Tutorial para crear Temas de WP con Bootstrap

Edwin Salvador

5 de enero de 2017

Índice

1. Introducción

- Ya conocemos un poco sobre Wordpress.
- *"Wordpress es un software para la web que se puede usar para crear un hermoso sitio web o blog. Nos gusta decir que Wordpress es tanto gratuito como invaluable al mismo tiempo"*
- El libro "GuíaWPFácil (Manual de Wordpress)" puede servir como una guía introductoria a Wordpress. Pueden encontrarlo en el repositorio en la carpeta de libros/Wordpress.
- En este tutorial explicaré los detalles paso a paso de como entender y crear su propio tema para Wordpress.
- Esto les servirá para desarrollar el proyecto de final del semestre.

1.1. Temas Wordpress

- Miles de temas ya desarrollados de excelente calidad.
- Gratuitos o de pago.
- Ahorran tiempo.
- Algunas tienen demasiadas funcionalidades que no son útiles para nuestro sitio.
- Muchas lineas de código no necesarias.
- Puede ser que nuestro cliente no quiera un tema genérico si no uno completamente personalizado y desarrollado exclusivamente para el.

1.2. Bootstrap

- Utilizaremos Bootstrap para desarrollar nuestro tema de Wordpress.
- Podemos adaptar la plantilla que hemos venido desarrollando para utilizarla en Wordpress.

2. Pasos Previos

1. Instalar Wordpress
2. Revisar rápidamente el libro "GuíaWPFácil"
3. Importar el contenido de prueba "Test Unit" (`http://codex.wordpress.org/Theme_Unit_Test`) (opcional). Sirve para probar nuestros temas en varias situaciones y asegurarnos que se comporta correctamente.

3. Estructura de archivos

Para que un tema de WordPress pueda funcionar, en su nivel más básico, necesita sólo de 2 archivos:

1. **index.php**, será el encargado de llamar a las funciones de WordPress y proveer el maquetado HTML.
2. **style.css**, se encargará de asignar el diseño a nuestro tema.

Estos archivos los debemos guardar en una carpeta con el nombre de nuestro tema `"mi-tema"` y esta debemos colocarla dentro de `wp-content/themes`

Pero para crear un tema realmente completo necesitaremos algunos archivos extra. La carpeta de su tema debe verse de la siguiente forma:

- mi-tema
  - css
    - bootstrap.min.css
  - fonts
    - glyphicons-halflings-regular.eot
    - glyphicons-halflings-regular.svg
    - glyphicons-halflings-regular.ttf
    - glyphicons-halflings-regular.woff
  - footer.php
  - functions.php
  - header.php
  - img
    - favicon.ico
  - index.php
  - js
    - bootstrap.min.js
    - mi-tema.js
  - page.php
  - screenshot.png
  - sidebar.php
  - single.php
  - style.css

Tenemos los archivos de base de Bootstrap.

Dentro de img tenemos el favicon.ico (`http://iconogen.com/`). Además del favicon.ico podemos incluir los siguientes archivos que pueden ser generados en la misma URL. Estas imágenes se mostrarán cuando creemos un acceso directo de nuestra página en los dispositivos apple.

Listing 1: Apple icon touch

```
1 <link rel="apple-touch-icon" href="img/apple-touch-icon.png" />
2   <link rel="apple-touch-icon" sizes="57x57" href="img/
      apple-touch-icon-57x57.png" />
3   <link rel="apple-touch-icon" sizes="72x72" href="img/
      apple-touch-icon-72x72.png" />
4   <link rel="apple-touch-icon" sizes="114x114" href="img/
      apple-touch-icon-114x114.png" />
5   <link rel="apple-touch-icon" sizes="144x144" href="img/
      apple-touch-icon-144x144.png" />
6   <link rel="apple-touch-icon" sizes="57x57" href="img/
      apple-touch-icon-60x60.png" />
7   <link rel="apple-touch-icon" sizes="72x72" href="img/
      apple-touch-icon-120x120.png" />
8   <link rel="apple-touch-icon" sizes="114x114" href="img/
      apple-touch-icon-76x76.png" />
9   <link rel="apple-touch-icon" sizes="144x144" href="img/
      apple-touch-icon-152x152.png" />
```

4. style.css

- Este archivo a más de darle estilos a nuestro tema, brinda cierta información sobre nuestro tema al administrador de WP.

Listing 2: cabecera del style.css

```
/*
Theme Name: ANP Primer Tema
Theme URI: http://chalosalvador.me
Author: Chalo Salvador.
Author URI: http://chalosalvador.me
Description: Tema de WordPress creado con propositos educativos.
Version: 1.0
License: GNU General Public License v2 or later
License URI: http://www.gnu.org/licenses/gpl-2.0.html
Tags:
*/
```

4.1. Cabecera

- **Theme name:** El nombre del tema
- **Theme URI:** La dirección web del tema, si el tema fuera parte del directorio oficial, sería algo como http://wordpress.org/themes/mi-tema
- **Author:** Quién es el autor del tema
- **Author URI:** La dirección web del autor del tema
- **Description:** Una breve descripción del tema.
- **Version:** Qué versión del tema estamos trabajando
- **License:** Cuál es la licencia que aplica a este tema
- **License URI:** La url con los detalles de la licencia
- **Tags:** Algunas etiquetas para ser encontrado en el repositorio oficial de WordPress

4.2. screenshot.png

- Este archivo sirve como una preliminar de nuestro tema que se muestra en el menú `apariencia/temas` del wp-admin. Es una captura de pantalla de nuestro tema de 600x450px.

4.3. Importar Bootstrap

En nuestro archivo `style.css` importamos los estilos base de bootstrap.

Listing 3: importar Bootstrap en el style.css

```
@import url('css/bootstrap.min.css');
```

Ahora podemos seleccionar y activar nuestro tema desde el wp-admin. Comprobamos que el sitio presente nuestro tema (aparecerá en blanco ya que aún necesitamos escribir nuestro archivos `index.php`, `header.php` y `footer.php`).

5. header.php

Listing 4: header.php (HTML)

```
<!DOCTYPE html>
<html lang="es_ES">
<head>
  <title>Nombre del sitio | Descripcion del sitio</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
      maximum-scale=1.0, user-scalable=0">
  <!-- Cargando la Hoja de Estilos  -->
  <link rel="stylesheet" href="style.css" >
  <!-- Cargando los iconos  -->
  <link rel="shortcut icon" href="img/favicon.ico" type="image/x-icon" />
</head>
<body>
  <!-- #MAIN HEADER  -->
  <header id="main-header">
    <div class="container">
      <div class="row">
        <!-- Titulo del sitio  -->
        <div id="logo" class="col-md-6">
          <h1>Nombre del sitio<br>
          <small>Descripcion del sitio</small></h1>
        </div>
        <div class="col-md-6">
          <nav>
            <ul class="menu nav nav-pills">
              <li><a href="">Inicio</a></li>
              <li><a href="">Acerca de</a></li>
              <li><a href="">Portafolio</a></li>
              <li><a href="">Volver al tutorial</a></li>
            </ul>
          </nav>
        </div>
      </div>
      <hr />
    </div>
  </header>
  <!-- #MAIN HEADER  -->
```

Por el momento tenemos simple HTML, pero debemos insertar las funciones de WordPress para que el contenido sea dinámico.

Empezamos modificando lo que está dentro del `<head>`. Reemplazamos el lenguaje `"es_ES"` de `<html>` por el lenguaje que estemos usando en WordPress. Así el lenguaje es dinámico.

Listing 5: Lenguaje del sitio dinámico

```
<html <?php language_attributes(); ?>>
```

Ahora reemplazamos el título de la página. Utilizaremos la función `wp_title()`, para mostrar el título de la página actual, seguido de `bloginfo("name")`, que mostrará el nombre del sitio, resultando algo como: “Nombre de la página actual | Nombre del sitio”:

Listing 6: Título de la página dinámico

```
<title><?php wp_title('|', true, 'right');?> <?php bloginfo('name'); ?></
    title>
```

Igual que con el lenguaje lo que hacemos acá es cambiar el set de caracteres, por aquel que hayamos configurado en WordPress:

Listing 7: Charset dinámico

```
1 <meta charset="<?php bloginfo('charset');?>">
```

Con la siguiente línea actualizamos la llamada a nuestro style.css utilizando la función `bloginfo ('stylesheet_url')`. Nótese que es tercera vez que utilizamos esta función pero pasándole diferentes parámetros:

Listing 8: Importar el style.css

```
1 <link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>" >
```

La función bloginfo() imprime en el HTML cierta información de nuestro sitio, como el nombre, la descripción, etc. Para saber más se puede consultar en el codex (`http://codex.wordpress.org/Function_Reference/bloginfo`).

