Aplicación venta online para CadenaTiendas

Primera iteración

Comenzamos la aplicación definiendo el núcleo central de la base de datos.

Tras agregar algunos datos, se hace un ejercicio de gestión de transacciones consistente en mover un artículo del almacén de una tienda al almacén de otra tienda.

Finalizado el ejercicio, se reparte a los alumnos la base de datos cargada con una serie de datos de prueba.

Se agrega la columna `foto` a la tabla `articulo` y tomando como base el código de ejemplo de Microsoft para agregar fotos a la base de datos `AdventureWorks`, se estructura ese código según el patrón Modelo-Vista-Controlador.

El resultado es una estructura de directorios en la que se reparten los archivos de código que controlan la aplicación (lógica de la aplicación-el controlador), los archivos de código que acceden a la base de datos (lógica de negocio-el modelo) y los archivos de código que permiten a los usuarios interactuar con la aplicación (GUI-la vista).

Además de estos directorios y archivos, en otros directorios se detallan archivos específicos para mantener las rutas (qué controlador y qué método del mismo (acción) debe tratar una solicitud del usuario cuando este cliquea en un enlace), archivos para mantener determinados aspectos de la configuración de la aplicación y archivos para establecer la conexión con la base de datos. El objetivo de esta distribución es el de ubicar de manera centralizada estos aspectos de la aplicación, teniendo en mente la implementación de permisos de acceso a través, o bien del servidor de la aplicación (Apache) o bien del sistema (Linux, Windows, etc.) que incrementan la seguridad de la aplicación al delimitar las ubicaciones a las que pueden acceder los usuarios.

Esta estructura/desarrollo del patrón MVC es la guía para estructurar el código en las sucesivas iteraciones que van a agregar funcionalidad nueva a la aplicación.

Segunda iteración

En esta iteración se plantea la identificación/ registro de usuarios. Para ello se agrega a la base de datos la tabla `usuario` con dos únicas columnas: `usuario` y `contrasenia` y, siguiendo las guías del patrón MVC, se agrega una nueva ruta al archivo de rutas y se crean el controlador, el repositorio y las vistas que permiten al usuario identificarse ante la aplicación.

Se reparte a los alumnos el código de los desarrollos realizados hasta este momento.

Realizadas estas tareas, se ve la necesidad de otorgar permisos a los diferentes usuarios, establecer y terminar sesiones e incrementar la seguridad de las contraseñas, examinando de cerca los algunos de los aspectos que rodean a estas características.

Se agrega código para crear sesiones cada vez que un visitante accede al sitio de la aplicación.

Para otorgar permisos y facilitar su gestión, se agrega a la tabla usuario la columna “`grupo`”. A través de la información que mantiene esta columna vamos a conceder permisos a grupos de usuarios en lugar de a usuarios individuales.

Se establecen tres tipos de usuarios `admins`, que pueden gestionar, supervisar y modificar cualquier aspecto de la aplicación; `clientes`, que pueden visitar, navegar por la parte pública de la aplicación para conocer los productos que ofrece la aplicación, reservar productos, comprarlos, modificar sus datos y conocer el estado de sus pedidos; `visitantes`, que pueden visitar, navegar por la parte pública de la aplicación para conocer los productos que ofrece la aplicación y reservar productos. El seguimiento de pertenencia a grupo se realiza a través de la sesión.

Se modifican el controlador y las vistas relacionadas con la identificación/registro de usuario para que solo los usuarios del grupo "`admins`" puedan subir/modificar las fotos de los artículos. El patrón MVC hace sencillas estas modificaciones.

Tercera iteración

Esta iteración completa la iteración anterior. Se plantea la creación de formularios que permitan a los nuevos clientes registrarse. Para ello se agregan nuevas columnas a la tabla usuario: `nombre`, `apellido`, `calle`, `ciudad`, `cp`, `provincia`, `tlfno` y `eCorreo`. Estos nuevos usuarios pertenecerán siempre al grupo "`clientes`".

Aspectos a tratar: evitar la inyección de código SQL, ataques XSS, obligatoriedad de contraseñas fuertes y cifrado de contraseñas.

Para evitar la inyección de código y ataques XSS se da entrada plena a las entidades. La entidad `Usuario` controla los valores que introduce el usuario en cada una de las columnas: se evitan combinaciones de caracteres potencialmente peligrosos como: `<`, `>`, `--`, "`;`", etc., que se pueden utilizar para inyectar tanto código SQL como de script. Los métodos `set` de la entidad `Usuario` emplean tanto patrones, que solo permiten una serie de caracteres adecuados a la información que debe mantener cada columna, como funciones PHP que analizan los caracteres de las cadenas, eliminan espacios iniciales y finales, eliminan backslashes y convierten determinados caracteres en combinaciones de caracteres `HTML` inofensivos.

