**Universidad Abierta Interamericana**



***Facultad de Tecnología Informática***

***Lenguajes de Programación para la Administración***

**Trabajo Práctico Final**

**BuyMotors**

**Profesor:** Sábato, Sebastián

**Alumnos:** Goncalves, Lucas

Jorreto, Mariano

López, Ariel

Schiter, Tomás

Zurriable, Jorge

**Comisión:** 4°A

**Turno:** Noche

**Sede:** Centro

Contenido

[Descripción del negocio 4](#_Toc45370876)

[Beneficios del negocio 4](#_Toc45370877)

[Alta Disponibilidad 4](#_Toc45370878)

[Variedad de productos 4](#_Toc45370879)

[Seguridad en la compra 4](#_Toc45370880)

[Propósito 4](#_Toc45370881)

[Objetivos 4](#_Toc45370882)

[Limitaciones 5](#_Toc45370883)

[Análisis del uso 5](#_Toc45370884)

[Gestión Login-Logout 6](#_Toc45370885)

[Objetivo 6](#_Toc45370886)

[Descripción 6](#_Toc45370887)

[Caso de Uso 6](#_Toc45370888)

[Algoritmo de encriptación utilizado 8](#_Toc45370889)

[Ciclo de vida del proceso de Login, dividido en capas 9](#_Toc45370890)

[Interfaz de usuario (UI) 9](#_Toc45370891)

[Lógica de negocio (BL) 13](#_Toc45370892)

[Acceso a datos (DAL) 14](#_Toc45370893)

[Entidades (BE) 17](#_Toc45370894)

[Logueo (Log) 17](#_Toc45370895)

[Diagrama de secuencia 18](#_Toc45370896)

[DER 19](#_Toc45370897)

[Diagrama de clases 20](#_Toc45370898)

[Gestión de bitácoras 21](#_Toc45370899)

[Objetivo 21](#_Toc45370900)

[Descripción 21](#_Toc45370901)

[DER 21](#_Toc45370902)

[Diagrama de secuencia 22](#_Toc45370903)

[Guardar Bitácora 22](#_Toc45370904)

[Obtener Bitácora 22](#_Toc45370905)

[Diagrama de clases 22](#_Toc45370906)

[Capturas de pantalla del sistema 23](#_Toc45370907)

[Login 23](#_Toc45370908)

[Registro de nuevo usuario 24](#_Toc45370909)

[Pantalla principal (con webmaster logueado) 24](#_Toc45370910)

[Bitácora 25](#_Toc45370911)

[Backup 25](#_Toc45370912)

[Listado de Vehículos 26](#_Toc45370913)

[Detalle de un vehículo 26](#_Toc45370914)

[Carrito de compras 27](#_Toc45370915)

# Descripción del negocio

BuyMotors es una concesionaria de autos online que proporciona a los clientes una plataforma rápida, fácil y confiable para adquirir un vehículo. Esta posee un inventario detallado del stock, incluyendo las características de cada producto, su valor y disponibilidad.

El objetivo principal del negocio es servir como una solución tecnológica que permita la mayor disponibilidad posible en el servicio, permitiendo a los clientes acceder al inventario y realizar compras desde cualquier lugar y en cualquier momento del día.

En sus inicios, BuyMotors mostrará un inventario de vehículos que se encontrarán en su sede principal en Capital Federal, pero se proyecta a futuro una expansión tanto en más zonas de la capital como por el resto de Buenos Aires y del país, para imponerse como el servicio líder del país para la compra de vehículos.

# Beneficios del negocio

## Alta Disponibilidad

La plataforma estará en funcionamiento en todo momento, de forma que un cliente interesado puede rápidamente revisar el inventario y planificar una compra.

## Variedad de productos

El inventario poseerá vehículos de distintas marcas, colores, rangos de precios, capacidad de pasajeros y modelos. Se espera tener vehículos para satisfacer las necesidades de todos los tipos de clientes, desde padres de familias en busca de autos espaciosos hasta jóvenes con alto poder adquisitivo que buscan autos deportivos de las mejores marcas.

## Seguridad en la compra

Un cliente no debe dudar en utilizar la aplicación en cuanto a la seguridad de sus datos y de su dinero. Realizar una compra en BuyMotors es transparente en cuanto a la transacción del dinero y privada en cuanto a los datos del comprador.

# Propósito

El propósito de BuyMotors es utilizar la Web para proporcionar el mayor inventario de vehículos del país. Una persona que desea comprar un nuevo vehículo debe pensar en BuyMotors como la primera opción para realizar una compra basándose en la variedad de vehículos, la facilidad de uso y la confiabilidad de la plataforma para realizar una transacción segura.

# Objetivos

Para que BuyMotors se destaque de la competencia, será necesario que:

* Tenga un inventario extenso.
* Sea fácil de usar.
* Proporcione un entorno confiable para realizar transacciones de altas cantidades de dinero.
* Proveer un servicio a todo momento del día.

# Limitaciones

La solución no contará en sus comienzos con una versión para dispositivos móviles ni de una funcionalidad que facilite el uso a usuarios con discapacidades visuales. Esto se analizará a medida que se vea un crecimiento considerable en la cantidad de usuarios.

Se considerará incluir una versión del sistema en inglés cuando la estrategia a mediano plazo se enfoque en la expansión internacional (si es que ocurre).

# Análisis del uso

BuyMotors tiene como objetivo convertirse en el líder del mercado, por lo que apuntará a ser una solución fácil de usar por todo el mundo, incluyendo usuarios principiantes en cuanto a tecnología y personas con discapacidades. Además, se tendrán en cuenta las últimas tecnologías y se evaluará la posible inclusión de ellas en la aplicación para garantizar un sistema rápido y eficaz.

# Gestión Login-Logout

## Objetivo

Autenticar a todos los usuarios que utilizan el sistema. Solo podrán ingresar al sistema y realizar operaciones en él los que se encuentren registrados en el sistema previamente. Para esto siempre será necesario que se presenten credenciales válidas para poder ingresar y realizar las operaciones que esté autorizado a hacer. El usuario, cuando lo desee, podrá salir del sistema y cerrar sesión (logout).

## 

## Descripción

Antes de ingresar al sistema el usuario deberá ingresar su nombre de usuario y contraseña. El sistema deberá validar que las credenciales ingresadas sean válidas. Durante el proceso de validación se encripta la clave ingresada y se valida contra la almacenada en la base de datos, la cual se encuentra ya encriptada para prevenir que se puedan conseguir datos de accesos a partir de un ataque.

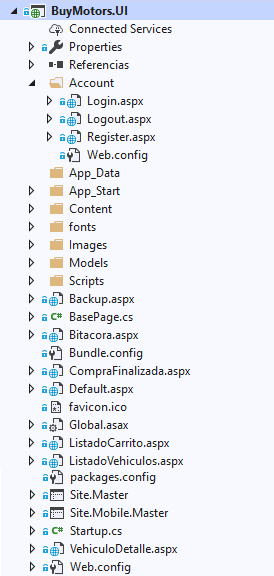
Una vez validada la autenticidad del usuario se lo asigna a la sesión actual. Para cerrar sesión (logout) el usuario manualmente deberá seleccionar la opción Cerrar Sesión y se lo enviará de regreso a la pantalla principal. Solo podrá haber una sesión simultánea por cada navegador.

