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 домашнему заданию.

Выполнил:

студент группы ИУ5-33Б

Иванов Николай

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Подпись и дата:

Москва, 2023 г.

Общее описание проекта.

Проект, реализованный в качестве домашнего задания по дисциплине, представляет собой разработанного на языке Python Telegram – бота с использованием библиотеки Aiogram, основной темой и идеей которого является работа с различными валютами, как национальными, так и криптографической. Бот был разработан в команде со студентом ИУ5-33Б Валерием Нагапетяном.

Тема и актуальность проекта.

Данная тема была выбрана в связи с высокой ролью операций над различными валютами в жизни человека и необходимостью поддерживать актуальный уровень информированности о происходящих на финансовом рынке процессах и изменениях, а также общим ростом популярности различных криптографических валют.

Функционал проекта.

Созданный Telegram – бот позволяет производить конвертацию любых существующих и следить за их наиболее ценными параметрами криптографических валют.

Выбор языка программирования.

Для создания проекта был выбран язык Python с целью получения возможностей изучения создания Telegram – ботов, работы асинхронных функций и взаимодействия с API.

Код проекта.

Bot.py

```
import asyncio
import logging

from aiogram import Bot, Dispatcher, types, F
from aiogram.filters import Command
from aiogram.fsm.context import FSMContext
from configs.config import configuration
from keyboards import keyboards
from FSM.StateMachine import ExchangeCurrency, InfoCurrency
from FSM.StateMachine import Menu
from api.api import api_crypto_exchange, api_crypto_info
from commands import set_commands

logging.basicConfig(filename="../static/logger.txt", level=logging.INFO)
dp = Dispatcher()
bot = Bot(token=configuration.BOT_TOKEN.get_secret_value(), parse_mode="html")


@dp.message(Command("start"))
async def cmd_start(message: types.Message, state: FSMContext):
    await message.answer(f"Hello, {message.from_user.username}! Welcome to
NIVACryptoBot 📈\n\n"
                         f"Choose an option on your keyboard 📲",
                         reply_markup=keyboards.main_keyboard)
    await state.set_state(Menu.option)


@dp.message(F.text.lower().in_(['back']))
async def back(message: types.Message, state: FSMContext, change_flag:
list[bool]):
    await state.clear()
    if change_flag[0]:
        change_flag[0] = False
    await state.set_state(Menu.option)
    return await message.answer(text='Back to Menu',
reply_markup=keyboards.main_keyboard)


# @dp.message(Menu.option, F.text.in_(Menu.menu))
@dp.message(Menu.option, F.text.in_(Menu.menu))
async def menu_option(message: types.Message, state: FSMContext):
    await state.update_data(option=message.text)
    # await message.answer("Choose a <i>base currency</i> from the following
list :)",
    #                      reply_markup=keyboards.currency_exchange_keyboard())
    if message.text.lower() == "currency exchange prices" or
message.text.lower() == "/exchange_prices":
        await message.answer("Choose a <i>base currency</i> on your
keyboard.\n\n"
                             "Can't find what you need? Enter a symbol of any
currency from the following <a
href='https://coinmarketcap.com/api/documentation/v1/#section/Standards-and-
Conventions'>list</a>!",
```

```
reply_markup=keyboards.currency_exchange_keyboard())
        await state.set_state(ExchangeCurrency.base_currency)

    elif message.text.lower() == "cryptocurrency info" or message.text.lower()
== "/cryptocurrency_info":
        await message.answer("Please specify a valid crypto symbol, for example,
btc or eth.")
        await state.set_state(InfoCurrency.info_currency)


@dp.message(ExchangeCurrency.base_currency,
F.text.upper().in_(keyboards.all_currencies))
async def exchange_target_currency(message: types.Message, state: FSMContext,
change_flag: list[bool]):
    await state.update_data(chosen_base_currency=message.text.upper())
    exchange = await state.get_data()
    if not change_flag[0]:
        await message.reply(f"You've chosen
<b>{exchange['chosen_base_currency']}</b> as base currency.\n\n"
                            f"Now choose a <i>target currency</i> on your
keyboard or from the following <a
href='https://coinmarketcap.com/api/documentation/v1/#section/Standards-and-
Conventions'>list</a>!",
                            reply_markup=keyboards.currency_exchange_keyboard())
        await state.set_state(ExchangeCurrency.target_currency)
    else:
        await message.reply('Set the currency amount for converting')
        await state.set_state(ExchangeCurrency.amount)


@dp.message(ExchangeCurrency.target_currency,
F.text.upper().in_(keyboards.all_currencies))
async def exchange_procedure(message: types.Message, state: FSMContext):
