**Create Eureka Discovery Server and Register Microservices**

Eureka Discovery Server holds a registry of all the services that are available for immediate consumption. Any client who wants to consume a RESTful Web Service can access the Discovery Server to find out which services are currently available. Eureka Discovery Server is part of the Spring Cloud module.

1. Create and Launch Eureka Discovery Server

Steps:

1. Go to [https://start.spring.io](https://start.spring.io/) and generate a project with the following configuration:
   * Group: com.cognizant
   * Artifact: eureka-discovery-server
   * Dependencies: Spring Cloud Discovery → Eureka Server
2. Download the project and build it using Maven in the command line:
3. mvn clean install
4. Import the project into Eclipse (or IntelliJ if preferred).
5. In the main class (EurekaDiscoveryServerApplication.java), add the annotation:
6. @EnableEurekaServer
7. In src/main/resources/application.properties, add the following configuration:
8. server.port=8761
9. eureka.client.register-with-eureka=false
10. eureka.client.fetch-registry=false
11. logging.level.com.netflix.eureka=OFF
12. logging.level.com.netflix.discovery=OFF
13. Run the Eureka Discovery Server application.
14. Open the Eureka Dashboard in your browser:
15. http://localhost:8761
16. The dashboard will initially show no registered instances.

2. Register Account REST API with Eureka Discovery

Steps:

1. Go to [https://start.spring.io](https://start.spring.io/) and provide the following configuration:
   * Group: com.cognizant
   * Artifact: account
   * Dependencies:
     + Spring Boot DevTools
     + Eureka Discovery Client
     + Spring Web
2. Click Explore, copy the generated pom.xml content, and replace it in your existing Account project’s pom.xml.
3. Build the project using Maven:
4. mvn clean install
5. In the AccountServiceApplication.java, add:
6. @EnableDiscoveryClient
7. In application.properties, add:
8. spring.application.name=account-service
9. eureka.client.service-url.defaultZone=http://localhost:8761/eureka
10. server.port=8081
11. Stop all services (Eureka, Account, Loan) using Eclipse Console.
12. First start the Eureka Discovery Server, and wait for it to fully initialize.
13. Open http://localhost:8761 – the service list should be empty.
14. Start the Account Service, wait for it to register.
15. Refresh Eureka Dashboard – you should see ACCOUNT-SERVICE registered.

3. Register Loan REST API with Eureka Discovery

Repeat the same steps as above with the following changes:

* Artifact: loan
* spring.application.name: loan-service
* server.port: 8082 (or any free port)

Once started, it should also appear in Eureka Dashboard.

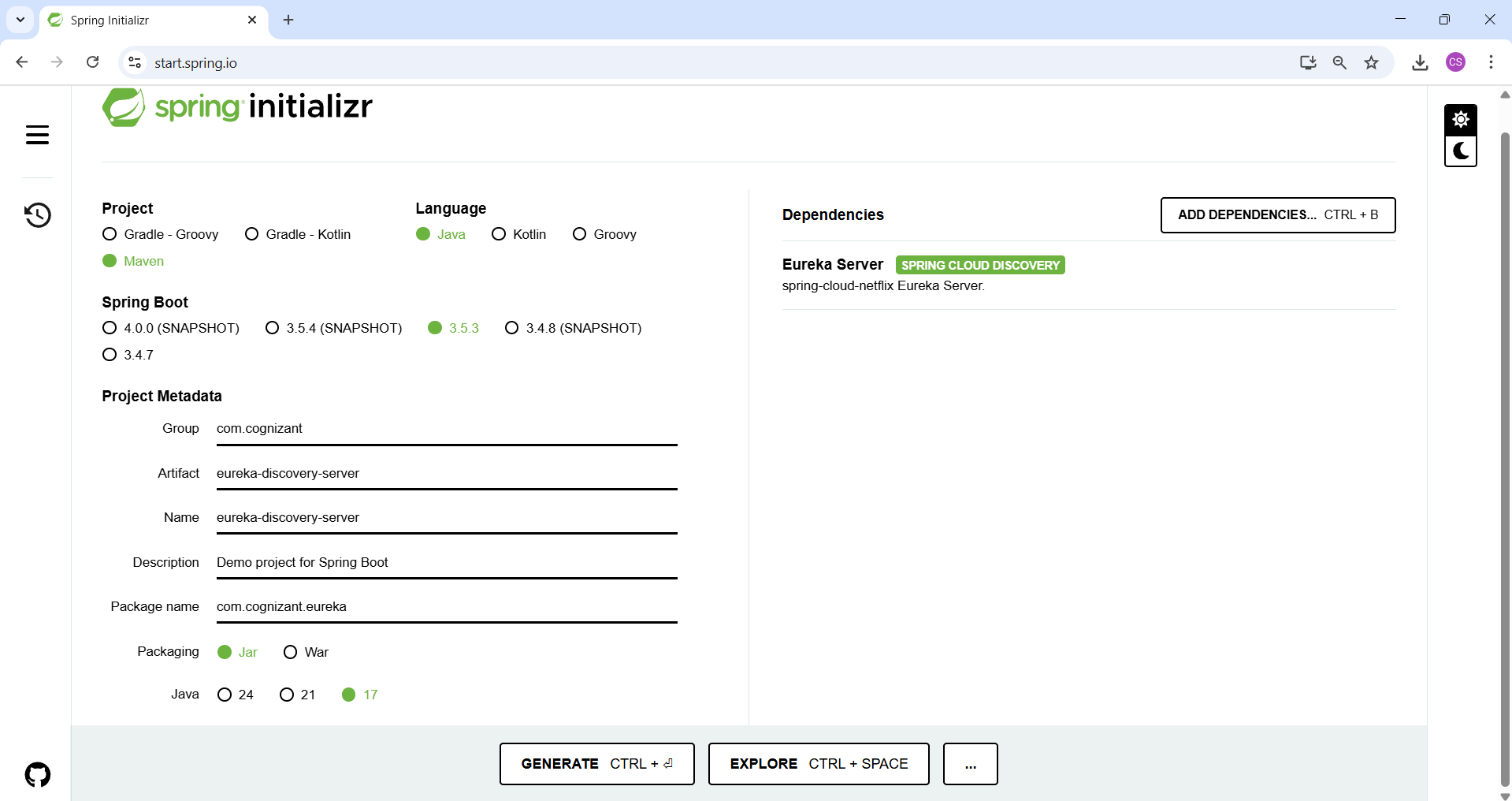
4. Create Spring Cloud API Gateway

Steps:

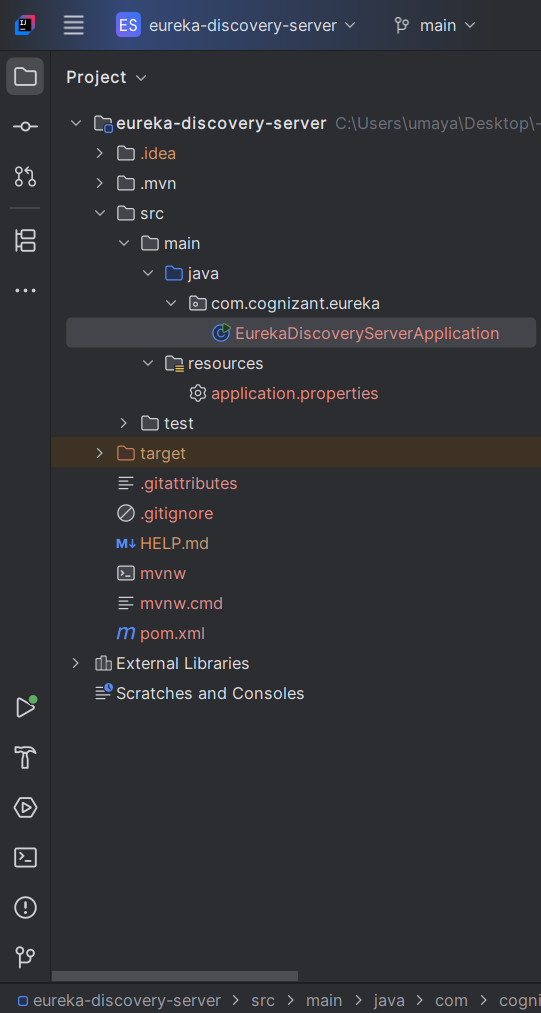
1. Go to [https://start.spring.io](https://start.spring.io/) and create a project:
   * Group: com.cognizant
   * Artifact: api-gateway
   * Dependencies:
     + Spring Cloud Gateway
     + Eureka Discovery Client
2. In application.yml (or application.properties), add the configuration:
3. server:
4. port: 8080
5. spring:
6. application:
7. name: api-gateway
8. cloud:
9. gateway:
10. discovery:
11. locator:
12. enabled: true
13. lower-case-service-id: true
14. routes:
15. - id: account-service
16. uri: lb://account-service
17. predicates:
18. - Path=/account/\*\*
19. - id: loan-service
20. uri: lb://loan-service
21. predicates:
22. - Path=/loan/\*\*
23. main:
24. web-application-type: reactive
25. eureka:
26. client:
27. service-url:
28. defaultZone: http://localhost:8761/eureka
29. Create a Global Logging Filter to log all requests:
    * Create a new class implementing GlobalFilter and log request details in the filter() method.
30. Run the API Gateway Application.

5. Verify Everything Works

* Access the following URLs:
  + http://localhost:8080/account/status → should return "Account Service is up!"
  + http://localhost:8080/loan/status → should return "Loan Service is up!"
* Check that logs are printed from the global filter for each request.
* Eureka Dashboard should show both ACCOUNT-SERVICE and LOAN-SERVICE.

