

EL PACTÓMETRO

Interfaces Gráficas de Usuario

Grado Ingeniería Informática USAL

Óscar Khergo

Contenido

1. Introducción	3
2. Manual de usuario.....	4
3. Manual del programador	7
3.1. Diagrama de clases	7
3.2. Glosario de clases	8
3.3. Relaciones y mecanismos de comunicación	13
4. Referencias bibliográficas.....	15

1. Introducción

Esta documentación recoge toda la información relevante sobre el proyecto final de la asignatura Interfaces Gráficas de Usuario del Grado en Ingeniería Informática de la Universidad de Salamanca.


Se trata de una aplicación orientada a almacenar y representar gráficamente los resultados electorales en elecciones nacionales o autonómicas de España. Para ello, cuenta con dos ventanas, una que muestra listados con toda la información, y otra para su representación. Además, cuenta con otras funcionalidades como permitir que se añadan nuevas elecciones de forma manual o mediante la importación de un fichero de texto, como el que se incluye en el proyecto y genera unos datos iniciales.

Aparte de todos estos requisitos básicos, se han añadido otras funcionalidades extras para dotar de más posibilidades a la aplicación. En primer lugar, al iniciarse, aparece una animación del logo realizada a partir del tema adicional de la asignatura *5.1. Animaciones*. También se presenta la posibilidad de cambiar entre dos temas de colores o modos, claro y oscuro, el configurar los textos de los menús en dos idiomas, español e inglés, o mostrar un icono en la esquina superior izquierda de la ventana. Estas funcionalidades se han añadido tras un proceso de investigación propia con el uso de *ResourceDictionary*. Por último, se han perfeccionado las interfaces y funcionalidades básicas, introduciendo mensajes de error, menús de ayuda, o la columna de abstenciones y la línea de mayoría simple en el gráfico de Pactómetro, otra posibilidad clave y no planteada originalmente.

2. Manual de usuario

Ventana Listados

Se muestra un listado con las elecciones introducidas en el sistema y un segundo con los resultados de aquella que este seleccionada en el primero.

 Listados







—
□
×

Archivo Ver Ayuda

LISTADO DE ELECCIONES

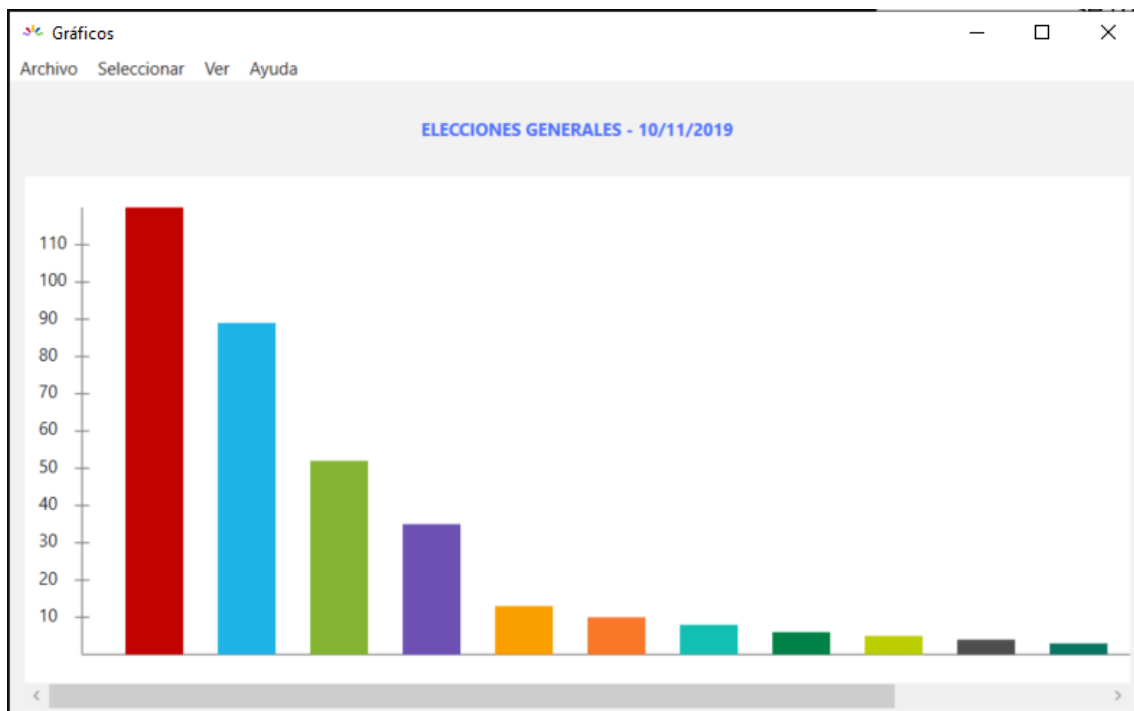
ELECCIÓN	FECHA	ESCAÑOS	MAYORÍA ABSOLUTA
ELECCIONES GENERALES	23/7/2023	350	176
ELECCIONES GENERALES	10/11/2019	350	176
ELECCIONES AUTONÓMICAS CYL	14/2/2022	81	41
ELECCIONES AUTONÓMICAS CYL	26/5/2019	81	41

