**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №5

«Разработка простого бота для Telegram с использованием языка Python»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б |  | преподаватель каф. ИУ5 |
| Мажитов В. |  | 1. Гапанюк Ю.Е. |
|  |  |  |

**Описание задания**

Разработайте простого бота для Telegram. Бот должен использовать функциональность создания кнопок.

**Текст программы**

Файл bot.py

import asyncio

import logging

from aiogram import Bot, Dispatcher

from aiogram.contrib.fsm\_storage.memory import MemoryStorage

from aiogram.contrib.fsm\_storage.redis import RedisStorage2

from tgbot.config import load\_config

from tgbot.handlers.dart import register\_dart

from tgbot.handlers.dice import register\_dice

from tgbot.handlers.bowl import register\_soccer

from tgbot.handlers.user import register\_user

logger = logging.getLogger(\_\_name\_\_)

def register\_all\_handlers(dp):

register\_user(dp)

register\_dice(dp)

register\_dart(dp)

register\_soccer(dp)

async def main():

config = load\_config()

storage = MemoryStorage()

bot = Bot(token=config.telegram.token, parse\_mode='HTML')

dp = Dispatcher(bot, storage=storage)

register\_all\_handlers(dp)

try:

await dp.start\_polling()

finally:

await dp.storage.close()

await dp.storage.wait\_closed()

await bot.session.close()

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

try:

asyncio.run(main())

except (KeyboardInterrupt, SystemExit):

logger.error("Bot stopped!")

Файл Dockerfile

FROM python:3.9-buster

WORKDIR /usr/src/app/tg\_bot

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD [ "python", "bot.py" ]

Файл deployments/Chart.yaml

apiVersion: v2

name: tgbot-service

description: A Helm chart for Kubernetes

# A chart can be either an 'application' or a 'library' chart.

#

# Application charts are a collection of templates that can be packaged into versioned archives

# to be deployed.

#

# Library charts provide useful utilities or functions for the chart developer. They're included as

# a dependency of application charts to inject those utilities and functions into the rendering

# pipeline. Library charts do not define any templates and therefore cannot be deployed.

type: application

# This is the chart version. This version number should be incremented each time you make changes

# to the chart and its templates, including the app version.

# Versions are expected to follow Semantic Versioning (https://semver.org/)

version: 0.1.0

# This is the version number of the application being deployed. This version number should be

# incremented each time you make changes to the application. Versions are not expected to

# follow Semantic Versioning. They should reflect the version the application is using.

# It is recommended to use it with quotes.

appVersion: "1.0.0"

Файл deployments/values.yaml

replicaCount: 1

image:

repository: registry.gitlab.com/trackerbot/pcpl\_lab5

pullPolicy: IfNotPresent

pullSecretName: trackerbot-cred

tag: "latest"

service:

secret\_name: tgbot-secret

Файл deployments/template/deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: {{ include "tgbot-service.fullname" . }}

namespace: {{ .Release.Namespace }}

labels:

{{- include "tgbot-service.labels" . | nindent 4 }}

spec:

replicas: {{ .Values.replicaCount }}

selector:

matchLabels:

app.kubernetes.io/name: {{ include "tgbot-service.name" . }}

app.kubernetes.io/instance: {{ .Release.Name }}

template:

metadata:

labels:

app.kubernetes.io/name: {{ include "tgbot-service.name" . }}

app.kubernetes.io/instance: {{ .Release.Name }}

spec:

containers:

- name: {{ .Chart.Name }}

image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .Chart.AppVersion }}"

imagePullPolicy: {{ .Values.image.pullPolicy }}

env:

- name: BOT\_TOKEN

valueFrom:

secretKeyRef:

name: {{ .Values.service.secret\_name }}

key: bot\_token

imagePullSecrets:

- name: {{ .Values.image.pullSecretName }}

Файл tgbot/handlers/bowl.py

from aiogram import types, Dispatcher

from aiogram.dispatcher.filters import Text

from tgbot.keyboards.reply import bowl\_button\_text

from tgbot.utils.process\_result import process\_result

async def bowl(message: types.Message):

result = await message.answer\_dice('🎳')

await process\_result(message, result.dice.value)

def register\_soccer(dp: Dispatcher):

dp.register\_message\_handler(bowl, Text(equals=bowl\_button\_text))

Файл tgbot/handlers/dart.py

from aiogram import types, Dispatcher

from aiogram.dispatcher.filters import Text

from tgbot.keyboards.reply import dart\_button\_text

from tgbot.utils.process\_result import process\_result

async def dart(message: types.Message):

result = await message.answer\_dice('🎯')

await process\_result(message, result.dice.value)

def register\_dart(dp: Dispatcher):

dp.register\_message\_handler(dart, Text(equals=dart\_button\_text))

Файл tgbot/handlers/dice.py

from aiogram import types, Dispatcher

from aiogram.dispatcher.filters import Text

from tgbot.keyboards.reply import dice\_button\_text

from tgbot.utils.process\_result import process\_result

async def dice(message: types.Message):

result = await message.answer\_dice('🎲')

await process\_result(message, result.dice.value)

def register\_dice(dp: Dispatcher):

dp.register\_message\_handler(dice, Text(equals=dice\_button\_text))

