
**Universidad de Valladolid**

# Escuela de Ingeniería Informática

**TRABAJO FIN DE GRADO**

Grado en Ingeniería Informática
( Mención en Tecnologías de la Información )

# Eficiencia y Flexibilidad en Desarrollo Web: Creación de un Framework Basado en Módulos Dinámicos para WordPress

Autor:

**Fernando Juan Mayor Rocher**

Tutor:

**Benjamín Sahelices Fernández**

# Agradecimientos

Me gustaría expresar mi más sentido agradecimiento a todas las personas que me han acompañado y apoyado en la realización de este Trabajo Fin de Grado. En primer lugar, a mi tutor, Benjamín, por su paciencia y sus valiosos consejos que han servido de motivación y de guía en el día a día. A mi familia, amigos y seres queridos por no darse nunca por vencido conmigo, apoyarme en los momentos difíciles y quererme tal y como soy por encima de todo. A Soraya, a David y a mis compañeros de trabajo, por ser mucho más que compañeros y jefes: auténticas guías a nivel profesional. Y por haber confiado en mí durante los cinco años que llevo trabajando con ellos.

# Resumen

El objetivo de este Trabajo Fin de Grado es abaratar los costes de desarrollo para el equipo de WordPress de la empresa de software en la que su autor trabaja (Brooktec) mediante el estudio de los módulos más solicitados por parte de los clientes, así como su unificación y desarrollo en un repositorio de código del cual poder hacer un fork en próximos desarrollos para clientes reales. Por lo tanto, la herramienta consiste en un repositorio de código listo para ser bifurcado en otros repositorios sobre los que trabajar a modo de framework modular de alta flexibilidad y customización, pero de rápida puesta en producción. Se seguirá una metodología Scrum, para la planificación y desarrollo de tareas.

# Abstract

The aim of this Final Degree Project is to reduce development costs for the WordPress team at the software company where its author works (Brooktec) by studying the modules most requested by clients and unifying and developing them in a code repository from which a fork can be made for future developments for real clients. Therefore, the tool consists of a code repository ready to be forked into other repositories to work as a modular framework with high flexibility and customization, but with rapid production deployment. A Scrum methodology will be followed for task planning and development.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# 1. Introducción

En este proyecto se propone la creación de un *framework* modular en forma de tema de WordPress que sea altamente flexible y que permita ahorrar costes y tiempo al equipo de desarrollo de una empresa.

Se trata de facilitar el trabajo al desarrollador de forma que prácticamente solo tenga que decidir qué módulos del *framework* serán necesarios en los casos concretos de la web en la que se utilizará dicho sistema. Y que de esta manera solo se tenga que preocupar de desarrollar las hojas de estilos específicos para la web que necesita el cliente final. De esta forma se le ahorrará todo el trabajo de definir y programar todos los campos de contenido de cada módulo y de hacer un *render* de dichos módulos en el frontal con los estilos básicos para su correcto funcionamiento en escritorio y dispositivos móviles.

# Capítulo 2

# 2. Contexto y Motivaciones

## 2.1. Contexto

### 2.1.1. Conceptual: Sistemas de información CMS

CMS son las siglas de Content Management System, o lo que es lo mismo, Sistema de gestión de contenidos. Un CMS es un programa desarrollado para que cualquier usuario pueda entrar y gestionar contenidos de una web con facilidad y sin conocimientos de programación. No todas las páginas web son iguales y para ello tenemos tantos CMS como tipos de webs. Los hay para blogs, páginas corporativas, inmobiliarias, noticias, revistas, contenido multimedia, etc. Algunos CMS son desarrollos con licencia gratuita y otros son de pago. En líneas generales, un CMS permitiría administrar contenidos en un medio digital y para el caso particular que nos ocupa un CMS permitiría gestionar los contenidos de una web. Dicho de otra forma, un CMS es una herramienta que permite a un editor crear, clasificar y publicar cualquier tipo de información en una página web.

Generalmente, los CMS trabajan contra una base de datos de modo que el editor utiliza esta misma incluyendo la información o editando la existente de forma transparente. Pongamos un ejemplo. Imaginemos un periódico o cualquier otra página medianamente compleja. Principalmente aquellas que tienen que ser actualizadas diariamente o varias veces al día, donde además las personas que editan o tienen la información que hay que incluir no tienen conocimientos de informática. A estos redactores se les tiene que facilitar el trabajo con una herramienta que les permita subir informaciones a la web y clasificarlas para que aparezcan en el lugar correcto. Por supuesto que estas personas no deben preocuparse por el código de la página o con las peculiaridades de la plataforma donde se aloja la web. Ellos solo deben concentrarse en escribir las noticias o cualquier tipo de contenido y luego subirlas a la página por un sistema intuitivo y fácil. Una vez publicadas y clasificadas las informaciones deben aparecer en el página web automáticamente en los lugares donde haya decidido el editor. Entonces aquí es donde entra el CMS. Es el CMS quien va a permitir a estos editores o creadores de contenido actualizar la página web y moldearla a su medida a golpe de ratón sin necesidad de adquirir conocimientos de programación para ello.

Al haber gran variedad de CMS especializados en diferentes tipos de portales plataformas web, cada usuario puede crear su propio portal web a través de un CMS sin necesidad de escribir una sola línea de código: simplemente instalando el CMS que se quiera usar y personalizándolo a su gusto. Esto quiere decir que la mayoría de los CMS, por no decir todos los CMS, no solo gestionan contenidos de una web, sino que muchos de ellos, o los más utilizados, son mucho más complejos ya que permiten personalizar una web y modelarla a gusto personal mediante plantillas, e incluso modificarlas o crear nuevas personalizadas si se tiene un mínimo de conocimiento de programación en *frontend*. Además, permiten administrar roles de usuarios para

que exista mayor complejidad en el portal web y una de las características más potente en los grandes CMS más usados es su gran variedad de plugins o módulos que pueden ser instalados para aumentar tanto la funcionalidad del portal web como su control. Poner una tienda al portal web o querer controlar estadísticas del sitio, así como crear tipos de contenidos más complejos, entre otros, son ejemplos de *plugins*. Los plugins en los CMS también ahorran la necesidad de crear un elemento concreto que se requiera en el portal web con una simple instalación que posteriormente permitirá también personalizarla según el gusto propio y las necesidades de la web.

Una de las grandes ventajas de los CMS es que ahorra una cantidad enorme de esfuerzo y dinero, ya que no hay que crear el gestor de contenidos desde cero. Así, el trabajo puede ser mejor focalizado en el desarrollo de las particularidades de la web, que en tener que desarrollar todo un sistema *backend* que permita al usuario poder gestionar su contenido.

Mencionemos cuatro CMS mundialmente conocidos por ser los más utilizados y por sus características:

**WordPress:** un CMS que el autor considera semigratuito, pues ofrece lo básico gratis, así como una vasta cantidad de *plugins*. Sin embargo, ciertos elementos como plantillas más elaboradas o algunos *plugins* de empresas son de pago. Su curva de aprendizaje es muy simple debido a que su interfaz ayuda fácilmente a gestionar y crear el sitio web con pocos pasos. Este CMS es *Open Source*, como también lo es el siguiente mencionado [1].

**Drupal**: es quizá el CMS ofrece mayor flexibilidad y manipulación para el usuario. Al ser *Open Source*, tiene una gran comunidad que desarrolla, actualiza y mejora Drupal día a día. El usuario mismo puede crear módulos según su gusto o explorar el código de Drupal y tener un CMS muy personalizado. La desventaja es que este CMS tiene una curva de aprendizaje bastante mayor que WordPress, pero es, sin duda, una muy buena opción si se tienen conocimientos de informática y se quiere elaborar una plataforma web gracias a todo lo que ofrece [2].

**Moodle**: es un CMS empleado por institutos y universidades, centrado en el ámbito educativo. Este CMS trae ya montado un sistema de roles y una base para que cualquier instituto lo pueda implementar y remodelar y tener así un sistema informático donde gestionar su sistema académico [3].

**Joomla**: también debe ser mencionado como un CMS bastante genérico, situado entre medias de Wordpress y Drupal. No es tan complejo como Drupal ni tan limitado como lo es WordPress [4].

### 2.1.2. Tecnológico

WordPress es una herramienta para crear webs que está englobada dentro de lo que se conocen como CMS o gestores de contenido. Estos gestores de contenido no solo permiten crear una web de manera más fácil y sencilla que si se tuviese que crear desde cero, sino que además permiten que se pueda administrar desde una parte trasera o *dashboard*.

WordPress no solo es un gestor de contenidos más, sino que es el gestor de contenidos más utilizado en el mundo. De hecho, según las estadísticas, un 43% de las web creadas en todo el mundo están diseñadas bajo este CMS como muestran la figura 1 y la figura 2.

| | 2013 1 Jan | 2014 1 Jan | 2015 1 Jan | 2016 1 Jan | 2017 1 Jan | 2018 1 Jan | 2019 1 Jan | 2020 1 Jan | 2021 1 Jan | 2022 1 Jan | 2023 1 Jan | 2024 1 Jan | 2024 9 Jun |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | 68.2% | 64.8% | 61.7% | 56.6% | 53.3% | 51.3% | 45.3% | 43.1% | 38.3% | 33.8% | 32.3% | 31.4% | 30.8% |
| WordPress | 17.4% | 21.0% | 23.3% | 25.6% | 27.3% | 29.2% | 32.7% | 35.4% | 39.5% | 43.2% | 43.1% | 43.1% | 43.4% |
| Shopify | | 0.1% | 0.3% | 0.4% | 0.6% | 0.9% | 1.4% | 1.9% | 3.2% | 4.4% | 3.8% | 4.1% | 4.4% |
| Wix | <0.1% | 0.1% | 0.1% | 0.2% | 0.3% | 0.4% | 1.0% | 1.3% | 1.5% | 1.9% | 2.4% | 2.6% | 2.7% |
| Squarespace | <0.1% | 0.1% | 0.2% | 0.4% | 0.5% | 0.7% | 1.4% | 1.5% | 1.4% | 1.8% | 2.0% | 2.1% | 2.1% |
| Joomla | 2.8% | 3.3% | 3.3% | 3.3% | 3.4% | 3.2% | 3.0% | 2.6% | 2.2% | 1.7% | 1.8% | 1.7% | 1.7% |
| Drupal | 2.3% | 1.9% | 2.0% | 2.1% | 2.2% | 2.3% | 1.9% | 1.7% | 1.5% | 1.3% | 1.2% | 1.1% | 1.0% |
| Adobe Systems | | | | | | | | | | | 1.1% | 1.0% | 1.0% |
| PrestaShop | 0.3% | 0.4% | 0.5% | 0.6% | 0.6% | 0.6% | 0.8% | 0.7% | 0.5% | 0.5% | 0.7% | 0.8% | 0.8% |
| Webflow | | | | | | | 0.1% | 0.1% | 0.2% | 0.4% | 0.6% | 0.7% | 0.7% |
| Google Systems | | | | | | | | | | | 0.8% | 0.7% | 0.7% |
| Bitrix | 0.3% | 0.3% | 0.4% | 0.6% | 0.7% | 0.7% | 0.6% | 0.9% | 1.1% | 0.9% | 0.8% | 0.7% | 0.6% |
| OpenCart | | | 0.3% | 0.4% | 0.4% | 0.4% | 0.4% | 0.5% | 0.6% | 0.6% | 0.5% | 0.5% | 0.5% |

**Figura 1:** Datos de tendencias históricas en el uso de los principales sistemas de gestión de contenido desde enero de 2013. [5]

¿Por qué WordPress es tan popular? ¿Por qué lo usa tanta gente y tantas web a nivel mundial? Principalmente los motivos por los que utilizar WordPress son los siguientes:

- Tener una curva de aprendizaje muy baja respecto a otros gestores de contenidos.
- Ser un gestor de contenidos de código libre, es decir, que está en constante desarrollo por una comunidad de programadores detrás del gestor y, por tanto, siempre en constante actualización.

Es muy ampliable, pues se pueden hacer muchos tipos o cualquier tipo de web que se desee gracias precisamente a una especie de programas que se pueden instalar llamados *plugins*. Gracias a ellos, se pueden crear tiendas online, sistemas de reservas con alojamiento turístico, reservas o pedidos para restaurantes, academias o escuelas online, etc.

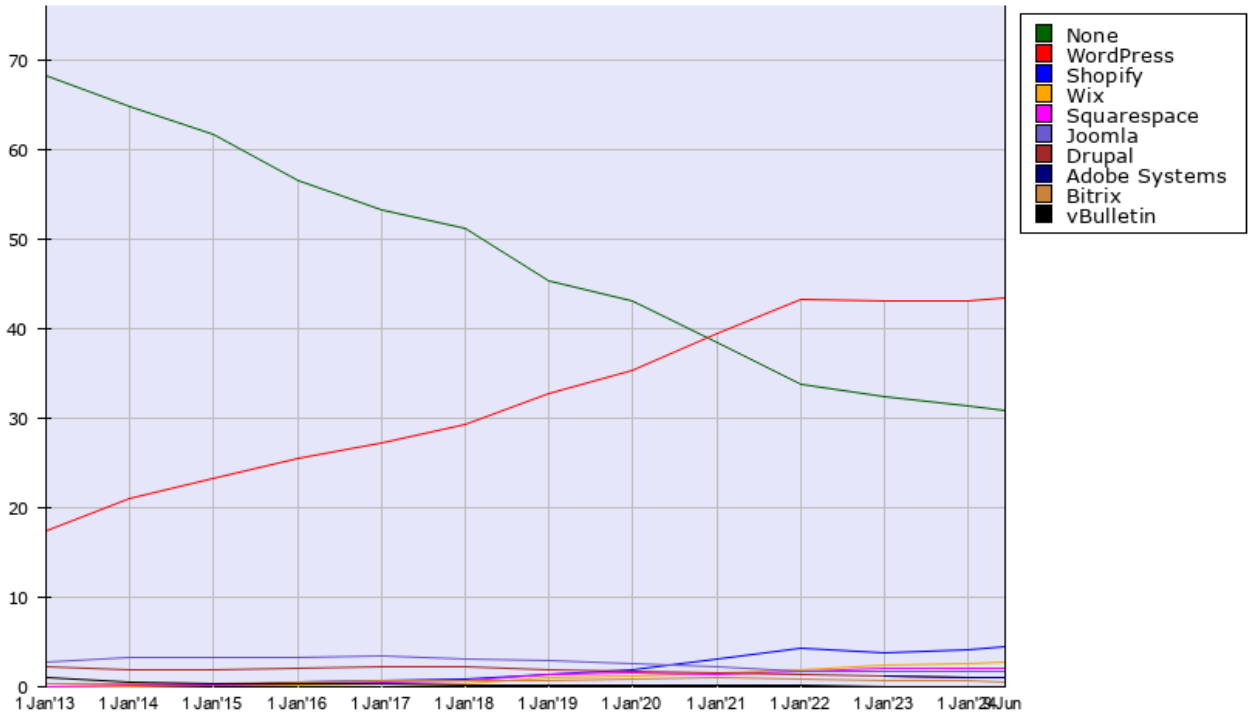



**Figura 2:** Gráfica de tendencias históricas en el uso de los principales cms desde enero de 2013. [5]

Es un sistema de gestión de contenidos (CMS) que resulta extremadamente sencillo de actualizar. En contraste, otros gestores de contenidos pueden representar un desafío considerablemente complejo, incluso para profesionales especializados en desarrollo web. De este modo, su contenido es fácilmente gestionable y modificable, permitiendo al usuario realizar cambios como la actualización de imágenes, la modificación de textos, o la inclusión de nuevos productos en una tienda en línea, sin necesidad de recurrir a un desarrollador.

Todas estas razones hacen que elegir WordPress frente a otros gestores de contenidos u otras herramientas de creación web sea una opción muy conveniente. Lo mejor de todo es que el único requisito para comenzar a utilizar WordPress, tanto como creador del sitio web como gestor de contenidos, es simplemente contar con un buen servicio de *hosting*.

Como dato interesante, existen dos plataformas basadas en el mismo software de código abierto, WordPress, pero con diferencias significativas en términos de funcionalidad, flexibilidad y control, WordPress.com y WordPress.org. Es importante aclarar que, para la realización de este proyecto, trabajaremos con WordPress.org. A continuación, se detallan las diferencias entre ambas plataformas:

**Alojamiento:** WordPress.com ofrece alojamiento gratuito y de pago para el sitio web, permitiendo su creación sin necesidad de buscar un servicio de alojamiento externo. En cambio, con WordPress.org, se debe buscar un propio servicio de alojamiento web, lo que brinda mayor flexibilidad y control sobre el sitio, con una amplia gama de proveedores de alojamiento disponibles.

**Limitaciones de personalización:** Con WordPress.com, la personalización del sitio web es limitada. Se tiene acceso a una selección de temas y *plugins*, pero no se pueden instalar temas o plugins personalizados de terceros. Por otro lado, al utilizar WordPress.org, se tiene control total sobre el sitio web. Es posible personalizarlo completamente a gusto, instalar temas y plugins de terceros, acceder y modificar el código fuente, y utilizar herramientas avanzadas de desarrollo.

**Monetización:** En la versión gratuita de WordPress.com, existen restricciones en cuanto a la monetización. No se permite mostrar anuncios en el sitio web a menos que se actualice a un plan de pago. En cambio, con WordPress.org no hay restricciones de monetización. Se pueden mostrar anuncios, vender productos o servicios, e implementar cualquier estrategia de monetización deseada.

**Mantenimiento:** WordPress.com se encarga del mantenimiento del software de WordPress, incluyendo actualizaciones de seguridad y copias de seguridad automáticas. En cambio, aunque WordPress.org ofrece actualizaciones regulares, el usuario es responsable de mantener el sitio web actualizado y de realizar copias de seguridad periódicas.

## 2.2. Motivación: Gutenberg vs. Visual Composer vs. sistema propio

La primera duda que puede surgir sobre la utilidad de este proyecto es la siguiente: ¿Por qué construir un sistema modular para WordPress si los propios desarrolladores de WordPress ya han creado uno oficial llamado Gutenberg? O bien, ¿por qué no utilizar plugins para la construcción de sitios con compositores visuales? En esta sección se responderán ambas preguntas.

La respuesta rápida a estas preguntas es: "por años de experiencia con los clientes". En la empresa en la que el autor trabaja (Brooktec) existen dos tipos de proyectos web: la creación y desarrollo de un nuevo sitio, y el mantenimiento de uno ya existente que se hereda.

**Desarrollos nuevos**

Cuando se hace un desarrollo nuevo, casi ninguno de los clientes quiere el sistema de módulos de Gutenberg, ya que lo consideran demasiado complejo de utilizar. Prácticamente la totalidad de ellos quieren soluciones más cerradas, en las que configurando unas pocas opciones en el *dashboard* el módulo se vea como se especificó en el diseño, sin necesidad de preocuparse por posicionar elementos o ajustar tamaños.

Gutenberg es el editor de bloques incorporado en WordPress a partir de la versión 5.0. Este editor ha sido objeto de mucha controversia entre la comunidad de WordPress debido a su enfoque en los bloques y a la forma en que reemplaza el editor clásico de WordPress [6].

Algunos de los motivos por los que los clientes no quieren utilizar el editor de bloques Gutenberg son los siguientes:

- **Problemas de compatibilidad:** Gutenberg es un editor relativamente nuevo, lo que significa que algunos temas y *plugins* de WordPress pueden no ser completamente

compatibles con él. Esto puede causar problemas con la visualización y la funcionalidad del sitio web.

- **Aprendizaje adicional:** Para los clientes que están acostumbrados al editor clásico de WordPress, la transición a Gutenberg puede requerir un período de aprendizaje adicional. Aunque la interfaz de usuario es intuitiva, puede llevar tiempo acostumbrarse al nuevo flujo de trabajo y a las características del editor.

- **Limitaciones de personalización:** Aunque Gutenberg ofrece más flexibilidad que el editor clásico de WordPress, algunas personalizaciones avanzadas pueden requerir conocimientos de codificación. Si bien esto no es un problema para los desarrolladores web, los usuarios menos técnicos como nuestros clientes pueden encontrar limitaciones o dificultades en la personalización de su sitio web.

**Mantenimientos de proyectos heredados**

Si bien es cierto que la construcción inicial de estos sitios pudo haber sido económica debido a la rapidez que ofrecen los sistemas de compositores visuales (como *plugins* del tipo Elementor o WPBakery), los problemas surgen cuando se desean hacer modificaciones específicas y personalizadas. Estos plugins tienen versiones gratuitas y de pago, pero independientemente de la versión utilizada, ninguno facilita la incorporación de código personalizado, lo que lleva a soluciones complicadas y poco elegantes, resultando en sitios difíciles y costosos de mantener. Además, las actualizaciones de WordPress y los *plugins* a menudo generan incompatibilidades. En algunos casos, los propios clientes solicitan desarrollar el sitio desde cero para deshacerse de estos compositores visuales y todas las dependencias que conllevan.

Por lo tanto, dados los argumentos anteriores, queda claro que ni utilizar Gutenberg ni utilizar un compositor visual de terceros es una opción ideal para el desarrollo de un nuevo proyecto en WordPress. Gutenberg no es deseado por los clientes debido a su complejidad, y los compositores visuales complican significativamente el mantenimiento del sitio a largo plazo.

