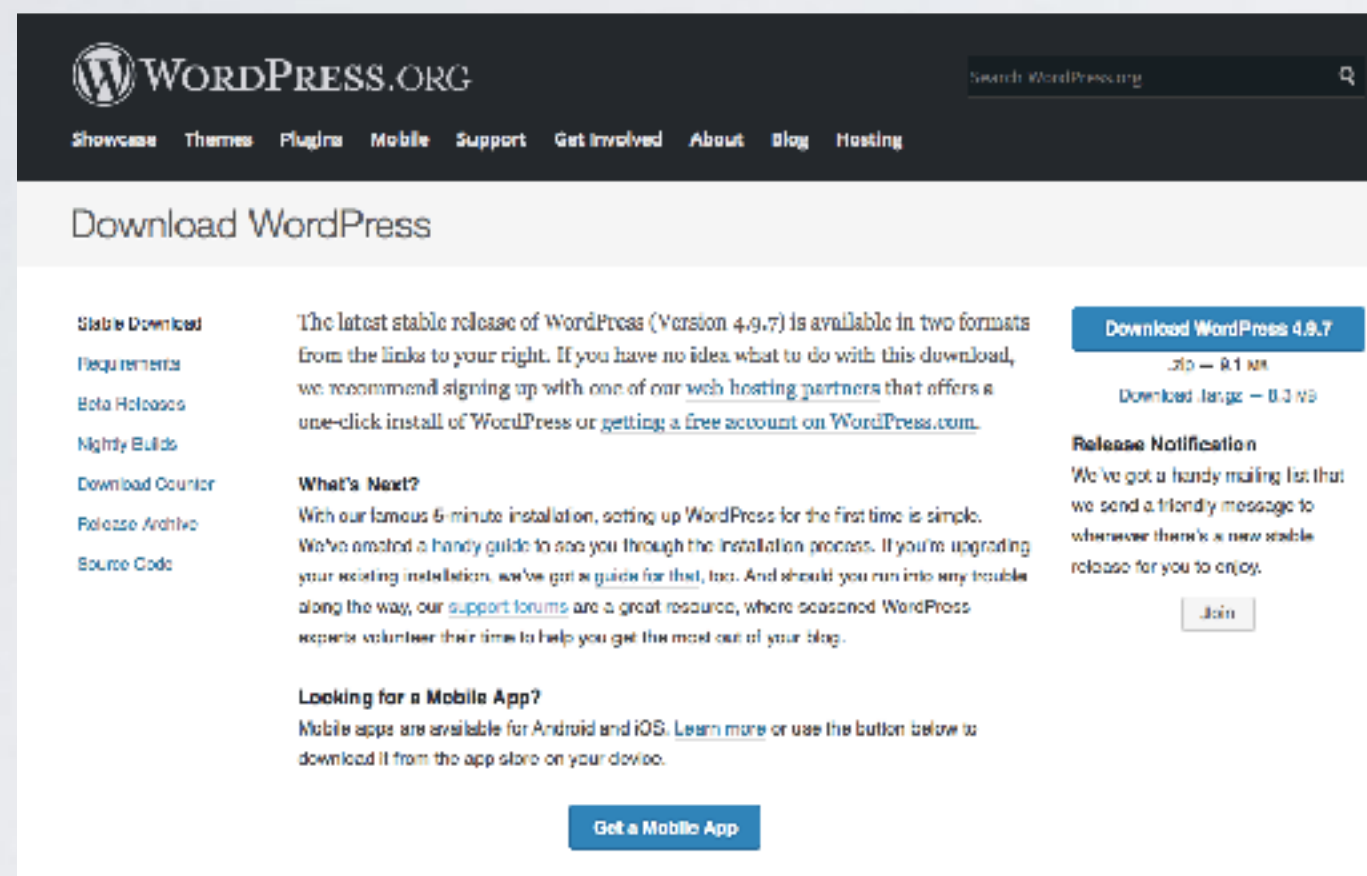




**CREAR TEMA DE WORDPRESS**

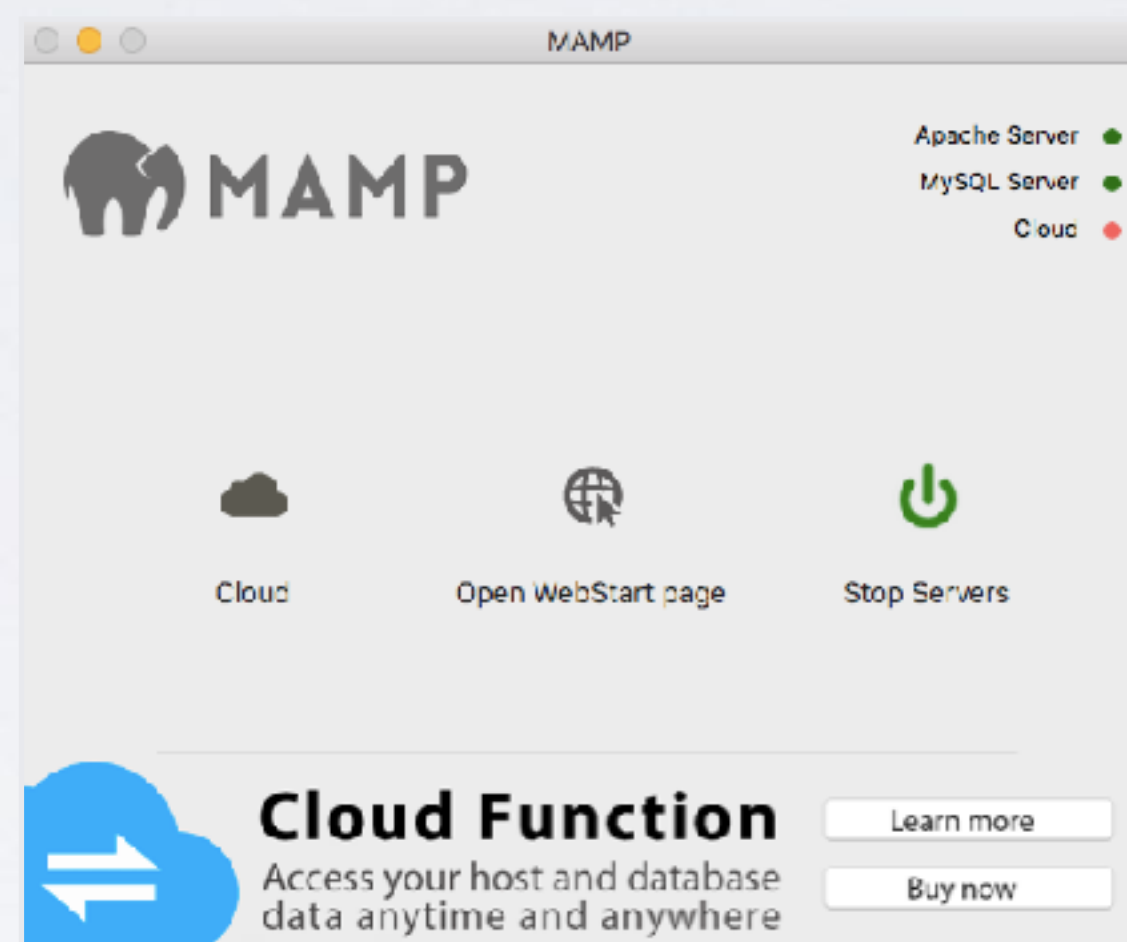
Antes de crear un tema de Wordpress, necesitaremos:

Wordpress



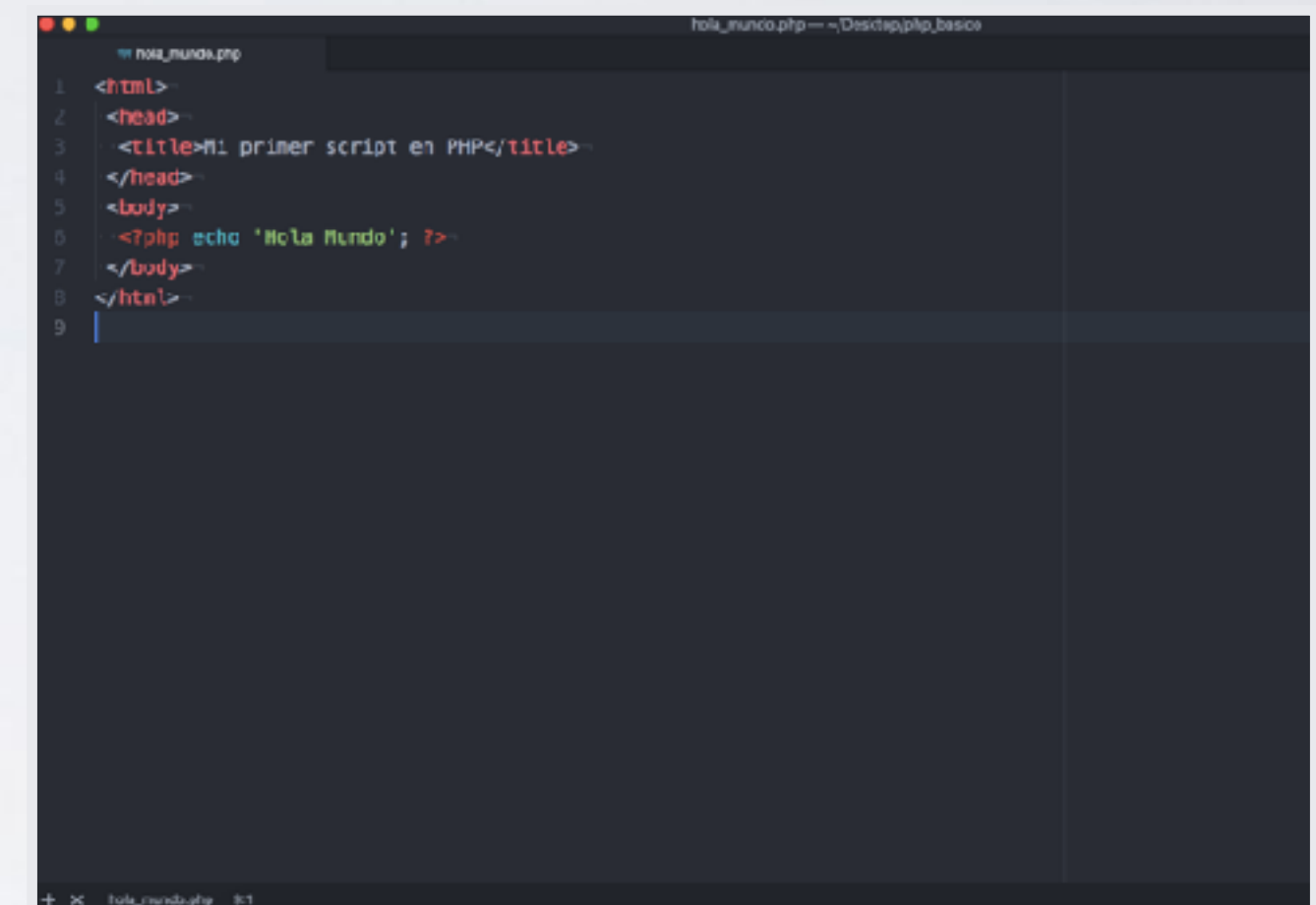
Descargar instalación

MAMP



Instalar Wordpress

Editor de texto



Activar modo debug

El siguiente paso luego de instalar Wordpress en nuestro computador es  
activar el modo debug