Ahora actualizamos la ruta para que WordPress encuentre las imágenes que estamos llamando. Usamos la función `get_stylesheet_directory_uri()` que obtiene la ruta a la carpeta en donde se encuentra nuestra hoja de estilos `style.css` y la imprimimos usando `echo`:

```
1  <link rel="shortcut icon" href="<?php echo get_stylesheet_directory_uri(); ?
       >/img/favicon.ico" type="image/x-icon" />
2  <link rel="apple-touch-icon" href="<?php echo get_stylesheet_directory_uri()
       ; ?>/img/apple-touch-icon.png" />
3  <link rel="apple-touch-icon" sizes="57x57" href="<?php echo
       get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-57x57.png" />
4  <link rel="apple-touch-icon" sizes="72x72" href="<?php echo
       get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-72x72.png" />
5  <link rel="apple-touch-icon" sizes="114x114" href="<?php echo
       get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-114x114.png" />
6  <link rel="apple-touch-icon" sizes="144x144" href="<?php echo
       get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-144x144.png" />
7  <link rel="apple-touch-icon" sizes="57x57" href="<?php echo
       get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-60x60.png" />
8  <link rel="apple-touch-icon" sizes="72x72" href="<?php echo
       get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-120x120.png" />
9  <link rel="apple-touch-icon" sizes="114x114" href="<?php echo
       get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-76x76.png" />
10 <link rel="apple-touch-icon" sizes="144x144" href="<?php echo
       get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-152x152.png" />
```

Las siguientes 3 líneas de código que son necesarias para el correcto funcionamiento de WordPress:

Listing 9: Líneas importantes para WP

```
1 <link rel="pingback" href="<?php bloginfo('pingback_url'); ?>" />
2 <?php if ( is_singular() ) wp_enqueue_script( "comment-reply" ); ?>
3 <?php wp_head(); ?>
```

La primera línea integra un link al archivo xml para que los pingbacks puedan funcionar correctamente, la segunda línea carga un script específico cada vez que se esté presentando una página singular (no un listado de artículos). Este script permitirá poder responder a los comentarios de una forma mucho más fluida. La tercera línea simplemente es un llamado a la función `wp_head();`. Es una llamada muy simple, pero fundamental para que nuestro tema funcione correctamente, en este lugar se cargarán los script y hojas de estilos que llamemos desde nuestro tema y desde los plugins que tengamos activos.

5.1. body

WordPress genera ciertas clases para el body de nuestro HTML. Aquí estamos cargándolas:

```
1 <body <?php body_class(); ?>>
```

Reemplazamos el título y subtítulo por el nombre y la descripción de nuestro sitio:

```
<div id="logo" class="col-md-6">
  <h1><?php bloginfo('name');?><br>
  <small><?php bloginfo('description'); ?></small></h1>
</div>
```

5.1.1. wp_nav_menu()

Listing 10: Incluir el menú en el body

```
<div class="col-md-6">
  <?php wp_nav_menu(array(
    'theme_location'  => 'menu-principal',
    'container'       => 'nav',
    'items_wrap'      => '<ul id="%1$s" class="%2$s nav nav-pills">%3$s</ul>
        '
  )); ?>
</div>
```

Le indica a a WordPress que en este lugar vamos a cargar un menú, y le damos algo más de información:

- Vamos a cargar el menú que hayamos asignado al área menu-principal
- El menú va a estar dentro de un elemento nav
- Los ítems estarán dentro de un elemento ul con la clase nav nav-pills

5.1.2. functions.php (register_nav_menus())

Para que esta llamada al menú funcione primero deberemos registrar el área de menú `'menu-principal'`. Para ello vamos a abrir `functions.php` y vamos a escribir lo siguiente:

Listing 11: Registrar el menú

```
//-----------------------------------------------------------------------
// REGISTRAMOS EL MENU
//-----------------------------------------------------------------------

register_nav_menus( array(
  'menu-principal' => __('Area principal de navegacion', 'anp')
) );
```

Utilizamos la función `register_nav_menus()` para registrar el área de menú `'menu-principal'` cuya descripción es 'Área principal de navegación' (Figura 1).

Ahora podemos crear un menú y asignarlo al Área principal de navegación.

El archivo `header.php` debe quedar de la siguiente manera:

Listing 12: header.php

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>

  <title><?php wp_title('|', true, 'right');?> <?php bloginfo('name'); ?></
    title>
```

```
6
7     <meta charset="<?php bloginfo('charset');?>">
8     <meta name="viewport" content="width=device-width, initial-scale=1.0,
          maximum-scale=1.0, user-scalable=0">
9
10    <!-- Cargando la Hoja de Estilos  -->
11    <link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>" >
12
13    <!-- Cargando los iconos  -->
14    <link rel="shortcut icon" href="<?php echo get_stylesheet_directory_uri();
          ?>/img/favicon.ico" type="image/x-icon" />
15    <link rel="apple-touch-icon" href="<?php echo get_stylesheet_directory_uri
          (); ?>/img/apple-touch-icon.png" />
16    <link rel="apple-touch-icon" sizes="57x57" href="<?php echo
          get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-57x57.png" />
17    <link rel="apple-touch-icon" sizes="72x72" href="<?php echo
          get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-72x72.png" />
18    <link rel="apple-touch-icon" sizes="114x114" href="<?php echo
          get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-114x114.png" /
          >
19    <link rel="apple-touch-icon" sizes="144x144" href="<?php echo
          get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-144x144.png" /
          >
20    <link rel="apple-touch-icon" sizes="57x57" href="<?php echo
          get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-60x60.png" />
21    <link rel="apple-touch-icon" sizes="72x72" href="<?php echo
          get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-120x120.png" /
          >
22    <link rel="apple-touch-icon" sizes="114x114" href="<?php echo
          get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-76x76.png" />
23    <link rel="apple-touch-icon" sizes="144x144" href="<?php echo
          get_stylesheet_directory_uri(); ?>/img/apple-touch-icon-152x152.png" /
          >
24
25    <!-- Pingbacks -->
26    <link rel="pingback" href="<?php bloginfo('pingback_url'); ?>" />
27
28    <?php if ( is_singular() ) wp_enqueue_script( "comment-reply" ); ?>
29
30    <?php wp_head(); ?>
31
32  </head>
33
34  <body <?php body_class(); ?>>
35
36    <!-- #MAIN HEADER  -->
37    <header id="main-header">
38
39      <div class="container">
40        <div class="row">
41
```

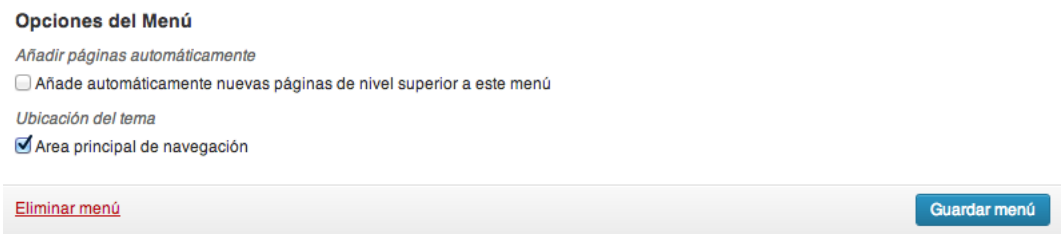

Figura 1: Localización del menú

```
42          <!-- Titulo del sitio  -->
43          <div id="logo" class="col-md-6">
44            <h1><?php bloginfo('name');?><br>
45            <small><?php bloginfo('description'); ?></small></h1>
46          </div>
47
48          <div class="col-md-6">
49            <?php wp_nav_menu(array(
50              'theme_location'  => 'menu-principal',
51              'container'       => 'nav',
52              'items_wrap'      => '<ul id="%1$s" class="%2$s nav nav-pills">
                    %3$s</ul>'
53            )); ?>
54          </div>
55
56        </div>
57
58        <hr />
59      </div>
60
61    </header>
62    <!-- #MAIN HEADER  -->
```

6. footer.php

Abrimos el archivo `footer.php` y escribimos lo siguiente:

Listing 13: footer.php (HTML)

```
1   <div class="container">
2
3     <hr />
4
5     <p>La descripcion de mi sitio</p>
6
7   </div>
8 </body>
9 </html>
```

Al igual que lo hicimos con `header.php`, acabamos de ingresar simple HTML, ahora vamos a hacerlo dinámico:

Listing 14: footer.php

```
1   <div class="container">
2
3     <hr />
4
5     <p><?php bloginfo('description'); ?></p>
6
7   </div>
8
9   <?php wp_footer(); ?>
10 </body>
11 </html>
```

Como podemos ver estamos imprimiendo la descripción del sitio y estamos usando la función `wp_footer()`, que es un símil de `wp_head()`. En este lugar se cargarán todos los scripts que el tema y los plugins necesiten poner en el pie de página.

7. index.php

Dependiendo de la página del sitio que estemos viendo WordPress cargará uno u otro de los archivos base de nuestro tema, esto se llama la jerarquia de plantilla (`https://developer.wordpress.org/themes/basics/template-hierarchy/`), es por ello que hemos separado nuestra cabecera y pie de página en 2 archivos separados, de esta manera desde cualquiera de los otros solo debemos insertar una llamada al `header.php` y al `footer.php` y eso asegurará que cualquier cambio que hagamos en uno de esos 2 archivos se verá reflejado en cualquiera de las páginas del sitio.

Así que lo primero que haremos en nuestro `index.php` va a ser llamar a nuestra cabecera y pie de página:

Listing 15: index.php base

```
1  <?php get_header(); ?>
2
3 <?php get_footer(); ?>
```

Luego, entre estas 2 llamadas insertaremos algo de HTML:

Listing 16: index.php HTML

