**UNIVERSIDAD ABIERTA INTERAMERICANA**

**Facultad de Tecnología Informática**



Lenguaje de Programación para la Administración

Versión 1.0

|  |  |  |  |
| --- | --- | --- | --- |
| Profesor:  **Ing.Santiago Sabato** | | Alumnos:  **Sebastian Travella**  **Mariano Kaimakamian** | |
|  | | | |
| Sede: **UAI – Centro** | **Turno Noche** | |  |
| **4to. Año** | **Comisión B** | | **2017** |

**DUSHANA**

**(CENTRO SPA)**

# Índice

[Índice 3](#_Toc468395891)

[Historial de Revisiones 6](#_Toc468395892)

[1. Introducción 7](#_Toc468395893)

[2. Estructura de las Tablas 8](#_Toc468395894)

[3. Mapa de Navegación 9](#_Toc468395895)

[3.1 Mapa de Navegación WebMaster 10](#_Toc468395896)

[3.2 Mapa de Navegación Cliente 11](#_Toc468395897)

[4. Estrategias de Programación 11](#_Toc468395898)

[4.1 Modelado de Capas 11](#_Toc468395899)

[5. Capa de Presentación 12](#_Toc468395900)

[6. Capa de Negocio 12](#_Toc468395901)

[7. Capa de Acceso a Datos 12](#_Toc468395902)

[8. Diagrama de Flujo 13](#_Toc468395903)

[8.1 Diagrama de Secuencia 14](#_Toc468395904)

[8.2 Diagrama de Comunicaciòn 15](#_Toc468395905)

[8.3 Explorador de Soluciones 16](#_Toc468395906)

[9. Seguridad 17](#_Toc468395907)

[9.1 Login – Logout 17](#_Toc468395908)

[9.2 Usuario, Roles, Permisos 17](#_Toc468395909)

[9.3 Backup y Restore 17](#_Toc468395910)

[9.4 Bitácora 17](#_Toc468395911)

[9.5 Dígitos Verificadores 17](#_Toc468395912)

[9.6 Encriptación 18](#_Toc468395913)

[10. Capa de Presentación (UI) 18](#_Toc468395914)

[10.1 MasterPage.master 18](#_Toc468395915)

[10.2 AboutUs.aspx 21](#_Toc468395916)

[10.3 ClientRegistration.aspx 21](#_Toc468395917)

[10.4 LogIn.aspx 22](#_Toc468395918)

[10.5 Client.aspx 23](#_Toc468395919)

[10.6 Product.aspx 24](#_Toc468395920)

[10.7 ProductInformation.aspx 25](#_Toc468395921)

[10.8 ProductList.aspx 25](#_Toc468395922)

[10.9 Backup.aspx 26](#_Toc468395923)

[10.10 Restore.aspx 26](#_Toc468395924)

[10.11 ViewerLog.aspx 27](#_Toc468395925)

[10.12 User.aspx 28](#_Toc468395926)

[10.13 UserPermission.aspx 29](#_Toc468395927)

[11. Capa de Negocio (BLL) 30](#_Toc468395928)

[11.1 LogInBLL 30](#_Toc468395929)

[11.2 LogBLL 32](#_Toc468395930)

[11.3 CustomerBLL 33](#_Toc468395931)

[11.4 CheckDataBaseBLL 33](#_Toc468395932)

[11.5 BackupRestoreBLL 34](#_Toc468395933)

[11.6 ProductBLL 35](#_Toc468395934)

[12. Capa de DATOS (DAL) 36](#_Toc468395935)

[12.1 Código BD Script de Creación 36](#_Toc468395936)

[12.2 Conexión con la Base de Datos 38](#_Toc468395937)

[12.3 LogInDAL 39](#_Toc468395938)

[12.4 LogDAL 41](#_Toc468395939)

[12.5 CustomerDAL 42](#_Toc468395940)

[12.6 CheckDBDAL 43](#_Toc468395941)

[12.7 BackupRestoreDAL 44](#_Toc468395942)

[12.8 ProductDAL 44](#_Toc468395943)

[13. Capa de Seguridad (Helper ) 45](#_Toc468395944)

[13.1 SecurityHelper 45](#_Toc468395945)

[14. Capa de Entidades (DTO) 46](#_Toc468395946)

[14.1 LogInDTO 46](#_Toc468395947)

[14.2 UserDTO 46](#_Toc468395948)

[14.3 LogDTO 47](#_Toc468395949)

[14.4 ProductDTO 49](#_Toc468395950)

[14.5 ResultDTO 50](#_Toc468395951)

[14.6 CustomerDTO 51](#_Toc468395952)

[14.7 DvhDTO 53](#_Toc468395953)

[14.8 DvvDTO 53](#_Toc468395954)

[15. Archivos de configuración 54](#_Toc468395955)

[15.1 Web.config 54](#_Toc468395956)

# Historial de Revisiones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fecha | Versión | Descripción | Revisor | Autor |
| 24.11.2016 | 0.1 | Creación del Documento | Santiago Sabato | Dushana |
| 29/11/2016 | 1.0 | Agregado de WebService, XML, Capas y flujo de datos | Santiago Sabato | Dushana |
|  |  |  |  |  |

# Introducción

**Dushana** es un centro de SPA, a través del sitio se realizaran las ventas de los servicios y difundir sus paquetes.

El usuario podrá realizar la auto-gestión, ya que el centro podrá paralelizar las ventas on site y vía internet de los consumidores, de esta manera reducir los tiempos de atención, ya que no será necesario que los clientes acerquen hasta el local para solicitar turno.

Ofreceremos dos servicios:

* Un paquete cerrado de tratamientos
* ‘A Gusto del Comprador’

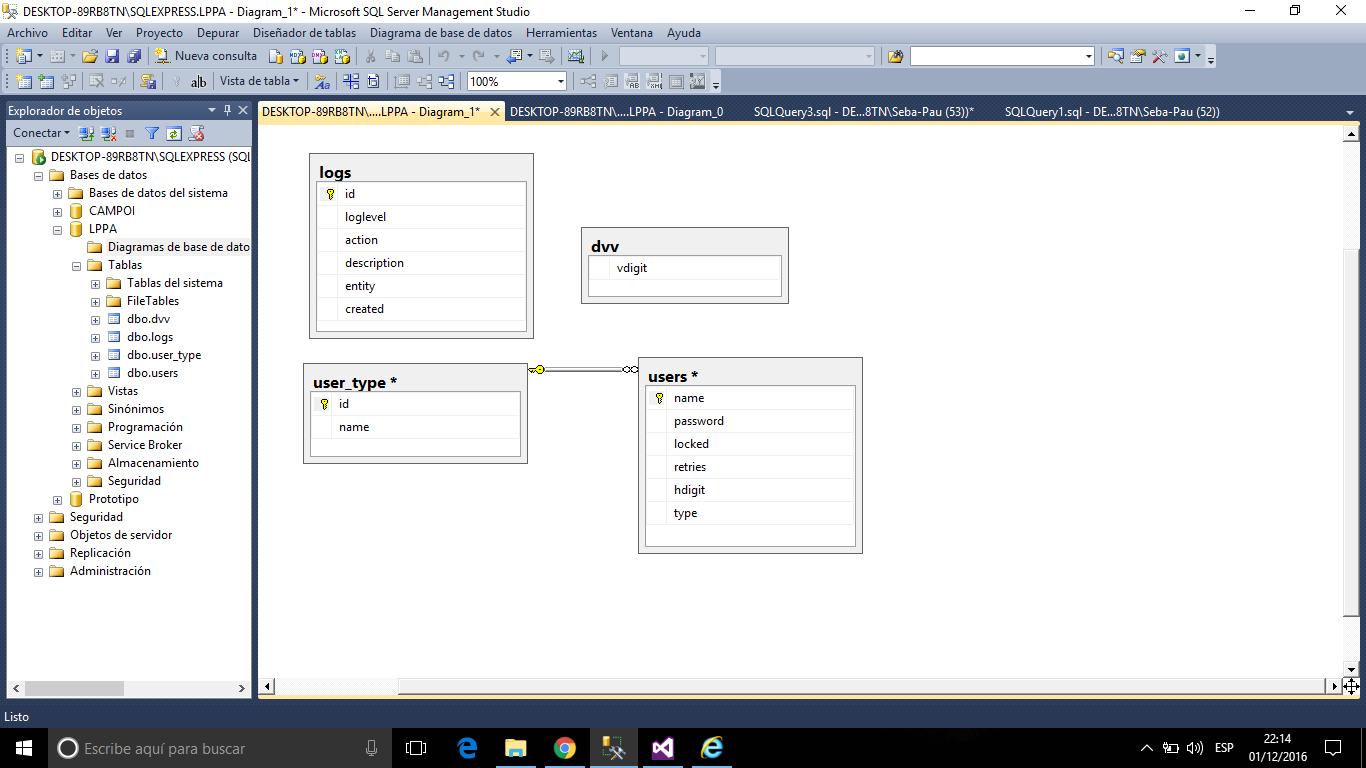
Ambos tipo de servicios requieren distinto manejo de las agendas del personal del centro de SPA.

Tendrá la ventaja de que la informatización de las ventas facilitará la localización de las vacantes según el tipo de servicio que el cliente desee, sin depender de la pericia del vendedor a la hora de encontrar la disponibilidad que más se adecue a las demandas del comprador.

El desarrollo de un sitio web permitirá:

* Publicitar las actividades de los profesionales empleados.
* Gestionar la contratación de servicios por parte de clientes.
* Administrar contenido por parte de los empleados.
* Realizar estadísticas de ingreso a la aplicación.
* Realizar encuestas de satisfacción a los consumidores.

# Estructura de las Tablas



# Mapa de Navegación

El diagrama Administración del Sistema en detalle:

## Mapa de Navegación WebMaster

## Mapa de Navegación Cliente

# Estrategias de Programación

Debido a que nuestro proyecto es a largo plazo, definiremos la estrategia de programación que utilizaremos en nuestra aplicación web.

El planteo que desarrollaremos es a través de la estructura por capas ya que consideramos que es el óptimo y el uso de las mismas simplifica el mantenimiento al momento de realizar una modificación a la aplicación.

## Modelado de Capas



# Capa de Presentación

Esta capa es la encargada de manejar las estructuras de datos abstractas y realizar las conversiones de representación de datos necesarias para la correcta interpretación de los mismos.

El usuario interactuará con la capa de presentación, y la misma le devolverá los datos procesados por el sistema.

# Capa de Negocio

En la capa de Negocio es donde se encuentra la programación referente a las validaciones y seguridad de nuestro sitio web.

Las capas que interactuaran con la capa de Negocio son: la capa de Presentación y la Capa de Acceso a Datos.

# Capa de Acceso a Datos

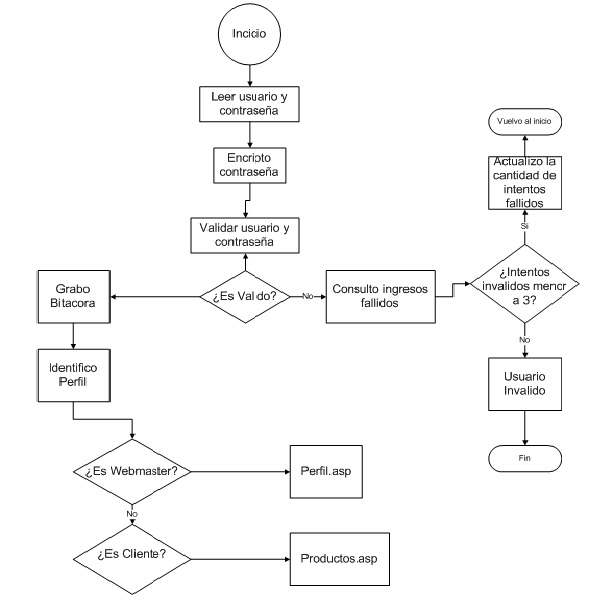
Esta capa contendrá las clases cuya responsabilidad será la de persistir el estado de la capa de negocio en la base de datos.

Consta de una clase la cual tiene la responsabilidad de conservar todos los datos de acceso a la base de datos. Entre ellos podemos mencionar los siguientes:

* Conexión/ Desconexión (Tipo de Driver, Servidor Referenciado, Nombre de Base de datos, usuario, y contraseña);
* Consultas (recuperación de registros, como ejecución de comandos);
* Inserción de registros en tablas de auditoria.

Se gestionarán las conexiones al servidor, se mantendrán disponibles para todas las clases que la extiendan, y se cerraran cuando no sea necesario disponer de ellas.

# Diagrama de Flujo



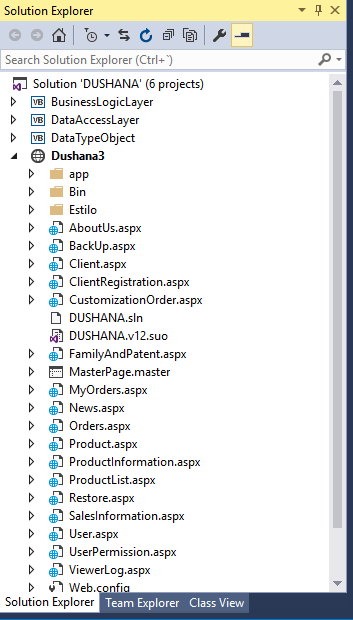
## Diagrama de Secuencia



## Diagrama de Comunicaciòn



## Explorador de Soluciones



# Seguridad

## Login – Logout

El control del Login se ejecutará cada vez que un usuario ingrese al sistema (después de verificar la consistencia de los datos) y en la salida se realizará el Logout.

Se utilizarán las tablas de la bitácora para registrar la entrada y salida del usuario al sistema documentando en los campos de auditoría de la tabla la fecha y hora de cada acceso y salida.

La única forma de acceder al sistema será ingresando un usuario registrado y habilitado con su correspondiente contraseña.

Si un usuario ingreso incorrectamente más de 3 veces, no podrá acceder al sistema.

## Usuario, Roles, Permisos

El sistema contará con un módulo de seguridad que controlará los accesos a los diferentes módulos. Para ello se utilizará un sistema de roles y permisos.

Cada función que realice el sistema estará identificada con una permiso, siendo un rol un conjunto de permisos relacionadas.

Una vez definidas las roles y permisos, se le asignarán a cada usuario en la medida en que les corresponda tener acceso a un módulo del sistema (permiso) o a un grupo de módulos (roles).

## Backup y Restore

Ofrece un servicio de backups para asegurar la recuperación del sistema.

Los backups pueden son manuales, guardando los datos en el disco duro de la propia máquina. Tras ello se recomienda copiarlos en algún soporte físico externo, ya que si la máquina sufriera un fallo grave podríamos perder también el backup de la configuración.

Para acceder a las opciones de la copia de seguridad de configuración iremos a Administración /Backup.

## Bitácora

El uso de la bitácora en el sistema permitirá registrar los movimientos del usuario desde que ingresa al sistema hasta que finaliza su sesión.

## Dígitos Verificadores

Los dígitos verificadores son utilizados para garantizar que la información sensible que reside en la base de datos no sea alterada de forma intencional o por un error del sistema. Debido a esto se establece un atributo llamado Dígito Verificador Horizontal para la suma de los campos de la tabla de forma horizontal y una entidad que contiene el nombre de cada tabla más el campo Dígito Verificador Vertical para hacer la verificación de los dígitos verticales.

El procedimiento de verificación de los dígitos se realizará cuando el usuario intente ingresar al sistema de manera tal que ante un error el mismo no podrá ingresar al sistema y deberá contactar al administrador para que realice un Restore. Debido a esto es recomendable realizar backups diarios del sistema.

## Encriptación

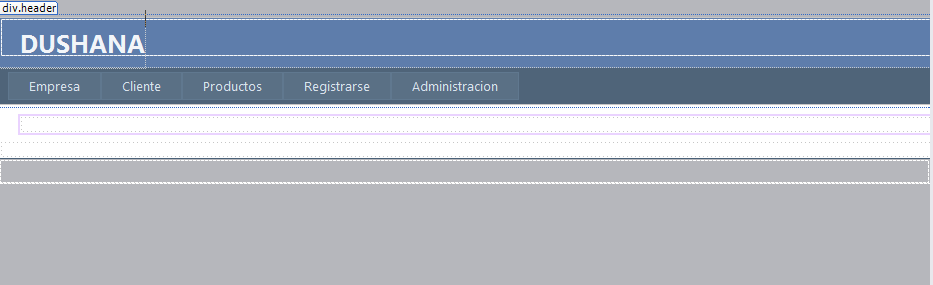
Para asegurar la confidencialidad de la información, se va a encriptar campos sensibles como la contraseña de un usuario.

El uso de la bitácora en el sistema permitirá registrar los movimientos del usuario desde que ingresa al sistema hasta que finaliza su sesión.

# Capa de Presentación (UI)

En este apartado mostraremos la interface del usuario al visualizar cada una de las páginas.

### MasterPage.master



**Código MasterPage.master**

<%@ Master Language="VB" CodeFile="MasterPage.master.vb" Inherits="MasterPage" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<title></title>

<link href="Estilo/MyStyle.css" rel="stylesheet" />

<asp:ContentPlaceHolder id="head" runat="server">

</asp:ContentPlaceHolder>

</head>

<body>

<form runat="server">

<div class="page">

<div class="header">

<div class="title">

<h1>

Dushana

</h1>

</div>

<div class="loginDisplay">

</div>

<div class="clear hideSkiplink">

<asp:Menu ID="NavigationMenu" runat="server" CssClass="menu" EnableViewState="false"

IncludeStyleBlock="false" Orientation="Horizontal">

<Items>

<asp:MenuItem NavigateUrl="~/AboutUs.aspx" Text="Empresa" />

<asp:MenuItem Text="Cliente" >

<asp:MenuItem NavigateUrl="~/Orders.aspx" Text="Carrito" />

<asp:MenuItem NavigateUrl="~/MyOrders.aspx" Text="Mis Compras" />

</asp:MenuItem>

<asp:MenuItem Text="Productos">

<asp:MenuItem NavigateUrl="~/ProductList.aspx" Text="Lista de Productos" />

<asp:MenuItem NavigateUrl="~/CustomizationOrder.aspx" Text="Customizaciones" />

</asp:MenuItem>

<asp:MenuItem NavigateUrl="~/ClientRegistration.aspx" Text="Registrarse" />

<asp:MenuItem Text="Administracion">

<asp:MenuItem Text="Administracion del Sistema">

<asp:MenuItem Text="Permisos" >

<asp:MenuItem NavigateUrl="~/FamilyAndPatent.aspx" Text="Registro y Asignacion de Patentes y Familias" />

<asp:MenuItem NavigateUrl="~/Client.aspx" Text="Alta de Clientes" />

<asp:MenuItem NavigateUrl="~/User.aspx" Text="Alta de Usuarios" />

<asp:MenuItem NavigateUrl="~/UserPermission.aspx" Text="Permisos de Usuarios" />

</asp:MenuItem>

<asp:MenuItem Text="Seguridad de Datos" >

<asp:MenuItem NavigateUrl="~/Backup.aspx" Text="Copia de Seguridad" />

<asp:MenuItem NavigateUrl="~/Restore.aspx" Text="Restauracion" />

</asp:MenuItem>

<asp:MenuItem Text="Actividades realizadas del sistema" >

<asp:MenuItem NavigateUrl="ViewerLog.aspx" Text="Bitacora" />

</asp:MenuItem>

</asp:MenuItem>

<asp:MenuItem Text="Administracion de Ventas">

<asp:MenuItem NavigateUrl="~/SalesInformation.aspx" Text="Informacion de Venta" />

<asp:MenuItem NavigateUrl="~/ProductInformation.aspx" Text="Informacion de Productos" />

<asp:MenuItem NavigateUrl="~/Product.aspx" Text="Alta de Clientes" />

</asp:MenuItem>

<asp:MenuItem Text ="Adminsitracion de Informacion">

<asp:MenuItem NavigateUrl="~/News.aspx" Text="Noticias" />

</asp:MenuItem>

</asp:MenuItem>

</Items>

</asp:Menu>

</div>

</div>

<div class="main">

<asp:ContentPlaceHolder ID="MainContent" runat="server" />

</div>

<div class="clear">

</div>

</div>

<div class="footer">

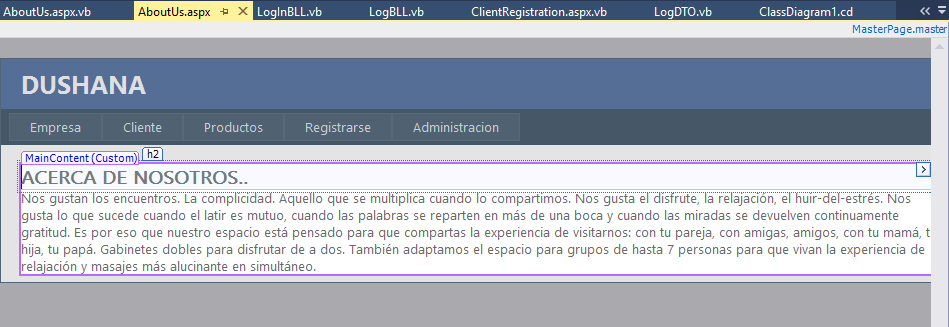
</div>

</form>

</body>

</html>

### AboutUs.aspx



**Código AboutUs.aspx**

<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master" AutoEventWireup="false" CodeFile="AboutUs.aspx.vb" Inherits="AboutUs" %>

<%-- Commit de prueba --%>

<%--Segundo Commit de prueba--%>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">

</asp:Content>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" Runat="Server">

<h2>Acerca de Nosotros..</h2>

Nos gustan los encuentros. La complicidad. Aquello que se multiplica cuando lo compartimos. Nos gusta el disfrute, la relajación, el huir-del-estrés.

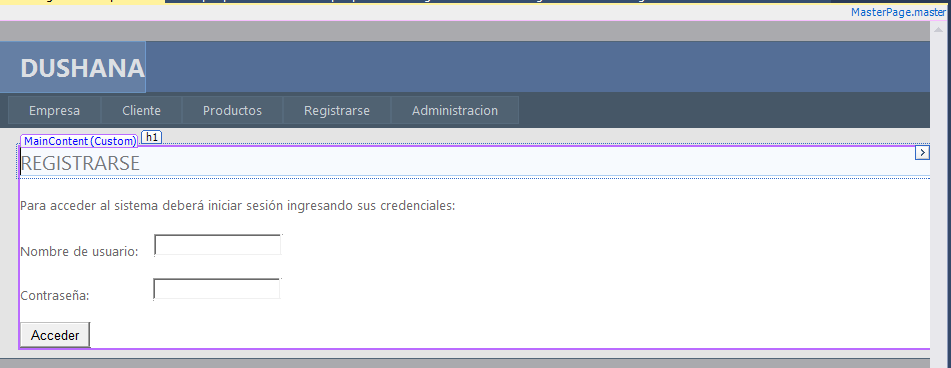
Nos gusta lo que sucede cuando el latir es mutuo, cuando las palabras se reparten en más de una boca y cuando las miradas se devuelven continuamente gratitud.

Es por eso que nuestro espacio está pensado para que compartas la experiencia de visitarnos: con tu pareja, con amigas, amigos, con tu mamá, tu hija, tu papá.

Gabinetes dobles para disfrutar de a dos. También adaptamos el espacio para grupos de hasta 7 personas para que vivan la experiencia de relajación y masajes más alucinante en simultáneo.

</asp:Content>

### ClientRegistration.aspx



**Código ClientRegistration.aspx**

<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master" AutoEventWireup="false" CodeFile="ClientRegistration.aspx.vb" Inherits="ClientRegistration" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">

</asp:Content>

<asp:Content ID="MainBody" ContentPlaceHolderID="MainContent" Runat="Server">

<h1>Registrarse</h1>

<p>Para acceder al sistema deberá iniciar sesión ingresando sus credenciales:</p>

<p>Nombre de usuario:&nbsp;&nbsp;&nbsp;

<asp:TextBox ID="TxtUser" runat="server"></asp:TextBox>

</p>

<p>Contraseña:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:TextBox ID="TxtPassword" runat="server"></asp:TextBox>

</p>

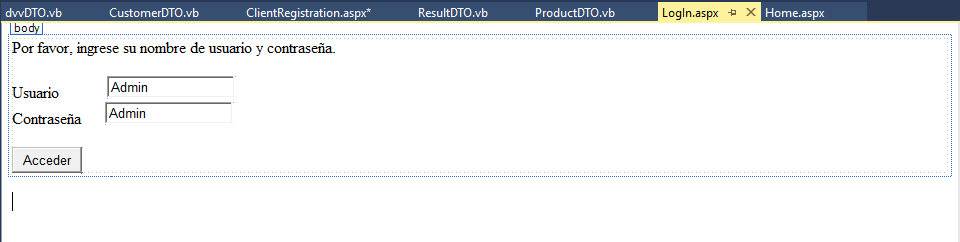
<p>

<asp:Button ID="BtnAccess" runat="server" Text="Acceder" />

</p>

</asp:Content>

### LogIn.aspx



**Código LogIn.aspx**

Partial Class LogIn

Inherits System.Web.UI.Page

Protected Sub BtnAccess\_Click(sender As Object, e As EventArgs) Handles BtnAccess.Click

Dim CheckDataBaseBLL As New CheckDataBaseBLL

Dim dvResultDTO As ResultDTO

dvResultDTO = CheckDataBaseBLL.CheckDVV

If dvResultDTO.IsCurrentError(ResultDTO.type.CORRUPTED\_DATABASE) Then

Response.Redirect("Restore.aspx")

Else

Dim LogInBLL As New LogInBLL()

Dim result As ResultDTO

result = LogInBLL.LogIn(TxtUser.Text, TxtPassword.Text)

If result.IsValid() Then

Session("user") = result.value

Response.Redirect("Home.aspx")

Else

MsgBox(result.description, vbOKOnly, "Error")

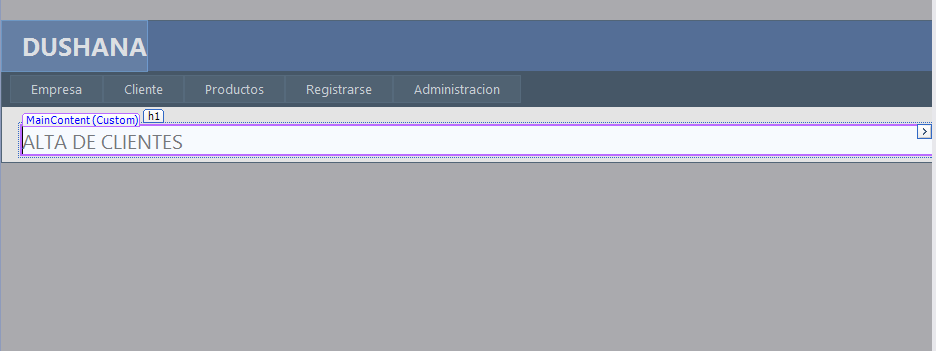
End If

End If

End Sub

End Class

### Client.aspx



**Código Client.aspx**

<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master" AutoEventWireup="false" CodeFile="Client.aspx.vb" Inherits="Client" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">

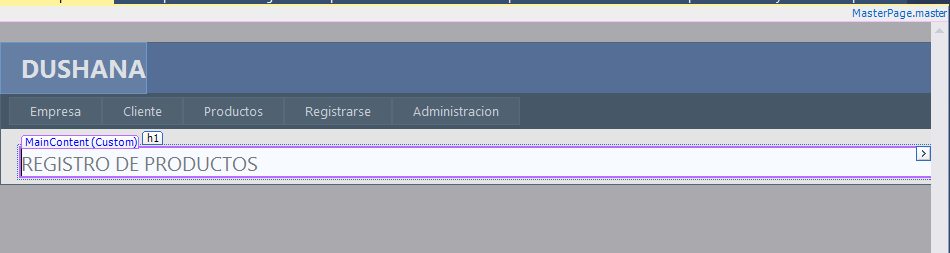
</asp:Content>

<asp:Content ID="MainBody" ContentPlaceHolderID="MainContent" Runat="Server">

<h1>Alta de Clientes</h1>

</asp:Content>

### Product.aspx



**Código Product.aspx**

<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master" AutoEventWireup="false" CodeFile="Product.aspx.vb" Inherits="Product" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">

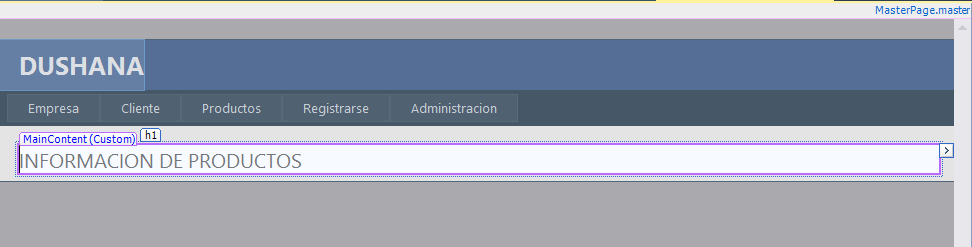
</asp:Content>

<asp:Content ID="MainBody" ContentPlaceHolderID="MainContent" Runat="Server">

<h1>Registro de Productos</h1>

</asp:Content>

### ProductInformation.aspx



<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master" AutoEventWireup="false" CodeFile="ProductInformation.aspx.vb" Inherits="ProductInformation" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">

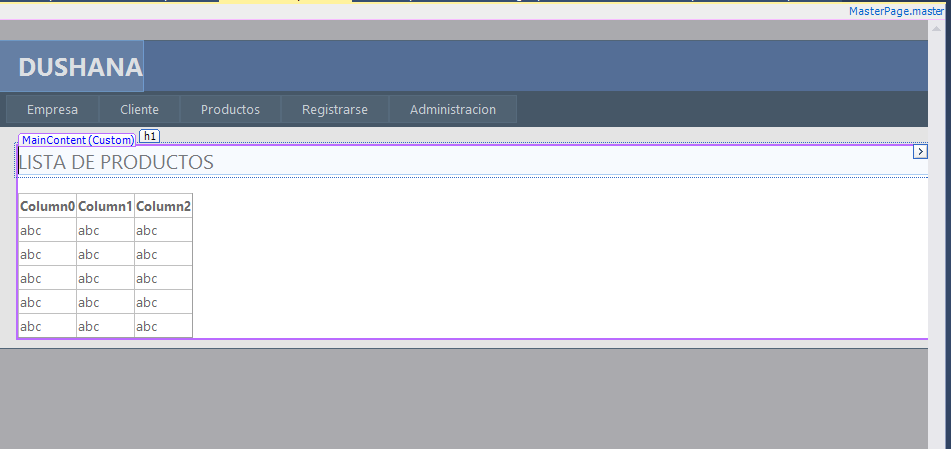
</asp:Content>

<asp:Content ID="MainBody" ContentPlaceHolderID="MainContent" Runat="Server">

<h1>Informacion de Productos</h1>

</asp:Content>

### ProductList.aspx



**Código ProductList.aspx**

<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master" AutoEventWireup="false" CodeFile="ProductList.aspx.vb" Inherits="ProductList" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">

</asp:Content>

<asp:Content ID="MainBody" ContentPlaceHolderID="MainContent" Runat="Server">

<h1>Lista de Productos</h1>

<p>

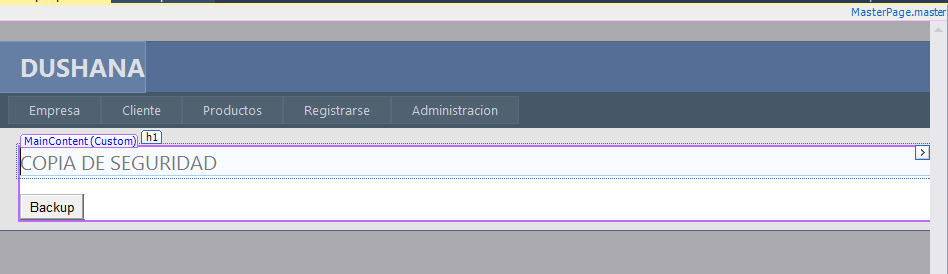
<asp:GridView ID="ProductGridView" runat="server">

</asp:GridView>

</p>

</asp:Content>

### Backup.aspx



**Código Backup.aspx**

Partial Class BackUp

Inherits System.Web.UI.Page

Protected Sub btnBackup\_Click(sender As Object, e As EventArgs) Handles btnBackup.Click

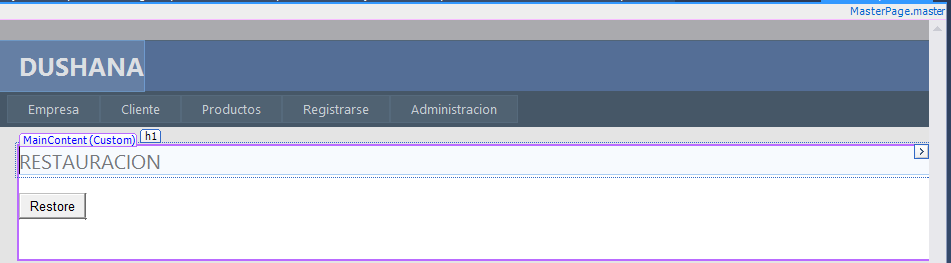
Dim BackUpRestoreBLL As New BackupRestoreBLL

BackUpRestoreBLL.doBackup()

End Sub

End Class

### Restore.aspx



**Código Restore.aspx**

Partial Class Restore

Inherits System.Web.UI.Page

Protected Sub btnRestore\_Click(sender As Object, e As EventArgs) Handles btnRestore.Click

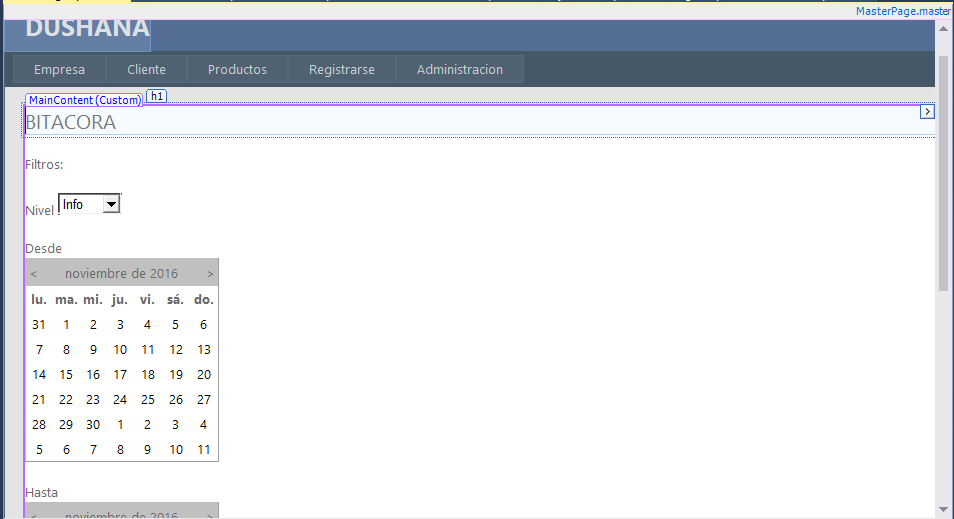
Dim BackUpRestoreBLL As New BackupRestoreBLL

BackUpRestoreBLL.doRestore()

End Sub

End Class

### ViewerLog.aspx





**Código ViewerLog.aspx**

Partial Class ViewerLog

Inherits System.Web.UI.Page

Private Sub ViewerLog\_LoadComplete(sender As Object, e As EventArgs) Handles Me.LoadComplete

If IsNothing(Session("LogCriteria")) Then

RetrieveLogs(LogDTO.level.INFO)

Else

Dim criteria As Dictionary(Of String, String) = Session("LogCriteria")

RetrieveLogs(criteria("loglevel"), criteria("since"), criteria("until"))

End If

End Sub

Protected Sub BtnAccept\_Click(sender As Object, e As EventArgs) Handles BtnAccept.Click

' Recarga la página con los criterios seleccionados

Dim criteria As New Dictionary(Of String, String)

criteria.Add("loglevel", CmbLevel.SelectedValue)

criteria.Add("since", SinceCalendar.SelectedDate.ToString())

criteria.Add("until", UntilCalendar.SelectedDate.ToString())

Session("LogCriteria") = criteria

Response.Redirect("ViewerLog.aspx")

End Sub

Private Sub RetrieveLogs(ByRef level As String, ByRef Optional since As String = Nothing, ByRef Optional until As String = Nothing)

If IsNothing(Session("user")) Then

' Si no hay usuario en sessión, redirige al login

Response.Redirect("LogIn.aspx")

Else

Dim logBll As New LogBLL()

Dim result As ResultDTO

result = logBll.GetLog(level, since, until)

LogGridView.DataSource = result.value

LogGridView.DataBind()

End If

End Sub

Private Sub ViewerLog\_Load(sender As Object, e As EventArgs) Handles Me.Load

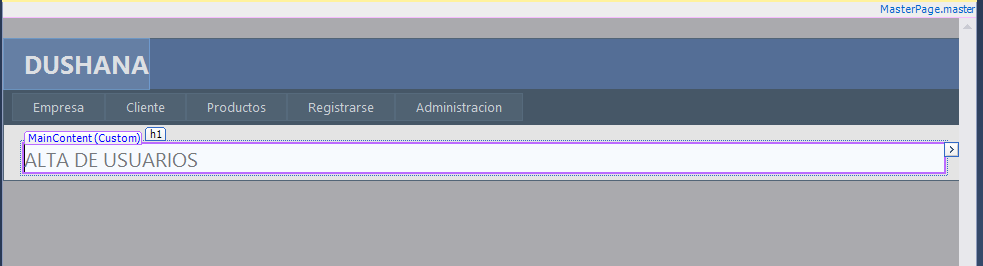
SinceCalendar.SelectedDate = Now.AddDays(-1)

UntilCalendar.SelectedDate = Now.AddDays(1)

End Sub

End Class

### User.aspx



**Código User.aspx**

<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master" AutoEventWireup="false" CodeFile="User.aspx.vb" Inherits="User" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">

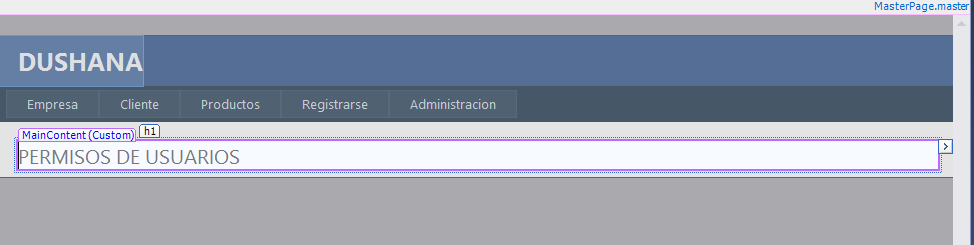
</asp:Content>

<asp:Content ID="MainBody" ContentPlaceHolderID="MainContent" Runat="Server">

<h1>Alta de Usuarios</h1>

</asp:Content>

### UserPermission.aspx



**Código UserPermission.aspx**

<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master" AutoEventWireup="false" CodeFile="UserPermission.aspx.vb" Inherits="UserPermission" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">

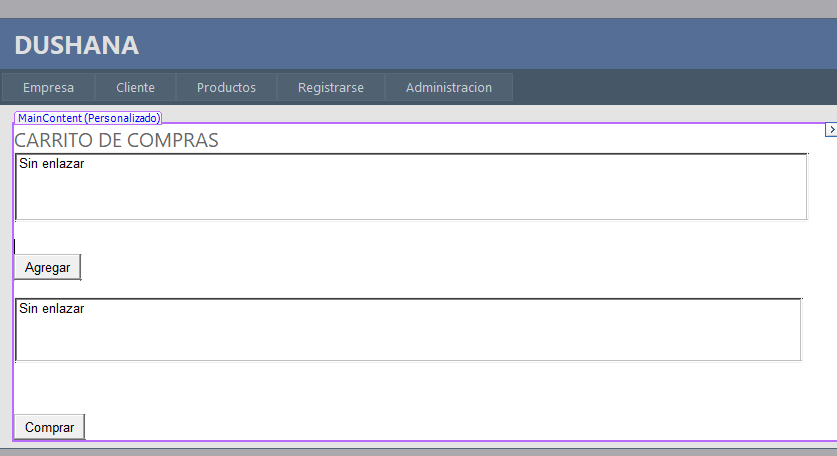
</asp:Content>

<asp:Content ID="MainBody" ContentPlaceHolderID="MainContent" Runat="Server">

<h1>Permisos de Usuarios</h1>

</asp:Content>

### Orders



**Código Orders.aspx**

Imports BusinessLogicLayer

Imports DataTypeObject

Imports System.Xml

Partial Class Orders

Inherits System.Web.UI.Page

Dim productList As New List(Of ProductDTO)

Dim Carrito As New List(Of ProductDTO)

Public Sub getProductList()

Dim productBll As New ProductBLL()

Dim result As ResultDTO

result = productBll.GetProducts()

For Each Product As ProductDTO In result.value

productList.Add(Product)

Next

End Sub

Private Sub Orders\_Load(sender As Object, e As EventArgs) Handles Me.Load

If Not IsPostBack Then

getProductList()

ListBox1.DataTextField = "name"

ListBox1.DataSource = productList

ListBox1.DataBind()

End If

End Sub

Protected Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

getProductList()

For Each producto As ProductDTO In productList

If producto.name = ListBox1.SelectedItem.Value Then

Carrito.Add(producto)

ListBox2.Items.Add(producto.name)

End If

Next

ListBox2.DataTextField = "name"

End Sub

Sub AppendXML(Billing As BillingDTO)

Dim Reader As New XmlTextReader(Server.MapPath("ventas.xml"))

Dim doc As New XmlDocument

doc.Load(Reader)

Dim VentaNode As XmlNode = doc.CreateNode(XmlNodeType.Element, "Venta", "")

Dim fecha As XmlElement = doc.CreateElement("fecha")

Dim fechaTexto As XmlText = doc.CreateTextNode(Date.Now)

fecha.AppendChild(fechaTexto)

Dim cliente As XmlElement = doc.CreateElement("clienteNombre")

Dim clienteTexto As XmlText = doc.CreateTextNode(Session("User").name)

cliente.AppendChild(clienteTexto)

Dim ServiciosNode As XmlNode = doc.CreateNode(XmlNodeType.Element, "Servicios", "")

For Each P As ProductDTO In Billing.products

Dim ProductoNode As XmlNode = doc.CreateNode(XmlNodeType.Element, "Producto", "")

Dim id As XmlElement = doc.CreateElement("id")

Dim idTexto As XmlText = doc.CreateTextNode(P.id)

id.AppendChild(idTexto)

Dim nombre As XmlElement = doc.CreateElement("nombre")

Dim nombreTexto As XmlText = doc.CreateTextNode(P.name)

nombre.AppendChild(nombreTexto)

Dim descripcion As XmlElement = doc.CreateElement("descripcion")

Dim descripcionTexto As XmlText = doc.CreateTextNode(P.description)

descripcion.AppendChild(descripcionTexto)

Dim precio As XmlElement = doc.CreateElement("precio")

Dim precioTexto As XmlText = doc.CreateTextNode(P.prices)

precio.AppendChild(precioTexto)

doc.Item("ventas.xml").AppendChild(ProductoNode)

Next

doc.Item("ventas.xml").AppendChild(ServiciosNode)

Dim FacturacionNode As XmlNode = doc.CreateNode(XmlNodeType.Element, "Facturacion", "")

Dim totalLista As XmlElement = doc.CreateElement("totaLista")

Dim totalListaTexto As XmlText = doc.CreateTextNode(Billing.listTotal)

totalLista.AppendChild(totalListaTexto)

Dim Bonificacion As XmlElement = doc.CreateElement("bonificacion")

Dim BonificacionTexto As XmlText = doc.CreateTextNode(Billing.discount)

Bonificacion.AppendChild(BonificacionTexto)

Dim totalFinal As XmlElement = doc.CreateElement("totaFinal")

Dim totalFinalTexto As XmlText = doc.CreateTextNode(Billing.finalPrice)

totalFinal.AppendChild(totalFinalTexto)

doc.Item("ventas.xml").AppendChild(FacturacionNode)

doc.Item("ventas.xml").AppendChild(VentaNode)

End Sub

Protected Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

Dim Billing As New BillingDTO

Dim MiWebService As New WebService

getProductList()

Billing.products = productList

For Each Product As ProductDTO In Billing.products

Billing.listTotal = Billing.listTotal + Product.prices

Next

Billing.finalPrice = MiWebService.CalculateDiscount(Billing.listTotal)

If Billing.listTotal > 500 Then

Billing.discount = "10%"

Else

Billing.discount = "0%"

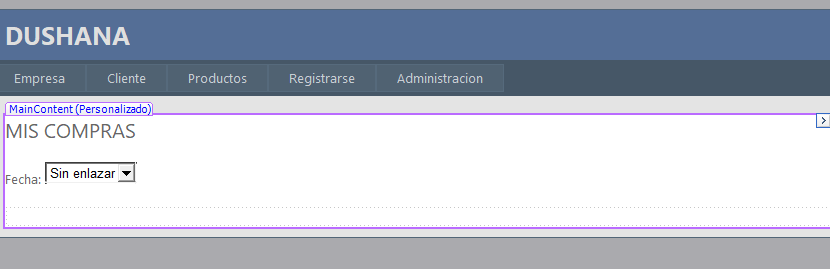
End If

AppendXML(Billing)

End Sub

End Class

### My Orders



**Código MyOrders.aspx**

Imports System.Xml

Imports System.Xml.XPath

Partial Class MyOrders

Inherits System.Web.UI.Page

Public sellTable As String

Protected Sub Page\_Load(sender As Object, e As EventArgs) Handles Me.Load

If Not IsPostBack Then

Dim docum As New XPathDocument(Server.MapPath("ventas.xml"))

Dim nav As XPathNavigator

Dim ite As XPathNodeIterator

nav = docum.CreateNavigator

Session.Add("Ventas", docum)

ite = nav.Select("dushana/venta/@fecha")

While ite.MoveNext

dropFechas.Items.Add(ite.Current.Value)

End While

PrintSells(dropFechas.SelectedValue)

End If

End Sub

Protected Sub dropFechas\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles dropFechas.SelectedIndexChanged

PrintSells(dropFechas.SelectedValue)

End Sub

Private Sub PrintSells(ByRef selectedDate As String)

Dim miDoc As XPathDocument = Session("Ventas")

Dim nav As XPathNavigator = miDoc.CreateNavigator()

Dim servicesNav As XPathNavigator = nav.SelectSingleNode("dushana/venta[@fecha='" + selectedDate + "']/cliente[@nombre='" + Session("user").name + "']")

sellTable += "Fecha seleccionada: " + selectedDate + "</p>"

sellTable += "<b>Productos</b> </p>"

If IsNothing(servicesNav) Then

sellTable = "No se han efectuado compras en el día de la fecha."

Return

End If

For Each productNav As XPathNavigator In servicesNav.Select("servicios/producto")

sellTable += "<b>Código&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:</b>&nbsp;" + productNav.SelectSingleNode("id").Value + "</br>"

sellTable += "<b>Tratamiento:</b>&nbsp;" + productNav.SelectSingleNode("nombre").Value + "</br>"

sellTable += "<b>Descripción&nbsp;:</b>&nbsp;" + productNav.SelectSingleNode("descripcion").Value + "</br>"

sellTable += "<b>Precio&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:</b>&nbsp;" + productNav.SelectSingleNode("precio").Value + "</br>"

sellTable += "</p></p>"

Next

sellTable += "</p></p>"

sellTable += "--------------------"

sellTable += "</p></p>"

sellTable += "Total parcial:&nbsp;" + servicesNav.SelectSingleNode("facturacion/totalLista").Value + "</br>"

sellTable += "Bonificación:&nbsp;&nbsp;" + servicesNav.SelectSingleNode("facturacion/bonificacion").Value + "</br>"

sellTable += "Total final:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;" + servicesNav.SelectSingleNode("facturacion/totalFinal").Value + "</br>"

End Sub

End Class

### Ventas.xml

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Modelo Objetos** | **Como está Relacionado** |
| Ventas.xml | XMLTextReader  XMLDocument  XMLText  XMLNode  XPathDocument  XPathNavigator | Orders.aspx  MyOrders.aspx |

Estructura:

<dushana>

<venta fecha="7/11/2017">

<cliente nombre="adminx">

<servicios>

<producto>

<id>5</id>

<nombre>Cavitación</nombre>

<descripcion> Tratamiento médico de estética que consiste en aplicar ultrasonidos en determinadas zonas del cuerpo para eliminar la grasa en ellas localizadas.</descripcion>

<precio>425</precio>

</producto>

</servicios>

<facturacion>

<totalLista>425</totalLista>

<bonificacion>0%</bonificacion>

<totalFinal>425</totalFinal>

</facturacion>

</cliente>

</venta>

</dushana>

# Capa de Negocio (BLL)

La capa de negocio es donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

### LogInBLL

Public Class LogInBLL

Const MAX\_RETRIES = 3

' Se encarga de realizar el logueo del usuario

Public Function LogIn(ByRef user As String, ByRef password As String) As ResultDTO

Dim result As ResultDTO

Try

' El proceso de logueo se basa en el cumplimiento de las reglas de negocio únicamente.

result = IsValid(user, password)

Return result

Catch ex As Exception

Return New ResultDTO(ResultDTO.type.EXCEPTION, "Se ha producido un error crítico: " + ex.Message)

End Try

End Function

' Valida el cumplimiento de los inputs y de las reglas de negocio

Private Function IsValid(ByRef user As String, ByRef password As String) As ResultDTO

Dim securityHelper As New SecurityHelper()

Dim logInDto As New LogInDTO()

Dim logInDal As New LogInDAL()

Dim userDto As UserDTO

Dim logBll As New LogBLL()

' 1. Chequeo de inputs

If Len(user) = 0 Or Len(password) = 0 Then

logBll.AddLogWarn("LogIn", "Campos incompletos.", Me)

Return New ResultDTO(ResultDTO.type.INCOMPLETE\_FIELDS, "Campos incompletos.")

End If

' 2. Chequeo de existencia de usuario

logInDto.user = user

logInDto.password = securityHelper.Encrypt(password)

userDto = logInDal.LogIn(logInDto)

' No se encontró el usuario para la combinación name + password... pero aún así puede existir el username y

' deben incrementarse los reintentos.

If IsNothing(userDto) Then

' Se busca el usuario pero sin usar el password

userDto = logInDal.GetUser(logInDto)

'Si existe, entonces se debe incrementar los reintentos y eventualmente lockearlo si los excedió

If Not IsNothing(userDto) Then

logInDal.IncrementRetries(logInDto)

userDto.retries += 1

If userDto.retries = MAX\_RETRIES Then

logInDal.LockUser(logInDto)

logBll.AddLogWarn("LogIn", "El usuario " + logInDto.user + " alcanzó los reintentos permitidos.", Me)

Return New ResultDTO(ResultDTO.type.MAX\_ATTEMPTS, "Credenciales inválidas: el usuario ha excedido la cantidad de reintentos.")

Else

logBll.AddLogWarn("LogIn", "Se intentó acceder con el usuario " + logInDto.user + ". Reintentos: " + CStr(userDto.retries), Me)

Return New ResultDTO(ResultDTO.type.INVALID\_CREDENTIAL, "Credenciales inválidas.")

End If

Else

logBll.AddLogWarn("LogIn", "Se intentó acceder con el usuario inexistente " + logInDto.user, Me)

Return New ResultDTO(ResultDTO.type.INVALID\_CREDENTIAL, "Credenciales inválidas.")

End If

Else

' 3. Chequeo de inconsistencia

If Not userDto.verified Then

logBll.AddLogCritical("LogIn", "Inconsistencia en los datos del usuario " + userDto.name + ".", Me)

Return New ResultDTO(ResultDTO.type.CHECKSUM\_ERROR, "El registro se encuentra corrupto.")

End If

' 4. Chequeo de usuario lockeado

If userDto.locked Then

logBll.AddLogWarn("LogIn", "El usuario ha excedido la cantidad de reintentos.", Me)

Return New ResultDTO(ResultDTO.type.MAX\_ATTEMPTS, "El usuario ha excedido la cantidad de reintentos.")

Else

logInDal.ResetRetries(logInDto)

userDto.retries = 0

logBll.AddLogInfo("LogIn", "El usuaro " + userDto.name + " se ha logueado exitosamente.", Me)

Return New ResultDTO(ResultDTO.type.OK, "Ok", userDto, True)

End If

End If

End Function

End Class

### LogBLL

Public Class LogBLL

' Obtiene los logs

Public Function GetLog(ByRef level As String, ByRef Optional since As String = Nothing, ByRef Optional until As String = Nothing, ByRef Optional entity As String = Nothing) As ResultDTO

Dim logDal As New LogDAL()

Dim logs As List(Of LogDTO)

Try

logs = logDal.GetLogs(level, since, until, entity)

Return New ResultDTO(ResultDTO.type.OK, "Ok", logs, True)

Catch ex As Exception

Return New ResultDTO(ResultDTO.type.EXCEPTION, "Se ha producido un error crítico: " + ex.Message)

End Try

End Function

' Crea un log de nivel info

Public Sub AddLogInfo(ByRef action As String, ByRef description As String, ByRef entity As Object)

CreateLog(LogDTO.level.INFO, action, description, entity.GetType().ToString())

End Sub

' Crea un log de nivel warn

Public Sub AddLogWarn(ByRef action As String, ByRef description As String, ByRef entity As Object)

CreateLog(LogDTO.level.WARNING, action, description, entity.GetType().ToString())

End Sub

' Crea un log de nivel critical

Public Sub AddLogCritical(ByRef action As String, ByRef description As String, ByRef entity As Object)

CreateLog(LogDTO.level.CRITICAL, action, description, entity.GetType().ToString())

End Sub

' Crea un log de nivel debug

Public Sub AddLogDebug(ByRef action As String, ByRef description As String, ByRef entity As Object)

CreateLog(LogDTO.level.DEBUG, action, description, entity.GetType().ToString())

End Sub

Private Sub CreateLog(ByRef loglevel As String, ByRef action As String, ByRef description As String, ByRef entity As String)

Dim logDal As New LogDAL()

Dim logDto As New LogDTO(loglevel, action, description, entity)

logDal.SaveLog(logDto)

End Sub

End Class

### CustomerBLL

Public Class CustomerBLL

' Trae el listado de clientes

Public Function GetCustomers() As ResultDTO

Dim customerDal As New CustomerDAL()

Dim logBll As New LogBLL()

Try

Dim result As New ResultDTO(ResultDTO.type.OK, "Ok", customerDal.GetCustomers(), True)

logBll.AddLogInfo("Get customers", "Listado de clientes", Me)

Return result

Catch ex As Exception

Return New ResultDTO(ResultDTO.type.EXCEPTION, "Se ha producido un error crítico: " + ex.Message)

End Try

End Function

'Se ingresa con un ID de cliente

Public Function GetCustomer(ByRef id As Integer) As ResultDTO

Dim customerDal As New CustomerDAL()

Dim logBll As New LogBLL()

Dim customer As CustomerDTO

customer = customerDal.GetCustomer(id)

If IsNothing(customer) Then

logBll.AddLogWarn("Get customer", "Búsqueda del cliente inexistente (id " + CStr(id) + ")", Me)

Else

logBll.AddLogInfo("Get customer", "Búsqueda del cliente " + customer.name + " " + customer.surname + " (id " + CStr(id) + ")", Me)

End If

Return New ResultDTO(ResultDTO.type.OK, "Ok", customer, True)

End Function

End Class

### CheckDataBaseBLL

Public Class CheckDataBaseBLL

'Se encarga de verificar el DVV de la base

Public Function CheckDVV() As ResultDTO

Dim logBll As New LogBLL()

Dim CheckDBDAL As New CheckDBDAL

Dim OriginDVV As dvvDTO

Dim CalculatedDVV As dvvDTO

Try

'Devuelvo el DVV calculado

OriginDVV = CheckDBDAL.getDVV

'Calculo el DVV

CalculatedDVV = CheckDBDAL.calculateDVV

'Comparo el DVV original contra el que fue calculado

If OriginDVV.dvv = CalculatedDVV.dvv Then

logBll.AddLogInfo("Check DVV", "La integridad del DVV esta ok", Me)

Return New ResultDTO(ResultDTO.type.OK, "DVV Ok")

Else

logBll.AddLogCritical("Check DVV", "El DVV fue Corrupto, realizar Restore.", Me)

Return New ResultDTO(ResultDTO.type.CORRUPTED\_DATABASE, "Error en el DVV, DB Corrupta")

End If

Catch ex As Exception

Return New ResultDTO(ResultDTO.type.EXCEPTION, "Se ha producido un error crítico: " + ex.Message)

End Try

End Function

End Class

### BackupRestoreBLL

Public Class BackupRestoreBLL

Public Sub doBackup()

Dim LogBLL As New LogBLL

Dim BackUpRestoreDAL As New BackUpRestoreDAL

Dim result As Object

result = BackUpRestoreDAL.doBackup()

If result = True Then

LogBLL.AddLogInfo("Backup", "Se realizo Backup de la DB", Me)

Else

LogBLL.AddLogCritical("Backup", "No se pudo realizar BackUp.", Me)

End If

End Sub

Public Sub doRestore()

Dim LogBLL As New LogBLL

Dim BackUpRestoreDAL As New BackUpRestoreDAL

Dim result As Object

result = BackUpRestoreDAL.doBackup()

If result = True Then

LogBLL.AddLogInfo("Restore", "Se realizo Restore de la DB", Me)

Else

LogBLL.AddLogCritical("Backup", "No se pudo realizar Restore.", Me)

End If

End Sub

End Class

### ProductBLL

Public Class ProductBLL

'Trae el listado de los productos

Public Function GetProducts() As ResultDTO

Dim productDal As New ProductDAL()

Dim logBll As New LogBLL()

Try

Dim result As New ResultDTO(ResultDTO.type.OK, "Ok", productDal.GetProducts(), True)

logBll.AddLogInfo("Get products", "Listado de productos", Me)

Return result

Catch ex As Exception

Return New ResultDTO(ResultDTO.type.EXCEPTION, "Se ha producido un error crítico: " + ex.Message)

End Try

End Function

'Busqueda por ID de producto

Public Function GetProduct(ByRef id As Integer) As ResultDTO

Dim productDal As New ProductDAL()

Dim logBll As New LogBLL()

Dim product As ProductDTO

product = productDal.GetProduct(id)

If IsNothing(product) Then

logBll.AddLogWarn("Get product", "Búsqueda del producto inexistente (id " + CStr(id) + ")", Me)

Else

logBll.AddLogInfo("Get product", "Búsqueda del producto " + product.name + " (id " + CStr(id) + ")", Me)

End If

Return New ResultDTO(ResultDTO.type.OK, "Ok", product, True)

End Function

End Class

### Webservice.asmx

|  |  |  |
| --- | --- | --- |
| **Nombre WebService** | **Descripción** | **Quien lo Consume.** |
| Webservice.asmx | Recibe un precio de lista (producto de los productos que forman parte del carrito del cliente). Si dicho precio de lista es mayor a 500, entonces realiza un descuento del 10%. Caso contrario, devuelve el mismo precio recibido. | Orders.aspx |

Imports System.Web

Imports System.Web.Services

Imports DataTypeObject

Imports System.Web.Services.Protocols

' Para permitir que se llame a este servicio web desde un script, usando ASP.NET AJAX, quite la marca de comentario de la línea siguiente.

' <System.Web.Script.Services.ScriptService()> \_

<WebService(Namespace:="http://tempuri.org/")> \_

<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1\_1)> \_

<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> \_

Public Class WebService

Inherits System.Web.Services.WebService

<WebMethod()>

Public Function HelloWorld() As String

Return "Hola a todos"

End Function

<WebMethod()>

Public Function CalculateDiscount(listPrice As Integer) As Integer

Dim finalPrice As Integer

If listPrice > 500 Then

finalPrice = 500 \* 0.9

Else

finalPrice = listPrice

End If

Return finalPrice

End Function

End Class

# Capa de DATOS (DAL)

### Código BD Script de Creación

USE [master]

GO

/\*\*\*\*\*\* Object: Database [LPPA] Script Date: 26/11/2016 17:24:34 \*\*\*\*\*\*/

CREATE DATABASE [LPPA]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'LPPA', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL12.UAI\_EXPRESS\MSSQL\DATA\LPPA.mdf' , SIZE = 5120KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )

LOG ON

( NAME = N'LPPA\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL12.UAI\_EXPRESS\MSSQL\DATA\LPPA\_log.ldf' , SIZE = 2048KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)

GO

ALTER DATABASE [LPPA] SET COMPATIBILITY\_LEVEL = 120

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [LPPA].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [LPPA] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [LPPA] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [LPPA] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [LPPA] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [LPPA] SET ARITHABORT OFF

GO

ALTER DATABASE [LPPA] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [LPPA] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [LPPA] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [LPPA] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [LPPA] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [LPPA] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [LPPA] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [LPPA] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [LPPA] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [LPPA] SET DISABLE\_BROKER

GO

ALTER DATABASE [LPPA] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [LPPA] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [LPPA] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [LPPA] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [LPPA] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [LPPA] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [LPPA] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [LPPA] SET RECOVERY SIMPLE

GO

ALTER DATABASE [LPPA] SET MULTI\_USER

GO

ALTER DATABASE [LPPA] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [LPPA] SET DB\_CHAINING OFF

GO

ALTER DATABASE [LPPA] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [LPPA] SET TARGET\_RECOVERY\_TIME = 0 SECONDS

GO

ALTER DATABASE [LPPA] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [LPPA] SET READ\_WRITE

GO

### Conexión con la Base de Datos

Public Class DBSql

'Private connStr = "Data Source=DESKTOP-89RB8TN\SQLEXPRESS;Initial Catalog=LPPA;Integrated Security=True"

Private connStr = "Data Source=.\UAI\_EXPRESS; Initial Catalog=LPPA; Integrated Security=True"

Public Function ExecuteNonQuery(ByRef sql As String)

Dim connection As New SqlConnection(connStr)

Dim command As New SqlCommand(sql, connection)

Dim result As Integer

connection.Open()

result = command.ExecuteScalar()

connection.Close()

Return result

End Function

Public Function ExecuteReader(ByRef sql As String) As List(Of List(Of String))

Dim connection As New SqlConnection(connStr)

Dim command As New SqlCommand(sql, connection)

Dim reader As SqlDataReader

Dim result As New List(Of List(Of String))

connection.Open()

reader = command.ExecuteReader()

If reader.HasRows Then

Do While reader.Read()

result.Add(Resolve(reader))

Loop

End If

connection.Close()

Return result

End Function

' Devuelve una lista de cadenas de texto respetando el orden de los campos

Private Function Resolve(ByRef reader As SqlDataReader) As List(Of String)

Dim result As New List(Of String)

For i As Integer = 0 To reader.FieldCount - 1

result.Add(reader.Item(i).ToString())

Next

Return result

End Function

End Class

### LogInDAL

Public Class LogInDAL

' Busca al usuario que se corresponde con las credenciales provistas

Public Function LogIn(ByRef userLogin As LogInDTO) As UserDTO

Dim dbsql As New DBSql

Dim sql As String

Dim reader As List(Of List(Of String))

sql = "SELECT name, locked, retries, CASE WHEN hdigit = BINARY\_CHECKSUM(name, password) then 1 ELSE 0 END verified FROM users "

sql += "WHERE name = '" + userLogin.user + "' AND password = '" + userLogin.password + "'"

reader = dbsql.ExecuteReader(sql)

If reader.Count > 0 Then

Return Resolve(reader.First)

Else

Return Nothing

End If

End Function

' Busca al usuario que se corresponde con las credenciales provistas

Public Function GetUser(ByRef userLogin As LogInDTO) As UserDTO

Dim dbsql As New DBSql

Dim sql As String

Dim reader As List(Of List(Of String))

sql = "SELECT name, locked, retries, CASE WHEN hdigit = BINARY\_CHECKSUM(name, password) then 1 ELSE 0 END verified FROM users "

sql += "WHERE name = '" + userLogin.user + "'"

reader = dbsql.ExecuteReader(sql)

If reader.Count > 0 Then

Return Resolve(reader.First)

Else

Return Nothing

End If

End Function

' Incrementa en uno los reintentos de logueo

Public Sub IncrementRetries(ByRef userLogin As LogInDTO)

Dim dbsql As New DBSql

Dim sql As String

Dim result As Object

sql = "UPDATE users SET retries = retries + 1 WHERE name = '" + userLogin.user + "'"

result = dbsql.ExecuteNonQuery(sql)

End Sub

' Resetea los reintentos

Public Sub ResetRetries(ByRef userLogin As LogInDTO)

Dim dbsql As New DBSql

Dim sql As String

sql = "UPDATE users SET retries = 0 WHERE name = '" + userLogin.user + "'"

dbsql.ExecuteNonQuery(sql)

End Sub

' Lockea al usuario

Public Sub LockUser(ByRef userLogin As LogInDTO)

Dim dbsql As New DBSql

Dim sql As String

sql = "UPDATE users SET locked = 1 WHERE name = '" + userLogin.user + "'"

dbsql.ExecuteNonQuery(sql)

End Sub

' Deslockea al usuario y resetea los retries

Public Sub UnlockUser(ByRef userLogin As LogInDTO)

Dim dbsql As New DBSql

Dim sql As String

sql = "UPDATE users SET locked = 0, retries = 0 WHERE name = '" + userLogin.user + "'"

dbsql.ExecuteNonQuery(sql)

End Sub

Private Function Resolve(ByRef item As List(Of String)) As UserDTO

Dim result As New UserDTO

result.name = CStr(item.Item(0))

result.locked = CBool(item.Item(1))

result.retries = CInt(item.Item(2))

result.verified = CBool(item.Item(3))

Return result

End Function

End Class

### LogDAL

Public Class LogDAL

' Busca los logs que se corresponden con los criterios

Public Function GetLogs(level As String, ByRef since As String, ByRef until As String, ByRef entity As String) As List(Of LogDTO)

Dim dbsql As New DBSql

Dim sql As String

Dim reader As List(Of List(Of String))

Dim result As New List(Of LogDTO)

Dim dateCriterion As String = ""

Dim entityCriterion As String = ""

If Len(since) > 0 And Len(until) > 0 Then

dateCriterion = " AND created BETWEEN convert(datetime, '" + since + "', 103) AND convert(datetime,'" + until + "', 103)"

ElseIf Len(since) > 0 And Len(until) = 0 Then

dateCriterion = " AND created >= convert(datetime,'" + since + "', 103)"

ElseIf Len(since) = 0 And Len(until) > 0 Then

dateCriterion = " AND created <= convert(datetime,'" + until + "', 103)"

End If

If Len(entityCriterion) > 0 Then

entityCriterion = " AND entity = '" + entity + "'"

End If

sql = "SELECT \* FROM logs WHERE loglevel >= " + level + entityCriterion + dateCriterion

reader = dbsql.ExecuteReader(sql)

If reader.Count > 0 Then

For i As Integer = 0 To reader.Count - 1

result.Add(Resolve(reader.Item(i)))

Next

End If

Return result

End Function

Public Sub SaveLog(ByRef log As LogDTO)

Dim dbsql As New DBSql

Dim sql As String

sql = "INSERT INTO logs (loglevel, action, description, entity, created) "

sql += " VALUES ("

sql += CStr(log.logLevel) + ", "

sql += "'" + log.action + "' , "

sql += "'" + log.description + "' , "

sql += "'" + log.entity + "' , "

sql += "GETDATE()"

sql += "); SELECT @@IDENTITY"

log.id = dbsql.ExecuteNonQuery(sql)

End Sub

Public Sub DeleteLogs()

Dim dbsql As New DBSql

dbsql.ExecuteNonQuery("TRUNCATE TABLE logs")

End Sub

Private Function Resolve(ByRef item As List(Of String)) As LogDTO

Dim result As New LogDTO()

result.id = item.Item(0)

result.logLevel = CInt(item.Item(1))

result.action = item.Item(2)

result.description = item.Item(3)

result.entity = item.Item(4)

result.created = item.Item(5)

Return result

End Function

End Class

### CustomerDAL

Public Class CustomerDAL

Public Function GetCustomers() As List(Of CustomerDTO)

Return ListOfCustomers()

End Function

Public Function GetCustomer(ByRef id As Integer) As CustomerDTO

Dim result As List(Of CustomerDTO)

result = ListOfCustomers()

For Each customer As CustomerDTO In result

If customer.id = id Then

Return customer

End If

Next

Return Nothing

End Function

Private Function ListOfCustomers() As List(Of CustomerDTO)

Dim result As New List(Of CustomerDTO)

result.Add(New CustomerDTO(1, "Alberto", "Gonzales", New Date(1981, 11, 17), "M"))

result.Add(New CustomerDTO(2, "Agustín", "Correa", New Date(1983, 12, 1), "M"))

result.Add(New CustomerDTO(3, "Juana", "Molina", New Date(1984, 9, 4), "F"))

result.Add(New CustomerDTO(4, "Laura", "Martínez", New Date(1986, 7, 30), "F"))

result.Add(New CustomerDTO(5, "Bárbara", "Monte", New Date(1990, 6, 28), "F"))

result.Add(New CustomerDTO(6, "Claudia", "Terremoto", New Date(1995, 5, 9), "F"))

Return result

End Function

End Class

### CheckDBDAL

Public Class CheckDBDAL

'Obtengo el DVV guardado en la Base

Public Function getDVV() As dvvDTO

Dim dbsql As New DBSql

Dim sql As String

Dim reader As List(Of List(Of String))

sql = "SELECT \* FROM dvv"

reader = dbsql.ExecuteReader(sql)

If reader.Count > 0 Then

Return ResolveDVV(reader.First)

Else

Return Nothing

End If

End Function

'Calculo el DVV

Public Function calculateDVV() As dvvDTO

Dim dbsql As New DBSql

Dim sql As String

Dim reader As List(Of List(Of String))

sql = "SELECT CHECKSUM\_AGG(hdigit) FROM users"

reader = dbsql.ExecuteReader(sql)

If reader.Count > 0 Then

Return ResolveDVV(reader.First)

Else

Return Nothing

End If

End Function

Private Function ResolveDVH(ByRef item As List(Of String)) As dvhDTO

Dim result As New dvhDTO

result.name = CStr(item.Item(0))

result.newDvh = CInt(item.Item(2))

result.dvh = CInt(item.Item(1))

Return result

End Function

Private Function ResolveDVV(ByRef item As List(Of String)) As dvvDTO

Dim result As New dvvDTO

result.dvv = CStr(item.Item(0))

Return result

End Function

End Class

### BackupRestoreDAL

Public Class BackUpRestoreDAL

Public Function doBackup() As Object

Dim dbsql As New DBSql

Dim sql As String

Dim result As Object

sql = "BACKUP DATABASE LPPA TO DISK = 'C:\LPPA\Dushana\Backup\LPPA.BAK'"

Return result = dbsql.ExecuteNonQuery(sql)

End Function

Public Function doRestore() As Object

Dim dbsql As New DBSql

Dim sql As String

Dim result As Object

sql = "USE MASTER RESTORE DATABASE LPPA FROM DISK = 'C:\LPPA\Dushana\Backup\LPPA.BAK' WITH REPLACE"

Return result = dbsql.ExecuteNonQuery(sql)

End Function

End Class

### ProductDAL

Public Class ProductDAL

Public Function GetProducts() As List(Of ProductDTO)

Return ListOfProducts()

End Function

Public Function GetProduct(ByRef id As Integer) As ProductDTO

Dim result As List(Of ProductDTO)

result = ListOfProducts()

For Each customer As ProductDTO In result

If customer.id = id Then

Return customer

End If

Next

Return Nothing

End Function

Private Function ListOfProducts() As List(Of ProductDTO)

Dim result As New List(Of ProductDTO)

result.Add(New ProductDTO(1, "Peeling", "El peeling consigue una disminución de las arrugas, una piel rejuvenecida y bien hidratada, con una secreción correcta de grasa y una buena consistencia y luminosidad.", 300))

result.Add(New ProductDTO(2, "Drenaje linfático", "El drenaje linfático manual es una técnica de masoterapia terapéutica.", 250))

result.Add(New ProductDTO(3, "Masajes descontracturantes", "Masajes.", 400))

result.Add(New ProductDTO(4, "Depilación definitiva", "Depilación definitiva.", 275))

result.Add(New ProductDTO(5, "Cavitación", "Tratamiento médico de estética que consiste en aplicar ultrasonidos en determinadas zonas del cuerpo para eliminar la grasa en ellas localizadas.", 425))

Return result

End Function

End Class

# Capa de Seguridad (Helper )

### SecurityHelper

Imports System.Security.Cryptography

Imports System.Text

Public Class SecurityHelper

'Se encarga de aplicar la encriptación md5

Public Function Encrypt(ByRef value As String) As String

Dim Bytes() As Byte

Dim sb As New StringBuilder()

If String.IsNullOrEmpty(value) Then

Throw New ArgumentNullException

End If

Bytes = Encoding.Default.GetBytes(value)

Bytes = MD5.Create().ComputeHash(Bytes)

For x As Integer = 0 To Bytes.Length - 1

sb.Append(Bytes(x).ToString("x2"))

Next

Return sb.ToString()

End Function

End Class

# Capa de Entidades (DTO)

### LogInDTO

Public Class LogInDTO

' Modelado del log in

Private \_user As String

Private \_password As String

Property user As String

Get

Return \_user

End Get

Set(value As String)

\_user = value

End Set

End Property

Property password As String

Get

Return \_password

End Get

Set(value As String)

\_password = value

End Set

End Property

End Class

### UserDTO

Public Class UserDTO

' Modelado del usuario

' No se incluye el password para evitar que siempre se transporte y se exponga.

Private \_name As String

Private \_locked As Boolean

Private \_retries As Integer

Private \_verified As Boolean

Property name As String

Get

Return \_name

End Get

Set(value As String)

\_name = value

End Set

End Property

Property locked As Boolean

Get

Return \_locked

End Get

Set(value As Boolean)

\_locked = value

End Set

End Property

Property retries As Integer

Get

Return \_retries

End Get

Set(value As Integer)

\_retries = value

End Set

End Property

Property verified As Boolean

Get

Return \_verified

End Get

Set(value As Boolean)

\_verified = value

End Set

End Property

End Class

### LogDTO

Public Class LogDTO

' Modela el registro de log

Enum level

DEBUG = 1

WARNING = 2

CRITICAL = 3

INFO = 4

End Enum

Private \_id As Integer

Private \_logLevel As Integer

Private \_action As String

Private \_description As String

Private \_entity As String

Private \_created As String

Public Sub New()

End Sub

Public Sub New(ByRef loglevel As String, ByRef action As String, ByRef description As String, ByRef entity As String)

\_logLevel = loglevel

\_action = action

\_description = description

\_entity = entity

' La fecha es agregada por la base de datos para evitar problemas de incompatibilidad

End Sub

Property id As Integer

Set(value As Integer)

\_id = value

End Set

Get

Return \_id

End Get

End Property

Property logLevel As Integer

Set(value As Integer)

\_logLevel = value

End Set

Get

Return \_logLevel

End Get

End Property

Property action As String

Set(value As String)

\_action = value

End Set

Get

Return \_action

End Get

End Property

Property description As String

Set(value As String)

\_description = value

End Set

Get

Return \_description

End Get

End Property

Property entity As String

Set(value As String)

\_entity = value

End Set

Get

Return \_entity

End Get

End Property

Property created As String

Set(value As String)

\_created = value

End Set

Get

Return \_created

End Get

End Property

End Class

### ProductDTO

Public Class ProductDTO

Private \_id As Integer

Private \_name As String

Private \_description As String

Private \_price As Integer

Public Sub New(ByRef id As Integer, ByRef name As String, ByRef description As String, ByRef price As Double)

\_id = id

\_name = name

\_description = description

\_price = price

End Sub

Property id As Integer

Set(value As Integer)

\_id = value

End Set

Get

Return \_id

End Get

End Property

Property name As String

Set(value As String)

\_name = value

End Set

Get

Return \_name

End Get

End Property

Property description As String

Set(value As String)

\_description = value

End Set

Get

Return \_description

End Get

End Property

Property prices As Integer

Set(value As Integer)

\_price = value

End Set

Get

Return \_price

End Get

End Property

End Class

### ResultDTO

Public Class ResultDTO

Enum type

OK

INVALID\_CREDENTIAL

CHECKSUM\_ERROR

MAX\_ATTEMPTS

INCOMPLETE\_FIELDS

EXCEPTION

CORRUPTED\_DATABASE

End Enum

Private \_type As type

Private \_description As String

Private \_keepGoing As Boolean

Private \_value As Object

Public Sub New(ByRef type As type, descripton As String, Optional keepGoing As Boolean = False)

\_type = type

\_description = descripton

\_keepGoing = keepGoing

End Sub

Public Sub New(ByRef type As type, descripton As String, value As Object, Optional keepGoing As Boolean = False)

\_type = type

\_description = descripton

\_keepGoing = keepGoing

\_value = value

End Sub

' Descripción del error

ReadOnly Property description As String

Get

Return \_description

End Get

End Property

' Indica si el error es suficientemente crítico como para que no se continue el flujo normal

ReadOnly Property keepGoing As Boolean

Get

Return \_keepGoing

End Get

End Property

' Almacena un objeto que puede ser el resultado de una operación exitosa o fallida, y sobre el que se requiere realizar una operación.

Property value As Object

Set(value As Object)

\_value = value

End Set

Get

Return \_value

End Get

End Property

' Informa si el error actual es OK

Public Function IsValid() As Boolean

Return IsCurrentError(type.OK)

End Function

' Informa si el error actual coincide con el pasado por parámetro

Public Function IsCurrentError(ByRef type As type) As Boolean

Return \_type = type

End Function

End Class

### CustomerDTO

Public Class CustomerDTO

'Modela el cliente

Private \_id As Integer

Private \_name As String

Private \_surname As String

Private \_birth As String

Private \_gender As String

Public Sub New(ByRef id As Integer, ByRef name As String, ByRef surname As String, ByRef birth As String, ByRef gender As String)

\_id = id

\_name = name

\_surname = surname

\_birth = birth

\_gender = gender

End Sub

Property id As Integer

Set(value As Integer)

\_id = id

End Set

Get

Return \_id

End Get

End Property

Property name As String

Set(value As String)

\_name = value

End Set

Get

Return \_name

End Get

End Property

Property surname As String

Set(value As String)

\_surname = value

End Set

Get

Return \_surname

End Get

End Property

Property birth As String

Set(value As String)

\_birth = value

End Set

Get

Return \_birth

End Get

End Property

Property gender As String

Set(value As String)

\_gender = value

End Set

Get

Return \_gender

End Get

End Property

End Class

### DvhDTO

Public Class dvhDTO

Private \_name As String

Private \_dvh As Integer

Private \_newDvh As Integer

Property name As String

Get

Return \_name

End Get

Set(value As String)

\_name = value

End Set

End Property

Property dvh As Integer

Get

Return \_dvh

End Get

Set(value As Integer)

\_dvh = value

End Set

End Property

Property newDvh As Integer

Get

Return \_newDvh

End Get

Set(value As Integer)

\_newDvh = value

End Set

End Property

End Class

### DvvDTO

Public Class dvvDTO

'Modelado del DVV

Private \_dvv As String

Property dvv As String

Get

Return \_dvv

End Get

Set(value As String)

\_dvv = value

End Set

End Property

End Class

# Archivos de configuración

### Web.config

<?xml version="1.0" encoding="utf-8"?>

<!--

Para obtener más información sobre cómo configurar la aplicación de ASP.NET, visite

http://go.microsoft.com/fwlink/?LinkId=169433

-->

<configuration>

<system.web>

<compilation debug="true" strict="false" explicit="true" targetFramework="4.5.2" />

<httpRuntime targetFramework="4.5.2" />

</system.web>

</configuration>