

Índice de contenido

Modificaciones realizadas en este documento.....	1
Creación 8/1/2015.....	1
Normas de obligado cumplimiento.....	1
Comentarios.....	2
Calendario.....	3
1. Aplicación tienda on-line (8 Ptos).....	3
Puntuación desglosada por apartados que hay que realizar:.....	7
2. Servicios web (2 ptos).....	8
Agregador de tiendas.....	8
API del proveedor de servicios que debemos implementar para el Agregador.....	8
3. Aplicación tienda on-line – Mejoras adicionales.....	9
Creación de librería carrito de la compra (1 pto).....	9
Importación y exportación en XML (2 ptos).....	9
Soporte para múltiples monedas en vuestra aplicación (1 pto).....	9
Importación de datos en formato de hoja de calculo excel (2 ptos).....	9
Utilización de Ajax con JSON o XML (1 ptos).....	10
Comentarios y evaluación.....	10
Puntuación ejercicios.....	10
Criterios de evaluación.....	10
Instrucciones para entregar la práctica.....	10

Modificaciones realizadas en este documento

Creación 8/1/2015

- Creación del documento. (Incompleto)

Normas de obligado cumplimiento

Para la realización de la aplicación será obligatorio el uso algún framework de los disponibles para php. Preferentemente se utilizará CodeIgniter, aunque si el alumno tiene conocimientos de otros, podrá utilizarlos, aunque en este caso no podrá contar con la colaboración del profesor.

El acceso a la base de datos se realizará utilizando los mecanismos de acceso a datos de los que disponga el framework.

Los problemas deberán ser documentados con comentarios, indicando en cada fichero con código php el autor, la fecha de creación, la versión del fichero.

Se documentará cada una de las funciones y variables de clase que se creen utilizando el formato [PHPDoc](http://es.wikipedia.org/wiki/PHPDoc) (<http://es.wikipedia.org/wiki/PHPDoc>) del cual podréis obtener más información en [este artículo](#).

Básicamente lo que debéis hacer es abrir comentarios de tipo bloque `/** . . . */` delante de

cada función y variable de clase, y el entorno se encargará de generarlos automáticamente las [etiquetas](#) (@ . . .) pertinentes











Se generará la documentación de nuestro programa usando alguno de los programas existentes de generación automática como: [ApiGen](#), [Doxygen](#), o [phpDocumentor](#)

Se verificará el funcionamiento de la aplicación en el servidor Gerión. Para ello se os ha proporcionado un espacio web y una base de datos con la que podréis trabajar.

Comentarios

Cada problema de los expuestos a continuación se creará en una carpeta independiente, que llamaréis “Problema 1”, “Problema 2”, etc.

Cada carpeta que contiene los problemas tendrá la siguiente estructura:

Tipo	Elemento	Descripción
	¿?	Carpetas que utiliza el framework
	install	En esta carpeta se incluirá cualquier información que se considere relevante para instalar la aplicación. Sería interesante que incluyérais un instalador. Aplicación que cree la base de datos y configure el resto de parámetros necesarios.
	bd.sql	Fichero que contiene un script sql que crea la base de datos que utilizará el programa. En el script se creará <ul style="list-style-type: none">- La base de datos- La estructura de las tablas- Los usuarios si utilizase alguno diferente
	Doc	Documentación de la aplicación.
	Assets <ul style="list-style-type: none">cssimgjs	Imágenes, ficheros de estilo (css), fichero de script (js), y otro tipo de ficheros.
	Vendor	Si utilizáis composer aquí se descargarán los paquetes utilizados
	¿?	Otras carpetas que sean necesarias. Si utilizáis bower creará una carpeta propia en la que se descargan sus dependencias

Los problemas tienen como propósito obligaros a que trabajéis los contenidos vistos así como que profundicéis en vuestros conocimientos sobre la metodología de la programación (como resolver los problemas). Esta práctica está pensada para que la realicéis en clase completando el trabajo en vuestra casa. Ante cualquier duda acerca de cómo afrontar un problema deberíais preguntar al profesor al respecto para que el os oriente.

Cada problema tendrá una fecha de entrega, fecha en la cual deberéis tener completado el ejercicio, esta fecha es orientativa y tiene por objeto evitar que os durmáis en los laureles, tan solo es obligatorio tenerlo todo completo en la fecha de entrega.

