

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS



***Paquete de software para el análisis de datos
fotométricos de cúmulos estelares***

T E S I S

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN CIENCIAS COMPUTACIONALES

PRESENTA

JORGE HUMBERTO MACFARLAND VENTURA

ENSENADA, BAJA CALIFORNIA A MARZO DE 2010

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE CIENCIAS

PAQUETE DE SOFTWARE PARA EL ANÁLISIS DE DATOS
FOTOMÉTRICOS DE CÚMULOS ESTELARES

TESIS PROFESIONAL

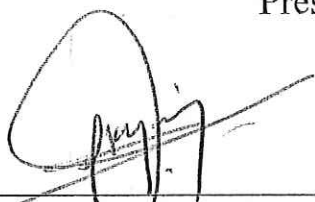
QUE PRESENTA

JORGE HUMBERTO MACFARLAND VENTURA

APROBADO POR:



DR. RAÚL MICHEL MURILLO
Presidente del Jurado



DR. ALBERTO LEOPOLDO
MORAN Y SOLARES
SECRETARIO



MC. ADAN HIRALES
CARBAJAL
IER. VOCAL

Agradecimientos

Le agradezco a Dios por haberme dado la oportunidad de este trabajo y las fuerzas y capacidad para completarlo. También agradezco a mi familia por todo su apoyo; en especial a mi esposa Giselle Martínez Guerrero por su paciencia y comprensión y mis hijos Génesis Giselle Macfarland Martínez y Jorge Isaías Macfarland Martínez que me entretuvieron en los tiempos difíciles haciéndolos pasar con risas y juegos.

Por último gracias a todas las personas que me apoyaron de alguna y otra manera; profesores que tendieron su apoyo y a Raúl Michel Murillo por su paciencia en la entrega del proyecto.

Resumen

Resumen de la tesis de **Jorge Humberto Macfarland Ventura** presentada como requisito parcial para la obtención del grado de **Licenciatura en Ciencias Computacionales**.

Ensenada, Baja California, México. Febrero 2010.

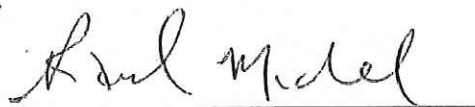
Paquete de software para el análisis de datos fotométricos de cúmulos estelares

Resumen aprobado:

En este proyecto de investigación y desarrollo se implementó un paquete de software para ayudar en el análisis de datos fotométricos de cúmulos estelares los cuales son usados para determinar, de mejor manera, la estructura y evolución de nuestra galaxia.

El paquete quedó completo bajo los requerimientos necesarios para poder ser utilizado en el trabajo de investigación de cientos de cúmulos con información obtenida en San Pedro Mártir con sus telescopios, detectores del tipo CCD y filtros del sistema UBVRI.

El paquete de software fue desarrollado en Java para que pudiera ser utilizado en cualquier sistema operativo y está siendo utilizado actualmente por grupos de investigación, tanto en la UNAM como en Turquía, en el campo de estudio de cúmulos abiertos.



Director DR. RAÚL MICHEL MURILLO

Summary

Thesis summary elaborated by **Jorge Humberto Macfarland Ventura** as partial requirement for the degree of ***Licenciado en Ciencias Computacionales.***

Ensenada, Baja California, México. February 2010.

Software package for the analysis of photometric data on stellar clusters

Approved summary:

In this research and development project a software package was implemented to help in the analysis of photometric data of open clusters that is being used to determine, in a better way, the structure and evolution of our galaxy.

This software package was completed covering the necessary requirements for being used in the research of hundreds of clusters observed at San Pedro Mártir Observatory with its telescopes, CCD detectors and filters in the UBVRI system.

This software package, developed in Java so that it can be run in any operating system, is currently in use by research groups in the field of open clusters both at UNAM and Turkey.



Director DR. RAÚL MICHEL MURILLO

Índice

I. Introducción	
I.1. Cúmulos estelares.	3
I.2. Campaña de observación en San Pedro Mártir.	7
I.3. Biblioteca de visualización de datos.	9
I.4. Objetivo de este trabajo.	9
II. Antecedentes	
II.1. Antecedentes Computacionales.	10
III. Metodología	
III.1. Metodología de la investigación.	12
III.2. Metodología del software.	13
III.3. Análisis.	15
III.4. Diseño.	16
III.5. Desarrollo.	17
III.6. Pruebas.	17
III.7. Mantenimiento y Documentación.	18
IV. Trabajo Realizado	
IV.1. Introducción al problema	
IV.1.1. Presentación del problema.	19
IV.1.2. Análisis del problema.	22
IV.2. Análisis	
IV.2.1. Toma de requerimientos.	23
IV.2.2. Análisis de requerimientos.	23
IV.3. Diseño del sistema	
IV.3.1. Diseño de paquetes.	30
IV.3.2. Diseño de clases.	32
IV.3.3. Diseño de interfaces.	40
IV.3.4. Diseño de algoritmos e implementación.	54
IV.4. Desarrollo	

IV.4.1.	Implementación del paquete de Astronomía.	59
IV.4.2.	Implementación del paquete de Gráficas.	60
IV.4.3.	Implementación del paquete de Interfaz.	60
IV.4.4.	Implementación del paquete de EventosDispositivos.	61
IV.4.5.	Implementación del paquete de TablaInteractiva.	61
IV.4.6.	Implementación del paquete de Archivos.	62
IV.4.7.	Implementación del paquete de Configuración.	63
IV.4.8.	Implementación del paquete de Procesos.	63
IV.5.	Estimación de duración y esfuerzo del desarrollo.	64
V.	Resultados y Discusión	
V.1.	Evaluación del sistema.	65
V.2.	Uso y conclusiones.	66
V.3.	Resultados.	67
V.4.	Trabajo futuro.	68
VI.	Conclusión.	69
VII.	Bibliografía y Referencias.	70
VIII.	Anexos	
VIII.1.	Apéndice A: Gráficas de cúmulos.	72
VIII.2.	Apéndice B: Manual de Usuario.	77
VIII.3.	Apéndice C: Diagramas de clases.	95

I - Introducción

Hoy en día las investigaciones acerca de visualización de datos más reales alrededor del mundo están esperando mayores expectativas, más y más técnicas de software están siendo desarrolladas e importantes grupos de investigación están jugando con interfaces humanas o dispositivos y software para conseguir más flexibilidad en la visualización de datos.

Muchas técnicas y paquetes ya han sido desarrollados pero pocos con la flexibilidad de interacción humana y de los cuales la mayoría están en venta. Tenemos mucha información almacenada en archivos acerca de cúmulos estelares y las técnicas actuales para poder analizarlos son muy complicadas, y por cada archivo toma semanas analizar. Hoy en día usamos la computadora para poder agilizar este tipo de estudios tardados y reducir tanto tiempo y costos.

Sería muy útil una herramienta computacional el cual tomara estos archivos, los analizara y despliegue la información de una manera grafica e interactiva. La información almacenada en estos archivos consiste en una posición de las estrellas y el sistema de colores UBVRI. Que si las pintáramos en un sistema de coordenadas X, Y tuviéramos una imagen del cúmulo estelar tal y como se necesita para ser analizado.

I.1. Cúmulos estelares

Los cúmulos estelares son grupos de estrellas que se formaron a partir de una misma nube de gas y polvo las cuales se mantienen juntas debido a la atracción gravitatoria mutua. Los cúmulos son objetos ideales para la investigación de diversos problemas astrofísicos debido a que sus estrellas se formaron prácticamente al mismo tiempo y bajo las mismas condiciones, además de estar situadas a prácticamente la misma distancia de nosotros y tener un amplio rango de masas que van desde 100 a menos de 0.1 veces la masa del sol. En el cielo se observan dos tipos de cúmulos de estrellas: Los llamados globulares (con forma de globo o

esfera) que contienen cientos de miles de estrellas muy viejas y los llamados abiertos, confinados al plano de la galaxia en sus brazos espirales, que contienen algunos cientos de estrellas jóvenes. Ejemplos de estos cúmulos se presentan en la Figura I.1.A.

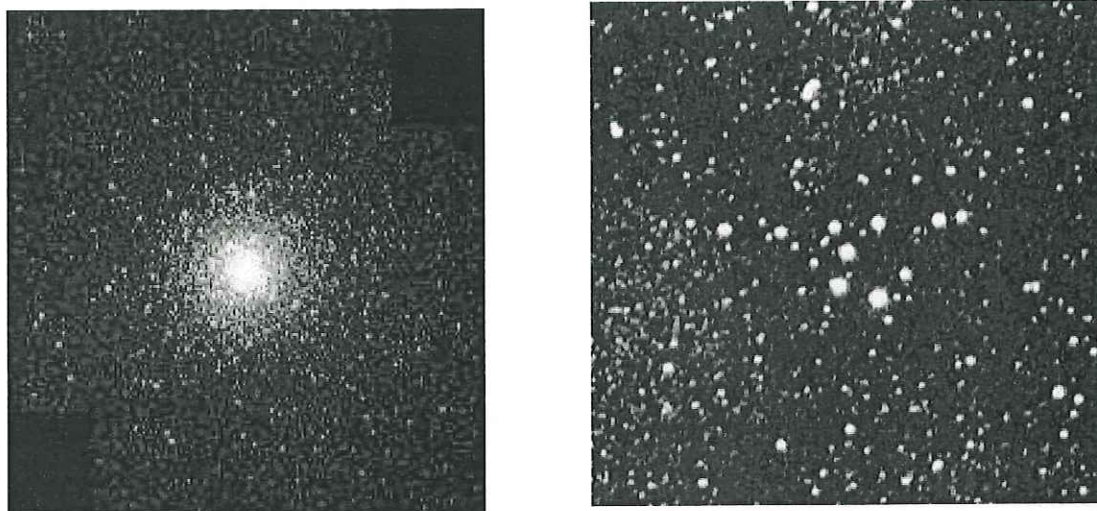


Figura I.1.A.- El cúmulo globular Messier 80 (izquierda) y el cúmulo abierto Collinder 399.

En su conjunto, los cúmulos estelares constituyen sondas excelentes para el estudio de la estructura y evolución de las galaxias ya que sus enrojecimientos, metalicidades, distancias y edades pueden determinarse con gran precisión. En efecto, los cúmulos abiertos jóvenes han sido utilizados repetidas veces como trazadores de brazos espirales (Moffat & Vogt 1973, Feinstein 1994, Becker & Fenka 1970), así como de la estructura tridimensional de la galaxia (Cabrera-Cano et al. 1990, Phelps & Janes 1994, Moitinho 1999). Los cúmulos abiertos más viejos contienen el registro de la composición química de épocas más remotas y han sido utilizados en el análisis de la estructura y evolución química del disco galáctico (Janes 1979, Friel 1995, Piatti et al. 1995, Twarog et al. 1997).

Individualmente, los cúmulos estelares imponen restricciones de observación a las teorías de formación y evolución estelar. La comparación detallada de diagramas fotométricos con las isocronas teóricas (calculadas con modelos de evolución estelar) restringen a los efectos de la composición química (Meynet et al 1993, Kassis et al. 1997). Los diagramas

fotométricos también se hacen particularmente útiles para la búsqueda de estrellas con características peculiares tales como las estrellas tipo Be (estrellas con un disco azul de tipo espectral B que presentan fuertes líneas en emisión) o las rezagadas azules (*blue stragglers*) formadas por la fusión de dos estrellas dando como resultado un objeto más caliente y azul que una estrella de secuencia principal de la misma luminosidad (Mermilliod 1982a, 1982b), así como las estrellas de presecuencia principal (Chavarría 1994, Delgado et al. 1998) o las gigantes rojas (Mermilliod & Mayor 1989).

En lo que respecta a los estudios galácticos, una de las más severas limitaciones en la observación ha sido la ausencia de datos fotométricos. De los ~1600 cúmulos identificados en la galaxia (WEBDA, <http://www.univie.ac.at/webda/navigation.html>) más de la mitad no tiene estudio alguno, excepto por la determinación de sus coordenadas y la estimación burda de su diámetro angular. La muestra de cúmulos que sí tiene algún estudio, ha sido observada utilizando sobre todo las técnicas fotométricas tradicionales (fotometría fotoeléctrica) que, pese a su precisión, están limitadas severamente en profundidad y en el número de estrellas medidas. Como consecuencia, la muestra actual de cúmulos estudiados está fuertemente sesgada en el sentido de una falta de cúmulos débiles lo que se traduce en una escasez de cúmulos lejanos, así como de cúmulos viejos (que son débiles puesto que sus estrellas más brillantes ya completaron sus ciclos de vida) y de cúmulos altamente enrojecidos (débiles por la extinción debida a la presencia de materia interestelar en la línea de visión). También está el problema de la falta de homogeneidad en los métodos de observación y análisis utilizados en los cúmulos estudiados. Todo esto ha dado origen a controversias tales como (a) la existencia, o no, de un gradiente de metalicidad en función del radio galactocéntrico (Janes 1979, Twarog et al 1997, Carraro et al. 1998), (b) la dependencia de la metalicidad con la altura relativa al disco galáctico (Piatti et al. 1995), y (c) la incertidumbre en la caracterización de la estructura galáctica,

A partir de las observaciones con CCD de cúmulos abiertos realizadas desde los años ochenta (Schild 1983, Walker 1985), hasta la fecha se han publicado estudios fotométricos de cerca de 200 cúmulos con la utilización de este tipo de detectores. Sin embargo, gran parte de estas observaciones han sido dedicadas a la obtención de datos más profundos de los cúmulos estudiados anteriormente. Uno de los motivos para la repetición de las observaciones está en la ausencia de criterios confiables para la no selección de los cúmulos que ya se estudiaron previamente y que también está asociada a la laboriosidad de la obtención de fotometría CCD de precisión en campos con estrellas muy cercanas entre sí, como es el caso de los cúmulos. Así, el astrofísico que busca cierta información sobre algún tipo de cúmulo, opta frecuentemente por no observar a ciegas, sino elegir a los objetos en base a criterios de selección preestablecidos en estudios anteriores ó escogidos de compilaciones tales como el catálogo de Lyngå (1987) o la base de datos WEBDA.

El catálogo Lyngå, que ha constituido la base observacional para innumerables estudios astronómicos, incluye distancias para 422 cúmulos y ha hecho posible una serie de estudios que han arrojado conclusiones interesantes sobre el disco galáctico, además de su utilidad en la planificación de observaciones. Sin embargo, este catálogo está construido con parámetros obtenidos por varios autores; con diversas técnicas observacionales, con distintas calibraciones y con diferentes criterios de edad, por lo que resulta poco homogéneo y de interés limitado para estudios que requieren una mayor precisión en la estimación de los parámetros fundamentales. Como ejemplo de la precisión que se puede esperar, debido al efecto de estas inhomogeneidades, nos referimos a Janes & Adler (1982) quienes encuentran que los módulos de la distancia estudiados por dos o más autores tienen una diferencia media de 0.55 mag (que se refiere a la medida del Brillo o magnitud estelar).

El reconocimiento de la importancia de la homogeneidad de las técnicas observacionales y métodos de análisis ha llevado a que Loktin y Matkin (1994), hayan vuelto a analizar la fotometría UBV de 330 cúmulos, lo que resultó en un catálogo de enrojecimientos,

distancias y edades, menos numeroso, pero más homogéneo. Este catálogo ya ha sido utilizado exitosamente en estudios de la estructura galáctica, permitiendo visualizar brazos espirales y corrugaciones del disco galáctico.

La homogeneidad de técnicas observacionales y métodos de análisis es reconocida como crucial. Pero no es el único efecto que controla la construcción de una lista precisa de parámetros fundamentales de cúmulos. La fuerte contaminación por estrellas de campo (enfrente y detrás del cúmulo), los efectos evolutivos, junto con la dificultad de obtener observaciones profundas y precisas con las técnicas fotométricas tradicionales, han llevado a resultados distintos para las distancias y edades de varios cúmulos. La importancia de la profundidad fotométrica (fácilmente obtenida con un CCD) se hace evidente en cúmulos jóvenes, donde las estrellas más masivas empiezan a abandonar la secuencia principal y que, sin una secuencia profunda pueden ser confundidas en un diagrama color-magnitud con estrellas de tipos espectrales más tempranos, conduciendo a la estimación sistemática de distancias menores.

I.2. Campaña de observación en San Pedro Mártir

De junio de 2001 a diciembre de 2003 se llevó a cabo una campaña de observación de cúmulos abiertos en nuestra galaxia en el Observatorio Astronómico Nacional ubicado en la Sierra de San Pedro Mártir, Baja California (OAN-SPM). Este estudio, dirigido por el Dr. André Moutinho, arrojó datos para más de 300 cúmulos observados con las mismas técnicas de observación utilizando el sistema de estrellas estándar de Landolt (1992) y los mismos instrumentos (telescopio, filtros y detector). Las imágenes obtenidas fueron reducidas (procesadas) generando bases de datos, para cada cúmulo, consistentes en las coordenadas y las magnitudes para cada una de las estrellas del campo. Se utilizó la colección de cinco filtros calibrados que permiten el paso de la luz en diferentes colores desde el ultravioleta hasta el infrarrojo llamado el sistema de Johnson-Cousins o $UBV(RI)c$. Estos datos fueron después

complementados con mediciones en el infrarrojo hechas durante la campaña del Two Micron All-Sky Survey (2MASS, <http://www.ipac.caltech.edu/2mass/>). En la Figura I.2.A se presentan las curvas de transmitancia de distinto tipo espectral.

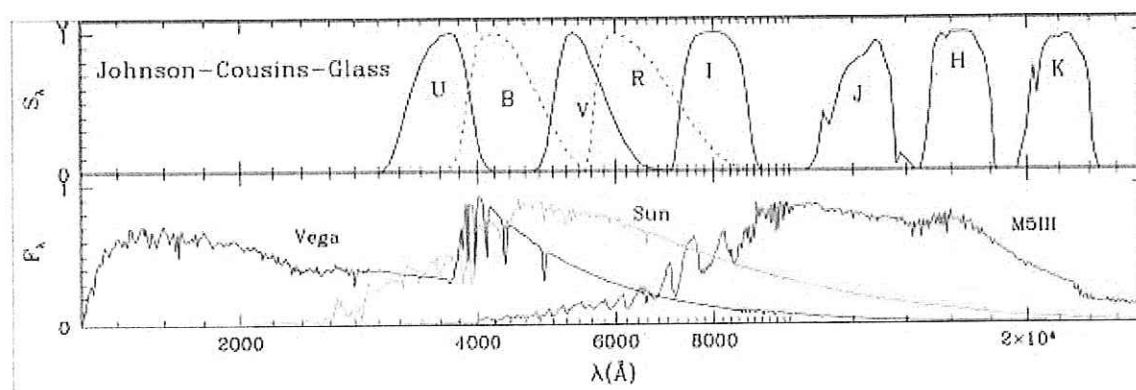


Figura I.2.A.- Transmitancia de los filtros UBVRJ utilizados para las observaciones en SPM y filtros JHKs del 2MASS. Tomada de http://pleiadi.pd.astro.it/isoc_photsys.00/filters1.jpg.

Con este tipo estudio se está utilizando la fotometría UBVRJ para establecer un sistema de referencia fotométrico altamente homogeneizado, para ser utilizado en transformar las observaciones de otros estudios y construir un atlas de diagramas de diagnóstico color-color y color-magnitud que proporcionará criterios de selección de objetos interesantes para su estudio más detallado (ej. cúmulos distantes, viejos, jóvenes, con alto enrojecimientos, con peculiaridades químicas o evolutivas, con estrellas singulares, etc.). Además, esta fotometría permitirá obtener estimaciones de un modo sistemático de enrojecimiento, distancia, edad, dimensión y número de estrellas miembros para todos los cúmulos, por lo que constituirá la base observacional de referencia para estudios galácticos futuros.

I.3. Biblioteca de visualización de datos

Para poder llevar a cabo de manera eficiente el análisis de los datos fotométricos de los cúmulos, se encontró que era necesario el desarrollo de un paquete computacional que permitiera, de manera interactiva, la visualización de los datos fotométricos de manera que, generando distintos diagramas simultáneos, fuera posible identificar a las estrellas que realmente pertenecen al cúmulo y con ellas estimar sus parámetros fundamentales (enrojecimiento promedio, distancia, metalicidad y edad). Este paquete también serviría para crear criterios de selección en la preparación de posteriores observaciones y como punto de partida para profundizar en el conocimiento de estos cúmulos.

I.4. Objetivo de este trabajo

Los objetivos principales de este trabajo de tesis son el desarrollar un paquete de visualización de datos fotométricos y auxiliar en el análisis de datos fotométricos de cúmulos estelares para, inicialmente procesar una muestra de más de 300 cúmulos abiertos (30,000 o más estrellas individuales) observados en el OAN-SPM. De acuerdo con los resultados de una revisión de literatura, se considera que este programa será el primer paquete especializado que se desarrolle para realizar esta tarea.

Algunas de las características principales obtenidas en las primeras reuniones del proyecto es que tiene que poder aceptar archivos como entrada y poder sacar graficas con diferentes coordenadas especificadas por el usuario y que sea suficientemente interactivo para poder rastrear estrellas con mas facilidad y determinar ciertas características o incluso si es parte o no del cúmulo en cuestión.

II - Antecedentes

II.1. Antecedentes Computacionales

Antes de comenzar el trabajo formal de esta tesis, se hizo una búsqueda bibliográfica, básicamente en Internet, de sistemas que pudieran servir para llevar a cabo las características antes mencionadas. Aun que no se descarta la posibilidad de que haya más, sólo se encontraron dos sistemas que podrían ser parcialmente útiles para este propósito. Estos sistemas fueron:

1. **HR Trace** (<http://xoomer.alice.it/hrtrace/index.htm>) es un software para el análisis de cúmulos estelares que permite realizar varias de las funciones anteriormente descritas pero, sin embargo, sólo corre en Windows y está basado en rutinas de Excel. Un ejemplo de las ventanas de este programa se presenta en la Figura II.1.
2. **Open Cluster Workshop** (<http://www.sao.ac.za/assa/html/ocw.html>), es un software para el análisis de campos estelares con el propósito de detectar cúmulos estelares. Este programa, que está siendo desarrollado por la Astronomical Society of Southern Africa, está hecho en lenguaje Delphi4 y sólo corre bajo el ambiente Windows. Sin embargo no es de distribución libre. En la Figura II.1.A se presenta una ventana de este programa.

Estos dos programas son relativamente nuevos pero no son muy accesibles ni fáciles de usar, mucho menos cuentan con una interfaz interactiva. Otras herramientas como generadores de gráficas o minería de datos pueden ser útiles para el mismo propósito mas no cubren totalmente las necesidades.

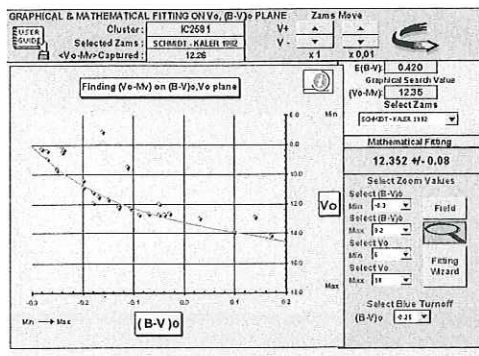


Figura II.1.A.- Vistas de los programas HR Trace (izquierda) y Open Cluster Workshop.

Como generadores de gráficas se encontraron varios paquetes unos muy buenos pero demasiado complejos, otros escuetos y carentes de interactividad. Se tomo como referencia algunas características de estos generadores de gráficas como base para crear un mejor diseño de un graficador interactivo y editable. Por mencionar algunos de estos generadores de gráficas:

- jfreechart-1.0.1 (<http://www.jfree.org/jfreechart/>)
- pplot-5.5.3 (<http://www.ugcs.caltech.edu/manuals/math/pplot-5.5.3/index.html>)
- jPlot-1.0.2 (<http://masaka.cs.ohiou.edu/software/tcptrace/jPlot/>)
- Plot Package (<http://www.exyonline.com/>)
- SimplePlot (<http://javaboutique.internet.com/SimplePlot/>)

III - Metodología

La metodología utilizada para el desarrollo de este trabajo se dividió en dos partes. La primera parte se refiere a la metodología utilizada en el proceso general de investigación, y la segunda se refiere más específicamente al proceso de desarrollo de software.

III.1 - Metodología de la investigación

A lo largo de varias campañas de observación con el telescopio de 0.84m del OAN-SPM se han obtenido imágenes CCD en las bandas UBVRI de muchos cúmulos abiertos visibles desde el OAN-SPM, así como de estrellas estándar de Landolt (1992), e imágenes de calibración (bias, campos planos, de obturador). Se ha tratado de seguir rutinas bien establecida que facilitarán el procesamiento subsecuente de los datos.

Las correcciones de bias, campos planos y rayos cósmicos se realizarán siguiendo los procedimientos usuales dentro del paquete CCDRED de IRAF prestando especial atención al efecto en los datos del tiempo finito de apertura, y cierre, del obturador.

Las reducciones fotométricas se realizaron con el paquete IRAF/DAOPHOT. En la mayoría de los casos y dadas las dimensiones de la muestra, se realizará fotometría de apertura.

La fotometría se realizará mediante la técnica de la PSF (Point Spread Function). PSF Es una función matemática que describe la forma que tienen las estrellas en las imágenes producidas por el detector. Como las estrellas están tan lejos, idealmente debería ser puntual pero debido a los efectos de la atmosfera y la óptica del telescopio, la luz se desparrama. Este

tipo de fotometría es muy laboriosa debido a que contrariamente a la fotometría de apertura que se basa en el conteo de fotones, ésta se basa en el ajuste simultáneo de perfiles de luminosidad de grupos de estrellas lo que permite obtener magnitudes más precisas en campos estelares densamente poblados.

La fotometría de las distintas bandas es reunida en un catálogo que se utiliza para construir un atlas de diagramas fotométricos de los cúmulos observados.

Los enrojecimientos, distancias y edades de los cúmulos se determinarán siguiendo el método de ajustes de la secuencia principal de edad cero y de isócronas calculadas a partir por los modelos de evolución estelar actuales en los diagramas fotométricos. En los casos en que se presente una extinción intra-cúmulo variable se procede a un desenrojecimiento estelar por estrella siguiendo técnicas similares al método 'Q' de Johnson & Morgan (1953).

