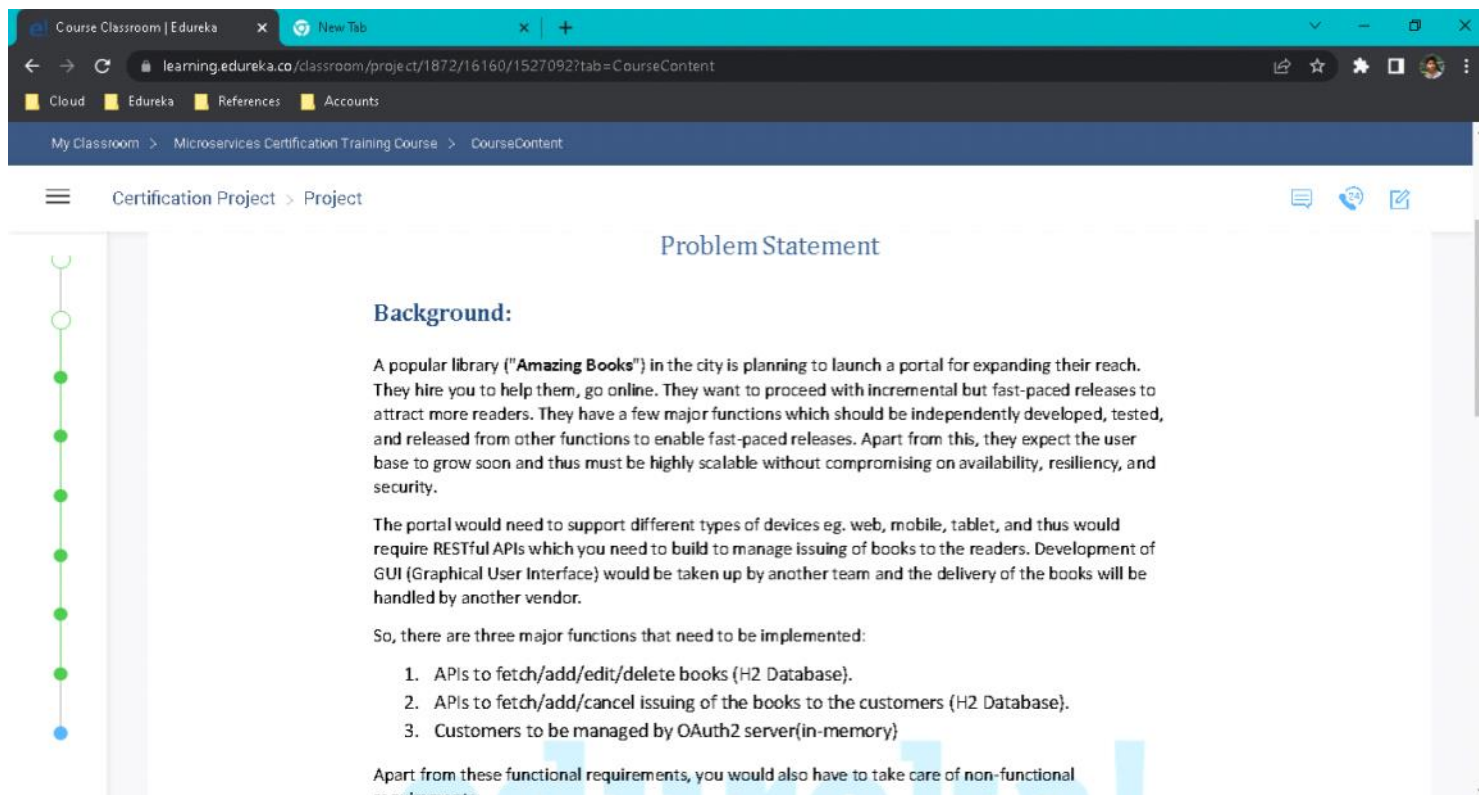


Project



The screenshot shows a web browser window with the URL `learning.edureka.co/classroom/project/1872/16160/1527092?tab=CourseContent`. The page title is "Certification Project > Project". The main content area is titled "Problem Statement".

Background:

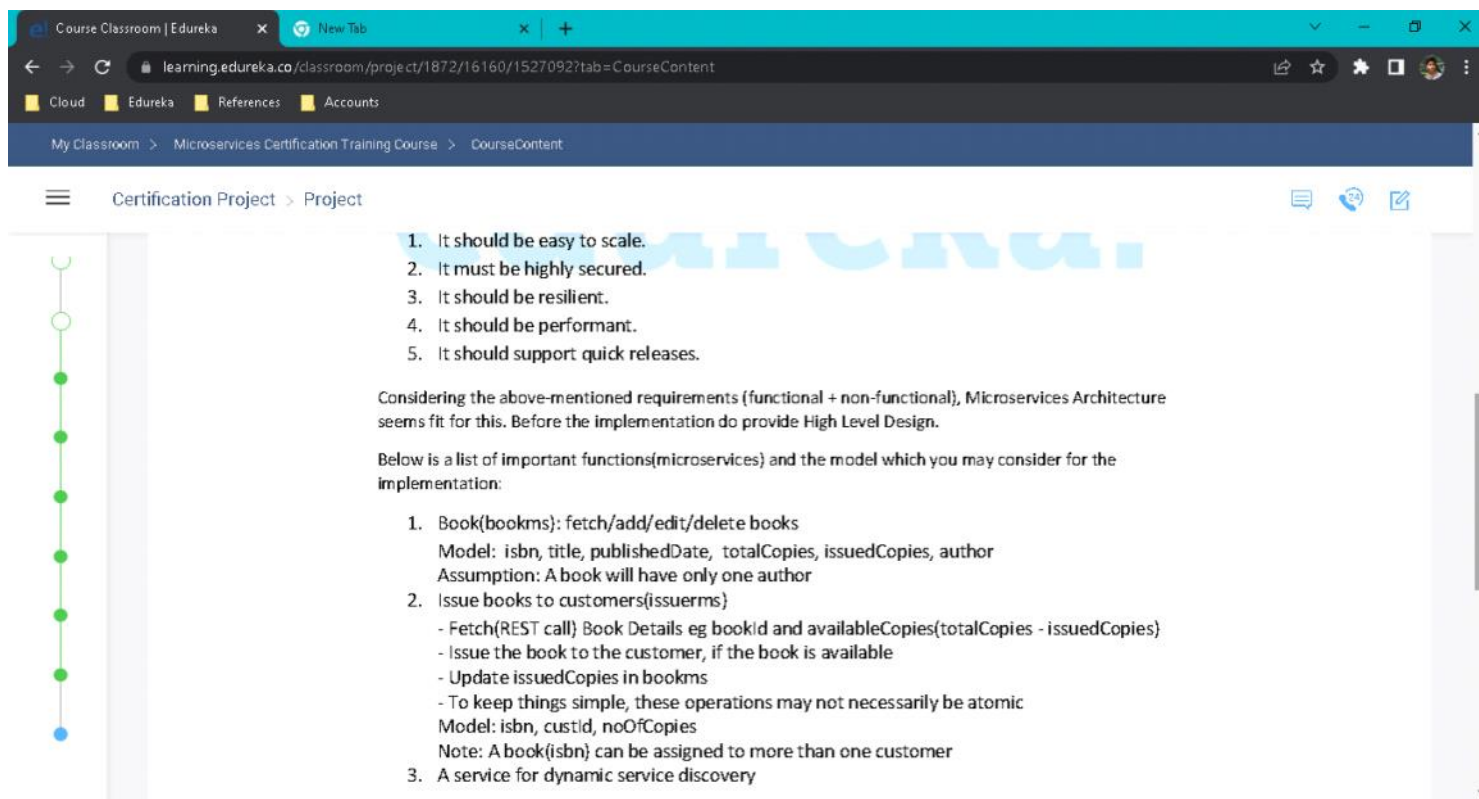
A popular library ("Amazing Books") in the city is planning to launch a portal for expanding their reach. They hire you to help them, go online. They want to proceed with incremental but fast-paced releases to attract more readers. They have a few major functions which should be independently developed, tested, and released from other functions to enable fast-paced releases. Apart from this, they expect the user base to grow soon and thus must be highly scalable without compromising on availability, resiliency, and security.

The portal would need to support different types of devices eg. web, mobile, tablet, and thus would require RESTful APIs which you need to build to manage issuing of books to the readers. Development of GUI (Graphical User Interface) would be taken up by another team and the delivery of the books will be handled by another vendor.

So, there are three major functions that need to be implemented:

1. APIs to fetch/add/edit/delete books (H2 Database).
2. APIs to fetch/add/cancel issuing of the books to the customers (H2 Database).
3. Customers to be managed by OAuth2 server(in-memory)

Apart from these functional requirements, you would also have to take care of non-functional requirements:



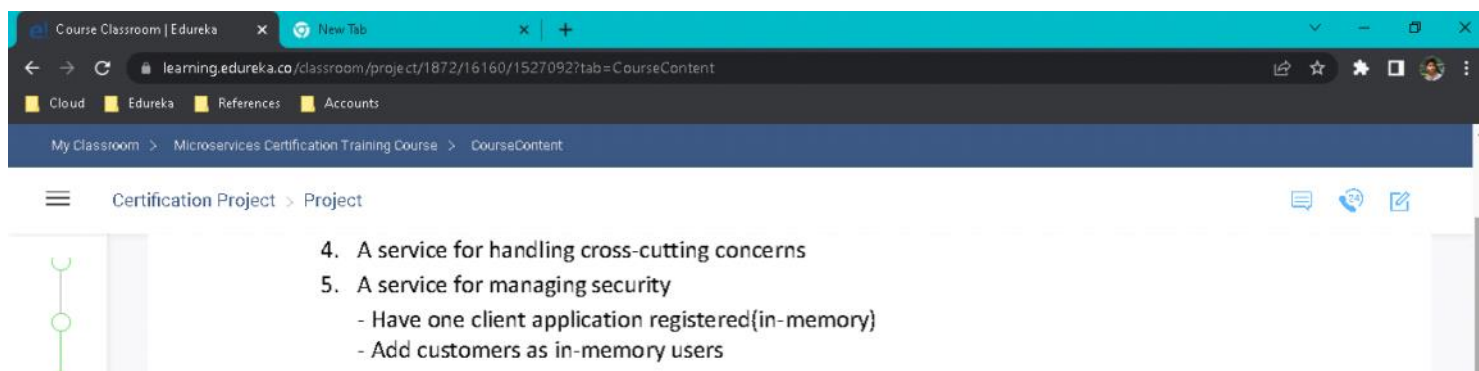
The screenshot continues the "Problem Statement" section. It lists five non-functional requirements:

1. It should be easy to scale.
2. It must be highly secured.
3. It should be resilient.
4. It should be performant.
5. It should support quick releases.

Considering the above-mentioned requirements (functional + non-functional), Microservices Architecture seems fit for this. Before the implementation do provide High Level Design.

