

# Criando um plug-in em WordPress

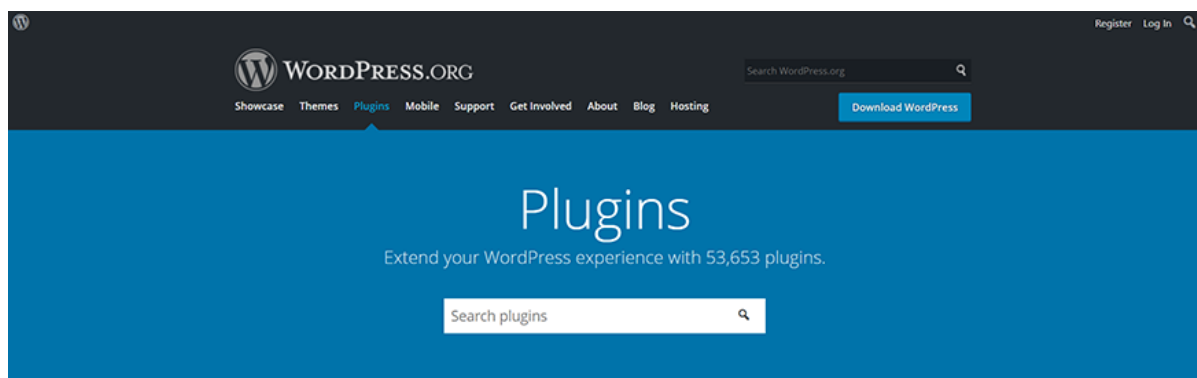
APRENDER A CRIAR PLUGINS PARA O CMS WORDPRESS

AUTOR(A): PROF. GERSON RISSO

## Plugins no WordPress - Construção de plugins

O WordPress é um CMS que possibilita a criação de, praticamente, qualquer sistemas on line. Por isso, tem um painel administrativo simples, com funcionalidades, módulos e plugins de uso geral. É possível reconfigurar as telas, módulos e outras funcionalidades, em um nível mais avançado, utilizando-se de linguagens de programação. E para todos os níveis, é possível adquirir plugins para uso específico, com esse tipo de arquitetura o WordPress favorece o desenvolvedor que vai incorporando recursos à medida que necessitar.

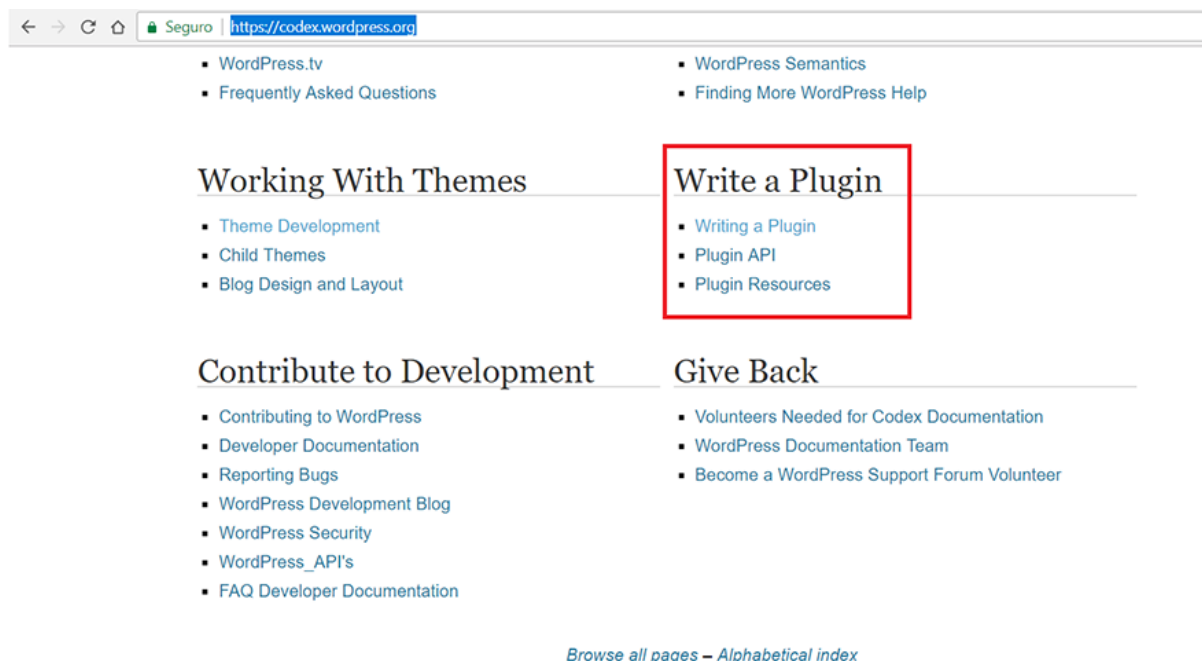
Por ser um CMS muito popular, mundialmente, o WordPress conta com uma grande comunidade que compartilha conhecimentos e recursos de modo a tornar a extensibilidade do WordPress, a característica mais marcante frente a outros tantos CMS disponíveis no mercado. E em grande parte, a extensibilidade do WordPress, vem da vasta coleção de plugins dedicados disponíveis em seu repositório..



