Representational State Transfer (REST) é um estilo arquitetural para desenvolvimento de Web Services baseado no protocolo HTTP. Isso mesmo, REST, ao contrário do que você ouve falar por aí, não é uma tecnologia específica, um padrão de desenvolvimento ou uma biblioteca. REST é um estilo arquitetural que utiliza determinados padrões, como: HTML, XML, JSON e outros! É um engano pensar que o fato de utilizar o protocolo HTTP com o padrão JSON torna a sua arquitetura RESTful. De acordo com a tese de doutorado de Roy Fielding, um dos principais autores das especificações HTTP, para que uma arquitetura seja considerada RESTful, um conjunto de regras e princípios devem ser seguidos:

Cliente-Servidor: A separação das responsabilidades é o princípio por trás do cliente-servidor. Ao separar as preocupações de interface de usuário (UI) do armazenamento de dados, é possível melhorar a portabilidade através de múltiplas plataformas de UI, simplificar os componentes do servidor, mas principalmente, permitir a evolução de forma independente uma vez que não há dependência entre os lados cliente/servidor.

Interface Uniforme: A característica principal que diferencia o estilo arquitetural REST dos demais é uma interface uniforme entre os componentes cliente e servidor. Como o cliente e servidor compartilham esta interface, deve-se ter um “contrato” bem definido para comunicação entre os lados. Há quatros princípios que devem ser seguidos para obter uma interface uniforme: Identificação dos Recursos, Representação dos recursos, Mensagens auto-descritivas e Hypermedia (HATEOAS).

Stateless: A comunicação entre cliente-servidor deve ocorrer independente de estado, não cabendo ao servidor armazenar qualquer tipo de contexto, ou seja, cada requisição deve possuir toda informação necessária para que seja inteiramente compreensível. Este princípio acaba gerando um alto tráfego de dados e redução de performance, porém pode ser contrabalanceado utilizando adequadamente o recurso de cache.

Cache: O cache ajuda a melhorar a performance, a escalabilidade e eficiência uma vez que reduz o tempo de resposta médio quando comparado entre uma série de interações cliente-servidor. As diretivas de cache são controladas pelo servidor através do cabeçalho HTTP (HTTP Header).

Camadas: Arquitetura deve ser construída através de camadas gerenciadas de forma independente, onde cada layer não pode ver além do layer adjacente e mudanças de um layer não devem impactar nos demais. É recomendável que o cliente nunca conecte-se diretamente no servidor de aplicação e que uma camada de balanceamento de carga seja adicionada entre cliente-servidor. A grande vantagem de trabalhar em camadas é que a arquitetura se torna menos complexa e fica mais propensa a mudanças.

Código sob demanda: Código sob demanda define a possibilidade de serviços baseados em REST responderem como representação de um recurso informação executável pelo cliente. Esta possibilidade de entrega de funcionalidades no entanto deve ser suportada pelo cliente criando um ambiente de execução propício para sua execução. Este não pode ser considerado uma restrição uma vez que Roy aponta ele como sendo opcional na estruturação de uma arquitetura baseada em REST. Ele contudo acredita que esta funcionalidade, dependendo do escopo da aplicação, poderia fornecer a Escalabilidade e a Performance de execução necessárias em detrimento da Visibilidade e Simplicidade da arquitetura.1’