**UNIVERSIDAD ABIERTA INTERAMERICANA**

**Facultad de Tecnología Informática**

Logotipo

Descripción generada automáticamente  
  
**Trabajo de Investigación – Grupo 2:**  
Aplicación del viento

**Materia:** Metodología y Desarrollo web

**Docente:** Santiago Juan Sábato

**Alumnos:**

Castillo Bryan

Fraguaga Lucas

Muñiz Lucía

**Año:** 2024

Contenido

[1.Introducción 3](#_Toc180759103)

[Descripción Global del Producto 3](#_Toc180759104)

[Propósito 3](#_Toc180759105)

[2.Diagramas Login 4](#_Toc180759106)

[Caso de Uso 4](#_Toc180759107)

[Diagrama de Secuencia 5](#_Toc180759108)

[Diagrama de clases 6](#_Toc180759109)

[DER 6](#_Toc180759110)

[Diagrama general de secuencia para la autenticación 7](#_Toc180759111)

[3.Seguridad 7](#_Toc180759112)

[Sistema de autenticación 7](#_Toc180759113)

[Registración 7](#_Toc180759114)

[Registración de clientes 7](#_Toc180759115)

[Inicio de sesión (Login) 7](#_Toc180759116)

[Cerrado de sesión (Logout) 8](#_Toc180759117)

[Gestión de Encriptado 8](#_Toc180759118)

[4.Codigo 11](#_Toc180759119)

[Pantallas 11](#_Toc180759120)

[PRINCIPAL (logeo) 11](#_Toc180759121)

[REGISTRO 12](#_Toc180759122)

[USER 13](#_Toc180759123)

[CARRITO DE COMPRAS 14](#_Toc180759124)

[PAGO 15](#_Toc180759125)

[ADMIN 16](#_Toc180759126)

[BITACORA 17](#_Toc180759127)

[BLOQUEO USUARIO 18](#_Toc180759128)

[ESTADISTICAS VENTA 19](#_Toc180759129)

[UI 19](#_Toc180759130)

[Admin.aspx.cs 19](#_Toc180759131)

[CarritoCompra.aspx.cs 22](#_Toc180759132)

[EstadisticasVenta.aspx.cs 24](#_Toc180759133)

[Pago.aspx.cs 26](#_Toc180759134)

[PantallaBitacora.aspx.cs 27](#_Toc180759135)

[PantallaUsuario.aspx.cs 29](#_Toc180759136)

[Principal.aspx.cs 30](#_Toc180759137)

[Registro.aspx.cs 32](#_Toc180759138)

[User.aspx.cs 33](#_Toc180759139)

[WebServices 34](#_Toc180759140)

[WebServiceEstadisticosVenta.asmx 34](#_Toc180759141)

[WebServicePago.asmx 35](#_Toc180759142)

[BE 36](#_Toc180759143)

[Bitacora.cs 36](#_Toc180759144)

[ClienteGasto.cs 37](#_Toc180759145)

[Producto.cs 37](#_Toc180759146)

[Rol.cs 38](#_Toc180759147)

[Usuario.cs 39](#_Toc180759148)

[Venta.cs 40](#_Toc180759149)

[BLL 40](#_Toc180759150)

[BLLConexion.cs 40](#_Toc180759151)

[BLLEncriptado.cs 42](#_Toc180759152)

[BLLGestorXML.cs 42](#_Toc180759153)

[BLLSesionManager.cs 43](#_Toc180759154)

[DAL 44](#_Toc180759155)

[Conexión.cs 44](#_Toc180759156)

[5.Diagrama loco 54](#_Toc180759157)

# 1.Introducción

## Descripción Global del Producto

El pronóstico del tiempo es una herramienta de vital importancia cuando debe utlizarse como herramienta para tomar decisiones en la agricultura.

Existen pequeños y medianos agricultores que deben invertir mucho tiempo de análisis e investigación en los fenómenos naturales que pueden afectar su producción, como por ejemplo tormentas, sondas, lluvias y granizos. Estos fenómenos no resultan irrelevantes ya que impactan en todos los estadíos del proceso del cultivo, y se deben tomar medidas preventivas ya que una planificación errónea puede significar pérdidas monetarias.

Nuestro sistema ofrece una plataforma web donde aquellos que contraten el servicio de predicción puedan acceder fácilmente con sus credenciales para configurar alertas de ciertos fenómenos meteorológicos y también mediciones que resulten de interés de acuerdo a la región del cultivo.

## Propósito

El producto pretende cubrir la necesidad de análisis y medición del tiempo meteorológico para alertar sobre aquellos fenómenos relevantes, automatizando la tarea de búsqueda y revisión del estado del tiempo. Así es más eficaz para los usuarios tomar medidas preventivas para minimizar el impacto en los ciclos de cultivo.

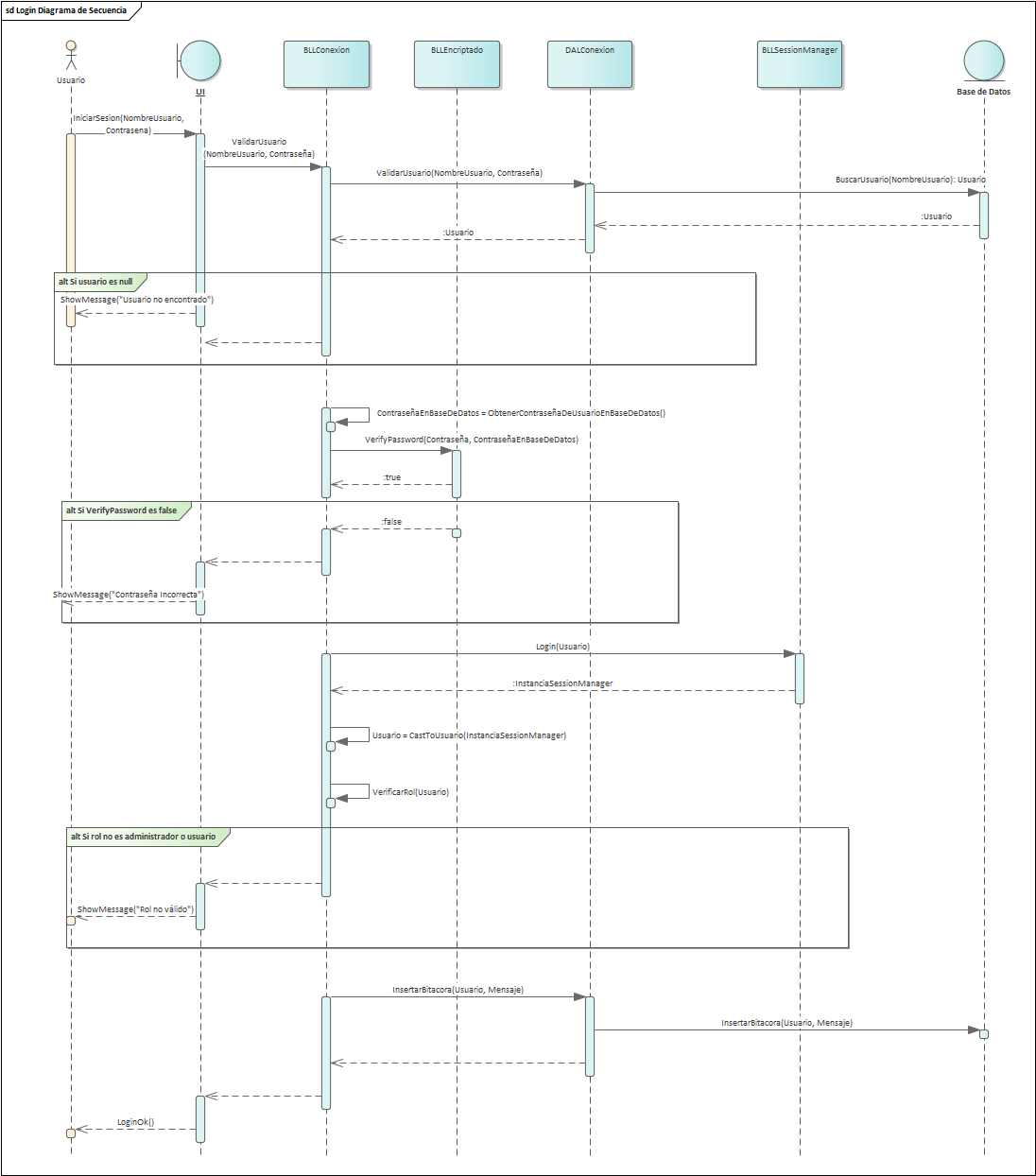
# 2.Diagramas Login

## Caso de Uso

Gráfico, Gráfico de burbujas

Descripción generada automáticamente

## Diagrama de Secuencia



## Diagrama de clases

Diagrama

Descripción generada automáticamente

## DER

Diagrama

Descripción generada automáticamente

## Diagrama general de secuencia para la autenticación

# 3.Seguridad

## Sistema de autenticación

El usuario será autenticado en el sistema ingresando su nombre de usuario y su contraseña. Ambos datos deben encontrarse persistidos en la base de datos previamente.

