UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Tecnologías de la Información

TRABAJO FIN DE GRADO

CREACIÓN DE UN GESTOR DE CONTENIDO Y APLICACIÓN DE ESTE EN TIENDA SOLIDARIA

Ángel Fernández-Arroyo López-Manzanares

Julio, 2019

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Departamento de Lenguajes y Sistemas de Información

Tecnologías de la Información

TRABAJO FIN DE GRADO

CREACIÓN DE UN GESTOR DE CONTENIDO Y APLICACIÓN DE ESTE EN TIENDA SOLIDARIA

Autor(a): Ángel Fernández-Arroyo López-Manzanares

Director(a): José Jesús Castro Sanchez

Julio, 2019

Lurión. Gestor de contenido

Tribunal:

Presidente: _____

Vocal: _____

Secretario: _____

Fecha de defensa: _____

Calificación: _____

Presidente	Vocal	Secretario
Fdo.:	Fdo.:	Fdo.:

Resumen

El gran auge de las nuevas tecnologías, y con ello la aparición de la Sociedad 2.0 y el uso constante de Internet, en cualquier lugar y momento, gracias a los dispositivos móviles, supone un gran reto y a la vez oportunidad para las empresas, sean del sector que sean. Hoy en día, Internet y las nuevas tecnologías, permiten a las empresas acceder a una cantidad mayor de potenciales clientes, el área de posible influencia aumenta enormemente. Aquellas empresas que no están en la web, quienes no emplean internet en sus procesos de negocio están dejando de ser competitivos.

En España, el gran grueso del sector empresarial, lo constituyen las pequeñas y medianas empresas y sobre todo las microempresas que, o bien no se atreven a dar el salto al uso de nuevas tecnologías y a la creación de una línea de negocio en Internet, desplegando sus procesos en ella, o bien no tienen recursos para hacerlos (de personal preparado ni económico para afrontarlo). Esto, obviamente, supone una gran desventaja, ya que, hace unos años, el hecho de no promocionarse a través de Internet suponía que no se obtendrían nuevos clientes, sin embargo, cada vez más, está cambiando esta tendencia hasta prácticamente poderse afirmar que no mostrarse a través de Internet supondrá la pérdida de clientes.

Ante este escenario surge la iniciativa de crear una herramienta que ayude y facilite el proceso de transformación digital de las pequeñas y medianas empresas, sobre todo las microempresas. La herramienta en cuestión que crea es un Gestor de Contenido o CMS específico de comercio electrónico, pero usable en otros campos. La herramienta desarrollada permitirá crear una tienda online y/o página web de una forma mucho más sencilla, además de cómoda, económica, etc. y aunque ya existen varios CMS, el enfoque seguido en el diseño es distinto ofreciendo una gran libertad al usuario para modificar, tanto el aspecto, imagen corporativa de la web o sus funcionalidades.

Abstract

The great rise of new technologies, and with it the emergence of Society 2.0 and the constant use of the Internet, anywhere and anytime, thanks to mobile devices, is a great challenge and at the same time opportunity for companies, whatever sector they may be. Today, the use of Internet and new technologies allow companies to access a greater number of potential customers and their area of possible influence increases greatly. Those companies that are not on the web, who do not use the internet in their business processes, are sadly no longer competitive.

In Spain, the large bulk of the business sector is composed by small and medium-sized enterprises and especially micro-enterprises (less than 9 employees) that either do not dare to make the leap to the use of new technologies and to the creation of a new line of business on the Internet (deploying their processes in it) or they do not have the resources to do them (staff or resources not prepared to deal with it). This, obviously, is a major disadvantage, since, a few years ago, the fact of not existing over the Internet meant that new customers would not be obtained, however, this trend is changing until it can practically be said that not existing over the Internet will result in the loss of customers.

Against this background, the initiative arises to create a tool that helps and facilitates the process of digital transformation to small and medium-sized enterprises, especially micro-enterprises. The tool to create is a Content Manager or CMS focused on e-commerce, but usable in other fields. The developed tool will allow anyone to create an online store and / or website in a much simpler, comfortable and cheaper way, and although there are already several CMS, the focus followed on the design is different offering a great freedom to the user to modify the appearance, corporate image of the website or its functionalities.

AGRADECIMIENTOS

Es muy difícil determinar a quién debo nombrar en estos agradecimientos, puesto que, a lo largo de este trabajo hay mucha gente que me ha ayudado, ya sea en un momento concreto o a lo largo de todo este proceso. Es por ello por lo que añado esta introducción para agradecer, al menos de forma genérica, a todos ellos, que en mayor o menor medida han hecho posible este proyecto. Sin embargo sí que hay ciertas personas que deben ser nombradas personalmente:

- A mi tutor D. José Jesús Castro, que me ha apoyado y se ha preocupado por mi, no solo profesional sino también personalmente y no solo durante este proyecto sino desde que le conozco.
- A los miembros de la Universidad de Castilla-La Mancha por confiar en mi proyecto para crear una tienda solidaria con su sello.
- A la academia Anna Sampermat por permitirme crearle una solución web con la herramienta a crear.
- A una de las personas que más ha contribuido en mi proceso de aprendizaje durante la carrera, a la que debo más que un pin.
- A mis compañeros de piso a lo largo de estos últimos años que, durante este tiempo, me han alegrado en muchos momentos.
- A mis amigos de toda la vida que me han permitido relajarme en más de una ocasión tomando un café.
- A mis padres que me han permitido centrarme por completo en este trabajo y dejar en segundo lugar ciertas preocupaciones que tendré que afrontar de ahora en adelante.

Ángel Fernández-Arroyo López-Manzanares

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE LISTADOS

CAPÍTULO 1

INTRODUCCIÓN

Cualquier empresa para poder competir en el mundo actual necesita ser conocida y llegar al mayor número posible de potenciales clientes, permitiéndoles acceder a sus productos/servicios en cualquier momento y en cualquier lugar. Es por esto que tiene una gran importancia estar donde están los posibles clientes, y este lugar no es otro que Internet. En este contexto, la creación de una página web para llevar a cabo actividades de comercio electrónico o darse a conocer es primordial. Sin embargo, no todas las empresas están en disposición de poder tenerla, ya que se necesita personal en la empresa que cuente con conocimientos tecnológicos, y las pequeñas empresas no suelen tener departamentos ni trabajadores que puedan asumir este trabajo. Así, se hace necesario subcontratar esta labor o buscar otros mecanismos para poder comercializar sus productos en la web.

1.1 TRANSFORMACIÓN DIGITAL

En los últimos años, en todo el mundo, se está viviendo un cambio que se ha denominado **Transformación Digital**, que implica la integración y explotación de las nuevas tecnologías en todas las áreas de una empresa y en todos sus procesos, con el objetivo de mejorar así su eficiencia y competitividad, ofrecer un mejor servicio o valor al cliente, descubrir nuevas oportunidades de negocio y proporcionar capacidad de respuesta rápida en un entorno cambiante.

Este proceso es uno de los mayores desafíos a los que se enfrentan las empresas hoy en día, ya que implica:

- Abandonar la zona de confort

  Existe cierto miedo o inseguridad a la hora de cambiar los procesos ya establecidos, ya que una decisión mal tomada en un ámbito tan importante y global como es la transformación digital podría suponer pérdidas de dinero, tiempo… Sin embargo, esta transformación no tiene por qué ser un proceso caótico con las herramientas adecuadas, además, no hacer frente a este cambio supone perder terreno ante empresas que sí han apostado por él.

- Fomentar las competencias digitales de todos los miembros de la empresa

  Por mínimos que sean, para realizar este cambio se necesitan conocimientos relacionados con las nuevas tecnologías, lo que supone una formación y esfuerzo por parte de los empleados para aprender dichas competencias.

- Repensar el negocio

  Al realizar la transformación digital, no suele funcionar el reproducir los procesos ya existentes en la empresa en un entorno informático, lo recomendable es que se aproveche este momento para modificar procesos o incluir en la empresa las relaciones necesarias

para que los distintos departamentos cooperen, compartiendo así datos e información. Hay que poner en marcha nuevas estrategias.

- Controlar el cambio

  Este control es necesario para medir el estado de la empresa en el nuevo entorno digital. Además, puesto que la estrategia digital puede implicar la modificación de procesos y la cooperación de varios departamentos, este control puede medir y evaluar la cohesión de estos.

Debido a estos desafíos, entre otros, **ninguna empresa, independientemente del sector al que pertenezca, puede sortear los efectos de la tecnología** y dar la espalda a la transformación digital, ya que, el potencial de estas tecnologías digitales que cambian el mercado es a menudo mayor que la propia actividad principal de la empresa, por lo que, para competir, todas deben adaptarse a esta nueva realidad.

Actualmente la mayoría de empresas, sobre todo las pequeñas y medianas (conocidas como PYMEs), eluden estos desafíos y rechazan el cambio, prefieren mantener su actual funcionamiento al ver que esta transformación afectaría a todos y cada uno de sus ámbitos, además no se consideran capacitadas para afrontarlos con los recursos de los que disponen. Es por este último motivo por lo que han aparecido empresas en el sector tecnológico cuyo producto o servicio consiste en facilitar este proceso. Obviamente esto supone unos costes económicos y un esfuerzo que no todas las PYMEs están dispuestos a aceptar a la ligera. Así, las empresas comienzan a implantar las nuevas tecnologías en distintos niveles. Véase la figura 1.1.

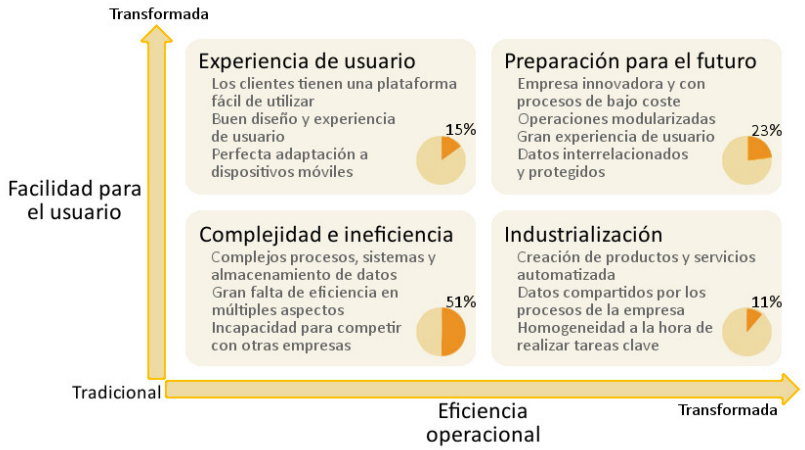


Figura 1.1: Estimación porcentual de empresas según su nivel tecnológico y operacional

En el contexto actual, la transformación digital no es una opción, es una obligación. **Las empresas, independientemente de su tamaño**: grandes, medianas y pequeñas; **deben afrontar este cambio para poder ser una empresa de los "nuevos tiempos"**, para poder ser competitiva en el entorno actual, en una sociedad cada vez más tecnificada, con personas conectadas a Internet más de 5 horas al día y con una capacidad de conexión desde cualquier parte o en cualquier momento debido a los dispositivos móviles y las distintas redes de conexión.

Es por esto, por lo que la transformación digital de una empresa debe también centrarse en mejorar la comercialización de sus productos y servicios en Internet, permitiendo a sus potenciales clientes el acceso a estos desde cualquier parte y en cualquier momento. La comercialización de

estos productos y servicios en Internet implica tener una web que permita llevar a cabo actividades de comercio electrónico. La web deberá ser accesible desde cualquier dispositivo y tener un diseño atractivo que haga uso de la imagen corporativa de la empresa y que sea de uso sencillo en el front-end (aquello que ve el usuario y que puede visitar), permitiendo su gestión desde un back-end completo (aquello que ve el dueño del negocio). Además la web sirve para crear sobre la empresa una imagen innovadora y para transmitir información sobre la misma, sus objetivos, sus valores, su forma de trabajar…

En este Trabajo Fin de Grado **nos centramos en aquella parte de la transformación digital que se enfoca en la mejora del proceso de comercialización de los productos y/o servicios** empleando las nuevas tecnologías e Internet. Se busca obtener un sistema que permita a las PYMEs la creación, de manera fácil, de webs de comercio electrónico, con un front-end para que cualquier cliente potencial desde cualquier lugar pueda ver lo que la empresa ofrece (i.e. catálogo de productos e información), y un back-end, que le permita tener un control de inventario, pedidos, gestión de productos, servicios, usuarios…

1.2 PÁGINA WEB EMPRESARIAL

Si un negocio aún no está en Internet estará perdiendo clientes, ya que el principal beneficio obtenido a través de Internet es que el alcance que conseguirá la empresa en cuestión se ampliará en gran medida, a priori **cualquier persona en cualquier lugar y en cualquier momento** con un navegador **podrá acceder a su web para conocer su información y los productos y servicios que ofrece**. Esto supone más clientes potenciales, pero también más competencia. Hoy en día, no tener web supone que los clientes se limiten a la población próxima a la empresa o clientes ya fidelizados, pero con un mayor número de nuevos competidores, tengamos en cuenta que cada vez más gente realiza compras a través de Internet por las ventajas que esta le aporta. Véase la figura 1.2.

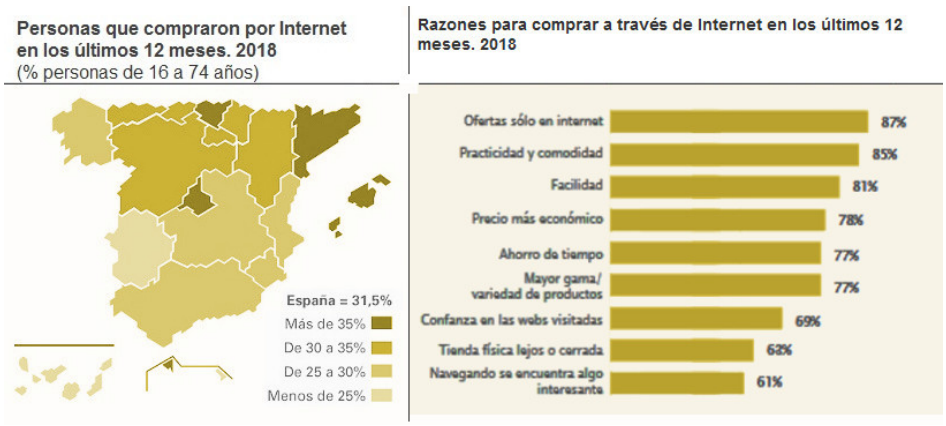


Figura 1.2: Razones y cantidad de compras online en España, 2018

La necesidad de estar en Internet para cualquier empresa que quiera promocionarse o vender algún producto/servicio, es evidente en el contexto actual. La pregunta que toda PYME deberá hacerse es ¿cómo creo mi página empresarial? Como ya se ha comentado anteriormente, el diseño y desarrollo de una página web de este tipo requiere de una serie de conocimientos y habilidades en procedimientos tecnológicos avanzados, de las que no todos disponen. Las grandes empresas cuentan con departamentos de informática que puedan afrontarlo. Pero ¿Cómo es el tejido empresarial de España que debe hacer esta transformación digital?

En España, desde hace unos años y aun hoy en día, **la mayor cantidad de empresas que hacen frente a la transformación digital**, son empresas con un número de asalariados muy reducido y

que por tanto **no disponen de personas con los conocimientos tecnológicos necesarios** para llevar a cabo este proceso (salvo escasas excepciones). Véase la figura 1.3.

Nº asalariados	Empresas	Porcentaje sobre el total	Porcentaje acumulado
Sin asalariados	1.845.881	55,30%	55,30%
De 1 a 2 asalariados	910.686	27,29%	82,59%
De 3 a 5 asalariados	303.574	9,10%	91,69%
De 6 a 9 asalariados	125.173	3,75%	95,44%
De 10 a 19 asalariados	80.860	2,42%	97,86%
De 20 a 49 asalariados	45.485	1,36%	99,22%
De 50 a 99 asalariados	13.116	0,39%	99,61%
De 100 a 199 asalariados	7.033	0,21%	99,82%
De 200 a 499 asalariados	3.925	0,12%	99,94%
De 500 a 999 asalariados	1044	0,03%	99,97%
De 1000 a 4999 asalariados	750	0,02%	99,99%
5000 o más asalariados	119	0,01%	100,00%
Total	3.337.646		

Fuente: DIRCE 2018

Figura 1.3: Empresas por estrato de asalariados, 2018

Como se aprecia en la Figura 1.3, más del 82 % de las empresas tienen menos de tres empleados, esto implica que estos trabajadores están formados en la actividad principal de la misma y no existen departamentos como tal. Además **las microempresas (menos de 9 asalariados) representan casi el 96 % de las empresas**, prácticamente el total, por lo que se puede presuponer que en la gran mayoría de ellas no existe un encargado de nuevas tecnologías o temas relacionados con la informática salvo que esta se dedique a ello (el menor de los casos). Véase la figura 1.4.

Nº	Nombre de la agrupación	Total microempresas	% de microempresas
1	Industria	143.057	4,49%
2	Construcción	412.523	12,95%
3	Venta y reparación de vehículos a motor	78.112	2,45%
4	Comercio mayorista	221.018	6,94%
5	Comercio minorista	451.472	14,17%
6	Hoteles campings y agencias de viaje	42.599	1,34%
7	Transporte y almacenamiento	186.204	5,85%
8	Informática y Telecomunicaciones	39.831	1,25%
9	Actividades inmobiliarias y administrativas	439.281	13,79%
10	Actividades profesionales	337.431	10,59%
Total de sectores abarcados por la encuesta		2.351.528	73,82%
Resto de microempresas		833.786	26,18%
Total de microempresas Españolas		3.185.314	100%

Fuente: DIRCE 2018

Figura 1.4: Agrupación sectorial de microempresas en España, 2018

1.2.1 Creación de la Página web empresarial

Una vez descartada la creación por cuenta propia de la página web empresarial, es decir con recursos humanos y tecnológicos propios, la distintas alternativas que tiene una empresa, una PYME, para crear una web empresarial que permita llevar a cabo actividades de comercio electrónico son las siguientes [1]:

1. *Solución Software como Servicio (Software as a Service) – SaaS.* Existen soluciones en la nube que proporcionan una colección de herramientas y recursos para crear y gestionar una tienda online,

procesar los pedidos y vender productos online, además de otros servicios. Estas soluciones tienen un coste asociado que depende del uso o tipo de servicios subscritos (existen diferentes planes). En este tipo de soluciones se sigue la máxima ”Céntrate en tu negocio, no te preocupes de la tecnología ni de los procesos, de eso se encargará el proveedor del servicio”. Algunos ejemplos de plataformas SaaS que ofrecen estos servicios son: Bigcommerce, Volusion, Shopify, ekmtienda, Palbin, Tiendy, y muchas más que aparecerán si se busca un poco por Internet. Todas ellas tienen en común que ofrecen una tienda estándar y nos proporcionan un usuario y contraseña para poder acceder a un panel de administración, a través del cual podremos personalizar la tienda con la plantilla que queramos, añadir nuestros datos, e instanciarla con nuestro catálogo.

2. *Gestores de Contenidos –CMS, específicos de comercio electrónico o generales, que se especializan para este propósito por medio de plantillas, plugins o extensiones.* Un Sistema de Gestión de Contenidos es una aplicación de software que se utiliza para crear, editar, administrar y publicar contenido de forma coherente y eficaz en la web. Debido a la relación con el producto que se propone en este Trabajo Fin de Grado, se dedicará una sección en este capítulo para hablar de ellos.

3. *Soluciones DIY*. Una tienda online es una aplicación web y como tal también puede ser diseñada y desarrollada a medida desde cero, esto es lo que se conoce como “hazlo tú mismo” o en inglés Do It Yourself (DIY). Esta aproximación implica un diseño y desarrollo a medida por parte del departamento de desarrolladores software (informáticos, diseñadores gráficos, especialistas en marketing) o por una empresa que se dedique a este tipo de desarrollos (existiendo unos contactos en la propia empresa que sirven de intermediarios y son los que establecen los requisitos y proporcionan el feedback sobre los desarrollos que vayan realizando).

El orden en el que se han mencionado las distintas alternativas no es casual, se ha hecho de menor a mayor complejidad. Lo más fácil para una PYME a la hora de tener una web empresarial es la alternativa SaaS, ya esta alternativa no requiere de muchos conocimientos de las tecnologías y permite implementar una tienda rápidamente y sin apenas costes de infraestructura. Estos proveedores de servicios suelen disponer de una alta escalabilidad, es decir, se puede acceder a un mayor número de recursos en caso de que sean necesarios. No obstante, las empresas que se decantan por esta opción, también deben saber que **todos los datos serán almacenados y gestionados por los proveedores del servicio, es decir, por terceros, y que perderán el control sobre estos; al igual que la disponibilidad del servicio, que dependerá de las capacidades e infraestructura del proveedor**.

La más compleja es la alternativa DIY que es la que requiere mayor cantidad de conocimientos y esfuerzos tanto personales como económicos por parte de la empresa, pero también es la que ofrece mayor libertad y control. Como ya se ha comentado, no es lo usual que las PYMEs tengan personal que pueda desarrollar la web. Así mismo, **las empresas cuya labor se basa en Informática y Telecomunicaciones suponen tan solo el 1.25 % del total de las microempresas**. Por supuesto no todas ellas se dedican a tareas relacionadas con la transformación digital ni con la creación de soluciones de comercio electrónico.

En un lugar intermedio se encuentra la alternativa que se basa en el uso de CMS, destacar que incluso hay empresas que los usan para crear webs que venden a terceros. Esta alternativa requiere de más conocimiento tecnológico por parte del personal de la empresa pero les da más control sobre sus datos.

Las PYMEs suelen emplear las dos primeras alternativas (soluciones SaaS y CMS), y la última es la usual en las grandes empresas (soluciones DIY).

1.3 SISTEMAS GESTORES DE CONTENIDO

Un Gestor de Contenido (Content Manager System / CMS) es una **herramienta informática compleja que se encarga del almacenamiento, gestión y representación del contenido y la información de un sistema web, así como su posterior mantenimiento**. Un CMS podría definirse como una plataforma software que facilita la implementación de una página web o una solución de comercio electrónico de forma rápida y sencilla. O en otras palabras, es una herramienta que permite la generación automática de los elementos visuales (HTML, CSS, etc.) que componen el front-end de una web, así como el funcionamiento interno o back-end de la misma.

Este concepto surge ante la necesidad de facilitar el trabajo a creadores de contenido web, ya que el desarrollo y publicación de estos contenidos de forma manual a través de un servidor implica tener unos conocimientos técnicos y un tiempo del que no todo el mundo dispone, además que supone un trabajo algo tedioso.

Los Gestores de Contenido además de facilitar la personalización de una página web, aceleran el proceso de actualización del contenido y permiten que estas operaciones se puedan realizar por un usuario sin conocimientos técnicos.

Como puede intuirse, aunque los CMS estén pensados para crear automáticamente páginas, hay muchos tipos y cada uno tiene sus peculiaridades y necesidades de conocimientos tecnológicos para manejarlos. Véase la figura 1.5.

Figura 1.5: Distintos tipos de Gestores de Contenido

A la hora de relacionarlos con el comercio electrónico se puede decir que exiten dos tipos de CMS, los generales y los específicos de comercio electrónico. A continuación se presentan cada uno de ellos:

- *CMS generales instanciados para actividades de comercio electrónico*: son CMS que se personalizan para crear tiendas online a través de extensiones, plugins y plantillas específicas para este tipo de aplicaciones, permitiendo llevar a cabo las funcionalidades esenciales que debe ofrecer una tienda online (gestión de catálogo, carrito de la compra, integración con medios de pago,...). Estas extensiones pueden ser configuradas desde la parte privada de la web (backend) por el administrador de la tienda, estableciendo su apariencia y funcionalidad en la parte pública (frontend). En la actualidad existen muchos CMS de este tipo, siendo los más utilizados los siguientes: WordPress, Drupal y Joomla. Estos CMS han sido desarrollados con componentes, tecnología y software libre y código abierto, por lo que son gratuitos y existen comunidades

de desarrolladores y usuarios que colaboran y contribuyen a su desarrollo y mantenimiento. VirtueMart, JoomShopping y HikaShop son extensiones de Joomla para la creación de una solución de comercio electrónico en dicha plataforma; WooCommerce, JigoShop o WP-Ecommerce son plugins de Wordpress para ecommerce y Ubercart es uno de los proyectos de comercio electrónico para Drupal, otro de la misma plataforma es Drupal Commerce.

- *CMS específicos de comercio electrónico*: son CMS orientados a la creación y mantenimiento de tiendas online, dando soporte a los negocios que quieren implantar un canal de venta por Internet. Como cualquier CMS presenta un frontend, que en este caso, es la parte donde los clientes pueden navegar por la tienda, ver los productos y si lo desean comprarlos, y una zona de administración, que es el backend, para permitir la introducción de la información del negocio, gestionar el catálogo de productos, sus precios, ofertas, el stock, los proveedores…Los CMS específicos de comercio electrónico más utilizados en la actualidad son: Magento y Prestashop. Desarrollados con tecnología open source, también son gratuitos en sus versiones más básicas, existiendo comunidades de desarrolladores y usuarios que colaboran y contribuyen a su desarrollo y mantenimiento o al de las extensiones de los mismos.

El uso de CMS requiere de ciertos conocimientos tecnológicos para poder instanciarlos o personalizarlos para cada empresa concreta, además la adaptación a la imagen de la empresa o inclusión de funcionalidades no estándar o más avanzadas requieren de su programación o compra en las plataformas de los CMS. Esto echa para atrás a muchas PYMEs que tienen miedo a enfrentarse a la instanciación del CMS.

1.4 PROBLEMÁTICA ACTUAL

Recopilando los datos obtenidos a lo largo de esta introducción puede verse que hay un nicho de mercado, compuesto por aquellas **PYMEs que no tienen una página web empresarial propia con la que poder realizar actividades de comercio electrónico**, y que desean crear una página web empresarial que ellos controlen, pero no poseen los conocimientos tecnológicos necesarios para manejar los CMS disponibles.

Ante este escenario **se propone la creación de un gestor de contenido eficiente y eficaz en el que, en un panel de administración lo más simple posible y siguiendo metodologías intuitivas, se puedan modificar todos los aspectos necesarios para la creación de una página web con su correspondiente panel de administración**. Por supuesto, este gestor no será restrictivo, es decir, personas con conocimientos más avanzados podrán modificar y adaptar su solución tanto como deseen.

CAPÍTULO 2

OBJETIVO

2.1 OBJETIVOS DEL TFG Y COMPETENCIAS DE TRABAJO

En este capítulo se presentan los objetivos que se persiguen en este Trabajo Fin de Grado (TFG), detallando tanto los objetivos propios del trabajo que se desarrollará como los objetivos formativos, esto es, aquellos orientados a completar la formación del estudiante y autor de este TFG.

2.1.1 Objetivos generales y específicos

Este trabajo tendrá una serie de objetivos que se deben alcanzar al finalizar, pudiendo distinguir en ellos un objetivo general y varios específicos. El objetivo general surge como una idea de emprendimiento, un proyecto de negocio que surge en vista del nicho de mercado que podría estar interesado en el producto desarrollado. A continuación se detalla cada uno de ellos:

Objetivo general

OG1. Diseñar y desarrollar un Sistema Gestor de Contenidos que permita la creación de páginas web empresariales de cualquier tipo, pero con capacidad para la creación de tiendas online.

Objetivos específicos

OS1. Crear una web para la Universidad de Castilla-La Mancha que permita la venta online, dentro de la comunidad universitaria, de productos de segunda mano que estén incluidos en un catálogo.

OS2. Crear una web empresarial para una microempresa que ofrezca servicios.

OS3. Crear una web para dar soporte a la idea emprendedora, permitiendo la comercialización del gestor de contenido obtenido como resultado del OG1, para ello se empleará el propio gestor.

2.1.2 Objetivos formativos

Objetivos formativos principales

TI3. *Adquirir capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.*

TI6. *Adquirir capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.*

Objetivos formativos secundarios

TI5. *Adquirir capacidad para seleccionar, desplegar, integrar y gestionar sistemas de información que satisfagan las necesidades de la organización, con los criterios de coste y calidad identificados.*

TI7. *Adquirir capacidad para comprender, aplicar y gestionar la garantía y seguridad de los sistemas informáticos.*

Objetivos formativos específicos

FE1. Aprender en profundidad un lenguaje de programación específico para servidores web (PHP).

FE2. Aprender en profundidad un sistema de gestión de bases de datos relacional (MySQL).

- FE2-a. Diseñar e implementar una base de datos relacional.

FE3. Aprender lenguajes y tecnologías específicas para clientes web.

- FE3-a. Lenguaje de hojas de estilo en cascada (CSS3).
- FE3-b. Lenguaje de marcas de hipertexto (HTML5).
- FE3-c. Lenguaje de programación interpretado en el cliente (JS).

FE4. Aprender lenguajes de marcado de propósito general (XML).

FE5. Aprender técnicas de posicionamiento de las páginas web en los buscadores (.htaccess, robots.txt, pagespeed, google analytics).

CAPÍTULO 3

TECNOLOGÍA, PLAN DE TRABAJO Y METODOLOGÍA

En este capítulo se detalla la tecnología que se usará en este Trabajo Fin de Grado, tanto el software para desarrollarlo o ponerlo en explotación, como el hardware necesario para usar la herramienta o sobre el que esta se usará. También se detallará el Plan de Trabajo que organiza las tareas que hay que llevar a cabo para alcanzar los objetivos, y la metodología que se seguirá.

3.1 TECNOLOGÍA

A continuación se definirán los componentes tanto hardware como software utilizados a lo largo de todo el desarrollo de la herramienta y en el despliegue de la solución a crear.

3.1.1 Hardware

En este apartado hablaremos del contexto en que el trabajo ha sido desarrollado y utilizado/desplegado en modo de pruebas. Hay que tener en cuenta que, aunque se pretende crear un producto desde cero, sin dependencia alguna con elementos de terceros, las necesidades computacionales, de procesamiento, etc. no son especialmente elevadas, por ello el equipo que se ha utilizado es un único dispositivo estándar con las siguientes características.

Tabla 3.1: Detalles del equipo utilizado en el desarrollo

Características hardware	
Datos básicos	
Marca	Asus
Modelo	R510V
Detalles	
Procesador	Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz 2.59 GHz
Memoria instalada (RAM)	8’00GB (7’89 GB utilizable)
Tipo de sistema	Sistema operativo de 64bits, procesador x64
Monitor	
Resolución máxima	1366x768px

Para ponerlo en explotación no existen grandes limitaciones, simplemente un servidor web como podría ser Apache 2.x, Nginx o Microsoft IIS.

3.1.2 Software

Las características software también son básicas, ya que, como se dijo previamente, todos los recursos software que se usarán en Lurión serán desarrollados desde cero.

Recursos software

- **XAMPP:** XAMPP es un software multiplataforma y libre que surge como una distribución de Apache HTTP Server y que contiene el servidor de bases de datos MariaDB así como todas las herramientas necesarias para usar y compilar los lenguajes de programación PHP y Perl. El nombre es un acrónimo del software mencionado anteriormente (Multiplataforma (X), Apache, MariaDB, PHP, Perl). Es utilizado para crear un servidor local donde testear y ver todos los resultados de un desarrollo web, evitando así la necesidad de un servidor real
- **Sublime Text:** Sublime Text es un editor de código fuente de cualquier tipo, multiplataforma, desarrollado con C++ y una API de Python (interfaz de programación de aplicaciones). Esta plataforma es muy ligera, sin embargo, tiene múltiples funcionalidades. Las características más visuales y valoradas son, por ejemplo, identificar rápidamente para cada uno de los lenguajes que tipo de elemento se está utilizando (mediante códigos de color) o analizar clases y librerías para conocer las distintas funciones, métodos, clases, etc. y facilitar el trabajo mediante su utilidad de auto-completado.
- **MySQL**: MySQL es un sistema basado en un modelo cliente-servidor de gestión de bases de datos relacionales de código abierto respaldado por Oracle, basado en lenguaje de consulta estructurado (SQL). MySQL se ejecuta en prácticamente todas las plataformas y, aunque puede usarse en una amplia gama de aplicaciones, la mayoría de las veces, MySQL se asocia con aplicaciones web y publicaciones en línea.
- **PHPMyAdmin:** phpMyAdmin es una herramienta software gratuita escrita en PHP, diseñada para administrar bases de datos relacionales basadas en MySQL a través de un navegador. Permite realizar operaciones de uso frecuente (gestión de bases de datos, tablas, columnas, relaciones, índices, usuarios, permisos, etc.) a través de la interfaz de usuario, aunque también tiene la capacidad de ejecutar directamente cualquier instrucción SQL.
- **MySQL Workbench:** MySQL Workbench es una herramienta que puede ser utilizada en local y tiene una interfaz visual unificada para crear arquitecturas de bases de datos MySQL. Además proporciona funcionalidades cómo el modelado de datos, desarrollo de SQL y herramientas de administración integrales para la configuración del servidor, la administración de usuarios o las copias de respaldo.
- **Gimp:** GIMP (siglas en inglés de GNU Image Manipulation Program) es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías que dispone de herramientas para el retoque y edición de imágenes comparables a otras aplicaciones como Photoshop.
- **Filezilla:** Filezilla es software libre para la gestión de archivos desde un computador local a un servidor externo a través de los protocolos de red FTP, SFTP y FTPS.

Lenguajes

- **PHP**: PHP es un lenguaje de programación interpretado de libre uso utilizado principalmente en servidores web de Linux. PHP se ejecuta en el servidor, mientras que una alternativa comparable, JavaScript, se ejecuta en el cliente. PHP es una alternativa a la tecnología Active

Server Page (ASP) de Microsoft. Al igual que con ASP, el código PHP está incrustado dentro de una página web junto con su HTML. Antes de que la página se envíe a un usuario que la haya solicitado, el servidor web llama a PHP para interpretar y realizar las operaciones requeridas en el script de PHP, devolviendo a la llamada el código HTML final que será la página.