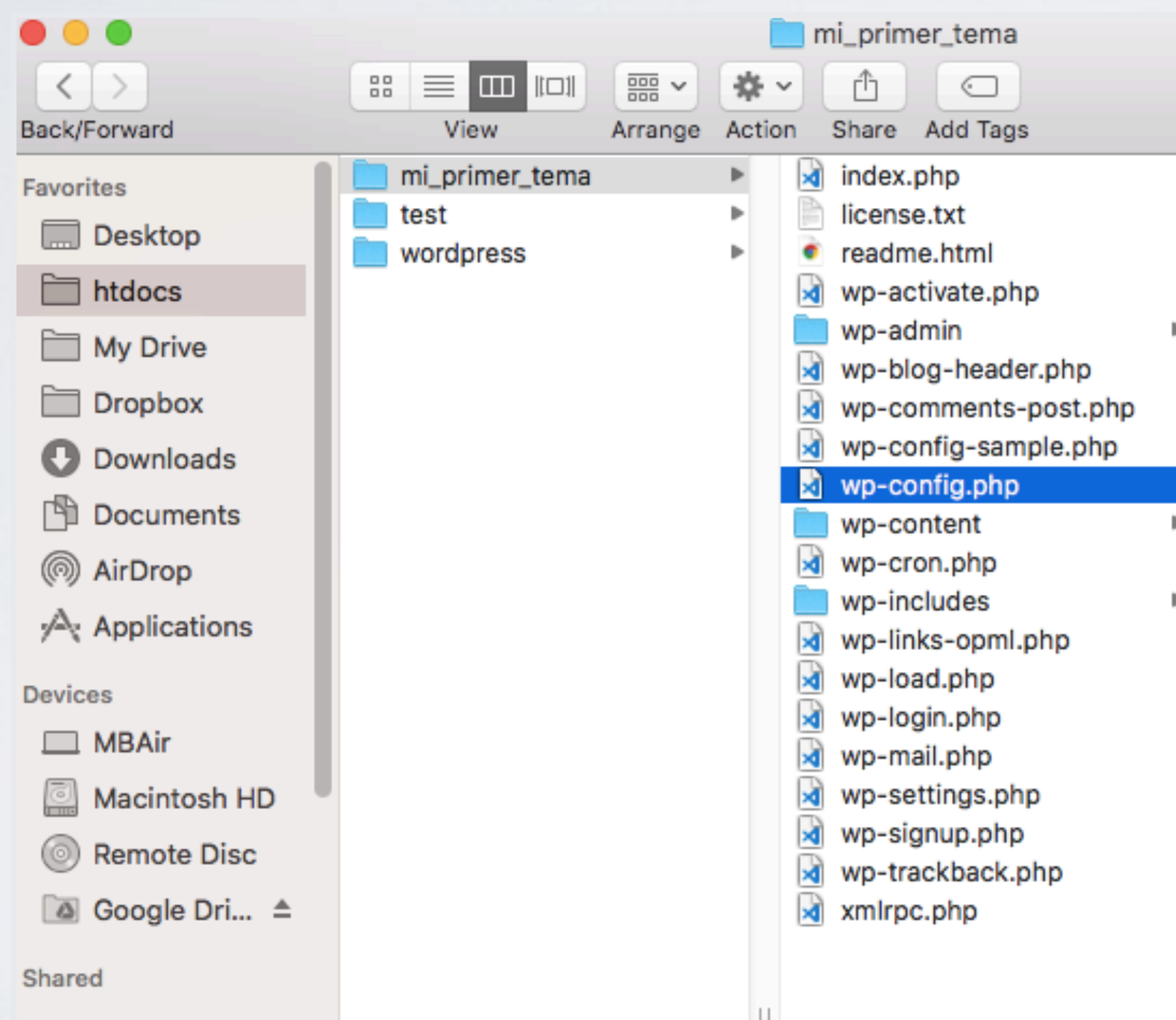
# ¿QUÉ ES EL MODO DEBUG?

El modo debug es una herramienta de depuración incluida dentro Wordpress que nos ayudará a encontrar problemas en nuestro código



# ¿CÓMO ACTIVAR EL MODO DEBUG?

Para activarlo debemos:

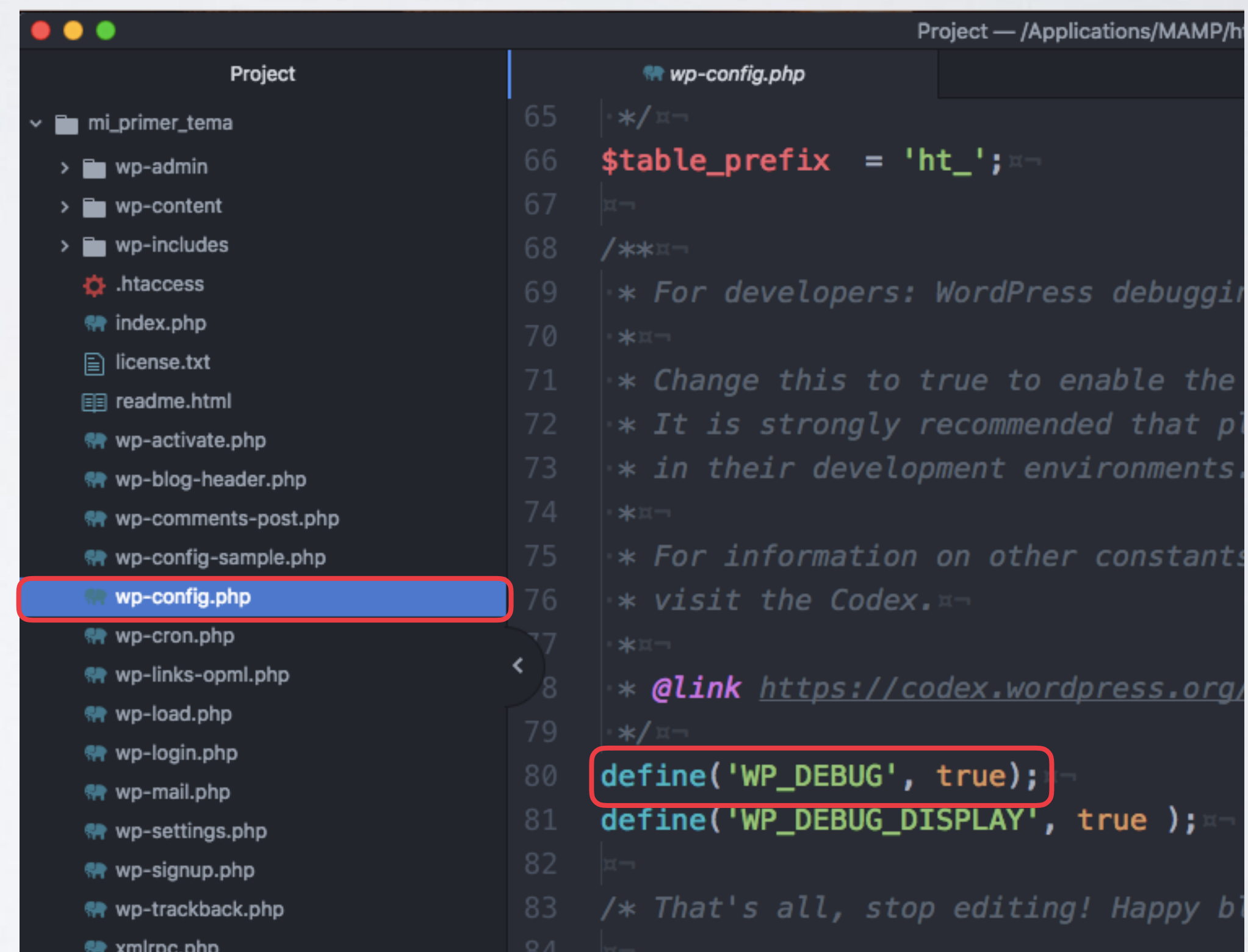


1. Buscar la instalación de Wordpress e ingresar a ella
2. Elegir archivo llamado **wp-config.php**
3. Abrir archivo en el editor de texto

# ¿CÓMO ACTIVAR EL MODO DEBUG?

Para activarlo debemos:

1. Hacer scroll hasta encontrar el texto: **For developers: WordPress debugging mode.**
2. Ir a **WP\_DEBUG**
3. Cambiar el **booleano false** por **true**



```
Project — /Applications/MAMP/h
Project
  mi_primer_tema
    wp-admin
    wp-content
    wp-includes
    .htaccess
    index.php
    license.txt
    readme.html
    wp-activate.php
    wp-blog-header.php
    wp-comments-post.php
    wp-config-sample.php
    wp-config.php
    wp-cron.php
    wp-links-opml.php
    wp-load.php
    wp-login.php
    wp-mail.php
    wp-settings.php
    wp-signup.php
    wp-trackback.php
    xmlrpc.php
  wp-config.php
65  /*
66  $table_prefix = 'ht_';
67
68  /**
69  * For developers: WordPress debugging mode.
70  *
71  * Change this to true to enable the display of WordPress errors on the screen. It is strongly recommended that you set this to true in your development environments.
72  *
73  * For information on other constants and how you can use them, see the Codex.
74  *
75  * @link https://codex.wordpress.org/Development/Debugging_in_WordPress
76  */
77
78  /**
79  *
80  * define('WP_DEBUG', true);
81  * define('WP_DEBUG_DISPLAY', true );
82
83  /* That's all, stop editing! Happy blogging */
84
```

# MODOS DE WP\_DEBUG

Podemos habilitar tres modos usando **WP\_DEBUG**

Habilita la depuración del código

```
define("WP_DEBUG", true);
```

Habilita la visualización del código  
en el HTML

```
define("WP_DEBUG_DISPLAY", true);
```