```
1  <?php get_header(); ?>
2
3  <section class="container">
4
5    <div class="row">
6
7      <div class="col-md-8">
8
9        <article class="clearfix">
10
11         <header>
12           <h2>Titulo del articulo</h2>
13           <div class="meta">15 marzo, 2012 &bull; <a href="">Categoria</a></
                 div>
14         </header>
15
16         <img src="http://lorempixel.com/150/150" alt="" class="" />
17
18         <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
               eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
               enim ad minim veniam, quis nostrud exercitation ullamco laboris
               nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
               in reprehenderit in voluptate velit esse cillum dolore eu fugiat
                nulla pariatur. Excepteur sint occaecat cupidatat non proident,
                sunt in culpa qui officia deserunt mollit anim id est laborum.<
               /p>
19
20         <p><a href="" title="" class="btn btn-default">Ver mas</a></p>
21
22       </article>
23
24       <hr />
25
26       <ul class="pager">
27         <li class="previous">
28           <a href="">&larr; Anteriores</a>
29         </li>
30         <li class="next">
31           <a href="">Recientes &rarr;</a>
32         </li>
```

```
33        </ul>
34
35      </div>
36      <div class="col-md-4">
37        Este es el sidebar
38      </div>
39
40    </div>
41
42  </section>
43
44  <?php get_footer(); ?>
```

Si observamos el HTML, hemos creado un layout de 2 columnas, donde en la columna izquierda se cargará el listado de artículos y en la columna derecha se cargará el sidebar. Hagamos que los contenidos se carguen de verdad, para ello usaremos el loop de WordPress (`http://codex.wordpress.org/The_Loop`).

Listing 17: El loop

```
1  <?php //El loop básicamente comprueba si hay posts para mostrar, luego
       mientras haya posts cargará cada uno de ellos usando el esquema que se
       ve a continuación
2  if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
3
4    <article <?php post_class('clearfix'); //Carga las clases específicas del
         post y agrega la clase clearfix ?>>
5
6      <header>
7        <h2><?php the_title(); //Muestra el título del post ?></h2>
8        <div class="meta">
9          <?php the_time(get_option('date_format')); //Muestra la fecha de
             publicación del artículo ?>
10         &bull;
11         <?php the_category(', '); //Muestra enlaces a las categorias
             separados por coma ?></div>
12     </header>
13
14     <?php //Verifica si el post tiene una imagen destacada
15     if ( has_post_thumbnail() ) {
16
17       //Si tiene imagen destacada entonces carga la imagen en tamaño
           thumbnail (miniatura), y le añade la clase alignleft
18       the_post_thumbnail('thumbnail', array(
19         'class' => 'alignleft'
20       ));
21     }?>
22
23     <?php //En esta parte muestra el extracto del post
24     the_excerpt(); ?>
25
26     <p>
27       <a href="<?php the_permalink(); //Imprime un link al detalle del post
           ?>" title="<?php the_title_attribute(); //Añade un título al
           enlace ?>" class="btn btn-default"><?php _e('Ver más', 'anp'); //
           El botón dirá Ver más. ?></a></p>
28
29   </article>
30
31   <hr />
32
33 <?php endwhile; //Acá termina la estructura con la que se presentará cada
     post
34 else: //Ahora bien, si no hay artículos para mostrar entonces cargará lo
     siguiente  ?>
```

```
35
36    <article>
37
38      <header>
39        <h2><?php _e('Este contenido no está disponible', 'anp'); //Un simple
              título ?></h2>
40      </header>
41
42      <?php get_search_form(); //Llama un formulario de búsqueda  ?>
43
44    </article>
45
46  <?php endif; // Aquí termina el loop?>
```

El código anterior tiene varias líneas con la explicación de lo que hace cada una de las funciones, pero lo que hace este código básicamente es buscar si hay posts para mostrar y mientras los haya, mostrará cada uno dentro del elemento `article`.

Ahora a continuación del loop, vamos a cargar enlaces a los posts más recientes y a los más antigüos:

Listing 18: Paginación

```
1  <ul class="pager">
2    <?php if( get_next_posts_link()) { //Si es que hay más posts anteriores ?>
3      <li class="previous">
4        <?php next_posts_link(__('&larr; Anteriores', 'anp')); //Muestra un
             enlace a los posts anteriores ?>
5      </li>
6    <?php } ?>
7    <?php if( get_previous_posts_link()) { //Si es que hay más posts más
       recientes ?>
8      <li class="next">
9        <?php previous_posts_link(__('Recientes &rarr;', 'anp'));  //Muestra
             un enlace a los posts más recientes?>
10     </li>
11   <?php } ?>
12 </ul>
```

Según lo que podemos ver en la jerarquía de plantilla `https://developer.wordpress.org/themes/basics/template-hierarchy/`dependiendo de la página que estemos viendo se cargará una u otro archivo desde nuestro tema, por ejemplo, si estamos viendo una página de categoría, WordPress buscará dentro de nuestro tema al archivo `category.php`, si no lo encuentra cargará `index.php`. Figura 2

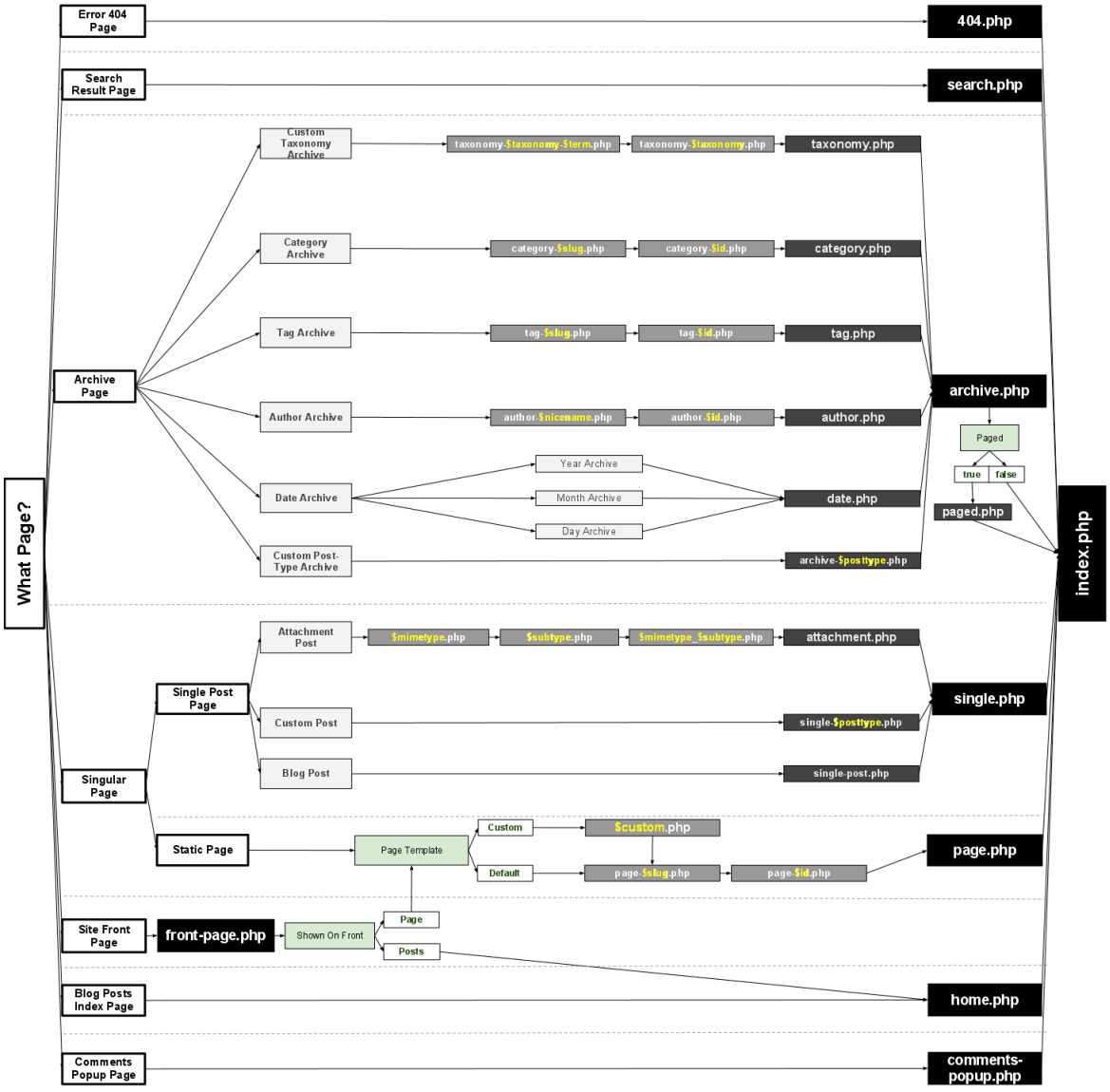

Figura 2: Jerarquía de Plantilla

Siguiendo esta lógica deberíamos crear una gran cantidad de archivos para presentar en diferentes circunstancias, sin embargo, si el único cambio que tendrán estos archivos extra es mostrar un mensaje para indicar que estamos viendo una página de categorías o de archivo diario (por ejemplo), es mucho más fácil y elegante usar algunas funciones condicionales e insertar esos mensajes directamente en nuestro archivo index.php de la siguiente forma:

Listing 19: Verificación de página

