# Descripción General

Este módulo contiene toda la interfaz visual que se utiliza en el sistema duetto Sound Lab. Su programación se basa principalmente en la utilización de las clases del framework PyQt para la manipulación de los eventos, ventanas, diálogos y funcionamiento general. La librería pyqtgraph es utilizada para la implementación de varios de los controles encargados de graficar las señales que integran el sistema así como para redefinir algunas clases de PyQt.

Módulo Graphic\_Interface

Documentación para el programador

# Sub-módulos

El módulo Graphic Interface publica los siguientes módulos:

* Dialogs 🡪 Las ventanas de diálogo que se muestran en las distintas funcionalidades del sistema.
* UI Files 🡪Los ficheros .ui que se generan por el framework PyQt para la generación de ventanas.
* Widgets 🡪 Los distintos controles visuales que se implementan o redefinen en la aplicación.
* Windows 🡪 Las distintas ventanas que componen el sistema.

A continuación se profundizará en cada módulo.

# Módulo Dialogs

## **Descripción General**

Este módulo define todas las ventanas de diálogos que se utilizan en el sistema para el ingreso de parámetros a las funcionalidades.

## **Clases**

1. Change Volume 🡪 parámetros para la modificación de volumen o amplitud.
2. Filter 🡪 parámetros para el filtrado de la señal en el dominio de la frecuencia.
3. Insert Silence 🡪 parámetros para insertar silencio.
4. New File 🡪 parámetros para la creación de una nueva señal.
5. ElemDetectSettingsDialog 🡪 parámetros para la detección de segmentos.
6. EditcategoriesDialog 🡪 edición de las opciones de clasificación. Permite editar las categorías y los valores de cada categoría de clasificación.

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

A excepción de ElemDetectSettingsDialog el resto de los diálogos solo son utilizados como simple vía de ingreso de parámetros y no poseen ninguna dificultad en su implementación.

## **Clase ElemDetectSettingsDialog**

Descripción

Este diálogo permite la definición de los parámetros que se utilizarán para la segmentación de la señal. Permite ….

***Variables***

***Métodos***

***Invariantes, Reglas y Convenciones***

# Módulo UI Files

## **Descripción General**

En este módulo se almacenan los recursos (íconos, fotos) de la aplicación y los ficheros .ui de las ventanas del sistema. No se relaciona directamente con el resto de la aplicación y puede ser excluido del producto final. Solo se incluye como parte de este módulo de interfaz visual con fines organizativos.

# Módulo Widgets

## **Descripción General**

En este módulo se ubican todos los controles visuales desarrollados como parte de la implementación del sistema. Se asume la programación de los controles como modular e independiente aunque existen dependencias entre ellos para su uso en el software. En principio cada control es posible reutilizarlo y extenderlo para dotarlo de nuevas funcionalidades.

## **Clases**

1. HorizontalHistogramWidget 🡪 extiende el control histogram de pyqtgraph para su visualización de forma horizontal.
2. HorizontalHistogramItem 🡪item relacionado con el control anterior.
3. SpectrogramPlotWidget 🡪 control que grafica el espectrograma. Se basa en reutilización de clases de pyqtgraph
4. SpecYAxis 🡪 clase que se integra con el control anterior. Extiende la funcionalidad de los axis de pyqtgraph.
5. OscilogramPlotWidget 🡪 control que grafica el oscilograma. Extiende la funcionalidad del control PlotWidget de pyqtgraph.
6. QSignalVisualizerWidget 🡪 es el control más importante del sistema ya que integra los demás controles y brinda la mayoría de las funcionalidades. Este control permite la manipulación de una señal de audio. Utiliza dos controles internos OscilogramPlotWidget y SpectrogramPlotWidget para graficar oscilograma y espectrograma respectivamente. Es el encargado de sincronizar y coordinar los dos controles en el procesamiento de la señal. En un inicio la manipulación de los distintos gráficos se hizo en un solo control (cuando se utilizaba matplotlib) pero al trasladar la aplicación a pyqtgraph se hizo imprescindible separarlos. Gran parte del procesamiento realizado por la aplicación resulta en wrappers a métodos en este control. Debe ser posible utilizarlo independientemente y ser exportado para su uso por terceros. Se planea incluirlo en el API visual de duetto como la vía para la manipulación de señales. Debe poseer una amplia documentación.
7. OscXAxis 🡪clase que se integra con el control anterior.
8. OscYAxis 🡪 clase que se integra con el control anterior
9. Tools 🡪 enum que define las herramientas que los usuarios podrán utilizar en el sistema. Se utiliza en el control anterior y en los controles OscilogramPlotWidget y SpectrogramPlotWidget.
10. RectROI 🡪 ?
11. UndoRedoManager,UndoRedoActions 🡪 clases para proveer al control principal de la funcionalidad de undo y redo para las acciones de edición sobre la señal.
12. EditCategoriesWidget 🡪 Widget que permite visualizar una categoría de clasificación con sus valores. Es usado manualmente al invocar el diálogo de editar las categorías de clasificación. Se adiciona manualmente tantos widgets al diálogo como categorías existan. Se utiliza relacionado con una instancia de ClasificationData se utiliza en SegmentationClasifficationWindow y en TwoDimensional..