- **HTML5**: HTML (Hypertext Markup Language) es un lenguaje de marcas que define cómo se mostrará el contenido en el navegador de un cliente web. Define a través de sus etiquetas elementos como el texto, títulos, imágenes, tablas y cualquier otro tipo de elemento multimedia. HTML es una recomendación formal del World Wide Web Consortium (W3C) y generalmente es respetada por todos los principales navegadores web, tanto de escritorio como móviles. HTML5 es la última versión de la especificación.

- **CSS3**: CSS es un mecanismo estándar y el preferido para dar formato a páginas HTML. De acuerdo con la separación de los patrones de diseño y prácticas de buen uso, las hojas de estilo en cascada proporcionan unas reglas genéricas con las que un estilo definido puede y debe aplicarse a varios elementos HTML dentro de una página web (evitando así repeticiones de código). Las hojas de estilo en cascada también pueden controlar cómo se ubican en la página sus elementos (siempre teniendo en cuenta las restricciones del HTML). Esto es extremadamente útil cuando el contenido debe distribuirse de una manera dramáticamente diferente dependiendo de si se está viendo en un ordenador de escritorio, tablet o teléfono móvil.

- **Javascript**: JavaScript es un lenguaje de programación utilizado principalmente para sistemas web, ejecutado en el navegador del cliente. Tiene múltiples opciones, ya que permite crear efectos atractivos y dinámicos en las páginas web e incluso modificar y añadir código HTML y CSS y además permite llamar a funciones propias en el cliente o llamadas al código PHP del servidor mediante llamadas a archivos o funciones (como por ejemplo con peticiones asíncronas AJAX).

- **JQuery**: JQuery es una biblioteca de código abierto de JavaScript que simplifica la manipulación del documento HTML (DOM), el manejo de eventos, llamadas JavaScript asíncronas y manejo de archivos XML. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran resultados en menos tiempo y espacio.

- **XML**: XML o lenguaje de marcado extensible es un lenguaje de marcas utilizado para crear de una forma eficiente datos estructurados y compartirlos electrónicamente. El código XML es una recomendación formal del World Wide Web Consortium (W3C) ya que, entre otros factores, los datos XML se conocen como autodescriptivos, lo que significa que no es necesario crear previamente una estructura para almacenarlos, sino que el propio lenguaje la define implícitamente.

- **SQL**: SQL (Structured Query Language) es un lenguaje estándar e interactivo de acceso a bases de datos relacionales. Este lenguaje permite especificar diferentes tipos de operaciones mediante el uso de álgebra y cálculos relacionales. SQL ofrece la oportunidad de realizar "consultas" para obtener o recopilar información de bases de datos de una manera sencilla. Las consultas toman la forma de un lenguaje de comandos que permite seleccionar, insertar, modificar, encontrar datos concretos, etc.

Para su posterior instalación por parte del cliente tan solo será necesario instalar en el servidor PHP y MySQL.

3.2 PLAN DE TRABAJO

Para alcanzar los objetivos generales y específicos que se persiguen en este Trabajo Fin de Grado, se va a establecer un plan de trabajo que organizará el conjunto de acciones o tareas que se llevarán a cabo. Las tareas propuestas son las siguientes:

- **Tarea 1.** Estudio de los CMS que existen en la actualidad.
- **Tarea 2.** Establecimiento de los requisitos funcionales y no funcionales del CMS general.
- **Tarea 3.** Establecimiento de los requisitos funcionales y no funcionales de la tienda solidaria de la UCLM.
- **Tarea 4.** Diseño y desarrollo del CMS general.
- **Tarea 5.** Diseño y desarrollo de la tienda solidaria de la UCLM empleando el CMS general.
- **Tarea 6.** Diseño y desarrollo de la web para la oferta de servicios empleando el CMS general.
- **Tarea 7.** Diseño y desarrollo de la web para la promoción del CMS desarrollado.

3.3 ELECCIÓN DE LA METODOLOGÍA A UTILIZAR

3.3.1 Análisis del proyecto

A la hora de elegir que metodología se adapta mejor al desarrollo de un sistema software no se puede analizar tan solo el producto a desarrollar o las personas que van a hacerlo, se deben tener en cuenta múltiples aspectos, que podrían ser entre otros, cómo se realizará la toma de requisitos, que tipo de software se obtendrá al finalizar el proyecto, retroalimentación que se espera obtener de los clientes, frecuencia con la que pueden surgir cambios, coste de retrasos, experiencia previa en proyectos similares...

Teniendo en cuenta las respuestas a los puntos anteriores se ha llegado a la conclusión de que el mejor tipo de metodología que se puede utilizar es una metodología ágil.

3.3.2 Metodologías ágiles

Estas metodologías surgieron alrededor del año 2011 por el rechazo a las metodologías tradicionales, de las que se afirmaban que no se adaptaban a todos los tipos de proyectos. Se realizó una reunión entre expertos en el tema para llegar a definir una nueva forma de plantear la gestión de proyectos, en ella se fijaron las reglas que definirían estas nuevas metodologías, que fueron recogidas en el manifiesto ágil, un documento compuesto por doce principios, que son los siguientes:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Y estos principios se pueden agrupar y han sido redactados siempre teniendo en cuenta cuatro valores principales:

- Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio sobre seguir un plan.

Por supuesto no se eliminan los elementos a la derecha, sino que se le da una menor prioridad. Estos principios y reglas generan múltiples metodologías de las que se han analizado:

Adaptative Software Development (ASD)

Esta metodología es el resultado directo del desarrollo rápido de aplicaciones (RAD) con un marco de desarrollo y entrega más veloz. Su propósito es adaptar de forma rápida y eficiente los equipos de trabajo a los requisitos del mercado o a los cambios durante su desarrollo para evitar la rápida obsolescencia del producto tras su finalización. Requiere una planificación ligera y aprendizaje continuo. El enfoque ASD tiene tres etapas de trabajo: especulación, colaboración y aprendizaje. El desarrollo de software adaptable puede ser una metodología viable en caso de que la organización le de más importancia a la entrega rápida de productos y se permita agregarle calidad posteriormente mediante el desarrollo y mejora continua.

EXtreme Programming (XP)

La Programación Extrema (XP) es un marco de desarrollo de software flexible que tiene como objetivo crear software de alta calidad y una mejor experiencia y eficacia para el equipo de desarrollo. Se crea un marco flexible único para aplicaciones de ingeniería adecuadas para el desarrollo de software. Este enfoque tiene gran énfasis en los usuarios que utilizarán el producto final y en los clientes, lo que supone tanto una ventaja (el producto final será lo que se esperaba y todos sus aspectos estarán adecuados al cliente) y una desventaja (genera una gran dependencia con el cliente y el grupo de trabajo).

SCRUM

Scrum es un marco flexible, adaptativo y empírico, en el sentido de que proporciona un medio para que los equipos establezcan una hipótesis de cómo piensan que algo funciona, lo prueban, reflexionan sobre la experiencia y realizan los ajustes apropiados. Scrum está estructurado de una manera que permite a los equipos incorporar prácticas de otros marcos donde tienen sentido para el contexto del equipo. Esta metodología se adapta mejor en equipos multifuncionales que trabajan en un entorno de desarrollo donde hay una cantidad de trabajo no trivial que puede dividirse o modularizarse en iteraciones de trabajo no superiores a 2 ó 4 semanas.

3.3.3 Metodología escogida

A la hora de decidir la metodología a utilizar se ha recurrido a Iacovelli (véase [4], página 97), que define un framework para poder clasificar las principales metodologías ágiles y poder así cuantificar y tener más información a la hora de decidir la metodología que más se adapta al proyecto a realizar. Este framework define cuatro vistas desglosadas en aspectos del proyecto y basándose en las respuestas binarias a cada uno de estos aspectos se puede ver cual será la metodología más apropiada, en nuestro caso SCRUM. A continuación se justifica tal decisión.

SCRUM

Scrum es la única de todas las metodologías ágiles que introduce la idea del control empírico de los procesos [2]. Esto significa que Scrum utiliza el progreso real de un proyecto para definir los siguientes pasos a seguir. En Scrum, los proyectos se dividen en tareas u objetivos breves, conocidos como “sprints“ y al terminarlos se realiza una reunión entre el cliente y el equipo de desarrollo para evaluar el progreso del proyecto y fijar las siguientes tareas. Esto permite que la dirección del proyecto se ajuste o se reoriente una vez finalizada cada tarea.

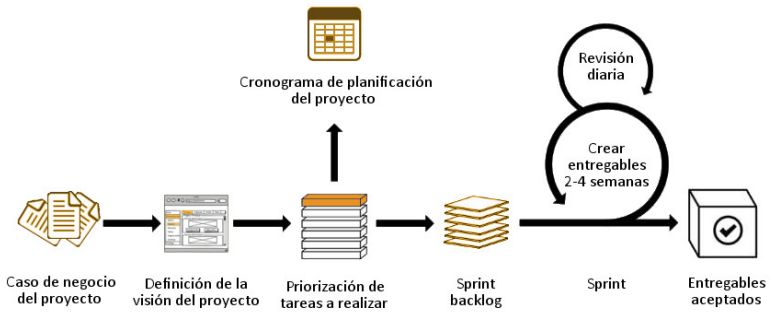


Figura 3.1: Flujo de trabajo en SCRUM

El éxito de Scrum no solo depende de esta capacidad de adaptación, también tiene vital importancia su organización. Existen tres roles fundamentales, Product Owner (propietario del producto), Scrum Master (especialista en Scrum) y Team Member (miembros del equipo), esto será lo que más necesite adaptación en este proyecto, ya que el número de involucrados es muy reducido y el cliente final (que podría considerarse como tal los representantes de la UCLM en el ámbito de la tienda benéfica a crear) no es cliente del proyecto, sino de una aplicación del mismo, de este modo la división de roles centrales será la siguiente:

- **Product Owner:** En Scrum, el Product Owner se dedica a comunicar la visión del producto al

equipo de desarrollo y representar los intereses del cliente. En nuestro caso este rol se asignará de forma conjunta al autor y al tutor de este proyecto.

- **Scrum Master:** El Scrum Master actúa como enlace entre el Product Owner y el equipo. El Scrum Master no dirige al equipo, sino que se encarga de que el equipo sea productivo, a la vez que permite que los logros del equipo sean visibles ante el Product Owner. En nuestro caso este rol se asignará al autor del proyecto.
- **Team Member:** En la metodología Scrum, el equipo es el responsable de terminar el trabajo. Idealmente, los equipos están formados por miembros multifuncionales, esto en nuestro caso es imposible. En nuestro caso este rol se asignará al autor del proyecto.

CAPÍTULO 4

ESTUDIO DE LOS GESTORES DE CONTENIDO ACTUALES

En este capítulo se presentan los diferentes tipos de gestores de contenido que existen, analizando las funcionalidades y servicios que ofrecen, y cómo lo hacen. Este análisis permitirá tomar conciencia de los primeros requisitos sobre el gestor de contenidos que se va a desarrollar en este TFG.

4.1 CLASIFICACIÓN DE CMS POR TIPO DE PÁGINA WEB

Las páginas web empresariales, al igual que los gestores de contenido diseñados para crearlas, deben tener varios factores en común, como tener una completa descripción de su actividad de negocio, tener una página de contacto para todos sus clientes potenciales, información sobre sus productos y servicios, elementos que aumenten la fiabilidad y eficacia de la empresa (testimonios de clientes, noticias relacionadas con la empresa...), **pero más allá de esto, pueden tener distintas formas de describir su actividad**, distintas formas de promocionar sus productos y servicios, etc. **por ello se necesitan distintos tipos de páginas web y por tanto distintos tipos de gestores de contenido**. En el ámbito del desarrollo de páginas de ámbito profesional, las más habituales son:

- Páginas estáticas

  Se refiere a páginas que muestran una información permanente en la que el usuario no puede interactuar excepto para leer el contenido. Las empresas más pequeñas pueden optar por esta opción, ya que es la que requiere menos mantenimiento, sin embargo, también son las páginas con las que se obtiene menos beneficio. Prácticamente cualquier CMS podría crearlas, pero todos ellos añadirán una cantidad innecesaria de código, carga de trabajo, déficit de procesamiento y rendimiento, etc. por lo que lo más apropiado sería escoger el gestor más sencillo o incluso evitarlos y crear una página HTML simple.

- Página corporativa

  Las webs corporativas son aquellas que describen las características de una compañía y recogen la información principal de la misma, como quiénes son, su ubicación, en qué proyectos están participando, así como otros datos sobre la propia empresa. Además, también suelen mostrarse los principales productos o servicios que la empresa ofrece, pero no existe la posibilidad de contratarlos u obtenerlos vía online. Estas páginas necesitan más mantenimiento, actualizaciones continuas y renovar constantemente sus nuevos trabajos y servicios. Actualmente, los gestores de contenido existentes más apropiados para este tipo de webs son Joomla o Drupal, estos desarrollan páginas mayormente eficientes y

profesionales, pero tienen un inconveniente de complejidad ya que requieren un cierto nivel de conocimiento técnico e implementación de código para adecuar completamente el contenido.

- Página de comercio electrónico

  Los e-commerce o tiendas online son sitios webs en los que se comercializan productos o servicios. Como se ha explicado anteriormente, en la actualidad están teniendo un gran crecimiento debido a que muchos usuarios ya realizan sus transacciones por internet. De entre todas las opciones dadas esta es la más productiva ya que algunas de estas páginas se han convertido en gigantes de la distribución y son algunas de las empresas más grandes del mundo como Ebay o Amazon. Se es tan consciente de la importancia de este tipo de páginas que incluso hay gestores de contenido que se encargan en exclusiva de ellas (Prestashop) o módulos/ampliaciones para ello en gestores de otros tipos (Woocommerce para wordpress).

Estos ejemplos son solo algunos tipos de páginas que pueden interesar y como se ha mencionado, para cada una de ellos sería necesario escoger un tipo de gestor de contenido u otro, sin embargo no siempre se escogen los más eficientes, por lo que a la hora de comparar los distintos gestores de contenido, no pondremos solo sobre la mesa los más adecuados, sino también los más populares.

4.2 FUNCIONALIDADES DE UN CMS PARA COMERCIO

Aunque este trabajo pretende generar un gestor de contenido de carácter general que permita la posibilidad de crear prácticamente cualquier tipo de página web, el principal nicho de mercado al que pretendemos acceder es el relacionado con microempresas, es por ello, por lo que es importante que el gestor desarrollado pueda crear soluciones de comercio electrónico. Además, el resto de aplicaciones que se podrían crear con el gestor desarrollado pueden verse como una simplificación del mismo.

A continuación se analiza qué funcionalidades debería tener una pagina web de compra/venta online, para ello se estudiarán las necesidades que tendrán los usuarios que accederán a dicha página web. **En este tipo de páginas se detectan dos tipos principales de usuarios**, los relacionados con la empresa (el administrador de la tienda o empleados gestores) y los visitantes y clientes potenciales, que interaccionarán de diversas formas con la aplicación.

La interacción principal con la aplicación de cada uno de ellos se realiza en dos partes claramente diferenciadas pero dependientes, el storefront o shopfront o frontend de la tienda y el back office o backend de la tienda. **El frontend puede definirse como el escaparate** y la misma tienda, lo que ve el visitante (potencial cliente) y por donde este puede moverse, para ver y examinar los productos o servicios que se comercializan en ella. **El backend es la trastienda, lo que hay detrás de la tienda, esa zona en la que el usuario no puede acceder, pero que es necesaria para el dueño o administrador** de la tienda, ya que le permite realizar una serie actividades de apoyo al negocio.

Los administradores serán los que se relacionan directamente con el backend, ya que este les permitirá realizar aquellas actividades de apoyo necesarias para la gestión y dirección del negocio, pero también les permitirá dar cierto énfasis sobre lo que quieren que los visitantes se encuentren en su tienda. Los usuarios o visitantes serán los que interactúen con el frontend, aquí es donde van a encontrar lo que buscan.

Seguidamente, se va a analizar qué es lo que piden cada uno de estos potenciales usuarios a la aplicación, esto será útil a la hora de definir las funcionalidades que debe generar el gestor que se está diseñando y desarrollando en este trabajo en relación a las aplicaciones de comercio electrónico.

4.2.1 Necesidades de los administradores de la tienda

Los administradores serán los que se relacionan directamente con el backend, ya que este les permitirá realizar aquellas actividades de apoyo necesarias para la gestión y dirección del negocio. No obstante, también tienen una visión sobre qué es lo que quieren que el visitante se encuentre cuando visite su tienda. Conforme a esto, el administrador tiene una serie de necesidades sobre la tienda que se genere. La tienda debe:

- Permitir el acceso desde distintos tipos de dispositivos.
- Permitir la gestión del catálogo de la tienda, tanto en el tipo como en los elementos que lo conforman.
- Proporcionar acceso al cliente al catálogo actual de productos ofertados.
- Proporcionar al cliente la información necesaria sobre los productos/servicios en venta, y hacerlo de la mejor forma posible dependiendo del tipo de producto/servicio.
- Mantener la integridad de los datos sensibles del negocio, y sobre todo en lo que al stock se refiere, se debe garantizar en caso de integración que el stock físico y el disponible coincida en todo momento.
- Permitir obtener información sobre los pedidos, así como mecanismos para procesarlos.
- Verificar el crédito de un cliente y en caso positivo permitir la compra.
- Permitir el pago de productos o servicios adquiridos con diferentes medios.
- Permitir la gestión de los envíos, permitiendo su procesamiento y seguimiento para garantizar que son entregados en tiempo.
- Proporcionar los medios para que visitantes compradores se registren en el sitio, hagan comentarios o puedan solicitar información adicional.
- Permitir ofrecer un buen servicio al cliente, con mecanismos para responder cuestiones de los clientes o poder trasladar las cuestiones al departamento que competa.
- Permitir analizar las compras con el fin de conocer como funciona el canal de venta, personalizar las experiencias de los compradores o tomar decisiones que afecten a la marcha del negocio.
- Proporcionar soporte pre y post-venta basado en la web.
- Permitir el cross-selling o venta cruzada (productos complementarios a los que ha comprado) o up-selling (productos más caros que los que ha comprado).
- Permitir la personalización de la tienda en función del lugar de visita (idioma y moneda).
- Permitir medir y analizar el tráfico en el sitio para tomar decisiones en la modificación y mantenimiento de la tienda online.

4.2.2 Necesidades de los clientes potenciales

Estos usuarios son los que están directamente relacionados con el frontend de la tienda online. Las necesidades que debe cubrir la aplicación, todas relacionadas con el frontend, y que los clientes demandan, son las siguientes:

- Sea accesible, tanto desde dispositivos móviles, como ordenadores.
- Ofrezca información interesante y detallada sobre el producto/servicio y sobre la empresa.
- Disponga de un sistema que les permita buscar, encontrar, evaluar y comparar productos/servicios.
- Permita la selección de productos/servicios.
- Permita comprar fácilmente productos/servicios.
- Permita el pago del pedido con varios sistemas (p.e. pasarelas de pago con banco, Paypal,...).
- Ofrezca información sobre el estado en el que se encuentra su pedido.
- Permita el acceso a sistemas de ayuda pre y post compra.

4.2.3 Funcionalidades del backend

Para satisfacer las necesidades que tiene el administrador de la página sobre el backend, en este se deberían ofrecer las siguientes funcionalidades:

- **Gestión de información sobre la tienda.** El administrador de la tienda debe tener la posibilidad de modificar las descripciones importantes del negocio.
- **Gestión del catálogo.** Una de las partes más importantes de una tienda online es su catálogo de productos. Desde el back office se debería permitir la creación de una estructura de categorías, así como establecer la información de cada producto de forma atractiva y de forma que permita distinguirse de la competencia.
- **Gestión de las relaciones en el catálogo.** Sería recomendable permitir el Cross-selling o Up-selling, para ello se debería incluir un mecanismo para establecer relaciones entre elementos de la jerarquía del catálogo.
- **Gestión de stock.** Se debería incluir una funcionalidad para permitir la introducción de los productos que están a la venta, además esta información es recomendable que sea coherente con el stock físico del negocio.
- **Gestión de ofertas y descuentos.** Sería recomendable incluir una funcionalidad para permitir la creación de ofertas o descuentos en productos del catálogo.
- **Gestión de productos destacados.** Se debería incluir una funcionalidad para permitir al administrador de la tienda agregar o quitar productos de la sección de productos destacados.
- **Gestión de clientes.** Los clientes son lo más valioso de una tienda online, por lo que se recomienda incluir un sistema para su gestión, permitiendo su registro, y el acceso a su información para su consulta, modificación de datos o incluso dar de baja.
- **Gestión de pedidos.** Se debería incluir un formulario para gestionar los pedidos que se hayan realizado y que permita actuar sobre ellos y conocer el estado en que se encuentran en cada momento.
- **Gestión de la comunicación con visitantes o clientes.** Sería recomendable establecer mecanismos que permitan al administrador de la tienda comunicarse y relacionarse con los visitantes o clientes.

- **Generación de informes.** Es recomendable incluir un módulo para generar información que permita conocer cuál es el rendimiento de un canal de venta y cómo de buena es la plataforma, es recomendable ofrecer y generar datos estadísticos. Esta información es esencial para tomar decisiones que afecten al negocio o al diseño de la plataforma.

A su vez estos deben buscarse la máxima claridad en uso y eficiencia en su diseño e implementación, ya que son operaciones sensibles sobre una tienda en funcionamiento a la que cualquier tipo de persona, con más o menos conocimiento específico, puede acceder desde cualquier tipo de dispositivo.

Por lo usual estas funcionalidades se harán accesibles en en backend a través de un panel de administración.

4.2.4 Funcionalidades del frontend

Puesto que el apartado visual, también llamado frontend, de una página web supone un gran elemento diferenciador y obtiene una gran importancia, se deben cumplir una serie de requisitos. Sin ellos la página web creada con el gestor sufriría una desventaja competitiva respecto a cualquier otra página de comercio electrónico. También cabe destacar que la satisfacción del cliente es muy importante, por tanto la robustez y la sencillez en cada uno de los siguientes apartados es vital:

- **Visualización de la tienda.** Es recomendable que sea elegante, usable, personalizada (mostrando productos vistos recientemente o los más vistos o recomendados en función del historial del visitante) y contar con un diseño responsive. Nunca hay que olvidar que es lo primero que ve el cliente y por tanto es recomendable que sea atractiva si queremos generar ventas.
- **Presentación del catálogo.** Se recomienda permitir el acceso al cliente a aquellas secciones o departamentos de la tienda que más le interesen, además es recomendable que sea intuitivo.
- **Presentación de productos.** Se recomienda proporcionar al cliente información sobre el producto a través de una interfaz web amigable. Puede contener las especificaciones detalladas del producto, imágenes, publicidad, valoraciones y opiniones de otros compradores, etc.
- **Buscador.** Se recomienda poner a disposición del cliente un buscador que permita localizar en el catálogo los productos que le interese, además este buscador en función de la complejidad del catálogo deberá tener filtros para refinar las búsquedas.
- **Cesta o carrito de la compra.** Se recomienda permitir al cliente la visita y recorrido por la tienda introduciendo productos en el carro, informándole en cada momento del precio de cada producto y por supuesto del precio final en la caja (incluyendo los gastos de envío).
- **Realización de pedidos.** Se recomienda proporcionar al cliente un mecanismo para pedir los productos que desee. Cada pedido consistirá en uno o varios productos (referencia y descripción), junto con la cantidad pedida de cada uno. La realización de un pedido implicará el registro del cliente con un proceso que sebe ser rápido, generalmente a través de la entrada de datos por medio de formularios sencillos (sin pedir más datos de los estrictamente necesarios).
- **Realización de pagos.** Se recomienda permitir al cliente emplear las formas de pago más populares, garantizando fiabilidad y seguridad.
- **Seguimiento del pedido.** Se recomienda informar del estado en el que se encuentra el pedido en cada momento.
- **Atención al cliente.** Se recomienda proporcionar asistencia al cliente que tenga problemas o cuestiones relacionadas con la tienda, o los productos.

4.3 PRINCIPALES GESTORES DE CONTENIDO

Conocer lo que ofrecen los gestores de contenido más usados, es también esencial a la hora de determinar que es lo que deber ofrecer el gestor de contenidos que se está diseñando y desarrollando en este trabajo. En esta sección se va a analizar aquellos aspectos que se consideran interesantes y que serán incluidos o mejorados en el gestor Lurión.

Es difícil determinar qué gestores de contenido se utilizan más, puesto que ninguna fuente puede recoger con objetividad el total de páginas creadas con cada uno de ellos, por ello se realizará una estimación de su importancia basada en las búsquedas realizadas en Google de cada uno de los gestores. Lógicamente, no es un indicador totalmente fiable del uso de cada gestor, entre otras cosas, porque hay empresas de desarrollo que crean múltiples soluciones web con un mismo gestor de contenido (y no buscan el gestor en Google para cada una de ellas), pero si nos permitirá realizar una estimación sobre su interés (ver Figura 4.1).

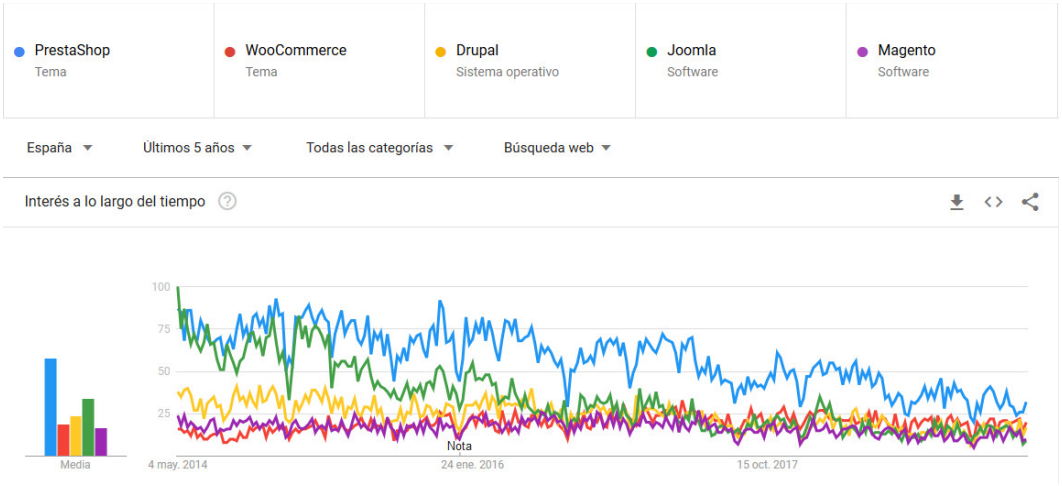


Figura 4.1: Trends Google. Gestores e-commerce más buscados, 2014-Actualidad

En la imagen anterior se muestran las cinco soluciones con mejores resultados. Al realizar el estudio se han consultado las búsquedas de los distintos gestores de contenido y soluciones de comercio electrónico en los últimos 5 años. Puesto que nuestro objetivo es el desarrollo de páginas web empresariales con posibilidad de venta online, y hay gestores cuyo foco lo ponen en blogs o foros, algunas búsquedas han sido adaptadas (como en el caso del gestor de contenido Wordpress, que es un CMS general empleado para crear blogs, pero que si incluye el módulo Woocommerce, permite la creación de soluciones de comercio electrónico). Es por esto por lo que se considerará solo las búsquedas de Woocommerce, el módulo de compra online de Wordpress.

En vista de los datos, **los gestores con posiciones más altas** como puede verse en la imagen anterior **son Prestashop, Joomla, Drupal, Woocommerce y Magento** con una superioridad de forma homogénea en toda España de Prestashop, excepto en las islas (ver Figura 4.2). No obstante, queremos resaltar que, sin considerar el propósito del CMS, sin tener en cuenta si es un gestor general o específico de comercio electrónico, el más popular, con cierta diferencia, es Wordpress.

Figura 4.2: Trends Google. Gestores buscados por provincias, 2014-Actualidad

4.3.1 Especificaciones

Todos los gestores de contenido cumplen funcionalidades similares y comparten tecnologías, lenguajes de programación, etc. Sin embargo, al igual que en prácticamente todo sistema software, son diferentes y se adecuan mejor o peor a ciertas circunstancias. Seguidamente se presentan brevemente cada uno de ellos.

Magento

Magento está orientado a crear atractivos y potentes sitios web para soluciones de comercio electrónico, usualmente para empresas de gran tamaño, por lo que necesita muchos recursos de almacenamiento y de computo. La personalización de la página, en términos generales, se basa en temas que modifican el aspecto de toda la web y deben ser subidos al servidor de forma manual. El panel de administración tiene una organización intuitiva y es similar a la mayoría de gestores por lo que es fácil navegar en él. Debido a que este CMS está orientado al e-commerce, cuenta con amplias funciones para crear catálogos de producto.

Prestashop

Es uno de los gestores de contenido web más populares para el desarrollo de e-commerces, el más popular en España siguiendo los datos antes presentados de Google Trends. De hecho, en términos generales, está muy cerca de WordPress en cuanto a popularidad. Sin embargo, se considera que es menos potente que Magento, también consume menos recursos. Su back-end es de los más sencillos, por lo que muchas tiendas online de pequeñas empresas lo elijen como CMS. La cantidad de plantillas a las que se tiene acceso es considerable, sin embargo, necesitan actualizaciones constantes (algo que las plantillas gratuitas no siempre hacen) y la gran mayoría son de pago.

Joomla

El diseño de Joomla está pensado para desarrolladores con experiencia en incorporar código, en lugar de principiantes o personas sin conocimientos de informática, que dependen más de los constructores de sitios web para crear contenidos. Tiene múltiples complementos específicos para negocios, en general este gestor funciona bien para medianas y grandes empresas que puedan invertir en la creación de un sitio web algo más atractivo, único, interactivo y profesional. Durante mucho tiempo fue el CMS más utilizado, o por lo menos el más buscado, todo el periodo previo a 2014, conforme a los datos proporcionador por Google Trends. Independientemente de los complementos

de negocio que se necesiten para la página, Joomla los tiene casi todos de forma gratuita. Sin embargo, la selección de temas y personalización del aspecto no es tan extensa como otras soluciones de CMS.

Drupal

Drupal es un sistema de gestión de contenidos destinado a ayudar a diseñadores profesionales a crear sitios web potentes. Espacios capaces de manejar un gran volumen de visitantes y cientos de páginas de contenido. Drupal es una de las soluciones fundadoras en los programas de gestión de contenidos por lo que tiene algunos de los mejores complementos y plugins disponibles y cuenta con una selección considerable de temas. Sin embargo, requiere de una gran cantidad de codificación para convertir tu sitio en algo realmente potente. Resumiendo, es el gestor que tiene más contenido y opciones de personalización pero requiere una cantidad de conocimientos sobre su estructura y sobre desarrollo web que ensombrece todo lo demás.

Woocommerce

Wordpress es uno de los gestores de contenido más conocidos y populares, pero no tanto por su contenido, sino por la comunidad, muy amplia y activa, de creadores que hay a su alrededor. Wordpress como tal es un creador de blogs con un panel de administración muy intuitivo y fácil de usar, pero ha trabajado en facilitar a creadores de código su labor, de este modo, se puede crear de forma bastante sencilla una plantilla que cambie por completo el aspecto de una web e implementa un sistema de alias para cambiar a través de módulos incluso la funcionalidad más interna del gestor. Uno de estos módulos es Woocommerce, que utilizando las funciones de Wordpress crea un nuevo menú en el panel de administración que permite crear productos (siguiendo la metodología en la que en Wordpress se crean posts) y guardarlos en una base de datos auxiliar que el gestor tiene preparado para los creadores. Por supuesto esto es una adaptación, por lo que no es algo eficiente ni contiene código limpio, pero debido a la facilidad de uso de Wordpress está ganando gran popularidad.

4.4 COMPARATIVA EN ELEMENTOS DISTINTIVOS

Una vez analizados de forma genérica los gestores de contenido más conocidos y utilizados pasaremos a puntualizar y analizar los elementos que supongan un cambio significativo en el gestor y que puedan marcar la diferencia entre elegir uno u otro. Para hacer más claras estas diferencias las dividiremos en varios apartados: Personalización, Catálogo, Usuarios y Pedidos.

4.4.1 Personalización

Esta subsección englobará todos los apartados que supongan un cambio en la visualización o que añadan información de la página web en el navegador del cliente que la visite y que dependan del administrador de la página:

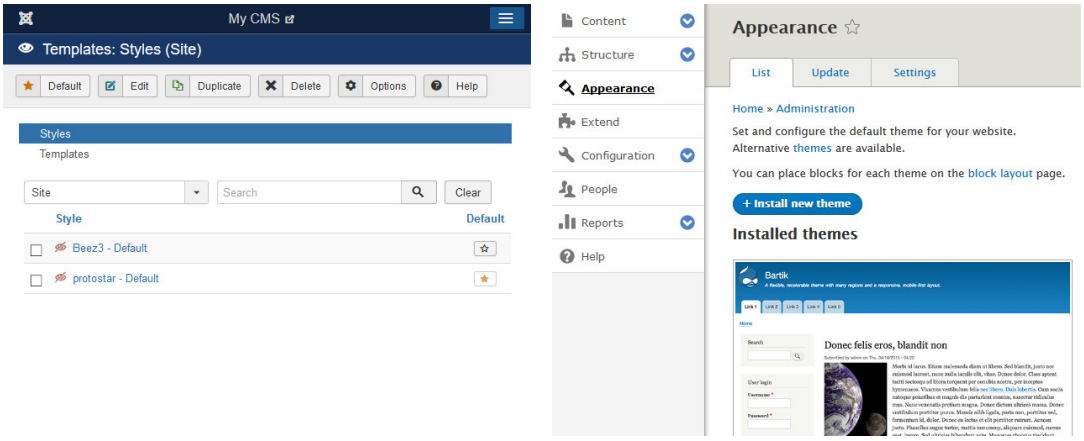
Aspecto

La modificación del aspecto es prácticamente el elemento más importante de todo CMS, supone un cambio diferenciador en la web y define la forma en la que los clientes/usuarios interactuarán con la web. Aquí es donde pretendemos crear el principal atractivo de Lurión. Este apartado englobará tanto el aspecto básico de toda página web (cabecera, pie de página, menú, etc.) tanto el aspecto del

catálogo de la tienda, aspecto del formulario de inicio de sesión, página de contacto, etc. (Véase las diferencias en este apartado entre distintos gestores de contenido en la tabla 4.1 y la figura 4.3).

