UNINOVE

# Tecnologia Ajax

APRESENTAR OS CONCEITOS INTRODUTÓRIOS SOBRE A TECNOLOGIA AJAX E SUA UTILIZAÇÃO EM SISTEMAS DINÂMICOS EM PÁGINAS WEB.

AUTOR(A): PROF. EDSON MELO DE SOUZA

## 1. Introdução

A navegação na internet nos proporciona um mundo quase que infinito de opções para o conhecimento, diversão, entre outros. É comum durante uma navegação acessarmos páginas de sites que possuem visuais totalmente diferentes e com funcionalidades variadas. As páginas da internet estão a cada dia evoluindo, trazendo uma nova experiência ao usuário, na medida que é possível personalizar uma página a nosso gosto, acessar recursos de áudio, vídeo, textos e muito mais.

Entretanto, existem alguns obstáculos que geram descontentamento durante o acesso a determinada informação ou recurso de uma página. Um exemplo disso é quando estamos realizando um cadastro e, quando enviamos pressionamos um botão para prosseguir, somos levados para outra página e recebemos a informação que algo não deu certo, ou seja, ocorreu um erro ou então ficou faltando preencher algum campo ou selecionar alguma opção.

Isso ocorre porque quando clicamos em um botão para prosseguir, são realizadas diversas operações nos bastidores do navegador e do computador que fazem com o que os dados digitados sejam enviados para algum lugar. Esse lugar é denominado servidor. Se o servidor é quem recebe os dados, então quem os envia é o cliente, que neste caso é um navegador tal como o Internet Explorer, Firefox, Google Chrome, entre outros.

Para melhorar a relação entre o cliente e o servidor, existem tecnologias e técnicas que auxiliam no processo, fazendo com que a navegação seja simplificada e, consequentemente, trazendo dinâmica para a página.

Neste tópico será abordado o AJAX, que é um conjunto de recursos para o desenvolvimento de aplicações dinâmicas que fornece funcionalidades para transferência de dados entre um cliente e um servidor.

Também serão introduzidos os conceitos de Cliente/Servidor, Sincronismo, Assincronismo, Requisição e Resposta, além de mostrar situações onde o AJAX pode ser empregado.

# 2. Cliente/Servidor

O termo cliente/servidor, também muito conhecido como *client/server* e, segundo Battisti (2011), "É uma arquitetura onde o processamento da informação é dividido em módulos ou processos distintos. Um processo é responsável pela manutenção da informação (Servidor), enquanto que o outro é responsável pela obtenção dos dados (Cliente).".

De outra forma, Vasckevitch (1995) diz que cliente/servidor "É uma abordagem da computação que separa os processos em plataformas independentes que interagem, permitindo que os recursos sejam compartilhados enquanto se obtém o máximo de benefício de cada dispositivo diferente, ou seja, Cliente/Servidor é um modelo lógico.".

Com base nas definições podemos dizer então que, quando estamos navegando na internet, o nosso computador é o cliente e quem está fornecendo os dados é o servidor.

O servidor, também conhecido como *Host*, é uma máquina (*Hardware*) que fornece serviços ou programas (*Software*) como: páginas web; troca de e-mail; acesso à internet; *streaming* de vídeo; etc., compartilhando recursos com seus clientes. Esse compartilhamento é realizado por meio de solicitações conhecidas como requisições (*request*).

Uma requisição ocorre quando um cliente solicita algo do servidor, por exemplo quando digitamos um endereço no navegador e pressionamos "Enter" ou então clicamos em "Ir", ou seja, "estamos" pedindo ao servidor que nos envie os dados que estão relacionados com o endereço que digitamos.

Quando o servidor recebe a solicitação (*request*) ele executa os procedimentos para poder atender à solicitação do cliente e, quando tudo estiver pronto, ele devolve para o cliente o que foi solicitado. Este processo de resposta é denominado *response*.

Usando o mesmo exemplo anterior, quando "acessamos" o endereço de uma página, estamos solicitando ao servidor que nos envie a página web que está relacionada aquele endereço. Portanto, o processamento da página ocorre no lado do cliente, o servidor apenas envia a página.

Vamos ver algumas características do cliente e do servidor:

# Cliente

- Faz solicitação ao servidor (pode fazer mais de uma requisição ao mesmo tempo)
- Aguarda resposta
- Recebe resposta
- Utiliza recursos de rede

# Servidor

- Aguarda solicitações o tempo todo
- Atende, ou não, o pedido dos clientes

- Fornece recursos de rede

Normalmente a interação é feita diretamente com o cliente, mas também pode receber requisições de outros servidores.