```
<?php if ( is_category() ) : // Si estamos en una página de categoría?>
  <div class="alert alert-info">
    <p><?php _e('Artículos en la categoría', 'anp');?>: <strong><?php
      single_cat_title(); //Muestra el nombre de la categoría ?></strong></p
      >
  </div>
<?php elseif ( is_tag() ) : // Si estamos en una página de etiqueta?>
  <div class="alert alert-info">
    <p><?php _e('Artículos etiquetados como', 'anp');?>: <strong><?php
      single_tag_title(); //Muestra el nombre de la etiqueta ?></strong></p>
  </div>
<?php elseif ( is_day() ) : // Si estamos en una página de archivo diario?>
  <div class="alert alert-info">
    <p><?php _e('Archivo', 'anp');?>: <strong><?php the_date(); //Muestra la
      fecha del archivo ?></strong></p>
  </div>
<?php elseif ( is_month() ) : // Si estamos en una página de archivo mensual
    ?>
  <div class="alert alert-info">
    <p><?php _e('Archivo', 'anp');?>: <strong><?php the_date('F Y'); //
      Muestra la fecha del archivo (mes y año) ?></strong></p>
  </div>
<?php elseif ( is_year() ) : // Si estamos en una página de archivo anual?>
  <div class="alert alert-info">
    <p><?php _e('Archivo', 'anp');?>: <strong><?php the_date('Y'); //Muestra
      la fecha del archivo (año) ?></strong></p>
  </div>
<?php elseif ( is_search() ) : // Si estamos en una página de resultados de
    búsqueda?>
  <div class="alert alert-info">
    <p><?php _e('Resultados para', 'anp');?>: <strong><?php echo
      the_search_query(); ////Muestra el término por el que se ha realizado
      la búsqueda ?></strong></p>
  </div>
<?php elseif ( is_author() ) : // Si estamos en una página de artículos
    escritos pur un autor específico?>
  <div class="alert alert-info">
    <p><?php _e('Artículos por', 'anp');?>: <strong><?php $curauth = (isset(
      $_GET['author_name'])) ? get_user_by('slug', $author_name) :
      get_userdata(intval($author)); echo $curauth->display_name; //Muestra
      el nombre del autor ?></strong></p>
  </div>
<?php elseif ( is_404() ) : // Si estamos en una página de error 404?>
  <div class="alert alert-danger">
    <p><strong><?php _e('Error 404', 'anp');?>:</strong> <?php _e('Página no
      encontrada', 'anp');?></p>
  </div>
<?php endif; // Fin de las comprobaciones condicionales ?>
```

El resultado será que WordPress verificará en qué página estamos actualmente y dependiendo de cuál sea el resultado de esa verificación se presentará un mensaje específico.

Al final nuestro archivo `index.php` quedará así:

Listing 20: index.php

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

<section class="container">

  <div class="row">

    <div class="col-md-8">

      <?php if ( is_category() ) : // Si estamos en una página de categoría?
          >
        <div class="alert alert-info">
          <p><?php _e('Artículos en la categoría', 'anp');?>: <strong><?php
            single_cat_title(); //Muestra el nombre de la categoría ?></s
            trong></p>
        </div>
      <?php elseif ( is_tag() ) : // Si estamos en una página de etiqueta?>
        <div class="alert alert-info">
          <p><?php _e('Artículos etiquetados como', 'anp');?>: <strong><?php
             single_tag_title(); //Muestra el nombre de la etiqueta ?></s
            trong></p>
        </div>
      <?php elseif ( is_day() ) : // Si estamos en una página de archivo
          diario?>
        <div class="alert alert-info">
          <p><?php _e('Archivo', 'anp');?>: <strong><?php the_date(); //
            Muestra la fecha del archivo ?></strong></p>
        </div>
      <?php elseif ( is_month() ) : // Si estamos en una página de archivo
          mensual?>
        <div class="alert alert-info">
          <p><?php _e('Archivo', 'anp');?>: <strong><?php the_date('F Y');
            //Muestra la fecha del archivo (mes y año) ?></strong></p>
        </div>
      <?php elseif ( is_year() ) : // Si estamos en una página de archivo
          anual?>
        <div class="alert alert-info">
          <p><?php _e('Archivo', 'anp');?>: <strong><?php the_date('Y'); //
            Muestra la fecha del archivo (año) ?></strong></p>
        </div>
      <?php elseif ( is_search() ) : // Si estamos en una página de
          resultados de búsqueda?>
        <div class="alert alert-info">
          <p><?php _e('Resultados para', 'anp');?>: <strong><?php echo
            the_search_query(); ////Muestra el término por el que se ha
            realizado la búsqueda ?></strong></p>
        </div>
      <?php elseif ( is_author() ) : // Si estamos en una página de artí
          culos escritos pur un autor específico?>
        <div class="alert alert-info">
          <p><?php _e('Artículos por', 'anp');?>: <strong><?php $curauth = (
            isset($_GET['author_name'])) ? get_user_by('slug', $author_name)
             : get_userdata(intval($author)); echo $curauth->display_name;
            //Muestra el nombre del autor ?></strong></p>
        </div>
      <?php elseif ( is_404() ) : // Si estamos en una página de error 404?>
        <div class="alert alert-danger">
          <p><strong><?php _e('Error 404', 'anp');?>:</strong> <?php _e('Pá
            gina no encontrada', 'anp');?></p>
        </div>
      <?php endif; // Fin de las comprobaciones condicionales ?>

      <?php //El loop básicamente comprueba si hay posts para mostrar, luego
```

```
          mientras haya posts cargará cada uno de ellos usando el esquema
          que se ve a continuación
44      if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
45
46        <article <?php post_class('clearfix'); //Carga las clases especí
          ficas del post y agrega la clase clearfix ?>>
47
48          <header>
49            <h2><?php the_title(); //Muestra el título del post ?></h2>
50            <div class="meta">
51              <?php the_time(get_option('date_format')); //Muestra la fecha
                    de publicación del artículo ?>
52              &bull;
53              <?php the_category(', '); //Muestra enlaces a las categorías
                    separados por coma ?></div>
54          </header>
55
56          <?php //Verifica si el post tiene una imágen destacada
57          if ( has_post_thumbnail() ) {
58
59            //Si tiene imágen destacada entonces carga la imagen en tamaño
                  thumbnail (miniatura), y le añade la clase alignleft
60            the_post_thumbnail('thumbnail', array(
61              'class' => 'alignleft'
62            ));
63          }?>
64
65          <?php //En esta parte muestra el extracto del post
66          the_excerpt(); ?>
67
68          <p>
69            <a href="<?php the_permalink(); //Imprime un link al detalle del
                   post ?>" title="<?php the_title_attribute(); //Añade ul tí
                  tulo al enlace ?>" class="btn btn-default"><?php _e('Ver más
                  ', 'anp'); //El botón dirá Ver más. ?></a></p>
70
71        </article>
72
73        <hr />
74
75      <?php endwhile; //Acá termina la estructura con la que se presentará
          cada post
76      else: //Ahora bien, si no hay artículos para mostrar entonces cargará
          lo siguiente  ?>
77
78        <article>
79
80          <header>
81            <h2><?php _e('Este contenido no está disponible', 'anp'); //Un
          simple título ?></h2>
82          </header>
83
84          <?php get_search_form(); //Llama un formulario de búsqueda  ?>
85
86        </article>
87
88      <?php endif; // Aquí termina el loop?>
89
90      <ul class="pager">
91        <?php if( get_next_posts_link()) { //Si es que hay más posts
              anteriores ?>
92          <li class="previous">
93            <?php next_posts_link(__('&larr; Anteriores', 'anp')); //Muestra
```

```
                    un enlace a los posts anteriores ?>
94            </li>
95          <?php } ?>
96          <?php if( get_previous_posts_link()) { //Si es que hay más posts más
                recientes ?>
97            <li class="next">
98              <?php previous_posts_link(__('Recientes &rarr;', 'anp'));  //
                    Muestra un enlace a los posts más recientes?>
99            </li>
100         <?php } ?>
101       </ul>
102
103     </div>
104     <div class="col-md-4">
105       Este es el sidebar
106     </div>
107
108   </div>
109
110 </section>
111
112 <?php get_footer(); ?>
```

Desde el loop estamos llamando a la imagen destacada de los posts (si es que la tienen), sin embargo para habilitar la posibilidad de que al escribir un post se le asigne una imagen destacada, necesitaremos activar esta característica mediante la función `add_theme_support()` desde el archivo `functions.php`. También vamos a aprovechar para habilitar otras características importantes. **Leer los comentarios en el código para entender bien de qué se trata:**

Listing 21: Características del tema

```
1  //------------------------------------------------------------------------
2  // CARACTERÍSTICAS DEL TEMA
3  //------------------------------------------------------------------------
4
5  // Ajustar el máximo ancho de las imagenes de acuerdo al diseño de este modo
       cualquier imagen que insertemos en el contenido de un artículo va a
      tener como máximo este ancho
6  if ( ! isset( $content_width ) )
7    $content_width = 750; //El ancho máximo será de 750 pixeles
8
9  // Creamos una función para registrar algunas características del tema
10 function anp_theme_features()  {
11
12   // Permitimos que el sitio soporte RSS Automáticos
13   add_theme_support( 'automatic-feed-links' );
14
15   // Permitimos qe el tema soporte imagenes destacadas
16   add_theme_support( 'post-thumbnails');
17 }
18
19 // Ejecutamos la función y registra las características
20 add_action( 'after_setup_theme', 'anp_theme_features' );
```

