

## Integrantes:

- Santiago Bernal López

- Luisa Fernanda Celis Gonzalez

- Andrés Mauricio Ocampo Arenas

Github Repositorio -> [Z4ND3RX/Microservices-practice \(github.com\)](https://github.com/Z4ND3RX/Microservices-practice)

## PROYECTO MONOLITICO – BIBLIOTECA

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "Project". It includes a "build" folder, a "gradle" folder, a "src" folder containing "main" (with "java", "resources", and "test" subfolders), and a "build" folder under "test".
- Code Editor:** Displays the file "ClienteController.java". The code implements a controller for managing clients. It includes methods for creating, reading, updating, and deleting clients, as well as a method for deleting by ID.
- Terminal:** Shows the command line output of a "gradlew bootRun" command. It includes logs from Spring Boot, Tomcat, and MySQL, indicating the application is running on port 8080 and the database is ready for connections.
- Docker Compose:** A "docker-compose.yml" file is visible in the top right corner.
- Services:** A "Docker" tab in the bottom left shows a network diagram with three services: "BibliotecaApplication", "LibroService", and "PrestamoService".

## Docker-compose up

The terminal output shows the logs for the "BibliotecaApplication" service as it starts up. It includes logs from MySQL, Spring Boot, Tomcat, and Hibernate, detailing the initialization of the application, the embedded MySQL database, and the configuration of JPA repositories.

```
db | 2024-03-20T03:59:41.396727Z [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.34' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.
db |
app | :: Spring Boot ::          (v3.0.2)
app | 2024-03-20T03:59:44.343Z INFO 1 --- [           main] c.e.e.biblioteca.BibliotecaApplication : Starting BibliotecaApplication using Java 17.0.10 with PID 1 (/biblioteca.jar started by root in /)
app | 2024-03-20T03:59:44.362Z INFO 1 --- [           main] c.e.e.biblioteca.BibliotecaApplication : No active profile set, falling back to 1 default profile: "default"
app | 2024-03-20T03:59:45.929Z INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
app | 2024-03-20T03:59:46.032Z INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 93 ms. Found 4 JPA repository interfaces.
app | 2024-03-20T03:59:47.158Z INFO 1 --- [           main] o.s.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
app | 2024-03-20T03:59:47.173Z INFO 1 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
app | 2024-03-20T03:59:47.174Z INFO 1 --- [           main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.5]
app | 2024-03-20T03:59:47.174Z INFO 1 --- [           main] o.s.w.context.ContextLoader : Root WebApplicationContext: initialization completed in 2776 ms
app | 2024-03-20T03:59:47.274Z INFO 1 --- [           main] o.s.c.support.webapp.WebApplicationContext : HHH000204: Processing PersistenceUnitInfo [name: default]
app | 2024-03-20T03:59:47.536Z INFO 1 --- [           main] o.hibernate.jpa.internal.util.LogHelper : HHH000412: Hibernate ORM core version 6.1.6.Final
app | 2024-03-20T03:59:47.606Z INFO 1 --- [           main] org.hibernate.Version : HHH000021: Encountered deprecated setting [javax.persistence.sharedCache.mode], use [jakarta.persistence.sharedCache.mode] instead
app | 2024-03-20T03:59:47.964Z WARN 1 --- [           main] org.hibernate.orm.deprecation : HHH000021: Encountered deprecated setting [javax.persistence.sharedCache.mode], use [jakarta.persistence.sharedCache.mode] instead
app | 2024-03-20T03:59:48.151Z INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
app | 2024-03-20T03:59:49.314Z INFO 1 --- [           main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@2e1792e7
app | 2024-03-20T03:59:49.334Z INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
app | 2024-03-20T03:59:49.516Z INFO 1 --- [           main] SQL dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
app | 2024-03-20T03:59:51.718Z INFO 1 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH0000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
app | 2024-03-20T03:59:51.738Z INFO 1 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
app | 2024-03-20T03:59:53.871Z WARN 1 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
app | 2024-03-20T03:59:55.031Z INFO 1 --- [           main] o.s.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
app | 2024-03-20T03:59:55.078Z INFO 1 --- [           main] c.e.e.biblioteca.BibliotecaApplication : Started BibliotecaApplication in 12.119 seconds (process running for 13.964)
app | 2024-03-20T04:00:22.099Z INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
app | 2024-03-20T04:00:22.100Z INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
app | 2024-03-20T04:00:22.108Z INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 7 ms
app | Hibernate: select c1_0.codigo,c1_0.email,c1_0.nombre,c1_0.password,c1_0.telefono from cliente c1_0
app | Hibernate: insert into cliente (email,nombre,password,telefono) values (?, ?, ?, ?)
app | Hibernate: select l1_0.isbn,l1_0.fecha_publicacion,l1_0.genero,l1_0.nombre,l1_0.unidades from libro l1_0 where l1_0.isbn=?
app | Hibernate: select l1_0.isbn,l1_0.fecha_publicacion,l1_0.genero,l1_0.nombre,l1_0.unidades from libro l1_0 where l1_0.isbn=?
app | Hibernate: select l1_0.isbn,l1_0.fecha_publicacion,l1_0.genero,l1_0.nombre,l1_0.unidades from libro l1_0
app | Hibernate: select l1_0.id,l1_0.nombre from autor l1_0 where l1_0.id in(?)
app | Hibernate: select l1_0.isbn,l1_0.fecha_publicacion,l1_0.genero,l1_0.nombre,l1_0.unidades from libro l1_0 where l1_0.isbn=?
```

## CLIENT – PRUEBAS HTTP

### MÉTODO POST

The screenshot shows the Postman interface for a POST request to `http://localhost:8080/api/cliente`. The request body is set to `JSON` and contains the following data:

```
1 {
2     "nombre": "Andres",
3     "email": "andres@gmail.com",
4     "telefono": "3176788278",
5     "password": "helado444"
6 }
```

The response body is also in `JSON` format:

```
1 {
2     "mensaje": "Cliente creado correctamente",
3     "dato": [
4         {
5             "codigo": 1,
6             "nombre": "Andres",
7             "email": "andres@gmail.com",
8             "telefono": "3176788278"
9         }
10    ]
11 }
```

