PACTÓMETRO: Memoria

Manual de usuario

Índice:

1. Introducción:

Bienvenido al manual de usuario de "El Pactómetro", una aplicación diseñada por Manipulaciones y Bulos S.A., enfocada en transformar la manera en que se visualizan y gestionan los resultados de procesos electorales en España. Esta herramienta está orientada a facilitar la presentación y análisis de datos en elecciones autonómicas y generales, es una plataforma intuitiva y eficaz para el manejo de datos electorales.

Este manual guiará a los usuarios a través de las diversas funcionalidades de "El Pactómetro", asegurando una comprensión completa de cómo aprovechar al máximo esta herramienta para el análisis y la presentación de resultados electorales. A lo largo de estas páginas, encontrará instrucciones detalladas, consejos útiles y recomendaciones para optimizar su experiencia con la aplicación.

2. Requisitos del sistema

Para garantizar una experiencia óptima y eficiente al utilizar "El Pactómetro", es esencial que el sistema del usuario cumpla con el siguiente requisito fundamental:

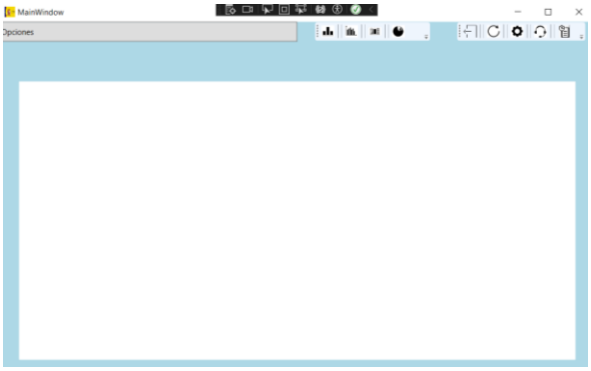
La Instalación de Visual Studio Code con Extensión .NET, el usuario debe tener instalado en su equipo el entorno de desarrollo integrado (IDE) Visual Studio Code. Es fundamental que, dentro de Visual Studio Code, se haya añadido la extensión correspondiente para .NET. Esta configuración es crucial para asegurar la compatibilidad total con las funcionalidades avanzadas de "El Pactómetro".

No se requieren especificaciones de hardware o sistema operativo adicionales, siempre y cuando se cumpla con el requisito anterior.

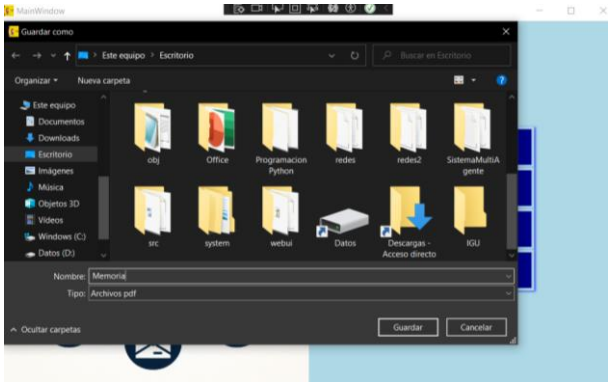
3. Inicio rápido

Al ejecutar "El Pactómetro", el usuario será recibido por la interfaz inicial del programa, diseñada para proporcionar un acceso rápido y sencillo a las principales funciones. Esta pantalla de inicio está organizada para facilitar una navegación intuitiva, ofreciendo las siguientes opciones:

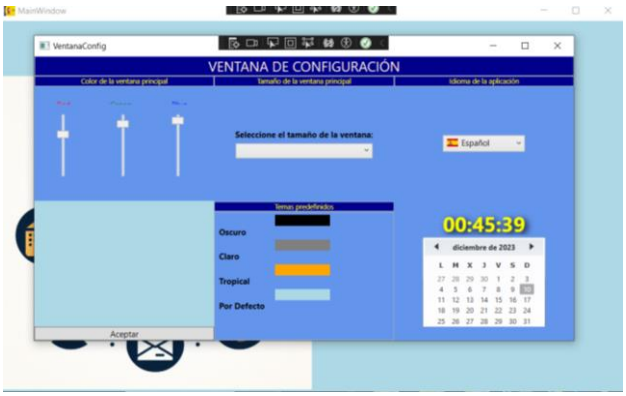
- **INICIO**: Permite al usuario iniciar el programa. Mostrándole la apariencia fundamental de la ventana principal. Esta es la ventana en la que se representarán todas las gráficas y resultados electorales.



- **AYUDA**: Permite al usuario descargar la memoria del programa "El Pactómetro", que a su vez contiene el manual de usuario.



- **SALIR**: Permite al usuario salir del programa.
- **CONFIGURACIÓN**: Permite al usuario acceder a la ventana de configuración, donde podrá modificar el color de fondo, el tamaño, el idioma y el tema de la ventana principal.

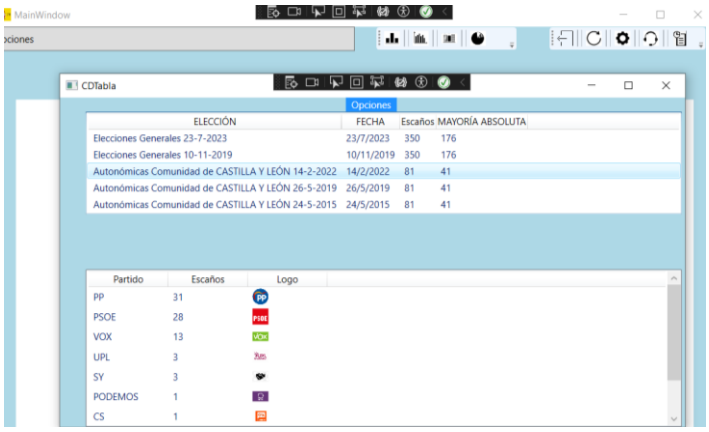


*Los cambios realizados en la ventana de configuración únicamente afectarán a la ventana principal, y no a las demás.

4. Navegación Básica

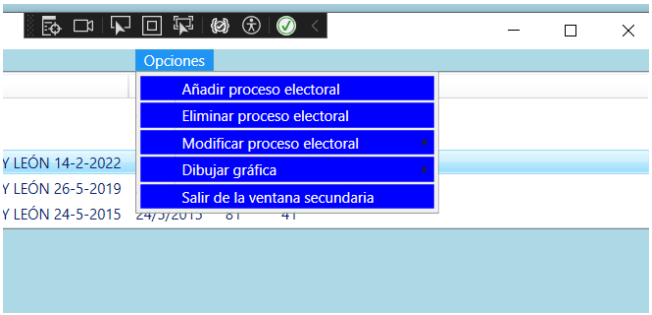
Los datos y la información recopilada relacionada con los procesos electorales se encuentran registradas en la ventana secundaria “CDTabla”, en esta el usuario podrá interactuar con una tabla en la que aparecerán los procesos almacenados en dicho momento. La ventana "CDTabla" de "El Pactómetro" está pensada para ser un centro de información y análisis completo, permitiendo al usuario no solo visualizar los datos de los procesos electorales, sino también interactuar con ellos de manera eficiente. Con esta herramienta, los usuarios podrán llevar a cabo una gestión y análisis exhaustivos de los datos electorales, fundamentales para una cobertura informativa precisa y detallada.

