**Interfaces Graficas de Usuario**

**Aplicación para la representación gráfica de funciones en WPF**

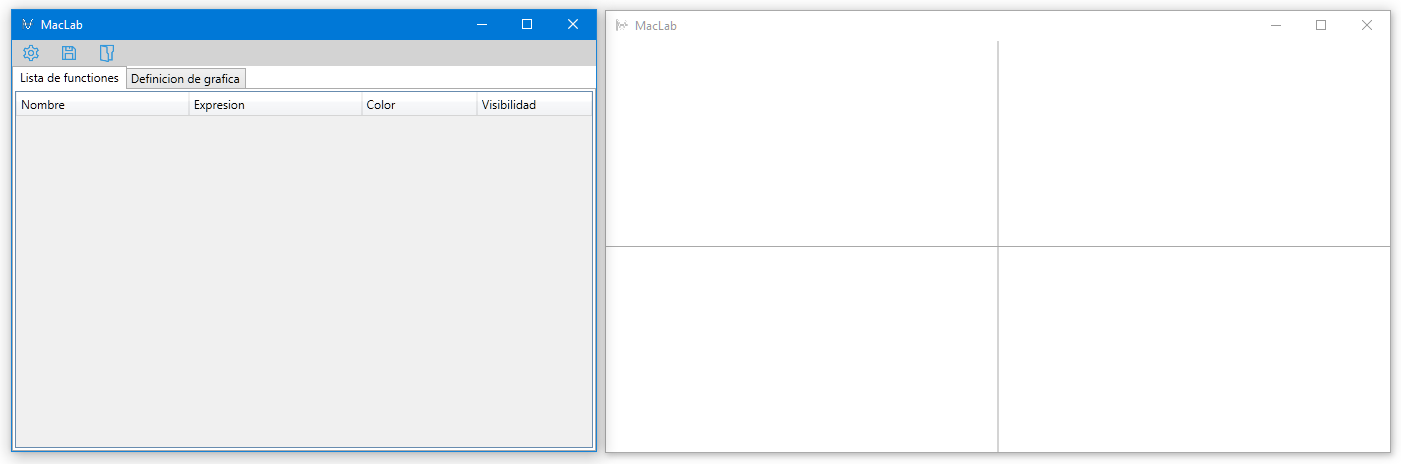
Francisco Pinto Santos

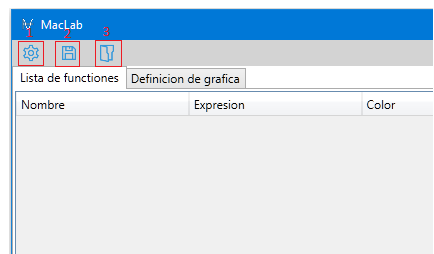
Curso 2018-19

**Manual de usuario**

**Ventanas**

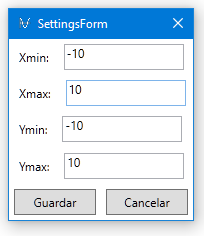
Al abrir el programa se desplegarán 2 ventanas:



La ventana mostrada a la izquierda en la figura (referida a partir de ahora como ventana de definición de representación) se utilizará para trabajar sobre la representación. Y en la ventana derecha (referida a partir de ahora para como ventana de representación) aparecerá la representación.

En la ventana de definición de representación en la parte superior se encuentran 3 botones.

* El botón identificado como 1 en la imagen superior, se utilizará para desplegar el menú de configuración de representación (se explicará más adelante).
* El botón identificado como 2, se utilizará para guardar el proyecto actual en un fichero de extensión “.maclab”.
* Por último, el botón 3 se utilizará para importar un proyecto “.maclab”, lo cual ocasionará el borrado del progreso actual.

