**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: |  | Проверил: |
| студент группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Нагапетян Валерий |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | 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 сfor i in sorted(d,key = lambda x: int(d[x])):

print(i) различными валютами, как национальными, так и криптографической.

Бот был разработан в команде со студентом ИУ5-33Б Николаем Ивановым.

**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'>View 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

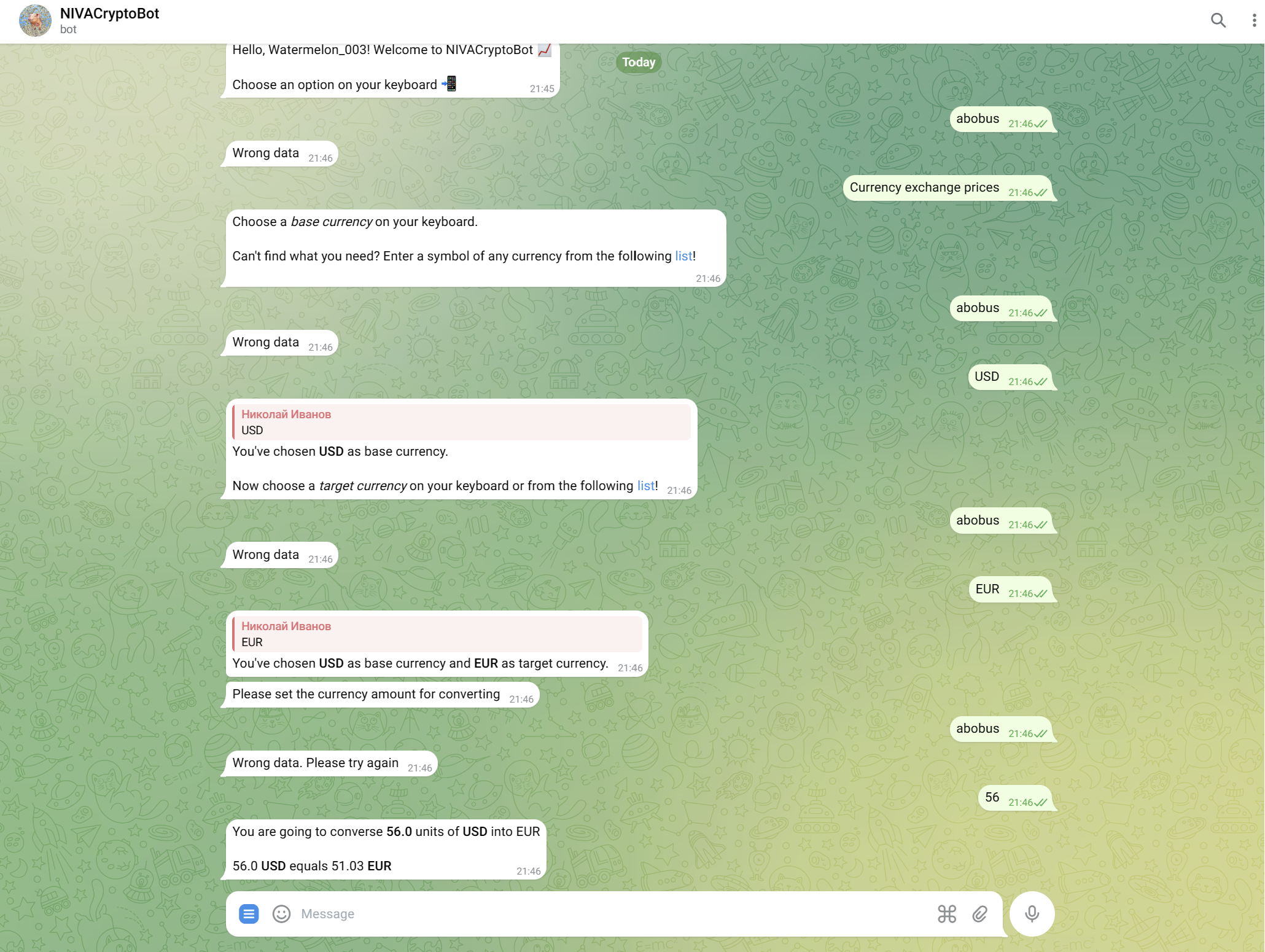
from aiogram.fsm.state import StatesGroup, State  
  
  
class ExchangeCurrency(StatesGroup):  
 base\_currency: str = State()  
 target\_currency: str = State()  
 amount: str = State()  
 next\_step: str = State()  
  
  
class InfoCurrency(StatesGroup):  
 info\_currency: str = State()  
  
  
class Menu(StatesGroup):  
 option: str = State()  
 menu = ["Currency exchange prices", "Cryptocurrency info", "/exchange\_prices", "/cryptocurrency\_info"]