### MÉTODO GET

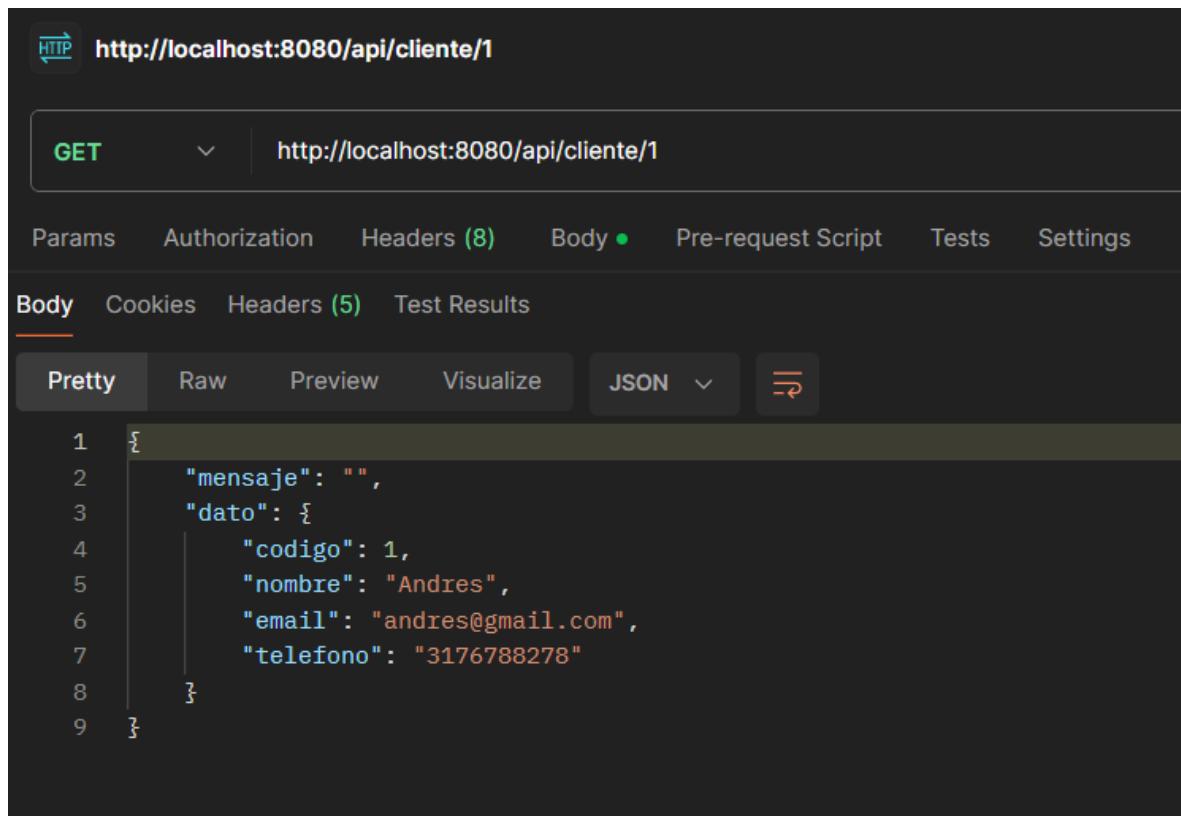
The screenshot shows the Postman interface for a GET request to `http://localhost:8080/api/cliente`. The request body is set to `JSON` and contains the following data:

```
1 {
2     "nombre": "Andres",
3     "email": "andres@gmail.com",
4     "telefono": "3176788278",
5     "password": "helado444"
6 }
```

The response body is in `JSON` format:

```
1 {
2     "mensaje": "",
3     "dato": [
4         {
5             "codigo": 1,
6             "nombre": "Andres",
7             "email": "andres@gmail.com",
8             "telefono": "3176788278"
9         }
10    ]
11 }
```

## MÉTODO GET BY ID



HTTP <http://localhost:8080/api/cliente/1>

GET <http://localhost:8080/api/cliente/1>

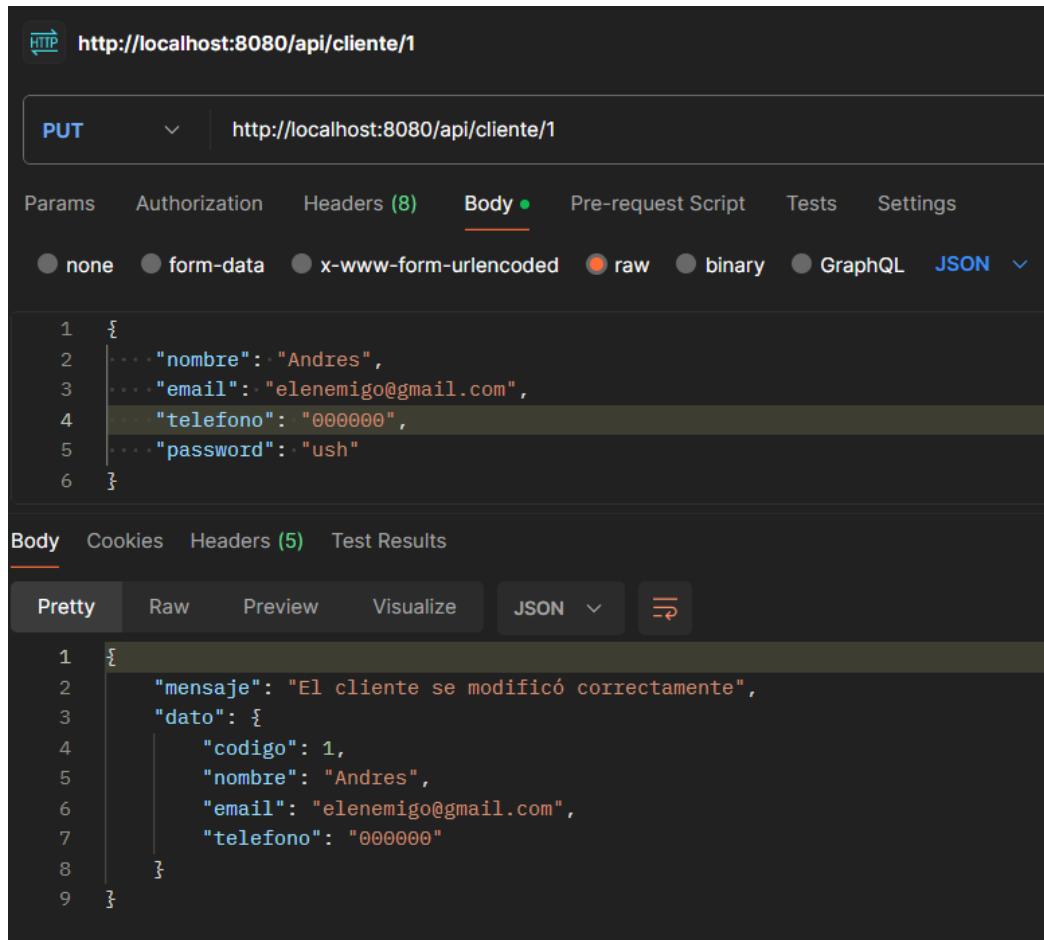
Params Authorization Headers (8) Body • Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 {
2     "mensaje": "",
3     "dato": {
4         "codigo": 1,
5         "nombre": "Andres",
6         "email": "andres@gmail.com",
7         "telefono": "3176788278"
8     }
9 }
```

## MÉTODO PUT



HTTP <http://localhost:8080/api/cliente/1>

PUT <http://localhost:8080/api/cliente/1>

Params Authorization Headers (8) Body • Pre-request Script Tests Settings

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

