

EjReportes

Tabla de contenido

Introduccion	3
Objetivos	3
Herramientas	3
C#	3
Descripcion	3
Base de datos	5
Crud	7
Reportes	11
PHP	14
Descripcion	14
Base de datos	15
Reportes	16

Introduccion

Este documento presenta el desarrollo de un sistema de gestión basado en el patrón de diseño Modelo-Vista-Controlador (MVC), implementado en dos entornos distintos: **Windows Forms en C# con ReportViewer** y **PHP utilizando DOMPDF** para la generación de reportes en formato PDF.

Creado con el Personal Edition de HelpNDoc: [Generar eBooks Kindle con facilidad](#)

Objetivos

El objetivo principal de este proyecto es desarrollar un sistema que permita **generar reportes automatizados** tanto en un entorno de **escritorio (C#)** como en un entorno **web (PHP)**, utilizando herramientas adecuadas para cada plataforma. Además, el sistema en C# incluye operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para la gestión de datos de forma eficiente. Este proyecto busca demostrar cómo integrar la gestión de datos con la generación de reportes en dos entornos distintos pero complementarios.

Creado con el Personal Edition de HelpNDoc: [Crear fácilmente libros de EPub](#)

Herramientas

Entorno de C# (Escritorio):

- Lenguaje: C#
- IDE: Visual Studio
- Motor de base de datos: SQL Server
- Herramienta de reportes: ReportViewer
- Estructura adicional: CRUD manual con formularios de Windows Forms

Entorno de PHP (Web):

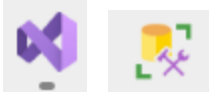
- Lenguaje: PHP
- Servidor local: XAMPP
- Motor de base de datos: MariaDB
- Generador de reportes PDF: DOMPDF
- Interfaz: HTML/CSS (si aplica)

Creado con el Personal Edition de HelpNDoc: [Noticias e información sobre software y herramientas de creación de documentación de ayuda](#)

Descripcion

Entorno de Desarrollo

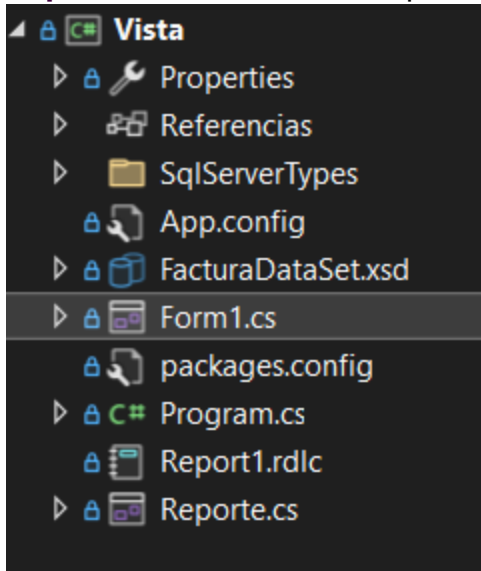
El desarrollo de este módulo se realizó en **Visual Studio**, utilizando el lenguaje **C#** bajo el enfoque de **aplicación de escritorio** con **Windows Forms**. La base de datos utilizada fue **SQL Server**, gestionada mediante SQL Server Management Studio.



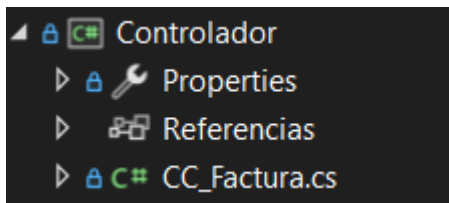
Arquitectura del Sistema

El sistema se estructura en **tres capas básicas**:

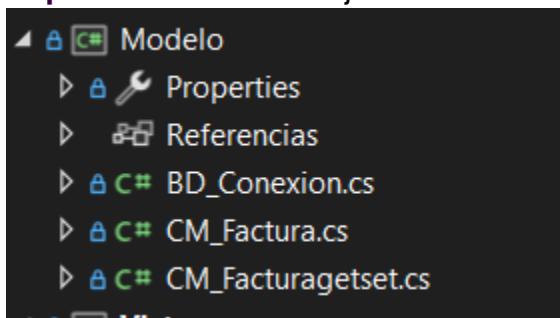
- **Capa de Vista:** formularios que interactúan con el usuario.