    await state.update_data(chosen_target_currency=message.text.upper())
    exchange = await state.get_data()
    base_currency = exchange['chosen_base_currency']
    target_currency = exchange['chosen_target_currency']
    await message.reply(
        f"You've chosen <b>{base_currency}</b> as base currency"
        f" and <b>{target_currency}</b> as target currency.")
    await message.answer(text='Please set the currency amount for converting')
    await state.set_state(ExchangeCurrency.amount)


@dp.message(ExchangeCurrency.amount)
async def currency_amount(message: types.Message, state: FSMContext,
change_flag: list[bool]):
    await state.update_data(amount=message.text)
    exchange = await state.get_data()
    amount_for_converse = exchange['amount']
    if message.text.isdigit():
        base_currency = exchange['chosen_base_currency']
        target_currency = exchange['chosen_target_currency']
        # await message.answer(
        #     f'You are going to converse <b>{amount_for_converse}</b> units of
<b>{base_currency}</b> into'
        #     f' {target_currency}')
        parameters = {
            "amount": int(amount_for_converse),
            "symbol": base_currency,
            "convert": target_currency
```

```
        }
        response = api_crypto_exchange(parameters)
        conversion =
round(response["data"][0]["quote"][target_currency]["price"], 2)
        await message.answer(
            f'You are going to converse <b>{float(amount_for_converse):,}</b>
units of <b>{base_currency}</b> into'
            f' {target_currency}\n\n'f"{float(amount_for_converse):,}
<b>{base_currency}</b> equals {conversion:,} <b>{target_currency}</b>",
            reply_markup=keyboards.currency_exchange_keyboard_expanded())
        change_flag[0] = False
        return await state.set_state(ExchangeCurrency.next_step)

    else:
        change_flag[0] = False
        await message.answer('Wrong data. Please try again')
        return await state.set_state(ExchangeCurrency.amount)


@dp.message(ExchangeCurrency.next_step)
async def next_step(message: types.Message, state: FSMContext, change_flag:
list[bool]):
    await state.update_data(step=message.text)
    if message.text.lower() == "change base currency":
        change_flag[0] = True
        await state.set_state(ExchangeCurrency.base_currency)
        return await message.answer(text='Choose a new base currency',

reply_markup=keyboards.currency_exchange_keyboard())
    if message.text.lower() == 'change target currency':
        change_flag[0] = True
        await state.set_state(ExchangeCurrency.target_currency)
        return await message.answer(text='Choose a new target currency',

reply_markup=keyboards.currency_exchange_keyboard())


@dp.message(InfoCurrency.info_currency)
async def info_currency(message: types.Message, state: FSMContext):
    await state.update_data(set_currency=message.text)
    info = await state.get_data()
    currency = info["set_currency"].upper()
    response = api_crypto_info(currency)

    keyboards.currency_info_array[0][0].url =
f"https://coinmarketcap.com/currencies/{response['Name'].lower()}/#Chart"
    keyboards.currency_info_array[0][1].url =
f"https://coinmarketcap.com/currencies/{response['Name'].lower()}/#News"
    keyboards.currency_info_array[1][0].url =
f"https://coinmarketcap.com/currencies/{response['Name'].lower()}/#Markets"
    keyboards.currency_info_array[1][1].url =
f"https://coinmarketcap.com/currencies/{response['Name'].lower()}/#Analytics"

    await message.reply(text=f"<b>{response['Name']}
({response['Symbol'].upper()})</b>\n"
                            f"<b>Price</b>: {response['Price']}\n"
                            f"<b>1hr Change</b>: {response['1hr Change']}\n"
                            f"<b>24hr Change</b>: {response['24hr Change']}\n"
                            f"<b>7d Change</b>: {response['7d Change']}\n"
                            f"<b>Volume</b>: {response['Volume']}\n"
                            f"<b>Market Cap</b>: {response['Market Cap']}\n"
```

```
                                f"<b>Circulating Supply</b>: {response['Circulating
Supply']}\n"
                                f"<b>Total Supply</b>: {response['Total
Supply']}\n\n"
                                f"🔗<a
href='https://coinmarketcap.com/currencies/{response['Name'].lower()}/#Chart'>Vi
ew on CoinMarketCap</a>🔗",
                            reply_markup=keyboards.currency_info_keyboard)
    await state.set_state(Menu.option)
    # await message.answer(reply_markup=keyboards.currency_info_menu)


@dp.message()
async def wrong_input(message: types.Message):
    await message.answer('Wrong data')


async def main():
    await bot.delete_webhook(drop_pending_updates=True)
    await dp.start_polling(bot, change_flag=[False], on_startup=set_commands)


if __name__ == "__main__":
    asyncio.run(main())
```