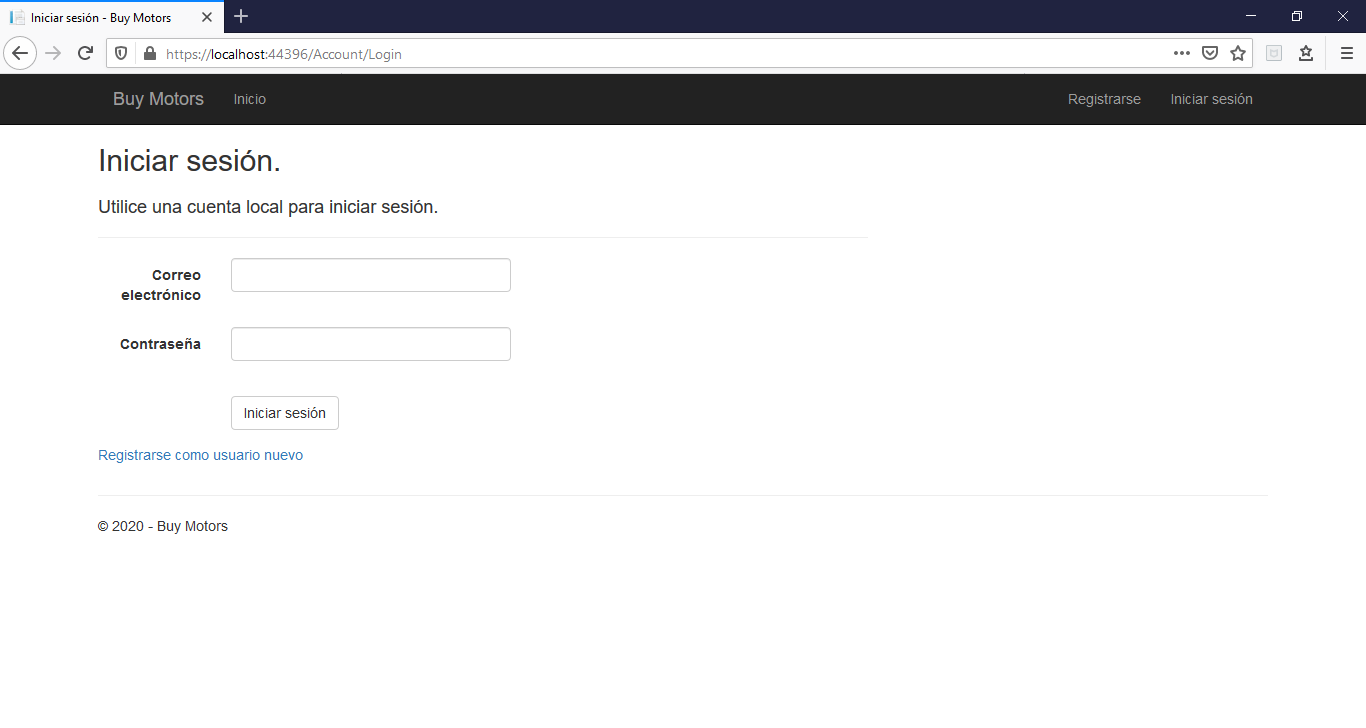
## Caso de Uso

|  |  |
| --- | --- |
| **Caso de uso: “Login”** | |
| **Propósito:** Validar la autenticidad del usuario que desea ingresar al sistema con el ingreso de su usuario y contraseña. | |
| **Actores:** Todos los usuarios del sistema | |
| **Referencia:** | |
| **Resumen:** El usuario se autentica con los datos ingresados y se buscará si se encuentra coincidencia con los datos guardados en la base de datos. | |
| **Incluye:** | |
| **Precondiciones:** El usuario tiene que estar registrado en el sistema. El usuario no debe tener sesión iniciada en el sistema. | |
| **Postcondiciones:** El usuario queda con la sesión iniciada. | |
| **Curso Normal** | |
| **Acción de los actores** | **Respuesta del sistema** |
| 1- El usuario ingresa a la pantalla de Login, haciendo clic en Iniciar Sesión. | 2- El sistema muestra la pantalla de Login. |
| 3- El usuario ingresa su email y contraseña, luego elige la opción “Ingresar”. | 4- El sistema valida que el email ingresado tenga el formato correcto y que se haya ingresado una contraseña. |
|  | 5- El sistema encripta la contraseña ingresada y busca en la base de datos si hay coincidencia con los datos ingresados por el usuario. |
|  | 6- El sistema verifica que el usuario encontrado tenga permisos para loguearse |
|  | 7- El sistema guarda en la bitácora el inicio de sesión del usuario |
|  | 8- El sistema verifica que el usuario logueado no tenga permisos para chequear los Dígitos Verificadores. |
|  | 9- El sistema redirige al usuario al menú principal de BuyMotors |
| **Alternativas** | |
|  | 4.1- Los datos ingresados no cumplen con los requerimientos de validez de datos, el caso de uso vuelve al paso 3. |
|  | 5.1- El sistema no encuentra coincidencia con los datos ingresados por el usuario, el caso de uso vuelve al paso 3 |
|  | 8.1- El sistema encuentra que el usuario logueado tiene permisos para chequear los Dígitos Verificadores.  8.2- El sistema verifica la integridad de los datos.  8.3- En caso de encontrar inconsistencia de datos, se le muestra al usuario un mensaje indicando la anomalía. |

## Ciclo de vida del proceso de Login, dividido en capas

### Interfaz de usuario (UI)





**Login.aspx**

<%@ Page Title="Iniciar sesión" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="Login.aspx.cs" Inherits="BuyMotors.Account.Login" Async="true" %>

<asp:Content runat="server" ID="BodyContent" ContentPlaceHolderID="MainContent">

<h2><%: Title %>.</h2>

<div class="row">

<div class="col-md-8">

<section id="loginForm">

<div class="form-horizontal">

<h4>Utilice una cuenta local para iniciar sesión.</h4>

<hr />

<asp:PlaceHolder runat="server" ID="ErrorMessage" Visible="false">

<p class="text-danger">

<asp:Literal runat="server" ID="FailureText" />

</p>

</asp:PlaceHolder>

<div class="form-group">

<asp:Label runat="server" AssociatedControlID="Email" CssClass="col-md-2 control-label">Correo electrónico</asp:Label>

<div class="col-md-10">

<asp:TextBox runat="server" ID="Email" CssClass="form-control" TextMode="Email" />

<asp:RequiredFieldValidator runat="server" ControlToValidate="Email"

CssClass="text-danger" ErrorMessage="El campo de correo electrónico es obligatorio." />

</div>

</div>

<div class="form-group">

<asp:Label runat="server" AssociatedControlID="Password" CssClass="col-md-2 control-label">Contraseña</asp:Label>

<div class="col-md-10">

<asp:TextBox runat="server" ID="Password" TextMode="Password" CssClass="form-control" />

<asp:RequiredFieldValidator runat="server" ControlToValidate="Password" CssClass="text-danger" ErrorMessage="El campo de contraseña es obligatorio." />

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<asp:Button runat="server" OnClick="LogIn" Text="Iniciar sesión" CssClass="btn btn-default" />

</div>

</div>

</div>

<p>

<asp:HyperLink runat="server" ID="RegisterHyperLink" ViewStateMode="Disabled">Registrarse como usuario nuevo</asp:HyperLink>

</p>

</section>

</div>

</div>

</asp:Content>

**Login.aspx.cs**

using System;

using BuyMotors.BL;

using BuyMotors.BE;

using BuyMotors.UI;

namespace BuyMotors.Account

{

public partial class Login : BasePage

{

protected string mensajeError;

protected void Page\_Load(object sender, EventArgs e)

{

if (UsuarioLogueado != null)

{

SalirDeLaPagina(false);

}

}

protected void LogIn(object sender, EventArgs e)

{

if (IsValid)

{

Usuario usuarioLogueado = UsuarioManager.Login(Email.Text, Password.Text);

if (usuarioLogueado != null)

{

UsuarioLogueado = usuarioLogueado;

BitacoraManager.Grabar(usuarioLogueado, "Login ok");

if (UsuarioManager.TienePermiso(usuarioLogueado, Permisos.CHEQUEO\_DV))

{

List<string> mensajesDeError = new List<string>();

Session["MostrarMensajeDV"] = !IntegridadDatos.Chequear(mensajesDeError);

Session["MensajeDV"] = string.Format(@"Error de integridad de datos:\n{0}", string.Join(@"\n", mensajesDeError));

}

SalirDeLaPagina();

}

else

{

// El login falló

FailureText.Text = "Intento de inicio de sesión no válido";

ErrorMessage.Visible = true;

}

}

}

private void SalirDeLaPagina()

{

string urlRegreso = Request.QueryString["ReturnUrl"];

if (!string.IsNullOrEmpty(urlRegreso))

{

Response.Redirect(urlRegreso);

}

else

{

// No hay Url especificada, se lo manda a la página principal

Response.Redirect("../Default.aspx");

}

}

}

}

**BasePage.cs**

using BuyMotors.BE;

using System.Web.UI;

namespace BuyMotors.UI

{

public class BasePage : Page

{

public Usuario UsuarioLogueado

{

get

{

return Session["UsuarioLogueado"] != null ? (Usuario)Session["UsuarioLogueado"] : null;

}

set

{

Session["UsuarioLogueado"] = value;

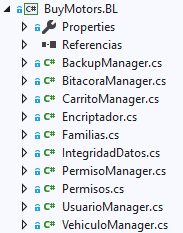
}

}

}

}

### Lógica de negocio (BL)



**UsuarioManager.cs**

using BuyMotors.BE;

using BuyMotors.DAL;

using System;

using System.Collections.Generic;

namespace BuyMotors.BL

{

public class UsuarioManager

{

public static Usuario Login(string email, string contrasenia)