RESULTADOS

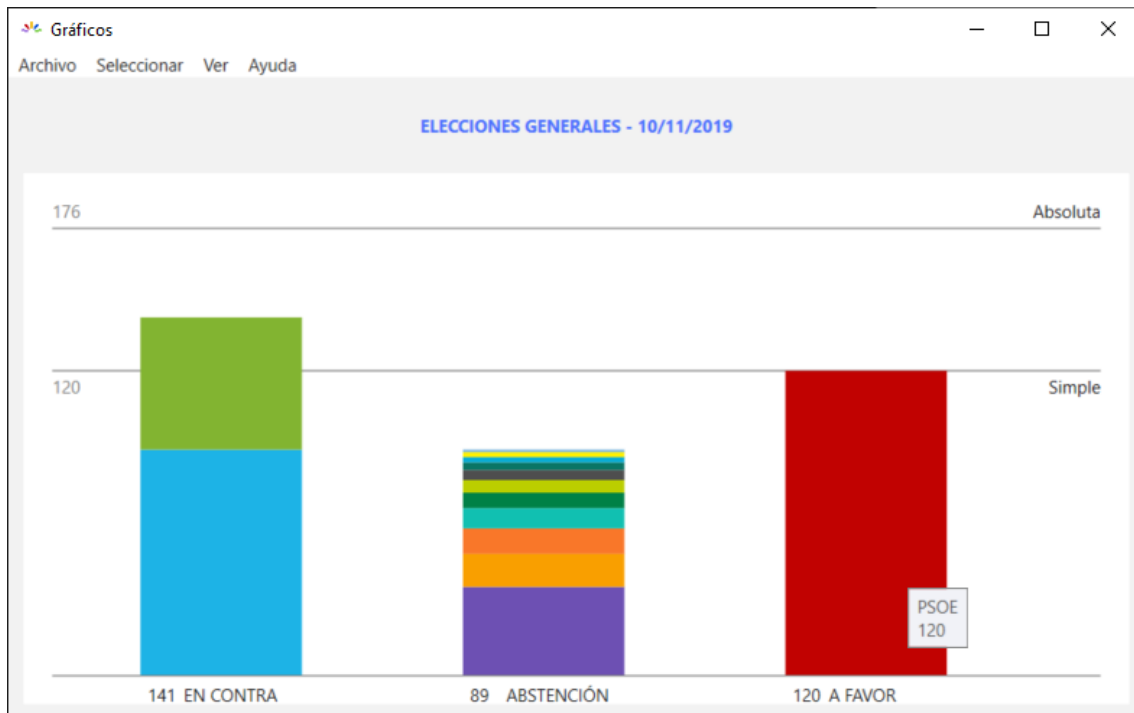
PARTIDO	ESCAÑOS	COLOR
PSOE	120	
PP	89	
VOX	52	
PODEMOS	35	
ERC	13	
CS	10	

Ventana Gráficos

Se muestra por medio de un gráfico de barras el resultado de los partidos de la elección seleccionada en la ventana de Listados.



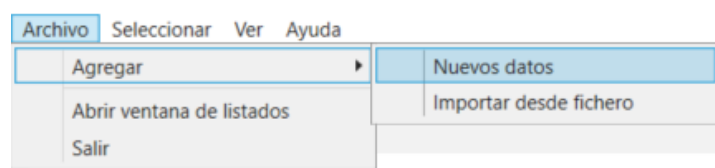
Se puede elegir entre tres tipos de gráficos con la opción *Seleccionar* del menú superior: el gráfico Elección, representa los votos de cada partido; Comparativa, compara los resultados de la elección seleccionada con los del resto de elecciones del mismo tipo; y Pactómetro, sitúa los partidos en tres columnas (a favor, en contra y abstención). Para conmutar entre cada columna tan solo hay que clickar sobre el rectángulo deseado y este se moverá a la siguiente. Además, colocando el cursor encima te indica el nombre y número de escaños.



