Tipos de informe.

Existen principalmente dos tipos de informes en el mercado. La dinámica de uso es muy parecida en ambos. Un visor, una plantilla y un origen de datos. Se podría decir que básicamente cambian algunos nombres de propiedades y funciones.

- Reporting services, de Microsoft. Existen dos versiones, la de SQL server, con extensión .rdl, y la versión de cliente, con extensión .rdlc.
- Cristal Reports, de SAP, durante mucho tiempo incluida en la instalación por defecto de Visual Studio, fácil de usar y muy extendida en el ámbito profesional. Desde Visual Studio 2010 es opcional y debe instalarse como un complemento.

Nosotros nos centraremos en reporting services ya que parece que es la nueva opción por defecto de Microsoft.

Fuentes de datos.

La fuente de datos se emplea tanto en la etapa de creación del report como en el momento de la ejecución de la aplicación, fundamentalmente existen dos tipos de fuentes y emplearemos una u otra en función de la arquitectura de la aplicación.

- Dataset. Mediante una consulta podemos cargar toda la información ya formateada. El dataset mantiene la estructura permitiendo mapearlo en las plantillas. En ejecución tendríamos que cargar el dataset antes de pasárselo al visor.
- Lista de objetos. Requiere definir un objeto con los datos que incluye el report, ya formateados. Al asociar la lista a la plantilla se mantiene la estructura del tipo de objeto de la lista.

Visores

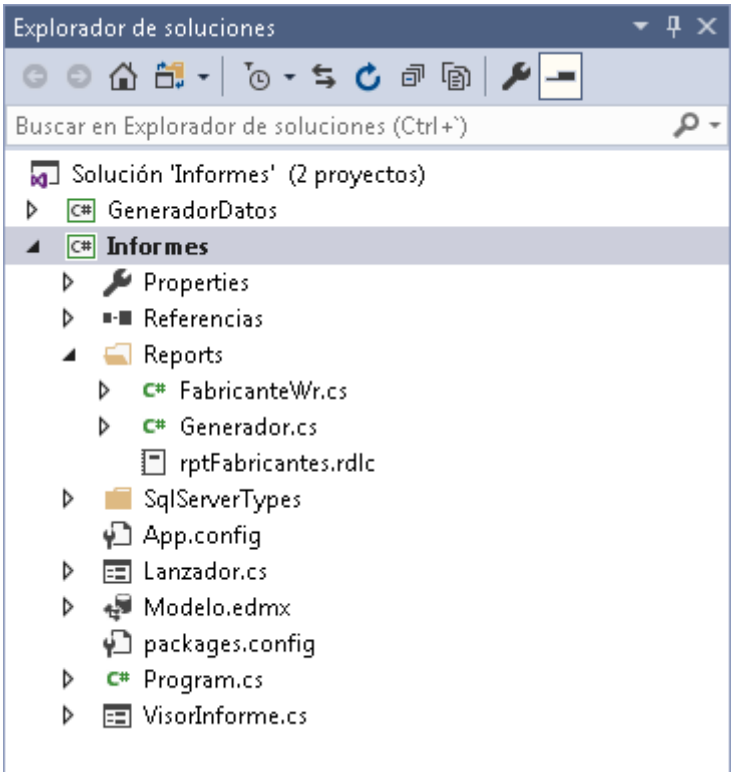
Existen dos tipos, uno para aplicaciones de escritorio, ReportViewer, y otro para entornos web, LocalReport, este último genera los bytes del fichero siendo labor del desarrollador definir de qué manera devolverlos al usuario.

NOTA: ReportViewer es un visor o envoltorio para LocalReport, internamente el visor emplea un objeto LocalReport para generar los informes, es decir, el visor nos facilita el trabajo pero podríamos emplear directamente un LocalReport sin la necesidad del visor.

Ejemplo

A partir de la base de datos de neumáticos vamos a construir un informe en el que podamos ver los distintos fabricantes disponibles, sus precios mínimos y máximos, así como el número de referencias distintas disponibles en el catálogo.

El proyecto final tendrá el siguiente aspecto.



En la carpeta “Reports” contamos con los siguientes elementos:

- Clase “Generador”, es una clase auxiliar que contara con una función por cada informe que generemos.
- Clase “FabricanteWr”, representa al modelo de datos empleado en el informe rptFabricantes.rdlc
- Informe “rptFabricantes.rdlc”, fichero con la estructura del informe, incluye la distribución de los datos en el documento y las fuentes de datos para el modelo “FabricanteWr”.

Los formularios disponibles son:

- Lanzador, cuya única función es la de invocar a la función encargada de generar y presentar el informe.
- VisorInforme, presenta el informe a través de un control con múltiples opciones de configuración y navegación.

