

Documentación Creación de Informes

A lo largo de este documento se va a desarrollar un informe en WinForms con RDLC. En este caso se va a crear un informe que muestre una lista de empleados con los siguientes datos:

- ID
- Nombre
- Cargo
- Salario

Antes de definir el informe, es necesario crear y configurar el proyecto en Visual Studio, seleccionando la opción de Windows Forms App (.NET Framework), y le damos un nombre.

Paso 1: Instalar las herramientas necesarias

Para poder crear el informe, es necesario tener instalados algunos componentes dentro de la aplicación Visual Studio.

a. Instalar el Diseñador de Informes (RDLC)

Debemos ir al menú **Extensiones** y seleccionar la opción **Administrar Extensiones**, en la barra de búsqueda tendremos que escribir **Microsoft RDLC Report Designer** y seleccionamos **Microsoft RDLC Report Designer for Visual Studio** y haz clic en Descargar. Si es necesario reiniciar Visual Studio se indicará en el proceso de instalación.

Es posible, también, descargarlo desde el Market Place de Visual Studio [Microsoft RDLC Report Designer - Visual Studio Marketplace](#)

b. Instalar ReportViewer desde NuGet

Lo siguiente es instalar el ReportViewer, necesario llevar a cabo este paso en cada uno de los proyectos en los que queramos realizar informes. Vamos al menú Herramientas y seleccionamos la opción **Administrador de paquetes NuGet**. Este paso se puede hacer de dos formas, por consola, o como en este paso haciendo uso del buscado de paquetes NuGet, seleccionando la opción **Administrar paquetes NuGet para la solución**.

Ahora, hay que buscar el paquete, dentro de la pestaña Examinar, Microsoft.ReportingServices.ReportViewerControl.WinForms. Seleccionar la última versión estable que nos ofrezca la aplicación, clicar en la opción Instalar y esperar a que termine la instalación.

Paso 2: Configurar la carga de ensamblados nativos

Para evitar que haya problemas al ejecutar el informe, es necesario agregar una porción de código en el constructor del formulario principal (Form1.cs) o en Program.cs antes de cargar cualquier informe.

En este caso se agregará en el constructor de Form1.cs

```
using System;
using System.Windows.Forms;

namespace WinFormsReportEmpleados
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            // Cargar ensamblados nativos para evitar errores en ReportViewer
            SqlServerTypes.Utilities.LoadNativeAssemblies(AppDomain.CurrentDomain.BaseDirectory);
        }
    }
}
```

Tras estos pasos, podemos comprobar que las DLLs están en el proyecto. Visual Studio agrega de forma automática las siguientes carpetas al proyecto:

- **SqlServerTypes\x86\SqlServerSpatial140.dll**
- **SqlServerTypes\x64\SqlServerSpatial140.dll**

Si no apareciesen en el Explorador de soluciones, puedes ir a la pestaña Ver y seleccionar la opción Mostrar todos los archivos y agrégalas de forma manual.

Paso 3: Crear la Base de Datos en localdb

En este paso se creará la base de datos de la misma forma que se hizo en el tema anterior, es decir, tenemos que abrir **Microsoft SQL Server Management Studio (SSMS)** y crear una nueva base de datos, a la que podemos llamar **EmpleadosDB**. Dentro de la base de datos creamos la tabla “Empleados” con la siguiente estructura:

Nombre de la columna	Tipo de datos	Permitir Nulos
Id	int	No (Clave primaria)
Nombre	nvarchar(100)	No
Cargo	nvarchar(50)	No
Salario	float	No

El campo Id se debe establecer como clave primaria y, además, en Propiedades de columna en la parte Especificación de identidad, marcar la opción Sí, para autoincrementar el Id.

Lo siguiente, aunque no es obligatorio, sí que es recomendable para la visualización de datos dentro del informe, por lo que deberíamos insertar datos en la tabla, abriendo una nueva consulta y ejecutando el script que se muestra a continuación:

```
USE EmpleadosDB;
```

```
INSERT INTO Empleados (Nombre, Cargo, Salario) VALUES  
( 'Juan Pérez', 'Gerente', 5000),  
( 'María López', 'Desarrollador', 3000),  
( 'Carlos Sánchez', 'Diseñador UX', 2800),  
( 'Ana García', 'Analista', 3200);
```

Y verificamos que los datos se guardaron correctamente, ejecutando:

```
SELECT * FROM Empleados;
```

Paso 4: Conectar Visual Studio con la Base de Datos

En este paso debemos crear el contexto de datos en ADO.NET, como se ha hecho en casos anteriores.

Volvemos a nuestro proyecto de Visual Studio y en el explorador de soluciones, seleccionamos Agregar → Nuevo elemento. En el panel izquierdo, seleccionamos Datos y elegimos el **Modelo de datos de Entity Framework (ADO.NET)** y le ponemos el nombre EmpleadosModel.edmx y seleccionamos Agregar.

Lo siguiente que debemos hacer es configurar la Conexión con la Base de Datos. En la ventana **“Elegir el contenido del modelo”**, hay que seleccionar la opción **EF Designer desde base de datos**.

A continuación, en la ventana **Elegir la conexión de datos**, hay que seleccionar la opción **“Nueva conexión”** y utilizar

- Servidor: (localdb)\MSSQLLocalDB
- Base de datos: EmpleadosDB
- Autenticación: “Autenticación de Windows

Marcamos la opción “Guardar configuración de conexión en App.config”.

