Aluno: Tomás de Farias Ribeiro Caldas

Web Service, módulo 02, Aula 03

**1 - Explique quais são as seis regras de uma API REST.**

**Realize essa atividade no WORD ou no Bloco de Notas, suba esse arquivo para algum repositório e compartilhe o link no campo ao lado para que outros desenvolvedores possam analisá-lo.**

**O que é REST?**

REST (Representational State Transfer) pode ser definido como um conjunto de diretrizes que o software usa para se comunicar pela internet para integrações simples e escaláveis. Uma API que segue essas diretrizes é conhecida como API REST (API RESTful).

Quando um cliente solicita um recurso usando uma API REST, o servidor responde transferindo de volta o estado atual desse recurso específico em uma representação padronizada. Isso significa que as APIs REST funcionam enviando solicitações para um recurso e retornando todas as informações relevantes sobre esse recurso. Essas informações são traduzidas em um formato que os clientes podem interpretar sem problemas.

REST é um estilo arquitetônico ou melhor, um conjunto de diretrizes. Por outro lado, a especificação OpenAPI é uma maneira padrão pela qual você pode usar texto para descrever os recursos de uma API REST. Por exemplo, se o código postal de um cliente puder ser recuperado usando uma API REST, a especificação OpenAPI descreverá essa API para mostrar aos desenvolvedores como chamar a API de forma que ela responda adequadamente com o código postal solicitado.

Além disso, se você quiser implementar a especificação OpenAPI com sua API, você pode usar [APITransform](https://apitransform.com/) para carregar sua coleção Postman para convertê-la em OpenAPI.

**As seis regras das APIs REST**

**1. Uniform Interface**

Esta é uma diretriz que exige que todas as solicitações e respostas sigam um protocolo comum e fornece uma maneira pela qual as mensagens são formatadas. Como os servidores e aplicativos vêm em linguagens diferentes que dificultam o trabalho em conjunto, essa interface uniforme oferece uma linguagem comum para qualquer cliente interagir com qualquer API REST. Se isso não estivesse lá, seria uma bagunça total traduzir solicitações e respostas entre aplicativos.

**2. Client Server Separation**

O servidor e o cliente só podem interagir de uma maneira na [arquitetura REST](https://medium.com/@ahmetozlu93/mastering-rest-architecture-rest-architecture-details-e47ec659f6bc) . O cliente é obrigado a enviar uma solicitação ao servidor, enquanto o servidor só pode enviar uma resposta de volta ao cliente. Todas as interações entre os dois começam a partir do cliente.

**3. Layered System**

Embora tenhamos falado sobre solicitações de API como apenas uma interação entre o servidor e o cliente, normalmente há muito mais servidores entre as duas entidades. Esses servidores podem ser chamados de camadas e fornecem segurança, lidam com tráfego e ajudam na realização de diferentes tarefas. No entanto, as mensagens entre o cliente e o servidor devem ser formatadas da mesma forma, independentemente das camadas que existem entre elas.

**4. Stateless**

As chamadas feitas na estrutura da API REST devem ser sem estado. Além de todas as interações serem independentes, cada solicitação e resposta deve fornecer todas as informações necessárias para concluir uma interação. O servidor não se lembrará de nada sobre solicitações anteriores, o que significa que ele interpreta cada solicitação como uma nova tarefa.

**5. Cacheable**

Ao construir APIs REST, os desenvolvedores devem manter o cache em mente. Por exemplo, quando um cliente faz uma solicitação e o servidor responde, a resposta precisa indicar se o recurso fornecido pode ser armazenado em cache e, se puder, por quanto tempo.

**6. Code on Demand**

Embora essa regra não seja obrigatória, ela permite que uma API envie código de computador para clientes ao responder a uma solicitação para permitir que o cliente execute o código em seu back-end.