Microservices Architecture and Programming

**Lab Practical and date** – Practical 6, 30th October 2020

**Name and Roll Number**- Het Shah, 17BIT103

**Practical Objective-** Developing complex Micro-services with integration of API Gateway

**Steps Involved-**

We implemented an Simple Api-gateway building using node.js

Dependencies used are fast-gateway & restana.

**Running this project**

Step 1 : open cmd/terminal in the current path.

Step 2 : go into the ApiGateway folder by running \_cd ./ApiGateway command in cmd.

Step 3 : run *npm install* command.

Step 4 : go into the OrderService folder by running *cd ./OrderService* command in cmd.

Step 5 : run *npm install* command.

Step 6 : go into the ProductService folder by running *cd ../ProductService* command in cmd.

Step 7 : run *npm install* command.

Step 8 : go into the BookService folder by running *cd ../BookService* command in cmd.

Step 9 : run *pip install flask* command.

Step 10 : TO run the gateway and services open 4 terminals/cmds.

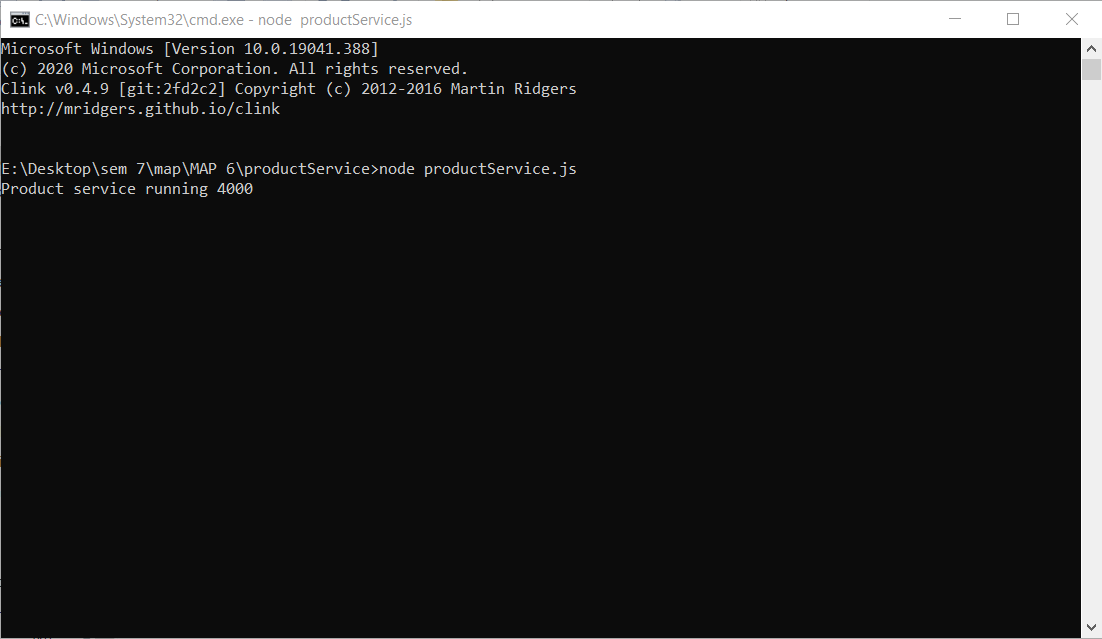
Step 11 : Set path to each cmd to its respective file which are *apiGateway.js* , *orderService.js* , *productService.js* and *api.py*.

Step 12 : Run first three files using *node fileName.js* command and to run BookService and movieService use *python api.py* command.