Otros elementos son:

- Modelo.edmx, con el contexto de datos que nos permite ejecutar consultas para recuperar la información de los informes.
- App.config, fichero de configuración que incluye entre otras cosas la cadena de conexión para el contexto “Modelo.edmx”.

Instalar el diseñador de informes

Visual Studio 2017 no trae de serie las herramientas para trabajar con informes por lo que deberemos instalar el complemento que integra el diseñador.

Buscamos en google "Microsoft Rdlc Report Designer for Visual Studio" y descargamos e instalamos el complemento.

https://marketplace.visualstudio.com/items?itemName=ProBITools.MicrosoftRdlcReportDesignerforVisualStudio-18001

NOTA: ESTE COMPLEMENTO SOLO SE INSTALA UNA VEZ Y SIN EL NO TENDREMOS LA OPCION DE CREAR INFORME EN EL ASISTENTE.

Instalar el control ReportViewer

El control que permite visualizar los informes en las aplicaciones de escritorio debe **descargarse de Nuget POR CADA PROYECTO**.

Buscamos en google o en la web de nuget "Microsoft.ReportingServices.ReportViewerControl.Winforms"

Posiblemente nos mande a la siguiente dirección.

https://www.nuget.org/packages/Microsoft.ReportingServices.ReportViewerControl.Winforms/

COMANDO:

Install-Package Microsoft.ReportingServices.ReportViewerControl.Winforms -Version 150.900.148

Creamos un modelo para el informe

Si simplificamos mucho el concepto de informe tenemos que un informe es el resultado de una consulta debidamente con la posibilidad de ser exportado a múltiples tipos de documento ofimático.

Partiendo de la premisa anterior tenemos que la consulta para este ejemplo es la siguiente:

```
select p.fabricante, min(p.Precio) as preciodesde, max(p.precio) as preciohasta,
count(*) as referencias from productos p group by p.Fabricante order by
preciodesde
```

Podemos ver que tenemos 4 campos, fabricante de tipo cadena, precio desde y precio hasta de tipo numérico y referencias que también es un número pero en este caso de tipo entero.

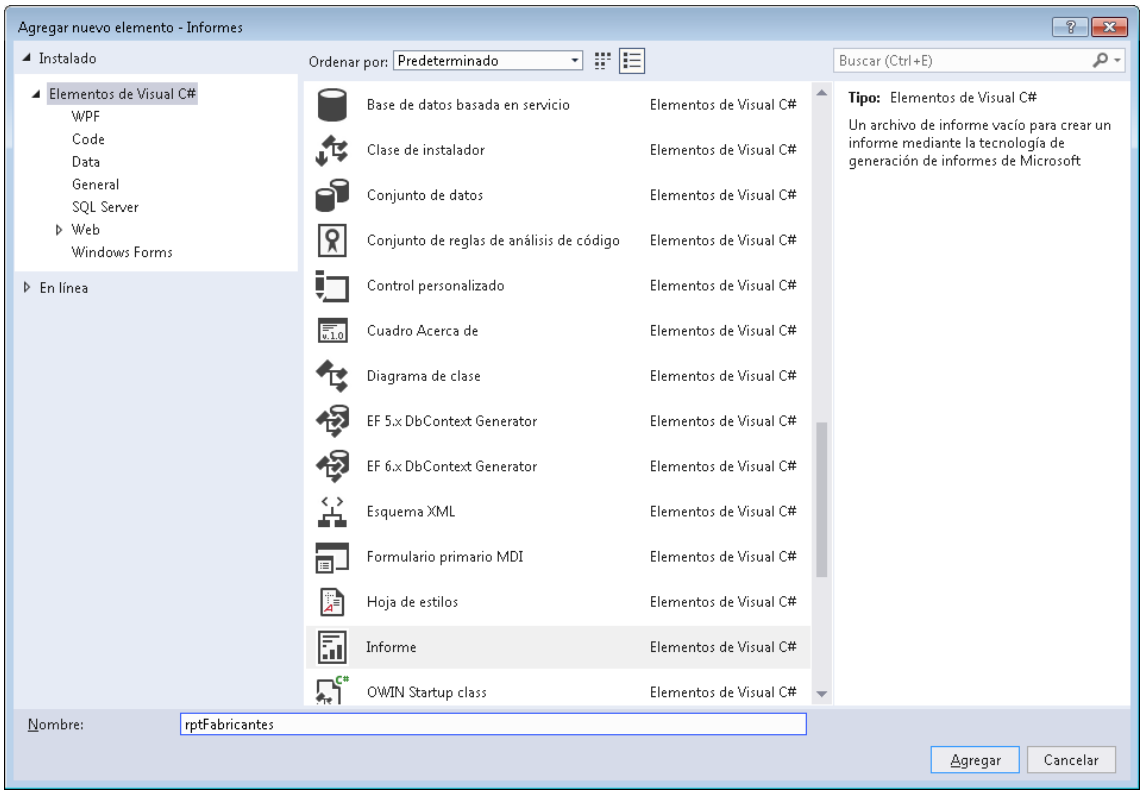
En algunos casos, como cuando creamos los informes dentro de un SQL Server, la propia consulta se convierte en la fuente de datos del informe. En el caso de los desarrolladores .Net lo normal desde hace algunos años es emplear un ORM en vez de gestionar manualmente las consultas, por este motivo necesitamos crear una clase envoltorio que se ajuste al formato de los campos de la consulta anterior. Para ello crearemos la siguiente clase que emplearemos como modelo para el dataset del informe.

```
public class FabricanteWr
{
    public string Fabricante { get; set; }
    public double PrecioDesde { get; set; }
    public double PrecioHasta { get; set; }
    public int Referencias { get; set; }
}
```