Aqui foi descrita apenas uma parte muito pequena desta arquitetura (Cliente/Servidor), mas você pode perceber a importância do entendimento dos conceitos apresentados. É bem provável que você já tenha lido e até mesmo ouvido sobre eles e, talvez, nem imaginasse que para acessar uma simples página há tantos elementos envolvidos.

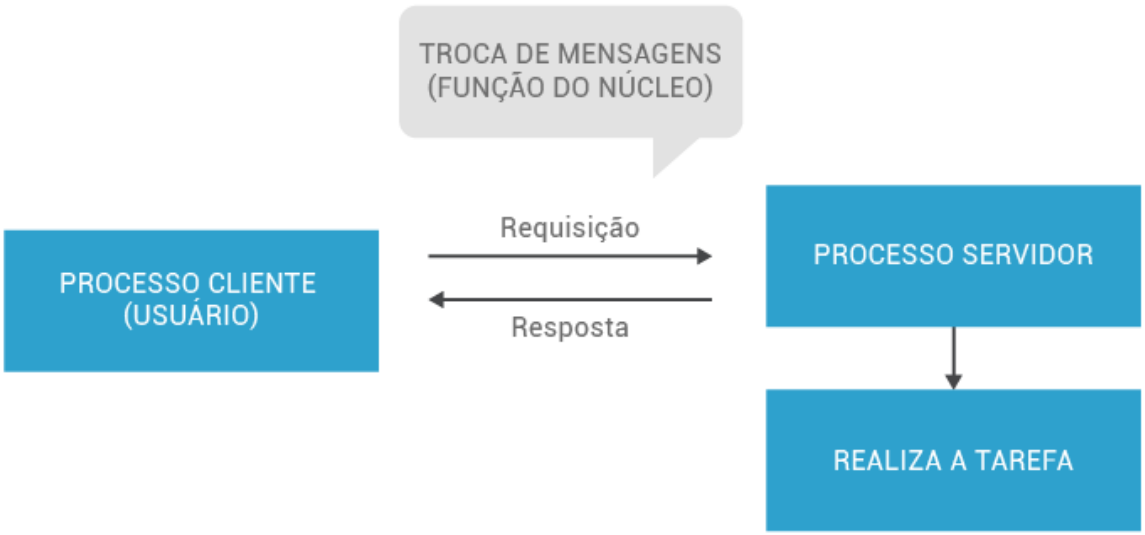
Apesar de agora sabermos vários termos importantes, ainda temos que aprender um pouco mais de como as coisas funcionam. Você já percebeu que quando acessa determinados endereço na internet o navegador demora um pouquinho ou então, na linguagem popular, fica "pensando" para carregar a página? Fique tranquilo, não é a sua internet que é lenta, mas sim o servidor que está demorando para responder. Essa espera ocorre por diversos fatores como: número elevado de requisições ao mesmo tempo, conteúdo a ser enviado ao cliente é muito grande, a internet está lenta, entre outros.

Essa espera ocorre pelo fato da comunicação do cliente com o servidor ser realizada de forma síncrona, ou seja, a requisição é atendida seguindo uma ordem. É necessário que o cliente e o servidor estejam sincronizados, desde o início da requisição até o término dela. Caso haja a perda ou falta de sincronia, a requisição não será atendida, pois algo aconteceu no meio da transmissão. Um exemplo do cotidiano é quando estamos assistindo um filme legendado e a fala não está ocorrendo aos mesmo tempo que a legenda, ou seja, está sem sincronia.

Por outro lado, temos as comunicações assíncronas que são um tipo de comunicação onde os dados são enviados com uma "*marcação*" ou "*flag*", não dependendo de uma conexão contínua entre as partes, ou seja, os dados podem ser enviados em "pedaços".

Agora que sabemos os conceitos de síncrono e assíncrono, é hora de avançarmos e vermos como melhorar experiência do usuário em uma navegação utilizando o AJAX.

Na figura a seguir é mostrada a relação entre o cliente e o servidor com base nos conceitos apresentados.




