Título: "AGROCOOP: DISEÑO, ELABORACIÓN, PROGRAMACIÓN DE TIENDAS ON-LINE PARA COOPERATIVAS AGRÍCOLAS."
SubTítulo: "COMO CREAR MERCADO ONLINE PARA COOPERATIVAS AGRICOLAS DESDE CERO"
TIENDA ONLINE AVANZADA(PHP)

AgroCoop Tienda Online Cooperativas Agrícolas

José Ignacio Segura Rama
MASTER OFICIAL SOFTWARE LIBRE
COMERCIO ELECTRONICO

Nombre Consultor:Francisco Javier Noguera Otero ,DANIEL RIERA TERREN -
TUTOR UOC:Ángel Baltasar Sánchez

CONSULTOR EMPRESA: CARLOS GONZALEZ CONTRERAS.

Fecha de Entrega:25/06/2016

FICHA DEL TRABAJO FINAL

Título del trabajo:	**AgroCoop Tienda Online Cooperativas Agrícolas**
Nombre del autor:	*JOSE IGNACIO SEGURA RAMA*
Nombre del consultor/a:	*Francisco Javier Noguera Otero*
Nombre del PRA:	***CARLOS GONZALEZ CONTRERAS***
Fecha de entrega :	06/2016
Titulación::	*MASTER OFICIAL SOFTWARE LIBRE*
Área del Trabajo Final:	*COMERCIO ELECTRONICO*
Idioma del trabajo:	*ESPAÑOL*
Palabras clave	*CSS,PHP,MYSQL,javascript*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El presente Trabajo Fin de Master versa sobre la elaboración, creación, programación, diseño y puesta en marcha de una **tienda online AVANZADA** para cooperativas agrícolas, queremos mostrar como poder crear una tienda e-comerce para cooperativas agrícolas de nuestro País. Con ello queremos así abrir el mercado de estas, ademas de mostrar como usar un gran compendio de tecnologías. Vamos a agrupar y poner en común, un conjunto amplio de distintas técnicas de programación web, que debidamente utilizadas podrán cooperar entre sí para lograr una aplicación totalmente operativa. Para ello usaremos una tecnología de Desarrollo de proyecto software rápido en espiral, de manera que se irán mejorando cambiando los requisitos y especificaciones del proyecto a medida que avanzamos y a medida
que se detectan nuevas necesidades. Sin embargo no nos olvidamos del ciclo de vida clásico de desarrollo de software también presente en la memoria.
Comprobaremos cada avance con versiones operativas, y una versión final adaptable a cualquier dispositivo cliente.
Dotar de una base a cualquier alumno, o profesional de la informática, de unos conocimientos y herramientas para crear un negocio online, totalmente programado, modular, flexible, ampliable y adaptable.
La tienda tiene herramientas de gestión automática back-end en la propia aplicación
Con este proyecto cerramos el circulo, pues se cumplen todos los requisitos previstos en el análisis, y todos los no previstos y detectados en el desarrollo. Se han cumplido el 100% de los objetivos. Aún así se proponen algunas mejoras, pues un proyecto es un Ente vivo que siempre se podrá mejorar, adaptar o crecer.

Abstract :

This Master's Thesis deals with the development, creation, programming, design and launch of an online store for ADVANCED agricultural cooperatives, weshow how to create an e-commerce store for agricultural cooperatives in our country. By this we open the market and these, in addition to showing how to use a large compendium of technologies. Let's group and bring together a broad set of different web programming techniques that properly used shall cooperate with each other to achieve a fully operational application. We will use a technology project development software fast spiral, so that will be improved by changing the requirements and specifications of the project as we move forward and as that new needs are identified. However we do not forget the classic cycle of software development also present in the memory life.

We check every advance operational versions, and a final version adaptable to any client device. Provide a basis to any student or computer professional, about knowledge and tools to create a fully programmed, modular, flexible, scalable and adaptable online business.

The store has automatic management tools on the back-end application itself. With this project we close the circle, as all the requirements are met analysis, and all unforeseen and detected in development. They have been met 100% of the targets. Still some improvements are proposed for a project is a living entity that can always improve, adapt or grow...

Thanks very Much!

TABLA DE CONTENIDO

AGRADECIMIENTOS

Agradezco a la comunidad educativa de la UOC, a mis compañeros de mi centro de trabajo, a mi mujer convaleciente en cama desde hace 6 meses (por haber soportado mis obligadas ausencias, pero que comprende la necesidad de atender mis responsabilidades) y a mis hijos mellizos de 3 años (que no entienden de responsabilidades, pero ayudan a mantener la moral alta). Todos habéis influido en la elaboración de este proyecto.
A todos vosotros, Gracias.

INTRODUCCIÓN

Nuestro **Trabajo de Fin de Master de Software Libre en la UOC**, trata la creación y puesta en marcha de una tienda online de productos producidos por Cooperativas agrícolas. Explicaremos los pasos más básicos y fundamentales del desarrollo del desarrollo de la tienda,para facilitar la labor de adaptar nuestro trabajo a nuevas webs con funciones similares, dinámicas y con un diseño estructurado y limpio.
Programaremos el código en texto plano, es decir, sin gestores de contenido, con el objetivo de aprender a escribir el código, pasando por el uso de **HTML, PHP, Javascript, jquery y CSS,** e implementación módulos externos como el de pago seguro mediante **PayPal,** resultando un proyecto libre, adaptable y gratuito al alcance de todos, ya que no requiere de conocimientos avanzados de ningún lenguaje, requiere capacidad lógica de programación y cierta experiencia en programar en cualquier lenguaje.

La web que programamos tiene en mente la sencillez, la seguridad y la accesibilidad, siempre desde los cimientos y mediante el uso de software libre, del mismo modo que este documento se atiene a **licencia de Reconocimiento-No Comercial-SinObraDerivada 3.0 España de Reactive Commons.**

Alojaremos el sitio durante su desarrollo en un servidor local de Apache y gestionaremos nuestra base de datos a través de phpMyAdmin. Todo esto instalado mediante el recopilatorio XampServer, cuyo nombre indica el uso de la infraestructura WAMP, aquella que usa Windows, Apache, MySQL y PHP.

Aunque programar sin el uso de herramientas más amigables (como el editor de páginas web) en principio puede parecer un trabajo arduo, no tiene por qué ser complicado, si tenemos paciencia y trabajamos paso a paso. Para ello usaremos editores de código libres.

Hemos querido realizar una web de venta de productos agrícolas debido a la escasa o nula facilidad para encontrar webs que englobe este tipo de productos, de distintas cooperativistas.

Creemos que al tener ciertas nociones de HTML, SQL, JavaScript, Jquery, PHP y CSS, este Proyecto es una gran oportunidad para consolidar, investigar y realizar un producto final, que a su vez pueda servir como guía en el futuro, a estudiantes, principiantes en la materia, o simplemente interesados.

A modo de introducción vamos a mencionar algunos de los objetivos que este proyecto pretende alcanzar:

Realizar una web no sólo funcional y de desarrollo libre que pueda ser replicada con facilidad. Crear una visualización y un manejo simple a la par de vistoso y práctico de cara al usuario. Afianzar los conceptos más fundamentales de PHP y CSS, así como conseguir una dominio y ayudarse de las posibilidades que ofrece JavaScript, Jquery, bootstrap a la hora de realizar operaciones en una página web. Aprender a estructurar correctamente la organización de los ficheros del sitio. Estudiar las soluciones de alojamiento más eficientes. Practicar la realización de consultas SQL a través de la web. Aprender todo lo posible sobre el mercado Online. Hospeda la tienda en un hosting. Que la tienda sea completamente operativa, por ejemplo,que realice perfectamente los pedidos de los clientes y el cobro de los mismos. Realizar pago seguro con paypal. Gestión automática de la tienda a través de la propia interfaz de la tienda(gestión clientes, pedidos, productos,estadísticas...).

OBJETIVOS

Enumeramos formalmente, a continuación, los objetivos finales que se pretenden alcanzar en este proyecto:

1. Realizar una web no sólo funcional y de desarrollo libre,y que pueda ser replicada con facilidad.
2. Desarrollar una visualización y un manejo simple a la par de vistoso y práctico de cara al usuario.
3. Afianzar los conceptos más fundamentales de PHP, SQL y CSS, así como conseguir una dominio de las posibilidades que ofrece JavaScript, Jquery, bootstrap a la hora de realizar operaciones en una página web.
4. Aprender a estructurar correctamente la organización de los ficheros de un sitio web.
5. Estudiar las soluciones de alojamiento más eficientes.
6. Dominar la ejecución eficiente de consultas SQL a través de la web.
7. Investigar todo lo posible sobre el mercado Online.
8. Hospedar en un hosting el resultado final.
9. Realizar correctamente los pedidos de los clientes y el cobro de los mismos.
10. Procesar pago seguro con paypal.
11. Procesar la gestión de la tienda, de forma automática, a través de la propia interfaz de la tienda(gestión clientes, pedidos, productos, estadísticas,trazabilidad,...).

1 - ESTUDIO DE VIABILIDAD

1.1 - Establecimiento del alcance del sistema, Coste económico:

Este proyecto debe ser eficiente en cuanto a costes de tiempo y dinero para ser viable. Teniendo en cuenta nuestra propia experiencia y las facilidades que daremos explicándolo y aportando el código fuente, estimamos entre 200 y 300 horas el tiempo de desarrollo, variando en función de los conocimientos previos del desarrollador y la complejidad y personalización que queramos alcanzar. Por tanto, el coste en tiempo es perfectamente asumible.

PRECIO €/HORA	NÚMERO HORAS	COSTE TOTAL €
8	200	1600
8	300	2400
10	200	2000
10	300	3000

1.2 - Estudio de la situación actual

Partiremos desde cero, pues aunque es una tienda online avanzada, se ha desarrollado todo desde cero, diseñando y creando la base de datos, insertando registros, haciendo prototipos, codificando y probando, etc. No usamos ningún software ni ningún hardware de base previo. Se necesitará alguna base física donde implantarlo, o alquilarlo, y el resto será software libre.

1.3 - Definición de los requisitos del sistema

Debemos tener en cuenta que una vez finalizada la web, deberíamos alojarla en internet, ya que no disponemos de servidor ni IP estática propia. O estudiar los costes de esta última opción.
Para ello, necesitamos:

- Un dominio para nuestra página.
- Una IP reservada (estática), necesaria para dominios que requieran un certificado SSL, es decir, las conexiones seguras que utilizaremos para el login y las pasarelas de pago.
- Un sistema de alojamiento para nuestra web.
- Soporte para mySql y PHP.
- Idealmente capacidad de manejar ficheros .htacces de Apache

1.4 - Estudio de las alternativas de solución

Los costes del dominio y la IP estática serán muy similares sin importar donde los pidamos, más allá de precios temporales de promoción para nuevos usuarios de servicios en línea. La diferencia la marca la decisión de alojar nuestra web en un servidor propio o en un servicio de alojamiento online.
Aunque el alojamiento online tiene la ventaja de que la empresa contratada se hace cargo del mantenimiento del equipo, el coste de estos servicios es, en todas las ofertas observadas, muy elevado para nuestras capacidades en esta fase temprana del proyecto. En cualquier caso es mucho más económico y eficiente a medio-largo plazo tener un servidor propio. Por ejemplo, un servicio de alojamiento con una capacidad de apenas 500GB supone unos costes de casi 500€ al año en el caso de solicitar un servidor con un procesador de 4 núcleos y 8 GB de RAM, cuando un equipo con una potencia similar y un almacenamiento ampliamente superior, y por tanto más duradero, tendría un coste inferior al de un año de alojamiento en línea.

Según fuentes de acceso público disponibles:

Servicio	Precio	Capacidad	Características	Pluses
OVH	10€ / mes	500GB	2GB RAM 1 Núcleo	Estructuras de BBDD ilimitadas
1&1	15$ / mes	100GB	2GB RAM Sin información sobre Núcleos	Protección DDOS Certificado propio SSL 1GB SSD para estructuras de BBDD
Servidor local	20€ / año de direccionamiento estático 10€ / año de dominio 50€ mensuales entre luz y fibra Inversión inicial de 350€ en servidor + dos discos duros de 1TB (producción y respaldo). Velocidad: 30Mgabits/segundo 300Megabits/segs	1TB	4GB RAM 2 Núcleos	

1.5 - Valoración de las alternativas

Hay que tener en cuenta que un servicio contratado en Internet debería asegurar las copias de respaldo.Por otra parte, al ser un servicio virtual no tiene obsolescencia en la que tengamos que intervenir, y por lo tanto renovar equipos y software. Además aportan ventajas muy interesantes como la protección frente a ataques o la gestión de las bandejas de email para evitar ataques. Aunque son buenas ventajas, no son lo suficientemente atractivas para competir con un equipo privado mucho más barato y que en principio tendremos que aplicar una serie de Políticas de Seguridad en los equipos en los dispositivos de red, en el software de Base, y el el proyecto (firewalls, IDS, acls, copias seguridad, procesos de respaldo,...).

1.6 - Selección de la solución

Así pues, alojaremos el sitio en un equipo privado, más concretamente el HP 819185-421 con dos discos duros WD Red, especiales para NAS y otros tipos de equipos preparados para estar encendidos ininterrumpidamente.
El equipo del que hablamos es el siguiente:

2 - ANÁLISIS DEL SISTEMA

2.1 - Definición del sistema

Nuestra tienda online debe cumplir con unos mínimos de usabilidad, vistosidad, ergonomía, y facilidad de uso. Es decir, comodidad a nivel de desplazamiento, funcionalidades y condiciones visuales.
Hemos elaborado el diseño conceptual con el esquema de BD relacionales, gráficos Entidad Relación y diseño Relacional.

2.2 - Establecimiento de requisitos

Debemos hacer accesible (y visible en todo momento), el "carrito de la compra" y permitir su gestión, para añadir, eliminar o variar la cantidad de los productos en el mismo. También hemos progrmados, con la ayuda de la API de Pay Pal, un sistema de pago seguro. Hemos estandarizado las imágenes de los productos en tamaño, colocación, orden, etc. También intentaremos plantear el modo de visualizar las imágenes dependiendo de la plataforma(PC,tablet, movil) para la que se está visualizando el producto en cuestión.

