**Chủ đề: Xây dựng webservice theo kiến trúc microservice**

Người thực hiện: Hoàng Mạnh Dũng – LA31 PM

Hà Nội – 2023

Phụ lục

[1. Mô hình khái quát hệ thống webservice xây dựng dựa trên kiến trúc microservice 4](#_Toc140268130)

[1.1 Mô hình 4](#_Toc140268131)

[1.1.1 Discovery server: 4](#_Toc140268132)

[1.1.2 Api gateway: 5](#_Toc140268133)

[1.1.3 Auth service: 5](#_Toc140268134)

[1.1.4 Employee service: 5](#_Toc140268135)

[1.1.5MySQL: 5](#_Toc140268136)

[2. Cách tạo các server, service trong dự án 5](#_Toc140268137)

[2.1 Discovery server 5](#_Toc140268138)

[2.2 Auth server 6](#_Toc140268139)

[2.3 Api gateway 6](#_Toc140268140)

[2.4 Employee server 7](#_Toc140268141)

[3. Cấu hình chi tiết cho từng server 7](#_Toc140268142)

[3.2 Discovery server 7](#_Toc140268143)

[3.2 Auth server 8](#_Toc140268144)

[3.3 Employee server 10](#_Toc140268145)

[3.4 Api gateway 11](#_Toc140268146)

[3.5 Cách đăng kí service trực tiếp vào api gateway không qua discovery server(không khuyến khích) 13](#_Toc140268147)

[4. Khởi chạy các server 14](#_Toc140268148)

[5. Xây dựng các api của employee service 15](#_Toc140268149)

[5.1 Tạo entity employee 15](#_Toc140268150)

[5.2 Tạo repository bằng JpaRepository 15](#_Toc140268151)

[5.3 Tạo service 15](#_Toc140268152)

[5.4 Tạo controller 16](#_Toc140268153)

[6. Xây dựng api của auth service 18](#_Toc140268154)

[6.1 Tạo entity account 18](#_Toc140268155)

[6.2 Tạo repository bằng JpaRepository 19](#_Toc140268156)

[6.3 Tạo service 19](#_Toc140268157)

[6.3.1 Auth service 19](#_Toc140268158)

[6.3.2 Password service 19](#_Toc140268159)

[6.4 Tạo controller 20](#_Toc140268160)

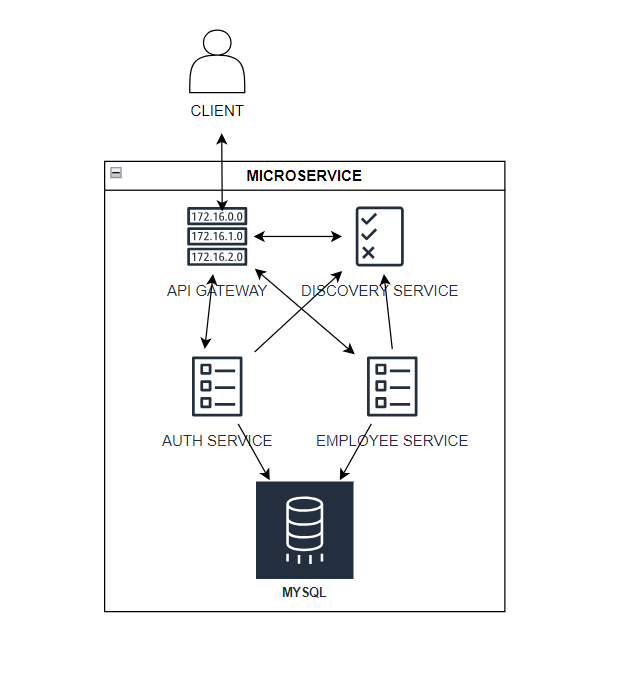
[7. Kiểm tra api gateway 21](#_Toc140268161)

[7.1 Empoyee service 21](#_Toc140268162)

[7.2 Auth service 24](#_Toc140268163)

# 1. Mô hình khái quát hệ thống webservice xây dựng dựa trên kiến trúc microservice

## 1.1 Mô hình



### 1.1.1 Discovery server:

Là 1 server dùng để đăng kí dịch vụ trong hệ thống microservice. Trong trường hợp này ta sẽ dùng eureka server của spring cloud. Eureka server sẽ tự động nhận tất cả các endpoint api của các service đăng kí đến nó. Eureka server sẽ chia sẻ các endpoint api cho api gateway kết nối đến nó.

### 1.1.2 Api gateway:

Là một server đóng vai trò là proxy điều hướng đến đúng các service trong hệ thống thông qua các endpoint api của service đã đăng kí trong discovery server. Để nhận được các endpoint api thì api gateway server phải kết nối đến eureka server với dạng là 1 eureka client

### 1.1.3 Auth service:

Là 1 server spring boot và cũng là 1 eureka client đảm nhiệm việc đăng kí, đăng nhập cho dự án.

### 1.1.4 Employee service:

Là 1 server spring boot và cũng là 1 eureka client đảm nhiệm việc thao tác dữ liêu nhân viên trong dự án.

### 1.1.5MySQL:

Là hệ quả trị cơ sở dữ liệu sẽ dùng cho dự án

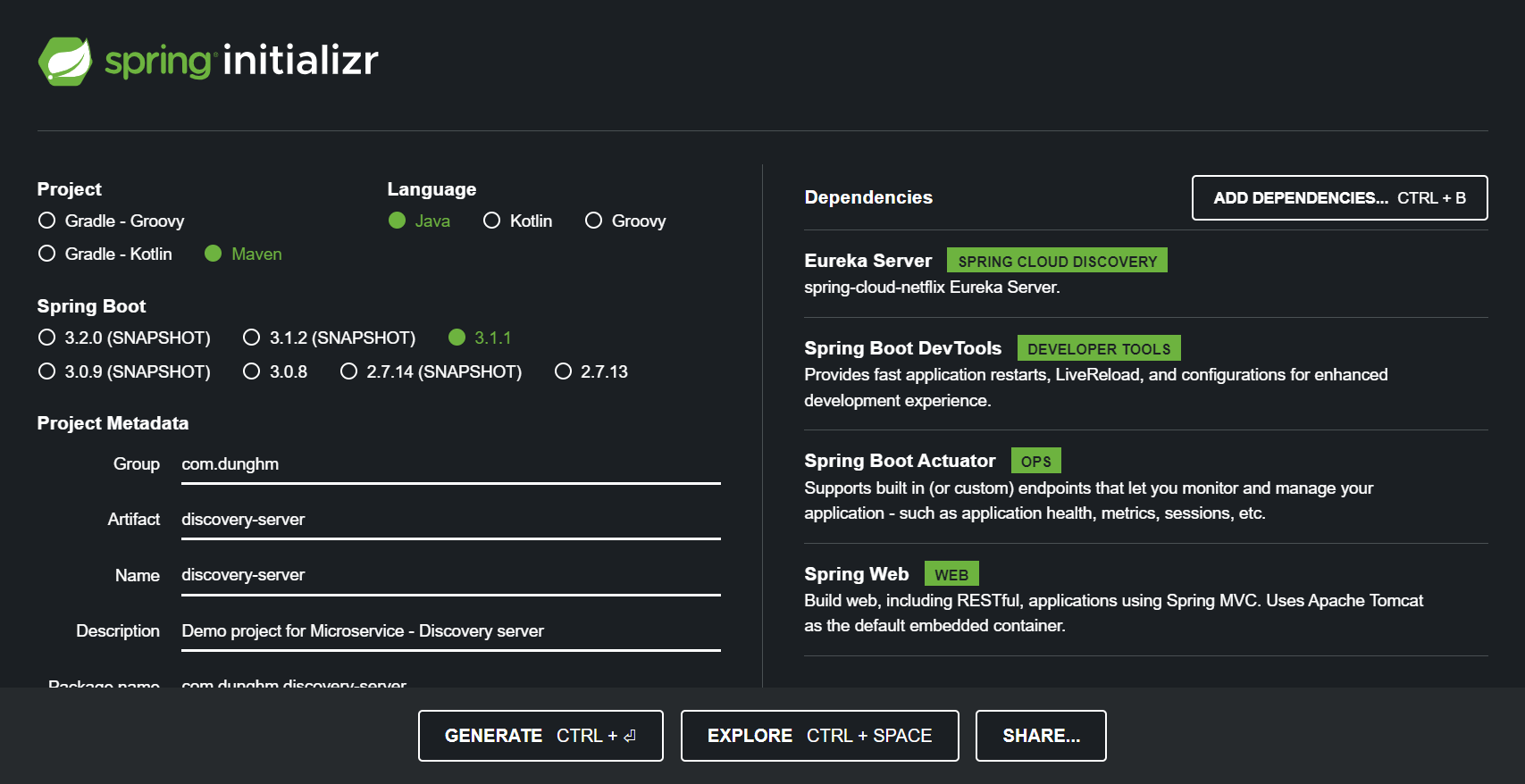
# 2. Cách tạo các server, service trong dự án

