

LINGUAGENS DE PROGRAMAÇÃO

Prof. Laercio Metzner



2015

1ª Edição



Copyright © UNIASSELVI 2015

Elaboração:

Prof. Laercio Metzner

Revisão, Diagramação e Produção:

Centro Universitário Leonardo da Vinci – UNIASSELVI

Ficha catalográfica elaborada na fonte pela Biblioteca Dante Alighieri

UNIASSELVI – Indaial.

005.276

M596p Metzner, Laercio

Programação Web I /Laercio Metzner. Indaial : UNIASSELVI, 2015.

233p. : il.

ISBN 978-85-7830-882-7

I. Programação para internet.

I. Centro Universitário Leonardo Da Vinci.

APRESENTAÇÃO

Atualmente, todas as pessoas utilizam a internet para algo. A *Web* está por toda parte ao nosso redor, sem ela não conseguíramos manter nosso ritmo de vida. Atividades como estudar, pagar contas, trabalhar, assistir vídeos e até mesmo nos locomover ficariam muito mais difíceis sem a *web*, afinal, não se trata de um mero meio de comunicação, ela auxilia em praticamente todos os aspectos de nossas vidas.

Você, como aluno de um curso de tecnologia, certamente fica instigado a respeito do que acontece quando você pressiona a tecla *Enter* após digitar o endereço de um *site* em seu *browser*. Há muita coisa envolvida neste simples ato, e este caderno de estudos tem por finalidade esclarecer a velha pergunta que ecoa na mente dos curiosos “Como se faz um *site*?”.

A *Web*, atualmente, não é apenas um ambiente de páginas de *marketing*, notícias e bate-papo. Ela é o ambiente de sistemas corporativos, educacionais, plataformas de finanças, e muitas outras coisas. Vivemos o advento dos sistemas *web*, pois “Se for *desktop* está obsoleto!”.

Há apenas uma forma de aprender a programar: programando. Vamos codificar muito ao longo de nossos estudos. Vamos praticar, acertar, errar, e saciar um pouco de nossas infinitas curiosidades. Bons estudos para nós!



Você já me conhece das outras disciplinas? Não? É calouro? Enfim, tanto para você que está chegando agora à UNIASSELVI quanto para você que já é veterano, há novidades em nosso material.

Na Educação a Distância, o livro impresso, entregue a todos os acadêmicos desde 2005, é o material base da disciplina. A partir de 2017, nossos livros estão de visual novo, com um formato mais prático, que cabe na bolsa e facilita a leitura.

O conteúdo continua na íntegra, mas a estrutura interna foi aperfeiçoada com nova diagramação no texto, aproveitando ao máximo o espaço da página, o que também contribui para diminuir a extração de árvores para produção de folhas de papel, por exemplo.

Assim, a UNIASSELVI, preocupando-se com o impacto de nossas ações sobre o ambiente, apresenta também este livro no formato digital. Assim, você, acadêmico, tem a possibilidade de estudá-lo com versatilidade nas telas do celular, tablet ou computador.

Eu mesmo, UNI, ganhei um novo layout, você me verá frequentemente e surgirei para apresentar dicas de vídeos e outras fontes de conhecimento que complementam o assunto em questão.

Todos esses ajustes foram pensados a partir de relatos que recebemos nas pesquisas institucionais sobre os materiais impressos, para que você, nossa maior prioridade, possa continuar seus estudos com um material de qualidade.

Aproveito o momento para convidá-lo para um bate-papo sobre o Exame Nacional de Desempenho de Estudantes – ENADE.

Bons estudos!

BATE SOBRE O PAPO ENADE!



Olá, acadêmico!



Você já ouviu falar sobre o ENADE?

Se ainda não ouviu falar nada sobre o ENADE, agora você receberá algumas informações sobre o tema.

Ouviu falar? Ótimo, este informativo reforçará o que você já sabe e poderá lhe trazer novidades.



Vamos lá!



Qual é o significado da expressão ENADE?

EXAME NACIONAL DE DESEMPENHO DOS ESTUDANTES

Em algum momento de sua vida acadêmica você precisará fazer a prova ENADE.



Que prova é essa?



É **obrigatória**, organizada pelo INEP – Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira.

Quem determina que esta prova é obrigatória... O **MEC – Ministério da Educação**.

O objetivo do MEC com esta prova é o de avaliar seu desempenho acadêmico assim como a qualidade do seu curso.



Fique atento! Quem não participa da prova fica impedido de se formar e não pode retirar o diploma de conclusão do curso até regularizar sua situação junto ao MEC.



Não se preocupe porque a partir de hoje nós estaremos auxiliando você nesta caminhada.

Você receberá outros informativos como este, complementando as orientações e esclarecendo suas dúvidas.



Você tem uma trilha de aprendizagem do ENADE, receberá e-mails, SMS, seu tutor e os profissionais do polo também estarão orientados.

Participará de webconferências entre outras tantas atividades para que esteja preparado para #mandar bem na prova ENADE.

Nós aqui no NEAD e também a equipe no polo estamos com você para vencermos este desafio.

Conte sempre com a gente, para juntos mandarmos bem no ENADE!



SUMÁRIO

UNIDADE 1 – CONCEITUAÇÃO, PRIMEIROS PASSOS E <i>HELLO WORLD</i>	1
TÓPICO 1 – PROTOCOLO HTTP E LINGUAGEM DE MARCAÇÃO HTML	
1 INTRODUÇÃO	3
2 UM POUCO DE HISTÓRIA	3
3 PARTIMOS DO HTML E CHEGAMOS AO HTML5	5
4 CSS: ADICIONANDO PERFUMARIAS A SUAS PÁGINAS HTML	11
5 O PROTOCOLO HTTP	15
RESUMO DO TÓPICO 1	17
AUTOATIVIDADE	18
TÓPICO 2 – SERVIDORES WEB	
1 INTRODUÇÃO	19
2 O QUE É UM SERVIDOR WEB	19
3 XAMPP, SEU SERVIDOR WEB	20
4 INSTALANDO XAMPP NO SISTEMA OPERACIONAL UBUNTU	20
RESUMO DO TÓPICO 2	36
AUTOATIVIDADE	37
TÓPICO 3 – DESTRINCHANDO O PHP, DESDE SUA HISTÓRIA ATÉ SEUS MÉTODOS	
1 INTRODUÇÃO	39
2 UM POUCO DE HISTÓRIA	39
3 DESTRINCHANDO O PHP	40
3.1 HELLO WORLD	41
3.2 VARIÁVEIS E TIPOS DE DADOS	43
3.3 DECLARAÇÃO DE CONSTANTES	45
3.4 OPERADORES LÓGICOS, MATEMÁTICOS E CONDICIONAIS	46
3.5 CONTROLANDO O FLUXO DO CÓDIGO COM <i>IF</i> , <i>ELSE</i> , <i>SWITCH</i> , <i>WHILE</i> E <i>FOR</i>	47
3.6 FUNÇÃO <i>ISSET()</i>	50
3.7 STRINGS	50
3.8 REDIRECIONANDO PARA OUTRA PÁGINA	55
3.9 ARRAYS	56
3.10 DECLARAÇÃO DE FUNÇÕES	64
3.11 SEPARANDO SEU CÓDIGO-FONTE EM MAIS DE UM ARQUIVO	66
LEITURA COMPLEMENTAR	68
RESUMO DO TÓPICO 3	73
AUTOATIVIDADE	74
UNIDADE 2 – EXPLORANDO O MYSQL, O JAVASCRIPT E CONSTRUINDO SUA PRIMEIRA APLICAÇÃO WEB	75
TÓPICO 1 – MYSQL	
1 INTRODUÇÃO	77
2 HISTÓRIA DO MYSQL	77

3 SQL: INSTRUÇÕES DDL E DML	77
4 PHPMYADMIN.....	78
5 CRIANDO OBJETOS DE BANCO.....	79
6 MANIPULANDO OS DADOS COM OS COMANDOS DML.....	83
RESUMO DO TÓPICO 1.....	92
AUTOATIVIDADE	93
 TÓPICO 2 – JAVASCRIPT	95
1 INTRODUÇÃO	95
2 UM POUCO DE HISTÓRIA.....	95
3 HELLO WORLD.....	96
4 TIPOS DE DADOS.....	97
5 DECLARAÇÃO DE VARIÁVEIS	98
6 OPERADORES LÓGICOS.....	100
7 OPERADORES ARITMÉTICOS.....	100
8 CONTROLANDO O FLUXO DE EXECUÇÃO DOS PROGRAMAS COM <i>IF</i> , <i>FOR</i> E <i>WHILE</i>	104
9 STRINGS	109
10 ARRAYS	117
11 DECLARANDO FUNÇÕES EM JAVASCRIPT	126
12 DOM, A PONTE ENTRE HTML E JAVASCRIPT	130
RESUMO DO TÓPICO 2.....	135
AUTOATIVIDADE	136
 TÓPICO 3 – DESENVOLVENDO UMA APLICAÇÃO WEB COM PHP, JAVASCRIPT E MYSQL.....	137
1 INTRODUÇÃO	137
2 O APPLICATIVO LISTA DE HÁBITOS.....	137
3 CRIAÇÃO DO BANCO DE DADOS.....	139
4 DESENVOLVIMENTO DA PARTE WEB DO APPLICATIVO	141
LEITURA COMPLEMENTAR.....	151
RESUMO DO TÓPICO 3.....	154
AUTOATIVIDADE	155
 UNIDADE 3 – A WEB NA ATUALIDADE DESACOPLAMENTO, API E JSON.....	157
 TÓPICO 1 – COESÃO E ACOPLAGEMTO	159
1 INTRODUÇÃO	159
2 COESÃO E ACOPLAGEMTO	159
3 JSON E ARQUITETURA REST	164
RESUMO DO TÓPICO 1.....	185
AUTOATIVIDADE	186
 TÓPICO 2 – DESIGN RESPONSIVO	187
1 INTRODUÇÃO	187
2 DESIGN RESPONSIVO COM BOOTSTRAP	187
RESUMO DO TÓPICO 2.....	196
AUTOATIVIDADE	197
 TÓPICO 3 – SINGLE PAGE APPLICATION	199
1 INTRODUÇÃO	199
2 TURBINANDO SUAS PÁGINAS WEB COM ANGULARJS	199

3 CONHECENDO SINGLE PAGE APPLICATION E IMPLEMENTANDO UM TESTE PARA NOSSA API.....	203
4 BOOTSTRAP, SINGLE PAGE APPLICATION, ANGULARJS E ACESSO A API NA NOVA VERSÃO DA APLICAÇÃO WEB LISTA DE HÁBITOS	216
LEITURA COMPLEMENTAR.....	227
RESUMO DO TÓPICO 3.....	230
AUTOATIVIDADE	231
REFERÊNCIAS	233

UNIDADE 1

CONCEITUAÇÃO, PRIMEIROS PASSOS E *HELLO WORLD*

OBJETIVOS DE APRENDIZAGEM

Após estudar esta unidade, você será capaz de:

- desenvolver páginas Web com HTML, CSS e PHP;
- instalar o XAMPP;
- descrever a função dos principais métodos do HTTP.

PLANO DE ESTUDOS

Esta unidade está dividida em três tópicos, sendo que, ao final de cada um deles, você encontrará atividades que o(a) auxiliarão a fixar os conhecimentos desenvolvidos.

TÓPICO 1 – PROTOCOLO HTTP E LINGUAGEM DE MARCAÇÃO HTML

TÓPICO 2 – SERVIDORES WEB

TÓPICO 3 – DESTRINCHANDO O PHP, DESDE SUA HISTÓRIA ATÉ SEUS MÉTODOS

PROTOCOLO HTTP E LINGUAGEM DE MARCAÇÃO HTML

1 INTRODUÇÃO

Caro(a) Acadêmico! Neste tópico, você poderá entender mais sobre os conceitos básicos da *Web* e de como ela foi desenvolvida. Visando colaborar ainda mais com sua formação acadêmica, este tópico abre uma visão rápida sobre as principais concepções e de como desenvolvê-las, pois você será levado a pensar em como a *Web* permitiu tal evolução, desde o aparecimento do HTML até o HTML5, que é o mais usado hoje em dia. Observando sempre, que a linguagem presente desde o início do HTML, está ainda presente no HTML5.

Ainda neste tópico, você verá também o desenvolvimento do CSS, e a forma correta de utilizá-lo para os desenvolvimentos de uma página *Web*. Você já deve saber que é uma linguagem utilizada para dar estilo e formatação a uma página *Web*. Veremos mais nas páginas a seguir. Bons estudos!

2 UM POUCO DE HISTÓRIA

O conceito de *Web* foi concebido em 1980, por Tim Berners-Lee, e não se tratava da internet como a conhecemos hoje. Tratava-se de um projeto denominado *ENQUIRE*, o qual era muito embrionário, mas o grande passo para o início da internet foi dado, pois neste projeto surgiu o primeiro processador de hipertexto.

Hipertexto é um conceito muito simples no qual, o usuário que está lendo o conteúdo de uma página tem a possibilidade de navegar por diferentes páginas através de *hiperlinks*. A navegação *Web* de hoje em dia baseia-se nisto, você faz assim no cotidiano certamente sem perceber. Por exemplo: você acessa o Youtube e clica sobre o título de um vídeo que possui um *hyperlink* para o vídeo a que você deseja assistir.

Em 1989, Tim Berners-Lee dava continuidade ao seu projeto quando conseguiu, juntamente com Robert Cailliau, estabelecer a primeira conexão bem-sucedida entre um cliente e um servidor HTTP. O protocolo HTTP (*Hypertext Transfer Protocol*) é o protocolo utilizado na transferência de arquivos de hipertexto.



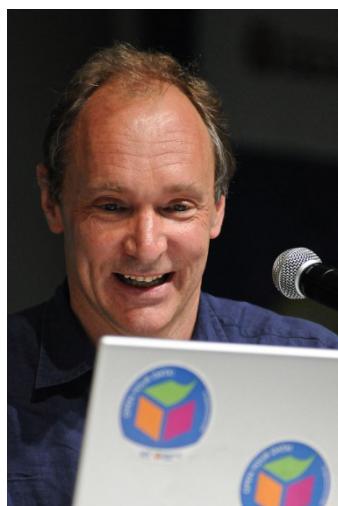
Mais adiante você aprenderá mais sobre HTTP. E na Unidade 3, você terá exemplos de como utilizar o protocolo HTTP na prática.

Em 1990, uma linguagem de marcação para a criação de arquivos de hipertexto foi criada, o HTML (*Hypertext Markup Language*), que está em constante evolução até os dias atuais. Tal evolução deu origem a diversas versões da linguagem, como HTML, HTML 2.0, HTML 3.0, HTML 3.2, HTML 4.0, XHTML, e recentemente o HTML5. Desde 1994, há um órgão responsável por determinar as características para as versões do HTML e coordenar o processo evolutivo da linguagem, o W3C (*World Wide Web Consortium*).

A evolução da *Web* ainda conta com a presença de seu criador, Tim Berners-Lee, conforme Silva (2011, p. 21):

Atualmente Tim é diretor do World Wide Web Consortium (W3C), pesquisador sênior do Laboratório da Ciência da Computação e Inteligência Artificial (CSAIL) do Instituto de Tecnologia de Massachusetts (MIT) e professor de Ciência da Computação na Universidade de Southampton, na Inglaterra.

FIGURA 1 - TIM BERNERS-LEE



FONTE: Disponível em: <http://pt.wikipedia.org/wiki/Tim_Berners-Lee#mediaviewer/File:Tim_Berners-Lee_CP.jpg>. Acesso em: 23 dez. 2014.

3 PARTIMOS DO HTML E CHEGAMOS AO HTML5

Após várias versões de HTML, o que temos em evidência hoje é o HTML5. Mas afinal, o que o HTML5 tem a nos oferecer, e por que ele é considerado uma evolução do HTML?



Esta unidade pretende introduzi-lo ao HTML5. Caso você não tenha conhecimento básico de HTML, recomendamos que você revise o Caderno de Estudos da disciplina de Introdução ao Desenvolvimento de Sistemas Web.

A grande diferença está no fato de que anteriormente o HTML era o ‘faz tudo’. Era no HTML que se dizia se um texto deveria ser pintado de determinada cor, qual o tamanho da fonte, qual a cor de fundo para determinada área da página etc. No entanto, o HTML5 vem com uma proposta mais enxuta. Ele se propõe a cuidar somente da semântica da página. A formatação, como já vinha ocorrendo na versão 4 do HTML, foi delegada completamente para o CSS (*Cascade Style Sheet*). É complicado perceber de início esta diferença, pois em muitos casos, os navegadores acabam interpretando HTML e HTML5 quando se encontram juntos em uma página.



Semântica: No contexto do HTML, a semântica compreende os cuidados necessários para inserir os recursos das páginas (textos, imagens, vídeos), dentro das tags corretas. Do ponto de vista da semântica é errado inserir um texto que representa um título entre uma tag `<p></p>`, pois esta tag representa um parágrafo.



Você aprenderá mais sobre CSS ainda neste tópico.

Além de ganhar uma responsabilidade mais específica, ou seja, cuidar somente da semântica, o HTML5 veio com novos atributos para controle de multimídia (áudio e vídeo, por exemplo) e novos controles para formulários. Iremos estudá-los mais adiante.

Caro(a) acadêmico(a)! Neste momento, você deve querer começar seus desenvolvimentos em HTML5. Reza uma antiga lenda, que sempre que conhecemos uma nova tecnologia, o primeiro programa que devemos implementar é o *Hello World*, senão ficamos amaldiçoados e não alcançamos sucesso em tal tecnologia. Então, é hora de erguer as mangas fazer nosso *Hello World* em HTML5 para evitar a maldição!

Em seu computador, abra o editor de texto e crie um arquivo chamado ‘helloworld.html’ (sem aspas). Em seguida, abra o arquivo e insira o conteúdo da listagem 1:

LISTAGEM 1 - CONTEÚDO DO ARQUIVO helloworld.html.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello World</title>
  </head>
  <body>
    <p>
      Hello World
    </p>
  </body>
</html>
```

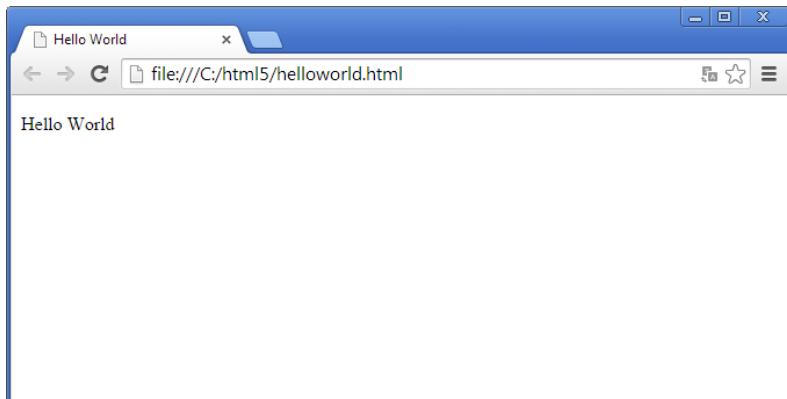


EDITOR DE TEXTO

Você pode usar o editor de texto de sua preferência. No caso do sistema operacional Windows, o padrão é o Notepad. No caso do sistema operacional Ubuntu, o padrão o Gedit. Há editores de textos com mais funcionalidades disponíveis no mercado (pagos e gratuitos, são inúmeras opções), fique à vontade para escolher o que for mais conveniente. Exemplo: Notepad++, Sublime, TextPad etc.

Após digitar todo o código-fonte da listagem 1 no seu arquivo helloworld.html, salve-o, feche o editor de texto e abra novamente o arquivo. Ele deverá abrir no seu *browser* (navegador de internet), e deve estar parecido com o da imagem a seguir:

FIGURA 2 - PÁGINA WEB HELLO WORLD EXIBIDA NO BROWSER GOOGLE CHROME



FONTE: O autor

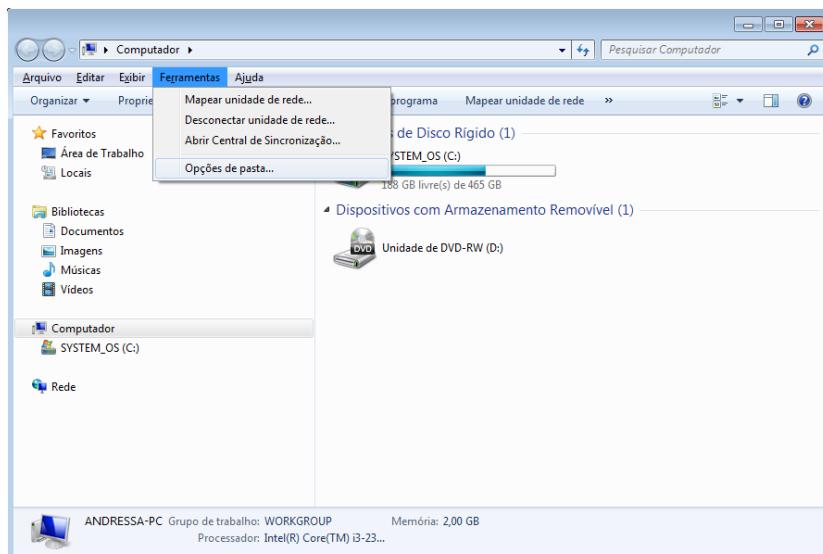


SEU ARQUIVO HELLOWORLD.HTML NÃO ESTÁ ABRINDO NO BROWSER?

Se você abriu o arquivo e em vez de aparecer o browser o arquivo foi aberto novamente no editor de texto, verifique se o nome do arquivo está correto, pois é possível que o arquivo esteja com o nome 'helloworld.html.txt' (sem aspas). O nome correto do arquivo é 'helloworld.html' que representa o nome do arquivo 'helloworld' e sua extensão 'html' separados por um ponto. Caso isto esteja acontecendo com você em um computador com o sistema operacional Windows, acesse as configurações do Windows Explorer e altere a configuração "Ocultar as extensões dos tipos de arquivo conhecidos", conforme a seguir:

- No Windows Explorer, pressione a tecla *alt*, acesse o menu Ferramentas e clique sobre Opções de Pasta:

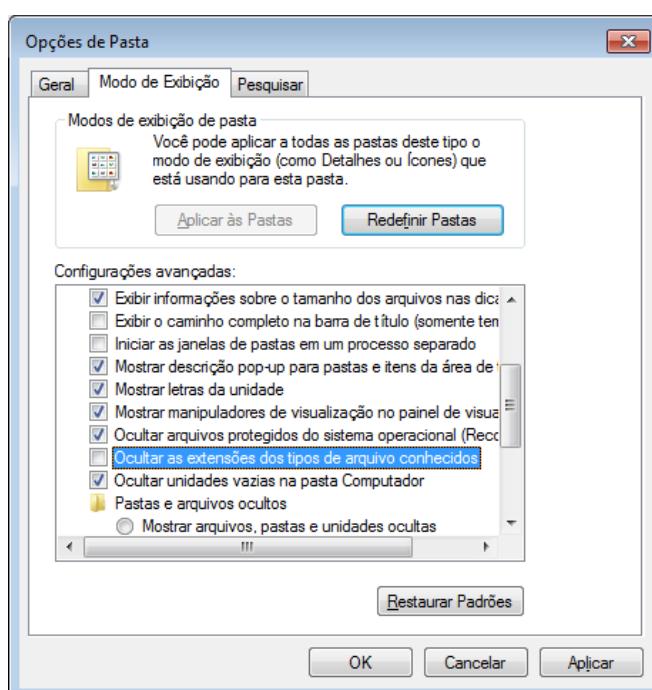
FIGURA 3 - MENU OPÇÕES DE PASTA DO WINDOWS EXPLORER



FONTE: O autor

Na janela de Opções de Pasta, acesse a guia Modo de Exibição, encontre a configuração “Ocultar as extensões dos tipos de arquivo conhecidos”, desmarque-a e clique em OK:

FIGURA 4 - JANELA OPÇÕES DE PASTA COM A CONFIGURAÇÃO OCULTAR AS EXTENSÕES DOS TIPOS DE ARQUIVO CONHECIDOS DESMARCADA



FONTE: O autor

Agora, vamos analisar o que cada linha do nosso *Hello World* representa:

- <!DOCTYPE html>: Declaração de que é um documento de hipertexto escrito em HTML5.
- <html lang="en"></html>: A tag <html> é sempre a tag raiz (a mais externa) de uma página. O atributo *lang* indica o idioma da página, no caso inglês. Tudo que estiver entre a tag <html> e seu fechamento </html> faz parte da página.
- <head> e </head>: A tag <head> continua sendo a mesma do HTML, ela traz informações sobre página. Neste exemplo estamos incluindo somente o *charset* e o título, mas ela poderia trazer *metatags* com informações adicionais como autor, descrição, palavras-chave etc.
- <meta charset="utf-8">: Indica que a codificação de caracteres usada na página é utf-8. O utf-8 é um padrão universalmente conhecido e bem recomendado. O *charset* utf-8 possibilita a utilização de caracteres acentuados (á, é ê, ã etc.) em nossas páginas HTML5.
- <title>Hello World</title>: Indica o título da página. Normalmente, os *browsers* exibem esta informação na *tab* (ou guia) que contém a página aberta.
- <body> e </body>: Tudo o que estiver entre a tag <body> e seu fechamento </body> será renderizado pelo *browser*.
- <p>Hello World</p>: Um simples parágrafo com o texto *Hello World*.

Como podemos perceber, o HTML5 possui uma estrutura muito semelhante a do HTML, você reaproveita boa parte do seu conhecimento de HTML para escrever páginas em HTML5. A tabela a seguir apresenta os principais elementos do HTML5, não se preocupe se você não os entender imediatamente, iremos abordar os principais elementos ao longo do caderno.

TABELA 1 - TAGS HTML5 PARA ESTUDOS DESTA DISCIPLINA

Tag	Descrição
<html>	Todo conteúdo HTML, exceto o <i>DOCTYPE</i> ou comentários, deve estar entre esta tag e seu fechamento.
<head>	Cabeçalho da página. Abriga informações como título da página, <i>charset</i> , palavras-chave, entre outros.
<body>	Corpo da página. Tudo que estiver entre esta tag e seu fechamento será renderizado pelo <i>browser</i> .
<!--...-->	Comentário, o conteúdo que estiver entre a abertura '<--' e o fechamento '-->' não será renderizado pelo <i>browser</i> .
<a>	<i>Hyperlink</i> .
<abbr>	Abreviação.
<address>	Endereço do autor do documento.
<article>	Artigo.

<audio>	Streaming de áudio.
<base>	URL base para todos os <i>links</i> da página.
<bdo>	Direção na qual o texto será mostrado.
<blockquote>	Citação longa.
 	Quebra de linha.
<button>	Botão (ação do usuário).
<canvas>	Cria na página uma área criação de gráficos em uma página Web.
<caption>	Legendas para tabelas.
<cite>	Citação curta.
<code>	Trecho de código.
<col>	Coluna em tabela.
<colgroup>	Grupo de colunas em tabelas.
<datalist>	Autocomplete para formulários.
<details>	Informações adicionais que podem ser ocultadas ou exibidas conforme preferência do usurário.
<div>	Seção no documento, agindo como um agrupador de elementos HTML.
<fieldset>	Grupo de elementos relacionados ao formulário.
<footer>	Especifica um rodapé para a seção de uma página.
<form>	Formulário.
<h1> .. <h6>	Cabeçalho de nível 1 até 6.
<header>	Cabeçalho para seção de página.
<hgroup>	Agrupador para elementos de título do nível h1 ao h6.
<hr>	Linha horizontal.
<iframe>	Sub janela no documento.
	Imagen.
<input>	Campo de texto.
<label>	Rótulo para elementos em um formulário.
<legend>	Título para um <i>fieldset</i> .
	Item de uma lista.
<mark>	Texto destacado para fins de referência.
<map>	Mapa de imagem.
<menu>	Menu.
<meta>	Meta informações para o cabeçalho da página, por exemplo, autor, palavras-chave etc.
<nav>	<i>Links</i> de navegação.
<object>	Objeto. Ex.: vídeo do YouTube.
	Lista ordenada.
<p>	Parágrafo.
<param>	Especifica um parâmetro para um objeto incorporado no documento HTML.
<progress>	Barra de progresso para tarefas.
<q>	Breve citação.
<script>	<i>Scripts</i> em um documento HTML, geralmente implementados na linguagem JavaScript.
<section>	Seção dentro de artigo.
<select>	Lista de itens selecionáveis.
<small>	Texto em tamanho pequeno.
<source>	Origem de recursos multimídia em uma página.

	Elementos in-line. Muitas vezes usado para mostrar o resultado de ações do usuário.
	Texto de maior importância.
<style>	Declaração de folhas de estilo.
<sub>	Texto subscrito.
<sup>	Texto sobreescrito.
<table>	Tabela.
<tbody>	Corpo da tabela.
<td>	Célula da tabela.
<textarea>	Área para entrada de texto multilinha.
<tfoot>	Rodapé da tabela.
<th>	Célula de cabeçalho da tabela.
<thead>	Cabeçalho de uma tabela.
<time>	Data e hora.
<title>	Título.
<tr>	Linha de uma tabela.
	Lista não ordenada.
<video>	Streaming de vídeo.

FONTE: O autor



Note que vários elementos do HTML5 existem no HTML. Isto se deve à coordenação da evolução da linguagem pelo W3C (World Wide Web Consortium). Além do mais, nem todas as tags do HTML5 estão na tabela acima, se você quiser conferir todas as tags do HTML5 a documentação da Mozilla é uma excelente fonte de informação: <https://developer.mozilla.org/en/docs/Web/Guide/HTML/HTML5/HTML5_element_list>.

Então, gostou de se envolver com HTML e HTML5? Tem domínio básico de inglês? Ou tem um dicionário de inglês? Então, você já está apto a dominar o HTML! Acesse <<http://www.w3schools.com/>> e comece a aprofundar seus conhecimentos!

4 CSS: ADICIONANDO PERFUMARIAS A SUAS PÁGINAS HTML

Já pensou se as páginas Web não tivessem cores, elementos gráficos e fontes diferentes? Certamente a Web seria um espaço monótono. Para criar uma página Web viva é necessário dar uma identidade visual a ela. Já imaginou tudo o que poderíamos fazer para embelezar nosso *Hello World*?



Você já conhece o CSS do Caderno de Estudos de Introdução ao Desenvolvimento de Sistemas Web, por este motivo, vamos apenas recapitular.

Quando falamos em CSS (*Cascade Style Sheets*) estamos falando da linguagem utilizada para dar estilo e formatação a uma página Web. Mas você deve pensar, mesmo antigamente, as páginas Web já tinham cores e eram atrativas. Lembra que falamos que no início o HTML era o ‘faz tudo’? Vamos ver como ficaria a versão retrô do nosso *Hello World*:

LISTAGEM 2 - PÁGINA *HELLO WORLD* ESTILIZADA UTILIZANDO OS ATRIBUTOS HTML.

```
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello World</title>
  </head>
  <body bgcolor="yellow">
    <p align="center">
      Hello World
    </p>
  </body>
</html>
```

FIGURA 5 - PÁGINA WEB *HELLO WORLD* ESTILIZADA UTILIZANDO OS ATRIBUTOS DE FORMATAÇÃO DO HTML



FONTE: O autor

Note que nossa página retrô não conta mais com o *DOCTYPE* no topo, pois não é HTML5. Mas se você não o retirou, não há problema, pois os *browsers* interpretam HTML misturado ao HTML5 sem problemas (mas é considerado uma má prática).

Veja que há um atributo na *tag body*, o *bgcolor* (*background color*), responsável por colorir o corpo da página. Há também um atributo no parágrafo, o *align*, responsável por centralizar o texto. Estes atributos para formatação no HTML estão obsoletos desde o HTML 4.0, quando o CSS entrou em cena.

A principal função do CSS é separar o código-fonte referente à formatação da página do código-fonte que contém o conteúdo (e trata a semântica) da página, e assim aumentar as possibilidades de reaproveitamento do mesmo. Então, vamos refatorar nosso *Hello World* para que fique coerente com as boas práticas de programação utilizadas atualmente:

1. No mesmo diretório em que se encontra seu arquivo *helloworld.html*, crie um arquivo chamado *styles.css* com o seguinte conteúdo:

LISTAGEM 3 - CONTEÚDO DO ARQUIVO *styles.css*

```
body {
    background-color: yellow;
}
p {
    text-align: center;
}
```

2. Sobrescreva o conteúdo do seu arquivo *helloworld.html* com o conteúdo a seguir:

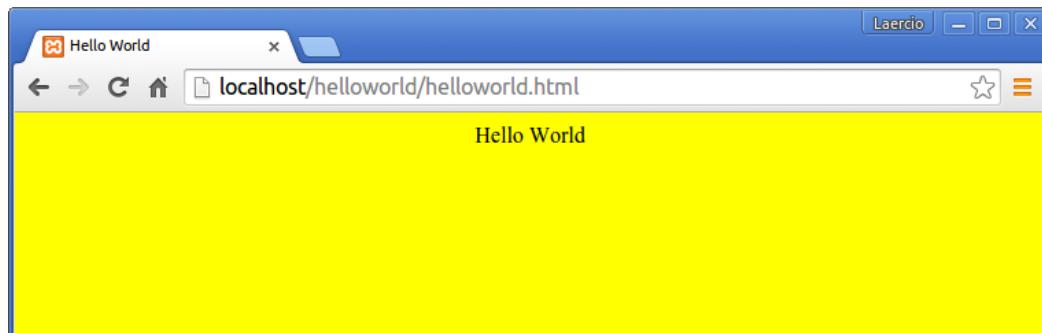
LISTAGEM 4 - *helloworld.html*

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Hello World</title>
        <link rel="stylesheet" type="text/css"
            href="styles.css">
    </head>
    <body>
        <p>
            Hello World
        </p>
    </body>
</html>
```

```
</body>  
</html>
```

3. Abra no navegador seu arquivo helloworld.html, deve estar conforme a figura a seguir:

FIGURA 6 - PÁGINA WEB HELLO WORLD ESTILIZADA UTILIZANDO CSS



FONTE: O autor

Talvez você esteja indignado, pois desta forma foi necessário escrever mais código-fonte. Porém, como é possível perceber, o código-fonte referente ao estilo e formatação (arquivo .css) fica completamente isolado da página (arquivo .html). O arquivo styles.css é referenciado dentro do cabeçalho da página helloworld.html, através da tag `<link rel="stylesheet" type="text/css" href="styles.css">`. O mesmo arquivo CSS pode ser referenciado em diversas páginas diferentes. Isto potencializa muito a reutilização de estilos.



MAS POR QUE DECLARAR O CSS SEPARADO?

Declarar o CSS dentro do arquivo da página HTML é considerado uma má prática, pois prejudica a principal vantagem do CSS no desenvolvimento Web, o reaproveitamento. Mesmo que seja possível, através da tag `style` e dos estilos *in-line*, é considerado poluição de código, pois mistura o código referente à formatação com o conteúdo da página, fazendo com que a utilização de CSS perca o sentido. Caso você tenha ficado em dúvida ao ouvir falar da tag `style` ou de estilos *in-line*, revise o conteúdo do Caderno de Estudos da disciplina de Introdução ao Desenvolvimento de Sistemas Web.

5 O PROTOCOLO HTTP

Caro(a) acadêmico(a)! Neste ponto, você já tem conhecimento de como as páginas *Web* são escritas, pois já conhece alguns elementos do HTML. Vamos ver então, o que acontece por baixo dos panos quando você digita a URL de um *site* no seu *browser* e pressiona a tecla *Enter* para visualizar a página *Web* que você deseja.

Quando você tem um computador conectado à internet, abre o seu *browser* e digita o *site* que você deseja visualizar (por exemplo, <http://www.grupouniasselvi.com.br/>) seu *browser* irá efetuar uma requisição *get* através do protocolo HTTP e irá obter a página *Web* como resposta, conforme a figura abaixo.

FIGURA 7 - PÁGINA WEB OBTIDA COMO RESPOSTA À URL <http://www.grupouniasselvi.com.br/>



FONTE: O autor



VÁRIOS CONCEITOS NOVOS? VAMOS RELEMBRAR ALGUMAS COISAS!

Você lembra que HTTP (*Hypertext Transfer Protocol*) é o protocolo para transferência de arquivos de hipertexto. O *get*, é um método do HTTP utilizado para buscar um arquivo em um servidor *Web*.

URL (*Unified Resource Locator*) é um endereço na *Web*. O exemplo acima ilustra o acesso à URL <http://www.grupouniasselvi.com.br/>. O *browser* automaticamente precede a URL com a descrição do protocolo a ser utilizado, o HTTP. A URL completa da requisição fica <http://www.grupouniasselvi.com.br>.



PERCEBEU O INDEX.ASPX NA URL?

É um comportamento comum a vários servidores *Web* que quando você faz uma requisição informando uma URL sem especificar o nome do arquivo desejado, ele retorne o arquivo *index*. Foi o que aconteceu no nosso exemplo, requisitamos a URL <<http://www.grupouniasselvi.com.br/>> e recebemos como resposta o arquivo *index.aspx*. Neste caso, a extensão do arquivo (*.aspx*) nos indica que a página foi implementada na plataforma da Microsoft ASP.NET.

Além do método *get* utilizado pelo *browser*, o HTTP possui outros métodos. A seguir, apresentamos os principais deles:

- ***GET***: como você já sabe, busca um arquivo em um servidor *Web*, pode receber parâmetros e os parâmetros ficam visíveis na URL.
- ***POST***: Busca um arquivo em um servidor *Web*, pode receber parâmetros, porém os parâmetros não ficam visíveis na URL, eles trafegam ocultos no corpo da requisição.
- ***PUT***: Armazena o conteúdo da requisição na URL solicitada, ou seja, pede para adicionar/alterar um arquivo no servidor *Web*.
- ***DELETE***: Deleta o arquivo da URL solicitada, ou seja, pede para excluir um arquivo no servidor *Web*.



Esta unidade forneceu a parte teórica do HTTP. Não se preocupe em saber como funcionam os métodos, apenas certifique-se de que entendeu o que cada um dos métodos acima faz. Até aqui, o conceito é o que realmente importa. Na Unidade 3, iremos apresentar o conceito de API REST e você irá utilizar os diferentes métodos do HTTP na prática.

RESUMO DO TÓPICO 1

Neste tópico, você aprendeu que:

- A *Web* tem suas origens na década de 90, e não para de evoluir até os dias atuais.
- Tim Berners-Lee, criador da *Web*, ainda hoje participa do seu processo evolutivo.
- Atualmente, a evolução da *Web* é coordenada por consórcio chamado W3C (*World Wide Web Consortium*).
- A versão do HTML em evidência hoje em dia é o HTML5.
- Você teve um *overview* das *tags* HTML.
- Você viu que não se deve usar o HTML para estilizar suas páginas, em vez disto, utilizar o CSS.
- Viu os principais métodos do protocolo HTTP (*Hypertext Transfer Protocol*).

AUTOATIVIDADE



Explore as tags HTML.

- 1 Implemente uma página *Web* para exibir uma lista de compras. Utilize uma lista não ordenada.

Arquivo listacompras.html:

- 2 Implemente uma página *Web* para exibir uma lista de atividades a cumprir (*to do list*). Utilize uma lista ordenada.

Arquivo todolist.html:

- 3 Em seu *Hello World*, implemente um *link* denominado ‘Lista de Compras’, direcionando para a página que você implementou na questão 1.

Arquivo helloworld.html:

- 4 Utilizando CSS, faça com que a cor do texto do *link* criado na questão 3 seja verde, e que o mesmo apareça alinhado à direita (crie um arquivo de CSS separado da página).

Arquivo styles.css

1 INTRODUÇÃO

Caro(a) acadêmico(a)! Você já deve saber escrever e formatar as suas páginas *Web*, utilizando HTML e CSS. Você também já conhece os conceitos básicos do Protocolo HTTP, o qual é utilizado pelo *browser* para obter páginas *Web* na internet. Mas quem responde às requisições do *browser*? Você conhecerá neste tópico. Iremos abordar as principais funções de um servidor *Web*, seus componentes e a forma que ele funciona. Tudo isto utilizando uma plataforma muito bem conceituada no mercado, o XAMPP.

2 O QUE É UM SERVIDOR *WEB*

No tópico anterior, enquanto estudava o protocolo HTTP, viu que durante seu processo natural de navegar na internet, você digita uma URL no seu *browser* e ele faz uma requisição *get* a um servidor *Web*. Mas o que é exatamente um servidor *Web* e qual é seu papel?

O servidor *Web* é um programa cuja função é responder a requisições. Em sua forma mais básica, as requisições são pedidas para o servidor *Web* devolver um determinado arquivo. Por exemplo, na URL <www.apachefriends.org/pt_br/index.html/> o texto ‘www.apachefriends.org/’ representa o caminho para alcançar o servidor *Web* na internet, enquanto o texto ‘pt_br/index.html’ representa o caminho para alcançar o arquivo index.html dentro do servidor *Web*.

O primeiro *software* de servidor *Web* foi concebido em 1990 e como você talvez já deve pensar, contou com a participação de Tim Berners-Lee na equipe do projeto. Este *software* foi denominado CERN httpd.

3 XAMPP, SEU SERVIDOR *WEB*

Vamos agora conhecer o servidor *Web* para esta disciplina, o XAMPP. Na verdade, o XAMPP é mais do que um servidor *Web*. Ele é constituído principalmente de um servidor *Web*, um banco de dados e um servidor FTP (*File Transfer Protocol*).

O termo XAMPP é um acrônimo, o significado diz praticamente tudo sobre ele:

- X: A letra X tem por objetivo apresentar a característica de que ele é genérico em relação ao sistema operacional no qual deve ser instalado. Há instalações para Linux, Windows e Mac OS.
- A: Apache, o servidor *Web* do XAMPP.
- M: MySQL, o banco de dados.
- P: PHP, a linguagem.
- P: Perl, mais uma linguagem.

Conseguiu perceber como o XAMPP é poderoso? Ele possui um servidor *Web*, um Banco de dados, duas linguagens de programação, um servidor FTP e ainda por cima roda em diversos sistemas operacionais. Querendo ter um XAMPP rodando em seu computador? Os próximos tópicos guiarão você.



Nesta disciplina não iremos utilizar o FTP nem o Perl. Mas é importante você saber que eles existem no XAMPP.

4 INSTALANDO XAMPP NO SISTEMA OPERACIONAL UBUNTU

O Ubuntu é um sistema operacional muito encontrado em empresas de desenvolvimento *Web*. Devido ao fato de ser um sistema operacional maduro, de código-fonte aberto e gratuito (opcionalmente você pode fazer doações à empresa mantenedora, a Canonical, após efetuar o *download*), é comumente utilizado para ambientes de desenvolvimento.

Geralmente, instalar programas no Linux é uma tarefa que exige bastante conhecimento e capacidade de pesquisar, mas isto não acontece com o Ubuntu. Ele é uma distribuição de Linux que cresceu com o objetivo de ser amigável ao usuário.



Grande parte dos sistemas Web roda em ambiente Linux. Se você não conhece o Linux, conhecer o Ubuntu é um bom início. É muito interessante conhecê-lo. Acesse: <<http://www.ubuntu.com/>> e obtenha informações.

1. Acesse o site <<https://www.apachefriends.org/>> e acesse o menu *download*:

FIGURA 8 - HOME PAGE DA APACHE FRIENDS, ONDE É POSSÍVEL FAZER DOWNLOAD DO XAMPP

O que é o XAMPP?

XAMPP é o ambiente de desenvolvimento PHP mais popular

XAMPP é completamente gratuito e fácil de instalar a distribuição Apache contendo MySQL, PHP e Perl. O pacote de código aberto do XAMPP foi criada para ser extremamente fácil de instalar e de usar.