- **Capa de Controlador:** contiene la lógica del programa (clases que procesan datos y validaciones).



- **Capa de Modelo:** maneja la conexión con SQL Server y ejecuta consultas.



Esta arquitectura facilita el mantenimiento y escalabilidad del sistema.

Dependencias

El proyecto utiliza las siguientes dependencias:

- **Microsoft.ReportViewer.WinForms:** para la generación de reportes.
- **System.Data.SqlClient:** para la conexión con la base de datos.

Base de datos

Nombre de la base de datos: **Factura**

La base de datos creada para este sistema lleva por nombre **Factura**. Esta base de datos se encarga de almacenar la información relacionada con las facturas ingresadas desde el formulario, facilitando la gestión de los registros.

Tabla principal: **Factura**

Dentro de esta base de datos se creó una única tabla, también llamada **Factura**, la cual contiene los campos esenciales para el registro de cada factura. Entre los campos se encuentran:

- Un identificador único llamado **ID**, que se genera automáticamente.
- El campo **DESCRIPCION**, donde se indica el nombre del producto o servicio facturado.
- La **CATEGORIA**, que clasifica el tipo de producto.
- La **CANTIDAD**, que representa cuántas unidades se están facturando.
- El **PRECIO_UNITARIO**, correspondiente al valor por unidad.
- El campo **ITEBIS**, que almacena el valor del impuesto aplicado.
- Y finalmente, el campo **DESCUENTO**, que refleja cualquier rebaja otorgada al total.

```
CREATE TABLE Factura (  
    ID INT PRIMARY KEY IDENTITY(1,1),  
    DESCRIPCION NVARCHAR(100),  
    CATEGORIA NVARCHAR(50),  
    CANTIDAD INT,  
    PRECIO_UNITARIO DECIMAL(10,2),  
    ITEBIS DECIMAL(10,2),  
    DESCUENTO DECIMAL(10,2)  
);
```

Inserción de datos de prueba

Una vez creada la base de datos y la tabla, se procedió a insertar registros de ejemplo para verificar que la estructura funciona correctamente. Estos datos permiten realizar

pruebas de lectura, edición y eliminación desde el sistema.

```
INSERT INTO Factura (DESCRIPCION, CATEGORIA, CANTIDAD, PRECIO_UNITARIO, ITEBIS, DESCUENTO)
VALUES
('Teclado mecánico', 'Periféricos', 2, 1500.00, 270.00, 100.00),
('Monitor LED 24"', 'Pantallas', 1, 8000.00, 1440.00, 500.00),
('Mouse gamer', 'Periféricos', 3, 1200.00, 216.00, 150.00);
```

Visualización de los datos

Para verificar los datos almacenados, se puede ejecutar una consulta que muestre todos los registros de la tabla. Esto permite comprobar que los datos ingresados desde el formulario se reflejan correctamente en la base de datos

SQLQuery2.sql - L...\MASIEL\miaun (56))* SQLQuery1.sql - L...\MASIEL\miaun (68))*

`select * from Factura`

121 %

Results Messages

	ID	DESCRIPCION	CATEGORIA	CANTIDAD	PRECIO_UNITARIO	ITEBIS	DESCUENTO
1	1	Laptop Lenovo	Electrónica	2	35000.00	0.18	0.05
2	2	Teclado mecanico	Electronica	1	1850.00	18.00	0.00
3	3	Celular Samsung A34	Tecnología	1	21000.00	0.18	0.10
4	4	Silla ergonómica	Muebles	3	7500.00	0.18	0.08
5	5	AUDICULARES	ELECTRONICA	15	100.00	18.00	5.00
6	6	hola	adad	3	340.00	12.00	3.00
7	8	muebles	madera	2	1500.00	18.00	25.00
8	9	TABLET	ELECTRONICA	4	100.00	18.00	0.00

Nombre del servidor y conexión

El nombre del servidor se obtiene al iniciar sesión en SQL Server Management Studio. Es importante usar este nombre exacto en la cadena de conexión del proyecto para garantizar la comunicación entre la aplicación y la base de datos.

