Desarrollo completo de un sitio web con buenas prácticas

MEMORIA PRESENTADA POR:

Javier Hernandez Sanz

GRADO EN INGENIERÍA INFORMÁTICA

CENTRO

Escuela Politécnica Superior de Alcoy

TITULACIÓN

Grado en Ingeniería Informática

Titulo

Desarrollo completo de un sitio web con buenas prácticas.

Autor

Javier Hernandez Sanz

Tutor

Josep Tomàs Alemany i Mollà

Cotutor

Departamento

Resumen

El TFG consta del desarrollo de una página web online para una empresa real, el sitio será informativo sobre la empresa, sus productos y características.

El proyecto parte de un diseño inicial en Photoshop aceptado ya por el cliente. En este diseño tenemos especificado todos los parámetros de la web (Anchura máxima, altura de cada caja, imágenes, etc.).
En primer lugar, vamos a crear nuestra máquina virtual (Vagrant) para realizar en local el desarrollo de esta web y la creación de nuestro repositorio online, este repositorio nos permitirá cuando publiquemos la web tener un control de versiones y cualquier miembro del equipo tendrá acceso.
El segundo paso nuestro será valorar que motor utilizamos para el desarrollo de esta web, en este caso como no es un e-commerce utilizaremos "WORDPRESS". Por lo que partimos de una plantilla base constructora y la creación de child theme para evitar problemas en futuras actualizaciones. Este desarrollo esta realizado con: php, javascript, html, sass, Bootstrap, Grid y flexbox. También incluye responsive, por lo que se podrá visualizar en cualquier dispositivo ya que se adapta a la resolución.
Para el desarrollo de la web necesitaremos programar nuevos tipos de objetos y modulos en nuestro wordpress para facilitar la autogestión del cliente final.
Una vez finalizado el desarrollo de la web, la subiremos a un servidor dev (online) para su testeo, una vez este testeada y finalizada, la subiremos al servidor final y se publicará. En este servidor final veremos la configuración realizada para la creación de la suscripción del cliente y daremos de alta los correos electrónicos que nos solicitan.

Resum

El TFG consta del desenvolupament d'una pàgina web online per a una empresa real, el lloc serà informatiu sobre l'empresa, els seus productes i característiques.
El projecte parteix d'un disseny inicial en Photoshop acceptat ja pel client. En aquest disseny tenim especificat tots els paràmetres de la web (Amplària màxima, altura de cada caixa, imatges, etc.).
En primer lloc, anem a crear la nostra màquina virtual (Vagrant) per a realitzar en local el desenvolupament d'aquesta web i la creació del nostre repositori online, aquest repositori ens permetrà quan publiquem la web tenir un control de versions i qualsevol membre de l'equip tindrà accés.
El segon pas nostre serà valorar que motor utilitzem per al desenvolupament d'aquesta web, en aquest cas com no és un e-commerce utilitzarem "WORDPRESS". Pel que partim d'una plantilla base constructora i la creació de child theme per a evitar problemes en futures actualitzacions. Aquest desenvolupament està realitzat amb: php, javascript, html, sass, Bootstrap, Grid i flexbox. També inclou responsive, per la qual cosa es podrà visualitzar en qualsevol dispositiu ja que s'adapta a la resolució.
Per al desenvolupament de la web necessitarem programar nous tipus d'objectes i modulos en el nostre wordpress per a facilitar l'autogestió del client final.
Una vegada finalitzat el desenvolupament de la web, la pujarem a un servidor dev (online) per al seu testeo, una vegada testeada i finalitzada, la pujarem al servidor final i es publicarà. En aquest servidor final veurem la configuració realitzada per a la creació de la subscripció

Abstract

The aim of the Final Grade Work (FGW) consists of the development of an online website for a real company. The purpose of the website is to communicate the benefit of their products.
The design of the project created in Photoshop was already accepted by the customer. In the design we specified all the parameters of the web (maximum width, the height of each box, images, etc.).
The first step of the project is to create our virtual machine (Vagant) to start the development of this website. It is also useful for the creation of our online repository, which will allow us to have a version control for any member of the team when the website is published.

The second step will be to judge what engine we use for the development of this website. In this case, as it is not an e-commerce, we will use "WORDPRESS". We start with a base template construction and the creation of child theme to avoid problems in future updates. This development is made with: php, javascript, html, sass, Bootstrap, Grid and flexbox. It also includes responsive, so it can be viewed on any device.
For the web construction, we will need to develop new types of objects and modules in our WordPress to help the self-management of the final client.
Once the development of the web is finished, we will upload it to a dev server (online) for testing. After the testing phase, we will upload it to the final server and the website will be published. Finally, on the final server, we will establish the last

Índice

Ilustraciones

Palabras clave o abreviaturas

- **Site:** página web.
- **CMS:** Un sistema de gestión de contenidos es un programa informático que permite crear una estructura de soporte para la creación y administración de contenidos, principalmente en páginas web.
- **CPT**: custom post type, tipo de entrada personalizada.
- **CMDER:** Terminal.
- **Theme:** Tema o plantilla.
- **Child theme:** Tema hijo.
- **Dashboard:** Panel de Wordpress.
- **Divi:** Nombre del tema padre.
- **SEO:** posicionamiento web en los resultados de los buscadores.
- **Minificación de archivos:** Juntar 5 archivos en uno (del mismo tipo) y comprimirlos para que el proceso de datos sea más rápido y efectivo.
- **Home:** Página principal de nuestro sitio web.
- **Header:** Cabecera de una página web (Se repite en todas las páginas)
- **Footer:** Pie de una página web (Se repite en todas las páginas).
- **Metalenguaje:** un metalenguaje es un lenguaje que se usa para hablar acerca de otro lenguaje.
- **Post type:** Tipo de entrada (productos, páginas, post, etc).
- **Carousel:** Listado de elementos que muestra un máximo definido y los demás los oculta, los elementos van apareciendo y ocultándose de forma vertical o horizontal.
- **widget:** es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de widgets o Widget Engine.
- **media queries:** Elemento css que permite adaptarse a la resolución de todas las pantallas.

1. Introducción

En el presente documento se explica el proceso de análisis y desarrollo de un sitio web para una empresa final. En el proyecto se sigue la metodología establecida en la empresa "acceseo" para realizar el desarrollo con buenas prácticas. Se expondrán con detalle todos los pasos realizados durante el proceso para conseguir los objetivos de la forma más correcta y óptima para que el cliente final pueda gestionar este sitio autónomamente en un futuro.

1.1 Restricciones y confidencialidad

Antes de realizar este documento, las dos partes acordamos NO incluir una serie de datos personales por confidencialidad y privacidad. Los datos acordados son los siguientes:

- Código desarrollado a medida.
- Diseño a medida en formato (.psd).
- Contenido Sass y css.
- Contraseñas
- Direcciones ip
- Configuraciones internas

Por lo tanto, el alumno se compromete a no difundir ningún dato de los nombrados en este punto, ya que son datos confidenciales y protegidos.

1.2 Propósitos y objetivos

El propósito de este proyecto es el desarrollo de un sitio web para una empresa final. Este sitio permitirá explotar más mercado con el Marketing Online.

Basándonos en un diseño inicial, hay que ajustarse con el máximo detalle al diseño y facilitar al cliente final (sin conocimientos) la gestión de la manera más fácil y eficiente. El diseño solo está implementado en resolución máxima (versión PC 1980px de ancho) y debe visualizarse correctamente en cualquier dispositivo, por lo que es necesario adaptarlo a resoluciones inferiores (mínima 320px de ancho).

En dicho diseño hay puntos que deberemos desarrollar para que el sitio web cumpla con las características necesarias.

1.3 Alcance

El alcance de este proyecto irá desde la recepción de un feedback con un diseño por parte del área de Gestión de proyectos. En este feedback nos dirán todos los elementos que tiene que tener esta web y cómo deben ser, hasta que este sitio esté publicado y funcionando en un servidor final.

2. Máquina virtual.

Cada vez que necesitamos desarrollar algo tenemos que hacer uso de los entornos controlados para poder hacer todas las pruebas necesarias sin comprometer el estado de producción.

Para empezar el desarrollo de nuestro proyecto, utilizaremos una máquina virtual en local. Al trabajar en local nos evitaremos el paso de comunicación entre Ordenador - Servidor ya que el servidor estará corriendo en nuestro ordenador. A este servidor únicamente se podrá acceder desde el mismo equipo en el que se inicia o en la misma red con los permisos adecuados.

Esto nos va a permitir operar del siguiente modo.

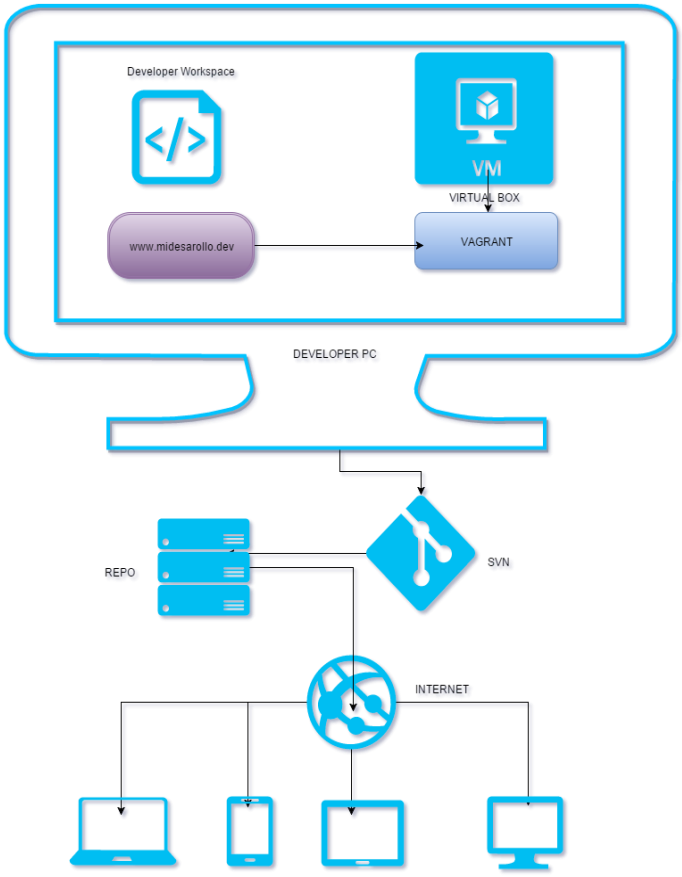

ILUSTRACIÓN 1 - FUNCIONAMIENTO MÁQUINA VIRTUAL - SERVIDOR FINAL

2.1 Vagrant

Vagrant es una herramienta para la creación y configuración de entornos de desarrollo virtualizados, disponible para todos los sistemas operativos (Windows, MacOS y GNU/Linux). Esta herramienta permite generar entornos de desarrollo reproducibles y compartibles de forma muy sencilla. Para ello, Vagrant crea y configura máquinas virtuales a partir de simples ficheros de configuración, basta con compartir el fichero de configuración de Vagrant (llamado "Vagrantfile"), con lo que todos los desarrolladores de un equipo pueden compartir su máquina virtual y entorno de desarrollo. Esto es especialmente útil en equipos formados por varias personas, asegura que todos los desarrolladores tienen el mismo entorno, con las mismas dependencias y misma configuración. Con Vagrant, compartir pesadas máquinas virtuales o el ya mítico "en mi ordenador funciona" (causado generalmente por diferentes configuraciones o versiones de software) son cosas del pasado. Además, cuando un miembro del personal añade una funcionalidad nueva al archivo, todos podremos utilizarla sin necesidad de añadir una a una.

Además, dado que la configuración de la máquina virtual es un simple fichero de texto plano, podremos incluir este fichero en nuestro repositorio en el control de versiones, junto con el resto del código del proyecto. De esta manera, un nuevo desarrollador que se incorpore al equipo, simplemente tendrá que clonar el repositorio del proyecto y ejecutar Vagrant para tener el entorno de desarrollo montado y funcionando en cuestión de minutos.

Vagrant utiliza VirtualBox como motor de máquinas virtualBox por defecto. (Pueden utilizar otros motores).

2.2 Línea de comandos. Cmder.

Vagrant es una herramienta gratuita de línea de comandos. En nuestro caso al trabajar con windows utilizaremos el software "Cmder". Cmder es un paquete creado por la frustración que genera la consola de windows. Con este terminal podremos utilizar los comandos de Linux en windows además de muchas más funcionalidades añadidas. Esta herramienta está realizada por la comunidad de programadores de GitHub, donde desde la misma web podremos obtener todos los repositorios. Tiene su versión portable.

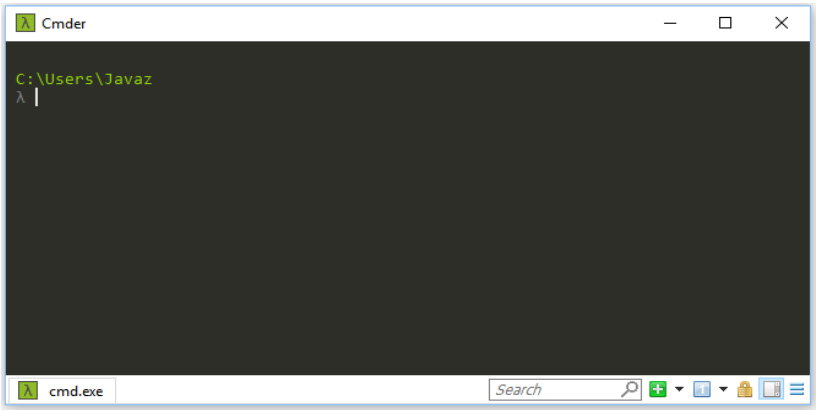

ILUSTRACIÓN 2 - SOFTWARE CMDER

2.3 Instalación de vagrant

El software necesario dependerá mucho de las necesidades particulares de cada uno, pero para una configuración básica podemos usar los siguientes programas.

- Virtual box
- Vagrant
- Navegador web de nuestra preferencia (Chrome).
- Editor de texto (Sublime Text).

El software Vagrant se compone principalmente de las siguientes 3 partes:

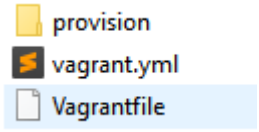

ILUSTRACIÓN 3 - COMPOSICIÓN VAGRANT

- **Provision:** Carpeta contenedora de los archivos .sh donde incluye las herramientas o paquetes que necesitamos en nuestra máquina virtual para el desarrollo de la página web. (Mysql, php, git, boostrap, apache, etc.).
- **Vagrant.yml:** Definición de nuestra máquina (Sistema operativo, procesadores, ram, nombre y correo para Git, contraseña, etc.).

```
vagrant.yml
1  box: 'bento/ubuntu-16.04'
2  machine:
3      cpus: 1
4      memory: 1024
5      sync_www: true
6
7  git:
8      name: 'Javier Hernandez Sanz'
9      email: 'javi@acceseo.com'
10
11 mysql:
12     root_password: 'root'
13
14 mailhog:
15     version: '1.0.0'
16
17 elasticsearch:
18     version: '5.4.0'
19
20 phpunit:
21     version: '6.1'
```

ILUSTRACIÓN 4 - ARCHIVO VAGRANT.YML

- **Vagrantfile**: Archivo de ejecución de los parámetros, asignación de puertos, control de versiones, etc.

```
require 'yaml'

settings = YAML.load_file 'vagrant.yml'

Vagrant.configure("2") do |config|
    config.vm.box = settings['box']

    # config.vm.network "private_network", ip: '127.0.0.1'

    config.vm.network :forwarded_port, guest: 80, host: 80 #http
    config.vm.network :forwarded_port, guest: 3306, host: 33317 #mysql
    #config.vm.network :forwarded_port, guest: 8025, host: 8025 #mailhog web client
    #config.vm.network :forwarded_port, guest: 9200, host: 9200 #elasticsearch
    #config.vm.network :forwarded_port, guest: 9300, host: 9300 #elasticsearch

    config.vm.provider "virtualbox" do |vb|
        vb.customize ["modifyvm", :id, "--cpus", settings['machine']['cpus']]
        vb.customize ["modifyvm", :id, "--memory", settings['machine']['memory']]
    end

    config.vm.synced_folder "./www", "/var/www", create: true

    config.vm.provision "file", source: "provision/.bash_aliases", destination: "~/.bash_aliases"

    config.vm.provision "shell", path: "provision/common.sh"
    config.vm.provision "shell", path: "provision/git.sh", args: [settings['git']['name'], settings['git']['email']]
    config.vm.provision "shell", path: "provision/apache.sh"
    config.vm.provision "shell", path: "provision/php7.sh"
    config.vm.provision "shell", path: "provision/composer.sh"
    # config.vm.provision "shell", path: "provision/phpunit.sh", args: [settings['phpunit']['version']]
    # config.vm.provision "shell", path: "provision/node.sh"
    # config.vm.provision "shell", path: "provision/bower.sh"
    config.vm.provision "shell", path: "provision/mysql.sh", args: [settings['mysql']['root_password']]
    #config.vm.provision "shell", path: "provision/mailhog.sh", args: [settings['mailhog']['version']]
    # config.vm.provision "shell", path: "provision/elasticsearch.sh", args: [settings['elasticsearch']['version']]
    #config.vm.provision "shell", path: "provision/memcached.sh"
    # config.vm.provision "shell", path: "provision/redis.sh" #todo
    config.vm.provision "shell", path: "provision/cleanup.sh"
end
```

ILUSTRACIÓN 5 - PARAMETROS VAGRANT

Una vez tenemos la configuración lista del vagrant, abrimos cmder, nos desplazamos a la ruta donde queramos levantar la máquina virtual (directorio de nuestro proyecto), añadimos los archivos de vagrant y levantamos la máquina con el comando “vagrant up”.

