**Assumption made while working:**

* Every product will have specific quantity which is nothing but stock. As someone will order the products stocks will go down and will be maintained as Item sold and same will be deducted from quantity.
* To support above point apart from asked end point URL some extra URL’s are added one such important URL is PATCH/stock/orderstocks which is responsible for reducing the quantity of stock and same increasing in item sold.
* While providing the statistics user will pass today and last month date in request as follows:

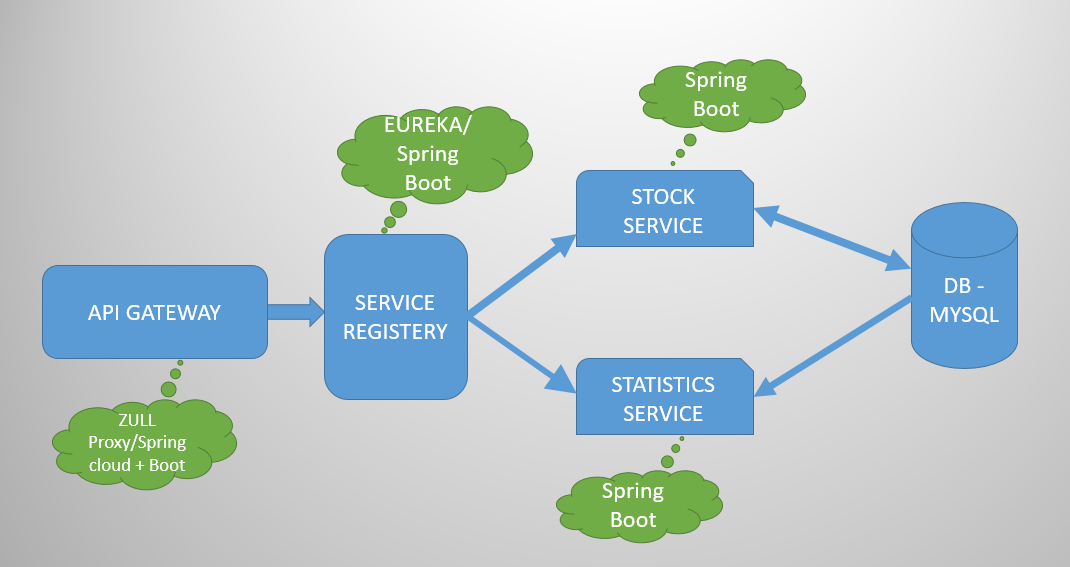
GET /statistics? Time=2019-06-12 22:30:56, 2019-06-01 22:30:56

* First date is today’s date which should not be after midnight. Second Date is current month date. User cannot enter date of last month. Ex. Range From 1St date of today’s current date of month is allowed. (This part was very hard to decide and confusing because of lack of clear requirements).
* Above described range is taken into consideration while showing Top Selling Products.
* While showing statistics for Top Available Product no range is taken into consideration.

