Cool Imaging - Manual de Ayuda

Contents

Manual de Ayuda

Contenido

Éste es el *manual de ayuda* para la aplicación **Cool Imaging** dentro del proyecto Cool Imaging Project. Este manual espera ser un referente para un usuario que desee hacer uso de la aplicación, así como un referente para cualquier desarrollador interesado en ampliar la potencialidad y funcionalidad de la aplicación.

¿Qué es Cool Imaging?

Cool Imaging es una aplicación RCP desarrollada para la **caracterización** y **tratamiento** de imágenes digitales mediante rasgos basados en contenido.

Los campos de acción de esta aplicación son diversos, puesto que cualquier imagen digital que sea suceptible de ser tratada o caracterizada puede procesarse a través de Cool Imaging. Asimismo, puede utilizarse tanto con fines *profesionales*, como *académicos*. Además, gracias al sistema de *ampliabilidad* desarrollado permite ser una aplicación que se adapta al usario, de manera que si la aplicación no satisface las necesidades de éste, él mismo puede desarrollar nuevos algoritmos que se integren en la aplicación.

Licencia

1. Cómo instalar *Cool Imaging*

Requisitos mínimos

La aplicación requiere:

- Java Runtime Environment (JRE)
  versión 1.6 o superior.
- Librería Java Advanced Imaging (JAI)
  versión 1.1.3 o superior.

Instalación en Linux

1. Si no dispone de la librería JAI. Descargar la versión de la librería JAI para el JRE de Linux.
   Seguir los pasos de instalación
2. Descomprimir el archivo con la aplicación coolImaging en la ubicación que se desee.
3. Ejecutar el archivo CoolImaging

Nota.- En caso que no se pueda ejecutar, comprobar que se tienen permiso de ejecución, sino ejecutar en la línea de comandos

```
> chmod u+x CoolImaging
```

Instalación en Windows

1. Si no dispone de la librería JAI. Descargar la versión de la librería JAI para el JRE de Windows.

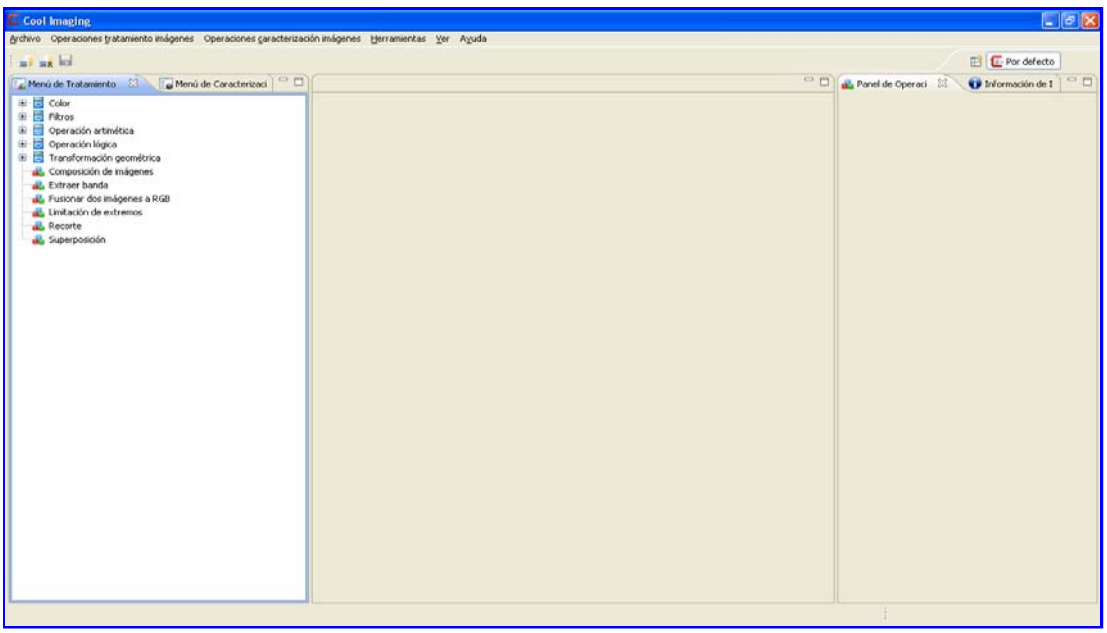
Seguir los pasos de instalación

2. Descomprimir el archivo con la aplicación coolImaging en la ubicación que se desee.
3. Ejecutar el archivo CoolImaging.exe

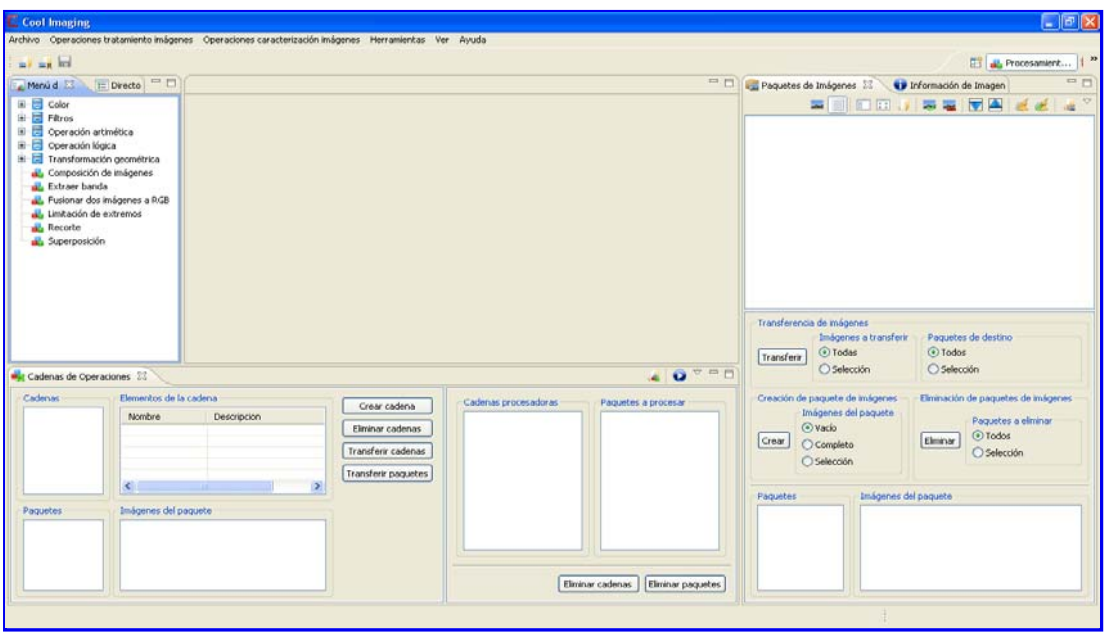
2.1. Procesar una imagen

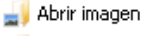
En esta sección se explicará de forma breve como aplicar una operación de tratamiento de imágenes activando previamente la perspectiva de *Procesamiento de Imágenes*. Si desea ayuda más detallada de la aplicación diríjase al *Manual de Usuario* de este mismo manual.

Cuando se arranca la aplicación por primera vez se mostrará la perspectiva por defecto, similar a la que se adjunta en la imagen. Se puede observar como aparece un botón arriba a la derecha de la ventana con el texto "Por defecto", indicando que ésta es la perspectiva activa.



Lo primero que haremos será cambiar la perspectiva. Para esto accederemos al menú Ver ->Cambiar perspectivas->Other... Donde en la ventana que se nos muestra elegiremos *Procesamiento de Imágenes*.

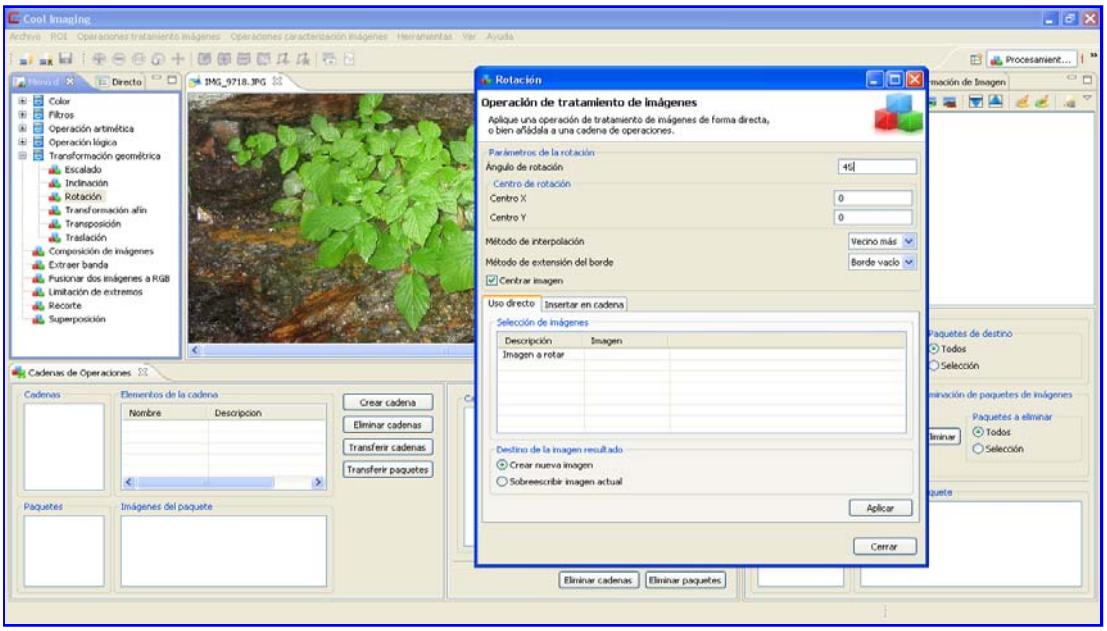


A continuación abriremos la imagen sobre la que queremos aplicar la operación de tratamiento con la acción *Abrir imagen* 

, que se encuentra en el *menú Rápido*, o también en el menú *Archivo* de la *barra de menú*. La imagen

seleccionada se abrirá en el editor.

El siguiente paso será aplicar una operación a la imagen. En este ejemplo aplicaremos la *operación de rotación*. Para ello nos desplazaremos a la vista de menú de Tratamiento de Imágenes y buscaremos en la categoría *Transformación geométrica* la operación de *rotación*. Haremos *click* o *doble click* sobre la entrada para mostrar el panel de la operación en el panel de Operaciones de Tratamiento o en una ventana emergente, como se presenta en la imagen adjunta.

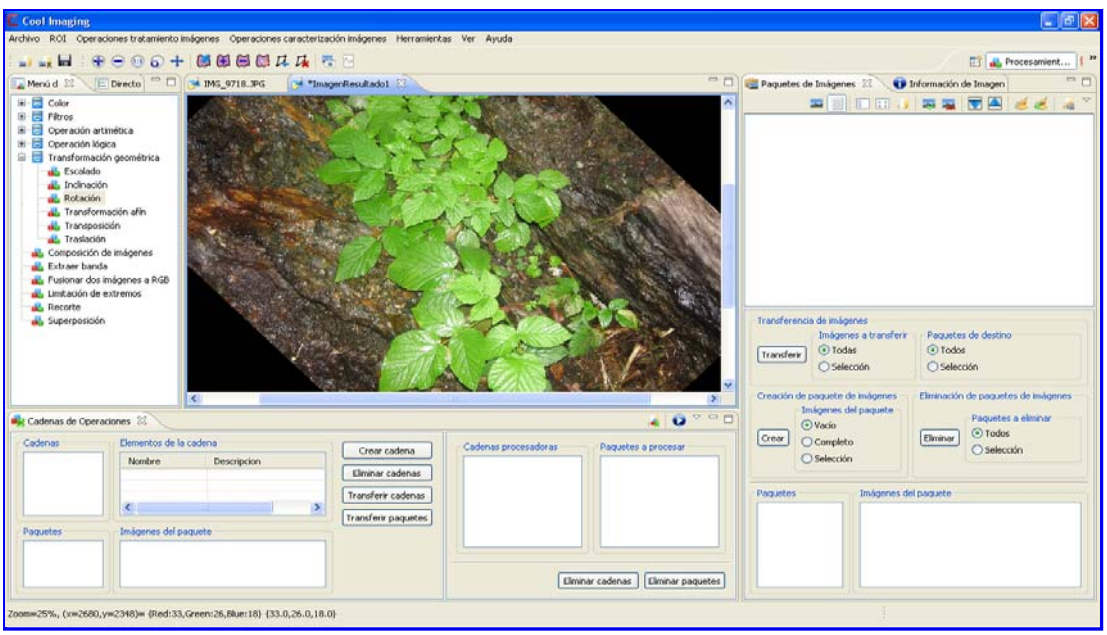


Seguidamente introduciremos los datos que requiere la *operación de rotación* para que se pueda ejecutar. Para el caso de ejemplo estos serán los parámetros utilizados:

- ángulo de rotación = 45
- centro X = 0
- centro Y = 0
- Centro de interpolación = Vecino más cercano
- Método de extensión del borde = Borde vacío

Sólo nos falta elegir las imágenes implicadas en la operación, que en este caso será una imagen. La añadiremos en el apartado "*Selección de imágenes*" donde se muestra una tabla con tantas entradas como imágenes requiera la operación.

Elegiremos la imagen abierta, o dejaremos este campo en blanco, de manera que la operación utilizará la imagen activa del editor. Elegimos "Crear nueva imagen" en el apartado "*Destino de la imagen resultado*" y hacemos *click* en el botón "**Aplicar**" para mostrar la imagen resultado.



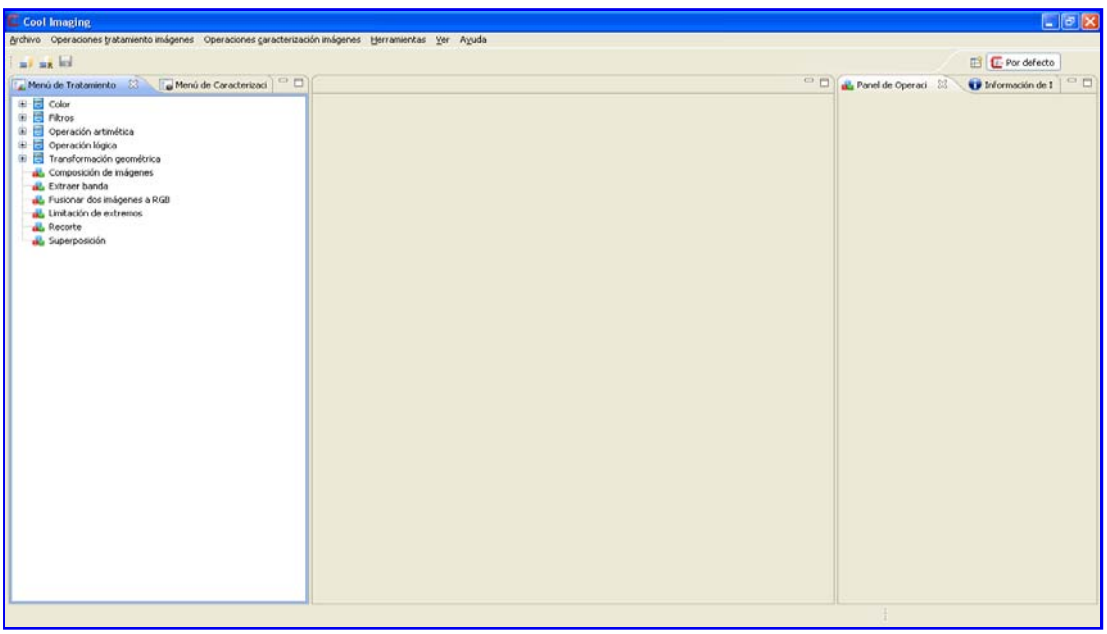
Si quiere seguir profundizando en el uso de esta aplicación puede dirijirse a las secciones Cómo crear y ejecutar cadenas de

operaciones para aprender a aplicar cadenas de operaciones, a la sección Caracterizar una imagen para empezar a aprender a caracterizar imágenes, o a la sección Cómo definir una región de interés para definir regiones de interés (ROI) sobre una imagen.

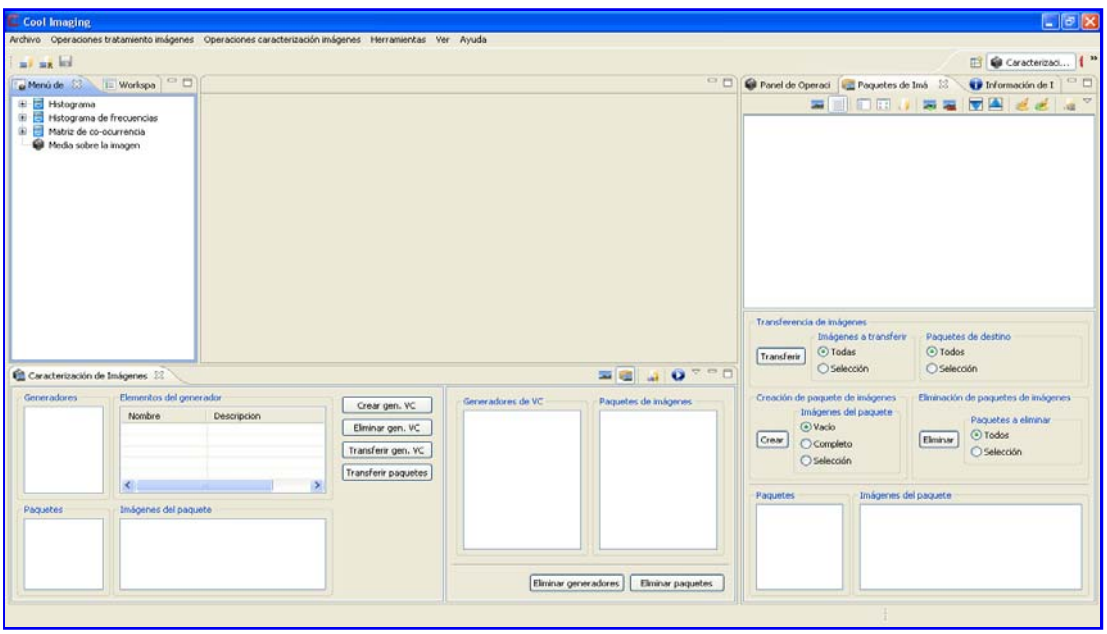
2.2. Caracterizar una imagen

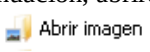
En esta sección se explicará de forma breve como aplicar una operación de caracterización a una imagen activando previamente la perspectiva de *Caracterización de Imágenes*. Si desea ayuda más detallada de la aplicación diríjase al *Manual de Usuario* de este mismo manual.

Cuando se arranca la aplicación por primera vez se mostrará la perspectiva por defecto, similar a la que se adjunta en la imagen. Se puede observar como aparece un botón arriba a la derecha de la ventana con el texto "Por defecto", indicando que esta es la perspectiva activa.



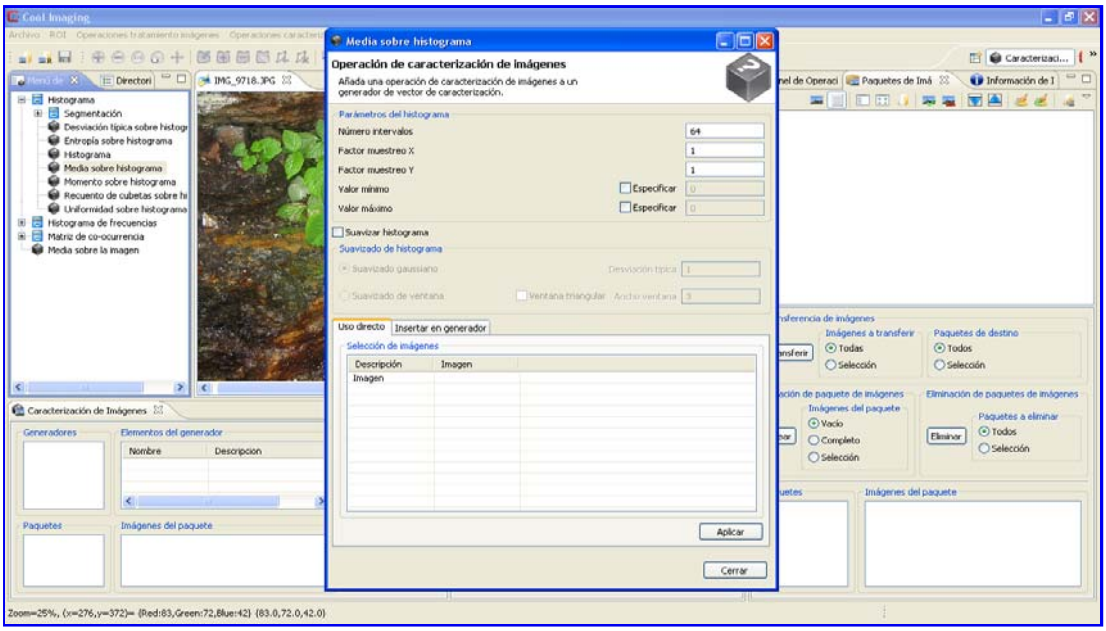
Lo primero que haremos será cambiar la perspectiva. Para esto accederemos al menú Ver ->Cambiar perspectivas->Other... Donde en la ventana que se nos muestra elegiremos *Caracterización de Imágenes*.



A continuación, abriremos la imagen sobre la que queremos aplicar la operación de caracterización con la acción *Abrir imagen* 

, que se encuentra en el *menú Rápido*, o también en el menú *Archivo* de la *barra de menú*. La imagen seleccionada se abrirá en el editor.

El siguiente paso será aplicar la medida a la imagen. En este ejemplo aplicaremos la *Media sobre el histograma*. Para ello nos desplazaremos a la vista de menú de Caracterización de Imágenes y buscaremos en la categoría *Histograma* la operación de *Media sobre el histograma*. Haremos *click* o *doble click* sobre la entrada para mostrar el panel de la operación en el panel de Operaciones de Caracterización o en una ventana emergente, como se presenta en la imagen adjunta.

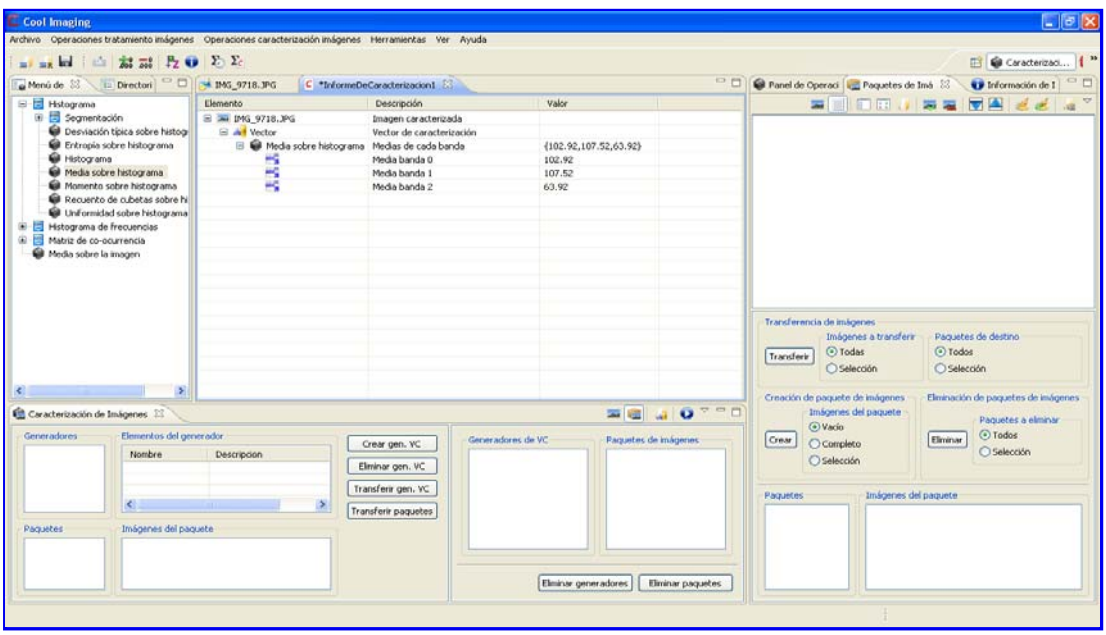


Seguidamente introduciremos los datos que requiere la *Media sobre el histograma* para que se pueda ejecutar. Para el caso de ejemplo estos serán los parámetros utilizados:

- Número de intervalos = 64
- Factor de muestreo X = 1
- Factor de muestreo Y = 1
- Valor mínimo = "sin especificar"
- Valor máximo = "sin especificar"
- Suavizar histograma = "sin suavizar"

Sólo nos falta elegir las imágenes implicadas en la operación, que en este caso será una imagen. La añadiremos en el apartado "*Selección de imágenes*" donde se muestra una tabla con tantas entradas como imágenes requiera la operación.

Elegiremos la imagen abierta, o dejaremos este campo en blanco, de manera que la operación utilizará la imagen activa del editor. Hacemos *click* en el botón "**Aplicar**" para mostrar el informe resultado.



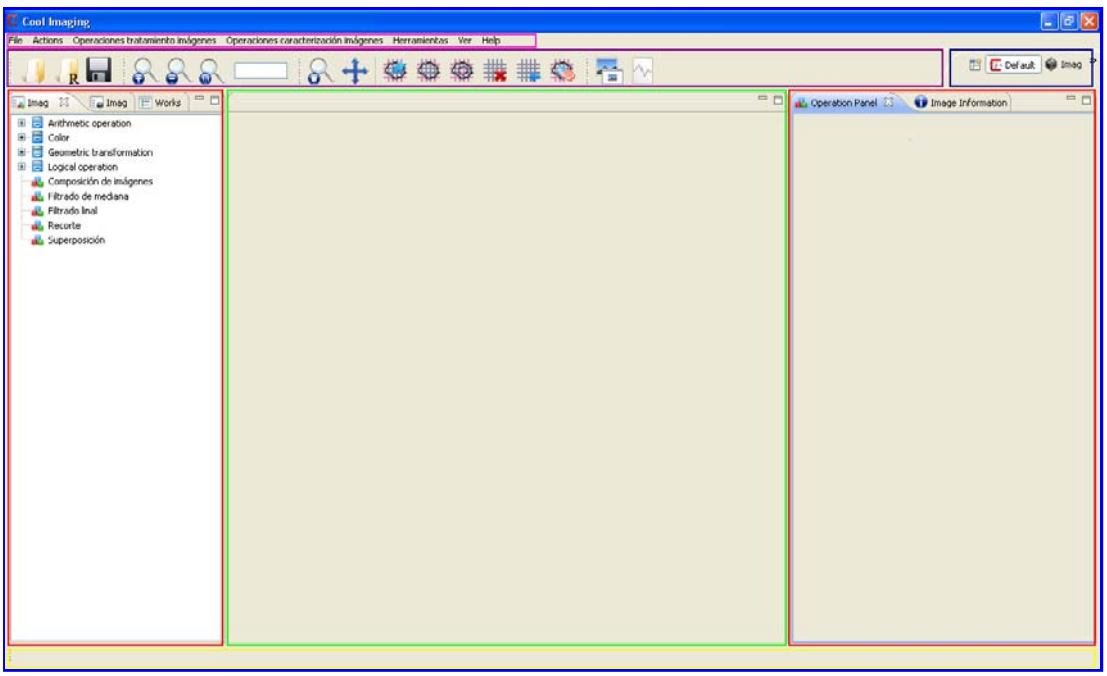
Si quiere seguir profundizando en el uso de esta aplicación puede dirijirse a las secciones Cómo crear y ejecutar vectores de caracterización para aprender a aplicar vectores de caracterización, a la sección Procesar una imagen para empezar a

aprender a procesar imágenes, o a la sección Cómo definir una región de interés para definir regiones de interés (ROI) sobre una imagen.

3.1. Componentes de la aplicación *Cool Imaging*

- Vista
- Editor
- Barra de Menú
- Menú Rápido
- Perspectivas
- Barra de Estado

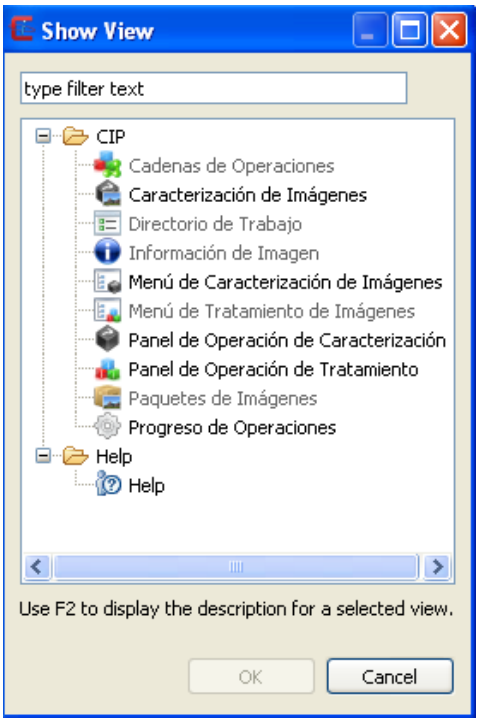
Esta sección pretende dar una visión detallada de cada uno de los componentes que conforman la aplicación.
La ventana principal de la aplicación puede ser dividida en varias secciones diferenciadas que pasaremos a describir a continuación.



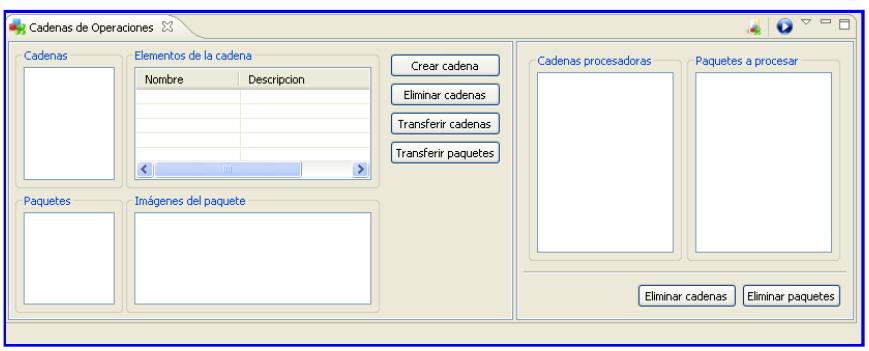
- La región de color *rojo* muestra las vistas. La misión de estas vistas es albergar los menús de operaciones, así como dar al usuario la posibilidad de realizar otras operaciones.
- La región de color *verde* es la zona del editor. Es la zona de trabajo de la aplicación. Esta muestra las imágenes abiertas con las que se puede trabajar y a las que aplicar operaciones, así como los informes de caracterización.
- La región de color *rosa* muestra la barra de menú. Aquí se organizan todas las acciones que se pueden llevar a cabo en la aplicación.
- La región de color *morado* es la zona del menú rápido. Este menú es más reducido y muestra parte de las acciones que se encuentran en la *barra de menú*.
- La región de color *azul* muestra las perspectivas disponibles. Aquí el usuario puede elegir entre varias de ellas, de manera que puede cambiar la apariencia de la aplicación dependiendo de las acciones que quiera llevar a cabo.
- La región de color *amarilla* muestra la barra de estado. Esta barra se utiliza para mostrar información que puede ser relevante para el usuario

3.2. Vista

La misión principal de las vistas es otorgar funcionalidad a una aplicación. Esta sección describe cada una de las vistas existentes en esta aplicación.