III.2 - Metodología del software

Para el desarrollo de este sistema se utilizó el modelo de desarrollo en cascada con retroalimentación (Sommerville 2001). Este es un modelo común y fácil de implementar. El modelo toma como procesos fundamentales las actividades de especificación, validación y evolución y los representa como diferentes fases de procesos (Figura III.2.A). Estos procesos incluyen la especificación de requerimientos o análisis, diseño de software, implementación, pruebas, operación y mantenimiento.

En la etapa de análisis se obtuvieron los requerimientos y especificaciones del sistema. En la etapa de diseño se diseñó el sistema de forma modular tratando de desarrollar cada uno de estos módulos por separado. En la fase final de la etapa de desarrollo se integraron los

módulos en un sistema. Por último se llevó a cabo una etapa de pruebas donde se pudo validar la confiabilidad e interactividad del sistema, terminando con la operación y mantenimiento del mismo.

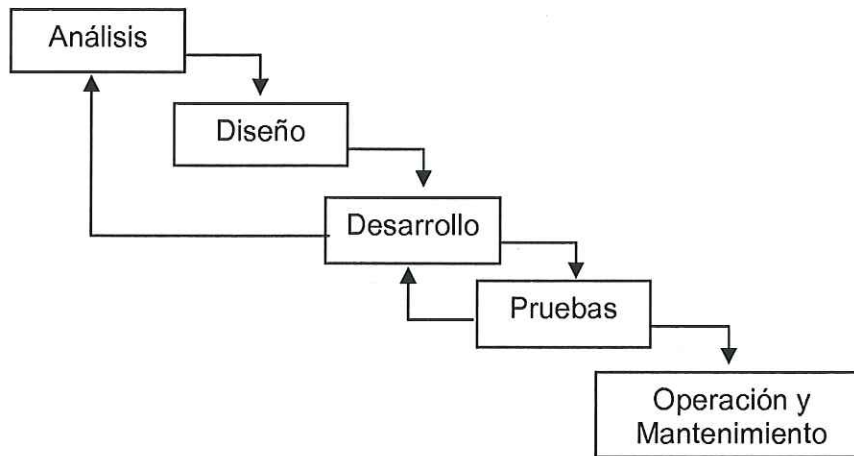


Figura III.2.A. Modelo de cascada con retroalimentación para el desarrollo de software (Sommerville 2001).

Además, dada la variabilidad de los requerimientos y las limitantes de tiempo, el modelo en cascada tuvo 3 iteraciones principales (Figura III.2.B), realizando ciclos relativamente rápidos de análisis, diseño e implementación para desarrollar funcionalidad específica del sistema en tiempos cortos. Esto dio como resultado un modelo de versiones sucesivas (Sommerville 2001) con 3 iteraciones.

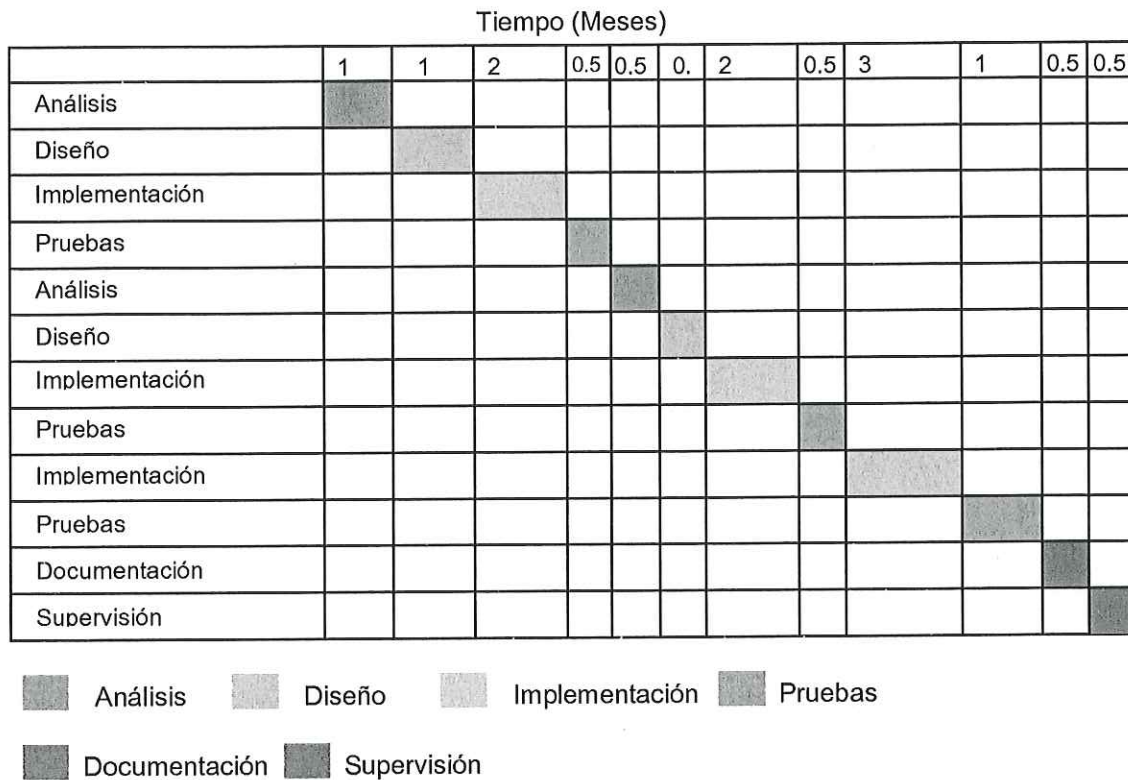


Figura III.2.B. Versiones Sucesivas con 3 iteraciones en Diagrama de Gantt

El diagrama de Gantt, gráfica de Gantt o carta Gantt es una popular herramienta gráfica cuyo objetivo es mostrar el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. Y es por esto que se utilizó este diagrama para poder visualizar mejor las iteraciones.

III.3 - Análisis

Durante las etapas de análisis se identificaron los requerimientos para determinar la posible funcionalidad del sistema con base en las necesidades del cliente/usuario. Los requerimientos se documentaron a través de especificaciones de funcionalidad, las cuales fueron informadas y aprobadas por el cliente en cada iteración. Como se mencionó

anteriormente, el subproceso de especificación de requerimientos se realizó en 2 etapas (2 primeras iteraciones), ya que conforme se avanzaba en el desarrollo surgían nuevos requerimientos. Aunque en general el proceso de desarrollo en cascada recomienda que se determinen todos los requerimientos antes de continuar con la etapa de diseño, en esta ocasión el proceso de desarrollo estuvo asociado a un proceso de investigación con una naturaleza “open-ended”, lo cual dificultó que todos los requerimientos se pudieran determinar desde el inicio. Así, cada que se generaban nuevos requerimientos, el documento de diseño se modificaba para mantener una versión actualizada del mismo.

III.4 - Diseño

En esta etapa se tomaron decisiones de cómo se debía realizar la funcionalidad especificada en la etapa anterior. Estas decisiones se reflejaron en un diseño de alto nivel (arquitectura) y en un diseño de bajo nivel. Dada la variabilidad de los requerimientos, este proceso se realizó de manera ágil e interactiva con el proceso de implementación. Por esta razón mientras se diseñaba las estructuras de alto y bajo nivel, se fueron las clases que serían necesarias para agregar cierta funcionalidad. Conforme se avanzaba en el diseño de cada requerimiento se iban modificando las clases e incluso generando nuevas. Se generaron diagramas de paquetes, luego diagramas de caso de uso, después se generaron diagramas de clases, seguido de diagramas de secuencia y para finalizar un diagrama de estructura general donde se veía claramente la comunicación que hay entre todas las clases y como funciona el sistema (arquitectura).

III.5 - Desarrollo

Como se mencionó anteriormente, dado que esta etapa se llevó a la par de la etapa de diseño, fue necesario realizar 3 iteraciones con retroalimentación, con lo cual se proponía un diseño, se implementaba este diseño y se retroalimentaba el diseño para generar una nueva implementación. Durante el proceso de desarrollo se vio una necesidad de controlar mejor el proceso general, herramientas que en la actualidad se utilizan para llevar control del proceso de desarrollo y que llevan registro de cada requerimiento, su proceso y los errores que van apareciendo.

En esta etapa se desarrollaron las diferentes librerías que conforman en su totalidad el sistema, el lenguaje de programación que se utilizó fue Java por su capacidad de ser independiente de plataforma y por la facilidad del lenguaje entre otras características, y el software que se utilizó para desarrollar gran parte de la interfaz fue NetBeans que es un Ambiente de Desarrollo Integrado o IDE (Integrated Development Environment) para Java.

III.6 - Pruebas

Se desarrollaron diversos casos de pruebas con base en los casos de uso de la etapa de diseño, tomando en cuenta los requerimientos especificados y generando un reporte de resultados, errores, fallas de funcionalidad, etc. El control de pruebas se realizó de manera manual, con pruebas recursivas.

Las pruebas visuales fueron más complejas y extensas que las de funcionalidad esto debido a la interactividad agregada.

III.7 - Mantenimiento y Documentación

Dado que la fase de mantenimiento quedó fuera del alcance de este documento de tesis, esta sección se centrará en el aspecto de la documentación.

Se generaron tres documentos principales; 1) *Documento del sistema* que trata de la estructura, como se le pueden hacer futuras modificaciones, y sobre el funcionamiento general. Consiste en un conjunto de documentos de diseño como diagramas de casos de uso, diagramas de clases (*ver Diseño de clases*), e interfaces, entre otros. 2) *Manual técnico* que trata sobre el código, diagramas de clases detallado, estructura interna y otros detalles arquitectónicos, y se encuentra en formato de documentación de Java, y 3) *Manual de usuario* (*ver Apéndice B*) que explica el funcionamiento del sistema, atajos, como lograr el mejor uso del sistema, entre otros detalles. El sistema se encuentra en uso por parte de varios astrónomos de la UNAM, quienes lo utilizan de manera cotidiana para realizar sus investigaciones. Estos usuarios llevan un control de las fallas que pudieran presentarse, con miras a poder dar mantenimiento de una fase posterior.

IV - Trabajo realizado

IV.1 - Introducción al problema

IV.1.1 - Presentación del problema

El Problema

Actualmente, el sistema con el que los astrónomos del IA-UNAM los cúmulos estelares no es interactivo ni cuenta con una automatización de los procesos involucrados. Dado que los cúmulos pueden constar desde más de pocas decenas hasta unos miles de estrellas, esto hace muy difícil el poder seguir una estrella en 3 o más gráficas, siendo esto también un consumidor de tiempo y de recursos. Además, no se cuenta con un método o sistema interactivo para poder visualizar diversas graficas de un cúmulo (por ejemplo, hasta 16), compararlas y poder determinar si ciertas estrellas pertenecen o no al cúmulo estudiado.

Un problema adicional se refiere a que el sistema no podría estar implementado en Windows debido al alto costo de las licencias de los lenguajes y entornos de desarrollo para Windows y del mismo sistema operativo Windows. Por esta razón se requiere que el sistema sea desarrollado y opere sobre el sistema operativo Linux, el cual es usado en la UNAM.

El Objetivo Astronómico

El objetivo principal de este proyecto es obtener, de una forma sistemática, un conjunto de datos CCD en el sistema fotométrico UBVRI para una muestra extensa de cúmulos abiertos prácticamente desconocidos o con datos extremadamente escuetos.

Este objetivo está ampliamente justificado cuando se tienen en cuenta los fuertes efectos de selección existentes en la muestra actual de cúmulos estudiados. En particular, la dificultad de observar cúmulos débiles utilizando las tradicionales técnicas de observación fotoeléctrica ha llevado a que el número de cúmulos lejanos, viejos o altamente enrojecidos estudiados sea muy escaso. Si se añade la dificultad de las observaciones fotoeléctricas en los campos más poblados de la vía láctea, fácilmente se entiende el porqué solo hay estudios para aproximadamente la mitad de los cerca de 1600 cúmulos de la galaxia. Además, los estudios existentes han sido realizados por distintos autores utilizando distintas técnicas y métodos de análisis, por lo que las conclusiones alcanzadas carecen de homogeneidad en su conjunto.

Esta situación es bien conocida por los astrónomos, y constituye una severa limitación para una diversidad de estudios que van desde la estructura galáctica hasta a la evolución estelar. La reciente aparición de los CCDs en la astronomía observacional ha proporcionado la oportunidad de superar las dificultades encontradas por los métodos tradicionales, sin embargo las reducciones exigen un largo proceso de aprendizaje, son lentas y consumen recursos computacionales considerables, lo que ha frenado la expansión de la muestra de cúmulos estudiados.

Con este proyecto pretendemos llenar el vacío observacional existente en el acervo astronómico aprovechando la infraestructura existente en el OAN, nuestra experiencia en reducciones de datos obtenidos con CCD, nuestro conocimiento en métodos de análisis fotométrico, así como los recursos de cómputo del grupo de investigadores participantes.

Así, se pretende utilizar la fotometría UBVRI obtenida para establecer un sistema de referencia fotométrico homogéneo, para ser utilizado en transformar las observaciones de otros estudios. Se pretende construir un atlas de diagramas de diagnóstico color-color y color-magnitud. Este atlas proporcionará criterios de selección de objetos para su estudio (ej. cúmulos distantes, viejos, jóvenes, con alto enrojecimientos, con peculiaridades químicas o evolutivas, con estrellas singulares, etc). Además, esta fotometría permitirá obtener

estimaciones de un modo sistemático de enrojecimiento, distancia, edad, dimensión y número de estrellas miembros para todos los cúmulos, por lo que constituirá la base observacional de referencia para estudios galácticos futuros.

Ciertos cúmulos que, por presentar características notables (ej. estrellas de presecuencia principal, enrojecimiento intercúmulo variable o anómalo, estrellas Be, etc) pueden requerir observaciones espectroscópicas posteriores. En esos casos, las observaciones se facilitarán de gran manera con una pre-selección fotométrica de los objetos de interés basada en la base de datos y atlas fotométrico que nos proponemos generar. Uno de los atractivos de este proyecto es que se puede realizar con un telescopio de pequeñas dimensiones como lo de 0.84 cm del Observatorio Astronómico Nacional de San Pedro Mártir (OAN-SPM), poniendo a este instrumento actualmente muy subutilizado a generar ciencia de punta. Además, como producto secundario los datos recogidos permitirán realizar una caracterización del sistema instrumental y determinar su relación con el sistema estándar de Johnson/Cousins.

Otro de los objetivos de este programa es el establecer un grupo de investigación activo, competitivo y generador de conocimiento de punta en los campos de la estructura y desarrollo de la galaxia, así como de la formación y evolución estelar. Este grupo combinará la experiencia y amplitud de visión de los participantes *seniores* en ambos campos con el dominio de las nuevas técnicas de observación, reducción y la experiencia en trabajo con cúmulos estelares del responsable de este proyecto.

Es importante remarcar la importancia del desarrollo de software, ya que sin este, no se pudieran estudiar a fondo, en el tiempo necesario, los datos CCD en el sistema fotométrico UBVRI que se desean obtener. Otro punto importante es el poder realizar una pre-selección fotométrica a través de la funcionalidad interactiva del sistema de software que se propone desarrollar y poder generar más imágenes para el atlas de diagramas de diagnóstico color-color

y color-magnitud, y usar este sistema de software para poder también tener un sistema de referencia fotométrico altamente homogeneizado.

El Objetivo Computacional

Crear un sistema interactivo y gráfico para el análisis fotométrico de cúmulos estelares, que cuente con la posibilidad de mostrar diferentes tipos de diagramas fotométricos para el estudio de estos cúmulos estelares de una manera automatizada, interactiva, sencilla y amigable y que reduzca de manera significativa el tiempo y recursos usados para el estudio de dichos cúmulos y avanzar en el estudio y realización de un atlas gráfico de cúmulos y sus características que incrementen el acervo cultural astronómico.

IV.1.2 - Análisis del problema

Primero se vio como prioridad establecer los requerimientos basados en el problema. De esto salió el más importante de todos, el sistema operativo y la base en la que debía implementarse. El sistema operativo usando en la UNAM es Linux, por lo que el sistema debía ser implementado para este sistema operativo. Sin embargo, de aquí surge otra pregunta de implementación, aplicación Web o de escritorio?. La decisión se tomó con base en lo siguiente: 1) La aplicación debe de manejar gráficos para cual es más fácil y mejor en escritorio, 2) El Web no tiene muchos lenguajes que manejen gráficas, pudiera ser un Applet de Java pero dada sus limitaciones, solo se usan para demostraciones y 3) El hecho de requerir tanta información a la mano serían demasiadas peticiones al servidor. Por otro lado se tenía Ajax, pero también iba a reflejarse que, tener más de 2 gráficas con más de 600 estrellas, incrementaría el peso y requiriendo demasiado ancho de banda para la comunicación con el servidor.

Por esto para empezar, se decidió optar por Linux en escritorio, tomando después la siguiente cuestión del lenguaje de implementación, entre los que estaban, java, tcl, octave y python.

Finalmente, se decide por utilizar Java. Los criterios considerados para tomar esta decisión incluyen requerimientos de interacción del usuario, el manejo de eventos y gráficos, el uso de funciones matemáticas, además de que es un lenguaje que no se tenía que estudiar por que ya se sabía programar en él, no se necesita pagar por la licencia y es multiplataforma por lo cual se podía programar en Linux y se pueden desarrollar aplicaciones para escritorio.

IV.2 - Análisis

IV.2.1 - Toma de requerimientos

Los requerimientos como mencionado anteriormente (III.3) se tomaron en varias etapas. La primera fue más sencilla que la subsecuente. En la primera se establecieron los requerimientos básicos, aquí surge la necesidad de que sea implementado en Linux, la interactividad como la parte más importante del sistema ya que hará más fácil el proceso de análisis de cúmulos estelares, algunas operaciones como poder seleccionar estrellas, etc. A continuación se muestran todos los requerimientos del sistema, los requerimientos reunidos de las dos etapas.

Requerimientos funcionales

- REQ_F_001: El sistema debe leer desde un archivo con formato ASCII, texto con formato estructurado con una línea inicial para las etiquetas empezando con las coordenadas X, Y, magnitudes y sus respectivos errores.

- REQ_F_002: Cuando se abra un archivo el usuario debe de poder ver el primer diagrama que es la graficación de la imagen, de la cual se tomaron los datos, es decir la fotografía del cúmulo tal y como se obtiene del telescopio y la cual se obtiene por medio de la representación de los valores X, Y de la estructura leída del archivo.
- REQ_F_003: El usuario tendrá la opción de poder elegir la combinación de diagramas que el desee.
- REQ_F_004: El sistema debe de permitir el uso del ratón para poder interactuar con las estrellas de un cúmulo, pudiendo elegir una estrella dentro de un diagrama y que esta misma estrella activa pueda verse con claridad en los demás diagramas generados.
- REQ_F_005: El usuario tiene que aplastar alguna tecla para poder aceptar la estrella activa y pueda marcarse como seleccionada.
- REQ_F_006: Una estrella eliminada no puede ser seleccionada, una estrella eliminada es automáticamente deseleccionada.
- REQ_F_007: Es de importancia que el sistema también permita la selección de varias estrellas por medio de selección de regiones, las cuales son creadas por el usuario usando el ratón y teniendo estas una forma poligonal (de la forma que el usuario desee).
- REQ_F_008: El sistema debe de presentar la opción de poder grabar a un archivo con formato ASCII, las estrellas elegidas por el usuario como parte del cúmulo, con el nombre que el usuario desee y de otra forma grabar el archivo con el que se está trabajando.
- REQ_F_009: El sistema debe de proporcionar al usuario información extra de los diagramas que ayude a la visualización y entendimiento de los cúmulos y de las estrellas que se eliminan, como las que se eligen como parte del cúmulo y así tener una referencia con lo que se esta trabajando. Ver REQ_INTFZ_003.
- REQ_F_010: El sistema debe contar con una funcionalidad para poder recuperar información dejada atrás del proceso (undo), es decir que tenga la posibilidad de dejar al cúmulo como estaba antes de la última operación.

- REQ_F_011 El sistema debe de poder usar un método de acercamiento para cada gráfica (zoom), esto es un acercamiento de cierta región del cúmulo con el que se trabaja.
- REQ_F_012: El sistema deberá poder facilitar al usuario la visualización de las estrellas seleccionadas o activas.
- REQ_F_013: El usuario debe de poder elegir en la configuración el tipo y tamaño de letra que el desee tener en el sistema.
- REQ_F_014: Se debe poder graficar el vector de enrojecimiento del cúmulo para cada gráfica.
- REQ_F_015: Se deberán poder guardar las gráficas a imágenes para uso futuro.

Requerimientos no funcionales

- REQ_NF_001: El sistema debe ser implementado en el Sistema Operativo Linux.
- REQ_NF_002: El sistema no debe de estarse recompilando para poder ajustar alguna variable o para modificar algún valor que el usuario necesite.
- REQ_NF_003: El sistema debe ser interactivo y amigable para el usuario.
- REQ_NF_004: Las etiquetas deben de ser descriptivas y no deben confundir al usuario.

Requerimientos Extras:

- REQ_EX_001: Se deberá tener un archivo de configuración para el sistema, para que cada usuario pueda definir sus preferencias.
- REQ_EX_002: El sistema debe de ser implementado de tal forma que se le puedan hacer futuras modificaciones.
- REQ_EX_003: Los colores de banda no deben de ser los únicos colores, el sistema debe de aceptar expresiones con los colores para poder crear nuevos colores, ej: $Q=(U-B) - .72*(B-$

V) y estas expresiones pueden contener sumas, restas, multiplicaciones, divisiones, colores, magnitudes y errores.

Requerimientos de Interfaz:

- REQ_INTFZ_001: El sistema debe permitir una interacción del usuario con el cúmulo en cuestión.
- REQ_INTFZ_002: El sistema debe de poder desplegar varias gráficas definidas por el usuario con la forma de color-color o color-magnitud y deben de ser como mínimo una gráfica y como máximo 320 gráficas, separadas por pestañas navegables donde el usuario puede tener 4 x 4 gráficas en cada pestaña y un máximo de 20 pestañas, donde todas las pestañas corresponden al mismo cúmulo.
- REQ_INTFZ_003: El sistema debe desplegar la información acerca del cúmulo y sus estrellas en forma que pueda ser legible y entendible para el usuario y que pueda servir de referencia para el estudio que se esté realizando.

Ahora listamos los requerimientos tal y como fueron dados.

1ra Etapa:

REQ_F_001, REQ_F_002, REQ_F_003, REQ_F_004, REQ_F_005, REQ_F_006,
REQ_F_008, REQ_F_009, REQ_NF_001, REQ_NF_002, REQ_NF_003,
REQ_EX_002, REQ_INTFZ_001, REQ_INTFZ_002, REQ_INTFZ_003

2da Etapa:

REQ_F_007, REQ_F_010, REQ_F_011, REQ_F_012, REQ_F_013, REQ_F_014,
REQ_F_015, REQ_NF_004, REQ_EX_001, REQ_EX_003

Como se puede observar, en la segunda etapa se incluyeron menos requerimientos, pero se les puede considerar del mismo o mayor peso debido a que los nuevos requerimientos necesitaban de tiempo para la implementación, además implicó cambios en lo que ya se tenía, lo cual ocasionó un atraso muy fuerte. Sin embargo, ésta es una experiencia que se encuentra a diario en la vida real del desarrollo de software. Muchas veces el cliente no sabe con exactitud lo que quiere o a veces hay varios clientes y no se ven todos a la vez, lo cual obviamente provoca que con la diferencia de ideas, se den requerimientos diferentes o incluso contradictorios.

IV.2.2 - Análisis de requerimientos

Teniendo en mente el sistema operativo Linux, que la aplicación será para escritorio, el lenguaje Java para la implementación y teniendo los requerimientos en mano; el siguiente paso fue llevar estos requerimientos a un modelo que sea entendible y pueda ser implementado. Debido a que Java solo maneja clases era necesario entrar al mundo de la programación orientada a objetos y tratar de ver los requerimientos de tal forma que pudieran considerarse objetos. Primero se pensó en lo más obvio y básico, lo que se trata de estudiar es un cúmulo, que es un conjunto de estrellas y las estrellas tienen ciertas características como lo tienen los cúmulos, así se crearon las 2 primeras clases, **Estrellas** que guarda todas las características necesarias de una estrella en particular y **Cumulo** que es un conjunto de estrellas con un nombre en particular.

Los requerimientos se fueron analizando de la siguiente manera:

- REQ_NF_001, que ya estaba visto, la implementación en Linux.

- REQ_NF_002, con Java se puede usar un script para recompilar y ejecutar sin tener que recompilar y ejecutar manualmente en caso de que haya cambios al código. En este caso no se requiere recompilación ya que el sistema no estará en constante cambio y es mas fácil correr un ejecutable así que se compiló en un JAR.
- REQ_NF_003, este requerimiento se cumple si se llevan a cabo los otros requerimientos sobre la interactividad y la descripción de las cosas en el sistema.
- REQ_NF_004, todas las etiquetas del sistema se trataron de hacer un poco más entendibles para que cualquier usuarios las pudiera leer y entender.
- REQ_F_008, después se creo una clase Escritor que guardara un objeto de tipo Cúmulo a un archivo de texto.

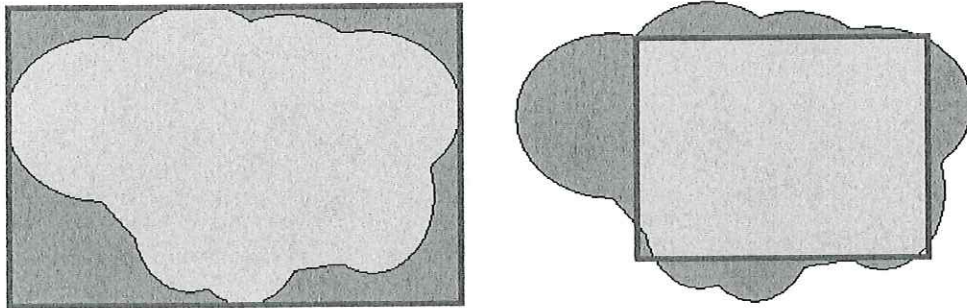
Los requerimientos no funcionales fueron de mayor prioridad debido a que limitan lo que el sistema va a hacer y no se quiere crear una funcionalidad que luego será limitada por un requerimiento no funcional, por esto deben de llevarse siempre en mente.