**Project is divided into 3 parts as:**

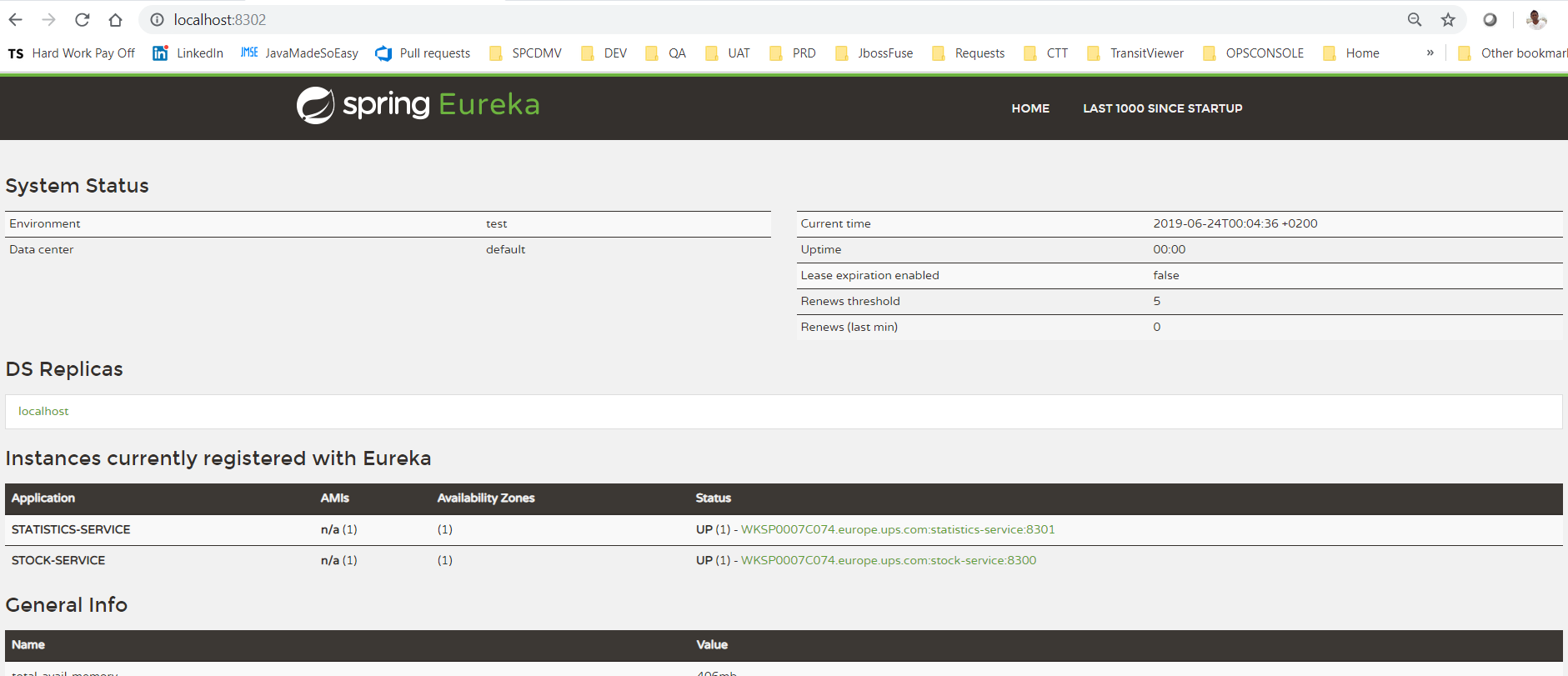
1. stock-service: Responsible for adding and updating the stocks for the product.
2. statistics-service: Responsible for providing statistics.
3. Eureka-service: Service Registry and API Gateway