Después tenemos que seleccionar las tablas para incluir en el Modelo. Expande las tablas y marca la tabla “Empleados” y clic en Finalizar. Visual Studio generará el modelo con:

- EmpleadosModel.edmx (el diagrama visual del modelo)
- EmpleadosEntities (el contexto de datos)
- Empleado (la clase mapeada a la tabla Empleados)

Paso 5: Crear un Método para Obtener los Datos

Ahora necesitamos un método para traer los datos de la tabla Empleados. Creamos la clase EmpleadoRepositorio.cs e insertamos en ella el siguiente código, dentro del namespace:

```
public class EmpleadoRepositorio
{
    public static List<Empleado> ObtenerEmpleados()
    {
        using (var contexto = new EmpleadosEntities()) // Usa el contexto ADO.NET
        {
            return contexto.Empleados.ToList(); // Devuelve todos los empleados
        }
    }
}
```

Si no tenemos claro cómo hemos llamado al contexto al generar el modelo ADO.NET, se puede verificar de la siguiente forma:

- Revisar en el explorador de soluciones si está el archivo con extensión .edmx. Haz doble clic en él para abrir el diseñador visual. Haz clic en un área en blanco del diseñador y en la ventana de Propiedades busca “Nombre del contenedor de entidades”, que será el nombre del contexto
- En el código generado, expande el archivo con extensión .edmx, abre el contexto (de extensión .cs) o un archivo similar que esté dentro de la carpeta y busca una clase que herede de DbContext. El nombre de la clase es el que debes usar.
- Revisar el App.config, busca la sección <connectionStrings> y dentro de name=”...” estará el nombre del contexto.

Paso 6: Crear el informe RDLC

Ahora diseñamos ya el informe RDLC para mostrar la lista de empleados. Lo primero que tenemos que hacer es irnos al explorador de soluciones y agregar un nuevo elemento. En la categoría Datos, debería aparecer “Informe” y lo seleccionamos y le ponemos de nombre EmpleadosReport.rdlc y hacemos clic en la opción “Agregar”.

- Agregar un conjunto de datos al Informe
Para ello abrimos EmpleadosReport.rdlc haciendo doble clic. En el panel Datos del informe, haz clic en “Conjunto de Datos” y seleccionamos la opción **“Agregar nuevo conjunto de datos”**.

Debemos configurar los datos


- Nombre del conjunto de datos: DataSetEmpleados
- Fuente de datos: “Nuevo origen de datos”
- Selecciona “Objeto”
- Expande WinFormsReportEmpleados y selecciona la clase Empleado

En este punto, si no nos aparece nuestra clase Empleado, compila el proyecto antes de hacer este paso.

- Diseña la Tabla del Informe

Haz clic (botón derecho del ratón) sobre la plantilla del formulario y selecciona la opción “Insertar” y elegimos la tabla.

En la tabla debes asignar los campos que se habían configurado previamente en la base de datos.

En cada una de las celdas de la tabla, cuando se pasa el ratón por encima, aparece el icono . Haz clic en él para ir insertando en cada una de ellas, los campos Id, Nombre, Cargo y Salario.

El informe ya estaría creado, ahora habría que mostrarlo en un formulario.

Paso 7: Crear el Formulario

En Visual Studio tenemos, ahora, que agregar un nuevo formulario de WinForms. Le ponemos, por ejemplo, VisorInforme.cs y lo abrimos en el diseñador. Desde la caja de herramientas arrastramos un ReportViewer.

Paso 8: Vincular el informe en el ReportViewer

La opción más rápida es abrir VisorInforme.cs en el diseñador, hacer clic en el ReportViewer y en la parte derecha haz clic en la flechita que aparece ►, en la opción “Choose Report” selecciona “**EmpleadosReport.dlc**”

Paso 11: Abrir el informe desde el FormularioPrincipal

Podemos hacer un botón en Form1.cs para abrir el visor. Una vez insertado hay que configurar el evento de clic en el botón

```
private void btnVerInforme_Click(object sender, EventArgs e)
{
    VisorInforme visor = new VisorInforme();
    visor.Show();
}
```

Aunque el informe RDLC está correctamente vinculado, el ReportViewer no recibe los datos desde la base de datos.

Cuando estamos utilizando una base de datos real con ADO.NET, el informe necesita que le enviemos los datos en tiempo de ejecución

a. Modificamos VisorInforme.cs para pasar los datos al ReportViewer

Abrimos VisorInforme.cs en modo código para modificar el constructor para que reciba los datos e incluimos el siguiente código:

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using Microsoft.Reporting.WinForms;

namespace WinFormsReportEmpleados
{
    public partial class VisorInforme : Form
    {
        public VisorInforme()
        {
            InitializeComponent();

            private void VisorInforme_Load(object sender, EventArgs e)
            {
                try
                {
                    // Obtener los datos desde la base de datos
                    List<Empleados> listaEmpleados = EmpleadoRepositorio.ObtenerEmpleados();

                    // Configurar la fuente de datos del informe
                    ReportDataSource fuenteDatos = new ReportDataSource("DataSetEmpleados",
listaEmpleados);
                    rpvEmpleados.LocalReport.DataSources.Clear();
                    rpvEmpleados.LocalReport.DataSources.Add(fuenteDatos);

                    // Refrescar el ReportViewer
                    rpvEmpleados.RefreshReport();
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error al cargar el informe: " + ex.Message);
                }
            }
        }
    }
}
```