commands.py

```
from aiogram import Bot
from aiogram.types import BotCommand, BotCommandScopeDefault


async def set_commands(bot: Bot):
    commands = [
        BotCommand(
            command='start',
            description='start chat with bot'
        ),
        BotCommand(
            command='exchange_prices',
            description='Currency exchange prices'
        ),
        BotCommand(
            command='cryptocurrency_info',
            description='Cryptocurrency info'
        )
    ]

    await bot.set_my_commands(commands, BotCommandScopeDefault())
```

api.py

```
import json
from typing import Dict

from requests import Request, Session
from configs.config import configuration
import pprint
```

```
api_key_coin = configuration.API_KEY_COIN.get_secret_value()
api_key_crypto = configuration.API_KEY_CRYPTO.get_secret_value()

headers = {
    'Accepts': 'application/json',
    'X-CMC_PRO_API_KEY': api_key_crypto
}


def api_coin(api_key: str):
    base_currency, target_currency = map(
        str, input("Введите основную и целевую валюты: ").split()
    )
    amount = int(input("Введите номинал: "))
    url =
f"https://api.freecurrencyapi.com/v1/latest?apikey={api_key}&base_currency={base
_currency}&currencies={target_currency}"

    response = Request("GET", url)

    info = response.json()

    if info["data"] and info["data"][target_currency]:
        converted = amount * info["data"][target_currency]
        print(f"{amount} {base_currency} составляют {converted}
{target_currency}")
    else:
        print("Указанная валюта не найдена")


def api_crypto_exchange(parameters: Dict):
    url = "https://pro-api.coinmarketcap.com/v2/tools/price-conversion"

    session = Session()
    session.headers.update(headers)
    response = session.get(url, params=parameters)

    return json.loads(response.text)


def api_crypto_info(crypto_symbol: str):
    url = "https://pro-api.coinmarketcap.com/v1/cryptocurrency/listings/latest"

    parameters = {
        "start": 1,
        "limit": 5000,
        "convert": "USD"
    }

    session = Session()
    session.headers.update(headers)
    response = session.get(url, params=parameters)

    data = json.loads(response.text)["data"]

    target = dict()
    for coin in data:
        if coin["symbol"] == crypto_symbol:
            target = coin
            break
```

```
    info = {
        "Name": target["name"],
        "Symbol": target["symbol"],
        "Price": f'${round(target["quote"]["USD"]["price"], 5):,} USD',
        "1hr Change": f'{round(target["quote"]["USD"]["percent_change_1h"],
2)}%',
        "24hr Change": f'{round(target["quote"]["USD"]["percent_change_24h"],
2)}%',
        "7d Change": f'{round(target["quote"]["USD"]["percent_change_7d"],
2)}%',
        "Volume": f'${round(target["quote"]["USD"]["price"] *
target["total_supply"], 2):,}',
        "Market Cap": f'${round(target["quote"]["USD"]["market_cap"], 2):,}',
        "Circulating Supply": f'{round(target["circulating_supply"], 2):,}',
        "Total Supply": f'{round(target["total_supply"], 2):,}',
    }

    return info
```

configs.py

```
from pydantic_settings import BaseSettings, SettingsConfigDict
from pydantic import SecretStr


class Settings(BaseSettings):
    BOT_TOKEN: SecretStr
    API_KEY_COIN: SecretStr
    API_KEY_CRYPTO: SecretStr

    model_config = SettingsConfigDict(
        env_file="../static/.env", env_file_encoding="utf-8"
    )


configuration = Settings()
```

StateMachine.py

```
from aiogram.fsm.state import StatesGroup, State


class ExchangeCurrency(StatesGroup):
    base_currency: str = State()
    target_currency: str = State()
    amount: str = State()
    next_step: str = State()


class InfoCurrency(StatesGroup):
    info_currency: str = State()


class Menu(StatesGroup):
    option: str = State()
    menu = ["Currency exchange prices", "Cryptocurrency info",
"/exchange_prices", "/cryptocurrency_info"]
```