Una vez finalizado el proceso, veremos que vagrant nos ha generado dos carpetas:

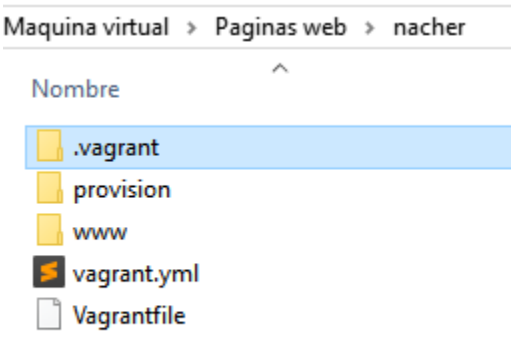

ILUSTRACIÓN 6 - ARCHIVOS MÁQUINA VIRTUAL INSTALADA

En la carpeta ".vagrant" tendremos nuestra máquina virtual y la carpeta "/www/html" será la raíz de nuestro proyecto. Para comprobar que la máquina está levantada y funcionando correctamente vamos a nuestro navegador y entramos a localhost.

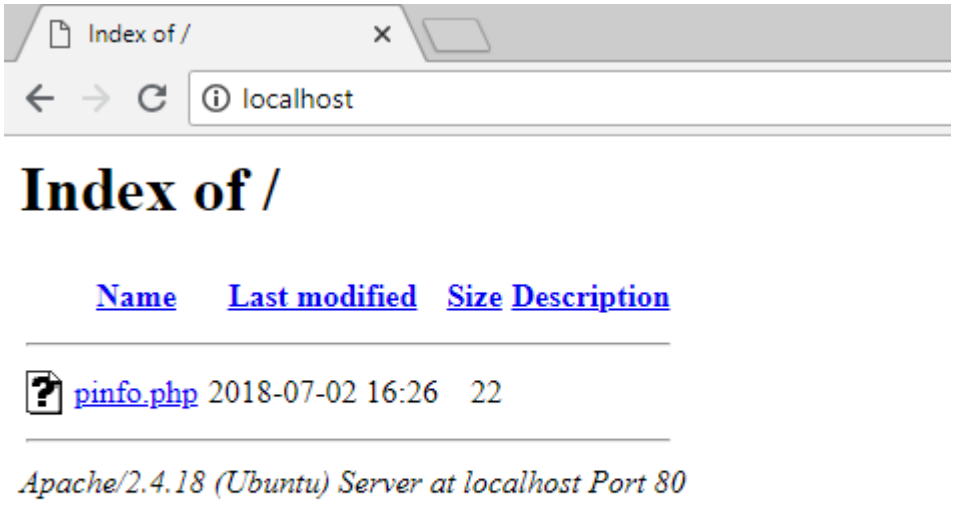

ILUSTRACIÓN 7- COMPROBACIÓN MÁQUINA VIRTUAL FUNCIONANDO

Actualmente dentro de la carpeta "/www/html" está únicamente el archivo pinfo.php, generado con la instalación de la máquina virtual. Por lo que ya tenemos nuestra máquina virtual lista para empezar con el desarrollo de nuestro proyecto.

3. CMS E INSTALACIÓN

La elección de un CMS depende de los requisitos de la empresa que lo solicita, en este caso la empresa solicita una web informativa, donde quiere dar a conocer su empresa y aparte mostrar sus productos (en este caso mesas, sillas, auxiliares y estanterías).

Con esta información ya podemos descartar CMS dedicados al eCommerce (Prestashop, Magento, etc), ya que la finalidad de nuestro cliente no es vender online. En nuestro caso debemos elegir un CMS que a largo plazo siga manteniendo una estabilidad garantizada sea actualizable y, para adaptarnos al diseño personalizado, debe ser lo más modular posible.

Una vez conocemos todos estos factores, debemos tener en cuenta también quien va a gestionar este sitio web una vez desarrollado. No todas las personas tienen la misma experiencia y formación, por lo tanto, debemos sobreentender que nuestro cliente no parte con las nociones necesarias para la gestión de este site y necesitará que le facilitemos las operaciones de gestión a largo plazo, por lo que no debe ser complejo ni dejarle pelearse con código.

Para este proyecto en concreto hemos seleccionado Wordpress por los siguientes motivos:

- La edición de **contenido** es sencilla y facilitará el trabajo al cliente final.
- Es un CMS puntero actualmente.
- Está en constante desarrollo (actualización y crecimiento).
- Es de código abierto, gratuito y tiene un gran soporte.
- Capacidad de multi idioma.
- Gran cantidad de plugins ya desarrollados (algunos de pago y otros no).
- Diseño y apariencia. Con conocimientos avanzados puedes desarrollar un tema a medida.
- Tiene buen rendimiento y potencia el posicionamiento SEO.

3.1 ¿Qué es wordpress?

Wordpress es un gestor de contenido (CMS) gratuito. Este gestor de contenido es idóneo para un sitio web que se actualice periódicamente. Empezó en 2003 siendo una plataforma para la creación de “Blogs” pero con el paso del tiempo ha ido evolucionando hasta convertirse en uno de los mejores CMS que existen en la actualidad. Este CMS es una gran opción cuando el cliente final va a gestionar su sitio, ya que la gestión de éste es para principiantes o gente con pocas nociones técnicas.

Además, este CMS dispone de un amplio catálogo de plugins que permiten ampliar los servicios de nuestro sitio web, por lo tanto hace que Wordpress sea un sistema muy flexible y que se adapte a las necesidades de nuestros clientes.

3.2 Instalación de wordpress

Lo primero que debemos hacer para realizar la instalación de un Wordpress es ir a su página web y descargarnos la última versión http://www.wordpres.org.

Una vez descargado, descomprimimos los archivos en el directorio de nuestra máquina virtual (/www/html/) sustituyendo el archivo php.info.

Una vez hemos añadido los archivos, si entramos a nuestra máquina virtual nos aparece el proceso de instalación de wordpress:

- Elección de idioma principal:

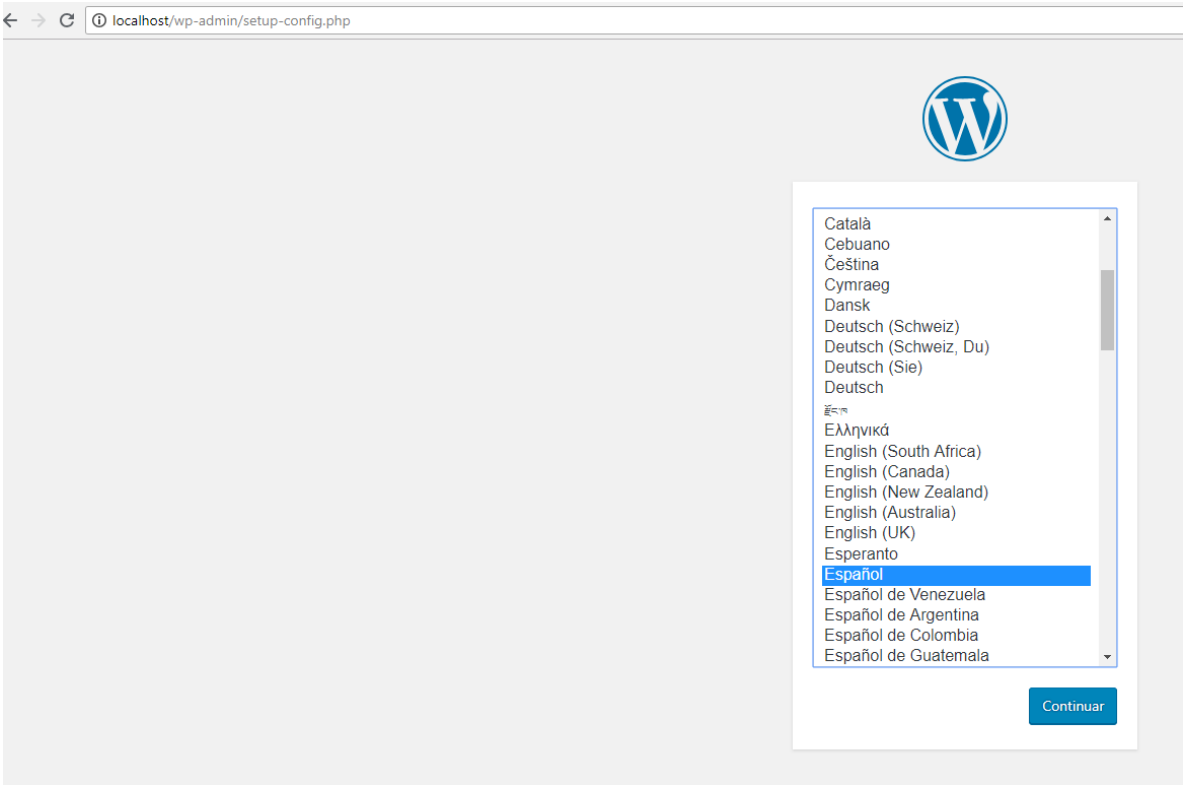

ILUSTRACIÓN 8 - INSTALACIÓN WORDPRESS 1

- Requisitos necesarios:

ILUSTRACIÓN 9 - INSTALACIÓN WORDPRESS 2

Como vemos, el wordpress almacena en base de datos parte de la información (contenido, configuraciones internas, configuración de los plugins, etc). Nuestra máquina virtual viene dotada con Mysql y en el archivo "vagrantfile" (mostrado anteriormente) está la configuración:

```
config.vm.network :forwarded_port, guest: 3306, host: 33317 #mysql
```

ILUSTRACIÓN 10 - PUERTOS MYSQL

En nuestro caso, para acceder a ella necesitaremos:

- **Dirección o ip:** Podemos utilizar "localhost" o lo que es equivalente "127.0.0.1"
- **Puerto:** Definido para los invitados como 3306 (imagen anterior).
- **Usuario y contraseña:** En nuestro caso usuario root y contraseña root.

Para acceder podemos hacerlo desde consola (cmder) o desde un cliente de base de datos (HeidiSQL).

En nuestro caso, lo vamos a gestionar por consola y para la creación de la base de datos realizaremos los siguientes pasos:

1) Accedemos a nuestra máquina virtual por el protocolo SSH

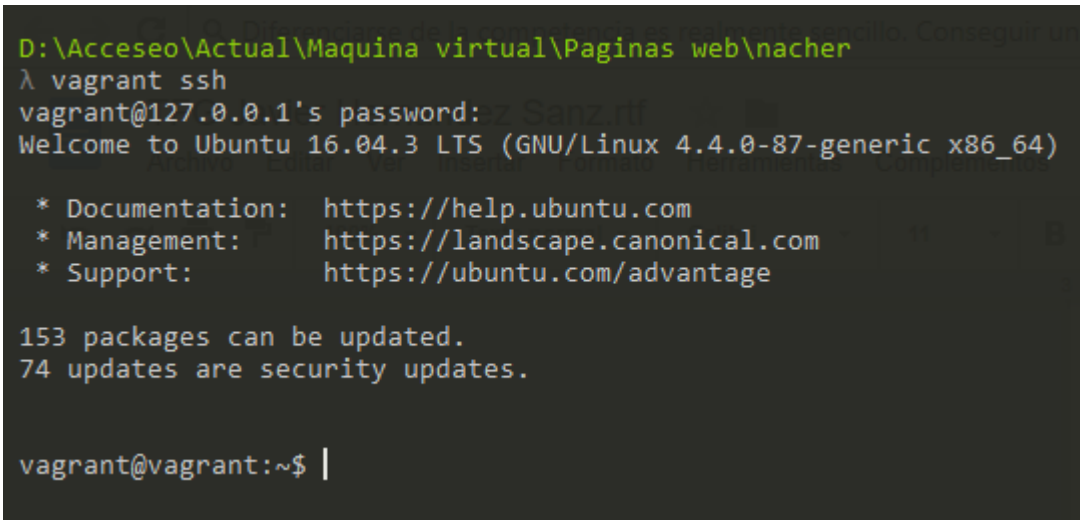

ILUSTRACIÓN 11 - ENTRANDO POR SSH A LA MAQUINA VIRTUAL

2) Iniciamos sesión con el cliente Mysql:

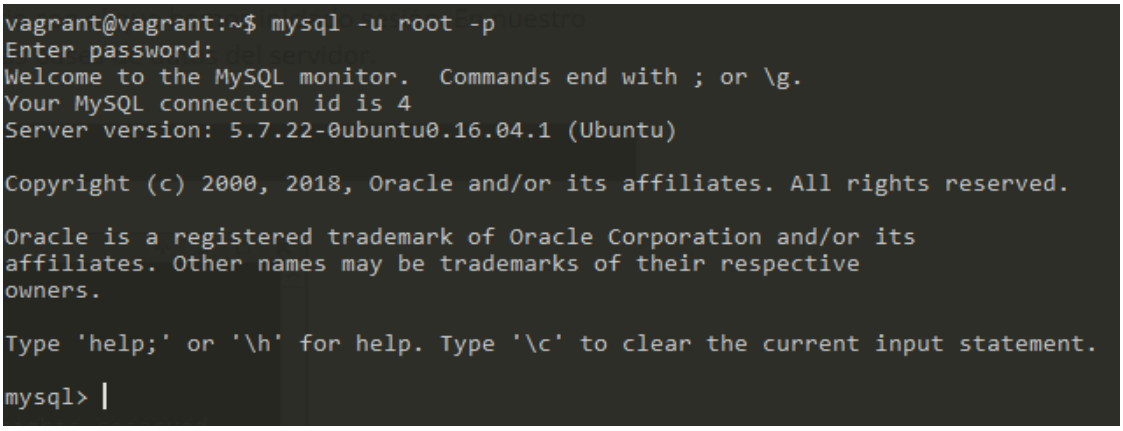

ILUSTRACIÓN 12 - CLIENTE MYSQL POR CONSOLA

3) Ahora creamos la base de datos:

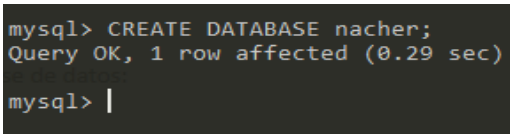

ILUSTRACIÓN 13 - CREACIÓN DE UNA BD

4) Por último, mostramos las bases de datos creadas, para comprobar que aparece correctamente:

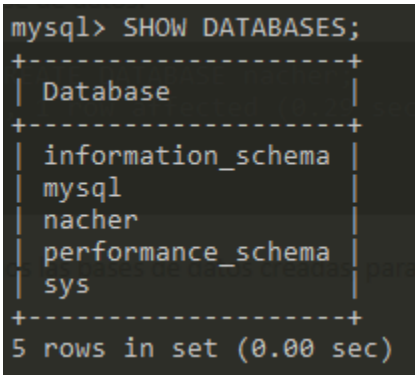

ILUSTRACIÓN 14 - TABLAS CREADAS EN EL SISTEMA

Como podemos ver ya aparece creada correctamente. Por lo que ya podríamos indicarle al instalador de wordpress los datos necesarios para que pueda ejecutar la instalación correctamente.