- REQ_F_001, primero se implementó una clase Lector para poder leer un archivo de texto con el formato especificado en el requerimiento, después se modificó para en vez de imprimir lo que iba leyendo regresara un objeto de tipo Cúmulo.
- REQ_F_002, después de poder leer al archivo se diseñaron 2 clases, Limites y Espacio que serían las clases para graficación de cúmulos.
- REQ_F_004, para cumplir con éste, se creó una clase que se podía pegar a otra escuchando sus eventos de ratón y tratando de analizar lo que sucedía. Con esto se pudo empezar a colorear un punto al tratar de seleccionarlo con el cursor, haciendo clic.
- REQ_INTFZ_002, una vez obtenido lo demás, se creo la pantalla principal que contendría un JTabbedPane de Java para poder separar en pestañas la información y dentro de cada

pestaña se creaba un JPanel que contendría de 1 a 16 objetos de tipo Espacio acomodados con una vista de tipo rejilla.

Esta primera parte del trabajo consistió en desarrollar prototipos de cómo iban a ser las diferentes partes del sistema que cubrirían todos los requerimientos identificados, ya que no se quería exceder la funcionalidad de lo especificado. Esto debido a que el desarrollo de software puede ser un problema que se parece a una masa sin forma, para poder resolverlo computacionalmente es necesario desarrollar un sistema que lo abarque todo pero sin exceder (Figura IV.2.A). Esto también se le llama programación modular (Programación por módulos, que en el caso de Java sería por paquetes).

Mala implementación de la solución



Buena implementación de la solución

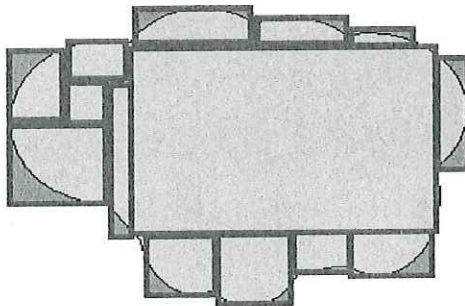


Figura IV.2.A: Diferencias entre la buena implementación por módulos o paquetes y una implementación no modular, donde lo gris es el Problema, los cuadros azules la Implementación y lo rojo la parte que no es del problema que se consideró o la parte que es del problema y no se considero.

IV.3 - Diseño del sistema

Con base en el conjunto de requerimientos identificados en la etapa anterior, se procedió a realizar una propuesta de diseño de alto nivel para el sistema. El sistema de visualización comprende un conjunto de clases organizadas en paquetes, de acuerdo con las recomendaciones de Java. Los paquetes principales se presentan y describen a continuación.

IV.3.1 - Diseño de paquetes

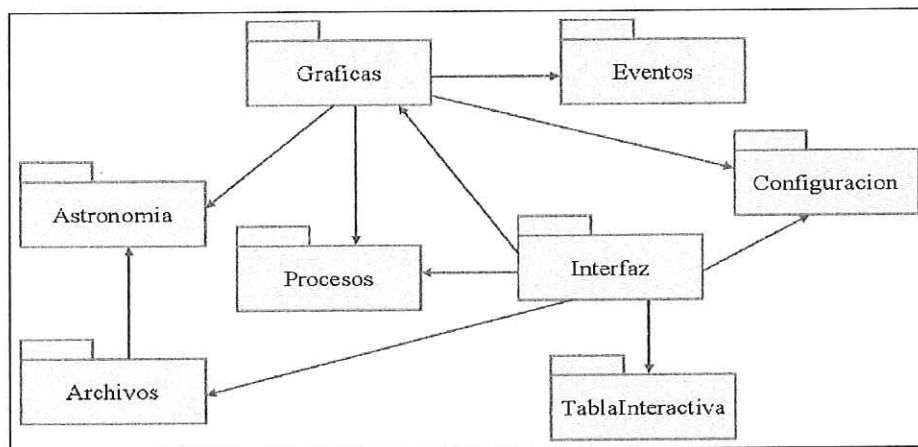


Figura IV.3.A: Diagrama de paquetes o módulos del sistema.

Interfaz: Este paquete fue creado con el propósito de que se encargara de la interfaz (forma) principal que funciona como la entrada del sistema y como un contenedor de formas, las pestañas, la distribución de las gráficas y la vista de la tabla de datos.

Archivos: Este paquete es el que se encarga de los archivos, cuenta con su propia interfaz para seleccionar, grabar y leer archivos de cúmulos y de configuraciones, guardar gráficas y otros detalles como los iconos y filtros de archivos.

Configuracion: Este paquete está encargado de la parte de la configuración, tiene su propia interfaz de presentación y una clase para guardar la configuración durante el uso del sistema.

Procesos: Este paquete es un conjunto de utilerías para el sistema, contiene funcionalidad para evaluar las expresiones matemáticas, guardar de imágenes con su propia interfaz y una clase para hacer y deshacer los cambios hechos a un cúmulo.

Astronomia: Este paquete sólo se encargar del área astronómica de este sistema, el manejo de cúmulos por medio de un vector de estrellas.

TablaInteractiva: Este paquete se encarga de presentar una tabla de datos (donde cada renglón es una estrella y las últimas 2 columnas son opciones para seleccionar y eliminar), que cumple ciertos requerimientos y con un ordenador de datos para la comodidad del usuario.

Eventos: Este paquete se encarga del manejo de todo tipo de eventos. Escucha y distingue eventos de un componente, escucha eventos del ratón y del teclado, y avisa al componente padre que lo inicio.

Graficas: Este paquete presenta la información de un cúmulo en gráficas dependiendo de la configuración que se le dé y dibuja en el panel padre que lo inició el cúmulo, una región poligonal y el objeto zoom.

El paquete **Interfaz** necesita de **Archivos** para cargar los archivos necesarios elegidos por el usuario (cúmulos, configuración) o para guardarlos (cúmulo cambiado, configuración cambiada, gráfica del cúmulo). También necesita de **TablaInteractiva** para mostrar el cúmulo en forma de una tabla de datos. **Configuración** es usado por **Interfaz** para cambiar ciertas características del sistema como colores de ambiente, el tipo de gráficas que se desplegaran en una cierta pestaña, etc. También usa **Procesos** para poder hacer y deshacer cambios hechos a un cúmulo o regresar al sistema a su estado anterior y por último usa **Graficas** que son desplegadas en una tabla, que se va acomodando conforme se agregan más gráficas.

El paquete **Astronomia** es utilizado en conjunto con otros. Es usado en **Eventos** para reportar el cambio en un cúmulo mediante otro cúmulo, **Graficas** dibuja cúmulos, **TablaInteractiva** pone los datos de un cúmulo (renglón por estrella del cúmulo), **Archivos** usa

cúmulos por que necesita leer sus valores para poder salvarlos o al leer un archivo poder crear un cúmulo y por último **Interfaz** usa **Astronomia** por que ocupa guardar el cúmulo abierto de la instancia actual del sistema. Por otro lado **Astronomia** usa procesos para poder calcular el valor de un color definido por el usuario donde se pueden usar varias expresiones matemáticas y usa **Configuracion** por que el cúmulo ocupa saber como va imprimir la gráfica dependiendo en la posición que le tocó dibujar.

IV.3.2 - Diseño de clases

Dentro de cada paquete existen una serie de clases que a su vez se comunican entre si. Cada paquete tiene señalada la clase de entrada por una flecha azul y una relación entre clases que en su totalidad forman la función objetivo de cada paquete.

Paquete Archivos

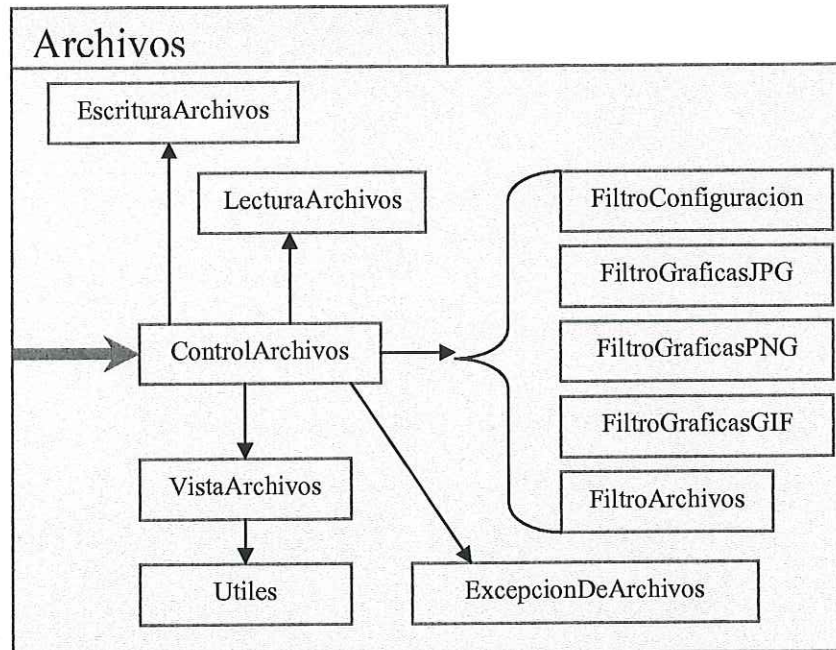


Figura IV.3.B: Diagrama de clases del paquete **Archivos**.

VistaArchivos necesita de **Utiles** para realizar ciertas operaciones en la vista cuando se presenta el diálogo. **ControlArchivos** usa **EscrituraArchivos** para escribir archivos en la memoria, ocupa **LecturaArchivos** para leer archivos de la memoria. También usa **FiltroArchivos** para mostrar la vista correspondiente al tipo de archivos que se está tratando, y también usa una serie de filtros para filtrar el tipo de archivos que se está tratando. **FiltroConfiguracion** se utiliza para filtrar sólo archivos de configuración, **FiltroGraficasJPG**, **FiltroGraficasPNG** y **FiltroGraficasGIF** filtran los formatos de imágenes que se pueden salvar como JPEG, PNG y GIF respectivamente. Por último todos los paquetes tiene un controlador de errores, **ExcepcionDeArchivos** se encarga de reportar los errores de este paquete.

Paquete Astronomia

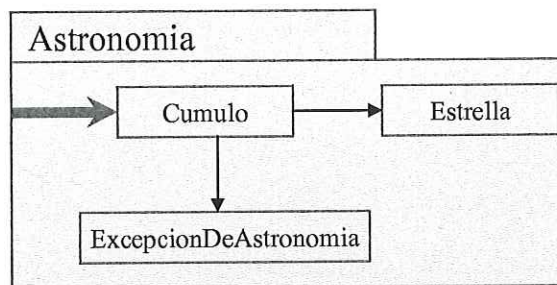


Figura IV.3.C: Diagrama de clases del paquete **Astronomia**.

En el paquete **Astronomia**, **Cumulo** necesita de **Estrella** para poder formarse, ya que en concepto un cúmulo es un conjunto de estrellas. En este paquete de software un cúmulo se representa como un vector o un arreglo de estrellas. Además, **Cumulo** usa **ExcepcionDeAstronomia** para reportar los errores de este paquete.

Paquete Configuracion

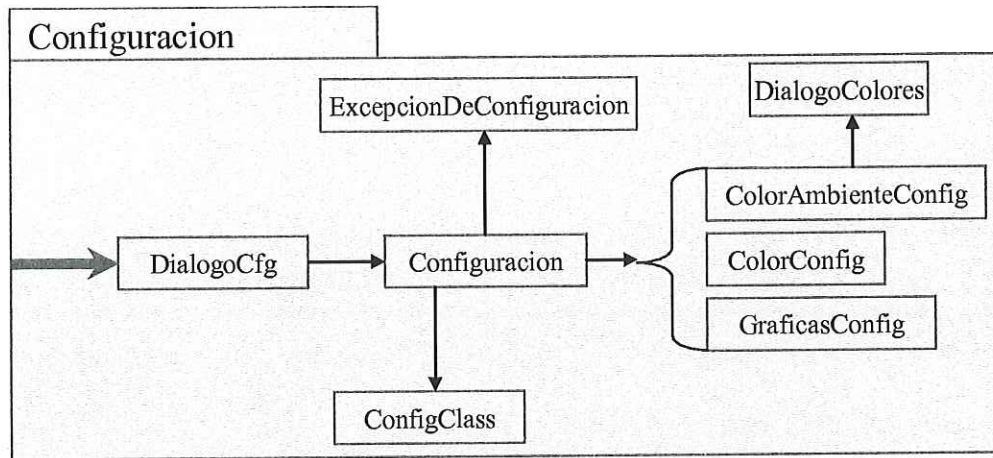


Figura IV.3.D: Diagrama de clases del paquete **Configuracion**.

En este paquete, **ColorAmbienteConfig** es una interfaz para la elección de los colores de ambiente y usa a **DialogoColores** que es una interfaz de elección de colores mediante un gamma de colores predefinidos por el sistema operativo. **Configuracion** es la interfaz de presentación de la configuración y agrega a **GraficasConfig** que se encarga de presentar la interfaz para re-organizar el orden de aparición y el tipo de las gráficas, a **ColorConfig** que es la interfaz para que el usuario cree nuevos colores o elimine estos mismos de manera que puedan ser usados en **GraficasConfig**, y a **ColorAmbienteConfig**. **Configuración** usa **ConfigClass** que es la representación interna de la configuración para llenar la interfaz con los datos de la configuración actual y regresarla modificada. También usa **ExcepcionDeConfiguración** que es el que reporta los errores de este paquete. Por último **DialogoCfg** es una clase que crea un dialogo de usuario la cual carga como interfaz principal a **Configuración**.

Paquete Eventos

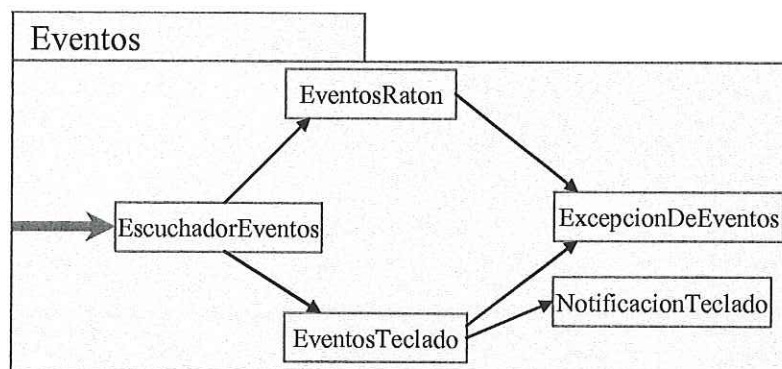


Figura IV.3.E: Diagrama de clases del paquete **Eventos**.

En este paquete, **EventosTeclado** escucha los eventos que ocurren del teclado y crea una **NotificacionTeclado** para almacenar el tipo de acceso que desee usar el usuario, enviándole esta clase al padre que lo creo, y usa **ExcepcionDeEventos** para reportar los errores que ocurran en este proceso.

EventosRatón escucha los eventos que ocurren del ratón y comunica al padre que lo creo, los cambios hechos en un objeto de tipo **Cumulo** y usa **ExcepcionDeEventos** para reportar los errores que ocurran en este proceso.

Por último **EscuchadorEventos** escucha los eventos del componente padre que lo creo y usa **EventosRatón** y **EventosTeclado** para escuchar eventos del ratón y del teclado también.

Paquete Graficas

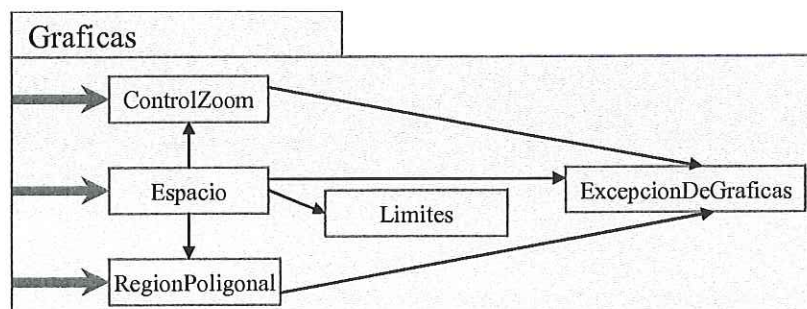


Figura IV.3.F: Diagrama de clases del paquete **Graficas**.

En este paquete, **ControlZoom** se encarga de poner el control para poder acercar una parte de la gráfica y ver con más claridad, usa **ExcepcionDeGraficas** para reportar los errores que pueda generar.

RegionPoligonal se encarga de poner los controles para poder crear regiones poligonales e interactivas para el usuario y poder seleccionar un conjunto de estrellas y usa **ExcepcionDeGraficas** para reportar cualquier error que pueda ocurrir. Por último,

Finalmente, **Espacio** que se encarga de graficar los cúmulos por medio de un objeto de tipo **Cumulo**, usa **Limites** para calcular los límites de graficación y poner los ejes, las etiquetas y los valores máximos y mínimos para el eje X y para el eje Y, también grafica **ControlZoom** y **RegionPoligonal**.

Paquete Interfaz

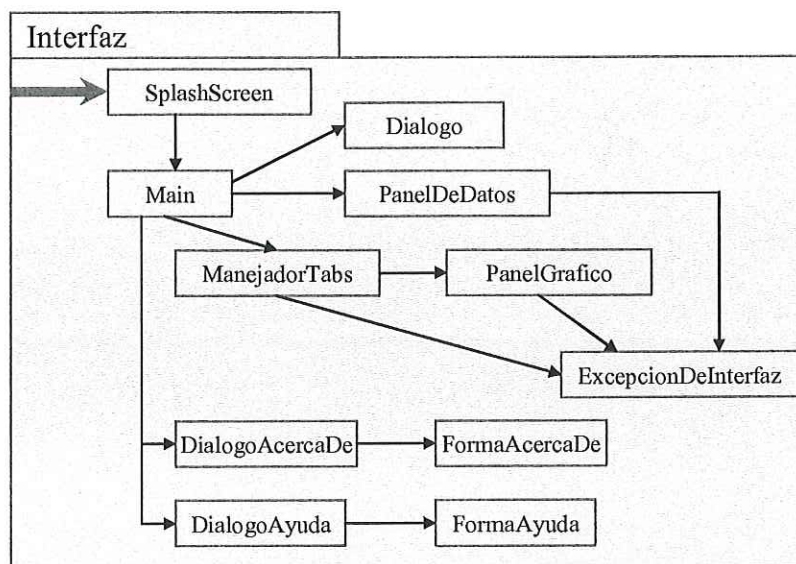


Figura IV.3.G: Diagrama de clases del paquete **Interfaz**.

Sin duda el paquete más grande, es el que se encarga de los varios detalles de la interfaz. **DialogoAcercaDe** es el encargado de mostrar un diálogo con **FormaAcercaDe** que, a su vez se encarga de mostrar la información típica acerca del sistema. **DialogoAyuda** es el encargado de mostrar un diálogo con **FormaAyuda** que, a su vez se encarga de mostrar una interfaz interactiva para la ayuda del sistema. **PanelGrafico** se encarga de acomodar las gráficas en una tabla (desde una gráfica hasta 16 usando un arreglo 4x4) y usa **ExcepcionDeInterfaz** para reportar los errores que genere. **PanelDeDatos** se encarga de mostrar la interfaz que genera **TablaDeDatos** del paquete **TablaInteractiva** en otra ventana y usa **ExcepcionDeInterfaz** para reportar cualquier error que se genere.

Main es la clase principal, la interfaz principal y la entrada del sistema, presenta el menú y presenta también **ManejadorTabs** que es el que se encarga de manejar las pestañas donde

estarán diferentes **PanelGrafico** con el mismo objeto de tipo **Cumulo** del paquete de Astronomía. **Main** es el que controla a **PanelDeDatos**, **DialogoAcercaDe** y **DialogoAyuda**.

La clase **SplashScreen** es la que se encarga de la interfaz de entrada al sistema, muestra datos del sistema y luego manda llamar a **Main**. Por último, **Main** tiene una clase que todos los paquetes usan para mostrar mensajes al usuario, **Dialogo**.

Paquete Procesos

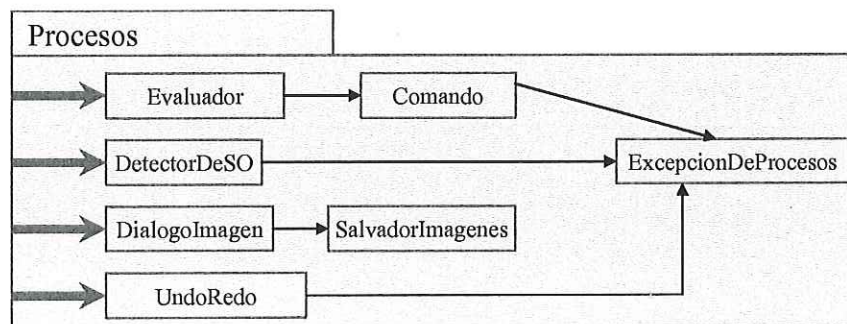


Figura IV.3.H: Diagrama de clases del paquete **Procesos**.

En este paquete, **Evaluador** es el encargado de evaluar expresiones matemáticas y proveer por medio de **Comando** una serie de funciones matemáticas (sen, cos, tan, exp...) para poder ampliar la cantidad de colores que se puedan crear para el estudio de los cúmulos.

DetectorDeSO es la clase que se encarga de detectar el Sistema Operativo en el que se está trabajando, actualmente contempla Windows y Linux y prepara ciertas variables especiales que difieren entre las dos plataformas. Aunque Java es multiplataforma, el ejecutar un sistema creado en Java en dos o más plataformas que lo soporten, es necesario considerar las diferencias entre ellos. De esta forma, el sistema no debe tener problemas en ser ejecutado sobre MacOS, Windows o Linux. Por ejemplo el carácter de separación de carpetas en Windows es "\" y en Linux es "/" lo cual crea conflictos si no se identifica.

DialogoImagen es un dialogo que se presenta con **SalvadorImágenes** para que el usuario elija la gráfica que desea guardar en forma de imagen en un archivo utilizando cualquiera de tres posibles formatos diferentes (JPG, PNG y GIF).

UndoRedo se encarga de generar eventos y cuando el usuario lo solicite regresar a un estado previo del sistema, es decir cualquier cambio hecho en un cúmulo poderlo deshacer e incluso poderlo rehacer.

Comando, **DetectorDeSO** y **UndoRedo** usan **ExcepcionDeProcesos** para reportar los errores generados en este paquete.

Paquete TableInteractiva

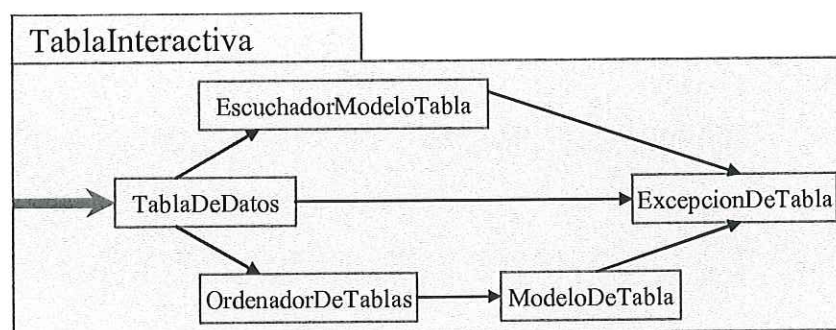


Figura IV.3.1: Diagrama de clases del paquete **TableInteractiva**.

En este paquete, **OrdenadorDeTablas** es el que se encarga de ordenar los datos cuando hay un evento de ratón en la cabecera de una columna. Usa **ModeloDeTabla** para especificar como estará construida esta tabla y los datos que llevará junto con los nombre de columna y usa **ExcepcionDeTabla** para reportar errores generados.

EscuchadorModeloTabla es un escuchador para **TablaDeDatos**, escucha los eventos de ratón y los reporta a **OrdenadorDeTablas**. Además reporta errores a través de **ExcepcionDeTabla**.

TablaDeDatos es la tabla que contiene un cúmulo descrito por sus datos fotométricos, y usa a **EscuchadorModeloTabla** y a **OrdenadorDeTablas**, y a su vez reporta cualquier error a través de **ExcepcionDeTablas**.

IV.3.3 - Diseño de interfaces

Configuración

La configuración se planeó en 4 pantallas o interfaces distintas para no confundir al usuario. Cada interfaz trata de englobar las opciones de configuración que más se acomoden y con base su propósito y manera de funcionar: Definir tipo de gráficas, Crear colores para gráficas, Colores de ambiente y Extras de ambiente, (Ver Figura IV.3.J, A). En si las 4 interfaces de configuración están dentro de un gran diálogo de Configuración. Además, el diálogo presenta tres botones con las opciones (Ver Figura IV.3.J, B):

1. Salvar: Para poder guardar la configuración en el archivo actualmente cargado, en caso de ser la primera vez en ejecución o no tener un archivo de configuración cargado, los datos se almacenaran temporalmente en el sistema mas no en un archivo por lo que si el sistema se cierra esos cambios se perderán.
2. Salvar Como: Para almacenar la configuración como una nueva en un distinto archivo.
3. Cerrar: Para cerrar la configuración sin salvar ningún cambio.

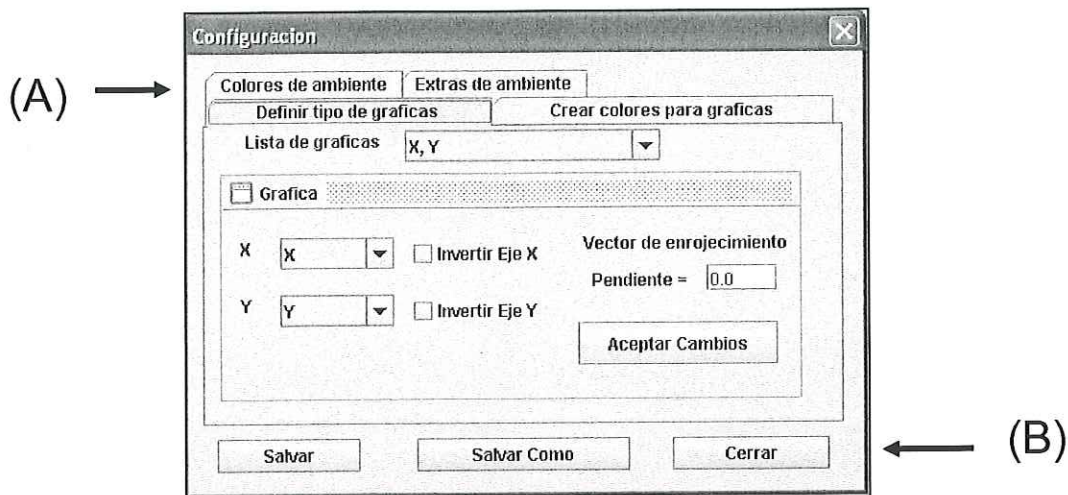


Figura IV.3.J: Interfaz de la primera pantalla de configuración.