Chú ý : trong kiến trúc microservice có thể không cần discovery server để đăng kí các service khác mà có thể trực tiếp đăng kí các service đó vào trong api gateway, tuy nhiên điều này không được khuyến khích vì khi mở rộng hệ thống ta sẽ phải đăng kí thủ công các service mới vào api gateway còn với discovery server điều này được diễn ra tự động. Bên dưới cũng sẽ hướng dẫn các đăng kí các service vào api gateway 1 cách thủ công.

Có thể tạo các project spring boot cho từng service tương ứng tại trang web: <https://start.spring.io/>

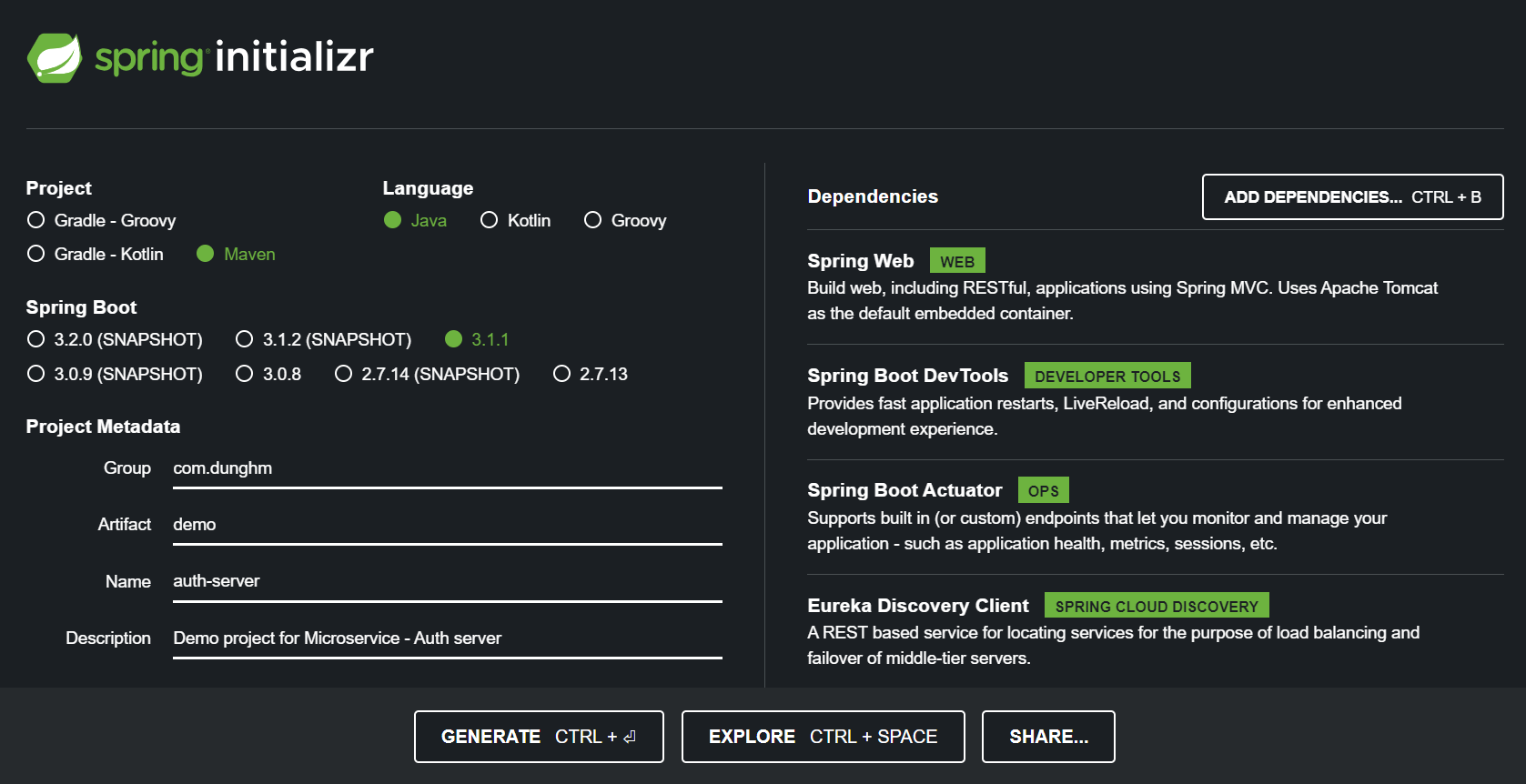
## 2.1 Discovery server

Chon các dependencies cần thiết như: eureka server, spring boot actuator



## 2.2 Auth server

Chon các dependencies cần thiết như: eureka client, spring boot actuator



## 2.3 Api gateway

Chon các dependencies cần thiết như: eureka client, api gateway của spring cloud, spring boot actuator



## 2.4 Employee server

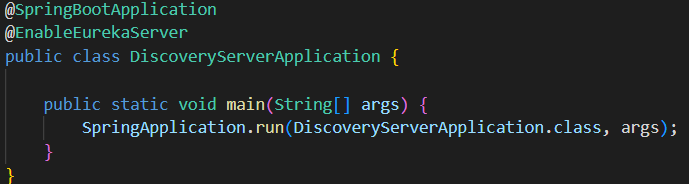
Chon các dependencies cần thiết như: eureka client, spring boot actuator



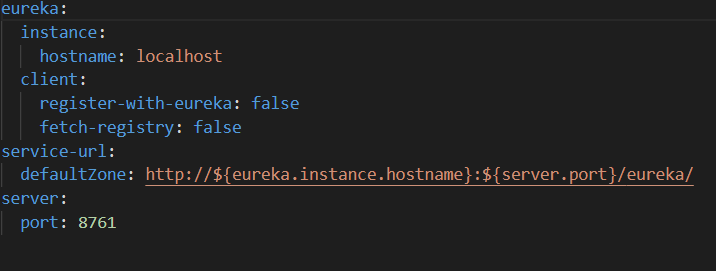
# 3. Cấu hình chi tiết cho từng server

## 3.2 Discovery server

Tại file application của server ta thêm annotation @EnableEurekaServer để đánh dấu đây là 1 eureka server



Tại file application.yml ta cấu hình như hostname, port



Đây là đường dẫn của server để các eureka client đăng kí quá đường dẫn này

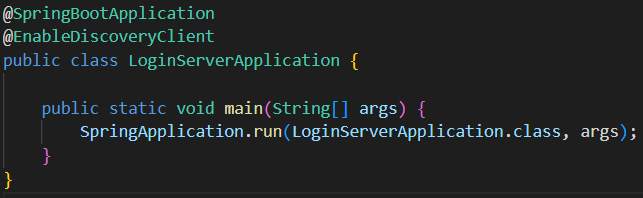


Đặt 2 thuộc tính này bằng false để không tự đăng kí bản thân eureka server lên server của chính nó



