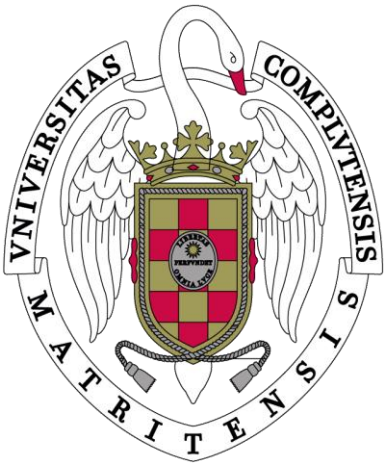
DNEB
Detección de Nuevas Estrellas Binarias

Sistemas Informáticos

2009 – 2010



Facultad de Informática

Universidad Complutense de Madrid

Dirigido por:

Rafael Caballero Roldán

Dpto. Sistemas Informáticos y Programación

Integrantes:

Daniel Godino Muncharaz

Iván Nicolás Peña Tovar

Diego Pérez Urbina

Autorización

Se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria como el código desarrollados en este proyecto.

Daniel Godino Muncharaz

Iván Nicolás Peña Tovar

Diego Pérez Urbina

Resumen. Una estrella binaria es un sistema estelar compuesto de dos estrellas que orbitan mutuamente alrededor de un centro de masas común. Encontrar este tipo de estrellas es para los astrónomos una tarea muy importante, ya que les provee el mejor método para determinar la masa de éstas, además de la temperatura y el radio. Con estos datos, los astrónomos son capaces de calcular la masa de otras estrellas no binarias.

Se propone una herramienta para la ayuda en la detección de estrellas binarias a partir de placas fotográficas POSSI y POSSII obtenidas por el observatorio de Monte Palomar. Estos dos conjuntos de imágenes fueron tomados con 40 años de diferencia y la comparación entre placas de la misma región permite apreciar el movimiento de las estrellas en ese periodo de tiempo, pudiendo así detectar binarias no conocidas hasta el momento.

Así mismo, y debido al rápido movimiento de este tipo de estrellas, se espera también encontrar las nuevas coordenadas de las estrellas ya conocidas, de forma que la información disponible se mantenga actualizada.

Palabras clave: astronomia, estrella, binaria, catálogo, búsqueda, aplicación, dneb, investigación

Abstract. A binary star is a star system composed of two stars orbiting each other around a common mass center. Finding this kind of stars is for astronomers a very important task, since they provide the best method to determine the mass of them, besides the temperature and radius. With this data, astronomers are able to calculate other no-binary stars mass.

A tool is proposed to help in binary stars detection from POSI and POSII photographic plates obtained by Monte Palomar. These two image sets were taken 40 years apart and the comparation between same region plates allows to observe stars movement in that time period, being able to detect unkown binaries so far.

Thus, due to the fast movement of this kind of stars, is also expected to find new coordinates of stars already known, so that available information is kept updated.

Keywords: astronomy, star, binary, catalog, searching, application, nbsd, investigation

Índice de contenido

Introducción

Objetivos

DNEB es un proyecto software de código libre desarrollado en el ámbito de la astronomía con el objetivo de ayudar a los observadores (ya sean astrónomos o aficionados a la astronomía) en sus tareas, especialmente en aquellas relacionadas con las estrellas binarias. Las posibilidades que ofrece van desde un conversor de coordenadas (entre grados decimales y grados sexagesimales), calculo de distancias entre dos coordenadas dadas, descarga de la imagen de unas coordenadas concretas o visor de imágenes en formato FITS hasta descarga de múltiples imágenes de una sola vez correspondientes a una parcela de la esfera celeste, búsqueda de nuevas estrellas dobles o localizar estrellas dobles conocidas y que se han movido desde la última vez que fueron observadas.

Mientras que muchas de estas funcionalidades son simplemente de apoyo y ya se encuentran disponibles en otras aplicaciones, lo novedoso de DNEB y por lo que nació este proyecto es todo aquello relacionado con las estrellas binarias.

Nuestro objetivo es claro, agilizar la localización tanto de nuevas estrellas binarias como de aquellas conocidas y que se han movido.

Para la búsqueda de nuevas estrellas binarias, DNEB automatiza la tarea de buscar estrellas que se han movido, ofreciéndole al usuario al finalizar una lista de posibles estrellas dobles. Esta información no siempre será fiable y es necesario que el usuario con conocimientos sobre la materia la compruebe. Recalcamos esto para que el lector sea consciente de que DNEB no va a encontrar estrellas binarias, sino candidatas a ser estrellas binarias, lo que ahorrará muchísimas horas de trabajo al astrónomo al evitarse este el tener que consultar fotografías de una en una.

Para la búsqueda de estrellas dobles que se han movido lo que se proporciona es una herramienta que permite consultar estrellas dobles mediante el uso un catálogo de estrellas dobles en el cual se posee todo tipo de información relacionada con las mismas. Mediante ésta búsqueda podemos crear tareas que descargan imágenes del lugar donde las estrellas fueron vistas por última vez, una vez descargadas, mediante el procesamiento de cálculo de posición el algoritmo se encargará de localizar las estrellas y dar su posición actual, ésta herramienta es especialmente interesante para buscar estrellas que hace mucho tiempo que no son observadas y que por lo tanto tienen mayor probabilidad de haber cambiado de posición desde la última observación.

Por tanto, con DNEB se ha querido conseguir una aplicación completa, orientada a la investigación pero que no se olvida de las tareas sencillas y cotidianas de los astrónomos, ganando así facilidad de uso y eficiencia al estar todo integrado en la misma herramienta.

Organización del documento

El documento presente está orientado a guiar y dirigir los procesos de analisis, diseño, implementación y pruebas del sistema propuesto. Para ello, se comenzará con una

descripción de los fundamentos teóricos que se deben conocer para entender y realizar la aplicación.

Seguidamente, se realizará un análisis de los requisitos que deberá cumplir el proyecto, y se determinarán los riesgos que se han tenido en cuenta para que la aplicación salga adelante sin problemas.

A continuación, se pasará al diseño de la aplicación, con una descripción de la arquitectura, de las tecnologías que se van a usar y de los algoritmos a realizar para que ésta tenga la funcionalidad que se describe en el proceso de análisis.

Posteriormente se pasará al desarrollo del proyecto especificado en los apartados anteriores y en el cuál se detallará la arquitectura y las diferentes tecnologías utilizadas para el desarrollo.

Finalmente las pruebas en las cuales se determina que la aplicación funciona correctamente y que cumple los objetivos fijados. En éste apartado se exponen los resultados obtenidos mediante el uso de la aplicación tanto para la detección de nuevas estrellas binarias como para determinar la posición de estrellas binarias que no habían sido observadas desde hace tiempo y que han cambiado de posición.

Fundamentos técnicos

En este apartado se explican los conceptos necesarios para el correcto entendimiento del desarrollo del proyecto, estos conceptos se complementan con la información del glosario.

Sistemas de coordenadas

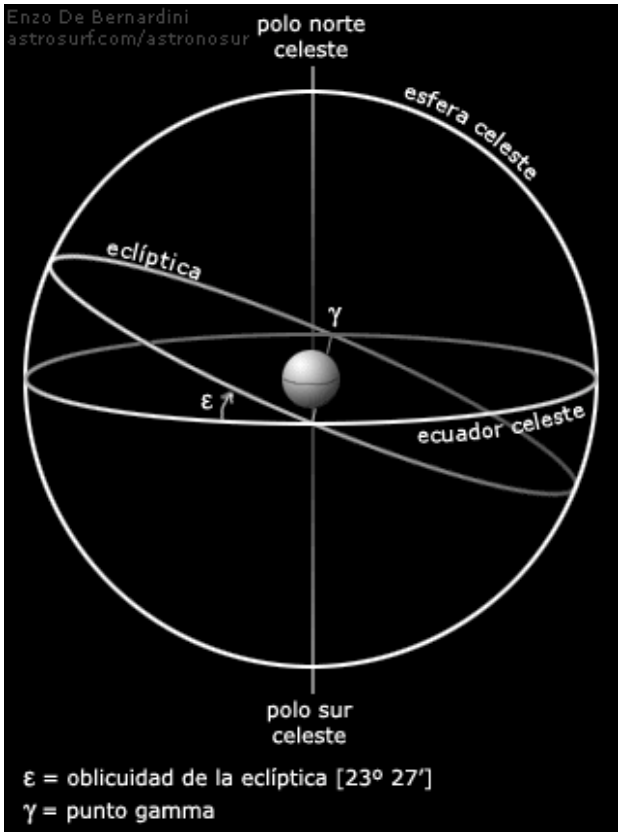
Las coordenadas celestes son el conjunto de valores que, de acuerdo con un determinado sistema de referencia ,dan la posición de un objeto en la esfera celeste. La esfera celeste es una esfera ideal, sin radio definido, concéntrica en el globo terrestre, en la cual aparentemente se mueven los astros.

Los planos fundamentales de la esfera celeste, en los que se basan los diferentes sistemas de coordenadas son los siguientes:

● El ecuador celeste el cuál es proyección del ecuador terrestre.

● Los polos norte y sur celeste, proyección del eje de rotación de la Tierra hacia el norte y sur.

● La eclíptica, la trayectoria media del centro de gravedad de la Tierra, que forma un plano (es la línea por la cual se mueve el Sol a lo largo del año, y los planetas en las cercanías de ella).



Existen los siguientes criterios para clasificar los sistemas de coordenadas celestes.

- Según el sistema de coordenadas
- Según la posición del observador
- Según el plano de referencia

Coordenadas Ecuatoriales

En nuestro caso nos vamos a centrar en las coordenadas ecuatoriales, es un sistema de coordenadas basado en plano de referencia.

- Las referencias fundamentales son el ecuador celeste y el equinoccio vernal.
- Origen: geocéntrico
- Coordenadas: ascensión recta y declinación

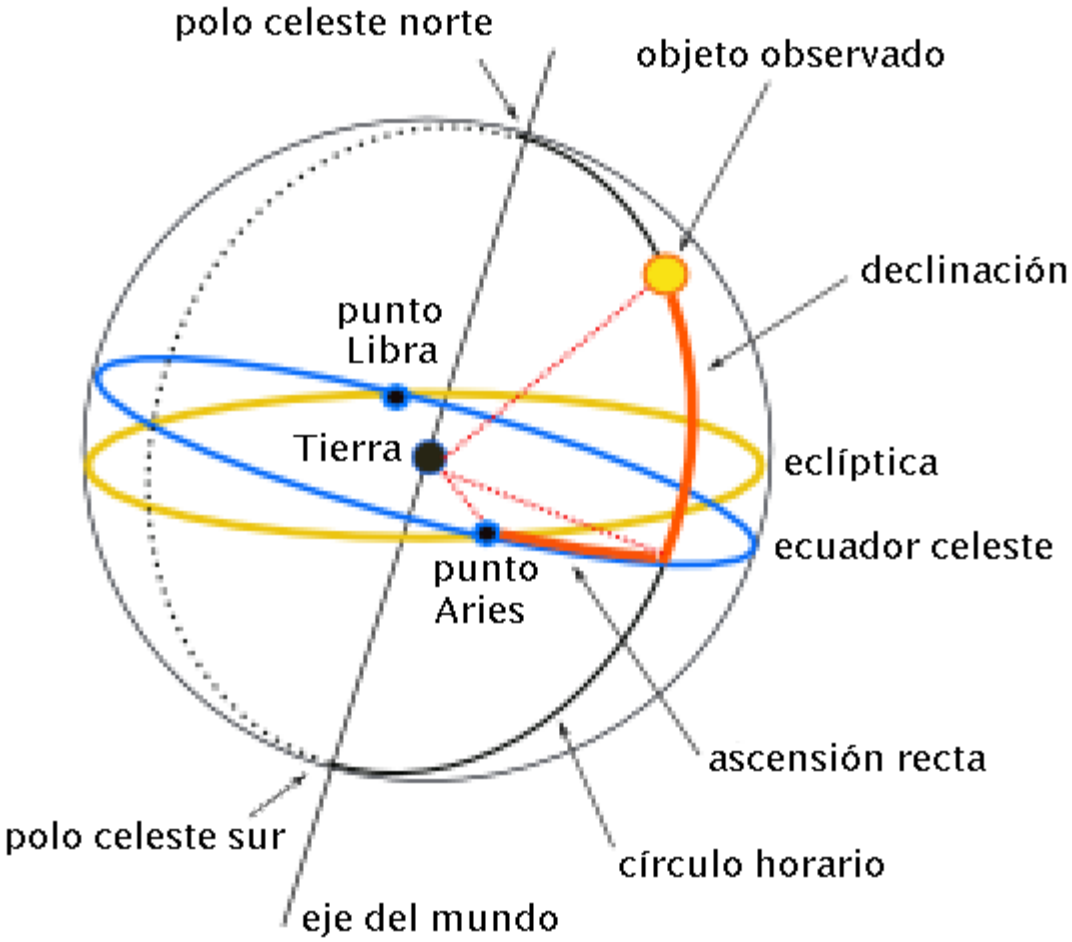
¿Qué es el equinoccio vernal?

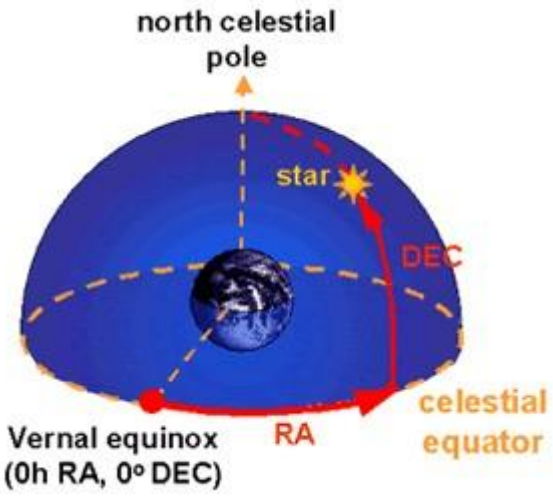
Los planos del ecuador celeste y la eclíptica se cortan en una recta que señala la dirección del punto Aries o punto vernal. Es el punto en el que el Sol pasa del hemisferio sur al norte, cosa que ocurre hacia el 21 de marzo (iniciándose la primavera en el hemisferio norte y el otoño en el hemisferio sur).

En las coordenadas ecuatoriales:

- La ascensión recta se mide a partir del Punto Vernal en horas (una hora igual a 15 grados), minutos y segundos hacia el Este a lo largo del ecuador celeste.

● La declinación es el ángulo que forma el astro con el ecuador celeste. La Declinación es comparable a la latitud geográfica, la diferencia es que ésta se mide sobre el ecuador terrestre. Se mide en grados y es positiva si está al norte del ecuador celeste y negativa si está al sur.





Estas coordenadas no están ligadas a un punto de observación en particular ya que el equinoccio vernal y el ecuador celeste no varían, se esté dónde se esté.

Otros fundamentos

Digitized Sky Survey (o Survey de ahora en adelante) es una versión digital de varios atlas fotográficos del cielo nocturno, y un proyecto en curso para producir más versiones digitales de conjuntos de datos astronómicos fotográficos.

Observatorio Monte Palomar, es un observatorio astronómico de propiedad privada localizado en San Diego, California, a 140 km al sudeste del Observatorio Monte Wilson. Posición: 33°21'22.80"N, 116°51'53.14"W; altitud: 1706 m. Es propiedad y está gestionado por el Instituto Tecnológico de California.

POSSI Palomar Observatory Sky Survey, es un catálogo de imáges obtenidas en los años 50 en el observatorio de Monte Palomar.

POSSII Palomar Observatory Sky Survey II, es un catálogo de imáges obtenidas en los años 80 en el observatorio de Monte Palomar.

Washington Double Star Catalog El Catálogo de Estrellas Dobles de Washington (WDS), mantenido por la Observatorio Naval de los Estados Unidos, es la principal base de datos mundial de información astronómica de estrellas dobles. El catálogo WDS contiene posiciones (J2000), información del descubridor, ángulos, separaciones, magnitudes, movimientos de 103,681 sistemas binarios.

FITS o **Flexible Image Transport System** es el formato estándar de datos astronómicos confirmado por la NASA y por la IAU (International Astronomical Union).

FITS es mucho más que un formato de imagen (como JPG o GIF) y está diseñado principalmente para almacenar datos científicos constituidos por una matriz multidimensional y por tablas bidimensionales que contienen filas y columnas de datos. Como en cada fichero se pueden guardar varias de estas tablas o arrays, un solo fichero puede trasportar gran cantidad de información, siendo posible incluso la representación de varias imágenes aunque esto no es lo usual.

Además, cada fichero contiene unas cabeceras en formato ASCII, siendo así fácilmente legibles. Estas cabeceras están compuestas por pares atributo-valor y contienen información sobre el fichero y su origen, como pueden ser el tamaño, el observatorio desde el que se tomó la fotografía, formato binario de los datos, comentarios, historia de los datos y cualquier otra información que el creador desee.

Planificación y organización

Organización del proyecto.

Para la organización del proyecto tomamos la decisión de realizar el desarrollo de manera conjunta y horizontal, coordinando la actividad mediante un grupo creado en Google Groups y mediante reuniones periódicas para tomar decisiones y realizar desarrollos críticos. La coordinación con el tutor se realiza de manera periódica pero no fija dependiendo de las necesidades cada momento.

Planificación del proyecto con fases y objetivos.

A continuación se detallan las diferentes fases del proyecto, dentro de cada una de ellas se detalla la documentación generada, los prototipos desarrollados, y las pruebas realizadas. Toda esa documentación generada se encuentra contenido en ésta memoria a lo largo de sus diferentes apartados.

Comienzo (01/07/2009 - 13/10/2009)

En esta fase se estudia el alcance del proyecto de manera global, estudiado las posibilidades del mismo. Para ello se realizan una serie de reuniones con el tutor en las que se acuerdan los objetivos, también se analizan los riesgos del mismo y las medidas para mitigar los mismos. Como consecuencia de los acuerdos y de su análisis se genera el documento de objetivos y el documento de gestión de riesgos que son consensuados con el tutor del proyecto.

En esta fase también se genera documentación formativa para el desarrollo del proyecto, toda esta documentación puede ser en parte consultada en el Glosario para los conceptos más sencillos y en el de fundamentos técnicos para aquellos conceptos más avanzados.

En esta fase se desarrollan dos prototipos para el estudiar al viabilidad de la descarga de imágenes de los cuáles uno es descartado.

Elaboración (14/10/2009 - 20/02/2010)

En la fase de elaboración se realizan dos desarrollos paralelos.

Por un lado se define la arquitectura del proyecto y se comienza a realizar prototipos del modelo de datos y de los motores de ejecución. Se comienza a estudiar también el funcionamiento que tendrá la interfaz.

Por otra parte se comienza a desarrollar algunos de los algoritmos definidos en el proceso de análisis, entre estos algoritmos el de cálculo del centroide, el de detección de estrellas y el de división del cielo en regiones.

Se genera un documento de requisitos precisos.

Se generan prototipos de los algoritmos planificados y una interfaz que permite el manejo de los mismos.

Se desarrollan pruebas con JUnit de los algoritmos para validar los resultados.

Construcción(21/02/2010 - 02/05/2010)

En la fase de construcción se termina el desarrollo de la totalidad de los algoritmos. Además se desarrollan los procesamientos de búsqueda de estrellas dobles y de cálculo de posición de estrellas dobles , también se realiza un refinamiento de la interfaz obteniendo su versión definitiva.

Se desarrollan herramientas propias para poder probar los algoritmos y que además permiten visualizar los resultados que obtienen en caso de no quererlos en modo texto.

Se generan los siguientes prototipos: versión de la aplicación con algoritmos funcionales pero no ajustados.

Transición (03/05/2010 - 06/06/2010)

En la fase de transición se comienza a dar uso a la aplicación para la obtención de resultados en los diferentes ámbitos de investigación del proyecto. Para ello se ejecutan importantes procesamientos cuyos resultados son analizados de manera exhaustiva.

En ésta fase se genera documentación de pruebas y de resultados.

Se genera la versión final de la aplicación.

Análisis

En este apartado se realizará una descripción detallada de los requisitos tanto funcionales como no funcionales del sistema, así como los riesgos que se pueden encontrar en la realización del proyecto.

Requisitos del sistema

Antes de empezar a enumerar y describir los distintos requisitos del sistema, se hará una breve descripción de los distintos conceptos que se van a usar en el apartado:

Requisitos funcionales

El sistema a desarrollar deberá cumplir los requisitos indicados a continuación:

RF01 - Crear tareas

Una tarea es un conjunto de imágenes que cubre una parcela de cielo. Las imágenes provendrán de dos *surveys* escogidos por el usuario, de forma que las fotografías estarán emparejadas.

Se debe permitir al usuario crear tareas introduciendo las coordenadas iniciales y finales de la porción de cielo deseado, así como los tipos de *surveys* a utilizar, el solapamiento entre imágenes consecutivas, el ancho y alto de la imagen y la ruta donde almacenar las imágenes.

RF02 - Crear descargas

De forma particular, llamamos descarga a una tarea de tan solo dos imágenes, una por cada *survey*. Esto es útil cuando se quiere la imagen de unas coordenadas concretas.

Se permitirá escoger los surveys desde donde se obtienen las imágenes, así como las coordenadas y el tamaño de estas.

RF03 - Crear configuración de descarga

Se puede dar el caso de que el usuario quiera crear varias descargas, y que éstas tengan varios parámetros de configuración iguales. Es por ello que se le ofrecerá la opción de crear tipos de configuraciones para evitar el tener que introducir varias veces los mismos datos.

RF04 - Gestionar tareas

El sistema permitirá que el usuario pueda eliminar, parar o reanudar las distintas tareas creadas anteriormente, así como visualizar la información de configuración de éstas y su progreso.

RF05 - Crear procesamientos

Un procesamiento es el tipo de operación que se ha de aplicar sobre todas las imágenes que forman una tarea. Existen dos tipos de procesamientos, procesamientos para la búsqueda de estrellas dobles y procesamientos para la localización de una estrella binaria ya conocida pero que puede haber variado su posición.

Siempre que haya alguna tarea creada, el usuario deberá poder asociar algún tipo de procesamiento a ésta. Para ello se le mostrarán todas las tareas en una tabla con toda su información y se le dará a elegir entre los dos tipos de procesamientos.

RF06 - Gestionar procesamientos

El sistema, al igual que con las tareas, deberá permitir al usuario eliminar, parar, reanudar y visualizar los procesamientos creados.

RF07 - Visor de imágenes

El usuario tendrá a su disposición tres tipos distintos de visores de imágenes:

- En el primero el usuario podrá visualizar el resultado del algoritmo de búsqueda de estrellas ejecutado sobre dos imágenes. Para ello deberá introducir un valor de brillo y otro de umbral. Como añadido dispondrá de la opción de escalar dichas imágenes para poder ver detalles más cerca.

- En el segundo el usuario podrá comprobar el resultado tanto del algoritmo de búsqueda de estrellas en movimiento como el de búsqueda de estrellas dobles sobre dos imágenes. El añadido de este visor es la opción de crear una animación con ambas imágenes, así se podrá observar más detalladamente las diferencias entre ambas.

- En el tercer visor el usuario podrá ver el resultado que produce la ejecución del algoritmo de cálculo de distancias sobre una imagen dada. Para complementar la funcionalidad se permite, también en este visor, el escalado de la imagen y el cálculo de la distancia en coordenadas celestes entre dos puntos seleccionados en la imagen por el usuario.

RF08 - Importar catálogo de estrellas dobles

Se podrá importar a la base de datos el catálogo de estrellas dobles de washington (Washington Double Star Catalog o WDS) en el formato de texto estándar definido en el siguiente documento: http://ad.usno.navy.mil/wds/Webtextfiles/wdsweb_format.txt

RF09 - Consultar catálogo de estrellas dobles

Se permitirá realizar consultas rápidas sobre el catálogo de estrellas dobles previamente importado, pudiendo usar filtros para todos aquellos valores numéricos o fechas. Una vez obtenido los valores de la consulta, la herramienta debe ser capaz de poder generar una tarea de descarga de imágenes con los resultados obtenidos en la búsqueda realizada, pudiéndose además poder elegir los surveys de los que se descargarán las imágenes.

RF10 - Conversor de coordenadas celestes

Se dispondrá de un sencillo conversor de coordenadas para pasar del sistema sexagesimal al sistema decimal y viceversa.

Es una sencilla funcionalidad pero bastante útil para disponer de las coordenadas celestes en el mismo formato, ya que usualmente la ascensión recta se representa sexagesimalmente mientras que la declinación se expresa en formato decimal

RF11 - Cálculo de distancias

La aplicación debe de poseer una herramienta que permita calcular la distancia entre dos puntos dentro del sistema de coordenadas determinado en la especificación de la aplicación. Para ellos la herramienta debe de tener como entrada dos puntos, especificándose la

ascensión recta y declinación de cada uno de ellos, como salida la herramienta devolverá la distancia en arco segundos y el ángulo en grados.

RF12 - Cálculo del histograma de una imagen

Entre los añadidos del sistema, el usuario también tendrá a su disposición un apartado donde podrá visualizar el histograma de una imagen cargada por él mismo.

RF13 - Exportar resultados a XML

Una vez creado y ejecutado un procesamiento, se permitirá exportar los resultados a un fichero en formato XML para mejorar el entendimiento por parte del usuario.

RF14 - Importar datos desde base de datos

Debe ser capaz de generar tareas de descarga de imágenes en base a unas tablas de referencia en las que se puede introducir al menos información sobre coordenadas de las imágenes que se desea descargar.

Requisitos no funcionales

Aquí se enumeran todos los requisitos no especificados en el apartado anterior.

● Facilidad de uso:

- Existencia de una interfaz sencilla e intuitiva.
- Existencia de un manual de usuario que describe el funcionamiento del sistema.
- Métodos de ayuda en la propia aplicación para mejorar el entendimiento de ciertos temas.