Ahora podemos ver que el tema ya va tomando forma y se muestra así: Figura 3

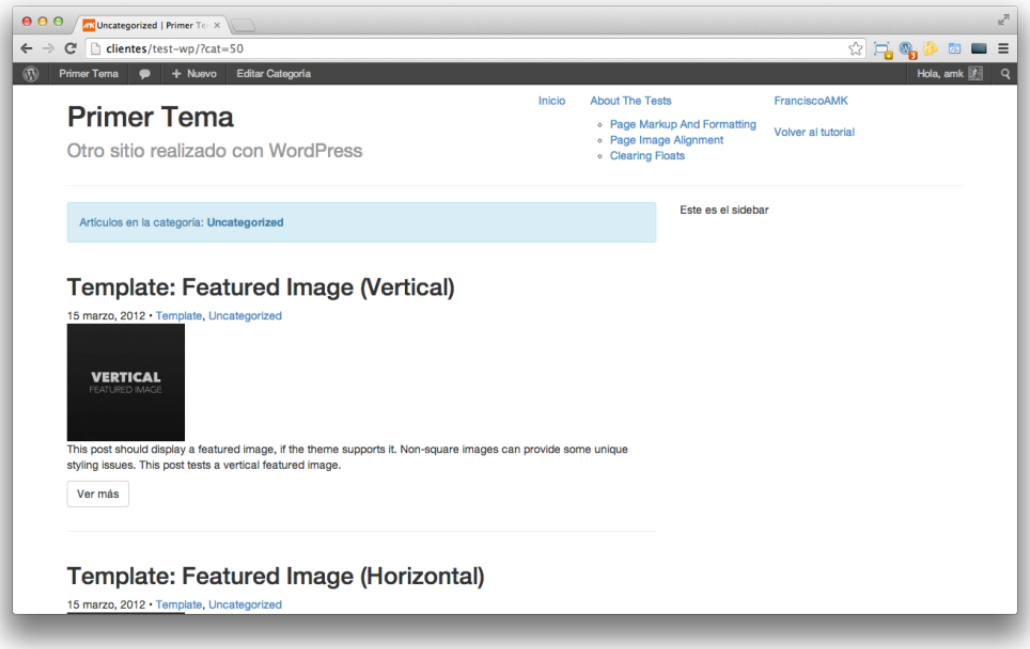

Figura 3: Sitio preliminar

8. Sidebar

El archivo sidebar.php es (generalmente) el que contiene la zona dinámica principal en la que podemos insertar widgets.

Actualmente la zona des sidebar dentro del archivo index.php está de la siguiente forma:

Listing 22: Sección del sidebar en index.php

```
<div class="col-md-4">
  Este es el sidebar
</div>
```

y vamos a editarla para que quede así:

Listing 23: Función WP para el sidebar

```
<div class="col-md-4">
  <?php get_sidebar(); ?>
</div>
```

La función `get_sidebar`, al igual que `get_header` y `get_footer`, importa un archivo específico desde nuestro tema. En este caso, el archivo sidebar.`php`.

Antes de modificar el archivo `sidebar.php`, debemos habilitar una zona dinámica de widgets desde nuestro `functions.php`, por lo que vamos a abrir este archivo y escribir lo siguiente a continuación de lo que ya tenemos:

Listing 24: functions.php (registro del sidebar)

```
//-----------------------------------------------------------------------
// REGISTRAMOS EL SIDEBAR
//-----------------------------------------------------------------------

```

```php
//Con la función register_sidebar, registramos una zona dinámica para nuestro tema y le pasamos algunos parámetros
register_sidebar(array(
  'name' => __('Bara lateral', 'anp'), //El nombre del área dinámica
  'id' => 'barra-lateral', //Un identificador único para la zona
  'description' => __( 'Este es el área de widgets del sitio.', 'anp'), // Una breve descripción
  'before_widget' => '<div id="%1$s" class="widget %2$s">', //Algo de HTML que irá antes de cada widget
  'after_widget'  => '</div>', //Algo de HTML que irá después de cada widget
  'before_title' => '<h3>', //La etiqueta que irá antes del título de cada widget
  'after_title' => '</h3>' //La etiqueta que irá después del título de cada widget
));
```

Lo que acabamos de hacer es registrar una nueva área dinámica mediante la función `register_sidebar` . A esta función le hemos dado algunos parámetros, como el nombre del área y un nombre identificador entre otras cosas.

Al actualizar nuestro archivo de funciones con este nuevo contenido veremos que en nuestro administrador (wp-admin) se habilita la opción de utilizar widgets y nos muestra el área que acabamos de crear para insertar los widgets que nosotros queramos Figura 4.

Figura 4: Sidebar

Ahora abrimos el archivo `sidebar.php` y escribimos lo siguiente:

Listing 25: Código para el sidebar.php

```php
<?php if( is_active_sidebar('barra-lateral') ) { // Verificamos si el área "barra-lateral" está activa
  dynamic_sidebar( 'barra-lateral' ); // Y si está activa, la cargamos acá
} //Aquí termina la verificación ?>
```

Con esto lo que hacemos utilizar el identificador único del área de widgets, para verificar si ésta se

encuentra activa, y si la respuesta es sí, entonces cargará en esta zona el área de widgets `barra-lateral`. Así, en la parte frontal del sitio ya vemos que los widgets ya se están mostrando Figura 5:

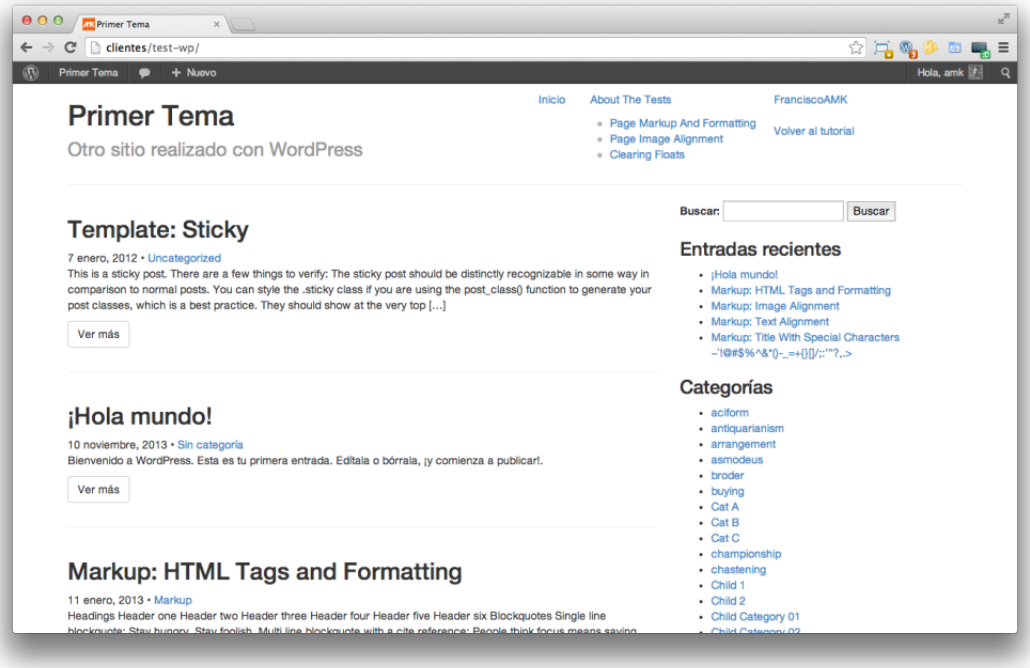

Figura 5: Zona de Widgets

9. Estilos CSS

En nuestro style.css hasta el momento sólo hemos declarado la información básica del tema y hemos cargado bootstrap, ahora vamos a agregar algo más de estilo para mejorar la apariencia de nuestro sitio, entonces abrimos `style.css` y añadimos lo siguiente:

Listing 26: style.css

```
body {
  font-family: 'Open Sans', Helvetica, Arial, sans-serif;
}

h1,
h2,
h3,
h4,
h5,
h6 {
  font-family: 'Open Sans', Helvetica, Arial, sans-serif;
  font-weight: 700;
}

h1 {
  font-size: 35px;
}

h2 {
  font-size: 25px;
}

h3 {
  font-size: 20px;
}

a {
  color: #80ca26;
}

img {
  max-width: 100%;
  height: auto;
}

code {
  white-space: normal;
}

/************ HEADER ************/
#main-header {
  padding-top: 40px;
  padding-bottom: 20px;
}

  #main-header h1 {
    margin: 0;
    font-size: 20px;
  }

    #main-header h1 small {
      margin: 0;
      font-size: 15px;
      color: #999;
    }
```

```
56
57    #main-header hr {
58      margin-top: 40px;
59    }
60
61    #main-header nav {
62      text-align: right;
63    }
64      #main-header nav ul li {
65
66      }
67        #main-header nav ul li ul {
68          -ms-filter:"progid:DXImageTransform.Microsoft.Alpha(Opacity=0)";
69          filter: alpha(opacity=0);
70          opacity: 0;
71
72          visibility: hidden;
73
74          -webkit-transition: all 0.2s linear;
75          -moz-transition: all 0.2s linear;
76          -o-transition: all 0.2s linear;
77          -ms-transition: all 0.2s linear;
78          transition: all 0.2s linear;
79
80          position: absolute;
81          z-index: 1;
82          left: 0;
83          top: 40px;
84          width: 200px;
85          background: #f4f4f4;
86          -webkit-border-radius: 5px;
87          -moz-border-radius: 5px;
88          border-radius: 5px;
89          margin: 0;
90          padding: 0;
91        }
92        #main-header nav ul li:hover ul {
93          -ms-filter:"progid:DXImageTransform.Microsoft.Alpha(Opacity=100)";
94          filter: alpha(opacity=100);
95          opacity: 1;
96
97          visibility: visible;
98        }
99          #main-header nav ul ul li {
100           margin: 0;
101           padding: 0;
102           list-style: none;
103           text-align: left;
104         }
105           #main-header nav ul ul li a {
106             padding: 10px 15px;
107             display: block;
108           }
109
110 /************ ALGUNOS ESTILOS EXTRA ************/
111 .sticky {
112   background: #fffde8;
113   padding: 15px;
114 }
115
116 .meta {
117   font-size: 0.8em;
118   margin-bottom: 15px;
```

```
119 }
120
121 .paginacion {
122   margin-top: 40px;
123   border-top: solid 1px #eee;
124   padding-top: 10px;
125 }
126
127 .widget {
128   margin-bottom: 40px;
129 }
```