Además, al pulsar sobre uno de esos procesos, se actualizará la segunda tabla en la que aparecerán los partidos participantes en ese proceso, así como sus escaños conseguidos e información adicional.



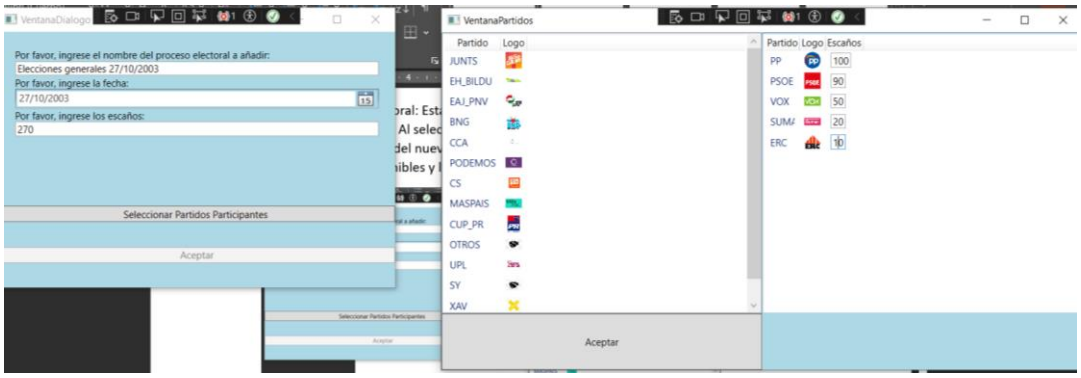
5. Gestión de Procesos Electorales

La gestión eficiente de procesos electorales es una funcionalidad clave. El usuario puede acceder a esta funcionalidad a través de la ventana "CDTabla", utilizando el botón "Opciones". Una vez dentro, se presentarán diversas alternativas para manejar los procesos electorales de forma detallada y personalizada.



Opciones Disponibles:

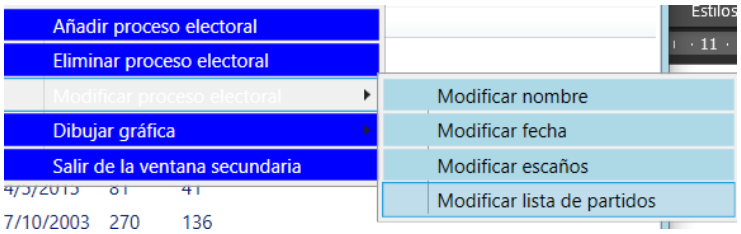
a) ***Agregar proceso electoral***: Esta opción permite al usuario introducir un nuevo proceso electoral en el sistema. Al seleccionarla, se abrirá una interfaz donde se podrán ingresar todos los detalles relevantes del nuevo proceso, como el nombre del proceso, la fecha, el número total de escaños disponibles y la lista de partidos involucrados.



Al pulsar sobre “Seleccionar Partidos Participantes” se abrirá una nueva ventana que permitiá al usuario introducir los partidos participantes.

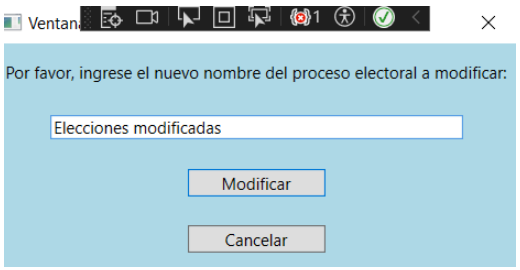
b) ***Eliminar proceso electoral*** : Mediante esta función, el usuario puede eliminar un proceso electoral previamente registrado. Esta opción permite mantener actualizada la base de datos, eliminando procesos obsoletos o incorrectamente ingresados

c) ***Modificar proceso electoral:*** Esta alternativa proporciona la posibilidad de realizar ajustes o actualizaciones a los procesos electorales ya existentes. Dentro de esta opción, los usuarios encontrarán varias subopciones:

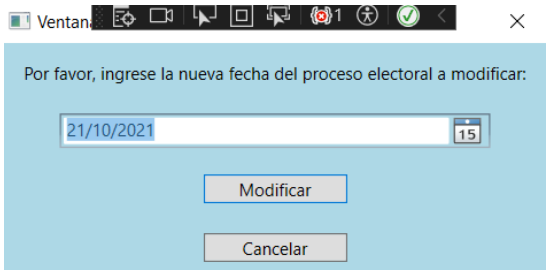


A su vez cada una de ellas creará una interfaz diferente:

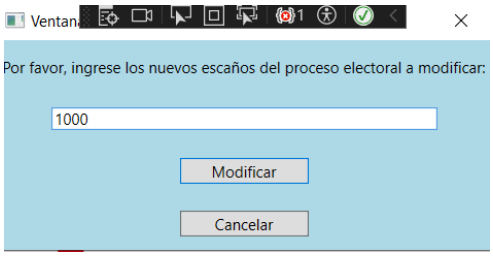
-Modificar nombre: Permite cambiar el nombre del proceso electoral seleccionado.



-Modificar fecha: Permite cambiar la fecha del proceso electoral seleccionado.



-Modificar escaños: Permite cambiar los escaños del proceso electoral seleccionado.

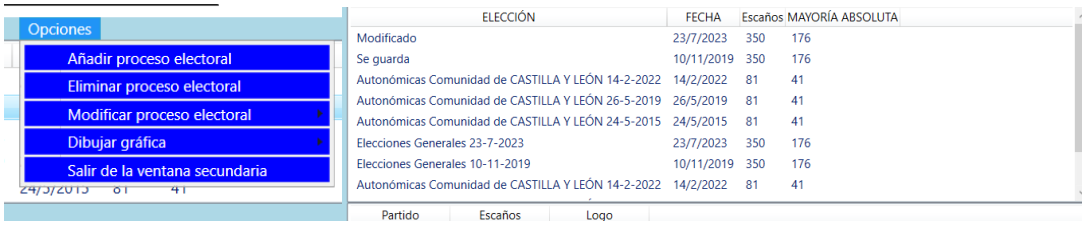


-Modificar lista de partidos: Permite cambiar los partidos involucrados en el proceso seleccionado.

Obviamente, todas las modificaciones realizadas se verán reflejadas en la tabla de registros.

