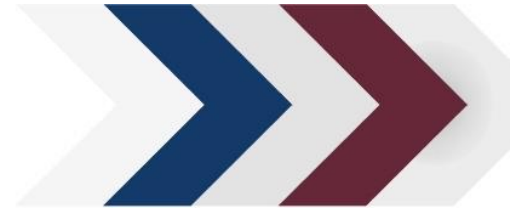


LENGUAJES DE ÚLTIMA GENERACIÓN

CONTROL DE USUARIO Y REPORTE

UNIDAD 4/CLASE ACTUAL:
CONTROL DE USUARIO Y REPORTE /CLASE

Autor de contenidos:
Mauricio Prinzo



PRESENTACIÓN

1. CONCEPTOS GENERALES

En esta clase abordaremos algunos conocimientos para el desarrollo de aplicaciones controles de usuarios y reportes.

Esperamos que que luego de esta clase, adquiera capacidad para:

- Ampliar los controles de la caja de herramientas
- Personalizar las funcionalidades de los controles
- Comprender las estructuras de datos utilizadas en un lenguaje de programación

2. CONTROLES DE USUARIO

Cuando desarrollamos aplicaciones de escritorio, podemos crear controles personalizados para mejorar la calidad de nuestros desarrollos.

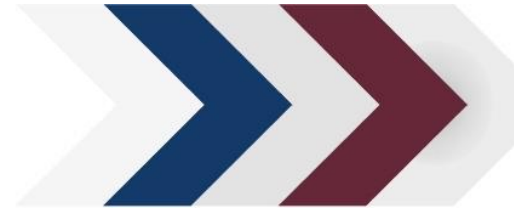
La ventaja de personalizar los controles es que se pueden extender las funcionalidades de los controles nativos de la plataforma de programación.

La clase **Control** es la clase base de los controles de formularios **WinForms**. Ofrece la interface necesaria de los controles básicos de .NET y toda la lógica para la presentación visual del control.

Las tareas son las siguientes según las fuentes de MSDN:

1. Expone un controlador de ventanas.
2. Administra el enrutamiento de mensajes.
3. Proporciona eventos del mouse (ratón) y del teclado y muchos otros eventos de la interfaz de usuario.
4. Proporciona características de diseño avanzadas.
5. Contiene muchas propiedades específicas a la presentación visual, como **ForeColor**, **BackColor**, **Height** y **Width**.
6. Proporciona la seguridad y compatibilidad para subprocesos necesarias para que un control de formularios Windows Forms actúe como un control de Microsoft® ActiveX®.





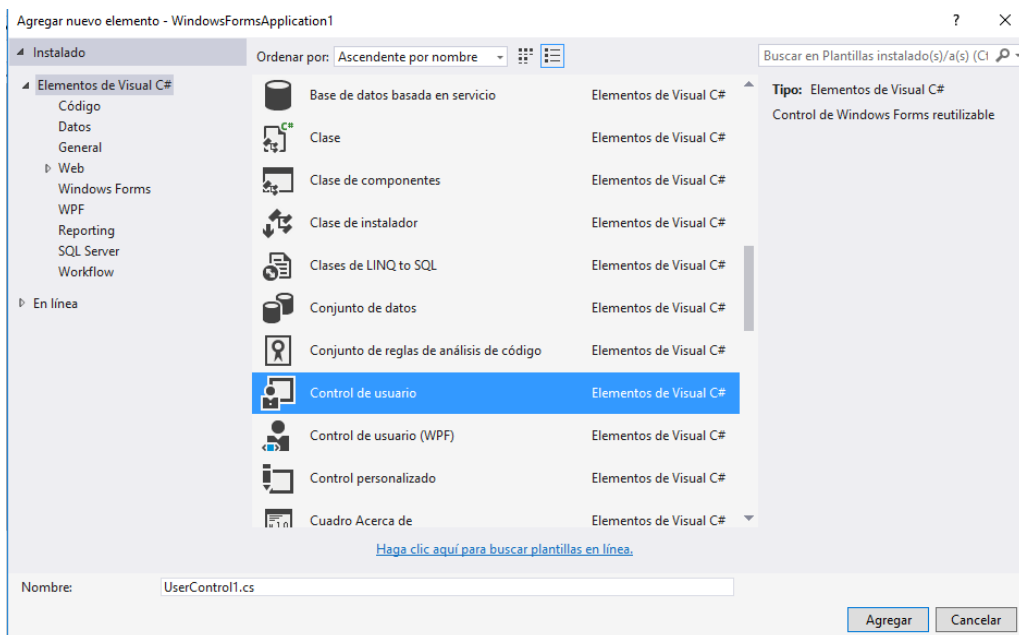
2.1 TIPOS DE CONTROLES

Se pueden desarrollar dos tipos de controles: compuesto y ampliado.

Controles compuestos

Se trata de un grupo de controles de formularios WinForms dentro de un contenedor común. En algunas bibliografías puede llamarse control de usuario.

Desde la opción Proyecto del menú y un nuevo elemento puede agregar un control de usuario



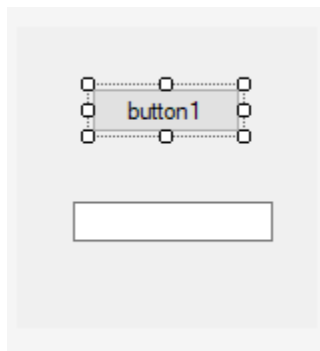
Veamos el siguiente código de ejemplo:

```
public partial class UserControl1 : UserControl
{
    public UserControl1()
    {
        InitializeComponent();
    }

    private void UserControl1_Load(object sender, EventArgs e)
    {
    }
}
```



Agregando los controles como se muestra en la siguiente figura:



Control Ampliado

Es cuando queremos preservar la funcionalidad de un control existente pero además le agregamos nuevas funcionalidades

```
public class cajaTextoNueva : System.Windows.Forms.TextBox {  
    protected override void OnPaint(System.Windows.Forms.PaintEventArgs e) {  
        base.OnPaint(e);  
    }  
}
```

3. REPORTES

Un reporte es un informe o una noticia. Este tipo de documento (que puede ser impreso, digital, audiovisual, etc.) pretende transmitir una información, aunque puede tener diversos objetivos. Existen reportes divulgativos, persuasivos y de otros tipos.

En el ámbito de la informática, los reportes son informes que organizan y exhiben la información contenida en una base de datos. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios.



El Diseñador de reportes de Microsoft Visual Studio proporciona una interfaz fácil de usar para crear reportes sólidos que incluyen datos procedentes de varios tipos de orígenes de datos.

Los reportes se guardan como archivos de definición de informe del cliente (.rdlc), estos archivos se basan en el mismo esquema que los archivos de definición de reporte (.rdl) publicados en los servidores de informes de SQL Server Reporting Services, pero se guardan y se procesan de manera distinta a los archivos .rdl.