Legenda: ARQUITETURA CLIENTE/SERVIDOR

# 3. AJAX

O AJAX (*Asynchronous JavaScript And XML*) não é uma linguagem, mas sim um conjunto de recursos para o desenvolvimento de aplicações dinâmicas baseadas em intensa interação de usuários que utiliza o objeto XMLHttpRequest que fornece a funcionalidade para transferência de dados entre um cliente (navegador) e um servidor. Os recursos que compõem o AJAX são JavaScript, XML, *Document Object Model* (DOM), porém os principais recursos são JavaScript e XML. (*Material AVA Uninove*).

A característica que apresenta maior destaque nesta tecnologia é a sua natureza assíncrona, ou seja, as operações podem ser realizadas sem que haja a necessidade do recarregamento da página HTML.

## SAIBA MAIS!

DOM – https://developer.mozilla.org/pt-BR/docs/DOM/Referencia_do_DOM (https://developer.mozilla.org/pt-BR/docs/DOM/Referencia_do_DOM)

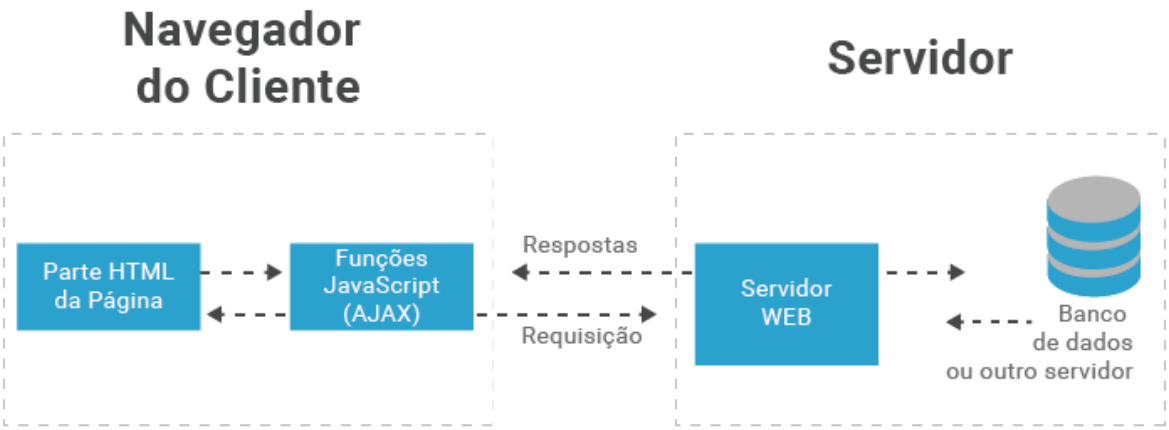
XML – https://www.w3schools.com/xml/ (https://www.w3schools.com/xml/)

XMLHttpRequest – https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest (https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest)

JavaScript – https://developer.mozilla.org/pt-BR/docs/Aprender/JavaScript (https://developer.mozilla.org/pt-BR/docs/Aprender/JavaScript)

# 3.1 Funcionamento

O AJAX é executado no navegador do usuário e, quando um dado é submetido ao servidor (requisição), por exemplo, você deve enviar esses dados a uma função JavaScript de envio e receber os resultados em uma função JavaScript de resposta. Dessa função você renderiza o retorno na página HTML. Em outras palavras, você envia uma requisição a um servidor e recebe a resposta, podendo manipular os dados recebidos utilizando uma função Javascript. Isso pode ser observado na figura a seguir:




## ATENçãO!

Para submeter dados ao servidor com AJAX você precisa de uma função JavaScript e para receber, de outra.

Para fazer uma requisição HTTP ao servidor usando JavaScript, é necessário instanciar uma classe que fornece essa funcionalidade. Essa classe é o XMLHttpRequest que foi originalmente introduzida no Internet Explorer (IE) como um objeto *ActiveX* chamado XMLHTTP. Posteriormente outros navegadores também incluíram esta classe para manterem a compatibilidade das páginas que utilizam esse recurso.
As requisições podem ser realizadas utilizando tanto o método POST como o GET. O método POST é mais seguro e tem uma capacidade de dados melhor que o GET. Nesse método uma conexão paralela é aberta e os dados são passados por ela. Não há restrição referente ao tamanho e os dados não são visíveis ao usuário. A função do método GET é simplesmente recuperar um recurso existente no servidor. O resultado de uma requisição GET fica no histórico do navegador. (Macêdo, 2017).