Caso você não consiga encontrar um plugin para a sua necessidade ou ainda, se você tiver necessidade de aplicação muito específica ou ainda, deseja uma ferramenta personalizada, pode criar o próprio seu plugin. Pode ainda, disponibilizar o seu plugin para outros desenvolvedores e até investir na carreira de desenvolvedor de plugin para WordPress.

A linguagem de programação base para a criação de plugins é o PHP, portanto, ter conhecimentos em PHP é um belo passo para desenvolver esse recurso fundamental em WordPress.

O WordPress disponibiliza uma vasta documentação para consulta, especificamente para plugins podemos encontrar material em: <https://codex.wordpress.org>.










Aqui você encontra informações completas e necessárias para a criação e compartilhamento de plugins. A documentação dá dicas de desenvolvimento, compartilhamento, internacionalização do plugin. É possível encontrar também, dicas de segurança e atualização do plugin.

Para a criação de um plugin vamos, primeiramente criar uma pasta dedicada aos nossos plugins.

Abra a pasta no Xampp, dedicada ao WordPress. Provavelmente deve estar na pasta htdocs, então, vá até a pasta wp-content. É nessa pasta que ficam os arquivos de plugins, temas, atualizações entre outros arquivos essenciais à estrutura do WordPress.

> Disco Local (C:) > xampp > htdocs > wordpress > wp-content >

Nome	Data de modificaç...	Tipo
 languages	27/11/2017 15:59	Pasta de arquivos
 ngg	17/12/2017 15:39	Pasta de arquivos
 plugins	06/01/2018 07:36	Pasta de arquivos
 themes	03/01/2018 13:58	Pasta de arquivos
 upgrade	06/01/2018 07:36	Pasta de arquivos
 uploads	05/01/2018 15:50	Pasta de arquivos
 index.php	08/01/2012 17:01	Arquivo PHP

Abra a pasta plugins e veja que há várias pastas, sendo que cada uma representa um plugin, caso você tenha instalado algum. Se não tem plugins instalados, isso não afetará em nada o desenvolvimento do seu plugin.

Dê um nome à pasta: Meus plugins é uma sugestão para armazenar, outras pastas que vão conter os nossos plugins. Crie uma nova pasta meuplugin\_teste1, o nome do plugin será exatamente o nome dessa pasta que acabamos de criar.

Abra um editor para desenvolver o conteúdo do plugin, o editor pode ser Brackets, Atom ou outro editor equivalente. Escolhido o seu editor, abra a tela de edição e salve o arquivo vazio na pasta meuplugin\_teste1, com o nome de meuplugin\_teste1.php. Salve o arquivo com o mesmo nome da pasta e extensão PHP.

