UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

VISUALIZACIÓN DE CONSULTAS SOBRE DATOS ASTRONÓMICOS EN UN WALL-DISPLAY

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL EN COMPUTACIÓN

AMANDA MARGARITA MEGAN IBSEN OSORIO

PROFESOR GUÍA:
NANCY HITSCHFELD KAHLER

MIEMBROS DE LA COMISIÓN:
BARBARA POBLETE LABRA
CLAUDIO GUTIÉRREZ GALLARDO

SANTIAGO DE CHILE
MARZO 2016

Resumen

Gracias al avance tecnológico en radioastronomía y en almacenamiento de datos, el estudio de objetos estelares está creciendo en Chile y en el mundo. Gran cantidad de información astronómica es recopilada y guardada diariamente en distintos formatos. FITS es el formato de imagen más utilizado en el ámbito científico, pues permite guardar información multidimensional (espectros en una dimensión, mapas estelares en dos dimensiones o cubos de datos de tres o más dimensiones) junto con un encabezado con referencias de metadata pertinente. Las imágenes FITS generalmente exceden en tamaño la resolución de un monitor de escritorio normal (de 1080p, por ejemplo), por lo que no es posible desplegarlas completas en una razón 1:1 entre los pixeles del monitor y los pixeles en la imágen. Dado que una parte importante del análisis de datos astronómicos aún no es un proceso automatizado, esto puede llevar a la pérdida de percepción de detalles al momento de analizar la imagen. Bajo este contexto y a pedido del Instituto de Astrofísica de la Universidad Católica, en conjunto con la fundación Inria Chile y el instituto de investigación Inria Francia, en particular con el equipo *Massive data* (en Chile) y el equipo *Ilda* (en Francia), se desarrolló una aplicación llamada **FITS-OW**. Esta aplicación tiene por objetivo visualizar imágenes astronómicas en formato FITS en un *wall display*, es decir, en un arreglo de monitores manejado por un *cluster* de computadores. La aplicación fue desarrollada utilizando el *Andes wall display* compuesto de trece computadores y veinticuatro monitores, ubicado en las oficinas de Inria Chile.
La presente memoria consiste en el diseño e implementación de la segunda etapa de desarrollo de **FITS-OW**. Esta etapa consiste en construir sobre la aplicación existente para poder conectar el sistema a la base de datos de objetos astronómicos extrasolares SIMBAD. Con esta contribución, la aplicación permite al usuario realizar consultas a esta base de datos externa respecto de la imagen FITS visualizada. Se delineó una interfaz de usuario compatible con el *hardware* utilizado, que permite definir distintos filtros y parámetros para las consultas y que despliega los resultados obtenidos desde la base de datos de manera intuitiva y coherente. Se definieron distintos tipos de criterios de filtro de consulta, que permiten seleccionar conjutos de objetos celestes con características particulares o con mediciones correspondientes a un catálogo específico. Por ejemplo, un usuario puede visualizar una imagen astronómica y consultar a SIMBAD por todos los objetos astronómicos en cierta región con cierto tipo espectral.
El desarrollo de la aplicación continuará además en una tercera etapa, en donde se implementará la conexión con otras bases de datos astronómicas externas y donde se le dará al usuario la posibilidad de contar con una base de datos local.

Esta memoria va dedicada a mi papá, Jorge Ibsen.

Agradecimientos

A lo largo del estudio de mi carrera muchos aspectos de mi vida fueron cambiando, a veces gradualmente y a veces inesperadamente. Durante todo este proceso tuve la suerte de contar con personas que me apoyaron y que me ayudaron en varios sentidos distintos.

Agradezco en primer lugar a mis abuelos, Cristina Herrera y Hermy Osorio, que me dieron apoyo incondicional cuando más lo necesité. A mis tías, Jimena Osorio y Cristina Osorio, y a mis primos, Cristina Belén Peña y Mauricio Peña, que me entregaron siempre todo su cariño.

Debo agradecer también a mi mamá, Marcela Osorio. Si bien hemos tenido muchas diferencias, estas me han hecho reflexionar profundamente respecto de mi vida y al final me han vuelto una persona más fuerte.

Hago una mención especial a mis amigos, que hicieron de mi paso por la universidad una experiencia mucho más amena: Elisa Anguita, Bernardo Küpfer, Fernando Navea, Tomás Wolf, Victor Caro, Nelson Higuera, Fernanda Muñoz, Benjamín Holloway y muchos otros, que demoraría mucho en listar. Principalmente agradezco a mi pololo, Mario Garrido, que ha sido un excelente amigo desde que lo conozco y que es también mi mejor amigo hoy.

Hay que mencionar también al equipo de INRIA. Especialmente agradezco a Emmanuel Pietriga, que me guió a lo largo del desarrollo de esta memoria, a Fernando del Campo, que me ayudó cada vez que tuve dudas técnicas y a Claude Puech, que me facilitó trabajar en INRIA al mismo tiempo que estudiaba. Agradezco también a José Miguel Piquer, que si bien ya no trabaja en INRIA, fue quien me dió la posibilidad de trabajar ahí en primer lugar.

Finalmente, agradezco a mi papá, Jorge Ibsen, que me regaló la carrera y a quién debo mi educación. Él influyó en gran medida mi interés por la ciencia y me ayudó cada vez que pudo a impulsar mi desarrollo profesional, entregándome un sin fin de oportunidades. No sería la persona que soy sin él.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Introducción y Motivación

La astronomía es un campo de estudio en crecimiento en Chile y en el mundo, día a día se producen grandes volúmenes de datos: sólo en ALMA (Atacama Large Millimeter Array)[1] se producen 250 Terabytes anuales, y eso es poco en escala, considerando que en ASKAP (Australian Square Kilometre Array Pathfinder)[2] se producen 2.5 Gbytes por segundo (75 Petabytes anuales). Toda esta información producida en masa es guardada en distintas bases de datos y catálogos astronómicos alrededor del mundo.

Esta información necesita ser estudiada y analizada para avanzar en nuestro entendimiento del universo[5]. Si bien parte importante de este trabajo se realiza de manera automatizada utilizando inteligencia computacional, estos métodos no son perfectos y una porción no menor del análisis aún se hace de forma no automática por astrónomos y otros científicos. Es aquí donde la visualización de la información toma importancia, pues el poder apreciar gráficamente los datos de tipo astronómico de manera intuitiva es un aspecto clave para poder reconocer patrones y realizar estudios sobre los mismos[4].

Los observatorios actuales recogen datos que se guardan en distintos formatos. Existen bases de datos que almacenan mediciones y atributos de los objetos estelares en formato de texto, como por ejemplo SIMBAD (Set of Identifications, Measurements, and Bibliography for Astronomical Data)[3]. Otras bases de datos, como Aladin, alamacenan observaciones en formato de imagen. El formato de imagen para observaciones astronómicas más utilizado actualmente es el formato FITS (Flexible Image Transport System), que almacena un encabezado con metadata (como por ejemplo, el tipo de coordenadas utilizadas) y un bloque de datos. El bloque de datos es un arreglo de dimesión arbitraria, aunque normalmente se utilizan dos o tres dimensiones, con la tercera dimensión representando, por ejemplo, un filtro de color. Las imágenes capturadas y guardadas como FITS generalmente exceden la resolución de un monitor de escritorio común. Esto se traduce en que cuando un astrónomo desea observar una imagen completa, generalmente debe hacerlo desplegándola en menor resolución, lo que puede llevar a que pierda de vista algunos detalles.

Dado que una buena visualización de la información puede hacer una diferencia sustancial en los resultados del estudio realizado y por lo tanto en nuestro desarrollo científico, es importante familiarizarse con las técnicas de visualización utilizadas actualmente. Bajo este contexto, se considera la posibilidad de desarrollar aplicaciones que ayuden en el estudio astronómico y que utilicen tecnologías innovadoras. Por ejemplo, resulta interesante la alternativa de desarrollar *software* diseñado para correr en *hardware* más apto para desplegar las imágenes en formato FITS, como un *wall display*[15].

Un *wall display* es un arreglo de monitores dispuesto de manera en que todas las pantallas en conjunto formen una sola gran pantalla. Este arreglo de monitores es controlado por un *cluster* de varios computadores con capacidad gráfica adecuada. Utilizando este tipo de estructura es posible desplegar una imagen FITS de gran tamaño completa, sin pérdida de detalles.

La fundación INRIA Chile[6], a pedido del Instituto de Astrofísica de la Universidad Católica de Chile, y en colaboración con el instituto de investigación INRIA Francia, desarrolló un *software* de visualización de imágenes astronómicas en formato FITS, llamado **FITS-OW**. Dado el tamaño de tales imágenes, la aplicación está pensada para ser utilizada principalmente en un *wall display* de alta resolución y no en un computador de escritorio, aunque también es posible correrla en uno. Este *software*, desarrollado a modo de primer prototipo, es capaz de desplegar la imagen y de aplicarle distintos filtros de color y escalas matemáticas.

Luego de varias conversaciones con los astrónomos de la Universidad Católica, se determina darle inicio a una segunda etapa de desarrollo a **FITS-OW**, que corresponde al presente trabajo de memoria, guiado por Emmanuel Pietriga. El equipo del Instituto de Astrofísica utiliza ampliamente la base de datos de objetos astronómicos extra solares SIMBAD, por lo que el proyecto consiste en equipar **FITS-OW** con una manera de consultar esta base de datos desde la misma aplicación. La idea es que **FITS-OW** sea capaz no sólo de visualizar imágenes, sino también de marcar en ella los objetos desplegados que estén guardados en SIMBAD, si el usuario así lo desea. Para facilitar el estudio de la imagen desplegada, se requiere también que **FITS-OW** permita hacer consultas con distintos filtros y criterios, desplegando así un subconjunto de los objetos en la imagen con características específicas o un objeto particular. Por ejemplo, dado un punto y un radio, resulta útil encontrar todos los objetos celestes de la vecindad. Además es conveniente desplegar información sobre un objeto en particular una vez encontrado (como su velocidad radial, brillo, etc) en paralelo con la imagen astronómica.

Se desea entonces implementar sobre la aplicación ya existente una interfaz gráfica que permita al usuario hacer consultas a la base de datos astronómica SIMBAD y visualizar los resultados obtenidos de forma intuitiva y útil. Para desarrollar esto, se utiliza ZVTM[12], una herramienta implementada en *Java* de desarrollo de interfaces de usuario que permite trabajar con grandes cantidades de datos eficientemente y que soporta el uso de *wall displays*, aunque también es ejecutable sobre un computador con una sóla pantalla.

ZVTM está hecha para trabajar con visualizaciones de datos en 2D y se basa en la metáfora de tener varios espacios virtuales a modo de universos separados. Cada uno de estos universos

es observado con una o más cámaras virtuales móviles; de esta forma se logra continuidad en la percepción de animación de objetos y movimientos de cámara. En esta herramienta cada objeto es representado como un *glyph* (objeto renderizable) que puede personalizarse. ZVTM puede desplegar imágenes, vectores gráficos, texto, aplicaciones y hasta documentos PDF. Para correr ZVTM en un *wall display* se utiliza la librería *JBricks*, que facilita el desarrollo, pues se encarga de coordinar la sincronización de renderización entre las pantallas.

La aplicación se desarrolla utilizando el Andes *wall display*[16] ubicado en las oficinas de la fundación INRIA Chile. **FITS-OW** debe adaptarse a esta forma de interacción con los datos para hacer uso provechoso del potencial del *wall display*. Andes *wall display* está compuesta por 24 paneles LED (la resolución es de 11520 x 4320 pixeles) manejados por un *cluster* de 13 computadores (12 de ellos son coordinados por el número trece).

1.2. Objetivo General

El objetivo de esta memoria es construir una interfaz gráfica sobre la aplicación **FITS-OW**, de visualización de imágenes astronómicas, que permita al usario realizar consultas respecto de una imagen FITS a la base de datos astronómica SIMBAD y que despliegue los resultados en paralelo a la imagen, de forma interactiva. La interfaz debe correr tanto en un computador de escritorio como en un *wall display*.

1.3. Objetivos Específicos

Con el fin de lograr el objetivo general, se definen a continuación varios objetivos específicos que deben cumplirse:

- Implementar las consultas que la base de datos debe ser capaz de realizar:
  - Consulta por coordenadas.
  - Consulta por identificador del objeto.
  - Consulta por *script*.
  - Filtrado por tipo de objeto astronómico.
  - Filtrado por rango y calidad de medición del movimiento propio del objeto.
  - Filtrado por rango y calidad de medición del paralaje del objeto.
  - Filtrado por rango y calidad de medición de la velocidad radial de objeto.
  - Filtrado por clase de temperatura, clase de luminosidad, peculiaridades y calidad de medición del tipo espectral del objeto.
  - Filtrado por rango del flujo del objeto.
  - Consultas acerca de mediciones en catálogos específicos.
- Diseñar e implementar las vistas necesarias de la interfaz de usuario, tanto para realizar consultas como para desplegar resultados.

- Diseñar e implementar las estructuras de datos pertinentes para almacenar las respuestas obtenidas y poder manejarlas como objetos.

1.4. Metodología

Para realizar el trabajo de memoria, la metodología a seguir es la siguiente:

- Revisar el estado del arte relevante.
- Estudiar la estructura de los datos astronómicos retornados por las consultas a SIMBAD.
- Diseñar las interfaces de usuario y el sistema a implementar de acuerdo a lo estudiado.
- Implementar una interfaz de usuario funcional. Para esto es necesario:
  - Diseñar e implementar una interfaz de usuario que permita realizar consultas con distintos filtros y que despliegue los resultados encontrados en paralelo con la imagen FITS visualizada, permitiendo interacutar con ellos de manera clara. Dado que el *Andes wall display* no cuenta con los dispositivos típicos de interacción con un computador (teclado y *mouse*) el manejo del cursor será vía dispositivos móviles como *smartphones* y *tablets*.
  - Construir un *parser* que transforme las opciones ingresadas por el usuario en consultas que la base de datos SIMBAD pueda entender.
  - Construir las estructuras de datos necesarias para almacenar los resultados de SIMBAD y poder manejarlos como objetos.
  - Implementar un *parser* que traduzca las respuestas de la base de datos en las estructuras de datos implementadas. Las respuestas de la base de datos son en texto plano ASCII, en la forma de un sólo objeto o de un listado de estos. En ambos casos estos resultados contienen parámetros generales como el nombre y el formato. Además, cada objeto tiene como atributo un tipo, un conjunto de coordenadas, un paralaje, un movimiento propio, velocidad radial, tipo espectral, flujo y distintas mediciones.

1.5. Resultados Esperados

El resultado esperado de esta memoria es un sistema que corra tanto en un *wall display* como en un computador de escritorio y que permita al usuario visualizar una imagen astronómica en formato FITS y realizar consultas respecto de ellas a la base de datos externa SIMBAD. Estas consultas deben poder efectuarse a través de una interfaz gráfica que permita definir distintos filtros y parámetros. Además, los resultados de las consultas deben visualizarse de forma coherente e intuitiva en paralalelo a la imagen.

1.6. Alcances

Debido a las limitaciones de tiempo para el desarrollo de una memoria (seis meses), el problema de la conexión de **FITS-OW** a una base de datos astronómica externa se limitará sólo a SIMBAD. Otras bases de datos astronómicas, como *Aladin* o bases de datos locales se contemplarán para trabajos futuros.

1.7. Contenidos de la Memoria

La presente memoria cuenta con varios capítulos que describen el diseño e implementación de la segunda etapa de desarrollo de **FITS-OW**. Para dar una idea más completa del problema, se comienza por una descripción del proyecto, en donde se describe el objetivo específico y los objetivos generales, se especifican los resultados esperados y de discuten los alcances del trabajo.

Con el fin de entregar el contexto del problema y la información necesaria para entender la solución entregada, se cuenta con un segundo capítulo dedicado al marco conceptual. En él se definen los conceptos involucrados en el problema, se analiza el estado del arte existente y se exponen las herramientas utilizadas en la solución.

El tercer capítulo corresponde a la especificación del problema, en donde se discute su relevancia, se establece el usuario objetivo, exponen los antecedentes necesarios, se detallan los requisitos y se precisan los casos de uso de la solución.

La solución misma se describe en el cuarto capítulo, en donde se detallan las componentes de la arquitectura de *software* y de *hardware* utilizadas.

En el capítulo cinco se expone el funcionamiento de la aplicación en detalle, explicando todas las funcionalidades de usuario añadidas a **FITS-OW**, con pantallazos de las interfaces de usuario pertinentes. Además se realiza una evalución a la solución entregada.

Finalmente, en el capítulo de conclusión, se analiza el cumplimiento de los objetivos, se discuten las dificultades encontradas a lo largo del proyecto y se habla respecto del trabajo futuro y escalabilidad de **FITS-OW**.

Capítulo 2

Marco Conceptual

En este capítulo se entregará todo el contexto necesario para entender a cabalidad tanto el problema como la solución del trabajo de memoria.

2.1. Conceptos involucrados

A continuación se detallan las nociones implicadas en el proyecto.

2.1.1. Conceptos de astronomía

Catálogo astronómico

Un catálogo astronómico es una lista o tabulación de objetos astronómicos, agrupados porque pertenecen a un tipo u origen común, o porque comparten el método de detección que se utilizó para recoger mediciones respecto de ellos. Los catálogos astronómicos generalmente son el resultado de un estudio particular del cielo.[19]

Coordenadas Astronómicas

Existen distintos sistemas de referencia[20] para especificar la posición en el cielo de los objetos celestes. Estos sistemas de referencia utilizan coordenadas esféricas, por lo que quedan definidos con un origen y un plano. Adicionalmente, se especifica un equinoccio y una época, que sirven para determinar el momento de tiempo en el que el objeto celeste estuvo en una posición dada. El equinoccio especifica el momento de tiempo en el cual el origen del sistema y el plano estuvieron en las coordenadas determinadas, por lo que para corregir las observaciones, se debe tomar en consideración la presición de la observación, la nutación y la aberración. La época determina el instante en el cual la posición dada para un objeto celeste

es válida, es decir, al corregir las observaciones se debe tomar en cuenta el movimiento propio, el paralaje y otros aspectos relacionados con las órbitas de los objetos.

Los sistemas de coordenadas más usados son:

- Sistema ecuatorial: Toma como origen el centro de la tierra y como plano el ecuador celeste, fijo en relación a estrellas lejanas y otras galaxias. Las coordenadas están basadas en la ubicación de las estrellas en relación al ecuador si este se proyectara hacia el infinito. Describe el cielo visto desde el sistema solar y la mayoría de los mapas estelares modernos usan exclusivamente este tipo de coordenadas.

  Este sistema tiene por latitud en ángulo de declinación δ y por longitud el ángulo de ascención derecha α.
- Sistema eclíptico: El plano fundamental es la eclíptica, es decir, el plano orbital de la tierra. Hay dos variantes de este tipo de coordenadas, las geocéntricas y las heliocéntricas. Las coordenadas eclípticas geocéntricas fueron utilizadas en la antiguedad para calcular los movimientos del sol, de la luna y los planetas. Las coordenadas eclípticas heliocéntricas son principalmente utilizadas para calcular la posición de los planetas y otros objetos del sistema solar y para definir sus respectivos elementos orbitales. La latitud corresponde a la latitud eclíptica β y la longitud a la longitud eclíptica γ.
- Sistema galáctico: Utiliza el plano galáctico como plano fundamental. El centro del sistema solar sigue siendo el origen, el punto cero se define en dirección al centro galáctico. La latitud se simboliza con b y la longitud con l.
- Sistema súper galáctico: El plano fundamental corresponde a un plano que contiene una alta concentración de galaxias locales en el cielo, visto desde la tierra.

Además de los sistemas de coordenadas, es necesario definir un equinoccio y una época para referenciar los objetos celestes. Un sistema de coordenadas astronómicas con estos dos valores definidos se denomina *frame*. Algunos de los *frames* más usados son:

- ICRS (International Celestial Reference System): Es el sistema de referencia celeste estándar adoptado por la IAU (International Astronomical Union). Su origen es en el centro de masa del sistema solar, por lo que es aproximádamente el mismo que el de las coordenadas ecuatoriales. Este sistema de referencia presenta un *frame* cuasi-inercial, definido por las posiciones medidas de 212 fuentes extra-galácticas (principalmente quasars). Esto quiere decir que no muestra movimiento angular medible, dado que las fuentes extragalácticas utilizadas para definir el sistema son muy lejanas.
- FK5 (Fifth fundamental catalog): Provee posiciones y movimientos propios promedio para equinoccio y época J2000 (1^{ero} de Enero del 2000, a las 12:00 UT).
- FK4 (Fourth fundamental catalog): Provee posiciones y movimientos propios promedio para equinoccio y época B1950 (31 de de Diciembre de 1950, a las 22:09 UT).

Identificadores astronómicos

Un identificador es un código que se le entrega a un objeto astronómico para indexarlo en una base de datos [21]. Un objeto astronómico puede estar asociado a más de un identificador,

ya que puede estar indexado en distintos catálogos con nombres distintos, o puede tener más de un nombre que guarde distintos significados. Por ejemplo, *Sirius* es llamada también $**AGC1$, un código que representa que es una estrella binaria indexada por Clark A. C. *Sirius* está también relacionada con los identificadores **alf CMa* y ** 9 CMa* de las estrellas que la componen y cuenta también con otros identificadores según el catálogo en el que se busque.

2.1.2. Conceptos tecnológicos

FITS (Flexible Image Transport System)

Es el formato digital más utilizado en astronomía para el almacenamiento, transmisión y procesamiento de imágenes. FITS [22] está diseñado especialmente para guardar datos científicos y por esto cuenta con facilidades para describir información fotométrica y de calibración espacial, además de metadata. El formato fue estandarizado en 1981 y ha ido evolucionando gradualmente desde entonces. La versión más reciente corresponde al año 2008. FITS tiene como objetivo el almacenamiento a largo plazo de la información, por lo que cada nueva versión del formato es compatible con las anteriores.

