**Interface Uniforme**

Ao aplicar o princípio da generalidade à interface dos componentes, podemos simplificar a arquitetura geral do sistema e melhorar a visibilidade das interações.

Múltiplas restrições arquitetônicas ajudam a obter uma interface uniforme e orientam o comportamento dos componentes.

As quatro restrições a seguir podem alcançar uma interface REST uniforme:

* **Identificação dos recursos** – A interface deve identificar exclusivamente cada recurso envolvido na interação entre o cliente e o servidor.
* **Manipulação de recursos por meio de representações** - Os recursos devem ter representações uniformes na resposta do servidor. Os consumidores da API devem usar essas representações para modificar o estado dos recursos no servidor.
* **Mensagens autodescritivas** - Cada representação de recurso deve conter informações suficientes para descrever como processar a mensagem. Ele também deve fornecer informações sobre as ações adicionais que o cliente pode executar no recurso.
* **Hipermídia como o mecanismo do estado da aplicação** – O cliente deve ter apenas o URI inicial do aplicativo. O aplicativo cliente deve conduzir dinamicamente todos os outros recursos e interações com o uso de hiperlinks.

**Cliente-Servidor**

O padrão de design cliente-servidor impõe a separação de preocupações, o que ajuda o cliente e os componentes do servidor a evoluir de forma independente.

Ao separar as preocupações da interface do usuário (cliente) das preocupações de armazenamento de dados (servidor), melhoramos a portabilidade da interface do usuário em várias plataformas e melhoramos a escalabilidade simplificando os componentes do servidor.

Enquanto o cliente e o servidor evoluem, temos que garantir que a interface/contrato entre o cliente e o servidor não seja quebrada.

**Stateless**

Statelessness exige que cada solicitação do cliente para o servidor contenha todas as informações necessárias para entender e concluir a solicitação.

O servidor não pode aproveitar nenhuma informação de contexto armazenada anteriormente no servidor.

Por esse motivo, o aplicativo cliente deve manter totalmente o estado da sessão.

**Armazenamento em cache**

Exige que uma resposta se rotule implícita ou explicitamente como armazenável em cache ou não armazenável em cache.

Se a resposta puder ser armazenada em cache, o aplicativo cliente obterá o direito de reutilizar os dados de resposta posteriormente para solicitações equivalentes e um período especificado.

**Sistema em Camadas**

O estilo de sistema em camadas permite que uma arquitetura seja composta de camadas hierárquicas, restringindo o comportamento do componente.

Por exemplo, em um sistema em camadas, cada componente não pode ver além da camada imediata com a qual está interagindo.

**Código sob demanda**

O REST também permite que a funcionalidade do cliente se estenda baixando e executando código na forma de applets ou scripts.

O código baixado simplifica os clientes reduzindo o número de recursos necessários para serem pré-implementados.

Os servidores podem fornecer parte dos recursos entregues ao cliente na forma de código, e o cliente precisa apenas executar o código.