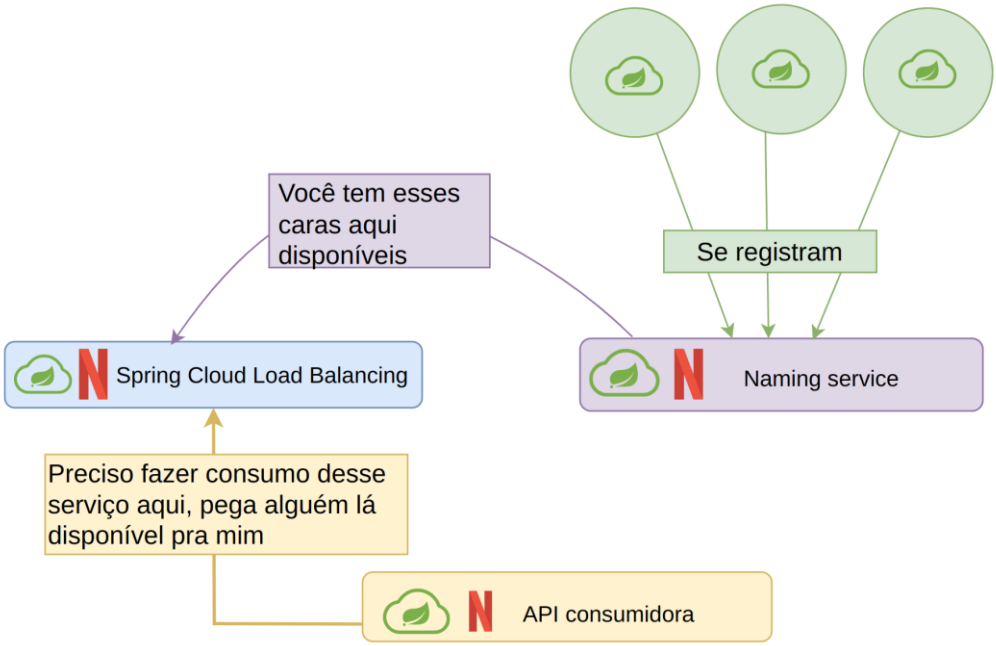
Balanceamento de carga - Teoria

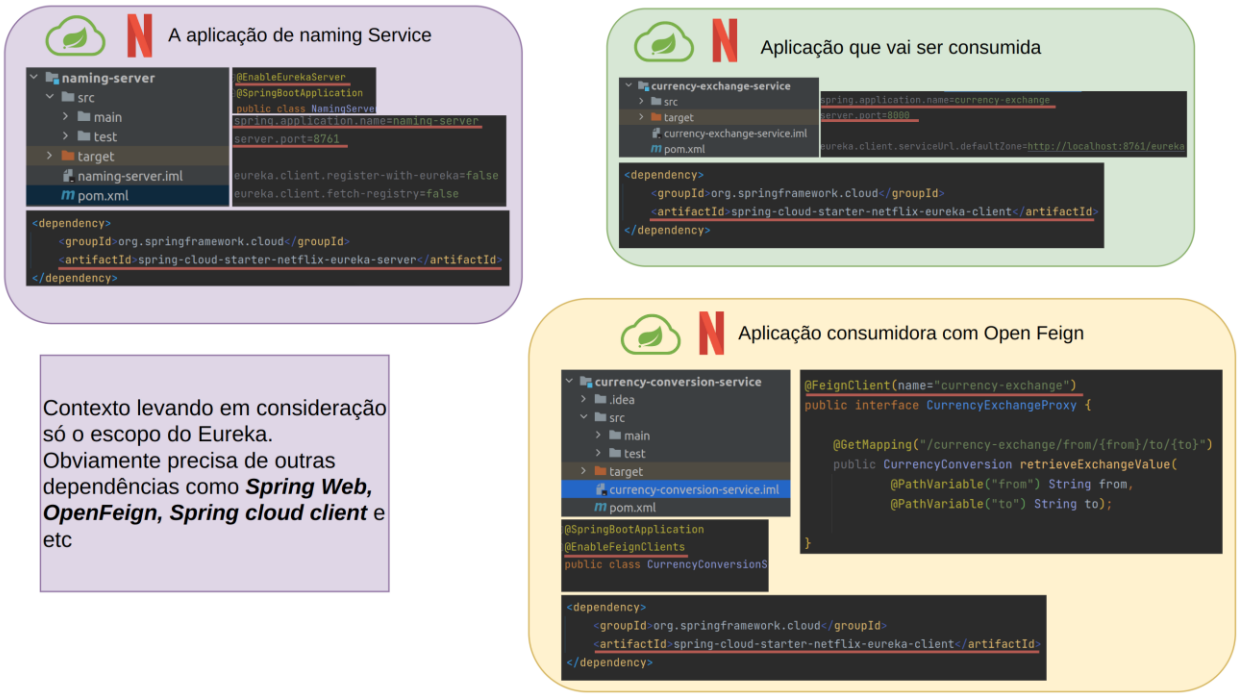
Com o **Naming Server** configurado e fazendo seu trabalho de **expor serviços para a aplicação consumidora**, ganhamos a possibilidade de fazer **Load Balancing**.