Interfaz “Definir tipo de gráficas”

La primera interfaz que se presenta y la más importante en el uso del sistema es la interfaz de configuración para “Definir tipo de gráficas”. Comienza con una lista “Lista de gráficas” donde el usuario puede seleccionar una de las 16 gráficas que tiene la configuración actual. Una vez seleccionada (por omisión se tiene X,Y), el bloque de Gráfica se modifica mostrando en X el valor correspondiente al eje de las X y en Y el eje valor correspondiente al eje de las Y en forma de lista donde el usuario ya puede elegir que color deberá ir en cada eje. A un lado de cada eje se encuentra una caja que se puede seleccionar, indicado que ese eje se deberá graficar de manera invertida. A la derecha se tiene un bloque de texto donde el usuario puede escribir el valor de la pendiente de esa gráfica para su correspondiente vector de enrojecimiento. Para poder almacenar los cambios sin alterar la configuración y tratar de evitar confusiones se debe de aplastar el botón “Aceptar Cambios”.

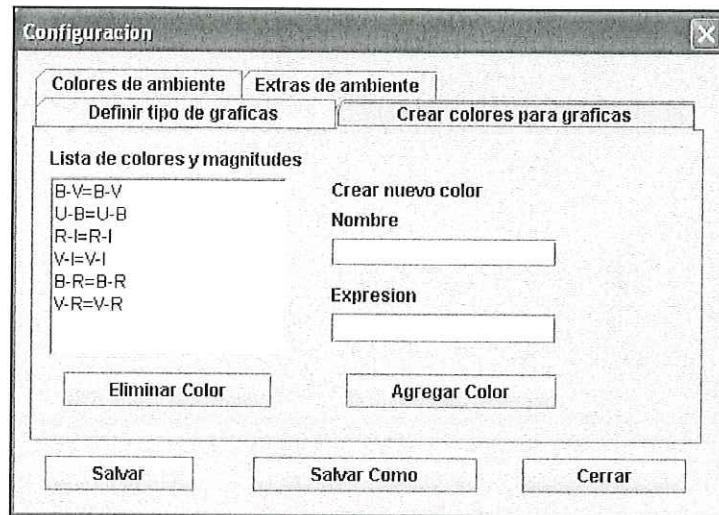


Figura IV.3.K: Interfaz de la segunda pantalla de configuración.

Interfaz “Crear colores para gráficas”

La segunda interfaz es para “Crear colores para gráficas”, en la parte izquierda se tiene una lista de colores y magnitudes que son los que el usuario ha creado y poder ver cuales ya están sin contar los que ya vienen en el archivo. En la parte derecha se tiene un cuadro de texto para el nombre que se le dará al color y abajo otro que es la expresión del color que puede ser formado por los símbolos (+-*/%) y por colores sacados del archivo.

Un ejemplo de un color creado puede ser:

Nombre: ColorEjemplo

Expresión: $(U-B) * 0.2 + (I-V)$

El botón “Agregar Color” acepta el color una vez definido y lo agrega a la lista de colores creados por el usuario.

El botón “Eliminar Color” elimina un color seleccionado de la lista de colores creados.

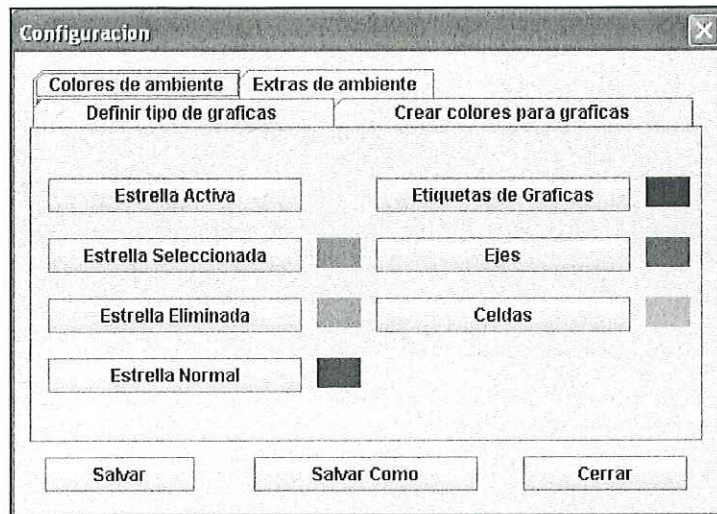


Figura IV.3.L: Interfaz de la tercera pantalla de configuración.

Interfaz “Crear colores para gráficas”

La tercera interfaz es para definir los “Colores de ambiente” y darle más personalización al sistema. Cada botón inicializa un diálogo de elección de colores donde se puede elegir, en una paleta de colores, el color deseado para cada parte del sistema y mostrándose a lado del botón el color actual. Por ejemplo el boton “Estrella Activa” cambiará el color que tomarán las estrellas que estén activas en el sistema y el color actual es el amarillo.

Se pueden cambiar de color las estrellas seleccionadas, activas, eliminadas o normales, el color de las etiquetas de las gráficas, ejes y celdas.

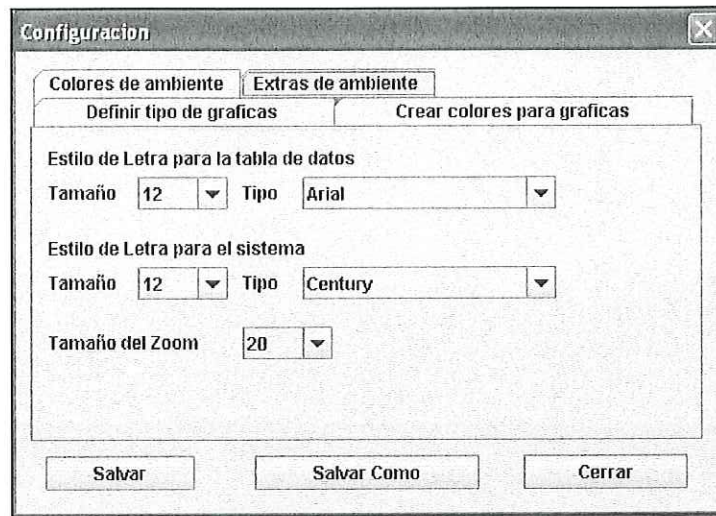


Figura IV.3.M: Interfaz de la cuarta pantalla de configuración.

Interfaz “Extras de ambiente”

La cuarta y última interfaz es para definir algunos valores “Extras de ambiente”, aquí se puede seleccionar el tipo y tamaño de letra para la tabla de datos y para el resto del sistema y también el tamaño global del control zoom (lente de acercamiento).

Interfaz “Objeto poligonal movable”

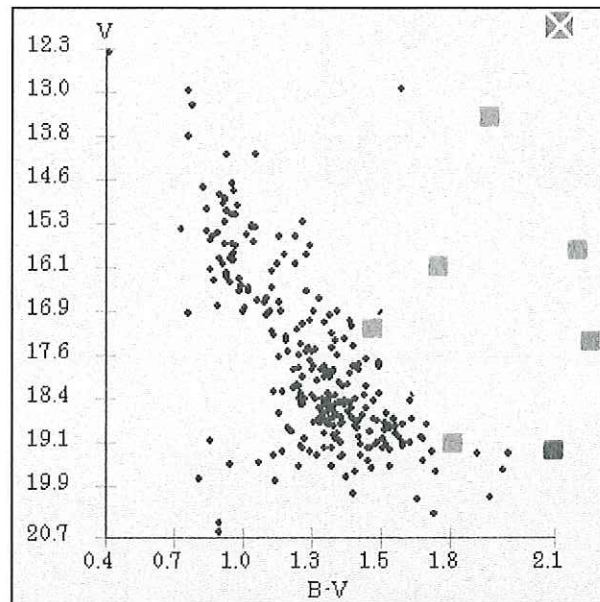


Figura IV.3.N: Interfaz de la de la región poligonal.

Una de las interfaces más difíciles de desarrollar fue un objeto poligonal movable que se pudiera poner en cualquier forma, tomando en cuenta que entre más vértices más flexibilidad de formas tiene (Ver Figura IV.3.N). Así con un límite de 10 vértices hasta mínimo tres se puede crear un polígono movable con la herramienta Región Poligonal. Cada cuadro del polígono es un cuadro de control, cada uno se puede mover El azul indica inicio y el rojo indica fin, si se le dá clic a un punto de control con el botón derecho se eliminará pero si se de clic en otra parte con el botón derecho se generará otro punto de control en ese lugar.

Interfaz “Vector de enrojecimiento”

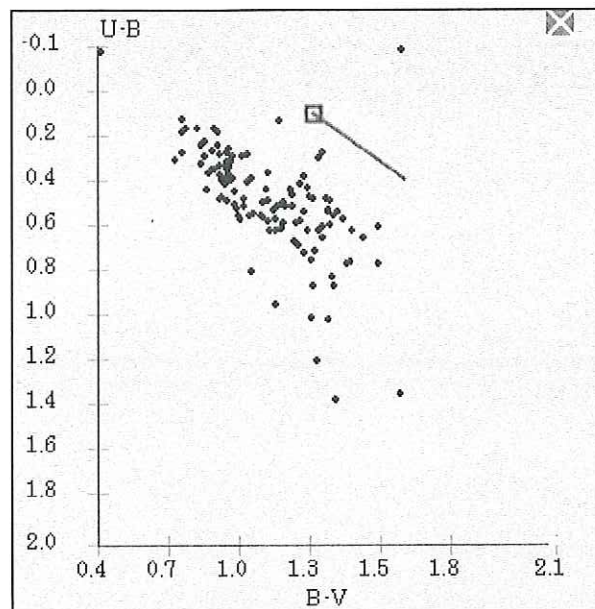


Figura IV.3.O: Interfaz del vector de enrojecimiento.

El vector de enrojecimiento es sólo una línea con un punto de control para poder mover en una parte donde no molesta la visualización del cúmulo y su pendiente tiene que ver con el parámetro de configuración que se le da a cada gráfica.

Interfaz “Menús”

Los menús fueron una de las partes de la interfaz general que tuvo demasiados cambios durante el desarrollo, ya que no quedaba bien definido donde iba a quedar cada opción. Principalmente el menú está basado en un menú típico de programas de edición como texto, etc.

Menú Archivo

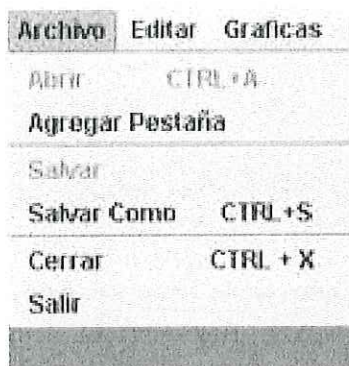


Figura IV.3.P: Menú Archivo.

El primer menú, Archivo, incluye las opciones para el manejo de archivos.

- Abrir: Abre un archivo *.txt con formato de datos fotométricos de cúmulo y despliega en forma de gráficas.
- Agregar Pestaña: Como se menciona en los requerimientos el sistema puede tener varias pestañas (máximo 20) y sirve para ir agregando pestañas al sistema.
- Salvar: guarda el cúmulo de trabajo en el archivo del cual fue cargado.
- Salvar Como: guarda el cúmulo de trabajo en un archivo distinto.
- Cerrar: Cierra la pestaña que se encuentra activa, es decir la que está actualmente seccionada.
- Salir: Para salir del sistema y cerrar, ningún cambio es guardado.

Menú Editar

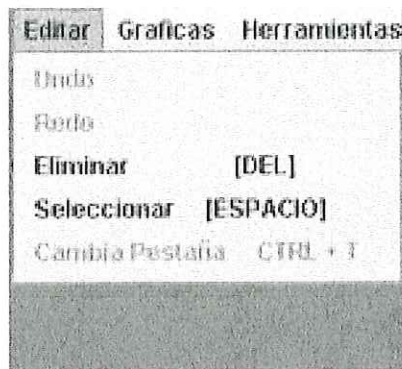


Figura IV.3.Q: Menú Editar.

El siguiente menú tiene que ver con la edición, Editar engloba las opciones de edición:

- Undo: Deshace un cambio realizado en el sistema.
- Redo: Vuelve a hacer un cambio deshecho con anterioridad.
- Eliminar: Elimina las estrellas que se encuentran actualmente seleccionadas en el sistema.
- Seleccionar: Selecciona las estrellas que están actualmente activas en el sistema.
- Cambiar pestaña: Cambia de una pestaña a otra.

Menú Gráficas

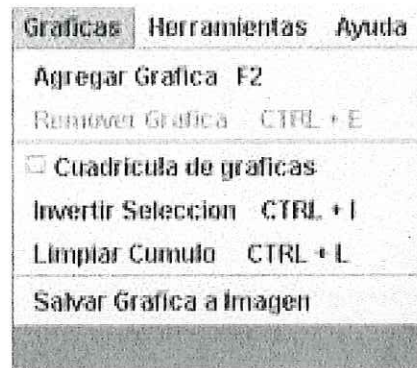


Figura IV.3.R: Menú Gráficas.

El tercer menú, uno de los más importantes, Gráficas engloba todo lo referente al manejo de las gráficas con excepción de la configuración que quedo englobada en Herramientas:

- Agregar gráfica: Agrega una gráfica respetando el orden en el que están en la configuración.
- Cerrar gráfica: Cierra la última gráfica que se agregó. Cada gráfica tiene en la parte superior derecha un botón [X] con el cual también se pueden cerrar sin importar el orden.
- Cuadrícula de gráficas: Muestra una cuadrícula en cada gráfica.
- Invertir selección: Invierte las estrellas que están actualmente seleccionadas en no seleccionadas y viceversa.
- Limpiar cúmulo: Restaura el cúmulo y lo deja como si ninguna operación se hubiera realizado.
- Salvar gráfica a imagen: Muestra un diálogo para guardar las gráficas a imágenes en 3 posibles formatos distintos (JPEG, GIF, PNG).

Menú Herramientas

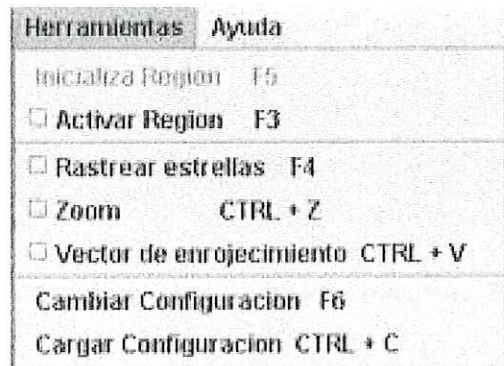


Figura IV.3.S: Menú Herramientas.

Otro de los más importantes menús, Herramientas que engloba la parte funcional del sistema:

- Inicializa región: Inicializa la región poligonal y la devuelve a su estado inicial en el centro de cada gráfica (La región poligonal no se refresca por lo que cada vez que se mueve el tamaño de la interfaz principal hay que inicializar la región ya que a veces se pierden puntos de control).
- Activa región: Activa la región poligonal activando las estrellas que encapsula.
- Rastrear estrellas: Persigue el ratón tratando de activar la estrella que está más cerca del cursor. Esta herramienta consume un poco de recursos. Las estrellas se puede activar con sólo pasar el ratón.
- Zoom: Abre una ventana mostrando el acercamiento del área en el que esté el control del zoom o lente de acercamiento.
- Vector de enrojecimiento: Muestra el vector de enrojecimiento.
- Cambiar configuración: Muestra el diálogo de configuración.
- Cargar configuración: Abre un diálogo para abrir una configuración de algún archivo.

Menú Ayuda

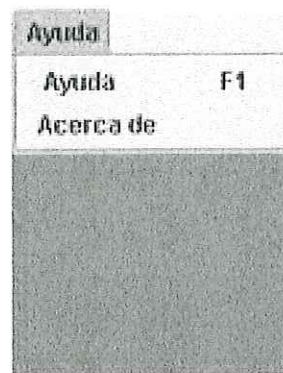


Figura IV.3.T: Menú Herramientas.

El último menú es el de Ayuda, cuenta con un mini sistema de ayuda y el Acerca del Sistema, diseñador y desarrollador, lenguaje, idea original, licencias, etc.

Interfaz “Menús”

ID	X	Y	O	B	V	R	I	EmL0	EmE0	EmO0	EmF0	EmD	Selecc	Eliminada
0	126.1	17.2	15.025	14.5311	12.2313	12.1133	59.999	0.012	0.0020	0.0030	0.02	99.999		
1	57.8	19.4	13.0416	17.4959	16.4609	15.9019	15.1297	0.016	0.0060	0.0040	0.0030	0.0030		
2	91.4	21.5	15.7634	16.419	15.3035	14.7518	13.2502	0.012	0.0010	0.0030	0.0030	0.0010		
3	101.7	42.4	16.9798	16.4903	15.578	15.0637	14.4843	0.0090	0.0010	0.0040	0.0030	0.0020		
4	131.4	64.9	17.1289	16.4924	15.651	15.0294	14.2532	0.013	0.0020	0.0020	0.0020	0.0010		
5	400.4	115.0	19.521	17.0302	16.0849	16.3302	15.6179	0.0070	0.0030	0.0050	0.0060	0.0040		
6	457.9	133.2	19.2608	15.9106	15.0311	14.5097	13.9564	0.012	0.0040	0.0010	0.0060	0.0020		
7	190.7	126.6	20.123	19.7737	18.4673	17.0776	16.7376	0.279	0.04	0.01	0.01	0.0099		
8	369.8	124.8	19.469	17.8290	16.625	16.261	15.5142	0.012	0.0030	0.01	0.0040	0.0040		
9	459.0	127.5	15.5852	16.2001	15.4369	15.0785	14.2786	0.044	0.018	0.013	0.028	0.063		
10	651.8	131.8	16.0652	15.1719	14.1234	13.5245	12.9638	0.029	0.0040	0.0030	0.0040	0.0020		
11	407.0	134.1	17.3841	17.0109	16.1345	15.645	15.0193	0.013	0.0040	0.0010	0.0040	0.0050		
12	385.1	150.8	19.7938	19.2327	17.9121	17.1291	16.76	0.047	0.0060	0.0020	0.0040	0.0050		
13	691.8	161.9	16.1036	16.014	15.0362	14.4675	13.7323	0.035	0.0020	0.0040	0.0020	0.0020		
14	427.6	163.6	19.0559	17.2012	16.9073	16.4074	15.7487	0.031	0.057	0.0030	0.0030	0.0020		
15	174.5	177.7	16.7495	16.346	15.4827	14.9819	14.3336	0.016	0.0060	0.0030	0.0030	0.0010		
16	696.8	195.2	19.0951	18.5925	17.2769	16.9421	15.8328	0.025	0.0060	0.0070	0.0030	0.0040		
17	314.1	195.9	17.7505	17.1302	15.8866	15.1207	14.2577	0.018	0.0040	0.0040	0.0050	0.0010		
18	711.4	190.2	17.386	16.9348	15.9736	15.4380	14.7197	0.0090	0.0040	0.0050	0.0010	0.0020		
19	511.4	223.3	17.6768	17.1259	16.1918	15.6426	14.9456	0.016	0.0050	0.0060	0.0020	0.0030		
20	136.9	232.5	17.2179	16.5533	15.3258	14.5965	13.8116	0.0080	0.0060	0.0030	0.0030	0.0020		
21	591.3	236.3	16.9215	16.5506	15.5361	14.9399	14.262	0.019	0.0090	0.0050	0.0030	0.0030		
22	370.3	237.1	17.6768	17.231	16.3393	15.8366	15.1996	0.011	0.0030	0.0050	0.0030	0.0030		
23	611.1	245.0	19.0708	18.4929	17.329	16.6828	15.9036	0.151	0.0040	0.0040	0.0050	0.0050		
24	669.1	251.3	17.1847	16.8343	15.8902	15.3482	14.6515	0.051	0.0020	0.015	0.0040	0.0010		
25	587.8	262.5	19.2093	18.5369	17.373	16.6994	15.9613	0.06	0.019	0.0030	0.0040	0.0050		
26	121.6	265.3	18.829	18.329	17.1051	16.3513	15.4958	0.032	0.01	0.0050	0.0030	0.0030		
27	762.5	277.0	19.1814	18.6039	17.2717	16.4536	15.524	0.061	0.0070	0.0060	0.0030	0.0030		
28	657.2	291.3	17.3246	17.3302	16.3372	15.8202	15.1326	0.02	0.0070	0.0070	0.0040	0.0020		
29	379.1	296.3	19.9199	17.3171	15.4918	14.434	13.3057	0.027	0.0050	0.0030	0.0020	0.0020		
30	489.9	309.7	14.6632	13.896	13.0166	12.5616	11.924	0.0090	0.0030	0.0020	0.0020	0.0010		
31	776.9	330.2	16.7599	16.4127	15.5104	15.0040	14.3409	0.013	0.0020	0.0030	0.0010	0.0010		
32	144.0	331.2	19.0744	18.3943	17.0773	16.2677	15.3468	0.029	0.014	0.0060	0.0040	0.0040		
33	521.3	334.5	19.1367	18.7463	17.5705	16.823	15.9823	0.079	0.0090	0.0059	0.0020	0.0040		
34	606.7	345.0	16.4697	16.0139	15.0373	14.4931	13.7977	0.013	0.0020	0.0010	0.0020	0.0020		
35	901.5	358.4	17.182	16.7005	15.7299	15.179	14.4639	0.011	0.0010	0.0020	0.0020	0.0030		
36	15.1	372.9	20.81	20.1069	18.6974	18.0196	17.1095	0.5	0.013	0.013	0.015	0.0050		
37	384.0	374.1	14.8171	14.6131	13.6215	13.3594	12.7520	0.0070	0.0020	0.0030	0.0020	0.0020		
38	236.4	380.1	17.1616	16.8321	15.6573	15.2517	14.5493	0.014	0.0010	0.0040	0.0020	0.0020		
39	424.8	393.4	17.8613	17.2977	16.3094	15.7387	15.0419	0.022	0.0060	0.0020	0.0040	0.0030		
40	569.3	414.1	16.909	16.4405	15.3968	14.779	14.0042	0.02	0.0020	0.0060	0.0020	0.0010		
41	236.2	418.3	12.7193	12.8208	12.3356	12.0424	11.6596	0.0090	0.0020	0.0030	0.0020	0.0010		
42	248.8	418.9	19.8041	18.2983	17.0414	16.2793	15.3593	0.04	0.012	0.0040	0.0030	0.0030		

Figura IV.3.U: Interfaz de la tabla de datos.

La interfaz de la tabla de datos estuvo por primera vez integrada con la interfaz de las gráficas y cada pestaña tenía su propia tabla de datos. Si embargo, debido a que en cada instancia del sistema sólo se puede abrir un cúmulo, no tenía caso tener muchas tablas y mantenerlas con la misma información. Así que se generó una ventana aparte donde las columnas de la tabla son las del archivo X, Y, los colores y luego los errores seguido por 2 cajas de marcado para seleccionar y eliminar y donde también se reflejan los cambios de las gráficas. Se pueden ordenar los datos de menor a mayor y viceversa y las estrellas actualmente seleccionadas se van poniendo en la parte superior de la tabla.

Interfaz “Acercamiento o Zoom”

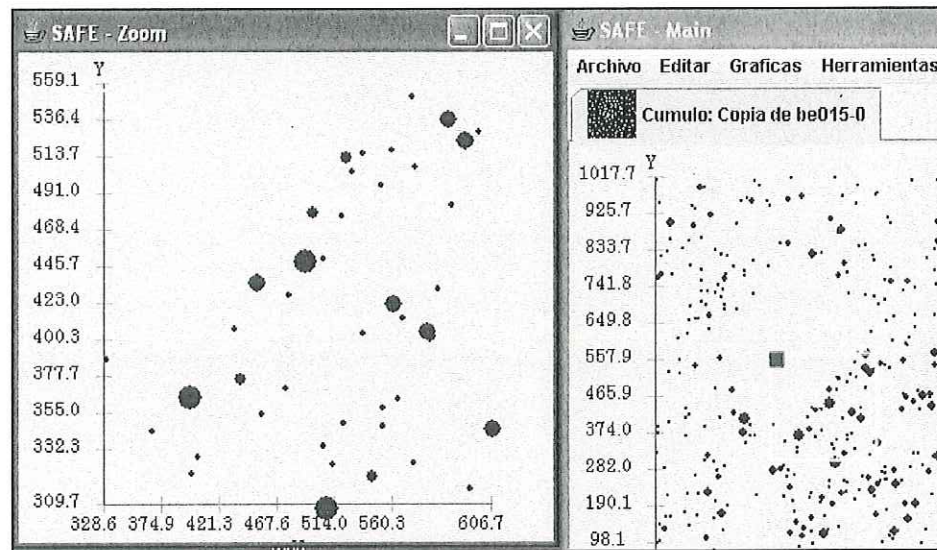


Figura IV.3.V: Interfaz del zoom.

La interfaz de zoom es una ventana donde se muestra una porción del tamaño original del cúmulo. Esta porción es representada por un cuadro en la gráfica original y que puede ser movida por el usuario. Realiza la misma funcionalidad que la original y el propósito es poder seleccionar las estrellas chicas o amontonadas con más facilidad.