A la hora de formalizar legalmente nuestro sitio online, el primer paso es darnos de alta en el epígrafe del Impuesto de Actividades Económicas **(IAE)**. En general, los requisitos para abrir una tienda en línea son los mismos que para una tienda física, salvo porque no es necesaria una licencia de apertura(en algunos municipios siquiera este requisito es necesario en la tienda física).

-Ley de Ordenación del Comercio Minorista, nos atenemos a los artículos que afectan a las tiendas de venta a distancia.

-La Ley de Servicios de la Sociedad de la Información y Comercio Electrónico exige datos que permitan identificar a la empresa como el nombre, los datos de contacto o el NIF, y asegura que los precios estén marcados y expliquen si se incluyen los impuestos, los gastos de envío, etc.

-La Ley Orgánica de Protección de Datos nos pide identificar los ficheros con información sensible, organizar estos datos en niveles de seguridad, elaborar un documento de seguridad, una política de privacidad propia,etc.

-Es obligatorio redactar las condiciones de uso, que recogen los derechos y obligaciones de clientes y usuarios, que además deben aceptar antes de comprar.

-**La ley de consumidores y el comercio electrónico** estipula ciertas condiciones de protección al comprador, como que se debe mostrar de forma clara e inequívoca el precio del producto, que el plazo de devolución de un producto es de 14 días naturales, etc.

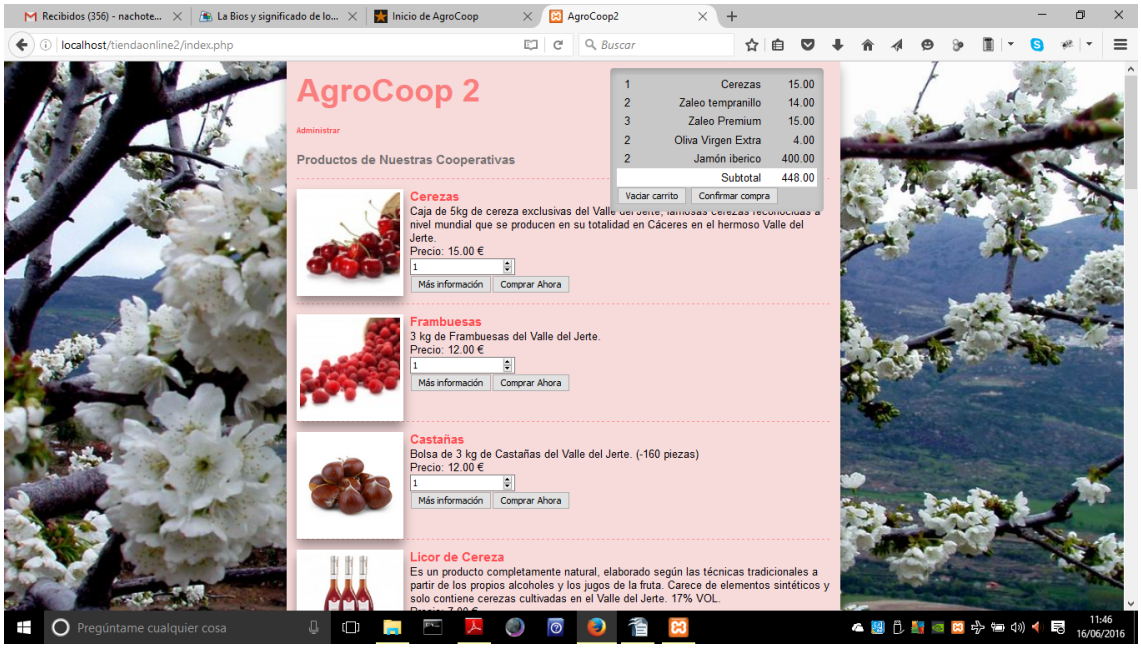
2.3 - Definición de interfaces de usuario

El diseño de la página será lo más minimalista posible, ya que es el diseño más aceptado y por tanto amigable, además de generar confianza en el usuario. Es fundamental para nosotros separar de forma adecuada los ficheros y módulos con el uso de "includes", los CSS, etc,
Para hacer la página lo más modular y práctica posible a la hora de la fase de MANTENIMIENTO, además para mayor facilidad de adaptación, mejora.
Para las pruebas, personalizaremos suites de tests de Selenium. El software y las licencias ya se han comentado con anterioridad, serán especificados más adelante junto al hardware elegido para soportar nuestro servidor.
El usuario estándar puede navegar por la página, añadir productos al **carrito,** mirar sus descripciones, realizar los encargos logueándose y pagando mediante **PayPal,** y acceder a su biblioteca, donde tendrá las claves de los productos que ha comprado a la vista. Los administradores pueden acceder a la página de administración desde la barra de navegación, pasarán por un login que sólo permitirá acceso a aquellos señalados como administradores en la tabla de usuarios.
Dentro de la página de administradores(gestín automática de la tienda), podrán gestionar los pedidos,gestionar los clientes, gestionar los productos ysin necesidad por parte del usuario de conocer SQL ni PHPMyAdmin, además de tener disponible una página de estadísticas sobre los productos más vendidos y los compradores que más beneficios nos reportan.

La parte de estadísticas, no servirá en un futuro, para conocer mejores clientes, y realizar ofertas o rankings, mejores productos para actualizar precios, y muchas otras utilidades que retro-alimentan a la propia tienda, y permiten adaptarla y mejorarla a las necesidades reales de los usuarios.

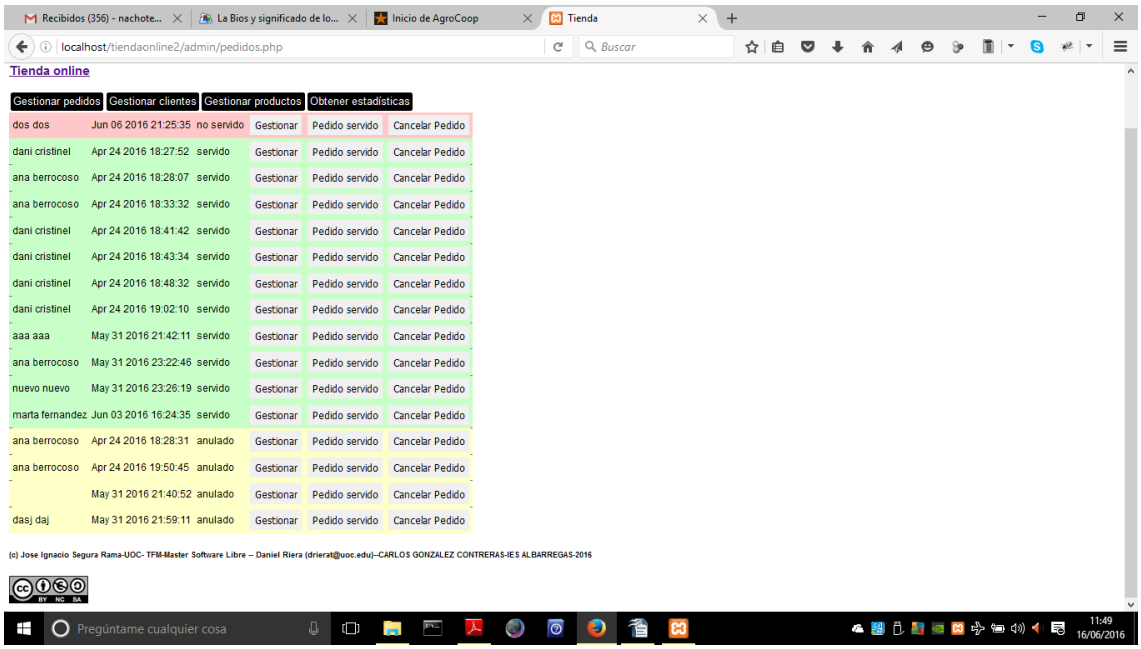
Queremos que en nuestra página principal en horario nocturno, se muestre más oscura, más agradable a la vista en horas de noche. Encontraremos los productos de los que tengamos existencias y que estén activados en nuestra base de datos. Los productos van acompañados de su imagen de cabecera, una descripción corta, el precio, un botón para acceder a su ficha y otro botón para añadir al carrito las unidades requeridas. También puedes acceder a tu biblioteca de productos comprados, desde la interfaz del carrito, vaciar el carrito y acceder al menú de confirmar compra.

La Interfaz de la página principal:

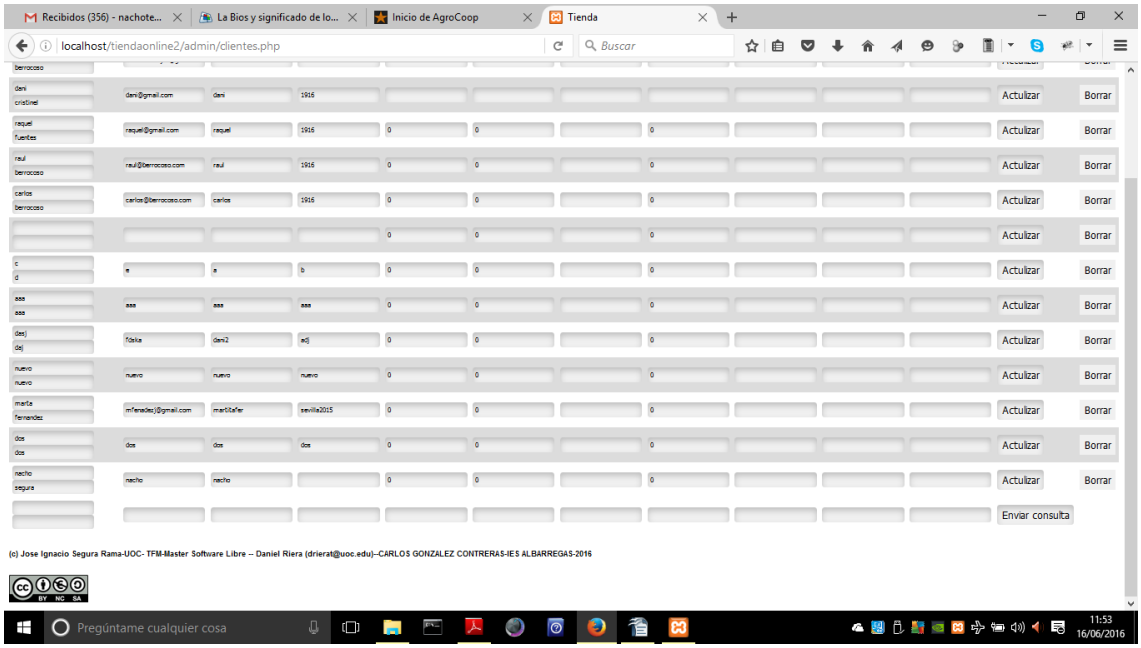


La siguiente es la ficha de uno de nuestros productos. Arriba la imagen de cabecera, el título, después la descripción larga, la galería de imágenes, y finalmente otros datos de interés como el género, el tamaño, el precio y un botón de añadir a carrito.

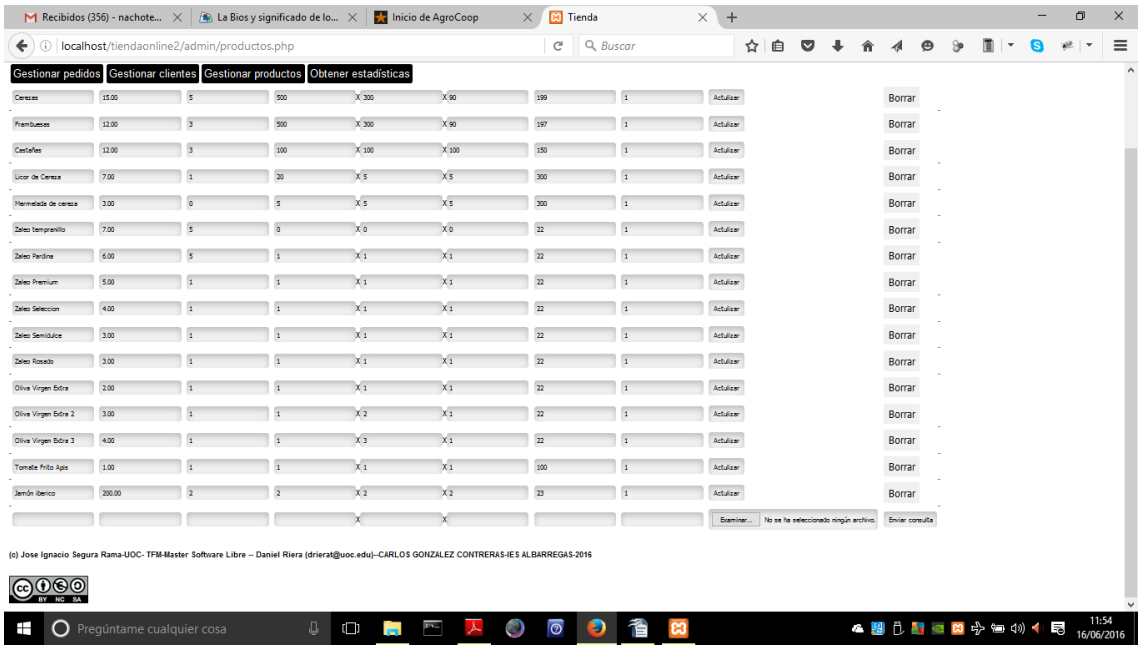
Gestión de Pedidos. Podemos de un vistazo observar el estado de los pedidos, cambiarlos, cancelarlos, etc