Cada archivo FITS cuenta con uno o más encabezados que contienen metadata en ASCII, por lo que su lectura es entendible para los usuarios y sirve para que una persona interesada en el archivo, que no sepa de su procedencia, pueda obtener un poco más de información respecto de él. Los encabezados tienen 80 caracteres de largo que codifican pares de palabras clave y valores y están intercalados con bloques de datos. Estos pares llave/valor proveen información respecto del origen, coordenadas, tamaño, o cualquier otro atributo relevante a la información que el creador estime conveniente.

El formato FITS es también usado para almacenar información que no necesariamente se ve como una imagen en dos dimensiones, como espectros, cubos de datos u otros tipos de estructuras de datos. El tipo de archivo FITS más utilizado es el de una imagen de dimensiones arbitrarias compuesta por un encabezado y un bloque de datos. El bloque de datos puede componerse de valores enteros, reales o de otro tipo, especificado en el encabezado.

En general, en astronomía el encabezado contiene información acerca de uno o más sistemas de coordenadas en la imagen. Las imágenes además contienen un sistema de coordenadas cartesianas implícito, que describe la ubicación de cada pixel en la imagen. Es muy común que en el encabezado se encuentre la palabra clave WCS (World Coordinate System), cuyo valor establece una relación entre las coordenadas de los pixeles y las coordenadas celestes.

Wall Display

Un *wall display*[23] es un una plataforma de *hardware* compuesta por un arreglo de monitores de alta resolución manejado por un *cluster* de computadores. Presenta una gran densidad de pixeles sobre una gran superficie física, típicamente del tamaño de una pared

(de ahí el nombre). Por ejemplo, WILD[24], el primer *wall display* construido en INRIA en Francia, tiene una resolución total de $20480 \times 6400 = 131$ megapixeles, en una superficie de $5,5m \times 1,8m$.

Este tipo de plataformas cuentan con propiedades que las vuelven ideales para la visualización de grandes conjuntos de datos. Pueden presentar la información con un gran nivel de detalle a la vez que mantienen en foco el contexto global, ya que los usuarios pueden pasar desde un nivel de detalle a otro simplemente acercándose a las pantallas. Otra funcionalidad destacable de los *wall display* es que ofrecen un buen soporte para el trabajo colaborativo, ya que permiten a varios usuarios visualizar e interactuar con los datos simultáneamente. Dado que un *wall display* no está pensado para usarse con teclado y *mouse* (no está sobre un escritorio y se usan por más de un usuario), para poder interactuar con ellos se usan otros tipos de tecnologías, como interfaces táctiles, sistemas de detección de movimiento o dispositivos *touch* inalámbricos.

Las áreas de aplicación para este tipo de plataformas de visualización son variadas y van desde herramientas para el monitoreo de infraestructuras complejas (como por ejemplo, un sistema de transporte público) hasta herramientas para la visualización exploratoria de datos científicos (como por ejemplo, datos astronómicos [15]).

Cluster de computadores

Un *cluster* de computadores es un conjunto de computadores conectados que trabajan juntos y que funcionan como un sólo sistema. En un *cluster*, cada computador resuelve una parte del mismo problema, y se coordinan a través de un *software* diseñado para esto. En general, en un *cluster* existe un computador que sincroniza al resto, llamado *master*. El resto de los computadores son llamados *slaves*.

Usualmente, los computadores se conectan por LAN (Local Area Networks) y cada nodo (computador que funciona como servidor) corre su propia instancia de sistema operativo. Es común que todos los nodos usen el mismo tipo de *hardware*.

2.2. Estado del Arte

Hoy en día existen numerosas bases de datos astronómicas públicas, dedicadas a guardar distintos tipos de observaciones indexadas en catálogos. Además se cuenta con una variedad de *software*, público y pagado, cuyo objetivo es facilitar el análisis de datos astronómicos. El enfoque de estos *software* se encuentra principalmente en consultas a bases de datos y visualizaciones de información. Hay una gran variedad de *software* dedicado a la astronomía y muchos de estos sistemas son versiones distintas de las mismas funcionalidades. En la mayoría de los casos, la razón para que existan diferentes alternativas es simplemente la preferencia de los usuarios en cuanto a plataformas y usabilidad.

2.2.1. Mapeos del cielo

Los *software* listados a continuación tienen como funcionalidad principal crear un mapa de alguna sección del cielo en alguna parte del espectro electromagnético. Son publicaciones de conjuntos de datos de distintos tipos en una región particular del cielo.

SDSS DR12 (Sloan Digital Sky Survey, Data Realease 12)

El SDSS[25] es un proyecto de investigación dedicado a producir un mapa de un cuarto del cielo. La última actualización incluye cuatro tipos de información: imágenes, espectro óptico, espectro infrarrojo e información de catálogos (parámetros medidos de imágenes y espectros, como magnitudes y corrimientos al rojo). La información de esta actualización es multiplataforma, ya que puede accesarse mediante varios clientes web:

- *Science Archive Server*: Provee una visualización interactiva de espectros e imágenes en mosaicos.
- *SkyServer*: Provee acceso a CAS (*Catalog Archive Server*), con recursos para aprender sobre SQL y proyectos para enseñar ciencia.
- *CasJobs*: Es una interfaz basada en SQL para accesar CAS. Requiere registrarse.
- DR12 FITS: Permite el acceso y descarga directa de imágenes FITS.
- *Data Model*: Entrega información respecto de la estructura, los formatos y el contenidos de variadas imágenes.

Two Micron All Sky Survey at IPAC (2Mass)

El 2MASS[26] es un proyecto de investigación dedicado a producir un mapa del cielo completo en espectro casi infrarrojo, utilizando las tecnologías actuales para completar la información que se tenía desde el último mapeo celeste en este espectro (hace más de 30 años). Este proyecto tiene como productos un atlas digital del cielo, un catálogo de fuentes (300 millones de estrellas y otros objetos) y un catálogo de fuentes extendido, que contiene posiciones y magnitudes de más de 1.000.000 galaxias y otras nebulosas. Es multiplataforma, pues se accede a través de la web.

Para accesar la data obtenida por este proyecto se puede utilizar la interfaz web o a través del servicio *VizieR*.

2.2.2. Visualizaciones y simulaciones de cielo

Los *software* listados a continuación tienen como funcionalidad principal visualizar una versión virtual del cielo, compuesta de mosaicos de imágenes. El usuario puede navegar a través de estas visualizaciones en tres dimensiones.

Celestia

Es un simulador multiplataforma del espacio en tiempo real, que permite experimentar el universo en 3D [27].

Orbiter

Es un *software* gratis que simula el viaje espacial a través del sistema solar, poniendo especial atención en que la física sea precisa y en los gráficos en 3D. La visualización se despliega en primera persona, desde la perspectiva de un astronauta [28].

Google Sky

Permite visualizar mapas de diversos objetos celetes (estrellas, constelaciones, galaxias, planetas). Incluye, entre otros, imagenes recogidas por el *Hubble*. Tiene como funcionalidad particular el poder centrar la visualización como si se estuviera haciendo desde el lugar en el que el usuario se encuentra físicamente[29].

Sky-Map.org

Es un mapa del cielo interactivo que se accesa por medio de un cliente web[30].

The SkyX by Software Bisque

Es otro software que emula un planetario. Tiene una base de datos extensiva y la habilidad de controlar remotamente un telescopio computarizado. Es un sistema multiplataforma y cuenta con una edición "de bolsillo"diseñada para correr en tablets[31].

Stellarium

Es un proyecto gratis y open source cuyo objetivo es actuar como un planetario de computador. Muestra imágenes astronómicas realistas en 3D. Cuenta con un catálogo de 600.000 estrellas[32].

Starry Night

Es un *software* pagado que simula un universo virtual de un billón de años luz de diámetro. Contiene un registro histórico de anomalías y un registro de predicciones[33].

Aladin

Es un *software* que provee un atlas interactivo del cielo. Permite al usuario ver imágenes digitales de cualquier parte del cielo y superponerlas[34].

SkyView

Es un observatorio virtual que genera imágenes de cualquier parte del cielo, en cualquier longitud de onda[35].

2.2.3. Herramientas de enseñanza

El enfoque de los *software* a continuación es entregar una herramienta de docencia útil para enseñar astronomía.

Nightshade Simultation Software

Nightshade es un *software* de visualización y simulación efocado en la enseñanza y la exploración astronómica. Además presenta material acerca de las ciencias de la tierra y otros tópicos relacionados. El proyecto es open source, por lo que cualquiera puede involucrarse reportando errores, pidiendo funcionalidades o de otra forma. *Nightsade* pone especial énfasis en el realismo y eficacia de sus simulaciones[36].

World Wide Telescope

Permite explorar imágenes astronómicas a través de la simulación de la navegación en 3D en el espacio. Permite tomar *tours* guiados, armados por atrónomos y profesores, acerca de algún tema en particular. Permite también crear *tours* personalizados[37].

2.2.4. Bases de datos con información variada

Las bases de datos a continuación contienen distintos tipos de información: registros de distintos tipos e imágenes de distintos objetos, en una, dos o más dimensiones.

SIMBAD (Set of Identifications, Measurements and Bibliography for Astronomical Data)

SIMBAD[3] es una base de datos astronómica que cuenta con un cliente web y que provee información básica, identificaciones cruzadas, bibliogragía y medidas observacionales de objetos extrasolares. Actualmente, SIMBAD contiene registros de cerca de 2.000.000 estrellas y 1.500.000 de otros objetos, como galaxias, nubes planetarias, clusters, novas, super novas, etc. Los únicos objetos astronómicos específicamente excluidos de SIMBAD son los objetos pertenecientes al Sistema Solar. SIMBAD también provee consultas a través de URLs, que se pueden anidar directamente en aplicaciones.

VizieR

VizieR[38] es un cliente *web* que provee acceso a una extensa bibliteca de catálogos astronómicos y tablas de datos. Cuenta con herramientas de consulta que permiten al usuario seleccionar tablas de datos relevantes y extraer y formatear registros que cumplan los criterios especificados. Actualmente, ViziR cuenta con 13348 catálogos.

VizieR puede consultarse utilizando una interfaz *web* o a través de una plataforma con Unix utilizando el paquete *cdsclient*. Este paquete es un conjunto de rutinas en el leguaje de programación C y que contiene el programa *viziquery*. El programa lee parámetros de consulta desde la entrada estandar y escribe los resultados en la salida estandar en varios formatos, como ASCII o FITS.

2.2.5. Bases de datos enfocadas a galaxias

HyperLeda

Es un sistema de información para astronomía que consiste en una base de datos y herramientas para procesar estos datos. Tiene como principal foco el estudio de la física y evolución de galaxias. Actualmente, la base de datos contiene más de 3 millones de objetos, de los cuales se sabe con relativa certeza que un millón y medio son galaxias. HyperLeda provee varios servicios: LEDA, un catálogo de parámetros homogéneos de galaxias; Hyercat, una colección de catálogos de los cuales se computa LEDA y un servicio *web* para accesar esta información, que admite consultas de tipo SQL[39].

2.2.6. Bases de datos enfocadas en fotometría e imágenes

Las siguientes bases de datos almacenan principalmente imágenes, de distintas dimensiones y enfocadas en distintas partes del espectro electromagnético.

HLA (The Hubble Legacy Archive)

Provee información respecto de las observaciones realizadas por el telescopio espacial Hubble. Contiene imágenes, imágenes compuestas más extensas, listados de fuentes y registros de espectros. La información se puede accesar mediante un cliente *web*[40].

NED (The NASA/IPAC Extragalactic Database)

Es una base de datos dedicada a objetos astronómicos extra-galácticos. Continene información respecto de la fotometría de ojetos en varios anchos de banda (Radio, visual, casi infrarrojo y rayos X). Además guarda distancias independientes del corriemiento al rojo, imágenes y espectros de distintos objetos. En total, contiene información de 514.678.591 fuentes no procesadas y de 214.813.768 objetos distinguibles[41].

NED puede consultarse a través de una interfaz *web* o a través de un *script batch*, sin embargo la habilidad de NED de soportar procesos automatizados de consulta es limitado, ya que muchos usuarios utilizan esta base de datos y las consultas respecto de grandes volúmenes de datos pueden colapsarla.

ESO data archive

Las medidas observacionales de la ESO (European Southern Observatory) quedan a disposición pública luego de un periodo propietario. Esta base de datos contiene todas las observaciones en bruto realizadas por los observatorios La Silla y Paranal, entre otros. La información en esta base de datos puede accesarse a través de un cliente *web*[42].

MAST (Mikulski Archive for Space Telescopes)

Es un proyecto fundado por la NASA dedicado a proveer a la comunidad astronómica una variedad de registros, con especial foco en información científica relacionada con el espectro óptico, ultrvioleta y casi infrarrojo. Toda la información almacenada en MAST puede accesarse a través su cliente *web*[43].

2.2.7. Edición de imágenes

Los *software* a continuación están dedicados a la edición y manipulación de imágenes astronómicas.

JSkyCat

Es un *software* desarrollado por la ESO que utiliza las clases contenidas en *JSky*, una librería *Java* también desarrollada por la ESO con enfoque en el uso astronómico. El software permite cargar imágenes astronómicas en formato FITS y operar con ellas. Tiene soporte para coordenadas WCS (World coordinates) y permite el acceso a catálogos astronómicos locales y web[44].

Montage

Permite al usuario crear mosaicos a partir de imágenes astronómicas [45].

2.2.8. Discusión del estado del arte

A continuación se entrega una tabla comparativa de los *software* anteriormente descritos, en la que se contemplan los siguientes criterios: si el *software* es pagado o no, si permite descargar imágenes en formato FITS, si cuenta con acceso a catálogos astronómicos, si cuenta con visualización de imágenes en alta resolución y finalmente, en qué plataformas corre (o si se accesa a través de un *web browser*).

	Pagado	Descarga de FITS	Acceso a catálogos	Permite visualización	Plataforma
SDSS DR12	no	sí	sí	no	acceso a través de web browser
2MASS	no	sí	sí	no	acceso a través de web browser
Celestia	no	sí	sí	sí	Windows, Linux, Mac OS X
Orbiter	no	no	no	sí	Windows
Google Sky	no	no	no	sí	acceso a través de web browser
Sky Map.org	no	no	sí	sí	acceso a través de web browser
Sky X	sí	no	sí	sí	Mac OS X, Windows
Stellarium	no	no	sí	sí	Windows, Linux, Mac OS X
Starry Night	sí	sí	sí	sí	Mac OS X, Windows
Aladin	no	sí	sí	sí	acceso a través de web browser
Nightshade	no	no	sí	sí	Windows, Linux, Mac OS X
WWTelescope	no	no	sí	sí	Windows 7 o Windows 8
SIMBAD	no	no	sí	no	acceso a través de web browser
VizieR	no	sí	sí	no	acceso a través de web browser
HyperLeda	no	sí	sí	no	acceso a través de web browser
HLA	no	sí	no	no	acceso a través de web browser
NED	no	sí	no	no	acceso a través de web browser
ESO Archive	no	sí	sí	no	acceso a través de web browser
MAST	no	sí	no	no	acceso a través de web browser
JSkyCat	no	no	sí	sí	Windows, Linux, Mac OS X
Montage	no	no	no	sí	Linux, Solaris, Mac OSX

Tabla 2.1: Tabla comparativa de los *software* considerados en el Estado del Arte

Luego de esta revisión es posible darse cuenta de que si bien las herramientas de software usadas en el área de astronomía son muchas, la mayoría sólo contempla una parte del análisis de datos: búsqueda o visualización. Las herramientas de visualización que sí contemplan búsqueda de datos generalmente están dedicadas a formar imágenes en tres dimensiones, a modo de planetario, no a una imagen particular cargada por el usuario. Además, ninguna de estas tecnologías está hecha para correr en un *wall display*, por lo que no cuentan con esta ventaja de *hardware* para el análisis colaborativo.

2.3. Herramientas utilizadas

2.3.1. SIMBAD

SIMBAD[3] es una base de datos astronómica que provee información básica, identificaciones cruzadas, bibliografía y mediciones de objetos astronómicos extra-solares. Se puede consultar SIMBAD a través de una interfaz web o utilizando URLs por medio de una aplicación. SIMBAD contiene 8.028.814 objetos, 22.588.353 identificadores y 12.523.089 citas de

objetos en papers. La base de datos permite consultas por el identificador, las coordenadas o el código bibliográfico del objeto y se pueden aplicar además distintos tipos de criterios.

Se elige SIMBAD por sobre otras bases de datos por varias razones. SIMBAD cuenta con un amplio registro de objetos astronómicos y con información muy variada en cuanto a ámbitos de investgación. Otras bases de datos, como NED, ESO Data Archive, HyperLeda y HLA, se descartaron por centrarse en una sección específica de datos astronómicos. NED muestra sólo objetos extragalácticos, ESO Data Archives presenta sólo la información recopilada por la ESO, HyperLeda es enfocada a galaxias y HLA contiene sólo información recopilada por el Hubble. Otras bases de datos, como *Aladin* y SDSS DR12, almacenan mayoritariamente imágenes, pero como la imágen es cargada por el usuario y la información consultada se codifica en texto, se descartaron por no ser idóneas. Además, SIMBAD cuenta con el valor de ser la base de datos utilizada por el cliente.

Los objetos astronómicos en SIMBAD se clasifican en una jerarquía de cuatro niveles[9], de esta manera es más fácil para el usuario filtrar el tipo exacto de objeto que busca. Por ejemplo, un objeto puede clasificarse como una estrella en el primer de nivel de jerarquía, una estrella variable en el segundo nivel, una estrella variable de tipo irregular en el tercero y una estrella variable de tipo Orión en el cuarto. Los tipos principales (primer nivel de jerarquía) son: Estrella, Galaxia, Materia Interestellar, Objeto múltiple, Objeto candidato, Fuente gravitacional, No-existente y *Wavelength* (Radio, IR, Red, Blue, UV, X o gamma). Cada objeto astronómico guardado en SIMBAD posee: información básica, información cruzada, descripción de medidas observacionales y referencias bibliográficas[10].

Información básica

- Identificadores.
- Tipo del objeto.
- Coordenadas: La sintaxis de las coordenas es `ra dec (wtype) [error ellipse] quality bibcode`, en donde:
  - ra: Ángulo de ascención. Las unidades se definen al momento de hacer la consulta.
  - dec: Ángulo de declinación.
  - wtype: Clase de longitud de onda (Rad, mm, IR, Optical, UV, Xray or Gam).
  - error ellipse: eje mayor y menor de la elipse de error.
  - quality: flag de calidad representada con las letras de la A a la E, con el siguiente significado:
    - A: Los datos de mejor calidad, viene desde el VLBI o desde Hipparcos.
    - B: Alta calidad.
    - C: Buena calidad.
    - D: Valores encontrados en la literatura, sin demasiado control. Tipicamente es información sacada de papers publicados.
    - E: Valores especulados.
  - bibcode: código bibliográfico indicando el origen de la información.

- Movimientos propios: La sintaxis es `pm-ra pm-dec [error ellipse] quality bibcode`, en donde:
  - pm-ra: Movimiento propio de ascención expresado en el sistema ICRS.
  - pm-dec: Movimiento propio de declinación.
- Flujos (Magnitudes): La sintaxis es "filter-name flux-value [error] quality flags bibcode", en donde:
  - filter-name: U, B, V, R, I, J, H, K, u, g, r, i, z.
  - flux-value: el valor de la magnitud.
  - error: valor de error asociado.
  - flags: Contiene dos flags, *mult* y *var*. La primera puede ser cero o el char 'J' y especifica si se utiliza fotometría conjunta. El segundo flag especifica si la magnitud es variable y de ser así se entrega además un estimado de la amplitud de esta variación.
- Velocidades radiales: La sintaxis es `value [error] (wavelength) quality bibcode`, en donde *value* es el valor numérico de la medida.
- Tipo Espectral: La sintaxis es `spectral type [error] (wavelength) quality bibcode`, en donde el atributo "spectral type"se compone de una clase de temperatura, una clase de luminosidad y peculiaridades espectrales (de haberlas).
- Tipo Morfológico: Este atributo es sólo aplicable a galaxias. La sintaxis es `mtype quality bibcode`, en donde *mtype* es una clase morfológica Hubble (espiral, elíptica, etc).
- Dimensión de la galaxia: Sólo aplicable a galaxias. La sintaxis es `maj-axis min-axis angle (incl) (wtype) quality bibcode`, en donde:
  - maj-axis: tamaño del eje mayor, en arcminutes.
  - min-axis: tamaño del eje menor, en arcminutes.
  - incl: inclinación, en unidades de 0 a 7.
- Notas: Anotaciones importantes acerca del objeto especificado.

Información cruzada

En caso de que el objeto astronómico en cuestión sea parte de un sistema múltiple o esté listado en más de un catálogo, SIMBAD permite obtener identificadores que codifican estas relaciones.

Medidas observacionales

Dado que existen varios catálogos y fuentes de información desde donde SIMBAD obtiene información, para cada medida básica se despliega la fuente. Además es posible consultar acerca de medidas observacionales que no estén dentro de la información básica. Los tipos de medidas disponibles son:

CEL: Fotometría ultravioleta desde Celescope.

Cl.G: Clusters de Galaxias. (Abell & Corwin).

diameter: Diametro estelar.

distance: Distancia al objeto.

Einstein: Lista de fuentes de Soft X-ray del Observatorio de Einstein.

GCRV: Velocidades radiales estelares.

GEN: Fotometría U, B, V, B1, B2 V1 en el sistema fotométrico de Ginebra.

GJ: Magnitudes absolutas y velocidades espaciales de estrellas cercanas.

Hbet: Índice $H\beta$.

HGAM: Ancho equivalente $H\gamma$.

IRAS: Catálogo de fuentes de puntos IRAS.

IRC: Fotometría KI del Two Micron Sky Survey.

ISO: Observatorio Infrarrojo espacial (Log observacional).

IUE: Explorador Ultravioleta Internacional (Log observacional).

JP11 UBVRIJKLMNH 11-colour Johnson's photometry.

MK: Clasificación espectral en el sistema Morgan-Keenan.

PLX: Paralajes trigonométricos.

pm: Movimientos propios (en el marco ecuatorial de 1950).

pos: Posiciones ecuatoriales.

ROT: Velocidades rotacionales.

RVEL: Velocidades radiales de objetos extragalácticos (quasars y galaxias).

SAO: Posiciones y movimientos propios del catálogo SAO.

TD1: Magnitudes ultravioleta del experimento TD1.

