PUC Minas

RESTful API

Prof. Rommel Carneiro

RESTful API – Tópicos

- Introdução
- Terminologia
- Princípios de Design

RESTful API

GET PUT POST DELETE

RESTful API – Introdução

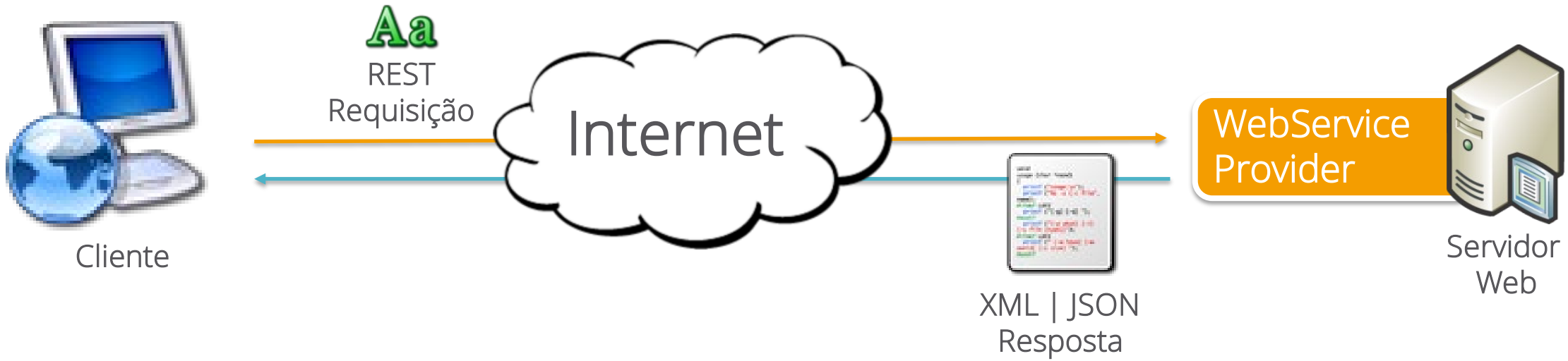
REST é um estilo arquitetural para construção de serviços Web que significa a Transferência de Estado Representacional (Representational State Transfer).

O termo **RESTful** tem a intenção de caracterizar serviços Web que seguem integralmente as recomendações REST, diferentemente daqueles (**RESTlike**) que implementam parcialmente suas recomendações.

RESTful API – Introdução

Motivação: Simplicidade e eficiência do padrão REST

- O cliente se preocupa com a apresentação p/ o cliente e o estado da aplicação
- O servidor se preocupa com o armazenamento dos dados e a lógica do negócio



RESTful API – Terminologia

Termo	Descrição e Exemplos
Recurso	O alvo conceitual de uma referência de hipertexto
Identificador do recurso	URL, URN
Representação	Documento HTML, imagem JPEG

RESTful API – Terminologia

Recurso

- Os recursos são as entidades expostas por um sistema Web via API como usuários, posts, fotos, localidades, bookmarks, empresas
- Recursos possuem um identificador pelo qual são acessíveis, baseado no padrão URI, conforme os seguintes exemplos:
  - `http://myservice.org/user/1`

RESTful API – Princípios de design

Alguns princípios importantes de design de uma API RESTful:

- Uso das definições do protocolo HTTP
- Estrutura uniforme de Endpoint
- Uso da abordagem stateless (sem estado)
- Uso da abordagem HATEOAS
- Controle do versionamento da API
- Controle adequado do cache
- Documentação clara e adequada da API

Fontes:

- Best Practices for Designing a Pragmatic RESTful API - https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api
- RESTful Web Services: The basics, Alex Rodriguez - http://www.ibm.com/developerworks/webservices/library/ws-restful/
- RESTful API Designing guidelines—The best practices - https://hackernoon.com/restful-api-designing-guidelines-the-best-practices-60e1d954e7c9
- RESTful API Design. Best Practices in a Nutshell - https://blog.philipphauer.de/restful-api-design-best-practices/
- API design - https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

RESTful API – Princípios de design

Uso das definições do protocolo HTTP – Métodos HTTP

Utilizar os métodos HTTP de maneira clara em função do seu propósito semântico:

- **GET** – Recuperar um recurso (SELECT)
- **POST** – Criar um recurso no servidor (INSERT)
- **PUT** | **PATCH** – Alterar um estado de um recurso ou atualizá-lo (UPDATE)
- **DELETE** – Remover um recurso (DELETE)
- **OPTIONS** – Lista as operações de um recurso

IMPORTANTE: Métodos GET e parâmetros de query não devem alterar estado de recursos.

