POLARIS
HERRAMIENTA DE MINERÍA DE USO PARA LA WEB

DIANA PATRICIA ANGULO URBANO
JENNY JOHANA DAZA BURBANO
ALEJANDRA ZULETA MEDINA

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
PASTO
2007

POLARIS
HERRAMIENTA DE MINERÍA DE USO PARA LA WEB

DIANA PATRICIA ANGULO URBANO
JENNY JOHANA DAZA BURBANO
ALEJANDRA ZULETA MEDINA

Trabajo de Grado presentado como requisito parcial para optar el título de Ingenieros de Sistemas

Dr. RICARDO TIMARÁN PEREIRA, Ph.D.
Director del Proyecto

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
PASTO
2007

”Las ideas y las conclusiones aportadas en el presente trabajo son responsabilidad exclusiva de sus autores”

Artículo 1, acuerdo No. 324 de octubre 11 de 1996, emanado por el Honorabe Consejo Directivo de la Universidad de Nariño.

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

San Juan de Pasto, 19 de Octubre de 2007.

CONTENIDO

LISTA DE FIGURAS

Pruebas Hpg

Manual De Ayuda

Instalación De Potsgresql™ 8.2

POLARIS

LISTADO DE TABLAS

Resumen

En este proyecto se muestra las etapas de análisis, diseño e implementación de Polaris, una herramienta de Minería de uso para la web para facilitar el análisis del tráfico web de un sitio y servir de apoyo en la acertada toma de decisiones.

La arquitectura Polaris consta de tres módulos, el módulo de utilidades para realizar la conexión de datos y que además contiene todas las clases y bibliotecas que son utilizadas en toda la aplicación, el módulo del kernel que incluye el preprocesamiento de los datos que van a ser analizados y los algoritmos utilizados en el proceso KDD y la minería de uso, y el módulo de la interfaz gráfica que permite la interacción amigable entre el usuario y la herramienta.

Los algoritmos de minería de datos implementados son Apriori, FPGrowth y EquipAsso para la tarea de asociación, C4.5 y Mate-Tree para la tarea de clasificación, y el algoritmo de minería de uso HPG.

Se analiza y evalúa el desempeño de los nuevos algoritmos de minería de datos EquipAsso en asociación y Mate-Tree en clasificación, con respecto a los algoritmos Apriori y FPGrowth y C4.5 respectivamente; así como pruebas de rendimiento con el algoritmo HPG.

Abstract

In this work it is shown the analysis stages, design and implementation of Polaris, A Web Usage Mining tool to facilitate the analysis of the Web traffic of a web site and to serve as support in the one guessed right taking of decisions.

Polaris architecture contains three modules, the module of utilities to make the data connection and it also contains all the classes and libraries that are used in the whole application, the Kernel module that includes the data preprocessing and the algorithms used in the KDD process, and the graphic interface module that allows the friendly interaction between the user and the tool.

The data mining algorithms implemented are Apriori, FPGrowth and EquipAsso for the association task, C4.5 and Mate-Tree for Classification task, and the algorithm of Web Usage Mining called HPG.

It's analyzed and evaluated the new data mining algorithms performance like EquipAsso for the association task and Mate-Tree for Classification task respect to Apriori, FPGrowth and C4.5 algorithms and the HPG algorithm.

1. INTRODUCCIÓN

La Internet es sin duda el mecanismo más importante, práctico y altamente difundido para el intercambio de información de todo tipo convirtiéndose en un recurso de disposición pública y general al cual acceden las personas en busca de satisfacer sus necesidades de información, obtener algún recurso en particular o realizar algún tipo de transacción, sin mencionar la gran cantidad de negocios que se manejan por Internet, la competencia y la creciente necesidad de mejorar los servicios para poder sobrevivir en un ambiente competitivo.

Sin embargo, el gran volumen de información contenido en la web hace cada vez más complejo y caótico el proceso de búsqueda de la misma, sobre todo cuando no se cuentan con los recursos necesarios para llegar a ella o se desconoce una forma de acceso rápida que garantice la obtención del recurso deseado en medio de un mar de sobre-información.

En los últimos años se han planteado nuevas tecnologías web en busca de una solución a este problema de sobre-información, como por ejemplo la aplicación de técnicas de Minería de Datos o Data Mining sobre los datos contenidos en la World Wide Web, haciendo especial referencia a la minería en la web o web mining cuyo principal objetivo es el aprovechamiento de técnicas de minería de datos para obtener conocimiento de la información disponible en Internet, abordando el descubrimiento y análisis de información útil en Internet con el fin de obtener, entre otros, los diferentes patrones de comportamiento de los usuarios que acceden a un sitio on-line.

Actualmente las empresas adquieren herramientas software para el análisis del tráfico web, sin embargo la mayoría de ellas, además de ser bastante costosas, generan únicamente reportes estadísticos, los cuales si bien son útiles, no son los suficientemente ricos en información que pueda dar una visión mas completa sobre los accesos de usuarios a un determinado sitio.

Una herramienta software que realmente este enfocada al análisis del tráfico web podría ser open source, multiplataforma e integrar una serie de componentes que permita la obtención de reportes textuales, reportes gráficos del trafico on-line y reportes estadísticos; los cuales deben ser el resultado de un proceso de minería de datos en el cual se apliquen las técnicas y los algoritmos apropiados con el fin de

asegurar la obtención de información útil para una acertada toma de decisiones de los usuarios.

En este documento se presenta el trabajo de grado para optar por el título de Ingeniero de sistemas. El resultado de la presente investigación es "Polaris: herramienta de minería de uso para la web", que implementando algoritmos de Minería de Datos facilita los procesos de encontrar, entender, visualizar e interpretar la información y los recursos web deseados. En ella se implementaron los algoritmos Apriori [4], Equipasso [38,39,41] y FPGrowth [24,13] para las tareas de Asociación; C4.5 [29,35], Mate-Tree [42] y HPG [25] para las tareas de clasificación; además se deja una base teórica para la implementación futura del algoritmo K-means [14] para la tarea de agrupación o Clustering.

El resto de este documento se encuentra organizado de la siguiente manera: En la siguiente sección se especifica el Tema propuesto, la línea de investigación a la cual pertenece y la delimitación del mismo, seguido de una descripción del problema que es objeto de estudio. A continuación, en la primera sección se especifican los objetivos generales y específicos del proyecto, en la segunda sección se presenta el estado general del arte en el área de Minería Web de Uso o Web Mining y en la tercera sección se presenta todo lo concerniente al desarrollo del proyecto, la implementación del mismo, las pruebas realizadas y los resultados obtenidos. Y finalmente en la cuarta sección se presentan las conclusiones del trabajo realizado.

1.1 TEMA

1.1.1 Título. Polaris: herramienta de minería de uso para la web.

1.1.2 Línea de investigación. El presente trabajo de grado, se encuentra inscrito bajo la línea de software y manejo de información, enmarcada dentro de la temática de las bases de datos específicamente en el proceso de descubrimiento en base de datos.

1.1.3 Alcance y delimitación. Polaris es una herramienta genérica para el análisis de tráfico web, que contempla las etapas del proceso de Descubrimiento de Conocimiento en Base de Datos DCBD, decir selección, preprocesamiento, transformación, minería de datos y visualización [2, 4,12]. En la etapa de minería de datos se implementaron las tares de asociación y clasificación. Para las cuales se utilizaron los operadores algebraicos relacionados y primitivas SQL,

desarrollados por Timarán [38,41], con los algoritmos Equipasso [38, 39,41] y Mate-Tree [40]. En la etapa de visualización se desarrollo una interfaz gráfica que le permite al usuario una sencilla interacción con la herramienta.

Las pruebas de rendimiento para los algoritmos implementados se realizaron utilizando conjuntos de datos reales; es decir, archivos logs de acceso del portal de la Universidad de Nariño y otros archivos públicos obtenidos en la web. La comparación de los resultados obtenidos fueron realizados con los algoritmos Apriori [4], FP- Growth [24,13], y C4.5 [29,35].

1.2. PROBLEMA OBJETO DE ESTUDIO

1.2.1 Descripción del problema. La web además de ser una fuente inagotable de información es también uno de los más grandes mercados para cualquier empresa. Sin embargo debido, principalmente, a la heterogeneidad y redundancia de la información y el creciente ascenso de usuarios en la red, hace caótico la identificación de las áreas de mercado y posibles clientes potenciales dentro de la misma. Los sitios web actuales están diseñados generalmente para proveer de información a los usuarios, sin tener en cuenta las particularidades de cada uno de ellos.

En el mercado, existen númerosas herramientas que generan reportes estadísticos y gráficos sobre el uso del servidor, de las cuales es posible destacar algunos productos conocidos como Webtrends, Getstats, Analog etc. [26], pero la mayoría de estas, no aplica las técnicas ni los algoritmos de minería de datos para las tareas de asociación, patrones secuenciales o clustering, además presentan inconvenientes en el pre y post procesamiento de los datos.

La mayoría de estas herramientas presenta las siguientes desventajas, que no solo impiden la adquisición de las mismas por las pequeñas y medianas empresas o usuarios del común; sino que además limitan la obtención de resultados incidentes para una adecuada toma de decisiones:

- Son costosas
- Las herramientas libres son muy grandes en tamaño, lo que incide en el incremento del tiempo empleado al momento de realizar una descarga.
- Los módulos para Web Usage Mining (Minería de Uso Web) no se encuentran integrados en una sola herramienta.

- No hay una interacción sencilla y agradable entre la interfaz gráfica y el usuario.
- El tiempo de ejecución y espera de resultados es bastante elevado.

Con base a lo anterior se plantea el desarrollo de una herramienta software de minería de Datos orientada hacia la Web, que permita encontrar, extraer, clarificar y evaluar la información y los recursos deseados, descubrir, seguir y analizar el comportamiento de los usuarios y sus patrones de acceso, mejorar la navegabilidad del sitio, mejorar el diseño de las aplicaciones, descubrir potenciales clientes de comercio electrónico, elaborar acertadas estrategias de Marketing… en una sola palabra brindar un acceso más fácil y eficiente en la Web.

1.2.2. Formulación del problema. ¿El desarrollo de una herramienta de minería web de uso bajo software libre facilitaría la toma de decisiones en la administración de un sitio web?

1.3. OBJETIVOS

1.3.1 Objetivo general.

Facilitar la toma de decisiones en la administración de un sitio Web a través de una herramienta de Minería Web de Uso bajo software libre.

1.3.2 Objetivos específicos

1. Analizar las diferentes herramientas de minería web de uso existentes.

2. Investigar las técnicas y algoritmos de minería de datos, incluyendo los algoritmos desarrollados por el Ph.D. Ricardo Timarán.

3. Analizar, diseñar e implementar los algoritmos para las tareas de asociación y clasificación.

4. Analizar, diseñar e implementar programas que permitan la visualización gráfica de los resultados obtenidos.

5. Integrar todos los programas en una sola herramienta software.

6. Obtener depósitos de datos reales del portal de la Universidad de Nariño y otros públicos obtenidos de la web.

7. Realizar las pruebas de rendimiento con los depósitos de datos reales en la herramienta Polaris.

8. Analizar los resultados de las pruebas realizadas.

1.4 JUSTIFICACIÓN

Actualmente, contamos con un enorme almacén de datos llamado Internet que permite a millones de usuarios navegar a través de él, mediante un sinnúmero de enlaces que llevan de un sitio a otro.

Cada vez que se da un clic, se desprende información valiosísima sin que nos demos cuenta de ello, información, que con un adecuado análisis puede brindar una clara idea de los diferentes patrones de comportamiento de los usuarios, modelos de invaluable importancia para todas las empresas que prestan servicios u ofrecen productos on-line. Para ello, la mayoría de las empresas, están optando por la adquisición de herramientas que permitan el análisis del tráfico Web; cada una de ellas con diferentes tipos de arquitecturas, características funcionales y costos. Generalmente la mayoría de estas herramientas necesitan de la adquisición de costosas licencias para su utilización y de un software específico para su uso.

El desarrollo de Polaris como una herramienta software bajo licencia pública GPL para el análisis del tráfico Web, permitirá a todo tipo de empresas, entidades y personas del común, utilicen tecnologías Web Mining como un mecanismo de apoyo en la toma acertada de decisiones.

Polaris, se convierte en un aporte significativo en cuanto a investigación científica de tecnología Web Mining, un reconocimiento mas para el área de Descubrimiento de Conocimiento en Base de Datos de la Universidad de Nariño que a través del programa de Ingeniería de Sistemas contribuye al desarrollo de la región y del país.

2. MARCO TEÓRICO

2.1 WEB MINING

"La minería web o web Mining trata de descubrir patrones interesantes en la estructura, el contenido y la utilización de los sitios web… ". [10]

Se puede definir Web Mining como el proceso de extracción de patrones potencialmente útiles e interesantes y de información implícita (*Data Mining*) de datos provenientes de la web o, de manera más sencilla, como "la aplicación de técnicas de *Data Mining* a grandes depósitos de datos web" [11].

Antes de comenzar a hablar de un proceso de minería, se debe conocer de antemano el tipo de datos al que se le va a aplicar dicha técnica. Hay tres tipos de datos principales en la web. El más importante y difícil de procesar es el contenido, que es multimedia, en el cual el texto juega un rol dominante. El segundo proviene de la estructura no lineal de la web; es decir, sus hyper-enlaces. Finalmente, el último procede del uso reflejado a través de los logs o bitácoras de los servidores web.

2.1.1. Tipos de minería web. Existen tres tipos de minería web: minería de la estructura web, minería del contenido de la web y minería de uso en la web.

Minería de la estructura de la web (web structure mining). Esta especialidad pretende revelar la estructura real de un sitio web a través de la recogida de datos referentes a su estructura y, principalmente a su conectividad y/o topología.

Minería del contenido de la web (web content mining). Su objetivo es la recogida de datos e identificación de patrones relativos a los contenidos de la web y a las búsquedas que se realizan sobre los mismos. Hay dos estrategias principales:

- Minería de páginas web, que extraen patrones directamente de los contenidos existentes en las páginas.

- Minería de resultados de búsqueda, que intenta identificar patrones en los resultados de los motores de búsqueda.

Minería de la utilización de la web (web usage mining). Intenta encontrar patrones sobre el uso que se le da a la Web a través del análisis de los registros de los servidores (log files) sobre todas las transacciones informáticas realizadas. Es posible distinguir también aquí:

De las tres áreas en las que se suele dividir la minería web o web mining, la que más éxito ha tenido es la minería web de uso o web usage mining, que se caracteriza por la aplicación de técnicas de data mining para la obtención de patrones acerca del uso que los usuarios le dan a la web.

Minería de uso en la web: La minería de uso en la web - web usage mining - es la aplicación de las técnicas de minería de datos para descubrir patrones de uso desde los datos web, con el fin de entender y mejorar el servicio basado en las necesidades web.

Incluye típicamente el desarrollo de tres fases:

- Preparación y transformación de datos.
- Descubrimiento de patrones.
- Análisis de patrones.

Preparación y Transformación de datos. Una de las tareas más importantes para el uso de técnicas de minería de datos es la creación de un grupo de datos sobre el cual sus algoritmos puedan ser aplicados. Este proceso involucra el preprocesamiento de los datos originales, la integración con múltiples recursos (si es el caso) y la transformación de los datos integrados, en una forma adecuada, para ser utilizados en operaciones específicas de la minería.

En el caso particular de minería web de uso, la preparación y transformación de datos incluyen subprocesos específicos como limpieza de datos, identificación de páginas vistas, identificación de usuarios, identificación de sesiones (también llamado sesionalización), inferencia de referencias perdidas debido a problemas de caché y la identificación de transacciones (ó episodios)

Descubrimiento de Patrones. Una vez los datos han sido preparados, métodos estadísticos y de máquina de aprendizaje (machine learning), son usados para extraer patrones de uso. Existe una gran variedad de modelos de aprendizaje utilizados para el descubrimiento de patrones en minería web de uso. Los modelos más comunes son: Descubrimiento de asociaciones y patrones secuenciales,

clasificación y agrupación (clustering). Los primeros modelos de minería de datos utilizados en minería web de uso fueron aquellos relacionados con clasificación, sin embargo, debido a la gran dificultad que representa etiquetar grandes cantidades de datos para realizar sobre ellos un aprendizaje supervisado, las técnicas más usadas hoy en día son aquellas que pertenecen a aprendizaje no supervisado, como agrupación.

Utilización de las técnicas:

- Reglas de asociación es un modelo para encontrar patrones frecuentes, asociaciones y correlaciones entre grupos de ítems. Las reglas de asociación son usadas para revelar correlaciones entre páginas accedidas durante una sesión. Dichas reglas indican, además, la posible relación existente entre páginas visitadas simultáneamente aunque no se encuentren conectadas de una forma directa, y relaciones entre grupos de usuarios sin intereses específicos.

- El descubrimiento de patrones secuenciales es una extensión de minería de reglas de asociación que refleja patrones de concurrencia incorporando la noción de secuencia de tiempo. En el dominio web, dichos patrones podrían ser grupos de páginas web accedidas inmediatamente luego de otro grupo de páginas. A través de esta aproximación se pueden descubrir diferentes tendencias de los usuarios y usarlas para realizar predicciones.

- Técnicas de agrupación son usadas para reunir ítems que tienen características similares. En el contexto de minería web, se pueden distinguir dos casos: agrupación de usuarios y agrupación de páginas. La agrupación de páginas identifica grupos de páginas que, desde el punto de vista de los usuarios, se relacionan conceptualmente. Por otra parte, la agrupación de usuarios genera grupos de usuarios que exhiben un comportamiento de navegación similar en la web.

- Clasificación es un proceso que transforma ítems de datos en una de diferentes categorías preestablecidas. De esta manera, en el dominio web se realizan diferentes tipos de categorizaciones como aquellas a documentos, tipos de usuarios, entre otras.

Análisis de Patrones. Dependiendo de los objetivos que se quieran alcanzar con la aplicación de técnicas y modelos de minería web de uso, deberá llevarse a cabo una fase de análisis y estudio de los resultados obtenidos. Es importante

mencionar que el trabajo con minería web de uso es independiente del dominio de aplicación sobre el que se esta realizando el estudio, sin embargo para casos en los cuales el fin de este trabajo esta enfocado en descubrir intereses de usuarios en un sitio web particular.

2.2 ETAPAS DE MINERÍA WEB

2.2.1. Tipos de Logs

Cada vez que un cliente realiza una visita a un sitio web, en el servidor queda registrado un archivo con toda la información sobre dicha transacción en un fichero de datos llamado fichero log o log file. Hay tres tipos de ficheros log de particular importancia:

- Log de acceso – Access log –
- Log de error – Error log -
- Log de referencia – Referer log -

Formato NCSA

Log de acceso. Pueden ser guardados en un formato de fichero log común - Common Log File CLF - (figura 2.1) o en un formato de fichero log extendido - Extended Log File ELF - (figura 2.2).

Figura 2.1. Formato de Fichero Log Común

```
in24.inetnebr.com - - [01/Aug/1995:00:00:01 -0400] "GET /shuttle/missions/sts-68/news/sts-68-mcc-05.txt HTTP/1.0" 200 1839
```

Los elementos de los que consta el CLF son:

Remotehost (host remoto). Será el IP si el DNS hostname no esta disponible o si DNSlookup está Off (Esta herramienta devuelve el Nombre de Dominio a una dirección IP correspondiente). En el ejemplo es in24.inetnebr.com.
Rfc931. El nombre de host remoto del usuario. Normalmente este campo no se llena y aparece como - .

Authuser. El nombre con el que el usuario se ha identificado. Normalmente este campo no se llena y aparece como - .

[fecha]. Fecha y hora de la solicitud. En el ejemplo: [01/Aug/1995:00:00:01 -0400]

"Solicitud". La línea exacta de petición según viene solicitada desde el cliente. En el ejemplo es "GET /shuttle/missions/sts-68/news/sts-68-mcc-05.txt HTTP/1.0".

Estado. El código de estado del HTPP devuelto al cliente. En el ejemplo es 200.

Bytes. La longitud que tiene el documento contenido. En el ejemplo es 1838 bytes.

Figura 2.2. Formato de Fichero Log Extendido

```
217.216.54.238 - - [13/Oct/2005:10:25:34 +0200] "GET /portalcpp/web/estilos.css
HTTP/1.1" 304 0 "http://altair.ugr.es/portalcpp/web/form.php" "Mozilla/4.0
(Compatible; MSIE 6.0; Windows NT 5.1)"
```

Los elementos de los que consta el ELF son:

Remotehost o (host remoto). Será el IP si el DNS hostname no esta disponible o si DNSlookup está Off (Esta herramienta devuelve el nombre de dominio a una dirección IP correspondiente). En el ejemplo es 217.216.54.238

Rfc931. El nombre de host remoto del usuario. Normalmente este campo no se llena y aparece como - .

Authuser. El nombre con el que el usuario se ha identificado. Normalmente este campo no se llena y aparece como - .

[fecha]. Fecha y hora de la solicitud. En el ejemplo: [13/Oct/2005:10:25:34 +0200]

"Solicitud". La línea exacta de petición según viene solicitada desde el cliente. En el ejemplo es "GET /portalcpp/web/estilos.css HTTP/1.1".

Estado. El código de estado del HTPP devuelto al cliente. En el ejemplo es 304.

Bytes. La longitud que tiene el documento contenido. En el ejemplo es 0 bytes.

Referente. URL desde donde se ha realizado la petición. En el ejemplo es "http://altair.ugr.es/portalcpp/web/form.php".

Agente. Tipo de navegador y sistema operativo usado. En el ejemplo es "Mozilla/4.0 (compatible; MSIE 6.0; Windows™ NT 5.1)".

Log de error. Contienen información de todos los tipos de posibles errores, como por ejemplo enlaces y los intentos de accesos no autorizados. Los campos que se encuentran en este tipo de archivo log para el formato NCSA:

[Día Mes Año Hora (hh:mm:ss) Zona Horaria] **[**Error**]** **[**client + ip cliente que produjo el error**]** mensaje_de_respuesta_ante_error**: /**ruta en el sistema de ficheros del documento solicitado. (figura 2.3).

Figura 2.3. Formato de Fichero Log de Error

```
[Wed Oct 11 14:32:52 2000] [error] [client 127.0.0.1] client denied by server configuration:
/export/home/live/ap/htdocs/test
```

Log de referencia. Es un fichero opcional que contiene información de las páginas webs desde donde una página concreta es accedida.

Formato. http://url-origen -> /url-destino. (figura 2.4).

Figura 2.4. Formato de Fichero Log de Referencia

```
http://www.w3.org/hypertext/DataSources/WWW/Servers.html -> /spain_www.html
http://guide-p.infoseek.com/WW/NS/tables/DB?C923,510&db=78 -> /bbedit-html-
extensions.html
http://www.compuserve.com:80/hot/wide.html -> /
http://www.yahoo.com/Regional/Countries/Spain/ -> /spain_www.html
http://www.uji.es/spain_www.html -> /mapes/spain_info.html
```

Formato de un archivo Log IIS™

El formato de Microsoft IIS es un formato ASCII fijo (no personalizable). Registra más datos que el formato común NCSA. El formato de Microsoft IIS™ incluye elementos básicos como la dirección IP del usuario, el nombre de usuario, la fecha y la hora de petición, el código de estado HTTP y el número de bytes recibidos.

Además, incluye elementos detallados como el tiempo transcurrido, el número de bytes enviados, la acción (por ejemplo, una descarga realizada con un comando GET) y el fichero de destino. Los elementos se separan con comas, por lo que resulta más sencillo leer el formato que con los demás formatos ASCII, que utilizan espacios como separadores. La hora de registro es la local.

2.2.2. Preprocesamiento. El preprocesamiento de los datos de entrada es una de las fases mas difícil en todo lo que es el web mining, por la complejidad y la cantidad de tiempo que requiere; es decir, es la fase en la que se integran las tareas de remover datos erróneos, identificar los visitantes y las acciones que los mismos realizan.

Esta fase incluye los siguientes pasos:

- Eliminación de robots de acceso
- Filtrado de imágenes y datos ruidosos
- Extracción de transacciones o sesiones de usuarios

Las tareas de eliminación de robots de acceso y el filtrado de las imágenes y datos ruidosos hacen parte de la limpieza de los datos, la tarea de extracción de transacciones o sesiones de usuarios hace parte de la transformación de datos.

2.2.3. Limpieza de Datos. La limpieza de datos, es el proceso mediante el cual, se eliminan todo tipo de datos erróneos, incompletos o inconsistentes que pueden conducir a una equívoca toma de decisiones. Dentro de ella se deben realizar por lo menos los siguientes pasos:

Eliminación de robots de acceso Web: Los logs, por lo regular, contienen entradas de robots Web, como los crawlers, spiders, índices y otros; por lo general son creados por el administrador del sitio web, para generar los permisos de acceso. Y por lo regular se encuentran en un archivo .txt. Es por ello que deben ser excluidos del análisis por que no aportan ningún tipo de información relevante que merezca ser analizada.

Filtrado de imágenes y datos ruidosos: Las páginas web, además cargan archivos de imágenes, sonidos o video con los datos. Por consiguiente, el servidor web graba las entradas que fueron solicitadas de imágenes, sonidos o videos, así como las que fueron enviadas. Generalmente, estas entradas se descartan a la hora de hacer el análisis de Logs.

El proceso de la limpieza de datos no es tan complejo como el de transformación de datos; a grandes rasgos lo que se pretende lograr es la consistencia e integridad de los datos, es decir, logra eliminar registros erróneos o no válidos e intentar que no falten campos.

Sin embargo, aunque la teoría diga lo contrario, en el desarrollo del proyecto, fueron muchos los problemas que se resolvieron al respecto, puesto que los datos de entrada para un proceso de minería web; es decir, los archivos logs, no presentan un formato con el cual se pueda dar inicio al trabajo; es por eso que se debe adaptar el mismo de manera que pueda ser factible la aplicación de técnicas de limpieza de datos como reducción, transformación, adición a los archivos CLF y ELF. De esta adaptación se logra obtener un registro log cuyo formato presenta los siguientes campos:

- **Hostremoto:** dirección que realiza la petición al servidor. Si el DNS hostname no está disponible, entonces aparecerá la dirección IP.
- **Auser**: El nombre de log remoto del usuario.
- **Fecha**: Fecha de la solicitud: año (aaaa) – mes(mm) – día(dd).
- **Hora**: Hora en la que se realizó la solicitud: hora(hh) : minutos(mm) : segundos(ss).
- **Método**: Método de la Petición.
- **Petición**: La solicitud exacta.
- **Protocolo**: Protocolo.
- **Estado**: El código de estado del HTPP devuelto al cliente..
- **Bytes**: cantidad de bytes descargados del archivo solicitado.
- **Referencia**: URL desde donde se ha realizado la petición.
- **AgenteU**: tipo de navegador y sistema operativo usado.

Para lograr el anterior cometido, se realizaron las siguientes adaptaciones a los datos de entrada:

- Se eliminaron todos los registros inconsistentes, es decir todos aquellos cuyos campos necesarios, aparecían vacíos.