UBV: Fotometría UBV en el sistema fotométrico de Johnson.

uvby: Fotometría en el sistema fotométrico de Strömgren.

V*: Información relacionada a estrellas variables.

z: Redshifts (de galaxias lejanas y quasars).

Referencias Bibliográficas

Al hacer una consulta, SIMBAD permite desplegar todos los códigos bibliográficos que se encuentren relacionados con el objeto. Se proporcionan también enlaces a los artículos correspondientes.

Consultas a SIMBAD por URL

SIMBAD cuenta con un cliente *web* con una interfaz de usuario para realizar consultas, pero también permite anidar consultas dentro de una aplicación. Este tipo de consultas se realizan a través de una URL[8] y pueden definirse por coordenadas, identificadores o criterios

Una URL de consulta debe comenzar con el siguiente prefijo:

```
http://simbad.u-strasbg.fr/simbad/sim-script?script=
```

El parámetro *script* consiste en conjunto de comandos separados por un salto de línea. Los comandos más importantes a considerar son:

- Especificación de *output*: Para remover la consola y el comienzo del *script* del resultado entregado se utiliza el comando:

  ```
  output console=off script=off
  ```

- Especificación del formato del resultado: Para definir el formato de un resultado (objeto astronómico) se utiliza el comando:

  ```
  format object "formato de los resultados."
  ```

  Por ejemplo:

  ```
  format object "% IDLIST(1) | %COO(A D) | %FLUX(B;F) | %SP(S)"
  ```

  Este comando especifica que el objeto obtenido debe contener un identificador, el ángulo de ascención y declinación de coordenadas, el flujo en la banda B de objeto y el tipo espectral. Todos estos parámetros deben ir separados por el caracter |.
- Consultas por identificador: Para especificar una consulta por identificador se utiliza el comando:

  ```
  query id identifier
  ```

  Por ejemplo:

  ```
  query id hd 100
  ```

- Consulta por coordenadas: Para especificar una consulta por coordenadas se utiliza el comando:

  ```
  query coo ra dec radius=... frame=... equi=... epoch=...
  ```

  Por ejemplo:

  ```
  query coo 13 29 52.37 +47 11 40.8 radius=30s
  ```

- Consulta por criterios: Para realizar consultas con distintos filtros, se utiliza el comando:

  ```
  query sample "condición"
  ```

  La forma de expresar una condición está especificada en la documentación de SIMBAD. Una consulta por criterio podría ser por ejemplo:

```
query sample dec > 85 & ubv.b < 6
```

En donde se especifica que el ángulo de declinación de los objetos debe ser mayor que 85 y que su flujo en la banda B en el catálogo UBV debe ser menor que 6.

2.3.2. JSky

Es una colección de componentes en *Java* con distintas utilidades en el área de astronomía[44]. El proyecto es desarrollado por la ESO (European Southern Observatory) desde marzo del año 2000. La colección es libre para el uso público bajo la licencia pública GNU.

El paquete JSky contiene clases en Java para visualizar imágenes astronómicas, para búsqueda en catálogos astronómicos y para graficar símbolos correspondientes a catálogos en imágenes. Una de sus funcionalidades más útiles es que contiene clases que permite trabajar con WCS (World Coordinate System) y hacer conversiones entre distintos tipos de coordenadas y *frames*.

2.3.3. JBricks

Es una librería desarrollada en *Java* que se utiliza en conjunto con ZVTM para renderizar las interfaces de usuario de una aplicación[14]. Permite separar el proceso de renderizado en varias partes e indicar a cada computador de un *wall display* qué partes deben ser renderizadas en qué monitor. La funcionalidad primaria de *JBricks* es que oculta la complejidad de este proceso al desarrollador, ya que al sólo agregar unas pocas líneas de código, la misma aplicación escrita en ZVTM para un computador de escritorio puede renderizarse también en un *wall display*. Cada computador del *cluster* de un *wall display* contiene una réplica de la aplicación o un cliente que realiza el *render* de una parte de una escena. Además se le asigna una cámara virtual ZVTM para mostrar el área que corresponde. Una aplicación llamada *master* sincroniza los cambios en las escenas y en las cámaras. El manejo de señales de entrada mediante *JBrick* es genérico, por lo que se permiten varios dispositivos y la comunicación de estos con distintas aplicaciones. Las maneras de interactuar con un *wall display* pueden modificarse sin cambios en el código ZVTM de la aplicación; mediante especificaciones en este protocolo cada instrumento sabe con qué objetos puede interactuar. Para la creación y edición de distintos tipos de entrada *JBricks* soporta una variedad de dispositivos: *mouse*, teclados, *tablets*, controles de *nintendo wii*, dispositivos de rastreo de movimientos, etc.

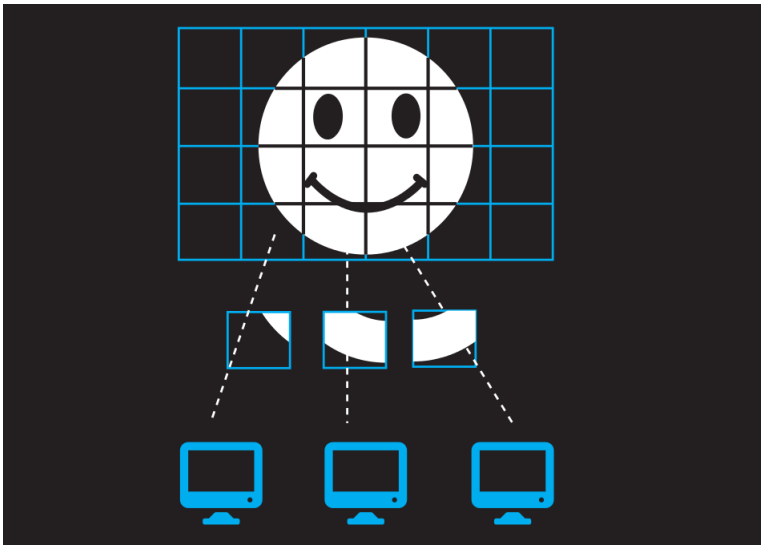


Figura 2.1: Representación gráfica del funcionamiento de *JBricks*, en donde se ilustra como la librería especifica qué parte de la imagen debe renderizar cada computador del *wall-display*.

2.3.4. Smarties

Smarties es una librería que permite la interacción con aplicaciones creadas para un *wall display* a través de múltiples dispositivos móviles *Android* simultáneamente[18]. *Smarties* se compone de una aplicación de entrada para dispositivos móviles y un protocolo de comunicación entre la aplicación en el muro, que actúa como servidor, y los dispositivos móviles, que actúan como clientes. Además, *Smarties* cuenta con una librería que esconde esta complejidad del desarrollador: agregando unas pocas líneas de código a la aplicación del muro se puede definir un nuevo dispositivo de entrada, que desencadena los mismos eventos de interacción ya existentes. Los usuarios interactúan con el muro mediante a cursores manejados de forma táctil en los dispositivos. También se pueden agregar en los dispositivos *widgets* que representen alguna funcionalidad, como por ejemplo, teclados para poder ingresar texto. Dado que *Smarties* permite que varios usuarios interactúen a la vez con el *wall display*, facilita un ambiente colaborativo.

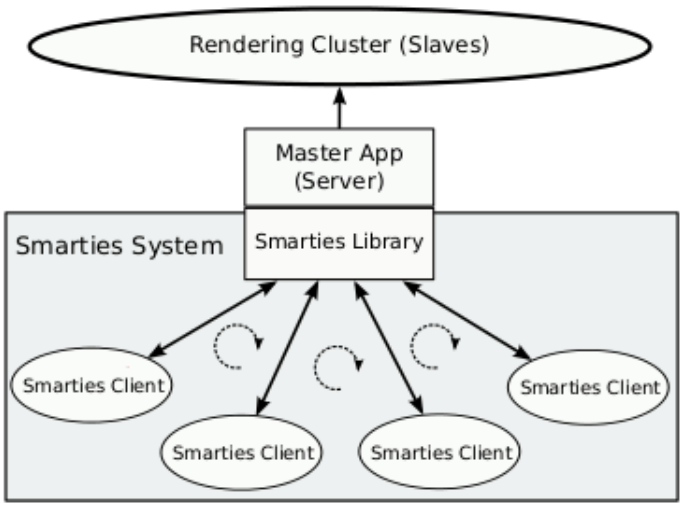


Figura 2.2: Representación gráfica del funcionamiento de *Smarties*. Cada dispositivo *tablet/smartphone* actúa como un cliente, que a través de la librería se conecta al computador *master* del *wall-display*, que actúa como servidor. Los eventos gatillados en el *master* son luego desplegados en los *slaves*.

2.3.5. ZVTM

Es una librería escrita en *Java* para la implementación de interfaces de usuario multi-escala [12](es decir, en donde se puede hacer *zoom* hacia dentro o hacia fuera) para trabajar en espacios complejos y plataformas de *hardware* no convencionales, como un *wall display*.

Otras herramientas de visualización como *Java Swing* o *QT* son muy útiles, pero se limitan a las interfaces de usuario convencionales, basadas en el modelo *WIMP*(*Window Icons Menus and Pointers*). Por otro lado, las herramientas de visualización más flexibles, como *OpenGL* y *Java2D* tienen el costo de requerir mucho más tiempo de desarrollo y esfuerzo de mantención. El objetivo principal de ZVTM es entregar a los desarrolladores una herramienta para la construcción de interfaces gráficas que resuelva estos problemas, dado que se encuentra a medio camino entre un nivel alto y un nivel bajo de abstracción.

El desarrollo utilizando ZVTM se basa en la metáfora de infinitos universos, llamados espacios virtuales (*virtual spaces*) que se observan a través de cámaras (*cameras*) móviles y que pueden contener varios objetos gráficos llamados *glyps*. Cada *glyph* se basa en el mismo modelo polimórfico de objeto gráfico. Los *glyphs* pertenecen sólo a un *virtual space*, pero pueden ser observados simultáneamente a través de varias cámaras virtuales. Las cámaras también se asocian con vistas (*views*), que actúan como ventanas de visibilidad en la interfaz de usuario. Cada cámara observa una región definida por el *view* del *virtual space* al que pertenece y tiene una altitud que corresponde al nivel de *zoom*. Para renderizar la interfaz completa de una aplicación, los espacios virtuales se disponen a modo de capas y se muestra una superposición de lo que cada cámara observa.

Los espacios virtuales son superficies virtuales infinitas en dos dimensiones. Para especificar la posición en donde agregar un *glyph* al espacio virtual se utilizan coordenadas cartesianas positivas. Al contrario de la mayoría de los sistemas de coordenadas en computadores, el eje Y de un *virtual space* crece hacia arriba. Los espacios virtuales no interactúan entre ellos, en el sentido de que un evento provocado por una interacción de usuario destinado a un espacio virtual particular no afecta a los otros.

Para manejar los *virtual spaces* se utiliza el objeto *Virtual Space Manager*. Existe sólo una instancia de este objeto por aplicación y es a través de este que los espacios virtuales son creados y administrados.

Los espacios virtuales no se representan directamente en la pantalla, sino que son observados por cámaras. La línea de visión de la cámara se encuentra siempre perpendicular al espacio virtual que observa, pero puede variar su posición respecto del sistema de coordenadas del *virtual space* y su altitud. Un espacio virtual puede tener varias cámaras, pero una cámara se asocia sólo con un espacio virtual.

Para crear la respresentación visual de la aplicación en la pantalla se necesitan también objetos de tipo *view*. Las cámaras no muestran nada por si solas, sólo definen la región del espacio virtual que se observa a través de una *view*. La *view* es la ventana de visión por donde el usuario observa la interfaz gráfica de la aplicación. Una vista tiene un ancho y un alto como atributos, que en conjunto con la posición y altitud de la cámara, definen lo que el usuario observa. Una vista puede estar asociada con varias cámaras, cada una de estas se asocia con una capa dentro de la vista, llamada *layer*. Cada *layer* es independiente de los otros. Los *layers* son transparentes y pueden configurarse para apagarse (ser invisibles) o prenderse (ser visibles). Las cámaras pertenecientes a distintos *virtual spaces* pueden combinarse en la misma vista, en donde los espacios virtuales se disponen en *layers*.

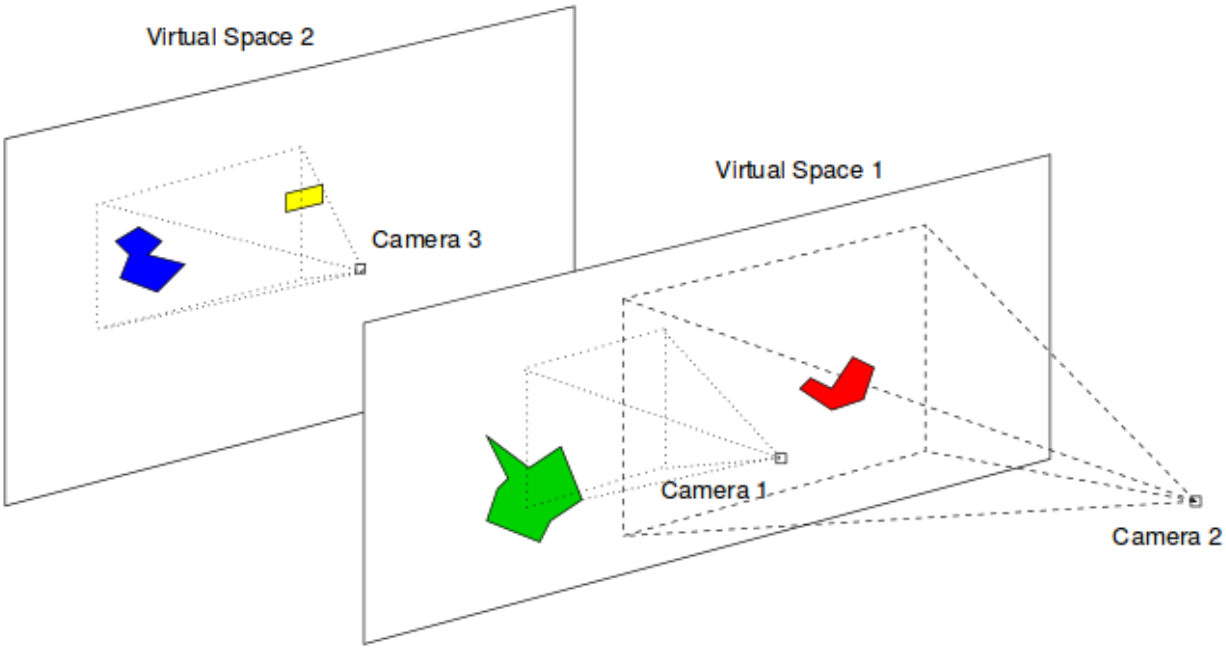


Figura 2.3: Representación gráfica de los espacios virtuales, cámaras, *glyphs* y *layers*

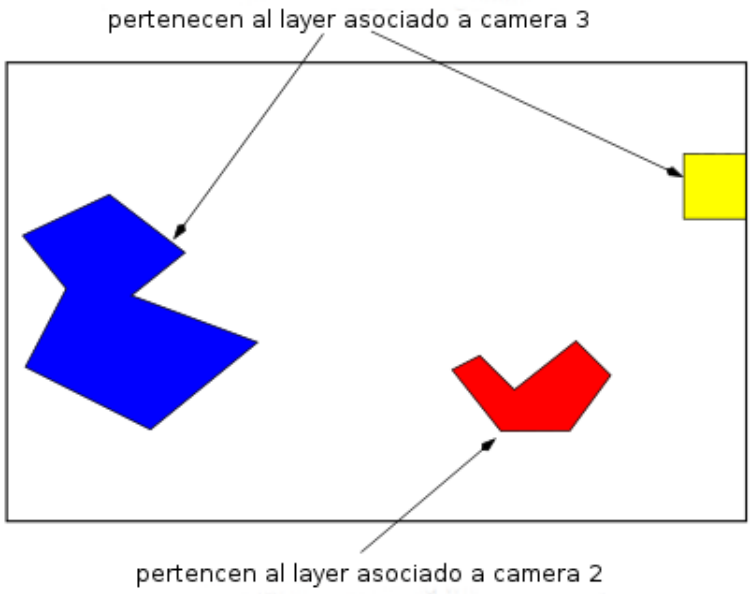


Figura 2.4: *View* compuesta de los elementos de la imagen 2.3 que observa el usuario

Capítulo 3

Especificación del problema

En este capítulo se detalla el problema a resolver. Para esto, se discute su relevancia, se definen los requisitos que debe cumplir la solución propuesta y se especifican los posibles casos de uso. Además, dado que el proyecto consiste en construir sobre la aplicación **FITS-OW** ya existente, se entregan antecedentes acerca de su estructura.

3.1. Relevancia del problema

Un *wall display* provee a la vez detalle y contexto sobre una imagen visualizada, además de fomentar el trabajo científico colaborativo, dada la naturaleza de la plataforma utilizada. El problema es relevante a resolver debido a que hasta el momento no existe una aplicación que combine estas ventajas con una herramienta de consulta a una base de datos de catálogos astronómicos, por lo que se espera que la solución ayude a enriquecer el análisis de este tipo de datos[15].

3.2. Usuario objetivo

El uso de la aplicación está enfocado principalmente en un conjunto de personas cuya profesión requiera del estudio de datos astronómicos, como astrónomos, astrofísicos u otros científicos. Dado que la aplicación está pensada para correr principalmente en un *wall display* y no se espera que cualquier usuario cuente con estas instalaciones, la aplicación también podrá ejecutarse en un computador de escritorio, por un usuario individual. Se tomó en principal consideración como usuario a Andrés Jordan, astrofísico de la Universidad Católica. El resto del equipo del Instituto de Astrofísica de la Universidad Católica se compone de Susana Eyheramendy Duerr, Maren Hempel y Roberto Muñoz.

3.3. Antecedentes

3.3.1. FITS-OW

La aplicación **FITS-OW** está escrita en el lenguaje de programación *Java* y usa la herramienta *Maven* para manejar la compilación del sistema. Utiliza la librería ZVTM para construir los elementos gráficos y *JBricks* para controlar el proceso de renderización entre las distintas pantallas del *wall display*.

FITS-OW permite visualizar una imagen en formato FITS en un *wall display* y además aplicarle a esta distintas escalas matemáticas y filtros de color.

3.3.2. Estructura ZVTM de FITS-OW

La aplicación **FITS-OW** cuenta con varias componentes de la libería ZVTM con las que estructura su interfaz gráfica, que se detallan a continuación:

- *VirtualSpaceManager vsm*
- Espacios virtuales:
  - *VirtualSpace dSpace*: espacio virtual en donde se despliegan las imágenes armadas a partir de una imagen FITS.
  - *VirtualSpace mnSpace*: espacio virtual en donde se muestra el menú principal de la aplicación.
  - *VirtualSpace crSpace*: espacio virtual en donde se mueve el cursor.
- Cámaras: cada espacio virtual cuenta con un una cámara asociada.
  - *Camera dCamera*: cámara asociada al espacio virtual que despliega imágenes *FITS*.
  - *Camera mnCamera*: cámara asociada al espacio virtual que muestra el menú principal
  - *Camera crCamera*: cámara asociada al espacio virtual en donde se mueve el cursor.
- EventListeners:
- *MenuEventListener me*: maneja los eventos asociados con el menú principal de la aplicación.
- *MVEventListener eh*: maneja eventos asociados con la interacción con los datos de la aplicación.
- *Layers*:
  - DATA_LAYER: asociado al *VirtualSpace dSpace*. Es el primer *layer*, es decir, los demás *layers* se sitúan por sobre este. Los eventos en este *layer* son manejados por el controlador de eventos *eh*.
  - MENU_LAYER: asociado al *VirtualSpace mnSpace*. Es el segundo layer. Los eventos son manejados por el controlador *me*.

  - CURSOR_LAYER: asociado al *VirtualSpace crSpace*. Este layer va por sobre todos los otros, ya que se asocia al cursor. Los eventos gatillados aquí son manejados por el controlador *eh*.
- *View mView*: **FITS-OW** cuenta con sólo una vista principal.

3.4. Requisitos

Con tal de definir las funcionalidades necesarias a implementar se decidió entrevistar a varios potenciales usuarios. Dado que la aplicación está pensada para ser usada por astrónomos, se preguntó a personas inmersas en el área respecto de su opinión sobre la aplicación y las funcionalidades que esta debería presentar.

Dentro de los astrónomos entrevistados se encuentran Francisco Foster, del Centro de Modelamiento Matemático de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile; Diego Mardones, del Departamento de Astronomía de la Universidad de Chile y Faviola Molina, Doctora en astrofísica, actualmente trabajando en las oficinas de Inria Chile. Además, se tomó en especial consideración la opinión del cliente, el equipo del Instituto de Astrofísica de la Católica, y particularmente de Andrés Jordan, a quien se entrevistó la mayor cantidad de veces. Cabe destacar que la aplicación **FITS-OW** también fue presentada en Pucón 2015, un simposio celebrado en Puerto Varas enfocado en Astro y Bio ingeniería, en el tercer taller de Astro informática y en las oficinas de Inria Chile, en donde se expuso ante el CEO de Inria Francia, Antoine Petit, y ante un equipo del LSST.

A continuación se exponen los requisitos con los que debe contar *FITS-OW* al finalizar el proyecto. Estos requisitos pueden ser usados posteriormente como criterios de aceptación del trabajo de memoria.

3.4.1. Requisitos funcionales

El sistema debe permitir al usuario realizar las siguientes acciones:

- Consultar la base de datos SIMBAD por un objeto astronómico particular o por varios objetos astronómicos. Las consultas deben hacerse en paralelo con la visualización de la imagen FITS y permitir los siguientes parámetros:
  - Consulta por coordenadas, definiendo un área de la imagen numéricamente o a través de la interfaz gráfica.
  - Consulta por identificador del objeto.
  - Consulta por *script* ingresado por el usuario.
- Consultar la base de datos Simbad aplicando los siguientes filtros adicionales:
  - Filtrado por tipo de objeto astronómico.