## Registración

En nuestro sistema la registración de un usuario va a depender del rol que corresponda al usuario a registrar. Un usuario administrador va a poder dar de alta a otros administradores o clientes (usuarios con menos privilegios), y va a poder elegir el nombre de usuario y una contraseña temporal que el usuario dado de alta deberá cambiar posteriormente.

## Registración de clientes

La registración de un cliente tiene como disparador cuando una persona (no administradora) envía una solicitud de contratación al servicio que proporciona nuestro sistema. Cuando se recibe la solicitud de contratación, el administrador tiene los datos suficientes para dar de alta al usuario en el sistema. Por defecto el nombre de usuario serán las dos iniciales del primer nombre, más el apellido y dos dígitos que funcionan a modo de contador, por ejemplo para el caso de un cliente llamado Mario Lopez , el administrador pondrá como nombre de usuario malopez01. La contraseña inicializada por defecto será una combinación de caracteres aleatorios que deberá cambiar el usuario una vez que inicie sesión por primera vez en el sistema.

## Inicio de sesión (Login)

Para poder utilizar la mayoría de las funcionalidades del sistema el usuario deberá iniciar sesión. Para poder iniciar sesión, el usuario deberá ingresar su nombre de usuario y su contraseña. El sistema deberá comprobar que los datos ingresados sean correctos. Se comprueba que el nombre de usuario exista en la base de datos y luego se procede a verificar la contraseña. Para verificar si la contraseña ingresada es correcta, esta se encripta con un HASH y se compara con la contraseña guardada. La acción de login va a ser registrada den la bitácora.

## Cerrado de sesión (Logout)

Una vez que el usuario haya finalizado su trabajo en el sistema, el mismo procederá a cerrar la sesión. Cuando el usuario solicite cerrar la sesión, el sistema deberá registrar la acción en la bitácora y eliminar los datos temporales que se posean del usuario. Una vez finalizados los pasos anteriores, se redirigirá al menú de inicio de sesión

## Gestión de Encriptado

Para la proteger los datos importantes del sistema se utilizarán dos formas de encriptación diferentes. Un cifrado irreversible para los datos como por ejemplo las contraseñas, y una encriptación reversible para los demás datos que puedan necesitar recuperarse posteriormente del encriptado.

**Cifrado irreversible**: Para poder cifrar un dato de una forma irreversible, tal que no se pueda volver a obtener el valor original, se utilizará el método de cifrado irreversible SHA256 (Security Hash Algorithm 256 bits) Para utilizar el método SHA256, se necesita proveerle de una cadena de caracteres como parámetro de entrada. Se obtendrá una clave estática la cual es utilizada para encriptar el dato y se aplica el método. Una vez aplicado el algoritmo, se obtiene un array de bytes. Al array de bytes se le convertirá nuevamente a una cadena de caracteres para que pueda ser almacenada en la base de datos.

Diagrama

Descripción generada automáticamente

**Encriptación reversible**: Para poder encriptar un dato de una forma reversible, en la cual se puede volver a aplicar un algoritmo de desencriptación posteriormente, se utilizará un algoritmo de encriptación reversible llamado AES (Advanced Encryption Estándar). Para utilizar este método de encriptación, obtendremos una clave estática la cual se aplicará en el algoritmo junto a una cadena de caracteres que será especificada en el parámetro de entrada. Se procede a encriptar el dato y se obtiene un array de bytes. El array de bytes es convertido es una cadena de caracteres para poder ser almacenado en la base de datos.

**Diagrama

Descripción generada automáticamente**

# 4.Codigo

## Pantallas

|  |
| --- |
| Presentación |
|  |
| PRINCIPAL (logeo) |
|  |

|  |
| --- |
| REGISTRO |
|  |

|  |
| --- |
| USER |
|  |

|  |
| --- |
| CARRITO DE COMPRAS |
|  |

|  |
| --- |
| PAGO |
|  |

|  |
| --- |
| ADMIN |
|  |

|  |
| --- |
| BITACORA |
|  |

|  |
| --- |
| BLOQUEO USUARIO |
|  |

|  |
| --- |
| ESTADISTICAS VENTA |
|  |

## UI

|  |
| --- |
| Admin.aspx.cs |
| public partial class admin : System.Web.UI.Page  {  BLLConexion conexion = new BLLConexion();  protected void Page\_Load(object sender, EventArgs e)  {  //llenado de datos de sesion  Label1.Text = (BLL.BLLSesionManager.GetInstance).Usuario.Usu;  Label2.Text = "WebMaster";  //Label2.Text = (BLL.BLLSesionManager.GetInstance).Usuario.Roll.ToString();  //verificado el estado de loss registros de bitacora e usuario y mensaje si estan corruptos o no  List<Bitacora> bitacora = new List<Bitacora>();  List<Usuario> listUsuarios = new List<Usuario>();  bitacora = conexion.listarBitacora();  listUsuarios = conexion.ListarUsuario();  string auxBitacora = "La tabla Bitacora tiene registros corruptos y son: ";  string auxUsuario = "La tabla usuario tiene registros corruptos y son: ";  bool valBitacora = false;  bool valUsuario = false;  foreach (var item in bitacora)  {  if(item.Corrupta == "true")  {  valBitacora = true;  auxBitacora += item.ID.ToString() + " ";  }  }  foreach (var item in listUsuarios)  {  if (item.Corrupto == "true")  {  valUsuario = true;  auxUsuario += item.Id.ToString() + " ";  }  }  string auxMensaje = "";  if (valBitacora)  {  auxMensaje += auxBitacora;  }  if (valUsuario)  {  auxMensaje += auxUsuario;  }  //registro de log si estan corruptos o no  if (valBitacora || valUsuario)  {  ScriptManager.RegisterStartupScript(this, this.GetType(), "corrupto", $"alert('{auxMensaje}');", true);  }  }  protected void Button1\_Click(object sender, EventArgs e)  {  //logica de desconexion e log  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "deslogeo de sesion admin");  BLL.BLLSesionManager.logaut();  string script = "window.onload = function(){ alert('Sesión cerrada correctamente'); window.location.href = 'Principal.aspx'; }";  ClientScript.RegisterStartupScript(this.GetType(), "alert", script, true);  }  //funcion para mostrar mensajes  private void ShowMessage(string message)  {  string script = $"alert('{message}');";  ScriptManager.RegisterStartupScript(this, GetType(), "showalert", script, true);  }  protected void Button2\_Click(object sender, EventArgs e)  {  //logica para ir a pantalla de usuario e log  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "pantalla de usuarios abierta");  Response.Redirect("PantallaUsuarios.aspx");  }  protected void Button4\_Click(object sender, EventArgs e)  {  //logica para ir a pantalla de bitacora e log  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "pantalla de bitacora abierta");  Response.Redirect("PantallaBitacora.aspx");  }  protected void Button5\_Click(object sender, EventArgs e)  {  //logica para hacer el backup con mensaje si se pudo o no e logs  if (conexion.generarBackup())  {  ShowMessage("Backup exitoso");  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "se genero un backup de la base de datos");  }  else  {  ShowMessage("Backup no exitoso");  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "no se genero un backup de la base de datos");  }  }  protected void Button6\_Click(object sender, EventArgs e)  {  //logica para hacer la restauracion de la base y log de si se pudo o no  if (conexion.restaurarBase())  {  ScriptManager.RegisterStartupScript(this, this.GetType(), "backupSuccess", "alert('Backup realizado correctamente.');", true);  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "se restauro la base a la ultima version disponible");  }  else  {  ScriptManager.RegisterStartupScript(this, this.GetType(), "backupError", $"alert('Error al realizar el backup');", true);  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "no se pudo restaurar la base");  }  }  protected void Button7\_Click(object sender, EventArgs e)  {  //creacion del xml de venta y redireccion a pantalla de estadisticas de venta  BLLGestorXML xml = new BLLGestorXML();  xml.CrearXmlDeVentas(conexion.ObtenerVentasCompleto());  Response.Redirect("EstadisticaVenta.aspx");  }  } |

