



Modelos computacionales de Gestión Administrativa

Trabajo práctico

Integrantes:

Fabrizio Crivella

Lucas Toffolon

Nicolas Dacunda

Federico Arango

Profesor:

Claudio Milio



UNIVERSIDAD ABIERTA INTERAMERICANA

Facultad de tecnología informática

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

EMPRESA

 UNIVERSIDAD ABIERTA INTERAMERICANA Facultad de tecnología informática	
Materia: Modelos computacionales de Gestión Administrativa	Docente: Milio, Claudio
Alumnos: Crivella, Toffolon, Dacunda, Arango	

1. Empresa

1.1 Introducción

Como el nombre lo indica, el servicio que ofrecemos fusiona la logística, almacenes y un control de abastecimiento de productos con dispositivos inteligentes y servicios que nos interconecta con nuestros clientes. Estos dispositivos suelen estar conectados en las góndolas de nuestros clientes, y permite calcular el stock total y el stock faltante para ese producto permitiendo controlar de forma eficaz y eficiente todo el proceso de cadena de suministros.

1.2 ¿Por qué eligieron internet?

Elegimos internet para maximizar nuestros servicios y llegar con la mejor calidad para nuestros clientes. Internet permite interconectarse por medio de nuestro microservicio con cada uno de nuestros clientes y centrandonos en la calidad que conforman la cadena de suministros: Almacenaje, distribución y abastecimiento haciendo el uso de internet.

1.3 ¿Qué beneficios tiene para el cliente?

Algunas de las cosas que podrás administrar son:

- Control de stocks/existencias propias o de terceros.
- Planificar viajes y transporte de forma más eficiente.
- Reducir gastos de viaje.
- Agilizar los procesos de cobranza.
- Tener un mejor control de los cobros o servicios brindados a clientes.
- Realizar un seguimiento constante de los envíos.
- Abastecimiento seguro por producto

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

1.4 ¿Qué beneficios tiene para la empresa?

Este nuevo concepto es aplicable a todas las áreas que forman parte de una cadena de suministro.

Monitoreo de vehículos: Registra el ingreso y salida de vehículos de la empresa. Asimismo, se realiza un seguimiento del trayecto que sigue hasta el punto de entrega y el estado del vehículo en todo momento.

1.5 ¿Cómo se hace eso?

Por medio de un GPS y sensores en los vehículos. Esto permitirá no sólo hacer un seguimiento del lugar y ruta del vehículo, sino que también podremos tener información en tiempo real del estado del vehículo y de la mercadería.

Manejo de inventarios: Se obtiene un inventario inteligente, registrando, monitoreando y evaluando constantemente los productos que se encuentran en los almacenes de cada cliente. Para ello, requerirá sensores que permitan llevar un conteo eficiente de la mercadería presente y la faltante.



UNIVERSIDAD ABIERTA INTERAMERICANA

Facultad de tecnología informática

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

NEGOCIO

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

2 Negocio

2.1 Requerimientos de sistema

Se desarrollará una plataforma de logística online del tipo B2B destinada a la provisión de medicamentos e insumos para veterinarias. El relevamiento de información se llevó a cabo mediante la realización de entrevistas con los socios de la compañía. De allí surgieron los siguientes requerimientos con los que el sistema deberá cumplir:

- La plataforma deberá ser accesible desde los browsers más populares, sin requerir la instalación de ningún tipo de software adicional.
- El sistema debe permitir llevar el control del stock de forma automatizada, en base a las paramétricas realizadas por nuestros usuarios.
- En proceso de registro, el usuario administrador, solo debe solicitar a los clientes el nombre de la empresa, CUIT, dirección postal, teléfono y correo electrónico. Además, será necesaria la generación de un nombre de usuario y una contraseña, con la que el cliente iniciará sesión en la plataforma.
- El cliente deberá poder acceder al sistema para cargar o modificar el stock parametrizado que recibirá periódicamente.

El acceso a los datos sensibles, como el nombre de usuario y la contraseña, deberá estar restringido, siguiendo las técnicas que el equipo de análisis y desarrollo considere más adecuadas.

2.2 Seguridad

2.2.1 Funcionalidades incluidas

El sistema deberá ofrecer las siguientes funcionalidades:

- Formulario para autorregistro por parte de los clientes.

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

- Acceso al sistema mediante el ingreso de nombre de usuario y contraseña.
- Posibilidad de realizar copias de seguridad y recuperación de los datos contenidos en la base de datos.
- Cifrado de los datos sensibles almacenados en la base de datos.
- Control de integridad de las tablas de la base de datos que contengan datos sensibles.
- Bitácora con los registros de las operaciones realizadas en el sistema.
- Gestión de control de acceso mediante patentes y roles de seguridad.

2.2.2 Patentes incluidas en el sistema

Nombre de patente	Descripción
Bitácora	Administración y consulta de todos los eventos de sistema
Inventario	Administración de los artículos, agregar, editar o eliminar artículos existentes en el sistema.
Backup	Realizar backups de la base de datos principal.
Restore	Realizar restore de la base de datos principal del sistema ante desastres.
Carrito	Permite al usuario agregar, eliminar y editar productos del carrito de compras.

2.2.3 Familias incluidas en el sistema

Familia	Descripción
---------	-------------

 UNIVERSIDAD ABIERTA INTERAMERICANA Facultad de tecnología informática	
Materia: Modelos computacionales de Gestión Administrativa	Docente: Milio, Claudio
Alumnos: Crivella, Toffolon, Dacunda, Arango	

Web Master	Tiene todos los permisos correspondientes a todas las funciones del sistema.
Seguridad	Administra las funciones de Backup, restore y la bitácora del sistema.
Cliente	Administra el carrito de compras.
Operativo	Habilita la tarea de administrar el inventario de artículos en el sistema

2.2.4 Encriptación

En la tabla usuario utilizaremos el método de encriptación MD5 para la contraseña y para el resto de los campos usaremos AES.

Tabla	Campo	Tipo
Usuario	Contraseña	MD5
Usuario	NombreUsuario	AES
Bitácora	Descripción	AES
Familia	Nombre	AES

2.2.5 Control de integridad

El sistema realizará verificaciones de la integridad de la base de datos para detectar modificaciones en la misma por fuera del sistema. Para esto, se incorporará:

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

- Una clase llamada GestorIntegridad en la capa DAL encargada de los cálculos necesarios para obtener los dígitos verificadores verticales y horizontales de la base de datos.
- Dígito verificador horizontal en la base de datos
Los DVH (Dígito verificador horizontal) utilizarán la información del registro para calcular el dígito que será almacenado al final del mismo en una columna llamada “DVH”.
- Dígito verificador vertical en la base de datos.
Los DVV (Dígito verificador vertical) utilizarán los DVH de todos los registros de un tabla para calcular el dígito de la misma, el cual será almacenado en una tabla llamada DVH donde se registrará el nombre de la tabla y el DVV correspondiente a la misma.

La verificación de la base de datos se realizará al momento del login de un usuario. Si los cálculos del DVH y/o DVV no coinciden con los almacenados en los registros significa que los datos fueron manipulados. En este caso, cuando un usuario administrador inicie sesión en el sistema aparecerá en pantalla un mensaje indicando los registros y tablas que fueron corrompidos dándole la opción de aceptar los cambios (los DVH y DVV serán recalculados y almacenados) o descartarlos restaurando el último backup de la base de datos.

Las tablas cuya integridad será verificada son las siguientes:

Tabla
Bitácoras
Usuario
Patente
FamiliaPUatente

Materia: Modelos computacionales de Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

2.2.6 Copia de seguridad y restauración

El sistema permitirá al usuario administrador de configuración realizar copias de seguridad a demanda, para esto deberá contar con los privilegios necesarios.

Esto permitirá guardar una copia de la información del sistema para ser utilizada con múltiples propósitos, como por ejemplo restaurar la información original después de una eventual pérdida de datos o para poder migrar la información a otro servidor.

El Backup realizado estará particionado en varios archivos para garantizar una mayor seguridad, y evitar así que alguien pueda acceder a los datos de forma ilegal. El Backup se podrá guardar en la ruta que indique el usuario en el momento de generarlo. Los nombres de los archivos generados serán sugeridos por el sistema contando con su número de partición y la fecha del día, pero podrán ser cambiados por el usuario en el mismo momento.

El sistema también permitirá, al usuario administrador de configuración, realizar un Restore de sistema, en el cual se puede restaurar el mismo con los Backup generados en esta misma sección.

Cabe aclarar que el usuario que tenga permiso de realizar una gestión de Backup no podrá tener permiso para realizar una gestión de Restore y viceversa.

2.2.7 Bitácora

La bitácora registrará cada transacción realizada, al igual que los intentos de log-in, tanto exitosos como fallidos. Tendrá el usuario que realizó una operación, fecha y hora, y acción realizada. Como seguridad tendrá dígitos verificadores verticales y horizontales, los cuales se encontrarán encriptados en la base de datos. Cada operación se registrará al momento de realizarse. Cada registro en la bitácora tendrá un nivel de criticidad definido, que puede ser alto, medio o bajo. La misma se podrá visualizar tanto en la base de datos como directamente en el Sistema. Será accesible por el administrador del Sistema.



UNIVERSIDAD ABIERTA INTERAMERICANA

Facultad de tecnología informática

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

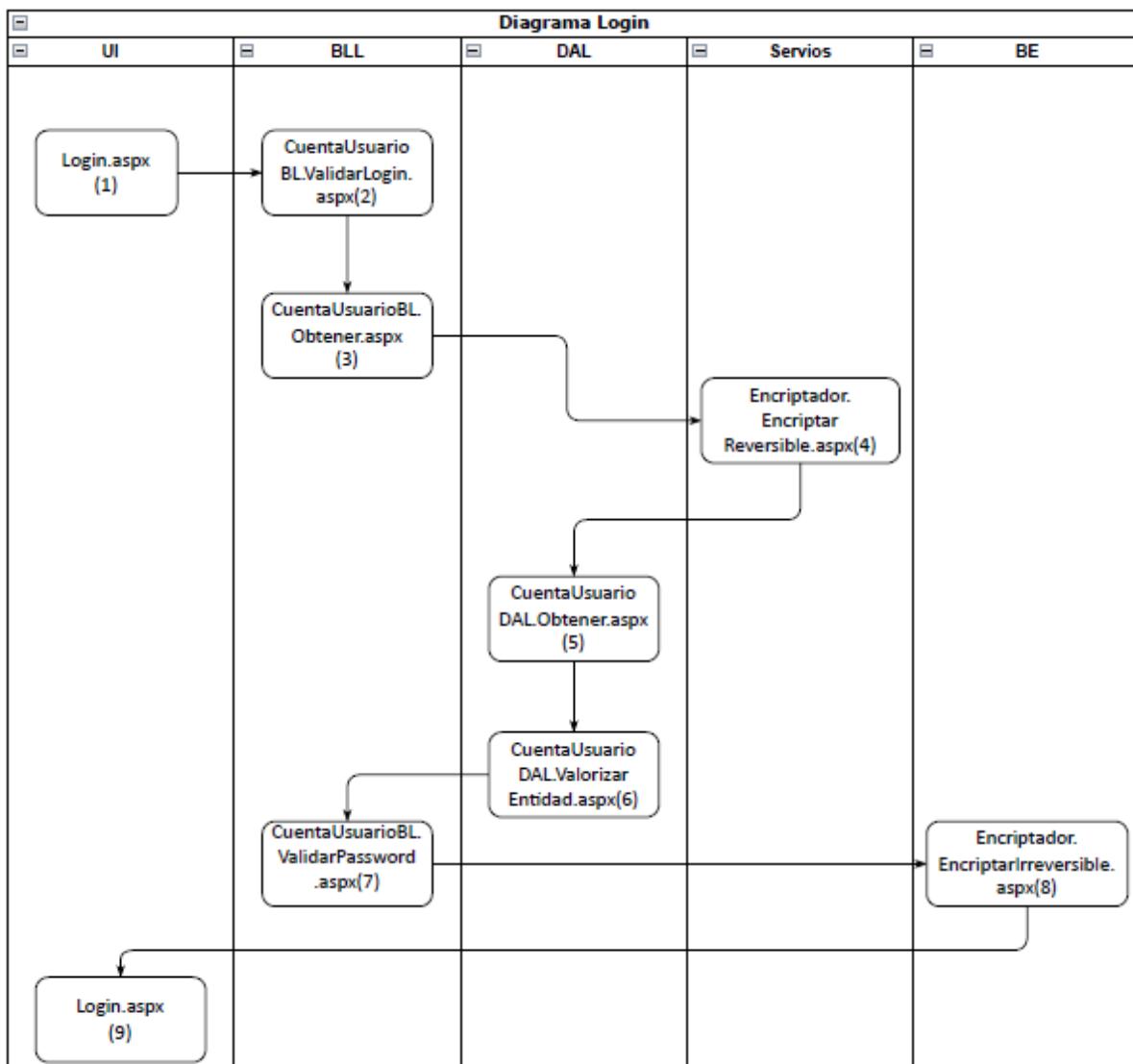
Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

2.3 Diagramas

2.3.1 Diagrama Login



(1)

```

try
{
    CuentaUsuario mCuentaUsuario = CuentaUsuarioBL.ValidatorLogin(txtUser.Value,
txtPassword.Value);
}

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

VerificarBase();
List<Familia> mFamilias = new List<Familia>();
mFamilias = CuentaUsuarioBL.ObtenerFamilias(mCuentaUsuario);
if (mFamilias.ElementAt(0).familia_nombre == "Administrador"){adminlogged = true;}
Session.Add("UsuarioLoggeado", true);
Session.Add("IDUsuario", mCuentaUsuario.Cuenta_usuario_id.ToString());
Session.Add("Usuario", mCuentaUsuario.Cuenta_usuario_username.ToString());
Session.Add("Familia", mFamilias.ElementAt(0).familia_nombre);
Session.Add("BaseCorrupta",false);

DVHBL.ValidarConsistenciaDVH();
DVVBL.ValidarConsistenciaDVV();
Response.Redirect("MenuPrincipalUI.aspx");
}

```

(2)

```

public static CuentaUsuario ValidarLogin(string pUsername, string pPassword)
{
    Encriptador mEncriptador = new Encriptador();
    CuentaUsuario mCuentaUsuario = Obtener(mEncriptador.EncriptarReversible(pUsername));
    Bitacora mRegistro = new Bitacora();
    mRegistro.cuenta_usuario_id = mCuentaUsuario.Cuenta_usuario_id;
    mRegistro.bitacora_fecha = DateTime.Now;
    mRegistro.bitacora_hora = DateTime.Now.TimeOfDay;
    if (mCuentaUsuario.Cuenta_usuario_id == 0 | mCuentaUsuario.cuenta_usuario_activa == 0)
    {
        mRegistro.cuenta_usuario_id = -1;
        mRegistro.bitacora_transaccion_id = 2;
        mRegistro.bitacora_criticidad = 3;
        BitacoraBL.Guardar(mRegistro);
        throw new UsuarioInexistenteException();
    }

    bool aux = ValidarPassword(mCuentaUsuario, mEncriptador.EncriptarIrreversible(pPassword));
    if (aux == false)
    {
        mCuentaUsuario.Cuenta_usuario_intentos_login += 1;
        mRegistro.cuenta_usuario_id = mCuentaUsuario.Cuenta_usuario_id;
        mRegistro.bitacora_transaccion_id = 3;
        mRegistro.bitacora_criticidad = 2;
    }
}

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

if (mCuentaUsuario.Cuenta_usuario_intentos_login == 3)
{
    string mNuevaClave = GenerarClaveAleatoria();
    mCuentaUsuario.Cuenta_usuario_password =
mEncriptador.EncriptarIrreversible(mNuevaClave);
    //ControlArchivos.EscribirArchivo("NuevaClave.txt", mNuevaClave);
    mCuentaUsuario.Cuenta_usuario_intentos_login = 0;
    mRegistro.bitacora_transaccion_id = 4;
    mRegistro.bitacora_criticidad = 1;
    BitacoraBL.Guardar(mRegistro);
    Guardar(mCuentaUsuario);
    throw new CuentaBloqueadaException(mCuentaUsuario, mNuevaClave);
}
Guardar(mCuentaUsuario);
BitacoraBL.Guardar(mRegistro);
throw new ContraseniaIncorrectaException(mCuentaUsuario);
}
else { mCuentaUsuario.Cuenta_usuario_intentos_login = 0;
    mRegistro.cuenta_usuario_id = mCuentaUsuario.Cuenta_usuario_id;
    mRegistro.bitacora_transaccion_id = 1;
    mRegistro.bitacora_criticidad = 3;
    BitacoraBL.Guardar(mRegistro); }
Guardar(mCuentaUsuario);
CuentaUsuarioBL.ObtenerFamilias(mCuentaUsuario);
return mCuentaUsuario;
}

```

(3)

```

public static CuentaUsuario Obtener(string pUsername)
{
    DAO mDAOObject = new DAO();
    DataSet mDs = mDAOObject.ExecuteDataSet("select Cuenta_usuario_id,
Cuenta_usuario_username, Cuenta_usuario_password, Cuenta_usuario_intentos_login,
cuenta_usuario_activa, cuenta_fecha_alta, year(Cuenta_fecha_alta) as anio,
month(Cuenta_fecha_alta) as mes, day(Cuenta_fecha_alta) as dia from cuenta_usuario where
cuenta_usuario_username = '" + pUsername + "' ");
    if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
    {

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
int pId = int.Parse(mDs.Tables[0].Rows[0]["cuenta_usuario_id"].ToString());
CuentaUsuario mCuentaUsuario = new CuentaUsuario(pId);
mCuentaUsuario.cuenta_usuario_activa =
int.Parse(mDs.Tables[0].Rows[0]["cuenta_usuario_activa"].ToString());
ValorizarEntidad(mCuentaUsuario, mDs.Tables[0].Rows[0]);
return mCuentaUsuario;
}
```