```
1 {
2     "nombre": "Andres",
3     "email": "elenemigo@gmail.com",
4     "telefono": "000000",
5     "password": "ush"
6 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 {
2     "mensaje": "El cliente se modificó correctamente",
3     "dato": {
4         "codigo": 1,
5         "nombre": "Andres",
6         "email": "elenemigo@gmail.com",
7         "telefono": "000000"
8     }
9 }
```

## MÉTODO DELETE

The screenshot shows the Postman interface with a successful DELETE request. The URL is `http://localhost:8080/api/cliente/1`. The response body is:

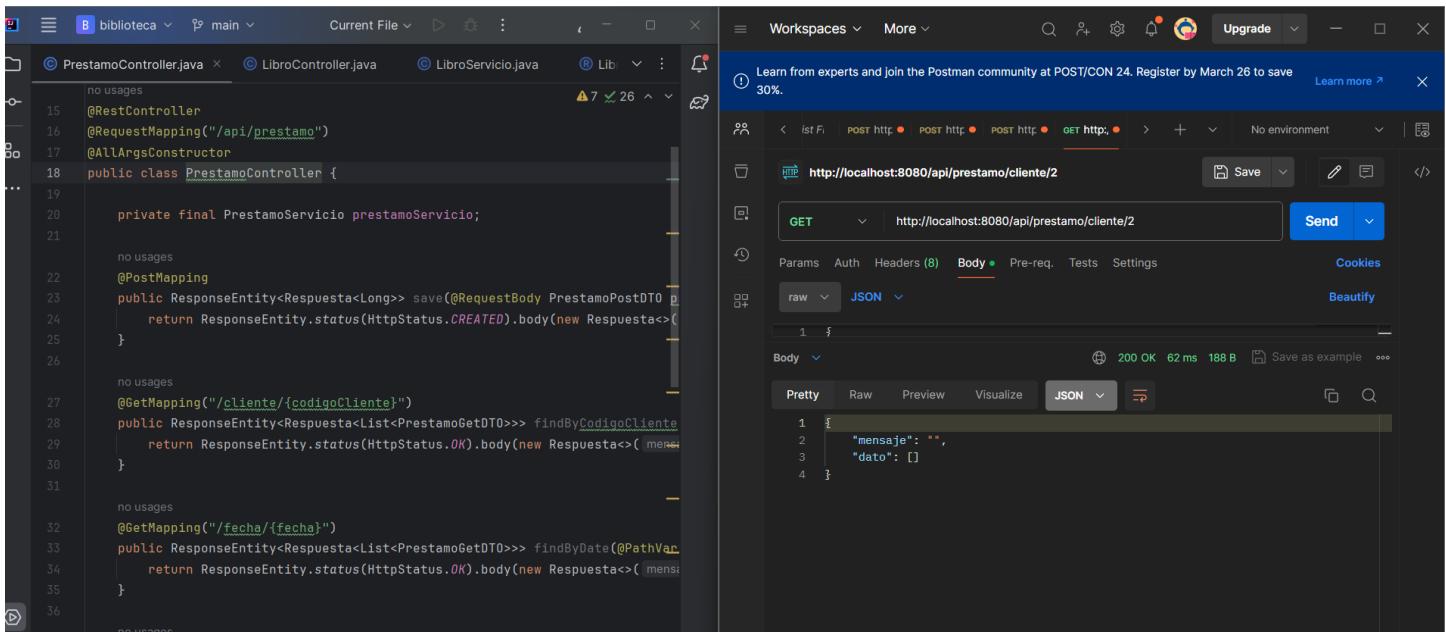
```
1 {  
2   "mensaje": "Se eliminó correctamente",  
3   "dato": null  
4 }
```

## LIBROS Y PRESTAMOS FUNCIONAN DE MANERA SIMILAR

The screenshot shows the VS Code editor and the Postman application side-by-side. The code editor displays the `LibroServicio.java` file, which contains methods for saving and finding books by ISBN. The Postman tab shows a successful GET request to `http://localhost:8080/api/libro`. The response body is:

```
1 {  
2   "isbn": "121212",  
3   "nombre": "patito feo",  
4   "genero": 1,  
5   "unidades": 4,  
6   "fechaPublicacion": "2023-04-04",  
7   "idAutores": [1]  
8 }
```

The Postman interface also shows the status `200 OK`, `35 ms`, and `188 B`.



## 2. PROYECTO MICROSERVICIOS – BIBLIOTECA

Realizamos unos cuantos ajustes en el Docker compose como los puertos:

```

49
50 >   clientes:
51     image: bib_clientes
52     build: ./clientes
53     restart: always
54     ports:
55       - "8082:8082"
56     depends_on:
57       - mongo_database
58     environment:
59       spring.data.mongodb.uri: mongodb://root@example@mongo_database:27017/clientes?authSource=admin
60
61 >   libros:
62     image: bib_libros
63     build: ./libros
64     restart: always
65     ports:
66       - "8083:8083"
67     depends_on:
68       - mysql_database_1
69     environment:
70       spring.datasource.url: jdbc:mysql://mysql_database_1:3306/libros?createDatabaseIfNotExist=true
71       spring.datasource.username: root
72       spring.datasource.password: root1234
73

```

## Abrimos y corremos el proyecto

```

Project
  biblioteca-microservicios [biblioteca] C:\Users\USUARIO\Desktop\biblioteca-microservicios
    .gradle
    clients
    gradle
    libros
    prestamos
      .gitignore
      build.gradle
      docker-compose.yml
      gradlew
      gradlew.bat
      settings.gradle
  External Libraries
  Scratches and Consoles

