**Tổng quan về Kiến trúc Microservices**

Microservices là một phương pháp kiến trúc phần mềm chia ứng dụng thành các dịch vụ nhỏ, độc lập, có thể triển khai riêng biệt. Mỗi service tập trung vào một business capability cụ thể và giao tiếp với nhau thông qua API.

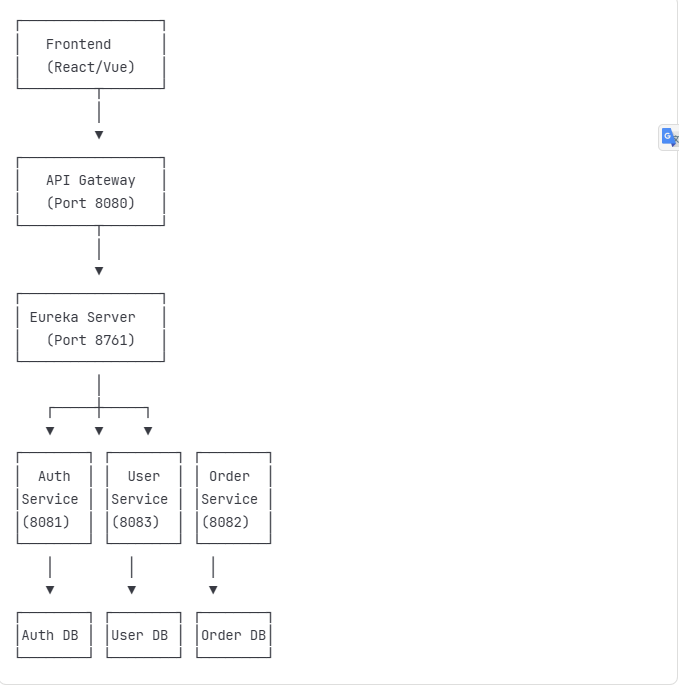
**Ưu điểm**

* **Khả năng mở rộng độc lập**: Mỗi service có thể scale riêng biệt
* **Công nghệ đa dạng**: Các team có thể chọn công nghệ phù hợp cho từng service
* **Triển khai độc lập**: Deploy service mà không ảnh hưởng đến toàn bộ hệ thống
* **Fault isolation**: Lỗi ở một service không làm sập toàn bộ hệ thống

**Nhược điểm**

* **Độ phức tạp cao**: Quản lý nhiều service và network communication
* **Data consistency**: Khó đảm bảo ACID transactions across services
* **Network latency**: Giao tiếp qua network chậm hơn in-process calls
* **Debugging khó khăn**: Trace lỗi qua nhiều services

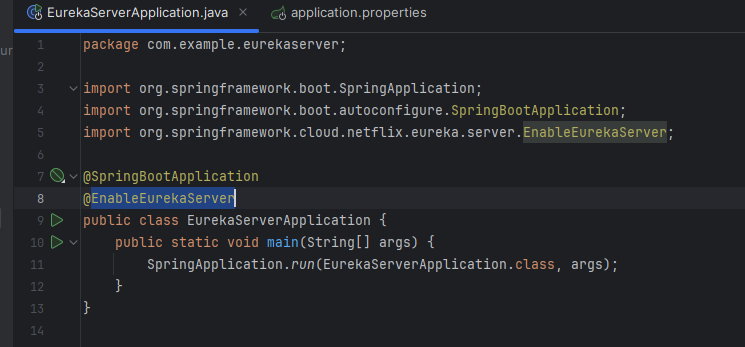
**Kiến trúc tổng thể**



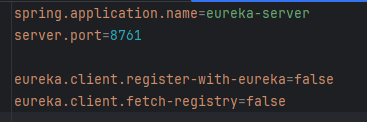
**Cách áp dụng Kiến trúc Microservices trong Website:**

**1. Service Registry - Eureka Server**

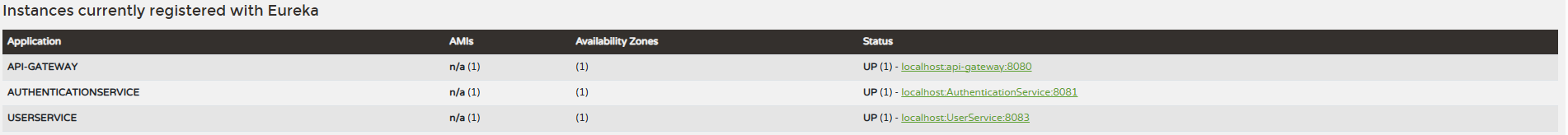
**Vai trò: Quản lý registry của tất cả microservices trong hệ thống.**

