**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 customização do processo padrão SpinOff a ser utilizado pelo projeto. O objetivo é orientar os membros do time e novos membros das mudanças realizadas em relação ao processo padrão.

## Referências

https://www.devmedia.com.br/teste-unitario-com-phpunit/41231#instalao

LARMAN, Craig. **Utilizando UML e Padrões:** Uma introdução à análise e ao projeto orientados a objetos e ao Processo Unificado. 2. ed. Porto Alegre: Bookman, 2004. 607 p.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. Metodologia do trabalho científico: métodos e técnicas de pesquisa e do trabalho acadêmico. 2. ed. Rio Grande do Sul: Feevale, 2013. 276 p.

# Organização e Estilo do Código

* Não existe nenhum limite de linhas ou caracteres para um método. O ideal é que ele resolve apenas um problema, ou seja, tenha apenas uma responsabilidade.
* A primeira regra de funções é que elas devem ser pequenas. A segunda regra é que elas deveriam ser menores que isso. Não devem possuir mais que 100 linhas. Dificilmente deveriam passar das 20 linhas.
* Os parâmetros nomeados entram exatamente na questão da legibilidade do código. Isso não significa que só por usar parâmetros nomeados o código vai ser incrivelmente legível, porém é um recurso simples que pode ajudar bastante

# Comentários

Há 3 maneiras de definir comentários dentro do código em PHP por exemplo:

**Duas Barras - //**

$c = 1; // esta variável tem o valor 1  
  
$d = 2; // esta variável tem o valor 2  
  
// Aqui fazemos a soma das variáveis  
  
$e = $c + $d;  
  
// Aqui exibimos o resultado na tela  
  
echo “$e”;

**Caractere Cerquilha - #**

# Option 1

# Option 2

# Option 3

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

/\* Title Here Notice the First Letters are Capitalized \*/

/\*This is a detailed explanation of something that should requireseveral paragraphs of information. \*/

# Nomeação

De acordo com o padrão de arquitetura de software MVC (Model-View-Controller)

* As classes criadas em controllers seguem o seguinte padrão: NomeDaClasseController.php;
* Já as classes criadas em models seguem o padrão: Nome.php;
* E as classes criadas em **views** seguem o padrão: **nome.php** ou **nome.blade.php** (no caso de utilizar o Blade para templates).

# Declaração

declaração de variáveis em PHP

$variavel;

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.**

**1° Instalação:**

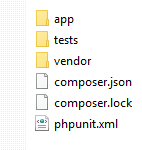
O PHPUnit pode ser instalado pelo Composer, um gerenciador de dependências do PHP que permite a declaração das bibliotecas das quais o seu projeto depende e as instala de forma automatizada.

Para isso execute o seguinte comando no diretório raiz do seu projeto:

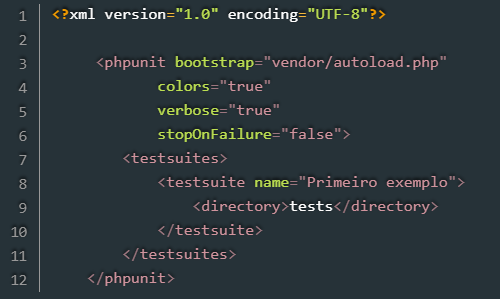


**2° Configuração:**

Para utilizar o PHPUnit dentro do seu projeto é preciso criar na raiz do mesmo um arquivo chamado phpunit.xml e uma pasta chamada **tests**.



No arquivo phpunit.xml adicione as seguintes configurações:



# Diretrizes Gerais