Current File docker-compose.yml
  18   "27017:27017"
  19

  MINGW64/c/Users/USUARIO/Desktop/biblioteca-microservicios
prestamos-1 | 2024-03-20T04:19:22.178Z INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost]           : Initializing Spring embedded
prestamos-1 | 2024-03-20T04:19:22.362Z INFO 1 --- [           main] m.a.s.S.standard.ServerApplicationContext : Root WebApplicationContext: initialized at : HHH000400: Using dialect: org
libros-1   | 2024-03-20T04:19:22.732Z INFO 1 --- [           main] SQL dialect
prestamos-1 | 2024-03-20T04:19:22.841Z INFO 1 --- [           main] org.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing Persist
prestamos-1 | 2024-03-20T04:19:22.841Z INFO 1 --- [           main] org.hibernate.Version                : HHH000412: Hibernate ORM core
mongo_database-1 | {"t": {"$date": "2024-03-20T04:19:23.225+00:00"}, "s": "I", "e": "NETWORK", "id": 22943, "ctx": "listener", "msg": "Connection accep
mongo_database-1 | {"t": {"$date": "2024-03-20T04:19:23.230+00:00"}, "s": "I", "e": "NETWORK", "id": 22943, "ctx": "listener", "msg": "Connection accep
mongo_database-1 | {"t": {"$date": "2024-03-20T04:19:23.290+00:00"}, "s": "I", "e": "NETWORK", "id": 51800, "ctx": "conn4", "msg": "client metadata", "a
dCompressors": [], "doc": {"name": "mongo-java-driver"}, "connectionId": 4, "connectionCount": 3}
mongo_database-1 | {"t": {"$date": "2024-03-20T04:19:23.301+00:00"}, "s": "I", "e": "NETWORK", "id": 51800, "ctx": "conn5", "msg": "client metadata", "a
dCompressors": [], "doc": {"name": "mongo-java-driver"}, "connectionId": 5, "connectionCount": 4}
mongo_database-1 | {"t": {"$date": "2024-03-20T04:19:23.301+00:00"}, "s": "I", "e": "NETWORK", "id": 51800, "ctx": "conn5", "msg": "client metadata", "a
orm": "Java/Eclipse Adoption/17.0_107"]
prestamos-1 | 2024-03-20T04:19:23.306Z WARN 1 --- [           main] org.hibernate.orm.deprecation : HHH00000021: Encountered depr
arta.persistence.sharedCache.mode instead
clientes-1   | 2024-03-20T04:19:23.352Z INFO 1 --- [__database:27017] org.mongodb.driver.cluster : Monitor thread successfully c
= mongo_database:27017, type=STANDALONE, state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=21, logicalSessionTimeout
prestamos-1 | 2024-03-20T04:19:23.526Z INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
prestamos-1 | 2024-03-20T04:19:24.059Z INFO 1 --- [           main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connecti
prestamos-1 | 2024-03-20T04:19:24.064Z INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start complete
prestamos-1 | 2024-03-20T04:19:24.241Z INFO 1 --- [           main] SQL dialect
libros-1   | 2024-03-20T04:19:24.876Z INFO 1 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000400: Using JtaPlatform
m.internal.NoJataPlatform]
libros-1   | 2024-03-20T04:19:24.893Z INFO 1 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManager
libros-1   | 2024-03-20T04:19:25.693Z WARN 1 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is en
d during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
libros-1   | 2024-03-20T04:19:26.591Z INFO 1 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : Tomcat started on port(s): 80
libros-1   | 2024-03-20T04:19:26.626Z INFO 1 --- [           main] co.edu.eam.biblioteca.ProductoApp : Started ProductoApp in 11.28
prestamos-1 | 2024-03-20T04:19:26.686Z INFO 1 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000400: Using JtaPlatform
m.internal.NoJataPlatform]
prestamos-1 | 2024-03-20T04:19:26.708Z INFO 1 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManager
mongo_database-1 | {"t": {"$date": "2024-03-20T04:19:26.756+00:00"}, "s": "I", "e": "NETWORK", "id": 22943, "ctx": "listener", "msg": "Connection accep
id": "36612b8c-620d-46c2-a11-3b45c03e5511", "connectionId": 6, "connectionCount": 5}
mongo_database-1 | {"t": {"$date": "2024-03-20T04:19:26.757+00:00"}, "s": "I", "e": "NETWORK", "id": 51800, "ctx": "conn6", "msg": "client metadata", "a
dCompressors": [], "doc": {"name": "nodejs"}, "version": "4.13.0"}, "os": {"type": "linux", "name": "linux", "architecture": "x64", "version": "5.15.133.1-nified}
prestamos-1 | 2024-03-20T04:19:28.319Z WARN 1 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is en
d during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
prestamos-1 | 2024-03-20T04:19:28.862Z INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 80
prestamos-1 | 2024-03-20T04:19:28.886Z INFO 1 --- [           main] co.edu.eam.biblioteca.PrestamoApp : Started PrestamoApp in 11.645

```

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	 libros-1 8c99c0cb9b92	<a href="#">bib_libros</a>	Running	0.28%	<a href="#">8083:8083</a>	22 minutes ago	  
<input type="checkbox"/>	 prestamos-1 efeb4af663ec	<a href="#">bib_prestamos</a>	Running	0.13%	<a href="#">8084:8084</a>	22 minutes ago	  
<input type="checkbox"/>	 clientes-1 34807abb965c	<a href="#">bib_clientes</a>	Running	0.12%	<a href="#">8082:8082</a>	43 minutes ago	  
<input type="checkbox"/>	 mongo-express-1 4ebca8ccb333	<a href="#">mongo-express</a>	Running	0%	<a href="#">8081:8081</a>	22 minutes ago	  

Showing 11 items

biblioteca-microservicios

C:\Users\USUARIO\Desktop\biblioteca-microservicios

3307:3306 ↗

Service	Status	Port
biblioteca-micros...	Running	8083:8083 ↗
biblioteca-micros...	Running	8084:8084 ↗
biblioteca-micros...	Running	8082:8082 ↗
biblioteca-micros...	Running	8081:8081 ↗
biblioteca-micros...	Running	8084:8084 ↗
biblioteca-micros...	Running	27017:27017 ↗
biblioteca-micros...	Running	8082:8082 ↗