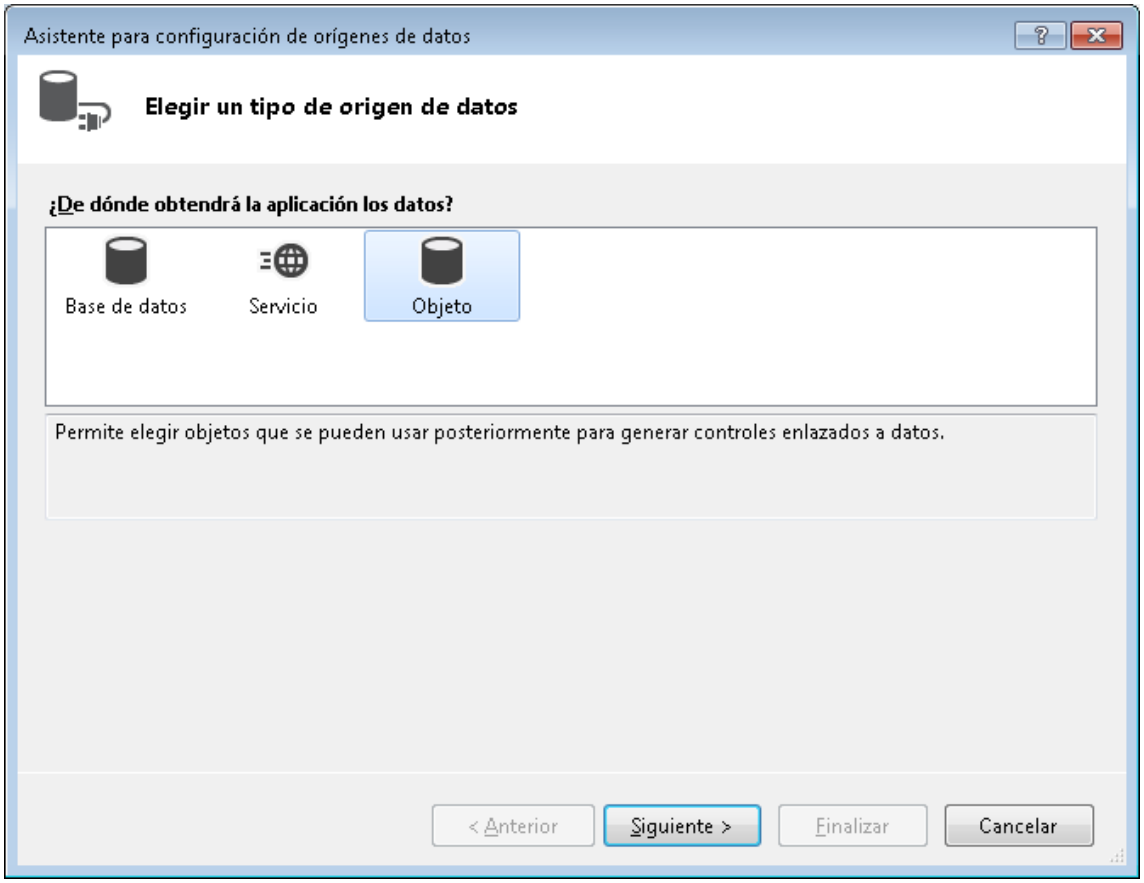
NOTA: si googleas un poco veras que el tipo recomendado para almacenar moneda es el “double” y no el “float”.

