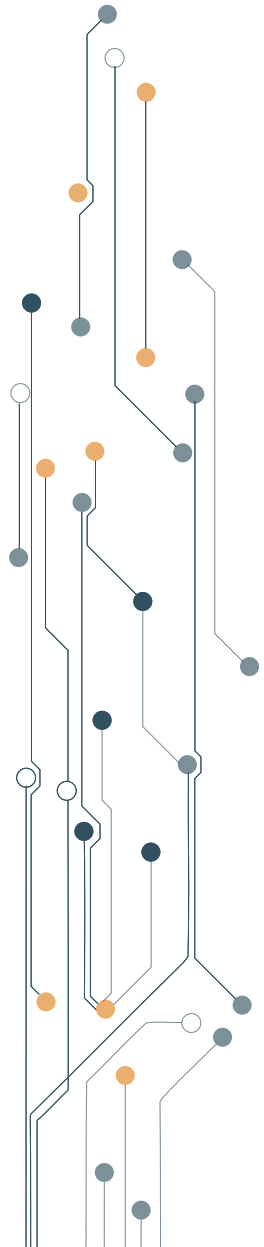
Posts personalizados

Telefónica
EDUCACIÓN DIGITAL

Índice



Posts personalizados

1 ¿Qué es un post?

Cuando usamos un CMS generalmente es porque queremos centrar los esfuerzos en el contenido. Afortunadamente WordPress tiene un sentido muy amplio de lo que puede ser ese contenido.

Como ya sabemos, el sistema cuenta con cinco tipos de posts diferentes por defecto:

- Entrada/Post ('post')
- Página ('page')
- Adjunto ('attachment')
- Revisión ('revision')
- Elemento de menú ('nav_menu_item')

Estos cinco tipos se puede ampliar con tipos de post personalizados (Custom post types)

Un ejemplo habitual de estos tipos de posts personalizados es el de los elementos de un portafolio. De hecho, muchas plantillas orientadas a diseñadores, fotógrafos y otros profesionales incluyen ya un tipo de post que permite registrar trabajos realizados. Esto permite contar con una apartado de portafolio o "selección de trabajos" que internamente funciona igual que los posts habituales, mostrándo un listado y permitiendo acceder a las fichas individuales de cada trabajo.

Podemos crear todos los tipos de post que queramos (y luego personalizar su comportamiento o presentación). Al hacerlo, nos aparecerán en nuestro panel de administración nuevas secciones permitiéndonos crear estos nuevos tipos de contenido.

Además, aunque todos los tipos de posts mantegan la misma estructura en la base de datos, podemos añadir campos personalizados a medida de cada tipo[1].

De la misma forma podremos añadir taxonomías a las ya existentes (categorías y etiquetas) que nos ayuden a organizar nuestro contenido.

1 Se almacenarán en la tabla postmeta, asociados al post en cuestión.

2 Tipos de post personalizados (Custom Post Types)

Como ya sabemos, WordPress cuenta con 5 tipos de posts predefinidos: entradas, páginas, adjuntos, revisiones y elementos de menú. Aunque para muchas webs pequeñas estos puedan ser suficientes, es común necesitar de otros como: productos, personas, testimonios, ... incluso elementos que queramos almacenar sin mostrar a los visitantes de la web.

La función que nos permite registrar un nuevo tipo de post personalizado es register_post_type[2], que acepta dos argumentos. El primero, obligatorio, es el nombre que le daremos internamente al tipo de post[3]. El segundo es un array (o string) que puede incluir una gran cantidad de parámetros como el nombre que se mostrará al usuario en el menú, su plural, una descripción, un icono...

```
register_post_type(
    string $post_type,
    array|string $args = array()
)
```

Para comprobar su funcionamiento, vamos a crear un tipo personalizado para guardar información sobre personas, ya sean los miembros de una empresa, de un grupo artístico o de cualquier otro equipo.

Más tarde veremos que podemos resgitrar nuestros tipos en plugins, pero en esta ocasión vamos a registrar este tipo en el archivo functions.php de nuestro tema[4].

