**Guia de Implementação**

SUMÁRIO

[1. Introdução 2](#_Toc485801465)

[1.1. Referências 2](#_Toc485801466)

[2. Organização e Estilo do Código 2](#_Toc485801467)

[3. Comentários 2](#_Toc485801468)

[4. Nomeação 2](#_Toc485801469)

[5. Declaração 3](#_Toc485801470)

[6. Reutilização 3](#_Toc485801471)

[7. Tratamento de Erros e de Exceções 3](#_Toc485801472)

[8. teste unitário 3](#_Toc485801473)

[9. Diretrizes Gerais 4](#_Toc485801474)

# Introdução

Este documento tem a finalidade de documentar a implementação do projeto de acordo com os padrões SpinOff. O objetivo é mostrar uma visão geral do sistema.

## Referências

# Organização e Estilo do Código

# Comentários

Os comentários em PHP são marcações que colocamos dentro do nosso código para indicar ao interpretador que uma linha ou então, um conjunto de linhas não deve ser interpretado.

Também é comum marcamos uma linha de código como comentário para que possamos observar como o programa reage sem essa linha ou então, quando precisamos alterar o nosso código, porém, não desejamos deletar a linha e então, só marcamos a mesma como sendo um comentário.

Há 3 maneiras de definir comentários dentro do código.

**Duas Barras - //**

A primeira forma e a mais comum, é utilizando os caracteres //. Essa é uma notação herdada do C e assim, a mais difundida dentre os programadores.

**Caractere Cerquilha - #**

Outra forma, é a notação utilizando o caractere cerquilha, comumente chamado de Jogo da Velha. Essa é uma outra maneira de informarmos que uma única linha não deve ser interpretada, pelo interpretador.

## Definição de Bloco de comentário - /\* ... \*/

Por fim, temos a notação que proporciona a demarcação de um conjunto de linhas como sendo comentário. Logo, conseguimos rapdiamente demarcar toda uma região a fim de informar ao interpretador que estas linhas não devem ser interpretadas.

# Nomeação

XAMPP

* htdocs
* delivery

Arquivo PHP

* .htaccess.php
* Autenticar.php
* Cabecalho.php
* Cadastrar-usuario.php
* Categorias.php
* Combos.php
* Conexao.php
* Config.php
* Index.php
* Lista-produtos.php
* Login.php
* Logout.php
* Produtos.php
* Recuperar.php
* Rodape.php
* Sobre.php

# Declaração

1. declaração de variáveis em PHP

$variavel;

1. Variáveis dinâmicas

*//Variavel $nome recebe o valor “variavel”*

*$nome = 'variavel';*

*//Declaração da nova variável. O identificador será o conteúdo de $nome*

*$$nome = 'Felipe';*

*//Exibe o conteúdo de $variavel*

*echo $variavel;*

# Reutilização

* Resolução de conflitos;
* Alterando a visibilidade do método;
* Utilizando o método de reutilização de código que é o Traits

# Tratamento de Erros e de Exceções

Um bom tratamento de erros precisa prever vários tipos de erros e, pra isso, existem funções específicas, como por exemplo:

[file\_exists](http://www.php.net/file_exists) – serve para verificar se um arquivo existe  
[defined](http://www.php.net/defined) – serve para verificar se uma constante foi definida.  
[isset](http://www.php.net/isset) – verifica se uma variável existe.  
[is\_array](http://www.php.net/is_array) – verifica se a variável é um array  
[is\_resource](http://www.php.net/is_resource) – verifica se a variável é um resource  
[is\_numeric](http://www.php.net/is_numeric) – verifica se a variável é um número ou uma string numérica.  
[is\_uploaded\_file](http://www.php.net/is_uploaded_file) – verifica se o arquivo foi uploaded via HTTP POST.  
[is\_writable](http://www.php.net/is_writable) – verifica se pode escrever para o arquivo (writable).  
[version\_compare](http://www.php.net/version_compare) – Compara a versão do php.  
[extension\_loaded](http://www.php.net/extension_loaded) – verifica se a extensão foi habilitada.

Com essas funções você pode testar, por exemplo, se um arquivo existe, antes de incluí-lo. Caso não exista, você pára o script e dá um aviso para o usuário. Outro exemplo: antes de gravar um arquivo texto, você verifica se ele pode ser escrito, ou antes de usar uma variável teste, se ela existe com isset()  
Outra validação muito importante é validar se o formulário foi enviado, assim evitando erro, por exemplo:

# teste unitário

PHPUnit: **O PHPUnit é um framework para realização de testes unitários em PHP com base na arquitetura xUnit.**

Teste unitário é uma metodologia de testes automatizados que leva em consideração a menor unidade do software, ou seja, ao invés de testar dentro do código fonte e executar o código manualmente para imprimir o resultado na tela, o programador cria um projeto de teste, que executa cada trecho do código de forma automática e exibe o resultado de todos os testes.

# Diretrizes Gerais