Linguagem e Programação para Web

Ewerton Mendonça

Curso Técnico em Informática

Educação a Distância

2016

EXPEDIENTE

Professor Autor
Ewerton Mendonça

***Design* Instrucional**
Deyvid Souza Nascimento
Maria de Fátima Duarte Angeiras
Renata Marques de Otero
Terezinha Mônica Sinício Beltrão

Revisão de Língua Portuguesa
Letícia Garcia

Diagramação
Izabela Cavalcanti

Coordenação
Anderson Elias

Coordenação Executiva
George Bento Catunda

Coordenação Geral
Paulo Fernando de Vasconcelos Dutra

Conteúdo produzido para os Cursos Técnicos da Secretaria Executiva de Educação Profissional de Pernambuco, em convênio com o Ministério da Educação (Rede e-Tec Brasil).

Maio, 2016

M539l

Mendonça, Ewerton.

Linguagem e Programação para Web: Curso Técnico em Informática: Educação a distância / Ewerton Mendonça. – Recife: Secretaria Executiva de Educação Profissional de Pernambuco, 2016.

89 p.: il.

Inclui referências bibliográficas.

1. Educação a distância. 2. Programação computacional. 3. Linguagem de programação. I. Mendonça, Ewerton. II. Título. III. Rede e-Tec Brasil.

CDU – 004.43

Sumário

Introdução

Neste caderno vamos aprender uma linguagem de programação chamada PHP. Esta foi a linguagem escolhida para a disciplina de Desenvolvimento para Web por ser a base para vários sistemas profissionais e muito popular entre os desenvolvedores, possuindo vários tutorias que podem ser encontrados na internet com uma simples busca.

Chamamos o PHP de uma linguagem porque ela é uma forma de comunicação com o computador. Ela possui regras de gramática e ortografia muito rígidas, ou seja, o computador só entende da maneira correta. Um errinho de grafia ou uma troca de lugar e o interpretador não vai entender o que você quer que ele faça. Por isso, tenha muita atenção e cuidado ao escrever os códigos. Se algo der errado, verifique letra a letra, palavra a palavra e linha a linha, para ver se você não escreveu algo errado. A maior parte dos erros no começo do aprendizado acontece devido a problemas de digitação.

Programação é uma arte na resolução de problemas e, muitas vezes, vamos ter que usar a criatividade para resolver algo com os comandos que temos, porque nem tudo pode ser copiado de algum lugar. Quando terminar os exemplos e atividades disponibilizados, procure outros exemplos na internet e se dedique para entender como o problema é resolvido. O ser humano aprende por repetição. A cada exemplo visto e atividade realizada você entende melhor um conceito. Repita os exemplos e atividades até se sentir confortável com a ideia apresentada. Assim, logo você estará criando suas próprias soluções.

Vamos começar com um pouquinho de história sobre o PHP e criando um ambiente de desenvolvimento para podermos trabalhar confortavelmente e testar as aplicações em nosso próprio computador, sem a necessidade de hospedar em um servidor externo.

Então? Vamos começar a nos divertir?

1.Competência 01 | Introdução à Programação para Web e Construção do Ambiente de Desenvolvimento

Antes de contarmos um pouquinho a história do PHP, vamos falar um pouco sobre linguagem de programação, os diferentes tipos de linguagens e onde o PHP está nesta classificação.

Depois vamos construir um ambiente de desenvolvimento para web passo a passo. Precisaremos baixar e instalar alguns programas gratuitos, faremos um teste construindo nossa primeira aplicação, o famoso “Hello world!” do mundo da programação.

Por fim, veremos como ficará a aplicação que iremos desenvolver nas próximas competências.

E então, prontos para começar?

1.1 Programação para Web

O computador ainda não entende exatamente o que o ser humano fala. O que o computador entende são zeros e uns. Graças à velocidade que possui, ele pode ler uma gigantesca quantidade de zeros e uns e fazer coisas incríveis, mas, para os seres humanos é impossível instruir o computador desta forma. Por isso, foram criados códigos que dão ordens básicas. A organização dessas ordens é uma gramática e seu conjunto de palavras chaves é sua ortografia, e assim temos uma linguagem. Uma forma de comunicação com um computador que diz a ele o que é para ele fazer e quando.

Dessa forma podemos escrever mais facilmente, utilizando palavras ao invés de zeros e uns, e pedir para uma aplicação traduzir esta linguagem para que o computador entenda, ou seja, dizemos que uma linguagem mais próxima dos seres humanos é de alto nível, já a linguagem de zeros e uns é de baixo nível.

Essa tradução pode possuir duas abordagens: compilativa e interpretativa.

Na compilação a tradução é feita de uma vez. Todo o programa é traduzido para a linguagem de baixo nível e pode ser executado indefinidamente sem precisar ser compilado novamente. É como se um tradutor profissional de inglês pegasse um livro e traduzisse ele para o português. O livro traduzido pode ser impresso várias vezes sem precisar ser traduzido novamente. Como exemplos de linguagem compilativas temos Pascal e C.

Na interpretação a tradução é realizada enquanto as linhas são lidas. Toda vez que a linha é lida, ela é traduzida e executada. É como se um tradutor estivesse em uma palestra e enquanto o palestrante fala ele vai traduzindo para o público. Se o palestrante repetir alguma coisa que já tenha dito, o tradutor terá que traduzir novamente. Então, se uma linha for lida mil vezes em um código, será interpretada e executada mil vezes. Como exemplo de linguagens interpretativas temos o PHP, Python e JavaScript.

Não existe uma forma melhor. Dependendo da finalidade do seu programa, você pode precisar das vantagens da interpretação ou da compilação. Atualmente, as novas tecnologias em linguagem de programação utilizam as duas técnicas, primeiro o programa é compilado em uma linguagem intermediária, depois essa linguagem é interpretada para a linguagem de baixo nível. Como exemplo desta abordagem temos a linguagem Java.

A forma como a internet é construída privilegia as linguagens interpretativas. Observe a Figura 1. No passo 1, o usuário solicita uma página para um servidor web. O servidor percebe que a página solicitada é uma aplicação em PHP e solicita sua interpretação no passo 2. Como resultado, o interpretador pode criar uma página web com o resultado da computação e envia como resposta para o usuário no passo 3.

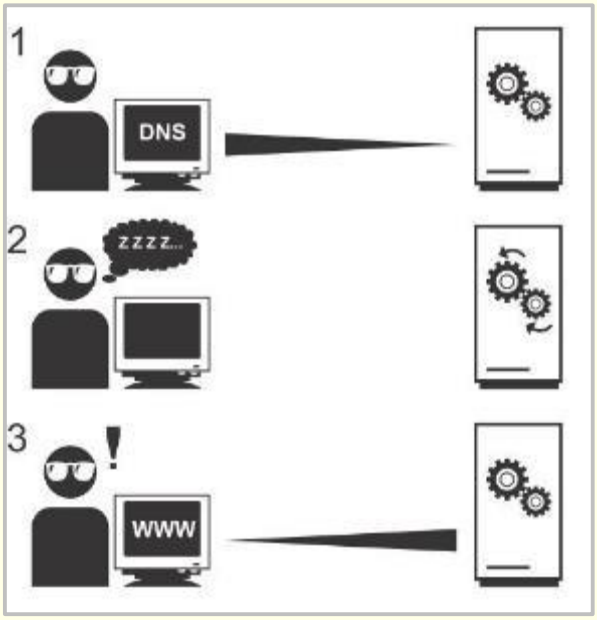


Figura 1 - Passo a passo da solicitação de recursos na internet.
Fonte: próprio autor (2016).
Descrição: A ilustração mostra os passos simplificados da solicitação de uma página web para uma aplicação. Passo 1, o usuário envia solicita um DNS para o servidor. Passo 2, o servidor processa a solicitação. Passo 3, o servidor envia para o usuário o recursos.

Não importa quantas vezes o usuário faça isso, ou quantos usuários diferentes solicitem a mesma página, toda vez o aplicativo (programa) vai ter que interpretar e executar.

A velocidade de tradução depende de vários fatores, mas geralmente a interpretação é mais lenta que a compilação.

Para saber mais sobre os tipos de tradução de linguagens de programação, acesse as páginas abaixo:
http://pt.wikipedia.org/wiki/Compilador
http://pt.wikipedia.org/wiki/Interpretador

É dessa forma que vamos pensar de agora em diante. O usuário solicita algo ao programa, ele faz a computação e devolve uma página como resposta. O computador onde o usuário vê as páginas é chamado lado do cliente e onde fica o computador servidor é chamado de lado do servidor. Por isso, chamamos de arquitetura cliente/servidor.

1.2 PHP: Histórico e Definição

PHP é uma linguagem de script executada do lado do servidor e é uma poderosa ferramenta para a criação de páginas Web dinâmicas e interativas. PHP significa "**P**HP: **H**ypertext **P**reprocessor". Ele é aberto para quem se interessar em melhorá-lo e gratuito para uso.

A versão atual do PHP é a 7, que possui diversas mudanças para se adaptar melhor às tecnologias atuais em linguagem de programação. Muitas destas mudanças são voltadas para o paradigma Orientado a Objetos, que é uma outra forma de se modelar um software. No entanto, vamos utilizar a versão 5, que podemos baixar no site **www.php.net.** Lá, você também pode encontrar novidades sobre o desenvolvimento, documentação, links para a comunidade de desenvolvedores e o código-fonte do PHP. Escolhemos a versão mais antiga por ela ser mais fácil e ter menos detalhes em relação à nova. Uma vez aprendida, ampliar os conhecimentos para a versão 7 será bem mais fácil.

Por ser uma linguagem de programação que é executada do lado do servidor, ela deve ser instalada junto a um servidor de páginas web. Se você for contratar um serviço de hospedagem de websites deve verificar se no servidor está instalado o PHP e qual a versão.

Agora vamos pegar mais intimidade com o PHP conhecendo um pouco de sua história de vida.

No início da década de 90, as páginas web pouco faziam. Naquela época as pessoas escreviam uma página e depositavam em um servidor web para serem acessadas. Apenas isso. Se o autor quisesse que ela mudasse algo, ele teria que editar a página e reenviar. Não existiam aplicativos web, e-commerce, nem blogs, como conhecemos hoje.

O link abaixo exibe um vídeo sobre a evolução da web e seus objetivos.
www.youtube.com/watch?v=V5kabPVohGY

Então, em 1994, um programador chamado Rasmus Lerdorf, criou uma série de programas do tipo CGI que faziam computação em páginas web. Esse conjunto de ferramentas foi chamado de Personal Home Page Tools. O PHP começou desta forma. E a ideia era muito boa.

Com o tempo, Rasmus foi melhorando suas aplicações e em 1997 ele lançou o PHP/FI, o FI é de Forms Interpreter, que era um interpretador de comandos SQL. O que possibilitou o uso de bancos de dados. Outros programadores entraram para o time de desenvolvimento do PHP, até que Zeev Suraski lançou o PHP 3.

Para saber mais sobre o PHP acesse o link abaixo:
http://pt.wikipedia.org/wiki/PHP

Pouco tempo depois, Zeev e Andi Gutmans escreveram o PHP 4, que substituiu totalmente o PHP 3 e deu muito mais recursos e poder ao PHP, além de implantar um novo paradigma de programação: a orientação a objeto. Nesta competência, utilizaremos o paradigma procedural, que é mais fácil de ser entendido. Na próxima, aprenderemos a programar em PHP em orientação a objetos, que é o paradigma dominante atualmente.

1.3 Servidor de Teste

Um servidor é um programa de computador que serve algo. Podemos ter um servidor de e-mail que serve e-mails, um servidor de stream de vídeos que é quem envia o vídeo, como no YouTube, e também temos um servidor que serve as páginas web.

A instalação do PHP deve ser feita em um servidor web já instalado. Então, para testarmos nossas páginas PHP precisamos de uma empresa que forneça isso? Não.

Podemos instalar um servidor em nosso computador junto com o PHP e testarmos nosso código PHP localmente. A isso, damos o nome de **servidor local**. Atente para o fato de que nosso servidor é local, ou seja, as páginas apenas serão vistas em nosso computador.

Quando nossa programação estiver pronta, podemos enviá-las para um servidor que contratarmos para que seja disponibilizada mundialmente. Existem algumas empresas que disponibilizam pequenas aplicações PHP de forma gratuita, algumas colocam propaganda nas páginas como forma de rendimento.

Parece muito complicado? O pessoal do **Apache Friends**, Figura 2, também achou e desenvolveu um pacote já com tudo de que precisamos para testar as páginas PHP, e o melhor, sem precisar instalar nada. Você pode levar todo o seu projeto, junto com servidor, banco de dados, etc., prontinho no pen drive. E ainda mostrar no computador do cliente ou continuar o desenvolvimento em outros computadores sem se preocupar em instalar o servidor de teste. Legal, né?

Então, vamos baixar esse pacote “mágico”, seu nome é **XAMPP**. Ele é todo em português e você não precisa instalar. O arquivo do XAMPP Portable tem 150 Mb e está compactado em ZIP, o link para baixar é :**https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/5.6.19/xampp-porta ble-win32-5.6.19-0-VC11.zip/download**

Portable é o tipo de programa que pode ser executado em uma máquina sem que precise ser instalado. Ótimo para ser transportado utilizando um pen drive.

Figura 2 – O site da Apache Friends possui versões para Linux e Mac OS X.
Fonte: www.apachefriends.org/pt_br/index.html. (2016).
Descrição: Captura de tela do endereço www.apachefriends.org/pt_br/index.html. Você pode baixar a versão para seu sistema operacional por aqui.

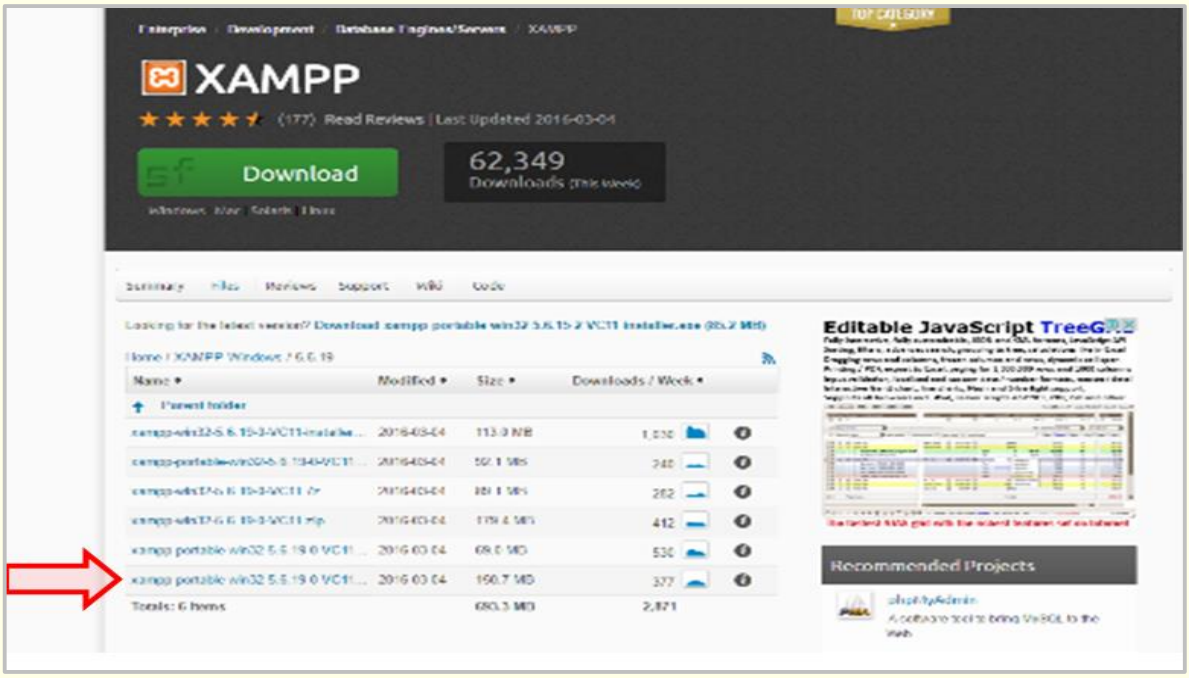


Figura 3 - Link para baixar a versão portable do XAMPP.
Fonte: próprio autor(2016).
Descrição: Captura de tela do site para donwload. Uma seta aponta o link correto para baixar a versão portable para Windows. O link direto é o https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/5.6.19/.

Descompacte o arquivo em um bom lugar, **preferencialmente na raiz de seu sistema operacional**, e memorize sua localização. Dentro há vários programas, entre eles o **Apache**, que é um servidor de páginas web com o **PHP** já instalado e o banco de dados web **MySQL**. Além disso, o pessoal ainda colocou o phpMyAdmin, um aplicativo web feito com PHP para gerenciar os bancos do MySQL. Com ele você pode criar as tabelas, campos, visualizar e gerenciar, tudo no MySQL.

Antes de podermos testar o servidor, temos que configurar sua localização. Não se preocupe que tudo será muito fácil. Dentro da pasta que você descompactou procure o arquivo **setup_xampp.bat** e execute-o. O que ele faz é descobrir onde estão os servidores. Isso só é preciso porque você pode levar esta pasta no pen drive para outros lugares, assim a localização pode mudar muito. Sempre que você mudar a pasta de lugar, vai ter que executar este arquivo para configurar a localização, mas se não mudar de lugar não precisa repetir o processo.

Irá aparecer a janela da Figura 4. Caso apareça mensagens de alerta do Firewall, permita o acesso. Aguarde até aparecer uma mensagem dizendo que a atualização foi realizada. Pressione, então, qualquer tecla para que a janela desapareça.

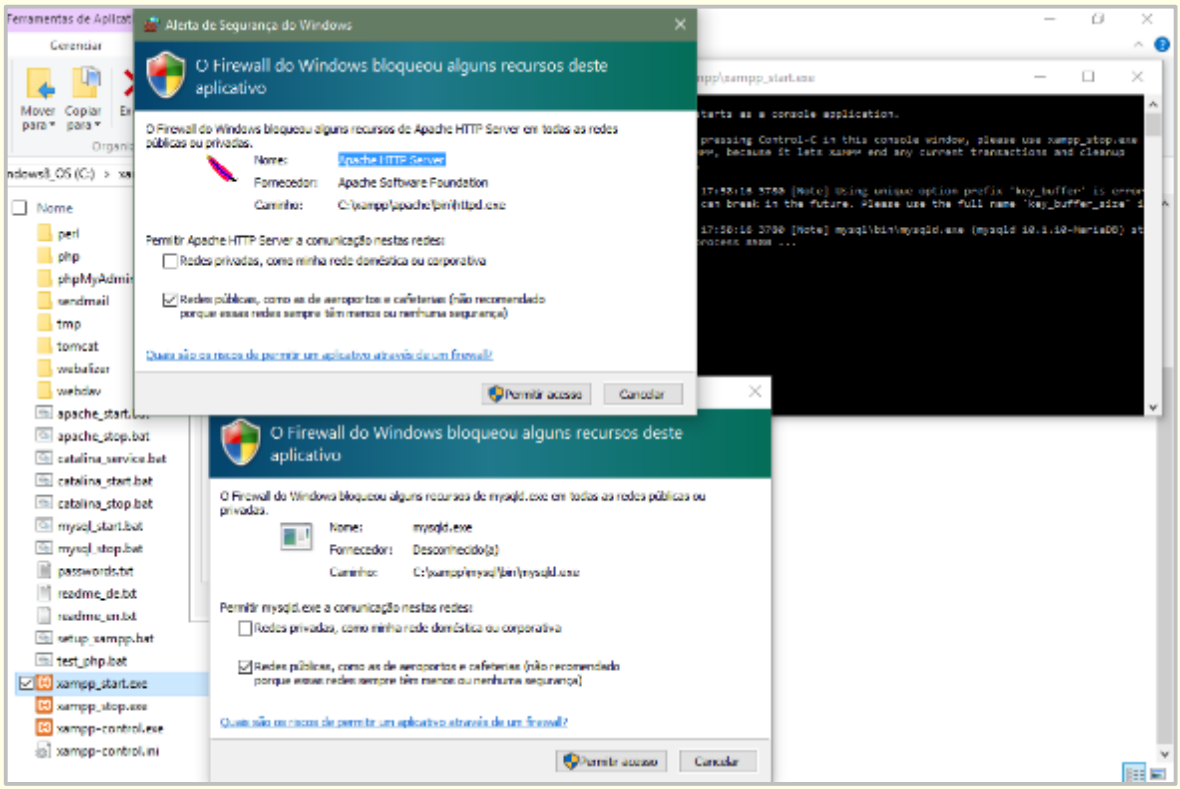


Figura 4 – Captura de tela com as caixas de diálogo do sistema operacional Windows 10, solicitando permissão para que os servidores acessem a rede.
Fonte: próprio autor (2016).
Descrição: Execute o arquivo setup_xampp.bat. Mensagens de alerta podem aparecer. Permita o acesso. Automaticamente o script irá atualizar a localização dos servidores.

Uma vez configurado vamos executar o arquivo **xampp-control.exe**. Ele é um programa que gerencia os servidores. Na primeira execução ele pergunta a linguagem que será utilizada, escolha a bandeira americana e pressione “Save”. A Figura 5 mostra sua tela, ela nos ajuda a ligar os outros programas. Pressione os botões “Start” para ligar o Apache e o MySQL. Vai aparecer a tela de permissão do Windows, **permita o acesso**. Só é necessário fazer isso na primeira vez.