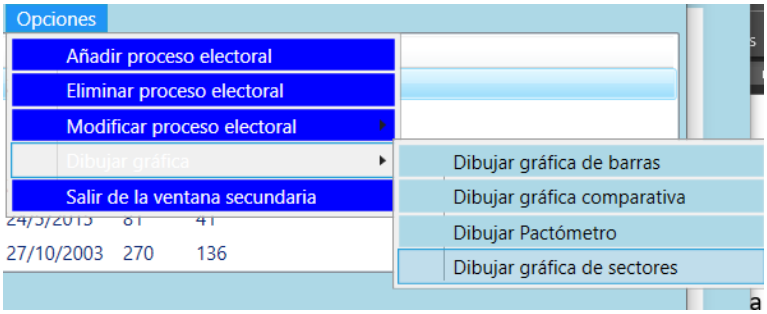
ELECCIÓN	FECHA	Escaños	MAYORÍA ABSOLUTA
Elecciones modificadas	21/10/2021	1000	176
Elecciones Generales 10-11-2019	10/11/2019	350	176
Autonómicas Comunidad de CASTILLA Y LEÓN 14-2-2022	14/2/2022	81	41

Además, si se desea guardar las modificaciones realizadas el usuario deberá hacer click sobre "Salir de la ventana secundaria". De esta manera los cambios quedarán registrados, a pesar de que la ventana secundaria esté cerrada.



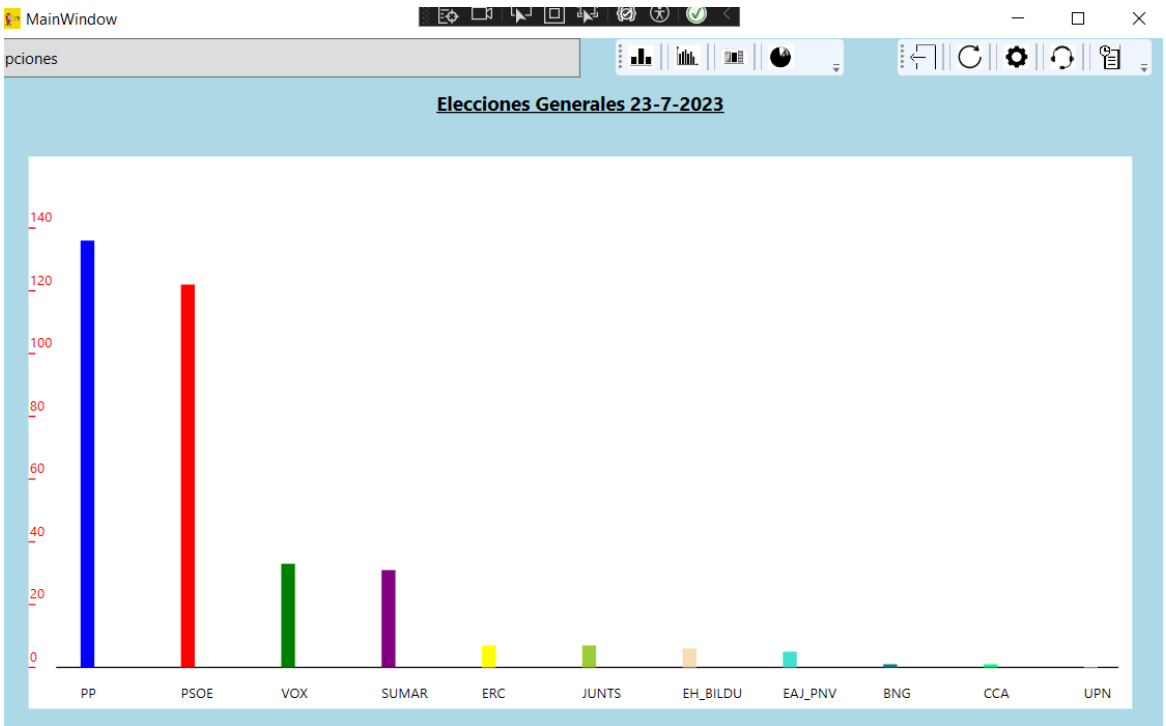
6. Visualización de resultados

La ventana principal desempeña un papel crucial como el centro de visualización para los resultados y las gráficas generadas a partir de los registros de la ventana secundaria "CDTabla". Esta disposición asegura que el usuario tenga un acceso directo y centralizado a las representaciones gráficas de los datos electorales.

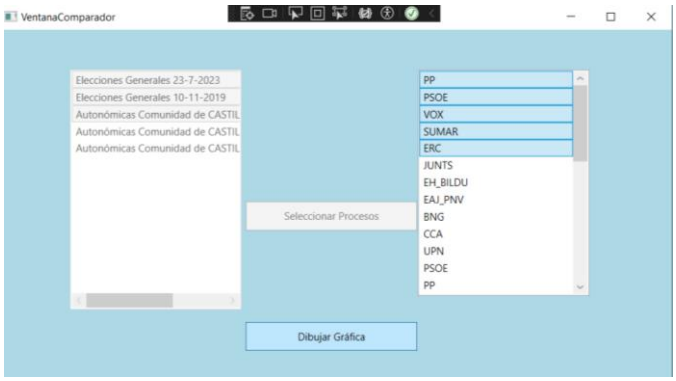


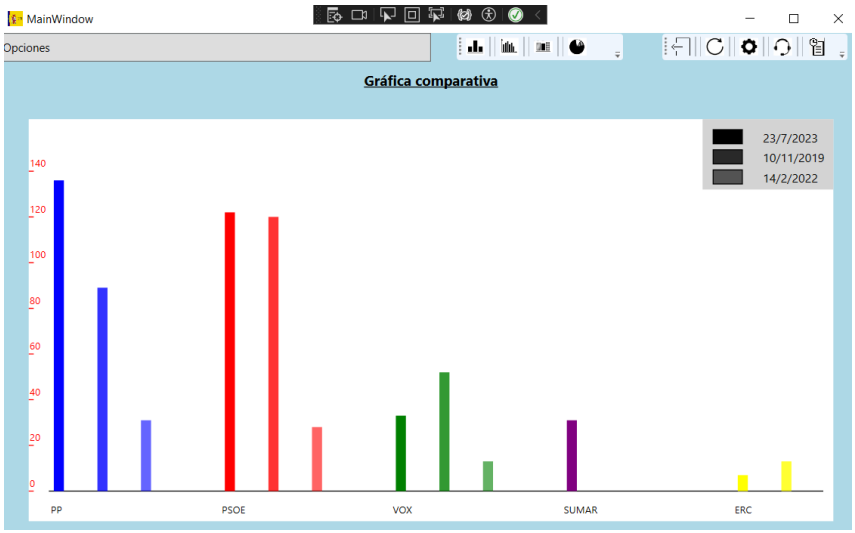
Tipos de Gráficas Disponibles:

a) ***Gráfica Normal de Barras***: Esta gráfica ofrece una representación clásica de los resultados electorales, mostrando el número de escaños por cada partido en forma de barras. Es ideal para una visualización rápida y clara de los resultados.