## **Clase HorizontalHistogramWidget**

Descripción

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase HorizontalHistogramItem**

Descripción

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

## **Clase SpectrogramPlotWidget**

Descripción

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase SpecYAxis**

Descripción

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase OscilogramPlotWidget**

Descripción

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase QSignalVisualizerWidget**

Descripción

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase OscXAxis**

Descripción

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase OscYAxis**

Descripción

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase RectROI**

Descripción

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase UndoRedoManager**

**Descripción**

Esta clase manipula las acciones de undo y redo en el sistema. Es una estructura de datos en forma de pila que no elimina los elementos al extraerlos. Dado que las acciones realizadas conviene almacenarlas para poder deshacerlas y este funcionamiento es propio de una pila (las últimas acciones se deshacen primero) también es conveniente no eliminarlas para poder rehacerlas. Entonces es necesaria una estructura en forma de pila que pueda recorrerse en ambos sentidos. Esta clase implementa este manipulador de acciones mediante una lista y un puntero para la posición de la última acción realizada. Para su utilización en el sistema se asume que el control principal posee una instancia de esta clase y que antes de cada acción de edición sobre la señal se le adiciona la correspondiente acción para undo y redo.

***Variables***

1. actionList (list) 🡪 La lista en la que se almacenan las acciones.
2. actionIndex (int) 🡪 El índice de la lista que apunta a la última acción realizada.
3. actionExec (signal) 🡪 La señal que se emite cuando se realiza una acción de undo o redo

***Métodos***

1. undo() 🡪 Deshace (si existen acciones por deshacer) la última acción de edición realizada sobre la señal. Si se ejecuta la acción se emite la señal actionExec.
2. redo() 🡪 Rehace (si existen acciones por rehacer) la última acción de edición realizada sobre la señal. Si se ejecuta la acción se emite la señal actionExec.
3. clearActions() 🡪 Borra las acciones que se han almacenado en la estructura.
4. addAction(UndoRedoAction) 🡪 Adiciona una nueva acción.
5. count() 🡪 Devuelve la cantidad de acciones que están almacenadas en la estructura.

***Invariantes, Reglas y Convenciones***

1. count() == 0 si no hay acciones adicionadas o se ha invocado a clearActions
2. count == cantidad de elem ¡= None en la lista actionList
3. actionList tiene tantos elem ¡= None como acciones en la estructura.
4. actionList tiene mayor o igual capacidad que la cantidad de acciones (count())
5. actionIndex apunta a la acción realizada anteriormente.
6. actionIndex == -1 si count == 0
7. si se hace redo y actionIndex < count -1 se incrementa en 1 actionIndex y se emite la señal actionExec.
8. si se hace undo y actionIndex > 0 se disminuye en 1 actionIndex y se emite la señal actionExec.
9. Count = count +1 luego de addAction

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase UndoRedoAction**

Descripción

Es una clase auxiliar para utilizarla en conjunto con la anterior. Contiene dos variables callables o dos métodos undo y redo. Las distintas acciones de undo y redo heredan de esta clase y redefinen

***Variables***

1. undo (callable) 🡪 método que se invoca cuando se hace undo a esta acción
2. redo(callable) 🡪 método que se invoca cuando se hace redo a esta acción***Métodos***

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

# Módulo Windows

## **Descripción General**

En este módulo se incluyen todas las ventanas que se utilizan en el sistema. Las ventanas de procesamiento están directamente vinculadas con los controles del módulo anterior y se relacionan con ellos para brindar las funcionalidades.