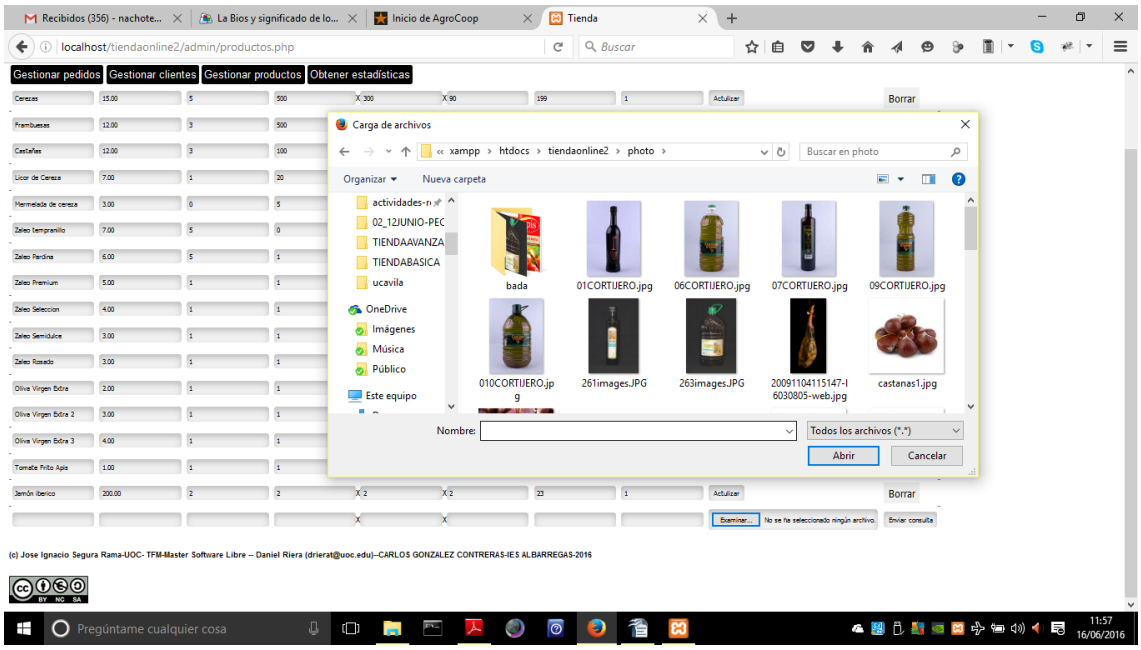
La anterior es una de las páginas de la interfaz de administrador, la de gestión de pedidos. Antes hemos tenido que pasar por el index de admin, con un login como el que hemos visto a la hora de confirmar compra, pero que sólo deja acceso a los usuarios autorizados en la base de datos.



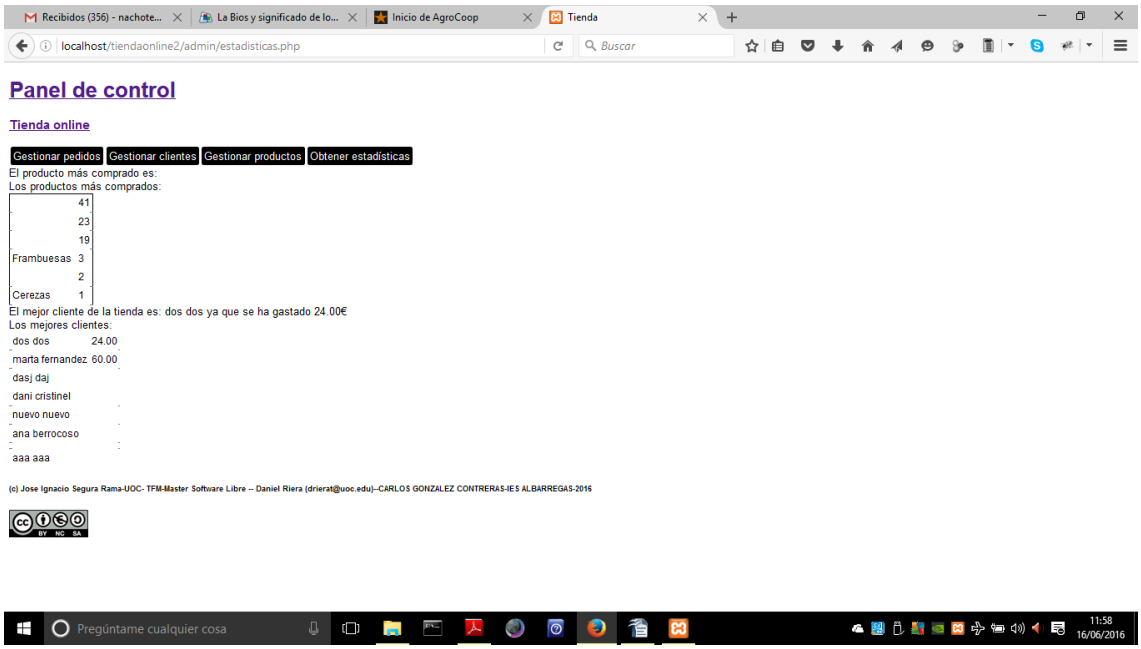
La anterior es la página de gestión de clientes, donde podemos añadir, editar o eliminarlos.



Aquí gestionamos los productos del mismo modo que con los clientes, con la siguiente excepción…



Al añadir un nuevo producto podemos explorar para añadir una imagen al producto, que se subirá automáticamente, creando la carpeta con el nombre del producto en el equipo servidor.



Y por último, la página de estadísticas. Aquí mostramos los productos más comprados y los clientes que más dinero han gastado. Algunas de las ideas de mejora que tenemos es mostrar el top de ventas con las imágenes de cabecera de los productos más vendidos, o incluso mostrar el top en la página principal. También tenemos pensado utilizar la tabla de registros que guardamos para mostrar nuevas estadísticas que nos permitan llevar un mejor seguimiento del comportamiento de nuestros clientes.

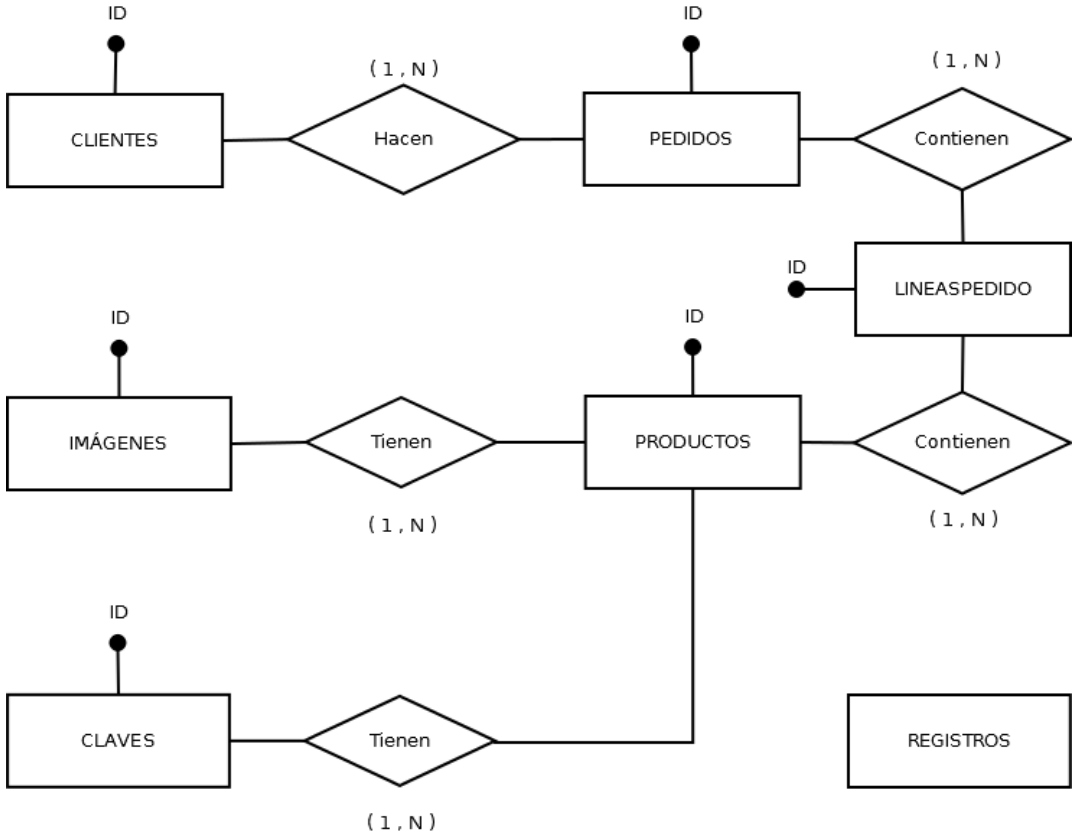
2.4 – Especificación del plan de pruebas

Hemos preparado una suite de pruebas con Selenium IDE, un add-on para Firefox que se encarga de simular nuestros movimientos y hacer las comprobaciones que le indiquemos.
La suite no es perfecta, puesto que al volver de PayPal a nuestra web, aún en una versión de desarrollo sin seguridad SSL, Firefox pide una confirmación que Selenium no puede dar. Para los demás casos, recorrerá la zona de usuarios comprobando que el carrito funciona bien y no se muestra ningún mensaje de error, y accederá a las páginas de administración para comprobar que tampoco se muestra un error en ninguna de ellas. Los tests están adjuntos como anexo.
Aparte de lo anterior se han hecho pruebas de Carga con muchos usuarios, y pedidos en un momento puntual.
Se han hecho pruebas de Límites de valores.
Se han diseñado y ejecutado pruebas caja negra, comprobando la funcionalidad del sistema, ante entradas controladas se han dado las salidas esperadas.
Se han ido realizando y pruebas caja blanca, comprobando la eficiencia de la codificación y algoritmia.
Se han hecho pruebas de integración entre componentes, y pruebas de implantación.

3 - Diseño del sistema

3.1 – Arquitectura

En este punto trataremos sobre la estructura y composición de la base de datos y de los archivos web. Para empezar mostraremos tras un **diagrama entidad-relación (E/R)** las tablas que componen la base de datos:



En el diagrama vemos que todas las tablas tienen una relación de (1, N) o de uno a muchos, todas las claves primarias de las tablas tienen de nombre ID y que la tabla productos está relacionada con otras dos tablas (imágenes y claves). Para terminar de hablar de la base de datos, para ello hemos incluido un **esquema Relacional que parte del anterior esquema E/R.**

- **Estructura para la tabla "claves":**
  la usaremos en un futuro para comprobar la trazibilidad del producto, o para generar un código QR único, o mejoras posibles.

Columna	Tipo	Nulo	Predeterminado
***id**	int(20)	No	
idproducto	int(11)	Sí	NULL
clave	varchar(50)	Sí	NULL

- **Estructura de tabla para la tabla "clientes":**

Columna	Tipo	Nulo	Predeterminado
***id**	int(4)	No	
nombre	varchar(30)	Sí	NULL
apellidos	varchar(30)	Sí	NULL
email	varchar(30)	Sí	NULL
user	varchar(30)	Sí	NULL
pass	varchar(30)	Sí	NULL
tlf	int(9)	Sí	NULL
dirección	varchar(30)	Sí	NULL
zip_code	int(5)	Sí	NULL
población	varchar(30)	Sí	NULL
país	varchar(30)	Sí	NULL
admin	varchar(3)	No	no

- **Estructura de tabla para la tabla “imágenes”:**

Columna	Tipo	Nulo	Predeterminado
***id**	int(4)	No	
idproducto	int(4)	Sí	NULL
imagen	varchar(200)	Sí	NULL
titulo	varchar(30)	Sí	NULL

- **Estructura de tabla para la tabla “lineaspedido”:**

Columna	Tipo	Nulo	Predeterminado
***id**	int(4)	No	
idpedido	int(4)	Sí	NULL
idproducto	int(4)	Sí	NULL
unidades	int(4)	Sí	NULL

- **Estructura de tabla para la tabla “pedidos”:**

Columna	Tipo	Nulo	Predeterminado
***id**	int(4)	No	
idcliente	int(4)	Sí	NULL
fecha	varchar(20)	Sí	NULL
estado	varchar(15)	Sí	NULL

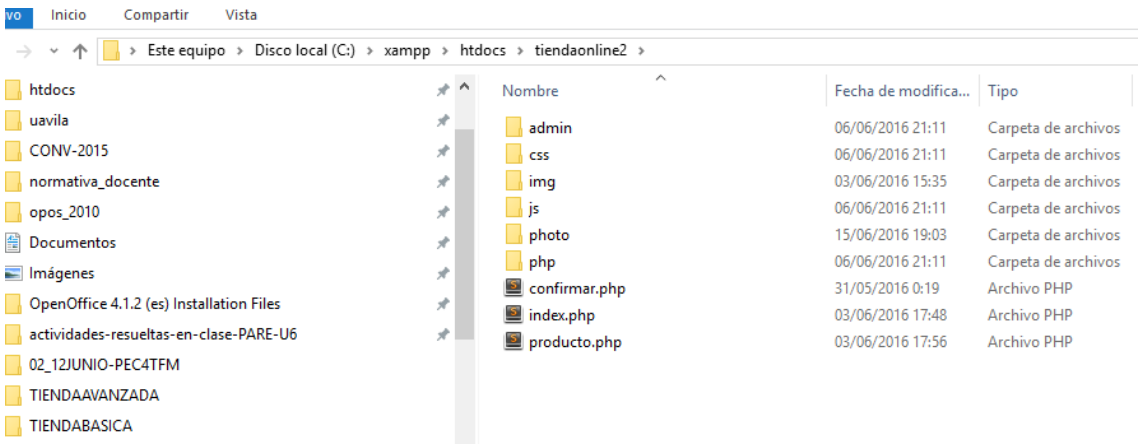
- **Estructura de tabla para la tabla “productos”:**

Columna	Tipo	Nulo	Predeterminado
***id**	int(4)	No	
nombre	varchar(50)	Sí	NULL
genero	varchar(30)	Sí	NULL
descripción	varchar(1000)	Sí	NULL
precio	float	Sí	NULL
tamaño	float	Sí	NULL
plataforma	varchar(30)	Sí	NULL
existencias	int(4)	Sí	NULL
activado	varchar(3)	Sí	NULL

- **Estructura de tabla para la tabla “registros”:**

Columna	Tipo	Nulo	Predeterminado
utc	int(100)	No	
anio	int(4)	No	
mes	int(2)	No	
dia	int(2)	No	
hora	int(2)	No	
minuto	int(2)	No	
segundo	int(2)	No	
ip	varchar(255)	No	
navegador	varchar(255)	No	
pagina	varchar(255)	No	

Referente a los archivos web, hemos de comentar que están separados en carpetas según su tipo o uso. La segmentación del directorio web es la siguiente:



Basándonos en el esquema anterior, podemos ver que en la raiz tenemos los archivos principales para la web en formato .php. Dentro de la carpeta "admin" tenemos los archivos necesarios para la administracion de la web. Estos archivos son los siguientes:

- Actualizarclave.php
- Actualizacliente.php
- Actualizarproducto.php
- Cabecera.inc
- Claves.php
- Clientes.php
- Eliminarclave.php
- Eliminarcliente.php
- Eliminarproducto.php
- Estadisticas.php
- Index.php
- Logadmin.php
- Nuevaclave.php
- Nuevocliente.php
- Nuevoproducto.php
- Pedidos.php
- Piedepagina.php
- Productos.php
- ...