{

try

{

Usuario usuario = UsuarioMapper.Obtener(email, Encriptador.Encriptar(contrasenia));

if (usuario != null && TienePermiso(usuario, Permisos.LOGIN))

{

return usuario;

}

}

catch (Exception ex)

{

Log.Log.Grabar(ex);

}

return null;

}

}

}

**Encriptador.cs**

using System;

using System.Text;

using System.Security.Cryptography;

namespace BuyMotors.BL

{

public static class Encriptador

{

private const string \_salt = "b5N3pWkRwzJH";

public static string Encriptar(string texto)

{

try

{

byte[] data = Encoding.ASCII.GetBytes(texto + \_salt);

data = new SHA256Managed().ComputeHash(data);

return Convert.ToBase64String(data);

}

catch (Exception ex)

{

Log.Log.Grabar(ex);

return "";

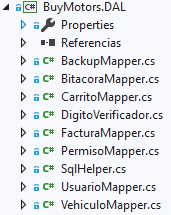
}

}

}

}

### Acceso a datos (DAL)



**UsuarioMapper.cs**

using BuyMotors.BE;

using System;

using System.Data;

using System.Data.SqlClient;

namespace BuyMotors.DAL

{

public class UsuarioMapper

{

public static Usuario Obtener(string email, string contrasenia)

{

string query = "SELECT TOP 1 Id, Nombre, Apellido, Telefono FROM Usuario " +

"WHERE Email = @email AND Password = @contrasenia";

SqlParameter[] parameters = new SqlParameter[]

{

new SqlParameter("@email", email),

new SqlParameter("@contrasenia", contrasenia)

};

DataTable table = SqlHelper.Obtener(query, parameters);

if (table != null && table.Rows.Count > 0)

{

int id = int.Parse(table.Rows[0]["Id"].ToString());

return new Usuario

{

Id = id,

Nombre = table.Rows[0]["Nombre"].ToString(),

Apellido = table.Rows[0]["Apellido"].ToString(),

Telefono = table.Rows[0]["Telefono"].ToString(),

Email = email,

Permisos = PermisoMapper.ObtenerPorUsuario(id)

};

}

return null;

}

}

}

**SqlHelper.cs**

using System;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Configuration;

namespace BuyMotors.DAL

{

class SqlHelper

{

private const string CONN\_STRING\_PRINCIPAL\_KEY = "Principal";

private const string CONN\_STRING\_BITACORA\_KEY = "Bitacora";

private const string CONN\_STRING\_MASTER\_KEY = "Master";

public static DataTable Obtener(string query, SqlParameter[] parameters)

{

return Obtener(query, parameters, CONN\_STRING\_PRINCIPAL\_KEY);

}

public static DataTable ObtenerBitacora(string query, SqlParameter[] parameters)

{

return Obtener(query, parameters, CONN\_STRING\_BITACORA\_KEY);

}

private static DataTable Obtener(string query, SqlParameter[] parameters, string connStringKey)

{

string connString = ConfigurationManager.ConnectionStrings[connStringKey].ConnectionString;

try

{

SqlCommand cmd = new SqlCommand

{

Connection = new SqlConnection

{

ConnectionString = connString

},

CommandText = query,

CommandType = CommandType.Text,

};

if(parameters != null && parameters.Count() > 0)

{

cmd.Parameters.AddRange(parameters);

}

DataSet ds = new DataSet();

cmd.Connection.Open();

using (SqlDataAdapter dataAdapter = new SqlDataAdapter())

{

dataAdapter.SelectCommand = cmd;

dataAdapter.Fill(ds);

}

cmd.Connection.Close();

return ds.Tables[0];

}

catch (Exception ex)

{

Log.Log.Grabar(ex);

return null;

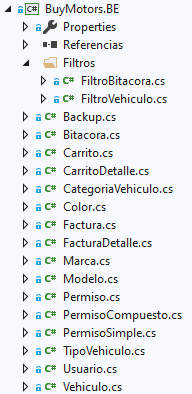
}

}

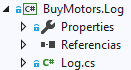
}

}

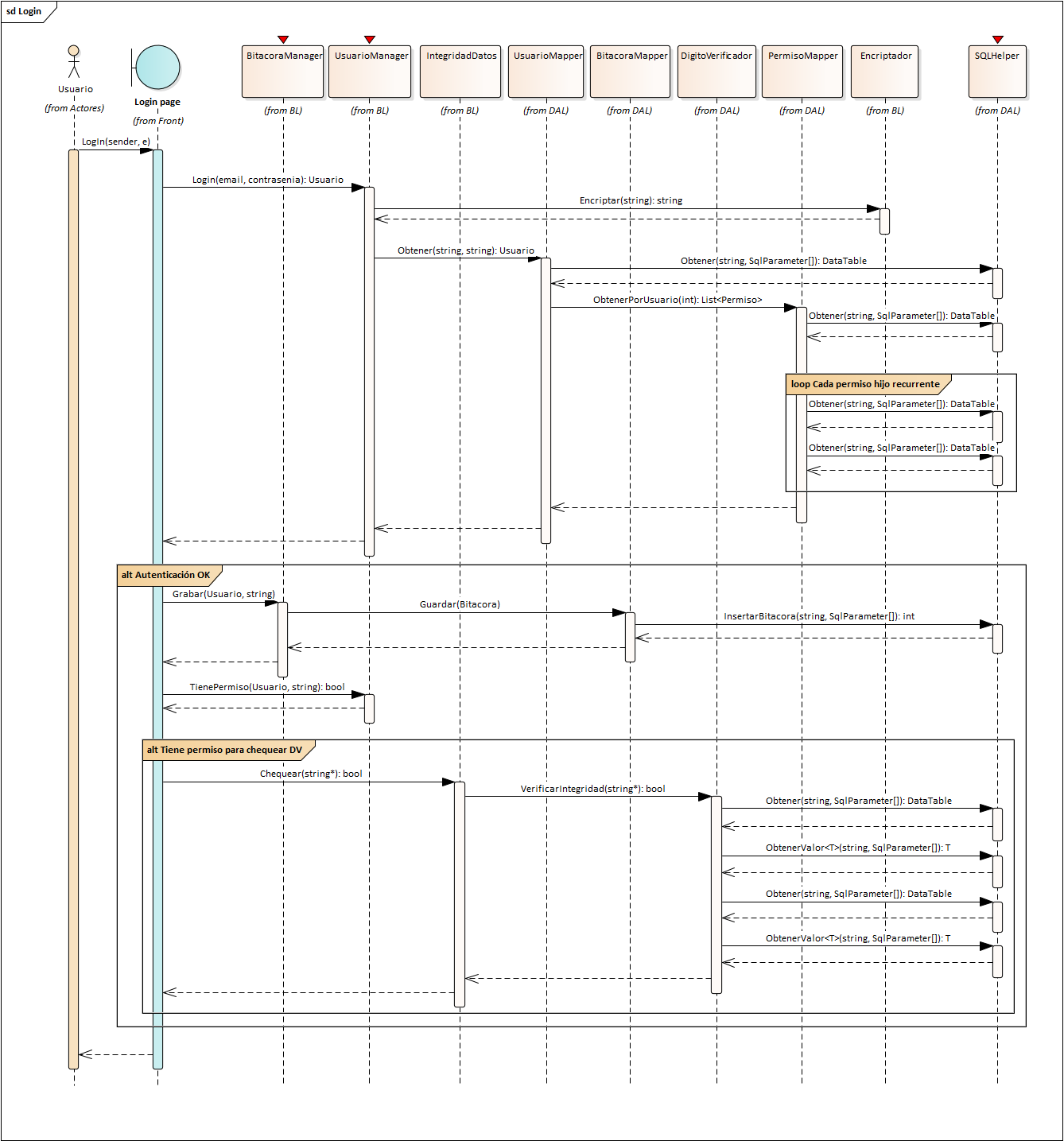
### Entidades (BE)