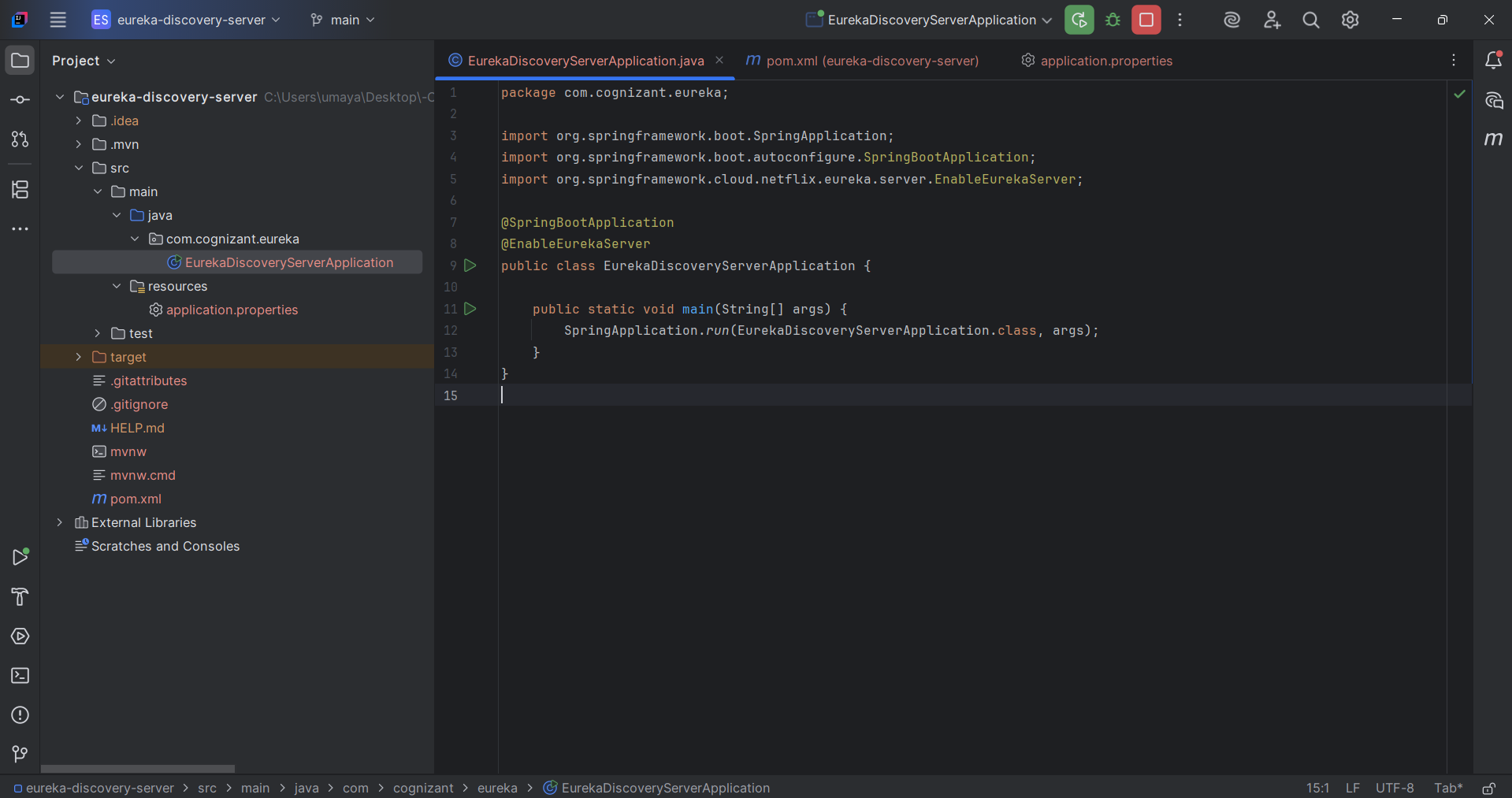
****

**Folder Structure**

****

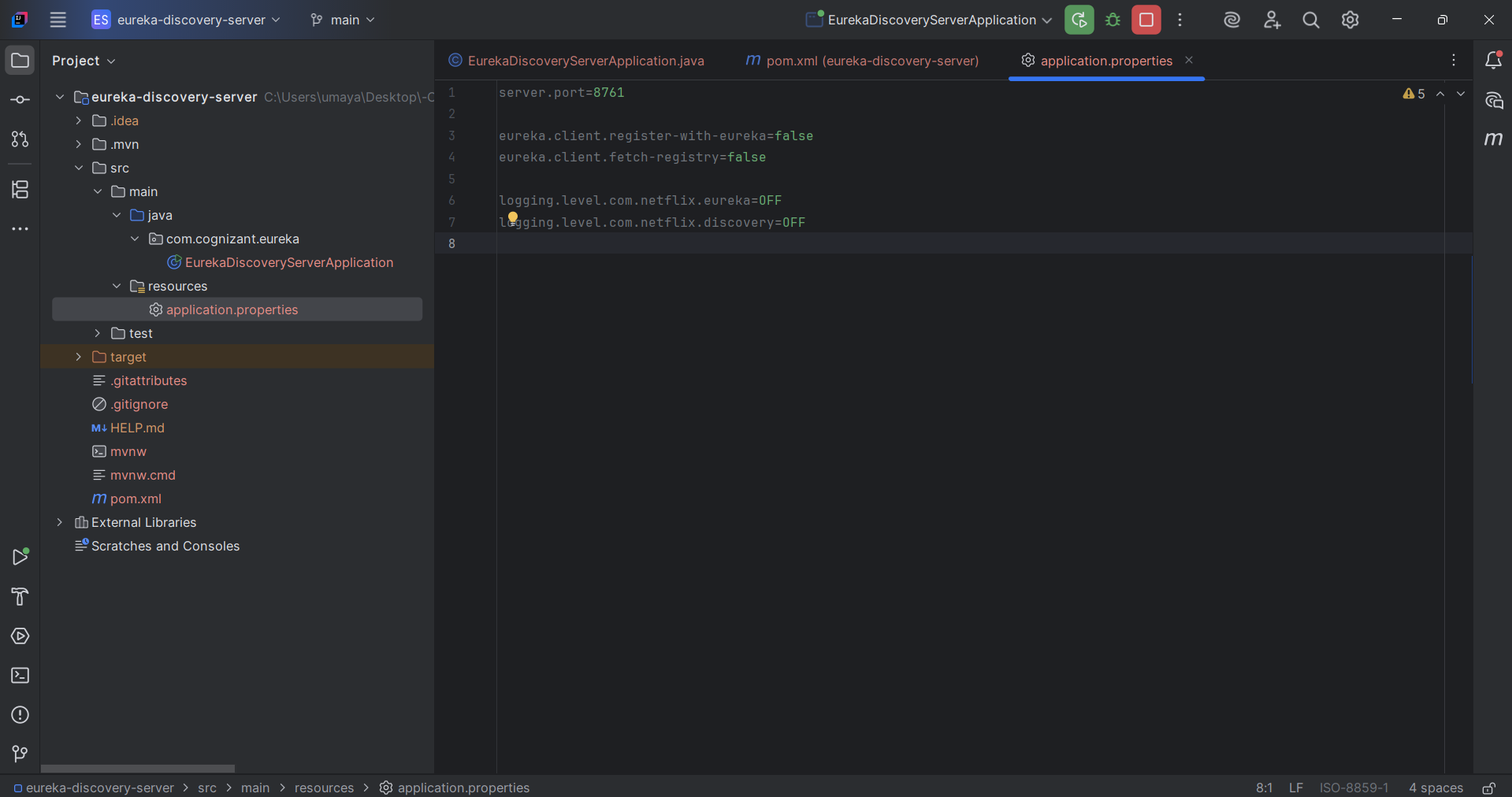
**EurekaDiscoveryServerApplication.java**

package com.cognizant.eureka;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;  
  
@SpringBootApplication  
@EnableEurekaServer  
public class EurekaDiscoveryServerApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(EurekaDiscoveryServerApplication.class, args);  
 }  
}



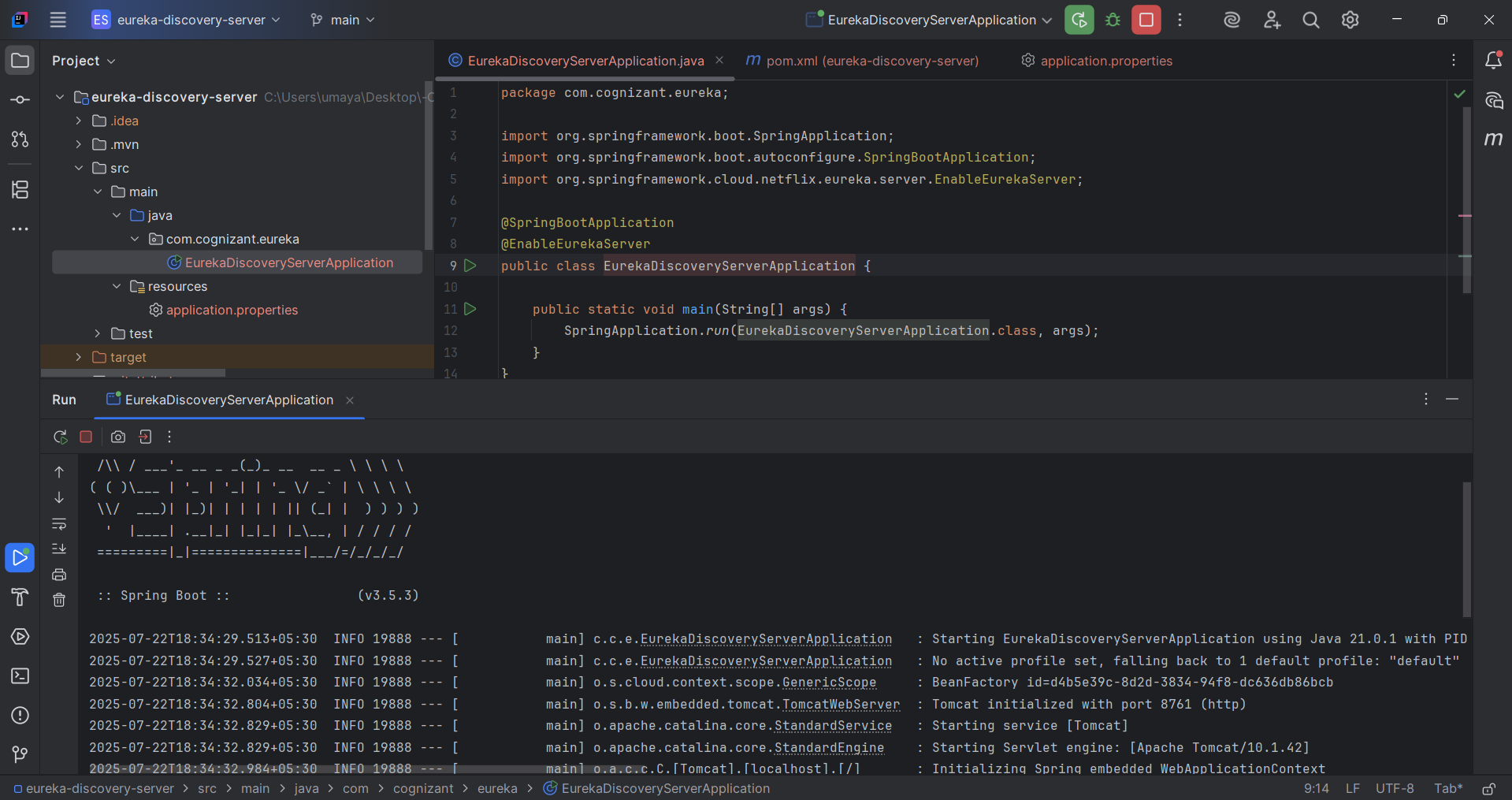
**application.properties**

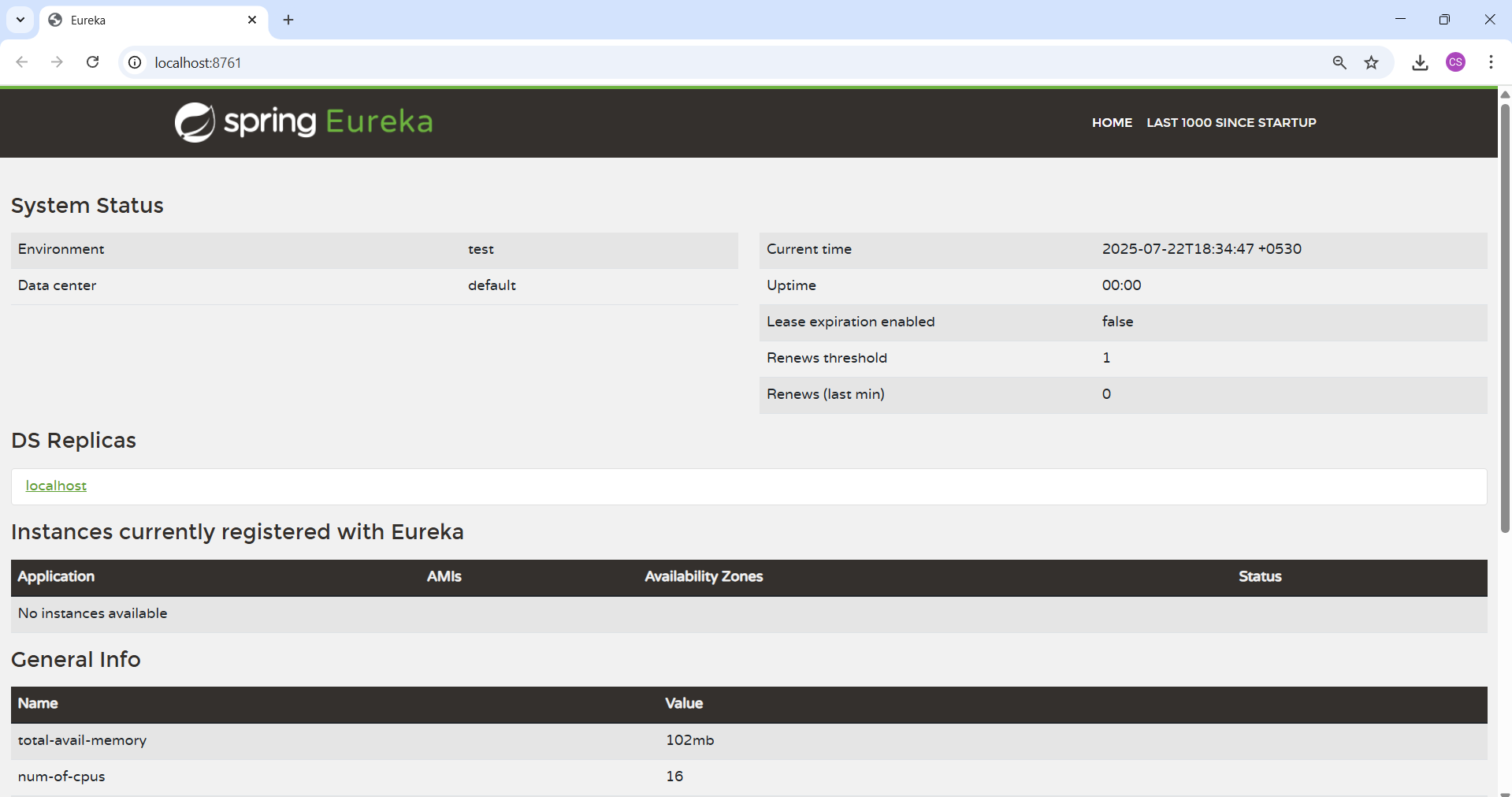
server.port=8761  
  
eureka.client.register-with-eureka=false  
eureka.client.fetch-registry=false  
  
logging.level.com.netflix.eureka=OFF  
logging.level.com.netflix.discovery=OFF

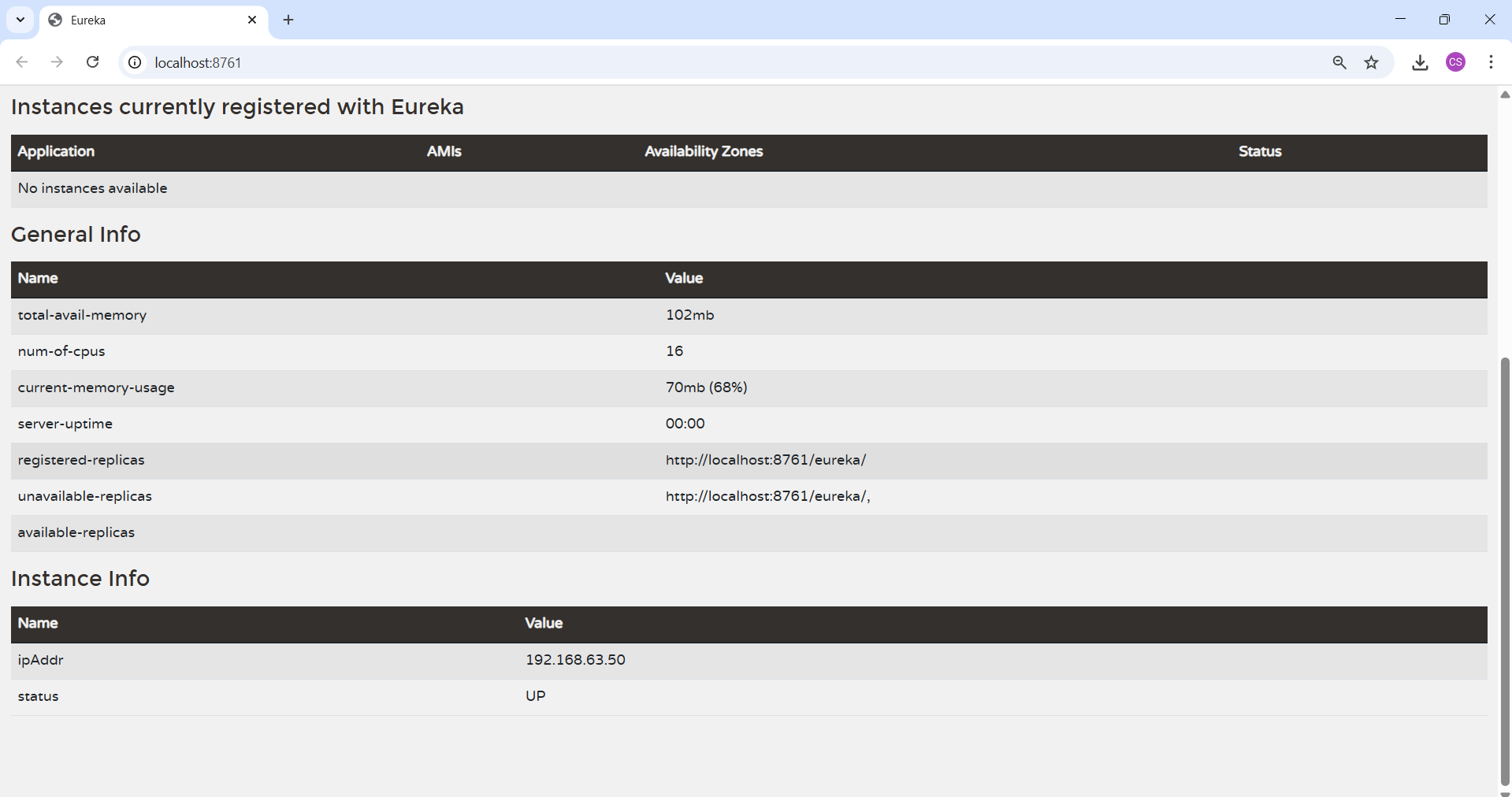


Verifying in browser:

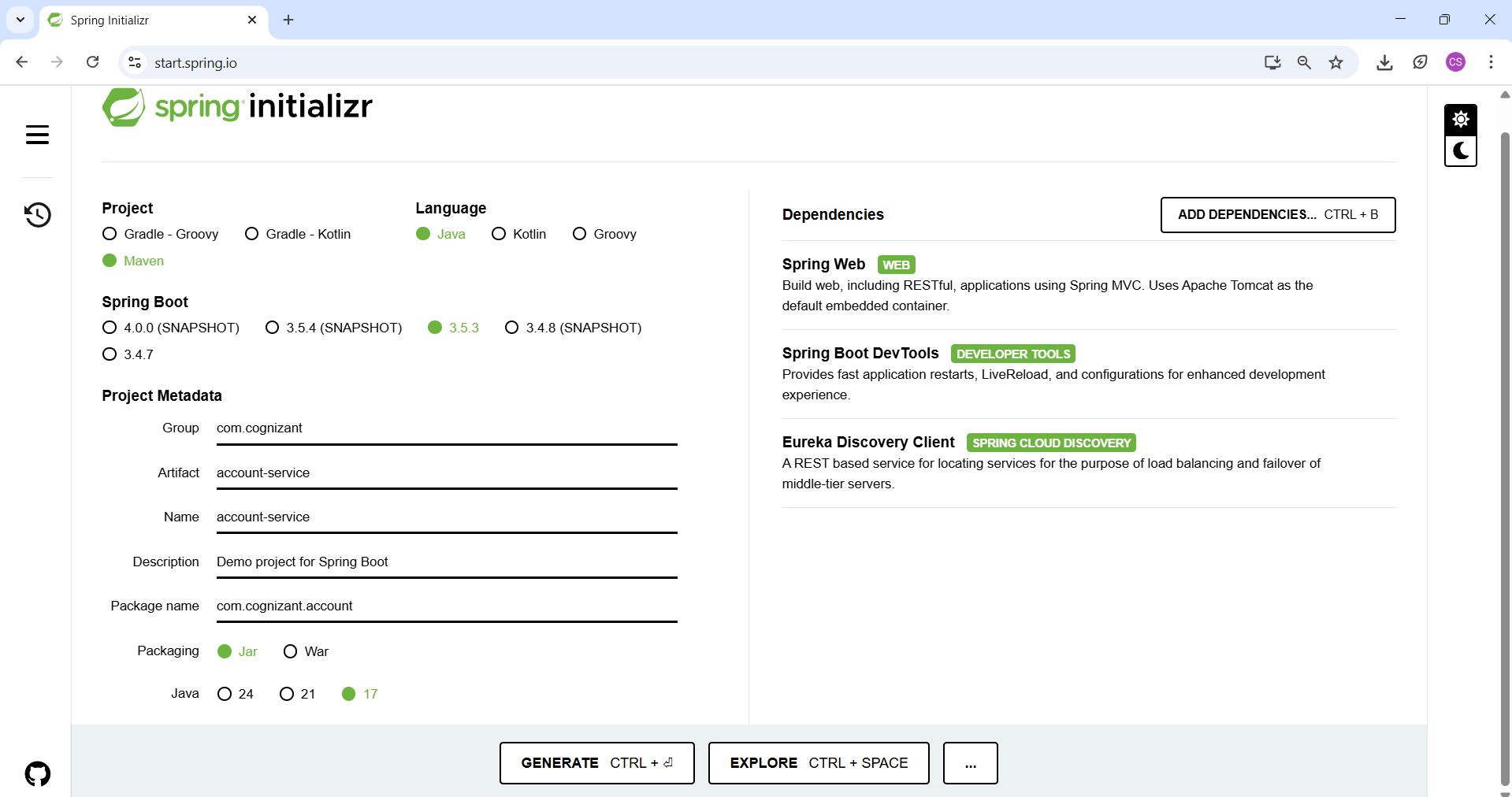
[**http://localhost:8761**](http://localhost:8761)