Los problemas, **obligatoriamente deberéis enseñárselos al profesor funcionando en clase antes de la fecha tope de entrega.**

Calendario

lunes 8 de febrero de 2016	Comienzo del bloque 3 Programación en JSP
miércoles 17 de febrero de 2015	Exámen del bloque 2 - PHP Avanzado -- Concretar con alumnos --
jueves 3 de marzo	Entrega de la práctica completa
miércoles 9 de de marzo	Examen del bloque 3 Programación en JSP
XX de marzo	Posibles recuperaciones de última hora
XX de febrero o enero	Recuperación de alumnos que tienen pendiente el bloque 1 – PHP -- Concretar con alumnos--
Aprox ¿>15 de marzo?	Fecha realización 2ª Evaluación

1. Aplicación tienda on-line (8 Ptos)

Para el desarrollo de esta aplicación será obligatorio que utilicéis un repositorio de código GIT público ([GitHub](#) u otros) que cada usuario creará utilizando las herramientas que disponga vuestro entorno de desarrollo para trabajar con el mismo. Se tratará de utilizar las herramientas que proporciona nuestro entorno de desarrollo para sincronizar dicho repositorio.

La aplicación debe ser mostrada al profesor funcionando en Gerión (Servidor del instituto). Se recomienda que se vaya probando en el servidor a medida que se va realizando para evitar problemas de última hora. Utilizando las herramientas de FTP es muy sencillo sincronizar nuestro proyecto con el remoto.

Las especificaciones siguientes puede que estén incompletas, tan solo son una recopilación de las necesidades iniciales que nos ha transmitido el cliente.

Una empresa de venta on-line decide crear su aplicación de tienda virtual, la cual tendrá dos partes diferenciadas "tienda on-line" y "área de administración" las cuales **serán desarrolladas por equipos independientes**. Se os ha encargado a vosotros la realización de la "tienda on-line", el cual deberá implementar la funcionalidad que a continuación se describe.

Se desea realizar una aplicación web que nos permita vender productos a través de una tienda on-line. Los productos a vender serán elección vuestra pero la aplicación debe cumplir los siguientes requisitos:

- La tienda dispondrá un área de cliente, tienda visible para el usuario, y de *un área de administración y venta*. El área de administración será desarrollada por un equipo independiente.
- Debe permitir organizar los productos en venta por categorías. Se podrán establecer más relaciones jerárquicas entre los productos si así se estima oportuno.
- Para los productos en venta almacenaremos al menos la siguiente información:
 - Nombre: Nombre del producto
 - Código: Código con el que se lo identifica en la organización.
 - Precio de venta: Precio del producto en euros.
 - Descuento aplicable: Descuento que se aplicará al producto.
 - Imagen del producto: Imagen del producto que se mostrará
 - Iva aplicable: Tipo de IVA que se aplicará al producto
 - Descripción: Texto explicativo con las características del producto
 - Anuncio: Texto o HTML de reclamo que aparecerá en una parte destacada cuando mostremos el producto. (Opcional)
- Para las categorías se almacenará al menos la siguiente información:
 - Nombre: Nombre de la categoría
 - Código: Código interno con el que se identifica en la organización
 - Descripción.
 - Anuncio: Texto (HTML) de reclamo que aparecerá en una parte destacada cuando mostremos el producto. (Opcional)
- La aplicación tendrá una página de inicio en la que se mostrarán los artículos seleccionados por el administrador de la tienda. Dichos artículos se mostrarán de forma destacada y podrán ser configurados para que se muestren siempre o en un rango de fechas.
- El cambio de precio en los productos no afecta a las ventas ya realizadas o en proceso de entrega.
- Los productos y las categorías podrán ocultarse, por motivos de gestión, de forma que no aparezcan en la información de la tienda.
- Para realizar una venta se utilizará la metáfora de carrito de la compra controlada por sesiones que se almacenarán en el servidor. Deberéis **crear una librería carrito** y utilizarla para implementar esta funcionalidad. Para la librería creada deberéis crear una batería de pruebas unitarias que pruebe su correcto funcionamiento. Esto será puntuado en los siguientes ejercicios. Para comenzar a trabajar podéis utilizar la librería "Cart" que viene con Codeigniter, aunque teniendo en mente que luego deberá ser sustituida por la vuestra.
- En todo momento el usuario podrá consultar el contenido de su carrito de la compra. Pudiendo eliminar algún producto o vaciándolo por completo.
- En una operación de compra un usuario puede seleccionar comprar una unidad o varias siempre que éstas no superen el stock máximo del artículo (nº de artículos que disponemos en la tienda). Cuando la compra se haga efectiva, el stock disminuirá en número equivalente a la compra realizada.
- En una operación de compra se puede comprar diferentes productos.