(4)

```
public string EncriptarIrreversible(string pCadena)
{
    return _encriptarIrreversible(pCadena);
}

public string EncriptarReversible(string plainText)
{
    byte[] iv = new byte[16];
    byte[] array;
    string clave = "globallogistics1";
    using (Aes aes = Aes.Create())
    {
        aes.Key = Encoding.UTF8.GetBytes(clave);
        aes.IV = iv;
        ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV);
        using (MemoryStream memoryStream = new MemoryStream())
        {
            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, encryptor,
CryptoStreamMode.Write))
            {
                using (StreamWriter streamWriter = new StreamWriter(cryptoStream))
                {
                    streamWriter.Write(plainText);
                }
                array = memoryStream.ToArray();
            }
        }
    }
}
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

(5)

```

public static CuentaUsuario Obtener(int pId)
{
    DAO mDAOObject = new DAO();
    DataSet mDs = mDAOObject.ExecuteDataSet("select Cuenta_usuario_id,
Cuenta_usuario_username, Cuenta_usuario_password, Cuenta_usuario_intentos_login,
cuenta_usuario_activa, cuenta_fecha_alta, year(Cuenta_fecha_alta) as anio,
month(Cuenta_fecha_alta) as mes, day(Cuenta_fecha_alta) as dia from cuenta_usuario where
cuenta_usuario_activa = 1 and Cuenta_usuario_id = " + pId);
    if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
    {
        CuentaUsuario mCuentaUsuario = new CuentaUsuario(pId);
        ValorizarEntidad(mCuentaUsuario, mDs.Tables[0].Rows[0]);
        return mCuentaUsuario;
    }
    else return null;
}

```

(6)

```

public static void ValorizarEntidad(CuentaUsuario pCuentaUsuario, DataRow pDr)
{
    pCuentaUsuario.Cuenta_usuario_id = int.Parse(pDr["Cuenta_usuario_id"].ToString());
    pCuentaUsuario.Cuenta_usuario_username = pDr["Cuenta_usuario_username"].ToString();
    pCuentaUsuario.Cuenta_usuario_password = pDr["Cuenta_usuario_password"].ToString();
    pCuentaUsuario.Cuenta_usuario_intentos_login =
int.Parse(pDr["Cuenta_usuario_intentos_login"].ToString());
    pCuentaUsuario.SetFechaAlta(int.Parse(pDr["dia"].ToString()), int.Parse(pDr["mes"].ToString()),
int.Parse(pDr["anio"].ToString()));
}

```

(7)

```

public static bool ValidarPassword(CuentaUsuario pCuentaUsuario, string pPass)
{
    bool aux;
    if (pCuentaUsuario.Cuenta_usuario_password == pPass) { aux = true; }
    else
    {

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
    aux = false;
}
return aux;
}
```

(8)

```
private string _encriptarIrreversible(string pCadena)
{
    // Create a SHA256
    using (SHA256 sha256Hash = SHA256.Create())
    {
        // ComputeHash - returns byte array
        byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(pCadena));

        // Convert byte array to a string
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < bytes.Length; i++)
        {
            builder.Append(bytes[i].ToString("x2"));
        }
        return builder.ToString();
    }
}
```

(9)

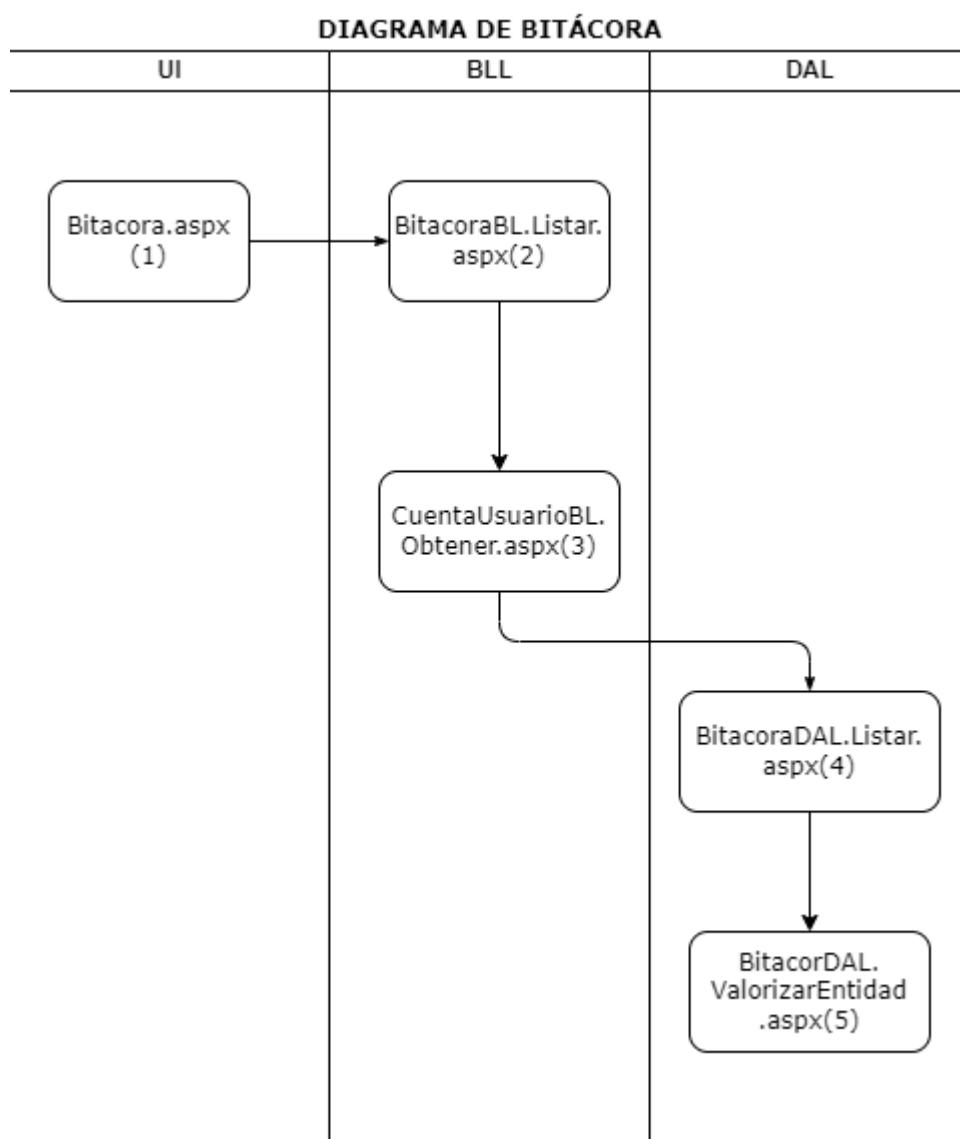
```
mUsuarioLogueado = Request.Cookies["UserLogueado"].Value;
mCuentaUsuario = CuentaUsuarioBL.Obtener(int.Parse(mUsuarioLogueado));
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

2.3.2 Diagrama Leer Bitácora



(1)

```

protected void Page_Load(object sender, EventArgs e)
{
    mBitacora = BitacoraBL.Listar();
    GridView1.DataSource = mBitacora;
    GridView1.DataBind();
}
  
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

(2)

```
public static List<Bitacora> Listar()
{
    return BitacoraDAL.Listar();
}
```

(3)

```
public static List<Bitacora> Listar()
{
    DAO mDAOObject = new DAO();
    DataSet mDs = new DataSet();
    List<Bitacora> mRegistros = new List<Bitacora>();
    mDs = mDAOObject.ExecuteDataSet("select B.bitacora_id, bitacora_criticidad,
        BTM.bitacora_transaccion_desc, B.bitacora_fecha, B.bitacora_hora from bitacora B left join
        cuenta_usuario CU on B.cuenta_usuario_id = CU.cuenta_usuario_id left join
        bitacora_tipo_movimiento BTM on B.bitacora_transaccion_id = BTM.bitacora_transaccion_id
        ");

    if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
    {
        foreach (DataRow mDr in mDs.Tables[0].Rows)
        {
            Bitacora mBitacora = new Bitacora(int.Parse(mDr["bitacora_id"].ToString()));
            ValorizarEntidad(mBitacora, mDr);
            mRegistros.Add(mBitacora);
        }
    }
    return mRegistros;
}
```

(4)

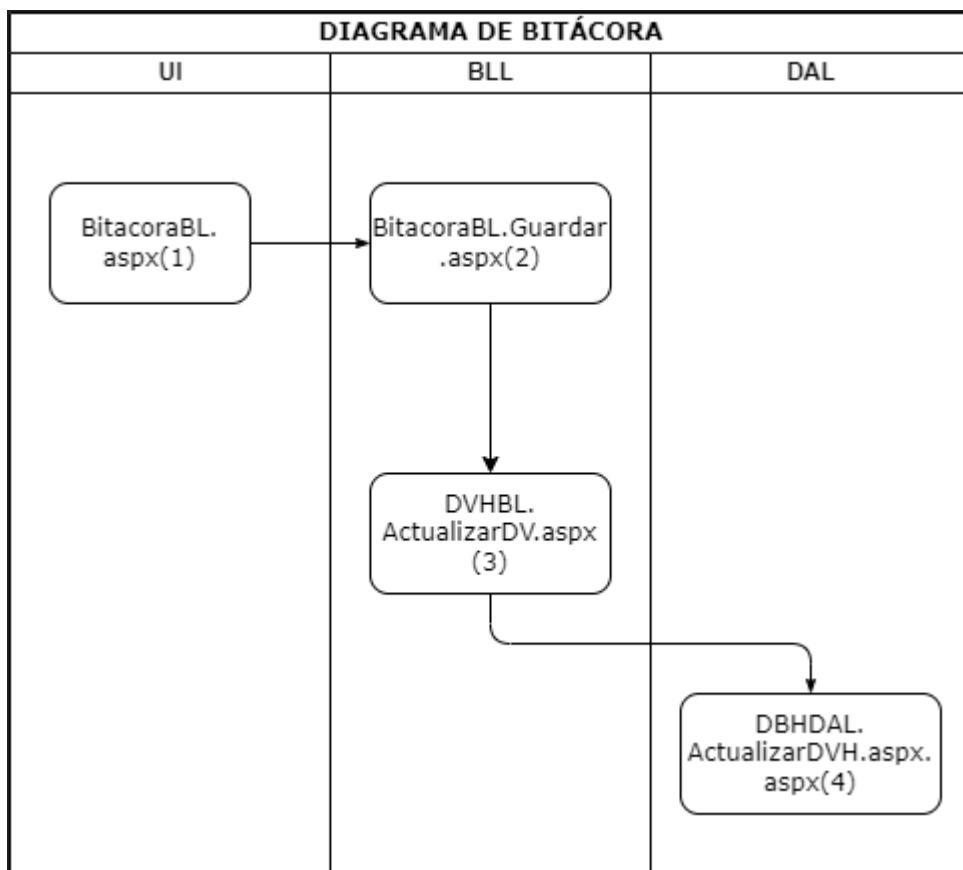
```
public static void ValorizarEntidad(Bitacora pBitacora, DataRow pDr)
{
    pBitacora.bitacora_id = int.Parse(pDr["bitacora_id"].ToString());
    pBitacora.bitacora_criticidad = int.Parse(pDr["bitacora_criticidad"].ToString());
    pBitacora.bitacora_hora = TimeSpan.Parse(pDr["bitacora_hora"].ToString());
    pBitacora.bitacora_fecha = DateTime.Parse(pDr["bitacora_fecha"].ToString());
    pBitacora.bitacora_transaccion = pDr["bitacora_transaccion_desc"].ToString();
}
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

2.3.3 Diagrama Escribir Bitácora



(1)

```

mCuentaUsuario.Cuenta_usuario_intentos_login = 0;
mRegistro.cuenta_usuario_id = mCuentaUsuario.Cuenta_usuario_id;
mRegistro.bitacora_transaccion_id = 1;
mRegistro.bitacora_criticidad = 3;
BitacoraBL.Guardar(mRegistro);
Guardar(mCuentaUsuario);
CuentaUsuarioBL.ObtenerFamilias(mCuentaUsuario);
return mCuentaUsuario;
  
```

(2)

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
public static int Guardar(Bitacora pBitacora)
{
    int i = BitacoraDAL.Guardar(pBitacora);
    DVV x = DVVBL.Obtener("bitacora");
    x.valorDVV = DVVBL.CalcularDVV(x.tabla);
    bool aux = DVVBL.ValidarDVV(x);
    if (aux == true)
    {
        DVHBL.ActualizarDV("bitacora", pBitacora.bitacora_id);
    }
}
```

(3)

```
public static int ActualizarDV(string pTabla, int PK)
{
    Encriptador mCripto = new Encriptador();
    DVH mDVH = ObtenerValorDVH(pTabla, PK);
    mDVH.valorDVHEncryptado = mCripto.EncriptarReversible(mDVH.valorDVH.ToString());
    ActualizarDVH(pTabla, mDVH);
    return DVVBL.ActualizarDVV(pTabla);
}
```

(4)

```
public static int ActualizarDVH(string pTabla, DVH mDVH)
{
    DAO mDAOObject = new DAO();
    string pCadenaComando;
    List<string> mNombreCamposClave = ClavePrimaria(pTabla);
    if (mNombreCamposClave.Count == 2)
    {
        pCadenaComando = "update " + pTabla + " set " + pTabla + "_dvh = '" +
mDVH.valorDVHEncryptado + "' where " + mNombreCamposClave[0] + " = " + mDVH.clavePrimaria + " and " + mNombreCamposClave[1] + " = " + mDVH.clavePrimaria2;
    }
    else
    {
        pCadenaComando = "update " + pTabla + " set " + pTabla + "_dvh = '" +
mDVH.valorDVHEncryptado + "' where " + mNombreCamposClave[0] + " = " + mDVH.clavePrimaria;
    }
    return mDAOObject.ExecuteNonQuery(pCadenaComando);
}
```



UNIVERSIDAD ABIERTA INTERAMERICANA

Facultad de tecnología informática

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

}

Capa User Interface

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

Login.aspx

```
1  %@ Page Language="C#" AutoEventWireup="true" CodeBehind="Login.aspx.cs" Inherits="GlobalLogistics.WebForm1" %>
2
3  <!DOCTYPE html>
4  <html>
5
6  <head>
7
8      <meta charset="UTF-8">
9
10     <title>Global Logistics</title>
11     <link rel="shortcut icon" href="https://i.imgur.com/jd9iHlx.png">
12
13     <style>
14         @import url(http://fonts.googleapis.com/css?family=Exo:100,200,400);
15         @import url(http://fonts.googleapis.com/css?family=Source+Sans+Pro:700,400,300);
16
17     body{
18         margin: 0;
19         padding: 0;
20         background: #fff;
21
22         color: #fff;
23         font-family: Arial;
24         font-size: 12px;
25     }
26
27     .body {
28         position: absolute;
29         top: -20px;
30         left: -20px;
31         right: -40px;
32         bottom: -40px;
33         width: auto;
34         height: auto;
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
34      height: auto;
35      background-image: url(https://i.imgur.com/eh6sCC2.jpg);
36      background-size: cover;
37      -webkit-filter: blur(5px);
38      z-index: 0;
39  }
40
41 .grad{
42     position: absolute;
43     top: -20px;
44     left: -20px;
45     right: -40px;
46     bottom: -40px;
47     width: auto;
48     height: auto;
49     background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,rgba(0,0,0,0)), color-stop(100%,rgba(0,0,0,0.65))); /* Chrome,Safari4+ */
50     z-index: 1;
51     opacity: 0.7;
52 }
53
54 .header{
55     position: absolute;
56     top: calc(50% - 35px);
57     left: calc(45% - 255px);
58     z-index: 2;
59 }
60
61 .header div{
62     float: left;
63     color: #0000bc;
64     font-family: 'Exo', sans-serif;
65     font-size: 35px;
66     font-weight: 200;
67 }
```

```
73 .header span {
74     color: #0043ff /*!important*/;
75 }
76
77 .login{
78     position: absolute;
79     top: calc(50% - 75px);
80     left: calc(50% - 50px);
81     height: 150px;
82     width: 350px;
83     padding: 10px;
84     z-index: 2;
85 }
86
87 .login input[type=text]{
88     width: 250px;
89     height: 30px;
90     background: transparent;
91     border: 1px solid rgba(255,255,255,0.6);
92     border-radius: 2px;
93     color: #fff;
94     font-family: 'Exo', sans-serif;
95     font-size: 16px;
96     font-weight: 400;
97     padding: 4px;
98 }
99
100 .login input[type=password]{
101    width: 250px;
102    height: 30px;
103    background: transparent;
104    border: 1px solid rgba(255,255,255,0.6);
105    border-radius: 2px;
106    color: #fff;
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

107     font-family: 'Exo', sans-serif;
108     font-size: 16px;
109     font-weight: 400;
110     padding: 4px;
111     margin-top: 10px;
112   }
113
114   .login input[type=button]{
115     width: 260px;
116     height: 35px;
117     background: #fff;
118     border: 1px solid #fff;
119     cursor: pointer;
120     border-radius: 2px;
121     color: #a1d6c;
122     font-family: 'Exo', sans-serif;
123     font-size: 16px;
124     font-weight: 400;
125     padding: 6px;
126     margin-top: 10px;
127   }
128
129   .login input[type=button]:hover{
130     opacity: 0.8;
131   }
132
133   .login input[type=button]:active{
134     opacity: 0.6;
135   }
136
137   .login input[type=text]:focus{
138     outline: none;
139     border: 1px solid rgba(255,255,255,0.9);
140   }
141

```