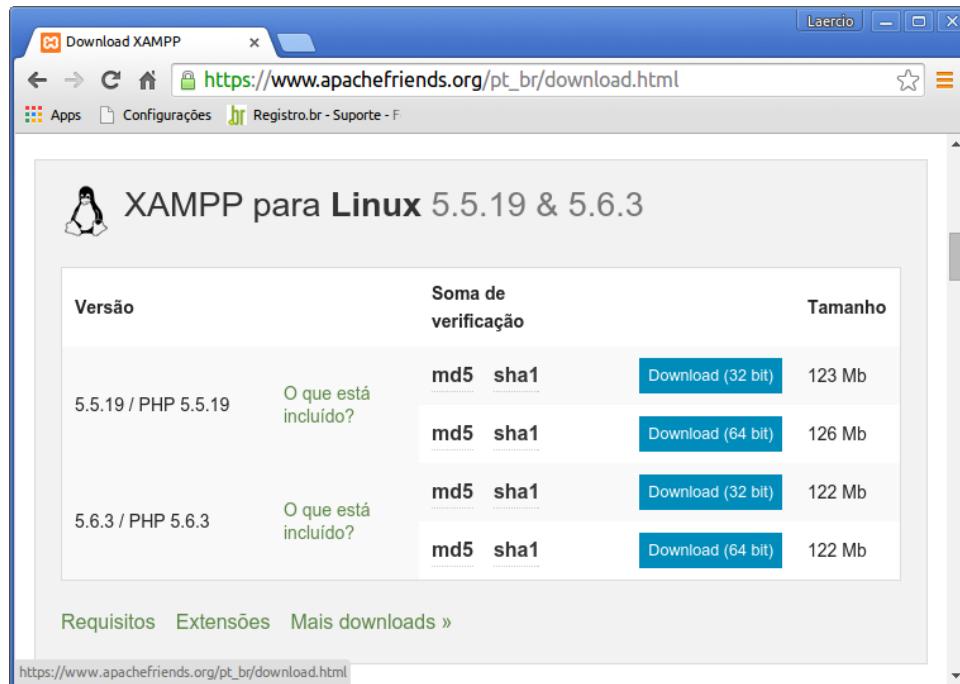
Por que usar o XAMPP?

https://www.apachefriends.org/pt_br/download.html

FONTE: O autor

2. Identifique a versão mais recente do XAMPP para Linux disponível para a arquitetura de sua máquina (32 ou 64 bits) e clique em *download* para baixar o instalador:

FIGURA 9 - ÁREA DE DOWNLOAD PARA LINUX DA PÁGINA DE DOWNLOADS DA APACHE FRIENDS



FONTE: O autor

3. Uma nova aba no seu navegador irá abrir e o *download* irá iniciar:

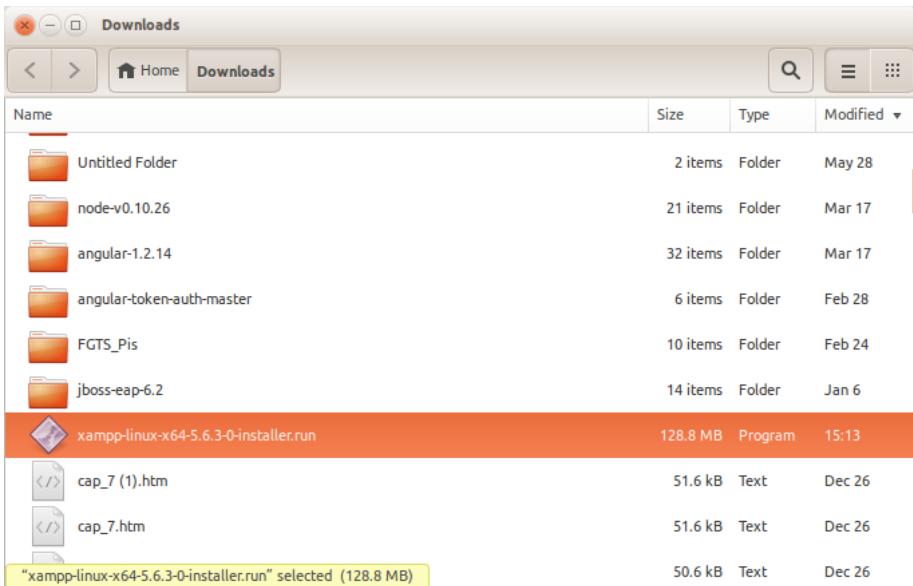
FIGURA 10 - DOWNLOAD DO XAMPP EM ANDAMENTO



FONTE: O autor

4. Quando o *download* estiver concluído, encontre o instalador na sua pasta de *downloads* e clique sobre ele com o botão direito e depois sobre propriedades:

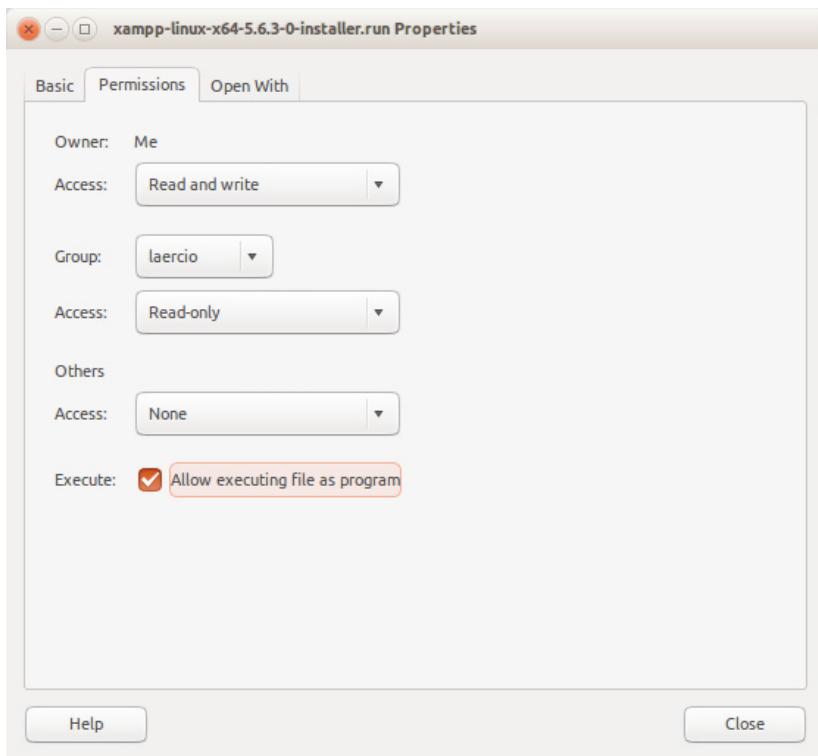
FIGURA 11 - INSTALADOR DO XAMPP NA PASTA DE DOWNLOADS



FONTE: O autor

5. Na janela propriedades, acesse a guia Permissões e marque a opção Executar arquivo como programa:

FIGURA 12 - PROPRIEDADES DO INSTALADOR



FONTE: O autor

6. Agora feche todas as janelas e abra um terminal. Nele, execute o comando ‘sudo nautilus’ e digite sua senha de administrador, conforme a seguir:



O comando sudo nautilus irá iniciar uma seção do *file explorer* com privilégios de acesso de administrador (*Root user*).

FIGURA 13 - COMANDO SUDO NAUTILUS

A screenshot of a terminal window titled "laercio@laercio-laptop: ~". The window shows the command "sudo nautilus" being run, followed by a password prompt "[sudo] password for laercio:". The terminal then displays several error messages from Nautilus, including warnings about failed registration and errors related to user sharing.

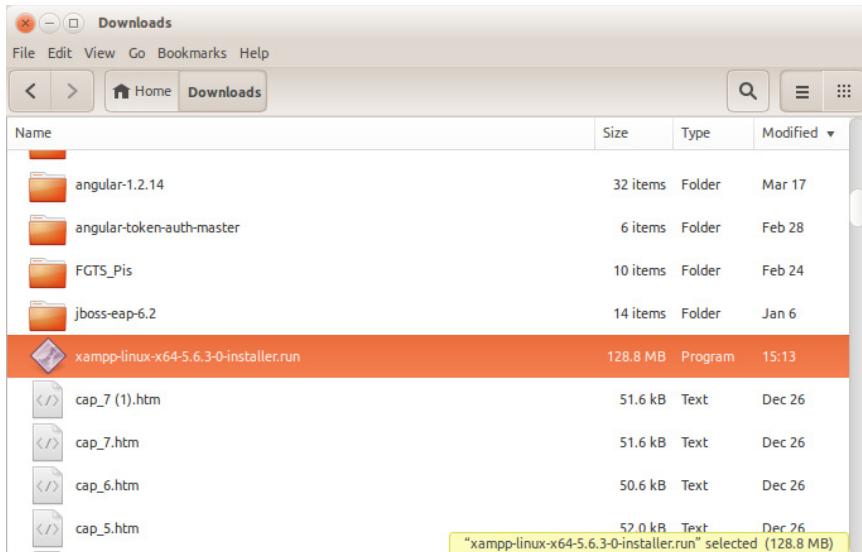
```
laercio@laercio-laptop:~$ sudo nautilus
[sudo] password for laercio:

(nautilus:3877): Gtk-WARNING **: Failed to register client: GDBus.Error:org.free
desktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not pro
vided by any .service files
Nautilus-Share-Message: Called "net usershare info" but it failed: 'net usershar
e' returned error 255: net usershare: cannot open usershare directory /var/lib/s
amba/usershares. Error No such file or directory
Please ask your system administrator to enable user sharing.
```

FONTE: O autor

7. No *file explorer* que foi aberto após a execução do comando, encontre o arquivo de instalação do XAMPP (neste caso, a pasta de *downloads*), selecione-o e tecle *Enter*:

FIGURA 14 - INSTALADOR DO XAMPP NO FILE EXPLORER COM PERMISSÃO DE ADMINISTRADOR



FONTE: O autor

8. O instalador será iniciado, clique em *Next*:

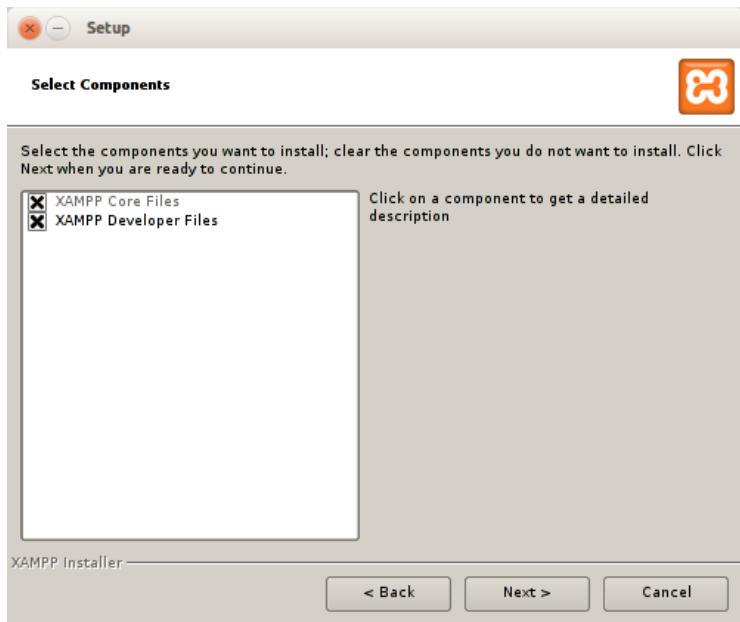
FIGURA 15 - INSTALADOR DO XAMPP NA PRIMEIRA ETAPA



FONTE: O autor

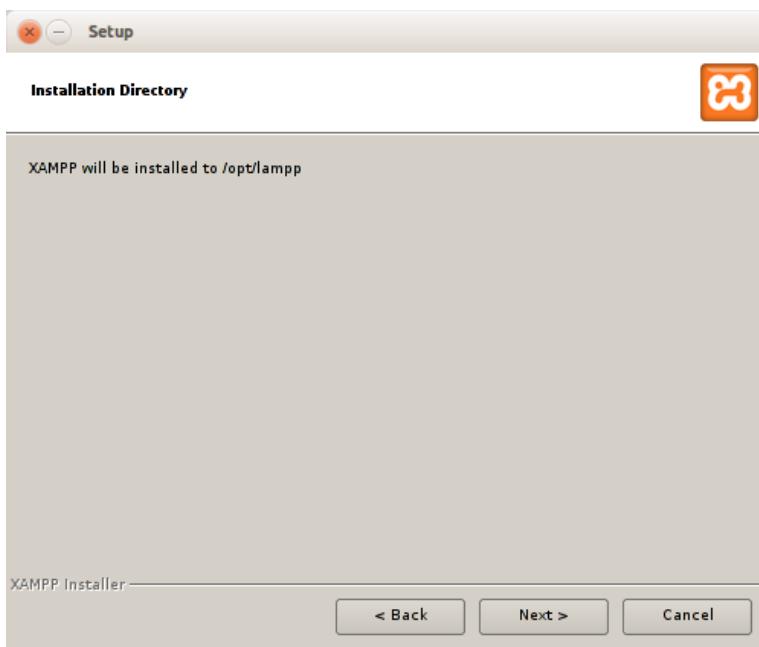
9. Na página seguinte, mantenha as opções marcadas e clique em *Next*:

FIGURA 16 - INSTALADOR DO XAMPP NA SEGUNDA ETAPA



FONTE: O autor

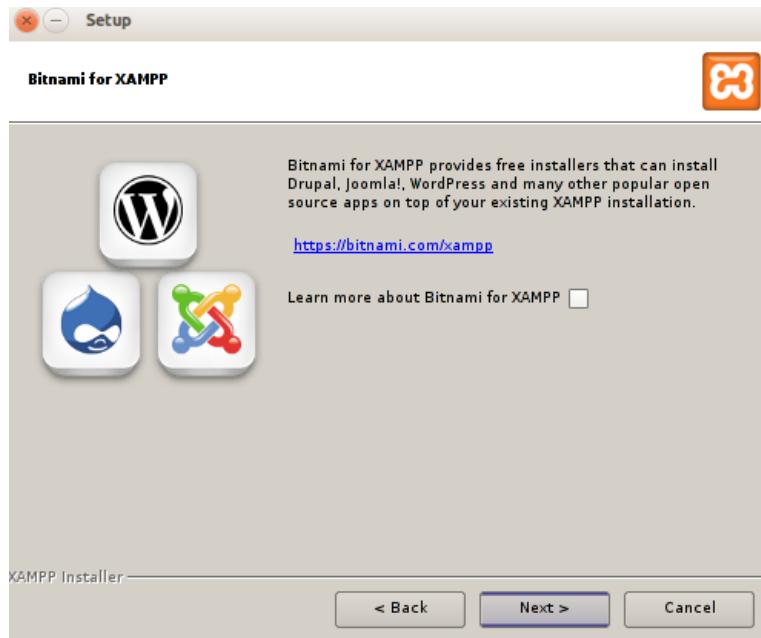
FIGURA 17 - INSTALADOR DO XAMPP NA TERCEIRA ETAPA



FONTE: O autor

11. Para exemplos deste Caderno de Estudos não será necessário instalar o Bitnami, mas caso você tenha interesse de aprender sobre WordPress, Moodle e/ou outros softwares *open source* em PHP, mantenha a opção “Aprender mais sobre Bitnami” marcada. Para fins deste tutorial, vamos desmarcá-la e clicar em *Next*:

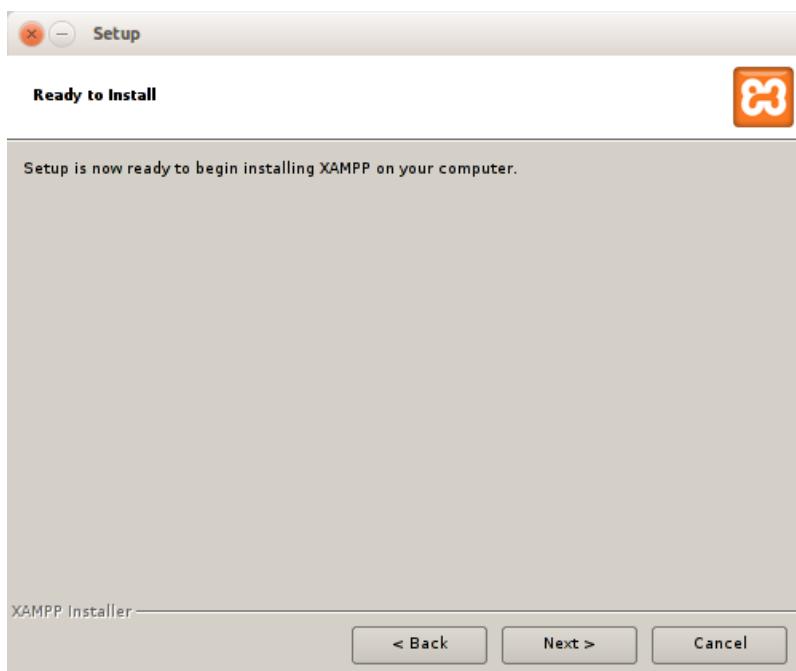
FIGURA 18 - INSTALADOR DO XAMPP NA QUARTA ETAPA



FONTE: O autor

12. A etapa a seguir é muito importante, pois é a sua última chance de cancelar a instalação sem que nenhuma mudança seja feita no seu computador. Mas no nosso caso, queremos sim instalar o XAMPP, portanto, vamos clicar em **Next**:

FIGURA 19 - INSTALADOR DO XAMPP NA QUINTA ETAPA



FONTE: O autor

13. A instalação será iniciada, acompanhe o progresso da operação, conforme a etapa a seguir:

FIGURA 20 - INSTALADOR DO XAMPP NA SEXTA ETAPA



FONTE: O autor

14. Se tudo ocorrer normalmente (sem erros), você visualizará a última etapa do Instalador, desmarque a opção “Iniciar o XAMPP” e clique em *Finish*:

FIGURA 21 - INSTALADOR DO XAMPP NA SÉTIMA E ÚLTIMA ETAPA



FONTE: O autor

15. Agora, é necessário dar permissões de administrador à pasta de Projetos. Abra um terminal, digite o comando ‘sudo chmod 777 /opt/lampp/htdocs’ e digite sua senha de administrador quando solicitada, conforme segue:



No comando acima '/opt/lampp/htdocs' é o diretório de instalação do XAMPP indicado na etapa 10.

FIGURA 22 - ATRIBUINDO PERMISSÕES AO DIRETÓRIO HTDOCS

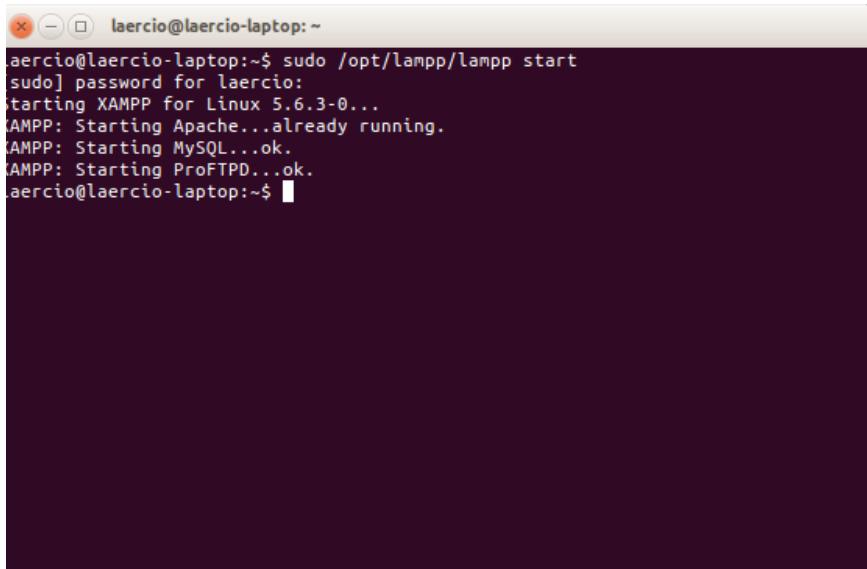
A screenshot of a terminal window titled "laercio@laercio-laptop: ~". The window contains the following text:
laercio@laercio-laptop:~\$ sudo chmod 777 /opt/lampp/htdocs
[sudo] password for laercio:
laercio@laercio-laptop:~\$ █

The terminal window has a dark background and light-colored text. The title bar is visible at the top.

FONTE: O autor

16. Agora é só testar! Feche todas as janelas e abra um novo terminal. Execute o comando ‘sudo /opt/lampp/lampp start’, digite sua senha de administrador e verifique se não ocorreu nenhum erro (se tudo ocorreu sem erros, deve estar como segue):

FIGURA 23 - INICIANDO O XAMPP

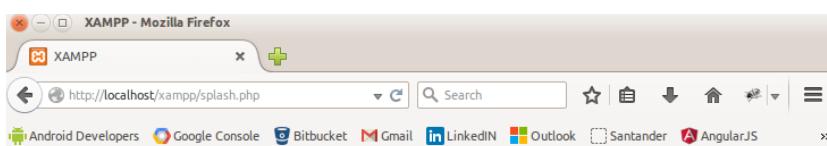


```
aercio@laercio-laptop:~$ sudo /opt/lampp/lampp start
[sudo] password for laercio:
Starting XAMPP for Linux 5.6.3-0...
XAMPP: Starting Apache...already running.
XAMPP: Starting MySQL...ok.
XAMPP: Starting ProFTPD...ok.
aercio@laercio-laptop:~$
```

FONTE: O autor

17. Abra seu *browser* e acesse a URL <http://localhost:80> e veja se o XAMPP está respondendo, conforme segue:

FIGURA 24 - PÁGINA INICIAL DO XAMPP RODANDO NO HOST LOCAL



[English](#) / [Deutsch](#) / [Français](#) / [Nederlands](#) / [Polski](#) / [Italiano](#) / [Norsk](#) / [Español](#) / [中文](#) / [Português \(Brasil\)](#) / [日本語](#)

FONTE: O autor

18. Agora vamos parar o servidor, feche todas as janelas e abra um novo terminal. Nele digite o comando 'sudo /opt/lampp/lampp stop', digite sua senha de administrador quando solicitado e se tudo ocorrer bem, o resultado deve ser conforme segue:

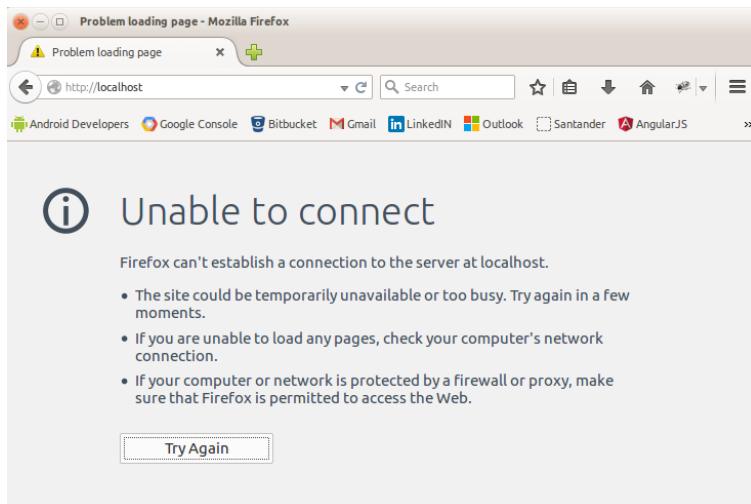
FIGURA 25 - FINALIZANDO O XAMPP

```
laercio@laercio-laptop:~$ sudo /opt/lampp/lampp stop
[sudo] password for laercio:
Stopping XAMPP for Linux 5.6.3-0...
XAMPP: Stopping Apache...ok.
XAMPP: Stopping MySQL...ok.
XAMPP: Stopping ProFTPD...ok.
laercio@laercio-laptop:~$
```

FONTE: O autor

19. Abra seu *browser* e acesse a URL <<http://localhost:80/>> e veja se o XAMPP está respondendo, certamente ele parou de responder, conforme segue:

FIGURA 26 - XAMPP FINALIZADO, A PÁGINA INICIAL NÃO RESPONDENDO MAIS



FONTE: O autor

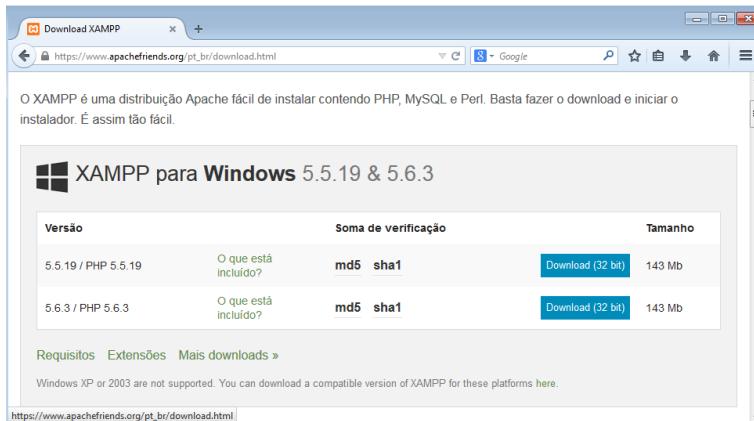
Instalando XAMPP no sistema Operacional Windows

Instalar o XAMPP no Windows leva algumas etapas a menos em relação ao Ubuntu. Então não iremos fazer um tutorial inteiro novamente, pois seria redundante. Em vez disto, iremos guiá-lo pelas etapas do tutorial anterior e focar nas que sofrem alteração. Mão à obra:

1. Acesse o site <<https://www.apachefriends.org>> e acesse o menu *download* (da mesma forma como descrito na primeira etapa do tutorial do Ubuntu).

2. Identifique a versão mais recente do XAMPP para Windows disponível e clique em *download* para baixar o instalador (No momento de escrita deste Caderno de Estudos, não havia distinção entre as arquiteturas 32 e 64 bits dos instaladores XAMPP para Windows, sendo que o de 32 bits funciona também em máquinas de arquitetura 64 bits):

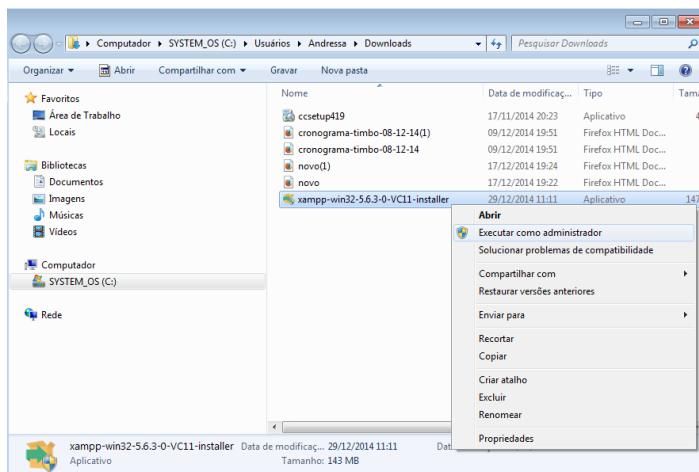
FIGURA 27 - ÁREA DE DOWNLOAD PARA WINDOWS DA PÁGINA DE DOWNLOADS DA APACHE FRIENDS



FONTE: O autor

3. Uma nova aba no seu navegador irá abrir e o *download* irá iniciar (da mesma forma como ocorre na terceira etapa do tutorial para Ubuntu).
4. Quando o *download* estiver concluído, encontre o instalador na sua pasta de *downloads* e clique sobre ele com o botão direito e selecione a opção Executar como administrador (ainda antes, desative seu *software* antivírus, caso tenha, e não se esqueça de reativá-lo ao término da instalação):

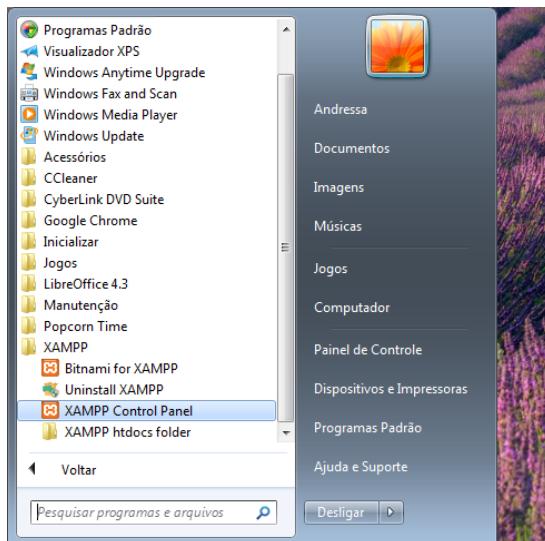
FIGURA 28 - EXECUTANDO O INSTALADOR COMO ADMINISTRADOR



FONTE: O autor

5. Siga as etapas de 8 a 14 do tutorial de instalação do XAMPP para Ubuntu (tutorial anterior).
6. Na barra de tarefas, acesse o menu Iniciar, Todos os Programas, XAMPP, XAMPP Control Panel:

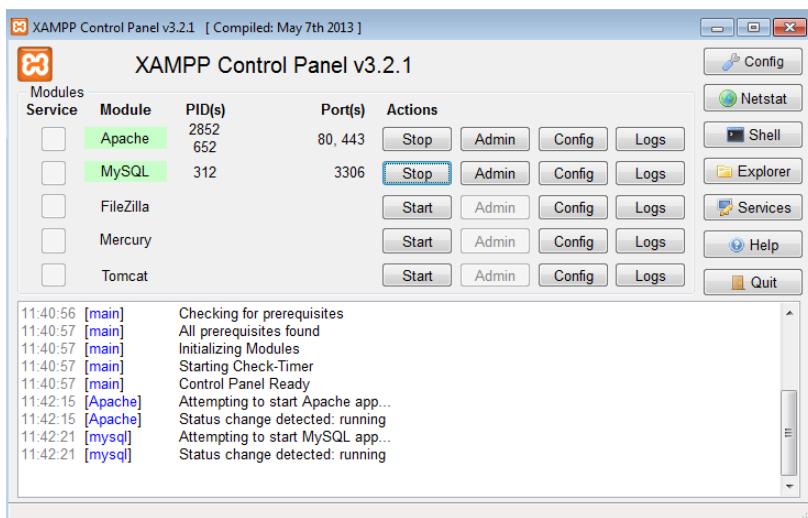
FIGURA 29 - ACESSANDO O XAMPP CONTROL PANEL



FONTE: O autor

7. O XAMPP Control Panel irá abrir. Encontre os serviços Apache e MySQL e pressione o botão Start ao lado de cada um. Quando os serviços estiverem iniciados corretamente, eles aparecerão com a cor de fundo verde:

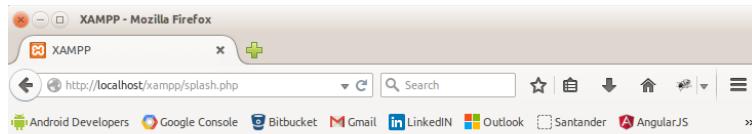
FIGURA 30 - XAMPP CONTROL PANEL COM OS SERVIÇOS APACHE E MYSQL EM EXECUÇÃO



FONTE: O autor

8. Abra seu *browser* e acesse <http://localhost:80> e verifique se o XAMPP está respondendo conforme a imagem a seguir:

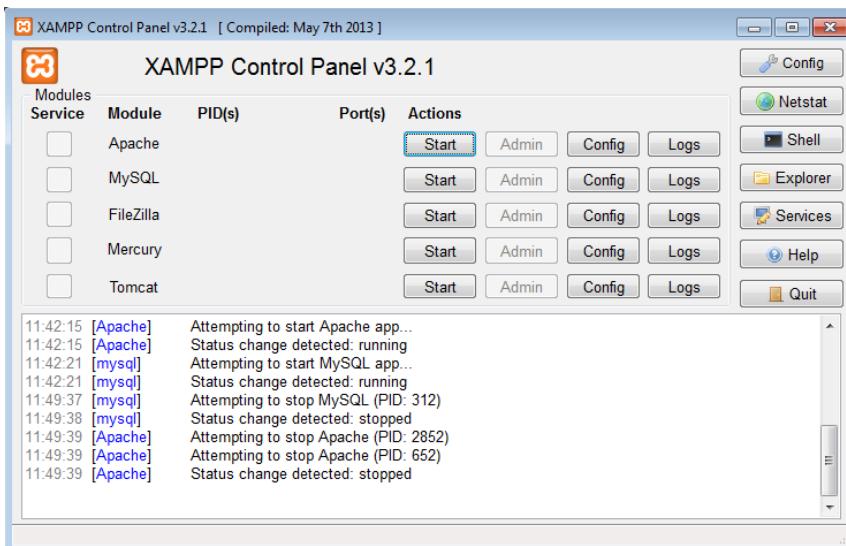
FIGURA 31 - PÁGINA INICIAL DO XAMPP



FONTE: O autor

9. No XAMPP *Control Panel*, localize os serviços Apache e MySQL e pressione o botão *Stop* ao lado de cada serviço. Note que feito isto, os serviços não aparecem mais destacados com a cor de fundo verde:

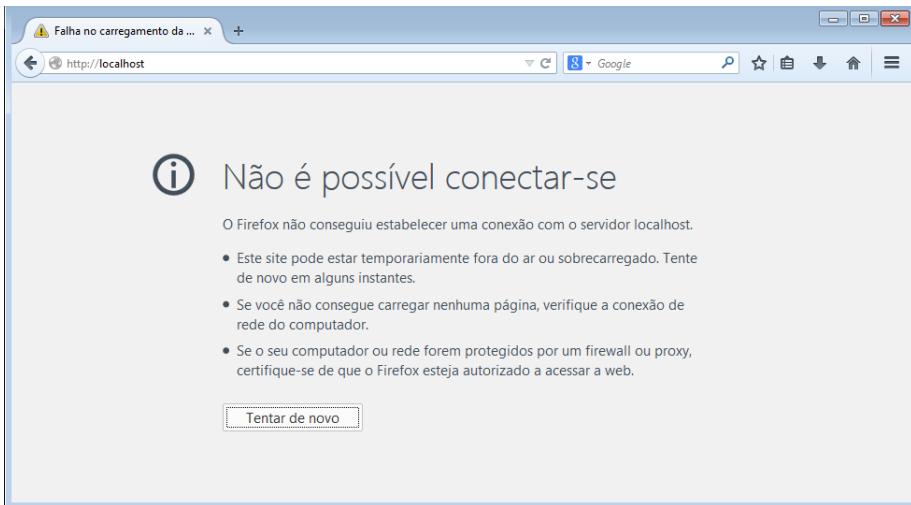
FIGURA 32 - XAMPP CONTROL PANEL COM OS SERVIÇOS APACHE E MYSQL FINALIZADOS



FONTE: O autor

10. No seu *browser* acesse novamente a URL <<http://localhost:80>> e verifique, obviamente, o XAMPP não está mais respondendo, pois finalizamos seus serviços.

FIGURA 33 - XAMPP FINALIZADO, A PÁGINA INICIAL NÃO RESPONDENDO MAIS



FONTE: O autor



A partir deste momento, precisamos que você entenda algumas ações comuns que serão mencionadas ao longo de seu Caderno de Estudos. Consideraremos que as ações a seguir estejam subentendidas.

- Daqui em diante, quando falarmos em iniciar o XAMPP, significa que se você estiver utilizando o sistema operacional Windows, você deve abrir o XAMPP Control Panel, e iniciar o MySQL e o Apache (da mesma forma que você fez na sexta e sétima etapa do tutorial de instalação para Windows). Caso você esteja utilizando o sistema operacional Ubuntu, você deve abrir um terminal e executar o comando 'sudo /opt/lampp/lampp start' (de forma idêntica à décima sexta etapa do tutorial de instalação do XAMPP para Ubuntu).
- Quando falarmos em finalizar o XAMPP, significa que se você estiver utilizando o sistema operacional Windows, você deve abrir o XAMPP Control Panel, e finalizar o MySQL e o Apache (da mesma forma que você fez na nona etapa do tutorial de instalação para Windows). Caso você esteja utilizando o sistema operacional Ubuntu, você deve abrir um terminal e executar o comando 'sudo /opt/lampp/lampp stop' (de forma idêntica à décima oitava etapa do tutorial de instalação do XAMPP para Ubuntu).
- Da mesma forma, daqui em diante, quando mencionarmos o 'Diretório de Projetos' ou o 'Diretório htdocs', estaremos mencionando o diretório <C:\xampp\htdocs> (no caso do Windows) ou </opt/lampp/htdocs> (no caso do Ubuntu)

RESUMO DO TÓPICO 2

Neste tópico, você aprendeu:

- Instalar o XAMPP no Windows e no Linux Ubuntu.
- Para criar um projeto *Web* no XAMPP, os arquivos do projeto devem estar dentro do diretório de projetos (htdocs).
- Para acessar seus projetos *Web*, deve-se iniciar o XAMPP e acessar a URL <<http://localhost:80/nome-do-projeto/>>.

AUTOATIVIDADE



Agora que temos o XAMPP instalado, é hora de explorá-lo:

- 1 Crie um novo diretório chamado 'helloworld' (sem aspas) abaixo do diretório htdocs.
- 2 Dentro do diretório helloworld cole o arquivo helloworld.html que você criou durante seus estudos no Tópico 1.
- 3 Inicie o XAMPP.
- 4 Abra seu Browser e acesse a URL <http://localhost:80/helloworld/helloworld.html>.
- 5 Edite a página helloworld.html que você copiou para dentro do diretório htdocs com o editor de texto e implemente algo novo nela.
- 6 Para visualizar as alterações é necessário somente atualizar a guia do seu navegador na qual você acessou a URL <http://localhost:80/helloworld/helloworld.html> (pressionar a tecla F5).
- 7 Não se esqueça de, quando terminar, finalizar o XAMPP.

The screenshot shows a Sublime Text window with the title bar reading '/opt/lampp/htdocs/helloworld/helloworld.html - Sublime Text (UNREGISTERED)'. The code editor displays the following HTML content:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8">
    <title>Hello World</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
  </head>
  <body>
    <p>
      Hello World
    </p>
    <h1>Seja bem-vindo ao desenvolvimento Web !!</h1>
  </body>
</html>
```



A figura mostra alterações sendo feitas no arquivo helloworld.html e a página *Hello World* já atualizada.

DESTRINCHANDO O PHP, DESDE SUA HISTÓRIA ATÉ SEUS MÉTODOS

1 INTRODUÇÃO

Caro(a) acadêmico(a)! O PHP é uma linguagem de programação, muito utilizada e especialmente adequada para o desenvolvimento *Web* e que pode ser inserida dentro do HTML. Ótimo, mas o que isso realmente significa? O HTML por si só não nos proporciona formas de dinamizar os conteúdos de uma página, por se tratar de uma linguagem de marcação. O PHP age como linguagem de programação unida ao HTML, fazendo com que as páginas *Web* sejam mais dinâmicas. Prepare-se para explorar o PHP!

2 UM POUCO DE HISTÓRIA

O surgimento do PHP é uma história um tanto quanto inusitada. Você possui um *site* pessoal? Ou conhece alguém que tenha um *site* pessoal? Ou por acaso, você conhece alguém que tenha criado uma linguagem de programação para desenvolver seu *site* pessoal?

Pois bem, em meados de 1995, um cidadão canadense chamado Rasmus Lerdorf desenvolvia seu *site* pessoal. Ele então começo a perceber que boa parte do código que estava criando poderia ser abstraído, e começou a criar *scripts* em Perl (linguagem que estava utilizando para desenvolver seu *site*) para utilizar como bibliotecas, otimizando o reaproveitamento. O código foi aprimorado com o tempo, e certo dia, Rasmus decidiu portá-lo para a linguagem C. Foi assim que surgiu o embrião do PHP, o PHP/FI (*Personal Home Page/Forms Interpreter*).

O PHP/FI foi ganhando muitos usuários, e foi necessário efetuar algumas manutenções e correções no *core* da plataforma. Estas alterações deram origem à versão 2.0. Dois anos depois, em 1997, os projetos *Web* demandavam mais do que o PHP tinha a oferecer, ao perceber isto, dois desenvolvedores, Andi Gutmans e Zeev Suraski, contando com o apoio do criador do PHP, reescreveram completamente a linguagem. Então, o PHP definitivamente decolou, devido as suas características extremamente convenientes para projetos *Web*. Atualmente, o que temos em evidência é o PHP 5.



Core: Núcleo, parte mais importante de uma plataforma de desenvolvimento.

O termo PHP é um acrônimo, que resume toda esta história, pois traduzido quer dizer 'Página Pessoal':

- P: Personal.
- H: Home.
- P: Page.

FIGURA 34 - RASMUS LERDORF



FONTE: Disponível em: <<https://goo.gl/T6ETFA>>. Acesso em: 3 de. 2014.

3 DESTRINCHANDO O PHP

Podemos citar duas características principais, primeiro, o PHP é uma linguagem *Case Sensitive*, ou seja, onde letras maiúsculas diferem de letras minúsculas no que diz respeito a palavras reservadas, comandos e nomes de variáveis. Segundo, PHP é uma linguagem de tipagem dinâmica (também conhecida como tipagem fraca), o que significa que o tipo da variável não necessita ser declarado e pode alterar em tempo de execução.



Se você não compreendeu como funciona a tipagem dinâmica, iremos explorar esta característica do PHP mais adiante. Não se preocupe.

3.1 HELLO WORLD

Por onde começar a estudar uma tecnologia? Você já sabe! Vamos fazer nosso *Hello World* em PHP:

1. Logo abaixo do diretório htdocs crie um diretório chamado helloworldphp.
2. Dentro do diretório helloworldphp crie um arquivo index.php.
3. No arquivo index.php, insira o conteúdo da listagem a seguir e salve o arquivo:

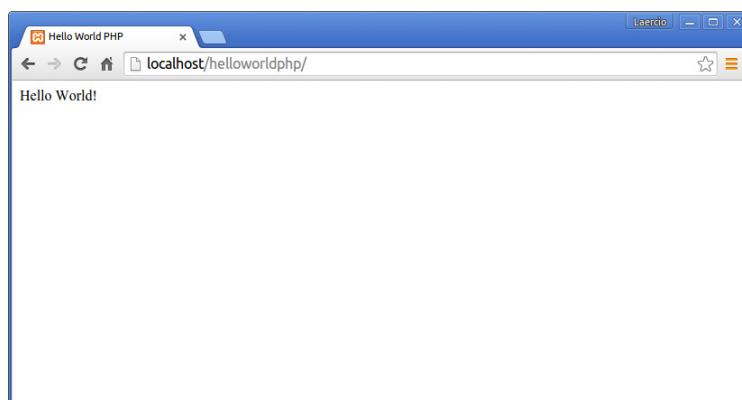
LISTAGEM 5 – PÁGINA WEB HELLO WORLD IMPLEMENTADA EM PHP

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Hello World</title>
    </head>
    <body>
        <?PHP
            echo "<p>Hello World!</p>";
        ?>
    </body>
</html>
```

4. Inicie o XAMPP

5. Em seu *browser*, acesse a URL <<http://localhost:80/helloworldphp/>>, a página obtida deve estar conforme segue:

FIGURA 35 - PÁGINA WEB HELLO WORLD IMPLEMENTADA EM PHP



FONTE: O autor

6. Obs.: Ao término, não esqueça de finalizar o XAMPP.

Pois bem, quanta coisa que certamente você já conhece! Mas para consolidar os conhecimentos, vamos explorar a fundo nosso *Hello World*:

1. Criamos um diretório dentro do diretório de projetos (`htdocs`) chamado `helloworldphp` e criamos neste diretório o arquivo `index.php`. O arquivo `index.php` é encontrado em praticamente todos os projetos PHP, pois é a página inicial quando o usuário acessa o nosso *site*. Você deve lembrar que quando estudávamos o protocolo HTTP, no Tópico 1, acessamos a URL `<http://www.grupouniasselvi.com.br/>` e recebemos a página `index.aspx` como retorno. Isto também acontece no caso do PHP, o Apache (servidor Web do XAMPP) retorna a página chamada `index.php` caso não tenha sido especificado nenhuma página na URL. Desta forma podemos acessar a URL do nosso *site* `<http://localhost:80/helloworldphp/>` e receberemos como resposta o arquivo `index.php`.
2. O conteúdo do arquivo `index.php` possui, em grande parte, código HTML5 (que dispensa explicações). Arquivos com extensão `.php` são páginas *Web* que contêm código-fonte em HTML juntamente com código-fonte PHP. A novidade no conteúdo do arquivo `index.php` fica na *tag* `<?php .. ?>` e no comando `echo`, vamos ver por partes:
 - Em uma página PHP, todo o código-fonte que se encontrar entre a *tag* de abertura do PHP (`<?php`) e seu fechamento (`?>`) é código-fonte PHP, enquanto o que estiver fora destas *tags* é código-fonte HTML. Ou seja, a *tag* de abertura e fechamento do PHP (respectivamente '`<?php`' e '`?>`') tem por finalidade separar o código de marcação (HTML) do código de programação (PHP). Uma página PHP pode conter quantos trechos de código em PHP forem necessários, a única regra é que eles devem se encontrar entre uma *tag* de abertura e uma *tag* de fechamento PHP.
 - O comando `echo` do PHP tem por finalidade imprimir conteúdo na página *Web*. A *string* que o comando `echo` recebe como parâmetro será impressa na página, ou seja, será concatenada ao HTML exatamente no ponto onde reside a *tag* de abertura PHP que contém o comando `echo`. No caso do comando `echo`, o uso de parênteses para passagem de parâmetros é opcional.



O COMANDO ECHO IRÁ IMPRIMIR ALGO?

Quando mencionamos que o conteúdo do parâmetro do comando `echo` será impresso na página, não quer dizer que será impresso em papel através de uma impressora. O que queremos dizer, é que o conteúdo será adicionado a página *Web*, exatamente onde o comando `echo` estiver. É um conceito análogo ao `Console.WriteLine()` do C#, `System.out.println()` do Java, porém, em vez de escrever no Terminal, o PHP escreve diretamente na página HTML.

Caro(a) acadêmico(a)! Você deve ter percebido que a versão PHP da nossa página *Hello World* é bem simples. É o mesmo *Hello World* que implementamos quando estudávamos HTML, porém o parágrafo com a mensagem '*Hello World!*', no corpo da página é impresso através do comando *echo* do PHP.

3.2 VARIÁVEIS E TIPOS DE DADOS

O PHP suporta os tipos de dados *integer*, *double*, *string*, *boolean*, *array* e *object*. Você provavelmente já conhece estes tipos de dados de alguma outra disciplina (Algoritmos ou Linguagem de Programação), pois geralmente eles são bem semelhantes em outras linguagens. De qualquer forma, vamos descrevê-los a seguir:

- Integer: Valores inteiros (positivos e negativos). Exemplo: 24; -36; 0; etc.
- Double: Valores com ponto flutuante, ou seja, frações (positivas ou negativas). Exemplo: 16,71; 0,00; -0,24; etc.
- String: Textos. Exemplo: "
Hello World!</br>", "UNIASSELVI - NEAD" etc.
- Boolean: Verdadeiro ou Falso.
- Object: Objetos são estruturas de dados definidos através de uma classe, que armazenam informações e encapsulam métodos durante a execução do programa.



OBJECT?

Caso você não teve contato com a Programação Orientada a Objetos (POO), certamente deve se perguntar: 'Object? Como assim?' Se for o seu caso, não se preocupe, pois não iremos focar em orientação a objetos nesta disciplina. Inclusive, o suporte a Programação Orientada a Objetos no PHP deixa a desejar, pois o PHP não suporta polimorfismo de métodos. Caso você seja um entusiasta da Orientação a Objetos, pesquise sobre o assunto. Se você já domina alguma linguagem orientada a objetos, reflita sobre a seguinte questão: Você pensa que vale a pena abrir mão do Polimorfismo de Métodos para usufruir da Tipagem dinâmica? A conclusão é subjetiva.

Para declarar uma variável em PHP é necessário utilizar o símbolo \$ (dólar) seguido do nome da variável, que não deve conter espaços em branco nem caracteres especiais e também não deve iniciar com números. Opcionalmente, a variável pode ser inicializada, caso contrário será nula. Vamos ver na prática?