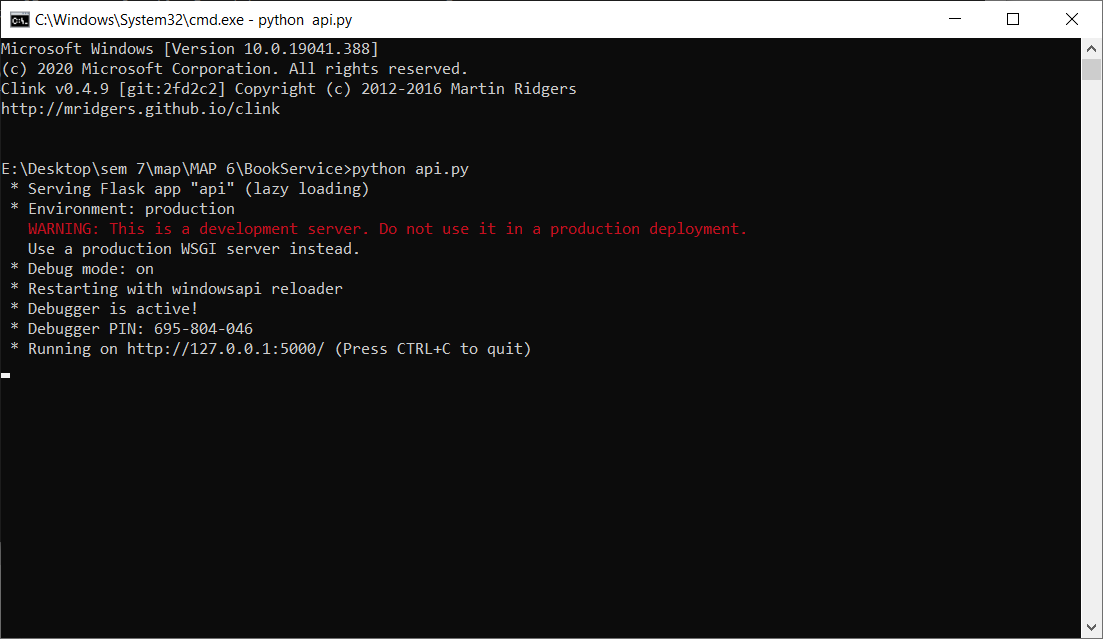
Step 13 : Test services using Postman and generate requests.

