Тестирование протокольных реализаций

Выполнили: Кандинский Алексей и Суханов Арсений.

Для реализации была выбрана модель Telegram-бота, который в определенном состоянии выдает определенную характеристику криптовалюты (в данном случае - цену и объем торгов на биржах для валют bitcoin и ethereum).

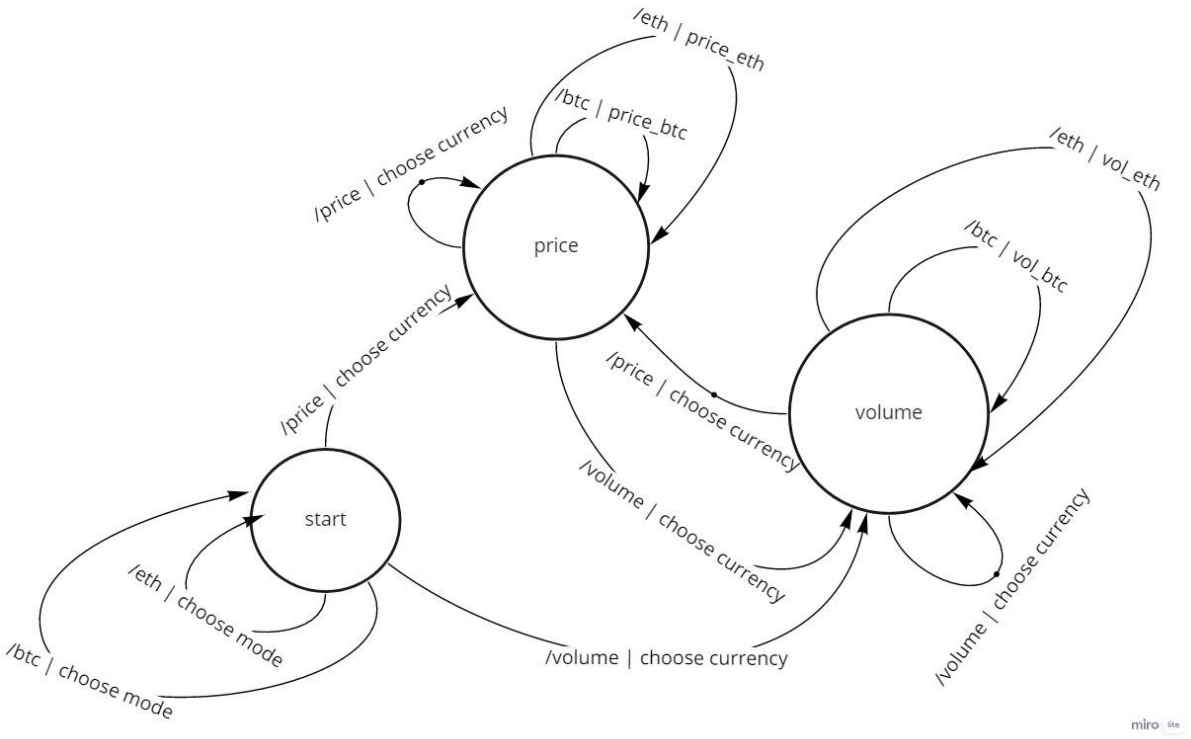

Рис 1. реализуемая модель.

Пояснения:

1. choose currency - сообщение пользователю с предложением выбрать нужную валюту (`“choose crypto currency name (/eth or /btc):”`);
2. choose mode - сообщение пользователю о том, что никакой режим не был выбран (`“no operating mode selected, please select a mode (/price or /volume):”`).

Код реализации вы можете посмотреть в конце отчёта.

Тест на основе формальной модели.

1. CC - choose currency;
2. PB - price_btc;
3. PE - price_eth;
4. VB - vol_btc;
5. VE - vol_eth;
6. MM - choose mode

```
F 0
s 3 START PRICE VOLUME
i 4 /price /volume /btc /eth
o 6 CC PB PE VB VE MM
n0 START
p 12
START /btc START MM
START /eth START MM
START /price PRICE CC
START /volume VOLUME CC
PRICE /btc PRICE PB
PRICE /eth PRICE PE
PRICE /price PRICE CC
PRICE /volume VOLUME CC
VOLUME /btc VOLUME VB
VOLUME /eth VOLUME VE
VOLUME /price PRICE CC
VOLUME /volume VOLUME CC
```

Тест построили при помощи http://fsmtestonline.ru/ и метода black box/W:

```
/btc/MM /btc/MM
/eth/MM /btc/MM
/price/CC /btc/PB /btc/PB
/price/CC /eth/PE /btc/PB
/price/CC /price/CC /btc/PB
/price/CC /volume/CC /btc/VB
/volume/CC /btc/VB /btc/VB
/volume/CC /eth/VE /btc/VB
/volume/CC /price/CC /btc/PB
/volume/CC /volume/CC /btc/VB
```

Были выполнены все тесты и реакции бота на входные последовательности совпали. Ниже приведены некоторые тесты и реакции бота на них.