1. Em seu diretório de projetos (htdocs) crie um diretório chamado 'variaveis'.
2. Dentro do diretório 'variaveis' crie um arquivo chamado index.php.

3. Insira o conteúdo da listagem a seguir em seu arquivo index.php.

LISTAGEM 6 - PÁGINA WEB HELLO PARA ESTUDOS SOBRE VARIÁVEIS EM PHP

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
        <title>Variáveis</title>
</head>
<body>
    <h1>Vamos estudar as variáveis!</h1>
    <h2>Variável nula:</h2>
    <?php
        /* Declara uma variável e deixa ela nula */
        $variavelnula;
        echo "<p>".$variavelnula."</p>"; // aqui vai
dar erro ;-)
    ?>
    <br />
    <h2>Variável String:</h2>
    <?
        /* Declara uma variável String */
        $variavelString123 = "um, dois, três";
        echo "<p>".$variavelString123."</p>";
    ?>
    <br />
    <h2>Variável Integer:</h2>
    <?
        /* Declara uma variável Intger */
        $variavelInteger = 123;
        echo "<p>".$variavelInteger."</p>";
    ?>
    <br />
    <h2>Variável Double:</h2>
    <?
        /* Declara uma variável Double */
        $variaveldouble = 0.24;
        echo "<p>".$variaveldouble."</p>";
    ?>
</body>
</html>
```

4. Inicie o XAMPP.

5. Em seu *browser* acesse a URL <<http://localhost:80/variaveis>> e visualize sua página.

6. Ao terminar, não esqueça de finalizar o XAMPP.



NOSSA, QUANTA COISA!

Inevitavelmente, nosso estudo sobre variáveis acabou revelando outros aspectos importantes da linguagem de programação PHP:

- Os comandos são finalizados com ponto e vírgula (;
- O operador de atribuição é o igual (=)
- O operador para concatenação é o ponto (.)
- A primeira tag de abertura de código PHP precisa ser escrita completamente ('<?php'), as demais podem ser abreviadas ('<?')
- Para abertura e fechamento de comentários multilinhas utiliza-se respectivamente os caracteres '/*' e '*/' (sem aspas)
- Para comentário somente de uma linha, utiliza-se //' (sem aspas)

3.3 DECLARAÇÃO DE CONSTANTES

Constantes em PHP podem ser consideradas variáveis globais cujo valor não pode ser alterado, apenas inicializado. A declaração de constantes no PHP é feita pela função *define*, o primeiro parâmetro é o nome da constante, o segundo parâmetro é o valor da constante, e o terceiro é opcional, é um booleano que indica se a variável deve ou não ser *case sensitive*. A função *define* ainda devolve um booleano indicando sucesso ou não na criação da variável. A sintaxe é apresentada a seguir:

```
(boolean) define (<nome_da_constante>, <valor_da_constante>, [<não_é_case_sensitive>]);
```

O trecho de código PHP da listagem a seguir exemplifica a criação de constantes, você pode inserir este trecho de código em qualquer arquivo PHP (por exemplo, seu *Hello World*) para fins de teste:

LISTAGEM 7 - DECLARAÇÃO DE CONSTANTES EM PHP

```
<?php
    // Declara uma constante Case Sensitive
    define ("constanteCaseSensitive", "CASE_SENSITIVE");
    echo constanteCaseSensitive;
    echo "<br />";
    // Declara uma constante não Case Sensitive
    define ("constanteNaoCaseSensitive", "NAO_CASE_SENSITIVE", true);
    echo ConsTanTeNaoCaseSensiTive;
?>
```



PARA APRENDER DE VERDADE

A partir de agora, não iremos mais indicar quando você deve criar um projeto novo no PHP (diretório `htdocs`). Mas alertamos, para aprender de verdade, teste todos os trechos de códigos deste tópico. Você pode utilizar o seu arquivo `index.php` da versão PHP do projeto *Hello World*, que criamos no início do tópico. Basta transcrever o código-fonte dos exemplos no corpo da página (entre a tag `body` e seu fechamento).

3.4 OPERADORES LÓGICOS, MATEMÁTICOS E CONDICIONAIS

Como ocorre em toda e qualquer linguagem de programação, o PHP possui operadores lógicos, matemáticos e condicionais. Veja a seguir os operadores que o PHP nos oferece.

TABELA 2 - OPERADORES MATEMÁTICOS DO PHP

Operador	Descrição
=	Atribui o valor de sua direita à variável em seu lado esquerdo.
.	Concatena dois valores de formato de texto (<i>string</i>).
+	Soma dois valores numéricos.
-	Subtrai dois valores numéricos.
++	Incrementa em 1 o valor da variável numérica à sua esquerda.
--	Decrementa em 1 o valor da variável numérica à sua esquerda.
*	Multiplica dois valores numéricos.
/	Divide dois valores numéricos.
%	Obtém o resto da divisão entre dois valores numéricos.
.=	Concatena a variável à sua esquerda o valor apresentado em sua direita.
+=	Soma à variável à sua esquerda o valor apresentado em sua direita.
-=	Subtrai da variável à sua esquerda o valor apresentado em sua direita.
*=	Multiplica a variável à sua esquerda o valor apresentado em sua direita.
/=	Divide à variável à sua esquerda pelo valor apresentado em sua direita.
%=	Atribui à variável da esquerda o resto de sua divisão pelo valor apresentado em sua direita.

FONTE: MILANI (2010, p. 40)

TABELA 3 - OPERADORES CONDICIONAIS DO PHP

Operador	Descrição
==	Compara se dois valores têm o mesmo valor.
==	Compara se dois valores são idênticos.
!=	Compara se dois valores são diferentes.
◊	Equivalente a !=.

<	Compara se o valor da direita é menor que o da esquerda.
>	Compara se o valor da direita é maior que o da esquerda.
<=	Compara se o valor da direita é menor ou igual ao da esquerda.
>=	Compara se o valor da direita é maior ou igual ao da esquerda.

FONTE: MILANI (2010, p. 41)

TABELA 4 - OPERADORES LÓGICOS DO PHP

Operador	Descrição
!	Nega a condição informada (inverte o resultado lógico).
&&	Retorna verdadeiro se ambas as condições da esquerda e direita forem satisfeitas.
	Retorna verdadeiro se pelo menos uma das condições (da esquerda ou da direita) for satisfeita.

FONTE: MILANI (2010, p. 41)

3.5 CONTROLANDO O FLUXO DO CÓDIGO COM *IF*, *ELSE*, *SWITCH*, *WHILE* E *FOR*

Todas as linguagens de programação oferecem comandos para controlar o fluxo de código. Você provavelmente teve seu primeiro contato com tais comandos quando estudava algoritmos. O PHP fornece todos os operadores de controle de fluxo de execução comuns às demais linguagens. Iremos em seguida apresentá-los e exemplificá-los a você.



O propósito deste tópico é demonstrar e exemplificar quais e como são os comandos para controle de fluxo de execução em PHP. Pressupõe-se que você já conta com vivência com estes comandos, de disciplinas anteriores, como Algoritmos ou Linguagens de Programação.

- ‘Se meu saldo bancário for maior ou igual a R\$ 42.390,00, irei comprar um carro 0km, senão, comprarei um carro usado’. O condicional *if* é sem dúvida o mais tradicional de todas as linguagens. Em PHP, o comando *if* recebe como parâmetro uma expressão booleana (obrigatoriamente entre parênteses) e um bloco de código para ser executado caso a expressão booleana a ser examinada seja verdadeira. Opcionalmente, pode-se adicionar a cláusula *else*, com um bloco de código a ser executado caso a expressão booleana analisada seja falsa. A proposição anterior seria escrita conforme a listagem a seguir:

LISTAGEM 8 - CONTROLE DE FLUXO DE EXECUÇÃO COM IF..ELSE EM PHP

```
<?php
    /* Demonstração de if em PHP
     * De acordo com o valor da variável
\$saldoBancario
    * o resultado do programa pode variar.
    */

\$saldoBancario = 42389.50;
if (\$saldoBancario >= 42390) {
    echo "Vou comprar um carro 0km :-D";
} else { // o else é opcional
    echo "Vou comprar um carro usado :-|";
}
?>
```

- ‘Nenhum erro encontrado’, ‘Um erro encontrado’, ‘Dois erros encontrados’, ‘Mais de dois erros encontrados’. Fazer uma mensagem bonita para validação de um formulário de cadastro não é tarefa fácil! No PHP, contamos com o comando condicional *switch* para facilitar. O comando *switch* recebe como parâmetro (entre parênteses) um valor a ser analisado, e dependendo do valor fornecido, ações distintas poderão ser implementadas. Veja o exemplo a seguir:

LISTAGEM 9 - CONTROLE DE FLUXO DE EXECUÇÃO COM SWITCH EM PHP

```
<?php
/* Demonstração de switch em PHP
 * Mensagem de erro/sucesso para validação de
formulários.
 * O algoritmo monta a mensagem de erro de acordo com
 * a quantidade de erros encontrada.
*/
\$quantidadeErros = 3; // tem que ser maior ou igual a 0
switch (\$quantidadeErros) {
    case 0:
        \$mensagemDeErro = "Nenhum";
        break;
    case 1:
        \$mensagemDeErro = "Um";
        break;
    case 2:
        \$mensagemDeErro = "Dois";
        break;
    default:
```

```

        $mensagemDeErro = "Mais de dois";
    }
$mensagemDeErro .= " erro(s) encontrado(s)";
echo $mensagemDeErro;
?>
```

- ‘Liste os algarismos de um a dez’. O controle de execução através de *loopings* é uma prática muito comum no desenvolvimento *Web*, e você certamente já possui conhecimentos dos *looping* de disciplinas anteriores. Vamos exemplificar a utilização dos *loopings for* e *while* em PHP. Acompanhe na listagem a seguir:

LISTAGEM 10 - CONTROLE DE FLUXO DE EXECUÇÃO COM LOOPINGS EM PHP

```

<H1>While</h1>
<? Php
    /* Demonstração de while em PHP
     * O while recebe como parâmetro (entre
parênteses)
        * um valor booleano e permanece em looping até
quanto
        * a condição for verdadeira.
        */
$contador = 0;
While ($contador++ < 10) {
    Echo $contador."<br />";
}
?>
<h1>For</h1>
<?
    /* Demonstração de for em PHP
     * O for recebe três parâmetros (entre parênteses
     *, separados por ponto e vírgula (;))
     * O primeiro é a inicialização do contador
     * O segundo é um booleano que define até quando o
looping será executado
        * O terceiro é um incremento para o contador
        */
For ($contador = 1; $contador <= 10; $contador++)
{
    Eco $contador."<br />";
}
?>
```

3.6 FUNÇÃO ISSET()

Em códigos-fonte muito extensos e complexos, muitas vezes perdemos a noção de se já criamos ou não uma variável. Em PHP, esta verificação pode ser feita utilizando a função `isset()`, que recebe como parâmetro a variável a ser verificada e retorna um booleano, informando se esta variável está ou não declarada na execução do programa. A listagem a seguir exemplifica a situação na qual uma variável é verificada e não está declarada e em seguida, ela é declarada e então verificada novamente.

LISTAGEM 11 - CONTROLE DE FLUXO DE EXECUÇÃO COM LOOPINGS EM PHP

```
<h1>A variável não está declarada:</h1>
<?php
    // Declara uma variável porém deixa-a nula:
    $variavel;
    if (isset($variavel)){
        echo "A variável está declarada!";
    } else {
        echo "A variável não está declarada!"; //
aqui :-)
    }
?>
<h1>A variável está declarada:</h1>
<?
    // Declara uma variável e a inicializa:
    $variavel = "variável";
    if (isset($variavel)){
        echo "A variável está declarada!"; // aqui
:-)
    } else {
        echo "A variável não está declarada!";
    }
?>
```

3.7 STRINGS

Você aprendeu no início do tópico como declarar uma variável *string*. Da mesma forma, já sabe que para imprimir uma *string* é possível utilizar a função `echo`. Mas há mais formas de imprimir uma *string* e há muitas outras manipulações possíveis além de simplesmente inicializar uma *string* e concatená-la a outro valor.

A primeira grande novidade é a diferença que há entre a utilização de aspas simples e aspas duplas em PHP. Quando você utiliza aspas simples, o valor da *string* é exatamente o que é, enquanto quando você utiliza aspas duplas, o valor da *string* passa por uma formatação (o PHP encontra nomes de variáveis e substitui por seus valores, por exemplo). Para seus estudos com *strings* criaremos um arquivo separado, então acompanhe em seu computador as etapas a seguir:

1. Em seu diretório de projetos (htdocs), crie um diretório chamado 'strings' (sem aspas).
2. Dentro do diretório *strings*, crie um arquivo chamado index.php.
3. Insira no arquivo index.php o código-fonte da seguinte listagem:

LISTAGEM 12 - TESTE DE STRINGS NO PHP

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
        <title>Strings</title>
</head>
<body>
    <h1>Estudando strings no PHP!</h1>
    <?php
        // Declaramos uma variável string para
        // o nome de uma linguagem de programação.
        $linguagemDeProgramacao = "PHP";
    ?>
    <h2>echo com aspas simples:</h2>
    <?
        echo '<p>Estudar $linguagemDeProgramacao é
        muito Legal !!  :-)</p>';
    ?>
    <h2>echo com aspas duplas:</h2>
    <?
        echo "<p>Estudar $linguagemDeProgramacao é
        muito Legal !!  :-)</p>";
    ?>
</body>
</html>

```

4. Inicie o XAMPP
5. Em seu *browser*, acesse a URL <http://localhost:80/strings/> e verifique se está de acordo com a figura a seguir:

FIGURA 36 - PROJETO STRINGS RODANDO LOCALMENTE NO XAMPP



FONTE: O autor

6. Quando pronto, não se esqueça de finalizar o XAMPP



Para testar os demais exemplos de seus estudos sobre *strings* em PHP, você poderá utilizar o projeto *strings* criado nas etapas acima.

Agora vamos apresentar outro método para impressão de *strings*, que possui a mesma função do *echo*, o *print*, teste o trecho de código a seguir no projeto da listagem anterior:

LISTAGEM 13 - EXEMPLOS COM O COMANDO PRINT DO PHP

```
<h1>Vamos estudar o método print</h1>
<?php
    // Declaramos uma variável string para armazenar
    // o nome de uma instituição de ensino.
    $instituicaoDeEnsino = "UNIASSELVI";
?
<h2>print com aspas simples:</h2>
<?
    print '<p>Estudar na $instituicaoDeEnsino é muito
Legal !! :-)</p>';
?
```

```
<h2>print com aspas duplas:</h2>
<?
    print "<p>Estudar na $instituicaoDeEnsino é muito
Legal !!  :-) </p>";
?>
```

Agora, vamos partir para as funções de manipulação de *strings*. O PHP oferece várias, vamos focar nas mais usadas por enquanto:

- **trim()**: remove os espaços em branco em volta da *string*.
- **ltrim()**: remove os espaços em branco à esquerda da *string*.
- **rtrim()**: remove os espaços em branco à direita da *string*.

LISTAGEM 14 - REMOVENDO ESPAÇOS EM BRANCO EM VOLTA DA STRING COM PHP

```
<h1>Estudando as funções trim()</h1>
<h2>Removendo espaços em branco em volta da string</h2>
<?php
    $stringComEspacosEmVolta = "      Três espaços em cada
lado      ";
    echo "{$stringComEspacosEmVolta."}<br />";

    // Remove os espaços em branco em volta da string
    $stringSemEspacosEmVolta =
trim($stringComEspacosEmVolta);
    echo "{$stringSemEspacosEmVolta."}<br />";

?>
<h2>Removendo espaços em branco à esquerda da string</h2>
<?php
    $stringComEspacosAesquerda = "      Três espaços à
esquerda";
    echo "{$stringComEspacosAesquerda."}<br />";

    // Remove os espaços em branco em volta da string
    $stringSemEspacosAesquerda =
trim($stringComEspacosAesquerda);
    echo "{$stringSemEspacosAesquerda."}<br />";

?>
<h2>Removendo espaços em branco à direita da string</h2>
<?php
    $stringComEspacosAdireita = "Três espaços à direita
";
```

```

echo "{$stringComEspacosAdireita."}<br />";

// Remove os espaços em branco em volta da string
$stringSemEspacosAdireita =
trim($stringComEspacosAdireita);
echo "{$stringSemEspacosAdireita."}<br />";
?>

```

- **strrev():** Inverte uma *string*.

LISTAGEM 15 - INVERTENDO UMA STRING EM PHP

```

<h1>Invertendo uma string em PHP</h1>
<?php
    $stringNormal = "UNIASSELVI";
    $stringInvertida = strrev($stringNormal);
    echo $stringInvertida;
?>

```

- **strtolower():** transforma uma *string* para minúsculo.
- **strtoupper():** transforma uma *string* para maiúsculo.
- **ucfirst():** transforma a primeira letra da *string* para maiúsculo e não altera as demais.

LISTAGEM 16 - MANIPULANDO O CASE DE UMA STRING EM PHP

```

<h1>Manipulando o CASE de strings em PHP</h1>
<?php
    $variavelCaseMisturado = "com a UNIASSELVI, posso
ver o meu futuro!";
?>
<h2>Invertendo para minúscula:</h2>
<?
    echo strtolower($variavelCaseMisturado);
?>
<h2>Invertendo para maiúscula:</h2>
<?
    echo strtoupper($variavelCaseMisturado);
?>
<h2>Invertendo a primeira letra para maiúscula:</h2>
<?
    echo ucfirst($variavelCaseMisturado);
?>

```

- `str_replace()`: substitui uma *string* por outra dentro de uma *string*.

LISTAGEM 17 - SUBSTITUINDO STRINGS EM PHP

```
<h1>Substituição de strings em PHP</h1>
<?php
    $variavelTextoOriginal = "A disciplina Algoritmos
é demais!";
    $variavelTextoAlterado = str_replace("Algoritmos",
"Programação Web I", $variavelTextoOriginal);
    echo $variavelTextoAlterado;
?>
```

- `strlen()`: retorna o tamanho da *string*.

LISTAGEMM 18 - OBTENDO O TAMANHO DE STRINGS EM PHP

```
<h1>Obtendo o tamanho de strings em PHP</h1>
<?php
    $nomeBonito = "PHP";
    $tamanho = strlen($nomeBonito);
    echo "O nome ".$nomeBonito." possui ".$tamanho."
letras";
?>
```

3.8 REDIRECIONANDO PARA OUTRA PÁGINA

Muitas vezes, durante o fluxo de execução de um código-fonte em PHP, pode ser necessário redirecionar para outra página. Por enquanto, iremos exemplificar através de uma página que simplesmente redireciona para o *site* da UNIASSELVI, mas as possibilidades são infinitas e o redirecionamento é uma prática bastante comum, principalmente quando envolve páginas do seu próprio projeto. Vamos criar este projeto juntos:

1. Em seu diretório de projetos (`htdocs`) crie um diretório chamado `gotouniasselvi`.
2. Em seu diretório `gotouniasselvi`, crie um arquivo chamado `index.php`.
3. No arquivo `index.php`, insira o conteúdo da listagem abaixo:

LISTAGEM 19 - REDIRECIONANDO PARA OUTRA PÁGINA EM PHP

```
<!DOCTYPE html>
<html>
```

```

<head>
    <meta charset="utf-8">
        <title>Indo para Uniasselvi</title>
</head>
<body>
<?php

// Redireciona para o site da Uniasselvi
header("Location: http://www.grupouniasselvi.com.br");
?>
</body>
</html>

```

4. Inicie o XAMPP.

5. Em seu *browser*, acesse a URL <http://localhost:80/gotouniasselvi/> e confira, seu script PHP redirecionou para a página da Uniasselvi.
6. Não esqueça de finalizar o XAMPP quando estiver pronto.

3.9 ARRAYS

Arrays agregam muito poder às linguagens de programação que os suportam. São estruturas de dados que contemplam uma lista de itens. Os *arrays* também podem ser chamados vetores. Trabalhar com *arrays* em PHP é considerado mais simples do que na maioria das linguagens. *Arrays* podem ser criados, destruídos, ou ter seu conteúdo alterado e/ou destruído. Os elementos de *arrays* em PHP podem ser de qualquer tipo de dados, inclusive um *array* suporta a adição de elementos de tipos diferentes.

Ao criar um *array*, é possível adicionar elementos para inicialização, ou também é possível deixar o *array* vazio. Para criar um *array*, faz-se o uso do comando *array()*. Acompanhe no seu computador o passo a passo a seguir para iniciar nossos estudos sobre *arrays* em PHP:

1. No seu diretório de projetos (htdocs) crie um subdiretório chamado *arraystudies*.
2. Dentro do diretório *arraystudies*, crie um arquivo chamado *index.php*.
3. Em seu arquivo *index.php*, insira o conteúdo abaixo.

LISTAGEM 20 - DECLARANDO ARRAYS EM PHP

```

<!DOCTYPE html>
<html lang="pt-br">

```

```

<head>
    <meta charset="utf-8">
    <title>Estudando arrays em PHP!</title>
</head>
<body>
<?php

    // Declaração de um array
    // Note que podemos inserir tanto tipos
    // numéricos quanto strings
    // Em seguida obtemos o valor de cada posição do
array
    // Utilizando o nome da variável seguido do índice
da posição
    // entre colchetes
    // Note que o índice inicia em zero
    $meuPrimeiroArray = array(1, 2, 3, "Quatro", 5);
    echo $meuPrimeiroArray[0]."<br />";
    echo $meuPrimeiroArray[1]."<br />";
    echo $meuPrimeiroArray[2]."<br />";
    echo $meuPrimeiroArray[3]."<br />";
    echo $meuPrimeiroArray[4]."<br />";

    // Declaração de um array vazio
    $arrayVazio = array();

    // Note que a variável do array está definida
    if (isset($arrayVazio)){
        echo "O array vazio está declarado !<br />";
    }

    // Porém nenhum índice possui valor
    // por exemplo, tentaremos acessar o índice zero
    if (isset($arrayVazio[0])){

        // Não vai passar aqui!
        echo "A primeira posição do array vazio está
declarada.<br />";

    }

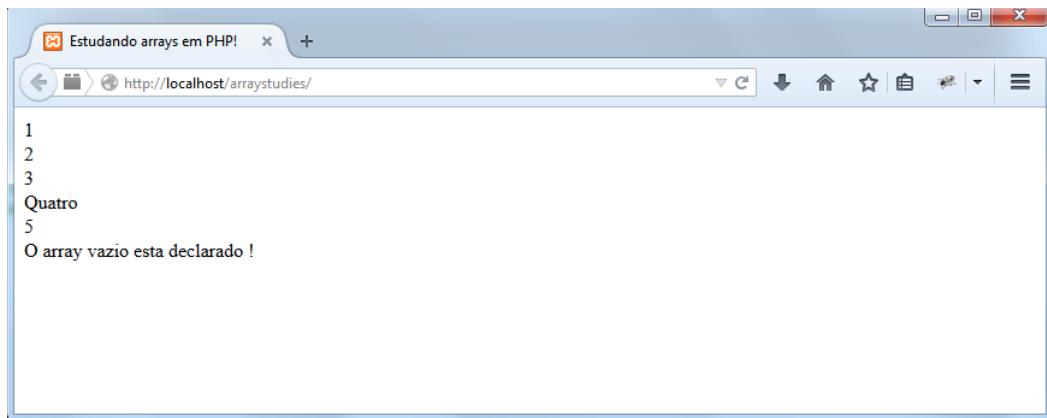
?>
</body>
</html>

```

4. Inicie o XAMPP

5. Em seu *browser*, acesse a URL <http://localhost:80/arraystudies/> e veja se está de acordo com a imagem a seguir:

FIGURA 37 - EXPLORANDO ARRAYS EM PHP



FONTE: O autor

5. Ao finalizar, não se esqueça de finalizar o XAMPP.

Agora vamos ver um comando muito importante do PHP, muito utilizado durante a depuração de uma página Web, o `print_r()`, que imprime o conteúdo de um *array*. Vamos lá, adicione o seguinte trecho de código PHP à página que você acabou de criar e verifique se ficou de acordo com a figura a seguir:

LISTAGEM 21 - IMPRIMINDO ARRAYS EM PHP

```
<h1>Imprimindo arrays com print_r</h1>
<?php
    print_r($meuPrimeiroArray);
    echo "<br />";
    print_r($arrayVazio);
?>
```

FIGURA 38 – IMPRIMINDO ARRAYS COM O PRINT_R EM PHP



FONTE: O autor

Agora vamos remover elementos de um *array*. No PHP, os elementos de um *array* são removidos através do comando `unset()`. O comando `unset()` recebe como parâmetro a posição exata do *array* do qual deve ser removida. Insira o conteúdo da listagem abaixo na sua página *arraystudies* para ver como funciona a remoção de itens de *arrays* em PHP, e verifique se ficou conforme a imagem a seguir:

LISTAGEM 22 - REMOVENDO ELEMENTOS DE ARRAYS EM PHP

```

<h1>Removendo elementos de arrays em PHP</h1>
<?php
    unset($meuPrimeiroArray[1]);
    print_r($meuPrimeiroArray);
    echo "<br />";
    if (isset($meuPrimeiroArray[1])){
        echo "Ainda está declarado!<br />";
    } else {
        echo "Foi destruído!<br />";
    }
?>

```

FIGURA 39 – ARRAY COM O SEGUNDO ELEMENTO REMOVIDO



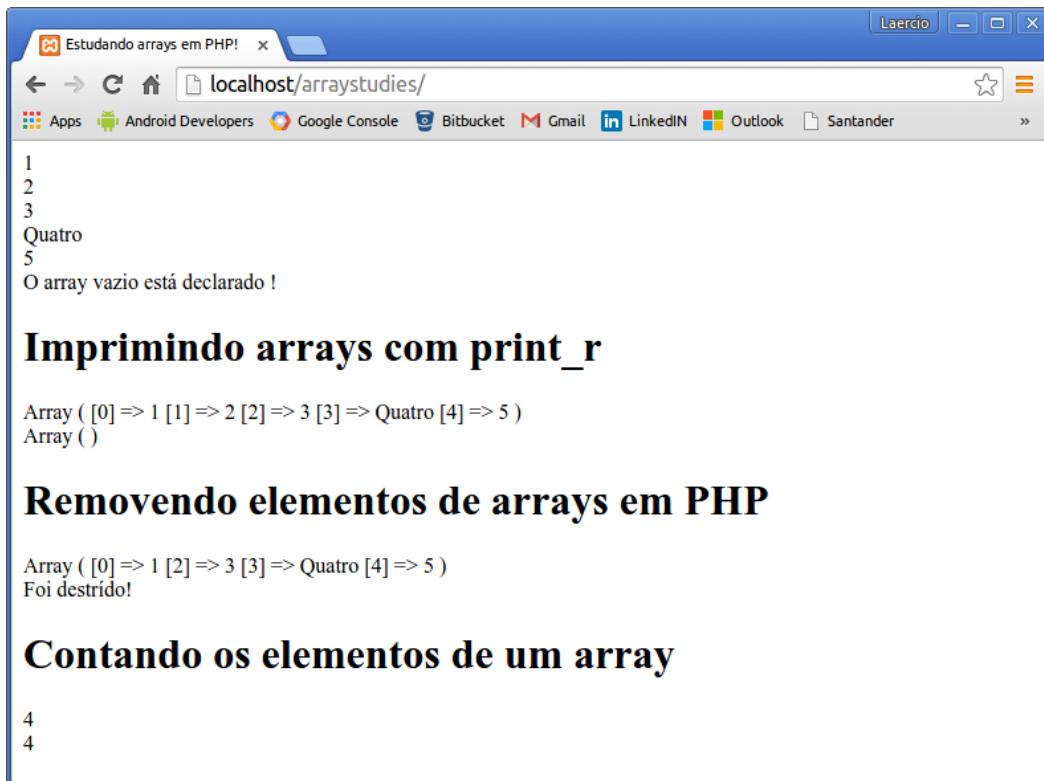
FONTE: O autor

Agora vamos ver como fazer para obter o número de elementos de um *array* em PHP. O PHP disponibiliza duas funções para esta finalidade, o `sizeof()` e `count()`. Vamos conferir? Em seu arquivo de testes para *strings* insira o código-fonte da listagem a seguir e confira se ficou de acordo com a imagem:

LISTAGEM 23 - OBTENDO A QUANTIDADE DE ELEMENTOS DE ARRAYS EM PHP

```
<h1>Contando os elementos de um array</h1>
<?php
    echo count($meuPrimeiroArray);
    echo "<br />";
    echo sizeof($meuPrimeiroArray);
    echo "<br />";
?>
```

FIGURA 40 – TESTANDO AS FUNÇÕES COUNT() E SIZEOF() DO PHP



FONTE: O autor

Para concluir nosso estudo sobre *arrays* em PHP, vamos começar uma nova página Web:

1. Em seu diretório de projetos (htdocs) localize o diretório *arraystudies* e crie dentro dele um arquivo chamado *continuando.php*.
2. Dentro do arquivo *continuando.php*, insira o código-fonte da listagem a seguir:

LISTAGEM 24 - PÁGINA PARA PROSSEGUIR OS ESTUDOS SOBRE ARRAYS

```

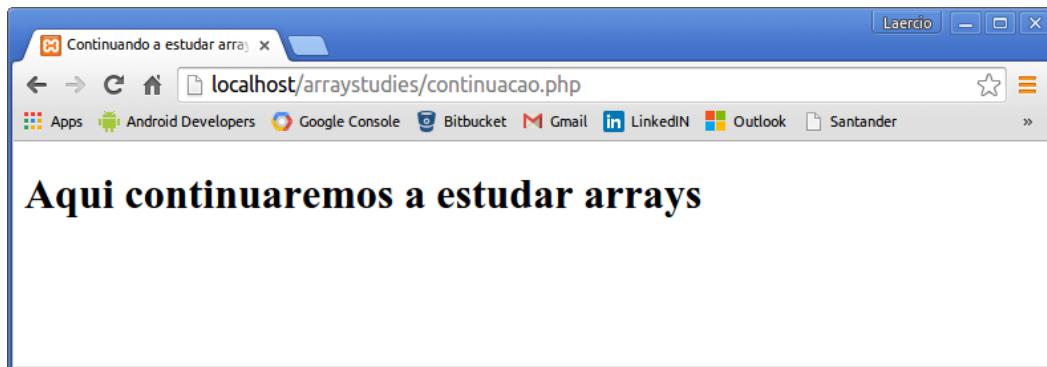
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Continuando a estudar arrays</title>
    </head>
    <body>
        <h1>Aqui continuaremos a estudar arrays</h1>
    </body>
</html>

```

3. Lembre-se que para executar os exemplos é necessário iniciar o XAMPP e, neste caso, acessar a URL <<http://localhost:80/arraystudies/continuacao.php>>, e que ao término, é necessário finalizar o XAMPP.

Se após ter iniciado o XAMPP e acessado a URL <<http://localhost:80/arraystudies/continuacao.php>> sua página estiver conforme a da figura abaixo, podemos continuar!

FIGURA 41 – PÁGINA WEB PARA CONTINUAR NOSSOS ESTUDOS SOBRE ARRAYS EM PHP



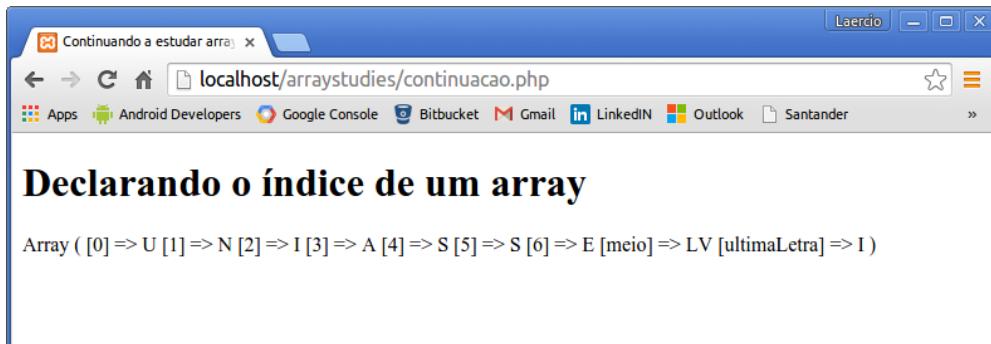
FONTE: O autor

No PHP é possível definir o índice de um elemento em um *array*. Os elementos sem índice declarado recebem um índice numérico definido automaticamente pelo PHP. Vamos conferir? Vamos criar um *array*, inicializá-lo com algumas letras, que receberão o índice numérico atribuído automaticamente pelo PHP e adicionaremos outros dois elementos no *array*, para os quais declararemos um índice no formato *string*. Substitua o texto 'Aqui continuaremos a estudar *arrays*' pelo código-fonte da listagem a seguir, e em seguida, confira se ficou semelhante ao da figura:

LISTAGEM 25 - ARRAYS COM ÍNDICES DECLARADOS EM PHP

```
<h1>Declarando o índice de um array</h1>
<?php
    $array = array("U", "N", "I", "A", "S", "S", "E");
    $array["meio"] = "LV";
    $array["ultimaLetra"] = "I";
    print_r($array);
?>
```

FIGURA 42 – ARRAYS COM ÍNDICES DECLARADOS EM PHP



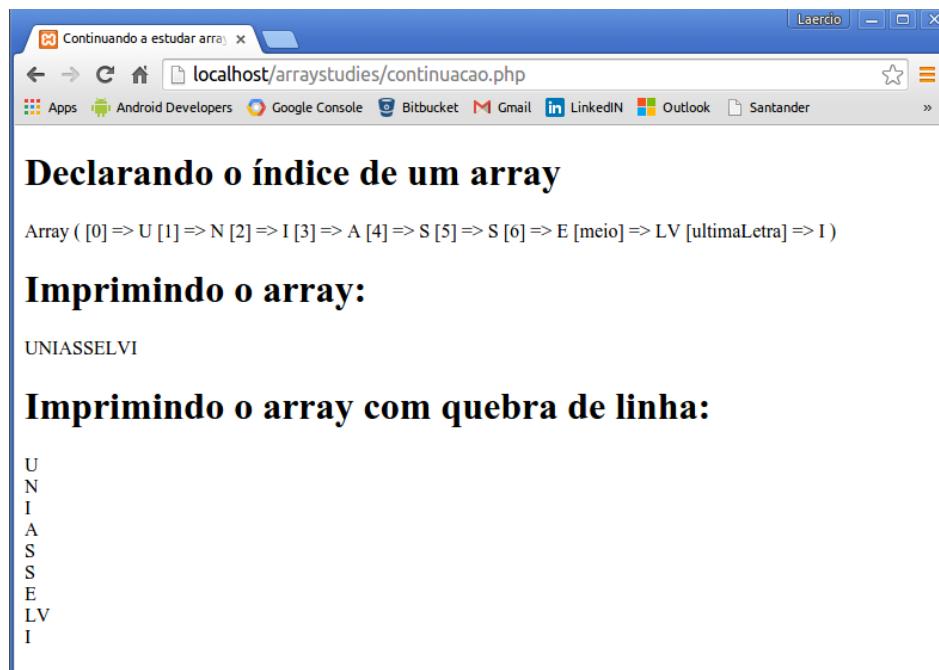
FONTE: O autor

Agora vamos navegar no conteúdo de um *array*. Para isto, o PHP oferece um recurso que facilita muito a implementação, o *for each*. O *for each* é um laço *for* que recebe um *array* como parâmetro e declara uma variável que receberá o valor da posição do *array* pelo qual ele está iterando. Pareceu complicado? Na verdade não é, vamos ver na prática. Adicione o código-fonte da listagem a seguir ao arquivo continuacao.php. Na seguida, confira o resultado com a imagem.

LISTAGEM 26 - PERCORRENDO OS ELEMENTOS DE UM ARRAY COM FOR EACH EM PHP

```
<h1>Imprimindo o array:</h1>
<?php
    foreach ($array as $valor) {
        echo $valor;
    }
?>
<h1>Imprimindo o array com quebra de linha:</h1>
<?php
    foreach ($array as $valor) {
        echo $valor."<br />";
    }
?>
```

FIGURA 43 – PERCORRENDO OS ELEMENTOS DE UM ARRAY COM FOR EACH EM PHP



FONTE: O autor

3.10 DECLARAÇÃO DE FUNÇÕES

Você deve se recordar da ocasião em que você aprendia a criar seus primeiros algoritmos (em qualquer linguagem de programação) e deparou-se com as inúmeras possibilidades que adquiriu quando aprendeu a declarar funções (ou métodos). Em PHP, uma função é declarada através do comando *function*, uma lista de parâmetros entre parênteses (opcional), um bloco de código (com ou sem o *return*), pois então, vamos explorar isto no PHP.

1. No seu diretório de projetos (htdocs), crie um diretório chamado *funcoes*.
2. Dentro do diretório *funcoes*, crie um arquivo chamado *index.php* e insira nele o código-fonte da listagem a seguir:

LISTAGEM 27 - DECLARANDO FUNÇÕES EM PHP

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Declarando funções</title>
    </head>
    <body>
```

```

<h1>Declarando funções</h1>
<?php

    // Declara uma função para somar
    // dois valores
    function soma($valor1, $valor2) {
        $resultado = $valor1 + $valor2;
        return $resultado;
    }
?>
<h2>Agora vamos usar a função soma com os valores
3 e 6!</h2>
<?
    $tres = 3;
    $seis = 6;
    echo soma($tres, $seis);
?>
</body>
</html>

```

3. Inicie o XAMPP e accese a URL <http://localhost:80/funcoes> e veja se está de acordo com a figura a seguir.

FIGURA 44 – DECLARANDO FUNÇÕES EM PHP



FONTE: O autor

4. Não se esqueça de finalizar o XAMPP quando tiver terminado.

3.1.1 SEPARANDO SEU CÓDIGO-FONTE EM MAIS DE UM ARQUIVO

Em projetos mais extensos, é muito interessante que haja a possibilidade de separarmos o código-fonte em vários arquivos. Tudo bem, mas se o código-fonte está em diversos arquivos distintos, como é possível utilizar uma função declarada em outro arquivo? No PHP, isto é feito através de dois comandos, *include* e *require*, que são responsáveis pela importação e execução do código-fonte contido no arquivo referenciado.

- *include()*: importa e executa o código-fonte passado como parâmetro, gerando alertas caso ocorra algum erro.
- *require()*: importa e executa o código-fonte passado como parâmetro, gerando erros caso ocorra algum erro.

Vamos conferir na prática? Vamos criar um pequeno projeto em PHP, que possui três arquivos PHP, sendo um utilizado para a definição de constantes, outro utilizado para declaração de funções, e o outro a página em si.

1. No seu diretório de projetos, crie um diretório chamado *multiplica*.
2. Dentro do diretório *multiplica*, crie um arquivo chamado *constantes.php* e insira o conteúdo da listagem abaixo (não é necessário escrever nenhum HTML):

LISTAGEM 28 - CÓDIGO-FONTE DO ARQUIVO *CONSTANTES.PHP*

```
<?php
    // Declara duas constantes numéricas
    define("dois", 2);
    define("quatro", 4);
?>
```

3. Também dentro do diretório *multiplica*, crie um arquivo chamado *funcoes.php* e insira o conteúdo da listagem abaixo (também não é necessário escrever nenhum HTML):

LISTAGEM 29 - CÓDIGO-FONTE DO ARQUIVO *funcoes.php*

```
<?php
    // Declara uma função para multiplicar dois
    números
        function multiplica($valor1, $valor2) {
            $resultado = $valor1 * $valor2;
            return $resultado;
        }
?>
```

4. Ainda no diretório multiplica, crie um arquivo chamado index.php e insira o código-fonte da listagem a seguir:

LISTAGEM 30 - CÓDIGO-FONTE DO ARQUIVO index.php DO PROJETO MULTIPLICA

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Projeto Multiplica</title>
    </head>
    <body>
        <h1>Multiplicando em PHP</h1>
        <?php

            // Referencia as constantes e funções contidas
            // em arquivos PHP externos
            require("constantes.php");
            include("funcoes.php");

            // Executa uma função de um arquivo externo
            echo multiplica(dois, quatro);
        ?>
    </body>
</html>
```

5. Inicie o XAMPP e accesse a URL <http://localhost:80/multiplica/> e verifique se está de acordo com a figura a seguir:

FIGURA 45 – PÁGINA WEB MULTIPLICA, UTILIZANDO MAIS DE UM ARQUIVO DE CÓDIGO-FONTE PHP



FONTE: O autor

LEITURA COMPLEMENTAR

O futuro da Internet (e do mundo) segundo o Google

Eric Schmidt, presidente do conselho administrativo do Google, e Jared Cohen, diretor de ideias da empresa, escreveram um livro em que fazem algumas previsões surpreendentes para o futuro. Veja quais são.

Por Reportagem: Anna Carolina Rodrigues Edição: Bruno Garattoni

Daqui a dez ou vinte anos, a internet será muito diferente do que é hoje. Mas como? Eric Schmidt, presidente do conselho administrativo do Google, e Jared Cohen, diretor de ideias da empresa, escreveram um livro em que tentam responder a essa pergunta: *The New Digital Age*, recentemente lançado nos EUA. Nele, fazem algumas previsões surpreendentes, e nem sempre otimistas, para o futuro. Veja quais são.

COMPORTAMENTO

1. O passado vai nos condenar

No mundo físico, você sempre pode mudar. Pode mudar de cidade, de aparência, de estilo, de profissão, de opinião. Na internet, não é assim: tudo o que você já fez ou disse fica gravado para sempre. Cada vez mais, usamos a rede para nos relacionar uns com os outros. Isso está gerando uma massa de dados tão grande, cobrindo tantos detalhes das nossas vidas, que no futuro será muito difícil de controlar – e poderá nos comprometer. “Nunca mais escreva [na internet] nada que você não queira ver estampado na capa de um jornal”, advertem Cohen e Schmidt.

A internet não esquece nada. E isso afetará a vida de todo mundo. Se uma criança chamar uma colega de “gorda” na rede, por exemplo, poderá manchar a própria reputação pelo resto da vida – pois todo mundo saberá que, um dia, ela praticou *bullying*. Inclusive potenciais empregadores, que poderão deixar de contratá-la. Uma foto, um comentário, um *post* infeliz poderá trazer consequências por muito tempo. “Os pais terão de conversar com os filhos sobre segurança e privacidade [*on-line*] antes mesmo de falar sobre sexo”, dizem os autores. Schmidt diz que a internet deveria ter um botão “delete”, que permitisse apagar para sempre eventuais erros que cometemos *on-line*. Isso é muito difícil, pois alguém sempre poderá ter copiado a informação que queremos ver sumir. Mas surgirão empresas especializadas em gerenciar a nossa reputação *on-line*, prometendo controlar ou eliminar informações de que não gostamos, e empresas de seguro virtual, que vão oferecer proteção contra roubo de identidade virtual e difamação na internet. “A identidade *on-line* será algo tão valioso que até surgirá um mercado negro, onde as pessoas poderão comprar identidades reais ou inventadas”, dizem os autores.

O fim do esquecimento terá consequências profundas – que, para o Google, incluirão até a escolha do nome das pessoas. Alguns casais batizarão seus filhos com nomes bem diferentes, que não sejam comuns, e registrarão esses nomes nas redes sociais antes mesmo do nascimento da criança, tudo para que ela se destaque. Outros preferirão nomes comuns e genéricos, do tipo “José Carlos”, que sejam muito frequentes e tornem mais difícil identificar a pessoa, permitindo que se esconda na multidão e mantenha algum grau de privacidade *on-line*. Hoje, esse tipo de coisa soa meio estranho.

No futuro, talvez não seja.

POLÍTICA

2. Haverá um ataque terrorista envolvendo a internet

O vírus Stuxnet, supostamente criado por Israel, foi usado para atacar o programa nuclear iraniano, e quase todas as semanas surge um novo caso de empresa ou universidade americana que teve seus computadores invadidos por *hackers* chineses. Ou seja: a guerra digital já é uma realidade. Ela tende a aumentar, tanto que o livro do Google fala no surgimento da Code War (guerra de códigos, em inglês), um conflito que envolveria vários países atacando as redes de computadores uns dos outros. Seria um conflito longo e cheio de pequenas sabotagens, sem declarações diretas de guerra, semelhante à Guerra Fria. “Os países vão fazer coisas *on-line* uns com os outros que seriam muito provocadoras [como sabotar usinas, espionar, derrubar o acesso à internet] de se fazer *off-line*. Isso vai permitir que os conflitos aconteçam no campo de batalha virtual, enquanto o resto permanece calmo.”

Mas o fato de a guerra ser digital não significa que ela não vá derramar sangue. Os executivos do Google imaginam um novo 11 de Setembro, que envolveria uma sequência de ações terroristas *on-line* e *off-line*. Um hacker poderia invadir o sistema de tráfego aéreo de algum país, por exemplo, e induzir os aviões a voarem na altitude errada – para que eles se choquem uns contra os outros. Aí, com a atenção mundial voltada para esse caos aéreo, viria a segunda fase do ataque: bombas posicionadas estratégicamente em Nova York, Chicago e em São Francisco explodiriam. Nas horas seguintes, uma nova onda de ataques virtuais atrapalharia a comunicação e a mobilização da polícia, dos bombeiros e ambulâncias. Em seguida, outro ataque poderia prejudicar os sistemas de distribuição de água, energia, óleo e gás do país. “No futuro, a força dos grupos terroristas não virá da disposição de morrer por uma causa, e sim do domínio tecnológico que eles possuírem”, preveem os autores.

3. O governo vai migrar para a web

Ir a uma repartição pública costuma ser uma experiência desagradável, cheia de burocracia e filas. Mas e se essa repartição fosse transformada num site – no qual você pudesse resolver todos os seus problemas? Eric Schmidt e

Jared Cohen propõem que o governo migre para a internet e seja capaz de funcionar por meio dela. Isso tornaria a operação mais eficiente, permitindo dar um atendimento melhor à população, e também seria uma vantagem em caso de desastres naturais. Se o prédio de um ministério fosse destruído por um terremoto, por exemplo, a instituição poderia continuar a funcionar *on-line*, com os funcionários se conectando de qualquer PC com acesso à internet.