Crear la plantilla del informe

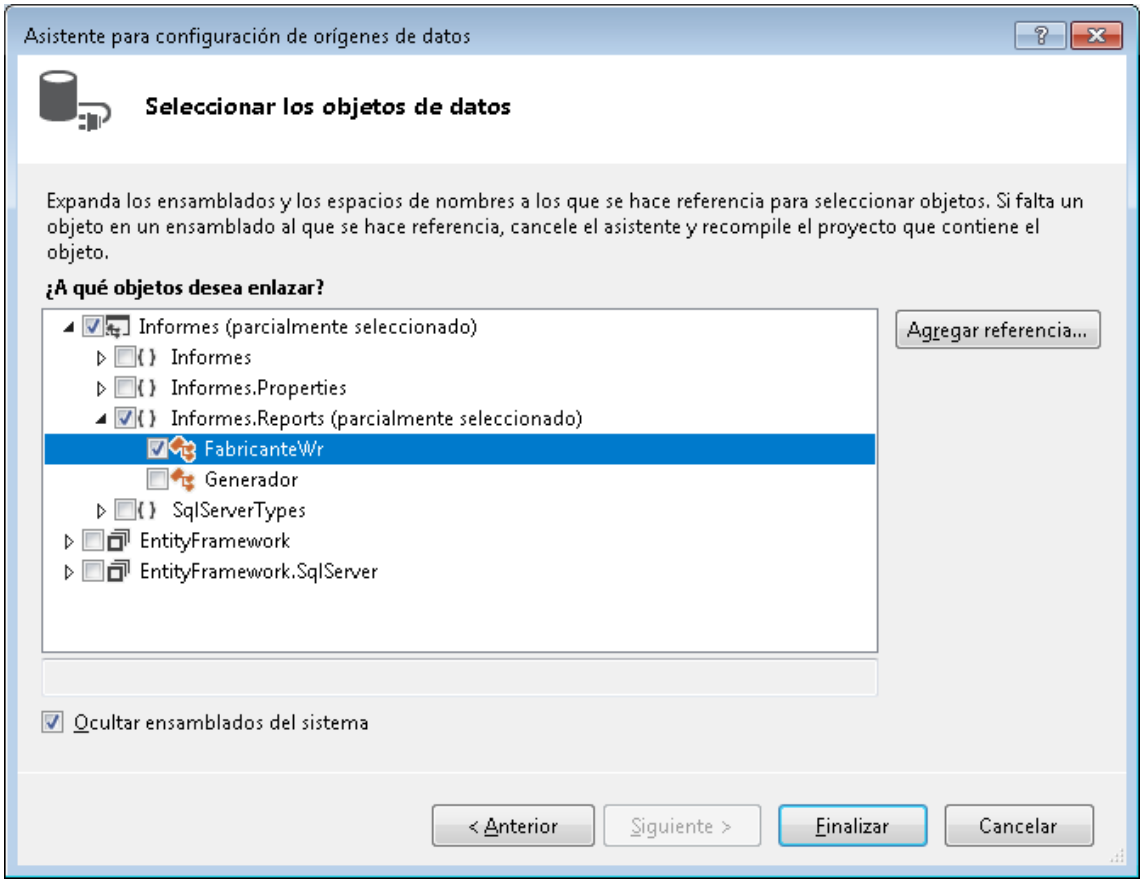
Si hemos instalado correctamente el complemento del diseñador ahora contaremos con el tipo “Informe” a la hora de añadir nuevos elementos al proyecto. Buscamos la plantilla y la llamamos “rptFabricantes”



El segundo paso es abrir la plantilla con el diseñador y en la opción “Datos del informe” nodo “Conjunto de datos” le damos a añadir nuevo conjunto de datos y seguimos el asistente como se indican en las siguientes imágenes.

