Softex – web servisse

Vandilma Candido

Pesquisa :

Regras do REST

* **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 sobre demanda:** Embora as API REST possam e frequentemente sirvam apenas dados para consumo pelo cliente, é cada vez mais comum que o código seja entregue para execução no cliente, como objetos Java ou aplicações web Javascript. Se for implementado, este código só pode ser executado a pedido do cliente.