**Interface uniforme**

É uma restrição chave que diferencia entre uma API REST e uma API não REST. Sugere que deve haver uma forma uniforme de interação com um determinado servidor, independentemente do dispositivo ou tipo de aplicativo (site, aplicativo móvel).

Existem quatro princípios de diretrizes de Interface Uniforme são:

Baseado em recursos : os recursos individuais são identificados nas requests. Por exemplo: API / usuários.

Manipulação de Recursos Através de Representações: O cliente possui representação de recurso e contém informações suficientes para modificar ou excluir o recurso no servidor, desde que tenha permissão para tal. Exemplo: Normalmente, o usuário obtém um ID de usuário quando solicita uma lista de usuários e, em seguida, usa esse ID para excluir ou modificar esse usuário específico.

Mensagens autodescritivas : cada mensagem inclui informações suficientes para descrever como processar a mensagem para que o servidor possa analisar facilmente a solicitação.

Hipermídia como Motor de Estado da Aplicação (HATEOAS): Necessita incluir links para cada resposta para que o cliente descubra facilmente outros recursos.

**Comunicação Stateless**

Requisições feitas por um cliente a um serviço REST devem conter todas as informações necessárias para que o servidor as interprete e as execute corretamente. Clientes não devem depender de dados previamente armazenados no servidor para processar uma requisição. Qualquer informação de estado deve ser mantida pelo cliente e não pelo servidor. Isso reduz a necessidade de grandes quantidades de recursos físicos, como memória e disco, e também melhora a escalabilidade de um serviço REST.

**Cliente Servidor**

Recurso

Um recurso pode ser qualquer objeto sobre o qual a API possa oferecer informações. Por exemplo, no caso de uma API do Twitter, um recurso pode ser um usuário, hashtag ou qualquer tipo de mídia como uma imagem. Todo recurso possui um identificador distinto que pode ser um nome ou número.

O recurso é a abstração primária de informações em REST. A API REST usa um identificador de recurso para reconhecer o recurso específico envolvido na comunicação entre diferentes elementos.

servidor

Um servidor é qualquer sistema que contém recursos que o cliente deseja. Ao receber solicitações do cliente, ele fornece o conteúdo ao cliente usando a interface da API. O servidor concederá apenas um estado representativo da fonte e não acesso completo ao cliente.

Um excelente exemplo disso é quando um aplicativo móvel mostra vídeos do YouTube por meio de sua interface. Ele usa uma API REST para chamar o conteúdo de vídeo do YouTube sem hospedá-lo em seu sistema.

**Armazenável em cache**

Cada resposta deve incluir se a resposta pode ser armazenada em cache ou não e por quanto tempo as respostas podem ser armazenadas em cache no lado do cliente. O cliente retornará os dados de seu cache para qualquer solicitação subsequente e não haverá necessidade de enviar a solicitação novamente ao servidor. Um cache bem gerenciado elimina parcial ou completamente algumas interações cliente-servidor, melhorando ainda mais a disponibilidade e o desempenho. Mas às vezes há chances de que o usuário receba dados desatualizados.

**Sistema em camadas**

Uma arquitetura de aplicativo precisa ser composta de várias camadas. Cada camada não sabe nada sobre qualquer camada além da camada imediata e pode haver muitos servidores intermediários entre o cliente e o servidor final. Os servidores intermediários podem melhorar a disponibilidade do sistema, permitindo o balanceamento de carga e fornecendo caches compartilhados.

**Código sob demanda**

É um recurso opcional. De acordo com isso, os servidores também podem fornecer código executável ao cliente. Os exemplos de código sob demanda podem incluir os componentes compilados, como miniaplicativos Java e scripts do lado do cliente, como JavaScript.