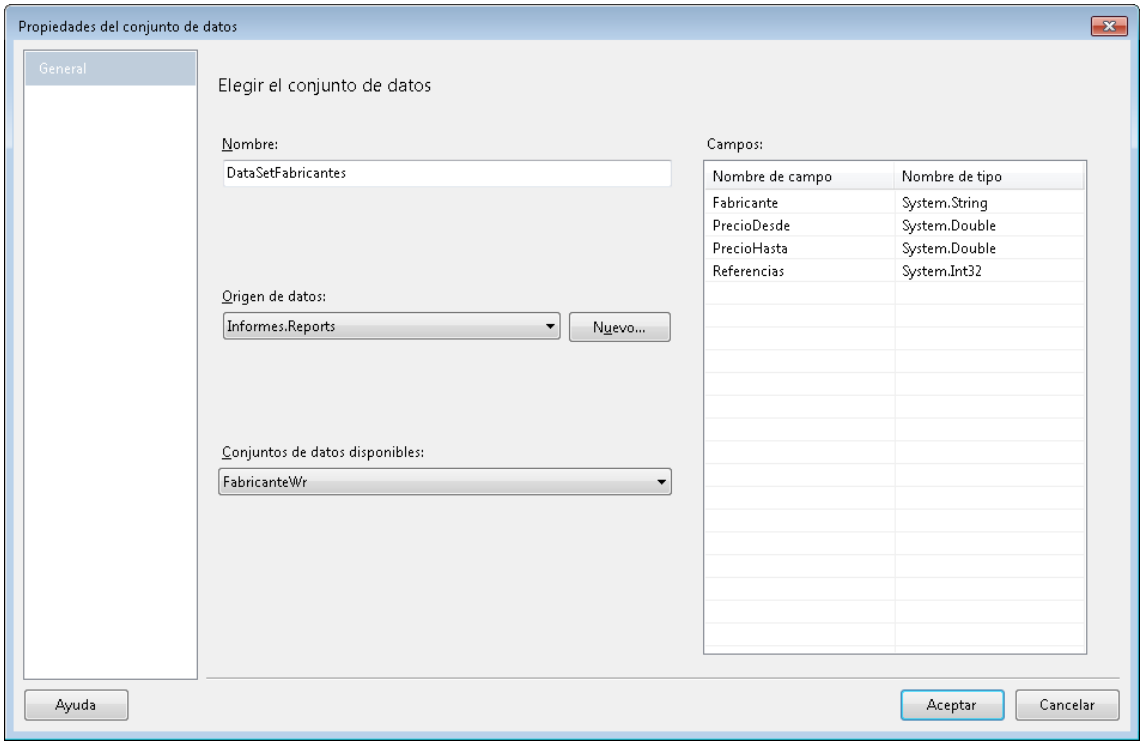


Y localizamos el tipo creado específicamente para este informe

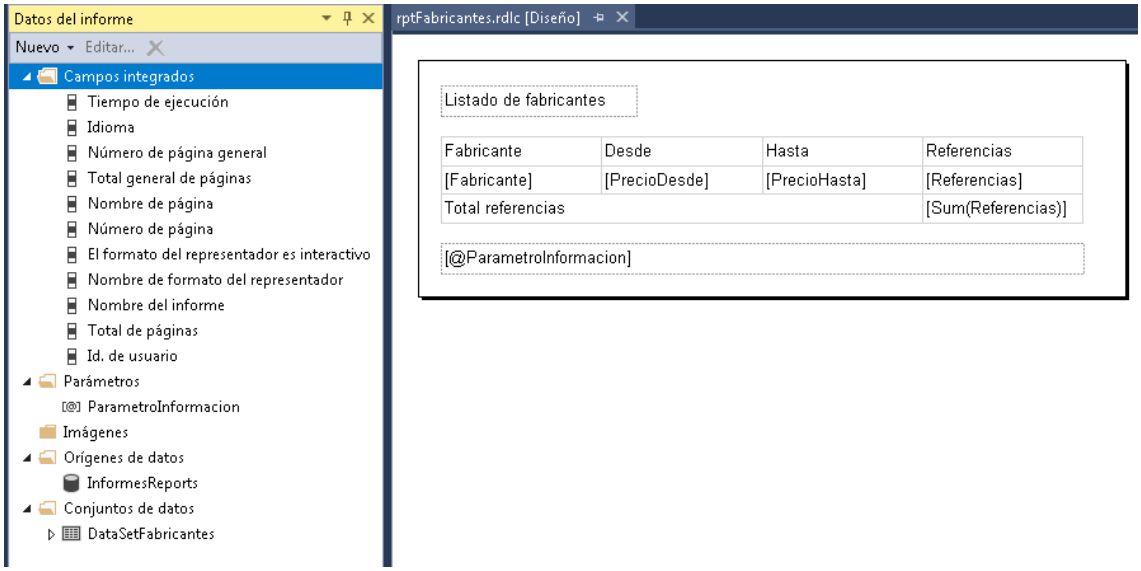


NOTA: si no aparece la clase y la has creado deberás compilar el proyecto antes de ejecutar el asistente.

Y le damos el nombre “DataSetFabricantes”. A través de este nombre más adelante asociaremos los datos de la base de datos con el informe e Intellisense NO LO SUGIERE, por lo que debería ser fácil de recordar.