keyboards.py

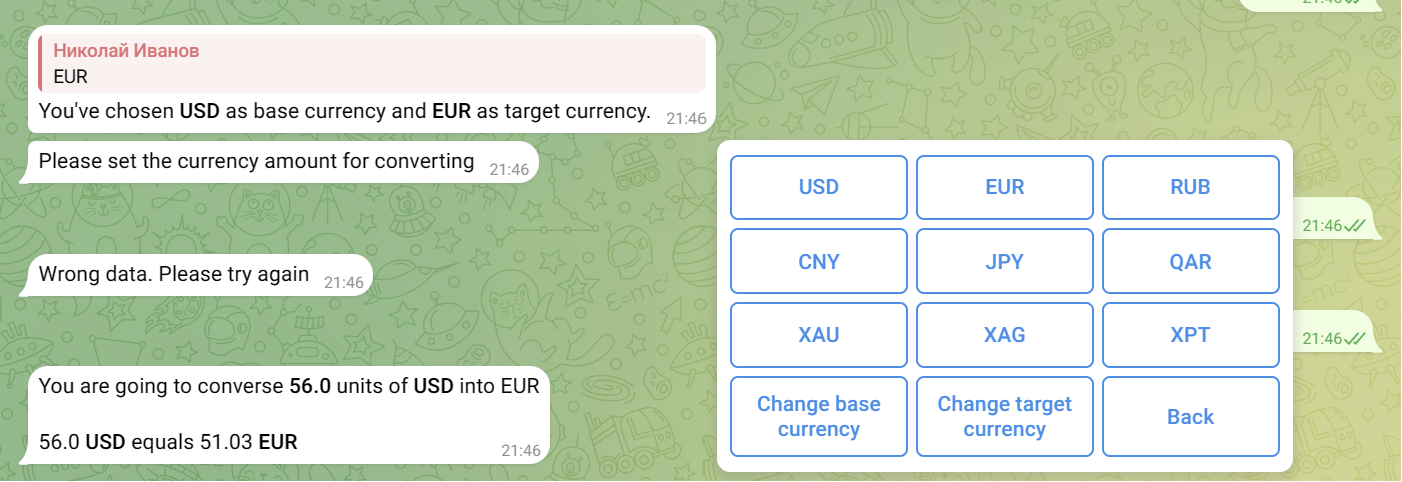
import os  
  
from aiogram.types import (  
 InlineKeyboardMarkup,  
 InlineKeyboardButton,  
 ReplyKeyboardMarkup,  
 KeyboardButton  
)  
from aiogram.utils.keyboard import ReplyKeyboardBuilder  
  
main\_keyboard = ReplyKeyboardMarkup(  
 keyboard=[  
 [  
 KeyboardButton(text="Currency exchange 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)

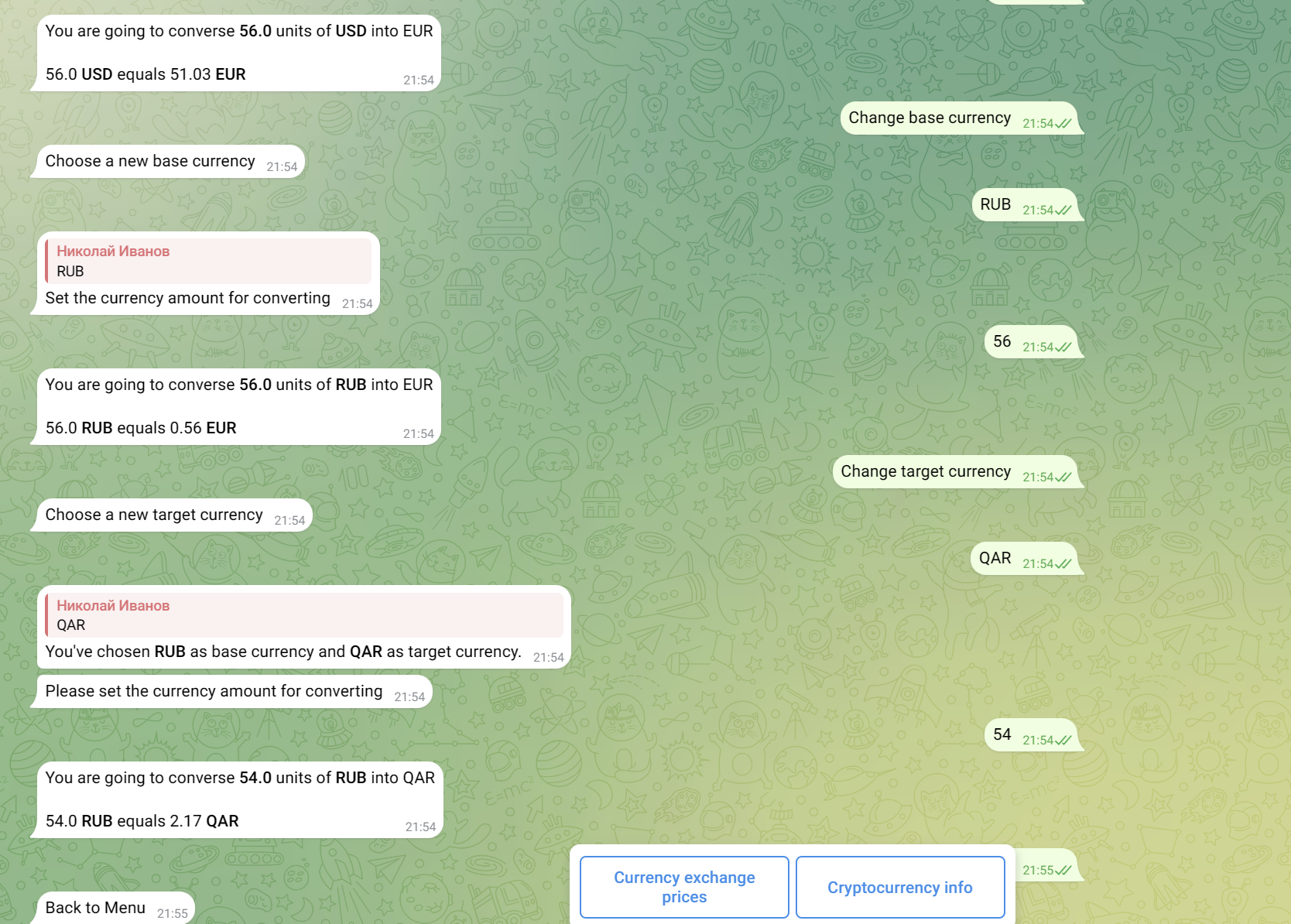
**Демонстрация работы проекта.**

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.

