Estándar de Programación Lenguaje ASP.NET

<BOLSA DE TRABAJO>

***Llantay Machaca Marjiory Grace***

***Soto Rodriguez Duanet***

Estándar de Programación Lenguaje ASP.NET

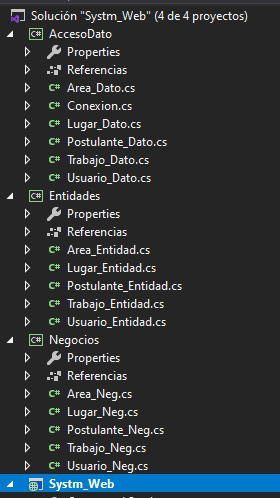
# Nombres de Ficheros

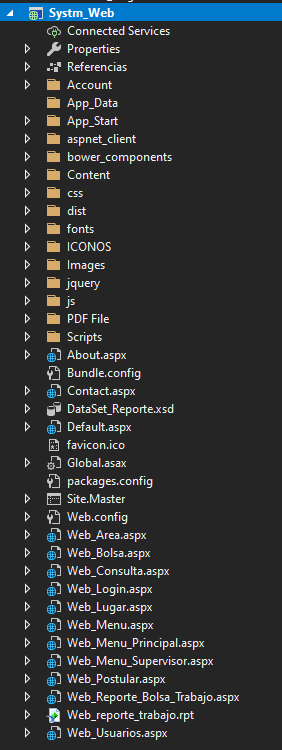
* Los nombres de archivos deben describir claramente su propósito.
* Para clases, el nombre del archivo debe coincidir con el nombre de la clase.
* Ejemplo: UsuarioNegocio.cs, Conexion.cs, Web\_Login.aspx.cs

# Extensiones de Ficheros

* **.aspx** para páginas web.
* **.cs** para archivos de lógica de negocio y datos.
* **.config** para configuración.
* **.xsd** para definiciones de esquemas de datos.

# Organización de los ficheros



****

## 

## 1. Organización de Using Statements

## 

**Orden de Import**

1. System namespaces (ordenados alfabéticamente)

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

1. Línea en blanco separadora
2. Namespaces personalizados

using Entidades;

1. **Estructura de Namespaces**

**Nomenclatura**

* Usar PascalCase
* Nombres descriptivos que indiquen la capa o funcionalidad

namespace AccesoDato

{

// Contenido

}

1. **Clases de Acceso a Datos**

**Nomenclatura**

* Sufijo descriptivo "\_Dato" para clases de acceso a datos
* PascalCase para nombres de clase

public class Area\_Dato

{

// Contenido

}

**Estructura**

* Declarar conexiones al inicio de la clase

private Conexion con = new Conexion();

1. **Métodos de Acceso a Datos**

**Nomenclatura**

* Nombres descriptivos que indiquen la operación
* Verbos como prefijo (Insert, Update, Delete, Get)

public void Insert(Area\_Entidad cliente\_entidad)

**Parámetros**

* Usar nombres descriptivos para parámetros
* Sufijo "\_entidad" para objetos de entidad

1. **Manejo de SQL**

**Comandos SQL**

* Usar SqlCommand para operaciones de base de datos
* Declarar como variables locales

SqlCommand cmd = new SqlCommand("insert into area values (@nombre)", con.con);

**Parámetros SQL**

* Usar siempre parámetros para prevenir SQL Injection
* Prefijo "@" para nombres de parámetros
* Especificar tipo y tamaño de datos

cmd.Parameters.Add("@nombre", SqlDbType.VarChar, 50).Value = cliente\_entidad.nombre;

1. **Manejo de Conexiones**

**Apertura y Cierre**

* Abrir conexión justo antes de ejecutar comando
* Cerrar conexión inmediatamente después

con.con.Open();

cmd.ExecuteNonQuery();

con.con.Close();

**7. Manejo de Excepciones**

**Try-Catch**

* Implementar try-catch en todos los métodos de acceso a datos
* Documentar excepciones esperadas

try

{

// Operaciones de base de datos

}

catch (Exception ex)

{

// Manejo de excepción

}

**8. Comentarios y Documentación**

**Métodos**

* Documentar propósito del método
* Describir parámetros y retorno

/// <summary>

/// Inserta un nuevo registro en la tabla area

/// </summary>

/// <param name="cliente\_entidad">Entidad con datos a insertar</param>

**9. Convenciones de Formato**

**Indentación**

* Usar 4 espacios para indentación
* Alinear llaves de apertura y cierre
* Línea en blanco entre métodos

**Espaciado**

* Espacio después de palabras clave (if, while, for)
* Espacio alrededor de operadores
* No espacios entre nombre de método y paréntesis

**10. Mejores Prácticas**

**Seguridad**

* Nunca exponer cadenas de conexión en código
* Usar siempre consultas parametrizadas
* Implementar manejo de errores robusto

**Rendimiento**

* Cerrar conexiones explícitamente
* Usar using statements para objetos disposables
* Optimizar consultas SQL