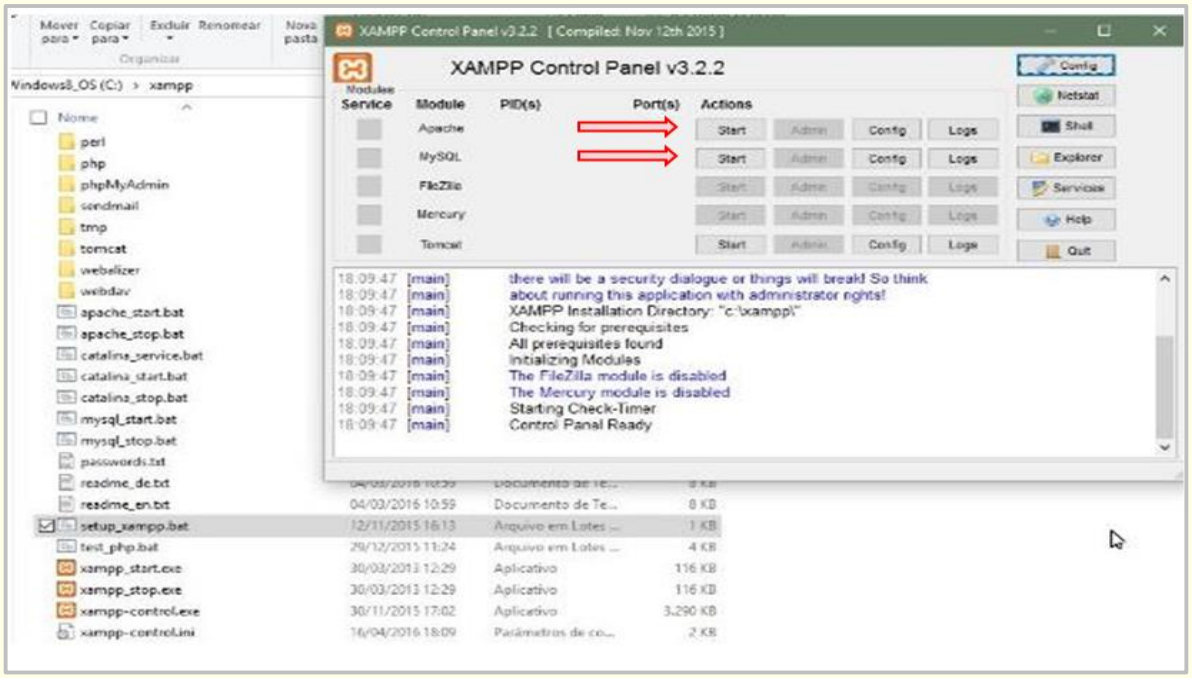


Figura 5 – Painel de Controle do XAMPP. Pressione os dois botões “Start” apontados.
Fonte: próprio autor.
Descrição: Captura de tela Controle do XAMPP.

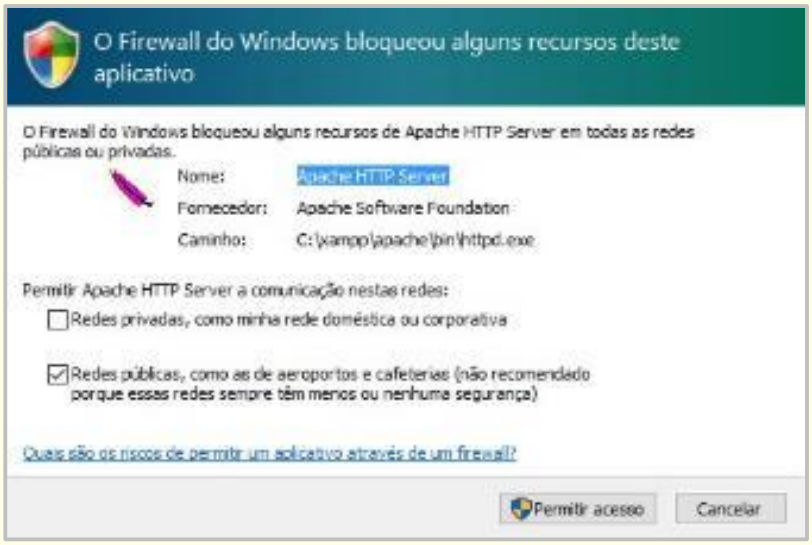


Figura 6 – Caixa de diálogo da tela de permissão do Windows 10.
Fonte: próprio autor (2016).
Descrição: Captura de tela da caixa de diálogo da tela de permissão do Windows.

Espere algum tempo para os servidores começarem e você verá a tela da Figura 7, com os nomes dos aplicativos destacados em verde.

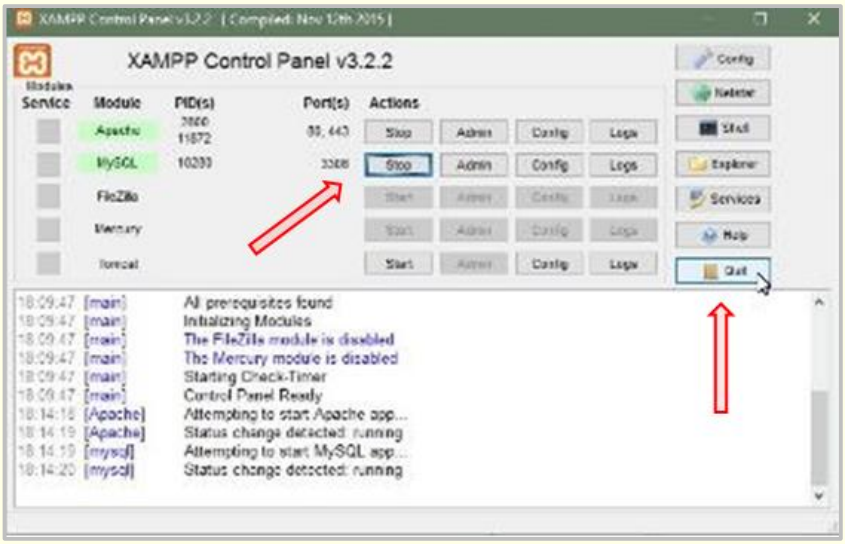


Figura 7 – Painel de Controle do XAMPP.
Fonte: próprio autor (2016).
Descrição: Captura de tela do painel de controle do XAMPP, mostrando, com setas, os botões para parar a aplicação e o botão de sair.

Para desligar tudo você primeiro deve clicar nos botões “Stop” e esperar para desligar os servidores, depois pressionar o botão “Quit” apontado na Figura 7. Se não fizer isso, o painel de controle pode desligar e deixar os servidores funcionando. Tenha atenção!

Agora que tudo está instalado, configurado e funcionando vamos testar nosso ambiente de teste. Abra seu navegador preferido e digite o endereço **http://localhost**. A página da Figura 8 deve abrir. A página está em inglês, mas o PHPMyAdmim está em português. Esta página não está na internet, foi servida pelo seu servidor local.

Algumas coisas mudaram nos últimos tempos. O MySQL pertence a uma empresa privada que não cobrava pelo seu uso, mas agora resolveu cobrar. A Google criou um projeto aberto para criar um substituto para o MySQL totalmente livre, este projeto é o MariaDB. Ele é igual ao MySQL e você não irá precisar mudar quase nada para utilizar o MariaDB ao invés do MySQL.

Figura 8 – Página inicial do XAMPP.
Fonte: próprio autor (2016).
Descrição: Exibição da página inicial do XAMPP da versão 5.6.19.

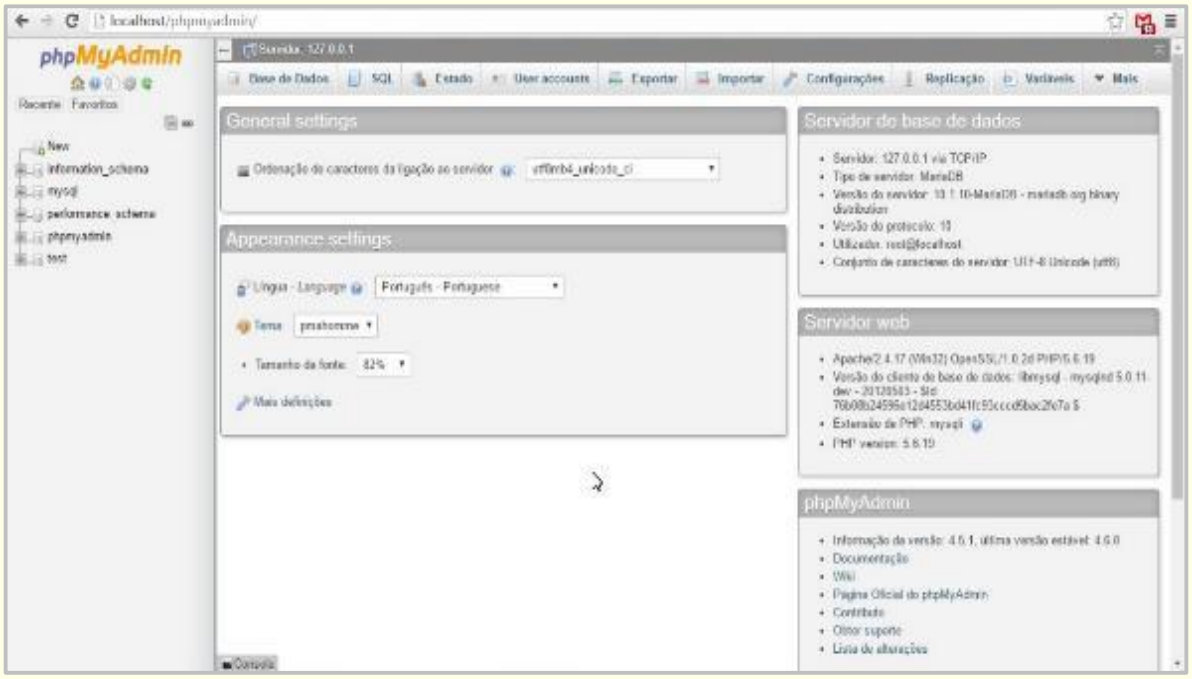


Figura 9 – Página do PHPMyAdmin.
Fonte: próprio autor (2016).
Descrição: Página inicial do aplicativo em PHP para gerenciar o banco de dados.

Pronto! Tudo está ok.

1.4 IDE

Uma IDE é um ambiente integrado de desenvolvimento, ou seja, um ambiente que reúne vários aplicativos que são necessários e úteis para o desenvolvimento. Existem diversas IDEs para desenvolvimento em PHP, algumas pagas, outras gratuitas.

Entre elas está o **Aptana Studio 3**. Ele foi desenvolvido com base em outra IDE muito famosa no mundo dos programadores chamada Eclipse. Ela é gratuita, completa e portável para vários sistemas operacionais. Além disso, muitos desenvolvedores profissionais a utilizam, Figura 10.

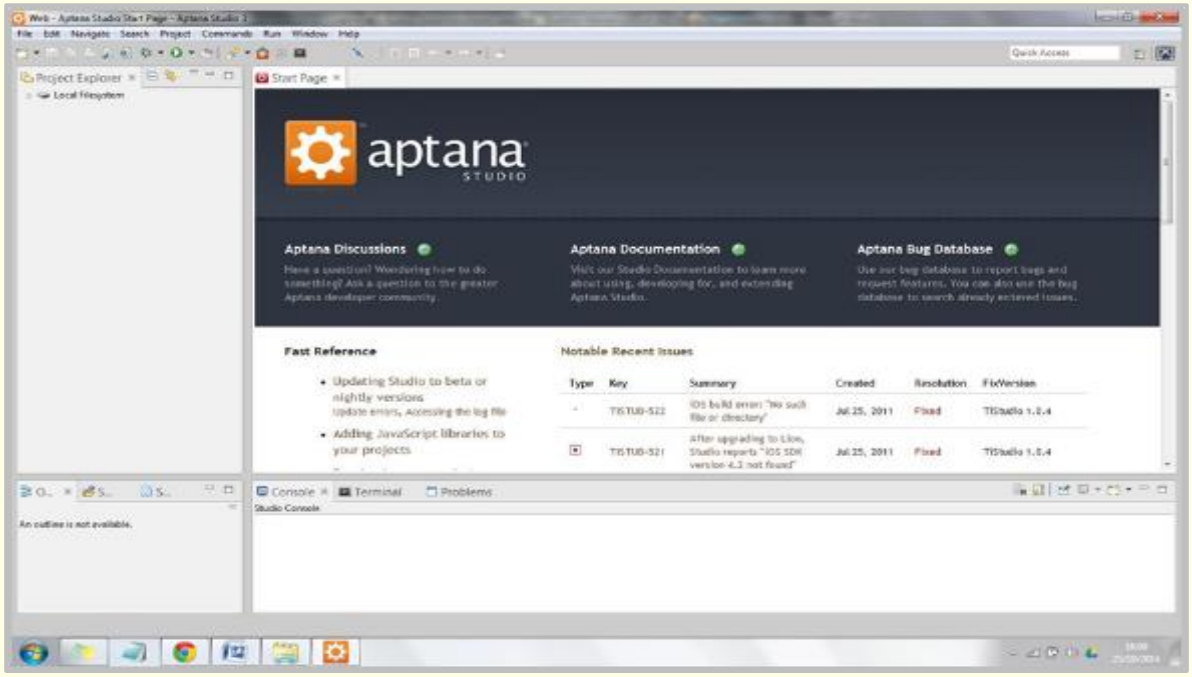


Figura 10 – Site do Aptana Studio 3.
Fonte: próprio autor (2016).
Descrição: Tela inicial do Aptana Studio 3.

Outra IDE bastante utilizada no mundo Linux é o Notepad++. Também gratuita, é bem mais leve que o Aptana e possui uma coleção de plug-ins que aumentam a capacidade do programa básico, Figura 11.

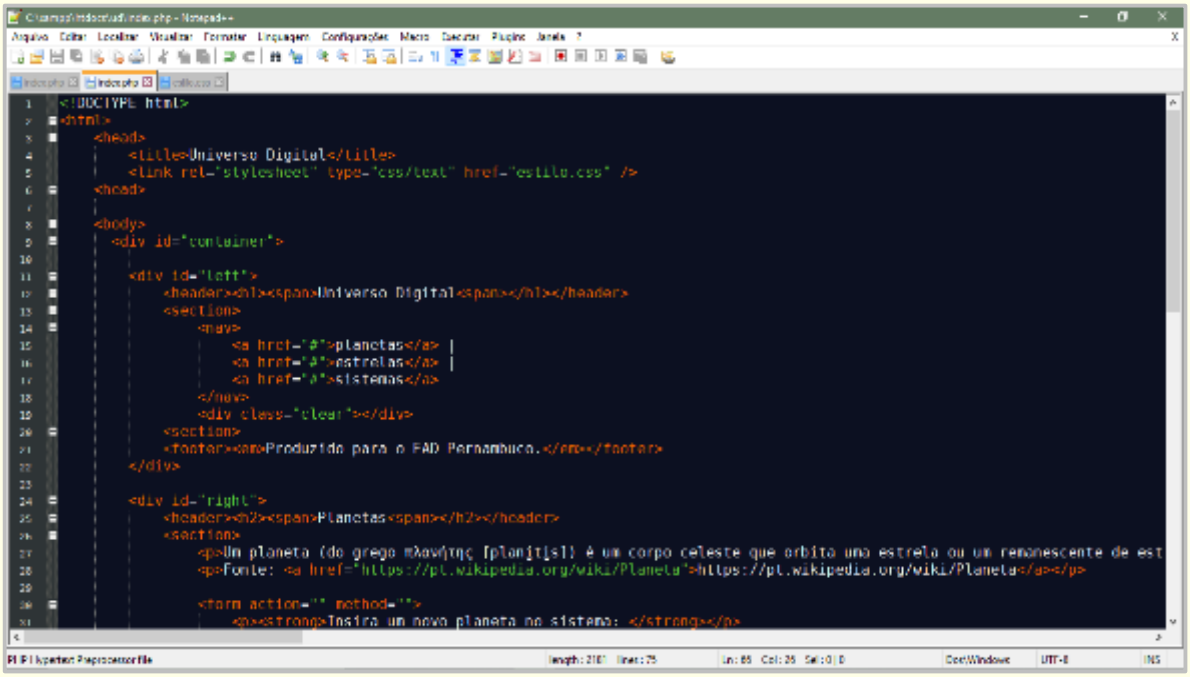


Figura 11 – Notepad++.
Fonte: próprio autor (2016).
Descrição: Tela inicial do Notepad++.

Existem diversas outras com capacidades desenvolvidas para agilizar e melhorar ao máximo o tempo de produção, porém todas elas apresentam um problema quando se está aprendendo. A concentração e o processo de procura de erros é fundamental no aprendizado. É através da repetição que aprendemos e várias pesquisas em neurocognição mostram que através da repetição aprendemos mais e melhor. Assim, o uso de uma destas ferramentas durante o aprendizado vai prejudicar e muito a absorção do conhecimento. Então, indicamos o uso do Bloco de Notas no Windows (Figura 12), gEdit no Linux (Figura 13) e text-edit no Mac (Figura 14).

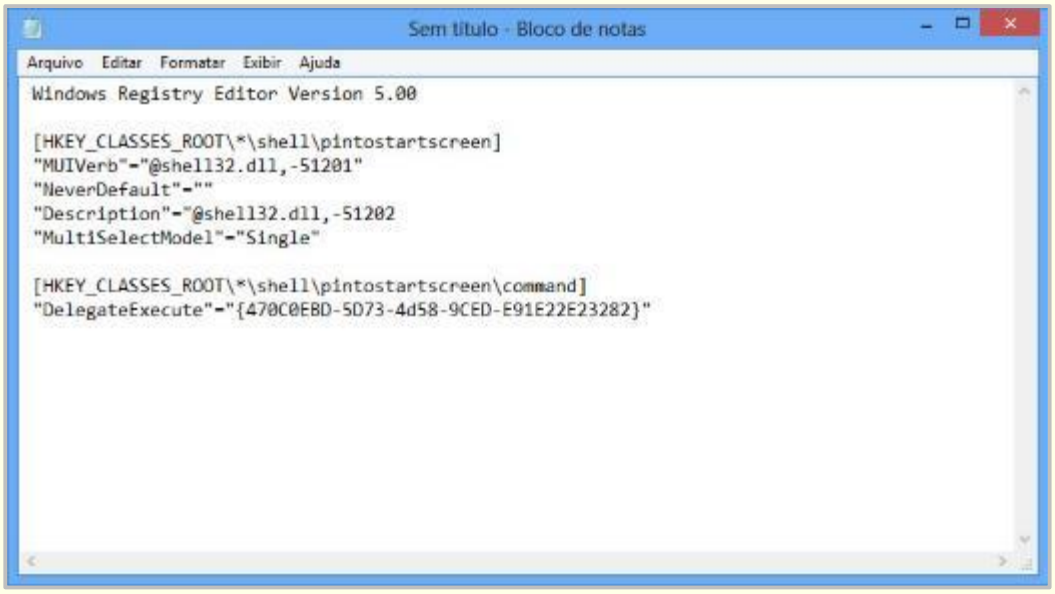


Figura 12 – Bloco de Notas do Windows 10.
Fonte: www.dicasparacomputador.com/files/u5/windows8-codigo-bloco-notas.jpg (2016).
Descrição: Tela do Bloco de Notas do Windows 10.

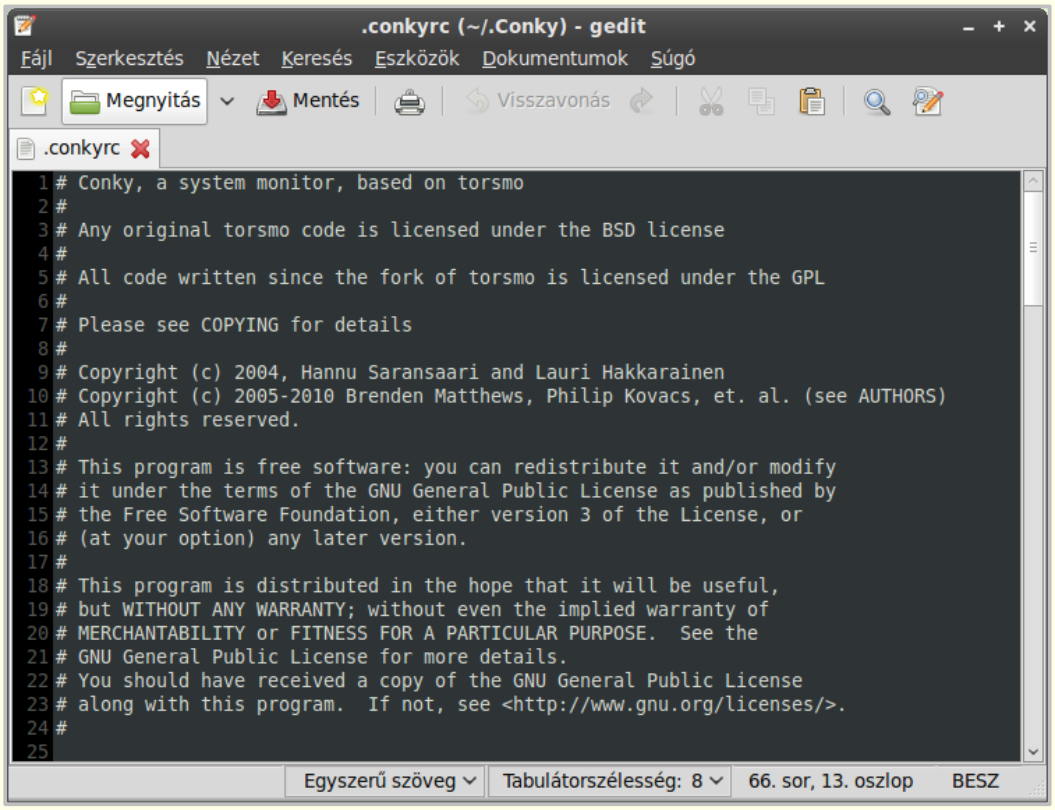


Figura 13 – gEdit do Linux.
Fonte: https://community.linuxmint.com/img/screenshots/gedit.png (2016).
Descrição: Tela do gEdit do Linux.