ILUSTRACIÓN 15 - INSTALACION DE WORDPRESS 3

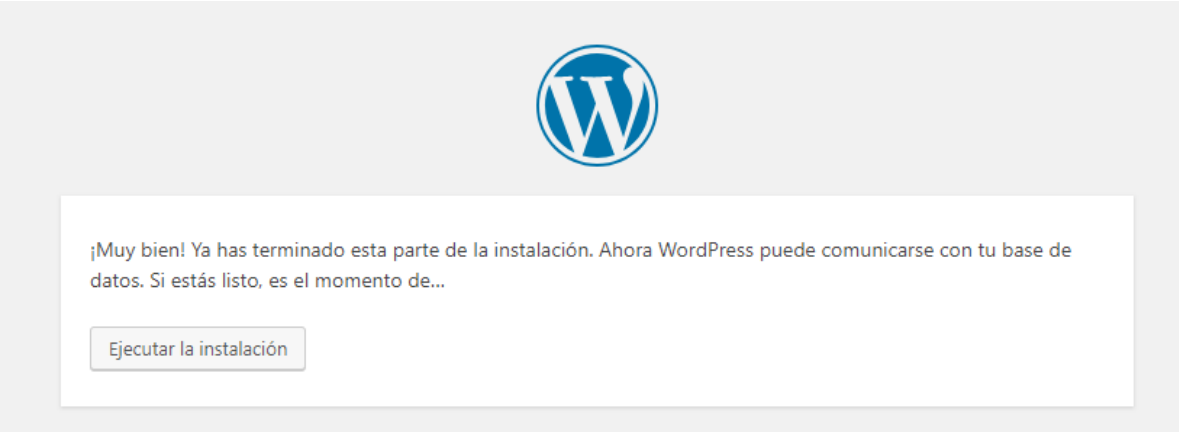

ILUSTRACIÓN 16 - INSTALACIÓN DE WORDPRESS 4

Una vez ejecutamos la instalación, nos piden la información necesaria que necesita el Wodpress:

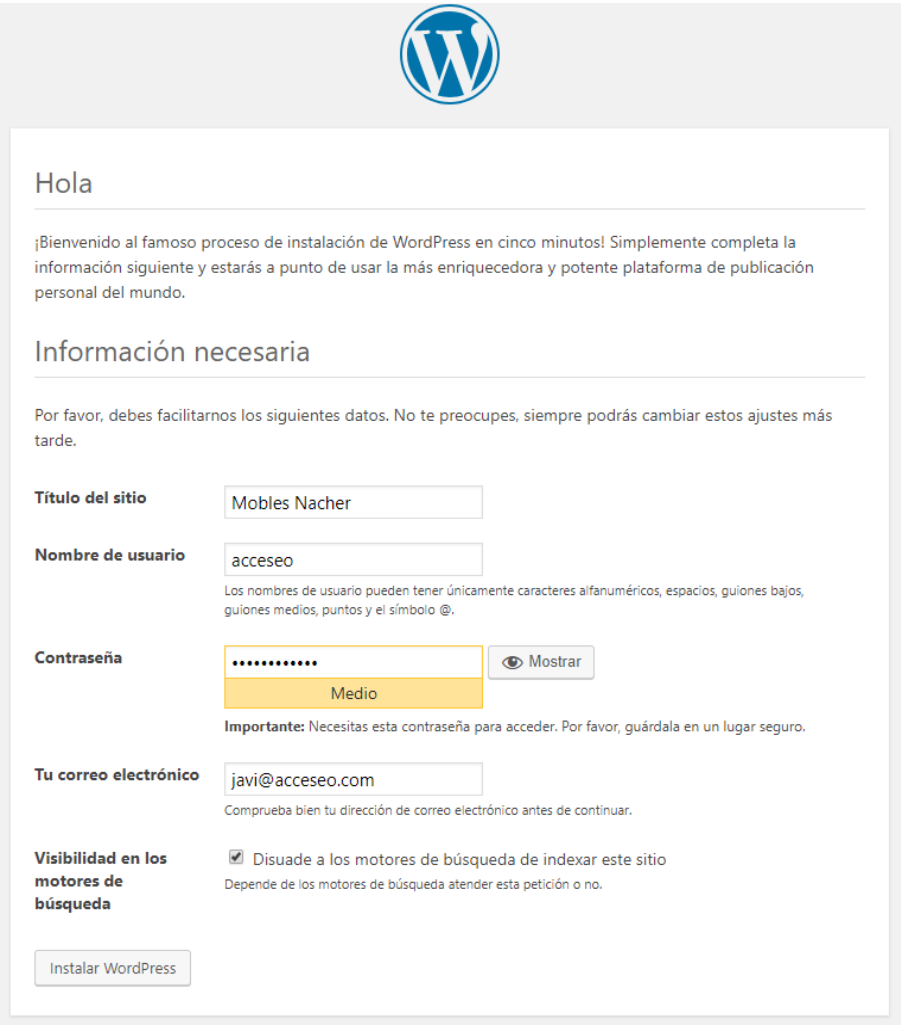

ILUSTRACIÓN 17 - INSTALACIÓN DE WORDPRESS 5

En este caso, no haría falta seleccionar “disuadir los motores de búsqueda” ya que no pueden acceder desde fuera de nuestra red, pero más adelante cuando lo subamos a un entorno de pruebas será necesario activarlo, por lo que ya lo tendríamos activado.

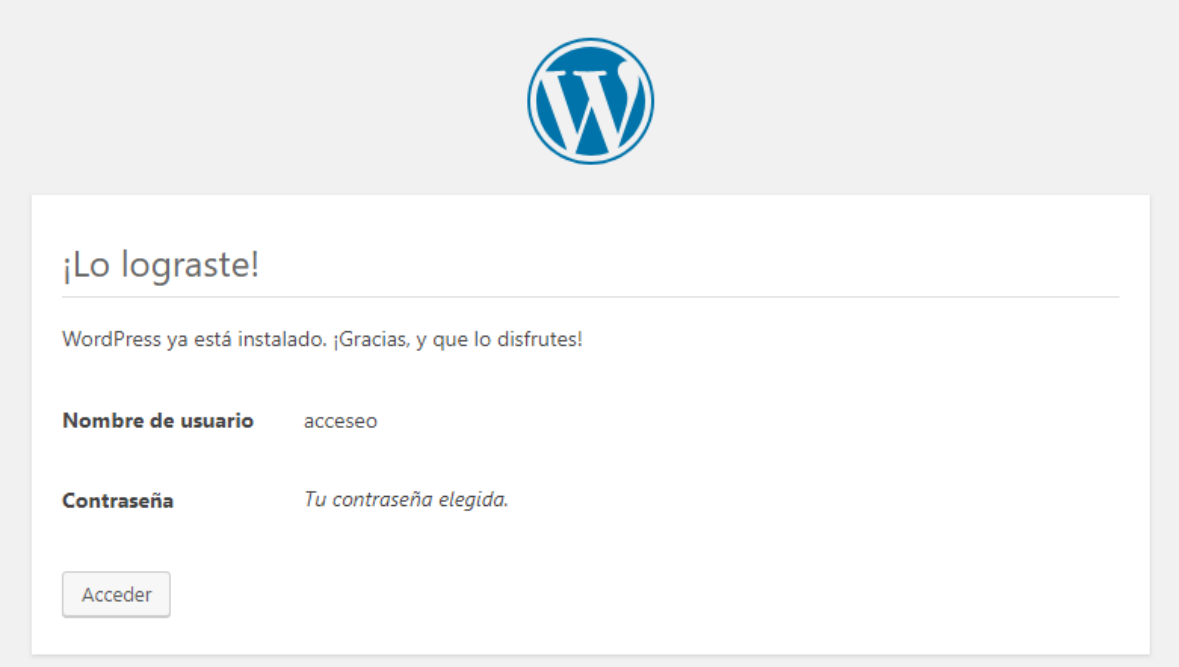

ILUSTRACIÓN 18 - INSTALACIÓN DE WORDPRESS 6

Por lo tanto, ya tendremos Wordpress instalado y vacío.

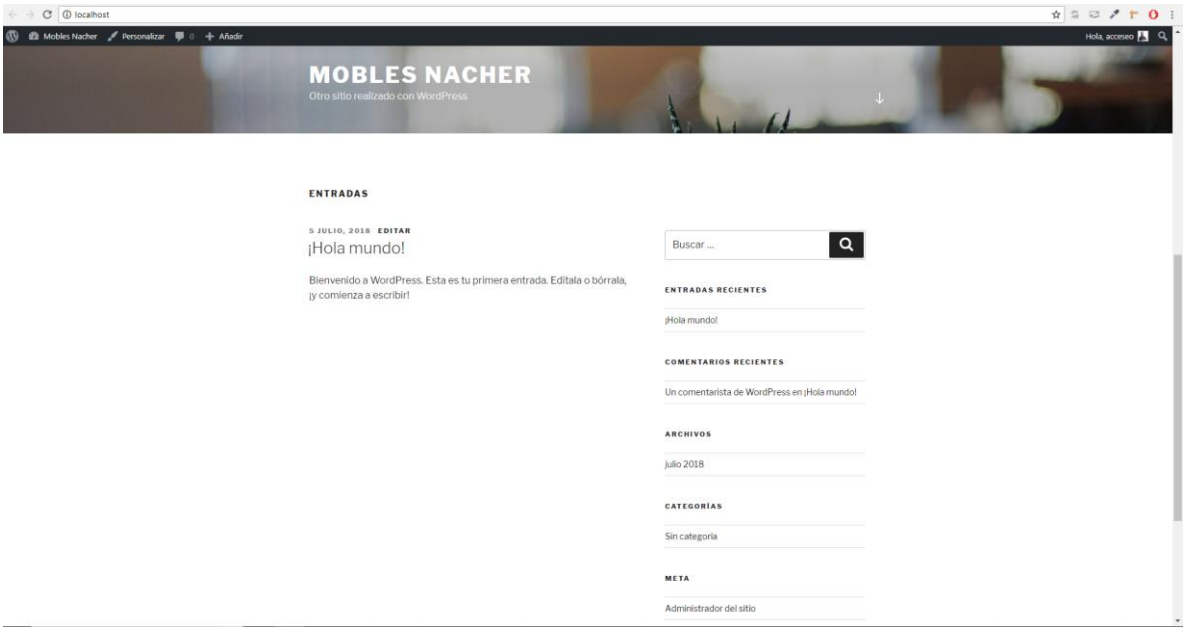

ILUSTRACIÓN 19 - INSTALACIÓN FINALIZADA

Ahora ya tenemos nuestro Worpdress listo para empezar a personalizarlo y empezar con el desarrollo para que el site sea igual al diseño inicial que aprobó el cliente.

4. Estructura de wordpress

Antes de empezar a gestionar Wordpress vamos a ver la estructura que utiliza a nivel de archivos. Si vamos a la raíz de nuestro proyecto podemos ver el listado de los siguientes archivos:

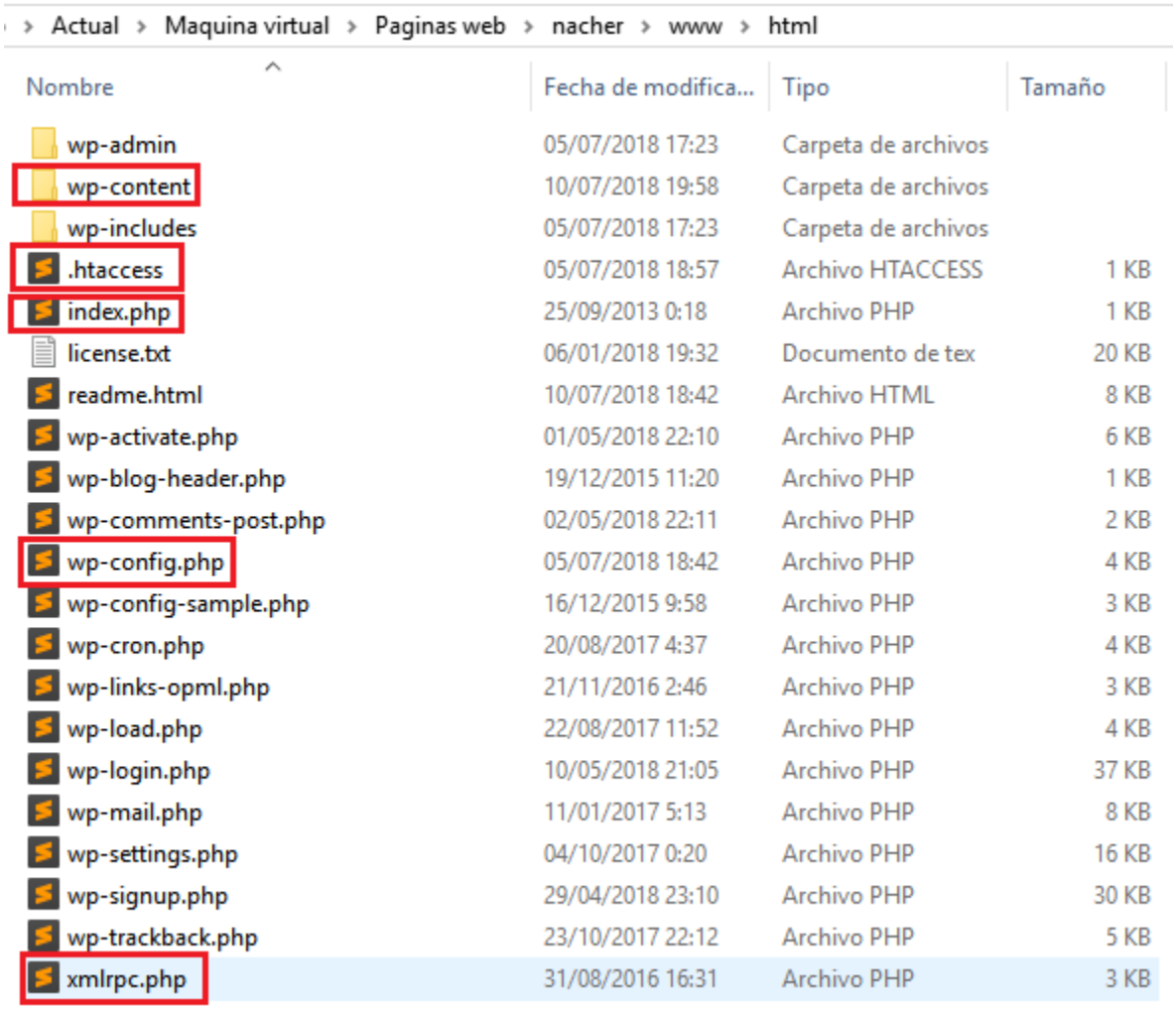

ILUSTRACIÓN 20 - ESTRUCTURA DE ARCHIVOS WORDPRESS

Hemos remarcado los archivos y directorios más importantes, todos los demás archivos son del Core de Wordpress y no se modifican, si modificamos los demás archivos cuando se actualizará el Wordpress perderíamos todos los cambios ya que se regeneran con la configuración de la nueva versión. Vamos a hablar de los archivos y directorios de la raíz.

Estos archivos y directorios que hemos remarcado son:

- **Directorio Wp-content**: En esta carpeta se almacena todo el contenido y configuración en formato "archivos" (Por lo tanto, no se almacena nada de esta configuración en base de datos). Dentro de este directorio se incluyen los plugins instalados, las imágenes, los themes, etc.
- **Archivo Index.php:** De este archivo poco podemos decir, su función es la de cargar todo el contenido. Cuando se accede a un servidor, la configuración de este busca los archivos Index y muestra el contenido del SITE (ya sea html o php).
- **Archivo wp-config.php**: Tras la instalación del CMS, el archivo WP-CONFIG.PHP es el archivo principal de configuración de WordPress, donde se guardan los datos de conexión con la base de datos de WordPress y algunos parámetros más como la activación del modo desarrollador o parámetros de administración.
- **Archivo htaccess:** Este archivo es únicamente para servidores que funcionen a través de Apache (la mayoría). En este archivo está definido el sistema de directorios para poder navegar por páginas internas de nuestra web, aparte el archivo controla las redirecciones, las URL amigables y permite el funcionamiento de algunos plugins (como los plugins de caché).
- **Archivo xmlrpc.php:** Este archivo es el encargado de ofrecer la comunicación por el protocolo XMLRPC.PHP. Actualmente Wordpress recibe multitud de ataques a través de este archivo, por lo que de vital importancia su protección.

Aparte de estos archivos o directorios que acabamos de nombrar, también hay más archivos o directorios que debemos tener en cuenta o saber que son, aunque nunca los editemos:

- **Carpeta wp-admin:** En esta carpeta está toda la parte Back-end de Wordpress.
- **Carpeta wp-includes:** En esta carpeta están todos los archivos necesarios para que Wordpress funcione, su API y librerías principales.
- **Archivo wp-login.php:** Este archivo es muy importante ya que es el encargado de gestionar el acceso al backoffice de todos los tipos de usuarios.

Ahora que hemos analizado la raíz de una instalación de WordPress, vamos a hablar del contenido de la carpeta WP-CONTENT, la carpeta de contenido de WordPress que hemos comentado anteriormente.

Dentro de la carpeta WP-CONTENT después de realizar la instalación, tenemos la siguiente estructura:

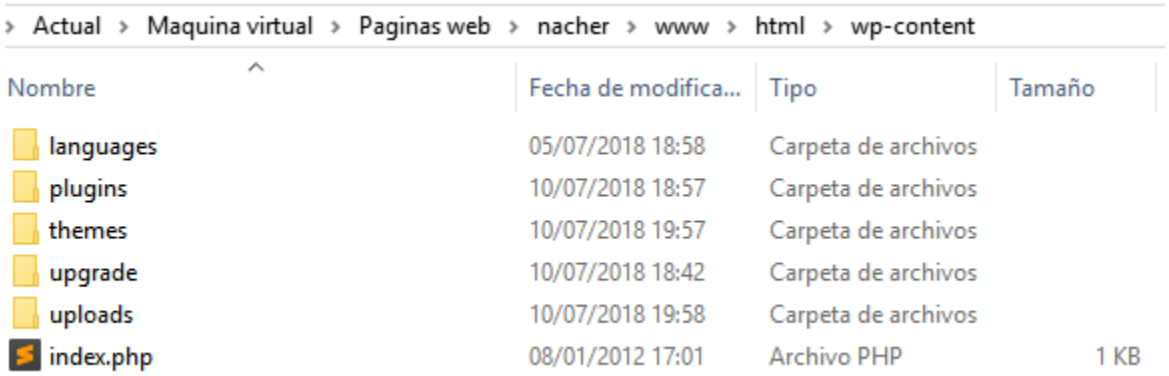

ILUSTRACIÓN 21 - DIRECTORIO WP-CONTENT WORDPRESS

Actualmente tenemos el Wordpress vacío y sin contenido, es posible que aparte de estos archivos y directorios nos encontremos alguno más, vamos a nombrar los más habituales:

- **Carpeta cache:** Esta carpeta almacena datos de plugins, estilos, etc., con la finalidad de que las futuras peticiones se ejecuten más rápidamente y no tenga que buscar directorio por directorio. Cada vez que realicemos modificaciones en nuestro Wordpress, esta carpeta se regenera.
- **Carpeta languages:** Esta carpeta almacena los archivos de traducción. Los archivos de traducción por lo general son de los plugins, de los temas o del propio Wordpress.
- **Carpeta themes:** Aquí se almacenan todos los temas de nuestro sitio web, tanto los activos como los no activos. Por lo general es recomendable únicamente tener los temas activos por seguridad.
- **Carpeta plugins:** Aquí se almacenan todos los archivos de los plugins instalados en nuestro Wordpress. La configuración de estos se almacena en Base de datos, por lo que únicamente están los archivos.
- **Carpeta Blogs.dir:** Esta carpeta únicamente existe cuando un Wordpress es multisite. El contenido de los blogs hijos se almacena en esta carpeta.
- **Carpeta uploads:** En esta carpeta se almacena todo el contenido multimedia que subamos a nuestro Wordpress desde el panel.
- **Carpeta upgrade**: Aquí se guardan algunos archivos mientras realizamos actualizaciones de Wordpress, por lo general está siempre vacía.