- Un fichero log con formato IIS recibió el mismo tratamiento realizado para un fichero log con formato común Comon Log File. De esta manera los campos del fichero log con formato IIS de la figura 2.19 toman los siguiente nombres del Comon Log File:

Client IP address - > Hostremoto
User name - > Auser
Date - > Fecha
Time - > Hora
Request type - > Método
Target of Operation - > Petición
Service status code - > Estado
Server bytes sent - > Bytes

- Si en un fichero Comon Log File –CFL- se encuentran datos pertenecientes a un fichero Extendido Log File_ELF- dichos datos adicionales no se tendrán en cuenta, y de manera idéntica se aplica para un fichero ELF en el que se encuentran datos de CLF.

- Se tomaron como archivos logs válidos aquellos que en su campo solicitud presenten el siguiente formato: "GET <petición>", donde <petición> es la solicitud que se realiza al servidor.

- En el campo estado se eliminaron todas aquellas peticiones que no son exitosas, es decir aquellas que comiencen con los códigos de error de servidor 400 y 500. Los valores de los códigos de error del servidor, también llamados códigos de petición http, se especifican en la tabla 2.1.

- Se eliminaron todos los archivos con extensiones gráficas, como: .gif, jpg, jpeg, png, swf, map.

- Se eliminaron todas las peticiones realizadas por los robots, como Googlebot, InfoSeek Robot 1.0, Infoseek Sidewinder, PerlCrawler 1.0, Scooter, SpiderBot, Site Searcher, entre otros.

- Se determinaron los siguientes valores por defectos para los diferentes campos cuyos valores aparecían como "-":

    - Host Remoto **->** " "
    - Usuario **->** "anónimo"
    - Método **->** "GET"
    - Solicitud **->** "nada"
    - Protocolo **->** "HTTP/1.0"

- Bytes **->** "0"
- Referente **->** " " ; esto en el caso de que el registro presente un Formato Común CLF
- Navegador **->** " " ; esto en el caso de que el registro presente un Formato Común CLF.

Tabla 2.1. Lista de Códigos de petición http

CODIGO DE ESTADO DEL SERVIDOR	
CODIGO	SIGNIFICADO
1xx	**Información**
100	Continue
101	Switching Protocolo
2xx	**Éxito**
200	Éxito
201	Creado
202	Aceptado
203	Información no autorizada
204	No hay contenido
205	Contenido reiniciado
206	Contenido Parcial
3xx	**Redirección**
300	Selecciones Multiples
301	Movido Permanentemente
302	Encontrado
303	Observar otro
304	No modificado
305	Uso de Proxy
307	Redirección temporal
4xx	**Error en el Cliente**
400	Petición erronea
401	No autorizado
402	Pago requerido
403	Prohibido
404	No encontrado
405	Método no permitido
406	No aceptado
407	Requiere autenticación del Proxy
408	Tiempo de espera agotado
409	Conflicto
5xx	**Error del Servidor**
500	Error interno del servidor
501	No implementado
502	Error del Gateway
503	Servicio no disponible
504	Tiempo agotado en el Gateway
505	Version http no soportada

- Si en un fichero Comon Log file – CFL - se encuentran datos pertenecientes a un fichero Extendido Log File – ELF - dichos datos adicionales no se tendrán en

cuenta, y de manera idéntica se aplica para un fichero ELF en el que se encuentran datos de CLF.

2.2.4. Transformación de Datos. El proceso de transformación de los datos es mucho más complejo que el de la limpieza, dentro de las tareas que en esta fase se deben realizar están:

Extracción de transacciones o sesiones de usuarios: Una vez se filtren las entradas erróneas, el paso siguiente es extraer las transacciones que pertenecen a los usuarios individuales. No hay definición natural de una transacción en el panorama de la navegación del sitio.

Una transacción se puede considerar como sola entrada en el registro o un sistema de entradas alcanzadas por un visitante de la misma máquina en un lapso de tiempo definido o sesión. La transacción deseada puede ser el sistema de entradas de registro de un visitante en una sola visita. Sin embargo, el uso de los servidores Proxys y la utilización de las cachés, hace difícil la identificación de las sesiones de usuarios.

Este es el punto crucial del análisis, la correcta identificación de las sesiones de los usuarios.

Una **sesión de host** se define como la secuencia de peticiones al servidor que transcurren desde que un determinado host hace la primera petición al servidor hasta que realiza la última. Desde esta primera petición hasta la última, se habrán realizado peticiones secuenciales en espacios cortos de tiempo. El usuario podrá especificar el tiempo máximo que debe transcurrir entre una petición de un host y otra petición del mismo host para que se considere como parte de la misma visita.

El método a utilizar para la identificación de sesiones es el del Time Out, en el cual se determina un periodo de tiempo preestablecido para determinar el número de peticiones de usuario que pertenecen a la misma sesión.

Ahora bien, para poder iniciar con el proceso de transformación de datos, se debe tener en cuenta que tipos de técnicas de minería de datos se les va a aplicar a los mismos; para este caso se aplicaron las técnicas de asociación y clasificación con sus respectivos algoritmos; es por eso que se debe lograr una transformación en los

datos que han sido limpiados de tal manera que pueda ser utilizada por cualquiera de la anteriores técnicas de data mining.

Para ello se transformaron los datos en un registro con formato general que para un fichero CLF toma el nombre de SesionComun y para un fichero ELF toma el nombre de SesionExtendida.

Tanto para los registros logs con formato común como para los registros log con formato extendido se realizó la siguiente transformación personalizada de los datos:

- Todos los códigos de estado de servidor fueron reducidos a sus unidades de centena quedando códigos de servidor desde el código 100 hasta el 500 respectivamente, de la siguiente manera:

    - 100: Información
    - 200: Exito
    - 300: Redireccionamiento
    - 400: Error en el Cliente.
    - 500: Error en el Servidor.

- La capacidad transferencia de Bytes, tomó rangos de descarga entre Alta – 0-8999 bytes-, Media - 9000-200000- y Baja - >200000- respectivamente.

- Se adicionó el campo jornada para determinar el momento en el día en el cual se realizó la visita al sitio, estableciendo "Mañana" para el periodo de tiempo comprendido entre las horas 00:00:00 – 11:59:59 , "Tarde" para el periodo de tiempo comprendido entre las horas 12:00:00 – 18:59:59 y "Noche" para el periodo de tiempo comprendido entre las horas 19:00:00 – 23:59:59.

Transformación de datos para una archivo log con formato común –CLF-

El registro SesionComun es una Clase que se compone de los siguientes campos:

- **Hostremoto**. Nombre del hostremoto que inicia la sesión.

- **FechahoraInicio**. Fecha y hora de inicio de la sesión.

- **FechahoraFin**. Fecha y hora del fin de la sesión.

- **NumSolicitudes**. Número total de solicitudes realizadas durante la sesión.
- **TotalBytes**. Total de bytes transferidos durante la sesión.
- **NomTipoFichero**. Vector que contiene el nombre del tipo de fichero que ha sido descargado.
- **TipoFichero**. Vector que cuenta el número de veces que ha sido descargado ese fichero.
- **NomPagVisitadas**. Vector que contiene el nombre de todas las páginas que componen el sitio web analizado.
- **PagVisitadas**. Vector que contiene el número de veces que cada una de las páginas del sitio web analizado han sido accedidas.

En la figura 2.5. Se aprecia la forma en la que un registro log de formato común es visualizado en la herramienta Polaris.

Figura 2.5. Registro log de formato común.

hostremoto [PK] text	fechahorainic [PK] timestar	fechahorafin timestamp w	numsolicitud integer	totalbytes integer	tiposficheros integer[]	pagvisitadas integer[]	nomtipofiche text[]	nompagvisita text[]
200.21.87.165	2007-03-13 12:	2007-03-13 13:	7	48863	{6,1}	{2,1,1,1,2}	{php,htm}	{/admocara.php
200.21.87.165	2007-03-13 14:	2007-03-13 14:	1	32086	{1}	{1}	{php}	{/matriculas/rep
200.21.87.165	2007-03-13 14:	2007-03-13 14:	4	9545	{4}	{1,1,2}	{php}	{/admocara.php
200.21.87.172	2007-03-13 16:	2007-03-13 16:	3	14600	{3}	{1,1,1}	{php}	{/admocara.php
201.228.48.6	2007-03-13 14:	2007-03-13 14:	4	111072	{2,2}	{2,2}	{html,php}	{/pasto.html,/in
201.228.53.104	2007-03-13 14:	2007-03-13 14:	1	16043	{1}	{1}	{html}	{/pasto.html}

De la anterior figura se aprecia, en el primer caso por ejemplo, que el Host Remoto 200.21.87.165, realizó siete (7) solicitudes en esta sesión, con una descarga total de

48863 bytes de información, descargándose 6 tipos de ficheros .php, y 1 .htm. El vector págvisitadas almacena para cada sitio web el número de veces que han sido solicita; así cada una de las páginas con extensiones .php, fueron visitadas 2, 1, 1,1 veces respectivamente y la página con extensión .htm fue visitada 1 vez.

De esta forma se pudo convertir una serie de registros Log (tabla 2.2) en varios registros correspondientes a la sesión de usuario denominado SesionComun. (tabla 2.3).

Los elementos necesarios para poder realizar dicha transformación de datos son:

- Extensiones web
- Tiempo de Sesión
- Sabiertas
- Tabla sesión común

Extensiones web. Lista de todas las extensiones de los archivos considerados como peticiones del servidor; es decir, el listado de los tipos de ficheros que consideramos como transacciones. (Anexo A.1.).

Proceso de Transformación de los Datos en un Registro Sesión Común –SesionComun-

Tabla 2.2. Serie de registros log de formato común.

Registro	Host Remoto	Fecha
1	A	1
2	A	2
3	B	3
4	A	4
5	C	5
6	B	6
7	B	7
8	A	8
9	C	9
10	A	10
11	C	11
12	A	12
13	C	13
14	B	14

Tabla 2.3. Registro sesión común

Registro Sesión Común – SesionComun-		
RegSesion	**Host Remoto**	**Fecha**
RS 1	A-1	Fecha Inicio
RS 2	B	Fecha Inicio
RS 3	C	Fecha Inicio
RS 4	A-2	Fecha Inicio

Tiempo de Sesión. Tiempo que debe transcurrir entre una petición de una visitante web y otra realizada por el mismo para considerarla como una nueva sesión. El valor por defecto que se toma en Polaris, para definir este Time Out es de 30 minutos.

Sabiertas. Estructura que almacena aquellos registros sesión –RegSesion- cuyos visitantes han iniciado una sesión pero de los cuáles aún no se tiene constancia de que hayan finalizado la sesión.

Tabla Sesión Común. Estructura –tabla- donde se almacenan los registros de aquellos visitantes que iniciaron y finalizaron la sesión.

Con base en los anteriores conceptos, se realizó el algoritmo para definir las sesiones de usuario en un registro sesión común. (figura 2.6).

Figura 2.6. Notación algoritmo sesión común

1. Leer cada uno de los Registros Log –RegLog-
2. RegLog tiene extensión válida:
   Si
   Host remoto está en la lista de Sabiertas
   No
   Ingresarlo en Sabiertos
   Si
   Entonces existe un registro sesión con ese Host remoto
   Ha transcurrido un tiempo mayor se sesión –
   Mayor de 30 minutos – entre los dos registros:

La siguiente es una breve explicación del algoritmo anterior: Se lee cada registro Log –RegLog- y se comprueba si el Host remoto se encuentra en la lista de los Sabiertos; si no se encuentra se debe ingresarlo. Si existe un registro con el mismo Host remoto, se debe comprobar si ha excedido el time out.

Si el tiempo de sesión aun no ha expirado entonces se debe actualizar la información de los registros de Sabiertos e introducirlos en la tabla de la base de datos donde se van a almacenar y dar inicio a una nueva sesión creando el nuevo registro de sesión de usuarios SesionComun a partir de los registros logs que se están leyendo e introducirlos en Sabiertos para iniciar nuevamente con el algoritmo.

Dos importantísimas funciones son la clave del anterior algoritmo, estas funciones se denominan: Actualizar y Crear RegSesion.

- **Actualizar.** Función encargada de actualizar el valor de la hora de finalización del registro sesión con la fecha del registro Log que se encuentre leyendo. También se encarga de incrementar el número de solicitudes, el número de páginas visitadas –bytes- y el tipo de fichero –extensiones páginas- descargado.

- **Crear Regresión.** Esta función se encarga actualizar el valor de la fecha de inicio correspondiente a los valores del registro Log que se encuentre leyendo además de inicializar todos los datos de los campos del mismo.

Transformación de datos para una archivo log con formato extendido –ELF-

El proceso de transformación de datos para un archivo log con formato extendido fue muchísimo más complejo, debido a la mayoría de la documentación estudiada únicamente incluía el análisis para los archivos logs con formato común.

Después de mucho indagar en textos, la mayoría de ellos en ingles, se pudo encontrar que el proceso de identificación de usuarios e identificación de sesiones de usuarios para un archivo log con formato extendido requiere de los siguientes pasos: [43].

- Limpiar el archivo log
- Identificación de usuarios
- Identificación de sesiones
- Completar rutas
- Identificación de transacciones : Algoritmos
  - Reference length
  - Time window.
  - Maximal forward reference

Identificación de Usuarios. En un archivo log con formato extendido la identificación de usuarios únicos se realizará teniendo en cuenta conjuntamente los campos IP y agenteuser del registro log a analizar.

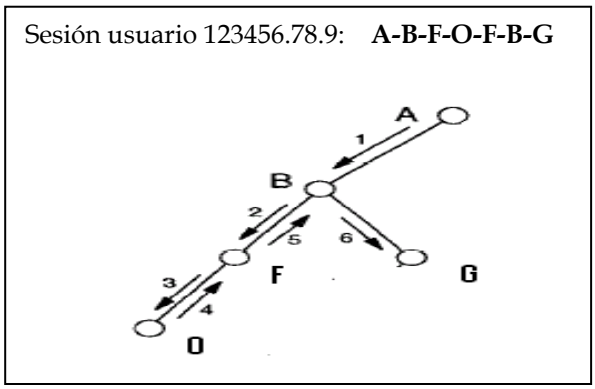
Identificación de Sesiones. La identificación de las sesiones de usuario puede hacerse manejando **intervalos** o utilizando el criterio de establecer un **límite de tiempo de sesión,** que para este caso será de 30 minutos, es decir, deben transcurrir máximo 30 minutos entre una petición de un visitante web y otra realizada por el mismo para considerarla como parte de la misma sesión, de igual forma como se realiza con los archivos logs de formato común –CLF-.

Figura 2.7. Identificación de sesiones por intervalos

Host:	123.456.78.9
AgenteUser:	Mozilla, Win 95
Fecha Inicio:	25-04-1998 03:04:41
Fecha Fin:	25-04-1998 03:12:23
NumSesiones:	7
Páginas Visitadas:	A B L F R O G
Referer:	- A - B L F B

De la figura 2.7 se aprecia que si se siguen las rutas de las páginas basadas en su referer entonces se tendrá que la Página "L" no tiene un referer definido y al proseguir la Página "R" proviene de la página "L", estas dos visitas no se las tendrá en cuenta para el establecimiento de las sesiones de usuario. De acuerdo con esto la sesión del usuario 123456.78.9 es: (figura 2.8).

Figura 2.8. Sesión de Usuario por manejo de intervalos



Completar Rutas. Como se vio en la figura 2.7, hay algunas páginas que deben descartarse, pero aún persiste el propósito de establecer las sesiones del usuario que ingreso al sitio. Es por eso, que se aplica esta técnica, con el fin de conservar la ruta realizada por el usuario. Para ello cada una de las páginas visitadas posee un índice de ruta. (tabla 2.4).

Tabla 2.4. Índice de ruta de páginas.

Página	A	B	L	F	R	O	G
Índice de Ruta	1	2	3	4	5	6	7

Anteriormente, se mencionó que las páginas "L" y "R" debían descartarse, pero la ruta de navegación del usuario no se perderá ya que cada página lleva un índice de ruta. Así es que la ruta del usuario de la tabla 2.4. Será:

Sesión usuario 123456.78.9: **A-B-F-O-F-B-G** **1-2-4-6-4-2-7**

Reference Length. Como su nombre lo indica, el referer lenght, tiene que ver con el manejo de las extensiones del referer o la página de referencia. En cuanto a esto, debe aclararse que no se menciona mucho al respecto, así que únicamente se lo hace parte de la documentación.

Time Window. Hace referencia al umbral o tiempo que se demora en cerrarse la ventana del navegador del usuario.

Maximal Forward References - MFR. Un Maximal Forward References de un usuario Web es la máxima secuencia de páginas visitadas por el usuario sin que regrese hacia alguna página que haya sido visitada previamente en dicha secuencia.

Para poder extraer los patrones de acceso de los usuarios a partir de la base de datos de registros logs, resulta conveniente enfocarse en las páginas que visita el usuario en su navegación, concentrándose en las páginas que realmente visita y no en las que se regresa a revisar por alguna razón.

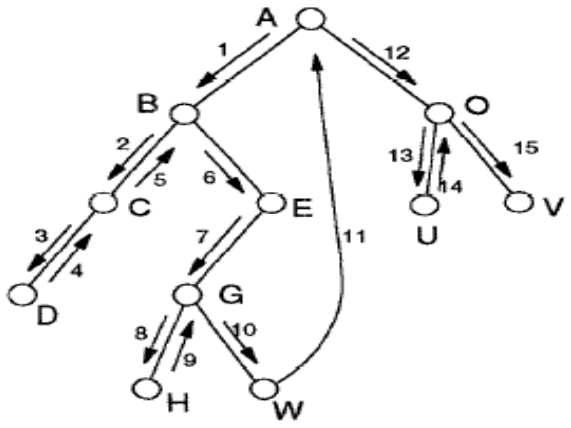
Cuando un usuario hace un backward o se regresa a alguna página que ya ha visitado se da por terminada el camino de referencia que hasta el momento ha realizado; es decir, este algoritmo se enfoca en determinar la ruta de acceso de un usuario mediante las páginas que visite hasta el momento que se regrese a una página anteriormente visitada. [7].

Cuando se trabaja con el algoritmo MFR, la base de datos de los log es leída una sola vez secuencialmente, al mismo tiempo que va generando las sesiones de usuarios. Mientras que una sesión es generada, los MFR dentro de cada sesión también son generados.

Para una mejor comprensión del algoritmo, se lo ilustra con el siguiente ejemplo:

Sean las siguientes transacciones realizadas por un usuario **{** A, B, C, D, C, B, E, G, H, G, W, A, 0, U, 0, V**}**, tal y como se muestra en la figura 2.9.

Figura 2.9. Transacciones de usuario.



Aplicando el MFR se que el conjunto máximo de referencias hacia adelante para este usuario son:

{ A,B, C,D} **{** A, B, E, G, H **}** **{** A,B,E,G,W **}** **{** A,O,U **}** y **{** A,O,V **}** , nótese que se va registrando el camino de acceso del usuario hasta el momento en el cual se regrese hacia un página ya visitada. [8].

Se deben tener en cuenta conceptos como **large refence sequence** que es la secuencia de referencias que aparece un suficiente número de veces – se repite varias veces- este número apariciones debe aprobar un soporte mínimo, es decir debe aparecer por lo menos un número de veces determinadas.

Una **k-large referente** es una secuencia de referencia grande con **k** elementos denotados como **K.**

Es importante mencionar que después de que las **large refence sequence** son determinadas, la **maximal referente sequence**, esta es una secuencia de referencias grandes que no esta contenida en ninguna otra **maximal refences sequence**.
Por ejemplo, supóngase que el conjunto: { AB, BE, AD, CG, GH, BG } es el conjunto de referencias grandes de tamaño 2. denotado como L2; y que { ABE,

CGH } es el conjunto de referencias grandes de tamaño 3 o L3. Entonces la secuencia máxima de referencias resultante es AD, BG, ABE, y CGH.

El procedimiento para realizar este tipo de minería es el siguiente:

Paso 1: Determinar las *maximal forward referentes* a partir del archivo log original.
Paso 2: Determinar *large referente sequences* a partir del conjunto de referencias delanteras máximas, es decir Lk, k ≥ 1.
Paso 3: Determinar la *maximal forward referentes* a partir de la *large referente sequences*.

Después del análisis de los anteriores conceptos, se realizó la transformación de los datos para un archivo log con formato log extendido de la siguiente manera:

El registro SesionExtendida es una Clase que se compone de los siguientes campos.

- **Hostremoto**: Nombre del hostremoto que inicia la sesión.
- **Agenteuser**: Tipo de navegador y sistema operativo usado
- **FechahoraInicio**: Fecha y hora de inicio de la sesión.
- **FechahoraFin**: Fecha y hora del fin de la sesión
- **NumSolicitudes**: Número total de solicitudes realizadas durante la sesión.
- **TotalBytes**: Total de bytes transferidos durante la sesión.
- **NomTipoFichero**: Vector que contiene el nombre del tipo de fichero que ha sido descargado.
- **TipoFichero**: Vector que cuenta el número de veces que ha sido descargado ese fichero.
- **NomPagVisitadas**: Vector que contiene el nombre de todas las páginas que componen el sitio Web analizado.
- **PagVisitadas**: Vector que contiene el número de veces que cada una de las páginas del sitio Web analizado han sido accedidas.

- Cada página tendrá además un ID, un valor secuencial que identifique las páginas diferentes.

Para la implementación de las técnicas y algoritmos en la herramienta Polaris se decidió:

- Realizar la identificación de usuarios Únicos teniendo en cuenta conjuntamente los campos de la dirección **IP y el Agente.**

- Realizar la identificación de Sesiones de usuario a través de **intervalos** y/o mediante la utilización del **límite de tiempo de sesión,** que para este caso será de 30 minutos.
- Completar rutas utilizando un índice de rutas en cada página con el fin de conservar la ruta de navegación del usuario una vez se tengan que descartar algunas páginas, por los motivos que anteriormente se mencionaban en cuanto a este tema.

- Aplicar el algoritmo **Maximal Forward References - MFR -** para poder extraer los patrones de acceso de los usuarios.

En la figura 2.10. se aprecia la forma en la que un registro log de formato extendido es visualizado en la herramienta Polaris:

Figura 2.10. Registro log de formato extendido.

hostremoto [PK] text	agenteu [PK] text	fechahorainicio [PK] timestamp wi	fechahorafin timestamp withou	numsolicitud integer	totalbytes integer	nomtipofiche text[]	tiposficheros integer[]	transacciones text[]
123.456.78.9	Mozila/3.01 (X11, I, IRIX6.2, IP22)"	1998-04-25 03:06:58	1998-04-25 03:10:45	4	16590	{html}	{4}	{A.html,B.html,A.html,C.html,J.h
123.456.78.9	Mozila/3.04 (Win95, I)"	1998-04-25 03:04:41	1998-04-25 03:12:23	7	32196	{html}	{7}	{A.html,B.html,F.html,O.html,F.h
209.456.78.2	Mozila/3.04 (Win95, I)"	1998-04-25 05:05:22	1998-04-25 05:05:22	1	3290	{html}	{1}	{A.html,-}
209.456.78.3	Mozila/3.04 (Win95, I)"	1998-04-25 05:06:03	1998-04-25 05:06:03	1	1680	{html}	{1}	{A.html,D.html,-}

2.3. MINERÍA DE DATOS

Después de los procesos de limpieza y transformación de los datos, éstos ya se encuentran listos para la aplicación de métodos estadísticos y de máquina de aprendizaje – machine learning – los cuales son utilizados para extraer patrones de

uso. Hay varios modelos de aprendizaje para el descubrimiento de patrones en la minería web de uso.

2.3.1. Descubrimiento de Patrones. Los modelos de aprendizaje más utilizados en minería de datos son agrupación – clustering, clasificación, descubrimiento de asociaciones y de patrones secuenciales. De los anteriores, los que más se utilizan en minería web de uso son los modelos de clasificación sin embargo debido el arduo trabajo que representa etiquetar grandes cantidades de datos para realizar sobre ellos un aprendizaje supervisado, se ha optado por utilizar actualmente los modelos que pertenecen al aprendizaje no supervisado, como agrupación.

- **Reglas de asociación.** Es un modelo para encontrar patrones frecuentes, asociaciones y correlaciones entre grupos de ítems.
-

  El problema de Asociación formulado por Agrawal et al. [3,4] a menudo se referencia como el problema de análisis de la canasta de mercado (market-basket analysis). La entrada del problema consiste de un conjunto de ítems y una colección de transacciones que son subconjuntos de estos ítems. La tarea es encontrar relaciones entre los ítems de esas transacciones.

  Encontrar reglas de asociación es básicamente calcular estas relaciones de manera que cumplan unas especificaciones mínimas dadas por el usuario, expresadas en términos de dos parámetros: ***soporte* y *confianza*** [3].

  Una regla de asociación es una implicación de la forma X C Y, donde X es un subconjunto de ítems, Y es un subconjunto de ítems y $X \cap Y = \theta$. La regla X Y lleva asociada una confianza c si el c% de las transacciones en D cumple que si tiene a X también contiene a Y. La regla X **->** Y tiene soporte s si el s% de las transacciones en D contienen S U Y.

  Un ejemplo de una regla de asociación es: El 35% de clientes / usuarios que accedieron a la página web con URL /entidad/anuncio/oferta-especial.html también accedieron a /entidad/productos/ productos1.html, el 3% de todos los accesos contienen a ambas páginas. En este caso, el 35% es la confianza de la regla y el 3% es el soporte de la regla, donde la confianza denota la fuerza de la implicación y el soporte indica la frecuencia de ocurrencia de los patrones en la regla.

Las reglas de asociación pueden usarse para revelar correlaciones entre páginas accedidas durante una sesión. Dichas reglas indican, además, la posible relación existente entre páginas visitadas simultáneamente aunque no se encuentran conectadas de una forma directa, y relaciones entre grupos de usuarios sin intereses específicos. Con ello se puede descubrir las correlaciones entre referencias a diversos archivos disponibles en un servidor por un cliente dado.

- **El descubrimiento de patrones secuenciales**. Es una extensión de minería de reglas de asociación que refleja patrones de concurrencia incorporando la noción de secuencia de tiempo. En el dominio Web, dichos patrones podrían ser grupos de páginas Web accedidas inmediatamente luego de otro grupo de páginas. A través de esta aproximación se pueden descubrir diferentes tendencias de los usuarios y usarlas para realizar predicciones.

- **Técnicas de agrupación**. Son usadas para reunir ítems que tienen características similares. En el contexto de minería web, se pueden distinguir dos casos: agrupación de usuarios y agrupación de páginas. La agrupación de páginas identifica grupos de páginas que, desde el punto de vista de los usuarios, se relacionan conceptualmente. Por otra parte, la agrupación de usuarios genera grupos de usuarios que exhiben un comportamiento de navegación similar en la web.

- **Clasificación.** Es un proceso que transforma ítems de datos en una de diferentes categorías preestablecidas. De esta manera, en el dominio Web se realizan diferentes tipos de categorizaciones como aquellas a documentos, tipos de usuarios, entre otras y es posible desarrollar un perfil de clientes quienes acceden a archivos particulares de un server basado en sus patrones de acceso.

- **Clustering.** Clustering es la agrupación automática de clientes o datos con características similares sin tener una clasificación predefinida. Es un aprendizaje no supervisado es decir no existe un atributo clase. Se intenta encontrar el grupo de usuarios, páginas o sesiones a partir de los archivos logs, donde cada clúster representa un grupo de objetos con características comunes. Esta técnica intenta descubrir estructuras en los datos de entrada, buscando agrupamientos entre los ejemplos de forma que cada grupo o clúster sea homogéneo y distintos de los demás.