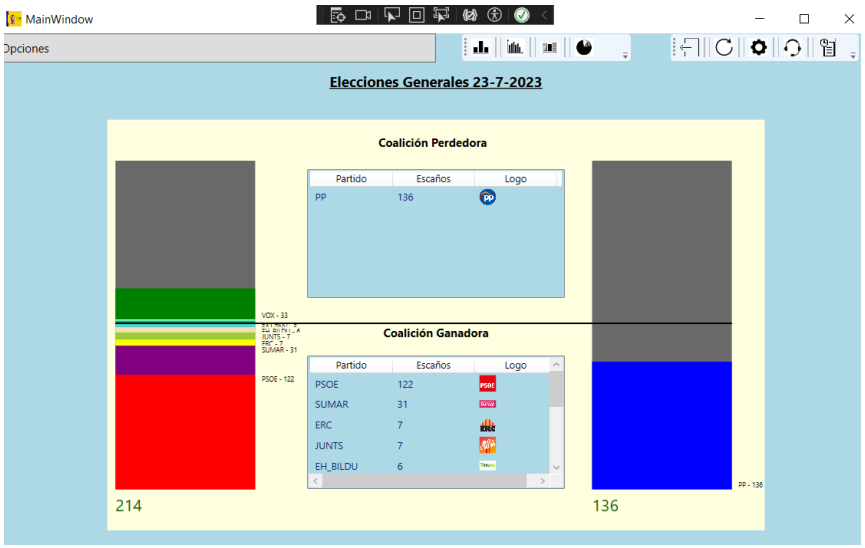
b) ***Gráfica Comparativa***: Esta opción permite al usuario comparar los resultados de diferentes procesos electorales. Es especialmente útil para analizar tendencias, ganancias o pérdidas de escaños entre distintas elecciones. Al seleccionar esta alternativa se abrirá una nueva interfaz que permitirá al usuario seleccionar los procesos y partidos de los que tiene interés y desea comparar.



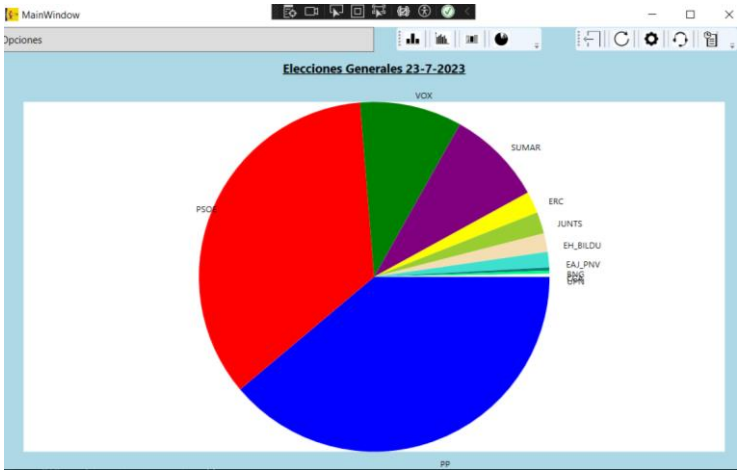


c) ***Gráfica Pactómetro***: La gráfica del Pactómetro es una herramienta que permite visualizar posibles coaliciones o mayorías gubernamentales. Muestra de manera gráfica qué partidos podrían formar gobierno juntos, basándose en la suma de sus escaños e indica cuál de ellas ganaría.

Para ello, al iniciar la gráfica del Pactómetro, todos los partidos políticos aparecen listados en la columna izquierda de la interfaz. Esta disposición inicial representa la situación antes de formar cualquier coalición. Para formar una coalición, el usuario debe hacer click sobre cada uno de los partidos que desea incluir. Al hacerlo, los partidos seleccionados cambiarán de lado en la interfaz, indicando que ahora forman parte de una nueva coalición. Una vez formadas las coaliciones, la aplicación compara automáticamente su representación. La coalición con mayor representación se clasifica como "Coalición Ganadora" y se muestra en una tabla. La otra coalición se clasifica como "Coalición Perdedora" y se visualiza en una tabla diferente. Esta herramienta permite a los usuarios experimentar con diferentes combinaciones de partidos para ver cómo estas uniones podrían afectar la formación de un gobierno o la mayoría parlamentaria.



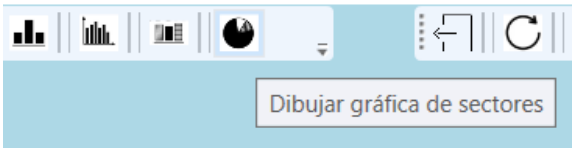
d) ***Gráfica de Sectores***: Esta gráfica presenta los resultados en forma de sectores circulares, proporcionando una perspectiva visual intuitiva de la distribución de escaños entre los distintos partidos. Es útil para obtener una rápida comprensión de la proporción de representación de cada partido.



7.Funciones adicionales

Se ha dotado a "El Pactémtro" con una serie de funciones nuevas que buscan optimizar la experiencia del usuario y facilitar el manejo del programa. Estas funciones están accesibles a través de una barra de operaciones, estratégicamente diseñada para mejorar la interacción con el programa. La barra de operaciones se divide en dos secciones principales:

1. ***Primera Barra - Dibujo de Gráficas***: En esta sección, el usuario tiene la posibilidad de dibujar distintos tipos de gráficas relacionadas con los datos electorales. Las opciones disponibles incluyen: Gráfica Normal, gráfica Comparativa, gráfica del pactómetro y gráfica de sectores.

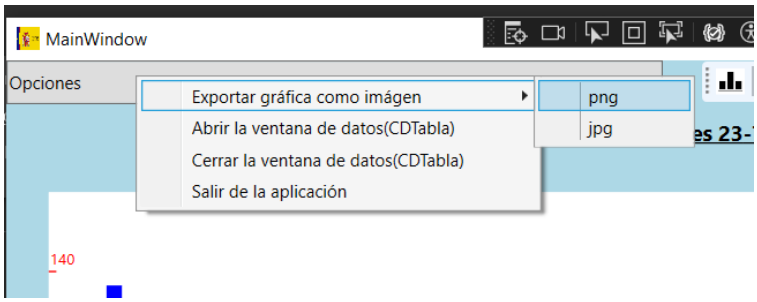


2. ***Segunda Barra - Gestión del Programa***: Esta barra proporciona herramientas útiles para la gestión general del programa, incluyendo:

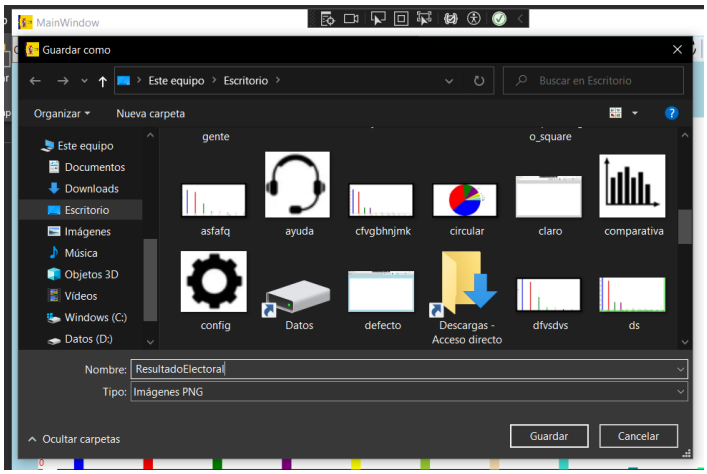
    1. ***Salir de la Aplicación***: Permite cerrar el programa de manera segura.
    2. ***Limpiar Canvas***: Esta función borra cualquier dato o gráfica presente en la pantalla, ofreciendo un inicio limpio para nuevos análisis.
    3. ***Abrir Ventana de Configuración***: Da acceso a las opciones de personalización de la ventana principal.
    4. ***Descargar Manual***: Permite al usuario descargar el manual de usuario para una consulta rápida y detallada.
    5. ***Abrir Historial de Procesos***: Facilita el acceso al registro de procesos electorales anteriores, permitiendo una revisión y análisis de datos históricos (sin tener que acceder a la ventana secundaria).

