MAURÍCIO MARCOS

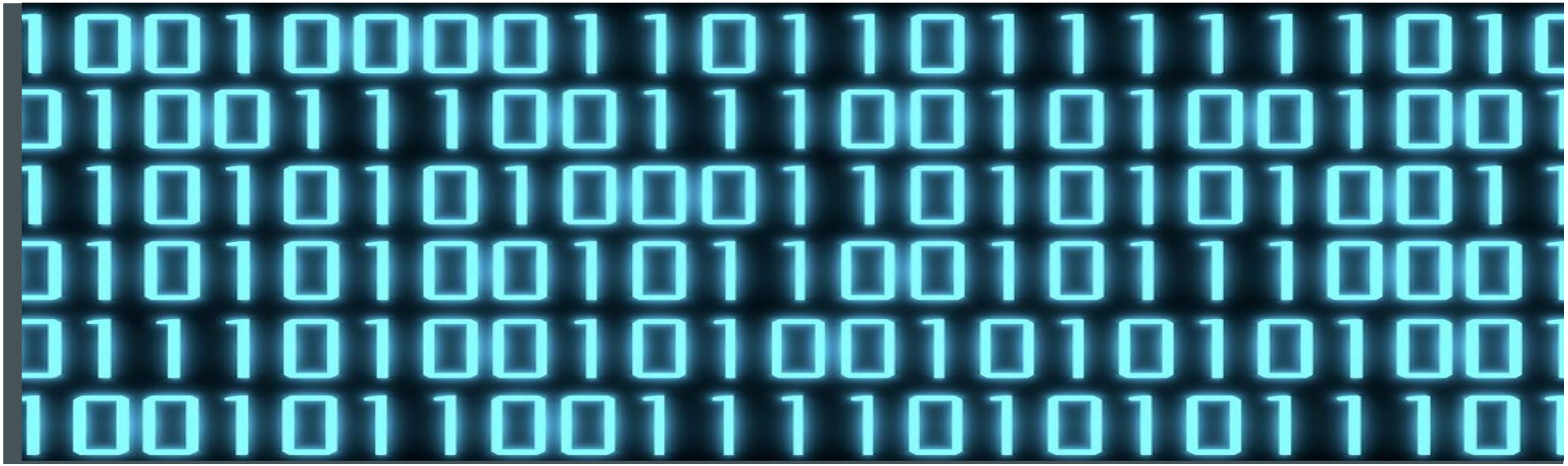
https://www.linkedin.com/in/mmarcos001/

https://github.com/mauriciomarcos

- ✓ **Nome**: Maurício Marcos
- ✓ **Formado em** Análise e Desenvolvimento pela FATEC-Taquaritinga
- ✓ **Pós-graduado** (*latu-senso*, Especialização) em Consultoria em Desenvolvimento de Software Web pela Fatec-São José do Rio Preto
- ✓ **Pós-graduado** (*latu-senso*, Master Business Administration) em Análise de dados com Business Intelligence e Big Data pela UNIFRAN
- ✓ **Pós-graduado** (*latu-senso,* Especialização) em Arquitetura de Software para Plataforma .NET
- ✓ Atualmente atuando como Engenheiro de Software na empresa Beblue

WEB API: IMPLEMENTANDO CONCEITOS REST

POR INTERMÉDIO DO PROTOCOLO HTTP



VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- HTTP é um protocolo que permite a obtenção de recursos, tal como documentos HTML. Esse protocolo é a base do intercâmbio de dados que ocorre na Web.

- O protocolo HTTP trabalha sob a perspectiva da arquitetura *cliente-server* (cliente-servidor), o que significa, que todas as requisições são iniciadas pelo destinatário, por exemplo um navegador Web, e essas requisições são processadas por um Servidor-Web e respondidas aos solicitantes de acordo com o recurso solicitado na requisição.

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- Os clientes e servidores se comunicam trocando mensagens individuais (ao contrário de um fluxo de dados). As mensagens enviadas pelo cliente, são chamadas de ***REQUESTS*** e as mensagens enviadas pelo servidor são chamadas de ***RESPONSES.***

- Exemplificando o vamos imaginar que abrimos um *Brower* e digitamos https://www.google.com**.** Essas quatro letrinhas iniciais *http* é justamente o nome do protocolo que estamos utilizando para se "relacionar" como o endereço google.com. Neste exemplo, é estabelecido por intermédio do protocolo http que tanto o cliente (nosso brower) quanto o servidor (computador remoto que armazenas documentos, imagens, áudios, documentos html e etc) utilizarão as mesmas regras de comunicação, regras essa estabelecida pelo protocolo http.

