



UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

PROYECTO DE FIN DE CARRERA EN
INGENIERÍA SUPERIOR EN INFORMÁTICA

Una herramienta de apoyo al tratamiento de catálogos astronómicos

Autores: Alicia Mireya Daza Castillo
Rosa María Rodríguez Navarro
Jorge González López

Director: Rafael Caballero Roldán

Madrid, Junio 2014

Alicia Mireya Daza Castillo, Rosa María Rodríguez Navarro, Jorge González López autorizamos a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes de los autores, la documentación y/o el prototipo desarrollado.

Fdo. Alicia Mireya Daza Castillo

Fdo. María Rodríguez Navarro

Fdo. Jorge González López

Índice de contenidos

ÍNDICE DE CONTENIDOS.....	IV
ÍNDICE DE FIGURAS.....	VI
RESUMEN	1
ABSTRACT.....	1
PALABRAS CLAVE.....	2
KEYWORDS	2
1. INTRODUCCIÓN	3
1.1. VIZIER.....	3
1.2. ESTRELLAS DOBLES.....	4
1.3. WASHINGTON DOUBLE STAR CATALOG	6
1.4. NUESTRA PROPUESTA.....	7
ESTRUCTURA DE LA MEMORIA	8
PARTE I	9
2. ANÁLISIS, DISEÑO E IMPLEMENTACIÓN.....	11
2.1. GESTIÓN DEL PROYECTO.....	11
2.1.1. Iteración 1.....	11
2.1.2. Iteración 2.....	12
2.1.3. Especificación y arquitectura	13
2.1.4. Implementación.....	15
2.1.5. Limitaciones y problemas encontrados	17
3. FUNCIONAMIENTO	19
PARTE II	26
4. ANÁLISIS, DISEÑO E IMPLEMENTACIÓN.....	28
4.1. GESTIÓN DEL PROYECTO.....	28
4.1.1. Iteración 1.....	28
4.1.2. Iteración 2.....	29
4.1.3. Iteración 3.....	29
4.1.4. Iteración 4.....	29
4.2. ESPECIFICACIÓN Y ARQUITECTURA.....	30
4.2.1. Especificación y arquitectura global	30
4.2.2. Especificación y arquitectura del parser	32
4.2.3. Especificación mandada al USNO	41
CATALOG SELECTION	41
COORDINATES.....	41
CATALOG DESCRIPTION.....	42
FILTERS.....	42
CRITERIA FOR SELECTING S ROWS.....	43
CRITERIA FOR DETECTING ERRORS.....	43

4.3.	IMPLEMENTACIÓN	44
4.3.1.	<i>Control de versiones y documentación</i>	44
4.3.2.	<i>Librerías y recursos</i>	44
4.4.	LIMITACIONES Y PROBLEMAS ENCONTRADOS	47
5.	FUNCIONAMIENTO	48
5.1.	CASO PRÁCTICO DE USO	55
6.	CONCEPTOS DE LA CARRERA UTILIZADOS	63
7.	CONCLUSIONES Y TRABAJO FUTURO	64
7.1.	CONCLUSIONES	64
8.	BIBLIOGRAFÍA	66

Índice de figuras

FIGURA 1. VIZIER	3
FIGURA 2. ESTRELLAS DOBLES: MOVIMIENTO	4
FIGURA 3. DIAGRAMA UML.....	13
FIGURA 4. PATRÓN OBSERVER.....	17
FIGURA 5. INTERFAZ	19
FIGURA 6. DESCARGA CATÁLOGO WDS	19
FIGURA 7. DESCARGA.....	20
FIGURA 8. INFORMACIÓN SOBRE LA DESCARGA.....	20
FIGURA 9. INFORMACIÓN DESCARGADA COMPLETADA	21
FIGURA 10. SELECCIÓN DE COORDENADAS	21
FIGURA 11. FILTRO POR CONSTELACIÓN.....	22
FIGURA 12. RESULTADOS.....	22
FIGURA 13. BÚSQUEDA POR CUALQUIER TEXTO.....	23
FIGURA 14. ALADIN SKY ATLAS.....	23
FIGURA 15. SLOAN DIGITAL SKY SURVEY.....	24
FIGURA 16. EXPORTACIÓN A EXCEL.....	24
FIGURA 17. NOTAS.....	25
FIGURA 18. EXPLORADOR DE PAQUETES	30
FIGURA 19. UML DEPENDENCIA ENTRE PAQUETES.....	30
FIGURA 20. UML PAQUETE VIEW	31
FIGURA 21. UML DEPENDENCIAS PRINCIPALES	32
FIGURA 22. UML PROGRAM.....	33
FIGURA 23. UML PAQUETE ELEMENTS.....	34
FIGURA 24. UML EXPRESIONES.....	35
FIGURA 25. UML FUNCIONES TRIGONOMÉTRICAS.....	35
FIGURA 26. UML OTRAS FUNCIONES.....	36
FIGURA 27. UML TIPOS.....	36
FIGURA 28. WINDOW BUILDER.....	45
FIGURA 29. NIVELES DEL LOGGER.....	46
FIGURA 30. INTERFAZ GRÁFICA	48
FIGURA 31. MENÚ EXPORT.....	48
FIGURA 32. MENÚ FUNCTIONS FOR FILTER.....	49
FIGURA 33. MENÚ PARSER.....	49
FIGURA 34. DETAILED GRAMMAR	50
FIGURA 35. DETAILED LEXICAL.....	50
FIGURA 36. MENÚ HELP	51
FIGURA 37. INFORMATION CATALOG SELECTION	51
FIGURA 38. EXAMPLE CATALOG SELECTION.....	51
FIGURA 39. INFORMATION CATALOG DESCRIPTION.....	52
FIGURA 40. EXAMPLE CATALOG DESCRIPTION	52
FIGURA 41. INFORMATION CRITERIUM FOR DETECTING ERRORS.....	52
FIGURA 42. EXAMPLE CRITERIUM FOR ERRORS.....	53
FIGURA 43. CATALOG SELECTION	53
FIGURA 44. CATALOG DESCRIPTION.....	53
FIGURA 45. FILTERS.....	54

FIGURA 46. CRITERIUM FOR DETECTING ERRORS.....	54
FIGURA 47. PASO 1.....	55
FIGURA 48. PASO 2.....	56
FIGURA 49. PASO 3.....	56
FIGURA 50. PASO 4.....	57
FIGURA 51. PASO 5.....	57
FIGURA 52. PASO 6.....	58
FIGURA 53. PASO 7.....	58
FIGURA 54. PASO 8.....	59
FIGURA 55. PASO 9.....	59
FIGURA 56. PASO 10.....	60
FIGURA 57. PASO 11.....	60
FIGURA 58. PASO 12.....	61
FIGURA 59. PASO 13.....	61
FIGURA 60. PASO 14.....	62
FIGURA 61. PASO 15.....	62

Resumen

El proyecto presenta dos herramientas destinadas a la extracción y combinación de información obtenida desde catálogos astronómicos disponibles online. En particular la primera herramienta está orientada a la consulta ágil del Washington Double Star Catalog (WDS), catálogo que contiene datos de estrellas dobles. La segunda herramienta permite descargar fragmentos de dos catálogos disponibles en el portal VizieR y especificar criterios que permitan filtrar y combinar uno a uno los datos de los objetos descargados. Se pretende que este proyecto sea de utilidad para la actualización permanente del WDS llevada a cabo por los astrónomos profesionales encargados de este catálogo.

Abstract

This project presents two software tools aimed at the extraction and combination of data obtained from the astronomical catalogs available online. In particular, the first tool is devoted to consult readily the Washington Double Star Catalog (WDS), a catalog specialized in data about double stars. The second tool allows the user to download fragments of two selected catalogs from the set available at the VizieR portal, and specify criteria for filtering and combining one-to-one the data representing each individual astronomical object. The purpose of the project is to constitute an useful tool for the permanent updates carried out by the professional astronomers in charge of the WDS.

Palabras clave

- Estrella doble
- Astronomía
- Astrometría
- Astrofísica
- Catálogo
- Pares ópticos
- USNO (Observatorio Naval de los Estados Unidos)
- WDS (Catálogo de Estrellas Dobles de Washington)
- SDSS (Sloan Digital Sky Survey)
- VizieR
- Aladin
- Filtrar
- Combinar
- Parser
- Coordenadas celestes
- Ascensión recta
- Declinación

Keywords

- Double Star
- Astronomy
- Astrometry
- Astrophysics
- Catalog
- Optical doubles
- USNO (United States Naval Observatory)
- WDS (Washington Double Star Catalog)
- SDSS (Sloan Digital Sky Survey)
- VizieR
- Aladin
- Filter
- Combine
- Parser
- Astronomical coordinates
- Right ascension
- Declination

1.Introducción

En este capítulo presentamos la motivación que nos ha llevado a desarrollar este trabajo. Para ello en primer lugar tenemos que introducir algunos conceptos que ayudan a comprender el contexto del que surge nuestro proyecto.

1.1. VizieR

En el campo de la astrofísica existen gran cantidad de catálogos astronómicos que recogen distintas características de los objetos (coordenadas, morfología, distancia, fotometría, etc.). En los últimos años han aumentado los esfuerzos para hacer estos catálogos disponibles de forma gratuita y en formato unificado. En particular el año 1993 el *Centre de Données astronomiques de Strasbourg (CDS)* creó el portal *VizieR*¹ con la idea de proporcionar un servicio de catálogos astronómicos que ser un marco de referencia para astrónomos de todo el mundo involucrados en la investigación, proporcionando acceso a nuevos datos con regularidad y publicados en diarios astronómicos.

En la actualidad, *VizieR* es conocido como el servidor de catálogos más extendido entre la comunidad de astrónomos, con más de 7300 catálogos registrados hasta la fecha, ganándose la importancia al ser la fuente de todos los datos catalogados dentro de la astronomía.



Figura 1. VizieR

¹ <http://vizier.u-strasbg.fr/>

El portal permite consultar los datos eligiendo el catálogo, creando filtros para limitar el número de filas obtenidas, e incluso ofreciendo la posibilidad de cruzar varios catálogos a partir de condiciones determinadas por el usuario. El formato de salida puede ser en un fichero, en texto plano o en HTML. A pesar de sus múltiples posibilidades y su importancia dentro de la investigación global de la astronomía, veremos que Vizier presenta algunas limitaciones que nosotros hemos intentando suplir, en la medida de lo posible, con el desarrollo de nuestra aplicación.

1.2. Estrellas dobles

En este trabajo estamos interesados en particular en catálogos que pueden ser de utilidad para el estudio de las *estrellas dobles*, parejas de estrellas que aparecen cercanas en el firmamento.

Naturalmente la cercanía de dos estrellas puede deberse simplemente a efecto de perspectiva: desde nuestra posición en la galaxia parecen encontrarse juntas, pero una está mucho más cercana que la otra y no guardan relación alguna. Se trata de los llamados *pares ópticos*, de poco interés para los astrónomos.

En cambio otras parejas corresponden verdaderamente a dos estrellas cercanas y que a menudo orbitan alrededor del centro común de masas. Podemos decir que se trata de sistemas con dos *soles*. A estas parejas se les llama sistemas binarios. Estos sistemas también pueden ser múltiples, existiendo la posibilidad de que existan tres, cuatro, cinco o incluso más estrellas interactuando entre sí.



Figura 2. Estrellas Dobles: Movimiento

Debido a la gran cantidad de estrellas dobles existentes en el Universo, los astrónomos necesitan desarrollar nuevas vías para distinguir las que son verdaderamente binarias de los pares ópticos.

Sin duda conocer la distancia de las dos componentes facilita mucho esta tarea, pero esta información solo está disponible en unos pocos casos. En el resto se utilizan métodos indirectos; por ejemplo estudios estadísticos (probabilidad de que se encuentren a esa distancia y/o que se muevan conjuntamente con un movimiento similar, o físicos (ver si las características físicas son compatibles con un sistema binario).

En contados casos es posible incluso *ver* cómo se mueve una estrella alrededor de la otra, es decir observar su órbita. Estos tipos de sistemas son de sumo interés para el astrónomo a partir de las órbitas se obtienen las masas de las estrellas, un dato muy relevante para entender la naturaleza física de estos objetos. Más aún, por las perturbaciones en la órbita se puede deducir, por ejemplo la presencia de materia oscura en las cercanías.

Sin embargo, en el caso de muchos sistemas abiertos (se llaman así a sistemas con una separación relativamente amplia, superior a 1-2 segundos de arco), una órbita completa puede tardar miles o incluso millones de años.

Por tanto cuando se descubre una par que puede ser una estrella binaria resulta muy importante apuntar con precisión las coordenadas de las dos estrellas, y repetir esta operación sistemáticamente cada poco tiempo, o al menos cada cierto número de años. Puede que al cabo de generaciones los datos recopilados hoy permitan determinar la órbita con exactitud.

Para registrar la astrometría (posición de las dos estrellas) se suele utilizar el siguiente sistema:

- Se registran las coordenadas de la primaria (la estrella más brillante, o en términos astronómicos la de *menor* magnitud). Las coordenadas astronómicas se representan mediante la pareja de números (ascensión recta, declinación) que pueden verse como los análogos en el cielo a la longitud, latitud sobre la esfera terrestre.
- Se registra la posición relativa de la secundaria (la estrella menos brillante) con respecto a la primaria, apuntando en particular la distancia o separación (en segundos de arco) y el ángulo que forman (en grados con respecto al polo norte celeste).

1.3. Washington Double Star Catalog

Como hemos mencionado anteriormente, *VizieR* contiene información de multitud de catálogos, entre los que en particular nos interesa el *Catálogo de Estrellas Dobles de Washington*, conocido también por sus siglas en inglés *WDS* (*Washington Double Star Catalog*).

El *Washington Double Star Catalog*², nombrado en el resto del trabajo mediante sus siglas *WDS*, es un catálogo astronómico de estrellas dobles, mantenido por el *Observatorio Naval de los Estados Unidos (USNO)*, que contiene posiciones, magnitudes, movimientos propios y tipos espectrales de más de 100.000 estrellas dobles y múltiples.

A diferencia de otros catálogos disponibles en *VizieR* este catálogo se actualiza casi a diario con nuevos datos de astrometría correspondiente a estrellas dobles.

Un aspecto importante son las medidas de magnitud, astrometría, posición precisa, etc., que en ocasiones pueden contener datos erróneos por medidas incorrectas o por la dificultad inherente a estos sistemas (resulta muy difícil diferenciar el brillo de dos estrellas próximas). Para detectar estas anomalías interesa *cruzar* el *WDS* con otros catálogos que puedan ayudar a corregir los datos poco fiables.

Sin embargo, las posibilidades que ofrece *VizieR* limitan a menudo la actualización de este catálogo. En efecto, a pesar de las numerosas posibilidades que ofrece este servicio, sus consultas resultan muy costosas, tanto computacionalmente como para el usuario, e ineficientes. Aunque el portal permite la utilización del lenguaje de consultas SQL, lo que imposibilita la comprobación de los detalles intermedios que se van generando en dichas consultas al carecer de *transparencia*.

² <http://www.usno.navy.mil/USNO/astrometry/optical-IR-prod/wds/WDS>

1.4. Nuestra propuesta

Nuestra propuesta será la de automatizar la recogida de datos, el filtro para reconocer una estrella del *WDS*, o de cualquier catálogo en otro y facilitar las consultas dividiéndolas en distintas fases que nos permitan ir filtrando la información.

Con este propósito hemos desarrollado dos aplicaciones que nos permitan:

- Realizar consultas de diferentes catálogos.
- Cruzar y combinar la información de los mismos.
- Filtrar de manera sencilla y eficiente los datos obtenidos a partir de diferentes condiciones seleccionadas por el propio usuario, trabajando tanto con estrellas primarias como estrellas secundarias.
- Generar un log con toda la información detallada de las diferentes fases del proceso.

La primera aplicación está pensada exclusivamente para la descarga y consulta del catálogo *WDS*. Permite buscar filas de dicho catálogo por coordenadas, constelaciones o por cualquier texto. La búsqueda se realiza a partir de las coordenadas celestes. Los resultados de las búsquedas están pensados para agilizar el tratamiento y estudio de las estrellas dobles por parte la comunidad astronómica.

La segunda aplicación permite al usuario descargar, de forma simultánea, dos de los catálogos alojados en *VizieR* para, posteriormente, especificar unos criterios de filtrado y combinación sobre dichos catálogos. Al contrario que la primera aplicación, ésta estará enfocada a un ámbito profesional.

Estructura de la memoria

La memoria se divide en dos partes, una para cada aplicación. Internamente ambas comparten la misma estructura:

- Un capítulo inicial describiendo el análisis y diseño de la herramienta
- Un capítulo destinado a describir la implementación: tecnologías utilizadas, problemas encontrados, etc.
- Cada parte contiene un capítulo describiendo su funcionalidad.
- Las dos partes se cierran con un capítulo común de conclusiones y trabajo futuro.

Parte I

Consulta del catálogo WDS

2. Análisis, diseño e implementación

2.1. Gestión del proyecto

En el presente apartado comentaremos las herramientas de apoyo para la organización, gestión y desarrollo del proyecto.

Para la realización de esta primera parte del proyecto y para poder cumplir con los objetivos establecidos anteriormente, realizamos una división del mismo en dos iteraciones.

2.1.1. Iteración 1

En esta primera iteración nos marcamos los siguientes objetivos:

- ✓ Familiarizarnos con el contexto de la astronomía y las estrellas dobles.
- ✓ Análisis de las herramientas existentes utilizadas en astronomía.
- ✓ Búsqueda de las herramientas de desarrollo necesarias a lo largo del proyecto.
- ✓ Estudio de los campos en los que se estructura el catálogo *WDS* así como el modo en el que se ordenan sus coordenadas.
- ✓ Comprobar la posibilidad de descargarnos dicho catálogo a partir de una URL específica.

Para poder desarrollar estos objetivos, comenzamos buscando información relacionada con las estrellas dobles que nos permitiera familiarizarnos con la terminología utilizada en astronomía, para así entender cómo son estudiadas por los astrónomos a partir de sus coordenadas.

Siguiendo las indicaciones de nuestro director de proyecto, estuvimos estudiando los usos y limitaciones de las herramientas actualmente existentes, centrándonos principalmente en el software *Aladin*, para posteriormente adaptar e integrar en nuestro proyecto dichas herramientas, mejorando de este modo las características que podíamos llegar a darle a nuestra aplicación de consulta.

Una vez teníamos definida la dirección de nuestro proyecto, la decisión que tomamos, dadas las necesidades de nuestra aplicación y nuestra familiarización con las herramientas, fue la de utilizar el lenguaje de programación Java y su entorno de desarrollo Eclipse. En un principio se estuvo barajando la posibilidad de integrar alguna herramienta que permitiera el estudio y catalogación de imágenes. Finalmente decidimos integrar la herramienta *Aladin* en nuestro proyecto para exprimir al máximo el potencial que ofrece de tratamientos de imágenes a partir de unas coordenadas dadas.

Con la ayuda de Rafael Caballero, dedicamos varios días al estudio del catálogo *WDS* para identificar los conceptos que definían cada uno de los campos y sus valores. Este proceso fue una de las labores más complicadas ya que implicaba comprender conceptos hasta ahora desconocidos para nosotros.

Para que la aplicación pudiese llevarse a cabo, nuestro principal objetivo era entender la forma en la que el catálogo se encontraba alojado para su consulta en el portal del *US Naval Observatory*³. Comprobamos que dichos datos se podían consultar a partir de una URL que, en principio, no varía a pesar de las modificaciones y actualizaciones que dicho catálogo sufra. Esta primera fase del proyecto consistía en el desarrollo de la aplicación la cual, a partir de esta URL, descargase la información en un archivo de texto.

2.1.2. Iteración 2

En esta segunda iteración, nuestros objetivos fueron:

- ✓ Recorrido del catálogo para su posterior consulta
- ✓ Integración de las herramientas de tratamiento de imágenes
- ✓ Integración del API Excel
- ✓ Depuración de la aplicación
- ✓ Comenzar a escribir la memoria

Una vez descargado el catálogo, el siguiente paso era recorrer y leer la información descargada para poder mostrarla en nuestra aplicación en función de los filtros que estableciera el usuario.

Como comentábamos anteriormente, decidimos implementar un script para lanzar la herramienta *Aladin* según las coordenadas establecidas por el usuario, de forma que pueda consultar las imágenes astronómicas digitalizadas del objeto celeste denominado por dichas coordenadas del catálogo. De forma análoga, se realiza la implementación de otro script que conecta la aplicación con el *Sloan Digital Sky Survey (SDSS)*, mostrando las imágenes de pequeñas porciones del cielo procesando listas de objetos observados, así como diversos parámetros asociados a dichas coordenadas.

Debido a la gran cantidad de información que se puede llegar a manejar, se integró la API de Excel, de forma que estos resultados pudieran ser exportados a una hoja de cálculo en formato .xls para agilizar su visualización y tratamiento.

Una vez finalizada la aplicación, procedimos a la depuración del código y a la corrección de diversos errores de funcionamiento.

Finalmente se comenzó a redactar la memoria con toda la información.