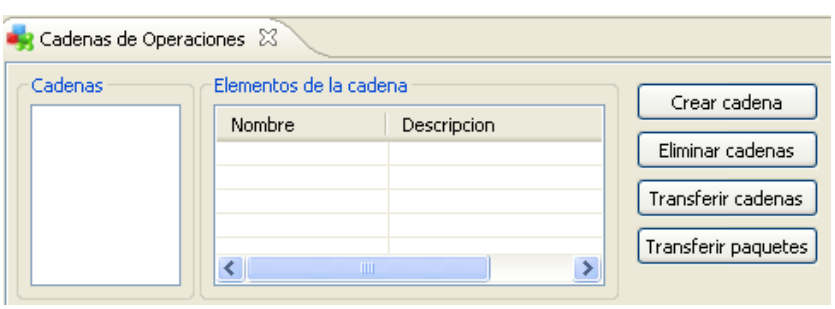
Cadenas de Operaciones



Esta vista permite al usuario definir cadenas de operaciones que se ejecutarán de forma automática a paquetes de imágenes que se definen en la vista paquetes de imágenes.

Esta vista se puede dividir en dos partes: la parte izquierda, que permite **gestionar** (crear, editar y eliminar) *paquetes de imágenes* y *cadenas de operaciones*; y la parte derecha que permite **manipular** los *paquetes de imágenes* y *cadenas de operaciones* para su ejecución.

El panel de *gestión* de paquetes que se ofrece en esta vista es un menú reducido del menú completo que se ofrece en la vista paquetes de imágenes. Mientras que el panel de *gestión* de cadenas de operaciones ofrece toda la funcionalidad necesaria para poder trabajar con cadenas de operaciones, aunque para poder añadir nuevas operaciones a una cadena debe hacerse a través del Panel de Operaciones de Tratamiento.



Crear o eliminar una cadena se hace a través de los botones **Crear cadena** y **Eliminar cadena**, respectivamente.

La parte derecha de la vista se utiliza para la ejecución de los paquetes y cadenas definidas. Estas tareas se pueden llevar a cabo con los botones **Transferir cadenas** o **Transferir paquetes**; o bien arrastrando con el ratón.

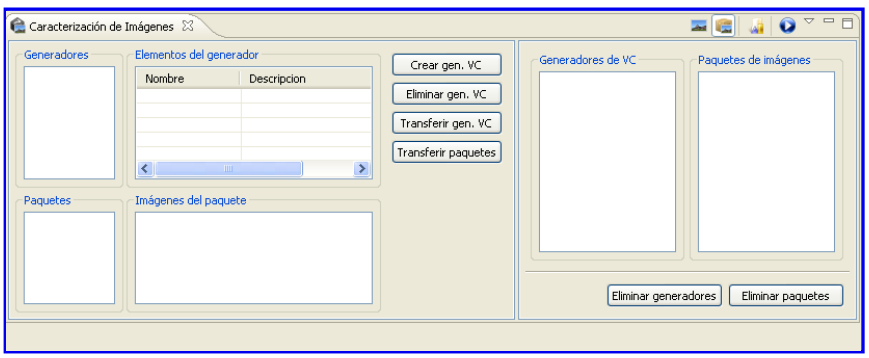
Cuando el usuario defina los paquetes y cadenas que desea ejecutar, deberá hacer *click* sobre el icono  para que se le muestre una ventana emergente donde podrá elegir la ubicación donde guardar las imágenes resultados entre otras opciones. Estas cadenas se ejecutarán en segundo plano, mostrando la progresión en la parte derecha de la barra de estado.

Si el usuario deseara *cargar* de disco una cadena de operaciones, este podría hacerlo a través del icono  que se muestra en la parte superior-derecha de la vista.

Esta vista no sólo ofrece la funcionalidad a través de los botones, sino que también se puede acceder a estas acciones a través del *botón derecho* del ratón.

Si desea saber más, dirijase a la sección cómo crear y ejecutar una cadena de operaciones.

Caracterización de Imágenes

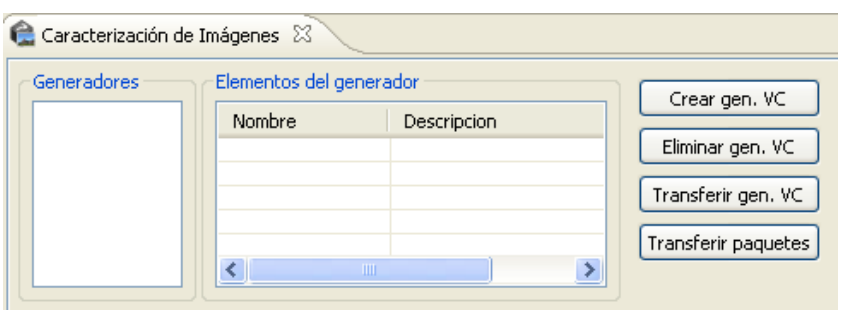


Esta vista permite al usuario definir generadores de vectores de caracterización (generadores VC) que podrá aplicar tanto *imágenes* como a *paquetes de imágenes* que se definan en la vista paquetes de imágenes.

Para aplicar los generadores VC a un **paquete** de imágenes deberá tener activa la opción  . En caso que deseara aplicarlos sobre una **imagen** la opción que deberá tener activa será  . Ambas acciones se encuentran en la parte superior-derecha de la vista.

Esta vista se puede dividir en dos partes: la parte izquierda, que permite **gestionar** (crear, editar y eliminar) *paquetes de imágenes* y *generadores VC*; y la parte derecha que permite **manipular** los *paquetes de imágenes* y *generadores VC* para su ejecución.

El panel de *gestión* de paquetes que se ofrece en esta vista es un menú reducido del menú completo que se ofrece en la vista paquetes de imágenes. Mientras que el panel de *gestión* de generadores VC ofrece toda la funcionalidad necesaria para poder trabajar con generadores VC, aunque para poder añadir nuevas operaciones a un generador debe hacerse a través del Panel de Caracterización de Imágenes.



Crear o eliminar un generador se hace a través de los botones **Crear gen. VC** y **Eliminar gen. VC**, respectivamente.

La parte derecha de la vista se utiliza para la ejecución de los paquetes y generadores definidos. Estas tareas se pueden llevar a cabo con los botones **Transferir gen. VC** o **Transferir paquetes**; o bien arrastrando con el ratón.

Cuando el usuario defina los paquetes y generadores que desea ejecutar, deberá hacer *click* sobre el icono  para que se le muestre una ventana emergente donde podrá elegir si desea hacer uso de alguna ROI entre otras opciones. Estos generadores se ejecutarán en segundo plano, mostrando la progresión en la parte derecha de la barra de estado.

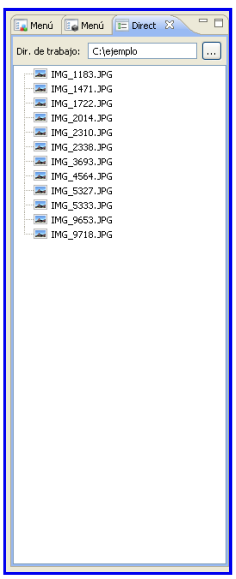
Si el usuario deseara *cargar* de disco un generador VC, este podría hacerlo a través del icono  que se muestra en la parte superior-derecha de la vista.

Si el usuario deseara aplicar uno o varios generadores VC sobre una o varias imágenes en vez de sobre algún paquete, la apariencia de la vista cambia. La parte de gestión de paquetes de imágenes desaparece y en la parte derecha aparece un cuadro de texto que recibe imágenes en vez de paquetes de imágenes. Estas imágenes se deberán insertar arrastrando desde las distintas vistas que utilizan imágenes o a través de la *vista paquetes de imágenes*.

Esta vista no sólo ofrece la funcionalidad a través de los botones, sino que también se puede acceder a estas acciones a través del *botón derecho* del ratón.

Si desea saber más, dirijase a la sección Cómo crear y ejecutar un vector de caracterización.

Directorio de Trabajo

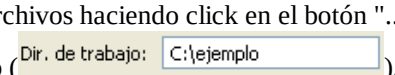


El usuario en esta vista puede elegir un directorio de su *sistema de archivos* como directorio de trabajo de forma que tenga accesible de una manera cómoda los archivos y carpetas que contiene para trabajar con la aplicación. Esta vista diferencia entre los archivos que pueden ser tratados por la aplicación ( , formatos de imagen soportados por ésta) y los archivos que **no** pueden ser tratados por la aplicación ( , cualquier otro archivo).

Los archivos de imagen soportados se pueden abrir directamente haciendo *doble click* sobre las entradas que se encuentran en la vista o arrastrando los iconos a la región del editor. Si la aplicación puede abrirlos, los abrirá en el mismo editor.

Si dentro del directorio de trabajo existen otros directorios () estos se podrán explorar haciendo *click* en el símbolo que aparece a la izquierda.

Si el usuario desea cambiar el directorio de trabajo este puede realizarlo de dos maneras:

- Explorando su sistema de archivos haciendo click en el botón "..." ()
- A través del campo de texto (

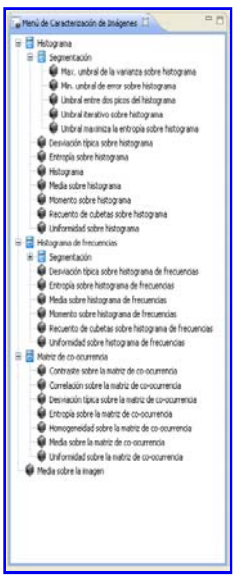
), donde el usuario puede escribir el directorio que quiere establecer como directorio de trabajo. Para confirmar el directorio introducido, el usuario debe pulsar ***Enter***."

Información de Imagen

La vista de información de imagen muestra información que puede serle útil al usuario relacionada con la imagen activa en el editor.

Se muestra información como el sistema de color, ancho y alto de la imagen, número de bandas de color... entre otros datos de interés.

Menú de Caracterización de Imágenes

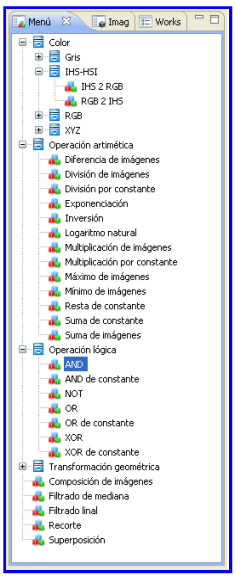


Este menú contiene las operaciones que se encargan de la caracterización digital de imágenes.Este menú variará dependiendo de los *plugins* de operaciones que tengamos instalados.

Para poder acceder a las operaciones ( ) y operar con ellas se puede hacer *un sólo click* sobre la operación, y se mostrará el panel de operación en la vista **Panel de Operaciones de Caracterización**; o hacer *doble click* sobre la operación y se mostrará una ventana emergente con el Panel de Operaciones de Caracterización.

Para poder desplegar las categorías ( ) y mostrar más operaciones se tiene que hacer *click* en el símbolo que se muestra a la izquierda del icono de la categoría y se mostrarán todas las operaciones y/o categorías que contenga la misma.

Menú de Tratamiento de Imágenes



Este menú contiene las operaciones que se encargan del tratamiento digital de imágenes. Este menú variará dependiendo de los *plugins* de operaciones que tengamos instalados.

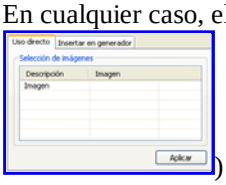
Para poder acceder a las operaciones ( ) y operar con ellas se puede hacer *un sólo click* sobre la operación, y se mostrará el panel de operación en la vista **Panel de Operaciones de Tratamiento**; o hacer *doble click* sobre la operación y se mostrará una ventana emergente con el Panel de Operaciones de Tratamiento.

Para poder desplegar las categorías ( ) y mostrar más operaciones se tiene que hacer *click* en el símbolo que se muestra a la izquierda del icono de la categoría. A continuación se mostrarán todas las operaciones y/o categorías que contenga la misma.

Panel de Operaciones de Caracterización

Esta vista es una de las encargadas de mostrar el panel de los parámetros requeridos por una operación de caracterización de imágenes de las operaciones que se encuentran en el Menú de Caracterización de Imágenes.

El usuario en esta vista puede introducir los parámetros requeridos por una operación para poder aplicarla y producir un informe de caracterización.

En cualquier caso, el usuario siempre puede elegir sobre que imágenes desea aplicar la operación (

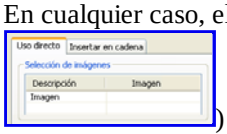
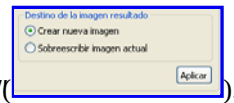
) a través de la pestaña *"Uso Directo"* o añadirlas a un generador a través de la pestaña *"Insertar en generador"*. Cuando se desee aplicar la operación debe pulsar el botón "Aplicar" (

).

Panel de Operaciones de Tratamiento

Esta vista es una de las encargadas de mostrar el panel de los parámetros requeridos por una operación de tratamiento de imágenes de las operaciones que se encuentran en el Menú de Tratamiento de Imágenes.

El usuario en esta vista puede introducir los parámetros requeridos por una operación (en caso que la operación los requiera) para poder aplicarla y producir una imagen resultado.

En cualquier caso, el usuario siempre puede elegir sobre que imágenes desea aplicar la operación () a través de la pestaña *"Uso Directo"* o añadirlas a una cadena de operaciones a través de la pestaña *"Insertar en cadena"*. En el menú inferior el usuario puede elegir si desea *"crear una nueva imagen"* o *"sobreescribir la imagen actual"* (). Para finalmente pulsar el botón "Aplicar" () y ejecutar la operación.

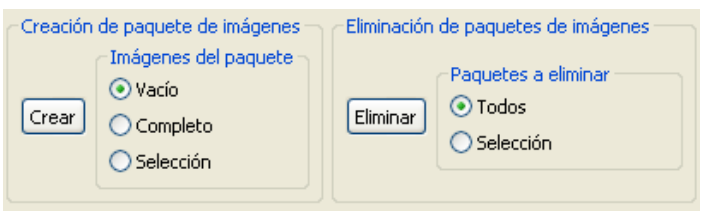
Paquete de Imágenes

Esta vista permite al usuario crear y editar *paquetes de imágenes* que se podrán utilizar con posterioridad en la ejecución de cadenas de operaciones y caracterización de imágenes.

Para definir paquetes de imágenes, primero necesitaremos poder elegir las imágenes que formarán parte del paquete. Se pueden cargar imágenes en la parte superior de la vista a través de los botones  . De izquierda a derecha son las acciones: cargar las imágenes del directorio de trabajo, cargar las imágenes que se encuentran abiertas en el editor o cargar una nueva imagen del sistema de archivos. (Esta última acción es similar a la acción abrir imagen, con la diferencia que no se cargan en el editor).

Las imágenes cargadas se podrán ver a través de los identificadores () o en miniaturas (). Estos botones están situados a la izquierda de los botones de *carga de imágenes*.

Con las imágenes cargadas en la parte superior de la vista, se estará en disposición de crear un paquete de imágenes.
Puede crear o eliminar un paquete de imágenes a través del siguiente menú:



Se le da la opción de crear un paquete vacío *(Vacío)*, con todas las imágenes que se encuentren cargadas en la parte superior de la vista *(Completo)* o aquellas que estén seleccionadas *(Selección)*. El menú de eliminación es similar al explicado de creación.

Los paquetes creados aparecerán en la parte inferior de la vista. Mostrando los paquetes en la parte izquierda y el contenido de los paquetes en la parte derecha.

Esta vista permite otras acciones que facilitan su utilización. Como la selección () o deselección () de todas las imágenes de la parte superior; mostrar sólo la parte superior de la vista () o sólo la parte inferior (); eliminar las imágenes cargadas en la parte superior de la vista () o sólo las seleccionadas ().

Si el usuario deseara *cargar* de disco un paquete de imágenes, este podría hacerlo a través del icono

 que se muestra en la parte superior-derecha de la vista.

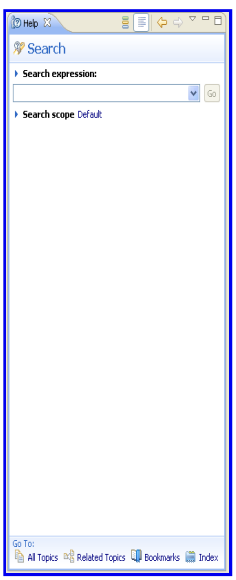
Esta vista no sólo ofrece la funcionalidad a través de los botones, sino que también se puede acceder a estas acciones a través del *botón derecho* del ratón.

Si desea saber más, dirijase a la sección cómo crear un paquete de imágenes.

Progreso de Operaciones

Esta vista muestra qué operaciones se están ejecutando de forma simultánea.

Ayuda



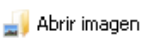
Esta vista provee un buscador de ayuda al usuario.

En ella el usuario puede introducir un término de búsqueda para buscar todas las entradas que coincidan con este término en los manuales de ayuda que se encuentren instalados.

Ésta vista puede ser lanzada desde el menú de ayuda->buscar ayuda.

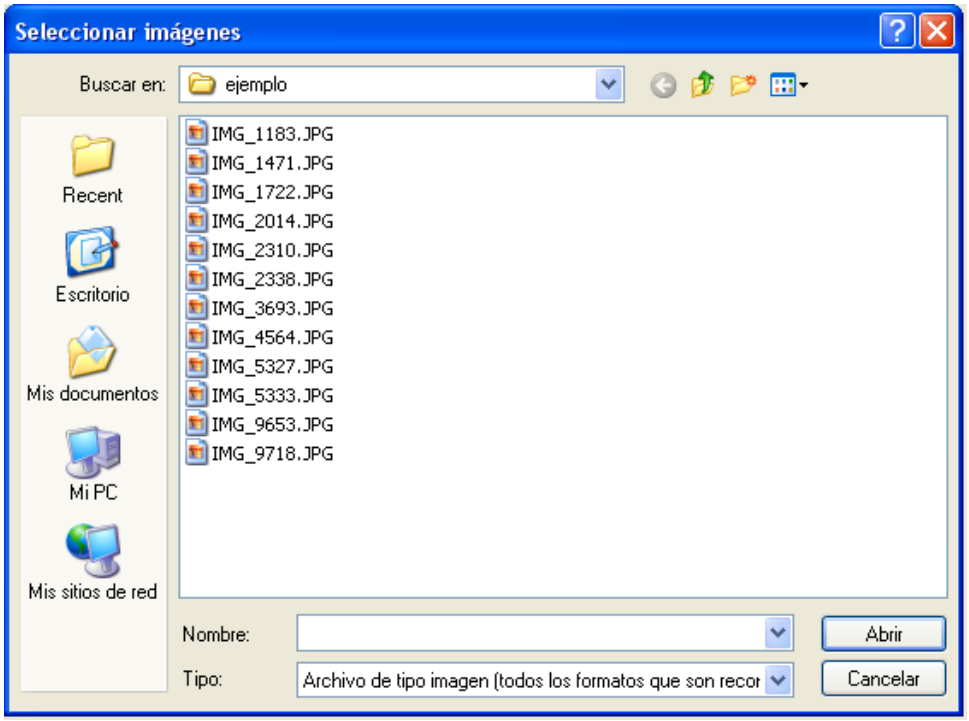
3.3. Acciones

Esta sección esta dedicada a enumerar las acciones principales que pueden llevarse a cabo dentro de la aplicación.

Abrir imagen 

Esta acción se encarga de abrir las imágenes con las que se podrá trabajar en el editor.

Cuando se ejecuta esta acción se muestra un diálogo emergente donde se da la posibilidad de abrir los formatos de imágenes soportados.



Una vez que se elige una o varias imágenes para abrir, todas estas se muestran en el editor para poder comenzar a trabajar con ellas.

Abrir imagen con ROI  Abrir imagen con ROI

Esta acción es similar a la acción *Abrir imagen*, con la diferencia que en esta acción además de abrir la imagen o imágenes seleccionadas por el usuario se intenta abrir la ROI asociada a cada una de las imágenes. En caso que la imagen no tenga ROI asociada, se abrirá la imagen como si se hubiera ejecutado la acción *Abrir imagen*.

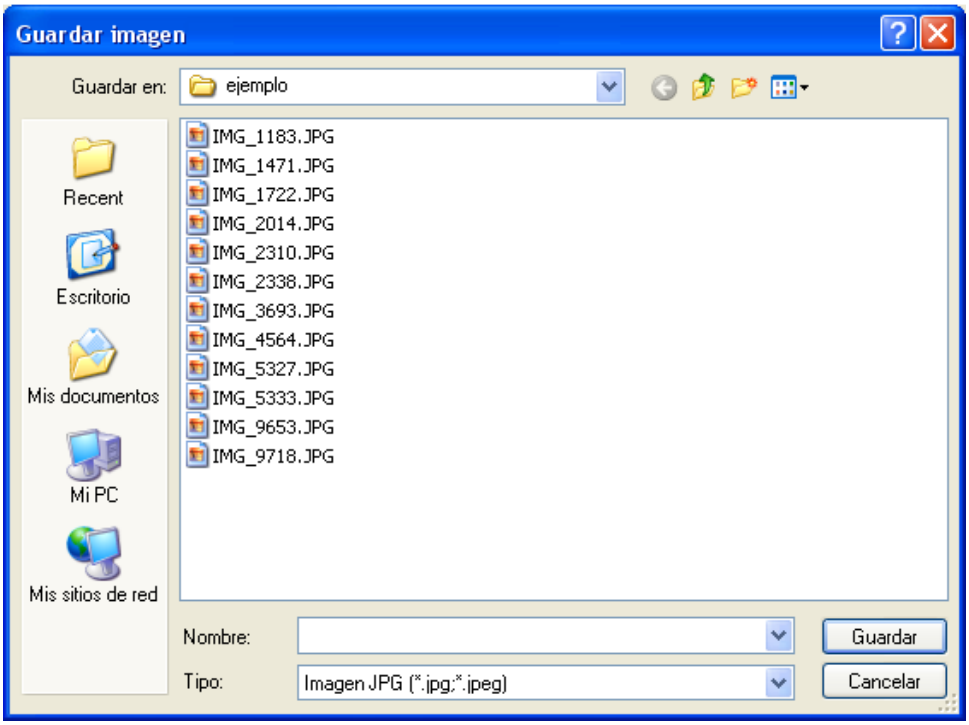
Abrir informe caracterización Abrir informe caracterización

Esta acción se encarga de abrir uno o varios informes de caracterización. Los pasos a seguir para abrir un informe son los mismos que los realizados para *Abrir una imagen*, salvo que ahora no abrimos imágenes, sino informes.

Guardar  Guardar

Esta acción se encarga de guardar la imagen o el informe de caracterización que se encuentre activo en el editor.

Cuando esta acción se ejecuta se le presenta al usuario una ventana emergente similar a la siguiente:



El usuario debe eligir un nombre con el que guardar el archivo, además de poder elegir la extensión con la que guardarlo. Cuando desee guardar la imagen deberá hacer *click* en el botón "**Guardar**". Si no ocurre ningún error, la imagen o el informe quedará guardado en el *sistema de archivos*. En caso que sea una imagen y tenga asociada una ROI esta también será guardada con el mismo nombre que la imagen y con la extensión *".roi"*

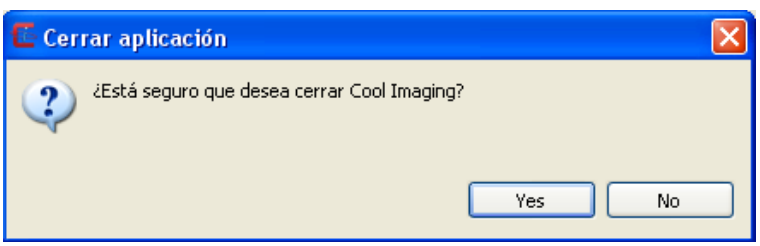
Guardar como Guardar como...

Esta acción permite guardar una imagen o informe de caracterización al igual que la acción Guardar, con la salvedad que esta acción se encuentra activa cuando la imagen o el informe ya se encuentra guardado. De esta forma se le da al usuario la opción de guardar el archivo en otra ubicación y/o con otro nombre.

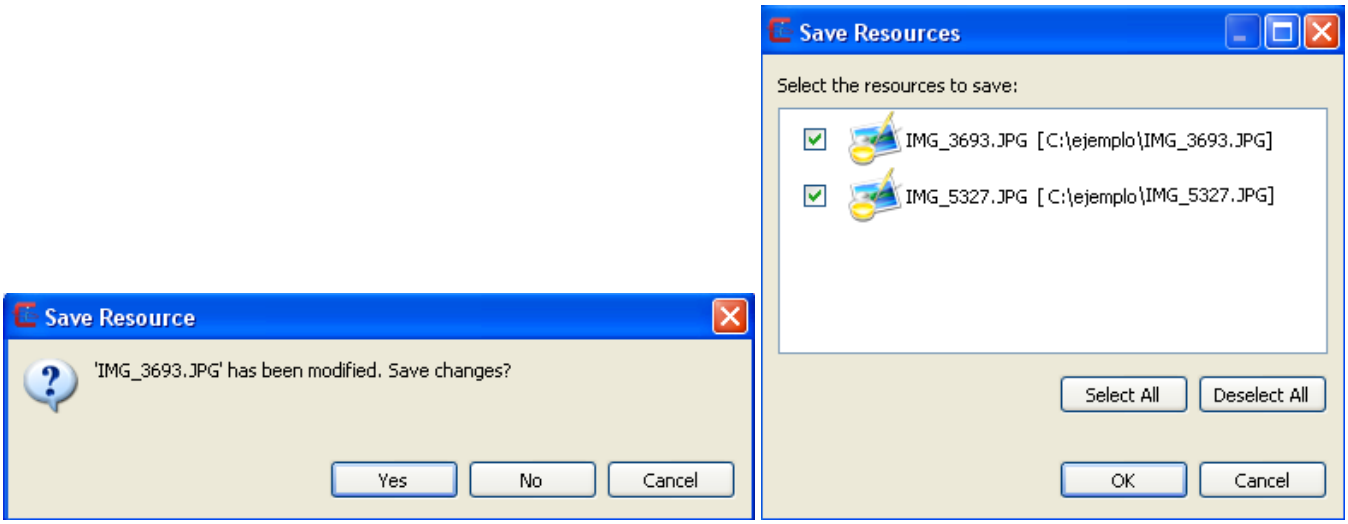
Salir Salir

Esta acción tiene la misión de cerrar la aplicación.

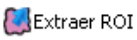
Cuando se ejecuta esta acción se presenta una ventana emergente similar a la que se adjunta.



Si se hace *click* sobre "**Sí (Yes)**" la aplicación se cerrará. En caso que haya elementos sin guardar se le dará al usuario la posibilidad de guardarlos. El diálogo mostrado para guardar antes de cerrar será similar a uno de los siguientes.



En caso de hacer *click* sobre "**No**" la aplicación no se cerrará y se volverá a la situación anterior antes de ejecutar la acción *Salir*.

Extraer ROI 

Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, se encarga de extraer una ROI a una nueva imagen.

La imagen resultado será una imagen rectangular y de igual tamaño a la ROI seleccionada. Los píxeles que formarán esta nueva imagen serán los píxeles internos a la ROI, mientras que los píxeles externos a la ROI se asignarán de color negro.

Un ejemplo de una posible extracción de una ROI se muestra en la imagen que se adjunta a continuación.

Puede encontrar más información de esta acción en cómo definir una región de Interés (ROI).

Definir ROI 

Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, se encarga de definir una ROI sobre una imagen. La imagen se verá modificada al añadir la ROI.

Puede encontrar más información de esta acción en cómo definir una región de Interés (ROI).

Substraer ROI 

Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, provee al usuario la posibilidad de substraer parte de una ROI en una imagen.

El proceso de substracción es análogo al método de definición de una ROI. La imagen se verá modificada al substraer parte de la ROI.

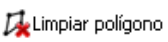
Eliminar ROI 

Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, ons permite eliminar la ROI asociada a una imagen. Si se ejecuta esta acción, se muestra un mensaje de confirmación.

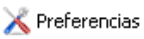
Eliminar punto Polígono 

Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, se encarga de eliminar el último punto añadido al polígono que se está definiendo sobre una imagen. Si se ejecuta varias veces consecutivas esta acción, se irán eliminando los puntos pertenencientes al polígono en orden inverso al que se añadieron.

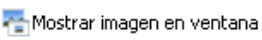
Puede encontrar más información de esta acción en cómo definir una región de Interés (ROI).

Limpiar Polígono 

Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, nos permite eliminar el polígono que se está definiendo sobre la imagen.

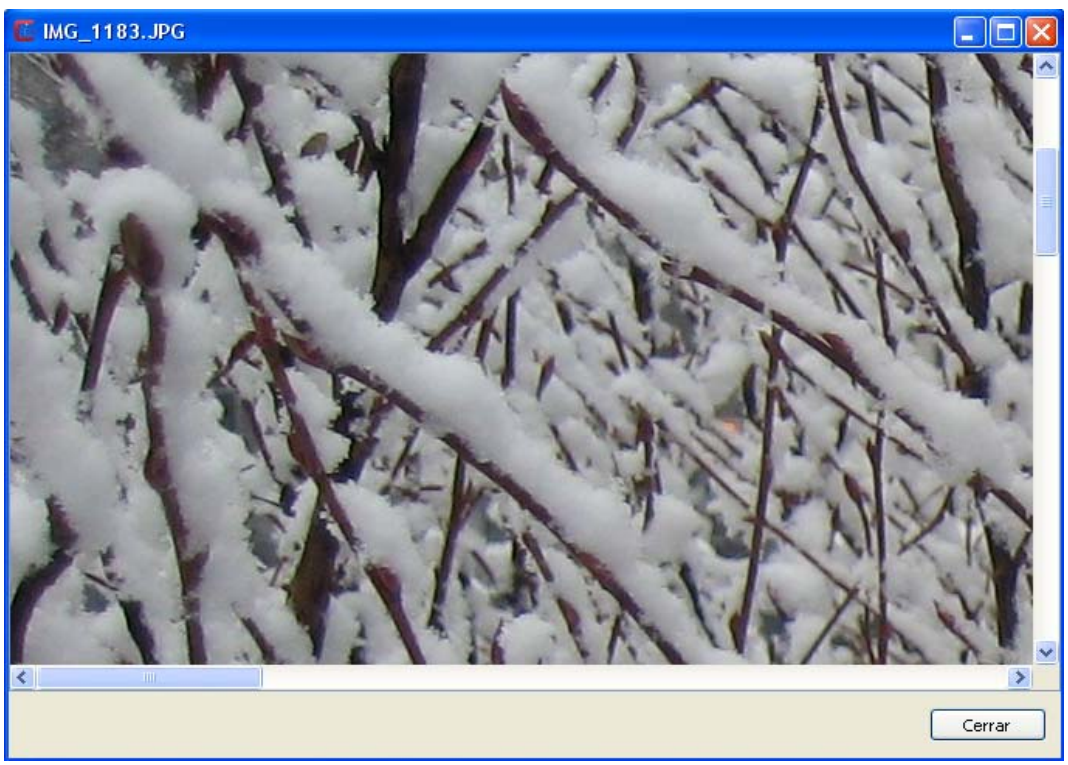
Preferencias 

Esta acción se encarga de abrir la ventana de preferencias de la aplicación. El usuario dentro de esta ventana podrá configurar algunos aspectos de la aplicación.

