1. **Introdução**

Nessa aula você irá aprender como construir um *Web Service* REST utilizando PHP. Primeiramente iremos definir o que são serviços REST e em seguida iremos mostrar um exemplo funcional de como construir esse serviço usando um arquivo como base de dados.

1. **Definição**

*Representational State Transfer* (REST), em português Transferência Representacional de Estado, é um estilo de arquitetura de *software* que define um conjunto de restrições a serem usados para a criação de *Web Services*. Os *Web Services* que estão em conformidade com o estilo arquitetural REST, denominados *Web Services* RESTful, fornecem interoperabilidade entre sistemas de computadores na Internet. Os *Web Services* RESTful permitem que os sistemas solicitantes acessem e manipulem representações textuais de recursos da *Web* usando um conjunto uniforme e predefinido de operações sem estado.

"Recursos da Web" foram definidos pela primeira vez na *World Wide Web* como documentos ou arquivos identificados por suas URLs. No entanto, hoje, eles têm uma definição muito mais genérica e abstrata que engloba qualquer coisa ou entidade que pode ser identificada, nomeada, endereçada ou manipulada, da forma que for, na *Web.* Em um *Web Services* RESTful, as solicitações feitas ao URI de um recurso provocarão uma resposta com uma carga útil formatada em HTML, XML, JSON ou algum outro formato. A resposta pode confirmar que alguma alteração foi feita no recurso armazenado e a resposta pode fornecer *links* de hipertexto para outros recursos ou conjuntos de recursos relacionados. Quando o HTTP é usado, como é o mais comum, as operações (métodos HTTP) disponíveis são *GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS* e *TRACE*.

Usando um protocolo sem estado e operações padrão, os sistemas RESTful buscam desempenho, confiabilidade e capacidade de crescimento rápido, reutilizando componentes que podem ser gerenciados e atualizados sem afetar o sistema como um todo, mesmo enquanto estiver em execução.

O termo transferência de estado representacional foi introduzido e definido em 2000 por Roy Fielding em sua tese de doutoramento. A dissertação de Fielding explicou os princípios REST que eram conhecidos como "modelo de objeto HTTP", a partir de 1994, e foram usados ​​no projeto dos padrões HTTP 1.1 e URI (*Uniform Resource Identifiers*). O termo destina-se a evocar uma imagem de como um aplicativo da *Web* bem projetado se comporta: é uma rede de recursos da *Web* (uma máquina de estados virtuais) na qual o usuário avança pelo aplicativo selecionando identificadores de recursos, como http://www.exemplo.com/artigos/21, e operações de recursos, como *GET* ou *POST* (transições de estado do aplicativo), resultando na próxima representação do recurso (o próximo estado do aplicativo) sendo transferida para o usuário final para seu uso.

1. **Exemplo em PHP**

if(isset($\_GET['show'])){

if($\_GET['show'] == 'all'){

$content = file\_get\_contents("db.json",true);

echo $content;

}else{

$content = file\_get\_contents("db.json",true);

$aContent = json\_decode($content);

for ($i=0; $i < count($aContent); $i++) {

if($aContent[$i]->id == intval($\_GET['show'])){

echo json\_encode($aContent[$i]);

}

}

if(!array\_key\_exists(intval($\_GET['show']), $aContent)){

echo '[]';

}

}

}

No exemplo apresentado acima vemos um código muito simples que utiliza um banco de dados (que é na verdade um arquivo JSON). Nesse arquivo temos os seguintes dados:

[

{

"id": 0,

"nome": "How I met your mother"

},

{

"id": 1,

"nome": "Friends"

}

]

O código recebe o parâmetro “*show*” onde é possível fornecer o valor “*all*” caso o usuário queira mostrar todas as séries do arquivo. Esse parâmetro também recebe o “*id”* do filme e mostra os dados caso seja encontrado.

$data = json\_decode(file\_get\_contents('php://input'), true);

if($data != null){

$content = file\_get\_contents("db.json",true);

$aContent = json\_decode($content, true);

array\_push($aContent, $data);

file\_put\_contents("db.json", json\_encode($aContent));

$updated = file\_get\_contents("db.json",true);

echo json\_encode($aContent);

}else{

echo 'No data received!';

}

O código acima mostra como um *POST* pode ser recebido e escrito dentro de um arquivo JSON. Essa simulação de banco de dados mostra que é possível realizar operações de adição simplesmente passando os dados necessários para o *Web Services.*