****

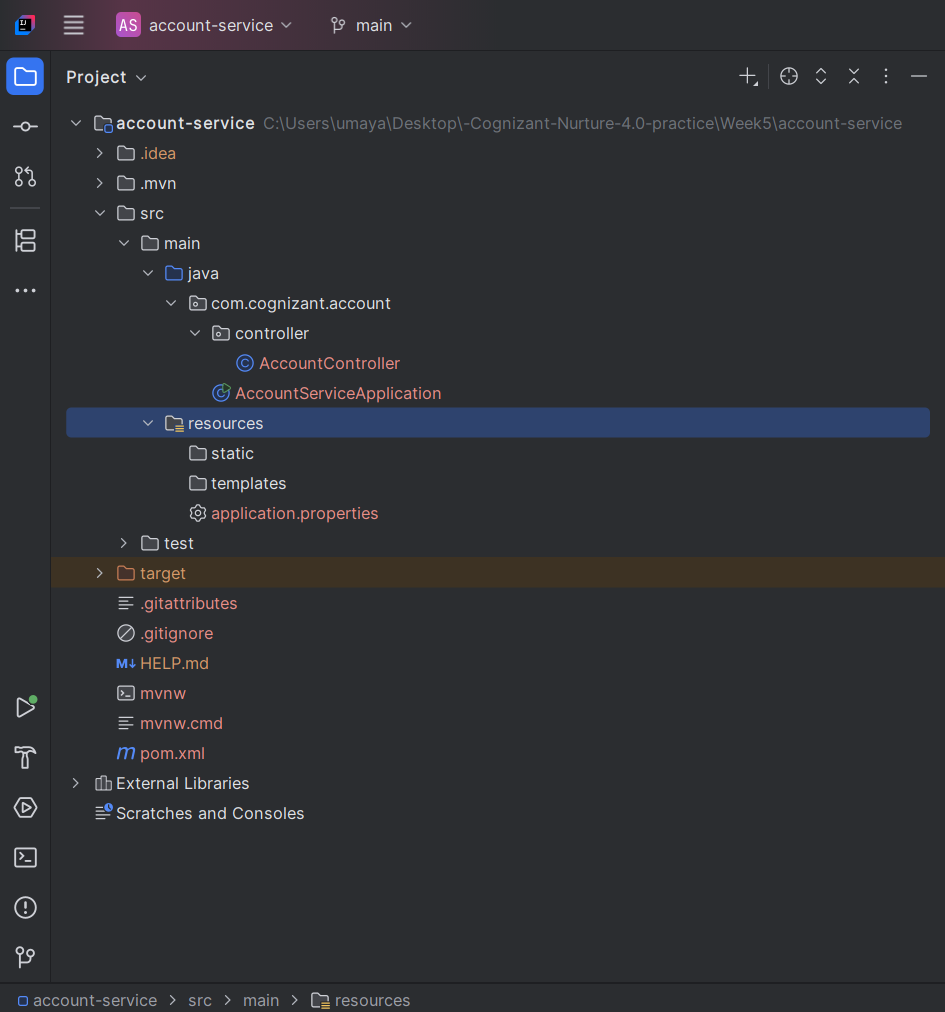
****

****

**Create Account Service Microservice**

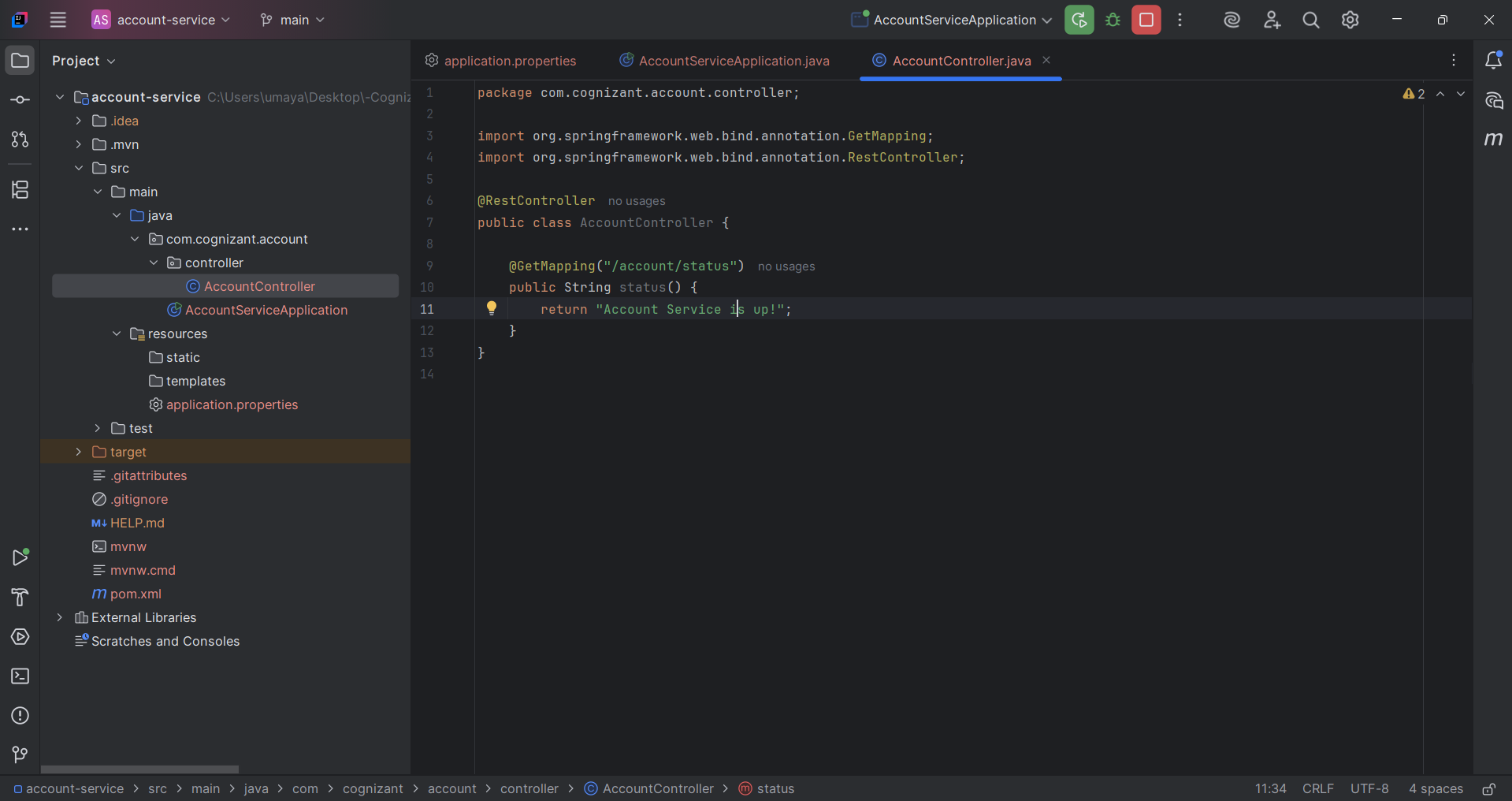
****

**Folder Structure**

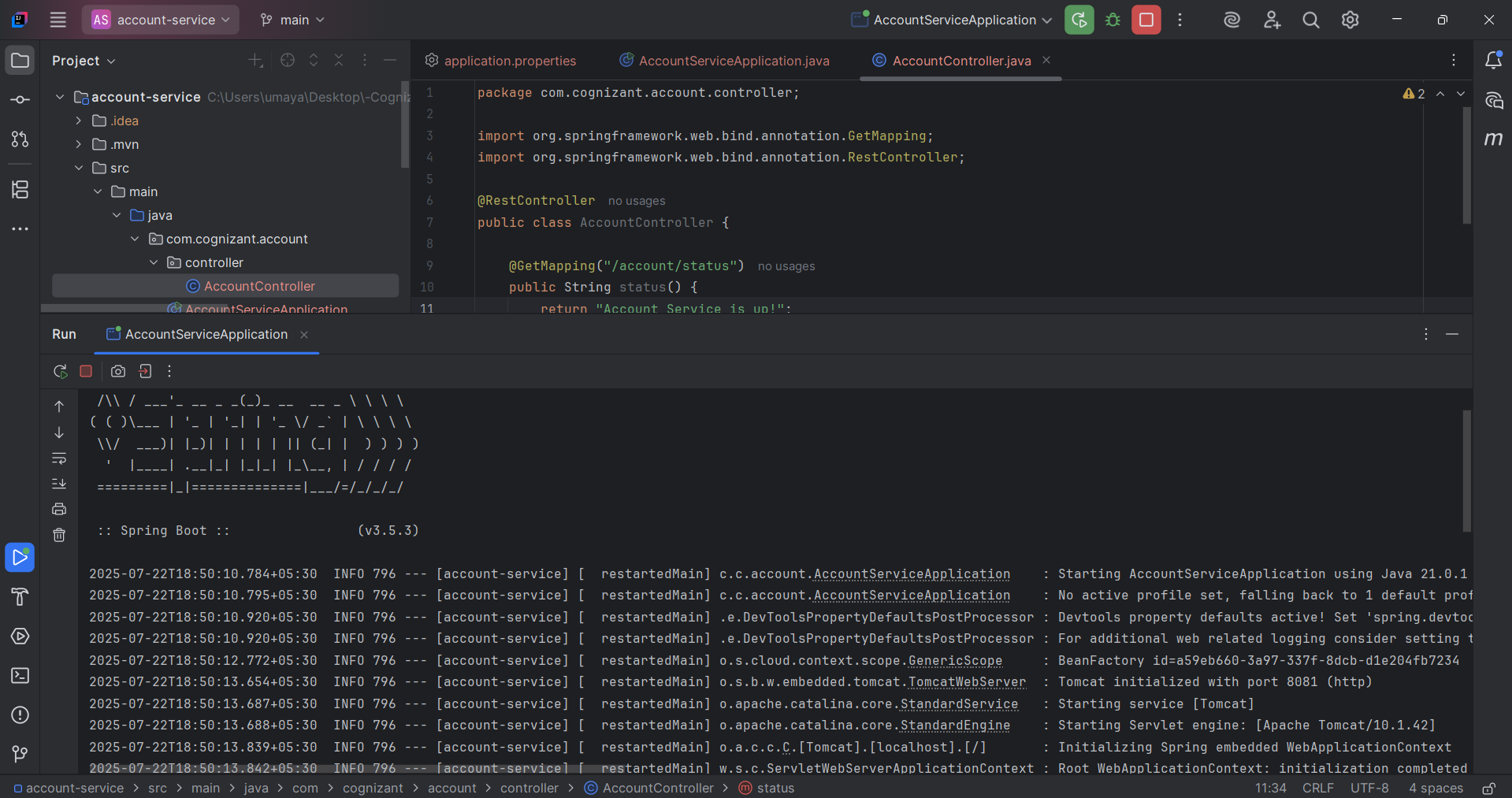
****

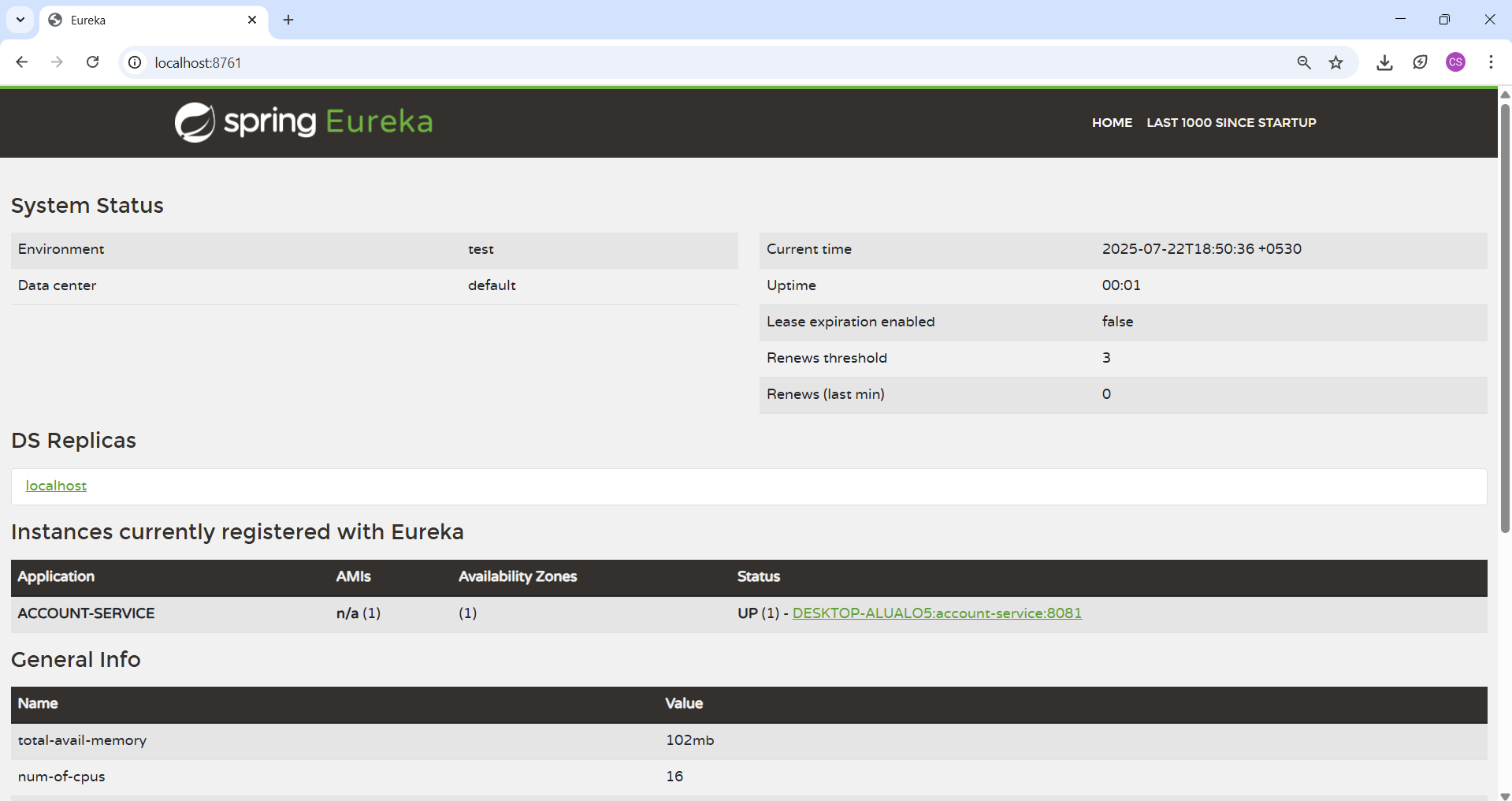
**AccountController.java**

package com.cognizant.account.controller;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class AccountController {  
  
 @GetMapping("/account/status")  
 public String status() {  
 return "Account Service is up!";  
 }  
}