Atenção para o nome do plugin, ele deve ser único, pois, pode gerar conflitos caso queira compartilhar o seu plugin no repositório. É uma dica importante e que consta na documentação do WordPress, que foi apresentada aqui. É claro que para o nosso caso isso não será necessário, mas, é desejável que você siga o padrão, a fim de se acostumar com as boas práticas.

Agora vamos escrever o nosso código PHP. Conforme documentação do WordPress, é recomendável que se coloque no início do código a instrução: `defined( 'ABSPATH' ) or die( 'No script kiddies please!' );` Afim de evitar invasão aos arquivos PHP.

Outra parte padrão de um plugin WordPress, é identificação do seu plugin. Segundo a documentação WordPress, a identificação (header) deve seguir o modelo:

Plugin Name: Nome do plgin

Plugin URI: A URI relacionada ao plugin

Description: Um breve texto sobre a funcionalidade do plugin

Version: A versão do plugin

Author: Nome do autor do plugin

Author URI: A URI relacionada ao autor

Há outras informações para o header que aqui foram omitidas mas, constam da documentação WordPress.

## QUER SABER MAIS SOBRE PHP?

Acesse o link

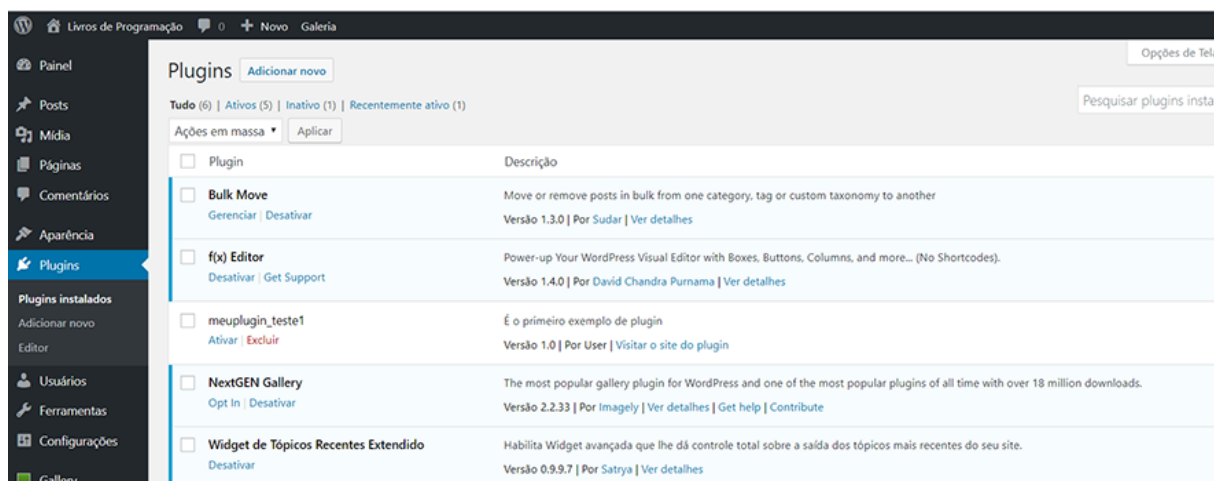
<http://php.net/> (<http://php.net/>)

Acesse também o link

<https://www.w3schools.com/> (<https://www.w3schools.com/>)

```
1. <?php
2. /*Impede a invasão aos arquivos PHP.*/
3. defined( 'ABSPATH' ) or die( 'No script kiddies please!' );
4.
5. /* Header padrão para um Plugin - Documentação WordPress
6. Plugin Name: meuplugin_teste1
7. Plugin URI: http://www.uninove.br
8. Description: É o primeiro exemplo de plugin
9. Version: 1.0
10. Author: User
11. */
12. ?>
```

Com esse pequeno trecho de script contendo as informações do plugin no header são suficientes para que o WordPress identifique o plugin criado.



Digite o script acima, usando o seu editor escolhido e salve as mudanças. Agora vá no painel do WordPress e verifique no menu Plugins, se o seu plugin foi reconhecido.