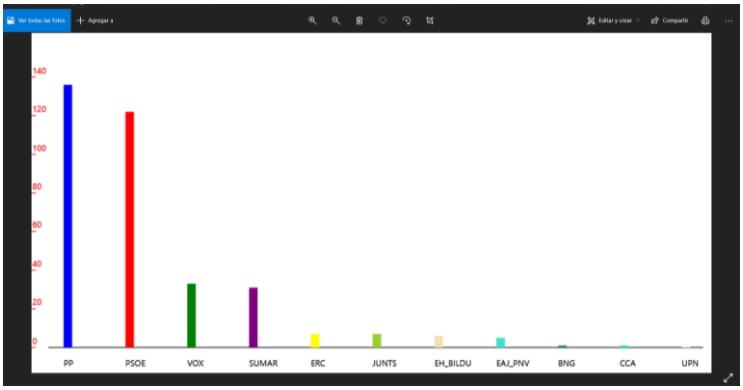
1 2 3 4 5

Además, se permite exportar el resultado permitiendo a los usuarios guardar y compartir las gráficas y resultados del análisis electoral en formatos de imagen, PNG y JPG. Esta característica es útil para incluir los resultados en informes y para compartirlos fácilmente con otras personas.



Se le pedirá el nombre de la imagen a exportar.





8.Información de contacto y soporte

Para asistencia y soporte relacionados con "El Pactómetro", los usuarios pueden contactar directamente al responsable del proyecto.

Contacto Principal

- Nombre: Marcos Rivas Kyoguro
- Correo Electrónico: marcos.rivkyo@usal.es

Para consultas, soporte técnico o cualquier otra pregunta relacionada con "El Pactómetro", por favor envíe un correo electrónico a la dirección proporcionada. Es recomendable incluir una descripción detallada de la consulta o del problema a resolver para facilitar una asistencia más eficaz y rápida.

Manual del Programador

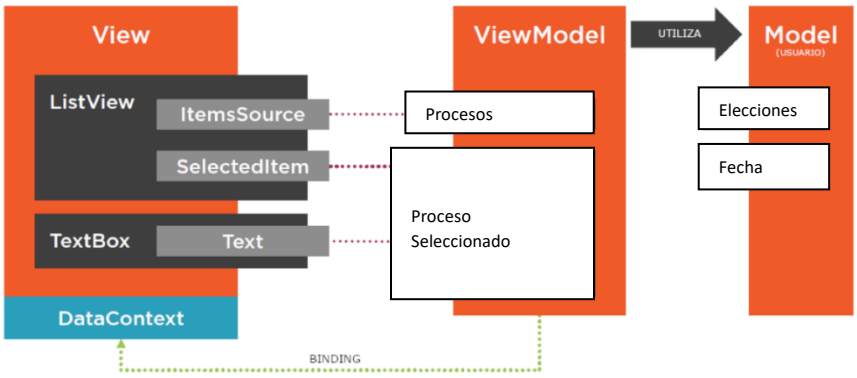
En el desarrollo de este proyecto, se ha adoptado el patrón de diseño Modelo-Vista-ViewModel (MVVM) debido a las múltiples ventajas que ofrece. Esta elección se fundamenta en la capacidad del patrón MVVM para mejorar la organización del código, facilitar el mantenimiento y la prueba de la aplicación, y proporcionar una separación clara de responsabilidades entre la interfaz de usuario y la lógica de negocio.

El enlace de datos usa un un método para que las ventanas presenten e interactúen con datos . Las propiedades de los elementos de interfaz se pueden enlazar a datos de diversos orígenes. Esto presenta varias ventajas, entre ellas:

- *Representación flexible de los datos en la interfaz de usuario*
- *Separación bien definida entre la lógica de negocio y la interfaz de usuario*

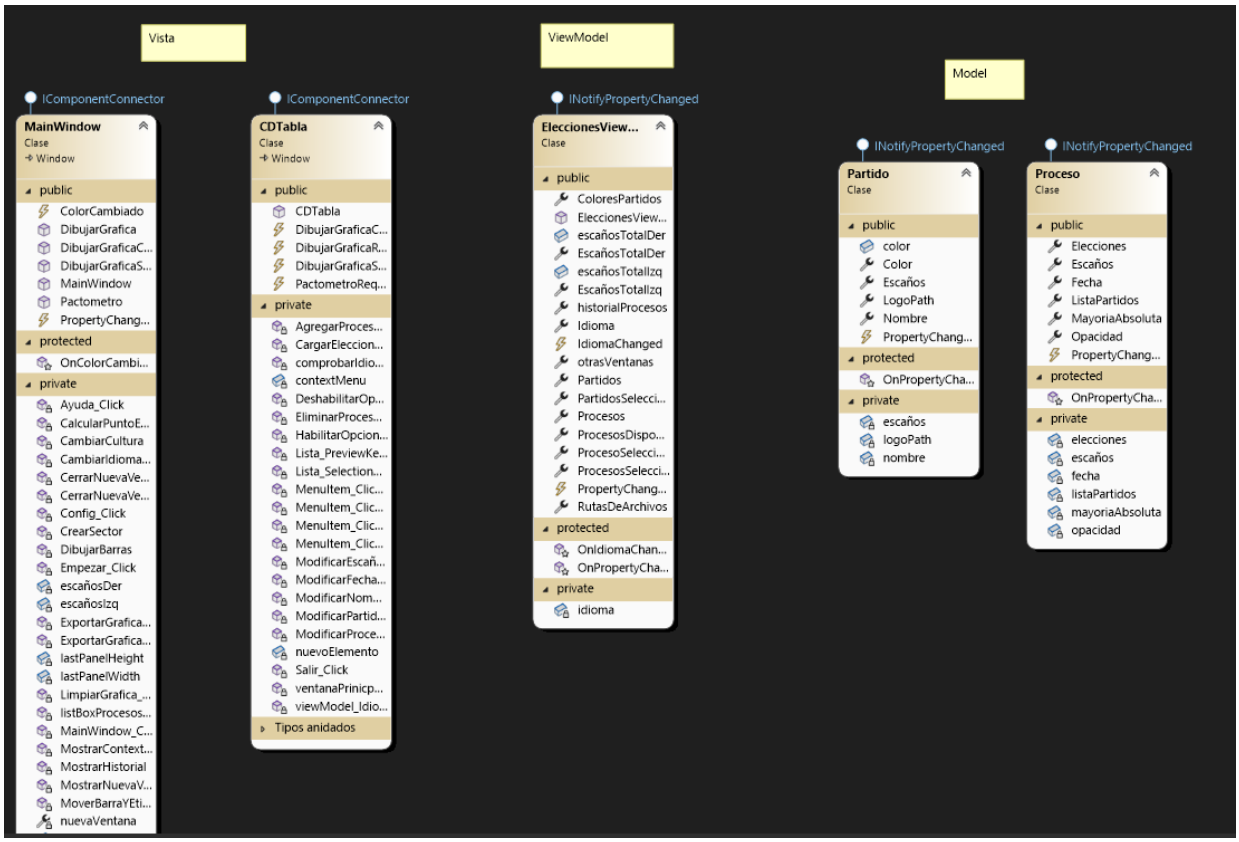
Al emplear el patrón MVVM, el enlace de datos implica la actualización automática de los datos que subyacen a una representación externa de los datos de un elemento, cuando esta representación cambia. Es decir, al modificar un dato del "Model", este notifica a la "View", para que realice los cambios correspondientes en la representación de la información.