Con estas líneas de CSS nuestro sitio luce considerablemente más ordenado que hasta ahora, sin embargo, podemos apreciar que estamos utilizando como tipografía base la fuente Open Sans `http://www.google.com/fonts/specimen/Open+Sans` y para poder utilizarla tranquilamente necesitaremos cargar la hoja de estilos desde Google Web Fonts `http://www.google.com/fonts`.

9.1. Cargando Hojas de estilos

Para cargar hojas de estilos tenemos 3 formas de hacerlo:

1. Insertarlo en el head de `header.php`
2. Importarlo desde `style.css`
3. Cargarlo desde `functions.php`

En la primera parte de este tutorial insertamos `style.css` y `bootstrap.min.css` de la primera y segunda forma respectivamente. Ese es un método que nos puede servir si estamos trabajando sólo con una o dos hojas de estilo, sin embargo cuando necesitamos agregar más, generar dependencia entre ellas y conocer la versión de cada una, debemos hacerlo desde `functions.php`. Es mucho más limpio y nos permite mantener nuestros archivos más simples.

9.1.1. wp_register_style

Para cargar archivos CSS desde `functions.php` usaremos las funciones `wp_register_style`, que sirve para registrar una nueva hoja de estilos y `wp_enqueue_style` que incluirá en el head de nuestro tema (justo en el lugar en donde pusimos `wp_head`) alguno de los archivos CSS registrados.

A la función `wp_register_style` le vamos a asignar ciertos parámetros, que incluiremos en el siguiente orden:

1. **Nombre**: Un nombre para la hoja de estilos
2. **URL**: la dirección donde se encuentra la hoja de estilos
3. **Dependencia**: Si este CSS depende de otras hojas de estilo para funcionar aquí añadimos su nombre
4. **Versión**: La versión de esta hoja de estilos
5. **Media**: A qué medios se aplicará este CSS (all, print, screen)

Ahora debemos modificar nuestro `functions.php` añadiendo estas líneas:

Listing 27: cargando los estilos en functions.php

```
//---------------------------------------------------------------------
// CARGANDO ESTILOS DEL TEMA
//---------------------------------------------------------------------
//Creamos una función para cargar los estilos
function anp_theme_styles() {

  //Registramos la fuente Open Sans
  wp_register_style( 'font-sans', 'http://fonts.googleapis.com/css?family=
      Open+Sans:400italic,700italic,400,700', '', '', 'all' );

  //Registramos Bootstrap
  wp_register_style( 'bootstrap', get_stylesheet_directory_uri().'/css/
      bootstrap.min.css', '', '3.0.0', 'all' );

  //registramos la hoja de estilos del tema
  wp_register_style( 'anp-style', get_stylesheet_uri(), array('font-sans', '
      bootstrap'), '1.0.0', 'all' );

  //Ahora cargamos los estilos. Nota que sólo cargamos 'anp-style' ya que en
       esta hoja de estilos declaramos dependencia de 'font-sans' y '
      bootstrap', éstas cargaran de manera automática
  wp_enqueue_style( 'anp-style' );
}
add_action('wp_enqueue_scripts', 'anp_theme_styles'); //Ejecutamos la funció
    n
```

Registramos 3 estilos pero sólo cargamos '`anp-style`' ya que como en este CSS declaramos dependencia de '`font-sans`' y '`bootstrap`', estos últimos estarán obligados a cargar antes que `style.css`.

Ahora en `header.php` eliminamos esta línea, ya que estamos cargando los estilos desde el archivo `functions.php`:

Listing 28: Eliminar linea de header.php

```
<link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>"
```

Y en `style.css` eliminamos esta línea:

Listing 29: Eliminar línea de style.css

```
@import url('css/bootstrap.min.css');
```

Refrescamos nuestro sitio en el navegador y verificamos que todo carga correctamente.

Por último en nuestro `style.css` agregamos los estilos para las clases que son asignadas por WordPress a algunos elementos en determinadas circunstancias, como por ejemplo, cuando decidimos alinear una imagen a la izquierda o a la derecha:

Listing 30: Estilos del core de WP

```
/************ WORDPRESS CORE ************/
.alignnone {
    margin: 5px 20px 20px 0;
}

.aligncenter,
div.aligncenter {
    display: block;
    margin: 5px auto 5px auto;
```

```
}

.alignright {
    float:right;
    margin: 5px 0 20px 20px;
}

.alignleft {
    float: left;
    margin: 5px 20px 20px 0;
}

.aligncenter {
    display: block;
    margin: 5px auto 5px auto;
}

a img.alignright {
    float: right;
    margin: 5px 0 20px 20px;
}

a img.alignnone {
    margin: 5px 20px 20px 0;
}

a img.alignleft {
    float: left;
    margin: 5px 20px 20px 0;
}

a img.aligncenter {
    display: block;
    margin-left: auto;
    margin-right: auto
}

.wp-caption {
    background: #fff;
    border: 1px solid #f0f0f0;
    max-width: 96%;
    padding: 5px 3px 10px;
    text-align: center;
}

.wp-caption.alignnone {
    margin: 5px 20px 20px 0;
}

.wp-caption.alignleft {
    margin: 5px 20px 20px 0;
}

.wp-caption.alignright {
    margin: 5px 0 20px 20px;
}

.wp-caption img {
    border: 0 none;
    height: auto;
    margin: 0;
    max-width: 98.5%;
    padding: 0;
```

```
73      width: auto;
74  }
75
76  .wp-caption p.wp-caption-text {
77      font-size: 11px;
78      line-height: 17px;
79      margin: 0;
80      padding: 0 4px 5px;
81  }
82
83  .gallery-caption {
84    font-size: 0.8em;
85  }
```

Actualizamos y nuestro tema debería verse similar a la Figura 6

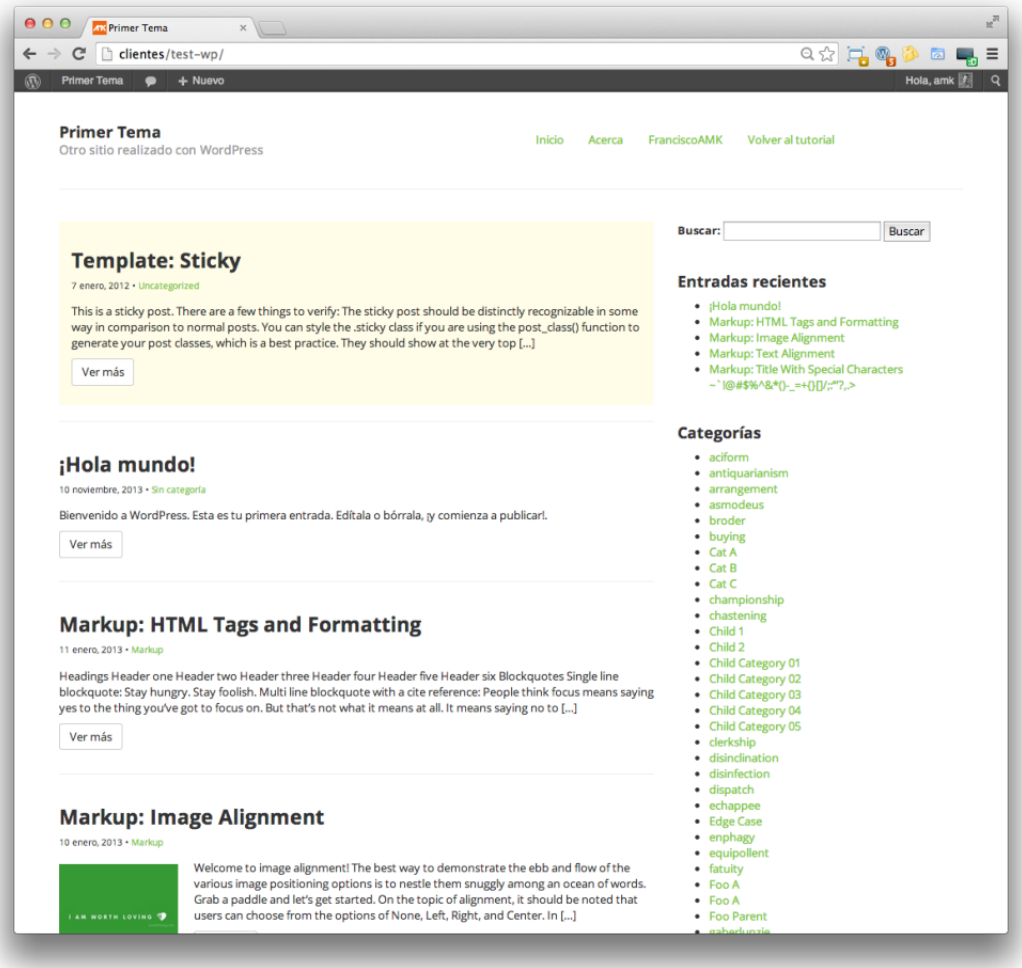

Figura 6: Tema con estilos

10. Javascript

Para cargar archivos javascript haremos algo muy similar a lo que hicimos para cargar archivos CSS, pero esta vez con las funciones `wp_register_script` y `wp_enqueue_script`.