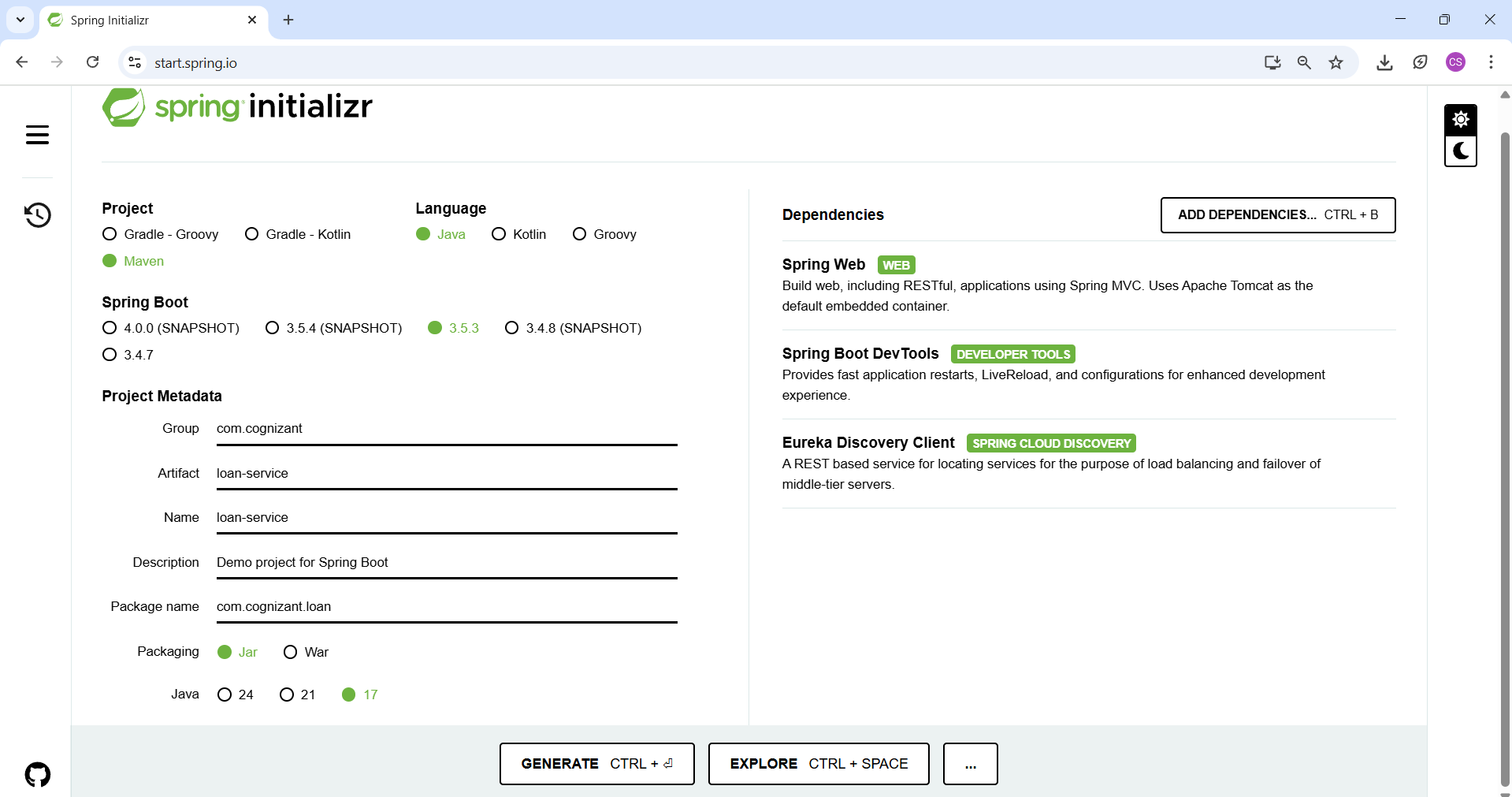
**On running Account Service, we see**

****

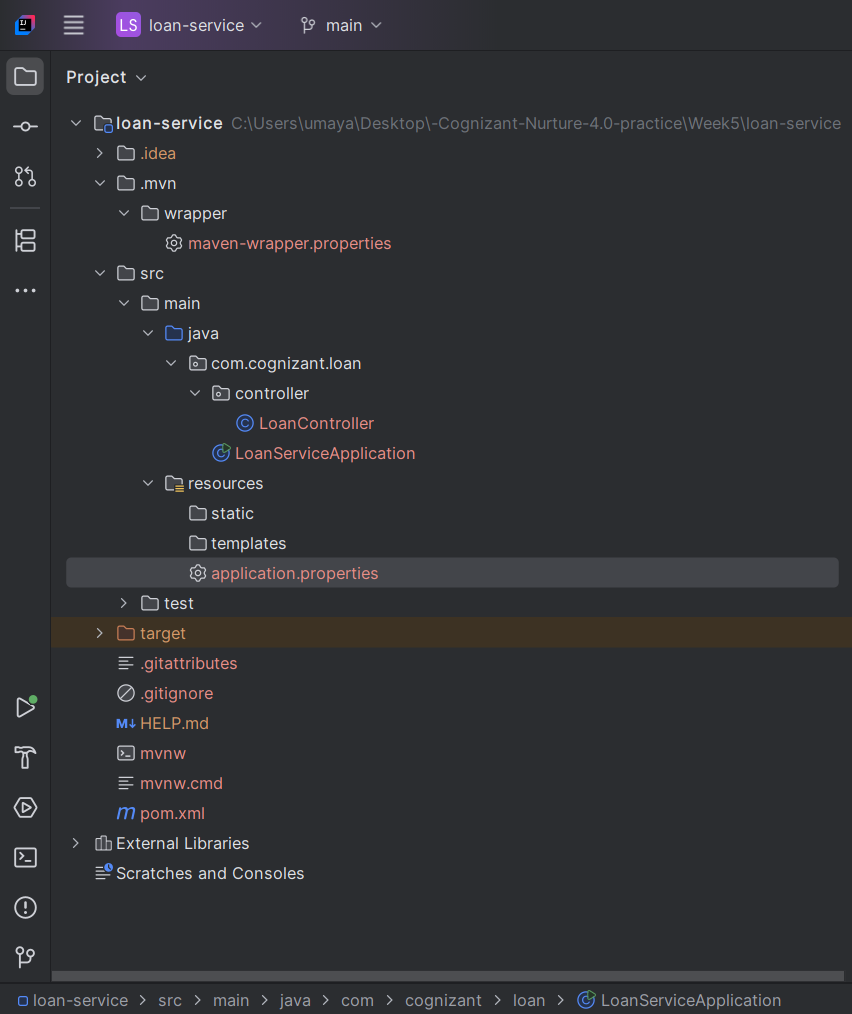
****

Our instance account service is created

**Create Loan Service Microservice**



**Folder Structure**

****

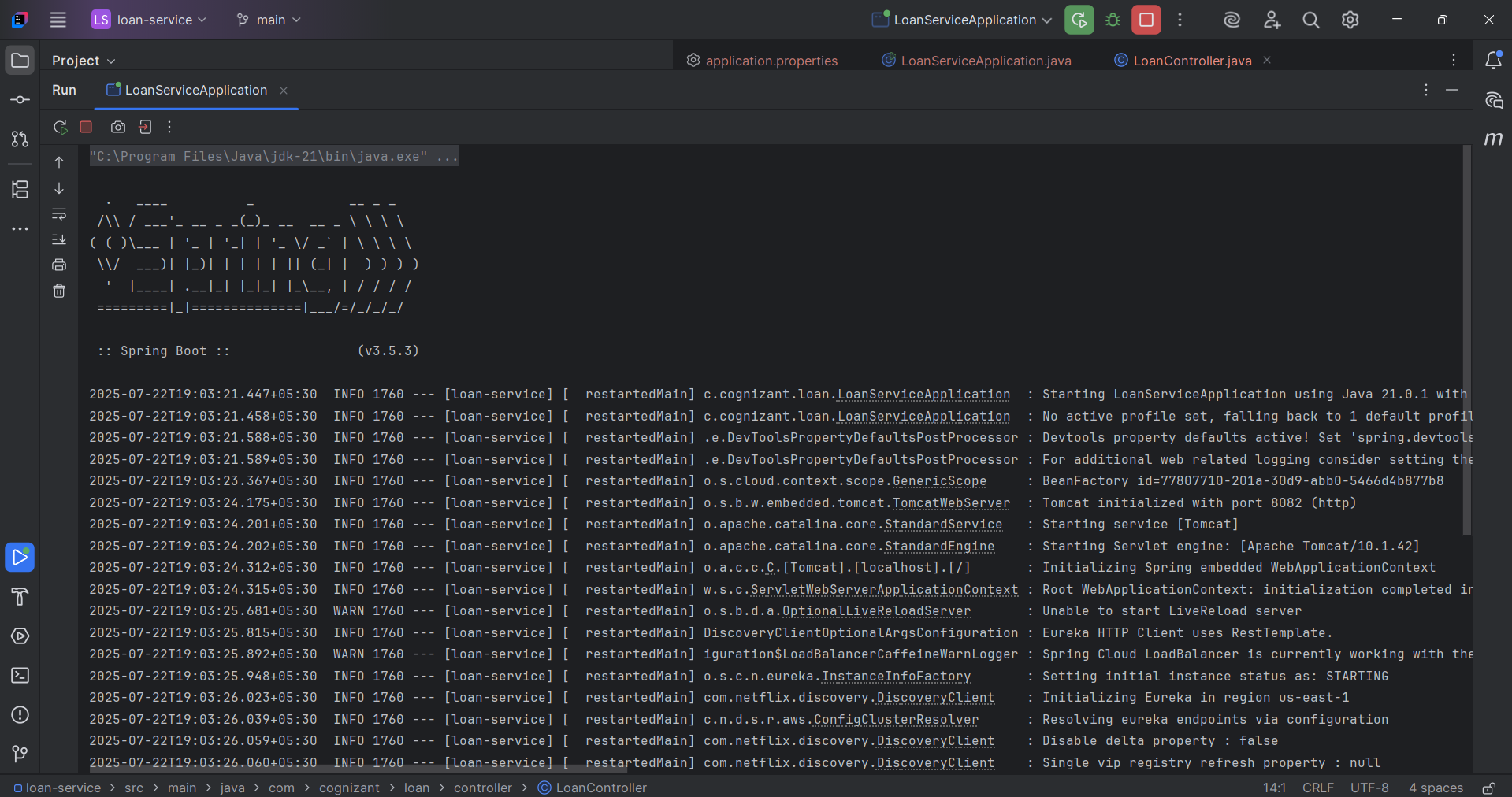
**LoanServiceApplication.java**

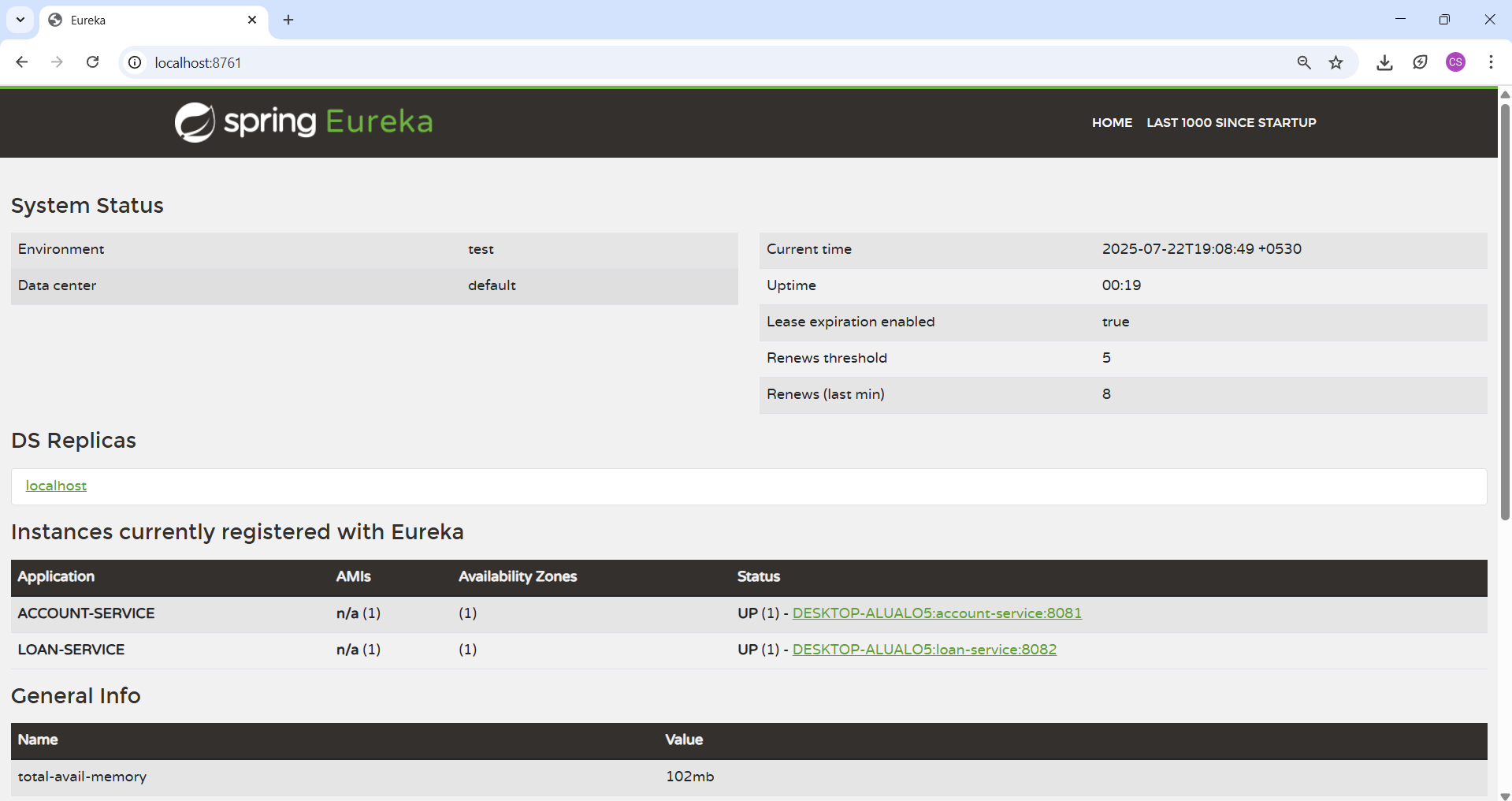
package com.cognizant.loan;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  
  
@SpringBootApplication  
@EnableDiscoveryClient  
public class LoanServiceApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(LoanServiceApplication.class, args);  
 }  
  
}

**LoanController.java**

package com.cognizant.loan.controller;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class LoanController {  
  
 @GetMapping("/loan/status")  
 public String status() {  
 return "Loan Service is up!";  
 }  
}