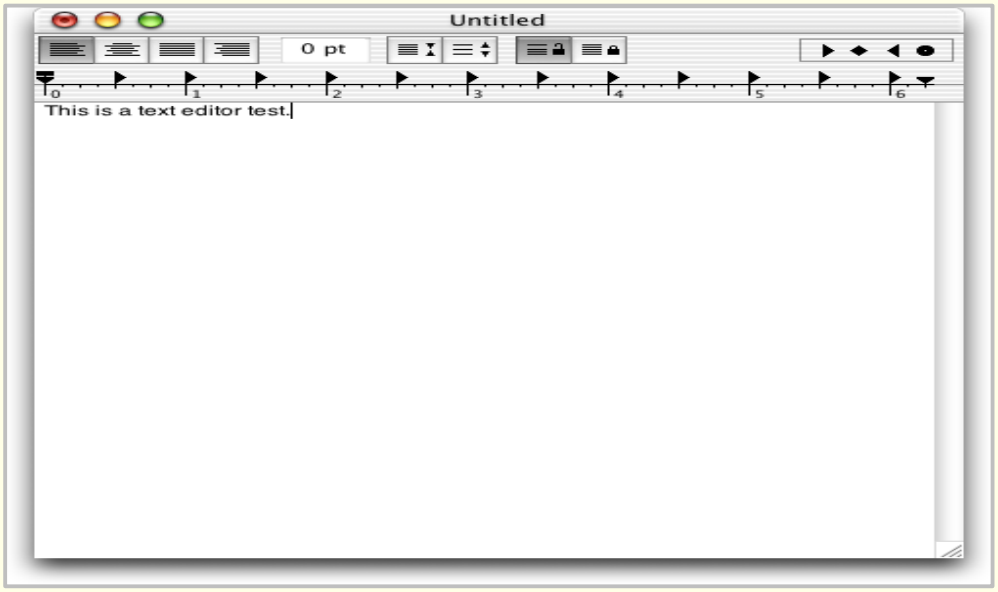


Figura 14 – Text-edit do Mac.
Fonte: www.guidebookgallery.org/pics/gui/applications/office/texteditor/macosx102.png (2016).
Descrição: Tela do Text-edit do Mac.

Esses programas vão servir muito bem para escrever os códigos. No futuro, quando você já conhecer a linguagem, utilize algumas IDEs e escolha aquela que melhor lhe agradar para ser sua ferramenta de trabalho. Assim, não é permitido o envio de atividades que utilizem IDEs profissionais.

1.5 Testando o Ambiente

Agora que temos tudo pronto, vamos testar o nosso ambiente de desenvolvimento. Para isso, vamos fazer uma página simples em PHP, o nosso “Hello world!”.

Abra o Bloco de Notas, ou equivalente, e selecione **Arquivo > Salvar como...**, de acordo com a Figura 15. Os passos nos outros aplicativos são semelhantes.

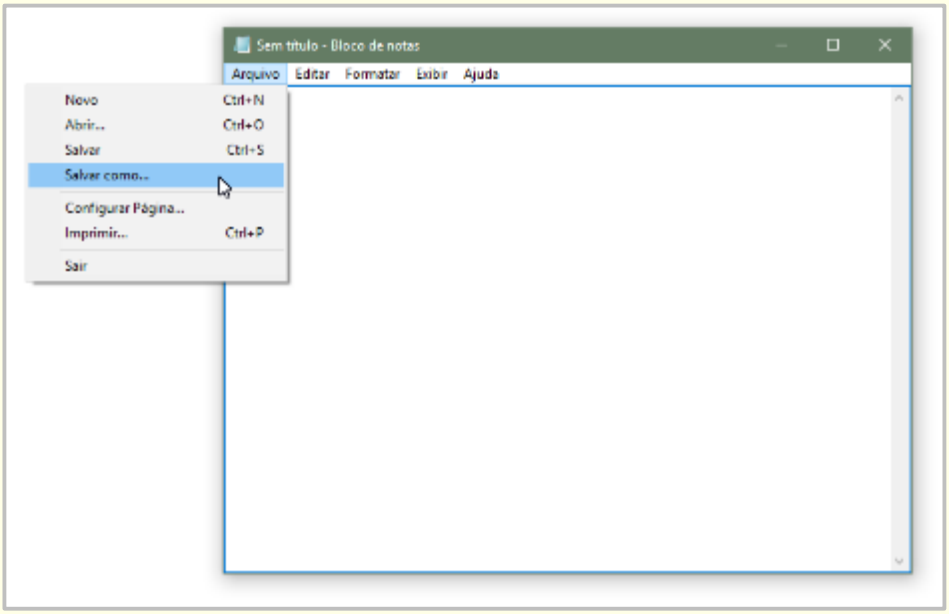


Figura 15 – Menu do Bloco de Notas.
Fonte: próprio autor (2016).
Descrição: A figura mostra o item “Salvar como...” no menu Arquivo no Bloco de Notas.

Na janela que se abre selecione a pasta do XAMPP que você descompactou, procure a pasta htdocs. É nesta pasta que ficam os arquivos disponibilizados para a internet pelo servidor. Nela vamos criar uma pasta com um nome simples e todo em minúsculo, não utilize caracteres especiais como cedilha, til ou palavras compostas. Vamos colocar apenas “helloworld” como nome da pasta. Entre nesta pasta e salve o arquivo como index.html. Este nome de arquivo é o principal em uma página web. A primeira página.

Tenha cuidado quando escrever os nomes. Qualquer erro e não funcionará da forma esperada. Caso isto aconteça, procure algum erro ou modificação da maneira que foi solicitada para fazer.

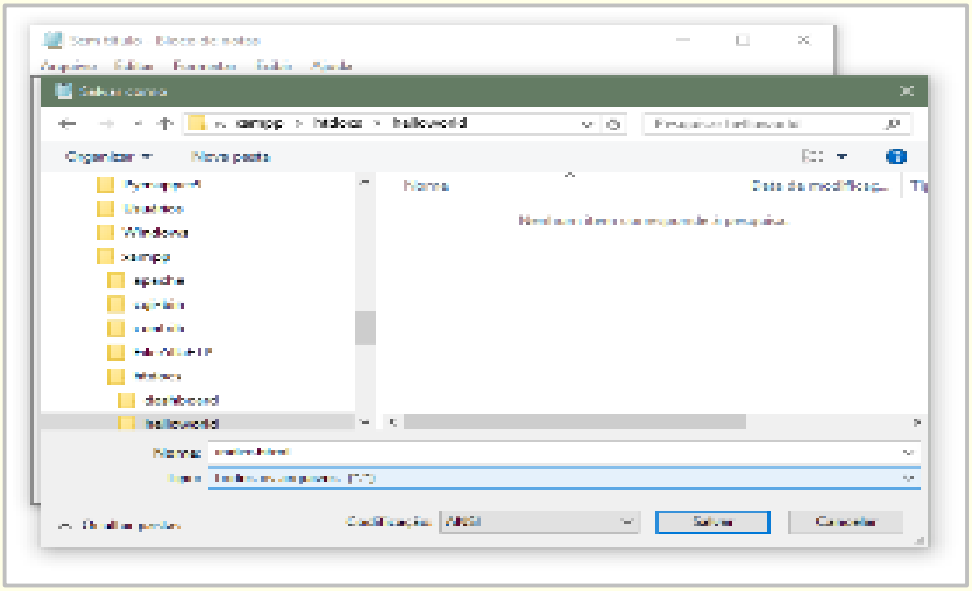


Figura 16 – Menu do Bloco de Notas.
Fonte: próprio autor (2016).
Descrição: A figura mostra o item “Salvar como...” no menu Arquivo no Bloco de Notas.

Como o projeto é destinado à web, não utilize espaçamentos para nomes compostos ou caracteres. Você também deve ter este cuidado para os nomes dos arquivos das páginas. Não são em todos os casos que podemos utilizar qualquer nome de arquivo, então, evite.

Para finalizar vamos editar o arquivo. Escreva **echo “Oi mundo!”;**. Por enquanto, irei explicar apenas que esta linha escreve o que está escrito entre aspas. Selecione **Arquivo > Salvar** para salvar o arquivo. Na janela que abre salve o arquivo na pasta **“helloworld”** com o nome **“index.php”** e modifique o tipo do arquivo para **“Todos os arquivo (*.*)”**de acordo com a Figura 17.

Existe uma configuração em todo servidor web que configura a primeira página a ser lida de um site como index.html ou index.htm ou default.html ou default.htm, mas também é aceito index.php quando o PHP está instalado no servidor.

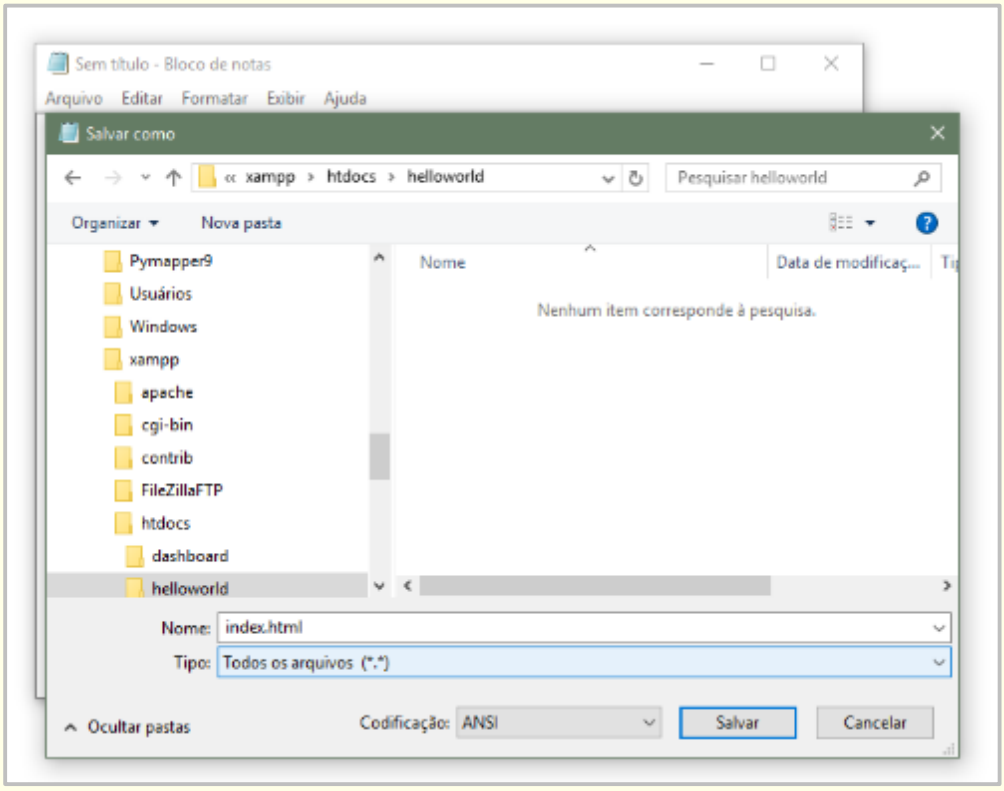


Figura 17 – Caixa de diálogo 'Salvar como' do Bloco de Notas.
Fonte: próprio autor(2016).
Descrição: Alerta para não se esqueça de modificar o tipo de arquivo para "Todos os arquivos (*.*)".

Para finalizar, vamos para o navegador onde digitaremos o endereço de nosso projeto: **http://localhost/helloworld/index.php**. Observe que uma página foi criada pelo PHP e seu conteúdo é o texto que colocamos.

Figura 18 – Exibição de nosso teste no navegador.
Fonte: próprio autor (2016).
Descrição: Navegador apresenta a página resultante do teste, exibindo o texto "Oi mundo!".

1.6 Projeto

Apenas aprender a nos comunicar com o computador para transmitir ordens não é suficiente. Nós temos que saber o que pedir. Para isso, uma aplicação web foi desenvolvida como exemplo de utilização da tecnologia. Nela foram utilizados um CSS personalizado. Você não precisa saber HTML

e CSS para aprender PHP, mas como o PHP gera página HTML é muito importante que você tenha algum conhecimento em HTML e o CSS.

Para fazê-lo é essencial que se dedique nas próximas competências para entender todas as palavras chaves e estruturas da linguagem PHP. Não fique com dúvidas. Caso o caderno de estudos ou os vídeos disponibilizados não sejam suficientes, procure ajuda no fórum do AVA. Se mesmo assim ainda não resolver, dedique-se mais um pouco e procure tutoriais na internet ou vídeo tutoriais no YouTube. Tem muita gente legal disposta a ajudar sem receber nada em troca e, às vezes, uma forma diferente de explicar pode ajudar bastante, mas tudo depende de seu esforço.

Figura 19 – Página inicial do projeto Universo Binário.
Fonte: Próprio autor (2016).
Descrição: Página inicial da aplicação de exemplo, Universo Binário, um cadastro de corpos celestes.

Este vídeo demonstra o nosso passo a passo de teste e contém mais algumas informações sobre a administração de projetos no Aptana Studio 3. Seria muito bom que você assistisse.
www.youtube.com/watch?v=MD69zTMobSM

2.Competência 02 | Fundamentos da Linguagem Php

Esta competência é muito importante. Ela é a base para todo o resto. Seu comprometimento no estudo deve ser total, pois é a fundação para as próximas competências até finalizarmos o sistema Universo Digital. Assim, faça todos os exemplos e atividades e os repita para a melhor compreensão.

Aqui, aprenderemos as palavras-chaves básicas e a forma de ordená-las. Assim, as ordens passadas para o computador farão sentido e ele poderá cumpri-las.

Atente que a maior parte dos erros, quando se está aprendendo programação são erros de digitação. Quando alguma mensagem de erro aparecer, veja o número da linha que deu erro e procure por algo digitado errado nela e nas anteriores.

2.1 Páginas PHP

Como dito anteriormente, o PHP é uma linguagem de programação que é executada do lado do servidor. Ela é diferente, por exemplo, do JavaScript, que também é uma linguagem interpretada, mas executada ao lado do cliente, ou seja, no navegador do usuário e não no servidor.

Vamos ver o que acontece quando se usa páginas PHP em um site.

Acompanhe o fluxo na Figura 19. Primeiramente o usuário faz um pedido de uma página pelo endereço DNS, ou clicando em um link que possui um endereço DNS (1). Quando a solicitação chega ao servidor, ele verifica se o que foi pedido é uma página PHP e entrega para o interpretador (2). O interpretador lê a página e quando encontra um código PHP o executa (3). Quando termina de ler, o interpretador constrói uma página HTML como resultado da computação realizada e entrega ao servidor (4). Finalmente o servidor envia a página HTML resultante para o usuário.

Todas as vezes que são feitas requisições de página PHP, esse processo é repetido.

Mais adiante, observe que as páginas PHP são páginas HTML com código PHP misturado. Para que o servidor saiba que são páginas PHP, ao invés de colocarmos a extensão **.html,** colocamos **.php**. Podemos enviar para o servidor uma página PHP contendo apenas código HTML. O servidor vai ter todo o trabalho descrito na Figura 20 e irá retornar à mesma página, sem realizar qualquer computação. Quando formos fazer nosso sistema de locadora teremos casos como este.

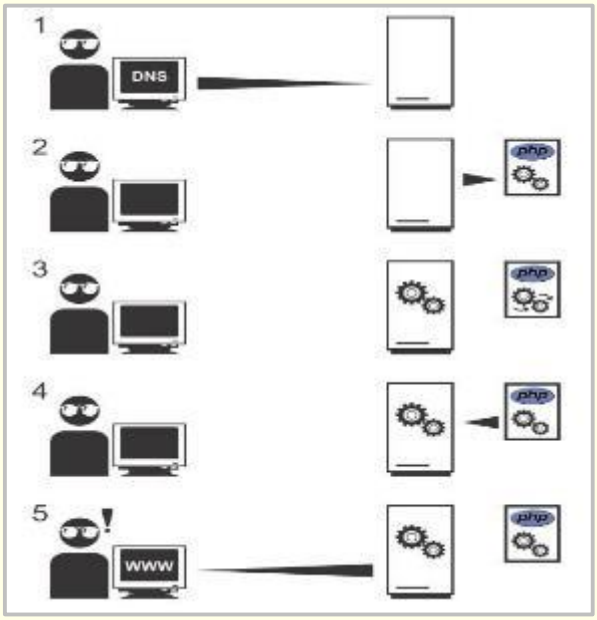


Figura 20 – Ilustração do passo a passo de uma solicitação de uma página PHP.
Fonte: Próprio autor (2016).
Descrição: Ilustração que mostra o fluxo de execução de páginas PHP.

2.2 Sintaxe Básica

Para que uma página seja interpretada pelo PHP, ela deve ter a extensão **.php**, mas não só isso, todo código PHP deve estar entre dois sinais que indicam o início e o fim de um pedaço de computação. Assim, o interpretador vai saber que o que está dentro é comando PHP.

Então, para marcar o início utilizamos **<?php** e para marcar o final utilizamos **?>**.

Vamos ver um exemplo.

O comando **echo** escreve o que está logo após, entre aspas duplas. Vamos escrever “Oi mundo!”, utilizando PHP. Veja a Figura 21 que exibe o conteúdo de oi.php.

Figura 21 – Código de exemplo “Oi mundo!”.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de sintaxe básica em PHP.

Você pode abrir e fechar várias sessões de códigos PHP misturados na mesma página HTML. Lembre-se de sempre começar com <?php e terminar com ?> antes de começar outra sessão.

Na sessão 2.5 vimos como testar o ambiente. Teste esta página para ver o resultado exibido na Figura 22. Tenha certeza de ter ligado o XAMPP adequadamente como explicado na sessão 2.3.

Figura 22 – Navegador exibindo o resultado da página oi.php.
Fonte: Próprio autor (2016).
Descrição: Exibição no navegador do resultado do código de exemplo.

Faça uma página PHP como no exemplo, mas coloque duas ou mais sessões PHP misturadas à página HTML. Veja como foi que o PHP gerou a página HTML para o usuário, exibindo o código HTML.

Quando escrevemos sistemas, podemos chegar a um número grande de linhas de código e esquecer porque fizemos determinado pedaço, pode acontecer também de outros programadores precisarem ler nossa codificação. Para ajudar o entendimento, podemos documentar com

comentários o código PHP.

Para fazer um comentário de uma linha colocamos **//**.

Para fazer um comentário de várias linhas colocamos no começo **/*** e no final das linhas ***/**. As linhas comentadas são ignoradas pelo interpretador na hora da computação. Veja o exemplo na Figura 23.

```
<?php
    // Comentário de uma linha
    echo "Oi mundo!";
    /*
     echo "esta linha é ignorada pelo interpretador.";
     */
?>
```

Figura 23– Comentários no código.
Fonte: Próprio autor (2016).
Descrição: Exibição do código de comentários.

2.3 Variáveis

Variáveis são como “gavetas” ou “caixas” na memória do computador onde guardamos algo. Para trabalhar com dados, precisamos guardar seus valores em um lugar que seja fácil de encontrar mais tarde. Por isso, cada “gaveta” da memória deve ser etiquetada. Estas gavetas chamam-se variáveis.

O nome das variáveis pode ser composto de mais de uma letra. Coloque nomes que relacionem com o conteúdo da variável. Outro ponto: podemos guardar não só valores nas variáveis, mas também outras coisas.

É muito parecido com matemática. Veja o exemplo:

x = 2
y = 5
z = x + y

Na matemática nós utilizamos as letras para representar um valor. Dessa forma podemos entender que **z** representa o valor de **7** em nosso exemplo. Em PHP estas letras são as variáveis.

Da mesma forma que na matemática, as variáveis em PHP podem ser utilizadas para guardar valores (x=2) e expressões (z=x+y).

Faça uma página PHP que some 2 * 34 – (40 / 2) e escreva o resultado dentro de uma tag <h1>.
Não se esqueça do restante das tags HTML de uma página.

2.3.1 Nome de Variáveis

O nome da variável não precisa ter apenas uma letra, como x e y. Seria muito melhor que descrevessem o que ela guarda, por exemplo, idade, nome, volume, etc.

Você pode colocar qualquer um desses, desde que respeite as regras abaixo:

- Uma variável deve começar com o sinal de cifrão $, seguido pelo nome da variável;
- O nome de uma variável deve começar com uma letra ou com os caracteres de sublinhado;
- O nome de uma variável só pode conter caracteres alfanuméricos e sublinhados (Az, 0-9 e _);
- O nome de uma variável não deve conter espaços;
- PHP distingue maiúsculas de minúsculas, dessa forma $nome, $Nome, $NOME e $nOmE são variáveis diferentes.

2.3.2 Declaração de Variáveis

Para criar uma variável em PHP temos apenas que atribuir um valor a ela. Utilizamos o sinal de igual (**=**) para isso. Por exemplo:

```
$nome = “Fulano de Tal”;
$idade = 37;
```

Quando guardamos uma sequência de caracteres utilizamos as aspas duplas para marcar seu começo e seu fim, como no exemplo anterior.

Quando executarmos o programa, o computador guarda na memória o nome **Fulano de Tal** e o número **37** para usarmos em outro momento.

No entanto, as variáveis não vivem para sempre. Em certo momento elas serão apagadas, normalmente ao final de seu escopo, mas o que é escopo de uma variável?

2.3.3 Escopo de uma Variável

Não podemos acessar nossas variáveis em qualquer lugar do código. Elas pertencem a um determinado local, a este local chamamos de escopo.

Temos quatro tipos de escopo:

- Local;
- Global;
- Estático;
- Parâmetro.

2.3.3.1 Escopo Local

Uma variável declarada dentro de uma função é local e só pode ser acessada dentro desta função. Veremos o que é uma função na próxima competência. Mas entenda, de antemão, que uma função é como um pequeno programa à parte. Vamos ver um exemplo na Figura 24:

```
<?php
function teste() {
    $x = 2 + 2; // A variável $x só existe localmente.
} // Quando a função termina a variável $x é apagada da memória.

echo $x; // Quando esta linha for lida a variável $x não existe.
?>
```

Figura 24 – Código exemplo de variável local.
Fonte: Próprio autor (2016).
Descrição: código que soma 2 + 2 e mostra o resultado.

O PHP não consegue escrever o conteúdo da variável $x porque ela não existe no escopo global do PHP, só dentro da função, ou seja, seu escopo é local.

Como as variáveis locais estão separadas do resto do código, você pode ter variáveis locais com o mesmo nome em funções diferentes.

Quando o interpretador executa a última linha da função, as variáveis locais dela são apagadas.

2.3.3.2 Escopo Global

Uma variável que é declarada na raiz do código, como as variáveis $x e $y na Figura 25, não são visíveis dentro da função. Possuem escopos diferentes. Caso precisemos delas dentro de uma função, declaramos elas como **global**, de acordo como é mostrado na Figura 24. Assim, o interpretador sabe que não é para criar novas e sim utilizar as que foram criadas na raiz do código.
O modificador global dentro de uma função amplia o escopo da variável. Quando a função encerrar, estas variáveis não são apagadas porque elas foram criadas fora da função.

```php
<?php
$x=2; // Escopo global
$y=5; // Escopo global

function teste() {
    global $x, $y; // Aqui chamamos as variáveis que estão fora da função.
    $y = $x + $y; // Aqui somamos o conteúdo de $x e $y e colocamos dentro de $y
}

teste();
echo $y; // Escreve 7
?>
```

Figura 25 – Exemplo de variáveis com escopo global
Fonte: Próprio autor (2016).
Descrição: Código de uma função com configuração de variáveis de escopo glogal em uma função.