Registraremos nuestro tipo usando la función de la siguiente forma:

```
register_post_type( 'personas' );
```

Pero para que Wordpress llame a la función y lo haga al inicio (tras inicializar el núcleo), debemos llamarla desde una función y asociar esta al hook 'init' de la siguiente forma:

/wp-content/themes/twentysixteen_child/functions.php

```
<?php
```

2 https://developer.wordpress.org/reference/functions/register_post_type/

3 De máximo 20 caracteres alanuméricos y guiones (medios y bajos)

4 Recuerda usar un tema propio aunque sea un tema hijo de otro. En el siguiente capítulo veremos como crear plugins y podrás mover facilmente a uno de ellos si no quieres que dependa del tema.

```
add_action( 'init', 'telwp_register_custom_post_
types' );

function telwp_register_custom_post_types() {
    register_post_type( 'personas' );
}
```

Hemos creado una función para registrar nuestros tipos personalizados, que por ahora es solo uno. Le hemos puesto un nombre que deja claro lo que hace, y al que le hemos añadido un prefijo 'telwp' que nos ayudará a distinguir las funciones de nuestro código de las del núcleo o desarrolladas por terceros.

Ahora si actualizamos el WP-Admin... no veremos ningún cambio, ya que por defecto los tipos que se crean no son públicos.

Podemos indicarle que debe ser público de la siguiente forma:

```
function telwp_register_custom_post_types() {
    register_post_type( 'personas',
        array(
            'public' => true
        )
    );
}
```

Ahora al actualizar nuestro panel de gestión veremos una nueva opción en el menú lateral (debajo de "Comentarios"), aunque tiene el nombre de "Entradas" que da lugar a confusión. Añadamos más opciones en nuestro array para indicar, entre otras cosas, el nombre de nuestro nuevo tipo.

```
function telwp_register_custom_post_types() {
    register_post_type( 'personas', array(
        'public' => true,
        'labels' => array(
            'name' => 'Personas',
        ),
        'menu_icon' => 'dashicons-id',
        'supports' => array('title', 'editor',
'excerpt', 'thumbnail', 'page-attributes')
    ));
}
```

Tras esta modificación podremos ver que el menú del panel administración ya muestra la etiqueta "Personas" acompañada de un icono.

Hemos indicado solo el nombre en plural, aunque podríamos haber

indicado más etiquetas para cambiar, por ejemplo, el texto del botón "Añadir nueva"[5]. Como icono del menú hemos indicado el nombre de uno de los iconos de dashicons[6], que es la fuente de iconos oficial de Wordpress. También podríamos pasarle la ruta a una imagen SVG o escribir la cadena 'none' para omitir el icono (que podríamos añadir via CSS).

Con la clave "supports", indicamos los campos que serán visibles al editar nuestro tipo personalizado. Si procedemos a añadir una nueva "persona", veremos que podemos incluir el título, el texto principal mediante el editor, el extracto, una imagen y el orden de la entrada respecto al resto del mismo tipo.

Este último campo nos permitirá controlar el orden en el que queremos que aparezcan las personas, y aparece al solicitar los atributos de página (page attributes). Si fueramos a representar un organigrama, podríamos haber activado la jerarquía.

```
register_post_type( 'personas', array(
    ...,
    'hierarchical' => true,
    ...
```

La jerarquía añade un nuevo atributo de página para seleccionar el post que actúa como padre[7] si fuera el caso.

Ya podemos incluir los datos de varias personas y acceder a sus "fichas" desde la web. Si al hacerlo obtienes un error 404 (página no encontrada) es posible que hayas modificado el nombre del custom type y no se estén generando correctamente los enlaces permanentes (permalinks). Una forma sencilla de solucioanrlo es acudir a Ajustes > Enlaces permanentes, cambiar el ajuste por otro y guardar. Acto seguido podemos volver a establecer el ajuste que teníamos. Esto volverá a crear los enlaces.

5 Las diferentes etiquetas que puedas incluir en el array están documentadas en https://developer.wordpress.org/reference/functions/get_post_type_labels/

6 https://developer.wordpress.org/resource/dashicons/

7 Es preciso que exista al menos otro post del mismo tipo para que se muestre la opción.

3 Taxonomías personalizadas (Custom Taxonomies)

Las taxonomías son las formas de agrupar elementos respondiendo a criterios de diversa índole. Por defecto, WordPress cuenta con etiquetas y categorías, pero podemos agregar diferentes taxonomías a cada uno de nuestros tipos de contenidos para definir como se relacionan entre si. Una taxonomía puede organizar un único tipo de post o varios y pueden tener una organización jerárquica (como las categorías) o no (como las etiquetas).

Las taxonomías se almacenan en la tabla term_taxonomy, mientras que los diferentes términos que pertenecen a ellas se registran en la tabla terms y la relación entre unos y otros en term_relationships.