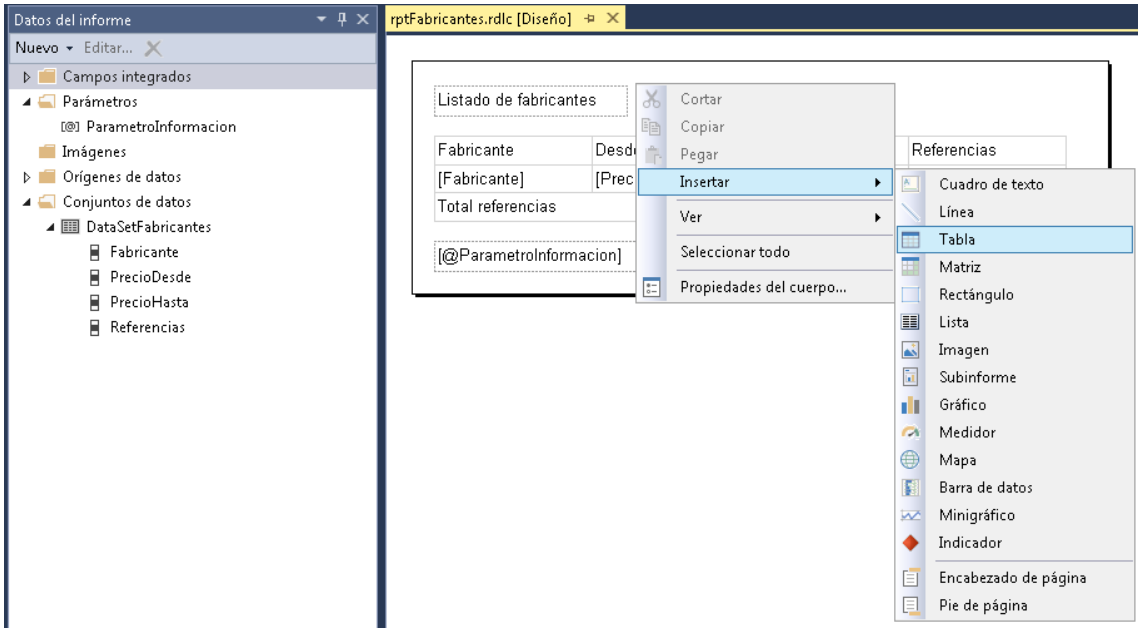
Ahora podremos dibujar nuestro informe a través del diseñador quedando algo parecido a la siguiente imagen.



Fíjate en las distintas opciones disponibles, el principal elemento con el que contamos para exponer la información de la base de datos es la tabla. Sobre la tabla podremos colocar los campos del dataset (fabricante, preciodesde, preciohasta, referencias) y automáticamente en tiempo de ejecución aparecerá una fila por cada dato. Fíjate que los nombres de estos campos aparecen entre [], por ejemplo **[Fabricante]**.

Los parámetros nos permiten a través de código intercambiar valores con el formulario, en este caso aparecen con el formato [@], por ejemplo **[@ParametroInformacion].**

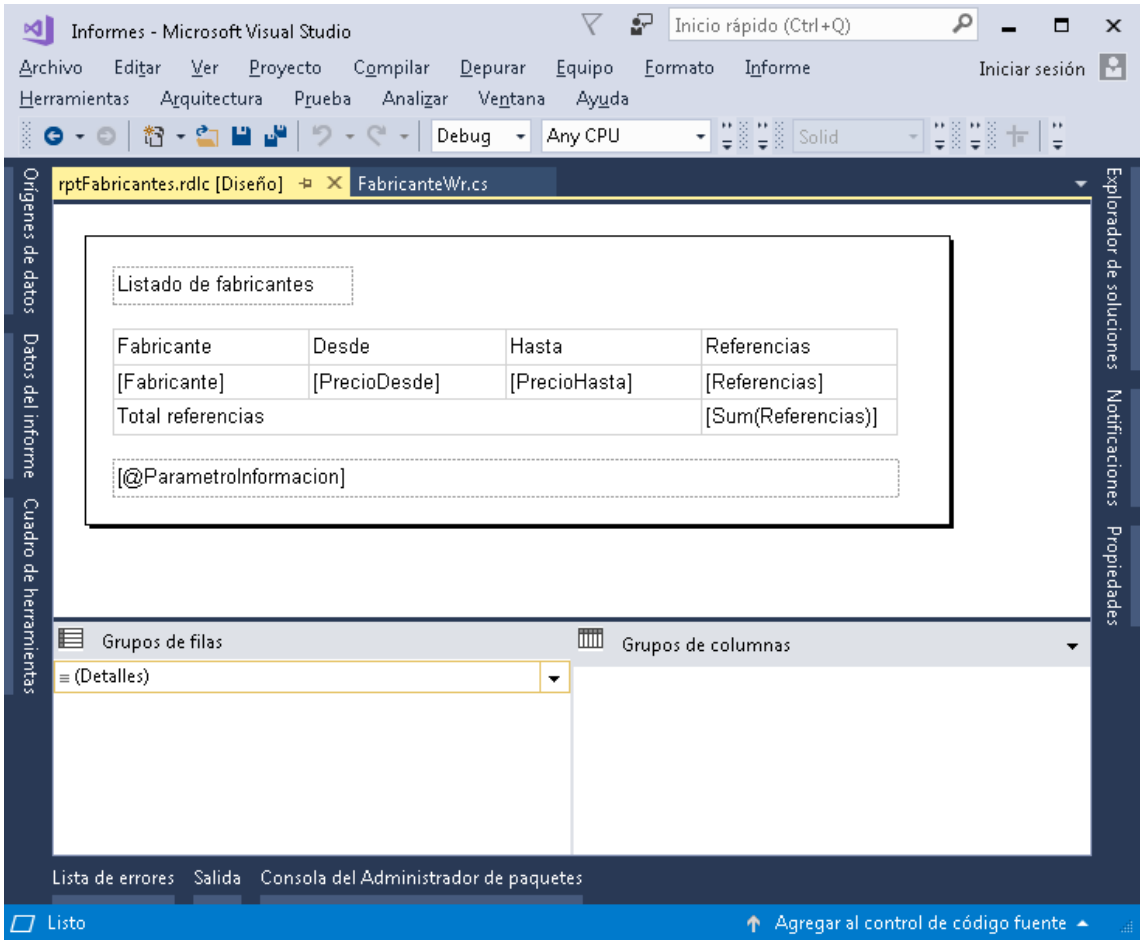
NOTA: los distintos elementos de diseño están disponibles a través del menú contextual en la sub opción “Insertar”, la siguiente imagen muestra las opciones disponibles.