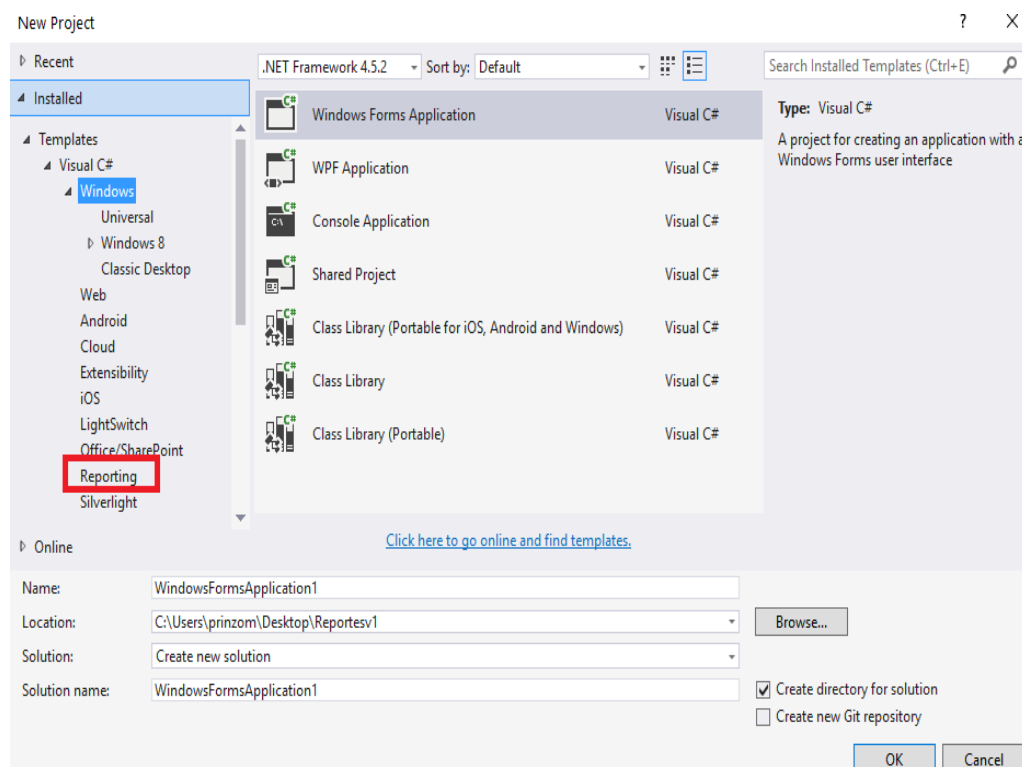
En tiempo de ejecución, los archivos .rdlc se procesan localmente, y los archivos .rdl se procesan remotamente

.NET tiene una librería:

using Microsoft.Reporting.WinForms;

El control ReportViewer admite un modo de procesamiento local que le permite ejecutar archivos de definición de informe de cliente (.rdlc) utilizando la capacidad de procesamiento integrada del control. Los informes de cliente que se ejecutan en modo de procesamiento local se pueden crear fácilmente en el proyecto de aplicación.

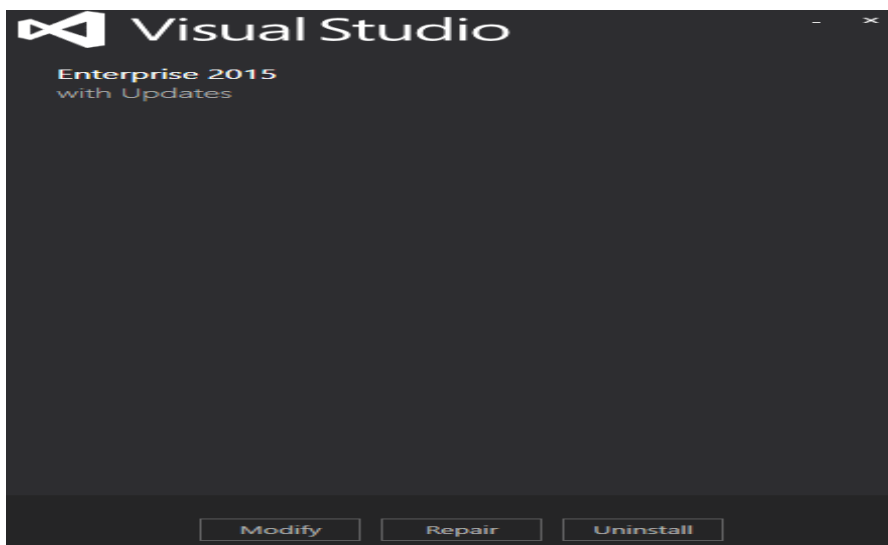
Para poder utilizar dicho control, tenemos que tener la opción de reportes en el menú de visual studio.





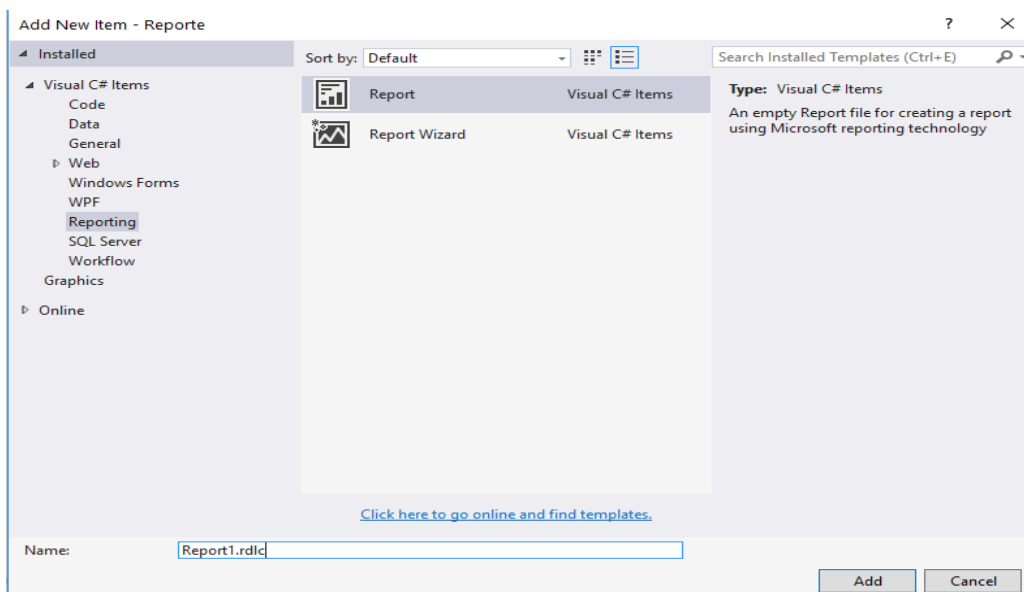
En caso de no tener la opción, tendrá q actualizar el VS, y seleccionar la siguiente opción:

VS2015

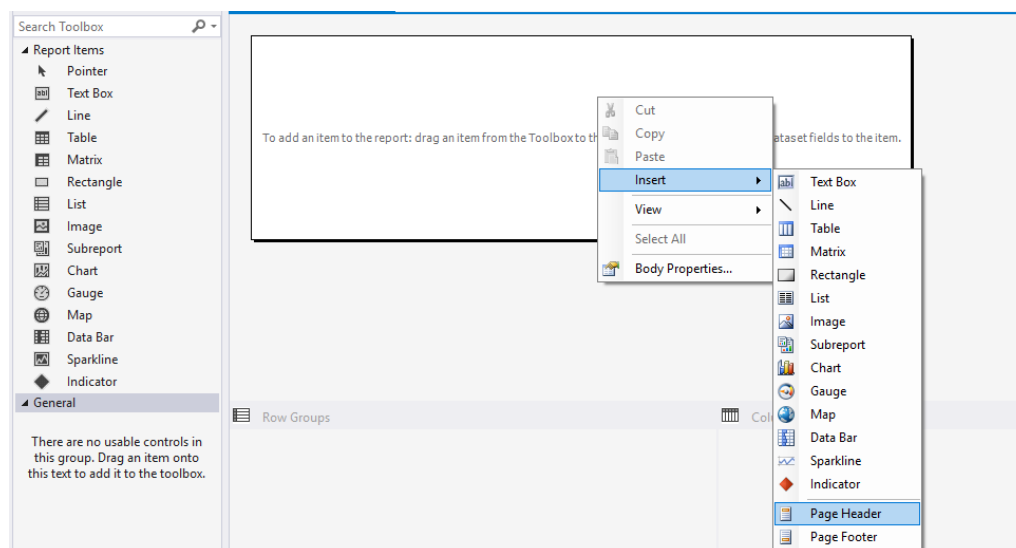


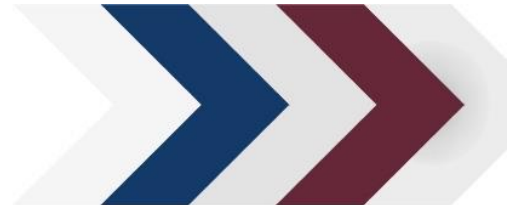


Para poder generar los reportes al sistema, se agrega el ítem de reportes el cual va a generar el archivo .rdlc.

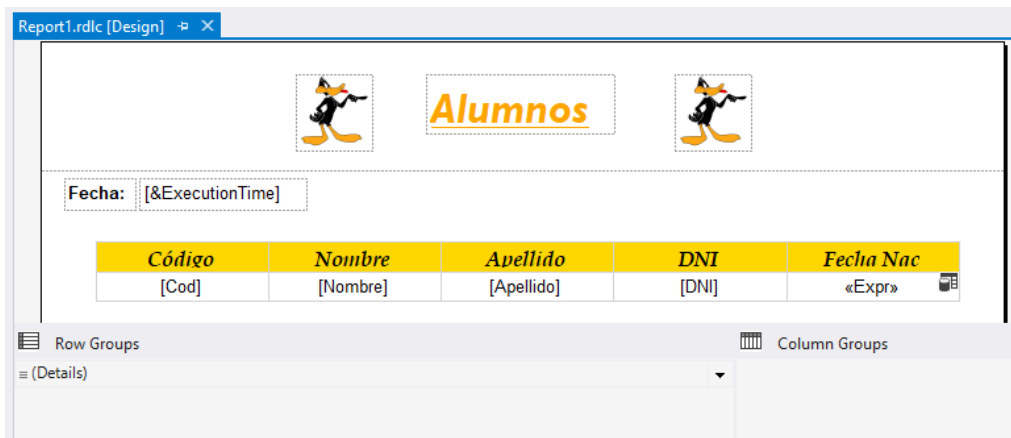


Luego se arma el diseño del informe como se desee: Encabezados, pie de página, gráficos, etc

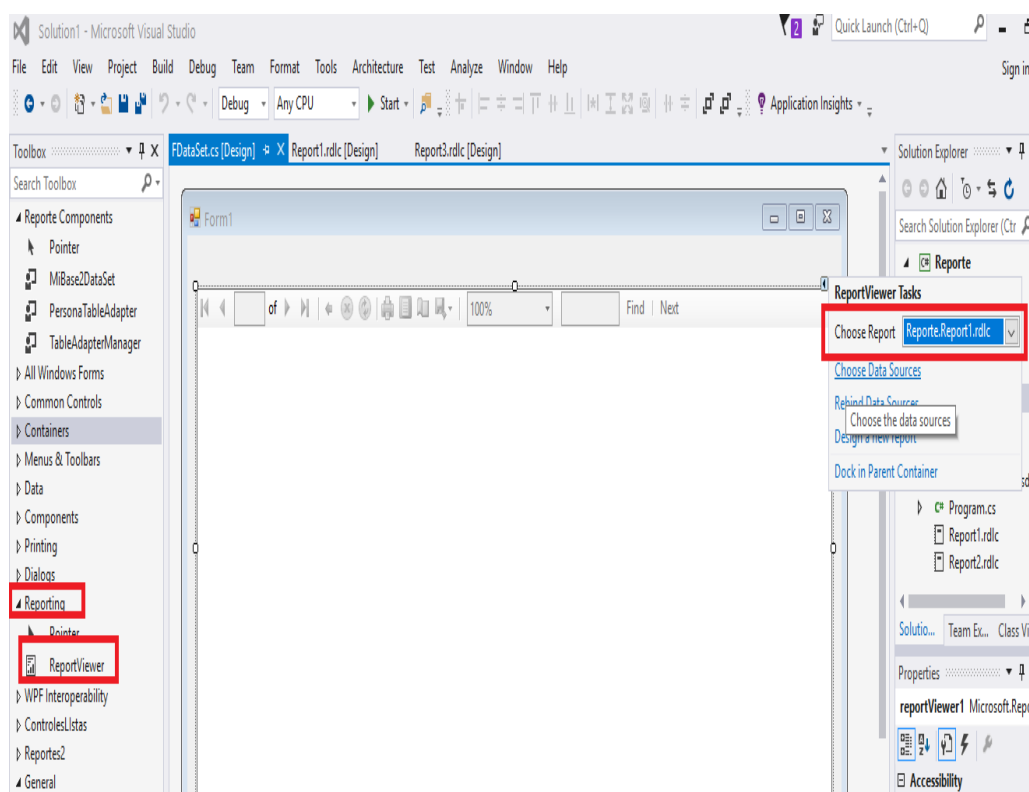




Ejemplo:



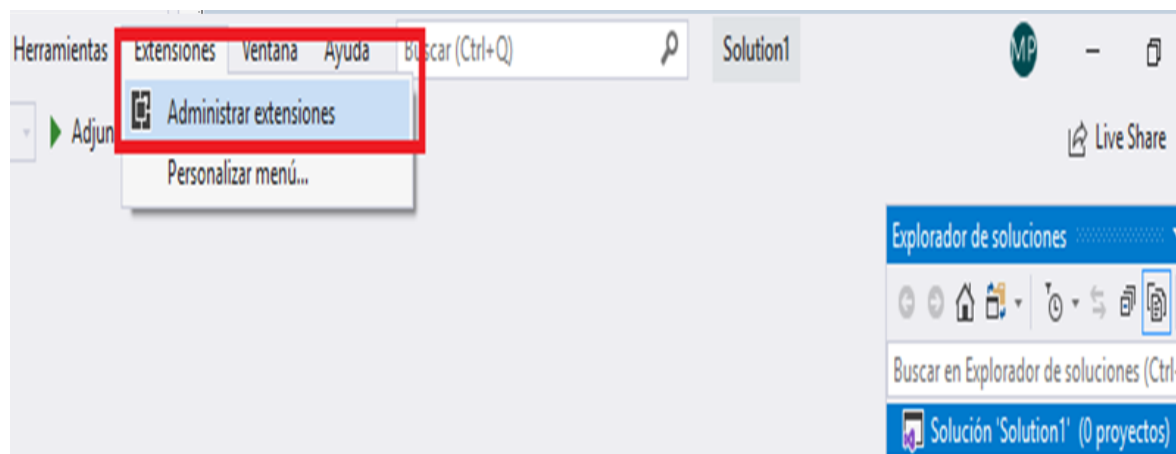
Para poder ver los reportes, se asocian al control ReportViewer, para ello arrastro el control reportviewer al form, y le asocio el archivo .rdlc.



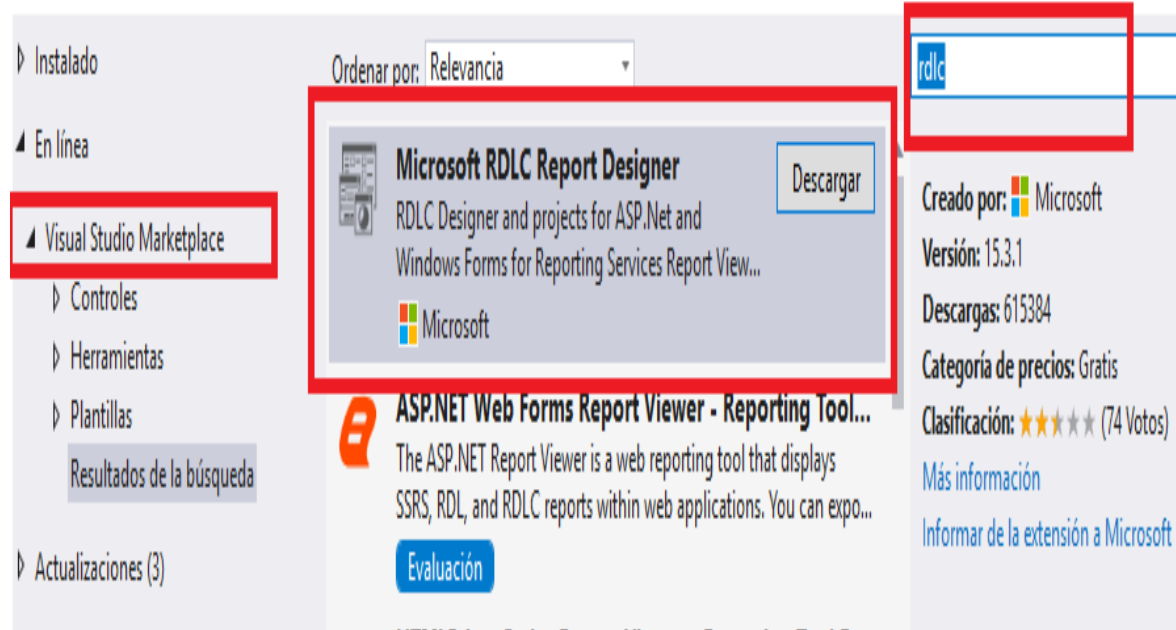


Para vs2017/19

Para el generador de reportes, tenemos que instalar:



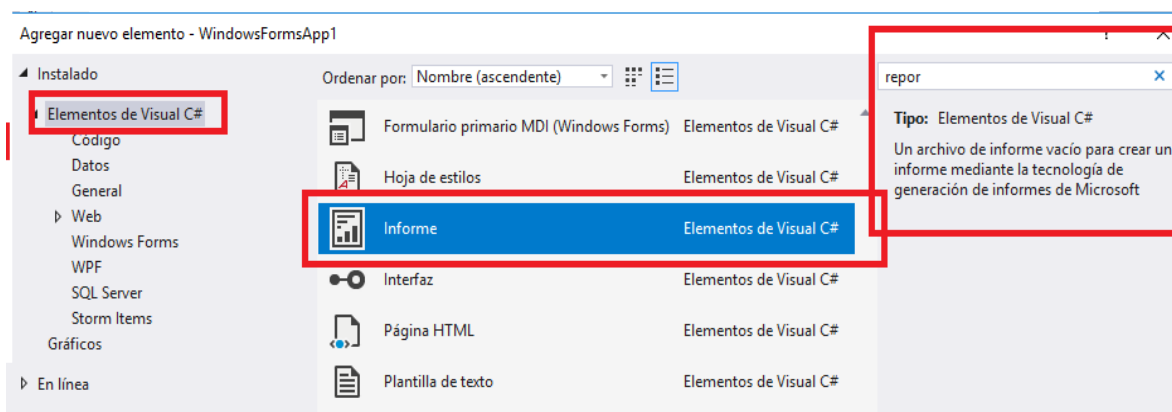
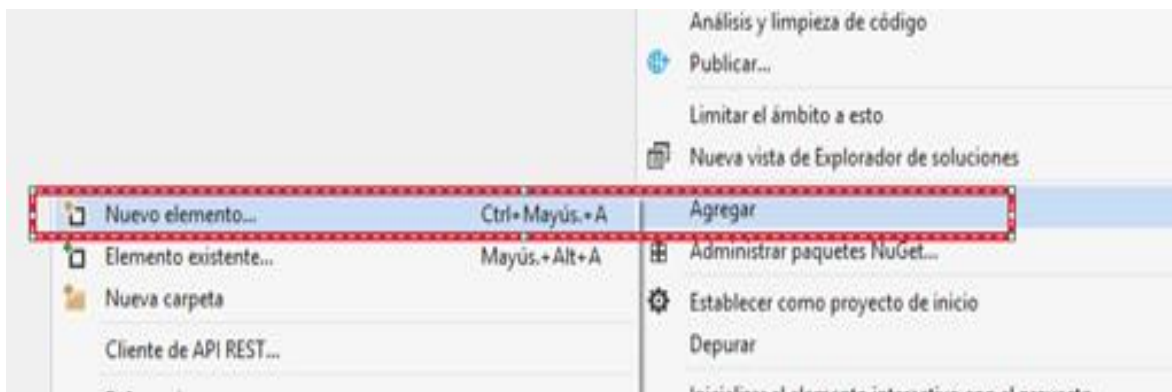
Administrar extensiones