● Soporte:

- Facilidad de instalación por parte del usuario.
- Se trata de un sistema multiplataforma.

Casos de uso

Previamente a definir los casos de uso elegidos, se hará una descripción de los distintos actores que van a interactuar con el sistema:

● Usuario

El usuario es el actor que interactúa con el sistema a través de su interfaz. Puede hacer uso de cualquiera de las funcionalidades que éste le ofrece.

● Servidor de imágenes

El servidor de imágenes es un actor externo al sistema. Es el encargado de suministrarle las imágenes cuando éste las solicita.

Casos de uso elegidos

CU01 - Arranque del sistema

● Descripción del escenario

El usuario arranca el sistema.

● Actores

Usuario.

● Condiciones iniciales

Ninguna.

● Condiciones finales

Inicio del sistema.

● Secuencia de interacciones entre los actores y el sistema

1. El usuario arranca el sistema tal y como se explica en el apartado del manua l de usuario.
2. El sistema se inicia y realiza un chequeo de consistencias de los datos almacenados. Esto es que comprueba que todas las tareas y procesamientos que aparecen como activos se marquen como inactivos.
3. El sistema crea un hilo de ejecución para cada tarea existente.

El sistema se queda en espera de recibir órdenes del usuario.

● Secuencia de interacciones alternativas

No hay.

CU02 - Obtención de imágenes

● Descripción del escenario

El usuario del sistema desea crear una tarea para obtener imágenes de una porción del cielo.

● Actores

Usuario y servidor de imágenes.

● Condiciones iniciales

Inicio del sistema.

● Condiciones finales

Creación de una tarea y obtención de un conjunto de imágenes asociado que representa un trozo de cielo.

● Secuencia de interacciones entre los actores y el sistema

Crear una descarga

1. Para comenzar con la configuración el usuario selecciona el submenú "Crear una descarga" del apartado "Descargas".
2. El usuario elige el tipo de *survey* para cada una de las imágenes.
3. Acto seguido introduce los valores para la ascensión recta y la declinación del trozo concreto de cielo que quiere descargar.
4. Especifica, en arcominutos, el ancho y el alto de cada una de las imágenes.
5. Elige el formato de la imagen.
6. Por último, selecciona la ruta donde descargar las imágenes.
7. Una vez configurada, el usuario selecciona el botón "Crear descarga". Una vez hecho esto la descarga, junto con sus valores, se almacenará en el sistema.
8. Ya creada, el usuario debe ir al submenú "Gestor de tareas" del menú "Tareas".
9. Una vez allí, selecciona la descarga en la tabla de tareas y selecciona el botón "Reanudar".
10. La descarga se inicia y el sistema hace la petición al servidor de imágenes de que quiere obtener las imágenes correspondientes.
11. El servidor de imágenes le proporciona al sistema todas las imágenes solicitadas.

Crear una tarea

1. En este caso, para configurar la tarea el usuario debe seleccionar el submenú "Crear una nueva tarea" del apartado "Tareas".
2. El usuario elige los dos tipos de *surveys* para cada par de imágenes.
3. Después, para especificar la porción de cielo a obtener, introduce la ascensión y declinación inicial y la final.
4. Al igual que antes, el usuario especifica, también en arcominutos, el ancho y el alto de las imágenes.
5. Como novedad, el usuario elige el solapamiento entre imágenes consecutivas.
6. Elige la ruta donde almacenar las imágenes.
7. Y, por último, selecciona el botón "Crear tarea".
8. Los pasos desde aquí son los mismos que en la creación de una descarga.

● Secuencia de interacciones alternativas

1. En ambos modos habrán campos obligatorios que el usuario deberá rellenar, y otros opcionales que no serán necesarios. Si los obligatorios no se rellenan el sistema deberá informar de ello con un mensaje de error.
2. El usuario puede cancelar la operación en cualquier momento.
3. Se deberá dar la opción al usuario de poder eliminar y parar las tareas.

CU03 - Búsqueda de estrellas dobles

● Descripción del escenario

El usuario, habiendo creado anteriormente al menos una tarea, quiere analizar las imágenes asociadas a dicha tarea en busca de estrellas dobles.

● Actores

Usuario.

● Condiciones iniciales

El usuario debe de haber creado antes una tarea o una descarga.

● Condiciones finales

El sistema almacena todas las estrellas encontradas.

● Secuencia de interacciones entre los actores y el sistema

1. El usuario debe seleccionar el submenú "Crear procesamiento" del apartado "Utilidades".
2. De entre todas las tareas que aparezcan en la tabla, el usuario selecciona la que más le interese, y elige la opción "Procesamiento busqueda dobles".
3. Acto seguido selecciona el botón para verificar la creación del procesamiento.
4. Una vez creado, el usuario debe elegir la opción "Monitor de procesamientos" del mismo apartado "Utilidades".
5. Selecciona el procesamiento que ha creado y presiona el botón "Reanudar".
6. El sistema inicia el procesamiento.
7. El sistema almacena el resultado para que pueda ser revisado por el usuario.

● Secuencia de interacciones alternativas

1. El sistema debe avisar de que el procesamiento se ha creado correctamente.
2. Se deberá dar al usuario la opción de eliminar y parar los procesamientos.

CU04 - Cálculo de posiciones

● Descripción del escenario

El usuario, teniendo al menos una tarea almacenada en el sistema, desea realizar una búsqueda de estrellas dobles que se hayan movido en las imágenes asociadas a una de esas tareas.

● Actores

Usuario.

● Condiciones iniciales

El usuario debe de haber creado antes una tarea o una descarga.

● Condiciones finales

El sistema almacena las estrellas dobles encontradas que se hayan movido.

● Secuencia de interacciones entre los actores y el sistema

1. El usuario debe seleccionar el submenú "Crear procesamiento" del apartado "Utilidades".
2. De entre todas las tareas que aparezcan en la tabla, el usuario selecciona la que más le interese, elige la opción "Procesamiento calculo distancia" y rellena los valores para los parámetros umbral, brillo y máximos candidatos.
3. Seguidamente selecciona el botón para verificar la creación del procesamiento.
4. Los pasos desde aquí son los mismos que en la búsqueda de estrellas dobles.

● Secuencia de interacciones alternativas

1. El sistema debe avisar de que el procesamiento se ha creado correctamente.
2. Se deberá dar al usuario la opción de eliminar y parar los procesamientos.

Riesgos

El objetivo de éste apartado es informar de los riesgos que se han detectado en el proyecto, de su evaluación, y de las medidas que se han tomado para evitarlos o, en su defecto, minimizar su daño.

Escala de posibilidad de aparición del riesgo: Se valora de 1 a 4 siendo 1 alto alto riesgo y 4 bajo riesgo.

Escala de impacto por la aparición del riesgo: Se valora de 1 a 4 siendo 1 alto impacto y 4 bajo impacto.

Imposibilidad de descarga de imágenes.

Descripción: Viabilidad para descargar de manera automática un gran volumen de imágenes de diferentes surveys, especialmente de los surveys POSSI y POSSII. Ésto podría tener como consecuencia el aborto del proyecto y la necesidad de buscar una nueva alternativa o un nuevo enfoque del proyecto.

Posibilidad de aparición: 3

Impacto: 1

Medidas para evitar el riesgo: Se realizó un profundo estudio acerca de las posibles fuentes a usar para acceder a los surveys.

Medidas para reducir su impacto: Se decidió abordar ésta parte del proyecto en el verano y así evitar perder tiempo durante el curso en caso de que la opción no fuera viable.

Tiempo demasiado elevado para el procesamiento de las imágenes.

Descripción: los algoritmos tardan tiempo en ejecutarse, y normalmente para obtener resultados necesitamos procesar un número muy elevado de imágenes, dado que el plazo para entregar el proyecto es acotado es importante tener un tiempo razonable para pode procesar un número suficiente de imágenes y poder analizar sus resultados antes del fin del plazo del proyecto.

Posibilidad de aparición: 2

Impacto: 3

Medidas para evitar el riesgo: No se toma ninguna medida.

Medidas para reducir su impacto: Se intenta adelantar el desarrollo para poder tener más tiempo para hacer pruebas con las que obtener resultados, de esta forma no afectaría que el algoritmo tarde puesto que tendríamos tiempo de sobra para probarlo.

Imposibilidad de rotar las imágenes adecuadamente.

Descripción: Teniendo dos fotografías centradas en las mismas coordenadas celestes, cada una procedente de un survey distinto, puede darse el caso de que una de ellas se encuentre ligeramente rotada, complicando el encaje de las dos imágenes. Se hace por tanto necesario rotar alguna de las imágenes para que las dos tengan la misma orientación. Sin embargo, el algoritmo de giro tiene un margen de error.

Posibilidad de aparición: 1

Impacto: 3

Medidas para evitar el riesgo: Se plantean diferentes alternativas para la realización del cálculo, desde el simple uso de matrices afines de rotación hasta la utilización de librerías que hacen uso de avanzadas técnicas de interpolación.

Medidas para reducir su impacto: Se estudia el impacto y se decide que en caso de que aparezca se descartarían las imágenes giradas y se estudiarían solo las imágenes que no estén giradas.

Desconocimiento de fundamentos para el procesamiento de las imágenes

Descripción: Se hace necesaria la localización de las estrellas en imágenes en blanco y negro y ningún integrante disponemos de conocimientos en el campo de tratamiento de imágenes digitales.

Posibilidad de aparición: 1

Impacto: 1

Medidas para evitar el riesgo: El riesgo es inevitable, es necesario localizar las estrellas.

Medidas para reducir su impacto: Se estudian distintas técnicas de eliminación de ruido y de detección de bordes. Se llega a la conclusión de que son técnicas demasiado costosas para aplicarlas a cientos de imágenes. Se desarrolla un algoritmo propio que se explicará en el apartado correspondiente.

Problemas por falta de precisión de los algoritmos.

Descripción: Problema que sucede debido a la acumulación de errores de los algoritmos.

Posibilidad de aparición: 2

Impacto: 2

Medidas para evitar el riesgo: Se toman todos los datos numéricos decimales como double para aumentar la precisión.

Medidas para reducir su impacto: No se toman medidas en este sentido.

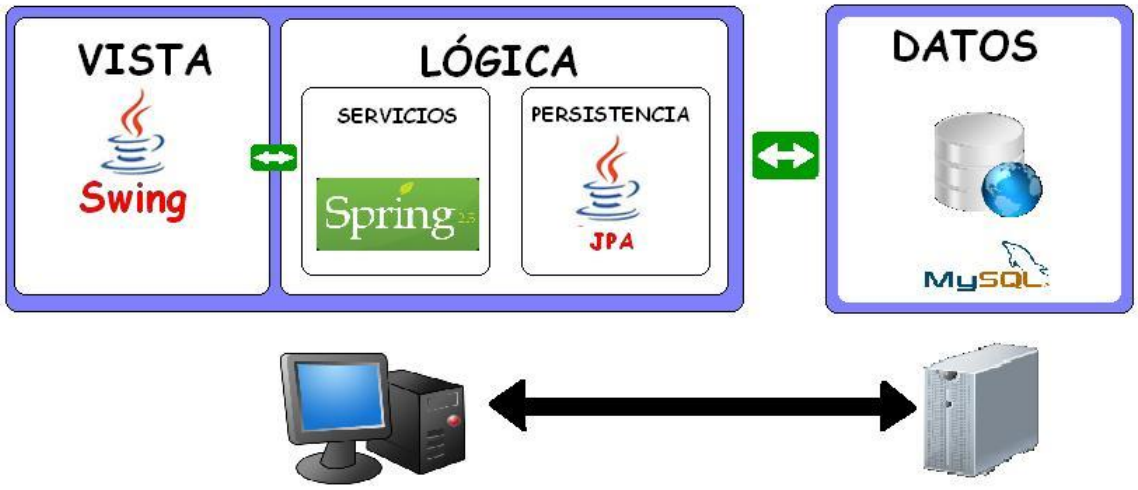
Gestión de la configuración

Para la gestión de la configuración hemos decido apoyarnos en diferentes herramientas. En concreto para el control de versiones del código hemos usado Subversion alojando el proyecto en Google Code. Para cada versión probada de la herramienta hemos realizado un nuevo tag en la herramienta subversión.

Para la gestión documental hemos utilizado la aplicación Google Docs. Además hemos distribuido los documentos finales mediante el apartado de descargas de Google Code.

Diseño

Arquitectura



La aplicación consta de una arquitectura de 2 capas, una es la capa de cliente y la otra es la capa de almacenamiento de datos. La capa de cliente consta a su vez de dos capas claramente diferenciadas, una es la vista y la otra es la lógica. A continuación se procede a detallar cada uno de los niveles de la aplicación.

Vista:

Para el desarrollo de la interfaz gráfica se ha tomado la decisión de utilizar Swing por ser una tecnología conocida por todos los miembros del grupo. Además dado que la idea es que cada investigador tenga su propia aplicación en la que almacene de forma local todos sus datos descartando de ésta forma el desarrollo de una aplicación web. Aunque dado el alto grado de independencia que existe entre la vista y la lógica de la aplicación es posible cambiarla y optar por otra tecnología en caso de ser necesario.

En nuestro desarrollo swing de la aplicación hemos decidido orientarla a pestañas, por ello consta de un menú superior donde se encuentran todas las funcionalidades de la aplicación y una barra de puestañas inferior en la cuál se abren cada una de las opciones seleccionadas en el menu. Esto permite poder tener abiertas múltiples pestañas de las aplicación permitiendo así poder trabajar de manera ágil.

Lógica:

Para el desarrollo de la lógica de la aplicación hemos decidido hacer el desarrollo apoyándonos en el uso del framework spring, el cual nos permite el uso de programación orientada a aspectos para la gestión de las transacciones y el uso de inyección de dependencias para suministrar objetos a las clases.

La aplicación posee una clase encargada de contener el contexto de la aplicación y garantizar que éste es único de forma que cada vez que sea necesario obtener un servicio se haga mediante el mismo contexto.

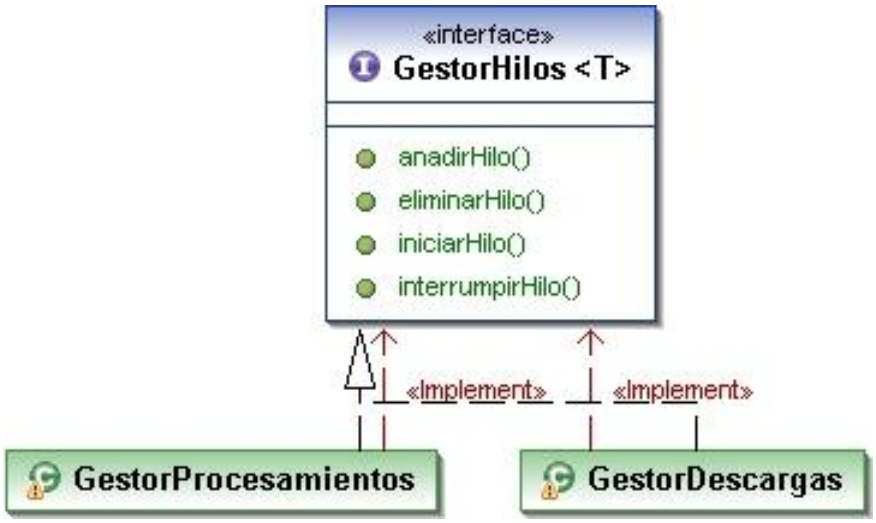
Toda la lógica de la aplicación está contenida dentro de servicios Spring, se han usado anotaciones en todo momento para la definición de los mismos y también para la configuración de las transacciones.

La descarga de imágenes.

La descarga de imágenes se realiza mediante el protocolo HTTP, para facilitar la comunicación con éste protocolo usamos la librería de Apache HTTP Client. Las imágenes se obtienen mediante una petición Post a la url **http://archive.stsci.edu/cgi-bin/dss_search** con una serie de parámetros como son las coordenadas, el alto y el ancho o el Survey entre otras.

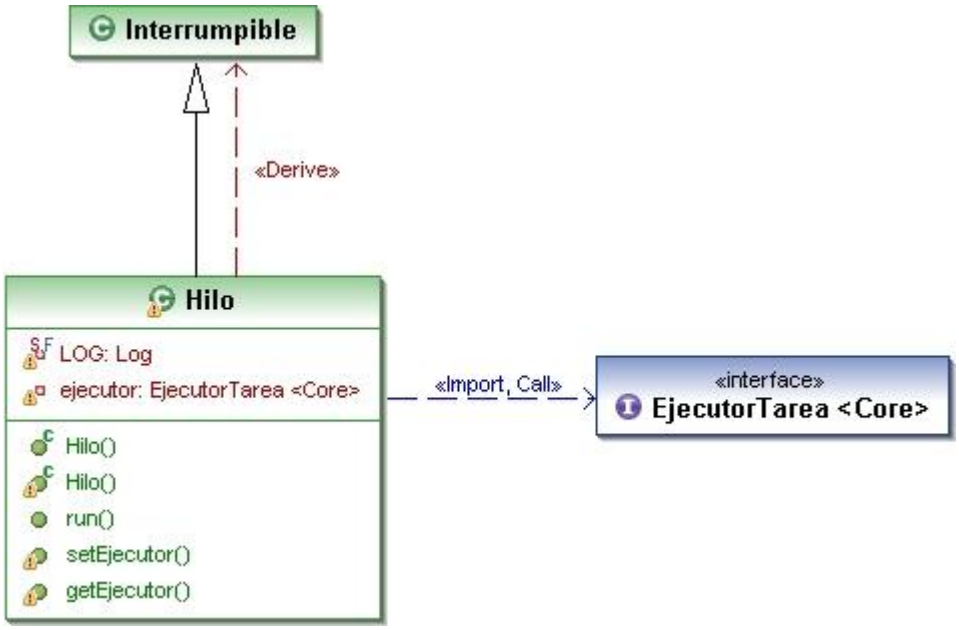
Los motores de Ejecución.

Son uno de los elementos principales del núcleo de la aplicación. Son los encargados de gestionar las tareas de descarga de imágenes y de la ejecución de los procesamientos. La aplicación consta de una interfaz que define las operaciones que realizarán los motores de ejecución. Define operaciones de gestión de hilos, parada, reanudación, eliminación y creación. Ésto permite ampliar fácilmente la aplicación ya que se pueden desarrollar nuevo motores para futuras necesidades que pueda tener la misma.



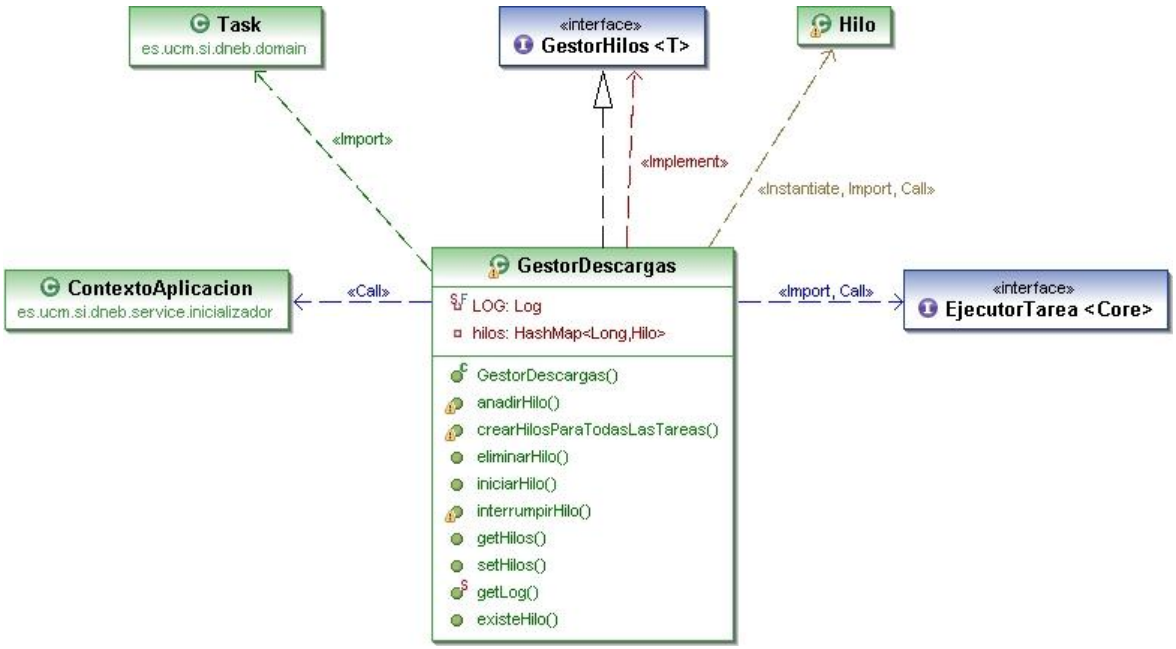
Los hilos de ejecución

Son hilos que extienden de la clase Interrumpible que a su vez extiende de la clase Thread de Java. La clase interrumpible proporciona métodos para poder detener y comprobar si ha sido detenido. La clase Hilo tiene la característica muy importante, contiene un Ejecutor de Tarea(Interfaz) y un método run que se encarga de llamar al método ejecutar del EjecutorTarea. El EjecutorTarea es el elemento encargado de ejecutar el algoritmo mediante una inyección de un servicio de Spring. Actualmente la aplicación tiene dos implementaciones de ejecutores de tarea, el de tareas de procesamiento y el de tareas de descarga. Sin embargo si quisiéramos tener más procesamiento sería tan sencillo como añadir nuevas clases que implementen EjecutorTarea.



El gestor de descargas

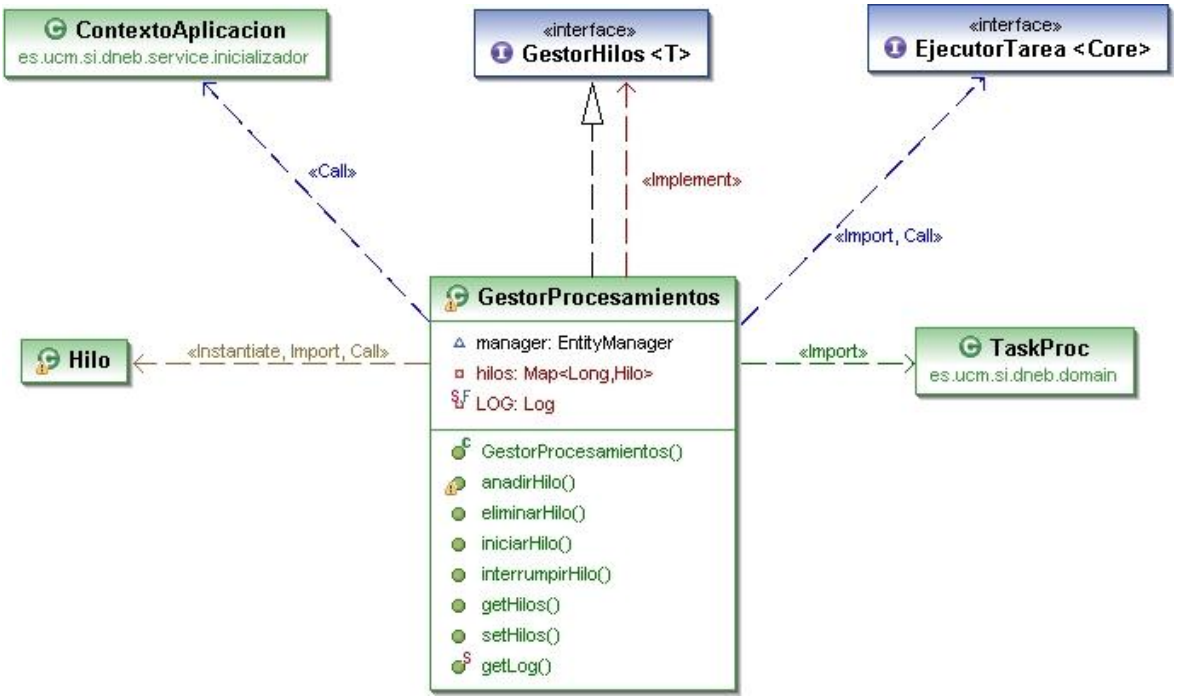
Es el motor de ejecución encargado de gestionar los hilos de descarga de imágenes procedentes de los surveys. A través de este gestor se controlan todas las tareas de descargas que tiene la aplicación y permite su ejecución simultánea. Los hilos de éste gestor contienen la implementación EjecutorDecarga del EjecutorTarea.



