**Creating Microservices for account and loan**

**EurekaDiscoveryServerApplication.java:**

package com.cognizant.eureka\_discovery\_server;

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:**

spring.application.name=eureka-discovery-server

server.port=8761

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

**AccountServiceApplication.java:**

package com.cognizant.account\_service;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class AccountServiceApplication {

public static void main(String[] args) {

SpringApplication.run(AccountServiceApplication.class, args);

}

}

**AccountController.java:**

package com.cognizant.account\_service;

import org.springframework.web.bind.annotation.\*;

import java.util.\*;

@RestController

@RequestMapping("/accounts")

public class AccountController {

@GetMapping("/{number}")

public Map<String, Object> getAccount(@PathVariable String number) {

Map<String, Object> data = new HashMap<>();

data.put("number", number);

data.put("type", "savings");

data.put("balance", 234343);

return data;

}

}

**Application.properties:**

spring.application.name=account-service

server.port=8080

spring.application.name=account-service

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

**LoanServiceApplication.java:**

package com.cognizant.loan\_service;

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\_service;

import java.util.HashMap;

import java.util.Map;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/loans")

public class LoanController {

@GetMapping("/{number}")

public Map<String, Object> getLoan(@PathVariable String number) {

Map<String, Object> data = new HashMap<>();

data.put("number", number);

data.put("type", "car");

data.put("loan", 400000);

data.put("emi", 3258);

data.put("tenure", 18);

return data;

}

}

**application.properties:**

spring.application.name=loan-service

server.port=8081

spring.application.name=loan-service

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

**ApiGatewayApplication.java:**

package com.cognizant.api\_gateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