**AS-IS Architecture Diagram:**

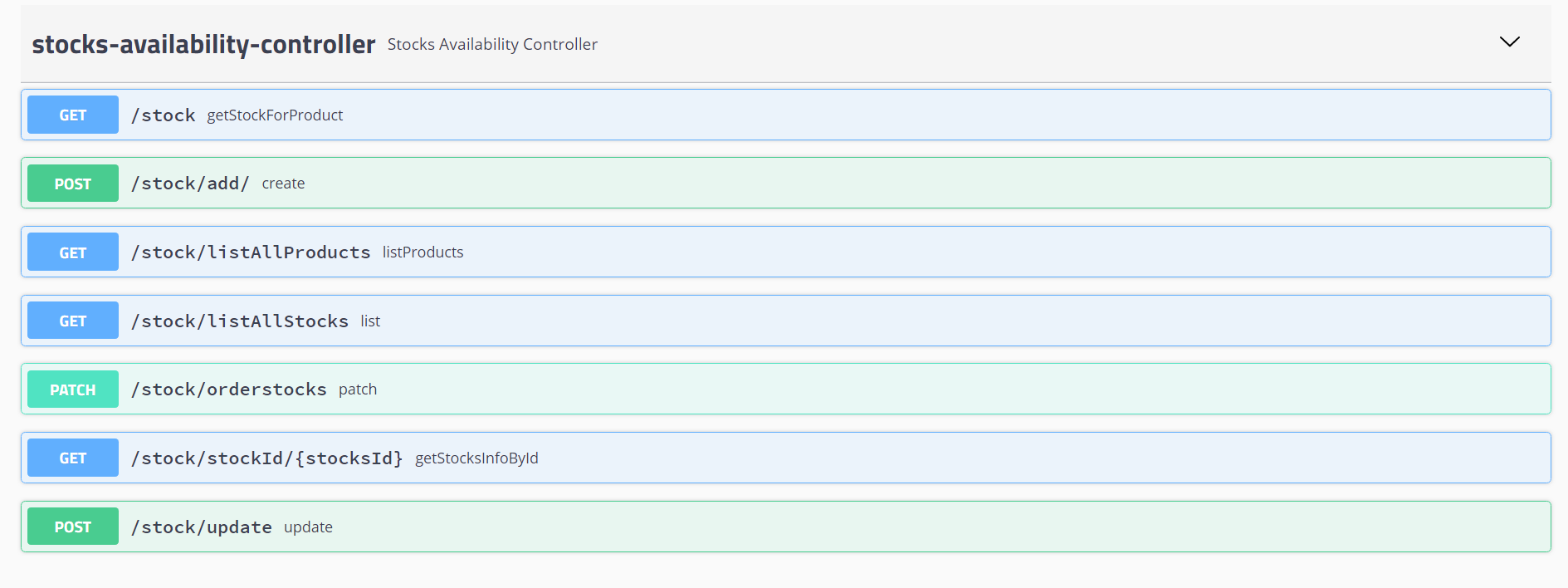


**IN IDEAL RUN TIME SCENRIO every micro service should stand alone and have their separate database and functionality independent of each other. Because of time constraint I was not able to granulize it also both the micro services here was needing same data. So first I developed Stock service and then I developed Statistics server and ended up using the same database**

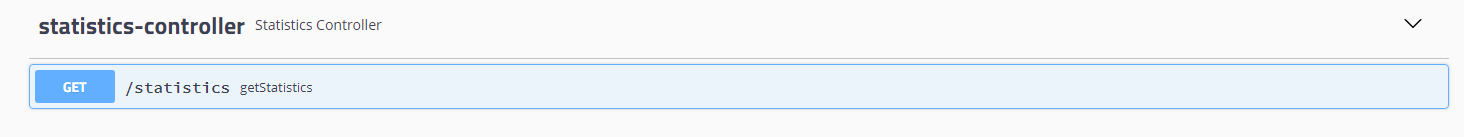
**STATISTICS SERVICE & STOCK SERVICE REGISTER IN EUREKA**



**Following URL Are Exposed In *stock-service* Project:**



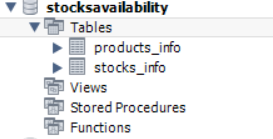
**Only one URL is exposed by Statistics Service:**



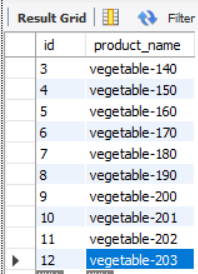
**It is found that there is ONE TO ONE Relationship between stocks and product. So 2 tables created prodcuts\_info & stock\_info as shown in below created.**

**prodcuts\_info table id is used as foreign key in stock\_info table as product\_id to maintain the relationship.**

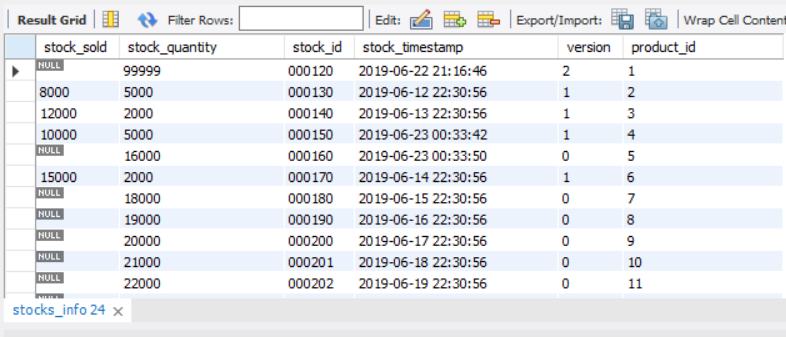
**Schema Name: stocksavailabiliy**



**product\_info:**



**stocks\_info:**



**How Concurrency Control is handled:**

For handling concurrency I have used **Optimistic concurrency control** mechanism which is preferred approach as REST HTTP is stateless.

Approach: Together with the data itself a versioning token is given to the client. When trying to modify the record the systems checks if this token is still valid, which means no change took place in the meantime. The change is only applied if no conflict is detected.

HTTP1.1 comes with an appropriate header field called *ETag*:  
“The ETag or entity tag is part of HTTP […]. ETags can also be used for optimistic concurrency control, as a way to help prevent simultaneous updates of a resource from overwriting each other.”