  - Filtrado por rango y calidad de medición del movimiento propio del objeto.
  - Filtrado por rango y calidad de medición del paralaje del objeto.
  - Filtrado por rango y calidad de medición de la velocidad radial de objeto.
  - Filtrado por clase de temperatura, clase de luminosidad, peculiaridades y calidad de medición del tipo espectral del objeto.
  - Filtrado por rango del flujo del objeto.
  - Consultas acerca de mediciones en catálogos específicos.
- Interactuar con los resultados encontrados por la consulta a través de una interfaz gráfica que muestre la información básica y las mediciones obtenidas para cada objeto.

3.4.2. Requisitos de usabilidad

- La aplicación debe poder usarse a través de un computador de escritorio, utilizando teclado y *mouse* como dispositivos de entrada.
- La aplicación debe poder usarse a través de un *wall display*, utilizando un *tablet* como dispositivo de entrada.
- La aplicación debe poder usarse a través de un *wall display*, utilizando el teclado y *mouse* del computador *master* como dispositivo de entrada.

3.4.3. Requisitos de calidad

- La aplicación debe ser fácil e intuitiva de manejar.
- La aplicación debe desplegar los resultados a las consultas de forma coherente.

3.5. Casos de uso

A continuación se despliega el diagrama de casos de uso para la aplicación y se detalla cada caso por separado:

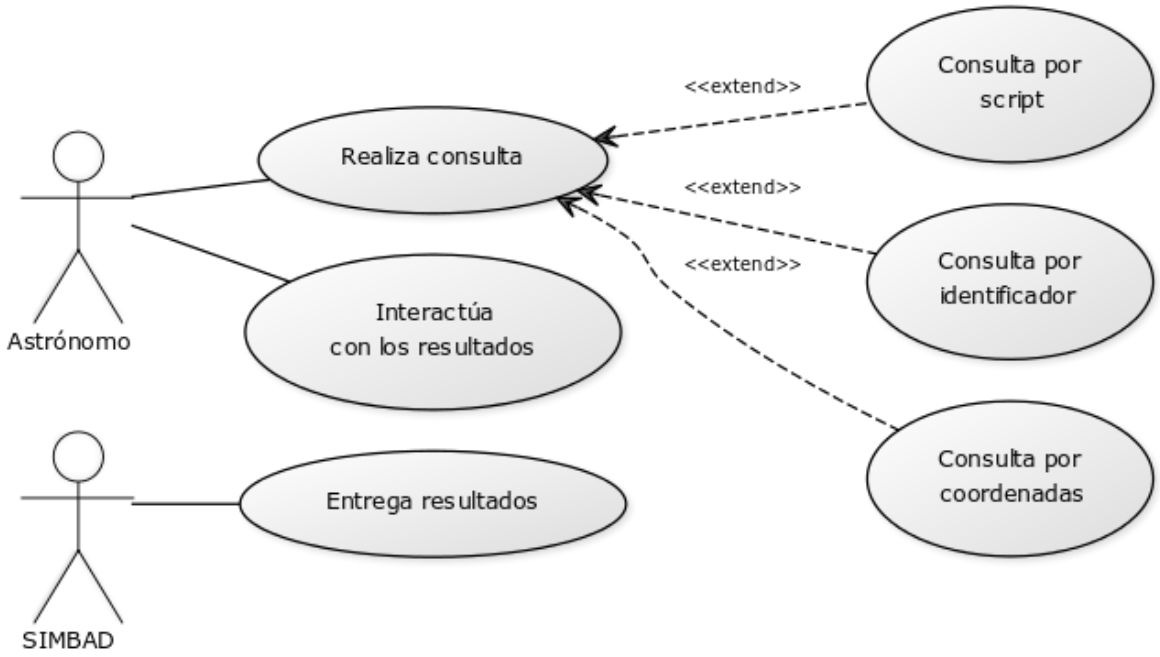


Figura 3.1: Diagrama de casos de uso de la aplicación FITS-OW

3.5.1. CU1: Realizar consulta

Caso de Uso:

Actores:	Astrónomo
Propósito:	Especificar la consulta a realizar a la base de datos.
Resumen:	El usuario debe ingresar el tipo de consulta que quiere realizar y los parámetros correspondientes. Opcionalmente, puede proveer filtros adicionales a la consulta.
Tipo:	Primario
Referencias cruzadas:	CU1.1, CU1.2, CU1,3, CU2

Curso Normal:

Acciones de los actores	Respuesta del sistema
1. El astrónomo selecciona el tipo de consulta que quiere realizar (por coordenadas, script o identificador)	
2. El astrónomo ingresa los parámetros correspondientes respecto del tipo de consulta que eligió.	
3. Opcionalmente, el usuario ingresa filtros adicionales a las consultas y selecciona la opción de ejecutar consulta	4. El sistema verifica que los parámetros ingresados sean válidos, ejecuta la consulta y despliega los resultados encontrados.

Cursos Alternos:

- 4a. Si los parámetros ingresados no son válidos, se le informa al usuario y se le pide que los ingrese nuevamente.
- Si no se encuentran resultados a la consulta en SIMBAD, se le notifica al usuario.

3.5.2. CU1.1: Realizar consulta por coordenadas

Caso de Uso:

Actores:	Astrónomo
Propósito:	Especificar las coordenadas para la consulta a realizar a la base de datos.
Resumen:	El usuario puede definir una región entregando un punto y un radio en valor numérico o marcando una región en la imágen.
Tipo:	Secundario
Referencias cruzadas:	CU1, CU2

Curso Normal:

Acciones de los actores	Respuesta del sistema
1. El astrónomo ingresa las coordenadas en forma numérica o marcando una región circular en la imágen. Opcionalmente se puede proveer un *frame*, de no ser así, se utiliza por defecto J2000.	2.El sistema verifica que los parámetros ingresados sean válidos, ejecuta la consulta y despliega los resultados.

Cursos Alternos:

- 2a. Si los parámetros ingresados no son válidos, se le informa al usuario y se le pide que los ingrese nuevamente.
- 2b. Si no se encuentran objetos en las coordenadas dadas, se le informa al usuario al respecto.

3.5.3. CU1.2: Realizar consulta por identificador

Caso de Uso:

Actores:	Astrónomo
Propósito:	Especificar el identificador para la consulta a realizar a la base de datos.
Resumen:	El usuario ingresar el identificador del objeto celeste que desea buscar.
Tipo:	Secundario
Referencias cruzadas:	CU1, CU2

Curso Normal:

Acciones de los actores	Respuesta del sistema
1. El astrónomo ingresa el identificador, es decir, el código que representa el objeto indexado en la base de datos	2.El sistema verifica que el parámetro ingresado es válido, ejecuta la consuta y despliega los resultados.

Cursos Alternos:

- 2a. Si el parámetro ingresados no es válido, se le informa al usuario y se le pide que lo ingrese nuevamente.
- 2b. Si no se encuentran objetos con el identificador dado, se le informa al usuario que el objeto buscado no existe en la base de datos.

3.5.4. CU1.3: Realizar consulta por script

Caso de Uso:

Actores:	Astrónomo
Propósito:	Especificar el script de la URL de la consulta a la base de datos.
Resumen:	El usuario ingresar el un script que representa la URL de consulta que se realiza a SIMBAD, especificando los parámetros pertinentes.
Tipo:	Secundario
Referencias cruzadas:	CU1, CU2

Curso Normal:

Acciones de los actores	Respuesta del sistema
1. El astrónomo ingresa el script de la URL de consulta. De usar esta opción, no se podrán marcar parámetros adicionales y el usuario tendrá que especificarlos en su script	2. El sistema verifica que el script sea válido, ejecuta la consulta y despliega los resultados

Cursos Alternos:

- 2a. Si el script ingresados no es válido, se le informa al usuario y se le pide que lo ingrese nuevamente.
- 2b. Si no se encuetran resultados se le informa al usuario de esto.

3.5.5. CU2: Desplegar resultados

Caso de Uso:

Actores:	SIMBAD
Propósito:	Entregar resultados
Resumen:	Luego de ejecutar la consulta SIMBAD entrega los resultados en texto plano ASCII.
Tipo:	Primario
Referencias cruzadas:	CU1, CU1.1, CU1.2, CU1.3

Curso Normal:

Acciones de los actores	Respuesta del sistema
1. SIMBAD ejecuta la consulta ingresada por el astrónomo y etrega los resultados encontrados en texto plano ASCII	2. El sistema utiliza un *parser* para traducir estos resultados a estructuras de datos y luego los despliega en paralelo con la imagen FITS.

Cursos Alternos:

- 2a. De no encontrarse resultados, SIMBAD entrega un error. El sistema atrapa este error y lo convierte en una notificación al usuario.

3.5.6. CU3: Interactuar con resultados

Caso de Uso:

Actores:	Astrónomo
Propósito:	Explorar resultados entregados
Resumen:	El usuario interactúa con los resultados visualizados por el sistema. Puede seleccionar distintos objetos y ver la información asociada a estos
Referencias cruzadas:	

Curso Normal:

Acciones de los actores	Respuesta del sistema
1. El astrónomo puede seleccionar alguno los resultados, que se entregan en una lista de objetos y además se marcan sobre la imagen FITS.	2.El sistema marca el objeto seleccionado tanto en la lista y como en la imagen.
3. El astrónomo puede cambiar entre desplegar la información básica o las medidas encotradas para un objeto particular	4.El sistema despliega el tipo de información que el usuario selecciona

Capítulo 4

Descripción de la solución

A continuación se describe la solución propuesta a este trabajo de memoria. Se discutirán los componentes de la libería ZVTM y módulos añadidos a **FITS-OW** para extenderla, las clases involucradas y la arquitectura de hardware considerada.

Dado que **FITS-OW** estaba escrita en el lenguaje de programación *Java*, se utiliza el mismo lenguaje para construir sobre ella. Se utiliza también la librería ZVTM para agregar las componentes gráficas y la librería *JSky* para el manejo de las coordenadas astronómicas.

4.1. Componentes ZVTM añadidas a FITS-OW

Para estructurar visualmente la aplicación **FITS-OW** se utilizan varias componentes de la librería ZVTM, detalladas en la sección 3.3.2. Para añadir las funcionalidades asociadas con las consultas a SIMBAD, se agregan las siguientes componentes:

- *VirtualSpace sqSpace*: el espacio virtual en donde se insertan todos los *glyphs* que construyen la interfaz de usuario para realizar consultas e interactuar con los resultados de estas.
- *Camera sqCamera*: Cámara asociada a *sqSpace*.
- *Layer* SIMBAD_LAYER: Se dispone por sobre el *layer* de la imagen *FITS* y bajo el *layer* de movimiento del cursor.

4.2. Arquitectura de software de la solución

Para dar solución al problema a resolver se decide agregar un módulo adicional a la aplicación **FITS-OW**, llamado *simbad*, que se encargue de realizar la conexión, ejecutar las consultas y desplegar los resultados. A continuación se presentarán los detalles de diseño e implementación de este módulo, indicando sus componentes y las de otros módulos de la

aplicación que sean relevantes.

4.2.1. Módulos y flujo de consulta

De los módulos que componen **FITS-OW** sólo dos son relevantes para lograr los objetivos de esta memoria. El primero se llama *fitsow* y contiene todas las clases que inicializan la interfaz de usuario de la aplicación, que manejan los eventos o tienen otras utilidades. El segundo es el módulo agregado para resolver el problema, *simbad*. Este módulo contiene todos los *glyphs* necesarios para desplegar las componentes de la interfaz de usuario utilizada para realizar consultas e interactuar con los resultados, las estructuras de datos requeridas para guardar la información extraída de SIMBAD y la lógica misma de las consultas. El módulo *simbad* interactúa con *fitsow* principalmente a través de los controladores de eventos, ya que es aquí en donde se maneja la lógica asociada a las interacciones con el usuario y en donde se gatillan las consultas y la creación de todos los elementos requeridos para desplegarlas.

La figura a continuación representa estos módulos del sistema, mostrando además las clases principales involucradas, agrupadas en sub-módulos. Además se simboliza a través de flechas como cada componente interactúa con las demás; estas flechas están numeradas según el órden de flujo que se da cuando un usuario realiza una consulta.

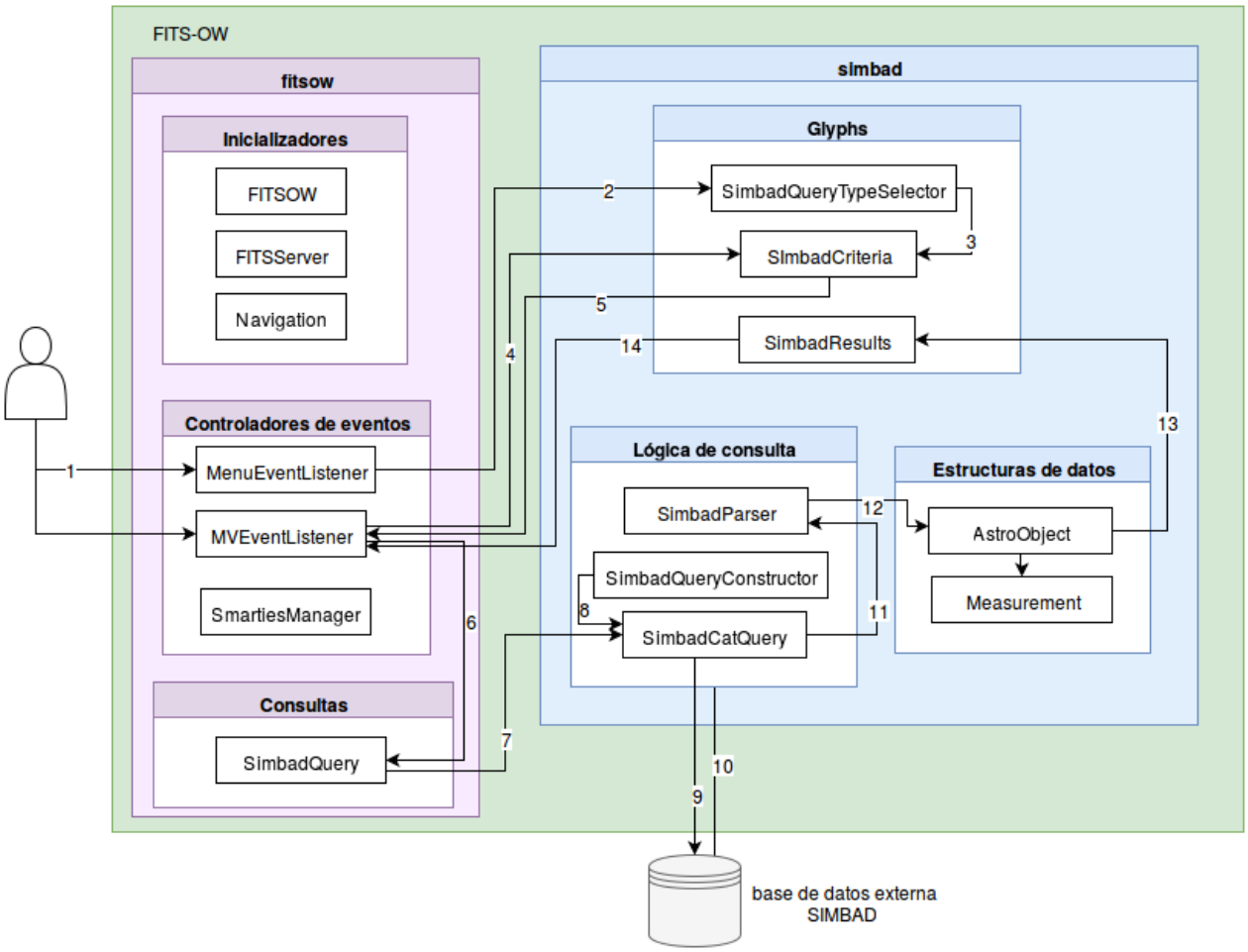


Figura 4.1: Módulos relevantes a esta memoria de la aplicación FITS-OW y sus interacciones.

El módulo *fitsow* (en morado en la figura 4.1) cuenta con tres sub-módulos: el primero contiene los inicializadores de la aplicación, el segundo se encarga de controlar eventos y el tercero fue agregado para encargarse de llamar a otros módulos de consulta.

El sub-módulo con inicializadores cuenta con tres clases principales: *FITSOW*, *FITSServer* y *Navigation*. *FITSOW* es la clase primordial de la aplicación ya que contiene todas las componentes de la librería ZVTM necesarias para incializar la interfaz de usuario, como espacios virtuales en donde agregar *glyphs* y cámaras asociadas a estos. Además esta clase contiene una instancia de la clase *Navigation*. *Navigation* se encarga de definir los comportamientos de navegación básicos de la aplicación, como la lógica que debe ejecutarse al hacer *zoom* hacia adentro o hacia afuera de la vista. La clase *FITSServer* es responsable de levantar un servidor en donde se disponen las imágenes FITS cargadas por el usuario. De esta manera, cada computador del *cluster* del *wall display* tiene acceso a la imagen y es capaz de renderizar la parte de esta que le corresponde.

El sub-módulo de controladores de eventos contiene tres clases principales. *MenuEventLis-*

tener se encarga de escuchar los eventos del menú principal de la aplicación, en donde se selecciona si se quiere aplicar un filtro de color o una escala a la imagen, o si se quiere realizar una consulta sobre esta. *MVEventListener* se encarga de manejar los eventos relacionados con la interacción con la imágen y con los elementos del módulo de consulta. Por otro lado, *SmartiesManager* define cómo cotrolar los eventos que provienen de un dispositivo externo, como *tablet* o *smartphone*, al correr la aplicación en un *wall display*. Esta clase utiliza la lógica ya definida en *MenuEventListener* y *MVEventListener* para manejar los eventos que le corresponden.

El sub-módulo de consultas sólo tiene una clase, *SimbadQuery*, que se encarga de mediar entre *MVEventListener* y el módulo *simbad*. Además es responsable de desplegar los resultados encontrados. Se espera que en el futuro se agreguen más componentes a este módulo, que se encarguen de realizar conexiones a otras bases de datos.

El módulo *simbad* (en azul en la imagen 4.1) cuenta con tres sub-módulos: *Glyphs*, Lógica de consulta y Estructuras de datos. El sub-módulo *Glyphs*, como su nombre lo indica, contiene todas las componentes gráficas que se deben desplegar para rederizar la interfaz de usuario de consulta y de interacción con los resultados de esta. Los tres *glyphs* principales son *SimbadQueryTypeSelector*, que representa un panel en donde elegir si la consulta debe realizarse por coordenadas, identificador o script, *SimbadCriteria*, en donde el usuario selecciona los criterios de su consulta y *SimbadResults*, en donde se muestran los resultados obtenidos luego de la consulta a la base de datos.

El sub-módulo de estructuras de datos contiene dos clases: *AstroObject* y *Measurement*. *AstroObject* se encarga de representar como objetos los elementos astronómicos obtenidos por medio de la consulta a SIMBAD, guardando como atributos el identificador, las coordenadas y otros atributos del objeto celeste. *Measurement*, por otro lado, guarda en matrices las medidas obtenidas en las consultas para cada objeto astronómico. Como es de esperarse, *AstroObject* contiene referencias a los objetos de la clase *Measurement* que corresponda.

El sub-módulo de lógica de consulta contiene tres clases: *SimbadQueryConstructor*, *SimbadCatQuery* y *SimbadParser*. *SimbadQueryConstructor* tiene como responsabilidad construir un *script* válido de consulta, en formato de texto, a partir de las opciones seleccionadas por el usuario. *SimbadCatQuery* se responsabiliza de ejecutar la consulta misma, obteniendo el *script* necesario de *SimbadQueryConstructor*. Además, *SimbadCatQuery* construye los objetos *AstroObject* y *Measurement* que corresponde, utilizando para esto la clase *SimbadParser*. *SimbadParser* posee los métodos necesarios para transformar los resultados entregados en texto plano ASCII por SIMBAD a las estructuras de datos definidas.

El flujo de consulta, marcada en la figura 4.1 por flechas numeradas funciona de la siguiente manera:

- 1. El usuario selecciona en el menú principal la opción de realizar una consulta, con lo que se gatilla un evento en la clase *MenuEventListener*.

- 2. El evento gatilla la creación de una instancia del *glyph SimbadQueryTypeSelector*, en donde el usuario puede seleccionar el tipo de consulta que desea realizar.
- 3. Según el tipo de consulta que el usuario elija, se instancia un *glyph* de tipo *SimbadCriteria*, en donde el usuario debe entregar el parámetro de consulta del tipo elegido y en donde se pueden además especificar otros filtros opcionales de consulta.
- 4. El usuario interactúa con el *glyph SimbadCriteria* a través del controlador de eventos *MVEventListener*, entregando todos los parámetros necesarios para ejecutar la consulta deseada.
- 5. *MVEventListener* recoge la información ingeresada por el usuario.
- 6. *MVEventListener* instancia un objeto *SimbadQuery*, que representa una consulta.
- 7. *SimbadQuery* se conecta con el sub-módulo de lógica de consultas del módulo *simbad*, específicamente con la clase *SimbadCatQuery*, que ejecutará la consulta.
- 8. *SimbadCatQuery* utiliza los métodos de *SimbadQueryConstructor*, que transforman la información recogida por el usuario en un *script* de consulta.
- 9. *SimbadCatQuery* se conecta con la base de datos externa **SIMBAD** y ejecuta el *script* de consulta previamente construido, especificando el formato de texto en el que la base de datos debe responder.
- 10. **SIMBAD** responde la consulta entregando los resultados en un texto plano ASCII, en el formato especificado por *SimbadCatQuery*.
- 11. *SimbadCatQuery* utiliza los métodos definidos en *SimbadParser*, entregando la información obtenida desde **SIMBAD**.
- 12. *SimbadParser* construye una lista de objetos *AstroObject* a partir de la información recogida desde **SIMBAD**. Además, con las mediciones adquiridas para cada objeto se construyen objetos *Measurement*, que representan una medida en particular. Estas instancias de *Measurement* se asocian con las instancias de *AstroObject* que representan el objeto al que pertenecen
- 13. A partir de la lista de objetos *AstroObject* se construye el *glyph SimbadResults*, que representa los resultados obtenidos a través de la consulta de forma gráfica.
- 14. Una vez que la consulta fue realizada y el *glyph* de resultados fue obtenido, este se agrega al espacio virtual definido para desplegar los elementos relacionados con las consultas a **SIMBAD**, *sqSpace*. La clase *MVEventListener* queda atenta a más eventos que el usuario pueda gatillar al interactuar con los resultados desplegados.