³ <http://ad.usno.navy.mil/proj/WDS/Webtextfiles/wdsnewframe1.html>

2.1.3. Especificación y arquitectura

A continuación mostramos la estructura de composición de la primera aplicación.

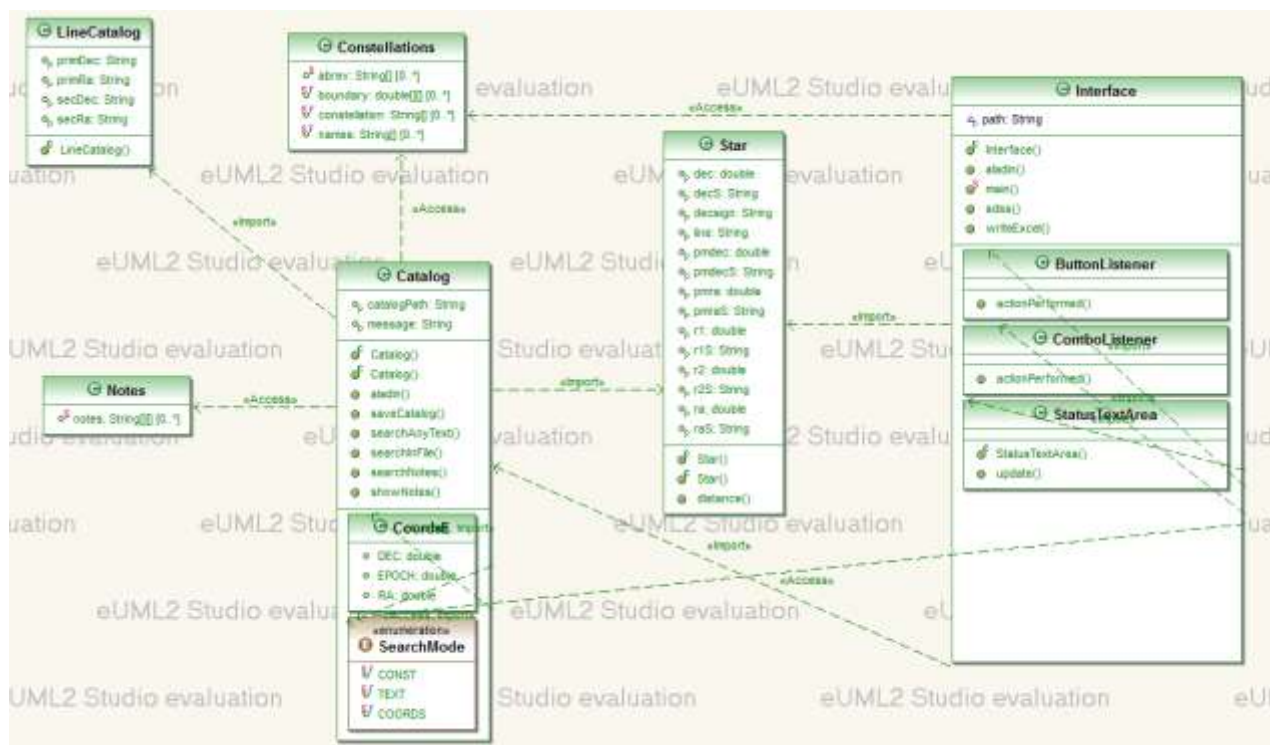


Figura 3. Diagrama UML

Como se observa, la aplicación consta de un único paquete *main*, donde se encuentran las siguientes clases:

- *Interface*: extiende de *JFrame* y se encarga de generar toda la interfaz gráfica de la aplicación, así como la funcionalidad de cada uno de sus botones. Implementa los métodos que conectan con el portal *SDSS*, el que ejecuta *Aladin*, los que manejan la API Excel para importar los datos y el que genera notas sobre la coordenada seleccionada. Consta de las siguientes clases auxiliares:
 - *StatusTextArea*: extiende de la clase *JTextArea* e implementa el patrón *Observer*. Define el tratamiento que se le da a los cuadros de texto de la aplicación, sus fuentes y estilos.
 - *ComboListener*: implementa la interfaz *ActionListener* para manejar los eventos de acción del *JComboBox* en el que podemos filtrar seleccionando una opción de la lista de constelaciones existente.
 - *ButtonListener*: implementa la interfaz *ActionListener* para el manejo de los eventos de acción de cada uno de los botones existentes en la interfaz de usuario.

- *Catalog*: extiende de la clase *Observable*, permitiendo que sea monitorizada por otras clases. En esta clase se encuentran tanto el manejo del catálogo como el método que permite a la aplicación ejecutar el software *Aladin*.
- *Constellations*: contiene la información que hace referencia al nombre y localización de cada una de las constelaciones.
- *LineCatalog*: clase utilizada para describir la información que se necesita de cada línea del catálogo. En este caso, se trata de la ascensión recta y la declinación tanto de la estrella primaria como de la secundaria.
- *Notes*: notas relacionadas con cada una de las coordenadas alojadas en el catálogo.
- *Star*: define información sobre la posición de una estrella concreta. Aquí guardamos la información de la ascensión recta y la declinación. La ascensión recta la separamos en horas, minutos y segundos, mientras que la declinación la separamos en grados, minutos y segundos. Así podemos calcular la distancia entre dos estrellas.

2.1.4. Implementación

La implementación del proyecto se ha realizado en su totalidad utilizando el lenguaje *Java* y el entorno de programación *Eclipse Platform*, compuesto por un conjunto de herramientas de programación de código abierto y multiplataforma. En concreto, la versión utilizada ha sido *Eclipse Kepler*.

Cada semana se mostraba una nueva versión de la aplicación al director de proyecto. Para ello, generábamos un *Runnable JAR File*, el cual permitía ejecutar la aplicación en cualquier equipo sin necesidad de utilizar ningún entorno de programación.

2.1.4.1. Control de versiones y documentación

Para llevar a cabo el control de versiones se hizo uso del *Subversion Tortoise SVN* y el repositorio gratuito *Google Code*, permitiéndonos trabajar de manera individual sin que los cambios del resto de los compañeros del proyecto se viesen afectados. A pesar de esto, y dada la complejidad de algunos desarrollos, solíamos reunirnos varias veces a lo largo de la semana para avanzar de manera conjunta.

La documentación del proyecto se ha estructurado, organizado y centralizado a través de la plataforma en la nube *Google Drive*, permitiéndonos almacenar, crear, compartir y acceder a nuestros archivos y carpetas desde cualquier dispositivo.

2.1.4.2. Librerías y recursos

Para el desarrollo de esta aplicación se usó la librería *jxl.jar*, el API de *JExcel*. Esta interfaz de programación de aplicaciones es *Open Source* lo cual permite a los desarrolladores leer, escribir y modificar hojas de cálculo *Excel* de una manera rápida, eficaz y ordenada.

La amplia difusión de este tipo de documentos, hacen de *Excel* uno de los formatos más reconocidos de intercambio de información de datos en todas las áreas, desde el ámbito laboral, hasta el educacional. Así mismo, se pueden diseñar entornos gráficos para *Excel* que hagan más amigable la presentación de datos utilizando las conocidas herramientas gráficas de *Java*.

Con la utilización de esta librería se pretendió que la presentación de la información fuera lo más sencilla posible, a la par que automatizada. En este sentido, esta API de *Java* presenta un gran potencial para llevarlo a cabo.

El siguiente fragmento de código⁴ *Java* muestra la base de la API desde la cual hemos partido para exportar nuestros resultados a una hoja de cálculo:

⁴ <http://www.nosolunix.com/2010/05/escribir-y-leer-archivos-excel-en-java.html>

```
import java.io.File;
import jxl.*;
import jxl.write.NumberFormat;
import jxl.write.WritableCell;
import jxl.write.WritableCellFormat;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;

public class ExcelWrite {

    public void writeExcel(String excel_file, String sheet_name, int row, int
column, Double value)
    {
        String cellData = new String();
        try{
            Workbook target_workbook = Workbook.getWorkbook(new File(excel_file));
            WritableWorkbook workbook = Workbook.createWorkbook(new File(excel_file),
target_workbook);
            target_workbook.close();
            WritableSheet sheet = workbook.getSheet(sheet_name);
            jxl.write.Number number = new jxl.write.Number(column, row, value);
            sheet.addCell(number);
            workbook.write();
            workbook.close();
        }
        catch(Exception e)
        {
            System.out.println("writeExcel ->" + e);
        }
    }
}
```


2.1.5. Limitaciones y problemas encontrados

Durante el desarrollo de esta aplicación, nos encontramos con diversos problemas, relacionados con la dependencia de la conexión a internet y la visualización de los resultados.

Los resultados de la búsqueda no se mostraban en la interfaz a medida que los íbamos encontrando en el catálogo. Como se trata de una aplicación que consta de un único flujo de trabajo, al recorrer el fichero realizando la búsqueda de las filas que se correspondían con los criterios, no conseguíamos mostrar la información al usuario hasta que no procesábamos el catálogo completo.

Para solucionarlo, nos planteamos la posibilidad de crear varios hilos, es decir, ir mostrando la información en la interfaz de forma concurrente a medida que se realizaba la búsqueda. Sin embargo, existe un patrón de diseño que nos permitía ir mostrando cada resultado según se encontraba, sin necesidad de programación concurrente. Se trata del patrón *Observer*:

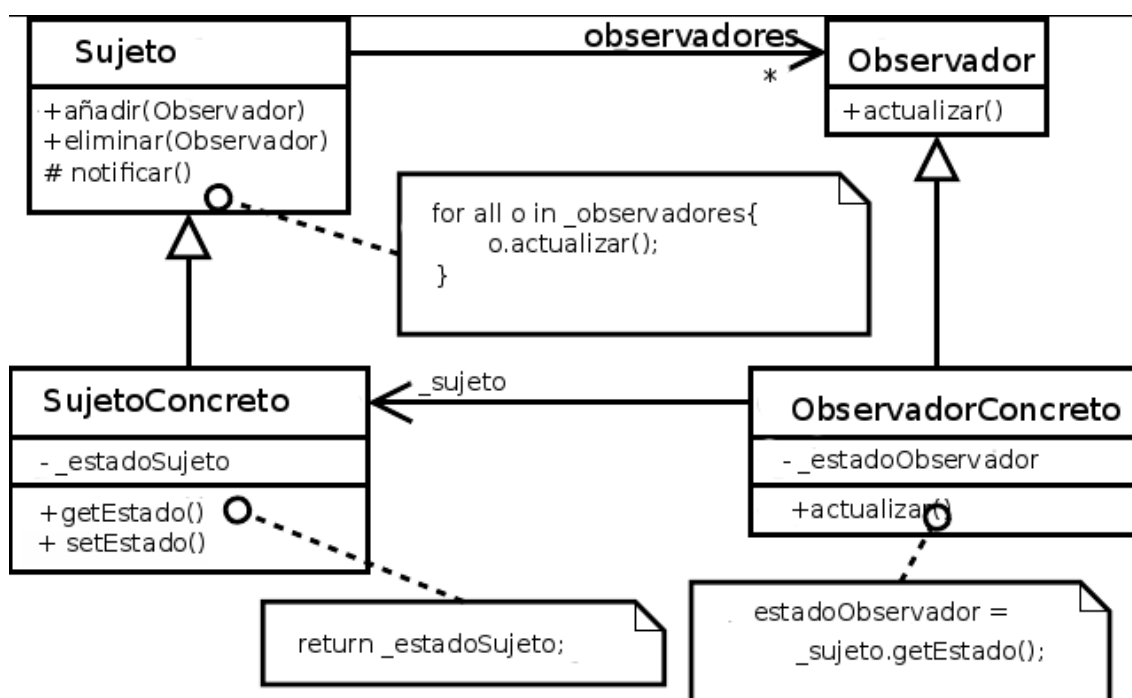


Figura 4. Patrón Observer

Dicho patrón⁵ permite implementar una estrategia que reaccione a los cambios de estado del objeto observado, definiendo una dependencia del tipo *uno-a-muchos* entre objetos. De esta forma, cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes.

⁵ [http://es.wikipedia.org/wiki/Observer_\(patr%C3%B3n_de_dise%C3%B1o\)](http://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o))

Así, el observable sería nuestra clase *Catalog*, la cual se encargaría de buscar la información en el catálogo. Por otra parte, el *StatusTextArea* sería el observador, el cual se encargaría de actualizar tanto el panel de resultados como al consola de estado de la aplicación. Usando este patrón, conseguimos que los resultados se visualizasen según la aplicación los iba encontrando.

En cuanto a la forma de mostrar la información, se observó que mostrar los resultados de las búsquedas únicamente en un panel de texto podría no ser suficiente, sobre todo cuando dichos resultados son muy extensos.

Para solucionarlo, añadimos la posibilidad de exportar dichos resultados a una hoja de cálculo en formato *Excel* utilizando la librería *jxl.rar* explicada anteriormente.

Debido al tamaño del catálogo del *WDS*, llegamos a plantearnos que la eficiencia de la aplicación pudiera verse afectada a la hora de descargar la información, principalmente con conexiones a la red de baja velocidad.

En el futuro, un posible problema que nos puede surgir tiene relación con la descarga del catálogo. Al existir una dependencia con una URL específica, la aplicación podría verse afectada por la modificación de dicha dirección, impidiendo de este modo la descarga de las actualizaciones que en él se produzcan, así como por la ausencia de una conexión a Internet. Cabe destacar que unas de las funcionalidades de la aplicación es cargar dicho catálogo desde el propio ordenador del usuario, por lo que los inconvenientes relacionados con la URL del catálogo no excluyen por completo el uso del programa.

Las búsquedas pueden realizarse por coordenadas, constelación o por cualquier texto introducido por el usuario. Al tratarse de una aplicación orientada a un único catálogo, una posible extensión de la misma sería permitir la búsqueda por más campos, centrándonos en aquellos que aparecen exclusivamente en el *WDS*.

Finalmente destacar que el desarrollo de esta aplicación necesitó más tiempo del previsto inicialmente. Estaba pensada como un inicio a la segunda. Sin embargo, según trabajábamos, nos dimos cuenta de que no podíamos reutilizarla directamente para el comparador de catálogos, se fueron añadiendo distintas funcionalidades que acabaron haciéndola única para su propósito.

3. Funcionamiento

Aspecto de la interfaz gráfica.

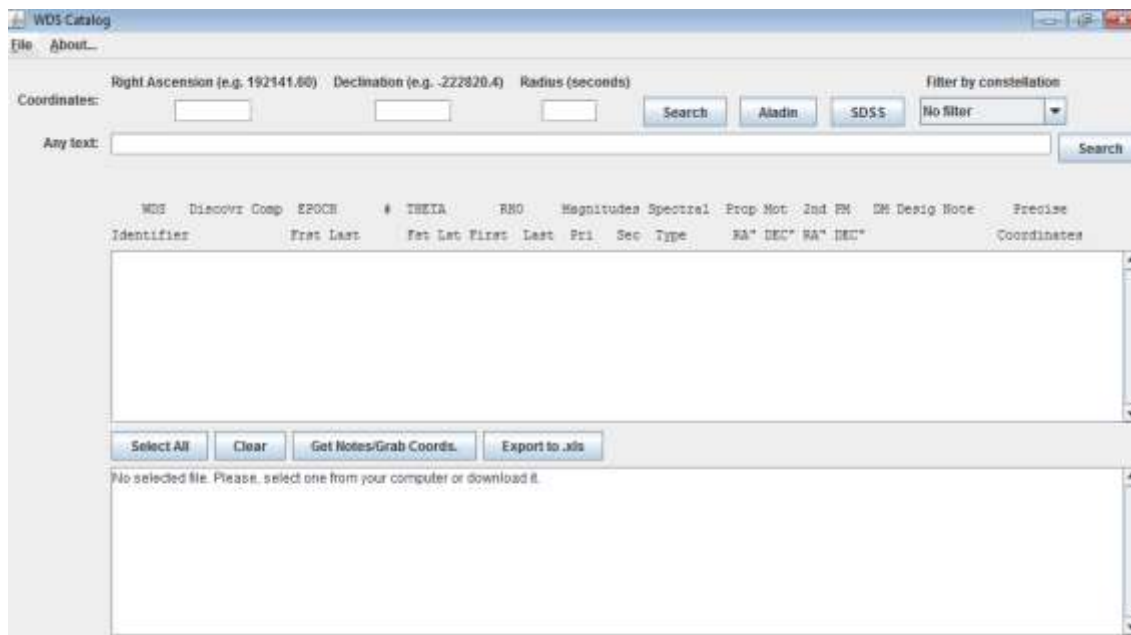


Figura 5. Interfaz

En primer lugar, se descarga el catálogo WDS. Para ello, en la pestaña *File* se selecciona *Download WDS Catalog*.

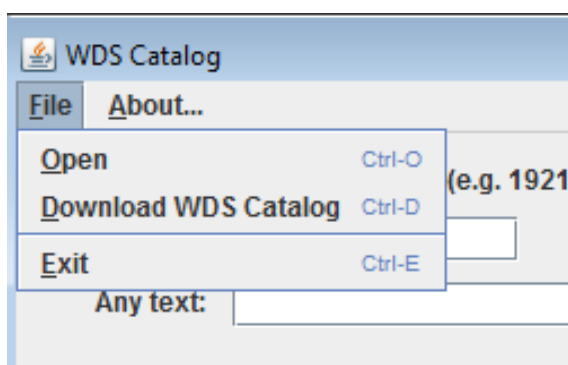


Figura 6. Descarga catálogo WDS

A continuación se indica el nombre del catálogo, la ruta donde queremos guardarlo (por defecto nos aparecerá la ruta desde la cual hemos ejecutado la aplicación).

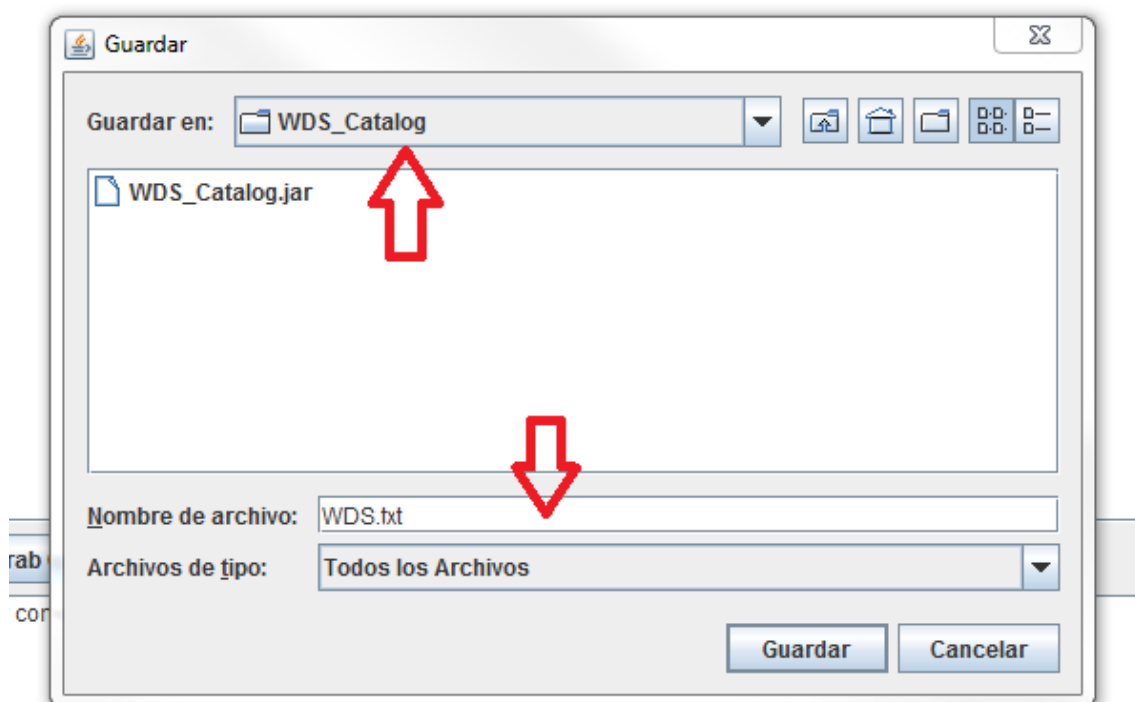


Figura 7. Descarga

Una vez iniciada la descarga del catálogo, nos aparecerá un mensaje indicando su estado, la URL a la que se ha conectado para realizarla, la ruta que hemos seleccionado y el tamaño del fichero una vez se haya descargado.

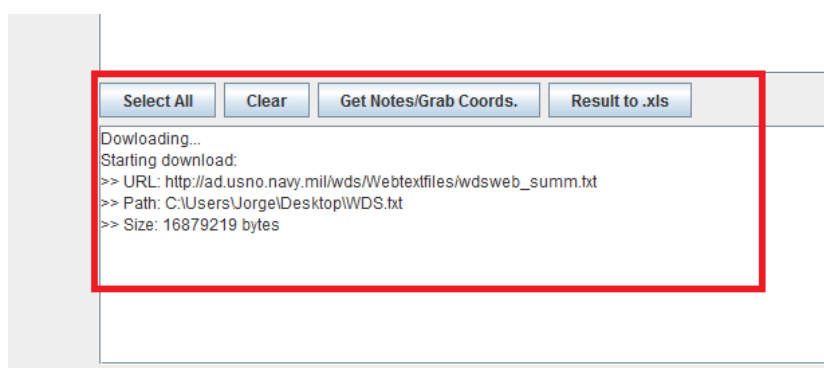


Figura 8. Información sobre la descarga

Finalizada la descarga, el programa nos mostrará un nuevo mensaje notificándolo, junto con la ruta definida.