## **Clases**

1. DuettoSoundLabWindow 🡪 ventana principal de la aplicación. Procesamiento y edición de señales.
2. TwoDimensionalWindow 🡪 ventana de visualización de procesamientos bidimensionales.
3. SegmentationAndClasificationWindow 🡪 Ventana que permite segmentar y clasificar la señal.
4. SerializedData 🡪 estructura para hacer persistente a disco las opciones visuales del sistema para una futura utilización.
5. DuettoListParameterItem 🡪 clase para la ordenación de los parámetros de algunas opciones visuales en la ventana DuettoSoundLabWindow

## **Clase DuettoSoundLabWindow**

Descripción

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase TwoDimensionalWindow**

**Descripción**

Esta ventana es utilizada en la ventana de segmentación y clasificación. Luego de segmentar y extraer parámetros. La ventana permite representar y graficar bidimensionalmente dos de las magnitudes medidas. Se utiliza un scatter plot para el gráfico de los elementos. La ventana permite analizar los segmentos que se detectaron en la ventana **SegmentationAndClasificationWindow.** Mediante esta ventana se pueden visualizar los parámetros medidos en cada segmento. Contiene un Parameter Tree en el que se insertan las opciones que brinda la ventana. Allí se puede encontrar las dos listas de parámetros (cada una con las mediciones realizadas) en las cuales se pueden seleccionar cual parámetro se graficará en cada eje (X e Y). Se pueden modificar de la misma manera algunas opciones visuales como la figura de los elementos bidimensionales (cuadrados, rombos etc), el color, la fuente para el texto de los ejes etc.

***Variables***

1. elementSelected (signal) 🡪 Esta señal es lanzada cuando se selecciona un elemento bidimensional del gráfico. Es la primera parte de la comunicación entre la ventana de segmentación y esta ventana. Permite avisar cuando se selecciona un elemento desde esta ventana.
2. elementsClasification (signal) 🡪 Esta señal es lanzada cuando se han seleccionado múltiples objetos del gráfico y se les han asignado categorías dentro de la clasificación. Es la implementación de la clasificación manual.
3. scatter\_plot (ScatterPLot) 🡪 Es el gráfico que se muestra. Está formado por tantos elementos como hayan sido detectados en la ventana de segmentación.
4. classificationData (ClasificationData) 🡪 Es la instancia de clasificación. En ella se almacena las categorías, los valores de cada categoría, así como los vectores de entrenamiento que son comunes a los métodos de clasificación. Es utilizada para la segmentación manual en esta ventana.
5. Columns (string[]) 🡪 Los nombres de los parámetros medidos.
6. previousSelectedElement (int) 🡪 El índice del último elemento seleccionado. Es utilizado para resaltar con otro color el elemento seleccionado así como para reorganizar visualmente la ventana de segmentación con el objetivo de hacerlo visible en la tabla de parámetros y en el control visual de oscilograma y espectrograma. Cuando se selecciona mediante click un elemento este valor es actualizado y es lanzada la señal elementSelected
7. data (array[,]) 🡪 Array de (cantidad de segmentos)x(cantidad de parámetros medidos) con la información de los parámetros.