Un **Clúster** es la agregación de puntos en el espacio de entrada donde la "similitud" entre cada par de objetos es menor que la "similitud" de cualquiera de ellos a otro objeto que no pertenece al clúster.

Hay distintos tipos de Clustering:

- Jerárquicos: los datos se agrupan de manera arborescente, pueden ser top-down o bottom-up. **Ejemplo:** Horizontal hierarchical tree.
- No jerárquicos: generar particiones a un solo nivel. **Ejemplo:** k-means.
- Paramétricos: asumen que las densidades condicionales de los grupos tienen cierta forma paramétrica conocida (gaussiana), y se reduce a estimar los parámetros. **Ejemplo:** Algoritmo EM (Estimated Means).
- No paramétricos: no asumen nada sobre el modo en el que se agrupan los objetos. **Ejemplo:** 1NN (Nearests Neighbour).

Se utiliza esta técnica cuando no existe un conocimiento suficiente acerca de las clases en que se pueden distribuir los objetos de interés o cuando existe un conocimiento completo de las clases y/o se desea comprobar la validez del conjunto de entrenamiento.

El resultado de un algoritmo de agrupamiento depende de:

- El algoritmo concreto empleado para encontrar los agrupamientos: Diferentes aproximaciones a este problema y números algoritmos de agrupamiento. Esta variedad hace que la elección de una estrategia de agrupamiento, e incluso entre diferentes implementaciones de un mismo algoritmo proporcione resultados diferentes.
- El valor de los parámetros del algoritmo
- Los patrones utilizados y en algunas ocasiones, hasta el orden en que se procesan.
- La medida de similitud adoptada: De esto se obtiene la dificultad inherente a la aplicación práctica de los algoritmos de agrupamiento, que puede resumirse en dos puntos claves:

    - Establecer medidas de similitud adecuadas.
    - Establecer buenos criterios de partición.

Con la aplicación del clustering se pretende encontrar grupos bien definidos entre las diferentes sesiones de los visitantes de una página web.

Esta clasificación de grupos puede hacerse con los siguientes criterios:

- Sitios web. Teniendo en cuenta sólo los sitios web que se visitaron durante la sesión, con esto se puede obtener los diferentes patrones de visita.
- Sitios web y día de la semana. Teniendo en cuenta los sitios web que se visitaron durante la sesión y los días de la semana en las que se produjeron las visitas. Con esto se puede obtener los sitios Web que son visitados cada día de la semana.
- Sitios web y hora. Teniendo en cuenta los sitios web que se visitaron durante la sesión y las horas en las que se produjeron las visitas. Con esto se obtiene los sitios web que son visitados cada hora del día.
- Sitios web y tiempo de sesión. Teniendo en cuenta los sitios web que se visitaron durante la sesión y el tiempo que duró la misma. Se obtiene los sitios web que son visitados según el tiempo que dura la sesión.
- Días de la semana y cantidad de bytes descargados.
- Días de la semana y número de solicitudes.
- Días de la semana y tiempo de sesión.
- Hora y tiempo de sesión.

Para obtener los anteriores clúster es necesario una transformación de datos de manera que:

- Se pueda obtener la fecha de inicio de sesión almacenada, los datos de la hora y del día de la semana, estableciendo para la hora un número entero entre 0 a 23 y para el día de la semana un número entero de 0 a 6, donde 6 corresponde al día Domingo.
- Para las páginas visitadas: Para este caso se asigna un número 1 si el número de solicitudes es mayor que 0 y 0 en otro caso.

Entradas:

- k, número de Clúster a generar
- E, conjunto de instancias del problema concreto

Salidas: Conjuntos de valores que tienen una determinada distancia de similitud entre ellos.

La aplicación, sobre datos web, de cada una de las técnicas mencionadas, requiere de una preparación de datos (o configuración) específica que este de acuerdo con el funcionamiento del modelo.

2.3.2. Análisis de Patrones. Dependiendo de los objetivos que se pretenden lograr con la aplicación de las técnicas y modelos de minería web de uso, se debe realizar una fase de análisis y estudio de los resultados obtenidos.

Por ejemplo con las reglas de asociación se pueden encontrar correlaciones como:

• El 45% de clientes / usuarios que accedieron a la página web con URL /entidad/productos/productos1.html también accedieron a /entidad/ productos/productos2.html

A través del análisis de asociaciones, se puede descubrir las relaciones sin que exista intervención alguna por parte del operador. El descubrimiento de estas reglas ayuda a las organizaciones dedicadas al e-commerce a definir sus estrategias de mercados efectivos.

Con las técnicas de Agrupación y Clasificación se puede ayudar a las organizaciones a predecir patrones de navegación de usuarios y a elaborar futuras estrategias de marketing ya que información como los grupos a los cuales es más rentable dirigir las diversas ofertas y promociones, será conocida de antemano.

2.4. ALGORITMOS IMPLEMENTADOS EN POLARIS

2.4.1. Algoritmos de Minería de Datos. Existen diferentes tipos de algoritmos que pueden implementarse en Minería de Datos tales como APRIORI [4], IC (Interval Classifier)[1], C4.5 (Programs for Machine Learning) [34], SLIQ (Supervised Learning in Quest) [27], SPRINT(Scalable Parallel Classifier for Data Mining) [36] , EGIS (Classifications on Rough Sets) [37],…y algoritmos de Minería Web de Uso como HPG [25], dependiendo del tipo de tarea a realizarse, como son las tareas de Asociación ,Clasificación y Agrupación o Clustering.

Es este caso en particular, se tienen las siguientes tareas y sus respectivos algoritmos:

- Tareas de asociación: A priori, FPGrowth, EquipAsso.
- Tareas de clasificación: C4.5, Mate-Tree
- Hypertex Probabilistic Grammar HPG.
- Tareas de agrupación o clústering: K-medias.

Algoritmo A priori

Figura 2.11. Notación Algoritmo Apriori

- **k-itemset:** Un itemset que tiene k items.
- **Lk:** Conjunto de k-itemsets frecuentes (que cumplen con el soporte mínimo).
- **Ck:** Conjunto de k-itemsets candidatos - potencialmente itemsets frecuentes - , asociados con los TIDS de las transacciones generadas.

Algoritmo A priori:

```
L1={1-itemsets frecuentes};
For (k=2; Lk-1 ≠ ∅; k++) do begin
    Ck = a priori_gen(Lk-1); // Nuevos candidatos
    Forall transacciones t ∈ D do begin
      Ct = subconjunto(Ck,t); // candidatos contenidos en t
      Forall candidatos c ∈ Ct do
            c.count++;
    end
    Lk = { c ∈ Ck | c.count ≥ minsup}
End
Respuesta = ∪k Lk;
```

En el primer paso del algoritmo simplemente cuenta el número de ocurrencias de cada item para determinar los 1-itemsets frecuentes.

En un subsecuente paso, en el paso k, consta de dos fases: En la primera, los itemsets frecuentes Lk-1 encontrados en el paso (k-1) son usados para generar los candidatos itemsets Ck, usando la función Apriori_gen(). En la segunda, la base de datos es recorrida y el soporte de cada candidato en Ck es contado.

Para la generación de Candidatos se utiliza la función Aprioi_gen(), la cual toma como argumento Lk-1, el conjunto de todos los (k-1) itemsets frecuentes y retorna un superconjunto del conjunto de todos los potenciales k-itemsets frecuentes.

La función primero, rea liza un join: Lk-1 ▸◂ Lk-1

Luego, en el paso de poda, se borran todos los itemsets **c** ∈ Ck tal que algún (k-1) subconjunto de c no esta en Lk-1. Se basa en la propiedad que todos los subconjuntos de un itemset frecuente deben ser frecuentes.

Así por ejemplo si te tiene L3 {{123},{124},{134},{135},{234}}, después del paso de join, C4 será {{1234}{1345}}. El paso de poda borrará el itemset {1345} porque el itemset {145} no esta en L3. Luego entonces C4 será únicamente {1234}.

Algoritmo FP-Growth

Con el algoritmo Apriori se pueden tener excelentes resultados con bases de datos de transacciones pequeñas, sin embargo se presentan grandes inconvenientes cuando el número de transacciones es grande y debe administrar un gran número de conjuntos candidatos. Además, se le suma la tediosa tarea de tener que acceder a la base de datos para revisar cada conjunto de candidatos.

Es por eso que como medida para la solución de dichos problemas se desarrolla el algoritmo FP-Tree, el cual halla los itemsets frecuentes sin generar candidatos. Con FP-Tree la base de datos de las transacciones –organizada lexicográficamente - se examina solamente dos veces. En la primera escaneada se determina la frecuencia de cada itemset; se elabora una lista de la frecuencia de los items de mayor a menor con la finalidad de descartar los que no pasan el mínimo soporte. Con el segundo escaneo, en cada transacción se remueven los items poco frecuentes y se construye el árbol.

Diseño y Construcción del Árbol de Patrones Frecuentes FP-Tree

Sea I = a1, a2,..., am un conjunto de items, y DB = T1, T2,..., Tm una base de datode transacciones, donde Ti(i □ [1...n]) es una transacción que contiene un conjunto de items en I. El soporte (u ocurrencia) de un patrón A o conjunto de items, es el número de veces que A esta contenida en DB. A es un patrón frecuente, si el soporte de A es mayor que el umbral o soporte mínimo (£).
Así por ejemplo, sea la Base de Datos de Transaccines de la tabla 2.5. y el soporte mínimo £=2.

A partir de lo anterior se generan los itemsets frecuentes tamaño 1, para lo cual se compara el soporte de los itemsets, se encuentra los que pasan el soporte mínimo de £=2 y se los ordena descendentemente tal y como se aprecia en la tabla 2.6.

Tabla 2.5 Base de Datos de Transacciones

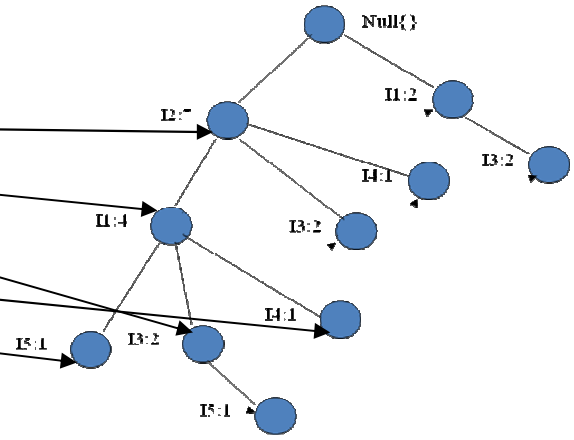
TID	Transacción	Transacción Ordenada
T100	I2, I5, I1	I1, I2, I5
T101	I2, I4	I2, I4
T102	I3, I2	I2, I3
T103	I2, I1, I4	I1, I2, I4
T104	I3, I1	I1, I3
T105	I3, I2	I2, I3
T106	I1, I3	I1, I3
T107	I2, I5, I3, I1	I1, I2, I3, I5
T108	I2, I1, I3	I1, I2, I3

Tabla 2.6. Generación Itemset frecuentes tamaño 1

Itemset	Soporte
{I2}	7
{I1}	6
{I3}	6
{I4}	2
{I5}	2

Figura 2.12. Tabla de Cabeceras y Construcción del FP-Tree

Item	Soporte	Nodo Enlace
I2	7	
I1	6	
I3	6	
I4	2	
I5	2	



Un FP-tree es una estructura que consiste de una raíz etiquetada como null, un conjunto de árboles hijos de la raíz y una tabla de cabeceras de items frecuentes. Los nodos del FP-tree tienen los campos del nombre del item que registra el item que el nodo esta representando, un contador que registra el número de transacciones representadas por la porción de la ruta que alcanzan a este nodo y finalmente los enlaces a los demás nodos, que llevan al siguiente nodo en el FP-tree.

Adicionalmente, cada entrada en la tabla de cabeceras de items frecuentes tiene dos campos, nombre del item y el primer nodo enlazado, al cual apunta la cabecera con el mismo nombre.

Para la construcción del FP-tree, primero debe leerse la base de datos con lo cuál se genera la lista de ítems frecuentes {(I2:7), (I1:6), (I3:6), (I4:2), (I5:2)} ordenados descendentemente. Posteriormente, se crea la raíz del árbol con un null.

Al leer la primera transacción se construye la primera rama del árbol {(I2:1), (I1:1), (I5:1)}. Para la segunda transacción ya que su lista de items frecuentes (I2, I4) comparte el prefijo común (I2) con la rama existente (I2, I1, I5), el conteo del nodo I2 es incrementado en 1, un nuevo nodo es creado, (I4:1) enlazado como hijo de (I2:2). Para la tercera transacción su lista de items frecuentes (I2, I3) comparte el prefijo común (I2) con la rama existente (I2, I1, I5), el conteo del nodo I2 es incrementado en 1, un nuevo nodo es creado, (I4:1) enlazado como hijo de (I2:2). Lo mismo ocurre en la cuarta transacción (I2, I1, I4) que comparte los prefijos comunes (I2, I1), el conteo de los nodos I2, I1 es incrementado en 1, un nuevo nodo es creado, (I4:1) enlazado como hijo de (I1:2). Para la siguiente transacción (I1, I3) se debe crear una nueva rama del árbol, que contiene estos dos nodos con (I1:1, I3:1). La transacción siguiente es (I2, I3), estos nodos ya se encuentran en el árbol y hacen parte de la primera rama, así es que simplemente se incrementa el conteo en 1 quedando (I2:5, I3:2). Para la transacción (I1, I3), estos nodos ya se encuentran en el árbol y hacen parte de la segunda rama, así es que simplemente se incrementa el conteo en 1 quedando (I1:2, I3:2). Para la octava transacción su lista de items frecuentes (I2, I1, I3, I5) comparte el prefijo común (I2, I1, I3) con la primera rama del árbol, el conteo de los nodos I2, I1, I3 son incrementados en 1, un nuevo nodo es creado, (I5:1) enlazado como hijo de (I3:2). Finalmente, para la transacción (I2, I1, I3), estos nodos ya se encuentran en el árbol y hacen parte de la primera rama, así es que simplemente se incrementa el conteo en 1 quedando (I2:7, I1:4, I3:3).

Para mejorar el funcionamiento del árbol una Tabla de Cabeceras es construida, en la cual cada item apunta a su ocurrencia en el árbol, además todos los nodos que tengan el mismo nombre deben ser enlazados en secuencia. Finalmente, se obtiene el árbol resultante de la figura 2.12.

Existen ciertas propiedades del FP-tree que facilitan la tarea de Minería de Patrones Frecuentes:

Propiedad de nodos enlazados: Para cualquier nodo frecuente *ai*, todos los posibles Patrones Frecuentes que contenga ai pueden ser obtenidos siguiendo los nodos enlazados de *ai*, comenzando desde *ai* en la tabla de cabeceras de items frecuentes.

Comenzando por los nodos enlazados de I5 su Patrón Frecuente resultante es (I5:2) y sus dos rutas en el FP-Tree son (I2:7, I1:4, I5:1) y (I2:7, I1:4, I3:2, I5:1). Así por

ejemplo, la primera ruta indica que la cadena (I2,I1,I5) aparece dos veces en la base de datos. Para saber que cadena aparece con I5, únicamente cuenta el prefijo de I5: (I2:1, I1:1) para el primer caso y (I2:1, I1:1, I3:1) para el segundo caso. Estos dos prefijos de I5, forman los Sub-patrones Base de I5 o también llamados Patrones Condicionales Base.

La construcción de un FP-tree sobre este patrón condicional base lleva a únicamente una rama (I2I1:2), la cual será su patrón condicional, tal y como se lo aprecia en la figura 2.13. Así que finalmente el patrón frecuente es (I2I1I5:2).

Figura 2.13. Patrón Condicional

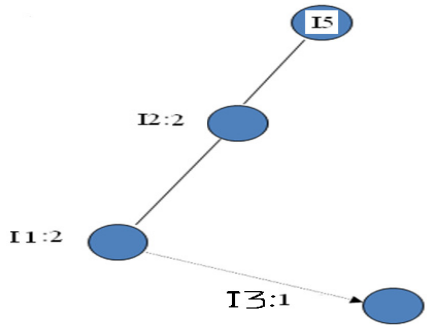


Tabla 2.7. Construcción de los Patrones Condicionales

Ítem	Patrones Condicionales Base	Patrones Condicionales	Patrones Frecuentes
5	{(I2I1:1) , (I2I1I3:1)}	(I2:2, I1:2)	(I2I5:2), (I1I5:2), (I2I1I5:2)
4	{(I2I1:1) , (I2:1)}	(I2:2)	(I2I4:2)
3	{(I2I1:2) , (I2:2) , (I1:2)}	(I2:4, I1:2), (I1:2)	(I2I3:4), (I1I3:2), (I2I1I3:2) => (I2I3:4), (I1I3:4), (I2I1I3:2)
1	{(I2:4)}	(I2:4)	(I2I1:4)

Itemsets frecuentes tamaño 3: {i2i1i5} {i2i1i3}

Itemsets frecuentes tamaño 2: {i2i5} {i1i5} {i2i4} {i2i3}{i1i3}{i2i1}

Itemsets frecuentes tamaño 1: {i5} {i4} {i3} {i1}{i2}

Equipasso: Un Algoritmo para el descubrimiento de reglas de asociación basado en operadores algebraicos

Equipasso [38], es un algoritmo para el cálculo de conjunto de ítems frecuentes basados en dos nuevos operadores del álgebra relacional: *Associator y Equikeep.*

Associator (α): Es un operador algebraico unario que a diferencia del operador Selección o Restricción (s), aumenta la cardinalidad de una relación. *Associator* genera, a partir de cada tupla de una relación, todas las posibles combinaciones de los valores de sus atributos, como tuplas de una nueva relación, conservando el esquema de la relación inicial.

Su sintaxis es la siguiente: a tamaño_inicial, tamaño_final(R); donde <tamaño_inicial > y <tamaño-final> son dos parámetros de entrada que determinan el tamaño inicial y tamaño final de las combinaciones.

El operador ***Associator*** genera, por cada tupla de la relación *R*, todos sus posibles subconjuntos (conjuntos de items) de diferente tamaño. ***Associator*** toma cada tupla ***t*** de ***R*** y dos parámetros: <tamaño_inicial> y <tamaño_final> como entrada, y retorna, por cada tupla ***t,*** las diferentes combinaciones de atributos Xi, de tamaño <tamaño_inicial> hasta tamaño <tamaño_final>, como tuplas en una nueva relación. El orden de los atributos en el esquema de *R* determina los atributos en los subconjuntos con valores, el resto se hacen nulos.

El tamaño máximo de un itemset y por consiguiente el tamaño final máximo (<tamaño_final>) que se puede tomar como entrada es el correspondiente al valor del grado de la relación.

Formalmente, sea A= {A1,..., An} el conjunto de atributos de la relación R de grado n y cardinalidad m, IS y ES el tamaño inicial y final respectivamente de los subconjuntos a calcular. El operador α aplicado a R

$$\alpha_{IS,\,ES}(R) = \{ \cup_{all}\ X_i \mid X_i \subseteq t_i, t_i \in R\ ,\ \forall_i\ \forall_k\ (X_i = <v_i(A_1), v_i(A_2), null,\ ..., v_i(A_k), null>,\ v_i(A_k) <> null),\ (i = (2^n - 1) * m),\ (k = IS\ ..ES),\ A_1 < A_2 < ... < A_k\ ,\ IS = 1,\ ES = n \}$$

produce una nueva relación cuyo esquema *R(A)* es el mismo de R de grado *n* y cardinalidad *m'= (2n -1) * m* y cuya extensión r(A) está formada por todos los subconjuntos Xi generados a partir de todas las combinaciones posibles de los valores no nulos vi(Ak) de los atributos de cada tupla ti de R. En cada tupla Xi únicamente un grupo de atributos mayor o igual que IS y menor o igual que ES tienen valores, los demás atributos se hacen nulos.

Ejemplo 1: Sea la relación R(A, B, C, D). Encontrar las diferentes combinaciones de tamaño 2 hasta tamaño 4, es decir R1=α2,4(R**).**

Figura 2.14. Relación R

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2

Figura de la Operación R1=α2,4(R**).**

A	B	C	D
a1	b1	null	Null
a1	Null	c1	Null
a1	Null	null	d1
null	b1	c1	Null
null	b1	null	d1
null	Null	c1	d1
a1	b1	c1	Null
a1	b1	null	d1
a1	Null	c1	d1
null	b1	c1	d1
a1	b1	c1	d1
a1	b2	null	Null
a1	Null	c1	Null
a1	Null	null	d2
null	b2	c1	Null
null	b2	null	d2
null	Null	c1	d2
a1	b2	c1	Null
a1	b2	null	d2
a1	Null	c1	d2
null	b2	c1	d2
a1	b2	c1	d2

Equikeep (χ). Es un operador unario, que se asemeja a la selección o restricción por tener una expresión lógica que evaluar sobre una relación R, y conserva su esquema. Se diferencia de la Restricción en que en lugar de aplicar la condición a las filas (tuplas) de la relación, EquiKeep aplica la expresión lógica a las columnas (atributos) de R, es decir restringe los valores de los atributos de cada una de las

tuplas de la relación *R,* a únicamente aquellos que satisfacen una condición determinada, haciendo nulos al resto de valores y conserva el esquema de la relación.

Su sintaxis es la siguiente: $\chi_{\text{expresión\_lógica}}(R)$, donde <expresión lógica> es la condición que deben cumplir los valores de los atributos de la relación R para no hacerse nulos. El operador ***EquiKeep*** restringe los valores de los atributos de cada una de las tuplas de la relación *R* a únicamente los valores de los atributos que satisfacen la expresión lógica <*expresión_lógica*>, formada combinando cláusulas de la forma *Atributo*=*Valor*, conectivos lógicos AND, OR y NOT. En cada tupla, los valores de los atributos que no cumplen la condición <*expresión_lógica*> se hacen nulos. ***EquiKeep*** elimina las tuplas vacías, es decir, aquellas tuplas en las cuales los valores de todos los atributos son nulos.

Formalmente, sea A= {A1,..., An} el conjunto de atributos de la relación *R* de esquema R(A), de grado *n* y cardinalidad *m*. Sea *p* una expresión lógica formada por cláusulas de la forma *Ai* = *const,* unidas por los operadores booleanos AND (^), OR (v), NOT (¬). El operador **χ** aplicado a la relación *R* con la expresión lógica *p*

Xp(R)={ ti(A) | $\forall i \forall j$ (p(vi(Aj))= vi(Aj) si p =true y p(vi(Aj))= null si p =false), i=1..m', j=1..n, m' ≤ m}, produce una relación de igual esquema R(A) de grado *n* y cardinalidad *m'*, donde m' ≤ m. En su extensión, cada n-tupla *ti*, esta formada por los valores de los atributos de R, vi(Aj), que cumplan la expresión lógica p, es decir *p*(vi(Yj)) es verdadero, y por valores nulos si p(vi(Yj)) es falso.

Ejemplo 2**.** Sea la relación R(A, B, C, D) y sea R1=c A=a1 v B=b1 v C=c2 v D=d1 (R).

Figura 2.16. Relación R

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2
a2	b2	c2	d2
a2	b1	c1	d1
a2	b2	c1	d2
a1	b2	c2	d1

Figura 2.17. Resultado operación R1=c A=a1 v B=b1 v C=c2 v D=d1 (R).

A	B	C	D
a1	b1	null	d1
a1	null	null	Null
null	null	c2	Null
null	b1	null	d1
null	null	null	Null
a1	null	c2	d1

En este ejemplo, la tupla {a2, b2, c1, d2} se eliminan por resultar todos sus valores nulos.

Algoritmo Equipasso. El primer paso del algoritmo cuenta el número de ocurrencias de cada item para determinar los 1-conjuntos de items frecuentes L1. En el subsiguiente paso, se aplica el operador ***EquiKeep*** para extraer de todas las transacciones de D, los conjuntos de items frecuentes tamaño 1, haciendo nulos el resto de valores. Luego, a la relación resultante R, se aplica el operador ***Associator*** para generar todos los conjuntos de items tamaño 2 (Is=2) hasta máximo tamaño n, donde n es el grado de D. Finalmente, se calculan todos los conjuntos de items frecuentes L, contando el soporte de las diferentes combinaciones generadas por ***Associator*** en la relación R'.

Figura 2.18. Algoritmo Equipasso

L1={1-conjuntos de items frecuentes};
Forall transacciones t Î D **do begin**
R=cL1(D) //Se aplica el operador EquiKeep
k=2
g=grado(R)
R'=ak,g (R) ={Èall Xi ½ Xi Í ti } //Genera todos los conjuntos de items posibles
End
L = { count (R')½ count ³ minsup} // Conjuntos de items frecuentes

Algoritmo C4.5

El algoritmo C4.5 surge por la ineficiencia del algoritmo ID3 al tratar con valores numéricos. Lo propuso Quinlan en 1993, como extensión de ID3. Es recursivo, y se basa en la estrategia "divide y vencerás" (aplicando el algoritmo a conjuntos de datos cada vez más pequeños). Pertenece a la familia TDIDT (Top Down Induction Decision Trees).

El algoritmo C4.5 permite la construcción de un árbol de decisión, que es un diagrama que representan en forma secuencial condiciones y acciones; muestra qué condiciones se consideran en primer lugar, en segundo lugar y así sucesivamente. Se construyen a partir de la descripción narrativa del problema.

Un árbol de decisión sirve para modelar funciones discretas, en las que el objetivo es determinar el valor combinado de un conjunto de variables, y basándose en el valor de cada una de ellas, determinar la acción a ser tomada. Además de eso, son una alternativa de visualización gráfica de resultados donde se pueden observar las variables evaluadas, las acciones a tomar y el orden de las mismas.

C4.5 permite:

- Incorporación de atributos tanto discretos como continuos.
- Utilización de la ganancia proporcional (gain ratio), en caso de que la ganancia no sea conveniente como heurística en la tarea concreta que se esté estudiando.
- Método de postprunning, para evitar el sobreajuste.
- Método probabilístico para solucionar el problema de los atributos con valor desconocido.

El algoritmo recibe las siguientes entradas:

- Fichero .target: Enumeración de las posibles clases, y nombre y tipo de los atributos.
- Fichero .data: Conjunto de entrenamiento (conjunto de casos clasificados).

El algoritmo genera la siguiente salida:

- Árbol de decisión.
- Opcionalmente, el árbol de decisión podado.
- Reglas de producción.

Algoritmo Mate-Tree

EL algoritmo Mate-Tree [40] se basa en los operadores algebraicos Mate, Entro, Gain y Describe Classifier que serán descritos a continuación.

El operador ***Mate*** (μ) genera, por cada una de las tuplas de la relación *R*, todas los posibles combinaciones formadas por los valores no nulos de los atributos pertenecientes a la <lista_atributos> y el valor no nulo del atributo <atributo_clase>. ***Mate*** toma como entrada cada tupla de *R* y produce una nueva relación con tuplas formadas por todas las posibles combinaciones de los atributos de la lista de atributos <lista_atributos> con el atributo clase <atributo_clase>, los demás valores de los atributos se hacen nulos.