4.2.2. Diagrama de clases del módulo *simbad*

Con el fin de entender el funcionamiento de la solución se entrega a continuación un diagrama de todas las clases involucradas en el módulo *simbad*, señalando los distintos sub-módulos en distintos colores. Debido al tamaño de la página, el diagrama de clases fue comprimido, mostrando las clases sólo como cajas con su nombre y omitiendo sus atributos y métodos. El diagrama completo se puede encontrar en la sección de Apéndices.

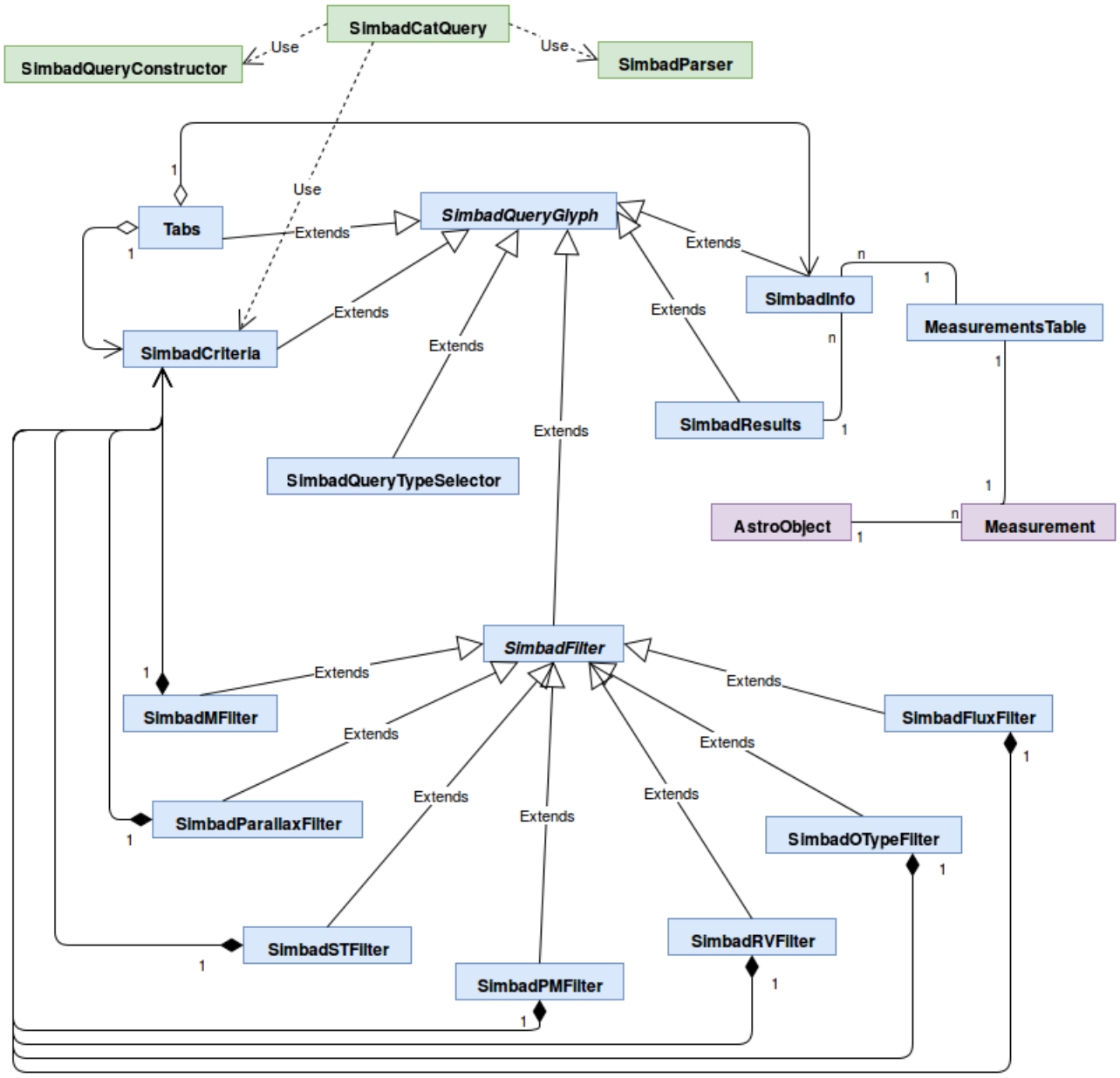


Figura 4.2: Diagrama de clases comprimido, mostrando las clases del módulo *simbad*

A continuación se detallan cada una de las clases y sus funcionalidades.

SimbadQueryGlyph

Todas las clases que representan figuras renderizables extienden la clase abstracta *SimbadQueryGlyph*, que a su vez extiende a la clase *Glyph*. Se decidió lo anterior con el fin de dar claridad conceptual a la construcción de *glyphs* y para poder reutilizar parámetros que poseen todos los objetos gráficos de la interfaz de usuario, como color de fondo, color de texto, tama-

ño de texto, tipo de letra, *VirtualSpace* y *Camera* asociada. Además, esta clase cuenta con la implementación de métodos que son comunes a todos los *glyphs* de la interfaz: `getWidth`, para obtener el ancho de la figura; `getHeight`, para obtener su alto; `getBackground`, para obtener el rectángulo que actúa como fondo de la figura y `coordInsideItem`, que calcula si el cursor de la aplicación está dentro del *glyph* en cuestión.

SimbadQueryTypeSelector

La clase *SimbadQueryTypeSelector* representa el objeto de la interfaz gráfica en donde se selecciona el tipo de consulta. Este objeto guarda como atributos tres botones en forma de rectángulos y un entero que codifica cuál de estos botones está seleccionado, si alguno lo está. Posee como métodos `getSelectedButton`, que calcula qué botón está siendo seleccionado por el cursor y `select` que selecciona el botón deseado.

SimbadCriteria

Una vez seleccionado el tipo de consulta, se instancia un objeto de la clase *SimbadCriteria*, que representa la interfaz para ingresar distintos criterios. *SimbadCriteria* se compone de varios filtros (de tipo de objeto, de tipo espectral, etc), que extienden a la clase abstracta *SimbadFilter*. La clase *SimbadCriteria* contiene también un objeto *Tabs*, que representa gráficamente pestañas para alternar entre los filtros de información básica y los filtros de mediciones. Esta es una funcionalidad que se agrega dinámicamente, usando *Tabs* como decorador de *SimbadCriteria*.

Tabs

Es una clase que se encarga de crear el *glyph* que permite alternar entre dos pestañas distintas: la pestaña de información básica y la pestaña de mediciones. Esta clase se añade como decorador a los *glyphs* que lo necesitan: *SimbadCriteria* y *SimbadInfo*. Guarda como atributos qué pestaña está seleccionada, además de un largo y ancho alternativo (en caso de que al cambiar de pestaña el *glyph* requiera cambiar de tamaño).

SimbadFilter

Esta es una clase abstracta común a todos los objetos gráficos en donde se seleccionan filtros de consulta. Se decidió implementar esta clase por claridad conceptual y para agregar métodos comunes a todos los filtros: `select`, que se encarga de seleccionar el item que esté bajo el cursor y `getItemSelected`, que se encarga de retornar qué item está seleccionado.

SimbadParallaxFilter

Esta clase extiende a la clase *SimbadFilter* y representa una componente gráfica de *SimbadCriteria*. Este filtro se encarga de renderizar los criterios de consulta relacionados con el paralaje astronómico.

SimbadOTFilter

También es un filtro que extiende a *SimbadFilter* y que es instanciado por *SimbadCriteria*. Representa el objeto gráfico en donde se puede seleccionar qué tipo de objeto astronómico se desea buscar en SIMBAD.

SimbadFluxFilter

Extiende a *SimbadFilter* y es instanciada en *SimbadCriteria*. Representa gráficamente las opciones de filtrado relacionadas con el flujo electromagnético de los objetos astronómicos buscados.

SimbadSTFilter

Tiene como clase padre a *SimbadFilter* y es una componente de *SimbadCriteria*. Este *glyph* se encarga de mostrar las opciones de búsqueda que tienen que ver con el tipo espectral de los objetos astronómicos.

SimbadPMFilter

Extiende a *SimbadFilter* y es un atributo de *SimbadCriteria*. Representa gráficamente los criterios de filtrado que tienen que ver con el movimiento propio de un objeto astronómico.

SimbadRVFilter

Es un filtro que exitiende a *SimbadFilter* y que compone a *SimbadCriteria*. Se encarga de entregar al usuario las opciones de consulta para filtrar según la velocidad radial del objeto astronómico.

SimbadMFilter

Extiende a la clase *SimbadFilter* y es instanciado por la clase *SimbadCriteria*. Sin embargo, este filtro se muestra al seleccionar en *SimbadCriteria* el *Tab* de mediciones, a diferencia de

los demás filtros, que se muestran al seleccionar en el *Tab* de información básica. Representa gráficamente las distintas opciones de catálogos de consulta con los que cuenta SIMBAD.

SimbadResults

Al desplegar los resultados de una búsqueda se instancia un objeto *SimbadResults*, que representa un listado de los registros obtenidos. Guarda como atributos todos los objetos gráficos que representan los textos de los identificadores de los resultados y líneas divisiorias entre ellos. Contiene la implementación de los siguientes métodos: `insideWhichObject`, que retorna qué objeto de la lista está siendo seleccionado por el cursor; `highlight`, que resalta el objeto del listado que se está seleccionando; `getCorrespondingGlyph`, que calcula qué cruz hay que cambiar de color sobre la imagen astronómica al seleccionar un objeto del listado; `highlightCorrespondingGlyph`, que cambia de color el *glyph* en forma de cruz que corresponda; y `getBasicInfo`, que retorna un *glyph* de tipo *SimbadInfo*, correspondiente a la información específica al objeto astronómico que se está seleccionando.

SimbadInfo

Al seleccionar algún ítem de *SimbadResults* se inicializa un objeto *SimbadInfo*, que contiene la información del objeto celeste en particular. *SimbadInfo* también es decorado por *Tabs*, para alternar entre la información básica de un objeto y sus mediciones. En la pestaña de mediciones se agregan *MeasurementsTable*, que son representaciones gráficas de tablas de mediciones.

MeasurementsTable

Es un *glyph* instanciado por *SimbadInfo* que representa de manera gráfica una tabla con la mediciones de un objeto astronómico en particular. Guarda como atributos el número de filas y columna de la tabla, además del fondo de esta.

SimbadQueryConstructor

La clase *SimbadQueryConstrutor* construye las consultas a SIMBAD haciendo uso de las opciones entregadas por el usuario en *SimbadCriteria*. Por cada filtro contenido en *SimbadCriteria* esta clase contiene un método que interpreta las opciones seleccionadas por el usuario y las transforma en un *String* válido de consulta.

SimbadQueryParser

SimbadQueryParser transforma los resultados en texto plano entregados por SIMBAD a estructuras de datos *AstroObjects* y *Measurements*.
Contiene los siguientes métodos: `splitURLintoStrings(URL url)`, que transforma el texto plano que retorna SIMBAD en varias *Strings* con información respecto de sólo un objeto; `stringsToAstroObjects`, que transforma todas las *Strings* en objetos *AstroObject*; `stringToAstroObject`, que transforma un *String* en particular a un *AstroObject*; `parseBasicData`, que se encarga de dar el formato adecuado a la información básica de un objeto astronómico; `parseMeasurements`, que entrega el formato correcto a las mediciones de un objeto astronómico; y finalmente `buildTable`, que construye las estructuras de datos de tipo *Measurement* que guardan mediciones. Todos los *parsers* de esta clase se implementaron utilizando las funciones nativas de la clase *String* de *Java*.

AstroObject

Es una estructura de datos que representa un objeto astronómico. Guarda como atributos un *String* identificador, las coordenadas astronómicas del objeto, un *HashMap* con la información básica del objeto, y un vector con todos los objetos *Measurement* que corresponda.

Measurement

Es una estructura de datos que corresponde a una medición de un objeto celeste en específico. Contiene el nombre de la medición y una matriz con los datos de esta.

SimbadCatQuery

Esta clase utiliza a *SimbadQueryParser* y a *SimbadQueryConstructor* para llevar a cabo la consulta. Se compone de métodos que construyen consultas por identificador, coordenadas o *script* y que retornan una lista con los *AstroObject* que representan los resultados obtenidos por SIMBAD.

4.3. Arquitectura de hardware

Dado que la aplicación está pensada para correr principalmente en un *wall display*, se necesita una plataforma de este estilo para realizar pruebas durante el transcurso del desarrollo y al término de este. Para esto se utilizó el *Andes wall display*, ubicado en las oficinas de la Fundación Inria Chile.

El *Andes wall display*[16] está compuesto por un arreglo de veinticuatro monitores manejados por un *cluster* de trece computadores. De estos computadores, doce controlan los monitores (a cada computador le corresponden dos monitores) y el número trece se encarga de la sincronización de los otros. Los detalles técnicos son los siguientes:

- 24 Monitores FullHD de bisel estrecho que forman una matriz de 6×4, con una resolución total de 11520×4320 pixeles.
- 24 servidores nVidia Quadro 2000 en 12 computadores Dell Precision R5500 de 64 bits, ejecutándose con Linux (Fedora) y cada uno equipdo con:
  - CPU: 2x Intel Xeon E5606
  - Memoria: 12GB DDR3 RAM
  - Capacidad de almacenamiento basada en SSD

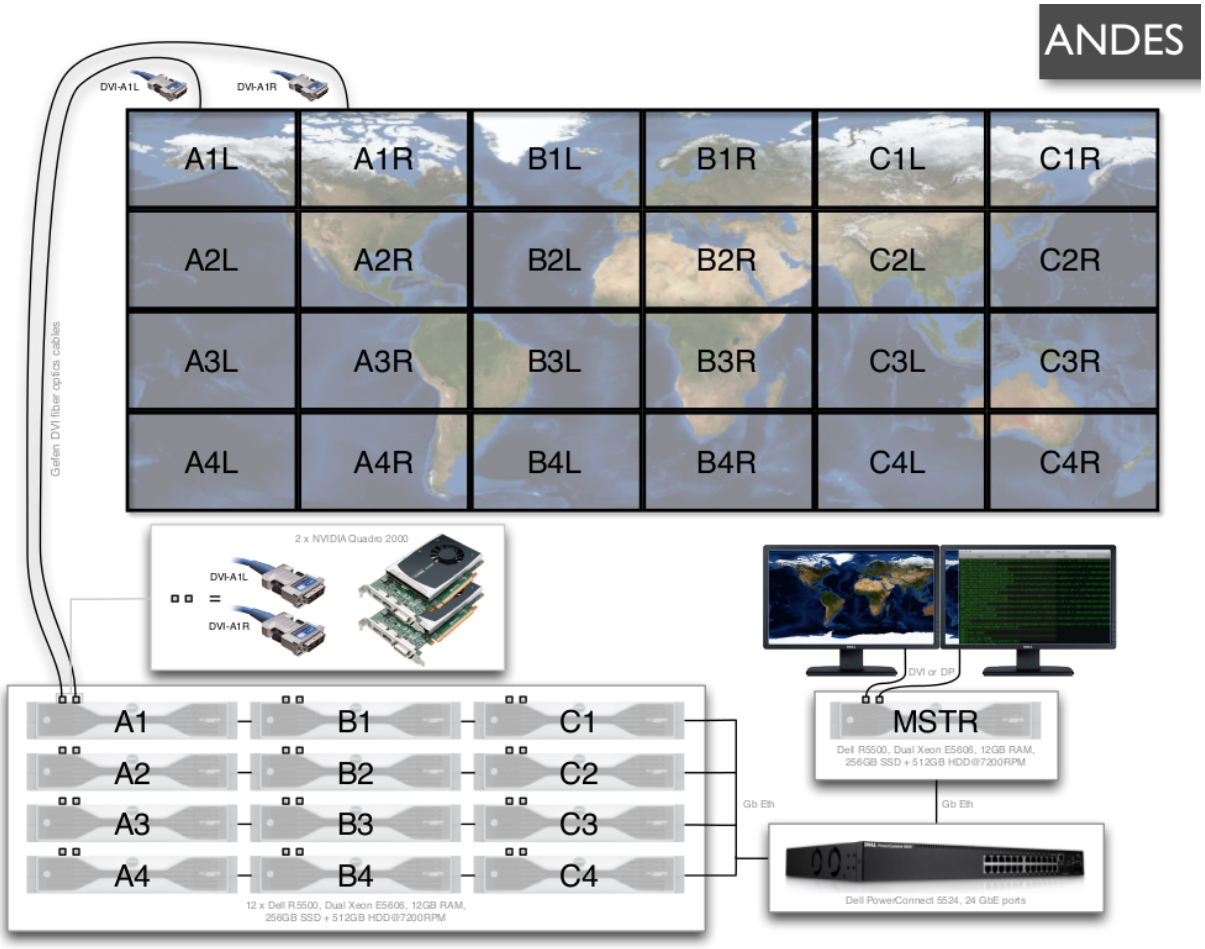


Figura 4.3: Representación gráfica de la arquitectura de hardware del *Andes wall display*

Capítulo 5

Funcionamiento de la solución

En este capítulo se presentarán las distintas funcionalidades implementadas para la aplicación **FITS-OW**, mostrando pantallazos de la interfaz de usuario y discutiendo las decisiones de diseño tomadas.

5.1. Pantalla de inicio

Cuando el usuario inicializa la aplicación, entregándole como parámetro la imagen *FITS*, el directorio de esta y la IP de su computador de escritorio o del computador *master* del *wall display*, se despliega una interfaz de usuario que visualiza la imagen.

Figura 5.1: Pantalla de inicio de **FITS-OW** en un computador de escritorio. En esta vista se muestra la imagen FITS que el usuario cargó al sistema al inicializarlo.

5.2. Menú principal

Para accesar el menú principal el usuario debe hacer *click* derecho en la imagen con el *mouse* en caso de estar usando un computador de escritorio. De estar ejecutando la aplicación en el *wall display*, debe seleccionar la opción de menú principal en el dispositivo que lo controle. Una vez en el menú principal, puede seleccionar la opción de aplicar una escala matemática o filtro de color a la imagen o seleccionar la opción para iniciar una consulta.

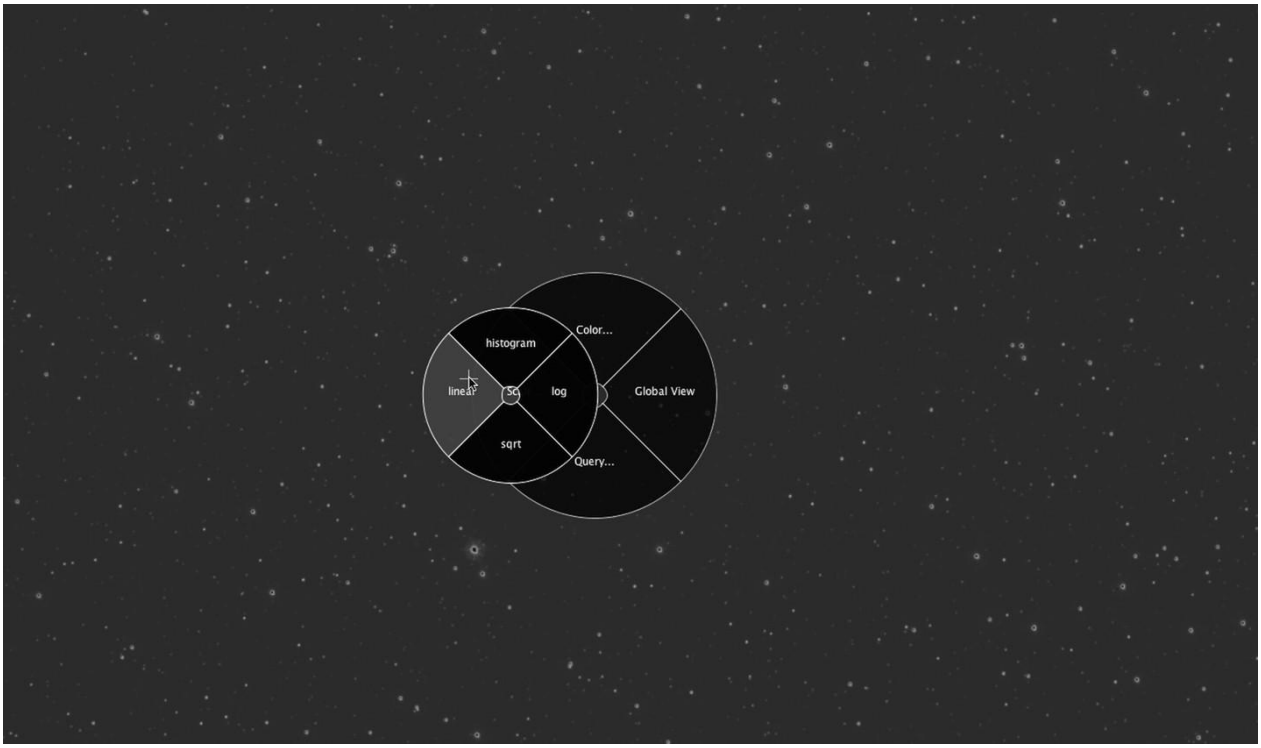


Figura 5.2: Menu principal de **FITS-OW**. En él pueden seleccionarse las siguientes opciones: aplicar una escala matemática a la imagen, aplicar un filtro de color o ejecutar una consulta.

5.3. Interfaz de elección de tipo de consulta

Una vez que el usuario selecciona la opción de consulta, aparece un panel en donde debe seleccionar si desea hacer una consulta por coordenadas, por identificador o *por script*. En caso de seleccionar la opción de consulta por *script*, aparecerá un panel en donde podrá ingresar el texto que desea ejecutar como consulta. En caso de seleccionar las opciones de consulta por identificador o por coordenadas aparecerá el panel de filtros de consulta.