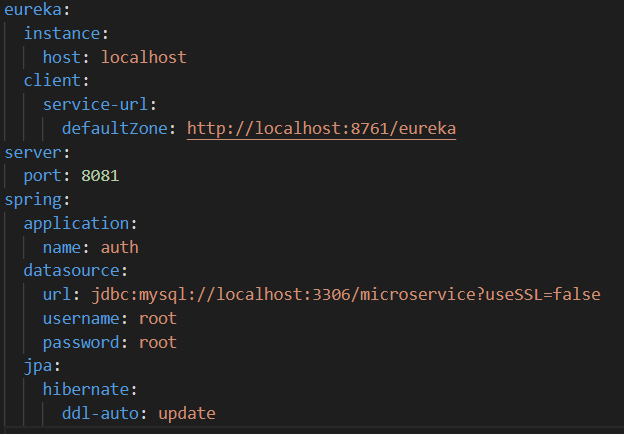
## 3.2 Auth server

Tại class application thêm các annotaion @EnableDiscoveryClient để dánh dấu đây là 1 eureka client mới có thể đăng kí lên eureka server

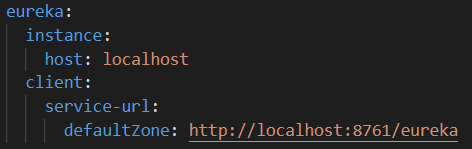


Tại file application.yml

Khai báo các thuộc tính như port, name(là tên được đăng kí lên eureka service), đường dẫn database

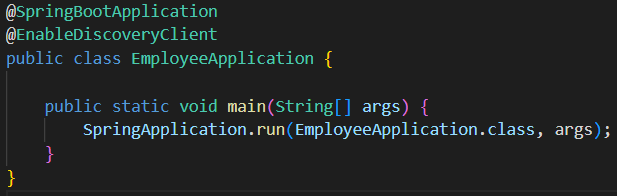


Auth server được đăng kí lên eureka server qua đường dẫn mà ta đã khai báo trong file yml của eureka server



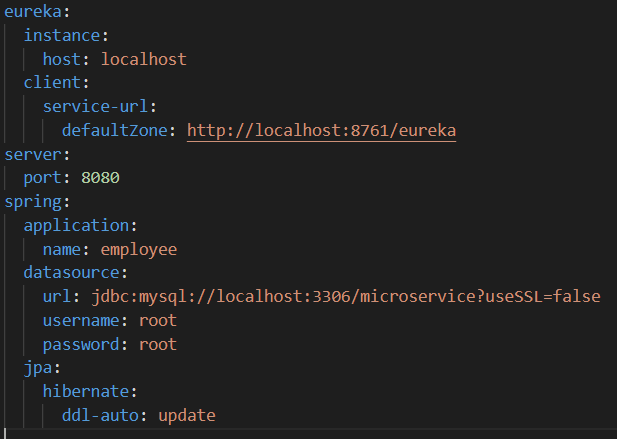
## 3.3 Employee server

Tại class application thêm các annotaion @EnableDiscoveryClient để dánh dấu đây là 1 eureka client mới có thể đăng kí lên eureka server

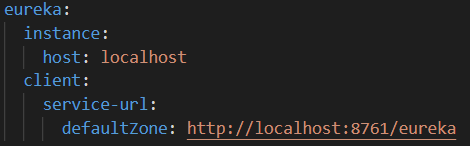


Tại file application.yml

Khai báo các thuộc tính như port, name(là tên được đăng kí lên eureka service), đường dẫn database

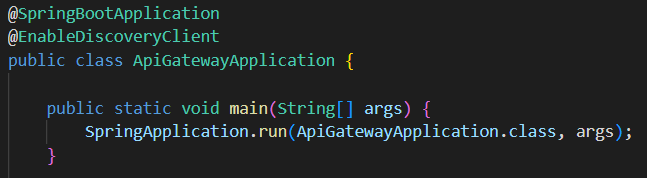


Employee server được đăng kí lên eureka server qua đường dẫn mà ta đã khai báo trong file yml của eureka server



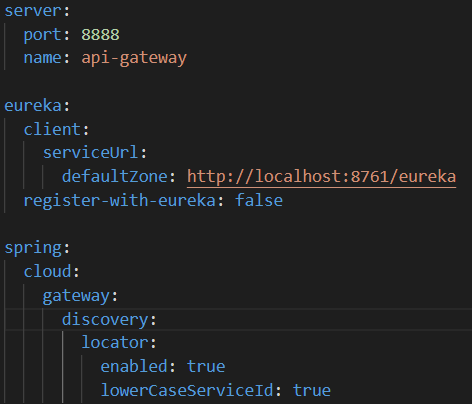
## 3.4 Api gateway

Tại class application thêm các annotaion @EnableDiscoveryClient để dánh dấu đây là 1 eureka client mới có thể đăng kí lên hoặc kết nối đến eureka server(trường hợp của api gateway có thể không cần đăng kí nhưng phải kết nối lên eureka server)

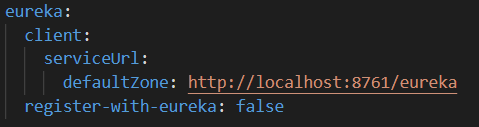


Tại file application.yml

Khai báo các thuộc tính như port, name

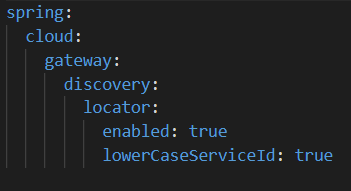


Kết nối đến eureka server nhưng không đăng kí



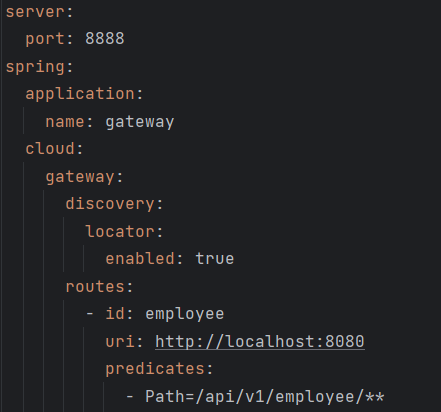
Cho phép lấy các endpoint api mà các service đăng kí trên eureka server

lowerCaseServiceId: true cho phép gọi đến các api qua tên(id service) service trên eureka server không kể chữ hoa hay chữ thường(mặc định tên được đăng kí lên eureka server sẽ là chữ in hoa)