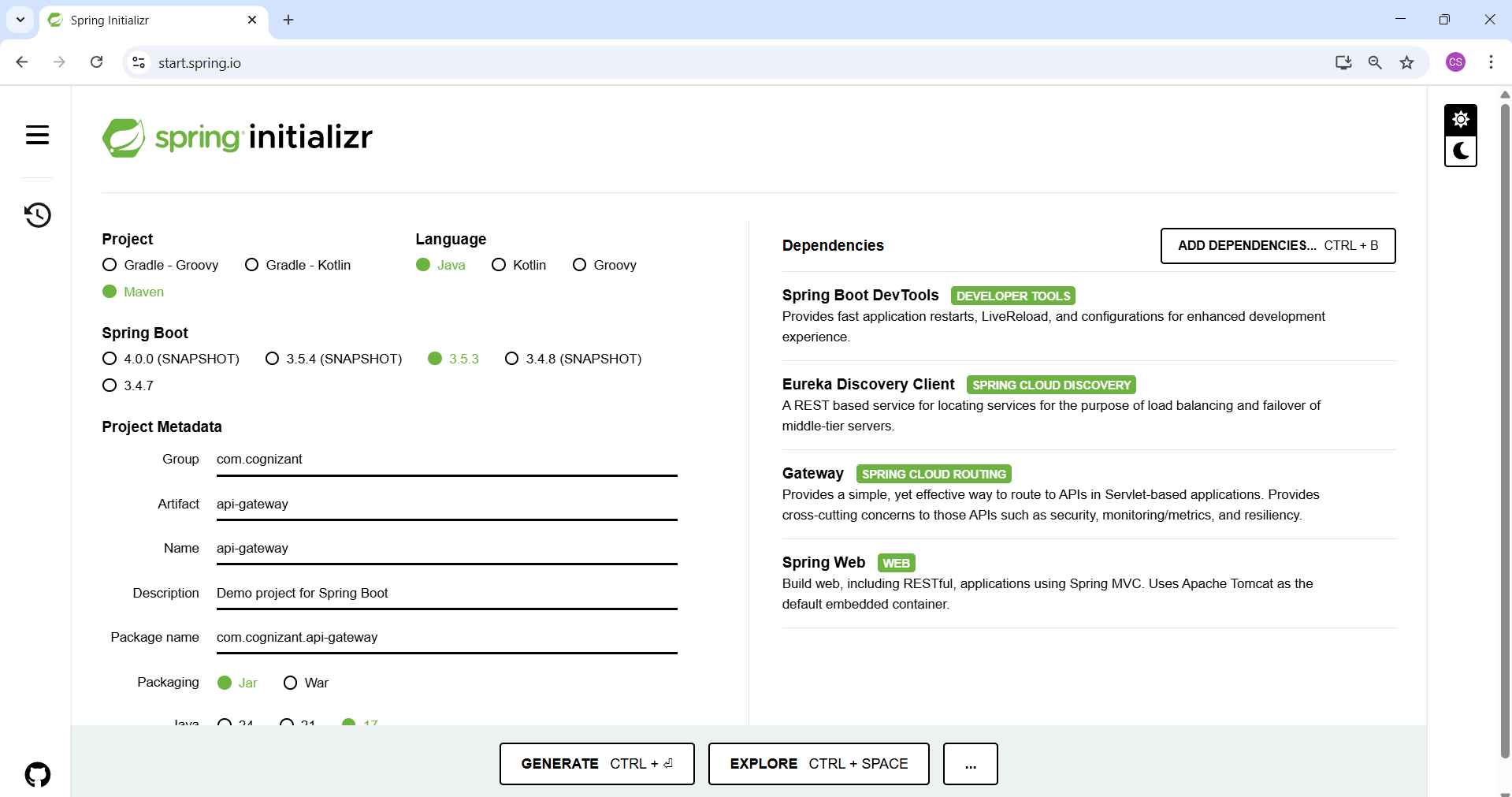
**On running Loan Service, we see**

****

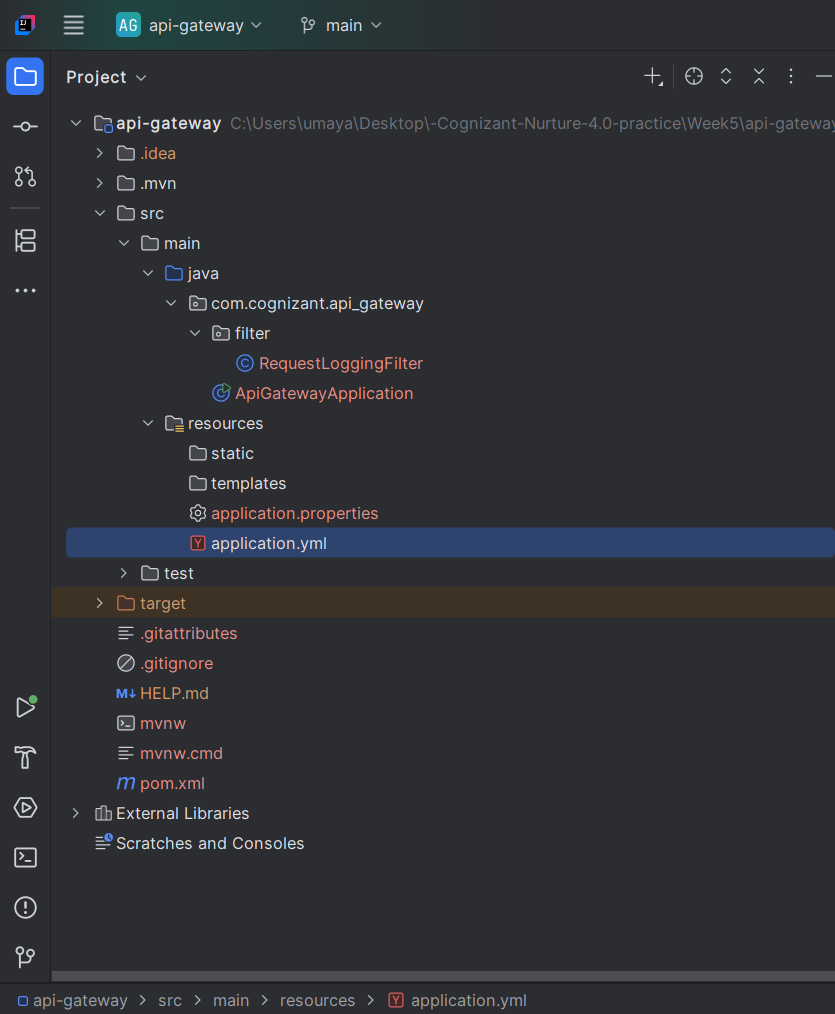
****

Our instance loan service is created

**Spring Cloud API Gateway**

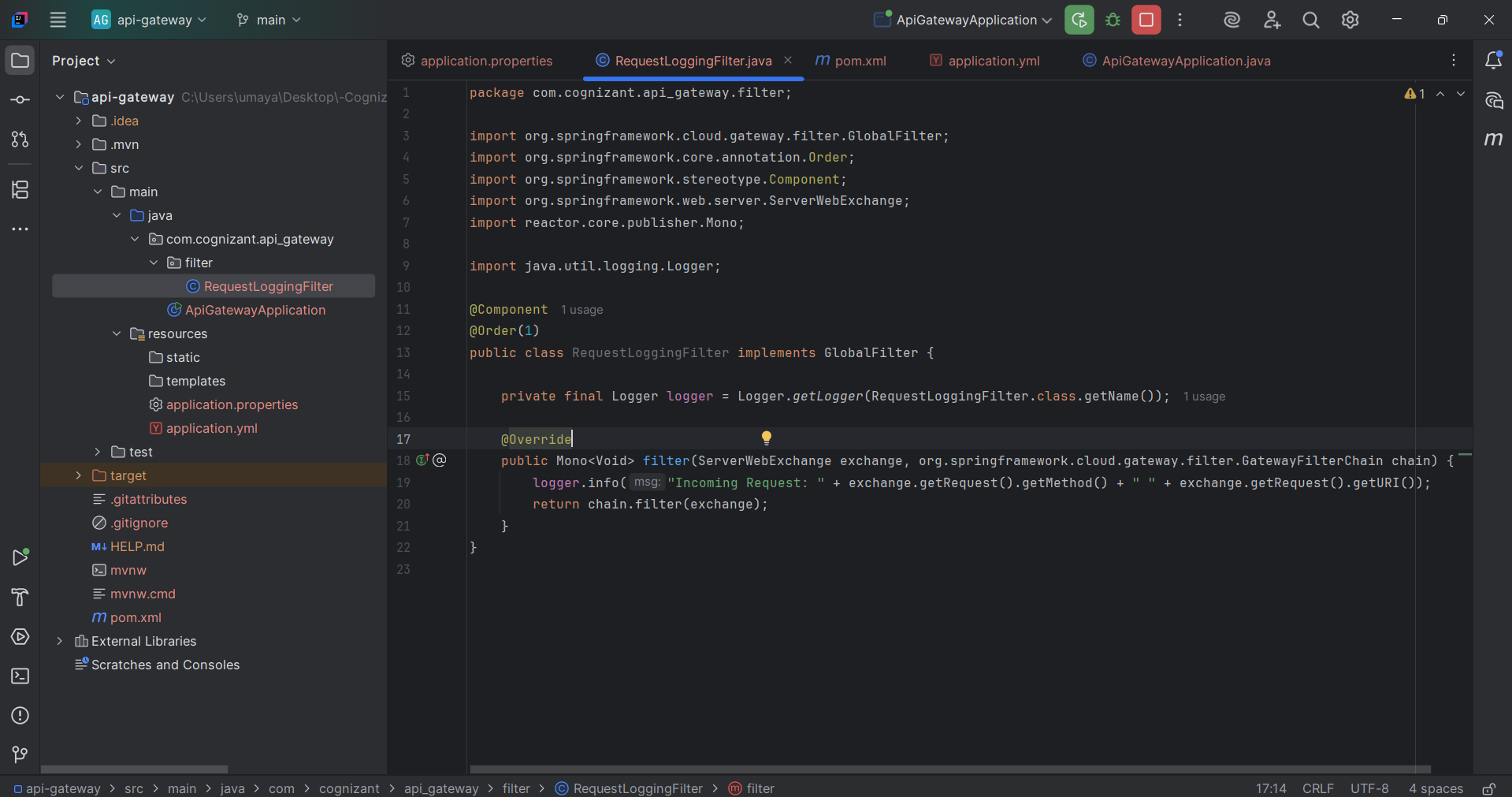
****

**Folder Structure**

****

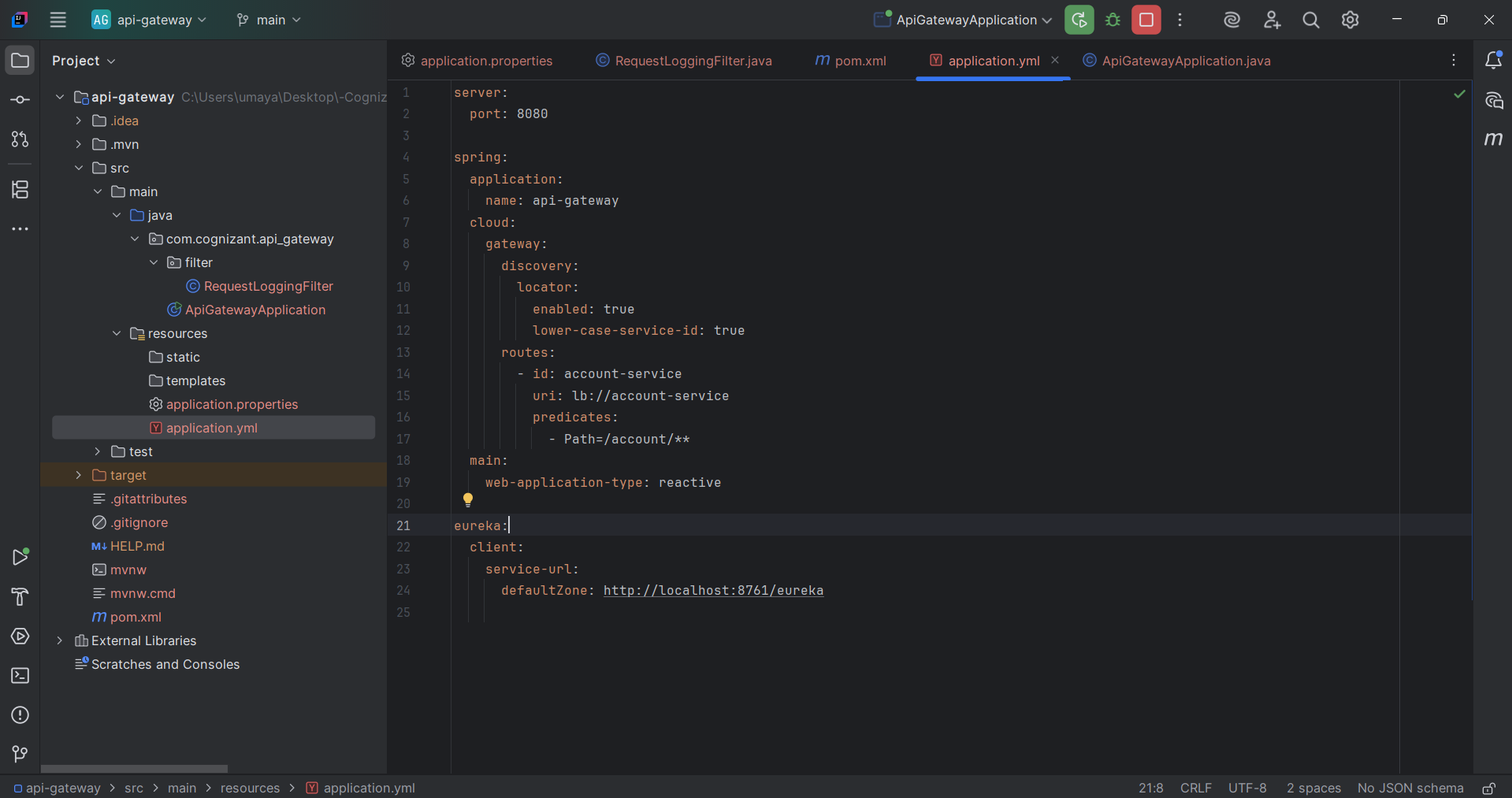
**RequestLoggingFilter.java**

package com.cognizant.api\_gateway.filter;  
  
import org.springframework.cloud.gateway.filter.GlobalFilter;  
import org.springframework.core.annotation.Order;  
import org.springframework.stereotype.Component;  
import org.springframework.web.server.ServerWebExchange;  
import reactor.core.publisher.Mono;  
  
import java.util.logging.Logger;  
  
@Component  
@Order(1)  
public class RequestLoggingFilter implements GlobalFilter {  
  
 private final Logger logger = Logger.*getLogger*(RequestLoggingFilter.class.getName());  
  
 @Override  
 public Mono<Void> filter(ServerWebExchange exchange, org.springframework.cloud.gateway.filter.GatewayFilterChain chain) {  
 logger.info("Incoming Request: " + exchange.getRequest().getMethod() + " " + exchange.getRequest().getURI());  
 return chain.filter(exchange);  
 }  
}