Tabla 4.1: Comparativa: Aspecto, Joomla vs Drupal

Joomla	Drupal
• La forma de modificar el aspecto es el más particular de entre todos los CMS. • No tiene estructura de páginas fija. Esta es configurable a través de módulos desde el panel de administración. • Los contenidos de cada página se hace a través de los llamados 'Archivos' (páginas estáticas genéricas). • Los "Archivos" son configurables y permiten realizar cualquier función o mostrar cualquier tipo de información. • Los "Archivos" son personalizables a partir de plantillas genéricas, de manera individual o en grupo (empleando etiquetas).	• La forma de modificar el aspecto es la más común entre los CMS. • La estructura de páginas de la web es configurable. Permite añadir distintos tipos de elementos desde el panel de administración. • El aspecto de la página será modificable a través de "Plantillas"que cambiarán por completo el aspecto de la web. • Las páginas comparten código y estilo entre ellas. • La plantilla debe modificarse entera si existe algún problema (diseño responsive, actualizaciones en el gestor, etc.).



(a) Aspecto en Joomla (b) Aspecto en Drupal

Figura 4.3: Aspecto, Joompla vs Drupal

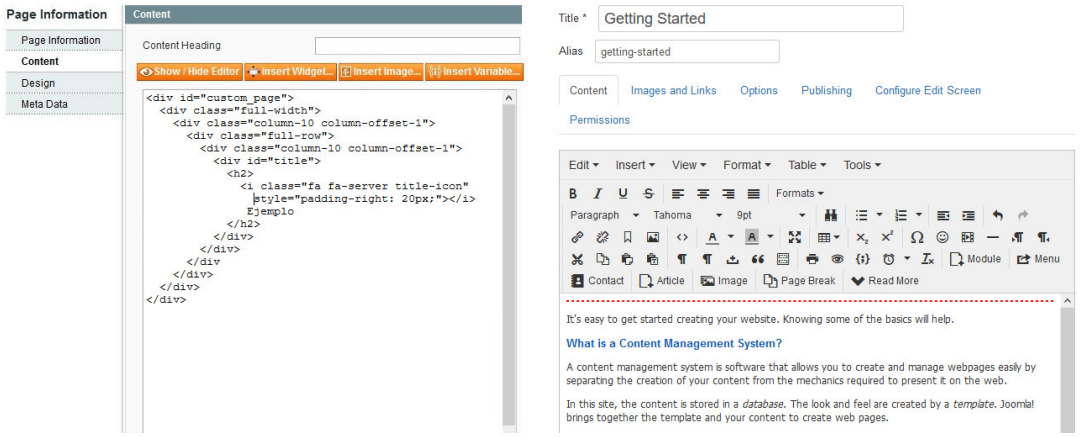
Páginas estáticas y entradas

Las páginas estáticas son la mejor forma de personalizar la página web y añadir información sobre la empresa para la que se está creando la solución. Permite añadir información de cualquier tipo y con cualquier estilo, ya que son páginas que no necesitan recursos ni información dinámica (como podrían ser los productos de la empresa, el árbol de categorías, etc.) y a la que se le puede aplicar

cualquier estilo y estructura. Existen muchas formas de crear páginas estáticas en un CMS como podría ser a través de un editor de texto (más restrictivo), un editor de código html (necesita más conocimientos), un sistema de ganchos (conocimientos específicos del CMS). (Véase las diferencias en este apartado entre distintos gestores de contenido en la tabla 4.2 y la figura 4.4).

Tabla 4.2: Comparativa: Páginas Estáticas, Magento vs Joomla

Magento	**Joomla**
• Las páginas estáticas son creadas gracias a un editor de código HTML, aunque no dispone de previsualización. • Las entradas no existen, aunque existe un plugin para visualizar las entradas de un blog Wordpress que esté instalado y funcionando en el mismo u otro dominio. • Las entradas, aun con estas limitaciones, tiene otras opciones como añadir metainformación para personalizarla.	• Los "Archivos" se pueden crear con un sistema de edición muy completo. No existen ni páginas estáticas ni entradas como tal. • Los "Archivos" se comportan de una forma u otra en función de la etiqueta que se le asocie. Dispone de un sistema de etiquetas, y en función de estas y del formulario asignado la página cumplirá una función u otra. • El panel de administración ofrece una funcionalidad para determinar si un "Archivo" es una página estática o página principal.



(a) Páginas estáticas en Magento (b) Archivos en Joomla

Figura 4.4: Páginas estáticas y entradas, Magento vs Joomla

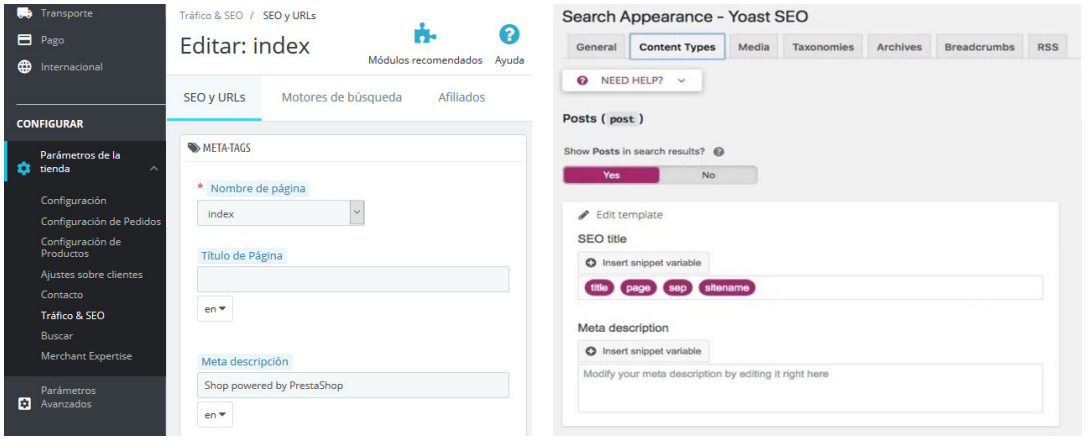
Metadatos

Con metadatos nos referimos a aquellos datos que está orientados a la comunicación con los navegadores, para que estos conozcan la temática de la web y lo incluyan en los resultados de búsqueda. Es esencial su uso cuando se desarrolla una web para mejorar el posicionamiento de la página. La idea es que estos metadatos incluyan las posibles palabras con las que se desea que los usuarios lleguen a la web cuando buscan en su navegador. Estos datos están directamente relacionados

con el posicionamiento orgánico o SEO. (Véase las diferencias en este apartado entre distintos gestores de contenido en la tabla 4.3 y la figura 4.5).

Tabla 4.3: Comparativa: Macrocontenido, Prestashop vs Wordpress

Prestashop	Wordpress
• Los metadatos, tanto palabras claves, como títulos y descripciones pueden añadirse en cada página. • Los metadatos no solo se añaden a las páginas estáticas sino que pueden añadirse también a las páginas de visualización de productos para que puedan indexarse. • El panel de administración incluye un menun donde elegir cada página creada y modificar sus metadatos.	• Los metadatos no pueden añadirse por defecto en Wordpress sin instalar un plugin, ni genéricos de la página web, ni por cada página. • La comunidad de creadores ha desarrollado plugins que permiten añadir metainformación a la página. • Los plugins existentes en sus versiones gratuitas proporcionan lo básico (p.e. Yoast SEO), para añadir metainformación a cada página o entrada hay que usar versiones de pago.



(a) Metadatos en Prestashop (b) Metadatos en Wordpress

Figura 4.5: Metadatos, Prestashop vs Wordpress

El gestor que se desarrollará en este TFG deberá permitir la personalización de la web en todos los aspectos esenciales, p.e. front-end, datos del negocio, etc,... Además debe dar libertad a la hora de crear páginas de información más estática, y siempre teniendo en cuenta que estas pueden incluir metadatos.

4.4.2 Catálogo

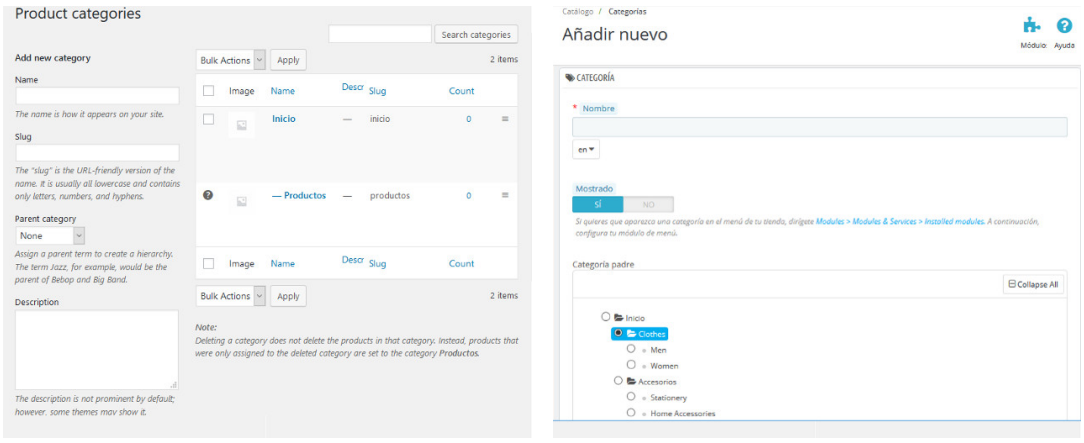
El catálogo es uno de los aspectos esenciales de una tienda online. Aquí hablaremos de cómo los distintos CMS tratan los aspectos relacionados con la creación de categorías y productos.

Categorías

Para crear un catálogo es imprescindible la creación de un sistema de categorías. Estas permiten navegar mucho más fácilmente y dar una idea al usuario sobre qué tipo de producto se está vendiendo. La forma más recomendable de crear las categorías es mediante una estructura de datos (grafo) que imita la estructura jerárquica de un árbol, en el que, cada nodo y hoja es una categoría. (Véase las diferencias en este apartado entre distintos gestores de contenido en la tabla 4.4 y la figura 4.6).

Tabla 4.4: Comparativa: Categorías, Woocommerce vs Prestashop

Woocommerce	Prestashop
• Los metadatos nativos de Wordpress sirven para determinar si la entrada es un producto o no, así como su categoría. • Las entradas, las páginas y los productos añadidos se almacenan juntos en una tabla de su base de datos. • La forma en la que se almacenan los "productos"se mantiene, aunque también dispone de sus propios menús en los que permite añadir las categorías como un elemento no puramente textual. • La estructura de árbol es empleada para estructurar las categorías creadas.	• La creación de categorías y de productos, y la asignación de estos a categorías se hace a través de menús diseñados para tal fin. • La estructura de categorías que emplea es la usual en forma de árbol, pero tiene funciones que no todos los gestores poseen. • La forma de mostrar el árbol de categorías previamente creado es visual y sencilla. • La información adicional, como imágenes de referencia a las categorías, metainformación, etc. se puede añadir fácilmente.



(a) Categorías en WooCommerce (b) Categorías en Prestashop

Figura 4.6: Categorías, WooCommerce vs Prestashop

Productos

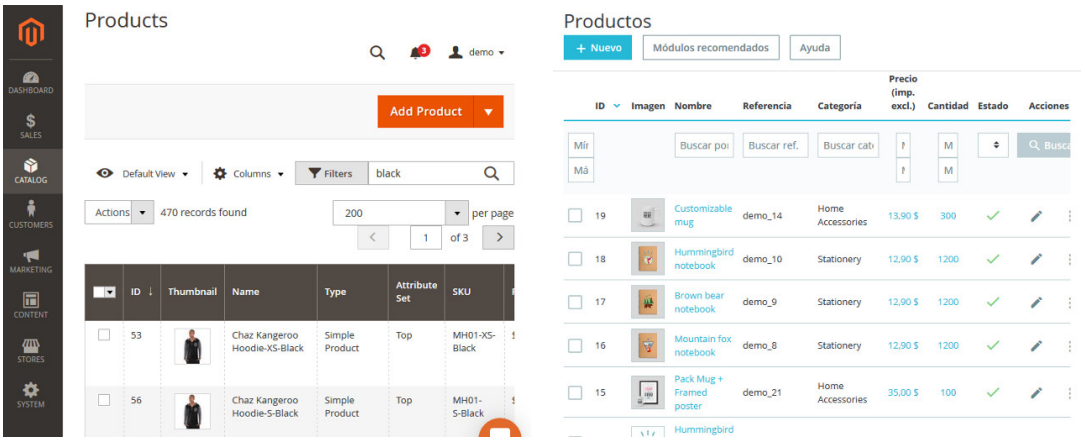
Puesto que el gestor de contenido que se pretende crear en este TFG dará apoyo para crear una solución de comercio electrónico, se le dará bastante importancia a la forma de gestionar los productos (i.e. crearlos, eliminarlos o modificarlos). A continuación, analizaremos cómo otros gestores de contenido gestionan los productos. (Véase las diferencias en este apartado entre distintos gestores de contenido en la tabla 4.5 y la figura 4.7).

Tabla 4.5: Comparativa: Productos, Magento vs Prestashop

Magento	Prestashop
• La zona de administración proporciona una lista con el stock de productos disponibles en la tienda. Además, permite desde esa misma añadir, modificar o visualizar productos. • Las acciones se puede hacer sobre grupos de productos. El concepto de grupo es diferente al de categoría. • Los atributos se pueden crear sobre una categoría, sobre una serie de productos fijos o con alcance global, pudiendo asignarles valores recomendados u obligatorios. • Los productos y los atributos se pueden relacionar por medio de la asignación de un valor de cada atributo por producto. En lugar de relacionar a través de claves ajenas. • La visualización no es correcta en pantallas con baja resolución, ya que no tiene un diseño adaptativo. • La inclusión de productos en la tienda se puede hacer automáticamente a través de archivos CSV, aunque su creación manual es compleja.	• La zona de administración proporciona una lista con el stock de la tienda, en la que están los enlaces para añadir, modificar o visualizar productos. • La búsqueda de productos se puede realizar por cada uno de sus elementos (nombre, categoría, etc.). • Las tasas e impuestos de los productos se puede hacer desde la zona de administración, e incluso añadir distintas monedas para permitir la internacionalización la página web. • Los atributos de cada categoría se pueden crear con total libertad, y añadir valores recomendados u obligatorios. • Los tipos de atributos son diversos, y además permiten las combinaciones, que muestran distintos tipos del mismo producto (diferencias en colores, tamaños, etc.). • La forma de añadir varios productos a la vez se puede hacer en bloque, para ahorrar tiempo. Esto se hace a través de archivos CSV, aunque crear manualmente el archivo es algo complejo.

Nuestro gestor deberá mostrar gran atención a la creación y mantenimiento del catálogo, puesto que el nicho de mercado al que está dedicado quieren una página web para mostrar sus productos/servicios.

Debemos permitir crear productos y personalizarlos, permitir crear un árbol de categorías genérico que pueda aplicarse a cualquier tienda, etc…



(a) Productos en Magento

(b) Productos en Prestashop

Figura 4.7: Productos, Magento vs Prestashop

4.4.3 Usuarios

La gestión de usuarios es vital en todo gestor de contenido, no solo en cuanto a gestión de clientes (que no lleva a ser vital ni siquiera en gestores dedicados al comercio electrónico) sino en cuanto a gestión de usuarios administradores que se encarguen de la solución web desarrollada.

Clientes

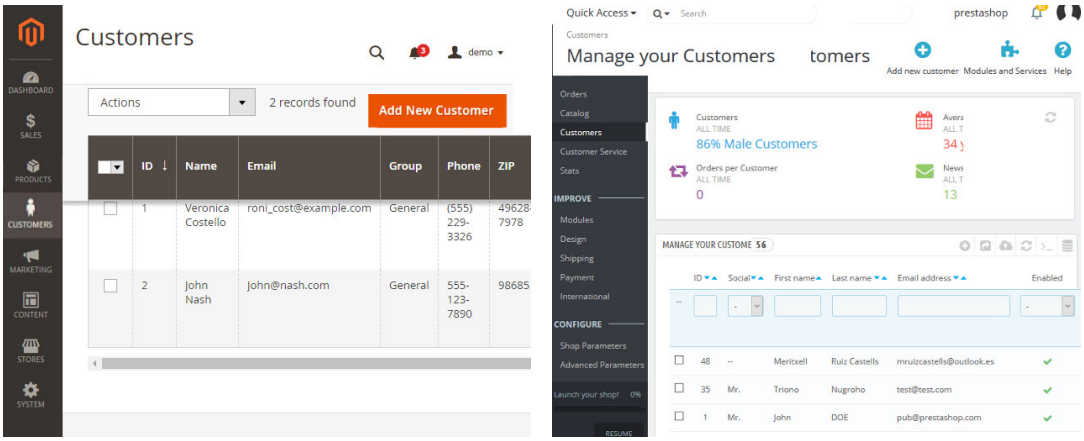
No todos los gestores de contenido contemplan el concepto de cliente o usuario de la página web, puesto que no todos están pensados para la creación de páginas web de compra-venta online, sin embargo, sí que estudiaremos los que incluyen esta opción de forma innata, aunque no existen prácticamente diferencias. Todos ellos permiten:

- La creación de clientes, tanto desde el frontend de la página web (si la plantilla lo incluye), como desde el panel de administración.
- La modificación y eliminación de clientes desde el panel de administración.
- La visualización de los listados de clientes con sus datos de contacto.
- La consulta de la información de los clientes y modificación de la información básica (excepto información sensible).
- La consulta de los usuarios registrados, ya hayan sido añadidos desde el backend como desde el frontend.

Hay que destacar que, como otros menús en los que se listan elementos, en algunos CMS no se dispone de un diseño responsive en la zona de administración y hay una fuerte dependencia con la plantilla o plugins que se empleen. (Véase la figura 4.8)

Administradores

Gestionar usuarios administradores de la página web es algo que todos los gestores deben tener, no es posible tener una página web con un back-end abierto en el que cualquiera pueda entrar (por razones obvias). Este apartado, al ser tan necesario, tiene una gestión bastante estandarizada.



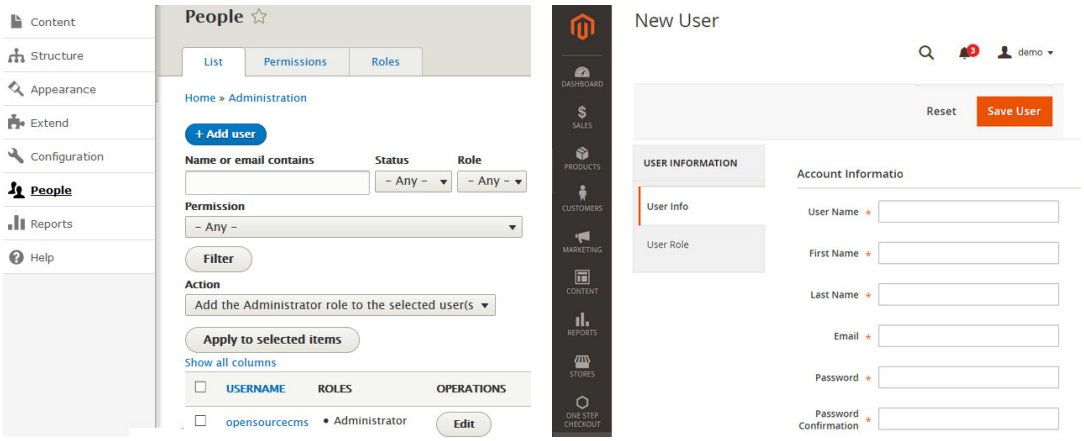
(a) Clientes en Magento (b) Clientes en Prestashop

Figura 4.8: Clientes, Magento vs Prestashop

Todos los estudiados disponen de una sección para crear roles, otra sección para asignar los permisos que tendrán estos roles y por último la creación del usuario con la asignación de uno de estos roles. Se mostrarán dos casos que agrupan las dos posibilidades de gestionar esto, aunque las diferencias son tan mínimas que residen en si la creación de roles y asignación de dichos roles están en un mismo menú o no. (Véase la figura 4.9)

Por lo usual todos los CMS:

- Proporcionan un usuario administrador principal, obligatorio, que se crea en el proceso de instalación del gestor.
- Asignan a dicho usuario administrador principal todos los permisos, no será un usuario como tal, sino el administrador del sitio web.
- Permiten la creación de otros usuarios del gestor con distintos roles y permisos desde una sección del menú del backend.



(a) Administradores en Drupal (b) Administradores en Magento

Figura 4.9: Usuarios administradores, Drupal vs Magento

4.4.4 Pedidos

Como ya se ha comentado antes, esta es una funcionalidad esencial, todos los gestores de contenidos deberán generar un módulo que permita la gestión de pedidos y todo lo que esto implica, como podría ser controlar el pago o la logística de envío.

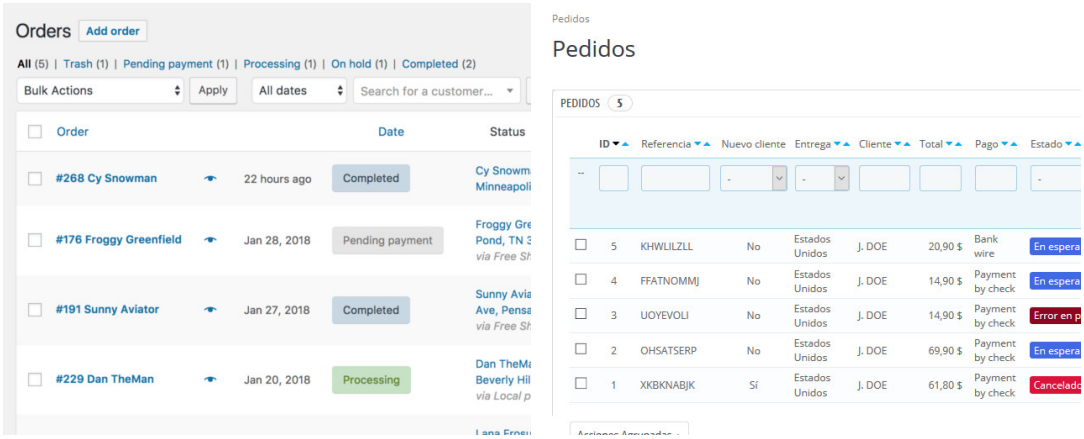
Gestión de pedidos

La gestión de pedidos supone la visualización de las compras realizadas por parte de los clientes a través de la página web y todo tipo de datos relacionados, como la información del cliente que ha realizado el pedido, la modificación o consulta del transportista que se encargará del envío, el método de pago o distribución que se utilizará, etc...(Véase las diferencias en este apartado entre distintos gestores de contenido en la tabla 4.6 y la figura 4.10).

Tabla 4.6: Comparativa: Pedidos, WooCommerce vs Prestashop

WooCommerce	Prestashop
• Los pedidos pueden listarse y consultarse en las últimas versiones de WooCommerce. • Los pedidos muestran en un primer vistazo la información relacionada y enlaces para ampliar dicha información. • Los pedidos tienen una serie de estados fijos que incluye el gestor de contenido. • Los estados de cada pedido se modifican automáticamente si se instala una plantilla que lo permita. • El administrador puede enviar por correo toda la información relacionada con el pedido desde el menú del back-end. • Los pedidos están relacionados con pasarelas de pago y transportistas. • La información que incluye, además de la de por defecto WooCommerce, dependerá del tema instalado, permitiendo añadir incluso una lista de deseos.	• Los pedidos pueden verse desde un menú dedicado en el panel de administración. • Los pedidos muestran toda la información del pedido, del cliente y de terceros relacionados (transportistas, tipo de pago, etc.) con enlaces para ampliar dicha información. • Los pedidos se organizan por una serie de estados que pueden ser creados por el usuario a partir de su última versión (Prestashop 1.7). • Los pedidos tienen información que otros gestores no tienen, como si el pedido es para regalo, si se necesita la generación automática de albaranes de entrega, etc. • Los pedidos están relacionados con pasarelas de pago y transportistas. • Los pedidos tienen más de un menú en el panel de administración, puesto que también se guardan los carritos de compra que no han llegado a completarse.

Es en este apartado es donde el gestor desarrollado tendrá más limitaciones con respecto a lo que existe en la actualidad. La gestión de pedidos dentro de la página web generada se podrá realizar sin problemas puesto que depende del propio gestor, sin embargo, la integración de funcionalidades dependientes de terceros como transportistas y pasarelas bancarias suponen un problema mayor puesto que nuestro gestor es un producto nuevo y actualmente sin promotores.



(a) Pedidos en WooCommerce (b) Pedidos en Prestashop

Figura 4.10: Gestión de pedidos, WooCommerce vs Prestashop

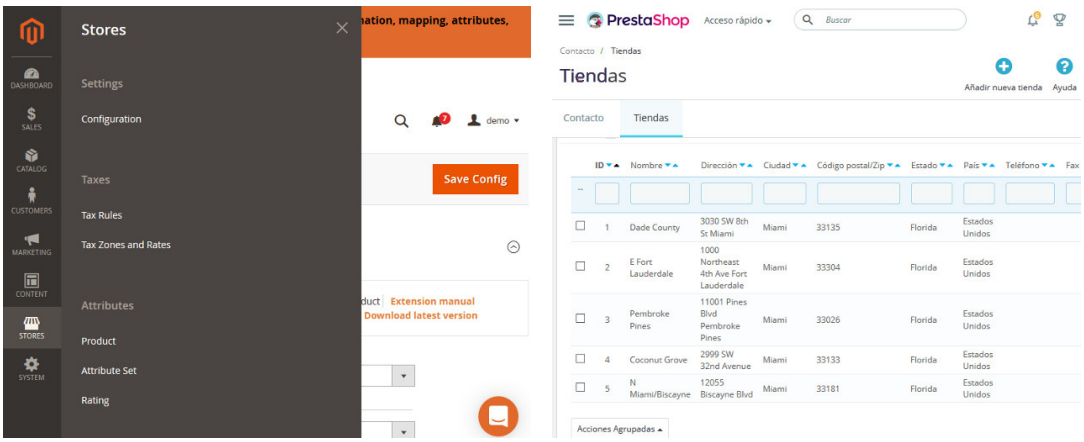
La problemática se debe a que requiere la conexión con empresas que a priori no ofrecen servicios de integración gratuitos, por lo que se deberían establecer contactos y hacer desarrollos específicos de conexiones con sistemas de terceros, no siendo este un objetivo del proyecto en este punto.

4.4.5 Establecimientos

Los establecimientos o tiendas no es un objetivo principal de un gestor sin embargo otorga un elemento diferenciador y permite crear nuevos tipos de gestores de contenido al permitir añadir distintos productos asignados a cada establecimiento o incluso crear un gestor que tenga productos de distintas empresas.

Creación de tiendas

Existen pocos gestores de contenido que permitan cumplir la función de crear distintos establecimientos en una misma tienda, aunque existen módulos que añaden esta funcionalidad en prácticamente todos. (Véase las diferencias en este apartado entre distintos gestores de contenido en la tabla 4.7 y la figura 4.11).



(a) Establecimientos en Magento (b) Establecimientos en Prestashop

Figura 4.11: Gestión de pedidos, WooCommerce vs Prestashop

Tabla 4.7: Comparativa: Establecimientos, WooCommerce vs Prestashop

Magento	Prestashop
• Las tiendas no son elementos dentro del propio gestor, sino que cada dominio/subdominio en el que está instalado Magento supone una tienda. • Las tiendas pueden ser configuradas para que los distintos subdominios tengan ciertos aspectos en común. • Existen módulos que recogen el concepto de tienda como establecimiento con localización y productos propios (e.g. Store Locator for Magento 1). • Las tiendas pueden ser simuladas a través de atributos en los productos (es una práctica habitual).	• Las tiendas pueden crearse, modificarse y eliminarse desde un subapartado en el menú de contacto. • Las tiendas tienen Nombre, Dirección, Ciudad, Código postal/Zip, Estado, País, Teléfono, Fax y si la tienda está activada o no. • Las tiendas tienen Nombre, Dirección, Ciudad, Código postal/Zip, Estado, País, Teléfono, Fax y si la tienda está activada o no. • Las tiendas tienen el total del catálogo, no se puede definir si los productos solo existen en una tienda salvo arreglos con funcionalidades.

En el gestor de contenido a crear pretendemos añadir un apartado destinado por completo a los establecimientos, en el que definir toda la información básica del mismo, a su vez, en esta primera versión o en versiones futuras se aspira a asignar a los productos la tienda en la que el producto esté actualmente.

CAPÍTULO 5

RESULTADOS

5.1 REQUISITOS DEL GESTOR DE CONTENIDO A CREAR

A continuación se detallarán todos los aspectos que han sido considerados importantes para desarrollar este trabajo tras realizar el estudio sobre los gestores de contenido más utilizados, lo cual ha tenido una influencia a la hora de establecer, tanto en el objetivo final del gestor, como las funcionalidades que se desean que ofrezca.

5.1.1 Requisitos del trabajo

Los requisitos funcionales y no funcionales que debe tener el sistema desarrollado se nombrarán divididos en sus dos grandes apartados, el frontend y el backend. El objetivo principal del frontend es mostrar en un navegador el aspecto de la página web con los formularios que el administrador haya elegido, esto significa que la distribución y contenido no será directamente dependiente del gestor, aunque se intentará en lo posible mantener un diseño adaptable. Por otra parte, el aspecto y distribución del backend será fijo, por lo que uno de los objetivos será la correcta distribución y visualización de su contenido, sin embargo, este es un objetivo secundario puesto que aquí deberá permitirse modificar todos y cada uno de los aspectos que se mostrarán en el frontend y automatizar en todo lo posible los distintos procesos de una tienda.

1. **Visualizar la página web creada con el gestor**
   (a) **RE.Front.Vis.1:** Capacidad para ver todos los aspectos creados desde el gestor en un navegador. Ver historia de usuario 5.1
   (b) **RE.Front.Vis.2:** Capacidad para que los usuarios no registrados puedan ver las páginas públicas de la web. Ver historia de usuario 5.1
   (c) **RE.Front.Vis.3:** Capacidad para que los usuarios no registrados puedan ver el catálogo de la empresa. Ver historia de usuario 5.1
   (d) **RE.Front.Vis.4:** Capacidad para que cualquier usuario de la web pueda ver la lista de entradas (posts) creadas por el administrador de la página así como cada una de las entradas individualmente. Ver historia de usuario 5.1
   (e) **RE.Front.Vis.5:** Capacidad para que los usuarios no registrados puedan registrarse como usuarios de la web. Ver historia de usuario 5.2
   (f) **RE.Front.Vis.6:** Capacidad para que los usuarios no registrados puedan iniciar sesión en la web. Ver historia de usuario 5.2
   (g) **RE.Front.Vis.7:** Capacidad para que los usuarios registrados o con la información necesaria para el tipo de compra puedan iniciar el proceso de compra/solicitud/recogida de uno o varios productos/servicios. Ver historia de usuario 5.3

2. **Personalizar la página web desde el backend**

   (a) **RE.Back.Pers.1:** Capacidad para personalizar de forma simple y sencilla la página web. Ver historia de usuario 5.8

   (b) **RE.Back.Pers.2:** Capacidad para modificar la información básica de la empresa/organización que use el gestor. Ver historia de usuario 5.8

   (c) **RE.Back.Pers.3:** Capacidad para modificar la imagen corporativa de la empresa (logotipo, colores, etc) en toda la página independientemente de la apariencia elegida. Ver historia de usuario 5.8

   (d) **RE.Back.Pers.4:** Capacidad para elegir de forma independiente el aspecto de cada parte de la página web de forma simple y visual. Ver historia de usuario 5.9

   (e) **RE.Back.Pers.5:** Capacidad para configurar la funcionalidad e información a mostrar de cada apartado de la web a través de una selección de formularios/plantillas. Ver historia de usuario 5.9

   (f) **RE.Back.Pers.6:** Capacidad para añadir recursos visuales y añadir código propio (Añadir imágenes, código CSS y código Javascript entre otros). Ver historia de usuario 5.10

   (g) **RE.Back.Pers.7:** Capacidad para crear/modificar/eliminar paginas estáticas sin utilizar código. Ver historia de usuario 5.10

   (h) **RE.Back.Pers.8:** Capacidad para crear menús y grupos de menús de enlaces personalizados. Ver historia de usuario 5.9

3. **Creación de un catálogo desde el backend**

   (a) **RE.Back.Cat.1:** Capacidad para crear un catalogo sencillo y visual. Ver historia de usuario 5.9

   (b) **RE.Back.Cat.2:** Capacidad para crear un catálogo de productos/servicios sin límites. Ver historia de usuario 5.11

   (c) **RE.Back.Cat.3:** Capacidad para elegir el aspecto y disposición del catálogo. Ver historia de usuario 5.9

   (d) **RE.Back.Cat.4:** Capacidad para crear un árbol de categorías apropiado para cada empresa. Ver historia de usuario 5.11

   (e) **RE.Back.Cat.5:** Capacidad para crear productos genéricos. Ver historia de usuario 5.12

   (f) **RE.Back.Cat.6:** Capacidad para definir propiedades de cada categoría a través de atributos. Ver historia de usuario 5.12

   (g) **RE.Back.Cat.7:** Capacidad para listar, buscar y ver el total de productos/servicios promocionados (stock) de la empresa. Ver historia de usuario 5.13

4. **Definir distintos establecimientos desde el backend**

   (a) **RE.Back.Shp.1:** Capacidad para crear, modificar o eliminar establecimientos de cualquier tipo, quizas solo lugar físico o incluso distintas empresas en una organización. Ver historia de usuario 5.16

5. **Controlar cualquier tipo de usuario de la web, tanto clientes como administradores desde el backend**

   (a) **RE.Back.Usr.1:** Creación de un usuario administrador único que controle la página web. Ver historia de usuario 5.4

   (b) **RE.Back.Usr.2:** Capacidad para crear otros usuarios administradores que tengan acceso a distintos apartados de la página en función de su rol. Ver historia de usuario 5.5

(c) **RE.Back.Usr.3:** Capacidad para crear, modificar o eliminar roles de usuarios que den privilegios a los distintos elementos del menú de administración. Ver historia de usuario 5.5

(d) **RE.Back.Usr.4:** Capacidad para crear clientes a través de la página web. Ver historia de usuario 5.2

(e) **RE.Back.Usr.5:** Capacidad para listar el número de clientes así como la información básica del mismo. Ver historia de usuario 5.6

(f) **RE.Back.Usr.6:** Capacidad para modificar datos básicos de clientes (no relacionados con cualquier tipo de información de carácter personal). Ver historia de usuario 5.7

6. **Gestionar y ver el estado de pedidos realizados a través de la web desde el backend**

(a) **RE.Back.Ord.1:** Capacidad para ver el total de pedidos actualmente en curso para poder actuar en consecuencia. Ver historia de usuario 5.14

(b) **RE.Back.Ord.2:** Capacidad para ver el total de pedidos cerrados en la web para tener constancia de ello. Ver historia de usuario 5.14

(c) **RE.Back.Ord.3:** Capacidad para identificar inequívocamente a un pedido, así como relacionar dicho código a su estado, productos y precio. Ver historia de usuario 5.15

5.2 REQUISITOS DE LA TIENDA SOLIDARIA

La tienda solidaria que se pretende crear supone el uso del gestor de contenido, así como la creación de formularios y módulos específicos para cumplir todas las necesidades de la organización, por esto, la tienda solidaria tendrá una serie de requisitos sobre el gestor de contenido adicionales.

