

# Consumindo uma API REST com JavaScript e Bootstrap

Prof. Daniel Di Domenico  
[daniel.domenico@ifpr.edu.br](mailto:daniel.domenico@ifpr.edu.br)



# Objetivo das aulas

- **Utilizar HTML, JavaScript, AJAX e Bootstrap a fim de criar uma interface para consumir uma API REST**
- Ferramentas necessárias:
  - **HTML e JavaScript** (já disponíveis no navegador Web)
  - **AJAX**: técnica de desenvolvimento para WEB que permite executar requisições a um servidor em segundo plano
    - Utiliza JavaScript
  - **Bootstrap**: framework de CSS para adicionar estilo às páginas
    - Suporta responsividade
  - IDE de desenvolvimento VSCode
  - API REST desenvolvida em SpringBoot com Java

# Bootstrap

- Versão 4.1.3
  - Documentação em português
  - **<https://getbootstrap.com.br/>**

# Adicionar o Bootstrap ao HTML

- <https://getbootstrap.com.br/docs/4.1/getting-started/introduction/>
- **Passos:**
  - **1-** Incluir o arquivo CSS na tag HEAD
    - É preciso ter as tags META conforme a documentação
  - **2-** Incluir os arquivos JavaScript antes do final da tag BODY
- **Funcionou?**
  - Verificar se a fonte da página foi alterada

# Tabela com o Bootstrap

- <https://getbootstrap.com.br/docs/4.1/content/tables/>
- Estilizar uma tabela:
  - .table
- Linhas *zebradas*:
  - .table-striped
- Bordas:
  - .table-bordered

# Links e botões com o Bootstrap

- <https://getbootstrap.com.br/docs/4.1/components/buttons/>
- Estilizar um botão:
  - .btn
- Botões primários:
  - .btn-primary .btn-secondary .btn-success .btn-danger .....
- Botões outline (sem cor de fundo):
  - .btn-outline-primary .btn-outline-secondary .btn-outline-success .btn-outline-danger .....

# Formulários com Bootstrap

- <https://getbootstrap.com.br/docs/4.1/components/forms/>
- Grupo (DIV com label + campo):
  - .form-group
- Campo:
  - .form-control

# JavaScript

- JavaScript (JS):
  - **Linguagem de programação**
    - Linguagem de script e interpretada
  - Também conhecida como linguagem de script para páginas Web
    - É suportada por todos os navegadores modernos (Chrome, Firefox, Safari...)
    - Comandos são executados pelo navegador
  - **ATENÇÃO:** não se deve confundir com Java
    - Ambas são linguagens de programação, mas possuem sintaxe, semânticas e usos muito diferentes



# Características do JS

- Código fonte JS é incluído junto ao HTML
  - Pode ser feito de diferentes formas
- Programação é dirigida por eventos e orientada a objetos
- Tipagem do JS
  - Fraca: pode-se alterar o tipo de uma variável
  - Dinâmica: a variável assume o tipo do valor atribuído
- JS é **case-sensitive**
  - **Diferencia letras maiúsculas de minúsculas**
  - Utilize o padrão camelCase

# AJAX

- **Asynchronous JavaScript And XML**
  - Técnica de desenvolvimento para WEB que permite executar requisições a um servidor em segundo plano
    - Não é uma linguagem de programação
  - Utiliza JavaScript
- **AJAX permite:**
  - Atualizar o HTML da página sem recarregá-la
  - Comunicação com o servidor após recarregar a página
  - Busca e envio de dados para o servidor WEB em segundo plano
    - Modo síncrono ou assíncrono
- [https://www.w3schools.com/js/js\\_ajax\\_intro.asp](https://www.w3schools.com/js/js_ajax_intro.asp)

# XMLHttpRequest

- **XMLHttpRequest** é o objeto JavaScript que permite a comunicação com um servidor WEB em segundo plano
- O objeto permite:
  - **1-** Criar uma requisição ao servidor utilizando os verbos HTTP
    - GET, POST, PUT, DELETE....
  - **2-** Definir se a requisição será enviada de forma síncrona ou assíncrona
    - Se assíncrona, é necessário definir uma função de retorno (callback)
  - **3-** Receber a resposta da requisição em dois formatos:
    - Texto (pode ser um JSON)
    - XML

# AJAX: requisição síncrona

- **Exemplo de requisição síncrona:**

```
var url = 'http://localhost:8080/api';  
var xhttp = new XMLHttpRequest();  
xhttp.open('GET', url, false);  
  
xhttp.send();  
  
var retornoTexto = xhttp.responseText;  
//var retornoXML = xhttp.responseXML;  
console.log(retornoTexto);
```

**false:** requisição síncrona (aguarda a resposta do servidor)

# AJAX: requisição assíncrona

- **Exemplo de requisição assíncrona:**

```
var url = 'http://localhost:8080/api';  
var xhttp = new XMLHttpRequest();  
xhttp.open('GET', url, true);  
  
xhttp.onload = function() {  
    //Executado após a resposta do servidor  
    var retornoTexto = xhttp.responseText;  
    console.log(retornoTexto);  
}  
  
xhttp.send();
```

**true:** requisição  
assíncrona

# AJAX: recebimento de dados JSON

- **Exemplo de requisição com JSON:**

```
var url = 'http://localhost:8080/api';  
var xhttp = new XMLHttpRequest();  
xhttp.open('GET', url, true);  
  
xhttp.send();  
  
var objeto = JSON.parse(xhttp.responseText);
```

O retorno JSON foi  
convertido para um  
objeto JavaScript

# AJAX: envio de dados JSON

- **Exemplo de envio de requisição com JSON:**

```
var objetoJson = {  
    "par": 1  
}  
  
var url = 'http://localhost:8080/api';  
var xhttp = new XMLHttpRequest();  
xhttp.open('POST', url, true);  
xhttp.setRequestHeader('Content-type',  
    'application/json');  
  
xhttp.send(JSON.stringify(objetoJson));  
  
var retornoTexto = xhttp.responseText;  
console.log(retornoTexto);
```

Objeto JavaScript  
convertido para  
envio na  
requisição