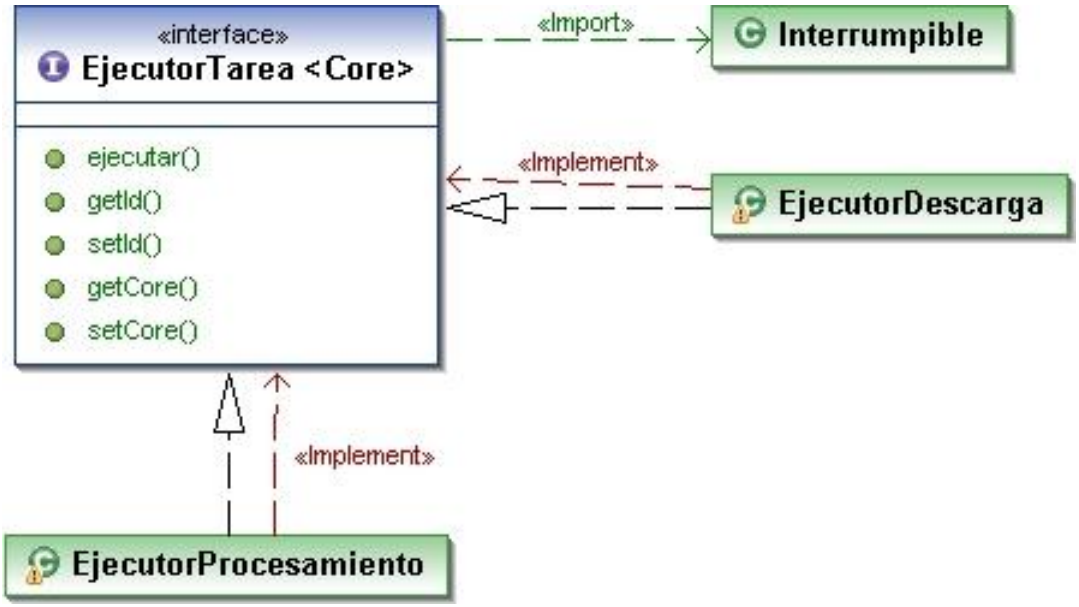
El gestor de procesamientos

Es el motor encargado de gestionar los diferentes procesamientos que ejecutan los diferentes algoritmos. A través de este gestor de se controlan todos los procesamientos que

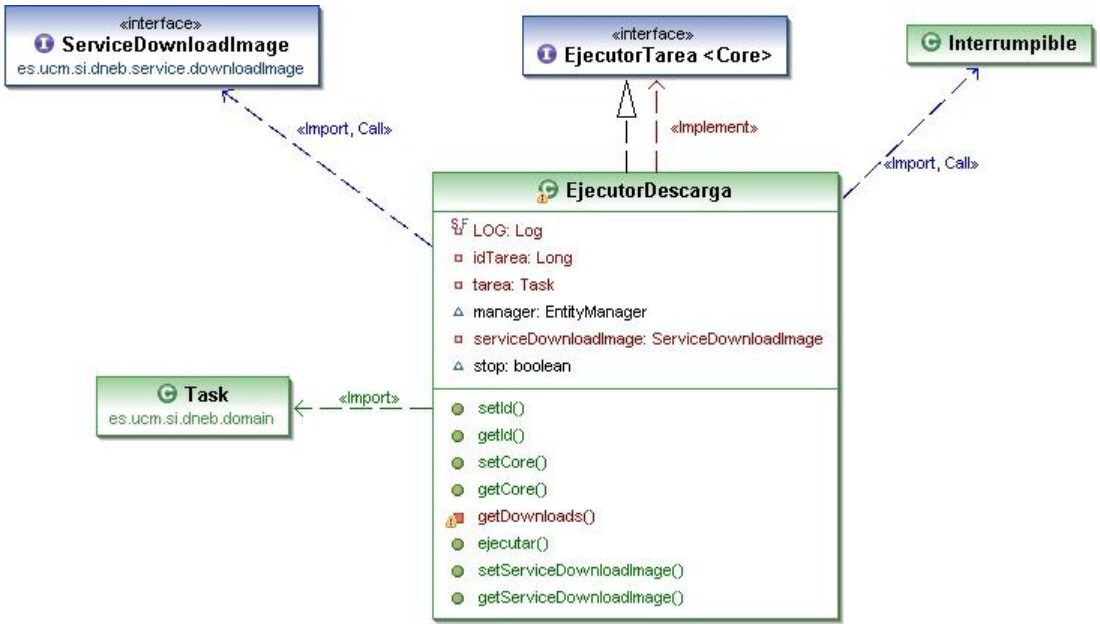
tiene la aplicación y permite su ejecución simultánea. Los hilos de éste gestor contienen la implementación EjecutorProcesamiento del EjecutorTarea.

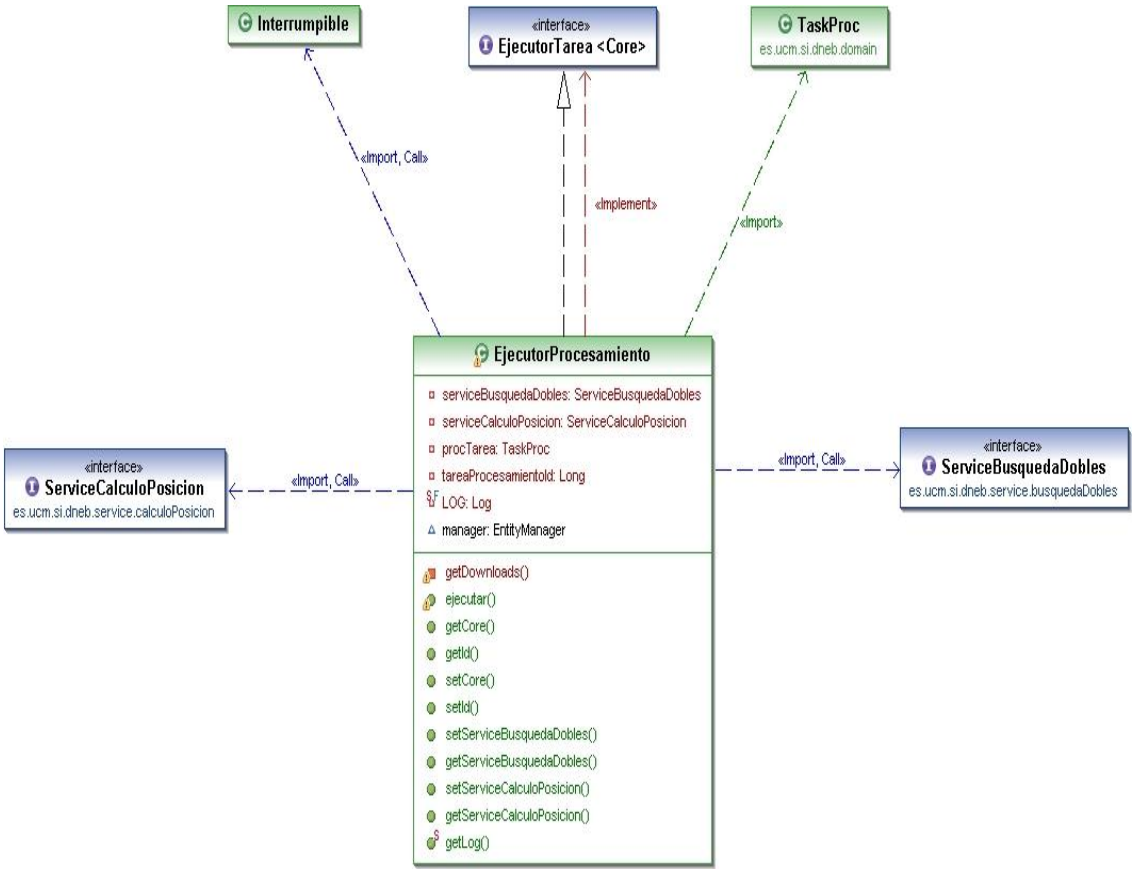


Cada Ejecutor de Tarea es un servicio Spring que realiza transacciones sobre la base de datos para marcar la información procesada, anotar cuando se realiza cada operación, y en el caso de los ejecutores de procesamientos, anotar la información relevante que proceda. Dentro de éstos ejecutores se llama al algoritmo correspondiente, ésto aporta una gran versatilidad ya que podemos cambiar el algoritmo que se ejecuta con sólo cambiar la invocación dentro del método ejecutar del ejecutor. Otra posibilidad de cambiar el algoritmo es mediante el uso de la inyección de dependencias de spring, para ello tenemos que cambiar el nombre del bean que define el ejecutor de tarea que se ejecuta para cada procesamiento dentro de los hilos.



Los ejecutores de tarea tienen dos implementaciones como anteriormente se mencionaba, el EjecutorDescarga y el EjecutorProcesamiento.





Algoritmos, son servicios de Spring y que a su vez pueden tener inyección de otros algoritmos, tienen gestión transaccional ya que en mucho casos realizan escrituras en base de datos.

Persistencia de datos. Las persistencia de datos se ha realizado mediante JPA(Java Persistence API) utilizando como implementación Hibernate . Para la mejora del rendimiento se han usado querys precompiladas.

Almacenamiento de datos

El almacenamiento de datos se realiza en una base de datos relacional. En concreto MySQL 5.1 aunque la aplicación es compatible con otros motores de base de datos relacionales si se realizan unos pequeños cambios en la configuración de la misma. Los datos relevantes además pueden ser exportados a XML mediante el uso de la librería JAXB incluida en la especificación JEE5.

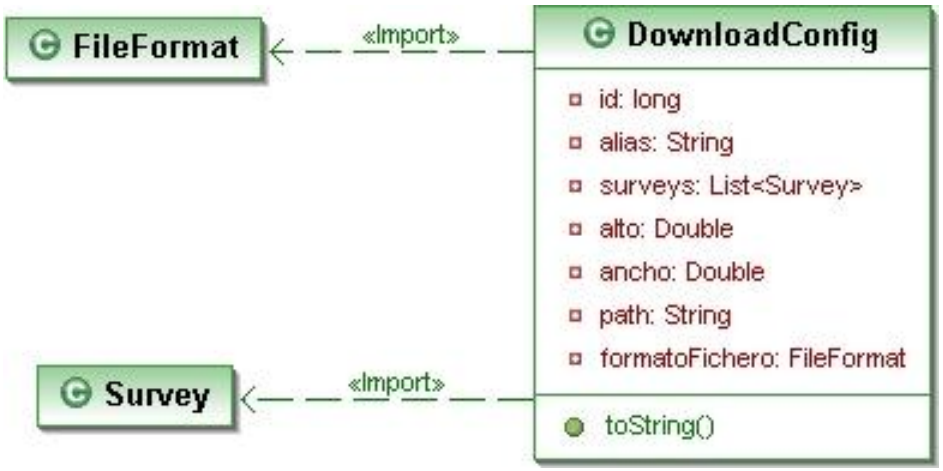
El modelo de datos

A continuación se procede a explicar el modelo de datos de la aplicación, explicando la funcionalidad de cada una de las entidades utilizando como apoyo para las mismas diagramas UML.

DoubleStarCatalog: es la entidad persistitente que almacena los datos del catálogo de estrellas dobles, contiene toda la información de la especificación del WDS. Cuando importamos un catálogo del WDS con la herramienta de importación de catálogo, ésta

queda almacenada en ésta table y es consultada mediante la herramienta de consulta del catálogo de estrellas dobles.

DownloadConfig: es la entidad persistible en la que se almacenan la configuraciones de descarga que la aplicación permite guardar para hacer más ágil la configuración de descarga de imágenes. Ésta se relaciona con la entidad survey ya que una configuración de descarga contiene uno o más Surveys, también se relaciona con FileFormat que representa el formato de fichero de la descarga y que también es guardado.



FileFormat: El formato de Fichero es la entidad en la que se almacenan los formatos de fichero disponibles para descargar imágenes, todas las tareas llevan asociado un único formato de fichero en el que se descargarán las imágenes.

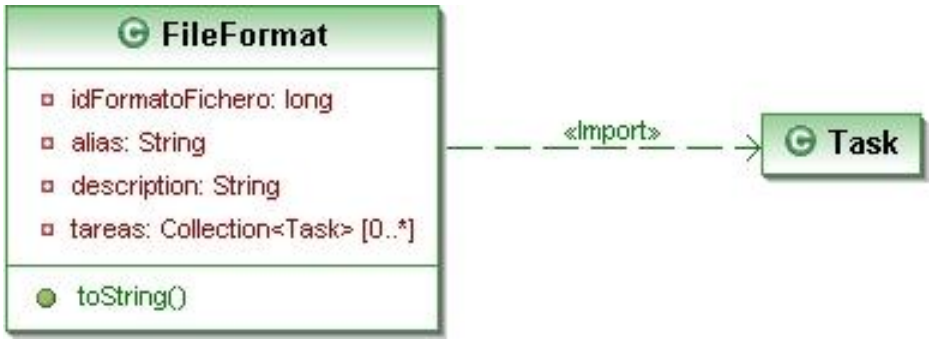
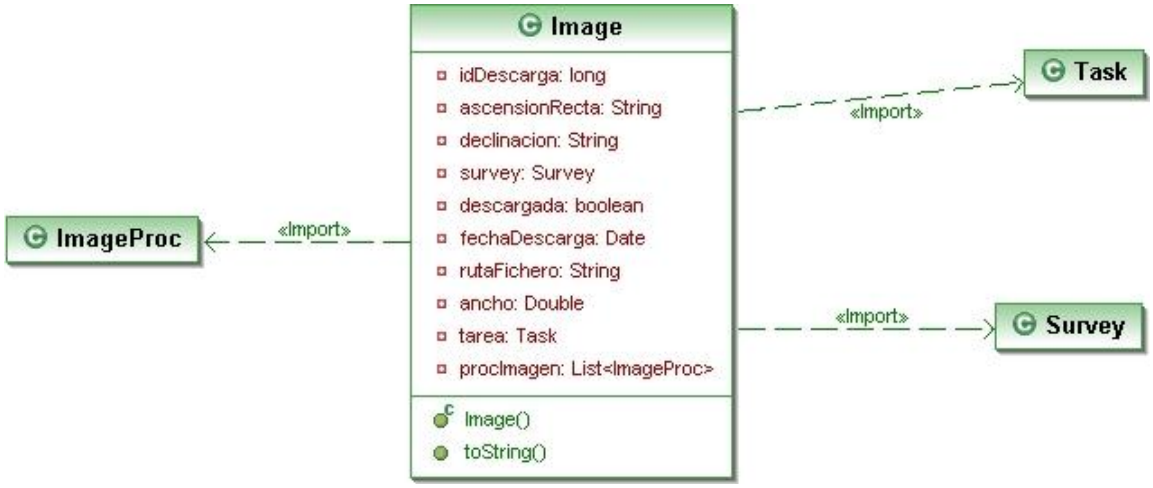
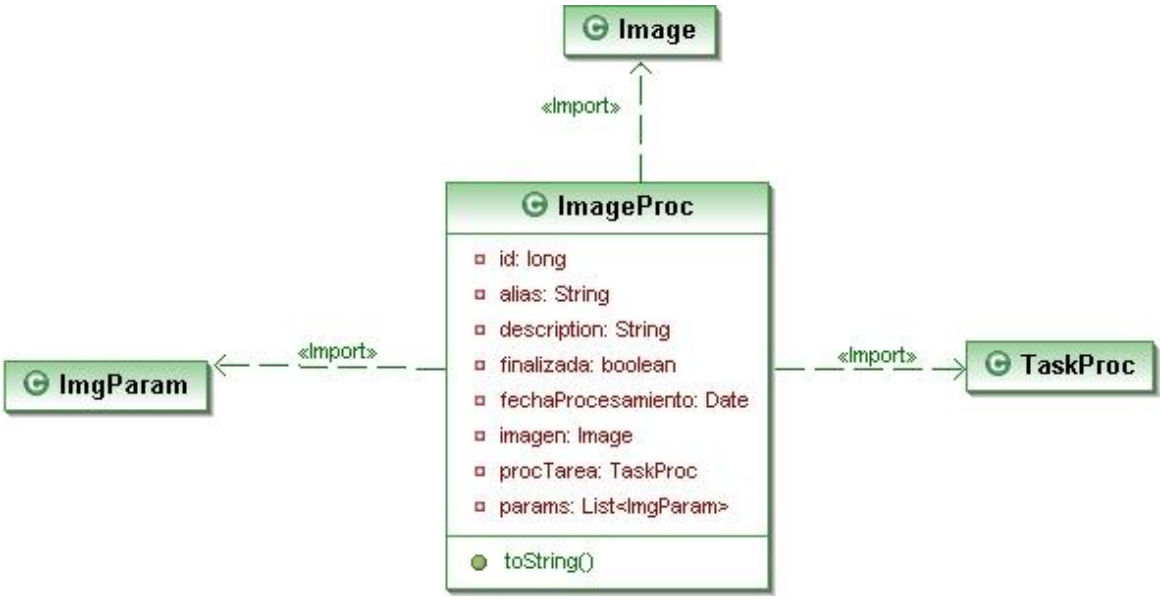


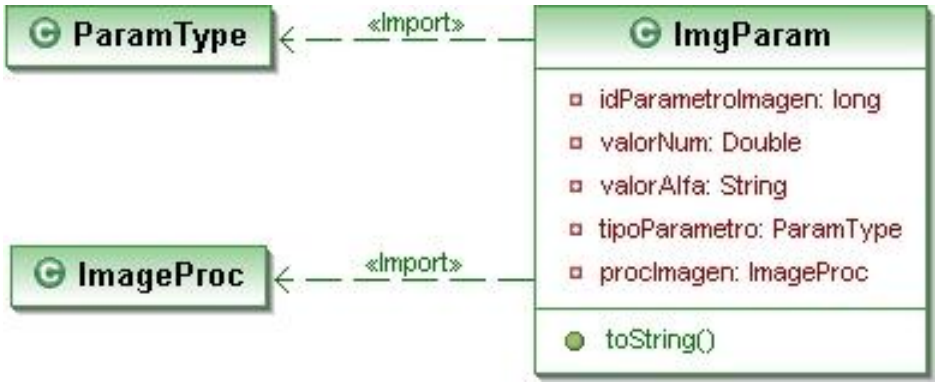
Image: es la entidad que representa las imágenes tanto descargadas como pendientes de descargar, contiene información sobre su tamaño, la región del cielo que representa y dónde se encuentra el fichero físico almacenado. Una imagen siempre está asociada a una tarea que es la que se encarga de gestionar su descarga, la imagen además contiene un campos descargada que determina si la imagen ha sido descargada ya o por el contrario está pendiente de descarga.Una imagen puede tener asociada varios procesamientos de imagen que se encargarán de aplicar algoritmos para la obtención de datos acerca de estrellas binarias que puedan existir en la misma. Además una imagen tiene un único Survey y que es uno de los N surveys que una tarea puede tener.



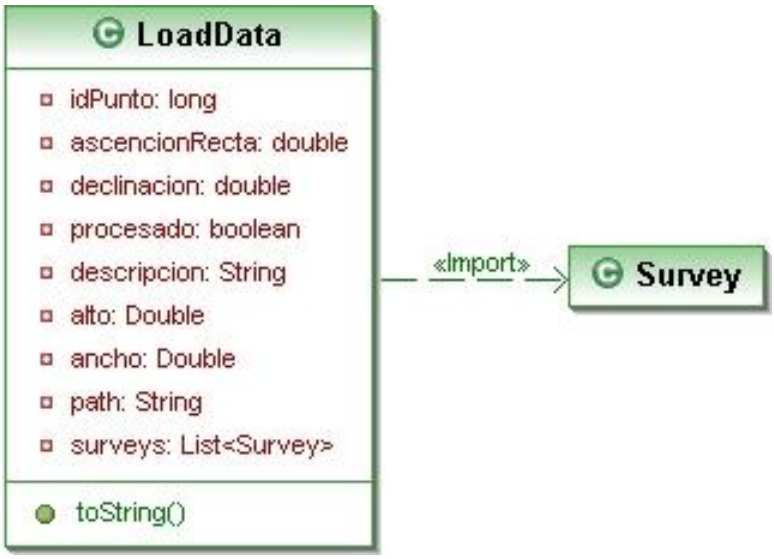
ImageProces: es la entidad que representa el Procesamiento de imagenes, en ella se almacena la imagen a procesar y los parámetros de procesamiento específicos para esa imagen que serán usados por los algoritmos de procesamiento. Una imagen de procesamiento está asociada a una tarea de procesamiento que contiene información sobre el tipo de procesamiento y la gestión de la misma. Además está asociado a la imagen que será procesada.



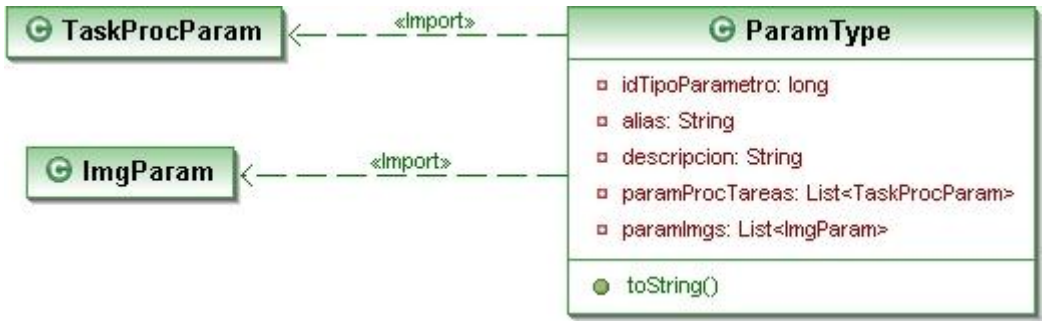
ImgParam: es la entidad que representa a los parámetros asociados a una imagen, éstos parametros son usados por los algoritmos para el procesamiento de imágenes, por ellos los parámetros están asociados a un procesamiento, el parámetro imagen tiene asociado un tipo de parámetro que contiene lo que el parámetro representa, ya que el parámetro sólo contiene el valor y el procesamiento de imagen al que está asociado. El parámetro sólo puede contener un valor, en caso de tener un valor numérico el valor alfanumérico será nulo y viceversa.



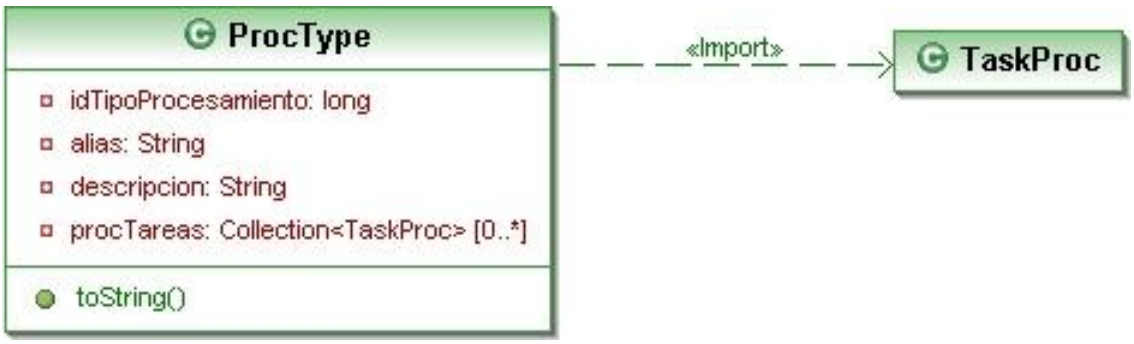
LoadData: es la entidad que permite que se generen tareas mediante scripts lanzados sobre la base de datos con información acerca de cada una de las imágenes que deseamos que sean descargadas. Ésta entidad tiene asociada el survey desde el cuál se descargará la imagen que se crearán en base a ésta entidad.



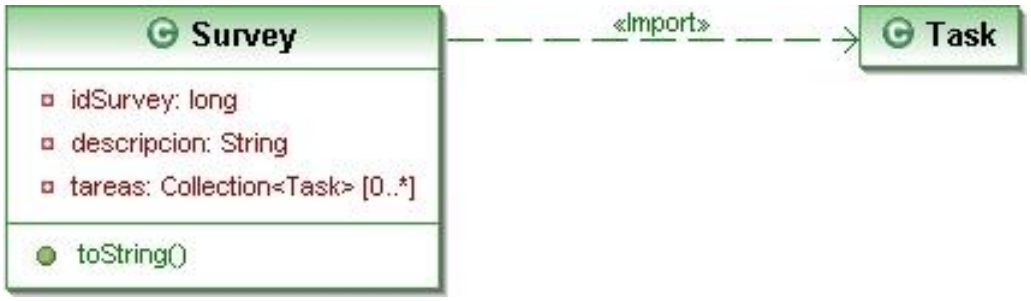
ParamType: ésta entidad repesenta los tipos de parámetros, un tipo de parámetro contiene un id que lo repesenta junto con un alias y una descripción, los valores de esta entidad son creados al instalar la aplicación. Los parámetros de tareas de procesamiento y los parámetros de imágenes tienen asociado un tipo de parámetro al que dan un valor específico, ya sea numérico o alfanumérico.



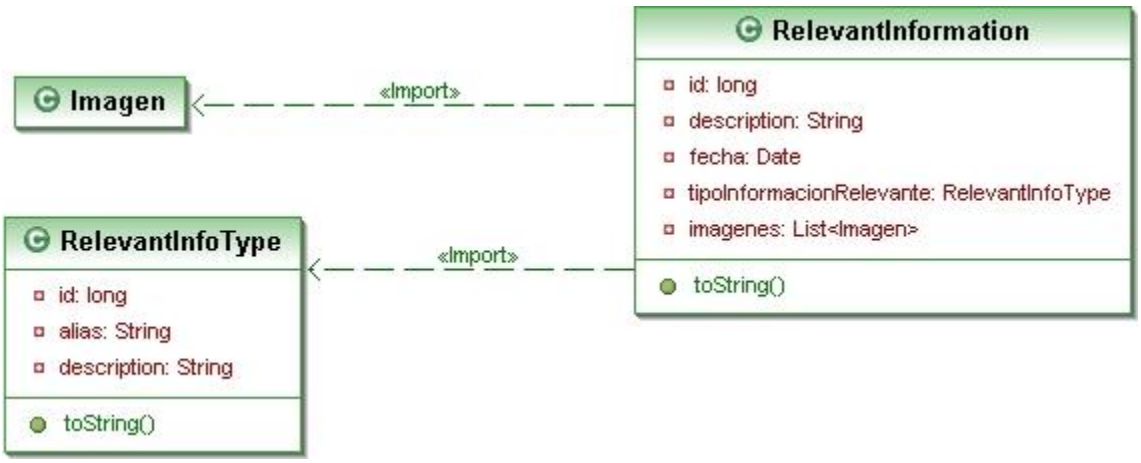
ProcType: es la entidad que representa los tipos de procesamiento, se relaciona con las Tareas de procesamiento(TaskProcess) que contienen un Tipo de Procesamiento, procType.