Below is a list of important functions(microservices) and the model which you may consider for the implementation:

1. Book(bookms): fetch/add/edit/delete books
Model: isbn, title, publishedDate, totalCopies, issuedCopies, author
Assumption: A book will have only one author
2. Issue books to customers(issuers)
 - Fetch(REST call) Book Details eg bookId and availableCopies(totalCopies - issuedCopies)
 - Issue the book to the customer, if the book is available
 - Update issuedCopies in bookms
 - To keep things simple, these operations may not necessarily be atomicModel: isbn, custId, noOfCopies
Note: A book(isbn) can be assigned to more than one customer
3. A service for dynamic service discovery



The screenshot continues the list of microservices:

4. A service for handling cross-cutting concerns
5. A service for managing security
 - Have one client application registered(in-memory)
 - Add customers as in-memory users

Course Classroom | Edureka x New Tab
learning.edureka.co/classroom/project/1872/16160/1527092?tab=CourseContent
Cloud Edureka References Accounts
My Classroom > Microservices Certification Training Course > CourseContent

☰ Certification Project > Project

4. A service for handling cross-cutting concerns
5. A service for managing security
 - Have one client application registered(in-memory)
 - Add customers as in-memory users

Recommendations:

- 1) Spring boot 2.5.x version or higher
- 2) Java 11
- 3) Use In-Memory(H2) Database wherever applicable
- 4) Use any IDE of your choice
- 5) Use JSON for exchanging data through RESTful APIs
- 6) For security, use OAuth2. Add customers to the OAuth2 server(in-memory)
- 7) Swagger Documentation
- 8) Dockerize your services
- 9) Focus should be on implementing several design patterns relevant in Microservices
- 10) Keep Data Model simple eg. A book may have only one author

Course Classroom | Edureka x New Tab
learning.edureka.co/classroom/project/1872/16160/1527092?tab=CourseContent
Cloud Edureka References Accounts
My Classroom > Microservices Certification Training Course > CourseContent

☰ Certification Project > Project

- 7) Swagger Documentation
- 8) Dockerize your services
- 9) Focus should be on implementing several design patterns relevant in Microservices
- 10) Keep Data Model simple eg. A book may have only one author
- 11) Call out assumptions that you may make in implementation in terms of domain
- 12) Use OAuth2 server to manage users(in-memory)
- 13) Use Prometheus and Grafana to monitor your application

Deliverables:

- 1) High Level Design
- 2) Code of all services (five)
- 3) API sample request/response
- 4) Highlight any improvements that you may have incorporated

Course Classroom | Edureka

Spring Initializr

start.spring.io

Cloud

Edureka

References

Accounts

spring initializr

Project

☐ Gradle - Groovy
 ☒ Gradle - Kotlin

☐ Java
 ☒ Kotlin
 ☐ Groovy

Spring Boot

☐ 3.1.0 (SNAPSHOT)
 ☐ 3.1.0 (RC2)
 ☐ 3.1.0 (M2)

☐ 3.0.7 (SNAPSHOT)
 ☐ 3.0.6
 ☐ 2.7.12 (SNAPSHOT)

☒ 2.7.11

Project Metadata

Group

com.vagish.edureka

Artifact

booksms

DEPENDENCIES

Eureka Discovery Client

SPRING CLOUD DISCOVERY

A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.

MySQL Driver

SQL

MySQLJDBC driver.

Spring Data JPA

SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

GENERATE

EXPLORE

SHARE...

Course Classroom | Edureka

Spring Initializr

start.spring.io

Cloud

Edureka

References

Accounts

spring initializr

Project

☐ Gradle - Groovy
 ☒ Gradle - Kotlin

☐ Java
 ☒ Kotlin
 ☐ Groovy

Spring Boot

☐ 3.1.0 (SNAPSHOT)
 ☐ 3.1.0 (RC2)
 ☐ 3.1.0 (M2)

☐ 3.0.7 (SNAPSHOT)
 ☐ 3.0.6
 ☐ 2.7.12 (SNAPSHOT)

☒ 2.7.11

Project Metadata

Group

com.vagish.edureka

Artifact

issuer-customermms

Name

issuer-customermms

DEPENDENCIES

Eureka Discovery Client

SPRING CLOUD DISCOVERY

A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.

MySQL Driver

SQL

MySQLJDBC driver.

Spring Data JPA

SQL

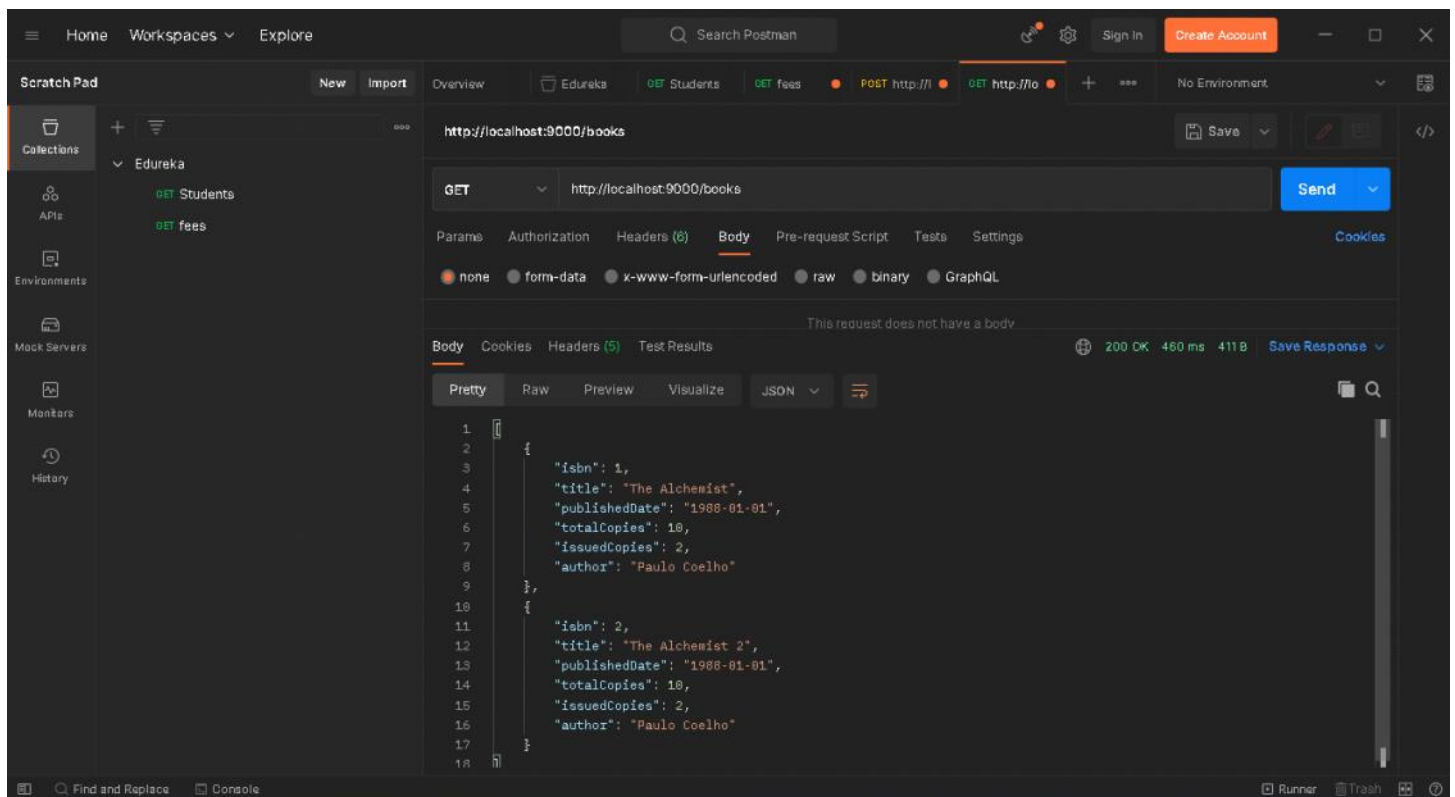
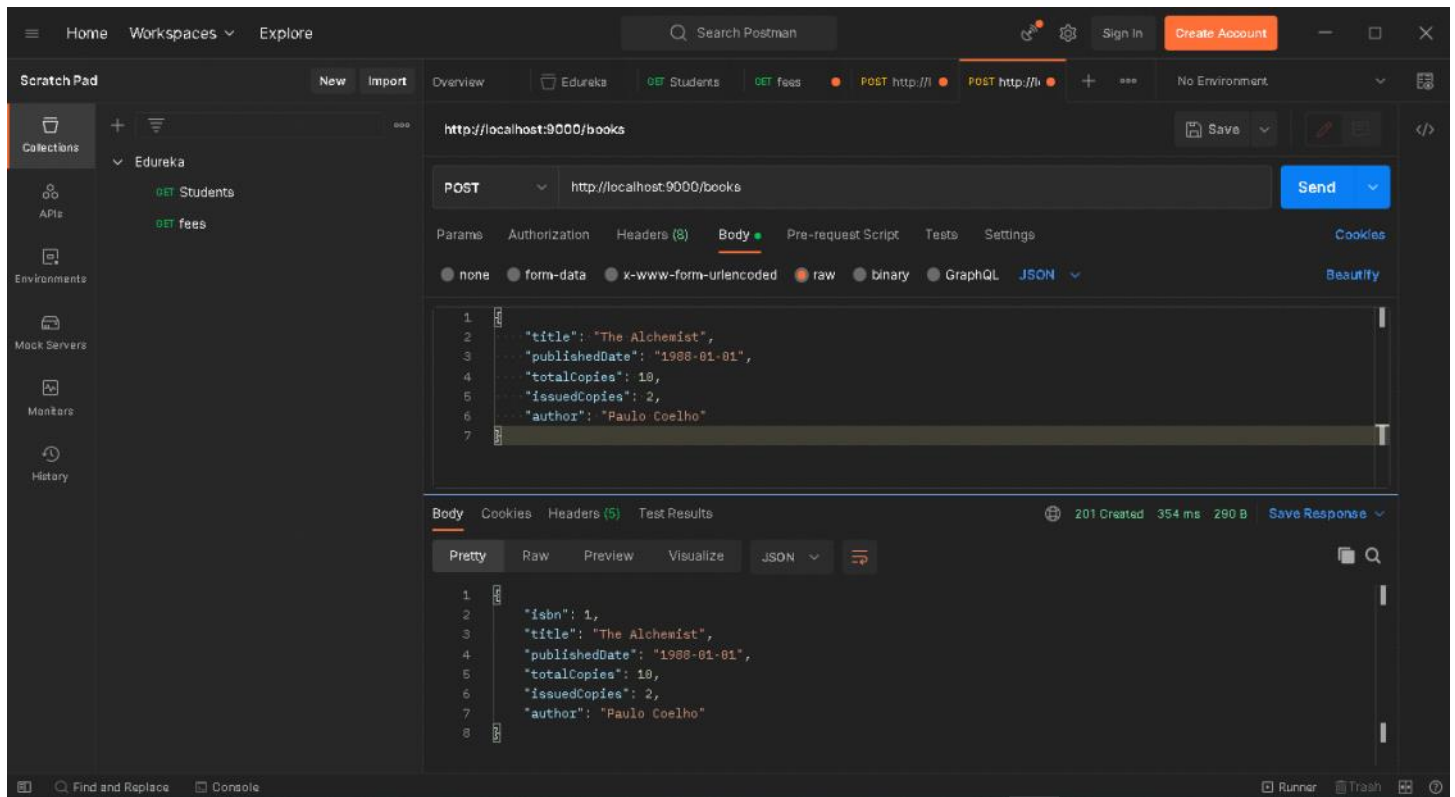
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

GENERATE

EXPLORE

SHARE...

My Edureka - MicroS Page 3



Home Workspaces Explore Search Postman Sign In Create Account

Scratch Pad New Import Overview Edureka GET Students GET fees POST http://lo GET http://lo No Environment

Collections + Edureka GET Students GET fees APIs Environments Mock Servers Monitors History