|  |
| --- |
| CarritoCompra.aspx.cs |
| public partial class CarritoCompra : System.Web.UI.Page  {  protected void Page\_Load(object sender, EventArgs e)  {  // instancia de usuario a mostrar  Label1.Text = BLLSesionManager.GetInstance.Usuario.Usu;  if (!IsPostBack)  {  gvProductos.DataSource = ObtenerProductos();  gvProductos.DataBind();  // Inicializar el carrito en sesión si no existe  if (Session["Carrito"] == null)  Session["Carrito"] = new List<ProductoHardcodeado>();  ActualizarCarrito();  }  }  public class ProductoHardcodeado  {  public int Id { get; set; }  public string Nombre { get; set; }  public int Precio { get; set; }  public string Descripcion { get; set; }  }  private List<ProductoHardcodeado> ObtenerProductos()  {  return new List<ProductoHardcodeado>  {  new ProductoHardcodeado { Id = 1, Nombre = "Equipo1", Precio = 100, Descripcion = "Equipo con 1 controlador de agua y 1 sensor de humedad" },  new ProductoHardcodeado { Id = 2, Nombre = "Equipo2", Precio = 150, Descripcion = "Equipo con 2 controladores de agua y 1 sensor de humedad" },  new ProductoHardcodeado { Id = 3, Nombre = "Equipo3", Precio = 200, Descripcion = "Equipo con 3 controladores de agua y 2 sensores de humedad" }  };  }  // Método para agregar productos al carrito  protected void gvProductos\_RowCommand(object sender, System.Web.UI.WebControls.GridViewCommandEventArgs e)  {  if (e.CommandName == "Agregar")  {  int index = Convert.ToInt32(e.CommandArgument);  int idProducto = index + 1;  ProductoHardcodeado producto = ObtenerProductos().FirstOrDefault(p => p.Id == idProducto);  if (producto != null)  {  List<ProductoHardcodeado> carrito = (List<ProductoHardcodeado>)Session["Carrito"];  carrito.Add(producto);  Session["Carrito"] = carrito;  ActualizarCarrito();  }  }  }  // Método para eliminar productos del carrito  protected void gvCarrito\_RowCommand(object sender, System.Web.UI.WebControls.GridViewCommandEventArgs e)  {  if (e.CommandName == "Eliminar")  {  int index = Convert.ToInt32(e.CommandArgument);  List<ProductoHardcodeado> carrito = (List<ProductoHardcodeado>)Session["Carrito"];  carrito.RemoveAt(index);  Session["Carrito"] = carrito;  ActualizarCarrito();  }  }  // Método para actualizar el GridView del carrito y el total  private void ActualizarCarrito()  {  List<ProductoHardcodeado> carrito = (List<ProductoHardcodeado>)Session["Carrito"];  gvCarrito.DataSource = carrito;  gvCarrito.DataBind();  int total = carrito.Sum(p => p.Precio);  lblTotal.Text = $"Total: {total:C}";  }  protected void Button1\_Click(object sender, EventArgs e)  {  List<ProductoHardcodeado> carrito = (List<ProductoHardcodeado>)Session["Carrito"];  if (carrito.Count == 0)  {  ShowMessage("Ingrese productos al carrito para pagar");  }  else  {  Response.Redirect("Pago.aspx");  }  }  //funcion para mostrar mensajes  private void ShowMessage(string message)  {  string script = $"alert('{message}');";  ScriptManager.RegisterStartupScript(this, GetType(), "showalert", script, true);  }  protected void Button2\_Click(object sender, EventArgs e)  {  Response.Redirect("User.aspx");  }  } |

|  |
| --- |
| EstadisticasVenta.aspx.cs |
| public partial class EstadisticaVenta : System.Web.UI.Page  {  //path donde se guarda el xml  private const string xmlAbsolutePath = @"C:\Users\lucas\Desktop\GIT\ventas.xml";  //web service para sacar estadisticas  WebServiceEstadisticoVenta estadisticoVenta = new WebServiceEstadisticoVenta();  protected void Page\_Load(object sender, EventArgs e)  {  //llamada a cargar ventas  if (!IsPostBack)  {  CargarVentas();  }  }  //funcion para levantar el xml y cargar las ventas en el controlador  private void CargarVentas()  {  try  {  //validacion de que exista el archivo  if (!File.Exists(xmlAbsolutePath))  {  lblTotal.Text = "El archivo XML no se encuentra en la ruta especificada.";  return;  }  //abriendo archivo y separando los id  XDocument xmlDoc = XDocument.Load(xmlAbsolutePath);  var ventas = from venta in xmlDoc.Descendants("Venta")  select new  {  Id = (int)venta.Element("Id")  };  ddlVentas.DataSource = ventas;  ddlVentas.DataTextField = "Id";  ddlVentas.DataValueField = "Id";  ddlVentas.DataBind();  }  catch (Exception ex)  {  lblTotal.Text = $"Error: {ex.Message}";  }  }  //boton para mostrar productos, seleccionando un id  protected void btnMostrarProductos\_Click(object sender, EventArgs e)  {  try  {  if (!File.Exists(xmlAbsolutePath))  {  lblTotal.Text = "El archivo XML no se encuentra en la ruta especificada.";  return;  }  XDocument xmlDoc = XDocument.Load(xmlAbsolutePath);  int idVentaSeleccionada = int.Parse(ddlVentas.SelectedValue);  var venta = xmlDoc.Descendants("Venta")  .Where(v => (int)v.Element("Id") == idVentaSeleccionada)  .FirstOrDefault();  if (venta != null)  {  var productos = venta.Element("Productos").Elements("Producto")  .Select(p => new  {  IdProducto = (int)p.Element("IdProducto"),  Nombre = (string)p.Element("Nombre"),  Precio = (decimal)p.Element("Precio")  }).ToList();  // Llenar el GridView con los productos  gvProductos.DataSource = productos;  gvProductos.DataBind();  // Mostrar el total de la venta  decimal precioTotal = (decimal)venta.Element("PrecioTotal");  lblTotal.Text = $"<b>Total de la venta:</b> ${precioTotal}";  }  }  catch (Exception ex)  {  lblTotal.Text = $"Error: {ex.Message}";  }  }  protected void Button1\_Click(object sender, EventArgs e)  {  //logica para mostrar al mejor cliente consumiendo el webservice  ClienteGasto clienteGasto = new ClienteGasto();  clienteGasto = estadisticoVenta.ObtenerClienteQueMasGasto(xmlAbsolutePath);  Label1.Text = "El mejor cliente es: " + clienteGasto.IdUsuario + " habiendo gastado: " + clienteGasto.TotalGastado.ToString();  }  protected void Button2\_Click(object sender, EventArgs e)  {  //redireccion a pantalla de admin  Response.Redirect("admin.aspx");  }  } |