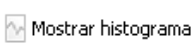
Mostrar imagen en ventana 

Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, se encarga de mostrar la imagen activa del editor en una nueva ventana emergente.

La ventana emergente que contendrá la imagen será similar a la que se muestra en la figura adjunta.

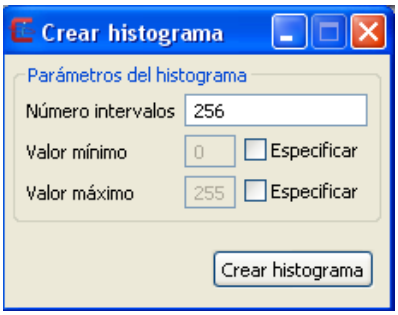


Esta ventana se mantendrá sincronizada con la imagen que se muestra en el editor, esto quiere decir que los cambios que se realicen en la ventana o en el editor se verán reflejados en la otra área de trabajo.

Mostrar histograma 

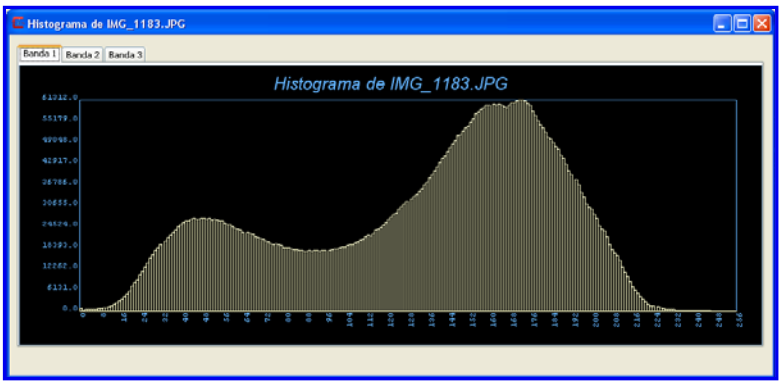
Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, permite al usuario ver el histograma de una imagen o de la ROI asociada a la imagen. Es decir, si la imagen tiene una ROI asociada, solamente se mostrará el histograma asociado a esa región, mientras que si la imagen no tiene asociada ninguna ROI el histograma que se mostrará será de toda la imagen.

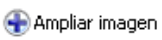
Ejecutada ésta acción, al usuario se le mostrará un diálogo similar al que se ofrece en la imagen inferior.



El usuario en este diálogo podrá elegir el *número de intervalos* que desea que posea el histograma, así como el *valor mínimo* y *valor máximo*. Normalmente los valores de los píxeles se establecen entre 0 y 255, aunque este intervalo puede variar.

La imagen inferior muestra un ejemplo de un histograma calculado en la aplicación sobre una imagen con tres bandas de color.

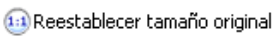


Ampliar imagen 

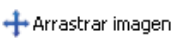
Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, se encarga de ampliar la imagen. Cada vez que se realiza un *click* sobre esta acción la imagen se amplia.

Reducir imagen 

Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, se encarga de reducir la imagen. Cada vez que se realiza un *click* sobre esta acción la imagen se reduce.

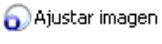
Reestablecer tamaño original 

Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, se encarga de reestabler el tamaño de la imagen a su tamaño original.

Arrastrar imagen 

Esta acción nos permite desplazarnos por la imagen.

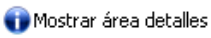
Para poder desplazarnos se debe hacer *click* y mantener pulsado mientras que se desplaza el ratón. La imagen se desplazará con el movimiento del cursor. Cuando se quiera dejar de mover la imagen se debe de dejar de hacer *click*.

Ajustar imagen 

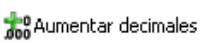
Esta acción, que sólo se encuentra activa cuando tenemos seleccionada una imagen en el editor, permite realizar un zoom sobre la imagen con las dimensiones deseadas por el usuario.

Para llevar a cabo esta acción, el usuario debe hacer *click* sobre una región del editor y mantener pulsado mientras desplaza el ratón. Aparecerá una región rectangular como la que se muestra en la imagen inferior.

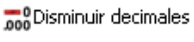
Una vez que ajuste el rectángulo a la región deseada puede dejar de hacer *click* y la acción se llevará a cabo realizando un zoom que se ajustará en la ventana del editor.

Mostrar área de detalles 

Esta acción, que sólo se encuentra activa cuando tenemos seleccionado un informe de caracterización en el editor, nos permite visualizar detalles relacionados con el informe de caracterización.

Aumentar decimales 

Esta acción, que sólo se encuentra activa cuando tenemos seleccionado un informe de caracterización en el editor, nos permite incrementar en una unidad el número de decimales que se muestran en las medidas numéricas que componen el informe de caracterización.

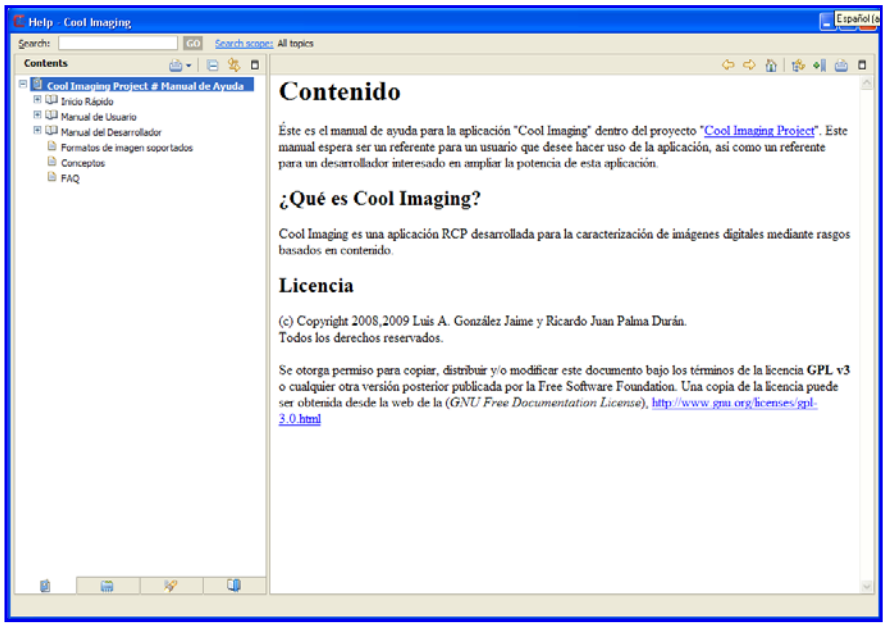
Disminuir decimales 

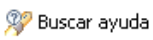
Esta acción, que sólo se encuentra activa cuando tenemos seleccionado un informe de caracterización en el editor, nos permite disminuir en una unidad el número de decimales que se muestran en las medidas numéricas que componen el informe de caracterización.

Mostrar ayuda 

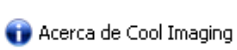
Esta acción se encarga de ejecutar este manual de ayuda.

La ventana que se ofrece es similar a la imagen adjunta.

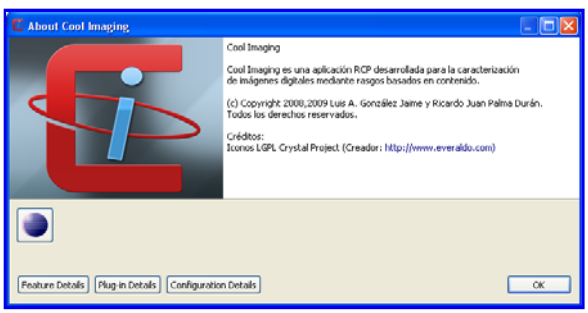


Buscar ayuda 

Esta acción se encarga de ejecutar la vista Buscar Ayuda, donde podrá realizar búsquedas por términos.

Acerca de Cool Imaging 

Esta acción muestra una ventana con información relacionada acerca de la aplicación.



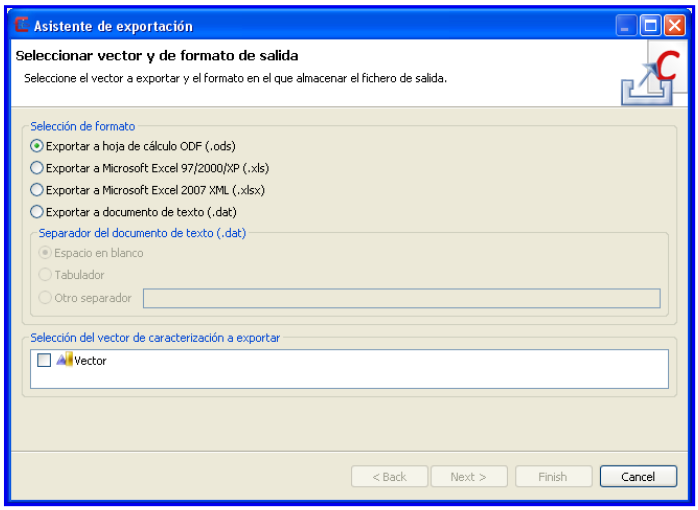
El botón Feature Details muestra información acerca de los *features* instalados en la aplicación.
El botón Plug-in Details muestra información acerca de los *plug-in's* instalados en la aplicación.
El botón Configuration Details muestra información relacionada con los parámetros de configuración de la aplicación.

Exportar informe de caracterización 

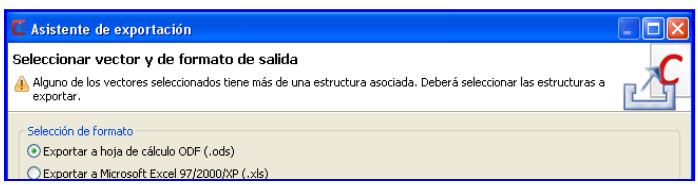
Esta acción, que sólo se encuentra activa cuando tenemos seleccionado un informe de caracterización en el editor, nos permite exportar los vectores de caracterización de los informes a distintos formatos de salida, tales como:

- Hoja de cálculo ODS (.ods)
- Microsoft Excel 97/2000/XP (.xls)
- Microsoft Excel 2007 XML (.xlsx)
- Documento de texto (.dat)

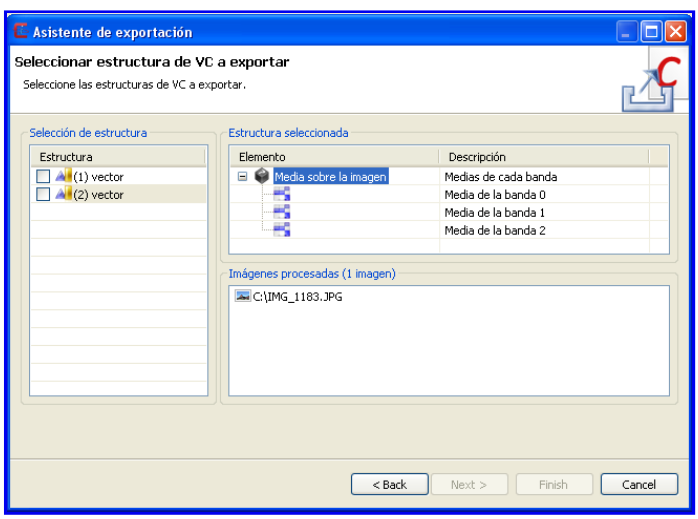
Cuando lanzamos esta acción se nos muestra una ventana similar a la de la imagen inferior, donde debemos elegir el formato de salida, así como el o los vectores de caracterización que queremos exportar. En caso que se elija la opción *Exportar a documento de texto (.dat)* nos permitirá elegir el separador en la sección *Separador del documento de texto (.dat)*.



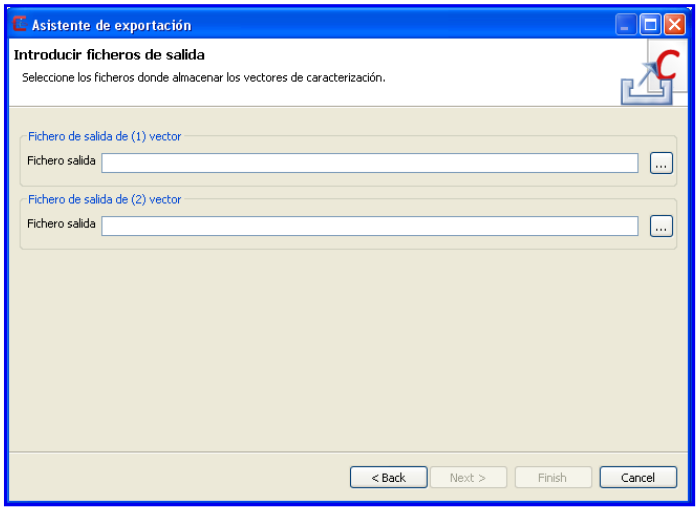
Un mismo vector de caracterización puede tener estructuras distintas para dos o más imágenes, esto podría deberse, por ejemplo, a que se hayan caracterizado imágenes con distinto número de bandas de color, y alguna de las medidas que se generen produzca una medida por banda. Si elegimos caracterizar un vector de estas características, se mostrará un mensaje en la parte superior del diálogo avisándonos de esta situación.



Si se elige exportar un vector con distintas estructuras, se mostrará una nueva página en el diálogo similar a la de la imagen inferior. Donde el usuario debe elegir cuál de estos vectores quiere exportar.



Si todos los vectores que se fueran a exportar tuvieran la misma estructura, la página anterior no se mostraría, y se pasaría directamente a la página siguiente, para elegir el nombre y la ubicación de los ficheros de salida.



El número de entradas de ficheros de salida variará, dependiendo del número vectores a exportar.

Ordenar alfabéticamente

Esta acción, que sólo se encuentra activa cuando tenemos seleccionado un informe de caracterización en el editor, ordena las medidas de caracterización de un informe por orden alfabético. Si se ejecuta dos veces se vuelve a la ordenación inicial del informe.

Mostrar sólo medias y desviaciones típicas

Esta acción, que sólo se encuentra activa cuando tenemos seleccionado un informe de caracterización en el editor, muestra sólo las medidas y desviaciones típicas calculadas sobre los vectores de caracterización del informe. Para que se muestren estas medidas deben haber sido calculadas previamente.

Calcula las medias y desviaciones típicas

Esta acción, que sólo se encuentra activa cuando tenemos seleccionado un informe de caracterización en el editor, calcula las medidas y desviaciones típicas sobre los vectores de caracterización del informe.

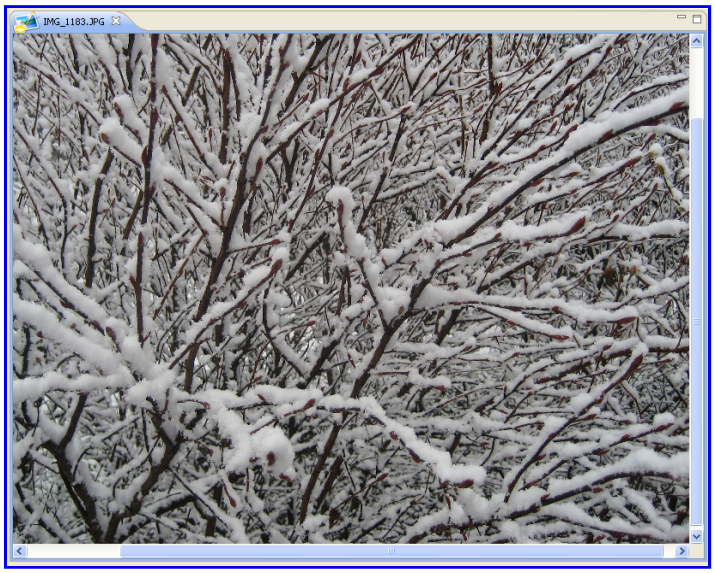
3.4. Editor

El *editor* es el área de trabajo de la aplicación donde el usuario puede trabajar con imágenes o con informes de caracterización.

Editor de imagen

El *editor de imagen* se encarga de dar soporte al usuario para poder trabajar con imágenes. Por ello, cualquier imagen que se abra dentro de la aplicación se abrirá en este área de trabajo.

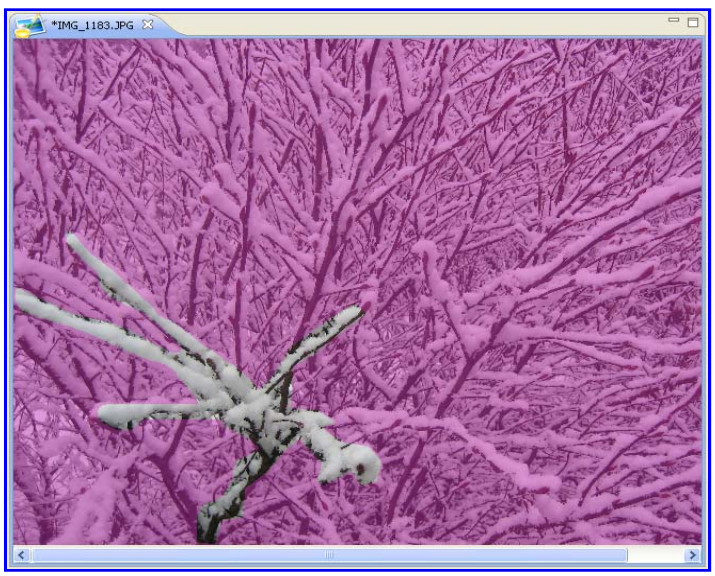
La apariencia de este editor es similar a la de la figura que se adjunta.



El editor puede trabajar con varias imágenes o informes de caracterización simultáneamente, si este fuera nuestro caso, conforme se fuesen abriendo más imágenes o informes se irían organizando en *pestañas* como se muestra en la imagen siguiente.

El editor también permite abrir imágenes simplemente arrastrándolas a este área. Se abrirán si tienen un formato de imagen soportado por la aplicación.

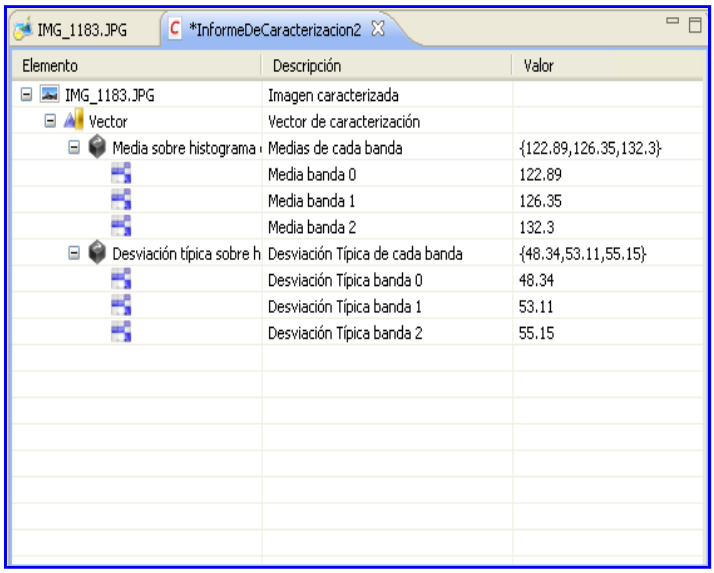
Otras de las tareas que se pueden llevar a cabo sobre el editor es la selección de una región de interés (ROI).



Puede encontrar más información de cómo definir una región de Interés (ROI).

Editor de Informe de Caracterización

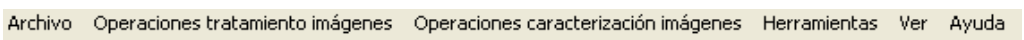
El *editor de informe de caracterización* se encarga de dar soporte al usuario para trabajar con informes de caracterización dentro de la aplicación.



Este editor permite visualizar los resultados de caracterización, así como realizar algunas acciones sobre él. Para más información de las acciones que se puede realizar dirijase a la sección Acciones.

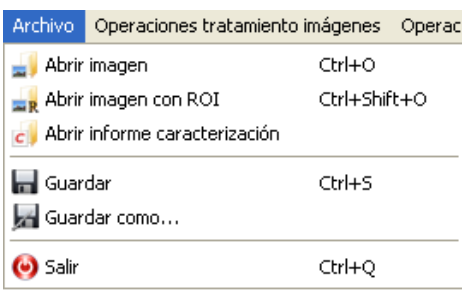
3.5. Barra de Menú

La *barra de menú* organiza todas las acciones que se pueden realizar en esta aplicación por menús que se pueden categorizar por el tipo de acciones que contienen.



Además, esta barra de menú se personaliza dependiendo del elemento activo que se encuentre en el editor, de forma que se muestran las acciones generales junto con aquellas acciones específicas de este tipo de elemento.

Archivo

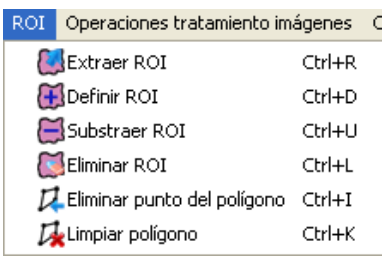


Este menú contiene acciones relacionadas con la gestión de archivos.

- abrir imagen
- abrir imagen con ROI
- abrir informe caracterización
- guardar
- guardar como...
- salir

Algunas de las acciones de este menú se activan o desactivan dependiendo del elemento activo del editor.

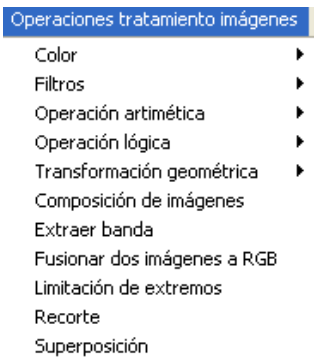
ROI



Este menú, que sólo estará visible cuando estemos trabajando con imágenes dentro del editor, nos permite llevar a cabo acciones relacionadas con la ROI de una imagen.

- extraer ROI
- definir ROI
- substraer ROI
- eliminar ROI
- eliminar punto Polígono
- limpiar Polígono

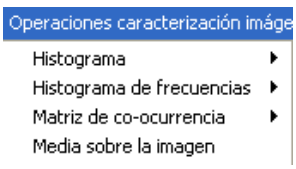
Operaciones tratamiento imágenes



Este menú se encarga de mostrar todas las operaciones de tratamiento de imágenes que se encuentren instaladas en la aplicación.

Estas operaciones se corresponderán con las operaciones que se muestran en la vista Menú de Tratamiento de Imágenes

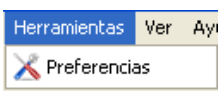
Operaciones caracterización imágenes



Este menú se encarga de mostrar todas las operaciones de caraceterización de imágenes que se encuentren instaladas en la aplicación.

Estas operaciones se corresponderán con las operaciones que se muestran en la vista Menú de Caracterización de Imágenes

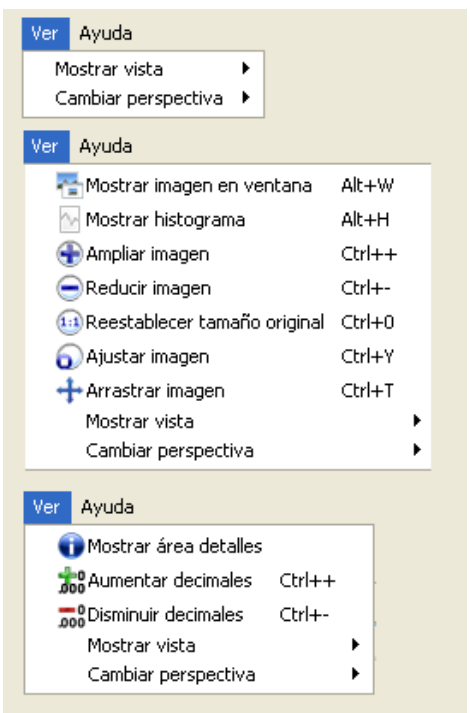
Herramientas



Este menú muestra las herramientas de las que dispone la aplicación para modificar su configuración.

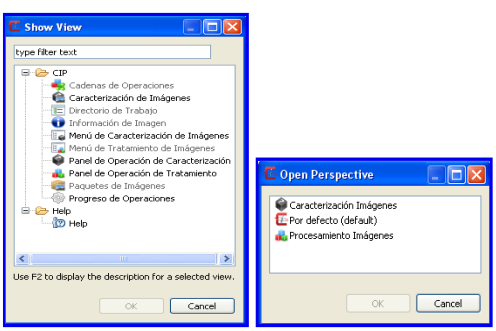
- preferencias.

Ver



Este menú se configura dependiendo del elemento activo en el editor. Si no tenemos ningún elemento activo, la barra de menú nos permitirá añadir nuevas vistas o cambiar la perspectiva con la que se muestra la aplicación.

Las acciones de añadir nuevas *vistas* o cambiar la *perspectiva* se podrán llevar a cabo a través de los siguientes menús.



En caso que sea una imagen lo que tengamos activo, las acciones que se añaden son las relacionadas con el *zoom*, mostrar el histograma o la imagen en una nueva ventana.

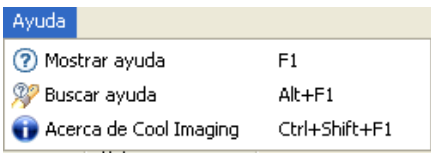
- mostrar imagen en ventana
- mostrar histograma

- ampliar imagen
- reducir imagen
- reestablecer tamaño de la imagen
- ajustar imagen
- arrastrar imagen

Si no es una imagen, y es un informe de caracterización el elemento activo del editor, entonces las acciones que se añaden son las relacionadas con la representación de los decimales o los detalles del informe.

- mostrar área detalles
- aumentar decimales
- disminuir decimales

Ayuda

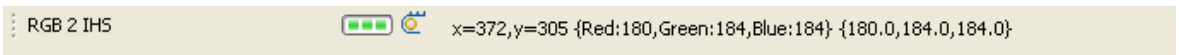


Este menú contiene las acciones relacionadas con mostrar este manual de ayuda e información relacionada con la aplicación (Acerca de *Cool Imaging*).

- Mostrar ayuda
- Buscar ayuda
- Acerca de Cool Imaging

3.6. Barra de Estado

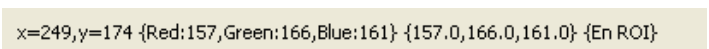
La misión de la *barra de estado* es mostrar información al usuario sobre la imagen activa en el editor o información sobre las operaciones en ejecución.



Barra de Progreso en la Barra de Estado

En la parte izquierda de la barra de estado se muestra información referente a las operaciones actualmente en ejecución. Se puede ver el nombre de la operación y el símbolo  que indica que existen operaciones en ejecución.

Información de imagen en la Barra de Estado



Junto a la barra de progreso, de la barra de estado, también se muestra información de la imagen activa del editor.

Esta barra se puede dividir en varias zonas.

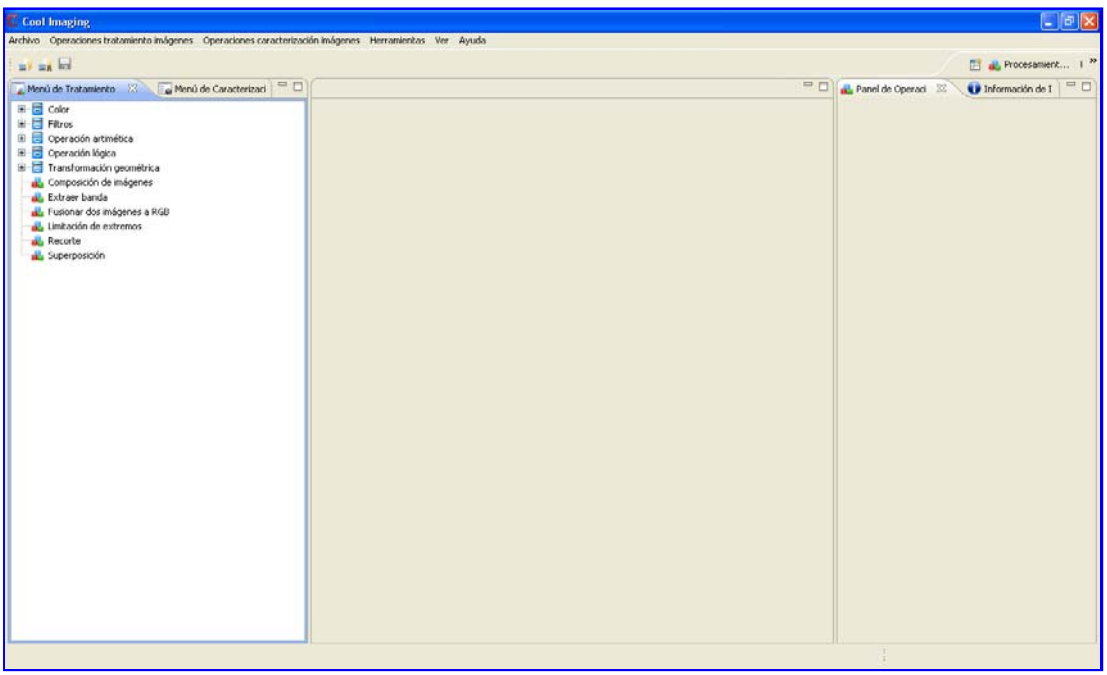
- La parte situada más a la izquierda posee información de la *posición (x, y)* del pixel apuntado por el ratón.
- La zona junto a la *posición del pixel* muestra información del *sistema de color* utilizado por la imagen. Estos valores se obtienen a partir de los pixeles almacenados en disco y se transforman al sistema de color utilizado. Cada uno de los componentes se corresponden con las bandas del sistema de color
- La región junto al *sistema de color* muestra los *valores de los píxeles* que se almacenan en disco. Estos no tienen porqué coincidir con los mostrados en el *sistema de color*, ya que pueden producirse transformaciones intermedias.
- *{En ROI}* se muestra cuando la imagen tiene asociada una región de interés y el píxel señalado pertenece a dicha región. En otro caso, este atributo no se muestra.

3.7. Perspectivas

Esta aplicación ofrece la posibilidad de mostrar varias *perspectivas* diferentes.