- Solo se podrá comprar un artículo si este está disponible en stock.
- Para completar una venta es preciso estar registrado en el sistema. En dicho registro los usuarios introducirán la siguiente información:
 - Nombre usuario
 - Contraseña
 - Correo electrónico
 - Nombre
 - Apellidos
 - Dni valido
 - Dirección
 - CP
 - Provincia

Se deben filtrar los datos, de forma que todos sean válidos.

- En un futuro puede que nuestro sistema permita que los usuarios se validen en servicios remotos, los cuales nos proporcionarán los mismos datos que tenemos almacenados en nuestro registro. Es por este motivo por el que se recomienda crear un diseño que sea independiente de la estructura de datos utilizada.
- Solo será necesario introducir todos los datos de usuario la primera vez, las siguientes veces el usuario se identificará con su nombre de usuario y contraseña.
- El usuario podrá registrarse en el proceso de compra y continuar con el proceso de compra una vez haya sido registrado.
- En la página principal se mostrará información sobre el usuario si este se ha validado.
- El usuario deberá poder restablecer su contraseña utilizando el correo electrónico.
- Por seguridad es obligatorio que las contraseñas de los usuarios no puedan obtenerse por parte de los técnicos del sistema. Se codificarán las claves mediante técnicas de cifrado asimétrico (md5, sha1, etc)
- Los usuarios podrán actualizar sus datos o darse de baja cuando lo deseen. La actualización de los datos del usuario no modifica la dirección de envío, ni ninguno de los datos legales relevantes a efectos de facturación de los pedidos ya realizados anteriormente.
- Cuando se haga efectiva una venta se mostrará un resumen de los productos vendidos. El usuario recibirá igualmente un correo con la misma información.
- El usuario recibirá un fichero adjunto en el que en PDF verá el detalle del pedido realizado.
- El usuario podrá consultar en todo momento la lista de pedidos que ha realizado, viendo los detalles de este (artículos, importe, datos personales).
 - Si el pedido aun no ha sido procesado podrá anularlo.
 - Podrá ver en pantalla o descargar en PDF el detalle del pedido realizado (el albarán). Éste será el mismo documento que se envía por correo
- La lista de productos disponibles deberá aparecer paginada. Se deberían introducir los suficientes productos para permitir ver la paginación. La opción del número de elementos

que aparecen en la página deberá ser un elemento fácilmente configurable a través de ficheros de configuración.

- Cuando se confirma una compra el pedido quedará marcado como pendiente de procesar. Luego cuando se cree el paquete y se entregue a la agencia de viajes se marcará el pedido como procesado. Finalmente cuando la agencia nos comunique su recepción se marcará como recibido.
- Nuestro proceso de compra debería parecerse a los modelos vigentes, persiguiendo facilitar la compra en todo momento. El usuario podrá iniciar un proceso de compra si estar registrado o haberse identificado en el sistema. Ya se identificará o registrará cuando sea preciso para completar nuestra compra.
- Nuestra tienda deberá ajustarse a la normativa vigente e incluir toda la información legal que se precise.
- En la medida de lo posible sería deseable que nuestra aplicación fuese versátil y configurable a través de ficheros de configuración. Por ejemplo sobre los datos de personas de contacto, nº de elementos al paginar, nº de artículos en portada.

Se describe a continuación los requisitos que se ha solicitado al equipo encargado de la realización del área de administración para que lo tengáis en cuenta a la hora de elaborar vuestro diseño de datos.

- *La aplicación tendrá uno o más usuarios administradores, los cuales podrán acceder al área de administración.*
- *El usuario administrador podrá realizar las siguientes operaciones:*
 - *Crear / Modificar / Eliminar categorías*
 - *Crear / Modificar / Eliminar productos*
 - *Actualizar stocks de productos.*
 - *Cuando realicemos una actualización de stock se registrará con el objeto de saber cuantos productos se han comprado.*
 - *Mostrar **ventas pendientes de procesar***
 - *Marcar **ventas como enviadas**.*
 - *Marcar ventas **como recibidas**.*
 - *Mostrar informe de ventas realizadas en un intervalo de fechas.*
 - *Generar documento de factura para una venta.*
 - *Mostrar resumen de importe de ventas mensual.*
 - *Los informes que tengan más de 10 (este valor será mayor en una aplicación real) elementos serán paginados.*

Los informes anteriores se deberán mostrar en HTML, aunque se aceptarán otro tipo de formatos.

Debido a que os describen unas especificaciones de una aplicación en primer lugar deberíais tratar de concretar y organizar lo que se os solicita, para ello se recomienda que procedáis de la siguiente forma:

- Realicéis digrama de clases (entidad/relación) indicando las entidades que forman parte del sistema. En nuestro caso deberemos por una parte indentificar nuestro modelo de

datos (entidades o clases) y por otra parte al utilizar el patrón MVC identificar los controladores, modelos y vistas que se utilizarán.