El sistema modular desarrollado en este proyecto busca ofrecer una solución rápida al cliente, evitando los inconvenientes mencionados. Para ello, lo primero que se ha realizado es una investigación de cuáles son los módulos más solicitados por nuestros clientes.

# Capítulo 3

# 3. Análisis y Planificación

En esta sección se proporciona un listado de requisitos funcionales y no funcionales para el sistema de módulos desarrollado. Asimismo, se hace un estudio de los módulos requeridos y una planificación de las fases del desarrollo y el presupuesto estimado de costes.

## 3.1. Requisitos funcionales

Los requisitos funcionales de un sistema de software son descripciones detalladas de las funciones, servicios y operaciones que el sistema debe ser capaz de realizar. Estos requisitos especifican lo que el sistema deberá hacer en términos de tareas y servicios, bajo ciertas condiciones, y cómo debe reaccionar a entradas específicas.

**FN-1 - Descarga del sistema:** los desarrolladores de la empresa deben poder descargarse el sistema modular para acceder a sus funcionalidades.

**FN-2 - Creación de módulos:** los desarrolladores deben poder crear nuevos módulos personalizados para su uso en los nuevos sitios web, o para el evolutivo de sitios web ya existentes que utilicen este sistema.

**FN-3 - Edición de módulos existentes:** los desarrolladores deben poder modificar y editar los módulos existentes según las necesidades específicas de cada sitio web.

**FN-4 - Gestión de estilos:** los desarrolladores deben poder agregar estilos personalizados a los módulos existentes para adaptarlos a las necesidades de diseño de cada sitio web.

**FN-5 - Asignación de módulos a sitios web:** los desarrolladores deben poder asignar los módulos existentes o personalizados a los sitios web en construcción.

**FN-6 - Vista previa y prueba:** los desarrolladores deben poder previsualizar y probar los sitios web antes de desplegarlos en producción.

**FN-7 - Control de versiones:** el sistema debe proporcionar un mecanismo para controlar las diferentes versiones de los módulos y permitir a los desarrolladores revertir a versiones anteriores si es necesario.

## 3.2. Requisitos no funcionales

Los requisitos no funcionales de un sistema de *software* se refieren a los criterios que describen cómo debe ser el sistema, en lugar de las funciones específicas que debe realizar. Estos requisitos están orientados a la calidad y el comportamiento general del sistema, y suelen incluir aspectos relacionados con la usabilidad, el rendimiento, la seguridad, la fiabilidad, entre otros. Son esenciales para asegurar que el *software* funcione de manera efectiva en el entorno previsto y satisfaga las expectativas de los usuarios en términos de experiencia y eficiencia.

**RNF-1 - Modularidad:** el sistema debe ser altamente modular y flexible, permitiendo la fácil adición y modificación de módulos sin afectar a la funcionalidad existente.

**RNF-2 - Rendimiento:** el sistema debe ser eficiente en términos de rendimiento, permitiendo la rápida carga y respuesta de los sitios web creados utilizando los módulos.

**RNF-3 - Usabilidad:** el sistema debe ser intuitivo y fácil de usar para los desarrolladores, proporcionando una interfaz clara y bien estructurada.

**RNF-4 - Escalabilidad:** el sistema debe ser capaz de manejar un número creciente de módulos y sitios web sin degradar su rendimiento.

**RNF-5 - Seguridad:** el sistema debe implementar medidas de seguridad adecuadas para proteger la integridad de los datos y prevenir accesos no autorizados.

**RNF-6 - Documentación:** debe haber una documentación clara y completa que explique el funcionamiento y la configuración del sistema, facilitando su uso y mantenimiento.

## 3.3. Análisis de módulos requeridos

Como se ha comentado, se ha realizado un estudio para determinar, dentro de todos los proyectos WordPress de la empresa (al menos a los que el autor ha tenido acceso), cuáles son los más demandados por los clientes. Estos serán los más interesantes y los candidatos a ser incluidos en el proyecto. No obstante, se podrán incluir en un futuro otros menos relevantes o de los que surja necesidad.

A continuación, se muestra una tabla con el porcentaje de aparición en los diferentes proyectos. Consideraremos módulo candidato de ser incluido en el sistema modular de este proyecto a aquellos que aparezcan en al menos el 50% del total de proyectos.

**Tabla 1:** Porcentaje de aparición de módulos en los proyectos WordPress de la empresa Brooktec.

| Nombre módulo | Breve descripción | Apariciones / Totales | Porcentaje | ¿Candidato a sistema modular? |
|---|---|---|---|---|
| Texto | Inserta un párrafo de texto simple | 9/10 | 90% | ✔ |
| Imagen | Inserta una imagen | 4/10 | 40% | ✘ |
| Texto + Imagen | Inserta un texto con una imagen al lado | 6/10 | 60% | ✔ |
| Banner | Inserta un banner con imagen de fondo, texto y posibilidad de CTA | 6/10 | 60% | ✔ |
| Cifras destacadas | Inserta varias columnas con cifras de datos destacados | 2/10 | 20% | ✘ |
| Timeline | Inserta una línea de tiempo con hitos en diferentes fechas | 1/10 | 10% | ✘ |
| Hero | Inserta una cabecera de página con una imagen y texto | 5/10 | 50% | ✔ |
| Enlaces destacados | Inserta enlaces a contenido destacado | 1/10 | 10% | ✘ |
| Columnas con iconos | Inserta columnas con un icono seguido verticalmente de un pequeño título y/o texto | 3/10 | 30% | ✘ |
| Distribuidor de contenido | Inserta tarjetas con la imagen destacada, título y entradilla de un | 6/10 | 60% | ✔ |

| | | | | |
|---|---|---|---|---|
| | post (puede llevar autor y fecha) | | | |
| Video | Inserta un video | 6/10 | 60% | ✔ |
| CTA/Botón | Inserta un botón con enlace | 4/10 | 40% | ✘ |
| Contenido destacado | Inserta tarjetas con la imagen destacada, título y entradilla de un post (puede llevar autor y fecha) | 7/10 | 70% | ✔ |
| FAQs (Frequently Asked Questions) | Inserta un bloque de preguntas y respuestas | 5/10 | 50% | ✔ |
| Formulario | Inserta un formulario previamente configurado a través de contact form 7 | 3/10 | 30% | ✘ |
| Script | Permite insertar un script de código javascript | 1/10 | 10% | ✘ |
| Textos numerados | Inserta una serie de bloques de texto con una numeración lateral izquierda | 1/10 | 10 | ✘ |
| Ubicación | Inserta un mapa de google maps con una ubicación | 2/10 | 20% | ✘ |
| Contenido relacionado | Inserta contenido relacionado del post actual (relacionado por categoría) | 5/10 | 50% | ✔ |
| Equipo de personas | Inserta tarjetas de información de empleados | 1/10 | 10% | ✘ |
| Carrusel | Inserta un carrusel | 3/10 | 30% | ✘ |

| | | | | |
|---|---|---|---|---|
| | de tarjetas o imágenes | | | |
| Separador | Inserta un separador que permite dar márgenes personalizados entre módulos | 4/10 | 40% | ✘ |
| Tarjetas de información | Inserta tarjetas de información con imagen y texto | 2/10 | 20% | ✘ |
| Galería | Inserta un bloque galería de imágenes en modo diapositivas | 5/10 | 50% | ✔ |
| Pestañas | Inserta un bloque de pestañas con información | 3/10 | 30% | ✘ |
| Título + texto | Inserta un título y un texto | 4/10 | 40% | ✘ |
| Texto + CTA | Inserta un texto más un botón con enlace | 3/10 | 30% | ✘ |
| Podcast | Inserta un reproductor embebido de podcast | 2/10 | 20% | ✘ |
| Cita | Inserta una cita destacada de un autor | 2/10 | 20% | ✘ |
| Compartir en redes | Inserta un módulo con iconos de redes para compartir | 2/10 | 20% | ✘ |
| Modal | Abre una modal con información | 1/10 | 10% | ✘ |
| Testimonios | Inserta una serie de testimonios con una texto, imagen y autor | 3/10 | 30% | ✘ |

En la tabla 2 se incluye información más detallada de los proyectos que usan cada uno de dichos módulos.

**Tabla 2:** Aparición de módulos en los proyectos WordPress de la empresa Brooktec.

| | Texto | Imagen | Texto + Imagen | Banner | Cifras destacadas | Hero | Columnas | Enlaces destacados | Timeline | Distribuidor de contenido | Video | CTA/Botón | Contenido Destacado | FAQs | Formulario | Script |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ACSG | x | | | x | | x | | | | x | | | x | x | x | |
| Club Matador | | | x | | | | | | | | | | | | | |
| EBN Multisite | x | x | | x | | | x | | | x | x | x | x | x | x | x |
| Forenex | x | | x | x | | x | | | | | | | | x | x | |
| FundsPeople Multisite | x | | | x | | x | | | | x | x | x | x | | | |
| KPMG - Alumni | x | x | x | | | | | | | x | x | x | x | | | |
| KPMG - Tendencias | x | x | x | x | x | x | | x | x | x | x | x | x | x | | |
| UE Alumni | x | | x | | x | | | | | | | | x | | | |
| UE El Economista | x | | | x | | | x | | | x | x | | x | x | | |
| Terralpa Multisite | x | x | x | | | x | x | | | | x | | | | | |
| **Total** | 9 | 4 | 6 | 6 | 2 | 5 | 3 | 1 | 1 | 6 | 6 | 4 | 7 | 5 | 3 | 1 |

| Textos numerados | Ubicación | Contenido relacionado | Equipo | Carrousel | Separador | Tarjetas de información | Galería | Pestañas | Título + Texto | Texto + CTA | Podcast | Cita | Compartir en redes | Modal | Testimonios |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | x | | | | | | | | | |
| | | x | x | x | x | | | | | | | | | | |
| x | x | | | | | | | | | | | | | | |
| | | | | | | | x | x | x | | | | | | x |
| | | | | | x | | | x | | | x | | | | |
| | | x | | x | x | | x | | x | x | | x | x | | |
| | | x | | | x | | x | | | x | x | x | x | | |
| | | x | | x | | x | x | x | x | x | | | | x | x |
| | | x | | | | | | | | | | | | | x |
| | x | | | | | | x | | x | | | | | | |
| 1 | 2 | 5 | 1 | 3 | 4 | 2 | 5 | 3 | 4 | 3 | 2 | 2 | 2 | 1 | 3 |

Por lo tanto, según esta tabla, los módulos candidatos a aparecer en nuestro *framework* serían los siguientes:

- Texto
- Texto + imagen
- Banner
- Hero
- Distribuidor de contenido
- Video
- Contenido destacado
- FAQs (Frequently Asked Questions)
- Contenido relacionado
- Galería

Sin embargo, vamos a intentar mejorar esta propuesta, ya que en esta lista hay varios módulos que son muy parecidos y podrían unificarse. También algunos de los módulos que no han salido candidatos se parecen a otros que sí han salido e igualmente podrían unificarse y enriquecer las opciones de cada módulo. Además, hay módulos que, aunque no sean solicitados expresamente por los clientes, son altamente útiles, como el módulo 'separador' o el módulo 'script'.

Los módulos 'texto', 'CTA/botón', 'título + texto', 'texto + CTA' pueden ser unificados en un solo módulo en el que se le dé la opción al usuario de introducir los elementos que desee sin que sea ninguno obligatorio. De esta forma, tenemos cuatro módulos unificados en uno. Por otro lado, los módulos 'distribuidor de contenido', 'contenido destacado' y 'contenido relacionado' son muy parecidos entre ellos, por lo que podremos unificarlos en uno solo que funcione de la

siguiente manera: habrá un selector que permita configurar si el contenido a mostrar se añade manualmente o de forma automática. Si se selecciona el modo automático, se preguntará al usuario si se desea mostrar contenido relacionado con el *post* o un listado de *posts* más recientes publicados. Con esas opciones podemos cubrir las tres modalidades.

También son módulos muy sencillos e interesantes de incluir el de 'script' y el de 'separador'. El módulo 'script' nos permitirá incluir *scripts* personalizados que necesite añadir el cliente a mitad de página, como, por ejemplo, un *script* de Trustpilot para mostrar opiniones registradas en la cuenta de Trustpilot de la empresa o cualquier otro tipo de script de estas características. El módulo separador nos permitirá añadir unos márgenes superiores e inferiores personalizados entre módulos.

Por último, se ha de remarcar que el módulo 'galería' no necesita ser incluido en la lista, ya que esta galería cambia radicalmente de aspecto de una web a otra, haciendo imposible unificar su código HTML para ser solo adaptado por CSS, por lo que se excluye.

Por tanto, el listado final de módulos sería el siguiente:

- Título + texto + CTA
- Texto + imagen
- Banner
- Hero
- Distribuidor de contenido (destacado/relacionado)
- Video
- FAQs (Frequently Asked Questions)
- Separador
- Script

Estos son los módulos prioritarios y los que primero se implementarán, aunque, como se ha comentado, dicho listado puede ser aumentando en un futuro.

## 3.4. Planificación

El proyecto debe tener una duración de 300 horas para cumplir con los requisitos de tiempo del Trabajo Fin de Grado. Se dividirá en fases y las fases tendrán diferentes tareas. Se seguirá una metodología Scrum en la que el autor del TFG hará de todas las figuras a la vez; Product Owner, Scrum Master y Development Team. Los *sprints* tendrán una duración de tres semanas cada uno y cada *sprint* tendrá una duración de 60 horas (20 horas semanales). Las diferentes fases serán las siguientes:

1. Definición de objetivos
2. Análisis tecnológico
3. Estudio de tendencias en clientes
4. Formación WordPress y Advanced Custom Fields
5. Implementación
6. Test y pruebas

7. Documentación

Dividiremos las diferentes fases en cada uno de los sprints. Para cumplir con el presupuesto de horas del proyecto debe haber 15 semanas de trabajo. Si cada sprint tiene una duración de 3 semanas, habrá un total de 5 sprints.

Sprint 1: Fase de definición, análisis y estudio de tendencias.
Sprint 2: Formación WordPress y Advanced Custom Fields
Sprint 3: Implementación
Sprint 4: Implementación
Sprint 5 : Test, pruebas y documentación.

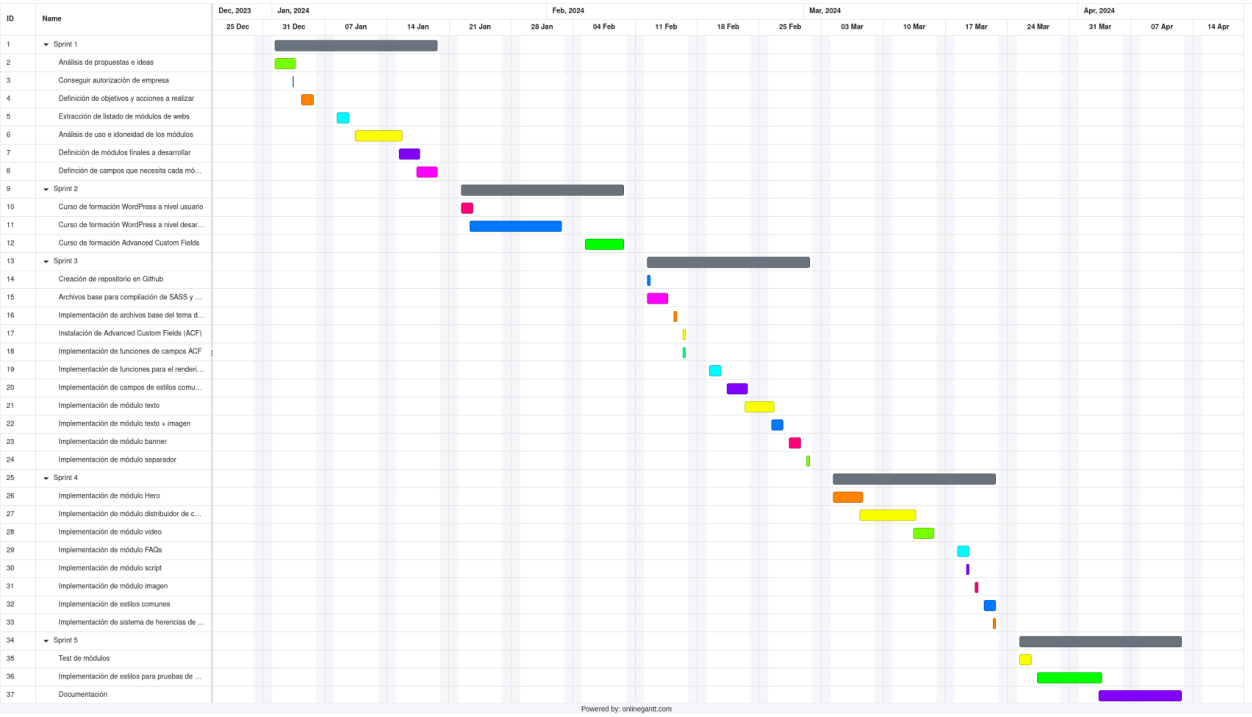



**Figura 3:** Diagrama de Gantt con toda la planificación del proyecto.

### 3.4.1. Tareas por sprints

**<u>Sprint 1:</u> 01/01/2024 - 19/01/2024**

En el primer sprint se pretende hacer una valoración de ideas para el proyecto para ver cuáles pueden resultar interesantes. También se pretende conseguir la autorización de la empresa para llevar a cabo el trabajo. Una vez conseguida dicha autorización, se procederá a hacer un estudio de las diferentes webs que la empresa ha desarrollado o de las que lleva su mantenimiento para extraer un listado de módulos, ver cuáles tienen sentido de ser homogeneizados y desarrollados,

y pensar qué campos debe tener cada módulo para poder acomodarse en la mayor cantidad de webs que se desarrollarán en el futuro con dicho *framework*.

**Tabla 3:** Planificación de tareas para el sprint 1.

| **Sprint 1** | | | |
|---|---|---|---|
| **User Story** | **Tiempo estimado en horas** | **Tiempo realizado en horas** | **Diferencia en horas** |
| Análisis de propuestas e ideas | 10 | 16 | 6 |
| Conseguir autorización de empresa | 2 | 2 | 0 |
| Definición de objetivos y acciones a realizar | 8 | 8 | 0 |
| Extracción de listado de módulos de webs | 8 | 13 | 5 |
| Análisis de uso e idoneidad de los módulos | 10 | 10 | 0 |
| Definición de módulos finales a desarrollar | 11 | 12 | 1 |
| Definición de campos que necesita cada módulo | 11 | 16 | 5 |
| **Total** | 60 | 77 | 17 |

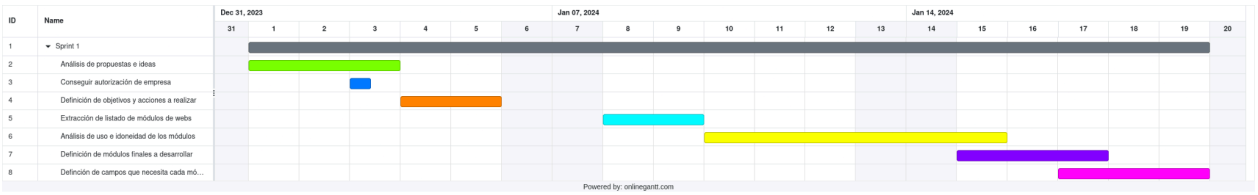



**Figura 4:** Diagrama de Gantt con la planificación del sprint 1.

**<u>Sprint 2:</u> 22/01/2024 - 11/02/2024**

En este sprint se adquirirán los conocimientos necesarios de desarrollo y uso de WordPress, así como de las particularidades y funciones de Advanced Custom Fields. Esto será realizado mediante cursos de formación además de la propia experiencia adquirida trabajando para Brooktec en tareas del equipo de desarrollo de WordPress.

**Tabla 4:** Planificación de tareas para el sprint 2.

| **Sprint 2** | | | |
|---|---|---|---|
| **User Story** | **Tiempo estimado en horas** | **Tiempo realizado en horas** | **Diferencia en horas** |
| Curso de formación WordPress a nivel usuario | 5 | 8 | 3 |
| Curso de formación WordPress a nivel desarrollador | 35 | 50 | 15 |
| Curso de formación Advanced Custom Fields | 20 | 30 | 10 |
| **Total** | 60 | 88 | 28 |


**Figura 5:** Diagrama de Gantt con la planificación del sprint 2.

**<u>Sprint 3:</u> 12/02/2024 - 01/03/2024**

En dicho sprint comenzará la implementación del sistema, empezando por crear un nuevo repositorio del que luego bifurcará el resto de los proyectos futuros que usen el sistema modular para la construcción de nuevos sites. Se empezará por crear una serie de archivos base para que se pueda desarrollar utilizando SASS en lugar de CSS, por lo que se creará un archivo *gulp* y una

configuración para que la compilación con Node traduzca el código SASS a CSS minificado y el Javascript (JQuery) a Javascript minificado.

Se crearán los archivos base del tema de WordPress y se instalará Advanced Custom Fields para poder comenzar el desarrollo. Se implementará la funcionalidad básica de campos ACF para una mejor y más cómoda definición y lectura de campos en el desarrollo de los módulos, así como para proporcionar las funcionalidades de renderizado flexible cuando los módulos sean reordenados desde el panel de administración de WordPress.