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

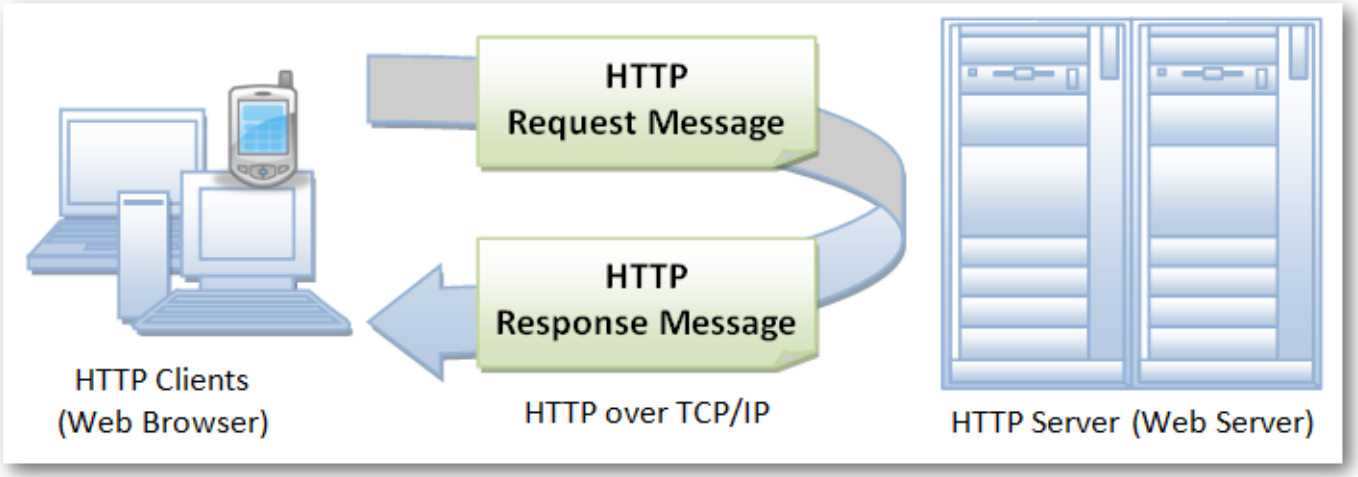
- É importante destacar que todo o transporte de dados, ao qual o protocolo http trafega é conhecido como TCP/IP – protocolo de transporte de dados.

  - ✓ **TCP/IP:** é um conjunto de protocolos sendo o ***INTERNET PROTOCOL (IP)*** responsável pela rota e o ***TRANSMISSION CONTROL PROTOCOL (TCP)*** é responsável por assegurar que a mensagem seja enviada e sem danos aos destinatário

- Dessa forma então podemos concluir que o TCP/IP é o protocolo responsável pelo transporte de dados e o http é o protocolo responsável pelas “regras de comunicação” entre o cliente e servidor.

VISÃO GERAL DO PROTOCOLO HTTP

Solicitação de um *client* para um servidor que responde a essa solicitação por intermédio do protocol http.