Para ver como podemos crear nuestras propias taxonomías, vamos a incluir una, llamada "departamento", para nuestras "personas". Eso nos permitirá poder mostrar solo a parte del equipo en un momento dado. Por ejemplo, si creamos páginas para hablar de los distintos departamentos de una empresa, podremos incluir las fichas del equipo que forma parte de ese departamento.

Usaremos la función register_taxonomy[8] indicando, al menos, el nombre (interno) que le daremos a la taxonomía y el tipo de objeto (u objetos) que la usarán. El tercer parámetro, opcional, es una lista de argumentos, de los cuales solo usaremos una pequeña parte en en ejemplo:

```
add_action( 'init', 'telwp_define_department_
taxonomy' );
function telwp_define_department_taxonomy() {
    register_taxonomy(
        'department',
        'personas',
        array(
            'hierarchical' => true,
            'label' => 'Departamentos',
            'query_var' => true,
            'rewrite' => true
        )
    );
}
```

8 https://developer.wordpress.org/reference/functions/register_taxonomy/

Hemos decidido que los departamentos tengan jerarquía y hemos definido el nombre a mostrar.

También permitimos que se pueda usar la taxonomia en la petición añadiendo a la url "?department=nombre_del_departamento". También podríamos haber pasado el valor "false" para impedirlo o un string para indicar el nombre de la clave a usar en lugar del propio de la taxonomía.

La última opción activa las reglas de reescritura para la taxonomía, de forma que si usarmos urls "amigables" podramos solicitar la página mi_web.com/department/nombre_del_departamento.

Una vez registrada la taxonomía, aparecerá en el menú del tipo (o tipos) de post al que pertenece, permitiendo la gestión de los términos que se incluirán en esa clasificación.

4 Campos personalizados (Custom Fields)

Registrando campos personalizados

Al crear nuestro nuevo tipo de post, indicamos los campos que el usuario podría manejar desde el panel de control. Si hubieramos incluido entre ellos los 'custom-fields', daríamos la oportunidad al usuario de añadir campos extra a las fichas de las personas.[9]

Sin embargo, si queremos solicitar unos campos específicos en un tipo de post específico, podemos registrarlos con la función register_meta[10]. La función que realice esta tarea la asociaremos al hook 'init'.

```
add_action( 'init', 'telwp_register_meta_fields' );
function telwp_register_meta_fields() {
   register_meta( 'personas', 'telwp_twitter', [
      'description'       => 'Nombre en Twitter',
      'single'            => true,
      'sanitize_callback' => 'sanitize_text_field',
      'auth_callback'     => 'telwp_custom_fields_
auth_callback',
      'show_in_rest'      => true
      ]);
}
```

Como vemos, le indicamos el tipo de post ('personas') al que queremos asociar el dato, y el nombre que le daremos a éste (telwp_twitter). Además, le pasamos una serie de argumentos indicando:

- Una descripción (description).
- Si las entradas solo pueden tener un valor con esta clave (single).
- El nombre de la función/método que procesará el texto para validarlo y "escaparlo" (sanitize_callback).
- El nombre de la función a la que se llamará cuando intente añadir, editar o eliminar el dato, para comprobar si se permite la acción (auth_callback).
- Si se trata o no de un dato público (show_in_rest).

9 Estos campos extra se almacenan en la tabla postmeta de nuestra base de datos.

10 https://developer.wordpress.org/reference/functions/register_meta/

La función sanitize_text_field[11] es propia de núcleo de WordPress, pero la que comprobará la si se permite la edición debemos escribirla.

```
function telwp_custom_fields_auth_callback(
$allowed, $meta_key, $post_id, $user_id, $cap,
$caps ) {

  if( 'personas' == get_post_type( $post_id ) &&
current_user_can( 'edit_personas', $post_id ) ) {
    $allowed = true;
  } else {
    $allowed = false;
  }

  return $allowed;
}
```

En esta función podemos hacer comprobaciones basadas en el usuario, el post y las capacidades ('capabilities'). Nos limitamos a comprobar si el tipo de post que se está editando es del tipo "personas" y si el usuario tiene permiso para editar dicho tipo de post.