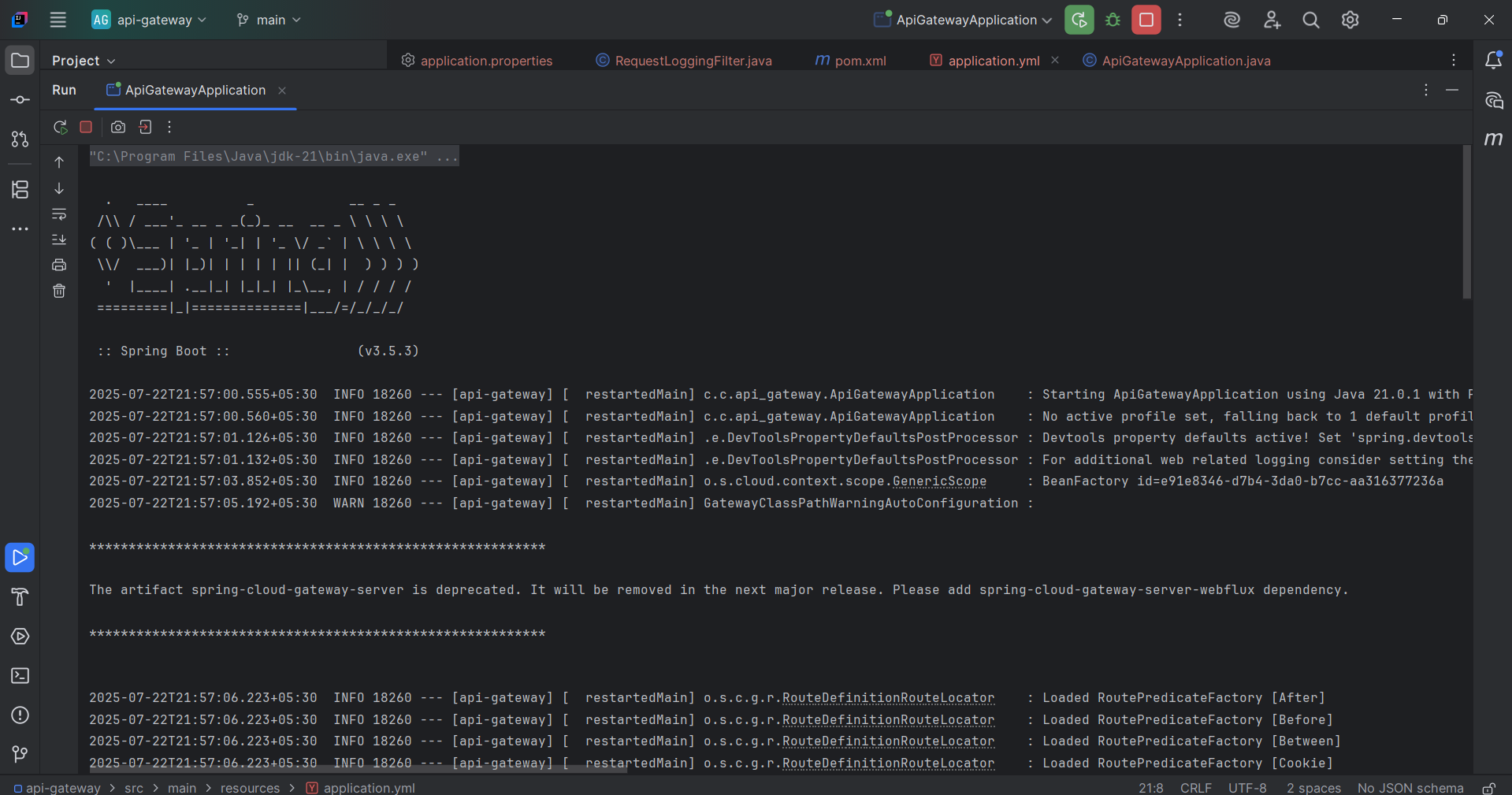


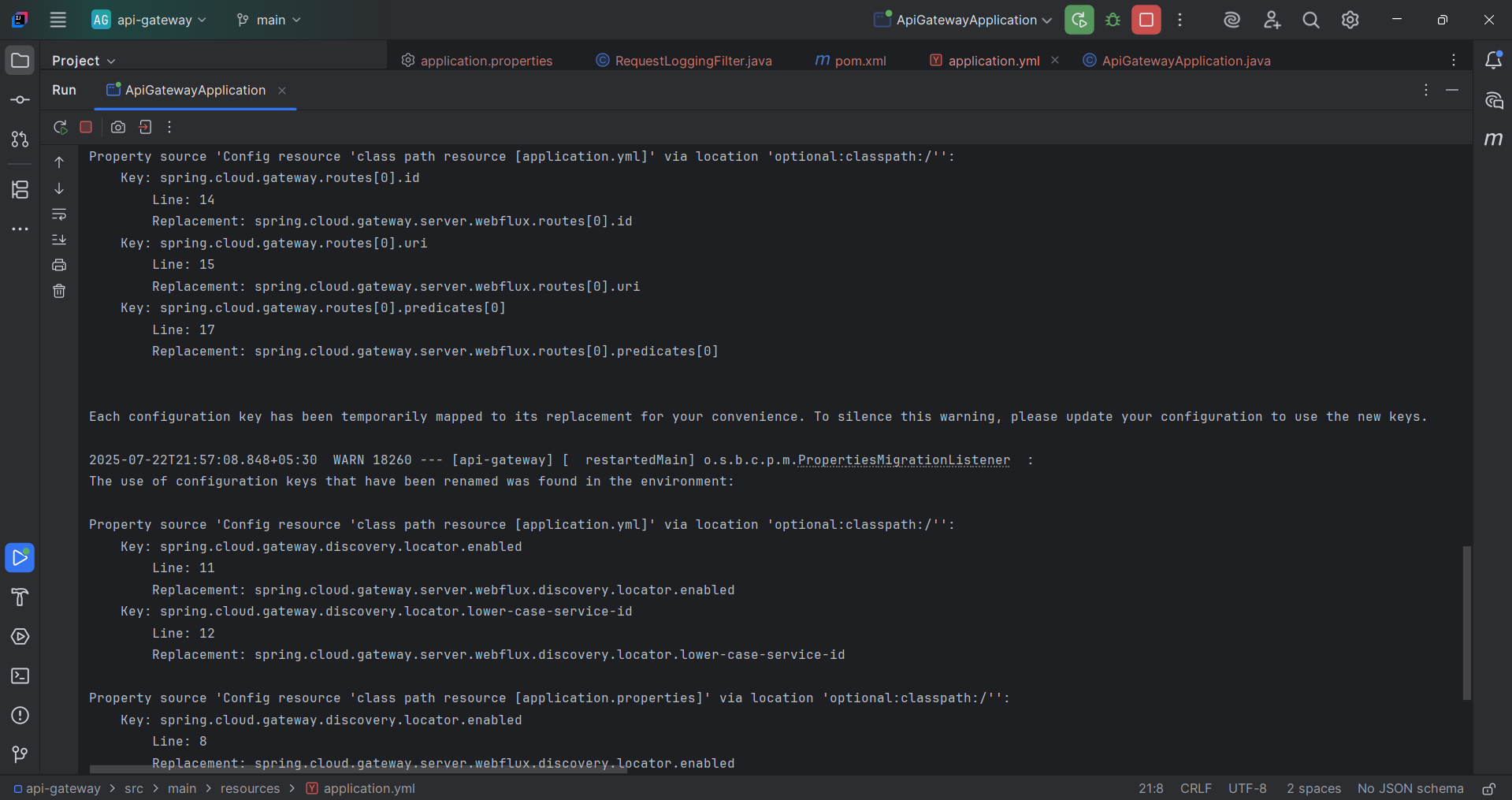
**application.yml**

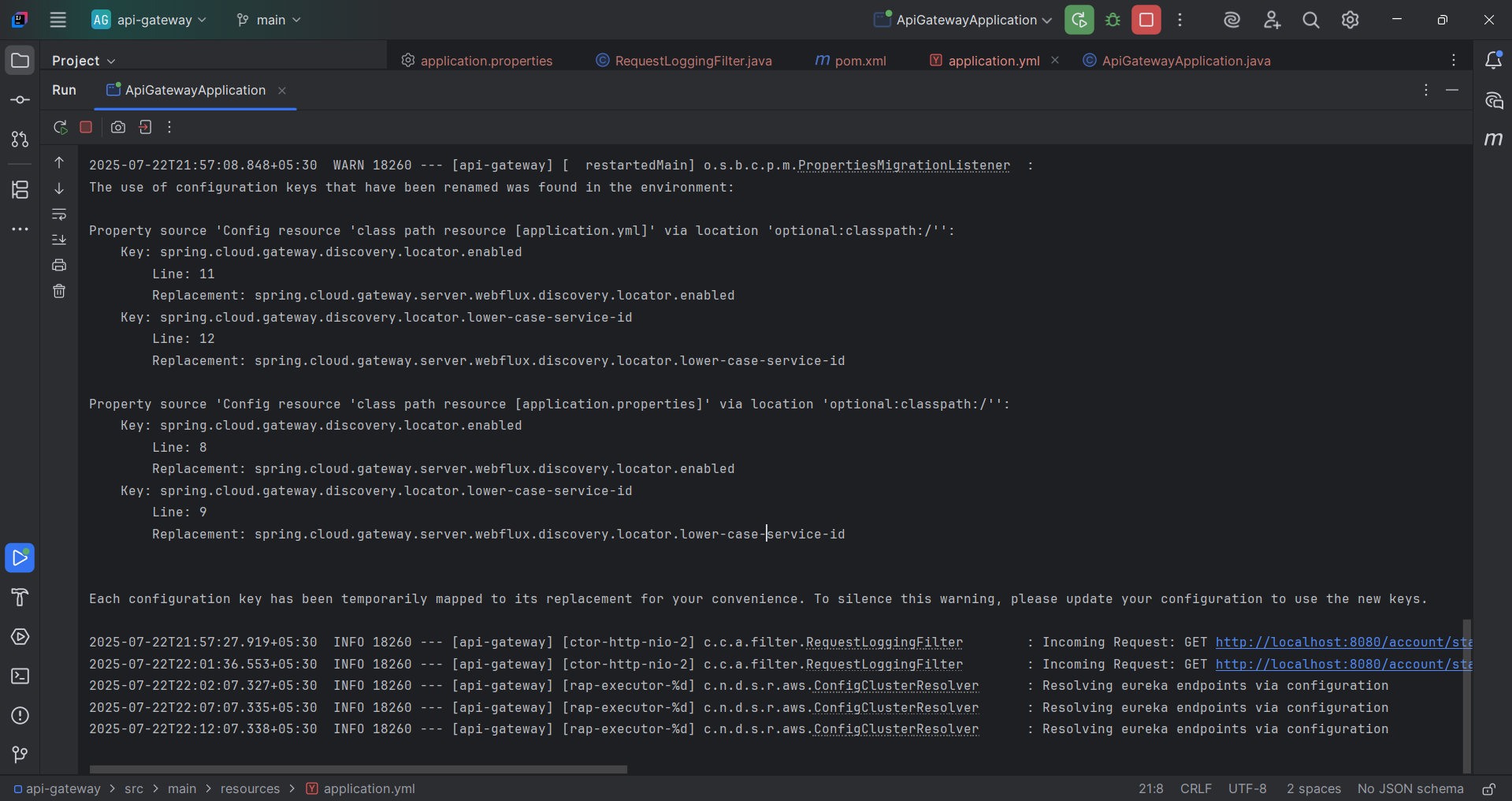
server:  
 port: 8080  
  
spring:  
 application:  
 name: api-gateway  
 cloud:  
 gateway:  
 discovery:  
 locator:  
 enabled: true  
 lower-case-service-id: true  
 routes:  
 - id: account-service  
 uri: lb://account-service  
 predicates:  
 - Path=/account/\*\*  
 main:  
 web-application-type: reactive  
  
eureka:  
 client:  
 service-url:  
 defaultZone: http://localhost:8761/eureka

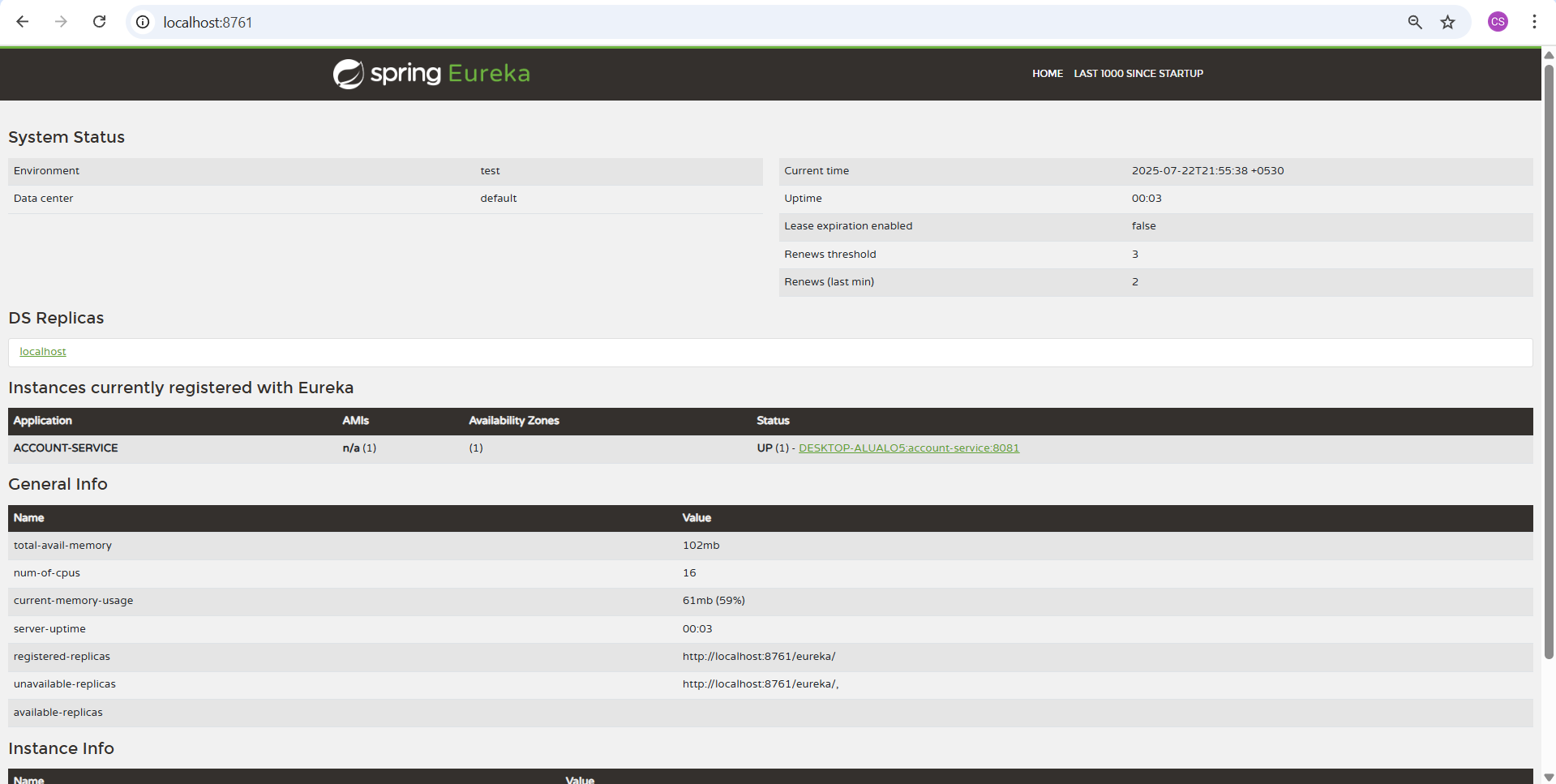
****

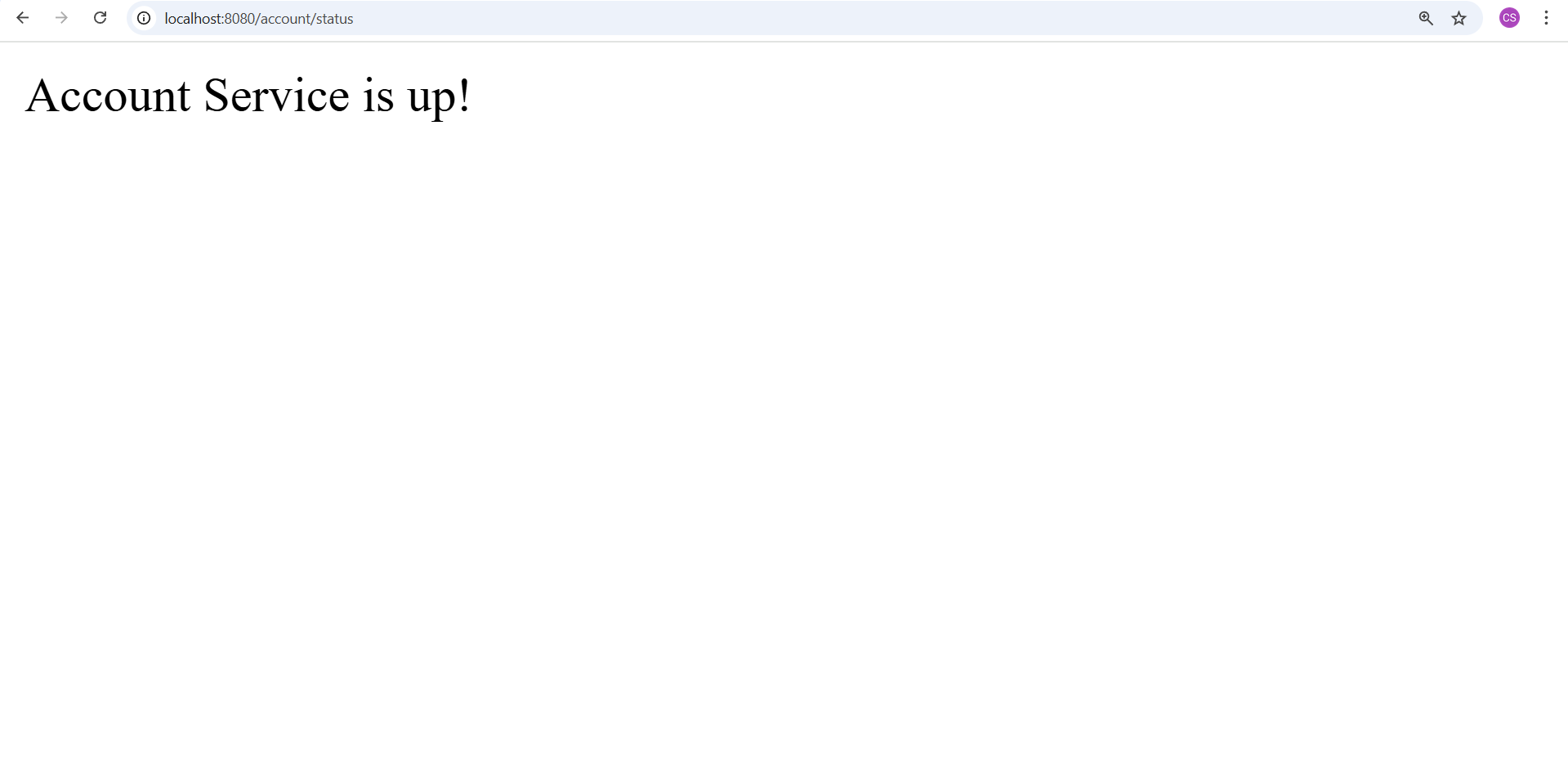
**Verify account routing via API Gateway(http://localhost:8080/account/status returns "Account Service is up!")**

