Daxxer - Golang Challenge

OVERVIEW

Create a system that will be responsible for managing funds and assets of an account or digital wallet.

SPECIFICATIONS

1. “/deposit” endpoint should add funds to an account. The client will send the currency [btc, eth, ada, xrp or doge], amount and an account identificator (this can be a simple string with the username).

2. “/withdraw” endpoint removes funds from an account. The parameters are the same as the deposit endpoint.

3. “/balance” endpoint shows the current state of an user account. The little catch here is that you will have to use an API to get the current value in US dollars and euros. It could be any HTTP API you want. Just get the current value of any crypto the user has in his possession and convert it to US dollars and/or euros. You can use the cryptocurrency price with no more than 1 minute old.

4. “/history” list of all deposits and withdrawals a user made in a given period.

Example:

- /deposit is called with the following parameters:
    - currency: btc, amount: 0.2, user: ben
- /deposit is called with the following parameters:
    - currency: doge, amount: 1.8, user: ben
- /withdraw is called with the following parameters:
    - currency: doge, amount: 1.2, user: ben
- when /balance is called for the user ‘ben’ it should return:

currency	amount	price in dollars	price in euros	time of the rate used	total euros	total dollars
btc	0,2	46.285,90	39.438,60	08/09/2021 23:34	9257,18	7887,72
doge	0,6	0,25447	0,21942	08/09/2021 23:34	0,152682	0,131652
					9257,332682	7887,851652

REQUIREMENTS

- Use any library, framework, databases, external services you like to
- We took easier on the business rules to let you build a more “production-ready system” you can. If you take a shortcut or make a simplification to earn some time, be explicit.
- We like to see that you know how to test your code, so ...
- A README file briefly describing your solution, how to run (make it easy to deploy/run your services locally) and what else you consider important to document.

BONUS

- It isn’t required but we are curious about how many requests per second your endpoints can make
- We like to see an easy way to test if your endpoints are working, what are their parameters and returns. Please, help us
- We don’t like empty databases =)
- If something wrong happens, how does your system let everybody know it?
- How can we measure the latency of the external APIs calls?
- Extra functionality, be creative. Don’t be shy, let our CTO happy

DELIVERY

We think 1 week is an ok time to take the challenge, but we know that not everyone has the same level of availability. So let us know if you need more time.

Once you have done, send us a zip/tar file containing the git repository of the project (and please, don’t make it public).