La función `wp_register_script` recibe ciertos parámetros, muy similares a los que recibe `wp_register_style` :

1. **Nombre**: Un nombre para el script.
2. **URL**: la dirección donde se encuentra el script.
3. **Dependencia**: Si este script depende de otros para funcionar aquí añadimos su nombre.
4. **Versión:** La versión de este script.
5. **Posición:** Si se cargará en el pie de página (true) o en la cabecera del HTML (false).

Listing 31: Cargando JS en functions.php

```
1  //----------------------------------------------------------------------
2  // CARGANDO EL JAVASCRIPT DEL TEMA
3  //----------------------------------------------------------------------
4  //Creamos una función para cargar los scripts
5  function anp_theme_scripts() {
6
7    //Registramos el script de bootstrap
8    wp_register_script( 'bootstrap', get_stylesheet_directory_uri().'/js/
       bootstrap.min.js', array('jquery'), '3.0.0', true );
9
10   //Registramos el script de personalizado del tema
11   wp_register_script( 'anp-scripts', get_stylesheet_directory_uri().'/js/anp
       -scripts.js', array( 'jquery', 'bootstrap' ), '2.1', true );
12
13   //Ahora cargamos el script del tema.Como declaramos dependendencia de '
       jquery' y 'bootstrap', éstas cargaran de manera automática
14   wp_enqueue_script( 'anp-scripts' );
15 }
16
17 add_action( 'wp_enqueue_scripts', 'anp_theme_scripts' ); //Ejecutamos la
     función
```

Con estas líneas hemos registrado y cargado los archivos bootstrap.min.js, anp-scripts.js y además estamos cargando la librería jQuery.

Estamos cargando jQuery ya que en ambos scripts declaramos dependencia, sin embargo, no hemos registrado a jquery como un script, esto es así porque jQuery ya viene registrado al momento de instalar WordPress, por ello sólo basta con que declaremos dependencia y se cargará de manera automática.

Ahora podemos utilizar las utilizades de bootstrap, jQuery o escribir nuestro código JS para nuestro tema.

11. Detalle de un artículo (single.php)

Nuestro tema va tomando más forma cuando estamos viendo el listado de artículos, sin embargo, cuando entramos a ver el detalle de un artículo, veremos el mismo resumen que podemos ver desde el listado. Para corregir esto vamos a utilizar el archivo `single.php`.

Lo que haremos será copiar el mismo contenido que tenemos en `index.php` y pegarlo en `single.php`, y hacer algunos cambios:

- Eliminamos las comprobaciones para mostrar mensajes de acuerdo a la página que estemos viendo.

- Cambiamos el título `h2` por `h1`.
- Quitamos la llamada a la imagen destacada.
- Reemplazamos `the_excerpt` por `the_content`, es decir, reemplazamos el extracto por el contenido completo del artículo.
- Usamos la función `wp_link_pages` para mostrar una paginación en caso de que el artículo contenga la etiqueta `<!-nextpage->`.
- Mostramos el listado de etiquetas asociadas al artículo usando `the_tags`.
- Eliminamos los links a artículos recientes y artículos antiguos.
- Cargaremos la plantilla de comentarios con `comments_template`

Habiendo realizado estos cambios nuestro single.php quedaría así:

Listing 32: single.php

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

<section class="container">

  <div class="row">

    <div class="col-md-8">

      <?php //El loop básicamente comprueba si hay posts para mostrar, luego mientras haya posts cargará cada uno de ellos usando el esquema que se ve a continuación
      if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

        <article <?php post_class('clearfix'); //Carga las clases específicas del post y agrega la clase clearfix ?>>

          <header>
            <h1><?php the_title(); //Muestra el título del post ?></h1>
            <div class="meta">
              <?php the_time(get_option('date_format')); //Muestra la fecha de publicación del artículo ?>
              &bull;
              <?php the_category(', '); //Muestra enlaces a las categorías separados por coma ?></div>
          </header>

          <?php //En esta parte muestra el contenido del post
          the_content();

          //Muestra una paginación si es que el post tiene la etiqueta <!--nextpage-->
          wp_link_pages(array(
            'before'           => '<p class="paginacion">' . __('Páginas', 'anp')
          ));

          //Muestra las etiquetas asociadas al artículo
          the_tags(__('Etiquetas: ', 'anp'), ', '); ?>

        </article>

        <hr />

      <?php endwhile; //Acá termina la estructura con la que se presentará cada post
```

```
38        else: //Ahora bien, si no hay artículos para mostrar entonces cargará
              lo siguiente  ?>
39
40          <article>
41
42            <header>
43              <h1><?php _e('Este contenido no está disponible', 'anp'); //Un
                simple título ?></h1>
44            </header>
45
46            <?php get_search_form(); //Llama un formulario de búsqueda  ?>
47
48          </article>
49
50        <?php endif; // Aquí termina el loop?>
51
52        <?php //Cargamos la plantilla de comentarios
53        comments_template(); ?>
54
55      </div>
56      <div class="col-md-4">
57        <?php get_sidebar(); ?>
58      </div>
59
60    </div>
61
62  </section>
63
64  <?php get_footer(); ?>
```

12. Comentarios

En el código anterior usamos la función `comments_template` para cargar la plantilla de comentarios, es decir el archivo `comments.php`.

Hasta la versión 2.9 de WordPress podíamos prescindir de este archivo y en su lugar WordPress cargaría una plantilla predeterminada, sin embargo desde la versión 3.0 en adelante es obligación que el tema cuente con este archivo.

El archivo `comments.php` que usaremos en nuestro tema es el siguiente:

Listing 33: comments.php

```
1   <div id="comments">
2
3     <?php
4     //-----------------------------------------------------------------------
5     // Evita que carguen directamente este archivo comments.php
6     //-----------------------------------------------------------------------
7     if (!empty($_SERVER['SCRIPT-FILENAME']) && basename($_SERVER['SCRIPT-
          FILENAME']) == 'comments.php') {
8       die(__('Sabes que no puedes acceder a esta página.', 'anp'));
9     }
10
11    //-----------------------------------------------------------------------
12    // Si el artículo está protegido muestra un texto
13    //-----------------------------------------------------------------------
14    if (post_password_required()) : ?>
15      <p><?php _e( 'Este artículo está protegido por contraseña.', 'anp'); ?><
            /p>
```

```
16      </div>
17      <?php return; ?>
18    <?php endif;
19
20    //-----------------------------------------------------------------------
21    // Si es que hay comentarios, se muestran
22    //-----------------------------------------------------------------------
23    if (have_comments()) : ?>
24
25        <?php //Muestra un titulo para los comentarios, hay 3 opciones: cuando
              no hay, cuando hay un solo comentario y cuando hay 2 o más  ?>
26        <h3><?php comments_number(__('Comenta. Sé un pionero:', 'anp'), __('
              Hay sólo 1 comentario. Yo sé que quieres decir algo:', 'anp'), __(
              '% comentarios. Quieres agregar algo?:', 'anp')); ?></h3>
27
28        <ol id="comments-list">
29          <?php wp_list_comments('avatar_size=40'); //Muestra el listado de
                comentarios y define el tamaño del avatar ?>
30        </ol>
31
32        <?php //Si es que los comentarios están paginados, muestra links para
              comentarios antiguos y recientes
33        if (get_comment_pages_count() > 1 && get_option('page_comments')) : ?>
34
35          <ul class="pager">
36
37            <li class="previous">&larr; <?php previous_comments_link(__( '
                  Comentarios antiguos', 'anp')); ?></li>
38            <li class="next"><?php next_comments_link(__( 'Comentarios
                  recientes', 'anp')); ?> &rarr;</li>
39
40          </ul> <!-- end .pager -->
41
42        <?php endif; ?>
43
44    <?php
45    //-----------------------------------------------------------------------
46    // Si no hay comentarios y los comentarios están cerrados muestra un texto
47    //-----------------------------------------------------------------------
48    elseif (!comments_open() && !is_page() && post_type_supports(get_post_type
          (), 'comments')) : ?>
49
50      <p><?php _e( 'No se permiten comentarios en este artículo.', 'anp');?></
            p>
51      </div>
52      <?php return; ?>
53
54    <?php endif;
55
56    //-----------------------------------------------------------------------
57    // Muestra el formulario de comentarios
58    //-----------------------------------------------------------------------
59    comment_form();
60
61    ?>
62
63  </div><!-- end #comments-area -->
```

Aunque aparente ser un archivo complejo, básicamente lo que hace es realizar ciertas comprobaciones, como si el artículo está protegido por contraseña o si los comentarios están permitidos o no antes de mostrar el listado de comentarios y el formulario.

Para dar una mejor apariencia a los comentarios en nuestro sitio agregaremos lo siguiente a `style.css`:

Listing 34: Estilos para los comentarios