Habilita el guardado de los errores en un archivo .log

```
define("WP_DEBUG_LOG", true);
```

Con la activación del modo debug ya estamos preparados para comenzar a crear nuestro primer tema de Wordpress





# CREAR UN TEMA DE WORDPRESS

Antes de comenzar, es bueno conocer la diferencia que existe entre un tema  
y una plantilla

# TEMA V/S PLANTILLA



**Plantilla**

Es solo un diseño de página que está disponible dentro de un tema

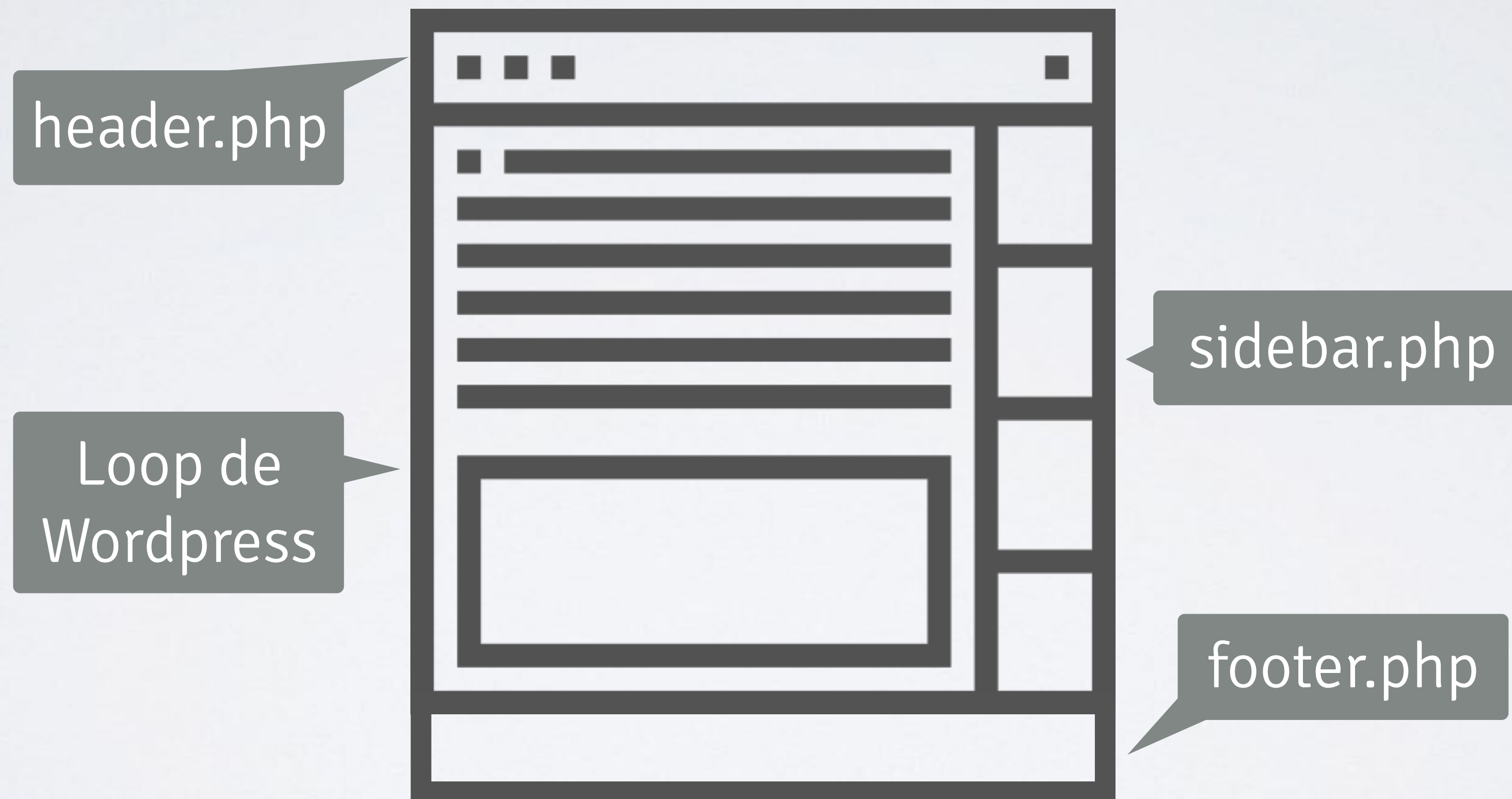


**Tema**

Está conformado por un conjunto de plantillas que conforman un sitio web

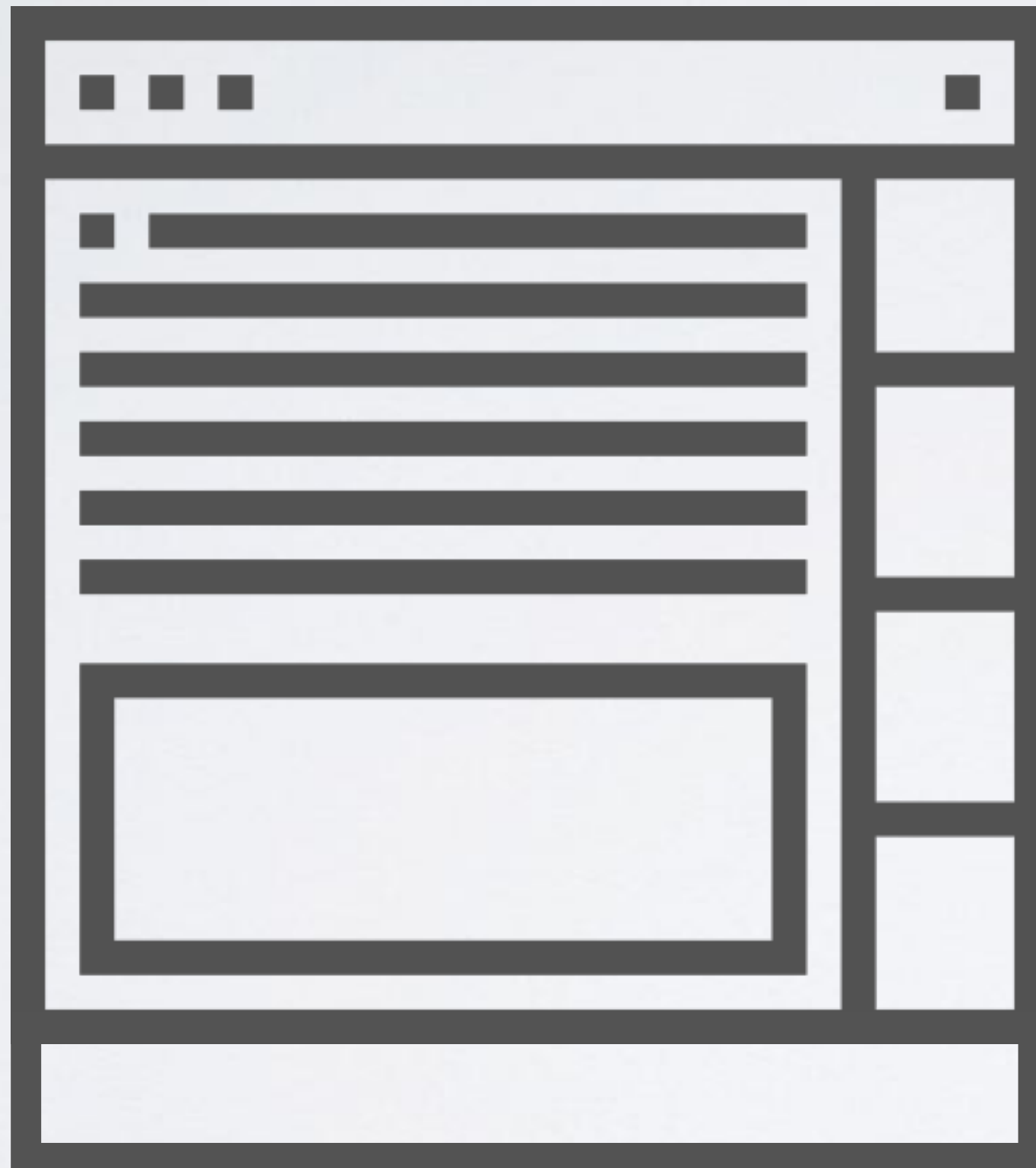
# ANATOMÍA DE UN TEMA DE WORDPRESS

Una página estándar de Wordpress está conformado por:



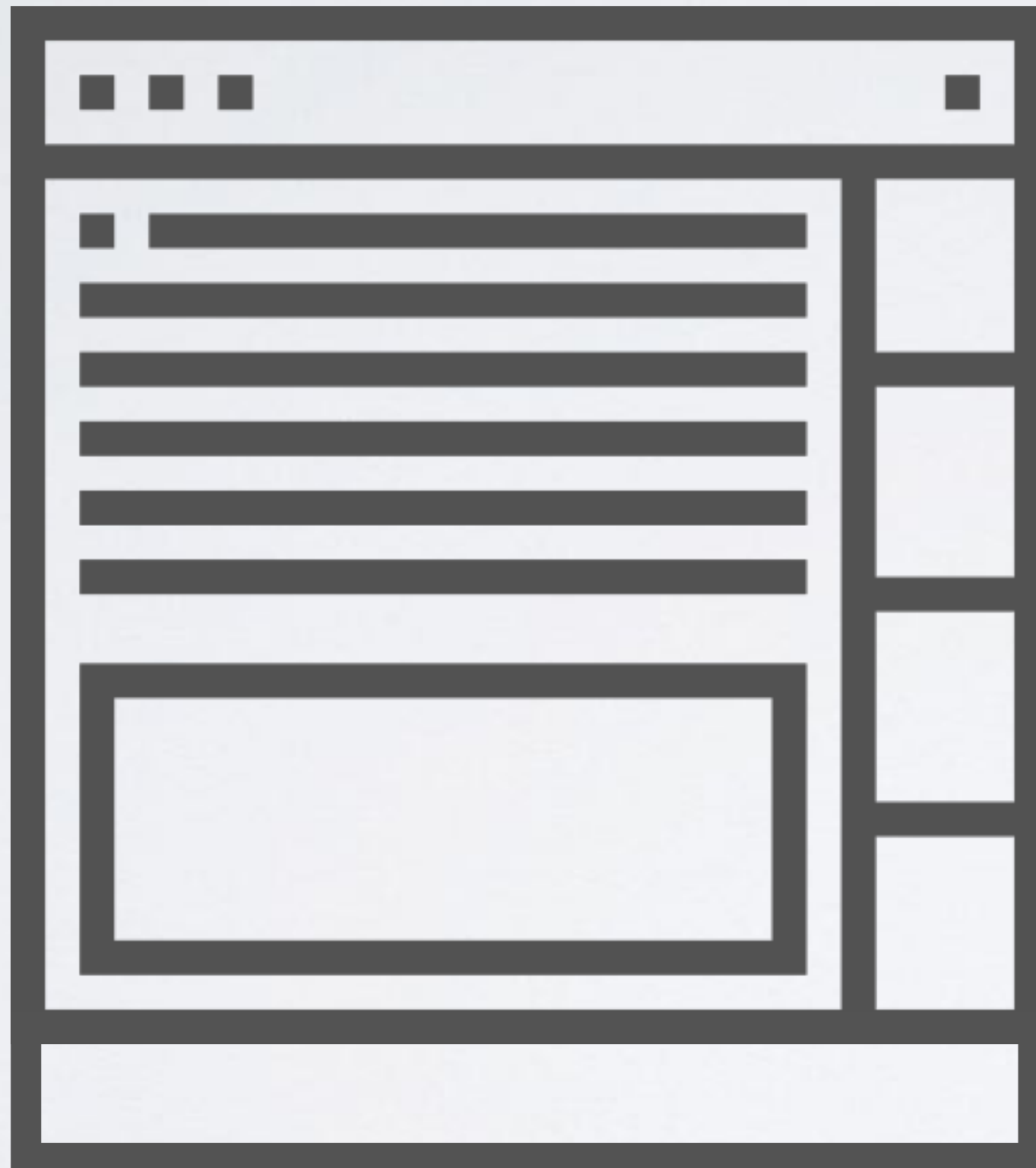


# INDEX.PHP



- El archivo index controla cómo se verá la página de inicio
- Por defecto, contiene un loop el cuál mostrará las últimas entradas de nuestro sitio
- Esto puede ser cambiado desde el administrador

# SINGLE.PHP



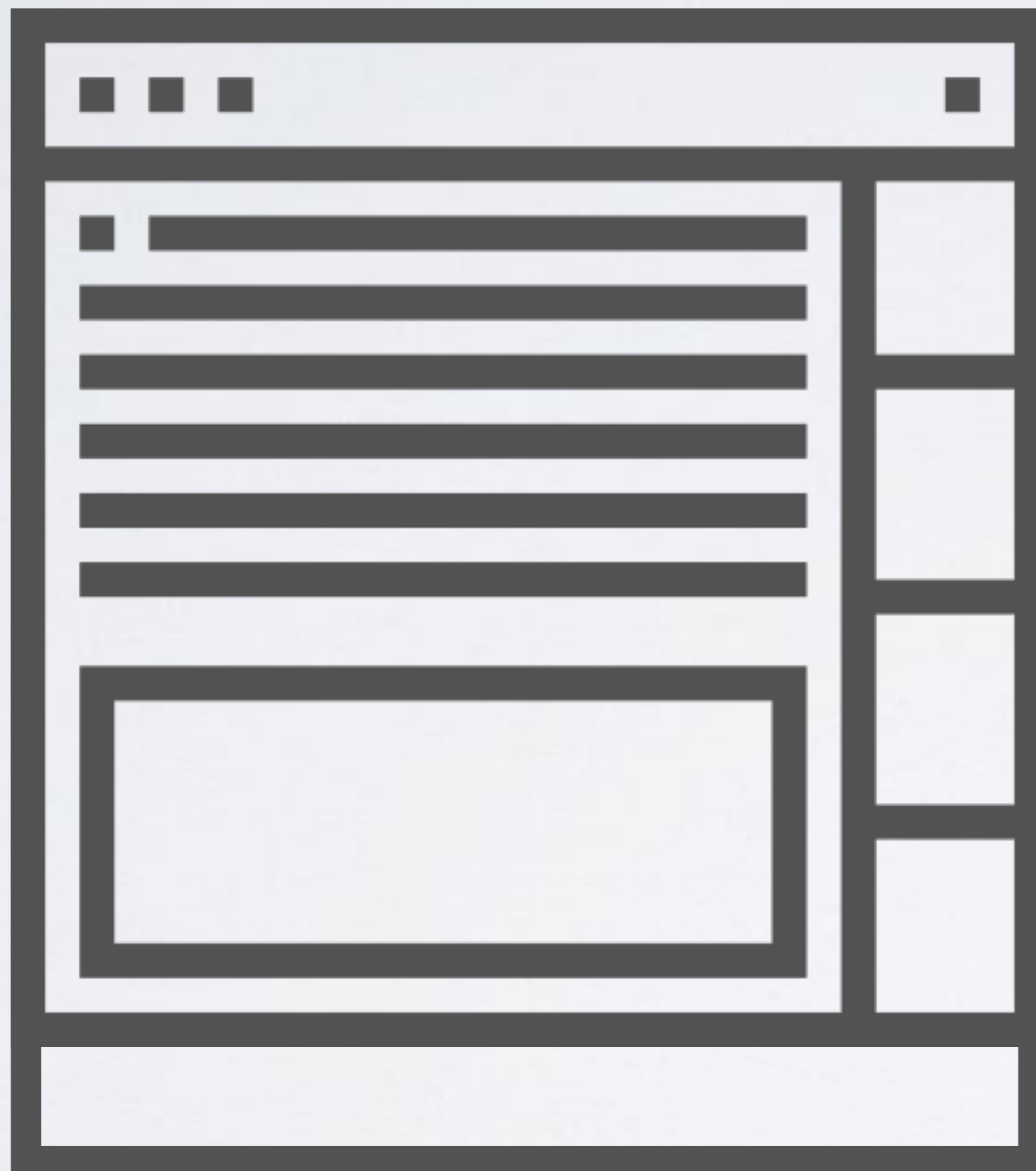
- Archivo que muestra una entrada de manera individual

# PAGE.PHP



- controla cómo se ven las páginas.

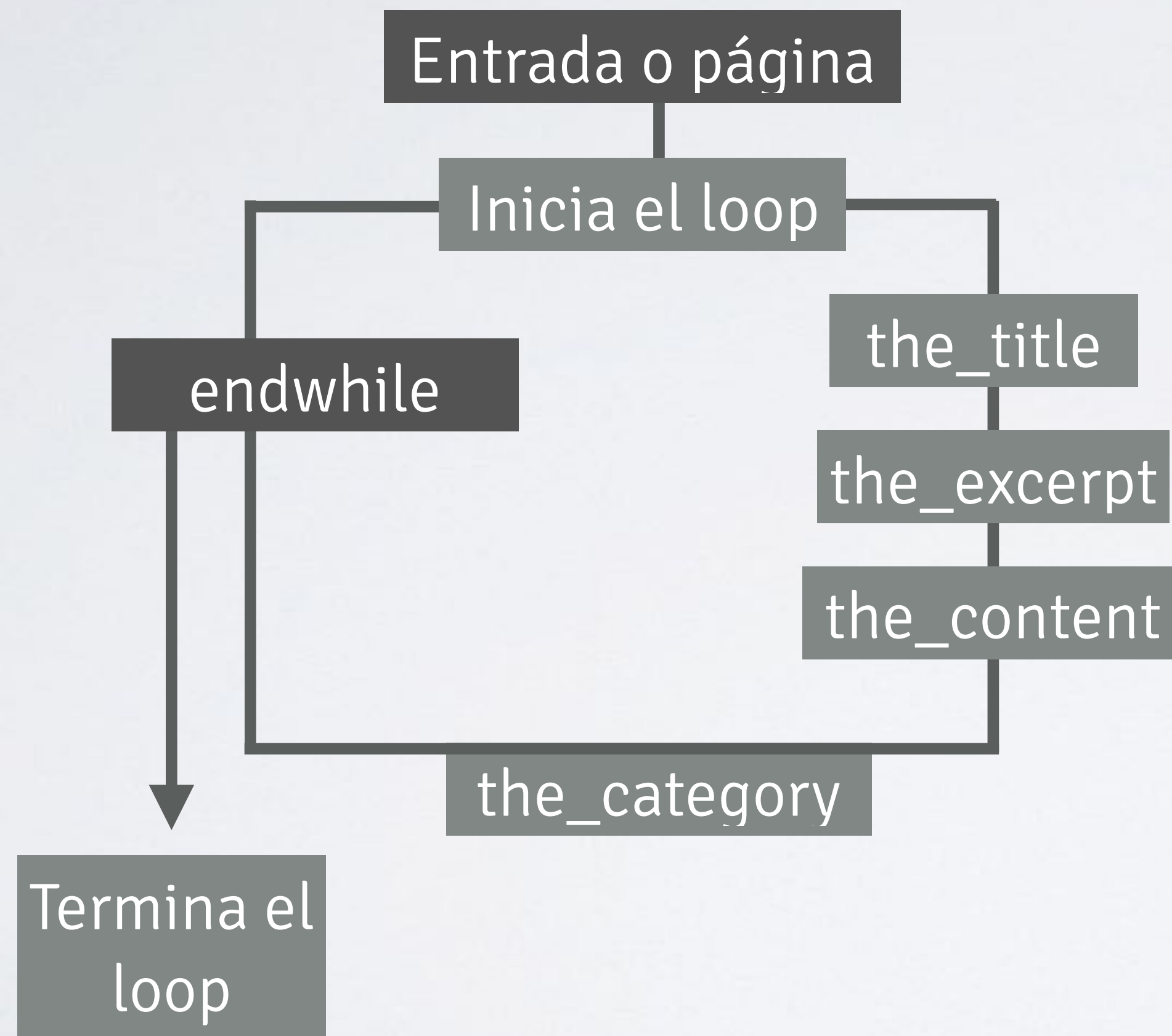
# ARCHIVE.PHP, CATEGORY.PHP, TAG.PHP



- Páginas de archivos, donde se despliegan las entradas categorizadas por categorías o etiquetas.