**application.properties:**

spring.application.name=api-gateway

server.port=9090

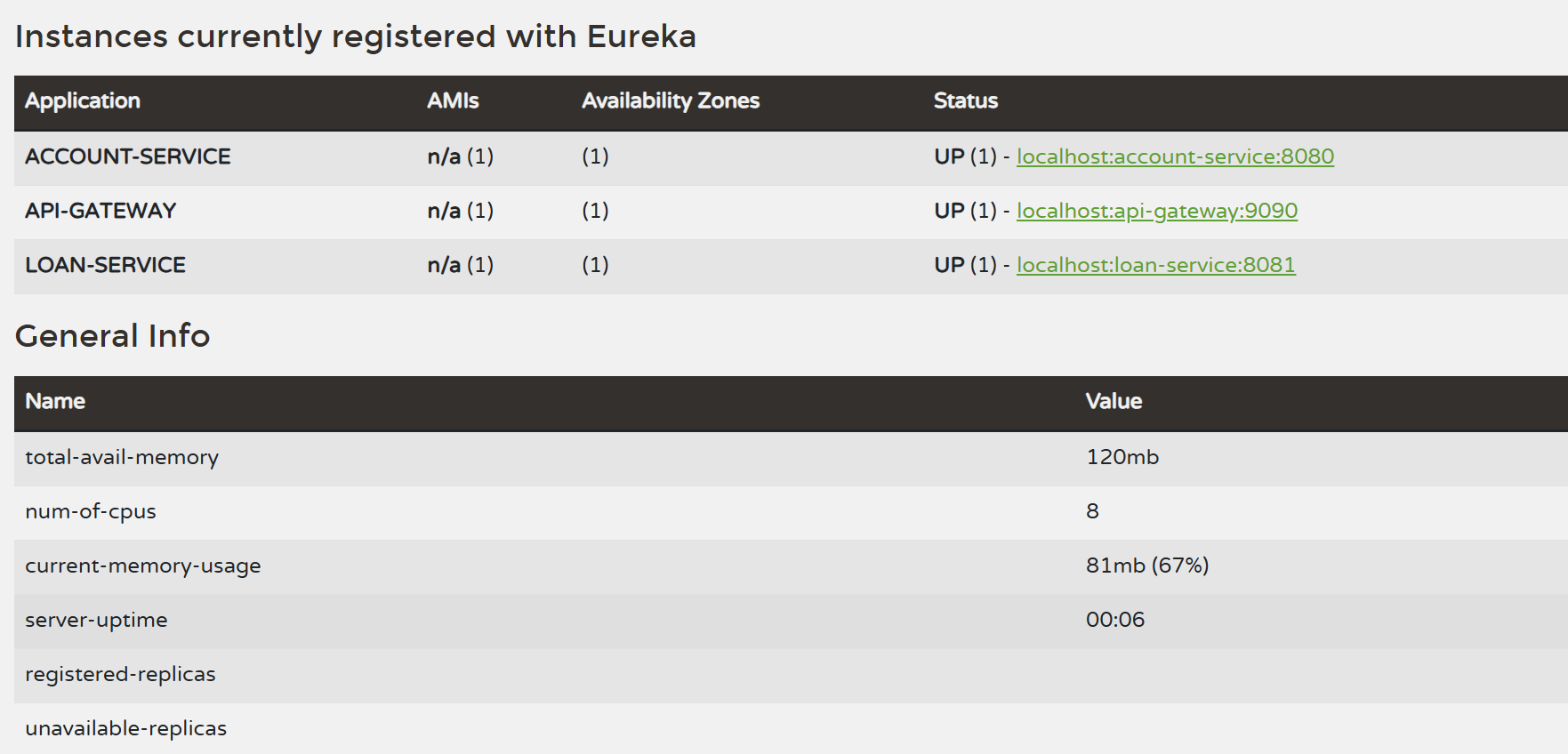
spring.application.name=api-gateway

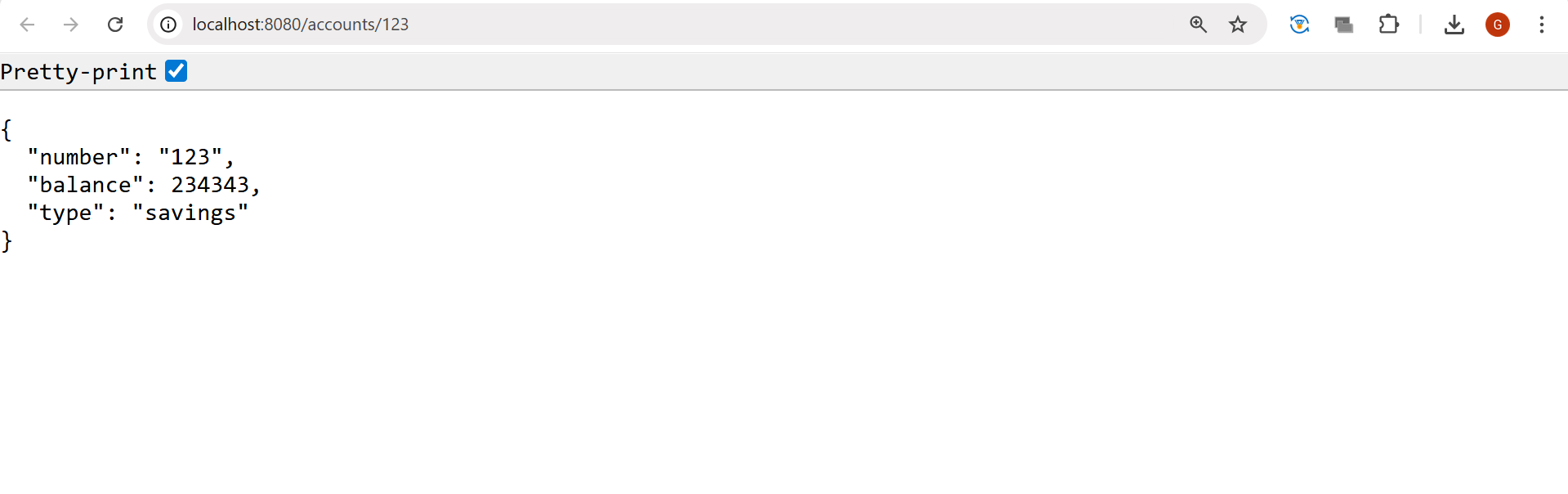
eureka.client.service-url.defaultZone=http://localhost:8761/eureka

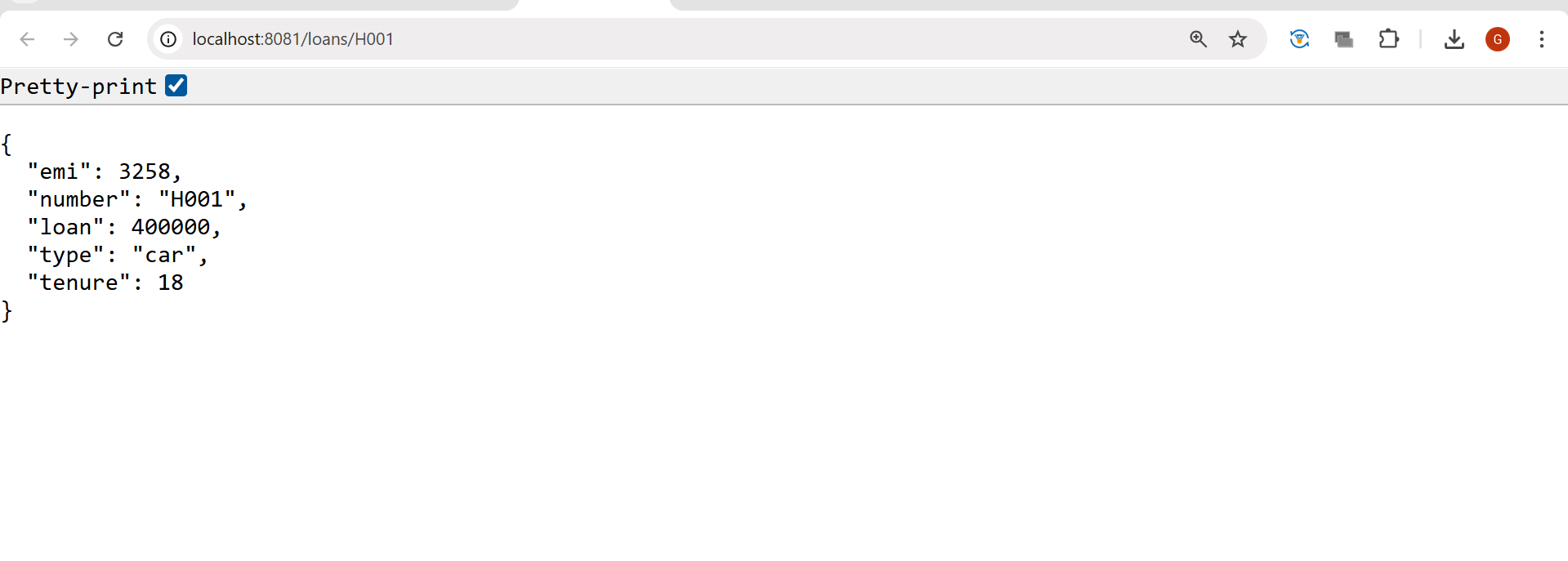
spring.cloud.gateway.discovery.locator.enabled=true

spring.cloud.gateway.discovery.locator.lower-case-service-id=true

**Output:**

****

****

****