```

2024-03-19 23:19:22 prestamos-1 | 2024-03-20T04:19:22.000Z INFO 1 --- [main] o.apache.catalina.core.S
standardEngine : Starting Servlet engine: [Apache Tomcat/10.1.5]
2024-03-19 23:19:22 prestamos-1 | 2024-03-20T04:19:22.178Z INFO 1 --- [main] o.a.c.c.C.[Tomcat].[loca
lhost].[] : Initializing Spring embedded WebApplicationContext
2024-03-19 23:19:22 prestamos-1 | 2024-03-20T04:19:22.183Z INFO 1 --- [main] w.s.c.ServletWebServerAp
plicationContext : Root WebApplicationContext: initialization completed in 3726 ms
2024-03-19 23:19:22 prestamos-1 | 2024-03-20T04:19:22.732Z INFO 1 --- [main] o.hibernate.jpa.internal
.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2024-03-19 23:19:22 prestamos-1 | 2024-03-20T04:19:22.841Z INFO 1 --- [main] org.hibernate.Version
: HHH000412: Hibernate ORM core version 6.1.6.Final
2024-03-19 23:19:23 prestamos-1 | 2024-03-20T04:19:23.306Z WARN 1 --- [main] org.hibernate.orm.deprec
ation : HHH00000021: Encountered deprecated setting [javax.persistence.sharedCache.mode], use [jakarta.persist
ence.sharedCache.mode] instead
2024-03-19 23:19:23 prestamos-1 | 2024-03-20T04:19:23.526Z INFO 1 --- [main] com.zaxxer.hikari.Hikari
DataSource : HikariPool-1 - Starting...
2024-03-19 23:19:24 prestamos-1 | 2024-03-20T04:19:24.059Z INFO 1 --- [main] com.zaxxer.hikari.pool.H
ikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@6b410923
2024-03-19 23:19:24 prestamos-1 | 2024-03-20T04:19:24.064Z INFO 1 --- [main] com.zaxxer.hikari.Hikari
DataSource : HikariPool-1 - Start completed.
2024-03-19 23:19:24 prestamos-1 | 2024-03-20T04:19:24.241Z INFO 1 --- [main] SQL dialect
: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
2024-03-19 23:19:26 prestamos-1 | 2024-03-20T04:19:26.686Z INFO 1 --- [main] o.h.e.t.j.p.i.JtaPlatform
mInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.N
oJtaPlatform]
2024-03-19 23:19:26 prestamos-1 | 2024-03-20T04:19:26.708Z INFO 1 --- [main] j.LocalContainerEntityMa
nagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-03-19 23:19:28 prestamos-1 | 2024-03-20T04:19:28.319Z WARN 1 --- [main] JpaBaseConfiguration$Jpa
WebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during vie
w rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2024-03-19 23:19:28 prestamos-1 | 2024-03-20T04:19:28.862Z INFO 1 --- [main] o.s.b.w.embedded.tomcat.
TomcatWebServer : Tomcat started on port(s): 8082 (http) with context path ''
2024-03-19 23:19:28 prestamos-1 | 2024-03-20T04:19:28.866Z INFO 1 --- [main] co.edu.eam.biblioteca.Pr
estamoApp : Started PrestamoApp in 11.645 seconds (process running for 12.96)

```

Mediante la estructura y las pruebas, identificamos que es el mismo proyecto, solo que en microservicios

## GET LIBROS

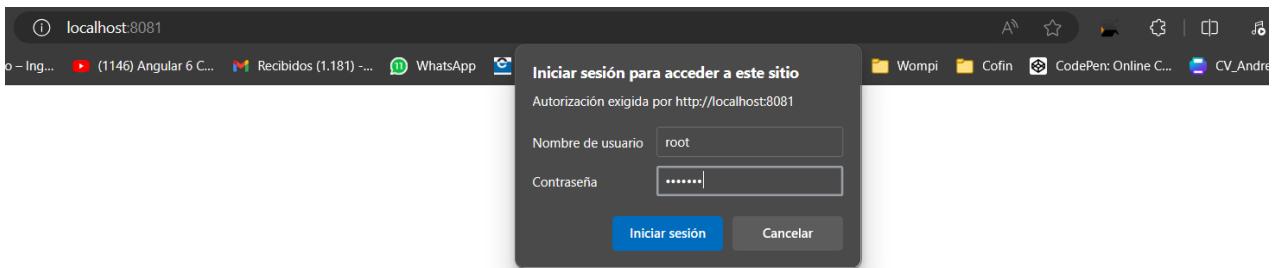
The screenshot shows a code editor with a Java controller named `LibroController` and a corresponding `docker-compose.yml` file. The `LibroController` contains methods for listing books and getting a specific book by ISBN. The `docker-compose.yml` file defines services for `biblioteca`, `prestamos`, and `mongo`. To the right, a Postman interface is open, showing a GET request to `http://localhost:8083/api/libro`. The response body is displayed in JSON format, showing a key-value pair where `mensaje` is an empty string and `dato` is an empty array.