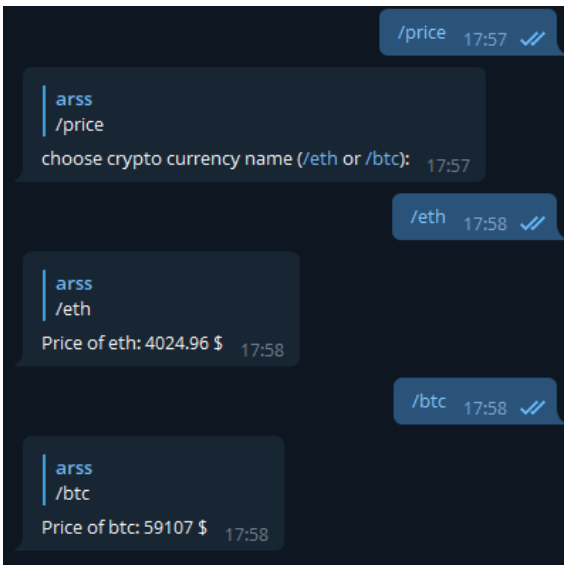

Рис.2 тест /price/CC /eth/PE /btc/PB.

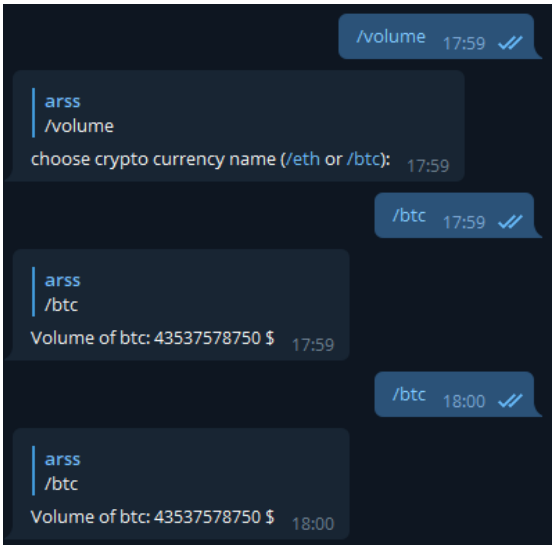

Рис. 3 тест /volume/CC /btc/VB /btc/VB.

Рис. 4 тест/volume/CC /price/CC /btc/PB.

Вывод - была построена формальная модель, реализован бот. На основе модели были построены тесты. В результате тестирования было выяснено, что бот полностью соответствует спецификации.

Код и пояснения к нему.
В файле TOKEN.py в переменной API_TOKEN хранится токен для доступа к боту. В файле tracker.py хранится логика для получения цены и объема торгов через coingecko api:

```
import requests
import json
import copy

class Tracker:
    def __init__(self, coins):
        url = ''
        for c in coins:
            url += c + '%2C'
        url = url[:-3]

        self.base_url =
f'https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd&ids={url}&or
der=market_cap_desc&per_page=100&page=1&sparkline=false'
        self.data = {}

        r = json.loads(requests.get(self.base_url).text)

        for i in r:
            self.data.update({i['symbol']:[i['current_price'],
i['total_volume']]})

    def get_data(self):
        save = copy.deepcopy(self.data)

        try:
            r = json.loads(requests.get(self.base_url).text)
            for i in r:
                self.data.update({i['symbol']:[i['current_price'],
i['total_volume']]})
            return self.data
        except:
            return save
```

В файле bot.py хранится логика бота:

```
import logging
from aiogram import Bot, Dispatcher, types
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.dispatcher.filters.state import State, StatesGroup
from aiogram.utils import executor
from tracker import Tracker
from TOKEN import API_TOKEN
```

```
logging.basicConfig(level=logging.INFO)

tr = Tracker(['ethereum', 'bitcoin'])

bot = Bot(token=API_TOKEN)

storage = MemoryStorage()
dp = Dispatcher(bot, storage=storage)

class StatesSet(StatesGroup):
    price = State()
    volume = State()

@dp.message_handler(state=None,commands=['btc', 'eth'])
async def cmd_not_mode(message: types.Message):
    await message.reply("no operating mode selected, please select a mode
(/price or /volume):")

@dp.message_handler(state='*',commands='price')
async def cmd_price(message: types.Message):
    await StatesSet.price.set()
    await message.reply("choose crypto currency name (/eth or /btc):")

@dp.message_handler(state='*',commands='volume')
async def cmd_volume(message: types.Message):
    await StatesSet.volume.set()
    await message.reply("choose crypto currency name (/eth or /btc):")

@dp.message_handler(state=StatesSet.price,commands='btc')
async def cmd_btcprice(message: types.Message):
    await message.reply(f"Price of btc: {tr.get_data()['btc'][0]} $")

@dp.message_handler(state=StatesSet.price,commands='eth')
async def cmd_ethprice(message: types.Message):
    await message.reply(f"Price of eth: {tr.get_data()['eth'][0]} $")

@dp.message_handler(state=StatesSet.volume,commands='btc')
async def cmd_btcvolume(message: types.Message):
    await message.reply(f"Volume of btc: {tr.get_data()['btc'][1]} $")

@dp.message_handler(state=StatesSet.volume,commands='eth')
async def cmd_ethvolume(message: types.Message):
    await message.reply(f"Volume of eth: {tr.get_data()['eth'][1]} $")

if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)
```