



# Benemérita Universidad Autónoma de Puebla

Materia: *Sistemas Operativos II*

Profesor: *Gilberto López Poblano*

Alumno: *Christian Amauri Amador Ortega - 201927821*

**- Exportar Excel -**

*Para esta serie de prácticas con código utilizamos el lenguaje C# en el IDE visual studio (la versión más reciente en septiembre de 2022). Puede haber variaciones en el funcionamiento de exactamente los mismos códigos pero con otra versión de visual studio u otro IDE. De todas formas estas variaciones deben de ser mínimas e insignificantes.*

*!!! Documentaremos el funcionamiento de las aplicaciones (en su ejecución) y haremos algunos comentarios / notas / consideraciones / advertencias al respecto. Para ahorrar tiempo y esfuerzo, trataremos de no ahondar demasiado en la estructura del código, sino simplemente explicar brevemente cómo funciona **(se asume que el lector tiene ya conocimientos suficientes sobre conceptos varios de programación estructurada, ambiente gráfico, variables, datos, y demás...)***

*Finalmente hay que tomar en cuenta que C# y Visual studio son herramientas pesadas y caprichosas. Y que el equipo que vayamos a utilizar debe tener cierta capacidad de rendimiento superior a sólo básica.*

Primero, el objetivo del proyecto es crear un programa en C# con el que podamos ingresar datos a un dataGridView, y luego poder depositarlos en un archivo tipo hoja de cálculo de Excel (.xls)

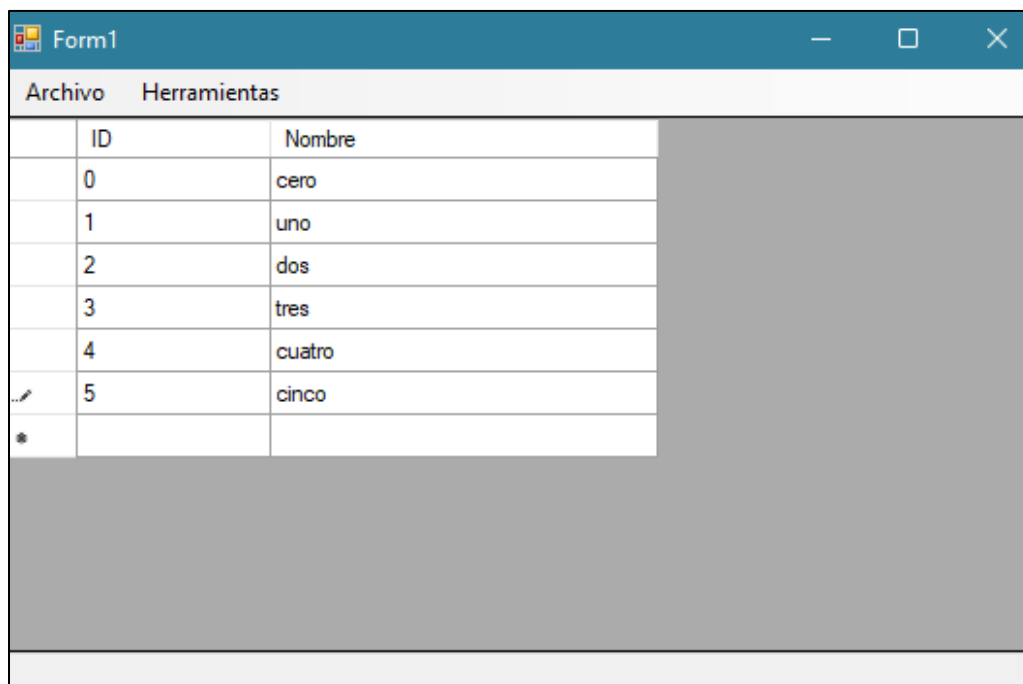
En cualquier caso, el repositorio en el que este código se encuentra (y otros códigos en C# del curso de Sistemas operativos 2, BUAP FCC, Otoño 2022) está disponible haciendo click en el siguiente enlace:

<https://drive.google.com/drive/folders/1yZujMI51XAnEZBMj-4ykUWMfHXBC9uYp?usp=sharing>

Y estará disponible al menos durante el resto de 2022. Gracias. 😊

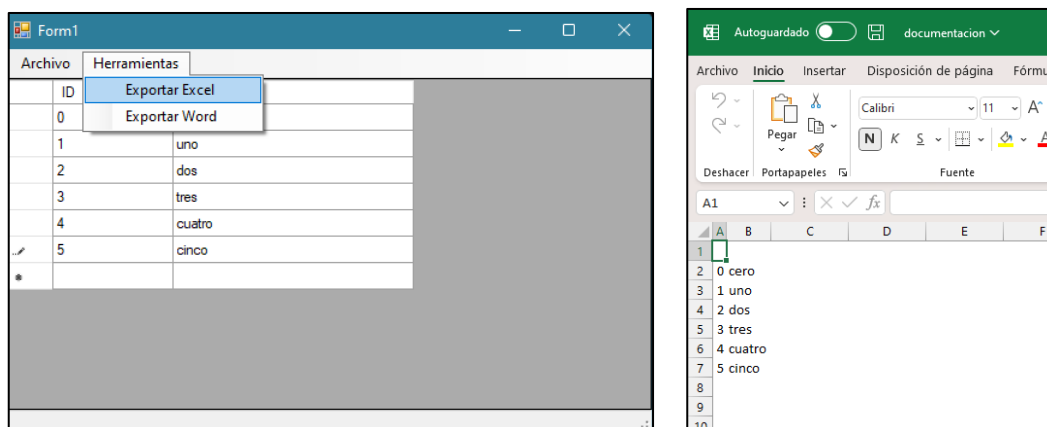
## Ejemplo rápido del funcionamiento general del proyecto, para resumir:

- Podemos registrar datos en un dataGridView:



ID	Nombre
0	cero
1	uno
2	dos
3	tres
4	cuatro
5	cinco

- Mandarlos a un archivo tipo hoja de cálculo de Excel (.xls)



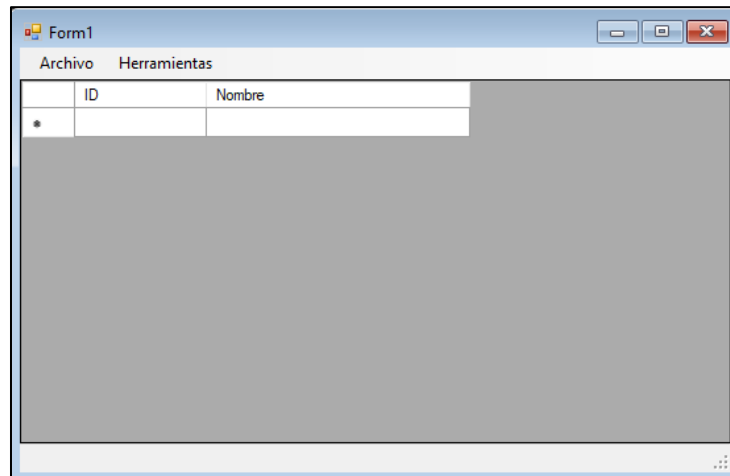
Left screenshot: 'Form1' DataGridView with 'Herramientas' menu open, showing 'Exportar Excel' and 'Exportar Word' options.

Right screenshot: Microsoft Excel spreadsheet showing the data from the DataGridView pasted into cells A2 through A7.

	A	B	C	D	E	F
1						
2	0					
3	1					
4	2					
5	3					
6	4					
7	5					
8						
9						
10						

Ahora, los archivos .cs del proyecto (que nosotros creamos) corresponden a un diseño y un script. Comencemos por el diseño...

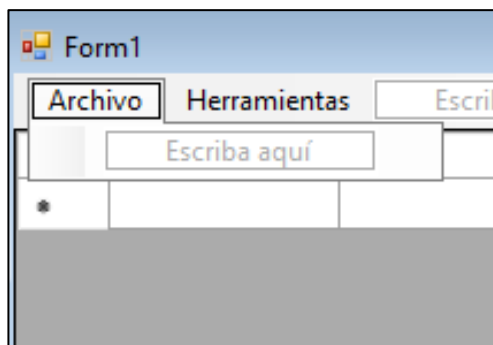
## - Diseño -



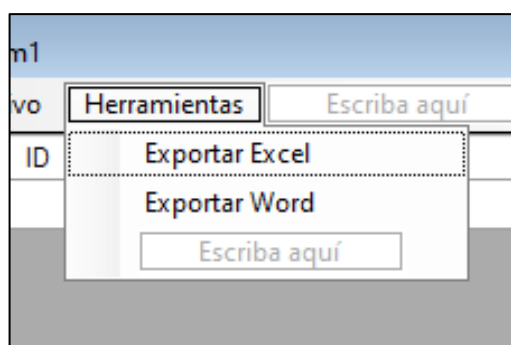
The screenshot shows a Windows-style window titled 'Form1'. It has a menu bar with two items: 'Archivo' and 'Herramientas'. Below the menu bar is a data grid. The grid has two columns: 'ID' and 'Nombre'. The first row of the grid contains a '\*' in the 'ID' column and is empty in the 'Nombre' column. The rest of the grid is shaded gray, indicating it is not visible or is a placeholder.

**Form1** (Form main)

**Contiene el panel en el que se encuentra el dataGridView en el que depositaremos los datos a exportar.** Contiene también los apartados “Archivo” y “Herramientas”, en este caso, “Archivo” se encuentra deshabilitado (no tiene ninguna funcionalidad). “Herramientas” tiene las opciones: “Exportar excel” y “ Exportar word”. “Exportar word” también se encuentra deshabilitado:



This screenshot shows the 'Form1' window with the 'Archivo' menu item highlighted. The 'Herramientas' menu item is also visible. Below the menu bar, there is a text box labeled 'Escriba aquí' and a data grid with columns 'ID' and 'Nombre'. The grid is partially visible, showing a '\*' in the 'ID' column.



This screenshot shows the 'Form1' window with the 'Herramientas' menu item highlighted. A dropdown menu is open, showing two options: 'Exportar Excel' and 'Exportar Word'. Below the dropdown menu, there is a text box labeled 'Escriba aquí' and a data grid with columns 'ID' and 'Nombre'. The grid is partially visible, showing a '\*' in the 'ID' column.

## - Script -

### Form1.cs

La estructura inicial de este script consta de:

- 10 librerías
- namespace definido como: \_06\_Exportar\_11\_Oct\_
- public partial class Form1 : Form

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.IO;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10 using System.Windows.Forms;
11
12 namespace _06_Exportar_11_Oct_
13 {
14     public partial class Form1 : Form
15     {
```

Dentro de Form1 : Form. se encuentran los siguientes componentes:

```
14     public partial class Form1 : Form
15     {
16         public Form1()
20
21         private void exportarExcelToolStripMenuItem_Click(object sender, EventArgs e)
34
35         public String ConvertToExcel()
65     }
```

- un constructor
- 2 métodos

El método `exportarExcelToolStripMenuItem_Click()` corresponde a la acción implementada por la opción “Exportar Excel” de la opción “Herramientas” de la ventana principal. Esta aplica un filtro para guardar el archivo correspondiente en formato estrictamente (.xls). Luego simplemente manda a exportar los datos del `dataGridView`

```
20
21 1 referencia
22 private void exportarExcelToolStripMenuItem_Click(object sender, EventArgs e)
23 {
24     saveFileDialog1.Filter = "Excel (*.xls)|.xls";
25
26     if (saveFileDialog1.ShowDialog() == DialogResult.OK)
27     {
28         String archivo = saveFileDialog1.FileName;
29         StreamWriter sw = new StreamWriter(archivo, false);
30         sw.WriteLine(ConvertoExcel());
31         sw.Close();
32     }
33     ConvertoExcel();
34 }
```

El método `ConvertoExcel()` define una variable tipo `String` que contiene los datos del `dataGridView` que queremos exportar a Excel, depositados dentro del formato que Excel puede leer (si nos damos cuenta; con cada concatenación que se hace a la variable `line`, se va agregando un fragmento de formato. Luego entramos a un `for` en el que hacemos lo mismo, pero agregando los datos a exportar, tupla por tupla). La variable `i` corresponde al iterador para contar las tuplas, `Cells[0]` corresponde al apartado “ID” del `dataGridView`. `Cells[1]` corresponde a “Nombre”.

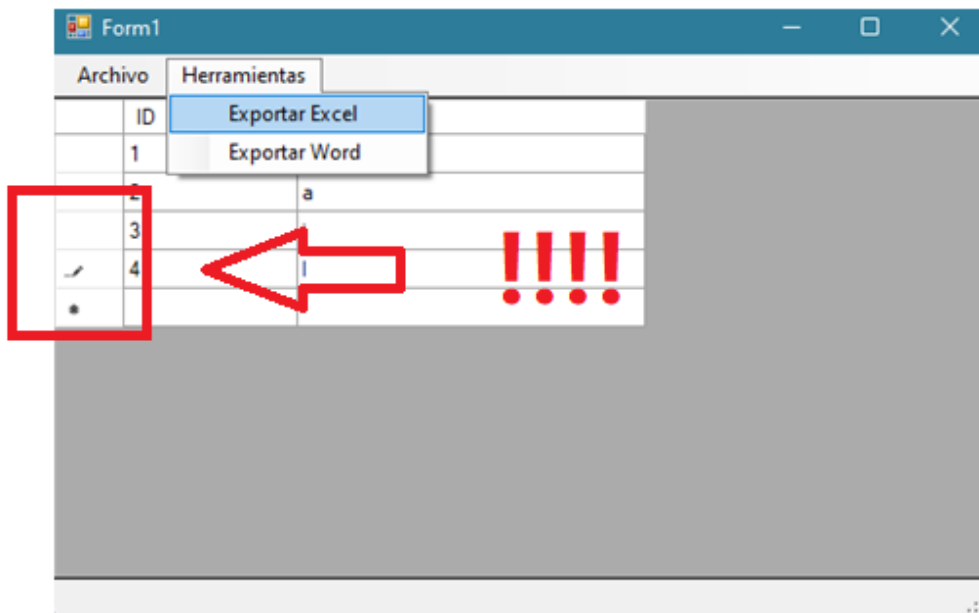
Finalmente se `devuelve` la línea construida en el proceso:

```
35 2 referencias
36 public String ConvertoExcel()
37 {
38     String line = "<table>";
39     line += "<tr>";
40     line += "<th>";
41     line += "<ID>";
42     line += "</th>";
43     line += "<th>";
44     line += "<Nombre>";
45     line += "</th>";
46     line += "</tr>";
47
48     for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)
49     {
50         line += "<tr>";
51         line += "<td>";
52         line += dataGridView1.Rows[i].Cells[0].Value.ToString();
53         line += "</td>";
54         line += "<td>";
55         line += dataGridView1.Rows[i].Cells[1].Value.ToString();
56         line += "</td>";
57         line += "</tr>";
58     }
59     line += "</table>";
60     return line;
61 }
```

## Como comentario final!!! :

Hay que tener cuidado en la ejecución a la hora de guardar los datos ingresados en la tabla. Hay que desactivar el ícono de lápiz, pues si este está activo, saltará un error (excepción no capturada) pero esto es algo que con cuidado, podemos ignorar:

Esto:



Provoca:

