# Sistema de Tiendas Virtuales

## Programación Web 3

## Tecnicatura en Desarrollo Web - UNLaM

*1do Cuatrimestre, 2013*

RESUMEN PROYECTO

## Propósito y Objetivos

Crear un sitio Web que permita a diferentes vendedores tener sus tiendas virtuales administrando sus productos y stock disponibles. Los usuarios finales ingresarán y, luego,elegirán el local en el cual podrán ver los productos del mismo y, en caso de estar interesados, realizar la compra de un producto.

Cada local podrá administrar los productos que venderán y el stock de los mismos.

El sitio web deberá cumplir con los siguientes objetivos

1. Proveer un portal donde el cliente ingresará para elegir la tienda virtual
2. Mostrar los productos de una tienda virtual seleccionada.
3. Permitir a un cliente comprar un producto.
4. Proveer un acceso a las tiendas virtuales mediante la registración de una cuenta.
5. Habilitar la administración de los locales al usuario administrador.
6. Habilitar la administración de productos a cada local.

La idea de este sistema es facilitar un local virtual a usuarios/empresas que no poseen un espacio físico para realizar sus ventas.

## Carga de Datos

Deberá existir una carga inicial de al menos 3 locales donde cada uno tenga al menos 3 productos, algunos con stock disponible y otros sin stock.

REQUISITOS GLOBALES

## Requisitos No Funcionales

Compatibilidad con exploradores.

- Internet Explorer 9,10
- Firefox (la última versión para Windows).
- Google Chrome (la última versión para Windows).

Errores

- Ninguna Excepción en momento de ejecución puede ser vista por ningún usuario del sitio.
    - (*) Pagina No Encontrada:
        - Se debe de crear una página para los errores del tipo 404 (página no encontrada).
    - (*) Descripciones de Error:
        - Los usuarios del sitio solo deben de ver mensajes amigables.
        - Se debe de crear una página para cualquier otro error arrojado por la aplicación y que

no haya sido atrapado.

- Logging:
  - Debe de existir un sistema de logging de errores, donde se guarde la fecha, url del error y toda la información relacionada al error para facilitar su corrección. Este logging puede hacerse guardando la información en una tabla o se puede guardar en un archivo (una opción es utilizar la biblioteca Log4Net)

* Esto se debe implementar utilizando la configuración de custom errors en el web.config

---

## ESPECIFICACIONES FUNCIONALES

---

En esta sección se detallarán los detalles funcionales para cada componente de la solución.

### 1. Registración/Login de Tiendas Virtuales

---

Corresponde a la primera acción que hará la tienda virtual sobre el sitio a fin de poder utilizar sus servicios.

1. Campos del formulario:
   a. *Razón social
   b. CUIT
   c. *Mail
   d. *Repetir contraseña
   e. *Contraseña
   f. *Captcha ( que es un Captcha?)

   *campos que el usuario esta obligado a completar.
2. Con el mail y la contraseña el administrador de la tienda virtual podrá ingresar con permisos para editar las páginas de administración de productos, administración de categorías y administración de compras.
3. Cada tienda virtual podrá editar sus datos personales.
4. Al registrarse, la tienda virtual debe recibir un mail con un link de activación de la cuenta.
   a. Hasta no entrar a ese link, la tienda virtual se encontrará en estado inactivo. En este estado, una tienda virtual no puede ingresar al sistema.
   b. Este link solo puede usarse una vez para activar la tienda virtual y debe expirar a la media hora de la registración.

### 2. Administración de Tiendas Virtuales (sólo para el administrador de la Tienda)

---

En esta página se editará la siguiente información acerca de las tiendas virtuales

1. Se podrá activar/desactivar a una tienda virtual.
2. Podrá visualizar todas las compras realizadas por los clientes finales y ver el total de ventas.

.