Menú

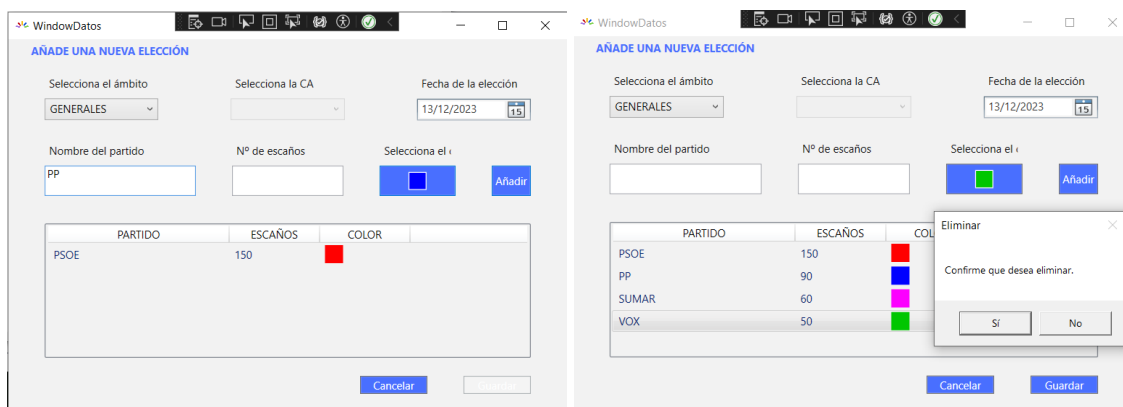
Archivo permite añadir nuevas elecciones al sistema por medio de dos métodos: “Nuevos datos”, que muestra una nueva ventana para añadir datos manualmente, e “Importar desde fichero”, que solicitará un archivo desde el que importar. También permite abrir la otra ventana, si esta se ha cerrado, y salir de la aplicación.

Ver permite cambiar entre el modo oscuro y el modo claro, que modifican la apariencia visual de la aplicación, y entre el idioma inglés o español. Por su parte, *Ayuda* ofrece las opciones “Ver la ayuda”, que muestra información sobre el manejo de la aplicación; y “Sobre el desarrollador”.



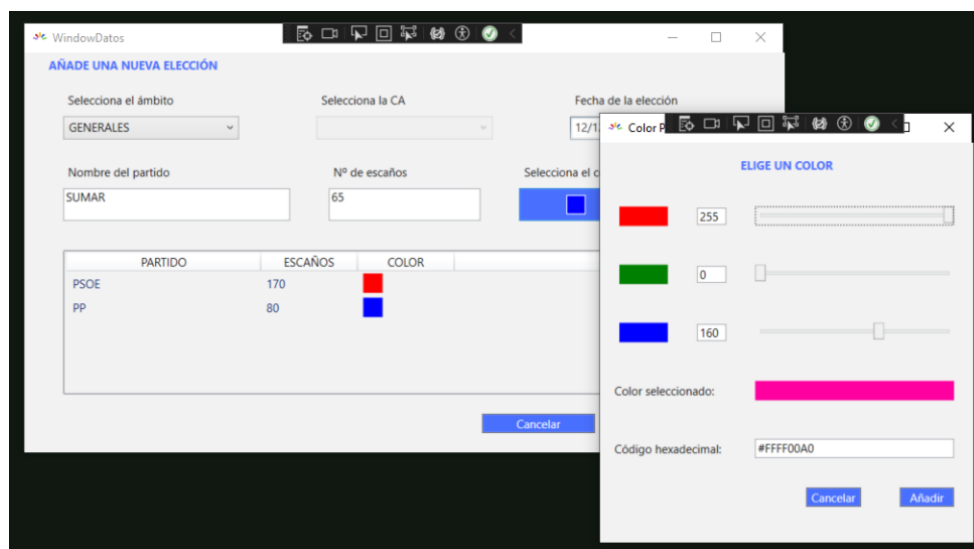
Ventana Nuevos Datos

Esta ventana, mostrada al seleccionar la opción homónima del menú, permite añadir nuevos procesos electorales de forma manual. La interfaz muestra un selector del ámbito de la elección (general o autonómica), un segundo selector que se activa solo en el segundo caso para mostrar las distintas Comunidades Autónomas, un selector de fecha y las cajas de texto para introducir el nombre y número de escaños obtenidos por cada partido. Además, permite seleccionar el color de cada uno con un botón que llama a una nueva ventana *ColorPicker*. Al pulsar sobre “Añadir” se mostrará el partido en listado inferior, siempre que los datos introducidos sean válidos, y en caso de querer eliminarlo, podrá hacerse pulsando sobre él. Una vez la suma de los escaños de todos los partidos concuerda con el asignado al tipo de elección seleccionada, se habilita el botón “Guardar” para almacenarlo en el sistema.