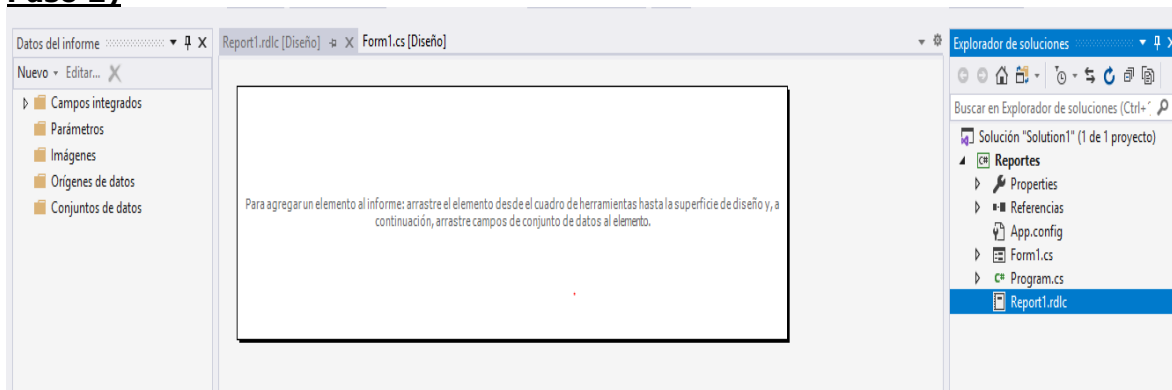
Paso a paso



1) Agrego el nuevo elemento para reportes, que es informes



Paso 2)

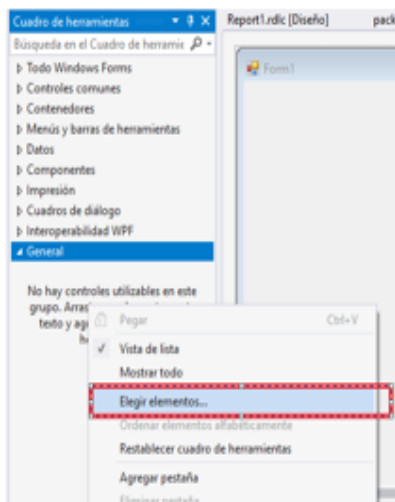


PASO 3) En el Nuget → se instala el paquete de ReportViewer, y para Win32 instalamos
→ Microsoft.ReportingServices.ReportViewerControl.WinForms



Paso 4)

- ✓ Vamos agregar el control a la caja de herramientas
- ✓ Boton derecho en la caja de herramientas y seleccionamos elegir elemento

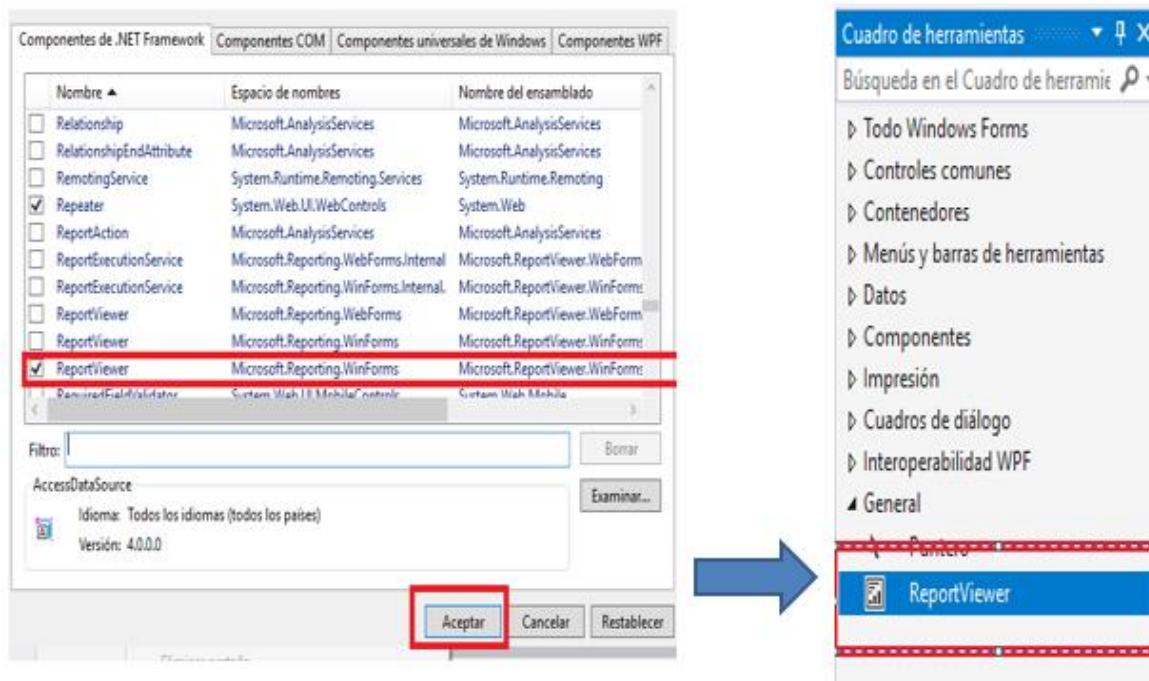


Voy a examinar y busco en el
package de instalación dentro del
proyecto

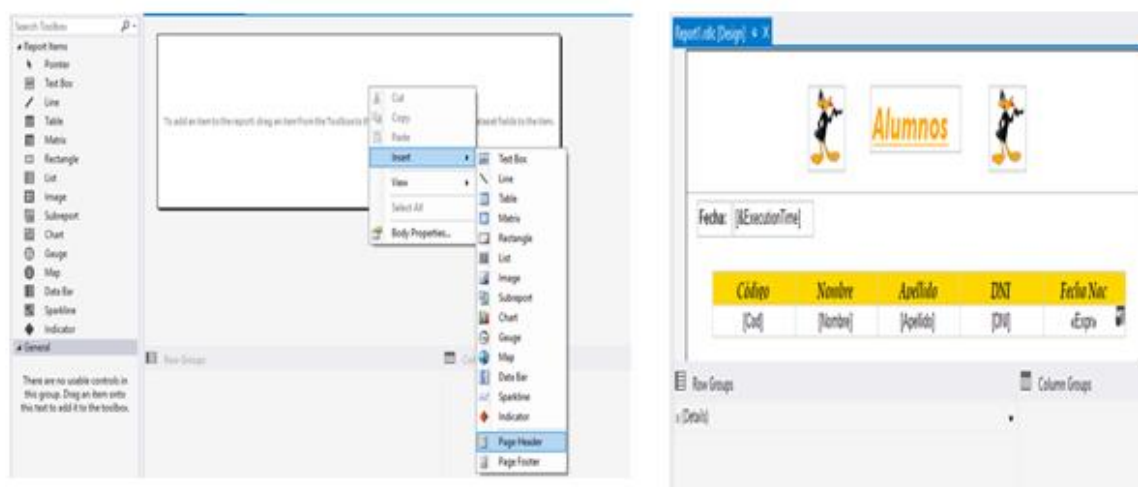
```
packages > Microsoft.ReportingServices.ReportViewerControl.WinForms.150.1404.0 > lib > net40
```