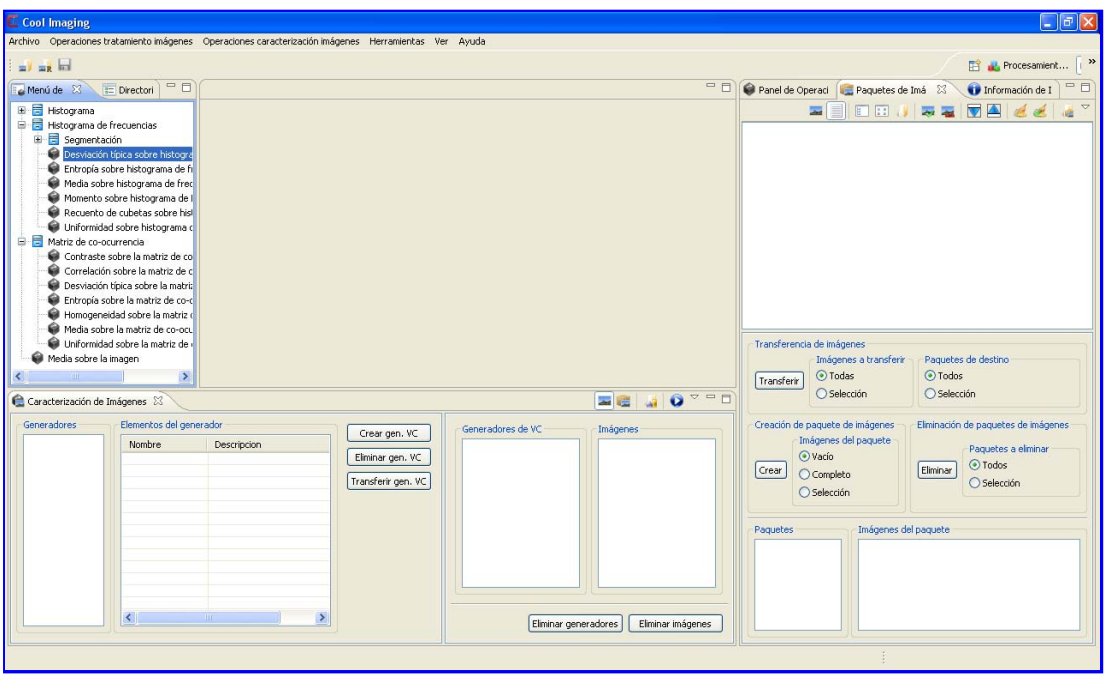
Podemos definir una perspectiva como una distribución de los componentes de la aplicación, como vistas y editores, de forma que cada una de ellas pretenden facilitar alguna de las tareas que puede llevar a cabo el usuario.

Por defecto



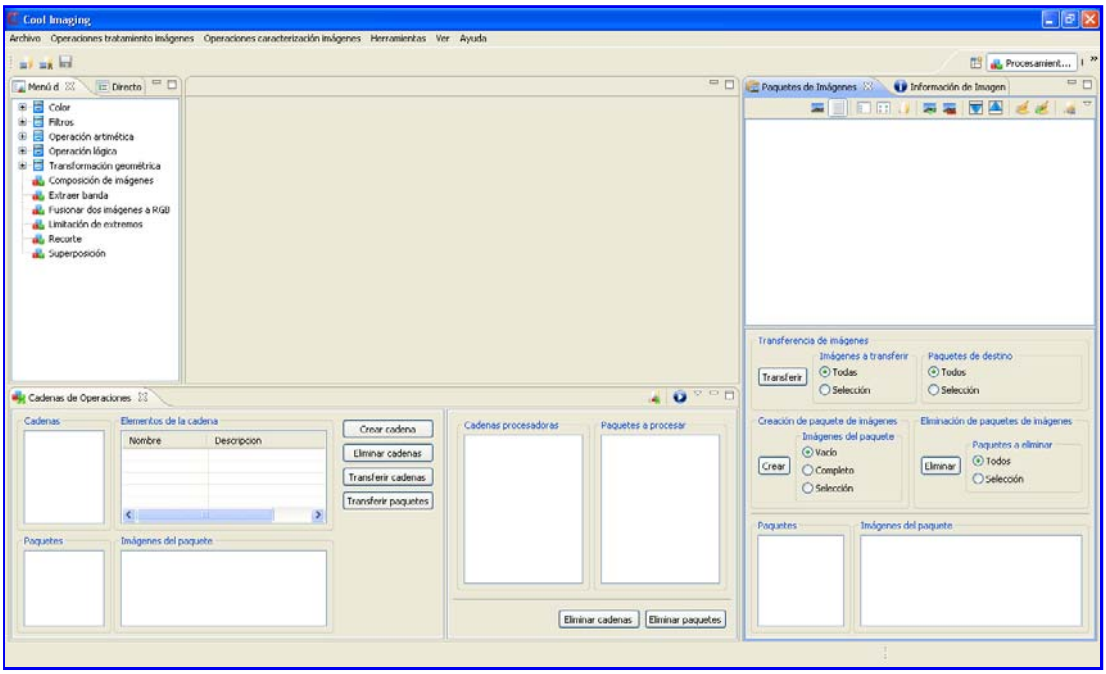
Esta perspectiva es una perspectiva por defecto. Los componentes no se organizan de niguna manera en particular.

Caracterización Imágenes



Esta perspectiva pretende facilitar al usuario las actividades de *caracterización de imágenes*. Las vistas se organizan con este cometido.

Procesamiento Imágenes



Esta perspectiva pretende facilitar al usuario las actividades de *procesamiento de imágenes*. Las vistas se organizan con este cometido.

3.8. Menú Rápido

Este es un menú rápido que ofrece parte de las acciones de la barra de menú, junto con otras adicionales que variarán dependiendo del elemento activo del editor. Esto quiere decir, que cuando se esté trabajando con imágenes se mostrarán acciones diferentes a cuando se esté trabajando con informes de caracterización.

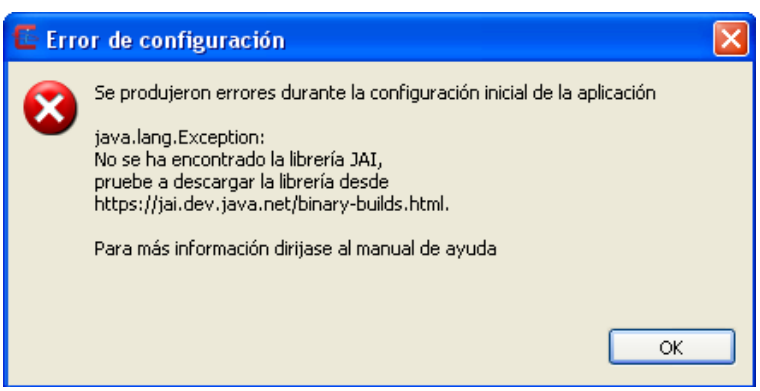
3.9. Errores y Excepciones

java.lang.RuntimeException: No application id has been found

```
java.lang.RuntimeException: No application id has been found
```

Este mensaje de error se muestra en la consola antes de arrancar el sistema de ventanas. Una de las causas que produce este error es tener una máquina virtual de *Java* (*JVM*) anterior a la máquina virtual requerida por la aplicación.

java.lang.Exception: No se ha encontrado la librería JAI.

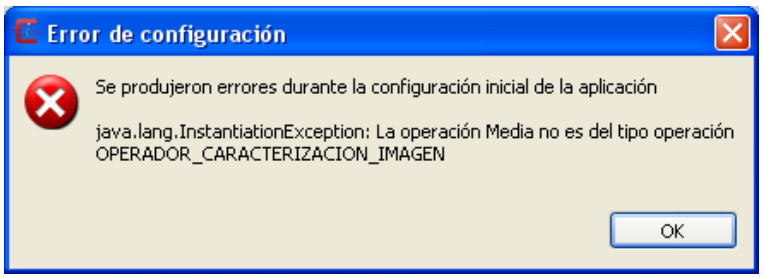


Este mensaje de error se muestra cuando no se ha instalado correctamente o encontrado la librería *Java Advanced Imaging (JAI)*.

Este error puede surgir por varios motivos:

- Por no tener instalada la librería JAI. En este caso, debe instalarla para que la aplicación funcione correctamente. En la siguiente dirección https://jai.dev.java.net/binary-builds.html puede encontrar más información de como instalar JAI. Normalmente la librería debe instalarse para el *Java Runtime Environment (JRE)*.
- Si tiene la librería JAI instalada, porque no se encuentre instalado en la ubicación correcta. Compruebe cual es la *JRE* que se está utilizando y si JAI se encuentra instalado en la ubicación correcta.

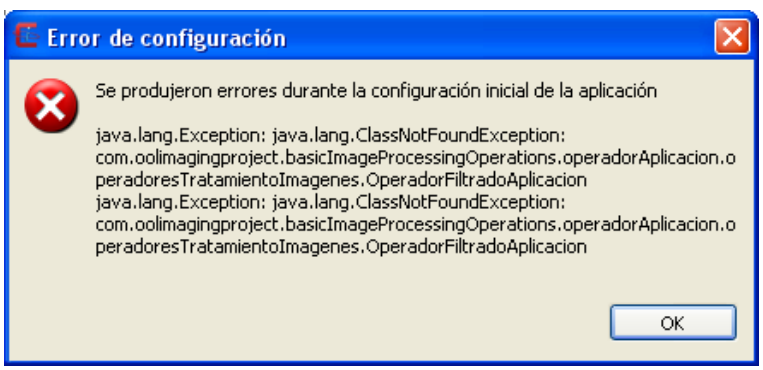
java.lang.InstantiationException: La operación ... no es del tipo operación ...



Este error se muestra cuando las operaciones no se han instanciado en el menú correcto. La aplicación se ejecutará con normalidad pero sin cargar estas operaciones.

Una de las causas es que no se haya instalado el indice del *plug-in* en el lugar correcto. Para solventarlo se debe cambiar de ubicación el índice de las operaciones. Para más información de como instalar un *plug-in* diríjase a la sección Cómo instalar un plugin de operaciones. En caso que persista el error, pongase en contacto con el desarrollador del *plug-in*.

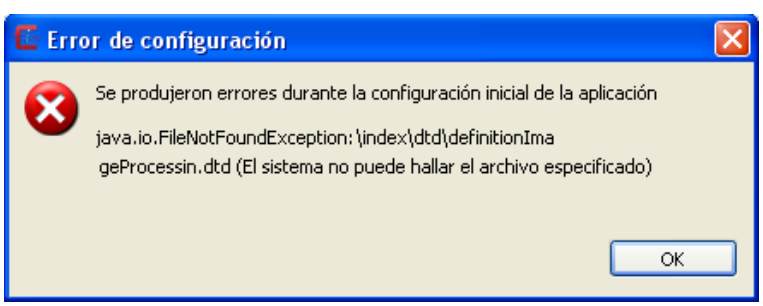
java.lang.Exception: java.lang.ClassNotFoundException: ...



Este error surge cuando la operación que se referencia en el índice de operacionces no se encuentra en alguno de los *plug-in's* instalados. La aplicación se ejecutará con normalidad pero sin cargar estas operaciones.

Para solventarlo debe consultar al desarrollador del *plug-in*.

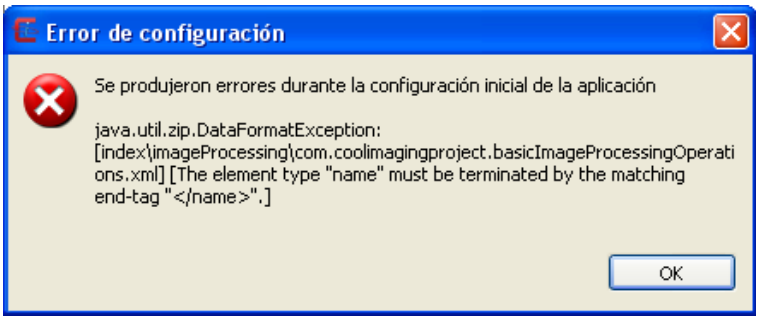
java.io.FileNotFoundException: ...



Este error se produce cuando no se encuentra un archivo. Una de las causas es cuando no se encuentra el archivo *dtd* que debe cumplir el índice. Si usted recibe un mensaje similar a este compruebe uno de los últimos archivos *xml* instalados o compruebe que el archivo existe en la ubicación indicada. Si el error persiste, pongase en contacto con el desarrollador del *plug-in*.

En cualquier caso la aplicación se ejecuta con normalidad, excepto que no carga ninguna de las operaciones que contenga el archivo de índices que produce el error.

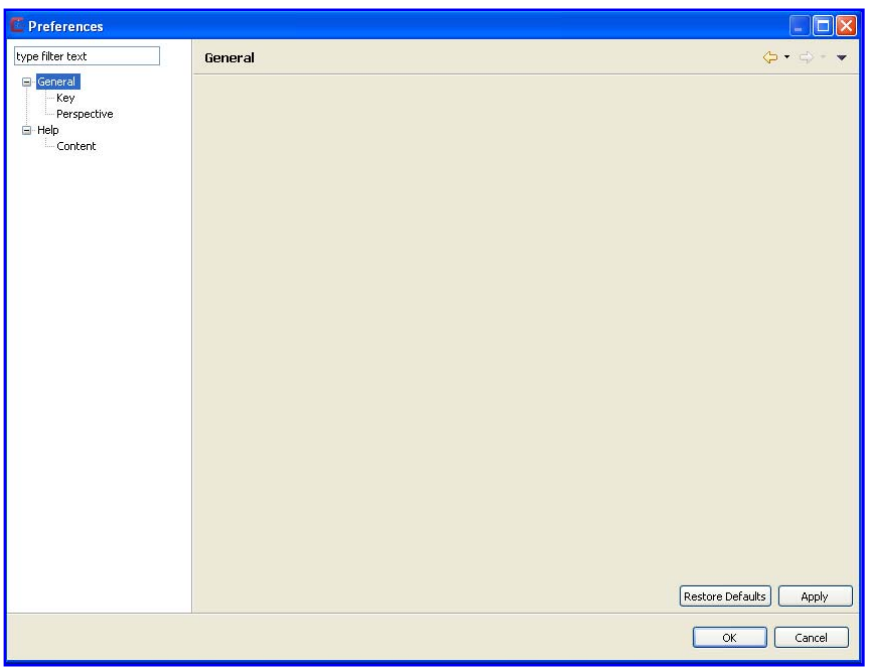
java.util.zip.DataFormatException: ...



Este error se produce cuando alguno de los archivos de índice *xml* no cumple el formato especificado en el *dtd*. Para solventarlo se debe comprobar la corrección del archivo de índice que produce el error. En caso de no saber como realizar dicha tarea pongase en contacto con el desarrollador del *plug-in* de operaciones que produce el error.

En cualquier caso la aplicación se ejecuta con normalidad, excepto que no carga ninguna de las operaciones que contenga el archivo de índices que produce el error.

3.10. Preferencias



La ventana de preferencias nos permite cambiar aspectos de la configuración de la aplicación. Se puede acceder a estos menús a través del menú izquierdo de la ventana, donde cada entrada permite realizar cambios sobre ciertos aspectos de la aplicación.

Atajos de tecla

Esta página permite modificar las combinaciones de teclas de *acceso rápido* para la aplicación. Para más información diríjase a Atajos de tecla.

Perspectivas

Esta página permite configurar y establecer una perspectiva "por defecto". Para más información diríjase a Perspectivas.

Help

Esta página se encarga de realizar cambios que se verán reflejados en la ayuda de la aplicación. Para más información diríjase a Ayuda.

3.10.1. Atajos de Tecla

La función del teclado puede ser personalizada en Cool Imaging usando la página de preferencias **General -> Atajos de tecla**. Con Cool Imaging las secuencias de teclas son asignadas como atajos de teclado para invocar acciones concretas.

Secuencias de teclas y Atajos de teclado

Una 'secuencia de teclas' es la acción de presionar una tecla de teclado, mientras que presionas una o más de las teclas: `Ctrl`, `Alt` (`Option` en Macintosh), `Shift`, o `Command` (sólo en Macintosh.) Por ejemplo, manteniendo pulsada la tecla `Ctrl` y después presionando `A` produces la secuencia de teclas `Ctrl+A`.

Un 'atajo de teclado' es la asignación de una secuencia de teclas a un comando.

Schemes

Un 'scheme' es un conjunto de atajos de tecla. Cool Imaging incluye dos schemes:

- Default
- Cool Imaging Default

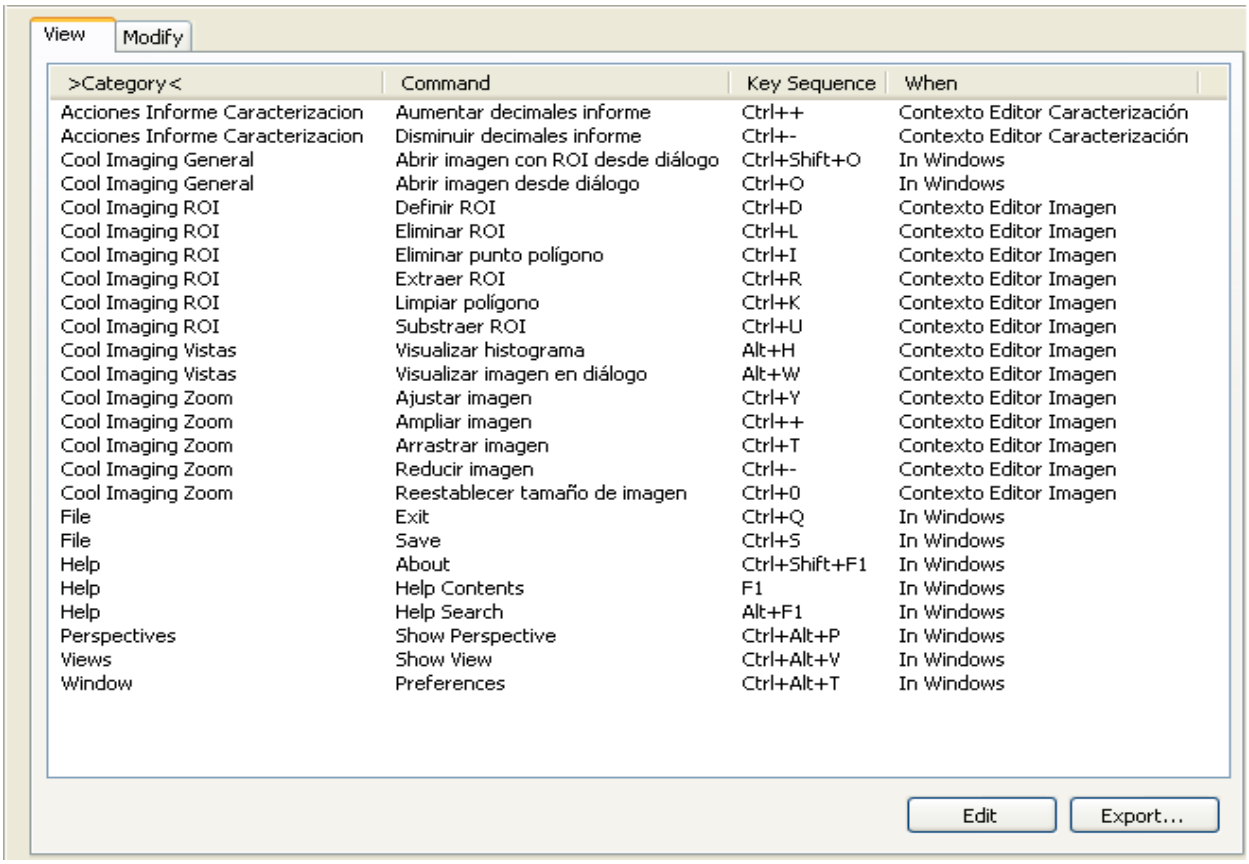
El scheme *Default* consiste en un conjunto general de atajos de teclado. Este es el más popular y reconoce bastantes comandos tradicionales. Por ejemplo, `Ctrl+S` para `Guardar`.

El scheme *Cool Imaging Defaults* consiste en un conjunto de atajos de tecla excluvisamente definido para Cool Imaging.

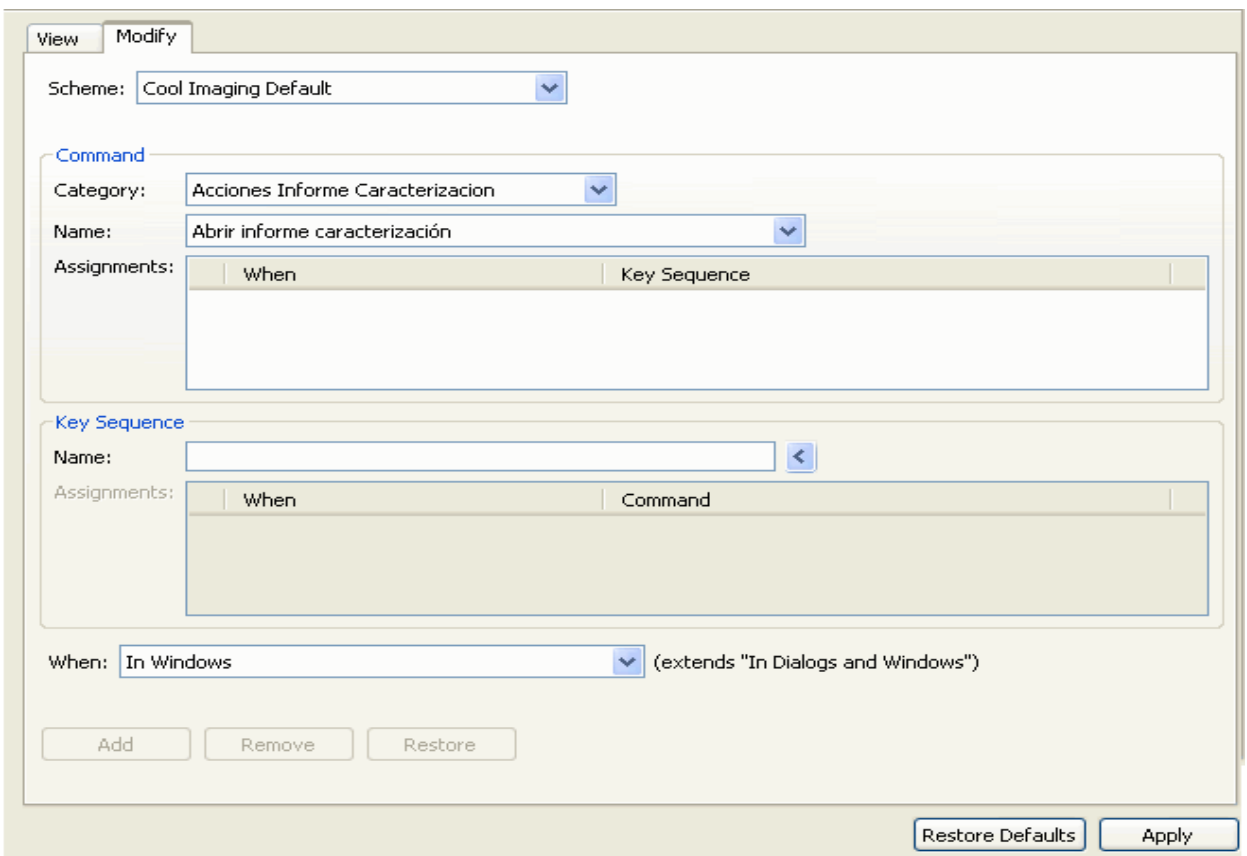
Elija el scheme con el que se encuentre más cómodo trabajando, y cambie el esquema desde 'Scheme' en la página de 'Atajos de tecla' de las preferencias. Si elige el scheme *Default*, todos los atajos de teclado de *Cool Imaging Default* se ignorarán, o si elige el scheme *Cool Imaging Default*, los atajos de teclado de *Default* se ignorarán.

Personalizando los atajos de teclado

Toda la configuración relacionada con los atajos de teclado y schemes se encuentra localizados en la página de preferencias General - > Atajos de tecla.



Esta página nos ofrece dos pestañas, una llamada "View" donde podremos ver los distintos atajos de teclado asignados a los distintos comandos; y una pestaña "Modify", donde podremos modificar los atajos de teclado definidos, además de permitirnos cambiar el scheme que utilicemos.



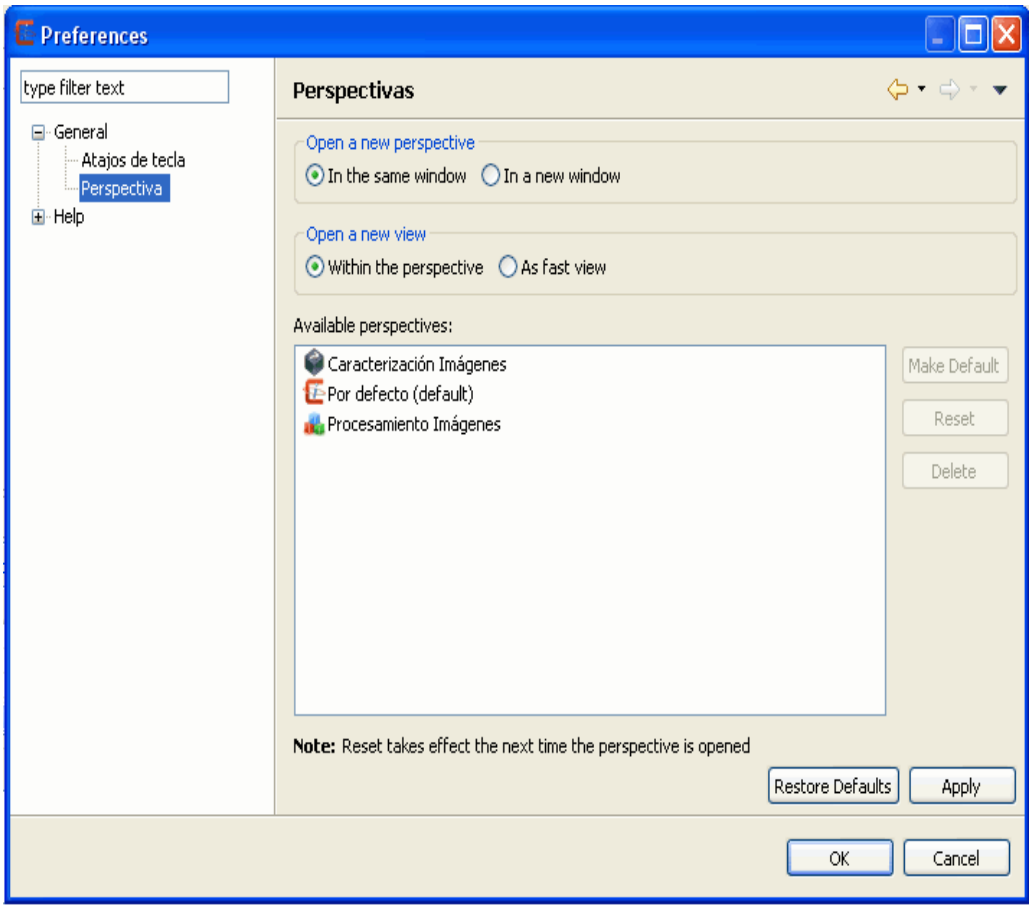
3.10.2. Perspectivas

Opción	Descripción	Por defecto
Open a new perspective	Use esta opción para elegir que pasará cuando abra una nueva perspectiva. ¿Quiere abrir la perspectiva con el Workbench acutal o en una nueva ventana?	In the same window
Open a new view	Use esta opción para especificar que pasará cuando abra una nueva vista. ¿Se abre en la posición por defecto dentro la perspectiva actual o se abre como una vista rápida y se acopla al lado de la perspectiva actual?	Within the perspective

Opciones de perspectivas disponibles:

Opción	Descripción	Por defecto
Make Default	Asigna la perspectiva seleccionada como perspectiva por defecto.	Resource
Reset	Reinicia la configuración de la perspectiva seleccinada a la configuración por defecto. Esta opción es solamente posible si son perspectivas previamente configuradas las cuáles han sido sobreescritas.	n/a
Delete	Elimina la pespectiva seleccinada. Esta opción es solamente posible sobre las perspectivas construidas por el usuario (perspectivas previamente configuradas no pueden ser eliminadas).	n/a

Aquí se muestra como se ve la página de preferencias Perspectivas:

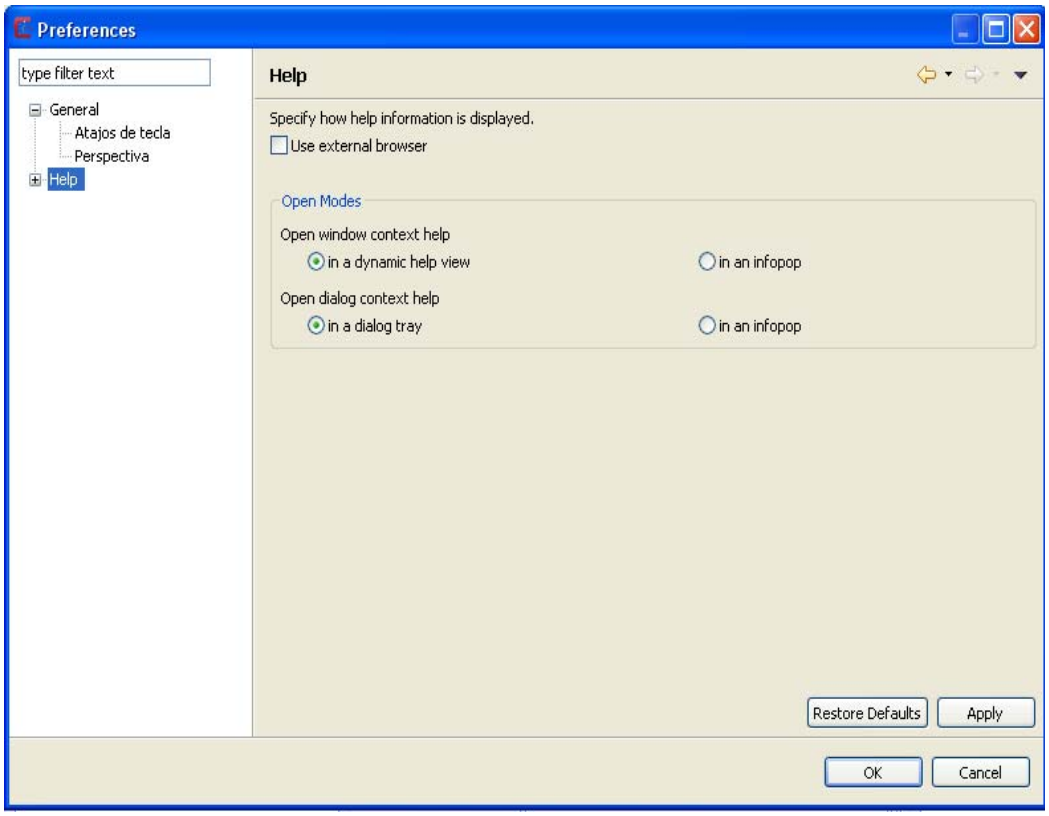


3.10.3. Ayuda

En la página de preferencias de la **Ayuda** puede elegir cómo mostrar la información de ayuda.