http://localhost:9000/books/1 Save

GET http://localhost:9000/books/1 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (5) Test Results 200 OK 9 ms 285 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "isbn": 1,
3   "title": "The Alchemist",
4   "publishedDate": "1988-01-01",
5   "totalCopies": 10,
6   "issuedCopies": 2,
7   "author": "Paulo Coelho"
8 }
```

Find and Replace Console Runner Trash

Home Workspaces Explore Search Postman Sign In Create Account

Scratch Pad New Import Overview Edureka GET Students GET fees GET http://lo GET http://lo No Environment

Collections + Edureka GET Students GET fees APIs Environments Mock Servers Monitors History

http://localhost:9001/customers/ Save

GET http://localhost:9001/customers/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

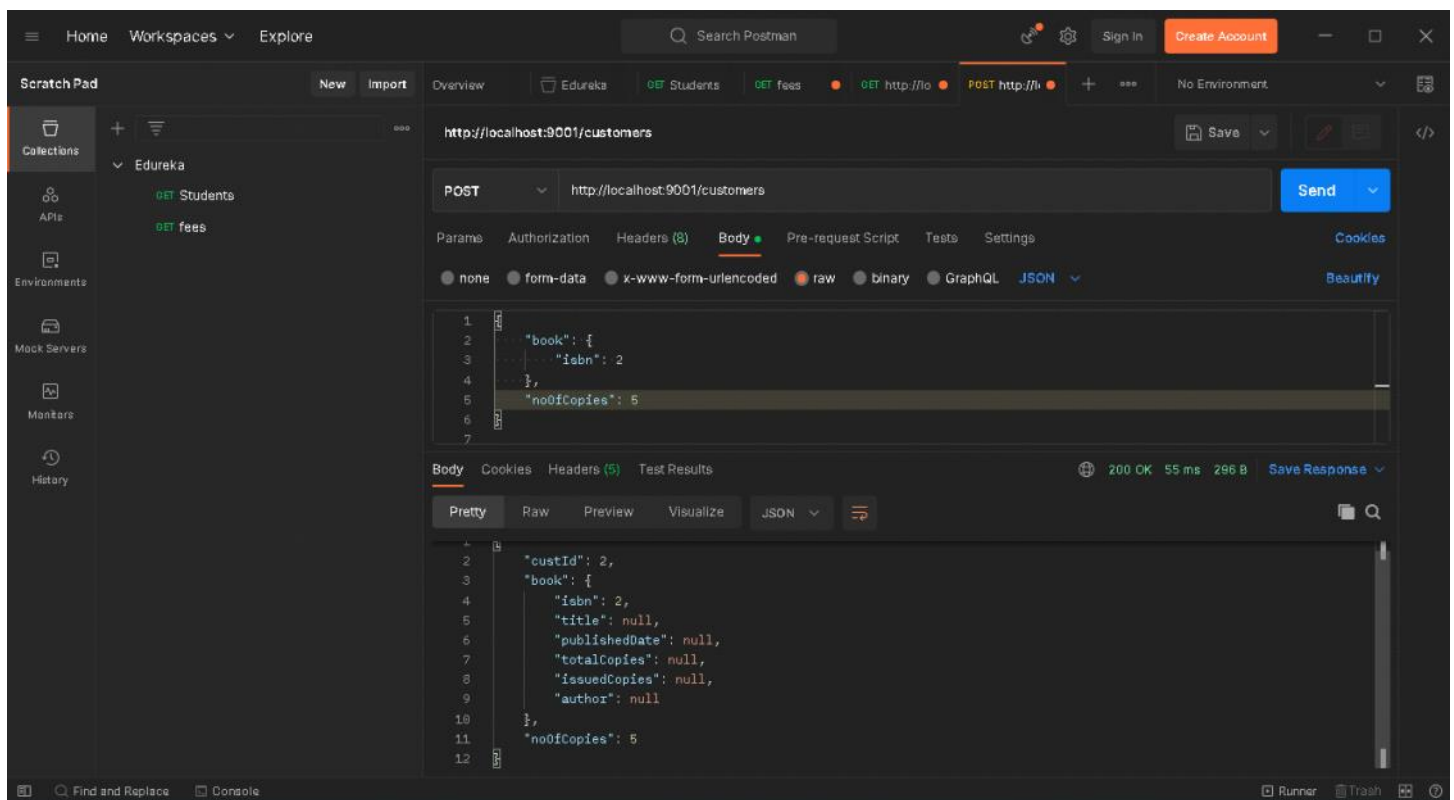
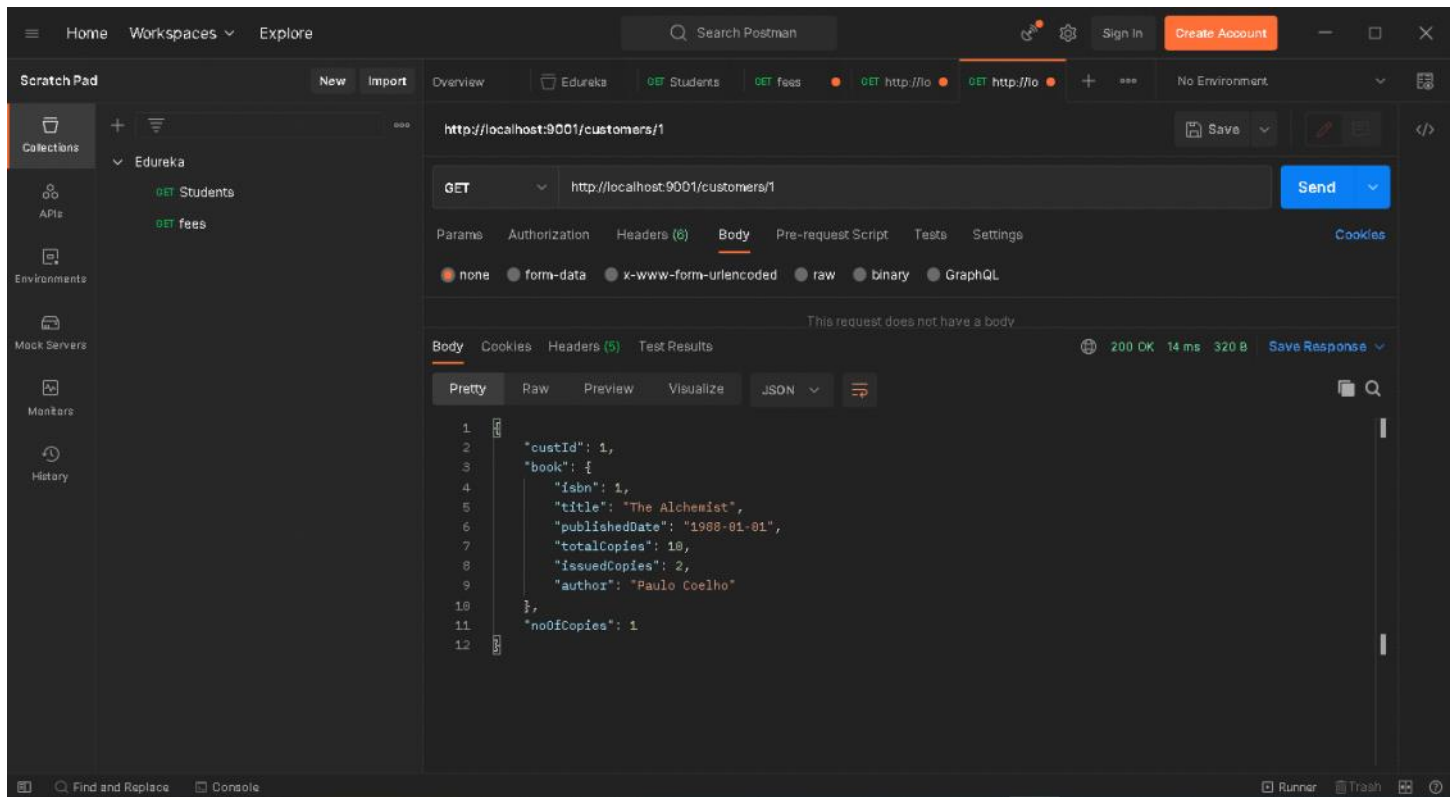
This request does not have a body

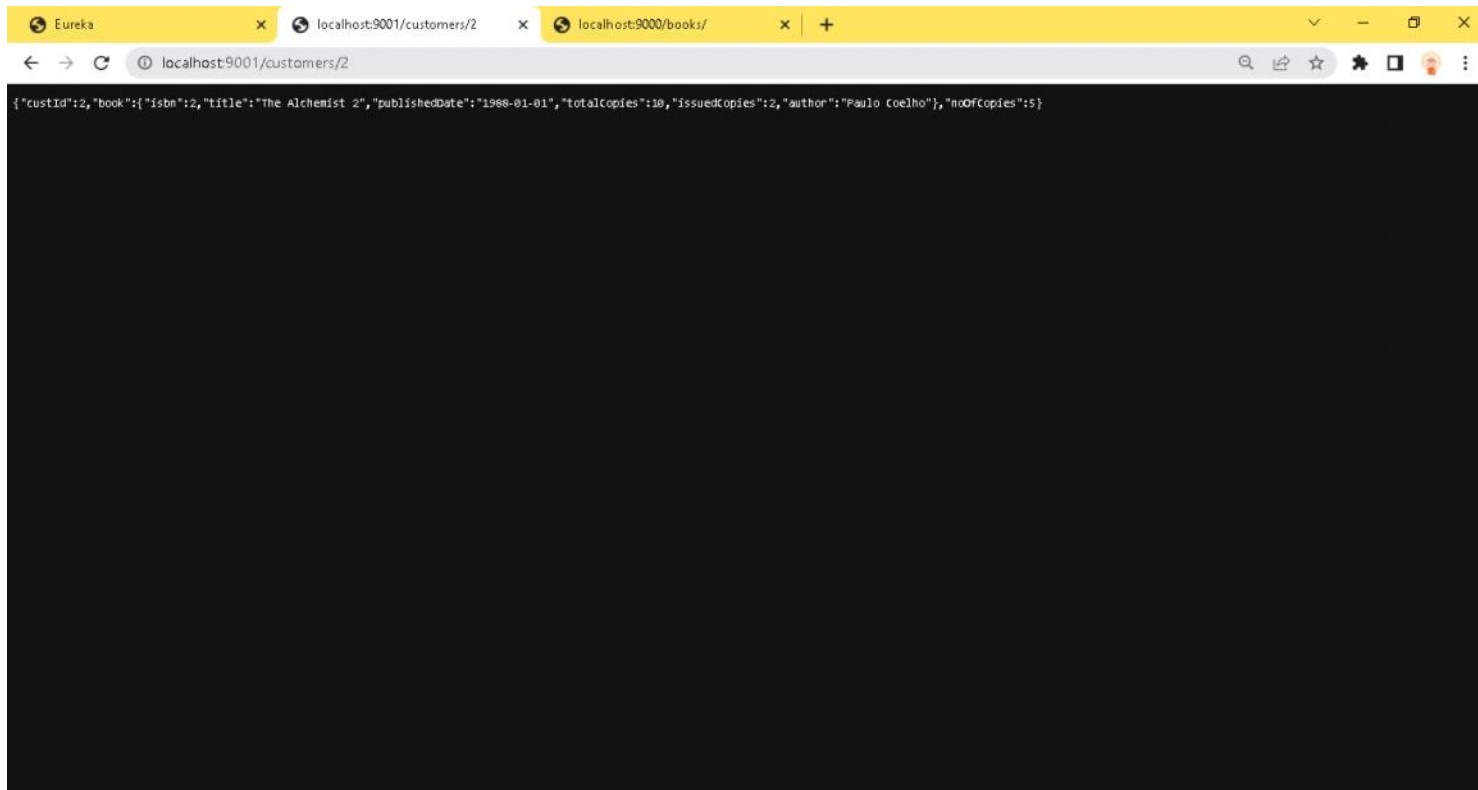
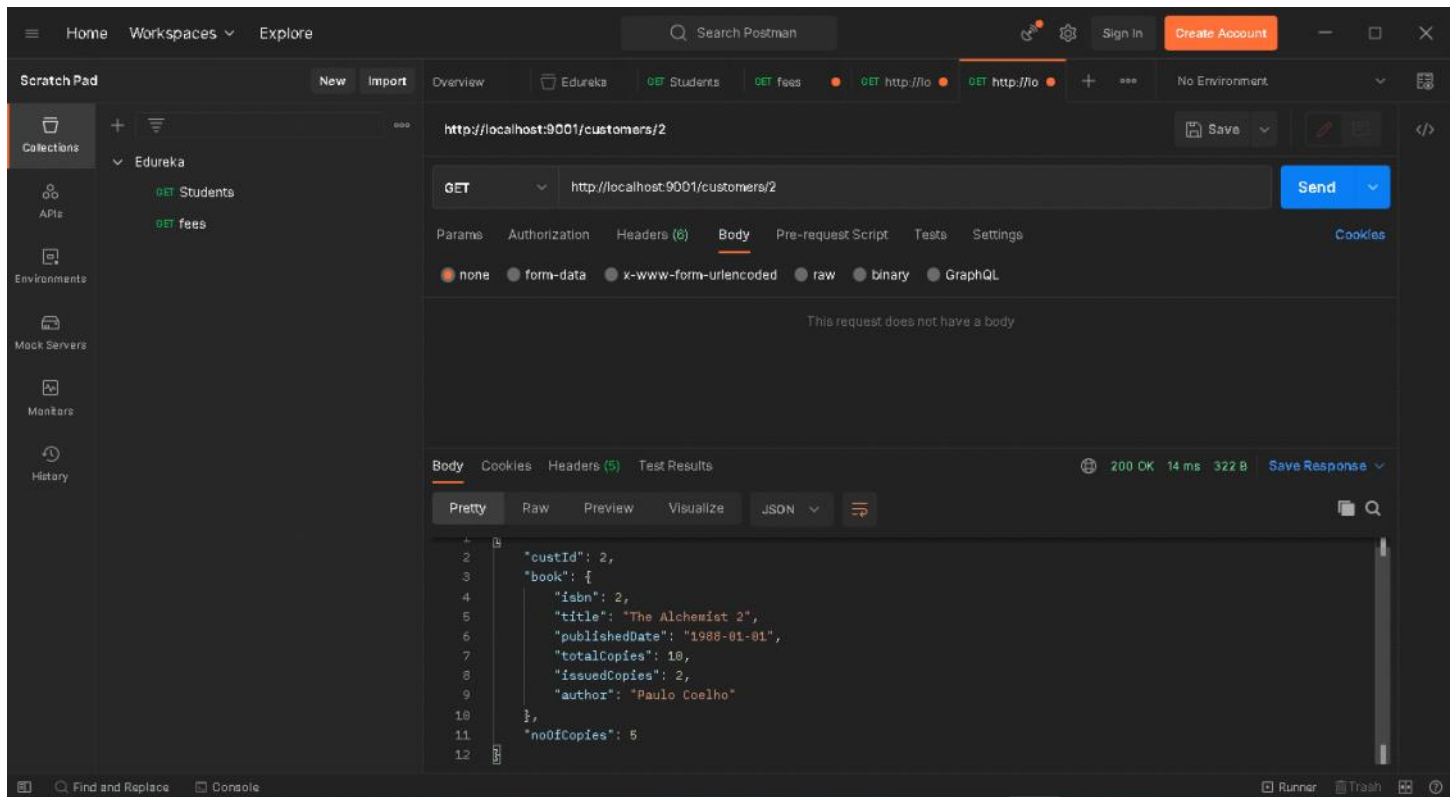
Body Cookies Headers (5) Test Results 200 OK 251 ms 322 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "custId": 1,
3   "book": {
4     "isbn": 1,
5     "title": "The Alchemist",
6     "publishedDate": "1988-01-01",
7     "totalCopies": 10,
8     "issuedCopies": 2,
9     "author": "Paulo Coelho"
10    },
11   "noOfCopies": 1
12 }
13
14 }
```

