WPF (WINDOWS PRESENTATION FOUNDATION)

Entrar en el mundo de **WPF (Windows Presentation Foundation)** es como pasar de pintar con los dedos a usar Photoshop. La clave de todo es **XAML** (Extensible Application Markup Language).

Piénsalo así: si tu aplicación fuera un cuerpo, el **C# es el cerebro** (la lógica) y el **XAML es la cara** (la interfaz).

¡Claro que sí! Vamos a construir tu conocimiento de WPF desde los cimientos. Olvida las interfaces antiguas; aquí vamos a separar el "cómo se ve" del "qué hace".

Esta es tu hoja de ruta para pasar de cero a tener una aplicación funcional.

Fase 1: El Entorno y el Proyecto

Para empezar, necesitas **Visual Studio** (la versión Community es gratuita).

1. Abre Visual Studio y selecciona **"Crear un nuevo proyecto"**.
2. Busca **"Aplicación WPF"** (asegúrate de que diga C#).
3. Elige un nombre y selecciona la versión de **.NET** más reciente (como .NET 8).

Al abrirlo, verás dos archivos principales:

- `MainWindow.xaml`: Donde "dibujas" con código XML.
- `MainWindow.xaml.cs`: Donde escribes la lógica en C#.

Fase 2: Estructura del XAML

Todo en XAML son etiquetas anidadas. La jerarquía típica es:

`Window > Contenedor (Layout) > Controles (Botones, Textos).`

El "Grid": Tu mejor amigo

Por defecto, WPF te da un `Grid`. Piensa en él como una red invisible.

XML

```
<Window x:Class="MiApp.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2000/xaml/presentation"
        Title="Mi Primera App" Height="300" Width="400">

    <Grid>
```

```
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="*"/>
            </Grid.RowDefinitions>
            <TextBlock Text="¡Hola, WPF!" FontSize="20" HorizontalAlignment="Center" Grid.Row="0"/>
            <Button Content="Presióname" Grid.Row="1" Width="100" Height="40" Click="Button_Click"/>
        </Grid>
    </Window>
```

Fase 3: El "Code-Behind" (C#)

Ahora vamos a darle vida a ese botón. En el XAML, pusimos `Click="Button_Click"`. Si haces doble clic en el botón en el diseñador o escribes el nombre manualmente en el archivo `.cs`, verás esto:

C#

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    // Cambiamos el título de la ventana al hacer clic
    this.Title = "¡Funciona!";
    MessageBox.Show("Has interactuado con el botón.");
}
```

Fase 4: Propiedades Fundamentales

Para que no te vuelvas loco con el diseño, memoriza estas 4 propiedades:

1. **`x:Name`**: Es el ID. Si le pones `x:Name="MiCaja"`, puedes usar `MiCaja.Text` en C#.
2. **`Margin`**: Espacio exterior (Izquierda, Arriba, Derecha, Abajo). Ejemplo: `Margin="10,20,10,20"`.
3. **`HorizontalAlignment` / `VerticalAlignment`**: Para centrar o pegar elementos a los bordes.

4. **`Padding`**: Espacio interior (dentro del control).

Fase 5: Conceptos Avanzados (El "Siguiente Nivel")

Una vez domines lo anterior, estos son los pilares de un experto en WPF:

- **Data Binding:** En lugar de hacer `txtNombre.Text = "Juan"`, conectas la propiedad directamente a una variable.
- **MVVM (Model-View-ViewModel):** Es el patrón de diseño estándar. Separa totalmente la interfaz de la lógica para que el código sea limpio y fácil de probar.
- **Resources & Styles:** Como el CSS. Defines cómo se ve un botón una vez y se aplica a todos los de la app.

Tu primera tarea práctica

Intenta crear una ventana que tenga:

1. Un `StackPanel` (para que los elementos se apilen solos).
2. Un `TextBox` con un nombre (`x:Name="txtUsuario"`).
3. Un `Button` que, al pulsarlo, muestre un `MessageBox` diciendo "Hola" + lo que escribiste en el cuadro de texto.

Aquí tienes un desglose rápido para que domines los conceptos básicos:

1. ¿Qué es exactamente XAML?

Es un lenguaje basado en XML que sirve para definir la interfaz de usuario de forma declarativa. En lugar de escribir 20 líneas de código C# para crear un botón, escribes una sola etiqueta en XAML.

Ejemplo básico:

XML