Agora vamos inserir alguma funcionalidade no plugin, porque se você ativá-lo ele simplesmente não vai fazer nada, já que não tem nenhuma linha de comando inserida em seu código.

Porém, é necessário entender um conceito fundamental do WordPress, os Hooks (Ganchos). São uma forma de realizar a interação entre o funcionamento do WordPress com os temas e plugins. Os hooks são a base de funcionamento de plugins e temas, mas conforme documentação do WordPress, também são usados extensivamente pelo próprio núcleo do WordPress.

Os hooks não funcionam de maneira independente, eles acionam as funções escritas pelo desenvolvedor. Há dois tipos de hooks, as ações e os filtros, eles são encontrados às centenas. No WordPress há um riquíssimo material com informações úteis e exemplos práticos no uso dos hooks (WordPress, 2017).

Os hooks são utilizados para modificar o comportamento do WordPress em um momento específico, por exemplo, quando um conteúdo é recuperado do banco de dados e antes de ser direcionado para um processamento específico, sofre uma modificação por um plugin que interagiu com um hook específico.

Ações e filtros são conceitos distintos, as ações executam algum tipo de tarefa em um comportamento específico no núcleo do WordPress. Já os filtros, recebem dados e podem alterar esses dados. Conforme documentação do WordPress, os filtros devem trabalhar de forma isolada, isto é, devem realizar modificações localmente, não podem realizar modificações de escopo global.

Para apresentar um exemplo prático vamos continuar a codificação do nosso plugin, que até o momento não faz nada. Abra o seu editor e adicione as linhas de código apresentadas a seguir.

Veja uma amostra da página de referência de filtros do WordPress. Há uma descrição do filtro `the_content` e exemplos práticos de uso. Nós vamos abordar alguns exemplos prático usando esse mesmo filtro, embora, como já afirmamos, há uma centena desse filtros que você pode examinar em mais detalhes no link:

[https://codex.wordpress.org/Plugin\\_API/Filter\\_Reference/the\\_content](https://codex.wordpress.org/Plugin_API/Filter_Reference/the_content)

([https://codex.wordpress.org/Plugin\\_API/Filter\\_Reference/the\\_content](https://codex.wordpress.org/Plugin_API/Filter_Reference/the_content)).



**WORDPRESS.ORG**

[Showcase](#)
[Themes](#)
[Plugins](#)
[Mobile](#)
[Support](#)
[Get Involved](#)
[About](#)
[Blog](#)
[Hosting](#)

[Download WordPress](#)

Codex

Codex tools: Log in

 Interested in functions, hooks, classes, or methods? Check out the new WordPress Code Reference!

## Plugin API/Filter Reference

Languages: [English](#) • [日本語](#) • [\(Add your language\)](#)

This article contains an extensive (but not 100% comprehensive) list of the filter hooks available for use in plugin development in Version 2.1 and above of WordPress. For more information:

- To learn more about what filter and action hooks are, see [Plugin API](#).
- To learn about writing plugins in general, see [Writing a Plugin](#).
- For a reference list of action hooks, see [Plugin API/Action Reference](#).
- For information about filter and action hooks in previous versions of WordPress, see [Plugin API/Hooks 2.0.x](#).
- For an automatically-generated list of all WordPress hooks, see the [WordPress Hooks Database](#)

Note: If you want to add to or clarify this documentation, please follow the style of the existing entries. Describe what data the filter is

[Home Page](#)
[WordPress Lessons](#)
[Getting Started](#)
[Working with WordPress](#)
[Design and Layout](#)
[Advanced Topics](#)
[Troubleshooting](#)
[Developer Docs](#)
[About WordPress](#)
  
  
**Codex Resources**
[Community portal](#)
[Current events](#)
[Recent changes](#)

```

1. function teste1_ModificaPost(){
2.     return "Esse conteúdo vem do plugin";
3. }

```

Escrevemos uma função com o cuidado de criar um nome bem específico, a fim de que não haja conflito com outras funções. Essa função retorna um conteúdo padrão, mas, retorna para quem?