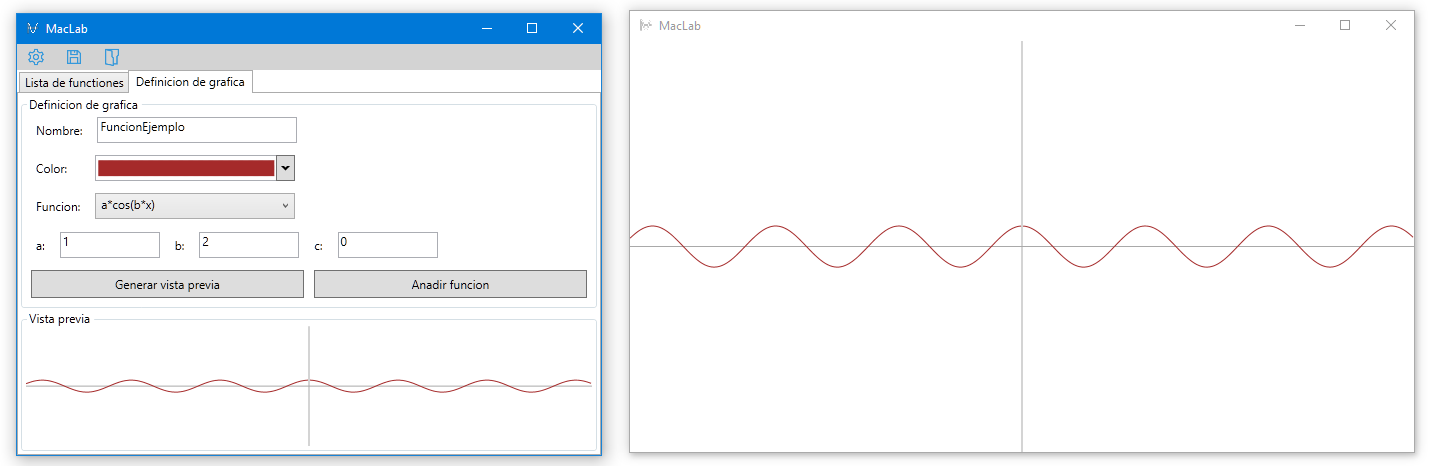


El Botón de configuración mencionado anteriormente desplegará la siguiente ventana, en la cual se podrá elegir el rango de valores de las variables X e Y, visibles en la ventana de representación.

**Añadir grafica**

En la ventana de definición de la representación hay dos pestañas: “Lista de funciones” y “Definición de gráfica”.

La pestaña “Definición de grafica”, presentada en la imagen inferior, será utilizada para añadir graficas a la representación:



En esta pestaña se presentan dos partes:

Un formulario llamado “Definición de grafica” donde se podrán definir los valores de la grafica a representar. Entre los cuales se encuentran Nombre, Color, función a representar y valores de las constantes implicadas en la función.

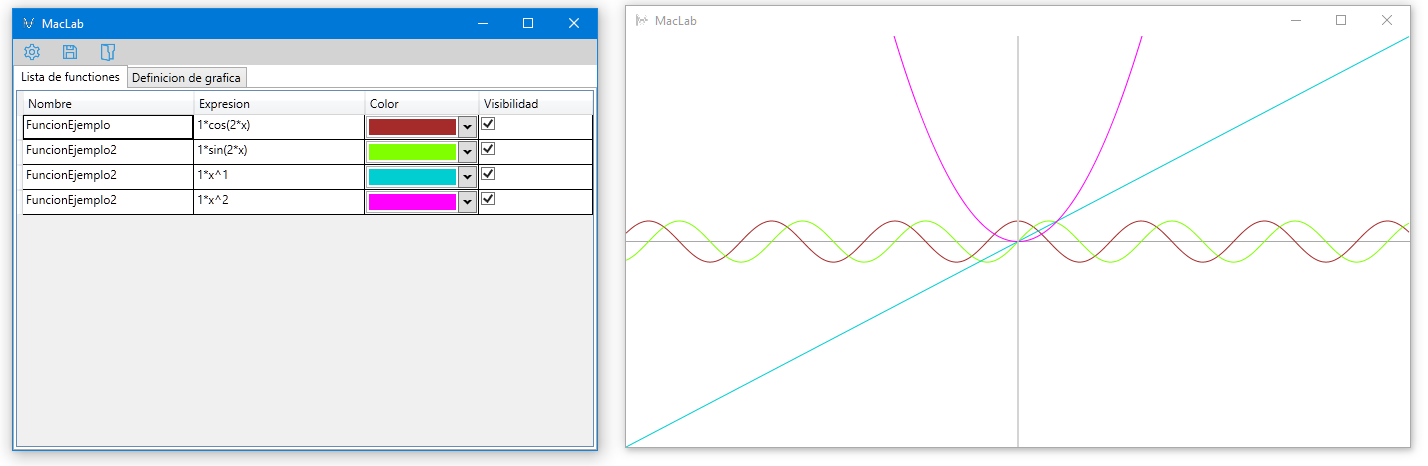
Y otra parte conocida como “Vista previa” en la cual se mostrará una vista previa de la función a añadir.