Output Screenshots

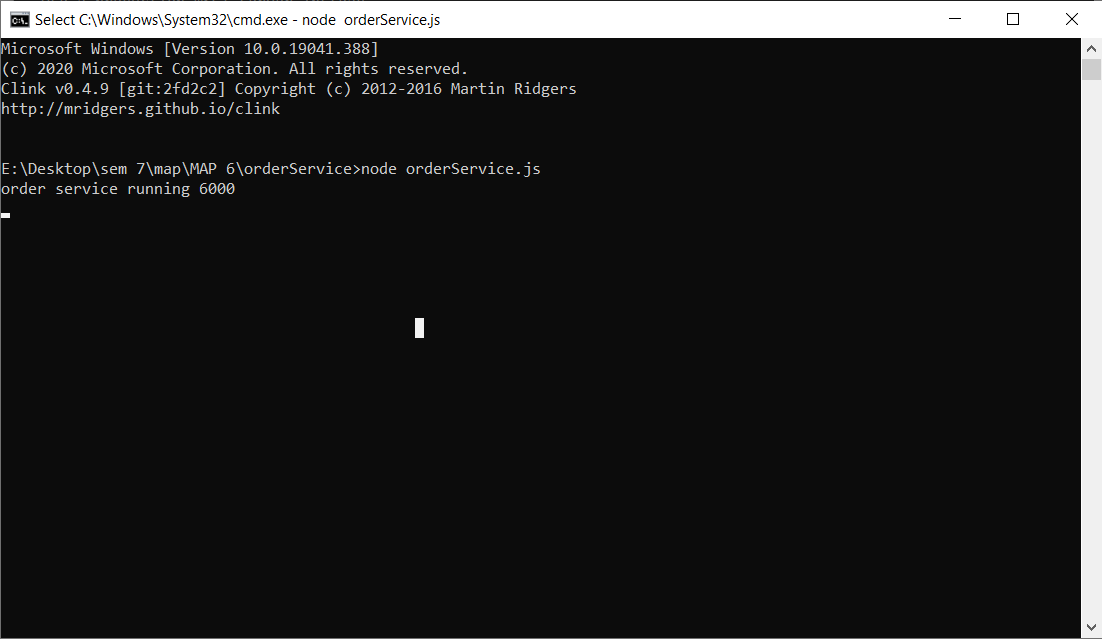
Services started



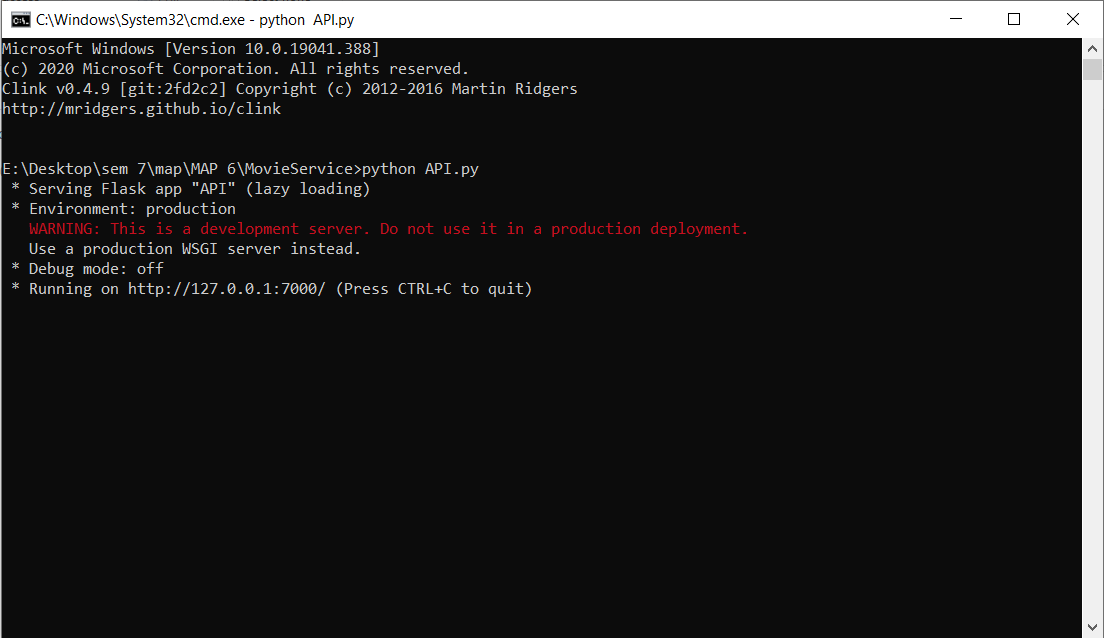
Product service at 4000