Opción	Descripción	Por defecto
Use external browsers	Si su sistema soporta un navegador web embebido, el sistema de ventanas de ayuda usará el navegador embebido para mostrar la ayuda, siempre que fuera posible y esta opción estuviera disponible. Márquela, para mostrar la ayuda en un navegador externo.	Off
Open window context help	Esta opción permite determinar donde quiere abrir la ventana de ayuda, en una ventana de ayuda dinámica o en una burbuja de información.	in a dynamic help view
Open dialog context help	Esta opción permite elegir donde abrir el diálogo de ayuda, en una sección de la ayuda dinámica de la vista Ayuda o en una burbuja de información.	in dialog tray

Aquí se muestra como se ve la página de preferencias de ayuda:



3.11.1. Cómo...

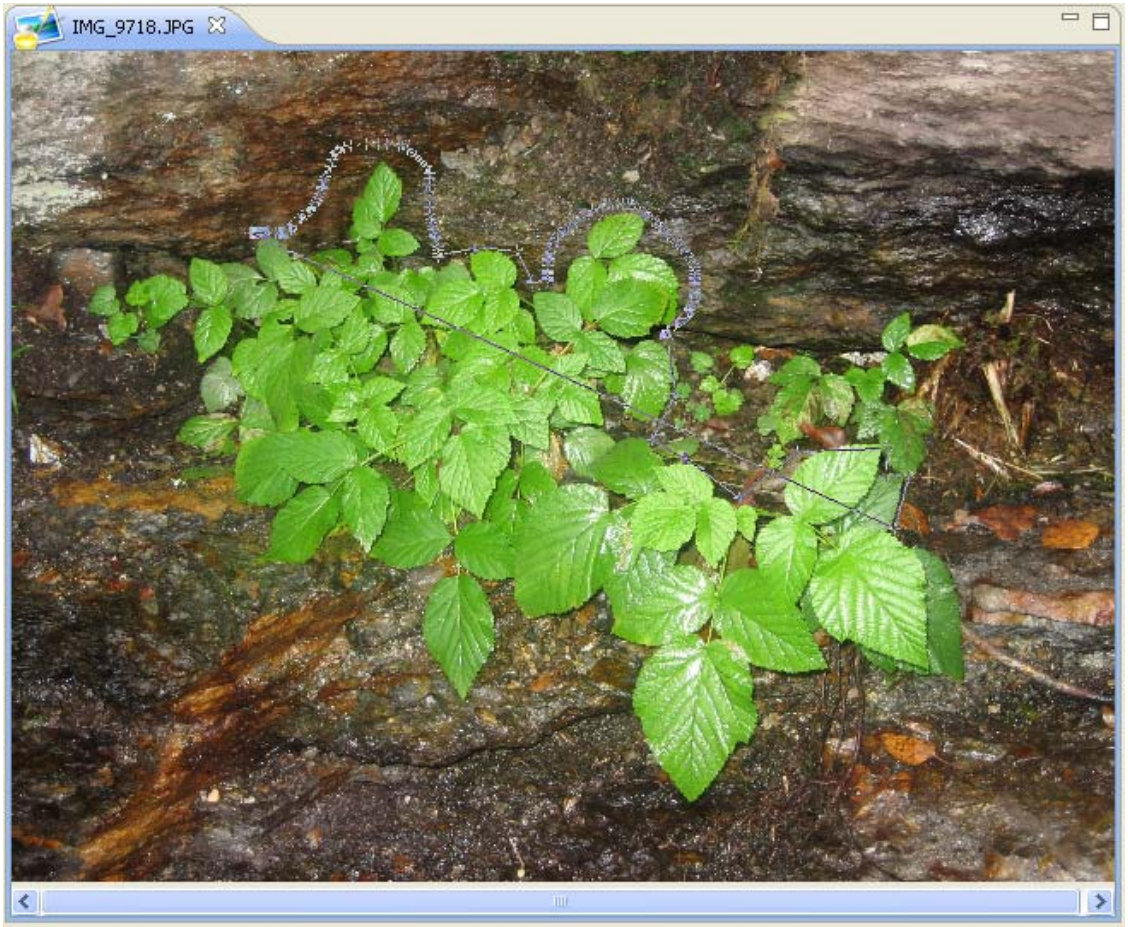
...definir una Región de Interés (ROI)

En esta sección se explica como *definir* un polígono que delimitará una *región de interés* (ROI). Tres son las acciones de la aplicación que permiten el uso de esta acción con intención de extraer una ROI, definir una ROI y/o substraer una ROI.

Cuando una de estas acciones se ecuentra activa, la aplicación permite definir un polígono sobre el editor con los botones *derecho* e *izquierdo* del ratón.

Mientras se quiera definir el polígono, el usuario debe hacer uso del botón derecho del ratón. Haciendo *clicks derechos* o manteniendo pulsado el botón derecho se añadirán nuevos puntos al polígono.

La imagen inferior muestra una captura donde se define el polígono que determinará la región de interés.



Para confirmar el polígono definido y crear o substraer el polígono de la ROI asociada a la imagen, el usuario debe hacer *click* con el botón izquierdo, de esta manera se termina el proceso de definición y se generan los cambios en la ROI asociada a la imagen.

Si en el proceso de definición el usuario quisiera rectificar alguno o todos los puntos añadidos al polígono, lo podrá hacer mediante las acciones eliminar punto Polígono o limpiar Polígono. E incluso si una vez definida la ROI desease eliminarla, lo podrá hacer a través de la acción eliminar ROI.

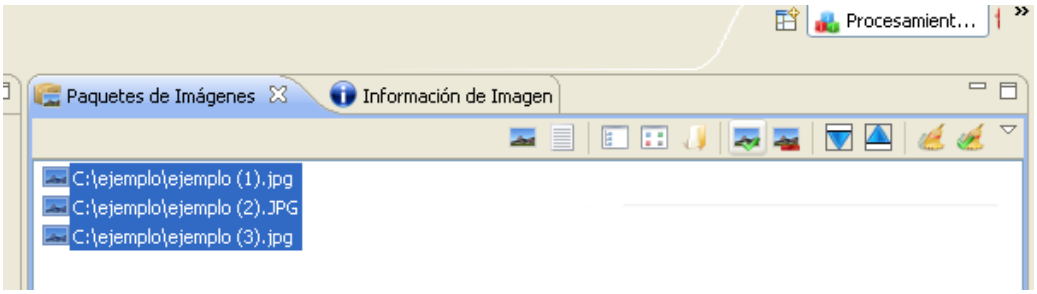
Todas estas acciones están localizadas en el menú ROI.

...crear un paquete de imágenes

La aplicación ofrece la posibilidad de crear un paquete de imágenes a través de la vista Paquetes de Imágenes.

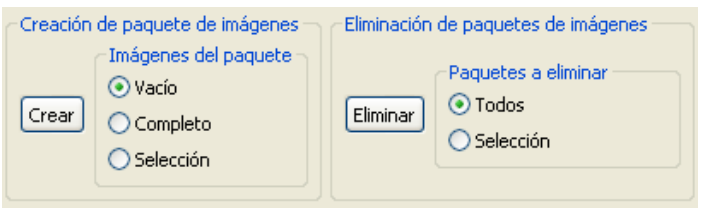
Explicaremos cómo crear un paquete de imágenes a través de un ejemplo. Para ello, lo primero que debemos hacer es seleccionar las imágenes que queremos que formen parte de nuestro paquete dei imágenes.

Nos dirijiremos a la parte superior de la vista *Paquetes de Imágenes* y haremos *click* sobre el icono  para seleccionar las imágenes que queremos abrir. En nuestro caso de ejemplo abriremos tres imágenes que serán las que formarán parte del paquete "paquete ejemplo".



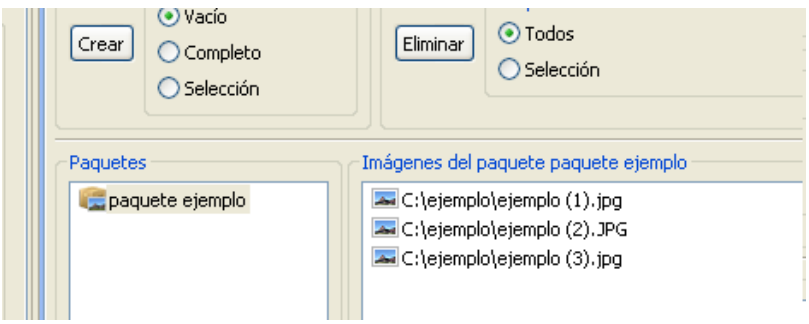
Las imágenes abiertas se irán añadiendo a la parte superior de la vista *Paquetes de Imágenes* en vez de al editor.

El siguiente paso será crear un paquete que contenga las imágenes abiertas. En el menú *Creación de paquete de imágenes* seleccionaremos la opción **Completo**, para que todas las imágenes existentes en la parte superior de la vista pasen a formar parte del nuevo paquete de imágenes. Haremos *click* sobre el botón **Crear** para asignar el nombre al paquete. Nosotros lo llamaremos "paquete ejemplo" como hemos dicho anteriormente.



Cuando se ejecute la acción **Crear** se añadirá una nueva entrada con el nombre del paquete a la parte inferior izquierda de la vista, en la sección *Paquetes*.

También se ofrece la posibilidad de crear un paquete **Vacío** o con las imágenes seleccionadas (**Selección**), al igual que añadir nuevas imágenes con posterioridad a la creación del paquete a través del menú *Transferencia de imágenes*.

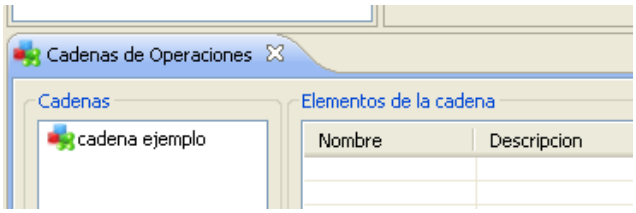


...definir y ejecutar una cadena de operaciones

Una de las posibilidades que ofrece la aplicación es la de definir y ejecutar cadenas de operaciones a paquetes de imágenes a través de la vista Cadenas de Operaciones.

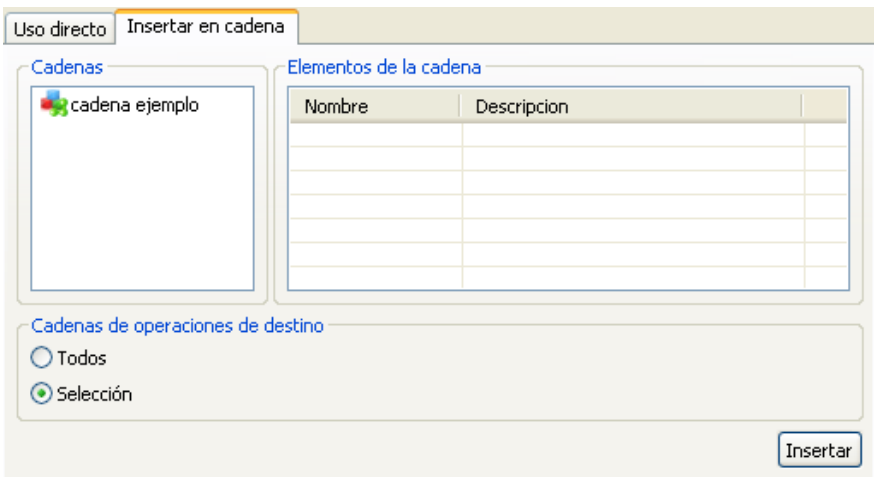
Para poder aplicar una cadena de operaciones, primeramente tendremos que tener definido al menos un paquete de imágenes al que aplicárselas. Los paquetes de imágenes se pueden definir en la vista Paquetes de Imágenes, o si lo desea puede acceder a la sección Cómo crear un paquete imágenes que se explica con un ejemplo cómo crear un paquete de imágenes.

En la vista *Cadenas de Operaciones* crearemos una cadena de operaciones, siempre vacía, a través del botón **Crear cadena**.



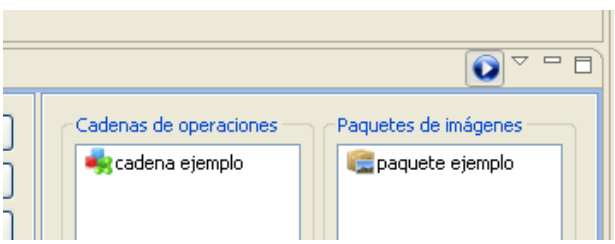
Para añadir nuevas operaciones a una cadena tendremos que hacerlo a través de los *paneles de operaciones*. El procedimiento que se seguirá será el siguiente:

- Abrir el *panel de operaciones* de la operación que queramos añadir a la cadena
- Establecer los parámetros
- Hacer click en la pestaña *"Insertar en cadena"*
- Seleccionar la cadena deseada
- Hacer click en **"Insertar"**

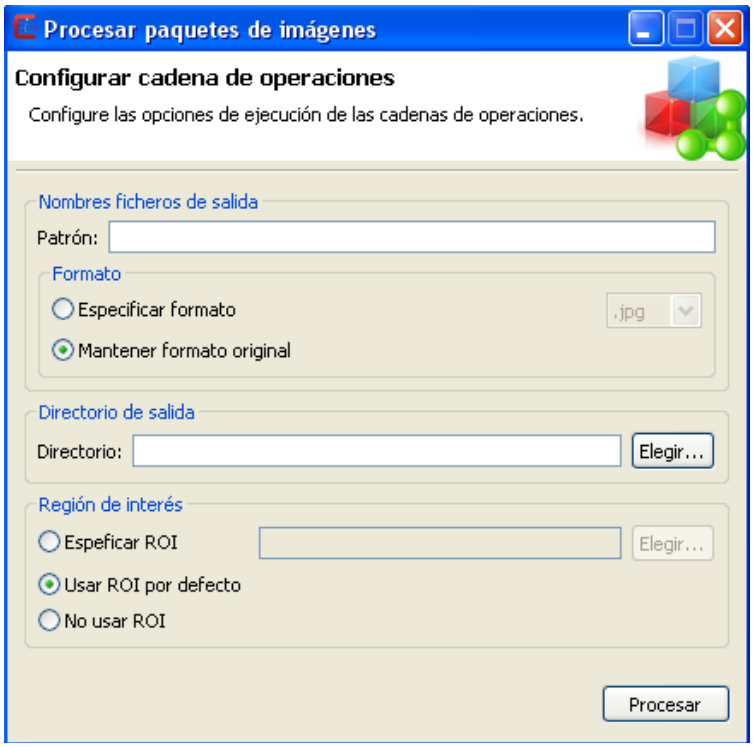


Estos pasos se realizarán tantas veces como operaciones se quieran añadir a las cadenas de operaciones.

Una vez que se tengan los paquetes y las cadenas de operaciones definidas, podremos pasar a ejecutarlas. Procederemos a *transferir* los paquetes de imágenes y las cadenas de operaciones a la parte derecha de la vista *Cadenas de Operaciones* con los botones **Transferir paquetes** y **Transferir cadenas** respectivamente; o bien arrastrando los elementos con el ratón.



Para ejecutar las cadenas de operaciones y los paquetes que se encuentren en la parte derecha de la vista haremos *click* sobre el icono  que se encuentra en la parte superior-derecha.



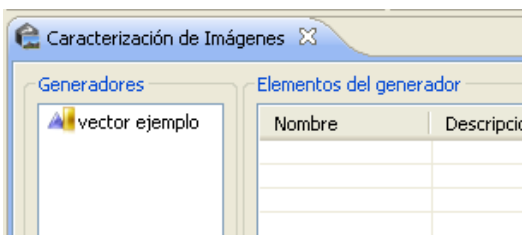
En el diálogo podremos establecer el patrón de nombre de los ficheros de la imagen de salida, en caso de dejarlo en blanco se usará el nombre original; mantener el formato de las imágenes de salida o cambiarlo a un nuevo formato; elegir el directorio de salida; y especificar si se quiere utilizar alguna ROI.

...crear y ejecutar vectores de caracterización

La aplicación ofrece la posibilidad de caracterizar imágenes a través de vectores de caracterización que se definan en la aplicación. Estos vectores recibirán el nombre de generadores de vectores de caracterización (generadores VC), y se gestionarán a través de la vista Caracterización de Imágenes.

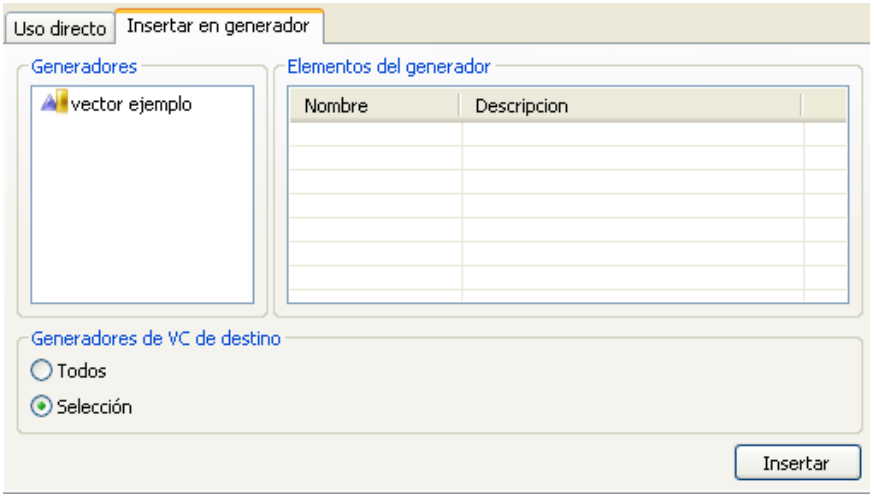
Un generador VC se puede aplicar tanto a imágenes como a paquetes de imágenes. Si lo que queremos es aplicarlo a un paquete de imágenes, entonces tendremos que tener definido al menos un paquete de imágenes. Los paquetes de imágenes se pueden definir en la vista Paquetes de Imágenes, o si lo desea puede acceder a la sección Cómo crear un paquete imágenes que se explica con un ejemplo cómo crear un paquete de imágenes.

En la vista *Caracterización de Imágenes* crearemos un generador VC, siempre vacío, a través del botón **Crear gen. VC**.



Para añadir nuevas medidas de caracterización a un generador VC tendremos que hacerlo a través de los *paneles de operaciones*. El procedimiento que se seguirá será el siguiente:

- Abrir el *panel de operaciones* de la medida que queramos añadir al generador VC
- Establecer los parámetros
- Hacer click en la pestaña *"Insertar en generador"*
- Seleccionar el generador deseado
- Hacer click en **"Insertar"**

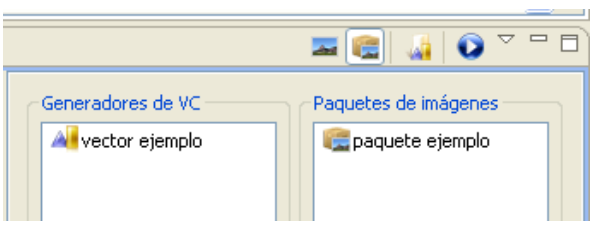


Estos pasos se realizarán tantas veces como medidas se quieran añadir a los generadores VC.

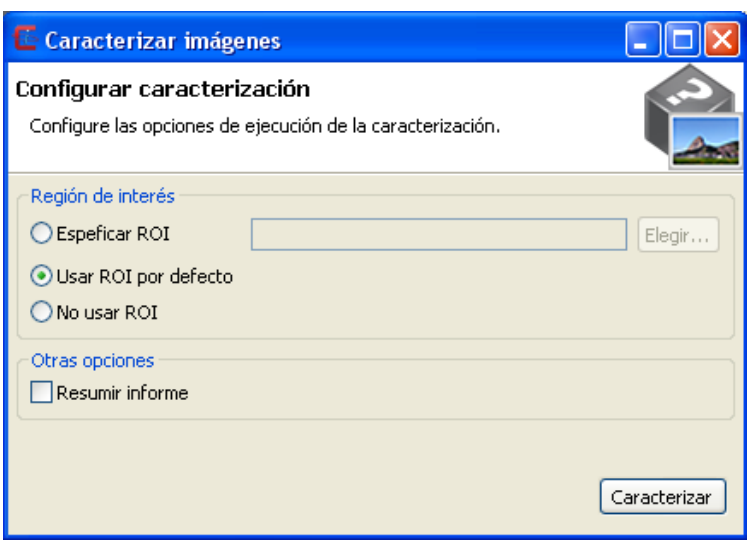
Una vez que se tengan definidos los generadores VC podremos pasar a ejecutarlos. Dependiendo si queremos aplicar estos vectores sobre imágenes o paquetes de imágenes deberemos tener activa la opción correcta. Para trabajar con imágenes tendremos que tener activa la opción  , mientras que si queremos trabajar con paquetes de imágenes la opción activa será  . Ambas acciones se encuentran en la parte superior-derecha de la vista *Caracterización de Imágenes*.

Trabajar con imágenes o con paquetes de imágenes es equivalente, con la diferencia que a la hora de la ejecución uno se aplicará sobre imágenes y el otro sobre paquetes, respectivamente.

Nosotros vamos a trabajar con paquetes de imágenes, por lo que usaremos el que hemos creado previamente en el apartado Cómo crear un paquete de imágenes, llamado "paquete ejemplo". Lo que tendremos que hacer es *transferir* los paquetes de imágenes y los generadores VC a la parte derecha de la vista con los botones **Transferir paquetes** y **Transferir gen. VC**, respectivamente; o bien arrastrando los elementos con el ratón.



Para ejecutar los generadores VC y los paquetes o imágenes que se encuentren en la parte derecha de la vista haremos *click* sobre el icono  que se encuentra en la parte superior-derecha.



En el diálogo podremos establecer si se quiere utilizar alguna ROI o resumir el informe.

Cuando se termine el proceso de caracterización se creará un informe en el editor de caracterización

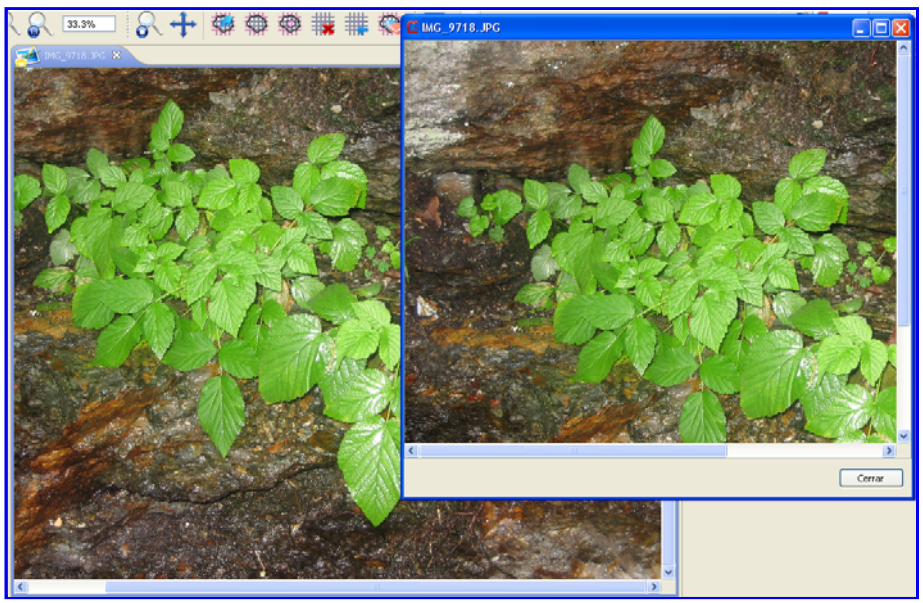
...comparar dos o más imágenes simultáneamente en el editor

Resulta interesante poder comparar dos o más imágenes con las que se está trabajando simultáneamente en el editor.

Esta aplicación permite realizar esta tarea de dos formas distintas:

- La primera de ellas es a través de la acción Mostrar imagen en ventana.

  Cuando se ejecuta esta acción se muestra una ventana emergente similar a la de la imagen de ejemplo que se adjunta.

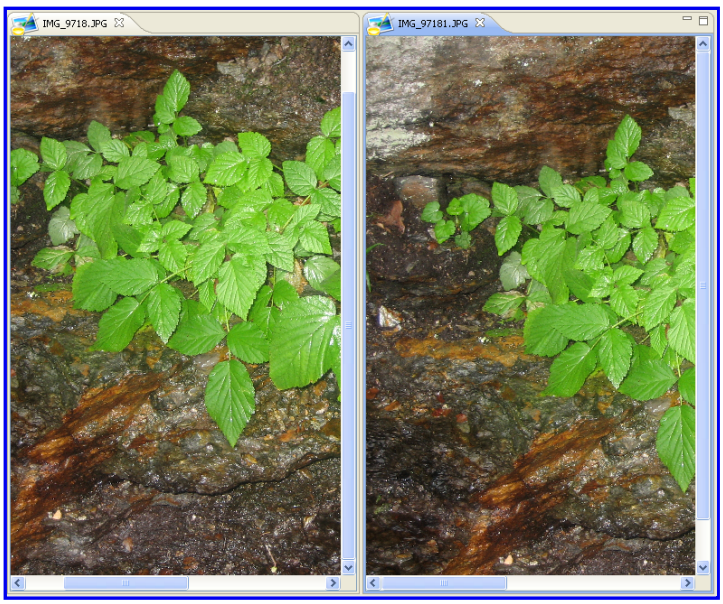


- La otra es a través del editor. Para ello hay que tener las imágenes abiertas en el editor y seguir los siguientes pasos.

  El primer paso consiste en hacer *click* con el botón izquierdo del ratón sobre la pestaña que tenga el nombre de la imagen que queremos comparar. Tal y como se muestra en la imagen.

  Una vez hayamos hecho *click* mantendremos pulsado mientras nos desplazamos hacia la parte derecha del editor. Hasta que aparezca un recuadro similar al que se muestra en la imagen inferior.

Cuando aparezca este recuadro podremos dejar de hacer *click*. Y obtendremos una comparativa de las dos imágenes parecida a la de la imagen.



Igual que se ha añadido la imagen a la parte derecha del editor, se podría haber añadido a la parte inferior desplazando el ratón hacia abajo en vez de hacia la derecha. O incluso, realizar de nuevo esta tarea con una nueva imagen para llegar a comparar varias.

...trabajar con dos o más imágenes simultánemente en el editor

Para poder trabajar con dos o más imágenes simultáneamente en el editor debemos poder *ver* varias imágenes simultáneamente. Esta tarea se realiza de una manera similar al segundo método que se ofrece para comparar dos o más imágenes simultáneamente en el editor, con la diferencia que una vez las tengamos abiertas, podemos empezar a trabajar con ellas igual que si tuvieramos una imagen abierta solamente.

...instalar un plugin de operaciones

Cada *plug-in* debería de explicar como instalarse, pero normalmente estos son los pasos que hay que seguir para instalar un *plug-in* en esta aplicación.

Un *plug-in* puede venir en algún formato comprimido. Si este es el caso de nuestro *plug-in* primeramente tendremos que

descomprimirlo.

En cualquier caso, el *plug-in* debe constar de al menos dos archivos:

- Uno o varios archivos "*.jar*".
- Uno o varios archivos "*.xml*".

Una vez se tengan localizados estos archivos, debemos buscar la carpeta en la que se encuentra instalada la aplicación.

Cuando localicemos la ubicación, debemos copiar los archivos "*.jar*" en la carpeta *plugins* que exista en el directorio. En el caso de los archivos "*.xml*" el proceso es algo más complejo. El desarrollador del *plug-in* debe de especificarnos si las operaciones que se han implementado son de *procesamiento de imágenes* o *caracterización de imágenes*, debido a que si las operaciones que implementa el *plug-in* son de *procesamiento de imágenes* estos archivos *xml* deben de copiarse a la carpeta "*index/imageProcessing*" que existe en el directorio, y en caso de ser operaciones de *caracterización de imágenes* estos archivos deben copiarse en la carpeta "*index/imageCharacterizing*".

Una vez realicemos estos pasos el *plug-in* quedará instalado. En caso que la aplicación estuviese abierta, se deberá reiniciar para que se apliquen los cambios.

Si el *plug-in* no se instala correctamente pueden aparecer errores al iniciar la aplicación. Si este es su caso, diríjase a la sección Mensajes de Error y Excepción de la aplicación, y sino encuentra ninguna solución posible, contacte con el desarrollador del *plug-in*.

3.12. Atajo de Teclas

Esta sección presenta las combinaciones de teclas de acceso rápido "por defecto" que tiene definidas la aplicación. Si el usuario deseára cambiar estas combinaciones podría hacerlo a través de las preferencias.

Las columnas llamadas *Tecla Español* y *Tecla Inglés* son las teclas que se activan al mantener pulsada la tecla *Alt* en los distintos idiomas.

Menú		**Tecla Español**	**Tecla Inglés**	**Atajos de Tecla Default**	**Atajos de Tecla Cool Imaging Default**
Archivo		*A*	*F*		
	Abrir imagen			Ctrl+O	Ctrl+O
	Abrir imagen con ROI			Ctrl+Shift+O	Ctrl+Shift+O
	Guardar			Ctrl+G	Ctrl+S
	Guardar como...				
	Salir			Ctrl+Q	Ctrl+Q
ROI		*R*	*R*		
	Extraer ROI			Ctrl+R	Ctrl+R
	Definir ROI			Ctrl+D	Ctrl+D
	Substraer ROI			Ctrl+U	Ctrl+U
	Eliminar ROI			Ctrl+L	Ctrl+L
	Eliminar punto polígono			Ctrl+I	Ctrl+I
	Limpiar polígono			Ctrl+K	Ctrl+K
Operaciones tratamiento imágenes		*T*	*P*		

Operaciones Caracterización imágenes		*C*	*C*		
Herramientas		*H*	*T*		
	Preferencias			Ctrl+Alt+T	Ctrl+Alt+T
Ver		*V*	*V*		
	Mostrar imagen en ventana			Alt+W	Alt+W
	Visualizar histograma			Alt+H	Alt+H
	Ampliar imagen			Ctrl++	Ctrl++
	Reducir imagen			Ctrl+-	Ctrl+-
	Reestablecer tamaño de imagen			Ctrl+0	Ctrl+0
	Ajustar imagen			Ctrl+Y	Ctrl+Y
	Arrastrar imagen			Ctrl+T	Ctrl+T
	Mostrar área detalles				
	Aumentar decimales			Ctrl++	Ctrl++
	Disminuir decimales			Ctrl+-	Ctrl+-
	Mostrar vista	Other		Ctrl+Alt+V	Ctrl+Alt+V
	Cambiar perspectiva	Other		Ctrl+Alt+P	Ctrl+Alt+P
Ayuda		*Y*	*H*		
	Mostrar ayuda			F1	F1
	Buscar ayuda			Alt+F1	Alt+F1
	Acerca de Cool Imaging			Ctrl+Shift+F1	Ctrl+Shift+F1

4.1. Cómo crear un *plug-in* de Operaciones

Este manual esta enfocado para ayudar a aquellos desarrolladores que deseen crear *plug-in's de operaciones* para esta aplicación.

En los siguientes apartados se explicará a través de un ejemplo todos los pasos necesarios para desarrollar un *plug-in de operaciones* de **cambio de espacios de color** que se podrá integrar en la aplicación con posterioridad. Los pasos seguidos para este ejemplo, serán los mismos que se deberán seguir para desarrollar otros *plug-in's de operaciones*. Mencionar, que parte del procedimiento aquí explicado es compartido con el sistema de extensibilidad utilizado por *Eclipse*.

El desarrollador debe tener en cuenta que la creación de un *plug-in* para la aplicación conlleva dos pasos:

- *la creación del plug-in en sí*
- y *la creación de un índice*.

Cuando estos pasos hayan sido realizados podrá instalar el plug-in en la aplicación.

Para comenzar a desarrollar su *plug-in* diríjase a la sección Antes de comenzar.

4.1.1. Antes de comenzar

Para poder desarrollar un *plug-in* para **Cool Imaging** necesitamos el entorno de desarrollo de *Eclipse*. Por eso, tenemos que disponer del **SDK de Eclipse para desarrolladores de RCP/Plug-in** (*SDK Eclipse for RCP/Plug-in Developers*). Este SDK puede encontrarse en el enlace siguiente http://www.eclipse.org/downloads.