2.3.3.3 Escopo Estático

Quando o escopo de uma variável local ou global termina, ela é excluída, mas você pode querer que ela continue a existir por mais tempo. Sendo assim, utilize a palavra-chave **static** quando declarar a variável. Vamos ver um exemplo na Figura 26:

```php
<?php

function teste() {
    static $x=0; // Escopo estático
    echo $x; // Escreve seu valor
    $x++; // Soma mais um ao seu valor
}

teste(); // Aqui, quando chegasse no final, a variável $x era para ser excluida.
teste(); // Mas como ela foi declarada com static, não é apagada.
teste(); // E chamada novamente mostrando o valor acrescido.

?>
```

Figura 26 – Exemplo de variáveis com escopo estático.
Fonte: Próprio autor (2016).
Descrição: Código configurando uma variável para escopo estático.

Esta aplicação escreve **012**, mostrado que o valor persistiu depois que a função foi encerrada.

2.3.3.4 Escopo de Parâmetro

Podemos passar valores para as funções através dos parâmetros. Veremos melhor funções e parâmetros na próxima competência, mas por agora entenda que variáveis de parâmetro possuem escopo local. Vamos ver um exemplo na Figura 27.

```php
<?php

function teste($x) { // Aqui é criado uma variável de parâmetro
    echo $x; // Escreve o conteúdo da variável
}

teste(5); // Chama a função e passa para ela o valor 5
          // Este valor será colocado na variável de parâmetro $x
?>
```

Figura 27 – Exemplo de variável de parâmetro.
Fonte: Próprio autor (2016).
Descrição: Código mostrando uma variável de parâmetro de uma função.

Parâmetros também são chamados de argumentos. Veremos este assunto melhor quando estudarmos funções em PHP na próxima competência.

Além dos exercícios e atividades, sempre faça os exemplos para um melhor aprendizado. Só aprendemos programação fazendo e refazendo.

Crie algumas variáveis de escopo global, local, estática e parâmetro e teste utilizando o comando 'echo' nelas para entender melhor o escopo de cada uma.

2.4 String

Como vimos em exemplos anteriores, podemos guardar em uma variável uma sequência de caracteres, por exemplo, que formem o nome "Fulano de Tal". A essa sequência de caracteres damos o nome de string. O PHP tem diversas funções que manipulam strings. Vamos ver algumas delas mais adiante.

Atente para a utilização de aspas duplas para marcar o início e o fim da sequência de caracteres.

Podemos utilizar strings diretamente ou guardar em uma variável. Veja o exemplo abaixo na Figura 28.

```php
<?php
$txt = "Oi mundo!";
echo $txt;

echo "Fulano de Tal";
?>
```

Figura 28 – Exemplo de utilização de strings.
Fonte: próprio autor (2016).
Descrição: Código mostrando a utilização de variável string.

O PHP disponibiliza várias funções para a manipulação de strings. Fazendo uma busca na internet você poderá encontrar vários tutoriais e exemplos destas funções. Aqui vamos abordar algumas.

Para saber todas as funções manipuladoras de strings, acesse:
http://php.net/manual/pt_BR/ref.strings.php

2.4.1 Operação de Concatenação

Concatenar é unir, juntar. O operador de concatenação realiza uma operação de unir duas strings. Seu símbolo é o ponto e podemos unir duas ou mais strings em uma mesma expressão. Veja o exemplo da Figura 29.

```php
<?php
$txt1 = "Oi!";
$txt2 = "Fulando de Tal.";

echo $txt1 . " " . $txt2;
?>
```

Figura 29 – Exemplo de concatenação.
Fonte: Próprio autor (2016).
Descrição: Código mostrando a concatenação de strings.

Será escrito **Oi! Fulano de Tal.** Neste caso concatenamos um espaço em branco no meio de duas variáveis.

Faça uma aplicação em PHP que tenha uma variável para cada parte de seu nome, concatene todas e coloque na tag <title> de uma página HTML.

2.4.2 Função strlen( )

Entregamos como parâmetro desta função uma string e ela conta a quantidade de caracteres. A Figura 30 mostra um exemplo.

```
<?php
echo strlen("Oi mundo!");
?>
```

Figura 30 – Exemplo da função strlen( ).
Fonte: Próprio autor (2016).
Descrição: Código que exibe em uma página o resultado da quantidade de letras de uma string.

É escrito 9 como saída.

Faça uma aplicação em PHP que conte as letras de seu nome e dê o resultado em uma página HTML.

2.4.3 Função strpos( )

Esta função é usada para procurar uma string dentro de uma string. Caso ache a função, retorna ao número da posição do primeiro caractere da string procurada, caso não ache, retorna **falso**. Veja o exemplo da Figura 31.

```
<?php
echo strpos("Fulano de Tal","Tal");
?>
```

Figura 31 – Exemplo de uso da função strpos( ).
Fonte: Próprio autor (2016).
Descrição: Código que exibe a posição de onde começa a string 'Tal' dentro da string 'Fulando de Tal'.

É escrito 10.

A resposta do exemplo é 10 e não 11 porque em PHP começamos a contar de zero e não de um.

Faça um algoritmo (programa) que guarda seu nome completo em uma variável string, exibindo-o concatenado com a quantidade de letras e a posição de início de seu último nome.

2.5 Operadores

Os operadores servem para executar uma operação em uma expressão, como as operações de soma, subtração, multiplicação e divisão. É muito parecido como a matemática, mas em linguagens de programação temos vários outros.

Em PHP temos os seguintes operadores:

- Operadores Aritméticos;
- Operadores de Atribuição;
- Operadores de Incremento e Decremento;
- Operadores de Comparação;
- Operadores Lógicos.

2.5.1 Operadores Aritméticos

OPERADOR	NOME	DESCRIÇÃO	EXEMPLO	RESULTADO
x + y	Adição	Soma de x e y	2 + 2	4
x - y	Subtração	Subtração de x e y	7 - 2	5
x * y	Multiplicação	Produto de x por y	3 * 2	6
x / y	Divisão	Quociente de x por y	4 / 2	2
x % y	Módulo	Resto da divisão de x por y	5 % 2	1
-x	Negativo	Inverte o sinal de x	-2	2

Tabela 01 – Operadores aritméticos.
Fonte: Próprio autor.
Descrição: Tabela com a primeira coluna de operadores aritméticos, posteriormente colunas com o nome da operação, sua descrição, um exemplo e o resultado da operação de exemplo.

2.5.2 Operadores de Atribuição

OPERADOR	COMO SE FOSSE...	DESCRIÇÃO
x = y	x = y	Coloca no operando da esquerda o resultado ou valor da direita
x += y	x = x + y	Adiciona y a x
x -= y	x = x – y	Subtrai y de x
x *= y	x = x * y	Multiplica x por y
x /= y	x = x / y	Divide x por y
x %= y	x = x % y	Coloca em x o resto da divisão de x por y
x .= y	x = x . y	Concatena duas strings

Tabela 02 – Operadores de atribuição.
Fonte: Próprio autor (2016).
Descrição: Tabela com a primeira coluna do operador de atribuição, posteriormente com uma coluna do equivalente em operadores aritméticos e coluna de sua descrição.

2.5.3 Operadores de Unários

OPERADOR	NOME	DESCRIÇÃO
++x	pré-incremento	Soma um ao valor de x e depois retorna x
x++	pós-incremento	Retorna x e depois soma um ao valor de x
- - x	pré-decremento	Subtrai um do valor de x e depois retorna x
x- -	pós-decremento	Retorna x e depois subtrai um do valor de x

Tabela 03 – Operadores unários.
Fonte: Próprio autor.
Descrição: Tabela com a primeira coluna do operador unário, seguido por colunas com seu nome e descrição.

Os operadores de incremento e decremento são os mais difíceis de entender. Estude o exemplo por mais tempo até compreender porque foi exibido para x 9, 9 e depois 10.

2.5.4 Operadores de Comparação

OP.	NOME	DESCRIÇÃO	EX.	RES.
x == y	Igual	Retorna verdadeiro se for igual.	2 == 3	false
x != y	Não igual	Retorna verdadeiro se for diferente.	2 != 3	true
x <> y	Diferente	Retorna verdadeiro se for diferente.	2 <> 3	true
x > y	Maior que	Retorna verdadeiro se x for maior que y	3 > 2	true
x < y	Menor que	Retorna verdadeiro se x for menor que y	2 < 3	true
x >= y	Maior que ou igual	Retorna verdadeiro se x for maior que ou igual a y	3 >= 3	true
x <= y	Menor que ou igual	Retorna verdadeiro se x for menor que ou igual a y	2 <= 3	true

Tabela 04 – Operadores comparativos.
Fonte: Próprio autor (2016).
Descrição: Tabela com a primeira coluna do operador comparativo, seguido por colunas com o nome, a descrição, um exemplo e o resultado do exemplo dos operadores.

2.5.5 Operadores Lógicos

OPERADOR	NOME	DESCRIÇÃO	EXEMPLO	RESULTADO
x and y	e	Retorna verdadeiro se x e y forem verdadeiro	x=1; y=9 (x < 3 and y > 5)	true
x or y	ou	Retorna verdadeiro se houver um verdadeiro	x=1; y=9 (x == 1 or y == 2)	true
x && y	e	Retorna verdadeiro se x e y forem verdadeiro	x=1; y=9 (x < 3 && y > 5)	true
x || y	ou	Retorna verdadeiro se houver um verdadeiro	x=1; y=9 (x == 1 || y == 2)	true
!x	não	Retorna verdadeiro se x for falso	x = 1, y = 9 !(x == y)	true

Tabela 05 – Operadores lógicos.
Fonte: Próprio autor (2016).
Descrição: Tabela com os operadores lógicos na primeira coluna, seguida de colunas de seu nome, descrição, exemplo e resultado do exemplo.

O exemplo da Figura 32 mostra a utilização de operadores matemáticos, de atribuição e incremento. Os operadores comparativos e lógicos poderão ser demonstrados na próxima sessão, utilizando uma estrutura de decisão **if...else**.

```php
<?php
$x = 3;
$x += 5; // Operador de atribuiçção: soma 5 ao que estiver em $x
$y = 12 + 2 - 5 * 2 / $x; // Operadores matemáticos
echo "O valor de y é " . $y;

echo "    O valor de x é ". ++$x; //Soma um para $x e depois mostra o resultado
echo "    O valor de x é ". $x++; //Mostra o $x e depois soma mais um
echo "    O valor de x é ". $x; //Exibe o valor final de $x
?>
```

Figura 32 – Exemplo com operadores.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo utilizando alguns dos operadores.

Para fixar o que foi lido, pratique o seguinte exercício.
Faça um programa em PHP que calcule e exiba os dias aproximados que você viveu até hoje.
Sabendo que cada ano tem 365 dias, um mês 30 e uma semana 7.

2.6 Echo

Já utilizamos algumas vezes o **echo** para escrever os resultados da computação, mas o que está sendo exibido é texto sem as tags HTML. Podemos melhorar esta saída colocando qualquer tag HTML na string.

Não só podemos escrever tags, como também podemos colocar as variáveis na string e o interpretador PHP vai substituir seus nomes pelos respectivos valores. Observe o exemplo da Figura 33 e veja o resultado na Figura 34.

```php
<?php
$x = "Fulano de Tal";

echo "<h1>Ola, $x</h1>";
?>
```

Figura 33– Exemplo de echo com saída em HTML.
Fonte: Próprio autor (2016).
Descrição: Código que cria uma página com um título HTML com 'Ola' e o valor de uma variável.

Figura 34 – Exibição no navegador do resultado.
Fonte: Próprio autor (2016).
Descrição: Exibição do resultado do exemplo de código da Figura 31.

E se precisarmos escrever um cifrão ou aspas duplas? Podemos utilizar como alternativa para o echo uma string limitada por aspas simples. Dessa forma, nada é tratado e o que estiver na string vai ser impresso. Vamos tentar com o mesmo exemplo. Observe na Figura 35 o código e o resultado na Figura 36.

```php
<?php
$x = "Fulano de Tal";

echo '<h1>Ola, "$x"</h1>';
?>
```

Figura 35 – Utilizando aspas simples no echo.
Fonte: Próprio autor (2016).
Descrição: Código demonstrando a utilização de aspas simples.

Utilizamos apenas a tag <h1> devido ao espaço deste caderno, mas uma página web tem diversas outras tags. Faça um algoritmo que escreva uma página web simples, mas completa.

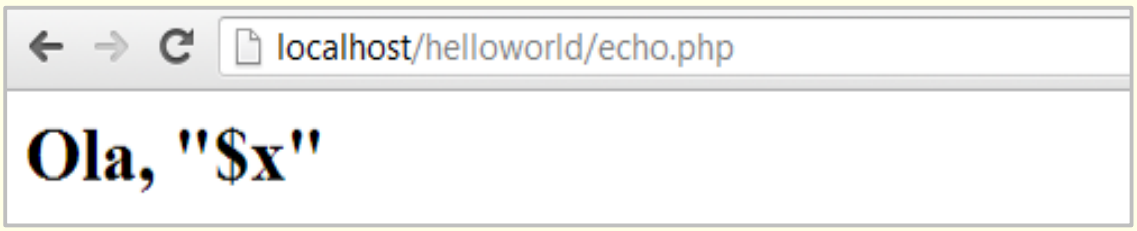


Figura 36 – Exibição com aspas simples.
Fonte: Próprio autor (2016).
Descrição: Exibição do resultado no navegador do código da Figura 33.

2.7 If...Else

Linguagens de programação têm duas estruturas em comum: estruturas de repetição e estruturas de condicionais. O **if...else** é uma estrutura condicional. Ela verifica uma condição, se for verdadeira faz uma coisa. Caso contrário, faz outra. Vamos ver sua sintaxe.

```
if (condição) comando;

if (condição) {
        // Bloco de código. Aqui pode ter várias linhas.
}
```

Nos casos acima o if verifica a condição, se for verdade ela executa um comando. Caso queira mais de um comando você deve criar um bloco de comando utilizando **{ e }**.

Caso a condição seja falsa, o comando ou bloco será ignorado.

O **else** significa "senão". Utilize-o para colocar um comando ou bloco de comandos, caso a condição seja falsa.

A Figura 37 mostra um exemplo.

```
<?php
$x = 10;
$y = 100;
if ($x < $y) echo "<p>x é menor do que y<p>";

if ($y > $x) {
    echo "<p>y é maior do que x, ";
    echo "o valor de y é " . $y . "</p>";
} else {
    echo "x é maior que y";
}
?>
```

Figura 37 – Exemplo da estrutura condicional if...else.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo da estrutura de decisão if...else que compara o valor de duas variáveis para saber se é maior ou menor.

O exemplo acima escreve **"x é menor do que y",** passa à próxima linha e escreve **"y é maior do que x, o valor de y é 100"**.

Chegou a hora de exercitar. Faça um algoritmo que tenha uma variável com do momento em que você está fazendo este exercício, por exemplo, 3hs da tarde. Utilize if...else para dar "Bom dia!" se for menor que 12 e "Boa tarde!" se não for. Veja se consegue dar também "Boa noite!" se passar das 18.

2.8 Switch

A estrutura **if...else** verifica se uma condição é verdadeira ou falsa. No entanto, podemos ter situações onde o resultado da verificação pode ser de mais de duas possibilidades. A escolha de um

menu é um exemplo dessa situação, onde podemos ter várias opções de escolha. Apesar de podermos utilizar várias estruturas de **if...else** para verificar o resultado da condição, serão necessárias várias linhas. Para encurtar e melhorar o entendimento é que foi criada a estrutura **switch**.

Switch é uma estrutura de condição, assim como o **if...else**, que verifica uma condição com várias possibilidades de resposta. Sua sintaxe está abaixo:

```
switch (n)  {
case x:
            código a ser executado se n igual a x;
            break;
case y:
             código a ser executado se n igual a y;
             break;
...
default:
             código a ser executado se n não for nenhuma das opções anteriores.
}
```

Como é que funciona?

O **switch** verifica a expressão **n**, que normalmente é uma variável. Ela é avaliada uma vez e, então, comparada com os valores de cada um dos **case** na estrutura, o **x** e o **y**. Se houver uma correspondência, o bloco de códigos depois dos dois pontos é executado.

Vamos fazer um exercício utilizando switch.
Faça o mesmo algoritmo da sessão anterior, dessa vez utilizando switch ao invés do if...else. Utilize as opções de “manhã”, “tarde” e “noite” para os cases. E exiba um “Bom dia!”, “Boa tarde!” e “Boa noite!” correspondente ao valor da variável condicional. Caso não seja nenhuma das opções, escreva um “Olá!”.

Utilize **break** para evitar que o código do próximo case seja executado automaticamente.

A declaração **default** é executada se não houver nenhuma correspondência.

A Figura 38 mostra um exemplo.

```php
<?php
$musica = "rock";
switch ($musica) {
    case "sertanejo":
        echo "Adoro sertanejo!";
        break;
    case "rock":
        echo "Rock'n Roll, baby!";
        break;
    case "samba":
        echo "Samba é o que tem de melhor!";
        break;
    default:
        echo "Não gosto nem de sertanejo, nem de rock, nem de samba, prefiro funk!";
}
?>
```

Figura 38 – Exemplo da estrutura Switch.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de uma estrutura de decisão switch.

O exemplo acima retorna **"Rock'n Roll, baby!"**. Você pode mudar o valor da variável **$musica** para ver os outros resultados. Também pode colocar um valor que não existe nas opções para exibir a opção **default**.

2.9 Array

Um **array** é uma variável que guarda variáveis. Ela é muito útil para armazenar coleções de um tipo, por exemplo, uma lista de nomes, onde cada nome é uma string. Se compararmos uma variável a uma gaveta onde colocamos coisas, um **array** é um armário com várias gavetas (Figura 39). **Arrays** também são chamados de matrizes, como em matemática.

$nomes =

0	Luis
1	Andreia
2	Marcos
3	Paula
4	Rodrigo

Figura 39 – Ilustração de uma variável array. direita.
Fonte: próprio autor (2016).
Descrição: Uma ilustração que mostra o conteúdo de uma variável array. A variável $nomes guarda a lista de nomes à direita.

Imagine que você possui uma lista de nomes de clientes. Se você quiser guardar cada um dos nomes, vai precisar de uma variável para cada nome. Mas se sua lista de clientes for enorme, aí vai ficar confuso ter tantas variáveis assim. Um **array** pode guardar toda a lista em uma única variável e podemos localizar cada uma das "gavetas" do **array** por um índice. Este índice é como uma etiqueta na gaveta.

Em PHP podemos ter dois tipos de matrizes:

- Matrizes indexadas;
- Matrizes associativas.

2.9.1 Arrays Indexados

Há duas maneiras de criar matrizes indexadas:

Na primeira não se coloca o índice e ele é atribuído automaticamente começando pelo 0. Veja a Figura 40.

```php
<?php
$nomes = array("Luis", "Andreia", "Marcos", "Paula", "Rodrigo");

echo "Agenda: " . $nomes[0] . ", " . $nomes[1] . ", " . $nomes[2] . ", " . $nomes[3] . ", " . $nomes[4] . ".";
?>
```

Figura 40 – Exemplo de array com índice automático.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de criação e utilização de uma variável array.

Na segunda forma é definido individualmente cada índice. Veja na Figura 41. Você pode definir qualquer numeração, inclusive começar pelo 1 ao invés do 0.

```php
<?php
$nomes[0] = "Luis";
$nomes[1] = "Andreia";
$nomes[2] = "Marcos";
$nomes[3] = "Paula";
$nomes[4] = "Rodrigo";

echo "Agenda: " . $nomes[0] . ", " . $nomes[1] . ", " . $nomes[2] . ", " . $nomes[3] . ", " . $nomes[4] . ".";
?>
```

Figura 41 – Exemplo de array com definição de índice.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de criação e utilização de uma variável array.

2.9.1.1 Contando os Elementos de um Array

A função **count( )** nos dá o total de elementos de um array. A Figura 42 nos traz um exemplo. Essa função é de muita importância, por isso decore-a bem. Ela é usada em estruturas de repetição para listas, de forma mais prática do que no exemplo da Figura 41. Estudaremos a estrutura de repetição mais adiante.

```php
<?php
$nomes = array("Luis", "Andreia", "Marcos", "Paula", "Rodrigo");
$tamanho = count($nomes);

for ($x=0; $x < $tamanho ;$x++) {
  echo $nomes[$x]; // O valor de $x começa em 0 e vai até 4
  echo "<br>";     // dessa forma imprimindo todo o array
}

?>
```

Figura 42 – Utilização da função count( ).
Fonte: Próprio autor (2016).
Descrição: Código que utiliza a função count() para contar os item de um array, junto com um laço de repetição que exibe o conteúdo do array.

2.9.2 Arrays Associativos

Arrays associativos são arrays que você associa com algum nome ao invés de um índice numérico. A Figura 43 mostra um exemplo. Tenha a atenção de observar que colocamos o nome da associação **'Luís'** com aspas simples, e retornamos o seu valor **'5555-3765'**.

```php
<?php
$telefones = array("Luis"=>"5555-3765", "Andreia"=>"5555-6537", "Marcos"=>"5555-6753");

echo "Telefone de Luis " . $telefones['Luis'];
?>
```

Figura 43 – Exemplo de array associativo.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de criação exibição de um array associativo.

Vamos fazer um exercício utilizando switch.
Faça o mesmo algoritmo da sessão anterior, dessa vez utilizando switch ao invés do if...else. Utilize as opções de "manhã", "tarde" e "noite" para os cases. E exiba um "Bom dia!", "Boa tarde!" e "Boa noite!" correspondente ao valor da variável condicional. Caso não seja nenhuma das opções, escreva um "Olá!".

