Введи текст для демотиватора"

Код подключения к базе данных:

from config import \*

from sqlalchemy import \*

from sqlalchemy.orm import declarative\_base, Session

Base = decarative\_base()

engine = create\_engine(

f'postgresql+psycopg2://{environ["POSTGRES\_USER"]}:{environ["POSTGRES\_PASSWORD"]}@{environ["URL\_DB"]}:{environ["PORT\_DB"]}/{environ["POSTGRES\_DB"]}')

session = Session(engine, future=True)

class User(Base):

\_\_tablename\_\_ = 'users'

id = Column(Integer, primary\_key=True)

chat\_id = Column(Integer, nullable=False)

state = Column(Integer, nullable=False, default=0)

name = Column(String, nullable=True)

text = Column(String, nullable=True)

count = Column(Integer, nullable=False, default=0)

def \_\_repr\_\_(*self*):

return ' | '.join(map(str, (*self*.id, *self*.chat\_id, *self*.state, *self*.name, *self*.text, *self*.count)))

@debug

def get(chat\_id):

result = session.query(User).filter(User.chat\_id == chat\_id).first()

if result:

logging.debug(f"{chat\_id} is {result}")

else:

result = User(chat\_id=chat\_id)

session.add(result)

session.commit()

return result

docker-compose.yml:

version: '3'

services:

database:

build: postgres

image: postgres\_init

env\_file:

- .env

- secret.env

volumes:

- database-data:/var/lib/postgresql/data/

ports:

- ${PORT\_DB}:${PORT\_DB}

restart: unless-stopped

healthcheck:

test: ["CMD-SHELL", "pg\_isready -U ${POSTGRES\_USER} -d ${POSTGRES\_DB}"]

interval: 5s

timeout: 3s

retries: 5

start\_period: 10s

bot:

build: bot

image: bot

env\_file:

- .env

- secret.env

depends\_on:

database:

condition: service\_healthy

restart: unless-stopped

volumes:

database-data:

**3 Вывод**

В результате проделанной работы мы изучили принципы создания Telegram-ботов и конечных автоматов.