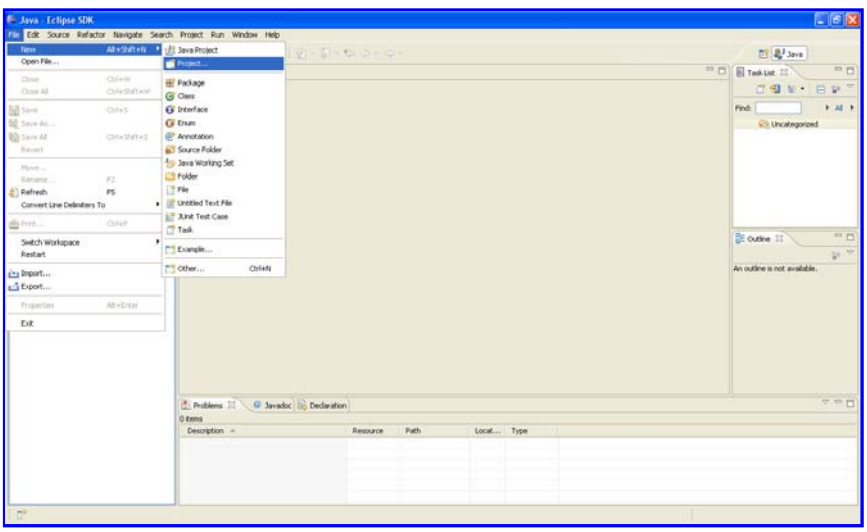
Una vez dispongamos del SDK de Eclipse instalado en el sistema, pasaremos a copiar el archivo *com.coolimagingproject.coolimaging_x.x.x.jar (se recomienda descargar la última versión de este jar)* en la carpeta *plugins* del SDK recién instalado. Este archivo no es más que el código y elementos que forman parte del núcleo de la aplicación **Cool Imaging**. Podemos disponer de este jar en la sección **Descargas** de la página coolimagingproject.com.

Con el *SDK instalado* y el *jar copiado en la carpeta plugins*, ya podemos *crear un proyecto plug-in en Eclipse*.

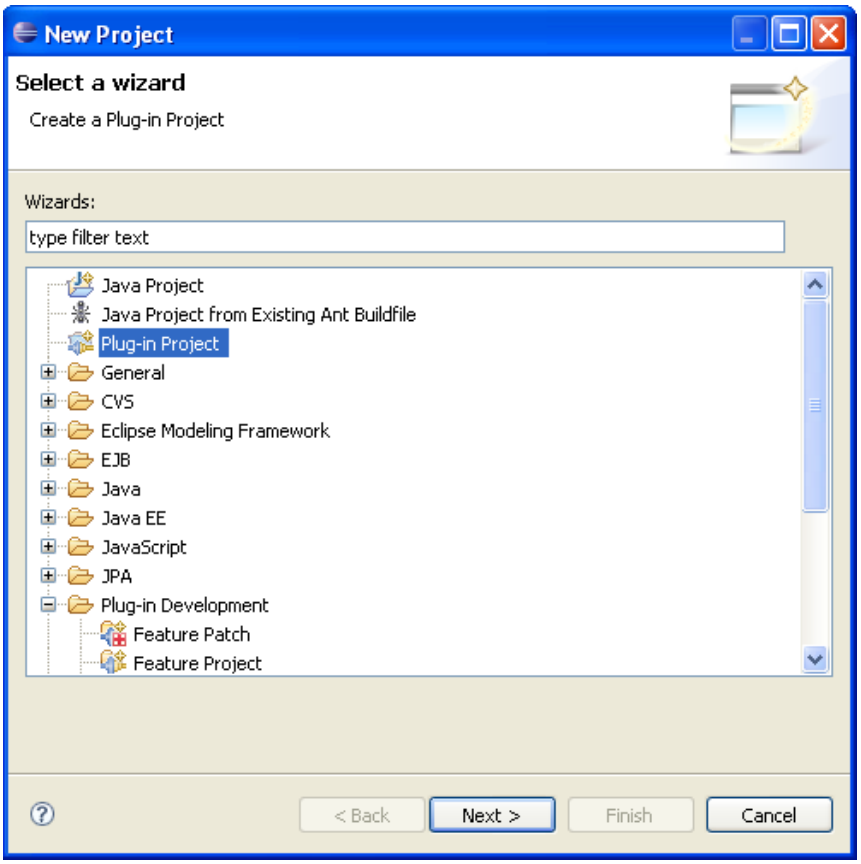
4.1.2. Cómo crear un proyecto Plug-in en Eclipse

En este apartado explicaremos los pasos que hay que seguir para crear un *proyecto plug-in*.

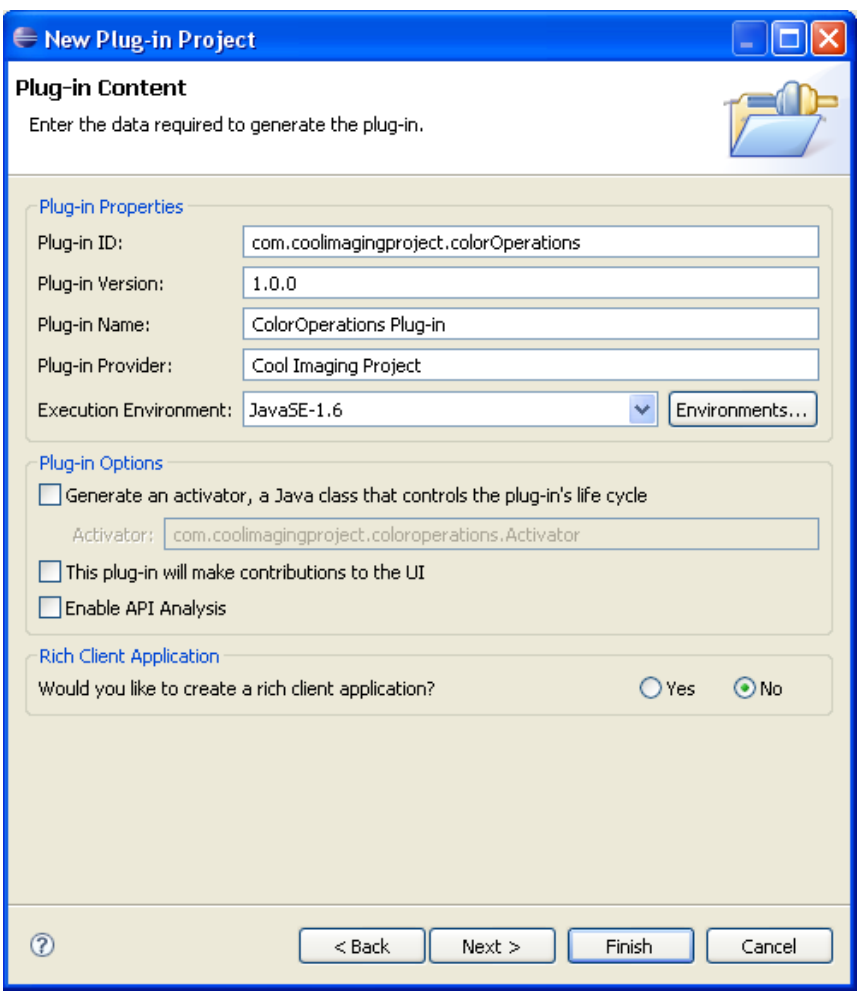
1. Debe dirigirse a *File->New->Project* dentro de Eclipse como se muestra en la imagen que se ofrece a continuación.



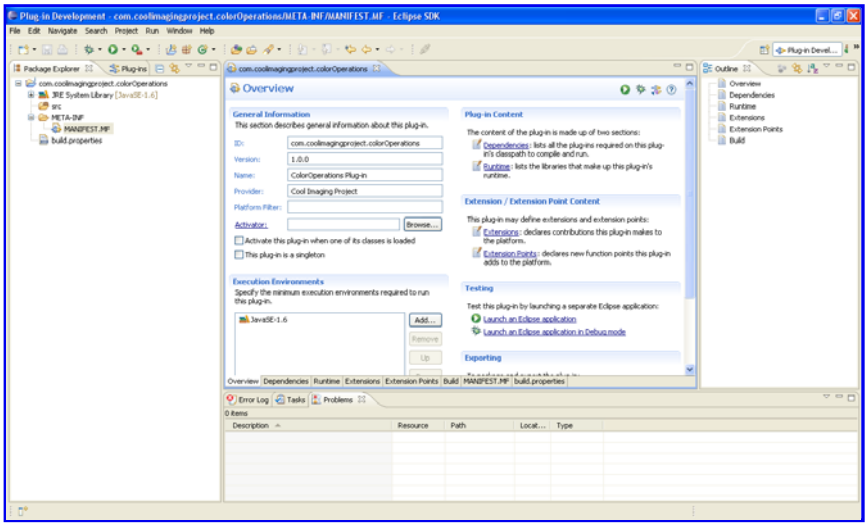
2. Dentro de la ventana que se muestra seleccionará *Plug-in Project* y hará *click* en "*Next >*".



3. Escribiremos el nombre del proyecto y volveremos a hacer *click* sobre "*Next* >". Para realizar el ejemplo propuesto completaremos los datos requeridos con los valores que se muestran en la imagen siguiente y cuando finalicemos haremos *click* en "*Finish*".
   Nosotros utilizaremos la convención de nombres de proyecto e identificadores propuesto por Eclipse.



Tras realizar estos pasos, es posible que Eclipse le pida confirmación para mostrar la perspectiva "*Plug-in Development perspective*". Esta perspectiva está optimizada para desarrollar *plug-in's*, a partir de este punto, las imágenes que se adjunten utilizarán esta perspectiva.

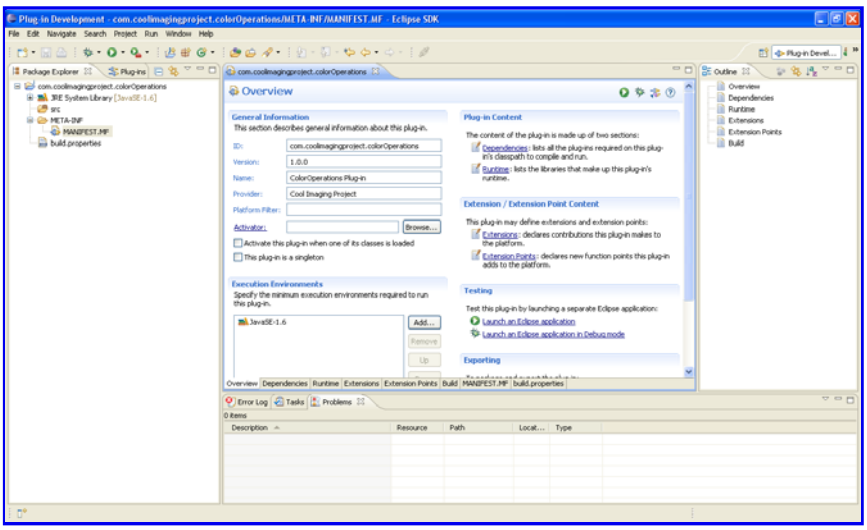


Ahora, nos encontramos en disposición de extender un punto de extensión.

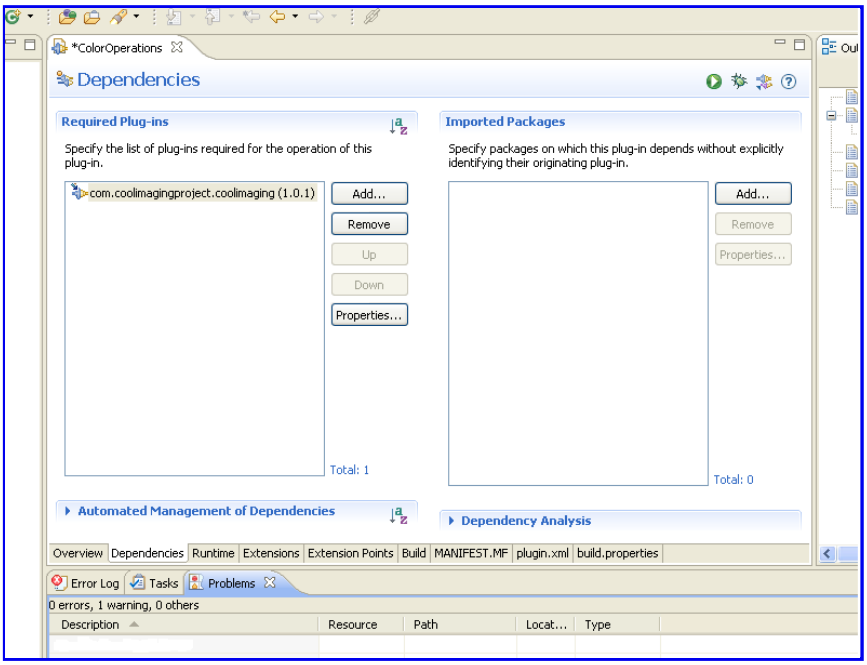
4.1.3. Cómo extender un punto de Extensión (*OperationApplication*)

Ya creado el proyecto, estamos en disposición de empezar a crear el *plug-in de operaciones* para la aplicación.

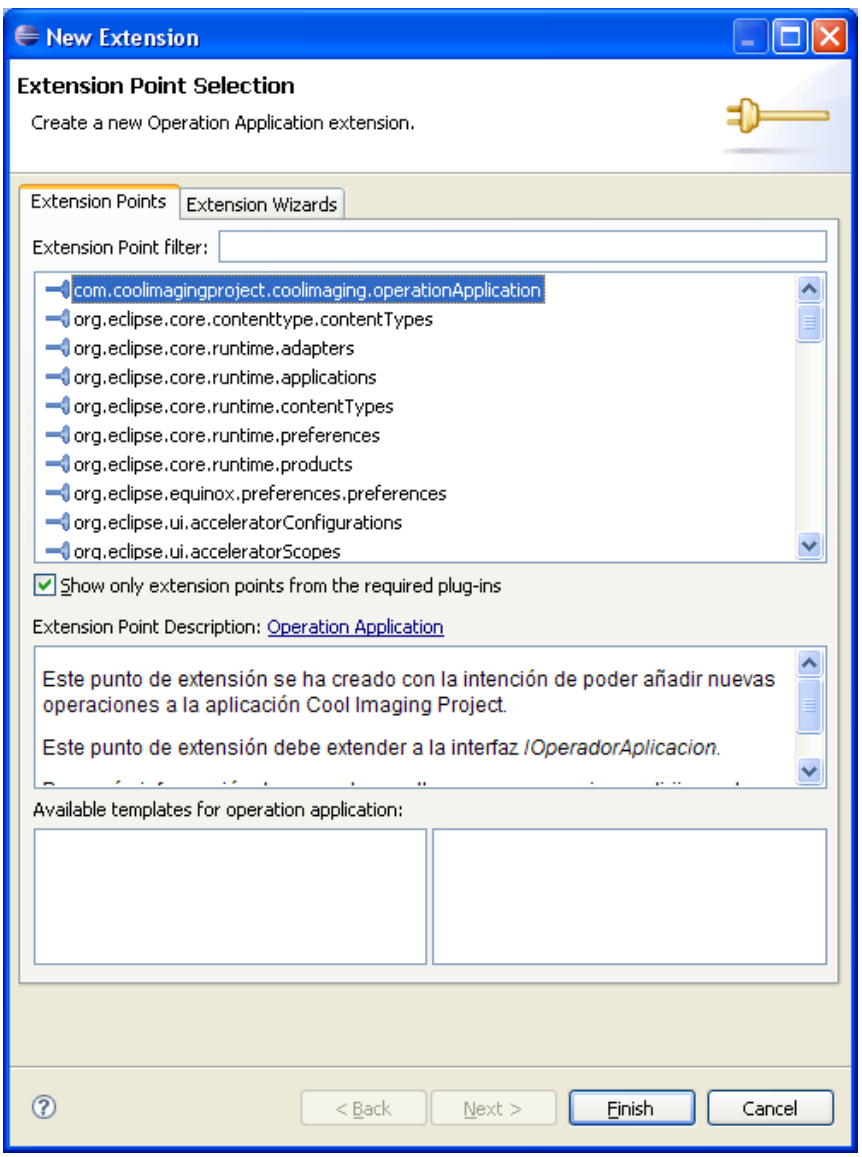
1. Para ello, abrimos el archivo *plugin.xml*, que enseñará una ventana similar a la de la imagen adjunta.



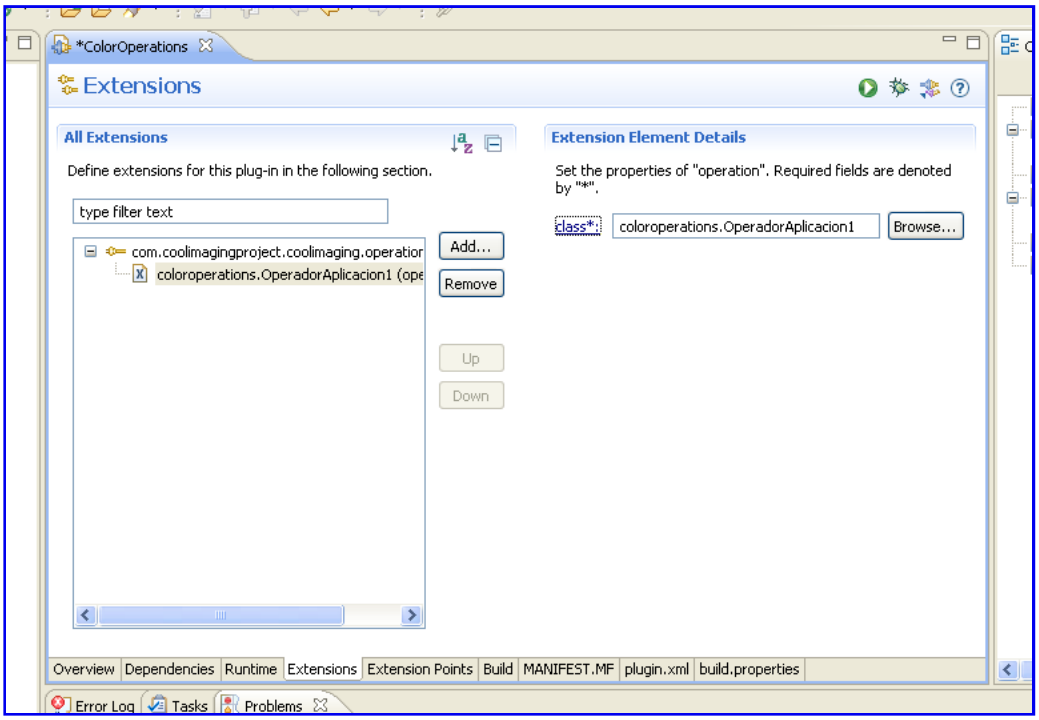
2. Accederemos a la pestaña "*Dependencies*" y en "*Required Plug-ins*" hacemos *click* en "**Add...**", donde añadiremos el proyecto *com.coolimagingproject.coolimaging* como se ve en la imagen inferior.



3. Pasaremos a la pestaña *Extensions*," donde haremos *click* en "**Add...**" y seleccionaremos "*com.coolimagingproject.coolimaging.operationApplication*".



4. Haremos *click* en "**Finish**" y se mostrará una ventana como la que se observa debajo.



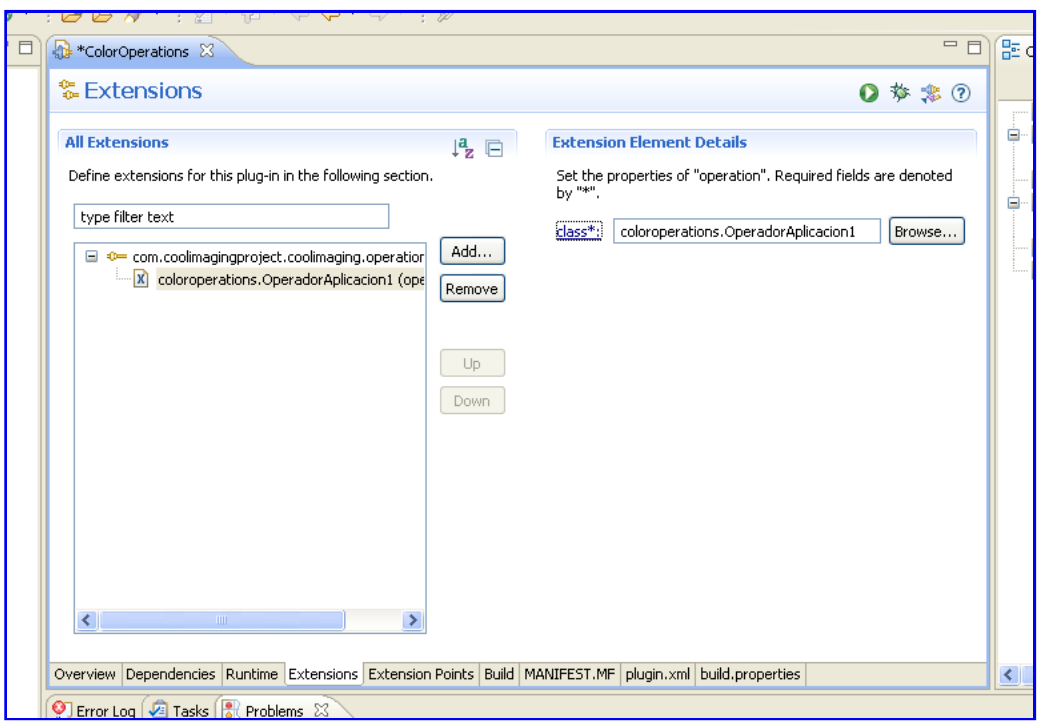
Llegados a este punto, ya podemos comenzar a implementar una operación para la aplicación.

4.1.4. Cómo implementar una nueva operación implementando la interfaz `IOperadorAplicacion`

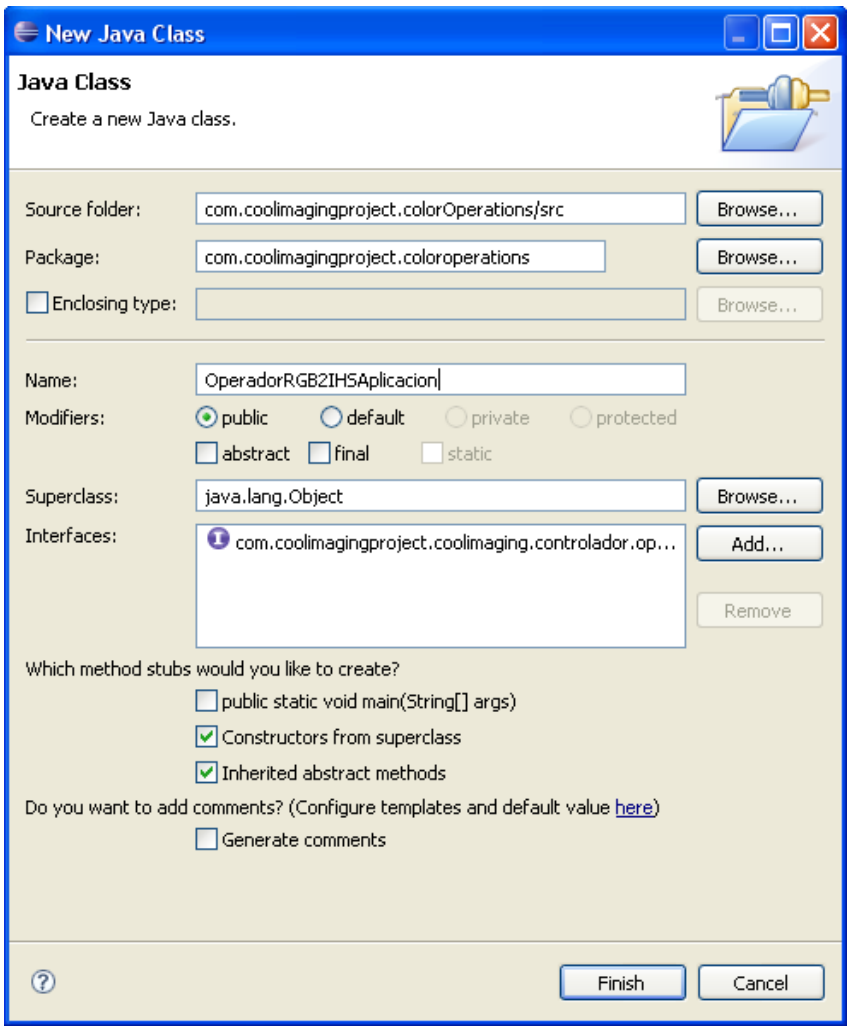
Una vez hayamos extendido el punto de extensión podemos empezar a implementar una operación que implemente la interfaz `IOperadorAplicacion`. En nuestro caso de ejemplo, implementaremos la operación de cambio de espacio de color RGB a IHS (`OperadorRGB2IHSAplicacion`).

Para implementar una operación que funcione en la aplicación, el desarrollador debe conocer algunas partes y clases que constituyen la aplicación. Por ello, esta sección se divide en varias secciones donde se explican cada una de las clases que pueden resultar de interés para el desarrollador.

Para empezar, el desarrollador debe definir una nueva clase que implemente la interfaz `IOperadorAplicacion`, y que a su vez, contendrá la nueva operación que queremos desarrollar, en nuestro caso `OperadorRGB2IHSAplicacion`.



Esta acción la llevaremos a cabo haciendo *click* sobre " **class*** " en la pestaña *Extensions* del archivo *plugin.xml*, como se observa en la imagen superior.
Tras ejecutar la acción, se verá una ventana similar a la de la imagen inferior.



Una vez se hayan rellenado los datos necesarios para la creación de la clase, se aceptará la operación haciendo "*click*" en **Finish**.

El código que genere la clase será similar al que se adjunta:

OperadorRGB2IHSAplicacion.java

```
package com.coolimagingproject.colorOperations.operadorAplicacion.color;

import org.eclipse.swt.widgets.Composite;

import com.coolimagingproject.coolimaging.controlador.operador.publico.IOperadorAplicacion;
import com.coolimagingproject.coolimaging.controlador.operador.publico.InfoOperadorAplicacion;
import
com.coolimagingproject.coolimaging.libreriaimagenes.operador.publico.ConjuntoParametroOperador;
import
com.coolimagingproject.coolimaging.vista.operador.panelesOperadores.publico.PanelInfoOperador;

public class OperadorRGB2IHSAplicacion implements IOperadorAplicacion {

  public OperadorRGB2IHSAplicacion() {
    // TODO Auto-generated constructor stub
  }

  @Override
  public InfoOperadorAplicacion getInfoOperador() {
    // TODO Auto-generated method stub
    return null;
```

```
  }

  @Override
  public PanelInfoOperador getPanelInfoOperador(Composite arg0) {
    // TODO Auto-generated method stub
    return null;
  }

  @Override
  public TipoIOperadorAplicacion getTipoOperacion() {
    // TODO Auto-generated method stub
    return null;
  }

  @Override
  public void insertarParametros(ConjuntoParametroOperador arg0) {
    // TODO Auto-generated method stub

  }

  @Override
  public Object operar() {
    // TODO Auto-generated method stub
    return null;
  }

}
```

Con la clase generada, estamos en disposición de empezar a implementar el `OperadorRGB2IHSAplicacion` de ejemplo.

Si desea más información acerca de los métodos que debe implementar, puede dirigirse a la documentación de la interfaz IOperadorAplicacion o a la sección Cómo actúa una operación en la aplicación de este mismo manual.

4.1.4.1. Cómo actúa una operación en la aplicación

Toda operación que quiera añadirse a los menús de operaciones de esta aplicación está representada mediante una clase que implementa la interfaz `IOperadorAplicacion.`

Dicha interfaz contiene una serie de métodos básicos que representan qué necesita una operación para poder ser incrustada dentro de ésta aplicación.
Toda operación tiene tres componentes principales, a saber el *panel de la operación*, la *operación en sí misma* e *información adicional* requerida para el funcionamiento de la operación.

Explicaremos estos tres conceptos de forma más detallada.

Cuando se hace referencia al *panel de la operación* nos referimos realmente (aunque no de forma exacta) al panel que se muestra en la aplicación cuando se hace *click* en alguna operación del menú de operaciones de tratamiento o del menú de operaciones de caracterización de imágenes.

Toda operación debe contener un panel donde el usuario debe ser capaz de seleccionar los parámetros necesarios para llevar a cabo la operación. Para realizar una convolución, por ejemplo, se necesita saber tanto cuál es el kernel de convolución como qué método es usado a la hora de tratar los bordes de la imagen. En el caso de una operación de rotación hay que especificar tanto el ángulo de rotación como el método de interpolación usado para corregir los errores que surgen cuando la imagen no se rota un múltiplo de 90 grados. Como se ve, cada operación contiene una serie de parámetros necesarios que especifican cómo dicha operación ha de operar sobre la imagen o imágenes de entrada. Por supuesto, también existen operaciones que no necesitan parámetros. Una operación NOT, por ejemplo, no requiere especificar un "ángulo de rotación", ni nada parecido.

La clase `PanelInfoOperador` es una clase abstracta que define la semántica del panel de datos de una operación concreta. Por ejemplo, el `PanelInfoOperadorRotacion` (panel perteneciente a la operación *Rotación*) es un panel (*Composite*) que define una serie de campos que especifican cómo la operación de rotación debe funcionar; dichos campos son: el ángulo de rotación; el centro de rotación; el método de interpolación usado para corregir los errores que surgen a consecuencia de

realizar la rotación; el método de extensión del borde para tratar los píxeles que caen fuera de la imagen, y un campo que indica si se quiere que la imagen sea centrada tras la rotación. Como se observa, el `PanelInfoOperadorRotacion` contiene una serie de campos que especifican cómo será llevada a cabo la rotación. El `PanelInfoOperadorRotacion`, como hemos dicho, extiende a la clase `PanelInfoOperador`, lo cual le proporciona la capacidad de ser un panel de operaciones que podrá ser insertado en la aplicación.

Esperamos que hasta aquí esté claro cómo funcionan los paneles asociados a las operaciones. En resumen, toda operación tiene asociado un `PanelInfoOperador` que recoge todos aquellos parámetros que especifican cómo la operación ha de ser llevada a cabo. Pasemos ahora a explicar cómo funciona esta clase.