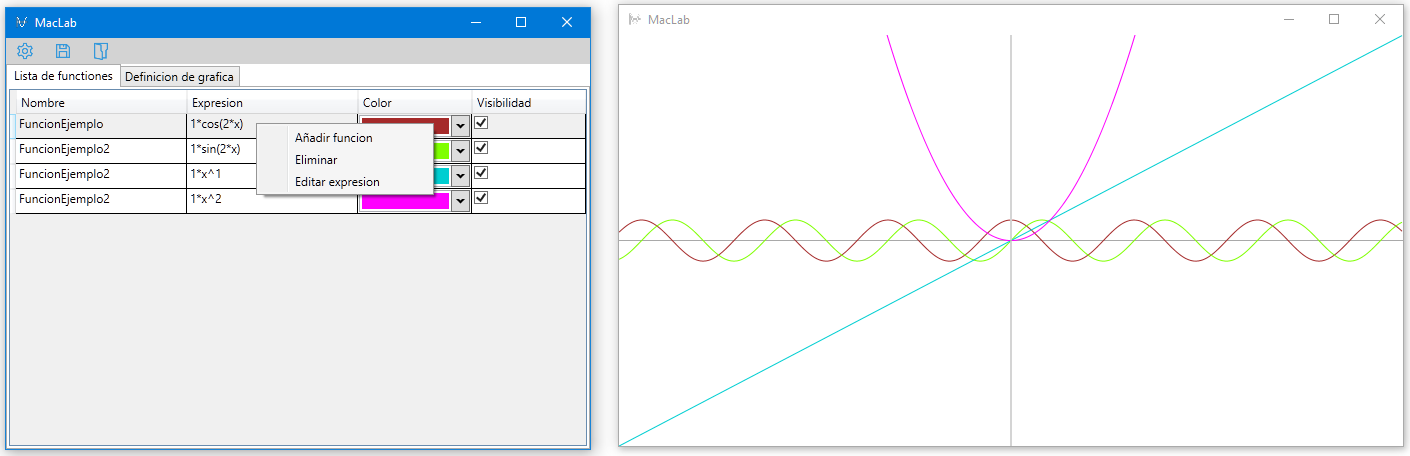
También se encuentran 2 botones:

* Generar vista previa: genera una vista previa de la función definida en el formulario superior en el panel de vista previa.
* Añadir función: añade la función a la ventana de representación.

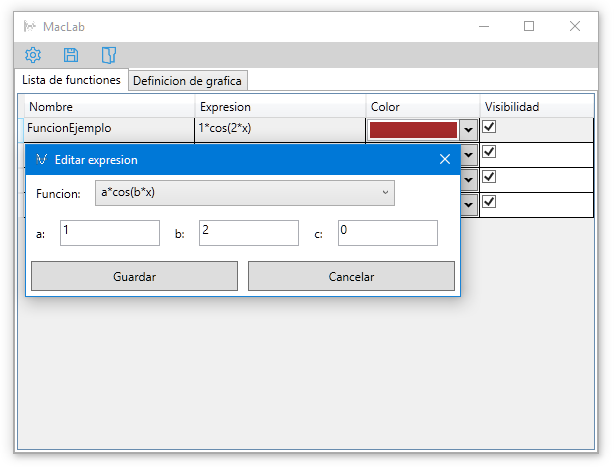
**Gestionar funciones:**

Para gestionar las funciones añadidas tenemos el panel “Lista de funciones”, en el cual como se ve en la imagen inferior se mantiene una lista de funciones activas, de las cuales se podrán modificar aspectos como el color, visibilidad y nombre.