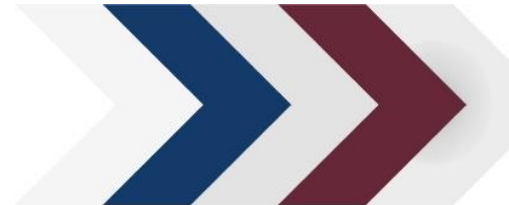
Y selecciono la DLL:
Microsoft.ReportViewer.WinForms.dll

Paso 5) Si no aparece lo agrego como componente

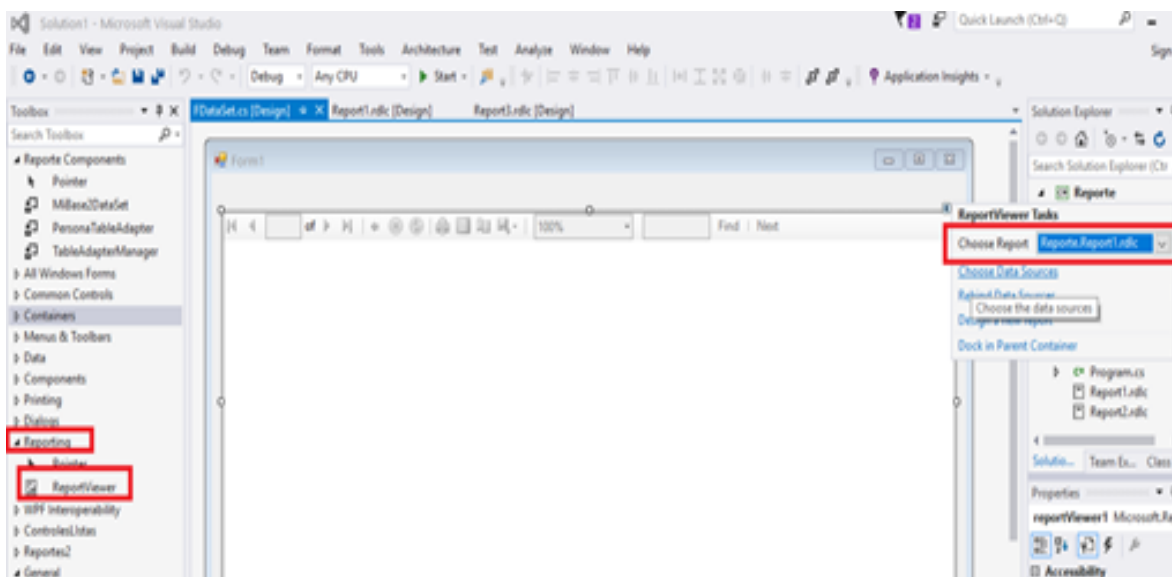


Luego se arma el diseño del informe como se desee: Encabezados, pie de página, gráficos, etc





Para poder ver los reportes en la versión 2019, se asocian al controlReportViewer, para ello arrastro el control reportviewer al form, y le asocio el archivo .rdlc.



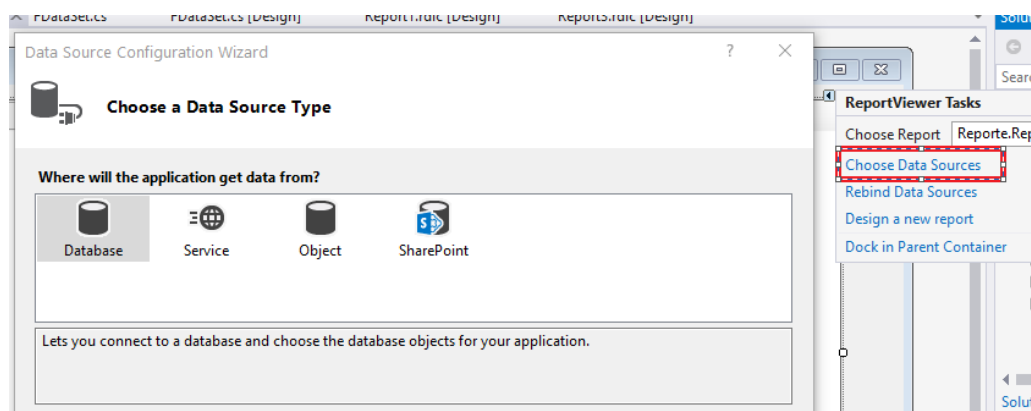
Luego selecciono el origen de datos que puede ser:

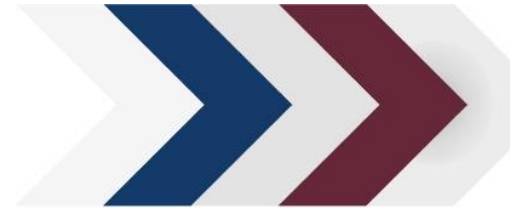
Database: Base de Datos, el mismo se manipula como un Dataset.

Service: Servicio para disponer de la información

Object: Objetos para vincular datos (listas)

Sharepoint: Le permite conectarse a un sitio de SharePoint y elegir los objetos de SharePoint para su aplicación.





Vamos a ver la opción de Dataset y Objetos.

- 1) Dataset, selecciono el origen de datos y el connectionString.

Data Source Configuration Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

MiBase2ConnectionString (Settings)

MiBaseConnectionString (Settings)

MiBase2ConnectionString (Settings)

t205033\sqlexpress.AgenciaA.dbo

t205033\sqlexpress.EmpresaCable2.dbo

☐ No, exclude sensitive data from the connection string. I will set this information in my application code.

☐ Yes, include sensitive data in the connection string.

☐ Connection string that you will save in the application (expand to see details)

Data Source=.\SQLEXPRESS;Initial Catalog=Mibase2;Integrated Security=True

< Previous Next > Finish Cancel

Mapeo la o las tablas que deseo mostrar en el informe.

Data Source Configuration Wizard

Choose Your Database Objects

Which database objects do you want in your dataset?