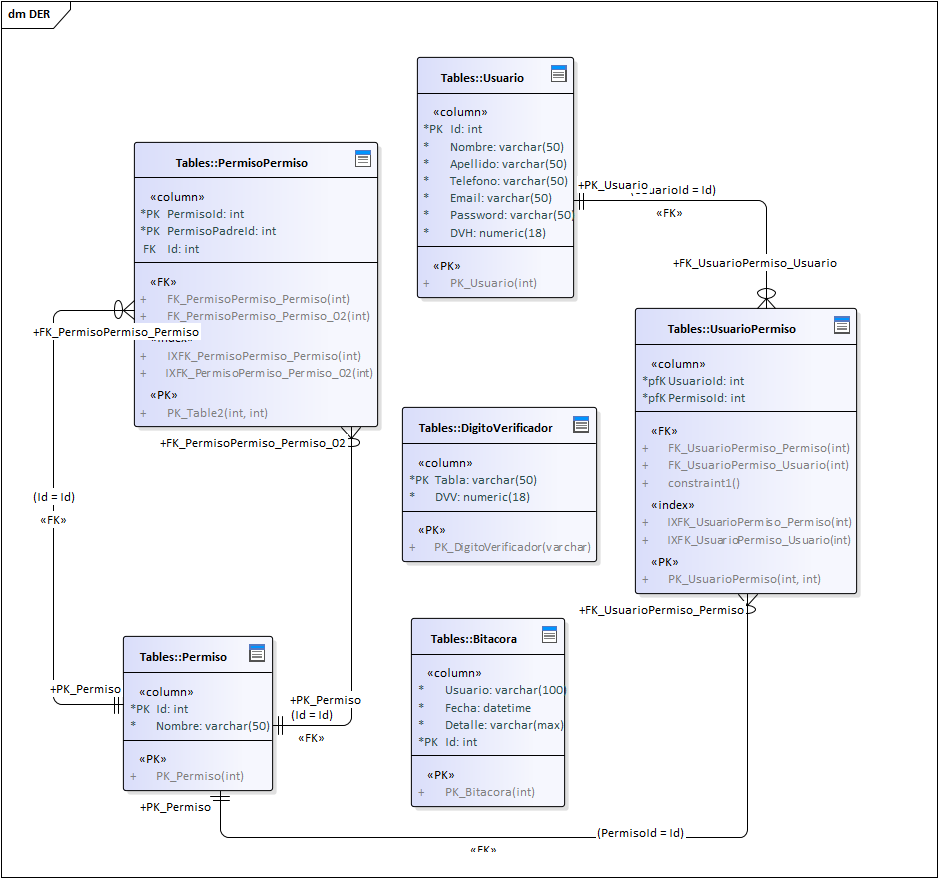
### Logueo (Log)



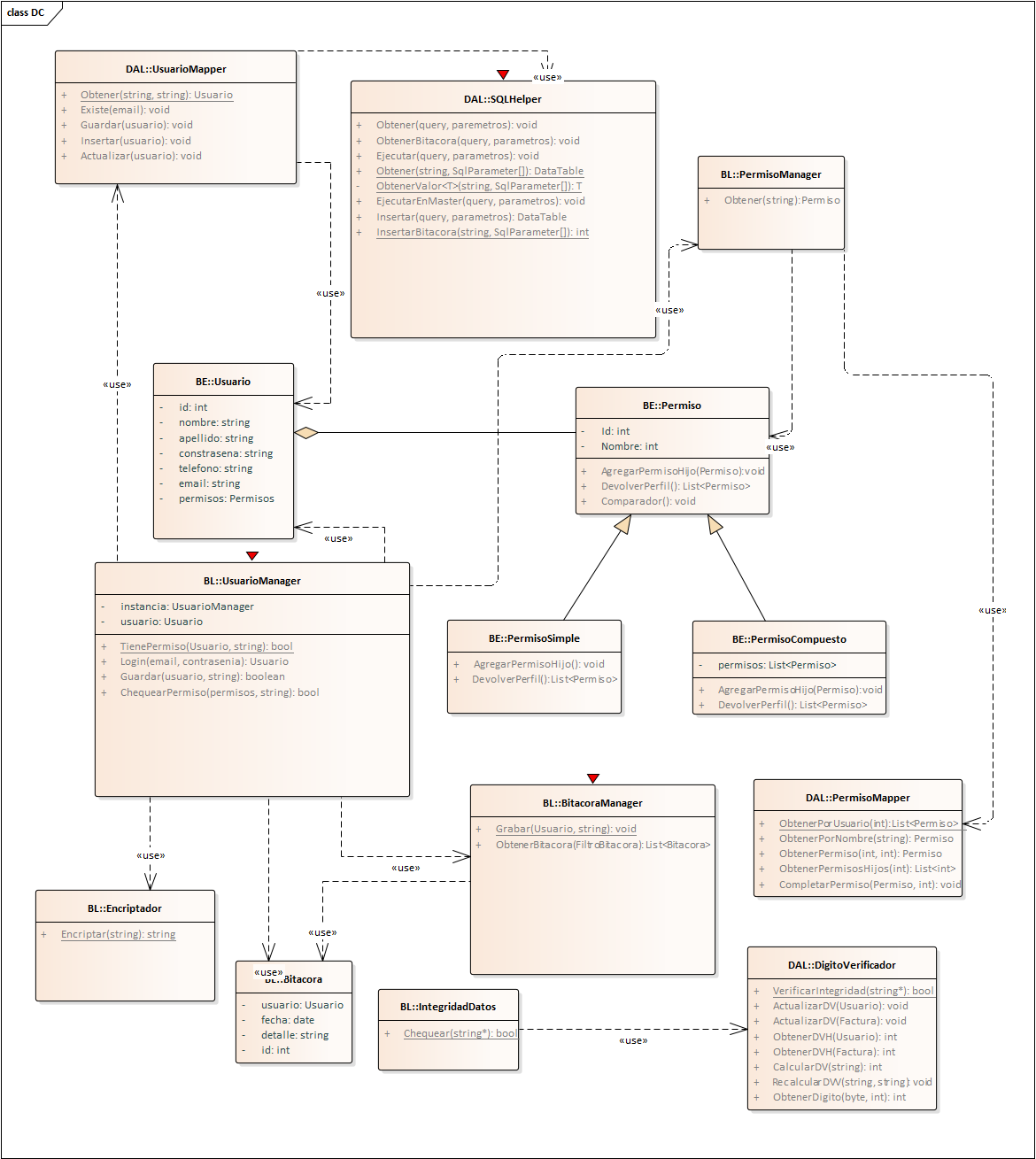
## Diagrama de secuencia



## DER



## Diagrama de clases



# Gestión de bitácoras

## 

## Objetivo

Se desea tener un registro auditable de los sucesos importantes del sistema. Este registro brindará una trazabilidad de los eventos registrados por el sistema, el cual podrá ser utilizado para fines legales y de seguridad.

## 

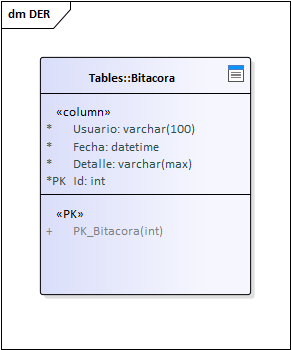
## Descripción

El sistema registrará en la base de datos, en una entidad dedicada exclusivamente a la bitácora del sistema, los eventos ocurridos en el sistema. Los eventos que se cubrirán son:

* Login: Inicio de sesión.
* Logout: Cierre de sesión.
* Creación de backup
* Restauración de backup