2.9.2.1 Loops em Arrays Associativos

Os loops em arrays associativos são diferentes de arrays indexados, já que estes não possuem o índice numérico. Observe no exemplo que usamos uma estrutura **foreach** que significa **"para cada".** Dessa forma, percorremos o array e pegamos seu nome de associação para poder exibir seu conteúdo. Observe na Figura 44 um exemplo. Estudaremos a estrutura **foreach** mais adiante.

```php
<?php
$telefones = array("Luis"=>"5555-3765", "Andreia"=>"5555-6537", "Marcos"=>"5555-6753");

foreach($telefones as $x => $valor) {
  echo "Associacao=" . $x . ", Value=" . $valor; // $x recebe a associação a cada repetição
  echo "<br>";                                  // $valor recebe o valor da associaçao
}
?>
```

Figura 44 – Exemplo de loop para arrays associativos.
Fonte: Próprio autor (2016).
Descrição: Código que mostra a utilização de uma estrutura de repetição foreach para exibição de um array associativo.

Este é um dos exemplos mais importantes. Reproduza até entendê-lo perfeitamente.

2.10 Estruturas de Repetição

Quando programamos em PHP precisamos repetir o mesmo código para que ele seja executado várias vezes. As estruturas de repetição foram criadas para fazer este serviço para nós, sem que precisemos repetir o código. Em PHP temos quatro estruturas de repetição:

- **while** - repete o bloco de código enquanto uma condição especificada for verdadeira;
- **do...while** - repete o bloco de código uma vez, e depois repete novamente enquanto a condição especificada for verdadeira;
- **for** - repete o bloco de código um determinado número de vezes;
- **foreach** - repete o bloco de código para cada elemento de um array.

2.10.1 While

Repete o bloco de código enquanto uma condição especificada for verdadeira. Veja a sintaxe abaixo.

```
while (condição)  {
      bloco de comandos;
}
```

O exemplo da Figura 45 define a variável $i inicialmente com o valor de 1 ($i = 1;). Em seguida, o loop **while** continuará a ser executado enquanto i for menor ou igual a 5 ($i <= 5). A variável i vai aumentar em 1 a cada vez que o loop é executado ($i++;).

```
<html>
<body>

<?php
$i=1; // $i começa com o valor de 1
while($i<=5) {  // repete enquanto for menor que ou igual a 5
  echo "The number is " . $i . "<br>";
  $i++; // soma 1 ao valor de $i
}
?>

</body>
</html>
```

Figura 45 – Exemplo de um loop while.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de utilização da estrutura de repetição while

Eis um desafio. Uma tabela HTML pode possuir várias linhas <TR> e várias colunas <TD>. Utilize uma estrutura WHILE dentro de outra para fazer uma tabela de 100 linhas por 10 colunas.

2.10.2 Do...While

Esta estrutura repete o bloco de código uma vez, e depois repete novamente enquanto a condição especificada for verdadeira. Veja a sintaxe abaixo.

```
do {
        bloco de comandos;
} while (condição);
```

O exemplo da Figura 46 define a variável $i inicialmente com o valor de 1 ($i = 1;). Então, ele começa o **do...while**. O ciclo irá somar a variável i com 1, e em seguida, escrever alguma saída. Em seguida, a condição é verificada (i é menor ou igual a 5), se for o bloco de código é repetido. Se não for, a repetição é interrompida e a próxima linha depois do **do...while** é lida.

```
<html>
<body>

<?php
$i=1; // $i inicia com o valor de 1
do { // executa o bloco
  $i++;
  echo "The number is " . $i . "<br>";
} while ($i<=5); // Se $i for menor ou igual a 5, repete novamente
?>

</body>
</html>
```

Figura 46 – Exemplo de repetição de código com a estrutura do...while.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de utilização da estrutura de repetição do...while.

O exercício anterior foi difícil? Este agora deve ser mais fácil. Faça o exercício anterior utilizando do...while, ao invés de while.

2.10.3 For

Esta estrutura de repetição é usada quando se sabe com antecedência o número de repetições. Veja a sintaxe abaixo.

```
for (inicial; condição; incremento/decremento)  {
    bloco de código;
}
```

Na sintaxe acima, inicial define uma variável que será o nosso contador. Condição é uma expressão para interromper a repetição. E incremento/decremento é usado para alterar o valor do contador.

O exemplo da Figura 47 define um ciclo que começa com $i = 1. A repetição continuará a funcionar enquanto a variável $i for menor que, ou igual a 5. A variável $i vai aumentar em 1 a cada vez que o loop é executado.

```php
<html>
<body>

<?php
for ($i=1; $i<=5; $i++) {
  echo "O número é " . $i . "<br>";
}
?>

</body>
</html>
```

Figura 47 – Exemplo da estrutura de repetição for.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de utilização da estrutura de repetição for.

Modifique o exercício anterior para utilizar for. Lembre-se de que você vai precisar de um for dentro de outro para conseguir.

2.10.4 Foreach

É usado para percorrer arrays associativas. Veja a sintaxe abaixo.

```
foreach ($array as $valor)   {
      bloco de código;
}
```

Para cada iteração do loop, o valor do elemento da matriz corrente é atribuído a $valor (e o ponteiro do array é movido por um) - assim na próxima iteração do loop, você estará olhando para o próximo valor do array. Dessa forma, sempre passando adiante enquanto houver.

O exemplo da Figura 48 demonstra seu uso.

```
<html>
<body>

<?php
$x = array("um", "dois", "três");
foreach ($x as $valor) {
  echo $valor . "<br>";
}
?>

</body>
</html>
```

Figura 48 – Exemplo de repetição com a estrutura foreach.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de utilização da estrutura de repetição foreach.

Repita este exemplo no seu computador para entendê-lo melhor.

3.Competência 03 | Fundamentos Avançados da Linguagem PHP

Aprendemos os fundamentos básicos da linguagem. Caso você tenha dúvida em alguma parte da competência passada, releia o caderno de estudos, faça alguns testes e pratique mais. Você também pode encontrar mais informações buscando na internet. É de vital importância que você entenda bem todos os pontos apresentados anteriormente para poder continuar.

Então, vamos começar com as Funções em PHP. O verdadeiro poder do PHP está em suas funções. Existem mais de 700 funções já construídas e podemos fazer as nossas próprias funções personalizadas.

Você pode encontrar toda a lista de funções PHP já construídas neste endereço:
http://php.net/manual/pt_BR/indexes.functions.php

3.1 Funções

Já utilizamos funções na competência passada. As funções de string strlen( ), que calculam o total de caracteres da string, e strpos( ), que retornam à posição de uma sequência de caracteres, são funções prontas do PHP. Nesta sessão vamos mostrar como criar suas próprias funções.

Funções são pequenos programas que ajudam sua aplicação em uma tarefa. Vamos supor que você tenha uma aplicação que calcula uma nota fiscal. Mas para dar o total da nota você tem que fazer outro cálculo, o do frete. Este cálculo do frete não faz parte do cálculo da nota fiscal, porque ele pode ter ou não. Além disso, o usuário pode tentar outros lugares de entrega para baratear o frete. Podemos criar uma função que faça só este cálculo.

Separar as responsabilidades é muito importante. Se o cálculo do frete é em outro lugar, estando separado da aplicação principal, isto simplifica para acharmos o código se algum problema acontecer. Outra vantagem de utilizarmos funções é que podemos chamá-la várias vezes em nossa página PHP. Assim, se precisarmos modificar o código desta função só será necessário fazer a modificação em um lugar e não em vários.

Usando funções, o código fica mais simples e de fácil manutenção. Podemos até utilizar funções de outras pessoas, sem nem mesmo saber como foi programado.

Mas como são criadas as funções e como podemos utilizá-las?

3.1.1 Criando Funções PHP

Uma função precisa de um nome e de um bloco de código que será executado quando chamamos a

função pelo seu nome. Observe sua sintaxe básica:

```
function nomeDaFuncao() {
        bloco de código;
}
```

Você deve ter os seguintes cuidados quando for nomear uma função:

- Coloque um nome que reflete o que ela faz;
- Nomes de funções podem começar com letras ou sublinhados, nunca por números.
- Evite utilizar caracteres especiais como cedilhas e letras acentuadas.

Na Figura 49 temos um exemplo com uma função e a chamada desta função em dois lugares diferente, na tag <title> e na tag <h1>. Esta função escreve a frase “Oi mundo!” quando é chamada. Na mesma figura temos o resultado no navegador. Como o PHP é interpretado linha após linha, a função deve ser criada antes de ser chamada.

Figura 49 – Exemplo de uma função e chamada de função.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de criação e utilização de uma função e exibição de seu resultado no navegador.

3.1.2 Adicionando Parâmetros

Quando utilizamos funções para automatizar alguma tarefa, podemos precisar de parâmetros. Parâmetros são como variáveis que recebem um valor quando a função é executada. Normalmente, esses parâmetros são necessários para a realização da tarefa.

Por exemplo, uma função que soma dois números. Para poder somar, ela vai precisar dos dois números que são os parâmetros.

No exemplo da Figura 50, a função recebe dois parâmetros, $nome e $sobrenome, para poder ser executado. Colocamos os valores na ordem, na chamada da função. Observe que chamamos duas vezes a função com valores diferentes.

```
<html>
<body>

<?php
// para trabalho a função precisa receber dois valores
// o primeiro vai para $nome, o segundo para $sobrenome
function apresentacao($nome, $sobrenome) {
    echo "Oi, " . $nome . " " . $sobrenome;
}


apresentacao("Fulano", "de Tal"); // na chamada da função passamos os valores.
apresentacao("Beltrano", "de Tal"); // observe a ordem, é importante.
?>

</body>
</html>
```

Figura 50 – Exemplo de passagem de parâmetros para funções.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de criação e utilização de mais de uma vez de uma função com valores diferentes sendo passados nos parâmetros.

Pode parecer simples a utilização de funções, mas antes de prosseguir, escreva e testes os exemplos apresentados, para melhor fixação. A próxima competência utilizará muito do que já foi visto, principalmente funções.

3.1.3 Retornando Valores

Utilizamos exemplos anteriormente em que a tarefa não dá um resultado. Mas no exemplo da soma de dois algarismos temos um resultado.

Podemos retornar um valor computado pela função utilizando a palavra-chave **return**. Este comando retorna o que estiver à direita e encerra a função.

A Figura 51 mostra um exemplo de retorno de função. A função calcula a soma de dois números, $x e $y. Ela deve escrever uma página com **“2 + 2 = 4”**.

```php
<html>
<body>

<?php
function soma($x, $y) {
    $total=$x+$y;
    return $total; // aqui o valor de $total é retornado
}

echo "2 + 2 = " . soma(2, 2); // soma(2, 2) é substituido pelo retorno da função
?>

</body>
</html>
```

Figura 51 – Exemplo de retorno de uma função.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de uma função que recebe dois valores, soma e retorna o resultado.

Agora, vamos exercitar o que vimos sobre funções. Já temos a operação de soma, faça as operações restantes (subtração, multiplicação e divisão). Não se esqueça de testar.

3.2 Formulários

Até agora não recebemos nenhuma informação do usuário nos exemplos anteriores. Utilizamos os dados que colocamos em variáveis. Para poder receber informações do lado do cliente, ou seja, do usuário, utilizamos hiperlinks ou formulários.

Recebemos informação do usuário quando ele clica em um hiperlink, chamado de método **GET**, ou quando nos envia um formulário, chamado de método **POST**. Então, vamos ver estas duas abordagens, suas vantagens e desvantagens.

3.2.1 Método GET

Este método foi uma das primeiras formas de comunicação cliente/servidor via web. Ele utiliza a URL do site para enviar dados.

Como vantagem, não precisa de um formulário HTML, pode ser digitado diretamente no hiperlink. No entanto, é visível para todos. Digitar senhas utilizando o método GET é inviável.

Como é realizado o envio de dados por GET: **www.meusite.com.br/index. php? nome= fulano&idade=30**

Depois da interrogação, são enviadas duas variáveis com seus respectivos valores: **nome = fulano**, **idade = 30**. O **&** separa uma variável da outra.

Dessa forma, podemos passar em um link alguns dados, mas existe um limite de 2000 caracteres neste método.

Para receber o dado no aplicativo PHP pelo método GET, temos uma variável chamada $_GET. Tudo que é enviado por este método se encontra nesta variável. Seguindo o exemplo de envio acima, podemos receber o nome da seguinte forma.

$_GET[“nome”]

Na Figura 52 temos uma página HTML que mostra um hiperlink para uma página PHP enviando dados por GET. Na Figura 53 a página PHP recebe os dados e escreve uma página de resposta dando um **Oi!** para o usuário.

```
<html>
    <body>
        <a href="get.php?nome=Fulano">Enviar nome Fulano</a>
    </body>
</html>
```

Figura 52 – Página HTML, enviar.html, que envia dados pelo método GET.
Fonte: Próprio autor (2016).
Descrição: Código HTML com um link que passa o valor ‘Fulano’ na variável nome e passa pelo método GET.

```
<html>
<body>

<?php
$usuario = $_GET["nome"]; //o parâmetro deve ser igual ao enviado.

echo "<h1>Oi! " . $usuario . "</h1>";
?>

</body>
</html>
```

Figura 53 – Página get.php que recebe os dados enviados e processa.
Fonte: Próprio autor (2016).
Descrição: Código de exemplo de recepção da variável nome pelo método GET no PHP.

Na Figura 54 vemos o resultado do algoritmo e o endereço com o nome da variável e o seu valor.

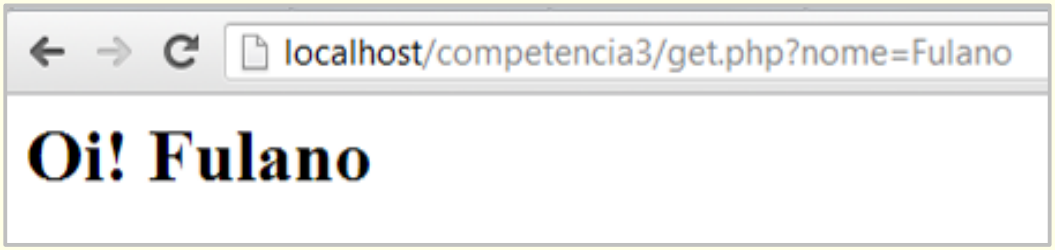


Figura 54 – Resultado no navegador. O endereço da página contém o nome.
Fonte: Próprio autor (2016).
Descrição: Captura de tela do navegador com o resultado do código da Figura 51.

3.2.2 Método POST

O método POST utiliza um formulário HTML para enviar a informação. A maioria dos formulários que você utiliza na web, incluindo aqueles do AVA, utiliza este método. As informações passadas por ele são invisíveis ao usuário e ele tem uma capacidade virtualmente infinita de envio. Seu limite é definido pela configuração do PHP no servidor, que por padrão é de 8Mb. No entanto, pode ser mais ou menos.

A forma de receber os dados é muito parecida com o método anterior, só que utilizamos $_POST[variável] para receber os dados.

Veja o exemplo na Figura 55. Nela temos um formulário HTML comum, que envia os dados pelo método POST para uma página PHP, Figura 56. O resultado é definido na Figura 57. Observe na URL que os dados são enviados de forma invisível.

```
<html>
    <body>
        <form action="post.php" method="post">
            Nome: <input type="text" name="nome">
            Idade: <input type="text" name="idade">
            <input type="submit" name="Enviar">
        </form>
    </body>
</html>
```

Figura 55 – Exemplo de um formulário utilizando o método POST.
Fonte: Próprio autor (2016).
Descrição: Código HTML com um formulário configurado para envido de dados pelo método POST.

```php
<html>
<body>

<?php
$usuario = $_POST["nome"]; //o parâmetro deve ser igual ao enviado.
$tempo = $_POST["idade"];

echo "<p>Oi! " . $usuario . "<br />";
echo "Voce tem " . $tempo . " anos.</p>";
?>

</body>
</html>
```

Figura 56 – Exemplo de uma página PHP que recebe dados de um formulário HTML pelo método POST.
Fonte: Próprio autor (2016).
Descrição: Código PHP de recepção dos dados do formulário da Figura 53 através do método POST

Figura 57 – Saída da página PHP. Observe a URL, não há dados.
Fonte: Próprio autor (2016).
Descrição: Captura de tela do navegador exibindo o resultado do código da Figura 54.

Observe na Figura 54 que na tag <form> temos a opção method. Esta opção seleciona se você quer utilizar o método GET no envio dos dados do formulário. No entanto, será totalmente visível o envio e teremos a limitação de caracteres.

3.2.3 Variável $_REQUEST

Os desenvolvedores do PHP implantaram a variável $_REQUEST para os casos onde não se sabe o método de envio utilizado. Esta variável dá acesso às variáveis dos dois formulários, tanto GET quanto POST.

Ela é usada da mesma forma que $_GET e $_POST só que escrevemos $_REQUEST no lugar. Veja o exemplo anterior utilizando $_REQUEST na Figura 58.

```
<html>
<body>

<?php
$usuario = $_REQUEST["nome"];
$tempo = $_REQUEST["idade"];

echo "<p>Oi! " . $usuario . "<br />";
echo "Voce tem " . $tempo . " anos.</p>";
?>

</body>
</html>
```

Figura 58 – Utilização da variável predefinida $_REQUEST.
Fonte: Próprio autor (2016).
Descrição: Código PHP de recepção de dados sem diferenciação de método.

Muito da comunicação entre o usuário e nossa aplicação será realizada por variáveis, por isso, é muito importante que você refaça os exemplos dados com bastante atenção. Depois, para exercitar, faça uma página HTML que envie sua cor preferida por um hiperlink utilizando o método GET. Quando terminar, construa uma página com um formulário que pede para o usuário digitar sua cor preferida enviando pelo método POST, e exiba uma mensagem dizendo qual foi a cor.

3.3 Include e Require

PHP nos dá duas formas de inserir o conteúdo de outros arquivos PHP antes que o interpretador execute nosso código.

As duas formas de inclusão são feitas através das palavras-chaves **include** e **require**. A única diferença entre eles está na forma como lidam com erros quando o arquivo que será acrescido não é encontrado.

No caso do **require** o erro será letal e a interpretação do compilador irá parar a execução no ponto do erro, enviando o código e mensagem de erro.

No caso do **include**, a interpretação não para e é emitido o aviso de erro no local em que acontece.

Você pode fazer seu sistema em PHP sem utilizar includes, mas que eles ajudam muito, ajudam. Como vamos utilizar includes em nosso sistema de locadora, vamos exercitar. Faça uma página PHP que contenha um título em <h1>. Faça outra página que inclua este código utilizando include. Caso tenha curiosidade, faça um erro proposital, colocando o nome do arquivo errado, e veja como se comporta o interpretador PHP quando você usa include e require. Veja o código HTML retornado para saber se ele terminou de ser escrito.

Qual deles você utilizará depende da importância do que está sendo acrescentado. Se o arquivo é essencial para a aplicação, então, utilizar o require pode impedir que sejam expostas partes importantes do sistema. Mas caso o que será incluído no arquivo não seja tão relevante, utilizar o include garante que o restante do script será executado normalmente.

A sintaxe para incluir os arquivos é:

include 'nomedoarquivo.php';
require 'nomedoarquivo.php';

Includes são utilizados frequentemente. Uma das formas mais comuns é para poupar tempo na construção de páginas HTML. Normalmente, o código HTML do cabeçalho e do rodapé repete-se muito nas diversas páginas do site. Então, podemos colocar o cabeçalho em uma página PHP e o rodapé em outra, e fazer os includes desses arquivos sempre que precisarmos. Outra vantagem desta metodologia é que se precisarmos alterar algo nessas partes, só faremos uma vez. Outra utilização é a colocação de funções genéricas em um arquivo util.php. Assim, sempre que for preciso, o incluímos no começo do arquivo.

Na Figura 59 temos uma página PHP que escreve um menu. Na Figura 60 temos outra página que faz o include do código do menu.

```
<?php
echo '<a href="/default.php">Início</a>
<a href="/tutorials.php">Contato</a>
<a href="/references.php">Sobre</a>';
?>
```

Figura 59 – Página menu.php.
Fonte: Próprio autor (2016).
Descrição: Código que gera links de um menu.

```
<html>
<body>

<nav>
<?php include "menu.php"; ?>
</nav>

<h1>Bem vido!.</h1>
<p>Texto do site...</p>

</body>
</html>
```

Figura 60 – Página que faz o include do conteúdo do arquivo menu.php.
Fonte: Próprio autor (2016).
Descrição: Código que insere o código de menu da figura 57 em outra página.

3.4 Sessões PHP

Uma sessão PHP é uma área da memória que todas as páginas de sua aplicação compartilham. Ela é útil para armazenar informações do usuário e/ou estado da aplicação. Já que todas as variáveis são apagadas com o fim do algoritmo, na sessão temos um lugar seguro para guardar variáveis quando o usuário pula de uma página para outra do nosso site.

Sessões são importantes para autenticação do usuário. Será que você já consegue misturar o que aprendeu? Vamos ver? Faça um formulário que envie um login e uma senha e guarda essas informações em uma sessão. Então, mostre um 'Bom dia, ' com o nome do usuário em uma página, caso seja digitado corretamente. Assim, utilizaremos o método POST, guardaremos em uma sessão as informações e podemos utilizar o include em toda página que quisermos manter em segredo.

Por exemplo, quando você vai fazer uma compra em uma loja virtual e digita seu login e sua senha. O aplicativo da loja abre uma sessão para você e registra algumas informações nela como nome do usuário e o seus produtos no carrinho, dessa forma, não importa o quanto você navegue nas páginas da loja, a aplicação vai continuar sabendo que é você até sair da loja, quando tudo na sessão é apagado.

Para utilizar uma sessão PHP, primeiramente temos que iniciá-la chamando a função **session_start().** A Figura 61 mostra como fazer isto.

```
<?php
session_start();
?>

<html>
<body>

</body>
</html>
```

Figura 61 – Inicializando uma sessão PHP.
Fonte: Próprio autor (2016).
Descrição: Código que inicializa uma sessão PHP através da função sessionn_start().

No exemplo da Figura 62 salvamos um valor na sessão através da variável predefinida **$_SESSION['nomedavariavel']** e a usamos da mesma forma como uma variável comum.

```php
<?php
session_start();
$_SESSION['acesso'] = 1; // Guardamos a variável 'acesso' com o valor de 1
?>

<html>
<body>

<?php
echo "<p>Visualizações=". $_SESSION['acesso'] . "</p>"; // Aqui usamos a variável acesso
?>

</body>
</html>
```

Figura 62 – Salvando variáveis na sessão e as usando.
Fonte: Próprio autor (2016).
Descrição: Código de acrescenta uma variável a sessão com um valor e recupera o valor desta variável.

Caso queira retirar uma variável da sessão, utilize a função **unset()** da seguinte forma:

```
unset($_SESSION['nomedavariavel'])
```

Uma sessão é destruída quando o usuário deixa o site. Mas caso você queira destruir uma sessão antes, por exemplo, por um botão de Sair, pode chamar a função **session_destroy()** para isso.

Como nosso último exercício desta competência faça um algoritmo em PHP que crie uma variável comum chamada nome e coloque seu nome nela. Verifique se ela existe. Caso exista, imprima seu conteúdo.

3.5 Função ISSET()

Perceba que nosso contador de páginas do exemplo anterior só conta uma página, o que não é muito bom. Vamos melhorar aprendendo uma função muito útil.

A função **isset()** verifica se a variável consta na memória. Se existir, retorna verdadeiro, se não, falso.

Vamos melhorar nosso contador de páginas utilizando esta função da forma como está na Figura 63. Se a variável estiver na sessão, ela soma um, caso não estiver, coloca na sessão com o valor de um.

```php
<?php
session_start();
if(isset($_SESSION['acesso']))
    $_SESSION['acesso'] = $_SESSION['acesso'] + 1; // Se a variável estive na sessão, soma 1.
else
    $_SESSION['acesso'] = 1; // Senão, inicia ela com o valor de 1

?>

<html>
<body>

<?php
echo "<p>Visualizações=". $_SESSION['acesso'] . "</p>";
?>

</body>
</html>
```

Figura 63 – Utilizando a função isset() para saber se uma variável existe na memória.
Fonte: Próprio autor (2016).
Descrição: Código que verifica se existe uma variável na sessão, se não existir cria, se existir soma mais um.

Veja em seu navegador a página. Cada vez que você atualiza o número de visualizações é acrescentado um às visualizações.

Você deve ler e reler o conteúdo deste caderno para melhorar seu entendimento nos assuntos abordados. Na internet existem tutoriais que explicam de outra forma esses assuntos. O AVA disponibiliza o fórum para você postar suas dúvidas. Fazer os exemplos e exercícios é essencial para o aprendizado. E só para lembrar que a maioria dos erros quando se está aprendendo são de digitação, então, preste bem atenção ao que está fazendo.

4.Competência 04 | Projeto: Incluindo, Alterando, Exibindo e Excluindo Informações

Se você leu com atenção o caderno de estudo, fez tanto os exemplos quanto os exercícios e não ficou com dúvida, então está preparado para iniciar esta nova competência. Vamos usar o que aprendemos até agora para construir um sistema da Universo Digital, para catalogar os corpos celestes do Universo visível. Mas não é só utilizar o que foi visto, nosso sistema necessita da persistência dos dados em um banco de dados. Para isso, aprendemos em outra disciplina como construir e manusear um banco de dados. Dessa forma, não iremos abordar esse assunto nesta competência. Outro assunto que não abordaremos também é a marcação HTML e o estilismo em CSS. Você também precisa já ter algum conhecimento sobre isto. Para que não tenhamos que perder tempo será fornecido o HTML e CSS prontos. O que aprenderemos, então, será focado em como o PHP se comunica com o banco de dados MySQL para incluir, alterar, excluir e recuperar dados.

Primeiramente você deve baixar no AVA o projeto base já iniciado, com o HTML e CSS já construídos. A partir dele é que montaremos nosso sistema.

Crie um banco de dados chamado **“universo”**, que deve possuir uma tabela chamada **“planetas”**, com um campo **"id"**, int(6) com auto incremento; um campo **"nome"**, varchar(255); um campo **"tipo"**, varchar(10); um campo **"tamanho"** do tipo float; e, por fim, um campo **"descricao"** do tipo text. Respeite os nomes dados e seus tipos para não aparecer erros desnecessários. Veja a Figura 64.

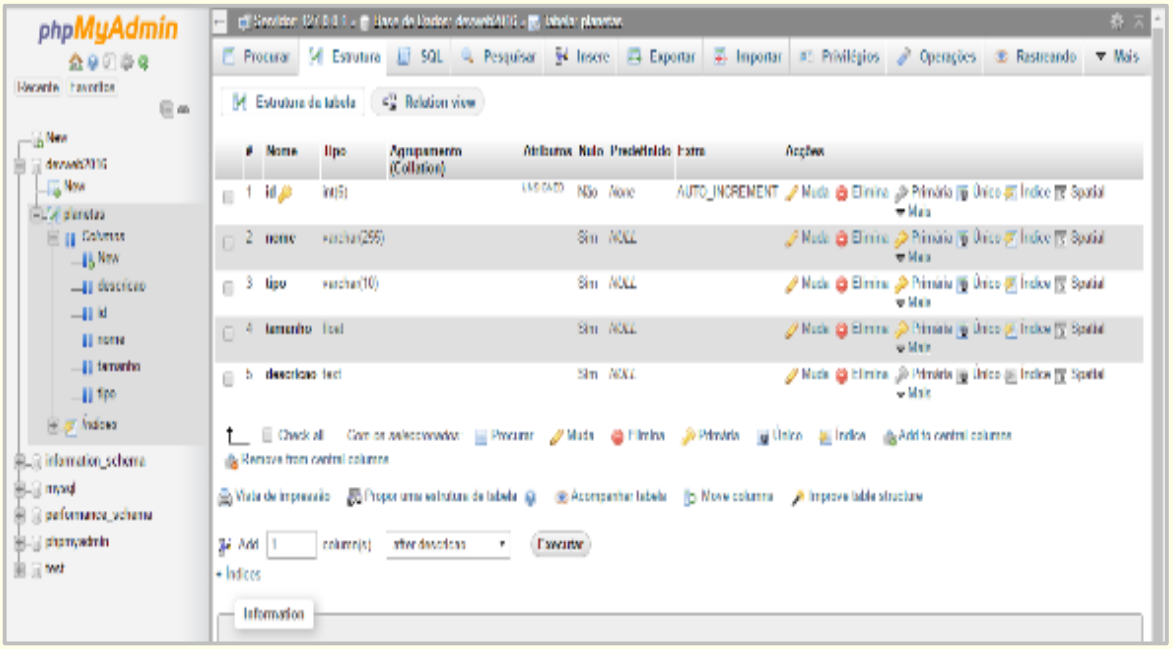


Figura 64 – phpMyAdmin exibindo os campos com seus tipos da tabela “planetas”.
Fonte: Próprio autor (2016).
Descrição: Exibição pelo phpMyAdmin do banco de dados planetas, com seus cinco campos: id, nome, tipo, tamanho e descrição.

Crie uma pasta **"du"** na pasta **htdocs** de seu XAMPP, e descompacte os arquivos do projeto base, Figura 65. Assim, você acessará o modelo pela url **"localhost/du"**. Coloque no navegar e veja.

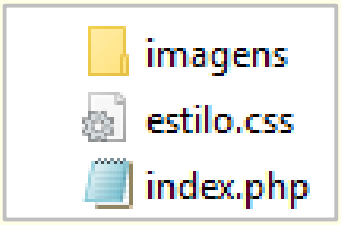


Figura 65 – Arquivos do projeto base.
Fonte: Próprio autor (2016).
Descrição: Árvore de arquivos e pastas do projeto base: pasta imagens com as imagens, arquivo estilo.css com o estilo de apresentação e index.php com o código HTML inicial do projeto.

O projeto é composto pelas seguintes pastas e arquivos, Figura 65:

- **images**: pasta que guarda as imagens do modelo;
- **estilo.css**: configurações de exibição em CSS do modelo;
- **index.php**: página inicial do sistema Universo Digital.

Pronto! Agora que tudo está no lugar podemos começar. Observe que o modelo não possui nenhuma programação em PHP. Nós é que vamos escrevê-las. Vamos começar?

4.1 Templates

Nosso sistema terá várias páginas PHP, cada uma terá uma responsabilidade, como incluir planetas, incluir estrelas, listar sistemas, etc. Teremos, então, que repetir muito código HTML. Se vamos repetir muito do mesmo código, vamos utilizar **include** (sessão 4.3) para diminuir nosso trabalho e nos ajudar.

Abra o arquivo **index.php** e recorte o conteúdo da primeira linha até a primeira tag ** **. Agora crie um novo arquivo PHP (sessão 2.5) com o nome de **template.cabecalho.php**. Cole as marcações HTML neste arquivo. Não se preocupe que o arquivo tenha a extensão php, mas dentro seja apenas HTML. O PHP é esperto o suficiente para saber a diferença. A Figura 66 mostra o que deve ser recortado.

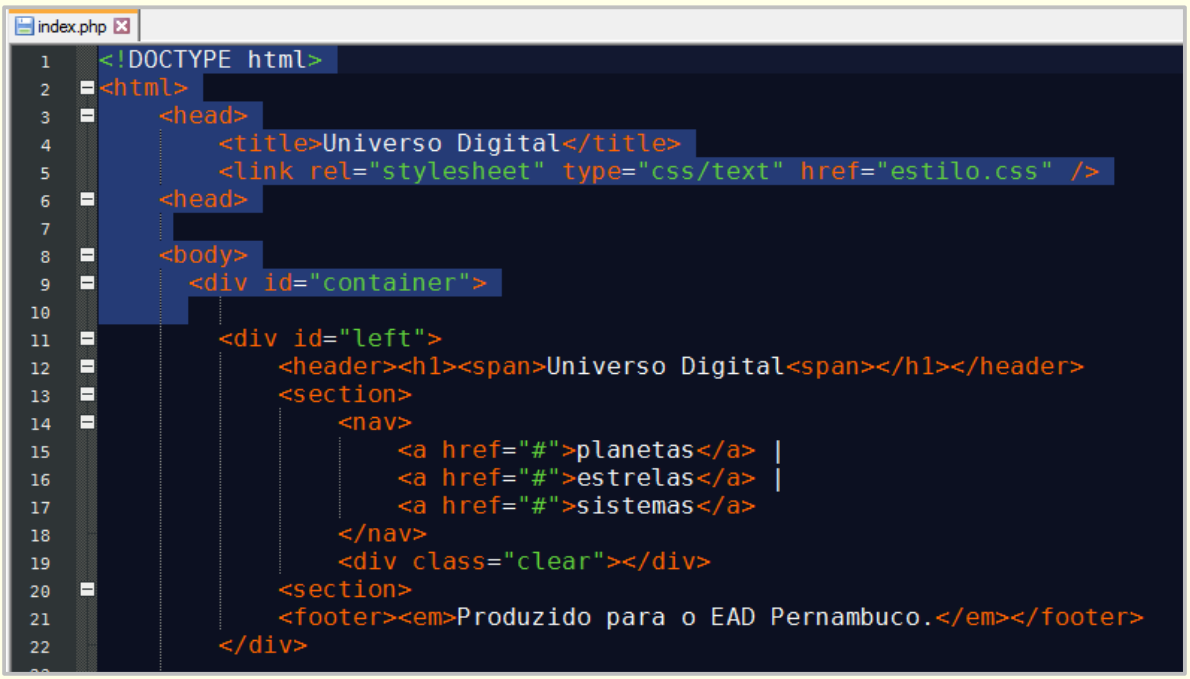


Figura 66 – Página index.php, código que deve ser retirado e colado na página template.cabecalho.php.
Fonte: Próprio autor (2016).
Descrição: Na imagem temos o código destacada que deve ser copiado e colado. São as 10 primeiras linhas que estão em destaque.

Agora crie a página **template.rodape.php**. Copie o final da página HTML de base em **index.php** para este arquivo. Você deve copiar as três últimas linhas. Começando pela tag que fecha a div container. Não se esqueça de salvar todos os arquivos.

Temos agora o modelo de começo e o final de todas nossas páginas. Vamos fazer uma página para testar e servir de base para as próximas.

Na página index.php acrescente o código PHP para incluir o conteúdo dos arquivos recém criados. No começo do arquivo apague o código que você copiou, faça um include (sessão 4.3) no início com o arquivo **template.cabecalho.php** e no final **template.rodape.php**. Agora toda vez que você precisar de uma nova página basta fazer o include inicial e final. O código no meio será o conteúdo do site.

```php
<?php include 'template.cabecalho.php'; ?>

        <div id="left">
            <header><h1><span>Universo Digital<span></h1></header>
            <section>
                <nav>
                    <a href="#">planetas</a> |
                    <a href="#">estrelas</a> |
                    <a href="#">sistemas</a>
                </nav>
                <div class="clear"></div>
            <section>
            <footer><em>Produzido para o EAD Pernambuco.</em></footer>
        </div>

<?php include 'template.rodape.php'; ?>
```

Figura 67 – Página index.php.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra como deve ficar a página index.php depois que o código inicial e final do arquivo for removido para seus devidos arquivos.

Vamos melhorar esta index. Na **index.php** recorte todas as linhas da ** **, deixaremos a index.php apenas com o menu. Assim, a index apenas mostrará apenas a marca e o menu. Salve.

Agora vamos salvar este arquivo com outro nome. Salve como **"planetas.form.php"** e cole o texto que você havia recortado no lugar onde estava. Salve novamente.

Agora tempos uma index mais interessante e uma página **"planetas.form.php"**, que será nosso formulário para incluir e alterar planetas. Se quiser fazer os exercícios propostos, crie a página "estrelas.form.php". Seria muito bom.

No link para "planetas" coloque o valor de href para **"planetas.form.php"**. Tanto na index.php quanto no menu de planetas.form.php. Observe como ficou o código da página na Figura 68. Nele há um código em destaque para ser acrescentado também. É um submenu.

```php
<?php
include 'template.cabecalho.php';
?>

        <div id="left">
            <header><h1><span>Universo Digital<span></h1></header>
            <section>
                <nav>
                    <a href="planetas.form.php">planetas</a> |
                    <a href="#">estrelas</a> |
                    <a href="#">sistemas</a>
                </nav>
                <br />
                <nav>
                    <a href="planetas.form.php">inserir planetas</a> |
                    <a href="planetas.listar.php">listar planetas</a>
                </nav>
                <div class="clear"></div>
            <section>
            <footer><em>Produzido para o EAD Pernambuco.</em></footer>
        </div>

        <div id="right">
            <header><h2><span>Planetas<span></h2></header>
            <section>
                <p>Um planeta (do grego πλανήτης [planítis]) é um corpo celeste que orbita uma estrela ou um remanescente
                <p>Fonte: <a href="https://pt.wikipedia.org/wiki/Planeta">https://pt.wikipedia.org/wiki/Planeta</a></p>
```

Figura 68 – Página planetas.form.php.
Fonte: Próprio autor (2016).
Descrição: Em destaque na imagem o código a ser inserido, correspondendo as linhas de 14 até a 17.

Teste a página **index.php** e veja se o resultado foi igual ao da Figura 69.

Figura 69 – Visualização da página index.php no navegador.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra como deve ficar a página inicial do site Universo Binário depois das modificações.

4.2 Conectando ao MySQL

Para podermos incluir, excluir, alterar e listar dados de um banco de dados MySQL devemos nos

conectar a ele.

O PHP possui muitas funções úteis e uma delas é a **mysqli_connect()**. Esta função é uma atualização da **mysql_connect()**. Tenha cuidado para não confundir. Ela vai fazer o trabalho de se conectar ao banco. Para isso, temos que passar algumas informações como argumentos: o endereço de acesso ao banco, a senha, o login e o nome do banco.

A conexão com o banco será fechada automaticamente quando o script terminar, mas podemos fechar quando quisermos com a função **mysqli_close()**. Para isso, ele recebe como argumento a conexão que será fechada.

A função **mysqli_connect_errno()** retorna verdadeiro se ocorrer algum erro. Nesse caso, mostramos uma mensagem e encerramos a execução da página.

Você deve ter o cuidado de saber o endereço de seu banco, a senha, o login e o nome do banco. Caso algum deles esteja errado, irá aparecer uma mensagem de erro informando o erro. Verifique também se o XAMPP está ligado com o Apache e o MySQL (sessão 2.3).

Então, vamos abrir uma conexão com nosso banco em um arquivo separado, já que vamos ter que fazer isso em várias páginas. Depois vamos verificar se conectou corretamente ou deu algum erro. A Figura 70 mostra a página PHP **conexao.php**.

index.php | conexao.php | planetas.inserir.php | planetas.listar.php

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$database = "devweb2016";

$conexao = mysqli_connect($servername, $username, $password, $database);

/* CHECK CONNECTION */
if (mysqli_connect_errno()) {
    printf("<p style='color: red;'>Conexão falhou: %s\n</p>", mysqli_connect_error());
    exit();
}

?>
```

Figura 70 – Página conexão.php.
Fonte: Próprio autor (2016).
Descrição: A imagem exibe o código necessário para fazer a conexão com o banco de dados.

Na sessão anterior incluímos outros arquivos PHP no **planetas.form.php**, vamos incluir este também, mas usando **require**. Observe na Figura 71. Teste a página para ver se aparece alguma mensagem de erro. Se não aparecer, então tudo ocorreu bem.

```php
<?php
require 'conexao.php';

include 'template.cabecalho.php';
?>

        <div id="left">
            <header><h1><span>Universo Digital<span></h1></header>
            <section>
                <nav>
                    <a href="planetas.form.php">planetas</a> |
                    <a href="#">estrelas</a> |
                    <a href="#">sistemas</a>
                </nav>
                <br />
                <nav>
                    <a href="planetas.form.php">inserir planetas</a> |
                    <a href="planetas.listar.php">listar planetas</a>
                </nav>
                <div class="clear"></div>
            <section>
            <footer><em>Produzido para o EAD Pernambuco.</em></footer>
        </div>

```

Figura 71 – Página index.php, require do arquivo conexao.php.
Fonte: Próprio autor (2016).
Descrição: A imagem exibe o código index.php com o require do arquivo conexao.php

4.3 Listando Dados

Fazemos ao MySQL pedidos. O pedido é uma string comum que possui comandos MySQL. A função **mysqli_query()** recebe como argumento nossa string com o comando e envia pela conexão aberta. Seu retorno é o resultado da solicitação.

Vamos utilizar esta função para pedir ao MySQL que nos envie uma listagem dos produtos que estão em nosso banco. Como já passamos pelas competências que mostram os comandos SQL, não vamos nos focar nelas. Caso não se lembre, pegue seu caderno de estudos daquela competência e revise para poder continuar.

A página que vamos construir irá exibir o conteúdo da tabela planetas. Temos algumas informações fictícias para podermos testar nossa aplicação. Se você não as tem, invente algumas.

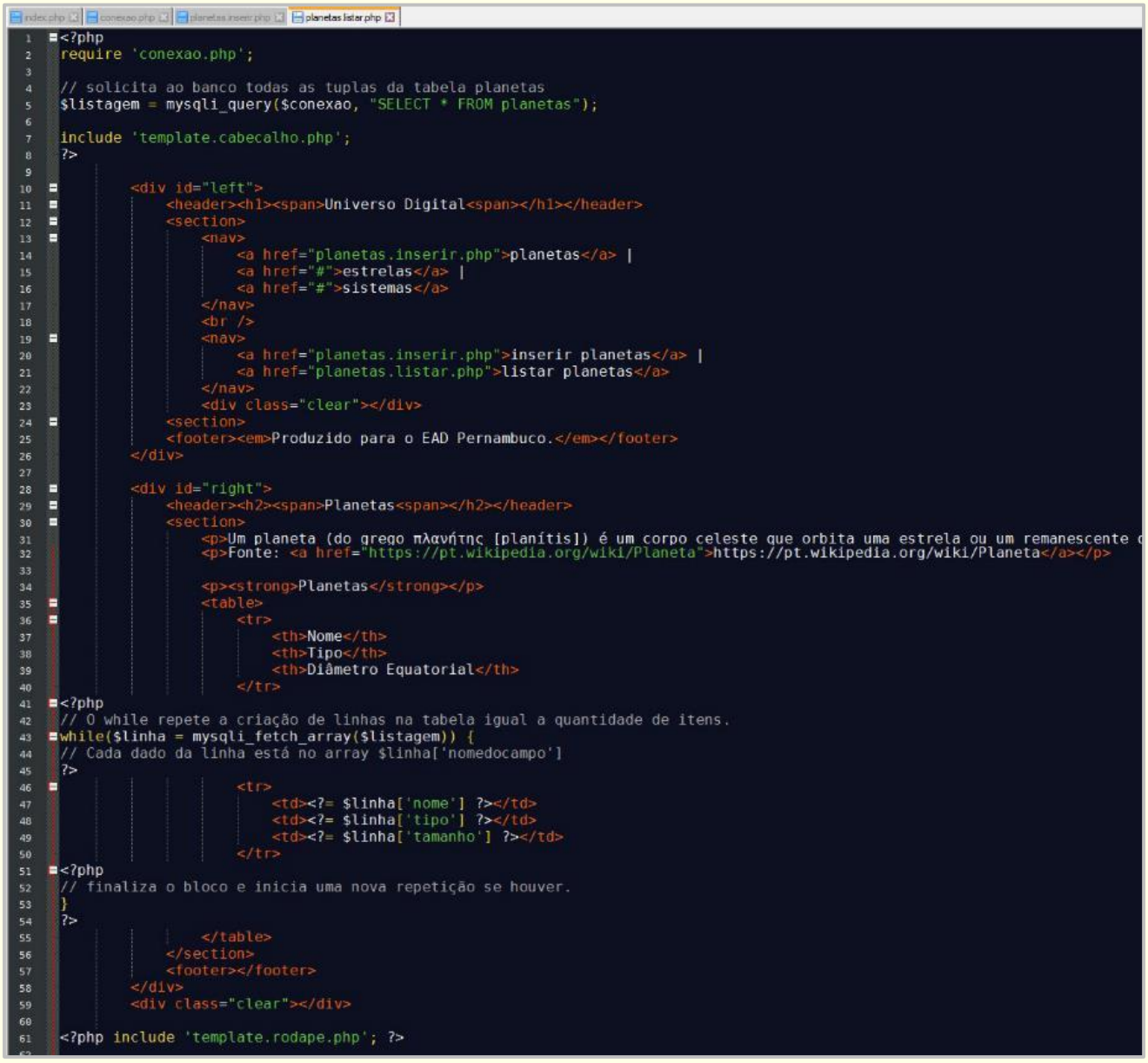
Então, abra nossa página **planetas.form.php** e salve ela com o nome de **planetas.listar.php**. Apague o formulário, vamos apenas listar os dados. A Figura 72 mostra o código para listar os dados. O código está comentado para você saber o que é ordenado em cada passo.

Primeiramente utilizamos **mysqli_query()** para receber a listagem e guardamos na variável **$listagem**. Então, exibimos algumas divs e o início de uma tabela em HTML. Na parte das linhas da tabela utilizamos um **while** para pegar cada linha da listagem, que é armazenada, uma de cada vez, na variável **$linha** como um **array associativo** (sessão 3.9). A chave para cada valor da linha é o

nome do campo. Assim, um campo id na tabela tem o array **$linha[‘nome’]** para pegar seu valor da coluna nome na tabela do banco. Quando termina a iteração no laço **while** a tabela HTML é fechada e é colocado o código do restante da página com um **include**.

Como existem diversas línguas no planeta e nem todas elas utilizam os mesmos caracteres que nós, como o ç e o ~, temos o CHARSET, que é a codificação de caracteres para o documento. Utilizamos a linha mysqli_query($conexao, "SET NAMES 'utf8';"); para dizer ao MySQL que nos envie os dados em um formato que possui os caracteres especiais de nossa língua.

Existe uma forma curta de imprimir o conteúdo das variáveis. Ao invés de <?php echo $linha['titulo'] ?> você pode substituir por esta formatação que utiliza o símbolo de igual (=).
Assim,
<?= $linha[‘titulo’] ?>
Faça a substituição para ver como funciona.