****

Khi tạo một ứng dụng Spring boot nếu có thêm Annotation @EnableEurekaServer, nó sẽ kích hoạt các các cấu hình cần thiết để biến một ứng dụng spring boot thành một Eureka Server

****

Đây là port cấu hình sẵn cho Eureka để các services tìm đăng ký

****

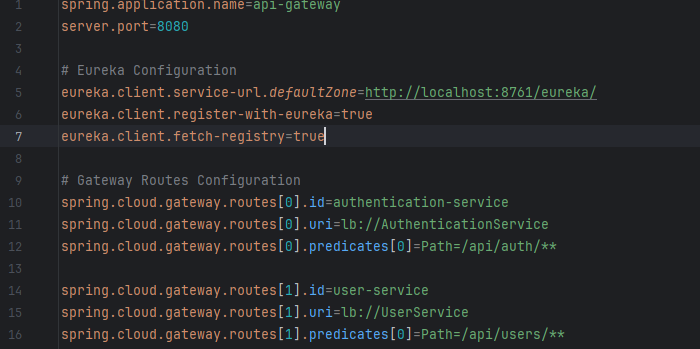
Khi truy cập vào hosting của Eureka chúng ta có thể thấy các service được đăng ký ở mục

Instances currently registered with Eureka

**2. API Gateway**

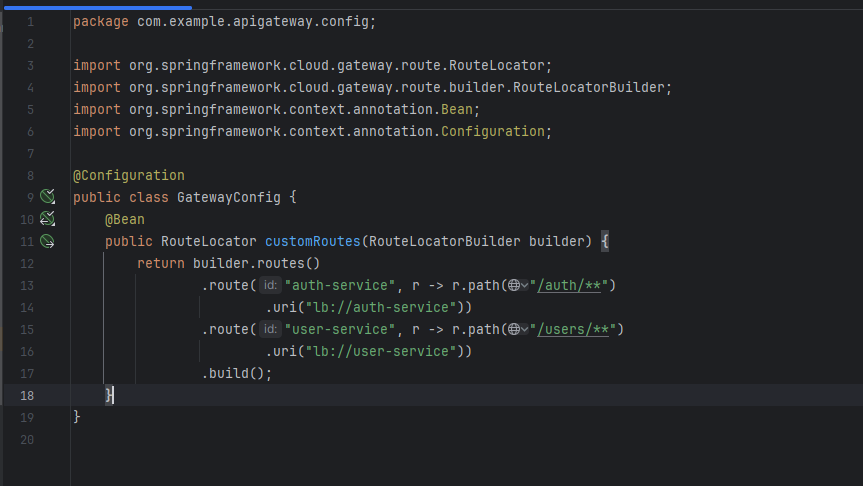
**Vai trò**: Single entry point cho tất cả client requests, routing, load balancing, authentication.

Đầu tiên là cấu hình properties cho API Gateway



Trong đó:

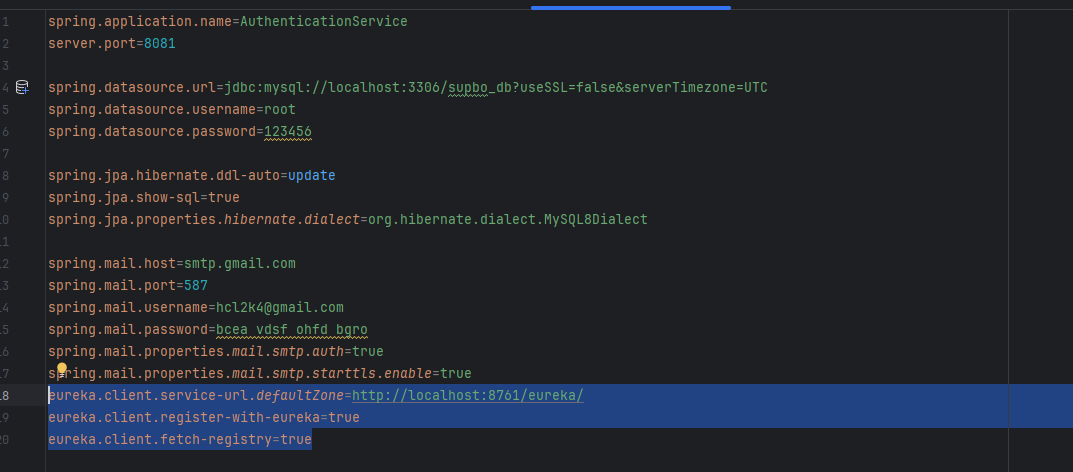
* + Id là trường để phân biệt các service với nhau để API gateway có thể điều hướng khi nhận request
  + Uri là tên Service được đăng ký trên Eureka
  + Predicates là đường dẫn để API gateway điều hướng request tới các service