Una vez realizadas estas tareas, se comenzará con el desarrollo de los módulos uno a uno que constará de una definición de la estructura de campos y de renderizado de dichos campos en el *front*. En este sprint se empezará por los más sencillos.

**Tabla 5:** Planificación de tareas para el sprint 3.

| **Sprint 3** | | | |
|---|---|---|---|
| **User Story** | **Tiempo estimado en horas** | **Tiempo realizado en horas** | **Diferencia en horas** |
| Creación de repositorio en github | 2 | 2 | 0 |
| Archivos base para la compilación de SASS y minificación de JS y CSS mediante Node | 10 | 12 | 2 |
| Implementación de archivos base del tema de WordPress | 4 | 4 | 0 |
| Instalación de Advanced Custom Fields (ACF) | 1 | 1,5 | 0,5 |
| Implementación de funciones de campos ACF | 3 | 3 | 0 |
| Implementación de funciones para el renderizado flexible de módulos | 8 | 8 | 0 |
| Implementación de campos de estilos comunes a todos los | 10 | 10 | 0 |

| | | | |
|---|---|---|---|
| módulos | | | |
| Implementación de módulo Texto | 5 | 5 | 0 |
| Implementación de módulo texto + imagen | 8 | 8 | 0 |
| Implementación de módulo banner | 6 | 6 | 0 |
| Implementación de módulo separador | 3 | 3 | 0 |
| **Total** | 60 | 62,5 | 2,5 |

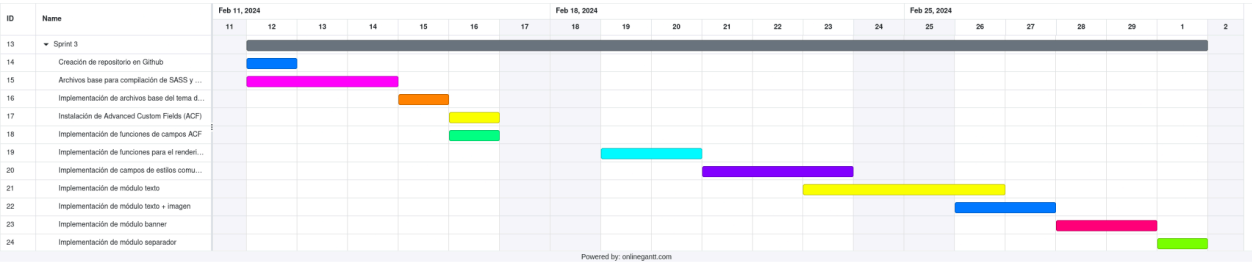



**Figura 6:** Diagrama de Gantt con la planificación del sprint 3.

## <u>Sprint 4:</u> 04/03/2024 - 24/03/2024

En este sprint se terminarán los módulos que faltan de los seleccionados para su implementación y se implementará el sistema de archivos de estilos básicos de los módulos para que los desarrolladores puedan heredar dichos estilos y extenderlos.

**Tabla 6:** Planificación de tareas para el sprint 4.

| **Sprint 4** | | | |
|---|---|---|---|
| **User Story** | **Tiempo estimado en horas** | **Tiempo realizado en horas** | **Diferencia en horas** |
| Implementación de módulo Hero | 15 | 12 | -3 |
| Implementación de | 15 | 14 | -1 |

| | | | |
|---|---|---|---|
| módulo distribuidor de contenido | | | |
| Implementación de módulo video | 10 | 8 | -2 |
| Implementación de módulo FAQs | 6 | 5 | -1 |
| Implementación de módulo script | 3 | 3 | 0 |
| Implementación de módulo imagen | 3 | 3 | 0 |
| Implementación de estilos comunes | 5 | 7 | 2 |
| Implementación de sistema de herencias de estilos | 3 | 2 | -1 |
| **Total** | 60 | 54 | -6 |

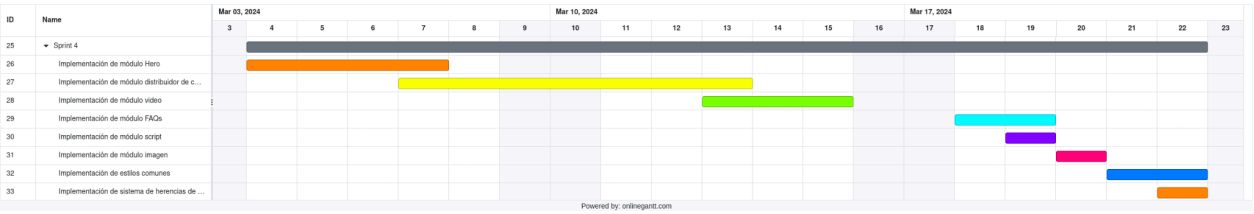



**Figura 7:** Diagrama de Gantt con la planificación del sprint 4.

## <u>Sprint 5:</u> 25/03/2024 - 14/04/2024

En este sprint se comprobará el funcionamiento de dichos módulos introduciendo datos reales de diferentes webs y simulando que un desarrollador a posteriori tuviera que implementar los estilos de cada módulo. En este sprint se escribirá la memoria de este Trabajo Fin de Grado.

**Tabla 7:** Planificación de tareas para el sprint 5.

| **Sprint 5** | | | |
|---|---|---|---|
| **User Story** | **Tiempo estimado en horas** | **Tiempo realizado en horas** | **Diferencia en horas** |
| Test de módulos | 8 | 8 | 0 |

| | | | |
|---|---|---|---|
| Implementación de estilos para pruebas de módulos | 22 | 16 | -6 |
| Documentación | 30 | 36 | 6 |
| **Total** | 60 | 60 | 0 |


**Figura 8:** Diagrama de Gantt con la planificación del sprint 5.

## 3.5. Presupuesto y beneficios

El autor aprovecha en este apartado su actual puesto de *Project Manager* y su conocimiento del coste por hora que se cobra a cliente para hacer un presupuesto acorde a lo que Brooktec cobraría a un cliente real y los beneficios que se obtendrían. Para ello, se desmenuzan los costes directos e indirectos y se hace una comparativa con el precio por hora estándar que se le cobra a un cliente.

### 3.5.1. Costes directos

En la siguiente tabla se muestran los costes directos asociados a los salarios de los trabajadores que llevarían a cabo el proyecto. En este caso, el propio autor realiza las funciones de desarrollador y de *Project Manager*, por lo que se mostrarán los costes de ambos perfiles:

**Tabla 8:** Costes de trabajadores.

| Nombre y apellidos | Experiencia (1) | Categoría profesional | Grupo cotización | Coste euros/hora (2) | Horas dedicadas | Presupuesto total |
|---|---|---|---|---|---|---|
| Juan Mayor | 3 | Técnico junior | 05 | 10,65 | 300 | 3196,35€ |
| Juan Mayor | 1 | Jefe de proyecto junior | 01 | 14,91 | 25 | 372,9 |
| | | | | | **Total** | **3569,25** |

(1) **Experiencia:** años de experiencia en el puesto.
(2) **Coste euros/hora:** Retribución bruta anual más Seguridad Social a cargo de la empresa, dividido entre el número de horas anuales

### 3.5.2. Costes indirectos

A los costes de los trabajadores habrá que añadir todos aquellos gastos añadidos que la empresa debe hacer frente que corresponden a costes indirectos como, por ejemplo, licencias de software, alquiler de oficinas, gasto energético, pago de servidores o cuentas asociadas a la empresa, etc. Estos gastos son difíciles de desmenuzar, por lo que se hace una estimación aproximada que se establece en un 10% del total facturado al cliente, por lo que son descontados a la hora de calcular los beneficios.

### 3.5.3. Facturación

El precio estándar por hora de desarrollo en Brooktec para cualquier proyecto está establecido en 60 euros/hora, IVA incluido. Por tanto, con un simple cálculo sería:

60 x 325 = 19500 euros

### 3.5.4. Beneficios

Para ver los beneficios obtenidos se ha de tener muy presente que el presupuesto no es negociable y que se pactaron inicialmente con el cliente 19500 euros que corresponden a los 60 euros la hora, con una estimación de 325 horas de desarrollo además de la gestión. Considerando todo esto, habrá que descontar todos los gastos directos, indirectos, el IVA y las desviaciones en las estimaciones de tiempos. Seguiremos este criterio por simplificación y porque la desviación en estimación de tiempos no es demasiado grande como para afectar a los beneficios en gran medida.

**Tabla 9:** Desglose de gastos y beneficio obtenido.

| **Facturado** | **19500** |
|---|---|
| Costes directos | 3569,25 |
| Costes indirectos (1) | 1950 |
| IVA 21% | 4095 |
| Desviación en horas de desarrollo (2) | 442 |
| **Total** | **9443,75** |

(1) **Gastos indirectos:** 10% del total de la facturación.
(2) **Desviación en horas de desarrollo:** Se calcula multiplicando las horas que se han desarrollado de más (41,5 horas) de las estimadas en un principio por el coste de desarrollo de cada hora (10,65).

Por tanto, el margen operativo es del 48,43%, lo cual es una cifra muy buena. Además, hay que tener en cuenta que este proyecto no es un producto final, sino una herramienta para rentabilizar aún más proyectos futuros, por lo que habría que añadir todas las horas de desarrollo ahorradas a futuro en otros proyectos que partan con este tema modular como base.

Las estimaciones del autor son de un posible ahorro del 30% del trabajo necesario para sacar adelante un *site* modular en WordPress, por lo que, cuanta mayor sea su aplicación en otros proyectos, mayor será el porcentaje de rentabilidad y beneficio obtenido en el tiempo.

# Capítulo 4

# 4. Metodología empleada

A la hora de desarrollar los diferentes módulos y secciones de código, se va a seguir en todo momento una metodología de flujos *git* con sus ramas principales, ramas de soporte y flujo de trabajo. Sin embargo, no se va a usar el universalmente conocido GitFlow, sino uno muy particular que se utiliza en nuestra empresa, llamado BrooktecFlow, y que permitirá, una vez puesto un proyecto en producción, poder sacar modificaciones rápidamente a producción sin que interfieran unas tareas con otras.

## 4.1. GitFlow vs BrooktecFlow

GitFlow es un modelo de flujo de trabajo para gestionar proyectos con Git, diseñado para ofrecer un marco robusto para manejar proyectos grandes. Es especialmente útil cuando se trata de proyectos en los que las publicaciones se programan regularmente [12]. Los componentes clave:

**Ramas principales Gitflow**

- **master:** Es la rama principal donde el código del producto final reside. En GitFlow, esta es considerada la rama de producción.
- **develop:** Esta rama sirve como un área de integración para características. Es donde todos los desarrollos en curso se combinan, y es de donde se ramifican todas las ramas de características.

**Ramas de soporte Gitflow**

- **feature:** ramas que se utilizan para desarrollar nuevas funcionalidades. Son creadas a partir de la rama ‘develop’ y deben fusionarse de vuelta en ‘develop’ una vez que la característica está completa. Usualmente, se nombran comenzando con "feature/" seguido por un nombre descriptivo.
- **release:** estas ramas ayudan a preparar lanzamientos de versiones nuevas. Se bifurcan desde ‘develop’ y se fusionan en ‘master’ y ‘develop’ cuando el lanzamiento está listo. Permiten ajustes finales y correcciones de errores. Generalmente, comienzan con "release/" seguido por el número de versión.
- **hotfix:** son ramas destinadas a producir parches rápidos en la rama ‘master’. Si surge un problema en producción, se crea una rama ‘hotfix’, se realiza la corrección, y luego se fusiona tanto en master como en ‘develop’. Estas ramas suelen comenzar con "hotfix/" seguido por el número de versión.

**Flujo de trabajo Gitflow**

- **Inicio de una funcionalidad:** se crea una rama ‘feature’ desde ‘develop’. Esta rama es el lugar de trabajo para la nueva funcionalidad.
- **Finalización de la funcionalidad:** una vez completada y probada, la funcionalidad se fusiona de nuevo en ‘develop’.
- **Preparación del lanzamiento:** cuando ‘develop’ tiene suficientes funcionalidades para una versión, se crea una rama ‘release’. No se añaden nuevas funcionalidades a esta rama; solo se hacen correcciones de errores y preparativos para el lanzamiento.
- **Lanzamiento:** una vez que la rama ‘release’ está lista, se fusiona en ‘master’ y en ‘develop’. ‘Master’ debería tener una etiqueta con el número de versión.
- **Corrección de errores en producción:** si se detecta un error en ‘master’, se crea una rama ‘hotfix’, que después de arreglar el problema, se fusiona de vuelta tanto en ‘master’ como en ‘develop’.

Este modelo está estructurado para asegurar que las funcionalidades se añadan de manera controlada, y que las versiones sean estables, haciendo las correcciones necesarias de forma eficiente. Este flujo ayuda también a mantener un historial claro y manejable de todas las modificaciones y mejoras en el proyecto. Sin embargo, este flujo de trabajo no encaja bien en la empresa una vez el producto ha salido a producción. Esto es porque los clientes que tienen su web en mantenimiento suelen pedir varias funcionalidades no relacionadas las unas con las otras. A algunas de ellas se les da la validación para su salida a producción en seguida y otras pueden pasar meses en el entorno de pruebas sin que sean validadas. Si siguiéramos Gitflow, no podríamos publicar las funcionalidades que tienen el visto bueno hasta que no validasen todas las restantes que se quedaron por validar, o nos veríamos obligados a hacer constantes *cherry picks* de código. Por ello, se sigue un flujo de trabajo llamado BrooktecFlow. Los principales componentes y diferencias con GitFlow son los siguientes:

**Ramas principales BrooktecFlow**

- **main:** es la rama principal donde el código del producto final reside y funciona igual que en Gitflow. La única pero fundamental diferencia es que todas las ramas de nuevas funcionalidades partirán de aquí.
- **develop:** esta rama sirve como banco de todas las funcionalidades publicadas en ‘main’ junto con las funcionalidades que todavía no se encuentran en ‘main’ y de las que el cliente está pendiente de validar en el entorno de preproducción.

**Ramas de soporte BrooktecFlow**

- **feature:** ramas que se utilizan para desarrollar nuevas funcionalidades. Son creadas a partir de la rama ‘main’ y deben fusionarse primero en ‘develop’ y una vez que ha pasado el visto bueno del cliente, se fusiona en ‘main’ de nuevo. Usualmente, se nombran comenzando con "feature/" seguido por un nombre descriptivo.
- **hotfix:** son ramas destinadas a producir parches rápidos en la rama master. Si surge un problema en producción, se crea una rama hotfix, se realiza la corrección, y luego se fusiona tanto en master como en develop. Estas ramas suelen comenzar con "hotfix/" seguido de un nombre descriptivo.

**Flujo de trabajo BrooktecFlow**

- **Inicio de una funcionalidad:** se crea una rama ‘feature’ desde ‘main’. Esta rama es el lugar de trabajo para la nueva funcionalidad.
- **Finalización de la funcionalidad:** una vez completada se fusiona en ‘develop’ y se sube al entorno de preproducción para que el cliente dé el visto bueno.
- **Lanzamiento:** una vez el cliente ha dado el visto bueno en preproducción, la rama se fusiona también en ‘master’.
- **Corrección de errores en producción:** si se detecta un error en ‘master,’ se crea una rama ‘hotfix’, que después de arreglar el problema, se fusiona de vuelta tanto en ‘develop’ como en ‘master’.

Para el desarrollo de este proyecto se ha de tener en cuenta que al no estar en producción se ha utilizado la rama main como main y develop al mismo tiempo. Sin embargo, una vez finalizado y puesto a disposición de la empresa, el flujo futuro de trabajo será el anteriormente descrito en la sección de BrooktecFlow para su ampliación.

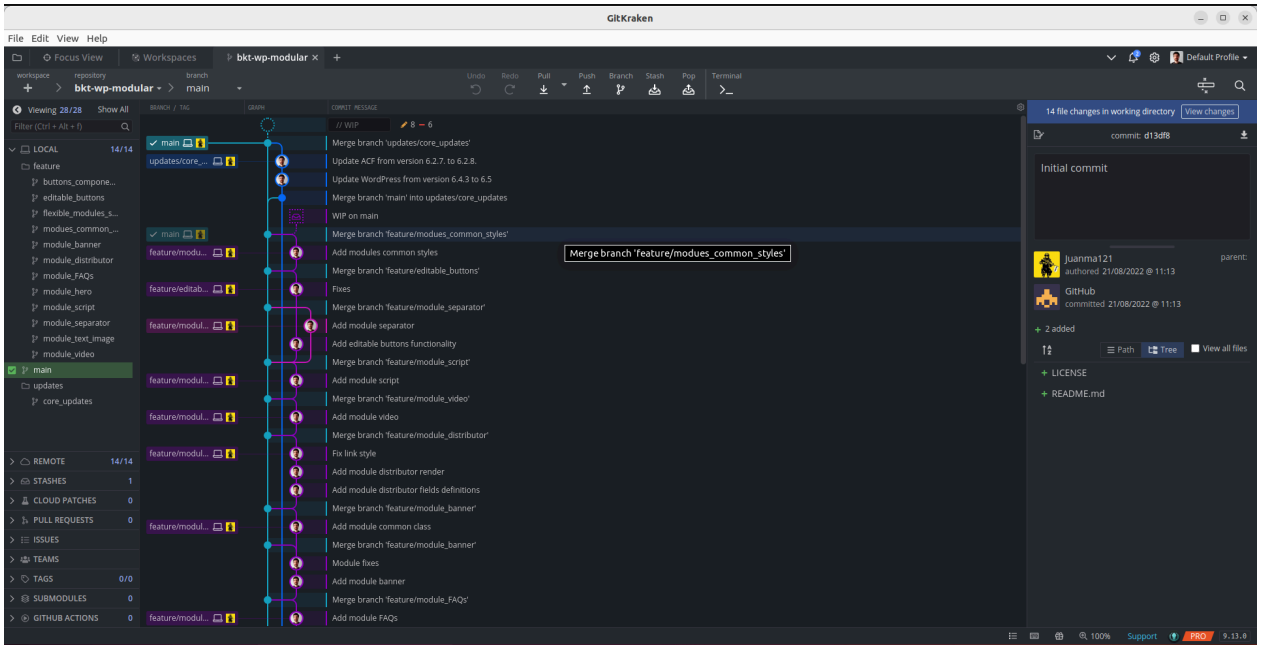



**Figura 9:** Uso de Gitkraken como herramienta visual de git para seguir el flujo de trabajo.

## 4.2. Metodologías ágiles

Las metodologías ágiles en el desarrollo de software son enfoques de gestión de proyectos que enfatizan la flexibilidad, la colaboración continua, la entrega rápida y la mejora iterativa. Surgieron como una respuesta a las limitaciones de los métodos tradicionales de desarrollo de software, como el modelo en cascada, que son más rígidos y lineales. Algunas de las características principales y metodologías ágiles más populares son las siguientes:

1. **Iterativo e Incremental**: las metodologías ágiles se basan en ciclos de desarrollo cortos y repetitivos llamados iteraciones o *sprints*, que típicamente duran entre una y cuatro semanas. Cada iteración es un miniproyecto en sí mismo y produce una versión de trabajo del producto que se puede demostrar a los *stakeholders*.
2. **Colaboración y Comunicación**: la colaboración continua entre todos los miembros del equipo y los *stakeholders* es crucial. Las reuniones diarias, las revisiones de *sprint* y las retrospectivas son comunes para asegurarse de que todos los miembros del equipo están alineados y pueden discutir desafíos y progresos.
3. **Respuesta a Cambios**: en lugar de seguir un plan fijo, las metodologías ágiles están diseñadas para ser muy adaptables a los cambios en los requisitos del cliente o del proyecto. Esto permite que el proyecto evolucione según las necesidades del mercado o del usuario final.
4. **Enfoque en el Usuario**: el desarrollo ágil pone un fuerte énfasis en la satisfacción del cliente y en la entrega de valor. Esto a menudo se logra a través de la implementación de funciones basadas en su prioridad para el negocio y el usuario final.

Algunas de las metodologías ágiles más populares son:

1. **Scrum:** es una de las metodologías ágiles más utilizadas. Divide el desarrollo en ciclos llamados *sprints* y utiliza roles específicos (Scrum Master, Product Owner, Team Members), artefactos (*backlog* del producto, *backlog* del *sprint*) y ceremonias (*sprint planning*, *daily standup*, *sprint review*, *sprint retrospective*) para gestionar el proceso. Esta metodología en concreto es la que se ha seguido como se ha explicado en el apartado 3 de análisis y planificación [13].
2. **Kanban:** se centra en la visualización del trabajo y la gestión del flujo de trabajo. Kanban utiliza un tablero Kanban para visualizar todas las tareas en varias columnas que representan diferentes etapas del proceso de desarrollo. Las reglas principales incluyen limitar el trabajo en progreso y optimizar el flujo de trabajo [14].
3. **Extreme Programming (XP):** pone un fuerte énfasis en la calidad técnica y la capacidad de adaptarse a los cambios de requisitos del cliente. XP utiliza técnicas como la programación en pareja, desarrollo basado en pruebas, integración continua y diseño simple para mejorar la calidad del *software* y la productividad del equipo [15].
4. **Feature-Driven Development (FDD):** este enfoque está orientado a características y se centra en la entrega de "características" tangibles y funcionales, organizadas y planificadas sistemáticamente [16].