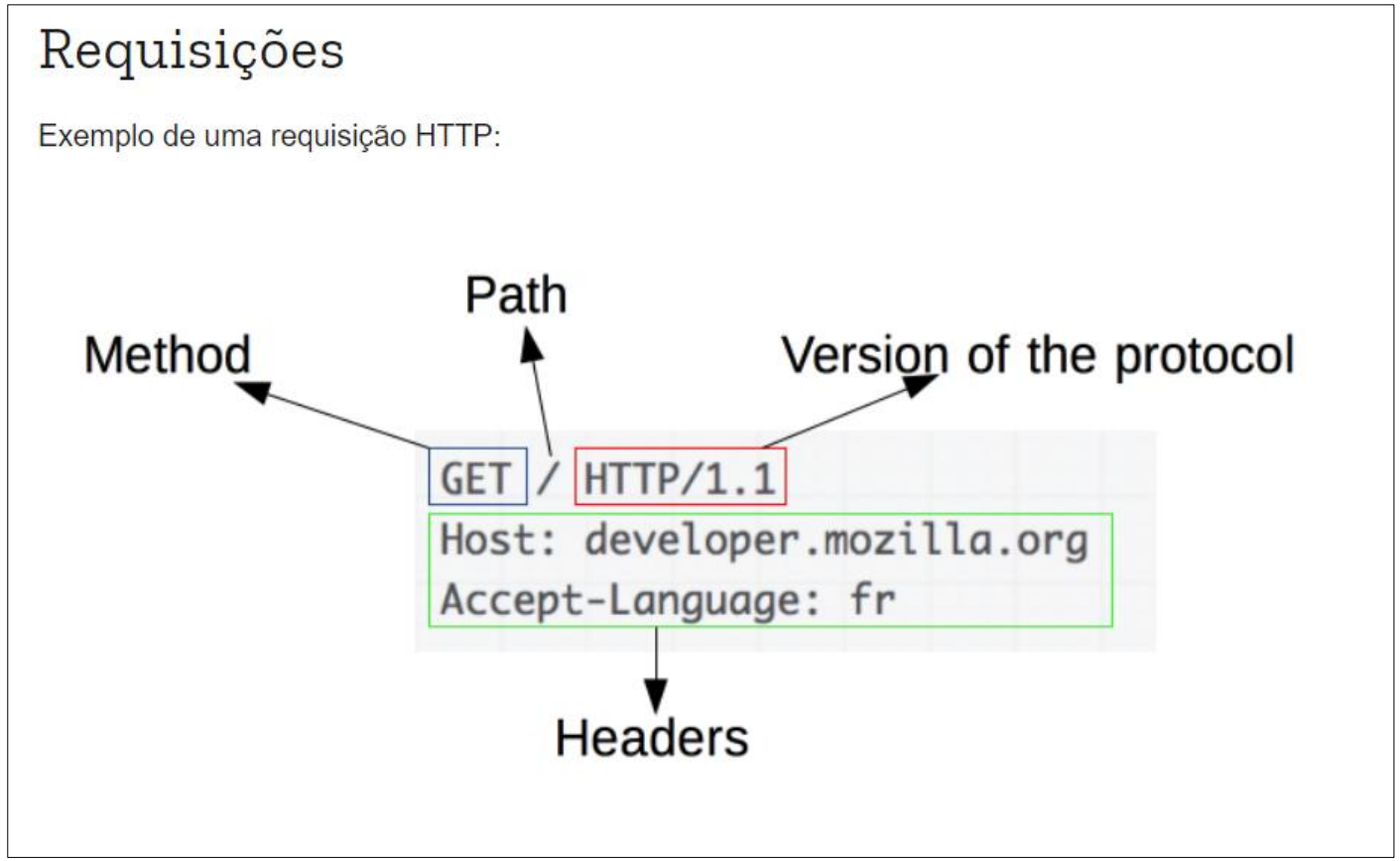
Um método HTTP, geralmente é um verbo como GET, POST, DELETE, PUT, etc, ou um substantivo como OPTIONS ou HEAD que define qual operação o cliente quer fazer. Tipicamente, um cliente quer pegar um recurso (usando GET) ou publicar dados de um formulário HTML (usando POST), embora mais operações podem ser necessárias em outros casos.

O caminho do recurso a ser buscado; a URL do recurso sem os elementos que são de contexto, por exemplo sem o protocolo protocol (http://), o domínio domain (aqui como developer.mozilla.org), ou a porta port TCP (aqui indicada pelo 80 que é ocultado por ser o número da porta padrão)

A versão do protocolo HTTP.

Cabeçalhos opcionais que contém informações adicionais para os servidores.

Ou um corpo de dados, para alguns métodos como POST, similares aos corpos das respostas, que contém o recurso requisitado.



Fonte: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview

Respostas consistem dos seguintes elementos:

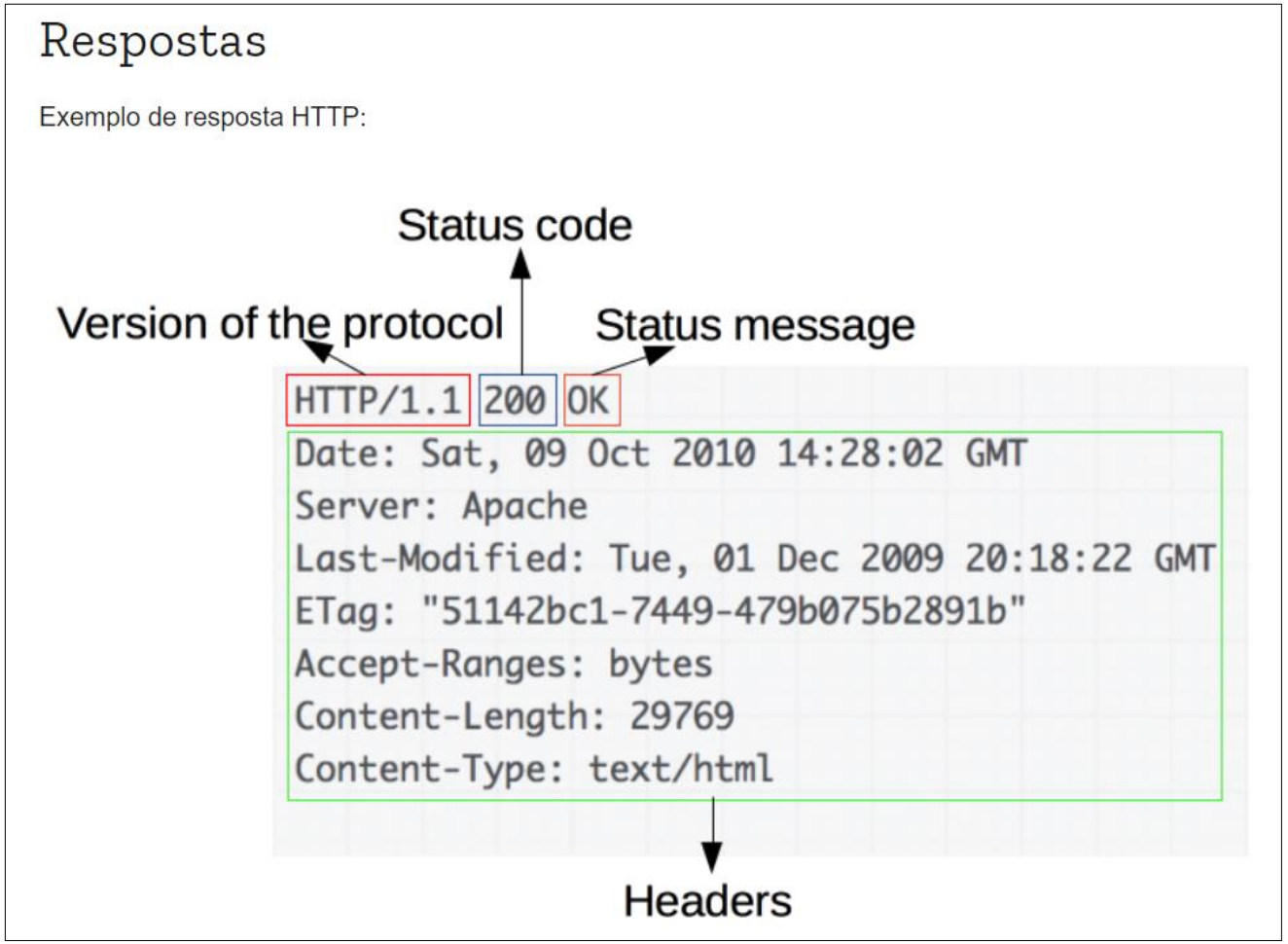
A versão do protocolo HTTP que elas seguem.

Um código de status, indicando se a requisição foi bem sucedida, ou não, e por quê.

Uma mensagem de status, uma pequena descrição informal sobre o código de status.

Cabeçalhos HTTP, como aqueles das requisições.

Opcionalmente, um corpo com dados do recurso requisitado.



Fonte: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- Existe um conceito muito importante no protocolo HTTP que é a **IDEMPOTÊNCIA** de métodos. Esse entendimento é imprescindível quando estamos implementando serviços para web ou Web API baseadas em **REST** (independentemente da linguagem de programação)

- Idempotência quer dizer que a quantidade de requisições que um cliente faça para um servidor, não implica em resultados diferentes, ou seja, um mesmo endpoit (recurso Web – “onde residem as API’s”) pode ser chamado diversas vezes e o resultado sempre será o mesmo.

Idempotente: significa que um endpoint pode ser chamado diversas vezes e apresentar os mesmos resultados (isso se aplica ao resultado e não ao recurso).

Safe: significa que as ações realizadas pelo verbo http são seguras, em outras palavras, são *Read-Only,* sendo assim, o estado do recurso nunca será alterado.