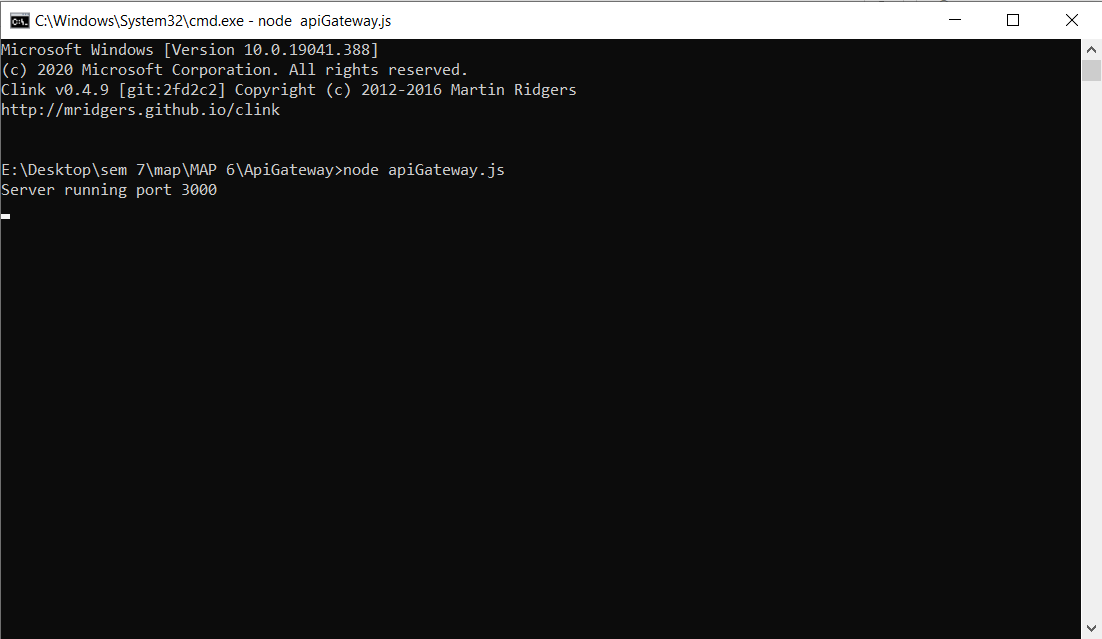
BookService at 5000



Order Service at 6000

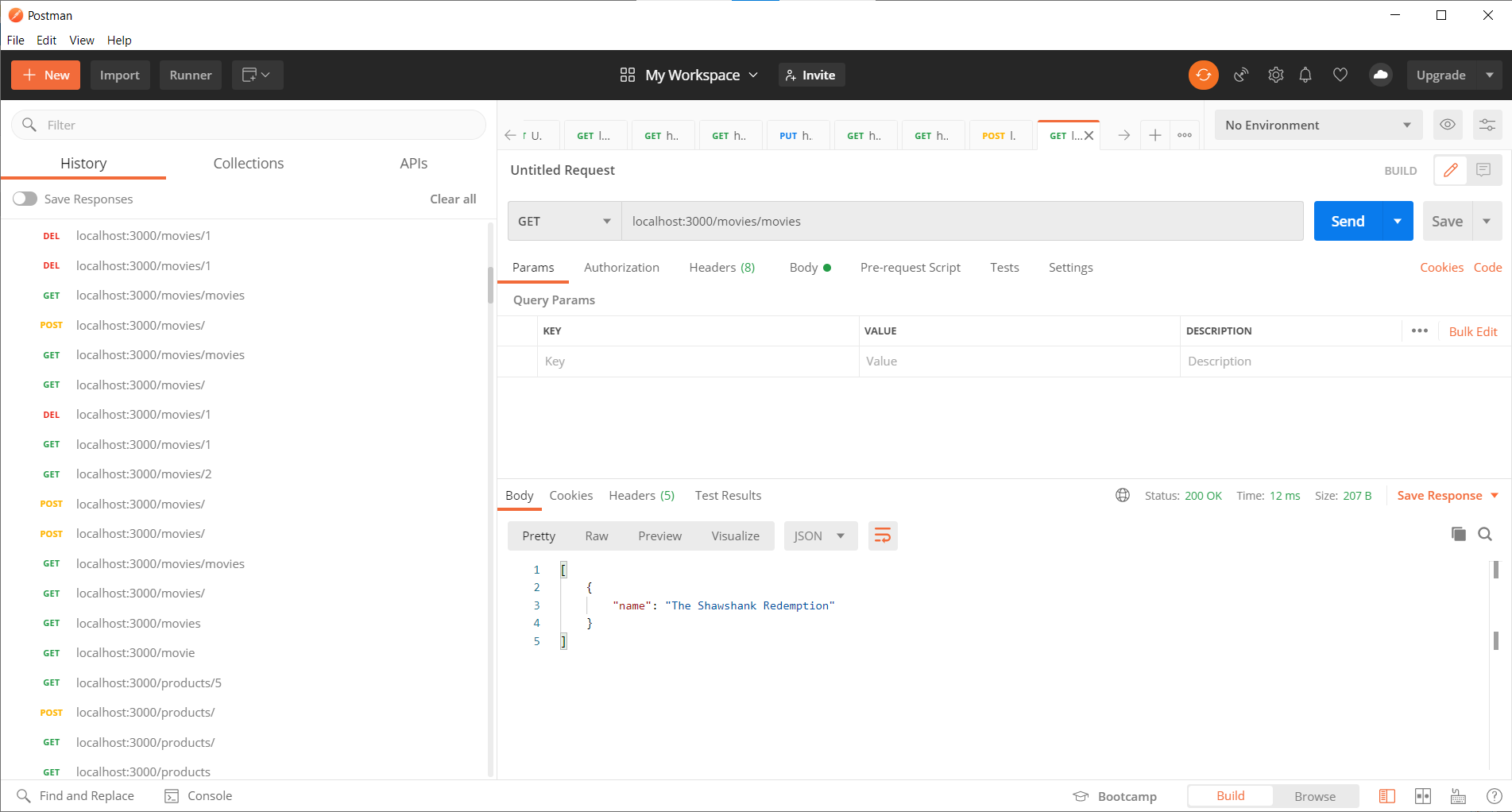


Movie service at 7000