Añadiendo la caja y el formulario

Aunque hemos registrado un campo extra, aún debemos indicar donde y como se mostrará en el gestor cuando vayamos a crear o editar una entrada. Para ello crearemos una caja personalizada (custom metabox) en el formulario con la función add_meta_boxes.

```
add_action( 'add_meta_boxes', 'telwp_meta_boxes' );
function telwp_meta_boxes() {
    add_meta_box( 'telwp-rrss-box', 'Redes
sociales', 'telwp_meta_rrss_callback', 'personas',
'side', 'high' );
}
```

Como vemos en la documentación de la función[12], los argumentos que le pasamos son: id, título, callback, pantalla donde se mostrará, el contexto (zona donde debe aparecer) y la prioridad respecto a otras cajas.

La función callback (que es llamada tras la anterior) será la que se encargue de "pintar" los campos de formulario en nuestra recién

11 https://developer.wordpress.org/reference/functions/sanitize_text_field/

12 https://developer.wordpress.org/reference/functions/add_meta_box/

estrenada caja para "Redes sociales".[13]

```
function telwp_rrss_box_callback( $post ) {
     wp_nonce_field( 'telwp_rrss_box', 'telwp_rrss_
box_noncename' );
     $post_meta = get_post_custom( $post->ID ); ?>

     <p>
          <label class="label" for="telwp_
twitter">Twitter</label>
          <input  name="telwp_twitter"
id="telwp_twitter" type="text"
value="<?php echo esc_attr( get_post_meta( $post-
>ID, 'telwp_twitter', true ) ); ?>">
     </p><?php

}
```

Primero usaremos la función wp_nonce_field[14] que generará un campo oculto con una cadena de texto que usará después para validar que la petición llega de esa página, añadiendo cierta seguridad.

Después rescatamos la información guardada en la base de datos (por si se trata de una edición de datos ya existentes) e imprimimos el formulario prestando especial atención al nombre del campo input (pues es la clave del array $_POST, que necesitaremos para procesar el formulario) y a su valor. Hemos usado la función esc_attr[15] para "escapar" los atributos HTML aunque al tratarse de un nombre de usuario en Twitter, no se espera código HTML en el texto.

Si accedemos a dar de alta o a editar a una persona, ya podremos comprobar la presencia de la caja de "Redes sociale" y el campo de texto que hemos añadido en ella.

Procesando el formulario

Ahora debemos registrar la función que se debe añadir al procesamiento habitual del formulario al guardar el post.

Si estuvieramos añadiendo campos a un post de tipo "normal" en

13 Hemos puesto este nombre para poder aprovecharla en caso de querer añadir otras redes. además de Twitter

14 https://developer.wordpress.org/reference/functions/wp_nonce_field/

15 https://developer.wordpress.org/reference/functions/esc_attr/

lugar de a un tipo personalizado, usaríamos el hook 'save_post'. En nuestro caso usaremos 'save_post_personas'.[16]

```
add_action( 'save_post_personas', 'telwp_save_
custom_fields' );
function telwp_save_custom_fields( $post_id ){
   if (! isset( $_POST['telwp_rrss_box_noncename'])
|| ! wp_verify_nonce( $_POST['telwp_rrss_box_
noncename'], 'telwp_rrss_box' ) ) {
      return;
   }
   if ( isset( $_POST['telwp_twitter'] ) && $_
POST['telwp_twitter'] != "" ) {
     update_post_meta( $post_id, 'telwp_twitter',
$_POST['telwp_twitter'] );
   } else {
     delete_post_meta( $post_id, 'telwp_twitter' );
   }
}
```

16 save_post_{$post->post_type} - https://developer.wordpress.org/reference/hooks/save_post_post-post_type/

Lo primero que hacemos en la función que procesará el envío del formulario es comprobar que existe el "nonce" y que es válido. En caso contrario salimos de la función con una sentencia return.

Después comprobamos que hemos recibido un valor para nuestro campo "telwp_twitter" distinto a una cadena vacía y lo guardamos. Si el valor fuera una cadena vacía, entendemos que el usuario no quiere incluir el dato o que lo quiere borrar (en caso de que ya existiera).

Con esto tendríamos listo nuestro campo personalizado. En este caso lo hemos añadido a un tipo de post propio, pero podríamos actuar sobre uno de los tipos por defecto o sobre un tipo creado por otro plugin.

Al controlar como se generan y procesan los campos de formulario, podríamos añadir otros tipos de input, incluyendo seleccionables que usaran otros datos almacenados en nuestra web.

Ten en cuenta, que cada campo personalizado se almacena en un registro de la tabla postmeta. Cuando se van a registrar distintos campos de este tipo, puede ser conveniente, almacenarlos en la base de datos como un único campo (un array) independientemente de como de muestren al usuario.

Telefónica

EDUCACIÓN DIGITAL