HTTP Method	Idempotente	Safe
OPTIONS	✓	✓
GET	✓	✓
HEAD	✓	✓
PUT	✓	✗
POST	✗	✗
DELETE	✓	✗
PATCH	✗	✗

Fonte: https://www.brunobrito.net.br/api-restful-boas-praticas/

Definições acerca dos verbos /métodos http mais utilizados:

HTTP Method	Descrição
OPTIONS	Retorna os verbos http de um resource e outras opções, como CORS, por exemplo.
GET	Busca um resource
HEAD	Busca apenas o header de um resource
PUT	Atualiza um resource
POST	Cria um resource
DELETE	Remove um resource
PATCH	Atualiza parcialmente um resource

Fonte: https://www.brunobrito.net.br/api-restful-boas-praticas/

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- Por definição, um endpoint (ponto de extremidade), que nada mais é do que uma URI/URL onde seu serviço pode ser acessado por uma aplicação/requisiçã cliente.
  - ✓ URL = Uniform Resource Locator
  - ✓ URI = Uniform Resorce Identifier

- Em resumo, uma URI identifica um recurso Web, enquanto que a URL localiza um recurso, incluindo a forma como acessar o recurso. Por exemplo:
  - ➢ URI: google.com/e-mails
  - ➢ URL: https:google.com/emails

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- Dentro do universo de desenvolvimento de aplicações Web há alguns padrões de nomenclaturas que são sugeridas a serem utilizadas, e entendidas como boas práticas para a construção de Web API's, sendo elas:
  - ✓ A raiz do recurso deve retornar uma coleção, caso desejarmos obter um item específico desse recurso, devemos explicitar utilizando o próximo nível da URI para esse propósito. Exemplo:

    - Para obter todos os usuários de um recurso: **/usuarios**
    - Para obter um usuário específico de um recurso: **/usuarios/9901** (qualquer dados que identifique o recurso naquele endpoint)

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- Exemplos de rotas consideradas dentro do padrão utilizado normalmente para endpoints de uma Web API e seus respectivos métodos http para consumo de um *client*:
  - **GET /usuarios =>** deve retornar uma lista de usuários daquele serviço/Web API;
  - **GET /usuarios/1234** => deve retornar o usuário com o identificador igual a 1234;
  - **POST /usuarios** => deve ser criado um novo recurso (usuário) no servidor remoto;
  - **PUT /usuarios/1234** => deve atualizar o recurso (usuário) com identificação igual a 1234;
  - **PATCH /usuarios/1234** => deve atualizar parcialmente o recurso (usuário) com identificação igual a 1234.
  - **DELETE /usuarios/1234** => deve deletar o recurso com identificação igual a 1234 do servidor remoto.

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- Quando houver algum tipo de relacionamento entre os recursos, onde por exemplo um recurso filho não exista sem um recurso pai, recomenda-se seguir o padrão abaixo para nomearmos nossas URI, dentro do mesmo recurso pai. Exemplo:
  - **GET /usuarios/mauricio/dependentes =>** deve retornar uma lista de dependentes daquele serviço/Web API;
  - **GET /usuarios/mauricio/dependentes/4321** => deve retornar o dependente com o identificador igual a 4321;
  - **POST /usuarios/mauricio/dependentes** => deve ser criado um novo recurso (dependente) no servidor remoto;
  - **PUT /usuarios/mauricio/dependentes/4321** => deve atualizar o recurso (dependente);
  - **PATCH /usuarios/mauricio/dependentes/4321** => deve atualizar parcialmente o recurso (dependente);
  - **DELETE /usuarios/mauricio/dependentes/4321** => deve deletar o recurso com identificação igual a 4321 do servidor remoto.

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- A partir do momento que um servidor recebe uma requisição HTTP, o lado que solicitou essa requisição (aplicação cliente) necessita saber se aquela requisição foi bem sucedida ou não. O protocolo HTTP possui diversos códigos padronizados e cabe ao servidor responder adequadamente os ***STATUS CODE*** correto, de acordo com o cenário ocorrido no servidor.
- O protocolo HTTP possui as seguintes categorias de ***Status Code***:
  - 2xx – Status de sucesso;
  - 3xx – Status pertencente a categoria de redirecionamento;
  - 4xx – Erro/problemas no cliente;
  - 5xx – Erro/problemas no servidor;

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- Categoria 2xx (mas comuns)
  - 200 Ok – Requisição foi bem sucedida.
  - 201 Created – A requisição foi bem sucedida e um novo recurso foi criado como resultado. Esta é a típica resposta enviada após uma requisição POST.
  - 202 Accepted – A requisição foi bem sucedida mas nenhuma ação foi tomada sobre ela. Isto é uma requisição não-comprometedora, o que significa que não há nenhuma maneira no HTTP para enviar uma resposta assíncrona indicando o resultado do processamento solicitado.
  - 204 No Content – Não há conteúdo para enviar para esta solicitação, mas os cabeçalhos podem ser úteis.