|  |
| --- |
| Pago.aspx.cs |
| public partial class Pago : System.Web.UI.Page  {  List<ProductoHardcodeado> productos;  WebServicePago webservicePago = new WebServicePago();  BLLConexion conexion = new BLLConexion();  protected void Page\_Load(object sender, EventArgs e)  {  // instancia de usuario a mostrar  Label1.Text = BLLSesionManager.GetInstance.Usuario.Usu;  if (!IsPostBack)  {  // Cargar productos seleccionados en el GridView  productos = (List<ProductoHardcodeado>)Session["Carrito"];  gvProductosSeleccionados.DataSource = productos;  gvProductosSeleccionados.DataBind();  }  }    protected void btnConfirmarPago\_Click(object sender, EventArgs e)  {  List<ProductoHardcodeado> carrito = (List<ProductoHardcodeado>)Session["Carrito"];  int total = carrito.Sum(p => p.Precio);  if (ValidarFormulario())  {  if (webservicePago.ValidarPago(txtNombreTarjeta.Text, txtNumeroTarjeta.Text, txtFechaExpiracion.Text, txtCVV.Text, total))  {  // Aquí podrías procesar el pago o llamar a un servicio de pago externo  lblMensaje.Text = "Pago realizado con éxito.";  int idVenta = conexion.InsertarVenta(BLLSesionManager.GetInstance.Usuario.Id, total);  foreach (var item in carrito)  {  conexion.InsertarVentaxProducto(idVenta, item.Id, item.Precio);  conexion.InsertarUsuarioxProducto(BLLSesionManager.GetInstance.Usuario.Id, item.Id);  }  lblMensaje.ForeColor = System.Drawing.Color.Green;  string script = "window.onload = function(){ alert('Pago realizado con éxito.'); window.location.href = 'User.aspx'; }";  ClientScript.RegisterStartupScript(this.GetType(), "alert", script, true);  }  else  {  lblMensaje.Text = "Pago rechazado, valide los datos.";  }  }  else  {  lblMensaje.Text = "Por favor, completa todos los campos correctamente.";  }  }  // Validar que los campos del formulario estén completos  private bool ValidarFormulario()  {  if (string.IsNullOrEmpty(txtNombreTarjeta.Text) ||  string.IsNullOrEmpty(txtNumeroTarjeta.Text) ||  string.IsNullOrEmpty(txtFechaExpiracion.Text) ||  string.IsNullOrEmpty(txtCVV.Text))  {  return false;  }  // Validar que el número de tarjeta y CVV contengan solo números  if (!long.TryParse(txtNumeroTarjeta.Text, out \_) ||  !int.TryParse(txtCVV.Text, out \_))  {  return false;  }  return true;  }  protected void Button1\_Click(object sender, EventArgs e)  {  Response.Redirect("User.aspx");  }  } |

|  |
| --- |
| PantallaBitacora.aspx.cs |
| public partial class PantallaBitacora : System.Web.UI.Page  {  BLLConexion conexion = new BLLConexion();  protected void Page\_Load(object sender, EventArgs e)  {  //logica para que no se pueda acceder a la bitacora mediante el link e mensajes de aviso  //https://localhost:44325/PantallaBitacora  if(BLLSesionManager.GetInstance == null)  {  string script = "window.onload = function(){ alert('Estas intentando entrar en las bitacoras ilegalmente'); window.location.href = 'Principal.aspx'; }";  ClientScript.RegisterStartupScript(this.GetType(), "alert", script, true);  }  else  {  //logica para no acceder si no es admin  if (BLLSesionManager.GetInstance.Usuario.Roll != 1)  {  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "el usuario intento entrar ilegalmente a las bitacoras");  BLL.BLLSesionManager.logaut();  string script = "window.onload = function(){ alert('Estas intentando entrar en las bitacoras ilegalmente, se te cerrara la sesion y queda guardado para un futuro analisis.'); window.location.href = 'Principal.aspx'; }";  ClientScript.RegisterStartupScript(this.GetType(), "alert", script, true);  }  else  {  //llenado de datos de bitacora e sesion, aca se logro entrar  Label1.Text = (BLL.BLLSesionManager.GetInstance).Usuario.Usu;  Label2.Text = "WebMaster";  List<Bitacora> bitacora = new List<Bitacora>();  bitacora = conexion.listarBitacora();  GridView1.DataSource = bitacora;  GridView1.DataBind();  }  }  }  //funcion para mostrar cartel  private void ShowMessage(string message)  {  string script = $"alert('{message}');";  ScriptManager.RegisterStartupScript(this, GetType(), "showalert", script, true);  }  protected void Button1\_Click(object sender, EventArgs e)  {  //logica de desconeccion con log  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "deslogeo de sesion admin");  BLL.BLLSesionManager.logaut();  string script = "window.onload = function(){ alert('Sesión cerrada correctamente'); window.location.href = 'Principal.aspx'; }";  ClientScript.RegisterStartupScript(this.GetType(), "alert", script, true);;  }  protected void Button2\_Click(object sender, EventArgs e)  {  //logica para volver atras con log  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "vuelta a la pantalla de admin");  Response.Redirect("admin.aspx");  }  } |

|  |
| --- |
| PantallaUsuario.aspx.cs |
| public partial class PantallaUsuarios : System.Web.UI.Page  {  BLLConexion conexion = new BLLConexion();  protected void Page\_Load(object sender, EventArgs e)  {  //seteo de variables de la sesion, busqueda de los usuarios en base y seteo en vista  Label1.Text = (BLL.BLLSesionManager.GetInstance).Usuario.Usu;  Label2.Text = "WebMaster";  List<Usuario> usuarios = new List<Usuario>();  usuarios = conexion.ListarUsuario();  GridView1.DataSource = usuarios;  GridView1.DataBind();  }  protected void Button1\_Click(object sender, EventArgs e)  {  //desconexion e llenado del log  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "deslogeo de sesion admin");  BLL.BLLSesionManager.logaut();  string script = "window.onload = function(){ alert('Sesión cerrada correctamente'); window.location.href = 'Principal.aspx'; }";  ClientScript.RegisterStartupScript(this.GetType(), "alert", script, true);  }  protected void Button2\_Click(object sender, EventArgs e)  {  //vuelta a la pantalla de admin e ingreso a bitacora  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "vuelta a la pantalla de admin");  Response.Redirect("admin.aspx");  }  protected void Button3\_Click(object sender, EventArgs e)  {  //logica para quitar el bloqueo de un usuario con mensaje de si se pudo o no  Usuario usuarios = new Usuario();  usuarios.Id = int.Parse(txtNumeros.Text);  if (conexion.ReiniciarIntentosFallidos(usuarios))  {  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "se quito el bloqueo del usuario" + txtNumeros.Text);  ScriptManager.RegisterStartupScript(this, this.GetType(), "bloqueo quitado", "alert('bloqueo quitado.');", true);  List<Usuario> listusuarios = new List<Usuario>();  listusuarios = conexion.ListarUsuario();  GridView1.DataSource = listusuarios;  GridView1.DataBind();  }  else  {  ScriptManager.RegisterStartupScript(this, this.GetType(), "no se pudo quitar el bloqueo al usuario dado", "alert('no se pudo quitar el bloqueo al usuario dado.');", true);  }  }  } |