- La descripción de todos estos archivos y sus funciones se hará en posteriores apartados.

4 - DESARROLLO

4.1 - Planificación de las actividades de desarrollo e integración de sistema:

Antes de iniciar el desarrollo, es necesario que tengamos en consideración algunos aspectos importantes que afectarán a nuestro trabajo.

En primer lugar debemos escoger un licencia para distribuir el Proyecto, dicha licencia deberá ser libre, para que otros interesados puedan estudiar el mismo y puedan mejorarlo o adaptarlo, en realidad esto es la base del conocimiento.

Si hemos decidido alojar nuestra aplicación web, deberemos elegir el hosting adecuado a nuestro presupuesto y nuestras necesidades, por ejemplo comprobar que tiene soportado PHP y la versión del mismo, que permite un servidor de mysql, y ficheros .htacces de configuración.

Si lo alojamos en un servidor dedicado propio, debemos contratar una linea de fibra de 30Mbs/seg de subida, IP publica estática, y un dominio, ademas de dedicar el servidor hardware elegido o virtualizarlo en otra máquina superior.

Deberemos descargarnos todo el software de desarrollo necesario, en este caso todo el software será libre, editores de código con resaltado, base de datos relacional, servidor web, navegadores...etc.

Una vez descargado todo el software anterior, lo instalamos y hacemos las pruebas de funcionamiento necesarias, así como su acoplamiento unos con otros y la configuración adecuada.

Aunque la documentación es una tarea que está planificada en todas las fases del proyecto desde el inicio hasta el final, es en esta fase cuando se inicia otro tipo de documentación la documentación técnica o de programador, y la menos técnica o de usuario. Ambas podrá servir posteriormente para elaborar los planes de formación.

Una vez iniciada la fase de desarrollo, tendremos que tener elaborado un plan de pruebas del código, al menos pruebas unitarias de los módulos programados, y pruebas de integración entre módulos.

Como en el Plan de trabajo habíamos dividido el proyecto en una serie de tareas básicas, vamosa ayudarnos en parte de esa planificación. Vamos a ir planificando pequeños Sprints o planes de trabajo autónomos, en los cuales se enumeran una serie de tareas concretas a realizar y se establece un periodo pequeño a concretar. Tras ese periodo se reúne el equipo de programadores y comprueban el nivel de cumplimiento de ese sprint, y se prepara el siguiente sprint con nuevas tareas detectadas. Con este sistema se consiguen entre otros una serie de beneficios, motivación, control del tiempo, rectificaciones y mejoras en el momento. No se pueden planificar los sprints, pero a modo de ejemplo podríamos haber realizado los siguientes sprints.

SPRINT 1 (2 semanas)	esfuerzo
Prototipado de base de datos para crear tienda online	1
Hoja de productos para nuestra tienda online	1
Hoja de clientes para nuestro proyecto de e-commerce	1
Almacenamiento de los pedidos de nuestra tienda	2
Líneas de pedido y simulación	2
Inserción de imágenes en en la BBDD	3
Elección del motor de base de datos en función de la seguridad	3
Imágenes de los productos de la tienda online	1

SPRINT 2 (1 semana)	esfuerzo
Creación de una estructura de directorios	1
Creación y maquetación de la plantilla principal de la tienda	2
Política de páginas múltiples	2
Carga de imágenes principales de productos	1
Petición a la base de datos	2
Segunda carga de imágenes secundarias de productos	1
Creación de botones y enlaces a productos	2
Página de producto concreto detallada	2

SPRINT 3 (1 semana)	esfuerzo
El carrito de compra	3
Descarga de jQuery para usarla en el proyecto; Botones con código;	1
Carrito en JavaScript	2
Uso en nuestra tienda online	1
Variables de sesión;	2
Múltiples líneas de pedidos;Suma de la compra	3
Vaciar carrito.	1

SPRINT 4 (1 semana)	esfuerzo
Implementación del proceso de compra;Formulario de log-in;	2
Validación del cliente en la base de datos;	1
Creación del pedido en la base de datos;	2
Obtención del ID de pedido;	2
Redirección a la pantalla de confirmación;	1
Vaciado de la memoria del carrito;	1
Control de existencias	2

SPRINT 5 (1 semana)	esfuerzo
Panel de control de los eventos de la tienda;	3
Gestión de pedidos de la tienda online;	2

Listado con Join;Alias de tabla;	1
Dar pedidos por servidos;	1
Cancelación de pedidos	1

SPRINT 6 (2 días)	esfuerzo
Estilizado con CSS de la tienda online;	3
Variables de configuración de la tienda online.	1

SPRINT 7 (1 semana)	esfuerzo
Gestión de los productos y de los clientes;	2
Creamos un listado de productos para nuestra tienda;	1
Un nuevo producto con un formulario;	2
Procesamiento del formulario anterior;	2
Eliminación de productos del listado;	1
Formulario de actualización de productos;	2
Procesado de la actualización de producto;	1
Obtención de un listado de clientes;	1
Añadido de un nuevo cliente a la base de datos;	2
Borrado de clientes de la base de datos;	1
Actualización de los datos de un cliente	1

SPRINT 8 (3 días)	esfuerzo
Registros de visitantes;	3
Creación de un archivo para registrar visitantes;	2
Creación de la tabla de registro en la base de datos;	1
Script de guardado de registros en la página.	2

SPRINT 9 (4 días)	esfuerzo
Estadísticas	2
Formulado de una petición SQL;	1
Mostramos las estadísticas en pantalla;	1
Petición SQL de consulta mejores clientes;	1
Matizando el alcance de la petición multitabla;	1
Devolución de datos de estadísticas en pantalla	1

SPRINT 10 (4 días)	esfuerzo
Registro de nuevos usuarios en nuestra tienda	2
Formulario de alta de nuevo cliente	2
Procesado del formulario de alta de nuevo cliente	1
Redirigir a pedidos;	1

Evitando duplicados.	2

SPRINT 11(1 semana)	esfuerzo
Pedido de múltiples unidades;	2
Introducción de un campo numérico;	1
Captura del campo numérico;	1
Pase de datos al carrito con AJAX;	3
Multiplicación y suma de unidades pedidas;	3
Número de unidades en la base de datos	1
Actualización de las estadísticas.	1

SPRINT 12(1 semana)	esfuerzo
Subida automática de imágenes de nuestros productos;	2
Formulario de subida de imágenes de productos;	2
Procesamiento de subida de imágenes de productos;	2
Directorios relativos;	2
Limitación de archivos de subida	1

SPRINT 13(1 semana)	esfuerzo
Estilizado del panel de control;	2
Cabecera y pie de pagina de la tienda online;	1
Estilo del menú de navegación de la tienda online;	2
Estilizado de tablas de la tienda online;	2
Contador para estilizado de la tienda online	2

4.2 – Desarrollo

Como hemos mencionado anteriormente, instalamos el software necesario en este caso xammp server que es un paquete que viene muy bien empaquetado con servidor apache, soporte php, servidor aplicaciones tomcat, mysqul y phpmyadmin. Usaremos gedit, notepad++. O sublime text, para editor de código con ayuda de remarcado de texto.

Para las pruebas usaremos Selenium IDE sobre Firefox, que utilizaremos al final para la automatización de pruebas

Empezamos a bosquejar lo que será el plan de purebas que se va a comentar más adelante.

4.3 – Documentación (Código)

Vamos a comentar de forma completa pero no demasiado detallada las distintas páginas de código de nuestra web, desde las páginas visibles al usuario hasta los documentos de las páginas de administración, cabecera, etc.

Index.php

```php
<?php include 'php/cabecera.inc' ?>
<?php include 'php/config.inc' ?>

<?php

$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion="select * from productos where existencias > 0 and activado='yes'";
$resultado=mysqli_query($conexion,$peticion);

while($fila=mysqli_fetch_array($resultado)){

    echo "<article>";
    $peticion2="select * from imagenes where idproducto=".$fila['id']." limit 1";
    $resultado2=mysqli_query($conexion,$peticion2);

    while($fila2=mysqli_fetch_array($resultado2)){
        echo "<img src='".$fila2['imagen']."'>";
    }

    echo "<a href='producto.php?id=".$fila['id']."'><h3 id='titulo'>".$fila['nombre']."</h3></a><br>";
    echo "<p>".$fila['descripcion']."</p>";
    echo "<br><h3>Precio: ".$fila['precio']." €</h3>";

    echo "<div id='botones'><center>";
    echo "<a href='producto.php?id=".$fila['id']."'><button>Más información</button></a>";
    echo "<button value='".$fila['id']."' class='botoncompra'>Añadir a carrito</button></a>";
    echo "<input type='number' value='1' max='".$fila['existencias']."' min='1' id='num".$fila['id']."'> unidad/es.";

    echo "</center></div></article>";
}
mysqli_close($conexion);

?>

<?php include "php/piedepagina.inc" ?>
```

Este es nuestro index.php, la página principal donde aparecen los productos a la venta. La parte en el recuadro **negro** es en la que incluimos dos archivos externos, uno es cabecera.inc, que carga algunos datos necesarios para que el resto de la página funcione correctamente, como nuestro **css, y config.inc,** que incluye los datos de conexión a la base de datos.

En el recuadro azul realizamos la conexión a la base de datos (que al final del documento terminará con mysqli_close) y especificamos el uso del set de codificación de caracteres utf-8. En el recuadro rojo tenemos la primera consulta sql, que especificamos en $peticion, pidiendo todos los datos de la tabla productos donde haya existencias y el campo de activación esté a “yes”, para luego ejecutar esta consulta con mysqli_query.

En el recuadro verde entramos en un while que recorre el array de resultados obtenidos en la consulta anterior, que se conservan en $resultado, de modo que en cada iteración situamos una etiqueta <article> y realizamos una consulta en busca de la imagen de cabecera del producto cuya id está siendo utilizada en ese momento. Nos basta con un “limit 1” porque el nombre de la imagen de cabecera siempre es “header.jpg” y el resto, las imágenes de la galería, tienen nombres numéricos, por lo que alfabéticamente es el primer resultado que va a devolver la consulta. También podríamos encontrarla con “and titulo=’header.jpg’”.

En el recuadro naranja seguimos dentro del anterior while cuando lanzamos un nuevo bucle que se encargará de mostrar la imagen encontrada en la anterior consulta, por lo que al ser una única imagen el bucle no es necesario, pero conservamos el código con la futura idea de mejora de mostrar las distintas imágenes en ese lugar de forma dinámica, quizá mediante una ventana interactiva o simplemente cargando todas las imágenes en bucle. Después, escribimos los datos del producto, sus botones y el selector de unidades, todo sujeto a la id del producto.

Cabecera.inc

```php
<?php   session_start();
    if(!isset($_SESSION['contador'])){$_SESSION['contador'] = 0;}
?>
<!doctype html>
<html lang="es">
    <head>
        <title>GetItCheap - Las mejores ofertas, en tus manos...</title>
        <?php
            if (date('H')>19 or date('H')<7){
                echo '<link rel=Stylesheet href="/css/moviln.css" media="screen and (max-width: 1365px)">';
                echo '<link rel=Stylesheet href="/css/pcn.css" media="screen and (min-width: 1366px)">';
            }else{
                echo '<link rel=Stylesheet href="/css/movil.css" media="screen and (max-width: 1365px)">';
                echo '<link rel=Stylesheet href="/css/pc.css" media="screen and (min-width: 1366px)">';}
        ?>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <script type="text/javascript" src="/js/jquery.js"></script>
        <script type="text/javascript" src="/js/codigo.js"></script>
    </head>
    <body>
        <div id="contenedor">
            <header>
                <?php
                    if (date('H')>19 or date('H')<7){
                        echo '<a href="/index.php"><img src=/img/blanco.png></a>';
                    }else{
                        echo '<a href="/index.php"><img src=/img/negro.png></a>';}
                    ?>
                <h2>Las mejores ofertas, en tus manos...</h2>
</h2>
            </header>
            <section>
            <div id="contienecarrito">
            <div id="carrito" style="background:rgb(200,200,200); color:black;"></div><center>
            <a href='/loginbiblioteca.php' id="botoncarrito"><button>Biblioteca</button></a>
            <a href='php/vaciacarrito.php' id="botoncarrito"><button>Vaciar carrito</button></a>
            <a href='/checkout.php' id="botoncarrito"><button>Confirmar compra</button></a>
            </center></div>
```

Nuestro archivo de cabecera es tan importante como nuestro index. Lo primero que hace es iniciar la sesión PHP y poner nuestro contador de productos a cero si todavía no está establecido. Después señalamos que es un documento de tipo html y en español, importante para validar el tipo de lenguaje de marca y para que el navegador sepa en qué lenguaje se compila. Empieza el encabezado del documento, ponemos el título a la pestaña y…

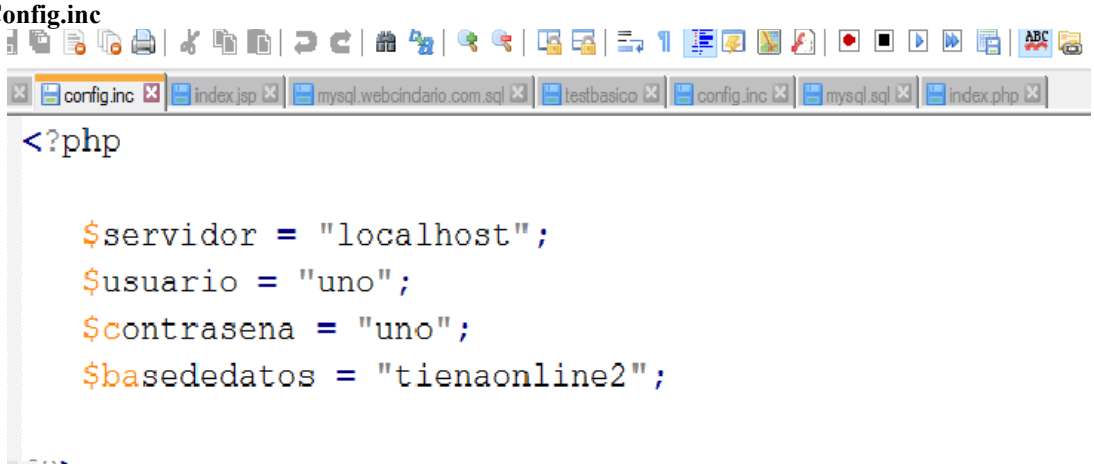
Entramos en el recuadro naranja, volvemos a php para establecer un if en el que distinguiremos mediante la hora del día si debe cargarse la versión diurna o la versión nocturna de la página, y dentro de cada respuesta, a partir de qué ancho de pantalla debe cargarse la versión para móviles.