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- Categoria 3xx (mais comuns)
  - 301 Moved Permanently – Esse código de resposta significa que a URI do recurso requerido mudou. Provavelmente, a nova URI será especificada na resposta.
  - 304 Not Modified – Essa resposta é usada para questões de cache. Diz ao cliente que a resposta não foi modificada. Portanto, o cliente pode usar a mesma versão em cache da resposta.

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- Categoria 4xx
  - 400 Bad Request – Essa resposta significa que o servidor não entendeu a requisição pois está com uma **sintaxe** inválida.
  - 401 Unauthorized – Embora o padrão HTTP especifique “unauthorized”, semanticamente, essa resposta significa “unauthenticated”, ou seja, o cliente deve se autenticar para obter a resposta solicitada.
  - 404 Not Found – O servidor não pode encontrar o recurso solicitado. Este código de resposta talvez seja o mais famoso, devido à frequência com que ele acontece na Web.
  - 422 Unprocessable Entity – A requisição está bem formatada mas inabilitada para ser seguida devido a erros semânticos.

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- Categoria 5xx
  - 500 Internal Server Error – O servidor encontrou uma situação com o qual não sabe lidar.
  - 501 Not Implemented – O método da requisição não é suportado pelo servidor e não pode ser manipulado. Os únicos métodos exigidos que servidores suportem (e portanto não devem retornar este código) são **GET** e **HEAD**

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

- **JSON** é um acrônimo de *Javascript Object Notation*, que na prática é um mecanismo de estruturação e/ou formatação de dados textuais que possibilita o intercâmbio de dados em um formato mais leve, e também, facilmente compreensível ao seres humanos. Não somente o JSON pode ser utilizado quando falamos de API's REST, um outro formato bastante conhecido mas que atualmente, está em desuso em aplicações modernas é o formato **XML**..
- Mais informações sobre JSON pode ser conferida em: https://www.json.org/json-pt.html

VISÃO GERAL DO PROTOCOLO HTTP E CONCEITOS REST

Formato JSON

```json
{
    "nome": "João da Silva",
    "idade": 20,
    "matricula": "2018123490",
    "curso": "Sistemas de Informação",
    "cadeiras": [
        "Estrutura de Dados",
        "Organização de Computadores",
        "Matemática Discreta"
    ]
}
```

Formato XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created from PDF via Acrobat SaveAsXML -->
<!-- Mapping Table version: 28-February-2003 -->
<TaggedPDF-doc>
<?xpacket begin=" id='W5M0MpCehiHzreSzNTczkc9d'?>
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta x:xmptk="Adobe XMP Core 5.2-c001 63.139439, 2010/09/27-13:37:26 " xmlns:x="adobe:ns:meta/">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description xmlns:xmp="http://ns.adobe.com/xap/1.0/" rdf:about="">
<xmp:CreateDate>2011-04-06T17:22:05Z</xmp:CreateDate>
<xmp:CreatorTool>ESRI ArcSOC 9.2.0.1324</xmp:CreatorTool>
<xmp:ModifyDate>2011-04-07T08:17:15-06:00</xmp:ModifyDate>
<xmp:MetadataDate>2011-04-07T08:17:15-06:00</xmp:MetadataDate>
</rdf:Description>
<rdf:Description rdf:about="" xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">
<xmpMM:DocumentID>uuid:323e04f1-1033-4485-be82-e60a7573f2ec</xmpMM:DocumentID>
<xmpMM:InstanceID>uuid:6329c275-adcd-4082-bf5a-708d7846e55c</xmpMM:InstanceID>
</rdf:Description>
<rdf:Description rdf:about="" xmlns:dc="http://purl.org/dc/elements/1.1/">
<dc:format>xml</dc:format>
</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>
<?xpacket end='r'?>
<Figure>
<ImageData src="images/WA_Dayton_20110406_TM_geo_img_0.jpg"/>
</Figure>
</TaggedPDF-doc>
```

OBRIGADO PELA ATENÇÃO E PARTICIPAÇÃO!

200 OK – “Fim”