## 3.5 Cách đăng kí service trực tiếp vào api gateway không qua discovery server(không khuyến khích)

Cấu hình tại file application.yml của api gateway

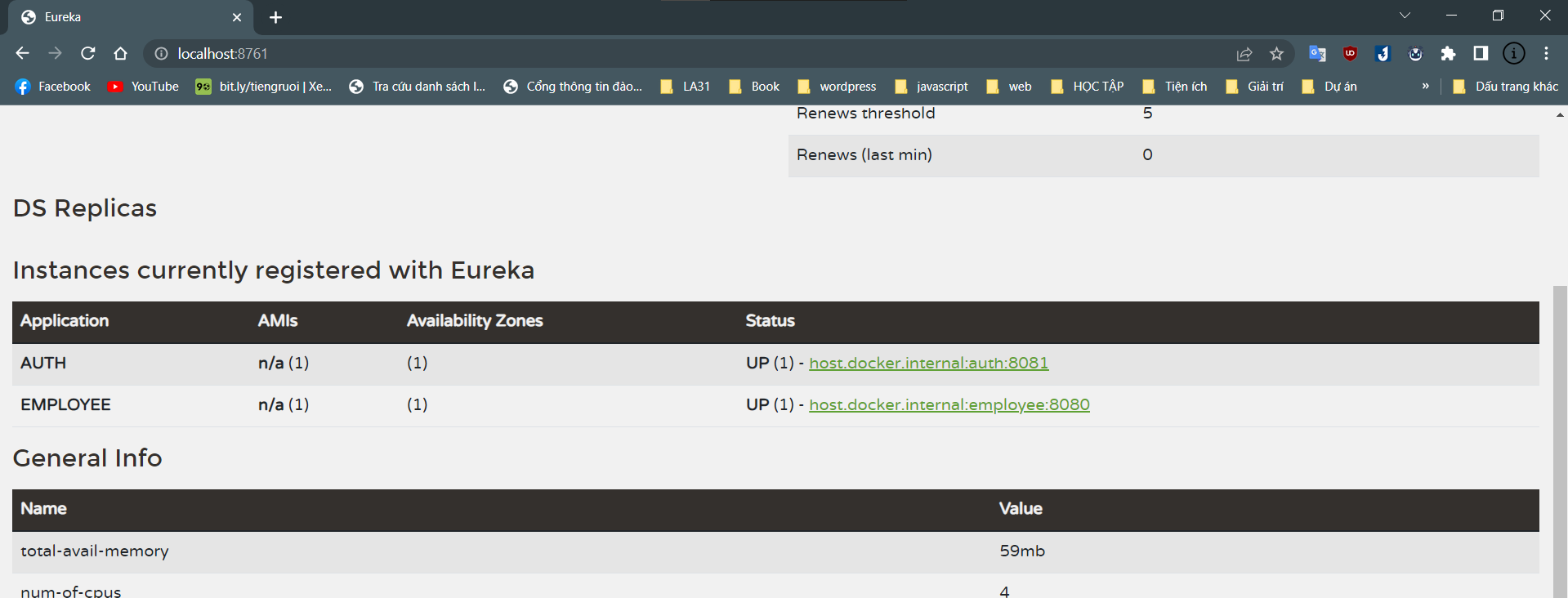


Phần cấu hình route ta sẽ phải khai báo trực tiếp base url api của service đó, trong trường hợp này là employee service

# 4. Khởi chạy các server

Ta sẽ chạy các server thông qua class application(nên chạy discovery server đầu tiên)

Kiểm tra trên eureka server tại <http://localhost:8761/> đã thấy 2 service bên trên đã được đăng kí thành công



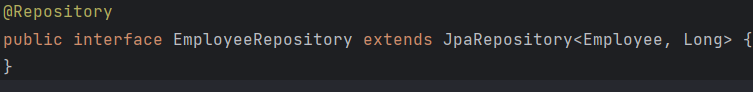
# 5. Xây dựng các api của employee service

## 5.1 Tạo entity employee



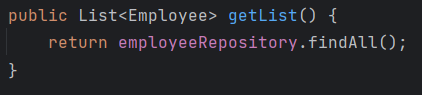
## 5.2 Tạo repository bằng JpaRepository

Để sử dụng các phương thức có sẵn của Jpa data thao tác với database

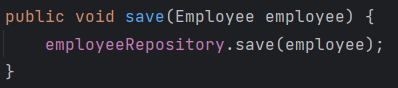


## 5.3 Tạo service

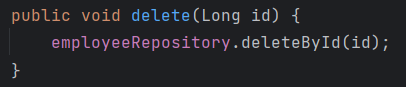
Trả về tất cả các bản ghi trong database dưới dạng List



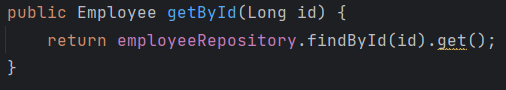
Lưu employee vào database



Xoá bản ghi thông qua id



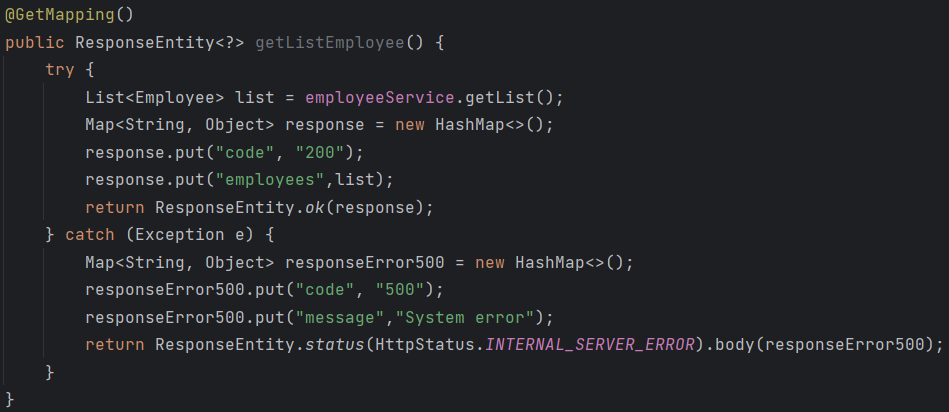
Lấy ra 1 bản ghi thông qua id



## 5.4 Tạo controller

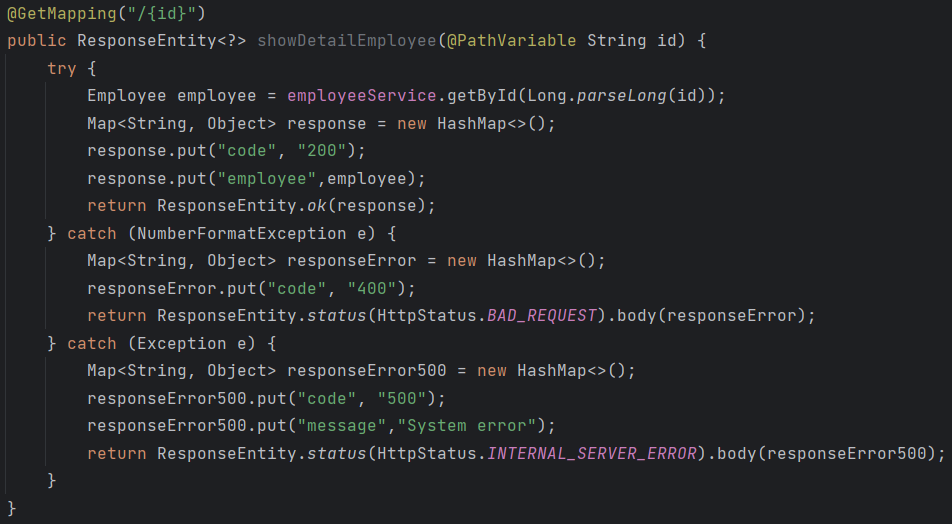
Trả về code 200 và list employee nếu lấy ra thành công, ngược lại trả về code 500 và message

<http://localhost:8080>



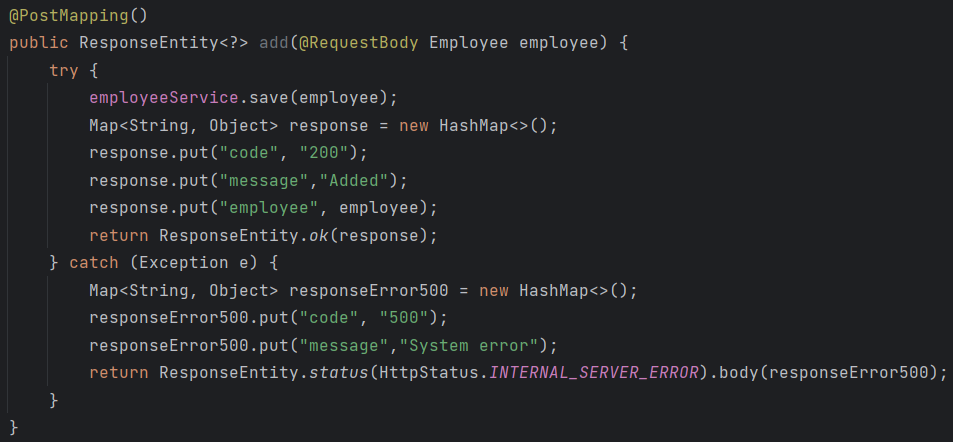
Lấy ra bản ghi employee theo id

[http://localhost:8080/{id}](http://localhost:8080/%7bid%7d)