Retorna para um filtro que modificará algum conteúdo, para esse exemplo vamos inserir a mensagem de retorno da função nos posts publicados. Para isso precisamos de um filtro, ele vai receber o conteúdo e alterá-lo.

```

1. add_filter('the_content','teste1_ModificaPost');

```

Na instrução acima, conforme a documentação do WordPress, the\_content é filtro para o conteúdo de posts. Esse filtro atua entre um momento após um conteúdo ser recuperado do banco de dados e antes de ser exibido no browser. O outro parâmetro da função add\_filter() refere-se ao nome da função criada por nós. Conforme o WordPress, a função add\_filter é utilizada para adicionar um filtro a uma ação específica.

Agora vamos modificar a função teste1\_ModificaPost() receberá o conteúdo post, incluirá a mensagem ao conteúdo original do post.

```
1. <?php
2. /*Impede a invasão aos arquivos PHP.*/
3. defined( 'ABSPATH' ) or die( 'No script kiddies please!' );
4.
5. /*
6.   Plugin Name: meuplugin_teste1
7.   Plugin URI: http://www.uninove.br
8.   Description: É o primeiro exemplo de plugin
9.   Version: 1.0
10.  Author: User
11. */
12.
13. function teste1_ModificaPost($parametro){
14.     $conteudo="<p style=color:red;>Essa mensagem vem do meu plugin</p>";
15.     $conteudo=$conteudo.$parametro;
16.     return $conteudo;
17. }
18.
19. add_filter('the_content','teste1_ModificaPost');
20.
21. ?>
```

izar 0 + Novo

12 DE DEZEMBRO DE 2017 EDITAR

## Linguagem C

Essa mensagem vem do meu plugin

A primeira dica é sobre um livro de linguagem C. C a linguagem padrão ANSI.

Os autores, Brian Kernighan e Dennis Ritchie são também os criadores da linguagem, o que torna essa obra em uma referência de peso.

Quando o WordPress é executado o filtro `the_content` recebe o conteúdo do banco de dados e a função `add_filter` chamada a função criada por nós, dentro da função `teste1_ModificaPost()` o conteúdo do post é concatenado com a mensagem da variável `$conteudo` e retornado para a função `add_filter`. O filtro `the_content`, então envia para a exibição no browser.

Agora vamos fazer uma alteração no título, para isso vamos criar outra função e escolher outro filtro, para esse caso será o filtro `the_title`.

```
1. function teste1_ModificaTitulo($parametro){
2.     $conteudo="<h2 style=font-family:arial;>".$parametro."</h2>";
3.     return $conteudo;
4. }
```

```
1. add_filter('the_title','teste1_ModificaTitulo');
```

```
1. <?php
2. /*Impede a invasão aos arquivos PHP.*/
3. defined( 'ABSPATH' ) or die( 'No script kiddies please!' );
4.
5. /*
6.   Plugin Name: meuplugin_teste1
7.   Plugin URI: http://www.uninove.br
8.   Description: É o primeiro exemplo de plugin
9.   Version: 1.0
10.  Author: User
11. */
12.
13. function teste1_ModificaPost($v){
14.     $conteudo="<p style=color:red;>Essa mensagem vem do meu plugin</p>";
15.     $modificado=$modificado.$conteudo.$v;
16.     return $modificado;
17. }
18.
19. add_filter('the_content','teste1_ModificaPost');
20.
21. ?>
```



Insira a função `teste1_ModificaTitulo()` e a função `add_title()` no código do plugin. Salve-o e atualize o WordPress, verá que agora o título dos posts ficaram formatados com o subtítulo e família de fontes Arial, conforme definida na função `teste1_ModificaTitulo()`.



12 DE DEZEMBRO DE 2017 EDITAR

## Linguagem C

Essa mensagem vem do meu plugin

A primeira dica é sobre um livro de linguagem C. C a linguagem padrão ANSI.