Posteriormente establecemos el utf-8, y entramos en el recuadro **negro**, muy importante porque incluye el javascript que hace funcionar jquery y el javascript que hemos programado para trabajar en tiempo real con nuestro carrito (además de que se encarga de introducir nuestro poncarrito.php).

Terminado el encabezado del documento, empieza el cuerpo, señalamos como contenedor esta parte para nuestro CSS, y volvemos a PHP en el recuadro rojo, donde haremos un if idéntico al anterior pero que esta vez se encargará de cargar la versión diurna o nocturna de nuestro banner.

Para finalizar, en el recuadro azul indicamos al CSS que entramos en la zona del carrito, colocamos su color gris de fondo, el color negro de la letra y los botones del carrito con los enlaces a los archivos php correspondientes.

Config.inc



Sencillamente asignamos a variables los valores de conexión de la base de datos. Esto nos es útil a la hora de trabajar con versiones alojadas en distintos sitios, ya que si estos datos no se recogiesen automáticamente de un único documento, nos encontraríamos con que cada vez que pasamos de nuestro equipo de producción (localhost) al equipo de hosting, habría que reescribir los datos de conexión en todos los documentos cada vez que se migrase de un servidor a otro.

Codigo.js

```
$(document).ready(inicio)
function inicio(){
    $(".botoncompra").click(anade)
    $("#carrito").load("php/poncarrito.php");
}
function anade(){
    var idnumero = $(this).val();
    var cantidad = $("#num"+idnumero).val();
    $("#carrito").load("php/poncarrito.php?p="+$(this).val()+"&cant="+cantidad);
}
```

Como hemos indicado anteriormente, esta parte es el javascript que se encarga de actualizar nuestro carrito en tiempo real, de ahí la necesidad de usar javascript. Utilizamos la función document ready para empezar a trabajar cuando inicio esté preparada. Inicio es una función que detecta los clicks en el botón de añadir al carrito, lanzando a su vez la función anade (añade) y posteriormente recargando donde esté la etiqueta #carrito el archivo **poncarrito.php.**

'Anade' funciona de la siguiente manera. Consigue y almacena en la variable idnumero el valor de "this", es decir, el valor que devuelve lo que hemos clickado. En la variable de cantidad, introducimos para el producto de id "#num" idnumero el valor introducido en el campo numérico. Finalmente, carga en #carrito el poncarrito actualizado con el id de producto clicado y la cantidad que hemos conseguido con este procedimiento.

Poncarrito.php (Parte 1)

```php
<?php
    session_start();
    $suma = 0;
    $unidades = 0;

    $bandera = 0;
    $posicion = 0;

    $conexion=mysqli_connect($servidor,$user,$pass,$bdd);
    mysqli_set_charset($conexion,"utf8");

    for($i=0;$i<$_SESSION['contador'];$i++){
        if ( isset($_GET['p']) and isset($_SESSION['producto']) and $_GET['p']==$_SESSION['producto'][$i]){
            $bandera=1;
            $posicion=$i;
            $i=$_SESSION['contador'];
            }
        else {
            $bandera=0;
        }
    }

    if(isset($_GET['p']) and $bandera==0){
        $_SESSION['producto'][$_SESSION['contador']] = $_GET['p'];
        $_SESSION['cantidad'][$_SESSION['contador']] = $_GET['cant'];
        $_SESSION['contador']++;
        }

    if(isset($_GET['p']) and $bandera==1){
        $_SESSION['cantidad'][$posicion] = $_SESSION['cantidad'][$posicion]+$_GET['cant'];
        }

    $conexion=mysqli_connect($servidor,$user,$pass,$bdd);
    mysqli_set_charset($conexion,"utf8");

    echo "<table id='productos'>";
    echo "<tr id='productoscarrito'><td id='ctd1'>UD.</td><td id='ctd2'><center>PRODUCTO</center></td><td id='ctd3'>PRECIO</td></tr>";
```

Poncarrito es otra de las clases fundamentales para la web. Empezamos iniciando su sesión php y creando algunas variables que vamos a necesitar además de realizar la conexión a la base de datos y la selección del charset utf-8. Ahora empieza lo interesante.

En el recuadro rojo encontramos un for que se repite tantas veces como diga el contador, es decir, una vez por cada producto distinto que haya. Dentro de este for, se pregunta si están puestos los valores “p” (concedido a través de la URL) y la variable de sesión vectorial “producto”. Si están puestas, además comprobamos si son iguales. Si lo son, colocamos la bandera a “1” para indicar que son iguales, guardamos en $posicion la posición en la que hemos encontrado el producto (el valor que en ese momento tenga $i), y ponemos $i al valor de contador para terminar el bucle. En caso contrario, la bandera estará a cero. De este modo descubrimos si el producto que añadimos al carrito ya está en este o no, para saber si es necesario añadir la id a la lista o aumentar la cantidad que tenemos.

En el recuadro azul, tomamos medidas para ambos casos. Si está puesto “p” y la bandera está a cero, significa que el producto no está en el carrito, por lo que en la última posición del carrito (la indicada por el contador) añadimos el producto de id “p” con la cantidad deseada “cant” y aumentamos el contador. Si por el contrario la bandera nos indica que el producto ya está en el carrito, lo que hacemos es sumar al campo de cantidad en la $posicion que hemos conservado el valor “cant”.

Lo que resta de esta parte del código es la creación de la tabla de los productos en el carrito junto a etiquetas para cada columna dedicadas a la personalización mediante css. Vamos a la segunda parte.

Poncarrito.php (Parte 2)

```php
for($i = 0;$i< $_SESSION['contador'];$i++){
    $peticion="select * from productos where id=".$_SESSION['producto'][$i]."";
    $resultado=mysqli_query($conexion,$peticion);


    if($_SESSION['cantidad'][$i]<1){
        $_SESSION['cantidad'][$i]=1;
    }


    while($fila=mysqli_fetch_array($resultado)){
        if($_SESSION['cantidad'][$i]>$fila['existencias']){
            $_SESSION['cantidad'][$i]=$fila['existencias'];
        }
    }

    $peticion="select * from productos where id=".$_SESSION['producto'][$i]."";
    $resultado=mysqli_query($conexion,$peticion);

    while($fila=mysqli_fetch_array($resultado)){

        echo "<tr id='productoscarrito'><td id='td1'>".$_SESSION['cantidad'][$i]."</td><td id='td2'>".$fila['nombre']."</td><td
        id='td3'>".number_format(($_SESSION['cantidad'][$i]*$fila['precio']),2)."€</td></tr>";
        $suma += $_SESSION['cantidad'][$i]*$fila['precio'];
        $unidades += $_SESSION['cantidad'][$i];
    }
}

echo "</table>";
echo "<table>";
echo "<tr id='totalcarrito'><td></td><td>Unidades en carrito</td><td> ".$unidades."</td></tr>";
echo "<tr id='totalcarrito'><td></td><td>Subtotal</td><td> ".number_format($suma,2)."€</td></tr>";
echo "</table>";

mysqli_close($conexion);
?>
```

Este gran bucle for señalado con el recuadro **negro** realmente tiene una razón de ser bastante sencilla. En el recuadro rojo, lo que hace es recorrer los productos del carrito evitando que se introduzcan cantidades inferiores a 1 o superiores a las unidades disponibles (reflejadas en el campo “existencias”).

Una vez se ha comprobado esto, lo que resta (en el recuadro azul) es escribir en la tabla los datos de las unidades, el nombre, el precio total al multiplicar las unidades por el precio unitario (controlando posibles errores especificando el formato de dos decimales con number_format), y finalmente sumando las unidades y el precio de cada conjunto de copias de cada producto en variables que al terminar el bucle for tendrán el importe total del carrito y el número total de unidades en el mismo, siempre señalando en cada columna las etiquetas correspondientes para CSS.

Una vez hecho esto, cerramos la tabla y abrimos una nueva tabla de una sola columna en la que hemos puesto las unidades totales y el importe total conseguidos al final del bucle anterior.

Producto.php

```php
<?php include 'php/cabecera.inc' ?>
<?php include 'php/config.inc'
$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion="select * from productos inner join imagenes on
productos.id=imagenes.idproducto where productos.id=".$_GET['id']." and
imagenes.titulo='header.jpg'";
$resultado=mysqli_query($conexion,$peticion);
$fila=mysqli_fetch_array($resultado);

    echo "<article id='div'>";
    echo "<center><img id='imgproducto' src='".$fila['imagen']."'></center><br>";
    echo "<h3 id='titulo'>".$fila['nombre']."</h3><br>";
    echo "<p>".$fila['descripcion_larga']."</p>";

    $peticionimgs="select * from imagenes where idproducto=".$_GET['id']." and titulo
    !='header.jpg'";
    $resultadoimgs=mysqli_query($conexion,$peticionimgs);
    echo "<br><center>";
    while($filaimgs=mysqli_fetch_array($resultadoimgs)){
        echo "<a href='".$filaimgs['imagen']."' target='_blank'> <img id='galeria'
        src='".$filaimgs['imagen']."'> </a>";
    }
    echo "</center>";
    echo "<br>";
    echo "<br><h3>Genero: ".$fila['genero'].".</h3>";
    echo "<br><h3>Tamaño: ".$fila['tamaño']." GB.</h3>";
    echo "<br><h3>Plataforma: ".$fila['plataforma'].".</h3>";
    echo "<br><h3>Precio: ".$fila['precio']." €</h3>";
    echo "<br><center>";
    echo "<button value='".$fila['id']."' class='botoncompra'>Añadir a
    carrito</button></a>";
    echo "<input type='number' value='1' max='".$fila['existencias']."' min='1'
    id='num".$fila['id']."'> unidad/es.";
    echo "</center></article>";
mysqli_close($conexion);
?> <?php include 'php/piedepagina.inc' ?>
```

En la ficha del producto, lo único que tenemos que hacer es pedir los datos del producto cuya id hemos transferido en la URL y mostrarlos.

La primera petición consigue los datos del producto y la imagen de cabecera del mismo, que presidirá la página. Justo debajo, el nombre y la descripción larga, seguidos de una nueva consulta que conseguirá las demás imágenes del producto, con la excepción de la imagen de cabecera que ya hemos mostrado. De este modo, conseguiremos una galería de imágenes colocándolas una por una con un while.

Después de la galería, mostramos otros datos de interés como son el género del producto, su tamaño en gigas, la plataforma donde es canjeable la clave, el precio del producto en euros y un botón de compra junto a la caja de número de unidades deseadas.

Finalmente, cerramos la conexión e incluimos el pie de página.

Loginbiblioteca.php

```php
<?php include 'php/config.inc' ?>
<?php include 'php/cabecera.inc' ?>

<article>
<div id="loginbiblioteca">
    <br>
    Login de Biblioteca<br>

    <form action="biblioteca.php" method="POST">

        <input type="text" name="user" placeholder="Nombre de usuario">
        <input type="password" name="pass"placeholder="Contraseña">
        <input type='submit'>

    </form>

    <br>

</div></article>

<?php include 'php/piedepagina.inc' ?>
```

El login hacia la biblioteca no tiene más que un formulario, lo interesante está en:

Biblioteca.php (Parte 1)

```php
<?php include 'php/config.inc' ?>
<?php include 'php/cabecera.inc' ?>

<div id="loginbiblioteca"><center><table id="tablabiblio">
<?php
echo "<article>";
$contador=0;
$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion="select * from clientes where user = '".$_REQUEST['user']."' AND pass = '".$_REQUEST['pass']."'";
$resultado=mysqli_query($conexion,$peticion);
while($fila=mysqli_fetch_array($resultado)){
    $contador++;
}
if ($contador>0){
    $conexion=mysqli_connect($servidor,$user,$pass,$bdd);
    mysqli_set_charset($conexion,'utf8');
    $peticion = "select clave, productos.nombre from claves inner join productos on claves.idproducto =
    productos.id inner join clientes on vendida = clientes.id where vendida=(select id from clientes
    where user='".$_POST['user']."') order by claves.id";
    $resultado = mysqli_query($conexion, $peticion);

    echo'
        <tr ALIGN=center id="trbiblio">
        <td>JUEGO</td>
        <td>CLAVE</td>
        </tr>';
```

Esta parte simplemente hace una selección del usuario y la contraseña dados, y si devuelve positivo, el contador hace de bandera, por lo que entra en el if y selecciona de las tablas claves, productos y clientes donde el campo “vendida” sea equivalente a la id del cliente conseguida en la primera consulta. Antes de empezar la siguiente parte, muestro el echo que empieza la tabla mostrando los títulos de cada columna.