## SAIBA MAIS!

O que é o ActiveX - http://searchenterprisedesktop.techtarget.com/definition/ActiveX (http://searchenterprisedesktop.techtarget.com/definition/ActiveX)

> Controles ActiveX - https://www.tecmundo.com.br/internet-explorer/6750-afinal-o-que-sao-controles-activex-e-por-que-voce-deve-instala-los-as-vezes-.htm (https://www.tecmundo.com.br/internet-explorer/6750-afinal-o-que-sao-controles-activex-e-por-que-voce-deve-instala-los-as-vezes-.htm)

A seguir será mostrado um exemplo de como utilizar o AJAX para realizar uma requisição ao site República Virtual que fornece um serviço de consulta de endereços por CEP. Ao informar o CEP em uma caixa de texto e clicar em "Buscar", será realizada uma requisição AJAX e o endereço será retornado, caso o CEP seja válido. *Neste exemplo não está sendo realizada a verificação do erro para o CEP informado*.

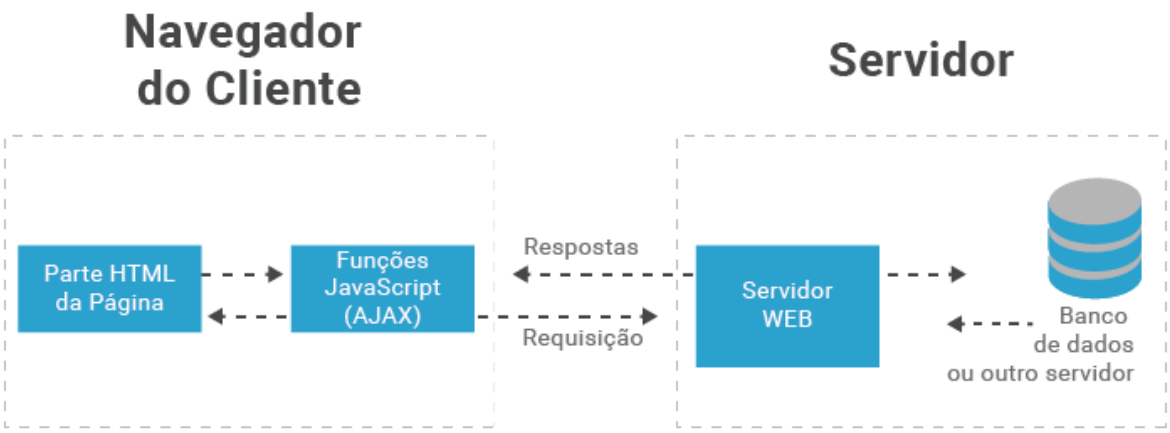
*Obs.: Os comentários estão incluídos diretamente no código para facilitar o entendimento.*

```
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <!-- Criação de uma função em JavaScript -->
        <script type="text/javascript">
            function buscaCep(varCep)
            {
                // Verifica se o campo CEP foi preenchido
                if (varCep.length == 0)
                {
                    // Não realiza a busca
                    return;
                }

                // Verifica se o recurso está disponível
                if (window.XMLHttpRequest)
                {
                    // Código para IE7+, Firefox, Chrome, Opera, Safari,
                    // Se estiver, cria um novo objeto XMLHttpRequest
                    xmlhttp = new XMLHttpRequest();
                } else
                {
                    // Código para IE6, IE5
                    // Criação do objeto baseado em ActiveX
                    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
                }

                // Coloca o método em estado de "atenção", ou seja,
                // fica aguardando uma ação
                xmlhttp.onreadystatechange = function ()
                {
                    // Verifica se o status da requisição está funcionan
                    if (xmlhttp.readyState == 4 && xmlhttp.status == 200
                    {
                        // Declara uma variável para receber a resposta
                        var resposta = JSON.parse(xmlhttp.responseText);

                        // Verifica se o endereço existe para o CEP info
```

```
40.                         if(resposta.resultado != 0)
41.                         {
42.                         // Abaixo são configurados os elementos <span>
43.                         // para receber os dados originados da requisiçã
44.                         document.getElementById("tipo_logradouro").inner
45.                         document.getElementById("logradouro").innerHTML
46.                         document.getElementById("bairro").innerHTML = re
47.                         document.getElementById("cidade").innerHTML = re
48.                         document.getElementById("estado").innerHTML = re
49.
50.                         // Limpa o campo de erro, caso ele esteja preenc
51.                         document.getElementById("erro").innerHTML = "";
52.
53.                         }else
54.                         {
55.                             // Emite uma mensagem de erro caso o endereç
56.                             document.getElementById("erro").innerHTML =
57.
58.                             // Limpa os campos de exibição, caso haja al
59.                             document.getElementById("tipo_logradouro").i
60.                             document.getElementById("logradouro").innerH
61.                             document.getElementById("bairro").innerHTML
62.                             document.getElementById("cidade").innerHTML
63.                             document.getElementById("estado").innerHTML
64.                         }
65.                     }
66.                 }
67.
68.                 // Realiza a montagem da URL requisitada, concatenando o
69.                 xmlhttp.open("GET", "http://cep.republicavirtual.com.br/
70.
71.                 // Executa a requisição AJAX
72.                 xmlhttp.send();
73.             }
74.         </script>
75.     </head>
76.     <body>
77.         <h1>Pesquisa de endereço por CEP informado</h1>
78.         <!-- Campo de texto para digitação do CEP-->
```

```
79.         <label>Digite o CEP a ser pesquisado:</label> 
80.         <input type="text" id="cep" name="cep" onchange="buscaCep(this.v
81.         <a href="#">Pesquisar</a><br><br>
82.         <span id="erro"></span>
83.
84.         <h3>Resultado da Pesquisa</h3>
85.
86.         <!-- TAGs span que receberão os valores retornados da requisição
87.         <span id="tipo_logradouro"></span>
88.         <span id="logradouro"></span><br>
89.         <span id="bairro"></span><br>
90.         <span id="cidade"></span><br>
91.         <span id="estado"></span>
92.
93.     </body>
94. </html>
```