1. **RE.UCLM.Sess.1:** Capacidad para registrarse en la página web a través de la API de inicio de sesión de la UCLM. Ver historia de usuario 5.17

2. **RE.UCLM.Sess.2:** Capacidad para utilizar los datos de usuario de la UCLM y evitar así que se se deban introducir datos por parte del usuario. Ver historia de usuario 5.17

3. **RE.UCLM.Ord.1:** Capacidad para generar códigos de pedido para identificar lo que ha comprado cada usuario. Ver historia de usuario 5.15

4. **RE.UCLM.Shp.1:** Capacidad para diferenciar distintos establecimientos y puntos de recogida de productos. Ver historia de usuario 5.16

5.3 HISTORIAS DE USUARIO

Tabla 5.1: HdU: VST-WEB-VIEW

VST-WEB-VIEW	
Cómo...	Visitante o usuario de la página
Quiero...	Ver todo el contenido al que el administrador de la página web me permita acceder desde mi navegador.
Para...	Obtener información sobre la página web, la empresa que la ha creado, los productos/servicios que promociona, la información de contacto, etc...

Tabla 5.2: HdU: VST-CLT-SESSION

VST-CLT-SESSION	
Cómo...	Visitante de la página
Quiero...	Tener la capacidad de registrarme e iniciar sesión en la página web.
Para...	Hacer uso de todas las funcionalidades que esto incluya en los formularios utilizados en la página web (compras, comentarios, envío de correos, etc).

Tabla 5.3: HdU: VST-CLT-ORDER

VST-CLT-ORDER	
Cómo...	Usuario de la página
Quiero...	Poder iniciar el proceso de compra/solicitud de un producto/servicio que se promocione en la página web.
Para...	Poder obtener dicho producto/servicio.

Tabla 5.4: HdU: BCK-ADM-LOGIN

BCK-ADM-LOGIN	
Cómo...	Administrador.
Quiero...	Tener un acceso exclusivo y controlado al panel de administrador.
Para...	Reducir riesgos y aumentar la seguridad de los datos sensibles y visuales de la página web creada.

Tabla 5.5: HdU: BCK-ADM-CREATE

BCK-ADM-CREATE	
Cómo...	Administrador.
Quiero...	Poder crear roles de usuarios que tengan acceso a funciones concretas del panel de administrador.
Para...	Añadir usuarios que trabajen en apartados concretos sin tener acceso al total de funciones del panel de administrador.

Tabla 5.6: HdU: BCK-CLT-LIST

BCK-CLT-LIST	
Cómo...	Administrador.
Quiero...	Ver la información de los clientes registrados en la página web.
Para...	Poder consultar los pedidos de cada cliente, su información de contacto.

Tabla 5.7: HdU: BCK-CLT-MOD

BCK-CLT-MOD	
Cómo...	Administrador.
Quiero...	Modificar la información de contacto de los clientes registrados.
Para...	Corregir errores u olvidos por parte de los clientes registrados en la web.

Tabla 5.8: HdU: BCK-PER-VISUAL

BCK-PER-VISUAL	
Cómo...	Administrador.
Quiero...	Poder modificar de la forma más sencilla posible toda la información de la empresa, tanto textual como visual.
Para...	Mantener la imagen corporativa de la empresa y mostrar información de contacto para poder así hacer distinguible la información que se muestra en la web.

Tabla 5.9: HdU: BCK-PER-FORMS

BCK-PER-FORMS	
Cómo...	Administrador.
Quiero...	Poder cambiar el aspecto, estructura y funcionalidades de la página web de forma individual por cada elemento manteniendo cierta homogeneidad.
Para...	Poder elegir en todo momento como quiero que sea y las funcionalidades que tendrá cada apartado de la web.

Tabla 5.10: HdU: BCK-PER-CUSTOM

BCK-PER-CUSTOM	
Cómo...	Administrador.
Quiero...	Añadir y crear en la página web contenido propio ya sea textual o visual.
Para...	Hacer la página más personal y permitir añadir contenido único y adecuado.

Tabla 5.11: HdU: BCK-CAT-CREATE

BCK-CAT-CREATE	
Cómo...	Administrador.
Quiero...	Crear un catálogo completo y que se adapte por completo a mis productos o servicios sean cuales sean.
Para...	Permitir a los clientes navegar entre los distintos tipos de productos y poder crear y organizar los productos creados más fácilmente.

Tabla 5.12: HdU: BCK-CAT-PRODUCT

BCK-CAT-PRODUCT	
Cómo...	Administrador.
Quiero...	Crear, modificar, eliminar, personalizar y representar los productos/servicios de los que disponga mi empresa.
Para...	Informar a los clientes y permitirles ver y posteriormente comprar los productos/servicios de los que se dispone.

Tabla 5.13: HdU: BCK-CAT-LIST

BCK-CAT-LIST	
Cómo...	Administrador.
Quiero...	Ver el total de productos que actualmente se muestran en la página web.
Para...	Poder seleccionarlos, comprobar la consistencia del catálogo con el stock de la empresa o facilitar la modificación de los mismos.

Tabla 5.14: HdU: BCK-ORD-LIST

BCK-ORD-LIST	
Cómo...	Administrador.
Quiero...	Ver todos los pedidos que han sido realizados en la web, tanto los que están en curso como los ya cerrados.
Para...	Modificar el estado de los pedidos así como corregir errores u olvidos por parte de los clientes registrados en la web.

Tabla 5.15: HdU: BCK-ORD-CONTROL

BCK-ORD-CONTROL	
Cómo...	Cliente.
Quiero...	Tener un método para identificar mi pedido y poder ver su estado, los productos que incluye, el precio total, etc...
Para...	Tener control del pedido realizado en todo momento y poder reclamar y reconocer mi pedido en caso de problemas.

Tabla 5.16: HdU: BCK-SHP-OP

BCK-SHP-OP	
Cómo...	Administrador.
Quiero...	Listar, crear, modificar o eliminar establecimientos dentro de la propia tienda.
Para...	Poder definir distintas localizaciones en las que se puedan comprar los productos que se ofrecen, a la vez que definir donde está disponible cada uno de ellos.

Tabla 5.17: HdU: UCLM-SES-API

UCLM-SES-API	
Cómo...	Visitante de la página.
Quiero...	Iniciar sesión en la página utilizando los datos de usuario de la UCLM.
Para...	No tener que crear un nuevo usuario en la página web y permitir el uso de los datos de usuario de la UCLM para la funcionalidad de la página.

5.4 DISEÑO Y DESARROLLO DEL CMS GENERAL

Con el objetivo de cumplir los distintos objetivos y requisitos que requiere este trabajo se ha creado Lurión, un gestor de contenido simple e intuitivo, que pretende ser una herramienta de utilidad para todas aquellas pymes que aún no tienen una página con la que poder promocionarse en la web

o realizar actividades de comercio electrónico. Lurión es de libre uso y nace con la intención de competir con el resto de gestores de contenido que existen en la actualidad, aportando una forma distinta de crear contenido. Se basa en un diseño modular, que es comprensible e intuitivo y que aporta mucha libertad a la hora de su personalización. Estas características quieren animar a los desarrolladores para que pasen a formar parte de una comunidad de creadores (o al menos establecer los primeros pasos para ello).

Figura 5.1: Logotipo del nuevo gestor de contenido Lurión

A continuación se concretarán las principales diferencias y elementos distintivos que se esperan conseguir, para ello se parte del estudio realizado anteriormente sobre los gestores de contenido más utilizados en España. Se hablará, tanto de requisitos técnicos o funcionales, como no funcionales o de elementos relacionados con la interacción del usuario. Debido a la complejidad del producto, no hablaremos de todos y cada uno de los elementos que incluirá el gestor de contenido, sino que nombraremos los aspectos que puedan contener decisiones importantes de diseño, dejando a parte los elementos más simples o con un diseño ya estandarizado.

Para englobar el mayor número de funcionalidades que incluirá el proyecto y no hacer este apartado excesivamente largo al definir cada una de ellas, se creará una estructura jerárquica lo más simple posible en la que agrupar los distintos requisitos. Además se relacionarán con las historias de usuario que les correspondan (como ya se ha comentado, un gestor de contenido tiene un gran número de funcionalidades, prácticamente todas las que una página web puede llegar a ofrecer, por ello no nombraremos todas en este documento, solo las más relevantes). La estructura narrativa será la siguiente:

1. **Diseño de la interfaz de usuario de la aplicación web**

   En este apartado nos vamos a referir al diseño visual que tendrá la aplicación, una vez sea instalada en el navegador. Por ello, puesto que este trabajo habla de un gestor de contenido, se dividirá en dos apartados:

   - La interfaz del backend o panel de administración, que tendrá una estructura y diseño fijos.
   - La interfaz del frontend o página web que dicho gestor creará, que dependerá de los formularios que utilice el administrador de la página, aunque si podrá definirse cómo se organiza y que posibilidades se facilitan para dichos formularios.

2. **Diseño e implementación de la lógica de la aplicación**

   Aquí hablaremos de todas las clases de dominio que existirán en el servidor web, el procesamiento de los datos introducidos por el usuario, la manipulación de la información almacenada en una base de datos…

5.4.1 Interfaz de usuario: Panel de administración

Esta interfaz tendrá una distribución y aspecto fijo, no configurable por parte del administrador de la página salvo por aspectos relacionados con la imagen corporativa de la empresa (como colores

en botones, textos, fondo, etc. o el logotipo de la empresa).

La estructura escogida para esta parte del gestor se divide en cuatro áreas principales que son la cabecera, el menú de funcionalidades, las opciones de dicho menú y el área de trabajo. Véase la figura 5.2

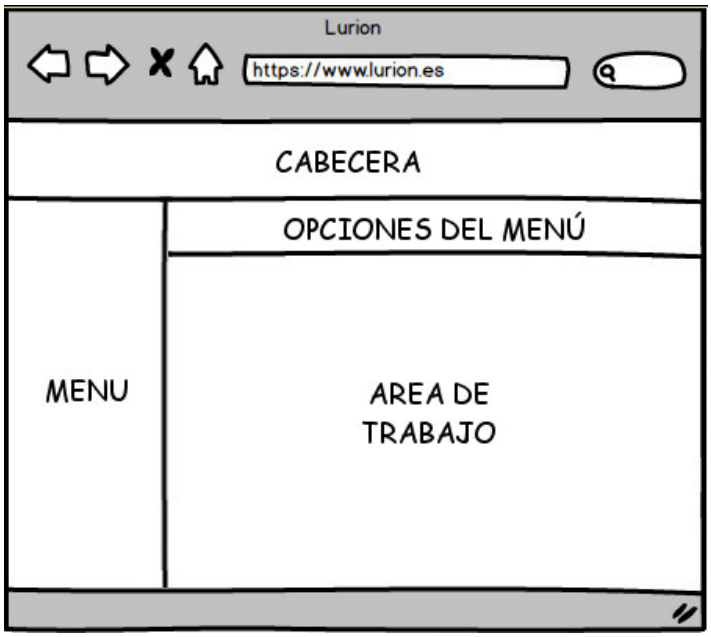


Figura 5.2: Logotipo del backend de Lurión

Cabecera

La cabecera del panel de administración se compondrá de dos elementos, el logotipo del gestor de contenido, adaptado a los colores corporativos de la empresa para guardar cierta relación con el estilo definido por el usuario para el frontend y una sección que recoja qué usuario está registrado en el panel y un botón que permitirá el cierre de sesión. La única funcionalidad de este apartado es el cierre de sesión del administrador, el resto es puramente visual. Véase la figura 5.3

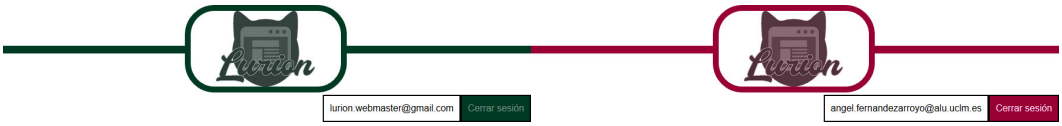


Figura 5.3: Cabecera usando los colores por defecto y los de la UCLM

Menú de funcionalidades

Existirá un menú lateral que tendrá todas las funcionalidades que contiene este gestor de contenido, agrupado en seis secciones para hacerlo más intuitivo. Dentro de cada sección habrá una serie de menús, en los que, en función de la tarea sobre la que estemos actuando, podremos realizar cambios, configurar o personalizar la página web.

Una vez elegido el menú existirán distintas funcionalidades sobre las que se podrá navegar desde el área **Opciones del menú**, para ejemplificar esto, consideremos la sección ***Personalización***, uno de sus menús podría ser ***Recursos*** (ambos se mostrarán en el menú lateral) y por último las opciones de este menú serán ***Imágenes***, ***Código CSS*** y ***Código JS***. Véase la figura A.6

Área de trabajo

. Esta área será la más importante de todo el backend, aquí se realizarán todas las operaciones que permiten modificar cualquier aspecto del gestor instalado. En ella se mostrarán todos los formularios con los que podrá realizar modificaciones el administrador del gestor.

A continuación mostraremos las diferentes opciones sobre las que este gestor actuará y mostraremos los enlaces a los prototipos que se realizaron para diseñar el aspecto final de la aplicación.

Inicio

- **Inicio:** Lurión tendrá una página de inicio que verá todo usuario, sea cual sea su rol. Es por ello, que será lo más simple posible, teniendo el objetivo de evitar errores o excesos de código en función del rol del usuario que ha iniciado sesión (como ocurre en Prestashop). Esto evitará cargas de código y consultas innecesarias (e.g. La primera página que puede ver el usuario es el stock y deberán obtenerse de la base de datos productos y su información aunque el usuario no vaya a trabajar con ellos). Véase la figura [A.1].

Personalización

- **Imagen corporativa:** El peso principal a la hora de crear una imagen corporativa, más allá incluso del contenido y estructura de la página, es el uso de un logotipo que represente a la empresa y un conjunto de colores (a ser posible únicos e identificativos) que haga recordar a la empresa al verlos, este aspecto es el que se tratará en este apartado.

  Lurión permitirá modificar la información de contacto de la empresa, enlaces a redes sociales, colores, logotipo de la empresa…desde un mismo menú. Toda esta información será principalmente utilizada por los formularios y algunos de estos datos se almacenarán en archivos como constantes (como los colores en constantes CSS). Esto ayudará a cumplir los requisitos [2a] [2b] [2c]. Véanse las figuras [A.2] [A.3] [A.4].

- **Metadatos:** En Lurión pretendemos permitir desde el propio panel de administración la modificación de estos datos, sin embargo, en esta primera versión, no se estudiará la opción de tener metadatos distintos en cada una de las páginas, sino unos generales que el administrador pueda definir y se hereden al resto (sí se pretende implementar la inclusión de metadatos en cada página, entrada o producto creado desde el propio panel de administración en versiones futuras del gestor de contenido). Esto ayudará a cumplir el requisito [2b]. Véase la figura [A.2].

- **Aspecto:** Lurión permite modificar el aspecto de todos y cada uno de los elementos distintivos de la web, desde el header o cabecera hasta el pie o footer, desde los elementos que incluirá la página principal hasta la página de perfil del usuario, permitiendo incluir menús horizontales o verticales y adecuando automáticamente el resto del contenido, compartiendo todos ellos un estilo común al compartir colores y estilos a través de la inclusión de clases CSS comunes y todo esto de una forma lo más simple posible para los creadores de aspectos. Esto ayudará a cumplir los requisitos [2a] [2d] [2e] [3c]. Véase la figura [A.5].

- **Recursos:** Para personalizar cualquier página estática creada por el administrador (tanto su aspecto como su funcionalidad) o incluso para modificar ciertos aspectos de los formularios instalados, es necesario insertar recursos propios, como podrían ser imágenes de las instalaciones de la empresa, código CSS que se comparta en toda la página, código JS para crear sencillos sliders o dropdowns en páginas estáticas…para ello se ha creado esta sección en el gestor. Esto ayudará a cumplir el requisito [2f]. Véanse las figuras [A.6] [A.7] [A.8].

- **Menu:** La creación de un menú es vital si se quiere permitir al usuario crear una página web adaptada a sus necesidades, ya que sin ellos la única opción sería una estructura de páginas fija y estricta. Por esto se permite la creación, modificación y eliminación de menús, con tantos enlaces como se elija, además se ha añadido un datalist para recomendar posibles enlaces (página de inicio, contacto, catálogo, etc.) y con la opción de crear direcciones relativas mediante códigos (e.g. si en un enlace se introduce el código *[domain_url]* se considerará como el dominio actual de la página y se evitará así introducir direcciones absolutas). Esto ayudará a cumplir el requisito [2h]. Véase la figura [A.9].

- **Páginas estáticas y entradas:** En cuanto a la creación de contenido personalizado por el usuario se pretende crear una división clara entre Páginas estáticas y Entradas/Noticias, a través de, por ejemplo, dos menús diferentes en el panel de administración. Dentro de cada uno se intentará crear un editor de texto similar a los que cualquier usuario utiliza comúnmente (como podría ser el editor de texto de Word o LibreOffice) añadiendo elementos visuales y configurables (añadir margenes laterales, hipervínculos, imágenes, etc.). Por otra parte, añadir un editor de código HTML sencillo y hacer que ambos editores se modifiquen mutuamente, permitiendo así al usuario adaptar de una forma u otra su contenido a sus preferencias. Esto ayudará a cumplir el requisito [2g]. Véanse las figuras [A.10] [A.11] [A.12] [A.13].

Catálogo

- **Categorías:** En el gestor a desarrollar no pretendemos crear un sistema de categorías demasiado ambicioso, sino un catálogo simple, sencillo y genérico que tenga una representación visual para que al administrador de la página le resulte sencillo navegar entre categorías. Sin embargo, si con suerte este gestor de contenido pasa a ser conocido y utilizado ampliaremos este catálogo añadiendo opciones como añadir imágenes representativas de cada categoría o definir tasas o descuentos por categoría. Esto ayudará a cumplir los requisitos [3a] [3b] [3d]. Véase la figura [A.15].

- **Productos:** Con Lurión la idea principal es crear distintos menús para, por una parte, listar los productos creados y, por otra, añadir, modificar o eliminarlos. A parte de esto pretendemos dividir la inserción y modificación de productos en tres apartados: la información básica que todo producto debe tener (nombre, descripción, precio, etc.); imágenes del producto e imagen principal; y por último atributos que el usuario podrá crear para definirlo. Esto ayudará a cumplir los requisitos [3a] [3b] [3e] [3f] [3g]. Véanse las figuras [A.14] [A.16] [A.17].

Tiendas

- **Tiendas:** Existirá una sección dedicada por completo a cualquier acción para la creación, modificación o eliminación de tiendas. Cada tienda tendrá un identificador único con el que relacionarlas a roles o productos, a su vez se podrá añadir su nombre y sus datos de contacto. Esto ayudará a cumplir los requisitos [4a] [4]. Véase la figura [A.18].

Usuarios

- **Administradores:** Existirá siempre un usuario administrador que tendrá acceso a todas las secciones del gestor, este usuario no podrá ser modificado y tendrá un trato especial. A parte de este usuario, teniendo los permisos necesarios, se podrá crear tantos usuarios como se quiera y estos podrán tener tantos roles como se desee. Esto ayudará a cumplir los requisitos [5a] [5b]. Véase la figura [A.19].

- **Roles:** Los roles estarán compuestos por un nombre, un identificador único y un conjunto de permisos. En la opción que contiene este menú se mostrarán todas las opciones de todos los submenús, ordenados del mismo modo que en el menú lateral y mediante varios checkbox se seleccionará cada opción a la que el rol tenga acceso. Esto ayudará a cumplir el requisito [5b] [5c]. Véase la figura [A.20].

- **Clientes:** Existirá una lista que incluya todos los clientes registrados en la página web, se mostrará su información de contacto y su código de cliente para poder consultarlos en caso de envío de productos o para cualquier otra operación. Por supuesto sus datos de inicio de sesión estarán cifrados y no serán representados. El registro e inicio de sesión de los clientes será a través del frontend. Esto ayudará a cumplir los requisitos [5d] [5e] [5f]. Véase la figura [A.21].

Pedidos

- **Pedidos:** Al igual que en los apartados de stock o clientes este apartado listará todos los pedidos en curso o ya finalizados que se hayan realizado a través de la página web. Estos pedidos se listarán con la misma información, sea del tipo que sea, puesto que este gestor está preparado para que todos los productos tengan un mismo proceso de pago y entrega o distinto (podría existir una tienda con productos físicos y virtuales con distintas formas de entregar los productos) y se identificarán por un código inequívoco. En cualquier caso se mostrará el tipo de compra, productos del pedido, precio total (puesto que los precios pueden cambiar en el tiempo pero eso no debería afectar al pedido)...Esto ayudará a cumplir los requisitos [6a] [6b] [6c] [3]. Véanse las figuras [A.22] [A.23].

- **Correo:** Lurión gestionará todos los envíos automáticos de correos de la misma manera, en la versión actual del gestor no se tendrán distintas configuraciones en función del tipo de correo a enviar. Se creará una configuración para el envío de correos que utilizará todo el gestor, configuración que se definirá en esta sección. Véase la figura [A.24].

Elementos adicionales

- **Inicio de sesión:** Este gestor tendrá que gestionar dos tipos de inicios de sesión, uno destinado a los Administradores y otro a los Clientes. En esta sección hablaremos del inicio de sesión de los Administradores para acceder al backend de Lurión.

  El índice del backend (index.php) controlará si el usuario que está intentando acceder al panel de administrador tiene una sesión iniciada en el servidor o no y si ésta es correcta.

  La forma de comprobar si el usuario ya ha iniciado sesión será un proceso en dos pasos, el primero recogerá si existe una sesión relacionada con el usuario que intenta acceder y en caso de que exista, el tipo de esta sesión (buscando así si es una sesión de administrador o cliente), si este paso es satisfactorio se avanzará al segundo paso, en este se comprobará si la dirección IP del usuario que intenta acceder coincide con la dirección previamente almacenada y relacionada con la sesión, si se superan estas comprobaciones se cargará el backend en su página de inicio y en caso de que cualquiera de estas comprobaciones falle se mostrará el formulario de inicio de sesión.

- **Instalación:** Al igual que cualquier otro gestor de contenido, antes de que este funcione es necesario realizar un proceso de instalación, en él se establecerán y crearán las bases que necesita esta herramienta para su correcto funcionamiento. Para iniciar este proceso tan solo será necesario acceder desde el navegador a la carpeta ***install*** que el gestor de contenido tiene y una vez hecho esto se tendrán que completar cinco sencillos pasos:

1. *Lenguaje:* En primer lugar se definirá el lenguaje en el que se instalará la herramienta. Los lenguajes disponibles se recogen de forma automática ya que la herramienta dispone de un control automático de lenguajes instalados. En la versión por defecto, existen tres opciones.
   - Todos: También podría entenderse como automático, Lurión, salvo que se establezca un lenguaje por defecto para el gestor, recogerá del navegador que esté usando el cliente la cookie de lenguaje y si existe un archivo de traducciones para dicho idioma mostrará la página usando dichas traducciones, en caso contrario se utilizará el Español.
   - Español: Lenguaje instalado por defecto, se recogerá el archivo de traducciones en Español (`strings_es.xml`) sin tener en cuenta el lenguaje por defecto del navegador del visitante.
   - Inglés: Lenguaje instalado por defecto, se recogerá el archivo de traducciones en Inglés (`strings_en.xml`) sin tener en cuenta el lenguaje por defecto del navegador del visitante.

   Actualmente la herramienta está completamente preparada para ser representada en distintos idiomas, podría crearse distintas traducciones de un mismo producto o categoría gracias al valor "lang"de la base de datos, sin embargo, el proceso de traducción a través de interfaz gráfica en el panel de administración del gestor de contenido no está incluido en esta primera versión. Véase la base de datos y una explicación más extensa de esto en la figura 5.4(a)
2. *Conexión a la base de datos:* Lurión no puede funcionar sin su propia base de datos, por ello, para su correcto funcionamiento, tanto en aspecto (puesto que se almacenan los formularios escogidos), como en funcionamiento (puesto que como cualquier otro gestor se almacenan sus productos, categorías…) la conexión con una base de datos, en el servidor de instalación o en cualquier otro, es vital. En este segundo paso el administrador deberá introducir sus datos de conexión y comprobar que el gestor pueda conectarse, una vez está comprobación esté hecha, pasaremos al siguiente paso. Véase este paso en la figura 5.4(b)
3. *Creación de la base de datos:* Este es un proceso automático y bastante rápido en el que el usuario no deberá hacer nada, simplemente dejar que los archivos SQL de la carpeta de instalación se ejecuten sobre la base de datos que se creó o sobre la que se conectó el gestor en el paso anterior. Véase este paso en la figura 5.4(c)
4. *Creación del usuario administrador:* Lurión tendrá un usuario administrador único, con acceso a todos los elementos del gestor sin ningún tipo de restricciones y este usuario será creado en este paso. El proceso es muy simple, será definir el correo que tendrá el administrador y crear una contraseña con una serie de requisitos de seguridad, una vez que esté creado se permitirá acceder al paso final. Véase este paso en la figura 5.4(d)
5. *Fin:* Una vez se llega a este paso el gestor de contenido estará completamente instalado, por lo que esta página tan solo servirá para mostrar los enlaces del frontend y del backend de la página web creada y avisar que sería recomendable eliminar la carpeta ***install*** para evitar posibles errores.

5.4.2 Interfaz de usuario: Web creada

La página web que creará Lurión tendrá tres índices y por tanto tres tipos de páginas distintos, estos serán un índice situado en la raíz de la página, destinado a la página principal, otro índice en un subdirectorio llamado ***custom*** para todo lo relacionado con páginas estáticas y entradas, tanto las creadas por el administrador como las páginas estáticas del gestor (página de contacto, inicio de

(a) Instalación, inicio

(b) Instalación, conexión con la base de datos

(c) Instalación, creación base de datos

(d) Instalación, creación del administrador

Figura 5.4: Pasos del proceso de instalación de Lurión

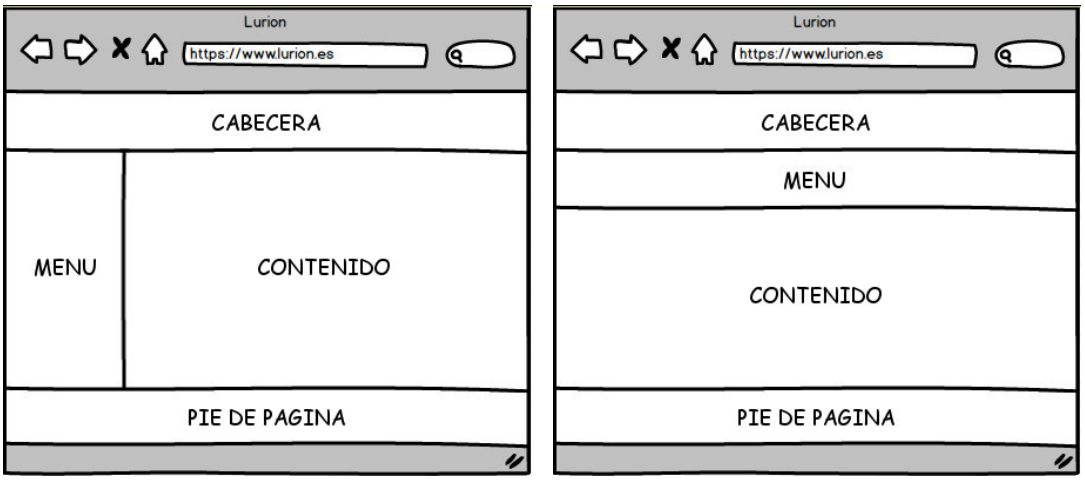
sesión, etc…) y un último índice destinado a todo lo relacionado con el catálogo (listado de productos, visualización del carrito…).

Aunque existan secciones distintas todos los índices heredan contenido unos de otros, es por ello que todas las páginas compartirán la misma distribución, modificando tan solo el bloque dedicado al contenido, esta distribución estará compuesta por cuatro secciones llamadas cabecera, menú (horizontal o vertical), contenido y pie de página. Esto ayudará a cumplir los requisitos [1a] [1b] [1c] [1d]. Véanse las figuras 5.5

Aclaraciones previas

Antes de explicar el contenido que tendrá cada sección de la distribución de la página que creará Lurión, hay que entender una serie de puntos sobre el funcionamiento interno del gestor y para analizarlos será necesario conocer antes ciertos aspectos del dominio. Estos puntos son:

- **Independencia de secciones**: Este gestor de contenido permitirá que cada administrador pueda cambiar de forma independiente cada una de las secciones, lo que supone que debe existir una forma de hacer ver al gestor qué formulario ha sido instalado en cada apartado.



(a) Frontend con menú lateral (b) Frontend con menú horizontal

Figura 5.5: Frontend de la página creada, Menú Horizontal y Vertical

Ante esto existirá una tabla en la base de datos que guardará el nombre de cada formulario para cada sección del gestor.

- **Independencia de formularios**: Cada formulario tendrá su propia estructura y funcionalidad, nada impide que el botón de inicio de sesión esté en el pie de página o que los productos se muestren en la cabecera, estos diseños se dejan a decisión del creador y existe completa libertad en cuanto a la creación de formularios, cada uno de ellos podrá tener su propio código CSS y JS y se podrán llamar a una serie de métodos PHP y Javascript ya creados para facilitar cada una de sus tareas.
- **Facilidades y preparación**: A la hora de llamar a un formulario, aunque exista esa libertad de funcionalidad de la que se hablaba, se presupone que los creadores utilizarán el formulario de catálogo para listar productos y el formulario de producto para mostrar la información de un producto dado, por ello, a la hora de llamar a cada formulario, se ha creado una variable PHP llamada $data que contendrá de forma sencilla y sin tener que recurrir a ningún tipo de método información relacionada con el formulario en cuestión (datos de un producto si existe su identificador en la URL del navegador, todos los productos que el cliente quiere comprar en el formulario del carrito…)
- **Funciones callback**: Para incluir los formularios se utiliza la sentencia *include* de PHP, sin embargo, para evitar que las variables PHP utilizadas en la preparación de la información necesaria para el tipo de formulario o de sesión puedan ser utilizadas libremente por parte del creador de formularios, se utiliza una función callback que analiza el contenido del formulario. Esto tiene una doble intención ya que también se analiza si existen códigos (como el código `[domain_url]` ya mencionado anteriormente) y en una versión futura se pretende utilizar dicha función para determinar las funciones del dominio que se podrán utilizar y las que no.
- **Plantilla CSS y JS propia**: Lurión, por necesidad en el panel de administración, tiene una plantilla CSS que permite crear estructuras y permite a los contenidos adaptarse a pantallas con baja resolución, sin embargo, esta también puede facilitar mucho el trabajo a creadores de plantillas ya que permite crear una estructura similar a bootstrap en la que se organice el contenido en filas y columnas (doce columnas ya que el número doce tiene múltiples divisores).

A continuación se expondrá con un ejemplo sencillo cual es el proceso a través del cual se incluye un formulario. El más claro y sencillo de todos es el pie de página y por ello, para clarificar el proceso lo máximo posible, será el que se analizará:

Listado 5.1: Recoger formulario, functions.php

```
#Para incluir el formulario dedicado al pie de página se llamará a esta función
function get_page_footer(){
    #Esta función recogerá el identificador elegido para la sección (footer).
    $footer = page_selected("footer");
    if(!empty($footer)){
        include_once_form(array_shift($footer), "/form", "php");
    }
    #Excepto el pie de página, cada sección incluye una particularidad:
    #En la cabecera se controla el cierre de sesión,
    #en el menú se averigua si será un menú lateral u horizontal, etc.
}
#Esta función será común para cualquier sección o las páginas estáticas/entradas.
#Se evita así arrastrar variables y se controla el contenido del archivo.
function include_once_form( $url_part, $form_name, $extension,
    $data = NULL ){
    $url = server_root() . $url_part . $form_name . "." .
        $extension;
    if (file_exists($url)){
        #En esta versión se controla si el archivo contiene algún código.
        ob_start("replace_file_codes");
        include_once $url;
        ob_end_flush();
    }else{
        include return_template("not_found");
    }
}
function replace_file_codes($buffer){
    return (str_replace("[domain_url]", server_address(),
        $buffer));
}
```

Cabecera

En la cabecera, aun teniendo en cuenta que existe libertad por parte de los creadores, se prepara el formulario para que controle el inicio y cierre de sesión y el acceso al carrito.