Accordingly if a REST client GETs a resource the back end service sends the versioning information in the response header field ETag:

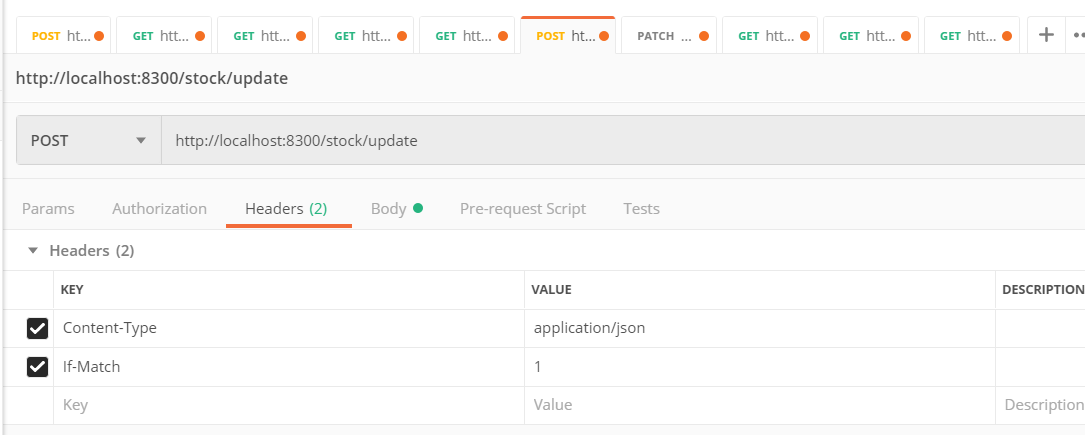
The ETag may contain a hash value of the resource’s content, a hash of the last modification timestamp or even just a revision number. The service consumer must cache the received versioning information and when calling a PUT operation add it to the request header field *If-Match*.

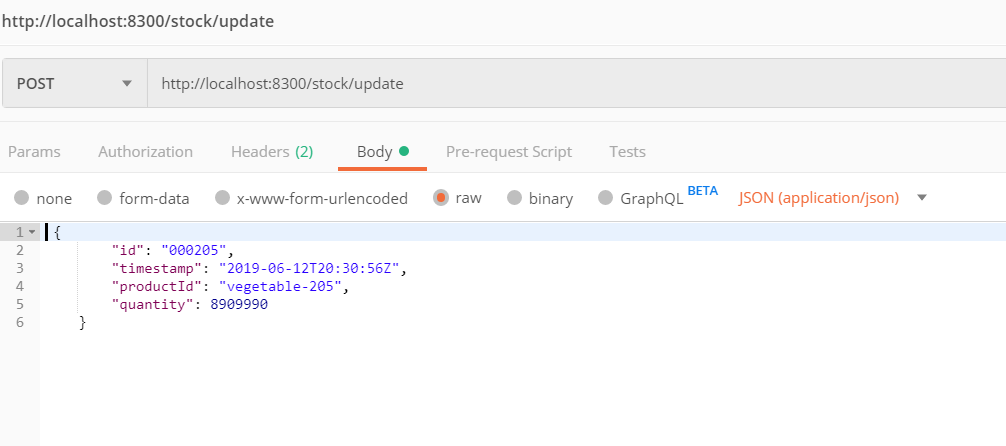
Basically the If-Match header makes the request conditional. In our case the back end service must check if the condition “resource was not modified” is fulfilled.

If the condition is satisfied, that means the update is possible, the HTTP status code*200 (OK)* with the updated resource as payload is sent (or alternatively *204 (No Content)*).  
If however the update cannot be applied because the resource was modified in the meantime the service will return the status code *412 (Precondition Failed)*. Receiving that status the client has to decide how to handle the conflict – by reloading the resource and discarding or merging the changes.