No vídeo a seguir é mostrado o resultado da consulta, sendo utilizado o modo de depuração do navegador Chrome (Tecla F12).




Legenda: DEMONSTRAçãO AJAX

Na linha 80 é utilizado um recurso do HTML que verifica o comportamento (evento) do objeto e, neste caso, o evento "onchange" (quando sofrer alteração) é disparado, a função JavaScript buscaCep() é invocada e tem o seu parâmetro informado utilizando a instrução *this.value*, significando que o valor do

CEP digitado será informado para a função. A partir deste ponto o controle do processamento passa para a função que acionará o AJAX.

## 3.2 Quando Utilizar o AJAX

Uma grande vantagem na utilização do AJAX é o fato de não haver a necessidade de recarregar a página quando uma requisição é realizada, ou seja, você envia uma requisição da página ao servidor e o processamento prossegue sem aguardar o retorno. Quando o retorno chega, os resultados podem ser renderizados na mesma página. O AJAX também permite o processamento síncrono, embora a grande vantagem está no processamento assíncrono.
A seguir, são apresentadas as vantagens e desvantagens na utilização desta tecnologia:

## Vantagens

- Quando você entra em um site de busca e à medida que você vai digitando o que deseja consultar, vai aparecendo um conjunto de sugestões ou resultados, isso normalmente é realizado com AJAX. A cada nova letra que você digita no campo de busca, tal conteúdo é enviado ao servidor, que busca no banco de dados e retorna os resultados normalmente na forma de uma lista que é armazenada em um arquivo XML. Esse arquivo é retornado, lido e carregado na página.
- Quando você seleciona um valor em uma caixa de combinação (combo) e, com base nesse valor, outra caixa é preenchida. Por exemplo, ao selecionar um nome de fabricante de automóveis, uma caixa é preenchida com todos os modelos do fabricante selecionado. Quando você seleciona o nome do fabricante, esse valor é enviado ao servidor, que busca no banco de dados e os resultados são retornados em um arquivo XML que é lido e carregada outra caixa de combinação.
- Quando você digita um valor em um campo-chave e aparece, na mesma página, uma mensagem dizendo que esse valor já está cadastrado. Ao digitar o valor, ele é enviando para o servidor, que busca no banco de dados. Se encontrado, uma mensagem é retornada e exibida na mesma página.

## Desvantagens

- Apesar das vantagens, há problemas também com esta tecnologia, pois o AJAX usa programação procedural e orientada a objetos com JavaScript, que naturalmente é uma linguagem com sintaxe complexa.
- Outro problema é que dá para desativar o processamento de JavaScript nas configurações do navegador. Dessa forma, se este recurso estiver desativado, todo o conteúdo AJAX da página não irá funcionar.

