REST - Representational State Transfer

- **REST** é um mecanismo mais simples para acesso a documentos e recursos remotos dentro de uma arquitetura de cliente/servidor.

- **RE**presentational **S**tate **T**ransfer foi criado em 2000 e desde então tem assumido papel importante como mecanismo de desenvolvimento e entrega de serviços de TI.

REST - Representational State Transfer

A maior implementação REST que se conhece é a própria World Wide Web. The WWW (**W**orld **W**ide **W**eb).

Em termos simples, REST significa usar o protocolo HTTP do jeito que sempre se planejou.

Conceitos

- Uma típica aplicação que utiliza **REST** deve ser consistente com o paradigma cliente-servidor.
- O **Cliente** inicia uma requisição que é atendida pelo servidor.
- O **Servidor** processa a requisição e retorna a resposta adequada.
- **Requests** e **Responses** são orientados a trabalhar com a ideia de **representação de recursos**.

Arquitetura REST

- A arquitetura **REST** é definida da seguinte forma:
  - ❑ Funções de uma aplicação remota são consideradas recursos.
  - ❑ Cada recurso somente está acessível por meio de um endereçamento padrão.
    - Pelos olhos do protocolo HTTP este endereçamento é definido como URIs.
  - ❑ Recursos possuem interfaces padrão por meio das quais tipos de dados são definidos. Por exemplo **XML, JSON** ou **HTML**.

Arquitetura REST

A troca de dados ocorre de acordo com o protocolo HTTP e deve seguir regras básicas de:

Paradigma Client-server.

Sem controle de estado.

Com a capacidade de armazenamento em cache.

Desenvolvido em camadas.

Independência de interfaces no que diz respeito ao acesso a proxies, firewall, gateways etc.

Princípios do REST

- Identificação dos recursos por URIs.
- Utilização de métodos comuns tais como GET, PUT, DELETE and POST para a manipulação dos recursos.
- Mensagens auto-descritivas por conta da ausência de controle de sessão.

Vantagens

- Facilidade de implementação.
- Acesso simplificado ao serviço. É necessário apenas o browser.
- Boa utilização do protocolo HTTP, cache e servidores proxy.
- Baixo consumo de memória.
- Não há necessidade de implementação além do que o protocolo HTTP oferece.
- Ao utilizar formatos padrão tais como **HTML** ou **XML** aumenta-se a portabilidade e integração com outros sistemas/aplicativos/serviços.
- Instalação independente de serviços, reduzindo o acoplamento entre cliente/servidor.

Desvantagens

- A troca de dados deve ser previamente conhecida.
- **Menos seguro** comparado com outros protocolos, o **SOAP** por exemplo.
- SOAP utiliza **WS-Security** que permite criptografar a troca de mensagens.
  - O que não é o caso do REST.

Serviços RESTFul

- **Exemplos**
  - **Amazon AWS**: Oferece serviços de armazenamento (S3) e interfaces SOAP e REST, onde 90% dos clientes utilizam REST.
  - **Google Maps** utiliza serviços RESTful para cache de mapas.
  - **APIs do Yahoo** utilizam Interfaces RESTful.

REST X RESTFul

(1) **REST** é um estilo arquitetural. É um estilo que esplora os protocolos web existentes.

(1) **RESTful** é utilizado para se referir a serviços que foram implementados dentro de uma arquitetura REST.

Frameworks REST

- **Implementações**
  - ✓ .**NET Open-Source** – OpenRasta.
  - ✓ **ColdFusion** - ColdFusion on Wheels, Mach-II, Taffy.
  - ✓ **Ext JS**.
  - ✓ **Java** - Jt Design Pattern Framework, Wink , **Restlet**, JBoss RESTEasy, Jersey, Apache CXF, NetKernel, Apache Sling, Restfulie, Play Framework.
  - ✓ **Microsoft's** Azure Services Platform e WCF Data Services.
  - ✓ **PHP** DooPHP, Symfony, Zend Framework, CakePHP, Kohana, CodeIgniter, Sapphire, FRAPI, RECESS.
  - ✓ **REST microkernel e** plataforma de software NetKernel.
  - ✓ **Ruby** Ruby on Rails, Sinatra, Restfulie.
  - ✓ **TurboGears2** provê o RestController.
  - ✓ **ZEST** - lightweight Struts-like Web framework.