5. Instalación del theme

El primer paso una vez está el Wordpress recién instalado (vacío), vamos a instalar el theme. Como en nuestro caso hay que basarse en un diseño inicial (realizado en photoshop) no podemos utilizar un theme ya prediseñado, por lo que vamos a utilizar un theme constructor basado en Bootstrap. Este theme tendrá para cada página un "constructor" visual con cajas y herramientas que nos facilitaran el tiempo de desarrollo. Este theme "constructor" también sirve para que el cliente final pueda gestionar de forma más fácil y rápida su Site.

El theme que nosotros vamos a utilizar se llama "Divi" y está desarrollado por la empresa Elegant Themes. Este theme es de pago. No es el único theme "constructor" que existe, hay otras opciones como Visual Compouser, themify o Live compouser. Nosotros utilizamos Divi porque tiene un buen soporte y sigue teniendo un mantenimiento periódico. Actualmente es de los más potentes que existe en el mercado.

La instalación del theme la realizaremos desde la parte visual del Wordpress, en el menú lateral izquierdo vamos a "/Apariencia/Temas".

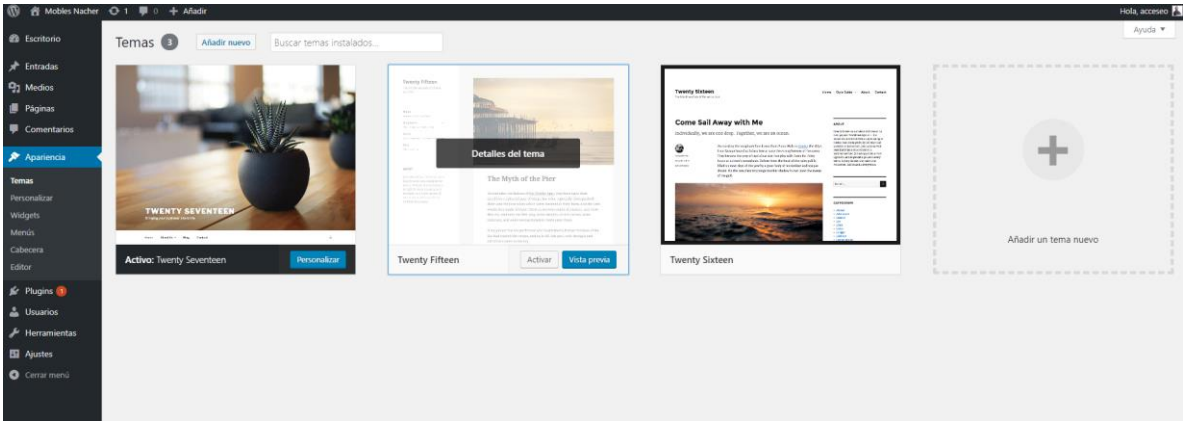

ILUSTRACIÓN 22 - PÁGINA TEMAS WORDPRESS

Como podemos comprobar hay 3 Temas instalados por defecto que vienen con Wordpress. Le damos a Añadir nuevo, seleccionamos el ZIP con el theme (Divi) y le damos a Añadir.

Una vez finalizado el proceso de Instalación, debemos crear el Child theme.

6. Creación del Child Theme

El child theme es un tema secundario que hereda la funcionalidad y el estilo de otro tema (tema padre). Cuando un theme se actualiza lo que hace es borrar todos los archivos y reemplazarlos por los nuevos actualizados, por lo tanto, en la instalación de una nueva actualización perderíamos todo el desarrollo realizado en este y volveríamos a tener el theme

por defecto. Estas actualizaciones suelen salir cada poco tiempo y, como nosotros tenemos el theme comprado, podemos aprovecharnos de sus actualizaciones.

Crear un child theme es muy sencillo, nos iremos al directorio de los themes en Wordpress (/wp-content/themes/) y creamos una nueva carpeta, como el theme es personalizado para la empresa final lo llamaremos "Nacher Alcoy".

Una vez creada la carpeta, tendremos que crear un nuevo archivo style.css donde incluiremos una cabecera para decirle que es un tema hijo y extenderá de su tema padre.

La cabecera que añadiremos los siguientes datos:

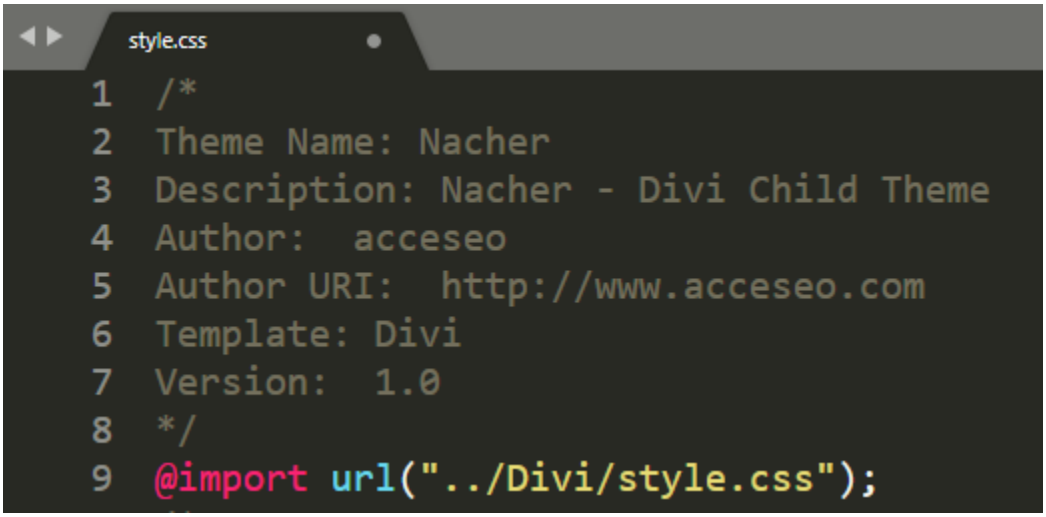

ILUSTRACIÓN 23 - CREACIÓN DE UN CHILD THEME

Con esto le estamos diciendo el nombre de nuestro theme, la descripción, autor, etc. Estos parámetros los utilizaremos para previsualizarlos desde Wordpress y el "import" donde le asignamos la ruta de su tema padre (donde cogerá todos los archivos).

Nuestro child theme ahora mismo está vacío y solo incluye el Style.css, como vayamos avanzando en el desarrollo de la web iremos cogiendo archivos del tema padre que queramos modificar y cuando se actualice éste no perderemos la modificación realizada.

Una vez tenemos nuestro tema padre y nuestro tema hijo, no necesitamos ya los temas que instala el Wordpress por defecto, por lo tanto vamos a proceder a borrarlos. Estos themes los borramos para liberar capacidad de nuestro servidor y sobre todo porque, para dejarlos debemos mantenerlos actualizados por motivos de seguridad. Por lo tanto, no los necesitamos y tenerlos nos perjudica en tiempo de mantenimiento futuro.

Para borrar estos temas, vamos a (/Apariencia/Temas) y hacemos Click en cada uno de ellos y le damos a eliminar o tenemos la alternativa de realizarlo vía FTP borrando directamente sus carpetas. Una vez desinstalados en nuestro directorio de temas solo aparecerá el tema padre (Divi) y el tema hijo (Nacher Alcoy).

Por último, hacemos click en nuestro tema hijo y le damos a activar.

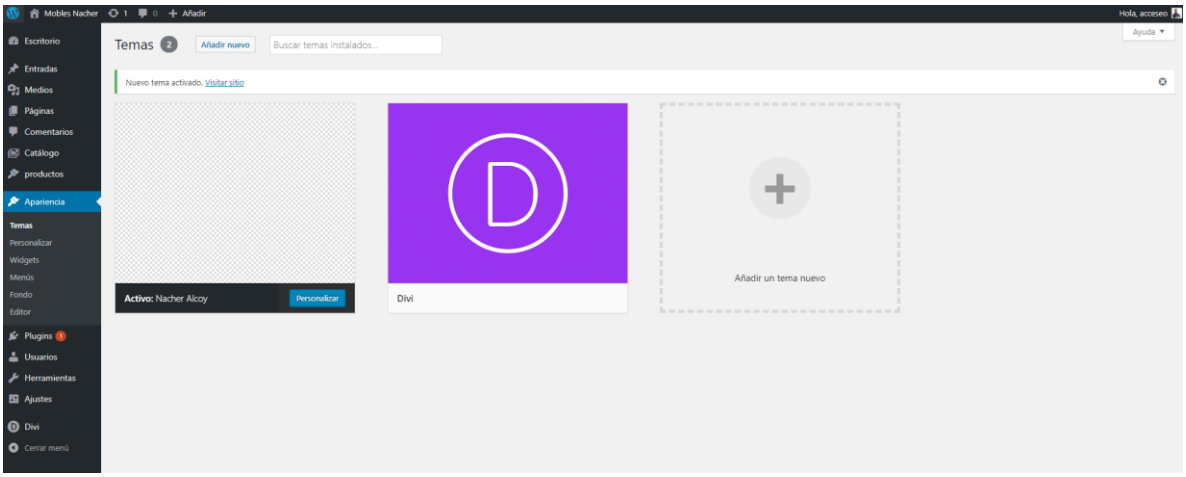

ILUSTRACIÓN 24 - ACTIVACIÓN DEL CHILD THEME

7. Custom post type

Una vez tenemos el Wordpress instalado, vemos que una instalación básica no cumple con todos los requisitos que necesitemos para el desarrollo del proyecto, ya que actualmente solo disponemos de páginas y de entradas para un blog.

En nuestro proyecto el cliente nos solicitó que debe tener un catálogo de productos, además estos productos se componen de varias familias. Las familias son: auxiliares, estanterías, mesas, mesas auxiliares, novedades, sillas y sillas lounge, por lo que a cada producto habrá que asignarle una de estas familias y, cuando entremos a una de estas familias, únicamente podremos visualizar los productos de esta.

Para poder cumplir con esta necesidad tendremos que desarrollar la creación de un Custom Post Type.

7.1 ¿Qué es un custom post type?

Cuando WordPress empezó, tan solo tenía las “entradas”. No tenía ni el apartado páginas, ya que WordPress solo servía para crear blogs. Con el tiempo los desarrolladores de WordPress vieron que las funcionalidades que tenían eran escasas y que necesitaban ir más allá, entonces crearon las “Páginas”, que son una “especie de post”, o bien dicho “Tipo de post” y de ahí viene el “Post Type”.
Los Custom Post Types son simplemente tipos de entradas personalizadas que podemos añadir a nuestro WordPress (una nueva sección dentro del dashboard). Con esto conseguimos

separar el apartado “Entradas” de nuestro apartado personalizado (Ya bien sea productos, libros, revistas, etc.) y no lo mezclaremos.
En la creación de un Custom Post Type podemos asignarle o quitarle todo lo que queramos. Las páginas se diferencian de los post en que no tienen fecha, no tienen autor, etc., por lo que es un apartado hecho a medida con solo la información o campos que necesitamos.
Y eso es lo que son los CPT: Nuevos contenidos que podemos agregar a WordPress, además de las Entradas (posts) y las Páginas (post types).

7.2 ¿Cómo creamos un custom post type?

La creación y personalización de Custom Post Types es muy sencilla. Tenemos varias opciones para la creación de estos y además cada una tiene su nivel de dificultad (unas no necesitas conocimientos y otras sí), las opciones son las siguientes:

- **A través de un plugin o plantilla**

Hay ciertos plugins que te crean custom post type por defecto al instalarlos, tenemos el caso de WooCommerce, que crea la sección “Productos” en el dashboard. Es la forma más simple, ya que lo único que tenemos que hacer es instalar y activar el plugin. Hay que tener en cuenta que este plugin no está destinado a crear CPTs, simplemente es un efecto de los mismos.
La otra opción es coger una plantilla enfocada al sector que te dedicas y posiblemente ya tenga el contenido que deseas como hemos comentado antes.

- **A través de un plugin de CPTs**

El plugin más famoso y además gratuito se llama Custom Post Type UI. Es un plugin creado únicamente para esta finalidad. Por lo general lo utiliza gente sin conocimientos técnicos, que no es nuestro caso.

Básicamente tendremos que ir rellenando las opciones que nos solicita en base a nuestras necesidades, fácil y sencillo, pero no es la forma óptima por los siguientes motivos:

- En primer lugar, porque tenemos que tener ese plugin siempre instalado y activo (si lo desinstalamos, el CPT desaparece) y por motivos de seguridad contra menos plugins utilicemos mejor.
- En segundo lugar, porque afecta al rendimiento de la web en sí, pues no deja de ser un plugin más en nuestra instalación, y todo suma.
- Y, en tercer lugar, porque el menú del plugin está a la vista, es muy probable que antes o después el cliente final lo acabe curioseando, y probablemente, rompiendo algo.

- **A través de un código**

Esta opción es la más recomendable, siempre y cuando se tenga la experiencia necesaria. En nuestro caso como la tenemos, es la opción que vamos a utilizar.

En primer lugar, siempre debemos ir de la mano del Codex (librería de los CPT) ya que hay muchas cosas las cuales podemos consultar (funciones, etiquetas, etc.).

El primer paso para el desarrollo de un CPT es la creación de un archivo PHP que lo vamos a llamar "cpt-productos.php". En este archivo vamos a incluir las siguientes líneas de código:

```php
<?php

add_action( 'init', 'productos_cpt_create' ); //Le decimos a Wordpress que ejecute la función productos_cpt_create


function productos_cpt_create() { //Función productos_cpt_create
    $labels = array(
        'name' => __( 'productos'),  //Nombre de nuestro custom post type
        'singular_name' => __( 'producto' ), //Nombre de un objeto solo (singular)
        'add_new' => _x( 'Añadir nuevo', 'producto' ), // Alta de un nuevo CPT (Base de datos)
        'add_new_item' => __( 'Añadir nuevo producto'), //Alta de los productos (Base de datos)
        'edit_item' => __( 'Editar producto' ), //Funcionalidad de editar productos (Wordpress)
        'new_item' => __( 'Nuevo producto' ), //Funcionalidad Añadir un nuevo producto (Wordpress)
        'view_item' => __( 'Ver producto' ), //Ver el producto ya creado (Wordpress)
        'search_items' => __( 'Buscar productos' ), //Buscador de productos
        'not_found' =>  __( 'No se ha encontrado ningún producto' ), //Busqueda sin resultados
        'not_found_in_trash' => __( 'No se han encontrado productos en la papelera' ), //Busqueda sin resultados en la papelera
        'parent_item_colon' => ''
    );

    // Creamos un array para $args, Esto nos sirve luego para en una página llamar a la clase producto y coger sus atributos.
    $args = array(
        'label' => __('productos'),
        'labels' => $labels,
        'public' => true,
        'can_export' => true,
        'show_ui' => true,
        '_builtin' => false,
        'capability_type' => 'post',
        'hierarchical' => false,
        'rewrite' => array( "slug" => "productos" , 'with_front' => true),
        'supports'=> array('title','editor','thumbnail','excerpt') ,
        'show_in_nav_menus' => true,
        'taxonomies' => array( 'productos_category'),
        'menu_icon' => 'dashicons-admin-appearance',
        'map_meta_cap' => true
        );

    register_post_type( 'productos', $args ); /* Registramos y a funcionar */
}
```

ILUSTRACIÓN 25 - CREACIÓN DE UN CUSTOM POST TYPE

Ahora ya tenemos el archivo de creación del CPT productos, ahora el siguiente paso es decirle a Wordpress que ejecute este archivo. Para decirle a Wordpress que ejecute nuestro archivo, tendremos que ir al directorio de nuestro tema y añadir un include de nuestro php al archivo functions.php (ubicado en la raíz del tema).

7.3 ¿Qué es el archivo functions.php?

El archivo functions.php es un archivo opcional de Wordpress. Es una biblioteca personal de funciones, es una manera muy fácil de ampliar, cambiar o alterar el comportamiento de éste. Se podría decir que el archivo functions se comporta igual que un plugin, añadiendo estas características o funcionalidades extra necesarias para poder cumplir nuestros objetivos.