# Capítulo 5

# 5. Especificaciones

## 5.1. Diagrama de casos de uso

En este apartado se muestra, a través de la Figura 10, los principales casos de uso teniendo en cuenta el usuario desarrollado, ya que es el más relevante e interesante. Se podría añadir un segundo usuario administrador, que sería una vez desplegado en producción la bifurcación del código de una web desarrollado con este sistema el que se encargaría de utilizar los módulos para construir visualmente los contenidos de la web, pero no es el objeto de estudio de este proyecto.

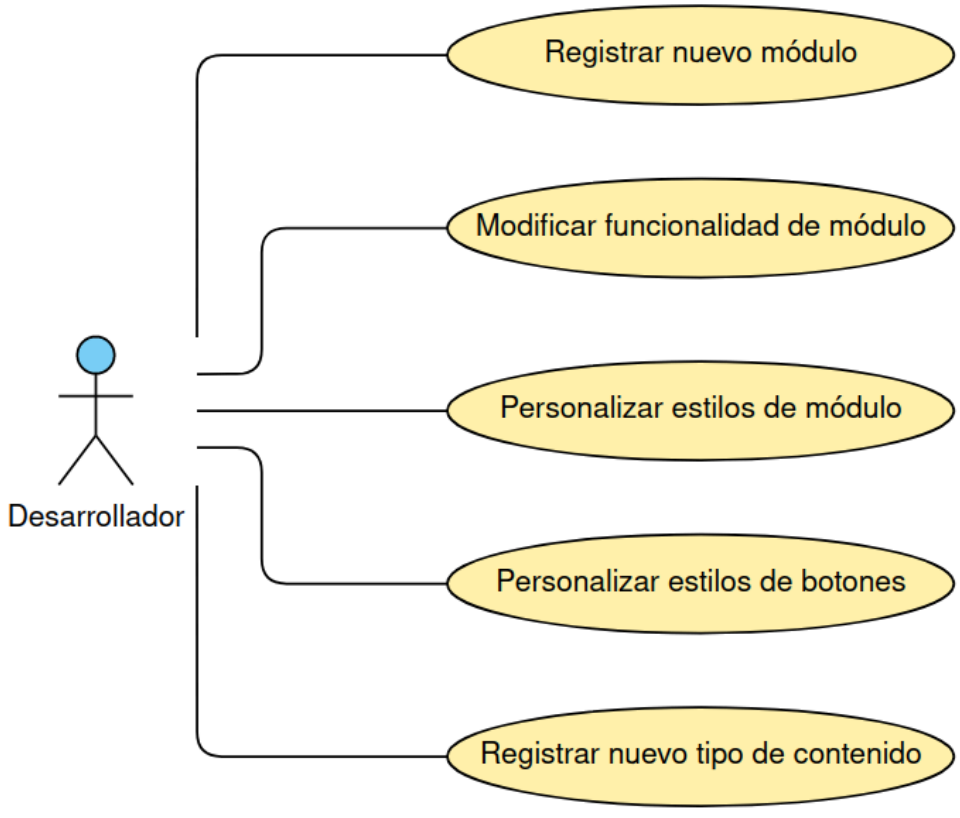



**Figura 10:** Diagrama de casos de uso bkt-wp-modular.

## 5.2. Descripción de casos de uso

En este apartado se explican los casos de uso más importantes del sistema. Se detallan cuáles son las interacciones entre el usuario y el sistema y el comportamiento esperado en cada caso.

**Tabla 10:** Caso de Uso registrar nuevo módulo.

| **CU.01** | |
|---|---|
| **Título** | Registrar nuevo módulo |
| **Descripción** | El usuario tendrá la posibilidad de añadir un nuevo módulo al sistema. |
| **Precondición** | |
| **Postcondición** | El módulo aparecerá disponible en la página de configuración de módulos para ser habilitado en los diferentes contenidos. |
| **Prioridad** | Alta |
| **Escenario Principal** | |
| 1. El usuario crea una carpeta dentro de la ruta correspondiente con el nombre del módulo.<br>2. El usuario crea un archivo definition.php en el que se incluye la definición de campos ACF del módulo dentro de la carpeta.<br>3. El usuario crea un archivo llamado functions.php que debe contener al menos una función de renderizado dentro de la carpeta.<br>4. El usuario crea sus estilos base en la carpeta de estilos correspondiente | |
| **Escenario Alternativo** | |
| | |

**Tabla 11:** Caso de Uso modificar funcionalidad de un módulo.

| **CU.02** | |
|---|---|
| **Título** | Modificar funcionalidad de un módulo |
| **Descripción** | El usuario tendrá la posibilidad de cambiar el comportamiento de un módulo. |
| **Precondición** | El módulo debe estar registrado en la carpeta correspondiente. |
| **Postcondición** | El módulo se comportará de diferente manera que la establecida por defecto. |
| **Prioridad** | Alta |
| **Escenario Principal** | |
| 1. El usuario modifica la definición de campos ACF en el archivo<br>2. El usuario modifica el renderizado del módulo en el archivo | |
| **Escenario Alternativo** | |
| | |

**Tabla 12:** Caso de Uso personalizar estilos de un módulo.

| **CU.03** | |
|---|---|
| **Título** | Personalizar estilos de módulo |
| **Descripción** | El usuario tendrá la posibilidad de añadir estilos al módulo. |
| **Precondición** | El módulo debe estar registrado en la carpeta correspondiente. |
| **Postcondición** | El módulo se renderizará con los estilos aplicados. |
| **Prioridad** | Alta |
| **Escenario Principal** | |
| 1. El usuario añade reglas de estilo al archivo correspondiente.<br>2. El usuario compila el tema con el comando npx gulp build. | |
| **Escenario Alternativo** | |
| | |

**Tabla 13:** Caso de Uso personalizar estilos de botones.

| **CU.04** | |
|---|---|
| **Título** | Personalizar estilos de botones |
| **Descripción** | El usuario tendrá la posibilidad de personalizar los estilos de los botones definidos. |
| **Precondición** | Deben existir las variables de colores que se deseen aplicar. |
| **Postcondición** | Los botones pasarán a tener los estilos de la customización. |
| **Prioridad** | Media |
| **Escenario Principal** | |
| 1. El usuario define las reglas de estilos de las diferentes clases de botones en el archivo correspondiente en el que se heredan sus estilos básicos. | |
| **Escenario Alternativo** | |
| | |

**Tabla 14:** Caso de Uso registrar nuevo tipo de contenido.

| **CU.05** | |
|---|---|
| **Título** | Registrar nuevo tipo de contenido |
| **Descripción** | El usuario tendrá la posibilidad de añadir nuevos tipos de contenido personalizado. |
| **Precondición** | |
| **Postcondición** | En la página de configuración de módulos aparecerá una nueva sección donde configurar el listado de módulos habilitados o deshabilitados para ese nuevo tipo de contenido. |
| **Prioridad** | Alta |
| **Escenario Principal** | |
| 1. El usuario crea un archivo en la carpeta correspondiente a los CPTs (custom post type) con el nombre del CPT y su extensión.<br>2. El usuario utiliza la función de registro de nuevo cpt en dicho archivo. | |
| **Escenario Alternativo** | |
| | |

# Capítulo 6

# 6. Diseño e implementación

## 6.1. Diseño

El primer paso fundamental para poder diseñar el sistema es analizar, dentro de los módulos requeridos descritos en el apartado 3.3 (Análisis de módulos requeridos), qué campos serán necesarios para cada uno de los módulos. Los campos del módulo video, por ejemplo, serán muy distintos a los campos del módulo distribuidor. Se hace un desglose de campos por cada uno de los módulos.

### 6.1.1. Estructura de archivos

Es fundamental implementar una estructura de archivos intuitiva y fácil de entender para que el usuario sepa en todo momento qué archivos debe y cuáles no debe modificar, y dónde se encuentra la información de cada funcionalidad, plantilla, etc. Dentro del tema de WordPress al cual hemos llamado bkt-modular, que contendrá todos los archivos del tema, distinguimos entre las siguientes carpetas:

- **dist** o *distribution*: contendrá todos los archivos css, javascript y de fuentes (separados por carpetas) que se obtendrán automáticamente fruto de la compilación del tema. Por tanto, el usuario no modificará nunca directamente estos archivos.
- **inc** o *include*: contendrá las configuraciones básicas del tema, funciones *helpers* de ayuda a la carga de elementos, definiciones de tipos de contenido *custom* (cpt) y el núcleo de definición de módulos, campos de los módulos, renderización de los módulos y plantillas a las que se aplican dichos módulos.
- **src** o *source*: contendrá, separado por carpetas, todos los archivos fuente de estilos, javascript, imágenes estáticas y fuentes. Dichos archivos serán recogidos por el compilador para producir sus compilados en sus correspondientes subcarpetas de la carpeta dist.

Existirán también, por supuesto, en el nivel de la carpeta padre del tema bkt-modular, los archivos indispensables para el funcionamiento de WordPress, styles.css, index.php, y functions.php, aunque este último lo utilizaremos para cargar los inicializadores (archivos init.php) del resto de funcionalidades de las carpetas descritas anteriormente.

### 6.1.2. Definición de campos de los módulos

Los módulos tendrán asociados una serie de campos para que el usuario pueda configurarlos y así poder meter el contenido de dicho módulo. En la documentación de ACF se explica una

definición y funcionamiento de cada uno de ellos [7]. A continuación se hace un inventario de campos que tendrá cada uno de los módulos:

Módulo Título + Texto + CTA

- **Nombre del módulo**: Text
- **Funcionamiento**: este módulo es el más básico de todos y debe poder incluir un título y/o subtítulo, un texto y un botón. Todo ello de forma opcional, de tal forma que el cliente pueda elegir meter solo uno de esos elementos, varios o todos a la vez.

En la Figura 11 y Figura 12 podemos ver ejemplos del módulo aplicado en EBN Banco y Universidad Europea respectivamente:

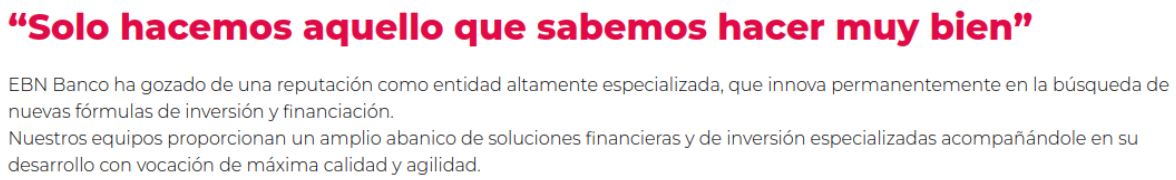



**Figura 11:** Ejemplo de módulo texto aplicado en la web www.ebnbanco.com


**Figura 12:** Ejemplo de módulo texto aplicado en la web www.alumni.universidadeuropea.es

- **Campos:**

**Tabla 15:** Campos del módulo de texto.

| Nombre | Tipo de campo ACF | Requerido | Condicionado a otro campo | Valor por defecto |
|---|---|---|---|---|
| Content | tab | - | no | - |
| Title | text | no | no | |
| Subtitle | textarea | no | no | |
| Text | wysiwyg | no | no | |

| | | | | |
|---|---|---|---|---|
| Styles | tab | - | no | - |
| Alignment | select | sí | no | left |
| Padding bottom | select | sí | no | normal |
| Padding top | select | sí | no | normal |

Módulo Texto + Imagen

- **Nombre del módulo**: Text + Image
- **Funcionamiento**: este módulo funciona de forma similar al anterior. Sin embargo, también permite meter una imagen, la cual será requerida para que tenga sentido el módulo. La disposición de la imagen con respecto al texto será escogida por el usuario pudiéndose colocar a la izquierda o a la derecha del texto.

En la Figura 13 podemos ver un ejemplo del módulo aplicado en la web de Universidad Europea.


**Figura 13:** Ejemplo de módulo texto + imagen aplicado en la web www.alumni.universidadeuropea.es

- **Campos:**

**Tabla 16:** Campos del módulo de texto + imagen.

| Nombre | Tipo de campo ACF | Requerido | Condicionado a otro campo | Valor por defecto |
|---|---|---|---|---|
| Content | tab | - | no | - |
| Title | text | no | no | |
| Subtitle | textarea | no | no | |
| Text | wysiwyg | no | no | |
| Image | image | sí | no | |
| Styles | tab | - | no | - |

| | | | | |
|---|---|---|---|---|
| Alignment | select | sí | no | left |
| Padding bottom | select | sí | no | normal |
| Padding top | select | sí | no | normal |
| Blocks alignment* | select | sí | no | text left, imageright |

*Las diferentes opciones de block alignment permitirán al usuario escoger la disposición de los bloques:

- Texto a la izquierda, imagen a la derecha.
- Texto a la derecha, imagen a la izquierda.

Módulo Banner

- **Nombre del módulo**: Banner
- **Funcionamiento**: este módulo funciona igual que el módulo de texto. Sin embargo, permite añadir una imagen de fondo que no será delimitada por un contenedor, sino que ocupará un *width* de 100% de la página.

En la Figura 14 podemos ver un ejemplo del módulo aplicado en la web de Universidad Europea.


**Figura 14:** Ejemplo de módulo texto + imagen aplicado en la web www.alumni.universidadeuropea.es

- **Campos:**

**Tabla 17:** Campos del módulo banner.

| Nombre | Tipo de campo ACF | Requerido | Condicionado a otro campo | Valor por defecto |
|---|---|---|---|---|
| Content | tab | - | no | - |
| Background image | image | sí | no | |
| Title | text | no | no | |

| | | | | |
|---|---|---|---|---|
| Subtitle | textarea | no | no | |
| Text | wysiwyg | no | no | |
| Button | link | no | no | |
| Button type* | select | no | no | Primary (square) |
| Styles | tab | - | no | - |
| Alignment | select | sí | no | left |
| Padding bottom | select | sí | no | normal |
| Padding top | select | sí | no | normal |

*Las diferentes opciones para el tipo de botón serán:

- Primary square: botón principal cuadrado.
- Secondary square: botón secundario cuadrado.
- Primary rounded: botón principal redondo.
- Secondary rounded: botón secundario redondo.
- Link: link subrayado.

Módulo Hero

- **Nombre del módulo**: Hero
- **Funcionamiento**: este módulo sirve para crear un *header* de página y dispondrá de los campos para que el cliente pueda meter los respectivos textos e imágenes. También se podrá usar el modo carrusel (*slider*) en el que habrá un pase de diapositivas con diferentes textos e imágenes.

En la Figura 15 y Figura 16 podemos ver un ejemplo del módulo aplicado en la web de IADE y de KPMG Tendencias respectivamente.

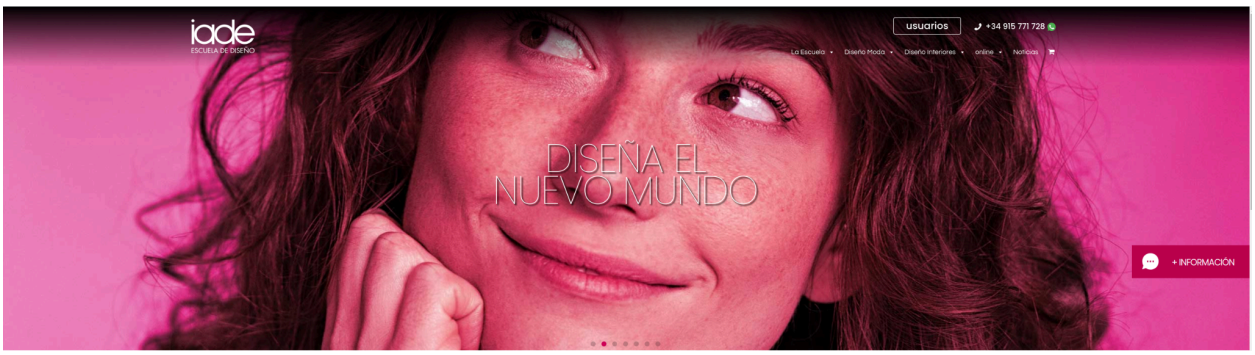



**Figura 15:** Ejemplo de módulo hero aplicado en la web www.iade.es


**Figura 16:** Ejemplo de módulo hero aplicado en la web [www.tendencias.kpmg.es](http://www.tendencias.kpmg.es)

- **Campos:**

**Tabla 18:** Campos principales del módulo hero.

| Nombre | Tipo de campo ACF | Requerido | Condicionado a otro campo | Valor por defecto |
|---|---|---|---|---|
| Content | tab | - | no | - |
| Mode* | select | sí | no | simple |
| Background image | image | sí | mode = simple | |
| Title | text | no | mode = simple | |
| Subtitle | textarea | no | mode = simple | |
| Text | wysiwyg | no | mode = simple | |
| Show slider controls | boolean | no | mode = slider | false |
| Show slider indicators | boolean | no | mode = slider | false |
| Slider interval in seconds | number | no | mode = slider | 5 |
| Cycle | boolean | no | mode = slider | true |

| | | | | |
|---|---|---|---|---|
| continuously | | | | |
| Slides** | repeater | sí | mode = slider | |
| Styles | tab | - | no | - |
| Alignment | select | sí | no | left |
| Padding bottom | select | sí | no | normal |

*El campo ‘mode’ permite seleccionar el modo en el que funcionará el módulo. Este podrá ser simple, que mostrará una imagen de fondo y los diferentes títulos, subtítulos y textos, o podrá ser tipo *slider*, el cual creará un carrusel de imágenes y textos.

**En caso de que el modo sea *slider*, aparecerá un repetidor de campos en el que cada elemento del repetidor dispondrá de los campos especificados en la Tabla 18 para formar las diferentes diapositivas del carrusel:

**Tabla 19:** Campos adicionales cuando el módulo hero se comporta como slider.

| Nombre | Tipo de campo ACF | Requerido | Condicionado a otro campo | Valor por defecto |
|---|---|---|---|---|
| Slide title | text | no | no | |
| Slide subtitle | textarea | no | no | |
| Slide text | wysiwyg | no | no | |
| Slide background image | image | sí | no | |

Módulo Distribuidor de contenido

- **Nombre del módulo**: Content distributor
- **Funcionamiento**: este módulo sirve para mostrar entradas a diferentes *posts* y contenidos de la web de forma visual y resumida mostrando por cada entrada una imagen destacada y su título, resumen, autor, etc. Los *posts* podrán ser seleccionados de forma manual o automática mostrando los de fecha más reciente, primero con un número máximo a mostrar y un número máximo de elementos por cada fila configurable por el cliente. Se podrá incluir un botón opcionalmente tipo “ver más”.

En la Figura 17 podemos ver un ejemplo del módulo aplicado en la web de Tendencias de KPMG.

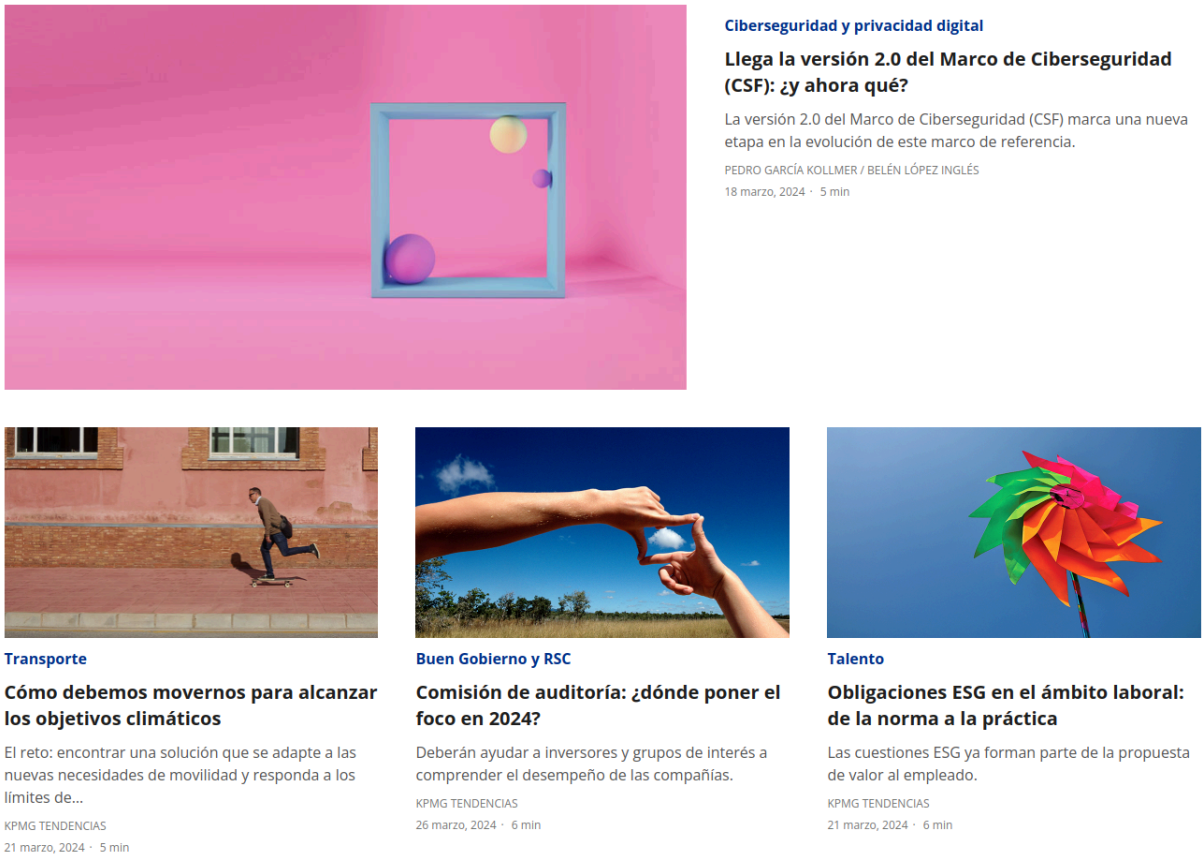



**Figura 17:** Ejemplo de módulo distribuidor de contenido en la web www.tendencias.kpmg.es

- **Campos:**

**Tabla 20:** Campos del módulo distribuidor de contenido.

| Nombre | Tipo de campo ACF | Requerido | Condicionado a otro campo | Valor por defecto |
|---|---|---|---|---|
| Content | tab | - | no | - |
| Title | text | no | no | |
| Subtitle | textarea | no | no | |
| Text | wysiwyg | no | no | |
| Items per row | number | sí | no | 3 |

| | | | | |
|---|---|---|---|---|
| Mode | select | sí | no | automatic |
| Maximum number of items | number | sí | mode = automatic | 3 |
| Button link to more content | link | no | no | |
| Button type* | select | no | no | Primary (square) |
| Posts** | relationship | sí | mode = manual | |
| Styles | tab | - | no | - |
| Alignment | select | sí | no | left |
| Padding bottom | select | sí | no | normal |
| Padding top | select | sí | no | normal |

*Las diferentes opciones para el tipo de botón serán:

- Primary square: botón principal cuadrado.
- Secondary square: botón secundario cuadrado.
- Primary rounded: botón principal redondo.
- Secondary rounded: botón secundario redondo.
- Link: link subrayado.