Connect to Server

SQL Server

Login | Connection Properties | Always Encrypted | Additional Connection Parameters

Server

Server type: Database Engine

Server name: LAPMASIEL

Authentication: Windows Authentication

User name: LAPMASIEL\miaun

Password:

☐ Remember password

Connection Security

Encryption: Mandatory

☒ Trust server certificate

Host name in certificate:

Connect Cancel Help Options <<

Crud

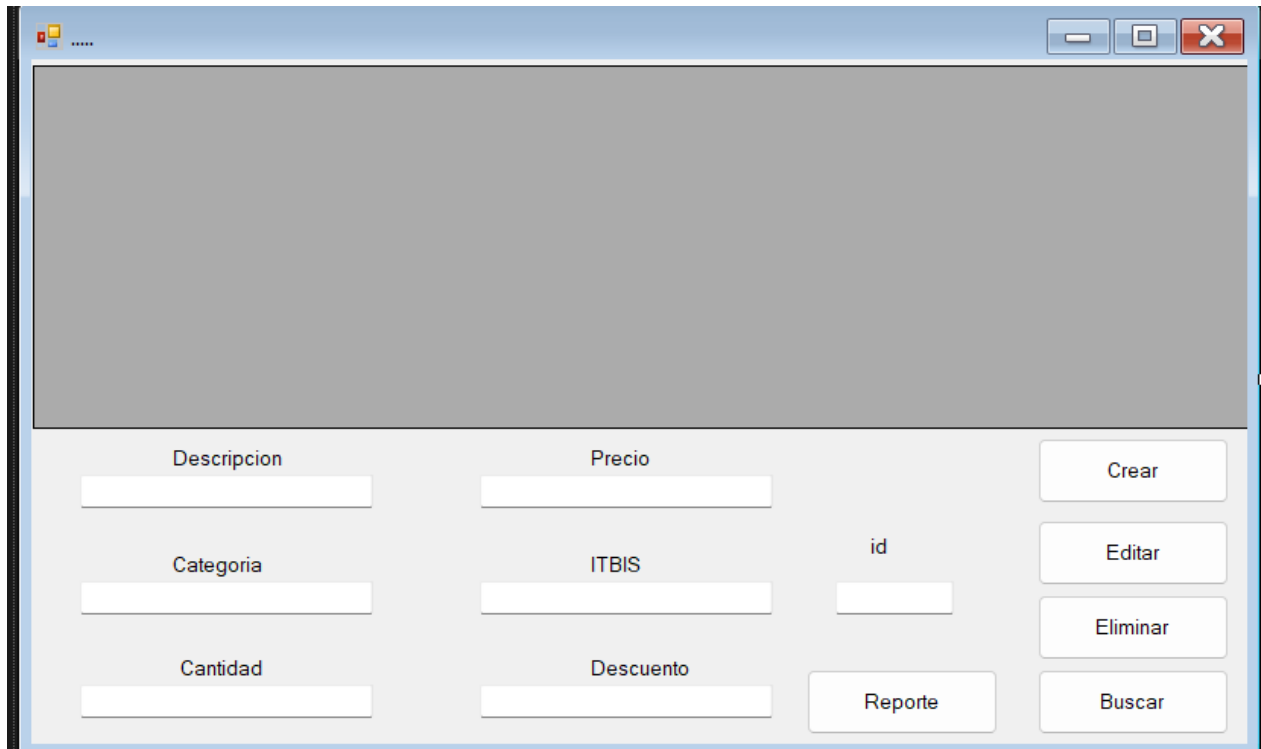
Paso 1: Crear el formulario

Abre Visual Studio y crea un proyecto de tipo Windows Forms App (.NET Framework).