***Métodos***

1. plot 🡪 refresca el gráfico bidimensional con los valores contenidos en data. Genera un elemento por cada fila en data y lo posiciona en el grafico según las coordenadas seleccionadas por el usuario. Las coordenadas son los valores de los parámetros medidos que han sido seleccionados para cada eje.
2. on\_actionSaveGraphImage\_triggered 🡪 Permite exportar la imagen del grafico a disco para su futuro uso.
3. on\_actionMark\_Selected\_Elements\_As\_triggered 🡪 Luego de seleccionar varios elementos visuales en la ventana (mediante dibujo de un rectángulo. Haciendo click para fijar la posición inicial y haciendo drag para fijar las dimensiones alto y ancho) este método procede a abrir el diálogo de clasificación manual para que el usuario fije los valores de cada categoría para todos los elementos que se hayan seleccionado.
4. on\_actionHide\_Show\_Settings\_triggered 🡪 Permite ocultar y volver a mostrar el Parameter tree con las opciones. Es conveniente si el usuario quiere observar con mayor comodidad el gráfico.
5. createParameterTreeOptions Método privado que crea el parameter tree con los datos de las mediciones.
6. loadData (columns(string[]), data (array[,])) 🡪Método que actualiza los valores de la ventana porque hubo algún cambio en las mediciones o en los segmentos detectados.
7. load\_Theme 🡪 Carga con las opciones visuales de la aplicación en esta nueva ventana
8. selectElement (index(int)) 🡪 Permite seleccionar el elemento que se encuentra en la posición index. Es la segunda parte de la comunicación entre la ventana de segmentación y esta.
9. deselectElement 🡪 El complemento del método anterior para revertir la selección

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase SegmentationAndClasificationWindow**

***Descripción***

En esta ventana se realiza la segmentación, clasificación y extracción de parámetros sobre la señal. Brinda las funcionalidades de la ventana principal que no modifican la señal. Contiene un control SignalDetectorWidget para la visualización de la señal. Este control extiende las funcionalidades del SignalVisualizer para la segmentación y visualización de segmentos.

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase SerializedData**

Descripción

Esta clase es la utilizada para serializar el tema visual utilizado en la aplicación. Contiene las variables que modifican visualmente la aplicación. Se serializa y recupera mediante el módulo pickle de la librería estándar.

***Variables***

1. osc\_background 🡪color de fondo
2. osc\_plot 🡪 color de linea en el oscilograma
3. osc\_GridX 🡪 líneas de grid coordenadas en el eje X del oscilograma
4. osc\_GridY 🡪 líneas de grid coordenadas en el eje Y del oscilograma
5. spec\_GridX 🡪 líneas de grid coordenadas en el eje X del espectrograma
6. spec\_GridY 🡪 líneas de grid coordenadas en el eje Y del espectrograma
7. minYOsc 🡪 Mínimo valor (% de los valores posibles que puede alcanzar la señal en función de la profundidad de bits) para visualizar en el eje Y del oscilograma.
8. maxYOsc 🡪 Máximo valor (% de los valores posibles que puede alcanzar la señal en función de la profundidad de bits) para visualizar en el eje Y del oscilograma.
9. minYSpec 🡪 Mínimo valor (en Khz según las frecuencias que aparecen representadas en la señal producto de la frecuencia de muestreo) para visualizar en el eje Y del espectrograma.
10. maxYSpec 🡪 Máximo valor (en Khz según las frecuencias que aparecen representadas en la señal producto de la frecuencia de muestreo) para visualizar en el eje Y del espectrograma.
11. pow\_Back 🡪 color de fondo de la ventana de procesamiento unidimensional. Posiblemente deprecated.
12. pow\_Plot 🡪 color de las líneas del gráfico de la ventana de procesamiento unidimensional. Posiblemente deprecated.
13. pow\_GridX 🡪 grid en el eje X para la ventana de procesamiento unidimensional. Posiblemente deprecated.
14. pow\_GridY 🡪 grid en el eje Y para la ventana de procesamiento unidimensional. Posiblemente deprecated.
15. spec\_background 🡪 Color de fondo para el gráfico de espectrograma.
16. colorBarState 🡪 Barra de color para visualizar el espectrograma.
17. histRange 🡪 Rango de umbral
18. endColor 🡪 Opciones de color para uso futuro
19. startColor 🡪 Opciones de color para uso futuro
20. quart1Color 🡪 Opciones de color para uso futuro
21. centerColor 🡪 Opciones de color para uso futuro
22. quart2Color 🡪 Opciones de color para uso futuro

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

## **Clase DuettoListParameterItem**

Descripción

***Variables***

1. …

***Métodos***

1. …

***Invariantes, Reglas y Convenciones***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****