### SAIBA MAIS!

> Mozila Developer Network - https://developer.mozilla.org/pt-BR/docs/AJAX (https://developer.mozilla.org/pt-BR/docs/AJAX)

# Resumo

Neste tópico foram abordados os conceitos de cliente/servidor, requsições, respostas, conexão síncrona e assiíncrona, além do AJAX, abordando sua utilização, vantagens e desvantagens, além de um exemplo de como utilizar em uma página Web para recuperar endereços por meio da pesquisa por CEP.

# Conclusão

A tecnologia AJAX é amplamente utilizada em páginas Web para dar “vida”, ou seja, ajudar a tornar as páginas mais dinâmicas. Apesar de ser muito útil e vantajosa, também apresenta desvantagens como desempenho, complexidade de implementação e dependência de outras tecnologias, como o Javascript. Portanto, deve ser utilizada com cautela para não haver perda de desempenho e risco de não funcionamento dos recursos desejados.

# ATIVIDADE FINAL

## O AJAX é classificado como:

A. Um conjunto de recursos que utiliza o objeto XMLHttpRequest.
B. Uma linguagem de programação baseada em Javascript.
C. Um conjunto de linguagens DOM com suporte ao XML.
D. Uma tecnologia baseada exclusivamente no lado do servidor.

## Qual a maior vantagem em utilizar o AJAX?

A. Permite que não haja o *refresh* da página na obtenção dos dados externos.
B. Permite ao navegador realizar o carregamento dos dados de forma mais otimizada.
C. Possui baixa complexidade de implementação.
D. Não necessita de outros recursos para ser executado, ou seja, tem independência de implementação e execução.

## Entre as alternativas a seguir, qual não pertence a tecnologia AJAX?

A. Java.
B. XML.

C. JavaScript.

D. DOM

# REFERÊNCIA

ABRAHIM, Marcia. Arquitetura cliente servidor. Disponível em: <https://pt.slideshare.net/marciaabrahim/arquitetura-cliente-servidor (https://pt.slideshare.net/marciaabrahim/arquitetura-cliente-servidor)>. Acesso em: 9 mai. 2017.

AVA Uninove. Serviços de Internet – Ajax. 2013.

BALLARD, Phil; MONCUR, Michael. Sams teach yourself Ajax, JavaScript, and PHP all in one. Pearson Education, 2008.

BATTISTI, Júlio. SQL server 2005: administração & desenvolvimento: curso completo. Axcel Books, 2005.

MACÊDO, Diogo. Entendendo a diferença entre os métodos POST e GET. Disponível em: < http://www.diegomacedo.com.br/entendendo-a-diferenca-entre-os-metodos-get-e-post-no-php/ (http://www.diegomacedo.com.br/entendendo-a-diferenca-entre-os-metodos-get-e-post-no-php/)>. Acesso em: 5 de abr. 2017.

MUNIZ, Wanderley. Mozilla Developer Network (Ed.). Primeiros passos: Uma introdução ao AJAX. 2017. Disponível em: <https://developer.mozilla.org/pt-BR/docs/AJAX>. Acesso em: 05 mai. 2017.

FAVARETTO, B. stackOverflow: Qual a diferença entre comunicação assíncrona e síncrona? Disponível em: <https://pt.stackoverflow.com/questions/51268/qual-a-diferen%C3%A7a-entre-comunica%C3%A7%C3%A3o-ass%C3%ADncrona-e-s%C3%ADncrona (https://pt.stackoverflow.com/questions/51268/qual-a-diferen%C3%A7a-entre-comunica%C3%A7%C3%A3o-ass%C3%ADncrona-e-s%C3%ADncrona)>. Acesso em: 11 mai. 2017.

SCHÖLNAST, Hubrt. How does Synchronous and Asynchronous communication work exactly. Disponível em: <http://stackoverflow.com/questions/10102580/how-does-synchronous-and-asynchronous-communication-work-exactly/10102768#10102768 (http://stackoverflow.com/questions/10102580/how-does-synchronous-and-asynchronous-communication-work-exactly/10102768#10102768)>. Acesso em: 11 mai. 2017.

TERUEL, Evandro Carlos. *Arquitetura de Sistemas para Web com Java utilizando Design Pattern e Frameworks*.Rio de Janeiro: Ciência Moderna, 2012.

VASKEVITCH, David. Estratégias cliente/servidor. Berkeley Brasil Editora, 1995.