```

142   .login input[type=password]:focus{
143     outline: none;
144     border: 1px solid rgba(255,255,255,0.9);
145   }
146
147   .login input[type=button]:focus{
148     outline: none;
149   }
150
151   ::-webkit-input-placeholder{
152     color: rgba(255,255,255,0.6);
153   }
154
155   ::-moz-placeholder{
156     color: rgba(255,255,255,0.6);
157   }
158   </style>
159
160   <script src="js/prefixfree.min.js"></script>
161
162 </head>
163
164 <body>
165
166   <form id="form1" runat="server">
167
168     <div class="body"></div>
169     <div class="grad">
170       </div>
171     <div class="header">
172       <div>Global<span>Logistics</span></div>
173     <br>
174     <div class="login">
175
176       <!-- -->
177       <input type="text" placeholder="Usuario" id="txtUser" runat="server"><br>
178       <input type="password" placeholder="Password" id="txtPassword" runat="server" name="textPassword"><br>
179       <input type="button" value="Login" onserverclick="BtnLogin_Click" runat="server" id="btnLogin"><br>
180       <br>
181       <label id="lblError" runat="server"></label>
182     </div>
183   </form>
184 </body>
185 </html>

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

Login.aspx.cs

```

namespace GlobalLogistics
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        List<string> mDVHCorruptos;
        List<string> mDVVCorruptos;
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void BtnLogin_Click(object sender, EventArgs e)
        {
            bool adminlogged = false;
            try
            {
                CuentaUsuario mCuentaUsuario = CuentaUsuarioBL.ValidatorLogin(txtUser.Value, txtPassword.Value);
                VerificarBase();
                List<Familia> mFamilias = new List<Familia>();
                mFamilias = CuentaUsuarioBL.ObtenerFamilias(mCuentaUsuario);
                if (mFamilias.ElementAt(0).familia_nombre == "Administrador"){adminlogged = true;}
                Session.Add("UsuarioLoggeado", true);
                Session.Add("IDUsuario", mCuentaUsuario.Cuenta_usuario_id.ToString());
                Session.Add("Usuario", mCuentaUsuario.Cuenta_usuario_username.ToString());
                Session.Add("Familia", mFamilias.ElementAt(0).familia_nombre);
                Session.Add("BaseCorrupta",false);

                DVHBL.ValidatorConsistenciaDVH();
                DVVBL.ValidatorConsistenciaDVV();
                Response.Redirect("MenuPrincipalUI.aspx");
            }
            catch (Servicios.UsuarioInexistenteException ex)
            {
                lblError.Visible = true;
                txtUser.Value = "";
                txtPassword.Value = "";
                lblError.InnerText = "Credenciales incorrectas.";
            }
            catch (Servicios.ContraseniaIncorrectaException ex)
            {
                lblError.Visible = true;
                txtUser.Value = "";
                txtPassword.Value = "";
                lblError.InnerText = "Credenciales incorrectas.";
            }

            catch (Servicios.CuentaBloqueadaException ex)
            {
                lblError.Visible = true;
                txtUser.Value = "";
                txtPassword.Value = "";
                lblError.InnerText = "Usuario bloqueado por intentos fallidos.";
            }
            catch (Servicios.ErrorConsistenciaDVHException ex)
            {
                if (adminlogged)
                {
                    Session["DVH"] = mDVHCorruptos;
                    Session["DVV"] = mDVVCorruptos;
                    Session["BaseCorrupta"] = true;
                    Response.Redirect("MenuPrincipalUI.aspx");
                }
                else
                {
                    lblError.Visible = true;
                    txtUser.Value = "";
                    txtPassword.Value = "";
                    lblError.InnerText = "Base de datos corrupta, contacte al webmaster.";
                }
            }
        }
    }
}

```

Materia: Modelos computacionales de Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
        }

    catch (Servicios.ErrorConsistenciaDVVException ex)
    {
        if (adminlogged)
        {
            Session["DVH"] = mDVHCorruptos;
            Session["DVV"] = mDVVCorruptos;
            Response.Redirect("MenuPrincipalUI.aspx");
        }
        else
        {
            lblError.Visible = true;
            txtUser.Value = "";
            txtPassword.Value = "";
            lblError.InnerText = "Base de datos corrupta, contacte al webmaster.";
        }
    }
}
```

```
    1 referencia  
    void VerificarBase()  
    {  
  
        Session.Add("BaseCorrupta", true);  
  
        mDVBCorruptos = DVBL.ListarDVVsAlterados();  
        mDVHCorruptos = DVHL.ListarDVHsAlterados();  
    }  
}
```

BackupRestore.aspx

```
4
5 <html xmlns="http://www.w3.org/1999/xhtml">
6   <head runat="server">
7     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
8     <title></title>
9     <style>
10       .Buttons{
11         background-color: #1199EE;
12         border:none;
13         color:white;
14         box-shadow:2px 2px 30px #000;
15         border-radius:5px;
16       }
17
18       .TextBox{
19         border-block-color: white;
20         background-color: RGBA(255,255,255,0.3);
21       }
22
23       .Labels{
24         color:aliceblue;
25       }
26     </style>
27   </head>
28   <body style="background-image:url(https://i.ibb.co/x5XK6vY/servidor-web.jpg); font-family:Arial;">
29     <form id="form1" runat="server">
30       <div class="Labels">
31         <b>BACKUP:</b><br />
32         <br />
33         &nbnbsp; Archivo:&nbnbsp;&nbnbsp;<asp:TextBox CssClass="TextBox" ID="txtNombre" runat="server"></asp:TextBox>
34         <br />
35         <br />
36         &nbnbsp;Volumen:&nbnbsp;&nbnbsp;&nbnbsp;&nbnbsp;&nbnbsp;&nbnbsp;&nbnbsp;&nbnbsp;
37         <asp:TextBox CssClass="TextBox" ID="txtVolumenes" runat="server"></asp:TextBox>
38       <br />
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

BackupRestore.aspx.cs

```
9  namespace Globallogistics
10 {
11     public partial class BackupRestore : System.Web.UI.Page
12     {
13         protected void Page_Load(object sender, EventArgs e)
14         {
15         }
16     }
17
18     protected void btnGenerar_Click(object sender, EventArgs e)
19     {
20         try
21         {
22
23             BackupedorBL.RealizarBackup(txtNombre.Text, txtRuta.Text, int.Parse(txtVolumenes.Text));
24             string errormsg = "<script>alert('Backup generado correctamente') </script>";
25             Response.Write(errormsg);
26         }
27         catch (Exception ex)
28         {
29             string errormsg = "<script>alert('" + ex.Message + "') </script>";
30             Response.Write(errormsg);
31         }
32     }
33
34     protected void btnRestore_Click(object sender, EventArgs e)
35     {
36         try
37         {
38             BackupedorBL.RealizarRestore(txtNombreRestore.Text, txtRutaRestore.Text, int.Parse(txtVolumenesRestore.Text));
39             string errormsg = "<script>alert('Backup generado correctamente') </script>";
40             Response.Write(errormsg);
41         }
42
43         catch (Exception ex)
44         {
45             string errormsg = "<script>alert('" + ex.Message + "') </script>";
46             Response.Write(errormsg);
47         }
48     }
49 }
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

ABMUsuarios.aspx

```

1  <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="ABMUsuario.aspx.cs" Inherits="GlobalLogistics.ABMUsuario" %>
2
3  <!DOCTYPE html>
4
5  <html xmlns="http://www.w3.org/1999/xhtml">
6  <head runat="server">
7      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
8      <title></title>
9  </head>
10 <body>
11     <form id="form1" runat="server">
12         <div>
13             <asp:Label ID="Label1" runat="server" Text="Usuario"></asp:Label>
14             <asp:TextBox ID="txtUsuario" runat="server"></asp:TextBox>
15         </div>
16         <asp:Label ID="Label2" runat="server" Text="Contrasenia"></asp:Label>
17         <asp:TextBox ID="txtContrasenia" runat="server"></asp:TextBox>
18         <p>
19             <asp:Button ID="btnCrear" runat="server" OnClick="btnCrear_Click" Text="Crear" />
20         </p>
21     </form>
22 </body>
23 </html>

```

ABMUsuarios.aspx.cs

```

11  namespace GlobalLogistics
12  {
13      public partial class ABMUsuario : System.Web.UI.Page
14      {
15          string mUsuarioLogueado;
16          CuentaUsuario mCuentaUsuarioLogueada;
17          Encriptador mCripto = new Encriptador();
18          0 referencias
19          protected void Page_Load(object sender, EventArgs e)
20          {
21              if (Request.Cookies["UserLogueado"] != null)
22              {
23                  if (Request.Cookies["UserLogueado"] != null)
24                  {
25                      mUsuarioLogueado = Request.Cookies["UserLogueado"].Value;
26                      mCuentaUsuarioLogueada = CuentaUsuarioBL.Obtener(int.Parse(mUsuarioLogueado));
27                  }
28              }
29          0 referencias
30          protected void btnCrear_Click(object sender, EventArgs e)
31          {
32              Encriptador mCripto = new Encriptador();
33              CuentaUsuario mCuentaUsuario = new CuentaUsuario();
34              mCuentaUsuario.Cuenta_usuario_username = mCripto.EncryptarReversible(txtUsuario.Text);
35              mCuentaUsuario.Cuenta_usuario_password = mCripto.EncryptarIrreversible(txtContrasenia.Text);
36              mCuentaUsuario.Cuenta_fecha_alta = DateTime.Now;
37              mCuentaUsuario.Cuenta_usuario_intentos_login = 0;
38              mCuentaUsuario.cuenta_usuario_activa = 1;
39              CuentaUsuarioBL.Guardar(mCuentaUsuario);
40              DVHBL.ActualizarDV("cuenta_usuario", mCuentaUsuario.Cuenta_usuario_id);
41              BE.Bitacora mRegistro = new BE.Bitacora();
42              mRegistro.cuenta_usuario_id = mCuentaUsuario.Cuenta_usuario_id;
43              mRegistro.bitacora_fecha = DateTime.Now;
44
45              mRegistro.bitacora_fecha = DateTime.Now;
46              mRegistro.bitacora_hora = DateTime.Now.TimeOfDay;
47              mRegistro.cuenta_usuario_id = mCuentaUsuarioLogueada.Cuenta_usuario_id;
48              mRegistro.bitacora_criticidad = 4;
49              mRegistro.bitacora_transaccion_id = 5;
50              BitacoraBL.Guardar(mRegistro);
51          }
52      }
53  }

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

ABMProducto.aspx

```
1  <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="ABMProducto.aspx.cs" Inherits="GlobalLogistics.ABMProducto" %>
2
3  <!DOCTYPE html>
4
5
6  <html xmlns="http://www.w3.org/1999/xhtml">
7  <head runat="server">
8  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
9  <title></title>
10 </head>
11 <style>
12   .Buttons{
13     border-style: none;
14     border-color: inherit;
15     border-width: medium;
16     background-color: #1199EE;
17     color:white;
18     box-shadow:2px 2px 30px #000;
19     border-radius:5px;
20     margin-top: 0px;
21     height: 30px;
22     width: 100px;
23   }
24
25   #btnAgregar:hover{
26     background-color: #117bee;
27     cursor:pointer;
28   }
29
30   #btnModificar:hover{
31     background-color: #117bee;
32     cursor:pointer;
33   }
34 
```

```
35   #btnEliminar:hover{
36     background-color: #117bee;
37     cursor:pointer;
38   }
39
40   .TextBox{
41     border-block-color: white;
42     background-color: RGBA(255,255,255,0.3);
43   }
44
45   .mGrid {
46     width: 100%;
47     background-color: #fff;
48     margin: 5px 0 10px 0;
49     border: solid 1px #525252;
50     border-collapse: collapse;
51   }
52
53   .mGrid td {
54     padding: 2px;
55     border: solid 1px #c1c1c1;
56     color: #717171;
57   }
58
59   .mGrid th {
60     padding: 4px 2px;
61     color: #fff;
62     background: #424242 url(grd_head.png) repeat-x top;
63     border-left: solid 1px #525252;
64     font-size: 0.9em;
65 } 
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
67  .mGrid .alt {
68    background: #fcfcfc url(grd_alt.png) repeat-x top;
69  }
70
71  .mGrid .pgr {
72    background: #424242 url(grd_pgr.png) repeat-x top;
73  }
74
75  .mGrid .pgr table {
76    margin: 5px 0;
77  }
78
79  .mGrid .pgr td {
80    border-width: 0;
81    padding: 0 6px;
82    border-left: solid 1px #666;
83    font-weight: bold;
84    color: #fff;
85    line-height: 12px;
86  }
87
88  .mGrid .pgr a {
89    color: #666;
90    text-decoration: none;
91  }
92
93  .mGrid .pgr a:hover {
94    color: #000;
95    text-decoration: none;
96  }
97  .Labels{
98    color:aliceblue;
99    font-family:Arial;
100   font-size:35px;
101 }

102 </style>
103 <body style="background-image:url(https://negociosyempresa.com/wp-content/uploads/2021/11/caja-registradora-tpv.jpg");>
104 <form id="form2" runat="server">
105   <div>
106
107     <asp:Label ID="Label1" runat="server" Text="Administracion de Productos" CssClass="Labels"></asp:Label>
108
109     <br />
110     <asp:GridView ID="grdProductos"
111       runat="server" AutoGenerateSelectButton="True"
112       OnSelectedIndexChanged="grdProductos_SelectedIndexChanged" CssClass="mGrid">
113       </asp:GridView>
114     </div>
115     <asp:TextBox class="TextBox" ID="TextBox1" runat="server" Width="249px" Height="25px" style="margin-top: 0px"></asp:TextBox>
116     &nbsp;&nbsp;
117     <asp:Button class="Buttons" ID="btnAgregar" runat="server" OnClick="btnAgregar_Click" Text="Agregar" />
118     &nbsp;&nbsp;
119     <asp:Button class="Buttons" ID="btnModificar" runat="server" OnClick="btnModificar_Click" Text="Modificar" />
120     &nbsp;&nbsp;
121     <asp:Button class="Buttons" ID="btnEliminar" runat="server" Text="Eliminar" OnClick="btnEliminar_Click" />
122   </form>
123 </body>
124 </html>
```

ABMProducto.aspx.cs

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

10  namespace GlobalLogistics
11  {
12      2 referencias
13      public partial class ABMProducto : System.Web.UI.Page
14      {
15          List<Producto> mProductos = new List<Producto>();
16          Producto mProductoSeleccionado;
17          0 referencias
18          protected void Page_Load(object sender, EventArgs e)
19          {
20              Actualizar();
21          }
22          4 referencias
23          public void Actualizar()
24          {
25              mProductos = ProductoBL.Listar();
26              grdProductos.DataSource = mProductos;
27              grdProductos.DataBind();
28          }
29          0 referencias
30          protected void grdProductos_SelectedIndexChanged(object sender, EventArgs e)
31          {
32              //Producto mProducto = ProductoBL.Obtener(grdProductos.SelectedIndex-1);
33              //TextBox1.Text = mProducto.producto_nombre;
34              GridViewRow row = grdProductos.Rows[grdProductos.SelectedIndex];
35              mProductoSeleccionado = ProductoBL.Obtener(Int.Parse(row.Cells[1].Text));
36              TextBox1.Text = mProductoSeleccionado.producto_nombre;
37              HttpCookie mNombreProducto = new HttpCookie("Nombre Producto");
38              mNombreProducto.Value = mProductoSeleccionado.producto_nombre;
39              Response.Cookies.Add(mNombreProducto);
40
41          }
42
43          0 referencias
44          protected void btnAgregar_Click(object sender, EventArgs e)
45          {
46              Producto mProducto = new Producto();
47              mProducto.producto_nombre = TextBox1.Text;
48              ProductoBL.Guardar(mProducto);
49              Actualizar();
50
51          }
52
53          protected void btnModificar_Click(object sender, EventArgs e)
54          {
55              Producto mProducto = ProductoBL.Obtener(Request.Cookies["Nombre Producto"].Value);
56              mProducto.producto_nombre = TextBox1.Text;
57              ProductoBL.Guardar(mProducto);
58              Actualizar();
59
60          }
61
62          protected void btnEliminar_Click(object sender, EventArgs e)
63          {
64              Producto mProducto = ProductoBL.Obtener(Request.Cookies["Nombre Producto"].Value);
65              ProductoBL.Eliminar(mProducto);
66              Actualizar();
67          }
68
69      }
70
71  }

```

Bitacora.aspx

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
1  %@ Page Language="C#" AutoEventWireup="true" CodeBehind="Bitacora.aspx.cs" Inherits="GlobalLogistics.Bitacora" %>
2
3  <!DOCTYPE html>
4
5  <html xmlns="http://www.w3.org/1999/xhtml">
6  <head runat="server">
7
8      <style type="text/css">
9
10     .main-background {
11         background-image: url(https://i.ibb.co/drgvK9q/4M2.jpg);
12         width: 70%;
13         height: 70%;
14         position: absolute;
15     }
16
17     * {
18         padding: 0;
19         margin: 0;
20     }
21
22     body {
23         font: 11px Tahoma;
24     }
25
26     h1 {
27         font: bold 32px Times;
28         color: #666;
29         text-align: center;
30         padding: 20px 0;
31     }
32
33     #container {
34         width: 700px;
35         margin: 10px auto;
36     }
37
38     .mGrid {
39         width: 100%;
40         background-color: #fff;
41         margin: 5px 0 10px 0;
42         border: solid 1px #525252;
43         border-collapse: collapse;
44     }
45
46     .mGrid td {
47         padding: 2px;
48         border: solid 1px #c1c1c1;
49         color: #717171;
50     }
51
52     .mGrid th {
53         padding: 4px 2px;
54         color: #fff;
55         background: #424242 url(grd_head.png) repeat-x top;
56         border-left: solid 1px #525252;
57         font-size: 0.9em;
58     }
59
60     .mGrid .alt {
61         background: #fcfcfc url(grd_alt.png) repeat-x top;
62     }
63
64     .mGrid .pgr {
65         background: #424242 url(grd_pgr.png) repeat-x top;
66     }
67
68     .mGrid .pgr table {
69         margin: 5px 0;
70     }
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

72     .mGrid .pgr td {
73         border-width: 0;
74         padding: 0 6px;
75         border-left: solid 1px #666;
76         font-weight: bold;
77         color: #fff;
78         line-height: 12px;
79     }
80
81     .mGrid .pgr a {
82         color: #666;
83         text-decoration: none;
84     }
85
86     .mGrid .pgr a:hover {
87         color: #000;
88         text-decoration: none;
89     }
90
91     /* #GridView1{
92         opacity: 0.4;
93     }*/
94     </style>
95     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
96     <title></title>
97 </head>
98 <body class="main-background" style="left: 56px; top: 0px; width: 92%">
99     <form id="form1" runat="server" translate="0.2">
100         <h1>Bitácora</h1>
101         <br /><br /><br /><br />
102         <div>
103             <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="True" GridLines="None"
104                 AllowPaging="true" CssClass="mGrid" PagerStyle-CssClass="pgr" AlternatingRowStyle-CssClass="alt"
105                 PageSize="200">
106
107             </asp:GridView>
108         </div>
109     </form>
110 </body>
111 </html>