4. A rede vai se fragmentar

A internet foi criada no final dos anos 60, para conectar as redes internas de universidades e instituições do governo americano. Ou seja: ela é, por definição, uma união de pequenas redes (daí seu nome, que significa “inter-rede”). É essa união que nos permite acessar qualquer *site*, de qualquer lugar do mundo, e foi ela a grande responsável pela universalização da internet. Mas, no futuro, não será assim. Com a desculpa de combater o terrorismo e os crimes *on-line*, e também por questões culturais, alguns países criarião suas próprias regras - e, na opinião do Google, isso acabará resultando em internets nacionais, com as características de cada lugar. E o que entra e sai de cada uma delas será monitorado, com direito a censura. Mais ou menos como já acontece em países como Irã e China – só que no mundo inteiro. Essa previsão pode parecer exagerada, mas tem certo respaldo no mundo real. Em março deste ano, o Parlamento Europeu discutiu uma lei que iria proibir o conteúdo pornográfico na internet (e acabou não sendo aprovada). É provável que, no futuro, os Estados tentem exercer algum controle sobre a internet.

Outra tendência, segundo Cohen e Schmidt, é a formação de alianças digitais entre países que tem costumes e opiniões semelhantes. Poderá surgir uma internet regional cobrindo vários países do Oriente Médio, por exemplo, com conteúdo e regras determinadas por eles. Em contrapartida, minorias ou insurgentes também poderão ter seu país *on-line*, como a criação de uma internet palestina, por exemplo. “O que começou como a *World Wide Web* começará a se parecer mais com o próprio mundo, cheio de divisões internas e interesses divergentes”, dizem os autores. Eles imaginam até a criação de uma espécie de visto, que controlaria quem pode ou não entrar na internet de cada país. “Isso poderia ser feito de forma rápida e eletronicamente, exigindo que os usuários se registrem e concordem com certas condições de acesso à internet de um país.”

SOCIEDADE

5. Um computador saberá tudo sobre você

Quer saber quais informações o Google tem a seu respeito? Acesse o *site* <google.com/dashboard> e você provavelmente irá se surpreender. São dezenas de informações, que incluem quais buscas você fez, quem são seus amigos, sua agenda de compromissos, seu endereço, onde você vai e todo o conteúdo dos seus *e-mails* e documentos. O Google já sabe muita coisa. Mas, no futuro, poderá saber ainda mais. Isso porque as informações que hoje ficam em bancos de dados

separados, como a sua identidade (RG), registros médicos e policiais e histórico de comunicações, serão unificadas em um único – e gigantesco – arquivo. Com apenas uma busca, será possível localizar todas as informações referentes à vida de uma pessoa. Algumas delas só poderiam ser acessadas com autorização judicial, mas sempre existe a possibilidade (e o receio) de que isso acabe sendo desrespeitado. Um exemplo recente: em maio, vazou na internet um documento no qual o FBI autoriza seus agentes a grampear os *e-mails* de qualquer pessoa, mesmo sem permissão de um juiz.

Lutar contra isso, e revelar poucas informações pessoais na internet, será visto como atitude suspeita. Cohen e Schmidt acreditam que o governo vá criar uma lista de “pessoas off-line”, gente que não posta nada nas redes sociais – e por isso supostamente tem algo a esconder. “Elas poderão ser submetidas a um conjunto de regras diferentes, como revista mais rigorosa no aeroporto ou até não poder viajar para determinados locais”, dizem.

6. Um grupo vai desvendar as mentiras da internet

É comum que os governos falsifiquem ou adulterem informações. Era assim na URSS (Stálin mandava apagar pessoas de fotos históricas) e é assim no Irã e na Coreia do Norte, que já foram pegos usando *photoshop* para manipular imagens militares. Por isso, os executivos do Google preveem a criação de uma entidade, independente de qualquer governo, que seria responsável pela fiscalização e investigação dos dados divulgados na internet, principalmente os que envolvessem política e conflitos armados. Uma espécie de Cruz Vermelha virtual, que teria representantes de vários países e funcionaria como referência para os órgãos de imprensa.

7. Mais pessoas terão (menos) poder

A internet permite que as pessoas se informem, se comuniquem e se organizem de forma livre e independente. Ou seja, ela dá poder às pessoas. Com o acesso a novas ideias, populações vão questionar mais seus líderes. Imagine o que acontecerá quando o habitante de uma tribo na África, por exemplo, descobrir que aquilo que o curandeiro local diz ser um mau espírito na verdade não passa de uma gripe. “Os governos autoritários vão perceber que suas populações serão mais difíceis de controlar e influenciar. E os Estados democráticos serão forçados a incluir mais vozes em suas decisões”, escrevem Jared Cohen e Eric Schmidt.

A Primavera Árabe é um bom exemplo disso. A internet teve um papel fundamental na organização dos grupos populares que derrubaram os governos de quatro países (Tunísia, Egito, Líbia e Iêmen) e abalaram vários outros. No caso egípcio, o próprio Google acabou sendo envolvido – pois Wael Ghonim, executivo da empresa no Egito, entrou por conta própria em mobilizações *on-line* (e ficou 11 dias preso por causa disso).

Na era da internet, minorias antes reprimidas também passam a ter uma voz. Mas, na opinião do Google, isso não terá necessariamente um grande efeito prático. É o chamado ativismo de sofá. A pessoa pode até curtir e compartilhar conteúdo relacionado a uma causa, mas, na hora de ir para as ruas, a coisa fica diferente. A mobilização virtual nem sempre se traduz em engajamento real. Além disso, a internet permite que os movimentos sociais surjam e cresçam muito rápido, de forma descentralizada e diluindo o poder entre muitas pessoas. Isso acaba fazendo com que esses movimentos tenham muitos líderes fracos, em vez de poucos líderes fortes.

Para sustentar essa tese, Cohen e Schmidt citam a Primavera Árabe, em que os regimes totalitários e os ditadores caíram, mas seu lugar acabou sendo tomado por governos muçulmanos, que não são particularmente democráticos, em vez de lideranças egressas da internet. “Sem estadistas, não haverá indivíduos qualificados o suficiente para levar um país adiante. Corre-se o risco de substituir uma forma de autocracia por outra”, dizem os autores. Em suma: a internet distribui o poder, mas isso não necessariamente resulta na formação de grandes líderes. Nelson Mandela não era uma celebridade de *Facebook*.

PARA SABER MAIS

The New Digital Age

Eric Schmidt e Jared Cohen Knopf, 2013. Disponível em: <<http://super.abril.com.br/tecnologia/futuro-internet-mundo-google-752917.shtml>>. Acesso em: 9 fev. 2014.

RESUMO DO TÓPICO 3

Neste tópico, você aprendeu que:

- O PHP nasceu das mãos de um desenvolvedor *Web* com a finalidade de facilitar o desenvolvimento de páginas *Web*. Conta com muitas características que tornam o desenvolvimento de páginas *Web* mais prático, e é muito utilizado hoje em dia.
- O PHP é uma plataforma gratuita e aberta.
- O PHP possui tipagem dinâmica.
- Tivemos um *overview* das principais funções do PHP para manipulação de *strings* e *arrays*.

AUTOATIVIDADE



- 1 Crie uma página PHP que declare um *array*, e que cada elemento do *array* corresponda a uma letra da palavra UNIASSELVI. Em seguida, imprima o valor do *array*, separado por linhas e indicando o índice da posição que está sendo imprimida. A página deve apresentar um conteúdo semelhante ao abaixo:

0: U
1: N
2: I
3: A
4: S
5: S
6: E
7: L
8: V
9: I

EXPLORANDO O MYSQL, O JAVASCRIPT E CONSTRUINDO SUA PRIMEIRA APLICAÇÃO WEB

OBJETIVOS DE APRENDIZAGEM

A partir desta unidade, você será capaz de:

- criar um banco de dados MySQL e tabelas utilizando o phpMyAdmin;
- conhecer o funcionamento dos comandos DML na linguagem SQL;
- criar Scripts poderosos com a linguagem JavaScript e adicioná-los a suas páginas Web;
- saber como o PHP, o MySQL e o JavaScript se combinam para formar uma aplicação Web.

PLANO DE ESTUDOS

Esta unidade de ensino está dividida em três tópicos, sendo que no final de cada um deles, você encontrará atividades que contribuirão para a apropriação dos conteúdos.

TÓPICO 1 – MYSQL

TÓPICO 2 – JAVASCRIPT

TÓPICO 3 – DESENVOLVENDO UMA APLICAÇÃO WEB COM PHP,
JAVASCRIPT E MYSQL

1 INTRODUÇÃO

A utilização de um banco de dados em um sistema *Web* é praticamente indispensável. Sem um banco de dados, o crescimento de um sistema *Web* ou *site* fica condenado, pois neste caso é necessário efetuar muitas manutenções no código-fonte de cada página (arquivo .html ou .php), apenas para alterar o conteúdo da página. No caso da plataforma que utilizamos para esta disciplina, o XAMPP, o banco de dados que temos à disposição é o MySQL. Vamos explorar o MySQL e aprender conceitos básicos de banco de dados ao longo deste tópico.

Bons estudos!

2 HISTÓRIA DO MYSQL

O MySQL é o banco de dados aberto mais conhecido no mundo. Foi desenvolvido por centenas de desenvolvedores do mundo inteiro, mais tarde foi comprado pela Sun Microsystems, que recentemente foi comprada pela Oracle. A Oracle ainda fornece a versão *Community*, que é a versão gratuita do MySQL. É muito popular no contexto da *web* por estar contido nas plataformas abertas mais conhecidas e utilizadas, o XAMPP, LAMP, E WAMP.

3 SQL: INSTRUÇÕES DDL E DML

Para utilizar um banco de dados é necessário ter domínio de uma linguagem de programação chamada SQL (*Structured Query Language*). O SQL possui um padrão, o ANSI, que nem todos os bancos de dados (MySQL, Oracle, SQL Server) adotam completamente.

A linguagem SQL é composta por comandos. Os comandos SQL dividem-se (principalmente) em dois grupos, o DDL (*Data Definition Language*) e o DML (*Data Manipulation Language*). A finalidade de cada um dos grupos é descrita a seguir:

- DDL: São os comandos que possibilitam a criação da estrutura do banco de dados. Com os comandos DDL é possível criar as estruturas que armazenarão de forma coerente os dados no banco de dados. A tradução literal do termo *Data Definition Language* do Inglês para o Português é Linguagem de Definição de Dados. O termo deixa bem evidente a finalidade dos comandos deste grupo.

- DML: São os comandos que possibilitam a manutenção dos dados armazenados no banco de dados. Sendo que tais dados estão abrigados nas estruturas criadas pelo desenvolvedor utilizando os comandos DDL. Através destes comandos é possível inserir, alterar, excluir e selecionar os dados do banco de dados. A tradução literal do termo *Data Manipulation Language* do inglês para o Português é Linguagem para Manipulação de Dados, reforça para que os comandos deste grupo são destinados.



Neste Caderno de Estudos, o tema banco de dados será abordado de forma bem resumida, pois você irá estudar (ou já estudou) os conceitos aqui apresentados na disciplina de Banco de Dados. Iremos focar somente aquilo que é essencial para a disciplina.

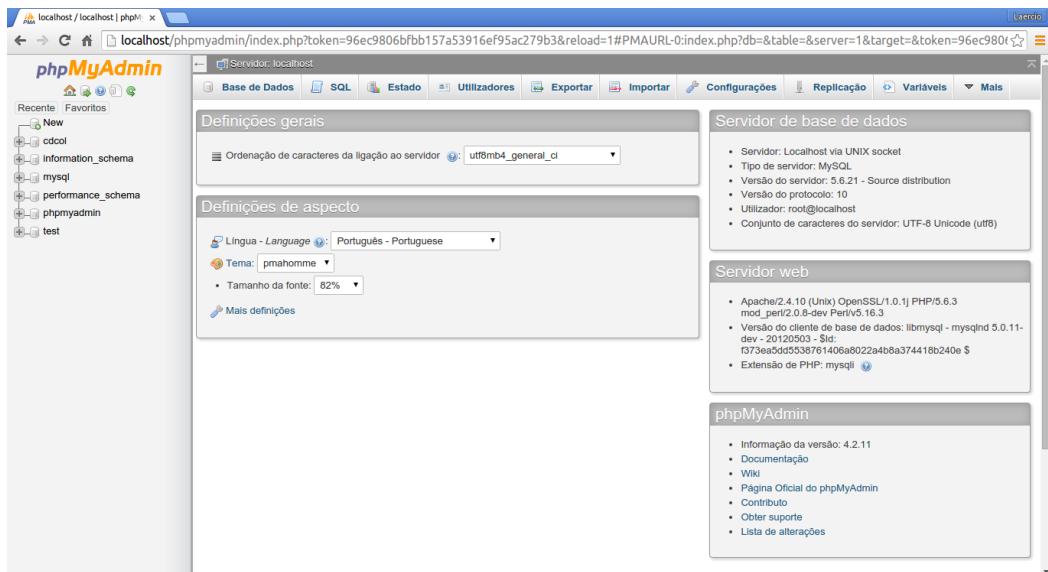
4 PHPMYADMIN

É hora de conhecermos a ferramenta que iremos utilizar para nosso desenvolvimento com MySQL, o phpMyAdmin. O phpMyAdmin é uma ferramenta que aumenta muito a produtividade no desenvolvimento de bases de dados, principalmente porque ele gera e executa os comandos DDL automaticamente conforme você vai utilizando.

Se você possui o XAMPP instalado em seu computador, você certamente já conta com o phpMyAdmin. Vamos conhecer o phpMyAdmin? Siga as seguintes etapas:

1. Inicie o XAMPP
2. Em seu *browser*, acesse a URL <<http://localhost:80/phpmyadmin/>>, você deve visualizar uma página semelhante à da figura a seguir:

FIGURA 46 - PHPMYADMIN



FONTE: O autor

3. Ao concluir, não se esqueça de finalizar o XAMPP.

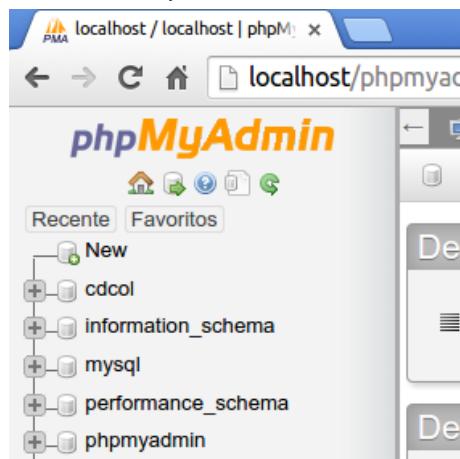
5 CRIANDO OBJETOS DE BANCO

Objetos de banco são as estruturas que nos permitem organizar a informação dentro do banco de dados. Por exemplo: tabelas, colunas de tabelas, usuários, sequências numéricas para geração de códigos, o próprio banco de dados etc. Todos os objetos de banco de dados são resultantes de comandos DDL bem-sucedidos (executados sem erros).

Nesta disciplina, faremos uso da ferramenta phpMyAdmin para criar nossos objetos de banco de dados, sendo que os comandos DDL serão gerados e executados automaticamente pela ferramenta. Vamos lá:

1. Inicie o XAMPP
2. Acesse o phpMyAdmin, ou seja, acesse a URL <<http://localhost:80/phpmyadmin/>>.
3. No phpMyAdmin encontre a listagem com o nome dos bancos de dados existentes e clique sobre a opção *New* para criar um novo banco de dados, conforme a figura a seguir:

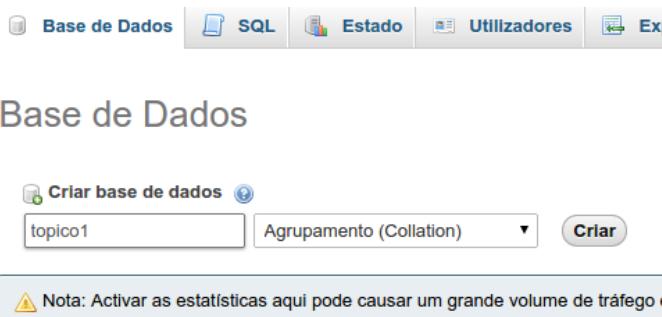
FIGURA 47 - OPÇÃO NEW NO PHPMYADMIN



FONTE: O autor

4. Na tela 'Base de dados' informe o nome do banco de dados, por exemplo 'topico1' (sem aspas), e clique em criar, conforme a figura a seguir:

FIGURA 48 - TELA PARA CRIAÇÃO DE BANCO DE DADOS NO PHPMYADMIN



FONTE: O autor

FIGURA 49 - MENSAGEM INDICANDO SUCESSO NA CRIAÇÃO DO BANCO DE DADOS



FONTE: O autor

5. Após ter clicado no botão Criar, a mensagem 'A base de dados topico1 foi criada' foi exibida. Localize-a na lista de nomes de bancos de dados, conforme a figura a seguir, e clique sobre ela, como nosso banco de dados ainda não possui tabelas, a janela Criar Tabela será exibida, informe 'pessoa' (sem aspas) no campo nome e 4 no campo número de colunas, em seguida clique em Executar:

FIGURA 50 - CRIANDO A TABELA PESSOA NO PHPMYADMIN



FONTE: O autor

6. Em seguida, você deverá definir as colunas para sua tabela, nossa tabela de pessoas terá as colunas CPF, Nome, RG e data de nascimento. Crie as colunas conforme a imagem a seguir e clique sobre o botão guarda:



Preste muita atenção ao fato de que a coluna CPF é um índice do tipo *PRIMARY*, e que as colunas RG e Data Nascimento são de preenchimento opcional, portanto possuem a opção Nulo checada.

FIGURA 51 - DEFININDO AS COLUNAS DA TABELA PESSOA NO PHPMYADMIN

The screenshot shows the phpMyAdmin interface for creating a table named 'pessoa'. The table structure is defined as follows:

Nome	Tipo	Tamanho/Valores*	Predefinido	Agrupamento (Collation)	Atributos	Nulo	Índice	A_I	Com.
cpf	VARCHAR	11	None					PRIMARY	
nome	VARCHAR	200	None						
rg	VARCHAR	20	None						
datanascimento	VARCHAR	12	None						

Below the table structure, there are fields for 'Comentários da tabela:', 'Motor de armazenamento:' (set to InnoDB), and 'Collation:'.

FONTE: O autor

7. Agora já é possível ver seu banco de dados `topico1` com a tabela que você criou, `pessoa`. Vamos examinar o que exatamente fizemos? Você já pode finalizar o XAMPP, caso não queira mais continuar explorando.

Você pode não ter percebido, mas acabou com vários objetos de banco. Entre os principais, um banco de dados, uma tabela e uma chave primária. O que exatamente significa isto?

- Banco de dados: onde são criados os objetos para armazenar os dados.
- Tabela: São estruturas que armazenam dados na forma de registros. Podem possuir diversas colunas.
- Chave Primária: É a identificação de cada registro na tabela. Em nosso exemplo, definimos a coluna CPF como chave primária da tabela pessoa, isto significa que não poderá haver CPFs repetidos na tabela pessoa.

Os comandos DDL executados para criar o banco de dados, a tabela e a chave primária são respectivamente `CREATE DATABASE`, `CREATE TABLE` e `ALTER TABLE ADD CONSTRAINT [...] PRIMARY`, e foram executados automaticamente pela ferramenta phpMyAdmin. Não iremos nos focar nestes comandos, pois para fins desta disciplina, basta saber que é possível criar objetos de banco de dados no MySQL utilizando o phpMyAdmin, que gera e executa comandos DML automaticamente.

6 MANIPULANDO OS DADOS COM OS COMANDOS DML

Os comandos DML são utilizados para efetuar seleção e/ou manutenção nos dados do banco de dados. São quatro os principais comandos DML: *SELECT*, *INSERT*, *UPDATE* e *DELETE*, que realizam respectivamente operações de seleção, inserção, atualização e exclusão de registros. Agora é hora de explorar estes comandos, pois eles farão parte do código-fonte de seus projetos *web* constantemente.

Vamos começar pelo comando *INSERT*. O comando *INSERT* é utilizado para inserir dados nas tabelas do banco de dados. A sintaxe do comando *INSERT* não é complexa, acompanhe no esquema explicativo a seguir:

- Deve-se começar escrevendo a cláusula *INSERT INTO*, seguido do nome da tabela.
- Após, deve-se informar entre parênteses e separado por vírgula o nome das colunas para as quais você possui os valores do novo registro.
- Então escreve-se a cláusula *VALUES*.
- E para finalizar, também entre parênteses e separado por vírgula, os valores respectivos às colunas, e um ponto e vírgula para finalizar o comando.

LISTAGEM 31 - SINTAXE DO COMANDO *INSERT*

```
INSERT INTO nome_tabela ( coluna_1
                          , coluna_2
                          , coluna_3)
    VALUES ( valor_1
              , valor_2
              , valor_3);
```

Veja o exemplo da listagem a seguir:

LISTAGEM 32 - *INSERT* NA TABELA PESSOA

```
INSERT INTO pessoa ( cpf
                      , nome
                      , rg
                      , datanascimento)
    VALUES ( '43807548767'
              , 'Luiz de Souza'
              , '67834'
              , '1989-12-04');
```



Ansioso para executar estes comandos? Calma, mais adiante iremos testá-los no phpMyAdmin!

Agora vamos explorar o comando *SELECT*, também conhecido como *query* (consulta em inglês). O comando *SELECT* tem a função de retornar dados das tabelas de um banco de dados. Dependendo do que precisa ser implementado, algumas *queries* podem ficar bem complexas. Mas nesta disciplina iremos focar somente na sintaxe básica do comando *SELECT*, acompanhe no esquema explicativo abaixo:

- A *query* inicia com a cláusula *SELECT* seguida dos nomes das colunas separados por vírgula.
- Então adiciona-se a cláusula *FROM*, e os nomes das tabelas que contém o resultado esperado, se for mais de uma tabela, estes nomes também são separados por vírgula.
- Por fim, opcionalmente, adiciona-se a cláusula *WHERE*, que tem por função estabelecer a ligação entre as tabelas da cláusula *FROM* e/ou adicionar filtros a *query*, para obter um resultado mais específico.

LISTAGEM 33 - SINTAXE BÁSICA DO COMANDO SELECT

```
SELECT coluna_1
      , coluna_2
      , coluna_3
  FROM tabela_1
 WHERE coluna_1 = 30;
```

Confira o exemplo da listagem a seguir:

LISTAGEM 34 - SELECT NA TABELA PESSOA

```
SELECT nome
      , rg
      , datanascimento
  FROM pessoa
 WHERE cpf = '43807548767'
```

Partimos agora para o comando *UPDATE*, cuja função é atualizar os dados do banco de dados. Acompanhe a explicação a seguir sobre a sintaxe básica do comando *UPDATE*:

- O comando *UPDATE* inicia com a cláusula *UPDATE* seguida do nome da tabela que contém os registros a serem atualizados.
- Em seguida vem a cláusula *SET* seguida do nome das colunas a serem atualizadas e seus novos valores. Cada nome de coluna é separado de seu novo valor por um sinal de atribuição (=), e se houver mais de uma coluna a ser atualizada, cada nome de coluna/valor deve ser separado por uma vírgula.
- Por último temos a cláusula *WHERE*, na qual deve constar uma expressão para atingir somente os registros necessários. E um ponto e vírgula para finalizar o comando.

LISTAGEM 35 - SINTAXE DO COMANDO *UPDATE*

```
UPDATE tabela
    SET coluna_2 = 'valor2'
        , coluna_3 = 'valor3'
        , coluna_4 = 4
WHERE coluna_1 = 'valor1';
```

LISTAGEM 36 - *UPDATE* NA TABELA PESSOA

```
UPDATE pessoa
    SET nome = 'João'
WHERE cpf = '43807548767'
```

Agora partiremos para o último comando DML importante para seus estudos nesta disciplina, o *DELETE*. O comando *DELETE* é utilizado para excluir registros em tabelas. Assim como fizemos com os comandos *INSERT*, *SELECT* e *UPDATE*, vamos analisar a sintaxe do comando *DELETE*.

- O comando *DELETE* inicia com a cláusula *DELETE FROM* seguida do nome da tabela que contém os registros que você deseja excluir.
- Em seguida, temos a cláusula *WHERE* na qual devemos adicionar uma expressão para filtrar somente os registros que desejamos excluir.

LISTAGEM 37 - SINTAXE DO COMANDO *DELETE*

```
DELETE FROM tabela
WHERE coluna_1 = 'M'
```

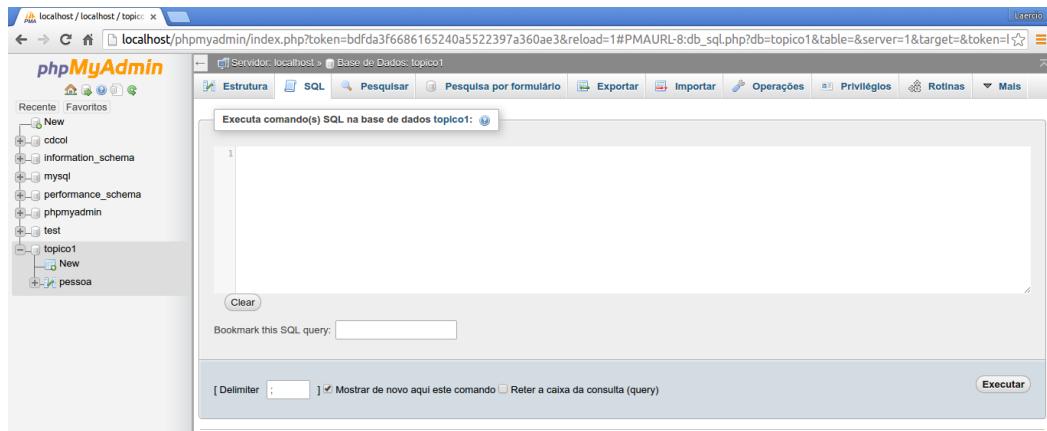
LISTAGEM 38 - DELETE NA TABELA PESSOA

```
DELETE FROM pessoa
WHERE cpf = '43807548767'
```

Quanta teoria! É hora de partirmos para a prática, pois no final das contas, é isto o que vai fazer a diferença. Vamos executar os comandos que vimos no phpMyAdmin, acompanhe conosco as etapas a seguir:

1. Inicie o XAMPP
2. Acesse o phpMyAdmin (ou seja, em seu *browser*, acesse a URL <http://localhost:80/phpmyadmin/>).
3. No phpMyAdmin, acesse o banco de dados 'topico1' (sem aspas), para fazer isto, localize-o na listagem de nomes de banco de dados e clique sobre ele, e acesse a guia SQL, conforme a figura a seguir:

FIGURA 52 - ACESSANDO O BANCO DE DADOS TOPICO1



FONTE: O autor

4. Digite o comando da listagem 4 no campo de texto da tela de execução de comandos, retirando a cláusula *WHERE* e suas expressões, e clique no botão executar, você poderá ver, conforme a Figura 9, que uma mensagem é exibida informando que não há registros na tabela. Isto é coerente, pois ainda não inserimos registro na tabela.

FIGURA 53 - COMANDO SELECT DIGITADO NA TELA DE EXECUÇÃO DE COMANDOS DO PHPMYADMIN

The screenshot shows the PHPMyAdmin interface with the 'SQL' tab selected. In the main query editor area, the following SQL code is entered:

```

1 SELECT nome
2      , rg
3      , datanascimento
4 FROM pessoa

```

Below the query, there are several buttons: 'Clear', 'Bookmark this SQL query:', '[Delimiter :]', 'Mostrar de novo aqui este comando' (checkbox checked), 'Reter a caixa da consulta (query)' (checkbox unchecked), and 'Executar' (execute button). A status bar at the bottom indicates: 'MySQL não retornou nenhum registro. (A consulta demorou 0.0003 segundos.)'.

FONTE: O autor

FIGURA 54 - RESULTADO DA EXECUÇÃO DA QUERY NA TABELA PESSOA

The screenshot shows the PHPMyAdmin interface with the 'SQL' tab selected. The query entered is the same as in Figure 53. Below the query, the status bar displays: 'MySQL não retornou nenhum registro. (A consulta demorou 0.0003 segundos.)'. At the bottom of the screen, there is a message: 'Mostrar Caixa do query'.

FONTE: O autor

Executar uma *query* sobre uma tabela sem a cláusula *where* irá retornar todos os registros existentes na tabela.

5. Agora clique novamente na guia SQL e na nova janela para execução de comandos que será aberta, insira o comando constante na listagem 2 e clique no botão executar, conforme as figuras a seguir.

FIGURA 55 - COMANDO INSERT PARA SER EXECUTADO NO PHPMYADMIN

The screenshot shows the MySQL Workbench interface with the SQL tab selected. A query window titled "Executa comando(s) SQL na base de dados topo1:" contains the following SQL code:

```

1 INSERT INTO pessoa ( cpf , nome , rg , datanascimento )
2 VALUES ( '43807548767' ,
3          'Luiz de Souza' ,
4          '67834' ,
5          '1989-12-04' );

```

Below the query window, there are buttons for "Clear", "Bookmark this SQL query:", and "Delimitador" (with a colon input field). To the right, there is a "Mostrar de novo aqui este comando" checkbox, a "Reter a caixa da consulta (query)" checkbox, and an "Executar" button.

FONTE: O autor

FIGURA 56 - EXECUÇÃO DO COMANDO INSERT NO PHPMYADMIN

The screenshot shows the MySQL Workbench interface with the SQL tab selected. A success message in a green bar states: "✓ 1 linha inserida. (A consulta demorou 0.0406 segundos.)". Below it, the executed SQL query is shown:

```
INSERT INTO pessoa ( cpf , nome , rg , datanascimento ) VALUES ( '43807548767' , 'Luiz de Souza' , '67834' , '1989-12-04' )
```

At the bottom, there are links "[Na linha]", "[Edita]", and "[Criar código PHP]".

FONTE: O autor

6. Clique novamente sobre a guia SQL e digite a *query* constante na listagem 4. Em seguida clique sobre o botão executar.

FIGURA 57 - QUERY SOBRE A TABELA DE PESSOAS NO PHPMYADMIN

The screenshot shows the MySQL Workbench interface with the SQL tab selected. A query window titled "Executa comando(s) SQL na base de dados topo1:" contains the following SQL code:

```

1 SELECT nome
2      , rg
3      , datanascimento
4   FROM pessoa
5 WHERE cpf = '43807548767';

```

Below the query window, there are buttons for "Clear", "Bookmark this SQL query:", and "Delimitador" (with a colon input field). To the right, there is a "Mostrar de novo aqui este comando" checkbox, a "Reter a caixa da consulta (query)" checkbox, and an "Executar" button.

FONTE: O autor

FIGURA 58 - RESULTADO DA EXECUÇÃO DA QUERY SOBRE A TABELA DE PESSOAS

The screenshot shows the phpMyAdmin interface for the 'topico1' database. The 'SQL' tab is selected. A message at the top states: "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available." Below this, a green bar indicates: "A mostrar registos de 0 - 0 (1 total, A consulta demorou 0.0004 segundos.)". The SQL query entered is: "SELECT nome , rg , datanascimento FROM pessoa WHERE cpf = '43887548767'". At the bottom, there are buttons for [Na linha], [Edita], and [Criar código PHP]. Below the query area, there are two tables showing the results of the query.

+ Opções		
nome	rg	datanascimento
Luiz de Souza	67834	1989-12-04

+ Opções		
nome	rg	datanascimento
Luiz de Souza	67834	1989-12-04

FONTE: O autor

7. Clique novamente sobre a guia SQL do phpMyAdmin e no campo destinado ao comando digite desta vez o comando *update* encontrado na listagem 36 e clique sobre o botão executar. Em seguida, execute novamente a *query* que consta na listagem 34. Após executar a query você vai perceber que você atualizou o campo nome, e o resultado da *query* já está atualizado.

FIGURA 59 - COMANDO UPDATE A SER EXECUTADO NO PHPMYADMIN

The screenshot shows the phpMyAdmin interface for the 'topico1' database. The 'SQL' tab is selected. In the main area, the SQL command is: "UPDATE pessoa SET nome = 'João' WHERE cpf = '43887548767';". Below the command, there is a 'Clear' button and a 'Bookmark this SQL query:' input field. At the bottom, there are buttons for '[Delimiter]', '[Mostrar de novo aqui este comando]', '[Retirar a caixa da consulta (query)]', and 'Executar'.

FONTE: O autor

FIGURA 60 - RESULTADO DA EXECUÇÃO DO COMANDO UPDATE NO PHPMYADMIN

The screenshot shows the phpMyAdmin interface with the SQL tab selected. A success message at the top states "1 linha afectada. (A consulta demorou 0.0425 segundos.)". Below it, the executed SQL command is shown: "UPDATE pessoa SET nome = 'João' WHERE cpf = '43807548767'". At the bottom right, there are links "[Na linha]", "[Edita]", and "[Criar código PHP]".

FONTE: O autor

FIGURA 61 - RESULTADO DA QUERY APÓS O UPDATE

The screenshot shows the phpMyAdmin interface with the SQL tab selected. A warning message at the top says "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.". Below it, a success message states "mostrar registos de 0 - 0 (1 total). A consulta demorou 0.0006 segundos.". The executed SQL command is "SELECT nome , rg , datanascimento FROM pessoa WHERE cpf = '43807548767'". At the bottom right, there are links "[Na linha]", "[Edita]", and "[Criar código PHP]".

+ Opções	nome	rg	datanascimento
	João	67834	1989-12-04

FONTE: O autor

8. Agora vamos explorar o comando *delete*, responsável por excluir registros das tabelas do banco de dados. No phpMyAdmin, clique na guia SQL e execute o comando *delete* contido na listagem 8 e então clique em Executar. O registro será excluído da tabela pessoa. Após feito isto, execute novamente a *query* da listagem 34, e confirme que o registro não existe mais.

FIGURA 62 - COMANDO DELETE A SER EXECUTADO NO PHPMYADMIN

The screenshot shows the phpMyAdmin interface with the SQL tab selected. In the SQL editor, there are two lines of code: "DELETE FROM pessoa" and "WHERE cpf = '43807548767'". Below the editor, there is a "Clear" button and a "Bookmark this SQL query:" input field. At the bottom, there are options "[Delimitador]", "[Executar]", and checkboxes for "Mostrar de novo aqui este comando" and "Reter a caixa da consulta (query)".

FONTE: O autor

FIGURA 63 - RESULTADO DA EXECUÇÃO DO COMANDO DELETE

The screenshot shows the MySQL Workbench interface. The top menu bar includes 'Estrutura', 'SQL', 'Pesquisar', 'Pesquisa por formulário', 'Exportar', 'Importar', 'Operações', 'Privilégios', 'Rotinas', and 'Mais'. The 'SQL' tab is selected. A green message box at the top states '1 linhas excluída. (A consulta demorou 0.0557 segundos.)'. Below it, the SQL query is displayed: 'DELETE FROM pessoa WHERE cpf = '43807548767''. At the bottom right of the main area are links '[Na linha]', '[Edita]', and '[Criar código PHP]'. A small note 'Mostrar Caixa do query' is visible at the bottom left.

FONTE: O autor

FIGURA 64 - QUERY EXECUTADA APÓS O COMANDO DELETE

The screenshot shows the MySQL Workbench interface. The top menu bar includes 'Estrutura', 'SQL', 'Pesquisar', 'Pesquisa por formulário', 'Exportar', 'Importar', 'Operações', 'Privilégios', 'Rotinas', and 'Mais'. The 'SQL' tab is selected. A green message box at the top states 'MySQL não retornou nenhum registo. (A consulta demorou 0.0005 segundos.)'. Below it, the SQL query is displayed: 'SELECT nome , rg , datanascimento FROM pessoa WHERE cpf = '43807548767''. At the bottom right of the main area are links '[Na linha]', '[Edita]', and '[Criar código PHP]'. A small note 'Mostrar Caixa do query' is visible at the bottom left.

FONTE: O autor

RESUMO DO TÓPICO 1

Neste tópico você aprendeu que:

- A plataforma XAMPP disponibiliza um banco de dados: o MySQL.
- Para utilizar um banco de dados é necessário conhecer a linguagem SQL (*Structured Query Language*).
- A linguagem SQL se divide em dois grupos principais, os comandos destinados a definição de dados (DDL – *Data Definition Language*) e os comandos destinados a manipulação de dados (DML – *Data Manipulation Language*).
- O phpMyAdmin é a ferramenta de administração de banco de dados MySQL disponibilizada no XAMPP, e permite criação dos objetos do banco de dados sem a necessidade de escrever os comandos DDL.
- Tivemos um *overview* dos comandos DML básicos.

AUTOATIVIDADE



- 1 Na tabela pessoa, criada ao longo dos exemplos anteriores, insira um registro com as suas informações pessoais.
- 2 Insira mais um registro com informações pessoais referentes a algum familiar próximo (namorado(a), pai, mãe).
- 3 Implemente uma *query* para obter somente as informações pessoais referentes ao seu familiar.
- 4 Implemente uma *query* para obter todas as informações da tabela pessoa.
- 5 Exclua o registro referente as suas informações pessoais.

1 INTRODUÇÃO

As páginas *Web* da atualidade exigem um nível de usabilidade, ergonomia e dinâmica muito maior do que a anos atrás. Se você é um internauta de longa data, deve lembrar que para cada *link* que você clicava uma página completamente nova era carregada, e no tempo da internet discada, isto levava alguns segundos (até mesmo minutos), ficávamos acompanhando na barra de *status* do *browser* o progresso do carregamento da página. Relembrar é viver, mas vamos deixar a nostalgia de lado e analisar. Hoje você deve provavelmente navegar sem sofrer tanto com os carregamentos de páginas, isto se deve ao fato de que as páginas *Web* carregam em si programas que são executados pelo *browser*, sem necessidade de requisição ao servidor *Web* nem recarregamento da página. Tais programas são desenvolvidos em JavaScript (em sua maioria) e vamos aprender o desenvolvimento desta linguagem fantástica ao longo deste tópico.

2 UM POUCO DE HISTÓRIA

JavaScript nasceu de um antigo navegador chamado *Netscape*. Em plena era de guerra dos navegadores, uma disputa severa entre a *Netscape* (empresa desenvolvedora do *Netscape*) e a *Microsoft* (empresa desenvolvedora do *Internet Explorer*). A linguagem ganhou originalmente outros nomes, como *Mocha* e *LiveScript*, mas a *Sun Microsystems*, empresa desenvolvedora do Java (na época, pois atualmente é a Oracle que mantém o Java), resolveu também entrar na guerra, a favor do JavaScript e forneceu o nome Java para batizar a linguagem por um motivo de *marketing*. A linguagem Java já era bem difundida na época, e batizando o LiveScript como JavaScript, acreditava-se que linguagem despertaria interesse dos desenvolvedores Java. Pode ser (ou não) resultado disto, mas a realidade é que o JavaScript hoje tem uma popularidade inabalável. É implementado em todos os navegadores. Quem ganhou a guerra dos navegadores? Isto é subjetivo, mas a *Microsoft* teve que incorporar o JavaScript à internet *Explorer* para satisfazer os desenvolvedores.

3 HELLO WORLD

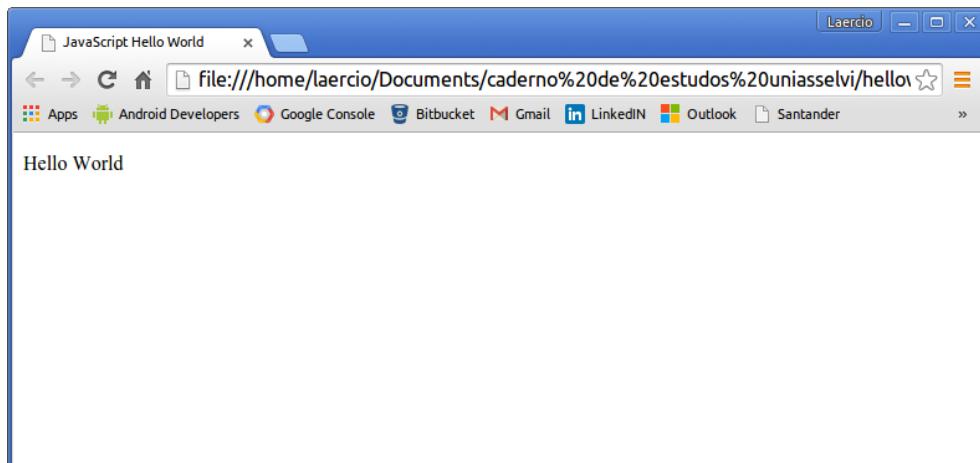
Nós programadores temos a crença de que precisamos começar a conhecer novas linguagens de programação através da implementação do programa *Hello World*. Vamos fazer nosso *Hello World* em JavaScript, siga as etapas a seguir:

1. Em seu computador, em qualquer diretório, crie um arquivo chamado helloworld.html.
2. Abra seu arquivo helloworld.html com seu editor de texto, e insira nele o código-fonte da listagem a seguir:

LISTAGEM 39 - *HELLO WORLD EM JAVASCRIPT*

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>JavaScript Hello World</title>
    <script type="text/javascript">
      // Este é o nosso Hello World
      document.write("<p>Hello World</p>");
    </script>
  </head>
  <body>
  </body>
</html>
```

3. Abra seu arquivo helloworld.html em seu *browser*, verifique se está de acordo com a figura a seguir:

FIGURA 65 - *HELLO WORLD EM JAVASCRIPT*

FONTE: O autor

Agora, vamos analisar o que exatamente fizemos. Vamos focar somente no conteúdo JavaScript, pois o restante é uma cópia do nosso *HelloWorld* em HTML5, e dispensa explicações. Os *scripts* JavaScript podem ficar na página, entre a *tag script* e seu fechamento. O trecho *document.write* é uma instrução que diz para escrever no documento HTML o texto que foi passado por parâmetro. Veja que foi bem intuitivo, *document.write* diz para escrever no documento. Em JavaScript, os comandos podem ou não ser delimitados por ponto e vírgula, mas se você utilizar o ponto e vírgula em todos os comandos, seu código fica muito mais legível, portanto adote esta prática. Os comentários são exatamente iguais ao PHP, ou seja, comentários de uma linha começam com duas barras (//) e comentários com mais de uma linha iniciam com barra asterisco e terminam com asterisco barra (* e */ respectivamente).



É muito recomendado que você separe seu código-fonte JavaScript do HTML, exatamente como se faz com o CSS. Para fins de exemplificação, neste tópico iremos manter o código JavaScript junto com o código HTML, para fornecer exemplos completos e diminuir a complexidade do aprendizado de JavaScript. Na próxima unidade, você irá conhecer o framework AngularJS, que além de agregar muitas outras vantagens, cuida da separação do código-fonte JavaScript e HTML da melhor forma possível.

4 TIPOS DE DADOS

A linguagem JavaScript, assim como o PHP, conta com o recurso de tipagem dinâmica, ou seja, ao declarar uma variável não é necessário especificar o tipo de dado. JavaScript conta com os tipos de dados a seguir:

- *Number*: para efetuar operações aritméticas. Pode ser ou não número com ponto flutuante.
- *Boolean*: Expressões com valor verdadeiro ou falso (*true* e *false*, respectivamente). Em JavaScript, os valores booleanos também podem ser convertidos para numéricos, sendo verdadeiro igual a 1 e falso igual a 0.
- *String*: Texto.
- *Object*: Objetos, porém JavaScript possui um modelo de orientação a objetos bem particular, muito diferente do modelo encontrado em linguagens puramente orientadas a objetos, como C++, Java e C#. Se você estiver interessado em aprender sobre o modelo de orientação a objetos do Javascript, pesquise sobre orientação a objetos baseada em protótipo após completar seus estudos deste tópico.
- *Array*: Coleções de elementos de quaisquer tipos de dados.

5 DECLARAÇÃO DE VARIÁVEIS

Declarar variáveis em JavaScript não tem muito mistério, porém é necessário estar atento a algumas particularidades. Para criar uma variável utiliza-se a instrução 'var' (sem aspas) seguido do identificador e opcionalmente uma inicialização. Para criar seu identificador, são aceitas letras maiúsculas e minúsculas, *underlines* (_), cifrão (\$) e números, porém, o primeiro caractere não deve ser um número. Há somente dois escopos para variáveis, o escopo global e o escopo de função. As variáveis de escopo de função são aquelas que são declaradas dentro de funções (Veremos como declarar funções mais adiante). A principal particularidade está no fato de que, ao contrário das demais linguagens de programação, os blocos anônimos de código (no caso do JavaScript {}) não influenciam no escopo das variáveis.

1. Em seu computador, crie um arquivo chamado variaveis.html.
2. Edite o arquivo variaveis.html, insira o código-fonte abaixo e salve-o.

LISTAGEM 40 - DECLARANDO VARIÁVEIS EM JAVASCRIPT

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8">
    <title>Variáveis em JavaScript</title>
    <script type="text/javascript">

      /*
       * Em JavaScript a inicialização de
variáveis é opcional
       * Mas vamos inicializar todas para
evitar problemas
       * Declarou, Inicializou :D
      */

      document.write("<p>");

      var variavelString = "Esta variável
armazena texto";
      document.write("Variável String: " +
variavelString +
          "<br />");
      // Resultado: Variável String: Esta
variável armazena
      //texto
```

```

var variavelNumerica = 10;
document.write("Variável Numérica: " +
variavelNumerica
    + "<br />");
// Resultado: Variável Numérica: 10

{
    var variavelDeBlocoAnonimo = "Esta
variável foi "
        + "declarada em um bloco anônimo";
}
document.write("A variável do bloco
anônimo permanece "
    + "declarada mesmo depois do
bloco: "
        + variavelDeBlocoAnonimo);
// Resultado: A variável do bloco anônimo
permanece
// declarada mesmo depois do bloco:
// Esta variável foi declarada em um
bloco anônimo

document.write("</p>");

</script>
</head>
<body>
</body>
</html>
```

3. Visualize a página Web variáveis.html em seu *browser*



As mesmas regras para criação de identificadores para variáveis também se aplicam aos demais identificadores que você pode criar em JavaScript, como funções, rótulos e objetos.