☒ Tables

☒ Persona

☒ Cod

☒ Nombre

☒ Apellido

☒ DNI

☒ Correo

☒ FechaNac

☒ Views

☒ Stored Procedures

☒ Functions

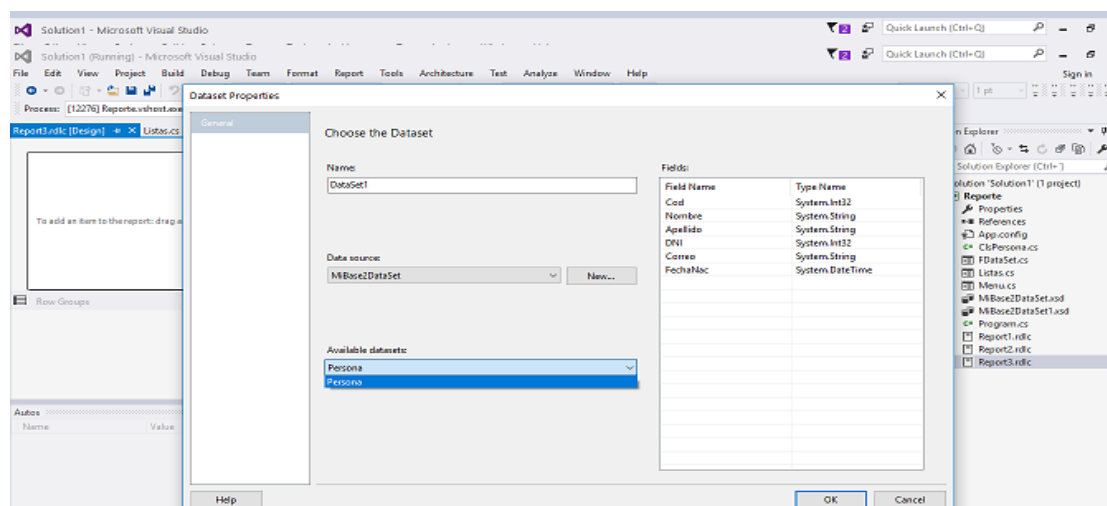
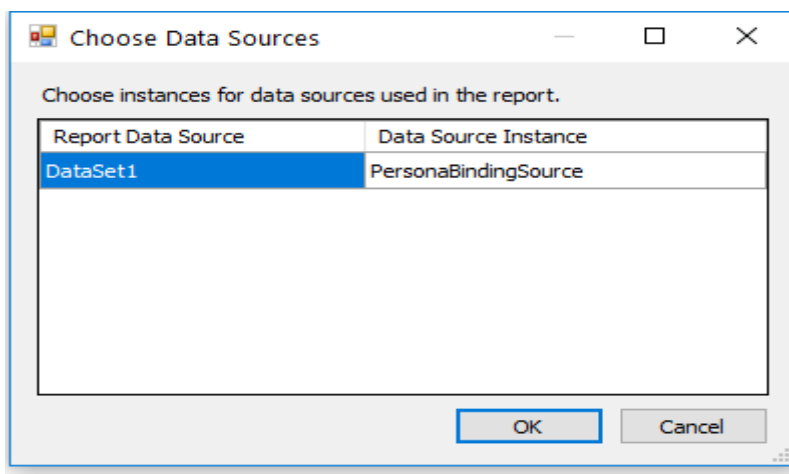
DataSet name:

Mibase2DataSet1

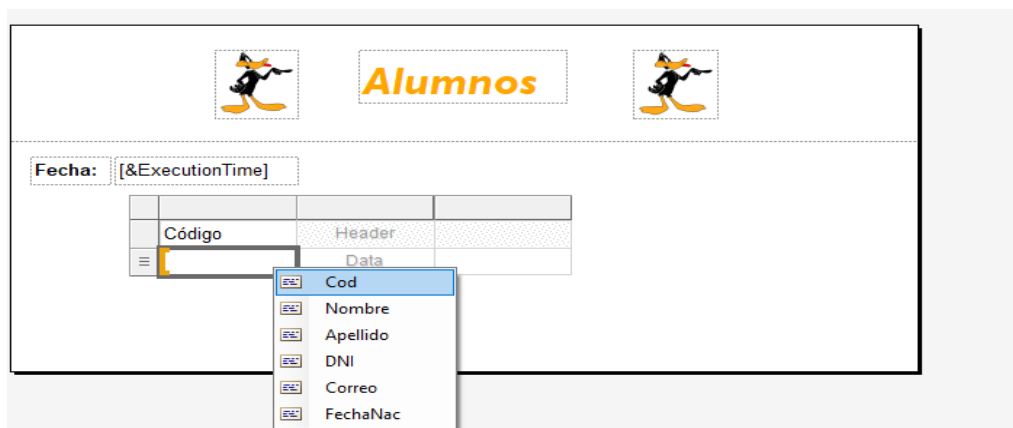
< Previous Next > Finish Cancel



Y me crea el origen de datos.

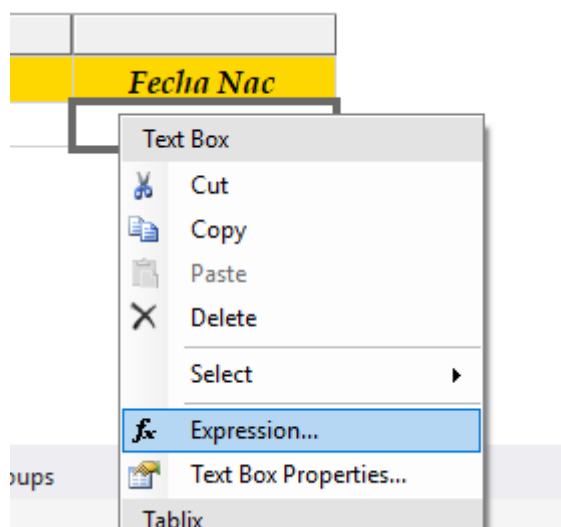


En la tabla agregada al .rdlc, asocio las propiedades a mostrar.

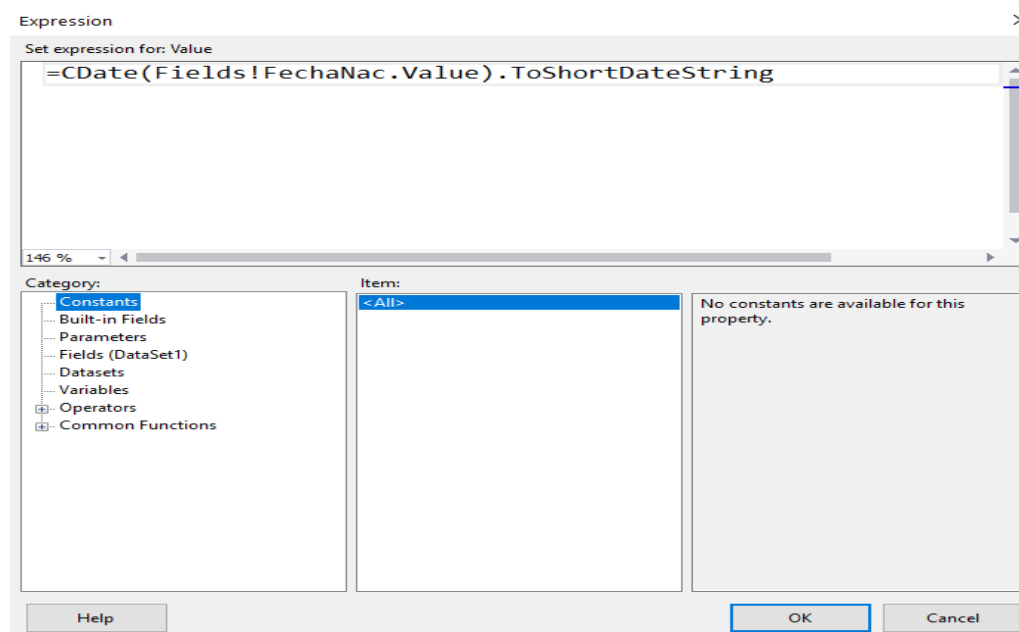




También puedo agregar expresiones.