En el tema padre existe este archivo pero como comentamos anteriormente, si modificamos cualquier archivo de este tema y actualizamos, perderemos todos los cambios. La solución a este problema es crear un archivo llamado functions.php en nuestro tema hijo y a continuación le añadiremos el include de nuestro CPT.

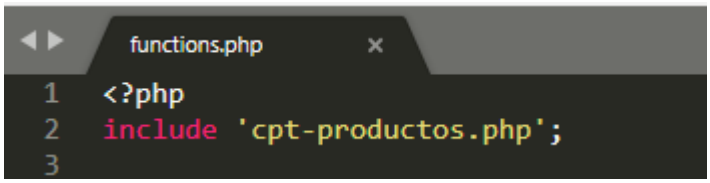

ILUSTRACIÓN 26 - INCLUDE DE UN ARCHIVO PHP

Una vez tengamos añadido esto en nuestro tema hijo, si vamos a nuestro panel de Wordpress, veremos que ya disponemos del apartado “Productos”.

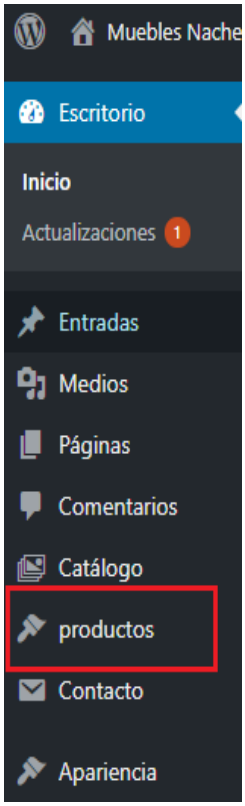

ILUSTRACIÓN 27 - NUEVA SECCIÓN PRODUCTOS

Ahora seguimos teniendo un problema, nuestro nuevo apartado de productos no dispone de categorías necesarias para catalogar a los productos según su tipo. Para ampliar nuestro CPT y decirle que queremos que también tenga categorías lo haremos añadiendo el siguiente código a nuestro archivo cpt-productos.php:

```php
add_action( 'init', 'productos_category_taxonomy', 0 ); //Llamamos a la función productos_category_taxonomy
function productos_category_taxonomy(){
    $labels = array(
        'name' => _x( 'Categorías', 'taxonomy general name' ), //Le decimos que la opción en el menu se llamara "Categorías"
        'singular_name' => _x( 'Categoría', 'taxonomy singular name' ), //Su nombre en singular "Categoría"
        'search_items' =>  __( 'Buscar Categorías' ), //Añadimos la funcionalidad del buscador de categorías
        'popular_items' => __( 'Categorías Populares' ), //Funcionalidad de categorías más populares
        'all_items' => __( 'Todas las Categorías' ), //Opción de mostrar todas las categorías
        'parent_item' => null, //Si estas categorías tendran padres y hijos (en nuestro caso no).
        'parent_item_colon' => null,
        'edit_item' => __( 'Editar Categorías' ), //Funcionalidad para poder editar las categorías
        'update_item' => __( 'Actualizar Categorías' ), //Funcionalidad de actualizar las categorías una vez editadas
        'add_new_item' => __( 'Añadir Categoría' ), //Funcionalidad para añadir nuevas categorías
        'new_item_name' => __( 'Nuevo Nombre de Categoría' ), //Nombre de la nueva categoría
        'separate_items_with_commas' => __( 'Separe las categorías con comas' ), //A nivel de codigo
        'add_or_remove_items' => __( 'Añadir o quitar categorías' ), //Añadir o quitar categorías
        'choose_from_most_used' => __( 'Elija de las categorías más usadas' ), //Categorías más usadas
        );

    register_taxonomy('familia','productos', array( //Registramos las categorías
        'labels' => $labels,
        'hierarchical' => true,
        'show_ui' => true,
        'query_var' => true,
        ));
}

```

ILUSTRACIÓN 28 - AMPLIACIÓN CPT CON CATEGORÍAS

Una vez añadamos este código al archivo, si vamos al panel de Wordpress y si dejamos el ratón encima de "productos" se nos abre un desplegable con las opciones básicas (ver los productos o añadir uno nuevo) más "añadir categoría".

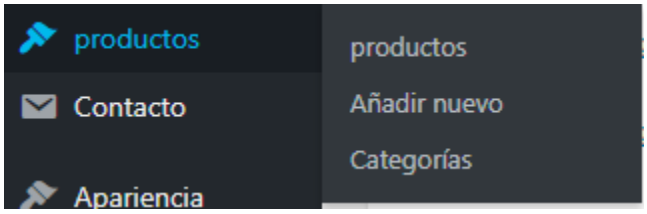

ILUSTRACIÓN 29 - NUEVO MENÚ DE CATEGORÍAS

Las categorías se crean desde la sección categorías y luego se asignan dentro de la creación de un producto. A un producto le puedes asignar más de una categoría, en nuestro caso no es necesario.

8. Instalación de plugins

Al igual que los CPT los plugins son herramientas que extienden las funcionalidades de Wordpress. Todos estos plugins que vamos a utilizar podríamos realizarlo por código, pero como hemos comentado anteriormente Wordpress dispone de un gran catálogo (algunos de pago otros gratuitos).

En nuestro caso no somos partidarios de utilizar plugins para el desarrollo de un site pero como hay funcionalidades muy complejas que tienen un costo de tiempo elevado para su desarrollo, la mejor opción es instalar plugins que cumplan las características necesarias. Todos estos plugins que utilizamos están previamente testeados y tienen actualizaciones periódicas para evitar pérdidas de rendimiento y conservar la seguridad de nuestro site.

Vamos a ver con más detalle los plugins necesarios para este site:

- **WPML (Wordpress multilenguaje):** Este plugin es de pago, su utilidad es poder añadir más de un idioma a nuestro site. Para cada página desarrollada puedes crear duplicados y traducirlos.
- **Yoast SEO:** Este plugin se utiliza para potenciar el SEO de nuestra página web, tiene múltiples funcionalidades. Está instalado para el departamento de Marketing online que son los encargados de potenciarlo. Mientras nosotros vamos realizando el desarrollo, los de Marketing trabajan en paralelo para optimizar el site.
- **Contact Form 7:** Este plugin es para realizar formularios de contacto, en este caso se instala para que el cliente final pueda editar cualquier campo y toda la información. En el caso de que realizáramos un FORM a nivel de código, el cliente no podría realizar ninguna acción sin conocimientos previos. Aparte tiene la ventaja que incorpora el recaptcha de google.
- **SF Taxonomy Thumbnail:** Este plugin añade imágenes destacadas a las categorías. En este caso lo necesitamos para las categorías de los productos, ya que cuando montemos el diseño cada categoría tendrá su imagen destacada.
- **Antispam Bee:** Este plugin es el encargado de evitar el SPAM en nuestra página web. Los comentarios en nuestro blog o los contactos se registran en una base de datos dentro del Wordpress, una vez hemos recibido uno podemos aceptarlo o denegarlo y se queda en cola pendiente. Muchos Hackers utilizan este sistema para llenar los servidores y tumbar las páginas.
- **Fastest Cache:** Este plugin también es de pago, aunque tiene su versión gratuita (con menos opciones). Este permanecerá desactivado hasta la finalización del desarrollo. Su función es la minificación de archivos css, javascript y además crea una caché para no tener que hacer peticiones al servidor de los archivos cacheados por cada usuario y mejorar el rendimiento de carga de nuestro sitio.

Estos plugins los podemos añadir de dos formas, por el panel de Wordpress o vía FTP. Para añadir estos plugins desde el panel de Wordpress, vamos a la sección plugins y los buscamos e instalamos uno a uno o por FTP, con los plugins previamente descargados y descomprimidos, subimos todas las carpetas (todos a la vez) a nuestro directorio de FTP.

Una vez añadidos los plugins solo quedará activarlos y configurarlos (en el caso de que sea necesario y en el momento que los vayamos a utilizar).

9. Sass

Una vez tenemos el Wordpress listo debemos empezar a realizar el diseño y darle estilo a nuestro site y antes de empezar a añadirlo debemos realizar la configuración de Sass, ya que en los proyectos hacemos uso de esta tecnología para ampliar nuestro CSS.

9.1 ¿Qué es Sass?

Sass es un metalenguaje de Hojas de estilo en cascada. Es un lenguaje script que traduce todo su contenido insertado en CSS. Con Sass conseguimos tener nuestro código mucho más organizado (estilo en cascada). Esto nos permite no tener "mil parches" a la hora de realizar futuras modificaciones en nuestro código y tener todo el código de un sector en concreto junto. Otra de las ventajas que incorpora Sass es que permite la creación de variables. Estas variables sobre todo las utilizamos para definir los colores y las fuentes de nuestra web, con esto evitamos tener que recordar el color en hexadecimal o copiarlo cada vez que lo necesitemos.

Vamos a ver un ejemplo:

- **Código CSS:**

```css
#barra-categorias{
    background: black;
    padding: 15px 0px;
    margin: 0px 15px;
}
#barra-categorias   ul{
    text-align: center;
}
#barra-categorias ul li{
    display:inline-block;
    margin:0px 15px;
}

#barra-categorias ul li a{
    font-size:22px;
    font-family: 'Jura', sans-serif;
    color:white;
    font-weight: 400;
}
```

ILUSTRACIÓN 30 - CÓDIGO CSS

- **Código Sass (con la variable $jura):**

```
#barra-categorias{
    background: black;
    padding: 15px 0px;
    margin: 0px 15px;

    ul{
        text-align: center;
        li{
            display:inline-block;
            margin:0px 15px;

            a{
                font-size:22px;
                font-family: $jura;
                color:white;
                font-weight: 400;
            }
        }
    }
}
```

ILUSTRACIÓN 31 - CÓDIGO SASS

9.2 ¿Que necesitamos para poder utilizar Sass?

Esta herramienta es muy fácil de instalar y utilizar, no requiere la instalación de nada en el lado del servidor, únicamente hay que convertir el texto Sass a CSS, por lo tanto, lo único necesario es un compilador de Sass CSS. Actualmente hay multitud de aplicaciones gratuitas que realizan esta función y solo tendremos que buscar un poco por Google para comprobarlo.

La aplicación que nosotros utilizamos se llama Koala. Esta aplicación nos solicita dos datos, el directorio del archivo de entrada (Sass) y el directorio del archivo de salida (css). Una vez agregamos nuestros archivos únicamente tendremos que tener la aplicación abierta, ésta cuando detecta un cambio en el archivo de entrada, automáticamente lo compila y añade el texto al archivo de salida.

10. Modificación de plantillas

Con el Wordpress ya configurado y todas las herramientas que necesitamos instaladas (plugins y herramientas de desarrollo) vamos a empezar a realizar el diseño de nuestro site. Los desarrollos se empiezan desde el home incluyendo el header y el footer que lo comparten todas las páginas.

Basándonos en el diseño nos damos cuenta de que hay apartados en los que nuestra plantilla constructora no dispone de la funcionalidad, por lo que tendremos que desarrollar con código para ampliar las funcionalidades.

Para desarrollar estas funcionalidades utilizaremos los archivos del FTP, bien modificándolos o creando nuevas plantillas. La ampliación del código la realizaremos dentro de nuestro Child Theme (para no perder nada en futuras actualizaciones).

El primer paso que tenemos que hacer es copiar los archivos Header.php, page.php y Footer.php de nuestro tema padre a nuestro tema hijo, ya que vamos a realizar modificaciones directamente en el código de estos archivos y si lo hiciéramos en el tema padre y, en un futuro lo actualizamos, perderíamos todos nuestros cambios.

10.1 Plantilla header

En este caso la plantilla no dispone de constructor para el Header por lo que todas las modificaciones tendremos que realizarlas manualmente desde el código. En este archivo tenemos que realizar 2 modificaciones, uno asignarle la imagen del logo manualmente ya que esté lo añadiremos en formato ".SVG" porque se visualiza correctamente en todos los tamaños y no pierde calidad. El logo tendremos que añadirlo manualmente porque Wordpress no permite la subida de archivos svg por seguridad ya que dentro de estos archivos puedes añadir código malicioso. Este archivo ha sido generado por el diseñador de la empresa, por lo que es fiable y no tendremos ningún problema.

Para añadirlo, únicamente tendremos que subir vía FTP a la carpeta de medios (/wp-content/uploads/) el logo y asignarle la ruta:

```
<img src="/wp-content/uploads/logo.svg" alt="<?php echo esc_attr( get_bloginfo( 'name' ) ); ?>" id="logo"
 data-height-percentage="<?php echo esc_attr( et_get_option( 'logo_height', '54' ) ); ?>" />
```

ILUSTRACIÓN 32 - SUSTITUIR LOGO POR UN SVG

Ahora vamos con la segunda parte de la modificación del header, lo que vamos a realizar es mediante código una nueva barra superior donde vamos a añadir las Redes sociales y aparte, el selector de idioma.

El código utilizado para el idioma lo hemos sacado de la librería del plugin para evitar utilizar el que viene por defecto, así personalizarlo y darle el estilo que nosotros queramos.

```
<div id="idiomas">
    <?= do_shortcode('[wpml_language_switcher]
        {% for code, language in languages %}
            <a href="{{ language.url }}">
                {% if language.code == \'es\' %}
                    ES
                {% endif %}
                {% if language.code == \'en\' %}
                    EN
                {% endif %}
                {% if language.code == \'fr\' %}
                    FR
                {% endif %}
                {% if language.code == \'it\' %}
                    IT
                {% endif %}
            </a>
        {% endfor %}
        [/wpml_language_switcher]');?>
</div>
```

ILUSTRACIÓN 33 - CÓDIGO IDIOMAS

Por otra parte, para los iconos de las redes sociales vamos a utilizar la librería "Fontawesome". Esta librería de iconos es una fuente por lo que su peso es mínimo y reducido. Estos iconos se tratan igual que el texto, dándole tamaño, etc. Con esto liberamos en gran medida la carga realizada, ya que en vez de cargar imágenes está cargando texto y la carga la realiza muchísimo más rápido, por lo que es la forma más óptima.

La librería Fontawesome es muy sencilla de instalar, hacemos la importación del script:

```
<script src="https://use.fontawesome.com/c9b6872fca.js"></script>
```

ILUSTRACIÓN 34 - IMPORTACIÓN JS

A continuación añadimos las clases que podemos encontrar en su web para cada icono a cada etiqueta <a> (enlace) y ya se muestran todos los iconos:

```
<div id="redes-area">
    <a href="/area-reservada" id="area-reservada">área reservada</a>
    <span>|</span>
    <a href="#" class="fa fa-facebook"></a>
    <a href="#" class="fa fa-twitter"></a>
    <a href="#" class="fa fa-instagram"></a>
    <a href="#" class="fa fa-pinterest-p"></a>
</div>
```

ILUSTRACIÓN 35 - CREACIÓN DE ICONOS REDES SOCIALES

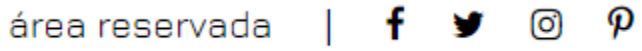

ILUSTRACIÓN 36 - EJEMPLO NUEVA BARRA WEB

Una vez tengamos la edición de la plantilla para añadir la nueva barra y cambiar el logo por uno en formato ".svg" solo nos faltará asignar el menú. Para ello nos vamos a "/Apariencia/Menús/" y creamos uno nuevo. Al nuevo menú le vamos a llamar "Menú principal" y lo vamos a asignar como "Menú principal", así Wordpress ya sabe que es el que tiene que mostrar en el header de la página web.

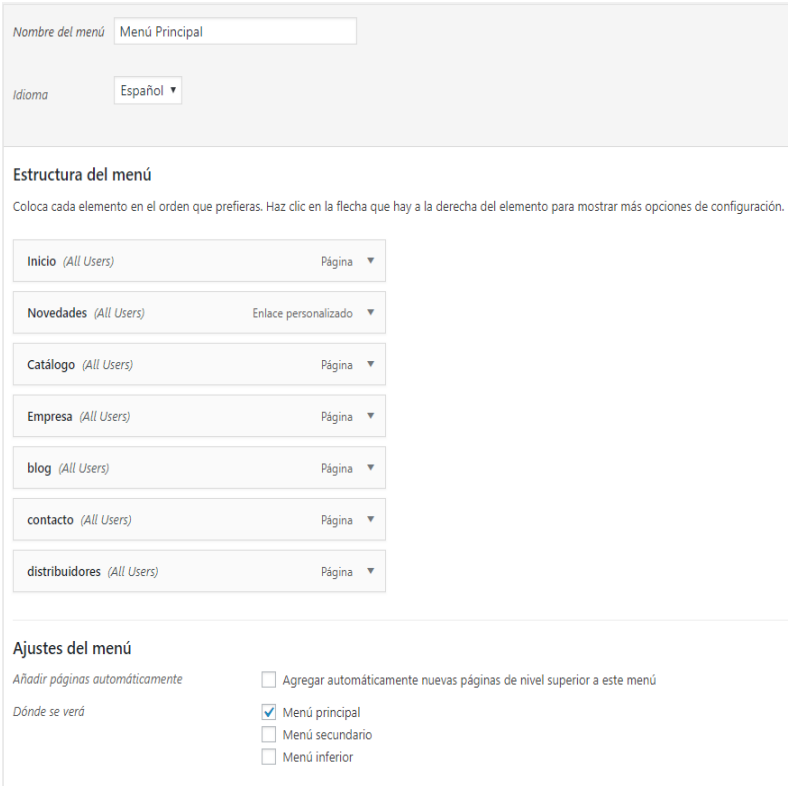

ILUSTRACIÓN 37 - CREACIÓN DE MENÚ

10.2 Plantilla home

Luego nos encontraremos otro problema con el archivo page.php. Este archivo hace referencia a la plantilla utilizada para todas las páginas creadas desde el Wordpress pero nosotros queremos que cada plantilla desarrollada sea independiente para cada página o grupo de páginas (por ejemplo la página de productos). Como actualmente nuestro child theme no dispone de este archivo vamos a copiarlo del tema padre y lo añadimos en nuestro tema hijo (desde divi a Nacher).