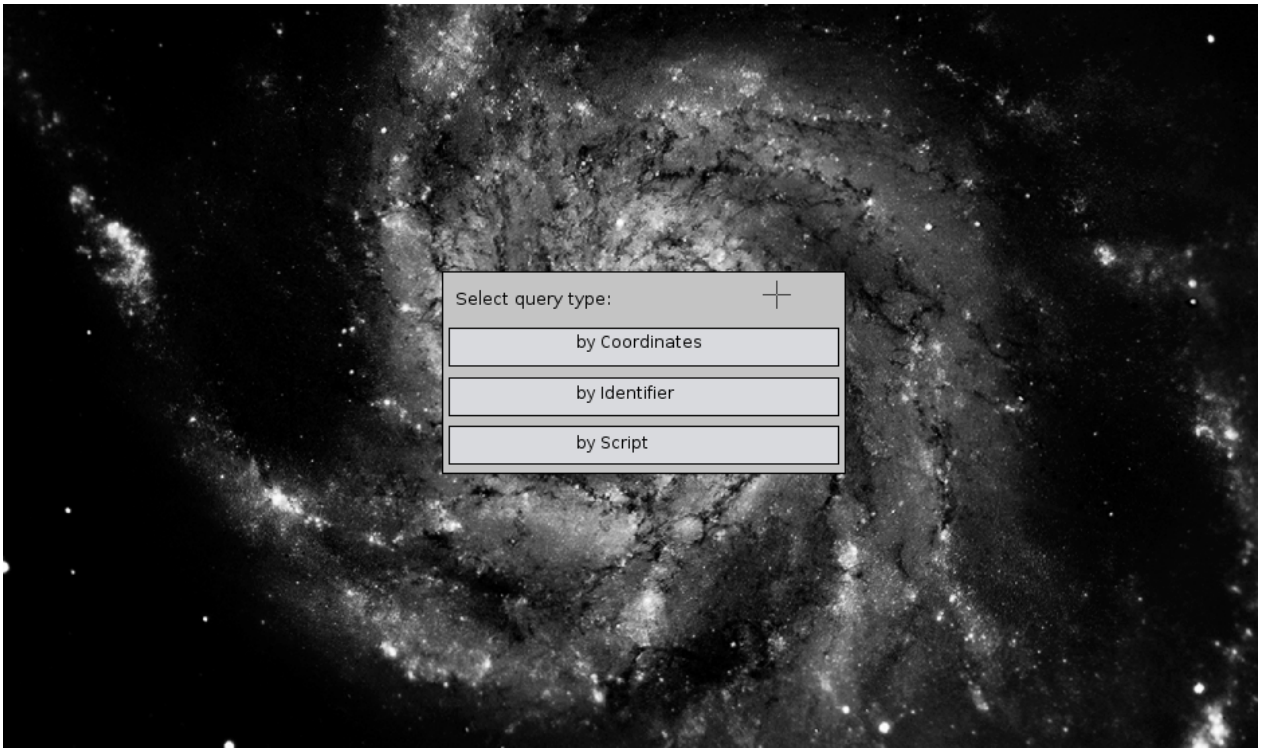


Figura 5.3: Interfaz de selección de tipo de consulta. Las consultas pueden hacerse por coordenadas, identificador o script, según elija el usuario.

5.4. Interfaz de selección de filtros de consulta

Existen dos tipos de filtros de consultas que el usuario puede seleccionar: por información básica del objeto celeste o por catálogo de mediciones. La selección se realiza mediante pestañas desplegadas en la parte superior de la interfaz. En caso de que se haya seleccionado la consulta por *script*, esta interfaz no se despliega, ya que el usuario sólo debe ingresar los comandos necesarios.

5.4.1. Interfaz de selección de filtros acerca de información básica

En esta interfaz se entregan los parámetros de consulta y los filtros opcionales. Los parámetros pedidos varían respecto a si se seleccionó una consulta por coordenadas o por identificador. La interfaz provee gráficamente las mismas opciones que la base de datos astronómica SIMBAD admite a través de consultas por *scripts*.

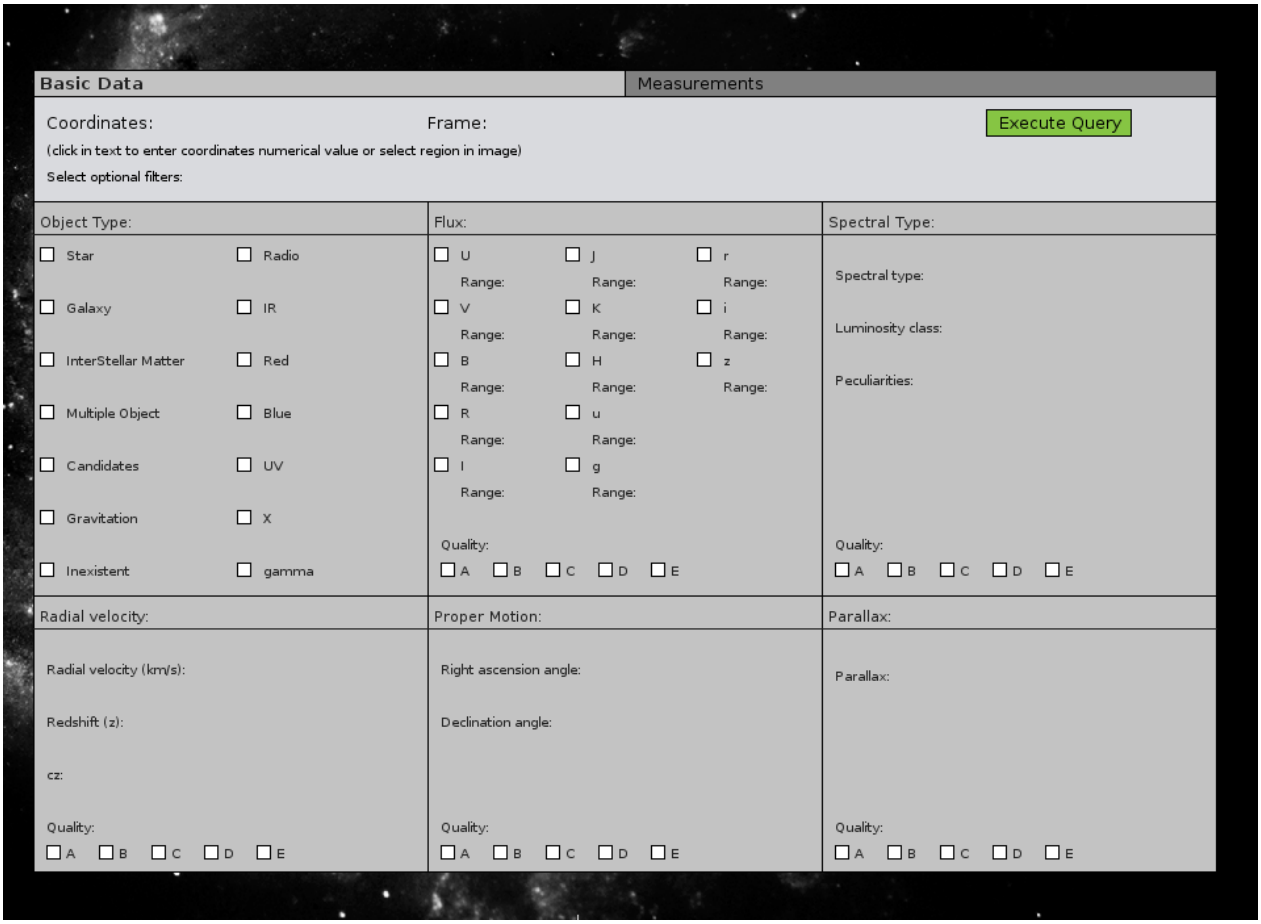


Figura 5.4: Interfaz de selección de información básica de consulta, por coordenadas. En el rectángulo superior el usuario puede ingresar las coordenadas en valor numérico.

En el caso de las consultas por coordenadas, el usuario puede seleccionar una región circular en la imagen, en vez de ingresar el valor numérico:

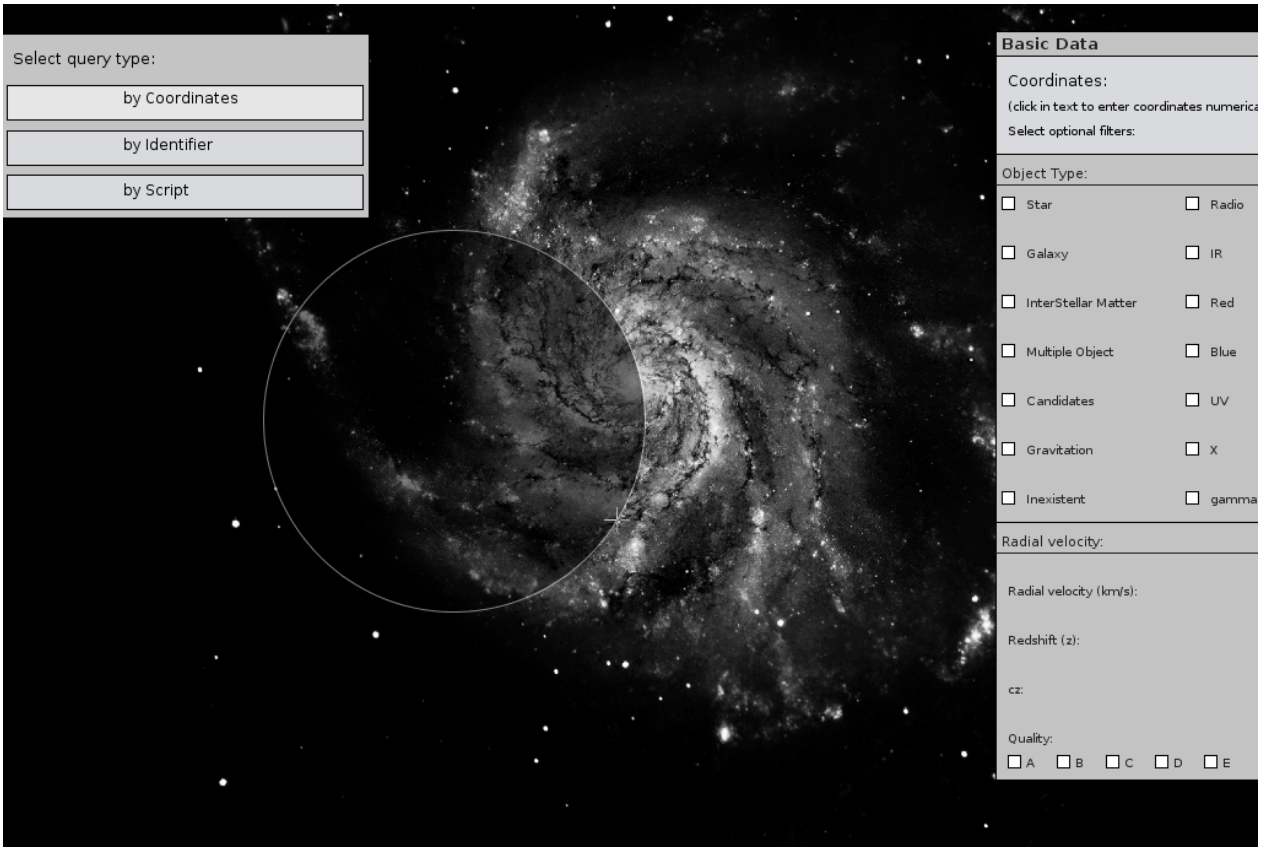


Figura 5.5: Interfaz de selección de información básica de consulta, en el caso de que el usuario decida seleccionar una región manualmente en vez de ingresar un valor numérico.

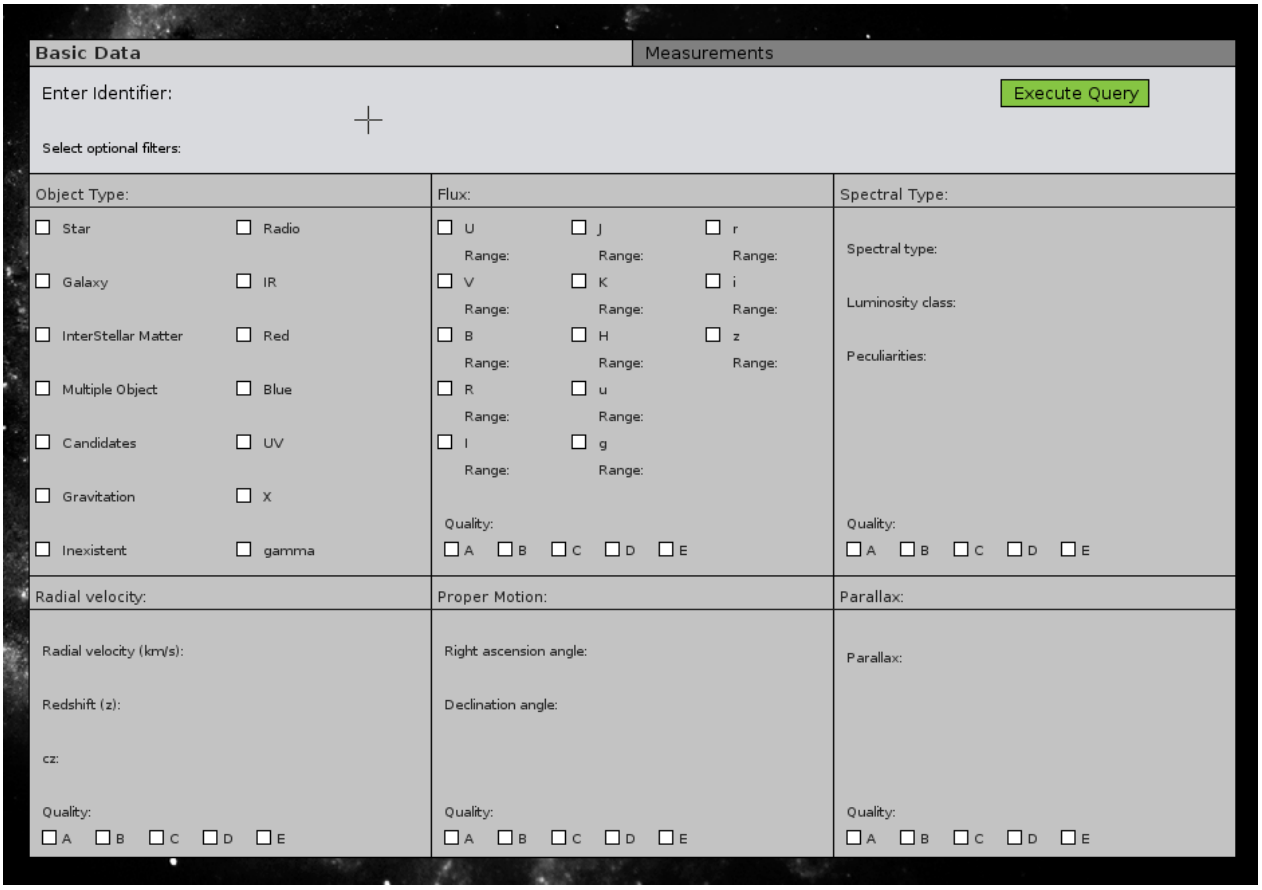


Figura 5.6: Interfaz de selección de información básica de consulta, por identificador. En la parte superior el usuario debe ingresar el identificador del objeto que desea consultar.

5.4.2. Interfaz de selección de filtros acerca de mediciones

Estas opciones de consulta fueron agregadas en caso de que el usuario esté buscando objetos con un tipo de medición específica, guardada en un catálogo particular. Se permite seleccionar cualquiera de los catálogos almacenados en la base de datos SIMBAD. Cabe destacar que este tipo de consultas afinan enormemente el tipo de búsqueda, pues generalmente los objetos astronómicos guardan mediciones de sólo unos pocos catálogos, o de ninguno.

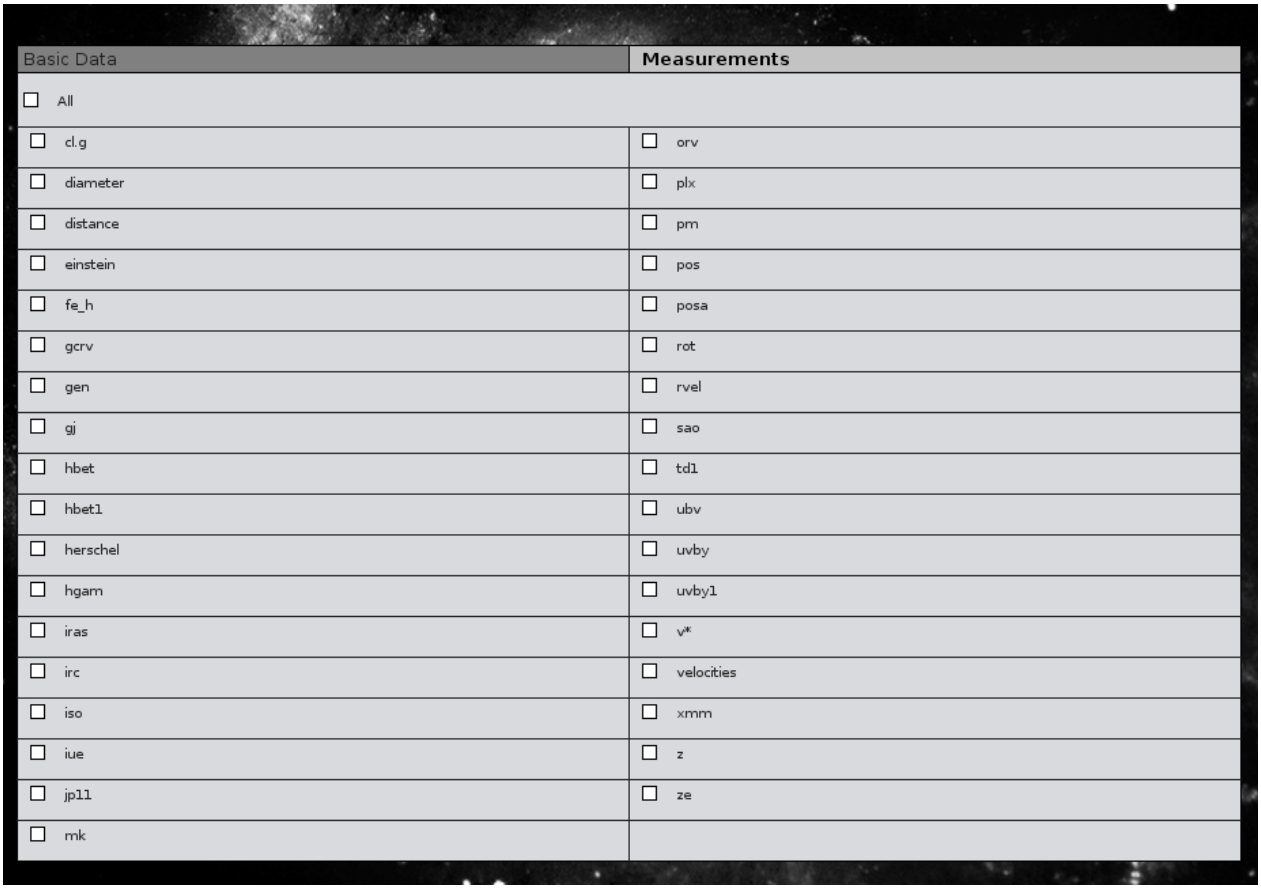


Figura 5.7: Interfaz de selección de filtros de catálogos. En ella se listan todos los catálogos a los que tiene acceso SIMBAD.

5.5. Interfaz de despliegue de resultados

La interfaz de despliegue de resultados marca los objetos encontrados en forma de cruces en la imagen FITS y además entrega un listado de estos. Al seleccionar un objeto en el listado, la cruz que marca la ubicación que le corresponde al objeto cambia de color rojo a verde. Al revés, al seleccionar una cruz marcando una ubicación, esta cambia de color y también selecciona el objeto que le corresponde en el listado.

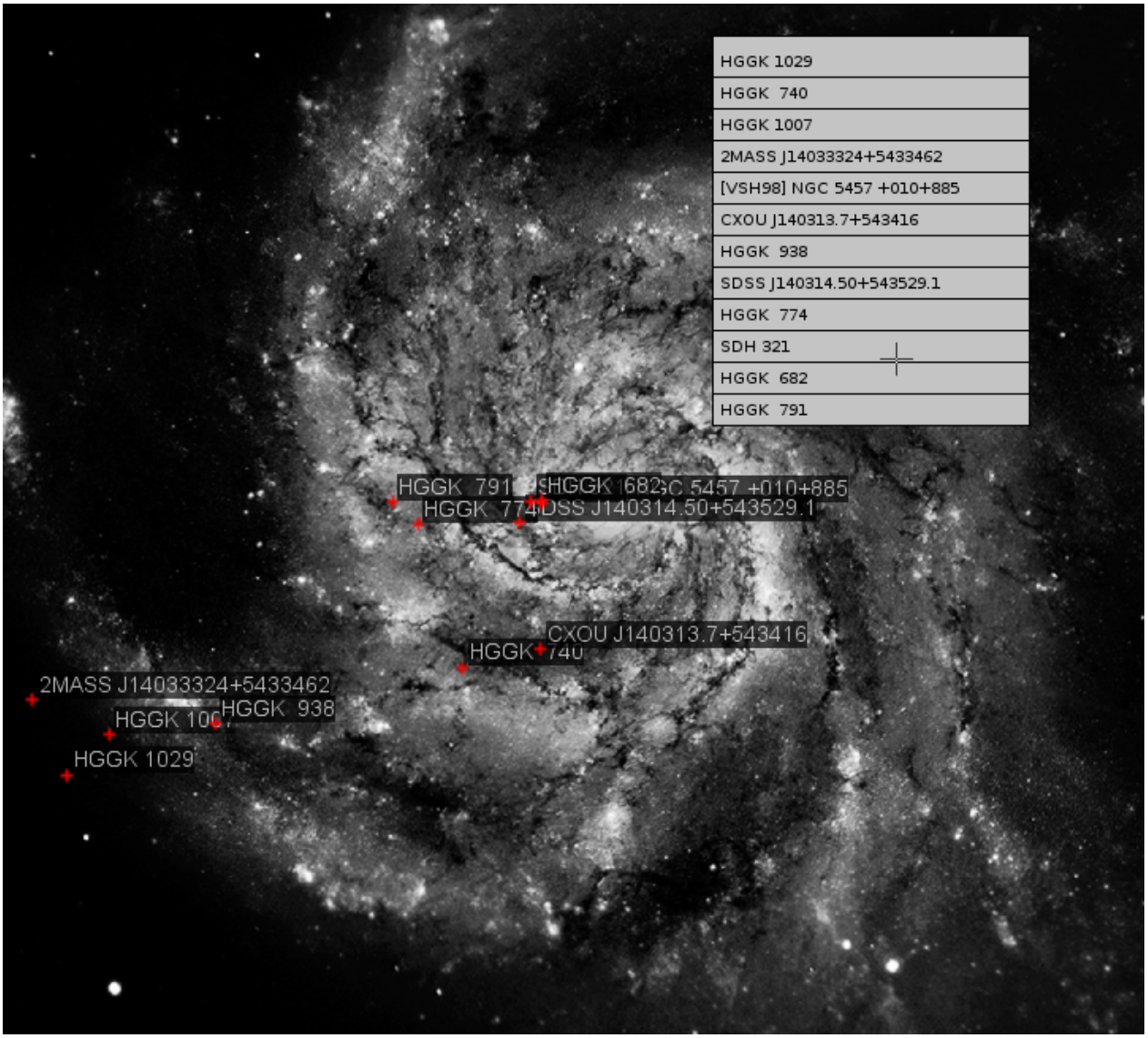


Figura 5.8: Interfaz de despliegue de resultados. Se marcan los objetos sobre la imagen y se despliega además una lista de estos.