|  |
| --- |
| Principal.aspx.cs |
| public partial class Principal : System.Web.UI.Page  {  BLLConexion conexion = new BLLConexion();  protected void Page\_Load(object sender, EventArgs e)  {  }  //variable para bloquear muchos intentos fallidos de usuario incorrecto  static int bloqueoComun = 0;  protected void Button1\_Click(object sender, EventArgs e)  {    BLLEncriptado encriptado = new BLLEncriptado();  //traer usuario de base de datos  Usuario usu = conexion.ValidarUsuario(TextBox1.Text, TextBox2.Text);  //if de bloqueo por 3 veces de usuario incorrecto  if (bloqueoComun != 2)  {  //if por usuario no encontrado  if (usu is null)  {  //conexion.insertarBitacora(usu, "Usuario no encontrado");  ShowMessage("Usuario no encontrado");  bloqueoComun = bloqueoComun+1;  }  else  {  //if por usuario bloqueado  if (!usu.bloqueado)  {  //if validado de usuario  if (encriptado.VerifyPassword(TextBox2.Text,usu.Contrasena))  {  //logeo de usuario, reinicio de intentos e log de bitacora, tambien se recetea el bloqueo comun  BLLSesionManager.login(usu);  conexion.ReiniciarIntentosFallidos(usu);  bloqueoComun = 0;  switch (usu.Roll)  {  case 1:  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "logeo de sesion admin");  string script = "window.onload = function(){ alert('Sesión iniciada correctamente'); window.location.href = 'admin.aspx'; }";  ClientScript.RegisterStartupScript(this.GetType(), "alert", script, true);  break;  case 2:  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "logeo de sesion usuario");  string script2 = "window.onload = function(){ alert('Sesión iniciada correctamente'); window.location.href = 'User.aspx'; }";  ClientScript.RegisterStartupScript(this.GetType(), "alert", script2, true);  break;  default:  ShowMessage("Roll no valido");  break;  }  }  else  {  //log de bitacora de contraseña incorrecte e incremento de intentos fallidos, tambien se recetea el bloqueo comun    if(usu.intentos == 2)  {  conexion.insertarBitacora(usu, "contraseña incorrecta y bloqueamos la cuenta por 3 intentos fallidos");  ShowMessage("Contraseña incorrecta y bloqueamos la cuenta por 3 intentos fallidos");  }  else  {  conexion.insertarBitacora(usu, "contraseña incorrecta");  ShowMessage("Contraseña incorrecta");  }  conexion.IncrementarIntentosFallidos(usu);  bloqueoComun = 0;    }  }  else  {  //receteo de bloqueo comun e mensaje de usuario bloqueado  bloqueoComun = 0;  ShowMessage("el usuario esta bloqueado, contactese con un administrador");  }  }  }  else  {  //mensaje de intentos comunes fallidos y bloqueo de acciones  ShowMessage("intentaste ingresar un usuario invalido 3 veces segidas, se bloqueo la posibilidad de logear");  Button1.Enabled = false;  Button2.Enabled = false;  }  }  //funcion para mostrar menssajes  private void ShowMessage(string message)  {  string script = $"alert('{message}');";  ScriptManager.RegisterStartupScript(this, GetType(), "showalert", script, true);  }  protected void Button2\_Click(object sender, EventArgs e)  {  //resgistro de log de ingreso a pantalla de registro e redireccion a la misma  Usuario usuBitacora = new Usuario();  usuBitacora.Id = 8;  usuBitacora.Usu = "bitacora";  usuBitacora.Roll = 1;  conexion.insertarBitacora(usuBitacora, "ingreso a pantalla de registro");  //ShowMessage("Sesión cerrada correctamente");  Response.Redirect("Registro.aspx");  }  } |

|  |
| --- |
| Registro.aspx.cs |
| public partial class Registro : System.Web.UI.Page  {  BLLConexion conexion = new BLLConexion();  BLLEncriptado encriptado = new BLLEncriptado();  protected void Page\_Load(object sender, EventArgs e)  {  //if para ver si la pagina se recargo por una respuesta de datos  if (!IsPostBack)  {  // Obtener los valores del enum  var roles = Enum.GetValues(typeof(Rol));  // Asignar los valores al CheckBoxList  foreach (var role in roles)  {  DropDownList1.Items.Add(role.ToString());  }  }  }  protected void Button1\_Click(object sender, EventArgs e)  {  //registro de usuario tomando los datos de los controladores e insertado en bitacora  Usuario usuario = new Usuario();  usuario.Usu = TextBox1.Text;  usuario.Contrasena = encriptado.HashPassword(TextBox2.Text);  usuario.Id = 8;  //switch para tener nombres envez de numeros  switch (DropDownList1.SelectedItem.ToString())  {  case "admin":  usuario.Roll = 1;  break;  case "usuario":  usuario.Roll = 2;  break;  default:  break;  }  conexion.insertarUsuario(usuario);  conexion.insertarBitacora(usuario, "se registro el usuario " + usuario.Usu);  //redireccion a principal despues de registrar  Response.Redirect("Principal.aspx");  }  protected void Button2\_Click1(object sender, EventArgs e)  {  //vuelta a principal sin registrar un usuario, genera un log en la bitacora  Usuario usuBitacora = new Usuario();  usuBitacora.Id = 8;  usuBitacora.Usu = "bitacora";  usuBitacora.Roll = 1;  conexion.insertarBitacora(usuBitacora, "vuelta de pantalla de registro sin registrar");  Response.Redirect("Principal.aspx");  }  } |

|  |
| --- |
| User.aspx.cs |
| public partial class User : System.Web.UI.Page  {  protected void Page\_Load(object sender, EventArgs e)  {  // if para verificar que sea la primera ves que se abre esta pagina  if (!IsPostBack)  {  // instancia de usuario a mostrar  usu.Text = BLLSesionManager.GetInstance.Usuario.Usu;  }  // codigo para mostrar la lista de productos del usuario  GridView1.DataSource = conexion.listarProductosUsuario(BLLSesionManager.GetInstance.Usuario.Id);  GridView1.DataBind();  }  BLLConexion conexion = new BLLConexion();  protected void Button1\_Click(object sender, EventArgs e)  {  //Deslogeo con guardado de bitacora y muestra de cartel de deslogeo satisfactorio  conexion.insertarBitacora((BLL.BLLSesionManager.GetInstance).Usuario, "deslogeo de sesion usuario");  BLL.BLLSesionManager.logaut();  string script = "window.onload = function(){ alert('Sesión cerrada correctamente'); window.location.href = 'Principal.aspx'; }";  ClientScript.RegisterStartupScript(this.GetType(), "alert", script, true);  }  protected void Button2\_Click(object sender, EventArgs e)  {  //logica para ir al carrito de compras  Response.Redirect("CarritoCompra.aspx");  }  } |

## WebServices

|  |
| --- |
| WebServiceEstadisticosVenta.asmx |
| public class WebServiceEstadisticoVenta : System.Web.Services.WebService  {  [WebMethod]  public string HelloWorld()  {  return "Hola a todos";  }  public ClienteGasto ObtenerClienteQueMasGasto(string filePath)  {  try  {  // Cargar el documento XML  XDocument xmlDoc = XDocument.Load(filePath);  // Crear un diccionario para almacenar el total gastado por cada cliente  Dictionary<int, decimal> gastosPorCliente = new Dictionary<int, decimal>();  // Iterar sobre cada venta en el XML  foreach (var venta in xmlDoc.Descendants("Venta"))  {  int idUsuario = (int)venta.Element("IdUsuario");  decimal precioTotal = (decimal)venta.Element("PrecioTotal");  // Sumar el gasto al cliente correspondiente  if (gastosPorCliente.ContainsKey(idUsuario))  {  gastosPorCliente[idUsuario] += precioTotal;  }  else  {  gastosPorCliente[idUsuario] = precioTotal;  }  }  // Determinar el cliente que más ha gastado  var clienteConMasGasto = gastosPorCliente  .OrderByDescending(x => x.Value)  .FirstOrDefault();  // Devolver el resultado  return new ClienteGasto  {  IdUsuario = clienteConMasGasto.Key,  TotalGastado = clienteConMasGasto.Value  };  }  catch (Exception e)  {  // Manejo de errores  Console.WriteLine("Error al procesar el XML: " + e.Message);  return null;  }  }  } |

|  |
| --- |
| WebServicePago.asmx |
| public class WebServicePago : System.Web.Services.WebService  {  [WebMethod]  public string HelloWorld()  {  return "Hola a todos";  }  public bool ValidarPago(string nombreTarjeta, string numeroTarjeta, string fechaTarjeta, string CVV, int precio)  {  if (string.IsNullOrEmpty(nombreTarjeta) ||  string.IsNullOrEmpty(numeroTarjeta) ||  string.IsNullOrEmpty(fechaTarjeta) ||  string.IsNullOrEmpty(CVV))  {  return false;  }  // Validar que el número de tarjeta y CVV contengan solo números  if (!long.TryParse(numeroTarjeta, out \_) ||  !int.TryParse(CVV, out \_))  {  return false;  }  if (!ValidarFechaExpiracion(fechaTarjeta))  {  return false;  }  return true;  }  protected bool ValidarFechaExpiracion(string fecha)  {  string[] partes = fecha.Split('/');  // Validar formato correcto (MM/YY)  if (partes.Length != 2 || !int.TryParse(partes[0], out int mes) || !int.TryParse(partes[1], out int anio))  {  return false ;  }  // Convertir año al formato completo (por ejemplo, "24" => 2024)  anio += 2000;  // Validar que el mes esté entre 1 y 12  if (mes < 1 || mes > 12)  {  return false;  }  // Crear la fecha del último día del mes de vencimiento  DateTime fechaExpiracion = new DateTime(anio, mes, DateTime.DaysInMonth(anio, mes));  // Comparar con la fecha actual (fin del día)  if (fechaExpiracion < DateTime.Now)  {  return false;  }  else  {  return true;  }  }  } |

