As seis regras de uma API REST:

**Separação cliente-servidor**

Esse princípio REST trabalha com o conceito de que cliente e servidor devem ser isolados um do outro e ter permissão para desenvolver de forma independente. Dessa forma, você pode melhorar a capacidade de gerenciamento em várias plataformas e aumentar a escalabilidade simplificando os componentes do servidor, pois as preocupações com a interface do usuário são separadas das preocupações com o armazenamento de dados.

**Stateless (ausência de estado)**

De acordo com este princípio REST, as APIs são sem estado, o que significa que as chamadas podem ser feitas independentemente umas das outras. Além disso, cada chamada inclui os dados essenciais para se completar de forma eficaz.

Em outras palavras, cada solicitação enviada do cliente ao servidor deve incluir todas as informações necessárias para compreender a solicitação.

**Armazenamento em cache**

Como uma API sem estado pode aumentar [sobrecarga de solicitação](https://books.google.com.sa/books?id=Gg0sCgAAQBAJ&pg=PA9&lpg=PA9&dq=stateless+API+can+increase+request+overhead&source=bl&ots=IhG2ZW-Tt_&sig=ACfU3U1CLkXJeAEo9Fr8-OJUVeu8boOLog&hl=en&sa=X&ved=2ahUKEwjHrv6ssJLnAhUCC2MBHX4cDu8Q6AEwCnoECAYQAQ#v=onepage&q=stateless%20API%20can%20increase%20request%20overhead&f=false) gerenciando grandes cargas de chamadas de entrada e saída, um design de API REST deve armazenar dados em cache. De acordo com este princípio de design de API, os dados em uma resposta devem ser indiretamente ou categorizados como armazenáveis ​​ou não armazenáveis ​​em cache.

Se uma resposta puder ser armazenada em cache, o cache do cliente terá o direito de reciclar os dados de resposta para solicitações semelhantes no futuro.

**Interface Uniforme**

Para desacoplar um cliente do servidor, é necessário ter uma interface unificada que permita o desenvolvimento autônomo do aplicativo sem acoplar firmemente seus serviços, modelos e ações à própria camada de API.

Este princípio de design otimiza toda a arquitetura do sistema e aumenta a visibilidade das comunicações. Para obter uma interface uniforme, vários controles de arquitetura são necessários para orientar o desempenho dos elementos dentro da arquitetura da API REST.

Os princípios REST são definidos por quatro controles de interface, incluindo identificação de recursos, gerenciamento de recursos por meio de representações, comunicações autodescritivas e hipermídia como mecanismo do estado do aplicativo.

**Sistema em camadas**

A arquitetura da API REST inclui várias camadas que operam juntas para construir uma hierarquia que ajuda a gerar um aplicativo mais escalável e flexível. Devido ao seu sistema em camadas, um aplicativo tem melhor segurança, pois os componentes em cada camada não podem interagir fora da camada subsequente. Além disso, ele equilibra cargas e oferece caches compartilhados para estimular [escalabilidade](https://www.edureka.co/blog/what-is-rest-api/#Need%20of%20REST%20API).

Um sistema de arquitetura REST API em camadas tem maior estabilidade porque restringe o desempenho do componente. de modo que cada componente não pode "ver" além da camada imediata com a qual está se misturando.

**Código sob Demanda**

Esse princípio REST permite que codificação ou applets sejam comunicados por meio da API usada no aplicativo.

Uma definição da API REST permite estender a funcionalidade do cliente baixando e implementando a codificação na forma de applets ou scripts. Isso simplifica os clientes, diminuindo o número de recursos essenciais a serem pré-implementados.

Na maioria das vezes, um servidor retorna representação estática de recursos no formato XML ou JSON. Porém, quando necessário, os servidores podem fornecer código executável ao cliente.