Diseña un formulario con los siguientes controles:

- TextBox para cada campo: ID, DESCRIPCION, CATEGORIA, CANTIDAD, PRECIO_UNITARIO, ITEBIS, DESCUENTO .
- Botones: Crear, Buscar, Editar, Limpiar.

DataGridView para mostrar los datos de la tabla.



The screenshot shows a Windows application window with a title bar containing the text 'EjReportes'. The main area of the window is a large, empty gray rectangle. Below this rectangle, there is a form with several text boxes and buttons. The text boxes are arranged in two columns. The left column contains three text boxes labeled 'Descripcion', 'Categoria', and 'Cantidad'. The right column contains three text boxes labeled 'Precio', 'ITEBIS', and 'Descuento'. To the right of these text boxes, there are five buttons arranged vertically: 'Crear', 'Editar', 'Eliminar', 'Reporte', and 'Buscar'. The 'Reporte' button is located below the 'Descuento' text box.

Paso 2: Conectar con la base de datos

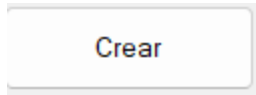
1. Usa la siguiente cadena de conexión para SQL Server local:
csharp

```
string connectionString = "Server=localhost;Database=Factura;Integrated Security=True;";
```

2. Esta cadena debe usarse con cada conexión SqlConnection.

Paso 3: Insertar datos en la tabla

En el evento del botón Crear, implementa el código para validar y luego insertar datos usando INSERT INTO. Verifica que los campos numéricos como CANTIDAD, PRECIO_UNITARIO, ITEBIS, DESCUENTO sean válidos antes de ejecutar la consulta.



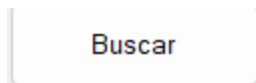
```
1 referencia
public void Insertar(Factura factura)
{
    using (var conexion = ObtenerConexion())
    {
        string query = @"INSERT INTO Factura (DESCRIPCION, CATEGORIA, CANTIDAD, PRECIO_UNITARIO, ITEBIS, DESCUENTO)
        VALUES (@Descripcion, @Categoria, @Cantidad, @PrecioUnitario, @Itebis, @Descuento)";
        SqlCommand comando = new SqlCommand(query, conexion);

        comando.Parameters.AddWithValue("@Descripcion", factura.Descripcion);
        comando.Parameters.AddWithValue("@Categoria", factura.Categoria);
        comando.Parameters.AddWithValue("@Cantidad", factura.Cantidad);
        comando.Parameters.AddWithValue("@PrecioUnitario", factura.PrecioUnitario);
        comando.Parameters.AddWithValue("@Itebis", factura.Itebis);
        comando.Parameters.AddWithValue("@Descuento", factura.Descuento);

        comando.ExecuteNonQuery();
    }
}
```

Paso 4: Buscar una factura existente

En el evento del botón Buscar, se usa el ID como clave para buscar en la tabla con SELECT * FROM Factura WHERE ID = @id. Si se encuentra, se llenan los campos del formulario con los datos.



```

1 referencia
private void btnBuscar_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrEmpty(txtBuscar.Text))
    {
        int id;
        if (int.TryParse(txtBuscar.Text, out id))
        {
            var factura = controlador.BuscarFacturaPorID(id);
            if (factura != null)
            {
                List<Factura> lista = new List<Factura> { factura };
                dgvFacturas.DataSource = null;
                dgvFacturas.DataSource = lista;
            }
            else
            {
                MessageBox.Show("Factura no encontrada.");
            }
        }
        else
        {
            MessageBox.Show("Introduce un ID válido.");
        }
    }
}

```

Paso 5: Editar una factura

El botón Editar permite actualizar los datos de una factura existente, usando el ID como referencia con UPDATE.