Файл tgbot/handlers/user.py

from aiogram import Dispatcher

from aiogram.types import Message

from tgbot.keyboards.reply import menu\_keyboard

async def user\_start(message: Message):

await message.reply(

"Привет! Выбери что хочешь сделать с помощью меню",

reply\_markup=menu\_keyboard(),

)

def register\_user(dp: Dispatcher):

dp.register\_message\_handler(user\_start, commands=["start"], state="\*")

Файл tgbot/keyboards/reply.py

from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

dice\_button\_text = "Бросить кубик"

bowl\_button\_text = "Сыграть в боулинг"

dart\_button\_text = "Сыграть в дартс"

def menu\_keyboard() -> ReplyKeyboardMarkup:

return (

ReplyKeyboardMarkup(resize\_keyboard=True)

.row(

KeyboardButton(text=dice\_button\_text),

)

.row(

KeyboardButton(text=dart\_button\_text),

)

.row(

KeyboardButton(text=bowl\_button\_text),

)

)

Файл tgbot/utils/process\_result.py

import asyncio

from aiogram.types import Message

async def process\_result(message: Message, value: int):

if value == 6:

await asyncio.sleep(2)

await message.answer("Ты выиграл!")

Файл tgbot/config.py

from dataclasses import dataclass

from environs import Env

@dataclass

class Telegram:

token: str

@dataclass

class Config:

telegram: Telegram

def load\_config():

env = Env()

return Config(

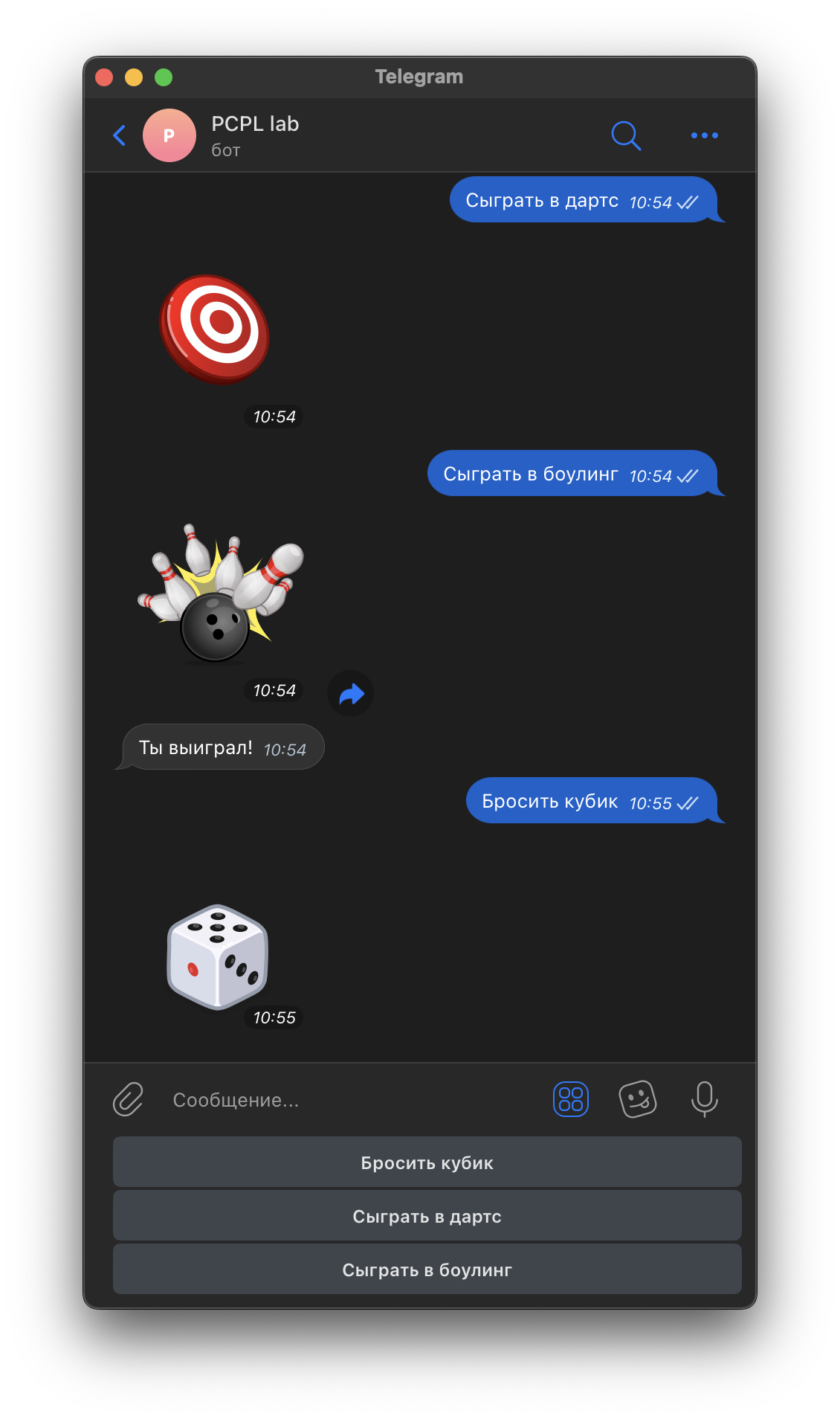
telegram=Telegram(

token=env.str("BOT\_TOKEN"),

),

)

**Пример выполнения программы**

****