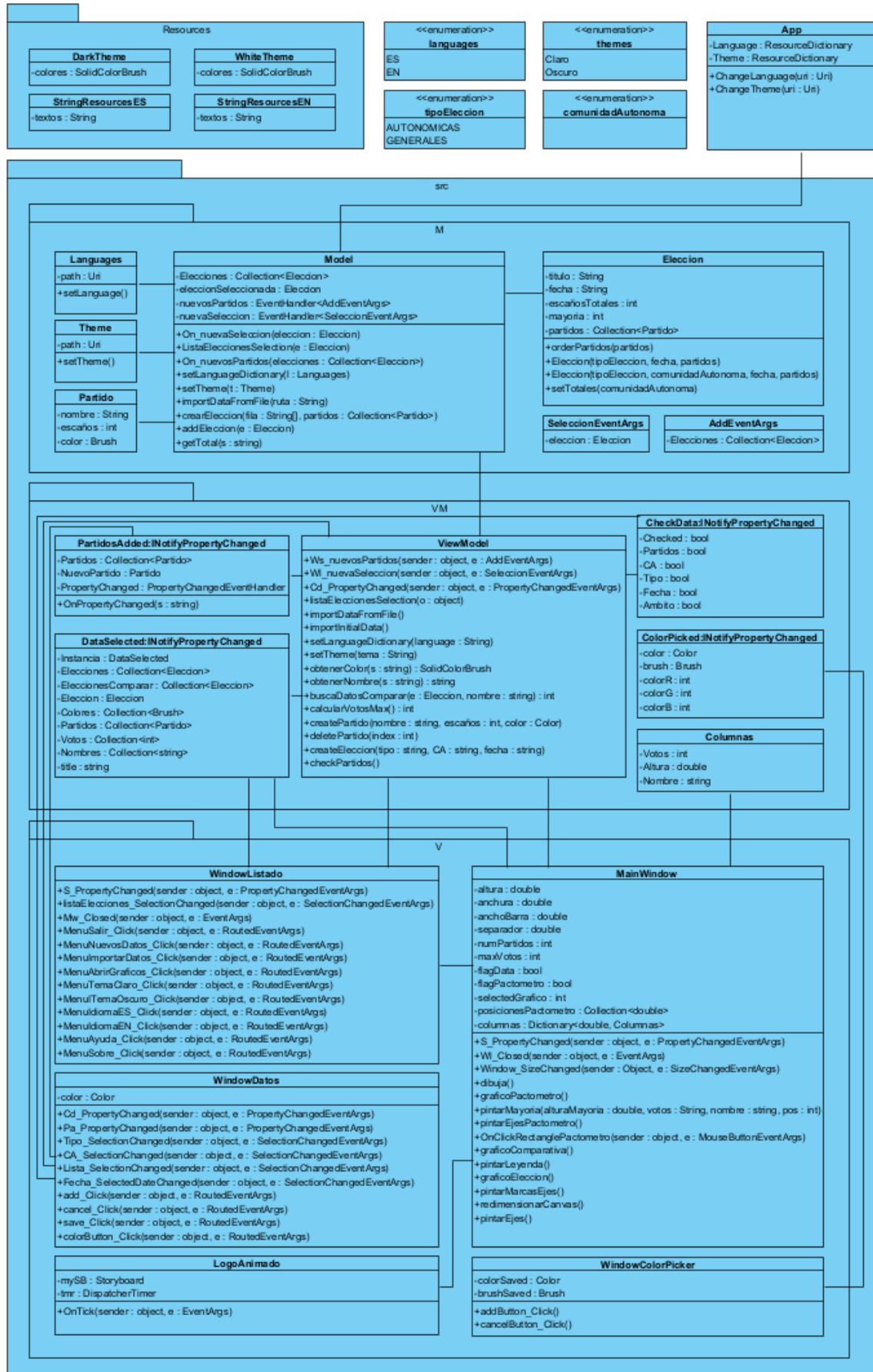
Ventana Color Picker

Esta ventana, mostrada al pulsar sobre el botón de color en el proceso de adición de nuevos partidos, permite seleccionar un color por medio de 3 sliders de las componentes R, G y B, o introduciendo directamente la referencia hexadecimal. Se guardarán los datos al pulsar sobre “Aceptar”.