En la siguiente imagen se muestra el esquema general de la arquitectura MVVM. La aplicación representada en el esquema muestra un listado de elecciones en la cual podemos seleccionar uno de ellos. Del proceso electoral seleccionado se mostraría su nombre y fecha en un campo de texto.

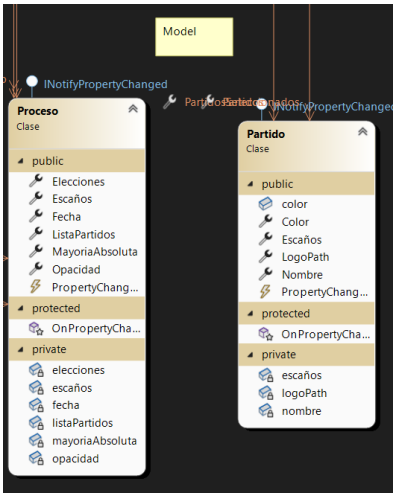


Para detectar los cambios en el "Model", y el "ViewModel", el origen implementa un mecanismo apropiado de notificación de cambios de propiedades, como "INotifyPropertyChanged".

A continuación, se verá en el diagrama de objetos las principales clases creadas, así como su explicación.



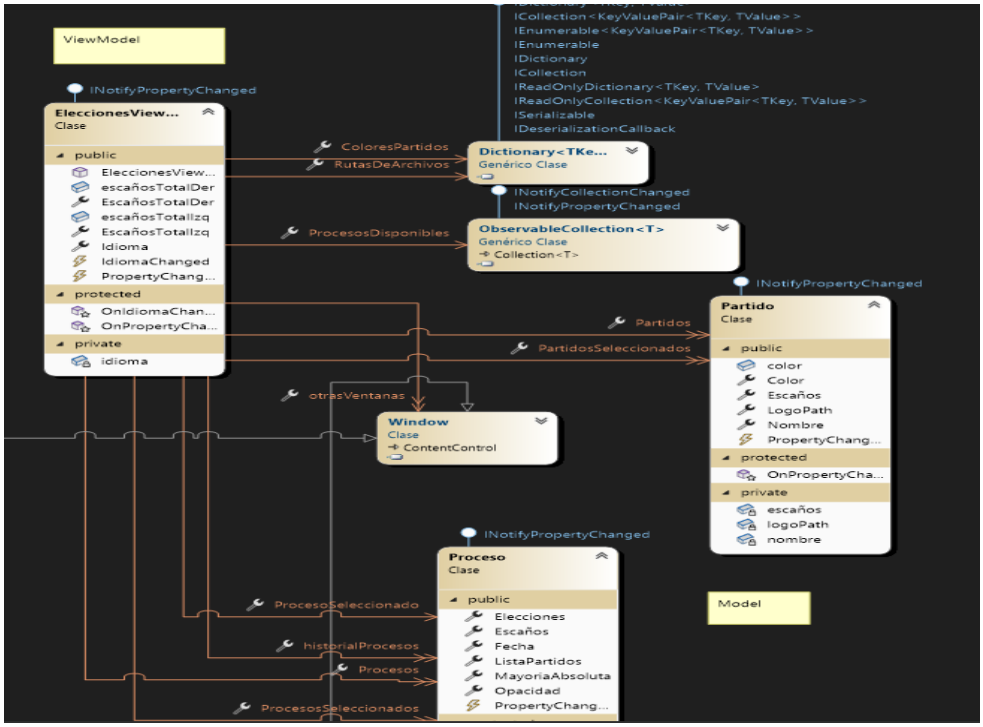
1. **<u>En la capa del Model</u>**, se han diseñado dos clases fundamentales, que son esenciales para representar los datos y la información manejados por la aplicación. Estas clases son Partido y Proceso, y cada una de ellas encapsula elementos clave de la estructura de datos.



1. **Clase Partido**: Representa entidades políticas con atributos específicos que son cruciales para la operación y visualización en la aplicación: Color: Escaños: Nombre: Logo.
2. **Clase Proceso:** Por otro lado, la clase Proceso se encarga de encapsular todos los detalles relevantes de un proceso electoral: Elecciones: Escaños: Fecha: ListaPartidos: Mayoria Absoluta: Opacidad.

La parte más significativa de este sector, es que ambas clases, Partido y Proceso, implementan la interfaz “INotifyPropertyChanged”. Esta permite la comunicación eficaz de cambios en las propiedades de los modelos a las “Views” correspondientes. Al modificar cualquier propiedad en estas clases, se genera un evento de notificación que asegura que la “View” se actualice automáticamente para reflejar estos cambios. Esta funcionalidad es clave para mantener la sincronización entre los datos y su representación visual, asegurando que la interfaz de usuario sea siempre un reflejo preciso y actualizado del estado de los datos de la aplicación.

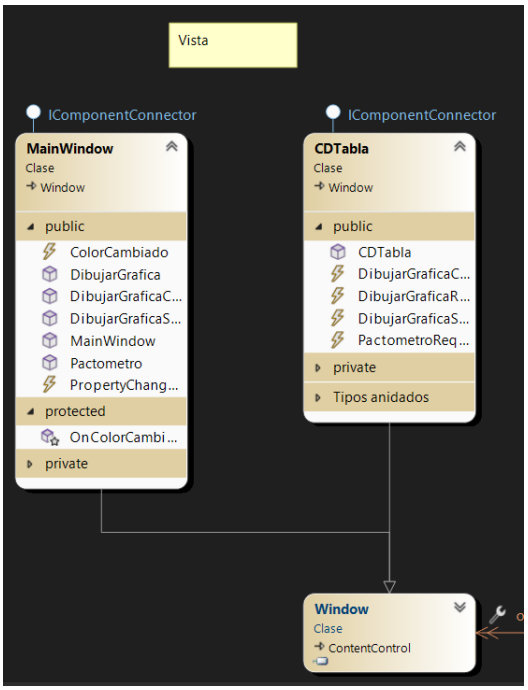
2. ***La capa ViewModel***, desempeña un papel fundamental en la mediación entre el "Model" y las "Views". Esta capa crea y mantiene colecciones y representaciones del modelo, que son esenciales para la interacción del usuario con la aplicación. A continuación, se detalla el funcionamiento y la importancia de la capa ViewModel:



- **Colecciones Representativas**: El "ViewModel" crea colecciones que representan y encapsulan los datos del "Model", como las entidades Partido y Proceso. Estas colecciones son utilizadas por todas las ventanas de la aplicación.
- **Acceso Compartido**: Dado que estas colecciones son compartidas entre diferentes partes de la interfaz de usuario, se garantiza que todas las ventanas obtengan una visión coherente y actualizada de la información. Para ello es creada en la ventana principal y se le pasa como argumento a las demás, que las establecen como su DataContext.
- **Sincronización de Datos**: Las colecciones en el "ViewModel" aseguran que cualquier cambio en los datos se refleje de manera uniforme en toda la aplicación, manteniendo la coherencia en todas las interfaces.