Biblioteca.php (Parte 2)

```
    while($fila = mysqli_fetch_array($resultado)) {

        echo '<tr';
         if($contador==1){
         echo " class='sombreado' ";
         }
        echo 'id="trbiblio">';

        echo '
        <td id="tdbiblio">'.$fila['nombre'].'</td>
        <td id="tdbiblio" align="center">'.$fila['clave'].'</td>
        </tr>';

        if($contador == 0){
            $contador++;
        }else{
            $contador--;
        }

    }

echo '</table></center></article></div>';

}else{
    echo "<br><h3>Login incorrecto. Volviendo a la pantalla de login.</h3>";
    echo '<meta http-equiv="refresh" content="5; url=loginbiblioteca.php">';

}

mysqli_close($conexion);

?>


<?php include 'php/piedepagina.inc' ?>
```

Bien, esta parte va sacando los resultados creándoles al tiempo nuevas filas en la tabla y asignándoles las etiquetas correspondientes de CSS además de utilizar el mismo contador para sombrear las columnas impares, simplemente por estética. Lo que vemos a partir del "else" es la respuesta que encontraremos en cualquier login de la web cuando escribamos mal el usuario o la contraseña, es decir, si la consulta del principio no encuentra ese combo de usuario y contraseña en la base de datos, devolverá un mensaje de error y recargará la página de login en cinco segundos.

Vaciacarrito.php

```
<?php

    session_start();
    session_destroy();

?>

<script>
    window.location = "../index.php";
</script>
```

Esta clase es muy simple, los datos de nuestro carrito se conservan mediante la sesión php, por lo que si queremos limpiar el carrito, basta con destruir esa sesión. Al final, un script que nos devuelve a nuestro index.

Checkout.php

```
<?php include 'php/cabecera.inc' ?>

<article id='div'>
    <br>
    ¿Ya eres usuario?<br>

    <form action="php/login.php" method="POST">

        <input type="text" name="user" placeholder="Nombre de usuario">
        <input type="password" name="pass"placeholder="Contraseña">
        <input type='submit'>

    </form>

    <br><br>

    ¿Eres un nuevo usuario?<br>
    <form action="php/nuevocliente.php" method="POST">
        <input type="text" name="nombre" placeholder="Nombre"><br>
        <input type="text" name="apellidos" placeholder="Apellidos"><br>
        <input type="text" name="email" placeholder="Email"><br>
        <input type="text" name="user" placeholder="Nombre de usuario"><br>
        <input type="password" name="pass" placeholder="Contraseña"><br>
        <input type="text" name="tlf" placeholder="Teléfono"><br>
        <input type="text" name="direccion" placeholder="Dirección"><br>
        <input type="text" name="zip_code" placeholder="Código postal"><br>
        <input type="text" name="poblacion" placeholder="Población"><br>
        <input type="text" name="pais" placeholder="País"><br>
        <input type="submit"">
    </form>

</article>

<?php include 'php/piedepagina.inc' ?>
```

Esta clase es bastante sencilla. El login es similar al de biblioteca pero además existe un segundo formulario que sirve para que los nuevos clientes se registren al comprar.

Nuevocliente.php

```php
<?php include 'config.inc' ?>
<?php
$contador=0;
$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion = "select * from clientes where user = '".$_POST['user']."'";
$resultado = mysqli_query($conexion,$peticion);
while($fila = mysqli_fetch_array($resultado)){
    $contador++;}
if ($contador == 0){
$peticion = "insert into clientes values (
NULL,
'".$_POST['nombre']."',
'".$_POST['apellidos']."',
'".$_POST['email']."',
'".$_POST['user']."',
'".$_POST['pass']."',
'".$_POST['tlf']."',
'".$_POST['direccion']."',
'".$_POST['zip_code']."',
'".$_POST['poblacion']."',
'".$_POST['pais']."',
'no')";
$resultado = mysqli_query($conexion, $peticion);
mysqli_close($conexion);
echo '
    <script>
        window.location="login.php?user='.$_POST['user'].'&pass='.$_POST['pass'].'"
    </script>';
}else{
echo '
    <script>
        window.location="../checkout.php"
    </script>
';
}
?>
```

Lo primero que hacemos en este php es consultar si existe ya un cliente con ese nombre de usuario. El resto de datos pueden ser idénticos, pero el nombre de usuario, indispensable para el login y los pagos, no puede repetirse en la base de datos. Igual que antes, usamos como bandera el contador, por lo que si está a cero, hacemos un insert con los datos que hemos traído de checkout.php y automáticamente redirigimos el usuario y la contraseña a login.php para que se pase directamente a PayPal sin que tenga que escribir de nuevo el usuario y la contraseña en el login. En caso contrario, volvemos a checkout.php.

Login.php (Parte 1)

```
sultado=mysqli_query($conexion,$peticion);
le($fila=mysqli_fetch_array($resultado)){
 $contador++;
 $_SESSION['usuario'] = $_REQUEST['user'];
 $_SESSION['contraseña'] = $_REQUEST['pass'];
 $_SESSION['idcliente'] = $fila['id'];


($contador>0){
 ?>
     <h2>Realizando el pedido...</h2>
     <form name='formTpv' method='post' action='https://www.sandbox.paypal.com/cgi-bin/webscr'>
     <input name="cmd" type="hidden" value="_cart">
     <input name="upload" type="hidden" value="0">
     <input name="business" type="hidden" value="tiendagetitcheap-facilitador@gmail.com">
     <input name="shopping_url" type="hidden" value="http://localhost/tiendaonline2/">
     <input name="currency_code" type="hidden" value="EUR">
     <input name="return" type="hidden" value="http://localhost/tiendaonline2/exitopaypal.php">
     <input type='hidden' name='cancel_return' value='http://localhost/tiendaonline2/errorPaypal.php'>
     <input name="notify_url" type="hidden" value="http://localhost/tiendaonline2/paypalipn.php">
```

La primera parte del login tomamos los datos del usuario requerido y guardamos los datos de usuario, contraseña y su id en variables de sesión para evitar que se pierdan durante el proceso con PayPal. Una vez entramos en el if (ya que se vuelve a comprobar user y pass), empieza el formulario que facilita PayPal para rellenar con los datos habituales. Simplemente le especificamos a dónde tiene que ir en caso de éxito o fracaso en la transacción mediante PayPal, y la cuenta del beneficiario.

Login.php (Parte 2)

```
            <?php
                for($i = 1;$i< $_SESSION['contador']+1;$i++){
                    $peticion="select * from productos where id='".$_SESSION['producto'][$i-1]."'";
                    $resultado=mysqli_query($conexion,$peticion);
                    while($fila=mysqli_fetch_array($resultado)){
                        $nombre=$fila['nombre'];
                        $precio=$fila['precio'];
                        $cantidad=$_SESSION['cantidad'][$i-1];
                        $id=$fila['id'];
                        ?>
                    <input name="item_name_<?php echo $i; ?>" type="hidden"  value="<?php echo $nombre; ?>">
                    <input name="amount_<?php echo $i; ?>" type="hidden" value="<?php echo $precio; ?>">
                    <input name="quantity_<?php echo $i; ?>" type="hidden"  value="<?php echo $cantidad; ?>">
                    <?php
                    }
                }
            ?>
            </form>

            <script type="text/javascript">
                document.formTpv.submit();
            </script>
        <?php

}else{
    echo "<br><h3>Login incorrecto. Volviendo a la pantalla de login.</h3>";
    echo '<meta http-equiv="refresh" content="5; url=../checkout.php">';

}

mysqli_close($conexion);

echo "</article>";

include 'piedepagina.inc';

?>
```

Aún dentro del if, entramos en php. Lo primero que haremos será iniciar un bucle for con contador a +1 (aunque podríamos haber dicho <= $_SESSION[‘contador’], pero así es más visible) para asegurarnos de que se revisa la última posición del carrito. De este modo en cada iteración hacemos un select de cada posición de la matriz de productos, y vamos guardando en variables nombre, precio, cantidad e id. Después, mediante esos tres inputs, se le pasan a PayPal los datos visibles a la hora de pagar para cada producto en cada iteración. Fuera del bucle, un script de subida del formulario cedido por PayPal, y finalmente el else para el login incorrecto y el cierre de conexión junto al pie de página.

Exitopaypal.php

```php
<?php include 'php/config.inc' ?>
<?php include 'php/cabecera.inc';

echo "<article id='div'>";
$contador=0;
$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion="select * from clientes where user = '".$_SESSION['usuario']."' AND pass = '".$_SESSION[
'contraseña']."'";
$resultado=mysqli_query($conexion,$peticion);

while($fila=mysqli_fetch_array($resultado)){
    $_SESSION['idcliente'] = $fila['id'];
}

$peticion="insert into pedidos values (null,".$_SESSION['idcliente'].",'".date('U')."','0')";
$resultado = mysqli_query($conexion,$peticion);

$peticion="select * from pedidos where idcliente= '".$_SESSION['idcliente']."' order by id desc limit 1";
$resultado = mysqli_query($conexion,$peticion);

while($fila=mysqli_fetch_array($resultado)){
    $_SESSION['idpedido'] = $fila['id'];
}
```

La primera posible respuesta de PayPal es el éxito en la transacción. En esta primera parte del código, utilizamos las variables de sesión para recuperar el id del cliente que nos ocupa en una nueva variable de sesión, e insertamos en la base de datos el nuevo pedido. Antes de continuar con el código, conseguimos la id de este pedido.

Exitopaypal.php (Parte 2)

```php
for($i = 0;$i< $_SESSION['contador'];$i++){
    $peticion="insert into lineaspedido values (null,'".$_SESSION['idpedido']."','".$_SESSION['producto'][$i
    ]."','".$_SESSION['cantidad'][$i]."')";
    $resultado=mysqli_query($conexion,$peticion);

    $peticion="select * from productos where id='".$_SESSION['producto'][$i]."'";
    $resultado=mysqli_query($conexion,$peticion);
    while($fila=mysqli_fetch_array($resultado)){
        $existencias=$fila['existencias'];
        $peticion2="update productos set existencias='".($existencias-$_SESSION['cantidad'][$i])."' where
        id='".$_SESSION['producto'][$i]."'";
        $resultado2 =mysqli_query($conexion,$peticion2);
    }

    for($x = 0;$x< $_SESSION['cantidad'][$i];$x++){
        $peticion="select * from claves where idproducto= '".$_SESSION['producto'][$i]."' and vendida=0
        order by id limit 1";
        $resultado=mysqli_query($conexion,$peticion);
        while($fila2=mysqli_fetch_array($resultado)){
            $peticion3="update claves set vendida='".$_SESSION['idcliente']."' where id='".$fila2['id']."'";
            $resultado3 =mysqli_query($conexion,$peticion3);
        }
    }
}

echo "<br>¡Pedido realizado!, podrá encontrar sus keys en la sección <n>Biblioteca</n>.";
echo "<br>Su pedido es el numero ".$_SESSION['idpedido'].".";
echo "<br>Redirigiendo a la tienda en 5 segundos...";

session_destroy();

echo '<meta http-equiv="refresh" content="5; url=../index.php">';
```

El código que resta es un bucle for que se encarga de introducir las líneas del pedido en la tabla lineaspedido. También recoge las existencias actuales del producto comprado y le resta las compradas para llevar el conteo de unidades disponibles, actualizando este valor (existencias) en la tabla de productos. Esto sucede, respectivamente, al principio del for y dentro del while.

El segundo for se encarga de tomar de una en una las claves disponibles que se van a asignar al comprador repitiendo tantas veces la asignación de una nueva clave como unidades se hayan comprado. El while dentro de este for se encarga de actualizar la clave conseguida en la consulta con la id del cliente. Finalmente, se devuelve el mensaje de éxito junto al número de pedido y se redirige al index en cinco segundos, no sin antes destruir la sesión para limpiar el carrito. Aunque en la captura no es visible, más abajo están las líneas habituales de cierre de mysqli y el include del pie de página.

Errorpaypal.php

```php
<?php include 'php/config.inc' ?>
<?php include 'php/cabecera.inc';

echo "<article id='div'>";
echo "<br>Su pedido no ha sido realizado";
echo "<br>Redirigiendo a la tienda en 5 segundos...";

session_destroy();

echo '<meta http-equiv="refresh" content="5; url=../index.php">';

mysqli_close($conexion);

echo "</article>";

include 'php/piedepagina.inc';

?>
```

El php que devuelve una compra no finalizada en PayPal es tan sencillo como cabría esperar, una redirección destruyendo la sesión.

/admin/index.php

```php
<?php include '../php/config.inc' ?>
<?php include '../php/cabecera.inc' ?>

<article>
<div id="loginadmin">
    <br>
    Login de Administradores<br>

    <form action="loginadmin.php" method="POST">

        <input type="text" name="user" placeholder="Nombre de usuario">
        <input type="password" name="pass"placeholder="Contraseña">
        <input type='submit'>

    </form>

    <br>

</div></article>

<?php include '../php/piedepagina.inc' ?>
```

Nuestro index de admin es un formulario de login normal y corriente, que contiene:

/admin/loginadmin.php

```php
<?php include '../php/config.inc' ?>
<?php include '../php/cabecera.inc';

echo '<div id="loginadmin">';
$contador=0;
$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion="select * from clientes where user = '".$_REQUEST['user']."' AND pass = '".$_REQUEST['pass']."'AND
admin = 'yes'";
$resultado=mysqli_query($conexion,$peticion);
while($fila=mysqli_fetch_array($resultado)){
    $contador++;
}
if ($contador>0){
        echo '<meta http-equiv="refresh" content="0; url=pedidos.php">';
    }else{
    echo "<br><h3>No eres un usuario con las credenciales suficientes. Volviendo a la pagina principal.</h3>";
    echo '<meta http-equiv="refresh" content="5; url=../index.php">';
    }
echo '</div>';
mysqli_close($conexion);
?>
<?php include '../php/piedepagina.inc' ?>
```

Pero loginadmin no sólo comprueba el usuario y la contraseña sino también que el usuario esté registrado en la base de datos como administrador.