# LOOP DE WORDPRESS



- Es una de las partes más poderosas de un tema de Wordpress
- Con podemos determinar que página o post queremos visualizar

Además de esos archivos, existen dos archivos muy importantes que son necesarios para comenzar a crear un tema

# ARCHIVOS NECESARIOS PARA CREAR UN TEMA



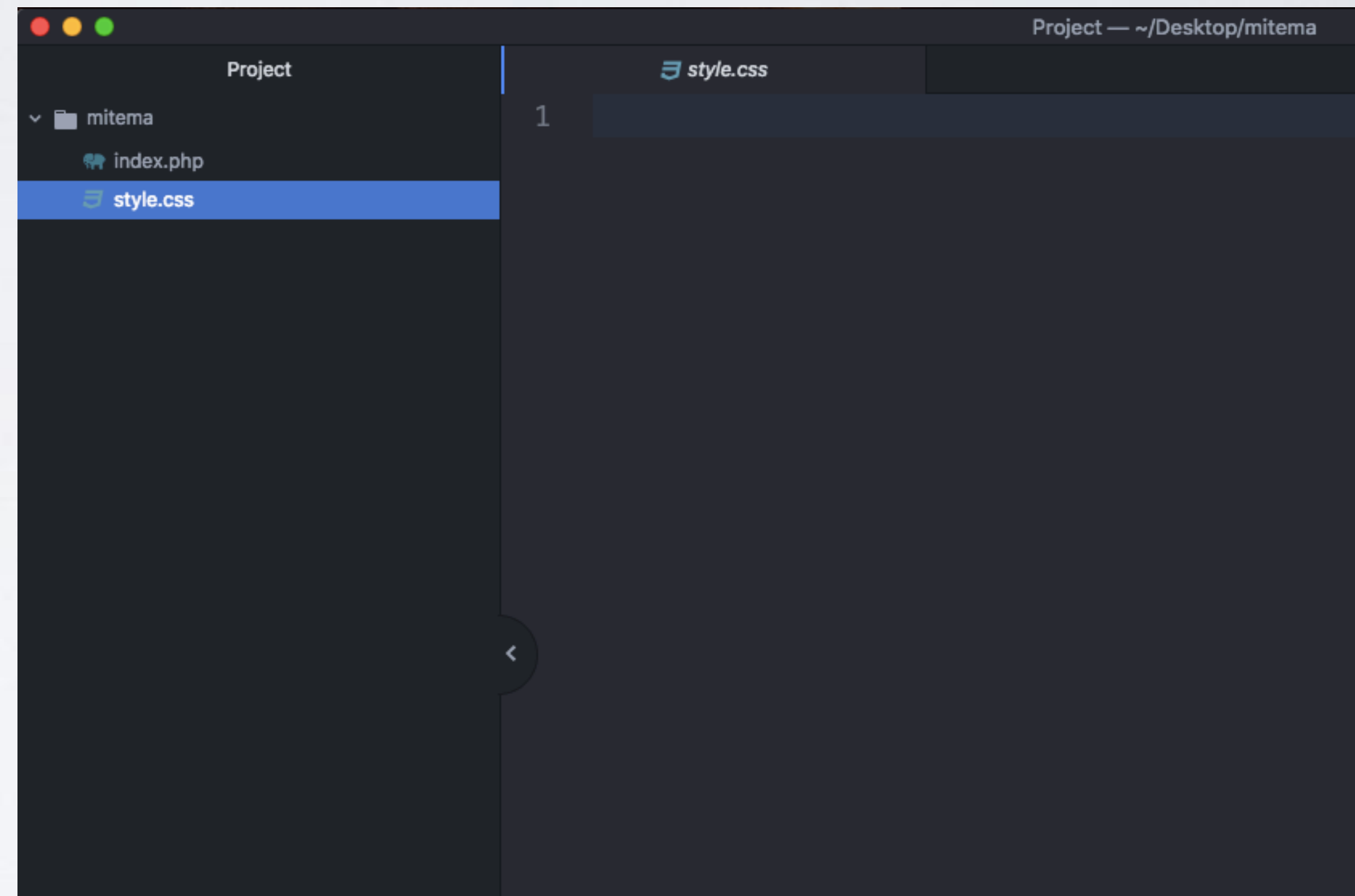
index.php



style.css

# CREAR TEMA DE WORDPRESS

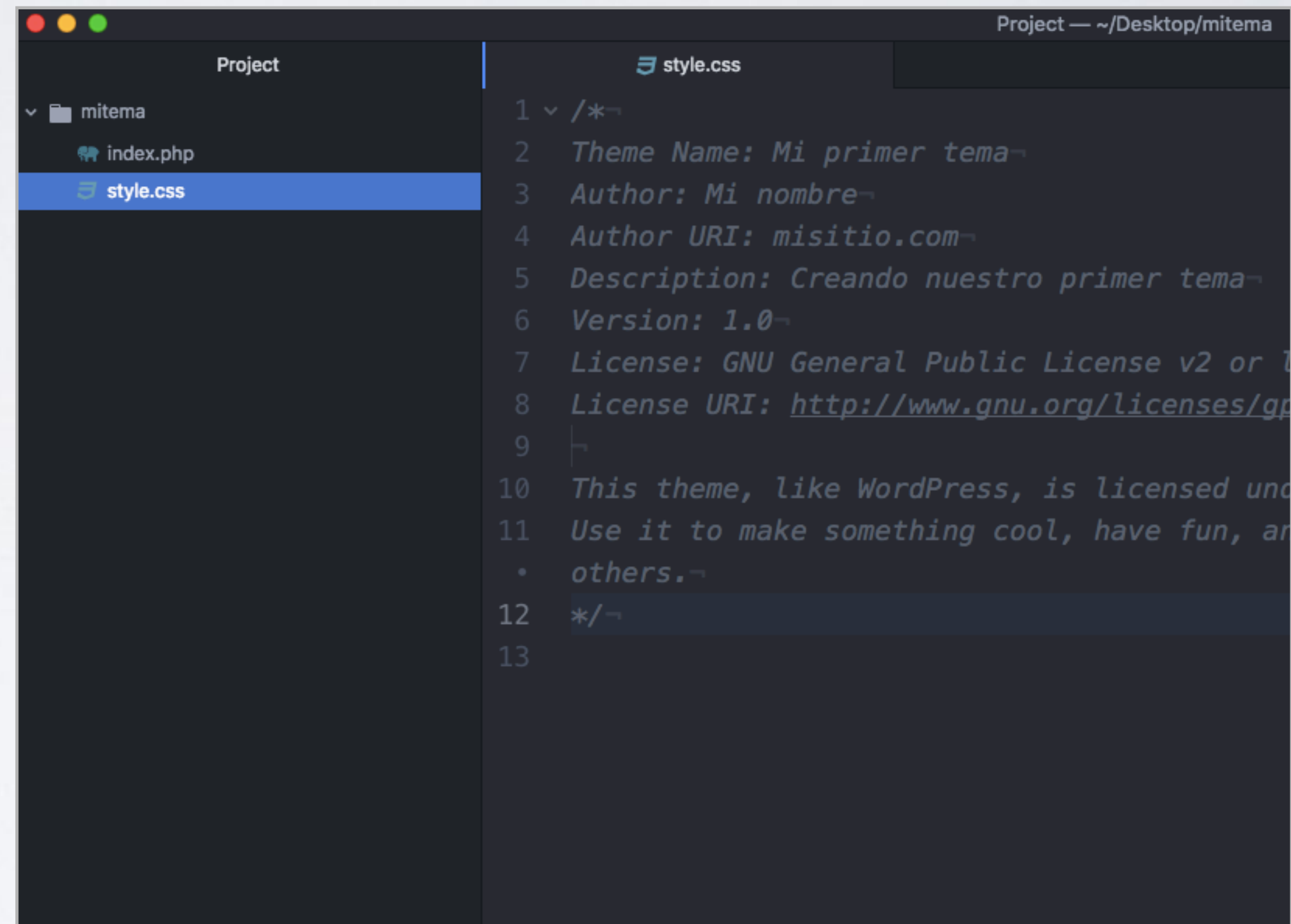
- Crear una carpeta con el nombre del tema
- Abrir carpeta en el editor de texto
- Crear los archivos **index.php** y **style.css**





# STYLE.CSS

- Contiene toda la descripción necesaria de tu tema.
- En él se proporciona la información y detalles del tema a modo de comentarios en la cabecera
- Estos comentarios dan la posibilidad de que el tema sea identificado por el panel de administración de WordPress



```
1  /*
2   Theme Name: Mi primer tema
3   Author: Mi nombre
4   Author URI: misitio.com
5   Description: Creando nuestro primer tema
6   Version: 1.0
7   License: GNU General Public License v2 or later
8   License URI: http://www.gnu.org/licenses/gpl-2.0.html
9
10  This theme, like WordPress, is licensed under the GPL
11  Use it to make something cool, have fun, and share
12  with others.
13  */
```

Dentro de estos comentarios se pueden agregar muchas características

```
/*
```

```
Theme Name: nombre del tema.
```

```
Theme URI: Sitio web donde podremos encontrar información del tema.
```

```
Author: empresa o persona que ha desarrollado el tema.
```

```
Author URI: página web de la empresa o persona que ha desarrollado el tema.
```

```
Version: versión actual del tema.
```

```
License: tipo de licencia que se aplica al tema.
```

```
Tags: etiquetas para ayudar a identificar el tema, señalando algunas de sus características, como las posiciones disponibles para añadir widgets, los colores disponibles, personalizaciones que permite, etc.
```

```
Text Domain: identificador único con el que WordPress puede distinguir entre todas las traducciones de constantes de idioma subidas.
```

```
*/
```