Editar

```

1 referencia
private void btnEditar_Click(object sender, EventArgs e)
{
    Factura f = new Factura
    {
        ID = int.Parse(txtID.Text),
        Descripcion = txtDescripcion.Text,
        Categoria = txtCategoria.Text,
        Cantidad = int.Parse(txtCantidad.Text),
        PrecioUnitario = decimal.Parse(txtPrecio.Text),
        Itebis = decimal.Parse(txtITEBIS.Text),
        Descuento = decimal.Parse(txtDescuento.Text)
    };
    controlador.ActualizarFactura(f);
    CargarDatos();
}

```

Paso 6: Mostrar los datos

La función CargarDatos() ejecuta una consulta SELECT y carga el resultado en el DataGridView para visualizar todas las facturas registradas.

```
private void CargarDatos()
{
    dgvFacturas.DataSource = null;
    dgvFacturas.DataSource = controlador.ObtenerFacturas();
}
```

Creado con el Personal Edition de HelpNDoc: [Crea sin esfuerzo un sitio web de documentación de calidad profesional con HelpNDoc](#)

Reportes

Instalación de los paquetes necesarios

Para poder generar un reporte en Windows Forms, es necesario instalar el control de ReportViewer. Esto se realiza desde el Administrador de paquetes NuGet, buscando e instalando el paquete llamado:

- **Microsoft.Reporting.WinForms**

Este paquete permite incorporar el visor de reportes dentro del formulario y vincularlo con los datos provenientes del sistema o base de datos.

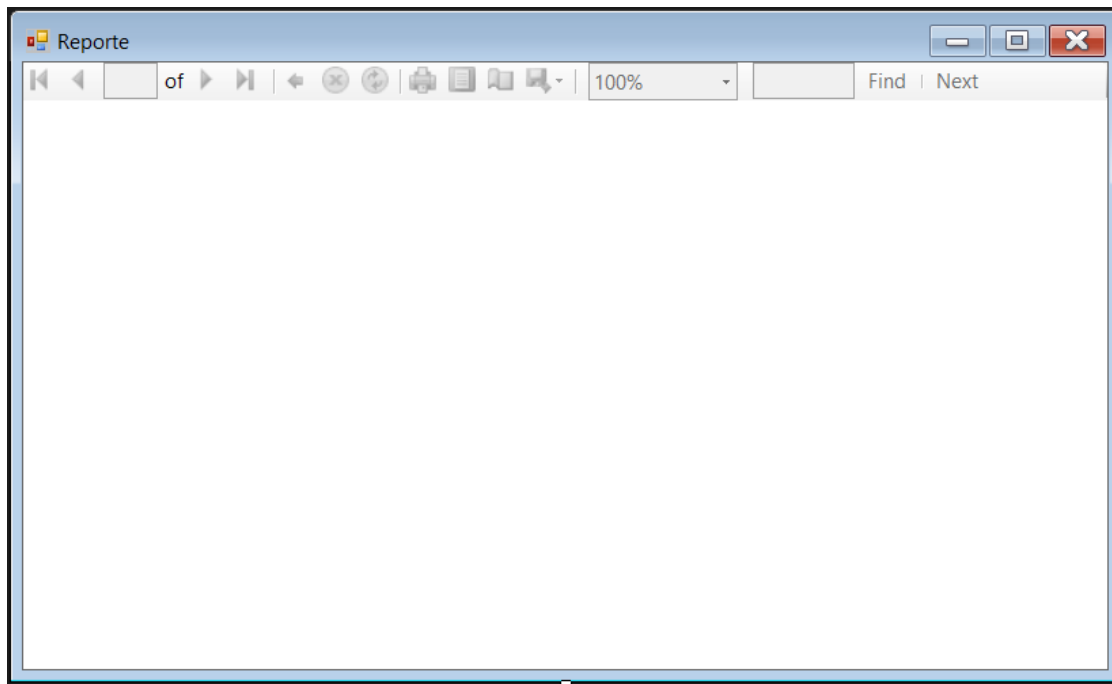
Creación del formulario de reportes

Se debe agregar un nuevo formulario al proyecto, el cual se utilizará exclusivamente para mostrar el reporte.