Os autores, Brian Kernighan e Dennis Ritchie são também os criadores da linguagem, o que torna essa obra em uma referência de peso.

Agora vamos criar um outro plugin para realizar a contagem de palavras do conteúdo publicado. Assim, será exibido o conteúdo do post e a respectiva quantidade de palavras,

Para isso, crie um novo arquivo na pasta `htdocs` do Xampp, procure a pasta dedicada ao WordPress e na sequência procure a pasta `wp-content`. Crie ali uma pasta nomeada como `meuplugin_teste2`.

Em seguida abra o seu editor e escreva o código apresentado na sequência e salve como `meuplugin_teste2.php`.

### PARA SABER MAIS SOBRE HOOKS NO WORDPRESS

Acesse o link

<https://developer.wordpress.org/plugins/hooks/> (<https://developer.wordpress.org/plugins/hooks/>)

```
1. <?php
2. /*

3.   Plugin Name: meuplugin_teste2
4.   Plugin URI: http://www.uninove.br
5.   Description: É o segundo exemplo de plugin. Ele faz a contagem de palavras de um
6.   Version: 1.0
7.   Author: User
8. */
9.
10. /*Impede a invasão aos arquivos PHP.*/
11. defined('ABSPATH') or die('No script kiddies please!');
12.
13. function teste2_contaPalavrasPost($publicacao){
14.     $palavras = explode(" ", $publicacao);
15.     return $publicacao."<p style=color:green> Esse post contém: ".count($palavras).";
16. }
17.
18. add_filter('the_content','teste2_contaPalavrasPost');
19. ?>
```

Há algumas coisas que devemos explicar antes de ver o resultado, a primeira delas é que o filtro `the_content` é o mesmo do plugin anterior, já que, queremos tratar o conteúdo do post. Porém, esse filtro não altera o conteúdo da publicação porque a função `teste2_contaPalavrasPost()` está processando informação entre a consulta no banco de dados e a exibição no browser.

A função criada para esse plugin, `teste2_contaPalavrasPost()` contém duas funções nativas do php: `explode()` e `count()`.

A função `explode(" ", $publicacao)` separa a string, no nosso caso é o conteúdo postado representado pela variável `$publicacao`, por strings, no nosso caso são os espaços " ". Assim, teremos na variável `$publicacao` uma string com os seus elementos espaçados.

A função `count` realiza a contagem de elementos de uma string, nesse caso é o conteúdo do nosso post formatado pela função `explode()` (Niederaurer, 2017).

Como resultado a função `teste2_contaPalavrasPost()` retorna o conteúdo do post com a quantidade de palavras contidas nele. Agora, ative o plugin e atualize a página para conferir o resultado.

zar 0 + Novo

12 DE DEZEMBRO DE 2017 EDITAR

## Linguagem C

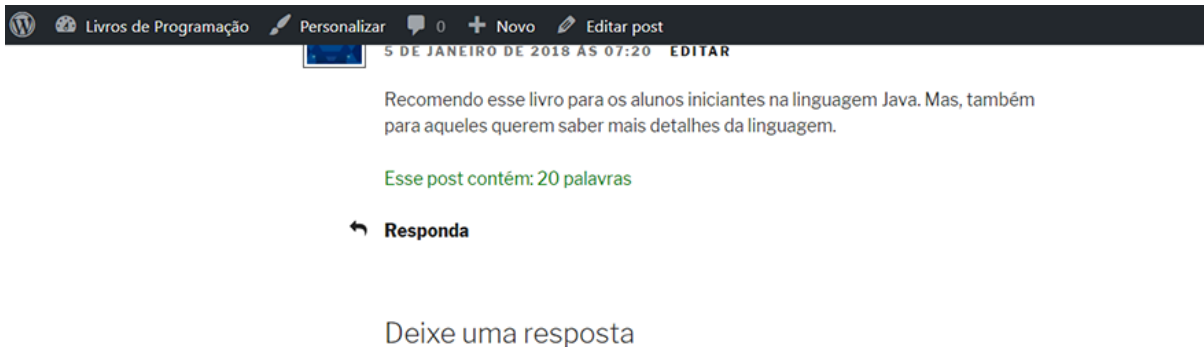
A primeira dica é sobre um livro de linguagem C. C a linguagem padrão ANSI.