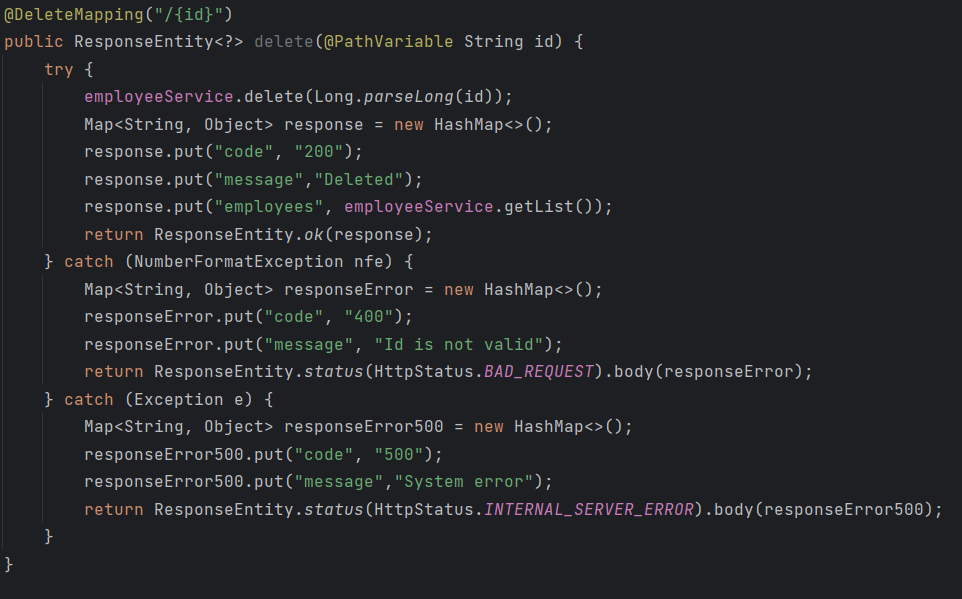
Lưu 1 đối tượng employee vào database và trả về code và message thông báo

<http://localhost:8080> Method: POST



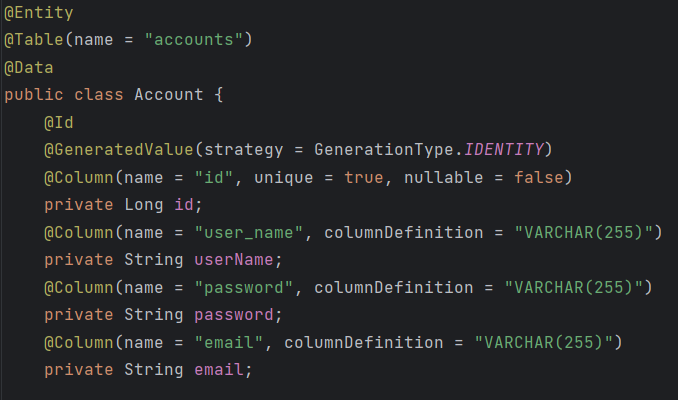
Xoá 1 đối tượng theo id và trả về danh sách bản ghi trong database để kiểm tra

[http://localhost:8080/{id}](http://localhost:8080/%7bid%7d) Method: DELETE



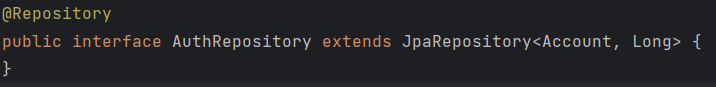
# 6. Xây dựng api của auth service

## 6.1 Tạo entity account



## 6.2 Tạo repository bằng JpaRepository

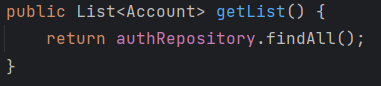
Để sử dụng các phương thức có sẵn của Jpa data thao tác với database



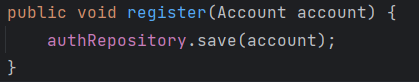
## 6.3 Tạo service

### 6.3.1 Auth service

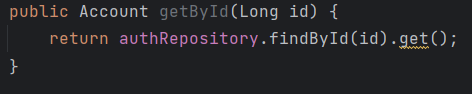
Trả về tất cả các bản ghi trong database dưới dạng List



Lưu account vào database



Lấy ra account theo id

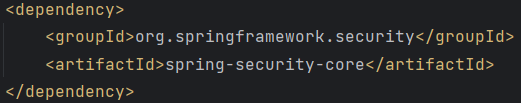


### 6.3.2 Password service

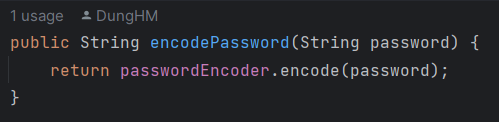
Sử dụng PasswordEncoder để mã hoá và giải mã mật khẩu



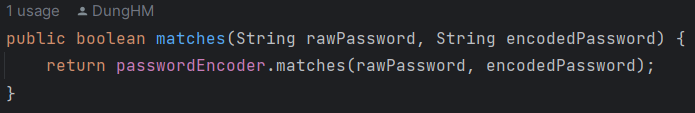
Có thể thêm PasswordEncoder thông ma add dependencies trong maven



Phương thức dùng để mã hoá password



Phương thức dùng để so sánh mật khẩu chưa mã hoá với mật khẩu đã mã hoá dùng trong đăng nhập



## 6.4 Tạo controller

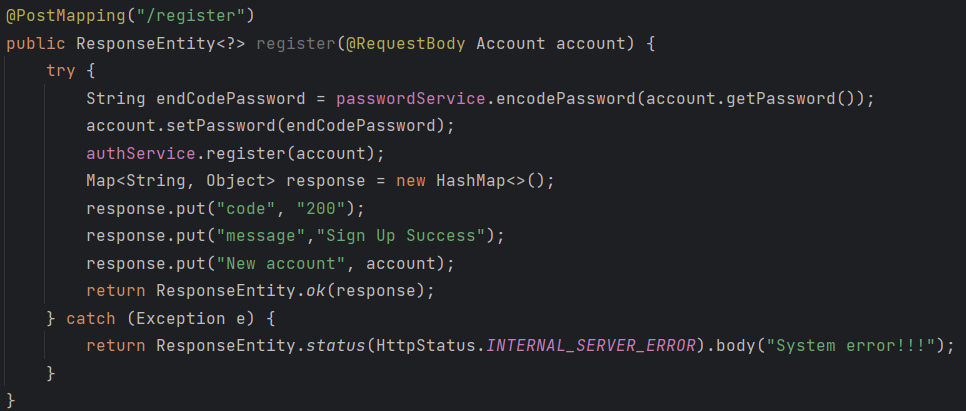
Controller dùng để login

<http://localhost:8081/login>



Controller dùng để đăng kí

<http://localhost:8081/register>



# 7. Kiểm tra api gateway

## 7.1 Empoyee service

Chú ý:

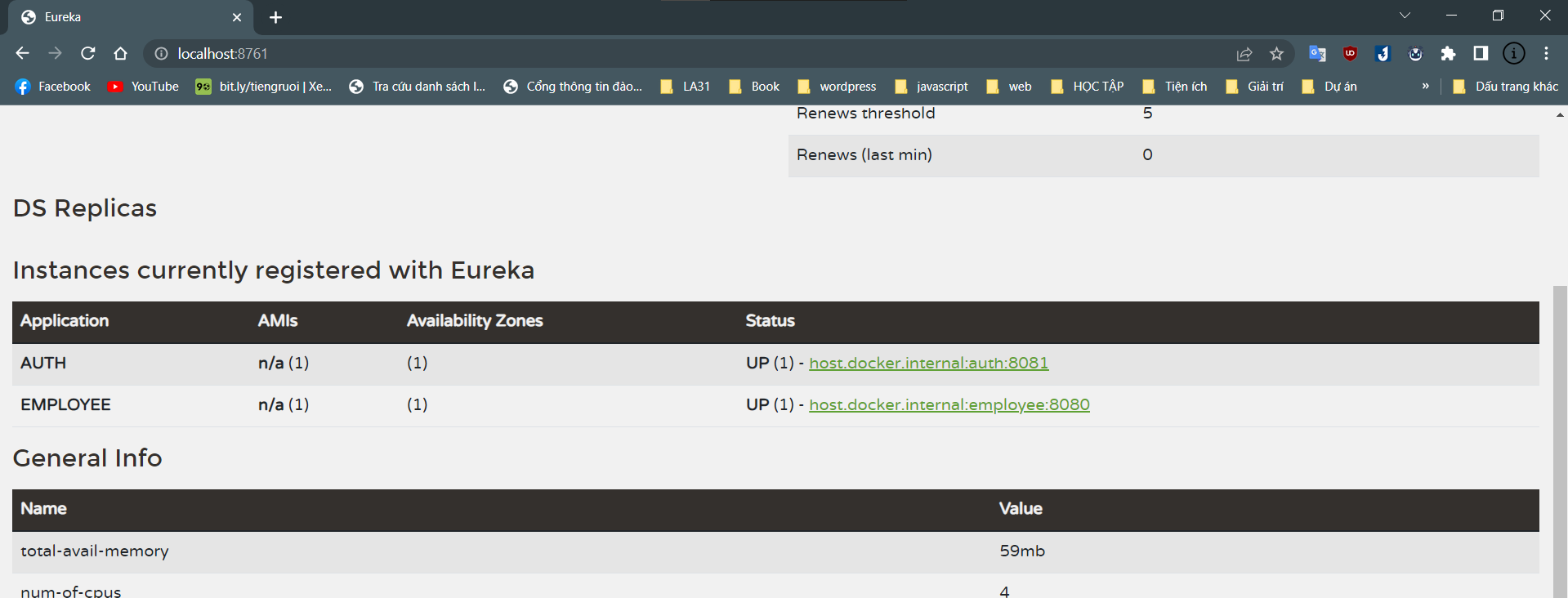
Employee service được chạy ở cổng 8080

Api gateway service được chạy ở cổng 8888

Để có thể kiểm tra được api gateway có điều hướng thành công hay không ta kiểm tra qua api của api gateway

[http://localhost:8888/[serviceId](http://localhost:8888/%5bserviceId)]

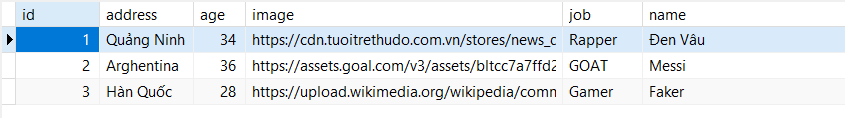
serviceId của employee service là tên của service đó đăng kí lên eureka server



Trong trường hợp này serviceId của employee service là “employee”

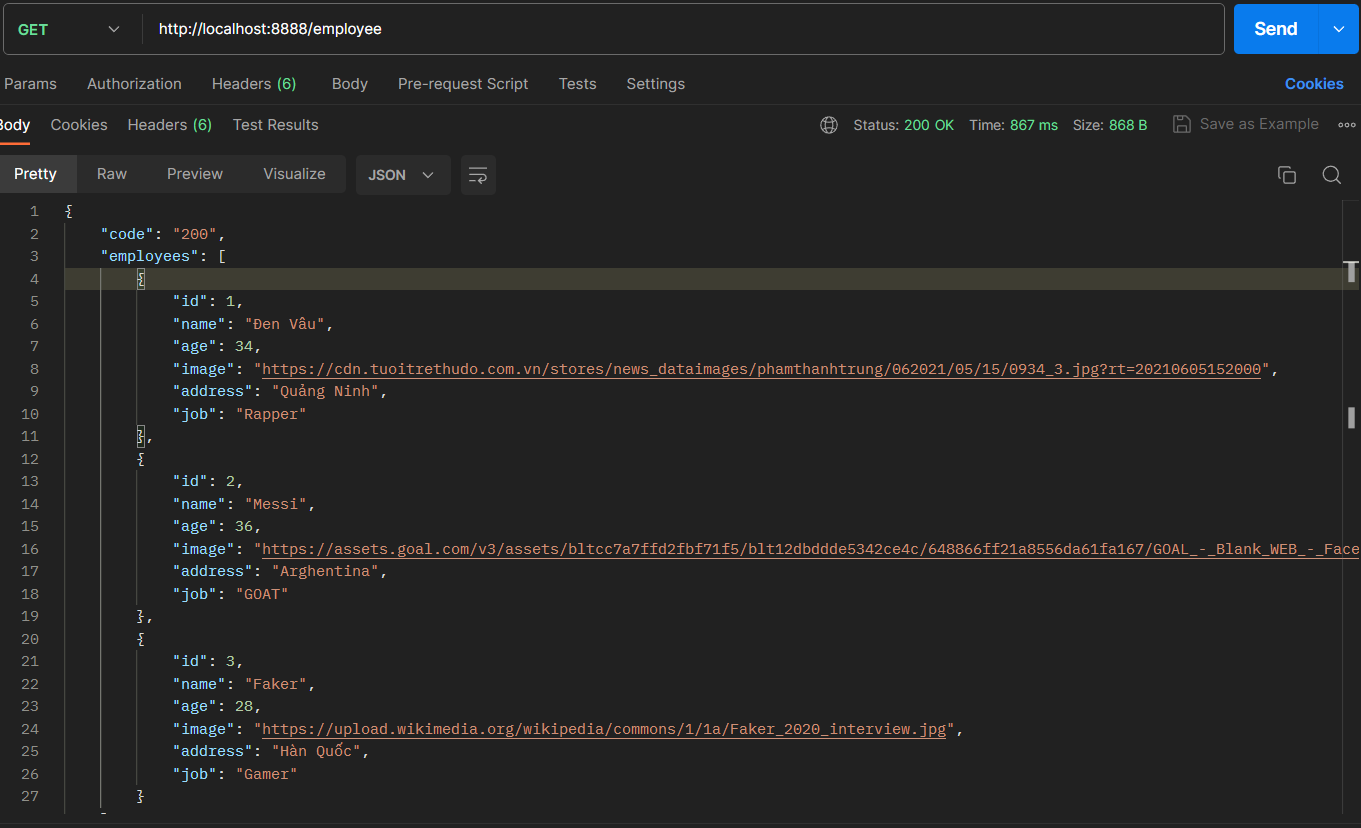
Vậy ta sẽ kiểm tra tại <http://localhost:8888/employee>