La clase `PanelInfoOperador` define un método llamado **`public void getParametros(ConjuntoParametroOperador parametros)`**. Este método es un punto clave dentro de la clase `PanelInfoOperador`. La idea es que dicho método debe insertar en el objeto `parametros` los parámetros de la operación que se pueden extraer del panel. Debe tenerse en cuenta que la clase `ConjuntoParametroOperador` representa un conjunto de parámetros asociados a una determinada operación. El método **`getParametros(ConjuntoParametroOperador parametros)`** debe insertar en el objeto `parametros` los parámetros que pueden extraerse del panel y que sean necesarios para la realización de la operación. Debe tenerse en cuenta que cada *parámetro* está representado por un objeto de la clase `ParametroOperador` (para más información acerca de cómo funcionan las clases `ConjuntoParametroOperador` y `ParametroOperador`, se recomienda acceder al *javadoc* de ambas clases). Por ejemplo, el `PanelInfoOperadorRotacion`, en su método **`getParametros(ConjuntoParametroOperador parametros)`**, inserta en `parametros` los parámetros asociados al operador de rotación y que pueden ser extraídos del panel en sí, es decir: el ángulo de rotación, el centro de rotación, etc. En el siguiente extracto de código se puede ver la implementación de dicho método para la clase `PanelInfoOperadorRotacion`:

```
public void getParametros(ConjuntoParametroOperador parametros){

  parametros.insertarParametro(new ParametroOperador(
    OperadorRotacion.PARAMETRO_ANGULO_ROTACION, new Float(
     this.textoAnguloRotacion.getText())
     * new Float(Math.PI) / 180.0f));

  parametros.insertarParametro(new ParametroOperador(
     OperadorRotacion.PARAMETRO_CENTRO_ROTACION_X, new Float(
     this.textoCentroRotacionX.getText())));

  parametros.insertarParametro(new ParametroOperador(
     OperadorRotacion.PARAMETRO_CENTRO_ROTACION_Y, new Float(
     this.textoCentroRotacionY.getText())));

  parametros.insertarParametro(new ParametroOperador(
     OperadorRotacion.PARAMETRO_METODO_INTERPOLACION,
     this.listaMetodosInterpolacion.getInterpolacion()));

  parametros.insertarParametro(new ParametroOperador(
     OperadorRotacion.PARAMETRO_METODO_BORDE, this.listaMetodosBorde
     .getTratadorBorde()));

  parametros.insertarParametro(new ParametroOperador(
     OperadorRotacion.PARAMETRO_ROTAR_Y_CENTRAR,
     this.botonCentrarImagen.getSelection()));
}
```

Este método es el más importante de la clase `PanelInfoOperador`, ya que él es el que se encarga de recuperar los parámetros necesarios para la realización de la operación. Analicemos con mayor detalle el código fuente de arriba.

En el objeto `parametros` se insertan todos los parámetros que se pueden extraer del panel de la rotación. Como puede apreciarse, cada parámetro insertado (mediante la función `insertarParametro()`) es un objeto de tipo `ParametroOperador`. Cada parámetro tiene asociado dos elementos: un identificador del parámetro (valor `int`) y un `Object` que representa el valor del parámetro. Por ejemplo, el primer parámetro que se inserta se corresponde con el ángulo de rotación de la imagen. El valor numérico del ángulo de rotación es extraído del campo textual del panel que permite al usuario introducir el ángulo de rotación. Dicho valor numérico se corresponde con el valor del parámetro.

El parámetro, aparte de un valor, tiene asociado un identificador en forma de entero. Así por ejemplo, el primer parámetro insertado en el ejemplo de arriba tiene un identificador cuyo valor es la constante `OperadorRotacion.PARAMETRO_ANGULO_ROTACION`. El identificador del parámetro no es más que un valor entero que el

usuario, de forma arbitraria, asigna a cada uno de los parámetros que requiere su operación para ejecutarse. Por ejemplo, el usuario podría asignar el identificador 100 al ángulo de rotación o el identificador 101 al centro de rotación. Es importante que, sin embargo, **no haya parámetros con identificadores repetidos**. Así pues, el usuario no podría asignar tanto al ángulo de rotación como al centro de rotación identificadores ambos de 100. Es también importante notar que **el usuario no puede hacer uso de los identificadores que van del 0 al 99**. Dichos identificadores están reservados por la aplicación, y si el usuario hace uso de ellos, el resultado de la operación puede ser inesperado.

Como habrá pensado, el método `getParametros()` de la clase `PanelInfoOperador` realmente no recupera todos los parámetros necesarios para hacer la operación. Por ejemplo, habrá visto en el código fuente de arriba que el método `getParametros()` de la clase `PanelInfoOperadorRotacion` no inserta en el objeto `parámetros` de entrada el parámetro correspondiente a la imagen a rotar. Para que el `OperadorRotacionAplicacion` (que implementa la interfaz `IOperadorAplicacion`) pudiera operar de forma correcta, sería necesario que éste recibiera, además de los parámetros indicados ahí arriba, otro parámetro que contuviera la imagen a rotar, y que sería el siguiente:

```
new ParametroOperador(OperadorRotacion.PARAMETRO_IMAGEN_FUENTE, "objeto Imagen")
```

Se puede entender que, como parece ser, la clase `PanelInfoOperador` no es capaz de proporcionar al operador subyacente todos los parámetros que necesita para su funcionamiento. Más tarde aclararemos este pequeño detalle.

¿Dónde aparece la clase `PanelInfoOperador` dentro de la estructura de la aplicación?. Es aquí cuando es necesario regresar a la interfaz `IOperadorAplicacion`. La interfaz `IOperadorAplicacion` define un método, **`getPanelInfoOperador()`** que simplemente devuelve el `PanelInfoOperador` asociado al operador que implementa dicho `IOperadorAplicacion`. La aplicación, en última instancia, hará que se visualice dicho `PanelInfoOperador` cuando el usuario haga uso de la operación, y a partir de él, obtendrá los parámetros necesarios para la ejecución de la operación (pero no todos; recuerde que las imágenes necesarias para la ejecución de la operación no son extraídas mediante el `PanelInfoOperador`).

La interfaz `IOperadorAplicacion` define otro método importante, a saber, **`public void insertarParametros (ConjuntoParametroOperador parametros)`**. Este método es llamado desde la aplicación para registrar en el `IOperadorAplicacion` todos los parámetros que éste necesita para llevar a cabo la operación. La idea es que, registrados todos los parámetros que el operador necesita para su ejecución, se pueda llamar posteriormente al método **`operar()`**, para obtener así el resultado de la operación. Así, por ejemplo, en el `OperadorRotacionAplicacion`, la función **`insertarParametros(ConjuntoParametroOperador parametros)`** se debe encargar de, "de algún modo", guardar una copia de todos los parámetros que se le pasan mediante la función **`insertarParametros()`**. Cuando se llame al método **`operar()`** del `OperadorRotacionAplicacion`, dicho operador se encargará de, usando los parámetros que se le han pasado mediante el método **`insertarParametros`**, realizar la operación de rotación, y devolver el resultado.

Cada operación puede necesitar de una o varias imágenes de entrada para obtener el resultado. Por ejemplo, una operación de inversión de imagen, la NOT, requiere una única imagen para operar. Hay operaciones, sin embargo, que requieren de varias imágenes: una suma de imágenes, por ejemplo, requiere de dos imágenes para obtener la imagen resultado. El operador debe indicar a la aplicación, de algún modo, cuántas imágenes necesita para obtener su resultado. Ha de tenerse en cuenta que, dentro del paradigma del `IOperadorAplicacion`, las imágenes de entrada son otros parámetros de la operación. La diferencia es que dichas imágenes no se extraen del `PanelInfoOperador`. Como parámetros que son, el usuario, además, deberá informar a la aplicación de cuáles son los identificadores de parámetro asociados a las imágenes de entrada. La clase `InfoOperadorAplicacion` se encarga de ello.

El método `IOperadorAplicacion.getInfoOperador()` debe devolver un objeto de la clase `InfoOperadorAplicacion`, el cual indique tanto cuántas imágenes requiere el operador para obtener el resultado, como cuáles son los identificadores de parámetro asociados a dichas imágenes. La clase `InfoOperadorAplicacion`, además, proporciona un nombre para el operador y una breve descripción. Ambos podrán ser usados por la aplicación cuándo ésta estime oportuno.

La parte más importante del `InfoOperadorAplicacion` es la relativa a la información de las imágenes requeridas por la operación. El método `getInfoImagenes()` de esta clase devuelve un objeto de tipo `ConjuntoInformacionImagenesOperador`. Dicho objeto informa de cuántas imágenes se requieren para llevar a cabo la operación así como de cuál es su función y cuál es el identificador del parámetro asociado a cada una de esas imágenes. Así, la aplicación es capaz de extraer los últimos parámetros que faltan para la ejecución de una determinada operación, a saber, las imágenes sobre las que se aplica.

Así pues, una visión general de cómo funciona un operador dentro de la aplicación es la siguiente; usaremos el ejemplo del `OperadorRotacionAplicacion`:

- El método `getPanelInfoOperador()` de la clase `OperadorRotacionAplicacion` devuelve un objeto de la clase `PanelInfoOperadorRotacion`. Dicho panel, que extiende a la clase `PanelInfoOperador`, es el que se muestra dentro de la aplicación cuando el usuario quiere hacer uso de la operación, y el cual le permite al usuario introducir todos los parámetros de la rotación (ángulo de rotación, centro de rotación, etc.).

- Cuando la aplicación hace uso de la operación, ésta extrae del `PanelInfoOperadorRotacion` los parámetros necesarios para la ejecución de la operación de rotación. Para ello, llama al método `getParametros(ConjuntoParametroOperador parametros)`. Si la operación requiere imágenes de entrada, la aplicación construirá los parámetros asociados a la imágenes de la operación. Para ello, hará uso de la información proporcionada por el `InfoOperadorAplicacion` devuelto por el método `getInfoOperador()`. Todos estos parámetros se almacenan en un objeto `ConjuntoParametroOperador`
- Extraídos los parámetros, la aplicación llama al método `insertarParametros()` del `OperadorRotacionAplicacion` con el objeto `ConjuntoParametroOperador` anteriormente construido. El `OperadorRotacionAplicacion`, entonces, almacena todos los parámetros necesarios para la ejecución de la operación de rotación.
- Cuando se ejecuta la operación, la aplicación llama al método `operar()` del `OperadorRotacionAplicacion`. Dicho método, internamente, hace uso de los parámetros que se le registraron mediante `insertarParametros()`, y obtiene el resultado de la rotación, que devuelve.

Observe que lo bueno de la interfaz `IOperadorAplicacion` es que es independiente de cómo se realiza la operación a bajo nivel. El usuario no tiene que preocuparse más que de definir un `PanelInfoOperador` concreto, y un objeto `InfoOperadorAplicacion`. Del `PanelInfoOperador`, la aplicación extrae los *parámetros fundamentales* que son necesarios para la ejecución de la operación. el objeto `InfoOperadorAplicacion`, por otro lado, es usado para construir los parámetros de tipo Imagen que el operador requiera. Una vez que el `IOperadorAplicacion` ha recibido los parámetros que necesita para la operación, los usa como desee en su método `operar()`, para devolver posteriormente el resultado. En el caso del `OperadorRotatacionAplicacion`, simplemente se llevará a cabo la rotación de la imagen, que se devolverá como resultado de la operación.

Para finalizar esta sección, comentar que se proporcionan los códigos fuentes de la operación de rotación aquí explicada (OperadorRotacion.java, PanelInfoOperadorRotacion.java, OperadorRotacionAplicacion.java). Recordar también, que el `OperadorRotacion` hace uso de la biblioteca *Java Advanced Imaging (JAI)*. Si está interesado en utilizar esta biblioteca para sus operaciones, puede encontrar algunos enlaces de interés en la sección Bibliografía -> Java Advanced Imaging (JAI). Señalar también que se proporciona una clase, `PanelInfoOperadorVacio`, que representa un PanelInfoOperador vacío, sin capacidad de proporcionar parámetros. Esta clase puede ser usada en operaciones que no requieran parámetros, como la NOT, una suma de imágenes, etc.

4.1.4.2. Implementando `OperadorRGB2IHSAplicacion`

Operador

Ya nos encontramos en situación de empezar a implementar la clase `OperadorRGB2IHSAplicacion`. Pero antes de implementar esta clase, implementaremos la clase `OperadorRGB2IHS.java`, que contendrá el operador independiente a la aplicación, y que pretende hacer más fácil su reutilización en caso de ser necesario. Esta práctica de programación se recomienda en Buenas prácticas de programación.

A continuación, adjuntamos el código utilizado para esta operación.

`OperadorRGB2IHS.java`

```
package com.coolimagingproject.coloroperations.operador;

import com.coolimagingproject.coolimaging.libreriaimagenes.imagen.publico.Imagen;
import java.awt.Dimension;
import java.awt.image.ComponentColorModel;
import java.awt.image.RenderedImage;
import java.awt.image.renderable.ParameterBlock;

import javax.media.jai.IHSColorSpace;
import javax.media.jai.JAI;

import
com.coolimagingproject.coolimaging.libreriaimagenes.operador.publico.ConjuntoParametroOperador;
import com.coolimagingproject.coolimaging.libreriaimagenes.operador.publico.OperadorUnario;

/**
  * El operador {@link OperadorRGB2IHS} se encarga de cambiar una imagen del
  * espacio de color RGB a IHS (intensity-hue-saturation). Esta operación está
  * implementada con los operadores de JAI.
```

```
 *
 * @see OperadorUnario
 *
 * @author L.A González Jaime
 * @author R.J Palma Durán
 *
 */
public class OperadorRGB2IHS extends OperadorUnario {
   // Espacio de color IHS
   private IHSColorSpace CS_IHS;

   // Modelo de Color
   private ComponentColorModel sistemaColorIHS;
   // Imagen a la que cambiaremos el espacio de color
   private Imagen imagenFuente;
   // Parámetros de la operación a nivel de JAI
   protected ParameterBlock parametrosJAI = new ParameterBlock();

   /**
    * Constructor
    */
   public OperadorRGB2IHS(ConjuntoParametroOperador parametros){
     super(parametros);
     // Cogemos la imagen de los parámetros
     this.imagenFuente = (Imagen)parametros.getParametro(
     PARAMETRO_IMAGEN_FUENTE).getValorParametro();

     // Comprobamos que la imagen tenga tres bandas de color
     if(this.imagenFuente.getColorModel().getNumColorComponents() < 3)
       throw new IllegalArgumentException("El espacio de color tiene menos de 3 componentes");
     // Creamos el espacio de color
     this.CS_IHS = IHSColorSpace.getInstance();

     // Extraemos el tamaño de los componentes
      int [] numBits = this.imagenFuente.getColorModel().getComponentSize();

     // Creamos el sistema de color.
     this.sistemaColorIHS = new ComponentColorModel(this.CS_IHS,
     new int[]{ numBits[0], numBits[1], numBits[2]},
     this.imagenFuente.getColorModel().hasAlpha(), this.imagenFuente
       .getColorModel().isAlphaPremultiplied(),
     this.imagenFuente.getColorModel().getTransparency(),
     this.imagenFuente.getColorModel().getTransferType());
   }

  /*
   * (non-Javadoc)
   * @see operador.Operador#operar()
   */
   public Object operar(){

     // Esta línea se debe a un bug que tiene JAI. Para más información en
     // {https://jai-core.dev.java.net/issues/show_bug.cgi?id=30}
     JAI.setDefaultTileSize(new Dimension(this.imagenFuente.getTileWidth(),
       this.imagenFuente.getTileHeight()));

     this.parametrosJAI.addSource(this.imagenFuente);
     this.parametrosJAI.add(this.sistemaColorIHS);
     // Hacemos la conversión.
     Imagen ImagenResultado = new Imagen((RenderedImage)JAI.create(
       "colorconvert", this.parametrosJAI));
     return ImagenResultado;
```

```
    }
}
```

En la implementación de esta operación hemos hecho uso de clases que no se han explicado en secciones anteriores. Por ejemplo, la clase que extiende `OperadorRGB2IHS`, `OperadorUnario`. Esta clase se le ofrece al desarrollador para hacer uso de ella en caso que las operaciones que implemente requieran de una sola imagen fuente. Pero ésta, no es la única clase que se puede utilizar, sino que se ofrecen otras que pretenden facilitar la tarea del desarrollador. Si desea más información de las clases que se pueden utilizar, dirijase a la sección Clases Públicas.
Otra clase relevante que se ha utilizado, es la clase `Imagen`, objeto que retorna y del cuál hacen uso los operadores de tratamiento de imágenes, así como los operadores de caracterización.

A diferencia de los operadores de tratamiento, los operadores de caracterización retornan objetos de tipo `MC`. Por ser esta estructura propia de esta aplicación, en la sección Medidas de Caracterización se explica en detalle su funcionamiento y utilización.

El operador que se ha implementado no requiere el uso de constantes o parámetros adicionales a parte del que proporciona la clase `OperadorUnario` **`(PARAMETRO_IMAGEN_FUENTE)`**. Pero en caso que un operador requiera de parámetros adicionales, estos se pueden definir tal y cómo se expone en la sección Identificadores de Parámetros. O si desease información más detallada, puede encontrarla en la sección Cómo actua una operación en la aplicación de este manual.

Operador Aplicación

Habiendo definido el operador `OperadorRGB2IHS`, podemos pasar a implementar el operador del que hará uso la aplicación, es decir, el que tiene que implementar la interfaz `IOperadorAplicacion`.

Tal y como se ha explicado anteriormente, esta operación no requiere de parámetros adicionales, es decir, no hará uso de ningún otro parámetro a parte de la imagen fuente. Por eso utilizaremos la clase `PanelInfoOperadorVacio` que nos proporciona la aplicación para crear el panel de operaciones.

OperadorRGB2IHSAplicacion.java

```
package com.coolimagingproject.coloroperations;

import
com.coolimagingproject.coolimaging.libreriaimagenes.operador.publico.ConjuntoParametroOperador;
import com.coolimagingproject.coolimaging.libreriaimagenes.operador.publico.OperadorUnario;

import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Composite;

import com.coolimagingproject.colorOperations.operador.color.OperadorRGB2IHS;
import
com.coolimagingproject.coolimaging.controlador.operador.publico.ConjuntoInformacionImagenOperador;
import com.coolimagingproject.coolimaging.controlador.operador.publico.IOperadorAplicacion;
import com.coolimagingproject.coolimaging.controlador.operador.publico.InfoOperadorAplicacion;
import com.coolimagingproject.coolimaging.controlador.operador.publico.InformacionImagenOperador;
import
com.coolimagingproject.coolimaging.vista.operador.panelesOperadores.publico.PanelInfoOperador;
import
com.coolimagingproject.coolimaging.vista.operador.panelesOperadores.publico.PanelInfoOperadorVacio;


/**
  * Operador que realiza el cambio del espacio de color de RGB a IHS de una
  * imagen en la aplicación.
  *
  * @author L.A González Jaime
  * @author R.J Palma Durán
  */
  public class OperadorRGB2IHSAplicacion implements IOperadorAplicacion{
   protected PanelInfoOperador panelInfoOperador;
   protected OperadorRGB2IHS operador;
```

```
   protected ConjuntoParametroOperador parametros;
   protected InfoOperadorAplicacion infoOperador;

/**
  * Constructor por defecto sin argumentos.
  */
  public OperadorRGB2IHSAplicacion(){
     crearInfoOperador();
  }

/*
  * (non-Javadoc)
  * @see
  * com.coolimagingproject.controlador.operador.IOperadorAplicacion#getPanelInfoOperador
  * (org.eclipse.swt.widgets.Composite)
  */
  public PanelInfoOperador getPanelInfoOperador(Composite parent){
     this.panelInfoOperador = new PanelInfoOperadorVacio("Realiza la conversión de RGB a IHS",
       "Operación que realiza el cambio de espacio de color \n"
       + "de una imagen RGB a IHS \n\n"
       + "Esta operación realmente no comprueba si el sistema de \n"
       + "color es RGB, solo comprueba si es una imagen con tres bandas \n"
       + "de color. Esto quiere decir, que si el sistema de color es otro \n"
       + "que no sea RGB el resultado puede ser inesperado.", parent, SWT.NONE);
     return this.panelInfoOperador;
  }

  /*
   * (non-Javadoc)
   * @see com.coolimagingproject.controlador.operador.IOperadorAplicacion#operar()
   */
  public Object operar(){
     this.operador = new OperadorRGB2IHS(this.parametros);
     return this.operador.operar();
  }

  /*
   * (non-Javadoc)
   * @see
   * com.coolimagingproject.controlador.operador.IOperadorAplicacion#insertarParametros
   * (operador.ConjuntoParametroOperador)
   */
   public void insertarParametros(ConjuntoParametroOperador parametros){
     this.parametros = parametros;
   }

  /*
   * (non-Javadoc)
   * @see
   * com.coolimagingproject.controlador.operador.IOperadorAplicacion#getTipoOperacion()
   */
   public TipoIOperadorAplicacion getTipoOperacion(){
     return IOperadorAplicacion.TipoIOperadorAplicacion.OPERADOR_TRATAMIENTO_IMAGEN;
   }

  /*
   * (non-Javadoc)
   * @see
   * com.coolimagingproject.controlador.operador.IOperadorAplicacion#getInfoOperador()
   */
   public InfoOperadorAplicacion getInfoOperador(){
     return this.infoOperador;
   }
```

```
  /**
   * Método auxuliar que inicializa la variable this.infoOperador, creando el objeto
   * InfoOperadorAplicaicon que contiene la información asociada a este IOperadorAplicacion.
   */
   private void crearInfoOperador(){
     ConjuntoInformacionImagenOperador info = new ConjuntoInformacionImagenOperador();
     info
       .insertarInformacionImagen(new InformacionImagenOperador("Imagen",
       OperadorUnario.PARAMETRO_IMAGEN_FUENTE));
     this.infoOperador=new InfoOperadorAplicacion(
       "RGB 2 IHS", "Parámetros del operador RGB 2 IHS", info);
   }
}
```

Una vez implementada esta clase, ya se da por finalizada la implementación de este operador.

A partir de este punto, puede volver a repetir el proceso de *extender el punto de extensión* e implementar nuevas operaciones tantas veces como desee.

Cuando haya terminado de implementar todas las operaciones querrá poder incluirlas dentro de la aplicación. Para ello, tendrá que pasar a crear el *plug-in* que se podrá incrustar en la aplicación principal.

De manera adicional, podría resultarle de interés que las cadenas de caracteres que aparecen inmersas en el código fuente estuviesen parametrizadas en ficheros externos, de tal forma, que se pudiesen traducir a otros idiomas sin tener que modificar el código. Si usted quiere llevar a cabo esta tarea, conocida como internacionalización, en la bibliografía le adjuntamos varios manuales que le pueden servir de ayuda.

4.1.4.3. Medidas de Caracterización

Las operaciones de caracterización de imágenes computan datos a partir de una imagen de entrada, la imagen caracterizada.

Dentro de *Cool Imaging*, todo dato de caracterización computado a partir de una imagen es conocido como *"medida de caracterización"*.Conceptualmente, una medida de caracterización no es más que una estructura que almacena valores numéricos. Así por ejemplo, una medida de caracterización podría representar la media de los píxeles de cada banda de una imagen. Esta medida de caracterización estaría formada por tantos valores numéricos como bandas tuviera la imagen, cada valor numérico representando la media de los píxeles de cada banda. Otra medida de caracterización podría representar la desviación típica calculada sobre todos los píxeles de todas las bandas de la imagen. En este caso, la medida de caracterización consistiría en un único valor numérico, a saber, la desviación típica.

Una medida de caracterización está representada por la clase abstracta `MC`. Todas las clases que representan medidas de caracterización heredan de la clase `MC`.

Como hemos visto, la complejidad de una medida de caracterización puede variar. Volvamos al ejemplo de la medida de caracterización que calculaba la media de los píxeles de cada una de las bandas de la imagen. Supongamos que la imagen sobre la que se calcula dicha medida consta de tres bandas. En ese caso, la medida estaría compuesta por tres valores numéricos, las medias de los valores de los píxeles de cada banda. Dicha medida de caracterización podría representarse como {X, Y, Z}, donde X, Y y Z serían los valores de las medias. Un ejemplo concreto podría ser {32.4, 234.1, 109.47}. Por otro lado, la medida de caracterización que calcula la desviación típica de los píxeles de todas las bandas de la imagen constaría de, simplemente, un valor numérico. Un ejemplo de dicha medida podría ser 10.45.

Se puede apreciar que, en efecto, la estructura de una medida de caracterización puede variar en complejidad. La clase `MC` da soporte para construir medidas de caracterización tan complejas como se quiera, siempre siguiendo una *filosofía vectorial.*

La clase `MC` tiene dos clases hijas principales, a saber, la clase `MCNumerica` y la clase `MCArray`.

La clase `MCNumerica` representa toda medida de caracterización que puede ser representada como un único valor numérico. Por ejemplo, la medida de caracterización que calcula la desviación típica de todas las bandas de la imagen sería una `MCNumerica`, ya que ésta puede ser representada mediante un único valor numérico. La clase `MCNumerica` es una clase abstracta. Sus clases hijas son `MCByte`, `MCDouble`, `MCFloat`, `MCInteger`, `MCLong` y `MCShort`. Cada una de estas subclases representa una medida de caracterización numérica cuyo valor numérico consta de un tipo concreto. Por ejemplo, el valor numérico de la clase MCDouble es de tipo Double, mientras que el valor numérico de la clase `MCLong` es de tipo `Long`.

La clase `MCNumerica` es la clase más básica de las que representan medidas de caracterización, pues representa un simple número, un valor numérico.

Es la clase `MCArray` la que añade complejidad a la estructura de una medida de caracterización. Por ejemplo, la medida de caracterización que contiene las medias de cada una de las bandas de la imagen no puede representarse como una `MCNumerica` en caso de que la imagen tenga más de una banda. Volviendo al ejemplo anterior, supongamos que tenemos las medias de las tres bandas de la imagen: {32.4, 234.1, 109.47}. Esta medida de caracterización no puede reprentarse como una `MCNumerica`, ya que consta de tres valores numéricos, en vez de uno sólo. Es aquí donde interviene la clase `MCArray`. La clase `MCArray` representa una `MC` capaz de albergar uno o varios objetos de tipo `MC`. De este modo, la medida de caracterización anterior podría modelarse fácilmente como una `MCArray` que contuviera tres objetos `MCNumerica`, cada uno con las respectivas medias de la imagen.

La potencia de la clase `MCArray` va más allá. Obsérvese que un objeto de la clase `MCArray` representa un conjunto con uno o varios objetos de tipo `MC`. Dado que la clase `MCArray` hereda de `MC`, tenemos que una `MCArray` puede albergar a su vez otras `MCArray`.

De este modo, la estructura de una `MC` no tiene por qué quedarse al nivel de un simple array de números. Una `MC` puede representar una estructura de valores numéricos anidados tan compleja como se quiera.

Por ejemplo, la siguiente estructura de valores numéricos podría ser representada: {3, 5, 7, {4, 5, 6}, 6, {7, {8, 9}, 10}, 13}, y su estructura sería la siguiente:

```
-- MCArray
   |
   |-- MCNumerica (3)
   |
   |-- MCNumerica (5)
   |
   |-- MCNumerica (7)
   |
   |-- MCArray
   |    |
   |    |-- MCNumerica (4)
   |    |
   |    |-- MCNumerica (5)
   |    |
   |    |-- MCNumerica (6)
   |
   |-- MCNumerica (6)
   |
   |-- MCArray
   |    |
   |    |-- MCNumerica (7)
   |    |
   |    |-- MCArray
   |    |    |
   |    |    |-- MCNumerica (8)
   |    |    |
   |    |    |-- MCNumerica (9)
   |    |
   |    |-- MCNumerica (10)
   |
   |-- MCNumerica (3)
```

La clase `MC`, a parte de definir valores numéricos, tiene asociados tanto un nombre como una descripción. Cuando el usuario crea un objeto de tipo `MC`, tiene la posibilidad de especificar tanto un nombre como una descripción para dicha medida de caracterización. Mediante dichos atributos se le puede asociar una cierta semántica a cada uno de los valores numéricos de los que consta la medida de caracterización. Volviendo al ejemplo de la medida de caracterización que contiene las medias de las tres bandas de la imagen, se le podrían asociar nombres y descripciones como los que siguen:

```
-- MCArray; nombre: "Media"; descripción: "Medias de cada una de las bandas de la imagen".
   |
   |-- MCNumerica; nombre: "Media 1"; descripción: "Media de la banda 1 de la imagen".
```

```
|
|-- MCNumerica; nombre: "Media 2"; descripción: "Media de la banda 2 de la imagen".
|
|-- MCNumerica; nombre: "Media 3"; descripción: "Media de la banda 3 de la imagen".
```

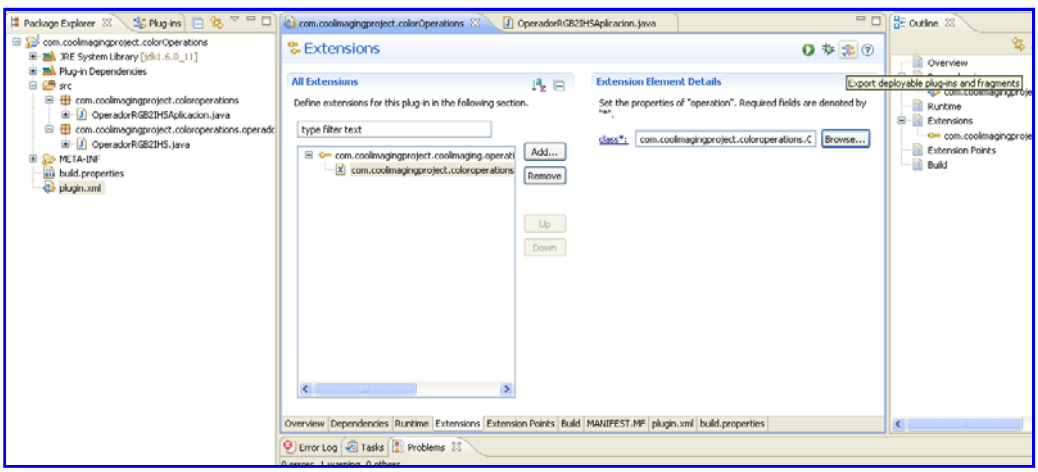
De este modo, la `MC` no tiene solamente valores numéricos, sino que a cada valor numérico se le asocia un significado. Obsérvese, de todos modos, que no es obligatorio asignar un nombre o una descripción a cada `MC` creada, aunque así se recomienda.

Toda operación de caracterización devuelve un objeto de tipo `MC`. Dicho objeto representa una serie de datos calculados a partir de la imagen de entrada. La complejidad del objeto `MC` devuelto depende de la semática de la operación. Unas operaciones podrán devolver `MC` muy complejas, mientras que la mayoría devolverán `MC` muy simples.