Esta sección fue la primera que se llevo a cabo por lo que todo lo relacionado con el inicio y cierre de sesión de usuarios se pretende reformar (intentar facilitar a los creadores de plantillas dichas tareas), sin embargo, el estado y la forma de operar actual es la siguiente:

Listado 5.2: Recoger formulario de cabecera, functions.php

```
function get_page_header(){
  if(isset($_POST['logout'])){
    if (isset($_SESSION['userEmail']) ){
      include_class("LoginFactory");;
      LoginFactory::log_out_client($_SESSION['userEmail']);
    }
  }
  $header = page_selected("header");
  if(!empty($header)){
    include_once_form(array_shift($header), "/form", "php");
  }
}
```

De aquí puede verse que el creador de plantillas, para permitir al usuario cerrar sesión, deberá utilizar un ***form*** como el utilizado en el formulario de cabecera por defecto:

Listado 5.3: Funcionalidad propia de la cabecera, form.php

```
<form action="<?php␣$_SERVER['PHP_SELF'];?>" id="form_login"
    method="post">
    <button class="base-element␣base-button␣float-right␣
        base-no-selection" type="submit" name="logout" id="logout"
        onclick="this.form.submit()">
        <i class="default-header-icon␣fa␣fa-sign-out"></i> <?php
            echo strings("Login", "out"); ?>
  </button>
</form>
```

Por último determinar que implementar un acceso y representación del carrito supone tener en cuenta varias consideraciones:

1. Existe una función llamada ***products_in_cart()*** que devolverá el número de productos que estén en el carrito.
2. Existe una función llamada ***link_of_interest($key)*** que devuelve direcciones URL de distintos enlaces de interés de la página (inicio, página de contacto…) y si la variable $key es ”cart” te llevará a la página que visualiza el estado actual del carrito.
3. Existe una función llamada ***get_profile_page()*** que te llevará a la página en la que se verá el formulario elegido para el perfil del usuario. Este enlace también podría obtenerse usando la clave ”profile” en la función anteriormente nombrada.

Menú

En el menú se controlará de forma previa si el menú será horizontal o vertical, siguiendo la distribución mostrada en las figuras 5.5. Esta división en realidad es puramente conceptual ya que ambos tipos de menús siguen el mismo funcionamiento y elaboración, lo único que cabe destacar es una metaetiqueta que se debe añadir en la plantilla llamada ***content_width*** que definirá el ancho que tendrá el contenido, es decir, el número de columnas.

La plantilla CSS anteriormente nombrada se utiliza para el esqueleto de la disposición de la página, por ello conocer el valor de esta variable también determinará el ancho del menú (si content_width tiene un valor de 9, el contenido tendrá un ancho de 9 columnas y el menú tendrá un ancho de 12-9=3 columnas), surgiendo así la diferencia entre un menú horizontal y vertical.

- **Menú horizontal: <meta name=’content_width’ content=’12’>**
- **Menú vertical: <meta name=’content_width’ content=’10’>**

Cabe destacar que existe una función para facilitar el trabajo a los creadores de plantillas en el que, dado el id de uno de los menús creados en el panel de administración, devuelve todos sus enlaces:

Listado 5.4: Funcionalidad propia del menú, form.php

```
#Esta función devolverá un vector con dos posiciones:
#La primera determinará si en la función se ha producido algún error
#La segunda devolverá todos los enlaces del menú introducido
$menu_items = get_menu_items_by_id($menu_id);
if ($menu_items[0]){
  $menu_items = $menu_items[1];
  ?>
  <div id="main-menu-resp" class="full-row␣form-responsive-mobile">
    <div id="default-menu-responsive" class="hide">
      <ul class="default-holder-resp">
```

```
11          <center>
12            #Los enlaces del menú se mostrarán de la siguiente forma:
13            #[Posición en el menú, [nombre del enlace, enlace]]
14            <?php foreach ($menu_items as $key => $value){ ?>
15              <li class="default-element-resp">
16                <a class="default-link-resp" href="<?php␣echo␣
                      $value[1];␣?>">
17                  <h4 class="default-text-resp"><?php echo
                        $value[0]; ?></h4>
18                </a>
19              </li>
20            <?php } ?>
21          </center>
22        </ul>
23      </div>
24    </div>
25  }
```

Contenido

Esta sección será la más distinta de todas con diferencia, ya que, en función del índice y el enlace seleccionado cambiará completamente, teniendo esto en cuenta hablaremos por separado de cómo se comportará el contenido en cada selección:

- **Índice en la raiz**: El comportamiento del contenido en este índice será el más estable puesto que hace referencia a la página principal. La página principal se dividirá en cinco elementos, ***"Top", MiddleTop", "Middle", "MiddleBottom", "Bottom"*** que hacen referencia a la posición que tendrá, pudiendo añadir un formulario en cada uno. El resto de formularios comúnmente hacen referencia a una sección como podrían ser el listado de entradas, el formulario de fin de compra…, sin embargo aquí se podrán definir formularios con funciones concretas (mostrar productos, un slider, bloques de información, hacer uso de un webservice…) para mostrarlas en cualquiera de las posiciones disponibles. Véase la figura 5.6

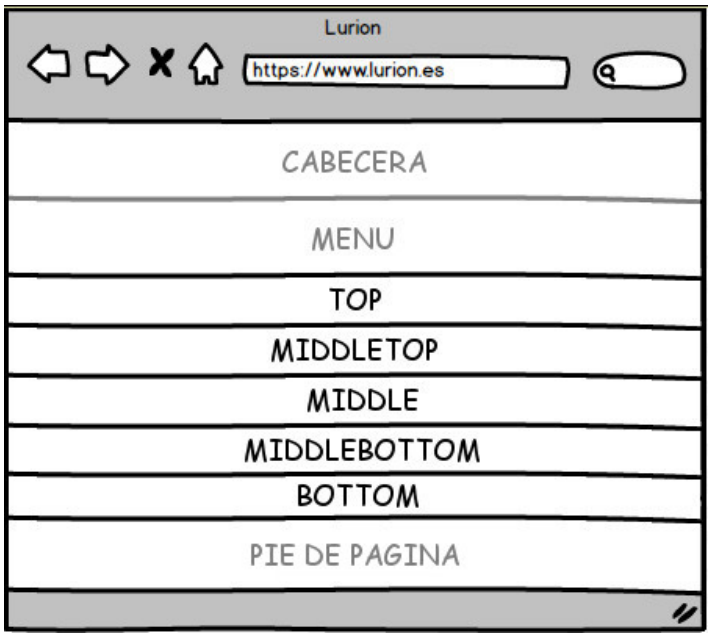


Figura 5.6: Distribución por secciones de la página principal

- **Índice del catálogo**: Este índice englobará cuatro casos diferenciados: el catálogo como listado de productos, la página de producto y su información relacionada, el carrito y la confirmación de la compra/interés por los productos/servicios en el carrito. Todos ellos recogerán el formulario seleccionado por el administrador en función del identificador de página situado en la dirección URL a la que se acceda.

Listado 5.5: Recoger formulario de catálogo, functions.php

```
#Al acceder al índice del catálogo siempre se llamará a esta función
function get_page_catalogue(){
  include_class("ProductFactory");
  #La variable ''id'' definirá el formulario que se cargará
  $page = get_url_variable("id");
  #En caso de que la variable no exista se mostrará la lista de productos
  if (is_null($page)){
    $catalogue = page_selected("catalogue");
    if(!empty($catalogue)){
      include_once_form(array_shift($catalogue), "/form",
          "php");
    }
  #Si la variable tiene el valor ''view'' mostrará el carrito
  }elseif ($page == "view"){
      #Antes de mostrar el carrito se comprobará hay una sesión iniciada
      #Nombrar que hay que controlar los accesos directos a la página
    if (user_logged()[0]){
      get_page_cart();
    }else{
      go_to_homepage();
    }
  #El valor ''payment'' por otro lado mostrará el proceso de compra
  }elseif ($page == "payment"){
    if (user_logged()[0]){
      get_page_payment();
    }else{
      go_to_homepage();
    }
  #Si la variable tiene otro valor habrán dos opciones:
  }else{
    $productFactory = new ProductFactory();
    $product_data = $productFactory->get_product_by_id($page,
        lang());
    #Primera opción:
    #El ''id'' no representa ningúna página ni producto
    if(isset($product_data["error"])){
      go_to_homepage();
    #Segunda opción:
    #El ''id'' será un código de producto y se mostrará su información
    }else{
      $product = page_selected("product");
      if(!empty($product)){
        include_once_form(array_shift($product), "/form",
            "php", $product_data);
      }
    }
  }
}
```

- **Índice en las páginas estáticas y entradas**: El índice del que hablaremos a continuación englobará, por lo general, una gran porción de la página web. Toda la información de la empresa/organización/particular en cuestión se mostrará a través de páginas estáticas y en caso de crear un blog prácticamente todo lo que se vea en la página serán entradas. Además,

este índice también incluirá una serie de páginas preestablecidas (enlaces de interés). Vease la tabla 5.18

Tabla 5.18: Enlaces de interés y palabras reservadas definidas por Lurión

Página	Dirección
Página de contacto	[domain_url]/custom/?page=contact
Inicio de sesión	[domain_url]/custom/?page=login
Página de registro	[domain_url]/custom/?page=register
Perfil de usuario	[domain_url]/custom/?page=profile
Lista de entradas	[domain_url]/custom/?post=blog

Como puede verse en la tabla referenciada, la dirección URL que utilizará este índice será común e identificará si se trata una entrada o una página estática creada por el usuario a través de una variable (***post*** para páginas estáticas y ***page*** para entradas) y su valor determinará si se debe incluir un formulario ya que es una página creada por el gestor o si simplemente es el identificador de la entrada/página a mostrar.

Por supuesto, los creadores de contenido también tendrán aquí facilidades para el desarrollo de sus formularios, aunque no directamente puesto que serán funciones a las que se podrá acceder desde cualquier formulario (recoger los datos de cliente, solicitar el inicio de sesión a través de una petición ajax…). Esto ayudará a cumplir los requisitos [1e] [1f] [1g] [1] [2].

Pie de página

El pie de página será, de todas las secciones de la distribución, la más sencilla de explicar y desarrollar puesto que no necesita ningún tipo de preparación previa o datos del dominio, es por eso que ya ha sido parcialmente explicada en el inicio de la sección. Las funcionalidades habituales en un pie de página suelen ser mostrar de forma minimalista y simple distintos menús de enlaces, uso de webservices para obtener servicios de información, iframes de google maps…Analizando todos estos casos se puede ver que no requiere información (al menos de forma previa) de ningún elemento del dominio del gestor y si se necesitase el creador siempre podrá acceder a las funciones ya preparadas a las que cualquier formulario.

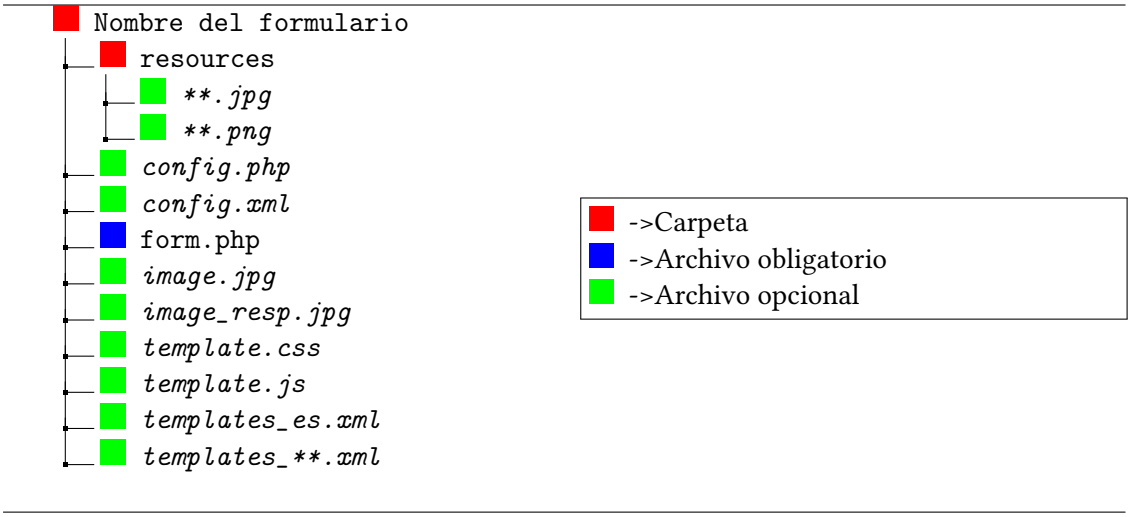
5.4.3 Creación de plantillas

Composición de las plantillas

Bajo el punto de vista parte del creador, las plantillas desarrolladas tendrán una misma metodología, por lo que no serán necesarios conocimientos o reglas de creación distintas entre crear una plantilla de catálogo o una plantilla de inicio de sesión. Por supuesto existirán una serie de métodos y resultados de dichos métodos que deberán conocerse que serán más o menos apropiados y usados en función del formulario pero la estructura para crearlo, independientemente de la función que cumpla. Véase la tabla 5.19

Para la creación de un formulario solo es necesaria la creación de una carpeta cuyo nombre será el título y el identificador que tendrá la plantilla (puesto que no pueden existir dos carpetas con el mismo nombre tampoco ocurrirá esto con las plantillas) y un archivo llamado ***form.php*** que contendrá todo el código y contenido HTML de la página. Aun teniendo en cuenta esto existen otros múltiples archivos opcionales que pueden añadir funcionalidades, apartados visuales o incluso código propio, estos archivos pueden añadir información valiosa y se recomienda su uso. A continuación explicaremos la estructura para permitir entender los distintos aspectos que supone el crear un formulario:

Tabla 5.19: Archivos y subdirectorios de un formulario



- **Configuración** Los formularios, en términos generales, necesitarán conocer ciertas preferencias del usuario para el correcto funcionamiento o la correcta adecuación de su contenido (número de productos que se desean mostrar por página en el catálogo, texto que tendrá el bloque de información…), es por ello por lo que se ha creado todo un sistema que facilite a ambas partes, creadores y administrador de la página, el configurar el formulario.

  - ***Archivo de configuración "config.php":*** Este archivo opcional definirá qué elementos del formulario podrán ser modificados por el administrador de la página, en caso de que no exista este archivo la opción de configuración simplemente no se mostrará. Véase la figura A.5

    Se ha creado un formato para poder definir los distintos elementos configurables del formulario, manteniendo una misma estructura sea cual sea el tipo de elemento necesario. Este formato se basa en la creación vectores que almacenarán variables, siendo cada variable una representación de lo que se quiera configurar, esta variable almacenará ciertos datos fijos independientes al tipo, como el identificador del elemento a crear o su texto explicativo y otras variables que almacenarán datos dependientes del tipo (URL de la imágen, placeholder de un input, máximo número de caracteres en un textarea, etc.) a su vez se podrá determinar en qué orden se mostrará cada elemento en el formulario de configuración que este archivo creará.

    A continuación se mostrará un ejemplo global para ver su funcionamiento y posteriormente se mostrarán las variables y estructuras a definir para cada tipo de elemento (select, input, textarea, image, color).

Listado 5.6: Ejemplo simple, config.php

```
#Todos los archivos de configuración definirán su grupo y nombre
# $group: A qué se dedica este formulario (listado de entradas)
# $name: Cuál es el nombre de este formulario (default)
$group = "post_list";
$name = "default";
#Las variables que almacenarán cada uno de los elementos
#configurables podrán tener cualquier nombre
$title = [
     #El ''id'' será común en todos, independientemente del tipo
     #de elemento (select, input, etc) y servirá para recoger
     #posteriormente el valor que haya introducido el usuario
     #en este campo.
  "id" => "tags_title",
```

```
    #''name'' también será común y será el texto que vea el
    #administrador para saber a qué se refiere este elemento.
    #La función ''templates_string()'' servirá para recoger el
    #texto en función del idioma del administrador.
  "name" => templates_string($group, $name,
      "posts_title"),
  "placeholder" => templates_string($group, $name,
      "posts_title_placeholder")];
#La siguiente variable hace referencia al número de entradas que
#se podrán ver en el formulario, y como puede verse sigue la
#misma estructura que cualquier otro campo de texto (aunque
#podría sustituirse por un select con valores numéricos).
#Actualmente el control de errores en los valores introducidos
#por el administrador en estos casos será responsabilidad del
#creador de formularios
$post_number = [
  "id" => "posts_per_page",
  "name" => templates_string($group, $name,
      "posts_number"),
  "placeholder" => templates_string($group, $name,
      "posts_number_placeholder")];
#Una vez estén definidos todos los elementos configurables
#se deberán añadir al vector que se corresponda con su tipo
$input_array = [$title, $post_number];
#Si existe el caso en que se cree un elemento pero no se
#añada a su correspondiente vector este elemento no existirá
#a ojos de Lurión
```

Existirán tantos vectores en los que almacenar los elementos como tipos de elementos, así pues, tendremos las variables $select_array, $input_array, $textarea_array, $image_array y $color_array. El orden en que todos estos elementos se mostrarán en el menú de configuración será secuencial y su orden será el mismo en el que se han nombrado los distintos tipos. Puesto que este orden puede suponer inconsistencias y problemas en la comprensión de lo que se está guardando existirá otro vector llamado $order_array en el que se podrá añadir cada elemento de forma individual y definir así el orden en que se mostrará cada uno de ellos.

Listado 5.7: Vectores según tipo y orden en el formulario de configuración, config.php

```
$group = "header";
$name = "ejemplo";
#Un campo obligatorio a la hora de crear un ''select'' es las
#opciones, estas podrán ser definidas en una variable o podrán
#utilizar una palabra reservada:
# options == variable =>Opciones personalizadas
# options == ''menu'' =>Enlaces de un menú del backend
# options == ''parts'' =>Posiciones de la página principal
$ejemplo_opciones_select = [
  ["id" => "true",
  "value" => templates_string($group, $name, "yes")],
  ["id" => "false",
  "value" => templates_string($group, $name, "no")]];
$ejemplo_select = [
  "id" => "ej_select",
  "name" => templates_string($group, $name, "ej_select"),
  "options" => $ejemplo_opciones_select];
$ejemplo_input = [
  "id" => "ej_input",
  "name" => templates_string($group, $name, "ej_input"),
  "placeholder" => templates_string($group, $name,
      "input_ph"),
  "max_chars" => 15];
```

```
$ejemplo_textarea = [
  "id" => "ej_textarea",
  "name" => templates_string($group, $name,
     "ej_textarea"),
  "placeholder" => templates_string($group, $name,
     "textarea_ph"),
  "max_chars" => 760];
$ejemplo_imagen = [
  "id" => "ej_imagen",
  "img_id" => "nombre_imagen_directorio",
  "name" => templates_string($group, $name, "ej_imagen")];
$ejemplo_color = [
  "id" => "ej_color",
  "name" => templates_string($group, $name, "ej_color")];
$select_array = [$ejemplo_select];
$input_array = [$ejemplo_input];
$textarea_array = [$ejemplo_textarea];
$image_array = [$ejemplo_imagen];
$color_array = [$ejemplo_color];
#Al crear este vector y asignar de este modo sus variables el orden
#en que se mostrarán en el panel de configuración del formulario
#será el orden inverso al orden por defecto
$order_array = [$ejemplo_color, $ejemplo_imagen,
    $ejemplo_textarea, $ejemplo_input, $ejemplo_select];
```

– ***Archivo de información "config.xml":*** La configuración que complete el usuario o que el creador de contenido quiera añadir por defecto debe guardarse en algún sitio, y este almacenamiento debe estar preparado para todo tipo de datos y cualquier cantidad de ellos, por ello se ha recurrido al uso de un archivo XML. Dentro de este archivo se almacenarán como etiquetas los identificadores de los elementos configurables, guardando en ellos su valor, a su vez es posible que dependiendo del tipo sea necesario añadir otras etiquetas auxiliares (de uso exclusivo por el gestor, ni el creador ni el administrador tendrán por qué conocerlas ni utilizarlas).

Listado 5.8: Ejemplo básico, config.xml

```
<?xml version="1.0"?>
<!-- Se utilizará el ejemplo previo en el que se veían
    todos los tipos de elementos y como crear el vector de
    orden -->
<config>
    <!-- Como puede verse se utilizarán como etiquetas
        para guardar su valor sus identificadores -->
    <ej_select>true</ej_select>
    <!-- En algunos casos son necesarias variables
        auxiliares como en el ''select'', que se debe
        definir qué tipo de opciones tendrá:
       mode == 0 => Opciones personalizadas
       mode == 1 => Enlaces de un menú del backend
       mode == 2 => Posiciones de la página principal -->
    <ej_select_mode>0</ej_select_mode>
    <ej_input>Lorem ipsum</ej_input>
    <ej_textarea>Lorem ipsum dolor sit amet, consectetur
        adipiscing elit, sed do eiusmod tempor
        incididunt.</ej_textarea>
    <!-- Las imágenes por otro lado no necesitan guardar
        ningún  valor, puesto que la imagen se almacenará
        en la carpeta ''resources'' del formulario con el
        identificador que se definió por parte del creador
        en el archivo ''config.php'', sin embargo sí será
        necesario guardar su extensión -->
    <ej_imagen>png</ej_imagen>
```

```
    <ej_color>#95b8a6</ej_color>
</config>
```

– ***Tratamiento de la configuración en el backend:*** Una vez se ha creado el archivo de configuración por parte del creador, es necesario el procesamiento de esta información para conseguir una página comprensible y sencilla de utilizar por parte del administrador de la página web. Este proceso se llevará a cabo dentro del menú de aspecto en el panel de administración, tras pulsar el correspondiente botón de configuración. Véase la figura A.5

Al pulsar el botón de configuración se recargará la página incluyendo una variable en la dirección URL llamada ”config” que indicará el formulario a personalizar.

Como se ha visto, existen distintos tipos de elementos que pueden utilizarse (entradas de texto, imágenes…) y por tanto también deben existir distintas formas de introducir la información necesaria, es por ello que son tan necesarios los vectores por tipo. Se analizará cada uno de estos vectores ($input_array, $input_color…) y en función del vector en el que esté el elemento tendrá una forma de mostrarse u otra.

Listado 5.9: Creación del formulario de configuración, form_appearance_config.php

```
#Se conoce dónde está el archivo de configuración del formulario
#en cuestión utilizando las variables ''config'' y ''name'' de la
#dirección URL, haciendo referencia a la sección del formulario
#y al nombre del formulario respectivamente
include server_root() . "/admin/templates/" . $config .
    "/" . $name . "/config.php";
#Al incluir dicho archivo se añaden todos los vectores creados por
#el creador de contenido, en este paso se comprobará si existe el
#vector que determina el orden o no
if (isset($order_array)){
    #Se recogen todos los elementos del vector y se comprueba su tipo
    foreach($order_array as $generic_element){
        #Como se ve a continuación hay un método para cada tipo:
        if(isset($select_array) and
            in_array($generic_element, $select_array)){
          html_select($generic_element, $config, $name);
        }
        if(isset($input_array) and
            in_array($generic_element, $input_array)){
          html_input($generic_element, $config, $name);
        }
        if(isset($textarea_array) and
            in_array($generic_element, $textarea_array)){
          html_textarea($generic_element, $config, $name);
        }
        if(isset($image_array) and
            in_array($generic_element, $image_array)){
          html_image($generic_element, $config, $name);
        }
        if(isset($color_array) and
            in_array($generic_element, $color_array)){
          html_color ($generic_element, $config, $name);
    } }
}else{
    if (isset($select_array)){
        foreach ($select_array as $select_element){
            html_select($select_element, $config, $name);
    } }
    if (isset($input_array)){
        foreach ($input_array as $input_element){
            html_input($input_element, $config, $name);
    } }
```

```
37      if (isset($textarea_array)){
38          foreach ($textarea_array as $textarea_element){
39              html_textarea($textarea_element, $config,
                    $name);
40      } }
41      if (isset($image_array)){
42          foreach ($image_array as $image_element){
43              html_image($image_element, $config, $name);
44      } }
45      if (isset($color_array)){
46          foreach ($color_array as $color_element){
47              html_color($color_element, $config, $name);
48  } } }
```

- **Recursos**

  - ***Carpeta "resources":*** Esta carpeta es utilizada principalmente para almacenar imágenes. Las imágenes almacenadas pueden ser utilizadas por el propio creador para algún elemento fijo que tenga el formulario como por ejemplo sería una imagen de fondo, pero también para añadir imágenes que pueda seleccionar el usuario desde su respectivo formulario de configuración del formulario.

  - ***Archivo de estilo "template.css":*** Este archivo permite añadir todos los estilos y clases CSS que quiera añadir el creador de contenido y usar en su plantilla, el simple hecho de tener este archivo no es suficiente, también habrá que llamar a una función ya preparada desde el archivo "form.php" que se explicará posteriormente.

  - ***Archivo de funcionalidades del cliente "template.js":*** Este archivo supone añadir al formulario todas las funciones o código directo, al igual que en el caso anterior para incluir todo el código habrá que llamar a una función ya preparada. El código javascript es la principal razón de que se tenga que llamar a una función en lugar de añadirlo automáticamente, dando así más posibilidades puesto que el añadir código directo puede querer hacerse al principio (crear timeouts, definir variables…) o al final (recoger elementos del árbol DOM por su identificador…).

    Para evitar problemas de rendimiento y velocidad de carga de la página la inclusión de jquery es una llamada asíncrona, es por ello que en este archivo es preciso comprobar que ya ha sido incluido el código antes de usar cualquiera de sus métodos como código directo.

- **Visualización**

  - ***Archivo "image.jpg" y archivo "image_resp.jpg":*** Estos archivos no afectan en absoluto a la funcionalidad de la plantilla, es un elemento puramente decorativo utilizado para mostrar o crear una imagen que represente al formulario en el panel de administración.

    Actualmente, en los formularios creados para este gestor las imágenes hacen referencia a cómo se verá una vez instalado, por ello existen dos imágenes, una para demostrar cómo se verá en pantallas de resolución alta y otra para ver cómo será la distribución en pantallas de baja resolución, aunque la imagen la seleccionará el creador, por lo que puede ser cualquier cosa. Véase la figura 5.7

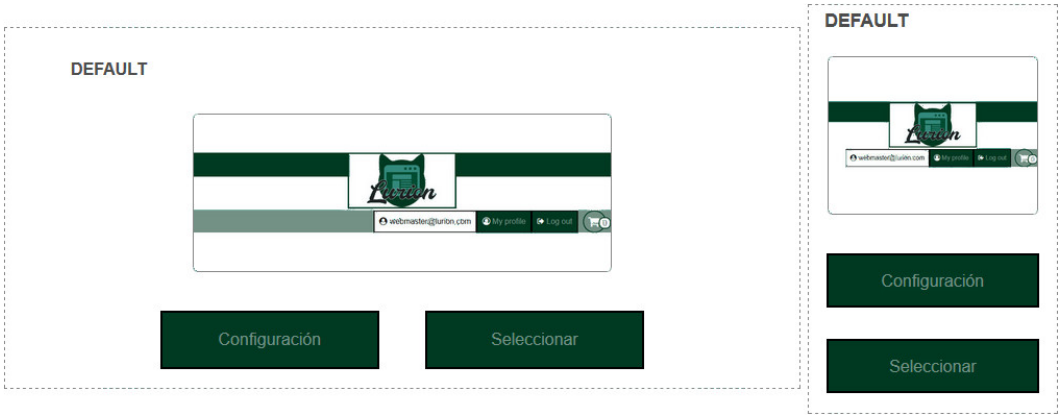


Figura 5.7: Formulario de cabecera, Alta resolución vs baja resolución de pantalla

- **Lenguajes**
  - ***Archivo de traducciones "templates_**.xml":*** Este gestor está perfectamente preparado para utilizar cualquier tipo de idioma y esto puede verse en que:
    * El panel de administración tiene todos sus textos agrupados por etiquetas que pueden ser traducidas a cualquier idioma, estando traducidas por defecto a inglés y español.
    * El contenido estático y los textos que añade el administrador por lógica los hará en el idioma que haya elegido para el gestor de contenido. También está perfectamente preparado para soportar múltiples idiomas gracias a su valor "lang" en las tablas de la base de datos (para traducciones de categorías, productos, atributos…) y a la capacidad de crear y guardar entradas/páginas distintas en distintos idiomas (aunque aun no existe una opción de externalización en el menú de administración, lo que hace que usuarios sin conocimientos técnicos no puedan utilizar aún una página multilenguaje).
    * Las plantillas disponen de sus propios archivos de traducción con la capacidad de utilizar sus propios textos en los lenguajes que vean apropiados, estos archivos se llamarán ***templates_[lang].xml***.

    Estos archivos siguen la misma lógica que la usada para traducir a distintos idiomas el panel de administración. Se recogerá o bien el lenguaje fijado en la instalación de Lurión o bien el lenguaje por defecto que tiene guardado el visitante en su navegador, una vez teniendo este valor ("es" para Español, "en" para Inglés, etc.) se buscará el archivo que contenga las traducciones correspondientes, strings_[lang].xml para los textos del gestor y templates_[lang].xml para los textos del formulario.

- **Funcionalidad**
  - ***Archivo de contenido "form.php":*** La funcionalidad del formulario así como el contenido HTML es dependiente por completo del creador, lo que significa que no se podrá explicar su contenido o estructura, sin embargo si que se pueden nombrar ciertas funciones desarrolladas para facilitar su labor y ciertos requisitos que deben cumplirse:

    Variables PHP

    * **$actual_dir**: Esta variable es necesaria en todo formulario para indicar dónde están localizados sus archivos. Su contenido será una sentencia fija y su contenido no tendrá ser modificado en ningún momento por el creador, sin embargo, se debe utilizar esta sentencia dentro del archivo form.php puesto que utiliza la palabra reservada ***__FILE__*** de PHP.

      $actual_dir = get_actual_dir(dirname(__FILE__));

Esta variable es utilizada en la gran mayoría de funciones creadas en PHP para el creador (incluir el CSS o JS, recoger los resultados almacenados por el administrador en el menú de configuración…) y es por ello por lo que incluir esta sentencia se considera un requisito prácticamente indispensable

* **$group** y **$name**: Al igual que en el caso de ***"config.php"*** estas variables serán necesarias para utilizar la función ***"templates_string()"*** y recoger así los archivos de traducción. El mantener esta forma de recoger los textos traducidos y no crear un nuevo método, evitando la necesidad de tener estas variables (utilizando en cambio $actual_dir), se debe al objetivo de facilitar al máximo la creación de formularios y no tener que utilizar métodos distintos en función del archivo que se está utilizando.

Funciones PHP

* **get_actual_dir()**: Esta función fue utilizada para darle valor a la variable previamente nombrada ***$actual_dir*** y su objetivo es, dada la dirección absoluta recoger el subdirectorio en el que se encontrarán los archivos del formulario (post_list/default, header/uclm…).

Listado 5.10: Recoger subdirectorio del archivo actual, functions.php

```
function get_actual_dir( $dir ){
    #La variable pasada siempre será dirname(__FILE__) o en su
    #defecto __DIR__.
    #Esta variable devuelve la dirección absoluta del archivo
    $parts = explode("\\", $dir);
    #Se comprobará que estamos hablando de un formulario al ver
    #la profundidad de su directorio.
    #La dirección de las plantillas será
    #(/admin/templates/[seccion]/[nombre del formulario]
    if(count($parts) < 4 ){
      $parts = explode("/", $dir);
    }
    $actual_dir = $parts[count($parts)-2] . "/" .
        $parts[count($parts)-1];
    return $actual_dir;
}
```

* **include_template_css($actual_dir)** y **include_template_js($actual_dir)**: La inclusión de los recursos almacenados en el directorio del formulario, tanto de estilo como de funcionalidad de cliente, se recogerán gracias a estas dos funciones preparadas para ello. Hay que tener en cuenta que no se añadirán como enlaces de descargas sino como etiquetas de estilo y script por lo que se podrá incluir en cualquier momento (en este gestor se recomienda incluir los recursos al principio, pero hay casos concretos en los que añadir el código javascript tras cargar el contenido HTML tiene ventajas).
* **config_info($actual_dir, $identificador)**: Esta función te permitirá recoger toda la información que haya introducido el administrador en el menú de configuración del formulario. Necesitará dos variables, la primera es la que define el subdirectorio para saber de dónde recoger el archivo ***"config.xml"*** y el segundo será el identificador del elemento del que se quiere recoger el valor.

Listado 5.11: Recoger configuración del formulario, functions.php

```
function config_info($dir, $id){
    #Se utilizará la librería de PHP SimpleXML para todo uso de
    #archivos XML
  $use_errors = libxml_use_internal_errors(true);
    #Se accederá al archivo donde se almacena la configuración
```

```
6       #gracias a la variable $actual_dir
7     $config_dir = server_root() . "admin/templates/" .
          $dir . "/config.xml";
8     $xmlAddress = str_replace(server_address(),
          server_root(), $config_dir);
9       #Se realizarán todos las comprobaciones necesarias y se
10      #evitará todo posible error antes de operar con el archivo
11      if (!file_exists($xmlAddress)){
12      $newXML = new SimpleXMLElement('<config>␣
            </config>');
13      $newXmlAddress = str_replace(server_address(),
            server_root(), $config_dir);
14      $file = fopen($newXmlAddress, "w");
15      fwrite($file, $newXML->asXML());
16      fclose($file);
17    }
18    $xml = simplexml_load_file($config_dir);
19    if ( !$xml ) {
20      return NULL;
21    }
22    libxml_clear_errors();
23    libxml_use_internal_errors($use_errors);
24    #Por último se recogerá el valor almacenado en el identificador
25    #pasado, en caso de que no exista se devolverá una cadena vacía
26    $info = $xml->{$id};
27    if($info == NULL){
28      $info = "";
29    }
30    return urldecode($info);
31  }
```

* **get_config_image($actual_dir, $identificador, $nombreImagen)**: Gracias a esta función se podrán obtener las direcciones URL de las imágenes añadidas a través del menú de configuración, tan solo se deberá introducir el directorio del formulario, el identificador para obtener la extensión de la imagen de su archivo ***"config.xml"*** y el nombre de la imagen para recoger su dirección URL.
* **get_logo_url()**: Devuelve la dirección de la imagen almacenada como logo de la empresa en el panel de administración.
* **get_main_language()**: Permite conocer si el usuario ha elegido un lenguaje en el que mostrar todo el contenido de la página o no. Esto en principio no tiene por qué utilizarlo ningún creador, pero permite crear distintas distribuciones o contenido en función del lenguaje.
* **get_currency()**: Devuelve el tipo de moneda que se utilizará en la página (€, $...).
* **has_contact_shop_info()**: Permite saber si el administrador ha introducido algun tipo de información de contacto. Devuelve un valor booleano.
* **get_web_title()**: Devuelve el nombre de la empresa/página web.
* **get_web_description()**: Devuelve la descripción de la empresa/página web.
* **get_web_keywords()**: Devuelve las palabras clave (metacontenido) de la empresa/página web.
* **shop_data($identificador)**: Devuelve toda la información introducida por el administrador en la página web siempre que se incluya el identificador que la represente. Los identificadores que existen actualmente son **shop_name, shop_description, shop_keywords, email, telephone, fax, address, url_facebook, url_twitter, url_instagram, url_linkedin, url_google, url_youtube, url_logo, main_language** y **currency**.