```
<Button Content="¡Haz clic aquí!"
        Width="120"
        Height="40"
        Background="LightBlue"
        Click="Button_Click" />
```

2. La relación con el "Code-Behind"

Cada archivo `.xaml` tiene un archivo "hermano" `.xaml.cs`.

- **XAML:** Define que el botón es azul.
- **C#:** Define qué pasa cuando presionas el botón (el evento `Click`).

3. Los Contenedores (Layouts)

En WPF no "dibujas" los elementos en coordenadas exactas (como se hacía en WinForms). Usas contenedores que organizan todo automáticamente:

Contenedor	Función
Grid	El más flexible. Organiza por filas y columnas (como una tabla de Excel).
StackPanel	Apila elementos uno tras otro, ya sea vertical u horizontalmente.
DockPanel	"Pega" los elementos a los bordes (arriba, abajo, izquierda, derecha).
WrapPanel	Coloca elementos en línea y salta a la siguiente cuando se acaba el espacio.

4. Un ejemplo real (Grid)

Si quisieras un formulario sencillo con una etiqueta y un cuadro de texto, se vería algo así:

XML

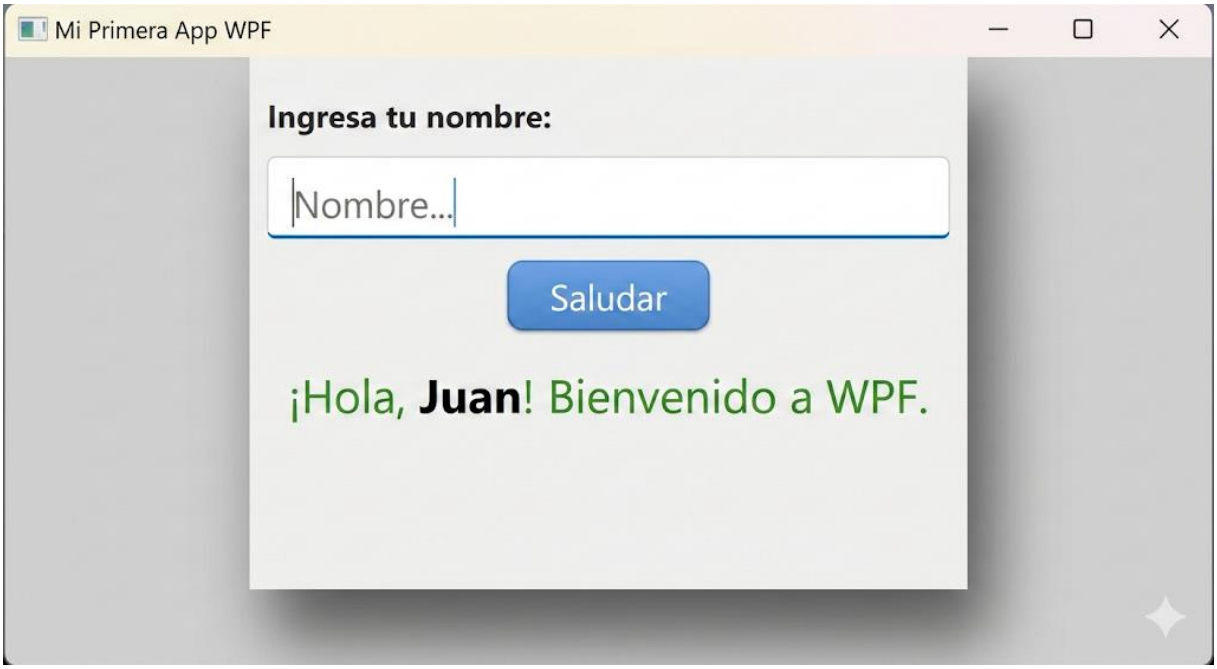
```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>

    <TextBlock Text="Nombre:" Grid.Row="0" Margin="10"/>
    <TextBox  x:Name="txtNombre"  Grid.Row="1"  Margin="10"
VerticalAlignment="Top"/>
```

```
</Grid>
```

Nota: El `Height="*"` significa "toma todo el espacio que sobre". Es super útil para interfaces responsivas.

5. ¿Por qué es mejor que lo antiguo?

1. **Separación de intereses:** El diseñador puede tocar el XAML mientras el programador escribe el C# sin estorbarse.
2. **Data Binding:** Puedes conectar los datos de tu base de datos directamente a la interfaz casi sin código.
3. **Estilos y Plantillas:** Puedes hacer que todos tus botones se vean iguales con un solo "Estilo", parecido a CSS en la web.



Descripción de la Imagen (Lo que estás viendo)

Esta imagen generada representa fielmente cómo el motor de renderizado de WPF dibuja los controles que definimos. Puedes notar:

1. **La Ventana (`Window`):** Tiene el título "Mi Primera App WPF" y los bordes estándar de Windows.
2. **El Diseño (`StackPanel`):** Todos los elementos están alineados verticalmente, uno debajo del otro, ocupando el ancho disponible.
3. **Los Controles:**
   - Ves la etiqueta (`TextBlock`) que dice "Ingresa tu nombre:".
   - Ves el cuadro de entrada (`TextBox`) donde el usuario escribiría.
   - Ves el botón azul (`Button`) con el texto "Saludar".
4. **La Interacción (Simulada):** En este render, simulamos que el usuario ya escribió "Juan" y presionó el botón. Por eso aparece el mensaje de bienvenida verde debajo, el cual programaríamos para que apareciera solo después del clic.

Este es el código XAML que generaría una interfaz muy similar a la de la imagen:

```
<Window x:Class="MiApp.MainWindow"
      xmlns="http://schemas.microsoft.com/winfx/2000/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2000/xaml"
      Title="Mi Primera App WPF" Height="350" Width="525">
      <StackPanel Margin="20">
      <TextBlock Text="Ingresa tu nombre:" FontSize="16" Margin="0,0,0,10"/>
       <TextBox x:Name="txtUsuario" FontSize="16" Padding="5" Margin="0,0,0,15"/>
       <Button Content="Saludar" FontSize="16" Padding="10,5"
              HorizontalAlignment="Left" Background="#FF2196F3" Foreground="White"/>
        <TextBlock x:Name="lblResultado" Text="¡Hola, Juan! Bienvenido a WPF."
               FontSize="18" Foreground="Green" FontWeight="Bold"
               Margin="0,20,0,0" HorizontalAlignment="Center"/>
   </StackPanel>
</Window>
```