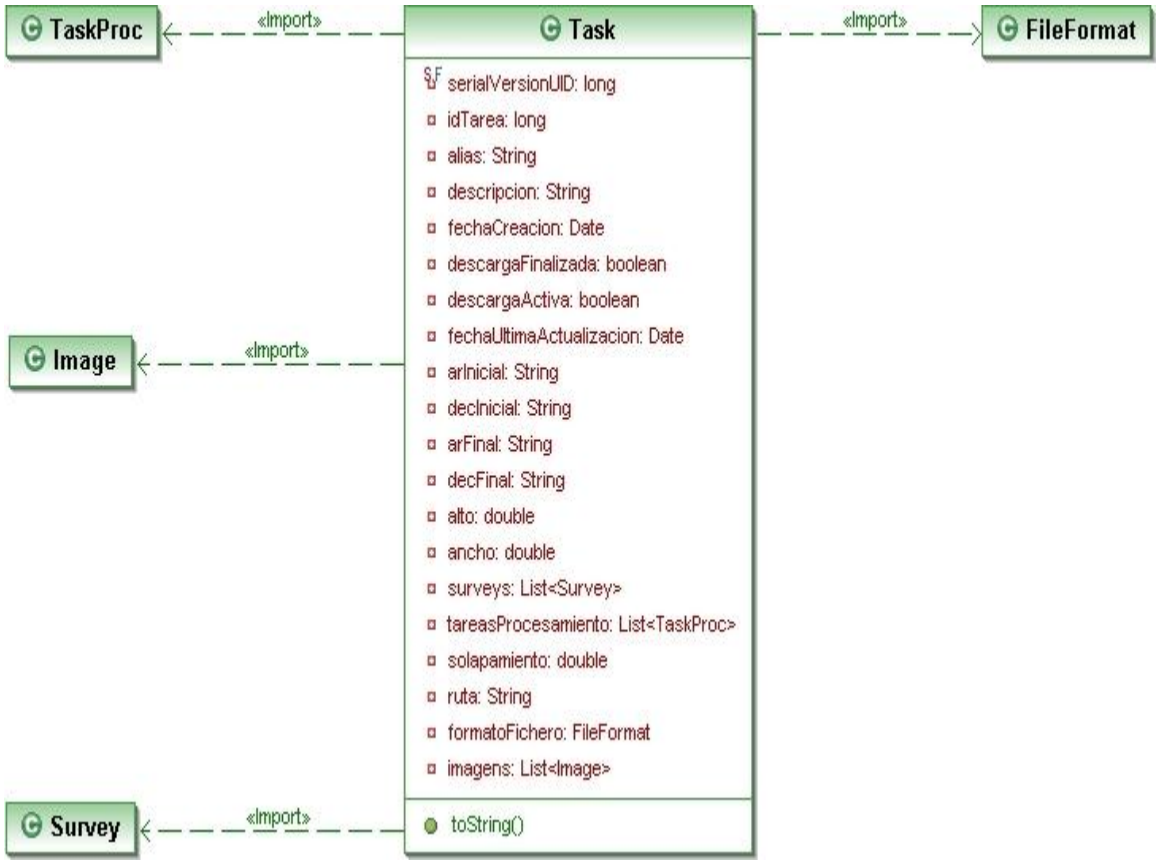
Survey: es la entidad que representa a los surveys de los cuáles se obtienen las imágenes.



RelevantInformation y RelevantInformationType: Son las entidades en las que se almacena la información relevante obtenida por los procesamientos. El tipo de información relevante permite clasificar por tipos la información relevante. La información relevante tiene un tipo de información relevante, posee un alias y una descripción en la que se almacenan todos los datos significativos sobre esa información.

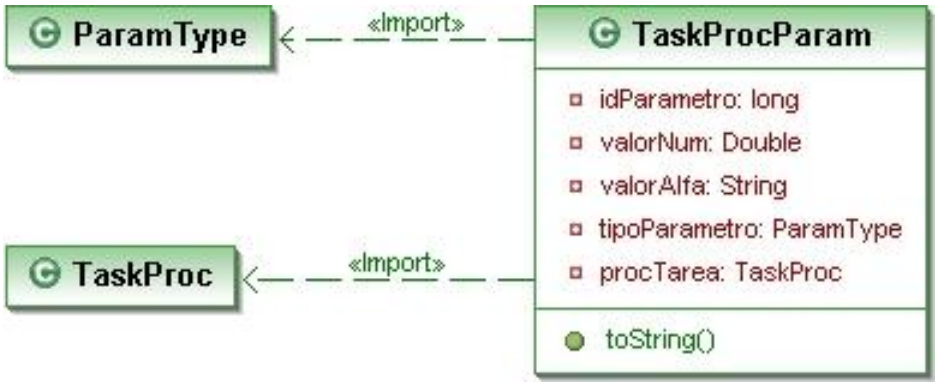


Task: La tarea es la entidad que representa un conjunto de imágenes ya sean de una región continua o de una serie de imágenes que no están relacionadas. La tarea contiene un conjuntos de imágenes y un conjuntos de Surveys. Además posee un formato único de fichero en el que se descargarán las imágenes. Una tarea puede tener asociada múltiples procesamientos.



TaskProcess: El procesamiento de tarea es la entidad que representa el procesamiento sobre una tarea ya descargada. Contiene un conjuntos de procesamientos de imágenes que está asociados a las imágenes de la tarea. Además posee una serie de parámetros asociados al procesamiento que serán usados en la ejecución de los algoritmos. Posee un tipo de procesamiento que determinará el algoritmo que se encargará de hacer el procesamiento sobre las imágenes.

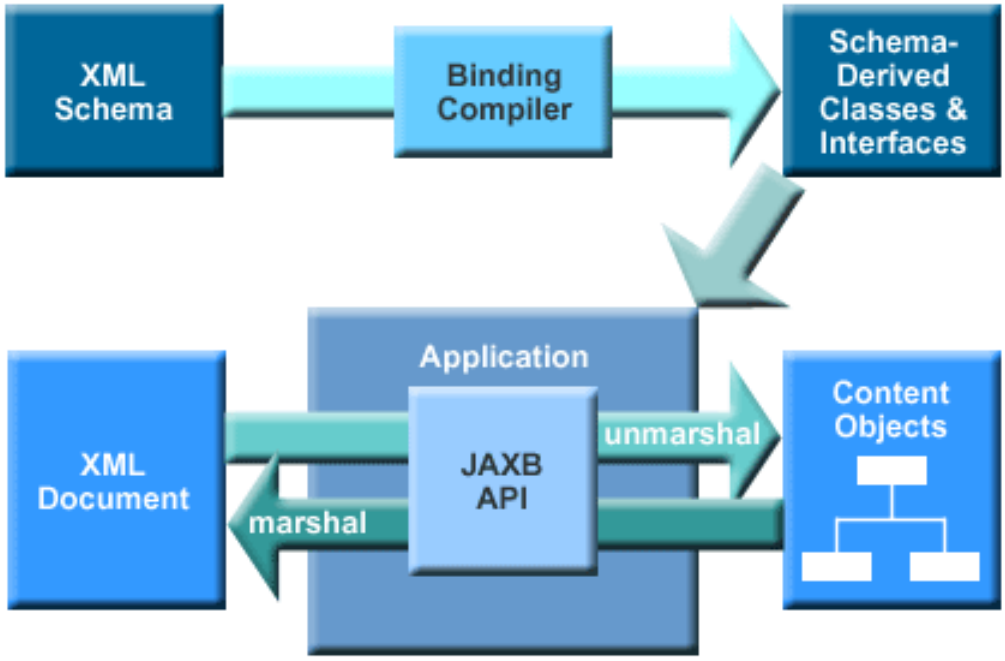
TaskProcParam: representa la entidad encargada de almacenar los parámetros asociados a los procesamientos.



Tecnologías

Java Advanced Imaging (JAI) es una API para Java que proporciona un conjunto de interfaces orientadas a objetos que soportan un modelo de programación de alto nivel, el cual permite a los desarrolladores crear sus propias rutinas de manipulación de imágenes sin coste adicional o restricciones de licencia.

Java Architecture for XML Binding (JAXB): permite a los desarrolladores de Java mapear clases Java a formato XML. JAXB proporciona dos funcionalidades principales: la habilidad de presentar Objetos Java en XML (marshal), y la inversa, obtener Objetos Java mediante un XML (unmarshal).



Commons Math es una librería de componentes matemáticas y estadísticas que aborda los problemas más comunes no disponibles en el lenguaje de programación Java o en los Commons Lang.

JAMA es un paquete básico de álgebra lineal para Java que proporciona clases a nivel de usuario para construir y manipular matrices.

Spring Framework, también conocido simplemente como Spring, es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java.

Apache HTTP Client provee una manera eficiente, actualizada y rica de implementar la parte cliente de los estándares más recientes del protocolo HTTP.

Java Persistence API (JPA), es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional.

JUnit es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.

Nom.tam.fits es una librería open source de manejo de ficheros fits que permite lectura y escritura eficiente de ficheros FITS en Java.

Algoritmos

Se presentan a continuación los algoritmos más importantes usados en DNEB. Cada uno de ellos representa una fase dentro de la aplicación y la mayoría se apoyan en los resultados obtenidos por algoritmos previos. Por esta razón, vamos a explicarlos de forma cronológica, intentando así que el lector comprenda el funcionamiento intrínseco de la aplicación que nos ocupa.

Apertura y modificación de la imagen

FITS, al igual que todos los formatos de imagen basados en mapas de bits, representa la imagen con una o varias matrices de números, donde cada posición representa el color (o una de las componentes RGB) del píxel. Aun con estas similitudes, FITS presenta varias particularidades que hay que salvar en el momento de cargar la imagen.

El primero de los problemas lo encontramos en que para FITS, a la matriz que contiene los datos se accede usando coordenadas cartesianas, esto es, la fila de abajo se corresponde con Y=0 y crece hacia arriba. Esto no supone un gran problema hasta que trasladamos esa matriz a cualquier lenguaje de programación, donde por convenio es la fila de arriba la que se corresponde con Y=0 y crece hacia abajo. Así pues, es necesario invertir la matriz verticalmente para que la imagen se represente correctamente por pantalla.

```
para i=1 hasta alto/2
      para j=1 hasta ancho
            temp = imagen[i][j];
            imagen[i][j] = imagen[alto – 1 - i][j];
            imagen[alto – 1 - i][j] = temp;
      fpara
fpara
```

El segundo de los problemas que nos encontramos es la representación de la intensidad del píxel. Vamos a trabajar con imágenes de 2 bytes por pixel y mientras que en otros formatos

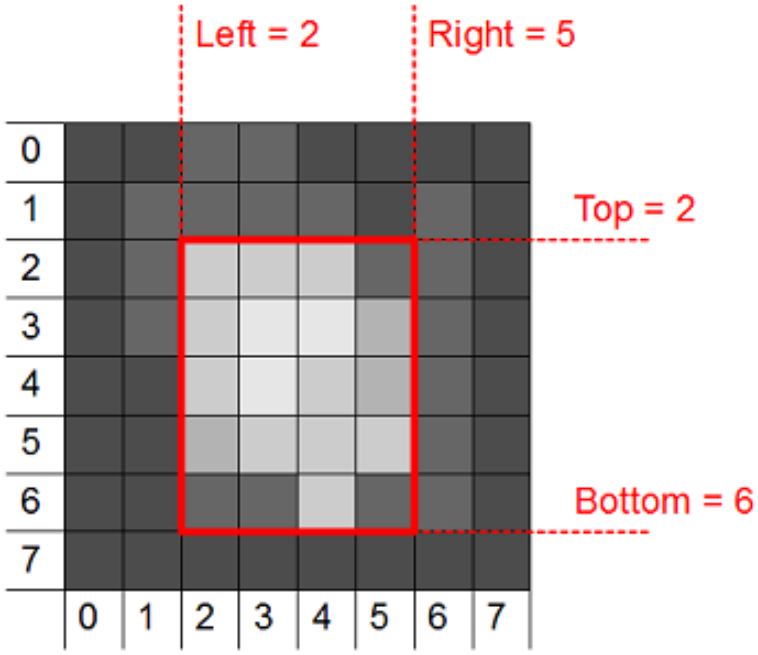
esto significaría que el valor mínimo (el 0) representa el negro y el valor máximo (2^{16} - 1 = 65535) representa el blanco, en FITS no es así.

FITS tiene en cuenta el número de intensidades distintas presentes en la imagen (tomemos 15000 para esta explicación) y un valor mínimo cualquiera (por ejemplo 35000). Con estos valores, el negro sería el valor 35000 y el blanco el 35000+15000 = 50000. Los valores intermedios son los distintos niveles de intensidad de la imagen. Como esta forma de representación no es la habitual y dará problemas a la hora de mostrar la imagen por pantalla, debemos escalar los valores para que se muevan en el rango apropiado.

```
min = min(imagen);
max = max(imagen);
escala = (max - min) / 65535;
para i=1 hasta alto
      para j=1 hasta ancho
            imagen[i][j] = (imagen[i][j] – min) / escala;
      fpara
fpara
```

Algoritmo de búsqueda de estrellas

Una vez que tenemos la imagen cargada en memoria, nuestro siguiente paso es identificar las estrellas. Representaremos una estrella como la porción rectangular que ocupa dentro de la imagen, delimitando este rectángulo a través de cuatro valores (top, bottom, left y right).



La búsqueda de estrellas se realiza en base a los dos parámetros:

Brillo: Es el valor de intensidad a partir del cual consideramos que un píxel pertenece a una estrella. A medida que vamos recorriendo la matriz que representa la imagen, todo aquel píxel que encontremos cuyo valor sea mayor o igual que el brillo establecido activará la identificación de una estrella.

Umbral: Es el valor de intensidad a partir del cual consideramos que un píxel deja de pertenecer a una estrella. Así pues, los pixeles cuyo valor sea menor que umbral son candidatos a no pertenecer a una estrella. Usamos esta definición para delimitar los rectángulos que encuadran las estrellas, de tal forma que los valores de todos los pixeles del borde del rectángulo son menores que el umbral.

El algoritmo que busca las estrellas es el siguiente:

```
buscarEstrellas(imagen[1:alto][1:ancho], brillo, umbral)

{
      para i=1 hasta alto
            para j=1 hasta ancho
                  brilloPixel = imagen[i][j];
                  si (brilloPixel >= brillo and not enOtroRect(i,j))
                        rect = crearRectangulo(i,j);
                        expandirRectangulo(rect, imagen, umbral);
                        añadirAListaRectangulos(rect);
                  fsi
            fpara
      fpara
}
```

donde utilizamos las siguientes funciones auxiliares:

`enOtroRect(i,j)` que devuelve cierto si el píxel (i,j) se encuentra en alguno de los rectángulos ya creados.

`crearRectangulo(i,j)` que crear un rectágulo inicial que solo engloba al píxel (i,j).

`expandirRectangulo(rect, imagen, umbral)` que expande el rectángulo dado hasta que todos los píxeles de su borde contengan una valor menor que el umbral.

Algoritmo de cálculo del centroide

Tenemos identificadas las estrellas en la imagen, nuestra siguiente tarea es encontrar los centroides de estas. Para ello nos basamos en el algoritmo explicado en *[Galadí-Enriquez 1998, pag. 58]* que funciona de la siguiente forma:

$$x_{centroide} = \frac{\sum_i (i(\sum_j I_{ij}))}{\sum_i \sum_j I_{ij}}$$

$$y_{centroide} = \frac{\sum_j (j(\sum_i I_{ij}))}{\sum_i \sum_j I_{ij}}$$

Siendo I_{ij} la intensidad del píxel en la posición (i,j).

```
calcularCentroide(imagen[1:alto][1:ancho]): Centroide
{
      total = 0;
      centroideY = 0;
      centroideX = 0;
      para i=1 hasta alto
            temp = 0;
            para j=1 hasta ancho
                  // temp es la suma de la fila i-esima
                  temp = temp + imagen[i][j];
                  total = imagen[i][j];
            fpara
            // el valor de temp se pondera por el número de fila
            // y se acumula en centroideY
            centroideY = centroideY + i*temp;
      fpara

      para j=1 hasta ancho
            temp = 0;
            para i=1 hasta alto
                  // temp es la suma de la columna j-esima
                  temp = temp + imagen[i][j];
            fpara
            // el valor de temp se pondera por el número de
            // columna y se acumula en centroideX
            centroideX = centroideX + j*temp;
      fpara

      centroide.x = centroideX/total;
      centroide.y = centroideY/total;
      devolver centroide;
}
```

Algoritmo de conversión de coordenadas decimales a sexagesimales

Éste algoritmo nos permite realizar conversiones de coordenadas en grados decimales a coordenadas en grados sexagesimales. Consiste en pasar a horas minutos y segundos los grados de ascensión recta y declinación de las coordenadas decimales.

```
decimalASexagesimal(CoordenadaDecimal dc): CoordenadaSexagesimal
{
      /*Creación de nuevo objeto coordenada sexagesimal donde se guardará
      la información de las coordenadas sexagesimales*/
      CoordenadaSexagesimal sc = nuevo CoordenadaSexagesimal();

      //Calculamos la Ascensión Recta en horas, minutos y segundos.
      sc.ARhoras = (int) (dc.AR * 24 / 360);
      sc.ARmin = (int) ((dc.AR - 15*sc.ARhoras) * 4);
      sc.ARseg = (dc.AR - (sc.ARhoras*60 + sc.ARmin)/4)*240;

      //Calculamos la declinación en horas minutos y segundos
      sc.DEChoras = (int) dc.DEC;
      sc.DECmin = (int) (|(dc.DEC - sc.DEChoras) * 60|);
      si (dc.DEC*3600 > sc.DEChoras*3600 – sc.DECmin*60)
      sc.DECseg = dc.DEC*3600 - sc.DEChoras*3600 - sc.DECmin*60;
```

```
      sino
      sc.DECseg = -dc.DEC*3600 + sc.DEChoras*3600 - sc.DECmin*60;

      //Retorno del objeto con las coordenadas sexagesimales
      devolver sc;
}
```

Algoritmo de conversión de coordenadas sexagesimales a decimales

Éste algoritmo nos permite realizar conversiones de coordenadas en grados sexagesimales a coordenadas en grados decimales.

```
sexagesimalADecimal(CoordenadaSexagesimal sc): CoordenadaDecimal
{
      //Creamos un nuevo objeto de coordenada decimal
      CoordenadaDecimal dc= nuevo CoordenadaDecimal ();

      //Pasamos las horas minutos y segundos de AR y declinación a grados
      dc.AR = (sc.ARhoras*3600 + sc.Armin*60 + sc.ARseg) / 240;

      si (sc.DEChoras >= 0)
            dc.DEC = sc.DEChoras + sc.DECmin/60 + sc.DECseg/3600;
      sino
            dc.DEC = sc.DEChoras - sc.DECmin/60 - sc.DECseg/3600;

      devolver dc;
}
```

Algoritmo de cálculo de posición

Éste algoritmo nos permite poder buscar estrellas dobles conocidas y de las cuáles conomos entre otras cosas la posición en la que estaban la última vez que fueron observadas y la distancia y ángulo entre ambas, gracias a estos datos el algoritmo es capaz de calcular la posición actual en la que se encuentran la dos estrellas que forman parte del sistema binario. Es un algoritmo muy útil ya que con la información adecuada nos permite mantener actualizada la posición de estrellas dobles con el paso del tiempo evitando de ésta manera que se pierdan.

Éste algoritmo depende de otros algoritmos explicados en ésta memoria y se hará referencia a ellos de manera breve siendo necesario consultar la especificación de los mismo para más información.

Los parámetros de entrada de éste algoritmo son los siguientes:

- Brillo y umbral como parámetros de ajuste del algoritmo de búsqueda de estrellas .
- Número máximo de candidatos(maxCandidatos) que queremos obtener a ser la estrella binaria que estamos buscando, en caso de sobrepasar el límite el algoritmo no devuelve nada, ésto sirve para evitar devolver un número elevado de candidatos (lo cual significa que ha habido algún problema con la imagen y que no es apta para el algoritmo) y confundir al investigador,
- Máximo número de estrellas(maxEstrellas), en caso de que la imagen contenga un número de estrellas superior al límite se descarta el procesamiento.

- Margen de ángulo(margenAngulo) es el porcentaje en tanto por uno de diferencia que puede haber entre el ángulo de la última observación y el actual.
- Margen de distancia(margenDistancia) es el porcentaje en tanto por uno de diferencia que puede haber entre la distancia de la última observación y la actual.
- Distancia mínima(distanciaminima) es la mínima distancia que debe de haber entre las dos estrellas del sistema binario para que el algoritmo procese la imagen.
- Imagen(imagen) imagen que contiene datos relativos a la información que contiene una imagen que se encuentra que se encuentra en el atributo ruta del objeto Imagen.

Primero se describirá el funcionamiento del algoritmo y posteriormente se pasará a mostrar el código de la implementación del mismo. Es importante tener en cuenta que éste algoritmo accede a base de datos para consultar información en un catálogo de estrellas dobles que ha sido importado previamente mediante la herramienta Importar Catálogo de Estrellas Dobles que viene incluido en ésta aplicación.

El primer paso a dar por el algoritmo es buscar la información de la estrella doble a buscar en el catálogo de estrellas dobles, para hacer la búsqueda nos basamos en las coordenadas de la estrella más brillante y que son las coordenadas del punto central de la imagen. una vez obtenida la referencia del catálogo de estrellas dobles tenemos mucha más información para realizar el procesamiento, ya que tenemos la distancia entre las dos estrellas y el ángulo en su última observación.

Procedemos a cargar la imagen a memoria la cual se encuentra en formato FITS.

Ahora se procederá a realizar un proceso iterativo de ajuste limitado hasta que se encuentre un resultado relevante o bien se descarte el proceso por considerarse imposible debido a la complejidad del caso o el error en los datos de entrada(por ejemplo imagen mal enfocada o con un filtro en el cuál la estrella que se busca no aparece).

Este proceso repite el siguiente proceso que a continuación se detallará mientras no se encuentren candidatos a ser la estrella buscada, además se realizará un proceso iterativo de orden cúbico, variando en el primer nivel de anidamiento los márgenes de error de manera que en caso de que no encontremos resultados con los márgenes dados, el algoritmo los cambiará levemente con la esperanza de encontrar de ésta forma algún candidato, en el segundo nivel se recorrerán los diferentes tipos de ajuste, siendo éstos los siguientes:

- Ajuste 0= Se reduce continuamente el umbral para encontrar las estrellas.
- Ajuste 1= si se considera que pueden estar pegadas se aumentan brillo y umbral hasta encontrar las estrellas deseadas.
- Ajuste 2= Si la separación es muy grande es un problema porque el rango es amplio y hay demasiados datos, falsos positivos, considerar como arreglarlos.

Una vez decido el tipo de ajuste se pasa al tercer nivel de anidamiento en el que se ajustan los parámetros de brillo y umbral según el tipo de ajuste.

Una vez dentro de éste bucle se ejecuta la siguiente secuencia:

Con los parámetros de brillo y umbral buscamos las estrellas mediante el algoritmo de búsqueda de estrellas, una vez obtenidas se calculan los centroides, con los centroides, éstos

son posiciones relativas dentro la imagen, para pasarlos a coordenadas celestes aplicamos un algoritmo de conversión de coordenadas relativas a absolutas con las que ya podemos operar tranquilamente, a continuación lo que hacemos es calcular la distancia entre todos esos centros y guardarlos. A continuación procedemos a evaluar todas esas distancias y comprobar si las distancias se encuentran dentro de los márgenes de error de distancia y ángulo, en caso de ser así se considera cadidata, se continua evaluando las otras distancias y se termina al algoritmo guradando los datos en base de datos para ser posteriormente exportados mediante la herramienta de exportación a XML.

El algoritmo devuelve además una lista de puntos que son los centroides de las estrellas candidatas a ser el sistema binario buscado y puede ser visualizada con el visor específico que la aplicación dispone para este fin. La información detallada se debe de consultar mediante la exportación a XML de los datos.