**Este campo mostrará un listado de posts entre los que seleccionar para establecer como post a mostrar en el distribuidor. En este caso el número máximo de ítems será ilimitado a todos los que el usuario seleccione.

Módulo Video

- **Nombre del módulo**: Video
- **Funcionamiento**: este módulo permite añadir un vídeo a la página pudiendo subirlo directamente como archivo al servidor o escogiendo uno de YouTube. Dispondrá de una imagen de preview para evitar cargar el video si el usuario no lo visualiza.

En la Figura 18 podemos ver un ejemplo del módulo aplicado en la web de Tendencias de KPMG.


**Figura 18:** Ejemplo de módulo video en la web www.tendencias.kpmg.es

- **Campos:**

**Tabla 21:** Campos del módulo video.

| Nombre | Tipo de campo ACF | Requerido | Condicionado a otro campo | Valor por defecto |
|---|---|---|---|---|
| Content | tab | - | no | - |
| Title | text | no | no | |
| Subtitle | textarea | no | no | |
| Source* | select | no | no | youtube |
| Video url** | oembed | sí | source = youtube | |
| Video file*** | file | sí | source = file | |
| Preview image**** | image | sí | no | |

| | | | | |
|---|---|---|---|---|
| Styles | tab | - | no | - |
| Alignment | select | sí | no | left |
| Padding bottom | select | sí | no | normal |
| Padding top | select | sí | no | normal |

*Se podrá seleccionar entre vídeos de YouTube o un archivo subido manualmente.

**Si se escoge modo YouTube aparecerá un campo para introducir la url de dicho video de YouTube.

***Si se selecciona *file*, aparecerá un campo para incluir el archivo. Los archivos permitidos para subir son mov, mp4, avi y wmv.

****Esta imagen servirá de portada para que quede visualmente más bonito y además para que el recurso del video no se descargue a menos que el usuario haga click en ella. Esto reduce el tiempo de carga de la web considerablemente, sobre todo para videos muy pesados.

Módulo FAQs

- **Nombre del módulo**: FAQs
- **Funcionamiento**: este módulo permite añadir una lista de preguntas y respuestas que se desplegarán o plegarán cuando el usuario haga clic en ellas.

En la Figura 19 podemos ver un ejemplo del módulo aplicado en la web de Alumni de Universidad Europea.


**Figura 19:** Ejemplo de módulo faqs en la web www.alumni.universidadeuropea.es

- **Campos:**

**Tabla 22:** Campos principales del módulo faq.

| Nombre | Tipo de campo ACF | Requerido | Condicionado a otro campo | Valor por defecto |
|---|---|---|---|---|
| Content | tab | - | no | - |
| Title | text | no | no | |
| Subtitle | textarea | no | no | |
| Text | wysiwyg | no | no | |
| Faqs* | repeater | sí | no | |
| Styles | tab | - | no | - |
| Alignment | select | sí | no | left |
| Padding bottom | select | sí | no | normal |
| Padding top | select | sí | no | normal |

*El campo repetidor faqs dispondrá a su vez de los siguientes campos para formar las diferentes preguntas y respuestas:

**Tabla 23:** Campos de los ítems repetibles del módulo faq.

| Nombre | Tipo de campo ACF | Requerido | Condicionado a otro campo | Valor por defecto |
|---|---|---|---|---|
| Faq question | text | sí | no | |
| Faq answer | wysiwyg | sí | no | |

Módulo Script

- **Nombre del módulo**: Script
- **Funcionamiento**: este módulo permite añadir un script al cliente.

Ejemplos del módulo aplicado a una web real (EBN Banco):


**Figura 20:** Ejemplo de módulo script en la web [www.ebnbanco.com](http://www.ebnbanco.com)

En la Figura 20 podemos ver un ejemplo del módulo aplicado en la web de EBN Banco.

- **Campos:**

**Tabla 24:** Campos del módulo script.

| Nombre | Tipo de campo ACF | Requerido | Condicionado a otro campo | Valor por defecto |
|---|---|---|---|---|
| Content | tab | - | no | - |
| Title | text | no | no | |
| Subtitle | textarea | no | no | |
| Script* | textarea | sí | no | |
| Styles | tab | - | no | - |
| Alignment | select | sí | no | left |
| Padding bottom | select | sí | no | normal |
| Padding top | select | sí | no | normal |

Módulo Separador

- **Nombre del módulo**: Separador
- **Funcionamiento**: este módulo permite añadir una separación extra entre módulos y cambiar los comportamientos de las separaciones.
- **Campos:**

**Tabla 25:** Campos del módulo separador.

| Nombre | Tipo de campo ACF | Requerido | Condicionado a otro campo | Valor por defecto |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Settings | tab | - | no | - |
| Desktop* | select | sí | no | Normal (90px) |
| Mobile** | select | sí | no | Normal (60px) |

*En el campo desktop se podrá seleccionar entre las siguientes opciones:

- Narrow (60px)
- Normal (90px)
- Wide (120px)

**En el campo mobile se podrá seleccionar entre las siguientes opciones:

- Narrow (40px)
- Normal (60px)
- Wide (90px)

## 6.2. Implementación

En esta sección se describe y explica cómo el código funciona en su conjunto y se profundiza también en sus mecánicas fundamentales. Por tanto, se enseñan algunas de las funciones más básicas del sistema.

### 6.2.1. Lógica genera

Todo el código está pensado para que sea autoinclusivo en su sistema de archivos. Es decir, todo está preparado para que el usuario, cuando desarrolle nuevos módulos o tipos de contenidos, no tenga que añadir a mano las importaciones de los nuevos archivos en los *arrays* de módulos disponibles para cada contenido.

En este apartado se van a explicar algunas de las funciones de ayuda implementadas para poder entender los segmentos de código de las siguientes subsecciones de este capítulo. Dentro de la carpeta /inc/acf tenemos dichas funciones.

**Función mod_acf_fieldkey y función mod_acf_fieldname:**

```
function mod_acf_fieldkey($module_name, $field_name, $parent_field_name = '') {
   $parent_field_name = ( !empty($parent_field_name) ? md5( "_{$parent_field_name}") : '' );
   return "field_{$module_name}_" . md5( "{$module_name}_{$field_name}{$parent_field_name}" );
}
```

```
function mod_acf_fieldname($module_name, $field_name) {
   return "{$module_name}_{$field_name}";
}
```

Lo primero que tenemos que entender es que ACF necesita un nombre y un identificador único para cada campo. Cuando se generan los campos de la forma tradicional (creándolo en la base de datos) ACF asigna un identificador único, sin embargo nosotros estamos definiendo el campo mediante código y no almacenamos dicha definición en base de datos. Cada vez que

WordPress cargue su inicialización tiene que tener la misma referencia de key y nombre para cada campo y de esta forma poder encontrar el valor del campo almacenado (ese sí) en base de datos.

### 6.2.2. Lógica de módulos

Como se ha explicado en secciones anteriores, la carpeta /inc/acf contiene toda la funcionalidad de módulos y plantillas asociadas. El archivo 'init' contiene la inicialización de funciones y la importación de cada módulo que se encuentre en la carpeta flexible_modules, además de la definición de campos comunes de estilos de módulo. Como se puede apreciar en la Figura 21, las carpetas que contienen los archivos de definición y renderizado tienen la siguiente nomenclatura

*module_nombre_del_módulo*

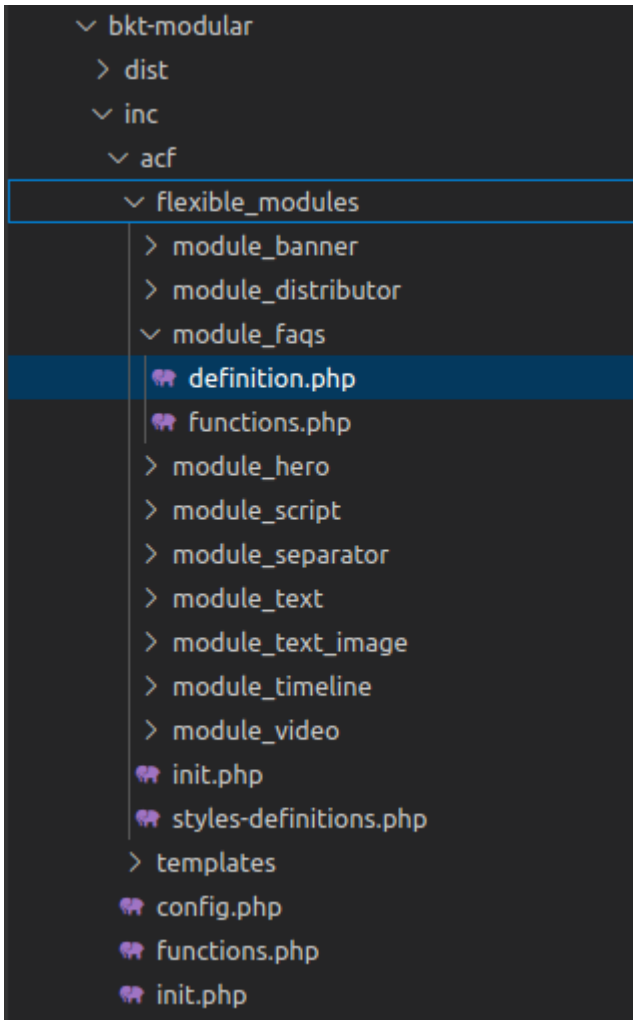



**Figura 21:** Estructura de archivos de los diferentes módulos.

Y dichos carpetas se componen siempre de dos archivos:

- definitions.php
- functions.php

El archivo definitions.php contiene toda la definición de campos ACF que el módulo necesita y que se ha detallado en la sección de diseño. Siempre tendrá la siguiente estructura:

1. Se define las variables module_name y group_ {module_name} a través del nombre de la carpeta. Esto nos servirá para pasárselo a las funciones anteriormente descritas, que mediante un md5 nos calculen un hash para tener identificadores únicos en cada campo del módulo.
2. Se retorna un *array* de *arrays* con la información del módulo y toda la colección definida de campos que tiene. A continuación se muestra el código de uno de los módulos más sencillos a modo de ejemplo:

```php
<?php
$module_name = basename( __DIR__ );
$group_{$module_name} = "bkt_{$module_name}_" . md5( $module_name );

return array(
   $group_{$module_name} => array(
       'key' => $group_{$module_name},
       'name' => $module_name,
       'label' => __('Text', BKT_MODULAR_THEME_LANG_DOMINE),
       'display' => 'block',
       'sub_fields' => array_merge(
           array(
               array(
                   'key' => mod_acf_fieldkey( $module_name, 'content' ),
                   'name' =>  mod_acf_fieldname( $module_name, 'content' ),
                   'label' => __('Content', BKT_MODULAR_THEME_LANG_DOMINE),
                   'type' => 'tab',
                   'instructions' => '',
                   'required' => 0,
                   'conditional_logic' => 0,
                   'wrapper' => array(
                       'width' => '',
                       'class' => '',
                       'id' => '',
                   ),
                   'placement' => 'left',
                   'endpoint' => 0,
               ),
               array(
                   'key' => mod_acf_fieldkey( $module_name, 'title' ),
                   'name' => mod_acf_fieldname( $module_name, 'title' ),
                   'label' => __('Title', BKT_MODULAR_THEME_LANG_DOMINE),
                   'type' => 'text',
                   'instructions' => '',
                   'required' => 0,
                   'conditional_logic' => 0,
                   'wrapper' => array(
                       'width' => '',
                       'class' => '',
                       'id' => '',
                   ),
                   'default_value' => '',
                   'placeholder' => '',
                   'maxlength' => '',
                   'rows' => 3,
                   'new_lines' => '',
               ),
               array(
```

```
                    'key' => mod_acf_fieldkey( $module_name, 'subtitle' ),
                    'name' => mod_acf_fieldname( $module_name, 'subtitle' ),
                    'label' => __('Subtitle', BKT_MODULAR_THEME_LANG_DOMINE),
                    'type' => 'textarea',
                    'instructions' => '',
                    'required' => 0,
                    'conditional_logic' => 0,
                    'wrapper' => array(
                        'width' => '',
                        'class' => '',
                        'id' => '',
                    ),
                    'default_value' => '',
                    'placeholder' => '',
                    'maxlength' => '',
                    'rows' => 1,
                    'new_lines' => '',
                ),
                array(
                    'key' => mod_acf_fieldkey( $module_name, 'text' ),
                    'name' => mod_acf_fieldname( $module_name, 'text' ),
                    'label' => __('Text', BKT_MODULAR_THEME_LANG_DOMINE),
                    'type' => 'wysiwyg',
                    'instructions' => '',
                    'required' => 0,
                    'conditional_logic' => 0,
                    'wrapper' => array(
                        'width' => '',
                        'class' => '',
                        'id' => '',
                    ),
                    'default_value' => '',
                    'placeholder' => '',
                    'maxlength' => '',
                    'rows' => 1,
                    'new_lines' => '',
                ),
            ),
            styles_options(
                $module_name, array(
                    'alignment',
                    'bottom',
                    'top'
                )
            ),
        ),
        'min' => '',
        'max' => '',
    )
);
```

El archivo functions.php se encarga de la lectura de campos y renderizado del módulo, además de ser aquí donde se tuviera que definir cualquier otra función relacionada con el módulo.

```
<?php

function bkt_flex_module_text_content () {
    $module_name = basename( __DIR__ );
    $title = get_sub_field($module_name.'_title');
    $subtitle = get_sub_field($module_name.'_subtitle');
    $text = get_sub_field($module_name.'_text');
    $alignment = get_sub_field($module_name.'_alignment');
    $padding_bottom = get_sub_field($module_name.'_padding_bottom');
    $padding_top = get_sub_field($module_name.'_padding_top');
    ?>
```

```
        <section class="mod-text mod-flexible mod-alignment-<?php echo $alignment; ?> mod-padding-top-<?php echo $padding_top;
?> mod-padding-bottom-<?php echo $padding_bottom; ?>">
            <div class="container">
                <?php if($title || $subtitle) : ?>
                    <header class="header">
                        <?php if($title) : ?>
                            <h2 class="title"><?php echo $title; ?></h2>
                        <?php endif; ?>
                        <?php if($subtitle) : ?>
                            <div class="subtitle"><?php echo $subtitle; ?></div>
                        <?php endif; ?>
                    </header>
                <?php endif; ?>
                <div class="content">
                    <div class="text"><?php echo $text; ?></div>
                </div>
            </div>
        </section>
    <?php
    }
```

Con estos archivos base ya podemos entonces hacer tanto que se puedan rellenar los campos con los valores correspondientes en el *dashboard* de WordPress como renderizarlos en el front. Para ello utilizamos dos funciones:

Tpl_load_modules, cuando es llamada carga los la definición de módulos.

```
    function tpl_load_modules($modules_names = array()) {
        $modules_dir = __DIR__;
        $mods = array();
        foreach ($modules_names as $module_name) {
            $module_definition = "$modules_dir/$module_name/definition.php";
            $module_functions = "$modules_dir/$module_name/functions.php";
            if(file_exists($module_definition) && file_exists($module_functions)) {
                $fields = require($module_definition);
                require_once($module_functions);
                if(is_array($fields)) {
                    $mods = array_merge($mods, $fields);
                }
            }
        }

        return $mods;
    }
```

Tpl_renders, cuando es llamada renderiza todos los módulos que estén guardados en la página/post.

```
    function tpl_renders($field_name) {
        if(have_rows($field_name)) {
            while (have_rows($field_name)) : the_row();
                $module_name = get_row_layout();
                $function_to_call = "bkt_flex_{$module_name}_content";
                if(function_exists($function_to_call)) {
                    $function_to_call();
                }
            endwhile;
        }
    }
```

### 6.2.3. Lógica de tipos de contenido

Existe una funcionalidad que permite al usuario habilitar y deshabilitar el uso de ciertos módulos para cada distinto tipo de contenido (páginas, *post*, *custom post types*) desde una página de configuración de módulos para cada tipo de contenido. Para ellos son básicas dos funciones.

Función get_flexible_module_list:

```
function get_flexible_modules_list() {
    $modules = array();
    $path = get_template_directory() . '/inc/acf/flexible_modules';
    if (is_dir($path)) {
        $dir = opendir($path);
        while (($file = readdir($dir)) !== false) {
            if ($file != '.' && $file != '..' && is_dir($path . '/' . $file)) {
                $modules[$file] = ucfirst(str_replace('_', ' ', $file));
            }
        }
        closedir($dir);
    }
    return $modules;
}
```

Esta función se encarga de buscar todas las carpetas dentro /inc/acf/flexible_modules. Estas carpetas tendrán el nombre *module_nombre_del_módulo*, y por ello podremos identificar el nombre de todos los módulos existentes.

Función get_content_types_list:

```
function get_content_types_list() {
    $templates_path = get_template_directory() . '/inc/acf/templates';
    $content_types = array();

    if (is_dir($templates_path)) {
        $dir = opendir($templates_path);
        while (($file = readdir($dir)) !== false) {
            if ($file != '.' && $file != '..' && is_file($templates_path . '/' . $file)) {
                $file_parts = pathinfo($file);
                if ($file_parts['extension'] === 'php') {
                    $content_types[] = str_replace('_', '', $file_parts['filename']);
                }
            }
        }
        closedir($dir);
    }

    return $content_types;
}
```

Dicha función busca todas las plantillas de contenido disponibles, inicialmente habrá las básicas de WordPress (*page* y *post*), pero el desarrollador puede añadir diferentes *custom post type* añadiendo a esta carpeta su plantilla y el sistema incluirá todo automáticamente a la página de configuración de módulos, cuyo código se muestra a continuación:

```
<?php
if( function_exists('acf_add_options_page') ) {
    acf_add_options_page(array(
        'page_title'    => __('Configuración de Módulos', BKT_MODULAR_THEME_LANG_DOMINE),
        'menu_title'    => __('Configuración de Módulos', BKT_MODULAR_THEME_LANG_DOMINE),
        'menu_slug'     => 'module-configuration',
        'capability'    => 'edit_posts',
        'redirect'      => false
```

```
    ));
}

add_action('acf/init', 'create_acf_fields_for_modules');
function create_acf_fields_for_modules() {
    if( function_exists('acf_add_local_field_group') ) {

        $content_types = get_content_types_list();
        $modules = get_flexible_modules_list();

        foreach ($content_types as $content_type) {
            $fields = array();

            foreach ($modules as $module_key => $module_label) {
                $fields[] = array(
                    'key' => 'field_' . $content_type . '_' . $module_key,
                    'label' => $module_label,
                    'name' => $content_type . '_' . $module_key,
                    'type' => 'true_false',
                    'instructions' => '',
                    'required' => 0,
                    'conditional_logic' => 0,
                    'wrapper' => array(
                        'width' => '25',
                        'class' => '',
                        'id' => '',
                    ),
                    'message' => '',
                    'default_value' => 1,
                    'ui' => 1,
                    'ui_on_text' => __('Habilitado', BKT_MODULAR_THEME_LANG_DOMINE),
                    'ui_off_text' => __('Deshabilitado', BKT_MODULAR_THEME_LANG_DOMINE),
                );
            }

            acf_add_local_field_group(array(
                'key' => 'group_' . $content_type,
                'title' => ucfirst($content_type) . ' Modules Configuration',
                'fields' => $fields,
                'location' => array(
                    array(
                        array(
                            'param' => 'options_page',
                            'operator' => '==',
                            'value' => 'module-configuration',
                        ),
                    ),
                ),
                'menu_order' => 0,
                'position' => 'normal',
                'style' => 'default',
                'label_placement' => 'top',
                'instruction_placement' => 'label',
                'hide_on_screen' => '',
                'active' => true,
                'description' => '',
            ));
        }
    }
}
```

### 6.2.4. Lógica de botones

Además de la modularidad se ha incluido una funcionalidad para añadir botones con selección del propio estilo del botón desde las cajas enriquecidas de edición de texto de WordPress como se muestra en la Figura 22.