Su sintaxis es: $\mu_{\text{lista atributos; atributo clase}}(R)$

Formalmente, sea A={A1, . . ., An} el conjunto de atributos de la relación *R* de grado *n* y cardinalidad *m*, $L \subset A$, $L \neq \varphi$ la lista de atributos (<*lista_atributos*>) a combinar y $|L| = n'$, $n' < n$, $Ac \in A$, $Ac \cap L = \varphi$ el atributo <*atributo _clase*> con el que se combinarán los atributos de L. El operador μ aplicado a la lista de atributos L, al atributo clase Ac de la relación R:

$$\mu_{L;\,Ac}(R) = \{\, ti(A) \mid L \subset A,\ |L| = n',\ n' < n,\ Ac \in A,\ Ac \cap L = \phi,\ ti = Xi,\ ),\ i = 1..m',\ m' = (2^{n'} - 1) * m,\ \forall_i \forall_k (\, X_i = <null, \ldots, v_i(A_k) \ldots, null, \ldots, v_i(Ac)>,\ v_i(A_k)\, v_i(Ac) <> null),\ k = 1..n' \,\}$$

produce una relación cuyo esquema es R(A) , de grado n y cuya extensión r(A) de cardinalidad m', m'=(2n' -1)* m es el conjunto de n-tuplas *ti,* tal que en cada n-tupla únicamente los atributos que forman la combinación Xi ∈ L (k =1..n') y el atributo Ac tienen valor, el resto de atributos se hacen nulos.

Ejemplo 3: Sea la relación R(A, B, C, D):

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2

Figura 2.19. Resultado de la operación μA,B,C,D (R)

A	B	C	D
a1	null	null	d1
null	b1	null	d1
null	null	c1	d1
a1	b1	null	d1
a1	null	c1	d1
null	b1	c1	d1
a1	b1	c1	d1
a1	null	null	d2

Función Algebraica Agregada Entro

Es conveniente en este punto especificar algunos conceptos como lo son los conceptos de entropía y ganancia.

El concepto básico de entropía en Teoría de la Información tiene mucho que ver con la **incertidumbre que existe en cualquier experimento** o señal aleatoria, es decir, la cantidad de información que posee una señal.

Shannon ofrece una definición de entropía que satisface las siguientes afirmaciones:

- La medida de información debe ser proporcional (continua). Es decir, el cambio pequeño en una de las probabilidades de aparición de uno de los elementos de la señal debe cambiar poco la entropía.
- Si todos los elementos de la señal son equiprobables a la hora de aparecer, entonces, la entropía será máxima.
- Si hay n mensajes con la misma probabilidad de ocurrencia, entonces, la probabilidad p de cada uno es 1/n y la información proporcionada por un mensaje es:

$$-\text{Log2}(p) = \log2(n)$$

$$I(p) = -(p1*\log(p1) + p2*\log(p2)+\ldots+pn*\log(pn))$$

Si un conjunto de T registros se divide en clases inconexas C1, C2,…,Ck, entonces la información necesaria para evaluar la clase de un elemento de T es Info(T)=I(P), donde P es la distribución de probabilidades de la partición (C1, C2,…,Ck):

$$P = (|C1| /T, |C1|/T, \ldots, |Ck)/T)$$

Si primero se divide T en la base de un valor no categórico X en conjuntos T1,T2,…,Tn, entonces, la información que se necesita para identificar la clase de un elemento de T es:

$$\text{Info}(X,T) = \sum_{i=1}^{n} \frac{|T_i|}{T} \cdot \text{Info}(T_i)$$

La ganancia representa la diferencia entre la información necesaria para identificar la clase de un elemento de T y la información necesaria para hacerlo después de obtener un valor de X, esto es, la ganancia en información debida al atributo X.

Se puede utilizar esta noción de ganancia para clasificar atributos y para construir árboles de decisión donde se localiza en cada nodo el atributo con mayor ganancia entre los atributos todavía no considerados desde el nodo raíz.

$$\text{Ganancia}(X,T) = \text{Info}(T) - \text{Info}(X,T)$$

Aclarados estos términos ya se puede proceder a explicar la función Entro. Función Algebraica Agregada Entro permite calcular la entropía de una relación R con respecto a un atributo denominado atributo condición y un atributo clase.

Tiene la siguiente sintaxis: Entro (Atributo; Atributo clase; R), donde atributo es el atributo condición de la relación R y atributo clase es el atributo con el que se combina el atributo.

Formalmente, sea A = {A1,..., An, Ac} el conjunto de atributos de la relación R o con esquema R(A), extensión r(A), grado n y cardinalidad m. Sea t el número de distintos valores del atributo clase Ac, Ac[]R(A) que divide a r(A) en t diferentes clases, Ci(i = 1..t). Sea ri el número de tuplas de r(A) que pertenecen a la clase Ci. Sea q el número de destintos valores {v1 (Ak), v2 (Ak), .., vq (Ak)} del atributo Ak , Ak []R(A), el cual particiona a r(A) en q subconjuntos {S1 , S2 , ...Sq }, donde Sj contiene todos las tuplas de r(A) que tienen el valor vj (Ak) del atributo Ak . Sea sij el número de tuplas de la clase Ci en el subconjunto Sj .

La función Entro (Ak ; Ac; R), retorna la entropía de R con respecto al atributo Ak, que se obtiene de la siguiente manera :

Entro(Ak; Ac; R) = {y | y = − pij log2 (pij), i = 1..t, j = 1..q, pij = sij / |Sj |}, donde pij = sij / |Sj es la probabilidad que una tupla en Sj pertenezca a la clase Ci.

La entropía de R con respecto al atributo clase Ac es:
Entro(Ac; Ac; R) = {y | y = − pi log2 (pi), i = 1..t, pi = ri /m}, donde p_i es la probabilidad que un tupla cualquier pertenezca a la clase C_i y r_i el número de tuplas de r(A) que pertenecen a la clase C_i .

Función Algebraica Agregada Gain

Función Algebraica Agregada Gain permite calcular la reducción de la entropía causada por el conocimiento del valor de un atributo de una relación.

Su sintaxis es: Gain (atributo; atrib clase; R), donde atributo es el atributo condición de la relación R y atributo clase es el atributo con el que se combina el atributo. La función Gain() permite calcular la ganancia de información obtenida por el particionamiento de la relación R de acuerdo con el atributo.
Se define como: Gain(Ak; Ac; R) = {y | y = Entro(Ac; Ac; R) − Entro(Ak; Ac; R)}

Operador Describe Classifier

El Operador Describe Classifier ([]μ), es un operador unario que toma como entrada la relación resultante de los operadores Mate By, Entro() y Gain() y produce una nueva relación donde se almacenan los valores de los atributos que formarán los diferentes nodos del árbol de decisión.

La sintaxis del operador Describe Classifier es la siguiente: []μ(R)

Formalmente, sea A = {A1,.., An, E, G} el conjunto de atributos de la relación R de grado n + 2 y cardinalidad m.
El operador [][] aplicado a R:

[][](R) = {ti (Y)[] Y = {N.P, A.V, C},
Si ti = val(N), null, val(A), null, null [] raíz,
Si ti = val(N), val (P), val(A), val (V), val(C) [] hoja,

Si ti =val(N), val (P), val(A), val (V), null [] nodo interno}

Produce una nueva relación con esquema R (Y), Y=N, P, A, V, C donde N es el atributo que identifica el número de nodo, P identifica el nodo padre, A identifica el nombre del atributo asociado a ese nodo, V es el valor del atributo A y C es el atributo clase.

Su extensión r (Y), está formada por un conjunto de tuplas en las cuales si los valores de los atributos son:

Si N null, P = null, A null, V = null y C = null corresponde a un nodo raíz;
Si N null, P null, A null, V null y C null corresponde a una hoja o nodo terminal y
Si N null, P null, A null, V null y C = null corresponde a un nodo interno.

Describe Classifier facilita la construcción del árbol de decisión y por consiguiente la generación de reglas de clasificación.

Ejemplo 4. Sea la relación: SINTOMAS (SID, D_CABEZA,D_MUSCULAR,TEMPERATURA, GRIPA).

SID	D_CABEZA	D_MUSCULAR	TEMPRATURA	GRIPA
1	No	Si	Alta	Si
2	Si	No	Alta	Si
3	Si	Si	Media	No
4	No	Si	Normal	Si
5	Si	No	Media	No
6	No	No	Normal	No
7	Si	No	Normal	No
8	Si	Si	Alta	Si

La relación del árbol que se obtiene se puede apreciar en la tabla 2.8. Paso seguido, agrupando los datos, contando la frecuencia de las parejas y proyectando los atributos D_Cabeza, D_Muscular, Temperatura y Gripa se obtiene la tabla 2.9 CLASESINTOMAS.

Tabla 2.8. Relación Árbol

D_CABEZA	D_MUSCULAR	TEMPERATURA	GRIPA
No	Null	Null	Si
Null	Si	Null	Si
Null	Null	Alta	Si
No	Si	Null	Si
No	Null	Alta	Si
Null	Si	Alta	Si
No	Si	Alta	Si
Si	Null	Null	Si
Null	No	Null	Si
Null	Null	Alta	Si
Si	No	Null	Si
Si	Null	Alta	Si
Null	No	Alta	Si

Null	No	Normal	No
No	No	Normal	No
Si	Null	Null	No
Null	No	Null	No
Null	Null	Normal	No
Si	No	Null	No
Si	Null	Normal	No
Null	No	Normal	No
Si	No	Normal	No
Si	Null	Null	Si
Null	Si	Null	Si
Null	Null	Alta	Si
Si	Si	Null	Si
Si	Null	Alta	Si
Null	Si	Alta	Si
Si	Si	Alta	Si

Tabla 2.9. CLASESINTOMAS: Resultado Mate y Agrupamiento

D_CABEZA	D_MUSCULAR	TEMPERATURA	GRIPA	COUNT
		Alta	Si	3
		Media	No	2
		Normal	Si	1
		Normal	No	2
	No		Si	1
	No		No	3
	No	Alta	Si	1
	No	Media	No	1
	No	Normal	No	2
	Si		Si	3
	Si		No	1
	Si	Alta	Si	2
	Si	Media	No	1
	Si	Normal	Si	1
No			Si	2
No			No	1
No		Alta	Si	1
No		Normal	Si	1
No		Normal	No	1
No	No		No	1
No	No	Normal	No	1
No	Si		Si	2
No	Si	Alta	Si	1
No	Si	Normal	Si	1
Si			Si	2
Si			No	3
Si		Alta	Si	2
Si		Media	No	2
Si		Normal	No	1
Si	No		Si	1
Si	No		No	2
Si	No	Alta	Si	1
Si	No	Media	No	1

Si	No	Normal	No	1
Si	Si		No	1
Si	Si		Si	1
Si	Si	Alta	Si	1
Si	Si	Media	No	1

Sea S la clase decisión (Tiene Gripa), formada por 8 objetos los cuales incluyen 4 objetos positivos (Si) y cuatro objetos negativos (No).

La entropía de S [4+,4-] es: Entro(S)=- (4/8) log2(4/8)- (4/8)log2(4/8)=-0.5(-1)- 0.5(-1)=1

Se calcula la ganancia de información de los atributos condición con respecto a la clase decisión.

Se inicia con el atributo temperatura: Valores = alta, media, normal

Temperatura= alta y gripa= si, count= 3
Temperatura=alta y gripa=no, count =0, es decir Salta=[3+,0-]
Temperatura= media y gripa= si, count= 0
Temperatura=media y gripa=no, count=2, es decir Smedia=[0+,2-]
Temperatura=normal y gripa=si, count=1;
Temperatura=normal y gripa=no, count=2, es decir Snormal=[1+,2-]

Gain(S, temperatura)=Entro(S) -3/8*Entro(Salta) -2/8*entro(Smedia)-3/8*entro(Snormal)= 1 – (3/8)*0 – (2/8)* 0 – (3/8) * 0.9182= 1 – 0.3443 = 0.6556

Para el atributo Dolor muscular: Valores = Si, No
D_muscular=si y gripa=si, count=3;
D_muscular=si y gripa=no, count=1, es decir: Ssi=[3+,1-].

Para D_muscular=no y gripa=si, count=1;
D_muscular=no y gripa=no, count=3, es decir: Sno=[1+,3-]

Gain(S,D_muscular)=Entro(S) -4/8*entro(Ssi) – 4/8 entro(Sno) = 1 – (4/8) *0.8112 – (4/8)*0.8112 = 1- 0.4056 -0.4056= 1- 0.8112=0.188

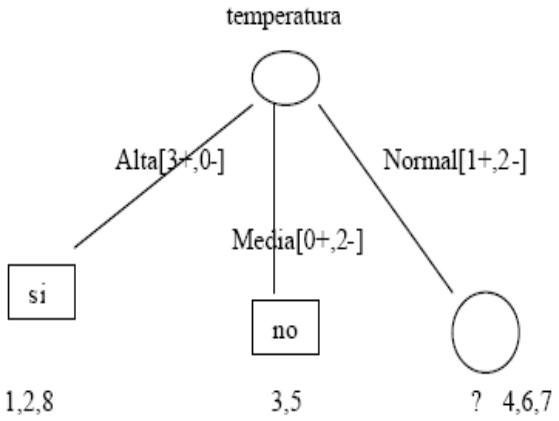
Para el atributo Dolor de cabeza:
Valores= si, no

D-cabeza=si y gripa=si, count=2;
D_cabeza=si y gripa=no, count=3, es decir Ssi=[2+,3-]
D-cabeza=no y gripa=si, count=2;
D_cabeza=no y gripa=no, count=1, es decir Sno=[2+,1-]

Gain(S,D_cabeza) =Entro(S) -5/8*entro(Ssi) – 3/8 entro(Sno) = 1 – (5/8) *0.9709 – (3/8)*0.9182 = 1- 0.6068 -0.3443= 1- 0.9511=0.0488

De acuerdo a la ganancia de información, el atributo que mayor ganancia tiene es *temperatura*, por lo tanto este atributo se selecciona como el nodo raíz, como se aprecia en la figura 2.20.

Figura 2.20. Resultado parcial Árbol de Decisión.



Para cada objeto donde *temperatura =alta*, se presenta una clasificación positiva sobre la clase decisión Gripa, por lo tanto el nodo esta formado por un nodo hoja en la cual la clasificación Gripa=si y el valor de la entropía =0.

Lo mismo pasa para cada objeto donde *temperatura = mediana*, donde todos son de la misma clase, es decir gripa=no y su entropía=0. Por lo tanto en la rama mediana se crea un nodo hoja Gripa=no.

Para el caso *de temperatura=normal*, se debe continuar con la construcción del árbol, utilizando solo los datos de entrenamiento asociados a este nodo:

Ahora el conjunto S es Snormal=[1+,2-] y entro(Snormal)=0.9182

Se calcula la ganancia de la rama temperatura=normal con respecto a los otros atributos:

Para el atributo D_muscular
Valores= Si, No
D_muscular=si, temperatura=normal y gripa=si, count=1
D_muscular=si, temperatura=normal y gripa=no, count=0, es decir Ssi=[1+,0-]
D_muscular=no, temperatura=normal y gripa=si, count=0,
D_muscular=no, temperatura=normal y gripa=no, count=2,
es decir Sno=[0+,2-].

Gain(Snormal, D_muscular) = entro(Snormal) – (1/3)*entro(Ssi) – (2/3)* entro(Sno) = 0.9182 – (1/3)* (0) – (2/3)* (0) = 0.9182 -0=0.9182.

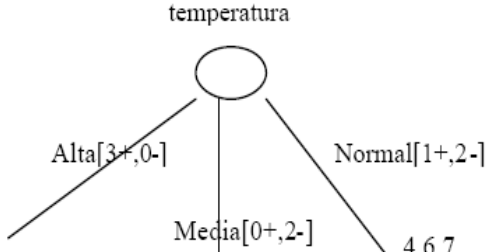
Para el atributo D_cabeza
Valores= Si, No

Temperatura:
D_cabeza=si, temperatura=normal y gripa=si , count=0
D_cabeza=si, temperatura=normal y gripa=no , count=1 , es decir Ssi=[0+,1-]
D_cabeza=no, temperatura=normal y gripa=si, count=1
D_cabezar=no, temperatura=normal y gripa=no, count=1, es decir Sno=[1+,1-]

Gain(Snormal, D_cabeza) = entro(Snormal) – (1/3)*entro(Ssi) – (2/3)* entro(Sno)
= 0.9182 – (1/3)* (0) - (2/3)* (1) = 0.9182 -0.6666 =0.2516

Al tener el atributo D_muscular mayor ganancia, se selecciona D_ muscular en este nodo, tal y como se lo aprecia en la figura 2.21.

Figura 2.21. Árbol de Decisión



Algoritmo K- Medias

El algoritmo de las K-medias [15], es el algoritmo de aprendizaje no supervisado más sencillo que resuelve problemas de *Clustering* bien conocidos. Este algoritmo utiliza los *centroides* de cada grupo para ir formando los diferentes grupos iterativamente.

Sin embargo, requiere que los patrones sean numéricos para poder determinar centroides y no se puede utilizar cuando se conocen los patrones sino sólo la similitud entre ellos, como ocurre cuando existen características de tipo cualitativo (atributos).

La entrada del sistema puede ser un conjunto de p patrones o individuos de una población, y cada uno de ellos constituido por N características dadas, junto con una medida de similitud que se establece entre dichos patrones y el número de grupos a formar. Además de la formación de los grupos el sistema puede determinar los patrones prototipo de cada grupo.

K-medias es utilizado para encontrar los k puntos más densos en el conjunto de datos, el algoritmo k-means consiste en:

Inicio

1. Se seleccionan aleatoriamente k centros
2. Repetir

- Asignar cada ejemplo al conjunto con el centro más cercano
- Calcular los puntos medios de los k conjuntos

Mientras los conjuntos no varíen
3. Devolver los k centros

Fin Algoritmo

Determinar k no es una tarea fácil y pueden presentarse los siguientes casos:
- Si k es muy pequeño, hay grupos que se quedan sin centro
- Si k es muy grande, hay centros que se quedan huérfanos
- Incluso con k exacto, puede haber algún centro que quede huérfano

El valor de k se suele determinar heurísticamente, el número de grupos obtenidos depende de los centros iniciales, con lo que el algoritmo se puede ejecutar con distintas semillas.

Medidas de Similitud

• **Distancia del Coseno Normalizada:** Dados dos vectores q y d la distancia del coseno se define como:

$$sim(q,d) = \frac{\sum_{i=1}^{n}(q_i \cdot d_i)}{\sqrt{\sum_{i=1}^{n} q_i^2} \cdot \sqrt{\sum_{i=1}^{n} d_i^2}} = (\|q\|^{-1} \cdot q) \bullet (\|d\|^{-1} \cdot d)$$

• **Distancia Euclidiana:** Se denomina distancia euclídea entre dos puntos A(x1,y1) y B(x2,y2) a la longitud del segmento de recta que tiene por extremos A y B. Se expresa matemáticamente como:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Estructura general del algoritmo k-medias

1. Sitúa K puntos en el espacio representado por los objetos que van a ser agrupados.
2. Estos puntos representarán el grupo inicial de centróides.
3. Asignar cada objeto al grupo cuyo centróide sea el más cercano al objeto.

4. Cuando todos los objetos hayan sido asignados, recalcular la posición de los k centróides.
5. Repetir los pasos 2 y 3 hasta que los centróides no varíen (Esto producirá una separación de los objetos dentro de los grupos en los que la distancia entre los miembros ha sido minimizada.)

Una vez preparados los datos de entrada se debe:

- Seleccinar del número de grupos a realizar (k) y la agrupación deseada
- Establecer de los centros iniciales, que se eligen de manera aleatoria de entre los posibles ejemplos
- Aplicacar el algoritmo:
    - Paso1: Mientras varíen los centros:
        - Paso1.1: Calcular la medida de Similitud
        - Paso1.2: Asignar los ejemplos a los centros
        - Paso1.3: Recalcular los centros
- Establecer los patrones representativos de cada grupo

Lo más importante de todo esto es la elección de las distintas operaciones y el establecimiento de la medida de similitud dependiendo de la operación elegida, que son:

Operaciones que implican páginas web. En función de las páginas que han sido visitadas en la misma sesión, las páginas que se visitan y el día de la semana, las páginas de usuario y la hora a la que empiezan la sesión y las páginas visitadas y el tiempo total de sesión.

Para este grupo las características principales son las siguientes: Entra en consideración el concepto de **soporte** y sólo se considerarán válidas aquellas páginas que lo superen.

La medida de similitud usada para este grupo se divide en tres posibles enfoques:

1. **Distancia del coseno normalizada**. Esta es la medida utilizada para establecer la similitud entre dos sesiones de usuario en lo referente a las visitas web realizadas, el proceso consiste en:

   Se consideramos las **páginas visitadas** en una sesión de usuario como un **vector,** ejemplo:

<0,1> Vector de visitas
<0,0> Vector de centro a comparar

Para medir el grado de similitud se lo hace calculando la **distancia del coseno normalizada** según la siguiente fórmula:

$$\cos(\vec{u}, \vec{v}) = \frac{x_1x_2 + y_1y_2 + z_1z_2}{\sqrt{x_1^2 + y_1^2 + z_1^2} - \sqrt{x_2^2 + y_2^2 + z_2^2}}$$

De esto se obtiene un número comprendido entre [-1,1] pero este valor no es útil para continuar con este método, por lo cual una vez obtenido se normaliza el intervalo a un intervalo entre [0,1] la fórmula empleada para esto es la siguiente:

$$V_i = \frac{a_i - \min a_i}{\max a_i - \min a_i}$$

Se aplica esta medida de similitud y se aproxima el conjunto de ejemplos al centro con el que tengan la menor distancia. Para recalcular el centro se aplica la **media**, es decir, para cada coordenada del vector se suma el valor para éste en esa posición y se lo divide entre el número total de sesiones de las que se dispone para el grupo recién creado.

2. Distancia Euclídea: Para los valores continuos (tiempo total de sesión) esta es la medida que se utiliza, el método es análogo al anterior, una vez establecido el valor y el valor del centro a comparar se aplica la siguiente fórmula:

De lo cual se obtiene: $\delta^2 E(X_i, X_j) = \| X_i - X_j \|^2 = (X_i - X_j)^T (X_i - X_j)$

$$\delta^2 E(X_i, X_j) = (X_{ik} - X_{jk})^2$$

Se aplica esta medida de similitud y se aproxima el conjunto de ejemplos al centro con el que tengan la menor distancia. Para recalcular el centro se aplica la **media**, es decir, para cada coordenada del vector se suma el valor para éste en esa posición

y se lo divide entre el número total de sesiones de las que se dispone para el grupo recién creado.

3. Distancia de valores discretos: Para los valores discretos la forma de actuar es ligeramente distinta, se calcula la medida de similitud de la siguiente manera:
La distancia(x,y) es 1 si x=y, y es 0 si x≠y.

Para recalcular el centro se aplica la **moda**, es decir, para cada coordenada del vector se sume el número de elementos que hay para este valor en esa posición y se comprueba cual es el más repetido para dicha coordenada, siendo ese el que se establece como nuevo valor para el centro.

Operaciones que Implican Valores Continuos. En función del tiempo total de sesión y el número de solicitudes. Estos son atributos compuestos de valores continuos y como tales deben normalizarse en datos comprendidos en datos comprendidos entre 0 y 1 (ambos inclusive).

La medida de similitud usada para este grupo se divide en tres posibles enfoques:

1. **Distancia Euclídea:** Para los valores continuos (tiempo total de sesión) esta es la medida que se utiliza, el método es análogo al Clustering anterior, una vez establecido el valor y el valor del centro a comparar se aplica:

$$\delta^2 E\ (X_i, X_j) = \| X_i - X_j \|^2 = (X_i - X_j)^T (X_i - X_j)$$

De lo cual se obtiene:

$$\delta^2\ E\ (X_i, X_j) = (X_{ik} - X_{jk})^2$$

Se aplica esta medida de similitud y se aproxima el conjunto de ejemplos al centro con el que tengan la menor distancia. Para recalcular el centro se aplica la **media**, es decir, para cada coordenada del vector se suma el valor para éste en esa posición y se lo divide entre el número total de sesiones de las que se dispone para el grupo recién creado.

2. Distancia de valores discretos: Para los valores discretos la forma de actuar es ligeramente distinta, se calcula la medida de similitud de la siguiente manera:
La distancia(x,y) es 1 si x=y, y es 0 si x≠y. Para recalcular el centro se aplica la **moda**, es decir, para cada coordenada del vector se sume el número de elementos que hay para este valor en esa posición y se comprueba cual es el más repetido

para dicha coordenada, siendo ese el que se establece como nuevo valor para el centro.

Es preciso aclarar en este momento que se han dado las bases teóricas de la tarea de Agrupación o Clustering con el algoritmo K-medias, sin embargo la implementación del mismo se posterga para un trabajo futuro.

2.4.2. Algoritmo de Minería Web de Uso HPG. Hypertext Probabilistic Grammar HPG [25], es un nuevo modelo que se ocupa del problema de minar datos logs que directamente se capturan de la semántica de las sesiones de navegación de los usuarios.

Las rutas grabadas de las navegaciones de los usuarios son modeladas, como una gramática probabilística de hipertexto cuya probabilidad más alta genera las rutas usadas por los usuarios. Este modelo tiene las ventajas de ser autónomo, compacto y basado en la teoría de gramáticas probabilísticas.

Hay dos contextos en que tal modelo es potencialmente útil. Por una parte puede ayudar al proveedor de servicio para entender lo que el usuario necesita y como resultado, mejorar la calidad de su servicio. La calidad del servicio puede ser mejorada proporcionando páginas que se adapten a los requerimientos del usuario, construyendo de antemano páginas dinámicas para reducir el tiempo de espera, o proporcionando un servicio especulativo que envía, además del documento pedido, varios otros documentos que se espera que serán solicitados en un futuro.

Por otro lado, teniendo un escenario en dónde el navegador del usuario está configurado para coleccionar los log files, se puede tener datos que caracterizan la interacción del usuario con la web permitiendo la actualización incremental de una gramática probabilística de hipertexto. Dicha gramática será la representación del conocimiento que los usuarios tienen de la web y puede ser utilizada como una herramienta para la predicción de las páginas que un usuario encuentra interesante.

Como ya se ha mencionado anteriormente un archivo log puede verse como un conjunto de demandas a páginas web realizadas por un usuario de manera que es posible inferir las sesiones de navegación del mismo. Una sesión de navegación de usuario es una sucesión de páginas solicitadas tal que no hay dos páginas

consecutivas que estén separadas por más de X minutos, dónde X es un parámetro que debe ajustarse para la característica específica de cada sitio web.

Los autores propusieron para X el valor de 25.5 minutos que corresponden a 1 ½ de las desviaciones estándar del tiempo entre los eventos de interfaz de usuario. Desde entonces muchos autores han adoptado el valor de 30 minutos. Sin embargo, pueden usarse técnicas de pre-procesamiento de datos más avanzadas para aprovechar totalmente toda la información disponible en los archivos log.

Las sesiones de navegación de usuario inferidas de los archivos log se modelan como un lenguaje probabilístico de hipertexto generado por una gramática probabilística de hipertexto (HPG) [25].

HPG es una gramática probabilística regular que tiene el mapa de cada uno del conjunto de los símbolos no terminales y los conjunto de los símbolos terminales. Cada símbolo no terminal corresponde a una página web y la regla de la producción corresponde a un link entre las páginas.