Agregar el control ReportViewer

Dentro del nuevo formulario, se debe arrastrar el control **ReportViewer** desde la caja de herramientas (Toolbox) hacia el formulario. Este control se ajusta al tamaño deseado para que ocupe todo el espacio visible.

Posteriormente, es necesario hacer clic en la pequeña flecha de configuración del control y seleccionar la opción "**Elegir un informe**", lo cual permite vincularlo con un archivo .rdlc.



Crear el archivo del reporte (.rdlc)

A continuación, se debe agregar un nuevo archivo de tipo **"Informe"** al proyecto, con extensión `.rdlc`. Este archivo permitirá diseñar visualmente cómo se mostrará la información del reporte.

Durante su diseño, se agregan los campos deseados a través de una tabla o matriz, arrastrándolos desde el **origen de datos del informe**.

FACTURA

ID	DESCRIPCION	CATEGORIA	CANTIDAD	PRECIO UNITARIO	ITEBIS	DESCUENTO
[ID]	[DESCRIPCION]	[CATEGORIA]	[CANTIDAD]	[PRECIO_UNITARIO]	[ITEBIS]	[DESCUENTO]

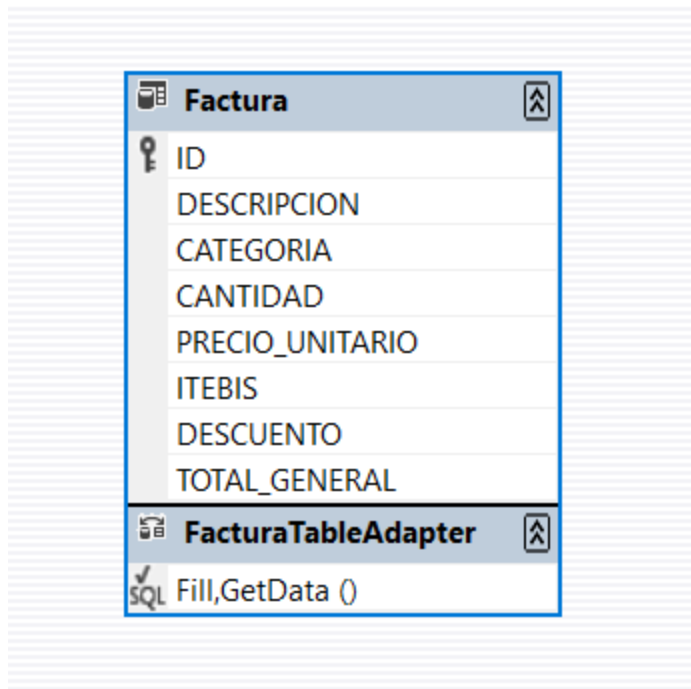
Para agregar un elemento al pie de página: agregue un elemento al informe y arrástrelo aquí.

Creación del Dataset

Es necesario crear un **Dataset tipado**, que actúe como puente entre la base de datos y el reporte. Este Dataset debe tener un **DataTable** con columnas que coincidan con los

nombres y tipos de los campos en la base de datos.

El Dataset también define una **consulta de selección**, que puede traer todos los registros de la tabla **Factura** o aplicar filtros si se desea.

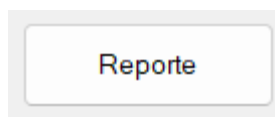


Configuración de los orígenes de datos

En el archivo **.rdlc**, se debe configurar el origen de datos del informe para que utilice el Dataset creado. Esto se hace desde la opción "**Orígenes de datos del informe**", vinculando el reporte con el DataTable definido.

Botón para mostrar el reporte

En el formulario principal, se debe agregar un botón con una función que abra el formulario del reporte. Este botón simplemente crea una nueva instancia del formulario de reporte y lo muestra con el método **.Show()** o **.ShowDialog()**.