6 OPERADORES LÓGICOS

O JavaScript possui basicamente os mesmos operadores lógicos encontrados no PHP. Operadores lógicos são operadores aplicados a um ou mais valores (geralmente um à esquerda e um à direita do operador) e retornam um valor booleano (*true* ou *false*). Abaixo listamos os principais, e o significado de cada um, apresentando o nome do operador e o operador entre parênteses:

- Igualdade (==): testa se dois valores são iguais. Não considera o tipo de dado, uma comparação de *string* com *number* pode retornar *true* se as variáveis possuírem valores com o mesmo significado, por exemplo, a expressão `1 == "1"` retorna *true*.
- Identidade (===): testa se dois valores são idênticos. Para que dois valores sejam idênticos, eles precisam ser do mesmo tipo de dado e possuírem o mesmo valor. Por exemplo, a expressão `1 === "1"` retorna *false*, pois apesar de ambos os valores representarem o algarismo um (1), o do lado esquerdo é um *number*, e o do lado direito é uma *string*.
- Desigualdade (!=): testa se dois valores são diferentes. Assim como o operador de igualdade, não considera o tipo de dado, apenas testa o conteúdo de seus operandos. Por exemplo, a expressão `1 != "1"` retorna *false*, pois os valores são iguais.
- Não Identidade (!==): verifica se dois valores não possuem a mesma identidade. Sua proposta é fazer exatamente o contrário do que o operador de identidade, ou seja, verificar se dois valores são diferentes, levando em consideração o tipo de dado. Por exemplo, a expressão `1 !== "1"` retorna *true*, pois o tipo de dado dos dois operandos é diferente, enquanto os valores são iguais.
- Menor que (<): verifica se o primeiro operando é menor que o segundo.
- Maior que (>): verifica se o primeiro operando é maior que o segundo.
- Menor ou igual a (<=): testa se o primeiro operando é menor ou igual ao segundo.
- Maior ou igual a (>=): testa se o primeiro operando é maior ou igual ao segundo.
- E (&&): junta duas expressões booleanas, retornando verdadeiro somente se ambas forem verdadeiras.
- Ou (||): junta duas expressões booleanas, retornando verdadeiro somente se uma das duas ou ambas forem verdadeiras.

7 OPERADORES ARITMÉTICOS

O JavaScript possui os mesmos operadores aritméticos que as demais linguagens de programação que possuem a estrutura léxica baseada em C. Listamos os operadores aritméticos de JavaScript a seguir:

- Adição (+): soma os dois operandos.
- Subtração (-): subtrai o operando da direita do da esquerda.
- Multiplicação (*): multiplica os dois operandos.
- Divisão (/): divide o primeiro operando pelo segundo.
- Módulo (%): retorna o resto da divisão do primeiro operando pelo segundo.

- Incremento (++): incrementa em 1 o valor de seu operando. Caso o operador estiver à esquerda do operando, ele primeiro incrementa e depois retorna o novo valor. Caso o operador estiver à direita do operando, ele primeiro retorna o valor antigo e depois incrementa (teremos exemplo em seguida).
- Decremento (--) subtrai em 1 o valor de seu operando. Caso o operador estiver à esquerda do operando, ele primeiro decrementa e depois retorna o novo valor. Caso o operador estiver à direita do operando, ele primeiro retorna o valor antigo e depois decrementa (teremos exemplo em seguida).



C?

C é uma linguagem de programação que foi muito utilizada no passado. Atualmente C ainda é muito encontrado em código-fonte de baixo nível (referente às camadas baixas de sistemas, por exemplo, comunicação com um *hardware* específico), muitas vezes rodando em microcontroladores (por exemplo Arduino). A estrutura léxica do C foi utilizada como base para a implementação das linguagens C#, Java e PHP. A linguagem sucessora do C é o C++. Aprender a escrever algoritmos em C é muito interessante, pois fornece um embasamento muito bom para qualquer outra linguagem que você for aprender. Atualmente, a plataforma de prototipação Arduino utiliza o C como linguagem de programação e tem o objetivo de introduzir pessoas à eletrônica e aos microcontroladores, é muito divertido e interessante.

A lógica de precedência dos operadores matemáticos é também a mesma da maioria das linguagens, primeiro são executadas as operações de divisão e multiplicação e depois as de adição e subtração. A ordem de precedência das operações pode ser alterada através do uso de parênteses na instrução. Quando ocorre um erro na execução de um cálculo, JavaScript atribui *infinite* ou *Nan* (*Not a Number*) à variável que receberia o resultado da operação. Agora vamos ver em termos de código-fonte a utilização dos operadores aritméticos em JavaScript, pois é programando que se aprende a programar.

1. Em seu computador, crie um arquivo chamado operadoresaritmeticos.html, edite-o, insira o código-fonte da listagem a seguir e salve-o:

LISTAGEM 41 - OPERADORES ARITMÉTICOS EM JAVASCRIPT

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8">
    <title>Operadores matemáticos</title>
    <script type="text/javascript">
```

```
document.write("<p>Fazendo contas !<br />");

var x = 1;
var y = 2;

var soma = x + y;
document.write(soma);
document.write("<br />");
// Resultado: 3

var diferenca = x - y;
document.write(diferenca);
document.write("<br />");
// Resultado: -1

var multiplicacao = y * 3;
document.write(multiplicacao);
document.write("<br />");
// Resultado: 6

var divisao = multiplicacao / 2;
document.write(divisao);
document.write("<br />");
// Resultado: 3

var modulo = divisao % 2;
document.write(divisao);
document.write("<br />");
// Resultado: 3

// Incremento anterior:
// Primeiro incrementa depois retorna
// o valor da operação
var incrementoAnterior = 2;
document.write(++incrementoAnterior);
document.write(" -> ");
document.write(incrementoAnterior);
document.write("<br />");
// Resultado: 3 -> 3

// Incremento posterior:
// Primeiro retorna o valor original
// depois incrementa
```

```
var incrementoPosterior = 2;
document.write(incrementoPosterior++);
document.write(" -> ");
document.write(incrementoPosterior);
document.write("<br />");
// Resultado: 2 -> 3

// Decremento anterior:
// Primeiro decremente depois retorna
// o valor da operação
var decrementoAnterior = 2;
document.write(--decrementoAnterior);
document.write(" -> ");
document.write(decrementoAnterior);
document.write("<br />");
// Resultado: 1 -> 1

// Decremento posterior:
// Primeiro retorna o valor original
// depois decrementa
var decrementoPosterior = 2;
document.write(decrementoPosterior--);
document.write(" -> ");
document.write(decrementoPosterior);
document.write("<br />");
// Resultado: 2 -> 1

// Equação sem uso de parênteses:
var semParenteses = 2 + 2 * 3;
document.write(semParenteses);
document.write("<br />");
// Resultado: 8

// Equação com uso de parênteses:
var comParenteses = (2 + 2) * 3;
document.write(comParenteses);
document.write("<br />");
// Resultado: 12

// Erro na execução de um cálculo
// divisão por zero
var valor = 0;
var resultado = 4 / valor;
```

```

        document.write(resultado);
        document.write("<br />");
        // Resultado: Infinity

        document.write("</p>");

    </script>
</head>
<body>
</body>
</html>

```

2. Abra seu arquivo operadoresaritmeticos.html em seu *browser*.

8 CONTROLANDO O FLUXO DE EXECUÇÃO DOS PROGRAMAS COM *IF*, *FOR* E *WHILE*

As estruturas para controle do fluxo de execução em JavaScript são muito semelhantes às encontradas em PHP. Na verdade, você já percebeu que há várias semelhanças. JavaScript possui sua estrutura léxica baseada em Java (mas as semelhanças entre JavaScript e Java param por aí), que por sua vez, possui sua estrutura léxica baseada em C. O PHP também usou a estrutura léxica do C como base, a isto devemos tantas semelhanças nestas linguagens de programação. Mas vamos focar agora no JavaScript, vamos explorar as estruturas de controle do fluxo de execução.

Se meu saldo bancário for maior ou igual a R\$ 35.340,00, vou comprar um carro 0km, senão, comprarei um carro usado. Você já conhece esta proposição de nossos estudos em PHP, e também já deve imaginar o que devemos utilizar para implementar uma condição em um programa JavaScript que conte a regra desta proposição. A instrução *if* é a ideal para implementações como esta em JavaScript. Ela recebe como parâmetro um booleano (entre parênteses) que define para onde o fluxo de execução deve seguir, se a expressão booleana retornar verdadeiro, segue para o bloco de código do *if*, se retornar falso, segue para o bloco de código do *else*. A proposição que define se iremos comprar um carro zero ou usado, pode ser implementada em JavaScript conforme a listagem a seguir.

LISTAGEM 42 - INSTRUÇÃO IF EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">

```

```

<title>if em JavaScript</title>
<script type="text/javascript">

    document.write("<p>");

    // Define a variável para armazenar o saldo bancário
    var saldoBancario = 40000;

    if (saldoBancario >= 35340) {
        document.write("Irei comprar um carro 0 km :-)");
    } else {
        document.write("Irei comprar um carro usado   :-|");
    }
    // Resultado: Irei comprar um carro 0 km :-(

    document.write("</p>");
</script>
</head>
<body>
</body>
</html>

```

A instrução *if* também é muito utilizada para verificar se uma operação matemática foi executada com sucesso ou se houve erro durante tal operação. Para isto, fazemos uso da função *isFinite()*, que recebe como parâmetro uma variável numérica e retorna *true* se o resultado for um valor numérico válido ou *false* se for um valor numérico inválido (*NaN*, *Infinity* ou *-Infinity*), caracterizando erro na operação matemática. Veja o exemplo da listagem a seguir.

LISTAGEM 43 - VERIFICANDO SUCESSO DE OPERAÇÕES MATEMÁTICAS EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Verificando erro ao fazer contas</title>
        <script type="text/javascript">

            document.write("<p>");

            var contaLouca = 30 / 0;

            if (isFinite(contaLouca)){
                document.write("A conta louca deu certo :-)");
            } else {
                document.write("Deu erro na conta louca :-(");
            }
            // Resultado: Deu erro na conta louca :-(

</script>

```

```

        document.write("</p>");
    </script>
</head>
<body>
</body>
</html>

```

Agora vamos ver os *loopings* em JavaScript. Os *loopings* você já conhece, desde quando estudava algoritmos e/ou linguagem de programação iniciante, ou principalmente de nossos estudos de PHP. São estruturas de controle que permitem executar um bloco de comandos em determinado número de vezes ou enquanto determinada condição for verdadeira. Em JavaScript, há os *loopins for, while* e *do while* disponíveis para utilizar em nossas páginas. Para exemplificar a utilização de cada um deles, iremos implementar um exemplo que imprime os algarismos de um a dez utilizando cada tipo de *looping*. Confira nas listagens a seguir.

O *looping for* tem seu funcionamento baseado na quantidade de vezes em que seu bloco de código deve ser executado. Recebe três parâmetros, separados por ponto e vírgula. Em nosso exemplo, o primeiro parâmetro, é a inicialização da variável i utilizada como contador, o segundo parâmetro é a condição booleana que estabelece até quando o *looping* ficará executando, e o terceiro parâmetro é o incremento do contador. Veja a listagem a seguir:

LISTAGEM 44 - IMPRIMINDO OS ALGARISMOS DE UM A DEZ COM O LOOPING FOR EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Looping for</title>
        <script type="text/javascript">

            document.write("<p>");

            // Imprimindo os algarismos de um a dez
            // com o looping for
            for (var i = 1; i <=10; ++i){
                document.write(i + "<br />");
            }
            // Resultado 1 2 3 4 5 6 7 8 9 10
            // um número por linha

            document.write("</p>");

        </script>

```

```
</head>
<body>
</body>
</html>
```

O *looping while* baseia-se somente em uma condição booleana para identificar se deve ou não continuar executando seu bloco de código. Seu único parâmetro é uma expressão booleana que quando verdadeira, o *looping* continha executando seu bloco de código, se for falsa, o *looping* termina e passa para o comando seguinte. O *looping while* sempre verifica se a condição é verdadeira antes de iniciar a execução de seu bloco de código. Confira a listagem a seguir.

LISTAGEM 45 - IMPRIMINDO OS ALGARISMOS DE UM A DEZ COM O LOOPING WHILE EM JAVASCRIPT

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Looping com while</title>
        <script type="text/javascript">

            document.write("<p>");

            // Imprimindo os algarismos de um a dez
            // com o looping while
            var i = 1;

            while (i <= 10){
                document.write(i + "<br />");
                ++i;
            }
            // Resultado 1 2 3 4 5 6 7 8 9 10
            // um número por linha

            document.write("</p>");

        </script>
    </head>
    <body>
    </body>
</html>
```

O *looping do while* possui todas as mesmas características do *looping while* exceto pela característica de executar seu bloco de código antes de verificar se a condição para saída ainda é verdadeira. Desta forma, o bloco de código do *looping* sempre é executado pelo menos uma vez. Veja no exemplo a seguir.

LISTAGEM 46 - IMPRIMINDO OS ALGARISMOS DE UM A DEZ COM O *LOOPING DO WHILE* EM JAVASCRIPT

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Looping com do while</title>
        <script type="text/javascript">

            document.write("<p>");

            // Imprimindo os algarismos de um a dez
            // com o looping do while
            // Note que o bloco de código entre
            // do e while é executado pelo menos
            // uma vez
            var i = 1;

            do {
                document.write(i + "<br />");
                ++i;
            } while (i <= 10)
            // Resultado 1 2 3 4 5 6 7 8 9 10
            // um número por linha

            document.write("</p>");

        </script>
    </head>
    <body>
        </body>
</html>
```



Loopings também são muito importantes para efetuarmos operações sobre elementos de *arrays*. JavaScript disponibiliza uma sintaxe específica para iterar sobre *arrays*, através do operador *in*. Trata-se do mesmo conceito de *enhanced for* encontrado em outras linguagens de programação, como o Java, o C# e o PHP. Quando estudarmos *arrays*, você conhecerá esta outra sintaxe para *loopings*.

9 STRINGS

Strings em JavaScript podem ser representadas de duas formas, com aspas simples ou aspas duplas. Em JavaScript, as variáveis que armazenam *strings* podem ser vistas como objetos, ou seja, ao contrário do PHP, onde as funções de manipulação de *string* são chamadas a partir do contexto global, em JavaScript os métodos de manipulação de *string* são chamados diretamente da variável que contém as *strings* através do operador ponto (.). Se você não está familiarizado com a orientação a objetos, analise bem as próximas listagens e perceba a diferença que há entre a manipulação de *strings* em JavaScript e PHP.

Há várias operações que podem ser efetuadas sobre uma *string*. Você pode querer substituir determinado conteúdo da *string* por outro, ou pode querer descobrir qual caractere está em determinada posição da *string*. Iremos abordar as principais funções de manipulação de *strings* em JavaScript, todas com exemplos neste tópico. Para declarar uma *string* basta declarar uma variável e atribuir a ela um valor entre aspas simples ou duplas. Você também, certamente, já sabe, de exemplos anteriores, que *strings* são concatenadas através do operador mais (+). Em JavaScript, o valor da variável na qual o método de manipulação de *strings* foi chamado não sofre alteração, em vez disto, é retornado um novo valor, por este motivo, as *strings* são consideradas imutáveis. Veja na listagem a seguir.

LISTAGEM 47 - DECLARANDO STRINGS EM JAVASCRIPT

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8">
    <title>Declarando Strings</title>
    <script type="text/javascript">

      // Note que "<p>", "<br />" e "</p>" são strings
      // Você já deve estar acostumado a ver
      // literais strings em seus códigos
      // fonte em JavaScript
      // Você certamente também já sabe que strings
      // São concatenadas através do operador +
      document.write("<p>");

      // Declara uma string usando aspas duplas
      var primeiraString = "Aspas duplas";

      // Declara uma string usando aspas simples
      var segundaString = 'Aspas simples';
```

```

        document.write(primeiraString + "<br />");
        document.write(segundaString);
        document.write("</p>");

    </script>
</head>
<body>
</body>
</html>

```

Agora vamos explorar como manipular o conteúdo de *strings* através dos métodos do objeto *string*. Veja a seguir os principais métodos, para cada um deles, um tópico explicativo e uma listagem exemplificando sua utilização.

- concat(): concatena *strings*, pode ser escrito de forma mais prática utilizando o operador mais (+)

LISTAGEM 48 - CONCATENANDO STRINGS EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Concatenando strings</title>
        <script type="text/javascript">

            document.write("<p>");

            var fraseBonita = "Estudar JavaScript é demais!";

            // Efetua uma concatenação com o método concat
            var testeConcat = fraseBonita.concat(" :-)");
            document.write(testeConcat + "<br />");
            // Resultado: Estudar JavaScript é demais! :-)

            // efetua uma concatenação com o operador mais (+)
            var testeMais = fraseBonita + " :-D";
            document.write(testeMais + "<br />");
            // Resultado: Estudar JavaScript é demais! :-D

            // Exibe a frase original.
            // Note que o conteúdo da variável fraseBonita
            // não sofre alterações.

```

```

    // por este motivo, strings são consideradas
    // imutáveis em JavaScript
    document.write(fraseBonita);
    // Resultado: Estudar JavaScript é demais!

    document.write("</p>");
</script>
</head>
<body>
</body>
</html>

```

- `replace()`: substitui um conteúdo da *string* por outro.

LISTAGEM 49 - SUBSTITUINDO STRINGS EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>Substituindo strings</title>
    <script type="text/javascript">

        document.write("<p>");

        var fraseBonita = "Estudar JavaScript é demais!";

        // explorando o método replace()
        // ele substitui a string do primeiro parâmetro
        // pela string do segundo parâmetro.
        var testeReplace =
fraseBonita.replace("JavaScript", "na"
+ "Uniasselvi");
        document.write(testeReplace);
        // Resultado: Estudar na Uniasselvi é demais!

        // Exibe a frase inicial
        document.write("<br />");
        document.write(fraseBonita);

        document.write("</p>");
</script>
</head>
<body>

```

```
</body>
</html>
```

- `toLowerCase()`: transforma todas as letras da *string* em minúsculas.

LISTAGEM 50 - TRANSFORMANDO LETRAS PARA MINÚSCULAS EM JAVASCRIPT

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Transformando letras para minúsculas</title>
        <script type="text/javascript">

            document.write("<p>");

            var fraseBonita = "Estudar JavaScript é demais!";

            // explorando o método toLowerCase()
            // ele transforma todas as letras da string
            // em minúsculo
            var fraseBonitaMin = fraseBonita.toLowerCase();
            document.write(fraseBonitaMin);
            // Resultado: estudar javascript é demais!

            // Exibe a frase inicial
            document.write("<br />");
            document.write(fraseBonita);

            document.write("</p>");
        </script>
    </head>
    <body>
        </body>
    </html>
```

- `toUpperCase()`: transforma todas as letras da *string* em maiúsculas.

LISTAGEM 51 - TRANSFORMANDO LETRAS PARA MAIÚSCULAS EM JAVASCRIPT

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Transformando letras para maiúsculas</title>
```

```

<script type="text/javascript">

    document.write("<p>");

    var fraseBonita = "Estudar JavaScript é demais!";

        // explorando o método toUpperCase()
        // ele transforma todas as letras da string
        // em maiusculo
    var fraseBonitaMai = fraseBonita.toUpperCase();
    document.write(fraseBonitaMai);
    // Resultado: ESTUDAR JAVASCRIPT É DEMAIS!

        // Exibe a frase inicial
    document.write("<br />");
    document.write(fraseBonita);

    document.write("</p>");
</script>
</head>
<body>
</body>
</html>

```

- `trim()`: remove espaços em branco do início e do final da *string*.

LISTAGEM 52 - REMOVENDO ESPAÇOS EM BRANCO EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Removendo espaços em branco</title>
        <script type="text/javascript">

            document.write("<p>");

                // explorando o método trim();
                // ele tira os espaços em branco ao redor
                // da string
            var stringComEspacos = "      tem espaços      ";

```

```

        document.write("/*" + stringComEspacos +
"*/<br />");

// Resultado: /* tem espaços */

var stringSemEspacos = stringComEspacos.trim();
document.write("/*" + stringSemEspacos +
"*/<br />");

// Resultado: /*tem espaços*/

        document.write("</p>");
    </script>
</head>
<body>
</body>
</html>

```

- *substr()* e *substring()*: retorna uma *string* com apenas parte da *string* original. Para entender o funcionamento destes métodos, entenda que em JavaScript todos os caracteres de uma *string* possuem um índice numérico iniciado em zero. Por exemplo, em UNIASSELVI os caracteres são indexados da seguinte forma: 0=U, 1=N, 2=I, 3=A, 4=S, 5=S, 6=E, 7=L, 8=V, 9=I

LISTAGEM 53 - OBTENDO PEDAÇOS DA STRING EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>Obtendo pedaços da string</title>
    <script type="text/javascript">

        document.write("<p>");
        var fraseBonita = "Estudar JavaScript é demais!";

        // explorando o método substr()
        // lembra-se de que as strings são indexadas
        // a partir de zero
        // o primeiro parâmetro é o índice do qual a
        // pesquisa deve iniciar
        // o segundo parâmetro é a quantidade de
        // caracteres a serem extraídos
        // no exemplo abaixo, busca-se uma string
        // que inicia no oitavo índice e possui
        // dez caracteres de comprimento

```

```

var testesSubstr = fraseBonita.substr(8, 10)
document.write(testesSubstr + "<br />");
// Resultado: JavaScript

// explorando o método substring()
// ele possui a mesma finalidade do substr()
// mas faz isto de um jeito diferente
// ele recebe o índice inicial e o índice final
// como parâmetros
// No exemplo, buscamos uma palavra que inicia
// no índice oito e termina no índice dezoito
var testeSubstring = fraseBonita.
substring(8, 18);
document.write(testeSubstring + "<br />");
// Resultado: JavaScript

// Exibe a frase original
document.write(fraseBonita);

document.write("</p>");
</script>
</head>
<body>
</body>
</html>

```

- **search():** busca determinado valor em uma *string* e retorna a posição na qual este valor foi encontrado.

LISTAGEM 54 - PESQUISANDO EM UMA STRING EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>Pesquisando em uma string</title>
    <script type="text/javascript">

        document.write("<p>");
        var fraseBonita = "Estudar JavaScript é demais!";

        // explorando o método search()
        // ele pesquisa uma string dentro de outra
        // e retorna o índice no qual a string foi

```

```

        // encontrada
        var testeSearch = fraseBonita.
search("JavaScript");
        document.write(testeSearch);
        // Resultado: 8

        // Exibe a frase original
        document.write("<br />");
        document.write(fraseBonita);

        document.write("</p>");
    </script>
</head>
<body>
</body>
</html>

```

- `length`: propriedade que retorna a quantidade de caracteres em uma *string*.

LISTAGEM 55 - TAMANHO DE UMA STRING EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Tamanho de uma string</title>
        <script type="text/javascript">

            document.write("<p>");
            var fraseBonita = "Estudar JavaScript é demais!";

            // explorando a propriedade length
            // que retorna a quantidade de caracteres
            // de uma string
            var testeLength = fraseBonita.length;
            document.write(testeLength);
            // Resultado: 28

            // Exibe a frase original
            document.write("<br />");
            document.write(fraseBonita);

            document.write("</p>");

```

```

    </script>
</head>
<body>
</body>
</html>

```

Percebemos que o JavaScript disponibiliza muitos métodos para manipular *strings*. É possível fazer o que quisermos, mas é necessário ter cautela. Como já mencionamos, as *strings* em JavaScript são imutáveis, ou seja, a cada operação uma *string* é gerada. O uso abusivo de operações com *string*, por exemplo, uma concatenação dentro de um *looping*, resulta em uso abusivo de memória, podendo travar nossa aplicação Web. De qualquer forma, se avaliarmos bem as circunstâncias, temos em mãos um arsenal poderosíssimo para efetuarmos manipulação de *strings*.

10 ARRAYS

Arrays são listas de elementos, onde cada elemento possui um índice. Em JavaScript, os *arrays* são indexados por números inteiros positivos iniciando em 0. Como JavaScript é uma linguagem com tipagem dinâmica, os *arrays* podem conter elementos de qualquer tipo de dado. Há duas maneiras de declarar um *array*, através do construtor `Array()` ou através dos literais de *arrays* (colchetes `[]`).

Há três maneiras de declarar um *array* através do construtor. A primeira é utilizando o construtor vazio, o que resulta em um *array* vazio. A segunda utilizando o construtor informando como parâmetro a lista de elementos separados por vírgula (`,`), o que resulta em um *array* com tais elementos. A terceira é utilizando o construtor passando um valor numérico inteiro positivo, o que resulta em um *array* com a quantidade de elementos informado no parâmetro, porém todos os elementos são *undefined*, pois não foram inicializados. Confira na listagem a seguir.

LISTAGEM 56 - UTILIZANDO O CONSTRUTOR DE ARRAY EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>Declarando arrays com o construtor</title>
    <script type="text/javascript">

        //declara um array vazio
        var array1 = new Array();

```

```

    // declara um array com os elementos
    // "um" e 2
    var array2 = new Array("um", 2);

    // declara um array com oito
    // elementos, todos eles não
    // inicializados, ou seja
    // undefined
    var array3 = new Array(8);

</script>
</head>
<body>
</body>
</html>

```

Uma abordagem mais elegante e atualizada é utilizar os literais de *array* para declarar *arrays*. Os literais de *array* são os colchetes ([]), e podem ser usados de duas formas na declaração de *arrays*. A primeira forma é utilizar os colchetes vazios, sendo que neste caso a operação resulta na criação de um *array* vazio. A segunda maneira é passar como parâmetro (dentro dos colchetes) a lista de elementos do *array*, separando-os com vírgula (,), sendo que neste caso é criado um *array* com os elementos informados. Veja na listagem a seguir.

LISTAGEM 57 - UTILIZANDO LITERAIS DE ARRAY PARA DECLARAR ARRAYS EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Declarando arrays com literais de
array</title>
        <script type="text/javascript">

            //declara um array vazio
            var array1 = [];

            // declara um array com os elementos
            // "um" e 2
            var array2 = ["um", 2];

</script>
</head>
<body>

```

```
</body>
</html>
```

Para acessar os elementos de *arrays*, utilizamos novamente os colchetes. É necessário informar a variável do *array*, seguido de colchetes, e o índice do elemento entre os colchetes. Por exemplo, considerando a listagem 57 (anterior), para acessar o primeiro elemento do array2 usa-se a expressão array2[0] e para acessar o segundo elemento utiliza-se a expressão array2[1]. Da mesma forma, podemos inserir ou atualizar os elementos do *array* utilizando os colchetes. Por exemplo, ainda considerando o código-fonte da listagem 57, poderíamos adicionar um terceiro elemento ao array2 apenas atribuindo um valor ao índice 2 do *array*, conforme a expressão array2[2] = "Terceiro elemento";.

Os *arrays* em JavaScript possuem, assim como as *strings*, uma propriedade para retornar o seu comprimento. Porém no caso das *strings* a propriedade *length* retorna a quantidade de caracteres da *string*, enquanto que nos *arrays*, a propriedade *length* retorna o número do maior índice + 1. Isto pode confundir o desenvolvedor, pois em JavaScript os *arrays* podem ser esparsos, ou seja, um *array* com três elementos, não necessariamente possui os índices 0, 1 e 2, ele pode possuir os índices 2, 5 e 7, sendo que no caso, a propriedade *length* retorna 8. Não é uma característica simples de ser entendida, portanto, analise com muita atenção cada linha de código e cada comentário das duas próximas listagens.

LISTAGEM 58 - ACESSANDO OS ELEMENTOS DE ARRAYS E ANALISANDO O FUNCIONAMENTO DA PROPRIEDADE LENGTH

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Acessando elementos de arrays e
testando length</title>
        <script type="text/javascript">

            document.write("<p>");

            // delcara um array
            // Quando se utiliza este construtor, os elementos
            // são indexados automaticamente a partir de zero
            var arrayBom = ["primeiro", "segundo", "terceiro"];

            // acessa os elementos do array
            document.write(arrayBom[0]); // primeiro
            document.write("<br />");
            document.write(arrayBom[1]); // segundo
```

```

        document.write("<br />");  

        document.write(arrayBom[2]); // terceiro  

        document.write("<br />");  
  

        // Ao tentar acessar o elemento de um  

        // índice não declarado, recebemos undefined  

        document.write(arrayBom[5]);  

        document.write("<br />");  

        // Resultado: undefined  
  

        // A propriedade length retorna o  

        // comprimento do array. Para obter o comprimento  

        // ela identifica o maior índice existente no  

        // array e soma um. No caso do array arrayBom,  

        // o maior índice é 2. Portanto ao acessar  

        // arrayBom.length, irá retornar 3, pois 2+1=3  

        document.write(arrayBom.length);  

        document.write("<br />");  

        // Resultado: 3  
  

        document.write("</p>");  

    </script>  

</head>  

<body>  

</body>  

</html>

```

LISTAGEM 59 - INSERINDO ELEMENTOS EM ARRAYS E ANALISANDO O FUNCIONAMENTO DA PROPRIEDADE LENGTH EM ARRAYS ESPARSSOS

```

<!DOCTYPE html>  

<html lang="pt-br">  

    <head>  

        <meta charset="utf-8">  

        <title>Array esparsos</title>  

        <script type="text/javascript">  
  

            document.write("<p>");  
  

            // Declaramos um array vazio  

            var arrayEsparsos = [];  
  

            // Inserimos o primeiro elemento no  

            índice 2

```

```

arrayEsparsso[2] = "primeiro";

// inserimos o segundo elemento no índice 5
arrayEsparsso[5] = "segundo";

// inserimos o terceiro elemento no índice 7
arrayEsparsso[7] = "terceiro";

// Podemos dizer que o array da variável
// arrayEsparsso é esparsso, pois a sequência de
// índices dele não corresponde ao padrão do
// JavaScript, que inicia de zero e incrementa
// de um em um

// Para obter o comprimento de um array a
propriedade
// length identifica o maior índice existente no
// array e soma um. No caso do array
arrayEsparsso,
// o maior índice é 7. Portanto ao acessar
// arrayEsparsso.length, irá retornar 8, pois
7+1=8
document.write(arrayEsparsso.length);
document.write("<br />");
// Resultado: 8

document.write("</p>");
</script>
</head>
<body>
</body>
</html>

```

Agora que sabemos declarar um *array*, e inserir elementos nele, iremos ver como imprimi-lo e como iterar sobre seus elementos. Você deve se lembrar, de nossos estudos de PHP, da função `print_r`, que imprime todos os elementos do *array*. No JavaScript, contamos com dois métodos para imprimir o conteúdo do *array*, o primeiro é o `toString()`, que imprime os elementos separados por vírgula, e o segundo é o `join()`, que nos permite escolher se queremos usar a vírgula ou algum outro separador. Veja na listagem seguir.

LISTAGEM 60 - IMPRIMINDO OS ELEMENTOS DE UM ARRAY

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Imprimindo elementos de um array</
title>
        <script type="text/javascript">
            document.write("<p>");

            var meuArray = ["um", "dois", "tres",
"quatro", "etc"];

            // Imprime os elementos do array através
            // do método toString()
            document.write(meuArray.toString());
            document.write("<br />");
            // Resultado:    um,dois,tres,quatro,etc

            // Imprime os elementos do array
            // através do método join()
            document.write(meuArray.join("; "));
            // Resultado: um; dois; tres; quatro; etc

            document.write("</p>");

        </script>
    </head>
    <body>
        </body>
    </html>

```

Enquanto estudávamos as estruturas para controlar o fluxo de execução do código, vimos o *looping for*. Ele é importantíssimo quando desejamos efetuar operações sobre os elementos de *arrays*. Há duas maneiras de trabalhar com o *looping for* para iterar sobre *arrays* em JavaScript. A primeira é iterando com o auxílio de uma variável contadora, e acessar os elementos um a um através de seu índice. Infelizmente esta não é uma prática recomendada, pois pode causar 'dor de cabeça' quando trabalhamos com *arrays* esparsos. A segunda é através do *enhanced for*, que por sua vez, abstrai a utilização do índice. A sintaxe do *enhanced for* declara uma variável que representa o elemento do *array* sobre o qual a iteração está ocorrendo. Mais uma vez, é algo difícil de ser entendido por meio de palavras, então vamos ver em termo de código-fonte. Preste muita atenção nos comentários e nas linhas de código da listagem a seguir.

LISTAGEM 61 - ITERANDO SOBRE OS ELEMENTOS DE ARRAYS EM JAVASCRIPT

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Imprimindo elementos de um array</
title>
        <script type="text/javascript">
            document.write("<p>");

            var meuArray = ["um", "dois", "tres",
"quatro", "etc"];

                // Iterando sobre os elementos do array
através de um
                // looping for utilizando a escrita
tradicional, com
                // uma variável contadora incrementada a
cada iteração
                document.write("Iterando com for: <br
/>");

                for (var i = 0; i < meuArray.length; i++)
{

                // A variável i representa o índice
                document.write(i + " = ");

                // Acessa o elemento de meu array no
índice i
                document.write(meuArray[i]);
                document.write("<br />");
                // Resultado: o elemento do array no
índice i
}

                // Iterando sobre os elementos do array
através de um
                // looping enhanced for. A variável
índice (a esquerda
                // do in) recebe o índice que está
sofrendo iteração
```

```

    // no momento, sem necessidade de
incrementar
    // contador algum
document.write("Iterando com enhanced
for: <br />");

for (indice in meuArray) {

    // A variável indice representa o índice
    document.write(indice + " = ");

    document.write(meuArray[indice]);
    document.write("<br />");
    // Resultado: o elemento do array no índice

}

document.write("</p>");
</script>
</head>
<body>
</body>
</html>

```

Já sabemos declarar um *array*, inserir elementos nele e iterar sobre seus elementos, mas e se for necessário excluir elementos de um *array*? Então devemos usar o método *splice()*. O método *splice* exclui elementos de um *array* e retorna um novo *array* contendo os elementos excluídos. Recebe dois parâmetros, o primeiro é o índice do qual a exclusão deve começar e o segundo parâmetro é a quantidade de elementos a serem excluídos. Se o segundo parâmetro for omitido, todos os elementos restantes do *array*, ou seja, do índice indicado no primeiro parâmetro até o final do *array*, serão excluídos. Veja na listagem a seguir o funcionamento do método *splice*.

LISTAGEM 62 - EXCLUINDO ELEMENTOS DE UM ARRAY EM JAVASCRIPT

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>Excluindo elementos de um array</
title>

```

```
<script type="text/javascript">
    document.write("<p>");

    // Declara um array
    var meuArray = ["um", "dois", "tres",
"quatro", "etc"];

    // Imprime os elementos do array e seu
índice
    document.write("Imprimindo o array
original: <br />");
    for (indice in meuArray) {
        document.write(indice + " = ");

        document.write(meuArray[indice]);
        document.write("; ");

    }
    document.write("<br />");
    // Resultado: 0 = um; 1 = dois; 2 = tres;
    // 3 = quatro; 4 = etc;

    // exclui a o terceiro e o quarto
elemento
    // do array e armazena os ítems excluidos
    // em um novo array dentro da variável
    // excluidos
    var excluidos = meuArray.splice(2, 2);

    // Imprime os elementos do array e seu
índice
    // é importantíssimo notar que o método
splice
    // reorganiza os índices do array, ou
seja, não
```

```
// torna o array esparso
document.write("Array após a exclusão:
<br />");

for (indice in meuArray){
    document.write(indice + " = ");

    document.write(meuArray[indice]);
    document.write("; ");

}

document.write("<br />");
// Resultado: 0 = um; 1 = dois; 2 = etc;

// Iterando sobre os itens excluídos
document.write("Ítems excluídos: <br
/>");

for (indice in excluidos){
    document.write(indice + " = ";

    document.write(excluidos[indice]);
    document.write("; ");

}

document.write("<br />");
// Resultado: 0 = tres; 1 = quatro;

document.write("</p>");
</script>
</head>
<body>
</body>
</html>
```

11 DECLARANDO FUNÇÕES EM JAVASCRIPT

Declarar funções agrupa muita reusabilidade a seu código. Uma função é uma estrutura que possui um nome, opcionalmente uma lista de parâmetros, um bloco de instruções, e opcionalmente um retorno. Você deve se lembrar da ocasião em que você aprendeu sobre funções a primeira vez, talvez na disciplina de algoritmos ou linguagem de programação iniciante, e viu as infinitas possibilidades que a declaração de funções traz ao desenvolvimento de *software*.

Em JavaScript podemos declarar funções de diversas maneiras. Iremos focar na abordagem mais atual, que são os literais de função. Os literais de função nos permitem atribuir as funções a uma variável (isto mesmo, a uma variável!). Para quem nunca viu isto antes, é bem incomum, mas não na programação JavaScript. Na verdade, é até simples, você declara uma variável e atribui a ela uma função. A função inicia com a instrução *function*, entre parênteses uma lista opcional de parâmetros separados por vírgula e um bloco de instruções que pode ou não conter a instrução *return*. Vamos ver em termos de código-fonte, uma função que soma dois números, na listagem a seguir.

LISTAGEM 63 - DECLARANDO FUNÇÕES EM JAVASCRIPT

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8">
    <title>Declarando funções</title>
    <script type="text/javascript">

      var fSoma = function(x, y) {
        var resultado = x + y;
        return resultado;
      }

      document.write("<p>");

      var soma = fSoma(3, 5);
      document.write(soma);
      // Resultado: 8
    </script>
  </head>
  <body>
    <h1>Declarando funções</h1>
    <p>A soma de 3 e 5 é:</p>
    <p>8</p>
  </body>
</html>
```

```
        document.write("</p>");  
    </script>  
  </head>  
  <body>  
  </body>  
</html>
```

Um aspecto muito interessante no JavaScript é que as funções são a única forma de reduzir o escopo das variáveis. Nas demais linguagens, em geral, qualquer bloco anônimo já interfere no escopo das variáveis. Se você estudou C# ou Java deve se lembrar que se você declarar uma variável dentro do bloco de instruções de um *if*, ela sobrevive somente dentro deste bloco. No JavaScript, somente as variáveis de dentro de funções tem seu escopo limitado, as variáveis declaradas dentro de outros blocos de código tem o mesmo escopo do que as demais. Observe a listagem a seguir e leia atentamente os comentários.

LISTAGEM 64 - ANALISANDO O ESCOPO DAS VARIÁVEIS EM JAVASCRIPT

```
<!DOCTYPE html>  
<html lang="pt-br">  
  <head>  
    <meta charset="utf-8">  
  <title>Escopo de variáveis</title>  
  <script type="text/javascript">  
  
    // Declara uma função  
    var funcao = function() {  
  
      // Declara uma variável dentro da função  
      var internaFuncao = "dentro da função";  
  
      // A variável está viva somente até aqui  
      document.write(internaFuncao);  
      document.write("<br />");  
      // Resultado: dentro da função
```

```
}

// Se acessar a variável internafuncao aqui
// ela não existe mais. Vai ocorrer um erro
// de execução e vai abortar a execução
// do script
// Comente a linha seguinte se quiser
// ver o script funcionar
document.write(internaFuncao);
document.write("<br />");
// Resultado: erro, script abortado !!

// Iniciamos um looping para termos um
// bloco de instruções para ilustrar a
// situação na qual as variáveis de blocos
// de instruções permanecem declaradas
// mesmo após o bloco de instruções terminar
for (var i = 0; i < 1; i++) {
    var internaFor = "dentro do for";

    document.write(internaFor);
    document.write("<br />");
    // Resultado: dentro do for
}

document.write("Saiu do for<br />");
document.write(internaFor);
// Resultado: dentro do for

document.write("</p>");
</script>
</head>
<body>
</body>
</html>
```

12 DOM, A PONTE ENTRE HTML E JAVASCRIPT

Até o momento, as páginas *Web* que utilizamos para aprender JavaScript não faziam nada além de demonstrar o funcionamento do JavaScript. Páginas *Web* de verdade tem que proporcionar interação com o usuário. Nós já aprendemos, na Unidade I, como desenvolver páginas *Web* com HTML e CSS. Onde fica o JavaScript neste contexto? Para responder a esta questão, é necessário entender primeiro o conceito de DOM (*Document Object Model*).

O Modelo de Objeto Documento, como é conhecido o DOM, é na verdade uma transcrição dos elementos da página *Web* para objetos JavaScript. Se você adicionar uma imagem a uma página *Web*, tenha certeza de que ela é manipulável via JavaScript através do DOM.

O DOM possui um contexto global chamado *window*, que abriga as variáveis globais de nossos códigos-fonte. Mas o objeto mais interessante do DOM é mesmo o *document*, o qual você já vem usando a bastante tempo para escrever na página *Web* através do método *document.write*. A manipulação do DOM fica muito simples se você adicionar a seus elementos HTML o atributo *id* com um identificador para cada elemento. Então, o elemento HTML nomeado com *id* pode ser acessado em JavaScript através do método *document.getElementById()*. Como acabamos de conhecer o DOM, vamos escrever uma espécie de *Hello World* DOM! Nosso *Hello World* DOM será uma página *Web* que contém um parágrafo e um *script* que busca o texto do parágrafo e mostra-o em uma mensagem. Analise atenciosamente a página *Web* da listagem a seguir.

LISTAGEM 65 - ACESSANDO O DOM VIA JAVASCRIPT

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Acessando o DOM</title>
  </head>
  <body>

    <!-- um simples parágrafo com nosso
        tradicional Hello World.
        note que possui um id -->
    <p id="paragrafo">Hello World!</p>
```

<!-- Em páginas Web nas quais queremos acessar objetos do DOM através de código JavaScript, o código fonte JavaScript deve ficar depois do HTML, pois o browser começa a renderizar a página de cima para baixo, então quando o script rodar, os objetos do DOM já estarão criados ! Tudo bem, JavaScript junto com HTML é feio, mas quando for necessário, inclua seus scripts sempre antes do fechamento da tag body -->

```

<script type="text/javascript">

    // Aqui estamos acessando o DOM para
    // obter o texto do nosso parágrafo
    // Este comando busca no documento HTML
    // o elemento que possui id "paragrafo"
    // e obtém o conteúdo que está entre
    // sua tag de abertura e fechamento
    var texto = document.

getElementById("paragrafo").innerHTML;
    alert(texto);
    // Resultado: Hello World

</script>
</body>
</html>

```



Mais uma vez vamos ressaltar, código-fonte JavaScript, de acordo com as boas práticas, deve ficar separado da página HTML. Estamos deixando-os juntos somente com a intenção de fornecer exemplos completos. Na Unidade 3, você irá conhecer o AngularJS, framework que promove a segregação do código-fonte JavaScript e HTML da melhor forma possível.

Uma aplicabilidade muito comum do acesso ao DOM através de JavaScript é para validar formulários HTML. Veja, na página Web da listagem a seguir, a utilização de JavaScript para validar um formulário de cadastro de pessoas.

LISTAGEM 66 - VALIDANDO FORMULÁRIOS HTML COM JAVASCRIPT

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Validação de formulários</title>
    </head>
    <body>
        <h1>Formulário de Cadastro</h1>

        <!-- Formulário para cadastro de pessoas
            Note a utilização do atributo name, que faz
            o link entre os elementos do DOM e o
JavaScript-->
        <form id="formulario">
            <p><label>Nome: <input type="text" id="nome">
            </label></p>
            <p><label>e-mail: <input type="text" id="email">
            </label></p>
            <p><input type="submit" value="enviar"></p>
        </form>

        <!-- Em páginas Web nas quais queremos acessar
objetos
            do DOM através de código JavaScript, o código
fonte
            JavaScript deve ficar depois do HTML, pois o
browser
            começa a renderizar a página de cima para
baixo,
            então quando o script rodar, os objetos do DOM
já estarão criados !
            Tudo bem, JavaScript junto com HTML é feio,
            mas quando for necessário, inclua seus scripts
            sempre antes do fechamento da tag body -->
<script type="text/javascript">

    // Declaramos uma função de validação
    var validaForm = function() {
```

```

    // Obtém o valor informado no input nome
    var nome = document.getElementById("nome") .
value;

    // Obtém o valor informado no input email
    var email = document.
getElementById("email").value;

    if (nome == "") {
        alert("É necessário informar o nome!");
    } else if (email == "") {
        alert("É necessário informar o
e-mail!");
    } else {
        // Aqui deve enviar para o servidor!
    }
}

// Aqui fazemos o link entre a ação
submit e
// a nossa função validaForm
document.getElementById("formulario") .
onsubmit = validaForm;
</script>
</body>
</html>

```

Porém, atualmente, a prática de utilizar JavaScript para validar formulários vem sendo substituída pelos novos atributos de validação de formulários disponibilizados pelo HTML5. Por exemplo, para tornar um campo obrigatório, é necessário apenas incluir o atributo *required* em cada *input*, sem necessidade de *scripts*.