****

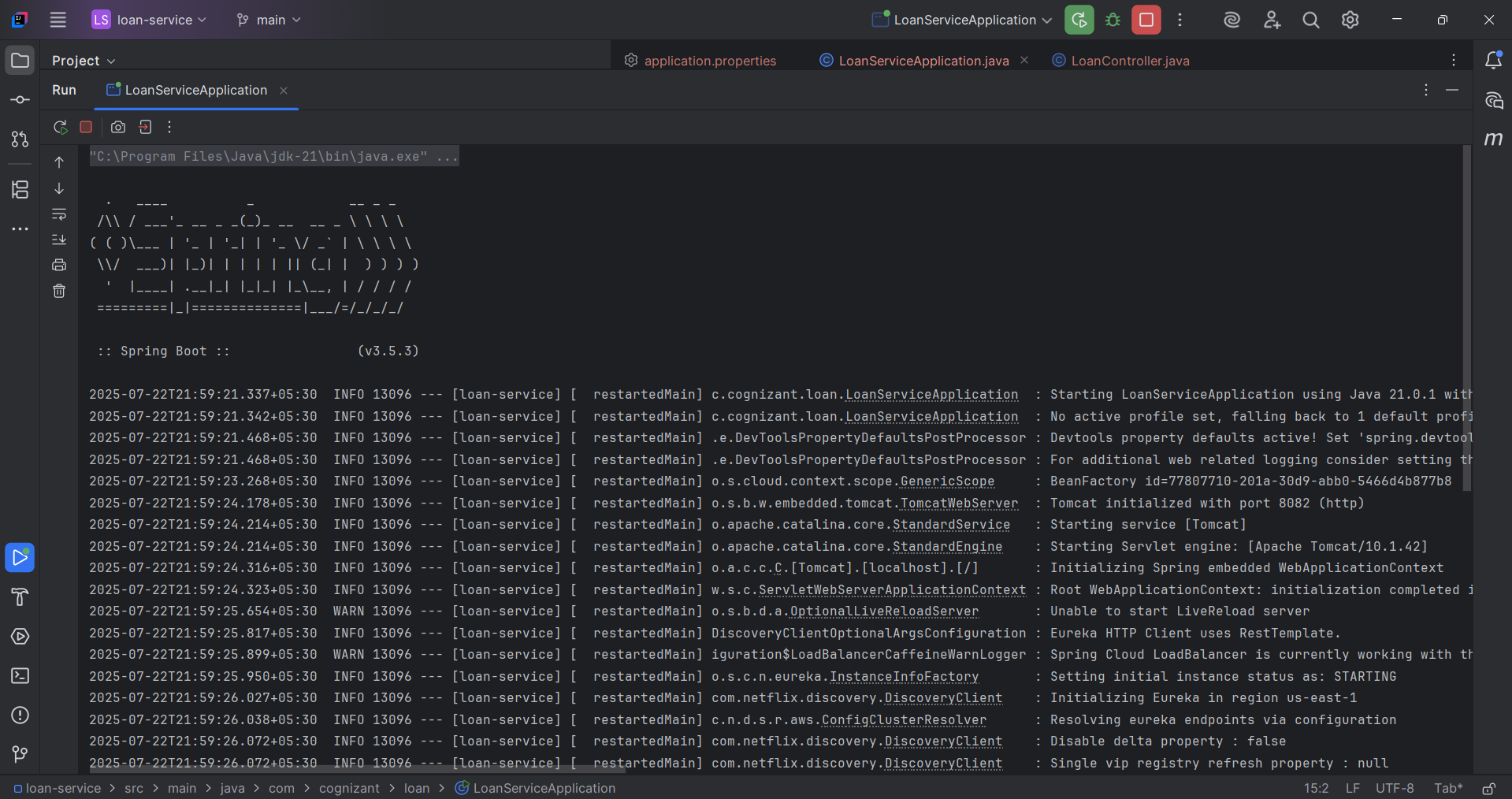
****

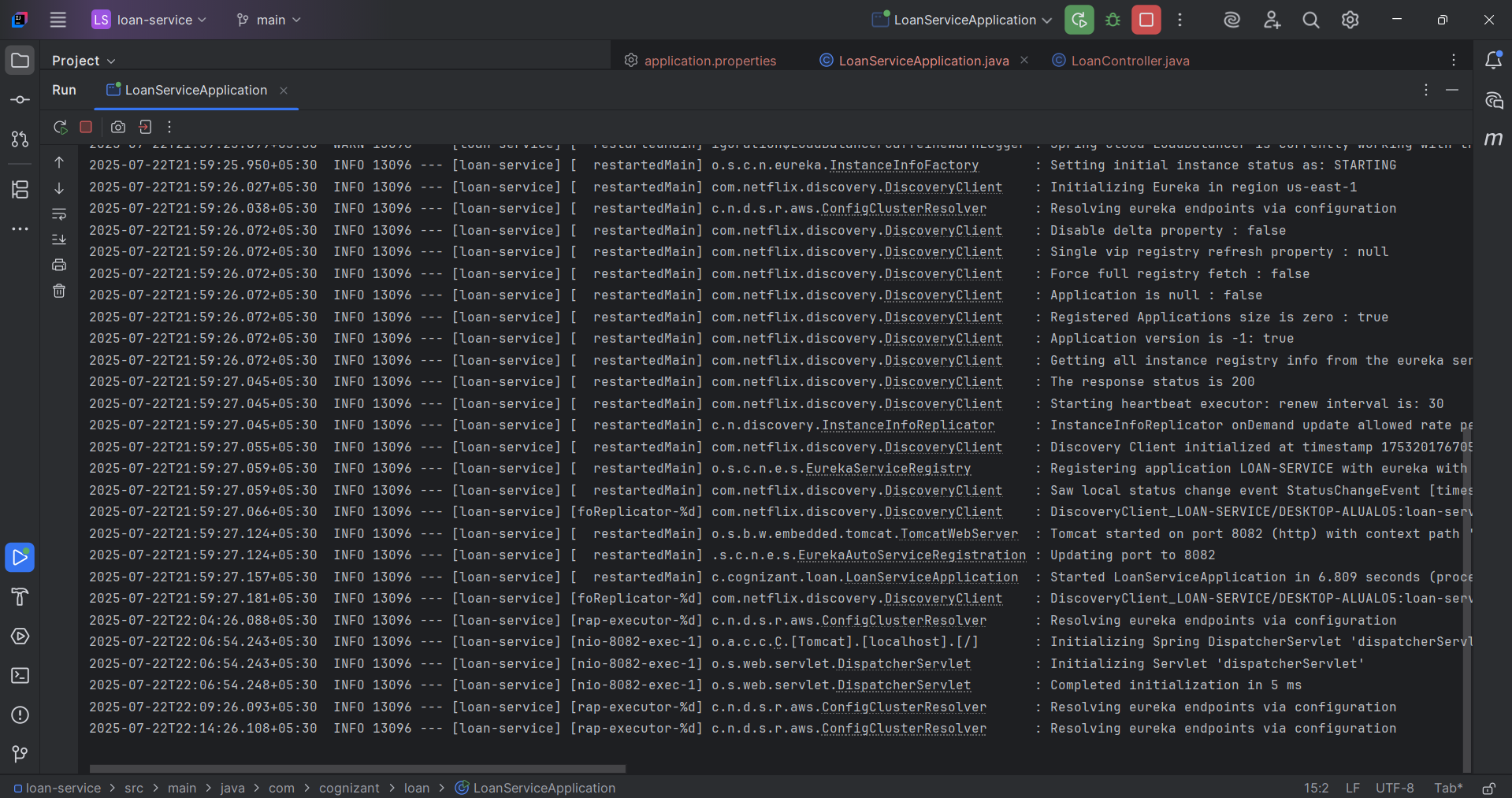
****

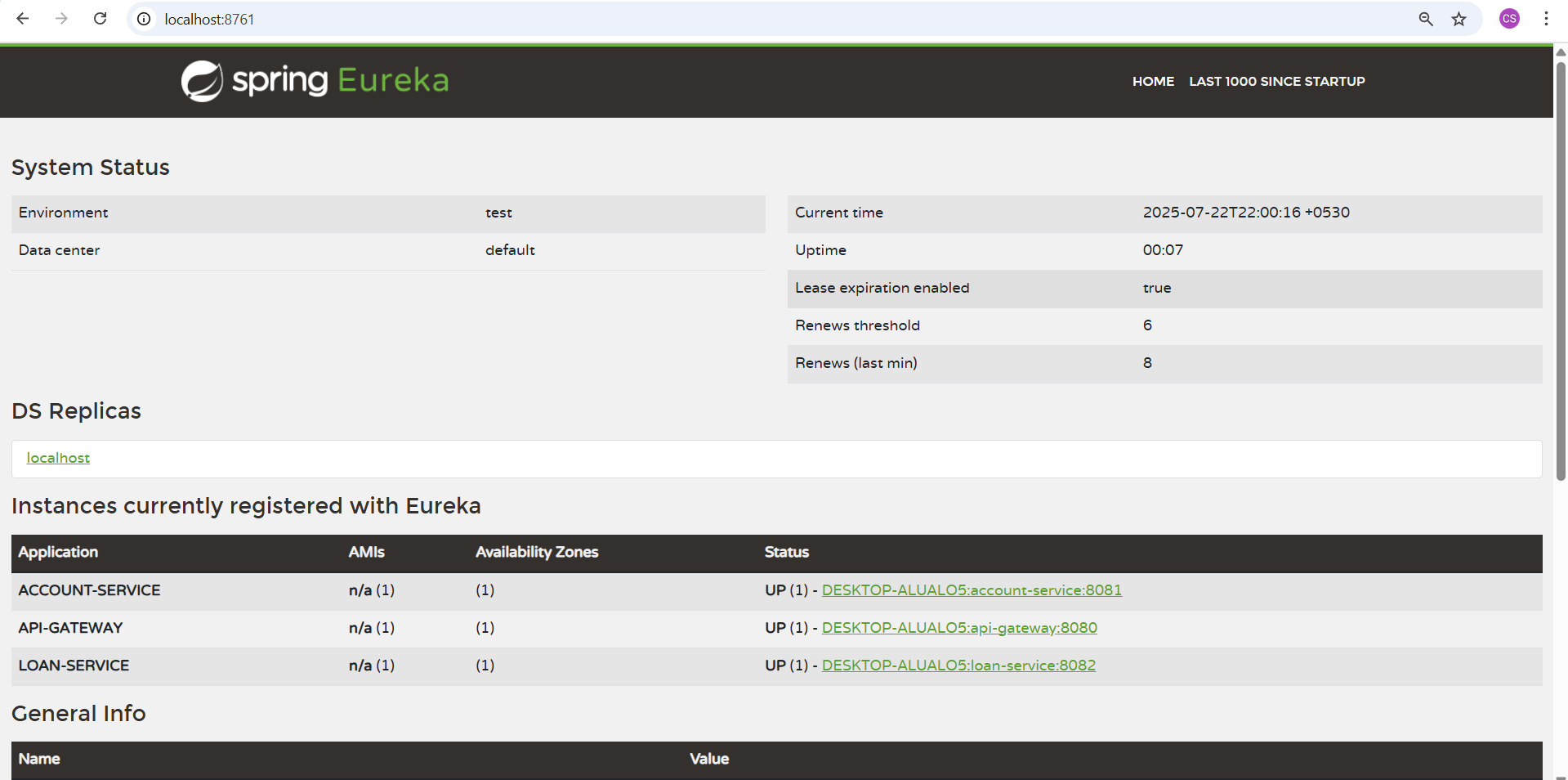
****

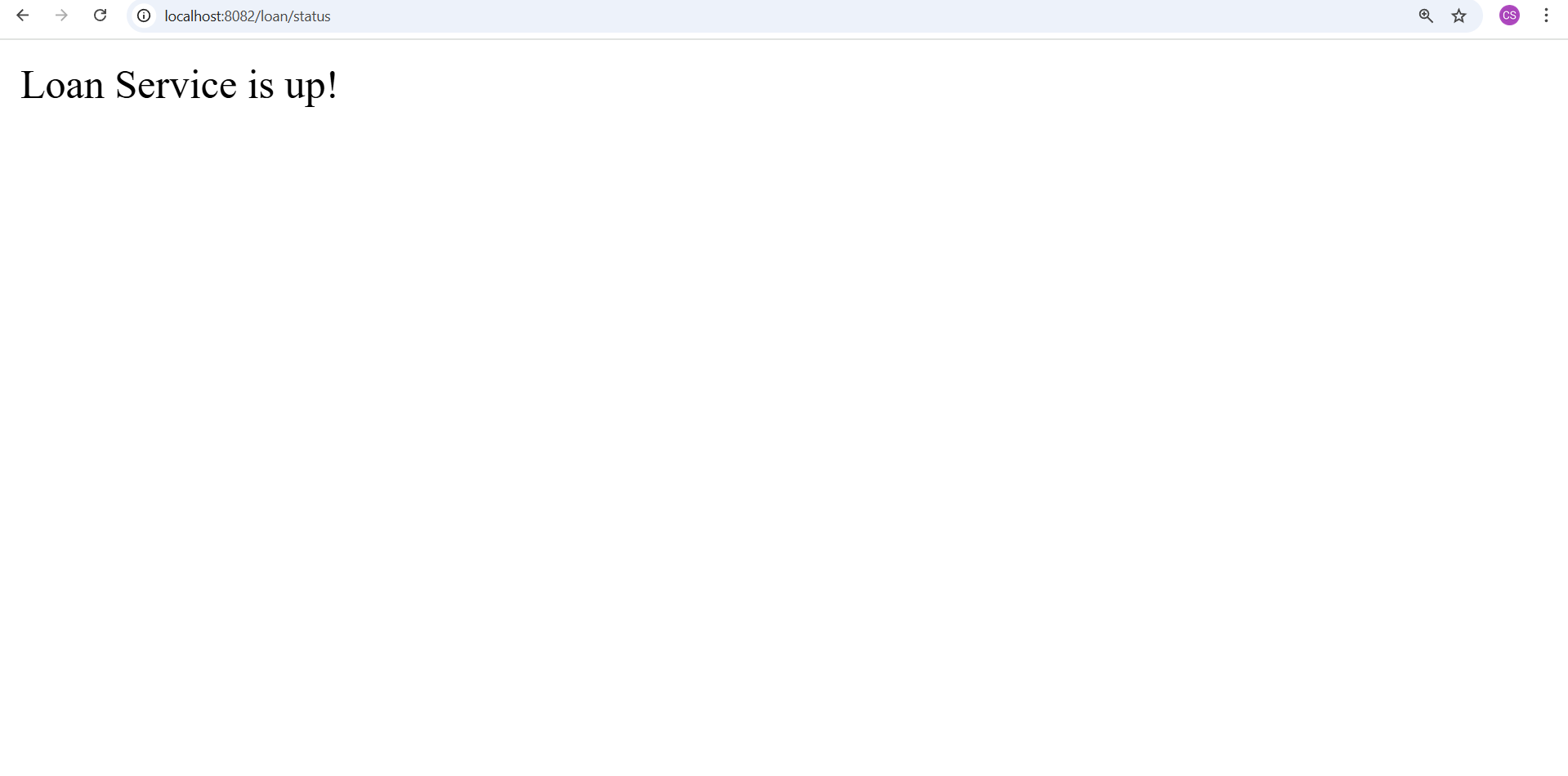
****

**Verify loan routing via API Gateway(http://localhost:8082/loan/status returns "Loan Service is up!")**

****

****

****

****

**Summary**

| **Step** | **Description** |
| --- | --- |
| Eureka Server Setup | Created and launched eureka-discovery-server on port 8761. |
| Account Service Registered | Created account-service, registered with Eureka, exposed /account/status. |
| Loan Service Registered | Created loan-service, registered with Eureka, exposed /loan/status. |
| API Gateway Setup | Created api-gateway, configured routes for account and loan services. |
| Routing via Gateway Verified | Successfully hit /account/status and /loan/status through gateway. |
| Global Filter Configured | Added request logging filter to log incoming gateway requests. |