Como la propiedad es del tipo Datatime, para mostrar la fecha solamente realice una expresión





Cuando muestro el informe, a nivel código lo asocia al datasource automáticamente.

```

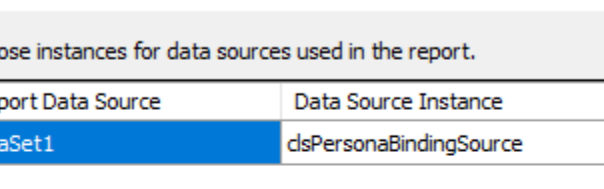
1 reference
private void Form1_Load(object sender, EventArgs e)
{
    // TODO: This line of code loads data into the 'MiBase2DataSet.Persona' table. You
    this.PersonaTableAdapter.Fill(this.MiBase2DataSet.Persona);

    this.reportViewer1.RefreshReport();
}

```

Código	Nombre	Apellido	DNI	Fecha Nac
1	Juan	Perez	2345678	12/12/1977
2	Lorena	Silvia	25890456	19/09/1977
3	Ana	Giunta	30234873	10/10/1980

En el caso de objetos, lo que realiza es el mapeo de la/las clases que requiero para el informe.

[illegible]

Choose Data Sources

Choose instances for data sources used in the report.

Report Data Source	Data Source Instance
DataSet1	clsPersonaBindingSource

OK Cancel





[&ExecutionTime]

Alumnos por Lista

	<i>Código</i>	<i>Nombre</i>	<i>Apellido</i>	<i>DNI</i>	<i>Fecha Nac</i>
☰	[Cod]	[Nombre]	[Apellido]	[DNI]	«Expr»

[&PageNumber] / [&TotalPages]

Al ser por objeto la expresión del tipo Datetime es distinta a la de la BD

Expression
✕

Set expression for: Value

```
=FormatDateTime(Fields!FechaNac.Value, DateFormat.ShortDate)
```

146 %

Category:

- Constants
- ... Built-in Fields
- ... Parameters
- ... Fields (DataSet1)
- ... Datasets
- ... Variables
- ⊕ Operators
- ⊕ Common Functions

Item:

<All>

No constants are available for this property.

Help
OK
Cancel



A nivel código:

```
1 reference
private void CargarReporte()
{
    //le paso el metodo donde se cargan las listas
    this.reportViewer1.LocalReport.DataSources[0].Value = CargarPersonas();
    this.reportViewer1.RefreshReport();
}
```

```
1 reference
private void Listas_Load(object sender, EventArgs e)
{
    this.reportViewer1.RefreshReport();
    CargarReporte();
}
```

```
private List<ClsPersona> CargarPersonas()
{
    List<ClsPersona> ListaPersona = new List<ClsPersona>();

    ClsPersona op1 = new ClsPersona(1, "Juan", "Perez", 20345678,
"juan@algo.com", Convert.ToDateTime("10/10/1980"));

    ClsPersona op2 = new ClsPersona(2, "Pedro", "Lopez", 27345678,
"pedrito@algo.com", Convert.ToDateTime("16/ 09 / 1989"));

    ClsPersona op3 = new ClsPersona(3, "Analia", "Sanchez", 33987459,
"anita@algo.com", Convert.ToDateTime("30/ 08 / 1999"));

    ClsPersona op4 = new ClsPersona(4, "Julieta", "Capuleto", 35113936,
"jula@algo.com", Convert.ToDateTime("11 / 11 / 2000"));

    ClsPersona op5 = new ClsPersona(5, "Romeo", "Montesco", 29085312,
"romeo@algo.com", Convert.ToDateTime("10 / 10 / 2007"));

    ClsPersona op6 = new ClsPersona(6, "Lorena", "Ramish", 28999742,
"lorena@algo.com", Convert.ToDateTime("13 / 04 / 1979"));

    ClsPersona op7 = new ClsPersona(7, "Sergio", "Redel", 39345678,
"sergio@algo.com", Convert.ToDateTime("10 / 01 / 2005"));

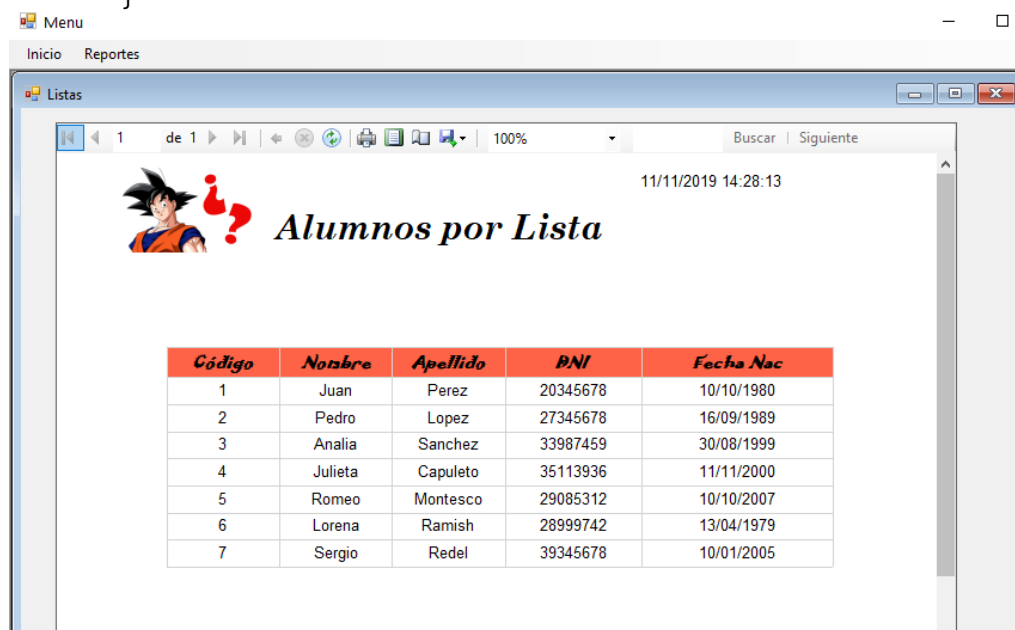
    ListaPersona.Add(op1);
    ListaPersona.Add(op2);
```



```
ListaPersona.Add(op3);
ListaPersona.Add(op4);
ListaPersona.Add(op5);
ListaPersona.Add(op6);
ListaPersona.Add(op7);
```

```
return ListaPersona;
```

```
}
```



5. CONCLUSIÓN

A medida que nos adentramos en el desarrollo de aplicaciones iremos notando que muchas veces las funcionalidades se repiten. Un buen programador se alerta de estas vivencias y toma cartas en el asunto. Crear recursos propios nos permite tener un mayor control y nos asegura aplicaciones escalables y seguras.

Le proponemos ampliar y profundizar los contenidos en la bibliografía de la materia.