Opciones interesantes

Encabezado de página y pie de página permiten insertar regiones en el report que se imprimirán por cada hoja del informe.

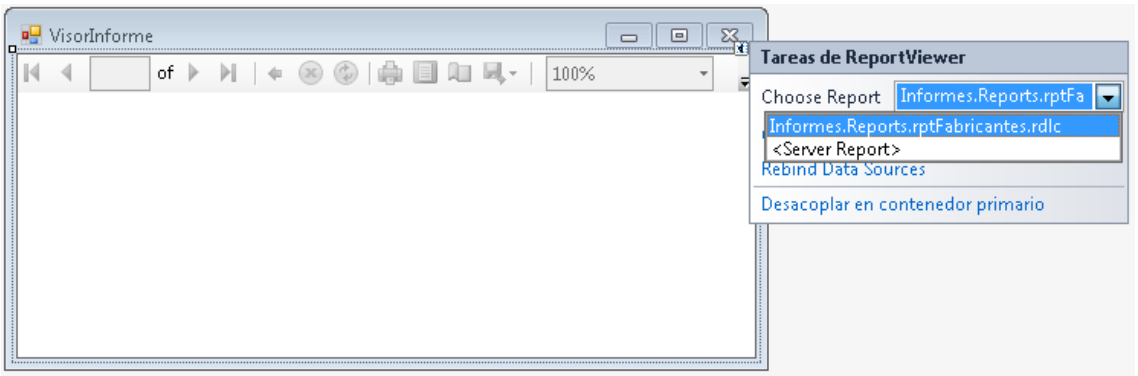
Además “Grupos de filas” y “Grupos de columnas” permiten agrupar los datos de los dataset por si quisiéramos sacar resúmenes de grupo.



Formulario VisorInforme

Creamos un formulario y le añadimos un único control ReportViewer y lo asociamos a nuestro informe (esto es opcional porque lo reasignaremos después por código).

Las propiedades que deberemos ajustar son el Dock, el nombre de la variable del visor (rpvVisorInforme) y su propiedad “Modifiers” como “Public”.



Clase Generador

La clase generador tendrá el siguiente código que comentaremos posteriormente.

NOTA: Antes deberás crear el contexto “NeumaticosBD” por tu cuenta.

```
    public class Generador
    {
        public void CargarInformeFabricantes()
        {
            VisorInforme visor = new VisorInforme();
            visor.rpvVisorInforme.LocalReport.ReportEmbeddedResource =
"Informes.Reports.rptFabricantes.rdlc";

            string consultaFabricantes =
                "select p.fabricante, min(p.Precio) as preciodesde, max(p.precio)
as preciohasta, " +
                "count(*) as referencias from productos p "+
                "group by p.Fabricante order by preciodesde";

            NeumaticosBD ctx = new NeumaticosBD();
            List<FabricanteWr> listaFabricantes =
ctx.Database.SqlQuery<FabricanteWr>(consultaFabricantes, new object[0]).ToList();

            ReportDataSource funteDatosInforme = new
ReportDataSource("DataSetFabricantes", listaFabricantes);
            visor.rpvVisorInforme.LocalReport.DataSources.Add(funteDatosInforme);

            ReportParameter parametro = new
ReportParameter("ParametroInformacion", "Esto es un mensaje de prueba");
            visor.rpvVisorInforme.LocalReport.SetParameters(parametro);

            visor.rpvVisorInforme.SetDisplayMode(DisplayMode.PrintLayout);
            visor.rpvVisorInforme.RefreshReport();

            //visor.rpvVisorInforme.LocalReport.SubreportProcessing +=
LocalReport_SubreportProcessing;

            //si quisieramos generar documentos sin el visor deberemos trabajar
con el metodo render
            //que devuelve el array de bits
            //visor.rpvVisorInforme.LocalReport.Render(multiples opciones de
configuracion);
            visor.Show();
        }

        /*
        private void LocalReport_SubreportProcessing(object sender,
SubreportProcessingEventArgs e)
        {
            //Podemos conocer el subreport a traves de la propiedad ReportPath,
ojo que es el nombre del
            //report cuando se incrusta en el contenedor
            //e.ReportPath

            //debemos asignar los datasource de los subreports
            //e.DataSources.Add(new ReportDataSource("nombre", datos));
            //los parametros hay que asignarselos al report padre y al insertar
el subreport en la pestaña parametros
            //hacer una asignacion manual entre los parametros del contenedor y
los del hijo
            throw new NotImplementedException();
        }
        */

    }
```