Hoặc có thể thêm filter hay xử lý phức tạp trong file config (Thường là không cần thiết)

**3. Individual Microservices**

Để các services có thể tìm tới Eureka để đăng ký nhằm lắng nghe request từ API gateway hay lắng nghe từ các services khác thì phải cấu hình properties để trở thành eureka client

eureka.client.service-url.defaultZone=http://localhost:8761/eureka/  
eureka.client.register-with-eureka=true  
eureka.client.fetch-registry=true

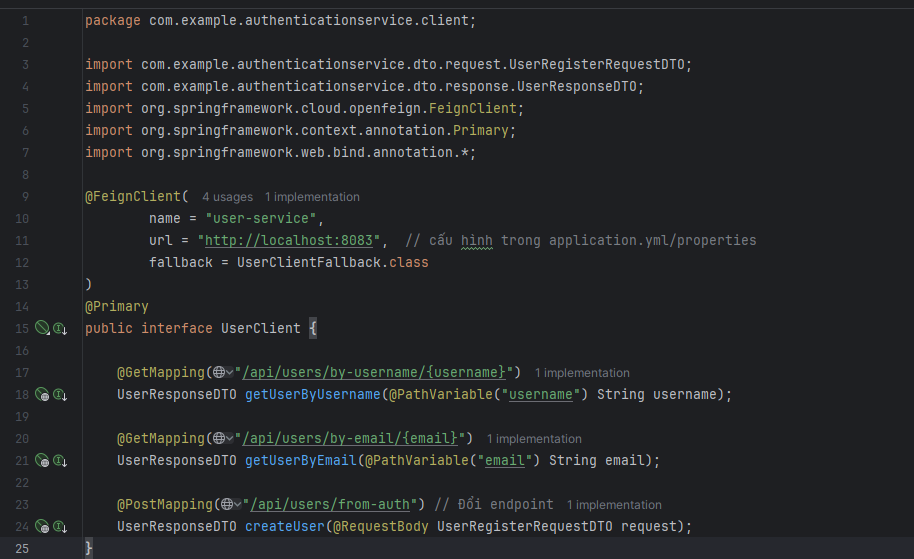
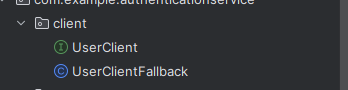
đây là 3 đoạn code cần thêm vào properties

\*Sau khi đã cấu hình như sau thì mỗi khi Client request tới các service bất kì thì chỉ cần gửi request qua port 8080 (port của API gateway) và API gateway sẽ tự điều hướng các request tới các service

**4. Feign Client - Service Communication**

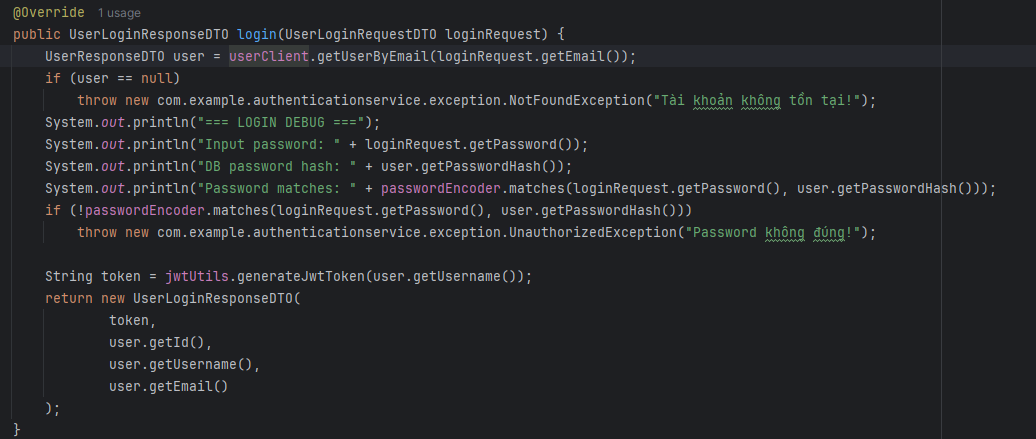
**Vai trò**: Declarative HTTP client để giao tiếp giữa các services.

Khi muốn giao tiếp tới các service khác chúng ta cần thiết lập service client muốn giao tiếp cho service cần giao tiếp (ví dụ Auth service muốn hỏi thông tin về người dùng thì phải giao tiếp với User service) như sau:



Ở đây chúng ta khai báo tên service và port của service đó cho service cần giao tiếp

Sau đó tạo mapping request Auth service gọi tới UserService



Ví dụ ở đâu chúng ta tạo đối tượng userClient để gửi request tới UserService sau đó trả lại response cho Auth Service