# INDEX.PHP

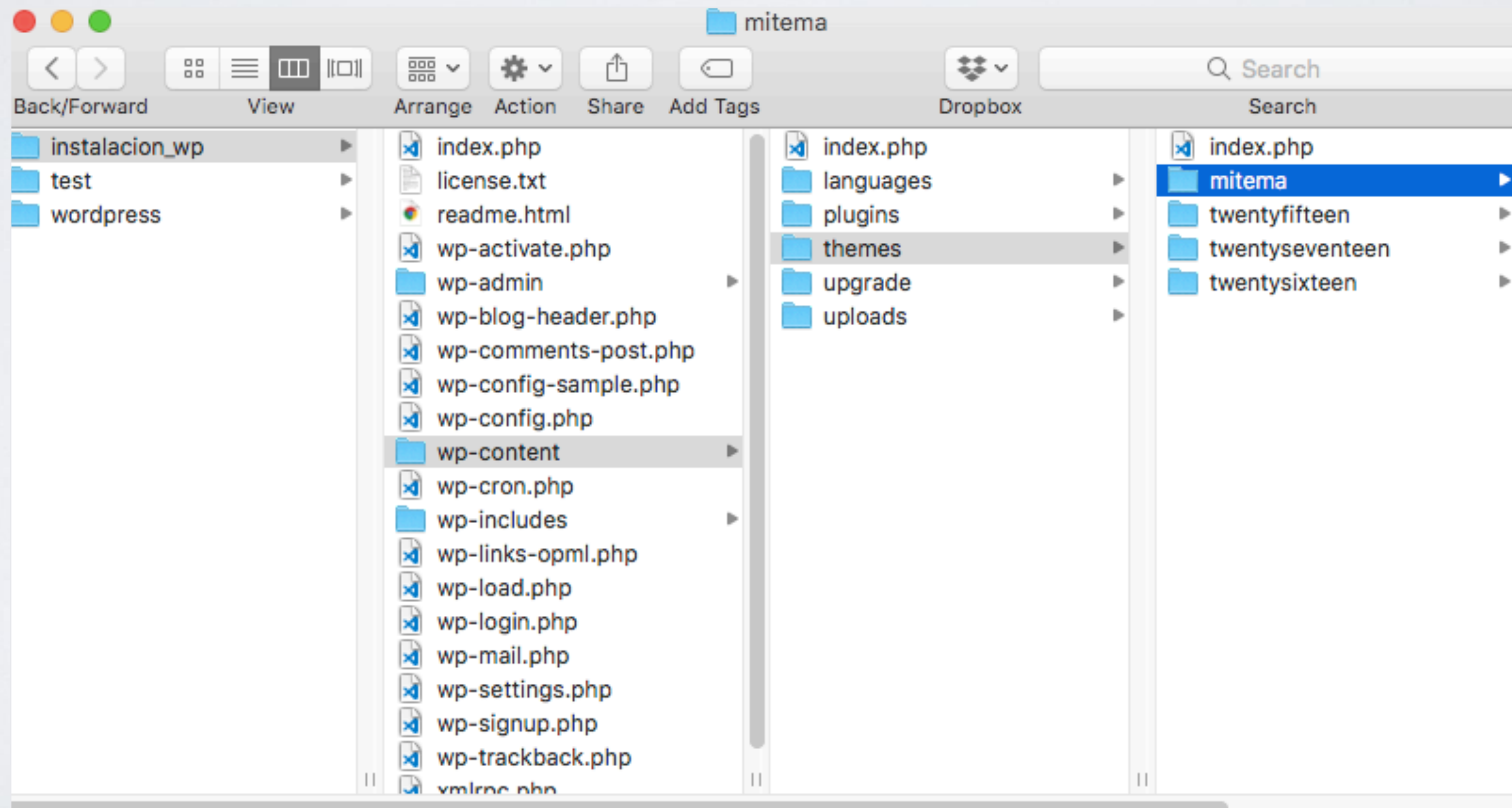
El archivo index controla cómo se verá la página de inicio

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Mi primer tema</title>
</head>
<body>
  <h1>iHola Mundo!</h1>
</body>
</html>
```

Podemos agregar contenido en HTML y PHP dentro de él

# ACTIVAR TEMA

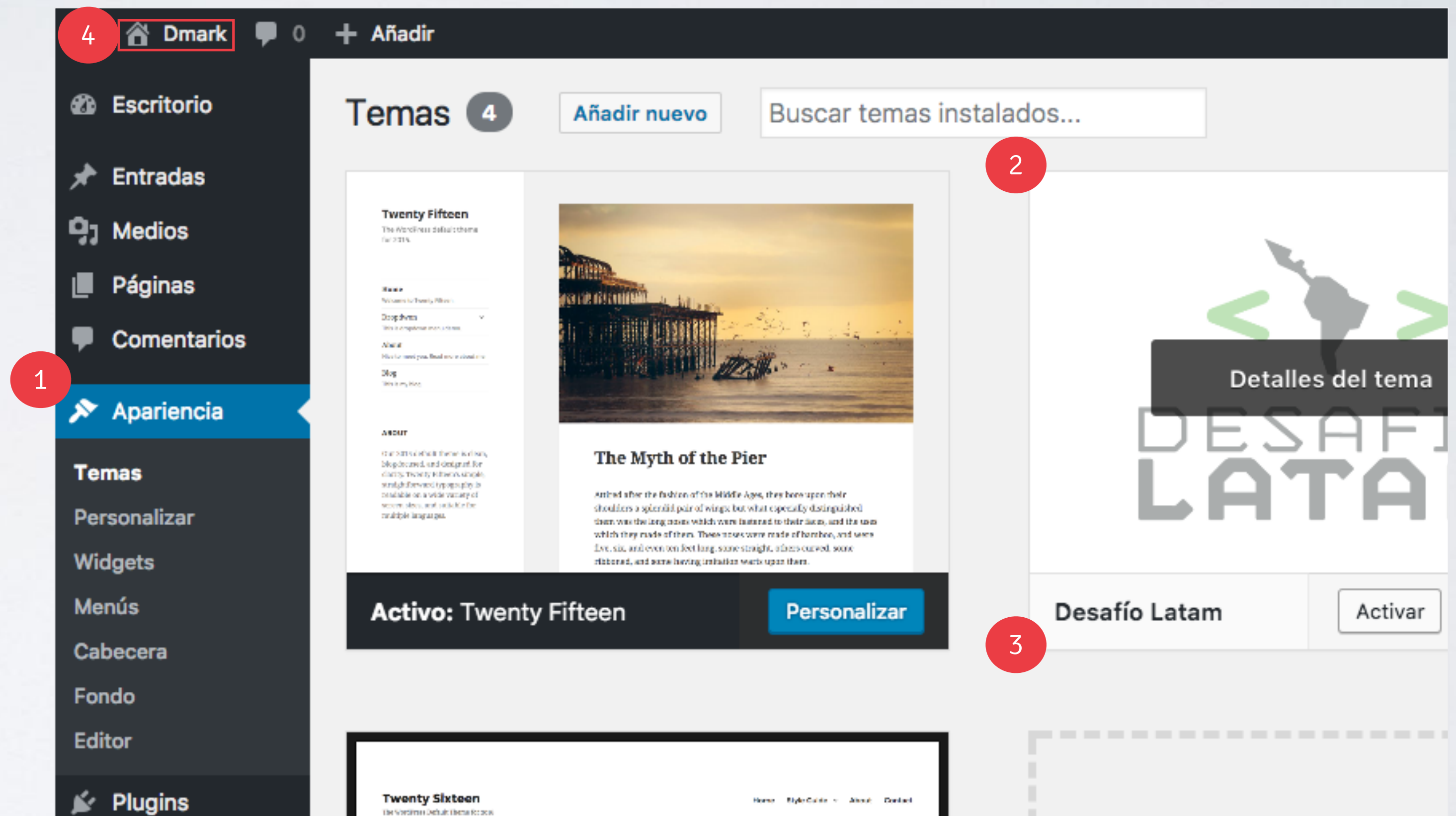
Primero debemos mover nuestro tema a la carpeta themes dentro de la instalación de Wordpress





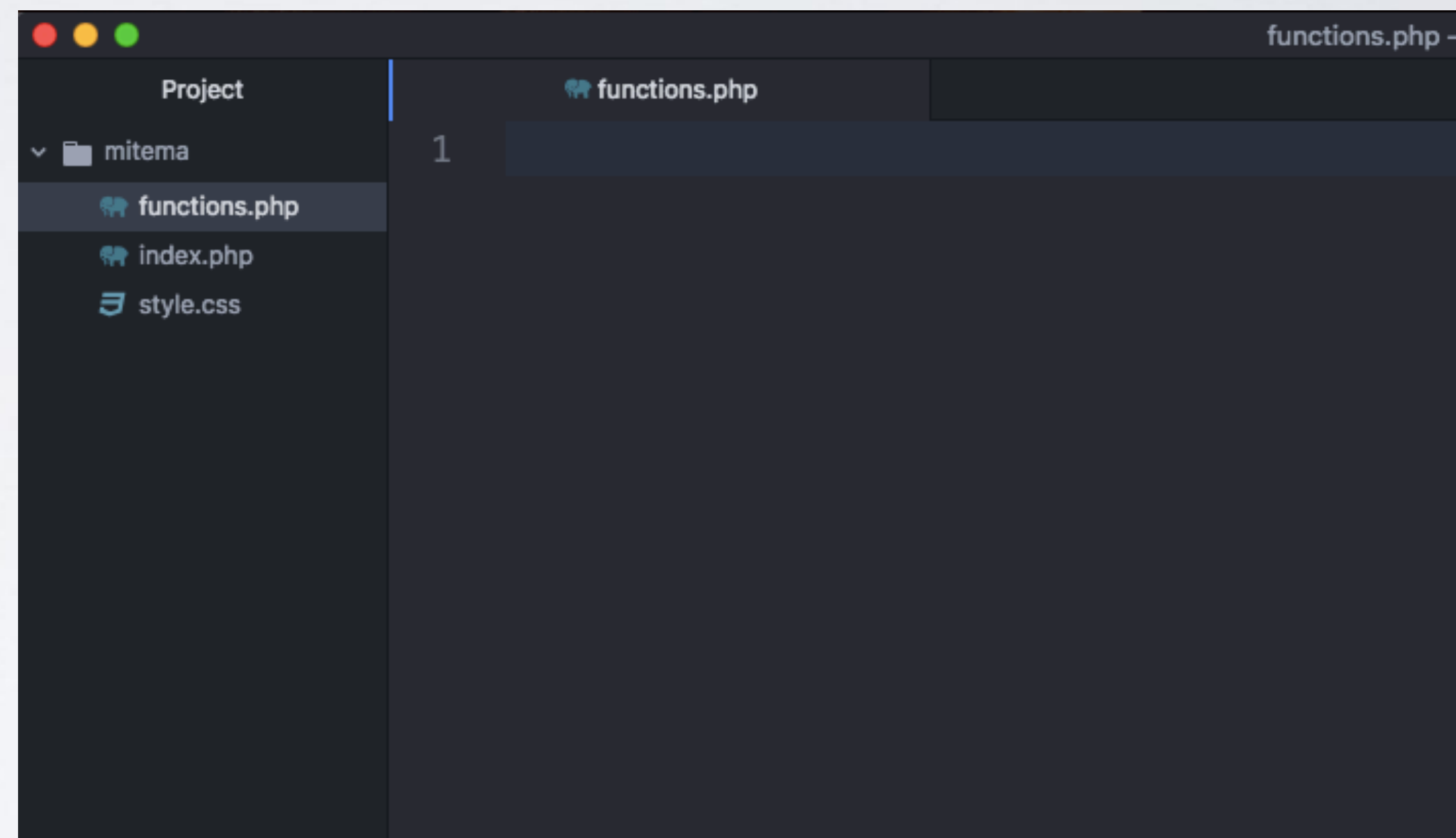
# ACTIVAR TEMA

- Ir a **Wordpress**
- Ingresar a **apariencia**
- Dentro de temas, busca el **nombre del tema**
- **Activar** tema
- Presionar **ver sitio**



# FUNCTIONS.PHP

- Archivo donde se añaden funciones en PHP para modificar elementos de tu tema.
- En él se pueden añadir todo tipo de funcionalidades.
- Se puede llamar a los estilos CSS y scripts de JavaScript, entre otras cosas.



# AÑADIR ESTILOS A FUNCTIONS.PHP

Para añadir estilos CSS a un tema debemos agregar el siguiente código en functions.php:

# WP\_REGISTER\_STYLE()

Función de Wordpress usada para registrar estilos css

The diagram illustrates the `wp_register_style` function with callouts for its parameters:

- Nombre del estilo**: Points to the `$handle` parameter.
- ¿Depende de un archivo?**: Points to the `$src` parameter.
- Tipo de medio**: Points to the `$media` parameter.
- Versión**: Points to the `$ver` parameter.
- Lugar donde se encuentra**: Points to the `$deps` parameter.

