1. Instalation

The following software are mandatory to be installed.

- Docker

  https://docs.docker.com/docker-for-windows/install/

- Java jdk

  https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

- Maven

  https://maven.apache.org/download.cgi

  The following software are optional and can be replaced by others for the same purpose.

- Eclipse

  https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2019-09/R/eclipse-jee-2019-09-R-win32-x86_64.zip

- Postman

  https://www.getpostman.com/downloads/

2. General information

Internet can be used with no restriction.

The project with all the source code produced must be delivered at the end of the test.

3. Configuration

The application to be developed will use two services that run on docker containers.

You will need two images to run the services below which also will be consumed by application to be developed (more information will come later).

- MySQL: https://hub.docker.com/_/mysql
- stock-manager: https://hub.docker.com/r/lucasvilela/stock-manager

The first service is a MySQL database service. To start it, run the following command line at Windows Power Shell.

```
docker container run -e MYSQL_ROOT_PASSWORD=root -e
MYSQL_DATABASE=bootdb -p 3306:3306 -p 33060:33060 -d mysql:8
```

The second service is a REST application that will be consumed by application to be developed (more information will come later). To start it, run the following command line at Windows Power Shell.

```
docker container run -p 8080:8080 -d lucasvilela/stock-manager
```

Any docker image can be stoped and re-started at any point in time. Any data stored on it is wiped on this process.

To stop a container, first issue [docker ps] to get container's ids.

Then, issue [docker stop {id}] to stop a specific container.

4. Quotation Management application

The following sections describes the needed information to create the application from scratch.

Each section contains a new requirement. It is recommended to implement the application in the sequence the sections are presented.

The application should preferably be developed using Maven.

Use the most suitable low level architecture for the given problem.

4.1 Store and read stock quotes

The application to be developed, called quotation-management, is a REST based application which purpose is to store stock quotes from stock market.

For example, a user wants to register that stock PETR4 quotation at 01/12/2018 was $35.00.

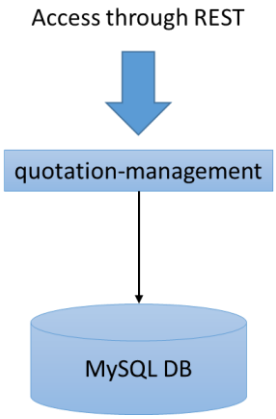
A user can register as many quotes as he/she wants for the same stock. Also, a user can register quotes from different stocks.

Finally, a user can read stock quotes registered.

To store and read stock quotes, application should expose REST APIs, through port 8081.

The stock quotes should be stored at MySQL DB running on docker container, described in previous section.

The following picture illustrates the application.



Quotation-management application should be developed using Spring Boot.

The payload for creating or reading a quote should follow the depicted example.

```
{
     "id": "c01cede4-cd45-11eb-b8bc-0242ac130003",
     "stockId": "petr3",
     "quotes":
     {
          "2019-01-01": "10",
          "2019-01-02": "11",
          "2019-01-03": "14"
     }
}
```

In total, the following operations should be exposed at quotation service:

- Create a Stock Quote
- Read a Stock Quote by stockId
- Read all Stock Quotes

4.2 Validating a stock before processing

A new service is introduced to the solution in which the application being developed is part of.

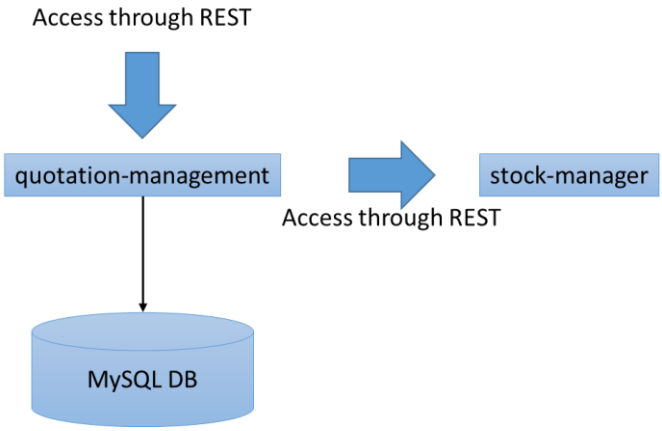
It service is called stock-manager. It is responsible for maintaining a list of stocks that can have quotes stored at quotation-management.

In other words, a user is allowed to create stock quotes on quotation-management only if the stock is registered at stock-manager.

Hence, quotation-management should be modified to verify, before creating a stock quote, if stock received is registered at stock-manager.

In case it is not, an error should be returned at create quotation REST API.

The following picture illustrates the application.



Stock-manager service runs on a docker container on port 8080, as described in previous section.

As mentioned in the picture, stock-manager service exposes REST APIs to fulfill its needs.

Any outside user can register stocks on stock-manager.

Quotation-management just reads registered stocks to validate the stock quote create operation.

- GET http://localhost:8080/stock

  Get the list of stocks registered. By default, when server is started, stocks PETR3 and VALE5 are already registered.

- POST http://localhost:8080/stock

```
{
    "id": "petr7",
    "description": "test petr"
}
```

  Register new stock.

4.3 Caching registered stock quotes

To avoid accessing stock-manger service at each create stock quote operation, quotation-management application should cache locally (in memory) the registered stocks.

When validating a stock, whenever the cache is empty, it should populate it from stock-manager service.

If cache is not empty, it should validate with cached data.

The cache should always be cleaned when a new stock is registered at stock manager.

To achieve this, two new functionalities should be introduced.

Quotation-management should register itself at stock-manager during startup. To register itself, the following stock-manager service must be invoked.

- POST http://localhost:8080/notification

The body of the service should follow the below example. It should contain the host and the port of the application being registered (in our case, quotation-management).

```
{
    "host": "localhost",
    "port": 8081
}
```

Whenever stock-manager receives a request to register a new stock, it tries to invoke a notification service at all registered applications.

Hence, quotation-management should expose a new REST service to accept notifications from stock-manager.

Stock-manager expects the following URL.

DELETE http://[host]:[port]/stockcache

Stock-manager replaces [host] and [port] by the proper data from registered applications.

Quotation-management, upon reception of this notification, should clean the cache.

4.4 Aditional requirements

Create on unit test for any developed class.

Create a docker image for quotation-management, so that it can run in a container.