Find and Replace Console Runner Trash





```
Eureka
localhost:9001/customers
localhost:9000/books/
localhost:9001/customers
[{"custId":1,"book":{"isbn":1,"title":"The Alchemist","publishedDate":"1988-01-01","totalCopies":10,"issuedCopies":2,"author":"Paulo Coelho"},"noOfCopies":1}, {"custId":2,"book":{"isbn":2,"title":"The Alchemist 2","publishedDate":"1988-01-01","totalCopies":10,"issuedCopies":2,"author":"Paulo Coelho"},"noOfCopies":5}]
```

Eureka

localhost:9001/customerslocalhost:9000/books/localhost:1111

HOME LAST 1000 SINCE STARTUP

spring Eureka

System Status

Environment	test	Current time	2023-05-06T15:27:16 +0530
Data center	default	Uptime	00:02
		Lease expiration enabled	false
		Renews threshold	5
		Renews (last min)	2

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
BOOKMS	n/a (1)	(1)	UP (1) - localhost:bookms:9000
ISSUERMS	n/a (1)	(1)	UP (1) - localhost:issuermis:9001

General Info

Name	Value
total-avail-memory	298mb


```
1 server:
2   port: 3500
3
4 eureka:
5   client:
6     serviceUrl:
7       defaultZone: http://localhost:1111/eureka/
8   instance:
9     appname: apigateway
10
11 spring:
12   main:
13     web-application-type: reactive
14   application:
15     name: apigateway
16   security:
17     oauth2:
18       resourceserver:
19         opaquetoken:
20           introspection-uri: http://localhost:8080/oauth/token/check_token
21           client-id: testclientid
22           client-secret: test123
23
24 cloud:
25   gateway:
26     routes:
27       - id: booksmodule
28         uri: "http://localhost:9000/books"
29         predicates:
30           - "Path=/books/**"
31       - id: customersmodule
32         uri: "http://localhost:9001/customers"
33         predicates:
34           - "Path=/customers/**"
```

Eureka

localhost:9001/customers localhost:9000/books/ +

localhost:1111

System Status

Environment	test	Current time	2023-05-06T15:30:16 +0530
Data center	default	Uptime	00:05
		Lease expiration enabled	false
		Renews threshold	6
		Renews (last min)	4

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

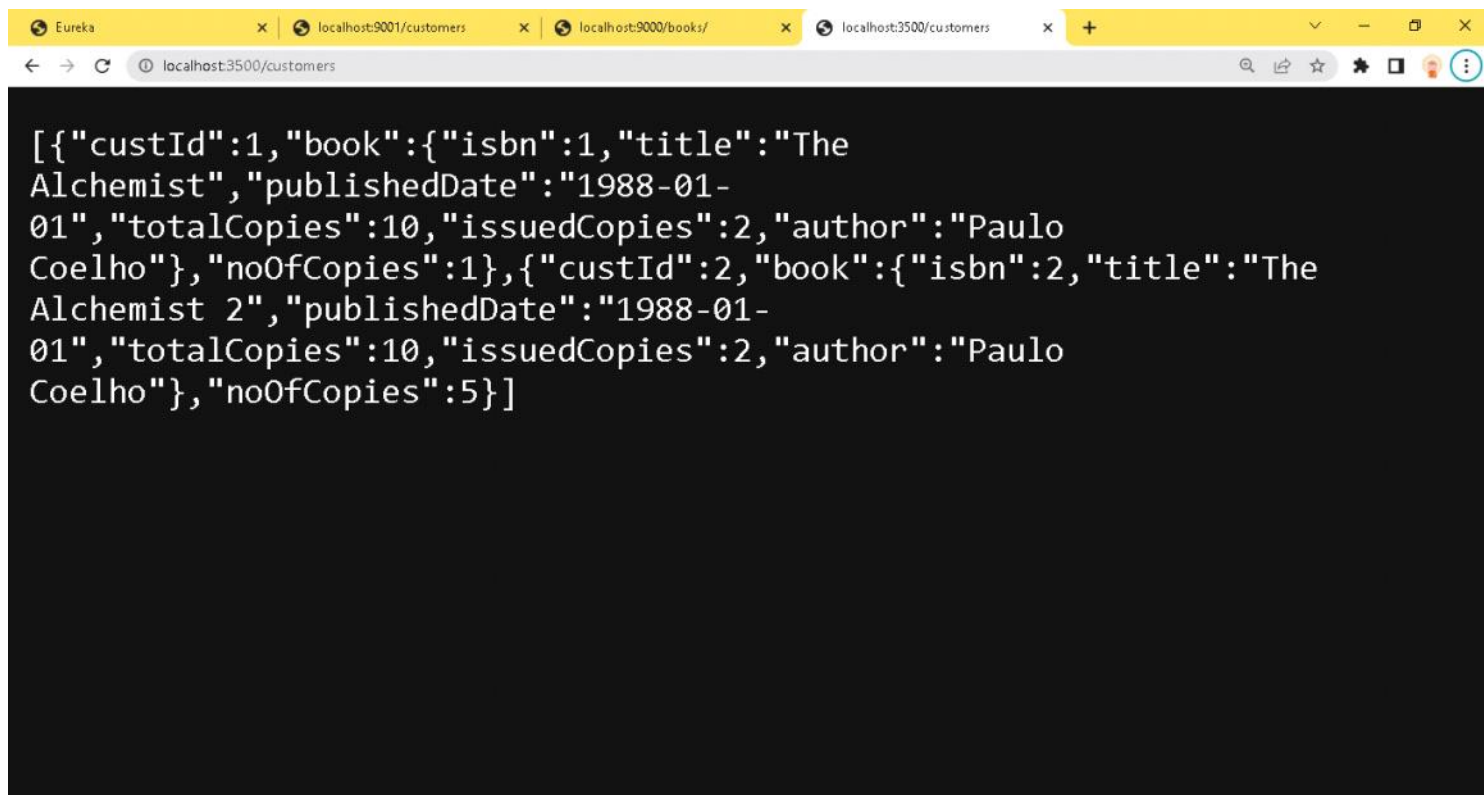
localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
APIGATEWAY	n/a (1)	(1)	UP (1) - localhost.apigateway.3500
BOOKMS	n/a (1)	(1)	UP (1) - localhost.bookms.9000
ISSUERMS	n/a (1)	(1)	UP (1) - localhost.issuerms.9001