- Realicéis digrama de casos de uso, y los casos de uso correspondientes, para identificar los roles de actores que hay en el sistema y la funcionalidad que debe aportar.
- Diagrama de estados para todas aquellas entidades que evolucionan con el tiempo.
- Otros diagramas que consideréis pertinentes.

Antes de comenzar con la programación se recomienda que reviséis vuestro modelo de datos con el profesor para ver si tiene alguna carencia difícil de subsanar más adelante.

Nota:

- En la web Gnome-look <http://gnome-look.org/> podéis encontrar múltiples paquetes de iconos gratuitos que os permitirán realizar una aplicación con una presentación bonita y coherente. Igualmente podréis buscar por "webs de iconos" y tendréis múltiples opciones.
- Se valorará la utilización de framewos de CSS y Javascript en la presentación de la aplicación. Como JQueryUI, BootStrap, HTML5 Boilerplate, etc.

Puntuación desglosada por apartados que hay que realizar:

*** en revisión ***

1. Mostrar productos (1 pto)
 - Destacados en página inicio (0,5 ptos)
 - Lista de categorías (0,5 ptos)
2. Carrito de la compra (1,5 ptos)
 - Añadir y consultar lista de productos en carrito (0,75 ptos)
 - Eliminar y vaciar producto del carrito (0,25 ptos)
3. Usuarios (2 ptos)
 - Registro (0,75 ptos)
 - Modificación de datos (0,5 ptos)
 - Baja (0,25 ptos)
 - Restablecer contraseña utilizando el correo (0,5 ptos)
4. Proceso de venta (creación de pedido) (1,5)
 - Mostrar resumen de la lista de productos que forman parte del pedido (0,5 ptos)
 - Envío por correo del detalle con una presentación apropiada (0,5 ptos)
 - Envío por correo del fichero PDF con los detalles. (0,5 ptos)
5. Pedidos (1,5 ptos)
 - Mostrar pedidos realizados (0,5 ptos)
 - Anular pedido, sólo si aun no se ha enviado (0,5 ptos)
 - Generar albarán/factura de un pedido seleccionado en PDF (0,5 ptos)
6. Valoración global de la aplicación completa (2 ptos).

Nota: Para que sea la puntuación de los siguientes ejercicios/apartados sea considerada la aplicación debe ser mínimamente operativa.

2. Servicios web (2 ptos)

*** En revisión ***

Agregador de tiendas

Se desea que nuestra aplicación funcione como proveedor de servicios y permita que otras aplicaciones puedan mostrar nuestros productos destacados. Vamos a trabajar en colaboración con el [agregador de tiendas](#).

Disponéis del código fuente del mismo, en el que encontraréis ejemplos sobre como implementar la API en el fichero "[Agregador_Tiendas.zip](#)".

En dicho código tenéis implementado la parte cliente del agregador y la parte del proveedor de servicios (carpeta "server") en una situación normal esto se encuentra en máquinas diferentes. Para vosotros lo interesante es solamente la parte del servidor, pues la parte cliente ya está implementada y será la que haga uso de vuestro servicio.

API del proveedor de servicios que debemos implementar para el Agregador

Vuestra aplicación creará un servicio haciendo uso de la librería "JSON_WebClient" que hemos creado en la que implementará la siguiente funcionalidad.

Para que el agregador funcione correctamente hará uso de los siguientes servicios. Los cuales implementaremos en nuestra aplicación tienda on-line.

Servicio: Proveedor productos

Descripción: Servicio web nos proporciona informacion sobre los productos que hay en nuestra tienda

URL: [Dominio_y_ruta_APP]/index.php/srv/Usuario

Operaciones

<i>Total()</i>	Devuelve el número total de productos que hay en nuestra tienda
<i>Lista(offset, limit)</i>	<p>Devolverá la lista de productos que existen en nuestra tienda comenzando por "offset" y limitado a un número de elementos "limit". Como el limit de MySQL.</p> <p>Se devolverá un array de objetos que contendrán la información de los productos, con el siguiente esquema:</p> <pre>[['nombre'=>texto, 'descripcion'=>text, 'precio'=>precio del producto, 'img'=>url donde está la imagen, 'url'=>url en la que se puede comprar el producto en vuestra aplicacion] ...];</pre>

Para que el agregador reconozca vuestra tienda deberéis [registrarla](#) en la aplicación agregador. El

registro consistirá en la introducción de varios campos:

- Nombre: Nombre de la tienda que aparecerá en la listadas
- Descripción: Texto explicativo de nuestra tienda
- URL: URL en la que está operativo nuestro servicio

Podéis registrar tantas tiendas como deseéis, si deseais borrar alguna registrada incorrectamente notificarmelo por correo.

