1. ***Installation:***

With the project on maven, select **Run As** -> **Maven Build…** then type:

"**package -DskipTests**" on "**goal**" field

The "jar" file need to be located on the "target" folder (default folder) and MUST be named

"**Quotation-Management-0.0.1-SNAPSHOT.jar**".

Before we put all the applications into containers, we just need to create the image from the project application (quotation-management) with the Dockerfile present on the project root file, to do that, we need point our bash to the application main file (where the Dockerfile is located).

After that, type the following command:

```
PWorkspace/Quotation-Management$ docker build --tag quotation-management .
```

The project contained in this GitHub has a "docker-compose.yml" file, it can be used to install all the necessary packages for the project to work. That .yml file will create 3 containers on docker: "mysql8", "stock-manager" and "quotation-management-app", they will all be inside a network called "mynetwork" and will be initialized to run properly.

Command to run the .yml file: (your bash must be pointing to the project root file)

```
oIDPWorkspace/Quotation-Management$ docker compose up
```

To access the project endpoints, you must access the localhost through the port 8081.

Documentation via Swagger found on localhost:8081/swagger-ui/

2. ***StartUp:***

During the first time loading the containers using the .yml file, the database (mysql8) container will take about 10 seconds to load, and the main application will crash because there is no database running. The main application is setted to restart every time it stops, so it will run smoothly on the second or third time, so do not worry.

3. ***EndPoints:***

3.1 GET localhost:8081/stock:

Will return a list with all Stocks present on the “stock-manager” API.

Response example:

When accessed, the response will be stored in cache. (the console will display a warn)

3.2 POST localhost:8081/stock:

Will receive the information to create a single new Stock on the “stock-manager” API using trim() and toLowerCase() on “id” to respect the pattern.

Request example:

When accessed, the cache from the “GET /stock” will be erased. (the console will display a warn)

3.3 GET localhost:8081/quotation:

Will return a list with all Quotations (with their quotes) present on the mysql8 database

Response example:

3.4 GET localhost:8081/quotation/{stockId}:

Will return the Quotation (with his quotes) with the stockId passed on the uri (if exists):

Response example:


```
[
    {
        "id": "949530b5-672b-49ef-aa02-6e03f4d79291",
        "stockId": "mag4",
        "quotes": {
            "2019-01-30": "27",
            "2019-01-29": "28",
            "2019-01-28": "21"
        }
    }
]
```


If the Id does not exist on the database, it will return a 404 not found error.

3.5 POST localhost:8081/quotation:

Will receive the information to create a single new Quotation with a list of quotes inside of it. The quotation will only be created if the stockId exists on the “stock-manager” database.


```
{
    "id": "949530b5-672b-49ef-aa02-6e03f4d79291",
    "stockId":"maG4",
    "quotes":
    {
        "2019-01-30": "27",
        "2019-01-29": "28",
        "2019-01-28": "21"
    }
}
```


If the quotation does not exist in the “stock-manager” database, it will return a 404 not found error.