Ahora necesitamos decirle a Wordpress que esta plantilla será únicamente para la home, por lo que vamos a renombrarla y la llamaremos page-home.php. Esto no es suficiente para decirle a Wordpress que es una plantilla nueva, si no que tendremos que añadirle la siguiente cabecera al archivo:

```
<?php
    /*
    Template Name: Página Inicio
    */
?>
```

ILUSTRACIÓN 38 - CREACION DE UNA PLANTILLA

Ahora en la edición de la página Home dentro del Wordpress, tenemos que asignarle que cuando cargue esta página, debe cargar la plantilla “Página Inicio”. Para ello tendremos que ir al panel de Wordpress, Páginas y editamos la página de inicio.

Una vez estamos dentro de la página de inicio, vemos en la barra derecha, que tenemos diferentes opciones, como previsualizar cambios, actualizar cambios, añadirle traducciones, etc.
El apartado que a nosotros nos interesa es el de “Atributos de página” y en la sección “Plantilla” cargamos “Página Inicio”:

ILUSTRACIÓN 39 - ASIGNACIÓN DE UNA PLANTILLA

Con esto Wordpress ya sabe que tiene que coger la plantilla personalizada y no la que utiliza por defecto para todas las páginas “page.php”.

Una de las modificaciones que vamos a realizar en esta plantilla es añadir una caja la cual debe salir un carrusel de productos. Los productos los cogeremos de la base de datos utilizando la función de Wordpress wp_query, la cual le vamos a decir que queremos que nos muestre todos los “post type” Productos, con un máximo de 50 y que los seleccione de forma aleatoria. La función wp_query nos genera una vector con todos los elementos que le hemos solicitado, por lo que para mostrar estos elementos vamos a utilizar un WHILE para ir mostrando uno a uno con sus clases y la información que necesitamos:

```
<div id="slider-productos">
    <div class="container">
        <p class="titulo"><span class="fa fa-stop"> </span>productos</p>
        <p class="texto">descubre nuetros productos más actuales</p>
    </div>
    <div class="owl-carousel owl-theme">
        <?php
            $default_query_args = array(    //Definimos la query que vamos a realizar
                'post_type' => 'productos',
                'posts_per_page'    => 50,
                'nopaging' => true,
                'orderby' => 'rand'
            );

            $query = new WP_Query( $default_query_args ); //Invocamos a la query para obtener los datos
         ?>
        <?php
            if ( $query->have_posts() ) {   //Hacemos la comprobación de que no este vacia la Array
                // Start looping over the query results.
                while ( $query->have_posts() ) { //Invocamos producto a producto para mostrarlo

                    $query->the_post();
                ?>
                    <div class="item">
                        <a href="<?php the_permalink(); ?>" title="<?php the_title_attribute(); ?>">
                            <?php
                            the_post_thumbnail('full');
                            ?>
                            <p class="nombre-producto"><?php the_title(); ?></p>
                        </a>

                    </div>

                <?php
                }
            }
        ?>
    </div>
</div>
```

ILUSTRACIÓN 40 - CONSULTA DE PRODUCTOS POR QUERY

Ahora tendremos que decir que estos 50 productos no los debe mostrar todos inicialmente, si no tendrá que mostrarlos como hemos comentado antes con el carrusel.

Para la realización de este carrusel vamos a utilizar dos librerías, Jquery y CSS, aunque esta segunda la vamos a modificar para adaptarla al estilo del diseño y no dejaremos el estilo base inicial. La librería que hemos utilizado se llama “Owl carousel” y la podemos encontrar en los

repositorios de GITHUB. Para su instalación únicamente tendremos que descargarnos los dos archivos comentados anteriormente (Jquery y CSS) y los añadimos a la carpeta del tema hijo.

Como este carrusel solo se va a utilizar en la HOME vamos a decirle a nuestro archivo "page-home.php" que cada vez que se cargue, tiene que incluir la carga de estos dos archivos con un import. Si estos dos imports los ponemos en el Header, cada vez que se carga cualquier página el servidor los carga, por lo que no es óptimo ya que estas realizando carga innecesaria para cada página:

```
<link rel="stylesheet" href="/wp-content/themes/nacher/owl/owl.carousel.min.css" />
<link rel="stylesheet" href="/wp-content/themes/nacher/owl/owl.carousel.min.js" />
```

ILUSTRACIÓN 41 - IMPORTACIÓN DE LIBRERIAS EXTERNAS

Para añadir las funcionalidades del carrusel a nuestro listado de productos, tendremos que añadir tanto las clases que nos dicen en la documentación de la librería como la invocación Jquery para que el carrusel esté a nuestro gusto.

```
<script> //Abrimos la etiqueta Script para hacer uso de Jquery
    jQuery(document).ready(function(){ //Lo invoca cada vez que se crea la plantilla
      var owl = jQuery('.owl-carousel'); //Le añadimos estos parametros a la clase .owl-carousel
              owl.owlCarousel({
                loop:true, //Que corra siempre, no tenga ni inicio ni final
                margin:10, //10 px de margen entre ellos
                autoplay:true, //Que corra siempre sin necesidad de darle a flechas
                animateIn:'fadeIn', //Animación de entrada
                animateOut:'fadeOut', //Animación de salida
                autoplayTimeout:3000, //Velocidad de movimiento
                autoplayHoverPause:true, //Que pare si ponemos el raton encima
                responsiveClass:true, //Que tenga responsive
                responsive: {
                  0:{ //Más de 0px (0px - 600px)
                    items:1, //Muestra 1 producto
                    nav:true
                  },
                  600:{ //Más de 600px (600px - 1000px)
                    items:3, //Muestra 3 productos
                    nav: false
                  },
                  1000:{ //Más de 1000px  (>1000px)
                    items:4, //Muestra 4 productos
                    nav:true,
                    margin:20
                  }
                }

              });
    });
    //cerramos la etiqueta script
</script>
```

ILUSTRACIÓN 42 - DEFINICIÓN DEL SLIDER

Las clases que hemos añadido al listado de productos son "owl-carousel owl-theme" a la caja contenedora de todos los productos y para cada producto la clase "item" (podemos verlo en la ilustración 39 – consulta de productos por query).

10.2 Plantilla footer

La plantilla footer se compone de dos secciones. La primera es “et-footer-nav” donde estarán todas las columnas con la información. Desde dentro del Wordpress podemos editar y modificar estas columnas (en nuestro caso serán 4) y añadir los Widgets que deseemos.

Para asignarle 4 columnas nos vamos al dashboard Wordpress y en el menú lateral izquierdo seleccionamos “/Apariencia/Personalizar”, donde se nos abrirá el personalizador y nos iremos a la sección “PIE” y entramos dentro de “DISEÑO”. Una vez estamos dentro vemos que hay diferentes variables:

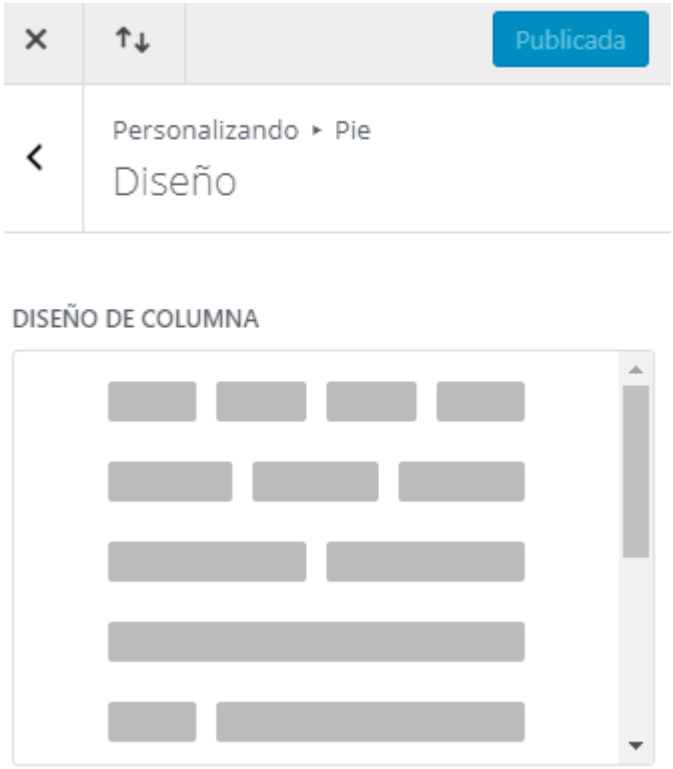

ILUSTRACIÓN 43 - PERSONALIZACIÓN ESTRUCTURA FOOTER

Ahora ya hemos seleccionado el diseño de 4 columnas, por lo que vamos a decirle al Wordpress que queremos que aparezca en cada columna con los “Widgets”, para ello vamos a “/Apariencia/Widgets” dentro del Dashboard de Wordpress:

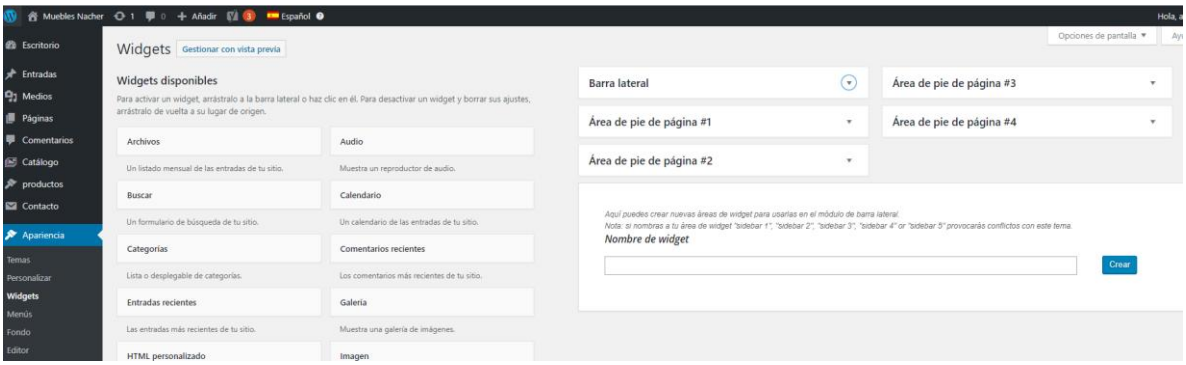

ILUSTRACIÓN 44 – WIDGETS

Como vemos en la imagen, a la izquierda tenemos los diferentes Widgets que podemos seleccionar y a la derecha las áreas donde debemos añadirlo. Nosotros para cada una de las columnas hemos elegido el Widget "texto" donde podemos añadir nuestro contenido HTML personalizado:

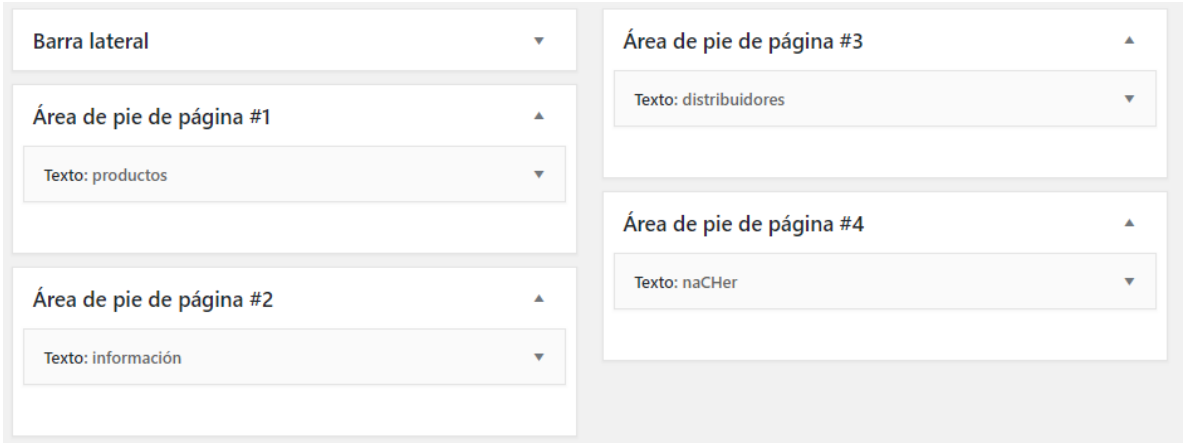

ILUSTRACIÓN 45 – RESULTADO WIDGETS AÑADIDOS

Luego una vez hemos añadido todo el contenido le damos los estilos necesarios para que se adapte al diseño.

Una vez tengamos esto acabado, faltará añadir la segunda parte del footer, esta segunda parte se llama "footer-bottom" y es el pie del footer.
En este pie por lo general únicamente va el Copyright y la publicidad de desarrollo. Esta modificación la vamos a hacer directamente en el archivo Footer y le daremos estilo con nuestro css.

11. Páginas productos

Ahora vamos a seguir con el desarrollo creando las páginas tanto de las categorías como de los productos. Como los productos son un "Custom Post Type" tendremos que duplicar los archivos "single.php" y otra vez "page.php". Single será la plantilla de un producto en concreto y Page la plantilla de la categoría de productos.

Para diferenciarlos del archivo de plantilla genérico y no aplicar los cambios a todas las páginas que utilicen estas plantillas, vamos a volver a renombrarlos. A "page.php" le llamaremos "page-categoria-productos.php" y a "single.php" le llamaremos "single-productos.php".

Wordpress tiene una funcionalidad que al añadirle "productos" busca en su base de datos todos los tipos de "post" para ver si hay alguno que coincide con "productos". En nuestro caso como previamente creamos el Custom Post Type "productos" lo enlaza automáticamente, por lo que en ningún momento desde dentro del Dashboard tendremos que asignarle nada. Con esto ya podremos empezar la modificación de nuestras plantillas.

En la plantilla de la categoría de los productos, vamos a añadir una barra donde aparezcan todas las categorías existentes con los enlaces a su página, por lo que tendremos que realizar otra "wp_query" diciéndole que queremos que nos de el array con todas las categorías creadas para la taxonomía "Productos".

```php
$query_args2  = array( //Le añadimos los parametros que deseemos
            'post_type'      => 'productos', //Tipo de post "productos"
            'taxonomy' => 'category', //Queremos que nos devuelva las categorías
            'term' => 'category-1', //Todas las categorías
            'nopaging' => true, //No queremos paginación
);

$query = new WP_Query( $query_args2 ); //Invocamos la query en la variable
```

ILUSTRACIÓN 46 - QUERY CATEGORÍAS

Con esto conseguimos que si en un futuro añaden nuevas categorías salgan automáticamente y no tengamos que ir añadiendo manualmente para cada nueva categoría.

La plantilla single es únicamente para mostrar los productos, ya que al ser "custom post type" no los muestra si no le dices al Wordpress que existe una plantilla específica para mostrarlos.

12. Creación de contenido

Una vez ya tenemos todas las plantillas con las nuevas funcionalidades creadas, solo tendremos que llenar de contenido nuestro sitio. Para la creación del contenido vamos a utilizar la plantilla constructora “Divi”, con esto conseguiremos que nuestro cliente final pueda gestionar su web de la forma más fácil y eficiente.

