Microservices Exercise

Thank you!

Task:

1) Create a REST API according to the functional requirements.

2) Use a programming language and relational database of your choice.

3) Responses should be all in JSON.

4) Provide an automated test suite for the app.

5) Use CoinDesk Bitcoin price API to source exchange rates. Please refer to the documentation at https://www.coindesk.com/coindesk-api

6) The final API will be tested using CURL, so you will need to include a read.me file in your project's root folder with examples of curls for testing and how to start your API.

7) Wrap up everything in Docker. Use several containers. Put them together using Docker compose.

8) Follow the Twelve-Factor App methodology

Functional requirements:

API allows its clients to manage BTC buy orders. Clients place orders to purchase BTC for fiat at a market price. API does not create transactions on the Bitcoin blockchain, but simply stores the order information in its database for further processing.

1) Create Buy Order

a. Creation of a Buy Order requires the following data at minimum:

* currency - represents the currency (ISO3 code one of: EUR, GBP, USD)
* amount - represents the amount of currency (0 < x < 1,000,000,000)

You can introduce additional data if that helps to fulfill the requirements.

b. Successful call should store the order in the database. The following info should be stored at a minimum:

* id - order's unique identifier
* amount - requested fiat amount
* currency - requested fiat currency
* exchange rate - value of BTC versus the requested fiat; BTC is the base currency and requested fiat is the quote currency; use the third-party API to source the exchange rates
* bitcoin amount - amount of BTC which the requested amount of fiat buys at the exchange rate. Use a precision of 8 decimal digits, and always round up. Do not lose precision in calculations

c. Buy Order creation must be idempotent.

d. Sum of bitcoin amount of all orders stored in the system must not exceed 100BTC. System must not allow creation of new orders which would cause the constraint to be violated.

2) Fetch Buy Order collection

Returns Buy Orders stored in the database in reverse chronological order. Resources must have the following attributes at minimum: id, amount, currency, exchange rate, bitcoin amount. Responses must be paged. Response time should be the same regardless which page is requested.

You can introduce additional data if that helps to fulfill the requirements.

TIPS:

* BE YOU! Use your coding style, code as you normally do.

* Don't be nervous if you find out you don't know a technology involved in this exercise enough to use it, just do what you normally do, google it and read about it and then implement what you learnt, but please don't copy and paste others code, without knowing what it does, as you may be asked to explain your code.

* It's better to have something working than nothing so try to stick to the requirements as much as possible, but if you can't, do what you need to finish the exercise.

* Last but not least, this is a useless API, it is probably just a trivial example of your capabilities, but it does not mean you need to be careless about how you code it.

This exercise aims to measure your skills and performance so "show yourself off", use your best code, experience, and knowledge. Code this as a world-class production-level software, but do not overcomplicate the resolution.

Good luck!