Visualización del reporte

Al ejecutar la aplicación y hacer clic en el botón correspondiente, se abre el formulario que contiene el **ReportViewer**, y este muestra todos los datos de la tabla **Factura** de forma estructurada y profesional, ideal para imprimir o guardar como PDF.

Reporte

1 de 1 100 % Buscar | Siguiente

FACTURA

ID	DESCRIPCION	CATEGORIA	CANTIDAD	PRECIO UNITARIO	ITEBIS	DESCUENTO
1	Laptop Lenovo	Electrónica	2	35000,00	0,18	0,05
2	Teclado mecanico	Electronica	1	1850,00	18,00	0,00
3	Celular Samsung A34	Tecnología	1	21000,00	0,18	0,10
4	Silla ergonómica	Muebles	3	7500,00	0,18	0,08
5	AUDICULAR ES	ELECTRONICA	15	100,00	18,00	5,00
6	hola	adad	3	340,00	12,00	3,00
8	muebles	madera	2	1500,00	18,00	25,00
9	TABLET	ELECTRONICA	4	100,00	18,00	0,00

Creado con el Personal Edition de HelpNDoc: [Generador de documentación y EBooks gratuito](#)

PHP

Creado con el Personal Edition de HelpNDoc: [Generador de documentación CHM de ayuda gratuito](#)

Descripción

Este proyecto consiste en una página web simple desarrollada en PHP que permite **generar y descargar un reporte en formato PDF** con todo el contenido de una tabla de base de datos. La generación del PDF se realiza utilizando la librería **DOMPDF**, una herramienta popular en PHP que convierte código HTML y CSS en documentos PDF.



Generar Reporte PDF

La interfaz del sistema está compuesta por una sola página funcional con un botón principal identificado como “**Descargar reporte**” o “**Generar PDF**”, el cual, al ser presionado, ejecuta un script PHP que:

- Realiza la conexión a una base de datos MySQL diferente a la usada en el sistema anterior.
- Recupera todos los registros de una tabla específica (por ejemplo, una tabla de facturas o productos).
- Organiza estos datos en una estructura HTML que actúa como plantilla del reporte.
- Convierte dicha plantilla en un documento PDF utilizando DOMPDF.
- Fuerza la descarga del PDF automáticamente o lo abre en una nueva pestaña del navegador, según la configuración establecida.

Este tipo de implementación es ideal para exportar datos a un formato portátil y visualmente organizado, lo que facilita su impresión, archivado o envío por correo electrónico.

El sistema está compuesto por:

- Un archivo principal que muestra el botón.
- Un script que genera el PDF usando DOMPDF.
- Un archivo de conexión a la base de datos.
- La plantilla HTML que define el formato visual del reporte PDF.

Creado con el Personal Edition de HelpNDoc: [¿Qué es una herramienta de creación de documentación de ayuda?](#)

Base de datos

Para que el sistema funcione correctamente, es necesario contar con una base de datos MySQL que almacene la información que se mostrará en el reporte PDF.

Creación de la Base de Datos

Primero se debe crear una nueva base de datos desde un gestor como **phpMyAdmin**, **MySQL Workbench**, o directamente desde la consola de MySQL. Esta base de datos se utiliza exclusivamente para este sistema, separada del proyecto en C#.

```
CREATE DATABASE Factura;
```

Estructura de la Tabla

Dentro de la base de datos se crea una tabla que contiene los datos a exportar. Esta tabla puede incluir campos como ID, Descripción, Categoría, Cantidad, Precio, entre otros, dependiendo de la naturaleza del reporte que se quiere generar.

```
CREATE TABLE facturas (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    Descripcion VARCHAR(100),
    Categoria VARCHAR(50),
    Cantidad INT,
    PrecioUnitario DECIMAL(10,2),
    Itebis DECIMAL(10,2),
    Descuento DECIMAL(10,2)
);
```