LISTAGEM 67 - VALIDAÇÃO DE FORMULÁRIOS EM HTML5

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Validação de formulários HTML5</title>
    </head>
    <body>
        <h1>Formulário de Cadastro</h1>

```

```
<!-- Formulário para cadastro de pessoas  
Note a utilização do atributo name, que  
faz  
    o link entre os elementos do DOM e o  
JavaScript-->  
    <form id="formulario">  
        <p><label>Nome: <input type="text"  
required /></label></p>  
        <p><label>e-mail: <input type="email"  
required /></label></p>  
        <p><input type="submit" value="enviar"></  
p>  
    </form>  
    </body>  
</html>
```

RESUMO DO TÓPICO 2

Neste tópico você aprendeu que:

- O JavaScript é atualmente a linguagem mais utilizada em páginas *Web* do lado do cliente (ou seja, no *browser*).
- O JavaScript é uma linguagem de tipagem dinâmica.
- Tivemos um *overview* sobre os tipos de dados, *arrays*, objetos, e estruturas de controle e principais funções da linguagem JavaScript.
- O JavaScript foi por muito tempo utilizado para fazer a validação de formulários, mas esta prática tornou-se obsoleta em vista dos atributos para validação de formulários fornecidos pelo HTML5.

AUTOATIVIDADE



- 1 Crie uma página *Web* com um parágrafo que tenha um *id*, dentro do parágrafo coloque apenas seu nome (parágrafos são declarados em HTML através da tag <p>).
- 2 Através de um *script* JavaScript, obtenha o texto do parágrafo, que deve conter apenas o seu nome.
- 3 Se a quantidade de letras do seu nome (obtido de dentro do parágrafo através de acesso via DOM, na questão anterior) for um número par, exiba um alerta com o texto “Par”, senão exiba um alerta com o texto “Ímpar”

OBS.: Consulte a listagem 65 em caso de dúvidas.

DESENVOLVENDO UMA APLICAÇÃO WEB COM PHP, JAVASCRIPT E MYSQL

1 INTRODUÇÃO

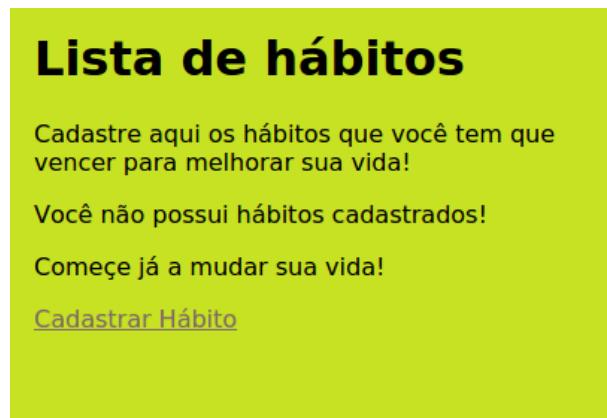
Desenvolvedores *Web* tem que lidar com diferentes tecnologias durante o desenvolvimento de uma página *Web*. Uma linguagem de programação no servidor, uma linguagem de *script* no *browser*, e um banco de dados são os elementos básicos para uma aplicação *Web*. Este tópico destina-se ao desenvolvimento de uma aplicação *Web* usando as tecnologias que aprendemos até agora, PHP, JavaScript e MySQL. Tenha seu computador a mãos ao fazer esta leitura, implemente todas as listagens junto conosco.

2 O APlicativo LISTA DE HÁBITOS

Para que sejamos pessoas mais produtivas e realizadas, o correto seria que cultivássemos hábitos que colaborem positivamente com nossos objetivos e eliminássemos hábitos que nos prejudicam. Por exemplo, imagine que você gostaria de aprofundar seus conhecimentos sobre JavaScript. Você trabalha o dia todo, coloca seus estudos da faculdade em dia quando chega do trabalho, e por fim assiste à televisão à noite para relaxar. Neste caso para colaborar com seus objetivos, seria muito interessante se você cultivasse o hábito de estudar JavaScript e eliminasse o hábito de assistir televisão à noite. Neste tópico, iremos desenvolver um aplicativo que vai lhe ajudar a vencer os maus hábitos!

Na página inicial da nossa aplicação *Web*, aparecerão os hábitos que você tem cadastrados. Inicialmente, não haverá nenhum hábito cadastrado, portanto a lista aparecerá vazia em um primeiro acesso. Neste caso, a aplicação irá exibir uma mensagem de que não há hábitos cadastrados e disponibilizar o *link* para inclusão. O primeiro acesso ao aplicativo resulta na tela mostrada na figura a seguir.

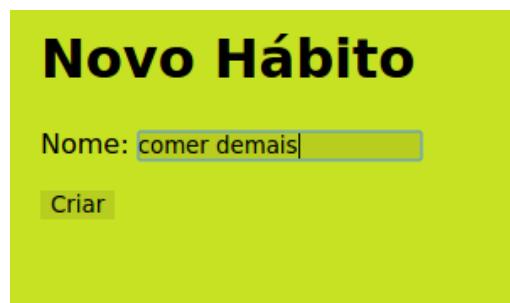
FIGURA 66 – PRIMEIRO ACESSO NO APLICATIVO LISTA DE HÁBITOS



FONTE: O autor

Ao clicar no botão Cadastrar Hábito, a aplicação redirecionará o usuário a uma tela para cadastro de hábitos. A tela será bem simples, terá apenas um campo obrigatório validado através de JavaScript (poderia ser através do atributo *required* do HTML5). A tela de cadastro de hábito deve ficar conforme a seguinte figura.

FIGURA 67 – TELA PARA CADASTRO DE HÁBITOS



FONTE: O autor

Após ter cadastrado o hábito, a aplicação volta para a página inicial, exibindo o hábito cadastrado. Se você cadastrar mais hábitos, uma tabela irá exibi-los na tela inicial da aplicação, conforme a imagem abaixo. Se você conseguiu vencer um hábito, você pode clicar em vencer, então o hábito não será mais exibido na tabela. Se você desistir de lutar contra um hábito, você pode clicar em desistir, também neste caso o hábito não constará mais na tabela da página inicial. A diferença entre a ação vencer e a ação desistir é visível somente por baixo dos panos. Ao clicar em vencer, o *status* do registro que contém o hábito no banco de dados é atualizado para 'V' (venceu), enquanto que ao clicar em desistir, o registro é excluído do banco de dados.

FIGURA 68 – TELA INICIAL COM QUATRO HÁBITOS CADASTRADOS

Lista de hábitos

Cadastre aqui os hábitos que você tem que vencer para melhorar sua vida!

comer demais	Vencer	Desistir
Beber muita cerveja	Vencer	Desistir
Dormir Pouco	Vencer	Desistir
fazer pouco exercício físico	Vencer	Desistir

Continue mudando sua vida!

Cadastre mais hábitos!

[Cadastrar Hábito](#)

FONTE: O autor



E esta será a nossa aplicação Web que implementaremos juntos para reforçar vários dos conceitos de PHP, MySQL e JavaScript que estudamos até agora. Tenha um bom aproveitamento de cada detalhe, analise como as três tecnologias se combinam na construção de uma solução Web. Agora partiremos para o desenvolvimento do aplicativo, iremos apresentar primeiro a construção dos objetos de banco de dados, depois o projeto Web no XAMPP. O código-fonte PHP está todo comentado, de forma bem objetiva para facilitar seu entendimento. Analise com extrema atenção cada linha de código e cada comentário das listagens que seguem até o final deste tópico. Bons estudos!

3 CRIAÇÃO DO BANCO DE DADOS

Nosso aplicativo Lista de hábitos irá precisar de um banco de dados para manter as informações referentes aos hábitos que queremos vencer. Acompanhe as etapas a seguir para criar o banco de dados da nossa aplicação Web.

1. Inicie o XAMPP.
2. Acesse o phpMyAdmin, ou seja, em seu *browser*, acesse a URL <<http://localhost:80/phpmyadmin/>>.
3. No phpMyAdmin, crie um novo banco de dados chamado listadehabit, conforme a figura a seguir.

FIGURA 69 – CRIANDO O BANCO DE DADOS LISTADEHABITO



Base de Dados

Criar base de dados

listadehabit | Agrupamento (Collation) | Criar

⚠️ Nota: Activar as estatísticas aqui pode causar um grande volume de tráfego entre os servidores web e MySQL.

FONTE: O autor

4. No banco de dados listadehabit, crie uma tabela chamada *habito*, com três colunas (*id*, *nome* e *status*), conforme as figuras a seguir (preste atenção na coluna *id*, ela é A_I e *primary*):

FIGURA 70 – CRIANDO A TABELA HABITO

Estrutura | SQL | Pesquisar | Pesquisa por formulário | Exportar | Importar | Operações | Privilégios | Rotinas | Mais

Nenhuma tabela encontrada na base de dados.

Criar tabela

Nome: habito Número de colunas: 3

Executar

FONTE: O autor

FIGURA 71 – CRIANDO AS COLUMNAS DA TABELA HABITO

Procurar | Estrutura | SQL | Pesquisar | Inserir | Exportar | Importar | Privilégios | Operações | Rastreando | Acionadores

Table name: habito Add 1 column(s) Executar

Nome Tipo Tamanho/Valores* Predefinido Agrupamento (Collation) Atributos Nulo Índice A_I Comentários Tipo MIME

id	INT	5	None			PRIMARY	<input checked="" type="checkbox"/>	
nome	VARCHAR	100	None			---	<input type="checkbox"/>	
status	VARCHAR	2	As defined:			---	<input type="checkbox"/>	A

Comentários da tabela: Motor de armazenamento: InnoDB Collation:

PARTITION definition: Guarda

FONTE: O autor

Pronto, temos o banco de dados para nossa aplicação Web construído. Se você quiser dar uma pausa antes de começar a parte Web, finalize o XAMPP.

4 DESENVOLVIMENTO DA PARTE *WEB* DO APLICATIVO

Nesta disciplina, este será o seu primeiro contato com uma aplicação *Web* que contém mais de uma página. Então fique atento a cada detalhe, analise atenciosamente as chamadas da função *header* em PHP. Vamos começar:

1. Em seu diretório de projetos (*htdocs*) crie um diretório chamado *listadehabitos*
2. Dentro do diretório *listadehabitos*, crie um arquivo chamado *styles.css* e insira neste arquivo o conteúdo da listagem a seguir:

LISTAGEM 68 - ARQUIVO CSS DA APLICAÇÃO WEB LISTA DE HÁBITOS

```
.center {
    margin-left: auto;
    margin-right: auto;
    width: 350px;
}

body {
    background-color: #C7E123
}

a {
    font-family: sans;
    color: #7C6B6B
}

p, td, h1, input {
    font-family: sans;
}

input{
    background-color: #B7CE20;
    border: 0;
}

table {
    border-collapse: collapse;
}
```

```
table, td, th {
    border: 1px solid black;
}
```

3. Dentro do diretório listadehabitos, crie um arquivo chamado index.php e insira neste arquivo o conteúdo da listagem a seguir:

LISTAGEM 69 - PÁGINA INICIAL DA APLICAÇÃO WEB LISTA DE HÁBITOS

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <link rel="stylesheet" type="text/css"
        href="styles.css">
        <title>Lista de hábitos</title>
    </head>
    <body>
        <div class="center">
            <h1>Lista de hábitos</h1>
            <p>Cadastre aqui os hábitos que você tem que
            vencer para melhorar sua vida!</p>
            <?php
                // Obtém a lista de hábitos do
                // banco de dados MySQL

                $servidor = "localhost";
                $usuario = "root";
                $senha = "";
                $bancodedados = "listadehabitoo";

                // Cria uma conexão com o banco de dados
                $conexao = new mysqli( $servidor
                    , $usuario
                    , $senha
                    , $bancodedados);
```

```

    // Verifica a conexão
    if ($conexao->connect_error) {
        die("Falha na conexão: " .
            $conexao->connect_error);
    }

    // Executa a query da variável $sql
    $sql = " SELECT id ".
        " , nome ".
        " FROM habito ".
        " WHERE status = 'A'";

    $resultado = $conexao->query($sql);

    // Verifica se a query retornou registros
    if ($resultado->num_rows > 0) {

        ?>
        <br />
        <table class="center">
            <tbody>
                <?
                    // Looping pelos registros retornados
                    while($registro = $resultado->fetch_
assoc()) {
                        ?>

                        <tr>
                            <td><? echo $registro["nome"]; ?></
td>
                            <td><a href="vencerhabito.php?id=<?
echo $registro["id"]; ?>">Vencer</a></td>
                            <td><a href="desistirhabito.php?id=<?
echo $registro["id"]; ?>">Desistir</a></td>
                        </tr>

```

```

<?
} // fim do looping
?>

</tbody>
</table>
<p>Continue mudando sua vida!</p>
<p>Cadastre mais hábitos!</p>
<?
} else {
?>
<p>Você não possui hábitos cadastrados!</p>
<p>Comece já a mudar sua vida!</p>
<?

} // fim do if

// Fecha a conexão com o MySQL
$conexao->close();

?>

<a href="novohabito.php">Cadastrar Hábito</a>
</div>
</body>
</html>

```

4. Dentro do diretório listadehabitos, crie um arquivo chamado novohabito.php e insira neste arquivo o conteúdo da listagem a seguir:

LISTAGEM 70 - PÁGINA PARA CADASTRO DE HÁBITOS

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css"
href="styles.css">

```

```

<title>Lista de hábitos</title>
</head>
<body>
    <div class="center">
        <h1>Novo Hábito</h1>

        <!-- Formulário para cadastro de pessoas
            Note a utilização do atributo name, que
            faz
                o link entre os elementos do DOM e o
            JavaScript-->
        <form id="formulario" action="inserthabito.
        php">
            <p><label>Nome: <input type="text"
            id="nome" name="nome" autofocus /></label></p>
            <p><input type="submit" value="Criar"></
            p>
        </form>
    </div>
    <script type="text/javascript">

        // Declara uma função para
        // validar o formulário
        var validaForm = function() {
            var nomeHabito = document.
            getElementById("nome").value;
            if ("" == nomeHabito) {
                alert("É necessário informar o
                nome do Hábito");
                return false;
            }
        }

        // Aqui declaramos que esta
        // função ocorre sempre no
        // submit do formulário
    
```

```

        document.getElementById("formulario") .
onsubmit = validaForm;
    </script>
</body>
</html>

```

5. Dentro do diretório listadehabitos, crie um arquivo chamado inserthabito.php e insira neste arquivo o conteúdo da listagem a seguir:

LISTAGEM 71 - IMPLEMENTAÇÃO EM PHP PARA INSERIR UM REGISTRO NO MYSQL

```

<?php

$servidor = "localhost";
$usuario = "root";
$senha = "";
$bancodados = "listadehabit";

// Abre a conexão com o banco
// de dados listadehabit
$conexao = new mysqli( $servidor
                      , $usuario
                      , $senha
                      , $bancodados);

// Verifica se houve erro ao
// abrir a conexão
if ($conexao->connect_error) {
    die("A conexão falhou: " . $conexao->connect_
error);
}

// Busca nome que foi recebido
// via get através do formulário
// de cadastro
$nome = $_GET["nome"];

```

```

// Insere o hábito na tabela
// habito do banco de dados
$sql = "INSERT INTO habitos (nome, status)
VALUES ('".$nome."', 'A')";

// Verifica se ocorreu tudo bem
// Caso houve erro, fecha a conexão
// e aborta o programa
if (!$conexao->query($sql) === TRUE) {
    $conexao->close();
    die("Erro: " . $sql . "<br>" . $conexao->error);
}

// Fecha a conexão com o
// Banco de dados
$conexao->close();

// Envia para a página index
// onde aparece a lista de hábitos
// já com o novo hábito cadastrado
header("Location: index.php");

?>

```

6. Dentro do diretório listadehabitos, crie um arquivo chamado vencerhabitoo.php e insira neste arquivo o conteúdo da listagem a seguir:

LISTAGEM 72 - IMPLEMENTAÇÃO EM PHP PARA ATUALIZAR UM REGISTRO NO MYSQL

```

<?php

$servidor = "localhost";
$usuario = "root";
$senha = "";
$bancodados = "listadehabitoo";

```

```

// Cria a conexão
$conn = new mysqli( $servidor
                    , $usuario
                    , $senha
                    , $bancodedados);

// Verifica se conectou
// com sucesso
if ($conn->connect_error) {
    die("Falha na conexão: "
        . $conn->connect_error);
}

// Atualiza o status de A - ativo
// para V - vencido
$id = $_GET["id"];
$sql = " UPDATE habito "
        ." SET status='V' "
        ." WHERE id=".$id;

// Verifica se o comando foi
// executado com sucesso
if (!$conn->query($sql) === TRUE) {
    $conn->close();
    die("Erro ao atualizar: " . $conn->error);
}

// Fecha a conexão
$conn->close();

// Redireciona para index
header("Location: index.php");

?>

```

7. Dentro do diretório listadehabitos, crie um arquivo chamado desistirhabit. php e insira neste arquivo o conteúdo da listagem a seguir:

LISTAGEM 73 - IMPLEMENTAÇÃO EM PHP PARA EXCLUIR UM REGISTRO NO MYSQL

```
<?php

$servidor = "localhost";
$usuario = "root";
$senha = "";
$bancodados = "listadehabitó";

// Cria a conexão
$conn = new mysqli( $servidor
                    , $usuario
                    , $senha
                    , $bancodados);

// Verifica se conectou
// Com sucesso
if ($conn->connect_error) {
    die("A conexão falhou: "
        . $conn->connect_error);
}

// Obtém o id do registro
// que foi recebido via get
$id = $_GET["id"];

$sql = "DELETE FROM habitó WHERE id="
      . $id;

// Executa o comando delete
// da variável $sql
if (!$conn->query($sql) === TRUE) {
    die("Erro ao excluir: "
        . $conn->error);
}
```

```
// Fecha a conexão  
$conn->close();  
  
// Redireciona para a página inicial  
header("Location: index.php")  
  
?>
```

8. Inicie o XAMPP.

9. Em seu *browser*, acesse a URL <<http://localhost:80/listadehabitos/>>.

10. Quando tiver verificado que tudo funciona conforme o esperado, você pode finalizar o XAMPP.



Se você criou todos os arquivos dentro do diretório correto (`listadehabitos`, abaixo de `htdocs`), e digitou todo o código-fonte das listagens acima, sua aplicação Web deve funcionar corretamente. Caso não esteja, confira novamente o código-fonte de forma minuciosa, buscando saber se você não digitou por acaso algum comando de forma errada. O PHP, fornece o número da linha na mensagem de erro, isto ajuda muito na correção de código-fonte quando apresenta problemas.

Parabéns, você conta agora com o conhecimento que serve de base para desenvolver aplicações Web. Desenvolvedores Web são verdadeiros programadores ninja que lidam com diversas tecnologias diferentes em um só projeto. Pode-se notar, para um projeto simples como a nossa lista de hábitos, já utilizamos três tecnologias distintas, PHP, MySQL e JavaScript. A próxima unidade irá focar na apresentação de técnicas mais atualizadas para desenvolvimento de aplicações Web. Até lá!

LEITURA COMPLEMENTAR

Qual a motivação do desenvolvimento de *Software Livre*?

Adelson Junior

Caro desenvolvedor ou qualquer pessoa que se interesse por tecnologia! Hoje em dia, consagrado o amplo sucesso dos *softwares* livres, com a maturidade destes, nos sentimos confortáveis em utilizá-los em nossas tarefas. *Softwares* como o próprio Linux, Apache, Asterisk, Openser são exemplos de maturidade, estabilidade, escalabilidade e Customização baseando-se neste tipo de desenvolvimento de *Software*.

Bem, é bom lembrar que *Software Livre* não é “Cerveja Grátis” mas sim, *Softwares* que têm seu código-fonte aberto e com isso qualquer pessoa pode adaptá-lo às suas necessidades, o tornando melhor. Por isso, não há um conjunto de programadores em uma determinada empresa pagos para desenvolvê-los, mas sim programadores espalhados por todo o Globo. Deste modo: NÃO há RENDA a estes programadores... eu lhe pergunto: Você, que utiliza *Software Livre*, já parou para pensar o que motiva esses caras (programadores) a desenvolver esses *softwares* para nós utilizarmos? O que leva essas almas a gastarem seu tempo, seus conhecimentos, suas habilidades desenvolvendo estes *softwares*?

Simplesmente é algo que não têm preço: o Conhecimento. É o que diz um estudo denominado “*Toward an Understanding of the Motivation of Open Source Software Developers*”. Para entendermos o estudo, precisamos dividir o tema em dois *Posts*. O primeiro vai desmistificar a estrutura por dentro do desenvolvimento e o segundo irá tratar o estudo da motivação dessas pessoas.

Comunidades Open Source

A peça principal para o sucesso de um Desenvolvimento de *Software Livre* (projeto) é a comunidade envolvida. O que é esta “Comunidade”? São Grupo de pessoas que pelos seus interesses em comum se tornam informalmente “parceiros” em um assunto/projeto/busca/ideal específico. São voluntários cuja a motivação os levam a participar e contribuir com o projeto. A comunidade provê ao projeto uma plataforma de compartilhamento de conhecimento com os membros, dessa forma disseminando o conhecimento.

Diferente do desenvolvimento fechado, que há um chefe, que se encarrega de alocar uma equipe designada a programar, onde já está definido quem serão os usuários do sistema em um cenário em que há uma separação clara: usuários e desenvolvedores. No desenvolvimento fechado um usuário não se tornará um desenvolvedor, a menos que quando o chefe achar que ele mereça, ele suba de cargo e venha a tornar-se um programador. No desenvolvimento *Open Source* não há uma distinção clara de quem é usuário e quem é desenvolvedor, qualquer membro da comunidade pode se tornar um programador em potencial, bastando

este ter o conhecimento para tal, sendo que, como membro da comunidade, ele buscou e busca conhecimento e contribua para o *software*. Esta “transição” de, digamos, “*status*”, os autores do estudo denominam de “*role transformation*”.

Dentro da comunidade temos uma espécie de Distinção Social. No começo pessoas envolvidas criam uma comunidade envolta de um projeto particular motivados por um interesse compartilhado em utilizar e/ou desenvolver o *Software*. Membros da comunidade assumem por si mesmos as “*roles*” de acordo com seus interesses pessoais no projeto. Estudando quatro projetos *Open Source*, os autores do estudo puderam classificar oito (8) “*roles*”:

- **Project Leader** (Líder de Projeto): A pessoa que iniciou o projeto. O *Project Leader* é o responsável pela visão e os rumos que o projeto seguirá. (Mark Spencer é o criador e assim o *Project Leader* do Asterisk).
- **Core Member**: São responsáveis por guiar e coordenar o desenvolvimento do projeto. São aquelas pessoas que estão envolvidas com um projeto durante muito, muito tempo, fizeram e fazem contribuições significantes para o desenvolvimento e a evolução do *Software*.
- **Active Developer**: São os que regularmente contribuem com novas *features* e *bug fixes* para o *software*. São as maiores forças do desenvolvimento do *Software Livre*.
- **Peripheral Developer**: São pessoas que ocasionalmente contribuem com novas funcionalidades ou *features* para o sistema. Sua contribuição é irregular e o período de envolvimento é curto e esporádico.
- **Bug Fixer**: São pessoas que corrigem os *bugs* encontrados por elas mesmas ou reportados por outros membros. *Bug Fixer* precisam ler e entender algo do código do *software*.
- **Bug Reporter**: Descobrem e reportam *bugs*. (“Hmm’. Quando o seu Firefox bugou e abriu aquela tela de *Bug Report*, você se tornou um *Bug Reporter*? Ou cancelou?). Ele não corrige o *bug* e não possui conhecimento do código-fonte. São como se fossem a área de *Quality Assurance* do projeto. (Legal não!?) Por isso, para um *Software* com muita qualidade e com poucos *bugs*, é essencial vários membros que contribuam como *Bug Reporter* (Lembre-se disso!).
- **Reader**: São usuários ativos do *Software*. Não somente usam o *Software*, mas tentam entender como o *software* funciona, lendo o código, documentação... Dado a alta qualidade dos *Softwares Livres*, alguns *Readers*, para aprender a programar, leem o código do *software* ou usam-no como referência a desenvolvimento de *softwares* similares.
- **Passive User**: Apenas utilizam o *software* para alguma determinada necessidade. (A maioria de nós) São atraídos a utilizá-los pela alta qualidade e o potencial de ser alterado quando surgir a necessidade.

Essas oito *Roles* não são REGRAS em todas as comunidades e a porcentagem de membros em cada uma varia. Apesar de não existir uma distinção hierárquica, a estrutura da comunidade não é completamente plana. Dependendo de qual role o membro participa, mais ou menos influência este terá no *Software* e na Comunidade. Considere as camadas da imagem acima como os “*roles*”. Quanto mais perto do centro, mais influência este terá na comunidade e no *Software*. Em outras palavras, a atividade do *Project Leader* influenciará mais membros do que o *Core Member*, no qual terá mais influência do que o *Active Developer* e por aí vai.

Como exemplo, citamos a responsabilidade do *RoadMap* que é uma atividade do *Project Leader* que impacta toda a comunidade e, partindo para outra “camada”, uma alteração em um determinado ponto do código pelo *Bug Fixer* não necessariamente impactará para todos, mas para aqueles que utilizam a *feature* que for corrigida. Não que uma *role* seja “mais importante” que a outra. As camadas externas têm sua importância, tanto social dentro da comunidade, como psicológica, atraindo e motivando mais pessoas, podendo esses serem potencialmente membros mais ativos e consequentemente, aprofundando seus conhecimentos e contribuindo maisativamente, alcançando outros *roles*. Como analogia, imaginemos as camadas do centro como atores de uma peça de teatro, e o restante como a plateia... o que são os atores sem aplausos? Não serão motivados? O grau de influência dentro da comunidade pode ser considerado mais uma motivação? Gostaria que expressasse suas opiniões para discutirmos, e com o segundo *post* teremos a visão do estudo sobre a questão.

FONTE: Disponível em: <<https://ensinar.wordpress.com/2008/09/25/qual-a-motivacao-do-desenvolvimento-de-software-livre/>>. Acesso em: 26 jan. 2014.

RESUMO DO TÓPICO 3

Neste tópico, você aprendeu que:

- Uma aplicação *Web* contempla diversas tecnologias em um só projeto. Normalmente, uma aplicação *Web* conta com no mínimo uma tecnologia para processamento no servidor (no caso usamos o PHP), uma para processamento na própria página, que roda no *browser* (no caso usamos JavaScript) e um banco de dados (no caso usamos o MySQL). Isto demanda que os desenvolvedores *Web* tenham conhecimento de diversas tecnologias.
- Uma técnica muito comum no desenvolvimento *Web* é criar uma página para mostrar as informações e outras páginas separadas para fazer a interação com o banco de dados, e fazer a ligação entre elas através de *links*, *buttons* e formulários.
- Um projeto *Open Source* é um projeto cujo código-fonte está exposto para qualquer pessoa interessada. Normalmente, o desenvolvimento de um projeto *open source* conta com diversos desenvolvedores. Há vários projetos *open source* no mercado, e você pode participar deles.

AUTOATIVIDADE



- 1 Você participaria de algum projeto *Open Source*? Qual?
- 2 O que lhe motivaria a participar de um projeto *Open Source*?
- 3 E quando outras pessoas começarem a fazer dinheiro usando seu projeto *Open Source*, você se sentiria bem ou mal?

A WEB NA ATUALIDADE DESACOPLAMENTO, API E JSON

OBJETIVOS DE APRENDIZAGEM

Após estudar esta unidade, você será capaz de:

- conceituar coesão e acoplamento no contexto do desenvolvimento de *software*;
- conhecer e aplicar as práticas para o desenvolvimento de *software* modularizado;
- construir um sistema *web* baseado em camadas;
- separar a camada de acesso a dados e serviços da camada de UI (User Interface) em um sistema *web*;
- implementar páginas *web single page application*.

PLANO DE ESTUDOS

Esta unidade de ensino está dividida em três tópicos, sendo que no final de cada um deles, você encontrará atividades que contribuirão para a apropriação dos conteúdos.

TÓPICO 1 – COESÃO E ACOPLAMENTO

TÓPICO 2 – DESIGN RESPONSIVO

TÓPICO 3 – SINGLE PAGE APPLICATION

COESÃO E ACOPLAMENTO

1 INTRODUÇÃO

Durante nossos estudos, na Unidade 2, aprendemos a construir uma aplicação *Web* completa. O aplicativo Lista de Hábitos que construímos é um cadastro completo que nos permite criar, atualizar e excluir registros em um Banco de Dados. Porém, ele foi implementado de uma maneira contraindicada. Se você conhece orientação a objetos e/ou *Design Patterns* (Padrões de projetos), certamente você já ouviu falar em coesão e acoplamento. Caso não tenha ouvido falar, não tem problema, este tópico irá abordar exatamente isto.

2 COESÃO E ACOPLAMENTO

Um projeto de *software*, normalmente, envolve muitas questões a serem tratadas. Uma pessoa que não se organiza antes de começar a programar certamente vai tratar destas questões como UM grande problema. Se pensarmos com calma, começaremos a perceber que este grande problema pode ser dividido em VÁRIOS outros problemas menores, e que desta forma podemos pensar em cada um destes problemas de forma específica, tratando-os individualmente. Consequentemente, cada problema tratado individualmente se torna uma parte do programa. Um programa bem modularizado trata cada questão individualmente, enquanto um programa mal modularizado, ou seja, um programa onde todas as questões foram tratadas como UM grande problema, trata todas as questões de forma junta e misturada.

Quando você trabalha com linguagem de programação, você deve entender o que é uma unidade de programa. Em linguagens puramente orientadas a objetos, como o C#, o C++ e o Java, a menor unidade de programa são as classes. Em linguagens procedurais (não orientadas a objetos), a menor unidade de programa são procedimentos e funções.

Supondo que você esteja desenvolvendo uma solução para um problema (para a empresa na qual você trabalha) em PHP sem utilizar orientação a objetos, a menor unidade de programa será a *function*. Você já idealizou uma maneira de resolver o problema, e dividiu-a em três etapas distintas. Você tem, neste caso, duas opções. A primeira é desenvolver a solução em uma grande *function*, já que você sabe exatamente o que fazer, e assim você consegue entregar sua tarefa pronta mais rápido. A segunda é desenvolver a solução em três *functions*, e esforçar-se para que elas também possam ser utilizadas por outros desenvolvedores. Se você

seguiu a primeira opção, sua solução não ficou modularizada. Se você seguiu a segunda opção, a sua solução ficou modularizada e seus colegas de trabalho ficarão contentes com você.

A modularização de código-fonte possui inúmeras vantagens, mas duas são as principais: o aumento da coesão e a diminuição do acoplamento. Mas o que é isto? Confira abaixo:

- Coesão: é o grau de direcionamento de uma unidade de programa para/com um (e somente um) objetivo específico.
- Acoplamento: é o grau de conhecimento exigido sobre uma determinada unidade de programa para poder utilizá-la.

Quando falamos em coesão e acoplamento, uma coisa vai contra a outra. Um código-fonte coeso apresenta baixo grau de acoplamento, enquanto um código-fonte com alto grau de acoplamento possui sua coesão condenada. É importantíssimo entender que a coesão é uma vantagem e o acoplamento é uma desvantagem. Ao programar, devemos sempre buscar desenvolver um código-fonte coeso e com baixo grau de acoplamento.

Agora, vamos relembrar nossos estudos da unidade 2, mais especificamente o tópico três, na parte em que construímos a aplicação *Web Lista de Hábitos*. Visualize, a seguir, o código-fonte do arquivo *index.php* que desenvolvemos na época e reflita: o código-fonte é coeso? Veja como é grande e confuso este código-fonte, imagine fazer alguma manutenção ou adicionar alguma característica nova. Note também que ele é uma espécie de 'faz tudo' para que na página inicial a nossa aplicação apareça no *brower*. Sem dúvida, a resposta da nossa reflexão é NÃO, o código-fonte do arquivo *index.php* da nossa aplicação *Web Lista de hábitos* não é coeso, e apresenta alto grau de acoplamento. Infelizmente, por muito tempo, os desenvolvedores *Web* trabalhavam assim, desamparados de padrões de projeto e com poucas possibilidades para modularizar seu código-fonte. Atualmente, porém, contamos com arquiteturas, padrões e tecnologias que nos ajudam a superar estes problemas.

LISTAGEM 74 - PÁGINA INICIAL DA APLICAÇÃO LISTA DE HÁBITOS, EXEMPLIFICANDO A FALTA DE MODULARIZAÇÃO NO DESENVOLVIMENTO WEB

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <link rel="stylesheet" type="text/css"
        href="styles.css">
        <title>Lista de hábitos</title>
    </head>
    <body>
        <div class="center">
            <h1>Lista de hábitos</h1>
            <p>Cadastre aqui os hábitos que você tem que
            vencer para melhorar sua vida!</p>
            <?php
                // Obtém a lista de hábitos do
                // banco de dados MySQL

                $servidor = "localhost";
                $usuario = "root";
                $senha = "";
                $bancodedados = "listadehabitoo";

                // Cria uma conexão com o banco de dados
                $conexao = new mysqli( $servidor
                    , $usuario
                    , $senha
                    , $bancodedados);

                // Verifica a conexão
                if ($conexao->connect_error) {
                    die("Falha na conexão: " .
                    $conexao->connect_error);
                }

```

```
// Executa a query da variável $sql
$sql = " SELECT id ".
        " , nome ".
        " FROM habito ";

$resultado = $conexao->query($sql);

// Verifica se a query retornou registros
if ($resultado->num_rows > 0) {

    ?>
<br />
<table class="center">
    <tbody>
        <?
            // Looping pelos registros retornados
            while($registro = $resultado->fetch_
assoc()) {
                ?>

                <tr>
                    <td><? echo $registro["nome"]; ?></
td>
                    <td><a href="vencerhabito.php?id=<?
echo $registro["id"]; ?>">Vencer</a></td>
                    <td><a href="desistirhabito.php?id=<?
echo $registro["id"]; ?>">Desistir</a></td>
                </tr>

                <?
            } // fim do looping
        ?>

    </tbody>
</table>
<p>Continue mudando sua vida!</p>
```

```

<p>Cadastre mais hábitos!</p>
<?
} else {
?>
<p>Você não possui hábitos cadastrados!</p>
<p>Comece já a mudar sua vida!</p>
?>

} // fim do if

// Fecha a conexão com o MySQL
$conexao->close(); ?>

<a href="novohabito.php">Cadastrar Hábito</a>
</div>
</body>
</html>

```

O que podemos fazer para melhorar a qualidade do código-fonte da nossa aplicação *Web* Lista de Hábitos? Em primeiro lugar, o que temos a fazer é estudar as arquiteturas, os padrões e *frameworks* que irão nos auxiliar na modularização da nossa aplicação *Web*. Ao passo que vamos conhecendo todas estas coisas, iremos criar uma segunda versão da nossa aplicação *Web* Lista de Hábitos.



Framework: traduzido do Inglês para o português, significa estrutura. Como exemplo, podemos citar uma estrutura pré-moldada para a construção de prédios e galpões. No contexto do desenvolvimento *Web*, os *frameworks* são geralmente bibliotecas que você adiciona ao seu projeto para usufruir de um modelo para organizar seu código-fonte.



Programador Ninja

Desenvolvedores *Web* precisam dominar diversas tecnologias, por exemplo, na primeira versão do nosso aplicativo Lista de Hábitos, utilizamos três tecnologias distintas, o PHP, o JavaScript e o MySQL. E para desenvolver uma boa aplicação *Web*, além de conhecer diversas tecnologias, o desenvolvedor *Web* deve conhecer arquiteturas, padrões de projetos e frameworks que propiciem a modularização de seu código-fonte. Não é à toa que falamos que desenvolvedores *Web* são verdadeiros ninjas!

E por onde começar os ajustes de nossa aplicação *Web* Lista de Hábitos? Em primeiro lugar, temos que ter em mente que temos um grande problema, e precisamos dividi-lo em vários problemas menores. O primeiro problema a ser percebido é que as páginas *Web* da nossa aplicação não têm uma responsabilidade exata, elas fazem de tudo, desde a escrita do HTML ao acesso ao Banco de Dados, tudo misturado no mesmo código-fonte. A partir disto, podemos afirmar que elas não são coesas, pois não estão direcionadas a um objetivo específico.

3 JSON E ARQUITETURA REST

A primeira parte da nossa aplicação *Web* Lista de Hábitos que iremos separar é o acesso ao Banco de Dados. Para separar o acesso ao Banco de Dados do código-fonte de *front-end* (HTML), iremos introduzir o conceito de API REST. Uma API REST é uma implementação baseada nos métodos do protocolo HTTP, que você conheceu na Unidade 1 (*get*, *put*, *post* e *delete*). As APIs REST precisam retornar dados ao sistema que a utiliza. Atualmente, os formatos mais utilizados são o Json e o XML. Em nossos estudos, utilizaremos o formato Json.

O padrão Json possui uma escrita simplificada e trafega na *Web* de forma mais leve do que o XML, por este motivo vem se tornando cada vez mais interessante. No formato Json, podemos citar duas estruturas de dados principais, os objetos e os *arrays*.

Os objetos Json são estruturas que agrupam propriedades. Propriedades são estruturas de dados do tipo chave/valor. Os objetos Json são declarados através do literal-chaves ({}). As propriedades são separadas por vírgulas (,), e os nomes das chaves são separados do valor por eles indexados através do literal dois pontos (:). O valor das propriedades deve estar entre aspas, caso não seja numérico, *array* ou objeto, e a chave das propriedades pode ou não estar entre aspas.

LISTAGEM 75 - EXEMPLO DE OBJETO JSON

```
{
    "nome": "Luiz de Souza",
    "hobbie": "Video Game",
    "datanascimento": "08\05\1974 12:20:03 pm"
}
```

Os *arrays* Json são estruturas que armazenam uma coleção de elementos. São declarados através do literal colchetes ([]) e seus elementos são separados por vírgulas.

LISTAGEM 76 - EXEMPLO DE ARRAY JSON

```
[1, 2, "tres", 4]
```

Uma estrutura Json pode alcançar formas complexas, porém, ainda assim, a formatação deste padrão proporciona uma boa legibilidade. No exemplo a seguir, podemos ver um objeto Json, no qual há uma propriedade que é um *array*.

LISTAGEM 77 - OBJETO JSON COM UMA PROPRIEDADE ARRAY

```
{
    "nome": "Eduardo Machado",
    "hobbies": ["video game", "cachorros", "t\u00e3nis de mesa"],
    "datanascimento": "08\05\1991 12:30:04 pm"
}
```



t\u00e3nis de mesa?

Alguns programas, assim como a maioria dos *browsers*, fazem o escape das *strings* de Json. Neste caso, ele substituiu o caractere e com circunflexo (ê) pelo código *unicode* (\u00ea). As barras da propriedade datanascimento também possuem escape.

É importante perceber que Json não é uma linguagem de programação, é simplesmente um padrão que visa facilitar o envio, recebimento e até mesmo o armazenamento de informações. Ele surgiu como uma alternativa ao XML e, devido à escrita simplificada e à transmissão leve, ganhou muito espaço no contexto dos sistemas *Web*.

Agora que você já conhece o padrão Json, vamos conhecer a arquitetura REST. A arquitetura REST tem como objetivo, no contexto dos sistemas *Web*, promover a separação do código-fonte responsável por manipular os dados do sistema do código-fonte de apresentação do sistema (telas do sistema, ou páginas *Web*). Deste ponto de vista, analisando em termos de macro contexto, seu sistema *Web* vai possuir duas camadas, a API REST (camada de manipulação de dados) e o *front-end* (páginas *Web* que o usuário utiliza). Perceba que a utilização de uma API REST torna o código-fonte de sua aplicação *Web* mais coeso, pois trata de um grande problema (um sistema *Web* manipulando informações) separando-o em dois problemas menores (uma API e um *front-end*).

Na unidade 1 do Caderno de Estudos desta disciplina, você conheceu o protocolo HTTP, porém apenas na teoria, não na prática. A forma mais comum de se implementar uma arquitetura REST é fazendo uso do protocolo HTTP, utilizando os métodos *get*, *post*, *put* e *delete*. Define-se um recurso sobre o qual as operações devem ser efetuadas (por exemplo, hábito), e para cada um dos métodos HTTP citados, define-se uma operação de manutenção deste recurso, conforme abaixo:

- *get*: indica que queremos obter informações do recurso que estamos acessando;
- *post*: indica que queremos criar um novo recurso;
- *put*: indica que queremos alterar um recurso existente;
- *delete*: indica que queremos excluir um recurso existente.

Ou seja, em nossa aplicação Lista de Hábitos, teremos (assim que implementarmos nossa API REST) um recurso chamado hábito, e quando acessamos este recurso, através do método HTTP *get*, estamos consultando um hábito, ou todos os hábitos caso não seja especificado nenhum hábito como parâmetro. Quando acessamos o recurso hábito, através do método HTTP *post*, estamos inserindo um novo hábito. Quando acessamos o recurso hábito, através do método HTTP *put*, estamos alterando informações de um hábito existente. E por fim, quando acessamos o recurso hábito, através do método *delete*, estamos excluindo um hábito. Ansioso para iniciar a implementação de nossa API REST para a aplicação Lista de Hábitos? Vamos começar agora!

1. Em seu diretório de projetos (*htdocs*), crie um diretório chamado *listadehabitorest-api*
2. Dentro do diretório *listadehabitorest-api* crie um arquivo chamado *habito.php* e insira neste arquivo o código-fonte da listagem abaixo.

LISTAGEM 78 - INICIANDO A IMPLEMENTAÇÃO DO RECURSO HÁBITO DA API REST DA APLICAÇÃO WEB LISTA DE HÁBITOS

```
<?php

/*
 * Função que converte os parâmetros
 * de requisições HTTP
 * POST e PUT em um Hábito
 *
 */
function f_parametro_to_habito(){

    // Obtém o conteúdo da requisição
    $dados = file_get_contents("php://input");

    // Converte para Json e retornar
    $habito = json_decode($dados, true);
    return $habito;
}

/*
 * Função que retorna os hábitos
 *
 */
function f_select_habito(){

    echo "Esta função irá buscar hábitos!";
}

/*
 * Insere um novo hábito na tabela habitos
 *
 */
function f_insert_habito(){
```

```
echo "Esta função irá inserir um novo hábito!";  
}  
  
/*  
 * Atualiza um hábito existente  
 *  
 */  
function f_update_habito(){  
  
    echo "Esta função irá alterar um hábito existente!";  
}  
  
/*  
 * Exclui um hábito existente  
 *  
 */  
function f_delete_habito(){  
  
    echo "Esta função irá deletar um hábito existente!";  
}  
  
// A variável de servidor REQUEST_METHOD  
// contém o nome do método HTTP através  
// qual o arquivo solicitado foi  
// acessado  
$metodo = $_SERVER['REQUEST_METHOD'];  
  
// Verifica qual ação a ser tomada  
// de acordo com o método HTTP  
// que foi utilizado para acessar  
// este recurso  
switch ($metodo) {  
  
    // Se foi GET  
    // deve consultar  
case "GET":
```

```

    f_select_habito();
    break;

    // Se foi POST
    // deve inserir
    case "POST":
        f_insert_habito();
        break;

    // Se foi put
    // deve alterar
    case "PUT":
        f_update_habito();
        break;

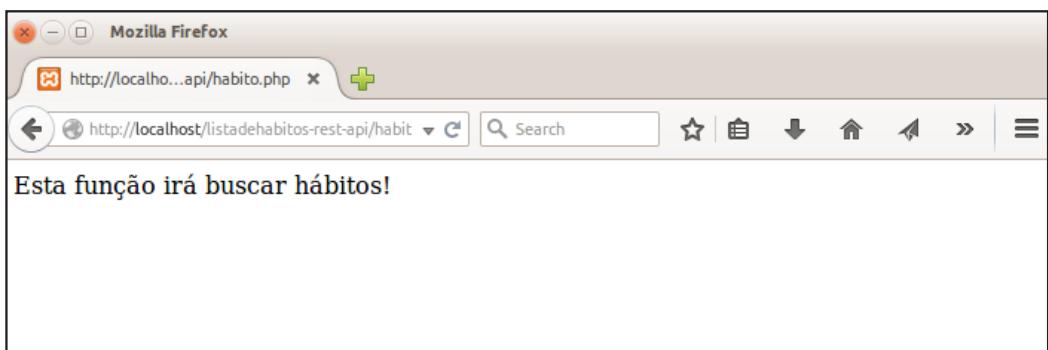
    // Se foi delete
    // deve excluir
    case "DELETE":
        f_delete_habito();
        break;
}

?>

```

3. Inicie o XAMPP.
4. Acesse a URL <http://localhost:80/listadehabitostest-api/habito.php> em seu browser, se tudo estiver certo, deve estar conforme a figura a seguir.

FIGURA 72 – TESTANDO A API REST DA APLICAÇÃO LISTA DE HÁBITO



FONTE: O autor

5. Após entender o que vamos explicar abaixo, você pode finalizar o XAMPP.

Você já deve ter identificado que o arquivo ainda não está completamente implementado, ele conta apenas com a declaração das funções principais. Porém, a lógica do corpo de cada função iremos implementar mais adiante. Agora, vamos entender o que isto tudo significa. Abra o arquivo `habito.php` que você acabou de criar e acompanhe a explicação contida nos tópicos abaixo para entender os detalhes do que ele se propõe a fazer.

- A primeira função declarada no arquivo `habito.php` é a `f_parametro_to_habito()`, que tem como objetivo converter os parâmetros recebidos através das requisições HTTP *post* e *put*. Ainda não estamos utilizando-a, mas iremos utilizá-las mais tarde quando implementarmos as funções `f_insert_habito()` e `f_update_habito()`.
- A segunda função encontrada no recurso hábito, `f_select_habito()`, tem por objetivo retornar os hábitos cadastrados na tabela `habito` (no MySQL) no formato Json. Iremos implementá-la logo em seguida.
- A terceira função, `f_insert_habito()`, será responsável por inserir o hábito recebido como parâmetro na tabela `habito` no MySQL.
- A quarta função, `f_update_habito()`, irá atualizar no MySQL o hábito recebido como parâmetro.
- A quinta função declarada no recurso hábito, `f_delete_habito()`, irá excluir do Banco de Dados o hábito informado como parâmetro.
- Logo após declarar todas as funções, declara-se a variável `$metodo`, e inicializa-se ela com o valor do parâmetro de servidor `REQUEST_METHOD`, que indica através de qual método HTTP o arquivo foi acessado. O parâmetro `REQUEST_METHOD`, retorna em maiúsculo o nome do método HTTP que foi utilizado para acessar o recurso. Por exemplo, se acessarmos o arquivo `habito.php` através do método *get*, o valor de `REQUEST_METHOD` será “GET”, se acessarmos o arquivo `habito.php` através do método *post*, o valor de `REQUEST_METHOD` será “POST”, e assim sucessivamente.
- E por último, analisamos o valor da variável `$metodo`, e com base nele, tomamos uma ação específica. Se o valor for “GET”, executamos a função `f_select_habito()`, se for “POST”, executamos a função `f_insert_habito()`, se for “PUT” executamos a função `f_update_habito()` e, por fim, se for “DELETE”, executamos a função `f_delete_habito()`.
- E em nosso teste percebemos que, como o *browser* acessa arquivos no servidor *Web*, sempre através do método HTTP *get*, ao acessar a URL `<http://localhost:80/listadehabitost-rest-api/habito.php>` a função `f_select_habito()` é executada, e o texto “Esta função irá buscar hábitos!” é impresso na página através do comando `echo`.