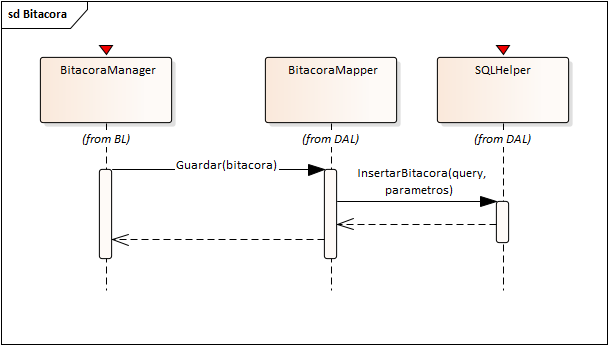
## 

## DER

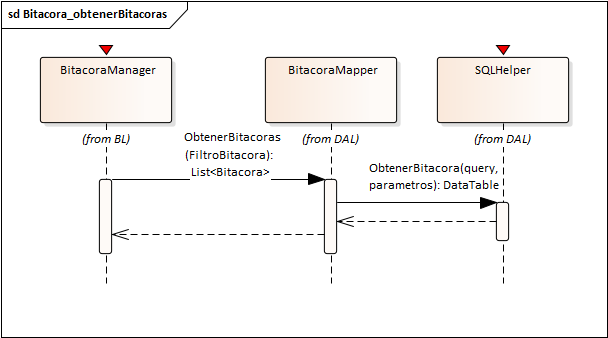


## Diagrama de secuencia

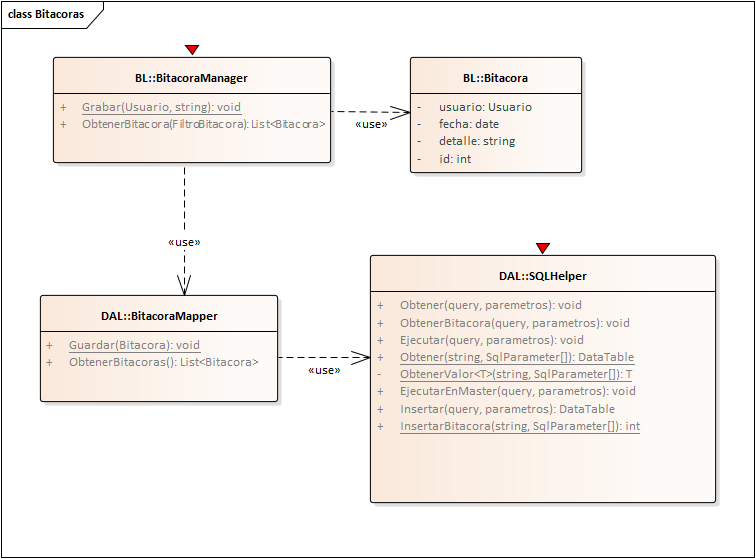
### Guardar Bitácora



### Obtener Bitácora

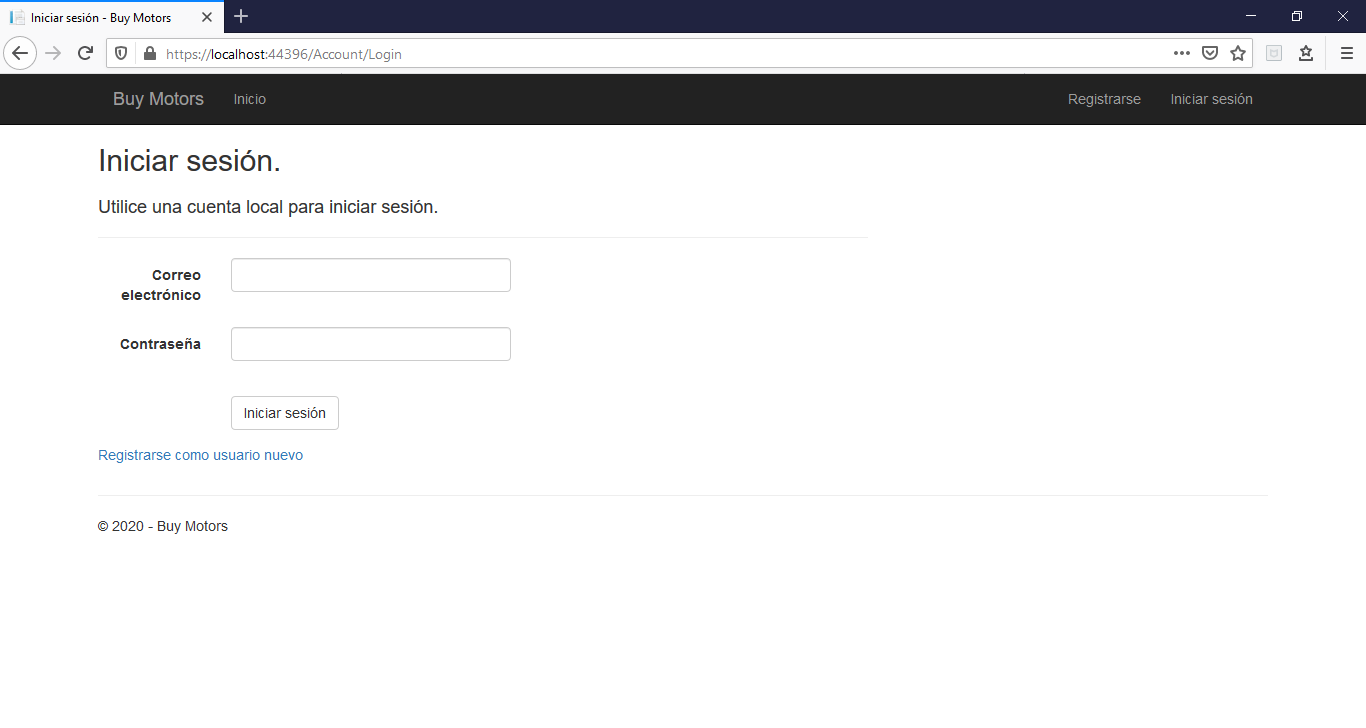


