**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 №4

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-35Б |  |  |
| Талаев А.П. |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г

**Задание:**

Телеграмм бот, отправляет актуальную информацию по криптовалюте через API CoinGecko.

**Выполнение:**  
from telegram import Update

from telegram.ext import ApplicationBuilder, CommandHandler, ContextTypes

import requests

import json

TG\_KEY = '7600099992:AAHX1q7jHmEmPHDaInHZTXSHwAwx2WGT9Hw'

API\_DEMO\_KEY = 'CG-zvKXVQbn9hzZ3Reh2ZC2v55v'

def get\_crypto\_data(crypto):

try:

response = requests.get(f'https://api.coingecko.com/api/v3/coins/markets?vs\_currency=usd&ids={crypto}&api\_key={keys.API\_DEMO\_KEY}')

response.raise\_for\_status()

data = response.json()[0]

return data

except requests.exceptions.RequestException as e:

print ("e^^rrr#ror:", e)

def get\_top\_cryptos():

try:

response = requests.get(f'https://api.coingecko.com/api/v3/coins/markets?vs\_currency=usd&order=market\_cap\_desc&per\_page=10&page=1&sparkline=false&api\_key={keys.API\_DEMO\_KEY}')

response.raise\_for\_status()

data = response.json()

return data

except requests.exceptions.RequestException as e:

print ("e^^rrr#ror:", e)

async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

welcome\_message = 'Hola amigo, используй /pleaseHELP чтобы узнать доступные команды;'

await update.message.reply\_text(welcome\_message)

async def data(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

crypto = update.message.text.split()[1]

data = get\_crypto\_data(crypto)

if data:

await update.message.reply\_text(f'The current price of {crypto} is ${data["current\_price"]}. The price change in the last 24 hours is {data["price\_change\_percentage\_24h"]}%.\nThe market cap is ${data["market\_cap"]}.\nThe total volume in the last 24 hours is ${data["total\_volume"]}.')

else:

await update.message.reply\_text('хз что это')

async def high\_low(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

crypto = update.message.text.split()[1]

data = get\_crypto\_data(crypto)

if data:

await update.message.reply\_text(f'The highest price in the last 24 hours for {crypto} is ${data["high\_24h"]}, and the lowest price is ${data["low\_24h"]}.')

else:

await update.message.reply\_text('хз что это')

async def supply(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

crypto = update.message.text.split()[1]

data = get\_crypto\_data(crypto)

if data:

await update.message.reply\_text(f'The circulating supply of {crypto} is {data["circulating\_supply"]}, and the total supply is {data["total\_supply"]}.')

else:

await update.message.reply\_text('хз что это')

async def top10(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

data = get\_top\_cryptos()

if data:

message = "Топ 10, 10 лучших криптовалют из первых 10 входящих топ 10, первые 10 из топ 100:\n"

for i, crypto in enumerate(data, start=1):

message += f'{i}) \*{crypto["name"]}\* #{crypto["symbol"].upper()}:\n- Price: ${crypto["current\_price"]}\n- Market capitalization is ${crypto["market\_cap"]}\n- Total volume in the last 24 hours is ${crypto["total\_volume"]}\n\n'

await update.message.reply\_text(message, parse\_mode='Markdown')

else:

await update.message.reply\_text('хз что это')

def check\_rank(query) -> None:

try:

url = "https://api.coingecko.com/api/v3/search"

params = {'query': query}

response = requests.get(url, params=params)

response.raise\_for\_status()

data = response.json()

return data

except requests.exceptions.RequestException as e:

print ("e^^rrr#ror:", e)

async def rank(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

data = check\_rank(update.message.text.split()[1:])

await update.message.reply\_text(f'Coin rank: \*{data["coins"][0]['market\_cap\_rank']}\*', parse\_mode='Markdown')

print(data["coins"][0]['market\_cap\_rank'])

async def help(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

await update.message.reply\_text("/data <crypto name> - Получает текущие данные для конкретной криптовалюты;\n/supply <crypto name> - Получает информацию о циркулирующем предложении и общем объеме определенной криптовалюты;\n/high\_low <crypto name> - Получает информацию о самой высокой и самой низкой цене конкретной криптовалюты за последние 24 часа;\n/top10 - топ 10 криптовалют по объему торгов, капитализации и т.д.;\n/rank <crypto name>- рейтинг валюты;")

app = ApplicationBuilder().token(keys.TG\_KEY).build()

app.add\_handler(CommandHandler("start", start))

app.add\_handler(CommandHandler("data", data))

app.add\_handler(CommandHandler("high\_low", high\_low))

app.add\_handler(CommandHandler("supply", supply))

app.add\_handler(CommandHandler("top10", top10))

app.add\_handler(CommandHandler("rank", rank))

app.add\_handler(CommandHandler("pleaseHELP", help))

app.run\_polling()

**Результаты:**