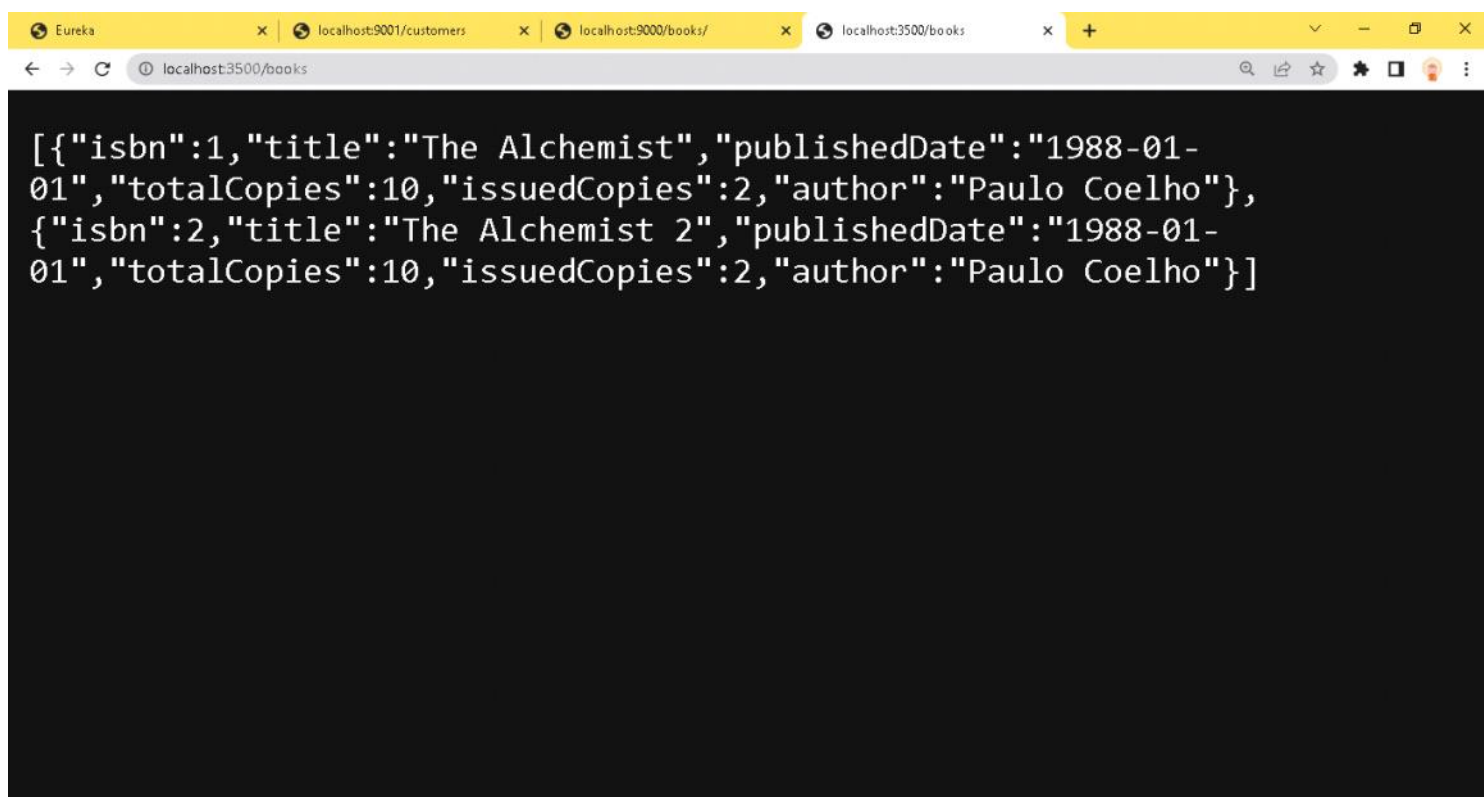
General Info

Name	Value
total-avail-memory	341mb
num-of-cpus	4



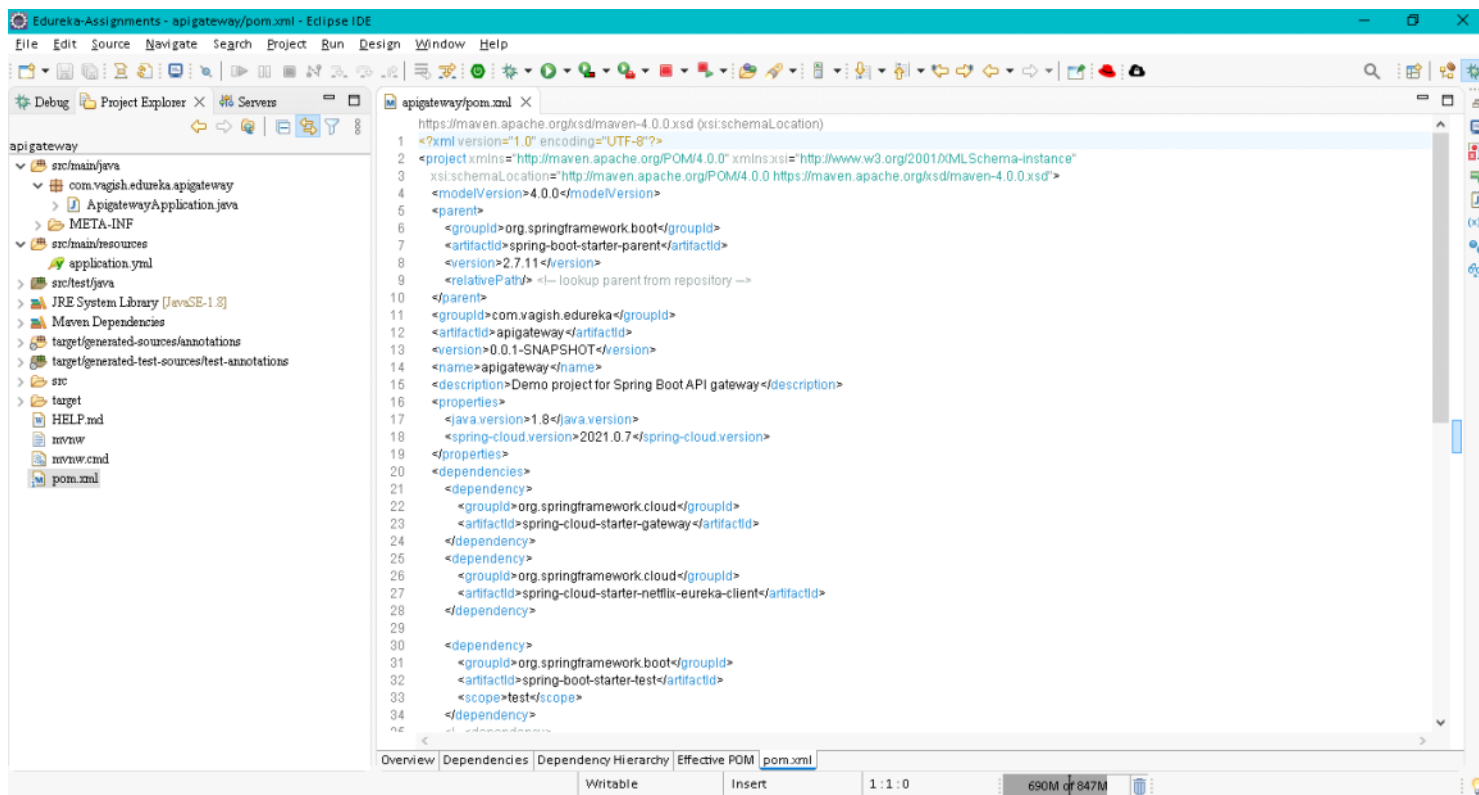
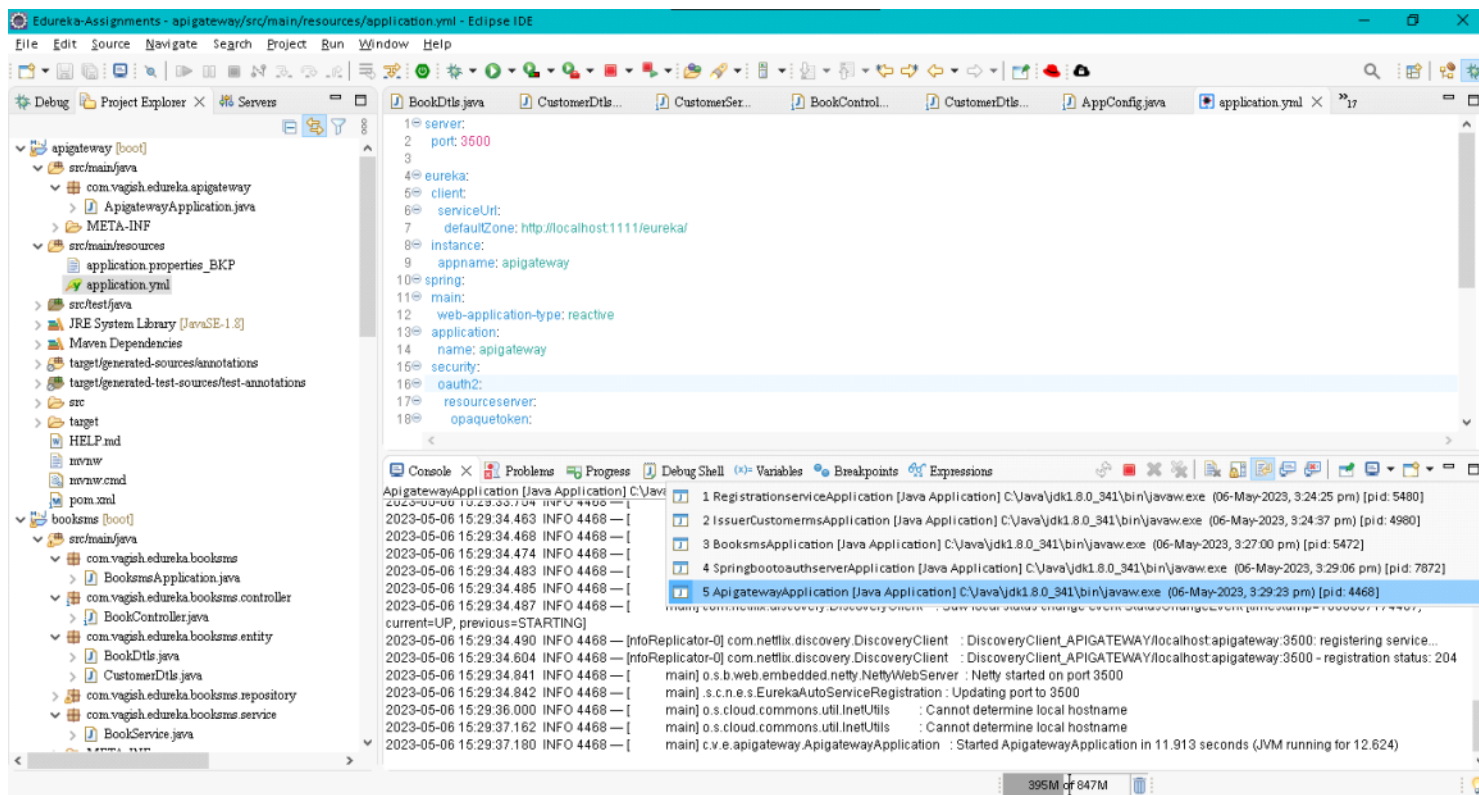
A screenshot of a web browser window with a yellow header bar. The address bar shows 'localhost:3500/customers'. The page content is a dark-themed area displaying a JSON array. The browser's tab bar at the top shows four tabs: 'Eureka', 'localhost:9001/customers', 'localhost:9000/books/', and 'localhost:3500/customers'.

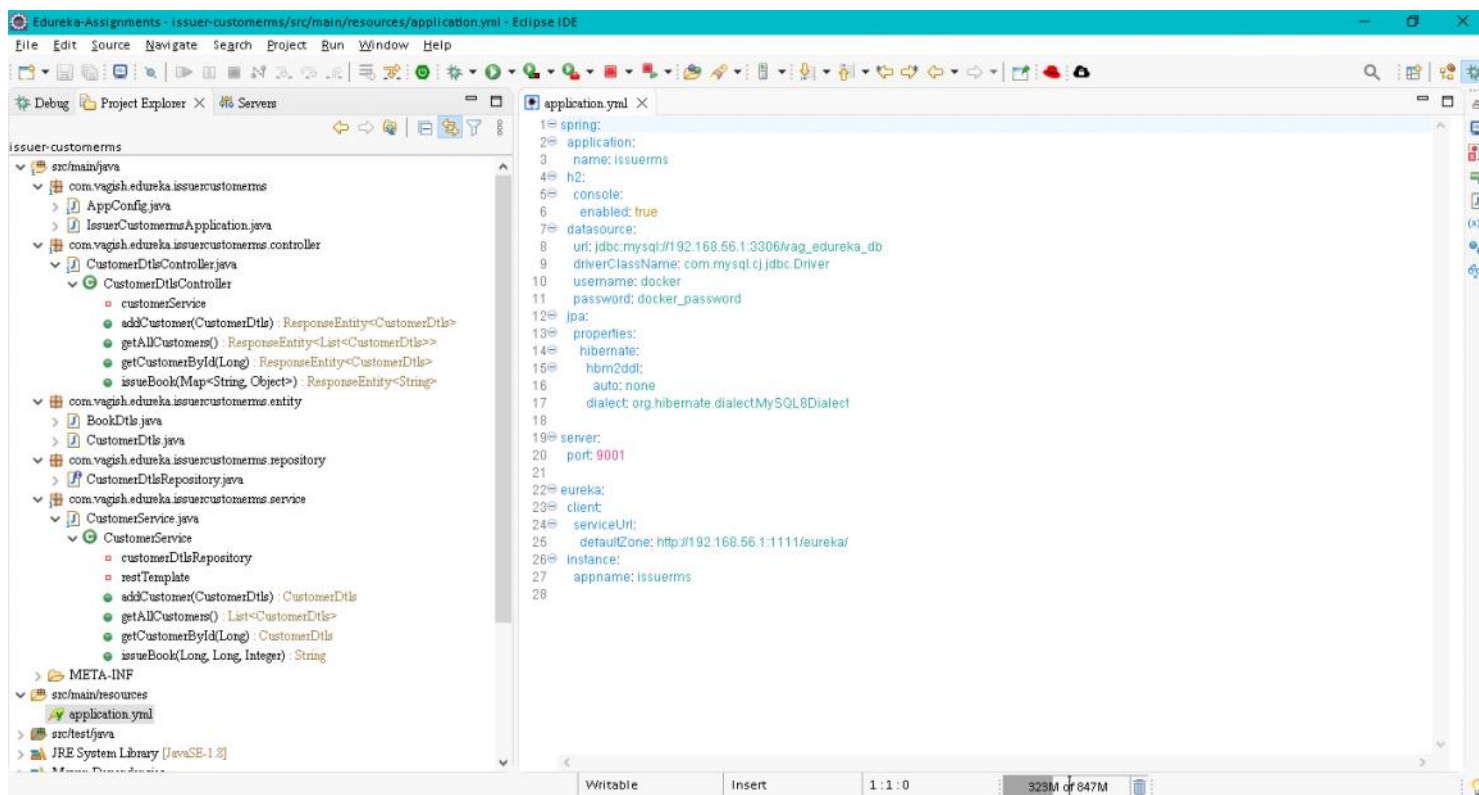
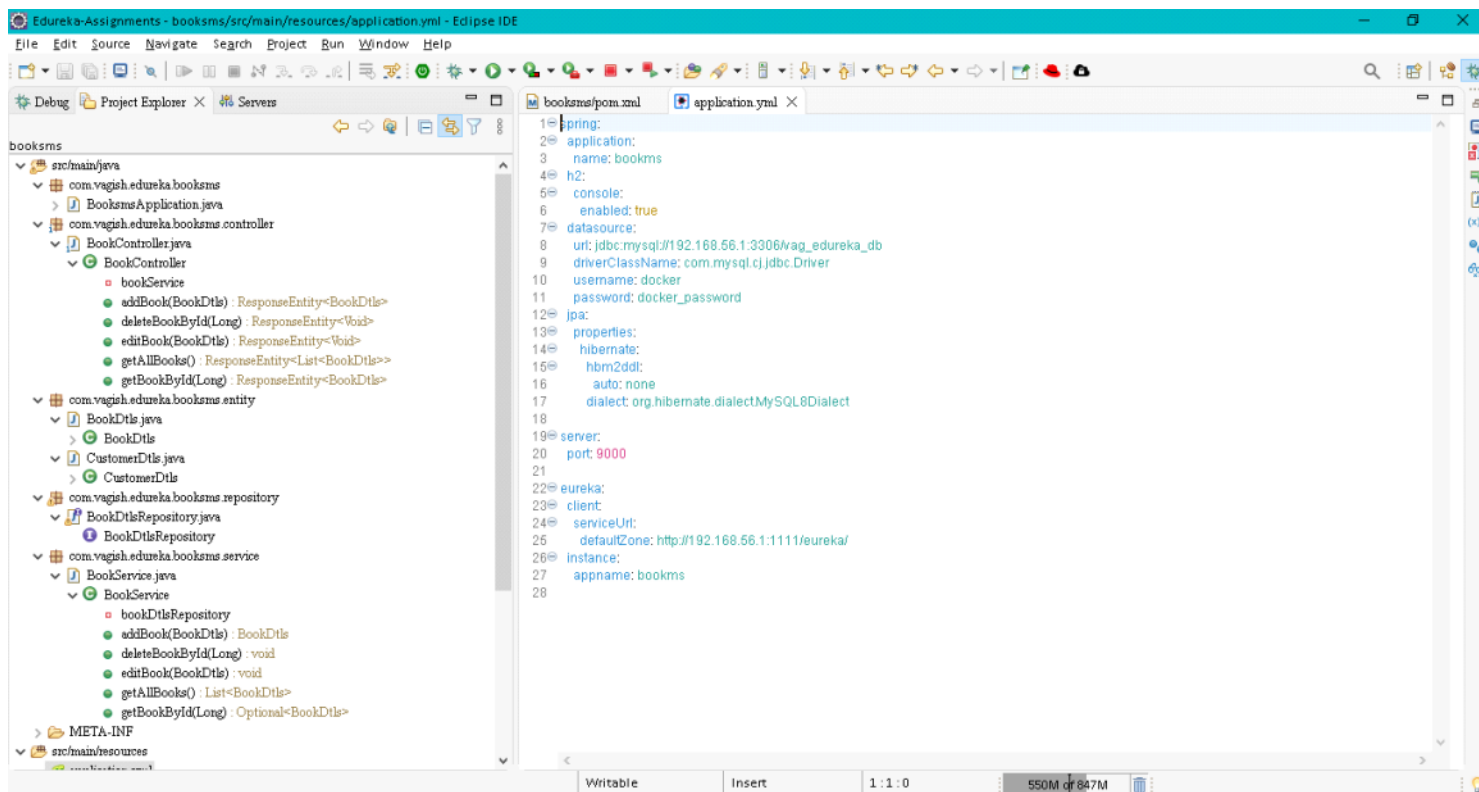
```
[{"custId":1,"book":{"isbn":1,"title":"The Alchemist","publishedDate":"1988-01-01","totalCopies":10,"issuedCopies":2,"author":"Paulo Coelho"},"noOfCopies":1}, {"custId":2,"book":{"isbn":2,"title":"The Alchemist 2","publishedDate":"1988-01-01","totalCopies":10,"issuedCopies":2,"author":"Paulo Coelho"},"noOfCopies":5}]
```