Interfaz “Generador de imágenes”

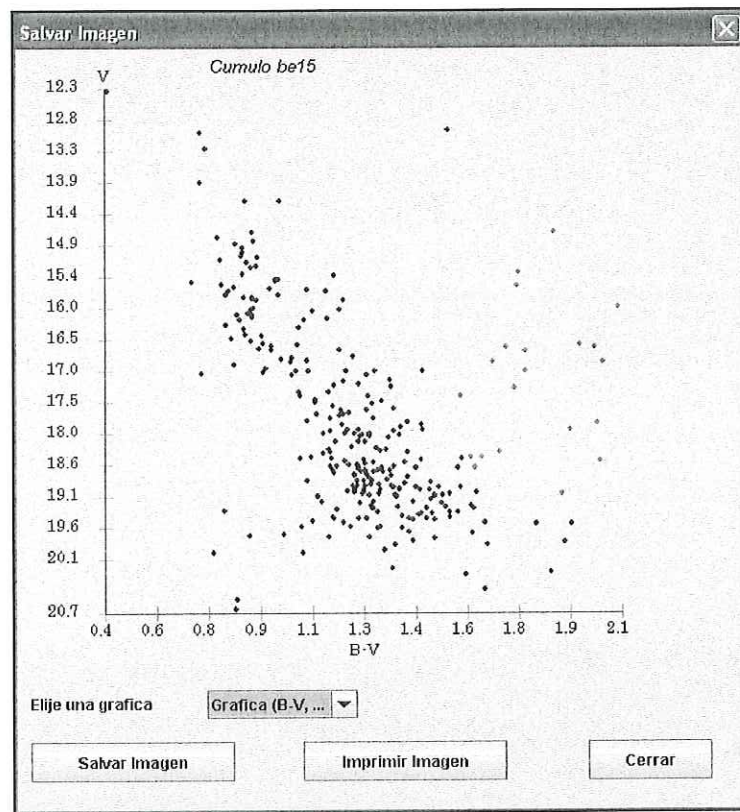


Figura IV.3.X: Interfaz del salvador de imágenes.

La interfaz Generador de imágenes genera la imagen de una de las gráficas definidas por el usuario. Con esta imagen generada se puede imprimir o salvar en archivo para futuro uso. La imagen respeta las estrellas que estaban seleccionadas en el momento e imprime el nombre del cúmulo.

IV.3.4 - Diseño de algoritmos e implementación

Durante el desarrollo de la herramienta se tuvieron que diseñar algoritmos para resolver ciertos problemas que no tenían solución directa, tales como el control de diálogos, del zoom y de las regiones poligonales, operaciones para deshacer y rehacer cambios de eventos realizados y para la graficación de los cúmulos, entre otros. Esto motivó el uso patrones de diseño, los cuales son soluciones recurrentes a problemas de diseño comunes (Alpert Sherman K. Brown B. Woolf, 1998), y que constituyen una serie de reglas describiendo como lograr ciertas tareas en el mundo del desarrollo de software (Pree, 1994). Los patrones de diseño se enfocan más en reusar diseño de arquitectura recurrente, mientras que Framework se centra en un diseño e implementación detallado. Son especificadas en forma genérica (sin lenguaje especificado). Los patrones de diseño se encuentran en situaciones específicas y presentan la solución a esta situación (Buschman and Meunier, et al., 1996). No siempre pueden ser útiles, sólo si la ocasión lo requiere por el tipo de software que se esté diseñando.

Los patrones de diseño se pueden dividir en tres tipos:

- Patrones de creación: Crean objetos para ti, en vez de que tengas que instanciar objetos directamente. Tu programa gana mas flexibilidad en decidir cual objeto necesita ser creado para cierto caso. (Ej. Patrón de método de fábrica conocido como factory method pattern y patrón singleton conocido como singleton pattern).
- Patrones estructurales: Te ayudan a componer grupos de objetos en estructuras más grandes, tales como interfaces complejas de usuario y contando información. (Ej. Patrón proxy conocido como proxy pattern y patrón decorador conocido como decorator pattern).
- Patrones de comportamiento: Te ayudan a definir la comunicación entre objetos en tu sistema y como el flujo es controlado en un programa complejo. (Ej. Patrón de la

cadena de responsabilidad conocido como chain of responsibility pattern y patrón de interpretación conocido como interpreter pattern).

Aprender el uso de patrones de diseño es un proceso de pasos múltiples: Aceptación, Reconocimiento e Interiorización. Primero aceptas la premisa de que los patrones de diseño son importantes en tu trabajo. Luego, reconoces que necesitas leer acerca de los patrones de diseño para poder saber cuando pudieras usarlos. Finalmente, interiorizas los patrones en suficiente detalle para saber cuales pudieran ayudarte a resolver un cierto problema de diseño (J. W. Cooper, 2000).

De los patrones de diseño encontrados, muchos de ellos ya implementados en Java incluyendo:

1. Adaptador (Adapter): Sirve para convertir la interfaz de una clase en otra. Usamos adaptadores cuando clases no relacionadas trabajan en el mismo programa. El concepto de un adaptador es simple; escribimos una clase que tiene la interfaz requerida y la comunicamos con la clase que tiene la otra interfaz. Este patrón se usa cuando una clase no puede encapsular una serie de eventos, algunos ejemplos incluyen: *ComponentAdapter*, *ContainerAdapter*, *FocusAdapter*, *KeyAdapter*, *MouseAdapter* y *MouseMotionAdapter*.
2. Puente (Bridge): Sirve para separar la implementación de una clase de su interfaz para variar la implementación sin cambiar el código cliente. En Java esto es muy usado en manejo de eventos, donde cada evento tiene una interfaz donde el que la implementa puede tener diferente respuesta. Como ejemplo de esto se tienen a los escuchadores (listeners): *TableModelListener*, *MouseListener*, *MouseMotionListener*, *KeyListener*.
3. Decorador (Decorator): Provee una forma de cambiar el comportamiento de objetos individuales sin tener que crear clases derivadas. Según el patrón de diseño

implementado en Java resultaría el tener que usar clases abstractas que se deberían de derivar (extender en Java) donde está el código de control y usarlo desde las clases abstractas.

4. **Peso ligero (Flyweight):** Hay casos en los que en la programación necesitas representar un gran número de clases pequeñas para representar la información. Se puede reducir el número de estas clases si se reconoce que son exactamente iguales a excepción de ciertos parámetros. Si se mueven esas variables a una clase externa y se pasan en forma de llamada a un método, el número de clases internas se reduce. Este patrón provee una forma de resolver este tipo de problemas donde un valor común o calculable puede ayudar a reducir el número de clases. Este patrón no pudo implementarse debido a que se descubrió un poco tarde. Hubiera sido útil y con ganancia en el procesamiento debido a la cantidad de objetos de tipo **Estrella** que se tienen que manejar. Otra razón por la cual no hubiera convenido implementarse es que son datos totalmente distintos para cada estrella y no hay parámetros que sean calculables o comunes mas que el nombre del cúmulo y la cantidad de estrellas que están dentro de la clase **Cumulo**.
5. **Interprete (Interpreter):** Algunos programas se benefician de tener un lenguaje para definir operaciones las cuales puedan procesar. Generalmente este patrón describe una gramática o una serie de reglas para ese lenguaje y usar esa gramática para interpretar sentencias de ese lenguaje. Usualmente se usa cuando hay que ejecutar una cadena algebraica, cuando un sistema tiene que dar diferentes salidas como por ejemplo un generador de reportes o comunicación con una base de datos por medio de SQL (Structured Query Language). Este patrón fue usado para definir una configuración manejable tanto para el sistema, leyendo y entendiendo lo que había que hacer y manejable de manera manual por el usuario. También para poder generar nuevos

colores basados con expresiones algebraicas y poder ejecutarlas al momento de graficar.

6. Iterador (Iterator): Es uno de los más sencillos y más frecuentes. Permite el movimiento a través de una colección de datos usando una sola interfaz sin tener que conocer detalles de las representaciones internas. La implementación usada en Java es la interfaz *Enumeration* y que sólo consta de dos métodos. Es usado muy comúnmente en el sistema para recorrer las estrellas sin necesidad de saber cuantas existen realmente y de una forma más rápida.
7. Recuerdo (Memento): Supongamos que queremos guardar el estado interno de un objeto para que se pueda volver a cargar. Lo ideal sería poder guardar y restaurar una acción o estado sin que el objeto a guardar tenga que preocuparse por esta tarea y sin violar la encapsulación, este es el propósito de este patrón que requiere tener un acceso privilegiado al estado interno del objeto que se quiere guardar, y se definen tres objetos: 1) **Originador**, el objeto que se quiere guardar. 2) **Memento**, es otro objeto que guarda el estado del **Originador**. 3) **Encargado**, se encarga de cuidar los objetos de tipo **Memento** que se van guardando en orden y usa el **Memento** que es necesario para restaurar el estado del objeto. Este patrón se implementó pero no resultó satisfactorio ya que no realizaba las operaciones deseadas, posiblemente a una mala implementación del patrón. En su lugar se usó el patrón de Java UndoableEdit para resolver este problema. UndoableEdit usa cinco clases: *AbstractUndoableEdit*, *UndoManager*, *UndoableEditSupport*, *UndoableEditListener*, *UndoAdapter*.
8. Observador (Observer): En este nuevo mundo de ventanas sofisticado, nos gustaría desplegar información en más de una forma al mismo tiempo y que se mostraran todos los cambios realizados a esa información. Este patrón asume que el objeto con la información es distinta que los objetos que la muestran y que estos objetos observan cambios en esta información. Este patrón es usado en el sistema debido a la

comunicación entre las 3 vistas (tabla de datos, gráficas y zoom cuando está activado) las cuales observan a un cierto cúmulo que se reporta cambiado por un escuchador de eventos asociado a una gráfica.

Muchos algoritmos fueron usados, pero describirlos tomaria de más espacio debido a que los más usados son muy grandes y usan muchas clases. Cuando se implementaba un algoritmo se escribía directamente, situación que no se debe de presentar en el desarrollo de software debido a problemas que se puedan presentar. Es preferible realizar algoritmos, diseñarlos en papel y analizar los posibles detalles que pudieran ocurrir y tratar de hacer metódico por futuros cambios que se le pudiere requerir realizar.

IV.4 – Desarrollo

IV.4.1 - Implementación del paquete de Astronomía

El paquete de **Astronomia** se enfoca concentrado en los aspectos astronómicos, estrellas y cúmulos, que es lo que el sistema tiene como fin. Este módulo es el más pequeño de todos y fue el primer módulo en ser desarrollado. Se tuvo que modificar después de la segunda entrega de requerimientos para poder cumplir con otros requerimientos y se agregó como controlador de errores una clase de tipo **Exception** para **Astronomia** donde se reportan los errores.

Al principio la clase principal cúmulo tenía un arreglo de objetos de tipo **Estrella**, pero era más complicado seguir el rastro de los cambios y de las actualizaciones. Esto se resolvió usando la clase *Vector* de Java con la cual se pueden mantener un tipo de lista de objetos y se puede usar el *Enumerator* para recorrer de forma rápida los elementos de dicha clase *Vector*.

La mayoría de los métodos en **Cumulo** sirven para controlar de una manera óptima las estrellas que forma parte de este cúmulo. Hay métodos para seleccionar, activar o eliminar estrellas en grupos y solas, sin tener que acceder en código externo a las estrella y modificarla.

Las estrellas se referencia en el programa por una variable ID que se guarda en el archivo modificando el formato del cúmulo cargado. Con esta variable no sólo se controlan mejor las estrellas internamente, si no que se puede tener registro de cuales y por que fueron las estrellas que se eliminaron del cúmulo.

IV.4.2 - Implementación del paquete de Gráficas

El paquete de gráficas fue el segundo módulo en ser desarrollado y está enfocado en proporcionar información y cálculos para mostrar gráficas, zoom, regiones poligonales, vector de enrojecimiento, los ejes y la cuadrícula.

Eje, Espacio y Límites sirven para mostrar las gráficas elegidas por el usuario en el tamaño de la ventana especificado por el usuario. **RegionPoligonal, ControlZoom y VectorR** son los controladores que sirven para mostrar los controles de las herramientas que llevan su nombre.

Al principio este paquete tenía un graficador simple y fácil de utilizar, pero la graficación variaba al cambiar el cúmulo, debido a que el cálculo de los valores (regla de tres) estaba mal. Después de corregir este error, se encontró que los valores no respetaban si se salían de la gráfica. El problema se resolvió creando la clase eje, que se encarga de calcular los valores de cada punto. Si la clase es para eje X entonces calcula el valor de un punto para el eje X, calcula los separadores y se auto dibuja automáticamente en un controlador de gráfico.

IV.4.3 - Implementación del paquete de Interfaz

El paquete de Interfaz fue el tercer módulo en ser desarrollado y comprende un conjunto de interfaces. Es el encargado de mostrar el ambiente visual al usuario, incluyendo menús, ayuda, diálogos, la ventana de zoom y el panel de datos. Además controla las pestañas de la interfaz principal. En este paquete se encuentra la clase principal **Main** que es el punto de inicio del sistema. Sin esta clase no arrancaría ningún paquete. Con el tiempo se le agregó una interfaz nueva que se llama **SplashScreen** que es una presentación del sistema y desaparece pasado un lapso pequeño de tiempo, mostrando la interfaz principal.

El sistema soporta diferentes configuraciones para cada pestaña así que cada una tiene que ser controlada por una clase que maneja el cúmulo de esa pestaña, la configuración, cuantas y cuales gráficas están en el momento. Al principio parecía un poco difícil pero al final se construyó una clase estable que podía manejar todo lo que se necesitaba, pero a costo de memoria. Es decir, entre más pestañas estén abiertas más memoria consume, lo cual puede resultar en un bajo rendimiento. El panel zoom fue uno de los últimos requerimientos cumplidos y uno de los más difíciles, al tratar de reflejar toda acción a escala y poder también obtener información de actividad del usuario, reflejándola a su vez de vuelta en la original y la tabla de datos.

IV.4.4 - Implementación del paquete de EventosDispositivos

Este tiene como fin de dar soporte a todos los eventos que vinieran tanto del ratón o como del teclado. Se implementa utilizando el patrón de diseño *Observer* implementado en Java. Solo es necesario especificar a quién va a reportar y qué tipo de reportes hace. De esta manera, manejar los eventos sin importar quién envía la señal. La clase escuchador de eventos (listener) se dividió para poder manejar más fácil si eran eventos de teclado o de ratón, convirtiéndose sólo la cara del paquete. **EventosRaton** se encarga de los diferentes eventos que se puede disparar por el ratón en el sistema, **EventosTeclado** se encarga de los diferentes eventos que pueden disparar a partir del teclado en el sistema, validando claves de atajo, etc. Se usa **NotificacionTeclado** para notificar en retorno a quién hizo la llamada los resultados.

IV.4.5 - Implementación del paquete de TablaInteractiva

Este paquete se encarga de manejar una gran cantidad de información en una tabla, actualiza ciertas operaciones (Ej. Seleccionar) en más de 16 gráficas, re-ordena la información

de la tabla, entre otras operaciones. Este paquete tiene comunicación directa con el paquete de **Graficos y Eventos**, a través de eventos del ratón se pueden seleccionar las dos opciones por estrella, seleccionar y eliminar, se puede re-ordenar, se muestra al tope las estrellas actualmente seleccionadas o activas y cualquier cambio hecho en la tabla se refleja en las gráficas mostradas en el momento.

La clase **EscuchadorModeloTabla** es sólo para poder mandar registro de eventos al paquete de **Eventos**, enviados desde **ModeloTabla**, que es la clase que da el formato a la tabla. La **TablaDeDatos** es la que se encarga de mostrar al usuario la información usando un **ModeloTabla** y un **OrdenadorDeTablas** para poder ordenar los datos contenidos según cuando así lo haya decidido el usuario.

IV.4.6 - Implementación del paquete de Archivos

Este fue de los primeros paquetes en estar listos con funcionalidad básica para guardar y cargar cúmulos con un cierto formato. Posteriormente creció para poder guardar imágenes de los cúmulos. Es un paquete sencillo que cuenta con filtros para tres tipos de imágenes (archivos) incluyendo los formatos *GIF (CompuServe Graphic Interchange)*, *JPG (JPEG, Joint Photographic Experts Group)*, *PNG (Portable Network Graphics)*. Otros tipos de archivos incluyen el TXT para la información de cúmulos (texto) y los archivos de configuración.

La clase **VistaArchivos** es la interfaz típica para abrir / cargar archivos en un sistema Linux gráfico (Por ejemplo: SUSE, Red Hat, Madrake, etc.), Windows o MacOs.

La clase **Utiles** proporciona métodos comunes o reusables, como una vista de iconos para la vista de archivo que proporciona un icono distinto para cada tipo de archivo.

IV.4.7 - Implementación del paquete de Configuración

Este paquete sirve para manejar la configuración del sistema y cuenta con su interfaz gráfica. Fue uno de los que más costó implementar ya que tuvo que rediseñarse más de dos veces. Cuenta con una interfaz principal **Configuration** que contiene un control de pestañas que contiene las diferentes opciones de configuración. **ColorAmbienteConfig** se encarga de la interfaz para elegir los colores del sistema. **DialogoColores** es la interfaz para seleccionar el color deseado para cierta parte del sistema. **ColorConfig** configura los colores que se usarán y el cómo se crean nuevos colores, los cuales puedan ser usados en las gráficas. **GraficasConfig** es la interfaz para controlar qué gráficas y en qué orden aparecerán. **DialogoCfg** es el que se encarga de mostrar a toda esta interfaz en forma de diálogo. Por último está la clase **ConfigClass** que es la que se encarga de almacenar la configuración y la cual se usa para abrir o guardar una configuración.

IV.4.8 - Implementación del paquete de Procesos

El paquete de procesos es el que se encarga de diversos procesos del sistema y fue de los más tardados en implementar ya que poco a poco fueron surgiendo nuevas operaciones de sistema que requerían su propia clase para poder ser procesados.

Estos procesos incluyen:

- Rehacer / Deshacer, operaciones para deshacer un cambio hecho en el cúmulo o hacerlo de nuevo.
- Detector de sistema Operativo, detecta en que sistema operativo se está ejecutando la aplicación para poder saber como manejar la información, archivos, etc.

- Evaluador de expresiones y Comandos, para evaluar expresiones de un lenguaje sencillo, usando comandos como Tan – para tangente, Cos – para coseno, Sen – para seno, entre otras operaciones matemáticas.
- Guardado de Imágenes, registra un cúmulo y sus cambios en un visor y transforma esta información visual en una imagen con la opción de poderla guardar a un archivo.

Este paquete es usado en todo el sistema, por su tipo de funcionalidad se pudiera ver como un paquete de utilidades.

IV.5 – Estimación de duración y esfuerzo del desarrollo

Paquete	Duración	No. Clases	Propiedades	Constructores	Métodos	Líneas de código
Astronomia	½ mes	3	22	5	27	900
Graficas	5 meses	7	83	13	61	2000
Interfaz	3 meses	12	106	13	103	3500
Eventos Dispositivos	1 mes	5	48	7	37	1000
Tabla Interactiva	4 meses	5	18	6	35	800
Archivos	5 meses	11	31	4	48	2200
Configuracion	5 meses	8	67	8	34	1600
Procesos	5 meses	7	55	6	35	1800
Total	28 meses	58	430	62	380	13800

Tabla IV.5.A: Tabla de estimación de duración y esfuerzo.

La duración aproximada establecida en los paquetes no es lineal, es más bien un proceso en paralelo, el tiempo es un calculo en horas de trabajo diario para realizar lo descrito en cada clase del paquete y completar el número de líneas de código descritas tambien en cada paquete, sin contar el hecho de que sufrió cambios entre versiones del paquete.

El tiempo total del proyecto fue de un año de desarrollo, el cual se distribuye en dos meses de investigación, nueve meses de trabajo y un mes de documentación.

V - Resultados y Discusión

V.1 - Evaluación del sistema

Para evaluar el sistema se entrevistó a un astrónomo que uso el sistema durante un tiempo, mientras otros usuarios desarrollaban otras tareas de investigación, para poder observar como impactó el uso del sistema y analizar la situación.

El Usuario1 comentó: "He encontrado **SAFE** (Sistema Analizador de datos Fotométricos de Estrellas) muy útil para ciertas cosas, pero seguramente no he usado todas sus herramientas. Por ejemplo, casi no he usado la Tabla de Datos. El otro día revise que las estrellas que estaba seleccionando y eliminando estaban así en la Tabla. Estoy usando las gráficas, nada mas, para casi todo.

Encuentro la interfaz fácil de usar para echar andar **SAFE**. Nada más estoy cambiando todo el tiempo la Configuración; encuentro otra lista de gráficas más útil."

Como observación, el Usuario1 sugirió que cada investigador que pudiera usar el sistema pudiera tener una propia configuración por omisión, esto impacta al sistema debido a que actualmente se carga siempre desde código su propia configuración. Esto se debe a que un requerimiento especifica que debía cargar una serie de gráficas antes analizadas que fueran útiles para los investigadores, pero debido al diferente estudio que se debe tener con cada cúmulo y dependiendo de los gustos de cada investigador estas gráficas cambian. Por lo que se puede asegurar un cambio futuro en el sistema de tener la configuración por omisión en un archivo.

V.2 - Uso y conclusiones

Algunas conclusiones que se analizaron del investigador que usó el sistema.

El Usuario1 comentó: "He usado **SAFE** mucho con [Activar Región], por ejemplo, en la gráfica X,Y para concentrar más en el centro del cúmulo, eliminando las externas; también en los diagramas color-magnitud para identificar y exportar ciertos grupos para sobre-graficarlas con *SuperMongo*, como las estrellas del "Red Clump" o gigantes rojas; o, las estrellas más brillantes, como comparan entre los diagramas de color-magnitud y de color-color; etc. Una vez que he identificado las del "Red Clump" y gigantes rojas, puedo usarlas en el diagrama color-color (B-V,U-B) para revisar con ellas si mis soluciones para enrojecimiento interestelar y metalicidad son consistentes entre las estrellas enanas y estas estrellas más evolucionadas.

También, he usado **SAFE** para comparar en detalle y simultáneamente varios de los diagramas de color-magnitud, como B-V,V; V-I,V; B-R,V; V-R,V; Esto me ayuda ver si todo está mas o menos consistente en los datos.

También, he usado **SAFE** para eliminar algunas de las estrellas más discrepantes. Probablemente no son miembros de los cúmulos sino estrellas en frente o atrás del cúmulo. Me gusta comparar sus posiciones en varios de los diagramas antes de eliminarlas."

V.3 - Resultados

Los resultados obtenidos se discutieron con el investigador y concluyó lo siguiente.

El Usuario1 comentó: "Hasta el momento he usado **SAFE** para "limpiar" en la gráfica X, Y siete cúmulos: Be89, Be92, Be95, Ki09, Ki18, NGC7296, y Ru135, y también he usado **SAFE** para analizar, yo mismo, cuatro de estos cúmulos: Be89, Be92, Be95, y NGC7296, para comparar los resultados con las de una estudiante. Dos de estos, Be89 y Be92, tienen las estrellas "Red Clump", y he usado **SAFE** para identificar y exportar éstas. Estos cuatro cúmulos todos están en la línea de visión de un brazo espiral de la Vía Láctea, y he intentado de usar **SAFE** para ayudarme separar las estrellas de los cúmulos de estrellas atrás, en el brazo espiral."

En el Apéndice A se puede observar un ejemplo con el cúmulo Be15 de algunas de las gráficas usadas para estudiar estos cúmulos, usando el sistema **SAFE**.

El sistema resultante permite manejar la información de un solo cúmulo mostrando 20 paneles (pestañas) y 16 gráficas distintas por panel (pestaña). Esto da un total de 320 graficas interactivas y editables a través de las cuales se puede visualizar y estudiar cada cúmulo de una manera más rápida y con mejor aproximación.

V.4 - Trabajo futuro

Como trabajo futuro se recomienda el desarrollo de funcionalidad adicional que facilitara en el trabajo los astrónomos que se dedican al análisis profundo de cúmulos abiertos y sus estrellas.

Por mencionar los más importantes:

- Graficación de las curvas isócronas, una curva que ya está calculada y se encuentra en un archivo de texto, y que ayuda a ajustar las estrellas y poder ver su enrojecimiento, del cual luego se obtiene el valor de exceso de una gráfica y es el valor que actualmente se proporciona en el sistema como la pendiente del vector de enrojecimiento.
- El registro de eventos como comandos, es decir, cada evento que se registra actualmente en el sistema de Undo-Redo se pueda guardar en un archivo de comandos o script, en el cual después se puedan guardar para que un cúmulo quede tal y como se dejo por última vez siguiendo estos comandos. Además, que el sistema Undo-Redo los pueda registrar para que todo el sistema reinicialice desde la última ejecución o incluso poder borrar manualmente los últimos comandos realizados.
- Conversión a formato *EPS (Encapsulated Post Script)*, el cual es un formato que usan mucho los astrónomos para presentaciones y carteles. Sin embargo es un formato difícil ya que es a base de comandos.
- Poder agregar etiquetas de texto a las gráficas cuando se guarden a archivo.

Aún quedan muchos cúmulos abiertos por analizar, pero se considera con base en la evidencia de los usuarios que con la ayuda de este sistema el tiempo de estudio se reducirá significativamente e incluso, en un futuro se podrán incluso en un futuro realizar estudios sobre otros cúmulos aún no observados.

VI - Conclusión

El trabajo realizado resultó un apoyo fuerte en el estudio de los cúmulos estelares. Después de un largo proceso de desarrollo y pruebas se logró tener un sistema en operación de una manera satisfactoria y seguramente, a través del cual se podrá adelantar al trabajo en una manera significativa. Dado que hicieron falta algunos requerimientos por especificar, los cuales se identificaron al final del desarrollo de la presente versión del sistema, se considera que el sistema permite realizar un 80% del trabajo. El otro 20% restante se realiza utilizando *SuperMongo* (Software de graficación de datos desarrollado por Robert Lupton and Patricia Monger encontrado en <http://www.supermongo.net/supermongonet-the-official-sm-website>) permitiendo analizar mejor los datos con mejores gráficas. Esto se logra al permitir al astrónomo generar un archivo con datos a graficar en un formato legible por *SuperMongo*. Con esto se puede decir que se logró el objetivo planteado ya a un año de desarrollo.

Con base a los testimonios de los usuarios, se concluye que el sistema cumple con los requerimientos, y que se puede mejorar. El grupo de usuarios (el equipo de astronomía en el área de cúmulos estelares de la UNAM) queda conforme con el trabajo realizado. Cabe mencionar que parte de los resultados obtenidos del presente trabajo fueron sometidos y publicados en el congreso anual de astronomía que realiza la UNAM, el cual se llevó a cabo en Puebla, Tonanzintla en Marzo del 2006.