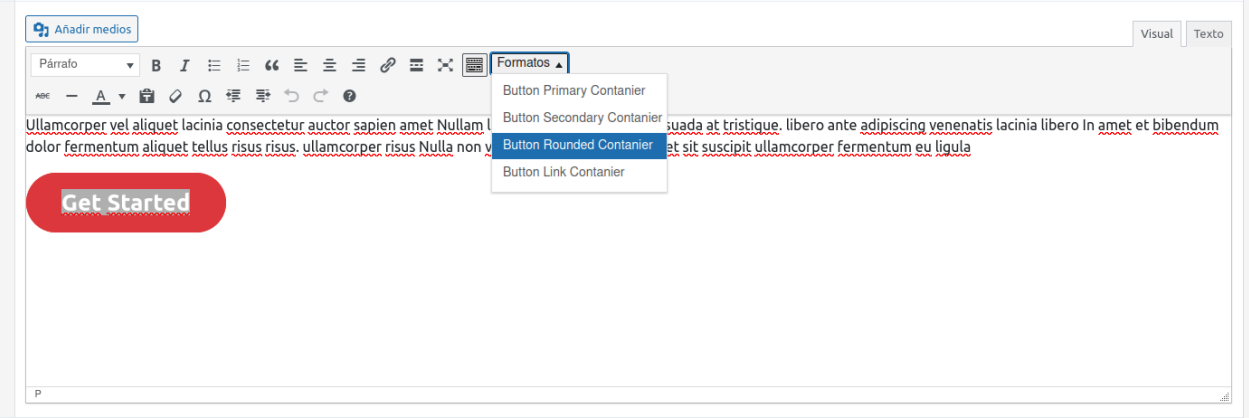



**Figura 22:** Funcionalidad de estilizado de botones en las cajas enriquecidas.

Para ello, se ha definido en el filtro que carga antes de la inicialización de las cajas de texto enriquecido los formatos que debe haber disponibles para los botones. Estos formatos son botón primario, secundario, redondeado y link. Además se han añadido los estilos al *admin* de WordPress por lo que pueden ser visualizados desde la propia caja de edición.

```php
<?php

add_filter(
    'init',
    function(){
        add_theme_support('editor-styles');

        $content_css = get_stylesheet_directory_uri() . '/dist/css/tinymce.css';
        add_editor_style($content_css);
    }
);

add_filter(
    'tiny_mce_before_init',
    function($settings) {

        // Content CSS
        $content_css = get_stylesheet_directory_uri() . '/dist/css/tinymce.css';
        if ( isset($settings['content_css']) ) {
            $content_css = $settings['content_css'] . ',' . $content_css;
        }
        $settings['content_css'] = $content_css;

        // Style Formats

        $style_formats = array(
            array(
                'title'   =>  __('Button Primary Contanier', BKT_MODULAR_THEME_LANG_DOMINE),
                'block'   =>  'div',
                'classes' =>  'btn-primary'
            ),
            array(
                'title'   =>  __('Button Secondary Contanier', BKT_MODULAR_THEME_LANG_DOMINE),
                'block'   =>  'div',
                'classes' =>  'btn-secondary'
```

```
            ),
            array(
                'title'   =>  __('Button Rounded Contanier', BKT_MODULAR_THEME_LANG_DOMINE),
                'block'   =>  'div',
                'classes' =>  'btn-rounded'
            ),
            array(
                'title'   =>  __('Button Link Contanier', BKT_MODULAR_THEME_LANG_DOMINE),
                'block'   =>  'div',
                'classes' =>  'btn-link'
            )
        );
        $settings['style_formats'] = json_encode ($style_formats);

        return $settings;
    }
);
```

### 6.2.5. Lógica de estilos

Hasta ahora se ha mostrado todo el código encargado de la lógica básica de plantillas de tipo de contenido y módulos para poder configurar y renderizar los módulos en el *front,* ya que es la parte más importante del sistema. Pero también se necesita una lógica de estilos que proporcione unos estilos básicos a cada módulo y que permita al usuario modificarlos a su gusto. Para ello, el código se ha organizado en una serie de archivos *core* (los que son básicos y no se deben modificar) y unos archivos *custom* (los que sí se pueden modificar) que, mediante herencia, contendrán todo los estilos base más los que el usuario defina para la web en cuestión que se necesite construir. Dicha estructura se puede ver en la Figura 23.

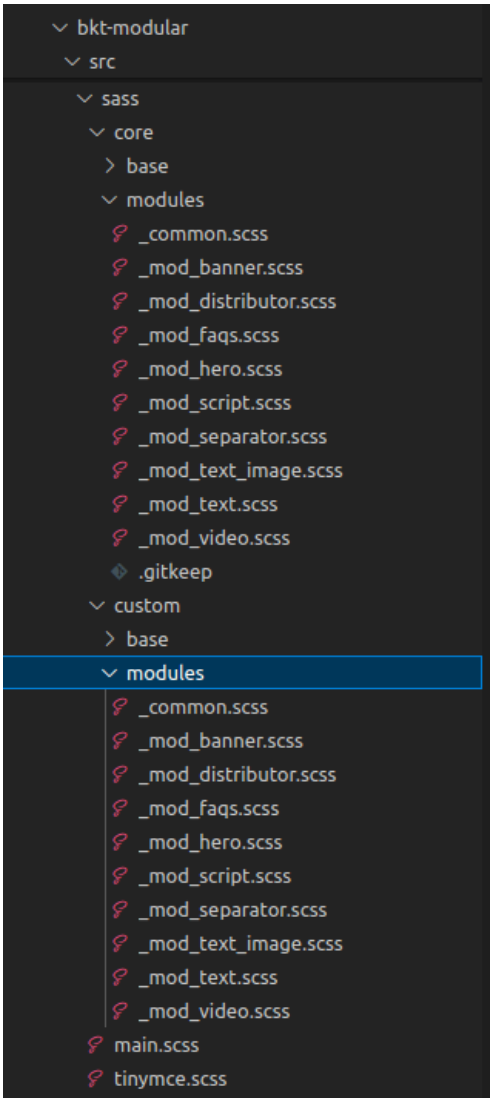



**Figura 23:** Estructura de archivos de estilos de los diferentes módulos.

Pongamos de ejemplo el módulo de preguntas y respuestas (faqs). Su archivo de estilos básicos estará en */src/sass/core/module/_mod_faqs.scss:*

```
%mod-faqs {
 color: $black;

 .accordion-item {
   text-align: left;
 }
}
```

En el core del módulo le diremos que queremos que el color del texto sea negro y que los textos de los ítems estén alineados a la izquierda.

En su archivo custom tendremos:

```
.mod-faqs {
 @extend %mod-faqs;
}
```

Y un ejemplo de personalización custom para aplicarlo al desarrollo de una web en concreta podría ser el siguiente:

```
.mod-faqs {
 @extend %mod-faqs;

 font-size: 1.6rem;

 header {
   padding-bottom: 2rem;

   .title {
     font-size: 3.2rem;
     position: relative;
     text-transform: uppercase;
   }
 }

 &.mod-alignment-center {
   .title {
     padding-bottom: 2rem;

     &::after {
       background: $theme-red;
       bottom: 0;
       content: '';
       display: block;
       height: 3px;
       left: calc(50% - 25px);
       position: absolute;
       width: 50px;
     }
   }
 }

 .accordion-header {
   .accordion-button {
     box-shadow: $theme-shadow 0 -1px 0 0 inset;
     cursor: pointer;
     font-size: 1.8rem;
     font-weight: 400;
     line-height: 2.4rem;

     &:hover {
       color: $theme-red;

       &::after {
         border-color: $theme-red;
       }
     }

     &:not(.collapsed) {
       background-color: $transparent;
       color: $theme-red;

       &::after {
         border-color: $theme-red;
         transform: rotate(-135deg);
       }
     }
```

```
      &::after {
        background-image: none;
        border: solid $theme-text;
        border-width: 0 .2rem .2rem 0;
        content: '';
        display: inline-block;
        height: .8rem;
        margin-left: auto;
        transform: rotate(45deg);
        transition: transform .2s;
        width: .8rem;
      }
    }
  }

  .accordion-item {
    border: 0;
    color: $theme-text;
  }
}
```

### 6.2.6. Herramientas utilizadas

**XAMP**

Según la definición de su propia página web, XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl. El paquete de instalación de XAMPP ha sido diseñado para ser increíblemente fácil de instalar y usar [8].

**Gitkraken**

Es una herramienta que nos permite gestionar los repositorios de git de manera fácil y sencilla a través de una interfaz muy intuitiva. Es ideal para separar en ramas las diferentes funcionalidades elaboradas de forma independiente a lo largo del desarrollo del código tal y como se ha expuesto en la sección Git Flow de este documento [9].

**NPM**

NPM (Node Package Manager o Administrador de Paquetes de Node) es un sistema de gestión de paquetes para el lenguaje de programación JavaScript. Es el administrador de paquetes por defecto para el entorno de ejecución de JavaScript Node.js, pero también puede ser utilizado para gestionar paquetes de *frontend* para aplicaciones web. NPM facilita a los desarrolladores compartir y reutilizar código. Permite a los usuarios instalar, compartir y gestionar dependencias (bibliotecas y herramientas) en sus proyectos de software. Estas dependencias se definen en un archivo llamado “package.json”, el cual contiene información sobre el proyecto y sus dependencias. Los desarrolladores pueden publicar sus propios paquetes en el registro de NPM para que otros los usen, haciendo que la colaboración y el intercambio de soluciones a problemas comunes sean mucho más accesibles. Además de gestionar paquetes, NPM también ofrece herramientas para ayudar a los desarrolladores a administrar versiones de paquetes, ejecutar scripts y generar proyectos. Con miles de paquetes disponibles, NPM se ha convertido en una herramienta esencial para el desarrollo [10].

**NVM**

NVM (Node Version Manager o Administrador de Versiones de Node). Es una herramienta que permite a los desarrolladores instalar y gestionar múltiples versiones de Node.js. Esto es especialmente útil porque diferentes proyectos pueden requerir diferentes versiones de Node.js para funcionar correctamente. Con nvm, los desarrolladores pueden cambiar fácilmente entre versiones de Node.js, permitiéndoles trabajar en varios proyectos con diferentes requisitos de versión sin conflictos.

Algunas características clave de nvm incluyen:

- La capacidad de instalar cualquier versión de Node.js.
- La facilidad para cambiar entre versiones instaladas con un simple comando.
- La opción de usar versiones específicas de Node.js por proyecto, especificadas en un archivo .nvmrc.

NVM es una herramienta de línea de comandos que se instala en el sistema del usuario y es ampliamente utilizado en el desarrollo de software para gestionar entornos de Node.js de forma eficiente [11].

**VSC**

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es gratuito, de código abierto y está disponible para Windows, macOS y Linux. VS Code está diseñado para ser un editor de código ligero pero poderoso que puede usarse para el desarrollo de una amplia variedad de aplicaciones. Ofrece soporte para la depuración, control de versiones Git integrado, resaltado de sintaxis, completado inteligente de código, refactorización de código y muchas otras características.

VS Code soporta una amplia gama de lenguajes de programación como JavaScript, TypeScript, Python, PHP, C++, C#, Java entre otros, gracias a su sistema de extensiones. Estas extensiones permiten a los usuarios añadir lenguajes, depuradores y herramientas para un conjunto más amplio de desarrollo, adaptando el editor a sus necesidades específicas.

Una de las características más apreciadas de VS Code es su eficiencia y rendimiento, operando rápidamente incluso en proyectos grandes, y su capacidad de personalización, permitiendo a los usuarios modificar aspectos de su entorno de desarrollo para que se ajuste mejor a sus preferencias o necesidades.

# Capítulo 7

# 7. Conclusiones y trabajo futuro

El resultado final de este Trabajo Fin de Grado ha sido la implementación de un sistema modular en formato de tema de WordPress que permitirá a la empresa ahorrar costes de desarrollo para futuros proyectos que necesiten de un tema modular para WordPress. El marco de desarrollo incluye unas plantillas de módulos y de tipos de contenido, que permiten tanto la personalización de los módulos como la habilitación o deshabilitación de estos en los diferentes tipos de contenido que puede tener una página en WordPress. Todo ello pensando en la máxima flexibilidad posible.

Teniendo en cuenta las diferentes tareas que se deben realizar a la hora de desarrollar una página web en WordPress con un sistema modular, se estima que este tema modular pueda ahorrar un 30% del trabajo total de desarrollo. Es decir, que si un desarrollo WordPress costase por ejemplo un total de 450 horas de desarrollo, podría reducirse a 315 horas lo que es muy significativo.

En el futuro se pretende ampliar el sistema para que contenga muchos más módulos ya definidos. Hasta ahora se han implementado los más habituales, pero se puede ir haciendo una inclusión de todos los módulos que los clientes puedan solicitar a medida que se vayan necesitando para tenerlos disponibles para siguientes proyectos y ahorrar aún más tiempo de trabajo.

Otra línea de investigación futura podría ser la optimización de los Core Web Vitals que usa Google para clasificar las puntuaciones de velocidad, usabilidad y accesibilidad de una web para dejar el sistema totalmente preparado para obtener las máximas puntuaciones posibles y escalar en el posicionamiento orgánico.

# Apéndice A

# Manual de usuario

<29.05.2024><v1>

**Histórico de revisiones:**

| Día | Versión | Descripción |
|---|---|---|
| 29.05.2024 | 1.0 | Guía para uso y ampliación de la plantilla Brooktec WP Modular |

Este documento pretende ser una pequeña guía de uso, modificación y ampliación de la plantilla tema de WordPress Brooktec WP Modular para uso de desarrollador. En ningún momento pretende ser una guía para el usuario cliente.

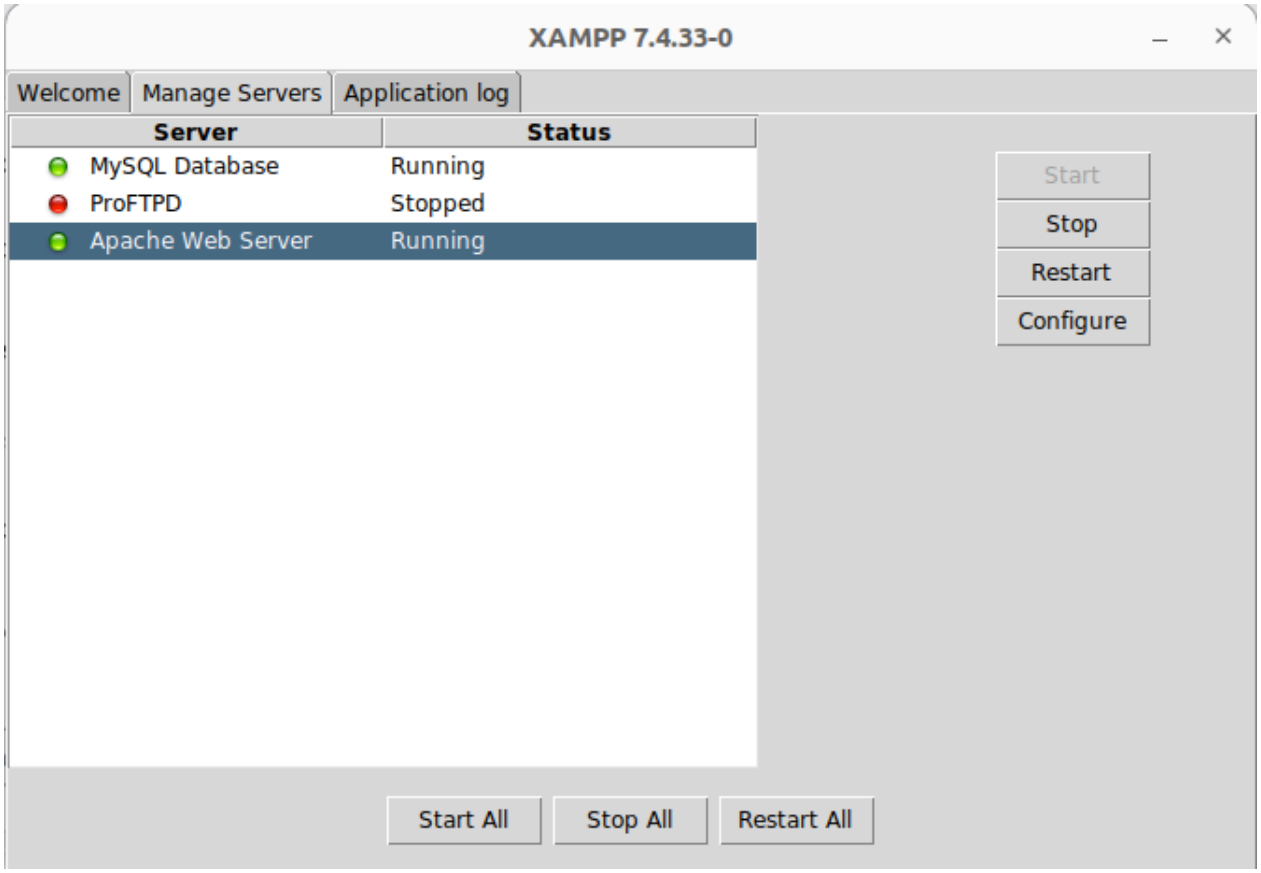
**Nota previa**

Se asume que el usuario tiene unos conocimientos mínimos de administración de WordPress, desarrollo WordPress y lo que ello conlleva (conocimiento de PHP, HTML, CSS, Javascript y funciones de WordPress). En esta guía no se explica el funcionamiento básico de WordPress, ni a nivel administrativo ni a nivel de programación.

## Descarga, instalación y puesta a punto

En primer lugar, tendremos que hacer una descarga del código del proyecto desde el repositorio correspondiente, actualmente albergado en https://github.com/Juanma121/bkt-wp-modular, aunque posteriormente se moverá a un repositorio del equipo de Brooktec. Por lo tanto, se deberán disponer de permisos para realizar dicha acción.

Una vez obtenidos dichos permisos y realizada la descarga lo primero que debemos hacer es montarnos el proyecto como si fuese una instalación de WordPress limpia. Para ello, podemos montarlo con Xamp (Windows), Lamp (Linux), Mamp (Mac Os) o con cualquier programa que nos permita levantar los servicios de Apache, MySQL y PHP.




Una vez tengamos la ruta asociada a la instalación de la carpeta del proyecto en nuestro programa AMP e inicializado el programa, entraremos a nuestro navegador a la ruta establecida del proyecto, en este caso http://localhost/bkt-wp-modular/.

Al tratarse de una instalación limpia, WordPress detectará automáticamente que no hay una base de datos creada para el proyecto y por tanto pedirá al usuario crear una nueva base de datos. Entre dichos datos solicitará un nombre de la base de datos, un usuario, un contraseña y el host a elección del usuario:


Tras este paso, tendremos creada una base de datos limpia de WordPress en donde se empezarán a almacenar todos los datos de nuestra instalación. Si vamos al código del proyecto y vemos su estructura de archivos, nos debería haber creado un archivo llamado **wp-config.php**. Dicho archivo está ignorado en el gitignore y no se debe nunca subir al repositorio, ya que contiene los datos locales de conexión a la base de datos local de cada desarrollador. Si necesitamos conectar con otra base de datos ya creada o anterior podemos modificar dicho archivo a nuestro gusto para hacerlo.

En primer lugar, vamos a establecer el tema modular que se ha desarrollado en este proyecto como tema por defecto de WordPress. Normalmente, cuando hacemos una instalación ordinaria de WordPress viene con una serie de temas ejemplo por defecto, las series twentytwenty ( twentytwentyone, twentytwentytwo, twentytwentythree, twentytwentyfour ). Sin embargo, para ahorrar espacio en repositorio y no meter archivos innecesarios se han eliminado del repositorio, siendo ignorados en el gitignore. Por tanto, la instalación limpia de WordPress no va a encontrar un tema que asociar por defecto si no se lo especificamos.