```

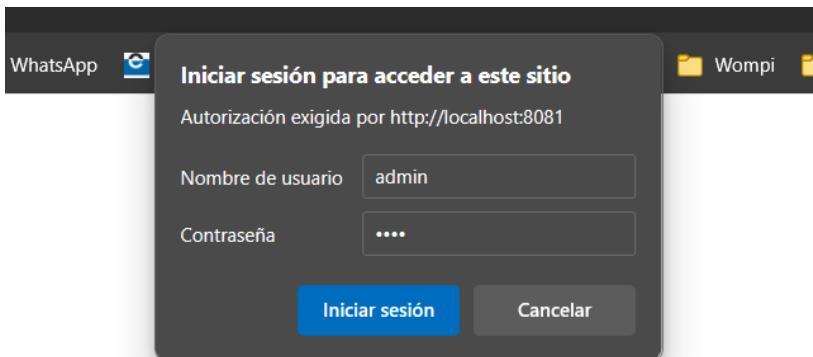
15 @RequestMapping("/api/libro")
16 @AllArgsConstructor
17 public class LibroController {
18     private final LibroService libroService;
19
20     no usages
21     @PostMapping
22     public ResponseEntity<Libro> crearLibro(@RequestBody Libro libro) {
23         return ResponseEntity.ok(libroService.guardar(libro));
24     }
25
26     no usages
27     @GetMapping
28     public ResponseEntity<List<Libro>> obtenerTodosLosLibros() {
29         return ResponseEntity.ok(libroService.listar());
30     }
31
32     no usages
33     @GetMapping("/{isbnLibro}")
34     public ResponseEntity<Libro> obtenerLibroPorISBN(@PathVariable String isbnLibro) {
35         return ResponseEntity.ok(libroService.buscarPorISBN(isbnLibro));
36     }
37 }

```

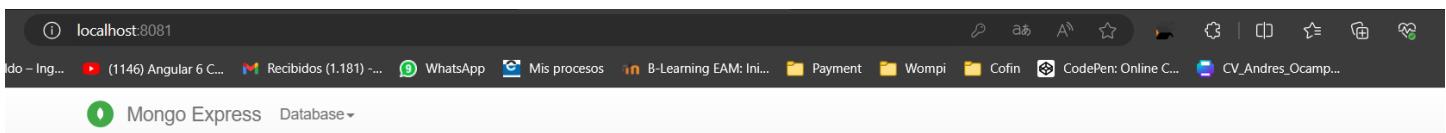
## Base de datos



Bueno, no es, pero esta si es 😊



Allí, encontramos el panel de administración de mongo express



### Databases

Database Name

+ Create Database

View	admin	Del
View	config	Del
View	local	Del

### Server Status

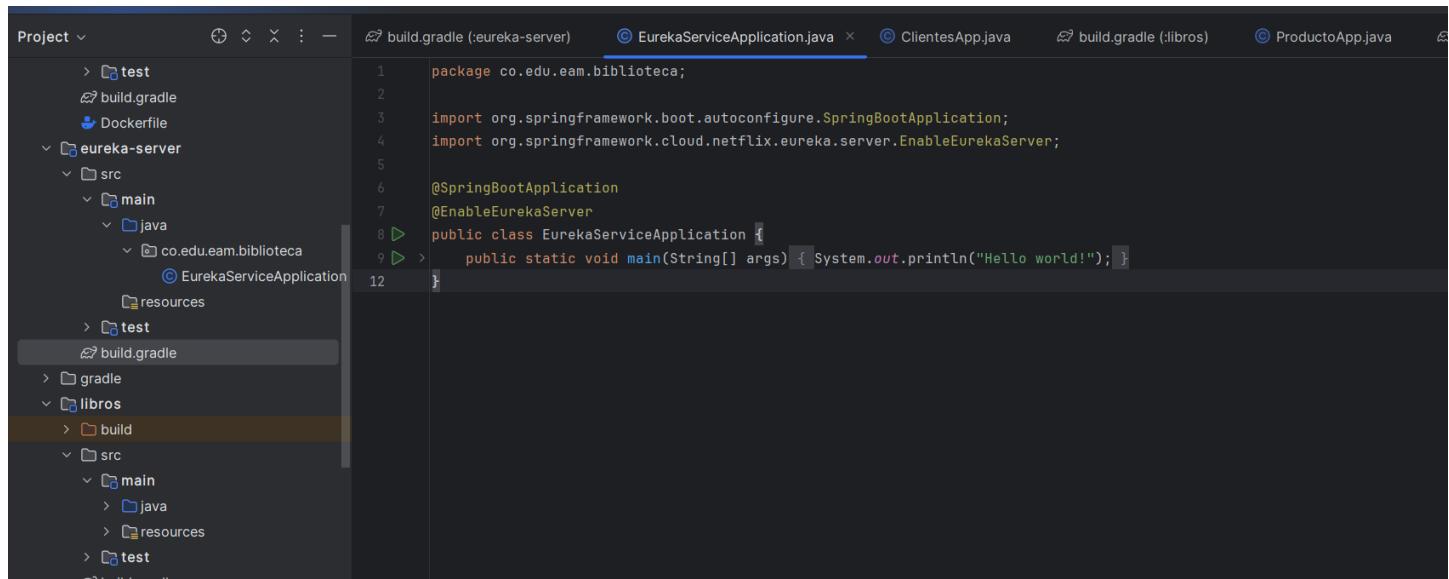
Hostname	c98e5ed90ef5	MongoDB Version	7.0.6
Uptime	120 seconds	Node Version	18.19.1
Server Time	Thu, 21 Mar 2024 04:23:28 GMT	V8 Version	10.2.154.26-node.28
Current Connections	5	Available Connections	838855
Active Clients	0	Queued Operations	0

RestTemplate es una clase proporcionada por el framework Spring para realizar peticiones HTTP a servicios RESTful. Facilita la comunicación entre aplicaciones Java y servicios web que siguen el estilo de arquitectura REST.

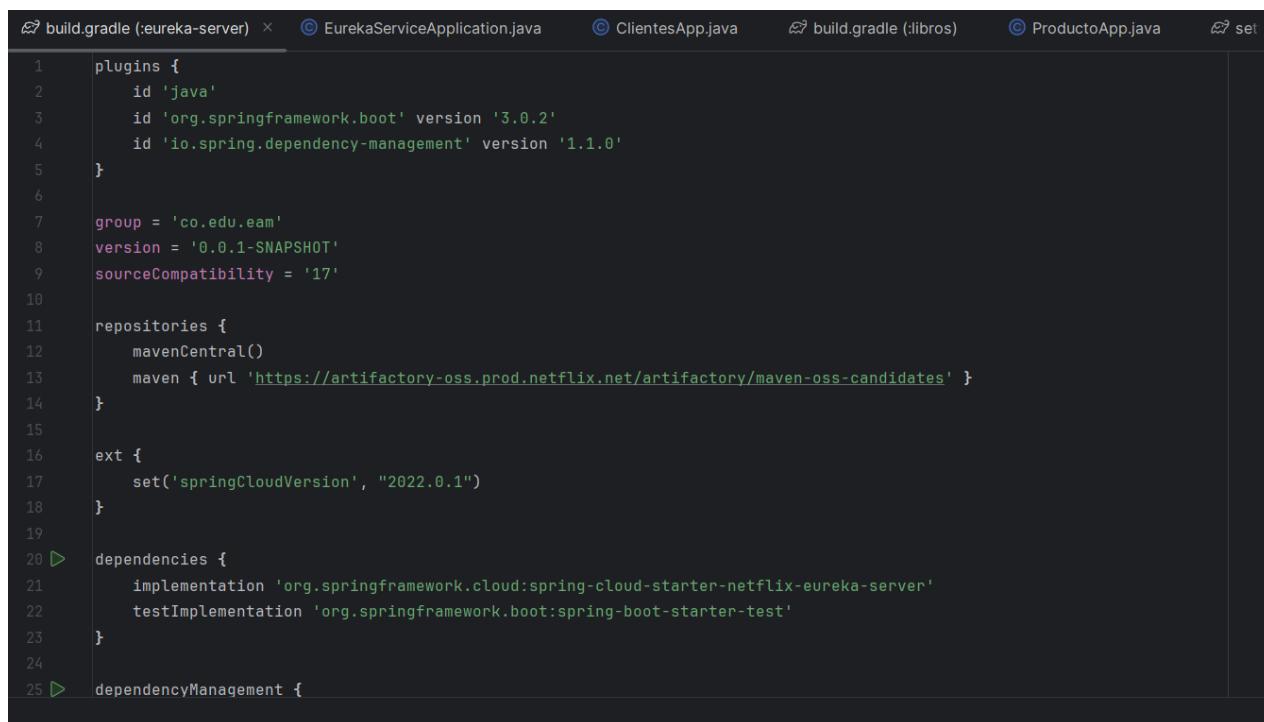
Con RestTemplate, puedes enviar solicitudes HTTP GET, POST, PUT, DELETE, y procesar las respuestas recibidas. Proporciona métodos convenientes para manejar los diferentes tipos de datos que se pueden enviar y recibir en una solicitud HTTP, como JSON, XML, texto simple, etc.

Fuente - Tutorial: [A Guide to the RestTemplate | Baeldung](#)

## EUREKA-

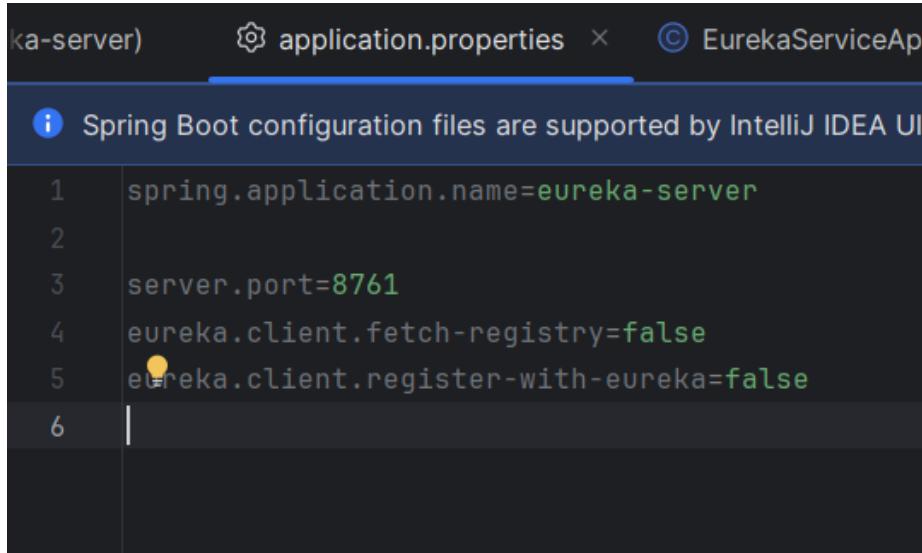