Então **Naming Server** provê as informações a **respeito das instâncias (portas e etc)** e a partir daí os requests são distribuídos pelo **próprio cliente:**



Spring Cloud load Balancer - Spring Cloud Netflix Eureka

Esse **ecossistema** funciona com configurações relativamente simples. Eu só preciso que meus micros serviços que se registram sejam **Clientes** do **Naming Server,** e minha **API consumidora só precisa do Open Feign** se referenciando pelo nome do serviço que for consumir:

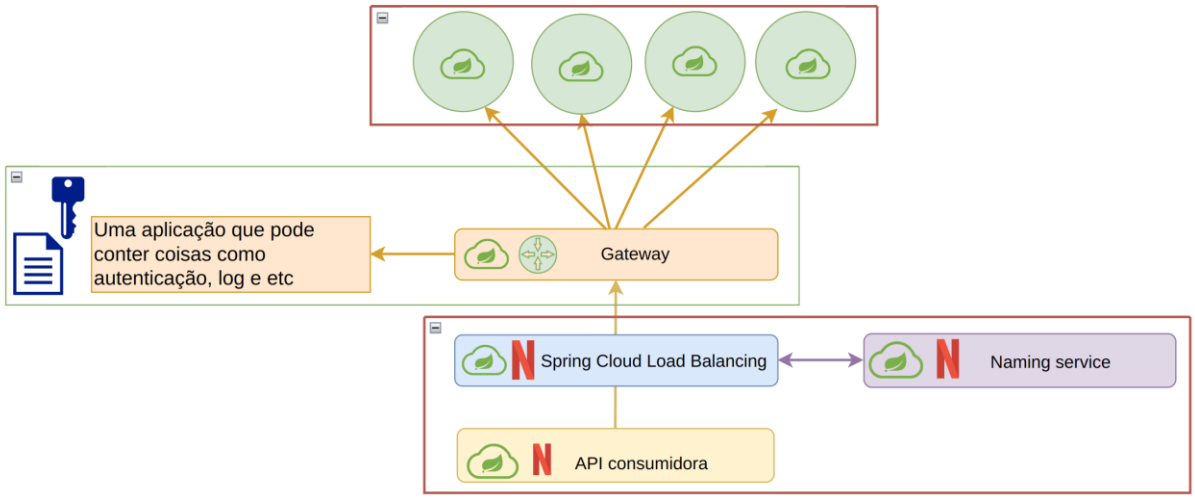


O **Load balancing** acontece pelo **client** que já possuí integrado a dependência do **Spring Cloud Load Balancer** dentro da dependência **Spring Cloud Netflix Eureka Client**.