A continuación se detalla el código del algoritmo.

```
algotirmoCalculoPosicion(double brillo, double umbral, int maxCandidatos,
                         int maxEstrellas, double margenAngulo,
                         double margenDistancia, double distanciaMinima,
                         Imagen imagen): List<Point>
{

/* Guardamos el valor inicial del brillo puesto que la variable brillo
 * será modificada posteriormente */
double brilloInicial = brillo;
double umbralInicial = umbral;
/* Ahora buscaremos la estrella en el catálogo de estrellas dobles. En
 * esta implementación se encuentra almacenado en una base de datos
 * relacional y tenemos acceso a ella mediante el framework de
 * persistencia JPA */
ArrayList<Point> points = new ArrayList<Point>();

/* MARGEN PARA PODER HACER CONSULTAS SOBRE DOUBLES EN BASE DE DATOS YA
 * QUE EL OPERADOR = NO ESTÁ DISPONIBLE */
double ar = Double.parseDouble(imagen.getAscensionRecta());
double dec = Double.parseDouble(imagen.getDeclinacion());
double param1 = ar + 0.0000001;
double param2 = ar - 0.0000001;
double param3 = dec + 0.000001;
double param4 = dec - 0.000001;

/* Se realiza una consulta sobre la base de datos y se obtienen las
 * entradas del catálogo */
List<DoubleStarCatalog> dscList = manager.createQuery("select dsc from
      DoubleStarCatalog dsc where dsc.ascRecGrados < ? and
      dsc.ascRecGrados > ? and dsc.decGrados < ? and dsc.decGrados >
      ?").setParameter(1, param1).setParameter(2, param2).setParameter(3,
      param3).setParameter(4, param4).getResultList();

//En caso de no haber entradas no se puede ejecutar el algoritmo
if (dscList.size() < 1) {
      LOG.error("NO SE HA PODIDO LOCALIZAR LA INFORMACIÓN EN EL CATALOGO"
            + imagen.toString());
      throw new ServiceCalculoPosicionException("NO SE HA PODIDO
            LOCALIZAR LA INFORMACIÓN EN EL CATALOGO");
```

```
}

//Cogemos la primera entrada del catálogo
DoubleStarCatalog dsc = dscList.get(0);

//Obtenemos la ruta donde se encuentra el fichero
String filename1 = imagen.getRutaFichero();
Fits imagenFITS;
List<RectStar> recStars = new ArrayList<RectStar>();
try {

	int tipoAjuste = 0;
	/* Ajuste 0 = Se reduce continuamente el umbral para encontrar las
	 * estrellas.
	 * Ajuste 1 = si se considera que pueden estar pegadas se aumentan
	 * brillo y umbral hasta encontrar las estrellas deseadas.
	 * Ajuste 2 = Si la separación es muy grande es un problema porque
	 * el rango es amplio y hay demasaidos datos, falsos positivos,
	 * considerar como arreglarlos.*/

	//Apertura de fichero FITS
	imagenFITS = new Fits(new File(filename1));
	BasicHDU imageHDU = imagenFITS.getHDU(0);
	LectorImageHDU l = new LectorImageHDU(imageHDU, filename1);
	//Creamos un objeto que se encargará de buscar las estrellas
	StarFinder sf = new StarFinder();

	/* Variable que indica si es cierta que no hemos encontrado ninguna
	 * estrella binaria candidata */
	boolean sinRelevantes = true;

	/* Iteramos mientras no encontremos candidatos y los márgenes estén
	 * dentro de un rango razonable */
	while (sinRelevantes && margenAngulo <= 0.04 && margenDistancia <=
		0.10) {

		/* Mientras no tengamos candidatos hacemos pruebas con todos
		 * los ajustes */
		while (tipoAjuste < 2 && sinRelevantes) {
			brillo = brilloInicial;
			umbral = umbralInicial;

			/* Mientras no tengamos relevantes se hacen ajustes en
			 * brillo y umbral según el tipo de ajuste y hasta un
			 * límite */
			while (sinRelevantes) {

				//Se buscan las estrellas que hay en la imagen
				sf = new StarFinder();
				sf.buscarEstrellas(l, new Float(brillo), new
					Float(umbral));
				LOG.info("Numero de estrellas encontradas: " +
					sf.getNumberOfStars());
				recStars = sf.getRecuadros();
				/* Se calculan los centros de las estrellas
				 * encontradas
				ArrayList<Point> centroides = new
```

```
      ArrayList<Point>();
CalculateBookCentroid cc = new
      CalculateBookCentroid();
Point cent;
int nRecuadros = recStars.size();
if (nRecuadros > maxEstrellas) {
      throw new ServiceCalculoPosicionException(
            "DESCARTADA PORQUE EL NÚMERO DE
            ESTRELLAS SUPERA EL LÍMITE" +
            dsc.toString());
}
//Se calcula el centroide para cada estrella
for (int i = 0; i < nRecuadros; i++) {
      cent = cc.giveMeTheCentroid(
            l.getPorcionImagen(
                  recStars.get(i).getxLeft(),
                  recStars.get(i).getyTop(),
                  recStars.get(i).getWidth(),
                  recStars.get(i).getHeight()));
      cent.setX(recStars.get(i).getxLeft() +
            cent.getX());
      cent.setY(recStars.get(i).getyTop() +
            cent.getY());
      centroides.add(cent);
}

// PASAR DE PÍXELES A COORDENADAS
HashMap<DecimalCoordinate, Point> dcToPoint =
      new HashMap<DecimalCoordinate, Point>();
ArrayList<DecimalCoordinate> centroidesDC =
      new ArrayList<DecimalCoordinate>();
for (Point pointIter : centroides) {
    dcToPoint.put(dc, pointIter);
}

// CALCULAR DISTANCIAS ENTRE TODOS LOS CENTROIDES
List<Distance> distancesList =
      new ArrayList<Distance>();
for (DecimalCoordinate dc1: centroidesDC) {
      for (DecimalCoordinate dc2: centroidesDC) {
            Distance distanceAux =
                  this.mathService.
                  calculateDecimalDistance(
                        dc1.getAr(),
                        dc1.getDec(),
                        dc2.getAr(),
                        dc2.getDec());
            distanceAux.setPoint1(dc1);
            distanceAux.setPoint2(dc2);
            distancesList.add(distanceAux);
      }
}
List<InformacionRelevante> infoRels =
      new ArrayList<InformacionRelevante>();
for (Distance dist : distancesList) {
      double sep = dsc.getLastSeparation();
      double ang = dsc.getLastPosAnges();
```

```
                if (((sep * (1 - margenDistancia)) <=
                        dist.getDistanceSeconds() &&
                        dist.getDistanceSeconds() <= (sep *
                        (1 + margenDistancia))) && ((ang *
                        (1 - margenAngulo)) <=
                        dist.getAngle() && dist.getAngle() <=
                        (ang * (1 + margenAngulo)))) {

                        /* Si está dentro de rango se
                         * considera que es candidata y por
                         * lo tanto se guarda en Base de
                         * Datos en la tabla asignada para
                         * este fin */

                        InformacionRelevante ir =
                                new InformacionRelevante();
                        ir.setDescription("CALCULO DISTANCIA:
                                INFO DISTANCIA Y PUNTOS:" + " (
                                (LAST SEPARATION - CURRENT
                                DISTANCE)= " +
                                (dsc.getLastSeparation() -
                                dist.getDistanceSeconds()) +
                                ")" + dist + "INFO DSC" +
                                dsc.toString());
                        ir.setFecha(Util.dameFechaActual());
                        List<Imagen> imagenes =
                                new ArrayList<Imagen>();
                        imagenes.add(imagen);
                        ir.setImagenes(imagenes);
                        ir.setTipoInformacionRelevante(
                                manager.find(
                                        TipoInformacionRelevante.
                                                class, 2L));
                        infoRels.add(ir);
                        points.add(dcToPoint.get(
                                dist.getPoint1()));
                        points.add(dcToPoint.get(
                                dist.getPoint2()));
                        sinRelevantes = false;
                }
        }

        if (infoRels.size() <= maxCandidatos) {
                for (InformacionRelevante ir : infoRels) {
                        manager.persist(ir);
                }
        } else {
                throw new ServiceCalculoPosicionException(
                        "El número de resultados relevantes
                        supera el máximo: Se descarta la
                        imagen" + dsc.toString());
        }
        if (sinRelevantes) {
                if (tipoAjuste == 0) {
                        if (umbral < brillo) {
                                brillo -= brillo * 0.04;
                                LOG.info("NO SE HAN ENCONTRADO
```

```
                                                DATOS RELEVANTES,
                                                AJUSTANDO PARÁMETROS:
                                                BRILLO: " + brillo +
                                                "UMBRAL: " + umbral);
                                          } else {
                                                sinRelevantes = false;
                                          }
                                    } else {
                                          if (tipoAjuste == 1 &&
                                              brillo < 80000) {
                                            brillo = brillo + 200;
                                            umbral = umbral + 200;
                                          } else {
                                                sinRelevantes = false;
                                          }
                                    }
                              }
                        }
                        tipoAjuste++;
                  }
                  margenAngulo = margenAngulo * 1.25;
                  margenDistancia = margenDistancia * 1.10;
            }
      } catch (FitsException e) {
            e.printStackTrace();
      } catch (IOException e) {
            e.printStackTrace();
      }
      return points;

      }
```

Algoritmo de cálculo de distancias

Éste algoritmo nos permite calcular la distancia en la esfera celeste de dos puntos, devolviendonos como resultado la longitud entre los dos puntos y el ángulo entre los mismos. Para ellos se hace uso de la clase Math de Java y de dos funciones locales, rad y grados que convierten de grados a radianes y viceversa. Tiene como parámetros de entrada la ascención recta y la declinación de los puntos entre los que se quiere calcular la distancia, devuelve como resultado el objeto distancia que contiene la distancia entre los dos puntos.

```
calculateDecimalDistance(Double ar1, Double dec1, Double ar2, Double
                        dec2): Distance
{
      // Math.pow eleva un número a una potencia determinada
      double a = Math.pow(Math.sin(rad((dec2 -
                  dec1) / 2)), 2) + Math.cos(rad(dec1)) *
                  Math.cos(rad(dec2)) * Math.pow(Math.sin(rad(((ar2 -
                  ar1) / 2))), 2);
      double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt((1 - a)));
      double ap1 = grados(Math.asin(Math.cos(rad(dec2)) *
                   Math.sin(rad((ar2 - ar1))) / Math.sin(c)));
      double dist = grados(c) * 3600;
      double ang;
      if (ar2 > ar1){
```

```
            if (dec2 < dec1){
                ang = 180 - ap1;
            } else {
                ang = ap1;
            }
        } else {
            if (dec2 >= dec1){
                ang = 360 + ap1;
            } else {
                ang = 180 - ap1;
            }
        }
        double distSec = dist / 3600;
        Distance distance = new Distance(distSec, ang);
        distance.setDistanceSeconds(dist);
        return distance;
}

// Funciones auxiliares
rad(Double degree): Double
{
    return Math.toRadians(degree);
}

grados(Double rad): Double
{
    return Math.toDegrees(rad);
}
```

Algoritmo de búsqueda de estrellas dobles

La búsqueda de estrellas dobles consiste en, a partir de dos imágenes del mismo trozo de cielo y de diferente catálogo, encontrar el conjunto de estrellas binarias que hay en dicho trozo.

El proceso de búsqueda se divide en dos subprocesos. Por un lado tenemos la búsqueda de estrellas candidatas a haberse movido y, por otro, del resultado obtenido cogemos las candidatas a que sean dobles.

1. Búsqueda de estrellas candidatas a haberse movido

Este primer subproceso comienza una vez aplicado el algoritmo de la búsqueda de estrellas y del cálculo de centroides explicados anteriormente a las dos imágenes a analizar. La asignación de los parámetros de umbral y brillo para el primer algoritmo se realizó después de hacer varios estudios con el algoritmo de búsqueda de estrellas para diferentes valores de ambos parámetros. Los que mejor se comportaron fueron 40000 unidades para el umbral y 2000 unidades más para el brillo (42000 unidades).

Una vez se encuentran todas las estrellas, se toma la imagen con menos estrellas encontradas como la imagen a analizar (imagen de referencia), y la otra será con la que se compare (imagen auxiliar).

```
// Búsqueda de estrellas de dos imágenes im1 e im2
real umbral, brillo;
umbral = 40000;
brillo = umbral + 2000;
```

```
// Llamada al algoritmo de búsqueda de estrellas
Estrella[] es1 = buscarEstrellas(im1, brillo1, umbral1);
Estrella[] es2 = buscarEstrellas(im2, brillo2, umbral2);
int maxNumEstrellas = max(es1.longitud, es2.longitud);

// Obtención de los centroides
Centroide[] ce1, ce2;
for (int i = 0; i < maxNumEstrellas; i++) {
      // Llamada al algoritmo de cálculo de centroides
      if (i < es1.longitud)
            ce1.cambiaElem(i, dameCentroide(es1.dameElem(i)));
      if (i < es2.longitud)
            ce2.cambiaElem(i, dameCentroide(es2.dameElem(i)));
}

// Tomamos ce1 como los centroides referencia y ce2 como los auxiliares
if (es1.longitud > es2.longitud)
intercambiar(ce1, ce2);
```

Debido a la diferencia de años en los que se tomaron las dos imágenes a analizar, la de referencia puede estar escalada, trasladada o rotada con respecto de la auxiliar, por lo que un centroide puede tener unas coordenadas (de píxel) en una imagen y unas diferentes en la otra. Para poder cuadrar ambas coordenadas al principio se realiza un primer emparejamiento. Cada uno de los centroides obtenidos en la imagen de referencia se empareja con los de la imagen auxiliar. El método de emparejamiento consiste en ver, dentro de un radio dado, cuál es el centroide más cercano al de referencia (escalado con respecto a la imagen auxiliar) y que cuya estrella asociada tenga el brillo más parecido a la de dicho centroide.

```
/* Dado un centroide c, su estrella e y un radio de búsqueda, encontrar
 * en el conjunto de centroides ce auxiliares (y con la ayuda del
 * conjunto de estrellas asociadas es) el centroide auxiliar
 * correspondiente.
 */

real distancia, distanciaMin = radioBusqueda;
real brillo1 = e.dameBrillo(), brillo2, brilloParecido = 0;
Centroide aux, resultado = null;
for (int i = 0; i < ce.longitud; i++) {
      aux = ce.dameElem(i);
      distancia = distancia(c, aux);
      brillo2 = es.dameElem(i).dameBrillo();
      if (resultado == null) {
            if (distancia <= distanciaMin) {
                  /* Si el centroide auxiliar está dentro de un radio
                   * máximo de búsqueda */
                  distanciaMin = distancia;
                  brilloParecido = brillo2;
                  resultado = aux;
            }
      } else {
            if (distancia <= distanciaMin &&
                abs(brillo1 - brillo2) <= abs(brillo1 - brilloParecido)){
```

```
                    /* Si el centroide auxiliar está dentro de un radio
                     * máximo de búsqueda y su brillo es más parecido que
                     * el almacenado anteriormente */

                    brilloParecido = brillo2;
                    distanciaMin = distancia;
                    resultado = aux;
              }
        }
}
```

En este primer emparejamiento el radio de búsqueda es inferior al del segundo, ya que se ha de evitar que las estrellas que se han movido en la realidad se utilicen para cuadrar las imágenes.

```
// Emparejamiento de los centroides
Centroide[] centroides, parejas;
Centroide c1, c2;
real radBus = 2.25;
for (int i = 0; i < ce1.longitud; i++) {
      // Se escala según los tamaños de ambas imágenes
      c1 = ce1.dameElem(i);
      c2 = dameCentroideAuxiliar(c1, es1.dameElem(i), ce2, es2, radBus);
      if (c2 != null) {
            if (parejas.esta(c2)) {
                  int pos = parejas.posicion(c2);
                  if (distancia(c1, c2) <
                      distancia(centroides.dameElem(pos), c2)) {
                        centroides.eliminar(centroides.dameElem(pos));
                        parejas.eliminar(c2);
                  } else continue; // No se incluye
            }
            centroides.insertar(c1);
            parejas.insertar(c2);
      }
}
```

Una vez obtenidos los emparejamientos se realiza el cálculo, mediante matrices (con la ayuda de la librería JAMA), del siguiente sistema lineal:

R * X = A donde R son los centroides de referencia y A los auxiliares.

Ahora, con la ayuda de la matriz X, se puede saber donde irían (de manera aproximada) cada uno de los centroides de referencia en la imagen auxiliar. A partir de estos datos se sabrá qué estrellas se han movido, ya que éstas no se encontrarán donde deberían estar. Pero, ¿cuál es el error por el que se determina si una estrella se ha movido? Éste es el que sea mayor que el doble de la desviación típica de todas los errores.

```
// Cálculo de la matriz de transformación
// Crear matrices con las coordenadas de los centroides en cada columna
Matriz R = crearMatriz(centroides);
Matriz A = crearMatriz(parejas);
Matriz X = A / R;
```

Para calcular todos los errores se multiplica cada uno de los centroides de referencia por la matriz de transformación X, y se realiza un segundo y último emparejamiento, pero ahora con un radio mayor para que se emparejen también las estrellas que se han movido.

```
// Segundo emparejamiento pero ahora con todos los centroides
R = crearMatriz(ce1);
A = R * X;
radBus = 10;
for (int i = 0; i < ce1.longitud; i++) {
      c1 = A.dameFila(i); // Coge el centroide que está en la fila i
      c2 = dameCentroideAuxiliar(c1, es1.dameElem(i), ce2, es2, radBus);
      if (c2 != null) {
            if (parejas.esta(c2)) {
                  int pos = parejas.posicion(c2);
                  if (distancia(c1, c2) <
                      distancia(centroides.dameElem(pos), c2)){
                        centroides.eliminar(centroides.dameElem(pos));
                        parejas.eliminar(c2);
                  } else continue; // No se incluye
            }
            centroides.insertar(c1);
            parejas.insertar(c2);
      }
}
```

Hecho el emparejamiento hallamos todos los errores mediante el cálculo de las distancias entre el centroide transformado y el centroide auxiliar correspondiente, y el que sea mayor que el doble de la desviación típica de todos se considerará candidato a haberse movido. Como estamos tratando imágenes en las que la precisión no es exacta se deja un poco más de cota para considerar a los candidatos, por lo que se usa un valor de 1.75 en vez de 2.

```
// Cálculo de los errores
real[] errores;
for (int i = 0; i < centroides.longitud; i++) {
      errores.cambiaElem(i, distancia(centroides.dameElem(i),
      parejas.dameElem(i)));
}

real desviacion = calcularDesviacionTipica(errores);

// Cálculo de los candidatos a haberse movido
Par<Centroide>[] resultado;
Par<Centroide> par;

for (int i = 0; i < centroides.longitud; i++) {
      if (errores.dameElem(i) > 1.75*desviacion) {
            par = (centroides.dameElem(i), parejas.dameElem(i));
            resultado.cambiaElem(i, par);
      }
}
```

2. Búsqueda de candidatas a ser estrellas dobles

Para determinar, dentro del conjunto de estrellas candidatas a haberse movido, cuáles son dobles, primero se calcula el vector director que va desde el centroide de referencia transformado con X al centroide auxiliar correspondiente. Para cada vector se miran cuáles son semejantes (un determinado porcentaje) en módulo, dirección y sentido, y los que lo cumplan serán los que su estrella asociada es candidata a ser doble.

```
// Búsqueda de estrellas dobles de dos imágenes im1 e im2

Par<Centroide>[] movimiento = busquedaEstrellasMovimiento(im1, im2);
Par<Centroide>[] resultado;
Par<Centroide> par;
boolean[] usados = {false, ..., false};
Centroide r1, r2, a1, a2;
real modulo1, modulo2, direccion1, direccion2;
Distancia dis; // Distancia en coordenadas celestes
real difModulo = 35; // 35 % de diferencia entre módulos
real difDireccion = 25; // 25 grados de diferencia entre direcciones
for (int i = 0; i < movimiento .longitud; i++) {
      r1 = movimiento.dameElem(i).damePrimerElemento();
      a1 = movimiento.dameElem(i).dameSegundoElemento();

      // distancia = sqrt ( ( r1.x - a1.x ) ^ 2 + ( r1.y - a1.y ) ^ 2 )
      modulo1 = distancia(r1, a1);
      if (modulo1 < 0.70) continue;

      // direccion = atan ( ( a1.y - r1.y ) / ( a1.x - r1.x ) )
      direccion1 = direccion(r1, a1);
      for (int j = i+1; j < movimiento .longitud; j++) {
            r2 = movimiento.dameElem(j).damePrimerElemento();
            a2 = movimiento.dameElem(j).dameSegundoElemento();
            modulo2 = distancia(r2, a2);
            if (modulo2 < 0.70) continue;
            direccion2 = direccion(r2, a2);

            // Cálculo de la distancia decimal en arcosegundos
            dis = calcularDistanciaDecimal(im1, r1, r2);

            /* Si la distancia es mayor de dos arcominutos que se
             * descarte */
            if (dis > 120) continue;

            // Comprobar si los módulos y las direcciones son semejantes
            if (min(modulo1, modulo2) >= max(modulo1, modulo2) *
               (1 - difModulo) && abs(direccion1 - direccion2) <=
                difDireccion) {
                  // Comprobar que los sentidos sean los mismos
                  if ((a1.x-r1.x) * (a2.x-r2.x) + (a1.y-r1.y) *
                      (a2.y-r2.y) > 0) {
                        if (!usados.dameElem(i)) {
                              usados.cambiaElem(i, true);
                              par = (r1, a1);
                              resultado.insertar(par);
                        }
                        if (!usados.dameElem(j)) {
```

```
                                usados.cambiaElem(j, true);
                                par = (r2, a2);
                                resultado.insertar(par);
                            }
                        }
                    }
                } // fin for j
            } // fin for i
```

Construcción y pruebas

Desarrollo

A continuación se procede a detallar cómo ha progresado el desarrollo del software explicando cada versión importante del software desarrollada.

Herramienta de descarga de imágenes individuales de catálogos.

La primera versión consta de un cliente HTTP y un cliente Web Service para la descarga de imágenes de catálogos astronómicos. Es capaz de guardar estas imágenes en formato FITS y JPEG a disco de diferentes tamaños y lugares.

Herramienta de descarga de regiones de catálogos.

Desarrollada sobre la primera versión pero con la eliminación del cliente Web Service debido a que los catálogos soportados por el mismo no nos son relevantes para la investigación, se deja abierta la posibilidad de integrarlo en un futuro de nuevo en caso de ser relevante para la investigación.

Además de crea una nueva capa sobre el cliente HTTP que permite realizar descargas de amplias regiones divididas en diferentes imágenes. Ésto permite hacer algo que niguna herramienta hasta ahora era zappaz de hacer y quie nos permitirá estudiar grandes regiones del cielo en busca de estrellas binarias en el futuro.

Herramienta de descarga y visualización de imágenes en formato FITS.

Una vez finalizada y probada la descarga de imágenes procedemos al desarrollo de la lectura de ficheros de imágenes en formato FITS, ésto nos permitirá poder estudiar las imágenes mediante algoritmos que son desarrollados en las siguientes fases. Además en esta fase desarrollamos un visor de imágenes en formato FITS que nos permite mostrar las imágenes descargadas.

Herramienta de descarga y procesamiento de imágenes.

Una vez finalizada la parte de descarga y lectura de imágenes se procede a implementar el modelo de datos que dará soporte a la gestión de procesamientos y por lo tanto a la ejecución de los algoritmos sobre las imágenes. También se implementan los ejecutores y demás elementos del motor de procesamientos dejando pendiente la inclusión de la llamada a los diferentes algoritmos dentro de los ejecutores.

Se realiza un cambio en la interfaz gráfica pasando a un modelo basado en pestañas que se considera mucho más cómodo de usar.

En esta versión se comienza el desarrollo y test de los diferentes algoritmos de manera paralela al desarrollo de la herramienta y son integrados en las fases siguientes.

Se desarrolla una utilidad de consulta básica y otra de importación sobre el catálogo WDS.

Se desarrolla también una utilidad que permite exportar datos a XML mediante el uso de la librería JAXB para poder tener en texto información encontrada a lo largo de las diferentes investigaciones.

Herramienta de búsqueda de estrellas binarias.

En ésta fase se desarrolla y prueba el grueso de los algoritmos de búsqueda de estrellas dobles y de cálculo de posición y se comienzan las pruebas, para facilitar las mismas se integran los algoritmos con los ejecutores. Además se incorpora a los algoritmos la posibilidad de almacenar los datos relevantes que encuentran en una base de datos y posteriormente ser exportados a XML con la utilidad anteriormente comentada.

Se implementa una herramienta de consulta en el catálogo de estrellas binarias avanzada y que permite filtrar con cotas superiores e inferiores sobre la mayoría de los parámetros del WDS.

Herramienta de estudio de estrellas binarias.

Se desarrollan diferentes utilidades para la aplicación que están derivadas de los diferentes algoritmos y estudios realizados a lo largo del proyecto y que consideramos que tienen utilidad de forma independiente para los astrónomos.

Además se afinan los algoritmos gracias a las diferentes pruebas que se detallan en el siguiente apartado.

Pruebas

Para el desarrollo de las pruebas hemos usado el framework de test Junit 4. En las pruebas en las que hemos necesitado realizar validaciones visuales de los resultados hemos desarrollado herramientas propias que nos han permitido validar los resultados, como es el caso de los visores.

Las pruebas realizadas en la búsqueda de estrellas dobles han sido bastante satisfactorias. La aplicación ha sido capaz de encontrar dos estrellas dobles que nadie había encontrado aún. Para ello se tuvieron que hacer muchas comprobaciones para ajustar determinados parámetros del algoritmo. A continuación se explican algunas de ellas.

En un principio, para realizar la búsqueda de estrellas en las imágenes, se pensó en dar como valores de brillo y umbral un determinado percentil de los valores de intensidad de la imagen, así éstos se amoldarían según el caso. Pero tras varias pruebas se llegó a la conclusión de que, para imágenes con pocas estrellas (pocos valores de intensidad), este método proporcionaba valores muy pequeños de umbral y brillo, y se encontraban estrellas donde no las habían, por lo que se decidió poner un valor fijo para ambos. El valor que mejor resultados dio fue el de 40000 unidades para el umbral y 42000 para el brillo.

Una de las cosas que más tuvo que ser probada fue el de los emparejamientos de estrellas. Como criterios a seguir para saber cuál era la pareja de cada estrella se realizaba una búsqueda dentro de un radio dado y se comparaban las intensidades de ambas estrellas para ver si coincidían. Al inicio el radio de búsqueda era el mismo para los dos emparejamientos que se realizan en el algoritmo. Luego se llegó a la conclusión de que éste tenía que ser menor en el primero, ya que se pretendía evitar que se encontrasen las estrellas que se habían movido, pues esto provocaría un mal encaje luego. Esta reducción de radio llevó a que en imágenes muy rotadas una respecto de la otra no se encontrasen más del 50% de emparejamientos, por lo que se descartaban.

Respecto al tema del brillo de las estrellas, primero se miraba si ambos brillos eran semejantes dentro de un porcentaje. Tras varias pruebas se llegó a la conclusión de que para bastantes imágenes el escalado entre los pares correspondientes de imágenes podía ser muy alto, como cuatro veces más. También se observó que el algoritmo de emparejamiento siempre emparejaba la primera estrella de la lista bien, por lo que se tomó como escala de referencia entre ambas imágenes el cociente entre las intensidades de ambas estrellas.