/admin/cabecera.inc

cabecera.inc | actualizarproducto.php | Ejercicio_3_solucion_sort_2.html | ejercicios alumno | EXAMEN3EVA2.html | Ejercicio_3.html | index.php | db

```php
<?php
session_start();
if(!isset($_SESSION['contador'])){$_SESSION['contador'] = 0;}
?>
<!doctype html>
<html lang="es">
  <head>
    <title>Tienda AgroCoop2</title>
    <link rel=Stylesheet href="estiloadmin.css">
    <meta http-equiv="content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
     <div id="contenedor">
        <header>
           <a href="index.php"><h1>Panel de control</h1></a>
           <a href="../index.php"><h3>Tienda online</h3></a>
    <a href='pedidos.php'class="botonmenu">Gestionar pedidos</a>
    <a href='clientes.php' class="botonmenu">Gestionar clientes</a>
    <a href='productos.php' class="botonmenu">Gestionar productos</a>
```

Antes de empezar con las páginas de admin, os enseñamos el código de la cabecera de las páginas de administración. Básicamente tiene su propio CSS para embellecer las tablas y, principalmente, tiene un menú en forma de botones para desplazarnos entre las distintas páginas de administración cómodamente.

/admin/pedidos.php

```php
<?php
$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion="select pedidos.id as idpedido,fecha,estado,nombre,apellidos from pedidos left join clientes on
pedidos.idcliente=clientes.id order by estado,fecha asc";
$resultado=mysqli_query($conexion,$peticion);
echo'   <tr ALIGN=center>
    <td>CLIENTE</td>
    <td>FECHA - HORA</td>
    <td>ESTADO</td>
    </tr>';
while($fila=mysqli_fetch_array($resultado)){
    $estado=$fila['estado'];
    echo '<tr';
     if ($estado == 0){
        echo ' style="background:rgb(250,255,200);"';
        $diestado="En proceso";
    }
    if ($estado == 1){
        echo ' style="background:rgb(200,255,200);"';
        $diestado="Listo";
    }
    if ($estado == 2){
        echo ' style="background:rgb(255,200,200);"';
        $diestado="Cancelado";
    }
    echo '>
    <td>'.$fila['nombre'].'</td>
    <td>'.date("d/M/Y - H:i:s",$fila['fecha']).'</td>
    <td>'.$diestado.'</td>
    <td><a href="../php/pedidoservido.php?id='.$fila['idpedido'].'"><button>Pedido servido</button></a></td>
    <td><a href="../php/enproceso.php?id='.$fila['idpedido'].'"><button>En proceso</button></a></td>
    <td><a href="../php/cancelarpedido.php?id='.$fila['idpedido'].'"><button>Cancelar
    pedido</button></a></td></tr>';
}
mysqli_close($conexion);
?>
```

En esta captura obviamos los includes al inicio y al cierre, etc. **Pedidos.php** es la página por defecto en el acceso a la administración. Lo primero que hacemos es conseguir los pedidos junto a los datos del cliente correspondiente, después llega el echo en el que ponemos nombre a cada columna de la tabla en la que mostramos los pedidos.

En el while, lo que hacemos es diferenciar el estado de cada pedido por el campo “estado”, con correspondencia en un código numérico. De este modo, si el código es 0, se mostrará como un pedido en proceso y se coloreará la fila en amarillo. Si está a 1, es un pedido listo y se coloreará en verde. Si está a 2, es un pedido cancelado y se coloreará en rojo.
Después de crear la fila con el color correspondiente, se crean las distintas columnas con los datos obtenidos de la consulta y se colocan botones para cambiar el estado de cada pedido.

/admin/[pedidoservido.php, enproceso.php y cancelarpedido.php]

```php
<?php include "config.inc" ?>
<?php

$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion=" update pedidos set estado = 1 where id = '".$_GET['id']."'";
$resultado=mysqli_query($conexion,$peticion);

mysqli_close($conexion);
?>
<script>
    window.location = "../admin/pedidos.php";
</script>
```

Este es pedidoservido, el php que lanzamos para actualizar el estado del pedido a servido.

```php
<?php include "config.inc" ?>
<?php

$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion=" update pedidos set estado = 0 where id = '".$_GET['id']."'";
$resultado=mysqli_query($conexion,$peticion);

mysqli_close($conexion);
?>
<script>
    window.location = "../admin/pedidos.php";
</script>
```

Este es **enproceso**:

```php
<?php include "config.inc" ?>
<?php

$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion=" update pedidos set estado = 2 where id = '".$_GET['id']."'";
$resultado=mysqli_query($conexion,$peticion);

mysqli_close($conexion);
?>
<script>
    window.location = "../admin/pedidos.php";
</script>
```

Y **cancelarpedido**. Los tres son muy similares, tan sólo cambia el código que se corresponde en el código fuente con los distintos estados de pedido.

admin/clientes.php (Parte 1)

```php
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<?php include '../php/config.inc' ?>
<?php include 'cabecera.inc' ?>
<center><table>
<?php
$contador=0;
$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,'utf8');
$peticion = "SELECT * from clientes";
$resultado = mysqli_query($conexion, $peticion);
echo'
    <tr ALIGN=center>
    <td>NOMBRE</td>
    <td>APELLIDOS</td>
    <td>EMAIL</td>
    <td>USER</td>
    <td>PASS</td>
    <td>TLF</td>
    <td>DIRECCIÓN</td>
    <td>ZIP_CODE</td>
    <td>POBLACIÓN</td>
    <td>PAÍS</td>
    <td>¿ADMIN?</td>
    </tr>';
```

Esta es la primera parte del código de la página de gestión de clientes, básicamente consigue todos los datos y pone título a las columnas de la tabla de datos.

/admin/clientes.php (Parte 2)

```
while($fila = mysqli_fetch_array($resultado)) {
    echo '<tr';
     if($contador==1){
     echo " class='sombreado' ";
     }
    echo '>';
    echo'
    <form action="actualizarcliente.php?id='.$fila['id'].'" method="POST">
    <td><input type="text" name="nombre" value="'.$fila['nombre'].'"></td>
    <td><input type="text" name="apellidos" value="'.$fila['apellidos'].'"></td>
    <td><input type="text" name="email" value="'.$fila['email'].'"></td>
    <td><input type="text" name="user" value="'.$fila['user'].'"></td>
    <td><input type="text" name="pass" value="'.$fila['pass'].'"></td>
    <td><input type="text" name="tlf" value="'.$fila['tlf'].'"></td>
    <td><input type="text" name="direccion" value="'.$fila['direccion'].'"></td>
    <td><input type="text" name="zip_code" value="'.$fila['zip_code'].'"></td>
    <td><input type="text" name="poblacion" value="'.$fila['poblacion'].'"></td>
    <td><input type="text" name="pais" value="'.$fila['pais'].'"></td>
    <td><input type="text" name="admin" value="'.$fila['admin'].'"></td>
    <td><input class="boton" type="submit" value="Actualizar cliente"></td>
    </form>
    <td><a href="eliminarcliente.php?id='.$fila['id'].'"><button>Eliminar cliente</button></a></td>

    </tr>';

    if($contador == 0){
        $contador++;
    }else{
        $contador--;
    }
}
```

Esta es la segunda parte, en la que un bucle while va colocando fila tras filas los datos de uno y otro cliente en cajas editables, con un botón de actualizar y otro de eliminar, además de la bandera “contador” para sombrear las filas impares.

```
mysqli_close($conexion);
?>
    <tr>
        <form action="nuevocliente.php" method="POST">
            <td><input type="text" name="nombre"></td>
            <td><input type="text" name="apellidos"></td>
            <td><input type="text" name="email"></td>
            <td><input type="text" name="user"></td>
            <td><input type="text" name="pass"></td>
            <td><input type="text" name="tlf"></td>
            <td><input type="text" name="direccion"></td>
            <td><input type="text" name="zip_code"></td>
            <td><input type="text" name="poblacion"></td>
            <td><input type="text" name="pais"></td>
            <td><input type="text" name="admin"></td>
            <td><input type="submit" class="boton"></td>
            <td></td>
        </form>
    </tr>
</table></center>


<?php include 'piedepagina.inc' ?>
```

Y para acabar, una última fila-formulario para introducir clientes nuevos.

/admin/actualizarcliente.php

```php
<?php include '../php/config.inc' ?>
<?php include 'cabecera.inc' ?>
<?php
$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion="update clientes set
nombre='".$_POST['nombre']."',
apellidos='".$_POST['apellidos']."',
email='".$_POST['email']."',
user='".$_POST['user']."',
pass='".$_POST['pass']."',
tlf='".$_POST['tlf']."',
direccion='".$_POST['direccion']."',
zip_code='".$_POST['zip_code']."',
poblacion='".$_POST['poblacion']."',
pais='".$_POST['pais']."',
admin='".$_POST['admin']."'
where id=".$_GET['id']."";

$resultado=mysqli_query($conexion,$peticion);
mysqli_close($conexion);
?>

<script type="text/javascript">
    window.location="clientes.php";
</script>

<?php include 'piedepagina.inc' ?>
```

Este código, lanzado con el botón correspondiente, simplemente actualiza el cliente de la fila editada y nos devuelve a clientes.php

/admin/eliminarcliente.php

```php
<?php include '../php/config.inc' ?>
<?php include 'cabecera.inc' ?>
<?php
$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion="delete from clientes where id=".$_GET['id']."";
$resultado=mysqli_query($conexion,$peticion);

mysqli_close($conexion);
?>

<script type="text/javascript">
window.location="clientes.php";
</script>

<?php include 'piedepagina.inc' ?>
```

Con un funcionamiento similar a **actualizarcliente**, elimina el cliente seleccionado.

/admin/nuevocliente.php

```php
<?php include '../php/config.inc' ?>
<?php include 'cabecera.inc' ?>

<?php
$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,'utf8');
$peticion = "insert into clientes values (
NULL,
'".$_POST['nombre']."',
'".$_POST['apellidos']."',
'".$_POST['email']."',
'".$_POST['user']."',
'".$_POST['pass']."',
'".$_POST['tlf']."',
'".$_POST['direccion']."',
'".$_POST['zip_code']."',
'".$_POST['poblacion']."',
'".$_POST['pais']."',
'".$_POST['admin']."')";
$resultado = mysqli_query($conexion, $peticion);
mysqli_close($conexion);

echo $peticion;
?>
<script type="text/javascript">
    window.location="clientes.php";
</script>

<?php include 'piedepagina.inc' ?>
```

Nuevocliente tiene un código similar al de actualizarcliente, básicamente porque, del mismo modo, requiere que le pasemos todos los campos a introducir en la base de datos, con la única diferencia de que esto es una inserción y actualizarcliente una actualización sobre el cliente de id conocida.

/admin/productos.php

```php
<tr>
    <form action="nuevoproducto.php" method="POST" enctype="multipart/form-data">
        <td><input type="text" name="nombre"></td>
        <td><input type="text" name="genero"></td>
        <td><input type="text" name="descripcion"></td>
        <td><input type="text" name="descripcion_larga"></td>
        <td><input type="text" name="precio"></td>
        <td><input type="text" name="tamaño"></td>
        <td><input type="text" name="plataforma"></td>
        <td><input type="text" name="existencias"></td>
        <td><input type="text" name="activado"></td>
        <td><input type="file" name="imagen"></td>
        <td><input type="submit"></td>
    </form>
</tr>
```

Este no es todo el código de Productos. Las distintas páginas de administración y la forma en la que presentan la información son idénticas, por lo que me ceñiré a los puntos que tienen cambios a destacar. En este caso, el método sigue siendo “POST”, pero se sitúa un “enctype=”multipart/form-data” para poder agregar imágenes, junto al campo de imagen con tipo “file”. De este modo, vamos a obviar actualizarproducto y eliminarproducto al funcionar exactamente igual que para los clientes, pero hay que explicar el funcionamiento de nuevoproducto.

/admin/nuevoproducto.php

```
$peticion = "select * from productos order by id desc limit 1";
$resultado = mysqli_query($conexion, $peticion);
    while($fila = mysqli_fetch_array($resultado)) {
        $id = $fila['id'];
        $juego = $fila['nombre'];
        $carpeta = 'photo/'.$juego;
    }
    if (!file_exists($carpeta)) {
    mkdir('../'.$carpeta, 0777, true);
}

move_uploaded_file($_FILES['imagen']['tmp_name'],'../'.$carpeta.'/'.$_FILES['imagen']['name']);
$ruta = $carpeta.'/'.$_FILES['imagen']['name'];
$peticion = "insert into imagenes values (
NULL,
'$id',
'$ruta',
'".$_FILES['imagen']['name']."'
 )";
```

Bien, esta es la parte interesante del código. Después de haber ingresado los datos del producto, se consultan los datos del producto con la última id, es decir, el que se acaba de añadir, conservamos la id, el nombre y la carpeta para las fotos del producto, no sin crear después la carpeta del producto si no existe ya..

Ahora, movemos el archivo subido del lugar de subida con su nombre temporal a la carpeta con el nombre del producto con el nombre propio de la imagen. Esta ruta la conservamos en una variable para luego insertar en la tabla de imágenes la id del producto de la imagen, la ruta del archivo y el título de la imagen.