Note que da forma que fragmentamos nosso código-fonte em funções temos um baixo grau de acoplamento, portanto um código-fonte coeso, pois cada função possui uma responsabilidade bem específica e atende a um objetivo único e pontual. Para dar continuidade à implementação do recurso hábito, temos que implementar a lógica interna de cada função, vamos começar pela `f_select_habito()`. Edite o arquivo `habito.php` com seu editor de texto e substitua o código-fonte da função `f_select_habito()` pelo código-fonte da listagem abaixo.



Para implementar a função a seguir, você deverá ter implementado o Banco de Dados da aplicação Web Lista de Hábitos, que implementamos no Tópico 3 da Unidade 2. Caso você não tenha o Banco de Dados implementado, volte ao segundo subtítulo do Tópico 3 da Unidade 2 (Criação do Banco de Dados) e crie o Banco de Dados.

LISTAGEM 79 - IMPLEMENTAÇÃO DA FUNÇÃO F_SELECT_HABITO() DO RECURSO HÁBITO

```

/*
 * Função que retorna os hábitos
 *
 */
function f_select_habito(){

    // cria uma cláusula WHERE com os
    // parâmetros que foram
    // recebidos através da requisição
    // HTTP get
    $queryWhere = " WHERE ";
    $primeiroParametro = true;
    $parametrosGet = array_keys($_GET);
    foreach ($parametrosGet as $param) {
        if (!$primeiroParametro){
            $queryWhere .= " AND ";
        }
        $primeiroParametro = false;
        $queryWhere .= $param." = '".$_GET[$param]."'";
    }

    // Obtém a lista de hábitos do
    // banco de dados MySQL
    $servidor = "localhost";
    $usuario = "root";
    $senha = "";
    $bancoDados = "listadehabit";
}

```

```
// Cria uma conexão com o banco de dados
$conexao = new mysqli( $servidor
                      , $usuario
                      , $senha
                      , $bancodedados);

// Verifica a conexão
if ($conexao->connect_error) {
    die("Falha na conexão: " .
        $conexao->connect_error);
}

// Executa a query da variável $sql
$sql = " SELECT id ".
        "         , nome ".
        "         , status ".
        "     FROM habito ";

// utiliza o where criado com base
// nos parâmetros do GET
if ($queryWhere != " WHERE ") {
    $sql .= $queryWhere;
}

// Executa a query
$resultado = $conexao->query($sql);

// Verifica se a query retornou registros
if ($resultado->num_rows > 0) {

    // Inicializa o array para
    // a formação dos objetos JSON
    $jsonHabitoArray = Array();
    $contador = 0;
    while($registro = $resultado->fetch_assoc()) {
        // Monta um objeto Json
        // através de um array associativo ,

```

```

        // ou seja, indexado através de
strings

    $jsonHabito = Array();
    $jsonHabito["id"] = $registro["id"];
    $jsonHabito["nome"] = $registro["nome"];
    $jsonHabito["status"] =
$registro["status"];
    $jsonHabitoArray[$contador++] =
$jsonHabito;
}

// Transforma o array com
// os resultados da query
// em um array Json e imprime-o
// na página
echo json_encode($jsonHabitoArray);

} else {
    // Se a query não retornou
    // registros, devolve um
    // array Json vazio
    echo json_encode(Array());
}

// Fecha a conexão com o MySQL
$conexao->close();

}

```

A princípio, a função parece ser complexa, mas vamos analisar cada detalhe dela. Vamos começar pelas novidades, temos três grandes novidades para explorar nesta função.

- **\$_GET:** A variável `$_GET` armazena em um *array* os parâmetros passados via *get* na requisição da página. Para passar parâmetros via *get* é necessário informá-los no final da URL, após um ponto de interrogação (?) separados por um E comercial (&). Por exemplo, se quisermos efetuar uma requisição via *get* para o recurso hábito, passando o *id* 2 e o *status* "A" como parâmetros, podemos acessar através de nosso *browser* a URL <<http://localhost:80/listadehabitos-rest-api/habito.php?id=2&status=A>>. Neste caso, a variável `$_GET` teria como conteúdo o *array* `Array ([id] => 2 [status] => A)`. Teste isto em seu *browser*.

- **array_keys:** No PHP, temos o conceito de *arrays* associativos, que são os *arrays* indexados através de *strings*. A função `array_keys()` retorna um novo *array* cujos elementos são os índices do *array* que ela recebe como parâmetro. Por exemplo, uma chamada à função `array_keys()` passando como parâmetro um *array* cujo conteúdo é `Array ([id] => 2 [status] => A)`, resulta em um novo *array* cujo o conteúdo é `Array ([0] => id [1] => status)`.
- **json_encode:** converte para Json o objeto ou *array* passado como parâmetro.

Agora, vamos analisar o que a função `f_select_habito()` efetivamente faz. Em primeiro lugar, ela obtém os parâmetros passados via `get` na requisição e formula uma cláusula *where* (SQL) com base neles. Por exemplo, ao acessar a URL <<http://localhost:80/listadehabitostest-api/habito.php?id=2&status=A>>, ela está produzindo a cláusula `WHERE id = '2' AND status = 'A'`, que ela irá utilizar para executar a *query*. Por fim, a função armazena todo o resultado da *query* em um *array* e o retorna no formato Json. Vamos testar nossa API neste ponto? Siga as etapas abaixo.

1. Inicie o XAMPP, caso não esteja iniciado.
2. Em seu *browser*, acesse o phpMyAdmin (<<http://localhost:80/phpmyadmin/>>), selecione o Banco de Dados `listadehabito` e execute a query `SELECT * FROM habito`, conforme a figura a seguir.

FIGURA 73 - SELECIONANDO OS REGISTROS DA TABELA HABITO

The screenshot shows the phpMyAdmin interface with the following details:

- Toolbar:** Procurar, Estrutura, SQL, Pesquisar, Insere, Exportar, Importar, Privilégios, Operações, Rastreando, Mais.
- Message Bar:** A mostrar registos de 0 - 3 (4 total, A consulta demorou 0.0006 segundos.)
- Query Editor:** SELECT * FROM habito
- Result Preview:**

	Id	nome	status
<input type="checkbox"/>	55	comer demais	A
<input type="checkbox"/>	56	Beber muita cerveja	A
<input type="checkbox"/>	57	Dormir pouco	A
<input type="checkbox"/>	58	Fazer pouco exercício físico	A
- Buttons:** Número de registos: 25, Filtrar registos: Pesquisar esta tabela, Ordenar por chave: Nenhum, + Opções, Perfil [Na linha] [Edita] [Explicar SQL] [Criar código PHP] [Actualizar].
- Action Buttons:** Todos, Com os seleccionados: Muda, Apagar, Exportar.
- Bottom Navigation:** Número de registos: 25, Filtrar registos: Pesquisar esta tabela, Operações resultantes das consultas, Vista de impressão, Vista de impressão (com texto inteiro), Exportar, Mostrar gráfico, Criar visualização.

FONTE: O autor

3. Em seu *browser*, acesse a aplicação Web Lista de Hábitos, através da URL <<http://localhost:80/listadehabitostest-api/habito.php?id=2&status=A>> e clique sobre o *link* vencer de um dos hábitos, após isto, re-efetue a consulta da etapa anterior, e veja que o *status* está 'V' para o hábito que você venceu, conforme as figuras a seguir.

FIGURA 74 – APLICAÇÃO LISTA DE HÁBITOS COM OS HÁBITOS CADASTRADOS, IREMOS CLICAR SOBRE O LINK VENCER DE UM DOS HÁBITOS

nome	Vencer	Desistir
comer demais	Vencer	Desistir
Beber muita cerveja	Vencer	Desistir
Dormir pouco	Vencer	Desistir
Fazer pouco exercício físico	Vencer	Desistir

Continue mudando sua vida!
Cadastre mais hábitos!
[Cadastrar Hábito](#)

FONTE: O autor

FIGURA 75 – REEFETUANDO A CONSULTA DA ETAPA 2 APÓS TER VENCIDO UM HÁBITO

Opções	Id	nome	status
<input type="checkbox"/> Edita <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	55	comer demais	A
<input type="checkbox"/> Edita <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	56	Beber muita cerveja	A
<input type="checkbox"/> Edita <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	57	Dormir pouco	V
<input type="checkbox"/> Edita <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	58	Fazer pouco exercício físico	A

Número de registo: 25 Filtrar registo: Pesquisar esta tabela

Operações resultantes das consultas

Vista de impressão Vista de impressão (com texto inteiro) Exportar Mostrar gráfico Criar visualização

FONTE: O autor

4. Agora, vamos ao que interessa à API. Em seu *browser*, acesse a URL <<http://localhost:80/listadehabitost-rest-api/habito.php>> e confira, um Json array contendo todos os hábitos da tabela habito é retornado, conforme a figura a seguir.

FIGURA 76 – ACESSANDO A API REST SEM INFORMAR PARÂMETROS



FONTE: O autor

5. Agora, vamos filtrar nossa consulta, retornando apenas os hábitos vencidos. Para isto, precisamos informar um parâmetro status=V, portanto, acesse a URL <<http://localhost:80/listadehabitosts-rest-api/habito.php?status=V>> e veja se foram retornados somente os hábitos vencidos (neste caso, somente um).

FIGURA 77 – ACESSANDO A API REST FILTRANDO PELO STATUS



FONTE: O autor

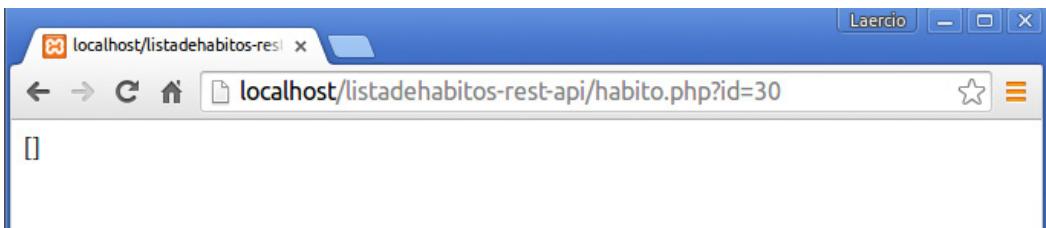
6. Agora, vamos filtrar nossa consulta através do id do hábito. Para isto, precisamos informar um parâmetro id, portanto, acesse a URL <<http://localhost:80/listadehabitosts-rest-api/habito.php?id=55>> (substituindo 55 pelo id de algum hábito constante na sua tabela de hábitos) e veja somente se o hábito correspondente a este id foi retornado dentro do array Json. Em seguida, faça o mesmo, porém informando um id não existente na sua tabela de hábitos, e veja que um Json array vazio foi retornado, conforme as imagens a seguir.

FIGURA 78 – FILTRANDO HÁBITOS ATRAVÉS DO ID, NESTE CASO, COM UM ID EXISTENTE



FONTE: O autor

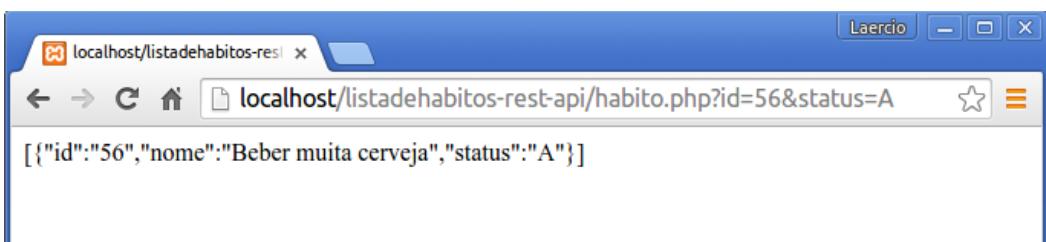
FIGURA 79 – FILTRANDO HÁBITOS ATRAVÉS DO ID, NESTE CASO, COM UM ID INEXISTENTE



FONTE: O autor

7. Você também pode informar mais de um parâmetro se os filtros conferem com a informação cadastrada no Banco de Dados, os registros serão retornados. O exemplo a seguir ilustra uma consulta à API informando o id e o *status*.

FIGURA 80 – USANDO MAIS DE UM PARÂMETRO EM REQUISIÇÕES GET



FONTE: O autor

8. Você pode continuar explorando o primeiro método da nossa API; assim que terminar, finalize o XAMPP.

Já temos uma funcionalidade do elemento hábito da API REST da aplicação Web Lista de Hábitos funcionando. Vamos prosseguir, nosso próximo passo é implementar (no arquivo *habito.php*) as demais funções. Edite seu arquivo *habito.php* do projeto *listadehabitos-rest-api* e altere seu conteúdo para que ele fique conforme o código-fonte da listagem a seguir. Transcreva atentamente cada linha de código. Leia atentamente cada linha de comentário. Perceba que, além de implementarmos as funções *f_insert_habito()*, *f_update_habito()* e *f_delete_habito()*, criamos uma nova função, a *f_obtem_conexao()*, e alteramos o código da função *f_select_habito()* implementada anteriormente.

LISTAGEM 80 - RECURSO HABITO.PHP IMPLEMENTADO

```
<?php
/*
 * Função que converte os parâmetros
 * de requisições HTTP

```

```
* POST e PUT em um Hábito
*
*/
function f_parametro_to_habito(){

// Obtém o conteúdo da requisição
$dados = file_get_contents("php://input");

// Converte para Json e retornar
$habito = json_decode($dados, true);
return $habito;
}

/*
* Função que retorna uma conexão
* com o banco de dados.
*
*/
function f_obtem_conexao(){

// Parâmetros
$servidor = "localhost";
$usuario = "root";
$senha = "";
$bancodedados = "listadehabitó";

// Cria uma conexão com o banco de dados
$conexao = new mysqli( $servidor
                      , $usuario
                      , $senha
                      , $bancodedados);

// Verifica a conexão
if ($conexao->connect_error) {
    die("Falha na conexão: " .
        $conexao->connect_error);
}
```

```

    return $conexao;
}

/*
 * Função que retorna os hábitos
 *
 */
function f_select_habito() {

    // cria uma cláusula WHERE com os
    // parâmetros que foram
    // recebidos através da requisição
    // HTTP get
    $queryWhere = " WHERE ";
    $primeiroParametro = true;
    $parametrosGet = array_keys($_GET);
    foreach ($parametrosGet as $param) {
        if (!$primeiroParametro) {
            $queryWhere .= " AND ";
        }
        $primeiroParametro = false;
        $queryWhere .= $param." = '".$_GET[$param]."'";
    }

    // Executa a query da variável $sql
    $sql = " SELECT id ".
            "       , nome ".
            "       , status ".
            "  FROM habito ";

    // utiliza o where criado com base
    // nos parâmetros do GET
    if ($queryWhere != " WHERE ") {
        $sql .= $queryWhere;
    }

    // Obtém a conexão com o DB
    $conexao = f_obtem_conexao();
}

```

```
// Executa a query
$resultado = $conexao->query($sql);

// Verifica se a query retornou registros
if ($resultado->num_rows > 0) {

    // Inicializa o array para
    // a formação dos objetos JSON
    $jsonHabitoArray = Array();
    $contador = 0;
    while($registro = $resultado->fetch_assoc()) {
        // Monta um objeto Json
        // através de um array associativo
        // ou seja, indexado através de strings
        $jsonHabito = Array();
        $jsonHabito["id"] = $registro["id"];
        $jsonHabito["nome"] = $registro["nome"];
        $jsonHabito["status"] = $registro["status"];
        $jsonHabitoArray[$contador++] = $jsonHabito;
    }
    // Transforma o array com
    // os resultados da query
    // em um array Json e imprime-o
    // na página
    echo json_encode($jsonHabitoArray);
} else {
    // Se a query não retornou
    // registros, devolve um
    // array Json vazio
    echo json_encode(Array());
}

// Fecha a conexão com o MySQL
$conexao->close();

}

/*
 * Insere um novo hábito na tabela habito

```

```

*
*/
function f_insert_habito() {

    $habito = f_parametro_to_habito();

    // Busca nome que foi recebido
    // via post através do formulário
    // de cadastro
    $nome = $habito["nome"];

    // Insere o hábito na tabela
    // habito do banco de dados
    $sql = "INSERT INTO habito (nome)
VALUES ('".$nome."')";

    // Obtem a conexão
    $conexao = f_obtem_conexao();

    // Verifica se ocorreu tudo bem
    // Caso houve erro, fecha a conexão
    // e aborta o programa
    if (!$conexao->query($sql) === TRUE) {
        $conexao->close();
        die("Erro: " . $sql . "<br>" . $conexao->error);
    }

    // Insere as demais informações
    // no Json
    $habito["id"] = mysqli_insert_id($conexao);
    $habito["status"] = "A";
    echo json_encode($habito);

    // Fecha a conexão com o
    // Banco de dados
    $conexao->close();
}

}

```

```
/*
 * Atualiza um hábito existente
 *
 */
function f_update_habito(){

    $habito = f_parametro_to_habito();

    $id = $habito["id"];
    $nome = $habito["nome"];
    $status = $habito["status"];

    // Atualiza o status de A - ativo
    // para V - vencido
    $sql = " UPDATE habito "
        . "      SET status = '". $status ."' "
        . "      , nome = '". $nome ."'"
        . " WHERE id = ".$id;

    // Obtém a conexão com o banco
    // de dados
    $conn = f_obtem_conexao();

    // Verifica se o comando foi
    // executado com sucesso
    if (!$conn->query($sql) === TRUE)) {
        $conn->close();
        die("Erro ao atualizar: " . $conn->error);
    }

    // retorna o Registro
    // atualizado
    echo json_encode($habito);

    // Fecha a conexão
    $conn->close();
}
```

```
/*
 * Exclui um hábito existente
 *
 */
function f_delete_habito() {

    // Obtém o id do registro
    // que foi recebido via get
    $id = $_GET["id"];

    $sql = "DELETE FROM habitos WHERE id="
        . $id;

    // Obtém a Conexão
    $conn = f_obtem_conexao();

    // Executa o comando delete
    // da variável $sql
    if (!$conn->query($sql) === TRUE)) {
        die("Erro ao deletar: "
            . $conn->error);
    }

    $conn->close();
}

// A variável de servidor REQUEST_METHOD
// contém o nome do método HTTP através
// qual o arquivo solicitado foi
// acessado
$metodo = $_SERVER['REQUEST_METHOD'];

// Verifica qual ação a ser tomada
// de acordo com o método HTTP
// que foi utilizado para acessar
// este recurso
switch ($metodo) {
```

```
// Se foi GET
// deve consultar
case "GET":
    f_select_habito();
    break;

// Se foi POST
// deve inserir
case "POST":
    f_insert_habito();
    break;

// Se foi put
// deve alterar
case "PUT":
    f_update_habito();
    break;

// Se foi delete
// deve excluir
case "DELETE":
    f_delete_habito();
    break;
}

?>
```

Agora, temos a API REST implementada. Os próximos assuntos abordados serão referentes a *front-end*. Mais tarde, iremos unir nosso novo *front-end* com a API que acabamos de desenvolver.

RESUMO DO TÓPICO 1

Neste tópico, você aprendeu que:

- Por muito tempo, os desenvolvedores *Web* trabalhavam desamparados de padrões de projetos, isto aumentava muito a complexidade dos códigos-fonte.
- Código-fonte bem escrito é código-fonte coeso e com baixo grau de acoplamento.
- Uma forma de tornar as aplicações *Web* coesas e diminuir o grau de acoplamento é separar as páginas *Web* do acesso aos dados. Este objetivo é alcançado construindo uma API, que separa completamente o acesso ao Banco de Dados das páginas HTML.

AUTOATIVIDADE



- 1 Utilizando seu *browser*, faça novos testes na API. Através do método *GET* faça pelo menos uma:
 - a) Requisição sem parâmetros.
 - b) Requisição o parâmetro *ID*.
 - c) Requisição com o parâmetro *Status*.
 - d) Requisição com algum parâmetro que não seja *id* nem *status*, e descreva o que acontece.

DESIGN RESPONSIVO

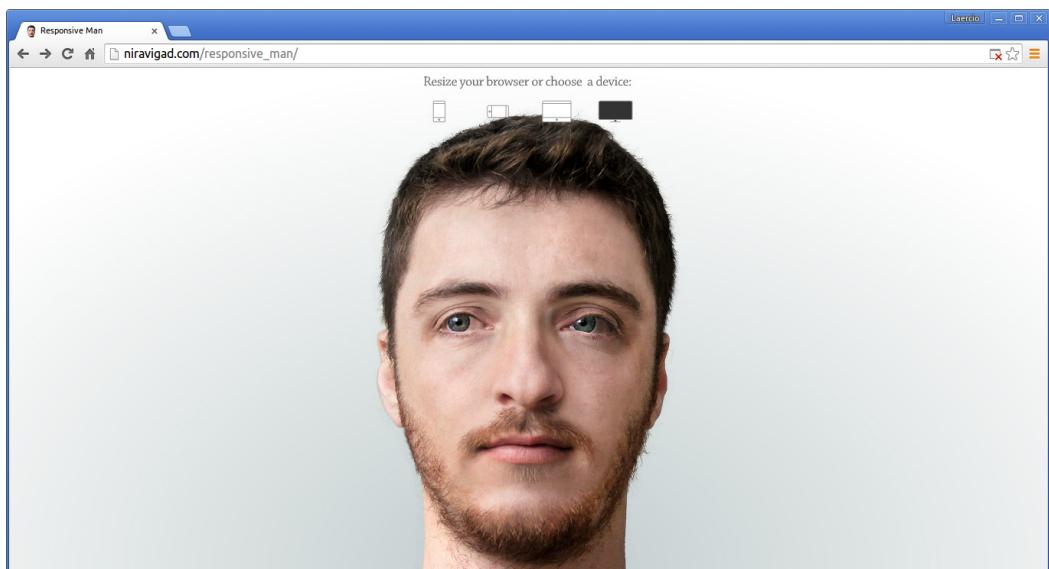
1 INTRODUÇÃO

No contexto dos sistemas Web, quando falamos em *design* responsivo, falamos em páginas Web que se adaptam ao tamanho do dispositivo que o usuário está utilizando. É muito importante que ao desenvolvermos uma página Web, levemos em consideração que o usuário pode estar utilizando um *tablet* ou *smartphone* para acessar nossa página. Por este motivo, atualmente quase não se fala em criar um *website* que não atenda a este requisito.

2 DESIGN RESPONSIVO COM BOOTSTRAP

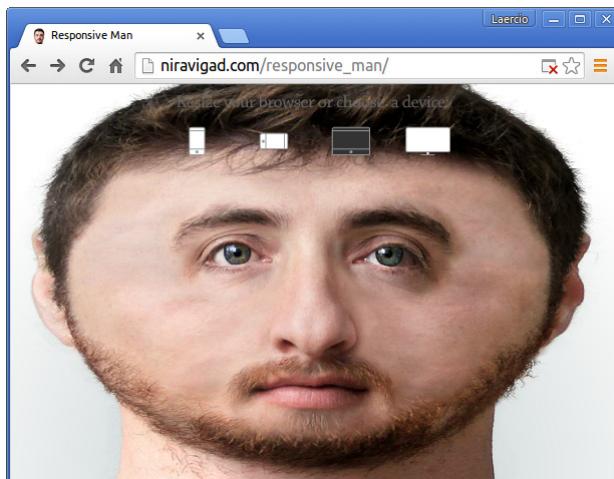
Uma excelente forma de exemplificar o que é *design* responsivo, caso você não esteja familiarizado com esta prática, é conhecendo o homem responsivo. Para que uma página Web seja considerada responsiva, ela deve adaptar seu conteúdo ao tamanho da tela. Isto pode ser percebido redimensionando o tamanho da tela do navegador, por exemplo. É exatamente isto que acontece com o rosto do homem responsivo, ele se adapta ao tamanho da janela do navegador. Acesse <http://niravigad.com/responsive_man/> e divirta-se com o homem responsivo. É necessário redimensionar a janela no *browser* para ver as alterações no rosto dele.

FIGURA 81 - O HOMEM RESPONSIVO EM UMA TELA DE LAPTOP



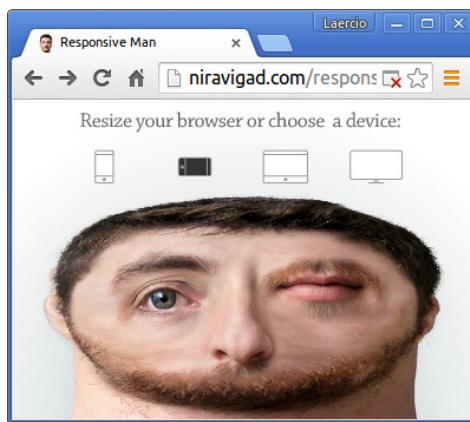
FONTE: O autor

FIGURA 82 - O HOMEM RESPONSIVO EM UMA TELA SEMELHANTE À DE TABLET



FONTE: O autor

FIGURA 83 – O HOMEM RESPONSIVO EM UMA TELA SEMELHANTE À DE SMARTPHONE EM ORIENTAÇÃO HORIZONTAL



FONTE: O autor

Conhecer o homem responsável faz com que explicações a respeito de responsividade sejam dispensadas, mas vamos pincelar este conceito mais uma vez. Quando o usuário acessa sua página Web em um dispositivo com tela grande, tal como um *laptop*, sua página deve aparecer da forma mais completa possível. Quando o usuário acessa sua página Web utilizando um dispositivo de tela média, tal como um *tablet* ou *iPad*, pode haver redimensionamento dos componentes ou até mesmo alguns componentes podem mudar de lugar. Quando o usuário acessa sua página Web, por meio de um dispositivo com tela pequena, tal como um *smartphone*, informações não essenciais podem ser ocultadas e certas áreas da página podem mudar de lugar.

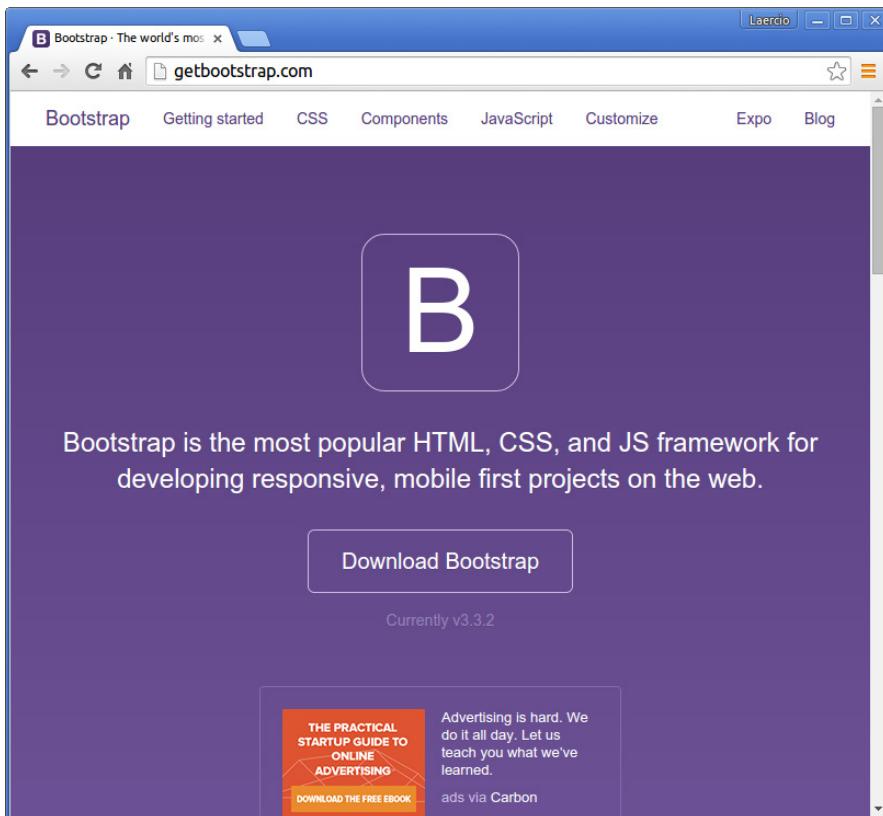
Então, como fazer com que nossas páginas Web sejam responsivas? Afinal, queremos que os usuários das nossas páginas Web tenham acesso ao nosso conteúdo independente do dispositivo que estejam utilizando. Por muito

tempo, fazer um *design* responsivo era um privilégio somente das empresas mais sofisticadas, devido às dificuldades de desenvolvimento. Até que surgiu o *framework Bootstrap*, que nasceu no Twitter. Um projeto *open source* que colaborou muito com quem efetivamente pensa em *design*. Há duas maneiras de utilizar o *Bootstrap* em seus projetos:

- Se você está sempre conectado à internet enquanto desenvolve seus projetos e/ou estuda, você pode utilizar um CDN (*Content Delivery Network*), que dispensa o *download* da biblioteca para seu projeto.
- Se, às vezes, você não possui conexão com a internet enquanto desenvolve, é necessário fazer o *download* da biblioteca.

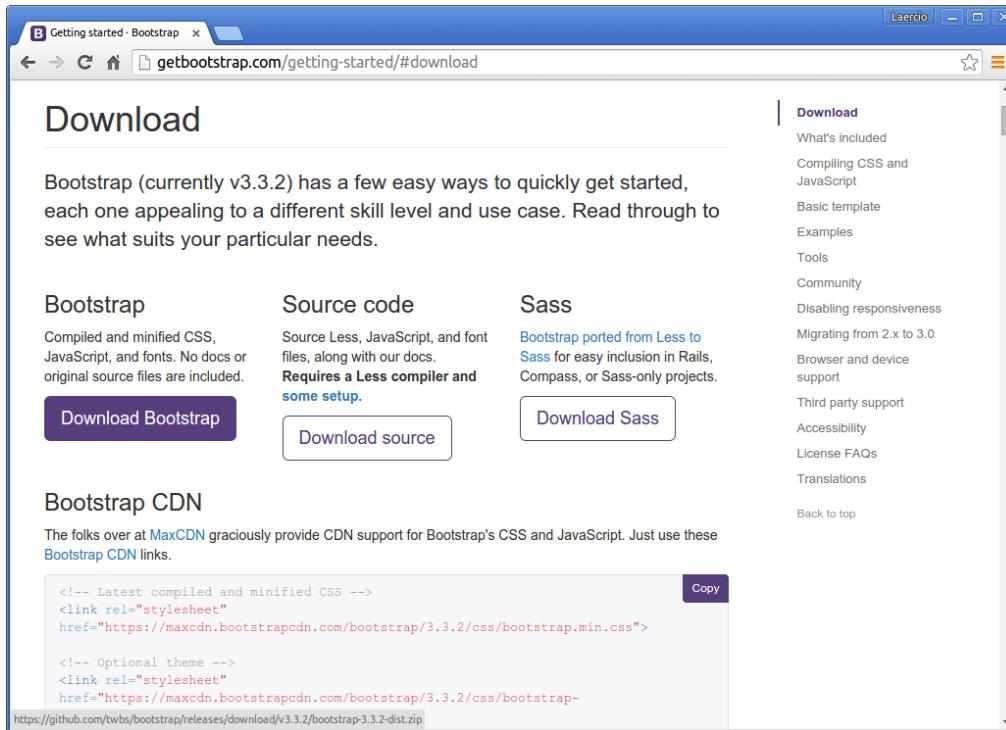
Em nossos exemplos, iremos fazer o *download* da biblioteca. Para isto, acesse <<http://getbootstrap.com/>> e clique sobre o botão *download*. Após clicar no botão de *download*, outra página será exibida para você, nesta página, clique em *Download Bootstrap*, o *download* da biblioteca irá iniciar em seguida.

FIGURA 84 – PÁGINA INICIAL PARA DOWNLOAD DO BOOTSTRAP



FONTE: O autor

FIGURA 85 – SEGUNDA PÁGINA PARA DOWNLOAD DO BOOTSTRAP



FONTE: O autor

Quando o *download* estiver concluído, o arquivo da biblioteca estará no seu diretório de *downloads*. A biblioteca é, na verdade, um diretório compactado (arquivo zip). Descompacte-o em algum local de sua preferência. Depois de descompactado, analise seu conteúdo. A biblioteca *Bootstrap* conta com diversos arquivos, mas, para finalidade de aprendizado, iremos utilizar somente os dois principais, *bootstrap.min.css* (localizado dentro do diretório css) e *bootstrap.min.js* (localizado dentro do diretório js).

Vamos começar nossos estudos implementando a versão *Bootstrap* da página *Hello World*. No seu diretório de projetos (htdocs), crie um diretório chamado *bootstrap-hello-world*. Dentro deste diretório, crie um arquivo *index.html*, um diretório chamado *css*, um diretório chamado *js* e um diretório chamado *lib*. Dentro do diretório *css*, crie um arquivo chamado *app.css*. Dentro do diretório *lib*, crie um diretório chamado *bootstrap*. Dentro do diretório *bootstrap*, cole os arquivos *bootstrap.min.css* e *bootstrap.min.js* que você encontra dentro da biblioteca do *Bootstrap*. A listagem a seguir ilustra como deve ficar sua estrutura de diretório após ter criado todos os diretórios e arquivos necessários.

LISTAGEM 81 – ESTRUTURA DE DIRETÓRIO DO PROJETO *HELLO WORLD BOOTSTRAP*

```

bootstrap-hello-world
├── css
│   └── app.css
├── index.html
└── js
    └── lib
        └── bootstrap
            ├── bootstrap.min.css
            └── bootstrap.min.js

```

Agora que você já tem os diretórios criados, acompanhe as etapas abaixo:

1. Insira o conteúdo da listagem a seguir no arquivo app.css.

LISTAGEM 82 - Conteúdo do arquivo app.css

```

body {
    text-align: center;
}

```

2. Insira o conteúdo da listagem a seguir no arquivo index.html.

LISTAGEM 83 - CONTEÚDO DO ARQUIVO INDEX.HTML DO PROJETO *HELLO WORLD BOOTSTRAP*

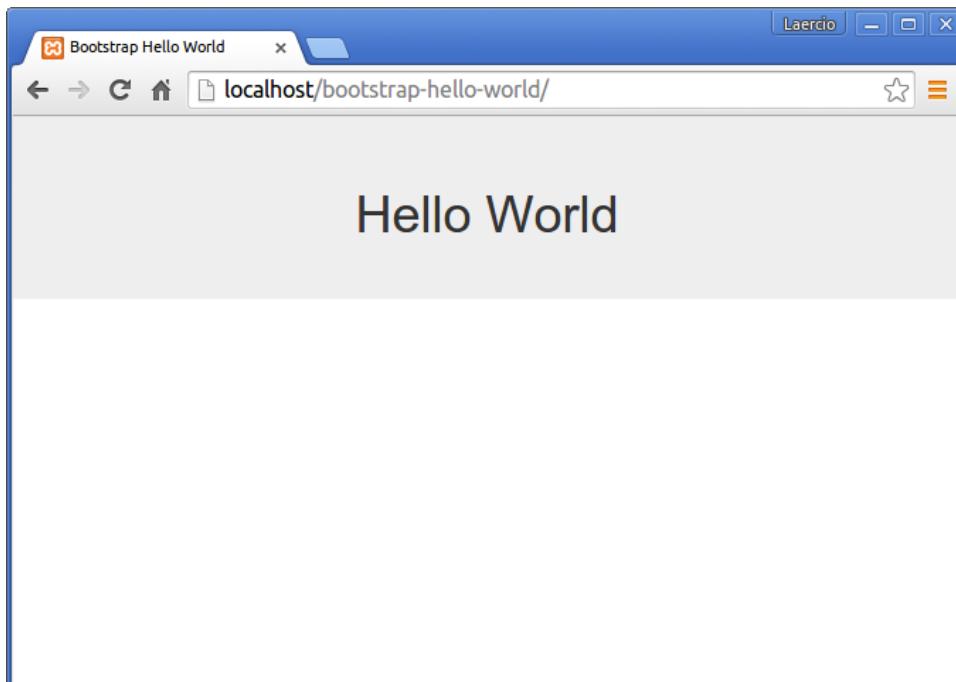
```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Bootstrap Hello World</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="css/app.css">
    <link rel="stylesheet" href="lib/bootstrap/bootstrap.min.css">
        <script src="lib/bootstrap/bootstrap.min.js"></script>
</head>
<body>
    <div class="main jumbotron">
        <h1>Hello World</h1>
    </div>
</body>
</html>

```

3. Inicie o XAMPP (caso ainda não esteja rodando) e acesse a URL do projeto que acabamos de criar, ou seja, <<http://localhost:80/bootstrap-hello-world/>>. Você deve estar visualizando uma página semelhante à da figura.

FIGURA 85 – PÁGINA WEB DO PROJETO HELLO WORLD



FONTE: O autor

Nosso *Hello World* pode até não parecer muito emocionante, mas, se você reduzir a largura da tela, vai ver que o *Bootstrap* irá redimensionar a fonte automaticamente. Vamos partir para algo mais interessante. Que tal fazer com que nossa aplicação *Web Lista de Hábitos* tenha um *design* responsivo? Há muito trabalho para ser feito! Então, vamos adotar outra estratégia. Na área de *Web design*, é muito comum que os *designers* forneçam *mockups* (ou seja, protótipos) das páginas para avaliação. Então, antes de colocar a mão na massa para definitivamente refatorar nosso app *Lista de hábitos*, iremos fazer um *mockup*. Vamos nessa!

1. Em seu diretório de projetos (*htdocs*) crie um diretório chamado *listadehabitoshack*. Dentro deste diretório, crie um diretório chamado *css*, um diretório chamado *js*, um diretório chamado *lib* e um arquivo chamado *index.html*. Dentro do diretório *lib* crie um diretório chamado *bootstrap*. Dentro do diretório *bootstrap*, cole os arquivos *bootstrap.min.css* e *bootstrap.min.js* que você encontra na biblioteca. E dentro do diretório *css* crie um arquivo chamado *app.css*. Por fim, sua estrutura de diretórios deve estar conforme a mostrada na listagem a seguir.

LISTAGEM 84 – ESTRUTURA DE DIRETÓRIO DO PROJETO LISTA DE HÁBITOS MOCKUP

```

listadehabitoshabitos-mock
├── css
│   └── app.css
├── js
└── lib
    └── bootstrap
        ├── bootstrap.min.css
        └── bootstrap.min.js
├── listadehabitoshabitos.html
└── styles.css

```

2. Insira o seguinte código-fonte no arquivo app.css.

LISTAGEM 85 – CONTEÚDO DO ARQUIVO APP.CSS DO PROJETO LISTA DE HÁBITOS MOCKUP

```

.main {
    width: 350px;
    margin-top: 30px;
    margin-left: auto;
    margin-right: auto;
    border-radius: 10px;
}

body {
    text-align: center;
    background-color: #C7E123
}

a {
    font-family: sans;
    color: #7C6B6B
}

p, td, h1, input {
    font-family: sans;
}

```

3. Insira o seguinte código-fonte HTML no arquivo index.html.

LISTAGEM 86 - CONTEÚDO DO ARQUIVO INDEX.HTML DO PROJETO LISTA DE HÁBITOS
MOCKUP

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="lib/bootstrap/
bootstrap.min.css">
    <link rel="stylesheet" type="text/css" href="css/
app.css">
    <script src="lib/bootstrap/bootstrap.min.js"></
script>
    <title>Lista de hábitos</title>
</head>
<body>
    <div class="main jumbotron">
        <h1>Lista de hábitos</h1>
        <p>Cadastre aqui os hábitos que você tem que
vencer para melhorar sua vida!</p>
    </div>
    <div class="main jumbotron">
        <table class="table table-striped">
            <tbody>
                <tr>
                    <td>Beber muita cerveja</td>
                    <td><button class="btn btn-
primary">Vencer</button></td>
                    <td><button class="btn btn-
danger">Desistir</button></td>
                </tr>
                <tr>
                    <td>Fazer pouco exercício físico</td>
                    <td><button class="btn btn-
primary">Vencer</button></td>
                    <td><button class="btn btn-
danger">Desistir</button></td>
                </tr>
            </tbody>
        </table>
    </div>
```

```

<div class="main jumbotron">
    <p>Continue mudando sua vida!</p>
    <p>Cadastre mais hábitos!</p>
    <button class="btn btn-primary">Inserir novo Hábito</button>
</div>
</body>
</html>
</html>

```

4. Inicie o XAMPP (caso ele ainda não esteja executando) e em seu *browser* acesse a URL <<http://localhost:80/listadehabitosh-mock/>>, a página exibida deve estar semelhante à da figura a seguir.

FIGURA 86 – MOCKUP PARA O NOVO LAYOUT DE NOSSA APLICAÇÃO WEB LISTA DE HÁBITOS



FONTE: O autor

5. Finalize o XAMPP, caso não queira continuar explorando os recursos do *Bootstrap*.

RESUMO DO TÓPICO 2

Neste tópico, você aprendeu que:

- Uma página *Web* deve se adaptar ao tamanho da tela do dispositivo que o usuário está utilizando para acessá-la. Isso consiste em *design* responsivo.
- Uma boa forma de entrar no mundo do *design* responsivo é utilizando a biblioteca *Bootstrap*.

AUTOATIVIDADE



- 1 Encontre, na documentação do *Bootstrap*, ou da W3C, um exemplo de utilização do navbar e implemente-o em uma página web.

SINGLE PAGE APPLICATION

1 INTRODUÇÃO

Muitas vezes, o código-fonte JavaScript pode se tornar uma colcha de retalhos. Funções gigantes, muitas variáveis globais, entre outros problemas eram muito comuns no desenvolvimento de páginas *web*, devido à falta de uma ferramenta que auxiliasse na modularização do código-fonte JavaScript.

2 TURBINANDO SUAS PÁGINAS *WEB* COM ANGULARJS

Com o intuito de auxiliar na modularização de código-fonte JavaScript, a *Google* publicou um *framework* gratuito chamado AngularJS. Com ele é possível separar completamente seu código JavaScript do código-fonte da página e modularizar o código-fonte de forma correta, aumentando muito o grau de legibilidade do código-fonte. O AngularJS é um universo, possui uma enorme quantidade de *plugins* (chamados diretivas) disponíveis para utilização gratuita. Neste Caderno de Estudos, iremos conhecer os aspectos básicos (introdutórios) do AngularJS, mas queremos encorajá-lo a não se limitar somente ao conteúdo deste caderno e explorar por sua própria conta este novo (e melhor) jeito de desenvolver páginas *web*.

Vamos começar pelo *download* da biblioteca. Acesse a URL <<https://angularjs.org/>> e clique no botão *download*. Uma próxima tela será apresentada, clique novamente em *download*.

FIGURA 87 – PÁGINA INICIAL PARA DOWNLOAD DO ANGULARJS



FONTE: O autor

FIGURA 88 – PÁGINA PARA DOWNLOAD DO ANGULARJS

Download AngularJS

This screenshot shows the 'Download AngularJS' page. It has several dropdown menus and input fields for configuration:

- Branch:** Options for '1.3.x (stable)' and '1.4.x (latest)' with a help icon.
- Build:** Options for 'Minified', 'Uncompressed', and 'Zip' with a help icon.
- CDN:** A text input field containing the URL 'https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js' with a help icon.
- Bower:** A text input field containing the command 'bower install angular#1.3.14' with a help icon.
- npm:** A text input field containing the command 'npm install angular@1.3.14'.
- Extras:** A link to 'Browse additional modules'.

At the bottom, there are two buttons: 'Previous Versions' and a large blue 'Download' button with a download icon.

FONTE: O autor

Após clicar no botão *download* da tela acima, o *download* da biblioteca irá iniciar (o nome do arquivo é angular.js ou angular.min.js). A biblioteca do AngularJS é super leve, por este motivo o *download* é bem rápido. Vamos partir agora para a implementação de nosso *Hello World*, no qual teremos uma visão geral sobre as vantagens que o AngularJS pode nos trazer.

Já sabemos que o AngularJS melhora muito a modularização de nosso código-fonte JavaScript. Para isto, ele faz uso de três conceitos que você deve entender: Diretivas, Injeção de dependência e *Dynamic binding*, conforme segue.

- Diretivas: É a ponte entre o código-fonte HTML e o código-fonte desenvolvido em JavaScript com auxílio do AngularJS. Você verá várias delas em seus códigos-fonte HTML daqui por diante, elas são sempre prefixadas pelas letras `ng-`.
- Injeção de dependência: É o conceito análogo ao `include` do PHP, o `uses` do C# e o `import` do Java. Por incrível que pareça, JavaScript não contava com um mecanismo que permitia “ligar” um código-fonte ao outro. O AngularJS veio com a solução para esta situação.
- *Dynamic Binding*: É um recurso que automatiza a atualização dos componentes da página com as alterações feitas nos dados do modelo. Você entenderá melhor depois do nosso *Hello World*.

Vamos criar nosso *Hello World*, afinal é programando que se aprende a programar. Siga as etapas.

1. No seu diretório de projetos, crie um diretório chamado `angular-js-hello-world`. Dentro do diretório `angular-js-hello-world` crie um diretório chamado `css`, outro diretório chamado `js`, outro diretório chamado `lib` e um arquivo `index.html`. Dentro do diretório `lib`, cole o arquivo da biblioteca do angular que você baixou (`angular.js` ou `angular.min.js`). Dentro do diretório `js`, crie um arquivo chamado `app.js`. Por fim, sua estrutura de diretórios deve estar conforme a seguinte.

LISTAGEM 14 – ESTRUTURA DE DIRETÓRIOS DO PROJETO HELLO WORLD ANGULARJS