Además, S y F representan los estados de inicio y fin de las sesiones de navegación. El periodo de tiempo de secuencia de dos páginas esta dado por el periodo de tiempo que un usuario permanece en un link escogido. La probabilidad de una producción desde el inicio del estado es proporcional al periodo de tiempo en el que el estado fue visitado, lo que implica que el nodo destino de una producción con la probabilidad más alta corresponde a un estado que se visitó más a menudo.

Se define α como un parámetro que liga el peso deseado para la primera sesión de navegación de usuario. Si $\alpha = 0$ que solamente los estados que estuvieron de primeros en una sesión tiene una probabilidad mayor que cero de estar en una producción desde el primer estado inicial, si $\alpha = 1$ todos los estados visitados tienen un peso proporcional.

Esto quiere decir que cuándo $\alpha = 0$ el lenguaje de la gramática contiene sólo las cadenas cuya primeras cadenas comienzan en cada estado de la gramática. Por consiguiente, el valor de α debe ajustarse según la importancia dada al estado que esta siendo el primero en una sesión de navegación.

Finalmente las probabilidades de la producción desde el estado inicial corresponde al vector de probabilidades iniciales $\prod$ y la probabilidad de las otras transiciones corresponde a la matriz de transición de una cadena de Markov.

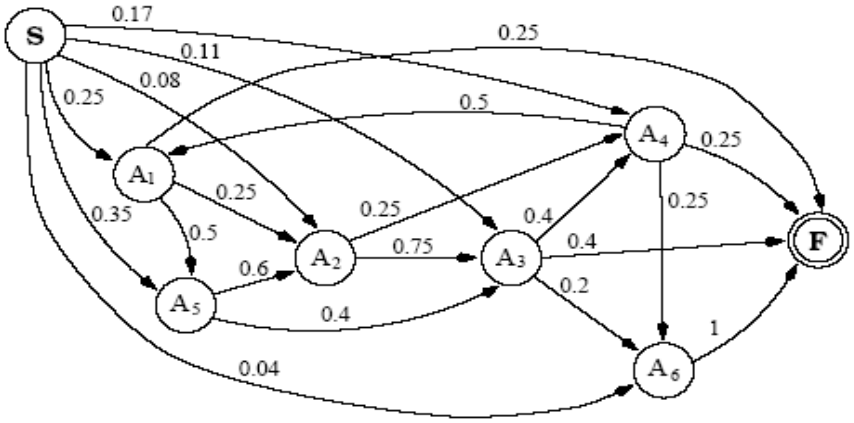
En la tabla 2.10. se tiene 6 sesiones de usuario con un total de 24 solicitudes de páginas, en donde el estado A1 se visitó 4 veces, 2 de la cuales tienen a A1 como el primer estado en la sesión del usuario, por consiguiente, para α =0.5 se tiene que:

∏(A1)=(0.5*4)/24+(0.5*2)/6=0.25.

Tabla 2.10. Sesiones de Usuario.

Session ID	User trail
1	$A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_4$
2	$A_1 \rightarrow A_5 \rightarrow A_3 \rightarrow A_4 \rightarrow A_1$
3	$A_5 \rightarrow A_2 \rightarrow A_4 \rightarrow A_6$
4	$A_5 \rightarrow A_2 \rightarrow A_3$
5	$A_5 \rightarrow A_2 \rightarrow A_3 \rightarrow A_6$
6	$A_4 \rightarrow A_1 \rightarrow A_5 \rightarrow A_3$

Figura 2.22. Gramática de hipertexto para N=1 y α =0.5.



En un HPG se evalúa la probabilidad del primer paso de la derivación de una cadena evaluada contra su ***soporte de apoyo φ***, y no se factoriza en la probabilidad de la derivación. Así, el umbral de apoyo se usa para restringir las cadenas que de otra manera podrían tener la probabilidad alta pero que corresponden a un subconjunto raramente visitado del sistema de hypertexto.

Es más, una cadena esta incluida en el lenguaje de la gramática si su probabilidad de derivación es anterior al punto de **corte λ**, dónde el punto de corte corresponde al umbral de confianza de la gramática. Los valores de soporte y confianza dan al usuario el control sobre la cantidad y la calidad de las rutas incluidas en el conjunto de reglas.

Las cadenas generadas por la gramática corresponden a las rutas de navegación del usuario, y el objetivo es identificar el subconjunto de esas cadenas que mejor caractericen el comportamiento del usuario cuando éste visita un sitio web. Parámetro como la confianza y el soporte se define como medidas que rankean las cadenas generadas.

El algoritmo utilizado para la minería que tienen la confianza y el soporte sobre los umbrales especificados es un caso especial de un grafo dirigido conocido como **"Primera búsqueda en profundidad"** que realiza una búsqueda exhaustiva de todas las cadenas con las características requeridas.

En la figura 2.23. se muestra las reglas que se obtuvieron con los diferentes modelos de configuración. De los primeros dos conjuntos de reglas se puede ver cómo el número de reglas y su media longitud disminuyen mientras el punto de corte aumenta. Se aprecia que aunque A3 nunca estuvo como primer estado en la sesión de navegación, ambos conjuntos de reglas incluyeron reglas que comenzaron con A3. Esto ocurre porque A3 tiene una alta frecuencia y el valor de parámetro α =0.5 dan un peso positivo.

El tercer conjunto de reglas tiene un alto soporte y como consecuencia las reglas que empiezan en el estado A3 están incluidas. Finalmente, el cuarto conjunto de reglas, corresponda a una gramática inferida ya que se dio el mismo peso todos, por ejemplo, α=1.0, y por consiguiente reglas que empiezan con A2 y A3 son incluidas aunque estos estados nunca estuvieron de primeros en una sesión de la navegación.

Figura 2.23. Reglas obtenidas con varios modelos de configuración

rule-set 1		rule-set 2		rule-set 3		rule-set 4	
$\alpha = 0.5$						$\alpha = 1.0$	
$\theta = 0.1$				$\theta = 0.15$		$\theta = 0.1$	
$\lambda = 0.2$		$\lambda = 0.3$		$\lambda = 0.3$		$\lambda = 0.3$	
rule	conf.	rule	conf.	rule	conf.	rule	conf.
A_1A_2	0.25	$A_1A_5A_2$	0.3	$A_1A_5A_2$	0.3	$A_1A_5A_2$	0.3
$A_1A_5A_3$	0.2	A_3A_4	0.4	A_4A_1	0.5	$A_2A_3A_4$	0.3
$A_1A_5A_2A_3$	0.23	A_4A_1	0.5	A_5A_3	0.4	A_3A_4	0.4
$A_3A_4A_1$	0.2	A_5A_3	0.4	$A_5A_2A_3$	0.45	A_4A_1	0.5
A_3A_6	0.2	$A_5A_2A_3$	0.45			A_5A_3	0.4
$A_4A_1A_5$	0.25					$A_5A_2A_3$	0.45
A_4A_6	0.25						
A_5A_3	0.4						
$A_5A_2A_3$	0.45						

The N gram model. Este modelo es utilizado para determinar la ruta supuesta de un usuario cuando navega en un sitio web dónde N, N≥1, es llamada ***profundidad de la historia***. Por consiguiente, cuando el usuario está visitando una página se asume que sólo en las N páginas previamente visitadas influye el vínculo que será escogido a continuación.

Esto dice que un usuario tiene una memoria limitada de las páginas previamente visitadas y que la próxima opción o la próxima página visitada sólo depende de las últimas N páginas que ha hojeado. En un N gran model, cada uno de los estados del HPG corresponde a una secuencia de N páginas visitadas.

El inconveniente del N gram model, es que el número de estados aumenta a medida que aumenta la profundidad de la historia de las páginas que ha visitado un usuario. Para un sitio web con n páginas y una N historia de profundidad el número de estados de la gramática esperados es: $n.b^{(N-1)}$, dónde b es el número medio de outlinks en una página.

Finalmente, el modelo de gramática de hipertexto es incremental en el sentido que cuando más datos logs se tengan disponibles se pueden incorporar en el modelo sin la necesidad de reconstruir la gramática desde el principio. El HPG es una manera compacta de guardar la historia de navegación donde el tamaño de la gramática sólo depende del número de páginas visitado y no de la cantidad de datos del archivo log disponible.

En la figura 2.24. se muestra el algoritmo utilizado para construir una HPG incrementalmente: Donde **T** es la colección de sesiones de navegación de usuario y **|T |** es la cardinalidad T_i es i^{th} sesión del conjunto y $T_i[j]$ es la j^{th} página de la sesión *i*. N es la profundidad de historia de la gramática, y StateSet es el conjunto de estados de la gramática; la función StateSet.update (), actualiza StateSet con el último estado inferido. Un Estado corresponde a N páginas solicitadas consecutivamente State[1],…,State [N]; la función **shiftleft()** realiza la asignación State[i]:=State[i+1], para i=1 hasta N-1. Además **Productions** es el conjunto de todas las producciones de la gramática y la función **Productions.update()** actualiza el conjunto con la última producción inferida.

Figura 2.24. Algoritmo para construir una gramática HPG

```
Algorithm 1 (Build_Grammar (T, N))
1. begin
2.   for i = 1 to |T|
3.     for k = 1 to N do
4.       State[k] = T_i[k];
5      end for
6.     StateSet.update(State);
7.     for j = N + 1 to |T_i|
8.       LeftState = State;
9.       State.shiftLeft();
10.      State[N] = T_i[j];
11.      StateSet.update(State);
12.      Productions.update(LeftState,State);
13.    end for
14.  end for
15.end.
```

La Entropía de un HPG. La entropía en un HPG es un estimador de las propiedades estadísticas del lenguaje generado por la gramática. La entropía es una medida de la incertidumbre del resultado de una variable al azar y en este contexto el espacio de la muestra es el conjunto de todas las cadenas generadas por la gramática.

Si no se tuviera información en absoluto sobre la interacción del usuario con la web lo más racional sería asumir que todas las páginas tendrían la misma probabilidad de ser visitadas y que todos sus links tendrían la misma probabilidad de ser escogidos. La entropía es máxima en este caso y no hay ningún punto en busca de los modelos que intente predecir una conducta al azar.

Si la entropía es cercana a cero el comportamiento o conducta del usuario tendrá pocas incertidumbres, y un pequeño conjunto de short-rules debe contener la suficiente información para caracterizar su conducta. En conclusión, si la entropía de una gramática probabilística está cerca de cero debe haber un pequeño conjunto de cadenas con alta probabilidad y si la entropía es alta entonces debe haber un gran número de cadenas con similar o baja probabilidad.

Asumiendo una transición con probabilidad uno desde el estado F hasta S, una HPG corresponde a una irreducible y no periódica cadena de Markov con un vector de distribución estacionaria μ y matriz de transición A.

Se puede estimar la entropía con la siguiente expresión:

$H = H(\pi) + (-\sum_{ij} \pi_i A_{ij} log\ A_{ij})$, donde $H(\pi)$ se incluye para tener en cuenta la aleatoriedad de la escogencia de la página inicial. Nótese que se utiliza el vector de probabilidad inicial $\prod$ como un estimador del vector estacionario μ que es proporcional al periodo de tiempo que cada estado fue visitado. El valor de H puede normalizarse para estar en el rango entre 0 y 1 que es la proporción al azar de la gramática correspondiente. La gramática del azar es una gramática con la misma estructura pero en la cual todos los estados tienen la probabilidad de sus out-links conforme a una distribución uniforme.

Iterative Deepening Heuristic. La idea detrás de esta heurística es llevar a cabo una búsqueda reiterativa de profundidad (un ID) donde el conjunto de reglas se construyen de forma incremental hasta que tenga las características deseadas.

Para tal efecto un ***parámetro de parada*** determina la medida de distancia entre el conjunto de reglas hasta el momento construido y el conjunto de reglas máximo. El analista es el que especifica el valor del ***criterio de parada*** o umbral. La búsqueda de las reglas termina cual el valor del parámetro de parada supera el valor del criterio de parada; es decir, cuando sobrepasa dicho umbral. Al colocar un valor a dicho umbral, el analista está dando el control sobre el número y la longitud de las reglas obtenidas.

El ID trabaja de la siguiente manera: Un árbol de exploración es construido desde el estado de salida y en la primera iteración únicamente se exploran los caminos con longitud uno. Teniendo el árbol construido con profundidad uno se evalúa el parámetro de parada y el criterio de parada que específico el analista. Si el parámetro supera el umbral la exploración se detiene, de otra manera la profundidad del árbol se incrementa en uno y se exploran todos los caminos con longitud dos.

De esta manera se incrementa el parámetro de parada y es nuevamente evaluado para ver si ha superado el criterio de parada o umbral, y de esta manera la búsqueda continua hasta que se los dos valores coincidan; es decir, hasta que el criterio de parada sea encontrado en la búsqueda.

Cuando un camino es explorado cada out_link desde el último estado es evaluado para determinar si esta es una ***expansión admisible*** o una ***expansión inadmisible.***

Una expansión del camino es admisible si llevará a un camino con la probabilidad sobre el punto de corte e inadmisible en caso contrario. Además un camino es máximo cuando no tiene una ***expansión admisible*** y un camino es candidato tiene por lo menos una ***expansión admisible.***

Sea $\mathcal{L}^{\lambda}$ las posiciones para la gramática, el punto de corte λ y cardinalidad $|\mathcal{L}^{\lambda}$. El lenguaje $\mathcal{L}^{\lambda}$ es obtenido mediante el algoritmo **BFS** e incluye únicamente al camino o camino máximo, los subcaminos son implícitamente definidos a partir del prefijo del camino máximo.

Sea $\boldsymbol{k}$ la profundidad actual del árbol de exploración medida por el número dlinks y $\mathcal{L}^{\lambda}$ el conjunto de caminos en el árbol de exploración donde cada camino tiene una longitud menor o igual a $\boldsymbol{k}$ y la probabilidad esta sobre el punto de corte.
EL $\mathcal{L}^{\lambda}$ incluye tanto al camino máximo como a los caminos candidatos del árbol de exploración. Además $\mathcal{T}^{\lambda}$ es una variable randómica que denota la longitud del camino y P($\mathcal{T}^{\lambda} \leq \mathbf{k}$) es la probabilidad del camino, y $\mathcal{L}^{\lambda}$no debe ser mas grande que $\boldsymbol{k}$.

Dado $\mathcal{L}_k^{\lambda}$, P($\mathcal{T}^{\lambda} \leq \mathbf{k}$) se calcula con la observación realizada del primer paso que se realiza para identificar a la expansión admisible e inadmisible y por consiguiente al máximo camino y a los caminos candidatos en arrastra en $\mathcal{L}_k^{\lambda}$. La probabilidad P($\mathcal{T}^{\lambda} \leq \mathbf{k}$) corresponde a la sumatoria de las probabilidades de los caminos máximos y a la expansión admisible de los caminos candidatos.
P($\boldsymbol{\mathcal{L}}$) es la sumatoria de las probabilidades de de todos los caminos en $\boldsymbol{\mathcal{L}}$ donde $\boldsymbol{\mathcal{L}} \in$ { $\mathcal{L}^{\lambda}, \mathcal{L}_k^{\lambda}$ }.

Note que 0 ≤ P($\mathcal{L}_k^{\lambda}$.) ≤ 1 que es la sumatoria de las probabilidades del conjunto de caminos, sin incluir a los sub-caminos, en el árbol de exploración, que corresponde a un árbol estocástico, es siempre menor a igual a uno. Además, P($\mathcal{L}_k^{\lambda}$.) ≥ P($\mathcal{T}^{\lambda} \leq \mathbf{k}$) donde ambos incluyen los caminos máximos con longitud menor a igual a $\boldsymbol{k}$, $\mathcal{L}_k^{\lambda}$ incluye los caminos candidatos y que además tienes una probabilidad mayor, de sus expansiones admisibles incluidas en $\mathcal{T}^{\lambda} \leq \mathbf{k}$. Además, P($\mathcal{L}_k^{\lambda}$.) ≥ P($\mathcal{L}^{\lambda}$), donde cada camino en $\mathcal{L}_k^{\lambda}$ es un prefijo apropiado del camino máximo en $\mathcal{L}^{\lambda}$. Finalmente, para la longitud suficiente para $\boldsymbol{k}$ es $\mathcal{L}_k^{\lambda} = \mathcal{L}^{\lambda}$.

El ID heurístico se calcula iterativamente con $\mathcal{L}^{\lambda}$ *for* k { 1,2, …} hasta $\mathcal{L}_k^{\lambda}$ que es una muy buena aproximación de $\mathcal{L}^{\lambda}$. Para una $\boldsymbol{k}$ dada se debe calcular cuan cercana esta $\mathcal{L}_k^{\lambda}$ de $\mathcal{L}^{\lambda}$ en base al valor del parámetro de parada que es definido como la

probabilidad restante de la exploración. El parámetro de parada es calculado con el primer paso y se define como:

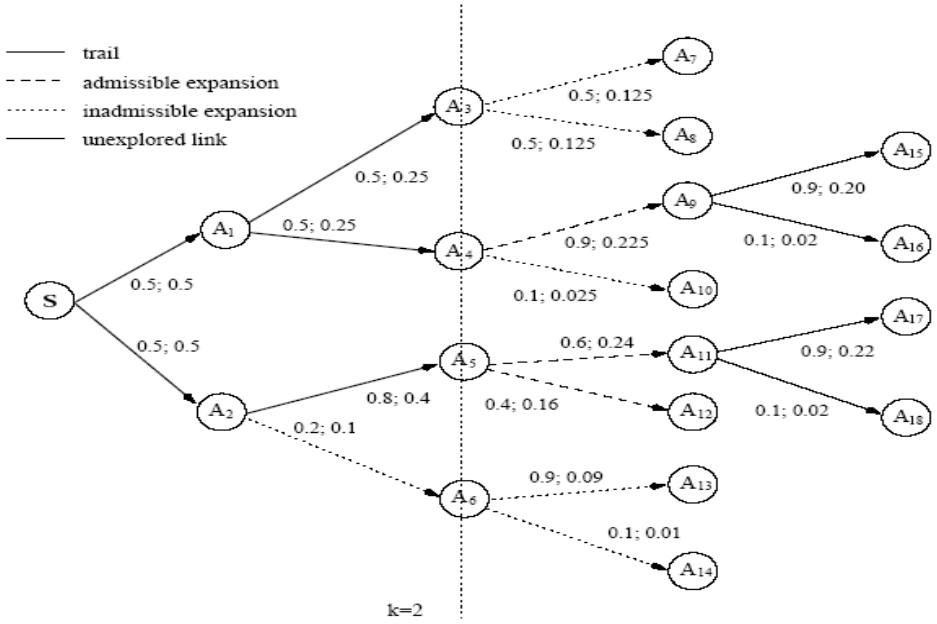
$$\eta k = P(\mathcal{L}_k^{\lambda}.) - P(\mathcal{T}^{\lambda} \leq \mathbf{k})$$

Nótese que el valor de parada está definido con la sumatoria de las probabilidades de las expansiones admisibles de los caminos candidatos. La exploración se detiene cuando el parámetro de parada es lo suficientemente pequeño y en este caso será menor que τ, donde $\tau > 0$ y se denomina criterio de parada. Al final de la exploración el conjunto de reglas estará dado por $\mathcal{L}_k^{\lambda}$ donde el camino máximo y los caminos candidatos son considerados como reglas cuando el criterio de parada es conocido.

La figura 2.25. muestra un ejemplo de un árbol de exploración para clarificar un poco los anteriores conceptos. Se puede observar que el par de números cercanos al link seguidos por la probabilidad del camino que inicia en el estado S. Para k= 2 y λ=0.15 el conjunto de caminos candidatos es {A1A3,A1A4,A2A5}.

Como primer paso, a partir del último estado del camino candidato A1A3 se verifica que los caminos A3A7, A3A8 son expansiones inadmisibles, por consiguiente A1A3 es un camino máximo. Además, también se puede identificar que A4A10 es una expansión inadmisible de A1A4 , A4A9 como una expansión admisible del mismo camino así que A1A4 es un camino candidato. Finalmente A5A11 y A5A12 son identificados como expansiones admisibles de A2A5 lo que significa que A2A5 es también un camino candidato.

Figura 2.25. Árbol de exploración.



Una vez se clasifican todos los caminos con longitud menor o igual a k= 2 se puede calcular el valor del parámetro de parada. Para tal efecto P($\mathcal{L}_2^{\lambda}$), representa la probabilidad del conjunto de caminos máximo o candidatos cuya longitud es menor o igual a 2, y que se evalúa de la siguiente manera:

$$P(\mathcal{L}_2^{\lambda}.) = P(A1A3)+ P(A1A4) + P(A2A5) = 0.25+0.25+0.4= 0.9$$

Adicionalmente, P($\mathcal{T}^{\lambda} \leq 2$), es la probabilidad de la longitud de la longitud del camino máximo que debe ser menor o igual a 2, y está dada por la sumatoria delas probabilidades de los caminos máximos y de las expansiones inadmisibles de los caminos candidatos.

$$P(\mathcal{T}^{\lambda} \leq 2) = P(A1A3)+ P(A1A4A10)= 0.25+0.025=0.275$$

Por consiguiente el valor del parámetro de parada, es:

$$\eta 2 = P(\mathcal{L}_k^{\lambda}.) - P(\mathcal{T}^{\lambda} \leq k)= 0.9 - 0.275 = 0.625.$$

El anterior valor corresponde a la sumatoria de las probabilidades de todas las expansiones admisibles de los caminos candidatos. Finalmente, el lenguaje con punto de corte λ=0.15 es:

$\mathcal{L}^{\lambda}$= {A1A3, A1A4A9A15, A2A5A11A17, A2A5A12}

Además, P($\mathcal{L}^{\lambda}$) = 0.25 + 0.2 + 0.22 + 0.16 = 0.83; por consiguiente se puede verificar que:

(P($\mathcal{L}_2^{\lambda}$.) = 0.9) > (P($\mathcal{L}^{\lambda}$) = 0.83) > (P($T^{\lambda} \leq 2$)=0.275)

Fine – Grained Iterative Deepening Heuristic FG. El FG explora los caminos selectivamente, en contraste con el ID donde todas las expansiones admisibles de los caminos candidatos son explorados al mismo tiempo.

Para una formalización del FG es necesario introducir un nuevo concepto *los* ***subcaminos –candidatos,*** que no son otra cosa que los subcaminos de los caminos más largos que tiene por lo menos una expansión admisible sin explorar.

Para clarificar este concepto en la figura 2.25, se puede apreciar que desde el estado A1 se identifican A1A3 y A1A4 como expansiones admisibles.

Si A1A3 es escogido para ser expandido primero deberá convertirse en un camino candidato y el camino A1 deberá ser un subcamino candidato. Nótese que el camino A1 tubo una expansión admisible sin explorar A1A4, y esta es un subcamino del camino candidato A1A3. La fundamental diferencia entre un subcamino candidato y un camino candidato es que el subcamino es siempre un prefijo de otro camino máximo o camino candidato.

Cuando la exploración del árbol se detiene, porque el criterio de parada es encontrado, el conjunto de reglas retornadas se compone de los caminos máximos y de los caminos candidatos, los subcaminos candidatos están descartados desde que haya prefijos de uno o mas de los otros caminos.

En FG se puede redefinir el valor de *k* para que sea el número total de links que constituyen el árbol de exploración y donde $\mathcal{L}_k^{\lambda}$es el conjunto de caminos en el árbol; $\mathcal{L}_k^{\lambda}$ incluye al camino máximo y al camino candidato en el árbol de exploración. Además $\mathcal{L}_k^{\lambda}$ será la familia de todos los conjuntos de l_k^{λ}, donde $\mathcal{L}_k^{\lambda}$= $\{l_k^{\lambda} | \# l_k^{\lambda} = \lambda\}$, y $\# l_k^{\lambda}$ representa el número de links diferentes que componen los caminos en l_k^{λ}. Además, (l_k^{λ}) es la sumatoria de las probabilidades para todos los caminos en l_k^{λ}.

Además t^{λ} es una variable que denota la suma de las longitudes del conjunto de caminos. Además l_k^{λ}, P($t^{\lambda} \leq k$), representa la probabilidad del camino en l_k^{λ} siendo el camino máximo. P($t^{\lambda} \leq k$), es calculado en el primer paso desde el camino candidato con el fin de identificar los caminos máximo y candidato. Nótese que P($t^{\lambda} \leq k$), corresponde a la sumatoria de las probabilidades de los caminos máximo y de las expansiones inadmisibles de los caminos candidatos. Similar que con el ID, $0 \leq (l_k^{\lambda}) \leq 1$, $P(l_k^{\lambda}) \geq$ P($t^{\lambda} \leq k$), y $P(l_k^{\lambda}) \geq$ P($\mathcal{L}^{\lambda}$).

Además l_a^{λ} es el conjunto de subcaminos candidatos en l_k^{λ} y P(l_a^{λ}) es la sumatoria de las expansiones inadmisibles que no han sido exploradas de los subcaminos candidatos, entonces para una k dada y l_k^{λ} el parámetro de parada se define como:

$$\eta k = P(l_k^{\lambda}) - P(t^{\lambda} \leq k) + P(l_a^{\lambda})$$

La búsqueda de las reglas termina cuando el parámetro de parada toma un valor suficientemente cercano al criterio de parada τ. El conjunto de reglas inducidas, es el conjunto de caminos explorados, l_k^{λ}que contiene al camino máximo y al camino candidato. Note que en cada fase el valor del parámetro de parada corresponde a la cantidad de probabilidad que resulta de explorar y que está dada por la sumatoria de las probabilidades de los caminos candidatos y los subcaminos candidatos.

Se puede decidir cual es el camino que se expande con el ***criterio del mejor camino***, se expande el camino de acuerdo a la mas alta probabilidad del camino de todas las expansiones admisibles de los caminos en (l_k^{λ}).

Tomando el anterior ejemplo de la figura 2.17, pero para FG con k=5 y λ=0.15. Aplicando el criterio del mejor camino, en este caso, se tiene que:

$$l_5^{\lambda} = \{A1A3, A1A4\ A2A5\}\ Y\ P(l_5^{\lambda}) = P(A1A3) + P(A1A4) + P(A2A5) = 0.9$$

Además que:
P($t^{\lambda} \leq 5$)=P(A1A3)+P(A1A4A10)=0.275 y $\eta 5 = P(l_5^{\lambda}) - P(t^{\lambda} \leq 5) + P(l_a^{\lambda})$=0.625

En este caso en particular no hay subcaminos candidatos, de acuerdo al criterio del mejor camino el próximo camino a ser expandido es A2A5 a A2A5A11, que es uno de los que tiene la más alta probabilidad a través de los caminos correspondientes a la expansión admisible.

Adicionalmente se presenta un algoritmo de minería de datos para inferir las reglas de la gramática para un punto de corte mediante la utilización de un método de exploración gráfica como el Depth First Search DFS [Tarjan, 1972] que es conceptualmente mas simple que el Breadth First Search RTF, adicionalmente el DFS consume menos memoria debido a su naturaleza recursiva.

Un DFS o recorrido en profundidad es un algoritmo que permite recorrer todos los nodos de un grafo o árbol de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto.

Cuando ya no quedan más nodos que visitar en dicho camino, regresa (backtracking), de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.