## Diagrama de clases

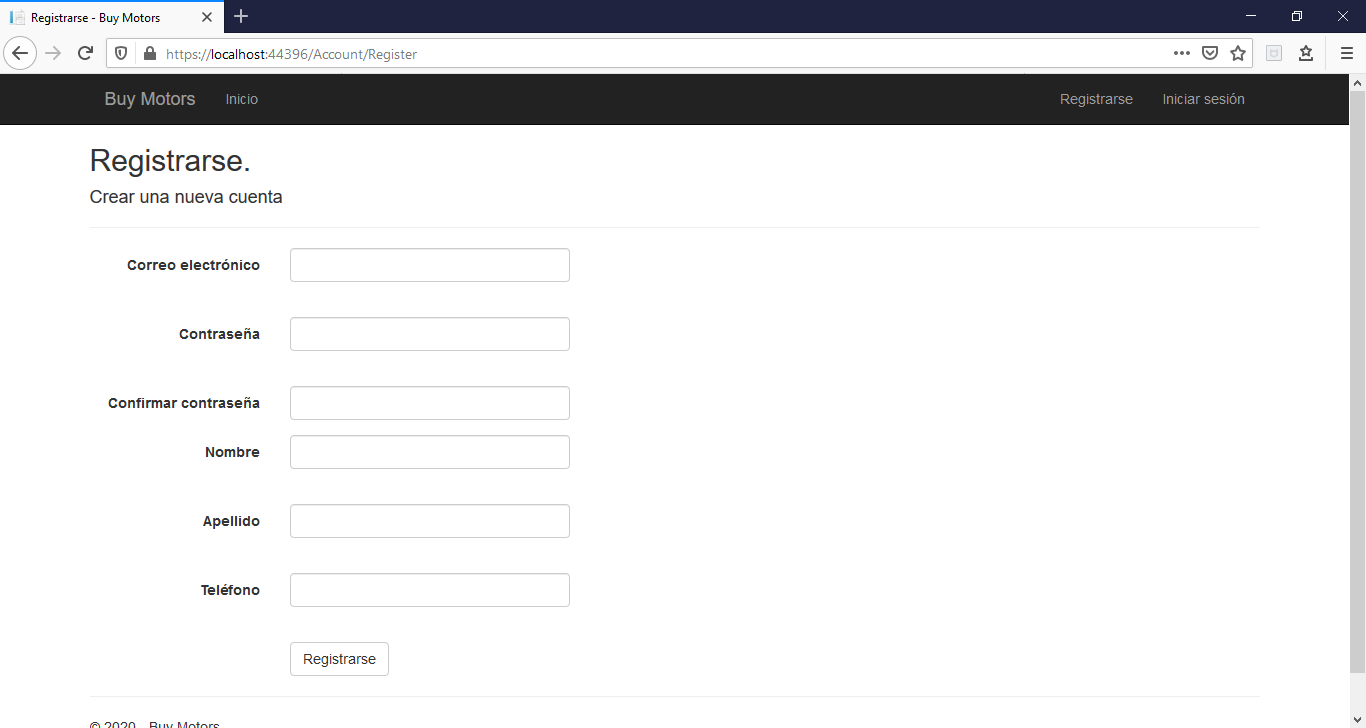


# Capturas de pantalla del sistema

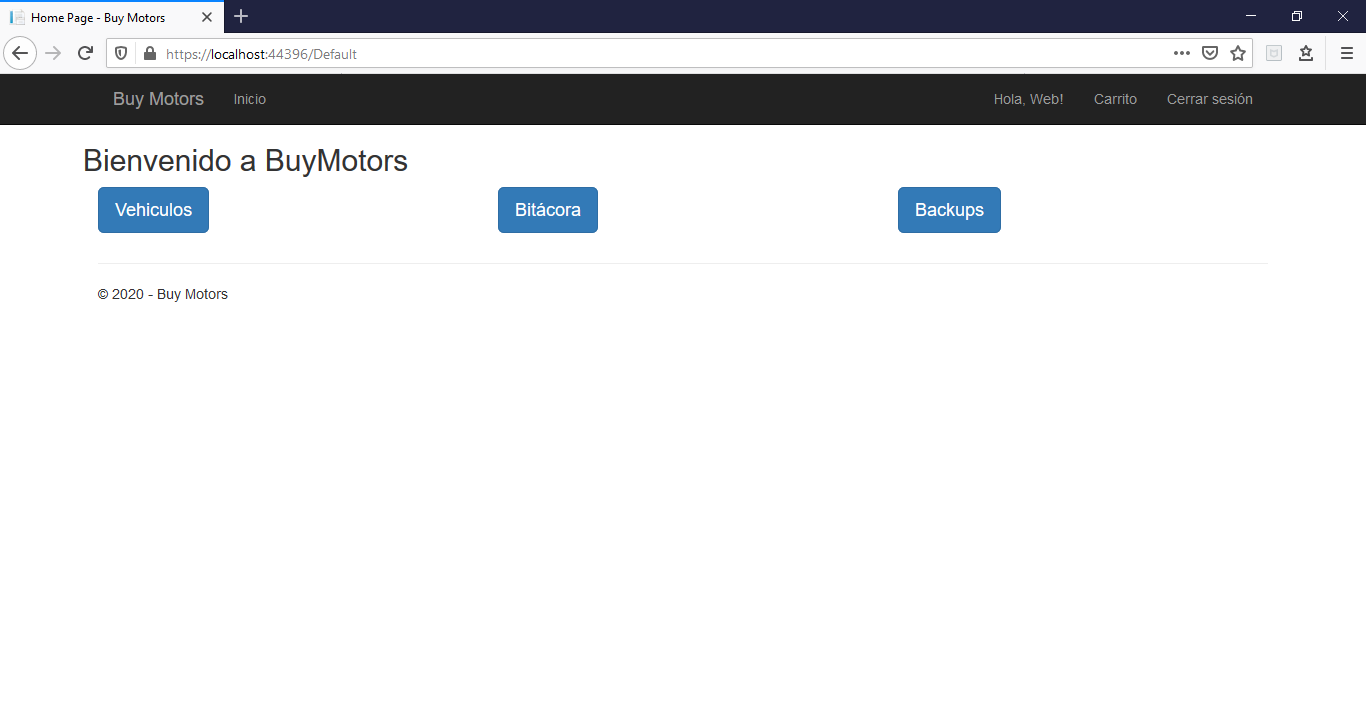
## Login



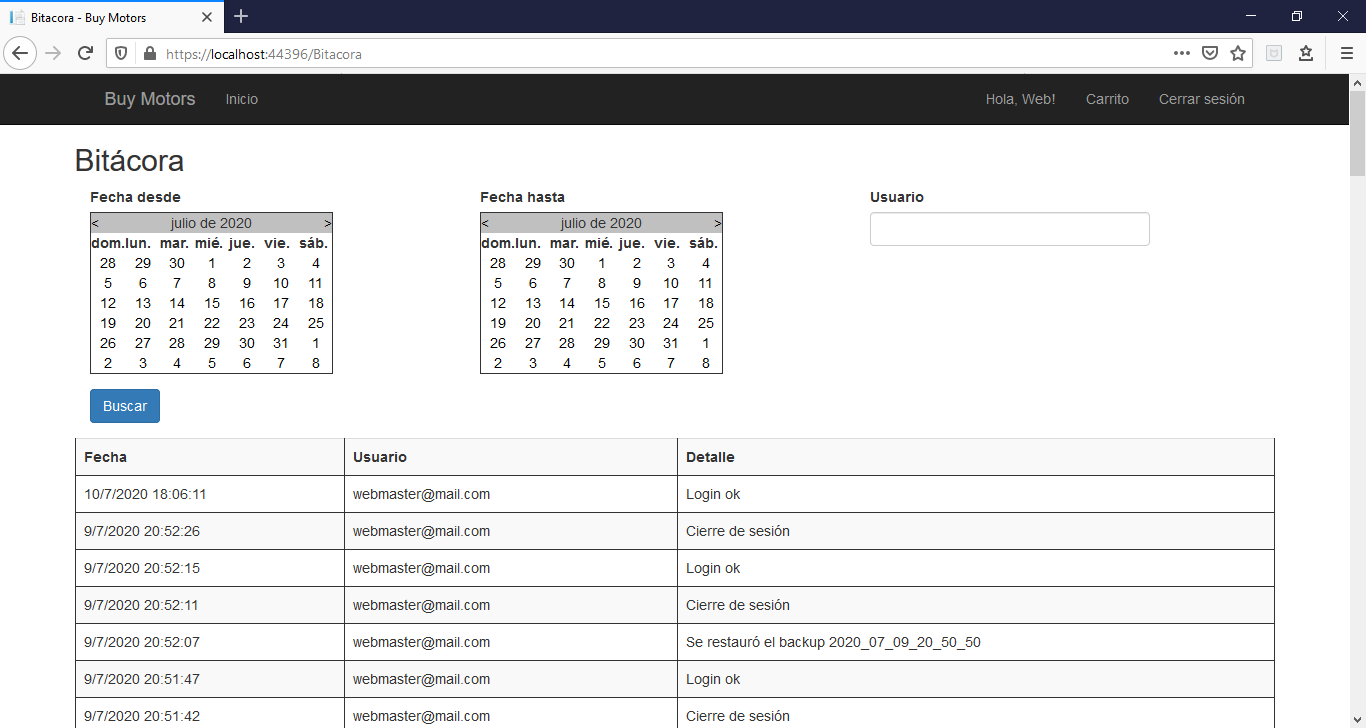
## Registro de nuevo usuario



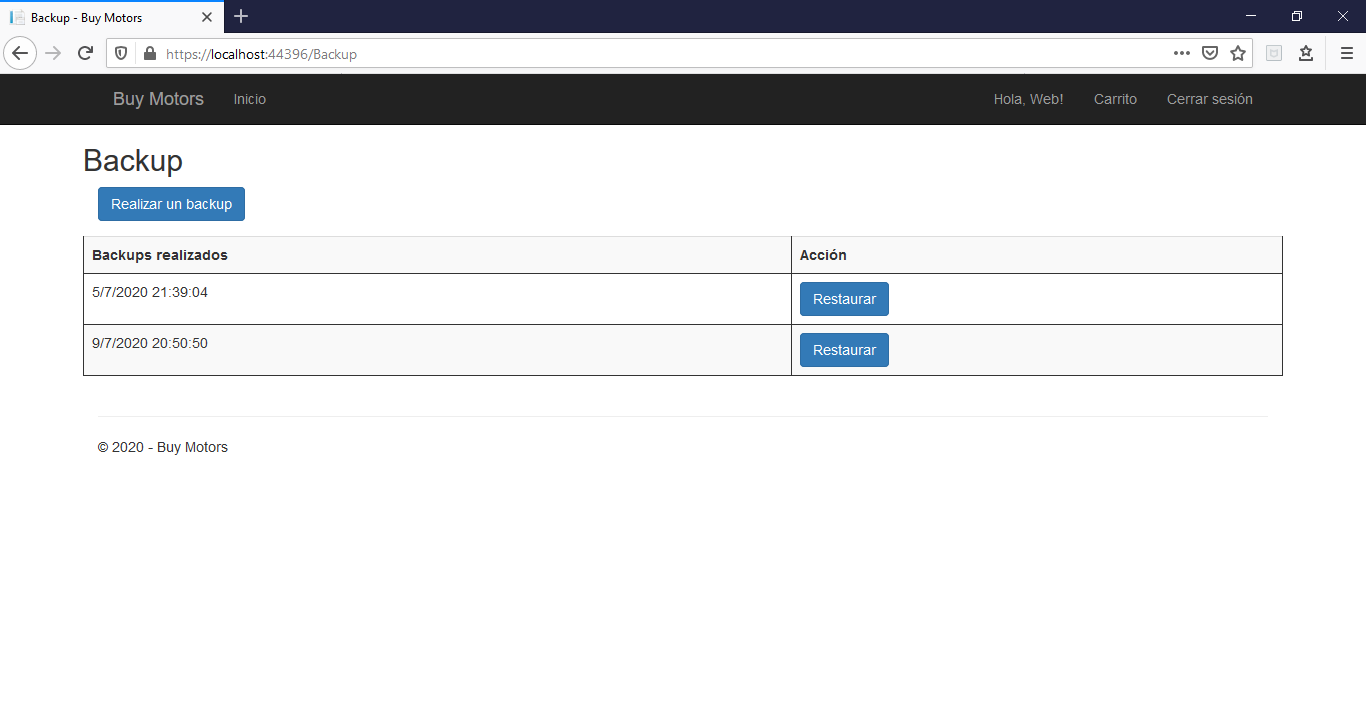
## Pantalla principal (con webmaster logueado)