```
1 package co.edu.eam.biblioteca;
2
3 import org.springframework.boot.autoconfigure.SpringBootApplication;
4 import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
5
6 @SpringBootApplication
7 @EnableEurekaServer
8 public class EurekaServiceApplication {
9     public static void main(String[] args) { System.out.println("Hello world!"); }
10 }
```



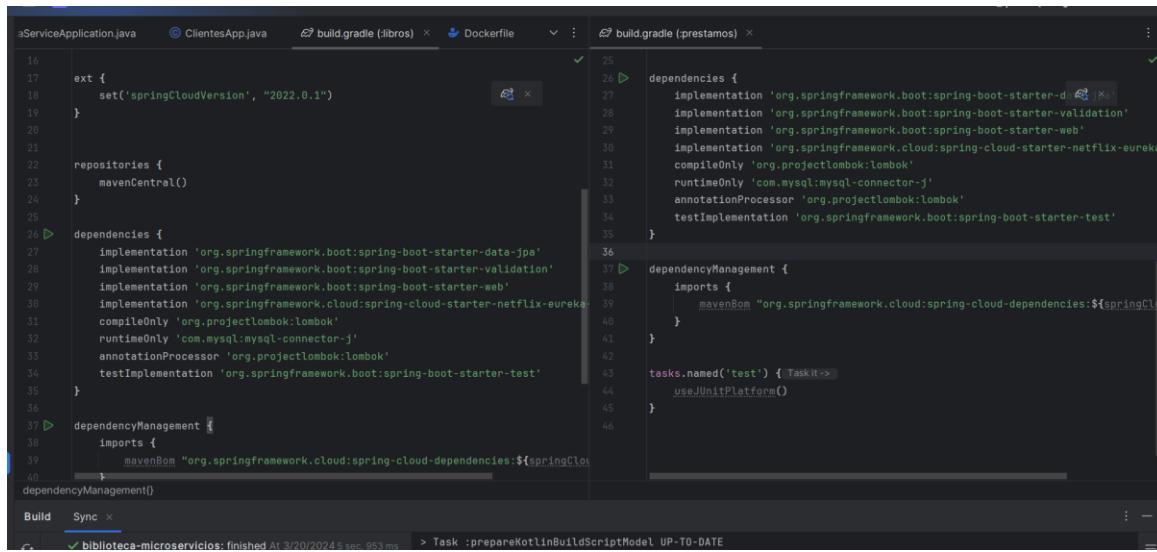
```
1 plugins {
2     id 'java'
3     id 'org.springframework.boot' version '3.0.2'
4     id 'io.spring.dependency-management' version '1.1.0'
5 }
6
7 group = 'co.edu.eam'
8 version = '0.0.1-SNAPSHOT'
9 sourceCompatibility = '17'
10
11 repositories {
12     mavenCentral()
13     maven { url 'https://artifactory-oss.prod.netflix.net/artifactory/maven-oss-candidates' }
14 }
15
16 ext {
17     set('springCloudVersion', "2022.0.1")
18 }
19
20 dependencies {
21     implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-server'
22     testImplementation 'org.springframework.boot:spring-boot-starter-test'
23 }
24
25 dependencyManagement {
```

## Application.properties:



```
spring.application.name=eureka-server
server.port=8761
eureka.client.fetch-registry=false
eureka.client.register-with-eureka=false
```

## Ext, dependency and management dependency para eureka añadido en los otros microservicios



```
ext {
    set('springCloudVersion', "2022.0.1")
}

repositories {
    mavenCentral()
}

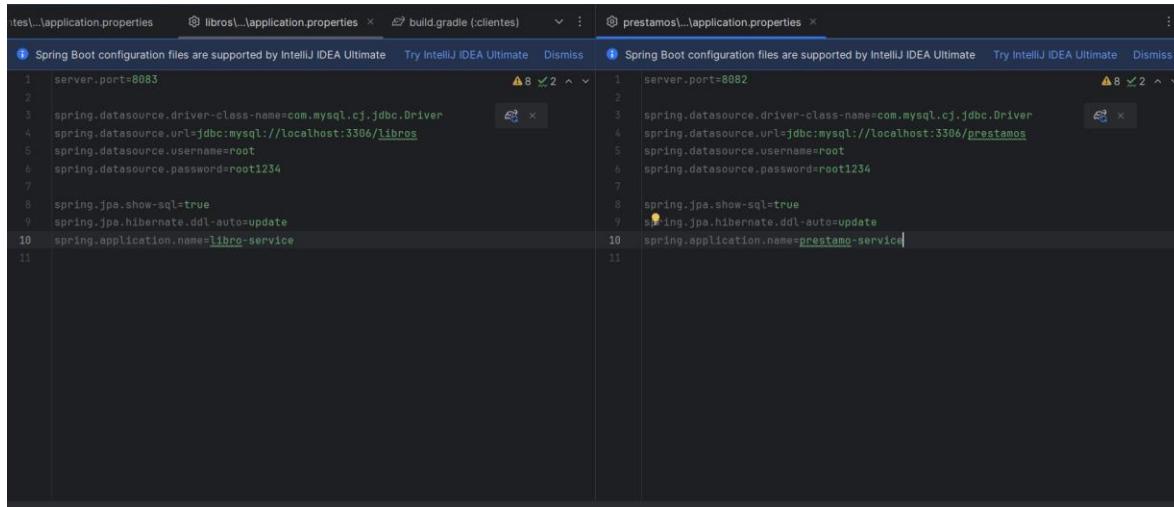
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-dependencies:${springCloudVersion}"
    }
}
```

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-dependencies:${springCloudVersion}"
    }
}
```

## Archivos de properties actualizados



```
server.port=8083
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/libros
spring.datasource.username=root
spring.datasource.password=root1234
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto:update
spring.application.name=libro-service
```

```
server.port=8082
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/prestamos
spring.datasource.username=root
spring.datasource.password=root1234
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto:update
spring.application.name=prestamo-service
```

Añadimos la dependencia al módulo y la etiqueta @LoadBalanced al método de RestTemplate

The screenshot shows three tabs in an IDE:

- RestTemplateConfig.java**: Contains code for a RestTemplate configuration.
- docker-compose.yml**: Configuration for Docker services.
- build.gradle(:clientes)**: Gradle build file for the Clientes service.

```

RestTemplateConfig.java
package co.edu.eam;
import ...;
no usages
@Configuration
public class RestTemplateConfig {
    no usages
    @LoadBalanced
    @Bean
    public RestTemplate restTemplate() {
        ...
    }
}