El DFS surge como una modificación para la construcción de un árbol con el estado inicial S. Cada rama del árbol es explorada hasta que la probabilidad no sobrepasa el punto de corte.

Una rama que tiene una probabilidad anterior al punto de corte se denomina Ruta Candidato, si esta todavía tiene alguna expansión será evaluada llamándola ***regla o máxima ruta.*** La probabilidad dada al lado de un enlace corresponde a la probabilidad de l estado de inicio S.

Algoritmo DFS

```
1.  Begin
2.  for i = 0 to n
3.          if (p(S→Ai) > α then
4.                  CT.push (aiAi, p(S→Ai));
5.          end if
6.  end for
7.  while (CT.empty() == false) do
8.          for each wAi ∈ CT
9.                  flag=false;
10.                 for j=1 to Pi
11.                         If(p(WAi).(Pij) >αλ) then
```

```
12.                         CT1.push(WajAj, p(Wai).P(I,j));
13.                         flag=true
14.                     end if
15.                 end for
16.                 if (flag==false) then
17.                     RS.push (w,p(w))
18.                 end if
19.             end for
20.             CT =  CT1;
21.             Ct1 = null;
22.             end while
23.  End.
```

Para el anterior algoritmo se defina α como el punto de corte y n como el número de estados. Además CT y CT1 son dos diferentes conjuntos de rutas candidatas y RS son las reglas. El método CT.empty() verifica si un conjunto de rutas candidatas es vacío y el método x.push() inserta una nueva ruta dentro de la colección de rutas donde x ∈ {CT, CT1, RS}. Adicionalmente una ruta es representada por WAi, donde W representa la colección de símbolos terminales y Ai es el único símbolo no terminal; Pij ≤ j ≤ Pij representa el conjunto de producciones cuyo lado izquierdo es Ai y Pi es cardinalmente.

Considérese un HPG con n estados, l enlaces un α punto de corte. El valor de las ramas de la gramática se define como BF=l/n y la extensión de una ruta está dada por el número de símbolos terminales contenidos. Así es que ignorando el punto de corte, el número de rutas gramaticales que tienen extensión m, # Wm esta dado por:

$$\mathbf{E\ (\ \#Wm\ ) = n.BF(1/m\text{-}1) = n(1/n)(1/m\text{-}1)}$$

Además cuando α > 0, es decir cada estado tiene una probabilidad inicial positiva, la probabilidad promedio de una producción a partir del estado inicial es 1/n y la probabilidad promedio de un estado correspondiente a una página Ai ∈ V – {S,F} es i/BF = n/l.

Por tanto la probabilidad promedio de una ruta con extensión m,p(Wm) es:

$$\mathbf{E\ p(Wm\ ) = 1/n(n/l)(1/m\text{-}1)}$$

Dado una máxima confianza y un factor de ramas BF = l/n el promedio esperado de la extensión de la regla es constante e independiente del número de estados.

Además el máximo soporte es considerado para ser proporcionar a los números de estados.

En Polaris se realizó la implementación completa del algoritmo HPG obteniendo como resultado la visualización de la gramática y la construcción del árbol de exploración.

3. ANTECEDENTES DE HERRAMIENTAS DE MINERÍA WEB

3. DOWNLOAD ANALYZER 17

El Danalyzer es un software para el análisis de las descargas de archivos de texto, audio, video y gráficos.

Crea reportes estadísticos sobre descargas, refers, frases y palabras de búsqueda, generando informes sobre la cantidad de descargas completas de cada archivo, la cantidad de visitantes únicos en un sitio que ha empezado la descarga de un archivo y el volumen de tráfico en el mismo; así como también información sobre los sitios o las páginas desde las cuales un visitante realiza la descarga. [20].

Requerimientos

- Requerimientos de Hardware
  Mínimo de memoria RAM libre de 32MB.
  Procesador Pentium™-100 MHz o superior
  4 MB de espacio libre en disco.

- Requerimientos de Software
  Danalyzer está disponible para Windows™ 95/98/ME/NT/2000/XP.

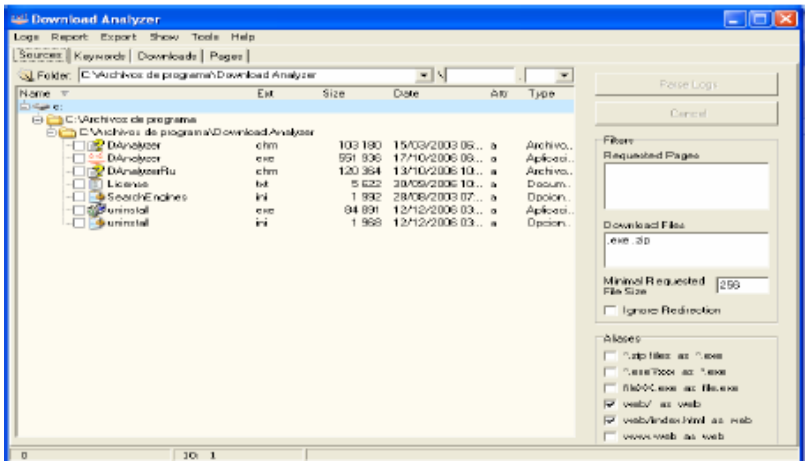
Instalación

Para descargar la copia de evaluación de la versión Download Analyzer debe ir a la página: http://www.DownloadAnalyzer.com/.

El proceso de instalación es sencillo, basta con hacer clic en el botón de Next o siguiente. Al final de este proceso aparece la ventana de instalación exitosa, la cual indica que toda la configuración de instalación se ha realizado satisfactoriamente y el Danalyzer está listo para ser utilizado.

Manejo de la Herramienta

El trabajo con la herramienta es sencillo, sin embargo el usuario puede ir a la opción Help – Ayuda – del menú principal, donde se desplegará el manual de uso de Danalyzer, con toda la información que se requiere para el adecuado uso del producto:



El proceso se inicia indicando indicar la ubicación del archivo Log que se va a analizar. El Danalyzer 17 acepta archivos de registro común –CLF- y archivos de registro Combinado –ELF-.

Permite el análisis de los siguientes parámetros:

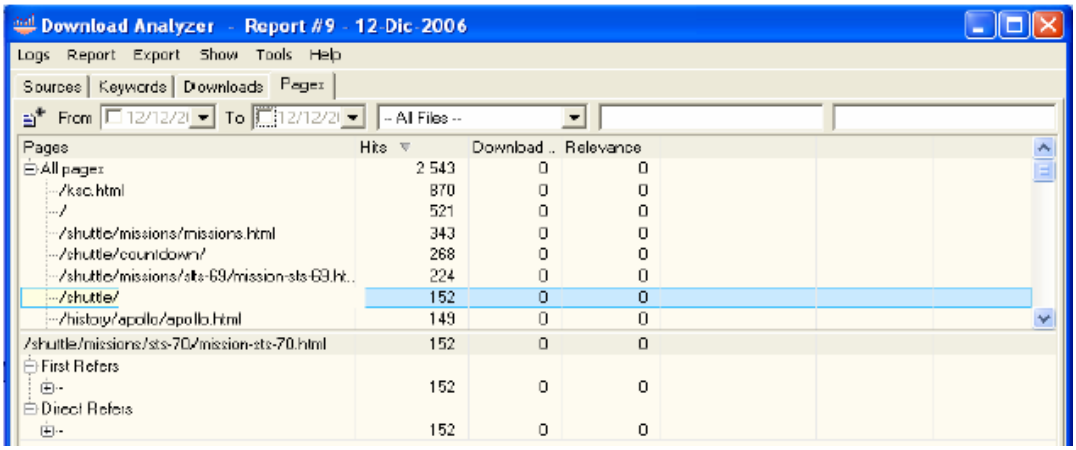
- Hits: El número de visitantes que llegan como resultado de una frase o palabra utilizada en una búsqueda.
- Download hits: El número de visitantes quienes, después de realizar una búsqueda con una determinada palabras, visitan el sitio y descargan por lo menos uno de los archivos.
- Relevancia: La proporción de descargas, medida como un porcentaje. Este valor define hasta que punto la frase de búsqueda género trafico en un buscador.
- Google Position: La posición del sitio como resultado de una búsqueda con una palabra clave o
- frase de búsqueda a través de Google.
- Además, para facilitar el análisis en el reporte de acuerdo a las palabras claves o frases de

- Rating de tráfico: Para indicar el tráfico generado por una palabra clave utilizada en un buscador.

En Downloads se puede hacer el análisis cuantitativo de los archivos descargados, definido por tres maneras, reflejadas en las tres columnas del informe al lado de cada archivo:

- Complete Downloads: Muestra la cantidad real de descargas completas de cada archivo. La descarga es considerada completa si el tamaño total del archivo que se descarga a una única dirección IP (durante un periodo de 24 horas) es igual o mayor al tamaño del archivo transmitido.

- Download Hits: Muestra el número de visitantes únicos en un sitio que han empezado una descarga. Por ejemplo, el número de direcciones IP únicas desde donde se ha iniciado la descarga (donde el visitante oprimió "iniciar descarga de archivo), pero sin garantizar que el archivo fue descargado completamente.



- Pages, se puede apreciar el reporte detallado por páginas. Si se ubica el cursor sobre alguna de las páginas consolidadas, entonces en el informe sobre descargas para una página seleccionada, se despliega el número de descargas de cada archivo resultante de una visita a la página.

Ventajas

- Es una herramienta libre.
- Requerimientos mínimos de software y de hardware.

- Manejo de Filtros para controlar la información que va a ser analizada en el programa sobre las páginas, buscadores, los archivos pedidos, y las subcadenas incluidas en una frase de búsqueda.
- Capacidad para analizar archivos logs bastante grandes –que exceden 1 gigabyte-, analizar un archivo o todos los archivos que hayan sido descargados conjuntamente.

Desventajas

- La mayor limitación del Download Analyzer 17 es que no realiza procesos de minería de Datos Data Mining- por lo tanto no implementa ninguna de sus técnicas.
- La herramienta no genera reportes gráficos. Los reportes generados son únicamente textuales.

3.2 WEB MINING ANALIZER V1.1

El Web Mining Analaizer v1.1 es una herramienta para el análisis de ficheros log, que proporciona una información muy útil para la optimización de los contenidos de una página web y datos orientativos sobre el uso que se realiza de la misma. [44].

El Web Mining Analaizer v1.1 se caracteriza por realizar un proceso de Minería de Datos, mediante la aplicación técnicas como el Clustering, Aplicación del C4.5, y Asociación.

Requerimientos

- Requerimientos de Hardware
  Mínimo de memoria RAM libre de 32MB.
  Procesador Pentium™-100 MHz o superior
  767.52 KB de espacio libre en disco.

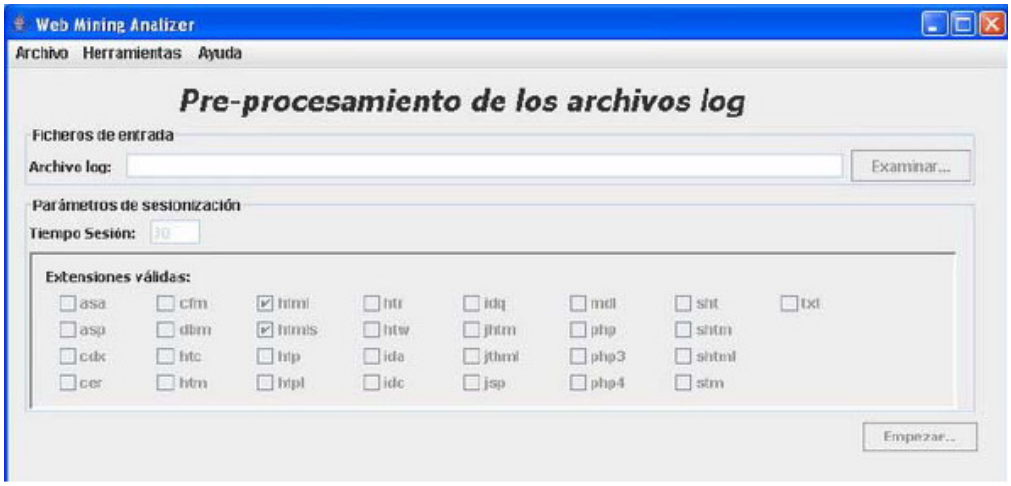
- Requerimientos de Software
  Para la ejecución de web Mining Analaizer v1.1. es necesario tener instalada la máquina virtual de java™, versión mínima JDK1.5.0

Instalación

Para descargar la copia de evaluación de web Mining Analaizer v.1.1 debe ir a la página: http://www-etsi2.ugr.es/usuarios/anabelo.

El proceso de instalación es sencillo, basta con hacer clic en el botón de Siguiente. Al final de este proceso aparece la ventana informando que la instalación del programa fue exitosa; toda la configuración de instalación se ha realizado satisfactoriamente y el Web Mining Analaizer v1.1 está listo para ser utilizado.

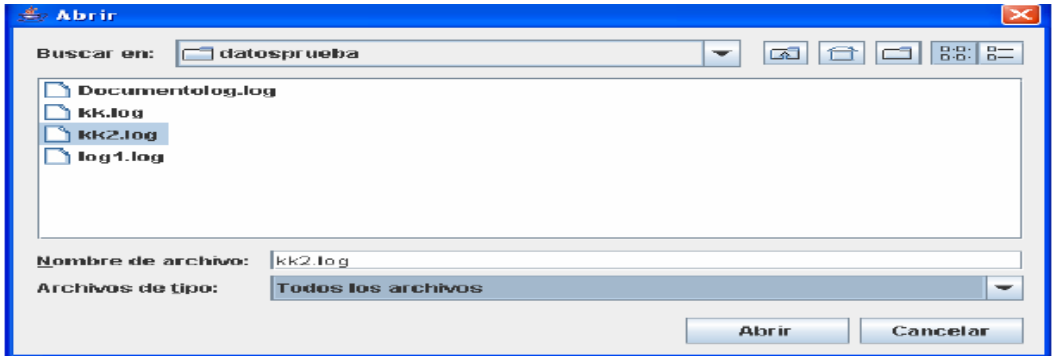
Manejo de la Herramienta
El entorno principal del Web Analaizer es el siguiente:



La aplicación consta en su entorno inicial de tres partes bien diferenciadas:

• Menú principal: Permite la administración del proyecto. Contiene los submenús de Archivo, Herramientas y Ayuda.

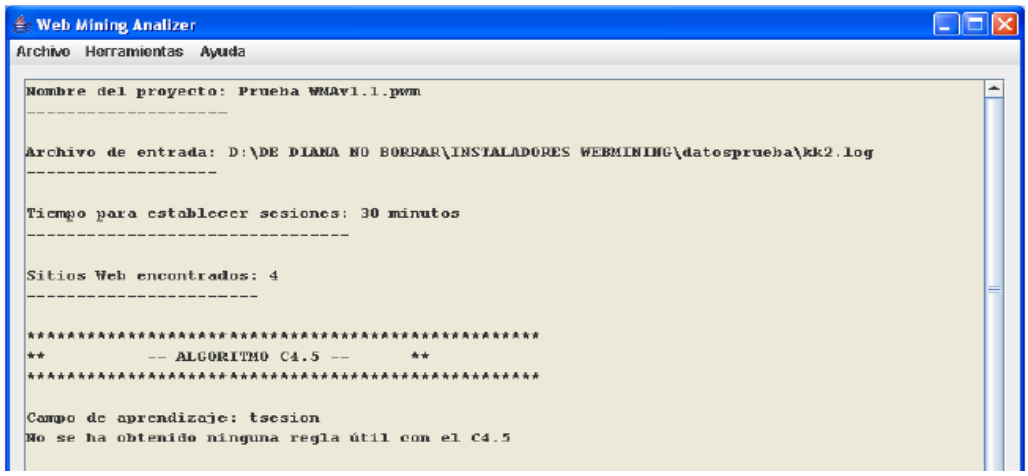
El proceso se inicia creando un nuevo proyecto y se selecciona el archivo log – CLF o ELF- sobre el que se desee realizar el análisis:



Posteriormente se despliega una pantalla donde se muestra información del fichero log analizado y se permite al usuario seleccionar la técnica que desea emplear para realizar su análisis, para lo cual es necesario que ingrese los parámetros correspondientes para cada una de ellas:

- En caso de pulsar C4.5, se puede seleccionar de entre la lista suministrada, la clase sobre la cual se va a realizar el análisis.
- En caso de elegir reglas de asociación se debe establecer el mínimo soporte y la confianza.
- Y finalmente puede decidir la aplicación del clustering.

El archivo del reporte suministrado, presenta el siguiente formato:



Ventajas

- El Web Mining Analaizer v1.1 es una herramienta especializada para el análisis del tráfico Web, que se caracteriza principalmente por realizar un proceso de Minería de Datos, mediante la aplicación conjunta o separada de técnicas como el Clustering, Aplicación del C4.5, y las reglas de asociación; es por eso que el descubrimiento de nuevo conocimiento está asegurado.

- El Web Mining Analaizer v1.1 presenta grandes ventajas de rendimiento, empleando un tiempo significativamente pequeño para el preprocesamiento de datos y entrega de resultados.
- La herramienta presenta un buen sistema de filtros que le da al usuario la posibilidad de escoger el tipo de información que va a incluir en el análisis.
- Adicionalmente, puede generar reportes estadísticos, de fácil entendimiento para cualquier usuario que no tenga muchos conocimientos en Minería de Datos.
- Genera un documento HTML que contiene la información que se ha obtenido de la aplicación, facilitando la portabilidad de la misma.

Desventajas

- Para la ejecución de web Mining Analaizer v1.1. es necesario tener instalada la máquina virtual de java™, versión mínima JDK1.5.0
- El Web Mining Analaizer v1.1 no genera reportes gráficos. Los reportes generados son únicamente textuales.

3.3. WUSAGE 8.0

El Wusage es un sistema de estadística que ayuda a determinar el verdadero impacto del servidor web, midiendo la popularidad de los documentos, así como identificando los páginas que son más visitadas, es ideal para saber con precisión cuales son los portales y sitios web que generan mayor tráfico para tu sitio web. El Wusage es una herramienta de marketing, que determinando la trayectoria que los usuarios siguen y analizando los sitios de los cuáles accesan, ayuda a descubrir cuáles son los sitios externos más importantes para un usuario. [19].

Requerimientos

- Requerimientos de Hardware:

  Mínimo de memoria RAM libre de 32MB, dependiendo del Web Server se necesitara más. También, se puede utilizar memoria virtual.

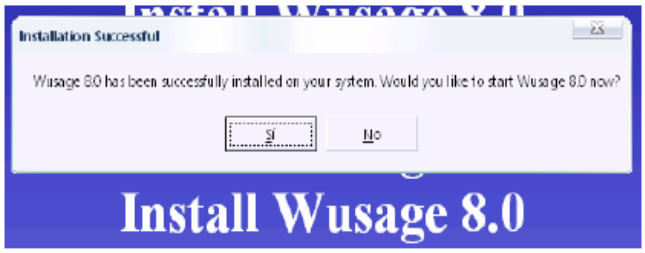
- Requerimientos de Software:

  Wusage esta disponible para la mayoría de los sistemas operativos (SO) y computadoras: Unix® , Windows™ 95, 98 o NT, o MacOS™.

Instalación

Para descargar la copia de evaluación de la versión WUSAGE 8.0 debe ir a la página: http://www.boutell.com/wusage/download.html

Donde usted leerá (en ingles) las opciones de descarga incluyendo en que sistema operativo lo necesita. Una vez usted tiene descargado el instalador, se inicia la instalación con la ayuda de un sencillo asistente. Con la ventana de 'Installation Successful' se informa que la instalación ha sido exitosa.



Manejo de la Herramienta

Una vez inicia la herramienta por primera vez aparece la siguiente ventana y advertencia: La ventana 'Wusage 8.0: Conectando con el servicio Wusage' dice que la interfaz de usuario del WUSAGE aparecerá momentáneamente en el navegador Web predeterminado, es decir al hacer clic en 'Aceptar' el navegador Web estará conectado al servicio Wusage, el cual estará corriendo en el computador. El sitio Web que se mirará en la barra de localización es el localhost (para referirse al propio computador) No es posible comenzar contactando otro computador. Esta advertencia solo aparece una vez.

El proceso se inicia indicando la ubicación del archivo de log Web Server o el directorio que contiene uno o más archivos de log Web Server, manejando los formatos de archivo log:

- El 'common' Log file format,
- El 'combined' log format (una popular extensión del common log format)
- El EMWAC log format (Windows™ NT)
- El Microsoft IIS log format
- El Extended log format definido por la W3C e implementado por Microsoft en el IIS™ 4.0
- El WebStar log format usado por Macintosh.
- Reconoce las variaciones del 'common' Log format incluida la última variación hecha por el Oracle Web Server. Y también reconoce si se le ha añadido más campos a este log.

El programa puede acceder a los archivos locales dentro del disco duro o una LAN de la empresa y también accede a los archivos remotos mediante protocolos como FTP y HTTP.

Por tanto aparecen 4 opciones:

- Archivos Locales, se debe escoger el drive y directorio donde se encuentra el o los archivos log finalmente al presionar 'OK' se remite a la página principal.
- Archivos Remotos por FTP, se debe ingresar los datos como: sitio FTP, usuario FTP, contraseña FTP, Directorio Inicial FTP (opcional) una vez listos presionar 'Abrir FTP' y se devuelve a la página principal.
- Archivos Remotos por HTTP, debe llenar los datos como: sitio HTTP, usuario HTTP, contraseña, HTTP, Directorio Inicial HTTP (opcional) una vez listos presionar 'Abrir HTTP' sino 'Cancel' y se devuelve a la página principal.
- Cancelar, se devuelve a la página principal sin ningún cambio en la fuente de datos del programa.

En la página de control, es donde se realiza todo el análisis de los archivos, y presenta las siguientes opciones:

• Actualizar estadísticas ahora: Muestra datos estadísticos tales como páginas vistas.
• Regenerar completamente: Refresca todo y vuelve a generar las estadísticas
• Consultas rápidas: Datos específicos que se quiera saber sobre el sitio
• Editar configuración: Permite configurar datos como cuantos días tiene un mes, que reportes obtener, el directorio donde se encuentra el log
• Editar cuentas de acceso remoto: Configuración de usuario remoto nombre y contraseña
• Actualizar resumen.
• Cambiar configuración global: Datos necesarios del computador y la red.
• Si se quiere generar reportes con un intervalo de tiempo determinado,

Ventajas

- Fácil de instalar
- Copia de evaluación gratuita
- Usa como interfaz un navegador web que hace que sea posible usarla en cualquier tipo de SO.
- Genera atractivos informes en HTML, de forma relativamente más clara y entendible, con la ayuda de barras y gráficas en 3D. Los informes se pueden

enviar por E-mail. El programa usa el navegador para conseguir un atractivo entorno, que sea lo más consistente posible a través de muchos sistemas operativos.

Desventajas

- Para cargar un log es necesario seguir muchos pasos divididos en páginas que hace tedioso el proceso.
- El Wusage 8.0 no realiza procesos de minería de datos data mining- por lo tanto no implementa ninguna de sus técnicas.
- Le falta mejorar el diseño de los reportes y de las páginas.

3.4. ANALOG 6.0

ANALOG es un programa que analiza los archivos log de un servidor web. Funciona en casi cualquier sistema operativo. Su diseño es sencillo, rápido, y produce sencillos informes, aunque combinado con el report magic, se puede mejorar mucho más los informes.

Esta herramienta puede generar una gran gamma de reportes acerca de estas solicitudes, que pueden variar desde los dominios que más visitan un sitio, las horas de mayor acceso, reportes anuales, entre otros más. [17].

Requerimientos

ANALOG esta disponible para la mayoría de los sistemas operativos (SO) (Linux™, Windows™) y computadoras, dependiendo de la máquina mejorará su rendimiento.

Instalación

Instalación en Linux: El archivo .Tar es posible descargarlo de la página: http://www.analog.cx, y contiene el código fuente de apache, este debe ser descomprimido en un directorio temporal (/tmp por lo general), para poder iniciar la instalación. Seguidamente dentro del directorio temporal (/tmp), y en el subdirectorio src se debe ejecutar el comando:

```
[root@OSMOSIS src]# pwd
```

```
/tmp/analog-5.1/src
[root@OSMOSIS src]# make
gcc -O2 -DUNIX -c alias.c
gcc -O2 -DUNIX -c analog.c
gcc -O2 -DUNIX -c cache.c
....
....
```

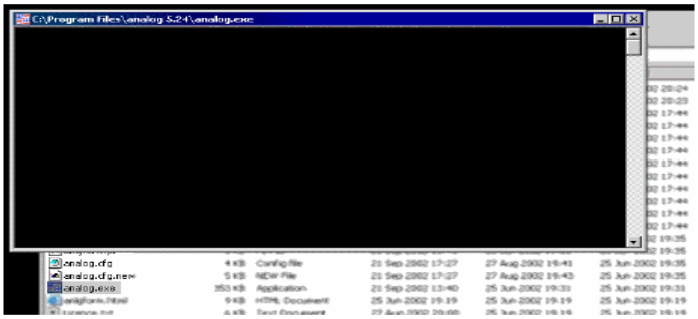
Que compila Analog y genera su ejecutable bajo el mismo directorio (/tmp/analog-<número_de_version>), el ejecutable también lleva por nombre Analog; para ser ejecutado Analog basta lo siguiente:

```
[root@OSMOSIS analog-5.1]# pwd
/tmp/analog-5.1/
[root@OSMOSIS analog-5.1]# ./analog
./analog: analog version 5.1/Unix
./analog: Warning D: Turning all pie charts off because
OUTFILE is stdout and
```

Instalación en Windows™: ANALOG para Windows™ es distribuido con todos los archivos organizados en un archivo compreso .zip, así no es necesaria ninguna instalación. Lo que se debe hacer es extraer los archivos en un directorio apropiado en el PC. El archivo .zip contiene la información de directorio de cada archivo, así dentro del directorio donde es extraído el .zip se creara el directorio de instalación que usualmente es: Analog [versión], donde versión es la versión del ANALOG que se descargo (en este caso la 6.0).

El archivo se puede descargar en: http://www.analog.cx/analog_60w32.zip

Dentro del directorio 'analog 6.0' se hace doble clic en el ejecutable 'analog.exe' lo que sucederá es que correrá por default una configuración de ejemplo: 'analog.cfg' entonces se abrirá una ventana DOS y que una vez termine el proceso cerrará por sí solo.



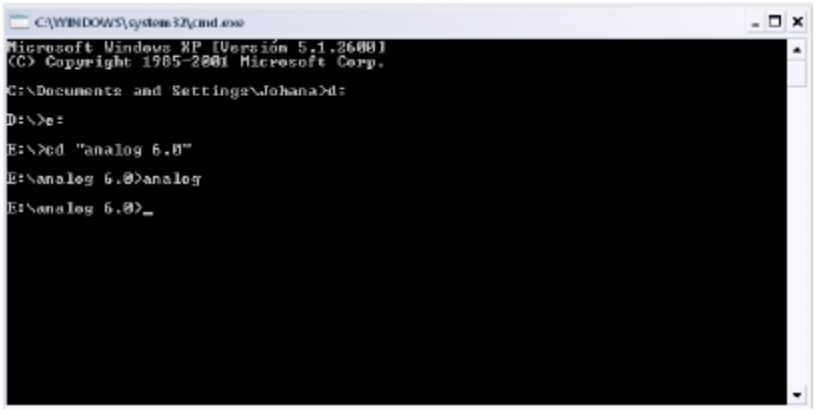
Lo más probable es que analog haya corrido exitosamente pero si queda duda de ello analog escribe cualquier error en un archivo de texto (en el mismo directorio) llamado 'error.txt'. El resultado del testeo también es una página llamada 'report.html' y varias imágenes de apoyo, dentro de esta página existe un reporte del ejemplo, si esto no se encuentra o hay un error se debe volver a instalar.