Resultados

Resultados de búsqueda de estrellas dobles:

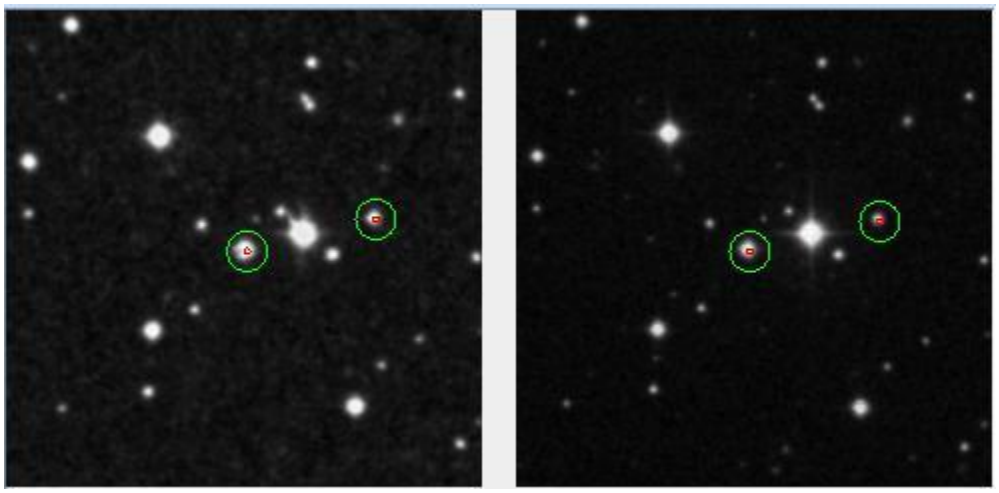
Como ya se ha explicado en el apartado anterior, los resultados del algoritmo de búsqueda de estrellas dobles han sido bastante buenos debido a las nuevas estrellas dobles encontradas por la aplicación. Además de éstas, se han encontrado otras estrellas que ya se encuentran en el catálogo WDS. Aquí se muestran algunos de los resultados obtenidos por la aplicación. Para cada caso se mostrará el resultado del algoritmo en el visor dedicado a la búsqueda de estrellas dobles y, gracias a Aladin, se mostrará una imagen adicional formada por las dos imágenes superpuestas con un filtro verde para una y otro rojo para la otra. De esta manera se apreciará mejor el movimiento de ambas estrellas.

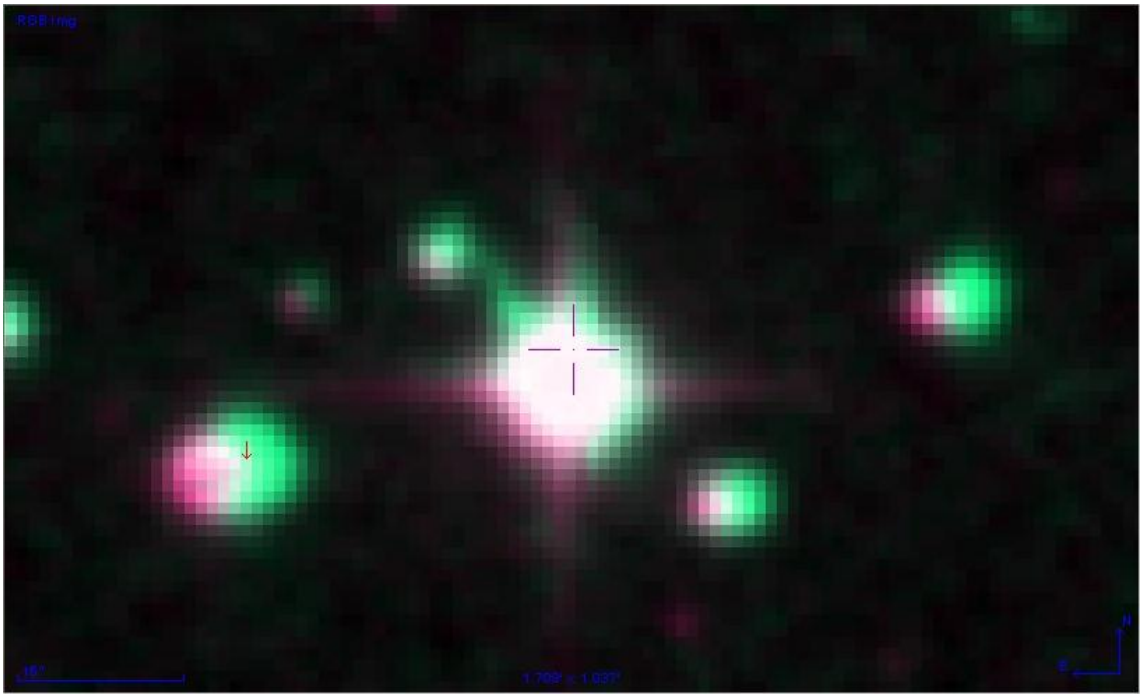
Nuevas estrellas dobles

1) Coordenadas:

AR: 10.4919

DEC: 45.1270



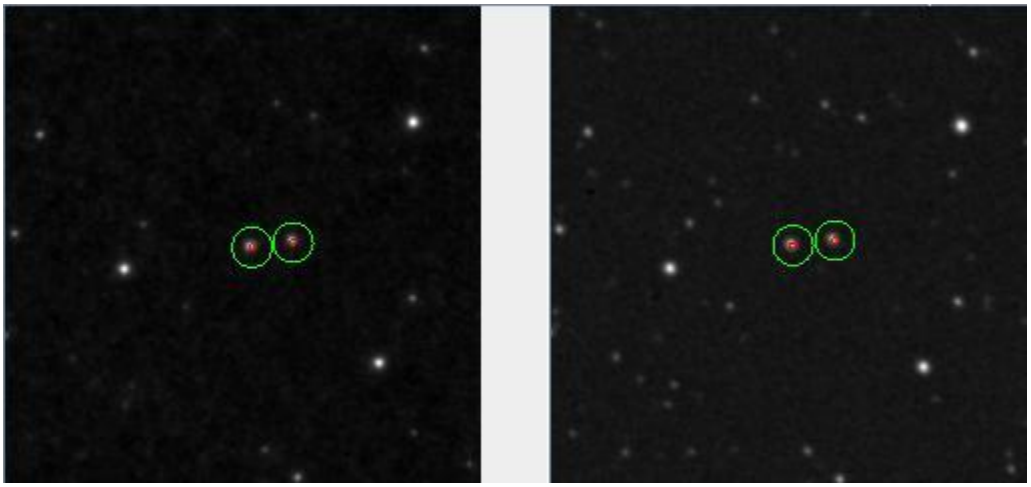


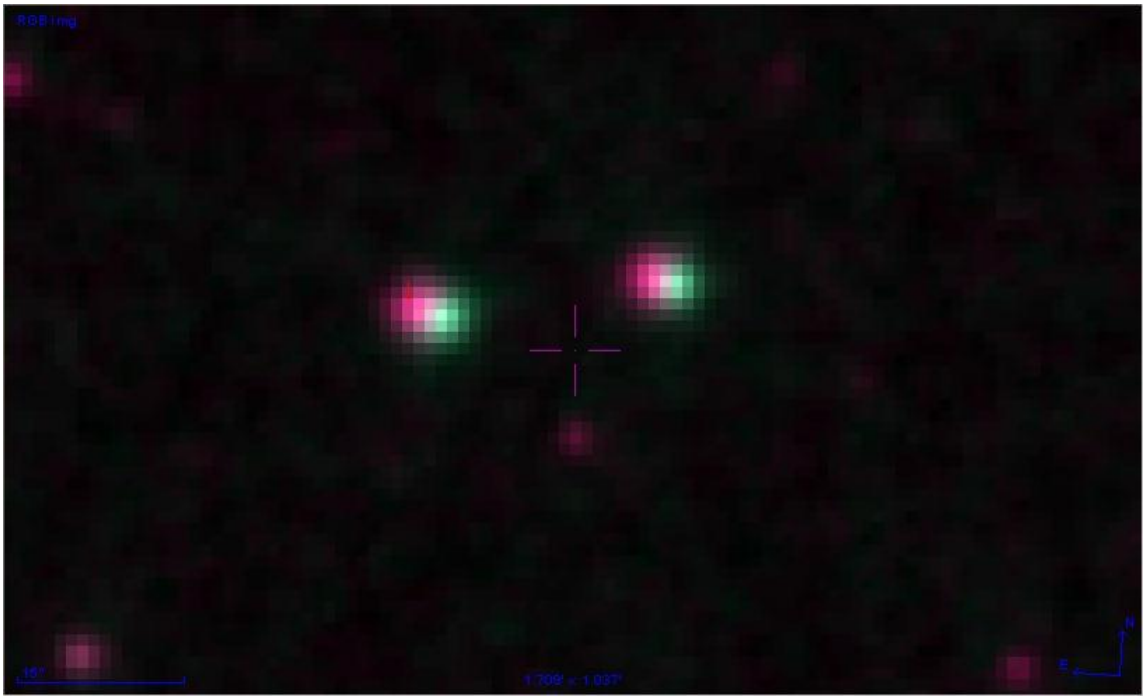
Estrellas dobles ya encontradas

1) Coordenadas:

AR: 5.5728

DEC: 67.1471

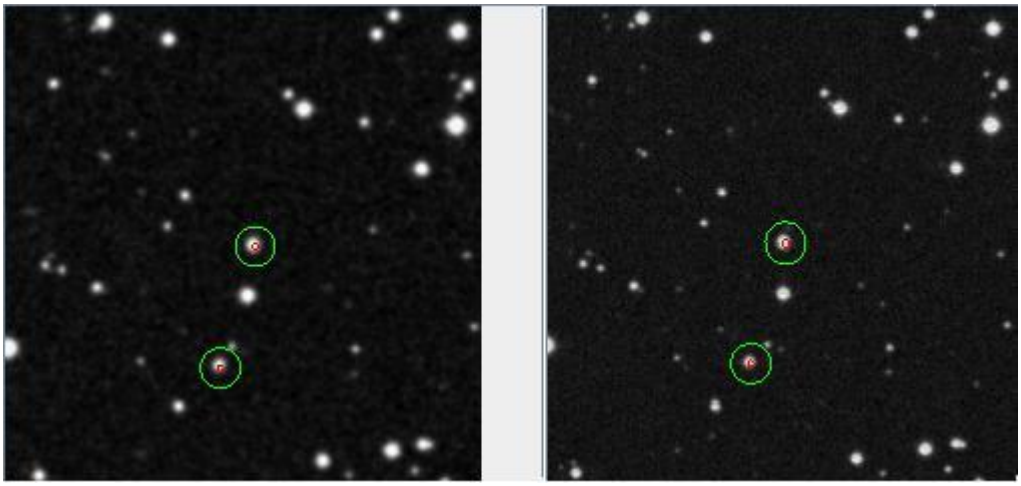


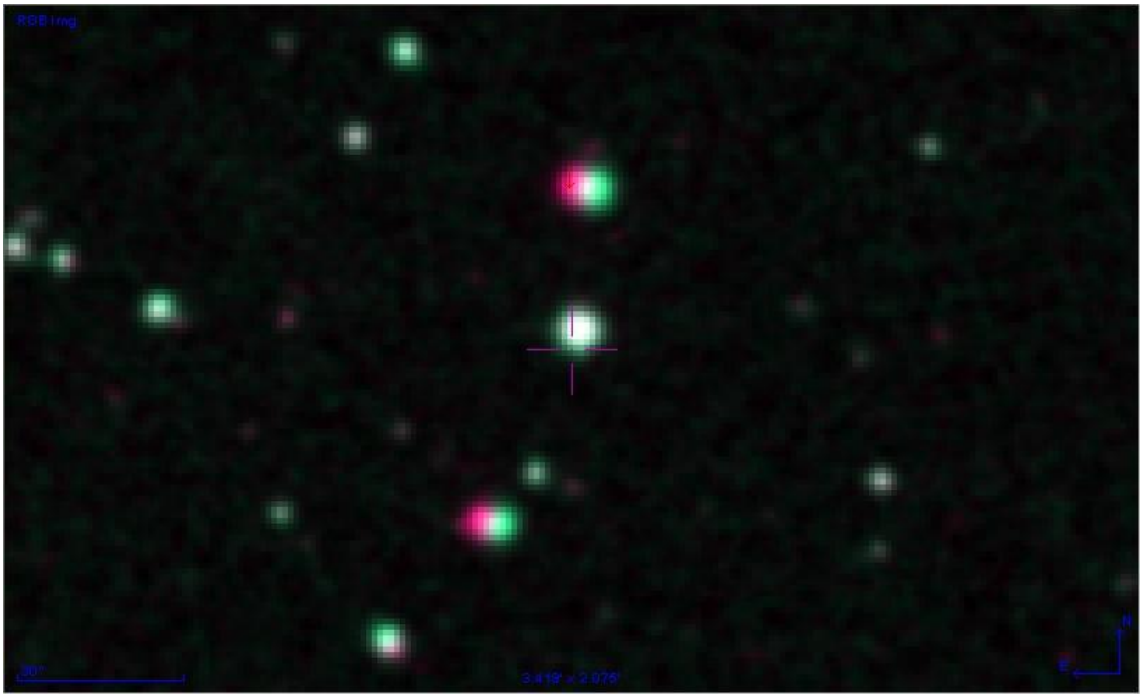


2) Coordenadas:

AR: 310.4797

DEC: 2.2938

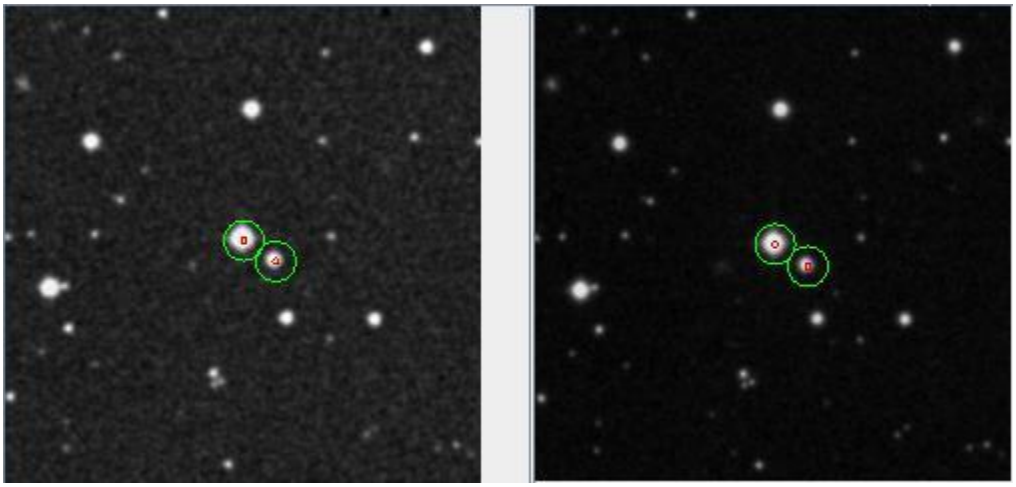


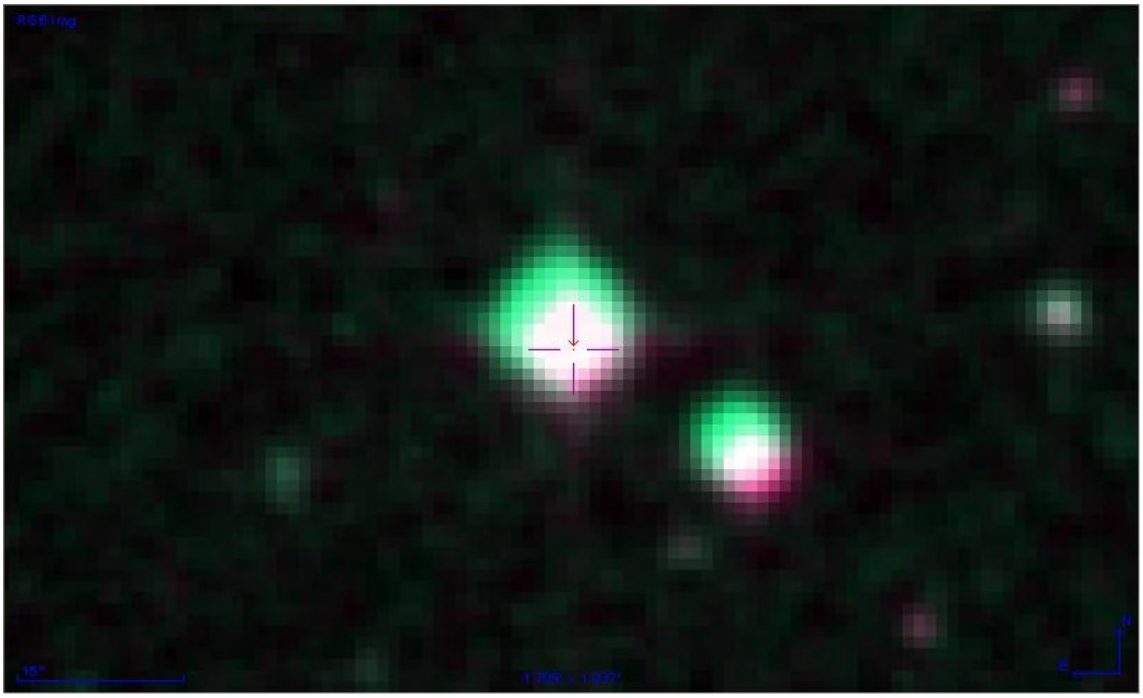


3) Coordenadas:

AR: 346.2734

DEC: 38.6827





Resultados de cálculo de posición de estrellas dobles ya conocidas y que se han movido:

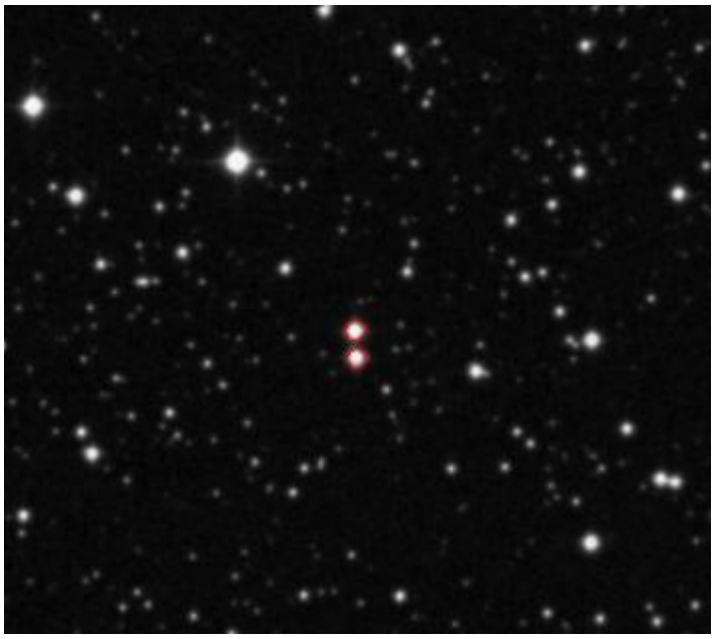
Los resultados han sido bastante buenos ya que se ha conseguido localizar un gran número de estrellas dobles del catálogo, aunque en muchos de los casos el movimiento de las mismas había sido escaso, ha permito obtener información de pequeños ajustes en los catálogos y localizar estrellas que había mucho tiempo que no eran observadas.

Los principales problemas han surgido con estrellas localizadas en lugar con alta concentración de estrellas y que han provocado la aparición de falsos testigos, en estos casos se ha decido descartar los datos.

En algunos de los casos hemos podido verificar que se han producido movimientos relevantes, en especial en estrellas que hace más de 50 años que no son observadas y localizadas de manera oficial en los catálogos de estrellas dobles.

A continuación se procede a explicar algunos de los resultados concretos que se han obtenido localizando estrellas binarias.

Localización de la estrella binaria STI1873 de la que no se tenía datos oficiales desde 1908 y que se puede observar en la siguiente imagen marcada en color rojo, ésta imagen ha sido generada por la aplicación DNEB.



Los datos obtenidos para ésta estrella binaria son los siguientes.

Posición estrella 1: ar = 36.463116550240606 dec = 58.35537491052963

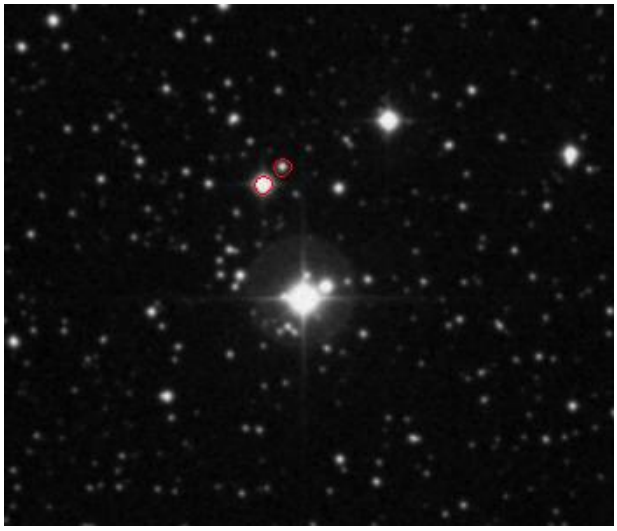
Posición estrella 2: ar = 36.46282346331172 dec = 58.35155931529572

Distancia (grados): 0.003818692718725281

Distancia (segundos): 13.747293787411012

Ángulo: 182.30801008708062

Localización de la estrella binaria STI1895 de la que no se tenía datos oficiales desde 1908 y que se puede observar en la siguiente imagen marcada en color rojo, ésta imagen ha sido generada por la aplicación DNEB.



Posición estrella 1: ar = 37.46577986356257 dec = 58.527490992348596

Posición estrella 2: ar = 37.45992575566473 dec = 58.53025554505287

Distancia (grados): 0.004121091936667393

Distancia (segundos): 14.835930972002616

Ángulo: 312.1336646486493

Localización de la estrella binaria STI1971 de la que no se tenía datos oficiales desde 1978 y que se puede observar en la siguiente imagen marcada en color rojo, ésta imagen ha sido generada por la aplicación DNEB.

Posición estrella 1: ar = 49.32691666533825 dec = 58.16024566880858

Posición estrella 2: ar = 49.31902525981958 dec = 58.161276758541476

Distancia (grados): 0.06030883723651925

Distancia (segundos): 15.43969116276348

Ángulo: 283.91434782772797

Posibles ampliaciones

Reconocimiento de estrellas mejorado

Seria interesante poder mejorar el reconocimiento de estrellas en las fotografías. Aunque el algoritmo descrito en la página 41 funciona en la mayoría de casos, depende de dos parámetros que se han tenido que ajustar empíricamente y puede devolver resultados erróneos en según que ocasión.

Hemos apreciado que si dos estrellas están demasiado juntas, el algoritmo puede tomarlas como solo una, perdiéndose así posibles candidatas a estrellas binarias. Una posible manera de resolver esto sería aplicar un complejo algoritmo de detección de formas a la imagen, calculando primero los bordes de las figuras y comprobando después que estas tienen formas casi circulares. Esta idea se desecho debido a la complejidad, a la falta de conocimientos para abordarla y al tiempo que nos habría llevado.

No descartar imágenes rotadas

Debido a como funciona el algoritmo de búsqueda de estrellas dobles, se hace necesario descartar las imágenes que por defecto estén rotadas, ya que el emparejamiento de las estrellas entre imágenes se hace prácticamente imposible.

Si pudiésemos conocer el ángulo con el que esta rotada cada imagen y fuésemos capaces de deshacerlo, podríamos modificar el algoritmo de búsqueda de estrellas de forma que no tuviésemos que descartar ninguna imagen, aumentando así la posibilidad de encontrar nuevas estrellas. Somos conscientes de que esta información se encuentra disponible en la cabecera del fichero FITS que contiene la imagen, pero la falta de tiempo y sobretodo de información en la Web, hizo que decidiésemos descartar imágenes.

Ampliación de los surveys soportados por la aplicación.

En la actualidad la aplicación soporta un número limitado de surveys. Sería muy interesante poder incluir soporte para nuevos surveys, lo cual nos permitiría investigar zonas no soportadas por los catálogos actuales, como es el caso del hemisferio sur. Éste caso es especialmente interesante puesto que el hemisferio es una zona menos estudiada que el hemisferio norte y por lo tanto hay mayor posibilidad de encontrar nuevos datos relevantes. Para ello una de las soluciones más sencillas sería integrar el web service implementado en la primera versión de la aplicación y que posteriormente fue descartado debido a que no nos aportaba datos interesantes a corto plazo.

Conclusiones

Consideramos que hemos cumplido con los principales objetivos planteados ya que hemos abordado de manera satisfactoria la mayoría de los retos planteados para éste proyecto, a pesar de ello los resultados podrían haber sido mucho mejores si el periodo dedicado a las pruebas hubiera sido más largo, esto nos habría permitido afinar más los parámetros de los algoritmos. Además un periodo más largo de pruebas nos habría permitido validar el gran número de candidatas a ser estrellas binarias nuevas o conocidas, y que por falta de tiempo no han podido ser comprobadas.

Uno de los retos más ambiciosos del proyecto y que por problemas de tiempo no ha podido ser realizado a tiempo para la presentación de éste proyecto es la publicación de los resultados obtenidos en el catálogo WDS.