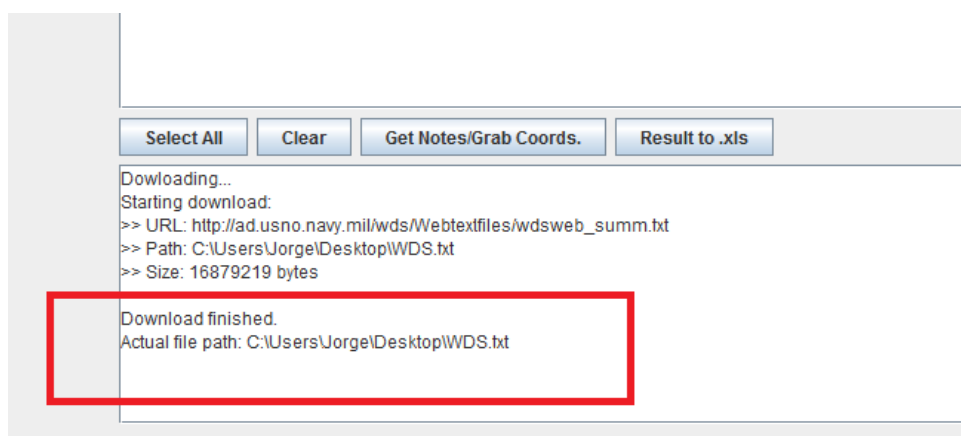


Figura 9. Información descargada completada

La descarga del catálogo únicamente será necesaria la primera vez que se ejecute la aplicación o en el caso de que queramos actualizar nuestro fichero con las últimas actualizaciones disponibles en el *WDS*. Si ya disponemos de un fichero con el catálogo y no consideramos necesaria su descarga, seleccionaremos *Open* en el mismo menú *File* y se cargará el archivo a la aplicación.

Para realizar una búsqueda por proximidad sobre el catálogo, el usuario introducirá las coordenadas de la ascensión recta, la declinación y el radio en torno al cual realizar dicha búsqueda, o bien seleccionar una constelación concreta en el menú desplegable *Filter by constellation*.

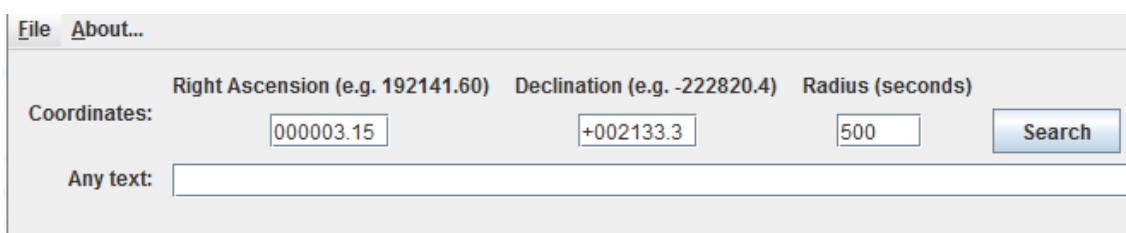


Figura 10. Selección de coordenadas

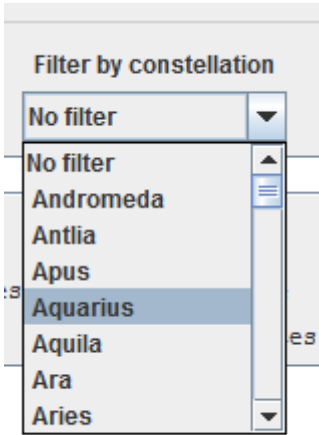


Figura 11. Filtro por constelación

Una vez pulsado el botón *Search*, la aplicación mostrará los resultados del WDS que correspondan con los datos introducidos previamente por el usuario. De esta forma se reducen de manera significativa los resultados que serán objeto de estudio.

En el caso de la búsqueda por constelación, los resultados serán mayores debido a que únicamente estamos filtrando los resultados basándonos en dicho criterio.

Los resultados se mostrarán usando el mismo formato y mostrando los mismos campos que podemos encontrar en el propio catálogo del WDS:

WDS	Discover	Comp	EPOCH	#	THETA	RND	Magnitudes	Spectral	Prop	Hot	2nd	PM	SN	Design	Note	Precise
Identifier			Year		Lat	First	Last	Par	Sec	Type	RA°	DEC°	RA°	DEC°		Coordinates
00001+00225KF1060			2001 2006	3	187 185	2.1	2.1 20.1	21.9	M3		-003+000					000003.13+002133.3
00002+00145KF1061			2001 2004	2	174 174	1.4	1.4 17.3	18.0	M0		-011+027					000013.00+001404.9
00002-00211TF 44			1951 2006	5	16 28	3.9	3.8 13.0	15.8	G1V+K3V		+009-011	-013-033			V	000010.39-002017.6

Select All

Clear

Get Notes/Graph Coords.

Result to .xls

Search finished

Figura 12. Resultados

Ahora que tenemos filtrados los resultados según el criterio deseado podemos comenzar a trabajar con ellos.

Con el objetivo de agilizar y facilitar aún más el tratamiento de la información, es posible realizar búsquedas *concretas* introduciendo el texto deseado en el campo *Any text*. Mediante esta opción, la aplicación filtrará la información basándose únicamente en el texto introducido, lo que nos permitirá ampliar el rango de búsqueda a campos específicos, como pueden ser el año de observación de dicha estrella, su brillo o la persona que la descubrió, entre otros. El texto buscado aparecerá resaltado en color rojo.

WDS		Discover	Comp	EPOCH	#	THETA	RBD	Magnitudes	Spectral	Prop Mot	2nd PM	RM	Design	Note	Precise
Identifier	First	Last	First	Last	Pri	Sec	Type	RA°	DEC°	RA°	DEC°	Coordinates			
2305+3528 ^{RB} 149	1999	1999	1 160	160	21.4	21.4	12.7	19.1		+090+055	+090+057		V		230555.49+352816.7
2308+3310 ^{RB} 150	2000	2001	2 146	146	31.3	31.2	9.84	15.2		-098-061	-093-065		V		230848.81+331003.1
2310+3626 ^{RB} 151	1999	1999	1 141	141	31.0	31.0	14.8	16.8		+050-018	+054-019		V		231052.22+362606.6
2314+5530 ^{RB} 215	2000	2003	2 145	145	20.7	20.6	10.99	12.12		-035-048	-037-044		V		231407.29+553029.4
2315+3336 ^{RB} 152	1999	1999	1 4	4	12.4	12.4	17.7	18.4		-027-048	-026-049		V		231551.50+333534.5
2316+3039 ^{RB} 153	1999	1999	1 162	162	16.4	16.4	18.7	18.8		+062+017	+065+018		V		231613.34+303915.6
2312+3206 ^{RB} 154	1999	1999	1 66	66	9.1	9.1	18.4	19.1		+020-065	+021-065		V		231216.80+320550.1
2318+0703 ^{RB} 22	1997	2000	3 325	325	14.9	15.0	14.2	15.1		+147-032	+151-035		V		231850.34+070325.3
2336+7250 ^{RB} 216	2000	2000	1 3	3	12.3	12.3	16.39	16.50		-044+041	-046+041		V		233632.59+724946.0
2341+5528 ^{RB} 217	1999	1999	1 116	116	30.1	30.1	14.90	18.64		+065-022	+067-023		V		234057.46+552821.1
2341+5619 ^{RB} 218	1999	1999	1 54	54	29.2	29.2	16.76	19.14		+019-087	+020-090		V		234112.18+561824.6
2345+5836 ^{RB} 219	1999	2003	2 333	334	22.1	21.8	16.28	17.35		+051-037	+063-036		V		234541.75+583603.1
2356+3524 ^{RB} 155	1998	1998	1 30	30	17.4	17.4	12.9	18.1		-020-054	-021-055		V		235602.30+352427.6

Figura 13. Búsqueda por cualquier texto

También se pueden consultar las imágenes correspondientes a las coordenadas seleccionadas mediante el *Aladin Sky Atlas*. Se trata de un software interactivo que permite al usuario visualizar las imágenes astronómicas digitalizadas, superponer entradas de catálogos o bases de datos astronómicas.

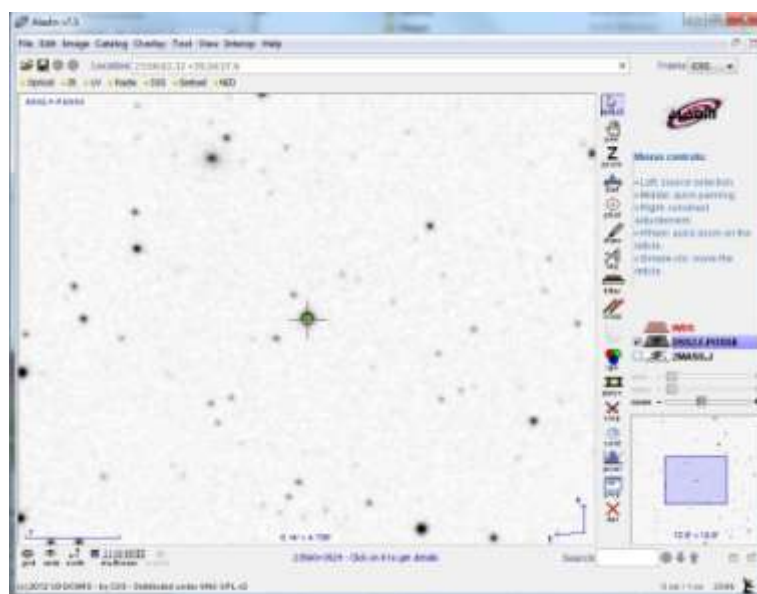


Figura 14. Aladin Sky Atlas

Otra forma de visualizar imágenes es conectándose al portal *Sloan Digital Sky Survey* (SDSS). Se trata de un ambicioso proyecto de inspección del espacio mediante imágenes en el espectro visible realizadas por un telescopio situado en el observatorio *Apache Point* de *Nuevo México*.

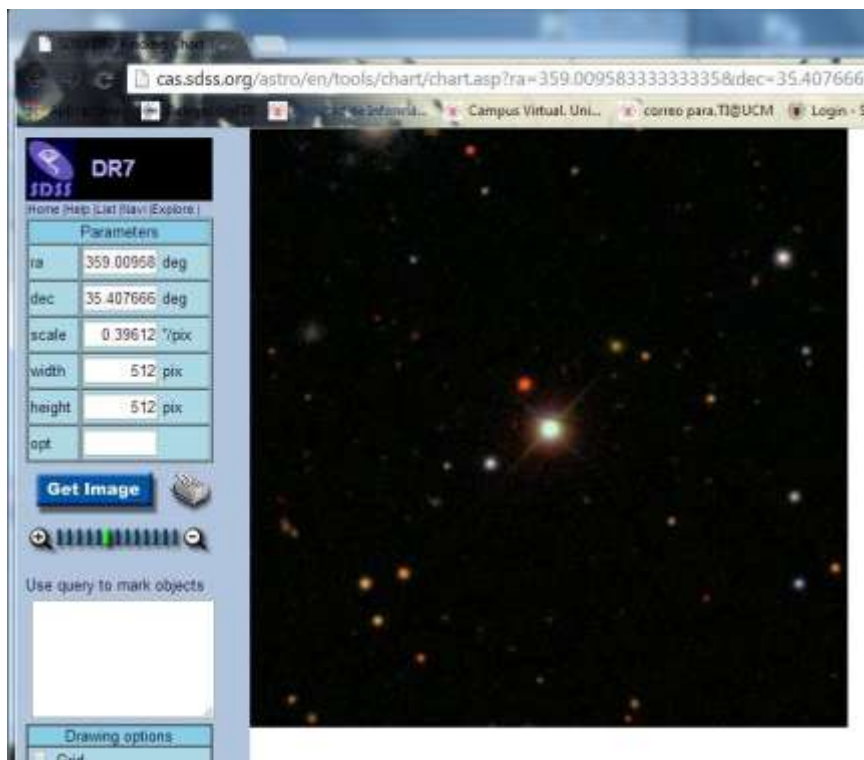


Figura 15. Sloan Digital Sky Survey

En la imagen se muestran los datos exportados a formato Excel tras pulsar el botón *Result to .xls*.

Combinar y centrar										condicional - como tabla - color -										estilos - como tabla - color -										y filtro - selecciona -																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
Portapapeles %		Fuente		Alineación		Número		Estilos		Celdas		Modificar																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													

Finalmente, si se selecciona una de las estrellas y se pulsa el botón *Get Notes/Grab. Cords*, la aplicación mostrará algunos datos en la consola, y generará un fichero (con extensión .txt.notes) con las notas existentes en el *WDS* relacionadas con dichas coordenadas.

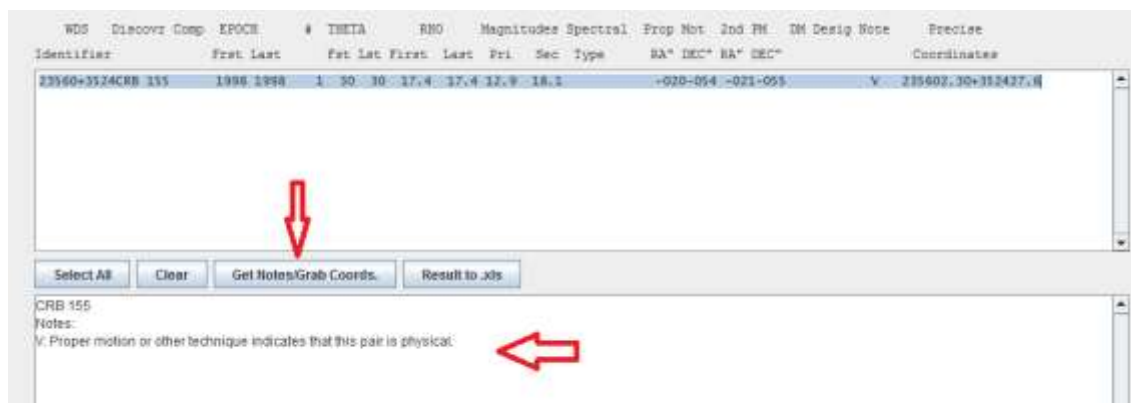


Figura 17. Notas

Parte II

Combinación de catálogos astronómicos

4. Análisis, diseño e implementación

4.1. Gestión del proyecto

En este apartado comentaremos las herramientas de apoyo para la organización, gestión y desarrollo de la segunda aplicación que compone nuestro proyecto.

Según las especificaciones iniciales la idea era poder reutilizar las funcionalidades implementadas en la primera aplicación, para ampliarlas en la segunda. Después de varias reuniones con el director de proyecto las especificaciones fueron cambiando y se llegó a la conclusión de que nos habíamos distanciado del objetivo final. Se decidió comenzar el desarrollo desde cero y convertirla en una aplicación totalmente independiente de la primera.

Se desarrolló en cuatro iteraciones:

4.1.1. Iteración 1

Objetivos establecidos para esta primera iteración:

- ✓ Finalizar primera parte de la memoria
- ✓ Bases del nuevo proyecto
- ✓ Plazos desarrollo
- ✓ Búsqueda de herramientas de desarrollo

Al tomar la decisión de realizar un nuevo proyecto independiente del anterior, decidimos completar la memoria de la primera aplicación así como dividirla en dos partes separadas.

En las primeras reuniones con nuestro director de proyecto, Rafael Caballero, fuimos estableciendo las bases a partir de las cuales iniciaríamos el desarrollo de esta aplicación. Se trataron de definir de la forma más clara y concisa posible los siguientes aspectos sobre la aplicación:

- ¿Qué vamos a desarrollar?
- ¿Para quién lo vamos a desarrollar?
- ¿Cómo lo vamos a desarrollar?

Se establecieron nuevos plazos de desarrollo, limitados al segundo cuatrimestre. Se decidió mantener las reuniones semanales con el director de forma que el desarrollo de la aplicación tuviera un seguimiento constante y fuera lo más ágil posible.

En cuanto a las herramientas de desarrollo, decidimos continuar con la plataforma utilizada en la primera aplicación una vez analizadas las diferentes alternativas de las que podíamos hacer uso.

4.1.2. Iteración 2

En esta segunda iteración nos centramos en el siguiente objetivo:

- ✓ Diseño de la interfaz de usuario

Dicha interfaz debía ser lo más sencilla, intuitiva y fácil de utilizar. Por tanto, después de varias versiones previas, se dividió en varias secciones diferenciadas, cada una de las cuales serviría para las distintas funcionalidades que se implementarían.

4.1.3. Iteración 3

Una vez definida la interfaz, nuestra prioridad se centró en los siguientes objetivos:

- ✓ Definición de los filtros
- ✓ Definición de los flujos de control

En esta iteración, estuvimos discutiendo sobre la estructura del filtrado. Realizamos un esquema con las distintas posibilidades para poder abordar dicho problema.

En cuanto a los flujos de control, planteamos dos posibilidades. Una de ella era crear un fichero *log.txt*, en el cual se mostrase una traza de control de uso. La otra, añadir un fichero *readme.txt*. Dicho fichero mostraría información sobre el contenido de los directorios y de los ficheros. En este caso, es interesante para poder saber el número de estrellas de cada filtrado. Al final, se decidió hacer ambos.

4.1.4. Iteración 4

Objetivos de esta iteración:

- ✓ Realización del parser
- ✓ Puesta en marcha de los filtros
- ✓ Finalización de la memoria

Para el parser, primero definimos cuál iba a ser nuestra gramática. Después, pasamos a implementarla, para poder filtrar las distintas estrellas. Una vez filtradas, se guardan en el directorio correspondiente, según hayan pasado o no los filtros.

A medida que finalizaba la implementación de la aplicación, se fue completando la memoria, añadiendo todos los aspectos referentes a esta segunda parte.

4.2. Especificación y arquitectura

A continuación mostramos la estructura de composición de la segunda aplicación.

4.2.1. Especificación y arquitectura global

Como se observar en la siguiente imagen, la aplicación consta de varios paquetes.

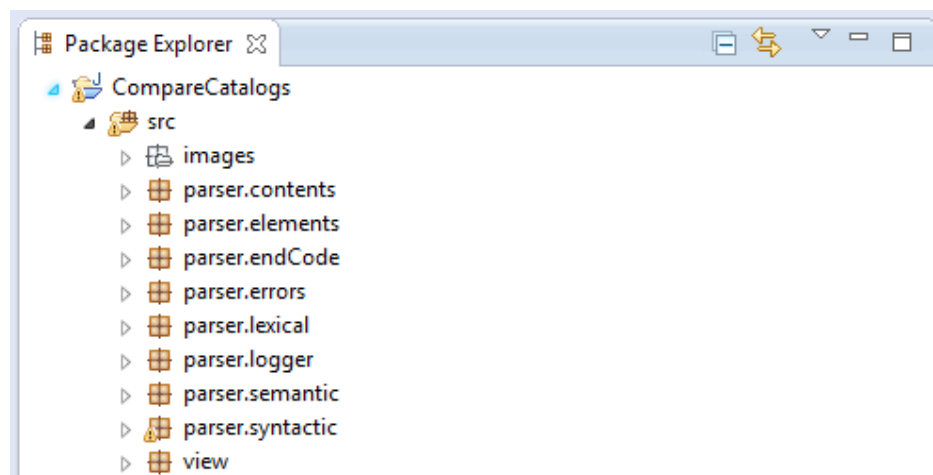


Figura 18. Explorador de paquetes



Figura 19. UML dependencia entre paquetes

- *Images*: paquete utilizado para guardar las distintas imágenes que utilizamos en la sección de ayuda para mostrar ejemplos de uso.
- *Parser*: todos los paquetes y las clases que tienen que ver con el procesador del lenguaje usado para este proyecto. Lo describiremos con más detalle más adelante.
- *View*: consta de las siguientes clases
 - *Main*: clase simple que se encarga de ejecutar la aplicación.
 - *Interface*: clase encargada del aspecto visual. Además implementa funciones que se encargan de gestionar el log y de hacer las llamadas a las distintas acciones.
 - *Catalog*: guarda la información de la URL desde la que se descarga el catálogo. Además, incluye funciones para descargar y guardar dicho catálogo en un fichero de texto.
 - *DescriptionData*: se encarga de la gestión de la parte del catálogo descargada. Obtiene la descripción de su información, es decir, los campos que define y cómo los define.
 - *DataSetructure*: sirve para especificar la estructura de los campos del catálogo.

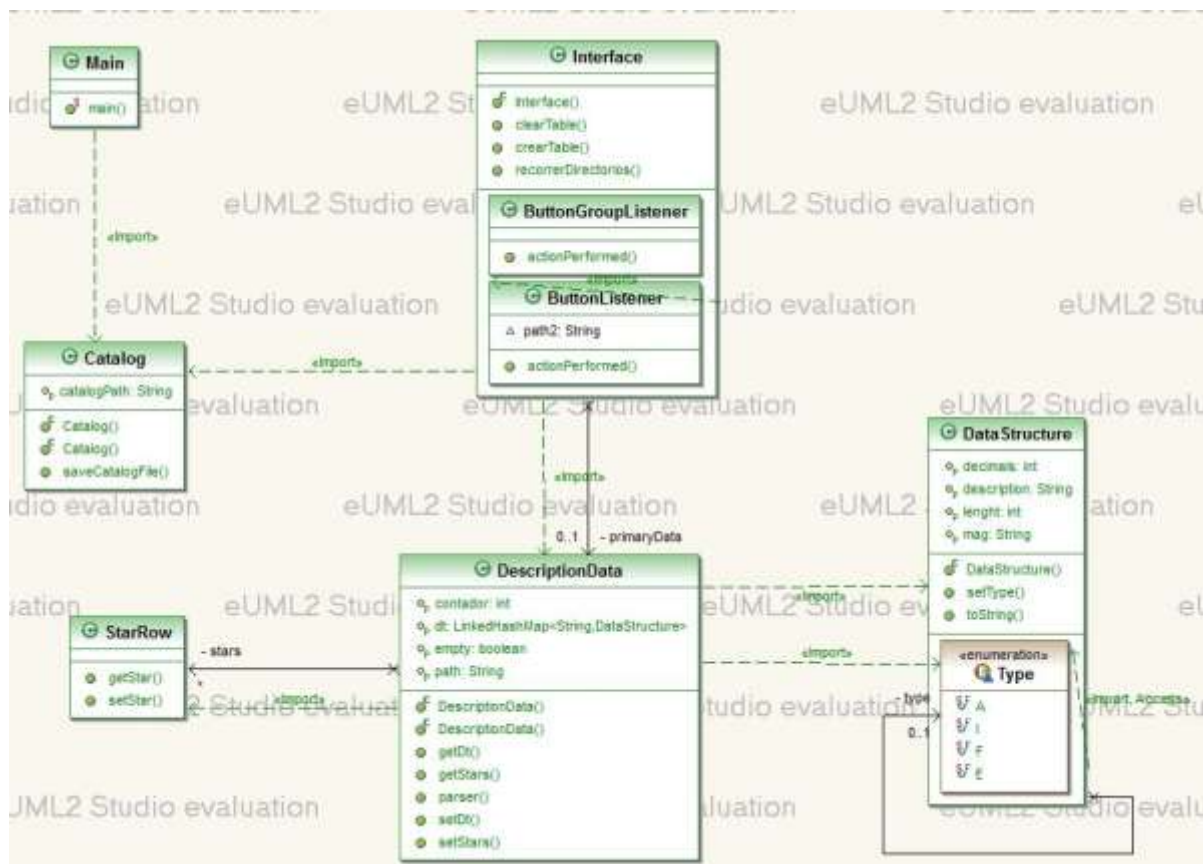


Figura 20. UML package view

4.2.2. Especificación y arquitectura del parser

Debido a la complejidad de esta parte de la aplicación, le dedicamos un apartado especial. Para realizar el parseado de los distintos filtros que el usuario pueda indicar, teníamos que definir un lenguaje. En un principio hicimos un boceto inicial de lo que sería la gramática. Después, decidimos reutilizar el proyecto de la asignatura de *Procesadores de Lenguaje*. Este proyecto consistía en un compilador del pseudocódigo utilizado en la asignatura de *Metodología y Tecnología de la Programación*.

Aquí se muestran las relaciones de dependencias entre las clases más representativas del parser.

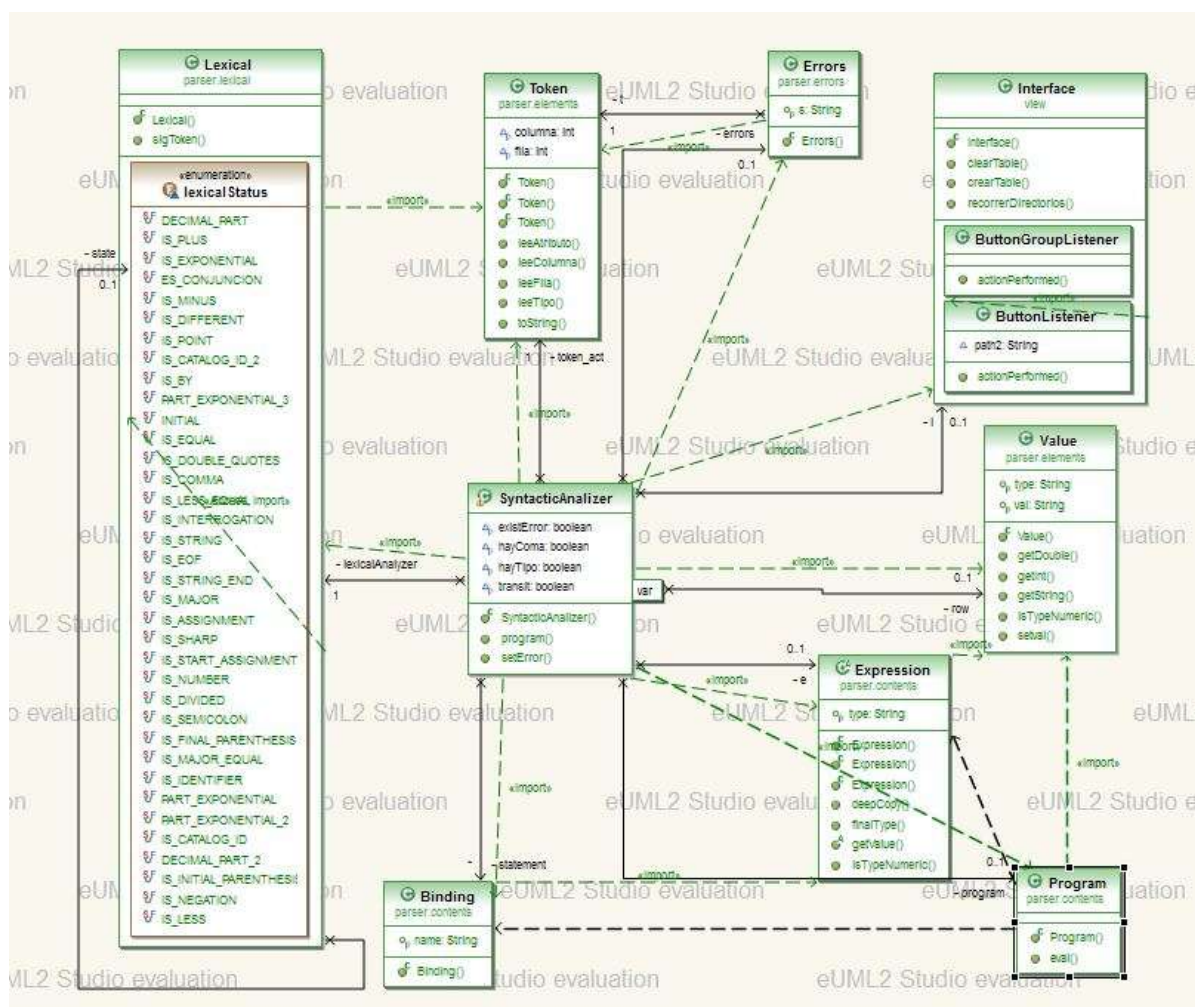


Figura 21. UML dependencias principales

La clase *Program* contiene información de las distintas instrucciones definidas por el usuario, gestionando el filtrado.

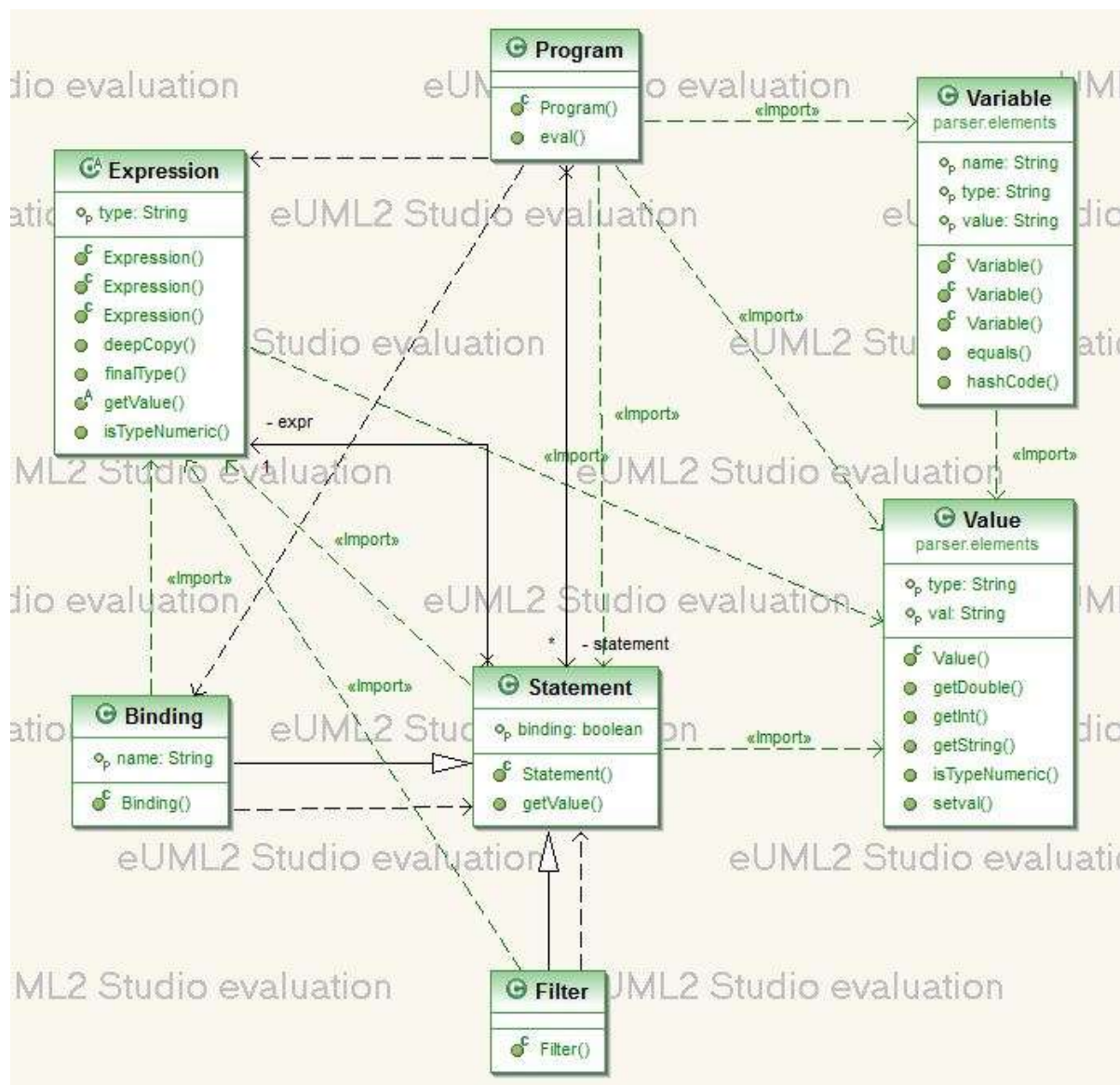


Figura 22. UML Program

En el paquete *Elements* se encuentran varias clases que utiliza el parser, para estructurar la información necesaria para el procesador.

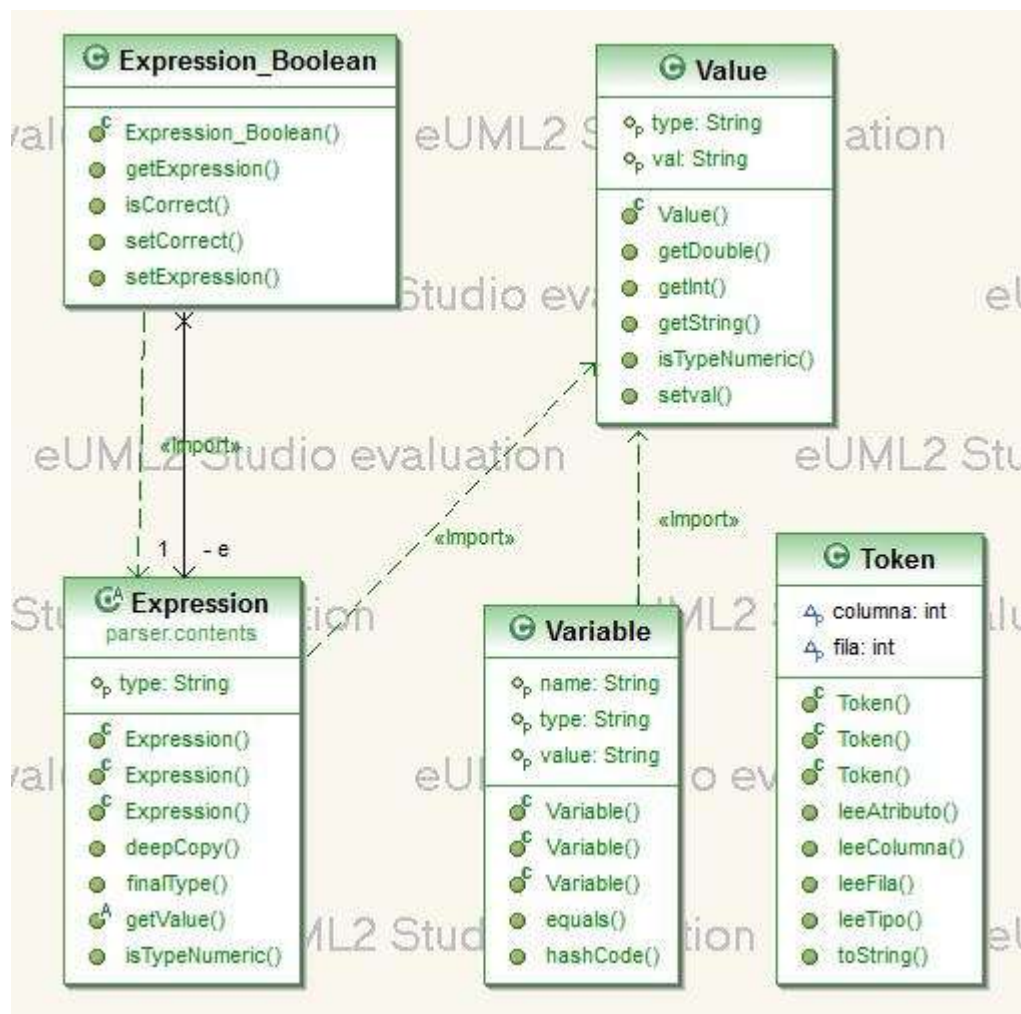


Figura 23. UML paquete Elements

Expression es una clase abstracta de la cual heredan todas las clases que definen las reglas de la gramática. Debido a la cantidad de clases que heredan de ella, hemos separado la visualización en cuatro diagramas UML.

1. Expresiones

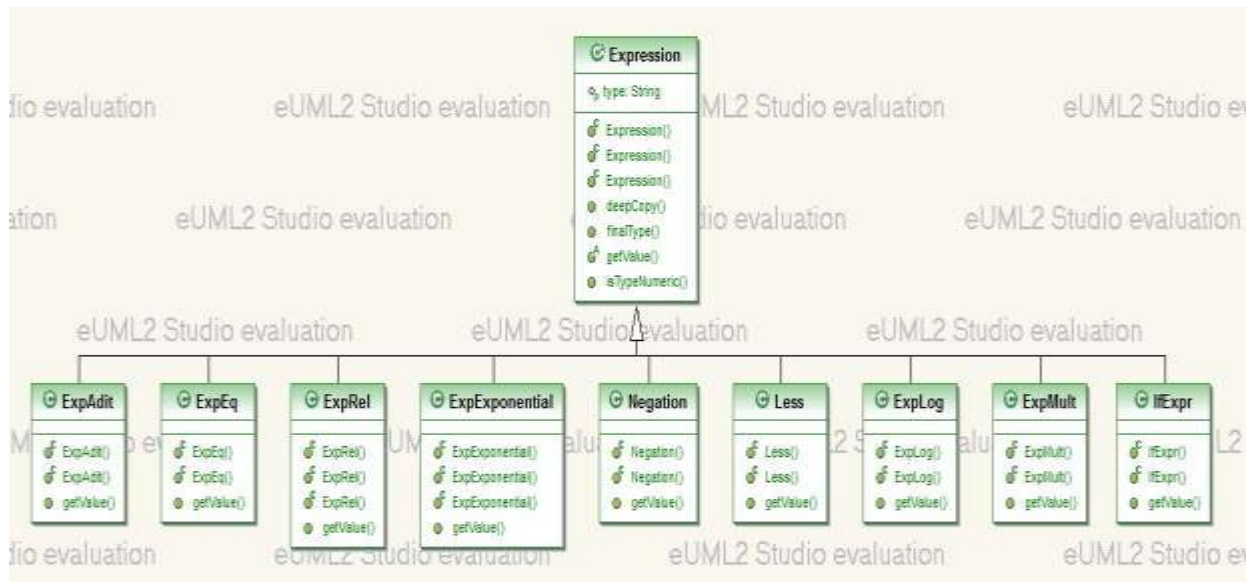


Figura 24. UML Expresiones

2. Funciones trigonométricas

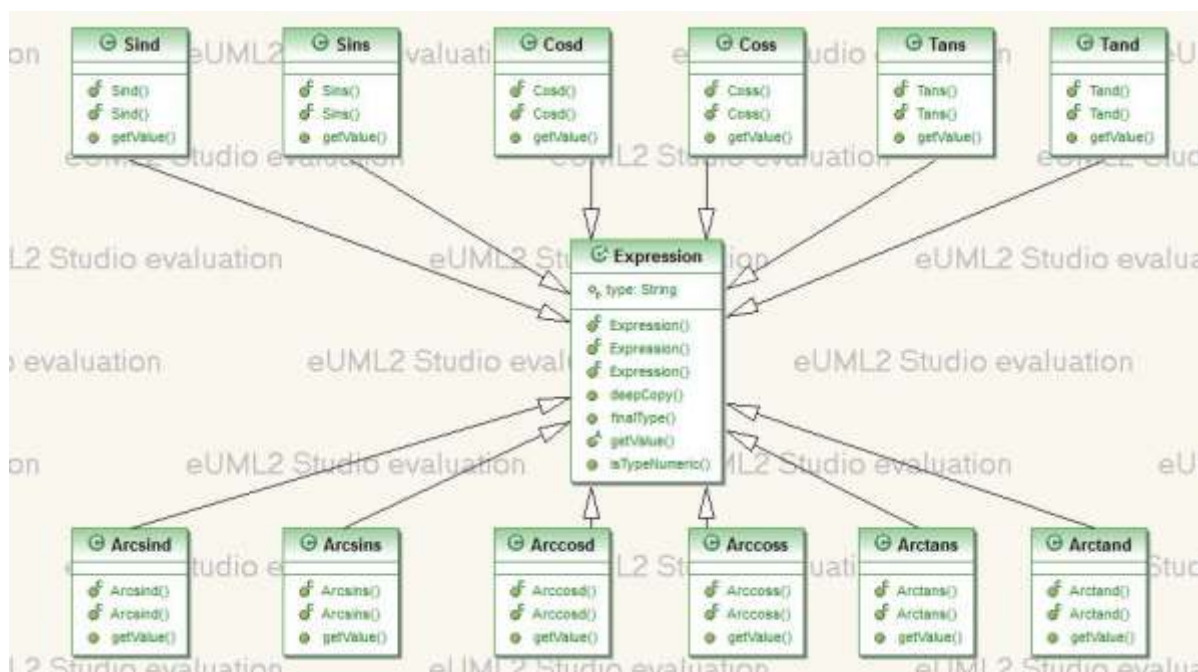


Figura 25. UML Funciones trigonométricas

3. Otras funciones

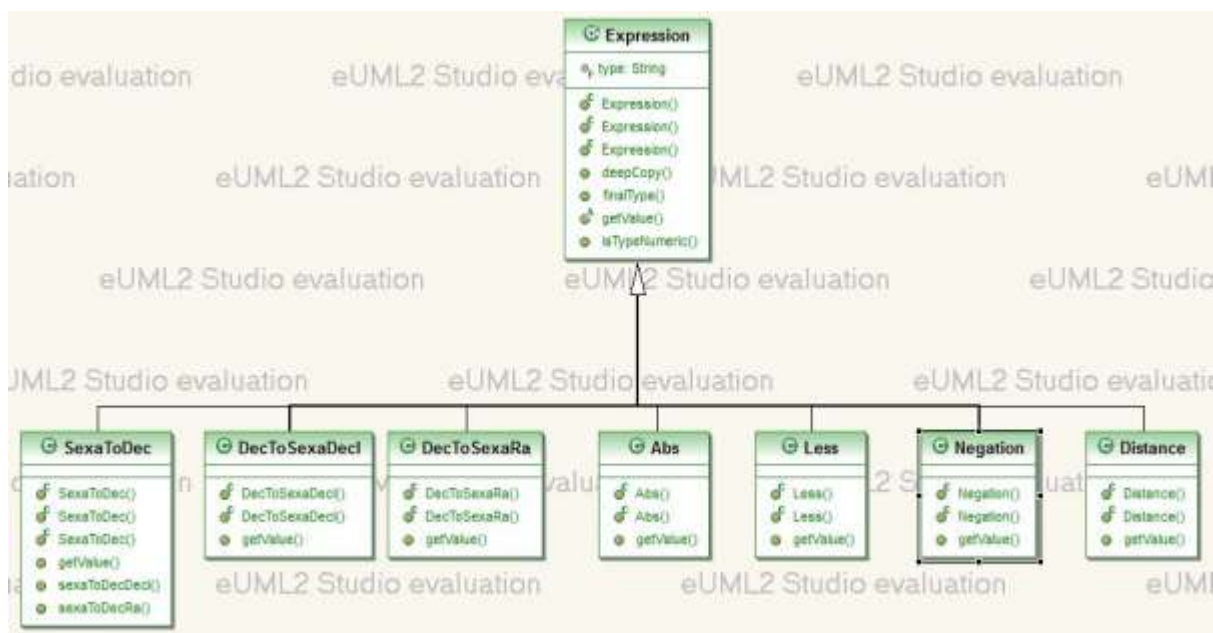


Figura 26. UML Otras funciones

4. Tipos

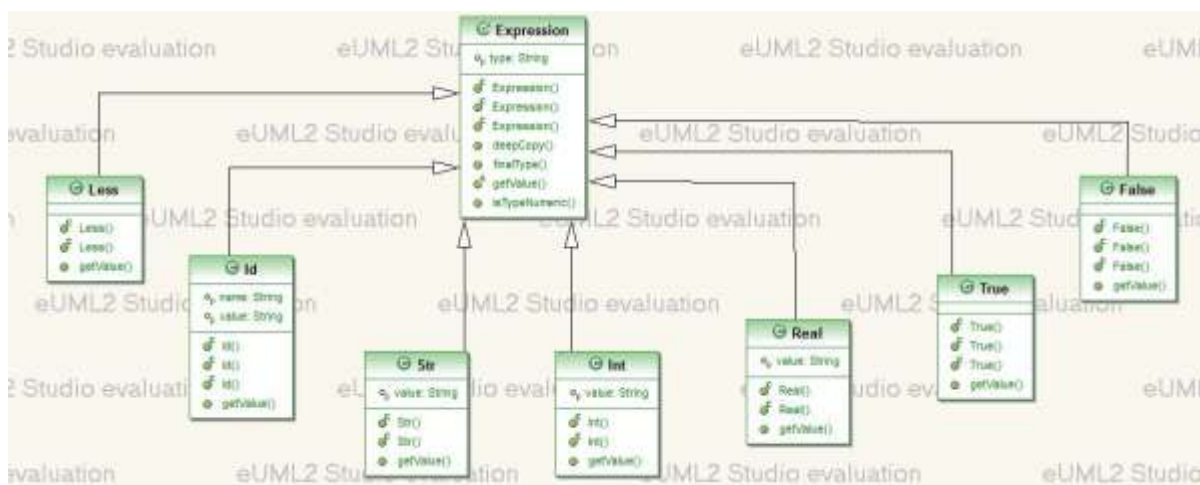


Figura 27. UML Tipos

4.2.2.1. Especificación inicial de la gramática

```

program ::= (statement';')*
statement ::= filter | assignment
assignment ::= varname ':=' expr
filter ::= '#'Bexpr

```

```

expr ::= Bexpr | lexpr | Fexpr | Aexpr | IFexpr
IFexpr ::= 'IF' Bexpr 'THEN' expr 'ELSE' expr

```

```

Bexpr = false | true | '(' Bexpr ')' | Bexpr RelOp Bexpr | 'not(' Bexpr ')' | QFieldName

```

```

RelOp = '>' | '<' | '>=' | '<=' | '=' | '<>'

```

```

Fexpr ::= FNumber | sind '(' Fexpr ')' | sins | cosd | coss | dist | abs | exp(Fexpr,Fexpr) | s2d( Fexpr) |
d2s(Fexpr) | QFieldName

```

Léxico:

```
varname ::= id
```

```
FNumber ::= int '.' int
```

```
QFieldName ::= [ P | S ].FieldName
```

4.2.2.2. Gramática de la asignatura de Procesadores del Lenguaje

- R1: principal → **Programa** id cuerpo **fPrograma**
- R2: cuerpo → [declaraciones] [Insts]
- R3: declaraciones → [decTipo] [**var** declaracion] [bloque]
- R4: decTipo → **tipos** objetos **ftipos**
- R5: objetos → id = tipo '**n**' [objetos]
- R6: objetos → id = id [rangos] de tipo '**n**' [objetos]
- R7: tipo → **reg** campos **freg**
- R8: tipo → rangos
- R9: tipo → **puntero a** tipo
- R10: tipo → **nat**
- R11: tipo → **ent**
- R12: tipo → **real**
- R13: tipo → **nat**⁺
- R14: tipo → **ent**⁺
- R15: tipo → **real**⁺
- R16: tipo → **nat**[∞]
- R17: tipo → **real**[∞]
- R18: tipo → **ent**[∞]
- R19: tipo → **booleano**
- R20: tipo → (valores)
- R21: tipo → **num**
- R22: tipo → **elemento**
- R23: tipo → **car**
- R24: tipo → id
- R25: campos → campo [, campos]
- R26: campo → listalIdentificadores : tipo
- R27: campo → (id [rangos])* **de** tipo
- R28: listalIdentificadores → id (, id)*
- R29: rangos → rango [, rangos]
- R30: valores → id (, id)*
- R31: declaracion → listalIdentificadores : tipo [, declaracion]
- R32: declaracion → (id [rangos])* **de** tipo [, declaracion]

- R33: bloque \rightarrow función [bloque]
- R34: bloque \rightarrow procedimiento [bloque]
- R35: función \rightarrow **fun** id (entradas) **dev** salida cuerpo **ffun**
- R36: procedimiento \rightarrow **proc** id (entradas) cuerpo **fproc**
- R37: entradas \rightarrow entrada [, entradas]
- R38: entrada \rightarrow [**e** | **e/s** | **s**] listaIdentificadores : tipo
- R39: entrada \rightarrow [**e** | **e/s** | **s**] (id [rangos])* **de** tipo
- R40: salida \rightarrow id : tipo | id [rangos] **de** tipo
- R41: salida \rightarrow < (listaIdentificadores : tipo | (id [rangos])* **de** tipo) , declaracion >
- R42: Insts \rightarrow Inst (; | '\n') [Insts]
- R43: Insts \rightarrow Inst
- R44: Inst \rightarrow asig
- R45: Inst \rightarrow casos0
- R46: Inst \rightarrow instSi
- R47: Inst \rightarrow instPara
- R48: Inst \rightarrow instMientras
- R49: Inst \rightarrow metodo
- R50: Inst \rightarrow **nada**
- R51: asig \rightarrow Expid := Exp0
- R52: asig \rightarrow <Expid(,Expid)⁺ := <Exp0(,Exp0)⁺>
- R53: casos0 \rightarrow **casos** Exp0 \rightarrow Insts [casos1] **fcasos**
- R54: casos1 \rightarrow [] Exp0 \rightarrow Insts [casos1]
- R55: instSi \rightarrow **si** Exp0 **entonces** Insts **fsi**
- R56: instSi \rightarrow **si** Exp0 **entonces** Insts **si no** Insts **fsi**
- R57: instPara \rightarrow **para** asig-para **hasta** Exp0 **hacer** Insts **fpara**
- R58: instPara \rightarrow **para** asig-para **hasta** Exp0 **paso** Exp0 **hacer** insts **fpara**
- R59: asig-para \rightarrow id = Exp0
- R60: instMientras \rightarrow **mientras** Exp0 **hacer** Insts **fmientras**
- R61: metodo \rightarrow id (parametros)
- R62: parametros \rightarrow [Exp0 [, parametros]]

- R63: Exp0 \rightarrow Exp1
- R64: Exp0 \rightarrow Exp1 op_log Exp1
- R65: Exp1 \rightarrow Exp2
- R66: Exp1 \rightarrow Exp2 op_relacional Exp1
- R67: Exp1 \rightarrow Exp2 op_igual Exp1
- R68: Exp2 \rightarrow Exp3
- R69: Exp2 \rightarrow Exp3 op_adit Exp2
- R70: Exp3 \rightarrow Exp4
- R71: Exp3 \rightarrow Exp4 op_mult Exp3
- R72: Exp4 \rightarrow Exp5
- R73: Exp4 \rightarrow Exp5 ^ Exp4
- R74: Exp5 \rightarrow neg Exp5
- R75: Exp5 \rightarrow men Exp5
- R76: Exp5 \rightarrow naturales
- R77: Exp5 \rightarrow enteros
- R78: Exp5 \rightarrow reales
- R79: Exp5 \rightarrow naturales⁺
- R80: Exp5 \rightarrow enteros⁺
- R81: Exp5 \rightarrow reales⁺
- R82: Exp5 \rightarrow naturales ^{∞}
- R83: Exp5 \rightarrow enteros ^{∞}
- R84: Exp5 \rightarrow reales ^{∞}
- R85: Exp5 \rightarrow booleanos
- R86: Exp5 \rightarrow caracteres
- R87: Exp5 \rightarrow Expid
- R88: Exp5 \rightarrow (Exp0)
- R89: Exp5 \rightarrow [elem_vector | rango | Expid]
- R90: Expid \rightarrow id
- R91: Expid \rightarrow id [elem_vector | rango | Expid]

- R92: Expid → id ‘.’ Expid
 - R93: Expid → id
 - R94: Expid → id (parámetros)
 - R95: elem_vector → Exp0 [, elem_vector]
 - R96: rango → Exp0 op_rango Exp0
 - R97: op_rango → ‘.’
 - R98: op_adit → + | -
 - R99: op_mult → * | / | **div** | **mod**
 - R100: op_relacional →
 - R101: op_igual →
 - R102: op_log →
 - R103: neg → ¬
 - R104: men → -
 - R105: d → 0 | ... | 9
 - R106: l → a | ... | z | A | ... | Z | á | é | í | ó | ú | ü | Á | É | Í | Ó | Ú | Ü
 - R107: d₊ → 1 | ... | 9
 - R108: id → l (l | d | _)*[?]
 - R109: naturales → [+] d d*
 - R110: enteros → [+ | -] d d*
 - R111: reales → [+ | -] d d* [.d d*]
 - R112: naturales⁺ → [+] d⁺ d*
 - R113: enteros⁺ → [+] d⁺ d*
 - R114: reales⁺ → [+] (d⁺ d* [.d d*] | 0 [.d *] d*[d*])
 - R115: naturales[∞] → [+] ∞ | naturales
 - R116: enteros[∞] → [+ | -] ∞ | enteros
 - R117: reales[∞] → [+ | -] ∞ | reales
 - R118: booleanos → cierto | falso
 - R119: caracteres → ‘(l | d | s)’
 - R120: s → ‘ ‘ | ‘(‘ |)’ | ‘_’ | ‘:’ | ‘+’ | ‘-’ | ‘.’ | ‘*’ | ‘/’ | ‘<’ | ‘≤’ | ‘>’ | ‘≥’ | ‘=’ | ‘≠’ | ‘,’ | ‘|’ | ‘.’
 - R121: comentario → ‘{ ’ }
- delim → ‘ ‘ | ‘(‘ |)’ | EOL | EOF | TAB | ‘:’ | ‘+’ | ‘-’ | ‘.’ | ‘*’ | ‘/’ | ‘div’ | ‘mod’ | ‘<’ | ‘≤’ | ‘>’ | ‘≥’ | ‘=’ | ‘≠’ | ‘,’ | ‘|’ | ‘.’

4.2.2.3. Gramática final

- R1: program \rightarrow (statement ';')*
- R2: statement \rightarrow filter | binding
- R3: binding \rightarrow id ':'= Exp0
- R4: filter \rightarrow '#' Exp0 | binding
- R5: Exp0 \rightarrow Exp1
- R6: Exp0 \rightarrow Exp1 op_log Exp1
- R7: Exp1 \rightarrow Exp2
- R8: Exp1 \rightarrow Exp2 op_eq Exp1
- R9: Exp2 \rightarrow Exp3
- R10: Exp2 \rightarrow Exp3 op_rel Exp2
- R11: Exp3 \rightarrow Exp4
- R12: Exp3 \rightarrow **dist** '(' Exp4 ',' Exp4 ',' Exp4 ',' Exp4 ')'
- R13: Exp3 \rightarrow **abs** '(' Exp4 ')'
- R14: Exp3 \rightarrow **sind** '(' Exp4 ')'
- R15: Exp3 \rightarrow **cosd** '(' Exp4 ')'
- R16: Exp3 \rightarrow **sins** '(' Exp4 ')'
- R17: Exp3 \rightarrow **cooss** '(' Exp4 ')'
- R18: Exp3 \rightarrow **tans** '(' Exp4 ')'

R19: Exp3 → **tand** '(' Exp4 ')'
 R20: Exp3 → **arcsins** '(' Exp4 ')'
 R21: Exp3 → **arcsind** '(' Exp4 ')'
 R22: Exp3 → **arccoss** '(' Exp4 ')'
 R23: Exp3 → **arccosd** '(' Exp4 ')'
 R24: Exp3 → **arctans** '(' Exp4 ')'
 R25: Exp3 → **arctand** '(' Exp4 ')'
 R26: Exp3 → **s2d** '(' Exp4 ')'
 R27: Exp3 → **d2sra** '(' Exp4 ')'
 R28: Exp3 → **d2sdec** '(' Exp4 ')'
 R29: Exp4 → Exp5
 R30: Exp4 → Exp5 op_adit Exp4
 R31: Exp5 → Exp6
 R32: Exp5 → Exp6 op_mult Exp5
 R33: Exp6 → Exp7
 R34: Exp6 → Exp7 ^ Exp6

 R35: Exp7 → neg Exp5
 R36: Exp7 → men Exp5
 R37: Exp7 → numbers
 R38: Exp7 → booleans
 R39: Exp7 → strings
 R40: Exp7 → (Exp0)
 R41: Exp7 → QFieldName
 R42: Exp7 → id | binding
 R43: Exp7 → IfExpr

 R44: IfExpr → 'if' Exp0 'then' Exp0 'else' Exp0

 R45: op_adit → + | -
 R46: op_mult → * | / | **div** | **mod**
 R47: op_rel → '>' | '<' | '>=' | '<='
 R48: op_eq → = | <>
 R49: op_log → **and** | **or**
 R50: neg → **not**
 R51: men → -

 R52: QFieldName → ['p' | 's'] '.' id ['?']
 R53: id → l (l | d | _)

 R54: numbers → integers | reals | exponentials
 R55: integers → [+ | -] d d*
 R56: reals → [+ | -] d d* [.d d*]
 R57: exponentials → [+ | -] d d* [.d d*] 'e' ('-' | '+') d d*
 R58: strings → "" [s | l | d]* ""
 R59: booleans → **true** | **false**

 R60: s → ' ' | '(' | ')' | '_' | '+' | '-' | ';' | '*' | '/' | '<' | '>' | '=' | ',' | ':' | '#' | '"' | '?'
 R61: d → 0 | ... | 9
 R62: l → a | ... | z | A | ... | Z

 delim → ' ' | '(' | ')' | **EOL** | **EOF** | **TAB** | '+' | '-' | ';' | ':' | '*' | '/' | **'div'** | **'mod'** | '<' | '>' | '=' | ',' | '#' | '"' | '?'

4.2.3. Especificación mandada al USNO

Boceto inicial enviado a los astrónomos del *Observatorio Naval de los Estados Unidos* a quienes va dirigida la aplicación.

INTERFACE DESCRIPTION

The suggested general aspect of the interface application is the following one:

The screenshot displays a web-based interface for astronomical catalog comparison. It is divided into several sections:

- Catalog Selection:** Includes input fields for 'Catalog P' (set to 'WDS') and 'Catalog S' (set to 'GSC2.3'), a 'Coordinates' field with the value '284915.29+504754.8', a 'Radius (")' field set to '10', and a 'Load' button.
- Filters:** Contains a 'Catalog P' dropdown and a 'Catalog S' dropdown (set to 'WDS (mag)').
- Criterion for selecting S rows:** A text input field containing 'abs(P Mag)-S.V)>2'.
- Criterion for detecting errors:** A text input field containing 'abs(P Mag)-S.V)<0.5', with a 'Show Oldest Candidate' checkbox and a 'Start' button below it.
- Result:** A large empty box for displaying the search results.
- Catalog Description:** A vertical sidebar on the right with a green background, listing various catalog attributes like RA, DEC, Epoch, etc., for both Catalog P and Catalog S.

In the rest of the document each part of the interface is described.

Catalog Selection

The two catalogs to be compared. Usually WDS and another VizieR catalog,

- Catalog P: The VizieR identifier of the primary catalog with which to compare. By default WDS.
- Catalog S: The VizieR identifier of secondary catalog.

In the case of common catalogs, the application admits directly the usual identifier (WDS, 2MASS, GSC2.3...). For other catalogs the VizieR identifier is required (e.g. I/301)

Coordinates

Fragment of the catalogs that will be considered

- Catalog P Coordinates and Radius ("): Consider only P stars within these coordinates and radius (interpreted in seconds). If the user writes ALL in the coordinates field, then the whole catalog is considered.
- Catalog S Radius("): For each P star, consider only S stars within the given radius (interpreted in seconds). In the case of WDS we consider that each WDS row contains in fact two stars: the primary and the secondary. Therefore, the system will look for S stars around the primary, and also for S stars around the secondary.
- The button load consults VizieR loading internally the data

Example: suppose we want to check the V mag in the WDS entry

18496+5048UC 3686 1998 2003 2 24 24 49.2 48.9 14.8 15.8 -035-049 -018-064 V
184935.29+504754.8

And we choose the GSC2.3 catalog, which contains information about V.

The value 2 in the WDS-radius is because we are sure that there is a star in this position. In the case of GSC2.3 we choose a wider radius 10 seconds.

Internally this generates the following information:

For the primary, star the coordinates are the WDS coordinates and the VizierR information obtained

Full	r	RAJ2000	DEJ2000	GSC2.3	RAJ2000	DEJ2000	Epoch	Fmag	imag	Vmag	Nmag	Class	a	e
	arcsec	"h:m:s"	"d:m:s"		deg	deg	yr	mag	mag	mag	mag		pix	
1	0.152	18 49 35.275	+50 47 54.85	N109000643	282.396979	+50.798570	1992.653	14.90	15.50	15.55	14.43	3	3.53	0.07
2	8.111	18 49 36.131	+50 47 56.31	N109000642	282.400544	+50.798975	1992.653	14.34	15.05	14.86	13.98	0	3.83	0.04
3	8.734	18 49 34.688	+50 48 01.41	N109004803	282.394535	+50.800393	1992.653	15.20	15.78	15.77	14.64	3	3.35	0.02

For the secondary the application computes the coordinates: 18 49 37.394 +50 48 39.64 and with radius 10, only one GSC 2.3 star is obtained:

Full	r	RAJ2000	DEJ2000	GSC2.3	RAJ2000	DEJ2000	Epoch	Fmag	imag	Vmag	Nmag	Class	a	e
	arcsec	"h:m:s"	"d:m:s"		deg	deg	yr	mag	mag	mag	mag		pix	
1	0.468	18 49 37.393	+50 48 40.11	N109004849	282.405804	+50.811141	1992.653	14.80	16.60	15.74	13.21	0	3.52	0.02

The rest of the fields are used to discard/select the row candidates, and to obtain the possible updates for WDS.

Catalog description

a) Catalog P: Label of each column in catalog P.

If the primary catalog is WDS, this column additionally includes the coordinates for B star.

b) Catalog S: Label of each column in catalog S.

This information is obtained automatically from VizierR by the application (except the coordinates of the secondary star in the case of WDS, which are computed from the primary coordinates, the last separation and the last PA in the catalog).

The names of the fields are used in the rest of the application.

Filters

c) Catalog P: In this field, you can apply filters for selecting P rows (before comparing).

d) Catalog S: In this one, you can apply filters for selecting S rows (before comparing too).

In the example above, we do not need any filter for P (filters for P are useful only when a set of pairs are selected and we wish to discard some of them). However, in the case of catalog GSC 2.3 we are interested only in those rows containing information about magnitude V. In this example the rows obtained verify this condition, thus no row is discarded at this point

Filters	
Catalog P	Catalog S
	notnull(magV)

Criteria for selecting S rows

- e) In the text field, you can define which criteria will be used to compare rows between both catalogs. For instance we can say that only GSC 2.3 rows with magnitude difference with respect to WDS less than 2 should be considered. This condition is expressed as $(\text{abs}(\text{P.VMag?}-\text{S.VMag}) < 2)$. The idea is that greater difference means that we are selecting an erroneous star in GSC2.3

For filters and criteria, there will be able some arithmetic and predefined functions, like ' $>$ ' or `notnull(K)`. Also, for WDS, values for both A and B stars can be referenced with an '?'.

Criteria for detecting errors

- f) Here, you can define in which case you want the program to detect some error. For instance we consider that magnitude difference over 0.5 should be reported.

The One to One check-box indicates that the WDS star must be selected only if after the previous criterium (the selection of S rows) only one row remains for the star. The second check-box "Show Closest Candidate" is used when the previous check-box is not selected, that is several S candidates are allowed, but we only want to select the closest one (in terms of distance).

The result in this case in one suggested correction: for the primary, WDS includes V mag of 14.8, but GSC2.3 contains the value 15.55. The result indicates that the star has been found at 0.152" of the corresponding WDS star.

Observe that for the secondary no information is reported. This is because the GSC star found has magnitude 15.74, while the WDS indicates 15.8. That is, the magnitude difference does not verify the condition to be reported as a possible update.

4.3. Implementación

La implementación de esta segunda aplicación se ha realizado utilizando las mismas herramientas de trabajo que en la primera.

En cada una de las reuniones semanales se presentaban los avances de la aplicación. Debido a la complejidad del proyecto, no todas las semanas era posible ejecutar la aplicación para comprobar su funcionamiento, por lo que se mostraba el código hasta la fecha implementado y se marcaban las pautas para continuar desarrollando.

4.3.1. Control de versiones y documentación

Para controlar las versiones del código comenzamos utilizando *Google Code*, repositorio gratuito para el alojamiento de proyectos *Open Source*, para posteriormente migrar todo el código a *GitHub*, plataforma de desarrollo colaborativo de software. Algunos de los motivos por los que decidimos cambiar *GitHub* fueron:

- Visor de ramas para ir comparando los progresos que se van realizando en las distintas ramas de nuestro repositorio.
- Un sistema de seguimiento de problemas, clásico sistema de *tickets*. Esto permite que cualquier miembro del grupo, incluido el director del proyecto, puedan abrir *tickets* para detallar alguna incidencia o comentario.

En cuanto a la documentación, toda la segunda parte del proyecto se continuó alojando en la plataforma *Google Drive*.

4.3.2. Librerías y recursos

En esta segunda parte del proyecto, hemos hecho uso de dos librerías, *log4j.jar* para la creación de trazas y *wb.jar* para el diseño de la interfaz.

Window Builder API es un editor *WYSIWYG* (*What You See Is What You Get*) que soporta *SWT* y *Swing*. Proporciona una interfaz *drag-and-drop* que permite a los desarrolladores crear interfaces de usuario gráficas en Java en menor tiempo, con código legible. El diseño visual y la fuente siempre están sincronizados.

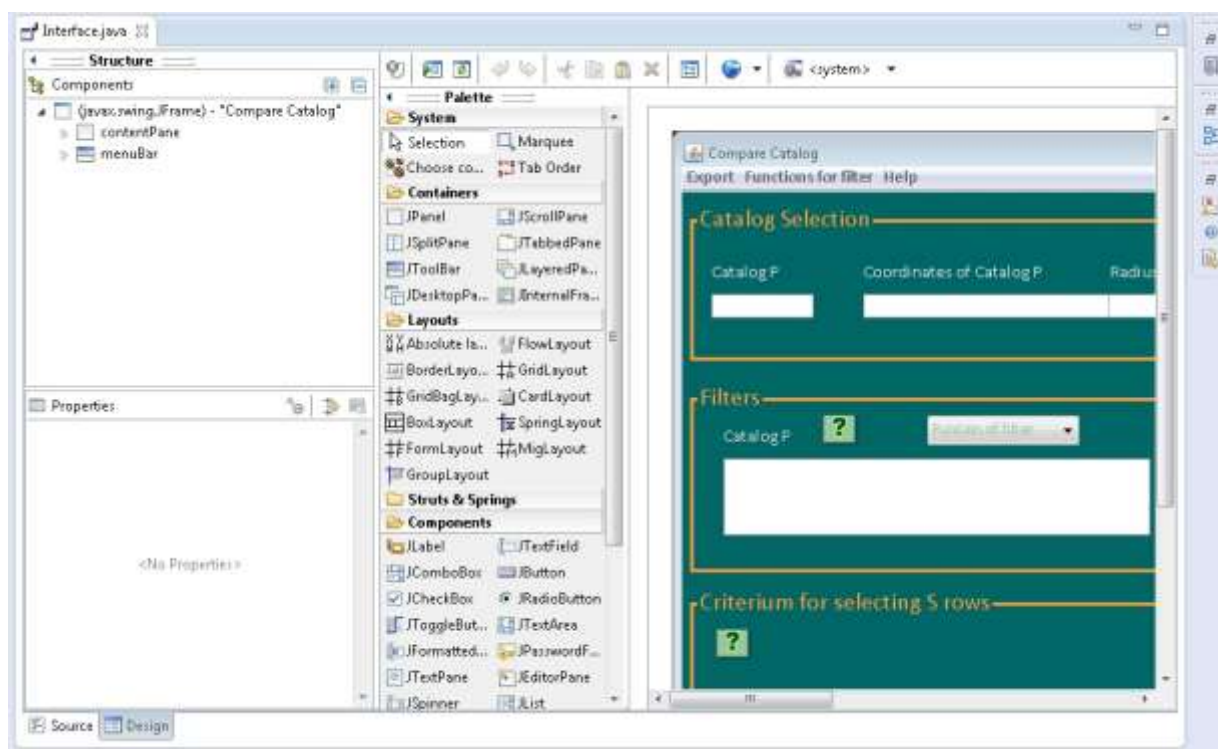


Figura 28. Window Builder

En cuanto a *log4j.jar*, es una librería de *Java* que permite a nuestra aplicación mostrar mensajes de información de lo que está sucediendo en ella, lo que de forma habitual conocemos como un log, así como su granularidad, todo ello a tiempo de ejecución y no a tiempo de compilación. Haciendo uso de este *framework*, se puede hacer un seguimiento de uso. Al ser dinámico, es fácilmente configurable.

Estos mensajes pueden variar según su nivel de importancia, lo que se conoce como niveles de prioridad de traza. En nuestro caso, hemos hecho uso del nivel *WARN*, usado para mostrar mensajes de alerta sobre los eventos de los que queremos mantener constancia pero que no afectan al correcto funcionamiento del programa, así como del nivel *INFO*, que se utiliza para mensajes de información que resaltan el progreso de la aplicación de una forma general.

El comportamiento de los loggers es jerárquico, como podemos observar en la siguiente tabla⁶:

⁶ <http://www.javatutoriales.com/2011/04/log4j-para-creacion-de-eventos-de-log.html>

Nivel de los mensajes que se mostrarán

	DEBUG	INFO	WARN	ERROR	FATAL
DEBUG					
INFO					
WARN					
ERROR					
FATAL					
ALL					
OFF					

Figura 29. Niveles del Logger

4.4. Limitaciones y problemas encontrados

Para el desarrollo de esta aplicación nos hemos encontrado varios problemas.

El primer problema con el que nos encontramos fue la posibilidad o no de obtener la información necesaria. Tuvimos que dedicar tiempo a investigar y después, encontrar la mejor forma de obtenerla.

Inicialmente, investigamos distintas páginas y distintos recursos sobre cómo descargar la información de cada catálogo. Nos encontramos con que la mayoría de los programas orientados a realizar consultas en Vizier estaban implementados en lenguaje *python*. Estudiamos este lenguaje (desconocido hasta entonces para nosotros) e intentamos adquirir la información con las librerías disponibles. Sin embargo, no obtuvimos los resultados esperados.

Al final, decidimos utilizar el método basado en URL que proporciona *VizieR* para obtener información de sus catálogos. En esta página <http://vizier.u-strasbg.fr/doc/asu-summary.htx> se explican las distintas opciones de realizar consultas con este método. Concretamente, elegimos el que, a partir de una URL determinada y con unos parámetros definidos, devuelve toda la información en un fichero de texto.

En nuestro caso, necesitamos descargar un trozo de catálogo que esté alrededor de ciertas coordenadas con un radio definido. La construcción de la URL nos queda de esta forma:

```
String url = "http://vizier.u-strasbg.fr/viz-bin/asu-txt?-source="+  
catalog + "&-c="+ coord. + "&-c.rs="+ rad;
```

Donde *catalog* es el nombre del catálogo, *coord* las coordenadas (separadas por espacios), y *rad* el radio en segundos.

Otro problema con el que nos encontramos tuvo que ver con el diseño de la interfaz. En un principio era demasiado grande. Había monitores en los que no se veía completamente. Al plantearlo, nos dimos cuenta de que los dos paneles que habíamos hecho para mostrar los resultados no eran necesarios, pues íbamos a ir guardando todos nuestros resultados en carpetas, subcarpetas y en ficheros de texto.

Además, nos dimos cuenta de que el criterio para seleccionar filas de S (catálogo secundario) se podía englobar en el filtro de S, y por tanto no era necesario.

Por último, el principal problema que tuvimos fue la falta de tiempo. Estuvimos un cuatrimestre entero con la primera aplicación, cuando en un principio estaba pensada para servir para ésta. Por otra parte, el diseño y la implementación del parser llevaron más tiempo del estimado, al tratarse de una tarea compleja.

5. Funcionamiento

En este apartado explicaremos detalladamente el funcionamiento de la aplicación.

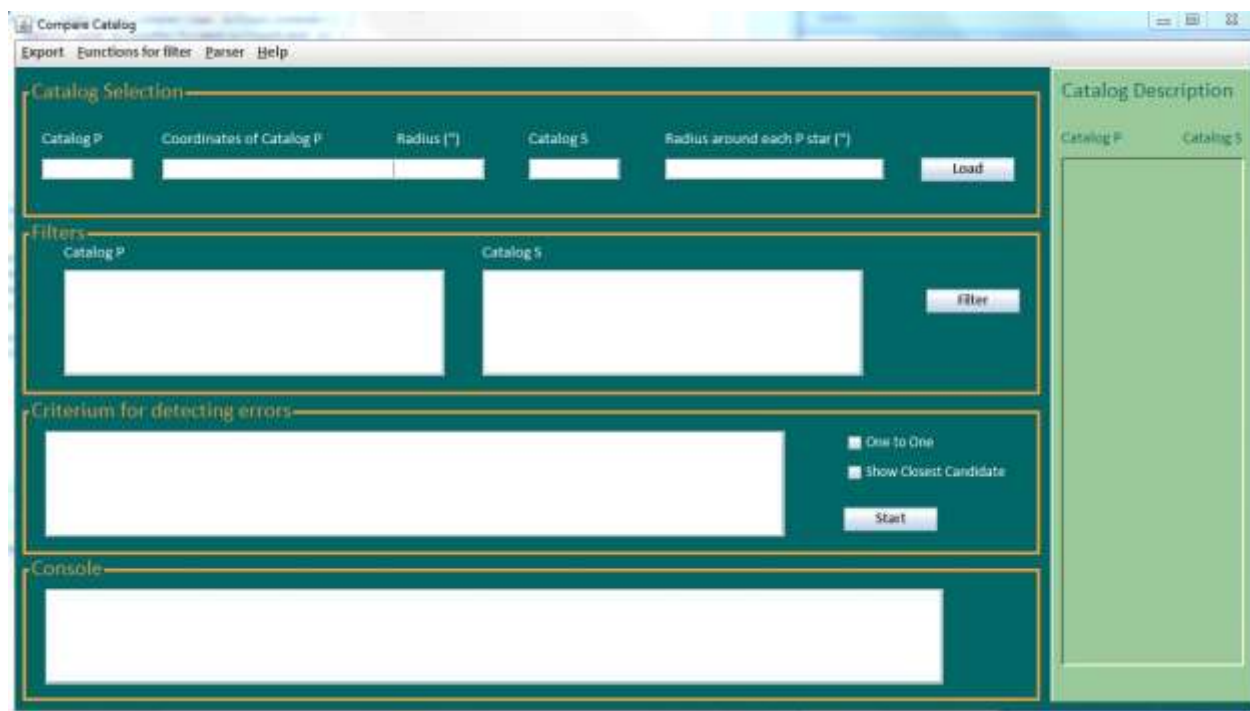


Figura 30. Interfaz gráfica

Como se puede apreciar está dividida en secciones claramente diferenciadas.

Sección 1. Menú

1.1. Export



Figura 31. Menú Export

- 1.1.1. *Save session*: nos permitirá guardar el estado actual de nuestras consultas actuales.
- 1.1.2. *Open session*: volverá a cargar la información en el punto donde lo dejamos anteriormente, evitándonos de este modo tener que volver a escribir los identificadores de los catálogos, sus coordenadas, filtros, etc.
- 1.1.3. *Exit*: para salir de la aplicación.

- 1.2. *Functions for filter*: aquí nos encontraremos un desplegable con los diferentes filtros que podemos aplicar a nuestras consultas, evitando tener que teclearlos cada vez que necesitemos hacer uso de cada uno de ellos.

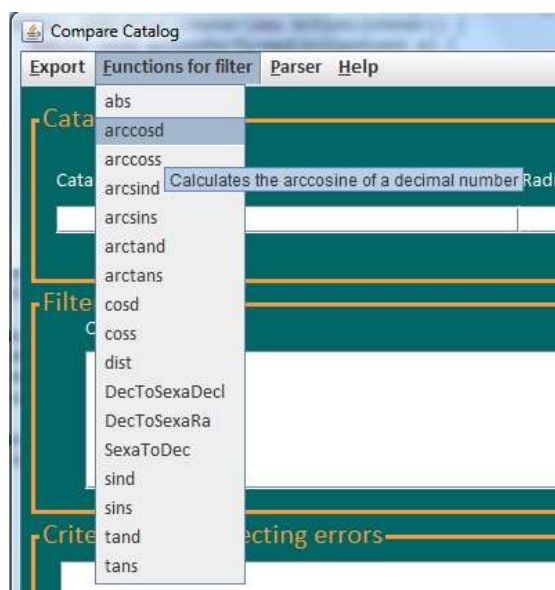


Figura 32. Menú Functions for filter

- 1.3. *Parser*: observamos dos nuevas opciones, *Grammar* y *Lexical*:

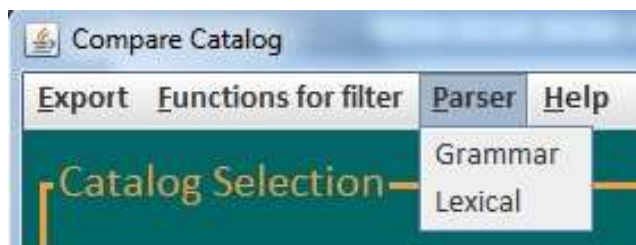


Figura 33. Menú Parser

Seleccionando cualquiera de las dos, nos aparecerá una nueva ventana donde se podrá observar de forma concreta la especificación tanto de la gramática como del léxico para la creación del compilador.

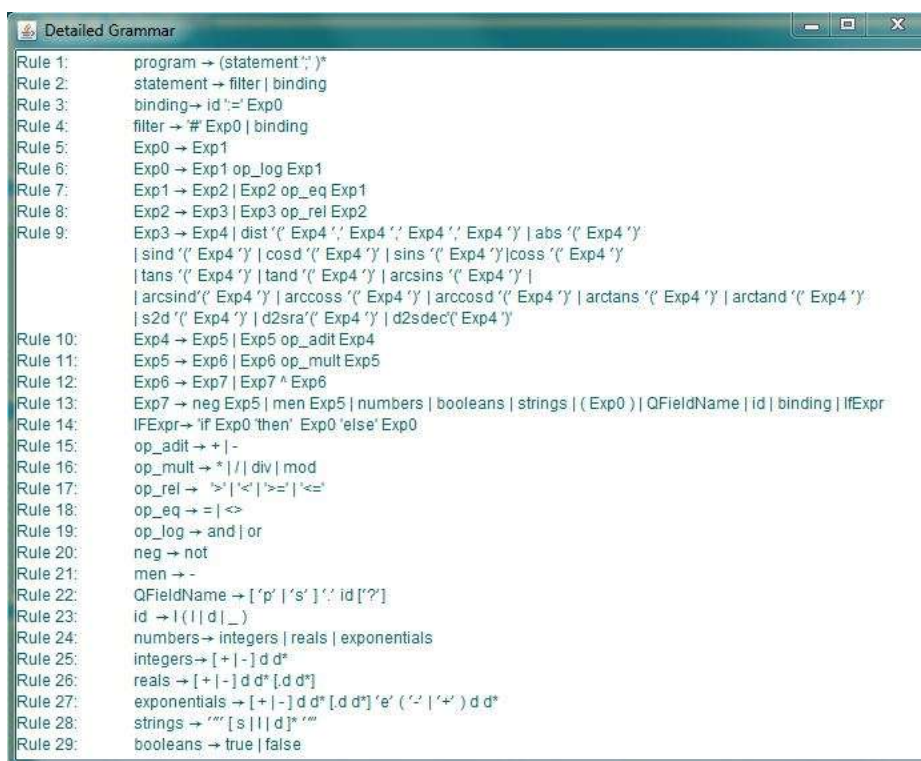


Figura 34. Detailed Grammar

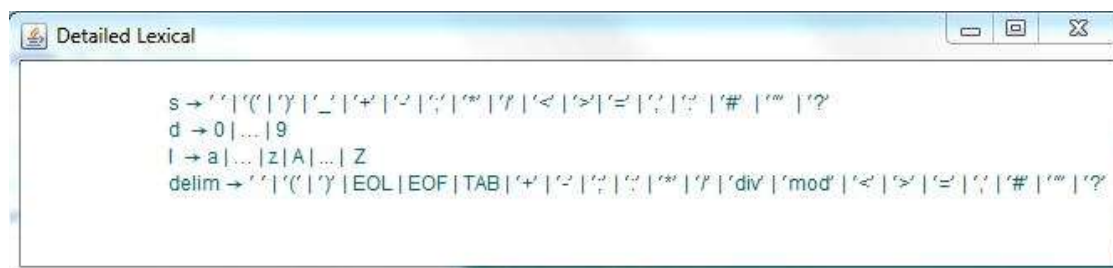


Figura 35. Detailed Lexical

- 1.4. *Help*: desplegable con los nombres de las distintas secciones de la aplicación. Seleccionando cada uno de ellos, aparecerá una nueva ventana con la explicación detallada y necesaria para la utilización de cada una de ellas. Además, con el botón *Example* se mostrará una imagen con un ejemplo de uso.

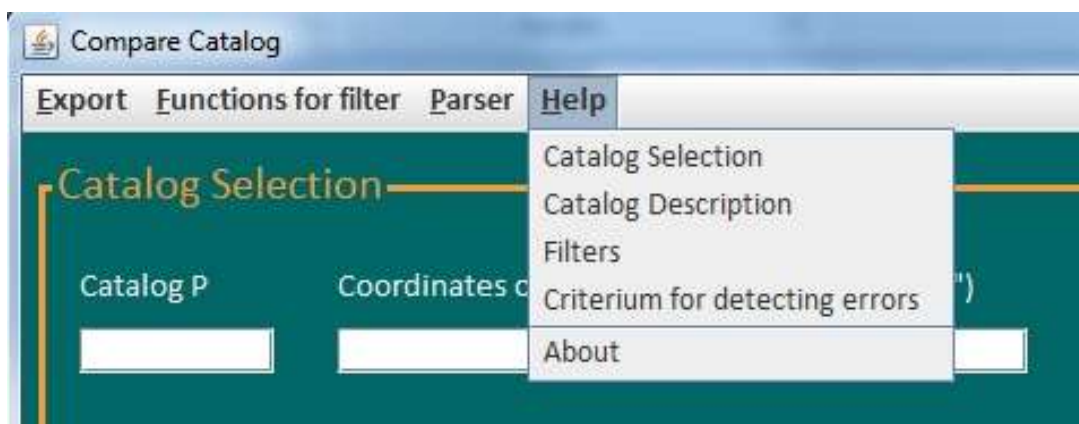


Figura 36. Menú Help

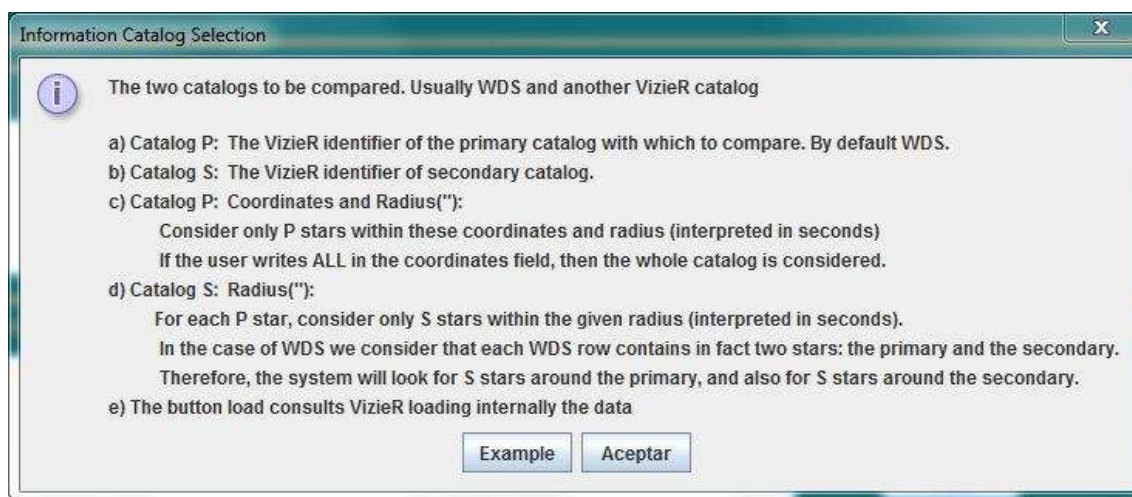


Figura 37. Information Catalog Selection



Figura 38. Example Catalog Selection



Figura 39. Information Catalog Description

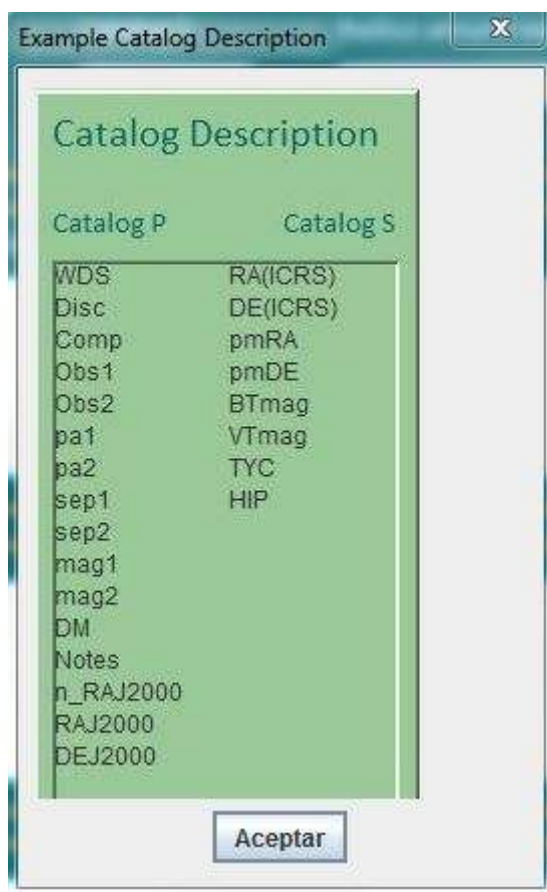


Figura 40. Example Catalog Description

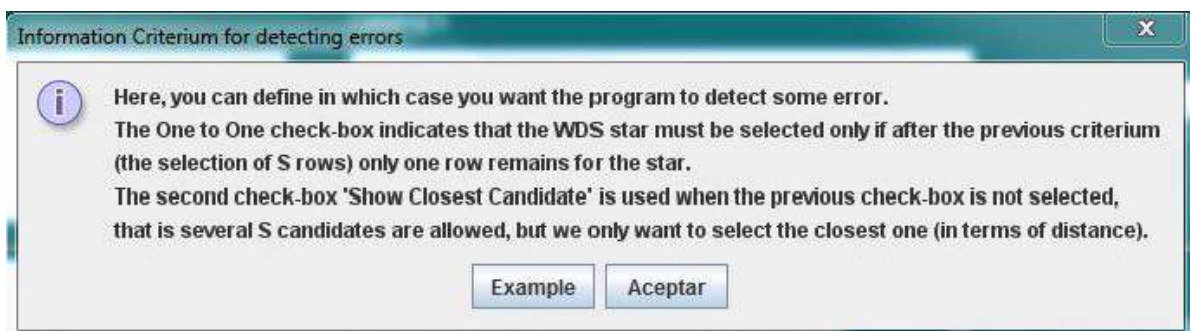


Figura 41. Information Criterium for detecting errors

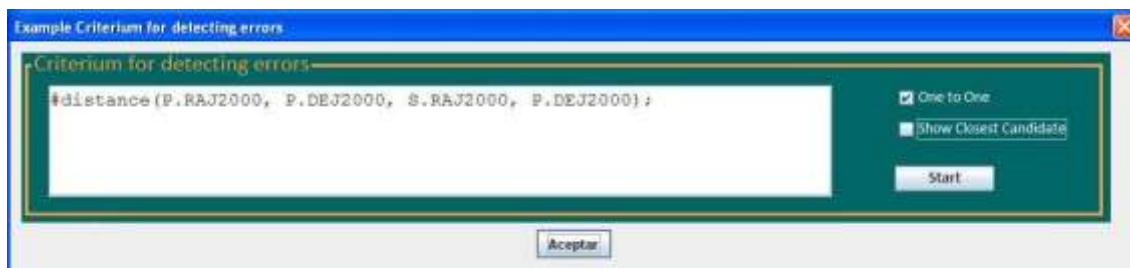


Figura 42. Example Criterion for errors

Sección 2. *Catalog Selection*: se podrá introducir la información relativa a los catálogos que queremos comparar. Los campos *Catalog P* y *Catalog S* se completarán con el identificador de los catálogos principal y secundario sobre los que se va a realizar la comparación. *Coordinates of Catalog P* y *Radius* se utilizarán para establecer las coordenadas y el radio, interpretado en segundos, del que partirá. Finalmente, en el campo *Radius around each P star* se introducirá el radio alrededor del cual se descargarán las estrellas del catálogo secundario.

Catalog P	Coordinates of Catalog P	Radius (")	Catalog S	Radius around each P star (")	
wds	184935.29+504754.81	1000	I/246	10000	Load

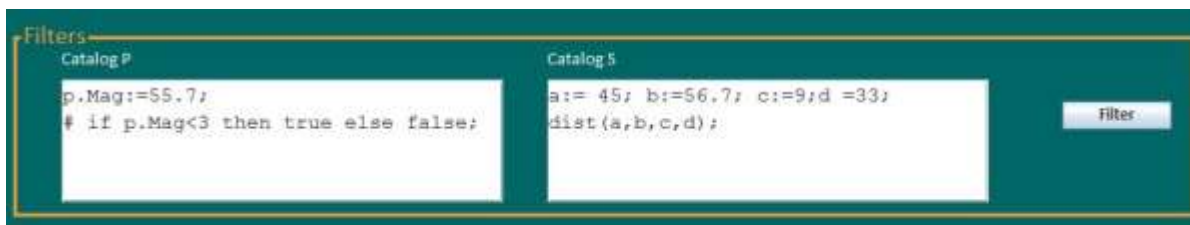
Figura 43. Catalog Selection

Sección 3. *Catalog Description*: permitirá listar las cabeceras de cada una de las columnas de los catálogos seleccionados en la sección anterior.

Catalog Description	
Catalog P	Catalog S
WDS	RA(ICRS)
Disc	DE(ICRS)
Comp	pmRA
Obs1	pmDE
Obs2	BTmag
pa1	VTmag
pa2	TYC
sep1	HIP
sep2	
mag1	
mag2	
DM	
Notes	
n_RAJ2000	
RAJ2000	
DEJ2000	

Figura 44. Catalog Description

Sección 4. *Filters*: el usuario podrá introducir los filtros que quiere aplicar a sus consultas, reduciendo aún más si cabe el rango de búsqueda y comparación de dichos catálogos, y permitiendo limitar los resultados a aquellos realmente deseados.



The 'Filters' interface consists of a teal header bar with the title 'Filters'. Below the header, there are two white input fields. The left field is labeled 'Catalog P' and contains the text: `p.Mag:=55.7;` and `# if p.Mag<3 then true else false;`. The right field is labeled 'Catalog S' and contains the text: `a:= 45; b:=56.7; c:=9;d =33;` and `dist(a,b,c,d);`. To the right of these fields is a button labeled 'Filter'.

Figura 45. Filters

Sección 5. *Criterion for detecting errors*: se definirán los filtros según los cuales se indicará que entre dos estrellas hay un error.

5.1. *One to one*: si se selecciona esta opción, sólo se tendrán en cuenta las estrellas del catálogo primario que sólo tengan una pareja en el catálogo secundario.

5.2. *Show Closest Candidate*: con esta otra opción, sólo se partirá de la estrella más cercana para comprobar si hay o no error.



The 'Criterion for detecting errors' interface has a teal header bar with the title 'Criterion for detecting errors'. Below the header, there is a large white text input field containing the text: `#distance(P.RAJ2000, P.DEJ2000, S.RAJ2000, P.DEJ2000);`. To the right of the input field are two checkboxes: 'One to One' (checked) and 'Show Closest Candidate' (unchecked). Below these checkboxes is a button labeled 'Start'.

Figura 46. Criterion for detecting errors

Estas opciones no se podrán seleccionar a la vez. Por otra parte, si no seleccionamos ninguna, el filtro se realizará sobre los resultados anteriores.

5.1. Caso práctico de uso

En este apartado vamos a mostrar el uso de la aplicación, con un caso de uso en concreto.

En primer lugar el usuario deberá introducir los datos asociados a *Catalog Selection*:

- Catalog P: catálogo primario.
- Coordinates of Catalog P: coordenadas iniciales.
- Radius ("): radio sobre el cual se buscarán estrellas en el catálogo, expresado en segundos.
- Catalog S: catálogo secundario.
- Radius around each P star ("): radio alrededor del cual se buscarán estrellas en el catálogo secundario (por cada una del primario).

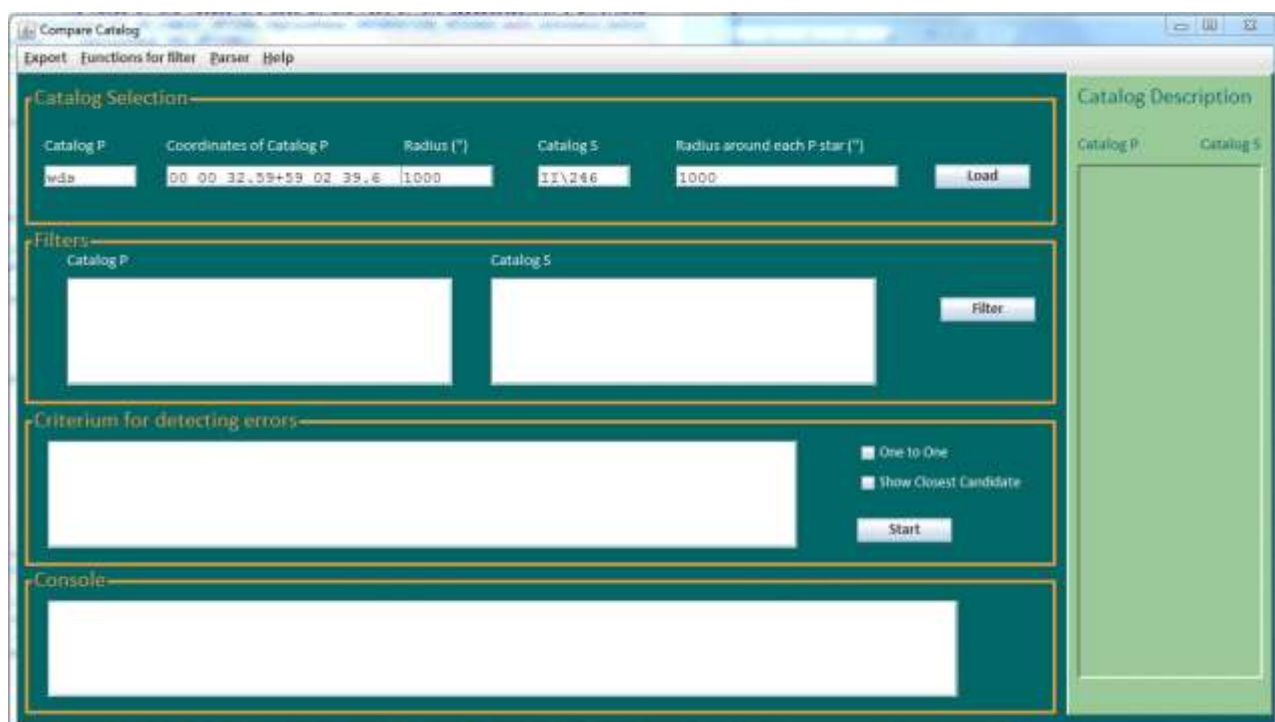


Figura 47. Paso 1

Una vez pulsado el botón *Load* aparecerá la siguiente ventana:

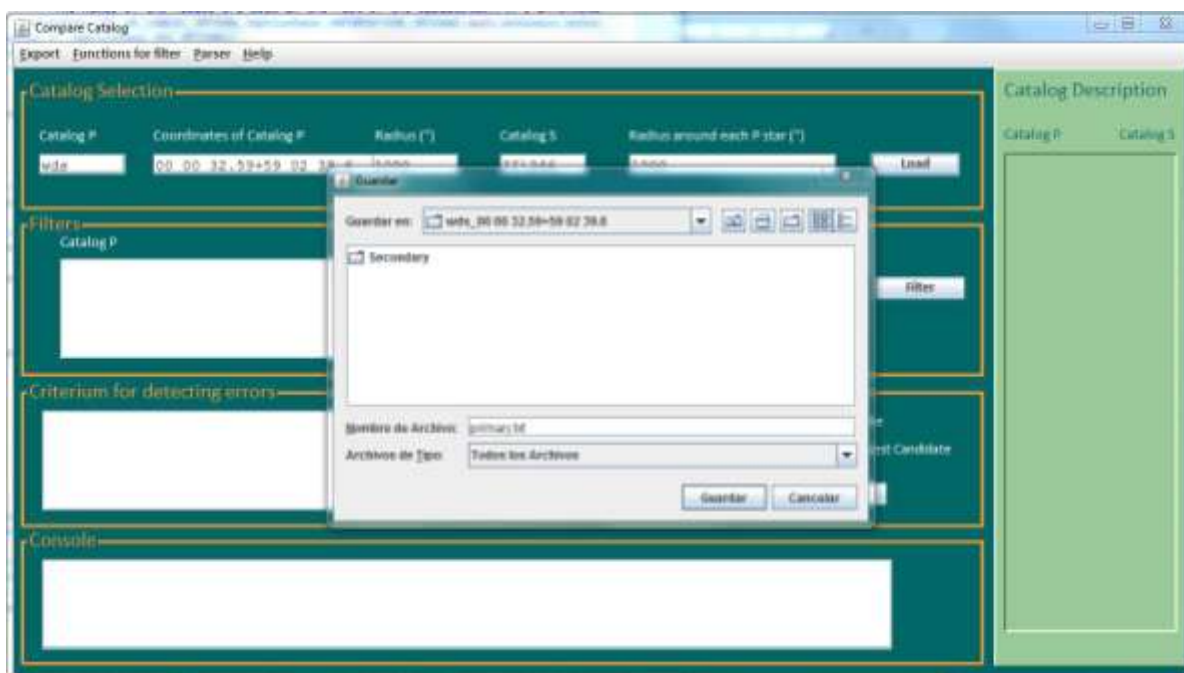


Figura 48. Paso 2

Se llamará al fichero *primay.txt*, que será el que guarde las estrellas del catálogo primario.

Después de guardar dicho fichero, se mostrarán en la sección *Catalog Description* los nombres de los campos de dichas estrellas.

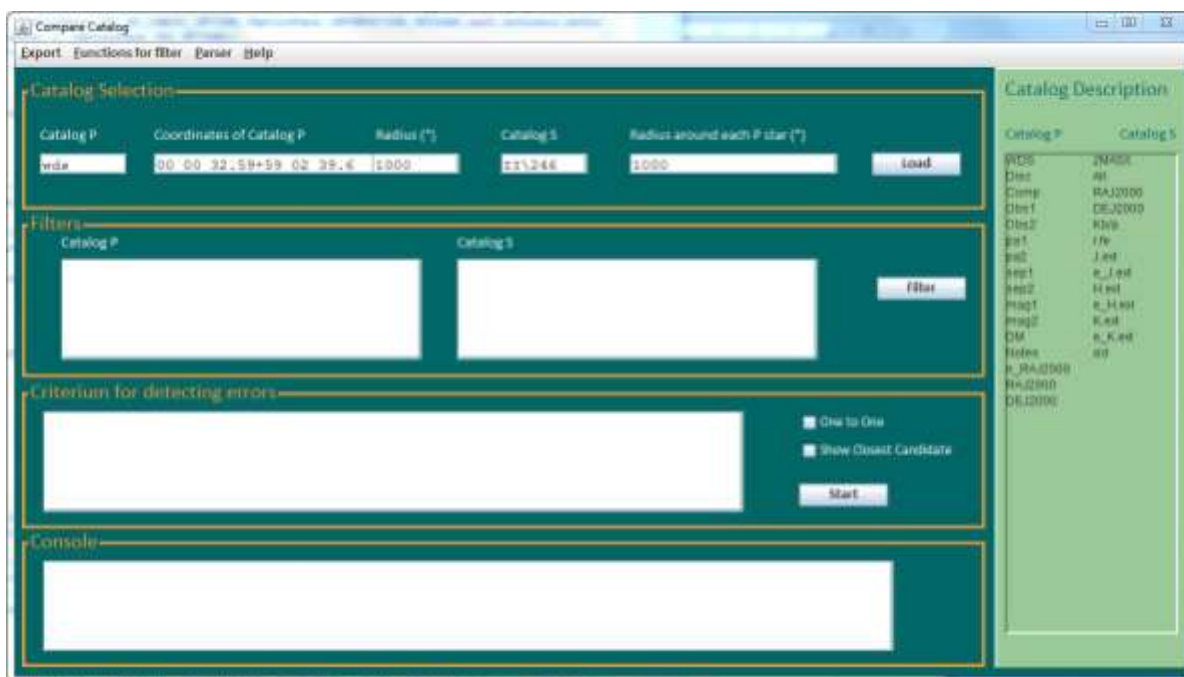


Figura 49. Paso 3

A continuación se muestra el contenido del directorio generado para estas coordenadas iniciales.

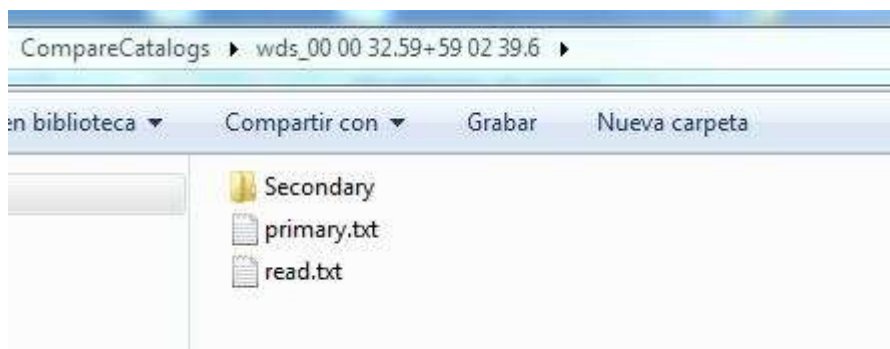


Figura 50. Paso 4

Después el usuario deberá rellenar los campos asociados de la sección *Filters*:

Primero se indicará el filtro para el catálogo primario.

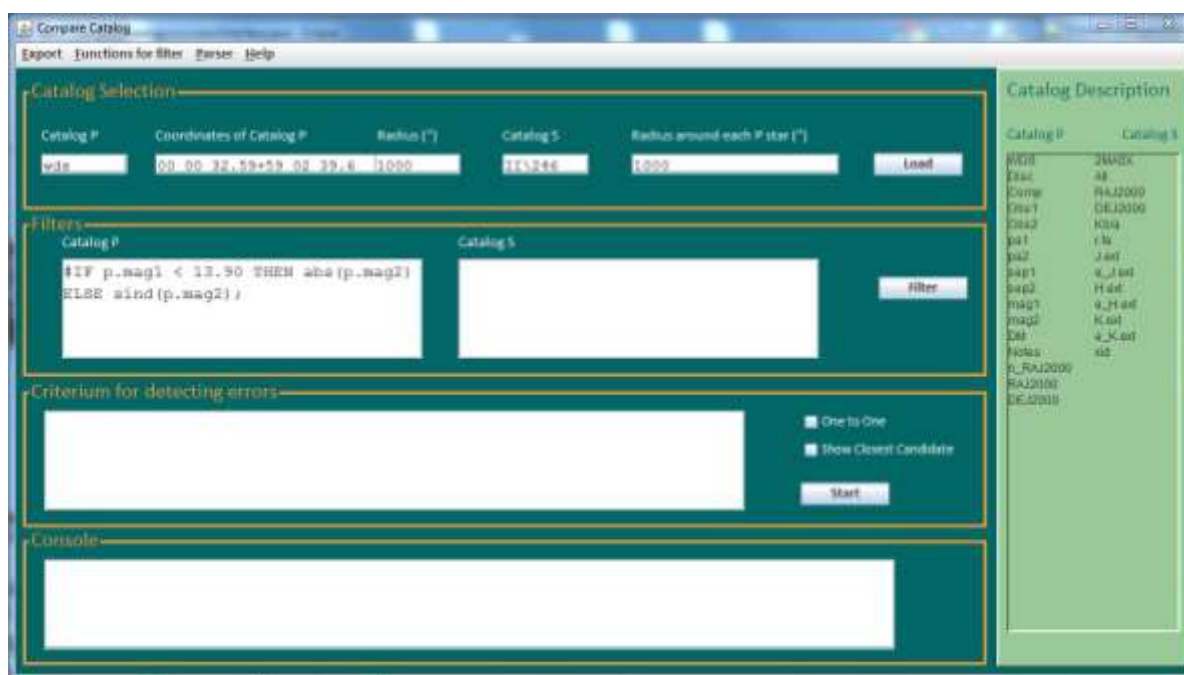


Figura 51. Paso 5

Una vez pulsado el botón *Filter*, se pasará a la comprobación sintáctica y semántica.