* **get_category_name($id)**: Devuelve el nombre de una categoría en función de su identificador.
* **get_shop_name($id)**: Devuelve el nombre de una tienda en función de su identificador.
* **get_attribute_name($id)**: Devuelve el nombre de un atributo de producto en función de su identificador.
* **get_url_variable($id)**: Esta función será utilizable en todo tipo de formulario ya sea de aspecto o del propio gestor y su objetivo será recoger el valor de una variable que exista en la dirección URL. Es principalmente utilizada en formularios para dividir un listado de productos, imágenes... en distintas páginas aunque también permite crear distintas secciones y páginas dentro de un mismo formulario (como podría ser el perfil de usuario que se divida en datos de usuario, pedidos, etc. y para pasar de una página a otra se utilicen direcciones URL con variables).
* **set_url_variables($name_array, $value_array)**, **set_url_variable($name, $value, $url = NULL)**, **remove_url_variables($name_array)** y **remove_url_variable($name, $url = NULL)**: Todas estas funciones completarán a la función previa. Todas ellas se basarán en obtener una dirección (pensadas principalmente para la dirección de la propia página web) y añadir, modificar o eliminar sus variables. Nombrar que estas funciones no redireccionarán, simplemente devolverán la dirección URL actual o pasada modificada. Las dos primeras funciones añadirán variables o modificarán su valor en caso de que existan y las últimas eliminarán variables, en caso de que existan.
* **links_of_interest()** y **link_of_interest($key)**: Ambas funciones trabajan de forma conjunta para mostrar distintos enlaces de interés existentes por la propia distribución del gestor de contenido.

Listado 5.12: Enlaces de interés, functions.php

```
function links_of_interest(){
    #Se devolverán todos los enlaces de interés en la página como
    #un diccionario.
    #Recordar que la función callback reemplazará el código
    #[domain_url]
    return [
      "home" => [strings("Common", "go_home"),
          "[domain_url]"],
      "contact" => [strings("Common", "contact"),
          "[domain_url]" . "custom/?page=contact"],
      "posts" => [strings("Common", "post_list"),
          "[domain_url]" . "custom/?post=blog"],
      "catalogue" => [strings("Common", "catalogue"),
          "[domain_url]" . "catalogue"],
      "cart" => [strings("Common", "cart"),
          "[domain_url]" . "catalogue/?id=view"],
      "payment" => [strings("Common", "payment"),
          "[domain_url]" . "catalogue/?id=payment"],
      "login" => [strings("Common", "login"),
          "[domain_url]" . "custom/?page=login"],
      "register" => [strings("Common", "register"),
          "[domain_url]" . "custom/?page=register"],
      "profile" => [strings("Common", "profile"),
          "[domain_url]" . "custom/?page=profile"]
    ];
}

```

```
function link_of_interest($key){
    #Al pasar una de las claves del diccionario esta función
    #devolverá el enlace
    $links = links_of_interest();
    $link = $links["home"][1];
    if(array_key_exists($key, $links)){
      $link = $links[$key][1];
    }
    return $link;
}
```

* **get_menu_items_by_id($menu_id)**: Esta función está preparada para recoger todos los enlaces y sus nombres asociados introduciendo el identificador del menú creado en el panel de administración.
* **product_has_main_image($img_url)** y **get_product_main_image($img_url, $display = True)**: Estos métodos permiten comprobar si un producto tiene una imagen principal y obtener dicha imagen. El dato que la función necesita ($img_url) es uno de los que se le otorga al creador de formularios gracias a la preparación previa que se realiza en el índice de catálogo ($data["image"]). Por otro lado la variable $display determinará la forma en que la imagen quiere ser mostrada (si se quiere una dirección URL o su dirección en el directorio de archivos).
* **get_product_images($img_url, $display = True)**: Devuelve todas las imágenes de un producto (excepto la imágen principal) a través de su identificador y al igual que en el caso anterior se puede definir el tipo de dirección en el que mostrar las imágenes.
* **user_logged()**: Permite conocer si la persona que tiene abierta la página en su navegador tiene una sesión iniciada y también mostrará el correo de dicho usuario.
* **get_client_data()**: En el momento en que un usuario inicia sesión se podrá usar este método. Está principalmente pensado para los formularios de perfil de usuario y devolverá toda la información del cliente si tiene su sesión iniciada.
* **get_all_posts($page = 0, $number =)**: Esta función recogerá por defecto todas las entradas que se hayan añadido desde el panel de administración, aunque también podrá añadirse paginación dándole valores a las variables del método ($page indica la página en la que se está buscando y $number el número de entradas por página).
* **get_posts_by_tags($tag, $page = 0, $number =)**: Recogerá todas las entradas que tengan asignada una etiqueta (tag) concreta e igual que en el caso anterior se podrán mostrar entradas por paginación.
* **get_posts_number()**: Devolverá la cantidad de entradas que han sido creadas.
* **get_all_tags()**: El resultado será un vector que tenga todas las etiquetas que han sido creadas y asignadas a alguna entrada por parte del administrador.
* **get_post_short_description($id)**: Esta función está pensada para mostrar la lista de entradas y devolverá los primeros caracteres de la entrada en forma de descripción breve (puramente textual) eliminando etiquetas HTML, imágenes y estilos.

Funciones Javascript

* **getProducts(page, entries, order, response)**: Permite recoger desde índices distintos al del catálogo a través de una solicitud AJAX los productos por página y orden, definiendo a su vez la función que tratará el resultado obtenido por dicha petición. Por ejemplo **getProducts(0, 4, "RAND()", 'catalogueTemplateProducts')** devolverá los cuatro primeros productos recogidos de la base de datos de forma aleatoria

y dichos productos se enviarán a la función javascript del creador del formulario llamada ”catalogueTemplateProducts”.

* **getTotalProducts()**: Informa al creador cual es el total de productos (visibles) almacenados en la página web.
* **changeVisibility(element)**: Permite cambiar la visibilidad (añadir o eliminar la clase CSS ”hide”) pasando el identificador de dicho elemento HTML.
* **completepurchaseFormulary(total_amount, total_types, email)**: Realiza el proceso de compra a través de un formulario en el que se almacena en el panel de administración el pedido en cuestión y se informa al usuario de que dicha compra se ha realizado. Es necesario incluir el precio total del pedido, el total de productos a comprar y el correo en el que notificar el pedido que se ha realizado (comúnmente el correo de la persona encargada de la tienda o del administrador de la página, aunque este es un campo opcional).
* **completepurchasePickUp(total_amount, total_types, email, shop_id)**: Supone el mismo proceso que el caso anterior con la diferencia de que la compra supondrá una recogida en tienda de los productos, por lo que también se tendrá que añadir en el pedido el identificador de la tienda en cuestión.

5.4.4 Lógica de la aplicación

Base de datos

Lurión, al igual que cualquier otro gestor de contenido, debe tener una base de datos en la que guardar todas las variables definidas por el administrador, sus categorías, sus productos, los textos de ambos…esta es la razón de que se haya creado desde cero una base de datos que se adapta totalmente a las necesidades actuales del gestor. Véase la figura 5.8

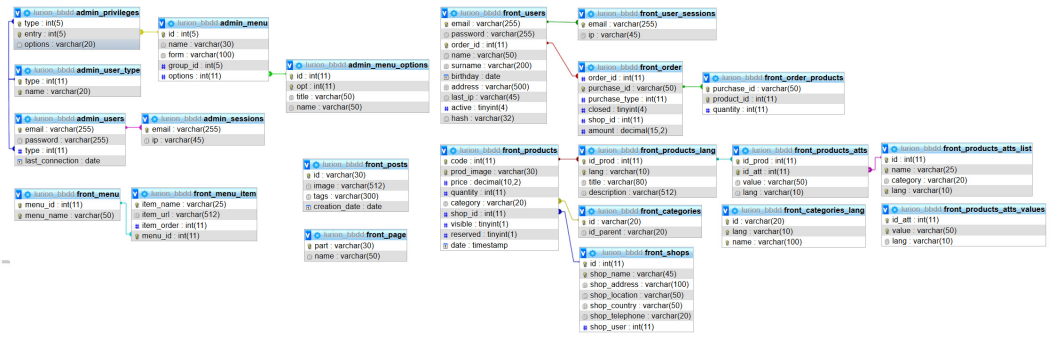


Figura 5.8: Base de datos de Lurión

Patrones

En este desarrollo aplicaremos un patrón modelo – vista – controlador, consiguiendo esta separación de la interfaz gráfica y las interacciones del usuario con el dominio gracias a metodologías híbridas que utilizan factorías y manejadores. Para ejemplificar esto un poco más diremos que:

- **El Modelo** es la representación de la información y el dominio con la que operará la aplicación.
- **La Vista** es la representación gráfica, la interfaz con la que interactuará y trabajará el usuario final.

- **El Controlador** será el manejador de eventos, usualmente llamado e iniciado por acciones determinadas del usuario, invoca peticiones tanto al Modelo como a la Vista.

Siguiendo estos conceptos podemos ver que hemos estado hablando en gran medida de las Vistas a lo largo de los capítulos anteriores (en los que hablábamos de la interfaz de usuario de la web creada y del panel de administración) por tanto en esta sección nos centraremos en las clases Modelo y los Controladores.

En este proyecto la Vista y el Modelo no tendrán en ningún caso una comunicación directa, sin embargo, el controlador podrá obtener toda la información que ésta necesite, cuando lo requiera, ya que es el encargado de otorgar toda la información que el usuario necesite del dominio desde la interfaz. La comunicación entre Vista y Controlador se puede realizar de varias maneras:

- La vista puede instanciar objetos controladores (factorías o manejadores) y llamar a sus distintos métodos.

- En caso de que se necesite la información en tiempo real se realizarán llamadas asíncronas AJAX que realicen peticiones POST a códigos del dominio, estos igualmente instanciarán clases controladoras, una vez resuelta la petición se gestionará sus respuestas en métodos Javascript.

Modelos y Controladores

Como se ha introducido, los Controladores solicitarán información a los Modelos de la aplicación y enviarán dicha información a la Vista, por ello se ha decidido que estas entidades sean identificadas por clases individuales y que cada clase Modelo tenga una comunicación directa con su correspondiente clase Controladora. Salvo excepciones, cada Modelo cumplirá una serie de funciones acotadas sobre una misma temática y cada Controlador estará relacionado con cada una de las clases Modelo. Los controladores por su parte tendrán dos opciones, actuar en forma de factoría (por cada solicitud de información se crea una entidad de la clase Modelo, tras obtener la información necesaria esta entidad se destruirá) o en forma de manejador (se crea la instancia del Modelo como una variable propia de clase y se mantiene su estado mientras exista su Controlador). Véase la figura 5.9

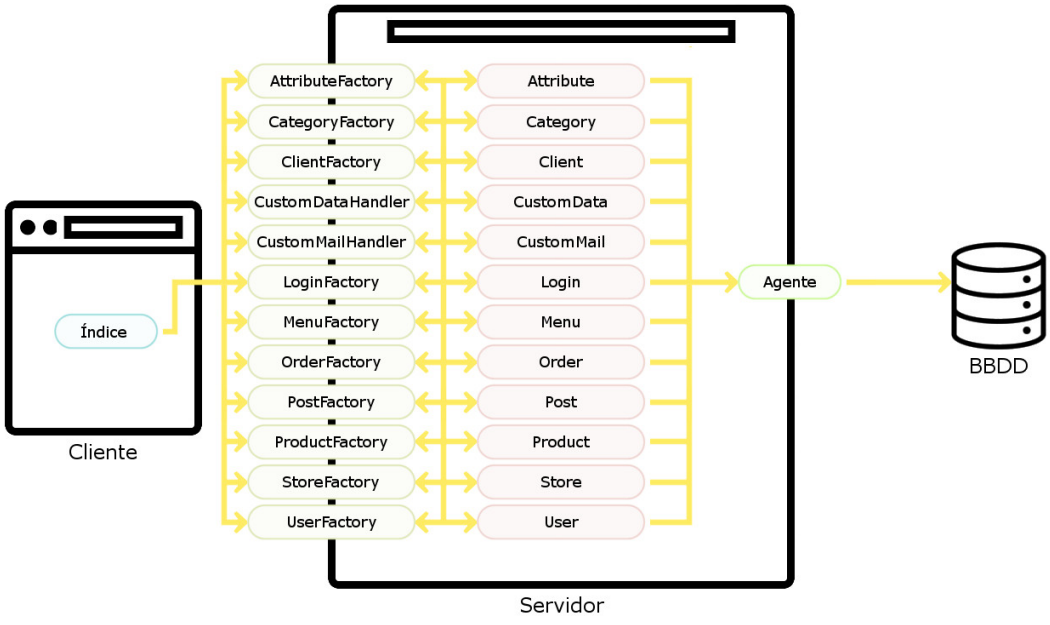


Figura 5.9: Clases que componen el dominio de Lurión

Menús en el backend

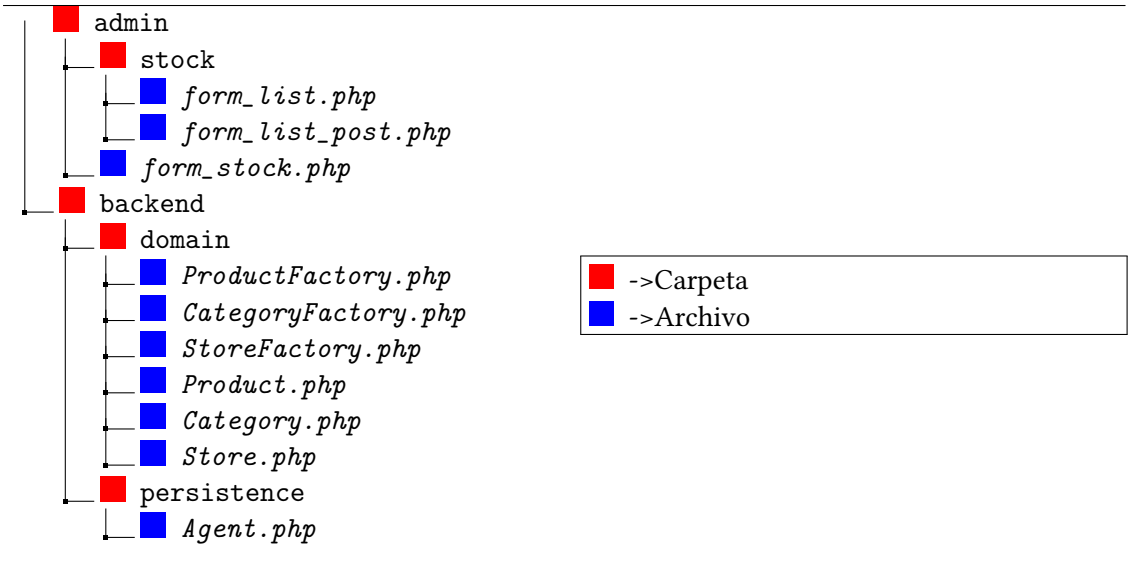
La frecuencia en la comunicación entre las Vistas y los Controladores tienen grandes diferencias dependiendo de si hablamos de las Vistas que componen el panel de administración o las que componen la web creada.

La web creada con el gestor no tendrá que hacer gran uso de las clases de dominio, solo deberá comunicarse con los Controladores cuando se solicite información almacenada en la base de datos (listado o información de productos, lista de entradas, información de una tienda…), sin embargo, en el panel de administración, cada página estará directamente relacionada con al menos una de las clases de dominio. Esto se debe a que sus funciones son el añadir, modificar o eliminar todo tipo de información almacenada, tanto en bases de datos como en archivos de configuración.

Para explicar más a fondo el proceso de comunicación entre la Vista y los Modelos utilizando los Controladores, nos centremos en uno de los menús del backend (todos siguen la misma filosofía), además se aprovechará para explicar cómo se controla el acceso a cada uno de ellos en función del rol de usuario asignado.

En este ejemplo hablaremos del listado de productos (Existencias) puesto que es uno de los que utilizarán más clases de dominio (Productos, Categorías, Atributos…) y a su vez es sencillo. Todo menú en el panel de administración deberá tener una serie de archivos que harán referencia a su aspecto y funcionalidad y todos ellos seguirán la misma estructura. A continuación, para explicar el proceso de uso de las clases de dominio, se mostrará un ejemplo con una de las funcionalidades de una opción del submenú nombrado. Véase la tabla 5.20 para ver los archivos que necesitará el menú a estudiar.

Tabla 5.20: Archivos utilizados por el menú de existencias



Como puede verse, todo menú (en nuestro caso ”existencias”) estará compuesto por un archivo que se cargará en primer lugar por el índice del panel de administración. Este archivo preparará las clases de dominio (los Controladores) que se necesitarán y analizará si el rol del usuario administrador que ha iniciado sesión tiene permisos sobre sus distintos submenús (evitando la carga de todos aquellos sobre los que no tenga permisos). En el caso que nos ocupa solo existirá un submenú (lista de productos) aunque lo común es tener varios (como por ejemplo en la sección de Páginas estáticas, cuyas opciones son añadir, modificar y eliminar). Véase la figura A.14 para tener referencia de lo que se hablará.

Listado 5.13: Inicio del menú de existencias, form_stock.php

```
<?php
  #Se añaden las distintas clases Controladoras que necesitarán todos los
  #submenus.
  include_class("ProductFactory");
  include_class("CategoryFactory");
  include_class("StoreFactory");
  $products = new ProductFactory();
  $categories = new CategoryFactory();
  $stores = new StoreFactory();
  #Se recoge la sesión del usuario para conocer sus permisos.
  $user_email = $_SESSION['email'];
  #El resultado de la variable ''page'' determinará que el menú al que se accede
  #es el dedicado a las existencias.
  $entry = get_url_variable("page");
  $options = $users->check_options_access($user_email, $entry);
  $allowed_list = strpos($options, '1') !== False ? True : False;
  $display_list = "none";
  #Una vez visto sobre qué opciones se tiene acceso se comprueba qué formulario
  #deberá ser visible (aunque en este caso solo hay uno).
  if(isset($_POST['form_list']) or isset($_POST['delete_id'])) {
    $display_list = "block";
  }else{
    if ($allowed_list){ $display_list = "block"; }
  }
?>
<div class="full-row"> <div id="stock" class="admin-layer">
    <center>
      <h1><?php echo strings("Products_menu", "prod_stock");?></h1>
      <h4><?php echo strings("Products_menu", "prod_desc");?></h4>
    </center>
    #Por cada opción se creará un botón en el menú superior horizontal,
    #si se tiene acceso.
    <div class="full-row␣tab">
      <?php if ($allowed_list){ ?>
        <?php include_once "stock/form_list_post.php"; ?>
        <button class="tablinks" onclick="openTabOption(event,␣
            'list')"><?php echo strings("Products_menu",
            "prod_list"); ?></button>
      <?php } ?>
    </div>
    #Por cada opción se añadirá el formulario que corresponda a cada opción,
    #si se tiene acceso.
    #El formulario será visible si se ha considerado la primera opción a la que
    #se tiene acceso en el paso previo.
    <?php  if ($allowed_list){ ?>
    <div id="list" class="tabcontent" style="display:␣
        <?=$display_list?>">
      <?php include_once "stock/form_list.php"; ?>
    </div>
    <?php } ?>
</div> </div>
```

En el código anterior puede verse como, de los dos archivos incluidos en el directorio (***form_list.php y form_list_post.php***), solo uno es considerado una opción del menú. Esto se debe a que todas las opciones se dividirán en dos archivos, su archivo de aspecto y primera carga y su archivo de funcionalidad (POST).

Listado 5.14: Uso de factoría para listar productos, stock/form_stock.php

```
#En primer lugar se analizará el número de productos que se mostrarán por
#página (actualmente es un valor fijo) y la página en la que está el usuario
$products_number = 7;
```

```
$search_page = get_url_variable("pagination");
if (is_null($search_page) or isset($_POST['search'])){
  $search_page = 1;
}
$sql_page = $search_page - 1;
#Posteriormente se utilizará la instancia creada previamente para llamar a la
#función que devolverá los productos.
if(!isset($total_products)){
  $id_list =
      $products->get_products_id_by_pagination($products_number,
      $sql_page);
  #A su vez se obtiene el número total de productos para conocer el total de
  #páginas necesarias.
  $total_products = $products->count_products();
}
if ($total_products[0]){
  $pages = intdiv(intval($total_products[1]), $products_number) +
      1;
}else{
  $pages = 0;
}
```

Por último veremos como actuará el Controlador para comunicarse con los Modelos, cabe destacar que las peticiones AJAX seguirán el mismo proceso en el que se instanciarán los objetos Controladores.

Listado 5.15: Código de una Factoria, ProductFactory.php

```
#Se incluye el Modelo con el que el Controlador está relacionado.
include_class("Product");
class ProductFactory{
  #Solo se mostrará la función de la que se está hablando, al ser una
  #factoría se creará una instancia del Modelo en cada función y se
  #realizarán las operaciones pertinentes.
  public static function get_products_id_by_pagination( $limit,
      $page = 0, $where = "", $order = "" ){
    $product = new Product();
    return $product->get_products_id_by_pagination($limit, $page,
        $where, $order);
} }
```

La diferencia entre factoría y manejador será, como se ha comentado previamente, que la instancia del modelo se realizará en cada función o será una variable de clase que mantendrá su estado.

5.5 DESARROLLO DE LA TIENDA SOLIDARIA USANDO LURION

El primer producto que se empezó a realizar con este gestor de contenido fue la tienda solidaria destinada a la donación de productos y la compra de los mismos por miembros de la UCLM. Esta iniciativa se inició como una forma de mostrar a la sociedad que cualquier tipo de desarrollo informático o tecnológico puede suponer un beneficio social.

Puesto que en la creación del propio gestor se incluyen los distintos requisitos que esta solución de comercio electrónico tendrá, su desarrollo se basa en la creación de formularios y asignación de colores e información corporativa de la Universidad de Castilla-La Mancha. También existen ciertos elementos distintivos, que puede que sean exclusivos de esta solución, como puede ser el proceso de pago que tendrán que realizar los usuarios de la universidad (se deberá optar por una solución que evite las entregas de dinero en mano y que sea lo más clara posible), actualmente dos de las opciones viables serán o el pago a través de la plataforma PayPal o a través de una pasarela bancaria. Por otro lado se pretende, al menos al principio, controlar qué tipos de personas accederán a la web, para tener

un control más objetivo, por ello para el inicio de sesión de los clientes se pretende usar la misma API utilizada por el campus virtual de la universidad. Como puede verse aun no se han definido los requisitos necesarios para estos casos, pero si se puede mostrar el estado actual de la tienda.

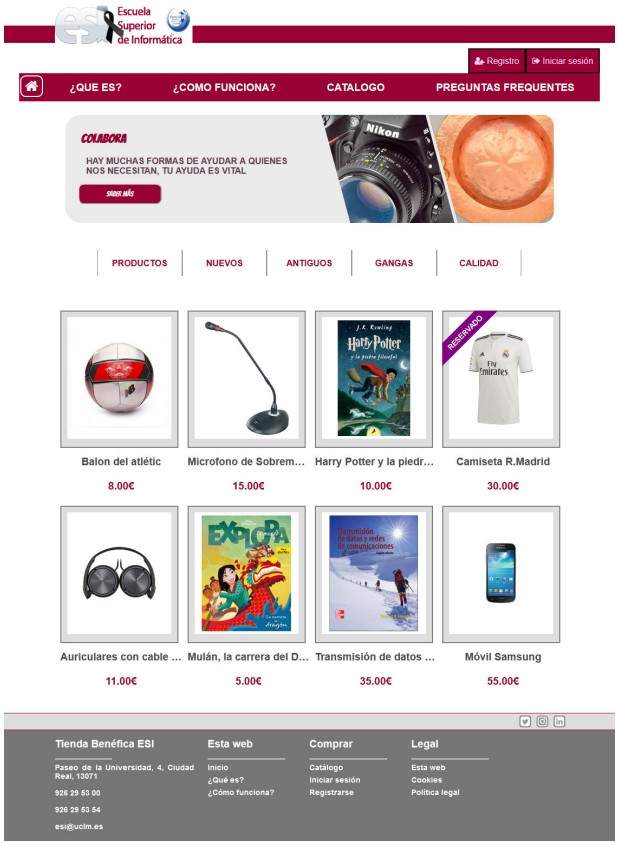


Figura 5.10: Tienda benéfica UCLM, Página de inicio

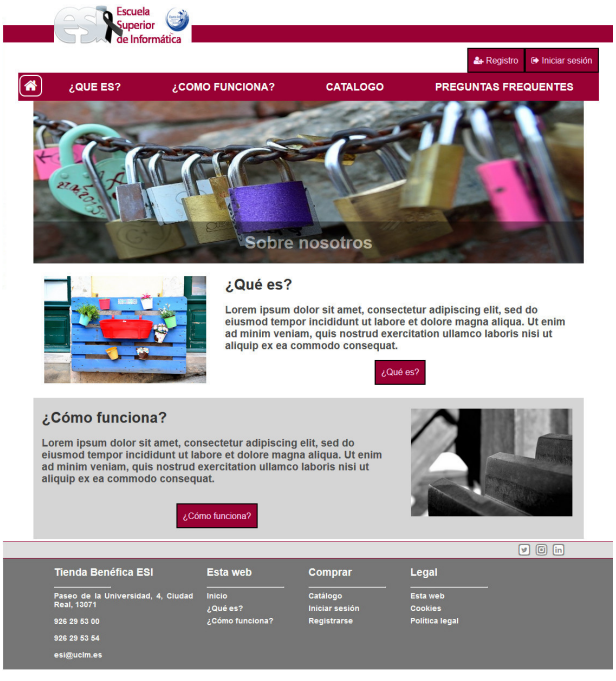


Figura 5.11: Tienda benéfica UCLM, Páginas estáticas

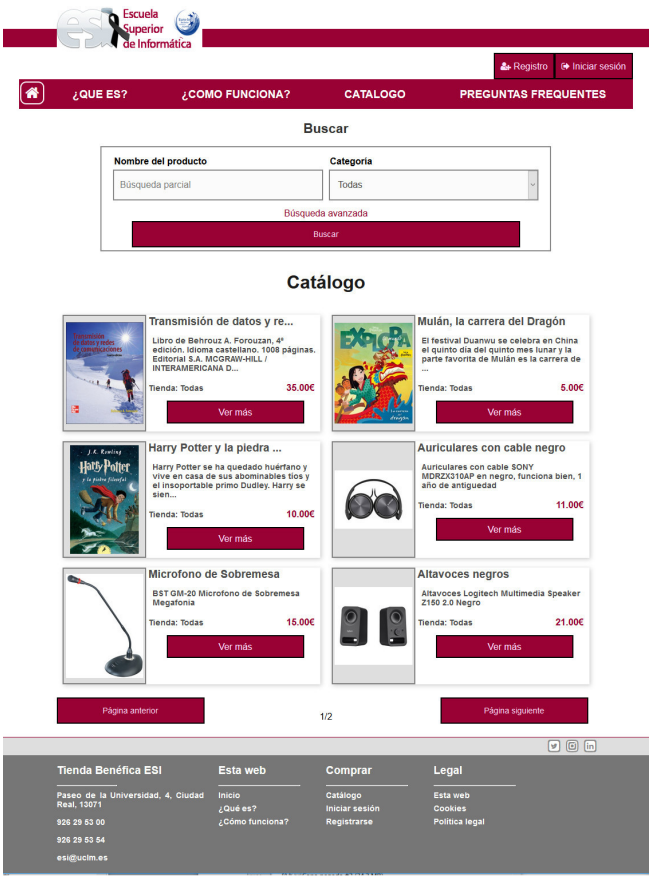


Figura 5.12: Tienda benéfica UCLM, Catálogo

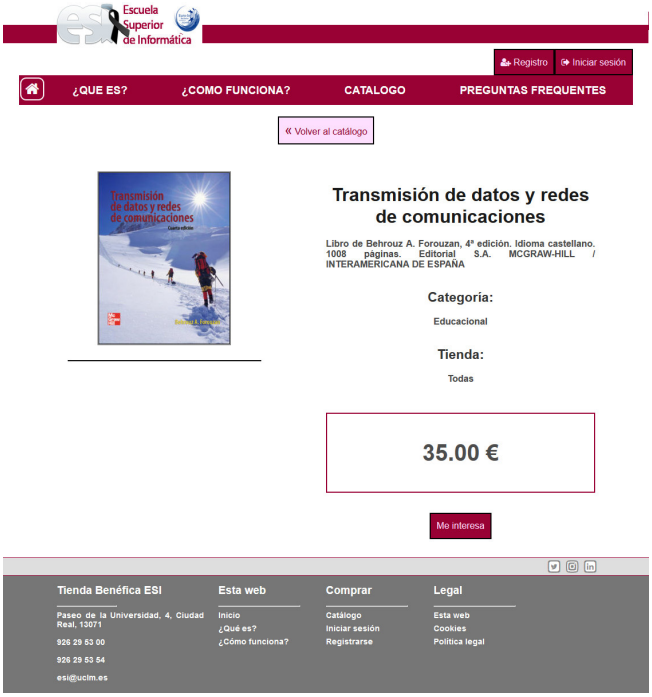


Figura 5.13: Tienda benéfica UCLM, Producto

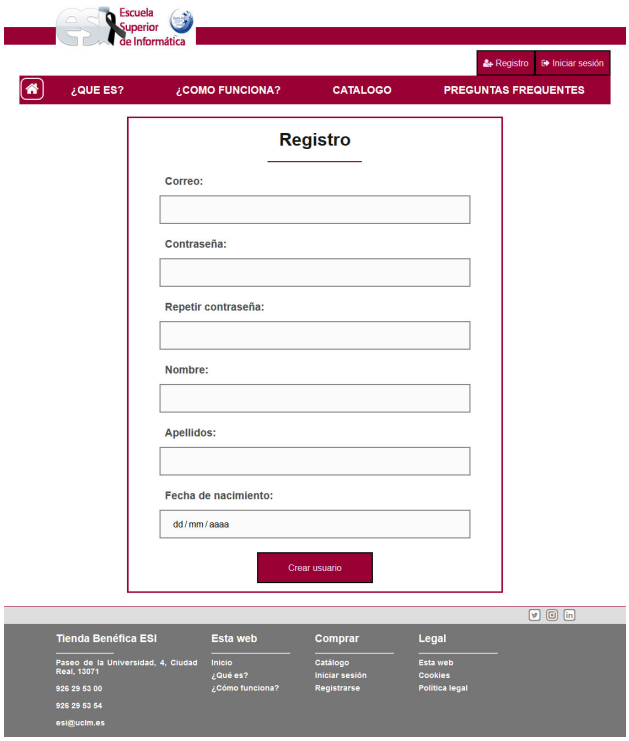


Figura 5.14: Tienda benéfica UCLM, Registro

Figura 5.15: Tienda benéfica UCLM, Cabecera con la sesión iniciada

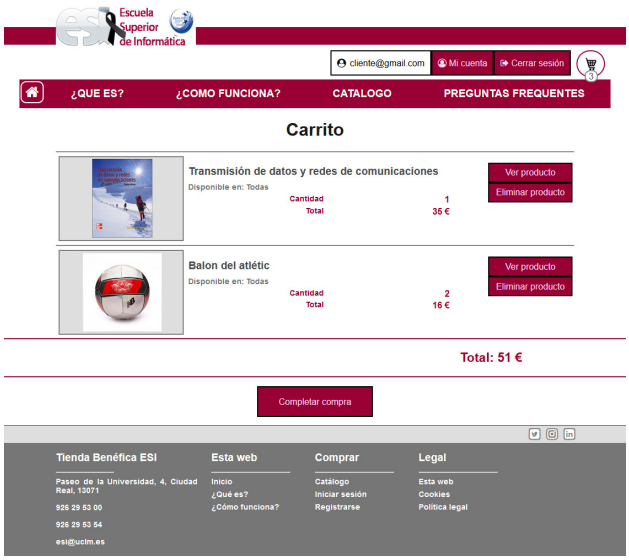


Figura 5.16: Tienda benéfica UCLM, Carrito

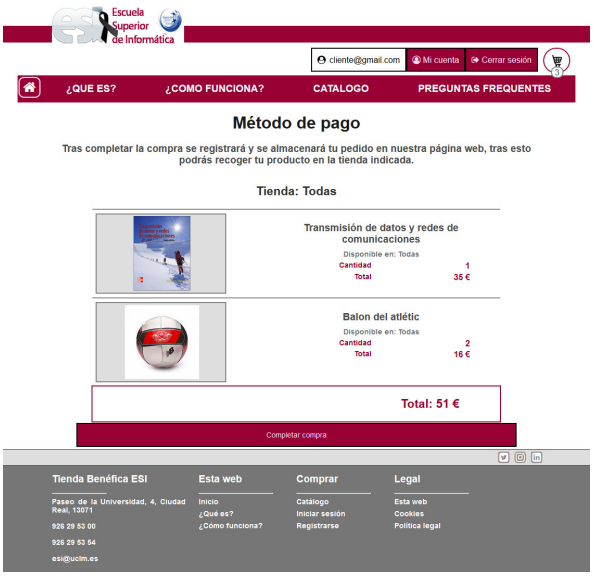


Figura 5.17: Tienda benéfica UCLM, Método de pago

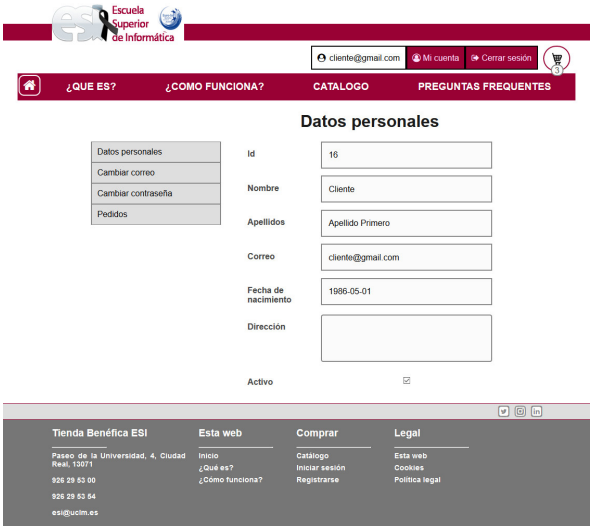


Figura 5.18: Tienda benéfica UCLM, Perfil de usuario, datos personales

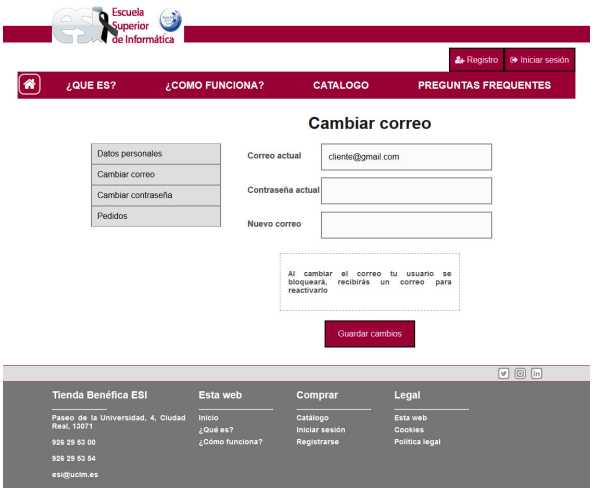


Figura 5.19: Tienda benéfica UCLM, Perfil de usuario, cambiar correo

Figura 5.20: Tienda benéfica UCLM, Perfil de usuario, cambiar contraseña

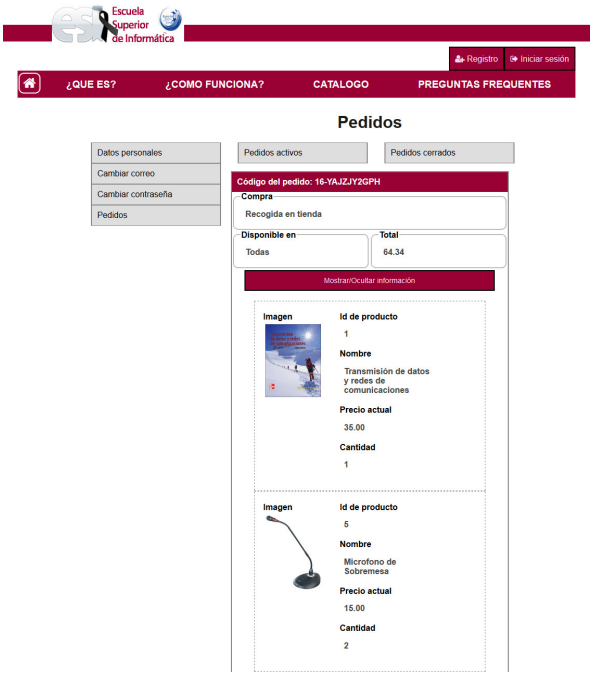


Figura 5.21: Tienda benéfica UCLM, Perfil de usuario, pedidos

Se debe hacer ver que estos formularios posiblemente cambiarán en su versión final, ya que actualmente está desarrollada la página web siguiendo las directrices que se nos han dado. Se realizarán las compras por parte de los clientes a través de la web, habrán ciertos puntos de recogida (representados por las tiendas del gestor) y tanto los donantes como los que compren dicho producto deberán ir a dicho punto de recogida con el producto a donar o con el código de compra obtenido a través de la página web (que también se enviará al cliente por correo).