/admin/claves.php

```
$peticion = "SELECT * from productos order by nombre";
$resultado = mysqli_query($conexion, $peticion);

    echo '<tr>
        <form action="nuevaclave.php" method="POST" enctype="multipart/form-data">
            <td>';
                echo "<select name='idproducto'>";
                echo "<option value='-1'>Seleccione el producto</option>";
                while ($fila = mysqli_fetch_array($resultado)){
                   echo "<option value=".$fila['id'].">".$fila['nombre']."</option>";
                }
                echo '</select>
            </td>
            <td><input type="text" name="clave"></td>
            <td><input type="text" name="vendida" value="0"></td>
            <td><input type="submit"></td>
        </form>
    </tr>
</table></center>';
```

Esta es la parte que varía para claves. Básicamente, a la hora de introducir una clave nueva, consultamos los productos que tenemos disponibles, y los mostramos en una combobox. De este modo, seleccionamos directamente el producto para el que queremos introducir una nueva key. Con “<select>” es con lo que abrimos y cerramos el selector, y los campos “<option>” con las opciones disponibles.

/admin/nuevaclave.php

```
$peticion3="select * from productos where id=".$_POST['idproducto']."";
$resultado3=mysqli_query($conexion, $peticion3);

$fila3=mysqli_fetch_array($resultado3);

$peticion2="update productos set existencias='".$fila3['existencias']."'+1 where id=".$_POST['idproducto']."";
$resultado2 = mysqli_query($conexion, $peticion2);
```

Este es el código especial de “nuevaclave”, puesto que conseguimos los datos del producto del que hemos introducido una clave, y con estos datos podemos actualizarlo en su tabla correspondiente añadiendo una unidad a la columna correspondiente. El código de eliminar y actualizar clave es idéntico al de las tablas anteriores, así que no es necesario comentarlo.

/admin/estadisticas.php (Parte Productos)

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<?php include '../php/config.inc' ?>
<?php include 'cabecera.inc' ?>

<?php
$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");

//PRODUCTOS


echo "<h2>Top ventas:</h2>";
echo "<table><ol type='1'>";
$peticion = "select idproducto, productos.nombre, count(idproducto), sum(unidades) from `lineaspedido` left join
productos on lineaspedido.idproducto = productos.id group by idproducto order by sum(unidades) desc limit 3";

$resultado=mysqli_query($conexion,$peticion);

while($fila = mysqli_fetch_array($resultado)) {
    echo "<tr><td><li>".$fila['nombre']."</td><td> ==> </td><td>".$fila['sum(unidades)']."</td></tr>";
}
echo "</li></ol></table>";

echo "</br>";


$peticion = "select idproducto, productos.nombre, count(idproducto), sum(unidades) from `lineaspedido` left join
productos on lineaspedido.idproducto = productos.id group by idproducto order by sum(unidades) desc limit 1";
$resultado = mysqli_query($conexion, $peticion);

while($fila = mysqli_fetch_array($resultado)) {
    echo "<UL type = circle><LI>El juego más comprado es: ' ".$fila['nombre']." ' con ".$fila['sum(unidades)']."
    copias vendidas.</UL>";
}

echo "</br></br></br>";
```

La página de estadísticas tiene, de momento, una parte de productos, y una parte de clientes. La parte de productos simplemente utiliza las tablas lineaspedido y productos para contar las ventas de los productos y mostrar los tres más vendidos de forma ordenada. La segunda consulta es idéntica pero muestra el producto más vendido explicando esto en una línea.

/admin/estadisticas.php (Parte Clientes)

```php
//CLIENTES

echo "<h2>Top clientes:</h2>";
echo "<table><ol type='1'>";
$peticion = "select clientes.nombre, clientes.apellidos, sum(unidades*precio) from `pedidos` left join lineaspedido
on pedidos.id = lineaspedido.idpedido left join productos on lineaspedido.idproducto = productos.id left join
clientes on pedidos.idcliente = clientes.id group by idcliente order by sum(unidades*precio) desc limit 3";

$resultado=mysqli_query($conexion,$peticion);

while($fila = mysqli_fetch_array($resultado)) {
    echo "<tr><td><li>".$fila['nombre']."</td><td> ==> </td><td>".$fila['sum(unidades*precio)']."€</td></tr>";
}
echo "</li></ol></table>";

echo "</br>";

$peticion = "select clientes.nombre, clientes.apellidos, sum(unidades*precio) from `pedidos` left join lineaspedido
on pedidos.id = lineaspedido.idpedido left join productos on lineaspedido.idproducto = productos.id left join
clientes on pedidos.idcliente = clientes.id group by idcliente order by sum(unidades*precio) desc limit 1";

$resultado = mysqli_query($conexion,$peticion);

while($fila = mysqli_fetch_array ($resultado)){
    echo "<UL type = circle><LI>Nuestro mejor cliente es ' ".$fila['nombre']." ".$fila['apellidos']." ' gastando ".
    $fila['sum(unidades*precio)']."€";
    echo ".</UL>";
}

echo "</br></br></br>";

mysqli_close($conexion);
?>


<?php include 'piedepagina.inc' ?>
```

La parte de clientes es muy similar a la de productos. Utilizamos las tablas pedidos, productos y clientes para descubrir el importe total de compras que han realizado los clientes y devolver los tres clientes que más dinero han gastado. También mostramos en una línea el cliente que más ha gastado.

/admin/log.php

```php
<?php include 'config.inc' ?>

<?php

error_reporting(0);

$conexion=mysqli_connect($servidor,$user,$pass,$bdd);
mysqli_set_charset($conexion,"utf8");
$peticion = "insert into registros values (
'".date('U')."',
'".date('Y')."',
'".date('m')."',
'".date('d')."',
'".date('H')."',
'".date('i')."',
'".date('s')."',
'".$_SERVER['REMOTE_ADDR']."',
'".$_SERVER['HTTP_USER_AGENT']."',
'".$_SERVER['REQUEST_URI']."')";
$resultado = mysqli_query($conexion, $peticion);
mysqli_close($conexion);
?>
```

Este archivo está incluido en el pie de página. El archivo de pie de página no ha sido comentado puesto que apenas muestra nuestros nombres e incluye este código. Lo que conseguimos con este archivo es que cada vez que el usuario se desplace por la web, se inserte en una tabla registros información sobre la navegación. Fecha, hora, dirección, navegador y URL.

5 – IMPLANTACIÓN Y MANTENIMIENTO

5.1 - Formación

Crearemos cursos de formación para los usuarios de la empresa, y para los directivos, para que todos en la empresa de destino tengan la suficiente destreza para sacar un optimo rendimiento de la aplicación. Elaboraremos videotutoriales y presentaciones adaptadas al rol de cada usuario.
Se estima un curso de 10 horas para los usuarios de la aplicación.
Se estima un curso de 10 horas para los directivos de la empresa.

5.2 - Implantación del sistema y pruebas

Como comentamos en la fase de estudio de nuestro sistema, no necesitamos más que el servidor elegido y los mismos programas con los que hemos gestionado en el desarrollo. No obstante hay que cargar o importar la base de datos en el servidor y la carga de los datos de la base de datos hasta ahora de forma local, todo esto está explicado en los documentos de instalación en los anexos.
El sistema de pruebas para la página web es el que implementamos en Selenium, de modo que la suites se pasa al completo cuando se hace algún cambio en el código fuente para comprobar que todo sigue funcionando correctamente y no aparece ningún mensaje de error no deseado.

Se han elaborado pruebas **de carga,** y **límites de valores** del sistema, así como **pruebas caja negra**(salidas esperadas ante entradas controladas) **y caja blanca** (eficiencia de código y algoritmia, rendimiento y optimización de las consultas a la base de datos,...)

5.3 - Nivel de servicios

Se han completado todos los requisitos planteados en la tienda,así como los nuevos requisitos detectados en el desarrollo y en el plan de Sprints. Por lo tanto el nivel de servicios que se ha logrado es máximo. Aún así se ha hecho propuestas de mejora.
Los requisitos planteados se han cumplido, y el nivel de Servicios alcanzado ha sido el máximo posible. Se han alcanzado todos los requisitos planteados y los objetivos que se marcaron.

5.4 - Aceptación del sistema

La aceptación del sistema, se prepara en el supuesto caso que está aplicación se aceptada por alguna organización. Dado que es una aplicación desarrollada con software libre y licienciada con licencia libre, no sería necesario esta fase del proyecto. Pues cualquiera podría descargarlo instalarlo, usarlo y modificarlo sin necesidad de aceptar el sistema.
No obstante haemos un contrato de aceptación con la organización interesada que incluiría la implantación y el mantenimiento de la aplicación y hardware involucrado por un tiempo pactado. Este sería el aprovechamiento económico de la aplicación con licencia libre, la instalación y el mantenimiento, recordamos que una aplicación libre no está obligatoria mente reñida con un aprovechamiento pecuniario.
El posible contrato de aceptación incluiría la formación, videotutoriales, cursos explicando las funcionalidades.
El modo en el que hemos decidido hacer funcionar todo el sistema para que sea lo más efectivo posible en rendimiento y económicamente en lo que es la venta de productos.
Puesta en producción la aplicación y visto, probado el funcionamiento se sugiere firmar el contrato de aceptación.
Una vez implantada la aplicación, hecho las pruebas de aceptación y de implantación se firma un contrato de aceptación del proyecto y mantenimiento por un periodo acordado.

Tal y como hemos explicado en la fase de implantación, el mantenimiento y posibles ideas de mejora están en constante revisión.
Se ofertará a la empresas interesadas un servicio de mantenimiento de 2 ó 4 años.

7– RESUMEN y CONCLUSIONES

Estamos muy satisfechos con el resultado final. Lo que en un principio era un proyecto de comercio web con uso de php, sql, html y css, ha terminado siendo toda una tienda online con mucho trabajo en todos estos lenguajes de programación, con muchas mejoras implementadas y en definitiva un resultado perfectamente funcional.
Este proyecto nació detectando y dando solución a las propuestas de mejora de experiencias previas del autor. Tiene también más mejoras posibles, pero nos hemos ceñido a los requisitos planteados, y el nivel de Servicios alcanzado. Se han alcanzado todos los requisitos planteados y los objetivos que se marcaron.
Como uno de los requisitos era tener una gestión de compra, es decir un carrito de compra, aunque lo hemos optimizado, revisión tras revisión, pero siempre sin alejarnos de los requisitos que se habían planteado, sin hacer más pero tampoco menos, más bien optimizando. Así por ejemplo,que se pudiese añadir productos al carrito de manera lineal (si añades un producto "X" al carrito, pero si lo vuelves a añadir, **sumara unidades**, pero se pondrá en la posición que ocupaba ese mismo producto comprado anteriormente. Es decir, un carrito funcional). Controlar que el número de unidades añadidas al carrito se sumen si el producto ha sido añadido con anterioridad.

Hacer que el carrito se despliegue al pasar por encima de él con el ratón, para evitar que se monte encima de la descripción de los productos.

Controlar el número de unidades en la BBDD a la hora de mostrar al cliente, cuando tienen que descontarse en la base de datos y siempre en cooperación entre la tabla de productos con su campo de unidades y la tabla de claves. Es decir si no hay stock no se muestra.

Implementar pagos vía PayPal, con la utilidad de **PayPal Sandbox,** siempre controlando la integridad de las transacciones en nuestra base de datos.

Modificar la BBDD de tal manera que la descripción que se da en la página principal del producto, es diferente la de la sección "más información".

Como se ha comentado se ha implementado todas las funcionalidades planteadas en análisis de requisitos, y algunas de ellas se han optimizado y revisado. Aun así consideramos que si este proyecto sigue vivo, habría una serie de propuestas de mejora que se podrían implementar. Algunas de ellas son importantes, y otras simplemente mejoras, pero a modo de ejemplo proponemos algunas.

Crear varias versiones móviles y otra nocturnas de la página mediante la edición del código CSS, de modo que los elementos toman dimensiones más cómodas en smartphones o tabletas y colores más oscuros durante las horas de noche para hacer más cómoda su visualización.

Mejorar la parte online para administradores, mejorando la página de estadísticas,registrar idiomas de usuario, países y lenguajes de los compradores, sistemas operativos,etc

Implementar una biblioteca para que los usuarios tengan un acceso privado a los productos que han comprado.

Subir la página y la base de datos a un servidor hostin ponerlo al 100% de funcionalidad.

Realizar más estadísticas de compra, para con ello retroalimentar la propia página, p.e.: productos más vendidos, (actualizar precio), mejores compradores(mejores precios), crear rankings de productos, incluir redes sociales, e incluir comentarios de usuarios...

<u>Como valoración personal</u> me he sentido muy reconfortado con este proyecto, pues si bien conocía en más o menos profundidad los distintos lenguajes de programación y tecnologías usadas, he podido profundizar en ellos, afianzarme, y darle una utilidad practica a todo ello. Me ha servido para investigar como realizar una acción concreta necesaria, y lo más importante tener una visión global de un proyecto completo y cómo gestionarlo.
Lo más complicado por motivos ajenos, ha sido realizar una memoria técnica acorde con el proyecto, pero con la ayuda del consultor, creo sinceramente que se ha hecho un trabajo más que digno.
Hemos usado metodología clásica en la memoria y la planificación del proyecto, y también metodologías de gestión de proyectos más actuales, metodologías de desarrollo rápido, usando lo más adecuado en nuestro proyecto de cada una de ellas. Con esto he repasado conceptos ya aprendidos, y además investigar nuevas metodologías, nuevas herramientas, nuevas técnicas.

8 – BIBLIOGRAFÍA

- *stackoverflow.com*
- *librosweb.es*
- *php.net*
- *comocreartuweb.com*
- *forosdelweb.com*
- *homeandlearn.co.uk*
- *colordeu.es*
- *jhosuepardo.com*
- *html.hazunaweb.com*
- *hosting.miarroba.com*
- *1and1.es*
- *hostinger.es*
- *ovh.es*
- *aprenderaprogramar.es*
- *tutorialhtml.net*
- *https://www.apachefriends.org/es/index.html*

9– ANEXOS (aportados como adjuntos a esta memoria)

Presentación en vídeo
Presentación en diapositivas
Suites y tutorial de Selenium
Página web completa junto con la BBDD y README.txt
Documentación del desarrollo del código comentado paso a paso.
Varios vídeos explicativos tipo Demo.
Otros proyectos relacionados o adaptaciones del mismo.