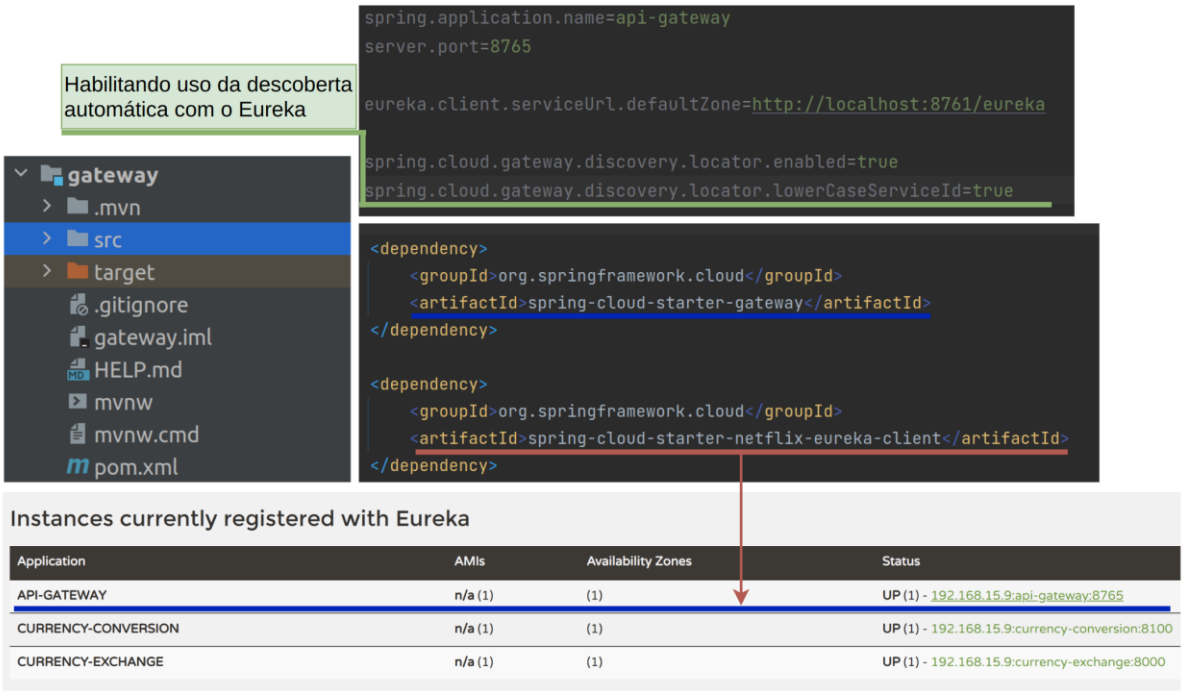
API Gateway - Spring Cloud API Gateway

Diferentes micros serviços possuem diversas features em comum como autorização e autenticação, logs e etc...

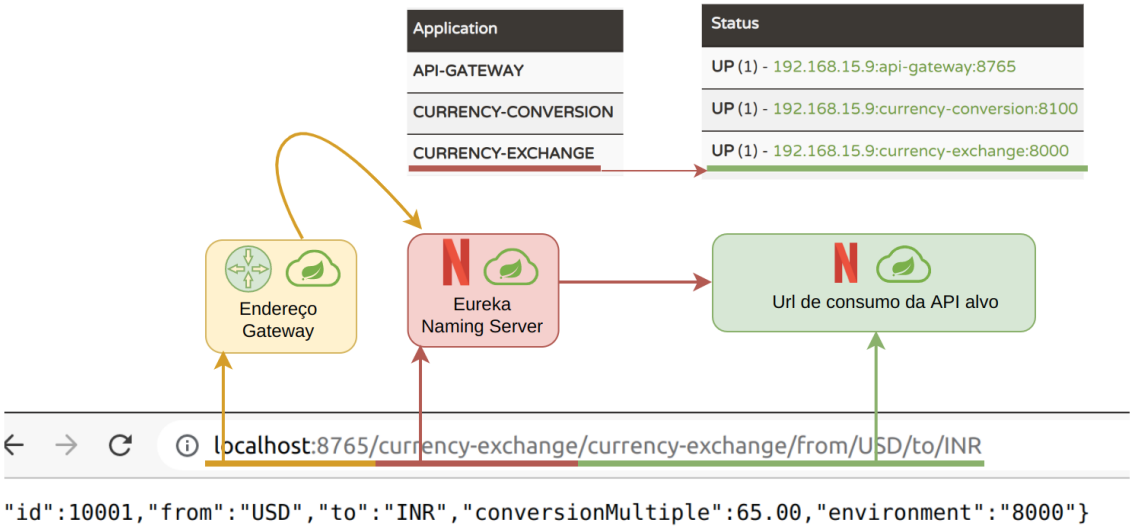
API GATEWAY é uma API que serve justamente para a i**mplementação de serviços comuns compartilhados entre vários micros serviços.** Então a partir de agora não existe mais a ideia de chamar outros micros serviços diretamente, antes você passa pelo **GateWay**:



O gateway é mais uma API que você precisa incluir com as outras, a partir de agora as chamadas serão feitas para o gateway que vai encontrar os serviços:



Uma questão que acontece com essa abordagem é termos uma **URL meio estranha**, agora eu preciso passar pelo **Gateway**, aí o **Gateway** tem que ir no **Eureka** atrás de um **serviço específico**, aí esse **serviço específico tem a rota que eu quero consumir**, o que me gera essa URL:



Rotas customizadas - Spring Cloud API Gateway

Um dos benefícios do API Gateway é centralizar algumas configurações, então eu posso interceptar URL's e customiza-las antes de redirecionar para algum micro service. Eu posso usar isso para **passar parâmetros e Headers** específicos (token e etc) ou para **melhorar o direcionamento corrigindo a duplicidade de nome na url**:

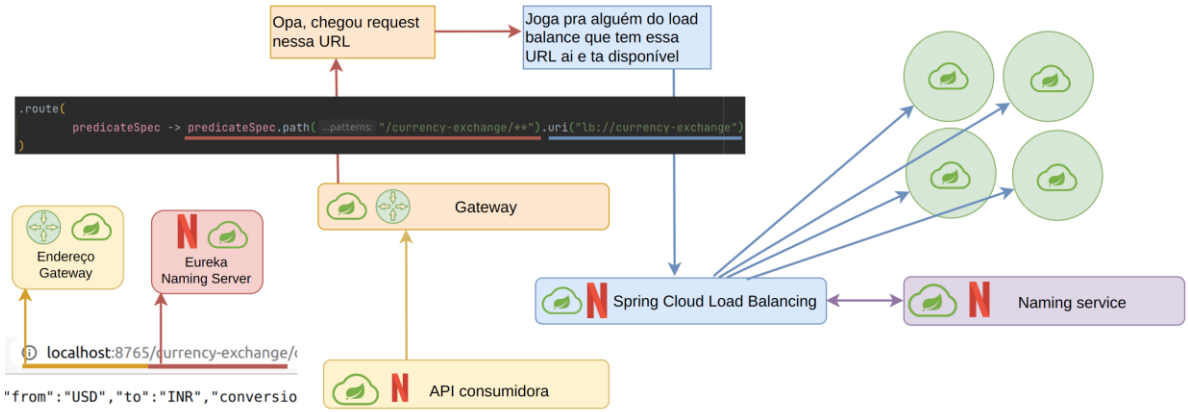
```java
@Configuration
public class ApiGatewayConfiguration {

    @Bean
    public RouteLocator gatewayRouter(RouteLocatorBuilder locatorBuilder){

        return locatorBuilder
                .routes()
                .route(
                        predicateSpec -> predicateSpec.path( ...patterns: "/get")
                                .filters(gatewayFilter -> gatewayFilter
                                        .addRequestHeader( headerName: "Hello", headerValue: "World")
                                        .addRequestParameter( param: "Olá", value: "Mundo")
                                ).uri("http://httpbin.org:80")
                )
                .route(
                        predicateSpec -> predicateSpec.path( ...patterns: "/currency-exchange/**")
                                .uri("lb://currency-exchange")
                )
                .route(
                        predicateSpec -> predicateSpec.path( ...patterns: "/currency-conversion/**")
                                .uri("lb://currency-exchange")
                )
                .build();
    }
}
```

Classe de configuração para **a injetar o Bean de RouteLocator**. A ideia é customizar nossas URL's de maneira específica invés de utilizar o padrão do application, inclusive, eu poderia **até comentar aquelas configurações já que estou injetando o Objeto eu mesmo**.

Eu posso usar o .path("") para pegar uma requisição que foi nesse caminho posso filtrar ela adicionando coisas e redirecionar no final.
Basicamente eu to **pegando um request que veio no gateway no caminho "/get" e redirecionando pra outro lugar**

Aqui eu to pegando as **requisições que vem até o Gateway em um serviço específico registrado no Naming Server barra qualquer coisa("/**") e direcionando-as para o Load Balance** que vai pegar o serviço disponível e acessar direto

Eu acredito que esse **último trecho de código precisa ser mais clarificado.** O que rola quando a **requisição** chega no **gateway** atrás de **um serviço registrado,** é que o gateway pega essa requisição e passa pro **Load Balancer** pra que ele encontre **algum serviço** disponível para responder a requisição:



O que resulta nessa URL: