Descripción de DataGrid y LINQ en C# (WPF)

Información general sobre DataGrid y LINQ

¿Qué es un DataGrid?

- Un DataGrid en XAML (WPF) es un control que se usa para mostrar y administrar datos en un formato de tabla, similar a Excel.
- Permite a los usuarios:
  - Ver varios registros al mismo tiempo
  - Edite, agregue o elimine registros directamente desde la interfaz
  - Enlazar a colecciones de datos como listas o bases de datos

Función principal

- La función principal de un DataGrid es presentar datos estructurados y permitir que los usuarios interactúen con ellos.
- Por ejemplo, puede mostrar una lista de usuarios (nombre, edad, correo electrónico) y permitir agregarlos o eliminarlos mediante botones o eventos.

¿Por qué es importante?

- DataGrid actúa como puente entre la interfaz de usuario y los datos reales
  - Ayuda a organizar y visualizar datos fácilmente
  - Admite el enlace de datos
  - Prepara a los estudiantes para mostrar los resultados de las consultas de la base de datos dentro de una aplicación.

Conexión con la unidad 3

- Hasta ahora, el DataGrid se llenaba con datos en memoria (como una lista<Persona>).
- En la siguiente unidad, esos registros provendrán de una base de datos real: DataGrid mostrará los resultados de la consulta y permitirá actualizaciones o eliminaciones de tablas.

Actividad de revisión: "El DataGrid y los datos". Preguntas de comprensión

1. ¿Qué tipo de información se muestra en un DataGrid?
2. Explique brevemente la diferencia entre un TextBox y un DataGrid.
3. ¿Qué acciones puede realizar dentro de un DataGrid?
4. ¿De dónde vinieron nuestros datos de DataGrid antes y de dónde vendrán después?
5. ¿Por qué es útil DataGrid cuando se trabaja con bases de datos?

Actividad de revisión: "El DataGrid y los datos". Preguntas de comprensión

1. ¿Qué tipo de información se muestra en un DataGrid? Datos organizados en filas y columnas
2. Explique brevemente la diferencia entre un TextBox y un DataGrid. Un TextBox muestra un solo fragmento de datos; un DataGrid muestra muchos registros a la vez.
3. ¿Qué acciones puede realizar dentro de un DataGrid? Agregar, editar, eliminar o ver registros
4. ¿De dónde vinieron nuestros datos de DataGrid antes y de dónde vendrán después? De listas u objetos en memoria → de tablas de bases de datos en la siguiente unidad.
5. ¿Por qué es útil DataGrid cuando se trabaja con bases de datos? Porque permite a los usuarios ver y manipular la información de la base de datos de una manera visual fácil.

Mini Práctica

```
<DataGrid x:Name="UserDataGrid" AutoGenerateColumns="True" />
```

- ¿Qué pasaría si llenáramos este DataGrid con una lista de objetos “User”?
- ¿Y si esos usuarios provienen de una tabla de base de datos llamada Usuarios?

Mini Práctica

```
<DataGrid x:Name="UserDataGrid" AutoGenerateColumns="True" />
```

- ¿Qué pasaría si llenáramos este DataGrid con una lista de objetos "User"?

DataGrid mostraría automáticamente cada "Usuario" como una fila, con una columna por propiedad (por ejemplo: Nombre, Edad, Correo electrónico).

Cada elemento de la lista se convierte en una fila en DataGrid y cada propiedad del objeto se convierte en una columna. Muestra los datos almacenados en la memoria (por ejemplo, una lista<User> creada en el programa).

Mini Práctica

```
<DataGrid x:Name="UserDataGrid" AutoGenerateColumns="True" />
```

- ¿Y si esos usuarios provienen de una tabla de base de datos llamada Usuarios?

El DataGrid mostraría el mismo tipo de información, pero ahora los datos provendrían de la base de datos: datos reales almacenados en lugar de objetos temporales en memoria.

A DataGrid no le importa de dónde provienen los datos (memoria, archivo o base de datos), siempre que estén vinculados a una fuente de datos.

Se convierte en una ventana para mostrar e interactuar con los datos almacenados permanentemente en la base de datos.

Resumen

- DataGrid nos ayudó a aprender a mostrar y administrar colecciones de datos en el interfaz. En la siguiente unidad, conectaremos esos elementos visuales a una base de datos real mediante consultas y enlace de datos.

LINQ y su conexión a DataGrid

❖ ¿Qué es LINQ?

LINQ (Language Integrated Query) le permite consultar y manipular datos directamente en C#, de forma similar a SQL

❖ ¿Por qué es útil?

- Filtra, ordena y selecciona datos fácilmente
- Reemplaza los bucles largos por consultas cortas y expresivas
- Funciona con: Listas en memoria, ficheros XML o Bases de Datos (Linq to SQL o Entity Framework).

Ejemplo: LINQ con una lista

```
List<User> users = new List<User>
{
    new User { Name = "Alice", Age = 25 },
    new User { Name = "Bob", Age = 30 },
    new User { Name = "Charlie", Age = 22 }
};

// LINQ query
var filteredUsers = from u in users
                    where u.Age > 24
                    select u;

UserDataGrid.ItemsSource = filteredUsers.ToList();
```

Ejemplo: LINQ con una lista

```
List<User> users = new List<User>
{
    new User { Name = "Alice", Age = 25 },
    new User { Name = "Bob", Age = 30 },
    new User { Name = "Charlie", Age = 22 }
};

// LINQ query
var filteredUsers = from u in users
                    where u.Age > 24
                    select u;

UserDataGrid.ItemsSource = filteredUsers.ToList();
```

Database → LINQ Query → DataGrid

LINQ con bases de datos

- Uso de LINQ to SQL o Entity Framework:

```
var users = from u in db.Users
            where u.Age > 24
            select u;

UserDataGrid.ItemsSource = users.ToList();
```

LINQ actúa como puente entre el código de C# y las consultas de base de datos. DataGrid es la parte visual que muestra esos resultados.