As seis regras de conceção de uma API REST

Para ser uma API REST, a API deve respeitar seis regras conhecidas como «restrições de arquitetura» ou «princípios de conceção».

1. Interface uniforme

Todos os pedidos efetuados através de uma API REST devem respeitar as regras de formatação dessa API. Independentemente do cliente que faz o pedido, deve colocar cada elemento de informação onde qualquer outro cliente o faria. Um exemplo é o URL (ou Uniform Resource Locator) utilizado para identificar recursos via HTTP, que é um exemplo de um URI (Uniform Resource Identifier) que pode ter alcances maiores.

2. Separação cliente-servidor

As API REST requerem que as aplicações cliente e servidor sejam totalmente independentes umas das outras. O cliente só precisa de saber o nome completo do recurso que deseja no espaço virtual permitido pela API. Caso contrário, o único conhecimento que o cliente e o servidor têm um do outro é o que é trocado por transações da API.

3. Ausência de estado

Cada pedido de cliente deve conter todas as informações necessárias ao seu tratamento, e o servidor não precisa de conservar quaisquer informações sobre esse pedido depois de o receber. Não há conceito de sessão no design API REST, e o servidor não tem estado em relação a qualquer cliente em particular.

4. Capacidade de cache

Em contraste com a ausência de estado por cliente do servidor, os recursos devem poder ser armazenados em um ou mais pontos dentro ou entre cliente e servidor. No caso do servidor, se um recurso específico tiver sido servido e for provável que seja novamente solicitado dentro de determinado período, deverá ser colocado em cache para uma resposta ulterior mais rápida. O cliente deve tomar uma decisão semelhante relativamente aos recursos recebidos. O servidor deve indicar através da API se um recurso pode ser colocado em cache de forma local e segura no cliente, incluindo a duração de vida dos dados, quando apropriado. O design da cache, embora não faça parte de uma especificação API RESTful, é essencial para o desempenho, e os designers da API devem estar cientes de como isto pode ser aplicado na prática.

5. Arquitetura de sistema por camadas

Uma consequência da separação cliente-servidor, da ausência de estado e da capacidade de cache é que um cliente não pode supor se está a comunicar diretamente com um servidor que possui um recurso específico, ou se está a ser servido por um intermediário como um mediador de serviços, um repartidor de carga, um sistema de distribuição de conteúdo ou outro subsistema mais próximo do cliente do que o servidor. Isto proporciona aos criadores de sistemas e infraestruturas uma flexibilidade considerável para maximizar a eficiência e a

fiabilidade da satisfação dos pedidos na infraestrutura global, com e sem fios. Está subjacente à funcionalidade de edge computing.

6. Código a pedido

Embora as API REST possam e frequentemente sirvam apenas dados para consumo pelo cliente, é cada vez mais comum que o código seja entregue para execução no cliente, como objetos Java ou aplicações web Javascript. Se for implementado, este código só pode ser executado a pedido do cliente.