Go Lang developer test

DESCRIPTION:

Create a service that collect data from cryptocompare.com using its API and stores it in a database (MySQL or PostgreSQL)

Example API request: GET https://min-api.cryptocompare.com/data/pricemultifull?fsyms=BTC&tsyms=USD,EUR

RESPONSE STRUCTS:

Your service should serve data as described by these structs:

```
type raw struct {
	CHANGE24HOUR    float64 `json:"CHANGE24HOUR"`
	CHANGEPCT24HOUR float64 `json:"CHANGEPCT24HOUR"`
	OPEN24HOUR      float64 `json:"OPEN24HOUR"`
	VOLUME24HOUR    float64 `json:"VOLUME24HOUR"`
	VOLUME24HOURTO  float64 `json:"VOLUME24HOURTO"`
	LOW24HOUR       float64 `json:"LOW24HOUR"`
	HIGH24HOUR      float64 `json:"HIGH24HOUR"`
	PRICE           float64 `json:"PRICE"`
	LASTUPDATE      int64   `json:"LASTUPDATE"`
	SUPPLY          int64   `json:"SUPPLY"`
	MKTCAP          float64 `json:"MKTCAP"`
}
type display struct {
	CHANGE24HOUR    string `json:"CHANGE24HOUR"`
	CHANGEPCT24HOUR string `json:"CHANGEPCT24HOUR"`
	OPEN24HOUR      string `json:"OPEN24HOUR"`
	VOLUME24HOUR    string `json:"VOLUME24HOUR"`
	VOLUME24HOURTO  string `json:"VOLUME24HOURTO"`
	LOW24HOUR       string `json:"LOW24HOUR"`
	HIGH24HOUR      string `json:"HIGH24HOUR"`
	PRICE           string `json:"PRICE"`
```

```
	LASTUPDATE	string `json:"LASTUPDATE"`
	SUPPLY		string `json:"SUPPLY"`
	MKTCAP		string `json:"MKTCAP"`
}
```

REQUIREMENTS:

1. Currency pairs must be configurable.
2. Database parameters must be configurable.
3. Service must store data in database by a scheduler.
4. If cryptocompare's api is not accessible service must serve data from its database.
5. API should accept as many fsyms/tsyms in one request as possible (ex.: GET service/price?fsyms=BTC,LINK,MKR&tsyms=USD,EUR,ETH,LTC should return all pair prices)
6. Data in response must be fresh (realtime). 2-3 minutes discrepancy is ok.

ADDITIONAL POINTS:

1. Service scalabillity is a plus.
2. Following standard go project layout is a plus.
3. Websocket API for the service is a plus.
4. Using docker to build and run the service is a plus.
5. Nicely written README with clear instructions is a plus.

APPENDIX:

Example of HTTP request:

GET service/price?fsyms=BTC&tsyms=USD

Example of response:


```
{
"RAW": {
 "BTC": {
    "USD": {
        "CHANGE24HOUR": -13.25,
        "CHANGEPCT24HOUR": -0.18152873223073468,
        "OPEN24HOUR": 7299.12,
        "VOLUME24HOUR": 47600.120073200706,
        "VOLUME24HOURTO": 348033250.4911315,
        "LOW24HOUR": 7197.22,
```

```
        "HIGH24HOUR": 7426.64,
        "PRICE": 7285.87,
        "LASTUPDATE": 1586433196,
        "SUPPLY": 18313937,
        "MKTCAP": 133432964170.19
    }
 }
},
"DISPLAY": {
 "BTC": {
   "USD": {
     "CHANGE24HOUR": "$ -13.25",
     "CHANGEPCT24HOUR": "-0.18",
     "OPEN24HOUR": "$ 7,299.12",
     "VOLUME24HOUR": "Ƀ 47,600.1",
     "VOLUME24HOURTO": "$ 348,033,250.5",
     "HIGH24HOUR": "$ 7,426.64",
     "PRICE": "$ 7,285.87",
     "FROMSYMBOL": "Ƀ",
     "TOSYMBOL": "$",
     "LASTUPDATE": "Just now",
     "SUPPLY": "Ƀ 18,313,937.0",
     "MKTCAP": "$ 133.43 B"
    }
 }
}
}
```


Example of WS request:

WS service/price

```
{ "fsyms": "DASH", "tsyms": "RUR" }
```