VII - Bibliografías y Referencias

- Cabrera-Cano J., Alfaro E.J., & Polvorinos 1990, in *Errors, Bias and Uncertainties in Astronomy*, ed. C. Jashek & F. Murtagh (Cambridge: Cambridge Univ. Press), 101
- Carraro G., Ng Y. K., & Portinari L. 1998, *MNRAS*, 296, 1045
- Chavarría C.K. 1994, *RMxAA*, 29, 137
- Delgado A.J., Alfaro, E., Moitinho A. & Franco J. 1998, *AJ*, 116, 180
- Feinstein A. 1994, *RMxAA*, 29, 141
- Friel E.D. 1995, *A&A*, 33, 381
- Janes K.A. 1979, *ApJ*, 39, 135
- Johnson H. L., Morgan W.W. 1953, *ApJ*, 117, 113
- Kassis M., Janes, K.A., Friel E.D. & Phelps R.L. 1997, *AJ*, 113, 1723
- Loktin A.V. & Matkin N.V. 1994, *A&A Trans.* 4, 153
- Meynet G., Mermilliod J.C. & Maeder A. 1991, *A&A*, 98, 477
- Mermilliod J.C. 1982a, *A&A*, 109, 37
- Mermilliod J.C. 1982b, *A&A*, 109, 48
- Mermilliod J.C. & Mayor M. 1989, *A&A*, 219, 125
- Mermilliod J.C. 1995, in *Information and on-line Data in Astronomy*, eds.
- Moffat A.F.J. & Vogt N. 1973, *A&A*, 23, 317
- Moitinho 1999, Ph.D. Thesis, (Univ. de Granada)
- Phelps R. L. & Janes K.A. 1994, *ApJ*, 90, 31
- Piatti A. E., Clari J.J. & Abadi M.G. 1995, *AJ*, 110, 2813
- Schild R. E. 1983, *PASP* 95, 1021
- Twarog B.A., Ashman K.M. & Anthony-Twarog B.J. 1997, *AJ*, 114, 2556
- Walker, A.R. 1985, *MNRAS*, 214, 45

Sommerville Ian, 2001, Software Engineering 6ta Edicion, Adison-Wesley

Alpert Sherman K. Brown B. Woolf The Design Patterns Smalltalk Companion. Reading, Mass.: Addison-Wesley, 1998.

Pree Wolfgang Design Patterns for Object-Oriented Software Development. Reading, Mass.: Addison-Wesley, 1994.

Buschman Frank R. Meunier Pattern Oriented Software Architecture: A System of Patterns. New York: John Wiley and Sons, 1996.

James W. Cooper Java Design Patterns: A Tutorial. Addison Wesley, 2000.

VIII –Anexos

VIII.1 - Apéndice A: Gráficas de cúmulos

Estudiando el cúmulo Be15

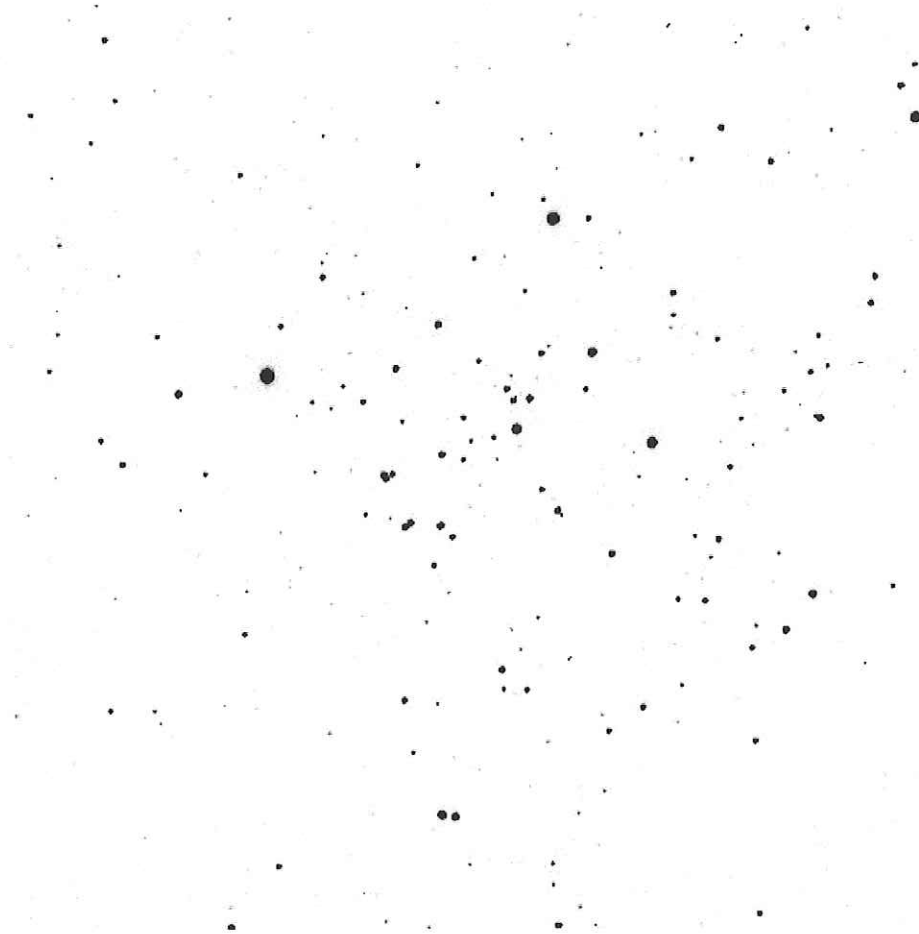


Figura VIII.1.K.1: Cúmulo Be15 tomada por telescopio.

Este es la imagen del cúmulo Be15, tal y como se puede observar en el espacio, de aquí con el CCD se obtendría lo que son los colores base UBVRl de los cuales se pudieran combinar para obtener más gráficas.

A continuación una imágenes sacadas con el sistema SAFE de algunos colores utilizados por los astrónomos.

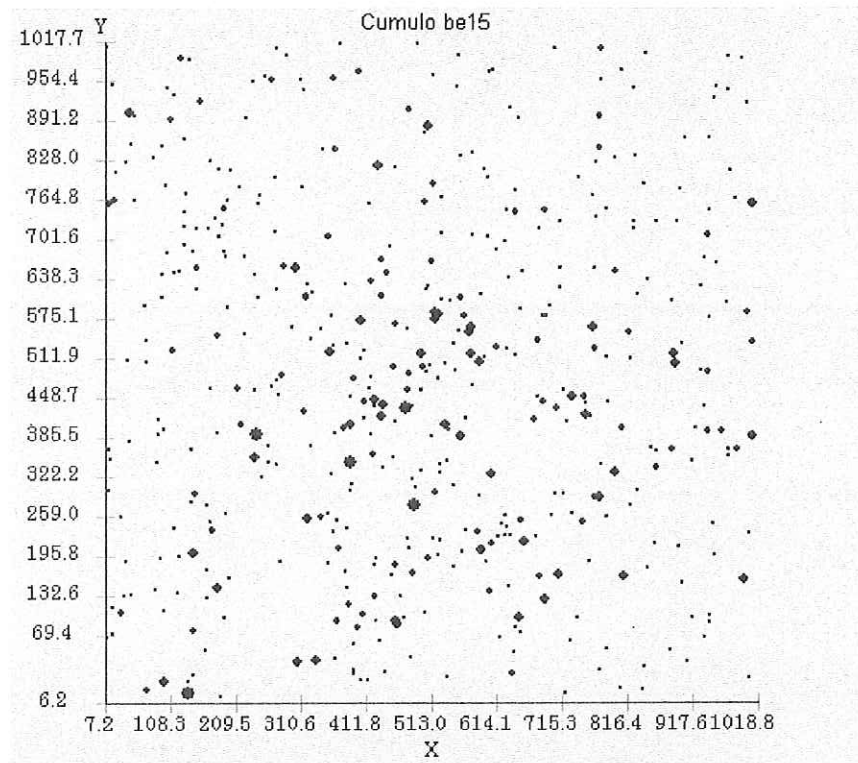


Figura VIII.1.K.1: Cúmulo Be15 posiciones X,Y el tamaño es proporcional al brillo de cada estrella.

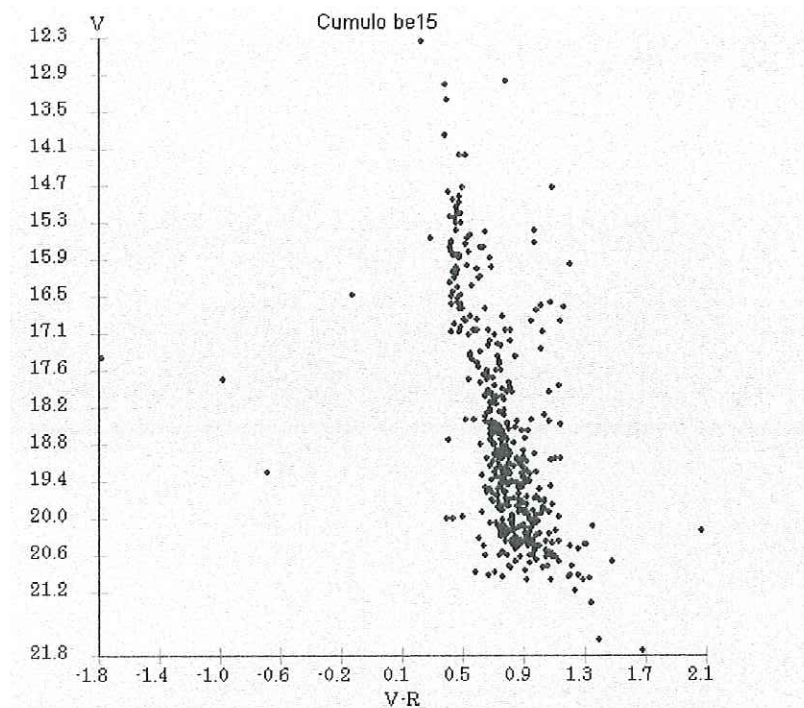


Figura VIII.1.K.1: Diagrama V , $(V - R)$ del cúmulo Be15.

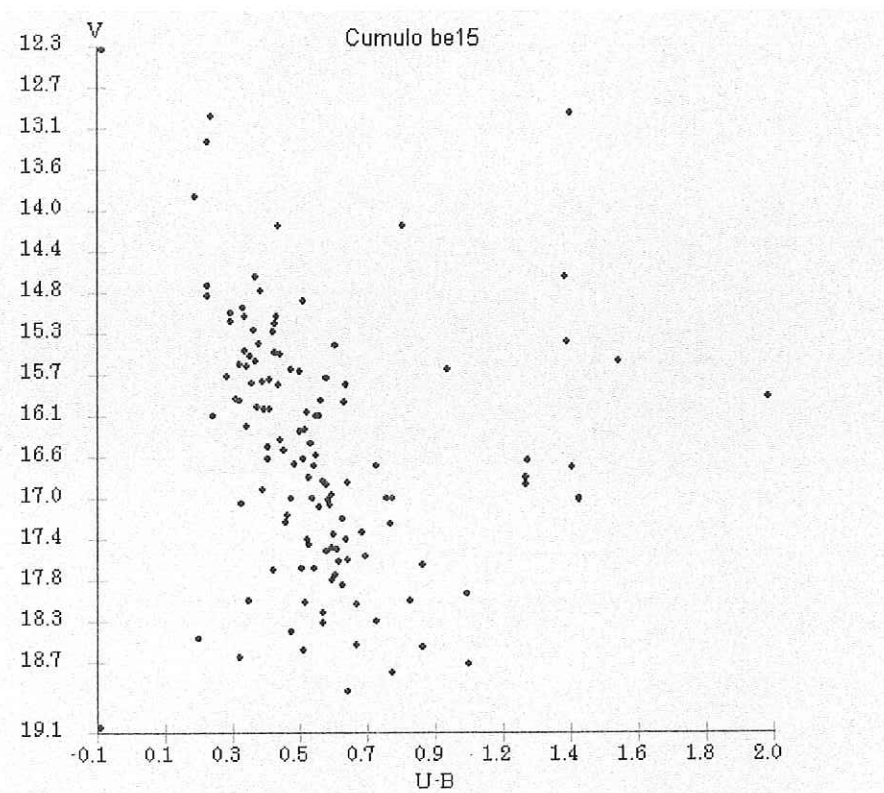


Figura VIII.1.K.1: Diagrama V , $(U - B)$ del cúmulo Be15.

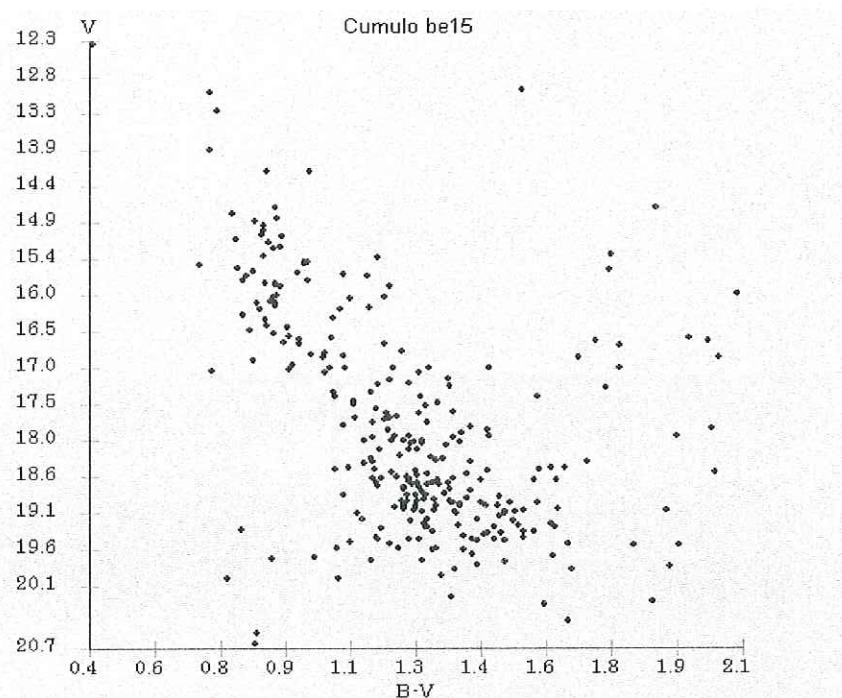


Figura VIII.1.K.1: Diagrama V , $(B - V)$ del cúmulo Be15.

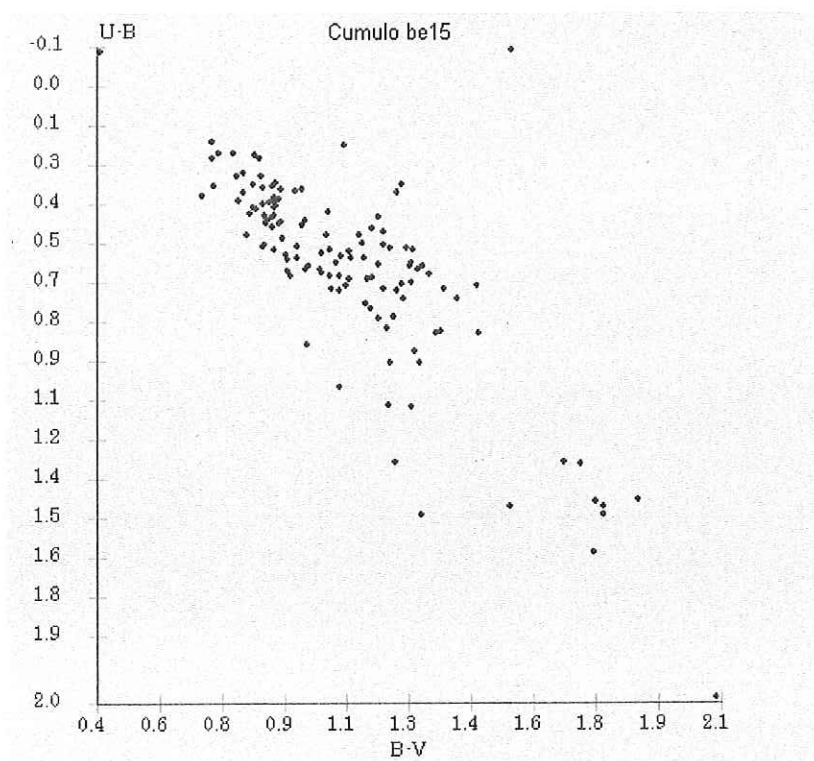


Figura VIII.1.K.1: Diagrama $(U - B)$, $(B - V)$ del cúmulo Be15.

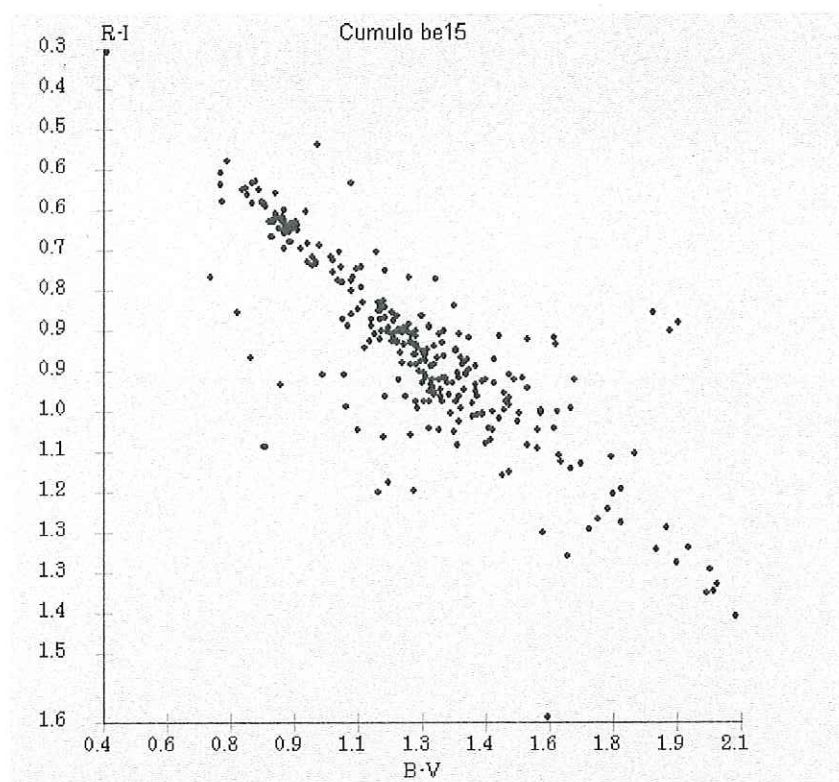


Figura VIII.1.K.1: Diagrama (R - I), (B - V) del cúmulo Be15.

VIII.2 - Apéndice B: Manual de Usuario

Manual de Usuario SAFE

Guía para poder usar, conocer e instalar el sistema para el análisis fotométrico de estrellas (SAFE).

Introducción

En este manual se describen los accesos del teclado, el uso y funciones del sistema y el formato de los archivos que utiliza el sistema SAFE.

El Sistema ha sido construído para el propósito de analizar cúmulos estelares de una forma gráfica, interactiva y editable.

El sistema está diseñado para que cualquier usuario pueda usarlo, configurarlo ya sea a través de la interfaz proporcionada o manualmente y entenderlo. Los menús son sencillos y explicativos, los mensajes del sistema se dan a través de una caja de dialogo.

Atajos para el menú

Abrir	CTRL + A
Salvar Como	CTRL + S
Cerrar	CTRL + X
Eliminar	DEL
Seleccionar	ESPACIO
Agregar Gráfica	F2
Remover Gráfica	CTRL + E
Invertir Selección	CTRL + I
Limpiar Cúmulo	CTRL + L
Inicializa Región	F5

Atajos para el menú (cont.)

Activar Región	F3
Rastrear Estrellas	F4
Zoom	CTRL + Z
Vector Enrojecimiento	CTRL + V
Cambiar Configuración	F6
Cargar Configuración	CTRL + C
Ayuda	F1
Cambiar Pestaña	CTRL + T

Estructura



Figura VIII.2.1: Estructura de directorios

SAFE es el directorio principal del código fuente. Normalmente se encuentra como un archivo JAR que es el formato de Java para ejecutables.

Imágenes es un contenedor de imágenes para el sistema que pueden ser cambiados, pero deben de mantener el mismo nombre y el mismo formato.

Cúmulos es un contenedor de cúmulos donde se pueden guardar los cúmulos y es el directorio por omisión.

Configuraciones es un contenedor de configuraciones donde se pueden tener varias configuraciones y es el directorio por omisión.

Gráficas es un contenedor para las gráficas grabadas en formato de imagen y es el directorio por omisión para guardar gráficas en PNG, JPEG y GIF.

Cómo empezar

La estructura del sistema SAFE está diseñada con un contenedor de cúmulos para no tener que estarlos buscando por diferentes partes y un contenedor de configuraciones, los cuales ya están direccionados automáticamente.

Primero se abre un cúmulo si ya se puso con anterioridad en el contenedor lo encontramos rápidamente, si no hay que buscar la dirección del cúmulo. No existe límite para el número de estrellas, pero entre más grande sea el cúmulo más lento será el sistema.

Después que se habrá el cúmulo, se puede, si así se desea, abrir una configuración previamente salvada o cambiar la actual que es la establecida por el sistema.

Se pueden activar las estrellas de 2 formas:

1. Dando clic a una estrella.
2. Activando la región poligonal.
3. Activando el rastreador.

Después de que se haya elegido una o varias estrellas activas, las se pueden seleccionar, es decir, que no se pierdan de vista aun que se activen otras, e incluso eliminar. Para activar las más fácilmente se puede usar el Rastreador.

Una vez que se haya estudiado el cúmulo se puede guardar (*ver Salvar Como en Archivo*).

Se pueden tener hasta 16 gráficas por pestaña y hasta 20 pestañas, la cual da un total de 320 gráficas interactivas con las cuales se puede estudiar el cúmulo. Se recomienda para cúmulos muy grandes tener hasta sólo 4 gráficas, si se ve lento el uso del sistema.

Menús

Menú Archivo

Archivo	Editar	Graficas	I
Abrir	CTRL+A		
Agregar Pestaña			
Salvar			
Salvar Como	CTRL+S		
Cerrar	CTRL + X		
Salir			

Figura VII.2.A: Menú de archivo.

Abrir: Sirve para abrir un cúmulo que se encuentra en un archivo de texto (*.txt) con cierto formato (*ver Formatos del Sistema*) y despliega la primera gráfica con coordenadas espaciales X, Y del archivo.

Agregar Pestaña: Agrega una pestaña con el mismo cúmulo abierto y despliega la primera gráfica con coordenadas espaciales X, Y del archivo.

Salvar: Guarda el cúmulo abierto anteriormente, sobre escribiéndose en el mismo sin preguntarle al usuario.

Salvar Como: Le pregunta al usuario donde desea guardar este cúmulo y como se va a llamar, si el usuario elige el mismo nombre el sistema preguntara si desea sobre escribirlo.

Cerrar Pestaña: Cierra la pestaña que actualmente está seleccionada.

Salir: Cierra la instancia actual del sistema.

Menú Edición

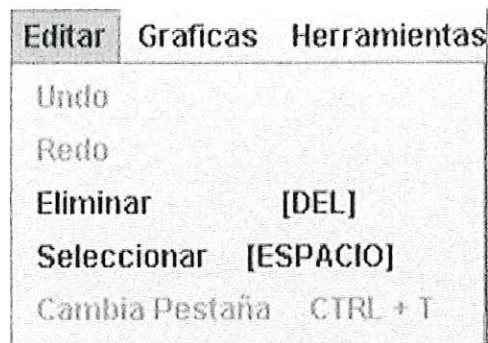


Figura VII.2.B: Menú de edición.

DesHacer: Deshace un cambio hecho al cúmulo de trabajo.

ReHacer: Rehace un cambio hecho al cúmulo de trabajo.

Eliminar: Sirve para borrar las estrellas que están actualmente seleccionadas.

Seleccionar: Sirve para seleccionar las estrellas que están actualmente activas.

Cambia Pestaña: Sirve para cambiar la pestaña.

Menú Herramientas



Figura VII.2.C: Menú de herramientas.

Inicializa región: Restablece la región poligonal a su estado inicial de todas las gráficas. En caso de que alguno de los objetos controladores se haya salido de los bordes o haya demasiados controles.

Activar Región: Sirve para crear una región poligonal con la cual se pueden activar un conjunto de estrellas. Inicialmente son 4 puntos pero si se le da clic con el botón derecho, se pueden agregar hasta 6 puntos más y se pueden mover conforme al usuario le convenga. En caso de estar dentro, dar clic con el botón derecho dentro de un control, este se eliminará.

Rastrear estrellas: Un rastreador de estrellas activa la estrella conforme se va moviendo el ratón, esto sirve cuando se tiene muchas estrellas, se tiene problemas de visualización o problemas al no poder dar clic a una estrella. La estrella no queda activa con esta función, sólo se activa si el ratón está encima, si el ratón pasa encima de otra estrella la anterior se desactivará.

Zoom: Se abre una nueva ventana con la vista del control de zoom, que es un cuadro que aparece en las gráficas, que si se mueve muestra esa porción del cúmulo en proporción al tamaño de la ventana.

Vector de enrojecimiento: Sirve para mostrar el vector de enrojecimiento en una gráfica dependiendo del parámetro de pendiente que se le haya dado a la gráfica dentro de la configuración.

Cambiar Configuración: Sirve para cambiar la configuración sin tener que cargar otra. Los cambios de esta configuración se pueden guardar. Esta nueva configuración sólo es para la pestaña seleccionada. Si no se guarda la primera configuración como una nueva, los cambios solo tendrán efecto durante esa ejecución del sistema.

Cargar Configuración: El usuario puede buscar un archivo de configuración y cargarlo al sistema. El sistema se encargará de rehacer todos los cambios necesarios. Esta nueva configuración sólo es para la pestaña seleccionada.