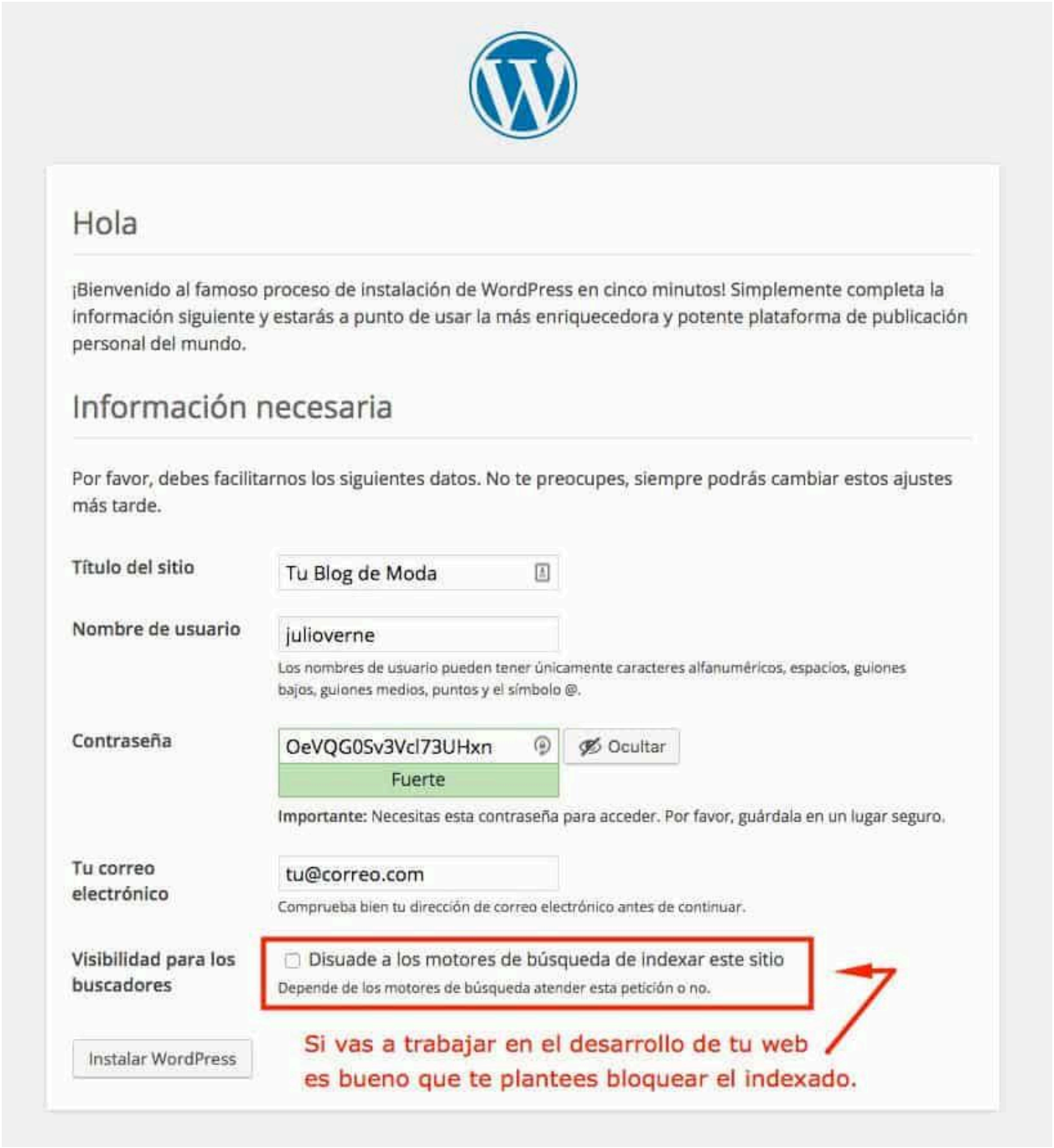
Para especificar dicho tema iremos al archivo wp-config-sample.php y buscaremos la definición de la variable **WP_DEFAULT_THEME**, la copiaremos y la pegaremos en el mismo

lugar en la el archivo wp-config.php. Esto hará que se establezca como tema predeterminado el tema Brooktec WP Modular.

```
define( 'WP_DEFAULT_THEME', 'bkt-modular' );
```

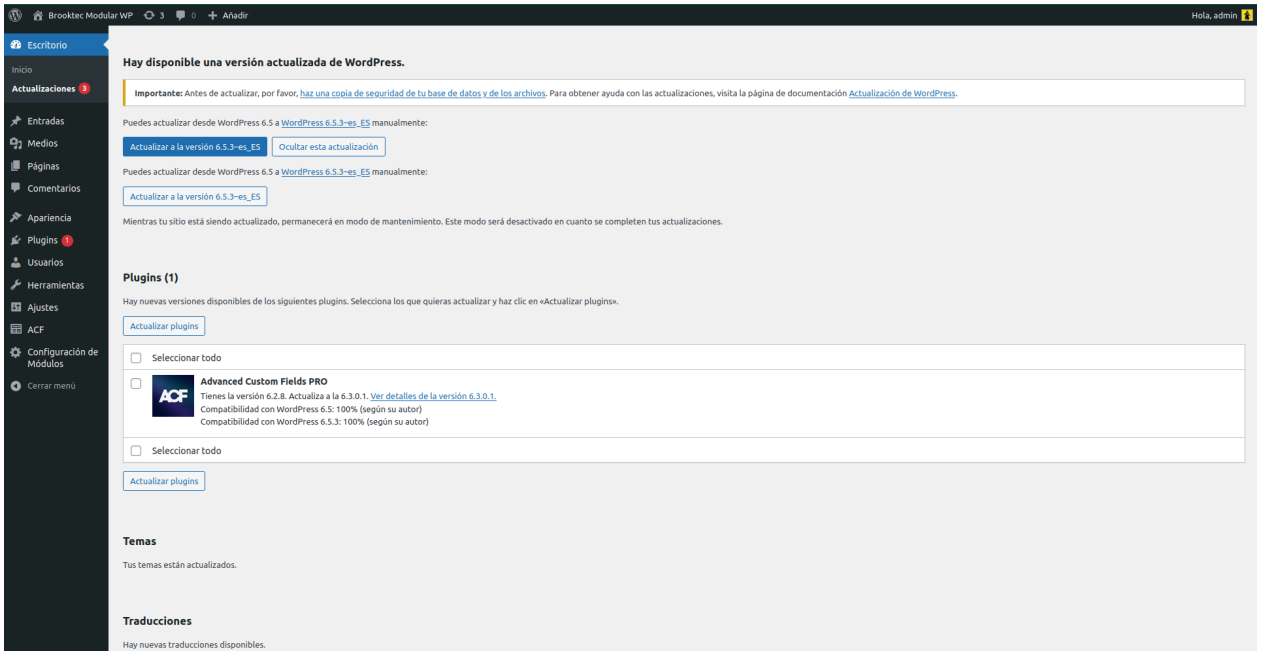
A continuación, en el navegador nos pedirá el título del *site*, el nombre del usuario administrador, una contraseña y un correo electrónico asociado. Podemos dar ya el título que tendrá el site que estamos desarrollando si lo sabemos y vamos a portar en un futuro la base de datos local a preproducción o producción. No obstante dicho título y el resto de datos podrán ser modificados posteriormente.

También nos dará la opción de evitar que los buscadores indexen el sitio web. Esta opción está bien si estamos trabajando directamente en un servidor de preproducción abierto al público para que no se indexe mientras se hace el desarrollo. Sin embargo, como no es nuestro caso, dejaremos la visibilidad estándar para evitar olvidarnos de desactivar esta opción en producción y perder visibilidad en los motores de búsqueda.




Una vez realizados estos pasos la instalación, estará lista y podremos acceder al *backoffice* entrando en nuestra ruta de desarrollo local seguido de /wp-admin. Nos pedirá iniciar sesión de administrador y tras introducir el usuario y contraseña accederemos al dashboard de WordPress. Iremos a Apariencia/Temas y en caso de que no se hubiera activado por defecto el Tema Brooktec Modular, lo activaremos pinchando en 'activar'. Hecho esto, pasaremos a comprobar que disponemos de las últimas actualizaciones de WordPress y *plugins*. El repositorio se actualizará periódicamente para intentar que siempre esté disponible nada más descargar el

repositorio la última versión de WordPress y de ACF PRO. Sin embargo, podría ser que en el momento de la instalación hubiera alguna actualización pendiente. Este es buen momento para realizar una actualización en local y empezar el desarrollo con todo actualizado. Para ello, nos metemos en Escritorio/Actualizaciones y llevamos todo a la última versión.




Vayamos ahora a la instalación de paquetes y compilación de tema. En terminal, accedemos a la ruta donde tengamos instalado el proyecto y hacemos change directory hasta la siguiente ruta:

*wp-content/themes/bkt-modular*

Una vez en ella, ejecutamos el comando nvm use para leer el archivo nvmrc y pasar a usar la versión de node que está especificada en dicho archivo.

```
juan@juan-Modern-14-C12M:~/Documentos/Proyectos/WordPress/bkt-wp-modular/wp-content/themes/bkt-modular$ nvm use
Found '/home/juan/Documentos/Proyectos/WordPress/bkt-wp-modular/wp-content/themes/bkt-modular/.nvmrc' with version <v16.16.0>
Now using node v16.16.0 (npm v8.11.0)
```

Instalamos las dependencias de módulos establecidos en el package.json con node ejecutando:

*npm install*

Y compilamos el tema con el siguiente comando:

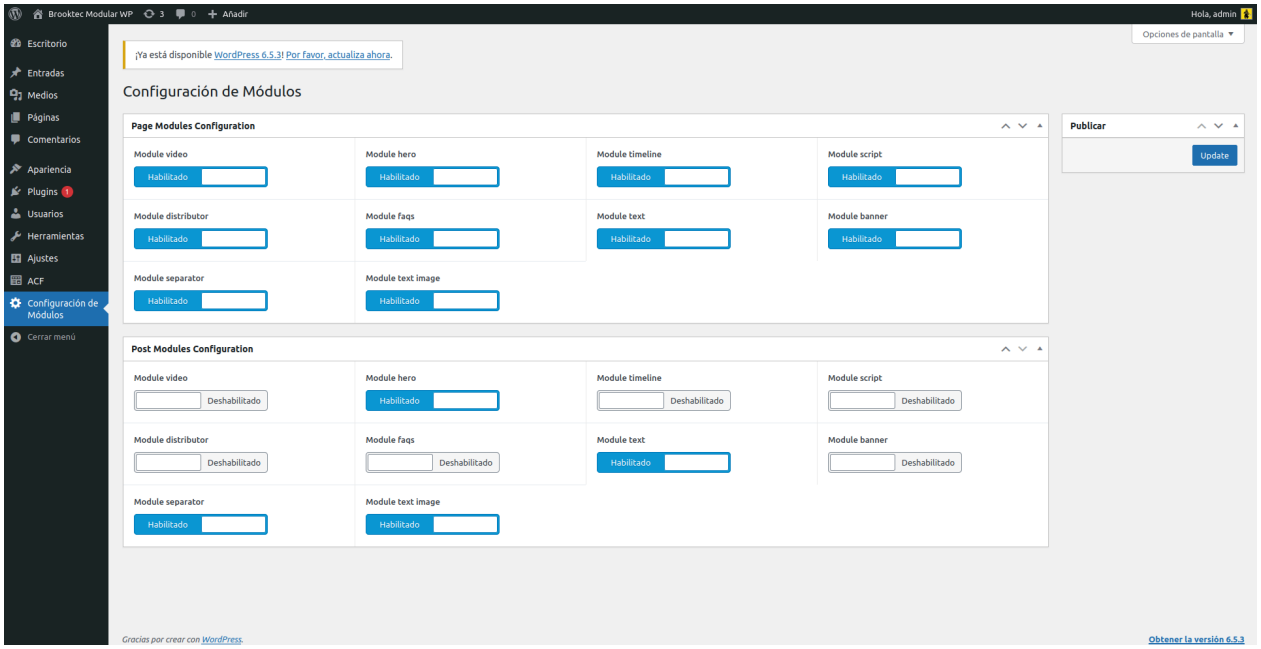
*npx gulp build*

Llegado a este punto, tenemos todo instalado y listo para empezar a trabajar con ello.

# Habilitar y deshabilitar módulos

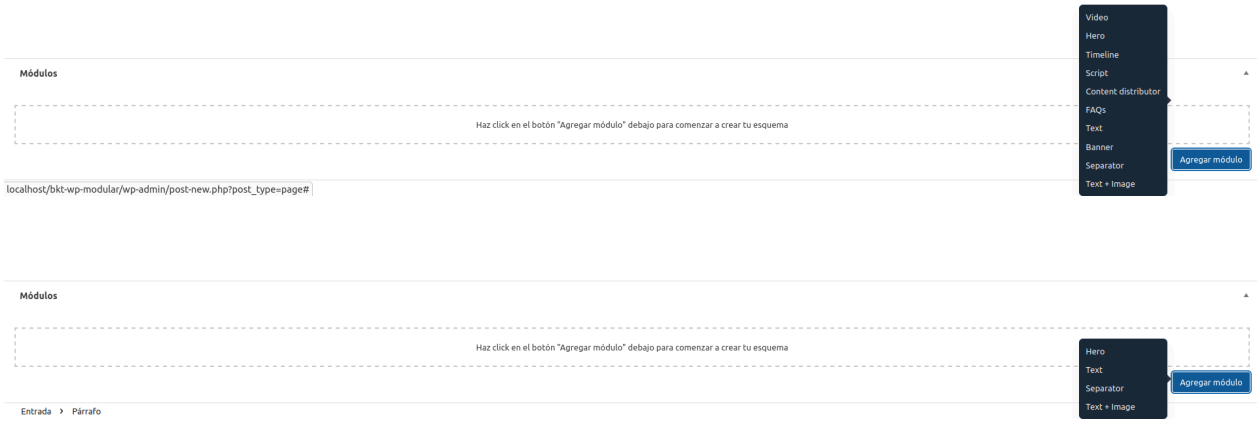
Para la construcción de un site, podemos tener requerimientos de que ciertos módulos estén disponibles o no disponibles según el tipo de contenido. Por ejemplo: podríamos necesitar que en las páginas todos los módulos estuvieran habilitados para su uso pero en las entradas de *posts* solo se pudiera usar el módulo cabecera (hero), el módulo texto, el módulo imagen, el módulo texto e imagen, y el separador de módulos.

Para fijar esta configuración de módulos, iremos dentro del *dashboard* de WordPress en el menú al apartado Configuración de Módulos. Aquí aparecerán los desplegables de Page Modules Configuration y de Post Modules Configuration. En cada desplegable habilitaremos y deshabilitamos los que necesitemos y haremos click en el botón Update.

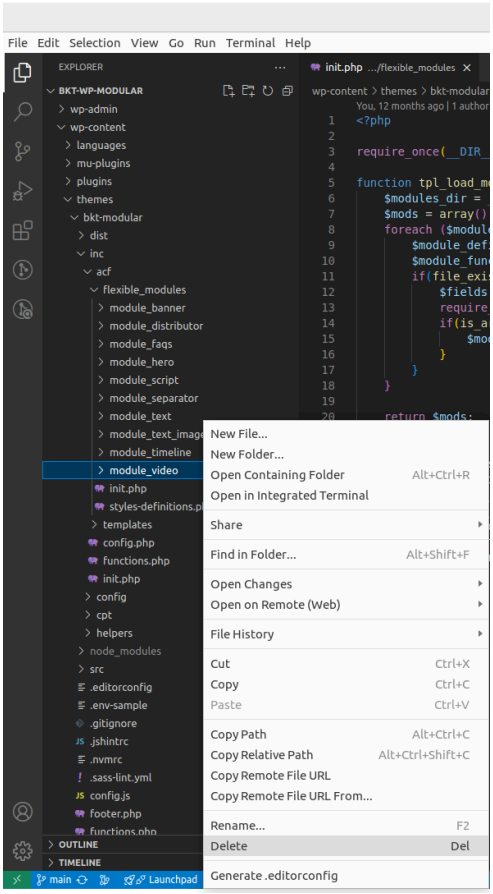



En caso de haber creado un nuevo tipo de contenido (custom post type) aparecerá en esta sección automáticamente para habilitar y deshabilitar sus respectivos módulos permitidos como otra sección desplegable más además de Page y Post. Más adelante, en este manual, se explica cómo crear un nuevo tipo de contenido.

En el ejemplo expuesto, si navegamos a la sección de páginas y creamos una nueva página veremos disponibles todos los módulos. Sin embargo si vamos a la sección entradas y creamos una entrada (post) nueva veremos que tenemos disponibles solo los módulos habilitados en la sección de configuración.




Si se prevé que la página en desarrollo no va a hacer nunca uso de ciertos módulos en concreto que el tema trae por defecto, se recomienda el borrado directo en código de dichos módulos para que, una vez puesta la web en producción, el cliente con su usuario administrador de WordPress, no habilite dichos módulos en alguno de los contenidos. Para ello, navegaremos en la estructura de archivos hasta la ruta *wp-content/themes/bkt-modular/inc/acf/flexible_modules* y borraremos la carpeta del módulo que deseamos eliminar por completo. De esta forma desaparecerá automáticamente de la pantalla de Configuración de Módulos.



# Crear un nuevo tipo de contenido (Custom Post Type) enganchado al sistema modular

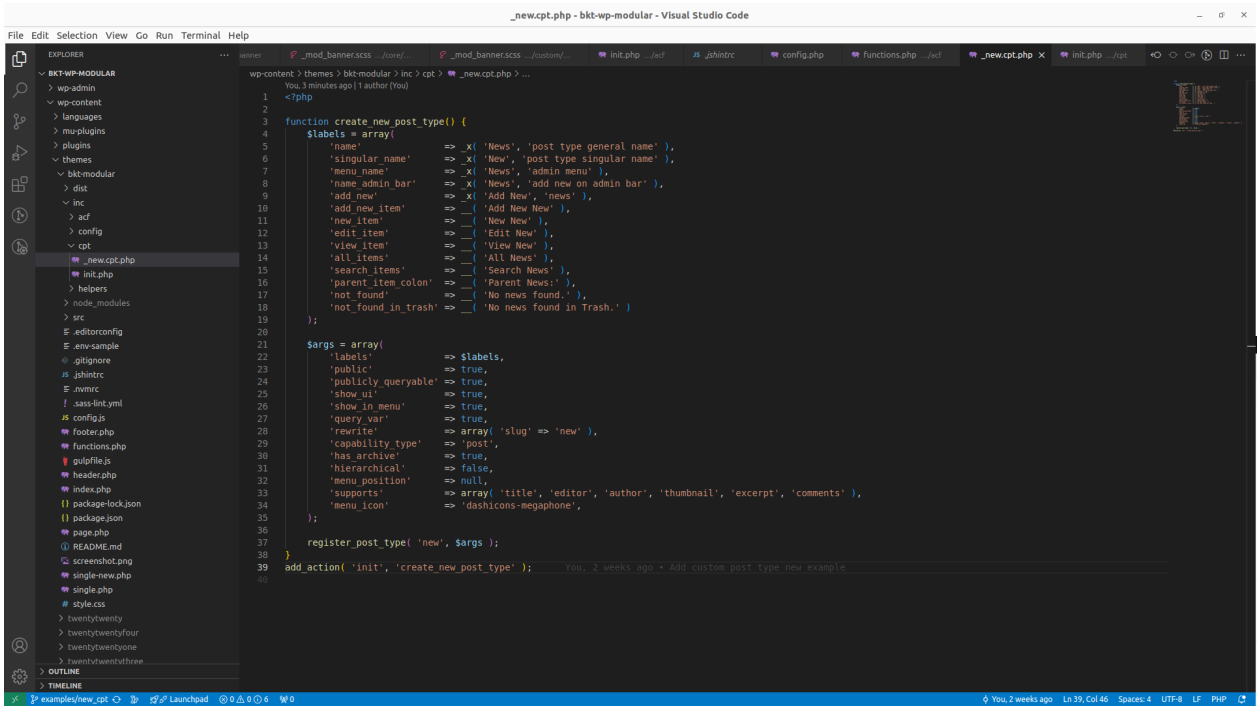
En ocasiones nos es indispensable crear nuevos tipos de contenidos diferenciados de las entradas ordinarias de WordPress. Por ejemplo, una página web de un cliente podría necesitar de un tipo de contenido “Noticia”. Brooktec WP Modular trae un mecanismo para que se añada automáticamente el sistema modular desarrollado. Lo único que habrá que hacer es registrar el post type con la función de wordpress correspondiente en un archivo cuyo nombre sea una barra baja delante del nombre del post, seguido de un punto, seguido de cpt, seguido de la extensión php:

**_nombre-del-post-type.cpt.php**

dentro de la ruta:

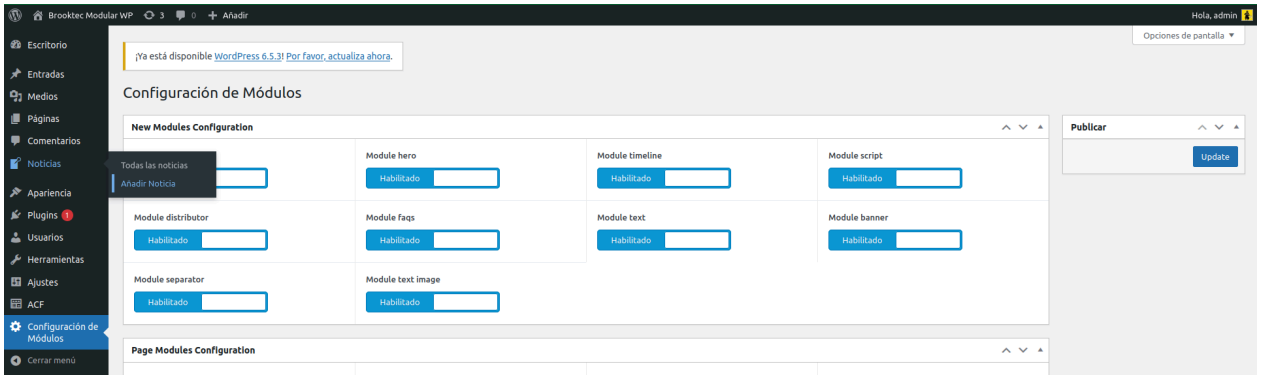
*wp-content/themes/bkt-modular/inc/cpt*

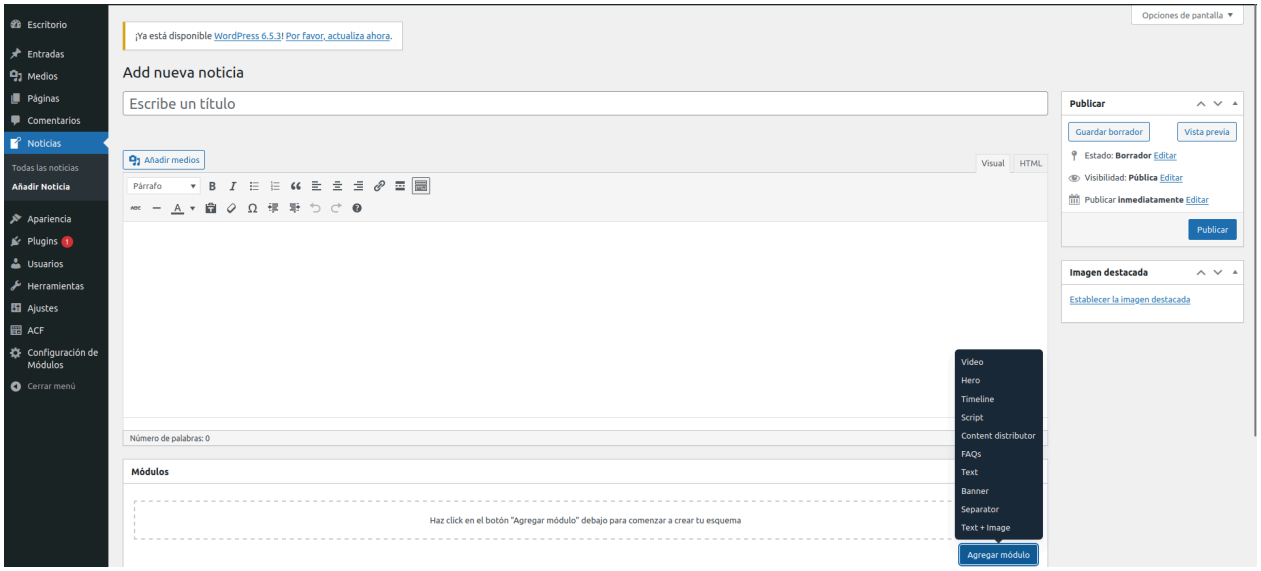
Una vez hecho esto el post type declarado tendrá completamente integrado el sistema modular con el sistema de habilitación de módulos incluido.