Lo primero instanciamos el formulario del visor y le asignamos la plantilla.

```
VisorInforme visor = new VisorInforme();
            visor.rpvVisorInforme.LocalReport.ReportEmbeddedResource =
"Informes.Reports.rptFabricantes.rdlc";
```

Lo segundo es cargar los datos, ojo, que vamos lanzar una consulta directamente a la base de datos y le vamos a decir que su tipo es el de la clase “FabricanteWr”

```
    string consultaFabricantes =
    "select p.fabricante, min(p.Precio) as preciodesde, max(p.precio) as
preciohasta, " +
    "count(*) as referencias from productos p "+
    "group by p.Fabricante order by preciodesde";

    NeumaticosBD ctx = new NeumaticosBD();
    List<FabricanteWr> listaFabricantes =
ctx.Database.SqlQuery<FabricanteWr>(consultaFabricantes, new object[0]).ToList();
```

En tercer lugar, rellenamos los datasets y los parametros del informe, fíjate que los nombres deben coincidir con lo indicado en el diseñador. En ambos casos debemos emplear los envoltorios **“ReportDataSource” y “ReportParameter”**.

```
            ReportDataSource funteDatosInforme = new
ReportDataSource("DataSetFabricantes", listaFabricantes);
            visor.rpvVisorInforme.LocalReport.DataSources.Add(funteDatosInforme);

            ReportParameter parametro = new
ReportParameter("ParametroInformacion", "Esto es un mensaje de prueba");
            visor.rpvVisorInforme.LocalReport.SetParameters(parametro);
```

Por ultimo configuramos algunos valores de presentación, refrescamos y mostramos el formulario visor.

```
            visor.rpvVisorInforme.SetDisplayMode(DisplayMode.PrintLayout);
            visor.rpvVisorInforme.RefreshReport();

            visor.Show();
```

Se ha incluido código comentado por si quisiéramos incrustar subinformes, en el caso de los subinformes antes de renderizarlos se producirá el evento “SubreportProcessing” en el cual podremos asignarle los datasets.

```
//visor.rpvVisorInforme.LocalReport.SubreportProcessing +=
LocalReport_SubreportProcessing;



        /*
        private void LocalReport_SubreportProcessing(object sender,
SubreportProcessingEventArgs e)
        {
            //Podemos conocer el subreport a traves de la propiedad ReportPath,
ojo que es el nombre del
            //report cuando se incrusta en el contenedor
            //e.ReportPath

            //debemos asignar los datasource de los subreports
```

```
            //e.DataSources.Add(new ReportDataSource("nombre", datos));
            //los parametros hay que asignarselos al report padre y al insertar
el subreport en la pestaña parametros
            //hacer una asignacion manual entre los parametros del contenedor y
los del hijo
            throw new NotImplementedException();
        }
        */
```

Recordatorio, como crear un contexto

1 crear con la plantilla el edmx, le hemos llamado Modelo.edmx

2 en nombre de contenedor NeumaticosBD, es el nombre que se da al contexto

3 en espacio de nombre “Informes”, el nombre del proyecto

4 marcar pluralizar/singularizar objetos

5 borrar las propiedades de navegación

Referencias

Hay que descargar el paquete de reporting-> “Microsoft Reporting Services Projects” -> no hace falta

https://marketplace.visualstudio.com/items?itemName=ProBITools.MicrosoftReportProjectsforVisualStudio

diseñador de reports ->” Microsoft Rdlc Report Designer for Visual Studio” ->INSTALAR ESTE

https://marketplace.visualstudio.com/items?itemName=ProBITools.MicrosoftRdlcReportDesignerforVisualStudio-18001

video paisano con nuget

https://www.youtube.com/watch?v=8UVqq8pgo9s

Instalamos -> HACE FALTA, INSTALAR POR PROYECTO

Via nuget -> Microsoft.ReportingServices.ReportViewerControl.Winforms

COMANDO:

Install-Package Microsoft.ReportingServices.ReportViewerControl.Winforms -Version 150.900.148

URL

https://www.nuget.org/packages/Microsoft.ReportingServices.ReportViewerControl.Winforms/

NOTA: primeros pasos, el report viewer hay que añadirlo a mano a la barra de herramientas (ahora es paquete nuget y no viene de serie)

https://docs.microsoft.com/en-us/sql/reporting-services/application-integration/integrating-reporting-services-using-reportviewer-controls-get-started?view=sql-server-2017