Dữ liệu đã có trên database



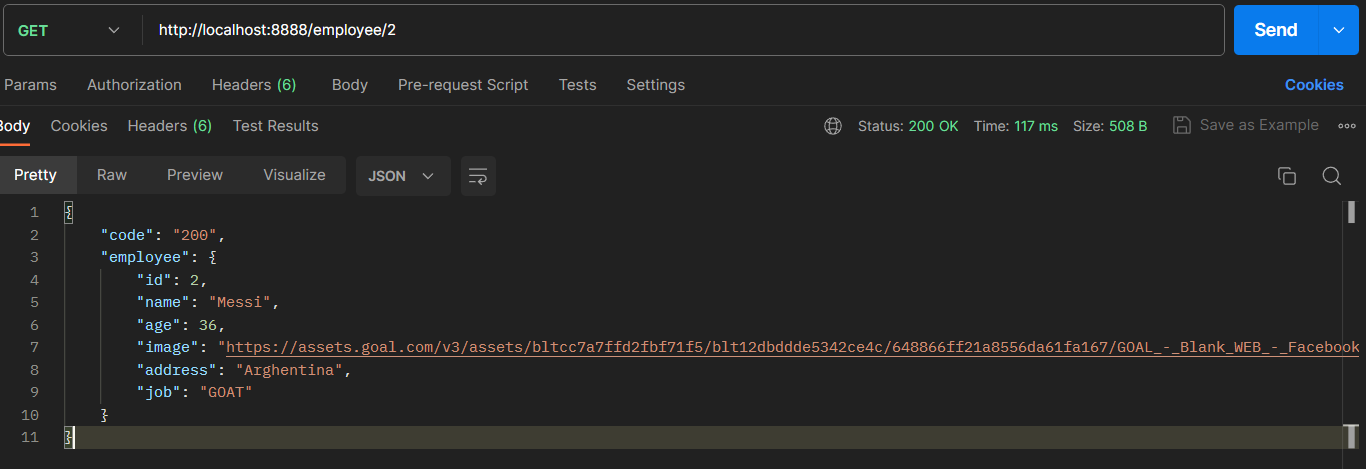
Kiểm tra qua postman

Lấy danh sách employee



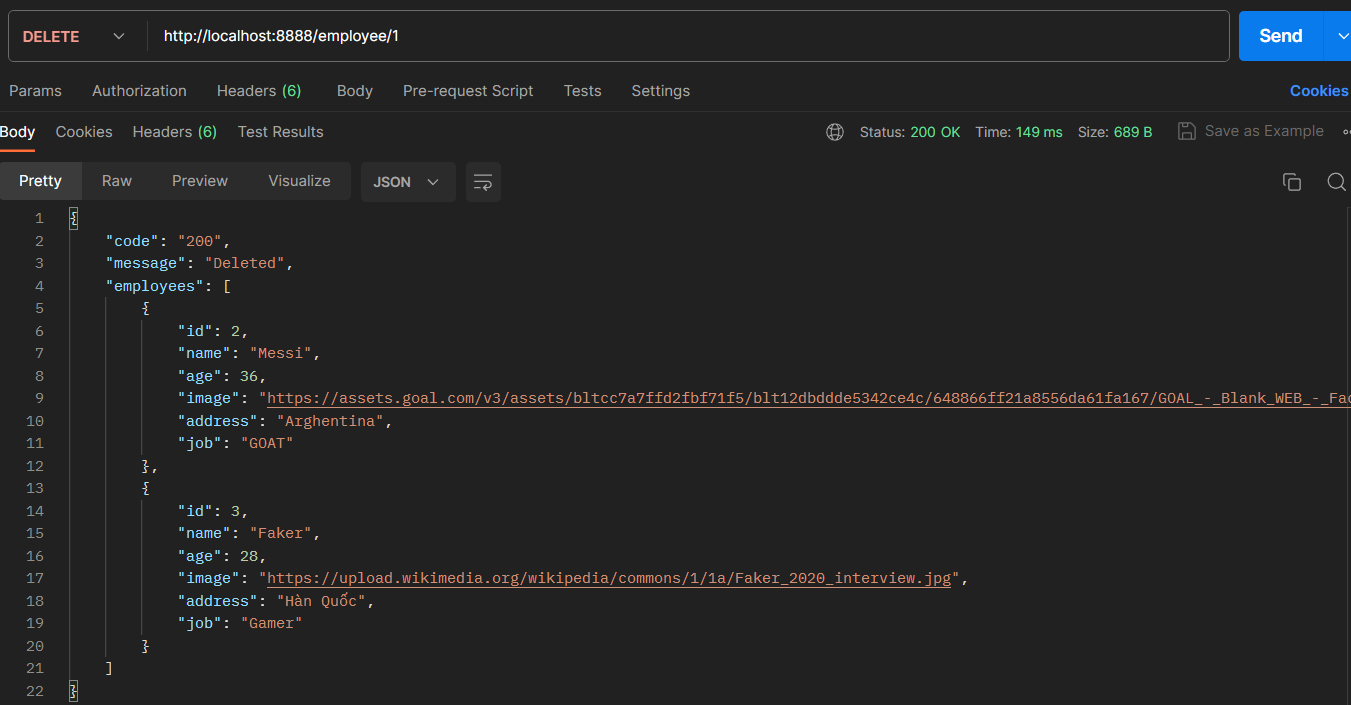
Lấy employee theo id

<http://localhost:8888/employee/2>



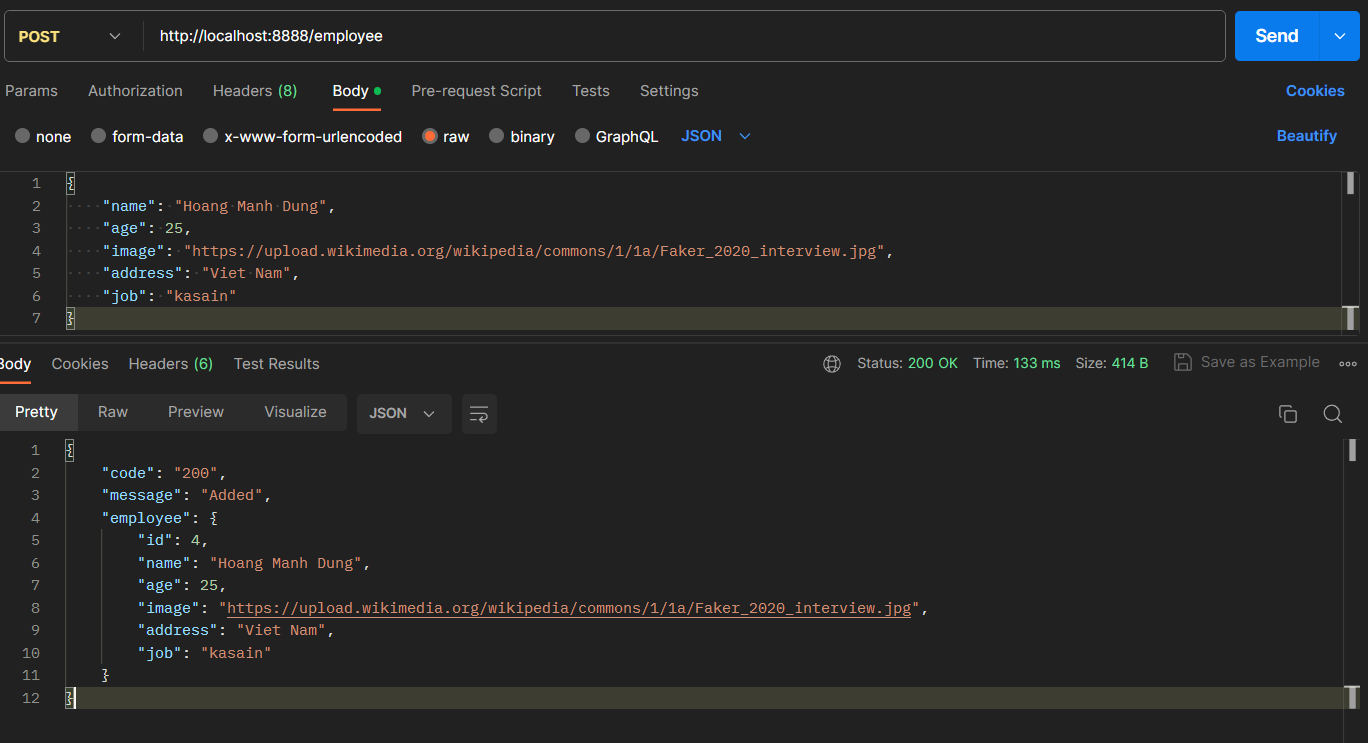
Xoá theo id

<http://localhost:8888/employee/1>

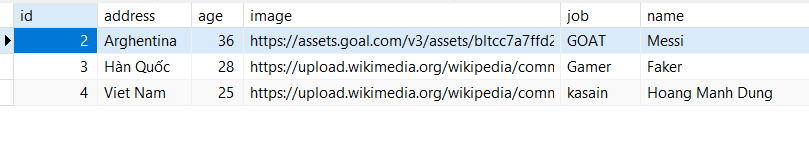


Thêm mới

<http://localhost:8888/employee>



Kiểm tra database



## 7.2 Auth service

Chú ý:

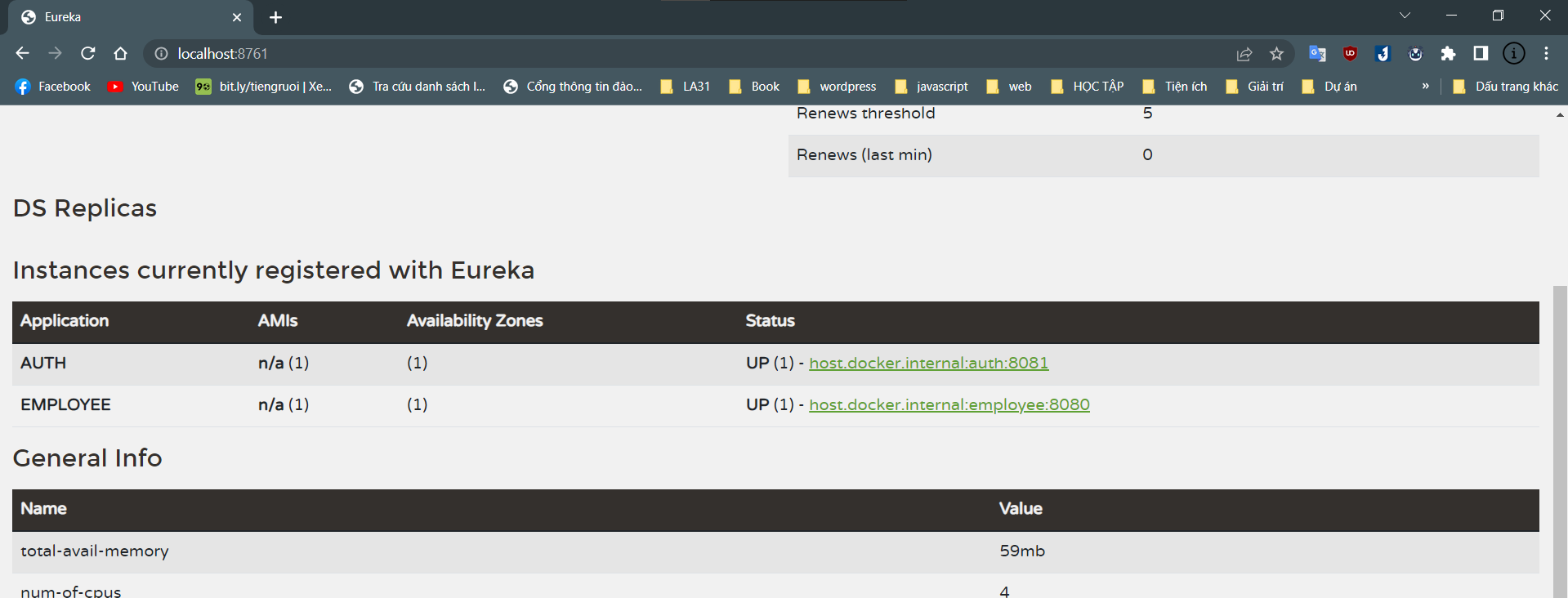
Auth service được chạy ở cổng 8081

Api gateway service được chạy ở cổng 8888

Để có thể kiểm tra được api gateway có điều hướng thành công hay không ta kiểm tra qua api của api gateway

[http://localhost:8888/[serviceId](http://localhost:8888/%5bserviceId)]

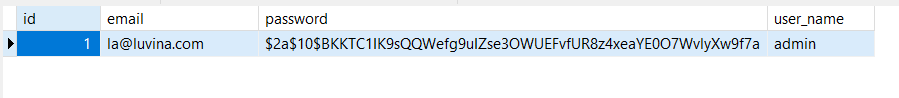
serviceId của auth service là tên của service đó đăng kí lên eureka server



Trong trường hợp này serviceId của auth service là “auth”

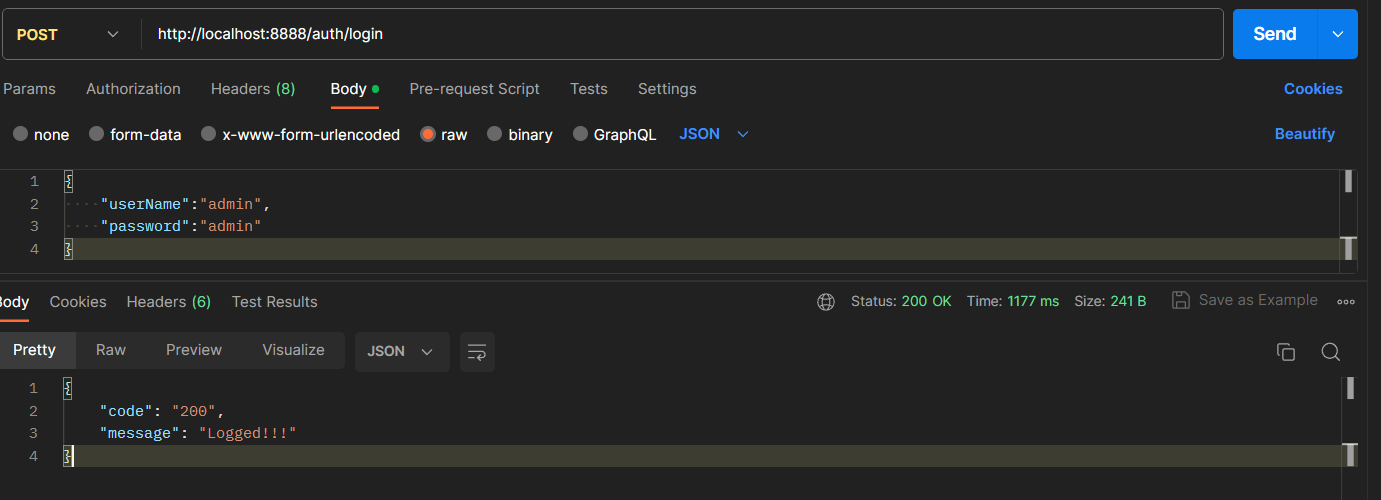
Vậy ta sẽ kiểm tra tại <http://localhost:8888/auth>

Trong database đã được tạo sẵn 1 account có username: admin và password: admin



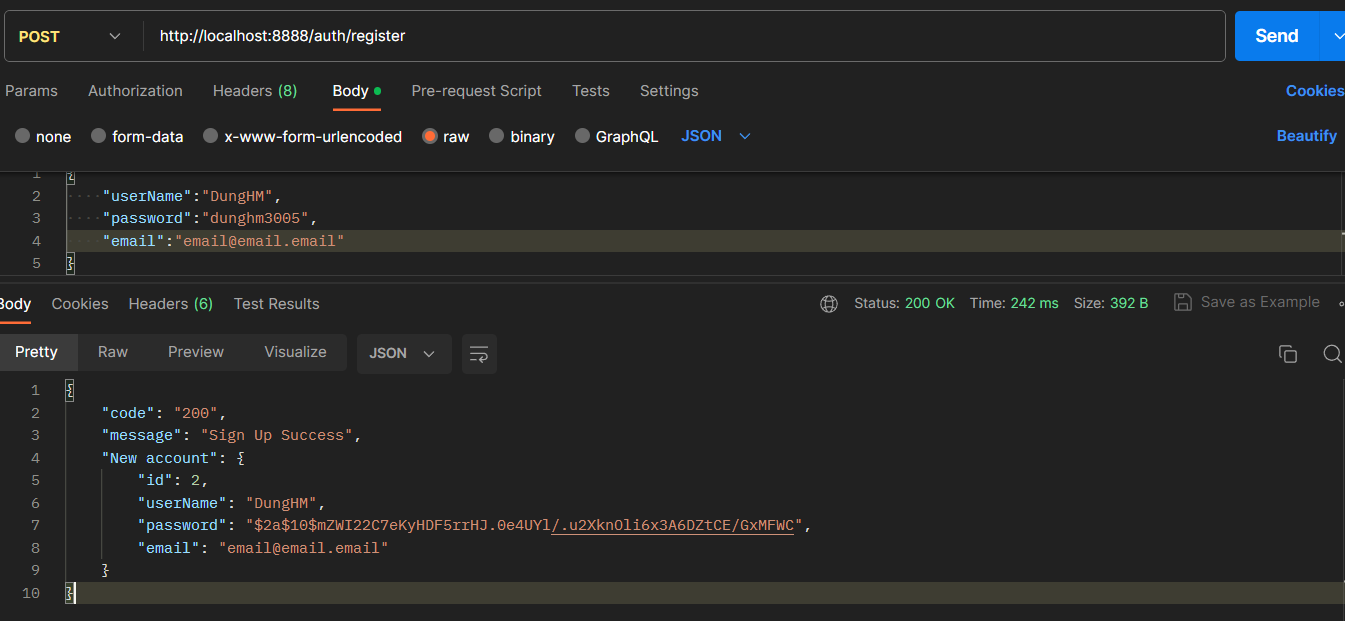
Kiểm tra login tại <http://localhost:8888/auth/login>

Login thành công thông qua api gateway



Kiểm tra đăng kí tại <http://localhost:8888/auth/register>

Đăng kí thành công thông qua api gateway



Kiểm tra database