Para probar correctamente vuestro servidor desde el agregador de tiendas deberéis tenerlo instalado en una máquina que tenga acceso público como Gerión. La prueba de funcionamiento se realizará sobre esta máquina.

3. Aplicación tienda on-line – Mejoras adicionales

*** En construcción ***

La puntuación de estas mejoras se añadirá a la de la nota obtenida en los apartados obligatorios

Creación de librería carrito de la compra (1,5 pto)

Se deberá crear una librería personal que nos implemente un carrito de la compra y realizar las pruebas unitarias pertinentes de funcionamiento

- Creación de librería carrito (0,5 ptos)
- Pruebas unitarias de la librería carrito (0,5 ptos)

Importación y exportación en XML (2 ptos)

El programa permitirá exportar e importar los datos contenidos en la base de datos relativos a artículos y categorías en un fichero XML.

Exportación 1 pto.

Importación 1 pto.

Esto se puede realizar de forma simple utilizando la librería SimpleXML

Soporte para múltiples monedas en vuestra aplicación (1 pto)

Utilizando la información prestada por el Banco Central Europeo (<http://www.ecb.europa.eu/stats/exchange/eurofxref/html/index.en.html>) se trata de que vuestra aplicación permita trabajar con diferentes monedas, y permita realizar transacciones acorde al cambio existente en el momento.

Importación de datos en formato de hoja de calculo excel (2 ptos)

El programa permitirá importar los datos de los artículos y las categorías de una hoja de cálculo excel.

Utilización de Ajax con JSON o XML (1 ptos)

*** En revisión ***

Implementar funcionalidades de vuestra web utilizando Ajax y JSON o XML.

Un lugar habitual en el que las webs usan ajax es en el proceso de añadir o suprimir al carrito de la compra.

Comentarios y evaluación

- La nota de la práctica supondrá el 50% de la calificación sobre el bloque de contenido evaluado, por lo que su entrega será obligatoria.
- La copia de todo o parte del ejercicio supondrá la inmediata eliminación de la parte copiada. Se dividirá la nota de los implicados entre el número de copias. Igualmente en caso de identificar a la persona que ha copiado se le penalizará en su calificación.
- Cada alumno enseñará individualmente el funcionamiento de la aplicación al profesor. Antes de la fecha de entrega.

Puntuación ejercicios

La puntuación de cada ejercicio es la que se indica en su enunciado.

La práctica puntúa sobre 10, el exceso de puntos que pudierais tener se considerará igualmente para el cálculo de la media.

Criterios de evaluación

La corrección y puntuación de cada apartado se realizará atendiendo a los siguientes parámetros:

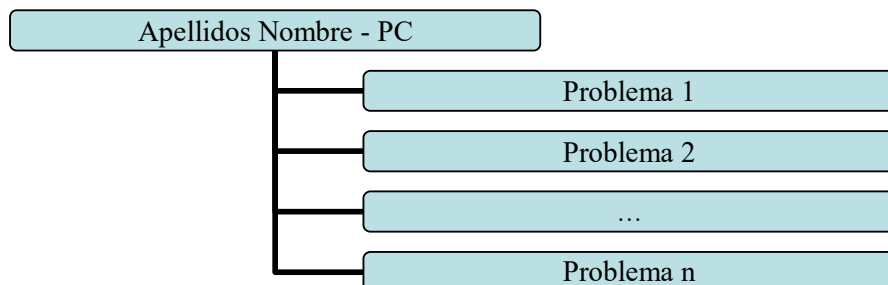
- Interfaz: Que el programa que utilicen recursos gráficos que faciliten la interacción con el usuario y se usen los recursos que proporciona el entorno de desarrollo
- Funcionalidad: Que el programa realice lo que se pide
- Estilo de programación: que el código del programa sea fácilmente entendible y modificable por otras personas. Para ello deberá regirse por las directrices de la programación estructurada, la programación orientada a objetos y utilizar patrones de diseño de software.

Instrucciones para entregar la práctica

La práctica se subirá comprimida al aula virtual en una tarea habilitada al efecto.

Dentro del fichero comprimido incluiréis el código fuente y ficheros de proyecto de vuestro programa. Los cuales están contenidos en la carpeta del proyecto.

El fichero estará organizado por carpetas siguiendo la siguiente estructura



Esta práctica deberá ser publicada en el dominio iessansebastian.com, en el espacio que se os ha reservado al efecto.