## BE

|  |
| --- |
| Bitacora.cs |
| public class Bitacora  {  private int id;  public int ID  {  get { return id; }  set { id = value; }  }  private int idUsuario;  public int IdUsuario  {  get { return idUsuario; }  set { idUsuario = value; }  }  private DateTime fecha;  public DateTime Fecha  {  get { return fecha; }  set { fecha = value; }  }  private string mensaje;  public string Mensaje  {  get { return mensaje; }  set { mensaje = value; }  }  private string corrupta;  public string Corrupta  {  get { return corrupta; }  set { corrupta = value; }  }  } |

|  |
| --- |
| ClienteGasto.cs |
| public class ClienteGasto  {  //clase para mostrar estadisticas de gastos del cliente  public int IdUsuario { get; set; }  public decimal TotalGastado { get; set; }  } |

|  |
| --- |
| Producto.cs |
| public class Producto  {  private int id;  public int ID  {  get { return id; }  set { id = value; }  }  private int idProducto;  public int IdProducto  {  get { return idProducto; }  set { idProducto = value; }  }  private string nombre;  public string Nombre  {  get { return nombre; }  set { nombre = value; }  }  private int precio;  public int Precio  {  get { return precio; }  set { precio = value; }  }  private string descripcion;  public string Descripcion  {  get { return descripcion; }  set { descripcion = value; }  }  private bool estado;  public bool Estado  {  get { return estado; }  set { estado = value; }  }  } |

|  |
| --- |
| Rol.cs |
| public enum Rol  {  admin,  usuario  } |

|  |
| --- |
| Usuario.cs |
| public class Usuario  {  private string usuario;  public string Usu  {  get { return usuario; }  set { usuario = value; }  }  private string contrasena;  public string Contrasena  {  get { return contrasena; }  set { contrasena = value; }  }  private int roll;  public int Roll  {  get { return roll; }  set { roll = value; }  }  private int id;  public int Id  {  get { return id; }  set { id = value; }  }  private string corrupto;  public string Corrupto  {  get { return corrupto; }  set { corrupto = value; }  }  private bool Bloqueado;  public bool bloqueado  {  get { return Bloqueado; }  set { Bloqueado = value; }  }  private int Intentos;  public int intentos  {  get { return Intentos; }  set { Intentos = value; }  }  } |

|  |
| --- |
| Venta.cs |
| public class Venta  {  public int Id { get; set; }  public int IdUsuario { get; set; }  public int PrecioTotal { get; set; }  public List<Producto> Productos { get; set; }  public Venta()  {  Productos = new List<Producto>();  }  } |

## BLL

|  |
| --- |
| BLLConexion.cs |
| public class BLLConexion  {  Conexion conexion = new Conexion();  public void guardarUsuario(string usuario, string contrasena)  {  conexion.guardarUsuario(usuario, contrasena);  }  public Usuario ValidarUsuario(string nombre, string contrasena)  {  BLLEncriptado encriptado = new BLLEncriptado();  return conexion.validadUsuario(nombre, encriptado.HashPassword(contrasena));  }  public void insertarBitacora(Usuario usu, string mensaje)  {  conexion.insertarBitacora(usu, mensaje);  }  public List<Bitacora> listarBitacora()  {  return conexion.listarBitacora();  }  public List<Usuario> ListarUsuario()  {  return conexion.ListarUsuarios();  }  public void insertarUsuario(Usuario usu)  {  conexion.insertarUsuario(usu);  }  public bool generarBackup()  {  return conexion.generarBackup();  }  public bool restaurarBase()  {  return conexion.restaurarBase();  }  public void IncrementarIntentosFallidos(Usuario usu)  {  conexion.IncrementarIntentosFallidos(usu);  }  public bool ReiniciarIntentosFallidos(Usuario usu)  {  return conexion.ReiniciarIntentosFallidos(usu);  }  public List<Producto> listarProductosUsuario(int usuario)  {  return conexion.listarProductosUsuario(usuario);  }  public void InsertarUsuarioxProducto(int usuario, int producto)  {  conexion.InsertarUsuarioxProducto(usuario, producto);  }  public int InsertarVenta(int usuario, int precio)  {  return conexion.InsertarVenta(usuario, precio);  }  public void InsertarVentaxProducto(int venta, int producto, int precio)  {  conexion.InsertarVentaxProducto(venta, producto, precio);  }  public List<Venta> ObtenerVentasCompleto()  {  return conexion.ObtenerVentasCompleto();  }  } |

|  |
| --- |
| BLLEncriptado.cs |
| public class BLLEncriptado  {  public string HashPassword(string password)  {  using (SHA256 sha256Hash = SHA256.Create())  {  // Convertir la cadena de texto en un array de bytes  byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(password));  // Convertir los bytes en una cadena hexadecimal  StringBuilder builder = new StringBuilder();  for (int i = 0; i < bytes.Length; i++)  {  builder.Append(bytes[i].ToString("x2"));  }  return builder.ToString();  }  }  public bool VerifyPassword(string enteredPassword, string storedPasswordHash)  {  string enteredPasswordHash = HashPassword(enteredPassword);  return string.Equals(enteredPasswordHash, storedPasswordHash, StringComparison.OrdinalIgnoreCase);  }  } |

|  |
| --- |
| BLLGestorXML.cs |
| public void CrearXmlDeVentas(List<Venta> ventas)  {  try  {  // Crear el documento XML  XDocument xmlDoc = new XDocument(  new XElement("Ventas", // Raíz del documento  // Agregar cada venta  new List<XElement>(ventas.ConvertAll(venta =>  new XElement("Venta",  new XElement("Id", venta.Id),  new XElement("IdUsuario", venta.IdUsuario),  new XElement("PrecioTotal", venta.PrecioTotal),  new XElement("Productos",  // Agregar cada producto  new List<XElement>(venta.Productos.ConvertAll(producto =>  new XElement("Producto",  new XElement("IdProducto", producto.IdProducto),  new XElement("Nombre", producto.Nombre),  new XElement("Precio", producto.Precio)  ))  )  )  )  ))  )  );  // Guardar el documento XML en el archivo  xmlDoc.Save(@"C:\Users\lucas\Desktop\GIT\ventas.xml");    }  catch (Exception e)  {  // Manejo de errores  Console.WriteLine("Error al crear el XML: " + e.Message);  }  } |

|  |
| --- |
| BLLSesionManager.cs |
| public class BLLSesionManager  {  private Usuario usuario;  public Usuario Usuario  {  get { return usuario; }  set { usuario = value; }  }  private static BLLSesionManager \_session;  public static BLLSesionManager GetInstance  {  get  {  return \_session;  }  }  public static void login(Usuario usuario)  {  if (\_session == null)  {  \_session = new BLLSesionManager();  \_session.usuario = usuario;  }  else  {  throw new Exception("Sesion no iniciada");  }  }  public static void logaut()  {  if (\_session != null)  {  \_session = null;  }  else  {  throw new Exception("Sesion no iniciada");  }  }  } |