Manejo de la Herramienta

Para iniciar el proceso, se debe seleccionar la ruta del archivo log a analizar. Posteriormente debe abrirse una consola de Símbolo de Sistema (DOS) en Inicio, y situarse en el directorio de ANALOG

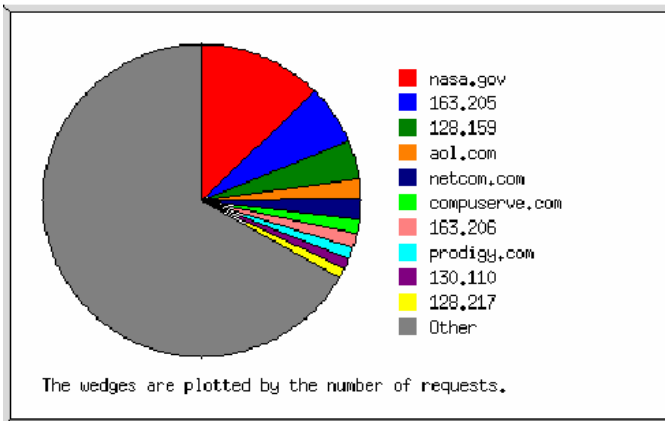
cd "program files\analog 6.0"; de esta manera es como se corre Analog.

Analog hace uso de los logs producidos por los Web Server como: APACHE HTTP Server ™, AOL Server, o un SERVLET ENGINE como Tomcat™.



El reporte se genera en un archivo .html. Si se desea cambiar el directorio donde se genera el reporte se debe modificar la línea: OUTFILE Report.html

En la siguiente figura se puede apreciar, uno de los formatos de reporte generado:



Ventajas

- La principal ventaja de Analog es que es totalmente Open Source eso posibilita su estudio y modificación.
- Analog es operable en diversas plataformas tales como: Windows™, *nix™, Mac™.

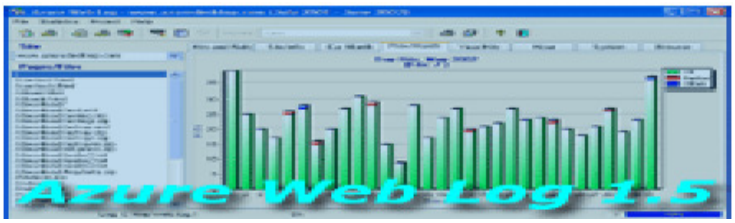
Desventajas

- No posee interfaz gráfica lo que hace que – ya sea en Linux™ o Windows™ – se maneje solo a través de comandos y siempre debe modificarse su archivo de configuración para hacer un cambio de log o en el reporte de salida, es por eso que se recomienza iniciar su uso con mínimos cambios de configuración y poco a poco ir avanzando hasta lograr los reportes al susto del usuario.
- El reporte es sencillo y estadístico. Los informes de resultados son ofrecidos hasta en 33 idiomas distintos, incluyendo el español.
- Su principal limitación es que no realiza Minería de Datos, por ello no aplica ninguna técnica ni algoritmo de la misma para la obtención de resultados.

3.5. AZURE WEB LOG 1.51

La herramienta Azure Web Log (AWL) en su versión 1.51 provee diversos tipos de estadísticas para analizar cada página como componente de un sitio. [18].

Requerimientos: Azure web Log está disponible para la mayoría de los sistemas operativos (SO) y computadoras: Windows ™ 9x/Me/NT/2000/XP.



Instalación

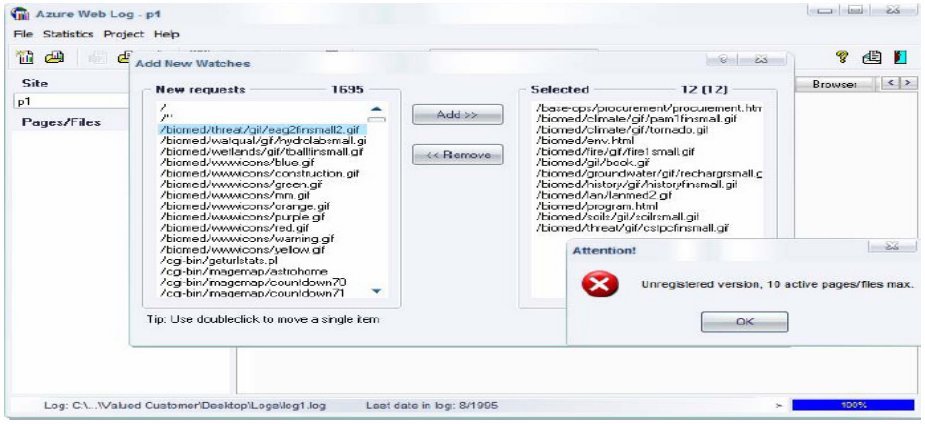
Se necesita ingresar como usuario con derechos de administración. Para instalaciones anteriores se instala la herramienta en el fólder de la versión anterior y Azure Web Log será actualizado sin necesidad de desinstalar la versión anterior, los archivos existentes serán almacenados en el proyecto por defecto (default Project). El proceso de instalación es sencillo, presentando un asistente para soporte del usuario.

Manejo de la Herramienta

El análisis da inicio cuando el usuario señala la ruta del archivo Log que desea analizar. Entre los formatos Logs que acepta la herramienta se tienen:

- Common Log Format –CLF-
- NCSA Combined Log Format –ELF-

- Microsoft IIS (W3 extended log format)

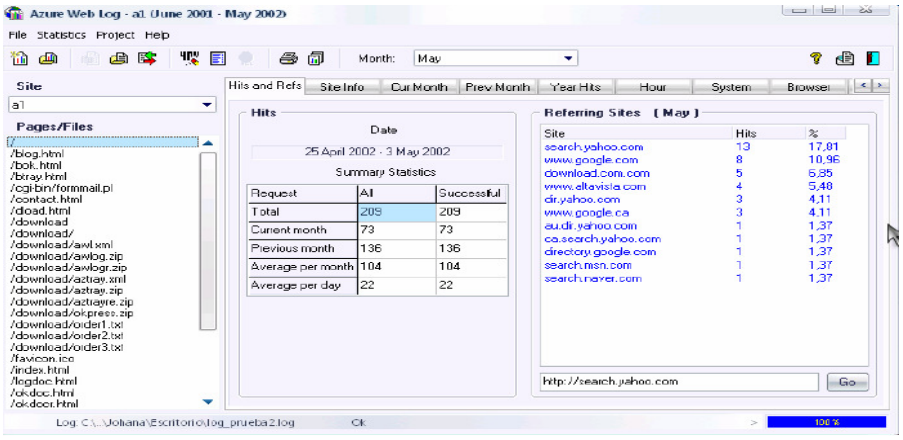


El procesamiento de AWL se hace en base a funciones estadísticas estandarizadas y orientadas hacia la búsqueda de características distintivas, funciones tales como las distribuciones conocidas.

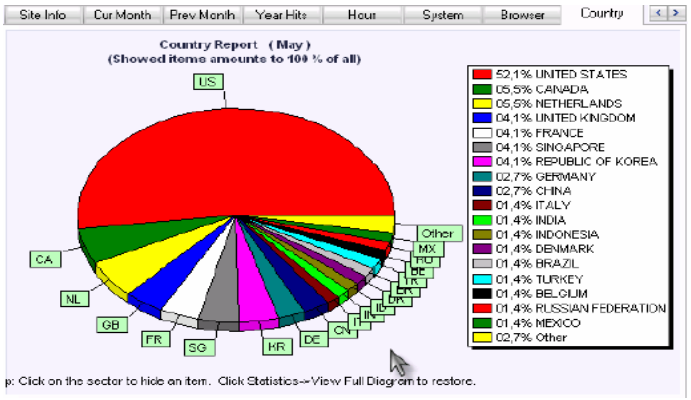
Los resultados del proceso son guardados automáticamente por la herramienta en su base de datos.

Dicho procesamiento permite encontrar:
- Url`s más visitadas
- Url`s de referencia
- El país de donde provienen los visitantes

Reportes generados:
- Hits el mes anterior y el actual
- Hits mensuales por un año
- Referring sites por mes
- Peticiones de error
- Distribución horaria de hits
- Tráfico mensual del mes
- Sistemas operativos y navegadores de los usuarios
- País del visitante



Los reportes pueden ser exportados en html para poder publicarlos o visualizarlos a través del navegador. Los reportes se pueden imprimir individualmente o de manera compuesta.



Ventajas

- Fácil de instalar
- Copia de evaluación gratuita
- Genera atractivos informes en HTML, de forma relativamente más clara y entendible, con la ayuda de barras y gráficas en 3D. Los informes se pueden enviar por E-mail.

Desventajas

- Permite un análisis muy pobre. El hecho de que en la versión de prueba solo se puedan abrir 10 registros implica que la versión comercial solo abra 400 registros.
- La versión de prueba a demás otorga un periodo de 10 días.
- Solo se limita a generar reportes estadísticos.
- No se aplica Minería de Datos para el análisis de resultados.

3.6. KM NAVIGATOR

El KM Navigator (KMN) es una herramienta poderosa para búsqueda y análisis de sitios establecidos en Internet. Está basado en una arquitectura multi-tarea, en una plataforma escalable y fiable. [21].

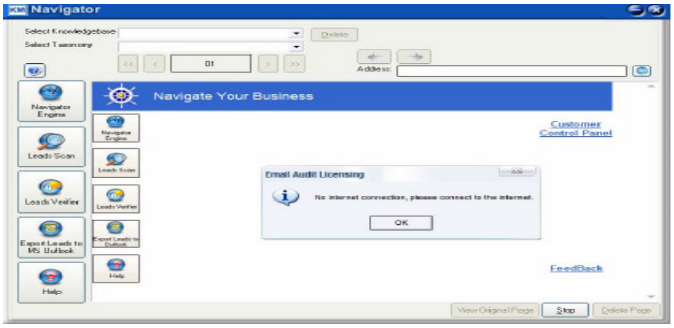
Requerimientos: KMN está disponible para SO Windows™: Windows™ 98/NT/2000/2003/XP.

Instalación

El proceso de instalación es sencillo, y ofrece un asistente para soporte del usuario.

Manejo de la Herramienta

El primer paso es conectarse a Internet para desarrollar la búsqueda.

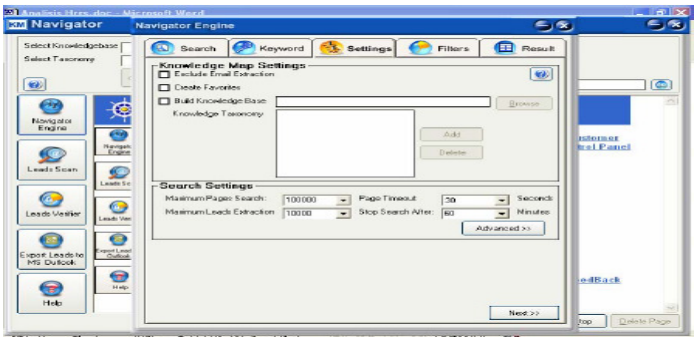


El principal objetivo de KMN es realizar Data Mining en clustering, para desarrollar un perfil de usuario en base a datos demográficos. Los usuarios son todas aquellas personas que acceden a KMN para realizar una búsqueda, lo que significa que el log es generado por cada persona que accede a esta herramienta los parámetros de búsqueda son palabras clave ingresadas o filtros activados por el usuario.

Para el análisis de los datos se pueden seleccionar los siguientes parámetros.

- Selecciona la búsqueda del sitio: dependiendo del motor que se desee utilizar
- Selecciona la lista URL's: Cuando la herramienta ha sido usada previamente genera una lista de URL's
- Desde la URL escrita: Cuando se conoce la URL que luego será registrada en la lista de URL's.
- Las búsquedas pueden ser sencillas o múltiples

En la pestaña Keyword, se digitan las palabras claves de búsqueda. En Settings se realizan las configuraciones de la búsqueda, estas configuraciones permiten búsquedas muy flexibles, a través de ellas se pueden crear favoritos. Los filtros ayudan a disminuir el listado de sitios coincidentes con las palabras claves y organizadas por su ranking.



Para la visualización de resultados se realiza una búsqueda de los archivos de registro que el usuario ya ha generado luego de las búsquedas realizadas. Los archivos de registro pueden ser exportados a MS Outlook.

Ventajas

- Permite el manejo de filtros activados por el usuario, para restringir criterios de análisis.
- Fue construido bajo el kernel de Windows™ por lo que posee una compatibilidad total con los sistemas operativos Windows™.

Desventajas

- Solo funciona bajo sistemas operativos Windows™.
- El periodo de prueba es por 15 días.
- Esta herramienta no abre ningún tipo de log externo.
- En la versión de prueba de KM Navigator no se pueden editar las bases de datos, y la base de datos creada solo contiene 100 páginas incluida su taxonomía.
- El Navigator Engine no puede guardar los resultados exitosos.
- Leads Scan, no puede guardar los logs generados por el usuario.
- Leads Verifier, no está disponible.
- Export leads to MS-Outlook, solo se pueden exportar 10 registros a MS out look
- No realiza ningún proceso de Minería de Datos.

3.7. WEB CUSTOMER ANALYSIS

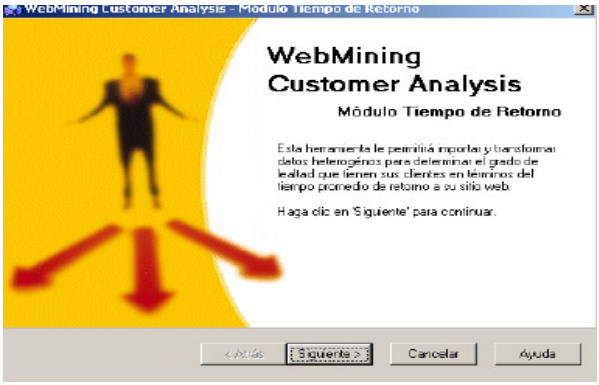
El WebMining Customer Analysis (WCA) trabaja con diversos tipos de datos que pueden ser importados, exportados y transformados para determinar la frecuencia de los visitantes de su sito; en términos del tiempo promedio de retorno del visitante. [21].

Requerimientos

- Windows™ 9x/2000 o superior
- Procesador Pentium™ o superior
- Aproximadamente 150Kb de espacio libre en disco
- Visual Basic 6.0 Run-time Files descargar
- Microsoft ActiveX Data Object 2.6 Library descargar
- Microsoft SQLDMO Object Library (versión 8.0, que viene con SQL Server 2000) descargar.

Instalación.

Como la mayoría de las herramientas, ya nombradas, el WCA, presenta un asistente de instalación para dar soporte al usuario.



Manejo de la herramienta

Para un adecuado uso del WCA, se deben seguir los siguientes pasos:

Paso 1: Seleccionar un origen de datos Seleccione de donde provienen sus datos (servidor sql, archivo excel, archivo de texto, etc.). Lo cual no se puede realizar en una versión de prueba, ya que en esta solo se puede ingresar un servidor tipo SqlServer, como se puede apreciar en la instalación.

Paso 2: Selección de campos de análisis; es decir a los que identifican al cliente y a la transacción realizada.

Paso 3: Rango de tiempo, se debe seleccionar un período de tiempo para realizar el análisis.

Paso 4: Análisis y tiempo de retorno realice el análisis y obtenga un reporte con los tiempos de retorno de sus clientes.

Desventajas

- Se realiza una minería de uso muy pobre, ya que solo muestra la incidencia de los usuarios desaprovechando toda la capacidad de este tipo de minería.
- Esta herramienta posee una gran desventaja en la versión de prueba y es que necesita que el log sea convertido a un objeto ODBC en Microsoft sql server

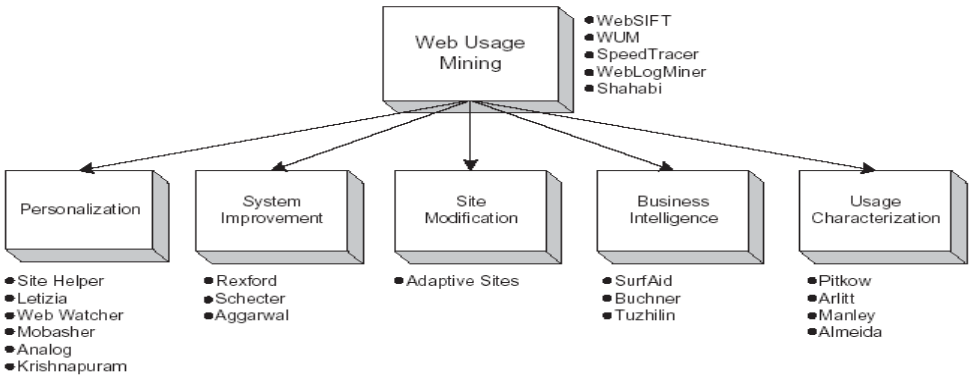
2000, este proceso tiene un índice de complejidad considerable, teniendo en cuenta que la mayoría de logs se presentan en archivos planos.

3.8. OTRAS HERRAMIENTAS

Existen también en el mercado otras muy reconocidas herramientas utilizadas por las grandes empresas en las diferentes áreas de la minería de uso, tales como WebSift, WUM, Letizia, entre otras.

En la figura 2.14 se incluyen algunas de ellas dentro de su específico campo de aplicación:

Figura 2.14. Herramientas de Minería Web de Uso



Aunque este tipo de herramientas son reconocidas, no fue posible encontrar la respectiva documentación de alguna de ellas. Sin embargo, a continuación se da una breve descripción de las principales herramientas encontradas:

OBIWAN. Esta herramienta trabaja con un sistema de agentes inteligentes y con una ontología personalizada que es creada a partir de una lista de conceptos organizados jerárquicamente. Posteriormente son asignados unos pesos para obtener un mapa de referencia ontológico para llegar finalmente a la ontología propia del usuario, la cual es usada para navegar y hacer búsquedas en la Web. [6]

Ventajas

- El sistema de agentes inteligentes define una ontología de acuerdo al perfil del usuario.

- Posee un agente que construye ontologías.
- Evalúa la eficiencia de la interfaz gráfica.
- Emplea agentes colaboradores para organizar la información que se recupera de la Web.

Desventajas
- No realiza identificación de perfiles de usuario.
- No existe categorización de usuarios.
- No modela grupos de usuarios (no hace clustering).
- No tiene un agente para analizar patrones de uso del usuario.
- No tiene sistema de filtros de información.

WEBMINER. El Webminer es un sistema que implementa una arquitectura en la que se divide el proceso de Web Usage Mining en dos partes: La primera parte, incluye el proceso de transformación de los logs en formatos que se ajusten a las transacciones. Esto incluye pre-procesamiento, identificación de transacción e integración de componentes de datos. La segunda parte incluye técnicas de minería de datos y reconocimiento de patrones. [28]

Ventajas
- Aplica técnicas de minería de datos, facilitando el descubrimiento de información a través de las reglas de asociación y los patrones secuenciales de los datos web.
- Aplica técnicas de descubrimiento de conocimiento.
- Define asociación de transacciones web.
- Hace análisis de patrones secuenciales.
- Propone como trabajo futuro el desarrollo de agentes autónomos que analicen el descubrimiento de reglas de clasificación.
- Propone otro trabajo futuro que desarrolle un mecanismo de consultas que pueda ser manipulado en los procesos de limpieza de los datos y la identificación de transacciones.

Desventajas

- No utiliza los perfiles de usuario para el análisis de patrones.
- No existe agrupamiento de perfiles de usuario.
- No tiene sistema de filtros de información.

WEB SITE INFORMATION FILTERING – WEBSIFT- . El Websift es un sistema que utiliza el contenido y la estructura de información de los sitios web para identificar los resultados potencialmente interesantes de los datos de uso.

El sistema Websift está diseñado para realizar minería de uso de los logs de servidores en el formato NSCA. El algoritmo de preprocesamiento incluye identificación de usuarios, sesiones de servidor e inferencias de las páginas guardadas en la memoria caché.

En la creación de una sesión de servidor, el sistema desarrolla preprocesamiento de contenido. Los archivos de las sesiones de servidores pueden ser llevados a través de algoritmos de análisis de patrones secuenciales, reglas de asociación, clustering o algoritmos de estadística. [9]

Ventajas

- Propone una solución para el problema de la identificación de los patrones de utilización de la Web que pueden se considerados como interesantes.
- Hace un seguimiento de la utilización de la web por parte de los usuarios.
- Realiza filtrado de información para encontrar patrones interesantes.
- Propone trabajos futuros para la incorporación de patrones secuenciales y descubrimiento de clusters de los datos de utilización de la web.
- También propone como trabajo futuro, que en la etapa de análisis de patrones específicamente en el filtrado, se empleen métodos probabilísticas o de lógica difusa.

Desventajas

- No utiliza los perfiles de usuario para el análisis de patrones.
- No se utilizan las preferencias de los usuarios.

WUM. El WUM es un sistema para el descubrimiento de patrones interesantes en la navegación. Los criterios para establecer los patrones interesantes son especificados dinámicamente utilizando el lenguaje MINT, el cual soporta las especificaciones de criterios estadísticos, estructurales y de la naturaleza del texto. Para descubrir los patrones de navegación, WUM aprovecha la información almacenada en los logs del servidor Web**.** [33]

Ventajas

- Descubre patrones de navegación interesantes.

- Utiliza los web logs para el descubrimiento de patrones.
- Hace agrupamiento de las páginas consecutivas requeridas por un usuario en una transacción.
- Además, tiene en cuenta la duración máxima entre los accesos de páginas subsecuentes.
- Incorpora un nuevo mecanismo para la agregación de patrones de datos a través de una estructura de árbol.
- Provee un lenguaje de consulta de conocimiento, en el cual el experto puede especificar las características que hacen interesante a un patrón de navegación.

Desventajas

- Los criterios para los patrones de navegación interesantes deben ser dinámicamente especificados por el usuario del sistema.
- No hace identificación de perfiles de usuario.
- No hace agrupamiento de perfiles.

LETIZIA. Letizia es un agente de interfaz de usuario que asiste al usuario en la navegación a través de la web. Cuando el usuario navega en un browser web convencional como Netscape™, el agente Letizia rastrea el comportamiento del usuario e intenta anticipar los artículos de interés haciendo exploración concurrente; además, explora de forma autónoma los links de las posiciones en que se encuentra el usuario. Este agente utiliza una estrategia que consiste en mostrar automáticamente la mejor búsqueda, la cual es aumentada con la inferencia heurística que deduce el interés del usuario en la navegación. [16]

Ventajas

- Posee un agente de interfaz eficiente.
- Acompaña y asesora al usuario en su navegación, resaltando los vínculos que son de su interés y mostrando cuáles debe evitar.
- Aprende un perfil a partir de los intereses del usuario guardando y analizando la actividad que realiza el usuario en el browser en tiempo real, suministrando un flujo continuo de recomendaciones a páginas Web.
- El usuario no necesita prestarle atención al agente de interfaz, pues trabaja autónomamente.
- Propone como trabajo futuro el análisis de patrones de los comportamientos de los usuarios al utilizar la web.

Desventajas

- Enfoque centralizado utilizando solo un agente de software.
- El sistema está localizado sobre una sola máquina.
- Hace exploración local.
- No hace categorización de usuarios.
- No identifica perfiles de grupos.
- No hace filtrado de información.

WEBWATCHER. El WebWatcher es un agente guía de un tour para la web. Una vez el usuario le dice al agente qué clase de información busca, el agente lo acompaña de página en página en su proceso de navegación, resaltando los enlaces que cree son de su interés. Su estrategia para dar consejos se basa en la realimentación basada en los tours anteriores. [22]

Ventajas

- Posee un agente que guía al usuario cuando navega en la web, éste actúa como un agente de interfaz entre el usuario y la web.
- Tiene en cuenta la localización de las colecciones de datos en la web.
- Tiene en cuenta cómo otros han interactuado con la colección en el pasado.
- Aprende de las experiencias para dar mejores consejos.
- Puede aprender de las relaciones entre consultas aunque no tengan ninguna palabra en común.
- Provee un motor de búsqueda con palabras clave usando una variante del motor de búsqueda, aplicándolo al conjunto de páginas previamente visitadas por WebWatcher.
- Tiene la posibilidad de mostrar páginas similares a las consultadas, basándose en el análisis de la estructura del hipertexto.
- Construye sus propios logs de acceso.
- Propone un método de aprendizaje basado en una función objetivo definida entre [0,1] llamada ***Calidad_del_Link,*** la cual da la posibilidad de que un usuario seleccione un link dada la página actual y el interés.
- Propone tres enfoques para aprender de la función objetivo: Aprendizaje de navegaciones previas, reforzamiento de aprendizaje o la combinación de ambos.

Desventajas

- No construye perfiles de usuario.
- Enfoque centralizado.
- No identifica perfiles de grupos de usuarios.

- Hace exploración local.
- Las pruebas del sistema fueron hechas en modo offline.
- No es un sistema personalizado.
- No tiene en cuenta el dominio de trabajo del usuario.
- No tiene en cuenta las preferencias del usuario.

SYSKILL Y WEBERT. Este sistema permite a los usuarios valorar a través de marcas las páginas que visitan y con ello el sistema construye los perfiles de usuario. Las marcas pueden ser positivas, negativas o indiferentes. Basado en el perfil, cuando el usuario navega a una nueva página, el sistema hace una clasificación previa de la información apuntada por los links sobre la página, resalta los links que deben ser recomendados e indica los links que deben ser evitados. [5].

Ventajas

- El sistema ofrece recomendación a los usuarios sobre los enlaces que debe visitar, haciendo diferencia entre los relevantes y los que debe evitar.
- Puede utilizar motores de búsqueda para devolver el tema de una consulta.
- Construye perfiles de usuario analizando la información de cada página.
- Representa el perfil del usuario en una forma probabilística.
- Aprende a valorar páginas a partir de la clasificación realizada por el usuario.
- Aprende un perfil diferente por cada uno de los temas manejados por el usuario.
- Hace selección de características con métodos probabilísticas.
- Los autores investigaron la precisión de 5 algoritmos de aprendizaje de máquina para construcción de perfiles: Redes Bayesianas, K-medias, PEBLS, Árboles de Decisión (ID3) y Redes Neuronales, llegando a concluir que el algoritmo más rápido para aprender y predecir era el Redes Bayesianas. Adicionalmente, en este algoritmo el tiempo de aprendizaje es lineal en el número de ejemplos y su tiempo de predicción es independiente del número de ejemplos.

Desventajas

- Enfoque centralizado.
- No hace categorización de perfiles de usuario.
- El usuario debe ingresar las marcas positivas, negativas e indiferentes sobre cada página visitada.
- Proceso semi-automático.
- Ofrece una forma más restringida de navegación que el Web Watcher y Letizia.