Menú Gráficas

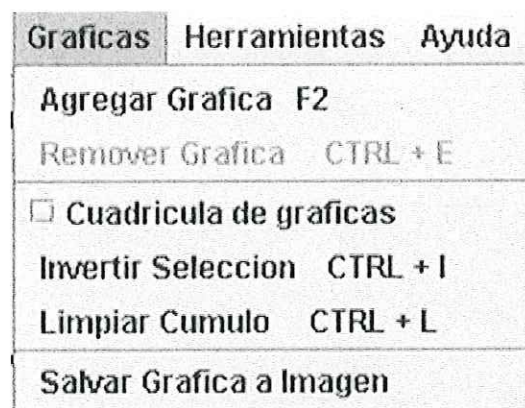


Figura VII.2.D: Menú de gráficas.

Agregar Gráfica: Agrega una gráfica al panel de gráficas para la pestaña seleccionada, tomando en cuenta la configuración actual y desplegando los ejes indicados por tal configuración.

Remover Gráfica: Remueve la última gráfica agregada de la pestaña seleccionada.

Nota: Para poder eliminar una gráfica sin importar el orden, dar clic en el cuadro rojo con la cruz blanca para cerrar esa gráfica; cada gráfica tiene su propio control de cierre.

Cambiar Selección: Hace un cambio entre las estrellas seleccionadas y las que no están seleccionadas, las estrellas seleccionadas se deselectan y las que no están seleccionadas se seleccionan.

Limpiar Cúmulo: Limpia todos los cambios realizados al cúmulo y lo restaura a como fue cargado del archivo.

Menú Configuración

La configuración del sistema consta de 4 partes:

1. – Definir tipo de gráficas: Se despliega una lista de gráficas donde se tiene que elegir una para poder editarla. Para poder editar se proporcionan 2 listas de datos y colores que se pueden elegir, un indicador para invertir un eje y la pendiente para el vector de enrojecimiento para esa gráfica. Los cambios no tomarán efecto para la gráfica editada si no se hace clic en *Aceptar Cambios*. Para poder continuar se debe salvar la configuración.

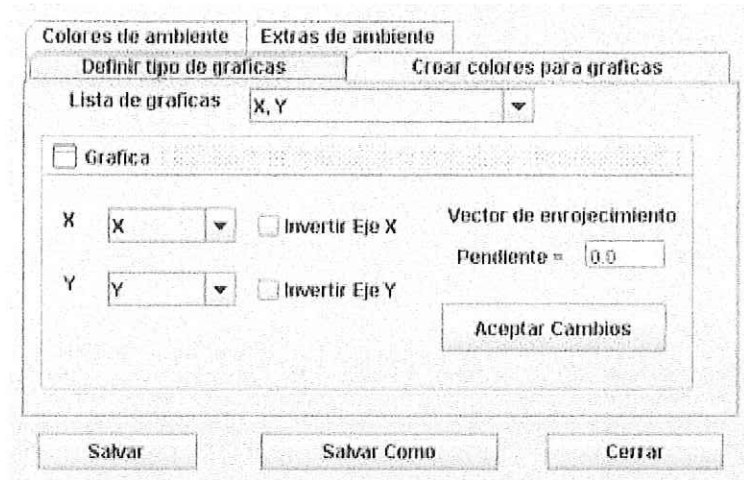


Figura VII.2.E: Menú de configuración, definir tipo de gráficas.

2. – Colores de ambiente: Se pueden cambiar los colores de ambiente del sistema como al usuario le acomode mejor, pero sólo puede ser a través de la interfaz.

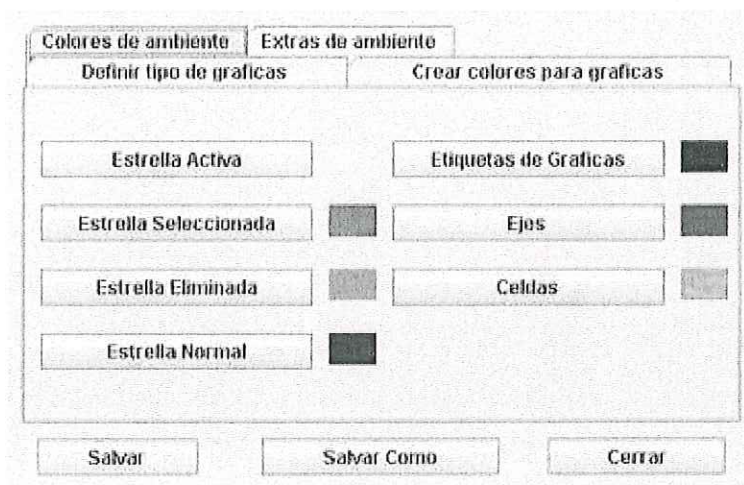


Figura VII.2.F: Menú de configuración, colores de ambiente.

3. – Crear colores para gráficas: Aquí se definen los colores de usuario, que es la mezcla de varios colores leídos por el sistema. Se escribe un nombre para el color (BU0.7), una expresión

$(B-U * 0.7)$ y se hace clic en agregar color. En caso de error también se pueden remover los colores.

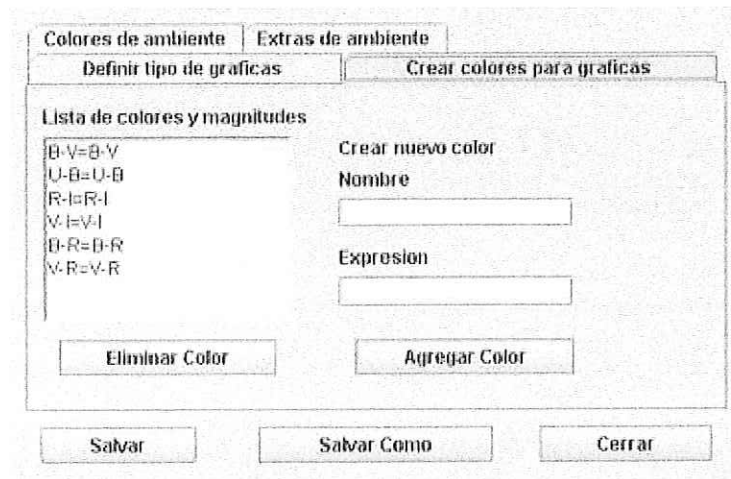


Figura VII.2.G: Menú de configuración, crear colores para gráficas.

4. – Extras de ambiente: Aquí se definen el tipo y tamaño de letra para el sistema y la tabla de datos, el tamaño del control para zoom y la tecla de selección.

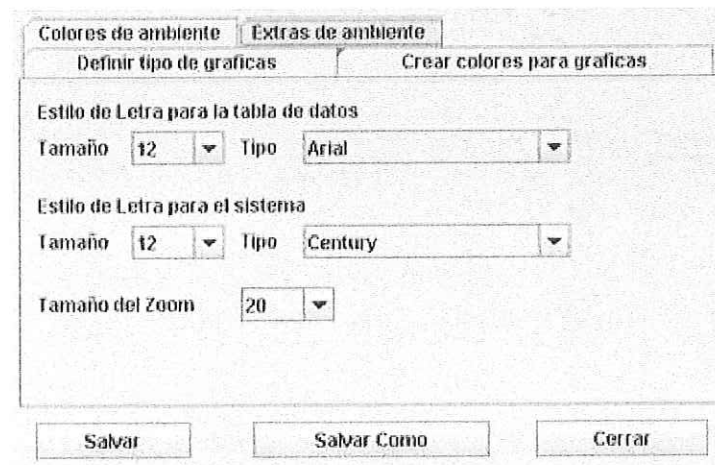


Figura VII.2.H: Menú de configuración, extras de ambiente.

Funciones Matemáticas

Instrucción	Descripción	Uso
sen	Regresa el Seno trigonométrico de un ángulo.	sen(X)
cos	Regresa el Coseno trigonométrico de un ángulo.	cos(X)
tan	Regresa la Tangente trigonométrica de un ángulo.	tan(X)
asen	ArcoSeno (sen-1) De -PI / 2 hasta PI / 2.	asen(X)
acos	ArcoCoseno (cos-1) De -PI / 2 hasta PI / 2.	acos(X)
atan	ArcoTangente (tan-1) De -PI / 2 hasta PI / 2.	atan(X)
exp	Regresa el número elevado a la N potencia.	exp(N)
log	Regresa el logaritmo natural (base e) de N	log(N)
log10	Regresa el logaritmo base 10 de N	log10(N)
raiz2	Regresa el positivo redondeado de la raíz cuadrada de N	raiz2(N)
raiz3	Regresa la raíz cubica de N	raiz3(N)
atan2	Convierte las coord. rectangulares (x,y) a polares (r, teta). Calcula teta por medio del ArcoTangente de y/x	atan2(X,Y)
pow	Regresa el 1er valor elevado al 2do valor = A^B	pow(A,B)
sinh	Regresa el seno hiperbólico que se calcula $(e^x - e^{-x})/2$	sinh(X)
cosh	Regresa el coseno hiperbólico que se calcula $(e^x + e^{-x})/2$	cosh(X)

tanh	Regresa el tangente hiperbólico que se calcula $\sinh(X) / \cosh(X)$	tan(X)
hypot	Regresa la raíz ² (A ² +B ²)	Hypot(A,B)
abs	Regresa el absoluto de X	abs(X)

Formatos del sistema

Cúmulos

La primera línea es una serie de etiquetas que definen los nombres de los datos. Los datos y las etiquetas van separados por espacio.

Cada renglón de los archivos, después de las etiquetas, es una estrella. El símbolo # es obligatorio al inicio del primer renglón y no importa si no lleva la columna de ID, el sistema automáticamente cambiará el formato del archivo.

Nota: Si no se desea tener modificado este archivo crear una copia antes.

Ej:

```
#ID X Y U B V ErrorU ErrorB ErrorV
```

```
0 2.0 3.0 4.2 5.9 6.5 0.2 0.3 0.4
```

```
1 2.0 3.0 4.0 5.3 6.4 0.2 0.3 0.4
```

.

.

Configuraciones

Los comentarios para el sistema llevan doble "/" y abarcan todo un renglón.

El archivo de configuración inicia por las gráficas. La palabra clave para gráficas es "GRAFICA"

+ No. Seguido de "=" y de las coordenada en X, luego si es invertido o no el eje X (true, false),

la coordenada en Y, luego si es invertido el eje Y (true, false), por último el valor de la pendiente para esa gráfica todos los datos separados por coma.

Ej:

GRAFICA1=X,false,Y,false,0.0

GRAFICA2=U-B,false,B-V,true,3.1

GRAFICA16=V,false,B-V,true,0.72

No pueden ser más de 16 gráficas ya que el sistema no lo soporta.

Después siguen los colores de ambiente: estos sólo se agregan por el sistema, si lo está creando manual deberá poner puros ceros y ajustarlos en el sistema o copiar estos valores iniciales por omisión.

Ejemplo:

ColorEstrella=0.0,0.2,0.6

ColorEliminada=0.6,0.6,0.6

ColorSeleccionada=1.0,0.0,0.0

ColorActiva=1.0,1.0,0.0

ColorLineas=0.2,0.2,1.0

ColorCeldas=0.2,0.8,0.0

ColorEtiquetas=0.4,0.0,0.2

A continuación siguen los colores de usuario que deben llevar la palabra clave "ColorCreado" + No. Seguido de "=", continuando con el nombre de la expresión, ":" como separador y luego la expresión.

Ej:

ColorCreado1=U-B:U-B

ColorCreado2=B-V:B-V

ColorCreado3=senR:sen(R)

Por último sigue el ambiente que debe tener el mismo orden, el tipo de letra dependerá del sistema operativo en el que se ejecute.

La tecla de activación por omisión es 32 que es la barra espaciadora.

TeclaActivacion=32

TamanoZoom=20

FontSistema="Century"

TamanoFontSistema=10

FontTabla="Arial"

TamanoFontTabla=10

Instalación en Windows

1) Descargar el JSDK de Java 1.5.0 <http://java.sun.com/javase/downloads/index.jsp>

Descargar - *JDK 5.0 Update 9* para WINDOWS

2) Ejecutar el archivo descargado y seguir las instrucciones de la instalación. Durante la instalación le pedirá donde desea guardar este paquete, usaremos como ejemplo C:/Java

3) Al terminar la instalación se ubica la aplicación donde sea más cómodo.

4) Ahora se puede ejecutar con doble clic el archivo "safe.jar" que se encuentra en el folder principal SAFE-MAIN.

5) Si se desea se puede crear también un atajo al escritorio.

Instalación en Linux

1) Descargar el JSDK de Java 1.5.0 <http://java.sun.com/javase/downloads/index.jsp>

Descargar - *JDK 5.0 Update 9* para LINUX

2) Mover el archivo descargado al directorio que se desea guardar, en este caso se usará de ejemplo:

```
/usr/java/ y el archivo descargado jdk-1_5_0_09-linux-i586-rpm.bin
```

3) Decirle al archivo que sea ejecutable

```
cumulos1:# chmod a+x jdk-1_5_0_09-linux-i586-rpm.bin
```

4) ejecutar el archivo

```
cumulos1:# ./jdk-1_5_0_09-linux-i586-rpm.bin
```

Se visualiza la licencia y al final preguntará si se acepta la licencia, escriba "yes"

```
Do you agree to the above license terms? [yes or no]
```

```
yes
```

```
Unpacking...
```

```
Checksumming...
```

```
0
```

```
0
```

```
Extracting...
```

```
UnZipSFX 5.42 of 14 January 2001, by Info-ZIP (Zip-Bugs@lists.wku.edu).
```

```
inflating: jdk-1_5_0_09-linux-i586.rpm
```

```
Preparing... ##### [100%]
```

```
1:jdk ##### [100%]
```

```
Done.
```

5) Este paso sólo se ejecuta si no se generó ninguna carpeta.

Ejecutar el archivo RPM que se generó

```
cumulos1:# rpm jdk-1_5_0_09-linux-i586.rpm
```

6) Se debió haber creado una carpeta jdk1.5.0_VERSION

7) Poner el directorio SAFE-MAIN en donde sea más cómodo, en este ejemplo se utiliza

```
cumulos1: /usr/SAFE-MAIN/
```

8) Se crea un *script bash*

en un archivo de texto con sólo 2 líneas

```
#!/bin/bash
```

```
/usr/java/jdk1.5.0_09/bin/java/java -jar /usr/SAFE-MAIN/safe.jar
```

y se salva como "safe"

9) Se cambian los permisos del script a ejecutable

```
cumulos1:# chmod a+x safe
```

10) Ahora se puede ejecutar con doble clic el archivo "safe" que se acaba de crear.

Nota: El archivo script "safe" puede estar donde sea, puede ponerse en el escritorio como atajo, las direcciones de java y del SAFE ya están especificadas.

Nota: También se puede ejecutar "safe" desde consola si se tiene instalado java y configurada una variable de ambiente.

```
cumulos1:# java -jar safe.jar
```

Gráficas y uso

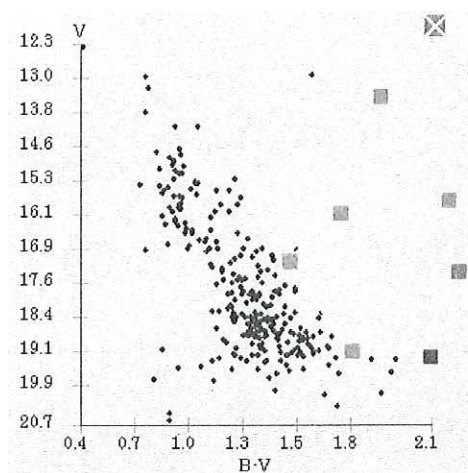


Figura VII.2.I: Uso de la funcionalidad de selección por región poligonal.

El control de la región poligonal inicialmente cuenta con 4 puntos, y se pueden crear hasta 10 puntos para figuras más complejas, es decir 6 o más.

Los de color morado se mueven a través del ratón, el de color rojo indica el fin del polígono y el azul indica el inicio del polígono. Cuando se agrega un nuevo control (cuadro morado) este aparecerá entre los cuadros azul y rojo, es decir entre el principio y el fin del polígono. Si se desea eliminar un control, se pone el ratón encima y se da clic con el botón derecho.

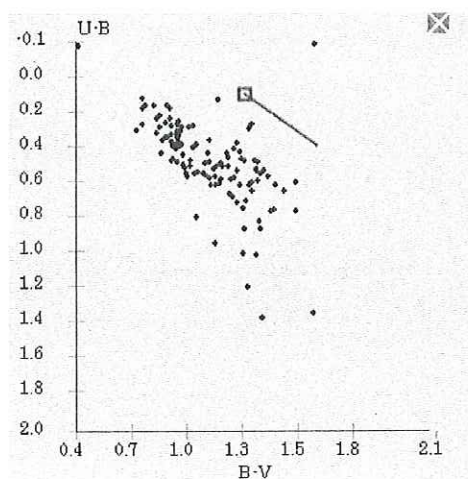


Figura VII.2.J: Uso de la funcionalidad de vector de enrojecimiento.

El vector de enrojecimiento sólo sirve para mostrar la dirección del vector más no la magnitud, todos los vectores tienen la misma magnitud. El vector de enrojecimiento se puede mover de lugar por si estorba la visualización del cúmulo.

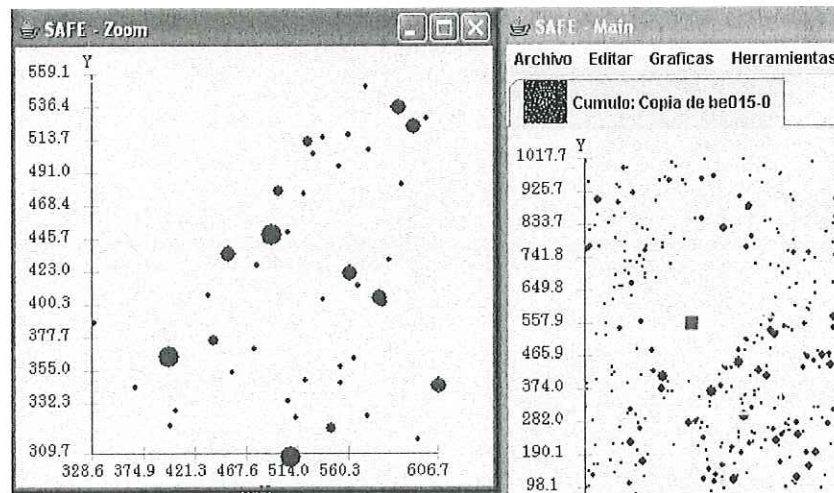


Figura VII.2.K: Uso de la funcionalidad de acercamiento o zoom.

El control para el acercamiento también es movable y es como si se visualizara una lupa en forma de cuadro donde en otra pantalla podemos ver lo que se encuentra dentro de este cuadro de control.

La vista será mejor entre más grande sea la ventana del acercamiento.

VIII.3 - Apéndice C: Diagramas de clases

Diagrama de clases del paquete: Archivos

ControlArchivos
<pre>public Cumulo cumulo public BufferedImage grafica public LecturaArchivo leer public EscrituraArchivo escribir public String AppDirectorio private Dialogo ctrlDialogo private boolean noAceptado private ConfigClass config private int opcion private javax.swing.JFileChooser DialogoArchivos private javax.swing.JFrame jFrame1</pre>
<pre>public ControlArchivos(int tipo, int newOpcion) throws ExcepcionDeArchivos public void ponerTipoDialogo(int tipo) private void initComponents() public void ponerGrafica(BufferedImage grafica) public void ponerConfiguracion(ConfigClass newConfig) public ConfigClass obtenerConfiguracion() public void ponerCumulo(Cumulo nCumulo) public Cumulo obtenerCumulo() private void DialogoArchivosActionPerformed(java.awt.event.ActionEvent evt) private void salvarImagen(BufferedImage g, String f, File file) throws ExcepcionDeArchivos public int mostrarDialogo(Component padre) throws ExcepcionDeArchivos public int Aceptar() public int Cancelar()</pre>

Figura IV.3.B.1: Estructura de la clase **ControlArchivos**.

EscrituraArchivos
<pre>private final int MAX_GRAFICAS public static String direccion private static Dialogo ctrlDialogo</pre>
<pre>public EscrituraArchivo(Dialogo ctrlDialogo) throws ExcepcionDeArchivos public static void ponerTrayectoria(String directorio) public static String obtenerTrayectoria() private static void crearArchivo(String info, int tipo) throws ExcepcionDeArchivos public void salvarArchivo(Cumulo cumulo) throws ExcepcionDeArchivos public void salvarConfiguracion(ConfigClass config) throws ExcepcionDeArchivos</pre>

Figura IV.3.B.2: Estructura de la clase **EscrituraArchivos**.

LecturaArchivos
<pre> public final static int FILE_MAX_SIZE private static String direccion private static char[] buffer private String token private int infoIndex private Cumulo cumulo private int contadorEtiquetas private String[] etiquetas private ConfigClass config private static Dialogo ctrlDialogo private final int FIN_ARCHIVO private final int NUMERO private final int ERROR private final int FDL private final int CADENA private final int DELIMIT </pre>
<pre> public LecturaArchivo(Dialogo newCtrlDialogo) throws ExcepcionDeArchivos public Cumulo obtenerCumuloLeido() public ConfigClass obtenerConfiguracionLeida() public static void ponerTrayectoria(String path) public static String obtenerTrayectoria() public static int cargarArchivo() throws ExcepcionDeArchivos public ConfigClass abrirConfiguracion() private int saltaComentarios() public void abrirArchivo() throws ExcepcionDeArchivos private int obtenerToken() private boolean esDelimitador(char c) private boolean esEspacioTab(char c) private boolean esFinDeLinea(char c) private boolean esFinArchivo(char caracter) </pre>

Figura IV.3.B.3: Estructura de la clase **LecturaArchivos**.

Utiles
<pre> public static String AppDirectorio public final static String ExtTexto public final static String ExtConfig public final static String ExtGraficaJPG public final static String ExtGraficaGIF public final static String ExtGraficaPNG public final static int SALVAR public final static int ABRIR public final static int CUMULO public final static int CONFIG public final static int GRAFICA </pre>
<pre> public static String obtenerExtension(File f) protected static ImageIcon crearImagenIcono(String nombreImagen) </pre>

Figura IV.3.B.4: Estructura de la clase **Utiles**.

VistaArchivos
<pre> public String getName(File f) public String getDescription(File f) public Boolean isTraversable(File f) public String getTypeDescription(File f) public Icon getIcon(File f) </pre>

Figura IV.3.B.5: Estructura de la clase **VistaArchivos**.

FiltroConfiguracion
<pre> public boolean accept(File f) public String getDescription() </pre>

Figura IV.3.B.6: Estructura de la clase **FiltroConfiguracion**.

FiltroArchivos
<pre> public boolean accept(File f) public String getDescription() </pre>

Figura IV.3.B.7: Estructura de la clase **FiltroArchivos**.

FiltroGraficasJPG
<pre> public boolean accept(File f) public String getDescription() </pre>

Figura IV.3.B.8: Estructura de la clase **FiltroGraricasJPG**.

FiltroGraficasPNG
<pre> public boolean accept(File f) public String getDescription() </pre>

Figura IV.3.B.9: Estructura de la clase **FiltroGraricasPNG**.

FiltroGraficasGIF
<pre>public boolean accept(File f) public String getDescription()</pre>

Figura IV.3.B.10: Estructura de la clase **FiltroGraricasGIF**.

ExcepcionDeArchivos
<pre>private String errStr</pre>
<pre>public ExcepcionDeArchivos(String str) public String toString()</pre>

Figura IV.3.B.11: Estructura de la clase **ExcepcionDeArchivos**.

Diagrama de clases del paquete: Astronomia

Cumulo
<pre>public final double VALOR_NULO public Vector estrellas public Vector etiquetas public String nombre public String etiquetaX public String etiquetaY public Dialogo ctrlDialogo</pre>
<pre>public Cumulo() public Cumulo(Cumulo copia) public void agregarEstrella(Estrella estrella) public void eliminarEstrella(int indice) throws ExcepcionDeAstronomia public void deSeleccionarEstrella(int indice) throws ExcepcionDeAstronomia public void eliminarEstrellasSeleccionadas() public void seleccionarEstrella(int indice) throws ExcepcionDeAstronomia public void activarEstrella(int indice) throws ExcepcionDeAstronomia public void seleccionaConjuntoEstrellas() public void deSeleccionarTodo() public void restauraTodo() public void desActivaTodo() public Estrella buscarEstrella(int x, int y) public double buscaValorMinimo(boolean inX) public double buscaValorMaximo(boolean inX) public void cambiaGrafica(String cambioX, String cambioY, ConfigClass config) public void cambiaPuntosADibujo(Limites limites) private String obtenerExpresion(String color, ConfigClass config) public void activaConjuntoEstrellas(Polygon poligono) public void activaConjuntoEstrellas(Rectangle r)</pre>

Figura IV.3.C.1: Estructura de la clase **Cumulo**.

Estrella
<pre> public final int TAMANIO_NORMAL public int nombre public Point2D.Double origen public Point2D.Double dibujo public Vector colores public Vector errores public boolean seleccionada public boolean activa public boolean eliminada public int tamano </pre>
<pre> public Estrella() public Estrella(Estrella copia) throws ExcepcionDeAstronomia public void eliminar() public void restaurar() public void seleccionar(boolean bandera) public void activar(boolean bandera) </pre>

Figura IV.3.C.2: Estructura de la clase **Estrella**.

ExcepcionDeAstronomia
<pre> private String errStr </pre>
<pre> public ExcepcionDeAstronomia(String str) public String toString() </pre>

Figura IV.3.C.3: Estructura de la clase **ExcepcionDeAstronomia**.

Diagrama de clases del paquete: Configuracion

Configuracion
<pre> private final int MAX_TAB private Dialogo ctrlDialogo private ConfigClass config private javax.swing.JButton cmdCancelar private javax.swing.JButton cmdSalvar private javax.swing.JButton cmdSalvarComo private javax.swing.JTabbedPane tabPanel </pre>
<pre> public Configuracion(ConfigClass newConfig, Vector etiquetas, Dialogo ctrlDialogo) public void salvar() public void salvarComo() public ConfigClass obtenerConfiguracion() private void initComponents() private void cmdSalvarComoActionPerformed(java.awt.event.ActionEvent evt) private void cmdCancelarActionPerformed(java.awt.event.ActionEvent evt) private void cmdSalvarActionPerformed(java.awt.event.ActionEvent evt) </pre>

Figura IV.3.D.1: Estructura de la clase **Configuracion**.