## DAL

|  |
| --- |
| Conexión.cs |
| public class Conexion  {  //conexionstring de la base principal  private string GetConnectionString()  {  var cs = new SqlConnectionStringBuilder();  cs.IntegratedSecurity = true;  cs.DataSource = @"LACUCA\SQLEXPRESS";  cs.InitialCatalog = "web";  return cs.ConnectionString;  }  //conexion string de la base master  private string GetConnectionStringMaster()  {  var cs = new SqlConnectionStringBuilder();  cs.IntegratedSecurity = true;  cs.DataSource = @"LACUCA\SQLEXPRESS";  cs.InitialCatalog = "master";  return cs.ConnectionString;  }  //guardado de usuario con store procedure  public void guardarUsuario(string nom, string con)  {  try  {  var cnn = new SqlConnection(GetConnectionString());  cnn.Open();  var cmd = new SqlCommand("guardarUsuario");  cmd.Connection = cnn;  cmd.CommandType = System.Data.CommandType.StoredProcedure;  cmd.Parameters.Add(new SqlParameter("nombre", nom));  cmd.Parameters.Add(new SqlParameter("contrasena", con));  var id = cmd.ExecuteScalar();  }  catch (Exception e)  {  throw e;  }  }  //logica para traer un usuario  public Usuario validadUsuario(string nom, string con)  {  try  {  var cnn = new SqlConnection(GetConnectionString());  cnn.Open();  var cmd = new SqlCommand("BuscarUsuario");  cmd.Connection = cnn;  cmd.CommandType = System.Data.CommandType.StoredProcedure;  cmd.Parameters.Add(new SqlParameter("nombre", nom));  using (SqlDataReader reader = cmd.ExecuteReader())  {  if (reader.Read())  {  string nombreUsuario = reader.GetString(0);  string contrasena = reader.GetString(1);  int roll = reader.GetInt32(2);  int id = reader.GetInt32(3);  bool bloqueado = reader.GetBoolean(4);  int intentos = reader.GetInt32(5);  //if (nombreUsuario == nom && contrasena == con)  //{  Usuario usu = new Usuario();  usu.Usu = nombreUsuario;  usu.Contrasena = contrasena;  usu.Roll = roll;  usu.Id = id;  usu.bloqueado = bloqueado;  usu.intentos = intentos;  return usu;  //}  //else  //{  // return null;  //}  }  else  {  Console.WriteLine("Usuario no encontrado.");  return null;  }  }  }  catch (Exception e)  {  throw e;  }  }  //logica para insertar en bitacora con store procedure  public void insertarBitacora(Usuario usu, string mensaje)  {  try  {  var cnn = new SqlConnection(GetConnectionString());  cnn.Open();  var cmd = new SqlCommand("InsertarBitacora");  cmd.Connection = cnn;  cmd.CommandType = System.Data.CommandType.StoredProcedure;  cmd.Parameters.Add(new SqlParameter("idUsuario", usu.Id));  cmd.Parameters.Add(new SqlParameter("mensaje", mensaje));  cmd.Parameters.Add(new SqlParameter("fecha", DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss")));  var id = cmd.ExecuteScalar();  }  catch (Exception e)  {  throw e;  }  }  //logica pra insertar bitacora con store procedure  public List<Bitacora> listarBitacora()  {  List<Bitacora> listBit = new List<Bitacora>();  try  {  var cnn = new SqlConnection(GetConnectionString());  cnn.Open();  var cmd = new SqlCommand("ListarBitacora");  cmd.Connection = cnn;  cmd.CommandType = System.Data.CommandType.StoredProcedure;    using (SqlDataReader reader = cmd.ExecuteReader())  {  while (reader.Read())  {  Bitacora bitacora = new Bitacora();  bitacora.ID = reader.GetInt32(0);  bitacora.IdUsuario = reader.GetInt32(1);  bitacora.Fecha = reader.GetDateTime(2);  bitacora.Mensaje = reader.GetString(3);  bitacora.Corrupta = reader.GetString(5);  listBit.Add(bitacora);  }  }  }  catch (Exception e)  {  throw e;  }  return listBit;  }  //logica para listar todos los usuarios  public List<Usuario> ListarUsuarios()  {  List<Usuario> listBit = new List<Usuario>();  try  {  var cnn = new SqlConnection(GetConnectionString());  cnn.Open();  var cmd = new SqlCommand("ListarUsuarios");  cmd.Connection = cnn;  cmd.CommandType = System.Data.CommandType.StoredProcedure;  using (SqlDataReader reader = cmd.ExecuteReader())  {  while (reader.Read())  {  Usuario usu = new Usuario();  usu.Usu = reader.GetString(0);  usu.Contrasena = reader.GetString(1);  usu.Roll = reader.GetInt32(2);  usu.Id = reader.GetInt32(3);  usu.Corrupto = reader.GetString(5);  usu.bloqueado = reader.GetBoolean(6);  listBit.Add(usu);  }  }  }  catch (Exception e)  {  throw e;  }  return listBit;  }  //logica para insertar un usuario con store procedure  public void insertarUsuario(Usuario usu)  {  try  {  var cnn = new SqlConnection(GetConnectionString());  cnn.Open();  var cmd = new SqlCommand("RegistrarUsuario");  cmd.Connection = cnn;  cmd.CommandType = System.Data.CommandType.StoredProcedure;  cmd.Parameters.Add(new SqlParameter("nombre", usu.Usu));  cmd.Parameters.Add(new SqlParameter("contraseña", usu.Contrasena));  cmd.Parameters.Add(new SqlParameter("roll", usu.Roll));  var id = cmd.ExecuteScalar();  }  catch (Exception e)  {  throw e;  }  }  //logica para generar el backup, asiganamos en que ruta se guarda el backup  public bool generarBackup()  {  string backupPath = @"C:\Users\lucas\Desktop\GIT\web.bak"; //direccion de guardado de backup  string connectionString = GetConnectionString();  using (SqlConnection connection = new SqlConnection(connectionString))  {  using (SqlCommand command = new SqlCommand("BackupDatabase", connection))  {  command.CommandType = System.Data.CommandType.StoredProcedure;  command.Parameters.AddWithValue("@backupPath", backupPath);  // Aumentar el tiempo de espera del comando a 600 segundos (10 minutos) ya que puede tirar time out  command.CommandTimeout = 600;  try  {  connection.Open();  command.ExecuteNonQuery();  return true;  }  catch (Exception ex)  {    return false;  }  }  }  }  //logica para restar la base  public bool restaurarBase()  {  string backupPath = @"C:\Users\lucas\Desktop\GIT\web.bak"; // direccion donde esta el backup  string connectionString = GetConnectionStringMaster();  using (SqlConnection connection = new SqlConnection(connectionString))  {  using (SqlCommand command = new SqlCommand("RestoreDatabase", connection))  {  command.CommandType = System.Data.CommandType.StoredProcedure;  command.Parameters.AddWithValue("@backupPath", backupPath);  // Aumentar el tiempo de espera del comando a 600 segundos (10 minutos) por posible time out  command.CommandTimeout = 600;  try  {  connection.Open();  command.ExecuteNonQuery();  return true;  }  catch (Exception ex)  {  return false;  }  }  }  }  //logica para incrementar los intentos fallidos de un usuario  public void IncrementarIntentosFallidos(Usuario usu)  {  try  {  var cnn = new SqlConnection(GetConnectionString());  cnn.Open();  var cmd = new SqlCommand("IncrementarIntentosFallidos");  cmd.Connection = cnn;  cmd.CommandType = System.Data.CommandType.StoredProcedure;  cmd.Parameters.Add(new SqlParameter("usuarioId", usu.Id));  var id = cmd.ExecuteScalar();  }  catch (Exception e)  {  throw e;  }  }  //logica para reiniciar los fallos e intentos de un usuario como el bloqueo  public bool ReiniciarIntentosFallidos(Usuario usu)  {  bool success = false;  try  {  string connectionString = GetConnectionString(); // Obtener cadena de conexión  using (SqlConnection cnn = new SqlConnection(connectionString))  {  cnn.Open();  SqlCommand cmd = new SqlCommand("ReiniciarIntentosFallidos", cnn);  cmd.CommandType = CommandType.StoredProcedure;  // Parámetro de entrada  cmd.Parameters.AddWithValue("@usuarioId", usu.Id);  // Parámetro de salida  SqlParameter successParam = new SqlParameter("@success", SqlDbType.Bit);  successParam.Direction = ParameterDirection.Output;  cmd.Parameters.Add(successParam);  // Ejecutar el comando  cmd.ExecuteNonQuery();  // Obtener el valor de retorno  success = (bool)successParam.Value;  }  }  catch (Exception e)  {  throw e;  }  return success;  }  //logica para mostrar los productos de los usuarios  public List<Producto> listarProductosUsuario(int usuario)  {  List<Producto> listBit = new List<Producto>();  try  {  var cnn = new SqlConnection(GetConnectionString());  cnn.Open();  var cmd = new SqlCommand("ObtenerProductosPorUsuario");  cmd.Connection = cnn;  cmd.CommandType = System.Data.CommandType.StoredProcedure;  cmd.Parameters.Add(new SqlParameter("IdUsuario", usuario));  using (SqlDataReader reader = cmd.ExecuteReader())  {  while (reader.Read())  {  Producto producto = new Producto();  producto.ID = reader.GetInt32(0);  producto.IdProducto = reader.GetInt32(1);  producto.Nombre = reader.GetString(2);  producto.Precio = reader.GetInt32(3);  producto.Descripcion = reader.GetString(4);  producto.Estado = reader.GetBoolean(5);  listBit.Add(producto);  }  }  }  catch (Exception e)  {  throw e;  }  return listBit;  }  //logica cargar productos por usuario  public void InsertarUsuarioxProducto(int usuario, int producto)  {  try  {  var cnn = new SqlConnection(GetConnectionString());  cnn.Open();  var cmd = new SqlCommand("sp\_InsertarUsuarioxProducto");  cmd.Connection = cnn;  cmd.CommandType = System.Data.CommandType.StoredProcedure;  cmd.Parameters.Add(new SqlParameter("IdUsuario", usuario));  cmd.Parameters.Add(new SqlParameter("IdProducto", producto));  cmd.Parameters.Add(new SqlParameter("Estado", true));  var id = cmd.ExecuteScalar();  }  catch (Exception e)  {  throw e;  }  }  public int InsertarVenta(int usuario, int precio)  {  try  {  using (var cnn = new SqlConnection(GetConnectionString()))  {  cnn.Open();  using (var cmd = new SqlCommand("sp\_InsertarVenta", cnn))  {  cmd.CommandType = System.Data.CommandType.StoredProcedure;  cmd.Parameters.Add(new SqlParameter("@IdUsuario", usuario));  cmd.Parameters.Add(new SqlParameter("@Precio", precio));  var outputIdParam = new SqlParameter("@NuevoIdVenta", SqlDbType.Int)  {  Direction = ParameterDirection.Output  };  cmd.Parameters.Add(outputIdParam);  cmd.ExecuteNonQuery();  return (int)outputIdParam.Value;  }  }  }  catch (Exception e)  {  throw new Exception("Error al insertar la venta", e);  }  }  public void InsertarVentaxProducto(int venta, int producto, int precio)  {  try  {  var cnn = new SqlConnection(GetConnectionString());  cnn.Open();  var cmd = new SqlCommand("sp\_InsertarVentaxProducto");  cmd.Connection = cnn;  cmd.CommandType = System.Data.CommandType.StoredProcedure;  cmd.Parameters.Add(new SqlParameter("IdVenta", venta));  cmd.Parameters.Add(new SqlParameter("IdProducto", producto));  cmd.Parameters.Add(new SqlParameter("Precio", precio));  var id = cmd.ExecuteScalar();  }  catch (Exception e)  {  throw e;  }  }  public List<Venta> ObtenerVentasCompleto()  {  var ventas = new List<Venta>();  try  {  using (var cnn = new SqlConnection(GetConnectionString()))  {  cnn.Open();  var cmd = new SqlCommand("sp\_ObtenerVentasCompleto", cnn);  cmd.CommandType = CommandType.StoredProcedure;  using (var reader = cmd.ExecuteReader())  {  Venta ventaActual = null;  while (reader.Read())  {  int idVenta = reader.GetInt32(0);  // Si cambiamos de venta, creamos una nueva instancia de Venta  if (ventaActual == null || ventaActual.Id != idVenta)  {  ventaActual = new Venta  {  Id = idVenta,  IdUsuario = reader.GetInt32(1),  PrecioTotal = reader.GetInt32(2)  };  ventas.Add(ventaActual);  }  // Agregamos el producto a la lista de productos de la venta actual  if (!reader.IsDBNull(3)) // Si existe un producto asociado  {  var producto = new Producto  {  IdProducto = reader.GetInt32(3),  Nombre = reader.GetString(4),  Precio = reader.GetInt32(5)  };  ventaActual.Productos.Add(producto);  }  }  }  }  }  catch (Exception e)  {  throw e;  }  return ventas;  }  } |

