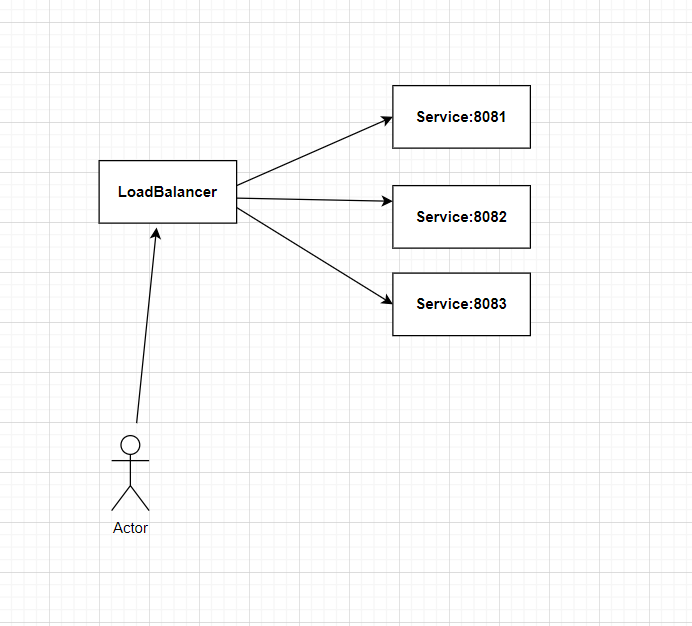
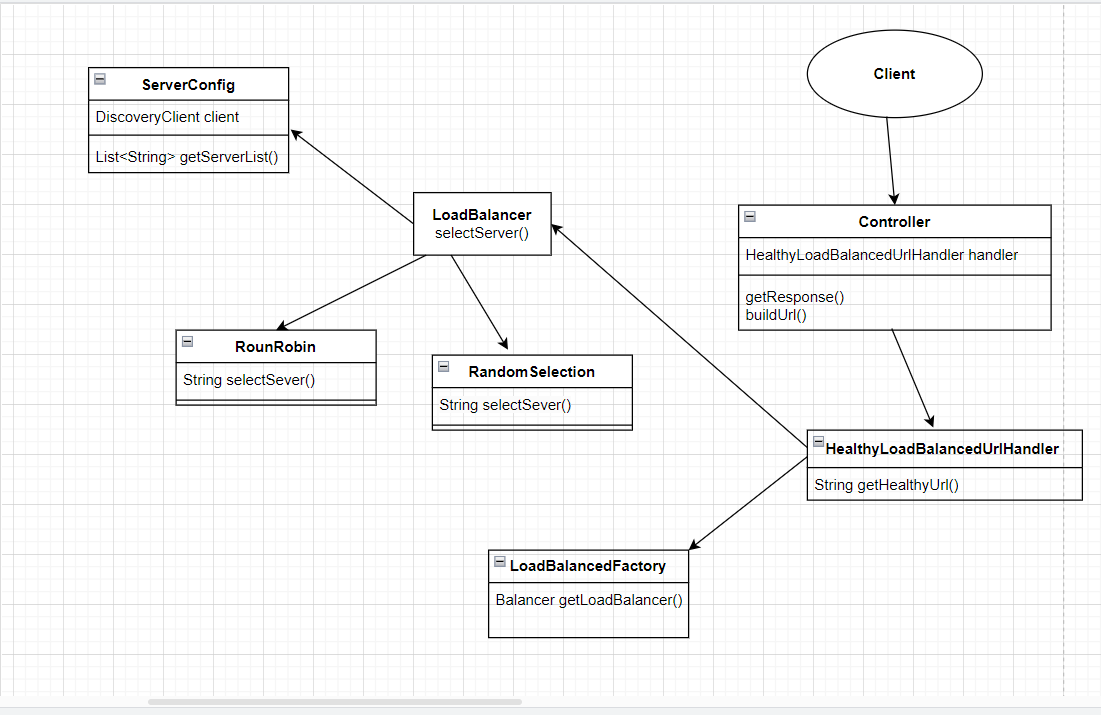
# High Level Design



