HTTP

Segurança: o conceito de segurança é definido de maneira oposta ao conceito de efeito colateral, um efeito colateral é uma alteração no servidor causada pela mudança de estado de um recurso. Já a segurança nada mais é que uma requisição que tenha uma resposta que não cause alteração no estado de um recurso, não invocando um efeito colateral no servidor.

Métodos seguros: Desta forma podemos definir dois metodos seguros HEAD e GET, que por via de regra independente de quantas vezes são chamados retornaram suas repostas sem disparar um efeito colateral.

Métodos Idempotentes: São os métodos que sempre que executados retornaram o mesmo retorno. Ou seja chamando o mesmo 0 ou 1000 vezes devem ter o mesmo efeito. Por via de regra todos metidos seguros são idempotentes, logo temos : Head , Get, Put, Delete. Já os metodos Options e Trace não tem efeitos colaterais logo são inerentemente idempotentes. Uma sequencia de invocação destes métodos podem ou não ser idempotente, Se o resultado de uma sequencia executada inteira não é alterado por uma reexecução de uma parte desta sequencia. Um sequencia que não causa efeito colateral é uma sequencia idempotente, por definição .

METODOS:

1. POST
2. PUT
3. GET
4. DELETE
5. OPTIONS
6. HEAD
7. PATCH: pode atualizar ou não um recurso, não é idempotente, porém pode ser implematado de maneira que garanta isto. As Colisoes de varias solicitações patchs podem causar efeitos colaterais mais agressivos que as de put
8. CONNECT
9. TRACE

Status code:

1xx: carater informativo
2xx: carater de sucesso de requiscao,
3xx: carater de direcionamento de requiscao
4xx: carater de falha na requicao vinda do cliente
5xx: carater de falha no servidor.

RESTful:

modelo de maturidade de richardson: define atraves de especificações quais regras sua API deve seguir para ser considerada RESTFul, ele classifica em 4 niveis de maturidade, e que ao garantir todos os níveis sua aplicação garante alta interoperabilidade enter sistemas, e baixo acoplamento que traz diversos benefícios.

Nivel 0 – Usa HTTP como veiculo de transporte: Richard afirma que o primeiro passo para que sua API, seja RESTFul ela deve se comunicar atraves do protocolo HTTP. Os exemplos mais simples são as aplicações SOAP.
Nivel 1 – Utiliza os verbos HTTP e respostas: Segundo o autor é necessário a utilização dos verbos HTTP de forma assertiva com a operação que sera realizada, além de devolver no corpo da response o status code, message e se ouve erros quais são.
Nivel 2 – Utilização de Recursos: Ao contrario do monolito devemos fragmentar nosso software em diversos recursos, cada um com sua URI que oferece por via de interface os metodos para alteração do estado de algum recurso.
Nivel 3 – Hiper midia como controle do motor de aplicação (HATEOAS): Segundo o autor este é o ultimo nivel necessário para garantir que a aplicação seja RESTFul pois é atraves deste que criamos uma independência entre o cliente e servidor garatindo que um pode progredir sem o outro. Isto se deve pelo baixo acoplamento gerado. Quando tem-se uma requisição em um recurso devemos através da response retornar também quais são os links e operações disponíveis para o estado daquele recurso. Desta forma a partir de uma URI podemos efetuar as operações disponíveis para este recurso sem conhecer previamente a URI para tal metodo.