3. Manual del programador

3.1. Diagrama de clases



3.2. Glosario de clases

PAQUETE MODELO – M

Elección

La clase Elección almacena toda la información necesaria sobre los procesos electorales, siendo esta la fecha, el número de escaños, la mayoría absoluta, el conjunto de partidos y sus resultados, el ámbito (generales o autonómicas) y, en el segundo de estos casos, la Comunidad Autónoma.

Para diferenciar el ámbito se utiliza una enumeración (`tipoEleccion`) y dos constructores que reciben 4 o 3 parámetros, dependiendo de si son o no autonómicas, especificando en este la Comunidad Autónoma concreta. El conjunto de ellas se presenta en una enumeración homónima que, además, se emplea para determinar el número de escaños totales y la mayoría absoluta mediante una sentencia condicional *switch*.

El conjunto de partidos y sus resultados se almacenan, tras haber sido ordenados por número de escaños, en una colección de objetos Partido.

Partido

La clase Partido recoge toda la información relevante a los partidos y sus resultados, como son el nombre, el color que lo identifica y el número de escaños obtenidos en la elección correspondiente.

Para evitar conflictos en la posterior representación gráfica de los resultados, el nombre se almacena convirtiéndolo a mayúsculas mediante *ToUpper()*.

Languages

La clase Languages contiene los idiomas disponibles de la aplicación, que se determinan en una enumeración de igual nombre, junto a la ruta del String Resource al que hacen referencia. En él se almacenan todas las cadenas de texto que se muestran por pantalla acompañadas de un identificador único.

Theme

La clase Theme contiene los modos de color disponibles de la aplicación, que se determinan en una enumeración de igual nombre, junto a la ruta del String Resource al que hacen referencia. En él se almacenan todas las brochas de color que se muestran por pantalla definidas por un identificador único.

Custom Event Args

CustomEventArgs es un paquete que incluye las clases SeleccionEventArgs y AddEventArgs. Ambas se utilizan para comunicar al Modelo con la Vista por medio de eventos. La primera contiene el objeto Eleccion que se envía cuando cambia la elección seleccionada; y la segunda, una colección de objetos Eleccion, que se lanza al producirse cambios en el conjunto de elecciones.

Modelo

La clase Modelo contiene la lógica de negocio de la aplicación y guarda, tanto la Eleccion que se encuentra seleccionada en cada momento, como el conjunto de todas las elecciones almacenadas en una Collection de objetos Eleccion. Para comunicar cambios en las mismas, emplea el uso de eventos con las nuevas clases definidas previamente, sus respectivas manejadoras y los métodos *On_nuevaSeleccion* y *On_nuevosPartidos* que los lanzan.

Estos cambios se producirán por parte del ViewModel a través de los métodos *listaEleccionesSelection*, que recibe la nueva Eleccion seleccionada, y el *addEleccion*, que recibe y añade una Eleccion a la colección, tras haberse comprobado que no existía ya otra de igual tipo y misma fecha. Este último método también es utilizado por *importDataFromFile* y *createEleccion*, que a partir de los datos obtenidos y formateados desde un fichero, crean nuevas elecciones para ser agregadas a la colección de elecciones.

Además de todos ellos, *setLanguageDictionary* y *setTheme*, que reciben un Language y Theme respectivamente, cambian el idioma o modo de color seleccionado para la aplicación. Para ello, se comunican con la clase App, que contiene dos ResourceDictionary de lenguaje y tema intercambiables.

Columnas

Es una clase que define un tipo de dato temporal utilizado para simplificar las colecciones necesarias en la representación gráfica del Pactómetro. Tiene por atributos las propiedades autoimplementadas Votos, Altura y Nombre, que representan por cada una de las tres columnas (nombre: abstención, a favor y en contra), el total de votos apilados, la altura que alcanzan y su nombre.

*Incluida en el paquete VM.

PAQUETE MODELO DE LA VISTA – VM

ViewModel

El ViewModel sirve para conectar la Vista y el Modelo. En él se definen los métodos empleados para importar los datos iniciales, el idioma, el tema y los añadidos desde fichero. Todo lo representado en la Vista se traduce a la lógica almacena en el Modelo, como también en aquellos métodos destinados a crear partidos, elecciones o eliminarlos.