5.6 DESARROLLO DE LA WEB DE CATÁLOGO USANDO LURION

Este desarrollo permitirá ver un tipo distinto de página web empresarial que podrá crear Lurión. El proyecto en sí se basa en el desarrollo de una página web para una profesora de clases particulares para primaria e infantil. En esta página se pretende crear un catálogo en el que dicha profesora muestre sus cualidades y la formación que puede ofrecer a los alumnos, a su vez habrá una nueva forma de ver a los productos.

En dicha página se pretende no solo crear una página estática, sino también se pretende tener una aplicación donde los alumnos puedan iniciar sesión y se permita, tanto a los alumnos como a sus padres, descargarse apuntes e informes sobre el desarrollo de dichos alumnos. Para ello se utilizará el concepto de producto virtual gratuito (representando así los apuntes).

Los productos como tal serán cada uno de los horarios disponibles (número de horas a la semana) y existirá una diferencia entre productos físicos (los horarios) y los productos digitales (apuntes), creando un formulario de catálogo que solo muestre los segundos.

Este subproyecto ya ha recogido los distintos requisitos y ya se ha creado la imagen corporativa, sin embargo, gran parte del diseño tan solo está prototipada.

Figura 5.22: Anna Samper, Cabecera y colores corporativos

Figura 5.23: Anna Samper, Página principal

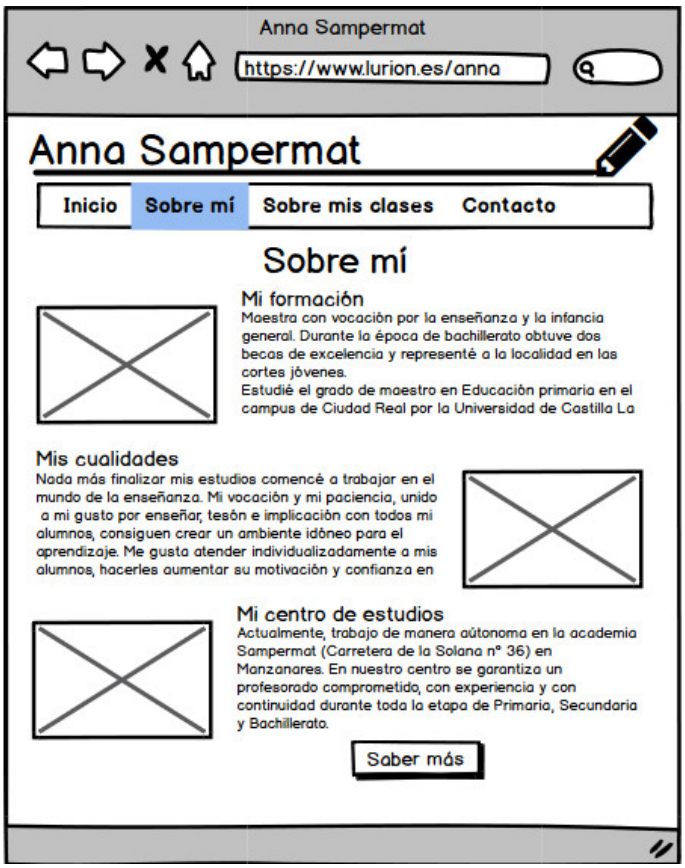


Figura 5.24: Anna Samper, Página estática, sobre mi

Figura 5.25: Anna Samper, Página estática, sobre mis clases

Figura 5.26: Anna Samper, Contacto

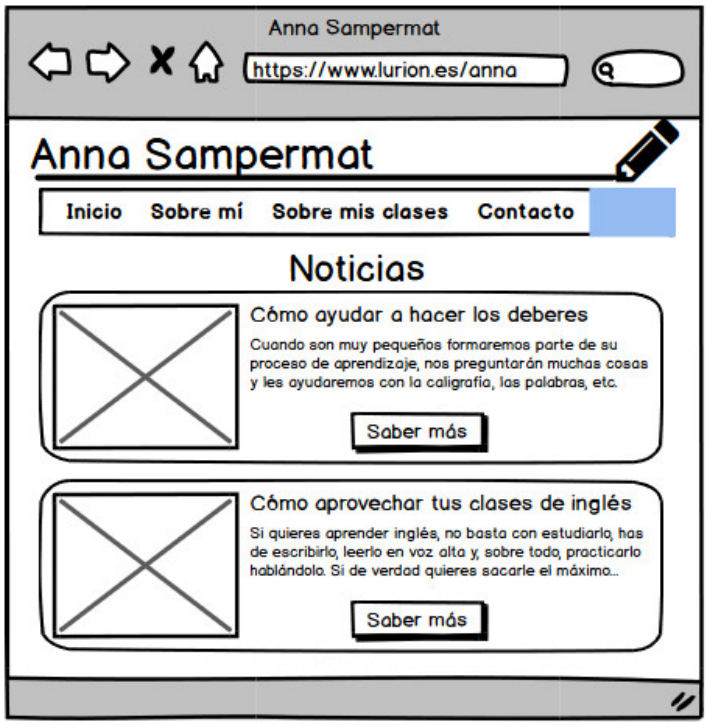
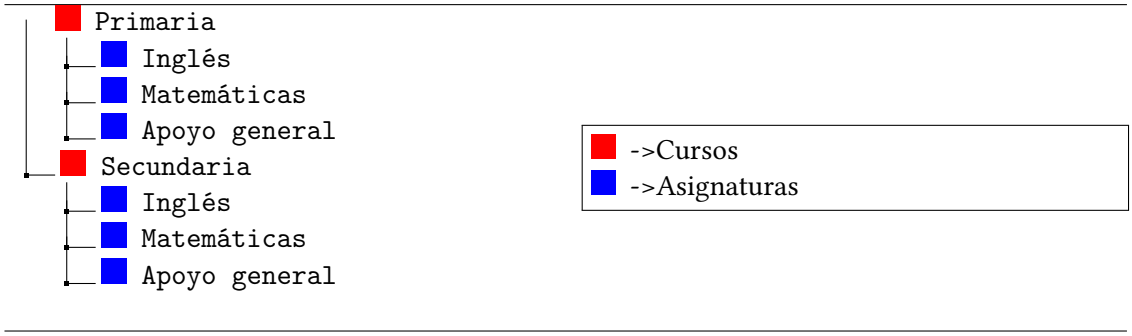


Figura 5.27: Anna Samper, Noticias

Como se ha nombrado previamente, el catálogo en este proyecto no tendrá un uso habitual, tendrá una funcionalidad y disposición distinta y se pretende crear una estructura como la mostrada a continuación:

Tabla 5.21: Categorías de la academia



5.7 DESARROLLO DE LA WEB PARA LA PROMOCIÓN DE LURIÓN

Como el resto de gestores de contenido, es recomendable que Lurión tenga una página web promocional en la que hablar de sus virtudes, su funcionamiento y las novedades sobre sus actualizaciones o mejoras. Con este objetivo se ha creado la página web de Lurión (***www.lurion.es***), usando lógicamente el propio gestor para ello. Aquí puede verse la creación de un tipo distinto de página web empresarial en la que no existirán usuarios ni se venderá ningún producto o servicio, sino que será utilizada como catálogo e información basada en páginas estáticas.

Esta web utilizará todos los formularios por defecto con el objetivo de intentar crear una imagen visual de marca (como Prestashop que tiene una plantilla por defecto bastante reconocible), y se hará ver lo que se puede crear gracias a esta herramienta en una sección de porfolio donde se almacenen ejemplos de páginas webs, atractivas visualmente hablando, creadas con Lurión.

Puesto que actualmente su código no es descargable ni tiene ningún tipo de comunidad de creadores la página web se utilizará principalmente como tutorial de uso del gestor, aunque existen múltiples proyectos de mejora como el crear un panel de administración sin funcionalidades para que cualquier usuario pueda navegar y familiarizarse, crear productos/servicios virtuales en los que se puedan descargar plantillas (de pago o gratuitas), crear un sistema de catálogo cuyas categorías sean los distintos aspectos de la página que puedan personalizarse (cabecera, página de producto…)

Sin embargo, viendo el estado de desarrollo en que se encuentra el gestor, esta página estará pensada principalmente para navegar por ella y hacerse una ligera idea de su potencial.

Por último cabe destacar que, aunque actualmente estén incluidos los formularios para ello y sean plenamente funcionales, existen ciertos aspectos de la página que están actualmente inactivos, como podría ser el inicio de sesión y registro en la página web o el envío de mensajes a través del formulario de contacto, así pues, a grandes rasgos, la página será la siguiente:

Figura 5.28: Página web de Lurión, Página principal

Figura 5.29: Página web de Lurión, Lista de entradas

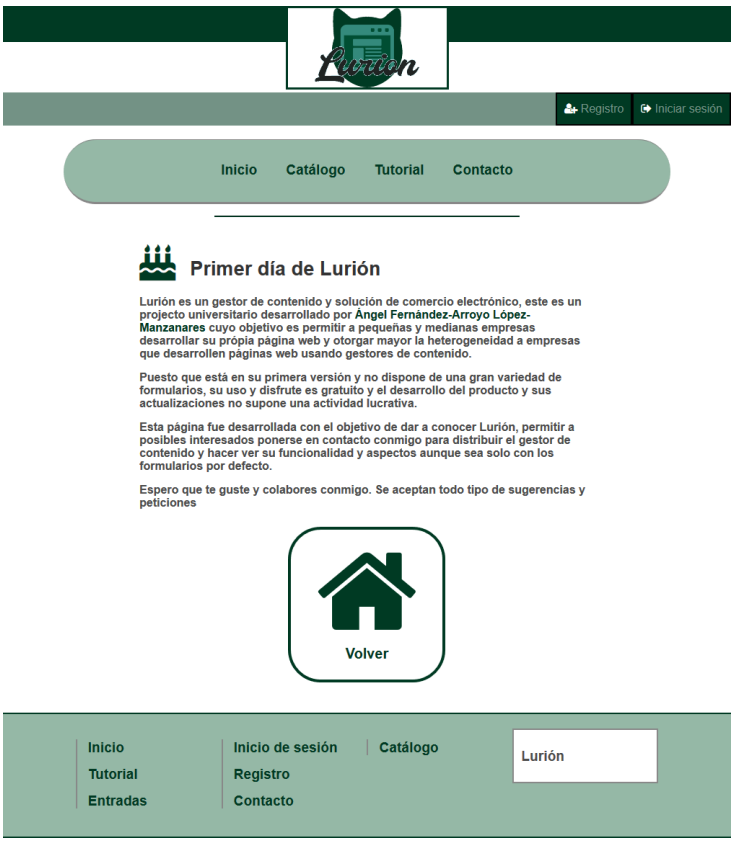


Figura 5.30: Página web de Lurión, Primera entrada

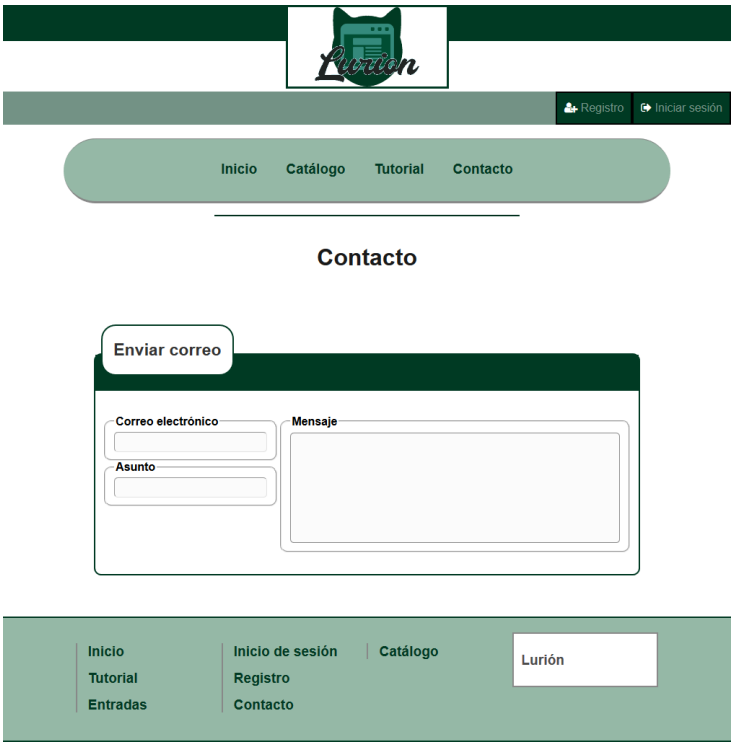


Figura 5.31: Página web de Lurión, Página de contacto

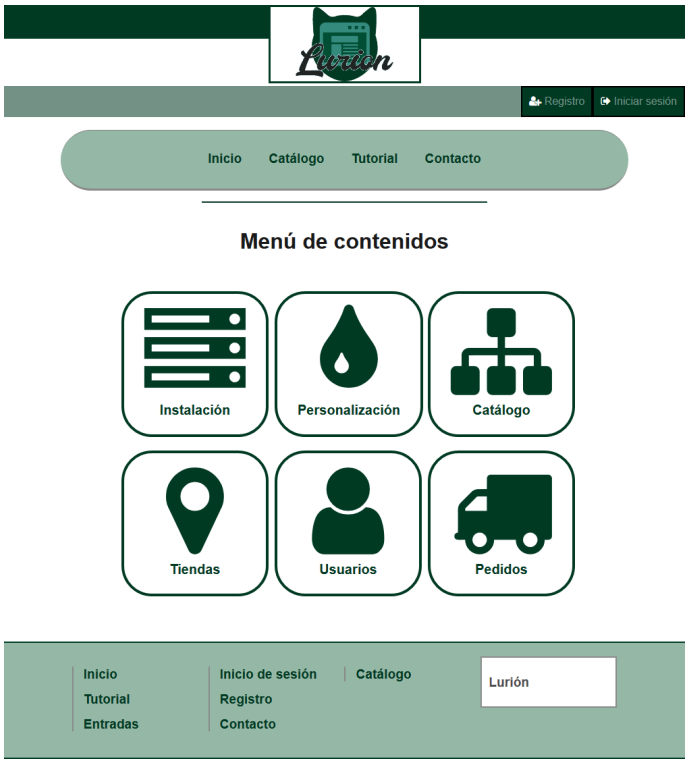


Figura 5.32: Página web de Lurión, Página estática, índice del tutorial

Figura 5.33: Página web de Lurión, Página estática, tutorial sobre tiendas

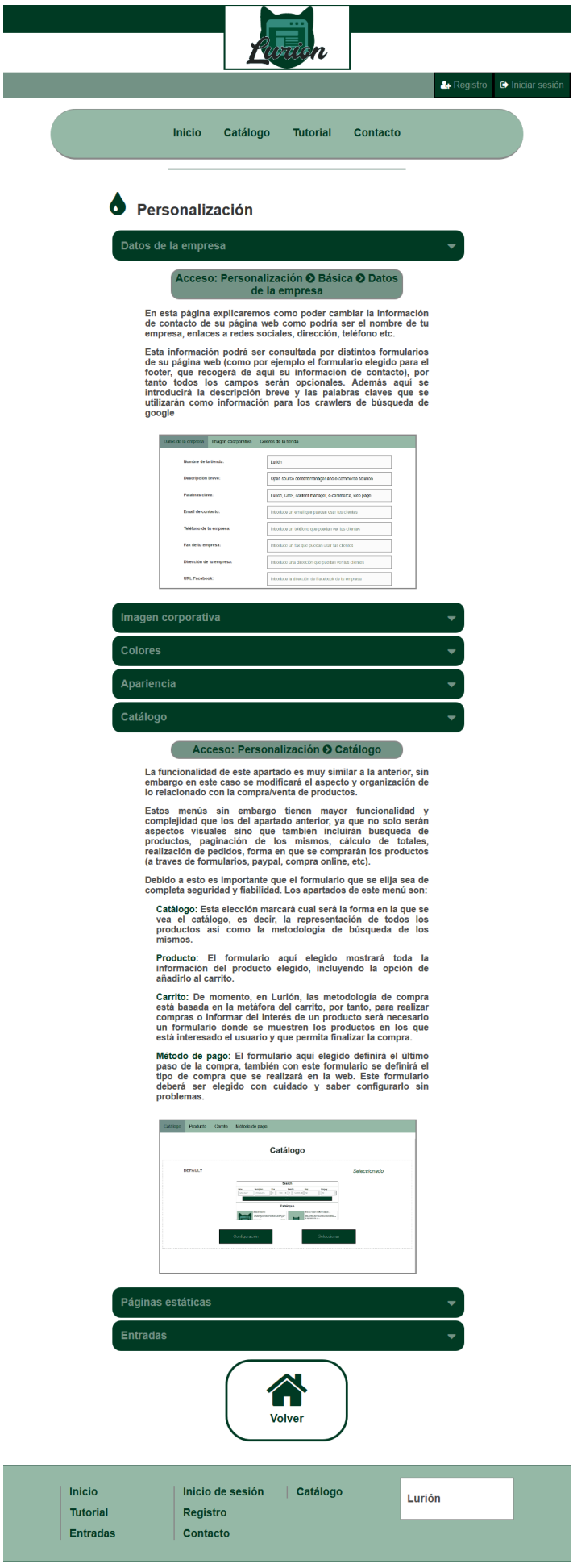


Figura 5.34: Página web de Lurión, Página estática, tutorial sobre personalización

Figura 5.35: Página web de Lurión, Página estática, tutorial sobre usuarios

CAPÍTULO 6

CONCLUSIONES

6.1 OBJETIVOS ALCANZADOS

El proyecto llevado a cabo ha sido un trabajo muy ambicioso, y quizás, por la excesiva complejidad, algo desproporcionado para ser un Trabajo Fin de Grado. Se ha diseñado y desarrollado una herramienta que permite crear una solución de comercio electrónico o simplemente tener una web para estar presente en Internet, destinada a aquellas microempresas que deseen atreverse con la transformación digital en sus procesos de venta o simplemente en el marketing. El sistema desarrollado es un gestor de contenido, como puede ser Prestashop, que pretende facilitar la creación de cualquier tipo de página web a cualquier persona en cualquier lugar, sin importar sus conocimientos, su lenguaje, los contenidos a incluir… y todo esto sin utilizar ni un solo recurso externo.

6.1.1 Objetivo general

[OG1.] Diseñar y desarrollar un Sistema Gestor de Contenidos que permita la creación de páginas web empresariales de cualquier tipo, pero con capacidad para la creación de tiendas online.

Se ha cumplido este objetivo ya que se ha diseñado y desarrollado un Sistema Gestor de Contenidos, que se ha nombrado Lurión, que permite la creación de páginas webs y soluciones de comercio electrónico. Lurión tiene las siguientes características:

- Personalizable en cualquier aspecto visual, i.e. colores, uso de logotipos para transmitir la imagen corporativa, imágenes propias para la web.
- Personalizable en aspectos funcionales y de estilo en función de las necesidades del cliente, gracias a su capacidad para añadir código propio CSS y JS.
- Particularizable, tanto en la información sobre la empresa, puesto que permite añadir todo tipo de información sobre ella, como en metadatos sobre la página, puesto que crea sesiones personalizadas para cada usuario, administrador o cliente, que quiera acceder a la página.
- Ilimitada, puesto que permite añadir tantas páginas estáticas y entradas como se requiera, pudiendo estar formadas, tanto por contenido estático, como dinámico.
- Adaptable a la actividad de la empresa, permite la creación de una página web o un comercio electrónico en el que se puede definir un catálogo que englobe el tipo de productos/servicios que la empresa pueda tener y a su vez permite añadir productos relacionados con dicho catálogo.
- Multilenguaje, permite cambiar el idioma de todo el gestor de forma sencilla, teniendo solo que modificar un archivo.

- Fácilmente instalable al crear todo un proceso de instalación en el que se instala la base de datos, se crean usuarios administradores, se define el lenguaje que se requiera.
- Extensible, pueden añadirse tantas funcionalidades como se desee por programadores con conocimiento más avanzado.

6.1.2 Objetivos específicos

[OS1.] Crear una web para la Universidad de Castilla-La Mancha que permita la venta online, dentro de la comunidad universitaria, de productos de segunda mano que estén incluidos en un catálogo.

Se ha cumplido este objetivo al aplicar Lurión para crear un portal que permite la compra de productos solidarios en la UCLM, a miembros de la comunidad universitaria. Este, aunque esta realizado prácticamente por completo, aún no está en explotación. Se espera que lo esté en los próximos meses. El portal tiene las siguientes características:

- Visual, manteniendo la imagen corporativa y creando formularios específicos para los distintos apartados de la página web creada.
- Funcional, adaptando la herramienta a todas las necesidades, tanto de inicio de sesión, permitiendo la entrada solo a los miembros de la UCLM, como de posibilidades de compra/venta, con gestión de tiendas o lugares de recogida y definición de administradores de esas tiendas.
- Usable, la tienda online es sencilla, facilita en todo momento la tarea de encontrar información sobre los productos o servicios ofertados, así como la tarea de la compra de dichos productos o servicios. Se busca una navegación sencilla y se implementan métodos de búsqueda para productos.

[OS2.] Crear una web empresarial para una microempresa que ofrezca servicios.

Se ha cumplido este objetivo puesto que también se ha aplicado Lurión para crear una página web para una emprendedora dentro del campo de la enseñanza que le permita publicitarse y poner a disposición de sus alumnos los apuntes o materiales propios de la actividad docente. Este aún no está en explotación, se espera que lo esté en los próximos meses. La web tiene las siguientes características:

- Visual, utilizando los colores corporativos de la academia y acompañando de formularios llamativos.
- Funcional, ya que permite la gestión de un centro de formación, con todas las funcionalidades que este requiere.
- Personalizado, permitiendo actividades propias de un centro de formación, los alumnos podrán descargar contenido diseñado para ellos y resúmenes de su progresol el profesor podrá subir documentos para los alumnos.

[OS3.] Crear una web para dar soporte a la idea emprendedora, permitiendo la comercialización del gestor de contenido obtenido como resultado del OG1, para ello se empleará el propio gestor.

Se ha cumplido este objetivo al crear, empleando Lurión, la web que materializa, en esta fase inicial, la idea emprendedora que hay tras este Trabajo Fin de Grado. La web tiene las siguientes

características:

- Visual, intentando crear una relación directa entre el gestor de contenido y los tonos verdes (según la psicología del color, esta tonalidad tiene un efecto armónico y se asocia con el crecimiento y la buena suerte).
- Divulgativo, pretende informar sobre el gestor, con un portfolio de webs realizadas con el gestor, la inclusión de un manual de usuario para el uso de Lurión.
- Actualizado, utilizando un blog propio y manejable a través del propio sistema, se informará del estado del gestor y las novedades relacionadas con él.

6.1.3 Objetivos formativos principales

[TI3.] Adquirir capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.

Se ha cumplido este objetivo ya que en todo momento se ha mantenido un enfoque de desarrollo centrado en el usuario. Durante las primeras etapas del desarrollo se ha utilizado un marco de trabajo basado en SCRUM e historias de usuario para identificar en todo momento qué necesita el cliente y adaptarlo así en cualquier momento, incluso en etapas posteriores. Durante la implementación de la herramienta también se ha analizado cómo atraer a potenciales clientes o visitantes, las necesidades que tendrá el administrador de la página y cómo facilitar su trabajo, incluso para futuros desarrolladores de funcionalidades en el propio gestor, sobre los que se ha creado una API de funciones que les permita trabajar de una forma mucho más sencilla.

[TI6.] Adquirir capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.

Se ha cumplido este objetivo, puesto que se ha desarrollado una aplicación del tipo cliente-servidor en la que la transmisión de información, así como entrega de formularios es a través del protocolo HTTP. A su vez, se permite la visualización de dicha aplicación en distintos tipos de dispositivos y resoluciones, adecuando por tanto la parte visual, así como los contenidos multimedia que contenga. Destacar también que como uno de los resultados del uso del gestor de contenido, se ha creado un portal de comercio electrónico.

[TI5.] Adquirir capacidad para seleccionar, desplegar, integrar y gestionar sistemas de información que satisfagan las necesidades de la organización, con los criterios de coste y calidad identificados.

Se ha cumplido este objetivo, ya que a lo largo del trabajo, tanto práctica como teóricamente, se han analizado las necesidades que cualquier empresa pueda tener a la hora de desarrollar un sistema web y el producto desarrollado se puede adaptar a cada una de ellas. A su vez, durante su desarrollo, se ha escogido cuidadosamente qué patrón y metodología, incluso orden de código será más eficiente y tendrá menos costes de rendimiento, velocidad de carga, limpieza de código y puntuación por los buscadores de Google.

[TI7.] Adquirir capacidad para comprender, aplicar y gestionar la garantía y seguridad de los sistemas informáticos.

Se ha cumplido este objetivo. Aunque un gestor de contenido debe dejar ciertos ámbitos de seguridad en mano del administrador de la página creada y los creadores de las plantillas seleccionadas, el gestor de contenido se encarga de la seguridad de los datos personales que deban ser utilizados o guardados (encriptación de contraseñas, evitar almacenar datos bancarios, prohibir el acceso a los datos almacenados del servidor, evitar que se pueda acceder desde el administrador de código a datos delicados...) y esto se hace en todo momento, tanto por las funciones de encriptación otorgadas por el lenguaje de programación como por parte de métodos de intercambio de caracteres.

6.1.4 Objetivos formativos específicos

[FE1.] Aprender en profundidad un lenguaje de programación específico para servidores web (PHP).

Se cumple este objetivo. El lenguaje de programación utilizado en el servidor ha sido lenguaje PHP debido a que es uno de los lenguajes de servidor más conocidos (facilitando así el trabajo de los creadores de formularios). Pueden demostrarse los conocimientos adquiridos tanto por las distintas funcionalidades y clases de dominio que se han desarrollado como por el total de 28.718 líneas de código en 392 archivos y clases, habiendo creado un total de 884 métodos.

[FE2.] Aprender en profundidad un sistema de gestión de bases de datos relacional (MySQL) y diseñar e implementar una base de datos relacional.

Se ha cumplido este objetivo, ya que, para el correcto funcionamiento del gestor de contenido y para almacenar distintos datos utilizados por el administrador de la tienda se ha creado una base de datos relacional que puede verse en la figura 5.8

[FE3.] Aprender lenguajes y tecnologías específicas para clientes web, siendo estos el lenguaje de hojas de estilo en cascada (CSS3), el lenguaje de marcas de hipertexto (HTML5) y el lenguaje de programación interpretado en el cliente (JS).

Se ha cumplido este objetivo al crear todos los formularios y funcionalidades que se ejecutarán en el cliente con estos lenguajes de programación. Este código ocupa en la herramienta 2.432 lineas de código a lo largo de 20 archivos, existiendo 193 funciones desarrolladas con javascript y que además pueden utilizar Jquery y AJAX.

[FE4.] Aprender lenguajes de marcado de propósito general (XML).

Se ha cumplido este objetivo puesto que la información dependiente de la solución, ya sea por el administrador, el lenguaje, etc. se almacenará en distintos grupos y etiquetas en archivos XML, a su vez la herramienta controlará toda la gestión de dicha información (añadir, modificar, eliminar). Más allá del código que controla dichos archivos, puede verse que la información que estos archivos almacenan se guardan en un total de 2697 etiquetas a lo largo de 108 archivos.

[FE5.] Aprender técnicas de posicionamiento de las páginas web en los buscadores (.htaccess, robots.txt, pagespeed, google analytics).

Se ha cumplido este objetivo, ya que, tras finalizar el proceso de creación de la herramienta se inicio una etapa de adaptación de la misma a navegadores y buscadores web para aumentar su eficiencia, velocidad de carga, método de inclusión de recursos de la herramienta, etc. Se hace uso del archivo .htaccess para bloquear el acceso a archivos de configuración que no deban ser vistos por parte del cliente, redirecciones de URL erróneas hacia una página 404 y establecer el tiempo en que los recursos se almacenarán en caché. A su vez el archivo robots.txt informa que no se deben recorrer ciertas direcciones de la página (traducciones, clases de dominio, etc.).

Gracias a este proceso actualmente, la página web de Lurión, creada con Lurión, página actualmente en funcionamiento y con formularios instalados, tiene una calificación de 99/100 en PageSpeed en dispositivos móviles y 100/100 en ordenador. Véanse las figuras 6.1 y 6.2.

Figura 6.1: Puntuación en PageSpeed de Lurión en dispositivos de alta resolución

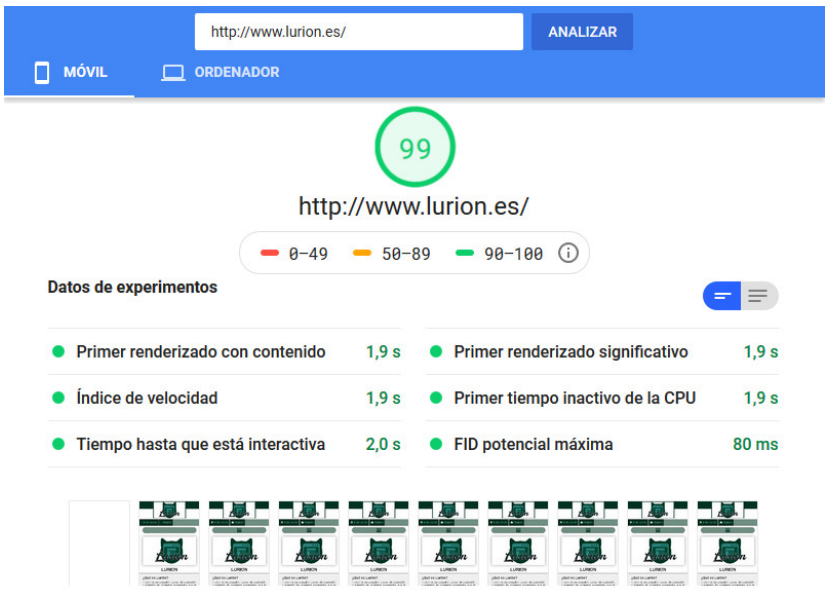


Figura 6.2: Puntuación en PageSpeed de Lurión en dispositivos móviles

6.2 PROYECTO FUTURO

Este proyecto que se ha comenzado con este Trabajo Fin de Grado no acabará aquí, es un proyecto vivo, un proyecto de emprendimiento con idea de futuro. Lurión sería el producto, se vendería como una herramienta para aquellas microempresas que no puedan realizar una transformación digital. Hay varias ideas de negocio, una de ellas es la comercialización de webs realizadas con Lurión o incluso permitir su uso a cambio de porcentajes sobre las ventas realizadas en la solución desarrollada.

Como negocio, en el futuro inmediato, se pretende potenciar su uso poniendo en marcha diferentes acciones.