In JPA we have @Version attribute which can be used for managing a dedicated revision number

Example of If-Match Tag:

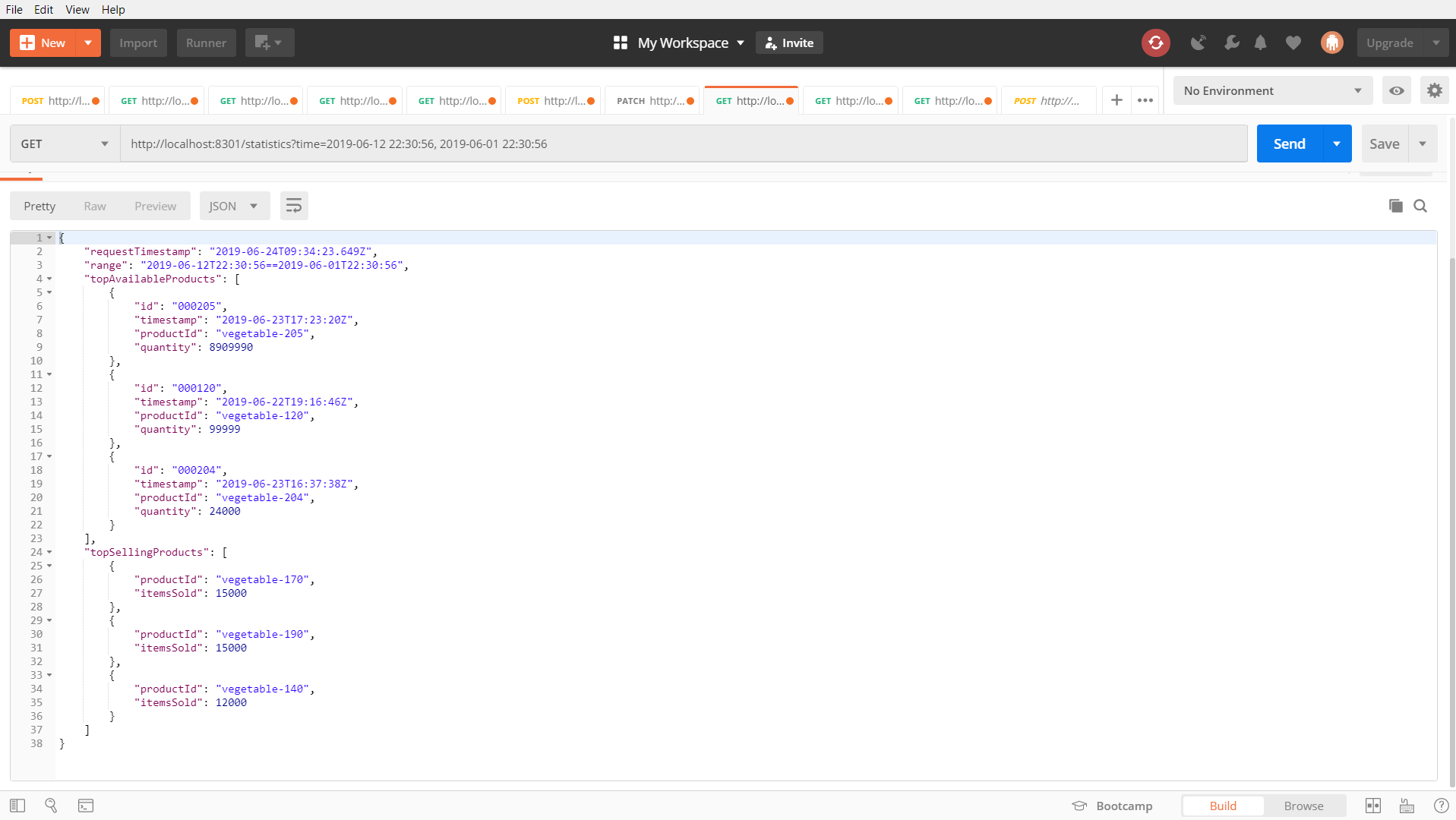




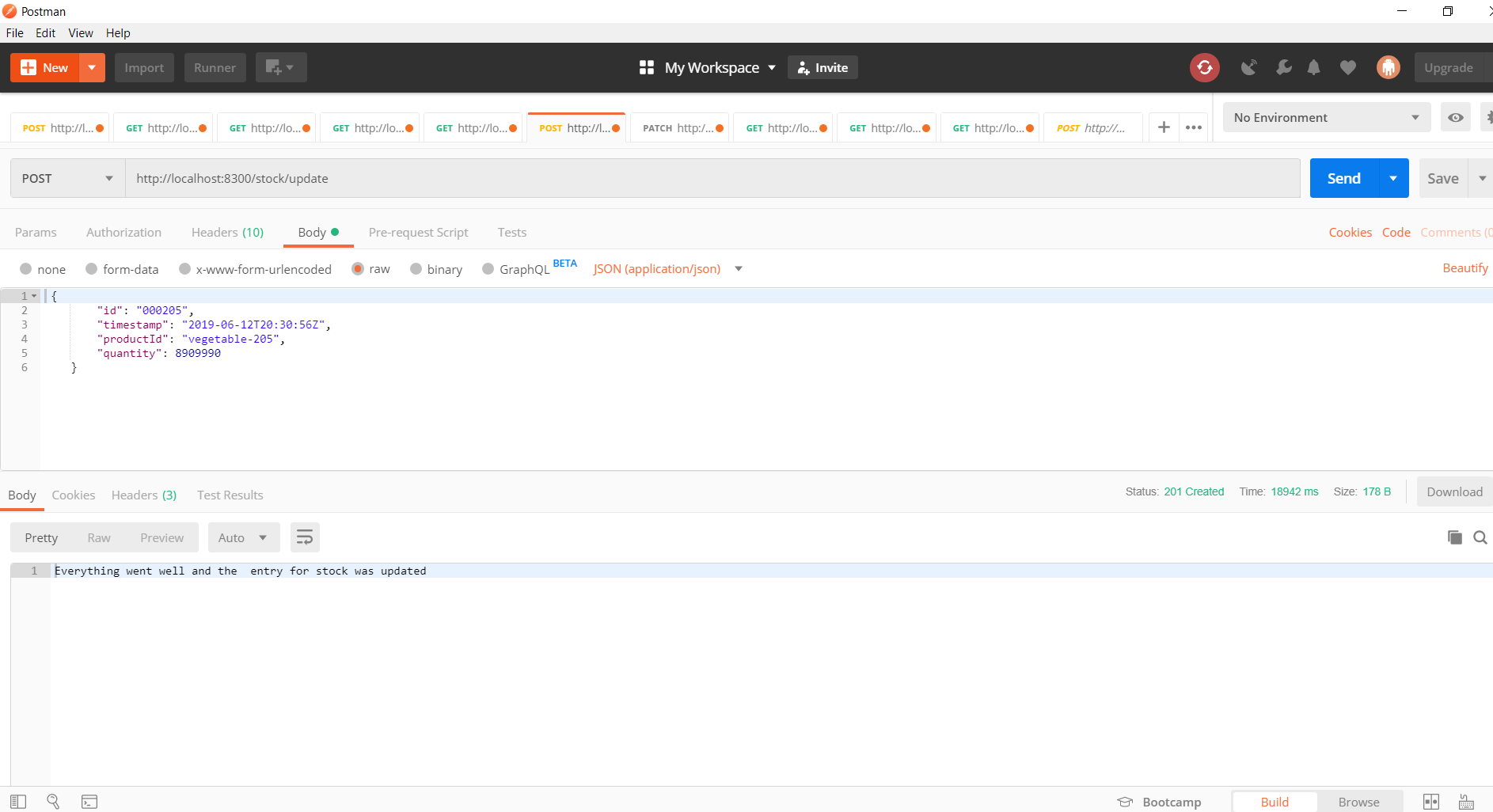
Some Request and Response Samples:

1. For Statistics:

http://localhost:8301/statistics?time=2019-06-12 22:30:56, 2019-06-01 22:30:56

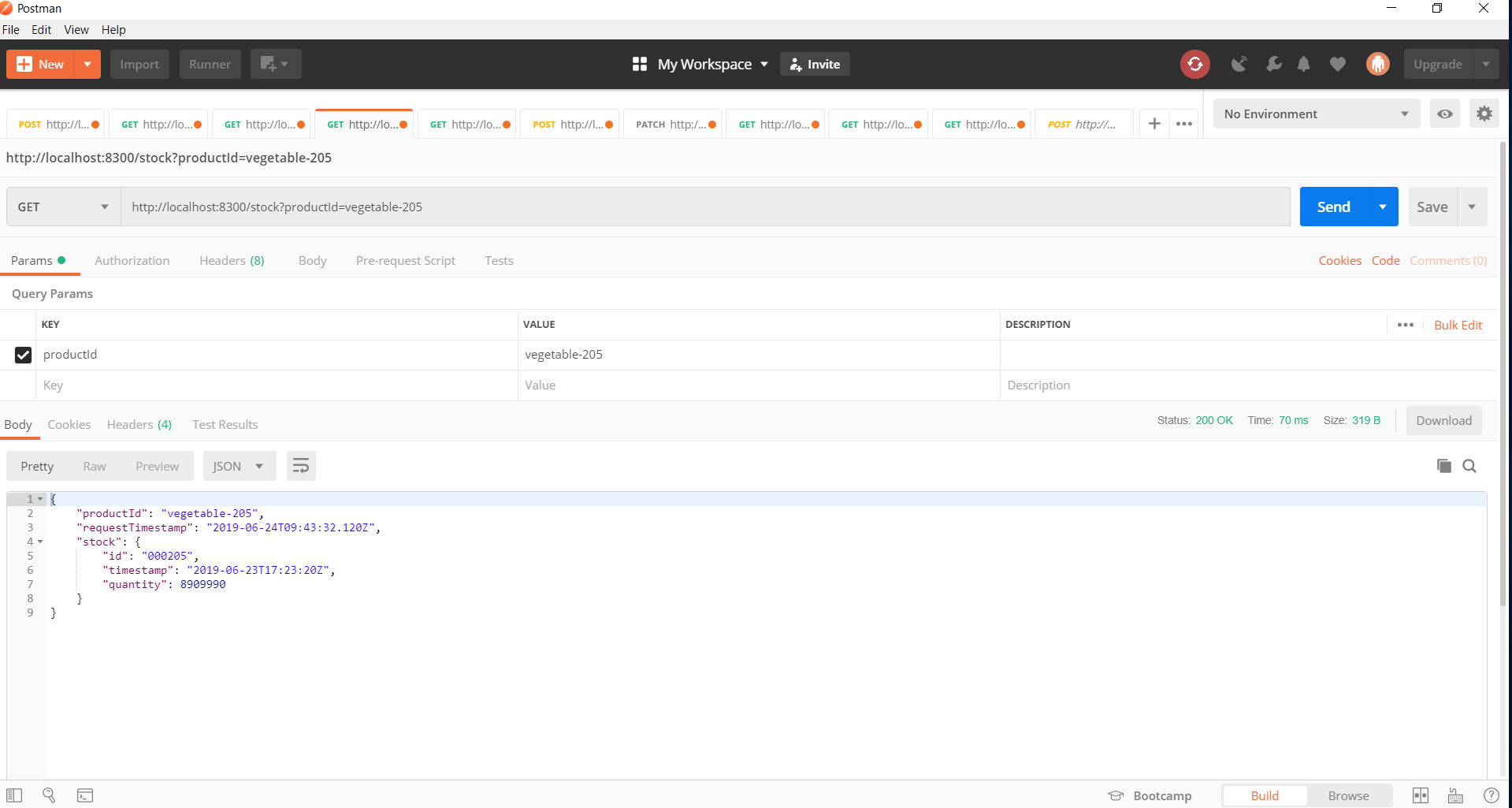


1. For Stock Update:

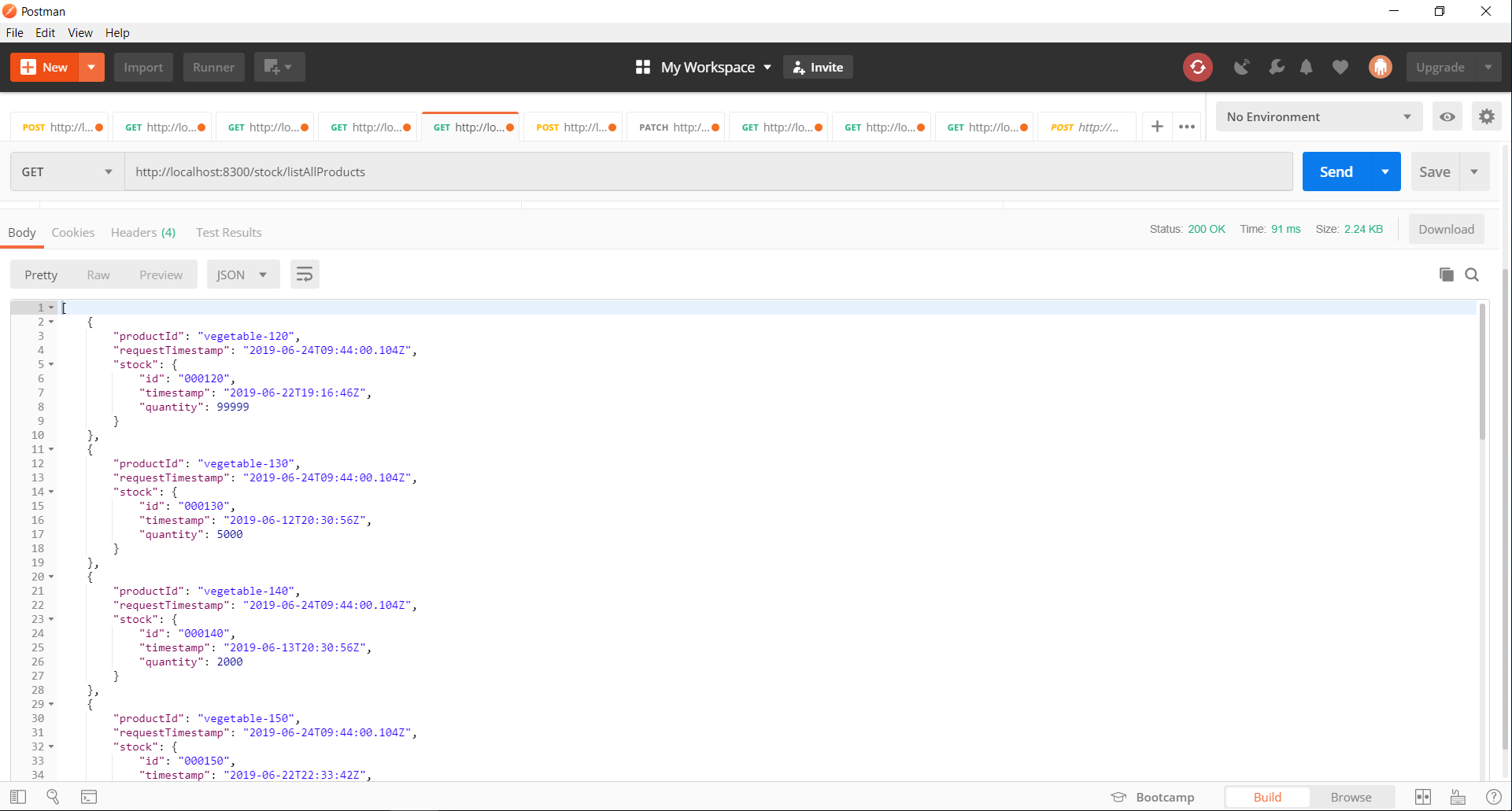
http://localhost:8300/stock/update

1. For GET Product Information:

http://localhost:8300/stock?productId=vegetable-205



1. GET LIST OF ALL PRODUCTS:



Custom Expectation Response handling is added for both stock-service and statics-service

1. EntityNotFoundException
2. DuplicateEntityException
3. ValidDateNotProvidedException
4. NoStocksAvailableException

**Running the Project:**

1. I have added h2 in memory data base in both stock service and statistics service so that first time code should run on the machine.
2. You will be able to add and update the date in **stock service** with h2 database but you thought that I would like to see same data in **statistics service** that is not possible as both the data set run in their separate in memory process.
3. To solve it we just need to following settings in both **stock service &** in **statistics service** :

* Install MYSQL database and please create stocksavailability schema inside it and update following properties files application-dev.properties
  + *spring.datasource.url=jdbc:mysql://localhost:3306/stocksavailability?useSSL=false*
  + *spring.datasource.username=root*
  + *spring.datasource.password=root*
* In *application.properties* please use *spring.profiles.active=dev* so that it will pick dev properties settings and connect to the data base

1. Sequence of running the modules.
2. eureka-service
3. stock-service
4. statistics-service

Also we can run both the services stock-service and statistics-service independently

For deploying in production we just need to change profile setting in main *application.properties* files please use *spring.profiles.active=prod* and change thedatabase detailsconnection details.

1. application-dev.properties (For Development)
2. application-test.properties (For unit Testing)
3. application-prod.properties(For Prdouction)

We can externalize this configuration using spring-cloud to make it centralize but because of time constraint I was not able to spend any time on doing this also I wanted to create one more common db.-service which will supply the data to stock-service and statistics-service so final flow would be:

API Gateway -> Service registry -> Stock Service/Statistics Service -> DB Service -> MSQL