```

Bitacora.aspx.cs

```

9
10     namespace GlobalLogistics
11     {
12         public partial class Bitacora : System.Web.UI.Page
13         {
14
15             List<BE.Bitacora> mBitacora = new List<BE.Bitacora>();
16
17             protected void Page_Load(object sender, EventArgs e)
18             {
19                 mBitacora = BitacoraBL.Listar();
20                 GridView1.DataSource = mBitacora;
21                 GridView1.DataBind();
22             }
23         }

```

MenuPrincipalUI.aspx

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
7  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
8  <title>Menu</title>
9  <link rel="shortcut icon" href="https://i.imgur.com/1d9iHlx.png">
10 <style>body {
11   margin: 0;
12   padding: 0;
13   background-color: #222;
14 }
15
16 /* {
17   font-family: Helvetica,sans-serif;
18   color: #555;
19 }
20
21 #mmeennuu {
22   display: none;
23 }
24
25 #mmeennuu:checked ~ .menu {
26   width: 500px;
27   border-radius: 5px;
28   background-color:RGBA(17,153,238,0.2);
29   border: 3px solid #1199EE;
30   height: 85px;
31 }
32 #mmeennuu:checked ~ .menu > ul {
33   display: block;
34   opacity: 1;
35 }
36 #mmeennuu:checked ~ .menu > .barry {
37   display: none;
38 }
39
40 .menu {
41   display: block;
42   margin: 30px auto;
43   width: 100px;
44   height: 100px;
45   background-color: #1199EE;
46   border: 3px solid transparent;
47   border-radius: 50%;
48   overflow: hidden;
49   cursor: pointer;
50   transition: all 0.5s ease-in-out;
51   -webkit-transition: all 0.5s ease-in-out;
52   -moz-transition: all 0.5s ease-in-out;
53   -o-transition: all 0.5s ease-in-out;
54   -ms-transition: all 0.5s ease-in-out;
55 }
56 .menu div.barry {
57   width: 40px;
58   margin: 35px auto;
59 }
60 .menu div.barry .bar {
61   display: block;
62   width: 100%;
63   height: 5px;
64   margin-top: 3px;
65   border-radius: 2px;
66   background-color: #fff;
67 }
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

68 .menu ul {
69   opacity: 0;
70   display: none;
71   transition: all 0.5s ease-in-out;
72   -webkit-transition: all 0.5s ease-in-out;
73   -moz-transition: all 0.5s ease-in-out;
74   -o-transition: all 0.5s ease-in-out;
75   -ms-transition: all 0.5s ease-in-out;
76   list-style-type: none;
77   padding: 0;
78   width: 500px;
79   text-align: center;
80   margin-bottom: 0;
81 }
82 .menu ul li {
83   display: inline-block;
84 }
85 .alerta {
86   vertical-align: bottom
87 }
88
89 .menu ul li a {
90   text-decoration: none;
91   display: inline-block;
92   padding: 15px 25px;
93   color: #ffff;
94   font-size: 20px;
95   font-weight: bold;
96   transition: all 0.3s ease-in-out;
97   -webkit-transition: all 0.3s ease-in-out;
98   -moz-transition: all 0.3s ease-in-out;
99   -o-transition: all 0.3s ease-in-out;
100  -ms-transition: all 0.3s ease-in-out;
101 }

102  border: 3px solid transparent;
103  border-radius: 5px;
104  text-shadow: 2px 2px 4px #000000;
105 }
106 .menu ul li a:hover {
107   border-color: #1199EE;
108 }
109 .menu ul li a:target {
110   border-bottom-color: #1199EE;
111 }
112 .alerrta {
113   margin-left: 40px;
114 }
115 #Button1{
116   position: absolute;
117   right:70px;
118   padding:10px 40px 10px 40px;
119   background-color: #1199EE;
120   border:none;
121   color:white;
122   box-shadow:2px 2px 30px #000;
123   border-radius:5px;
124 }
125 .labels{
126   position: absolute;
127   left:50px;
128   padding:0px 150px 0px 0px;
129 }
130 #Button1:hover{
131   background-color: #117bee;
132   cursor:pointer;
133 }
134 }
135

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

137     #btnBackup{
138         background-color: #1199EE;
139         border:none;
140         color:white;
141         box-shadow:2px 2px 30px #000;
142         border-radius:5px;
143     }
144
145     #btnBackup:hover{
146         background-color: #117bee;
147         cursor:pointer;
148     }
149
150     #btnProductos{
151         border-style: none;
152         border-color: inherit;
153         border-width: medium;
154         background-color: #1199EE;
155         color:white;
156         box-shadow:2px 2px 30px #000;
157         border-radius:5px;
158         margin-top: 0px;
159     }
160
161     #btnProductos:hover{
162         background-color: #117bee;
163         cursor:pointer;
164     }
165
166     </style>
167 </head>
168 <body style="background-image:url(https://i.ibb.co/YRy7zVx/store-g0842e754b-1920.jpg)">
169 <form id="form1" runat="server">
    <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Logout" />

```



```

171     <input type='checkbox' id='mmeennuu' />
172     <label class='menu' for='mmeennuu'>
173
174     <div class='barry'>
175         <span class='bar'></span>
176         <span class='bar'></span>
177         <span class='bar'></span>
178         <span class='bar'></span>
179     </div>
180
181     <ul>
182         <li><a id='bitacora' href='Bitacora.aspx' runat="server" visible="false">Bitacora</a></li>
183         <li><a id='usuario' href='ABMUsuario.aspx' runat="server" visible="False">Usuarios</a></li>
184         <li><a id='digitos' href='Verificadores.aspx' runat="server" visible="false">Digitos Verificadores</a></li>
185     </ul>
186
187     </label>
188
189     <div style="margin-left: 40px">
190         <br />
191         <asp:Label style="top:50px;" class="labels" ID="lblUser" runat="server" Text="Label" Font-Bold="True" Font-Names="Arial" Font-Size="Large" ForeColor="W
192         <br />
193         <br />
194         <asp:Label style="top:80px;" class="labels" ID="lblRol" runat="server" Text="Label" Font-Bold="True" Font-Names="Arial" Font-Size="Large" ForeColor="W
195         <br />
196         <br />
197         <br />
198         <!-->
199         <br />
200         <br />
201         <asp:Button ID="btnBackup" runat="server" OnClick="btnBackup_Click" Text="BackupRestore" Height="38px" />
202         <br />
203         <br />
204         <asp:Button ID="btnProductos" runat="server" OnClick="btnProductos_Click1" Text="Productos" Height="38px" />
205
206         <br />
207         <br />
208
209         <br />
210     </div>
211     <div class="alerrta">
212         <asp:Label ID="lblDVVDVH" runat="server" Text="Label" Font-Bold="True" Font-Names="Arial" Font-Size="Larger" ForeColor="#C00000" Visible="False"></asp
213         </div>
214     </form>
215 </body>
216 </html>

```

Materia: Modelos computacionales de Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

MenuPrincipalUI.aspx.cs

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

78     }
79     else
80     {
81         Response.Redirect("Login.aspx");
82     }
83 }
84
85     0 referencias
86     protected void btnProductos_Click(object sender, EventArgs e)
87     {
88         if (Convert.ToBoolean(this.Session["UsuarioLoggeado"]))
89         {
90             Response.Redirect("ABMProducto.aspx");
91         }
92         else
93         {
94             Response.Redirect("Login.aspx");
95         }
96     }
97
98     0 referencias
99     protected void btnProductos_Click1(object sender, EventArgs e)
100    {
101        if (Convert.ToBoolean(this.Session["UsuarioLoggeado"]))
102        {
103            Response.Redirect("ABMProducto.aspx");
104        }
105        else
106        {
107            Response.Redirect("Login.aspx");
108        }
109    }
110 }
111

```

Verificadores.aspx

```

1 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Verificadores.aspx.cs" Inherits="GlobalLogistics.Verificadores" %>
2
3 <!DOCTYPE html>
4
5 <html xmlns="http://www.w3.org/1999/xhtml">
6 <head runat="server">
7
8     <style type="text/css">
9
10    .main-background {
11        background-image: url(https://i.ibb.co/drgvK9q/4M2.jpg);
12        width: 70%;
13        height: 70%;
14        position: absolute;
15    }
16
17    * {
18        padding: 0;
19        margin: 0;
20    }
21
22 body {
23     font: 11px Tahoma;
24 }
25
26 h1 {
27     font: bold 32px Times;
28     color: #666;
29     text-align: center;
30     padding: 20px 0;
31 }
32
33 #btnRecalcularDigitos{
34     background-color: #1199EE;
35     border:none;

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

36         color:white;
37         box-shadow:2px 2px 30px #000;
38         border-radius:5px;
39     }
40
41 #container {
42     width: 700px;
43     margin: 10px auto;
44 }
45
46 .mGrid {
47     width: 100%;
48     background-color: #fff;
49     margin: 5px 0 10px 0;
50     border: solid 1px #525252;
51     border-collapse: collapse;
52 }
53
54 .mGrid td {
55     padding: 2px;
56     border: solid 1px #c1c1c1;
57     color: #717171;
58 }
59
60 .mGrid th {
61     padding: 4px 2px;
62     color: #fff;
63     background: #424242 url(grd_head.png) repeat-x top;
64     border-left: solid 1px #525252;
65     font-size: 0.9em;
66 }

```

```

68 .mGrid .alt {
69     background: #fcfcfc url(grd_alt.png) repeat-x top;
70 }
71
72 .mGrid .pgr {
73     background: #424242 url(grd_pgr.png) repeat-x top;
74 }
75
76 .mGrid .pgr table {
77     margin: 5px 0;
78 }
79
80 .mGrid .pgr td {
81     border-width: 0;
82     padding: 0 6px;
83     border-left: solid 1px #666;
84     font-weight: bold;
85     color: #fff;
86     line-height: 12px;
87 }
88
89 .mGrid .pgr a {
90     color: #666;
91     text-decoration: none;
92 }
93
94 .mGrid .pgr a:hover {
95     color: #000;
96     text-decoration: none;
97 }

103 <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
104 <title></title>
105 </head>
106 <body class="main-background" style="left: 56px; top: 0px; width: 92%">
107     <form id="form1" runat="server" translate="0,2">
108         <h1>Verificadores</h1>
109         <asp:Button ID="btnRecalcularDigitos" runat="server" OnClick="btnRecalcularDigitos_Click" Text="Recalcular Digitos" Height="38px" />
110         <br /><br /><br /><br />
111         <div>
112             <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="True" GridLines="None"
113                 AllowPaging="true" CssClass="mGrid" PagerStyle-CssClass="pgr" AlternatingRowStyle-CssClass="alt"
114                 PageSize="200">
115
116             </asp:GridView>
117         </div>
118     </form>
119 </body>
120 </html>

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

Verificadores.aspx.cs

```

10  namespace GlobalLogistics
11  {
12      2 referencias
13      public partial class Verificadores : System.Web.UI.Page
14      {
15          0 referencias
16          protected void Page_Load(object sender, EventArgs e)
17          {
18              mCorrupcionBD = new Dictionary<string, string>();
19              if (this.Session["DVV"] as List<string> != null)
20              {
21                  foreach (string x in this.Session["DVV"] as List<string>)
22                  {
23                      mCorrupcionBD.Add(x, "DVV");
24                  }
25              }
26              if (this.Session["DVH"] as List<string> != null)
27              {
28                  foreach (string x in this.Session["DVH"] as List<string>)
29                  {
30                      mCorrupcionBD.Add(x, "DVH");
31                  }
32              }
33              GridView1.DataSource = mCorrupcionBD;
34              GridView1.DataBind();
35          }
36          0 referencias
37          protected void btnRecalcularDigitos_Click(object sender, EventArgs e)
38          {
39              DVHBL.CalcularTodos();
40              DVVBL.CalcularTodo();
41              Session["DVH"] = null;
42              Session["DVV"] = null;
43              this.Session["BaseCorrupta"] = false;
44              Response.Redirect("MenuPrincipalUI.aspx");
45
46          void VerificarBase()
47          {
48              mCorrupcionBD.Clear();
49
50              foreach (string x in DVVBL.ListarDVVsAlterados())
51              {
52                  mCorrupcionBD.Add(x, "DVV");
53              }
54              foreach (string x in DVHBL.ListarDVHsAlterados())
55              {
56                  mCorrupcionBD.Add(x, "DVH");
57              }
58              GridView1.DataSource = mCorrupcionBD;
59              GridView1.DataBind();
60          }
61      }
62

```



UNIVERSIDAD ABIERTA INTERAMERICANA

Facultad de tecnología informática

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

Capa Negocio BLL

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

BitacoraBL.cs

```
9  namespace BLL
10 {
11     public class BitacoraBL
12     {
13         public static int Guardar(Bitacora pBitacora)
14         {
15             int i = BitacoraDAL.Guardar(pBitacora);
16             DVV x = DVVBL.Obtener("bitacora");
17             x.valorDVV = DVVBL.CalcularDVV(x.tabla);
18             bool aux = DVVBL.ValidarDVV(x);
19             if (aux == true)
20             {
21                 DVHBL.ActualizarDV("bitacora", pBitacora.bitacora_id);
22             }
23             return i;
24         }
25         public static List<Bitacora> Listar()
26         {
27             return BitacoraDAL.Listar();
28         }
29         public static Bitacora Obtener(int pId)
30         {
31             return BitacoraDAL.Obtener(pId);
32         }
33         public static List<Bitacora> ListarRango(DateTime pFechaDesde, DateTime pFechaHasta)
34         {
35             return BitacoraDAL.ListarRango(pFechaDesde, pFechaHasta);
36         }
}
```

BackupedorBL.cs

Materia: Modelos computacionales de Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
6
7     namespace BLL
8     {
9         2 referencias
10        public class BackapeadorBL
11        {
12            1 referencia
13            public static int RealizarBackup(string pNombreArchivo, string pRuta, int pVolumenes)
14            {
15                return BackapeadorDAL.RealizarBackup(pNombreArchivo, pRuta, pVolumenes);
16            }
17            1 referencia
18            public static int RealizarRestore(string pNombreArchivo, string pRuta, int pVolumenes)
19            {
20                return BackapeadorDAL.RealizarRestore(pNombreArchivo, pRuta, pVolumenes);
21            }
22        }
23    }
```

CuentaUsuario.cs

```
9  namespace BLL
10 {
11     public class CuentaUsuarioBL
12     {
13         1 referencia
14         public static List<CuentaUsuario> Listar()
15         {
16             return CuentaUsuarioDAL.Listar();
17         }
18
19         2 referencias
20         public static CuentaUsuario Obtener(int pId)
21         {
22             CuentaUsuario mCuentaUsuario = CuentaUsuarioDAL.Obtener(pId);
23             if (mCuentaUsuario is null) { mCuentaUsuario = new CuentaUsuario(); }
24             return mCuentaUsuario;
25         }
26
27         1 referencia
28         public static CuentaUsuario Obtener(string pUsername)
29         {
30             CuentaUsuario mCuentaUsuario = CuentaUsuarioDAL.Obtener(pUsername);
31             if (mCuentaUsuario is null) { mCuentaUsuario = new CuentaUsuario(); }
32             return mCuentaUsuario;
33         }
34
35         1 referencia
36         public static bool ValidarPassword(CuentaUsuario pCuentaUsuario, string pPass)
37         {
38             bool aux;
39             if (pCuentaUsuario.Cuenta_usuario_password == pPass) { aux = true; }
40             else
41             {
42                 aux = false;
43             }
44             return aux;
45         }
46     }
47 }
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

41
42
43     }
44
45     0 referencias
46     public static int ModificarContrasenia(CuentaUsuario pCuentaUsuario, string pContrasenia)
47     {
48         pCuentaUsuario.Cuenta_usuario_password = pContrasenia;
49         return Guardar(pCuentaUsuario);
50     }
51     1 referencia
52     public static CuentaUsuario ValidarLogin(string pUsername, string pPassword)
53     {
54         Encryptador mEncriptador = new Encryptador();
55         CuentaUsuario mCuentaUsuario = Obtener(mEncriptador.EncryptarReversible(pUsername));
56         Bitacora mRegistro = new Bitacora();
57         mRegistro.cuenta_usuario_id = mCuentaUsuario.Cuenta_usuario_id;
58         mRegistro.bitacora_fecha = DateTime.Now;
59         mRegistro.bitacora_hora = DateTime.Now.TimeOfDay;
60         if (mCuentaUsuario.Cuenta_usuario_id == 0 || mCuentaUsuario.cuenta_usuario_activa == 0)
61         {
62             mRegistro.cuenta_usuario_id = -1;
63             mRegistro.bitacora_transaccion_id = 2;
64             mRegistro.bitacora_criticidad = 3;
65             BitacoraBL.Guardar(mRegistro);
66             throw new UsuarioInexistenteException();
67         }
68
69         bool aux = ValidarPassword(mCuentaUsuario, mEncriptador.EncryptarIrreversible(pPassword));
70         if (aux == false)
71         {
72             mCuentaUsuario.Cuenta_usuario_intentos_login += 1;
73             mRegistro.cuenta_usuario_id = mCuentaUsuario.Cuenta_usuario_id;
74             mRegistro.bitacora_transaccion_id = 3;
75             mRegistro.bitacora_criticidad = 2;
76
77             if (mCuentaUsuario.Cuenta_usuario_intentos_login == 3)
78             {
79
80                 string mNuevaClave = GenerarClaveAleatoria();
81                 mCuentaUsuario.Cuenta_usuario_password = mEncriptador.EncryptarIrreversible(mNuevaClave);
82                 //ControlArchivos.EscribirArchivo("NuevaClave.txt", mNuevaClave);
83                 mCuentaUsuario.Cuenta_usuario_intentos_login = 0;
84                 mRegistro.bitacora_transaccion_id = 4;
85                 mRegistro.bitacora_criticidad = 1;
86                 BitacoraBL.Guardar(mRegistro);
87                 Guardar(mCuentaUsuario);
88                 throw new CuentaBloqueadaException(mCuentaUsuario, mNuevaClave);
89             }
90             Guardar(mCuentaUsuario);
91             BitacoraBL.Guardar(mRegistro);
92             throw new ContraseniaIncorrectaException(mCuentaUsuario);
93         }
94     }
95
96     1 referencia
97     public static void Logout(CuentaUsuario pUser)
98     {
99         Bitacora mRegistro = new Bitacora();
100        mRegistro.bitacora_criticidad = 4;
101        mRegistro.bitacora_transaccion_id = 6;
102        mRegistro.cuenta_usuario_id = pUser.Cuenta_usuario_id;
103        mRegistro.bitacora_fecha = DateTime.Now;
104        mRegistro.bitacora_hora = DateTime.Now.TimeOfDay;
105        BitacoraBL.Guardar(mRegistro);
106    }

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

111     public static string GenerarClaveAleatoria()
112     {
113         var chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
114         var stringChars = new char[8];
115         var random = new Random();
116
117         for (int i = 0; i < stringChars.Length; i++)
118         {
119             stringChars[i] = chars[random.Next(chars.Length)];
120         }
121
122         var finalString = new String(stringChars);
123         return finalString;
124     }
125
126     5 referencias
127     public static int Guardar(CuentaUsuario pCuentaUsuario)
128     {
129         int a = CuentaUsuarioDAL.Guardar(pCuentaUsuario);
130         return a;
131     }
132     0 referencias
133     public static int Eliminar(CuentaUsuario pCuentaUsuario)
134     {
135         List<CuentaUsuario> mCuentas = Listar();
136         if (mCuentas.Count >= 2)
137         {
138             List<Patente> mPatentesExcepto = PatenteBL.ListarExcepto(pCuentaUsuario);
139
140             if (mPatentesExcepto.Count == 0)
141             {
142                 int a;
143
144                 List<Patente> mPatente = ObtenerPatentes(pCuentaUsuario);
145                 foreach (Patente x in mPatente)
146                 {
147                     : EliminarPatente(x, pCuentaUsuario);
148                     a = CuentaUsuarioDAL.Eliminar(pCuentaUsuario);
149                     return a;
150                 }
151                 else
152                 {
153                     PatentesSinAsignarException ex = new PatentesSinAsignarException();
154                     ex.mPatentesSinAsignar = mPatentesExcepto;
155                     throw ex;
156                 }
157                 else throw new NoQuedanUsuariosException();
158             }
159
160             2 referencias
161             public static List<Patente> ObtenerPatentes(CuentaUsuario pCuentaUsuario)
162             {
163                 List<Patente> mPatentes = CuentaUsuarioDAL.ObtenerPatentes(pCuentaUsuario);
164                 return mPatentes;
165             }
166
167             4 referencias
168             public static List<Familia> ObtenerFamilias(CuentaUsuario pCuentaUsuario)
169             {
170                 List<Familia> mFamilias = CuentaUsuarioDAL.ObtenerFamilias(pCuentaUsuario);
171                 foreach (Familia mFam in mFamilias)
172                 {
173                     mFam.mPatentes = FamiliaBL.ObtenerPatentes(mFam);
174                 }
175             }
176         }
    
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

177     public static List<Patente> ListarPermisosDisponibles(CuentaUsuario pCuentaUsuario)
178     {
179         List<Patente> mPermisos = new List<Patente>();
180         mPermisos = ObtenerPermitentes(pCuentaUsuario);
181         List<Familia> mFamilias = ObtenerFamilias(pCuentaUsuario);
182         foreach (Familia x in mFamilias)
183         {
184             List<Patente> mPatentesAux = FamiliaBL.ObtenerPatentes(x);
185             mPermisos.AddRange(mPatentesAux);
186         }
187         return mPermisos;
188     }
189
190     1 referencia
191     public static bool VerificarPatente(int pPatenteId, CuentaUsuario pCuentaUsuario)
192     {
193         List<Patente> mPermisos = ListarPermisosDisponibles(pCuentaUsuario);
194         bool resultado = mPermisos.Any(x => x.patente_id == pPatenteId);
195         return resultado;
196     }
197
198     0 referencias
199     public static int AgregarPatente(Patente pPatente, CuentaUsuario pCuentaUsuario)
200     {
201         bool mAsignada = VerificarPatente(pPatente.patente_id, pCuentaUsuario);
202         if (mAsignada == false)
203         {
204             CuentaUsuarioDAL.AgregarPatente(pPatente, pCuentaUsuario);
205             DVHBL.ActualizarDV("cuenta_usuario_patente", pCuentaUsuario.Cuenta_usuario_id, pPatente.patente_id);
206         }
207         else
208             throw new PatenteYaAsignadaException();
209     }
210
211     1 referencia
212     public static int EliminarPatente(Patente pPatente, CuentaUsuario pCuentaUsuario)
213     {
214         int aux = 0;
215
216         List<CuentaUsuario> mCuentasUsuario = PatenteBL.ListarUsuarios(pPatente);
217
218         if (mCuentasUsuario.Count > 1)
219         {
220             aux = CuentaUsuarioDAL.EliminarPatente(pPatente, pCuentaUsuario);
221             DVVBL.ActualizarDV("cuenta_usuario_patente");
222         }
223         else
224         {
225             PatentesSinAsignarException ex = new PatentesSinAsignarException();
226             throw ex;
227         }
228
229         return aux;
230     }
231
232     0 referencias
233     public static int AgregarFamilia(Familia pFamilia, CuentaUsuario pCuentaUsuario)
234     {
235         int aux = CuentaUsuarioDAL.AgregarFamilia(pFamilia, pCuentaUsuario);
236         DVHBL.ActualizarDV("cuenta_usuario_familia", pCuentaUsuario.Cuenta_usuario_id, pFamilia.familia_id);
237         return aux;
238     }
239
240     1 referencia
241     public static int EliminarFamilia(Familia pFamilia, CuentaUsuario pCuentaUsuario)
242     {
243         int aux = CuentaUsuarioDAL.EliminarFamilia(pFamilia, pCuentaUsuario);

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
241     : DVHBL.ActualizarDVV("cuenta_usuario_familia");
242     : return aux;
243 }
244 public static List<CuentaUsuario> ObtenerUsuarios(Familia pFamilia)
245 {
246     return CuentaUsuarioDAL.ObtenerUsuarios(pFamilia);
247 }
248 }
249 ---
```

DVHBL.cs

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

10  namespace BLL
11  {
12      public class DVHBL
13      {
14          public static List<DVH> ObtenerValoresDVH(string pTabla)
15          {
16              return DVHDAL.ObtenerValoresDVH(pTabla);
17          }
18          public static DVH ObtenerValorDVH(string pTabla, int PK1, int PK2)
19          {
20              return DVHDAL.ObtenerValorDVH(pTabla, PK1, PK2);
21          }
22          public static DVH ObtenerValorDVH(string pTabla, int PK1)
23          {
24              return DVHDAL.ObtenerValorDVH(pTabla, PK1);
25          }
26          public static bool ValidarValorDVH(DVH mDVH)
27          {
28              return mDVH.valorDVH == mDVH.valorDVHBase;
29          }
30          public static List<string> ListarDVHsAlterados()
31          {
32              List<string> mRegistros = new List<string>();
33              mRegistros.Clear();
34              List<DVH> mValores = DVVBL.Obtener();
35              foreach (DVH y in mValores)
36              {
37                  ...
38                  ...
39                  ...
40                  ...
41                  ...
42                  ...
43                  ...
44                  ...
45                  ...
46                  ...
47                  ...
48                  ...
49                  ...
50                  ...
51                  ...
52                  ...
53                  ...
54                  ...
55                  ...
56                  ...
57                  ...
58                  ...
59                  ...
60                  ...
61                  ...
62                  ...
63                  ...
64                  ...
65                  ...
66                  ...
67                  ...
68                  ...
69                  ...
70                  ...
71                  ...
72                  ...
73                  ...

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

75      0 referencias
76      public static int ActualizarDVH(string pTabla, int PK, int PK2)
77      {
78          DVH mDVH = ObtenerValorDVH(pTabla, PK, PK2);
79          return ActualizarDVH(pTabla, mDVH);
80      }
81      2 referencias
82      public static int ActualizarDV(string pTabla, int PK)
83      {
84          Encriptador mCripto = new Encriptador();
85          DVH mDVH = ObtenerValorDVH(pTabla, PK);
86          mDVH.valorDVHEncryptado = mCripto.EncriptarReversible(mDVH.valorDVH.ToString());
87          ActualizarDVH(pTabla, mDVH);
88          return DVVBL.ActualizarDVV(pTabla);
89      }
90      3 referencias
91      public static int ActualizarDV(string pTabla, int PK, int PK2)
92      {
93          Encriptador mCripto = new Encriptador();
94          DVH mDVH = ObtenerValorDVH(pTabla, PK, PK2);
95          mDVH.valorDVHEncryptado = mCripto.EncriptarReversible(mDVH.valorDVH.ToString());
96          ActualizarDVH(pTabla, mDVH);
97          return DVVBL.ActualizarDVV(pTabla);
98      }
99      0 referencias
100     public static List<int> ObtenerValorDVHTotal(string pTabla, int PK, int PK2)
101     {
102         return DVHDAL.ObtenerValorDVHTotal(pTabla, PK, PK2);
103     }
104     1 referencia
105     public static bool ValidarConsistenciaDVH()
106     {

```

```

107         1 referencia
108         public static bool ValidarConsistenciaDVH()
109         {
110             List<string> mTablas = ListarTablas();
111             foreach (string mTabla in mTablas)
112             {
113                 ValidarTabla(mTabla);
114             }
115             return true;
116         }
117         1 referencia
118         public static List<string> ListarTablas()
119         {
120             List<string> mTablas = new List<string>();
121             mTablas.Add("cuenta_usuario");
122             mTablas.Add("bitacora");
123             return mTablas;
124         }
125         1 referencia
126         public static bool CalcularTodos()
127         {
128             List<string> mTablas = ListarTablas();
129             Encriptador mCripto = new Encriptador();
130             foreach (string mTabla in mTablas)
131             {
132                 List<DVH> mDVH = DVHDAL.ObtenerValoresDVH(mTabla);
133                 foreach (DVH x in mDVH)
134                 {
135                     x.valorDVHEncryptado = mCripto.EncriptarReversible(x.valorDVH.ToString());
136                     DVHDAL.ActualizarDVH(mTabla, x);
137                 }
138             }
139             return true;
140         }
141     }
142 }
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

DVVBL.cs

```

 9  namespace BLL
10 {
11     public class DVVBL
12     {
13         public static DVV Obtener(string pTabla)
14         {
15             return DVVDAL.Obtener(pTabla);
16         }
17
18         public static List<DVV> Obtener()
19         {
20             return DVVDAL.Obtener();
21         }
22
23         public static long CalcularDVV(string pTabla)
24         {
25             return DVVDAL.CalcularDVV(pTabla);
26         }
27
28         public static int ActualizarDVV(DVV pDVV)
29         {
30             return DVVDAL.ActualizarDVV(pDVV);
31         }
32
33         public static int ActualizarDVV(string pTabla)
34         {
35             DVV mDVV = Obtener(pTabla);
36             mDVV.valorDVV = CalcularDVV(pTabla);
37             return ActualizarDVV(mDVV);
38         }
39
40
41         public static bool ValidarDVV(DVV pDVV)
42         {
43             List<DVV> mValores = Obtener();
44             foreach (DVV x in mValores)
45             {
46                 x.valorDVV = CalcularDVV(x.tabla);
47                 bool aux = ValidarDVV(x);
48                 if (aux == false)
49                 {
50                     throw new ErrorConsistenciaDVVEception(x);
51                 }
52             }
53             return true;
54         }
55
56         public static List<string> ListarDVVsAlterados()
57         {
58             List<string> mTablas = new List<string>();
59             mTablas.Clear();
60             List<DVV> mValores = Obtener();
61             foreach (DVV x in mValores)
62             {
63                 x.valorDVV = CalcularDVV(x.tabla);
64                 bool aux = ValidarDVV(x);
65                 if (aux == false)
66                 {
67                     mTablas.Add("Se agregaron o eliminaron registros en la tabla" + x.tabla);
68                 }
69             }
70             return mTablas;
71         }
72
73

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

public static bool CalcularTodo()
{
    List<DVV> mDVV = DVVDAL.Obtener();
    foreach (DVV x in mDVV)
    {
        long aux = DVVDAL.CalcularDVV(x.tabla);
        x.valorDVV = aux;
        DVVDAL.ActualizarDVV(x);
    }
    return true;
}
}

```

FamiliaBL.cs

```

9  namespace BLL
10 {
11     2 referencias
12     public class FamiliaBL
13     {
14         1 referencia
15         public static List<Familia> Listar()
16         {
17             :
18             return FamiliaDAL.Listar();
19         }
20
21         0 referencias
22         public static int Guardar(Familia pFamilia)
23         {
24             :
25             return FamiliaDAL.Guardar(pFamilia);
26         }
27
28         0 referencias
29         public static Familia Obtener(int pId)
30         {
31             :
32             Familia mFamilia = FamiliaDAL.Obtener(pId);
33             return mFamilia;
34         }
35
36         0 referencias
37         public static Familia Obtener(string pNombre)
38         {
39             :
40             Familia mFamilia = FamiliaDAL.Obtener(pNombre);
41             return mFamilia;
42         }
43
44         4 referencias
45         public static List<Patente> ObtenerPatentes(Familia pFamilia)
46         {
47             :
48             List<Patente> mPatentes = FamiliaDAL.ObtenerPatentes(pFamilia);
49             return mPatentes;
50         }
51
52         0 referencias
53         public static int EncriptarTodo()
54     }
55 }

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

39      {
40          List<Familia> mFamilias = Listar();
41          Encriptador mCripo = new Encriptador();
42          foreach (Familia x in mFamilias)
43          {
44              x.familia_nombre = mCripo.EncriptarReversible(x.familia_nombre);
45              FamiliaDAL.Guardar(x);
46          }
47          return 1;
48      }
49      0 referencias
50      public static void Eliminar(Familia pFamilia)
51      {
52          List<Patente> mPatentes = ObtenerPatentes(pFamilia);
53          List<CuentaUsuario> mCuentasUsuario = CuentaUsuarioBL.ObtenerUsuarios(pFamilia);
54          if (mCuentasUsuario != null)
55          {
56              foreach (CuentaUsuario x in mCuentasUsuario)
57              {
58                  CuentaUsuarioBL.EliminarFamilia(pFamilia, x);
59              }
60              DVVBL.ActualizarDV("cuenta_usuario_familia");
61          }
62          if (mPatentes != null)
63          {
64              foreach (Patente x in mPatentes)
65              {
66                  EliminarPatente(pFamilia, x);
67              }
68              DVVBL.ActualizarDV("familia_patente");
69          }
70          FamiliaDAL.Eliminar(pFamilia);
71      }
72      0 referencias
73      public static int AgregarPatente(Familia pFamilia, Patente pPatente)

74      if (VerificarPatente(pPatente.patente_id, pFamilia) == false)
75      {
76          FamiliaDAL.AgregarPatente(pFamilia, pPatente);
77          DVVBL.ActualizarDV("familia_patente", pFamilia.familia_id, pPatente.patente_id);
78      }
79      else
80      {
81          throw new PatenteYaAsignadaException();
82      }
83      return 1;
84  }
85  1 referencia
86  public static bool VerificarPatente(int pPatenteId, Familia pFamilia)
87  {
88      List<Patente> mPatentes = ObtenerPatentes(pFamilia);
89      if (mPatentes != null)
90      {
91          bool resultado = mPatentes.Any(x => x.patente_id == pPatenteId);
92          return resultado;
93      }
94      else return false;
95  }
96  1 referencia
97  public static int EliminarPatente(Familia pFamilia, Patente pPatente)
98  {
99      int aux = FamiliaDAL.EliminarPatente(pFamilia, pPatente);
100     DVVBL.ActualizarDV("familia_patente");
101     return aux;
102 }
103

```

Patente.cs

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

8  namespace BLL
9  {
10 }
11 public class PatenteBL
12 {
13     public static List<Patente> Listar()
14     {
15         return PatenteDAL.Listar();
16     }
17     0 referencias
18     public static int Guardar(Patente pPatente)
19     {
20         return PatenteDAL.Guardar(pPatente);
21     }
22     0 referencias
23     public static Patente Obtener(int pId)
24     {
25         Patente mPatente = PatenteDAL.Obtener(pId);
26         return mPatente;
27     }
28     0 referencias
29     public static Patente Obtener(string pNombre)
30     {
31         Patente mPatente = PatenteDAL.Obtener(pNombre);
32         return mPatente;
33     }
34     1 referencia
35     public static List<Patente> ListarExcepcion(CuentaUsuario pCuentaUsuario)
36     {
37         List<Patente> mPatentes = PatenteDAL.ListarExcepcion(pCuentaUsuario);
38         return mPatentes;
39     }
40 }
```

```

37     public static int EncriptarTodo()
38     {
39         List<Patente> mPatentes = Listar();
40         Encryptador mCripto = new Encryptador();
41         foreach (Patente x in mPatentes)
42         {
43             x.patente_nombre = mCripto.EncriptarReversible(x.patente_nombre);
44             PatenteDAL.Guardar(x);
45         }
46         return 1;
47     }
48     1 referencia
49     public static List<CuentaUsuario> ListarUsuarios(Patente pPatente)
50     {
51         return PatenteDAL.ListarUsuarios(pPatente);
52     }
53 }
54 }
```

BackupadorBL.cs

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
7  namespace BLL
8  {
9      2 referencias
10     public class BackupeadorBL
11    {
12        1 referencia
13        public static int RealizarBackup(string pNombreArchivo, string pRuta, int pVolumenes)
14        {
15            return BackupeadorDAL.RealizarBackup(pNombreArchivo, pRuta, pVolumenes);
16        }
17        1 referencia
18        public static int RealizarRestore(string pNombreArchivo, string pRuta, int pVolumenes)
19        {
20            return BackupeadorDAL.RealizarRestore(pNombreArchivo, pRuta, pVolumenes);
21        }
22    }
```

ProductoBL.cs

```
8  namespace BLL
9  {
10     7 referencias
11     public class ProductoBL
12    {
13        1 referencia
14        public static List<Producto> Listar()
15        {
16            return ProductoDAL.Listar();
17        }
18        2 referencias
19        public static int Guardar(Producto pProducto)
20        {
21            return ProductoDAL.Guardar(pProducto);
22        }
23        1 referencia
24        public static Producto Obtener(int pId)
25        {
26            Producto mProducto = ProductoDAL.Obtener(pId);
27            return mProducto;
28        }
29        1 referencia
30        public static void Eliminar(Producto pProducto)
31        {
32            ProductoDAL.Eliminar(pProducto);
33        }
34        2 referencias
35        public static Producto Obtener(string pNombre)
36        {
37            Producto mProducto = ProductoDAL.Obtener(pNombre);
38            return mProducto;
39        }
40    }
```



UNIVERSIDAD ABIERTA INTERAMERICANA

Facultad de tecnología informática

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

Acceso a datos DAL

PatenteDAL.cs

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

9  namespace DAL
10 {
11     public class PatenteDAL
12     {
13         public static int mId;
14
15         public static int ProximoId()
16         {
17             if (mId == 0)
18             {
19                 DAO mDAOObject = new DAO();
20                 mId = mDAOObject.ObtenerId("Patente");
21             }
22             mId += 1;
23             return mId;
24         }
25
26         public static void ValorizarEntidad(Patente pPatente, DataRow pDr)
27         {
28             pPatente.patente_id = int.Parse(pDr["patente_id"].ToString());
29             pPatente.patente_nombre = pDr["patente_nombre"].ToString();
30         }
31         public static List<Patente> Listar()
32         {
33             DAO mDAOObject = new DAO();
34             DataSet mDs = new DataSet();
35             List<Patente> mPatentes = new List<Patente>();
36             mDs = mDAOObject.ExecuteDataSet("select patente_id, patente_nombre from patente");
37
38             if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
39             {
40
41                 foreach (DataRow mDr in mDs.Tables[0].Rows)
42                 {
43                     Patente mPatente = new Patente(int.Parse(mDr["patente_id"].ToString()));
44                     ValorizarEntidad(mPatente, mDr);
45                     mPatentes.Add(mPatente);
46                 }
47             }
48             return mPatentes;
49         }
50         public static Patente Obtener(int pId)
51         {
52             DAO mDAOObject = new DAO();
53             DataSet mDs = mDAOObject.ExecuteDataSet("select patente_id, patente_nombre from patente where patente_id = " + pId);
54             if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
55             {
56                 Patente mPatente = new Patente(pId);
57                 ValorizarEntidad(mPatente, mDs.Tables[0].Rows[0]);
58                 return mPatente;
59             }
59             else return null;
60         }
61         public static Patente Obtener(string pNombre)
62         {
63             DAO mDAOObject = new DAO();
64             DataSet mDs = mDAOObject.ExecuteDataSet("select patente_id, patente_nombre from patente where patente_nombre = '" + pNombre + "'");
65             if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
66             {
67                 int pId = int.Parse(mDs.Tables[0].Rows[0]["patente_id"].ToString());
68                 Patente mPatente = new Patente(pId);
69                 ValorizarEntidad(mPatente, mDs.Tables[0].Rows[0]);
70                 return mPatente;
71             }
72         }
73     }

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

76      public static List<Patente> ListarExcepto(CuentaUsuario pCuentaUsuario)
77      {
78          DAO mDAOObject = new DAO();
79          DataSet mDs = new DataSet();
80          List<Patente> mPatentes = new List<Patente>();
81          mDs = mDAOObject.ExecuteDataSet("select t1.patente_id, t1.patente_nombre from patente t1 left join cuenta_usuario_patente t2 on t1.patente_id = t2.p
82
83          if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
84          {
85              foreach (DataRow mDr in mDs.Tables[0].Rows)
86              {
87                  Patente mPatente = new Patente(int.Parse(mDr["patente_id"].ToString()));
88                  ValorizarEntidad(mPatente, mDr);
89                  mPatentes.Add(mPatente);
90              }
91          }
92          return mPatentes;
93      }
94      2 referencias
95      public static int Guardar(Patente pPatente)
96      {
97          DAO mDAOObject = new DAO();
98          string pcadenaComando;
99          pPatente.patente_id = ProximoId();
100         pcadenaComando = "insert into patente(patente_id, patente_nombre) values (" + pPatente.patente_id + ", '" + pPatente.patente_nombre + "')";
101
102         return mDAOObject.ExecuteNonQuery(pcadenaComando);
103     }
104     1 referencia
105     public static List<CuentaUsuario> ListarUsuarios(Patente pPatente)
106     {
107         DAO mDAOObject = new DAO();
108         DataSet mDs = new DataSet();
109         List<CuentaUsuario> mCuentasUsuario = new List<CuentaUsuario>();
110         string pcadenaComando = "select t2.cuenta_usuario_id, t2.cuenta_usuario_username, t2.cuenta_usuario_password, t2.cuenta_usuario_intentos_login, t2.
111         mDs = mDAOObject.ExecuteDataSet(pcadenaComando);

112         if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
113         {
114             foreach (DataRow mDr in mDs.Tables[0].Rows)
115             {
116                 CuentaUsuario mCuentaUsuario = new CuentaUsuario(int.Parse(mDr["cuenta_usuario_id"].ToString()));
117                 CuentaUsuarioDAL.ValorizarEntidad(mCuentaUsuario, mDr);
118                 mCuentasUsuario.Add(mCuentaUsuario);
119             }
120         }
121     }
122 }

```

FamiliaDAL.cs

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

8   namespace DAL
9   {
10      10 referencias
11      public class FamiliaDAL
12      {
13
14         public static int mId;
15
16         1 referencia
17         public static int ProximoId()
18         {
19             if (mId == 0)
20             {
21                 DAO mDAOObject = new DAO();
22                 mId = mDAOObject.ObtenerId("Familia");
23             }
24             mId += 1;
25             return mId;
26         }
27
28         4 referencias
29         public static void ValorizarEntidad(Familia pFamilia, DataRow pDr)
30         {
31             pFamilia.familia_id = int.Parse(pDr["Familia_id"].ToString());
32             pFamilia.familia_nombre = pDr["familia_nombre"].ToString();
33         }
34
35         1 referencia
36         public static List<Familia> Listar()
37         {
38             DAO mDAOObject = new DAO();
39             DataSet mDs = new DataSet();
40             List<Familia> mFamilias = new List<Familia>();
41             mDs = mDAOObject.ExecuteDataSet("select Familia_id, familia_nombre from Familia");
42
43             if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
44             {
45                 foreach (DataRow mDr in mDs.Tables[0].Rows)
46                 {
47                     Familia mFamilia = new Familia(int.Parse(mDr["Familia_id"].ToString()));
48                     ValorizarEntidad(mFamilia, mDr);
49                     mFamilias.Add(mFamilia);
50                 }
51             }
52             return mFamilias;
53         }
54
55         1 referencia
56         public static Familia Obtener(int pId)
57         {
58             DAO mDAOObject = new DAO();
59             DataSet mDs = mDAOObject.ExecuteDataSet("select Familia_id, Familia_nombre from Familia where Familia_id = " + pId);
60             if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
61             {
62                 Familia mFamilia = new Familia(pId);
63                 ValorizarEntidad(mFamilia, mDs.Tables[0].Rows[0]);
64                 return mFamilia;
65             }
66             else return null;
67         }
68
69         1 referencia
70         public static Familia Obtener(string pNombre)
71         {
72             DAO mDAOObject = new DAO();
73             DataSet mDs = mDAOObject.ExecuteDataSet("select Familia_id, Familia_nombre from Familia where Familia_nombre = '" + pNombre + "'");
74             if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
75             {
76                 int pId = int.Parse(mDs.Tables[0].Rows[0]["familia_id"].ToString());
77                 Familia mFamilia = new Familia(pId);
78                 ValorizarEntidad(mFamilia, mDs.Tables[0].Rows[0]);
79                 return mFamilia;
80             }
81             else return null;
82         }

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

79     public static int Eliminar(Familia pFamilia)
80     {
81         DAO mDAOObject = new DAO();
82         string pCadenaComando = "delete from familia where familia_id = " + pFamilia.familia_id;
83         return mDAOObject.ExecuteNonQuery(pCadenaComando);
84     }
85
86     1 referencia
87     public static List<Patente> ObtenerPatentes(Familia pFamilia)
88     {
89         List<Patente> mPatentes = new List<Patente>();
90         DAO mDAOObject = new DAO();
91         DataSet mDs = mDAOObject.ExecuteDataSet("select patente_id from Familia_patente where Familia_id = " + pFamilia.familia_id);
92         if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
93         {
94             foreach (DataRow mDr in mDs.Tables[0].Rows)
95             {
96                 Patente mPatente = PatenteDAL.Obtener(int.Parse(mDr["patente_id"].ToString()));
97                 mPatentes.Add(mPatente);
98             }
99         }
100        else return null;
101    }
102    2 referencias
103    public static int Guardar(Familia pFamilia)
104    {
105        DAO mDAOObject = new DAO();
106        string pCadenaComando;
107        if (pFamilia.familia_id == 0)
108        {
109            pFamilia.familia_id = ProximoId();
110            pCadenaComando = "insert into familia(familia_id, familia_nombre) values (" + pFamilia.familia_id + ", '" + pFamilia.familia_nombre + "')";
111        }
112        else pCadenaComando = "update familia set familia_nombre = '" + pFamilia.familia_nombre + "' where familia_id = " + pFamilia.familia_id;
113        return mDAOObject.ExecuteNonQuery(pCadenaComando);
114
115    }
116    1 referencia
117    public static int AgregarPatente(Familia pFamilia, Patente pPatente)
118    {
119        DAO mDAOObject = new DAO();
120        string pCadenaComando;
121        pCadenaComando = "insert into familia_patente(familia_id, patente_id, familia_patente_dvh) values (" + pFamilia.familia_id + ", " + pPatente.pa
122    }
123    1 referencia
124    public static int EliminarPatente(Familia pFamilia, Patente pPatente)
125    {
126        DAO mDAOObject = new DAO();
127        string pCadenaComando;
128        pCadenaComando = "delete from familia_patente where familia_id = " + pFamilia.familia_id + " and patente_id = " + pPatente.patente_id;
129    }
130
131
132
133
134

```

ProductoDAL.cs

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

8  namespace DAL
9  {
10     5 referencias
11     public class ProductoDAL
12     {
13         public static int mId;
14
15         1 referencia
16         public static int ProximoId()
17         {
18             if (mId == 0)
19             {
20                 DAO mDAOObject = new DAO();
21                 mId = mDAOObject.ObtenerId("Producto");
22             }
23             mId += 1;
24             return mId;
25         }
26
27         3 referencias
28         public static void ValorizarEntidad(Producto pProducto, DataRow pDr)
29         {
30             pProducto.producto_id = int.Parse(pDr["producto_id"].ToString());
31             pProducto.producto_nombre = pDr["producto_nombre"].ToString();
32         }
33
34         1 referencia
35         public static List<Producto> Listar()
36         {
37             DAO mDAOObject = new DAO();
38             DataSet mDs = new DataSet();
39             List<Producto> mProductos = new List<Producto>();
40             mDs = mDAOObject.ExecuteDataSet("select producto_id, Producto_nombre from Producto");
41
42             if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
43             {
44                 foreach (DataRow mDr in mDs.Tables[0].Rows)
45                 {
46                     Producto mProducto = new Producto(int.Parse(mDr["Producto_id"].ToString()));
47                     ValorizarEntidad(mProducto, mDr);
48                     mProductos.Add(mProducto);
49                 }
50             }
51             return mProductos;
52         }
53
54         1 referencia
55         public static Producto Obtener(int pId)
56         {
57             DAO mDAOObject = new DAO();
58             DataSet mDs = mDAOObject.ExecuteDataSet("select Producto_id, Producto_nombre from Producto where Producto_id = " + pId);
59             if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
60             {
61                 Producto mProducto = new Producto(pId);
62                 ValorizarEntidad(mProducto, mDs.Tables[0].Rows[0]);
63                 return mProducto;
64             }
65             else return null;
66         }
67
68         1 referencia
69         public static Producto Obtener(string pNombre)
70         {
71             DAO mDAOObject = new DAO();
72             DataSet mDs = mDAOObject.ExecuteDataSet("select Producto_id, Producto_nombre from Producto where Producto_nombre = '" + pNombre + "'");
73             if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
74             {
75                 int pId = int.Parse(mDs.Tables[0].Rows[0]["Producto_id"].ToString());
76                 Producto mProducto = new Producto(pId);
77                 ValorizarEntidad(mProducto, mDs.Tables[0].Rows[0]);
78                 return mProducto;
79             }
80             else return null;
81         }
82     }
83 }
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

75  public static int Eliminar(Producto pProducto)
76  {
77      DAO mDAOObject = new DAO();
78      string pCadenaComando = "delete from producto where producto_id = " + pProducto.producto_id;
79      return mDAOObject.ExecuteNonQuery(pCadenaComando);
80  }
81
82
83  public static int Guardar(Producto pProducto)
84  {
85      DAO mDAOObject = new DAO();
86      string pCadenaComando;
87
88
89      if (pProducto.producto_id == 0)
90      {
91          pProducto.producto_id = ProximoId();
92          pCadenaComando = "insert into Producto(Producto_id, Producto_nombre) values (" + pProducto.producto_id + ", '" + pProducto.producto_nombre + "'";
93      }
94      else pCadenaComando = "update Producto set producto_nombre = '" + pProducto.producto_nombre + "' where producto_id = " + pProducto.producto_id;
95      return mDAOObject.ExecuteNonQuery(pCadenaComando);
96  }
97
98  }
99

```

DVVDAL.cs

```

9  namespace DAL
10 {
11     7 referencias
12     public class DVVDAL
13     {
14         2 referencias
15         public static DVV Obtener(string pTabla)
16         {
17             Encriptador mCripto = new Encriptador();
18             DAO mDAOObject = new DAO();
19             DataSet mDs = new DataSet();
20             string pCadena = "select * from DVV where tabla = '" + pTabla + "'";
21             mDs = mDAOObject.ExecuteDataSet(pCadena);
22             DVV mDVV = new DVV();
23             if (mDs.Tables[0].Rows.Count > 0)
24             {
25                 mDVV.tabla = pTabla;
26                 DataRow mDr = mDs.Tables[0].Rows[0];
27                 mDVV.valorDVVBase = int.Parse(mCripto.Desencriptar(mDr["dvv_valor"].ToString()));
28             }
29             else { mDVV.tabla = pTabla; }
30             return mDVV;
31
32         2 referencias
33         public static List<DVV> Obtener()
34         {
35             Encriptador mCripto = new Encriptador();
36             DAO mDAOObject = new DAO();
37             DataSet mDs = new DataSet();
38             string pCadena = "select * from DVV";
39             mDs = mDAOObject.ExecuteDataSet(pCadena);
40             List<DVV> mValores = new List<DVV>();
41             if (mDs.Tables[0].Rows.Count > 0)
42             {
43                 foreach (DataRow mDr in mDs.Tables[0].Rows)
44             }
45         }
46     }
47 }

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

42
43
44     DVV mDVV = new DVV();
45     mDVV.tabla = mDr["tabla"].ToString();
46     mDVV.valorDVVBase = long.Parse(mCripto.Desencriptar(mDr["dvv_valor"].ToString()));
47     mValores.Add(mDVV);
48 }
49
50     return mValores;
51 }
52
53     2 referencias
54     public static long CalcularDVV(string pTabla)
55     {
56         Encriptador mCripto = new Encriptador();
57         DAO mDAOobject = new DAO();
58         DataSet mDs = new DataSet();
59         string mNombreCampo = pTabla + "_dvh";
60         string pCadena = "select " + mNombreCampo + " from " + pTabla;
61         mDs = mDAOobject.ExecuteDataSet(pCadena);
62         string mNombre = pTabla + "_dvh";
63         long mSuma = 0;
64         if (mDs.Tables[0].Rows.Count > 0)
65         {
66             foreach (DataRow x in mDs.Tables[0].Rows)
67             {
68                 if (x[0].ToString() != "0")
69                 {
70                     string mValorEncriptado = x[0].ToString();
71                     string mValorDesencriptado = mCripto.Desencriptar(mValorEncriptado);
72                     long mValorFinal = long.Parse(mValorDesencriptado);
73                     mSuma += mValorFinal;
74                 }
75             }
76         }
77         return mSuma;
78
79     1 referencias
80     public static int ActualizarDVV(DVV pDVV)
81     {
82         Encriptador mCripto = new Encriptador();
83         DAO mDAOobject = new DAO();
84         string pCadenaComando;
85         DVV mDVV = Obtener(pDVV.tabla);
86         if (pDVV.valorDVVBase !=0)
87         {
88             pCadenaComando = "update DVV set dvv_valor = '" + mCripto.EncryptReversible(pDVV.valorDVV.ToString()) + "' where tabla = '" + pDVV.tabla + "'";
89         }
90         else
91         {
92             pCadenaComando = "insert into DVV(tabla, dvv_valor) values ('" + pDVV.tabla + "', '" + mCripto.EncryptReversible(pDVV.valorDVV.ToString()) + ")";
93         }
94         return mDAOobject.ExecuteNonQuery(pCadenaComando);
95     }
96 }

```

DVHDAL.cs

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

11  // Recorridos:
12  public class DVHDAL
13  {
14      Encriptador mCripto = new Encriptador();
15      2 referencias
16      public static List<DVH> ObtenerValoresDVH(string pTabla)
17      {
18          DAO mDAOObject = new DAO();
19          DataSet mDs = new DataSet();
20          string pCadena = "select * from " + pTabla;
21          mDAOObject.ExecuteDataSet(pCadena);
22          List<DVH> mValores = new List<DVH>();
23          List<string> mNombreCamposClave = ClavePrimaria(pTabla);
24          List<string> mCamposEncriptados = CamposEncriptados(pTabla);
25          int cantColumnas = mDs.Tables[0].Columns.Count - 1;
26          if (mDs.Tables[0].Rows.Count > 0)
27          {
28              Encriptador mCripto = new Encriptador();
29              List<int> mIndexEncriptado = new List<int>();
30              foreach (DataColumn y in mDs.Tables[0].Columns)
31              {
32                  if (mCamposEncriptados.Contains(y.ColumnName))
33                  {
34                      mIndexEncriptado.Add(y.Ordinal);
35                  }
36              }
37              foreach (DataRow x in mDs.Tables[0].Rows)
38              {
39                  DVH mDVH = new DVH();
40                  int mValorAcumulado = 0;
41                  for (int i = 0; i < cantColumnas; i++)
42                  {
43                      if (mIndexEncriptado.Contains(i))
44                      {
45                          mValorAcumulado += ConvertirValor(mCripto.Desencriptar(x[i].ToString()));
46
47                      else mValorAcumulado += ConvertirValor(x[i].ToString());
48
49                  }
50                  mDVH.tabla = pTabla;
51                  string NombrePK1 = mNombreCamposClave[0];
52                  string NombrePK2;
53                  if (mNombreCamposClave.Count == 2)
54                  {
55                      NombrePK2 = mNombreCamposClave[1];
56                      mDVH.clavePrimaria2 = int.Parse(x[NombrePK2].ToString());
57
58                  }
59                  mDVH.clavePrimaria = int.Parse(x[NombrePK1].ToString());
60                  mDVH.valorDVH = mValorAcumulado;
61                  string campoDVH = pTabla + "_dvh";
62                  if (x[campoDVH].ToString() != "" & x[campoDVH].ToString() != "0")
63                  {
64                      mDVH.valorDVHBase = long.Parse(mCripto.Desencriptar(x[campoDVH].ToString()));
65
66                  }
67                  mValores.Add(mDVH);
68
69      }
70      2 referencias
71      public static int ActualizarDVH(string pTabla, DVH mDVH)
72      {
73          DAO mDAOObject = new DAO();
74          string pCadenaComando;
75          List<string> mNombreCamposClave = ClavePrimaria(pTabla);
76          if (mNombreCamposClave.Count == 2)
77          {
78              pCadenaComando = "update " + pTabla + " set " + pTabla + "_dvh = '" + mDVH.valorDVHEncriptado + "' where " + mNombreCamposClave[0] + " = " + mD
    
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
    pCadenaComando = "update " + pTabla + " set " + pTabla + "_dvh = '" + mDVH.valorDVHEncryptado + "' where " + mNombreCamposClave[0] + " = " + mD
    return mDAOObject.ExecuteNonQuery(pCadenaComando);
}

1 referencia
public static DVH ObtenerValorDVH(string pTabla, int PK)
{
    DAO mDAOObject = new DAO();
    DataSet mDs = new DataSet();
    List<string> mNombreCamposClave = ClavePrimaria(pTabla);
    List<string> mCamposEncriptados = CamposEncriptados(pTabla);
    string pCadena = "select * from " + pTabla + " where " + mNombreCamposClave[0] + " = " + PK;
    mDs = mDAOObject.ExecuteDataSet(pCadena);
    int cantColumnas = mDs.Tables[0].Columns.Count - 1;
    DVH mDVH = new DVH();
    if (mDs.Tables[0].Rows.Count > 0)
    {
        Encriptador mCripto = new Encriptador();
        List<int> mIndexEncriptado = new List<int>();
        foreach (DataColumn y in mDs.Tables[0].Columns)
        {
            if (mCamposEncriptados.Contains(y.ColumnName))
            {
                mIndexEncriptado.Add(y.Ordinal);
            }
        }
        DataRow x = mDs.Tables[0].Rows[0];
        int mValorAcumulado = 0;
        for (int i = 0; i < cantColumnas; i++)
        {
            if (mIndexEncriptado.Contains(i))
            {
                mValorAcumulado += ConvertirValor(mCripto.Desencriptar(x[i].ToString()));
            }
            else mValorAcumulado += ConvertirValor(x[i].ToString());
        }
    }
    mDVH.tabla = pTabla;
    string NombrePK1 = mNombreCamposClave[0];
    mDVH.clavePrimaria = int.Parse(x[NombrePK1].ToString());
    mDVH.valorDVH = mValorAcumulado;
    string campoDVH = pTabla + "_dvh";
    if (x[campoDVH].ToString() != "" & x[campoDVH].ToString() != "0")
    {
        mDVH.valorDVHBase = long.Parse(mCripto.Desencriptar(x[campoDVH].ToString()));
    }
}
return mDVH;
}

1 referencia
public static List<int> ObtenerValorDVHTotal(string pTabla, int PK, int PK2)
{
    DAO mDAOObject = new DAO();
    DataSet mDs = new DataSet();
    List<string> mNombreCamposClave = ClavePrimaria(pTabla);
    List<string> mCamposEncriptados = CamposEncriptados(pTabla);
    string pCadena = "select * from " + pTabla + " where " + mNombreCamposClave[0] + " = " + PK + " and " + mNombreCamposClave[1] + " = " + PK2;
    mDs = mDAOObject.ExecuteDataSet(pCadena);
    int cantColumnas = mDs.Tables[0].Columns.Count - 1;
    List<int> mValores = new List<int>();
    DVH mDVH = new DVH();
    if (mDs.Tables[0].Rows.Count > 0)
    {
        Encriptador mCripto = new Encriptador();
        List<int> mIndexEncriptado = new List<int>();
        foreach (DataColumn y in mDs.Tables[0].Columns)
        {
            if (mCamposEncriptados.Contains(y.ColumnName))
            {
                mIndexEncriptado.Add(y.Ordinal);
            }
        }
    }
}

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
149
150     DataRow x = mDs.Tables[0].Rows[0];
151     int mValorAcumulado = 0;
152     for (int i = 0; i < cantColumnas; i++)
153     {
154         if (mIndexEncriptado.Contains(i))
155         {
156             mValorAcumulado += ConvertirValor(mCripto.Desencriptar(x[i].ToString()));
157         }
158         else mValorAcumulado += ConvertirValor(x[i].ToString());
159     }
160     mDVH.tabla = pTabla;
161     string NombrePK1 = mNombreCamposClave[0];
162     string NombrePK2 = mNombreCamposClave[1];
163     mDVH.clavePrimaria = int.Parse(x[NombrePK1].ToString());
164     mDVH.clavePrimaria2 = int.Parse(x[NombrePK2].ToString());
165     mDVH.valorDVH = mValorAcumulado;
166     string campoDVH = pTabla + "_dvh";
167     if (x[campoDVH].ToString() != "" & x[campoDVH].ToString() != "0")
168     {
169         mDVH.valorDVHBase = long.Parse(mCripto.Desencriptar(x[campoDVH].ToString()));
170     }
171 }
172 return mValores;
173 }
174 }

175 //referencia
176 public static DVH ObtenerValorDVH(string pTabla, int PK, int PK2)
177 {
178     DAO mDAOobject = new DAO();
179     DataSet mDs = new DataSet();
180     string mNombreCamposClave = ClavePrimaria(pTabla);
181     List<string> mCamposEncriptados = CamposEncriptados(pTabla);
182     string pCadena = "select * from " + pTabla + " where " + mNombreCamposClave[0] + " = " + PK + " and " + mNombreCamposClave[1] + " = " + PK2;
183     mDs = mDAOobject.ExecuteDataSet(pCadena);
184     List<DVH> mValores = new List<DVH>();
```



```
184
185     int cantColumnas = mDs.Tables[0].Columns.Count - 1;
186     DVH mDVH = new DVH();
187
188     if (mDs.Tables[0].Rows.Count > 0)
189     {
190         Encriptador mCripto = new Encriptador();
191         List<int> mIndexEncriptado = new List<int>();
192         foreach (DataColumn y in mDs.Tables[0].Columns)
193         {
194             if (mCamposEncriptados.Contains(y.ColumnName))
195             {
196                 mIndexEncriptado.Add(y.Ordinal);
197             }
198         }
199         DataRow x = mDs.Tables[0].Rows[0];
200         int mValorAcumulado = 0;
201         for (int i = 0; i < cantColumnas; i++)
202         {
203             if (mIndexEncriptado.Contains(i))
204             {
205                 mValorAcumulado += ConvertirValor(mCripto.Desencriptar(x[i].ToString()));
206             }
207             else mValorAcumulado += ConvertirValor(x[i].ToString());
208         }
209         mDVH.tabla = pTabla;
210         string NombrePK1 = mNombreCamposClave[0];
211         string NombrePK2 = mNombreCamposClave[1];
212         mDVH.clavePrimaria = int.Parse(x[NombrePK1].ToString());
213         mDVH.clavePrimaria2 = int.Parse(x[NombrePK2].ToString());
214         mDVH.valorDVH = mValorAcumulado;
215         string campoDVH = pTabla + "_dvh";
216         if (x[campoDVH].ToString() != "" & x[campoDVH].ToString() != "0")
217         {
218             mDVH.valorDVHBase = long.Parse(mCripto.Desencriptar(x[campoDVH].ToString()));
219         }
220     }
221 }
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

219     }
220     return mDVH;
221   }
222 
223   5 referencias
224   public static List<string> ClavePrimaria(string pTabla)
225   {
226     List<string> PK = new List<string>();
227     if (pTabla == "cliente") PK.Add("cliente_id");
228     if (pTabla == "familia_patente") { PK.Add("familia_id"); PK.Add("patente_id"); }
229     if (pTabla == "cuenta_usuario_patente") { PK.Add("cuenta_usuario_id"); PK.Add("patente_id"); }
230     if (pTabla == "cuenta") PK.Add("cuenta_id");
231     if (pTabla == "bitacora") PK.Add("bitacora_id");
232     if (pTabla == "cuenta_usuario_familia") { PK.Add("cuenta_usuario_id"); PK.Add("familia_id"); }
233     if (pTabla == "cuenta_usuario") PK.Add("cuenta_usuario_id");
234     return PK;
235   }
236 
237   4 referencias
238   public static List<string> CamposEncriptados(string pTabla)
239   {
240     List<string> mCampos = new List<string>();
241     if (pTabla == "bitacora") mCampos.Add("bitacora_dvh");
242     if (pTabla == "bitacora_tipo_movimiento") mCampos.Add("bitacora_tipo_movimiento_desc");
243     if (pTabla == "cliente") { mCampos.Add("cliente_dvh"); mCampos.Add("cliente_doc_tipo"); mCampos.Add("cliente_doc_numero"); }
244     return mCampos;
245   }
246 
247   8 referencias
248   public static int ConvertirValor(string pCadena)
249   {
250     string textoOriginal = pCadena;
251     List<int> valores = new List<int>();
252     foreach (char c in textoOriginal)
253     {
254       int valor = System.Convert.ToInt32(c);
255       valores.Add(valor);
256     }
257   }
258 }
```

DAO.cs

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

12  public class DAO
13  {
14      public SqlConnection mCon = new SqlConnection("Data Source=.;Initial Catalog=GlobalLogistics;Integrated Security=True");
15
16      29 referencias
17      public DataSet ExecuteDataSet(string pCadenaComando)
18      {
19          DataSet mDs = new DataSet();
20          SqlDataAdapter mDa = new SqlDataAdapter(pCadenaComando, mCon);
21          mDa.Fill(mDs);
22          if (mCon.State != ConnectionState.Closed) mCon.Close();
23          return mDs;
24      }
25
26      17 referencias
27      public int ExecuteNonQuery(string pCommandText)
28      {
29          try
30          {
31              SqlCommand mCom = new SqlCommand(pCommandText, this.mCon);
32              this.mCon.Open();
33              int resultado = mCom.ExecuteNonQuery();
34              SqlConnection mCon = new SqlConnection("Data Source=.;Initial Catalog=Globallogistics;Integrated Security=True");
35              return resultado;
36          }
37          catch
38          {
39              throw;
40          }
41          finally
42          {
43              if (mCon.State != ConnectionState.Closed)
44                  mCon.Close();
45          }
46      }

```

```

46  public int ExecuteNonQuery(string pCommandText, string pDataBase)
47  {
48      try
49      {
50          SqlCommand mCom = new SqlCommand(pCommandText, mCon);
51          mCon.Open();
52          mCon.ChangeDatabase(pDataBase);
53          return mCom.ExecuteNonQuery();
54      }
55      catch
56      {
57          throw;
58      }
59      finally
60      {
61          if (mCon.State != ConnectionState.Closed)
62              mCon.Close();
63      }
64  }
65
66  5 referencias
67  public int ObtenerId(string pTabla)
68  {
69      try
70      {
71          SqlCommand mCom = new SqlCommand("SELECT ISNULL(MAX(" + pTabla + "_Id),0) FROM " + pTabla, mCon);
72          mCon.Open();
73          return int.Parse(mCom.ExecuteScalar().ToString());
74      }
75      catch (Exception ex)
76      {
77          throw (ex);
78      }
79      finally
80      {
81          if (mCon.State != ConnectionState.Closed)
82              mCon.Close();
83      }
84  }
85
86

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

CuentaUsuarioDAL.cs

```

10   13 referencias
11  public class CuentaUsuarioDAL
12  {
13      public static int mId;
14      1 referencia
15      public static int ProximoId()
16      {
17          if (mId == 0)
18          {
19              DAO mDAOObject = new DAO();
20              mId = mDAOObject.ObtenerId("Cuenta_usuario");
21          }
22          mId += 1;
23          return mId;
24      }
25      1 referencia
26      public static int Guardar(CuentaUsuario pCuentaUsuario)
27      {
28          DAO mDAOObject = new DAO();
29          string pCadenaComando;
30          if (pCuentaUsuario.Cuenta_usuario_id == 0)
31          {
32              pCuentaUsuario.Cuenta_usuario_id = ProximoId();
33              pCadenaComando = "insert into cuenta_usuario(Cuenta_usuario_id, Cuenta_usuario_username, Cuenta_usuario_password, Cuenta_usuario_intentos_login";
34          }
35          else pCadenaComando = "update Cuenta_Usuario set Cuenta_usuario_username = '" + pCuentaUsuario.Cuenta_usuario_username + "', Cuenta_usuario_password = '" + pCuentaUsuario.Cuenta_usuario_password + "' where Cuenta_usuario_id = " + pCuentaUsuario.Cuenta_usuario_id;
36          return mDAOObject.ExecuteNonQuery(pCadenaComando);
37      }
38      1 referencia
39      public static int Eliminar(CuentaUsuario pCuentaUsuario)
40      {
41          DAO mDAOObject = new DAO();
42          string pCadenaComando = "update Cuenta_Usuario set cuenta_usuario_activa = 0 where cuenta_usuario_id = " + pCuentaUsuario.Cuenta_usuario_id;
43          return mDAOObject.ExecuteNonQuery(pCadenaComando);
44      }
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

70     public static void ValorizarEntidad(CuentaUsuario pCuentaUsuario, DataRow pDr)
71     {
72         pCuentaUsuario.Cuenta_usuario_id = int.Parse(pDr["Cuenta_usuario_id"].ToString());
73         pCuentaUsuario.Cuenta_usuario_username = pDr["Cuenta_usuario_username"].ToString();
74         pCuentaUsuario.Cuenta_usuario_password = pDr["Cuenta_usuario_password"].ToString();
75         pCuentaUsuario.Cuenta_usuario_intentos_login = int.Parse(pDr["Cuenta_usuario_intentos_login"].ToString());
76         pCuentaUsuario.SetFechaAlta(int.Parse(pDr["dia"].ToString()), int.Parse(pDr["mes"].ToString()), int.Parse(pDr["anio"].ToString()));
77     }
78 }
79
80 1 referencia
81 public static List<CuentaUsuario> Listar()
82 {
83     DAO mDAOObject = new DAO();
84     DataSet mDs = new DataSet();
85     List<CuentaUsuario> mCuentaUsuarios = new List<CuentaUsuario>();
86     mDs = mDAOObject.ExecuteDataSet("select Cuenta_usuario_id, Cuenta_usuario_username, Cuenta_usuario_password, Cuenta_usuario_intentos_login, cuenta_
87     if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
88     {
89         foreach (DataRow mDr in mDs.Tables[0].Rows)
90         {
91             CuentaUsuario mCuentaUsuario = new CuentaUsuario(int.Parse(mDr["Cuenta_usuario_id"].ToString()));
92             ValorizarEntidad(mCuentaUsuario, mDr);
93             mCuentaUsuarios.Add(mCuentaUsuario);
94         }
95     }
96     return mCuentaUsuarios;
97 }
98
99 1 referencia
100 public static List<Patente> ObtenerPatentes(CuentaUsuario pCuentaUsuario)
101 {
102     List<Patente> mPatentes = new List<Patente>();
103     DAO mDAOObject = new DAO();
104     DataSet mDs = new DataSet();
105
106     mDs = mDAOObject.ExecuteDataSet("select t1.patente_id, t2.patente_nombre from cuenta_usuario_patente t1 left join patente t2 on t1.patente_id = t2.p
107     if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
108     {
109         foreach (DataRow mDr in mDs.Tables[0].Rows)
110         {
111             Patente mPatente = new Patente(int.Parse(mDr["patente_id"].ToString()));
112             PatenteDAL.ValorizarEntidad(mPatente, mDr);
113             mPatentes.Add(mPatente);
114         }
115     }
116     return mPatentes;
117 }
118
119 1 referencia
120 public static List<Familia> ObtenerFamilias(CuentaUsuario pCuentaUsuario)
121 {
122     List<Familia> mFamilias = new List<Familia>();
123     DAO mDAOObject = new DAO();
124     DataSet mDs = new DataSet();
125     mDs = mDAOObject.ExecuteDataSet("select t1.familia_id, t2.familia_nombre from cuenta_usuario_familia t1 left join familia t2 on t1.familia_id = t2.f
126     if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
127     {
128         foreach (DataRow mDr in mDs.Tables[0].Rows)
129         {
130             Familia mFamilia = new Familia(int.Parse(mDr["familia_id"].ToString()));
131             FamiliaDAL.ValorizarEntidad(mFamilia, mDr);
132             mFamilias.Add(mFamilia);
133         }
134     }
135     return mFamilias;
136 }
137
138 1 referencia
139 public static int AgregarPatente(Patente pPatente, CuentaUsuario pCuentaUsuario)
140 {
141 }
```

