Prueba Practica .NET

**Instrucciones**: Realice la siguiente prueba generando los scripts / archivos que aplique para cada uno y súbalo a un **repositorio GIT** en cualquier servicio en la nube. El link deberá ser enviado al correo electrónico de donde recibió la prueba para poderlo descargar y revisar.

1. Escriba el código de una página ASPX que solicite el nombre del usuario, contraseña, y que valide que los campos no vayan vacíos antes de enviar la solicitud al servidor.

R// Revisar la carpeta Prueba JavaScript

1. Escriba un documento XML que permita intercambiar información de tarjetas de crédito entre dos sistemas distintos: nombre del sistema, localización, número de tarjeta, ID único de cliente, etc.
2. Con el siguiente modelo de base de datos:

Diagrama

Descripción generada automáticamente

Descripción de las Tablas:

Catálogos:

* PROYECTO: Listado de proyectos de tarjetas que existen (Ejemplo: Premia, Konmi, Yujule, etc.)
* PRODUCTO: listado de productos de tarjetas que hay. (ejemplo Premia Clásica, Premia Oro, Premia Platinum)
* TIPO: Tipo de mensaje (ejemplo de mensaje de texto, mail, mensaje en el estado de cuenta)
* TIPO\_INFORMACION: Tipo del mensaje. Ejemplo mensaje de bienvenida, mensaje de mora, mensaje de promoción

Otras:

* FORMATO\_MENSAJE: son los formatos de los mensajes existentes.
* MENSAJE: si el mensaje aplica a que proyecto y que producto

1. Escriba la consulta en SQL que devuelva el nombre del proyecto y sus productos correspondientes del proyecto premia cuyo código es 1

R//

SELECT P.NOMBRE, P1.DESCRIPCION

FROM PRODUCTO\_PROYECTO A

INNER JOIN PRODUCTO P1 ON A.PRODUCTO = P1.PRODUCTO

INNER JOIN PROYECTO P ON A.PROYECTO = P.PROYECTO

WHERE P.PROYECTO = 1

1. Escriba una consulta SQL que devuelva los distintos mensajes que hay, indicando a qué proyecto y producto pertenecen.

R//

SELECT A.ASUNTO, B.NOMBRE, C.DESCRIPCION

FROM FORMATO\_MENSAJE A

INNER JOIN MENSAJE D ON D.COD\_FORMATO = A.COD\_FORMATO

INNER JOIN PROYECTO B ON D.PROYECTO = B.PROYECTO

INNER JOIN PRODUCTO C ON D.PRODUCTO = C.PRODUCTO

WHERE B.PROYECTO = 1

1. Escriba una consulta SQL que devuelva los distintos mensajes que hay, indicando a qué proyecto y producto pertenecen. Pero si el mensaje está en todos los productos de un proyecto, en lugar de mostrar cada producto, debe mostrar el nombre del proyecto y un solo producto que diga “TODOS”.

R//

DECLARE @CantidadProductos int

SET @CantidadProductos = (SELECT COUNT (\*) FROM PRODUCTO)

SELECT IIF (COUNT(PP.PRODUCTO) = @CantidadProductos,

‘TODOS’, FM.ASUNTO) AS ‘MENSAJE’, PP.PROYECTO

FROM MENSAJE M

INNER JOIN PRODUCTO\_PROYECTO AS PP ON M.PRODUCTO = PP.PRODUCTO

AND M.PROYECTO = PP.PROYECTO

INNER JOIN PRODUCTO P ON PP.PRODUCTO = P.PRODUCTO

INNER JOIN PROYECTO PR ON PP.PROYECTO = PR.PROYECTO

INNER JOIN FORMATO\_MENSAJE FM ON M.COD\_FORMATO = FM.COD\_FORMATO

GROUP BY PP.PROYECTO, FM.ASUNTO

1. Escrba el código de una clase, en sintaxis C# que permita hacer conexiones a una base de datos cualquiera, y que devuelva en uno de sus métodos un Dataset al hacer un SELECT, y en otro de sus métodos el número de filas afectadas al ejecutar un INSERT, UPDATE o DELETE. También debe poder devolver el número de filas que devolvería un SELECT si se ejecutara.

R//

class Conexion

{

public string cadena = “Server-Server;DataBase=BD; User ID = sa; Password = 123;”;

//Insertar cliente

public string InsertarCliente(string nombre, string dirección)

{

object id;

SqlConnection con = new SqlConnection(cadena);

try

{

con.Open();

SqlCommand cmd = new SqlCommand(“Clientes.InsertarCliente”, con)

{

CommandType = CommandType.StoredProcedure

};

cmd.Parameters.AddWithValue(“@NombreCliente”, nombre.ToUpper());

cmd.Parameters.AddWithValue(“@DireccionCliente”, direccion.TuUpper());

var returnParameter = cmd.Parameters.Add(“@Existente”, SqlDbType.Int);

returnParameter.Direccion = ParameterDirection.ReturnValue;

cmd.ExecuteNonQuery();

id = returnParameter.Value;

con.Close();

cmd.Dispose();

return id.ToString();

}

catch(Exception ex)

{

return ex.Message;

}

Finally

{

con.Close();

}

}

//Actualizar cliente

public string UpdateCliente(int idCliente, string nombre, string direccion)

{

object id;

SqlConnection con = new SqlConnection(cadena);

try

{

con.Open();

SqlCommand cmd = new SqlCommand(“Clientes.ModificarCliente”, con)

{

CommandType = CommandType.StoredProcedure

};

cmd.Parameters.AddWithValue(“@IdCliente”, idCliente);

cmd.Parameters.AddWithValue(“@NombreCliente”, nombre.ToUpper());

cmd.Parameters.AddWithValue(“@DireccionCliente”, direccion.TuUpper());

var returnParameter = cmd.Parameters.Add(“@Existente”, SqlDbType.Int);

returnParameter.Direccion = ParameterDirection.ReturnValue;

cmd.ExecuteNonQuery();

id = returnParameter.Value;

con.Close();

cmd.Dispose();

return id.ToString();

}

catch(Exception ex)

{

return ex.Message;

}

Finally

{

con.Close();

}

}

//Listar clientes

public DataTable Listar\_Clientes()

{

SqlConnection con = new SqlConnection(cadena);

DataTable dt = new DataTable();

try

{

string sql = “Select \* from dbo.Cliente”;

SqlCommand comando = new SqlCommand(sql, con);

comando.CommandType = CommandType.Text;

SqlDataAdapter da = new SqlDataAdapter(comando);

da.Fill(dt);

return dt;

}

catch (Exception ex)

{

throw ex;

}

finally

{

con.Close();

}

}

}

1. Construya un modelo Entidad Relación que permita llevar el control de Canje de sus puntos acumulados por premios.
   1. El artículo debe tener código, nombre, tipo y subtipo.
   2. Se debe manejar la existencia de artículos por sucursal. Ejemplo, está la sucursal Promerica Majadas y Promerica Columbus y en cada sucursal existen artículos como DVD de Disney, lapiceros, mouse, etc. Y cada sucursal debe saber qué cantidad posee en inventario.
   3. Van a existir varias promociones para canjear y estas van a estar compuestas por varios artículos. Estas promociones poseen rangos de fecha de vigencia y en dichas promociones estará el costo de los puntos y el precio del artículo
   4. El canje debe tener ID del artículo, Cantidad Canjeada, Precio del artículo, la promoción y observaciones.

R// Revisar el SQL ‘Query Prueba Promerica Canje.sql’

Diagrama

Descripción generada automáticamente