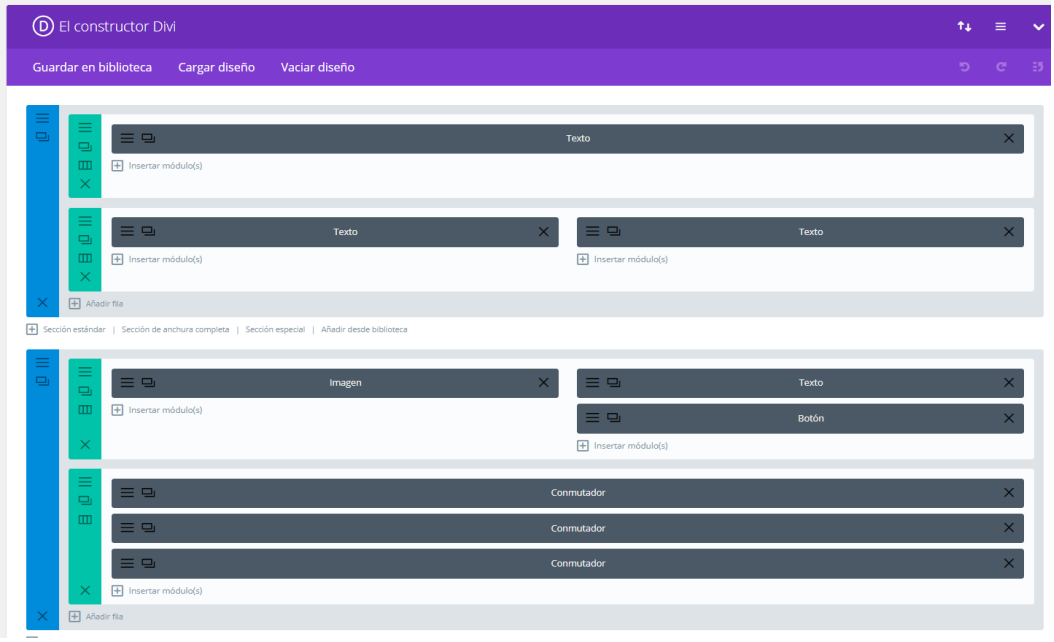

ILUSTRACIÓN 47 - CONSTRUCTOR DIVI

Como podemos ver en la imagen anterior, esto es el constructor “DIVI” el cual podremos añadir secciones y para cada sección sus cajas con las columnas necesarias. En estas cajas se puede insertar “widgets” para ajustarse a los requisitos que necesites:

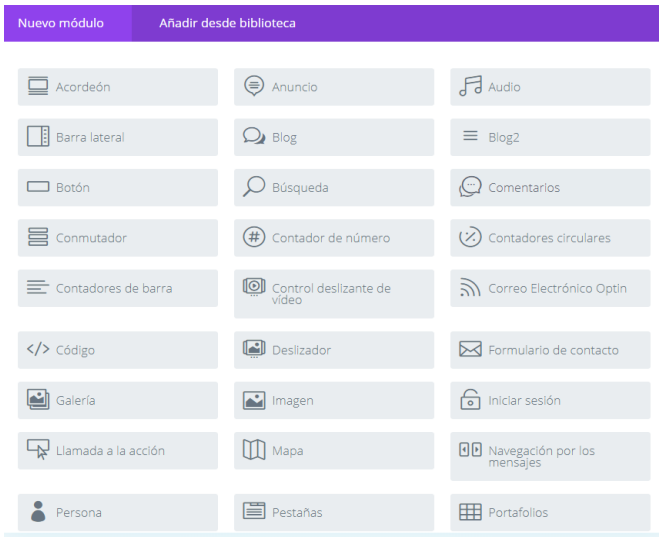

ILUSTRACIÓN 48 - MÓDULOS CONSTRUCTOR DIVI

Aparte permite añadir nuevas funcionalidades con desarrollo propio. En este caso en la segunda fila “Blog 2” es un desarrollo propio para los “custom post type”.

En las cajas donde está el contenido, nosotros lo realizaremos con contenido html (<p>, <a>, etc.) con las clases e ids necesarias. Luego el cliente podrá modificar la parte del texto sin romper el diseño y así podrá gestionar con facilidad su web. Además, podrá guiarse con las clases que le hemos asignado ya que dejan bastante claro que es cada elemento:

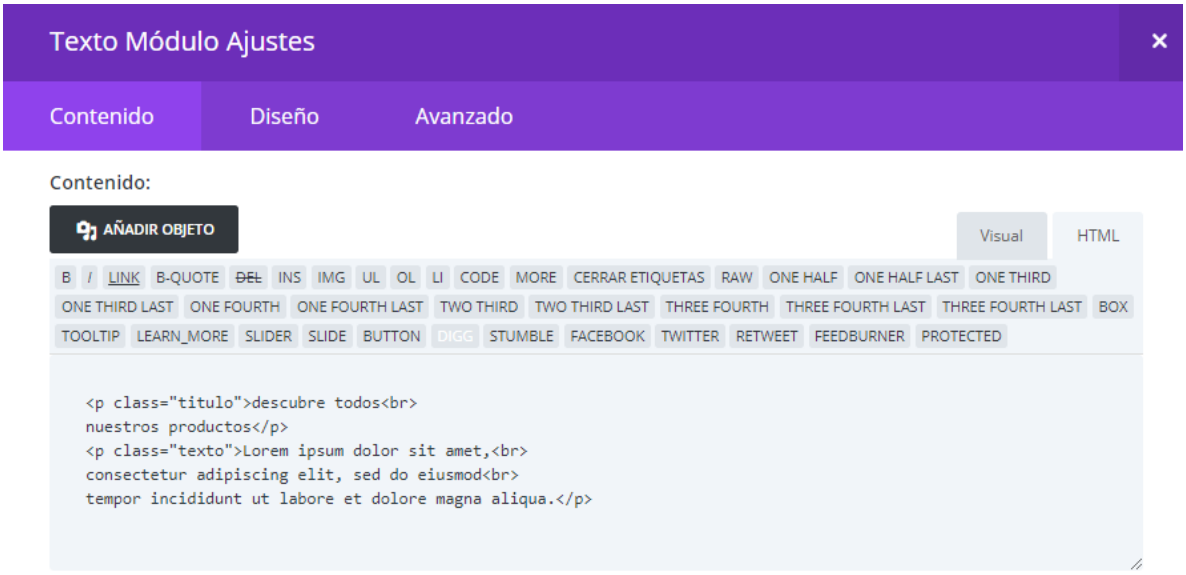

ILUSTRACIÓN 49 - CONTENIDO HTML

13. Responsive

Actualmente la mayoría de visitas que reciben las páginas web son desde dispositivos móviles, aunque seguimos priorizando el diseño de escritorio. En los eventos web más importantes ya se comenta que en los próximos años esto cambiará radicalmente y se empezará el diseño por la versión móvil y ampliará a la de escritorio. De momento aún empezamos con la versión escritorio y se adapta a la versión móvil.

Ahora que ya tenemos el desarrollo finalizado, con el diseño que nos han solicitado y todo el contenido añadido (en estos casos puede ser contenido de prueba o contenido definitivo, según si el cliente nos ha pasado ya los textos que deben colocarse en la web), nos queda adaptar nuestra web a las diferentes resoluciones de los dispositivos. Actualmente los diseños se realizan desde el tamaño máximo (1920px de ancho, versión escritorio) hasta al tamaño mínimo (320px de ancho, versión móvil), ésto quiere decir que la web debe visualizarse correctamente entre estos dos límites.

Para conseguir que se visualice correctamente entre estos dos intervalos, debemos hacer uso de las "media queries" en css y para las comprobaciones utilizaremos herramientas que nos permitirá saber si se visualiza correctamente en todos los dispositivos o si aún debemos solucionar el problema.

13.1 ¿Cómo se utilizan las media queries?

Una media querie se define como una o más expresiones, implicando características del medio, la cual se resuelve como verdadera o falsa. El resultado de la consulta es verdadera si el tipo de medio especificado en el media query concuerda con el tipo de dispositivo que está siendo mostrado y todas las expresiones en el media query son verdaderas.

Estas medias queries tienen operadores lógicos (not, and y only), además podemos anidar varias condiciones para que cumpla las características que nosotros necesitamos. Esto afectará a todos los estilos que le añadas dentro de los corchetes.

Vamos a poner varios ejemplos de media queries. En este caso vamos a decirle que queremos que aplique estos estilos **sólo** cuando la resolución sea mayor de "981px":

```
@media (min-width: 981px) {
    .et_pb_gutters3 .dialeg-calendar .et_pb_column_1_2 {
        width: 50%;
        margin-right: 0;
        padding-right: 0;
    }
}
```

ILUSTRACIÓN 50 - MEDIA QUERI 1

Ahora vamos a poner otro ejemplo y esta vez vamos a hacer uso de los operadores lógicos. En este caso vamos a decirle que queremos que su máximo sea "1280px" y su mínimo sea "1100px", como ahora necesitamos decirle un Max y un Min tendremos que utilizar el operador lógico AND:

```
@media (max-width: 1280px) and (min-width: 1100px){
    #top-menu li {
        padding-right: 6px;
    }
}
```

ILUSTRACIÓN 51 - MEDIA QUERI 2

Por último, vamos a ver otro ejemplo. Ahora vamos a hacer uso de las consultas anidadas, en este caso le vamos a decir que queremos que lo aplique cuando la resolución sea mayor que 700px pero, además, que solo lo aplique en dispositivos móviles y cuando la orientación de este sea horizontal.

```
@media (min-width: 700px), handheld and (orientation: landscape) {
    .et_pb_gutters3 .dialeg-calendar .et_pb_column_1_2 {
        width: 50%;
        margin-right: 0;
        padding-right: 0;
    }
}
```

ILUSTRACIÓN 52 - MEDIA QUERI 3

13.2 Herramienta de comprobación

Una vez sabemos cómo utilizar las media queries, necesitamos comprobar que se visualiza correctamente en todas las resoluciones y en todos los dispositivos, como nadie o casi nadie en su casa tiene todos los dispositivos móviles y tablets, los de google nos han facilitado mucho las cosas.

La herramienta que utilizamos para comprobar todas las resoluciones es una incorporada en el navegador GOOGLE CHROME y se llama "Toggle device toolbar". Esta herramienta nos va a ofrecer poner la resolución que nosotros queramos, además de poder aplicar ZOOM e incluso tiene la resolución de algunos dispositivos, sobre todo los más populares:

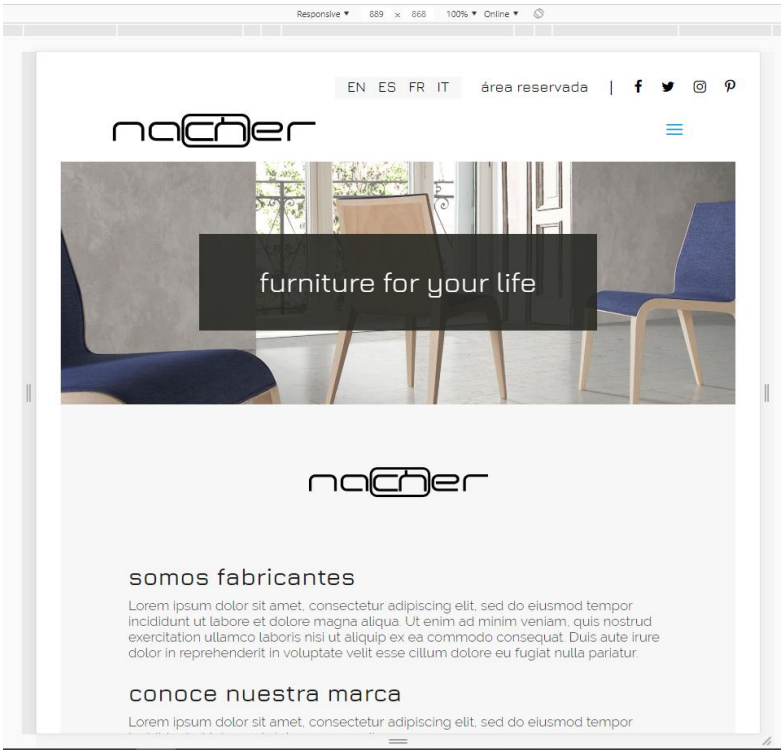

ILUSTRACIÓN 53 - EJEMPLO RESPONSIVE CON TOGGLE DEVICE TOOLBAR

Como podemos comprobar en la imagen anterior, actualmente todas las cajas encajan siguiendo una lógica y sin dañar la estructura del diseño. La imagen coincide con un ejemplo de versión tablet (889px de ancho).

14. Migración a entorno de pruebas controlado

El siguiente paso, una vez está la web finalizada, es subirlo a un entorno de desarrollo online, esto nos servirá para futuras modificaciones, las desarrollaremos en éste y una vez hecho lo pasaremos a producción, para que la web este el mínimo tiempo sin servicio mientras se realizan y aplican los cambios.

Para que no lo pueda visualizar el público la pondremos en modo mantenimiento, el cual solo podrá visualizarse si estás conectado como administrador de Wordpress.

El primer entorno en el cual hicimos el desarrollo era en local por lo que nadie, aparte de nosotros, podía visualizar la página web. Ahora el siguiente paso es una revisión general de la web de los miembros del equipo y el diseñador. En esta revisión buscamos todo tipo de fallos, tanto a nivel de diseño como a nivel de desarrollo. Esto se hace porque muchas veces uno mismo no ve sus propios fallos y cuatro ojos ven más que dos.

Una vez hemos realizado la revisión, el siguiente paso es pasarle accesos al cliente para que vea la web final (el cliente en principio solo había visualizado el diseño y lo había aceptado) por lo que ahora ya va a ver su web definitiva. En este punto los clientes también suelen pedir modificaciones o cambios y además es donde añade su contenido definitivo bajo nuestra supervisión.

Aparte, también aprovechamos para explicarle cómo gestionar su sitio web antes de su publicación definitiva.

Para realizar esta migración necesitaremos subir tanto la web como la base de datos a este servidor, aparte tendremos que modificar en la base de datos todas las “url” ya que actualmente nuestro sitio esta como “localhost” y a partir de ahora pasará a llamarse como el dominio definitivo “www.nacher.es” y también el archivo de configuración “wp-config.php” donde tendremos que modificar los parámetros de conexión a la base de datos.

En primer lugar vamos a preparar el servidor para poder subir nuestra web. Lo primero es crear el directorio “nacher” y crear una nueva base de datos. Una vez tenemos todo esto creado, modificaremos el archivo “wp-config.php” para que apunte a partir de ahora al servidor de desarrollo y no a nuestro localhost (los cuales por motivos de seguridad no podemos mostrar) y subiremos todos los archivos de la web a este directorio.

En segundo lugar modificaremos en la tabla “wp_options” de nuestra base de datos para decirle que la “url” va a ser la nombrada anteriormente:

Editar Copiar Borrar	1	siteurl	http://www.nacher.es	yes
Editar Copiar Borrar	2	home	http://www.nacher.es	yes

ILUSTRACIÓN 54 - MODIFICACIÓN BASE DE DATOS

Ahora, ya deberíamos visualizar la web correctamente, pero como la web no está en su dominio definitivo tendremos que modificar en nuestro ordenador el archivo "HOST" para que ese dominio apunte a la IP del servidor de desarrollo. Como actualmente el dominio definitivo ya está comprado las DNS apuntan a este dominio y nosotros queremos acceder al dominio de prueba.

Para editar nuestro archivo host tendremos que ir donde está alojado éste. El directorio es el siguiente: "C:\Windows\System32\drivers\etc". Una vez estemos en este directorio veremos un archivo llamado "hosts", lo abrimos con cualquier editor de texto (en modo administrador si no no deja guardarlo) y le indicamos que cuando en el navegador nos dirijamos al dominio definitivo, este tiene que apuntar a la ip asignada:

```
# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
[redacted]              nacher.es
                        www.nacher.es
```

ILUSTRACIÓN 55 - MODIFICACIÓN ARCHIVO HOST

Ahora sí que podemos acceder a nuestra web, aunque los enlaces internos (imágenes, menús, etc) no funcionarán correctamente ya que Wordpress utiliza enlaces absolutos y no relativos. En nuestro caso, en todo el desarrollo hecho por código utilizamos enlaces relativos, por lo que no debemos modificar nada.

Para modificar todos estos enlaces de nuestra página web utilizamos una herramienta llamada "SAR - SEARCH AND REPLACE". Es un archivo php que lo añadimos en la raíz de nuestro proyecto y accedemos a él por el navegador. Desde el navegador le asignamos la url vieja (la que queremos reemplazar) y la url actual (la que queremos que aparezca).

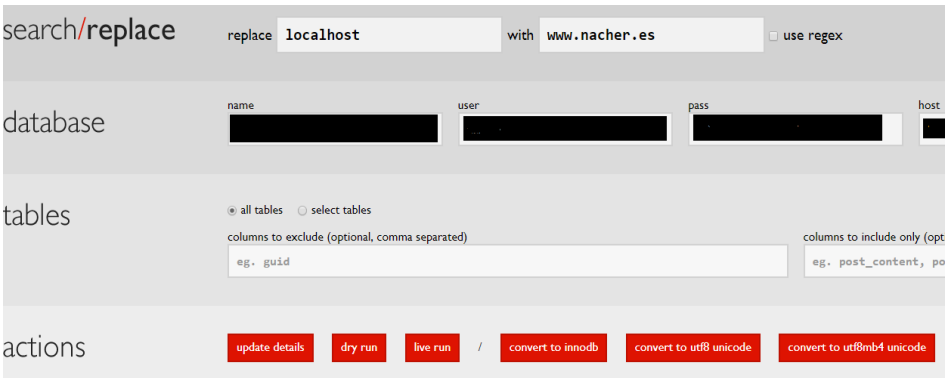

ILUSTRACIÓN 56 - PROGRAMA SAR

Una vez finalizado este proceso, debemos borrar el archivo por motivos de seguridad. Ahora ya tenemos nuestra web migrada y lista para empezar con la revisión.

15. Migración al servidor definitivo

Ahora el último paso es migrar nuestra web al servidor definitivo. Para ello debemos crear la suscripción al cliente en nuestro servidor. En nuestro caso los servidores se gestionan mediante el panel "Plesk". Rellenamos todos los datos del cliente y le asignamos la capacidad contratada.

ILUSTRACIÓN 57 - ALTA SUSCRIPCIÓN CLIENTE

Ahora le configuramos los accesos FTP, estos datos son los que utilizará el cliente ya que nosotros utilizamos los datos de administración con acceso a todas las suscripciones del servidor. Además, le vamos a incluir en su dominio un certificado de seguridad SSL para que la web sea segura.