```
 /************ COMENTARIOS ************/

#comments-list {
  padding-left: 0;
  list-style: none;
}
  #comments-list .comment-body {
    background: #f2f2f2;
    padding: 15px;
    margin-bottom: 10px;
  }

  #comments-list .comment-author {
    margin-bottom: 15px;
  }

  #comments-list .comment-meta {
    font-size: 0.8em;
    margin-bottom: 10px;
  }

  #comments-list .children {
    list-style: none;
  }

  #comments-list .bypostauthor .comment-body {
    background: #e3e3e3;
  }

#commentform {

}

  #commentform input[type="text"],
  #commentform input[type="email"],
  #commentform textarea {
    width: 100%;
  }
```

Con estos cambios, nuestro artículo “Hola Mundo” y sus comentarios se deben ver así, Figura 7:

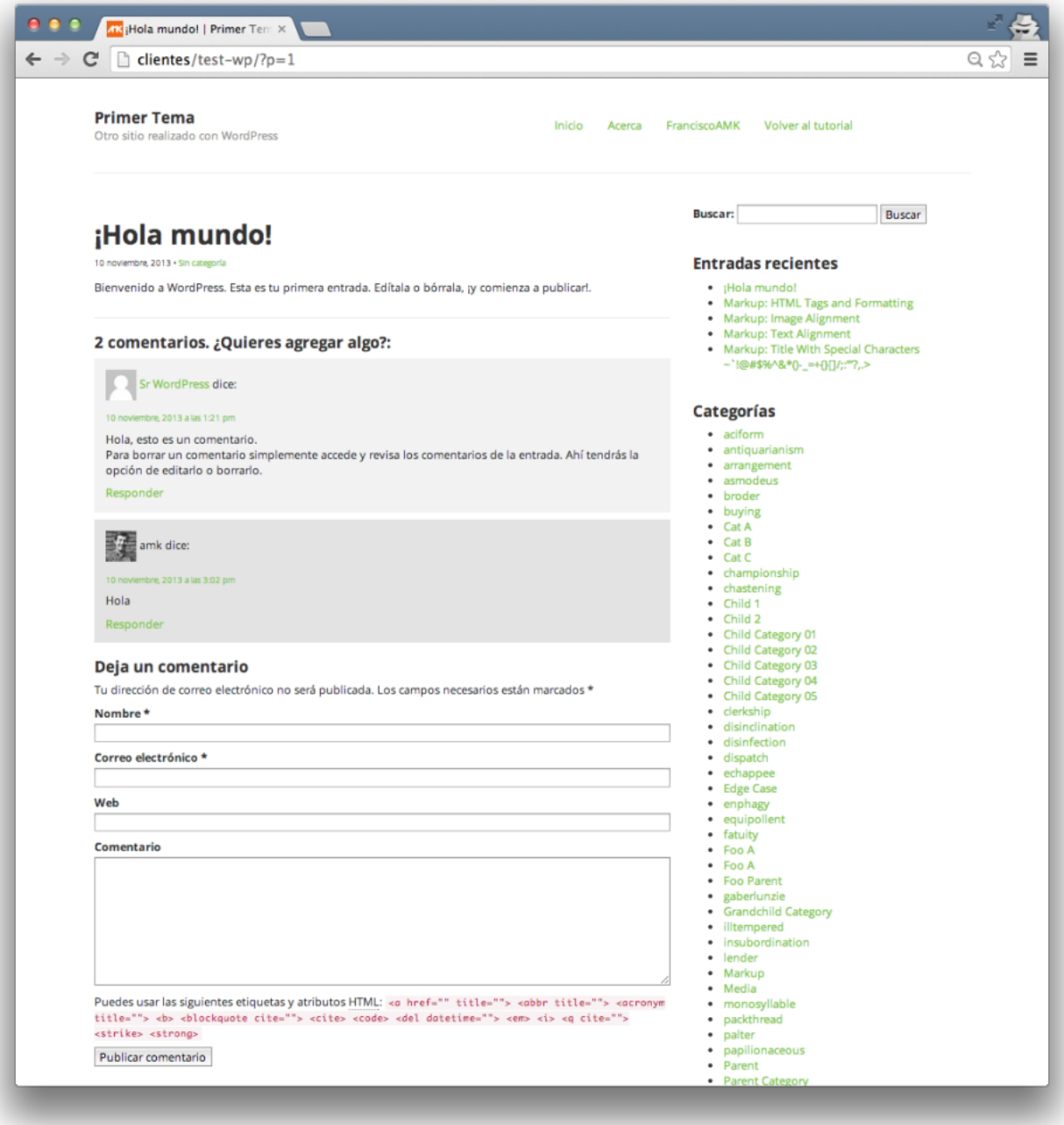

Figura 7: Detalles del artículo con comentarios

13. Detalle de una página, page.php

Para definir como se mostrará una página cada vez que entremos a ver el detalle deberemos utilizar el archivo `page.php`. Este archivo en esencia es bastante similar a `single.php` , por lo que utilizaremos el mismo contenido, realizando 2 cambios:

- Eliminamos el `div` con la clase meta en el cual mostramos la fecha de publicación y las categorías.
- Eliminamos el listado de etiquetas asociadas.

Luego de realizar estos cambios nuestro `page.php` quedaría así:

Listing 35: page.php

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

<section class="container">

  <div class="row">

    <div class="col-md-8">

      <?php //El loop básicamente comprueba si hay posts para mostrar, luego
          mientras haya posts cargará cada uno de ellos usando el esquema
         que se ve a continuación
      if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

        <article <?php post_class('clearfix'); //Carga las clases especí
            ficas del post y agrega la clase clearfix ?>>

          <header>
            <h1><?php the_title(); //Muestra el título del post ?></h1>
          </header>

          <?php //En esta parte muestra el contenido del post
          the_content();

          //Muestra una paginación si es que el post tiene la etiqueta <!--
              nextpage-->
          wp_link_pages(array(
            'before'           => '<p class="paginacion">' . __('Páginas', '
                anp')
          ));?>

        </article>

        <hr />

      <?php endwhile; //Acá termina la estructura con la que se presentará
          cada post
      else: //Ahora bien, si no hay artículos para mostrar entonces cargará
          lo siguiente  ?>

        <article>

          <header>
            <h1><?php _e('Este contenido no está disponible', 'amk'); //Un
                simple título ?></h1>
          </header>

          <?php get_search_form(); //Llama un formulario de búsqueda  ?>
```

```
40
41          </article>
42
43        <?php endif; // Aquí termina el loop?>
44
45        <?php //Cargamos la plantilla de comentarios
46        comments_template(); ?>
47
48      </div>
49      <div class="col-md-4">
50        <?php get_sidebar(); ?>
51      </div>
52
53    </div>
54
55  </section>
56
57  <?php get_footer(); ?>
```

Y las páginas se mostrarían así (Figura 8):

Figura 8: Detalles de páginas

Con todo lo visto en este tutorial tenemos una buena base para seguir diseñando nuestro tema. Mientras nuestro tema se vuelva más complejo serán necesarios más archivos de WP para cargar las diferentes páginas con diseños diferentes. Para esto es necesario consultar la jerarquía de páginas en el codex `https://developer.wordpress.org/themes/basics/template-hierarchy/`.

El resto del diseño depende de nuestras habilidades en CSS y JS, para esto tenemos a Bootstrap y debemos aprovechar todas sus utilidades.

Hay que tener en cuenta que nuestro tema debe ser probado en varios navegadores y diferentes tamaños de pantallas para asegurarnos que se comporta de la manera correcta y que la versión responsive se puede vsualizar corretamente. podemos también utilizar el plugin Theme-Check (`http://wordpress.org/plugins/theme-check/`) para comprobar funcionalidades y que el código esté bien escrito.

14. Referencias

- Developer.wordpress.org,. (2016). WordPress › Template Hierarchy | Theme Developer Handbook | WordPress Developer Resources. Retrieved 4 January 2016, from https://developer.wordpress.org/themes/basics/ter hierarchy/
- Aguilera G, F. (2013). Cómo crear un tema de WordPress (Parte 1) - Francisco Aguilera G.. Francisco Aguilera G.. Retrieved 4 January 2016, from http://franciscoamk.com/crear-tema-wordpress/
- Aguilera G., F. (2013). Cómo crear un tema de WordPress (Parte 2) - Francisco Aguilera G.. Francisco Aguilera G.. Retrieved 4 January 2016, from http://franciscoamk.com/crear-tema-wordpress-2/
- Jiménez, Y. (2014). Crear una plantilla de WordPress con Bootstrap. GeekPurple. Retrieved 4 January 2016, from http://geekpurple.com/crear-una-plantilla-de-wordpress-con-bootstrap/
- Hortin, A. (2015). Guía WP Fácil Manual de Wordpress. Madison Designs.
- Casabona, J. (2012). Building WordPress themes from scratch. [Lexington, Ky.]: Rockable.