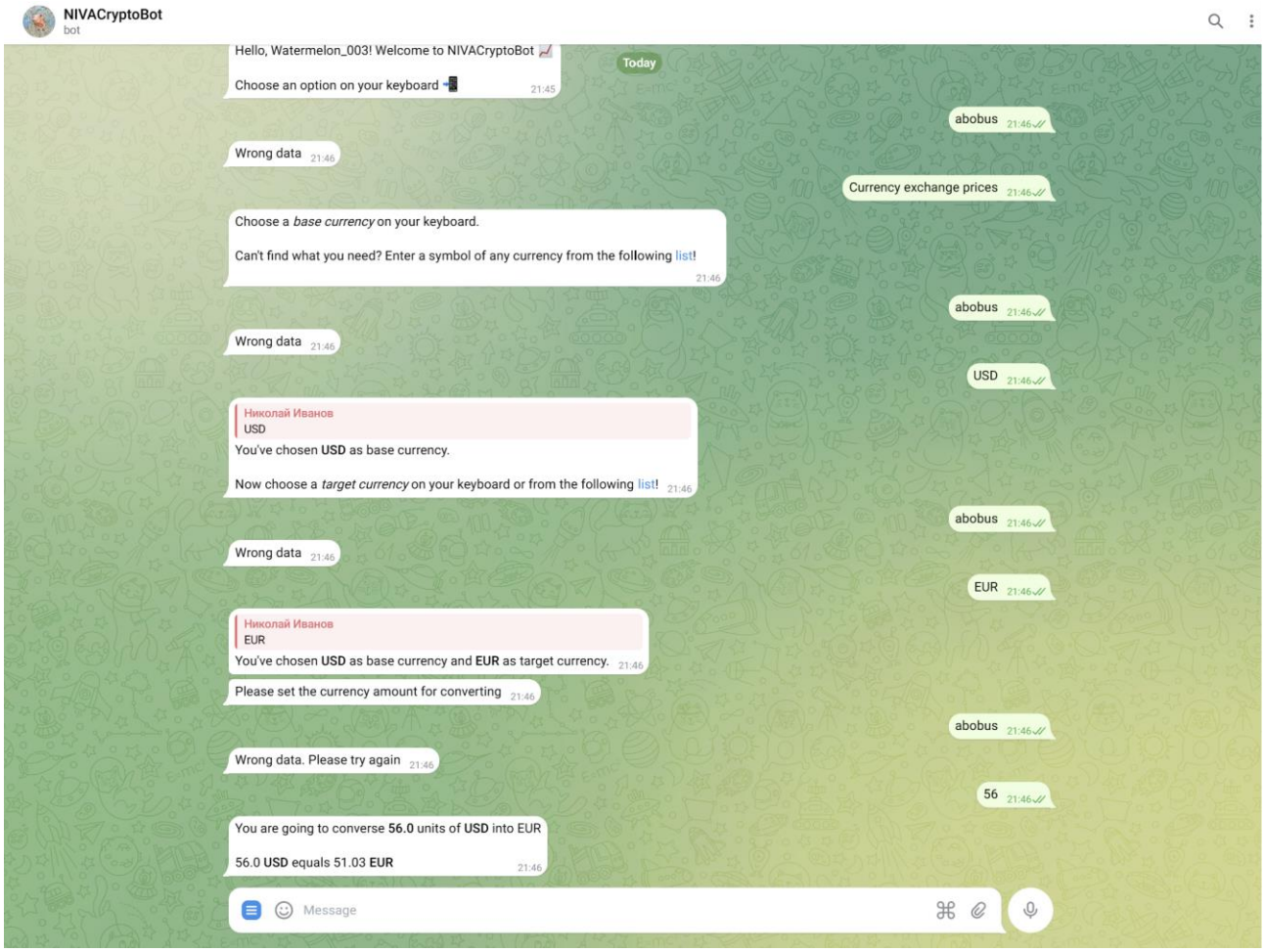
keyboards.py

```
import os

from aiogram.types import (
    InlineKeyboardMarkup,
    InlineKeyboardButton,
    ReplyKeyboardMarkup,
    KeyboardButton
)
from aiogram.utils.keyboard import ReplyKeyboardBuilder

main_keyboard = ReplyKeyboardMarkup(
    keyboard=[
        [
            KeyboardButton(text="Currency exchange prices"),
            KeyboardButton(text="Cryptocurrency info"),
        ]
    ],
    resize_keyboard=True,
    selective=True,
    one_time_keyboard=True,
    input_field_placeholder="Choose an option from menu..."
)

popular_currencies = [
    "USD", "EUR", "RUB",
    "CNY", "JPY", "QAR",
    "XAU", "XAG", "XPT"
]

currency_info_array = [
    [
        InlineKeyboardButton(text="Chart"),
        InlineKeyboardButton(text="News"),
    ],
    [
        InlineKeyboardButton(text="Markets"),
        InlineKeyboardButton(text="Analytics"),
    ]
]

currency_info_keyboard = InlineKeyboardMarkup(
    inline_keyboard=currency_info_array
)

currency_info_menu = ReplyKeyboardMarkup(
    keyboard=[
        [
            KeyboardButton(text="Change cryptocurrency"),
            KeyboardButton(text="Back"),
        ]
    ],
    resize_keyboard=True,
    selective=True,
    one_time_keyboard=True,
    input_field_placeholder="Choose an option from menu..."
)
```