Al igual que en las clases del "Model", el "ViewModel" implementa la interfaz "INotifyPropertyChanged". Esta implementación es crucial para la propagación de cambios, pues cuando se realiza un cambio en una propiedad dentro del "ViewModel", este cambio se notifica automáticamente a las "Views" asociadas.

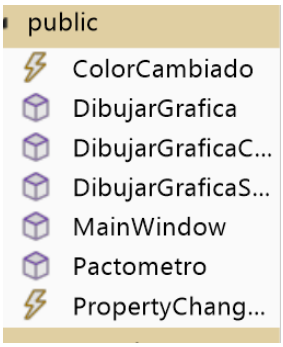
3. ***La capa Vista***, tiene un enfoque centrado en la presentación y la interacción del usuario con la interfaz gráfica. La "View" se beneficia de una interacción simplificada con el "Model" y el "ViewModel", lo que resulta en una menor necesidad de declarar variables de manera explícita en la "View". A continuación, se destacarán los métodos principales utilizados en la "View"para la generación de gráficas (Canvas) y la representación de datos en tablas (Listviews).



Métodos Clave en la Vista Generación de Gráficas (MainWindow)

Estos método son responsables de dibujar gráficas en el elemento Canvas. Utiliza datos vinculados (binding) del "ViewModel" para representar visualmente la información, como los resultados o procesos electorales.

- **DibujarGráfica():** Este método se encarga de generar gráficas normales, utilizando las medidas proporcionadas por el "ViewModel" y aplicando fórmulas matemáticas para determinar dimensiones como la altura y anchura de las barras.
- **DibujarGráficaComparativa()**: Encargado de la creación de gráficas comparativas, este método utiliza datos de procesos diferentes para ilustrar diferencias a lo largo del tiempo.
- **DibujarGráficaSectores():** Este método se enfoca en la representación de datos en formato de gráfica de sectores, empleando cálculos para la proporción y distribución de los sectores.
- **Pactómetro():** Un método utilizado para la visualización de posibles coaliciones políticas. Además, incorpora una funcionalidad interactiva que registra un evento al hacer click en un rectángulo, permitiendo el movimiento de barras de una columna a otra, facilitando así la formación y visualización de coaliciones.



Métodos Clave en la Vista Actualización de Tablas(CDTabla)

Métodos que responden a los cambios en los datos y actualizan las gráficas en consecuencia, asegurando que la representación visual siempre esté sincronizada con los datos más recientes. Encargado de llenar los elementos "ListView" con datos provenientes del "ViewModel".

La ListView desempeña un papel crucial en la visualización de los datos del modelo. Utiliza técnicas de Data Binding para asegurar que los datos mostrados estén siempre actualizados y sincronizados con el modelo subyacente. Para ello hace uso de “DisplayMemberBinding” para conectar con el “Model”. Este enlace asegura que cualquier dato representado en la ListView refleje directamente la información contenida en el “Model”. Esto significa que la ListView se actualiza en tiempo real para reflejar cualquier modificación, adición o eliminación de datos en el modelo sin necesidad de actualizarlo manualmente. (Evita el uso de List.Refresh())

EVENTOS PRINCIPALES

En el desarrollo del proyecto, se ha usado varios eventos para facilitar la comunicación y la delegación de operaciones entre diferentes componentes de la interfaz de usuario, especialmente entre ventanas secundarias y la ventana principal.

La implementación de estos eventos juega un papel crucial en la interacción del usuario y la lógica de la interfaz de usuario.

- **Se han definido eventos como DibujarGraficaEventHandler en las ventanas secundarias.**

Estos eventos están diseñados para delegar operaciones específicas, como la generación de gráficas, a la ventana principal.

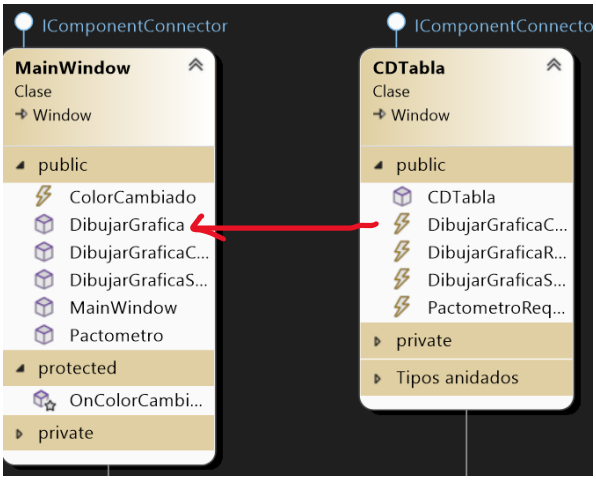
Creación del evento en la ventana secundaria:

```
public delegate void DibujarGraficaEventHandler(object sender, RoutedEventArgs e);
public event DibujarGraficaEventHandler DibujarGraficaRequested;
```

Registro de eventos en la ventana principal:

```
nuevaVentana.DibujarGraficaRequested += DibujarGrafica;
```

Este registro asegura que cuando el evento se dispare en la ventana secundaria, el método DibujarGráfica de la ventana principal se ejecute.



- **Eventos para Cambio de Idioma y Color**: Además de los eventos de delegación de operaciones, se han implementado eventos que permiten a los usuarios cambiar aspectos de la configuración de la aplicación, como el idioma y el color de la ventana principal.

1. **Evento de cambio de idioma**: En la ventana principal se registra =>

this.viewModel.IdiomaChanged += viewModel_IdiomaChanged;

De manera que cuando se modifique la propiedad Idioma del ViewModel, este ejecuta el métodoviewModel_IdiomaChanged() y cambie su idioma.