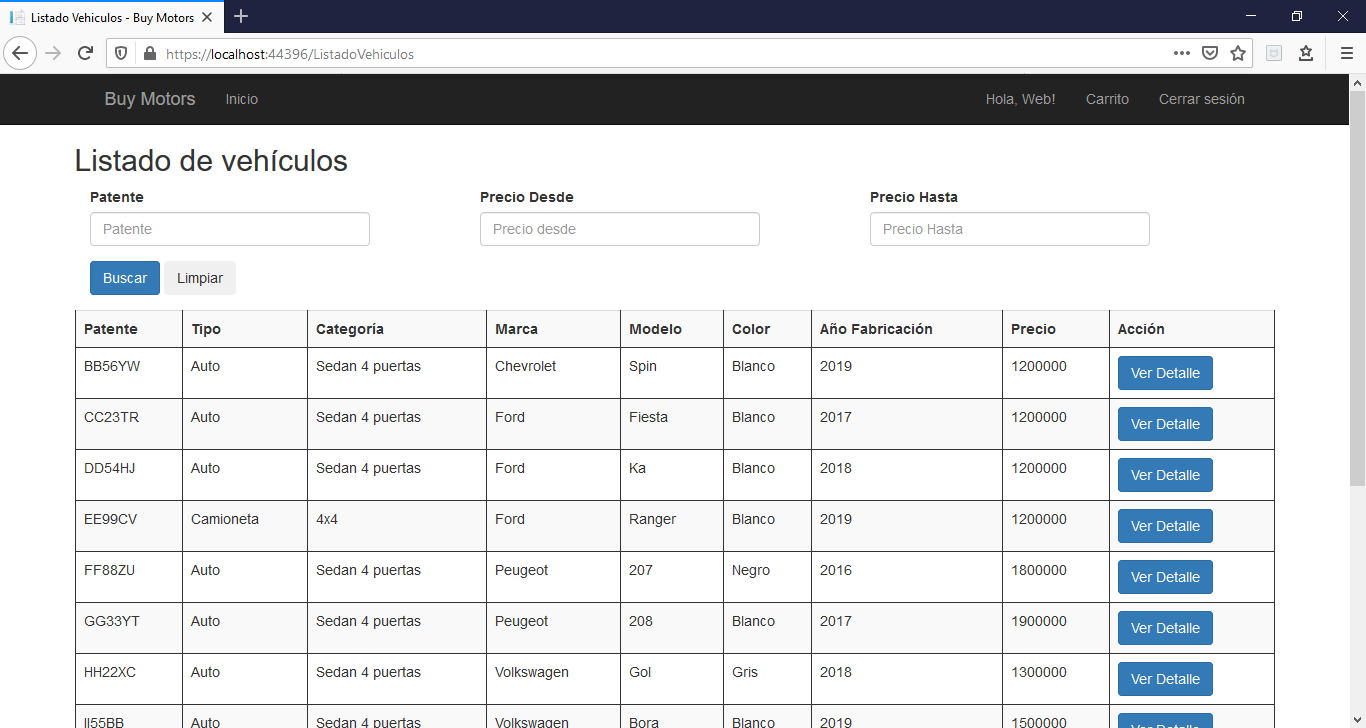
## Bitácora



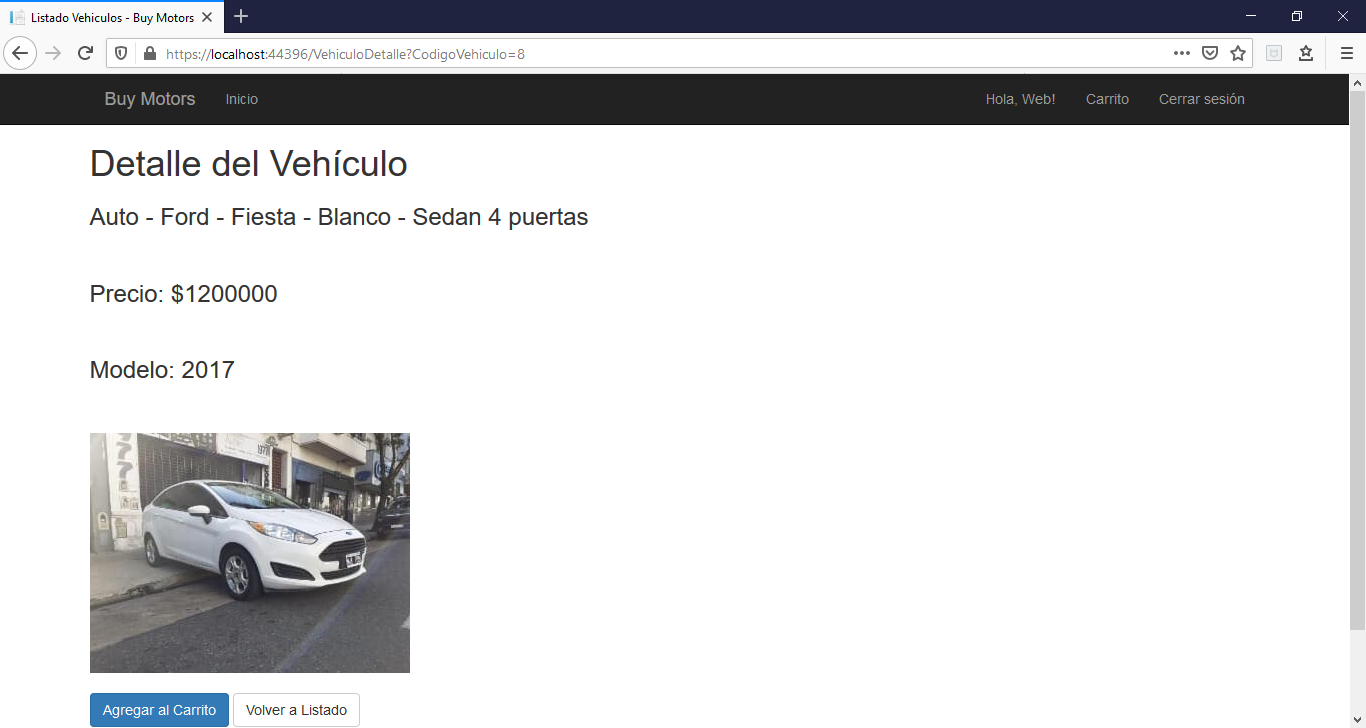
## Backup



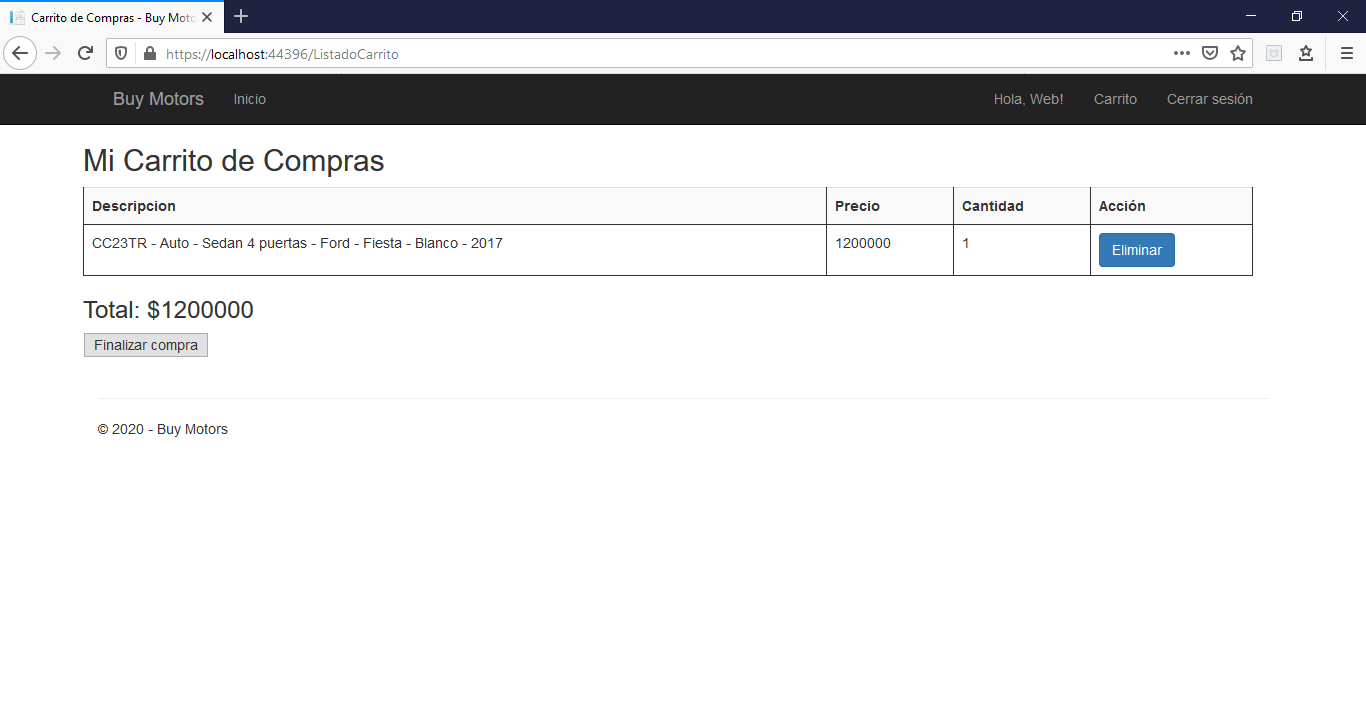
## Listado de Vehículos



## Detalle de un vehículo



## Carrito de compras



# Anexo

## Políticas de programación

### Olvido de contraseña

Si un usuario no recordase su contraseña al intentar loguearse al sistema, tendrá la opción de restablecerla mediante un enlace que se muestra en la misma pantalla de login. Al hacerle clic, se envía un email al usuario el cual contendrá un link. Al ingresar a ese link, el sistema mostrará los campos para introducir su nueva contraseña y, luego de completar esos datos, el usuario quedará listo para loguearse con su nueva contraseña.

Para evitar que un usuario malicioso intente restablecer la contraseña de otro usuario, al momento de solicitar el restablecimiento de la contraseña se generará un token alfanumérico aleatorio que quedará guardado en el sistema asociado a ese usuario. Este token estará incluido como parámetro en el link que recibirá el usuario por mail, y se validará cuando se acceda a ese link. Además, para evitar un ataque por fuerza bruta, al ingresar a esta página de restablecimiento de contraseña, si el token enviado no coincide con el almacenado, se invalidará el token generado previamente y el usuario deberá solicitar un nuevo link de restablecimiento si quiere restablecer su contraseña.

### Intentos fallidos de Login

Si un usuario llegase a introducir mal su contraseña 3 veces consecutivas al intentar ingresar al sistema, el mismo se bloqueará y no podrá loguearse aunque luego introduzca bien su contraseña. Para desbloquear este usuario, el mismo estará obligado a restablecer su contraseña como se indica en la sección anterior.

Este mecanismo se implementa como una seguridad para evitar que personas maliciosas intenten ingresar al sistema con el usuario de otra persona utilizando fuerza bruta, es decir, probando con todas las combinaciones posibles de contraseñas hasta acercar a la correcta.

### Backups

Los backups se realizarán de forma manual por el webmaster, dejando la frecuencia de generación de los mismos a su criterio, aunque se recomienda que se haga uno por semana como mínimo. Todos los backups que se realicen serán totales.

## Estrategias de programación

### Capas del sistema

El sistema se encuentra dividido internamente en 5 capas, las cuales se detallan a continuación:

* Interfaz de usuario (UI): Aquí se desarrollan todas las páginas aspx con las que el usuario interactúa.
* Lógica de negocios (BL): La interfaz de usuario se comunica con esta capa para solicitar o enviar información. Aquí se aplican los algoritmos relacionados con la funcionalidad del sistema en sí.
* Acceso a datos (DAL): Esta capa se encarga de hacer de nexo entre la lógica de negocios y el almacén de datos, que en este caso se trata de una base de datos SQL Server. Arma y ejecuta sentencias SQL para hacer operaciones sobre la base de datos.
* Entidades (BE): Contiene solamente clases con propiedades que sirven para armar objetos utilizados para pasar información entre las 3 capas anteriores.