Además, se incluyen diversos métodos de los eventos a los que está suscrito, que se detallarán [más adelante](#), y aquellos que acceden a atributos de las clases siguientes para facilitar cálculos sobre los mismos a las ventanas.

DataSelected

La clase *DataSelected*, que implementa *INotifyPropertyChanged*, contiene la Eleccion seleccionada en cada momento, el conjunto de elecciones de su mismo tipo con las que se puede comparar y la colección de todas las elecciones, que el Modelo notifica cuando ha sido modificada. Además, incluye una serie de colecciones y atributos generados a partir de la elección seleccionada (como Colores, Nombres o Votos, entre otros), que se utilizan para ocultar las clases del Modelo a la Vista con solo tipos de datos estándar.

ColorPicked

La clase *ColorPicked*, que implementa *INotifyPropertyChanged*, contiene el color y su correspondiente brocha, seleccionadas durante el proceso de adición de nuevos partidos. Ambos se crean internamente a partir de los valores enteros asignados a las propiedades autoimplementadas del R, G y B.

PartidosAdded

La clase *PartidosAdded*, que implementa *INotifyPropertyChanged*, contiene una colección de objetos Partido, que se han añadido, de manera interna mediante una propiedad Partido, en el proceso de adición de nuevos partidos.

CheckData

La clase *CheckData*, que implementa *INotifyPropertyChanged*, contiene una serie de banderas o flags booleanas que indican si, en el proceso de adición de nuevas elecciones, se han rellenado los distintos campos correctamente.

PAQUETE VISTA – V

MainWindow

En esta ventana se muestran las representaciones gráficas de los resultados electorales de la Elección seleccionada en la *WindowListados*, a cuyos datos se accede a través del VM *DataSelected*. Está suscrita a los cambios que se producen en dicha clase y pinta en el canvas cada vez que se actualiza la Elección, por medio del método *dibuja* que determina el gráfico elegido mediante la variable *selectedGrafico*. También dibuja cuando sucede el evento de redimensión de la ventana, para lo que actualiza las variables altura y anchura que almacenan las dimensiones efectivas del canvas.

Además, *MainWindow* también está suscrita a los eventos correspondientes al Menú, que pueden modificar el tema, el idioma o el gráfico seleccionado (0: Eleccion, 1: Pactómetro, 2: Comparativa).

Para la representación de gráficos, además de las propias funciones de cada tipo, se definen algunos métodos generales para los gráficos 0 y 2:

- *redimensionarCanvas*, que recalcula el tamaño de las barras en función del tamaño de la ventana, definiendo un tamaño mínimo
- *pintarEjes*, que dibuja los ejes x e y
- *pintarMarcasEjes*, que dibuja las marcas de altura en el eje y
- *pintarLeyenda*, que dibuja la leyenda de colores en el comparativo

Para el gráfico 1, Pactómetro, se necesita almacenar la posición de cada partido en una de las tres columnas (a favor, en contra y abstención). Para ello se crea un Dictionary, que tiene por clave la distancia desde la izquierda en la que se encuentra la columna y por valor un objeto *Columna*, y una colección, *posicionesPactómetro*, que almacena por cada partido (en el índice de su identificador) la columna en la que están situados. También cuenta con los métodos auxiliares *pintarEjesPactómetro*, que sitúa las 3 columnas con su nombre y total de votos acumulados, y *pintarMayoría*, usado para dibujar las líneas de mayoría absoluta y simple, si fuera el caso.

El método *gráficoPactometro* sigue la siguiente estructura: inicializar las colecciones si es la primera vez (*flagPactometro* nulo), limpiar las alturas anteriores, calcular altura de la columna más alta, pintar las líneas de mayoría según dicha altura y, por cada partido, pintar su rectángulo y colocarlo en la columna correspondiente.

En todos los gráficos, las alturas de las barras se asignan en función del tamaño de la más alta.

WindowListados

Es la ventana en la que se muestran, por medio de dos listados, los distintos procesos electorales registrados y, por cada uno, los resultados de cada partido. La colección de elecciones se obtiene a través del VM *DataSelected* cuando se producen cambios desde el modelo. Además, está suscrita al evento de cambio de selección en el primer *ListView* para actualizar el segundo, y a los correspondientes al menú.