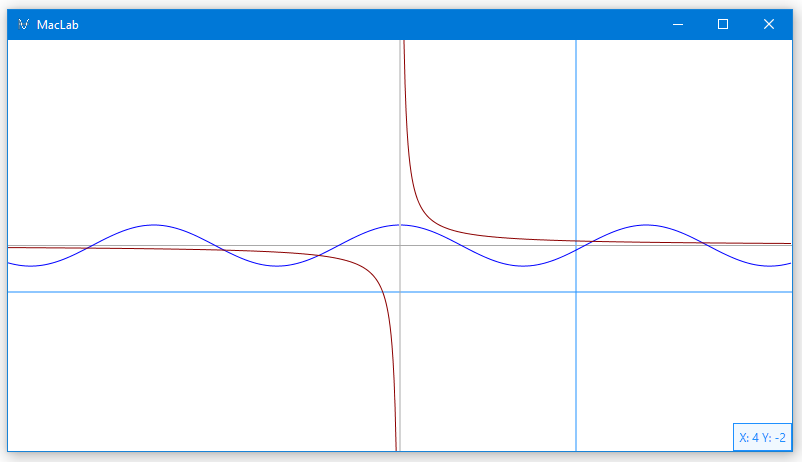


En esta ventana se muestra un menú contextual en el cual , como se aprecia en la imagen inferior:

* Añadir función: Desplaza el programa al pestaña “definición de grafica”.
* Eliminar: Si se realiza clic sobre una función, se eliminará de la representación y la lista de funciones activas.
* Editar expresión: Si se realiza clic sobre una función, nos desplegara la ventana que se muestra en la imagen inferior, con la cual se podrán editar los aspectos relacionados con la expresión que define la grafica correspondiente a esa función.



**Trabajando con la representación:**

En el panel de representación según se pasa el cursor por encima, en la esquina inferior izquierda aparece una etiqueta con la posición en la que se encuentra el cursor y unos ejes centrados en la posición de este.

Por último, cabe destacar la presencia de un menú contextual en esta ventana, con la única opción de exportar imagen; con la cual se nos permitirá exportar la representación actual como una imagen png.

**Manual de programador**

Se explicará la estructura y funcionamiento del programa dividido por paquetes.

Cabe destacar que la aplicación esta basada en un VMMV, en el cual la lógica de negocio se ha encapsulado en el paquete “services”.

La razón principal de uso del patron VMMV es porque facilita trabajar con WPF, y sincronizar varias ventanas.

**Modelo**

Se encuentra en el paquete “models”.

Aquí se encuentra la funcionalidad central de la aplicación, y esta dividida en tres subpaquetes:

* Model: aquí se encuentran a su vez 3 elementos:
  + IObservableModel: Es una interfaz con tipos parametrizados que define un modelo basado en gestión de IDs, y provee las operaciones básicas de CRUD, además de la operaciones de “Clear“ y “GetAllElements“. La peculiaridad que tiene es que trabaja con una “ObservableCollection“, por eso ofrece un método “GetAllElementsForBinding“, lo que hará posible el que se saquen los elementos que tiene almacenado el modelo para poder hacer un enlace con cualquier control.
  + IObservableModelImpl: Es una clase que implementa la interfaz “IObservableModel“ para tipos parametrizados también.
  + IModelable: Interfaz que tiene que implementar todo elemento que quiera ser almacenado en un “IObservableModel“, que tiene 3 funciones básicas para tratamiento de objetos con IDs que son GetID(), SetID() y Clone().
* ViewModel: en este paquete se implementa el modelo de la vista, para lo cual se han utilizado 3 elementos:
  + IViewModel: Es una interfaz con tipos parametrizados que provee las mismas operaciones que “IObservableModel“, pero a su vez tiene una serie de eventos básicos que son lanzados cuando un elemento es creado, eliminado o modificado.