Os autores, Brian Kernighan e Dennis Ritchie são também os criadores da linguagem, o que torna essa obra em uma referência de peso.

Esse post contém: 37 palavras

Podemos alterar o filtro do plugin para exibir a quantidade de palavras do comentário de um post, o filtro para esse caso é o `the_comment`. Faça as alterações e salve o arquivo. Agora veja o resultado.

```
1. <?php
2. /*
3.   Plugin Name: meuplugin_teste2
4.   Plugin URI: http://www.uninove.br
5.   Description: É o segundo exemplo de plugin
6.   Version: 1.0
7.   Author: User
8. */
9.
10. /*Impede a invasão aos arquivos PHP.*/
11. defined('ABSPATH') or die('No script kiddies please!');
12.
13. function teste2_contaPalavrasPost($publicacao){
14.   $palavras = explode(" ", $publicacao);
15.   return $publicacao."<p style=color:green> Esse post contém: ".count($palavras).";
16. }
17.
18. add_filter('comment_text','teste2_contaPalavrasPost');
19. ?>
```



Vamos criar um terceiro plugin cujo arquivo será nomeado meuplugin\_teste3.php. Não esqueça de criar uma pasta com o mesmo nome, no mesmo local onde foram criadas as outras pastas dos plugins. Nesse plugin vamos exibir os conteúdos juntamente com a informação de tempo transcorrido, da data de publicação até a data atual.

No editor, escreva o código apresentado a seguir.

```
1. <?php
2. /*
3.   Plugin Name: meuplugin_teste3
4.   Plugin URI: http://www.uninove.br
5.   Description: Esse plugin apresenta no post o tempo transcorrido desde a publicaç
6.   Version: 1.0
7.   Author: User
8. */
9.
10. /*Impede a invasão aos arquivos PHP.*/
11. defined('ABSPATH') or die('No script kiddies please!');
12.
13. function intervalo_de_tempo($post){
14.     return $post."<b> Publicado a ".human_time_diff(get_the_time('U'), current_tim
15. }
16. add_filter('the_content','intervalo_de_tempo');
17. ?>
```

Com relação à função teste3\_tempo(), a linha de código 14 retorna a string de publicação concatenada com uma mensagem e com o resultado da função human\_time\_diff() que, basicamente, exibe o intervalo de tempo em palavras.

O protótipo da função é human\_time\_diff(\$from, \$to). Onde \$from é a data inicial e \$to é a data final.

get\_the\_time('U') - Retorna a data em que a publicação foi realizada. O parâmetro U é um caractere do padrão PHP para representar os segundos, desde 1 de janeiro de 1970 (Unix Epoch).

current\_time('timestamp') - Retorna a data atual e o parâmetro timestamp é a formatação padrão PHP que representa o tempo transcorrido desde, 1 de janeiro de 1970.

## POSTS

7 DE JANEIRO DE 2018 EDITAR

# PUBLICAÇÃO TESTE

É um teste de publicação para verificar o plugin.

**Publicado a 4 horas**

Caso desejar publicar os seus plugins WordPress, o seu projeto deve conter um arquivo Readme.

Conforme orientação da documentação WordPress para hospedar o seu plugin no repositório, cujo o endereço é <https://wordpress.org/plugins/> (<https://wordpress.org/plugins/>) , é preciso criar um arquivo um readme com extensão txt dentro da pasta que contém o seu plugin. O formato do arquivo deve ser o padrão estabelecido pela documentação.

