GBM Software Engineer Challenge

How it works?

1. Analyze the challenge and create your own solution.
2. Share it with the GBM Team.
3. The GBM team will review it and if your solution is impressive, we will invite you to meet us.

Get ready!

1. Read the instructions and requirements for the challenge.
2. You have up to 7 days to submit your solution and keep in mind that we would like you to submit your best work and something you are really proud of.
3. Please use a local Git repository to version your changes and when you're finished send us your solution in a .zip file including the .git directory. Create as many commits as possible and, please, DO NOT push you code into any public repository.

Keep this in mind!

- There is no such thing as a perfect and complete solution. Do as much as you can, the best way possible.
- We will evaluate your skills and level based on:
  - How you translate business rules into code.
  - Your solution design.
  - Use of best practices (SOLID, DRY, KISS, and YAGNI).
  - Testing strategy.
- Your solution should be:
  - Scalable (able to handle millions of requests/processing per second).
  - Extensible (to easily plug a new interface on to it).
  - Testable (enable both unitary and regression tests).
- Include a README.md file with the instructions to setup the project (build and run). Keep in mind that an engineer will evaluate that your project meets all the requested requirements.

Scenario

A brokerage firm needs to have an API service to process a set of buy/sell orders.

This API must have two endpoints, the first one to create an account with the initial balance, and the second one to send the buy/sell orders.

For each order, the API expect to receive a timestamp (for when it took place), an operation type (buy or sell), issuer name (stock's identifier), a total amount of shares (a positive integer number), and the unitary price of the share (a positive real number).

Please make sure that all the following business rules are validated:

- Insufficient Balance: When buying stocks, you must have enough cash in order to fulfill it.
- Insufficient Stocks: When selling stocks, you must have enough stocks in order to fulfill it.
- Duplicated Operation: No operations for the same stock at the same amount must happen within a 5 minutes interval, as they are considered duplicates.
- Closed Market: All operations must happen between 6am and 3pm.
- Invalid Operation: Any other invalidity must be prevented.

A business rule violation is not consired an error, since in the real world they may happen. In case any happens, you must list them in the output as an array, and have no changes applied for that order, following to process the next order.

Required API endpoints

Create investment account

The first endpoint will be used to create an investment account. The expected contract is:

Request:

`POST /accounts`

Body:

```
{
  "cash": 1000
}
```

Expected response:

```
{
  "id": 1,
  "cash": 1000,
  "issuers": []
}
```

Send a buy/sell order

The second endpoint will be used to send orders into a specific account.

Request:

`POST /accounts/:id/orders`

Body:

```
{
  "timestamp": 1571325625,
  "operation": "BUY",
  "issuer_name": "AAPL",
  "total_shares": 2,
  "share_price": 50
}
```

Expected response:

```
{
  "current_balance": {
    "cash": 900,
    "issuers": [
      {
        "issuer_name": "AAPL",
        "total_shares": 2,
        "share_price": 50
      }
    ]
  },
  "business_errors": []
}
```

Multiple issuers are accepted, so given this other payload:

```
{
  "timestamp": 1583362645,
  "operation": "BUY",
  "issuer_name": "NFTX",
  "total_shares": 10,
  "share_price": 80
}
```

Expected respose:

```
{
  "current_balance": {
    "cash": 0,
    "issuers": [
      {
        "issuer_name": "AAPL",
        "total_shares": 2,
        "share_price": 50
      },
      {
        "issuer_name": "NFTX",
        "total_shares": 10,
        "share_price": 80
      }
    ]
  },
  "business_errors": []
}
```

If a business error occurs, the expected response should be:

```
{
  "current_balance": {
    "cash": 1000,
    "issuers": []
  },
  "business_errors": [
    "CLOSE_MARKET"
  ]
}
```

Use your imagination and creativity to Wow! us.
Think of cool and useful features or functionalities that you can add to this application.

Implement them as part of your solution to the challenge.