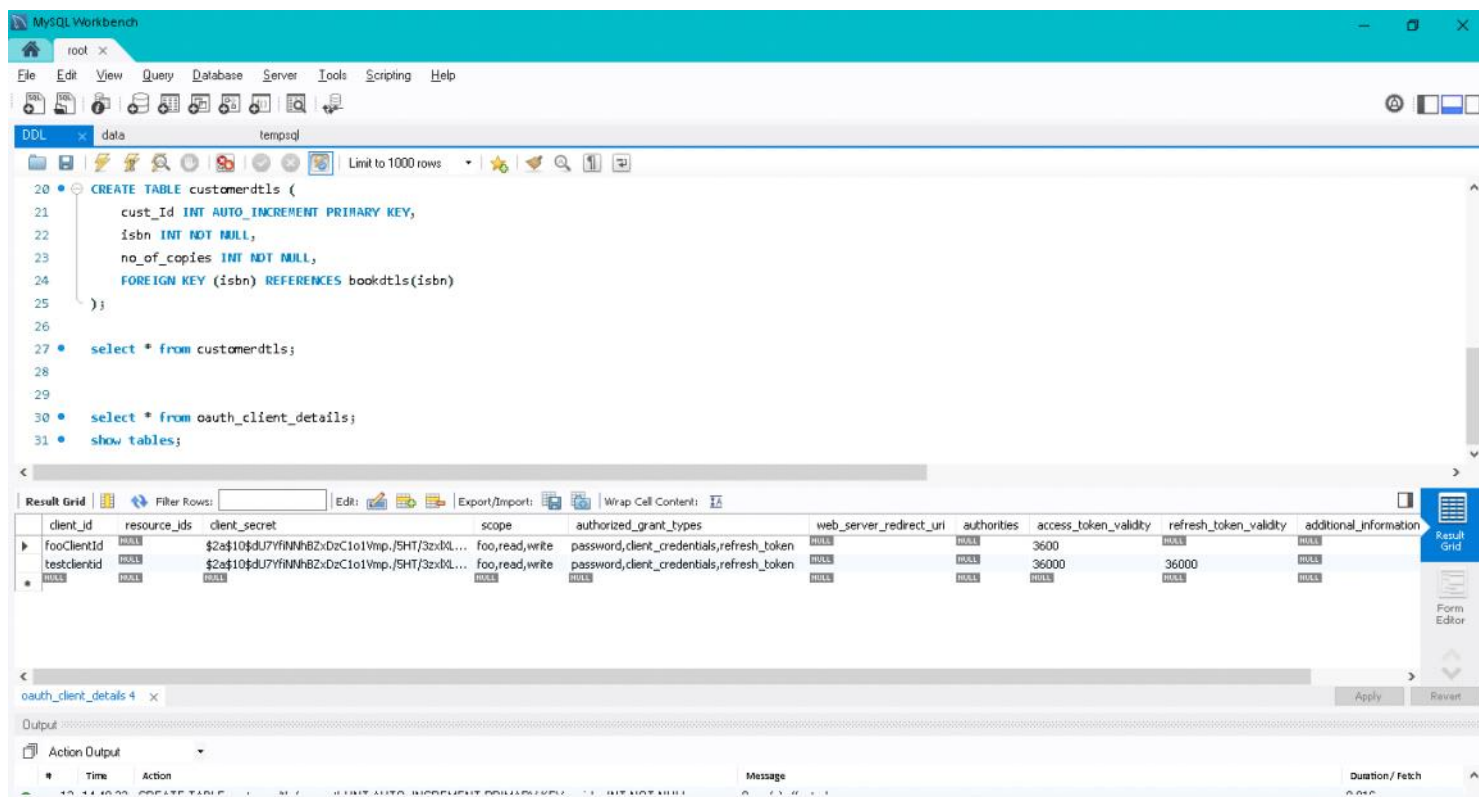
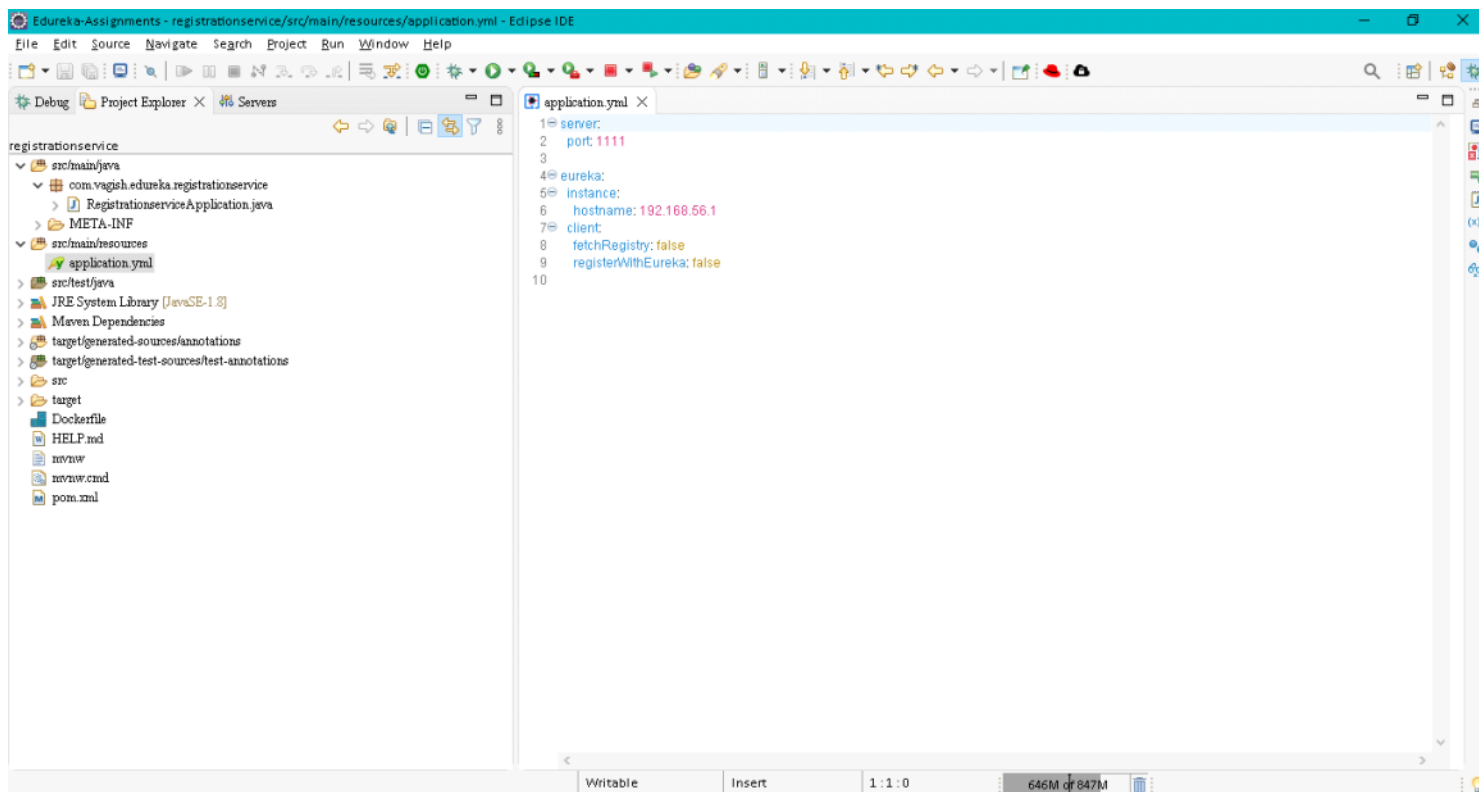


A screenshot of a web browser window with a yellow header bar. The address bar shows 'localhost:3500/books'. The page content is a dark-themed area displaying a JSON array. The browser's tab bar at the top shows four tabs: 'Eureka', 'localhost:9001/customers', 'localhost:9000/books/', and 'localhost:3500/books'.

```
[{"isbn":1,"title":"The Alchemist","publishedDate":"1988-01-01","totalCopies":10,"issuedCopies":2,"author":"Paulo Coelho"}, {"isbn":2,"title":"The Alchemist 2","publishedDate":"1988-01-01","totalCopies":10,"issuedCopies":2,"author":"Paulo Coelho"}]
```








```
I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\DDL.sql - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
DDL.sql
1
2
3 CREATE USER 'docker'@'%' IDENTIFIED BY 'docker_password';
4
5 GRANT ALL PRIVILEGES ON vag_edureka_db.* TO 'docker'@'%' ;
6
7 use vag_edureka_db;
8
9
10 drop table bookdtls;
11 CREATE TABLE bookdtls (
12     isbn INT AUTO_INCREMENT PRIMARY KEY,
13     title VARCHAR(255) NOT NULL,
14     published_date DATE NOT NULL,
15     total_copies INT NOT NULL,
16     issued_copies INT NOT NULL,
17     author VARCHAR(255) NOT NULL
18 );
19
20 drop table customerdtls;
21 CREATE TABLE customerdtls (
22     cust_id INT AUTO_INCREMENT PRIMARY KEY,
23     isbn INT NOT NULL,
24     no_of_copies INT NOT NULL,
25     FOREIGN KEY (isbn) REFERENCES bookdtls(isbn)
26 );
27
28 select * from customerdtls;
29
30 select * from oauth_client_details;
31 show tables;
```