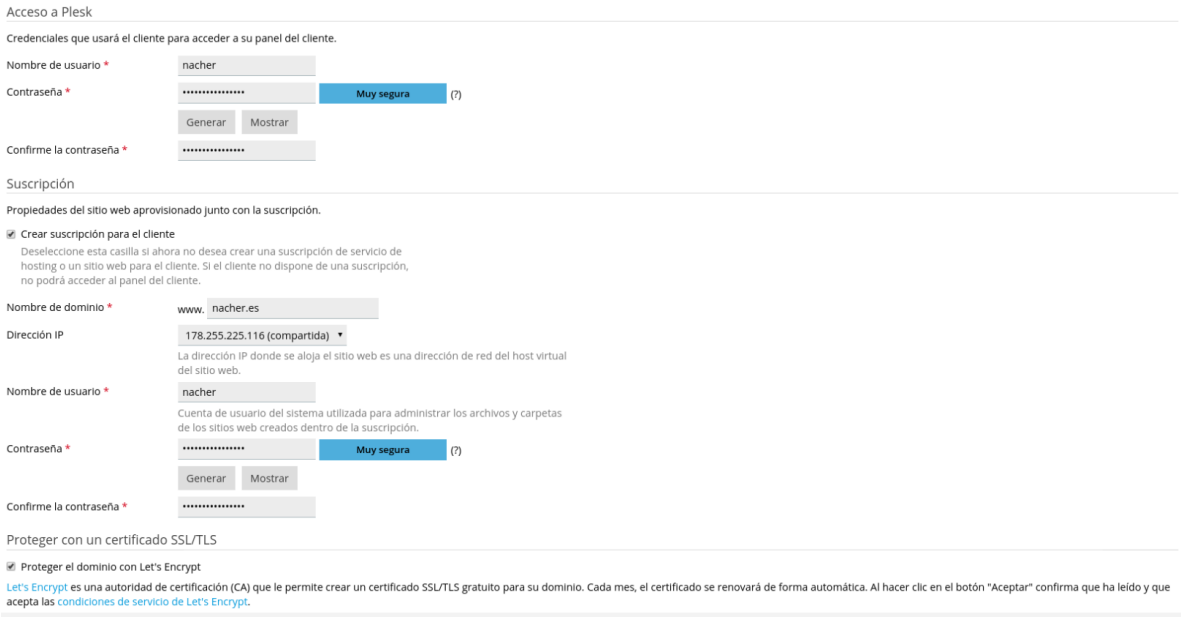

ILUSTRACIÓN 58 - CREACIÓN DE ACCESO FTP Y PLESK

Una vez finalizado el proceso de configuración, ya podremos gestionar nuestra suscripción, crear bases de datos, creaciones de correo, etc.

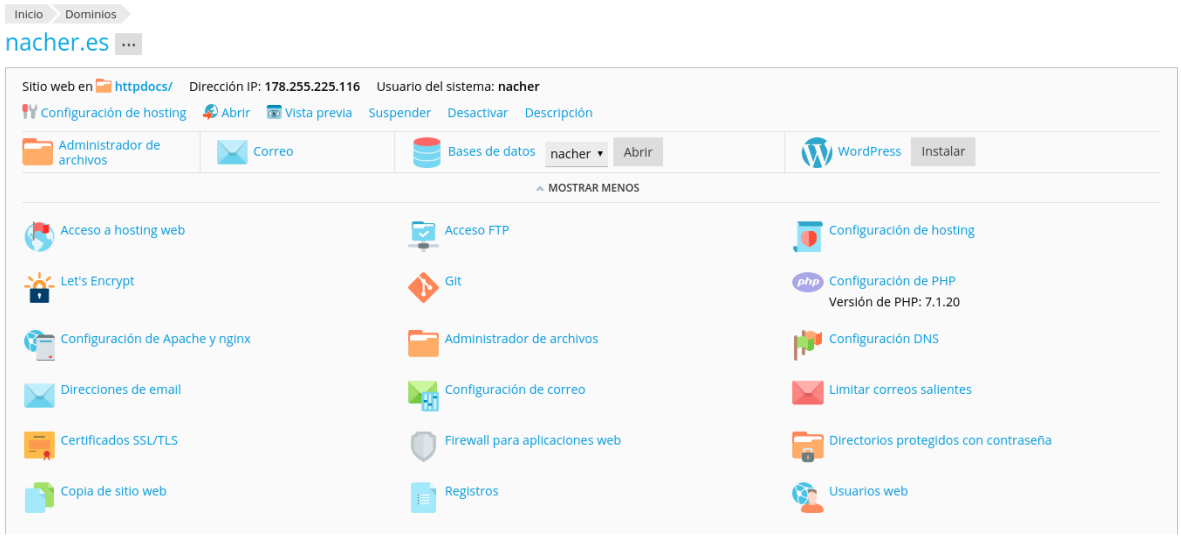

ILUSTRACIÓN 59 - PANEL PLESK SUSCRIPCIÓN

16. Creación de correos

Ahora vamos a dar de alta los correos electrónicos que nos han solicitado. Estos correos los podrán administrar mediante el webmail (vía navegador) o configurarlos en cualquier gestor de correo existente (outlook, thunderbird, etc.). Por lo general las configuraciones en gestores de correo las realizamos nosotros mediante control remoto (teamviewer).

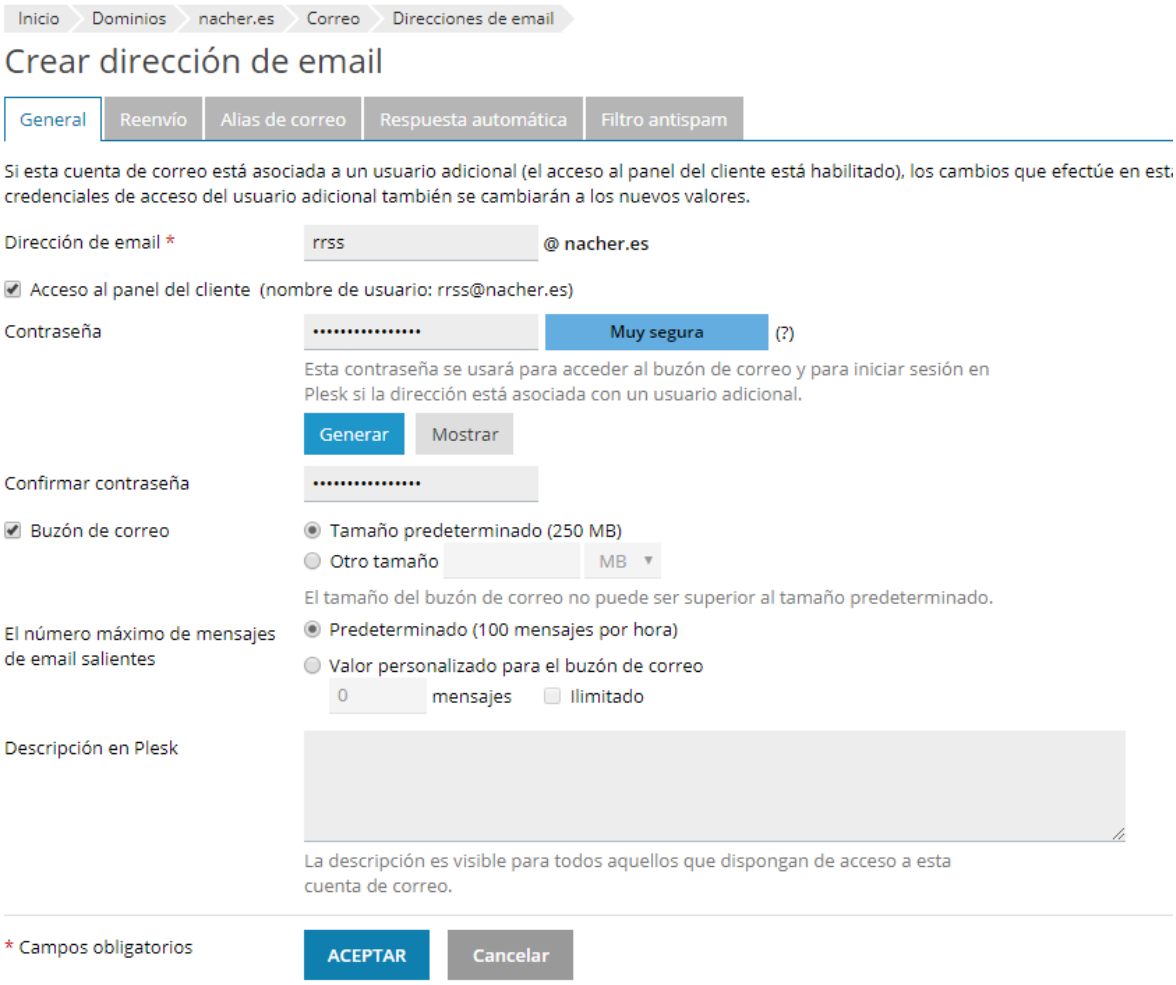

ILUSTRACIÓN 60 - CREACIÓN DE CORREOS ELECTRONICOS

16.1 Configuración correo externo

Cuando un cliente nos solicita la configuración desde una aplicación externa de gestión de correo debemos hacerla nosotros, ya que somos los que le brindamos el servicio de correo electrónico. Aparte por motivos de seguridad, nuestro servidor si detecta una mala

configuración o intentos de inicio de sesión fallidos (contraseña mal puesta) bloquea la IP, así podemos controlar nosotros el bloqueo.
Para ello el cliente nos envía los datos de acceso a control remoto, una vez estamos dentro le añadimos las cuentas de correos que nos solicita para cada equipo.

Para agregar las cuentas en las aplicaciones necesitaremos tener los datos de configuración SMTP, en nuestro caso los servidores son: "mail.dominio.x" y los configuramos con protocolo SSL (por seguridad). En el outlook cuando vamos a realizar la configuración debemos indicarle que queremos configurar manualmente los datos, la configuración la vamos a realizar por POP, únicamente dejaremos la copia de los correos en el servidor durante 14 días, esto se hace para tener un margen por posible pérdida de correos y para evitar que se llene la capacidad del buzón de correo y se suspenda el servicio.

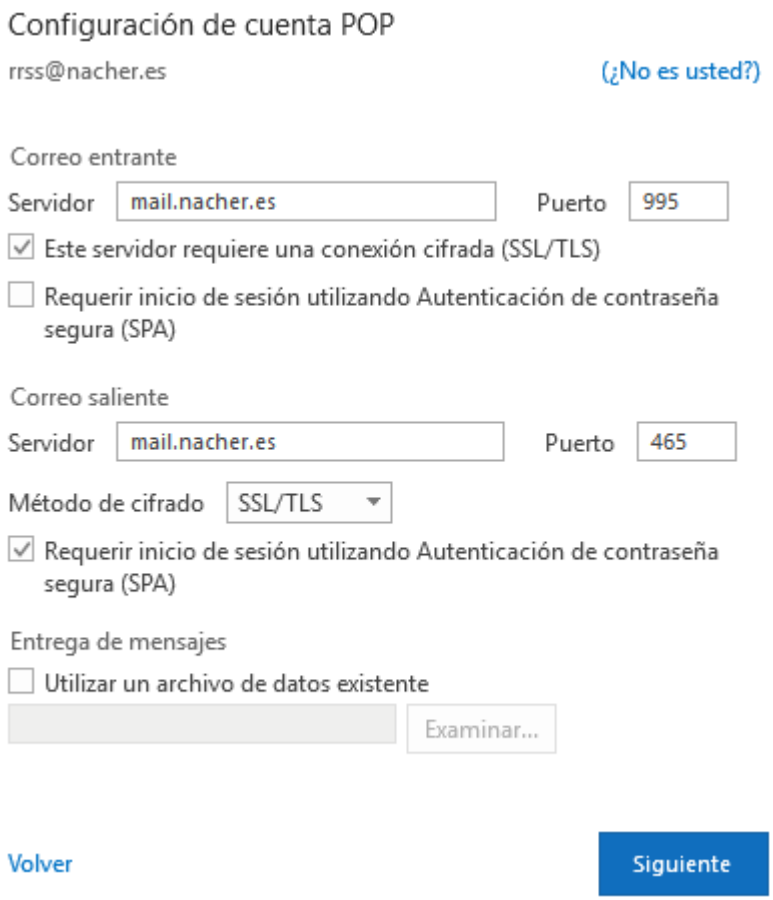

ILUSTRACIÓN 61- CONFIGURACIÓN OUTLOOK

Una vez configurado realizamos las comprobaciones necesarias (envío y recepción de un correo) para comprobar que funciona todo correctamente.

17. Repositorio Git

Ahora ya está la página web en el servidor definitivo y funcionando correctamente. Podemos decir que ya tenemos una versión inicial del sitio por lo que cerramos este proyecto para que el equipo de administración pueda gestionar todo el tema administrativo con el cliente.

Ahora, lo que debemos realizar es subir la carpeta de nuestro tema hijo a nuestro repositorio GIT. Git es un software de control de versiones, por lo que quedaran registrados todos los futuros cambios realizados en el proyecto y también quien los ha realizado. Esto sirve también para la actualización del css y scss ya que siempre vamos a tener el repositorio actualizado con los dos archivos.

Git se puede utilizar tanto por aplicación como por el terminal de comandos. Hay aplicaciones muy intuitivas, por lo que no es necesario conocimientos previos en el terminal.

En nuestro caso utilizamos la herramienta “gitlab.com” que es gratuita y muy completa.

Al ser desarrollos web “pequeños” no realizamos el seguimiento inicial hasta tener web lista. Esta herramienta supone un coste mínimo de tiempo y los beneficios que aporta son grandes, es prácticamente obligatorio su uso en grupos de trabajo.

18. Posibles mejoras (Rendimiento)

Con el proyecto inicial ya finalizado, vemos que tenemos la “web montada” visualmente, con todos sus productos, categorías, página de contacto, etc.

Pero una web requiere optimización para mejorar su carga y sobre todo para salir en las primeras posiciones de Google requiere que cumpla unos requisitos mínimos.

El primer paso que habría que realizar es la compresión de las imágenes sin pérdida de calidad, todas las imágenes deben estar en “jpg”, con la resolución máxima que se vaya a utilizar y guardadas para web (el Photoshop dispone de esta característica).

Una vez hemos reducido todas las imágenes de nuestra web, hay herramientas online que las reducen más sin perder calidad, un ejemplo de esta herramienta seria la web: “compressor.io”.

Con este paso ganaremos tiempo en la carga de nuestro servidor ya que es un dato muy importante y Google lo valora mucho.

Otra de las herramientas que nos ayudara con la optimización es el “Pagespeed” de Google. Herramienta gratuita y online. Esta herramienta hace una revisión de la web y nos saca una lista de los fallos de carga que tiene, aparte te da una valoración y puedes comprobar en tiempo real las mejoras de carga tras solucionar los fallos que nos ha encontrado.

Por último, deberíamos hacerle un análisis SEO, éste análisis consiste en solucionar todos los enlaces sin destino (con la herramienta Screamingfrog), ampliar los textos de búsqueda de la web, hacer redirecciones para evitar errores 404, etc.

19. Conclusiones

Éste TFG ha sido desarrollado en base a un proyecto realizado por mí en la empresa que actualmente trabajo. Consiste en la explicación de la metodología de trabajo utilizada en "acceseo". Este proyecto lo he desarrollado con la ilusión de que futuras incorporaciones puedan adaptarse con la mayor rapidez y de la forma más óptima posible a nuestro equipo de trabajo.

Aquí explico todas las herramientas y procedimientos que seguimos para desarrollar un sitio web personalizado y completo, en base a la experiencia generada y mis conocimientos técnicos, obtenidos tanto en la universidad como en el puesto que desempeño.

Como vemos día a día, una página web la puede desarrollar cualquier persona con pocos conocimientos, pero cuando vamos a desarrollar un sitio de esta envergadura y repercusión, no es suficiente realizarlo, si no también debemos plantearnos cual es la mejor forma y técnica.

Cuando yo empecé a desarrollar páginas no utilizaba las herramientas que utilizo ahora, ni tampoco tenía los conocimientos que he adquirido con el paso del tiempo. Poco a poco, con las necesidades que me iban surgiendo he ido indagando y buscando nuevas metodologías con la finalidad de facilitar mi desarrollo diario y mejorar el trabajo en equipo.

A lo largo de estos años trabajando, uno de los valores más grandes que he aprendido es realizar el desarrollo como me gustaría que lo hicieran si yo fuera él cliente. Es muy importante siempre escuchar al cliente en todo y guiarle en esta nueva aventura, ya que muchos de estos clientes no tienen experiencia en gestionar sus sitios web. Un punto muy importante es facilitarle las cosas al máximo, ya que en este caso concreto son vendedores de mobiliario y no tienen conocimientos previos en este campo.

20. Bibliografía

Toda la información utilizada para el proyecto la he obtenido de las siguientes páginas:

1. https://www.vagrantup.com/docs/index.html

2. http://www.conasa.es/blog/vagrant-la-herramienta-para-crear-entornos-de-desarrollo-reproducibles/

3. http://cmder.net/

4. https://raiolanetworks.es/blog/estructura-basica-archivos-carpetas-wordpress/

5. https://es.wikipedia.org/

6. https://owlcarousel2.github.io/OwlCarousel2/

7. https://es.wikipedia.org/wiki/Widget

8. https://codex.wordpress.org

FIN DEL PROYECTO