5.5.1. Interfaz de información básica de un resultado

Al seleccionar un objeto del listado de resultados se abre un panel desplegando la información básica del objeto en cuestión. Este panel cuenta con una pestaña para desplegar las mediciones encontradas. Además, al seleccionar un resultado en particular, este se selecciona en verde en la imagen FITS.

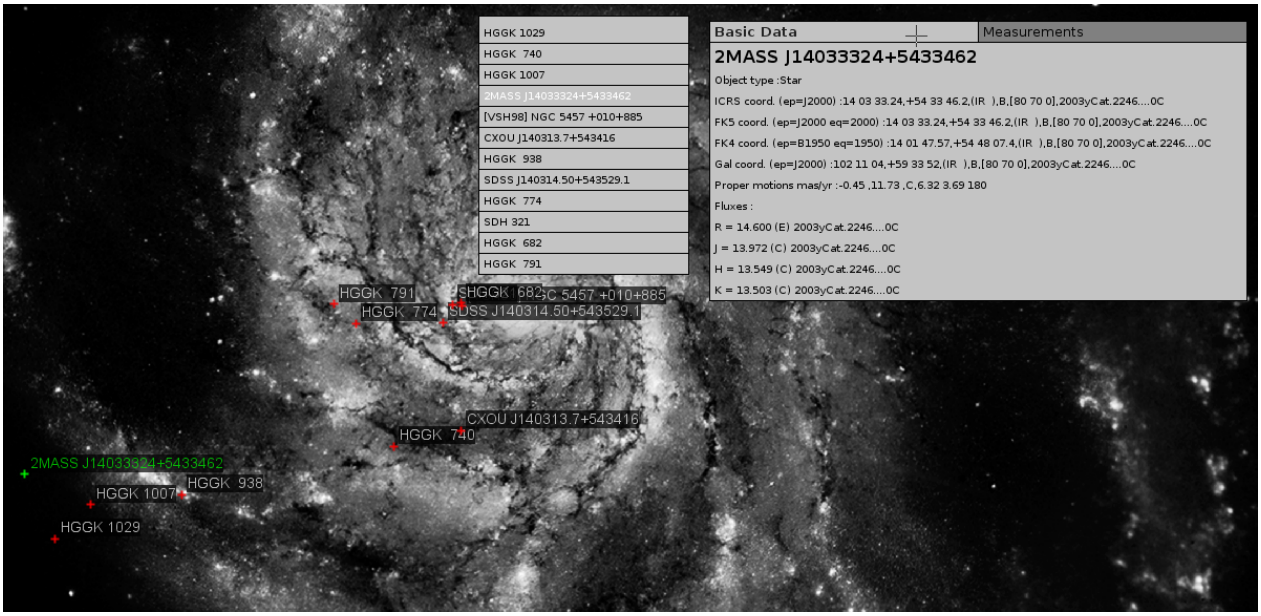


Figura 5.9: Interfaz de despliegue de información básica de resultados.

5.5.2. Interfaz de mediciones de un resultado

Al seleccionar en el panel de información básica la pestaña de mediciones, se despliega la información encontrada respecto de catálogos de medición. Cada medición se despliega en una tabla por separado.

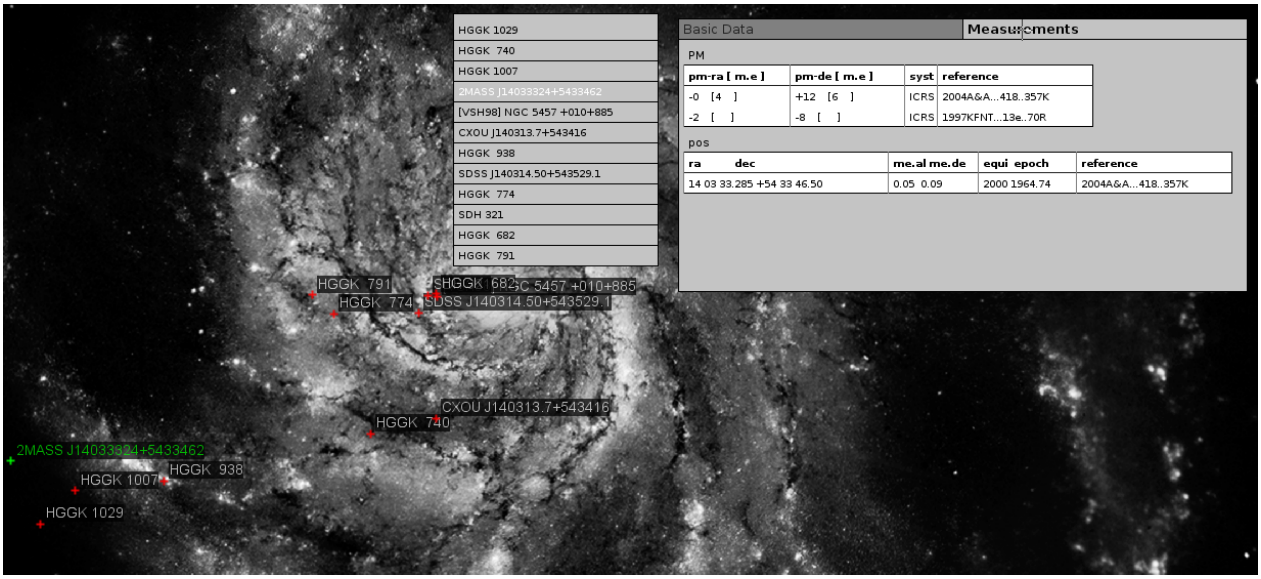


Figura 5.10: Interfaz de despliegue de mediciones.

5.6. Ejecución de la aplicación

Para ejecutar **FITS-OW** es necesario contar con la libería ZVTM. El código de esta librería está disponible en el recurso SourceForge.net y puede obtenerse a través de un repositorio SVN con el siguiente comando[13]

```
svn checkout svn://svn.code.sf.net/p/zvtm/code/ zvtm-code
```

Para agregar ZVTM al repositorio local se ejecutan los siguientes comandos:

```
mvn install
mvn clean
```

Para compilar la aplicación se utiliza la herramienta de administración de código *Maven*[11]. La compilación se lleva a cabo en un computador de escritorio con el siguiente comando:

```
mvn clean package
```

En un *wall display*, la compilación se realiza con el comando:

```
mvn -p Wall clean package
```

Es necesario asegurarse además de que cada computador del *cluster* cuente con una copia de la aplicación. Para ejecutar *FITS-OW* se utiliza el siguiente comando en un computador de escritorio:

```
./run.sh -IP localhost -fitsDir /ruta/a/la/imagen/fits/
-fits nombre_de_la_imagen.fits
```

Para correr la aplicación en el *Andes wall display* se utilizan dos comandos: uno para correr la aplicación en los nodos *slaves* del *cluster* y uno para ejecutarla en el computador *master*:

```
./andes_cluster_run.sh
```

```
./andes_master_run.sh -IP ip_del_coputador_master
-fitsDir /ruta/a/la/imagen/fits/ -fits nombre_de_la_imagen.fits
```

Para correr la aplicación en un *wall display* distinto de *Andes* se debe generar dos *scripts* de este tipo, en donde se especifica la arquitectura del *wall display* en cuestión (número de nodos, número de monitores, memoria requerida, etc).

5.7. Evaluación

Con el fin de evaluar el comportamiento de la aplicación, se realizaron consultas que retornan distintos órdenes de magnitud en número de resultados. Para poder comparar, las consultas se realizaron tanto en un computador de escritorio como en el *Andes wall display*.

5.7.1. Arquitectura de hardware del computador de escritorio

- Memoria: 57 GiB
- Procesador: Intel Core i5-240M
- CPU: 2,50 GHz × 4
- Sistema operativo: Ubuntu 14.04 LTS, 64-bits

5.7.2. Consultas realizadas

Se utilizó una imagen FITS que representa los *Pilares de la Creación* de la *Nebulosa de Águila* en blanco y negro. Para todas las consultas se utilizaron las mismas coordenadas como centro, pero se cambió el radio de consulta, con el fin de variar el número de resultados. Los radios utilizados fueron $0{,}5, 2, 5$ y 150 *arcmin*. Para cada radio se repitió la consulta cinco veces, con tal de obtener un promedio y aminorar anomalías. Las coordenadas centrales están en el *frame* ICRS y se definen por:

- Ángulo de ascensión recta $[ra] = 181852{,}56$
- Ángulo de declinación $[\mathrm{dec}] = -134941{,}6$;

5.7.3. Resultados

Para cada consulta se tomó el tiempo que la aplicación se demoró en ejecutar la consulta y el tiempo que se demoró en desplegar los resultados. A continuación se muestran los tiempos promedio medidos, ordenados por radio de consulta:

- Tiempos promedio en el computador de escritorio:

$r[arcmin]$	$n°$resultados	$t_{execution}[ms]$	$t_{display}[ms]$
$0,5$	17	$1256,26$	$41,25$
2	154	$2357,48$	$97,16$
5	1171	$10764,05$	$280,44$
150	13,999	$62285,66$	$11743,2$

Tabla 5.1: Tiempos promedio medidos en el computador de escritorio

- Tiempos promedio *en Andes wall display*:

$r[arcmin]$	$n°$resultados	$t_{execution}[ms]$	$t_{display}[ms]$
$0,5$	17	$1245,24$	$84,26$
2	154	$2131,98$	$385,5$
5	1171	$6613,19$	$3548,84$
150	13,999	$50631,34$	$34702,99$

Tabla 5.2: Tiempos promedio medidos en el *wall display*

En la figura siguiente se muestran gráficos con los datos anteriores. En el primer gráfico se muestran en líneas punteadas los tiempos de ejecución de consultas y en el segundo los tiempos de despliegue de resultados, para el computador de escritorio utilizado y para el *wall-display*.

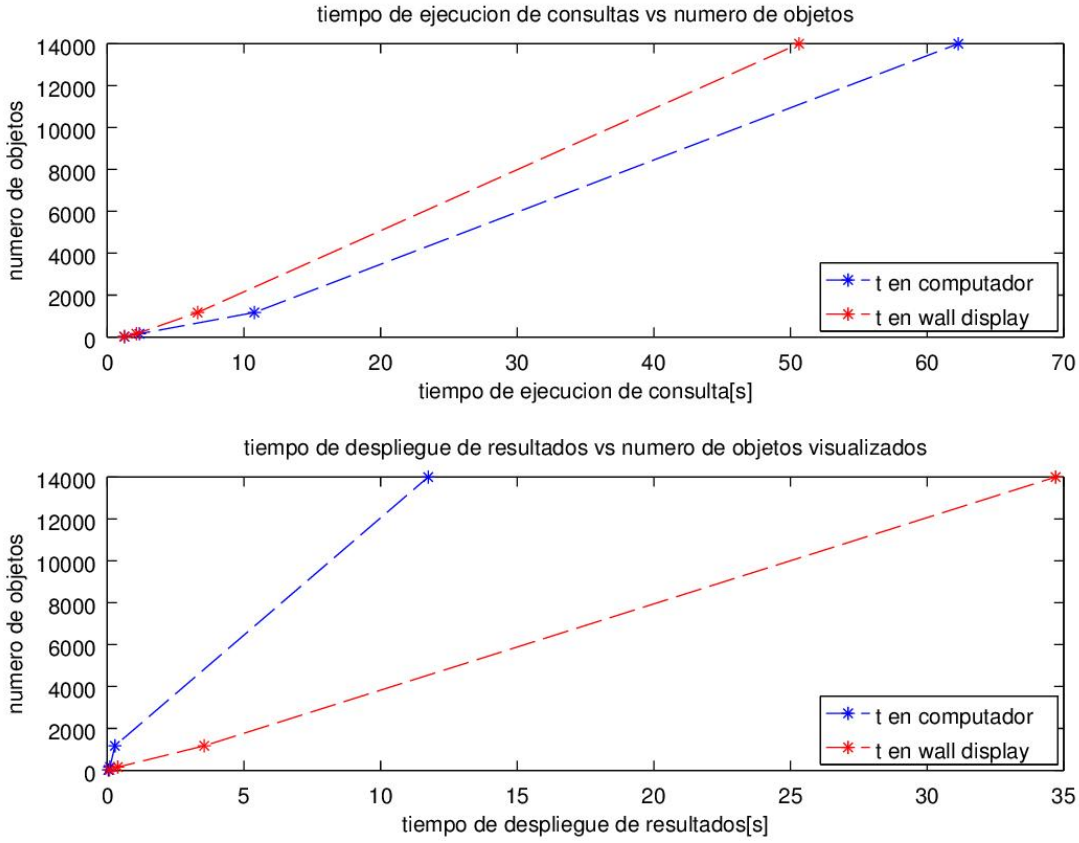


Figura 5.11: Gráficos que ilustran el crecimiento del tiempo de ejecución de consulta y de despliegue de resultados versus el número de objetos extraídos de SIMBAD.

Es posible apreciar que el tiempo de consulta no varía mucho entre el computador de escritorio y el *wall display*. Esto se debe a que estos tiempos dependen mayoritariamente de la respuesta de SIMBAD. Por otro lado, los tiempos de despliegue son mucho mayores en el *wall display* que en el computador de escritorio, contrario a lo que uno podría intuir. Esto se debe a que en el computador de escritorio no se despliegan todos los resultados para las consultas muy grandes, simplemente porque ZVTM renderiza dinámicamente lo que el usuario puede ver a través de un *view*. Dado que el *wall display* tiene una resolución mucho mayor que la del computador de escritorio, el espacio que el usuario observa es mucho mayor y por lo tanto, se renderizan muchos más objetos. Además, en el *wall-display* se debe calcular qué objeto es renderizado en qué monitor, lo que también añade tiempo extra.

Capítulo 6

Conclusión

En este trabajo de memoria se construyó sobre la aplicación **FITS-OW**, desarrollada por el equipo de *Massive data* de la fundación Inria Chile, en conjunto con el equipo *Ilda* de Inria Francia. **FITS-OW** es una aplicación para la visualización de imágenes astronómicas en un *wall dispay*; en este proyecto se le agregó un módulo que permite consultar la base de datos externa de objetos astronómicos extra solares SIMBAD acerca de la imagen visualizada.

La aplicación puede correr tanto en un computador de escritorio como en un *wall display*, por lo que se implementó el manejo de interfaz de usuario a través de dispositivos comunes, como *mouse* y teclado, y a través de dispositivos más aptos para manejar un *wall display*, como *tablets* y *smartphones*.

El módulo de consultas de **FITS-OW** agregado durante este proyecto permite al usuario pedir información a la base de datos SIMBAD entregando coordenadas astronómicas, un identificador de objeto astronómico o simplemente el código de *script* de consulta. Los resultados de la consulta se despliegan en paralelo a la imagen visualizada y el usuario puede interactuar con ellos, seleccionando información acerca de un objeto específico.

6.1. Cumplimiento de objetivos

El módulo *simbad*, que se encarga del correcto funcionamiento de la interfaz gráfica de consultas a la base de datos SIMBAD y de desplegar los resultados obtenidos en paralelo a la imagen FITS visualizada, fue implementado a cabalidad, por lo que se concluye que el objetivo general, definido en la sección 1.2 se cumple. A continuación se analizará el cumplimiento de los objetivos específicos, propuestos en la sección 1.3.

- *Implementar las consultas que la base de datos debe ser capaz de realizar*: El módulo agregado a **FITS-OW** posee la capacidad de realizar las consultas definidas, en particular: Consultas por coordendas, identificador de objetos y *script* de consulta. Además se permite añadir filtros opcionales: por tipo de objeto astronómico, por rango y cali-

dad de movimiento propio, paralaje, velocidad radial, tipo espectral y flujo, además de permitir filtrado por mediciones en catálogos específicos.

- *Diseñar e implementar una interfaz de usuario para un wall-display tanto para realizar consultas como para desplegar sus resultados*: La aplicación actual cuenta con una interfaz gráfica que funciona tanto en computador de escritorio como en un *wall-display*, en donde se pueden especificar los parámetros y filtros de consulta. **FITS-OW** se conecta con SIMBAD y es capaz de extraer los resultados que ahí se encuentran. Además cuenta con elementos gráficos que permiten desplegar un listado de resultados y que marcan en la imagen *FITS* la ubicación de estos. El usuario puede seleccionar un objeto astronómico particular y visualizar la información básica o las mediciones relacionadas con este.
- *Diseñar e implementar las estructuras de datos pertinentes para almacenar las respuestas obtenidas y poder manejarlas como objetos*: La aplicación cuenta con la estructura de datos *AstroObject*, que almacena la información básica de los objetos astronómicos encontrados por la consulta y que se relaciona con la estructura de datos *Measurement*, que guarda las mediciones correpondientes.

6.2. Dificultades técnicas

A lo largo del trabajo de memoria se encontraron algunas dificultades durante del proceso de implementación, que se discuten a continuación.

6.2.1. Construcción de glyphs

Para construir la interfaz de usuario de consultas para **FITS-OW** se utilizó la librería ZVTM, debido a su compatibilidad con *JBricks* y con el desarrollo en un *wall display*. ZVTM se encuentra en un nivel de abstracción medio entre librerías gráficas de alto nivel, como *Java Swing*, y de bajo nivel, como *OpenGL*. Esto en concreto significa que si se quiere construir, por ejemplo, un botón, en *Java Swing* basta con crear un objeto de tipo botón; en *OpenGL* es necesario definir todos los vectores gráficos que arman la forma de un botón; pero en *ZVTM* se debe construir primero un objeto que represente un rectángulo y luego un objeto que represente un texto dentro del rectángulo.

Dado que la interfaz de usuario contruida está compuesta de varios textos, secciones y botones, resultó bastante engorroso tener que definir la posición, tamaño y contención de todos los objetos que forman un *glyph* más complejo.

Debido a que ZVTM se utiliza sostenidamente en la fundación INRIA Chile para el desarrollo de aplicaciones en el *Andes wall display*, como trabajo futuro se pretende enriquecer la librería *ZVTM*, agregándole *glyphs* estándares más complejos. De esta forma, los desarrolladores de la fundación y en particular los miembros del equipo *Massive data* no tendremos

que implementar formas comunes (como botones) cada vez que se inicie el desarrollo de una aplicación nueva.

6.2.2. Glyphs compuestos en el wall display

La implementación de la solución a la memoria se llevó a cabo principalmente en un computador de escritorio, realizando pruebas regulares en el *Andes wall display*, para asegurarse de que los elementos desarrollados funcionaran en ambas plataformas. Inicialmente, se pensó que con tan sólo utilizar la librería *JBricks* el código podría ejecutarse sin problemas, pero esto en la práctica probó no ser el caso. Debido a que en un *wall display* cada cambio en la vista del usuario puede involucrar más de una sóla pantalla y más de un sólo computador, cada nodo del *cluster* debe ser notificado de los cambios de apariencia de los *glyphs*. Esto es automático y funciona muy bien para los *glyphs* ya definidos en ZVTM, pero no es así para los *glyphs* compuestos creados por los desarrolladores.

Dado que la interfaz gráfica de consultas de **FITS-OW** está formada principalmente por *glyphs* compuestos, la primera vez que se probó la aplicación en el *Andes wall display* los eventos de interacción con la interfaz no producían ningún cambio en los monitores, a pesar de que sí se podían apreciar en el monitor del computador *master*. Pare resolver este problema se añadió un archivo en *AspectJ* llamado *AutoReplay.aj* en donde se especifican todas las clases y métodos que alteran de alguna manera la apariencia de los *glyphs*. Con este archivo, los nodos del *cluster* saben que cuando cualquiera de estos métodos es ejecutado, deben refrescar la renderización en los monitores.

6.2.3. FITS-OW como prototipo

Dado que la primera etapa de desarrollo de **FITS-OW** se llevó a cabo a modo de prototipo, añadirle funcionalidades fue a ratos enredado, pues el código no siempre seguía un diseño de software pensado en escalabilidad de la aplicación. Varios archivos, principalmente los controladores de eventos, no estaban programados pensando en programación orientada al objeto, sino que más bien se parecían a la estructura de un programa escrito en C.

6.3. Otras dificultades

Si bien el cliente aprobó el funcionamiento de la aplicación **FITS-OW** en el *wall-display* de Inria Chile, acutalmente no existe el *hardware* necesario en el Instituto de Astrofísica de la Universidad Católica como para poder evaluar a cabalidad la usabilidad sostenida del *software*, por lo que no se llevaron a cabo pruebas de usario concluyentes. Se espera que el Instituto de Astrofísica de la Universidad Católica consiga los fondos necesarios para instalar un *wall-display* (aunque de menores proporciones que el de las instalaciones de Inria Chile) a futuro, con lo que se podría evaluar la usabilidad de **FITS-OW**.