Structured Query Language file length: 676 lines: 31 Ln: 1 Col: 1 Pos: 1 Unix (LF) UTF-8 INS

```
I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\registrationservice\Dockerfile - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
DDL.sql Dockerfile
1 # Set the base image to OpenJDK 8
2 FROM openjdk:8-jdk-alpine
3
4 # Set the working directory to /app
5 WORKDIR /app
6
7 # Copy the packaged JAR file into the container at /app
8 COPY target/registrationservice-0.0.1-SNAPSHOT.jar /app
9
10 # Expose port 1111 to the Docker host
11 EXPOSE 1111
12
13 # Run the JAR file when the container starts
14 CMD ["java", "-jar", "registrationservice-0.0.1-SNAPSHOT.jar"]
15
```

Normal text file length: 397 lines: 15 Ln: 1 Col: 1 Pos: 1 Windows (CR LF) UTF-8 INS

```
1 \Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\booksms\Dockefile - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window 2
DDL.sql Dockerfile Dockerfile Dockerfile
1 # Set the base image to OpenJDK 8
2 FROM openjdk:8-jdk-alpine
3
4 # Set the working directory to /app
5 WORKDIR /app
6
7 # Copy the packaged JAR file into the container at /app
8 COPY target/booksms-0.0.1-SNAPSHOT.jar /app
9
10 # Expose port 9000 to the Docker host
11 EXPOSE 9000
12
13 # Run the JAR file when the container starts
14 CMD ["java", "-jar", "booksms-0.0.1-SNAPSHOT.jar"]
15
```

Normal text file length: 373 lines: 15 Ln: 8 Col: 35 Pos: 209 Windows (CR LF) UTF-8 INS

```
1 \Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\issuer-custommerms\Dockefile - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window 2
DDL.sql Dockerfile Dockerfile Dockerfile
1 # Set the base image to OpenJDK 8
2 FROM openjdk:8-jdk-alpine
3
4 # Set the working directory to /app
5 WORKDIR /app
6
7 # Copy the packaged JAR file into the container at /app
8 COPY target/issuer-custommerms-0.0.1-SNAPSHOT.jar /app
9
10 # Expose port 9001 to the Docker host
11 EXPOSE 9001
12
13 # Run the JAR file when the container starts
14 CMD ["java", "-jar", "issuer-custommerms-0.0.1-SNAPSHOT.jar"]
15
```

Normal text file length: 389 lines: 15 Ln: 8 Col: 45 Pos: 219 Windows (CR LF) UTF-8 INS

Docker Commands for Project

--To Build the container image

```
docker build -t registrationservice -f Dockerfile.
```

```
docker build -t bookms -f Dockerfile.
```

```
docker build -t issuerms -f Dockerfile.
```

```
docker build -t apigateway -f Dockerfile.
```

--To Start the Docker Container

```
docker run -d -p 1111:1111 --name registrationservice-container registrationservice
```

```
docker run -d -p 9000:9000 --name bookms-container bookms
```

```
docker run -d -p 9001:9001 --name issuerms-container issuerms
```

```
docker run -d -p 3500:3600 --name apigateway-container apigateway
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. All rights reserved.

I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\registrationservice>docker build -t registrationservice -f Dockerfile .
[+] Building 1.1s (8/8) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 436B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine          1.0s
=> [internal] load build context                                                  0.0s
=> => transferring context: 95B                                                  0.0s
=> [1/3] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a0f2283343547b3 0.0s
=> CACHED [2/3] WORKDIR /app                                                     0.0s
=> CACHED [3/3] COPY target/registrationservice-0.0.1-SNAPSHOT.jar /app         0.0s
=> exporting to image                                                            0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:aeeee1a2d367e589e5c22c154f99ebdf6a1a8b2df21ed752f78e098e5178ae23 0.0s
=> => naming to docker.io/library/registrationservice                          0.0s

I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\registrationservice>docker run -d -p 1111:1111 --name registrationservice-container registrationservice
0593cc3b8ae1f941d8b14c94d1c7fdb8d19a7005d37323b1b4597473d1a95fa

I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\registrationservice>
```

```
C:\Windows\System32\cmd.exe

I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\booksms>docker build -t bookms -f Dockerfile .
[+] Building 1.1s (8/8) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 412B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine          1.0s
=> [internal] load build context                                                  0.0s
=> => transferring context: 83B                                                  0.0s
=> [1/3] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a0f2283343547b3 0.0s
=> CACHED [2/3] WORKDIR /app                                                    0.0s
=> CACHED [3/3] COPY target/booksms-0.0.1-SNAPSHOT.jar /app                    0.0s
=> exporting to image                                                            0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:06d65daffea0067a595cdd570630e23e9267783ab02c9794705a25a43234343b 0.0s
=> => naming to docker.io/library/bookms                                         0.0s

I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\booksms>docker run -d -p 9000:9000 --name bookms-container bookms
16c092936d458f35e6fe7721731544ef6a72d5a86a7e3a5145508ea2a71cdbaf

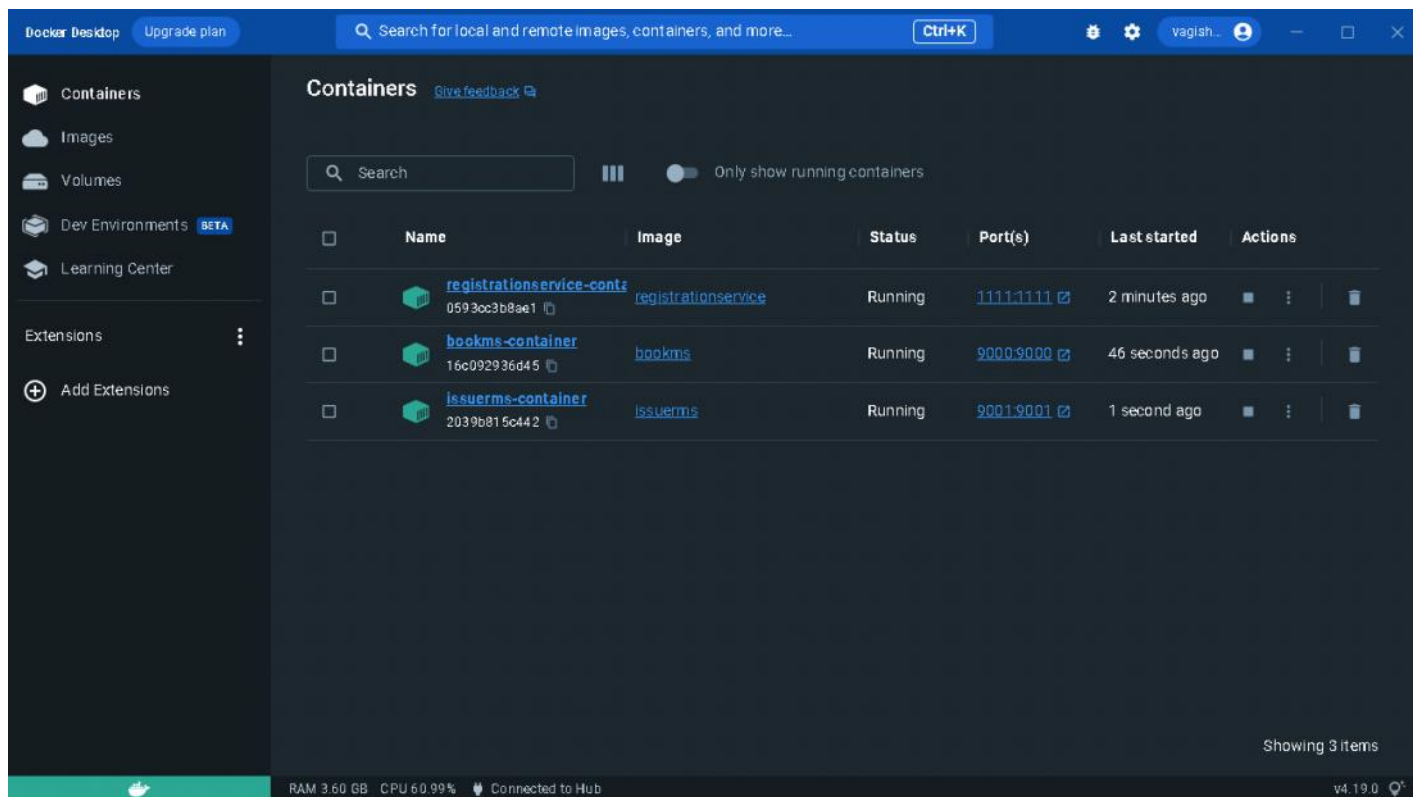
I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\booksms>
```

```
C:\Windows\System32\cmd.exe

I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\issuer-customermrs>docker build -t issuerms -f Dockerfile .
[+] Building 2.9s (8/8) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 432B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine          0.9s
=> [1/3] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a0f2283343547b3 0.0s
=> [internal] load build context                                                  0.9s
=> => transferring context: 60.23MB                                              0.9s
=> CACHED [2/3] WORKDIR /app                                                    0.0s
=> [3/3] COPY target/issuer-customermrs-0.0.1-SNAPSHOT.jar /app                0.7s
=> exporting to image                                                            0.4s
=> => exporting layers                                                            0.4s
=> => writing image sha256:88a865c0c41aca2cfa2a617dfe8dbd4231429c78f15cecb001ed761c91a9bb1 0.0s
=> => naming to docker.io/library/issuerms                                       0.0s

I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\issuer-customermrs>docker run -d -p 9001:9001 --name issuerms-container issuerms
2039b815c442b68997e08490d63fbde3aaeff463b7d49d0cf2f964d399e4ddca

I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\issuer-customermrs>
```



```

1 # Set the base image to OpenJDK 8
2 FROM openjdk:8-jdk-alpine
3
4 # Set the working directory to /app
5 WORKDIR /app
6
7 # Copy the packaged JAR file into the container at /app
8 COPY target/apigateway-0.0.1-SNAPSHOT.jar /app
9
10 # Expose port 3500:3600 to the Docker host
11 EXPOSE 3500
12
13 # Run the JAR file when the container starts
14 CMD ["java", "-jar", "apigateway-0.0.1-SNAPSHOT.jar"]
15

```

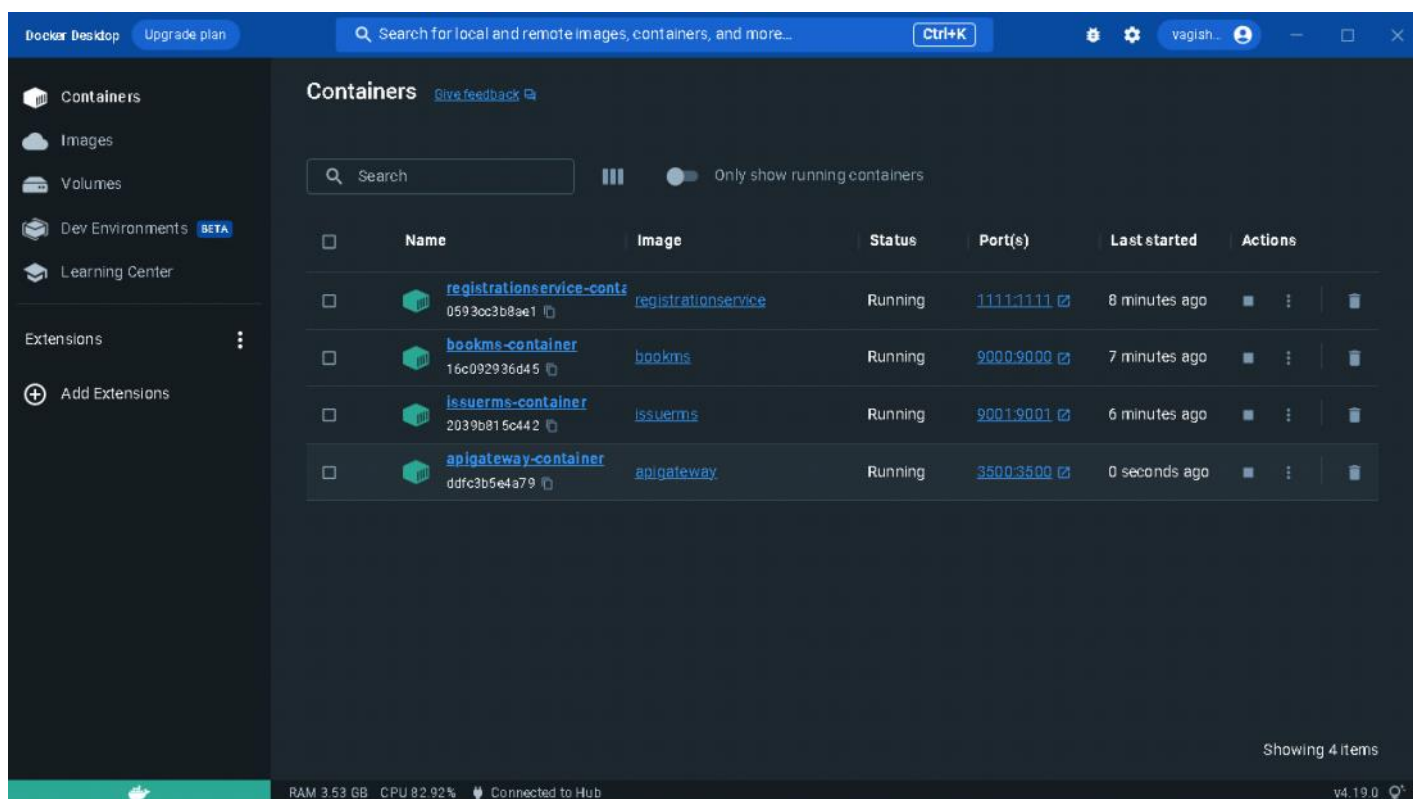
Normal text file | length: 384 | lines: 15 | Ln: 11 | Col: 12 | Pos: 280 | Windows (CR LF) | UTF-8 | INS