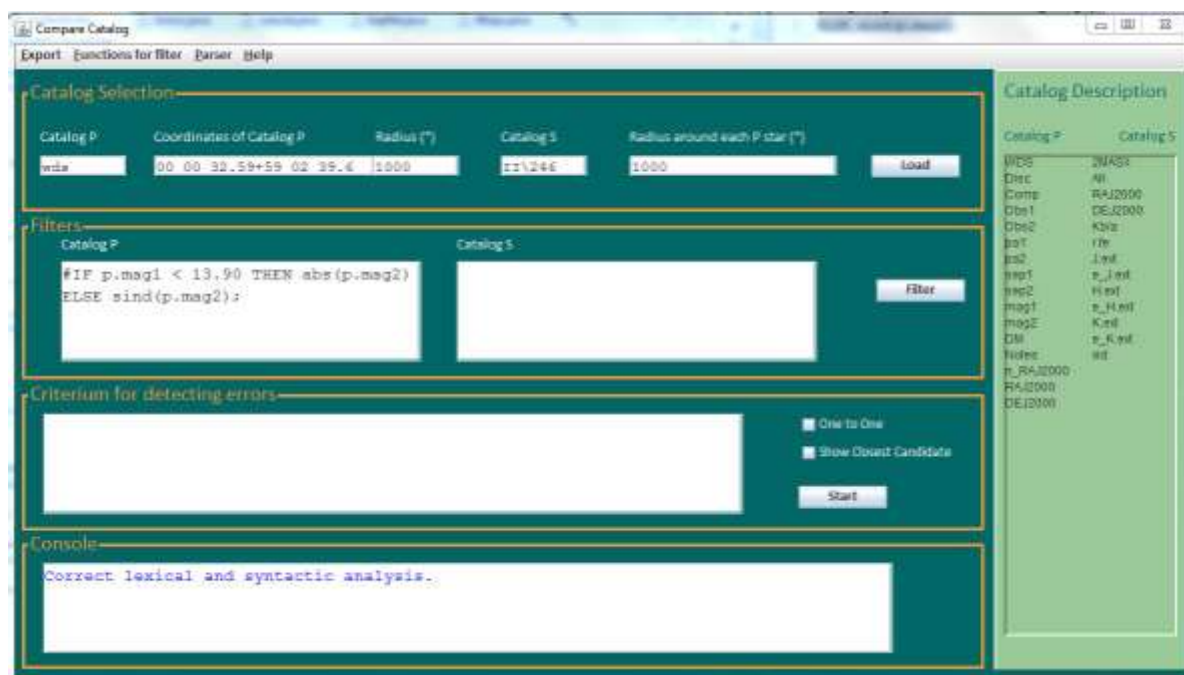


Figura 52. Paso 6

Habiendo comprobado que el filtro es correcto, se generarán los directorios y ficheros correspondientes. En la Figura 53, podemos observar un nuevo directorio llamado *Filtered_CatalogP*.

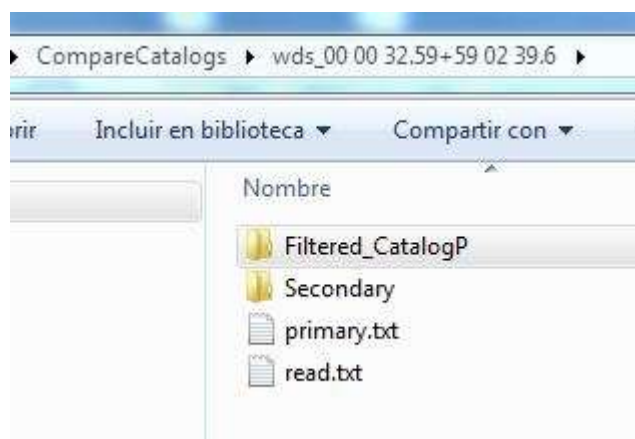


Figura 53. Paso 7

En la Figura 54 podemos ver el contenido de dicho directorio.

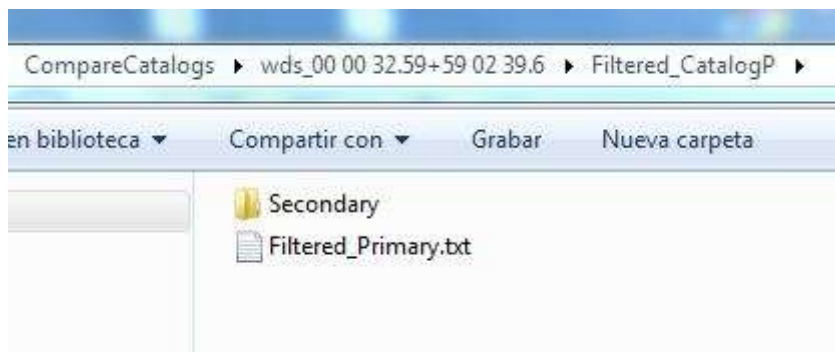


Figura 54. Paso 8

A continuación se indicará el filtro para el catálogo secundario, procediendo al proceso de análisis sintáctico y semántico.

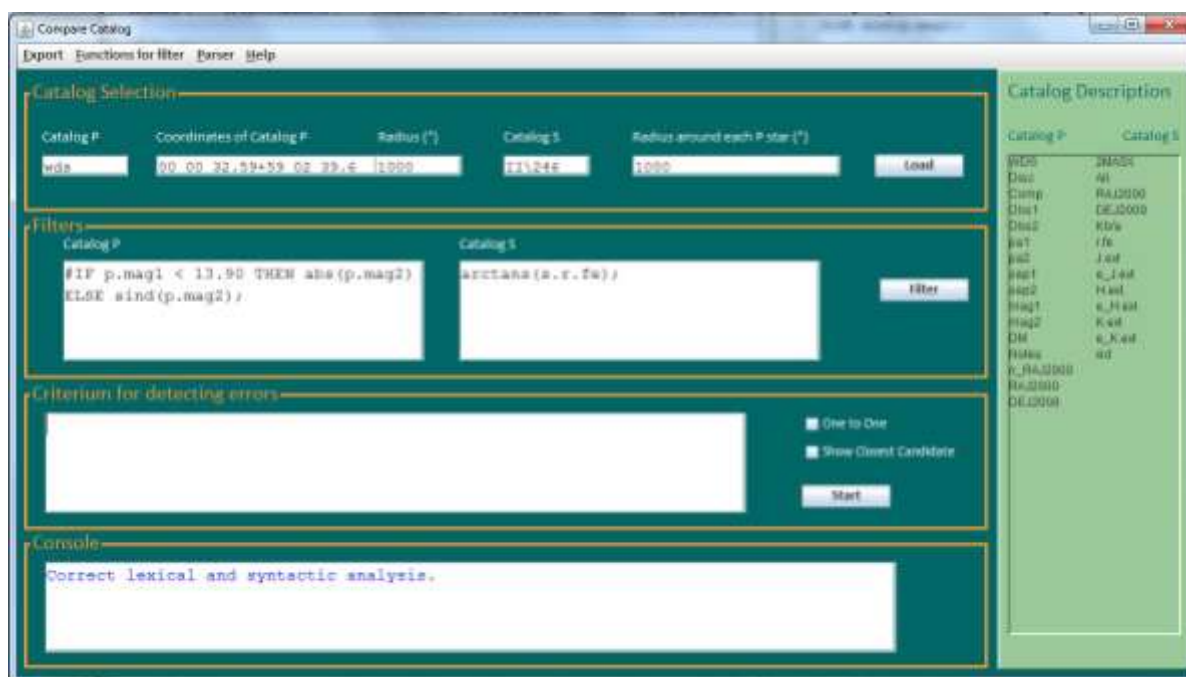


Figura 55. Paso 9

Generando un nuevo directorio *Filtered_CatalogS*:

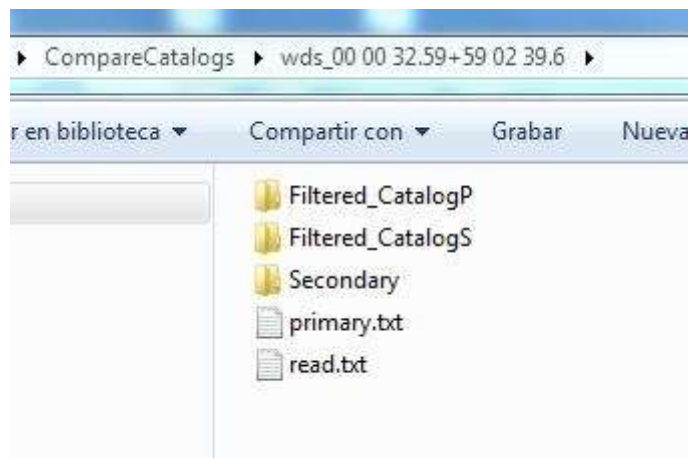


Figura 56. Paso 10

Con sus correspondientes ficheros:

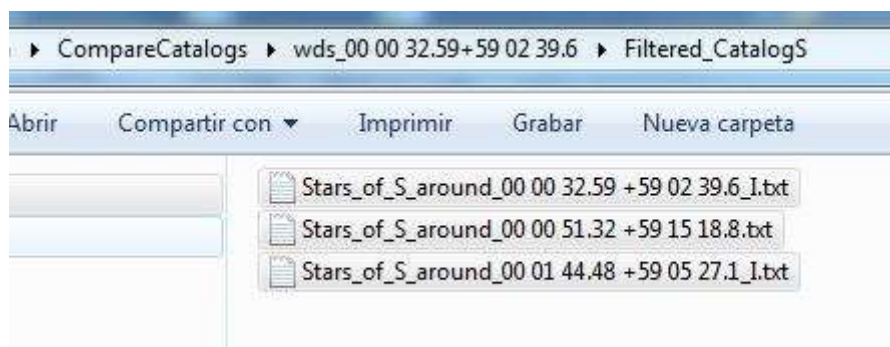


Figura 57. Paso 11

Como último paso, se podrá definir un caso de error, además se tendrá la opción de seleccionar el check-box *One to One* o *Show Closest Candidate*. Como observamos en la Figura 58 esta seleccionado *One to One*. Por tanto, sólo se tendrán en cuenta los casos en los que haya una estrella del catálogo secundario por cada una del primario.

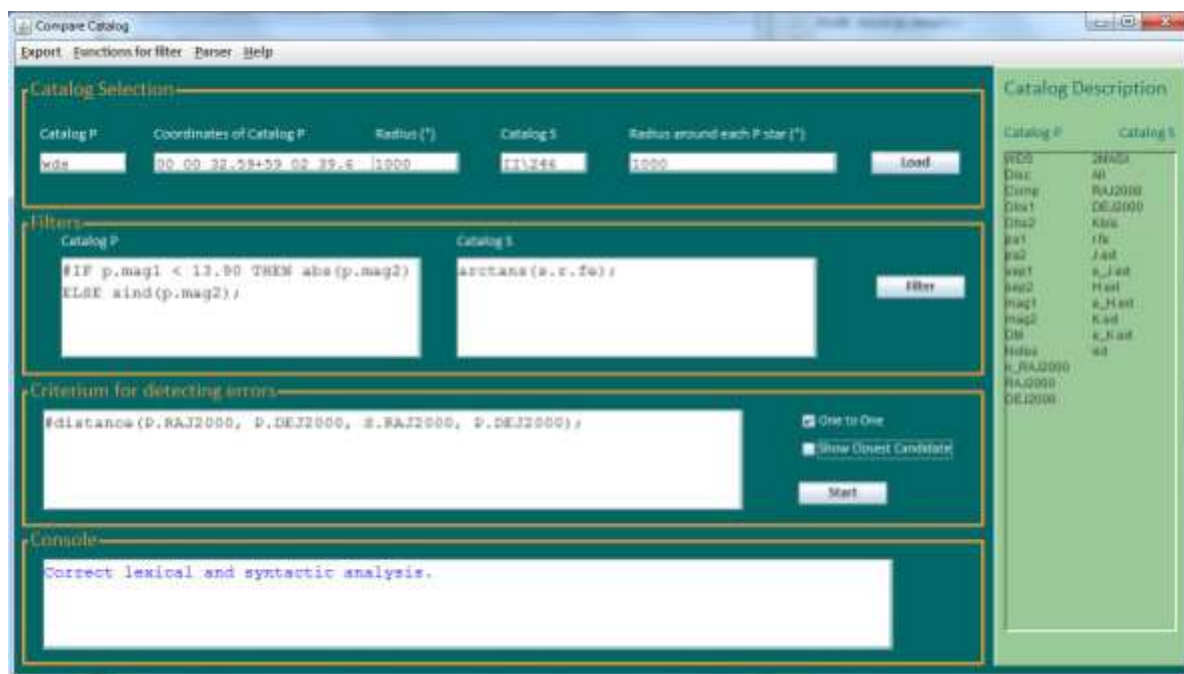


Figura 58. Paso 12

Dentro del directorio Errors, se generarán dos directorios *Error* y *Without_Error*,

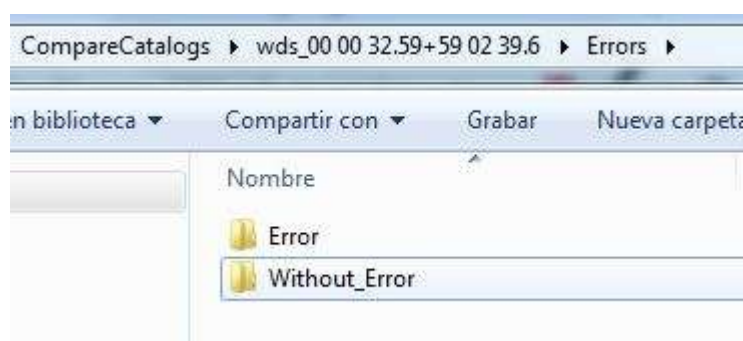


Figura 59. Paso 13

El directorio *Error*, contendrá los ficheros que cumplen dicho criterio.

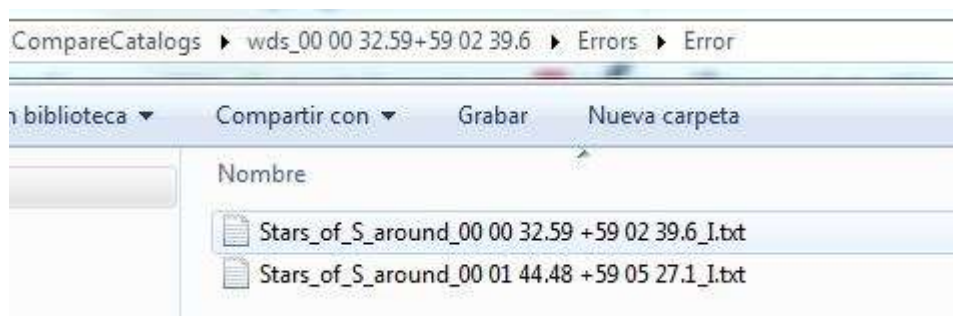


Figura 60. Paso 14

Y en el directorio *Without_Error*, los ficheros que no cumplen el criterio.

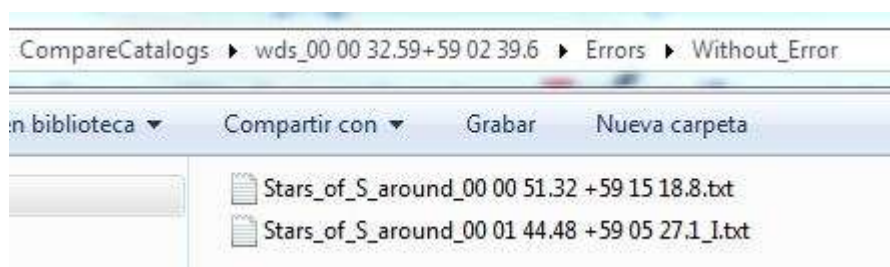


Figura 61. Paso 15

6. Conceptos de la carrera utilizados

Durante la realización de este proyecto hemos empleado los conocimientos adquiridos en asignaturas anteriores.

- Programación Orientada a Objetos: para diseñar ambas aplicaciones hemos usado este tipo de programación, por lo que dicha asignatura fue importante. Además, implementamos uno de los patrones aprendidos aquí.
- Laboratorios de programación: donde aprendimos diversas técnicas de programación
- Ingeniería del Software: sobre todo en lo que se refiere a organización y planificación de proyectos.
- Procesadores del Lenguaje: para realizar el parser de la segunda aplicación. No sólo utilizamos los conocimientos aprendidos, sino que también reutilizamos la práctica realizada en esta asignatura.

7. Conclusiones y trabajo futuro

Finalizado el proyecto podemos comentar las conclusiones a las que hemos llegado y el trabajo que esperamos se desarrolle en un futuro.

7.1. Conclusiones

Una vez realizado el estudio de las necesidades de los expertos en el momento de consultar y revisar los catálogos estelares y de las herramientas existentes que facilitan esta labor, se ha comprendido la importancia del desarrollo de una herramienta de apoyo.

Por ejemplo, en la página donde se encuentra *VizieR*⁷, se ofrece una herramienta para cruzar tablas, el *CDS X-Match Service*⁸. A pesar de las numerosas posibilidades que ofrece este servicio, sus consultas resultan muy costosas, tanto computacionalmente como para el usuario, e ineficientes.

La aplicación de consultas del catálogo *WDS*, fue más sencilla, tanto de entender como de implementar. Sin embargo, nos llevó mucho tiempo ya que según avanzábamos íbamos añadiendo más funcionalidades. Una vez finalizada se incluyó en el foro de estrellas dobles de la *Asociación Astronómica Hubble*⁹. Una de las extensiones realizadas fue por petición de astrónomos y aficionados a la astronomía de dicho foro, que consistía en poder exportar los resultados en un archivo Excel.

La aplicación *Combinación de catálogos astronómicos*, no fue tan sencilla. Nos llevó mucho tiempo entender el objetivo planteado por el director de proyecto y definir la idea final. El campo en el que nos movíamos era desconocido para nosotros. Con lo cual tuvimos que realizar un estudio previo, tanto de conceptos (estrella doble, coordenadas estelares, etc.) como la forma de tratarlos.

Las dos aplicaciones son software libre. La primera está disponible para su uso en *Google-Code*¹⁰, donde se encuentra el código de la aplicación. También dispone de un enlace¹¹ con el archivo ejecutable. Esta aplicación ya está siendo usada y con buenos resultados. El código de la segunda aplicación está disponible en el repositorio *GitHub*¹². Esperamos que esta segunda aplicación tenga tanta aceptación como la primera. Quedamos a la espera de las posibles críticas que nos puedan hacer desde el *USNO*.

⁷ <http://cdsportal.u-strasbg.fr/>

⁸ <http://cdsxmatch.u-strasbg.fr/xmatch>

⁹ <http://www.asociacionhubble.org/portal/index.php/foro/viewtopic.php?f=63&t=47758>

¹⁰ <https://code.google.com/p/wds-catalog-update/>

¹¹ <http://gpd.sip.ucm.es/rafa/astro/wds/wds.jar>

¹² <https://github.com/amdaza/Compare-Star-Catalogs>

8. Bibliografía

- USNO (Observatorio Naval de los Estados Unidos)
<http://www.usno.navy.mil/USNO>
- WDS (Washington Double Star Catalog)
<http://www.usno.navy.mil/USNO/astrometry/optical-IR-prod/wds/WDS>
<http://ad.usno.navy.mil/proj/WDS/Webtextfiles/wdsnewframe1.html>
- VizieR
<http://vizier.u-strasbg.fr/>
 - Formato de descarga utilizado
<http://vizier.u-strasbg.fr/doc/asu-summary.htx>
- Aladin
<http://aladin.u-strasbg.fr/aladin.gml>
- SDSS (Sloan Digital Sky Survey)
<http://www.sdss.org/>
- CDS X-Match Service
<http://cdsxmatch.u-strasbg.fr/xmatch>
- Escribir y Leer Archivos Excel en Java
<http://www.nosolounix.com/2010/05/escribir-y-leer-archivos-excel-en-java.html>
- Patrón Observer
[http://es.wikipedia.org/wiki/Observer_\(patr%C3%B3n_de_dise%C3%B1o\)](http://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o))
- Log4j para Creación de Eventos de Log
<http://www.javatutoriales.com/2011/04/log4j-para-creacion-de-eventos-de-log.html>
<https://logging.apache.org/log4j/1.2/download.html>
- Eclipse Modeling – UML2
<http://www.eclipse.org/modeling/mdt/downloads/?project=uml2tools>

- WindowBuilder
<https://projects.eclipse.org/projects/tools.windowbuilder>
- Foro de estrellas dobles en el Asociación Astronómica Hubble
<http://www.asociacionhubble.org/portal/index.php/foro/viewtopic.php?f=63&t=47758>
- Código proyecto en Google Code
<https://code.google.com/p/wds-catalog-update/>
- Código segunda aplicación en GitHub
<https://github.com/amdaza/Compare-Star-Catalogs>
- Descarga directa del ejecutable de la primera aplicación
<http://gpd.sip.ucm.es/rafa/astro/wds/wds.jar>