Inserción de Datos

Una vez creada la tabla, se insertan registros de prueba o datos reales que se desean mostrar en el PDF. Esto puede hacerse manualmente o mediante scripts SQL.

```
INSERT INTO facturas (Descripcion, Categoria, Cantidad, PrecioUnitario, Itebis, Descuento) VALUES
('Laptop Lenovo', 'Tecnología', 2, 1200.00, 216.00, 100.00),
('Impresora HP', 'Oficina', 1, 450.00, 81.00, 50.00),
('Silla ergonómica', 'Muebles', 3, 150.00, 81.00, 0.00),
('Monitor Dell', 'Tecnología', 2, 300.00, 108.00, 20.00),
('Escritorio', 'Muebles', 1, 250.00, 45.00, 10.00);
```

Creado con el Personal Edition de HelpNDoc: [Producir ayuda online para aplicaciones Qt](#)

Reportes

Generación del Reporte con DOMPDF en PHP

Para generar el reporte en formato PDF se utilizó la biblioteca **DOMPDF**, que permite convertir contenido HTML en archivos PDF desde PHP de forma sencilla.

Inclusión de DOMPDF

En lugar de usar Composer, la biblioteca DOMPDF fue descargada manualmente desde su sitio oficial. Luego, se descomprimió y se colocó dentro del proyecto PHP. Se incluyó en el archivo mediante `require` o `require_once`, apuntando directamente a la ruta del archivo autoload o del archivo principal de la librería.

Página Principal

Se creó una página PHP sencilla con un enlace que indica “Descargar reporte” o “Descargar PDF”. Al hacer clic, se redirige a otro archivo que es el encargado de generar el reporte.

```

1  <a href="reporte.php" target="_blank">Generar Reporte PDF</a>
2

```

Generación del Reporte

El archivo que genera el PDF realiza las siguientes tareas:

- Se conecta a la base de datos utilizada (diferente a la del proyecto en C#).
- Se ejecuta una consulta que selecciona todos los registros de una tabla.
- Se construye una estructura HTML con los datos, generalmente en forma de tabla.
- Se configura el tamaño y orientación del PDF.
- Se usa DOMPDF para cargar ese HTML, renderizarlo y generar el PDF.
- Finalmente, se envía el archivo al navegador con encabezados HTTP que fuerzan su descarga.

```

1  <?php
2  require 'conexion.php';
3  require 'vendor/autoload.php';
4
5  use Dompdf\Dompdf;
6
7  $sql = "SELECT * FROM facturas";
8  $resultado = $conn->query($sql);
9
10 $html = '<h1>Reporte de Facturas</h1>';
11 $html .= '<table border="1" width="100%" cellpadding="5" cellspacing="0">';
12 $html .= '<tr><th>ID</th><th>Descripción</th><th>Categoría</th><th>Cantidad</th><th>Precio</th><th>ITEBIS</th><th>Descuento</th></tr>';
13
14 while ($row = $resultado->fetch_assoc()) {
15     $html .= '<tr>';
16     $html .= '<td>' . $row['ID'] . '</td>';
17     $html .= '<td>' . $row['Descripcion'] . '</td>';
18     $html .= '<td>' . $row['Categoría'] . '</td>';
19     $html .= '<td>' . $row['Cantidad'] . '</td>';
20     $html .= '<td>' . $row['PrecioUnitario'] . '</td>';
21     $html .= '<td>' . $row['Itebis'] . '</td>';
22     $html .= '<td>' . $row['Descuento'] . '</td>';
23     $html .= '</tr>';
24 }
25 $html .= '</table>';
26
27 $dompdf = new Dompdf();
28 $dompdf->loadHtml($html);
29 $dompdf->setPaper('A4', 'landscape');
30 $dompdf->render();
31 $dompdf->stream("reporte_facturas.pdf", ["Attachment" => false]);
32 ?>
33

```

Estilo del PDF

El contenido HTML incluye etiquetas básicas de formato como tablas.