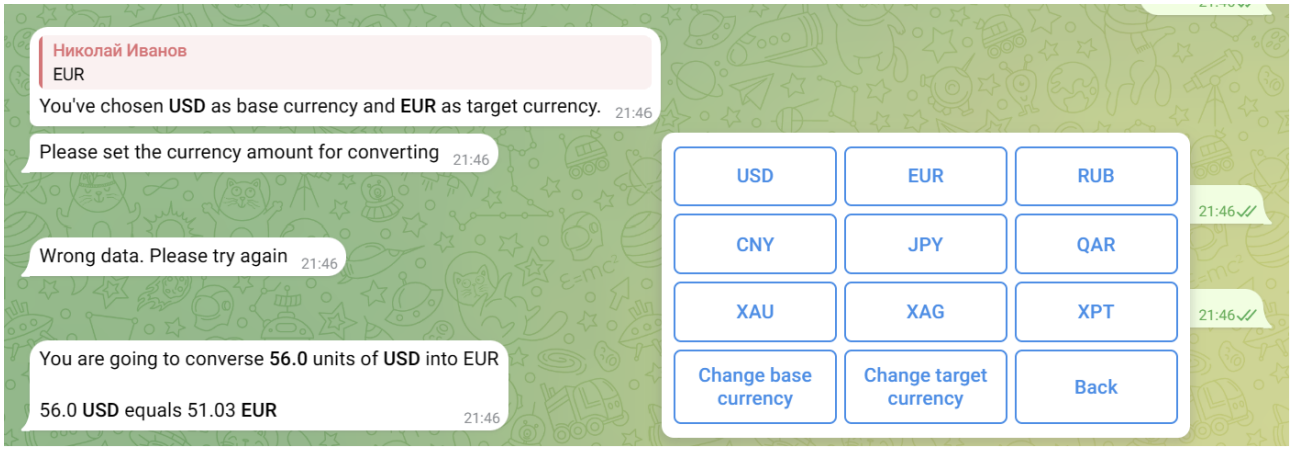
```
def get_currencies():
    array_of_currencies = []
    os.chdir(r"C:\Users\Asus\Desktop\papka\CryptoTelegramBot\keyboards")
    with open("currencies.txt", mode='r') as file:
        lines = file.readlines()
        for line in lines:
            currency = line.split(' ')[0]
            array_of_currencies.append(currency)
    return array_of_currencies




def currency_exchange_keyboard():
    keyboard = ReplyKeyboardBuilder()
    [keyboard.button(text=fiat) for fiat in popular_currencies]
    keyboard.button(text='Back')
    keyboard.adjust(*[3] * 3, 1)
    return keyboard.as_markup(resize_keyboard=True)


def currency_exchange_keyboard_expanded():
    keyboard = ReplyKeyboardBuilder()

    [keyboard.button(text=fiat) for fiat in popular_currencies]
    keyboard.button(text='Change base currency')
    keyboard.button(text='Change target currency')
    keyboard.button(text='Back')
    keyboard.adjust(*[3] * 4)
    return keyboard.as_markup(resize_keyboard=True)


all_currencies = get_currencies()
#print('MXN'.lower() in all_currencies)
```

