FRONT

Routes

Registration => registruj se

Login => uloguj se

Profile => pregled i izmena profila

Wallet => pregled stanja (**Account**), uplati sredstva (**Deposit**), posalji sredstva (**SendPayment**), razmeni sredstva (**Exchange**)

History => istorija deposit transakcija (**DepositTransactions**), istorija medju korisnickih transakcija (**UserToUserTransactions**), istorija exchange transakcija (**ExchangeTransactions**) (**Filter**, **Sort**?)

ExchangeRate => kursna lista

PUTANJE

FRONT: localhost:3000
BACK: localhost:5000

Iz perspektive back-a, kakve njegove metode treba da budu, sta da vracaju:

Login =>

method: POST

path: "/user/login"

payload: {email: string, password: string}

return: {user: object with all the user data}, OK / {}, 401

Logout =>

method: POST

path: "/user/logout"

payload: {email: string}

return: {}, OK / {}, 401

Verify =>

method: PUT

path: "/user/verify"

payload: {email: string, card_details: {card_number: int, card_name: string, card_expiration_date: string, card_security_code: int}}

return: {}, OK / {}, 401

Register =>

method: POST

path: "/user/register"

payload: {name, last_name, address, city, country, phone_number, email, password}

return: {}, OK

ChangeProfile =>

method: PUT
path: "/user/profile"
payload: {email: string, user: new user data}
return: {user: object with updated data}, OK

Wallet =>
method: GET
path: "/user/wallet"
payload: {email: string}
return: {wallet: six fields, same names as they are in model}, OK

Deposit =>
method: POST
path: "/transaction/deposit"
payload: {email: string, amount: double, card_details: {card_number: int, card_name: string, card_expiration_date: string, card_security_code: int}}
return: {}, OK /

SendPayment =>
method: POST
path: "/transaction/send"
payload: {email: string, email_sender: string, email_receiver: string, amount: double, currency: string (names from wallet model)}
return: {}, OK

Exchange =>
method: POST
path: "/transaction/exchange"
payload: {email: string, amount: double, currency_from: string (names from wallet model), currency_to: string (names from wallet model)}
return: {}, OK

History =>
helpingLinks: https://stackoverflow.com/questions/24892035/how-can-i-get-the-named-parameters-from-a-url-using-flask
method: GET
path: "/transaction/history/{transaction_type}?LegendForQuery"
payload: {email: string}
return: {transactions: [(e.g. json objects, each contains => fields from deposit: corresponding values)]}, OK

transaction_type == deposit || user-to-user || exchange
LegendForQuery:
fieldFromModel (one or more) =correspondingValue
sort_type ==naziv_polja_iz_modela
order_type == asc || desc

example == /transaction/history/user-to-user?email_reciver=someEmail&email_sender=someOtherEmail&sort_type=email&amount>500&sort_type=emailReceiver&order_type=asc

Fileds to use for query:

user-to-user => sender, receiver, currency, amount
exchange => to_currency, from_currency, to_amount, from_amount, user
deposit => user, amount

ExchangeRate =>

method: GET
path: "/crypto"
payload: {email: string}
return: e.g. *{'bitcoin': {'usd': 3461.27}, 'ethereum': {'usd': 106.92}, 'ripple': {'usd': 106.92}, 'tether': {'usd': 106.92}, 'dogecoin': {'usd': 106.92}}*, OK

NAPOMENE

* Koristiti socket.io da bi server obavestio klijenta kada se zavrsilo majnovanje.
Da bi back znao kada se zavrsilo majnovanje moze samo da pokrene jedan thread koji ce da ima timeout kojim ce se simulirati majnovanje i posle majnovanja da pozove od socket.io funkciju koja ce da obavesti servera.

* Back ne sme da ispunjava zahteve od nevalidiranog klijenta, treba da postoji zastita koja ce da onemoguci da nevalidirani klijent dobije uslugu jer na frontu ne moze da postoji adekvatna zastita za to.

* Da bi vratio data u flasku treba da uradis samo => return data, 200
Gde je 200 status code. Ima i komplikovanije ali ne znam da li je potrebno:

* Front rucno menja is_verified polje ako verifikacija vrati 200 OK

* "mining" should be the name of emit() first param

```
from flask import Flask, json

@app.route('/login', methods=['POST'])
def login():
    data = {"some_key":"some_value"} # Your data in JSON-serializable type
    response = app.response_class(response=json.dumps(data),
                                  status=200,
                                  mimetype='application/json')
    return response
```