```php
<?php
require 'conexao.php';

// solicita ao banco todas as tuplas da tabela planetas
$listagem = mysqli_query($conexao, "SELECT * FROM planetas");

include 'template.cabecalho.php';
?>

        <div id="left">
            <header><h1><span>Universo Digital<span></h1></header>
            <section>
                <nav>
                    <a href="planetas.inserir.php">planetas</a> |
                    <a href="#">estrelas</a> |
                    <a href="#">sistemas</a>
                </nav>
                <br />
                <nav>
                    <a href="planetas.inserir.php">inserir planetas</a> |
                    <a href="planetas.listar.php">listar planetas</a>
                </nav>
                <div class="clear"></div>
            <section>
            <footer><em>Produzido para o EAD Pernambuco.</em></footer>
        </div>

        <div id="right">
            <header><h2><span>Planetas<span></h2></header>
            <section>
                <p>Um planeta (do grego πλανήτης [planítis]) é um corpo celeste que orbita uma estrela ou um remanescente d
                <p>Fonte: <a href="https://pt.wikipedia.org/wiki/Planeta">https://pt.wikipedia.org/wiki/Planeta</a></p>

                <p><strong>Planetas</strong></p>
                <table>
                    <tr>
                        <th>Nome</th>
                        <th>Tipo</th>
                        <th>Diâmetro Equatorial</th>
                    </tr>
<?php
// O while repete a criação de linhas na tabela igual a quantidade de itens.
while($linha = mysqli_fetch_array($listagem)) {
// Cada dado da linha está no array $linha['nomedocampo']
?>
                    <tr>
                        <td><?= $linha['nome'] ?></td>
                        <td><?= $linha['tipo'] ?></td>
                        <td><?= $linha['tamanho'] ?></td>
                    </tr>
<?php
// finaliza o bloco e inicia uma nova repetição se houver.
}
?>
                </table>
            </section>
            <footer></footer>
        </div>
        <div class="clear"></div>

<?php include 'template.rodape.php'; ?>
```

Figura 72 – Página index.php com o código para listar.
Fonte: Próprio autor (2016).
Descrição: A imagem foi montada para mostrar o início do código e seu final.

A Figura 73 mostra o resultado no navegador.

Figura 73 – Exibição do resultado no navegador.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra o resultado no navegador depois das modificações para recuperação no banco de dados.

Sempre falamos da importância da produção dos exemplos e das atividades para o aprendizado. Copiar é muito simples, mas será que você entendeu os conceitos apresentados? Para testar, faça um exercício que é muito parecido com este. Faça a listagem dos dados da tabela 'estrelas'. Observe que os campos são diferentes da tabela 'planetas'.

Este código de exemplo exibiu apenas alguns valores para simplificar o exemplo, mas você pode pegar os outros valores das outras colunas do mesmo modo. É claro que você irá precisar alterar a tabela HTML para possuir mais colunas e exibir mais valores.

Se após uma consulta no banco você colocar as linhas da Figura 74, caso haja alguma coisa de errado, será mostrada uma mensagem na tela que dirá o que aconteceu.

```php
<?php
require 'conexao.php';

// solicita ao banco todas as tuplas da tabela planetas
$listagem = mysqli_query($conexao, "SELECT * FROM planetas");

// Mensagem de erro caso aconteça algo de errado.
if (mysqli_connect_errno()) {
    printf("<p style='color: red;'>Erro: %s\n</p>", mysqli_connect_error());
    exit();
}

include 'template.cabecalho.php';
?>
```

Figura 74 – Código para exibição de mensagem de erro.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra o código para listagem e controle de erros.

4.4 Incluindo Dados

Agora que podemos ver os dados no banco, vamos adicionar dados.

O processo é muito parecido, mas antes teremos que receber dados do usuário. Precisaremos de um formulário que receberá os valores dados pelo usuário.

O modelo já contém um formulário em HTML que vamos modificá-lo para nossas necessidades. Ele vai servir tanto para incluir como para alterar os dados. O formulário em HTML irá enviar os dados pelo método **post** para a página **planetas.inserir.php** que faremos mais a frente, será nele que conectaremos com o banco. Por agora, você deve terminar a tag **<form>,** incluindo os parâmetros action e post, que está na página **planetas.form.php**. Observe o exemplo na Figura 75.

O comando or die() significa que se a função retornar um erro, a execução deve parar imediatamente e o texto do argumento é apresentado.

```
30  <form action="planetas.inserir.php" method="post">
31      <p><strong>Insira um novo planeta no sistema: </strong></p>
32      Nome: <input type="text" name="nome" /><br />
33      Tipo:
34      <select name="tipo">
35          <option value="terrestre">Terrestre</option>
36          <option value="gasoso">Gigante Gasoso</option>
37          <option value="anão">Planeta Anão</option>
38      </select><br />
39      Diâmetro Equatorial: <input type="text" name="tamanho" /><br />
40      Descrição:
41      <textarea name="descricao"></textarea>
42      <input type="submit" value="enviar" />
43  </form>
```

Figura 75 – Página index.php com o formulário HTML, preenchimento da action e do method na tag.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra o código como o formulário para cadastro de planetas em HTML.

Observe que no final redirecionamos para outra página pela função header(). Dessa forma, depois de acrescentar no banco, listamos seu conteúdo para nos certificar de que está lá da forma correta.
Na última linha temos um exit para encerrar a execução daquela página.

Quando o usuário clica no botão de salvar, os dados são enviados para a página **planetas.inserir.php**, faremos ela do início sem reaproveitar código. Crie a página e digite o código da Figura 76. O código possui comentário em texto cinza para ajudar a entender.

Esta página se conecta ao banco e recebe os valores do formulário pela variável **$_POST**. Pegamos com esta variável os valores de cada campo do formulário pelo seu **name**. Em seguida usamos novamente a função **mysqli_query()** para enviar para o MySQL o comando de gravação.

```
4
5   // Colocamos em variáveis os dados enviados pelo formulário.
6   // Cada campo do formulário deve ter um name, ele será usado para
7   $nome = $_POST['nome'];
8   $tipo = $_POST['tipo'];
9   $tamanho = $_POST['tamanho'];
10  $descricao = $_POST['descricao'];
11
12  // Como tamanho corresponde a um valor numérico, não colocamos el
13  $sql = "INSERT INTO planetas (nome, tipo, tamanho, descricao)
14  VALUES ('$nome', '$tipo', $tamanho, '$descricao')";
15
```

Figura 76 – Página planetas.inserir.php.
Fonte: Próprio autor.
Descrição: A imagem exibe o código de recepção dos dados do formulário HTML através do método POST.

Chegamos a uma parte muito importante, o exercício. Agora, você deve usar o que foi visto e implantar o algoritmo para inserir as estrelas. O código é muito parecido com este, mas o formulário e os campos são outros. Preste bastante atenção e pense bem para saber o que vai mudar e o que vai ser igual.

4.5 Excluindo Dados

Para excluir uma linha em nosso banco de dados, antes de tudo, devemos receber qual será esta linha. Perguntar o código ao usuário não seria prático, já que ele teria que decorar o código do produto que gostaria de apagar. Então, vamos acrescentar nossa listagem com dois links para cada linha de dado: um link para excluir e outro para editar os dados.

A lógica é simples, no link do botão vamos enviar pelo método **GET** (sessão 4.2.1) o código da linha que iremos apagar para a página responsável por isto. Depois de apagar no banco, faremos como antes e redirecionaremos para a página de listagem para conferir se tudo foi feito a contento.

Vamos começar?

Primeiro iremos alterar nossa página de listagem um pouco. Preste bastante atenção, porque será necessário repetir o processo para o link editar na próxima sessão. A Figura 77 mostra o que será alterado.

```
39
40                  <p><strong>Planetas</strong></p>
41                  <table>
42                      <tr>
43                          <th>Nome</th>
44                          <th>Tipo</th>
45                          <th>Diâmetro Equatorial</th>
46                          <th></th>
47                      </tr>
48  <?php
49  // O while repete a criação de linhas na tabela igual a quantidade de itens.
50  while($linha = mysqli_fetch_array($listagem)) {
51  // Cada dado da linha está no array $linha['nomedocampo']
52  ?>
53                      <tr>
54                          <td><?= $linha['nome'] ?></td>
55                          <td><?= $linha['tipo'] ?></td>
56                          <td><?= $linha['tamanho'] ?></td>
57                          <td><a href="planetas.excluir.php?id=<?= $linha['id'] ?>">Excluir</a></td>
58                      </tr>
59  <?php
60  // finaliza o bloco e inicia uma nova repetição se houver.
61  }
62  ?>
```

Figura 77 – Página planetas.listar.php, acrescente as linhas 45 e 57.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra o código para exibição das linhas de dados, que utiliza um laço while.

A tag **<a>** no HTML define um hiperlink. No campo **'href'** colocamos o nome da página e, através do método **GET**, completamos a **URL** com a variável **'id'** e concatenamos seu valor com o número da respectiva linha. Veja o resultado na Figura 78, e no canto inferior esquerdo da imagem tem um retângulo azul com o link do primeiro campo, observe o ponteiro do mouse em cima do primeiro link de exclusão.

Figura 78 – Exibição no navegador da nova página de listar com um link excluir ao lado de cada linha.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra a listagem de dados no navegador. Resultado do código anterior.

Agora vamos criar a página **planetas.excluir.php**. Será muito parecido com o inserir que fizemos antes. O algoritmo da página **planetas.excluir.php** está na Figura 79 com comentário explicativo com texto cinza.

```php
<?php
// Conectamos no banco.
require "conexao.php";

// Colocamos na variável id o valor recebido pelo método GET referente ao id.
$id = $_GET['id'];

// Construímos o comando SQL para apagar o campo que possua o id da variável
$sql = "DELETE FROM planetas WHERE id='$id'";

// Enviamos o pedido ao banco e caso aconteça ao de errado escrevemos uma mensagem de erro.
mysqli_query($conexao, $sql) or die("Erro: " . mysqli_error($conexao));

// Se tudo der certo, chegamos até aqui e redirecionamos para a página de listagem.
header('Location: planetas.listar.php');
exit();
?>
```

Figura 79 – Página excluir_filme.php.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra o código que recebe uma variável com o código da linha a ser excluída, pelo método GET, exclui no banco e redireciona para a página de listagem.

A página **planetas.excluir.php** recebe a variável **'id'** pelo método **GET**. Montamos o pedido de exclusão em **SQL** acrescentando o conteúdo da variável **$id**. Dessa forma, o MySQL saberá qual linha excluir e não vai apagar tudo. Enviamos o pedido pela conexão utilizando a função **mysqli_query()** e, se algo der errado, paramos o script e mostramos a mensagem de erro. Depois de tudo feito corretamente, redirecionamos a exibição da página para **planestas.listar.php**.

O resultado será a exibição da listagem sem a linha que foi apagada.

Agora é a sua vez. Faça a exclusão de 'estrelas'.

4.6 Alterando Dados

Assim como excluir um registro, ou linha, no banco, precisamos saber com antecedência qual registro vamos editar. Vamos utilizar a mesma ideia utilizada na listagem para exclusão. Na página **planetas.listar.php** faça a modificação destacada na Figura 80.

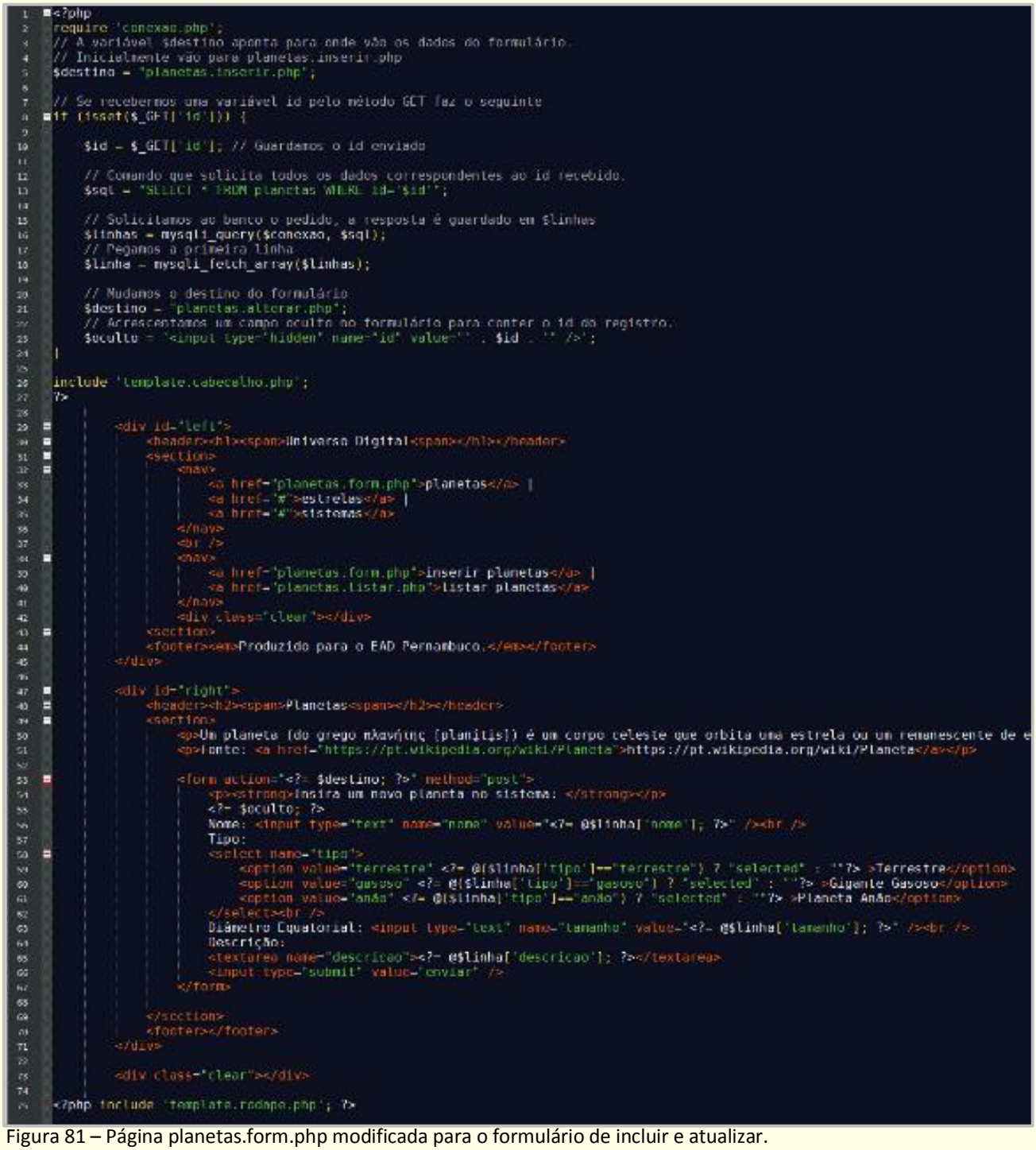
```
                <p><strong>Planetas</strong></p>
                <table>
                    <tr>
                        <th></th>
                        <th>Nome</th>
                        <th>Tipo</th>
                        <th>Diâmetro Equatorial</th>
                        <th></th>
                    </tr>
<?php
// O while repete a criação de linhas na tabela igual a quantidade de itens.
while($linha = mysqli_fetch_array($listagem)) {
// Cada dado da linha está no array $linha['nomedocampo']
?>
                    <tr>
                        <td><a href="planetas.form.php?id=<?= $linha['id'] ?>">Editar</a></td>
                        <td><?= $linha['nome'] ?></td>
                        <td><?= $linha['tipo'] ?></td>
                        <td><?= $linha['tamanho'] ?></td>
                        <td><a href="planetas.excluir.php?id=<?= $linha['id'] ?>">Excluir</a></td>
                    </tr>
<?php
// finaliza o bloco e inicia uma nova repetição se houver.
}
?>
                </table>
```

Figura 80 – Modificação na página planetas.listar.php, estão nas linhas 43 e 55.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra as linhas que devem ser modificadas na listagem para incluir os links de 'Editar' e 'Excluir'.

Dessa forma, quando exibirmos cada item da listagem, ao lado teremos um botão para sua edição. Esse botão envia pelo método **GET** o **'id'** do registro.

Seria bom vermos os dados de um registro já preenchidos em um formulário para facilitar a alteração. Podemos fazer isto diretamente no formulário, que inclui o que está na página **planetas.form.php**. Então, vamos aproveitar a página, enviando a ela o **'id'**.

Abra a página **planetas.form.php** e modifique para exibir os campos para edição já preenchidos. A Figura 81 mostra a página com a nova codificação e comentários em texto cinza. Preste bem atenção, porque temos modificações no começo do arquivo e no meio do formulário HTML.



```php
<?php
require 'conexao.php';
// A variável $destino aponta para onde vão os dados do formulário.
// Inicialmente vão para planetas.inserir.php
$destino = 'planetas.inserir.php';

// Se recebermos uma variável id pelo método GET faz o seguinte
if (isset($_GET['id'])) {

    $id = $_GET['id']; // Guardamos o id enviado

    // Comando que solicita todos os dados correspondentes ao id recebido.
    $sql = "SELECT * FROM planetas WHERE id='$id'";

    // Solicitamos ao banco o pedido, a resposta é guardado em $linhas
    $linhas = mysqli_query($conexao, $sql);
    // Pegamos a primeira linha
    $linha = mysqli_fetch_array($linhas);

    // Mudamos o destino do formulário
    $destino = 'planetas.alterar.php';
    // Acrescentamos um campo oculto no formulário para conter o id do registro.
    $oculto = '<input type="hidden" name="id" value="' . $id . '" />';
}

include 'template.cabecalho.php';
?>

        <div id="left">
            <header><h1><span>Universo Digital<span></h1></header>
            <section>
                <nav>
                    <a href="planetas.form.php">planetas</a> |
                    <a href="#">estrelas</a> |
                    <a href="#">sistemas</a>
                </nav>
                <br />
                <nav>
                    <a href="planetas.form.php">inserir planetas</a> |
                    <a href="planetas.listar.php">listar planetas</a>
                </nav>
                <div class="clear"></div>
            </section>
            <footer><em>Produzido para o EAD Pernambuco.</em></footer>
        </div>

        <div id="right">
            <header><h2><span>Planetas<span></h2></header>
            <section>
                <p>Um planeta (do grego πλανήτης (planētis)) é um corpo celeste que orbita uma estrela ou um remanescente de e
                <p>Fonte: <a href="https://pt.wikipedia.org/wiki/Planeta">https://pt.wikipedia.org/wiki/Planeta</a></p>

                <form action="<?= $destino; ?>" method="post">
                    <p><strong>Insira um novo planeta no sistema: </strong></p>
                    <?= $oculto; ?>
                    Nome: <input type="text" name="nome" value="<?= @$linha['nome']; ?>" /><br />
                    Tipo:
                    <select name="tipo">
                        <option value="terrestre" <?= @($linha['tipo']=="terrestre") ? 'selected' : ''?> >Terrestre</option>
                        <option value="gasoso" <?= @($linha['tipo']=="gasoso") ? 'selected' : ''?> >Gigante Gasoso</option>
                        <option value="anão" <?= @($linha['tipo']=="anão") ? 'selected' : ''?> >Planeta Anão</option>
                    </select><br />
                    Diâmetro Equatorial: <input type="text" name="tamanho" value="<?= @$linha['tamanho']; ?>" /><br />
                    Descrição:
                    <textarea name="descricao"><?= @$linha['descricao']; ?></textarea>
                    <input type="submit" value="enviar" />
                </form>

            </section>
            <footer></footer>
        </div>

        <div class="clear"></div>

<?php include 'template.rodape.php'; ?>
```

Figura 81 – Página planetas.form.php modificada para o formulário de incluir e atualizar.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra o código do formulário temos alterações nas linhas 53, 55, 56, 59, 60, 61, 63 e 65.

Como esta página vai servir tanto para inserir como para alterar os dados no formulário, criamos uma variável **$destino** para guardar onde vão ser enviados os dados do formulário. Inicialmente os dados vão para **planetas.inserir.php**.

Observe que o título do formulário não muda. Sempre permanece com 'Incluir'. Você conseguiria modificar ele quando fosse para alterar? Tente.

Depois verificamos se recebemos pelo método **GET** algum **id**. O uso da função **isset()** verifica isto. Se recebermos, a página servirá para alterar e não incluir. Devemos, então, pegar os dados que serão alterados para mostrar para o usuário. Para isso, utilizamos a conexão com o banco e pegamos os dados da linha que queremos alterar. Guardamos estes dados em **$linhas**, e depois pegamos a primeira linha, pois só ela deve ter o registro, colocando em **$linha**.

Alteramos **$destino** para **planetas.alterar.php** e guardamos em **$oculto** um campo oculto em HTML que contém o id. Dessa forma, a página **planetas.alterar.php** vai saber que registro deve alterar.

Acrescentamos no formulário as variáveis **$destino** e **$oculto**, assim como os valores dos campos recuperados. Utilizamos a notação **<?=** para ser mais simples para você ver todo o código e colocamos um **@** na frente da variável.

Este arroba **@** faz com que o PHP ignore as mensagens de erro. Ele é necessário, caso não recebamos nenhum **id** pelo método **GET**. As variáveis de valores de formulário não existirão, o que geraria um erro.

Ainda não terminou, mas como este código é mais complicado e junta muito do que já vimos anteriormente, releia até entender tudo que foi modificado no código para podermos prosseguir com a página **planeta.alterar.php**.

A página **planeta.alterar.php** é muito semelhante à página **planeta.inserir.php**. Esta página é a que tem a responsabilidade de pedir ao MySQL para fazer a alteração no registro onde o **id** for igual ao recebido. Veja a Figura 82 com o código pronto e comentado em texto cinza.

```php
<?php
// Conectamos no banco.
require 'conexao.php';

// Colocamos em variáveis os dados enviados pelo formulário.
// A variável id possui o valor do campo escondido.
$id = $_POST['id'];
$nome = $_POST['nome'];
$tipo = $_POST['tipo'];
$tamanho = $_POST['tamanho'];
$descricao = $_POST['descricao'];

// Cuidado para não esquecer o WHERE ou o UPDATE irá modificar todos os registros do banco.
$sql = "UPDATE planetas SET nome='$nome', tipo='$tipo', tamanho=$tamanho, descricao='$descricao'
        WHERE id=$id";

// Enviamos o pedido ao banco e caso aconteça ao de errado escrevemos uma mensagem de erro.
mysqli_query($conexao, $sql) or die("Erro: " . mysqli_error($conexao));

// Se tudo der certo, chegamos até aqui e redirecionamos para a página de listagem.
header('Location: planetas.listar.php');
exit();
?>
```

Figura 82 - Página planetas.alterar.php.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra o código PHP que faz a modificação no banco de dados.

O que esta página tem de diferente de **planeta.inserir.php** é que ela recebe nosso campo oculto, o **id**, e a **$sql** muda para o comando SQL para alterar um registro já gravado, nesse caso, o registro que corresponde ao código enviado. O resto é igual à página **planeta.inserir.php**.

Vamos utilizar este sistema como base para sua avaliação, então treine fazendo a parte das estrelas.

Como sempre temos um exercício muito importante para a fixação. Com o que você aprendeu faça o sistema de alteração de 'estrelas'.

5.Competência 05 | Projeto: Emissão de Relatórios

Atualmente o PHP é muito poderoso. Com o apoio de suas bibliotecas podemos editar imagens e até gerar relatório em PDF.

PDF, **Portable Document Format**, é um formato de arquivo que foi desenvolvido pela Adobe System. Seu objetivo era fornecer uma representação de documento que fosse independente de qual aplicativo, hardware ou sistema operacional você estivesse usando para visualizar.

Arquivos PDF podem conter texto, gráficos e imagens que podem ser visualizados em qualquer dispositivo de forma idêntica.

Para que nossa aplicação PHP possa gerar relatórios em PHP, primeiramente precisamos baixar a biblioteca que irá fazer o trabalho de transformar nossas informações para esse formato.

Você pode baixar em **www.fpdf.org/**. A Figura 83 mostra a página do FPDF.

Figura 83 – FPDF Library.
Fonte: www.fpdf.gov (2016).
Descrição: Navegador exibe a página da biblioteca FPDF.

Clique em Downloads e irá aparecer a página da Figura 84. A página oferece o manual em português do Brasil e a versão mais atual da biblioteca. No momento em que este caderno foi escrito temos disponível a versão 1.8.1. A Figura 82 destaca em vermelho os links para baixar o manual e a biblioteca.

Figura 84 – Página de downloads do FPDF.
Fonte: Próprio autor (2016).
Descrição: A imagem exibe o local do link para baixar o arquivo da biblioteca que está logo abaixo do nome da versão v1.81 (2015-12-20).

Faça o download da biblioteca e descompacte os arquivos na pasta onde você está fazendo seu projeto até agora. Veja como ficou na Figura 85. Os arquivos importantes são a biblioteca que é o arquivo **fpdf.php** e a pasta **font**, que contém as fontes necessárias para formatar o documento.

Figura 85 – Pasta da biblioteca do FPDF
Fonte: Próprio autor (2016).
Descrição: A imagem mostra a lista com pastas e arquivos da biblioteca FPDF.

Agora que a biblioteca faz parte de nosso projeto, vamos construir dois botões para gerar um relatório cada. No primeiro teremos um relatório textual para mostrar algumas propriedades básicas. No segundo vamos pegar as informações do banco de dados, montar uma tabela e gerar o arquivo PDF.

Primeiramente vamos colocar os dois botões de impressão de relatório. Na página **planetas.listar.php** insira o código destacado na Figura 86 perto do final do arquivo. Atente para o local correto.

```
40                  <p><strong>Planetas</strong></p>
41                  <table>
42                      <tr>
43                          <th></th>
44                          <th>Nome</th>
45                          <th>Tipo</th>
46                          <th>Diâmetro Equatorial</th>
47                          <th></th>
48                          <th></th>
49                      </tr>
50  <?php
51  // O while repete a criação de linhas na tabela igual a quantidade de itens.
52  while($linha = mysqli_fetch_array($listagem)) {
53  // Cada dado da linha está no array $linha['nomedocampo']
54  ?>
55                      <tr>
56                          <td><a href="planetas.form.php?id=<?= $linha['id'] ?>">Editar</a></td>
57                          <td><?= $linha['nome'] ?></td>
58                          <td><?= $linha['tipo'] ?></td>
59                          <td><?= $linha['tamanho'] ?></td>
60                          <td><a href="planetas.excluir.php?id=<?= $linha['id'] ?>">Excluir</a></td>
61                          <td><a href="planetas.pdf.php?id=<?= $linha['id'] ?>">PDF</a></td>
62                      </tr>
63  <?php
64  // finaliza o bloco e inicia uma nova repetição se houver.
65  }
66  ?>
67                  </table>
68                  <p><a href="planetas.pdf.php">Relatório completo em PDF</a></p>
69              </section>
```

Figura 86 – Página planetas.listar.php e o código HTML com botões para gerar um relatório para cada planeta e outro para gerar um relatório completo em PDF. Modificação nas linhas 48, 61 e 68.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra o código para incluir os links para gerar o relatório geral e o relatório por cadastro.

Os botões, o que está na listagem e o que aparece no final da tabela, vão para a mesma página: **planetas.pdf.php**. Só que nos botões dentro da tabela passamos um id. Assim, caso a variável id seja enviada com um valor, vamos gerar um pdf com a ficha do planeta, senão, geramos um relatório completo com todos os planetas cadastrados no banco.

Antes disso, temos duas observações importantes.

Primeiro a biblioteca FPDF utiliza a notação de orientação de objetos, que veremos no próximo módulo. Mas não precisa se preocupar, ela é parecida com as funções. A segunda é que o FPDF não é totalmente preparado para os caracteres especiais de nossa língua, como os acentos, til e cedilha. Teremos que utilizar uma função bem pequena para fazer esta conversão. Esta função está logo no início do arquivo.

O código completo e comentado está na Figura 87. Logo após, temos a Figura 86 com o resultado em PDF. Posteriormente, temos as instruções de cada método utilizado para compor o documento e suas possíveis opções.

Tenha bastante atenção com a grafia correta de toda a codificação e, com a ajuda da explicação abaixo, tente entender o que cada linha ordena.

```php
<?php
require 'fpdf/fpdf.php'; // inclui na página a biblioteca FPDF
require 'conexao.php';
//mysqli_query($conexao, "SET NAMES 'utf8';");

// função para converter caracteres especiais, com acentos.
function converte($string) {
    return iconv("UTF-8", "ISO-8859-1", $string);
}

// cria um documento PDF no formato retrato, métrica de pontos e tamanho A4
$pdf = new FPDF('P','pt','A4');

$pdf->AddPage(); // adicionamos uma página
$pdf->Image('imagens/logo.pb.png'); // primeiro colocamos uma figura.

// endereço da empresa
$pdf->setFont('arial','',12);
$pdf->Cell(0,20,"Rua Fulano de Tal, s/n, Bairro Industrial",0,1,'L');

// e-mail para contato
$pdf->setFont('arial','',12);
$pdf->Cell(70,20,"atendimento@universodigital.com",0,1,'L');

// dá um espaçamento de 20 pontos.
$pdf->ln(20);

// primeiro configuramos a fonte
$pdf->SetFont('arial','B',18);
// cada linha é colocada em uma célula
$pdf->Cell(0,5,converte("Relatório"),0,1,'C');
// esta próxima linha faz uma linha abaixo do título 'Relatório'
$pdf->Cell(0,5,"","B",1,'C');
$pdf->Ln(20);

// se for enviado uma variável codigo pelo método get e ela não for vazia
// pegamos o registro solicitado, senão pegamos a lista completa.
if (isset($_GET['id']) && $_GET['id']<>"") {
    $id = $_GET['id'];
    $sql = "SELECT * FROM planetas WHERE id='$id';";
} else {
    $sql = "SELECT * FROM planetas;";
}

// Recupera um registro ou toda lista do banco
$listagem = mysqli_query($conexao, $sql);

while($linha = mysqli_fetch_array($listagem)) {

    $pdf->SetFont('arial','B',12);
    // Quando o texto tem ou pode ter acento utilizamos a função converte
    $pdf->Cell(70,20,converte("Código: "),0,0,'L');
    $pdf->setFont('arial','',12);
    $pdf->Cell(0,20,$linha['id'],0,1,'L');

    // nome do planeta
    $pdf->SetFont('arial','B',12);
    $pdf->Cell(70,20,"Filme:",0,0,'L');
    $pdf->setFont('arial','',12);
    $pdf->Cell(0,20,converte($linha['nome']),0,1,'L');

    // diâmetro equatorial
    $pdf->SetFont('arial','B',12);
    $pdf->Cell(70,20,converte("Diâmetro: "),0,0,'L');
    $pdf->setFont('arial','',12);
    $pdf->Cell(70,20,$linha['tamanho'],0,1,'L');

    // descrição
    $pdf->SetFont('arial','B',12);
    $pdf->Cell(70,20,converte("Descrição: "),0,1,'L');
    $pdf->setFont('arial','',12);
    $pdf->MultiCell(0,20,converte($linha['descricao']),0,'J');

    // tipo do planeta
    $pdf->SetFont('arial','B',12);
    $pdf->Cell(70,20,"Tipo:",0,0,'L');
    $pdf->setFont('arial','',12);
    $pdf->Cell(70,20,$linha['tipo'],0,1,'L');

    $pdf->Ln(20);
}

//emissão do relatório
$pdf->Output('relatorio.pdf', 'I');
?>
```

Figura 87 – Página planetas.pdf.php.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra o código completo para impressão do relatório geral e individual utilizando a biblioteca FPDF.

Primeiramente, incluímos a biblioteca **fpdf.php**. Uma particularidade desta biblioteca é o tratamento parcial para codificação UTF-8. Devido a isso, os caracteres especiais devem ser tratados com outra função, para isso criamos a função **converte( )** que recebe como argumento o texto, trata e devolve já recodificado adequadamente.

Criamos, então, uma variável **$pdf,** que será nosso documento. Instanciamos com **new FPDF('P', 'pt', 'A4')**. Essas são as configurações para o documento. O primeiro argumento refere-se ao formato **P.** É retrato, mas pode ser **L** para paisagem; o segundo argumento é para a unidade de medida, podendo ser **'cm'** centímetro, **'pt'** pontos, **'mm'** milímetros e **'in'** polegadas; o terceiro argumento é o tamanho do papel, temos **'A3'**, **'A4'**, **'A5'**, **'letter'** e **'legal'**.

Agora chamamos os métodos do objeto para configurar o documento. Utilizamos o método **$pdf->AddPage();** para adicionar uma página e abaixo adicionamos a imagem **logo.pb.png** no PDF. Note que devemos fornecer o endereço da imagem PATH ou URL. Outros parâmetros que podemos inserir são (na sequência):

```
Image(string file [, float x [, float y [, float w [, float h [, string type]]]]])
```

- X: posição da imagem em relação ao eixo X.
- Y: posição da imagem em relação ao eixo Y.
- W: largura da imagem.
- H: altura da imagem.
- TYPE: tipo da imagem.

Então, para cada linha temos a configuração de fonte com **$pdf->SetFont('arial', '', 12).** O primeiro argumento é a fonte Arial. A string vazia '' é para configurar o efeito, se quiser negrito utilize 'B', e o 12 é o tamanho em pontos da fonte. Veja a pasta **"/fpdf/font/"** para verificar as fontes suportadas.

Só se consegue escrever no PDF dentro de uma cell ou multicell. Uma cell é como uma célula, um espaço para escrever uma linha. O primeiro parâmetro é a largura, o segundo a altura, o terceiro argumento é o conteúdo (o texto), o quarto argumento é a borda (1 para sim e 0 para não), o quinto é o espaçamento interno, o sexto é para o alinhamento (L esquerda, R direita e C centralizado). Quando o conteúdo possui caracteres especiais, chamaremos nossa função **converte()**.

```
$pdf->Cell(0,20,"Rua Fulano de Tal, s/n, Bairro Industrial",0,1,'L');
```

Para dar um espaçamento de 20 pontos entre uma linha e outra utilizamos

```
$pdf->ln(20);
```

Quando temos várias linhas utilizamos multicell. Seus parâmetros são iguais aos do cell com a opção a mais de alinhamento J para justificado. Atente de que já tratamos deste texto anteriormente.

```
$pdf->MultiCell(0, 20, “texto de várias linhas...”, 0, 'J');
```

Agora temos que definir o conteúdo, que pode ser de duas maneiras. Se a página receber um id, o documento gera apenas a ficha do planeta. Se não possuir nada, o documento gerado conterá uma listagem de todos os planetas cadastrados. Para isso, definimos a string SQL que será utilizada. Com isso, ou selecionamos um planeta pelo id ou todos, igual ao que fizemos na **planetas.listar.php**.

```
if (isset($_GET['id']) && $_GET['id']<>"") {
		$id = $_GET['id'];
		$sql = "SELECT * FROM planetas WHERE id='$id';";
} else {
		$sql = "SELECT * FROM planetas;";
}
```

Depois que conseguirmos a listagem do banco, iniciamos uma estrutura while para repetir o código para cada linha. Abaixo, temos o código inicial. A cada iteração ele constrói uma linha para a tabela, com os dados do DVD.

```
while($linha = mysqli_fetch_array($listagem)) {
		$pdf->SetFont('arial','B',12);
		// Quando o texto tem ou pode ter acento utilizamos a função converte
		$pdf->Cell(70,20,converte("Código: "),0,0,'L');
		$pdf->setFont('arial','',12);
		$pdf->Cell(0,20,$linha['id'],0,1,'L');
```

Na última linha temos o envio do documento através de

```
$pdf->Output('relatorio.pdf', 'I');
```

Os parâmetros são dois: O nome do arquivo e o destino (I para a saída padrão, D para download, F para salvar localmente e S para retornar como string).

Observe o resultado na Figura 88 e 89.

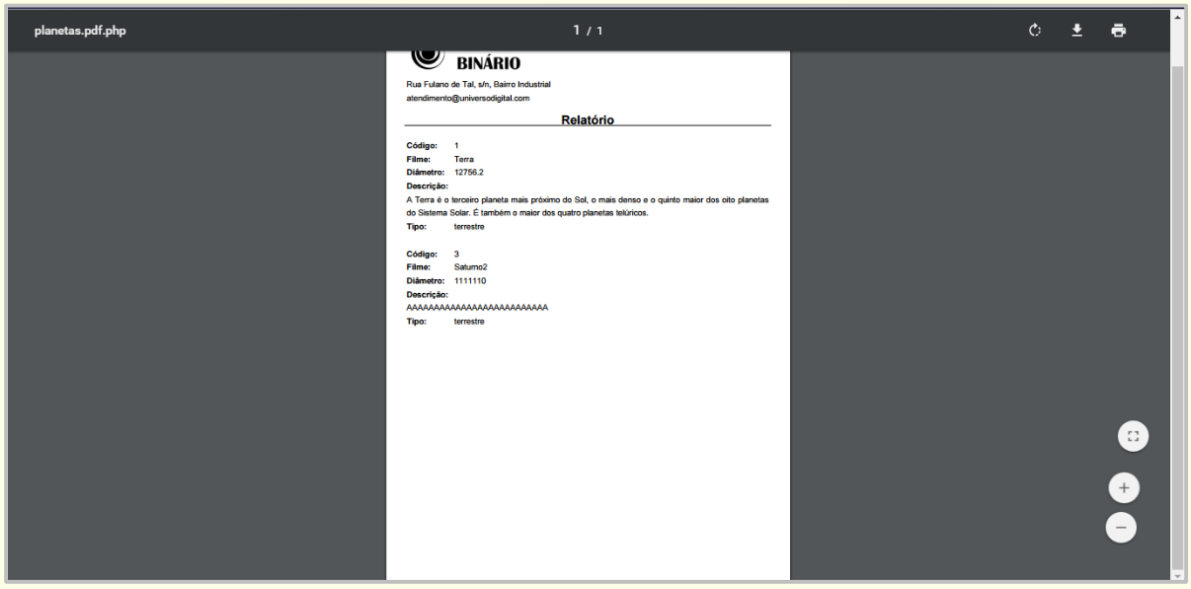


Figura 88 – Arquivo de relatório geral em PDF criado pelo FPDF do código de exemplo.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra o resultado no navegador Chrome. O Chrome abri os arquivos PDF que recebe

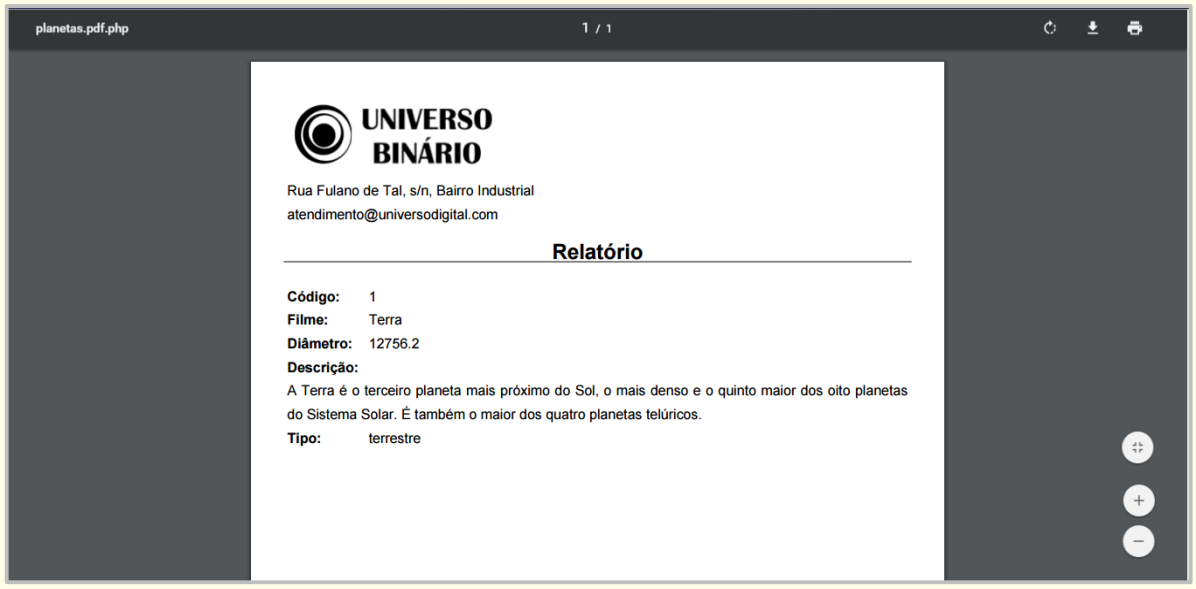


Figura 89– Arquivo de relatório individual criado pelo relatorio.php.
Fonte: Próprio autor (2016).
Descrição: A imagem mostra a exibição do relatório individual no navegador Chrome.

Esta codificação é complexa. Você precisa ler com atenção e reproduzir o exemplo para entender bem todos eles. Utilize outras opções para ver como fica.

Você pode ser criativo e tentar novas formas de relatórios e documentos. Quanto mais fizer, mais

aprenderá.

Agora, para exercitar, faça da mesma forma como fizemos anteriormente, só que utilize a entidade ‘estrelas’.

Conclusão

Com isto terminamos esta disciplina, mas entenda que você viu a ponta do iceberg, ou seja, apenas o comecinho. Existe muito mais conhecimento que deve ser procurado e aprendido para se aperfeiçoar. Mas apenas ler ou assistir uma aula não é suficiente. Para aprender você deve praticar, mais e mais. A diferença de um iniciante para um profissional é a experiência acumulada com horas e horas de prática.

Então, pesquise mais sobre HTML, CSS, JavaScrip e PHP. Na internet você encontrará muito material interessante. E quanto mais cedo começar a praticar, mais rápido se tornará um profissional competente.

Um grande abraço e até a próxima.

Referências

NIEDERAUER, Juliano. **Desenvolvendo websites com PHP**. São Paulo: Novatec, 2004.

CONVERSE, Tim; PARK, Joyce. **PHP:** a bíblia. 2003.

DALL'OGLIO, Pablo. **PHP:** programando com orientação a objetos. São Paulo: Novatec, 2007.

Minicurrículo do Professor

Ewerton Mendonça é formado em Sistemas de Informação pela UPE e Design pela UFPE, com mestrado em Ciência da Computação pela UFPE. Atualmente é professor na Faculdade de Ciências e Letras de Caruaru. Possui experiência na área de desenvolvimento WEB e design gráfico desde 1998.