**Visão Desenvolvimento**

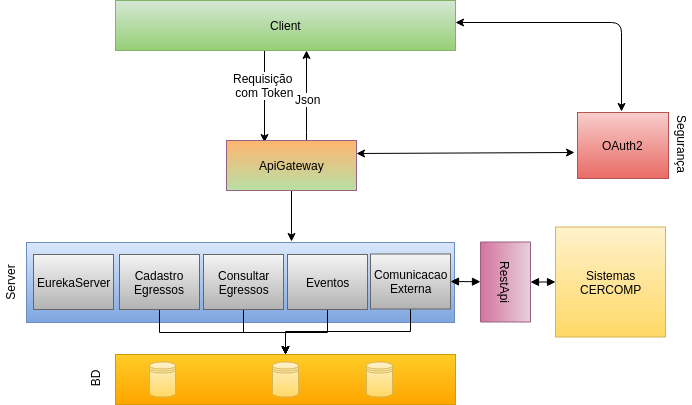
**1. Introdução**

O diagrama abaixo é organizado em camadas e microsserviços. Cada camada tem uma responsabilidade bem definida sendo que na camada de server temos subcamadas de serviços, onde estão os micro serviços e banco de dados.

Cada micro serviço provê uma função bem definida do SempreUFG, de forma que cada um desses serviços trabalham de forma independente se comunicando através Api’s REST.

O objetivo do diagrama então é mostrar como as camadas e micro serviços irão se comunicar entre si.

**2. Modelo**



**3. Elementos**

***3.1 Cliente***

A camada Cliente é responsável pela interface com o usuário. Ela que faz o intermédio entre o sistema e o cliente.

A responsabilidade dessa camada é unicamente fazer a apresentação do conteúdo ao usuário que rodará a aplicação através de um browser.

***3.2 API GateWay***

A ApiGateWay fará o papel de orquestrador das apis. Cada microserviço terá sua própria api REST pela qual o client irá se conectar, contudo deixar com que o client gerencie várias apis rests é contra produtivo, por isso a api gateway é importante para concentrar em un único ponto as apis, fazendo com que o client se preocupe apenas com se conectar à api gateway, deixando para ela fazer o gerenciamento das requisições.

***3.2 API REST***

API é uma interface de programação de aplicações, ou seja, é um conjunto de rotinas e padrões documentados por uma aplicação para que outras aplicações consigam utilizar funcionalidades desta aplicação sem precisar conhecer os detalhes da implementação do software.

Dessa forma o cliente pode ter acesso ao server através da API de forma independente, ou seja, sem conhecer em detalhes sua implementação. Essa camada então faz uma ponte entre os componentes do sistema.

O REST consiste em princípios que ajudam na criação de interfaces bem definidas para que as aplicações se comuniquem. Dessa forma o REST é apenas o conjunto de princípios e regras que utilizamos para construir nossas API’s.

Para que a aplicação faça uso do REST de forma plena ele precisa usar o protocolo HTTP para se comunicar, não possuir estado entre as comunicações e deve ser clara a definição do que faz parte do cliente e o que faz parte do servidor.

***3.3 Segurança***

A camada de segurança faz uso do protocolo OAuth2 que faz a parte de segurança entre client e server. Ela dá acesso ao REST API para que ele faça alguma requisição ao servidor em nome de um usuário.

O OAuth2 determina que quando o client precisa de uma informação protegida no server ele precisa de um token de acesso. Esse token de acesso pode ser personalizado de acordo com o usuário, assim poderemos definir diferentes níveis de acesso.

Com isso conseguimos atender ao requisito de segurança de nossa aplicação, além de poder definir perfis de acesso personalizado, que é uma das exigências do requisito do SempreUFG. É mister então lembrar que para a API ter acesso aos dados protegidos é preciso antes ter posse do token de acesso.

***3.4 Server***

Nessa camada é onde estarão os serviços, os quais se comunicarão entre si através de Apis Restfull. Haverá também um serviço específico que fará a comunicação com os sistemas externos do CERCOMP.

Cada micro serviço é independente e funciona separadamente, assim se um falhar o outro continuará funcionando.

Os microserviços contidos nesta camada também são responsáveis por fazer a intermediação entre a camada de apresentação e o banco de dados e é nessa camada que as regras de serviço são implementadas.

Atenção especial ao microserviço Eureka, que é na verdade um *Service Registry*, ou seja ela faz o gerenciamento de todos os outros microserviços, é usado para localizar serviços com o objetivo de balanceamento de carga e failover de servidores.

***3.5 BD***

Essa camada é responsável pela persistência de dados, pode haver mais de um banco de dados, pois eles serão usados por micro serviços independentes.

**4. Embasamento**

Essa visão foi escolhida para atender ao ponto de vista do projetista e é importante pois ela dá uma visão geral do sistema, dos módulos, camadas e como eles se relacionam entre si.

Com essa visão fica mais claro como o software será construído, qual responsabilidade de cada parte dele e assim facilitando a implementação de cada uma dessas partes já respeitando essas responsabilidades bem definidas.