# Mapa de navegación de web máster

El web master inicia sesión como en la misma pantalla que todos:  
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Una ves dentro de la sesión tiene 5 opciones.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

con la opción bitácora entra en la pantalla de bitácora, acá puede ver todos los registros de la bitácora.

Tabla

Descripción generada automáticamente

En la opción usuario puede ver todos los registros de los usuarios y quitarle el bloqueo a los que estén bloqueados.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Con la opción Backup genera la copia de seguridad de la base de datos.

Y con la opción reutaurar base, toma el archivo generado y restaura la base de datos.

Por último con la opción ventas, entra en la pantalla de ventas donde se pueden los productos de las ventas y estadísticas.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

# Anexo:

## Políticas de programación

## Políticas de tratamiento de olvido de contraseña:

Cuando ocurre un intento de ingreso a una cuenta y se equivocan tres veces de manera consecutiva en ingresar el usuario y/o contraseña, se bloquea el acceso a esta cuenta.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para poder realizar una recuperación del acceso a esta cuenta, se podrá pedir la restauración de la contraseña a un administrador, para esto el usuario se deberá contacta con un administrador.

## Políticas de receteo de usuario

El usuario se contacta con el administrador, este mediante la pantalla de usuario le puede quitar el bloqueo de contraseña, para el cambio de contraseña lo va a hacer en la misma pantalla ingresando la contraseña nueva. Luego le notifica al usuario el reseteo o la contraseña nueva.

## Políticas de backup

El administrador tiene la posibilidad de hacer un backup de la base de datos con el botón “Backup” dentro de la pantalla de admin, este botón llama a un store procedure que genera el archivo de la base.

Para recuperar la base con el archivo guardado el admin tiene que usar el botón “Restaurar base” dentro de la pantalla de admin, este botón toma el archivo generado y lo restaura en la base mediante un store procedure.

## Políticas de tratamiento de pantalla de bitácora

No se puede ingresar a esta pantalla sin estar logeado como administrador.

Mensaje que se muestra cuando se intenta ingresar sin un usuario logueado.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Cuando se intente entrar con usuario no valido a la pantalla de bitácora la sesión se cerra y se subida un log de lo que pago.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

## Estrategia de programación en capas

Diagrama

Descripción generada automáticamente

Utilizamos una arquitectura de 4 capas con la posibilidad de aplicar a una quinta capa de servicio, se eligió esta metodología para poder separar las funcionalidades logias.

## Uso de Ado .NET desconectado

En la tecnología .NET existen dos modos para trabajar con la base de datos, nosotros elegimos el modo desconectado.

En el modo desconectado, la conexión se abre y se usa un SqlDataAdapter para llenar los datos, generalmente en un dataset, para trabajar posteriormente con esos datos en memoria. El SqlDataAdapterr maneja la conexión internamente y solo abre la conexión cuando es necesario para recuperar los datos y luego la cierra automáticamente.

Utilizamos este modo ya que no manejamos un gran volumen de información. Cada función abre una conexión y la mantiene abierta mientras se ejecuta la operación (lectura o escritura) en la base de datos. Los datos se acceden directamente desde la base de datos a través de SqlCommand y SqlDataReader o mediante la ejecución directa de procedimientos almacenados (store procedures).

## Uso de “Store Procedures”

Utilizamos diferente store procedures para aprovechar sus virtudes del punto de vista del rendimiento, seguridad y mantenimiento.

Los procedimientos almacenados se compilan y almacenan en la base de datos. Esto significa que el plan de ejecución se guarda y se reutiliza.

## Uso de “Web Service”

|  |  |  |
| --- | --- | --- |
| **Web service** | **Descripción** | **Quien lo utiliza** |
| **WebServicePago** | Este web service valida el pago de la compra, le enviamos los datos de la tarjeta y el monto a pagar y nos devuelve true o false según analice. | Pago.aspx |
| **WebServiceEstadisticoVenta** | Este web service analiza los datos de venta que le proporcionemos y nos devuelve el dato por el cual lo llamamos, en nuestro caso le damos los datos de las ventas y nos devuelve el cliente que mas recaudo con todas sus compras. | EstadisticaVenta.aspx |

## Uso de “XML”

|  |  |  |  |
| --- | --- | --- | --- |
| **XML** | **Que modela** | **Para que lo utilizamos** | **Quien lo usa** |
| Ventas.xml | Modela las ventas que se realizaron, tiene el id, el usuario que compro, el precio total y todos los datos de los productos como sus precios individuales. | Se utiliza para sacar las estadísticas de las ventas y también para poder visualizarlas en la aplicación. | EstadisticaVenta.aspx  WebServiceEstadisticoVenta.asmx |