La gestión del idioma se ha realizado mediante el uso de archivos de recursos, “Resource_en.resx” y “Resource_en.resx”. De esta manera se puede modificar el idioma de los principales componentes de la ventana principal, como los botones, menucontext …

Inicialmente(En el Código XAML):

```
<Button x:Name="BotonInicio" Content="{x:Static local:Resources_es.InitiateText}" …/>
```

Al activarse el evento “IdiomaChanged” (En el código C#):

```
string nombreArchivoRecursos = "Pactometro.Properties.Resources_en";

ResourceManager resourceManager = new ResourceManager(nombreArchivoRecursos,
typeof(MainWindow).Assembly);

BotonInicio.Content = resourceManager.GetString("InitiateText");
```

2. **Evento de cambio de color**:

Este evento cambia el color de la ventana principal desde la ventana de configuración

En la clase VentanaConfig, se define un evento público llamado ColorSeleccionado.

```
public event EventHandler<ColorSeleccionadoEventArgs> ColorSeleccionado;
```

Este evento se dispara cuando el usuario selecciona un nuevo color en la ventana de configuración. El evento está asociado con un argumento de evento personalizado, ColorSeleccionadoEventArgs, que contiene la propiedad NuevoColor para almacenar el color seleccionado.

```
public class ColorSeleccionadoEventArgs : EventArgs
  {
     public Color NuevoColor { get; set; }
  }
```

Cuando el usuario ajusta los sliders de color (rojo, verde, azul), el método Slider_ValueChanged se activa. Dentro de este método, se crea un nuevo color basado en los valores de los sliders y se actualiza la interfaz de la ventana de configuración con el nuevo color, consecuentemente, se invoca el evento ColorSeleccionado, pasando el nuevo color como parte de ColorSeleccionadoEventArgs.

MainWindow se suscribe al evento ColorSeleccionado de VentanaConfig.

```
ventanaConfig.ColorSeleccionado += (s, args) =>
{
   OnColorCambiado(new ColorChangedEventArgs { NuevoColor = args.NuevoColor });
};
```

MainWindow define un método, OnColorCambiado, que es el manejador para el evento ColorSeleccionado. Cuando VentanaConfig dispara el evento ColorSeleccionado, se ejecuta OnColorCambiado en MainWindow. OnColorCambiado recibe los argumentos del evento, incluido el NuevoColor, y actualiza el color de fondo de la MainWindow con este nuevo color.

```
protected virtual void OnColorCambiado(object sender, ColorSeleccionadoEventArgs e)
{
        this.Background = new SolidColorBrush(e.NuevoColor);
}
```

FUENTES y REFERENCIAS A DOCUMENTOS y PÁGINAS WEB USADAS

**Junto a la referencia se pone para que sirvió la consulta a esa página

Apuntes teóricos de Interfaces Gráficas de Usuario

https://blog.clicko.es/patron-diseno-mvvm-usando-wpf-parte-1/ Fundamentos del patrón MVVM

https://learn.microsoft.com/es-es/dotnet/desktop/wpf/controls/how-to-use-the-image-element?view=netframeworkdesktop-4.8 Base para manejar imágenes

https://learn.microsoft.com/es-es/visualstudio/ide/class-designer/designing-and-viewing-classes-and-types?view=vs-2022 Guía para usar el class designer usado en el manual del programador

https://programmerclick.com/article/61941000992/ //Base para saber cómo funciona el reloj

https://learn.microsoft.com/en-us/dotnet/desktop/wpf/graphics-multimedia/geometry-overview?view=netframeworkdesktop-4.8 //Cómo dibujar sectores, pathGeometry, pathFigure

https://learn.microsoft.com/es-es/dotnet/desktop/wpf/app-development/wpf-application-resource-content-and-data-files?view=netframeworkdesktop-4.8 //Cómo dibujar sectores, pathGeometry, pathFigure

Trabajar con archivos .resx mediante programación - .NET | Microsoft Learn //Cómo usar los archivos de recursos .resx

https://learn.microsoft.com/es-es/visualstudio/debugger/how-to-use-the-wpf-tree-visualizer?view=vs-2022 //Base para saber cómo identificar objetos en el árbol visual dinámico

https://www.youtube.com/watch?v=6hIvPCkwMds //Fundamentos del patrón MVVM

https://www.youtube.com/watch?v=-xTqfilaYow //Fundamentos del patrón MVVM

https://youtu.be/77eHwdY45jY //Solución a bug que me impedía ver los errores en VSCode

https://learn.microsoft.com/es-es/dotnet/api/system.resources.resourcemanager?view=net-7.0 //Cómo usar resourceManager

https://learn.microsoft.com/es-es/dotnet/api/system.drawing.color.fromargb?view=net-7.0 //Como cambiar la opacidad de un color

https://stackoverflow.com/questions/40323022/how-can-i-capture-an-entire-image-from-a-canvas //Como capturar imagen del canvas

Código Fuente creado en prácticas de Interfaces Gráficas de Usuario en todas las sesiones.

Especialmente el creado para el enlace de datos, he partido de lo aprendido en la última sesión de prácticas, con INotifyPropertyChanged y avanzado con el patrón MVVM.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;

namespace EnlaceTabla
{
    public class Amigo : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        // Atributos
        string nombre;
        string apellido;
        int edad;

        // Propiedades
        public string Nombre
        {
            get { return nombre; }
            set { nombre = value; OnPropertyChanged("Nombre"); }
        }

        public string Apellido
        {
            get { return apellido; }
            set { apellido = value; OnPropertyChanged("Apellido"); }
        }

        public int Edad
        {
            get { return edad; }
            set { edad = value; OnPropertyChanged("Edad"); }
        }

        // Constructor
        public Amigo(string n, string a, int e)
        {
            Nombre = n;
            Apellido = a;
            Edad = e;
        }

        // Método que lanza el evento PropertyChanged cuando se cambia el valor de cualquier propiedad
        void OnPropertyChanged(String propertyname)
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(propertyname));
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Collections.ObjectModel;

namespace EnlaceTabla
{
    /// <summary>
    /// Lógica de interacción para MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        ObservableCollection<Amigo> listaamigos;
        public MainWindow()
        {
            InitializeComponent();
            listaamigos = new ObservableCollection<Amigo>();
            lista.ItemsSource = listaamigos;
        }

        private void Button_Click_1(object sender, RoutedEventArgs e)
        {
            Amigo amigo = new Amigo(Caja1.Text, Caja2.Text, 25);
            listaamigos.Add(amigo);

        }

        private void lista_SelectionChanged_1(object sender, SelectionChangedEventArgs e)
        {
            /* Cada vez que se selecciona un elemento, se muestra en la etiqueta de abajo su edad */
            Amigo amigo = (Amigo)lista.SelectedItem;
            if (amigo != null)
            {
                /* Al seleccionar un elemento de la tabla se muestra en la etiqueta de abajo la edad
                 * correspondiente al elemento.
                 * Además, para probar cómo funciona la modificación de las propiedades, hacemos que
                 * cuando se seleccione un elemento se incremente su edad */
                amigo.Edad += 10;     // Modificamos cualquier propiedad del elemento seleccionado
                CajaSalida.Content = amigo.Edad;
                /* En este caso,  como Amigo implementa la interfaz INotifyPropertyChanged, el cambio
                 * del valor de Edad se verá reflejado en la tabla de modo inmediato
                 * Para probarlo, añadir un par de elementos a la tabla y seleccionarlos
                 * de modo alternativo
                 */

            }

        }
    }
}
```