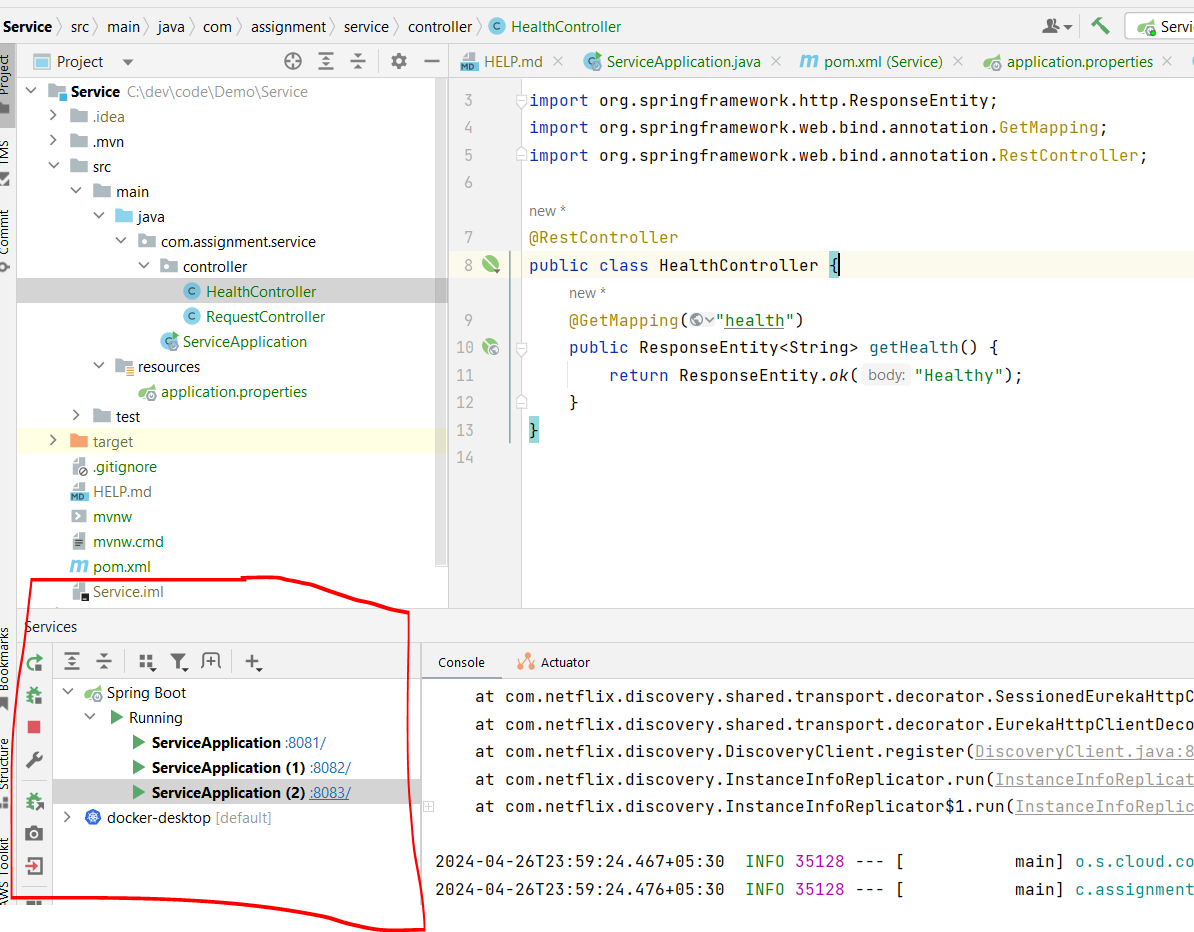
# Low Level Design

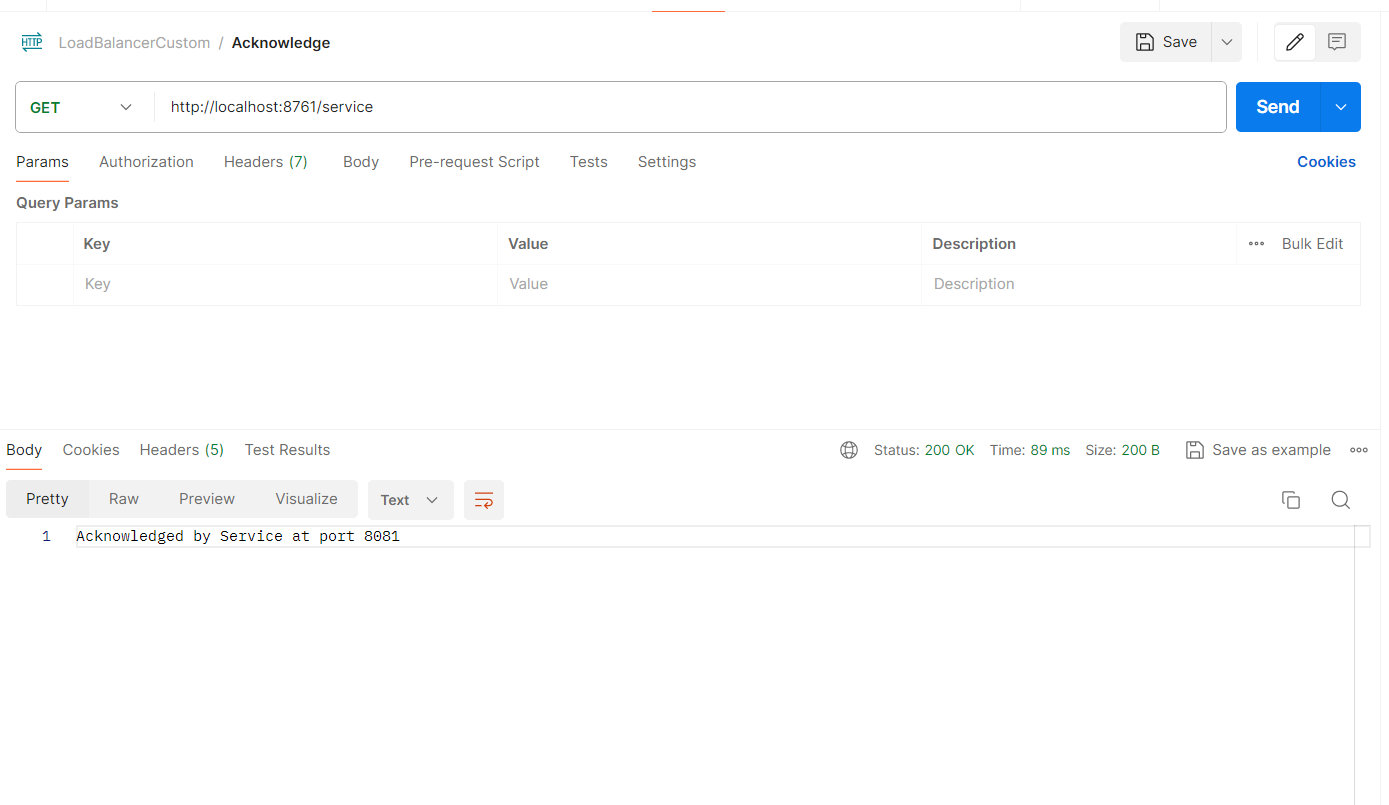
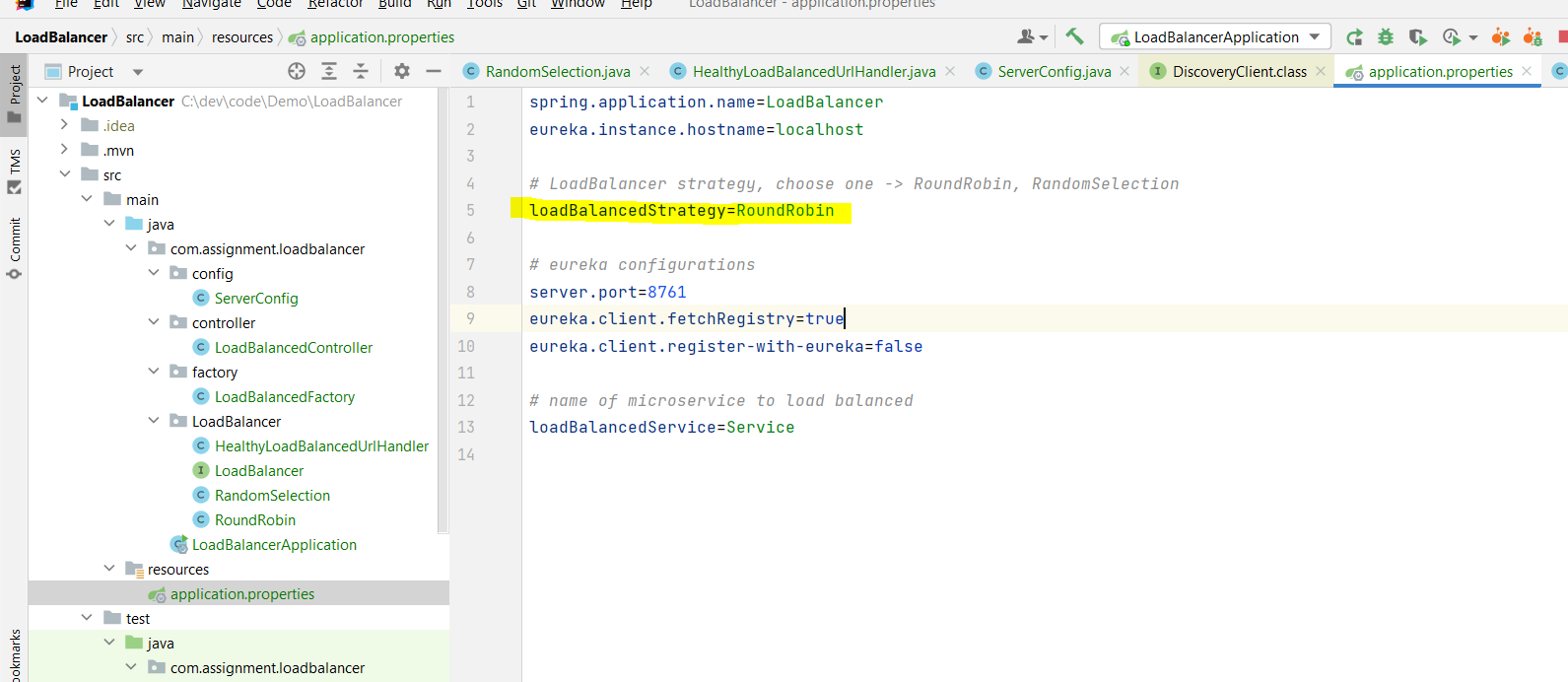


**Approach**

1. I have created a microservice **LoadBalancer** which will also work as EurekaServer, and another microservice named **Service** which will work as eureka clients, all of **Service** instance will register themselves to **LoadBalancer** Microservice.
2. Every enpoint first go to LoadBalancer microservice and from their it routes to Service microservice.

# How to use it?

1. Run both projects locally, we need java 17 here.
2. LoadBalancer microservice don’t need any additional steps but for **Service** microservice, we need to make sure we make multiple instances of it in IntelliJ. Like I did for my case.  
     
   A screenshot of a computer

   Description automatically generated
3. Once we start all services, you can use my postman collection. And run Acknowledge endpoint.   
   **Note:** Since eureka client take time to connect with eureka server, hence initial endpoint could take time around 30 secs for that a retry mechanism is created until client become available to server.  
   Once all client will be registered on server, request won’t take much time.  
   Since I am using RoundRobin strategy here, if I make continous request, I will be able to see different port each time in same order every time. The order however could be in any format like 2->1->3.
4. I can change strategy of load balancer inside application.properties of Loadbalancer microservice.  
   
5. There is also a health check implemented so we can stop any 1 instance of **Service** microservice and it will work fine.