6.4. Trabajo futuro

En esta fase de desarrollo se eligió **SIMBAD** como base de datos para ejecutar consultas, sin embargo, se pretende que en el futuro se puedan añadir otros catálogos existentes. Con esto en mente, esta fase se empaquetó en un sólo módulo llamado *simbad*, para que sea sencillo agregar otras bases de datos en la posteridad.

Además, al cargar la imagen FITS que se quiere visualizar a **FITS-OW**, se levanta un servidor que deja esta imagen a disposición de quien quiera usarla. Esto es útil en primera instancia para realizar la comunicación entre el computador *master* y los computadores *slaves* del *cluster* de un *wall display*, ya que de esta forma cada *slave* obtiene la imagen desde el servidor y calcula qué parte de esta debe renderizar. Esto es más rápido que copiar la imagen a cada uno de los computadores del *cluster* de antemano, pero además tiene la ventaja de permitir la posibilidad de conectar la aplicación a bases de datos que guarden imágenes. Así, la aplicación podría a futuro conectarse a *Aladin*, por ejemplo, descargar imágenes asociadas a la sección del cielo que se está estudiando y disponerlas en el servidor, para que cada *slave* del *cluster* tenga una copia y puedan desplegarse en el *wall display*.

Otra extensión posible para **FITS-OW** sería no sólo permitir búsqueda en catálogos externos, sino que también en catálogos locales.

Cabe destacar que la aplicación **FITS-OW** será presentada el 28 de Junio del 2016 en Edimburgo, Escocia, en la conferencia de SPIE *Software and Cyberinfrastructure for Astronomy IV*. El artículo final será enviado a SPIE el 30 de Mayo del 2016, pero acutalmente está disponible el *abstract* correspondiente en la página web [46].

Bibliografía

[1] ALMA home page, `http://almaobservatory.org/`, URL accesada el 2 de Marzo del 2016.

[2] ASKAP home page, `http://www.atnf.csiro.au/projects/askap/computing.html`, URL accesada el 2 de Marzo del 2016.

[3] SIMBAD home page, `http://simbad.u-strasbg.fr/simbad/`, URL accesada el 2 de Marzo del 2016.

[4] Amr Hassan y Christopher J. Fluke, *Scientific Visualization in Astronomy: Towards the Petascale Astronomy Era*, 2011, arXiv Astrophysics e-print, arXiv:1102.5123v1 [astro-ph.IM], `http://arxiv.org/pdf/1102.5123v1.pdf`, URL accesada el 2 de Marzo del 2016.

[5] Emmanuel Pietriga, *Languagues and Interaction Techniques for Visualization and Manipulation of Massive Datasets*, Human-Computer Interaction [cs.HC], Université Paris Sud-Paris XI, 2012, `https://tel.archives-ouvertes.fr/tel-00709533/document`, URL accesada el 2 de Marzo del 2016.

[6] INRIA Chile home page, `http://inria.cl`, URL accesada el 2 de Marzo del 2016.

[7] Alyssa A. Goodman, *Principles of High-Dimensional Data Visualization in Astronomy*, 2012, arXiv Astrophysics e-print, arXiv:1205.4747v1, `http://arxiv.org/pdf/1205.4747v1.pdf`, URL accesada el 2 de Marzo del 2016.

[8] Consultas a SIMBAD por URL, `http://http://simbad.u-strasbg.fr/guide/sim-url.htx`, URL accesada el 2 de Marzo del 2016.

[9] Niveles de jerarquía de datos de SIMBAD, `http://simbad.u-strasbg.fr/simbad/sim-display?data=o` URL accesada el 2 de Marzo del 2016.

[10] Estructura de los datos de SIMBAD, `http://simbad.u-strasbg.fr/guide/ch15.htxs`, URL accesada el 2 de Marzo del 2016.

[11] Emmanuel Pietriga y Romain Primet, *ZVTM Developer's Guide*, 2012, `http://zvtm.sourceforge.net/doc/zvtm_dev_guide.pdf`, URL accesada el 2 de Marzo del 2016.

[12] ZVTM home page, http://zvtm.sourceforge.net/, URL accesada el 2 de Marzo del 2016.

[13] Instrucciones para obtener la librería ZVTM, http://zvtm.sourceforge.net/download.html, URL accesada el 2 de Marzo del 2016.

[14] Emmanuel Pietriga, Stéphane Huot, Mathieu Nancel y Romain Primet,*Rapid Development of User Interfaces on Cluster-Driven Wall Displays with jBricks*, ACM Press. EICS '11: 2nd ACM SIGCHI symposium on Engineering interactive computing systems, Jun 2011, Pisa, Italy. 2011, EICS '11: Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems. https://hal.inria.fr/inria-00582640v2/document, URL accesada el 2 de Marzo del 2016.

[15] Bernard F. Meade, Christopher J. Fluke, Steven Manos, Richard O. Sinnott, *Are tiled display walls needed for astronomy?*, 2014, arXiv Astrophysics e-print, arXiv:1407.4881v1 [astro-ph.IM] http://arxiv.org/pdf/1407.4881v1.pdf, URL accesada el 2 de Marzo del 2016.

[16] Maria Lobo, Emmanuel Pietriga, Claude Puech, *Visualización de Big Data en Alta Resolución a Disposición de la Comunidad Científica y la Industria Chilena*. Big data. 2015, pp.6. ,hal-01134134, https://hal.inria.fr/hal-01134134/document, URL accesada el 2 de Marzo del 2016.

[17] Mathieu Nancel, Olivier Chapuis, Emmanuel Pietriga, Xing-Dong Yang, Irani Pourang, Michel Beaudouin-Lafon,*High-Precision Pointing on Large Wall Displays using Small Handheld Devices*. CHI '13: SIGCHI Conference on Human Factors and Computing Systems, 2013, Paris, France. ACM, pp.831-840, 2013, 10.1145/2470654.2470773. hal-00786532v2. https://hal.archives-ouvertes.fr/hal-00786532v2/document, URL accesada el 2 de Marzo del 2016.

[18] Olivier Chapuis, Anastasia Bezarianos, Stelios Frantzeskakis, *Smarties: An Input System for Wall Display Development*, Toronto, Canada. ACM, pp.763-2772, 2014, 10.1145/2556288.2556956. hal-00979034v2 https://hal.archives-ouvertes.fr/hal-00979034/document, URL accesada el 2 de Marzo del 2016.

[19] *A History of Star Catalogs*, Rick Thurmond, 2003, http://www.rickthurmond.com/HistoryOfStarCatalogs.pdf, URL accesada el 2 de Marzo del 2016.

[20] *The Foundation of Celestial Mechanics*, George W. Collins, Segunda edición, 2004, http://ads.harvard.edu/books/1989fcm..book/Chapter2.pdf, URL accesada el 2 de Marzo del 2016.

[21] *Naming of Astronomical Objects*, IAU (International Astronomical Union), https://www.iau.org/public/themes/naming/, URL accesada el 2 de Marzo del 2016.

[22] FITS (File Image Transport System) home page, http://fits.gsfc.nasa.gov/, URL accesada el 2 de Marzo del 2016.

[23] Christopher Andrews, Alex Endert, Beth Yost, Chris North, *Information visualization on large, high-resolution displays: Issues, challenges, and opportunities*, 2011,http://www.cc.gatech.edu/ aendert3/resources/AndrewsLHRD.pdf, URL accesada el 2 de Marzo del 2016.

[24] Emmanuel Pietriga y Michel Beaudoiun-Lafon, *WILD (Wall-sized interaction with large datasets)*, inSitu Situated Interaction Lab, http://insitu.lri.fr/Projects/WILD, URL accesada el 2 de Marzo del 2016.

[25] SDSS DR12 home page, http://www.sdss.org/, URL accesada el 2 de Marzo del 2016.

[26] 2MASS (The Two Micron All Sky Survey System) home page, http://www.ipac.caltech.edu/2mass/, URL accesada el 2 de Marzo del 2016.

[27] Celestia home page, http://www.shatters.net/celestia/, URL accesada el 2 de Marzo del 2016.

[28] Orbiter home page, http://orbit.medphys.ucl.ac.uk/, URL accesada el 2 de Marzo del 2016.

[29] Google Sky home page, https://www.google.com/sky/, URL accesada el 2 de Marzo del 2016.

[30] Sky-Map.org home page, http://www.sky-map.org/, URL accesada el 2 de Marzo del 2016.

[31] Sky-X Astronomy Software home page, http://www.bisque.com/sc/pages/TheSkyX-Editions.aspx, URL accesada el 2 de Marzo del 2016.

[32] Stellarium home page, http://www.stellarium.org/, URL accesada el 2 de Marzo del 2016.

[33] Starry Night home page, http://astronomy.starrynight.com/, URL accesada el 2 de Marzo del 2016.

[34] Aladin Sky Atlas home page, http://aladin.u-strasbg.fr/, URL accesada el 2 de Marzo del 2016.

[35] SkyView Virtual Observatory home page, http://skyview.gsfc.nasa.gov/current/cgi/titlepage.pl URL accesada el 2 de Marzo del 2016.

[36] World Wide Telescope home page, http://www.worldwidetelescope.org/, URL accesada el 2 de Marzo del 2016.

[37] Nightshade home page, http://www.nightshadesoftware.org/projects/nightshade, URL accesada el 2 de Marzo del 2016.

[38] VizieR home page, http://webviz.u-strasbg.fr/viz-bin/VizieR, URL accesada el 2 de Marzo del 2016.

[39] HyperLeda home page,http://leda.univ-lyon1.fr/, URL accesada el 2 de Marzo del 2016.

[40] HLA (The Hubble Legacy Archive) home page, http://hla.stsci.edu/STECF.org/archive/hla/, URL accesada el 2 de Marzo del 2016.

[41] NED home page (The NASA/IPAC Extragalactic Database), http://ned.ipac.caltech.edu/, URL accesada el 2 de Marzo del 2016.

[42] ESO Data Archive home page, http://archive.eso.org/eso/eso_archive_main.html, URL accesada el 2 de Marzo del 2016.

[43] Mikulski Archive for Space Telescopes home page, http://archive.stsci.edu/, URL accesada el 2 de Marzo del 2016.

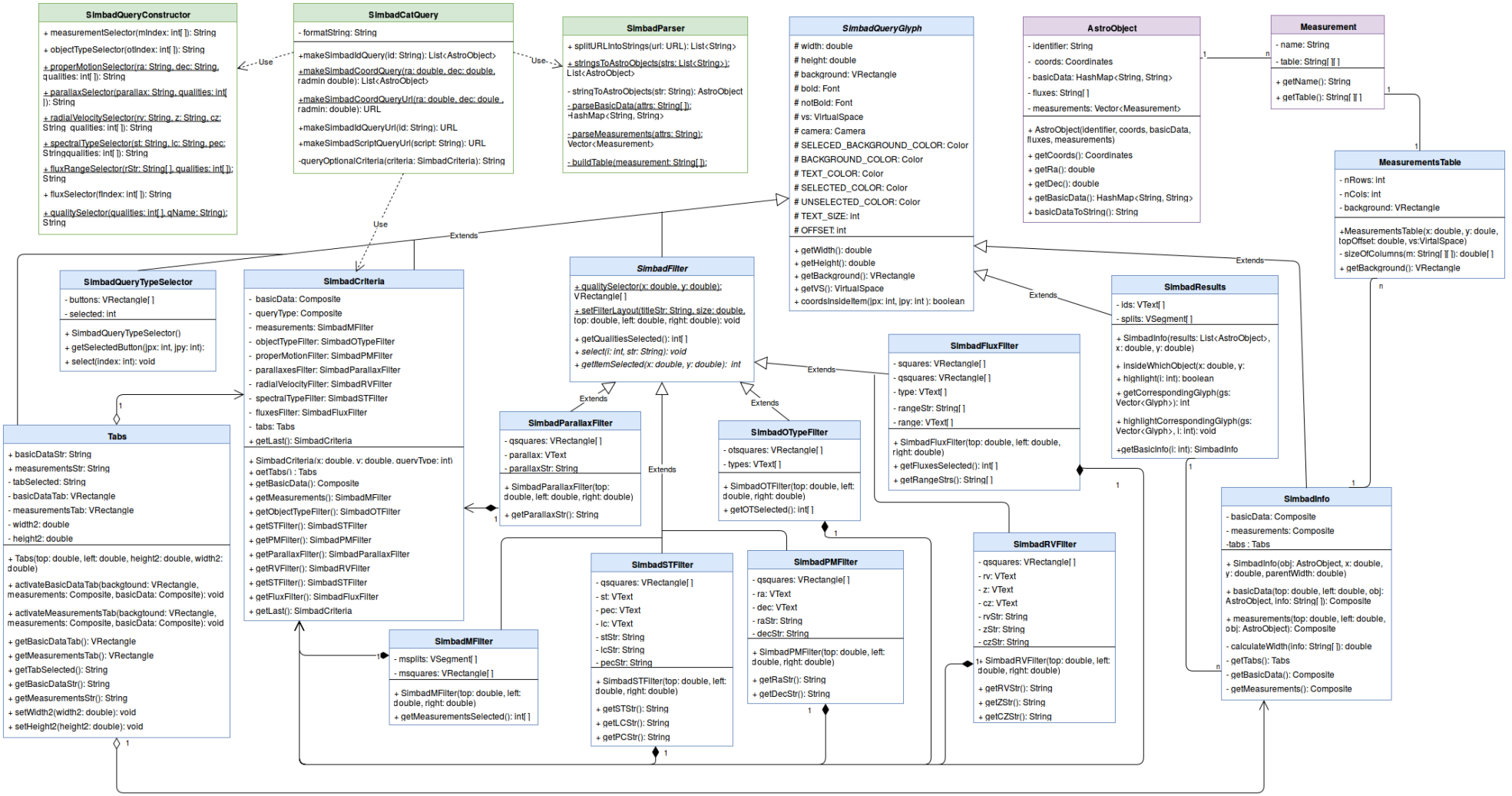
[44] JSky home page, http://archive.eso.org/cms/tools-documentation/jsky.html, URL accesada el 2 de Marzo del 2016.

[45] Montage *An Astronomical Image Mosaic Engine* home page, http://montage.ipac.caltech.edu/, URL accesada el 2 de Marzo del 2016.

[46] SPIE, detalles sobre la conferencia *Software and Cyberinfrastructure for Astronomy IV*, http://spie.org/AS/conferencedetails/astronomy-control-software#2231191, URL accesada el 2 de Marzo del 2016.

Anexos

Diagrama de clases


SimbadQueryConstructor
+ measurementSelector(mIndex: int[ ]): String
+ objectTypeSelector(otIndex: int[ ]): String
+ properMotionSelector(ra: String, dec: String, qualities: int[ ]): String
+ parallaxSelector(parallax: String, qualities: int[ ]): String
+ radialVelocitySelector(rv: String, z: String, cz: String qualities: int[ ]): String
+ spectralTypeSelector(st: String, lc: String, pec: Stringqualities: int[ ]): String
+ fluxRangeSelector(rStr: String[ ], qualities: int[ ]): String
+ fluxSelector(fIndex: int[ ]): String
+ qualitySelector(qualities: int[ ], qName: String): String
SimbadCatQuery
- formatString: String
+makeSimbadIdQuery(id: String): List<AstroObject>
+makeSimbadCoordQuery(ra: double, dec: double, radmin double): List<AstroObject>
+makeSimbadCoordQueryUrl(ra: double, dec: doule , radmin: double): URL
+makeSimbadIdQueryUrl(id: String): URL
+makeSimbadScriptQueryUrl(script: String): URL
-queryOptionalCriteria(criteria: SimbadCriteria): String
SimbadParser
+ splitURLIntoStrings(url: URL): List<String>
+ stringsToAstroObjects(strs: List<String>): List<AstroObject>
- stringToAstroObjects(str: String): AstroObject
- parseBasicData(attrs: String[ ]): HashMap<String, String>
- parseMeasurements(attrs: String): Vector<Measurement>
- buildTable(measurement: String[ ]);
SimbadQueryGlyph
width: double
height: double
background: VRectangle
bold: Font
notBold: Font
vs: VirtualSpace
camera: Camera
SELECED_BACKGROUND_COLOR: Color
BACKGROUND_COLOR: Color
TEXT_COLOR: Color
SELECTED_COLOR: Color
UNSELECTED_COLOR: Color
TEXT_SIZE: int
OFFSET: int
+ getWidth(): double
+ getHeight(): double
+ getBackground(): VRectangle
+ getVS(): VirtualSpace
+ coordsInsideItem(jpx: int, jpy: int): boolean
AstroObject
- identifier: String
- coords: Coordinates
- basicData: HashMap<String, String>
- fluxes: String[ ]
- measurements: Vector<Measurement>
+ AstroObject(identifier, coords, basicData, fluxes, measurements)
+ getCoords(): Coordinates
+ getRa(): double
+ getDec(): double
+ getBasicData(): HashMap<String, String>
+ basicDataToString(): String
Measurement
- name: String
- table: String[ ][ ]
+ getName(): String
+ getTable(): String[ ][ ]
MeasurementsTable
- nRows: int
- nCols: int
- background: VRectangle
+MeasurementsTable(x: double, y: doule, topOffset: double, vs:VirtalSpace)
- sizeOfColumns(m: String[ ][ ]): double[ ]
+ getBackground(): VRectangle
SimbadQueryTypeSelector
- buttons: VRectangle[ ]
- selected: int
+ SimbadQueryTypeSelector()
+ getSelectedButton(jpx: int, jpy: int):
+ select(index: int): void
SimbadCriteria
- basicData: Composite
- queryType: Composite
- measurements: SimbadMFilter
- objectTypeFilter: SimbadOTypeFilter
- properMotionFilter: SimbadPMFilter
- parallaxesFilter: SimbadParallaxFilter
- radialVelocityFilter: SimbadRVFilter
- spectralTypeFilter: SimbadSTFilter
- fluxesFilter: SimbadFluxFilter
- tabs: Tabs
+ getLast(): SimbadCriteria
+ SimbadCriteria(x: double, y: double, queryType: int)
+ getTabs() : Tabs
+ getBasicData(): Composite
+ getMeasurements(): SimbadMFilter
+ getObjectTypeFilter(): SimbadOTFilter
+ getSTFilter(): SimbadSTFilter
+ getPMFilter(): SimbadPMFilter
+ getParallaxFilter(): SimbadParallaxFilter
+ getRVFilter(): SimbadRVFilter
+ getSTFilter(): SimbadSTFilter
+ getFluxFilter(): SimbadFluxFilter
+ getLast(): SimbadCriteria
SimbadFilter
+ qualitySelector(x: double, y: double): VRectangle[ ]
+ setFilterLayout(titleStr: String, size: double, top: double, left: double, right: double): void
+ getQualitiesSelected(): int[ ]
+ select(i: int, str: String): void
+ getItemSelected(x: double, y: double): int
SimbadResults
- ids: VText[ ]
- splits: VSegment[ ]
+ SimbadInfo(results: List<AstroObject>, x: double, y: double)
+ insideWhichObject(x: double, y: double)
+ highlight(i: int): boolean
+ getCorrespondingGlyph(gs: Vector<Glyph>): int
+ highlightCorrespondingGlyph(gs: Vector<Glyph>, i: int): void
+getBasicInfo(i: int): SimbadInfo
SimbadFluxFilter
- squares: VRectangle[ ]
- qsquares: VRectangle[ ]
- type: VText[ ]
- rangeStr: String[ ]
- range: VText[ ]
+ SimbadFluxFilter(top: double, left: double, right: double)
+ getFluxesSelected(): int[ ]
+ getRangeStrs(): String[ ]
Tabs
+ basicDataStr: String
+ measurementsStr: String
- tabSelected: String
- basicDataTab: VRectangle
- measurementsTab: VRectangle
- width2: double
- height2: double
+ Tabs(top: double, left: double, height2: double, width2: double)
+ activateBasicDataTab(backgtound: VRectangle, measurements: Composite, basicData: Composite): void
+ activateMeasurementsTab(backgtound: VRectangle, measurements: Composite, basicData: Composite): void
+ getBasicDataTab(): VRectangle
+ getMeasurementsTab(): VRectangle
+ getTabSelected(): String
+ getBasicDataStr(): String
+ getMeasurementsStr(): String
+ setWidth2(width2: double): void
+ setHeight2(height2: double): void
SimbadParallaxFilter
- qsquares: VRectangle[ ]
- parallax: VText
- parallaxStr: String
+ SimbadParallaxFilter(top: double, left: double, right: double)
+ getParallaxStr(): String
SimbadOTypeFilter
- otsquares: VRectangle[ ]
- types: VText[ ]
+ SimbadOTFilter(top: double, left: double, right: double)
+ getOTSelected(): int[ ]
SimbadInfo
- basicData: Composite
- measurements: Composite
-tabs : Tabs
+ SimbadInfo(obj: AstroObject, x: double, y: double, parentWidth: double)
+ basicData(top: double, left: double, obj: AstroObject, info: String[ ]): Composite
+ measurements(top: double, left: double, obj: AstroObject): Composite
- calculateWidth(info: String[ ]): double
- getTabs(): Tabs
- getBasicData(): Composite
- getMeasurements(): Composite
SimbadMFilter
- msplits: VSegment[ ]
- msquares: VRectangle[ ]
+ SimbadMFilter(top: double, left: double, right: double)
+ getMeasurementsSelected(): int[ ]
SimbadSTFilter
- qsquares: VRectangle[ ]
- st: VText
- pec: VText
- lc: VText
- stStr: String
- lcStr: String
- pecStr: String
+ SimbadSTFilter(top: double, left: double, right: double)
+ getSTStr(): String
+ getLCStr(): String
+ getPCStr(): String
SimbadPMFilter
- qsquares: VRectangle[ ]
- ra: VText
- dec: VText
- raStr: String
- decStr: String
+ SimbadPMFilter(top: double, left: double, right: double)
+ getRaStr(): String
+ getDecStr(): String
SimbadRVFilter
- qsquares: VRectangle[ ]
- rv: VText
- z: VText
- cz: VText
- rvStr: String
- zStr: String
- czStr: String
+ SimbadRVFilter(top: double, left: double, right: double)
+ getRVStr(): String
+ getZStr(): String
+ getCZStr(): String
Use
Use
Use
Extends
Extends
Extends
Extends
Extends
Extends
Extends
1
n