Además, en este fuente están definidos el delegado “ViewModelEventHandler“ (que es el usado para definir los eventos recién mencionados), y la clase “ViewModelEventArgs“, la cual es usada como parámetro de argumentos para los eventos del modelo, y contiene el elemento afectado por la acción que ha disparado el evento.

Por último, sobre esta interfaz cabe destacar que se sustenta sobre un “IObservableModel“, por eso cualquier elemento que se quiera introducir en este modelo debe implementar “IModelabe“.

Además, los objetos que entren en este modelo deben implementar la interfaz “INotifyPropertyChanged“ porque si van a ser utilizados para hacer un enlace con la vista, cuando se cambie una propiedad, buscamos que se provoque un evento que haga al modelo de la vista lanzar otro evento del tipo “ElementUpdated“.

* + IViewModelImpl: Implementa la interfaz “IViewModel“ con tipos parametrizados y se sustenta sobre un “IObservableModelImpl“.
  + FunctionViewModelImpl: Es un “IViewModelImpl“ para funciones, además de que añade la gestión propiedades de representación de la gráfica ( estas están almacenadas en un “struct RepresentationParameters“, definido en el mismo fuente).

Añade un evento “RepresentationParametersChanged” que se dispara cuando algún parámetro de representación cambia

* POJO: Aquí se encuentran los objetos planos, en este caso, únicamente la clase “Function“ que es el objeto utilizado para guardar la información de las funciones.
  + Tiene 5 propiedades que son ID, Name, Color, Calculator y IsVissible.
  + Están presentes en el varias etiquetas “ [JsonIgnore] “ y “ [JsonConstructor] “ que son para indicarle a la biblioteca Json.Net como serializar y deserializar el objeto.
  + Implementa la interfaz “IModelable“ de la que hemos hablado antes por lo que presenta los métodos “GetID()“, “SetID()“ y “Clone()“.
  + Implementa la interfaz “INotifyPropertyChanged“, luego por cada propiedad que se desea enlazar ( todas menos ID), se tiene otra propiedad llamada “NombrePropiedadProperty”. Además, tiene la función “OnPropertyChanged “ para cumplir con las normas de estilo de los eventos.

**Funcionalidad**

Se encuentra en el paquete “services”.

**Utilidades**

Se encuentra en el paquete “utils”.

**Vista**

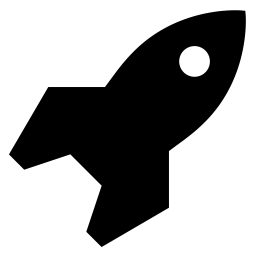
Se encuentra en el paquete “view” y en la raíz (MainWindow).

**Referencias y bibliotecas usadas**

**Bibliotecas usadas**

He usado 2 bibliotecas para realizar este trabajo. Ambas instaladas con el gestor de paquetes NuGet de Visual Studio.

Json.Net

La cual ha sido usada para serializar y deserializar con JSON los ficheros .maclab, ya que estos son un array de las funciones creadas por el usuario serializado a JSON.

https://www.newtonsoft.com/json

Wpf extended toolkit

Esta biblioteca ha sido usada para tomar el control “ColorPicker”, el cual ha se ha utilizado para usar seleccionar el color de las funciones en la tabla de funciones y en el la parte de definición de función.

https://github.com/xceedsoftware/wpftoolkit

**Recursos bibliográficos**

Libros consultados

Apuntes de la asignatura Interfaces Graficas de usuario.

Webs visitadas

[csharphelper.com](http://csharphelper.com) – he utilizado como ejemplo una serie de publicaciones que tienen sobre como dibujar graficas en C#

[stackoverflow.com](https://stackoverflow.com/) – varias páginas para resolver desde dudas básicas sobre C# hasta como utilizar controles complejos

[icons8.com](https://icons8.com) – de aquí tome los iconos que necesite para la aplicación