1. Se trabajará en la visibilidad de la web del proyecto, hay que conseguir que el gestor sea conocido y considerado a nivel nacional, y que aquellas personas interesadas lleguen a él a través de los buscadores.
2. Se presentará el proyecto a los premios Vodafone a la innovación, con el objetivo de darle visibilidad, aunque sea mínimamente.
3. Se pondrá en marcha la Tienda Solidaria UCLM, creemos que esto puede ser una buena operación de marketing a nivel regional.
4. Se utilizará el gestor para crear páginas webs o soluciones de comercio electrónico para microempresas, el objetivo es tener un portfolio de soluciones que atraigan a otros potenciales clientes.

Estas son tan solo algunas de las acciones a llevar a cabo, a nivel de promoción. Sin embargo, y como producto vivo, el gestor deberá seguir evolucionando y perfeccionándose. Algunos de los cambios que se pretenden hacer ya han sido planteados e incluso están las funcionalidades y métodos necesarios ya creados (aunque no se ha creado la interfaz visual para llevarlos a cabo), algunos de estos cambios son:

1. **Permitir que el gestor sea completamente multilenguaje**
   - **Ampliar el sistema de traducciones a las páginas estáticas y los menús de configuración de formularios**: Esto será un paso muy sencillo, ya que la única diferencia sería añadir un elemento HTML de tipo *select* que recoja el lenguaje en el que se está introduciendo la información.

     En el primer caso solo habría que añadir el lenguaje en que se ha creado (en, es…) en el nombre del archivo que representará la página estática, de esta forma al recoger la página estática *"quienes_somos.html"* en idioma español tan solo habría que buscar el archivo *"quienes_somos_es.html"* y lo mismo con cualquier otro idioma.

     En el segundo caso será incluso más sencillo puesto que tan solo será necesario seleccionar el idioma a través de un *select* añadido tanto a *inputs* como a *textareas* y añadir los textos introducidos en el sistema de traducciones del propio formulario (<**identificador_input_es**>Texto en español del identificador</**identificador_input_es**>).
   - **Ampliar el sistema de traducciones a los productos y categorías**: Los textos relacionados con los atributos, los productos y las categorías son almacenados en la base de datos junto a un valor de idioma, es decir, los textos de los productos que se almacenan en *front_products_lang* estan acompañados por un campo *lang* que define el lenguaje de los textos, tan solo es necesario crear un formulario sencillo y entendible donde poner tantas traducciones como archivos de lenguaje tenga instalados el gestor (se podría añadir una

nueva opción en dichos submenús para traducir todos los textos del producto/categoría llamado idiomas, permitir añadir en el panel de administración distintos formularios que sirvan para introducir los distintos idiomas…)

2. **Añadir funcionalidades relacionados con la compra/venta**

   - **Crear un control y seguimiento de transportistas**: Esta mejora no se prevé que sea a corto plazo puesto que para permitir esta funcionalidad es necesario, entre otras cosas, que los transportistas conozcan y muestren interés por el hecho de añadir su proceso de seguimiento (a través de una API o de peticiones a través de un web service generalmente) en el gestor.
   - **Incluir una metodología genérica para introducir pasarelas de pago**: Este tipo de servicio es muy heterogéneo, entre otros motivos, por seguridad, a su vez se necesita un permiso previo para incluir dicha pasarela, por lo que esto será una de las modificaciones más complicadas.
   - **Incluir formularios de pago a traves de PayPal**: Estos formularios no se incluyen actualmente en el gestor, pero todo indica que esto cambiará pronto. Esto podrá realizarse a través del *Orders API* creado por el propio PayPal.

3. **Añadir recursos visuales y funcionales para los creadores de contenido**

   - **Poder asignar imágenes a las categorías**: Añadir recursos gráficos opcionales con el objetivo de hacer más sencilla la navegación y más visuales los formularios creados.
   - **Crear un método que permita mostrar las migas de pan**: Puesto que los formularios, en función de su funcionalidad, se cargarán en una zona u otra, el mostrar las migas de pan para el usuario será algo sencillo.
   - **Restringir ciertas funciones a través del método callback**: Darle libertad a los creadores es algo que permitirá crear múltiples formularios y no frenar la creatividad de los mismos, sin embargo, también puede ser algo peligroso, por ello se pretende controlar, en el método callback al que se llama antes de cargar los formularios, que las funciones a las que llame el creador estén dentro de las funciones permitidas.

Por supuesto, en un proyecto de este tipo, el único límite de mejora se encuentra en la creatividad del equipo de desarrollo, es por eso que se podría mostrar una gran cantidad de elementos que podrían perfeccionar y mejorar a Lurión, sin embargo, los nombrados hasta aquí son los cambios que se pretenden realizar en un corto/medio plazo.

6.3 CONCLUSIÓN FINAL

En este punto del desarrollo es prácticamente imposible el saber si este trabajo será útil, creará una ventaja o una oportunidad de mejora sobre la sociedad (o al menos sobre parte de ella), etc… sin embargo si puede afirmarse que el objetivo de este trabajo nunca ha sido el de completar una asignatura, sino crear un producto que de verdad sea útil y permita, sino ayudar, al menos crear una oportunidad ante esa gran parte de la sociedad que se encuentra en una gran desventaja competitiva ante grandes empresas que actúan y operan a través de Internet, creando a su vez una oportunidad de negocio.

Mirando más allá del análisis práctico o de la utilidad de este proyecto puedo afirmar que este trabajo ha supuesto, tanto un sacrificio (a modo de apuesta puesto que podría llegar a aportarme grandes beneficios) como un proceso de aprendizaje como pocos otros he tenido, no solo en cuanto a conocimientos técnicos sobre las herramientas utilizadas, sino también en aprender cómo, una idea

previa y un análisis exhaustivo no puede englobar todos los apartados y pequeñas cosas que el crear un proyecto desde cero supone.

A su vez afirmar que todo lo hecho y por hacer siempre será mejorable, sin embargo hay que saber buscar un equilibrio entre los beneficios que reportarán dichas mejoras frente al esfuerzo que requiere, es por ello que se ha decidido presentar el proyecto en esta versión estable, aunque por supuesto, el desarrollo de la tienda benéfica de la UCLM, la web de catálogo y con suerte el desarrollo de otras muchas webs que decidan confiar en Lurión supondrán un proceso de mejora continua. Como dijo Winston Churchill, *" El esfuerzo constante, no la fuerza o la inteligencia, es la clave para liberar nuestro potencial"*.

ANEXO A

PROTOTIPOS DE LURIÓN

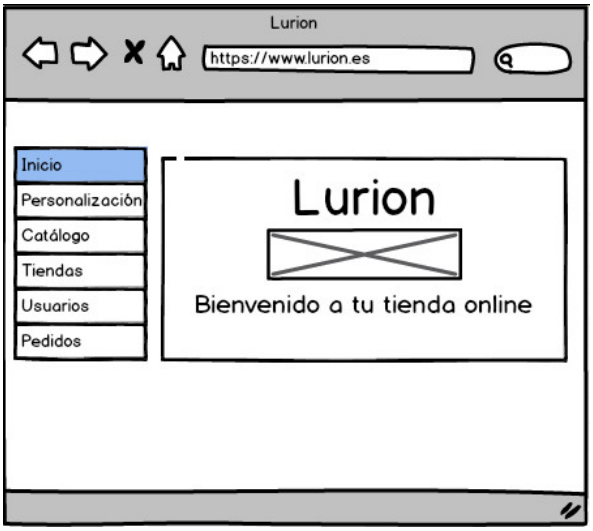


Figura A.1: Prototipo de baja calidad, Inicio

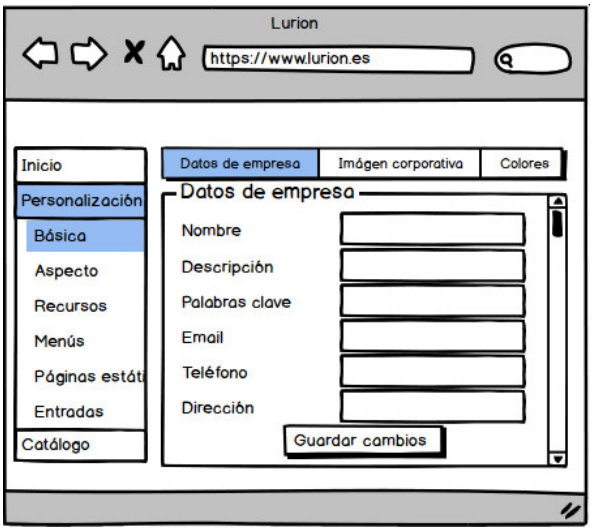


Figura A.2: Prototipo de baja calidad, Datos de empresa

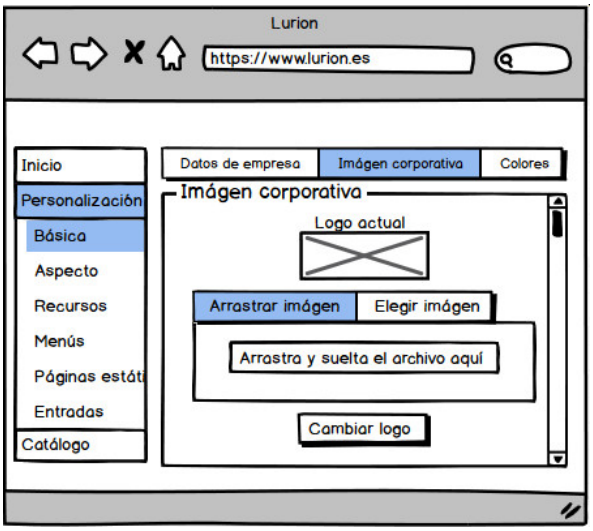


Figura A.3: Prototipo de baja calidad, Imagen corporativa

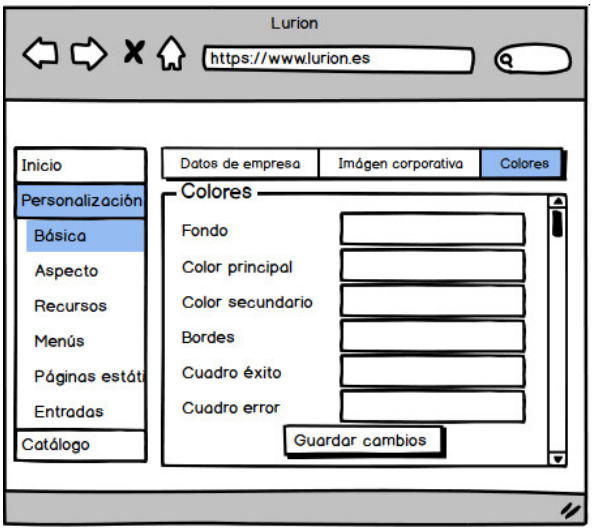


Figura A.4: Prototipo de baja calidad, Colores de la página

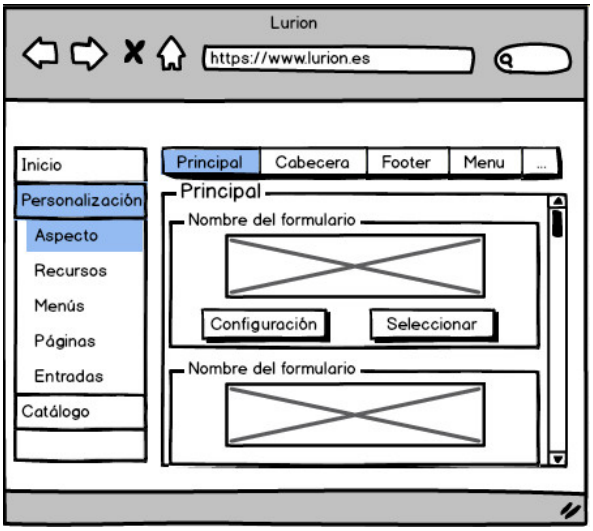


Figura A.5: Prototipo de baja calidad, Aspecto

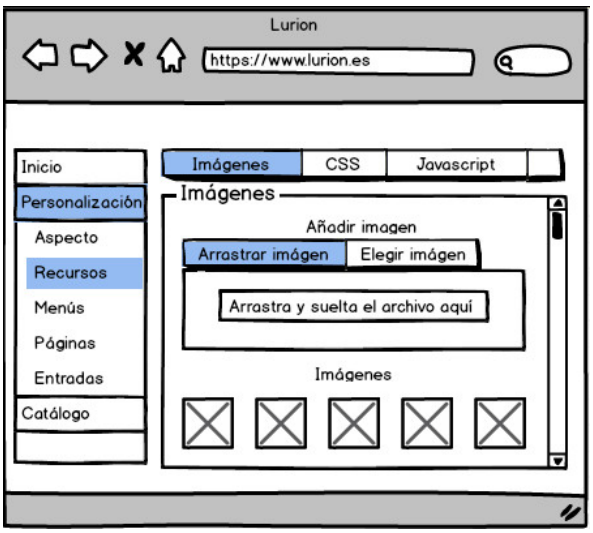


Figura A.6: Prototipo de baja calidad, Imágenes

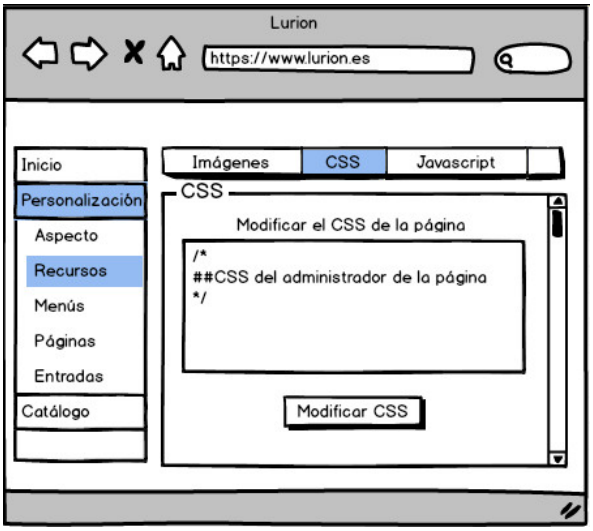


Figura A.7: Prototipo de baja calidad, CSS

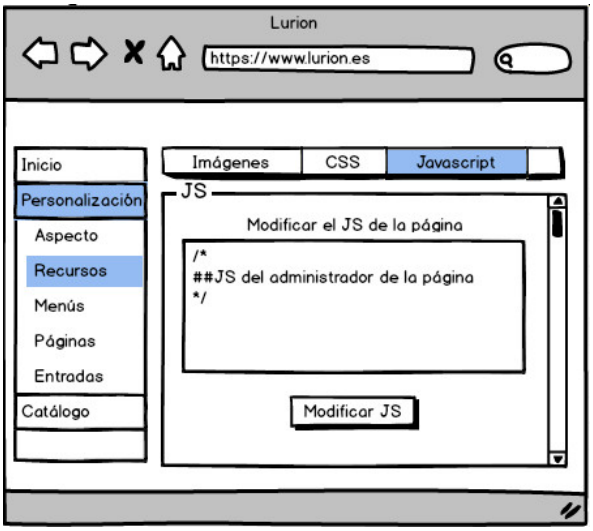


Figura A.8: Prototipo de baja calidad, JS

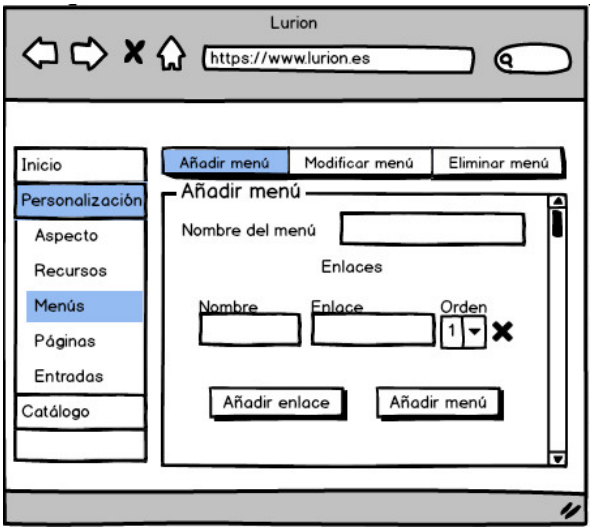


Figura A.9: Prototipo de baja calidad, Menú

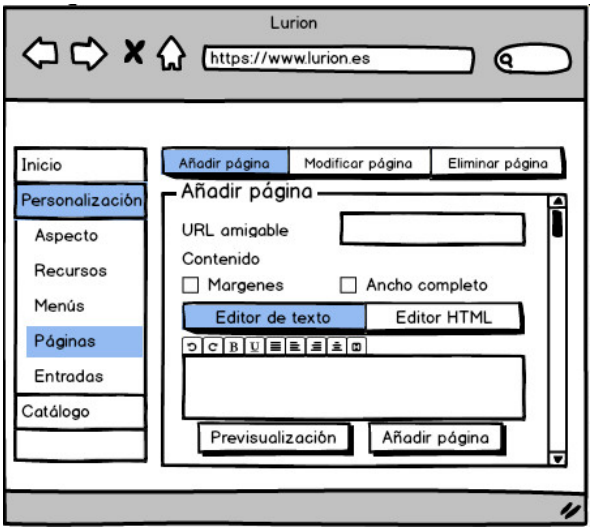


Figura A.10: Prototipo de baja calidad, Páginas estáticas, editor de texto

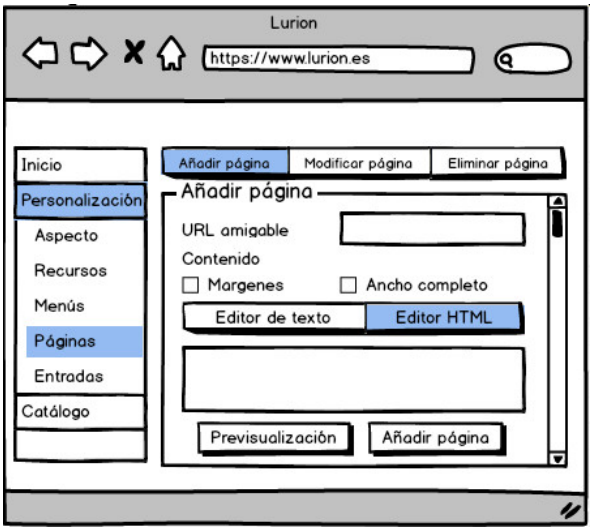


Figura A.11: Prototipo de baja calidad, Páginas estáticas, editor HTML

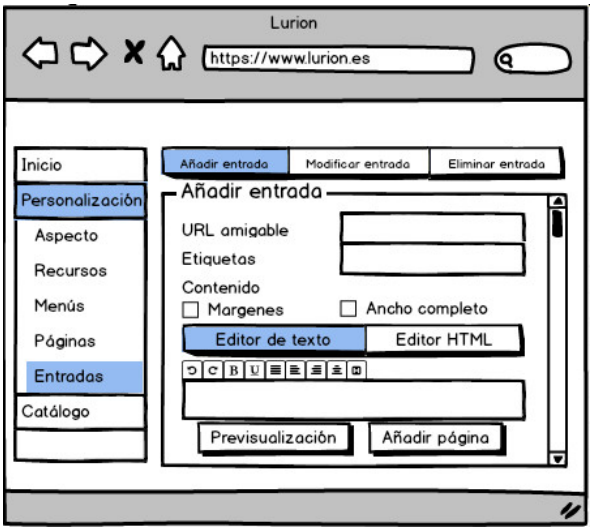


Figura A.12: Prototipo de baja calidad, Entradas, editor de texto

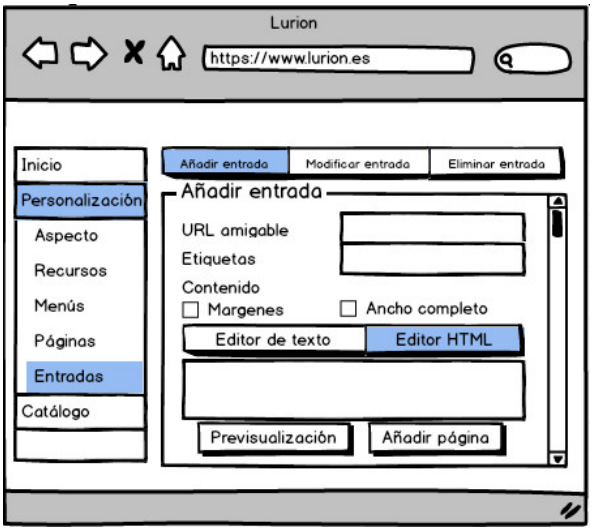


Figura A.13: Prototipo de baja calidad, Entradas, editor HTML

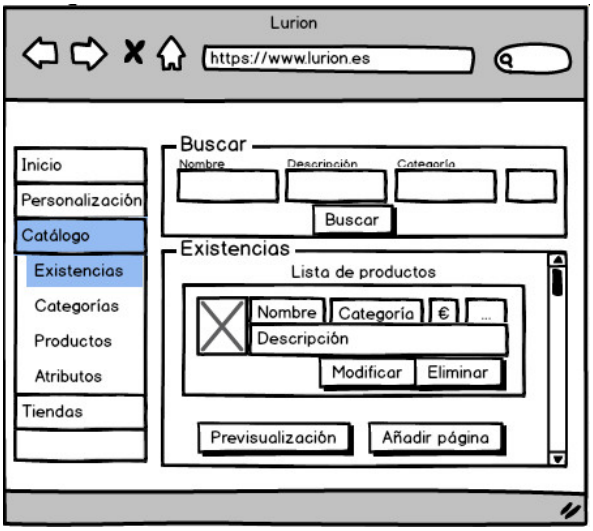


Figura A.14: Prototipo de baja calidad, Existencias

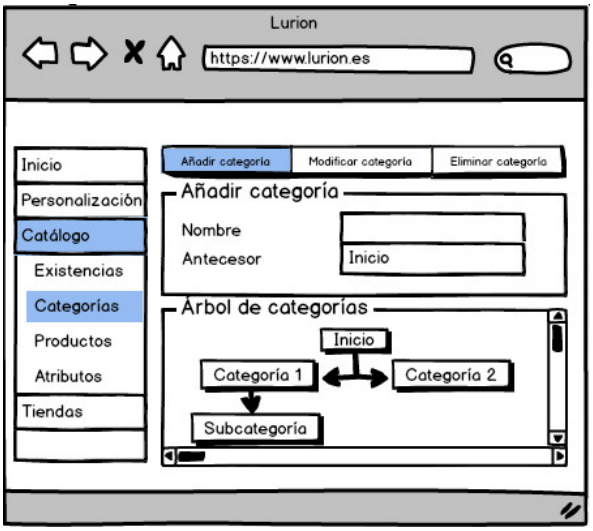


Figura A.15: Prototipo de baja calidad, Categorías

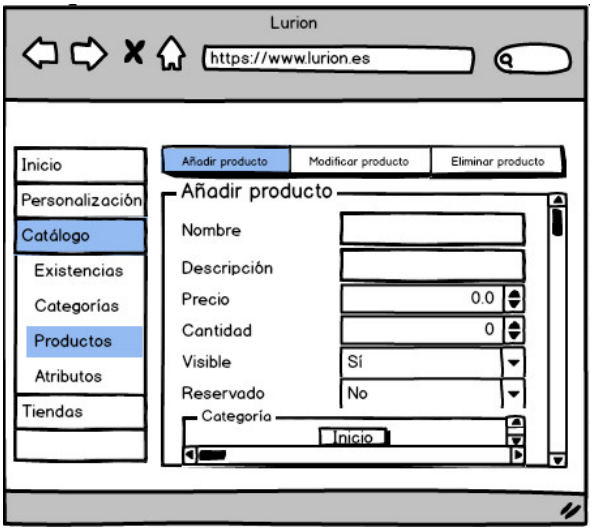


Figura A.16: Prototipo de baja calidad, Productos

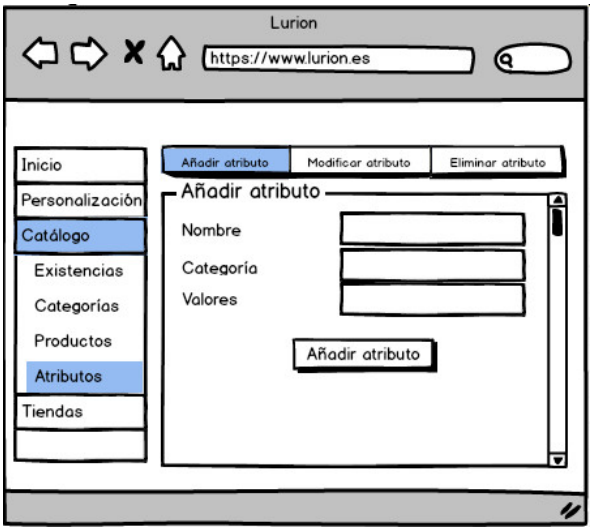


Figura A.17: Prototipo de baja calidad, Atributos

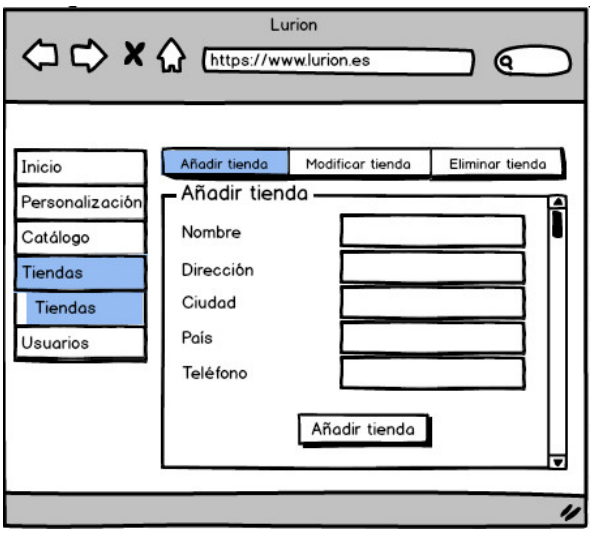


Figura A.18: Prototipo de baja calidad, Tiendas

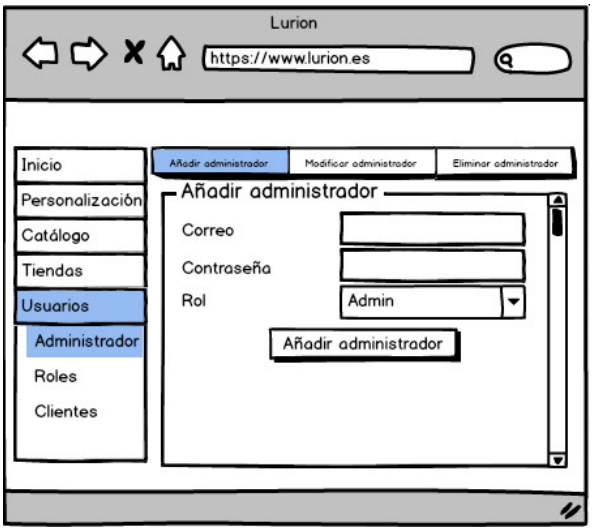


Figura A.19: Prototipo de baja calidad, Usuarios administradores

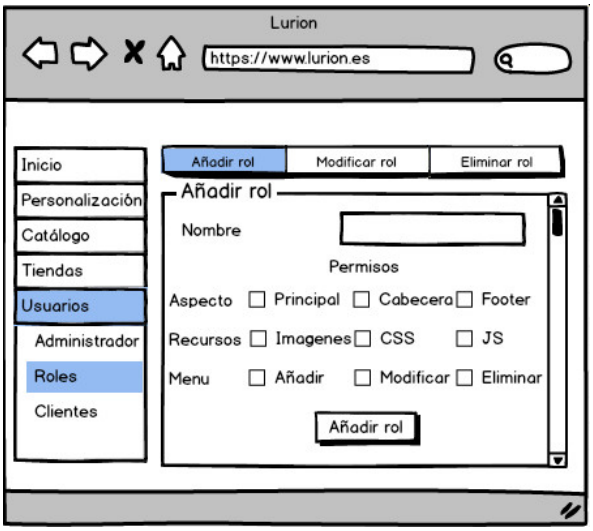


Figura A.20: Prototipo de baja calidad, Roles

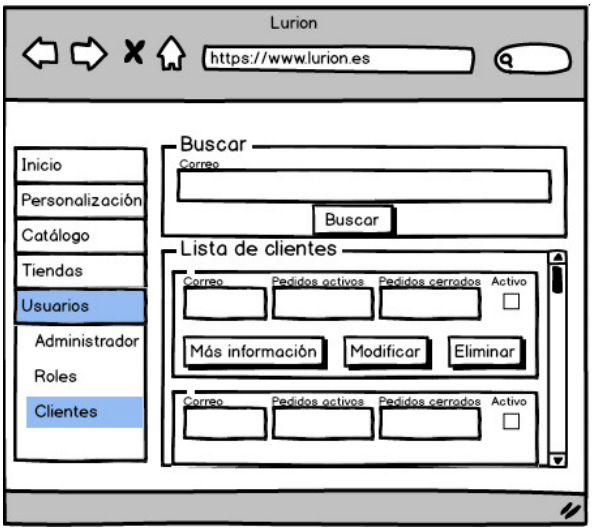


Figura A.21: Prototipo de baja calidad, Clientes

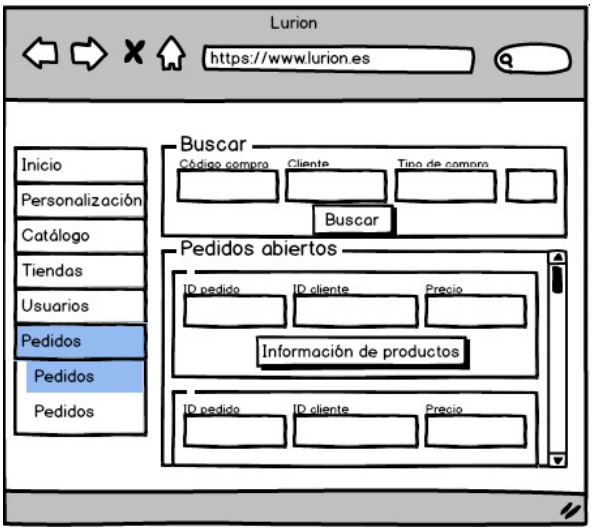


Figura A.22: Prototipo de baja calidad, Pedidos abiertos

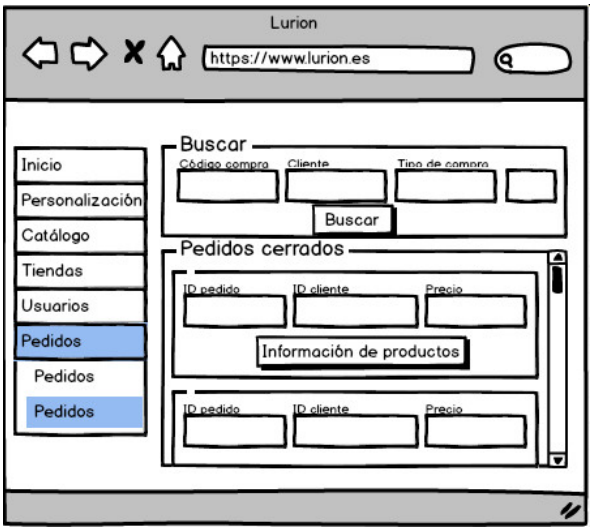


Figura A.23: Prototipo de baja calidad, Pedidos cerrados

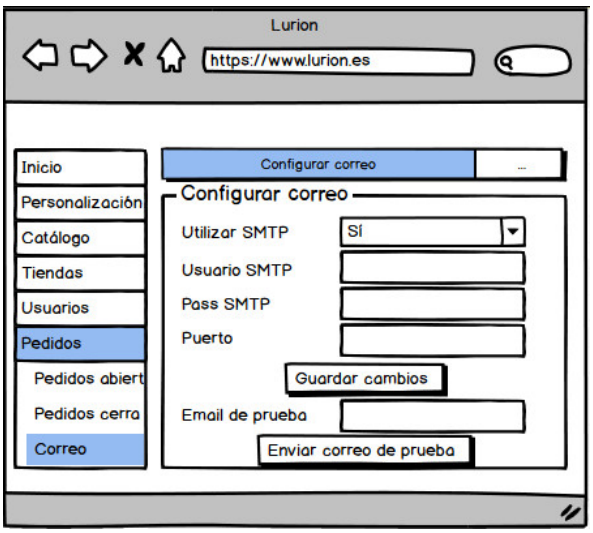


Figura A.24: Prototipo de baja calidad, Configuración de correos

BIBLIOGRAFÍA

FUENTES ONLINE

[1] J.J. Castro-Schez. *Comercio Electrónico*. 2019. URL: http://www.esi.uclm.es/www/jjcastro/coe (visitado 22-05-2019).

[2] Meet Jeff Sutherland y Meet Ken Schwaber. *Scrum Guides*. ScrumGuides. 2019. URL: https://www.scrumguides.org/.

FUENTES NO ONLINE

[3] Necla Bandirmali. «mtCMF: A novel memory table based content management framework for automatic website generation». En: *ScienceDirect* 58 (2018). DOI: http://dx.doi.org/10.1016/j.csi.2017.12.002.

[4] A. Iacovelli y C. Souveyet. *Framework for Agile Methods Classification, página 97*. Conference on Advanced Information Systems Engineering, 2008.

[5] Health Informatics Journal. «Identifying an appropriate Content Management System to develop Clinical Practice Guidelines: A perspective». En: *Sage* 23 (2015). DOI: http://dx.doi.org/10.1177/1460458215616264.

[6] José Manuel Martínez Caro - Antonio José Aledo Hernández - Antonio Guillen Pérez - Ramón Sánchez Iborra y María Dolores Cano. «A Comparative Study of Web ContentManagement Systems». En: *information* 9 (2018). DOI: http://dx.doi.org/10.3390/info9020027.

[7] Abdelkader Rhouati - Jamal Berrich - Mohammed Ghaouth Belkasmi y Toumi Bouchentouf. «Toward a solution to interoperability and portability of content between different content management system (CMS): Introduction to DB2EAV API». En: *Springer Link* 872 (2018). DOI: http://dx.doi.org/10.1007/978-3-319-96292-4_34.