* Logueo (Log): Contiene una clase encargada de registrar los errores que ocurren en el sistema dentro de un archivo de texto. Todas las capas pueden utilizarla.

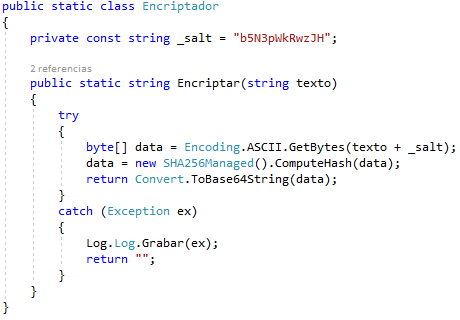
### Uso de patrones

Se utiliza el patrón de diseño Composite para representar el sistema de permisos que tiene un usuario.

Para el resto del sistema, no se utiliza ningún otro patrón estándar.

### Encriptación de la contraseña

Para encriptar la contraseña ingresada por el usuario, y así poder validarla contra la que está guardada en la base de datos utilizando el mismo algoritmo, se utiliza el siguiente bloque de código:



En primer lugar, se puede observar que al texto recibido para encriptar (la contraseña) se le concatena un texto fijo que se generó aleatoriamente, que está guardado en la variable “\_salt”. El objetivo de esto es aumentar la seguridad, evitando que un posible atacante que haya adivinado el algoritmo utilizado, y pueda obtener el mismo resultado que esta función; de esta manera, no podrá hacerlo a menos que sepa el valor exacto de la variable \_salt, la cual es secreta.

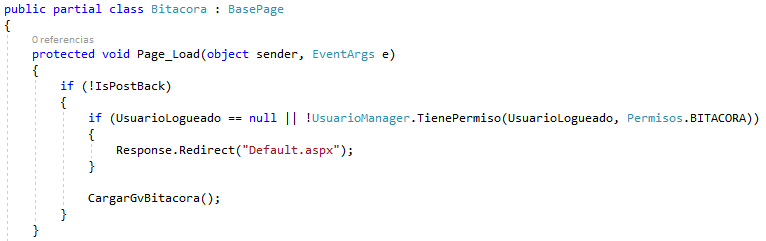
Con esta concatenación, se obtiene un array de bytes utilizando en encoding ASCII. Este array de bytes luego es utilizado para hacerle un hash (o codificación de una sola vía) usando el algoritmo SHA256Managed, el cual es de público conocimiento, dando como resultado otro array de bytes.

Por último, se toma el último array de bytes hasheado y se lo convierte a un string en Base64 para que sea un texto más manejable y se lo devuelve como resultado de la función de encriptación.

### Restricción de acceso por URL

Las páginas que no son públicas y contienen información que la deben ver solamente ciertos tipos de usuarios (como la bitácora, por ejemplo), poseen una validación de permisos en el evento Page Load. Es decir, al momento de cargar la página, se verifica que el usuario logueado tenga permisos para ver esa página. En caso de no tenerlo, se lo redirige inmediatamente a la página principal.

A continuación, se muestra un bloque de código que ejemplifica lo explicado:



### Integridad de datos de la BD

Con el objetivo de preservar la integridad de los datos guardados en la base de datos, se implementa un mecanismo de dígito verificador en las tablas Usuario y FacturaCabecera. Este mecanismo permitirá saber si un dato fue modificado, agregado o eliminado por fuera del sistema.

Para ello, las tablas mencionadas contienen una columna adicional llamada DVH, la cual contiene una cadena de caracteres alfanuméricos calculada a partir de cierto algoritmo aplicado a los datos de las demás columnas de ese registro, de manera tal que una modificación en los datos de esas columnas cambiará por completo la cadena calculada en DVH. Además, el sistema cuenta con otra tabla llamada DigitoVerificador que representa el dígito verificador vertical, es decir, el cálculo de una nueva cadena alfanumérica a partir de todos los DVH de una tabla. Así, al agregar o quitar un registro a esas tablas, el dígito verificador vertical calculado cambiará, poniendo en evidencia que la base de datos fue manipulada externamente.