Para el ejemplo expuesto en este manual llamaríamos al archivo *_new.cpt.php*, y en él definiríamos cómo se registra el nuevo post type. No es el objetivo de este manual enseñar cómo se registra un custom post type en WordPress, pero si se necesita más información comprobar la documentación oficial en el siguiente enlace:

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




Por defecto, todos los módulos estarán habilitados para un nuevo custom post type, para establecer qué módulos deben o no deben estar habilitados para dicho CPT iremos a la sección del menú Configuración de Módulos, buscaremos el desplegable del nuevo CPT y lo configuraremos a nuestro gusto como hemos explicado en la sección anterior.

# Personalizar estilos de un módulo

Este sistema modular viene implementado con unos estilos básicos que solo sirven para que dichos módulos se rendericen adecuadamente tanto en escritorio como en dispositivos móviles y se adecuen a la funcionalidad que representan. Sin embargo, la implementación de estilos no va más allá de eso. No obstante, el tema incluye una serie de archivos SASS preparados para la personalización de estilos de cada módulo.

Dentro de la carpeta *src* (source) encontramos la carpeta *sass* en la ruta *wp-content/themes/bkt-modular/src/sass* donde están todos los estilos del tema. En dicha carpeta

hay otras dos carpetas; core y custom. La carpeta *core* no debe modificarse, es en la carpeta *custom* donde debemos implementar los estilos del módulo que deseemos estilizar.

Si vemos el contenido de la carpeta custom veremos que hay una carpeta llamada modules. En dicha carpeta habrá un archivo por cada uno de los módulos existentes, más uno a mayores llamado *_common.scss* que contendrá estilos comunes a todos los módulos. Iremos al archivo del módulo que queremos personalizar y nos encontraremos siempre la siguiente estructura de código:

```
.mod-nombre-del-módulo {
   @extend %mod-nombre-del-módulo
}
```

Esta estructura no la debemos modificar pues es la encargada de heredar los estilos básicos implementados para dicho módulo en el archivo padre *wp-content/themes/bkt-modular/src/sass/core/modules/_mod_nombre-del-módulo.scss.* En el archivo hijo que estamos modificando a partir de la línea 2, y siempre dentro del selector de clase, podremos agregar o sobrescribir los estilos que queramos de dicho módulo. Por ejemplo, queremos darle un margen inferior a la cabecera de título del módulo vídeo:

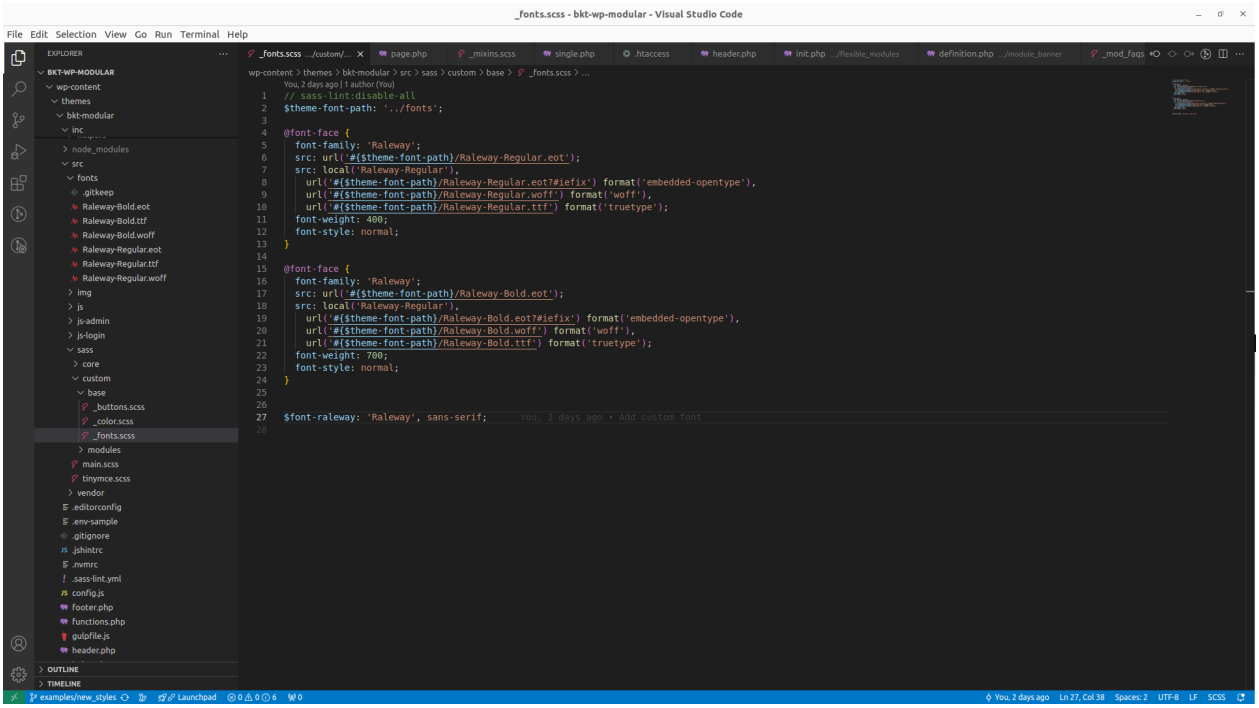
```
.mod-video {
 @extend %mod-video;

 .header {
   margin-bottom: 2.5rem;
 }
}
```

De esta forma heredará todos los estilos que hayan definidos por defecto en el archivo básico de estilos de dicho módulo y añadirá o sobrescribirá la regla para la clase header de margin-bottom: 2.5rem.

# Añadir fuentes personalizadas

El sistema permite añadir fuentes personalizadas para el tema que se va a desarrollar al cliente. Simplemente se debe disponer de los archivos en formato ttf (truetype), eof (embedded-opentype) y woff y realizar los siguientes pasos:

1. Añadir los archivos de fuentes en los formatos indicados a la carpeta src/fonts.
2. En el archivo sass de fuentes (wp-content/bkt-modular/src/sass/custom/base/_fonts.scss) definir el font-face con los nombres y rutas a los archivos del compilado.
3. Asignar en una variable sass el nombre de la fuente especificada en el font-face. Dicha variable será la que luego usemos en el resto de archivos de estilos para indicar que queremos ese font-family.




Al hacer la compilación el tema se encarga de generar los archivos de fuentes en las rutas de fuentes de la carpeta dist (distribution).

# Añadir colores

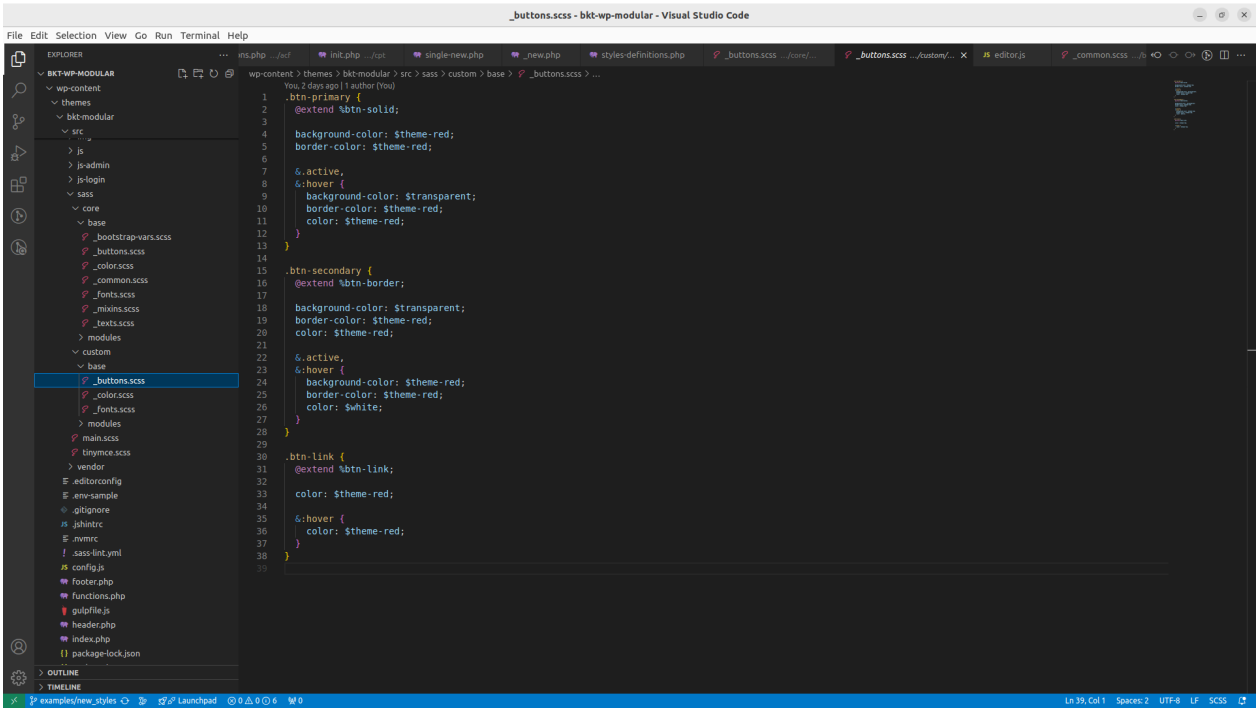
De la misma forma que hemos podido necesitar añadir fuentes personalizadas, también necesitaremos posiblemente añadir definición de colores para por ejemplo estilizar el site con los colores corporativos del cliente. Existe una serie de definiciones de colores básicas como las variables de colores blanco, negro, transparente o negro con opacidad del 50% en el archivo de estilos básico correspondiente. Sin embargo, podemos añadir nuestra definición de colores custom a mayores en el archivo *wp-content/themes/bkt-modular/src/sass/custom/base/_fonts.scss*. En dicho archivo se pueden añadir colores, pero nunca redefinirlos, si intentamos redefinir la variable $white, dará un error al compilar ya que la variable $white ya fue definida en el archivo de colores básicos.

# Personalizar estilos de botones

Los botones se han componentizado para poderse aplicar por igual a cualquier módulo sin tener que definir sus estilos en todos y cada uno de ellos. De forma que si se quisiera personalizar los estilos de los botones con los colores corporativos o los diferentes efectos deseados se puede hacer en el archivo *wp-content/themes/bkt-modular/src/sass/custom/base/_buttons.scss*.

Los botones existentes tienen los nombres de clase btn-primary, btn-secondary, btn-link y btn-rounded. Pudiendo esta última clase combinar con las dos primeras para crear diferentes botones redondeados. El código que nos encontraremos será similar al ya explicado en este manual en la sección de personalizar estilos de un módulo. Habrá una definición inicial de las tres clases que heredarán estilos de sus archivos base. De esta forma todo lo que definamos a partir de las líneas @extend %btn-solid o @extend %btn-border o @extend %btn-link, añadirá nuevas reglas o sobrescribirá las ya existentes mediante herencia.

```
.btn-primary {
  @extend %btn-solid;
}

.btn-secondary {
  @extend %btn-border;
}

.btn-link {
  @extend %btn-link;
}
```




# Añadir efectos javascript a un módulo

Si necesitáramos añadir por cualquier motivo javascript a un módulo existente debemos modificar un archivo en la carpeta 'src', anteriormente mencionada en los apartados de personalización de estilos, y dentro de la carpeta 'js' llamado 'modules.js'. En él está el javascript asociado a la reproducción del módulo video. Sin embargo, se pueden añadir más funcionalidades para otros módulos siempre siguiendo las siguientes directrices:

1. El código asociado a dicho módulo debe ir dentro de la función *jQuery(function() {});* para que cargue cuando el documento esté listo.
2. El código asociado a dicho módulo debe ir dentro de la siguiente función:

   *$('.mod-nombre-del-módulo').each(function() {var module = $(this); /*Código*/ });*

   De esta forma, si aparecen dos o más módulos del mismo tipo dentro de una misma página o post se ejecutará la función en ambos módulos y no solo en el primero que encuentre.

# Compilación de estilos, fuentes y javascript

El sistema ya está preparado para compilar mediante Node los estilos, fuentes y javascript del proyecto y minificarlos para que ocupen lo mínimo posible en un solo archivo en el caso de los archivos SASS y Javascript. El código fuente del SASS se transformará en CSS en el proceso de compilación, por lo que en la carpeta *dist* se encontrará bajo la carpeta *css*.

El proceso es sencillo, estando desde terminal en la carpeta *wp-content/themes/bkt-modular* y habiendo hecho previamente el oportuno cambio de versión de node descrito en el proceso de instalación (nvm use) utilizamos el siguiente comando:

*npx gulp build*

Esto ejecutará la compilación de estilos, fuentes y javascript. Aunque si solo se quisiera compilar alguna de las tres por separado se pueden ejecutar los siguientes comandos respectivamente:

*npx gulp build:sass*
*npx gulp build:fonts*
*npx gulp build:javascript*

# Modificar el comportamiento de un módulo

Si uno de los módulos necesitase un campo extra, o modificar la disposición de sus elementos por HTML o cualquier otra modificación en la que no se pueda hacer exclusivamente por CSS o Javascript, entonces no quedará más remedio que modificar el PHP del módulo. Para ello, el usuario debe saber que el código de los módulos se encuentra en *wp-content/themes/bkt-modular/inc/acf/flexible_modules* y que todo módulo está siempre compuesto de dos archivos:

- **definition.php:** en él se detalla la definición de campos ACF que posee el módulo.
- **functions.php:** en él se definirá las funciones necesarias para el renderizado del módulo. Siempre existirá al menos una función llamada *bkt_flex_module_nombre-del-módulo_content* y que contendrá el código específico de la renderización de ese módulo

# Creación de un módulo

Si quisiéramos crear un módulo nuevo es tan sencillo como crear una nueva carpeta dentro de la ruta *wp-content/themes/bkt-modular/inc/acf/flexible_modules* y llamarla module_nombre-del-módulo. Dentro de esa carpeta habrá un archivo definitions.php y un archivo functions.php como se ha descrito en la sección anterior donde definiremos los campos que posee dicho módulo y una función de renderización.

Una vez hecho esto el módulo se habrá añadido automáticamente al listado de módulos disponibles en el sistema, pudiendolo habilitar o deshabilitar en los diferentes tipos de contenido y usarlo en ellos en función de lo configurado en la sección de configuración de módulos.

# Subir un nuevo módulo al repositorio

Esta acción solo se llevará a cabo en el caso de que no estemos trabajando directamente en el desarrollo de una página de cliente, sino que el propósito del nuevo módulo desarrollado es añadirlo a la colección de módulos existentes en el propio sistema.

Para ello, naceremos de la rama 'main' y crearemos una rama feature/module_nombre-del-nuevo_módulo, en dicha rama desarrollaremos el módulo. Para hacer la subida al repositorio primero se mergeará con la rama 'develop' pudiendo hacerlo mediante una solicitud de Pull Request para su revisión.

Una vez probado el nuevo módulo en 'develop' y, si el equipo ha dado el visto bueno para su integración como nuevo módulo, se podrá mergear la rama en 'main' para pasar a estar a disposición de los nuevos desarrollos de *sites* que se hagan con este sistema.

# Bibliografía y referencias


[1] WordPress. (2024). *Herramienta de blog, plataforma de publicación y CMS – WordPress.org España.* WordPress.org Obtenido el 15 de mayo del 2024 de https://es.wordpress.org/

[2] Drupal. (2024). *Drupal - Open Source CMS.* Drupal.org Obtenido el 15 de mayo del 2024 de https://www.drupal.org/

[3] Moodle. (2024). *Moodle.org.* moodle.org Obtenido el 15 de mayo del 2024 de https://moodle.org/

[4] Joomla. (2024). *Joomla Content Management System (CMS) - try it! It's free!.* Joomla. Obtenido el 15 de mayo del 2024 de https://www.joomla.org/

[5] W3Techs. (2024). *Historical yearly trends in the usage statistics of content management systems, June 2024*. W3Techs. Obtenido el 9 de junio del 2024 de https://w3techs.com/technologies/history_overview/content_management/all/y

[6] Deyimar A. (2024). *Gutenberg WordPress: una guía completa del editor de bloques, May 2024*. Hostinger. Obtenido el 16 de mayo del 2024 de https://www.hostinger.es/tutoriales/gutenberg-wordpress

[7] Advanced Custom Field Team. (2024). *ACF | Resources, Documentation, API, How to & Tutorial Articles, March 2024*. advancedcustomfields.com. Obtenido el 5 de marzo del 2024 de https://www.advancedcustomfields.com/resources/#field-types

[8] Apache Friends. (2024). *XAMPP Installers and Downloads for Apache Friends, June 2024*. apachefriends.org. Obtenido el 2 de junio del 2024 de https://www.apachefriends.org

[9] Xosoft, LLC DBA GitKraken. (2024). *GitKraken Legendary Git Tools | GitKraken, June 2024*. gitkraken.com. Obtenido el 9 de junio del 2024 de https://www.gitkraken.com/

[10] NPM. (2024). *npm About, June 2024*. npmjs.com. Obtenido el 9 de junio del 2024 de https://www.npmjs.com/about

[11] NVM. (2024). *Desarrollo web, June 2024*. desarrolloweb.com. Obtenido el 9 de junio del 2024 de https://desarrolloweb.com/home/nvm

[12] Altassian. (2024). *Flujo de trabajo de Gitflow | Atlassian Git Tutorial, June 2024.* atlassian.com. Obtenido el 9 de junio del 2024 de https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow

[13] Altassian. (2024). *Qué es scrum y cómo empezar, June 2024.* atlassian.com. Obtenido el 1 de junio del 2024 de https://www.atlassian.com/es/agile/scrum

[14] Shore Labs. (2024). *Metodología Kanban | Kanban Tool, June 2024.* kanbantool.com. Obtenido el 1 de junio del 2024 de https://kanbantool.com/es/metodologia-kanban

[15] Elena Bello. (2021). *Descubre qué es el Extreme Programming y sus características, April 2021.* atlassian.com. Obtenido el 1 de junio del 2024 de https://www.iebschool.com/blog/que-es-el-xp-programming-agile-scrum/

[16] Rachaelle Lynn. (2024). *What is FDD in Agile? | Planview, June 1 2024.* planview.com. Obtenido el 1 de junio del 2024 de https://www.planview.com/resources/articles/fdd-agile/

[17] Web Empresa. (2024). *Cómo instalar WordPress - Tutorial en español (2024), June 1 2024.* webempresa.com. Obtenido el 1 de junio del 2024 de

https://www.webempresa.com/wordpress/como-instalar-wordpress-tutorial-de-instalacion-en-espanol.html

[18] Raelene Morey. (2018). *How to Set the Default Theme for WordPress Multisite - WPMU DEV, June 5 2018.* wpmudev.com. Obtenido el 1 de junio del 2024 de https://wpmudev.com/blog/how-to-set-the-default-theme-for-wordpress-multisite/