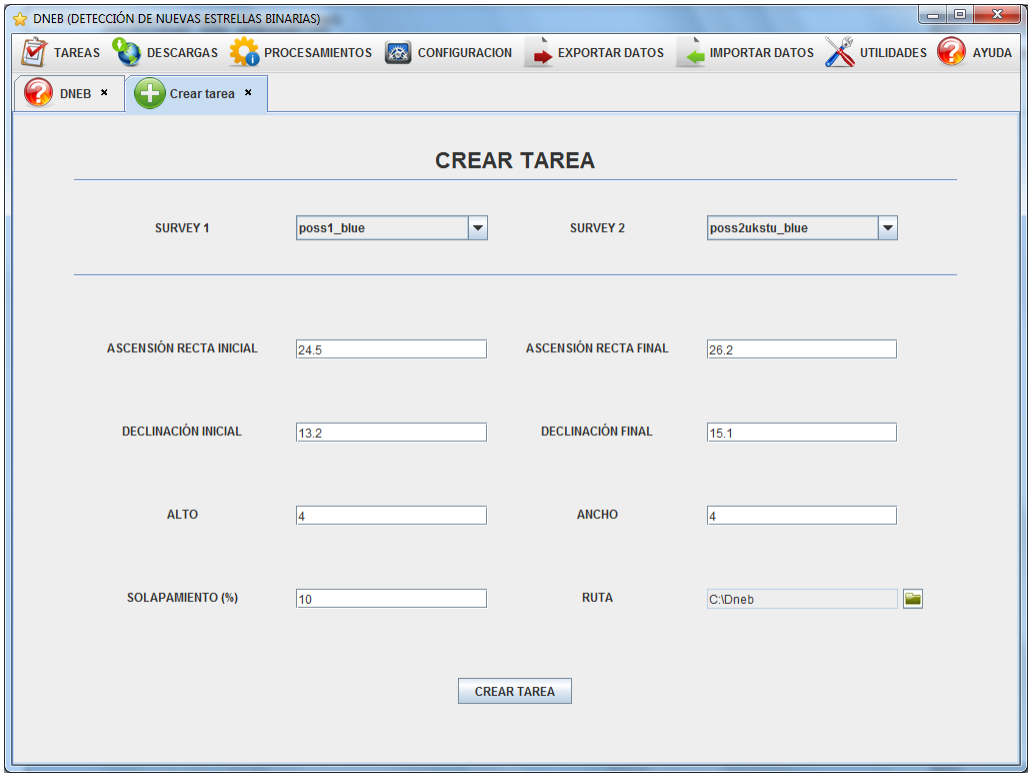
La aplicación es una buena base para futuros desarrollos en éste campo, ya sea mediante el ajuste de alguna de las funcionalidades existentes, o mediante la ampliación de las mismas. Además consideramos que mediante su uso, los astrónomos especializados podrán usar nuestra herramienta como apoyo para sus investigaciones y también aportar ideas o mejoras de la misma, ya que quién mejor que un astrónomo para orientar el futuro del proyecto DNEB.

Anexo I: Manual de usuario

A continuación se procede a explicar el funcionamiento de las diferentes utilidades de la aplicación.

Crear tareas

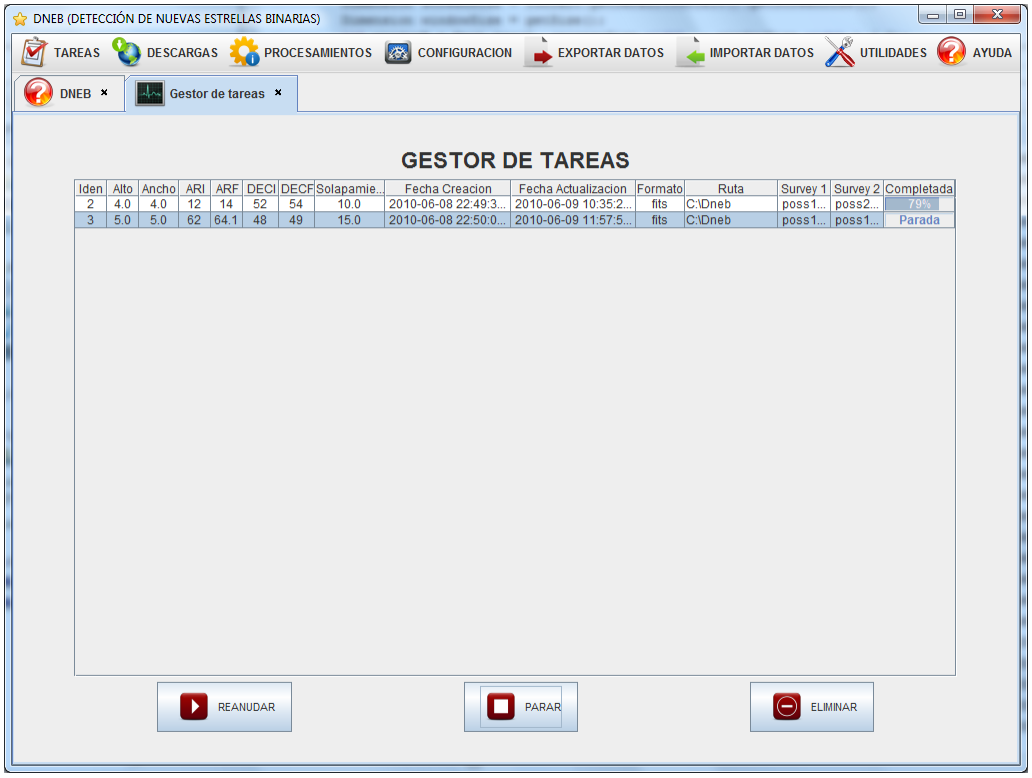
Para poder empezar a buscar estrellas dobles, necesitamos descargar las imágenes que recomponen la parcela de cielo deseada. Para ello, necesitamos crear una tarea en el menú *Tareas -> Crear una nueva tarea*.



Deberemos rellenar todos los campos. Las coordenadas de ascensión recta y declinación se expresan en grados y el alto y el ancho de la imagen en minutos de arco (arcmin). El solapamiento es útil para que una estrella no quede partida entre dos imágenes consecutivas. Por ultimo, debemos indicar el directorio en el que se almacenaran los ficheros FITS.

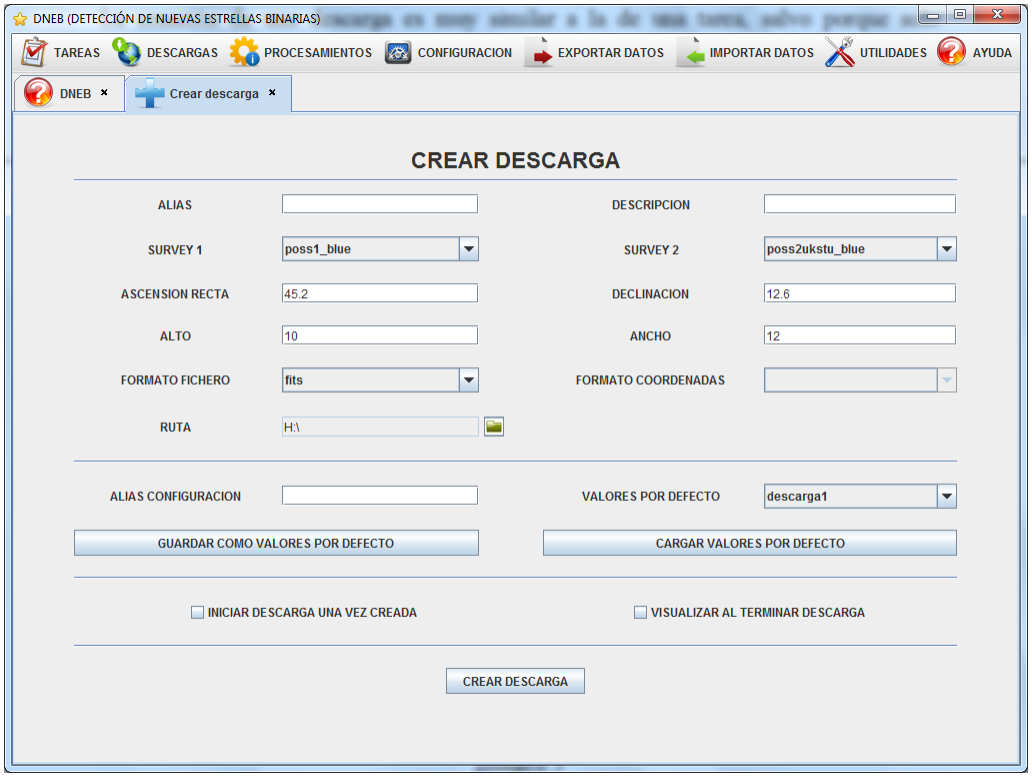
Gestionar tareas

Una vez que tengamos creadas tareas podremos gestionarlas. Con *Tareas -> Gestor de tareas* veremos la lista de las tareas creadas y no finalizadas y podremos reanudarlas, pararlas o eliminarlas. Es posible tener varias tareas activas a la vez e incluso podremos cerrar la pestaña *Gestor de tareas* y estas continuaran ejecutándose.



Crear descarga

Una descarga es una tarea compuesta únicamente por dos imágenes, una de cada survey. La creación de una descarga es muy similar a la de una tarea, salvo porque solo necesitamos introducir una ascensión recta y una declinación, que serán las coordenadas del centro de la imagen. Opcionalmente podremos darle un alias y una descripción a la descarga y cargar o guardar valores por defecto, pero explicaremos más detalladamente esto en el siguiente apartado.



Podemos crear una descarga con la opción *Descargas -> Crear una descarga*.

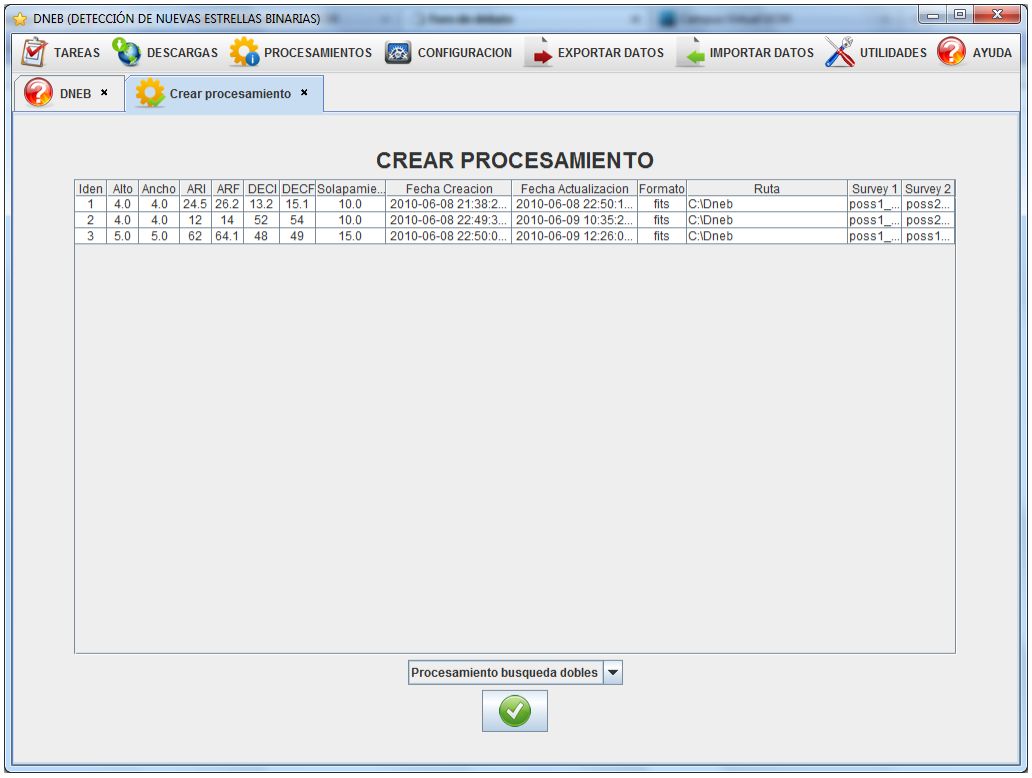
Configurar descargas

Si se van a realizar varias descargas que tengan en común algunos de los datos, como pueden ser los *surveys*, el alto, el ancho o la ruta, podemos crear una configuración de descarga para no tener que introducir los datos cada vez.

Para ello vamos a *Configuración -> Crear configuración de descarga*. Podremos gestionar distintas configuración; creándolas, actualizándolas o borrándolas.

Crear procesamientos

Una vez que una tarea se haya completado, podemos crear un procesamiento asociado a esta. Para esto, simplemente tenemos que ir a *Procesamientos -> Crear procesamiento*.



Solo hay que escoger una tarea, elegir el tipo de procesamiento que le queremos aplicar y aceptar. Se nos mostrará un mensaje advirtiéndonos de que el procesamiento se ha creado con éxito.

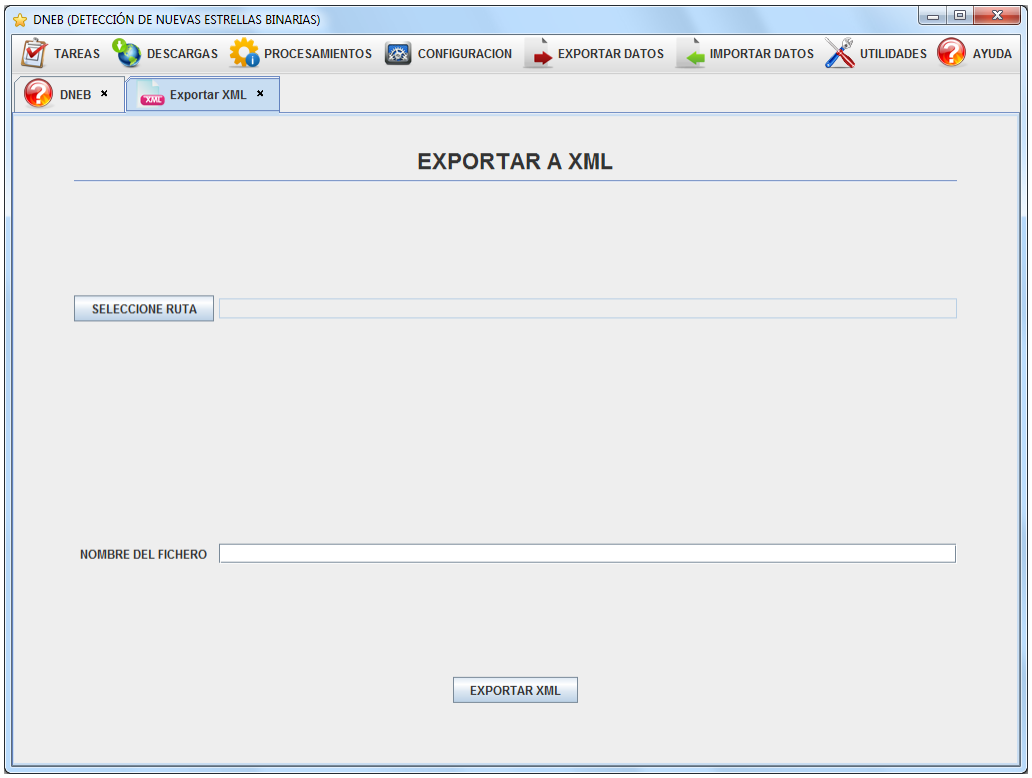
Gestionar procesamientos

Es idéntico a la opción de gestionar tareas pero con procesamientos. Una vez creado algún procesamiento, nos aparecerá en esta pantalla y podremos gestionar su funcionamiento; reanudándolo, parándolo o eliminándolo.

Accederemos a esta opción con *Procesamientos -> Gestor de procesamientos*.

Exportar resultados a XML

La aplicación permite exportar los resultados de los procesamientos a un fichero en formato XML para poder consultarlos más cómodamente. Simplemente deberemos escoger la ruta donde se creará el fichero y darle un nombre.

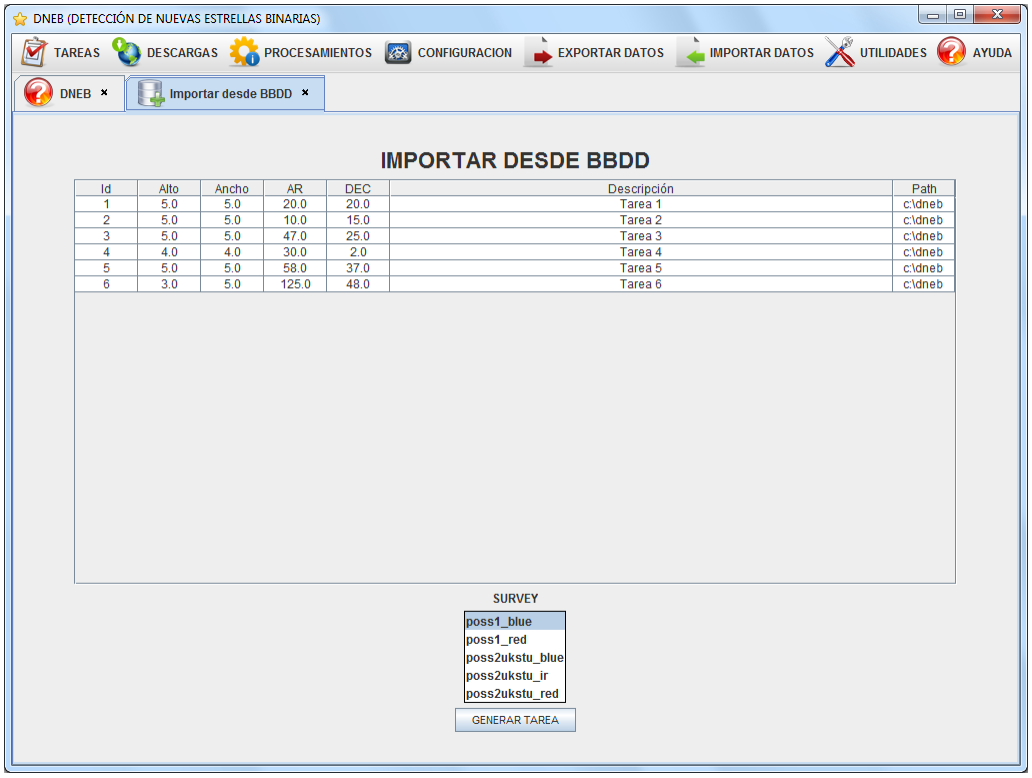


A esta opción se accede con *Exportar datos ->Exportar datos relevantes (XML).*

Importar desde base de datos

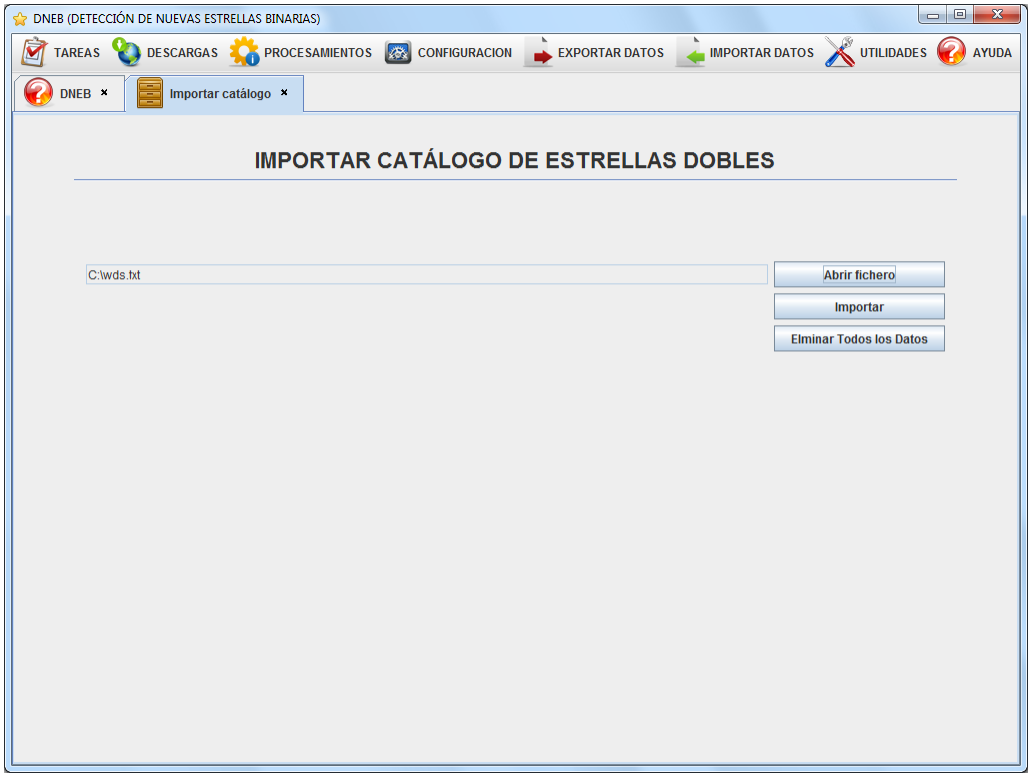
Accedemos a esta opción desde *Importar datos -> Desde BBDD.*

Nos permite generar tareas de descarga de imágenes en base a datos introducidos en la tabla CARGA_DATOS.



Importar catálogo

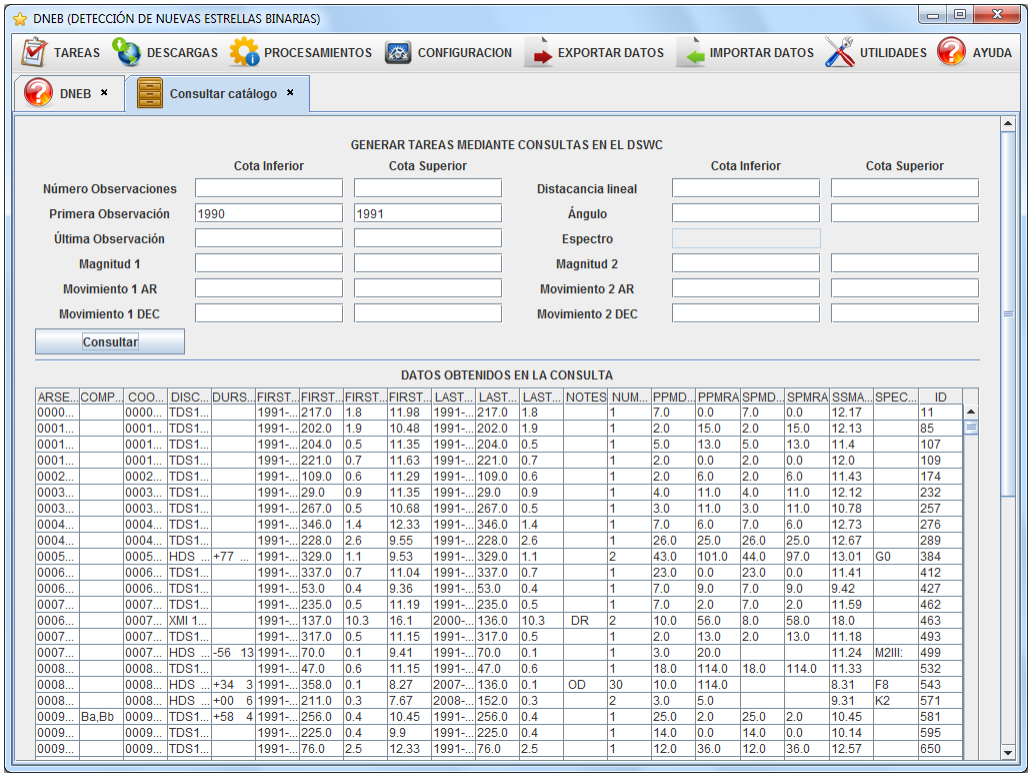
Para poder realizar consultas al catalogo, primero deberemos importarlo. Deberemos elegir el fichero de texto donde se encuentra el catalogo y pulsar en *Importar*. También podremos eliminar los datos de un catálogo anterior. El formato que debe tener el fichero de texto se puede consultar en *Ayuda -> Importar catálogo*.



Esta opción se encuentra disponible en *Importar datos -> Importar catálogo estrellas dobles*.

Consultar catálogo

Podemos consultar el catálogo que hayamos importado desde aquí y generar tareas con los resultados. Todos los campos que podemos rellenar son opcionales y actuarán a modo de filtro en la consulta al catálogo, pudiendo así consultar solo las estrellas que cumplan ciertos requisitos. Esta opción se encuentra disponible dentro del menú *Utilidades*.