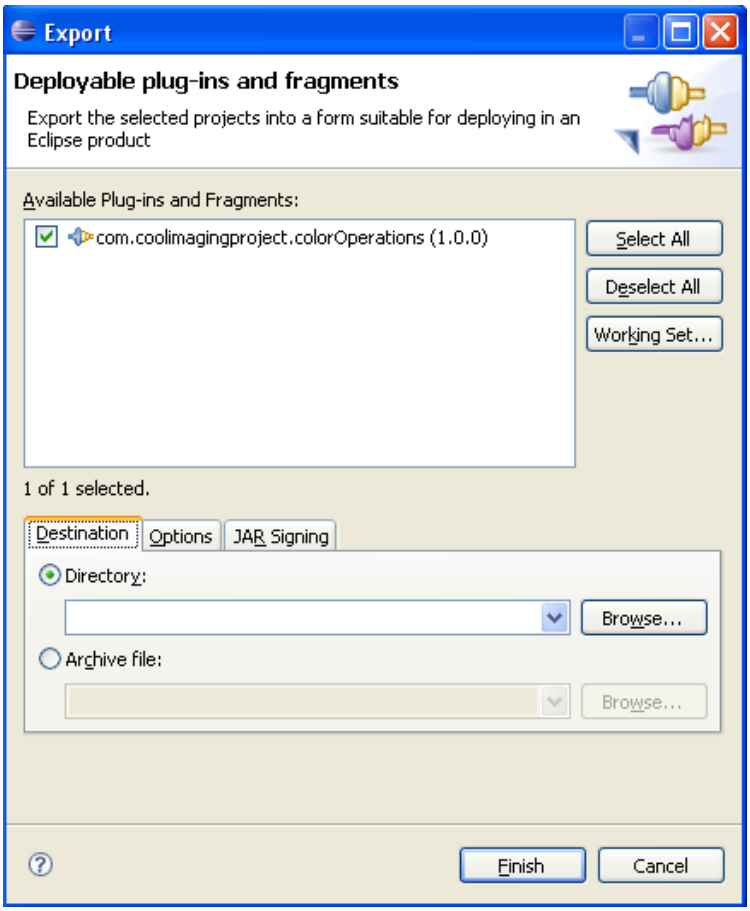
4.1.5. Crear un *Plug-in*

Esta sección la dedicaremos a explicar los pasos que hay que seguir para poder crear un *plug-in* de operaciones, que será el siguiente paso que se ha de realizar una vez que se desarrolle todo el conjunto de operaciones que se quieren englobar en un paquete *jar*.

Para ello, lo primero que debemos hacer es abrir el archivo *plugin.xml* del proyecto. Y en cualquiera de las pestañas, se mostrará un icono en la parte superior derecha similar al de la figura  sobre el que haremos *click*. Para más detalle, si situa el ratón sobre él, se podrá leer *'Export deployable plug-ins and fragments'* como se muestra en la imagen que se adjunta.



Tras realizar la acción, se mostrará una ventana como la siguiente:



Seleccionaremos el proyecto del cual queremos generar el *plug-in*, así como la ubicación donde guardarlo. Tras elegir la ubicación haremos *click* en **Finish** para generar el *plug.in*.

Con el *plug-in* generado, lo único que nos queda para poder integarlo en la aplicación es crear el índice.

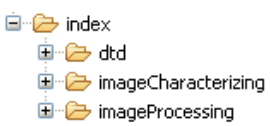
4.1.6. Crear un índice para la aplicación

Con el *plug-in* de operaciones ya generado, el último paso que nos queda para integrarlo en la aplicación es la creación del índice *xml* que indexará las operaciones en el menú de tratamiento de imágenes o en el menú de caracterización de imágenes, así como en la barra de menú.

Para una mejor comprensión a la hora de generar los índices dentro de la aplicación, dividiremos esta sección en dos partes, una primera en la que se detallará la estructura de carpetas que se utiliza dentro de la aplicación. Y una segunda parte, donde se describirán las etiquetas y estructura que se usa en el índice *xml*.

Estructura de directorios de los índices

Los índices que se utilizan para generar los menús de la aplicación se encuentran dentro de la carpeta *index* del directorio de la aplicación. Si accede a su interior, podrá observar que existen tres carpetas: *dtd*, *imageProcessing* y *imageCharacterizing*.



- **dtd**
  Esta carpeta contiene los *Document Types Declaration (DTD)* que deben cumplir los índices. No es una carpeta relevante para el desarrollador, pero se hará uso de los documentos que existen en su interior para comprobar la corrección de la estructura de los índices.
- **imageCharacterizing**
  Esta carpeta contiene los índices que se encargan de generar la estructura de menús de las operaciones de caracterización de imágenes

- **imageProcessing**
  Esta carpeta contiene los índices que se encargan de generar la estructura de menús de las operaciones de tratamiento de imágenes

La estructura de índices de ambos menús (menú de caracterización y menú de tratamiento) son similares, con la salvedad que si se introduce alguna operación de caracterización en los índices de tratamiento, o alguna operación de tratamiento en los índices de caracterización, las operaciones no se cargarán en la aplicación y se mostrará un mensaje de error al arrancarla. Por lo tanto, en caso que el usuario implemente un mismo *plug-in* con operaciones de tratamiento y caracterización *(práctica no recomendada)*, éste deberá crear al menos dos ficheros de índices.

Etiquetas y estructura de los índices

Tras la introducción a la estuctura de carpetas utilizada, nos encontramos en disposición de pasar a detallar las etiquetas y estructura interna de un índice.
Para facilitar la comprensión de esta estructura, explicaremos la estructura y cada una de las etiquetas a través de un posible índice que utilizaría el *plug-in* de operaciones de color generado en el apartado Crear un *plug-in*.

com.coolimagingproject.colorOperations.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE image_processing_index SYSTEM "index/dtd/definitionImageProcessing.dtd">
<image_processing_index>
  <category>
    <name>Color</name>
    <name lang="es">Color</name>
    <name lang="en">Color</name>
    <category>
      <name>IHS-HSI</name>
      <name lang="es">IHS-HSI</name>
      <name lang="en">IHS-HSI</name>
      <operation>

<class_location>com.coolimagingproject.coloroperations.OperadorRGB2IHSAplicacion</class_location>
      </operation>
    </category>
  </category>
</image_processing_index>
```

La primera linea que aparece en el documento, se refiere a la versión xml y el tipo de codificación utilizada.
Esta linea será dependiente de la versión y la codificación que el desarrollador desee utilizar.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

La segunda linea que aparece en el documento hace referencia al *document types declaration* que debe verificar el archivo de índice antes de ser cargado en la aplicación. Este archivo se encuentra en la carpeta *dtd* de la estructura de directorios de índices.

Esta linea dependerá del tipo de índice que se quiera generar. En caso de ser un índice de *tratamiento de imágenes* deberá insertar esta linea:

```
<!DOCTYPE image_processing_index SYSTEM "index/dtd/definitionImageProcessing.dtd">
```

Y en caso de querer generar un índice de *caracterización de imágenes* la línea que deberá insertar será esta otra:

```
<!DOCTYPE image_characterizing_index SYSTEM "index/dtd/definitionImageCharacterizing.dtd">
```

Estas dos lineas forman parte de la cabecera del documento, mientras que las etiquetas que vamos a pasar a describir a continuación, forman parte de las etiquetas propias de la estructura de árbol de un documento *xml*.

La siguiente etiqueta que aparece es `<image_processing_index>`, etiqueta que se utiliza dentro de los índices de *tratamiento de imágenes*. Si el índice que estamos creando es de caracterización, la etiqueta que debe aparecer es `<image_characterizing_index>` en vez de `<image_processing_index>`. Es importante que **solamente exista una etiqueta de este tipo** dentro de cada archivo de índice, así como considerar que la etiqueta de cierre se debe corresponder con esta etiqueta (`</image_processing_index>` o `</image_characterizing_index>`).

Seguidamente, en el ejemplo nos encontramos con la etiqueta `<category>`. Esta etiqueta representa una categoría dentro del menú. ¿Cómo se le asigna un nombre a una categoría? el nombre se expresa entre etiquetas `<name>` y puede ser codificado con caracteres *Unicode*. En nuestro ejemplo le hemos asignado el nombre *"Color"*.Es muy importante que esta etiqueta aparezca **siempre** a continuación de la etiqueta `<category>` y al menos una vez. ¿Por qué debe aparecer al menos una vez? Esto se debe a que la aplicación permite la internacionalización de los índices. Si se fija en el caso de ejemplo, las siguientes etiquetas `<name>` vienen acompañadas del atributo `lang`. Dependiendo del idioma en el que se ejecute la aplicación se utilizará la etiqueta `<name>` más adecuada en caso de aparecer varias de ellas. Por ejemplo, en caso que la aplicación se ejecute en *español (es)* se utilizará el nombre que contenga la etiqueta `<name lang="es">`, en vez de las otras que existan; pero si la aplicación se ejecutase en *francés (fr)*, se utilizaría la etiqueta `<name>` por ser la más genérica y no existir la etiqueta `<name lang="fr">`. No sólo se puede hacer uso del atributo *lang*, sino que también se pueden utilizar los atributos *country* y *variant* junto con la etiqueta *lang*.
Las combinaciones mostradas con el idioma español y país España son las consideradas como válidas:

- `<name>`
- `<name lang="es">`
- `<name lang="es" country="ES">`
- `<name lang="es" country="ES" variant="linux">`

En caso de existir varias etiquetas `<name>` con distintos atributos, la etiqueta que se utilizará para el índice será la más apropiada a las preferencias de la aplicación, es decir, fijándonos en la lista de arriba, se empezaría buscando de abajo a arriba.

Los valores que utilizan los atributos `lang` y `country` están definidos en dos estándares. El estándar *ISO 639:1988* define los códigos de idioma, mientras que el estándar *ISO 3166* define los códigos de país. Puede encontrar referencias a estos estándares en la sección bibliografía->Internacionalización.

Otra posibilidad que ofrece la etiqueta `<category>` es la de anidar tantas categorías como queramos. En el índice de ejemplo anidamos la categoría "IHS-HSI" dentro de la categoría "Color". La única restricción que debemos considerar es que esta nueva etiqueta `<category>` debe de aparecer tras terminar de definir todas las etiquetas `<name>` para una categoría.

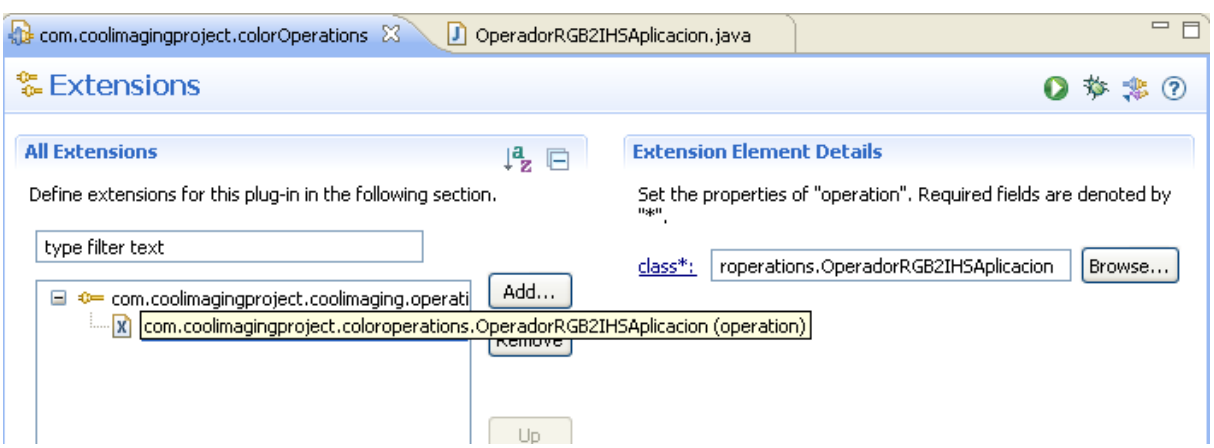
La siguiente etiqueta en la estructura de índices es la etiqueta `<operation>`, encargada de albergar toda la información relativa a una operación. En nuestro caso de ejemplo, esta etiqueta aparece tras la etiqueta `<name>` de la categoría "IHS-HSI". Pero ésta no es la única posición válida de esta etiqueta dentro de la estructura de índices, sino que se puede situar en otras posiciones. Las posiciones válidas dentro del índice son:

- Tras la última etiqueta `<name>` que defina a una categoría.
- Tras la etiqueta `<image_processing_index>` o `<image_characterizing_index>`, dependiendo del índice.
- Tras una etiqueta `</category>`

A diferencia de las categorías, las operaciones tienen una estructura rígida que siempre se debe cumplir, y es la siguiente:

```
<operation>
  <class_location>com.coolimagingproject.coloroperations.OperadorRGB2IHSAplicacion</class_location>
</operation>
```

Tras la etiqueta `<operation>` debe aparecer la etiqueta `<class_location>`. Entre estas etiquetas escribiremos la ubicación de la clase dentro de nuestro *plug-in*, que coincide con la *extensión* de la clase `OperadorRGB2IHSAplicacion` que se implementó y se encuentra definida en el archivo *plugin.xml*, tal y como se muestra en la imagen.



En el índice existirán tantas entradas operación como extensiones hayamos definido en el *plug-in*.

Con estas explicaciones debería poder generar sus propios índices personalizados, sin embargo, debe tener en cuenta que **las categorías no pueden ser vacías**, es decir, estas deben contener alguna operación, o el resultado puede ser inesperado; y los archivos de las categorías deben ser **.xml**.
En caso que cometa algún error, la aplicación le avisará con un mensaje de error.

4.2. Clases Públicas

Esta aplicación proporciona una serie de clases que el usuario puede utilizar para crear nuevas operaciones en la aplicación. Los paquetes que contienen estas clases se caracterizan por terminar el nombre del paquete con la palabra *publico*.

Puede encontrar más información de estas clases en el *Javadoc* que se encuentra en la sección Reference.

- Package com.coolimagingproject.coolimaging.controlador.operador.publico
  Este paquete contiene principalmente las clases necesarias para poder extender la aplicación mediante plug-ins. Si desea más información acceda a la sección como actúa una operación en la aplicación

- Package com.coolimagingproject.coolimaging.libreriaimagenes.imagen.publico
  Este paquete contiene la clase Imagen, que utilizan todas las operaciones de tratamiento y caracterización de imágenes. Esta clase es crucial en el desarrollo de operadores.

- Package com.coolimagingproject.coolimaging.libreriaimagenes.operador.publico
  Este paquete contiene abstracciones de operadores que el usuario puede utilizar para el desarrollo de sus propios operadores.

- Package com.coolimagingproject.coolimaging.modelo.caracterizacion.publico
  Este paquete contiene las clases necesarias para la gestión de las medidas de caracterización (MC). Estructura que manipulan las operaciones de caracterización.

- Package com.coolimagingproject.coolimaging.utilidades.publico
  Este paquete contiene clases que pueden resultar útiles a nivel general.

- Package com.coolimagingproject.coolimaging.vista.operador.panelesOperadores.publico
  Este paquete proporciona Paneles que puede utilizar en operadores para la aplicación.

- Package com.coolimagingproject.coolimaging.vista.utilidades.publico
  Este paquete proporciona soporte para mostrar diálogos emergentes en la aplicación.

4.2.1. Package Class **Use Tree Deprecated Index Help**

PREV PACKAGE NEXT PACKAGE **FRAMES NO FRAMES All Classes**

Package com.coolimagingproject.coolimaging.controlador.operador.publico

Interface Summary	
IOperadorAplicacion	La interfaz `IOperadorAplicacion.TipoIOperadorAplicacion` representa la interfaz que deben seguir todos los operadores que la aplicación contempla.

Enum Summary	
IOperadorAplicacion.TipoIOperadorAplicacion	El enumerado TipoIOperadorAplicación define dos constantes que indican si un IOperadorAplicacion es un operador de tratamiento de imágenes o bien uno de caracterización de imágenes.

Package Class **Use Tree Deprecated Index Help**

PREV PACKAGE NEXT PACKAGE **FRAMES NO FRAMES All Classes**

Package com.coolimagingproject.coolimaging.libreriaimagenes.imagen.publico

Class Summary	
ComCoolimagingprojectCoolimagingLibreriaimagenesImagenPublicoNLS	
Imagen	La clase Imagen representa una imagen bidimensional, que puede ser accedida tanto en modo lectura como en modo escritura.

Package com.coolimagingproject.coolimaging.libreriaimagenes.operador.publico

Class Summary	
ConjuntoParametroOperador	La clase `ConjuntoParametroOperador` representa un almacen de objetos tipo `ParametroOperador` .
Operador	La clase Operador representa la clase abstracta de la cual cuelga toda la jerarquía de clases de tipo operador.
OperadorBinario	La clase OperadorBinario representa la base de la jerarquía de aquellos operadores que necesitan dos imágenes para ser operados (a parte de, por supuesto, otros parámetros).
OperadorUnario	La clase OperadorUnario representa la base de la jerarquía de aquellos operadores que necesitan una sola imagen para ser operados (a parte de, por supuesto, otros parámetros).
ParametroOperador	La clase ParametroOperador representa un parámetro genérico de una operación.

Package com.coolimagingproject.coolimaging.modelo.caracterizacion.publico

Class Summary	
MC	La clase MC (de *Medida de Caracterización*) representa un elemento constitutivo de un VC (*Vector de Caracterización*).
MCArray	La clase MCArray representa una MC que internamente almacena un array de MC, es decir, un MC[].
MCByte	La clase MCByte representa una MC cuyo valor interno almacenado es un byte.

MCDouble	La clase MCDouble representa una MC cuyo valor interno almacenado es un double.
MCFloat	La clase MCFloat representa una MC cuyo valor interno almacenado es un float.
MCInteger	La clase MCInteger representa una MC cuyo valor interno almacenado es un int.
MCLong	La clase MCLong representa una MC cuyo valor interno almacenado es un long.
MCNumerica	La clase MCNumerica representa una MC cuyo valor es un número.
MCShort	La clase MCShort representa una MC cuyo valor interno almacenado es un short.

4.2.5. Overview **Package** Class **Use** **Tree** **Deprecated** **Index** **Help**

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES All Classes

Package com.coolimagingproject.coolimaging.utilidades.publico

Class Summary	
Pair<T,S>	Clase Pair.
UtilidadesGlobales	La clase UtilidadesGlobales contiene una serie de métodos y constantes que, por ausencia de otro lugar mejor donde colocarlos, se han decidido colocar en una clase global que los agrupe a todos.

4.2.6. Overview **Package** Class **Use** **Tree** **Deprecated** **Index** **Help**

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES All Classes

Package com.coolimagingproject.coolimaging.vista.operador.panelesOperadores.publico

Interface Summary	
IValidadorDatos	La interfaz IValidadorDatos representa una interfaz que implementan aquellas clases gráficas que disponen de una serie de campos editables por el usuario, y cuya validez debe ser evaluada.

Class Summary	
PanelInfoOperador	La clase PanelInfoOperador representa la base de la jerarquía de paneles (heredando de Composite) gráficos de los cuales se extraen los datos necesarios para la construcción del Operador asociado a dicho panel.
PanelInfoOperadorVacio	Clase que representa un PanelInfoOperador vacío, que no devuelve parámetros.

4.2.7. Overview **Package** Class **Use** **Tree** **Deprecated** **Index** **Help**

Package com.coolimagingproject.coolimaging.vista.utilidades.publico

Class Summary	
ComCoolimagingprojectCoolimagingVistaUtilidadesPublicoNLS	
DialogoDetalles	La clase DialogoDetalles representa un diálogo emergente que muestra un icono, un mensaje, y un área de detalles donde se puede mostrar texto plano.
DialogosPredefinidos	La clase `DialogosPredefinidos` define una serie de métodos estáticos que permiten al usuario abrir diálogos de diversos tipos dentro de la aplicación.

4.3.1. Nombre de Proyecto y Plug-in ID

La comunidad de Eclipse utiliza una convención para los identificadores de los *plug-in's*, así como para los espacios de nombres de los proyectos. Esto se debe a que muchos *plug-ins* terminan siendo las fuentes de otros *plug-ins*, y es necesario que estos sean identificados únivocamente mediante un identificador único global dentro de la comunidad.

Como cada *plug-in* se desarrolla como un proyecto, esta convención se adopta tanto para identificar el *plug-in* como para el nombre del proyecto, es decir, utilizaremos el mismo nombre como identificador y nombre de proyecto.

Otra utilidad que presenta este convenio, es que podemos saber el dominio al que pertenece el *plug-in*, es decir, si tenemos un *plug-in* que se llama *"com.coolimagingproject.coolimaging,"* sabremos que el propietario de este *plug-in* es *www.coolimagingproject.com.* Por lo que debería asegurarse que tiene los derechos del dominio al que hace referencia.

Ambas prácticas son convenciones y no reglas. Por ello, podríamos tener un proyecto llamado *colorOperations* y establecer el identificador del plug-in como *"com.coolimagingproject.colorOperations,"* pero se hace más difícil recordar que proyecto se empareja con que *plug-in*.

Buenas prácticas de programación

Como buena práctica de programación se recomienda a los desarrolladores realizar clases bien definidas. Por eso para los *plug-ins* de operaciones de la aplicación se recomienda realizar una clase *Operador*, la cuál podría utilizarse en otras aplicaciones al no ser dependiente de la aplicación. Y un *OperadorAplicacion*, que haga uso del *Operador* y que si será dependiente de la aplicación. Los ejemplos mostrados a lo largo de este manual hacen uso de esta filosofía de programación.

Identificadores de Parámetros

Algunas operaciones requieren de una serie de parámetros para poder ser ejecutadas, este es el caso del operador rotación, que necesita conocer el ángulo de rotación; el centro de rotación; el método de interpolación usado para corregir los errores que surgen a consecuencia de realizar la rotación; el método de extensión del borde para tratar los píxeles que caen fuera de la imagen, y un campo que indica si se quiere que la imagen sea centrada tras la rotación.

Estos parámetros se pueden tratar como `int` (y no utilizar ningún nombre simbólico), o como **CONSTANTES** como en el `OperadorRotacion.java`.

Recomendación: haga uso de constantes para facilitar la comprensión del código.

OperadorRotacion.java

```
...

/**
```

```
  * Identificador del parámetro que representa el ángulo de rotación, medido
  * en radianes y en sentido antihorario. El objeto asociado es de tipo
  * float.
  */
  public static final int PARAMETRO_ANGULO_ROTACION = 10;
/**
  * Identificador del parámetro que representa la coordenada X del centro de
  * rotación. El objeto asociado es de tipo float.
  */
  public static final int PARAMETRO_CENTRO_ROTACION_X = 11;
/**
  * Identificador del parámetro que representa la coordenada Y del centro de
  * rotación. El objeto asociado es de tipo float.
  */
  public static final int PARAMETRO_CENTRO_ROTACION_Y = 12;
/**
  * Identificador del parámetro que representa el método de interpolación
  * usado para tratar la imagen tras la rotación. El objeto asociado es de
  * tipo Interpolation.
  */
  public static final int PARAMETRO_METODO_INTERPOLACION = 13;
/**
  * Identificador del parámetro que representa el método usado para el
  * tratamiento del borde de la imagen en la rotación. El objeto asociado es
  * de tipo BorderExtender.
  */
  public static final int PARAMETRO_METODO_BORDE = 14;
/**
  * Identificador del parámetro que indica si la imagen, tras ser rotada,
  * debería ser trasladada para quedar centrada (su origen de coordenadas
  * sería (0,0), y no aquél impuesto por la rotación). El objeto asociado es
  * de tipo boolean.
  */
  public static final int PARAMETRO_ROTAR_Y_CENTRAR = 15;

...
```

Los valores de estas constantes, son las que se extraen del `PanelInfoOperador` a través del método `void getParametros(ConjuntoParametroOperador parametros)`, que posteriormente el operador usará para operar.

PanelInfoOperadorRotacion.java

```
public void getParametros(ConjuntoParametroOperador parametros){

  parametros.insertarParametro(new ParametroOperador(
    OperadorRotacion.PARAMETRO_ANGULO_ROTACION, new Float(
     this.textoAnguloRotacion.getText())
     * new Float(Math.PI) / 180.0f));

  parametros.insertarParametro(new ParametroOperador(
     OperadorRotacion.PARAMETRO_CENTRO_ROTACION_X, new Float(
     this.textoCentroRotacionX.getText())));

  parametros.insertarParametro(new ParametroOperador(
     OperadorRotacion.PARAMETRO_CENTRO_ROTACION_Y, new Float(
     this.textoCentroRotacionY.getText())));

  parametros.insertarParametro(new ParametroOperador(
     OperadorRotacion.PARAMETRO_METODO_INTERPOLACION,
     this.listaMetodosInterpolacion.getInterpolacion()));

  parametros.insertarParametro(new ParametroOperador(
     OperadorRotacion.PARAMETRO_METODO_BORDE, this.listaMetodosBorde
     .getTratadorBorde()));
```

```
  parametros.insertarParametro(new ParametroOperador(
     OperadorRotacion.PARAMETRO_ROTAR_Y_CENTRAR,
     this.botonCentrarImagen.getSelection()));
}
```

En el objeto `parametros` se insertan todos los parámetros que se pueden extraer del panel de rotación. Como puede apreciarse, cada parámetro insertado (mediante la función `insertarParametro()`) es un objeto de tipo `ParametroOperador`. Cada parámetro tiene asociado dos elementos: un identificador del parámetro (valor `int`) y un `Object` que representa el valor del parámetro. Por ejemplo, el primer parámetro que se inserta se corresponde con el ángulo de rotación de la imagen. El valor numérico del ángulo de rotación es extraído del campo textual del panel que permite al usuario introducir el ángulo de rotación. Dicho valor numérico se corresponde con el valor del parámetro.

El parámetro, aparte de un valor, tiene asociado un identificador en forma de entero. Así por ejemplo, el primer parámetro insertado en el ejemplo de arriba tiene un identificador cuyo valor es la constante `OperadorRotacion.PARAMETRO_ANGULO_ROTACION`. El identificador del parámetro no es más que un valor entero que el usuario, de forma arbitraria, asigna a cada uno de los parámetros que requiere su operación para ejecutarse. Por ejemplo, el usuario podría asignar el identificador 100 al ángulo de rotación o el identificador 101 al centro de rotación. Es importante que, sin embargo, **no haya parámetros con identificadores REPETIDOS**. Así pues, el usuario no podría asignar tanto al ángulo de rotación como al centro de rotación identificadores ambos de 100. Es también importante notar que **EL USUARIO NO PUEDE HACER USO DE LOS IDENTIFICADORES QUE VAN DEL 0 AL 99**. Dichos identificadores están reservados por la aplicación, y si el usuario hace uso de ellos el resultado de la operación puede ser inesperado.

4.5. Preguntas Frecuentes (FAQ)

¿Puede un mismo *plug-in* contener operaciones de caracterización y tratamiento de imágenes?

La respuesta es **SÍ**, pero el desarrollador debe de tener en cuenta que los índices de estos dos tipos de operaciones deben situarse en diferentes ficheros de índices. Más información en crear un índice para la aplicación.

4.6. Bibliografía

En esta sección queremos ofrecerle a los desarrolladores algunos enlaces y manuales que le pueden ser de utilidad.

Java Advanced Imaging (JAI)

Recomendación: Si el desarrollador desea hacer uso de JAI, este debe instalar la versión de JAI para el JDK.

- Descargar JAI (https://jai.dev.java.net/binary-builds.html)
- Manual con ejemplos de JAI (https://jaistuff.dev.java.net/)
- FAQ de JAI (http://java.sun.com/products/java-media/jai/forDevelopers/jaifaq.html)

Eclipse Rich Client Platform (Eclipse RCP)

- Eclipsepedia: Rich Client Platform (http://wiki.eclipse.org/Rich_Client_Platform)
- Eclipse Rich Client Platform: Designing, Coding, and Packaging Java Applications, by Jeff McAffer, Jean-Michel Lemieux
- Eclipse: Building Commercial-Quality Plug-ins, by Eric Clayberg, Dan Rubel
- Developing Eclipse RCP applications with Eclipse Ganymede (3.4) - Tutorial (http://www.vogella.de/articles/RichClientPlatform/article.html)
- Professional Eclipse 3 for Java Developers, by Berthold Daum

Internacionalización

- How to Internationalize your Eclipse Plug-in (http://www.eclipse.org/articles/Article-Internationalization/how2I18n.html)
- Professional Eclipse 3 for Java Developers, by Berthold Daum - Internationalizing Products (página 401-405)
- Listado de los códigos de países (ISO 3166) (http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html)
- Listado de los códigos de idioma (ISO 639:1988) (http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt)

5. Formatos de imagen soportados

Formatos de Lectura

- ***Bit Mapped Picture*** (**BMP**)
  extensiones: *.bmp*
- ***FlashPiX*** (**FPX**)
  extensiones: *.fpx*
- ***Joint Photographic Experts Group*** (**JPEG**)
  extensiones: *.jpg, .jpeg*.
- ***Graphics Interchange Format*** (**GIF**)
  extensiones: *.gif*
- ***Portable Network Graphics*** (**PNG**)
  extensiones: *.png*
- ***Tagged Image File Format*** (**TIFF**)
  extensiones: *.tiff, .tif*
- ***Portable BitMap*** (**PBM**)
  extensiones: *.pbm*
- ***Portable GrayMap*** (**PGM**)
  extensiones: *.pgm*
- ***Portable PixMap*** (**PPM**)
  extensiones: *.ppm*

Formatos de escritura

- ***Bit Mapped Picture*** (**BMP**)
  extensiones: *.bmp*
- ***Joint Photographic Experts Group*** (**JPEG**)
  extensiones: *.jpg, .jpeg*.
- ***Portable Network Graphics*** (**PNG**)
  extensiones: *.png*
- ***Tagged Image File Format*** (**TIFF**)
  extensiones: *.tiff, .tif*
- ***Portable BitMap*** (**PBM**)
  extensiones: *.pbm*
- ***Portable GrayMap*** (**PGM**)
  extensiones: *.pgm*
- ***Portable PixMap*** (**PPM**)
  extensiones: *.ppm*

6. Conceptos

Región de Interés (ROI)

La *región de interés* (ROI, "Region of Interest" en inglés) es la región asociada a una imagen que consideramos, como su nombre indica, de interés. Esta será la región sobre la que se podrá elegir operar cuando se realicen operaciones de tratamiento, caracterización u otro tipo de operaciones.

Para más información puede ir a la sección Cómo definir una región de Interés (ROI).

Internacionalización

La internacionalización es el proceso de diseñar software de manera tal que pueda adaptarse a diferentes idiomas y regiones sin necesidad de cambios de ingeniería ni de código.

Varios puntos que cubre la internacionalización son los siguientes:

- Varios idiomas disponibles.

- Diferentes convenciones culturales.
- Zonas horarias.
- Formatos de horarios.
- Formatos de fechas.
- Monedas internacionales.
- Sistema de pesos y medidas (pulgadas/centímetros, libras/gramos, etc.).
- Códigos de caracteres (Unicode resuelve fácilmente este problema).
- Formato de números (puntos decimales, separadores de miles, etc.).

Definición extraida de la Wikipedia en español (http://es.wikipedia.org/wiki/Internacionalización)

Si desea *internacionalizar* un *plug-in* puede encontrar algunos manuales en la bibliografía.

7. Preguntas Frecuentes (FAQ)

Esta sección se comenzará cuando tengamos preguntas por parte de los usuarios.

8. Créditos

Diseñadores y Desarrolladores

- **Luis A. González Jaime**
- **Ricardo J. Palma Durán**

Agradecimientos

- **Alba María Carretero García**
  Traductora de Inglés
- **María Civantos Hueso**
  Realización de algunos iconos de la aplicación
- **Everaldo Coelho**
  Por poder hacer uso de gran parte de los iconos utilizados en la aplicación (Crystal Project Icons con licencia LGPL)
- **Joaquín Fernández Valdivia**
  Director del proyecto
- **Raúl Jiménez Ortega**
  Webmaster de Cool Imaging
- **José Francisco Mantas Serrano**
  Realización del logo de la aplicación

Reconocimientos a este proyecto

- **Finalistas en la 1ª edición del Concurso Universitario de Software Libre Granadino (2009)**
- **Finalistas Innovación en la 3ª edición del Concurso Universitario de Software Libre Nacional (2009)**
- **Premio Emilio Herrera de la Universidad de Granada 2009**