Para las contraseñas se implementa obligatoriedad de contraseñas fuertes. Estas constan de un mínimo de ocho caracteres con al menos una letra mayúscula, una minúscula, un número y un carácter no alfanumérico. Para el cifrado se utilizan herramientas `PHP` específicas y se hace notar una posible debilidad: la combinación de caracteres "`\0`" que acortan la longitud de la contraseña. Para que la columna `contrasenia` de la tabla `usuario` pueda albergar la información de estas contraseñas cifradas se aumenta su capacidad a `255` caracteres `UTF`. En la entidad `Usuario` un método `set` específico trata e impone todos estos aspectos.

Se hace notar que todo esto no es suficiente, se necesitan certificados de seguridad para utilizar protocolos seguros, así como código en el lado cliente que realice parte de la tarea. Como ejemplo se utiliza CSS para controlar la entrada de usuario.

La iteración termina haciendo que la cabecera y la barra de navegación de aplicación sean dinámicas. La cabecera muestra un saludo al usuario con su nombre y si se pasa el puntero del ratón por encima aparece un menú contextual con dos opciones que permiten al usuario modificar sus datos y terminar la sesión.

En la barra de navegación desaparece el enlace de inicio de sesión cuando se registra un usuario, y si el usuario pertenece al grupo de `admins`, aparece un enlace para modificar/agregar fotografías.

Todas estas tareas se realizan siguiendo el patrón MVC mediante la estructura de archivos impartida. Se agregan nuevas rutas así como nuevos controladores, entidades/repositorios y vistas que permiten distribuir y crear fácilmente el código que implementa las nuevas funcionalidades.

Cuarta iteración

En esta iteración se aborda la gestión de las compras por parte de los clientes. Para mantener la información de las compras que han realizado los clientes se agrega a la base de datos la tabla `venta` con las columnas: `idVenta`, `idCliente`, `idArticulo`, `fCompra`, `fEntrega` y `pv`. `fCompra` recoge la fecha en la que el cliente realizó la compra del artículo, y `fEntrega` la fecha en la que el artículo fue entregado a cliente.

Un cliente puede pulsar el botón comprar desde diferentes ubicaciones:

- Puede acceder a mediante el enlace todos los productos.
- Puede acceder a los productos a través de las categorías de productos: fotografía, informática, etc.
- Puede acceder realizando una búsqueda mediante el control situado al efecto.
- Puede acceder cuando repasa la descripción del producto, vista a la que se accede desde cualquier de las anteriores.

Cada artículo comprado (reservado en realidad) se lista bajo el carrito de compra situado en la cabecera de la aplicación. Esta lista aparece cuando se pasa el ratón por encima del carrito. Este menú contextual da la opción de realizar efectivamente la compra o de vaciar el carro.

Cuando el cliente pulsa el botón de comprar, si el cliente se halla ya registrado accede al formulario de pago, sino aparece el formulario de registro.

En el formulario de pago se le solicitan una serie de datos (ficticios) de pago: tipo de tarjeta, número de tarjeta, caducidad, cvd, etc. Una vez cumplimentados estos datos se por finalizada la compra y se le pregunta al cliente si desea seguir comprando y se le recuerda que antes de salir debe terminar la sesión.

Quinta iteración

En esta última iteración se completa la iteración anterior y se aborda la problemática de la logística.

Durante la compra se le pregunta al cliente si desea recibir los artículos en su casa o se acerca a recogerlos a la tienda más cercana.

Si solicita recibirlos en su casa, el artículo se extraerá del almacén de los servicios centrales en Madrid. Si desea recogerlo en tienda se examina la disponibilidad del artículo en el almacén de la tienda y en caso de que no hayan existencias se informa al cliente de que deberá esperar dos días a que la tienda reciba el artículo.

El artículo le puede llegar a la tienda vía almacén de cualquiera de las otras tiendas. Estos desplazamientos de artículos entre almacenes se cumplimentan mediante transacciones.

Otras iteraciones

Otras posibles funcionalidades a agregar son:

- Modificación de los datos de usuario.
- Enviar los artículos comprados a un destino diferente del de la dirección de facturación.
- Envío de los artículos desde la tienda más cercana al cliente.
- Especificar artículo agotado pero que se espera recibir nuevas unidades.

- Diferenciar la reserva de artículos (cuando un cliente pulsa el botón comprar) de la compra del artículo (cuando el cliente realiza efectivamente la compra), para evitar que un artículo que aparecía como “con existencias” no exista cuando se intenta comprar.
- Cuando el cliente solicita recibir los artículos en su domicilio, diferenciar artículos en tránsito y baja en almacén hasta que el cliente recibe efectivamente los artículos en su domicilio.