RESTful API – Princípios de design

Uso das definições do protocolo HTTP – Códigos de status da resposta HTTP

Code	Propósito	Descrição	Exemplo
1xx	Informacional	Requisição recebida,	Processo em continuidade
2xx	Sucesso	200 – Sucesso na requisição	Requisição de informações (GET) com sucesso
		201 – Recurso Criado	Inclusão de recurso (POST) com sucesso
		202 – Requisição aceita para processamento	Processo assíncrono sem retorno imediato
		204 – Requisição processada com sucesso,	Exclusão de recurso (DELETE) com sucesso
3xx	Redirecionamento	301 – Movido permanentemente	Site transferido de servidor
		302 – Movido temporariamente	Recurso temporário no lugar
4xx	Erro no cliente	400 – Requisição incorreta	Problemas de sintaxe, rotas inexistentes
		401 – Autenticação Requerida	Primeira requisição sem dados de autenticação
		403 – Acesso negado	Recurso privado não acessível pelo requisitante
		404 – Recurso não encontrado	Recurso solicitado não existe
		405 – Método não permitido	PUT ou DELETE em endpoints de GET ou POST
5xx	Erro no servidor	O servidor falhou em completar um pedido aparentemente válido	

RESTful API – Princípios de design

Uso das definições do protocolo HTTP – Uso de MIME Types no Content Type

MIME-Type	Content-Type
JSON	application/json
XML	application/xml
XHTML	application/xhtml+xml

Requisição

```
GET /cliente/2 HTTP/1.1
Host: http://servidor
Accept: application/json
```

Resposta

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 109
{
    id: 2
    name: 'Rommel Vieira Carneiro',
    email: 'rommel@email.com',
}
```

RESTful API – Princípios de design

Estrutura uniforme de Endpoint – CRUD de Recursos

Ação	Operação (CRUD)	Mapeamento da URL
Incluir um curso	**C**REATE	**POST** /cursos/
Obter lista de cursos	**R**ETRIEVE	**GET** /cursos
Obter um curso específico	**R**ETRIEVE	**GET** /cursos/:id
Alterar um curso	**U**PDATE	**PUT** /cursos/:id **PATCH** /cursos/:id
Excluir um curso	**D**ELETE	**DELETE** /cursos/:id

RESTful API – Princípios de design

Estrutura uniforme de Endpoint – Relacionamentos

Ação	Mapeamento da URL
Listar alunos do curso	**GET** /cursos/:id/alunos ou **GET** /alunos/?curso_id=:id
Obter um aluno do curso	**GET** /cursos/:id/alunos/:id
Incluir aluno no curso	**POST** /cursos/:id/alunos
Remover aluno do curso	**DELETE** /cursos/:id/alunos/:id
Listar cursos do aluno	**GET** /alunos/:id/cursos ou **GET** /cursos/?aluno_id=:id
Obter um curso do aluno	**GET** /alunos/:id/cursos/:id

RESTful API – Princípios de design

Estrutura uniforme de Endpoint – CRUD de Recursos

Ação	Mapeamento da URL
Listar cursos	**GET** /cursos
Obter um curso	**GET** /cursos/:id
Pesquisar cursos	**GET** /cursos?search=param
Incluir curso	**POST** /cursos/
Alterar curso	**PUT** /cursos/:id **PATCH** /cursos/:id
Excluir curso	**DELETE** /cursos/:id

RESTful API – Princípios de design

Estrutura uniforme de Endpoint - Exemplo: The Movie DB

- Endpoint: http://api.themoviedb.org/3

Ação	Mapeamento da URL
Busca por empresas	**GET** /search/company?query=xxx&page=x
Busca por filmes	**GET** /search/movie?query=xxx&page=x **GET** /search/movie?year=XXXX&language=xx
Dados de filme específico	**GET** /movie/{movie_id}
Artistas e equipe técnica do filme	**GET** /movie/{movie_id}/credits
Dados de uma empresa específica	**GET** /company/{company_id}
Dados de uma pessoa específica	**GET** /person/{person_id}

RESTful API – Princípios de design

HATEOAS - Hipertexto como mecanismo de estado da aplicação

Resposta p/ saldo de 100,00

```
GET /account/12345 HTTP/1.1

------

HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">100.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
    <link rel="withdraw" href="/account/12345/withdraw" />
    <link rel="transfer" href="/account/12345/transfer" />
    <link rel="close" href="/account/12345/close" />
</account>
```


Fonte: http://en.wikipedia.org/wiki/HATEOAS

RESTful API – Princípios de design

HATEOAS - Hipertexto como mecanismo de estado da aplicação

```
GET /account/12345 HTTP/1.1

------

HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">-25.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
</account>
```

Resposta p/ saldo de -25,00

Fonte: http://en.wikipedia.org/wiki/HATEOAS

Obrigado!