```
angular-js-hello-world
├── css
├── index.html
└── js
    └── app.js
└── lib
    ├── angular.js
    └── angular.min.js
```

2. Transcreva o seguinte código-fonte para seu arquivo `app.js`

LISTAGEM 87 – CONTEÚDO DO ARQUIVO APP.JS DO PROJETO HELLO WORLD ANGULARJS

```
// Quando declaramos o módulo principal de uma
// aplicação Web, o primeiro parâmetro string deve
// ser correspondente ao valor que iremos
// utilizar na diretiva ng-app em seu código HTML
var helloWorldApp = angular.module("helloworld", []);
```

```
// Quando declaramos controllers, o primeiro
parâmetro
```

```
// string deve ser correspondente ao valor que iremos
// utilizar na diretiva ng-controller em seu
// código HTML
// Os demais parâmetros são injeção de dependência,
// neste caso, estamos injetando $scope
helloWorldApp.controller("helloworldcontroller",
["$scope", function($scope){
    $scope.msg = "Hello World!";
}]);
```

3. Transcreva o código-fonte da listagem abaixo para seu arquivo index.html

LISTAGEM 88 – CÓDIGO-FONTE DA PÁGINA WEB INDEX.HTML DO PROJETO HELLO WORLD ANGULAR

```
<!DOCTYPE html>
<html lang="pt-br" ng-app="helloworld">
<!-- Note, na abertura da tag html acima
a utilização da diretiva ng-app. --&gt;

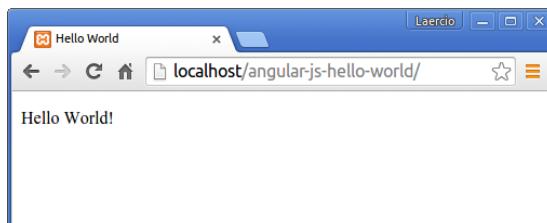
&lt;head&gt;
    &lt;!-- Fique atento a ordem de inclusão dos
        arquivos javascript abaixo --&gt;
    &lt;!-- Note também que se você estiver utilizando
        o arquivo angular.min.js, o mesmo deve ser
referenciado

    Porém se você estiver utilizando o arquivo
angular.js
        então este deve ser referenciado. --&gt;
    &lt;script type="text/javascript" src="lib/angular.
js"&gt;&lt;/script&gt;
    &lt;script type="text/javascript" src="js/app.js"&gt;&lt;/
script&gt;
    &lt;title&gt;Hello World&lt;/title&gt;
&lt;/head&gt;
<!-- Note na tag body a utilização da diretiva
ng-controller, que ligará as tags contidas
entre a abertura e fechamento de body a um
controller no código fonte javascript --&gt;
&lt;body ng-controller="helloworldcontroller"&gt;
    &lt;!-- Abaixo, note abaixo a utilização de uma
        expressão para imprimir o valor da variável
        de escopo msg --&gt;
    &lt;p&gt;{{msg}}&lt;/p&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>

```

4. Agora, inicie o XAMPP, caso ele ainda não esteja rodando, e acesse, em seu *browser*, a URL <<http://localhost:80/angular-js-hello-world/>>. Sua página deve estar conforme a da figura a seguir.

FIGURA 89 - HELLO WORLD EM ANGULARJS



FONTE: O autor

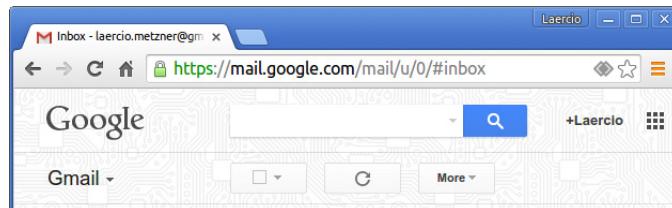
Agora, vamos entender o que fizemos, e de que forma os conceitos de Diretivas, Injeção de dependência e *Dynamic Binding* são aplicados ao nosso *Hello World*.

Em primeiro lugar, é necessário notar que há correspondência entre o valor da diretiva `ng-app` no código-fonte HTML e a declaração do módulo no código-fonte JavaScript. Na `tag html` inserimos a diretiva `ng-app="helloworld"` e no código-fonte temos a declaração do módulo `angular.module("helloworld", [])`. Isto quer dizer que tudo que se encontrar entre a abertura e o fechamento da `tag html` (ou seja `<html>` e `</html>`) poderá ser referenciado e controlado via JavaScript, através de um módulo chamado `helloworld`. Em segundo lugar, note que há correspondência entre a diretiva `ng-controller` da `tag body` e a declaração do `controller` no arquivo `app.js` (`ng-controller="helloworldcontroller"` e `helloworldApp.controller("helloworldcontroller", ...)` respectivamente). Isto quer dizer que o que estiver entre a `tag body` e seu fechamento será controlado através de um `controller` chamado `helloworldcontroller`. O terceiro ponto a ser notado é o parâmetro `$scope`. O parâmetro `$scope` constante na declaração do `controller` é uma injeção de dependência, neste caso, injetamos um objeto utilizado pelo *framework* para compartilhar valores entre o `controller` e o código-fonte HTML, através de expressões e diretivas. E em quarto lugar, mas não menos importante, note que não foi feita nenhuma referência a elementos de HTML através de `id` ou `name`, e que o valor da variável de escopo `msg` foi impresso no parágrafo sem muito esforço, isto se deve ao *dynamic binding*.

3 CONHECENDO SINGLE PAGE APPLICATION E IMPLEMENTANDO UM TESTE PARA NOSSA API

Se você é um internauta atencioso, deve ter notado que nas páginas mais legais que você navega aparece um *hashtag* no meio da URL. Caso você nunca notou, acesse agora o Gmail (e-mail da Google e confira).

FIGURA 90 – HASHTAG NA URL DO GMAIL



FONTE: O autor

Mas, o que na verdade significa este *hashtag* na barra de endereços? Significa que a aplicação *web* que você está acessando foi implementada tendo em mente o conceito de *Single Page Application*, o que certamente deixará o carregamento desta página menos penoso para quem a utiliza.

A ideia por trás do *Single Page Application* é muito simples, trata-se de evitar o completo recarregamento da página a cada iteração do usuário, carregar somente a parte que for necessária. Se você clicar sobre um *link*, no modelo “tradicional” a página inteira é recarregada, enquanto que utilizando *Single Page Application*, somente a área na qual o conteúdo desejado será mostrado sofrerá carregamento, o restante da página permanece como estava.

Para construir uma página *Web Single Page Application* com AngularJS é necessário fazer o *download* de uma biblioteca adicional chamada angular-route. Para fazer o *download*, acesse a URL <<https://github.com/angular/bower-angular-route>> e clique sobre o botão *download as a zip*.

FIGURA 91 – PÁGINA PARA FAZER O DOWNLOAD DA BIBLIOTECA ANGULAR-ROUTE NO GITHUB

FONTE: O autor

Agora que já fizemos o *download* da biblioteca, encontre o arquivo que você acabou de baixar e descompacte-o. Dentro da pasta haverá vários arquivos, localize o arquivo angular-route.js, este é o arquivo que iremos utilizar em nossos estudos.

Vamos começar nossa implementação. Vamos primeiro criar a estrutura de diretórios, depois codificar nossa aplicação para teste da API e, em seguida, vamos analisar o que foi feito. Siga as etapas.

1. No seu diretório de projetos (`htdocs`), crie um diretório chamado `listadehabitos-api-test`. Dentro do diretório `listadehabitos-api-test` crie um diretório chamado `js`, outro diretório chamado `lib` e outro diretório chamado `partials`. Ainda no diretório `listadehabitos-api-test`, crie um arquivo chamado `index.html`. Dentro do diretório `js`, crie um arquivo chamado `app.js`. Dentro do diretório `lib` cole os arquivos `angular.js` e `angular-route.js` (o primeiro é a biblioteca do AngularJS e o segundo foi baixado do `github` anteriormente). Dentro do diretório `partials`, crie cinco arquivos, um chamado `welcome.html`, outro chamado `api-test-put.html`, outro chamado `api-test-post.html`, outro chamado `api-test-get.html` e outro chamado `api-test-delete.html`. Por fim, após ter criado todos os diretórios e arquivos que compõem o nosso projeto, a estrutura de diretórios deve estar conforme apresentado na listagem a seguir:

LISTAGEM 89 – ESTRUTURA DE DIRETÓRIO PARA O PROJETO DE TESTE DA API DA APLICAÇÃO LISTA DE HÁBITOS

```
listadehabitos-api-test
├── index.html
├── js
│   └── app.js
├── lib
│   ├── angular.js
│   └── angular-route.js
└── partials
    ├── api-test-delete.html
    ├── api-test-get.html
    ├── api-test-post.html
    ├── api-test-put.html
    └── welcome.html
```

2. Transcreva o código-fonte da listagem abaixo para o arquivo `app.js`

LISTAGEM 90 – CÓDIGO-FONTE PARA O ARQUIVO APP.JS

```
// Declaramos o módulo, que é ligado à
// página Web através da diretiva ng-app
```

```
// ngRoute na linha abaixo é uma injeção de
// dependência, é um módulo que está contido
// do arquivo angular-route.js
var app = angular.module("app", [ "ngRoute" ]);

// Declaramos o controller para a página
// api-test-get.html
app.controller("apitestgetcontroller", [ "$scope",
"$http", function($scope, $http) {

    $scope.result = "";
    $scope.id = "";
    $scope.listaDeHabitosApiGet = function() {

        if ($scope.id == ""){

            $http.get("http://localhost:80/
listadehabitostest-api/habito.php").success(function(data)
{
                var output = JSON.
stringify(data);
                $scope.result = output;
            });
        }

        } else {

            $http.get("http://localhost:80/
listadehabitostest-api/habito.php", {params: {
                id: $scope.id
            }}).success(function(data) {
                var output = JSON.
stringify(data);
                $scope.result = output;
            });
        }
    }
}]);

// Declaramos o controller para a página
// api-test-post.html
app.controller("apitestpostcontroller", [ "$scope",
```

```
"$http", function($scope, $http) {  
  
    $scope.result = "";  
    $scope.nome = "";  
  
    // Criar  
    $scope.listaDeHabitosApiPost = function() {  
        $http.post("http://localhost:80/  
listadehabitostest-api/habito.php", {  
            nome: $scope.nome  
        }).success(function(data) {  
            var output = JSON.stringify(data);  
            $scope.result = output;  
        })  
    };  
  
}]);  
  
// Declaramos o controller para a página  
// api-test-put.html  
app.controller("apitestputcontroller", [ "$scope",  
"$http", function($scope, $http) {  
  
    $scope.result = "";  
    $scope.id = "";  
    $scope.nome = "";  
    $scope.status = "";  
  
    // Atualizar  
    $scope.listaDeHabitosApiPut = (function() {  
        $http.put("http://localhost:80/  
listadehabitostest-api/habito.php", {  
            id: $scope.id,  
            nome: $scope.nome,  
            status:$scope.status  
        }).success(function(data) {  
            var output = JSON.stringify(data);  
            $scope.result = output;  
        })  
    }());  
  
}]);
```

```
// Declaramos o controller para a página
// api-test-get.html
app.controller("apitestdeletecontroller", [ "$scope",
"$http", function($scope, $http) {

    $scope.result = "";
    $scope.id = "";

    // Deletar
    $scope.listaDeHabitosApiDel = (function() {
        $http.delete("http://localhost:80/
listadehabitost-rest-api/habito.php", {params: {
            id: $scope.id
        }}).success(function(data) {
            var output = JSON.stringify(data);
            $scope.result = output;
        });
    });

} ]);

// Declaramos o controller para a página
// welcome.html
app.controller("welcometcontroller", [ "$scope",
function($scope) {
}]);

// Aqui definimos a dinâmica de navegação da
// página. Para cada URL que a página possui
// define-se qual controller será usado
// e qual arquivo .html será exibido
app.config(function($routeProvider) {
    $routeProvider.when("/api-test-get", {
        controller : "apitestgetcontroller",
        templateUrl : "partials/api-test-get.html"
    }).when("/api-test-post", {
        controller : "apitestpostcontroller",
        templateUrl : "partials/api-test-post.html"
    }).when("/api-test-put", {
        controller : "apitestputcontroller",
        templateUrl : "partials/api-test-put.html"
    })
})
```

```

        }) .when("/api-test-delete", {
            controller : "apitestdeletecontroller",
            templateUrl : "partials/api-test-delete.
html"
        }) .otherwise({
            controller : "welcomecontroller",
            templateUrl : "partials/welcome.html"
        });
    });
}

```

3. Transcreva o código-fonte abaixo para o arquivo index.html

LISTAGEM 91 – CÓDIGO-FONTE PARA O ARQUIVO INDEX.HTML

```

<!DOCTYPE html>
<html ng-app='app' lang="pt-br">
<head>
<meta charset='UTF-8'>
<script src='lib/angular.js'></script>
<script src='lib/angular-route.js'></script>
<script src='js/app.js'></script>
<title>Teste api Lista de Hábitos</title>
</head>
<body>
    <div ng-view></div>
</body>
</html>

```

4. Transcreva o código-fonte a seguir para o arquivo welcome.html

LISTAGEM 92 – CÓDIGO-FONTE PARA O ARQUIVO WELCOME.HTML

```

<div>
    <h1>Lista de Hábitos API test</h1>
    <ul>
        <li><a href="#/api-test-get">Get</a></li>
        <li><a href="#/api-test-post">Post</a></li>
        <li><a href="#/api-test-put">Put</a></li>
        <li><a href="#/api-test-delete">Delete</
a></li>
    </ul>
</div>

```

5. Transcreva o código-fonte a seguir para o arquivo api-test-get.html

LISTAGEM 93 – CÓDIGO-FONTE PARA O ARQUIVO API-TEST-GET.HTML

```
<h1>Get Test</h1>
<p><label>Id: <input type="text" autofocus ng-model="id" /></label></p>
<p><button ng-click="listaDeHabitosApiGet()" class="btn btn-primary">Testar!</button></p>
<p>{{result}}</p>
<a href="#">Voltar</a>
```

6. Transcreva o código-fonte a seguir para o arquivo api-test-post.html

LISTAGEM 94 – Código-fonte para o arquivo api-test-post.html

```
<h1>Post Test</h1>
<p><label>Nome: <input type="text" autofocus ng-model="nome" /></label></p>
<p><button ng-click="listaDeHabitosApiPost()" class="btn btn-primary">Testar!</button></p>
<p>{{result}}</p>
<a href="#">Voltar</a>
```

7. Transcreva o código-fonte a seguir para o arquivo api-test-put.html

LISTAGEM 95 – CÓDIGO-FONTE PARA O ARQUIVO API-TEST-PUT.HTML

```
<h1>Put Test</h1>
<p><label>Id: <input type="text" autofocus ng-model="id" /></label></p>
<p><label>Nome: <input type="text" autofocus ng-model="nome" /></label></p>
<p><label>Status: <input type="text" autofocus ng-model="status" /></label></p>
<p><button ng-click="listaDeHabitosApiPut()" class="btn btn-primary">Testar!</button></p>
<p>{{result}}</p>
<a href="#">Voltar</a>
```

8. Transcreva o código-fonte a seguir para o arquivo api-test-delelte.html

LISTAGEM 96 – CÓDIGO-FONTE PARA O ARQUIVO API-TEST-DELELTE.HTML

```

<h1>Delete Test</h1>
<p><label>Id: <input type="text" autofocus ng-
model="id" /></label></p>
<p><button ng-click="listaDeHabitosApiDel()" 
class="btn btn-primary">Testar!</button></p>
<p>{{result}}</p>
<a href="#">Voltar</a>

```

9. Inicie o XAMPP, caso ainda não esteja iniciado, e acesse `<http://localhost:80/listadehabitos-api-test/>`. A página deve ser semelhante à mostrada na figura a seguir.

FIGURA 92 – PÁGINA INICIAL DA APLICAÇÃO WEB PARA TESTE DA API LISTADEHABITOS



FONTE: O autor

Agora, vamos analisar o que acabamos de implementar. Começando pelo arquivo `app.js`, nele temos a declaração de um módulo chamado `app`. Na declaração do módulo `app`, também estamos fazendo a injeção de dependência do módulo `ngRoute`, que é encontrado no arquivo de biblioteca que inserimos no projeto, `angular-route.js`. Em seguida, declararamos um *controller* para cada página da aplicação, são cinco no total, `apitestgetcontroller`, `apitestpostcontroller`, `apitestputcontroller`, `apitestdeletecontroller` e `welcomecontroller`. Na sequência, implementamos o esquema de navegação da nossa aplicação *web*, através do método `config` que recebe `$routeProvider` como parâmetro. A estrutura do comando, que consta no método `config` pode parecer complexa à primeira vista, mas é simples. Analise a primeira rota que está sendo configurada, quando a aplicação acessar a URL `/api-test-get` será exibida a página constante no arquivo `partials/api-test-get.html` utilizando o *controller* `apitestgetcontroller`. Já a última rota, declara que quando for acessada qualquer página que não for alguma das declaradas anteriormente, será exibida a página de `partials/welcome.html` utilizando o *controller* `welcomecontroller`.

O arquivo `index.html` é a página inicial da nossa aplicação, mas note que ele está um tanto quanto vazio. A div que consta no corpo da página (`<div ng-view></div>`) é o elemento cujo conteúdo é substituído pela página que o usuário acessou.

Durante a navegação (quando o usuário acessa um *link*, por exemplo), somente o conteúdo desta div será atualizado. Todo o restante ficará como estava no *browser*.

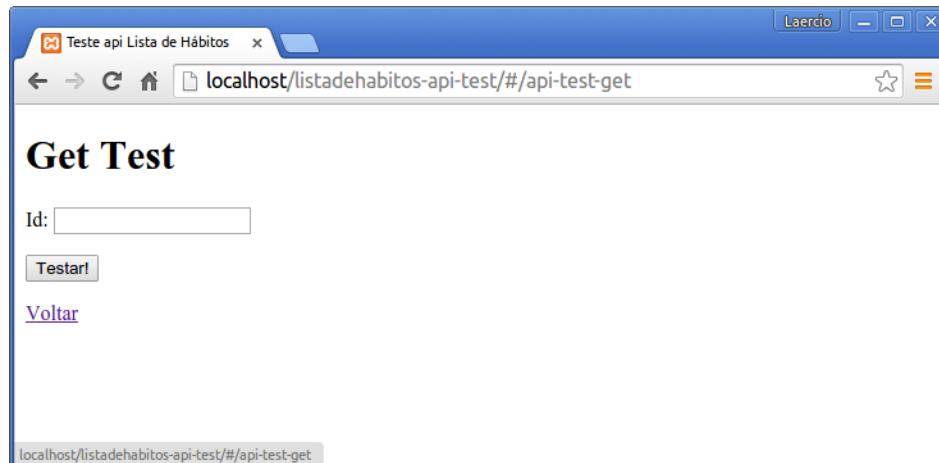
Agora, veja o arquivo welcome.html. Como você já deve ter notado, ele é o arquivo que é carregado inicialmente na página. Isto se deve à regra que definimos no método config (no arquivo app.css), que diz que quando for acessado qualquer *link* que não estiver mapeado, esta página é exibida. Note que em qualquer um dos arquivos html encontrados dentro do diretório partials não se encontram as *tags* html e/ou body, pois os arquivos de dentro do diretório partials são apenas pedaços de páginas web que serão exibidos dentro da div que consta no corpo da página index.html. Também é importante notar que todos os *links* da nossa aplicação iniciam com um *hashtag* (#), para denotar que se trata de uma navegação *Single Page Application*.

O arquivo api-test-get.html possui um campo de texto, ligado (através de *dynamic binding*, com a diretiva ng-model) à variável de escopo id. Também possui um botão, que quando acionado dispara a função listaDeHabitosApiGet() do controller apitestgetcontroller e exibe o resultado na expressão {{result}} que está ligada à variável de escopo result também através de dynamic binding. A função listaDeHabitosApiGet() executa uma requisição http get no servidor na URL da API. As páginas api-test-post.html, api-test-put.html e api-test-delete.html possuem seu funcionamento muito semelhante à api-test-get.html, e podem ser interpretadas através da observação do código-fonte.

Tanto trabalho e tanta teoria envolvida, é hora de testar nossa aplicação web. Siga as etapas abaixo:

1. Inicie o XAMPP, caso ele não esteja rodando, e acesse a URL <http://localhost:80/listadehabitos-api-test/> e na página inicial clique sobre o *link Get*. Você será redirecionado para a página de testes do método get da api listadehabitostest.

FIGURA 93 – PÁGINA PARA TESTES DE REQUISIÇÕES HTTP ATRAVÉS DO MÉTODO GET NA API LISTADEDADOS



FONTE: O autor

2. Clique no botão Testar! Irão aparecer todos os registros que você possui cadastrados na tabela *habito* no MySQL. Eles serão retornados em formato Json pela API.

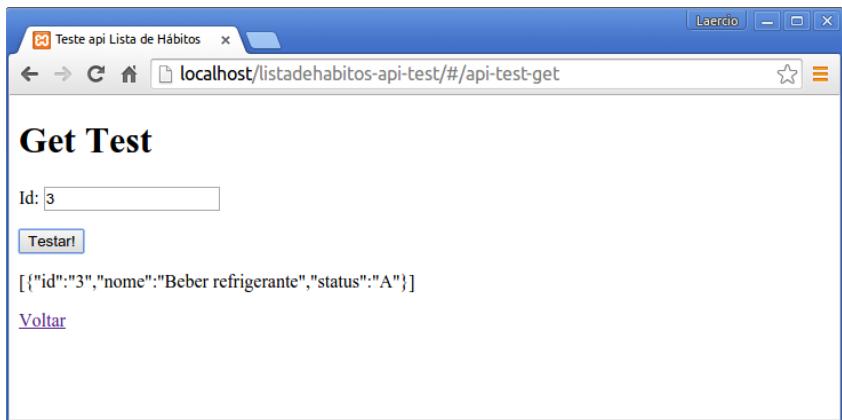
FIGURA 94 – RETORNO DA API COM TODOS OS HÁBITOS CADASTRADOS



FONTE: O autor

3. Agora informe algum ID e clique no botão Testar! Somente o objeto Json referente a este hábito será exibido.

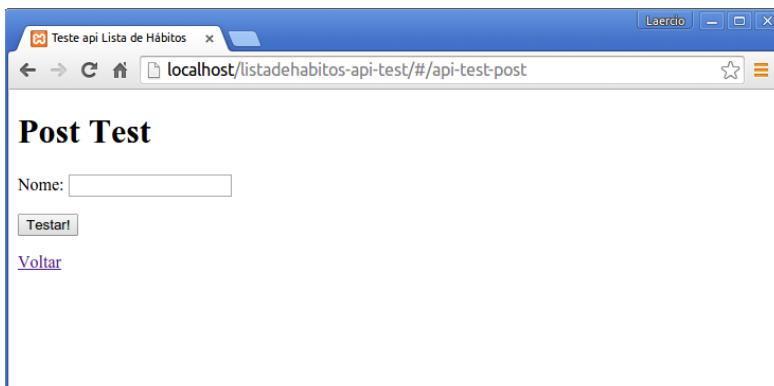
FIGURA 95 – CONSULTA NA API FILTRANDO PELO ID



FONTE: O autor

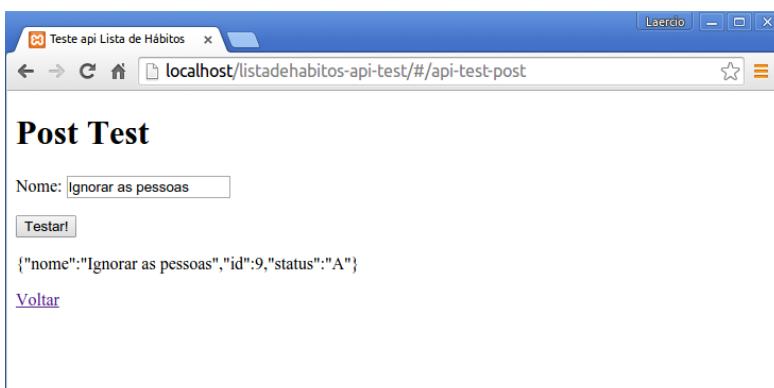
4. Agora, clique no *link* voltar e, em seguida, no *link Post*. A página para efetuarmos testes de requisições *post* será exibida. Lembre-se de que na arquitetura REST o método post indica que queremos criar um novo recurso, ou seja, no nosso caso, inserir um registro na tabela *habito*. Informe um nome para um novo hábito e clique no botão Testar!

FIGURA 96 – PÁGINA PARA TESTES DE REQUISIÇÕES HTTP POST NA API LISTADEHABITOS



FONTE: O Autor

FIGURA 97 – INSERINDO UM NOVO HÁBITO ATRAVÉS DA PÁGINA DE TESTES DA API



FONTE: O Autor

5. Para saber se o hábito foi realmente inserido, vamos efetuar uma requisição *get*. Clique sobre o *link* voltar e, em seguida, sobre o *link Get*. Na página para teste do *Get*, clique sobre o botão *Testar* e confirme a existência do hábito recém-criado.

FIGURA 98 – CONSULTANDO O REGISTRO RECÉM-CRIADO



FONTE: O Autor

6. Agora, volte à página inicial e clique sobre o *link Put*. Você deve se lembrar que na arquitetura REST o método HTTP Put é utilizado para atualizar recursos, ou seja, no nosso caso, alterar informações referentes a hábitos. Vamos fazer uma alteração em algum hábito e consultar através da tela de *Get*. Para isto, na tela de teste Put, utilize algum id existente na sua tabela hábito, e entre as novas informações no campo nome e status e na sequência clique no botão testar.

FIGURA 99 – ATUALIZANDO UM HÁBITO VIA HTTP PUT



FONTE: O Autor

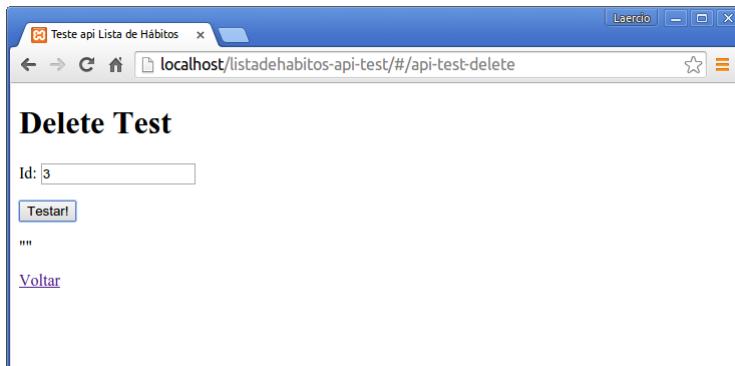
FIGURA 100 – VISUALIZANDO O REGISTRO RECÉM-ATUALIZADO



FONTE: O Autor

7. Por fim, volte à tela inicial e clique sobre o *link Delete*, vamos agora testar a exclusão. Você se lembra de que na arquitetura REST requisições HTTP *delete* representam exclusão de recursos, no nosso caso, exclusão de registros da tabela habitos. Informe um ID existente e clique sobre o botão testar! Em seguida, execute uma busca (tela de teste do *get*) para confirmar se o registro foi excluído com sucesso.

FIGURA 101 – EXCLUINDO UM HÁBITO



FONTE: O Autor

FIGURA 102 – CONFERINDO A EXCLUSÃO DO REGISTRO



FONTE: O Autor



Por fim, aprendemos o que é uma aplicação *Web Single Page Application* e implementamos um teste para nossa API. Espero que tenham gostado!

4 BOOTSTRAP, SINGLE PAGE APPLICATION, ANGULARJS E ACESSO A API NA NOVA VERSÃO DA APLICAÇÃO WEB LISTA DE HÁBITOS

Você deve lembrar-se do *mockup* que desenvolvemos para apresentar o novo *layout* da aplicação Lista de Hábitos. Então, vamos implementar este *layout* em uma nova versão da aplicação Web Lista de Hábitos, que contará com todas as mais recomendáveis práticas de programação para Web, API de acesso a dados, *Single Page Application*, e código-fonte coeso e com baixo grau de acoplamento. Siga as etapas abaixo. Dedique atenção especial a cada linha de código e de comentário deste projeto.

1. Em seu diretório de projetos (htdocs), crie um diretório chamado listadehabitos-spa. Dentro do diretório listadehabitos-spa crie um diretório chamado css, outro chamado js, outro chamado lib, outro chamado partials e um arquivo chamado index.html. Dentro do diretório css, crie um arquivo chamado app.css. Dentro do diretório js, crie um arquivo chamado app.js. Dentro do diretório lib crie um diretório chamado angularjs e outro chamado bootstrap. Dentro do diretório angularjs cole os arquivos angular.js (ou angular.min.js) e angular-route.js. Dentro do diretório bootstrap cole os arquivos bootstrap.min.css e bootstrap.min.js. Dentro do diretório partials crie um arquivo chamado listadehabitos.html e outro chamado novohabito.html. Por fim, a estrutura de diretórios do projeto deve estar conforme a listagem a seguir.

**LISTAGEM 97 – ESTRUTURA DE DIRETÓRIOS DA NOVA VERSÃO DA APLICAÇÃO
WEB LISTA DE HÁBITOS**

```

listadehabitos-spa
├── css
│   └── app.css
├── index.html
└── js
    └── app.js
└── lib
    ├── angularjs
    │   ├── angular.min.js
    │   └── angular-route.js
    └── bootstrap
        ├── bootstrap.min.css
        └── bootstrap.min.js
└── partials
    ├── listadehabitos.html
    └── novohabito.html

```

2. Transcreva o código-fonte a seguir para seu arquivo index.html

LISTAGEM 98 – CÓDIGO-FONTE PARA O ARQUIVO INDEX.HTML

```

<!DOCTYPE html>
<!-- Ligamos o código HTML ao código JavaScript
através da diretiva ng-app, que representa
um módulo do angularJS -->
<html lang="pt-br" ng-app="app">
<head>
    <meta charset="utf-8" />
    <!-- Adicionamos os arquivos das bibliotecas
    e da aplicação -->

```

```

<link rel="stylesheet" type="text/css" href="lib/
bootstrap/bootstrap.min.css" />
<link rel="stylesheet" type="text/css" href="css/app.
css">
<script src="lib/bootstrap/bootstrap.min.js"></script>
<script src="lib/angularjs/angular.min.js"></script>
<script src="lib/angularjs/angular-route.js"></script>
<script src="js/app.js"></script>
<title>Lista de Hábitos</title>
</head>
<body>
<!-- Esta div terá seu conteúdo
      recarregado durante as
      interações do usuário -->
<div ng-view></div>
</body>
</html>

```

3. Transcreva o código-fonte a seguir para seu arquivo app.css

LISTAGEM 99 – CÓDIGO-FONTE PARA O ARQUIVO APP.CSS

```

.main {
    width: 350px;
    margin-top: 30px;
    margin-left: auto;
    margin-right: auto;
    border-radius: 10px;
    text-align: center;
}

body {
    background-color: #C7E123
}

th {
    text-align: center;
}

a {
    color: #7C6B6B
}

```

4. Transcreva o código-fonte a seguir para o arquivo app.js

LISTAGEM 100 – CÓDIGO-FONTE PARA O ARQUIVO APP.JS

```
// Aqui declaramos o módulo, que é ligado
// ao HTML através da diretiva ng-app
var app = angular.module("app", ["ngRoute"]);

// Declaramos um value dentro do módulo
// Em AngularJs values são valores que podem
// ser utilizados em diversas páginas.
app.value("habitos", {
    habitos: []
});

// Declaramos o controller para a página
// na qual o usuário acessa sua lista de hábitos
// Em AngularJS, os controllers são ligados ao código
// HTML através da diretiva ng-controller
// Mas no nosso caso, estamos utilizando a
// biblioteca angular-route na qual o controller
// é associado ao HTML no método config
// quando configuramos as rotas de navegação
app.controller("listadehabitost", ["$scope", "$http",
"habitost", function($scope, $http, habitost) {

    $scope.habitost = habitost.habitost;
    // Caso o array esteja vazio, busca
    // os valores da API
    if (habitost.habitost.length == 0) {

        $http.get('http://localhost:80/listadehabitost-
rest-api/habito.php').success(function(data) {
            for (indice in data){
                habitost.habitost[indice] =
data[indice];
            }
        });
    }
    $scope.mostraLista = $scope.habitost.length == 0;
```

```
// Atualiza o status de um hábito para V
$scope.vencerHabito = function(habito) {
    indice = $scope.habitos.indexOf(habito);
    habito.status = "V";
    $http.put('http://localhost:80/
listadehabitost-rest-api/habito.php', habito).
success(function(data) {
    $scope.habitos[indice] = data;
}) ;
}

// Atualiza o status de um hábito para A
$scope.retomarHabito = function(habito) {
    indice = $scope.habitos.indexOf(habito);
    habito.status = "A";
    $http.put('http://localhost:80/
listadehabitost-rest-api/habito.php', habito).
success(function(data) {
    $scope.habitos[indice] = data;
}) ;
}

// Exclui um hábito
$scope.desistirHabito = function(habito) {
    $http.delete('http://localhost:80/
listadehabitost-rest-api/habito.php', {params: {
        id: habito.id
}}).success(function(data) {
    indice = $scope.habitos.
indexOf(habito);
    $scope.habitos.splice(indice, 1);
}) ;
}

}]);

// Controller para a página na qual o usuário
// efetua a inclusão de hábitos
app.controller("novohabito", ["$scope", "$http",
```

```

"habitost", function($scope, $http, habitos) {
    $scope.habitost = habitost.habitost;

    $scope.nome = "";

    // Insere um novo hábito
    $scope.inserirHabito = function(nome) {
        if (nome == "") {
            return;
        }
        $http.post('http://localhost:80/
listadehabitost-rest-api/habito.php', {
            nome: nome
        }).success(function(data) {
            $scope.habitost[$scope.habitost.length]
= data
            $scope.nome = "";
        });
    }

} ]);

// Configura as rotas de navegação
// da aplicação Web
app.config(["$routeProvider", function($routeProvider)
{
    $routeProvider.when("/listadehabitost", {
        controller: "listadehabitost",
        templateUrl: "partials/listadehabitost.html"
    }).when("/novohabito", {
        controller:"novohabito",
        templateUrl:"partials/novohabito.html"
    }).otherwise({
        controller: "listadehabitost",
        templateUrl: "partials/listadehabitost.html"
    });
} ]);

```

5. Transcreva o código-fonte a seguir para o arquivo listadehabitost.html

LISTAGEM 101 – CÓDIGO-FONTE PARA O ARQUIVO LISTADEHABITOS.HTML

```
<div class="main jumbotron">
    <h1>Lista de Hábitos</h1>
    <p>Cadastre aqui os hábitos que você tem que
vencer para melhorar sua vida!</p>
</div>
<div class="main jumbotron">
    <table class="table table-striped">
        <tbody>
            <thead>
                <th>Hábitos</th>
                <th colspan="2">Ações</th>
            </thead>
            <!-- A diretiva ng-repeat implementa um
looping de repetição
através do qual buscamos todos os
elementos do
array $scope.habitos e exibimo-os na
tabela HTML
abaixo, ordenando-os pelo seu status
-->
            <tr ng-repeat="habito in habitos |
orderBy:'status'">
                <td>{{habito.nome}}</td>
                <!-- Quando o usuário clica no botão
vencer,
é acionada a função $scope.
vencerHabito-->
                <td><button class="btn btn-primary" ng-
click="vencerHabito(habito)" ng-hide="habito.status !=
'A'">Vencer</button></td>
                <td>
                    <!-- Quando o usuário clica no botão
Desistir,
é acionada a função $scope.
desistirHabito-->
```

```

        <button class="btn btn-danger" ng-
click="desistirHabito(habito)" ng-hide="habito.status !=
'A'">Desistir</button>
        <!-- Quando o usuário clica no botão
Retomar,
        é acionada a função $scope.
retomarHabito-->
        <button class="btn btn-danger" ng-
click="retomarHabito(habito)" ng-hide="habito.status !=
'V'">Retomar</button>
        </td>
    </tr>
</tbody>
</table>
</div>
<div class="main jumbotron">
    <p>Mude sua vida!</p>
    <p>Cadastre seus maus hábitos!</p>
    <!-- Note o link Single page Application!! # -->
    <a class="btn btn-primary" href="#/
novohabito">Inserir novos hábitos</a>
</div>

```

6. Transcreva o código-fonte a seguir para o arquivo novohabito.html

LISTAGEM 102 – CÓDIGO-FONTE PARA O ARQUIVO NOVOHABITO.HTML

```

<div class="main jumbotron">
<h1>Cadastrar hábitos</h1>
    <p><label>Hábito: <input type="text" id="nome"
name="nome" autofocus ng-model="nome" /></label></p>
        <!-- Ao clicar no botão Cadastrar
            a função $scope.inserirHabito
            é chamada-->
    <p><button ng-click="inserirHabito(nome)"
class="btn btn-primary">Cadastrar</button></p>
        <!-- Note o link single page application!! # -->
        <a href="#/listadehabitosh">Voltar</a>
</div>
<div class="main jumbotron">

```

```
<table class="table table-striped">
  <thead>
    <th>Hábitos</th>
  </thead>
  <tbody>
    <tr ng-repeat="habito in habitos">
      <td>{ {habito.nome} }</td>
    </tr>
  </tbody>
</table>
</div>
```

7. Agora inicie o XAMPP, caso ainda não esteja iniciado, e acesse a URL <<http://localhost:80/listadehabitospa/>> e veja. Sua aplicação *web* deve estar conforme a da figura a seguir.

FIGURA 103 – PÁGINA INICIAL DA APLICAÇÃO WEB LISTA DE HÁBITOS

The screenshot shows the initial page of a web application titled "Lista de Hábitos". The main title is displayed prominently at the top. Below it is a subtitle encouraging users to list habits they want to overcome. The central feature is a table listing four habits with corresponding actions: "Vencer" (blue button) and "Desistir" (red button). The habits listed are "Dormir pouco", "Trabalhar até muito tarde", "Comer chocolate", and "Comer muito". At the bottom of the page, there is a call-to-action button labeled "Inserir novos hábitos".

Hábitos	Ações
Dormir pouco	Vencer Desistir
Trabalhar até muito tarde	Vencer Desistir
Comer chocolate	Vencer Desistir
Comer muito	Retomar

Mude sua vida!
Cadastre seus maus hábitos!

Inserir novos hábitos

FONTE: O autor

FIGURA 104 – PÁGINA PARA CADASTRO DE NOVOS HÁBITOS

The image shows a user interface for managing habits. At the top, there is a large button labeled "Cadastrar hábitos". Below it, a form field is labeled "Hábito:" followed by an empty input box. Underneath the input box is a blue button labeled "Cadastrar". To the right of the input box is a link labeled "Voltar". In the bottom section, there is a table titled "Hábitos" with five rows of data:

Hábitos
Comer muito
Dormir pouco
Trabalhar até muito tarde
Comer chocolate

FONTE: O autor

LEITURA COMPLEMENTAR

SUPER POTÊNCIA: QUAL É O FUTURO DA INTERNET?

Vinte anos depois de sua criação, a rede mundial de computadores world wide web, criada por um cientista como uma forma simples de dividir informações com colegas, já percorreu um longo caminho. Foi um milagre acidental que cresceu sem muita orientação de comitês, governos ou corporações. Mas agora, a rede está à beira de outra transformação. Rory Cellan-Jones, jornalista especializado em tecnologia da BBC, conversou com cientistas que estão tentando prever e até guiar o futuro da web.

Veja a seguir algumas dessas previsões:

Quer chamemos de rede semântica ou rede de dados interligados, os cientistas acreditam que agora estão construindo uma rede muito mais inteligente. Ao colocar mais dados *on-line* e depois ensinando a rede a entendê-los e questioná-los de novas maneiras, eles esperam oferecer aos usuários um recurso muito mais inteligente.

Pense na rede como um grande banco de dados descentralizado contendo de tudo, desde o horário de trens e lugares para comer a *sites* que informam onde encontrar a melhor oferta. O que a rede vai oferecer é um sistema de buscas muito mais refinado, será uma rede com ‘grãos mais finos’, disse o pesquisador Nigel Shadbolt, da Universidade de Southampton.

A ideia é que quando alguém fizer uma busca, como próximo trem para Manchester, em vez de aparecerem várias páginas com informações referentes a trens para Manchester, a rede traga ao usuário uma resposta real. Mas, claro, só se os dados estiverem disponíveis na *web*. Shadbolt faz parte de uma campanha liderada pelo criador da rede, Tim Berners-Lee, para convencer o público e órgãos privados a disponibilizar a maior quantidade possível de dados *on-line*.

A rede ‘onipresente’

Nós pensamos na rede como algo que acessamos por meio de um *browser*, usando um teclado. Mas, de acordo com outra cientista da Universidade de Southampton, Wendy Hall, isto está prestes a mudar: Vamos conseguir acessar a internet onde quer que estejamos, fazendo o que for, quase sem precisar de nenhum aparelho. Poderemosvê-la por nossos óculos, ou por meio de algum visor que passaríamos a usar, por exemplo.

A cientista acredita que o *browser* vai desaparecer e que vamos interagir com a rede por meio de aplicativos, como muita gente já faz com os chamados *smartphones*.

Objetos nas nuvens

E não apenas as pessoas estarão *on-line*. Mais e mais objetos - como carros, monitores cardíacos e sensores em nossas casas - estarão conectados à internet, contribuindo para um crescente fluxo de dados.

Onde serão armazenados todos esses dados? Na nuvem, claro, ou, em outras palavras, nos enormes bancos de dados sendo construídos pelos superpoderes da *web*, como a *Google* e a *Microsoft*. Em certo sentido, a rede está se tornando um grande computador, disse Andrew Herbert, à frente do laboratório da *Microsoft* em Cambridge.

A rede de celulares

Está claro que o futuro da rede está nos celulares – e para a maioria dos bilhões de pessoas que se juntarem a ela nos próximos anos sua primeira experiência de acesso à *web* será por telefone celular. Um dos grandes pensadores da indústria de celulares, Benoit Schillings, da empresa Myriad *Software*, afirma que isso vai nos tornar ainda mais dependentes da rede.

Nós partimos do princípio de que é algo que temos conosco o tempo todo. Então, quando você perde o seu telefone, se torna um desastre – é agora uma parte essencial de como seres humanos funcionam.

Mas Schillings afirma que as limitações de uma rede de celulares, em comparação com os dados sendo baixados por uma linha fixa, significa que pesquisas em áreas como compressão de informação se tornam ainda mais vitais.

Uma rede sustentável?

Então, como podemos garantir que esta rede inteligente, móvel e penetrante possa continuar crescendo sem engolir o planeta? O pesquisador Andy Hopper, da Universidade de Cambridge, lidera um programa chamado Computação pelo Futuro do Planeta. Ele está otimista com o que a rede pode fazer. “É um marca-passo para o planeta, uma parte indispensável para a nossa civilização”, diz ele. Mas ele agora está procurando meios para que tecnologias de computação possam ser usadas para controlar ou reduzir suas pegadas de carbono.

Um de seus alunos, por exemplo, está tentando criar um monitor pessoal de energia que use a nova rede de objetos para juntar todos os tipos de informação de sensores *on-line* que monitoram o uso de energia.

Mas, quanto mais a rede crescer, maiores serão as ameaças a sua estabilidade, ou não? A piada corrente entre a comunidade de engenheiros é que a internet está sempre à beira do colapso, afirma Craig Labowitz da Arbor Networks, que monitora o desempenho da rede.

Ele é otimista e acredita que a rede vai continuar se consertando, mas afirma que, cada vez mais, isso vai depender das grandes corporações que agora controlam o tráfego.

Nos últimos três anos, afirma Labowitz, a participação da *Google* no tráfego global da internet aumentou de 1% para 10%.

Quem controla?

O que nos traz à questão crucial: quem controla o futuro da rede? Até agora ela vem crescendo de acordo com os princípios de abertura e parâmetros acertados mutuamente – mas alguns temem o surgimento de uma rede corporativa onde a inovação e a liberdade de expressão serão prejudicadas.

Não há garantias de que ela continuará evoluindo da maneira como é hoje – aberta, gratuita e com parâmetros universais, afirma Wendy Hall. Se você perder isso, ou se os parâmetros forem superados por preocupações corporativas, então a rede vai mudar dramaticamente, conclui.

FONTE: Disponível em: <http://www.bbc.co.uk/portuguese/ciencia/2010/03/100315_futurodaweb_ba.shtml>. Acesso em: 4 mar. 2015.

RESUMO DO TÓPICO 3

Neste tópico, você aprendeu que:

- *Single Page Application* é uma técnica que possibilita diminuir o tempo gasto com carregamento de páginas e melhora muito a navegação para o usuário.
- AngularJS é um *framework* poderosíssimo, que o ajuda a construir páginas com código-fonte modularizado, entre outras inúmeras vantagens.

AUTOATIVIDADE



- 1 Desenvolva uma aplicação *web* utilizando AngularJS e *Bootstrap* que permita ao usuário cadastrar uma lista de compras (somente em memória, não necessita de Banco de Dados!).

REFERÊNCIAS

BRANAS, Rodrigo. **AngularJS Essentials**. First published, Birmingham, Packt Publishing Ltda., 2014.

FLANAGAN, David. **JavaScript**, o guia definitivo. 4. ed., São Paulo: Artmed, 2002.

MILANI, André. **Construindo aplicações Web com PHP e MySQL**. São Paulo: Novatec, 2010.

SILVA, Maurício Sami. **HTML5**: linguagem de marcação que revolucionou a Web. São Paulo: Novatec, 2011.

SOARES, Wallace. **PHP5**: Conceitos, Programação e Integração com Banco de Dados. 6^a ed. rev., atual. São Paulo: Érica, 2010.