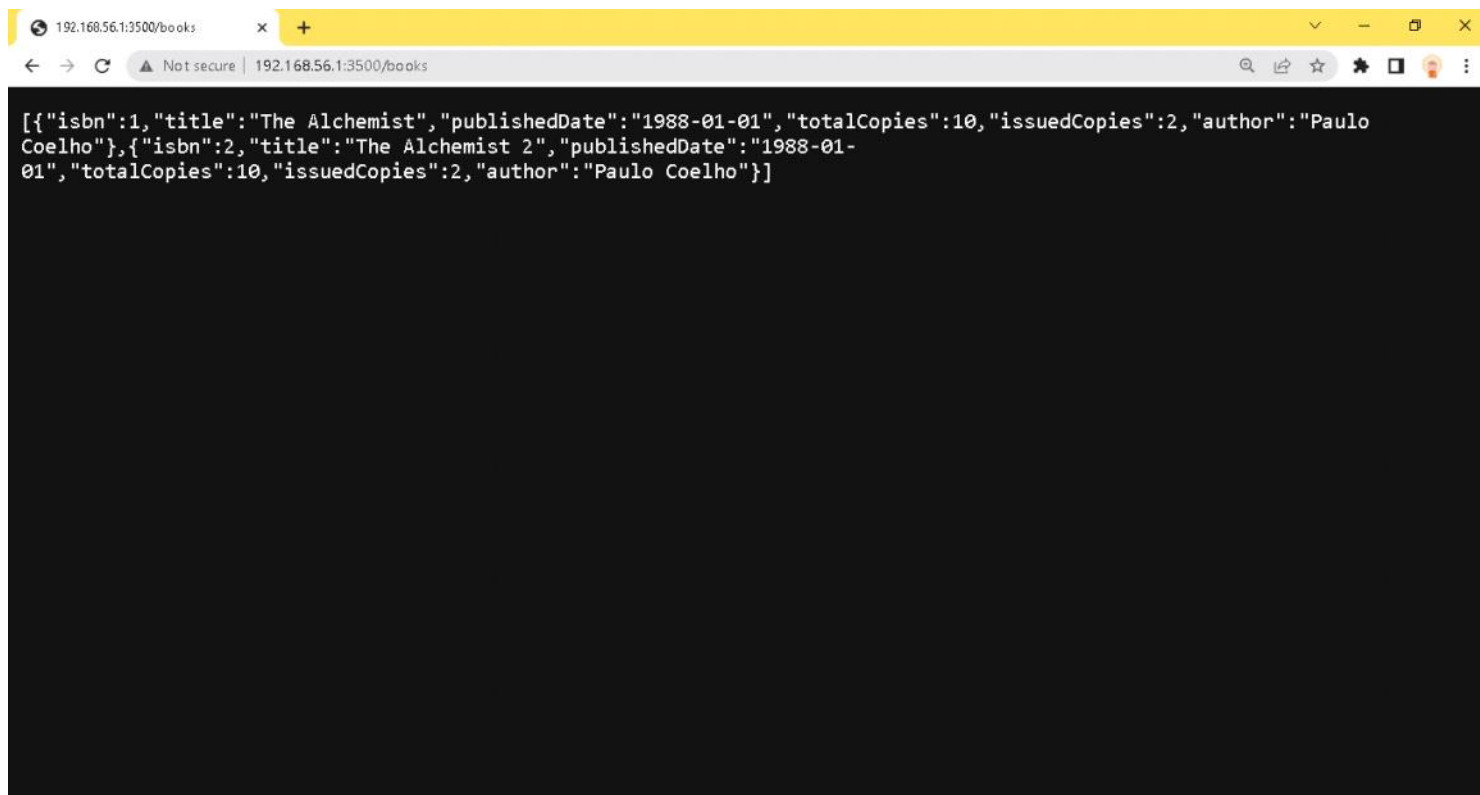

```
C:\Windows\System32\cmd.exe

I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\apigateway>docker build -t apigateway -f Dockerfile .
[+] Building 1.1s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 423B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine
=> [internal] load build context
=> => transferring context: 86B
=> [1/3] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a0f2283343547b3
=> CACHED [2/3] WORKDIR /app
=> CACHED [3/3] COPY target/apigateway-0.0.1-SNAPSHOT.jar /app
=> exporting to image
=> => exporting layers
=> => writing image sha256:7b2dcb743aea797912b9aa757ffdd86f2ebae5a70b6fb457fe4c638c82d4e256
=> => naming to docker.io/library/apigateway

I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\apigateway>docker run -d -p 3500:3500 --name apigateway-container apigateway
ddfc3b5e4a7998f946e027ae763ef9d32f1441319f8f2a6933b822dba8f937e1

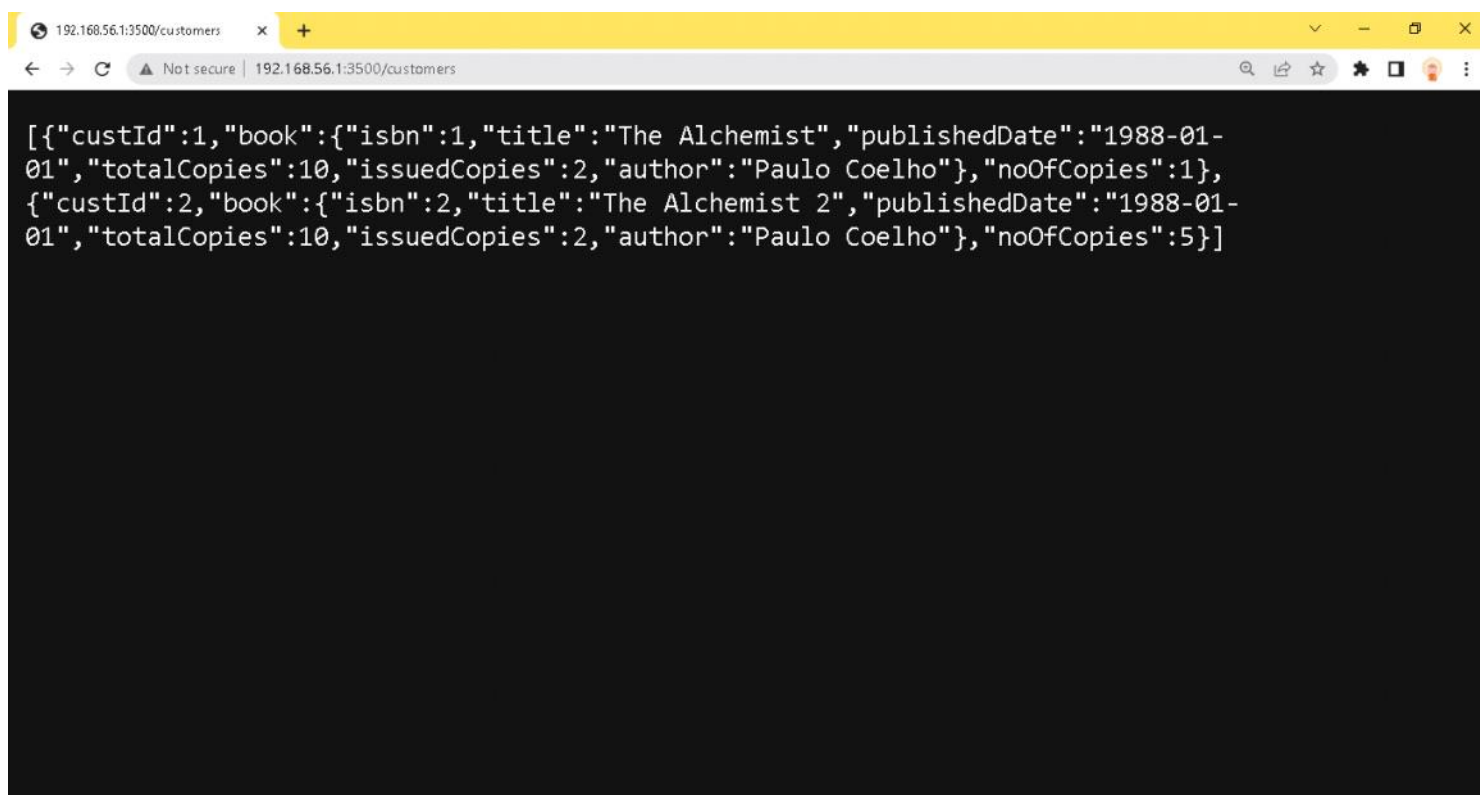
I:\Workspace\EclipseWorkspace\TempWorkspaces\Edureka-Assignments\apigateway>_
```





A screenshot of a web browser window. The address bar shows the URL "192.168.56.1:3500/books". The page content displays a JSON array of two book objects. The first object has isbn:1, title:"The Alchemist", publishedDate:"1988-01-01", totalCopies:10, issuedCopies:2, and author:"Paulo Coelho". The second object has isbn:2, title:"The Alchemist 2", publishedDate:"1988-01-01", totalCopies:10, issuedCopies:2, and author:"Paulo Coelho".

```
[{"isbn":1,"title":"The Alchemist","publishedDate":"1988-01-01","totalCopies":10,"issuedCopies":2,"author":"Paulo Coelho"}, {"isbn":2,"title":"The Alchemist 2","publishedDate":"1988-01-01","totalCopies":10,"issuedCopies":2,"author":"Paulo Coelho"}]
```



A screenshot of a web browser window. The address bar shows the URL "192.168.56.1:3500/customers". The page content displays a JSON array of two customer objects. Each object has a custId and a book object. The first customer has custId:1 and a book with isbn:1, title:"The Alchemist", publishedDate:"1988-01-01", totalCopies:10, issuedCopies:2, author:"Paulo Coelho", and noOfCopies:1. The second customer has custId:2 and a book with isbn:2, title:"The Alchemist 2", publishedDate:"1988-01-01", totalCopies:10, issuedCopies:2, author:"Paulo Coelho", and noOfCopies:5.

```
[{"custId":1,"book":{"isbn":1,"title":"The Alchemist","publishedDate":"1988-01-01","totalCopies":10,"issuedCopies":2,"author":"Paulo Coelho"},"noOfCopies":1}, {"custId":2,"book":{"isbn":2,"title":"The Alchemist 2","publishedDate":"1988-01-01","totalCopies":10,"issuedCopies":2,"author":"Paulo Coelho"},"noOfCopies":5}]
```

Uploaded PDF with consolidated Screenshots to create 5 Microservices based on the Problem Statement:

1. OAuth Security Server:8080
2. API Gateway:3500
3. Book MS: 9000
4. Issuer MS: 9001
5. Eureka Registry Service:1111

to perform all the mentioned Tasks.

Below Screenshots are attached:

1. Code - property files and Structure
2. Database - Queries and Results
3. Postman - Rest API results
4. Docker - Commands and Dashboard
5. Browser - Individual MS and API Gateway Results

Also, all the related result Screenshot PDF is saved on GIT Repo:

<https://github.com/Vagish619/Edureka-Microservices.git>

Branch Name:

Assignments_And_Projects