Materia: Modelos computacionales de Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
138 DAO mDAOObject = new DAO();
139 string pCadenaComando;
140 pCadenaComando = "insert into cuenta_usuario_patente (cuenta_usuario_id, patente_id) values (" + pCuentaUsuario.Cuenta_usuario_id + ", " + pPatente.Id;
141 return mDAOObject.ExecuteNonQuery(pCadenaComando);
142 }

144 1 referencia
145 public static int EliminarPatente(Patente pPatente, CuentaUsuario pCuentaUsuario)
146 {
147     DAO mDAOObject = new DAO();
148     string pCadenaComando;
149     pCadenaComando = "delete from cuenta_usuario_patente where cuenta_usuario_id = " + pCuentaUsuario.Cuenta_usuario_id + " and patente_id = " + pPatente.Id;
150     return mDAOObject.ExecuteNonQuery(pCadenaComando);
151 }
152 1 referencia
153 public static int AgregarFamilia(Familia pFamilia, CuentaUsuario pCuentaUsuario)
154 {
155     DAO mDAOObject = new DAO();
156     string pCadenaComando;
157     pCadenaComando = "insert into cuenta_usuario_familia(cuenta_usuario_id, familia_id) values (" + pCuentaUsuario.Cuenta_usuario_id + ", " + pFamilia.Id;
158     return mDAOObject.ExecuteNonQuery(pCadenaComando);
159 }
160 1 referencia
161 public static int EliminarFamilia(Familia pFamilia, CuentaUsuario pCuentaUsuario)
162 {
163     DAO mDAOObject = new DAO();
164     string pCadenaComando;
165     pCadenaComando = "delete from cuenta_usuario_familia where cuenta_usuario_id = " + pCuentaUsuario.Cuenta_usuario_id + " and familia_id = " + pFamilia.Id;
166     return mDAOObject.ExecuteNonQuery(pCadenaComando);
167 }
168 1 referencia
169 public static List<CuentaUsuario> ObtenerUsuarios(Familia pFamilia)
170 {
171     DAO mDAOObject = new DAO();
172     List<CuentaUsuario> mCuentasUsuario = new List<CuentaUsuario>();
173 }

174 List<CuentaUsuario> mCuentasUsuario = new List<CuentaUsuario>();
175 DataSet mDs = mDAOObject.ExecuteDataSet("select t2.cuenta_usuario_id, t2.cuenta_usuario_username, t2.cuenta_usuario_password, t2.cuenta_usuario_inte
176 if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
177 {
178     foreach (DataRow x in mDs.Tables[0].Rows)
179     {
180         int pId = int.Parse(mDs.Tables[0].Rows[0]["cuenta_usuario_id"].ToString());
181         CuentaUsuario mCuentaUsuario = new CuentaUsuario(pId);
182         ValorizarEntidad(mCuentaUsuario, mDs.Tables[0].Rows[0]);
183         mCuentasUsuario.Add(mCuentaUsuario);
184     }
185     return mCuentasUsuario;
186 }
187 else return null;
188 }
```

BitacoraDAL.cs

BackupadorDAL.cs

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

10      5 referencias
11      public class BitacoraDAL
12      {
13          public static int mId;
14
15          1 referencia
16          public static int ProximoId()
17          {
18              if (mId == 0)
19              {
20                  DAO mDAOObject = new DAO();
21                  mId = mDAOObject.ObtenerId("bitacora");
22              }
23              mId += 1;
24              return mId;
25          }
26
27          1 referencia
28          public static int Guardar(Bitacora pBitacora)
29          {
30              DAO mDAOObject = new DAO();
31              string pCadenaComando;
32              pBitacora.bitacora_id = ProximoId();
33              pCadenaComando = "Insert into bitacora(bitacora_id, cuenta_usuario_id, bitacora_criticidad, bitacora_transaccion_id, bitacora_fecha, bitacora_hora,
34              return mDAOObject.ExecuteNonQuery(pCadenaComando);
35
36          }
37
38          3 referencias
39          public static void ValorizarEntidad(Bitacora pBitacora, DataRow pDr)
40          {
41              pBitacora.bitacora_id = int.Parse(pDr["bitacora_id"].ToString());
42              pBitacora.bitacora_criticidad = int.Parse(pDr["bitacora_criticidad"].ToString());
43              pBitacora.bitacora_hora = TimeSpan.Parse(pDr["bitacora_hora"].ToString());
44              pBitacora.bitacora_fecha = DateTime.Parse(pDr["bitacora_fecha"].ToString());
45              pBitacora.bitacora_transaccion = pDr["bitacora_transaccion_desc"].ToString();
46          }
47
48      }

```

```

42      public static List<Bitacora> Listar()
43      {
44          DAO mDAOObject = new DAO();
45          DataSet mDs = new DataSet();
46          List<Bitacora> mRegistros = new List<Bitacora>();
47          mDs = mDAOObject.ExecuteDataSet("select B.bitacora_id, bitacora_criticidad, BTM.bitacora_transaccion_desc, B.bitacora_fecha, B.bitacora_hora from bi
48
49          if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
50          {
51              foreach (DataRow mDr in mDs.Tables[0].Rows)
52              {
53                  Bitacora mBitacora = new Bitacora(int.Parse(mDr["bitacora_id"].ToString()));
54                  ValorizarEntidad(mBitacora, mDr);
55                  mRegistros.Add(mBitacora);
56              }
57          }
58          return mRegistros;
59      }
60
61      1 referencia
62      public static Bitacora Obtener(int pId)
63      {
64          DAO mDAOObject = new DAO();
65          DataSet mDs = mDAOObject.ExecuteDataSet("select B.bitacora_id, bitacora_criticidad, BTM.bitacora_transaccion_desc, B.bitacora_fecha, B.bitacora_hora
66          if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
67          {
68              Bitacora mBitacora = new Bitacora(pId);
69              ValorizarEntidad(mBitacora, mDs.Tables[0].Rows[0]);
70              return mBitacora;
71          }
72          else return null;
73      }
74
75      1 referencia
76      public static List<Bitacora> ListarRango(DateTime pFechaDesde, DateTime pFechaHasta)
77      {

```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```

76
77 DAO mDAOObject = new DAO();
78 DataSet mDs = new DataSet();
79 List<Bitacora> mRegistros = new List<Bitacora>();
80 mDs = mDAOObject.ExecuteDataSet("select B.bitacora_id, bitacora_criticidad, BTM.bitacora_transaccion_desc, B.bitacora_fecha, B.bitacora_hora from bi
81
82 if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
83 {
84     foreach (DataRow mDr in mDs.Tables[0].Rows)
85     {
86         Bitacora mBitacora = new Bitacora(int.Parse(mDr["bitacora_id"].ToString()));
87         ValorizarEntidad(mBitacora, mDr);
88         mRegistros.Add(mBitacora);
89     }
90 }
91 return mRegistros;
92
93 1 referencia
94 public static List<string> TiposMovimiento()
95 {
96     DAO mDAOObject = new DAO();
97     DataSet mDs = new DataSet();
98     List<String> mTipos = new List<string>();
99     mDs = mDAOObject.ExecuteDataSet("select bitacora_transaccion_desc from bitacora_tipo_movimiento");
100 if (mDs.Tables.Count > 0 && mDs.Tables[0].Rows.Count > 0)
101 {
102     foreach (DataRow mDr in mDs.Tables[0].Rows)
103     {
104         string mTipo = mDr["bitacora_transaccion_desc"].ToString();
105         mTipos.Add(mTipo);
106     }
107 }
108 return mTipos;
109 }
110 }
```

BackupadorDAL.cs

```

8
9
10 public class BackupadorDAL
11 {
12     1 referencia
13     public static int RealizarBackup(string pNombreArchivo, string pRuta, int pVolumenes)
14     {
15         DAO mDAOObject = new DAO();
16         string mBase = mDAOObject.mCon.Database;
17         string cadena = "BACKUP DATABASE " + mBase + " TO ";
18         for (int i = 1; i <= pVolumenes; i++)
19         {
20             if (i < pVolumenes)
21             {
22                 cadena += " DISK = '" + pRuta + "/" + pNombreArchivo + i + ".bak'";
23             }
24             else
25             {
26                 cadena += " DISK = '" + pRuta + "/" + pNombreArchivo + i + ".bak'";
27             }
28         }
29     }
30
31     1 referencia
32     public static int RealizarRestore(string pNombreArchivo, string pRuta, int pVolumenes)
33     {
34         DAO mDAOObject = new DAO();
35         string mBase = mDAOObject.mCon.Database;
36         string cadena = "ALTER DATABASE " + mBase + " SET SINGLE_USER WITH ROLLBACK IMMEDIATE;RESTORE DATABASE " + mBase + " FROM ";
37         for (int i = 1; i <= pVolumenes; i++)
38         {
39             if (i < pVolumenes)
40             {
41                 cadena += " DISK = '" + pRuta + "/" + pNombreArchivo + i + ".bak'";
42             }
43         }
44     }
45 }
```

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

```
41     {
42         cadena += " DISK = '" + pRuta + "/" + pNombreArchivo + i + ".bak'";
43     }
44 }
45 cadena += " WITH REPLACE";
46 return mDAOObject.ExecuteNonQuery(cadena, "master");
47
48
49 }
```



UNIVERSIDAD ABIERTA INTERAMERICANA

Facultad de tecnología informática

Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

Mapa de navegación

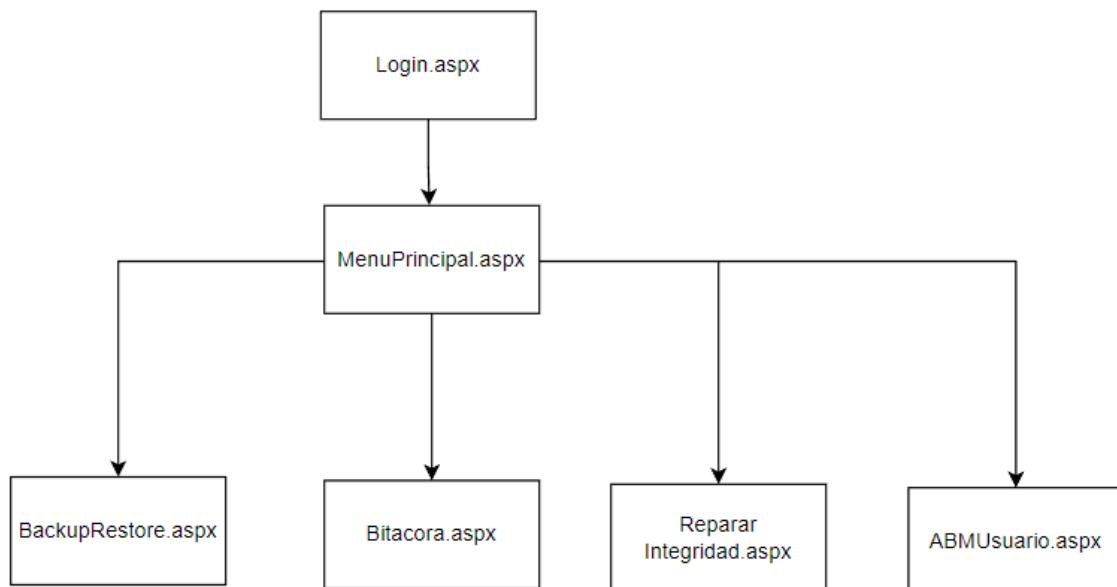
Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

Mapas de navegación por perfil

Perfil Webmaster

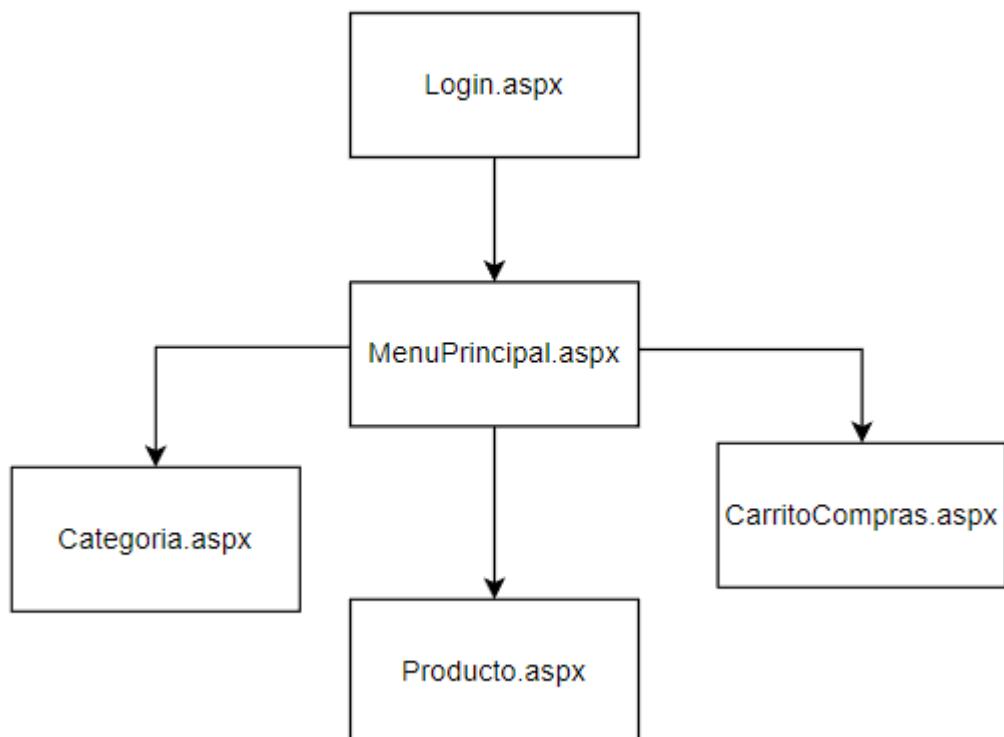


Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

Perfil Usuario

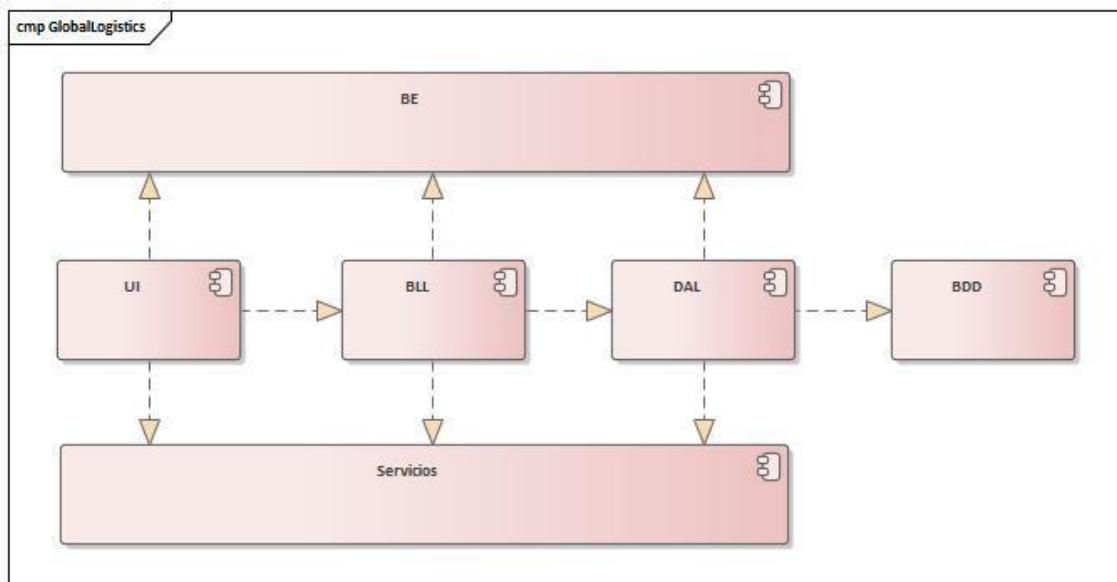


Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

Diagrama de Componentes



Materia: Modelos computacionales de
Gestión Administrativa

Docente: Milio, Claudio

Alumnos: Crivella, Toffolon, Dacunda, Arango

Bibliografía

- ASP.NET - Microsoft Documentation
<https://dotnet.microsoft.com/apps/aspnet>
- MD5 Class
<https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.md5?view=net-5.0>