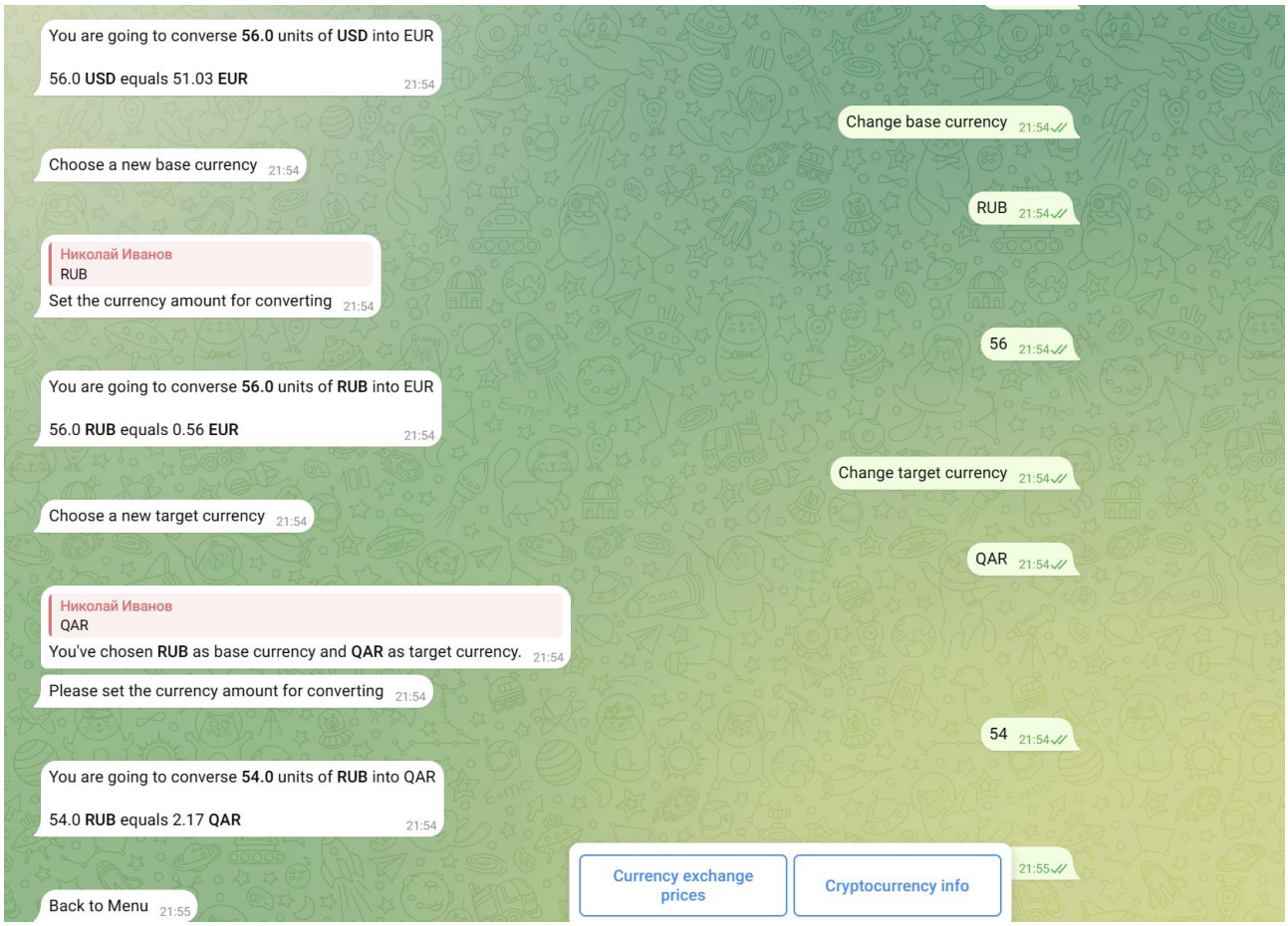
Демонстрация работы проекта.

Демонстрация функции конвертации валюты и проверки вводимых данных



Общий предлагаемый пользователю список валют. При отсутствии желаемой, пользователь может ввести свою. После конвертации пользователю предоставляется возможность смены какой – либо валюты, либо выход в главное меню.





Демонстрация функции получения актуальной информации о выбранной криптовалюте. После ее получения возможно перейти на сайт CoinMarketCap за более подробной информацией.

Ссылка на репозиторий.

vcreatorv/CryptoTelegramBot at Beta_bot (github.com)