Há um plugin gerador do arquivo readme no padrão estabelecido na documentação, o plugin está disponível em: [/https://generatewp.com/plugin-readme](https://generatewp.com/plugin-readme) (<https://generatewp.com/plugin-readme/>).

## PARA OBTER MAIS INFORMAÇÕES SOBRE COMO SE TORNAR UM DESENVOLVEDOR...

Acesse o link

<https://wordpress.org/plugins/developers/> (<https://wordpress.org/plugins/developers/>)

```

=== Plugin Name ===
Contributors: (this should be a list of wordpress.org userid's)
Donate link: http://example.com/
Tags: comments, spam
Requires at least: 4.6
Tested up to: 4.7
Stable tag: 4.3
License: GPLv2 or later
License URI: https://www.gnu.org/licenses/gpl-2.0.html

Here is a short description of the plugin. This should be no more than 150 characters. No markup here.

== Description ==

This is the long description. No limit, and you can use Markdown (as well as in the following sections).

For backwards compatibility, if this section is missing, the full length of the short description will be used, and
Markdown parsed.

A few notes about the sections above:

* "Contributors" is a comma separated list of wordpress.org usernames
* "Tags" is a comma separated list of tags that apply to the plugin
* "Requires at least" is the lowest version that the plugin will work on
* "Tested up to" is the highest version that you've *successfully* used to test the plugin*. Note that it might work on
higher versions... this is just the highest one you've verified.
* Stable tag should indicate the Subversion "tag" of the latest stable version, or "trunk," if you use `/trunk/` for
stable.

Note that the `readme.txt` of the stable tag is the one that is considered the defining one for the plugin, so
if the `/trunk/readme.txt` file says that the stable tag is `4.3`, then it is `/tags/4.3/readme.txt` that'll be used
for displaying information about the plugin. In this situation, the only thing considered from the trunk `readme.txt`
is the stable tag pointer. Thus, if you develop in trunk, you can update the trunk `readme.txt` to reflect changes in
your in-development version, without having that information incorrectly disclosed about the current stable version
that lacks those changes -- as long as the trunk's `readme.txt` points to the correct stable tag.

If no stable tag is provided, it is assumed that trunk is stable, but you should specify "trunk" if that's where
you put the stable version, in order to eliminate any doubt.

```

## NESSE TÓPICO VOCÊ APRENDEU SOBRE...

A criação de plugins.  
 Funções em PHP.  
 Hooks ou ganchos no WordPress.  
 Ações e filtros no WordPress  
 Padrões e boas práticas de desenvolvimento de plugins.

## ATIVIDADE

### O que são plugins?

- A. São recursos nativos do WordPress que não podem ser adicionados.
- B. São instruções PHP que ficam no núcleo de funcionamento do WordPress.
- C. São recursos desenvolvidos pela comunidade WordPress e que realizam tarefas específicas.
- D. São instruções CSS que ficam no núcleo do WordPress.

## ATIVIDADE

Veja as afirmações a seguir:

I) Hooks são plugins nativos do WordPress.

II) As ações e filtros manipulam informações, mudando um comportamento específico do WordPress.

III) Não é possível criar plugins, além daqueles disponíveis no repositório oficial.

Qual é alternativa correta?

- A. As três afirmações são falsas.
- B. A afirmação I é verdadeira.
- C. As afirmações I e III são verdadeiras.
- D. A afirmação II é a única correta.

## ATIVIDADE

O filtro `the_comment` é usado para

- A. Tratar informações de posts.
- B. Tratar informações de título de posts.
- C. Tratar comentários dos posts.
- D. Tratar contagem de palavras.

## REFERÊNCIA

WORDPRESS. Plugin HandBook. Disponível em: <<https://developer.wordpress.org/plugins/hooks/>>. Acesso em 05 de jan. 2018.

HENDENGREN, THORD DANIEL. Smashing WordPress: Além do blog. Porto Alegre: Bookman, 2012. 336 p.

NIEDERAUER, JULIANO, Desenvolvendo Websites com PHP. São Paulo: Novatec, 2017. 315 p.