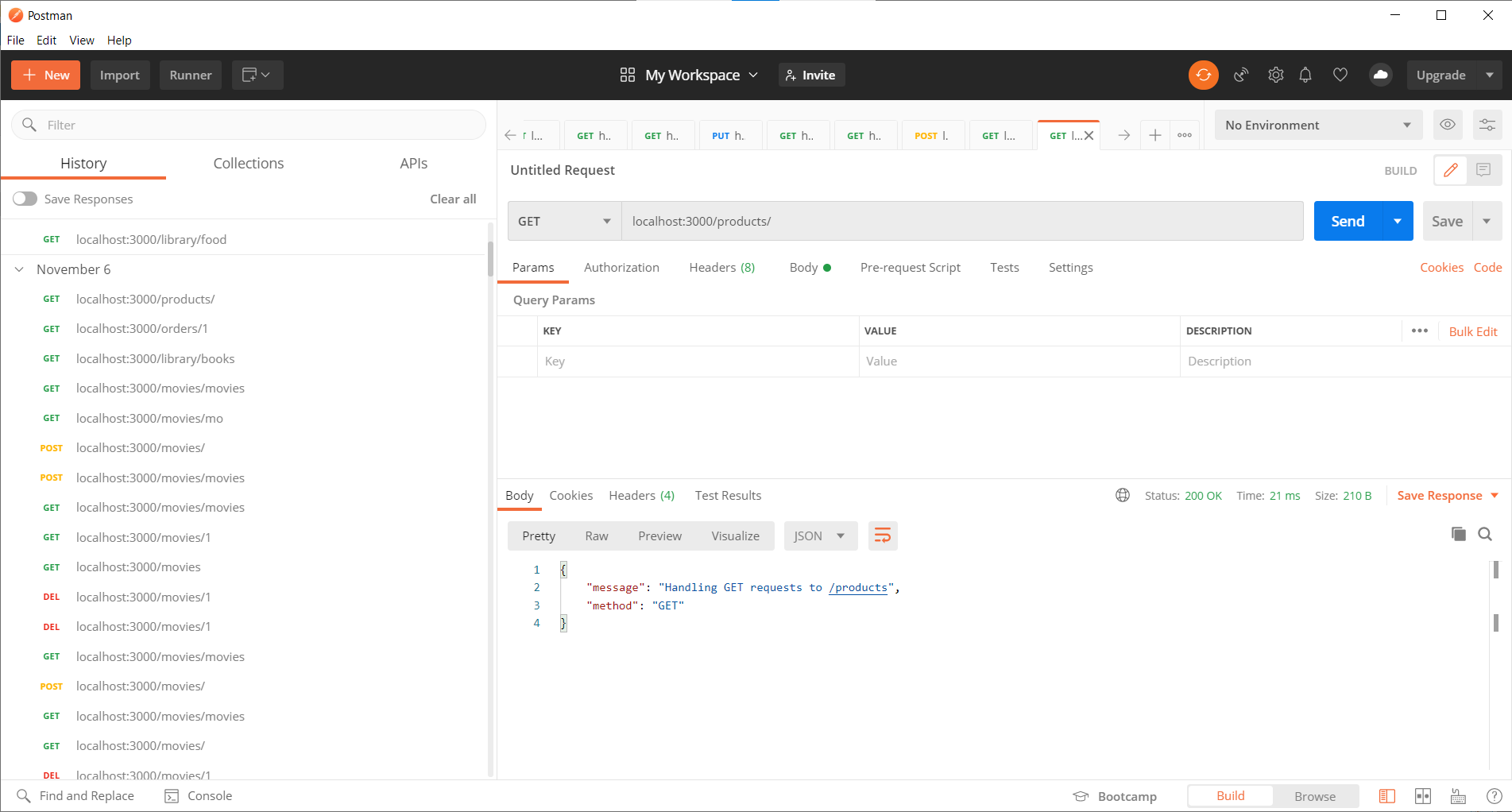


API gateway at 3000

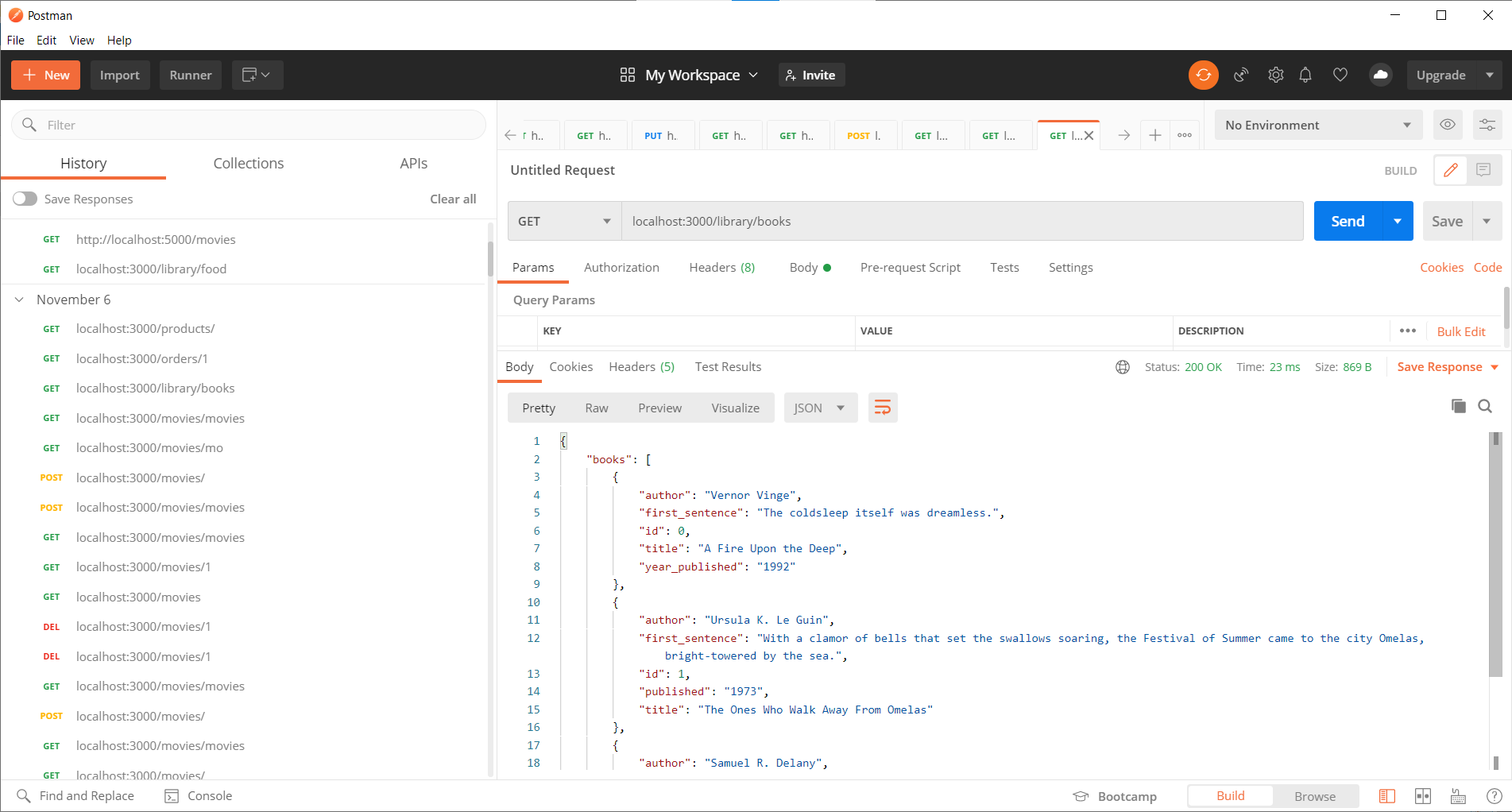
Testing using Postman

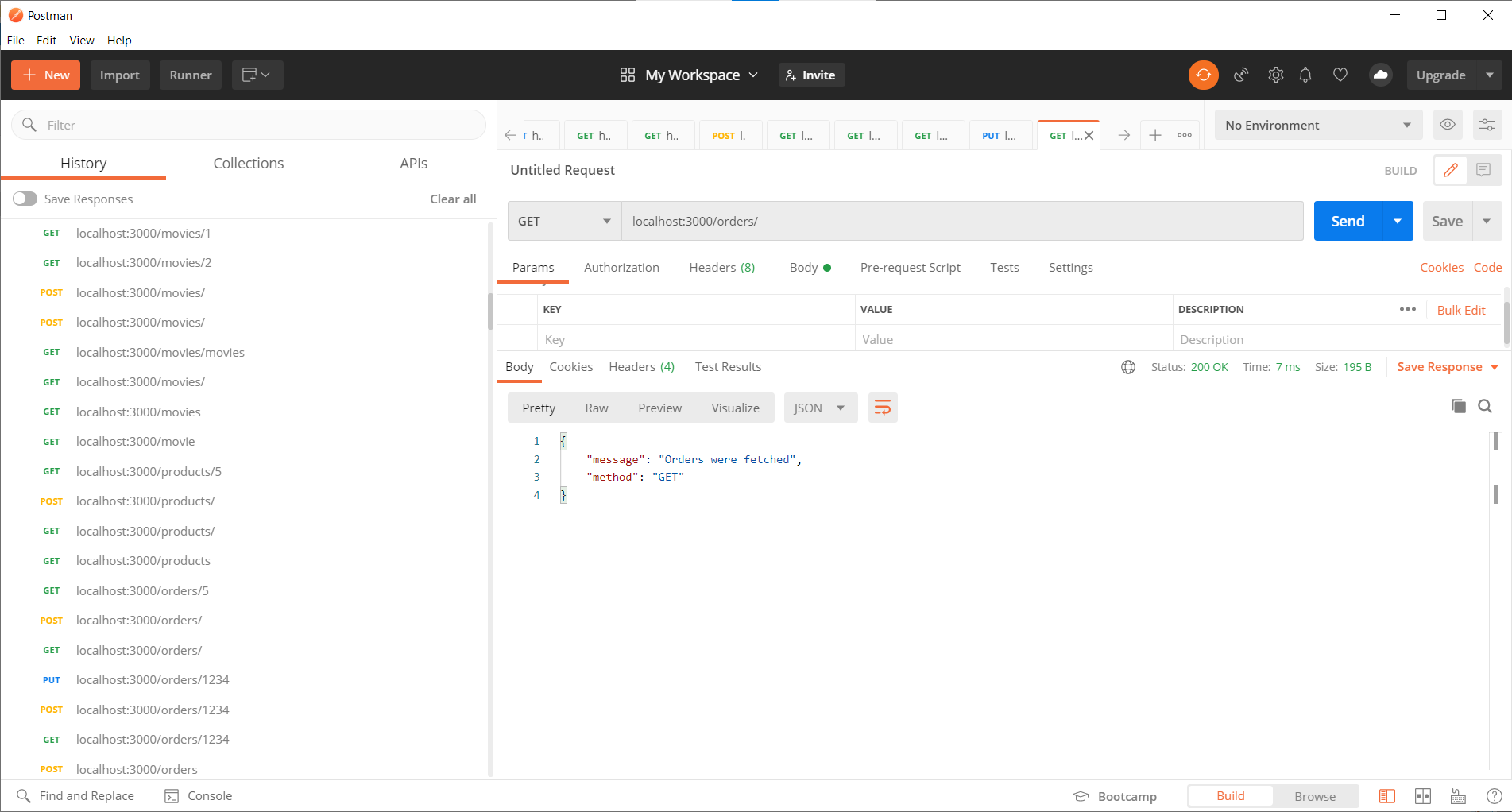


Get for Movie Service



Get for Product Service

Get for book Service



Get for orders service

As we can see we can call all the methods of all four services via the 3000 localhost

Conclusion

In this practical we learned about creating an API gateway and how it can enable security.