### 3. Categorías

---

No existirá una pagina para administrar Categorías. En aquellos lugares donde se deba seleccionar una categoria, se debera crear un **UserControl ucElegirCategoria** que contendrá un control dropdownlist.
Las categorías disponibles en el sistema serán:

| Id | Nombre | Id | Nombre |
|---|---|---|---|
| 1 | Accesorios para Vehículos | 14 | Entradas para Eventos |
| 2 | Animales y Mascotas | 15 | Hogar, Muebles y Jardín |
| 3 | Antigüedades | 16 | Industrias y Oficinas |
| 4 | Autos, Motos y Otros | 17 | Inmuebles |
| 5 | Bebés | 18 | Instrumentos Musicales |
| 6 | Cámaras y Accesorios | 19 | Joyas y Relojes |
| 7 | Celulares y Teléfonos | 20 | Juegos y Juguetes |
| 8 | Coleccionables y Hobbies | 21 | Libros, Revistas y Comics |
| 9 | Computación | 22 | Música, Películas y Series |
| 10 | Consolas y Videojuegos | 23 | Otras categorías |
| 11 | Deportes y Fitness | 24 | Ropa y Accesorios |
| 12 | Electrodomésticos y Aires Ac. | 25 | Salud y Belleza |
| 13 | Electrónica, Audio y Video | 26 | Servicios |

### 4. Administración de Productos (sólo para el administrador de la Tienda)

---

Página para la administración de Productos. Cada tienda virtual podrá administrar sus productos. Por cada producto, se registrarán los siguientes datos:

1. Campos posibles:
    a. Nombre
    b. Descripción.
    c. Cantidad en Stock.
    d. Precio.
    e. Categoría.
    f. Imagen (archivo jpg o png del producto). Solo se permitirá una imagen.
2. Se deberá validar que todos los datos son obligatorios. En el caso de cantidad de stock solo se permitirán valores enteros y en el caso del precio solo valores decimales.
3. Las imágenes deberán ser guardadas en un tamaño fijo de 150x150.
4. Una tienda virtual no podrá registrar productos repetidos (con mismo nombre).
5. Se permitirán las operaciones de crear, modificar y eliminar lógicamente un producto.
6. Al crear un producto, se debe invocar al método RegistrarProducto dela API de Stock a traves del code-behind.
7. Al modificar un producto, se debe invocar al método ModificarProducto dela API de Stock a traves del code-behind.
8. Al eliminar un producto, se debe invocar al método EliminarProducto dela API de Stock a traves del code-behind.

### 5. Administración de Compras (sólo para el administrador de la Tienda)

Página para la administración de Compras. Cada tienda virtual podrá administrar las compras realizadas por los clientes. Las compras serán realizadas desde el portal por lo clientes.

1. La tienda virtual podrá listar las compras de una fecha dada.
2. La tienda virtual confirmará un pedido cuando el mismo esté finalizado.
3. La tienda virtual verá la información del producto solicitado, la cantidad y el email del cliente.

### 6 Panel de Tiendas Virtuales

En esta página se podrán visualizar todas las tiendas virtuales activas y que tengan productos disponibles. Un cliente ingresará y elegirá una categoría de producto. Una vez que elija una categoría de producto, el usuario verá las tiendas virtuales que tengan productos de dicha categoría.

1. Se listarán las tiendas virtuales con productos de la categoría seleccionada.
2. En caso de que no tengan al menos un producto con la categoría seleccionada la tienda virtual no se deberá mostrar.
3. Al seleccionar una tienda, el cliente irá a una nueva pantalla donde se mostrarán los productos de la categoría seleccionada que la tienda posee.
4. Los productos se verán con la imagen y el nombre del mismo.
5. Al hacer click sobre el producto, el cliente visualizará el detalle del producto viendo el nombre, la descripción, la imagen y el stock disponible del mismo.
6. En caso de que el producto tenga stock, el cliente verá un botón de comprar. Al presionarlo el cliente, verá una opción para ingresar la cantidad de productos que desea, su email de contacto y confirmar el pedido.
7. Se deberá validar que la cantidad de productos seleccionados, tenga disponibilidad en el stock.
8. El cliente podrá compartir en su muro de Facebook, el producto con la descripción y link de visualización.
9. Al confirmar una compra, el stock disponible de un producto deberá decrementarse en la cantidad solicitada por el cliente.