```

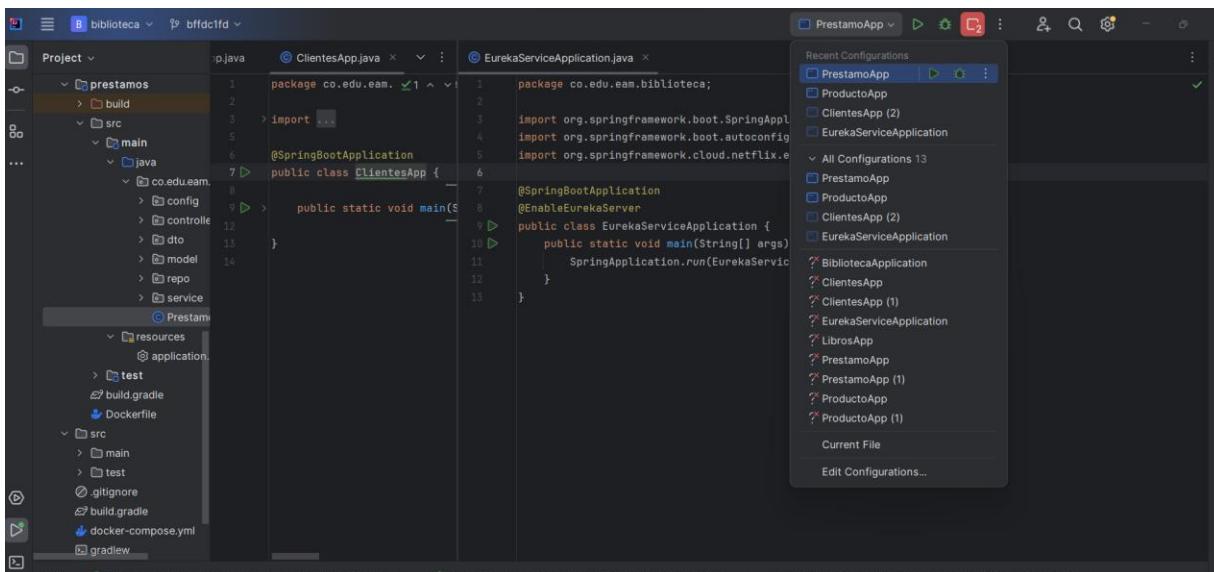
```

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-mongodb'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    implementation 'org.springframework.cloud:spring-cloud-starter-loadbalancer'
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-dependencies:${springCloudVersion}"
    }
}

```

## Corremos Eureka y los otros servicios uno por uno



Desde Eureka nos empiezan a aparecer uno por uno los servicios que hemos corrido

The screenshot shows the Spring Eureka dashboard at `localhost:8761`:

- System Status** table:

Environment	test	Current time	2024-03-21T14:12:44 -0500
Data center	default	Uptime	00:04
		Lease expiration enabled	false
		Renews threshold	3
		Renews (last min)	2

- DS Replicas** section: Shows registered instances.
- Instances currently registered with Eureka** table:

Application	AMIs	Availability Zones	Status
CLIENTE-SERVICE	n/a (1)	(1)	UP (1) - <a href="http://localhost:cliente-service:8082">localhost:cliente-service:8082</a>

Ahora ponemos Eureka en el docker-compose, editamos los archivos.properties, compilamos cada microservicio y levantamos el proyecto con el Docker-compose...

Si ingresamos a Eureka, deberán aparecer los servicios como se aprecian en la siguiente screenshot:

The screenshot shows the Spring Eureka dashboard. At the top, it displays 'System Status' with environment 'test', data center 'default', current time '2024-03-22T00:39:30 +0000', uptime '00:03', lease expiration enabled 'true', renew threshold '8', and renew count '12'. Below this is the 'DS Replicas' section, which lists registered instances under 'localhost':

Application	AMIs	Availability Zones	Status
CLIENTE-SERVICE	n/a (2)	(2)	UP (2) - cliente-service:bd767563606bf5a99a145b235bdae971 , cliente-service:72fb94b5f10aa9f88b0377896d91b28
LIBRO-SERVICE	n/a (1)	(1)	UP (1) - libro-service:9bfb643cd1f5c72f81845a5d43dc35ed
PRESTAMO-SERVICE	n/a (1)	(1)	UP (1) - prestamo-service:f3bbcd4c6981663d4aaad3bc71c1c30

Ahora bien, probemos 😊

The screenshot shows a Postman request to 'http://localhost:9094/api/cliente'. The method is 'POST', and the URL is 'http://localhost:9094/api/cliente'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   ... "nombre": "Andres",  
3   ... "email": "elenemigo@gmail.com",  
4   ... "telefono": "000000",  
5   ... "password": "ush"  
}
```

The response status is '201 Created', time '114 ms', and size '324 B'. The response body is:

```
1 {  
2   "mensaje": "Cliente creado correctamente",  
3   "dato": {  
4     "codigo": "65fcda86224722173b05abc",  
5     "nombre": "Andres",  
6     "email": "elenemigo@gmail.com",  
7     "telefono": "000000"  
8   }  
9 }
```

Creamos el DTO, Controlador y Servicio para Autores

The screenshot shows the IntelliJ IDEA interface with several tabs open:

- Project**: Shows the file structure of the `biblioteca-microservicios [biblioteca]` project.
- AutorServicio.java**: Contains annotations like `@Service`, `@AllArgsConstructor`, and a method `public Autor save(AutorDTO autor)`.
- LibroController.java**: Contains annotations like `@RestController`, `@RequestMapping("/api/libro")`, and methods for `getLibros`, `getLibro`, `createLibro`, and `updateLibro`.
- AutorController.java**: Contains annotations like `@RestController`, `@RequestMapping("/api/autor")`, and methods for `getAutores`, `getAutor`, `createAutor`, and `updateAutor`.
- AutorDTO.java**: Contains a record definition: `public record AutorDTO(Long id, String nombre){}`.

## Probamos con Postman

The screenshot shows a Postman collection with the following details:

- Overview**: Shows various API endpoints listed.
- Request**: A POST request to `http://localhost:9092/api/autor`.
- Body**: Set to `raw` and contains the following JSON payload:

```

1 {
2   "id": 1,
3   "nombre": "Luisa Celis"
4 }

```

- Response**: Status: `201 Created` Time: `104 ms`. The response body is:

```

1 {
2   "mensaje": "Autor creado correctamente",
3   "dato": {
4     "id": 1,
5     "nombre": "Luisa Celis"
6   }
7 }

```

## Creamos un libro

The screenshot shows the Postman interface with a successful POST request to `http://localhost:9092/api/libro`. The request body is a JSON object containing fields like `isbn`, `nombre`, `genero`, `unidades`, `fechaPublicacion`, and `autor`. The response status is `201 Created` with a message "Libro creado correctamente".

Para prestamos ajustamos nuestras urls

```
Respuesta<Boolean> respuesta = restTemplate.exchange(  
    url: "http://localhost:9092/api/libro/" + isbn,  
    HttpMethod.GET  
  
    Respuesta<Boolean> respuesta = restTemplate.exchange(  
        url: "http://localhost:9094/api/cliente/" + codigoCliente,
```

Y con esto podemos crear los préstamos y todo lo que queramos 😊 FIIIIIIIN