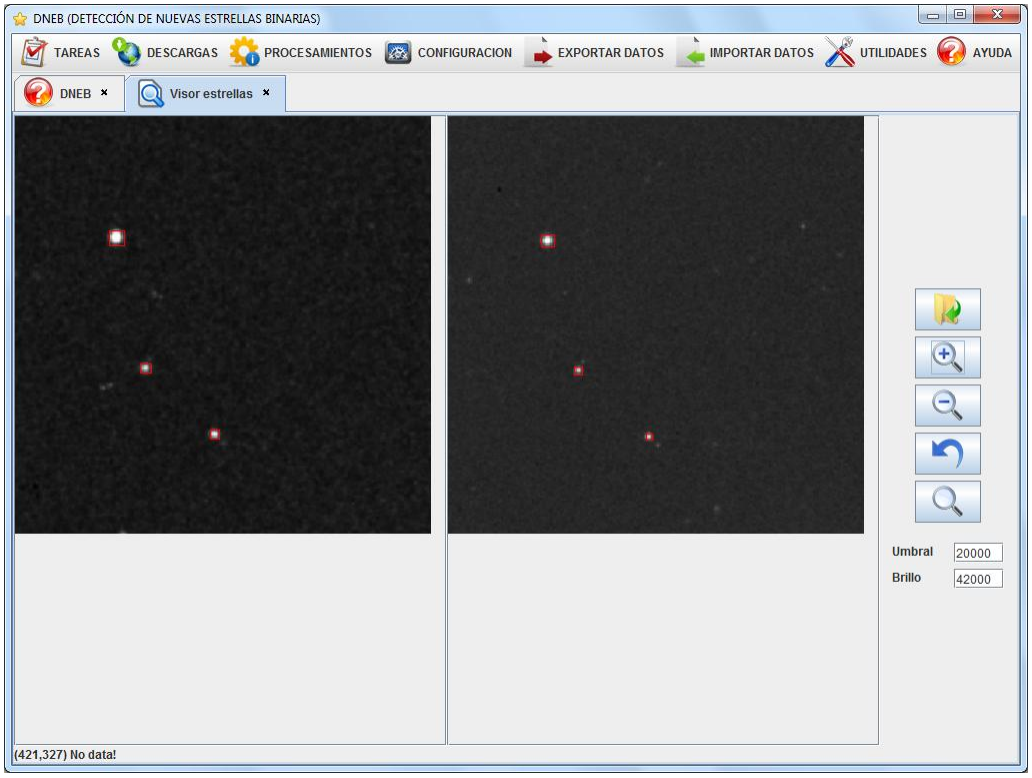


ARSE...	COMP...	COO...	DISC...	DURS...	FIRST...	FIRST...	FIRST...	FIRST...	LAST...	LAST...	LAST...	NOTES	NUM...	PPMD...	PPMRA	SPMD...	SPMRA	SSMA...	SPEC...	ID
0000...		0000...	TDS1...		1991-...	217.0	1.8	11.98	1991-...	217.0	1.8		1	7.0	0.0	7.0	0.0	12.17		11
0001...		0001...	TDS1...		1991-...	202.0	1.9	10.48	1991-...	202.0	1.9		1	2.0	15.0	2.0	15.0	12.13		85
0001...		0001...	TDS1...		1991-...	204.0	0.5	11.35	1991-...	204.0	0.5		1	5.0	13.0	5.0	13.0	11.4		107
0001...		0001...	TDS1...		1991-...	221.0	0.7	11.63	1991-...	221.0	0.7		1	2.0	0.0	2.0	0.0	12.0		109
0002...		0002...	TDS1...		1991-...	109.0	0.6	11.29	1991-...	109.0	0.6		1	2.0	6.0	2.0	6.0	11.43		174
0003...		0003...	TDS1...		1991-...	29.0	0.9	11.35	1991-...	29.0	0.9		1	4.0	11.0	4.0	11.0	12.12		232
0003...		0003...	TDS1...		1991-...	267.0	0.5	10.68	1991-...	267.0	0.5		1	3.0	11.0	3.0	11.0	10.78		257
0004...		0004...	TDS1...		1991-...	346.0	1.4	12.33	1991-...	346.0	1.4		1	7.0	6.0	7.0	6.0	12.73		276
0004...		0004...	TDS1...		1991-...	228.0	2.6	9.55	1991-...	228.0	2.6		1	26.0	25.0	26.0	25.0	12.67		289
0005...		0005...	HDS ...	+77 ...	1991-...	329.0	1.1	9.53	1991-...	329.0	1.1		2	43.0	101.0	44.0	97.0	13.01	G0	384
0006...		0006...	TDS1...		1991-...	337.0	0.7	11.04	1991-...	337.0	0.7		1	23.0	0.0	23.0	0.0	11.41		412
0006...		0006...	TDS1...		1991-...	53.0	0.4	9.36	1991-...	53.0	0.4		1	7.0	9.0	7.0	9.0	9.42		427
0007...		0007...	TDS1...		1991-...	235.0	0.5	11.19	1991-...	235.0	0.5		1	7.0	2.0	7.0	2.0	11.59		462
0006...		0007...	XMI 1...		1991-...	137.0	10.3	16.1	2000-...	136.0	10.3	DR	2	10.0	56.0	8.0	58.0	18.0		463
0007...		0007...	TDS1...		1991-...	317.0	0.5	11.15	1991-...	317.0	0.5		1	2.0	13.0	2.0	13.0	11.18		493
0007...		0007...	HDS ...	-56 13	1991-...	70.0	0.1	9.41	1991-...	70.0	0.1		1	3.0	20.0			11.24	M2III:	499
0008...		0008...	TDS1...		1991-...	47.0	0.6	11.15	1991-...	47.0	0.6		1	18.0	114.0	18.0	114.0	11.33		532
0008...		0008...	HDS ...	+34 3	1991-...	358.0	0.1	8.27	2007-...	136.0	0.1	OD	30	10.0	114.0			8.31	F8	543
0008...		0008...	HDS ...	+00 6	1991-...	211.0	0.3	7.67	2008-...	152.0	0.3		2	3.0	5.0			9.31	K2	571
0009...	Ba,Bb	0009...	TDS1...	+58 4	1991-...	256.0	0.4	10.45	1991-...	256.0	0.4		1	25.0	2.0	25.0	2.0	10.45		581
0009...		0009...	TDS1...		1991-...	225.0	0.4	9.9	1991-...	225.0	0.4		1	14.0	0.0	14.0	0.0	10.14		595
0009...		0009...	TDS1...		1991-...	76.0	2.5	12.33	1991-...	76.0	2.5		1	12.0	36.0	12.0	36.0	12.57		650

Visores

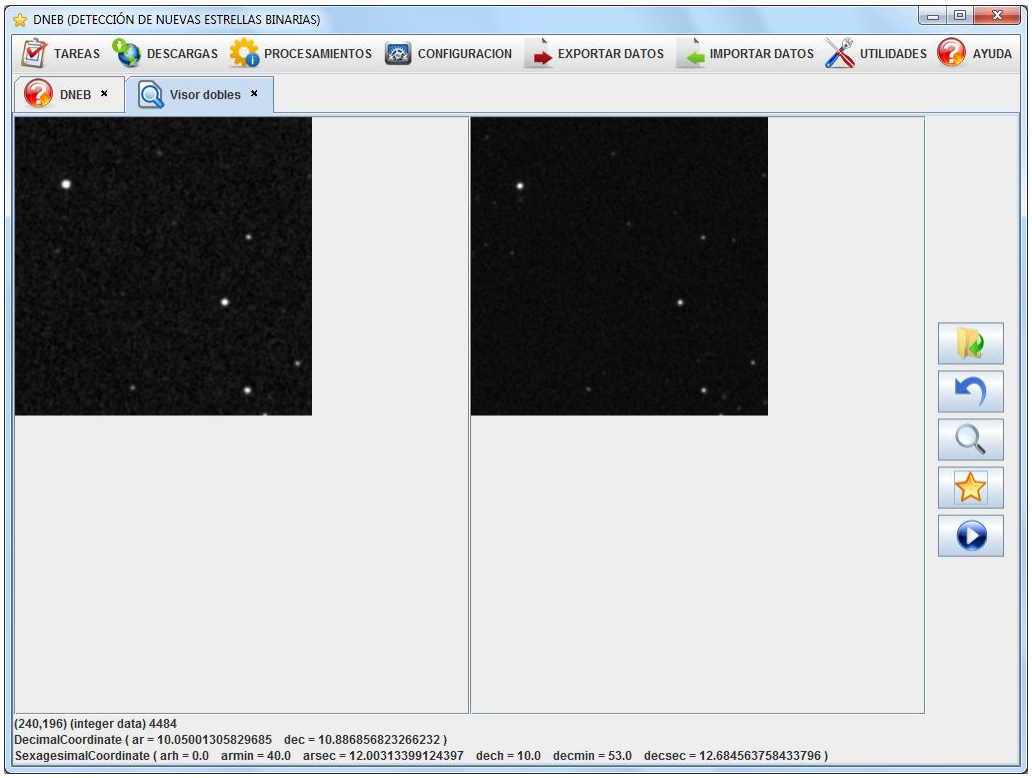
Los visores son herramientas que nos permiten visualizar imágenes en formato FITS y utilizar de manera visual alguna de las funcionalidades de la aplicación.

Visor de estrellas



Se accede desde el menú de utilidades y nos permite visualizar pares de imágenes, pudiendo hacer zoom de manera simultánea sobre las dos imágenes y usar el localizador de estrellas (Icono con lupa vacía) pudiendo configurar los parámetros de brillo y umbral. Además tiene un botón que permite deshacer todas las operaciones y restaurar las imágenes a las originales.

Visor de dobles



Se accede desde el menú de utilidades y nos permite realizar varias funcionalidades una vez cargamos dos imágenes de un mismo lugar y de épocas diferentes.

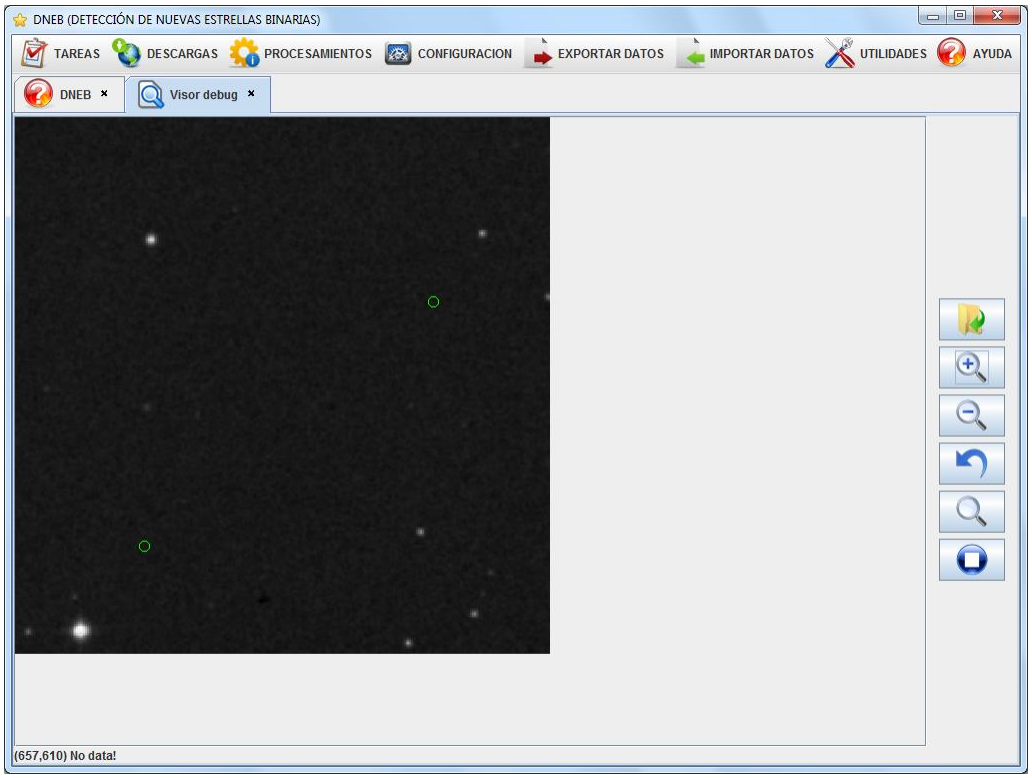
Buscar estrellas que se mueven (Icono lupa).

Buscar estrellas dobles (Icono estrella).

Realizar una animación basada en la superposición de las dos imágenes (Icono play).

Además tiene un botón que permite deshacer todas las operaciones y restaurar las imágenes a las originales.

El visor debug

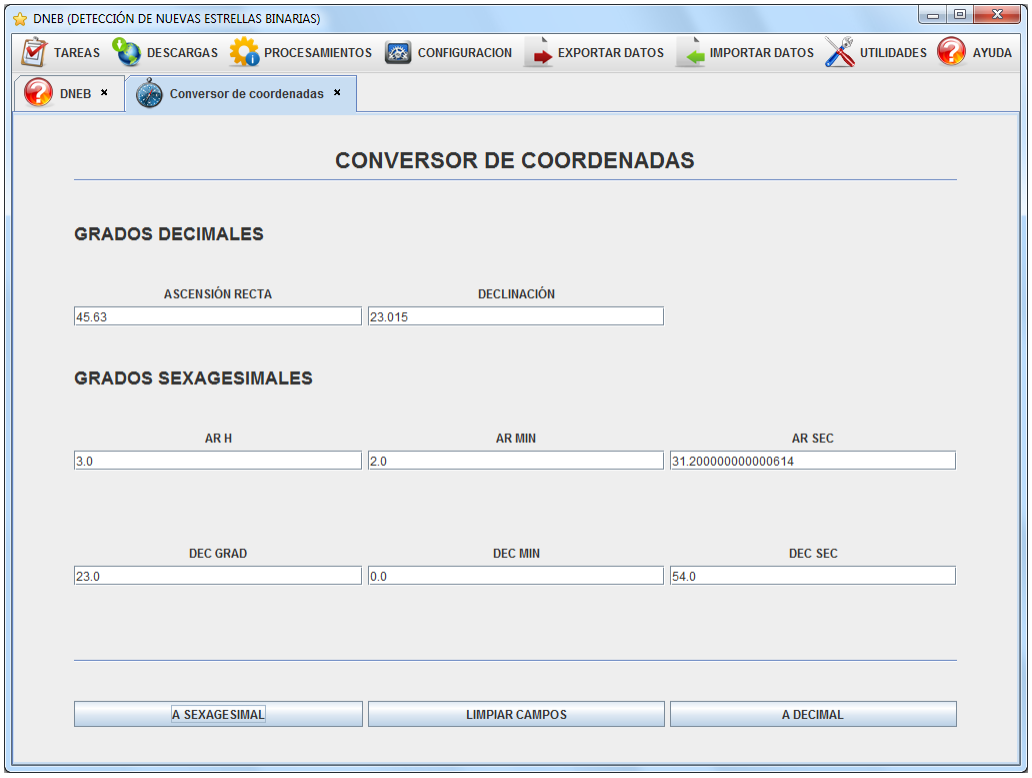


Nos permite localizar estrellas binarias ya conocidas, para ello hemos de abrir una imagen que haya sido descargada mediante una tarea generada al realizar una búsqueda en el catálogo de estrellas dobles y pulsar en el botón con la lupa.

Además nos permite calcular la distancia entre dos puntos, para ello pulsamos en el botón play, y posteriormente seleccionamos dos puntos pinchando sobre la imagen, nos aparecerá en una ventana la distancia entre los dos puntos.

Conversor de coordenadas

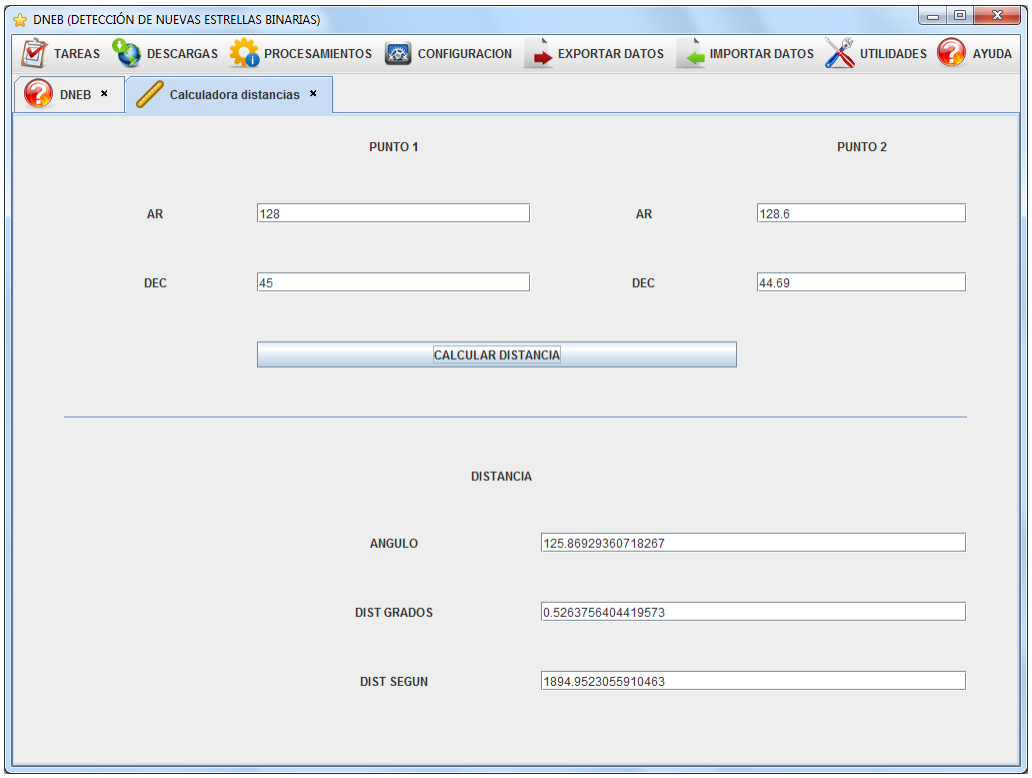
En *Utilidades -> Conversor de coordenadas* encontrados un sencillo conversor entre grados decimales y grados sexagesimales.



Cálculo de distancias

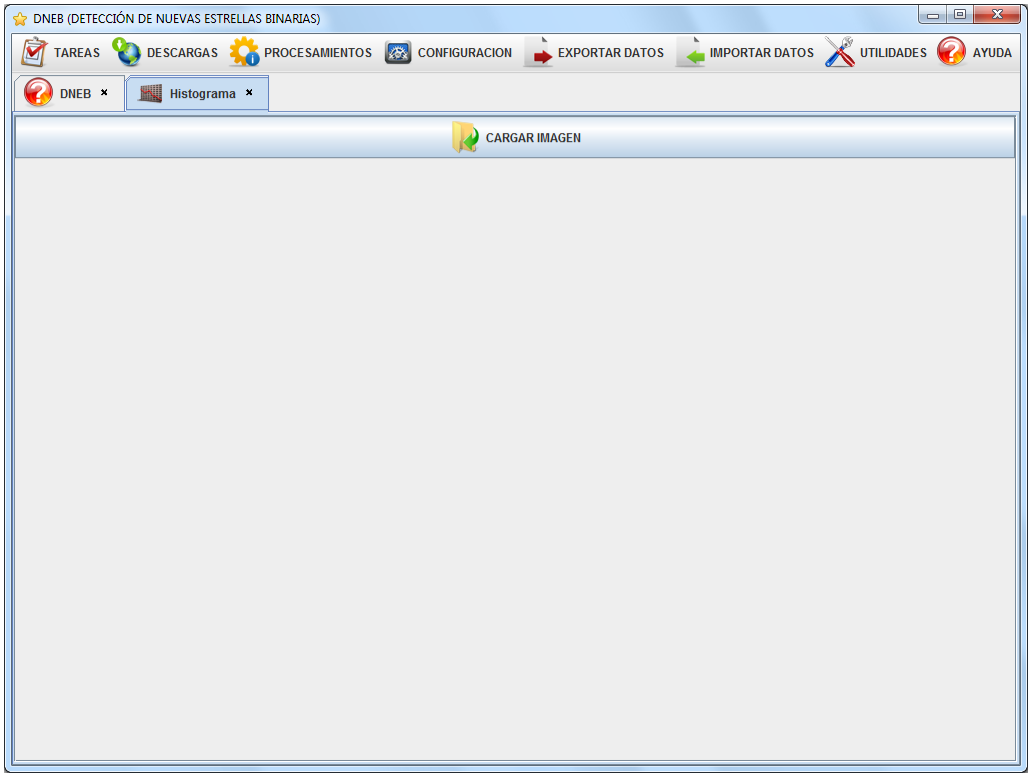
Con esta utilidad podremos calcular la distancia que hay entre dos puntos de la esfera celeste. Simplemente introducimos las coordenadas de los dos puntos y se nos mostrará el resultado tanto en grados como en segundos.

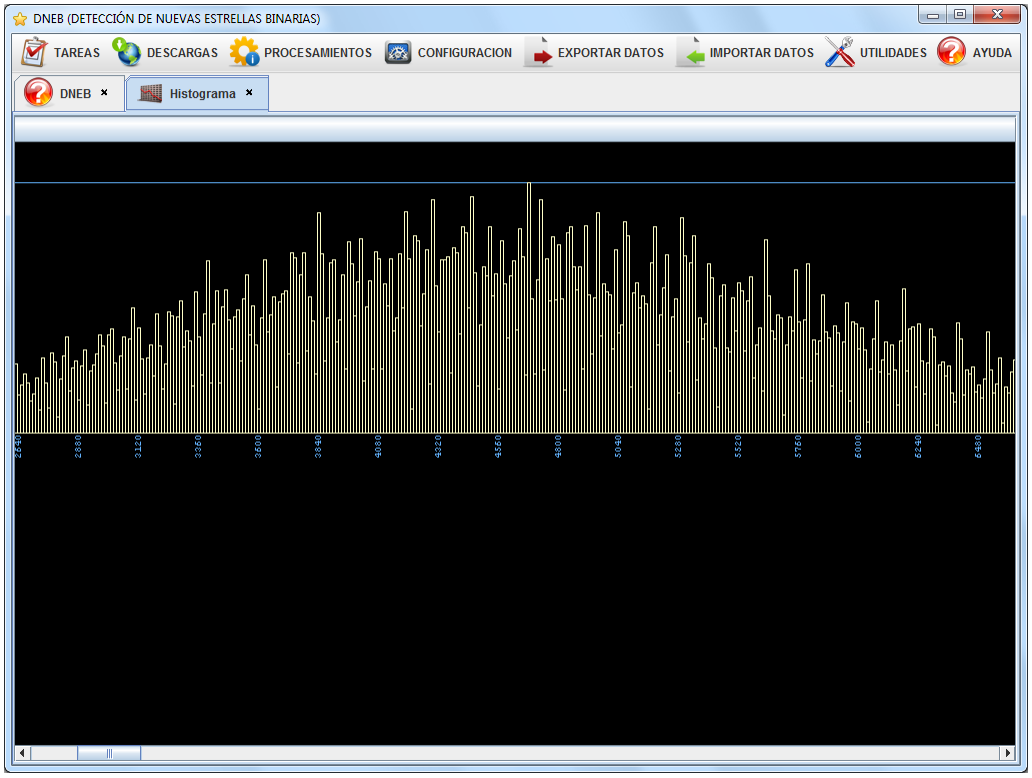
Esta opción se encuentra disponible en *Utilidades -> Cálculo distancias*.



Histograma

Podemos consultar el histograma de una imagen para poder ver de una manera rápida y sencilla los valores de intensidad que más se repiten en la imagen. Solo tendremos que pulsar en el botón de *cargar imagen* y seleccionar la imagen deseada para que el histograma aparezca.





La opción se encuentra en Utilidades -> Histograma.

Glosario

- **Ascensión Recta:** la ascensión recta es una de las coordenadas astronómicas que se utilizan para localizar los astros sobre la esfera celeste, equivalente a la longitud terrestre (coordenada geográfica). Se mide en horas, minutos y segundos, crece hacia el este y va desde 0h hasta 24h.
- **Centroide:** es el centro de masas del cuerpo material y el centro de gravedad del mismo. En nuestro caso cuando hablamos de centroide, nos referimos al centroide de una estrella.
- **Declinación:** la declinación es el ángulo que forma un astro con el ecuador celeste. Es comparable a la latitud geográfica, sólo que ésta se mide sobre el ecuador terrestre. Se mide en grados y va desde 90º en el polo norte hasta -90º en el polo sur.
- **Ecuador celeste:** el ecuador celeste es un gran círculo en la imaginaria esfera celeste en el mismo plano que el ecuador y, por tanto, perpendicular al eje de rotación de la Tierra. En otras palabras, es la proyección del ecuador terrestre en el espacio.
- **Esfera celeste:** esfera ideal, sin radio definido, concéntrica en el globo terrestre en la cual aparentemente se mueven los astros
- **Estrella binaria:** una estrella binaria es un sistema estelar compuesto de dos estrellas que orbitan mutuamente alrededor de un centro de masas común.
- **FITS:** FITS o Flexible Image Transport System es el formato de archivo más utilizado comúnmente en el mundo de la astronomía.
- **Procesamiento:** en la aplicación, operación que se realiza sobre una tarea, ya sea de búsqueda de nuevas estrellas binarias o de locaclización de la posición de estrellas ya conocidas.
- **Survey:** versión digital de varios atlas fotográficos del cielo nocturno.
- **Tarea:** en la aplicación, conjunto de imagenes que cubren una parcela determinada en la esfera celeste.
- **Washington Double Star Catalog:** el Catálogo de Estrellas Dobles de Washington (WDS) y está mantenido por la Observatorio Naval de los Estados Unidos.

Bibliografía

[CDS - Centre de Données astronomiques de Strasbourg]
http://cdsweb.u-strasbg.fr/

[FITS Data Format - NASA]
http://heasarc.gsfc.nasa.gov/docs/heasarc/fits.html

[Galadí-Enriquez 1998]
GALADÍ-ENRIQUEZ, David, RIBAS CANUDAS, Ignasi, Manual práctico de astronomía con CCD, Editorial Omega, primera edición, 1998

[JAMA: A Java Matrix Package]
http://math.nist.gov/javanumerics/jama/

[Java.net - Java Advances Imaging Stuff]
https://jaistuff.dev.java.net/

[Java Swing Tips - JProgressBar in a JTable Cell]
http://java-swing-tips.blogspot.com/2008/03/jprogressbar-in-jtable-cell.html

[Journal of Double Star Observations]
http://www.jdso.org/

[MAST - The Multimission Archive at STScI]
http://archive.stsci.edu/

[Mundo Java - La clase SwingWorker]
http://mundojava.blogspot.com/2006/12/jdk-16-la-clase-swingworker-con.html

[Nom.tam.fits Javadoc]
http://heasarc.gsfc.nasa.gov/docs/heasarc/fits/java/v1.0/javadoc/

[El observador de estrellas dobles]
http://elobservadordeestrellasdobles.wordpress.com/

[*The STScI Digitized Sky Survey*]
http://archive.stsci.edu/cgi-bin/dss_form

[The Washington Double Star Catalog]
http://ad.usno.navy.mil/wds/

[Wikipedia - Ascensión Recta]
http://es.wikipedia.org/wiki/Ascensión_recta

[Wikipedia - Declinación]
http://es.wikipedia.org/wiki/Declinación_(astronomía)

[Wikipedia - Ecuador Celeste]
http://es.wikipedia.org/wiki/Ecuador_celeste

[Wikipedia - Estrella Binaria]
http://es.wikipedia.org/wiki/Estrella_binaria

[Wikipedia - FITS]
http://es.wikipedia.org/wiki/FITS