```
<?php
wp_register_style( $handle, $src, $deps, $ver, $media );
?>
```



# WP\_REGISTER\_STYLE()

Función de Wordpress usada para registrar estilos css

```
<?php  
  
wp_register_style( $handle, $src, $deps, $ver,  
$media );  
  
?>
```

- **\$handle:** Nombre que le pondremos al archivo de estilo
- **\$src:** Dónde está ubicado el archivo
- **\$deps:** Si el CSS depende de otro archivo, escribir el **\$handle** del archivo que depende
- **\$ver:** Número de versión
- **\$media:** especifica qué tipo de media va a cargar este css: ***'all', 'screen', 'print' o 'handheld'.***



# WP\_ENQUEUE\_STYLE()

La función **wp\_enqueue\_style()** sirve para dejar en cola un archivo

```
<?php  
  
wp_enqueue_style($handle);  
  
?>
```

Podemos encapsular **wp\_enqueue\_style()** y **wp\_enqueue\_style()** dentro de una función creada por nosotros mismos

```
<?php

function register_enqueue_style() {

    /* Registrando estilos */
    wp_register_style( $handle, $src, $deps, $ver, $media );

    /* Enqueue estilos */
    wp_enqueue_style($handle);
}

?>
```

# ADD\_ACTION()

Función que nos ayudará a invocar nuestra función cuando se active una acción en Wordpress

```
<?php
```

Función a ser invocada

```
add_action( $action, $function );
```

```
?>
```

Nombre de acción

# EJEMPLO

```
<?php

function register_enqueue_style() {

    /* Registrando estilos */

    wp_register_style('main', get_theme_file_uri('/assets/stylesheets/styles.css'),
    'normalize', '1.0', 'screen');

    /* Estilos en cola */
    wp_enqueue_style('main');

}

add_action( 'wp_enqueue_scripts', 'register_enqueue_style' );

?>
```

# AÑADIR SCRIPTS A FUNCTIONS.PHP

Para añadir scripts de Javascript a un tema debemos agregar el siguiente código en functions.php:



# WP\_ENQUEUE\_SCRIPT()

Función de Wordpress usada para registrar los script de Javascript

```
<?php
```

```
wp_register_script( $handle, $src, $deps, $ver, $in_footer );
```

```
?>
```

Nombre del script

¿Depende de un archivo?

Carga de script en footer

Lugar donde se encuentra

Versión

# WP\_ENQUEUE\_SCRIPT()

Función de Wordpress usada para registrar scripts de Javascript

```
<?php  
  
wp_register_script( $ha  
ndle, $src, $deps,  
$ver, $in_footer );  
  
?>
```

- **\$handle:** Nombre que le pondremos al archivo de estilo
- **\$src:** Dónde está ubicado el archivo
- **\$deps:** Si el script depende de otro archivo. Escribir el **\$handle** del archivo que depende
- **\$ver:** Número de versión
- **\$in\_footer:** especificar si el script será cargado de manera normal o debajo del footer

# WP\_ENQUEUE\_STYLE()

La función **wp\_enqueue\_script()** sirve para dejar en cola un archivo

```
<?php  
  
wp_enqueue_script($handle);  
  
?>
```

Podemos encapsular el **wp\_enqueue\_script()** dentro de una función creada por nosotros mismos. Podemos usar al igual que los estilos la función **add\_action()**

```
<?php

function register_enqueue_script() {

    /* Registrando scripts */
    wp_register_script( $handle, $src, $deps, $ver, $in_footer );

    /* Enqueue script */
    wp_enqueue_script($handle);
}

add_action( $action, $function );

?>
```

# EJEMPLO

```
<?php

function register_enqueue_scripts() {

    /* Registrando Scripts */
    wp_register_script('mainJS', get_theme_file_uri('/assets/javascripts/
scripts.js'), array('jQuery3'), null, true);

    /* Enqueue Scripts */
    wp_enqueue_script('mainJS');

}

add_action( 'wp_enqueue_scripts', 'register_enqueue_scripts' );

?>
```



# LLAMAR ARCHIVOS DE FUNCTIONS.PHP

Para llamar a los archivos registrados desde en el archivo **functions.php**, debemos usar las funciones **get\_header()** y **get\_footer()**

```
<?php get_header() ?>  
  
<!-- Contenido aquí -->  
  
<?php get_footer() ?>
```

# ¿QUÉ ES GET\_HEADER Y GET\_FOOTER?

Son funciones las cuales llaman archivos parciales, que nos proporcionaran código que será ejecutado al inicio y al final de nuestro sitio.

# HEADER.PHP

Es un archivo en el cual podremos agregar la cabecera de nuestra página

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
  <head>
    <meta charset="<?php bloginfo('charset') ?>">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <title><?php bloginfo('name'); ?></title>
    <?php wp_head() ?>
  </head>
<body>
```

# WP\_HEAD()

Función que **llamará** todos los estilos y scripts agregados en **functions.php**. Estos deben ser agregados dentro de la etiqueta **<head>**

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
  <head>
    <meta charset="<?php bloginfo('charset') ?>">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <title><?php bloginfo('name'); ?></title>
    <?php wp_head() ?>
  </head>
<body>
```

# BLOGINFO

función que nos ayudará a obtener información relacionada al sitio como charset, título, descripción, url, idioma, entre otras cosas

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
  <head>
    <meta charset="<?php bloginfo('charset') ?>">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <title><?php bloginfo('name'); ?></title>
    <?php wp_head() ?>
  </head>
<body>
```



# FOOTER.PHP

```
<?php wp_footer() ?>  
</body>  
</html>
```

Es un archivo en el cual podremos agregar el footer de nuestra página

# WP\_FOOTER()

```
<?php wp_footer() ?>  
</body>  
</html>
```

Función que **llamará** todos los estilos y scripts en los que se especifique su carga en el footer

# AGREGAR UNA IMAGEN

Al usar una imagen dentro de un tema es probable que nos encontremos con el siguiente problema:

```

```

```
<!-- La dirección de la imagen no se encuentra -->
```

Para solucionarlo podremos usar:

## GET\_THEME\_FILE\_URI()

Esta función nos ayudará a obtener la url del tema.

```

```

# CONDITIONAL TAGS

Los conditional tags son etiquetas que ejecutan bloques de código sólo si se cumple la condición que presentan, por ejemplo:

```
<?php if (is_home()) : ?>
    <h2><?php the_title() ?></h2>
    <p><?php the_content() ?></p>
<?php else : ?>
    <h2>La página no se encuentra</h2>
<?php endif; ?>
```



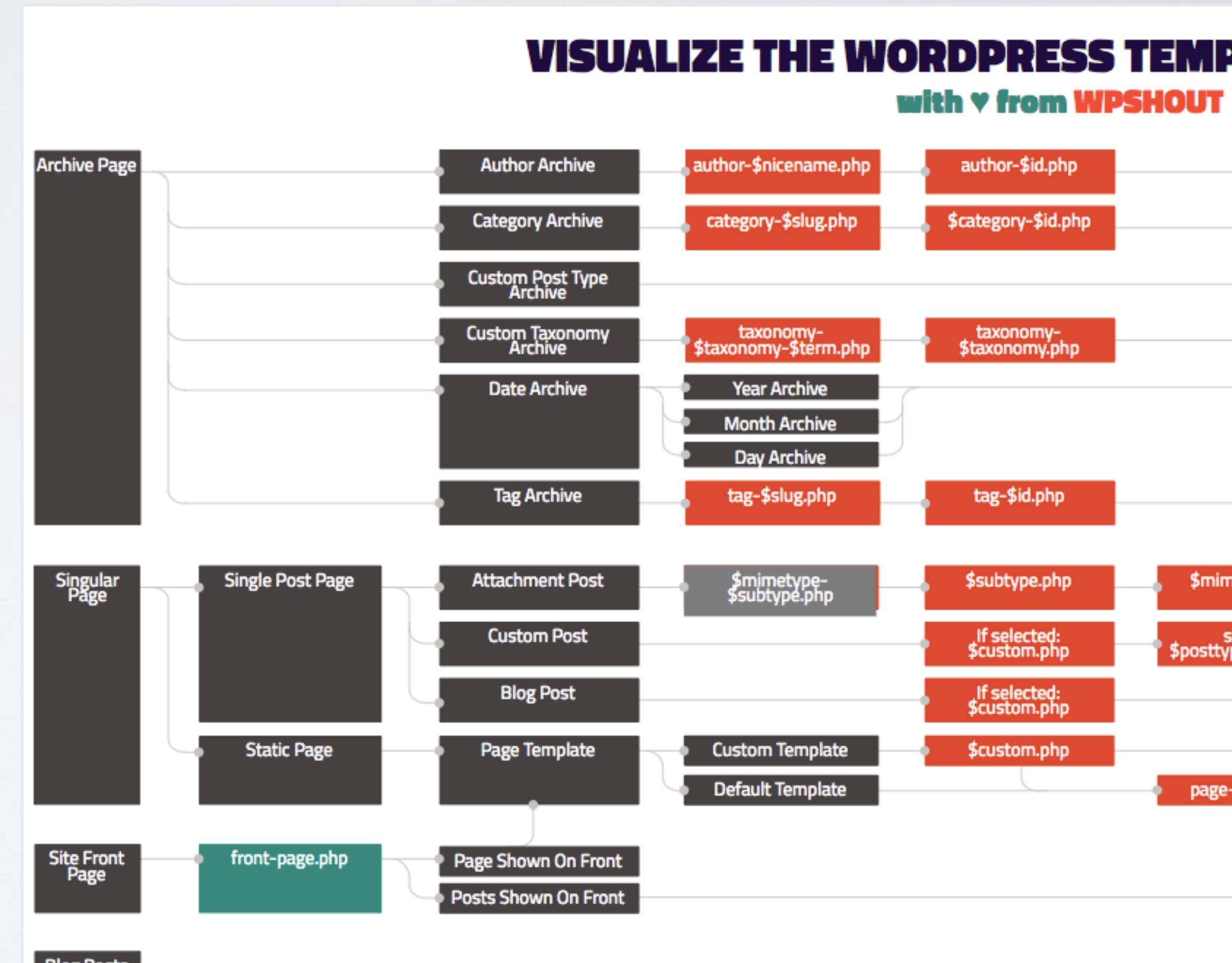
Si quieres conocer todas las condiciones que puedes usar, ingresa a:

<https://developer.wordpress.org/themes/basics/conditional-tags/>

# JERARQUÍA DE WORDPRESS

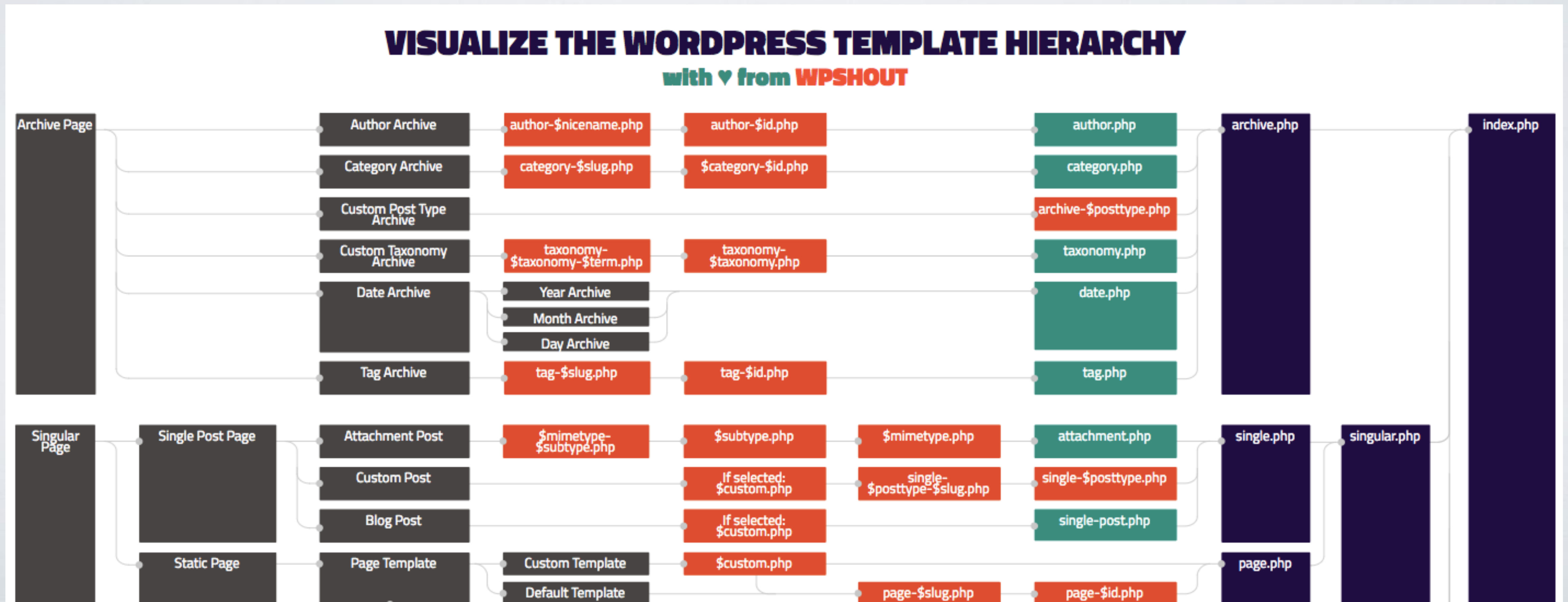
# ¿QUÉ ES LA JERARQUÍA DE WORDPRESS?

- La jerarquía hace referencia a las consultas que hace Wordpress para encontrar un archivo específico.
- Estas consultas se basan en una jerarquía de especificidad, la cual es usada en archivos usados dentro de la creación de temas en Wordpress.
- Existen diversos archivos basados en una **tabla**, con la cual podremos escoger para organizar de mejor manera nuestros temas.



Si quieres visualizar la jerarquía y conocer con detalle sus partes, ingresa al siguiente link:

<https://wphierarchy.com/>



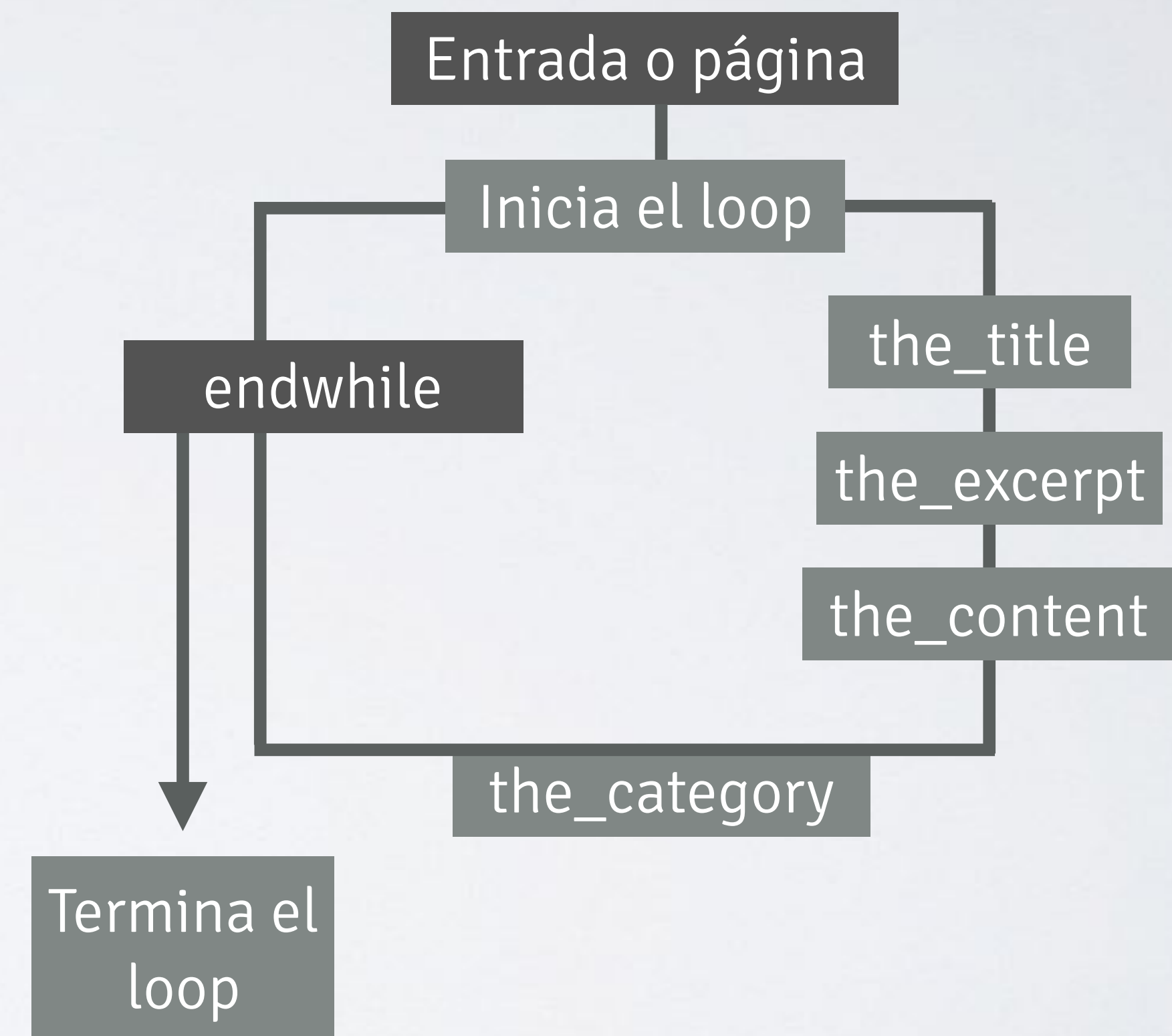


# LOOP DE WORDPRESS



# ¿QUÉ ES EL LOOP DE WORDPRESS?

- Mecanismo predeterminado que usa WordPress para generar publicaciones a través de los archivos de plantilla de un tema
- La cantidad de publicaciones que se recuperan está determinada por la cantidad de publicaciones definidas en la configuración de Lectura
- El loop extrae los datos de cada publicación de la base de datos de WordPress e inserta la información adecuada en lugar designado por el desarrollador



# LOOP BÁSICO DE WORDPRESS

- El loop evalúa si existen post. Mientras existan posts despliega el contenido de cada uno hasta que termine la iteración
- Podemos agregar etiquetas al loop para que se despliegue cierto contenido, como por ejemplo, cuántos post queremos ver, el título de una entrada, entre otras cosas

```
<?php if ( have_posts() ) : ?>
    <?php while ( have_posts() ) :
the_post(); ?>
        <!-- Muestra el contenido del post
-->
    <?php endwhile; ?>
<?php endif; ?>
```

# USANDO EL LOOP

El loop debe agregarse en **index.php** o en cualquier otra plantilla que se use para mostrar información acerca de una publicación.

```
<?php
    get_header();

    if ( have_posts() ) : while ( have_posts() ) : the_post();
        the_content();
    endwhile;
    else :
        _e( 'No hemos encontrado entradas', 'textdomain' );
    endif;

    get_footer();
?>
```

# QUÉ PODEMOS MOSTRAR EN EL LOOP

El loop de Wordpress puede mostrar diferentes elementos por cada entrada.  
Alguno de los elementos que puede mostrar son:

Nombre	Ejemplo
<code>the_author()</code>	Muestra el autor de la entrada o página
<code>the_content()</code>	Muestra el contenido principal de una entrada o página
<code>the_category()</code>	Muestra la categoría asociada a una entrada o página vista
<code>the_excerpt()</code>	Muestra un extracto del contenido de una entrada
<code>the_tags()</code>	Muestra una etiqueta asociada a una entrada
<code>the_title()</code>	Muestra el título de un post o página
<code>the_time()</code>	Muestra la fecha de publicación de una entrada o página