ConfigClass
<pre>private final int MAX_ESPACIOS private final int MAX_COLORES_USR public String listaPreferidaX[] public String listaPreferidaY[] public String coloresDeUsuario[] public String expresiones[] public int cantidadColores public Color fondo public Color linea public Color celdasDeCuadrícula public Color colorEstrella public Color estrellaSeleccionada public Color etiquetas public Color estrellaEliminada public Color estrellaActiva public int teclaActivacion public String directorio</pre>
<pre>public ConfigClass()</pre>

Figura IV.3.D.2: Estructura de la clase **ConfigClass**.

DialogoColores
<pre>private javax.swing.JColorChooser escogerColor</pre>
<pre>public DialogoColores() private void initComponents() public Color mostrarDialogo(Component padre, Color colorActual)</pre>

Figura IV.3.D.3: Estructura de la clase **DialogoColores**.

ColorConfig
<pre>private Dialogo ctrlDialogo private javax.swing.JToggleButton agregarColor private javax.swing.JToggleButton eliminarColor private javax.swing.JTextField expresionColor private javax.swing.JLabel jLabel1 private javax.swing.JLabel jLabel2 private javax.swing.JLabel jLabel6 private javax.swing.JLabel jLabel7 private java.awt.List listaColores private javax.swing.JTextField nombreColor</pre>
<pre>public ColorConfig(Dialogo newDialogo, ConfigClass newConfig) private void initComponents() private void eliminarColorActionPerformed(java.awt.event.ActionEvent evt) private void agregarColorActionPerformed(java.awt.event.ActionEvent evt) public java.awt.List obtenerLista()</pre>

Figura IV.3.D.5: Estructura de la clase **ColorConfig**.

ColorAmbienteConfig
<pre> private ConfigClass config private Dialogo ctrlDialogo private javax.swing.JButton botonActiva private javax.swing.JButton botonCeldas private javax.swing.JButton botonEliminada private javax.swing.JButton botonEtiqueta private javax.swing.JButton botonFondo private javax.swing.JButton botonLineas private javax.swing.JButton botonNormal private javax.swing.JButton botonSeleccionada private javax.swing.JPanel colorActiva private javax.swing.JPanel colorCeldas private javax.swing.JPanel colorEliminada private javax.swing.JPanel colorEtiquetas private javax.swing.JPanel colorFondo private javax.swing.JPanel colorLineas private javax.swing.JPanel colorNormal private javax.swing.JPanel colorSeleccionada </pre>
<pre> public ColorAmbienteConfig(Dialogo newCtrlDialogo, ConfigClass newConfig) private void initComponents() private void botonCeldasActionPerformed(java.awt.event.ActionEvent evt) private void botonLineasActionPerformed(java.awt.event.ActionEvent evt) private void botonFondoActionPerformed(java.awt.event.ActionEvent evt) private void botonEtiquetaActionPerformed(java.awt.event.ActionEvent evt) private void botonNormalActionPerformed(java.awt.event.ActionEvent evt) private void botonEliminadaActionPerformed(java.awt.event.ActionEvent evt) private void botonSeleccionadaActionPerformed(java.awt.event.ActionEvent evt) private void botonActivaActionPerformed(java.awt.event.ActionEvent evt) public ConfigClass obtenerConfiguracion() </pre>

Figura IV.3.D.4: Estructura de la clase **ColorAmbienteConfig**.

GraficasConfig
<pre> public final int MAX_ESPACIOS private Dialogo ctrlDialogo private javax.swing.JToggleButton agregarX private javax.swing.JToggleButton agregarY private javax.swing.JTextField colorX private java.awt.List graficasX private java.awt.List graficasY private javax.swing.JLabel jLabel1 private javax.swing.JLabel jLabel2 private javax.swing.JLabel jLabel6 private javax.swing.JLabel jLabel7 private java.awt.List listaColores </pre>
<pre> public GraficasConfig(Dialogo newDialogo, ConfigClass newConfig, Vector etiquetas) private void initComponents() public java.awt.List obtenerListaY() private void agregarXActionPerformed(java.awt.event.ActionEvent evt) private void listaColoresActionPerformed(java.awt.event.ActionEvent evt) private void graficasYActionPerformed(java.awt.event.ActionEvent evt) private void graficasXActionPerformed(java.awt.event.ActionEvent evt) private void agregarYActionPerformed(java.awt.event.ActionEvent evt) </pre>

Figura IV.3.D.6: Estructura de la clase **GraficasConfig**.

DialogoCfg
Configuracion cfg
<pre> public DialogoCfg(JFrame padre, ConfigClass config, Vector etiquetas, Dialogo ctrlDialogo) public ConfigClass obtenerCofiguracion() public void muestra() public void destruir() </pre>

Figura IV.3.D.7: Estructura de la clase **DialogoCfg**.

ExcepcionDeConfiguracion
private String errStr
<pre> public ExcepcionDeConfiguracion(String str) public String toString() </pre>

Figura IV.3.D.8: Estructura de la clase **ExcepcionDeConfiguracion**.

Diagrama de clases del paquete: EventosDispositivos

EscuchadorEventos
<pre> private EventosRaton evRaton private EventosTeclado evTeclado private Limites limites </pre>
<pre> public EscuchadorEventos(Component component) public void EventosDeTeclado(Component component, Observer observador) public void EventosDeRaton(Component component, Observer observador) public EventosRaton obtenerEventoRaton() public EventosTeclado obtenerEventoTeclado() public void componentResized(ComponentEvent e) public void componentMoved(ComponentEvent e) public void componentShown(ComponentEvent e) public void componentHidden(ComponentEvent e) </pre>

Figura IV.3.E.1: Estructura de la clase **EscuchadorEventos**.

NotificacionTeclado
public final int NINGUNO public final int ABRIR public final int SALVAR public final int CERRAR_PANEL public final int BORRAR public final int SELECCIONAR public final int AGREGA_GRAFICA public final int INVIERTE_SELECCION public final int LIMPIAR public final int REMOVER_GRAFICA public final int INICIALIZA_REGION public final int REGION public final int RASTREAR public final int ZOOM public final int CAMBIA_CFG public final int CARGA_CFG public final int MOSTRAR_AYUDA public final int CAMBIO_FOCUS public int tipoEvento
public NotificacionTeclado() public NotificacionTeclado(int nuevoEvento)

Figura IV.3.E.2: Estructura de la clase **NotificacionTeclado**.

EventosTeclado
public NotificacionTeclado notificador public boolean ctrlAplastado
public EventosTeclado(Component component, Observer observador) public void keyTyped(KeyEvent e) public void keyPressed(KeyEvent e) public void keyReleased(KeyEvent e) public void notificaCambios(NotificacionTeclado n)

Figura IV.3.E.3: Estructura de la clase **EventosTeclado**.

ExcepcionDeEventos
private String errStr
public ExcepcionDeEventos(String str) public String toString()

Figura IV.3.E.4: Estructura de la clase **ExcepcionDeEventos**.

EventosRaton
<pre> public final int NINGUNO public final int DESSELECCIONAR public final int TABLA public final int SELECCIONAR public final int REGION public final int ZOOM public RegionPoligonal region public ControlZoom regionZ public int tipoEvento NINGUNO private Espacio espacio private Limites limites </pre>
<pre> public EventosRaton(Component componente, Observer observador) public void mousePressed(MouseEvent e) public void mouseDragged(MouseEvent e) public void mouseReleased(MouseEvent e) public void mouseMoved(MouseEvent e) public void mouseClicked(MouseEvent e) public void mouseExited(MouseEvent e) public void mouseEntered(MouseEvent e) public void updateLocation(MouseEvent e) public void inicializaRegion() public void notificaCambios() </pre>

Figura IV.3.E.5: Estructura de la clase **EventosRaton**.

Diagrama de clases del paquete: Graficas

Limites
<pre> public int izquierdo public int derecho public int inferior public int superior public int distanciaX public int distanciaY public double maxX public double maxY public double minX public double minY </pre>
<pre> public Limites() public void calculaLimites(Cumulo cumulo, JPanel panel) </pre>

Figura IV.3.F.1: Estructura de la clase **Limites**.

Espacio
<pre> public int indice public Limites limites public ConfigClass configuracion public Cumulo cumuloEstelar public RegionPoligonal regionP public ControlZoom regionZ public boolean RegionActivada public boolean ZoomActivado public boolean RastreadorActivado public EscuchadorEventos escuchador </pre>
<pre> public Espacio() public Espacio(Cumulo cumulo, int largo, int alto, int posicion, Observer observador) public void paintComponent(Graphics g) public void plot(Graphics g) public void pintaCumulo(Graphics g) public void pintaLineas(int largoX, int largoY, String nombreX, String nombreY, Graphics g) public void pintaRegion(Graphics g) public void pintaZoom(Graphics g) </pre>

Figura IV.3.F.2: Estructura de la clase **Espacio**.

ExcepcionDeGraficas
<pre> private String errStr </pre>
<pre> public ExcepcionDeGraficas(String str) public String toString() </pre>

Figura IV.3.F.3: Estructura de la clase **Espacio**.

ControlZoom
<pre> public final int TAMANIO_CTRL public int x, y public Rectangle area, ctrlMove, ctrlSize, rect public Rectangle zoomArea public Point2D.Double point private boolean firstTime private boolean pressOut </pre>
<pre> public ControlZoom() public ControlZoom(ControlZoom nuevoControl) public boolean botonAplastado() public boolean esPrimeraVez() public void ponerArea(Dimension componentSize) public void botonAplastado(boolean bandera) public void esPrimeraVez(boolean bandera) public void checkPoint() </pre>

Figura IV.3.F.4: Estructura de la clase **ControlZoom**.

RegionPoligonal
public final int MAX_CUADROS public final int TAMANIO_CTRL public int x, y public Rectangle area, ctrls[], rect public int cantidadCuadros public Polygon poligono public Point2D.Double point private boolean firstTime private boolean pressOut
public RegionPoligonal() public RegionPoligonal(RegionPoligonal nuevaRegion) public boolean crearControl(int x, int y) public boolean botonAplastado() public boolean esPrimeraVez() public void ponerArea(Dimension componentSize) public void botonAplastado(boolean bandera) public void esPrimeraVez(boolean bandera) public void checkPoint() public boolean eliminaControl(int tx, int ty) public Rectangle buscaControl()

Figura IV.3.F.5: Estructura de la clase **RegionPoligonal**.

Diagrama de clases del paquete: Interfaz

SplashScreen
private Timer reloj private javax.swing.JLabel descripcion private javax.swing.JLabel fecha private javax.swing.JPanel jPanel1 private javax.swing.JLabel legal private javax.swing.JLabel legal1 private javax.swing.JLabel legal2 private java.awt.Canvas logo private javax.swing.JLabel programador private javax.swing.JLabel titulo private javax.swing.JLabel version
public SplashScreen() public void mostrarSplash(int retardo) private void initComponents() public static void main(String args[])

Figura IV.3.G.1: Estructura de la clase **SplashScreen**.

PanelGrafico
<pre> public final int MAX_ESPACIOS public final int MAX_RENGLONES public final int MAX_COLUMNAS public final int T1X1 public final int T1X2 public final int T2X2 public final int T2X3 public final int T3X3 public final int T4X4 public Vector espacios public Cumulo cumuloEstelar public ConfigClass configuracion private int tipoDespliege public UndoRedo undoRedo private int columnas private int renglones private boolean deCumulo </pre>
<pre> public PanelGrafico() public void inicializaPanel(Cumulo cumulo) public void paintComponent(Graphics g) public void ponerTipoDespliege(GridLayout layout) public void ponerTipoDespliege(int nuevoTipoDespliege) public boolean removerEspacio() public boolean crearEspacioNuevo(Cumulo cumulo) public void update(Observable o, Object arg) public void invertirSeleccion() public void limpiaCumulo() public void seleccionarEstrellasActivadas() public void eliminarEstrellasSeleccionadas() public void refrescar() public void cambiaConfiguracion(ConfigClass newConfig) </pre>

Figura IV.3.G.2: Estructura de la clase **PanelGrafico**.

PanelDeDatos
<pre> public TablaDeDatos tablaInterna </pre>
<pre> public PanelDeDatos (TablaDeDatos tabla) public void cerrarPanel() private void initComponents() </pre>

Figura IV.3.G.4: Estructura de la clase **PanelDeDatos**.

ExcepcionDeInterfaz
<pre> private String errStr </pre>
<pre> public ExcepcionDeInterfaz(String str) public String toString() </pre>

Figura IV.3.G.5: Estructura de la clase **ExcepcionDeInterfaz**.

Main
<pre> private ManejadorTabs tabPanel private ControlArchivos controlArchivos private Vector etiquetas private EscuchadorEventos escuchador private String ultimaDireccion private javax.swing.JMenuBar barraMenu private javax.swing.JMenuItem jmnuAbout private javax.swing.JMenuItem jmnuAbrirCumulo private javax.swing.JMenuItem jmnuAgregarGrafica private javax.swing.JMenuItem jmnuAgregarTab private javax.swing.JMenuItem jmnuAyuda private javax.swing.JMenuItem jmnuCambioSeleccion private javax.swing.JMenuItem jmnuCargarConfig private javax.swing.JMenuItem jmnuCerrar private javax.swing.JMenuItem jmnuConfig private javax.swing.JMenuItem jmnuDelete private javax.swing.JMenuItem jmnuEditar private javax.swing.JMenuItem jmnuGraficas private javax.swing.JMenuItem jmnuHelp private javax.swing.JMenuItem jmnuHerramientas private javax.swing.JMenuItem jmnuIniciaRegion private javax.swing.JMenuItem jmnuLimpiaCumulo private javax.swing.JCheckBoxMenuItem jmnuRastreo private javax.swing.JMenuItem jmnuReDo private javax.swing.JCheckBoxMenuItem jmnuRegion private javax.swing.JMenuItem jmnuRemoverGrafica private javax.swing.JMenuItem jmnuSalir private javax.swing.JMenuItem jmnuSalvar private javax.swing.JMenuItem jmnuSalvarComo private javax.swing.JMenuItem jmnuSalvarGrafica private javax.swing.JMenuItem jmnuSeleccionar private javax.swing.JMenuItem jmnuTabChanger private javax.swing.JMenuItem jmnuUndo private javax.swing.JCheckBoxMenuItem jmnuZoom private javax.swing.JSeparator lineSep2 private javax.swing.JSeparator lineSep3 private javax.swing.JSeparator lineSep4 private javax.swing.JSeparator lineSep5 private javax.swing.JSeparator lineSep6 </pre>

Figura IV.3.G.3.1: Estructura de la clase Main (parte 1, declaracion de variables).


```

public Main()
private void initComponents()
private void jmnuReDoActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuUndoActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuSalvarGraficaActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuTabChangerActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuRastreoActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuAboutActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuHelpActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuIniciaRegionActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuLimpiaCumuloActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuSeleccionarActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuDeleteActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuZoomActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuRegionActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuCargarConfigActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuConfigActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuCambioSeleccionActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuRemoverGraficaActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuSalirActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuCerrarActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuSalvarComoActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuSalvarActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuAgregarTabActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuAgregarGraficaActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuAbrirCumuloActionPerformed(java.awt.event.ActionEvent evt)
private void jmnuArchivoActionPerformed(java.awt.event.ActionEvent evt)
public static void main(String args[])
public void rastreoActivo(boolean f)
public void zoomActiva(boolean f)
public void regionActiva(boolean f)
public void cumuloAbierto(boolean bandera)
public void salvarGrafica()
public void accionAbrir()
public void removerGrafica()
public void agregarGrafica()
public void invertirSeleccion()
public void activaZoom()
public void activaRastreador()
public void activaRegion()
public void iniciaRegion()
public void seleccionarEstrellas()
public void accionSalvar()
public void accionSalvarComo()
public void accionAbrirConfig()
public void mostrarAyuda()
public void acercaDe()
public void limpiarCumulo()
public void borrarEstrellas()
public void cerrarPanel()
public void cambiaConfiguracion()
public void cambiarFocus()
public void update(Observable o, Object arg)
public void accionUndo() throws ExcepcionDeInterfaz
public void accionRedo() throws ExcepcionDeInterfaz

```

Figura IV.3.G.3.2: Estructura de la clase **Main** (parte 2, declaracion de métodos).

Dialogo
<pre>private JFrame ventana public final int SIOK public final int NO public final int CANCELAR public final int DESCONOCIDO</pre>
<pre>public Dialogo() public Dialogo(JFrame ventana) public void muestraError(String msg) public void muestraAdvertencia(String msg) public void muestraInformacion(String msg) public void muestraNormal(String msg) public void muestra(String msg) public int muestra_Si_No(String msg) public int muestra_Si_No_Cancelar(String msg) public int muestra_OK_Cancelar(String msg)</pre>

Figura IV.3.G.6: Estructura de la clase **Dialogo**.

ManejadorTabs
<pre>public final int MAX_TABS public final int EXTRA_COLS public final int INDICADORES public Cumulo cumuloEstelar public Dialogo ctrlDialogo public PanelDeDatos panelDatos private ImageIcon iconoCumulo</pre>
<pre>public ManejadorTabs(JFrame padre) protected static ImageIcon creaIconoImagen(String directorio) public void paintComponent(Graphics g) public boolean agregarTab() throws ExcepcionDeInterfaz public boolean removerTab() throws ExcepcionDeInterfaz public void crearVentanaTabla() public void refrescaTabla(Cumulo cumulo) throws ExcepcionDeInterfaz public void update(Observable o, Object arg) private void manejaCambios(Observable o, Object arg) throws ExcepcionDeInterfaz</pre>

Figura IV.3.G.7: Estructura de la clase **ManejadorTabs**.

DialogoAcercaDe
<pre>FormaAcercaDe fad private JButton cmdSalvar</pre>
<pre>public DialogoAcercaDe(JFrame padre) public void muestra() public void destruir()</pre>

Figura IV.3.G.8: Estructura de la clase **DialogoAcercaDe**.

DialogoAyuda
FormaAyuda fa private JButton cmdSalvar
public DialogoAyuda(JFrame padre) public void muestra() public void destruir()

Figura IV.3.G.9: Estructura de la clase **DialogoAyuda**.

FormaAcercaDe
private javax.swing.JButton cmdOk private javax.swing.JLabel jLabel1 private javax.swing.JLabel jLabel2 private javax.swing.JLabel jLabel3 private javax.swing.JLabel jLabel4 private javax.swing.JLabel jLabel5 private javax.swing.JLabel jLabel6 private javax.swing.JLabel jLabel7 private javax.swing.JLabel jLabel8
public FormaAcercaDe() private void initComponents() private void cmdOkActionPerformed(java.awt.event.ActionEvent evt)

Figura IV.3.G.10: Estructura de la clase **FormaAcercaDe**.

FormaAyuda
public Vector ayuda public Vector titulos public int indice private javax.swing.JButton cmdNext private javax.swing.JButton cmdOK private javax.swing.JButton cmdPrev private javax.swing.JTextArea textoAyuda
public FormaAyuda() private void construyeAyuda() private void initComponents() private void cmdNextActionPerformed(java.awt.event.ActionEvent evt) private void cmdPrevActionPerformed(java.awt.event.ActionEvent evt) private void cmdOKActionPerformed(java.awt.event.ActionEvent evt)

Figura IV.3.G.11: Estructura de la clase **FormaAyuda**.

Diagrama de clases del paquete: Procesos

DetectorDeSO
<pre> public String soActual public String formatoFinDeLinea public String separadorCarpetas public String direccionActual </pre>
<pre> public DetectorDeSO() public void detectaSistemaOperativo() throws ExcepcionDeProcesos </pre>

Figura IV.3.H.1: Estructura de la clase **DetectorDeSO**.

ExcepcionDeProcesos
<pre> private String errStr </pre>
<pre> public ExcepcionDeProcesos(String str) public String toString() </pre>

Figura IV.3.H.2: Estructura de la clase **ExcepcionDeProcesos**.

SalvadorImágenes
<pre> public Cumulo cumuloEstelar public ConfigClass config private Dialogo ctrlDialogo private javax.swing.JLabel Etiqueta private javax.swing.JButton OkSalvar private javax.swing.JButton cerrarSalvarImg private javax.swing.JPanel espacioImagen private javax.swing.JComboBox grafica </pre>
<pre> public SalvadorImágenes(ConfigClass cfg, Cumulo cumulo, Dialogo ctrlDialogo, int cantidadEspacios) private void cambiaEspacio() private void initComponents() private void graficaActionPerformed(java.awt.event.ActionEvent evt) private void cerrarSalvarImgActionPerformed(java.awt.event.ActionEvent evt) private void OkSalvarActionPerformed(java.awt.event.ActionEvent evt) </pre>

Figura IV.3.H.4: Estructura de la clase **SalvarImágenes**.

Evaluador
<pre> private final double VALOR_NULO private int indice private char[] expresion private String token private Dialogo ctrlDialogo private String[] nombreVar private double[] valorVar public final int SINTAXIS public final int FALTA_PARENTESIS public final int NO_EXPRESION public final int DIV_POR_ZERO public final int IGUAL_ESPERADO public final int NO_VARIABLE public final int COMA_ESPERADA public TipoToken tToken private Comando cmd private enum TipoToken Comando tablaCmd[] </pre>
<pre> public Evaluador(String exp, Vector nm, double[] valor, Dialogo ctrlDialogo) public double evaluar() throws ExcepcionDeProcesos private double evalExp2() throws ExcepcionDeProcesos private double evalExp3() throws ExcepcionDeProcesos private double evalExp4() throws ExcepcionDeProcesos private double evalExp5() throws ExcepcionDeProcesos private double evalExp6() throws ExcepcionDeProcesos private double atom() throws ExcepcionDeProcesos private double buscaVariable(String nombre) private TipoToken obtenerToken() private boolean validaNumero() private int buscaComando(String cmd) private boolean esComando(String cmd) private boolean esDelimitador(char c) private boolean esEspacioTab(char c) private boolean esFinDeLinea(char c) private boolean esFinExpresion() private void manejaError(int error) throws ExcepcionDeProcesos </pre>

Figura IV.3.H.3: Estructura de la clase **Evaluador**.

Comando
<pre> public String comando public int comandoTok public final int SEN public final int COS public final int TAN public final int ASEN public final int ACOS public final int ATAN public final int SENH public final int COSH public final int TANH public final int SQUAREROOT public final int RAIZ public final int CUBEROOT public final int ATAN2 public final int POW public final int EXP public final int LOG public final int LOG10 public final int ABS public final int HYPOT </pre>
<pre> public Comando(String str, int t) </pre>

Figura IV.3.H.5: Estructura de la clase **Comando**.

UndoRedo
private ArrayList buffer private int index
public boolean canUndo() public boolean canRedo() public boolean atTop() public int count() public void PushCurrentState(Memento mem) public void Do(Memento mem) public Memento Undo() throws ExcepcionDeProcesos public Memento Redo() public void Flush()

Figura IV.3.H.6: Estructura de la clase **UndoRedo**.

DialogoImagen
public Vector espacios private SalvadorImágenes salvadorImgs private JButton cmdSalvar
public DialogoImagen(JFrame padre, ConfigClass config, Cumulo cumulo, Dialogo ctrlDialogo, int cantidadEspacios) public void muestra() public void destruir()

Figura IV.3.H.7: Estructura de la clase **DialogoImagen**.

Diagrama de clases del paquete: TablaInteractiva

ExcepcionDeTabla
private String errStr
public ExcepcionDeTabla(String str) public String toString()

Figura IV.3.I.1: Estructura de la clase **ExcepcionDeTabla**.

TablaDeDatos
<pre> public TablaDeDatos() public TablaDeDatos(OrdenadorDeTablas ordenador, Observer observador) </pre>

Figura IV.3.1.2: Estructura de la clase **TablaDeDatos**.

EscuchadorModeloTabla
<pre> public void tableChanged(TableModelEvent e) </pre>

Figura IV.3.1.4: Estructura de la clase **EscuchadorModeloTabla**.

ModeloDeTabla
<pre> public String[] nombreColumnas public Object[][] informacion public void inicializaInformacion(int maxRen, int maxCol) public void inicializaNombres(int maxCol) public int getColumnCount() public int getRowCount() public String getColumnName(int col) public Object getValueAt(int row, int col) public Class getColumnClass(int c) public boolean isCellEditable(int row, int col) public void setValueAt(Object value, int row, int col) public void ponerValor(Object value, int row, int col) </pre>

Figura IV.3.1.5: Estructura de la clase **ModeloDeTabla**.

OrdenadorDeTablas
<pre> protected TableModel tableModel public static final int DESCENDING public static final int NOT_SORTED public static final int ASCENDING private static Directive EMPTY_DIRECTIVE public static final Comparator COMPARABLE_COMPARATOR public static final Comparator LEXICAL_COMPARATOR public static final Comparator NUMERIC_COMPARATOR private Row[] viewToModel private int[] modelToView private JTableHeader tableHeader private MouseListener mouseListener private TableModelListener tableModelListener private Map columnComparators private List sortingColumns </pre>
<pre> public OrdenadorDeTablas() public OrdenadorDeTablas(TableModel tableModel) public OrdenadorDeTablas(TableModel tableModel, JTableHeader tableHeader) private void clearSortingState() public TableModel getTableModel() public void setTableModel(TableModel tableModel) public JTableHeader getTableHeader() public void setTableHeader(JTableHeader tableHeader) public boolean isSorting() private Directive getDirective(int column) public int getSortingStatus(int column) private void sortingStatusChanged() public void setSortingStatus(int column, int status) protected Icon getHeaderRendererIcon(int column, int size) private void cancelSorting() public void setColumnComparator(Class type, Comparator comparator) protected Comparator getComparator(int column) private Row[] getViewToModel() public int modelIndex(int viewIndex) private int[] getModelToView() public int getRowCount() public int getColumnCount() public String getColumnName(int column) public Class getColumnClass(int column) public boolean isCellEditable(int row, int column) public Object getValueAt(int row, int column) public void setValueAt(Object aValue, int row, int column) </pre>
<pre> private class Row implements Comparable private class TableModelHandler implements TableModelListener private class MouseHandler extends MouseAdapter private static class Arrow implements Icon private class SortableHeaderRenderer implements TableCellRenderer private static class Directive </pre>

Figura IV.3.1.3: Estructura de la clase **OrdenadorDeTablas**.