WindowDatos

Es la interfaz gráfica para añadir un nuevo proceso electoral a la aplicación. Por medio de dos *ComboBox* permite seleccionar el ámbito y, en caso de que así sea, la Comunidad Autónoma, mientras que presenta un *DatePicker* de los controles de Windows para elegir la fecha. Los partidos se introducen por medio de dos cajas de texto para el nombre y el número de escaños, que solo permiten texto y enteros respectivamente, y un botón para seleccionar el color mediante el *WindowColorPicker* definido a continuación. En caso de que algún dato sea erróneo al intentar añadir el partido, salta un *MessageBox* con un mensaje de error, y de lo contrario, se mostrará en el listado inferior. Además, pueden ser eliminados clickando sobre ellos en dicha lista.

Una vez todos los campos estén completos y los partidos sumen exactamente el total de los escaños de la elección elegida, se activa el botón ‘añadir’. Todas estas comprobaciones se realizan mediante los eventos de cambio de los distintos controles a los que se encuentra suscrito, que modifican las banderas booleanas del VM *CheckData*.

WindowColorPicker

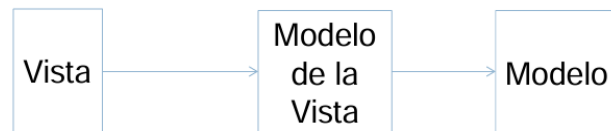
WindowColorPicker es un selector de color personalizado que permite, como su propio nombre indica, seleccionar el color de un nuevo partido añadido por medio de dos mecanismos: unos slider de las componentes R, G y B o introduciendo directamente la referencia hexadecimal. Hace uso del VM *ColorPicked* para formar los colores. Contiene dos botones de aceptar y cancelar y las propiedades *brush* y *color* a las que accede la ventana *WindowDatos* que invoca este cuadro de diálogo (de modo Modal).

LogoAnimado

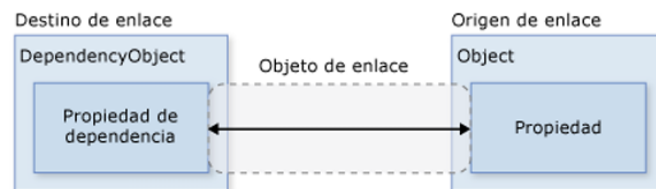
Pantalla de carga mostrada en el arranque de la aplicación. Tiene una animación del logo de tipo “From-To” controlada por eventos, tras la que se lanza la ventana principal al cabo de los 6 segundos de un *DispatcherTimer*. Cada vez que termina la animación de uno de los rectángulos, se lanza el evento *Copleted* y se pinta el siguiente.

3.3. Relaciones y mecanismos de comunicación

La arquitectura de la aplicación está basada en el patrón Modelo – Vista – Modelo de la Vista, o MVVM por sus siglas en inglés, lo que implica que la comunicación entre los distintos niveles por medio de la invocación de métodos se realiza únicamente en el sentido mostrado, mientras que en el contrario se hace a través de eventos.



Para simplificar las interacciones entre Vista y Modelo de la Vista se emplea el enlace de datos, que permite una representación más flexible y separar la lógica de negocio de la interfaz de usuario.



Para notificar los cambios producidos, las clases del VM implementan *INotifyPropertyChanged*, que avisa a la Vista por medio de eventos. Estas clases son *DataSelected*, que contiene todas las colecciones de elecciones o partidos; y otras secundarias como *PartidosAdded*, *CheckData* y *ColorPicked*, para los datos temporales del proceso de adición de elecciones.

Por su parte, las clases de la Vista se suscriben a estos eventos que lanza la interfaz *INotifyPropertyChanged* cuando se produce un cambio y actúan en función de lo sucedido con un método concreto como se muestra a continuación, en el que la ventana de gráficos se actualiza pintando el correspondiente a la nueva Eleccion seleccionada desde el listado.

```
//EVENTOS
2 referencias
private void S_PropertyChanged(object sender, PropertyChangedEventArgs e)
{
    if (e.PropertyName == "Eleccion")
    {
        flagData = true;
        posicionesPactometro.Clear();
        flagPactometro = false;
        dibuja();
    }
}
```

Por último, para comunicar al Modelo con el View Model, se definen unos EventArgs personalizados y definidos en el fichero CustomEventArgs. Son dos:

- *SelecciónEventArgs*, que envía la nueva elección seleccionada en la ventana de Listados. Esta se recibe a través de la invocación del método *ListaEleccionesSelection* y la vuelve a recibir el VM, que se encuentra suscrito con el método *WL_nuevaSeleccion*. Aquí almacena dicho objeto en la clase *DataSelected* explicada anteriormente que notifica a la vista con *INotifyPropertyChanged*.

```
1 referencia
private void WL_nuevaSeleccion(object sender, SeleccionEventArgs e)
{
    s.Eleccion = e.eleccion;
}

1 referencia
public void ListaEleccionesSelection(object o)
{
    m.ListaEleccionesSelection((Eleccion)o);
}
```