Nota: Toda la obtencion/actualización de valores referidos a stock, deberá ser utilizando la API de Stock desde el code-behind.

### 7 API de Stock

Se creara una API independiente para todas las operaciones de manejo de stock y poseerá los siguientes métodos:

1. RegistrarProducto(idProducto, stockInicial)
    a. Respuesta:
        i. bool OperacionExitosa
        ii. string Message **(1)**
2. ModificarProducto(idProducto, nuevoStock)
    a. Parametros:
        i. nuevoStock: determina cual sera el nuevo valor de stock del producto.
    b. Respuesta:
        i. bool OperacionExitosa
        ii. string Message **(1)**
3. ActualizarStockProducto(idProducto, cantidad)
    a. Parametros:
        i. cantidad: valor en que aumenta/disminuye el actual stock del producto.
    b. Respuesta:

            i. bool OperacionExitosa
            ii. string Message **(1)**
            iii. int Stock
                1. nuevo stock del producto
4. ObtenerStockProducto(idProducto)
    a. Respuesta:
        i. bool OperacionExitosa
        ii. string Message **(1)**
        iii. int Stock
            1. stock del producto
5. EliminarProducto(idProducto)
    a. Respuesta:
        i. bool OperacionExitosa
        ii. string Message **(1)**

**(1)** en caso de ser false OperacionExitosa, aquí debería de recibirse la de descripción del error o de lo que sucedió.

Nota: Esta api sera un web service .asmx que devolvera un xml.

---

ENFOQUE

---

## Diseño y Arquitectura

---

1. Estilo
    a. Todos los archivos .css deberán estar dentro de una carpeta.
    b. No utilizar estilos inline (atributo style="") ni definir estilos dentro de una pagina (tags <style>).
2. JavaScript
    a. No utilizar JavaScript inline dentro de una página, se deberá referenciar a archivos js.
    b. Todos los archivos .js deberán estar dentro de una carpeta.
        i. Si se decide utilizar algún js que no es propio, el mismo deberá estar dentro de una subcarpeta.
        ii. Recomendaciones
            1. Las funciones específicas de una página, deberían estar en un archivo .js con el mismo nombre de la página.
    c. Aquellas funciones utilizadas en más de una página, deberían de estar dentro de otro archivo .js.
3. HTML
    a. No utilizar tags table.
    b. Todo el contenido dentro debe ser SEO Friendly.
        i. Debe existir un archivo sitemap.xml (referencia: http://www.sitemaps.org/protocol.html)
    c. Debe utilizarse Master Pages.
4. Validación
    a. Utilizar validaciones tanto del lado del cliente (JavaScript) como del lado del servidor.
    b. Se puede utilizar una lista que detalle todos los campos que no cumplieron con las validaciones.

**Técnico**

1. Código:
    a. Utilizar la menor cantidad posible de código en los archivos aspx.cs, ascx.cs, master.cs, etc. e intentar que en los mismos haya llamadas a métodos dentro de otro proyecto que contenga las reglas de negocio.
    b. Las páginas **1. Registración/Login de Tiendas Virtuales** y **4. Administración de Productos** deben ser ASP.NET puras (con controles ASP.NET)
    c. La pagina **6 Panel de Tiendas Virtuales** debe estar hecha **sin controles asp.net** (la unica excepcion puede ser el control ucElegirCategoria). Y la carga/envio de informacion debera ser a traves de un **web service o web methods** que devuelvan **json** y que sea consumido a traves de **jquery ajax**.
    d. No hay restricciones para el resto de las páginas.
2. Seguridad
    a. Encriptar información sensible, como la contraseña de los usuarios.