- *AddEventArgs*, que envía la colección actualizada de todas las elecciones almacenadas en el sistema. Se lanza al modificarse la colección del Modelo, cuando se añade una nueva y el VM la recoge y actualiza en *DataSelected* por medio del método al que está suscrito.

```
4 referencias
public class SeleccionEventArgs : EventArgs
{
    private Eleccion e;
    2 referencias
    public Eleccion eleccion { get { return e; } set { e = value; } }
    1 referencia
    public SeleccionEventArgs(Eleccion eleccion)
    {
        this.eleccion = eleccion;
    }
}

4 referencias
public class AddEventArgs : EventArgs
{
    private ObservableCollection<Eleccion> elecciones;
    2 referencias
    public ObservableCollection<Eleccion> Elecciones
    {
        get { return elecciones; }
        set { elecciones = value; }
    }
    1 referencia
    public AddEventArgs(ObservableCollection<Eleccion> elecciones)
    {
        this.Elecciones = elecciones;
    }
}
```

4. Referencias bibliográficas

Puntos de interrupción en Visual Studio:

<https://learn.microsoft.com/es-es/visualstudio/debugger/using-breakpoints?view=vs-2022>

Enum en ComboBox:

<https://stackoverflow.com/questions/6145888/how-to-bind-an-enum-to-a-combobox-control-in-wpf>

Color con código hexadecimal:

<https://stackoverflow.com/questions/2109756/how-do-i-get-the-color-from-a-hexadecimal-color-code-using-net>

C# Dictionary:

<https://stackoverflow.com/questions/1273139/c-sharp-java-hashmap-equivalent>

Idea de aplicación multilenguaje: Programación III con Álvaro Lozano

Cambiar un diccionario dinámicamente:

<https://stackoverflow.com/questions/33867091/how-to-change-resourcedictionary-dynamically>

Resource Dictionary en WPF:

<https://stackoverflow.com/questions/11327840/multilingual-wpf-application>

Insertar un rectángulo en ListView:

<https://stackoverflow.com/questions/40240385/listview-coloring-of-rectangle-in-viewbox-wpf>

Singleton: Ingeniería del Software II con Jesús Fernando Rodríguez Aragón

Implementación del patrón Singleton en C#:

<https://albertcapdevila.net/patron-diseno-singleton-csharp/>

YesOrNo MessageBox:

<https://blog.harshgarg.com/2014/06/use-of-messageboxresult-in-c-and-vbnet.html>

Obtener el tamaño de la ventana:

<https://stackoverflow.com/questions/11013316/get-the-height-width-of-window-wpf>

Voltear el canvas:

<https://stackoverflow.com/questions/1185948/how-to-change-x-y-origin-of-canvas-to-bottom-left-and-flip-the-y-coordinates>

Regex: Informática Teórica con Álvaro Lozano

Implementación de regex en C#:

<https://es.stackoverflow.com/questions/33277/como-eliminar-caracteres-en-una-cadena-string-en-c>

Insertar en una colección en una posición concreta:

[https://www.geeksforgeeks.org/c-sharp-insert-an-element-into-collection-at-specified-index/#:~:text=Insert\(Int32%2C%20T\)%20method,int%20index%2C%20T%20item\)%3B](https://www.geeksforgeeks.org/c-sharp-insert-an-element-into-collection-at-specified-index/#:~:text=Insert(Int32%2C%20T)%20method,int%20index%2C%20T%20item)%3B)

String Resources desde C#:

<https://learn.microsoft.com/es-es/dotnet/desktop/wpf/advanced/how-to-use-a-resourcedictionary-to-manage-localizable-string-resources?view=netframeworkdesktop-4.8>

Salto de línea en String Resources:

<https://andresledo.es/csharp/salto-de-linea-c/>

Colores en un Resource Dictionary:

<https://stackoverflow.com/questions/53759536/wpf-binding-color-in-resourcedictionary>

Ordenar una lista:

<https://stackoverflow.com/questions/1976546/how-to-sort-a-list-collection-of-classes-with-properties-in-them>

OpacityProperty de una label:

<https://stackoverflow.com/questions/27744097/wpf-fade-out-animation-cant-change-opacity-any-more>