- No realiza agrupamiento de perfiles.
- No realiza filtrado de información.

4. HERRAMIENTAS PARA LA CONSTRUCCIÓN DE LA HERRAMIENTA POLARIS

4.1. LENGUAJE DE PROGRAMACION

El lenguaje de programación para la implementación de la herramienta es Java™, el cuál es en sí una plataforma independiente tanto de hardware como de software gracias al soporte de máquina virtual.

Java™ es un lenguaje de programación orientado a objetos que surgió en 1991 cuando un grupo de ingenieros de *Sun Microsystems* trataron de diseñar un nuevo lenguaje de programación destinado a electrodomésticos. La reducida potencia de cálculo y memoria de los electrodomésticos llevó a desarrollar un lenguaje sencillo capaz de generar código de tamaño muy reducido.

Java™, como lenguaje de programación para computadores, se introdujo a finales de 1995. La clave fue la incorporación de un intérprete ***Java***™ en el programa Netscape™ Navigator, versión 2.0, produciendo una verdadera revolución en Internet. ***Java***™ ***1.1*** apareció a principios de 1997, mejorando sustancialmente la primera versión del lenguaje. Al programar en ***Java***™ no se parte de cero.

Cualquier aplicación que se desarrolle se apoya en un gran número de clases preexistentes. Algunas de ellas las ha podido hacer el propio usuario, otras pueden ser comerciales, pero siempre hay un número muy importante de clases que forman parte del propio lenguaje (el ***API*** o ***Application Programming Interface*** de ***Java***™).

Java™ incorpora muchos aspectos que en cualquier otro lenguaje son extensiones propiedad de empresas de software o fabricantes de ordenadores (threads, ejecución remota, componentes, seguridad, acceso a bases de datos, etc.). Por eso es un lenguaje ideal para aprender la informática moderna, porque incorpora todos estos conceptos de un modo estándar, mucho más sencillo y claro que con las citadas extensiones de otros lenguajes. Esto es consecuencia de haber sido diseñado más recientemente y por un único equipo.

El principal objetivo del lenguaje ***Java***™ es llegar a ser el "nexo universal" que conecte a los usuarios con la información, esté ésta situada en el ordenador local,

en un servidor de ***Web***, en una base de datos o en cualquier otro lugar. ***Java***™ es un lenguaje muy completo que se está convirtiendo en un macro-lenguaje: ***Java***™ ***1.0*** tenía 12 packages; ***Java***™ ***1.1*** tenía 23 y ***Java***™ ***1.2*** tiene 59.
La compañía ***Sun*** describe el lenguaje ***Java***™ como "*simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico*".

4.1.1. Principales Características de Java™

Lenguaje Orientado a Objetos. Es una característica en la que el software se desarrolla de forma que los distintos tipos de datos que use están unidos a sus operaciones, de esta manera las clases, datos y el código se combinan en entidades llamadas objetos.

Una ***clase*** es una agrupación de ***datos*** (variables, campos o elementos declarados de la clase) y de ***funciones*** (métodos) que operan sobre esos datos. A estos datos y funciones pertenecientes a una clase se les denomina ***variables*** y ***métodos*** o ***funciones miembro***.

La programación orientada a objetos se basa en la programación de clases. Un programa se construye a partir de un conjunto de clases. Esto hace posible que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos. La **herencia** permite que se puedan definir nuevas clases basadas en clases existentes, lo cual facilita re-utilizar código previamente desarrollado.

Independencia de la plataforma. Debido a la existencia de distintos tipos de CPUs y a los continuos cambios, es importante conseguir una herramienta independiente del tipo de procesador utilizado. El desarrollo de un código "neutro" que no depende del tipo CPU utilizada permitió lograr lo que luego se ha convertido en el principal lema del lenguaje: "***Write Once, Run Everywhere***".

De esta manera se logra obtener un código capaz de ejecutarse en cualquier tipo de máquina, el cual una vez compilado no necesita de ninguna modificación por el hecho de cambiar de procesador o de ejecutarlo en otra máquina. La clave consiste en desarrollar un código "neutro" el cual esta preparado para ser ejecutado sobre una "*máquina hipotética o virtual*", denominada ***Java Virtual Machine*** (***JVM***).

Es esta ***JVM*** quien ***interpreta*** este código neutro convirtiéndolo a código particular de la CPU o chip utilizado, evitando así el tener que realizar un programa diferente para cada CPU o plataforma. La ***JVM*** es el intérprete de ***Java***™. Ejecuta los "***bytecodes***" (ficheros compilados con extensión ****.class***) creados por el compilador de ***Java***™ (***javac.exe***). Tiene númerosas opciones entre las que destaca la posibilidad de utilizar el denominado ***JIT*** (*Just-In-Time Compiler*) que interpreta el bytecode generado convirtiéndolo a instrucciones máquina del código nativo. El JIT puede mejorar entre 10 y 20 veces la velocidad de ejecución de un programa.
El recolector de basura: Uno de los aspectos más importantes en la programación orientada a objetos (OOP) es la forma en la cual son creados y eliminados los objetos. La eliminación de los objetos la realiza el denominado *garbage collector*, quien automáticamente libera o borra la memoria ocupada por un *objeto* cuando no existe ninguna referencia apuntando a ese objeto. El programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java™ (Java runtime) es el responsable de gestionar el ciclo de vida de los objetos.

Cuando no quedan referencias a un objeto, el recolector de basura de Java™ borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas de memoria, que ocurre cuando el sistema operativo piensa que una zona de memoria está siendo usada por una aplicación cuando en realidad no es así. Concluyendo se puede decir que el recolector de basura de Java™ permite una fácil creación y eliminación de objetos de una manera rápida y segura.

4.2. ENTORNO DE DESARROLLO

El entorno de desarrollo elegido para la implementación es Netbeans™ IDE 5.0, un software libre y con una buena interfaz que facilita la tarea de programación.

Netbeans™ es una plataforma para el desarrollo de aplicaciones utilizando Java™ y un entorno de desarrollo integrado (IDE), permite editar programas en java™, compilarlos, ejecutarlos, depurarlos y construir rápidamente la gráfica de una aplicación.

La plataforma NetBeans™ permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados *módulos*. Un módulo es un archivo Java™ que contiene clases de java™ escritas para interactuar con las APIs de NetBeans™ y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos.

Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans™ pueden ser extendidas fácilmente por otros desarrolladores de software.

Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas)
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas
- Framework basado en asistentes (diálogos pasos a paso).

4.3. CONTROLADOR JDBC

Un Java Database Connectivity - JDBC es un API de Java™ para acceder a sistemas de bases de datos, y prácticamente a cualquier tipo de dato tabular. El API JDBC consiste de un conjunto de clases e interfaces que permiten a cualquier programa Java™ acceder a sistemas de bases de datos de forma homogénea. En otras palabras, con el API JDBC no es necesario escribir un programa para accesar a Sybase™, otro programa para accesar a Oracle, y otro programa para accesar a MySQL; con esta API, se puede crear un sólo programa en Java™ que sea capaz de enviar sentencias SQL a la base de datos apropiada.

Al igual que ODBC, la aplicación de Java™ debe tener acceso a un controlador (*driver*) JDBC adecuado. Este controlador es el que implementa la funcionalidad de todas las clases de acceso a datos y proporciona la comunicación entre el API JDBC y la base de datos real. De manera muy simple, al usar JDBC se pueden hacer tres cosas:

- Establecer una conexión a una fuente de datos (ej. una base de datos).
- Mandar consultas y sentencias a la fuente de datos.
- Procesar los resultados.

Los distribuidores de bases de datos suministran los controladores que implementan el API JDBC y que permiten acceder a sus propias implementaciones de bases de datos. De esta forma JDBC proporciona a los programadores de Java™

una interfaz de alto nivel y les evita el tener que tratar con detalles de bajo nivel para acceder a bases de datos.

4.4. BIBLIOTECA GRÁFICA SWING DE JAVA™

Swing es una biblioteca gráfica para Java™ que hace parte de las Java™ Foundation Classes que comprende un grupo de componente para ayudar a construir interfaces gráficos de usuario (GUIs).

JComponent. La mayoría de los componentes Swing están implementados como subclases de la clase **JComponent**, que desciende de la clase **Container**. De **JComponent**, los componentes Swing heredan las siguientes funcionalidades:

- **Bordes.** Usando el método **setBorder**, se puede especificar el borde que muestra un componente alrededor de sus lados.
- **Doble buffer.** El doble buffer puede mejorar la apariencia de un componente que cambie frecuentemente. No hay necesidad de escribir el código del doble buffer ya que Swing lo proporciona.
- **Tool tips.** Especificando un string con el método **setToolTipText**, se proporciona ayuda al usuario de un componente. Cuando el cursor se para sobre el componente, el String especificado se muestra en una pequeña ventana que aparece cerca del componente.
- **Navegación con Teclado.** Usando el método **registerKeyboardAction**, el usuario puede usar el teclado en vez del ratón para moverse por el GUI.

JFrame. Es un contenedor Swing de alto nivel que proporciona ventanas para applets y aplicaciones. Un frame tiene decoraciones como un borde, un título, y botones para cerrar y minimizar la ventana. Un programa típico simplemente crea un frame, añade componentes al panel de contenido, y quizás añade una barra de menú.

JPanel. Es un contenedor de propósito general para componentes de peso ligero. Como todos los contenedores, utiliza un **Controlador de Distribución** para posicionar y dimensionar sus componentes. Como todos los componentes Swing, **JPanel** permite añadirle bordes y determinar si utiliza el doble buffer para aumentar el rendimiento.

JTabbedPane. Con la clase **JTabbedPane**, se puede tener varios componentes (normalmente objetos **JPanel**) compartiendo el mismo espacio. El usuario puede elegir qué componente ver se selecciona la pestaña del componente deseado.

JTree. Con la clase **JTree**, se puede mostrar un árbol de datos. **JTree** realmente no contiene datos, simplemente es un vista de ellos. **JTree** muestra los datos verticalmente. Cada fila contiene exactamente un ítem de datos (llamado un nodo). Cada árbol tiene un nodo raíz (llamado Root en la figura anterior, del que descienden todos los nodos. Los nodos que no pueden tener hijos se llaman nodos leaf (hoja).

JScrollPanne. Un **JScrollPane** proporciona una vista desplazable de un componente ligero. Cuando el estado de la pantalla real está limitado, se utiliza un ScrollPane para mostrar un componente que es grande o cuyo tamaño puede cambiar dinámicamente.

JSplitPane. Un **JSplitPane** contiene dos componentes de peso ligero, separados por un divisor. Arrastrando el divisor, el usuario puede especificar qué cantidad de área pertenece a cada componente. Un SplitPane se utiliza cuando dos componentes contienen información relacionada y queremos que el usuario pueda cambiar el tamaño de los componentes en relación a uno o a otro.

JTable. Con la clase **JTable**, se pueden mostrar tablas de datos, y opcionalmente permitir que el usuario los edite. **JTable** no contiene ni almacena datos; simplemente es una vista de los datos.

JFileChooser. La clase **JFileChooser** proporciona un UI para elegir un fichero de una lista. Un selector de ficheros es un componente que se puede situar en cualquier lugar del GUI. La clase **JFileChooser** hace sencillo traer un diálogo modal que contiene un selector de ficheros.

JtextArea. Es un área de texto que muestra múltiples líneas de texto y permite que el usuario edite el texto con el teclado y el ratón.

JradioButton. Son grupos de botones en los que, por convención, sólo uno de ellos puede estar seleccionado, exhibiendo de esta manera su estado al usuario.

JcomboBox. Es un componente con el que el usuario puede teclear un valor o elegirlo desde una lista. Un ComboBox editable ahorra tiempo de entrada proporcionando atajos para los valores más comúnmente introducidos.

JSpinner. Esta clase provee en una sola línea la posibilidad al usuario de entrar o seleccionar un valor de una secuencia pedida. Los JSpinner proporcionan

generalmente un par de los botones minúsculos de flechas para cambiar entre los elementos de la secuencia.

4.5. JAVA™ 3D

Java™ 3D es un proyecto que permite crear entornos tridimensionales en el lenguaje Java™. Es una API para gráficos 3D para el lenguaje de programación Java™ la cual corre sobre OpenGL o Direct3D. Desde la version 1.2 Java™ 3D es desarrollado bajo las especificaciones Java™ Community Process. JSR 926. Permite escribir programas que desplieguen gráficas tridimensionales además de interactuar con ellas.

Java™ 3D es una extensión del Java™ 2 JDK. El API contiene un conjunto de constructores de alto nivel para crear y manipular geometrías en 3D, así como de las técnicas necesarias para dibujar la geometría en una imagen. Java™ 3D contiene las funciones necesarias para crear imágenes, visualizaciones, animaciones y aplicaciones con objetos gráficos interactivos.

Todos los programas de Java™ 3D se forman, al menos parcialmente, uniendo distintos elementos de la jerarquía de clases de Java™ 3D. Esa gran colección de objetos describe un universo virtual, que será posteriormente renderizado.

La API define más de cien clases incluidas en el paquete javax.media.j3d. Estas clases son las normalmente denominadas como clases núcleo de Java™ 3D.

Hay cientos de campos y métodos en dichas clases, sin embargo, un universo virtual sencillo que contenga algo de animación se puede construir con sólo unas pocas de ellas.

A la hora de desarrollar programas en Java™ 3D se utilizan otros paquetes, además del paquete núcleo. Uno de ellos es el paquete normalmente conocido como de utilidad (com.sun.j3d.utils).El paquete núcleo incluye sólo las clases de nivel más bajo necesarias para programar en Java™ 3D.

Las clases de utilidad son extensiones muy prácticas y potentes. Como era de esperar, la utilización de clases de utilidad reduce significativamente el número de líneas de código a la hora de programar.

4.6. OTROS

Además de lo anterior, se utilizó:

postgresql.jar: para poder acceder al driver para realizar la conexión de la base de datos desde java™. Al compilar este archivo se puede realizar la conexión entre JDBC de java™ con el SGBD de Postgres.

JFreeChart: Este componente permite gráficar todo tipo de datos en diferentes tipos de vistas tales como gráficos en barra, en pastel o lineal.

Jdom.jar: esta librería es utilizada en todos los procesos de escritura en disco (abrir y guardar) de los archivos del área de trabajo con un formato xml.

substance.jar: Esta librería es utilizada para cambiar la apariencia de un programa –interfaz gráfica- facilitando la modificación de botones y ventanas.

5. ANÁLISIS, DISEÑO Y MODELADO

5.1. METODOLOGÍA DE ANÁLISIS Y DISEÑO

La metodología de desarrollo utilizada en Polaris es la **metodología de desarrollo de software orientada a objetos**, la cuál es extractada de las principales metodologías existentes, en particular:

- Object Oriented Design del autor Grady Booch [31], en la cual se desarrollan las tareas de análisis de requerimientos, análisis de dominio y diseño.
- Objectory del autor Ivar Jacobson [32], en la cual se desarrollan las tareas de análisis de requerimientos, análisis de robustez, diseño, implementación y pruebas.
- Object Modeling Technique del autor James Rumbaught [31], en la cual se desarrollan las tareas de análisis. diseño del sistema, diseño de objetos e implementación.

Estas tres metodologías se fusionaron a finales de 1997 en una sola, basada en la notación UML.

Etapas y actividades en el desarrollo orientado a objetos basado en UML

Las etapas comprendidas en esta metodología son las siguientes:

- Análisis de requerimientos
- Diseño del sistema
- Diseño detallado
- Implementación y pruebas

Análisis de requerimientos: En esta etapa se logra claridad sobre lo que desea el usuario y la forma en la cual se le va a presentar la solución que está buscando.

- Actividades técnicas:
  - Identificar casos de uso del sistema.
  - Dar detalle a los casos de uso descritos.
  - Definir, si es aplicable, una interfaz inicial del sistema.

  - Desarrollar el modelo del mundo.
  - Válidar los modelos.
- Documentos entregables:
  - Casos de uso iniciales.
  - Borradores de interfaz.
  - Modelo del mundo inicial.

Diseño del Sistema: En esta etapa se define una subdivisión en aplicaciones del sistema (si es lo suficientemente grande) y la forma de comunicación con los sistemas ya existentes con los cuales debe interactuar.

- Actividades técnicas:
  - Identificar la arquitectura del sistema.

- Documentos entregables:
  - Diagramas de ejecución versión inicial.

Diseño detallado: En esta etapa se adecúa el análisis a las características específicas del ambiente de implementación y se completan las distintas aplicaciones del sistema con los modelos de control, interfaz o comunicaciones, según sea el caso.

- Actividades técnicas:
  - Agregar detalles de implementación al modelo del mundo.
  - Desarrollar el modelo de interfaz.
  - Desarrollar los modelos de control, persistencia y comunicaciones.

- Documentos entregables:
  - Diagramas de clases y paquetes, con el detalle de la implementación.
  - Diagramas de interacción con el detalle de las operaciones más
  - importantes del sistema.

Implementación y pruebas: Se desarrolla el código de una manera certificada.

- Actividades técnicas:
  - Definir estándares de programación.
  - Codificación y pruebas unitarias.
  - Pruebas de módulos y del sistema.
- Documentos entregables:

- Código fuente.
- Soporte de pruebas unitarias.
- Documentación del código.

5.2. LENGUAJE UNIFICADO DE MODELADO

El lenguaje unificado de modelado es una consolidación de muchas de las notaciones y conceptos orientados a objetos. Empezó como una consolidación del trabajo de Grade Booch, James Rumbaugh, e Ivar Jacobson, creadores de tres de las metodologías orientadas a objetos más populares.

En 1996, el Object Management Group (OMG), un pilar estándar para la comunidad del diseño orientado a objetos, publicó una petición con propósito de un metamodelo orientado a objetos de semántica y notación estándares.

UML prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

En UML se cuentan con 9 diagramas que sirven modelar los diferentes tipos de sistemas:

- Diagramas de casos de uso. Son utilizados para modelar los procesos 'business'.
- Diagramas de secuencia. Son utilizados para modelar el paso de mensajes entre objetos.
- Diagramas de colaboración. Son utilizados para modelar interacciones entre objetos.
- Diagramas de estado. Son utilizados para modelar el comportamiento de los objetos en el sistema.
- Diagramas de actividad. Son utilizados para modelar el comportamiento de los casos de uso, objetos u operaciones.
- Diagramas de clases. Son utilizados para modelar la estructura estática de las clases en el sistema.
- Diagramas de objetos. Son utilizados para modelar la estructura estática de los objetos en el sistema.
- Diagramas de componentes. Son utilizados para modelar componentes.

- Diagramas de implementación para modelar la distribución del sistema.

Diagrama de clases: El diagrama de clases es un modelo estático del sistema en el que se representan clases, interfaces, colaboraciones y relaciones.

Los diagramas de clases se utilizan para modelar el vocabulario de un sistema, las colaboraciones y esquemas lógicos de bases de datos.

Gráficamente una clase se representa mediante un rectángulo que contiene el nombre de la clase. El nombre de la clase es una cadena de texto que la identifica de manera única e inequívoca, esta cadena debe comenzar con letra minúscula y si está formado por varias palabras se escribirán sin espacio y cada una comenzará con letra mayúscula.

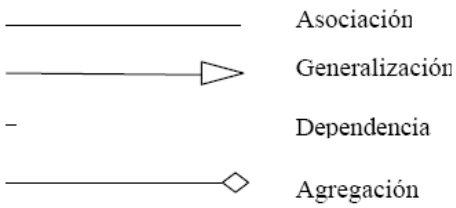
Cuando en el rectángulo solo aparece el nombre de la clase se dice que ésta tiene un nombre simple, cuando aparece el nombre de la clase precedido por el nombre del paquete, se denomina nombre de camino. Las clases abstractas se representan escribiendo el nombre en cursiva.

- **Atributos.** Una clase puede tener o no tener atributos, éstos pueden aparecer o no aparecer en el diagrama de clases. Si se incluyen en el diagrama, se listan en un rectángulo bajo el nombre de la clase. Los atributos se escriben en letras minúsculas, incluso la primera letra, pero si están formados por más de una palabra a partir de la segunda tendrán la primera letra en mayúscula.

- **Operaciones.** Las operaciones representan acciones que desarrollan los objetos, por lo general se escriben como una expresión verbal. Es opcional incluirlos en el diagrama de clases, si se desea especificarlos, se escribirán en un rectángulo debajo de los atributos. El nombre de las operaciones se escribirá en minúscula, la primera letra de cada palabra será mayúscula, excepto para la primera. Las operaciones pueden especificar el tipo de retorno, si es que lo tienen, y los parámetros con sus tipos.

- **Responsabilidades**. Son un contrato u obligación de la clase, indican lo que ha de hacer cada objeto. Las responsabilidades se cumplen a través de las operaciones, las cuales pueden apoyarse en los atributos y los estados del objeto. Una clase debe tener como mínimo una responsabilidad, pueden ser más de una, pero es preferible que no sean muchas. Las responsabilidades se especifican de manera textual en un compartimiento debajo de las operaciones.

Las relaciones modelan conexiones físicas, semánticas o lógicas entre elementos de un diagrama. La notación para representarlas es la siguiente:



- **Asociación**. Es una relación estructural que indica que los objetos de una clase están conectados con los objetos de otra.

- **Generalización**. Es una relación que asocia una clase general con una clase especializada que posee la estructura y operaciones de la primera y que podría sustituirla.

- **Dependencia**. Es una relación de uso donde una clase depende de otra, es decir, que un cambio en la clase independiente tendrá efecto en la clase dependiente. Una relación de dependencia se presenta cuando una clase utiliza objetos de otra como argumentos de sus operaciones.

- **Agregación**. Es una especialización de la asociación entre dos clases que indica que los objetos de una clase forman parte de los objetos de la otra, es decir que estructuralmente un objeto contiene a otro.

Diagrama de paquetes: El Diagrama de paquetes, muestra como un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones o paquetes. Un paquete es un elemento de propósito general que se utiliza para agrupar elementos. Los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema. Los paquetes se dibujan como rectángulos, las dependencias se muestran como flechas con líneas discontinuas. El operador "::" permite designar una clase definida en un contexto distinto del actual.
Diagrama de casos de uso: Un caso de uso especifica el comportamiento del sistema o parte de él en relación con la intervención de un usuario, define un conjunto de acciones que se realizan para generar un resultado observable e importante para un actor.

Un diagrama de casos de uso muestra los casos de uso de un sistema o parte de él, los actores y las relaciones que se establecen entre ellos. Los límites del sistema se indican mediante un cuadro o un rectángulo, dentro de éste se coloca los casos de uso y fuera, los actores. Los casos de uso se representan mediante elipses y los actores mediante figuras humanas.

Un actor representa un rol desempeñado por algún usuario, dispositivo o sistema externo que interactúa con el sistema en estudio. Los casos de uso ayudan a identificar y modelar los requisitos funcionales del sistema por este motivo son ampliamente utilizados en las primeras fases del proceso de desarrollo de software. Los casos de uso pueden organizarse especificando relaciones entre ellos como: especializaciones, inclusiones y extensiones.

La **especialización** o relación de generalización en los casos de uso tiene el mismo significado y funcionalidad que en las clases. Dado un caso de uso base, puede tenerse otro especializado (hijo) que hereda el comportamiento del primero. El caso de uso especializado puede añadir funciones o redefinir las que ha heredado. La representación de la generalización entre casos de uso se hace de igual manera que en las clases, mediante una línea continua terminada en punta de flecha vacía que se dirige desde el caso de uso especializado hasta el caso de uso general.

Las relaciones de **inclusión** se utilizan para evitar describir el mismo flujo de eventos repetidas veces, poniendo el comportamiento común en un caso de uso aparte. La inclusión consiste en delegar funciones.

Se identifica responsabilidades del sistema que figuren en más de un caso de uso, se elabora un caso de uso para dichas funciones y luego se incluye en otros casos de uso que requieran desarrollar dichas funciones. Para indicar una relación de inclusión entre casos de uso se utiliza una dependencia con el estereotipo **include**.

Una relación de **extensión** indica que el comportamiento de un caso de uso base puede extenderse con el comportamiento de otro caso de uso que extiende al primero. Este tipo de relaciones se representan como una dependencia estereotipada con la palabra **extend**.

Diagrama de secuencia: Es un diagrama para mostrar una interacción entre un conjunto de objetos indicando el orden en que se envían y reciben los mensajes
.

Para elaborar el diagrama de secuencia se colocan en la parte superior, siguiendo el eje X, los objetos que participan en la interacción, teniendo en cuenta de colocar a la izquierda el objeto que inicia la interacción y los objetos subordinados hacia la derecha. Debajo de cada objeto y siguiendo el eje Y, se traza la línea de vida y el foco de control.

- **La línea de vida.** Es una línea discontinua vertical, trazada debajo de un objeto e indica la existencia de éste en el tiempo. Algunos objetos existen durante toda la interacción, en cuyo caso el objeto se colocará en la parte suprior del diagrama y la línea de vida se trazará de arriba hacia abajo hasta el fin del diagrama. Algunos otros objetos serán creados y destruidos durante la interacción, en este caso el objeto se colocará a la altura del mensaje que lo crea y su línea de vida se extenderá a partir de esa posición.

- **El foco de control.** Es un rectángulo delgado que se traza de forma vertical sobre la línea de vida para indicar el tiempo que el objeto está ejecutando una acción, ya sea que ejecute la acción con sus propios métodos o con métodos subordinados, es decir, invocando métodos en otros objetos. La parte superior del foco de control indica el comienzo de la acción y debe estar a la altura del primer mensaje, mientras que la parte inferior corresponde al fin de la acción y puede marcarse con un mensaje de retorno.

Diagramas de Colaboración. Representan la organización de los objetos que participan en una interacción. Estos diagramas incluyen los mensajes que se envían entre los objetos, pero no se enfocan en el orden temporal de éstos. Un diagrama de colaboración se construye con los objetos que participan en la interacción y los enlaces que se dan entre ellos. Los mensajes que se envían entre objetos se colocan como adornos de los enlaces, utilizando un número para representar el orden en que se generan los mensajes.

Los enlaces entre los objetos corresponden a las relaciones entre clases en el diagrama de clases y suele ser suficiente para que los objetos puedan enviarse mensajes entre sí. Se dice que el enlace es el camino por donde se envían los mensajes.

6. ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE LA HERRAMIENTA POLARIS

En el desarrollo de este proyecto se diseñó e implementó una herramienta de minería de uso para la web que brinde apoyo en el análisis del tráfico de un sitio web y soporte la toma de decisiones de los usuarios. En Polaris se incluyen etapas del proceso de descubrimiento de conocimiento KDD con la conexión, preprocesamiento, minería de datos y visualización de resultados. Para ello se realizó el estudio de técnicas y algoritmos de minería de datos, incluyendo, los nuevos algoritmos para las tareas de asociación y clasificación propuestos por Timaran [38,39,40,41].

Para lograr a cabo este cometido fue necesario el estudio de varias herramientas para el análisis del tráfico web existentes en el mercado, para determinar debilidades y fortalezas de las mismas y de esta manera poder desarrollar una herramienta en la que se integraran todas las fortalezas vistas y se eliminaran las dificultades encontradas.

Los algoritmos implementados fueron A priori, FPGrowth y Equipasso para la tarea de asociación, C4.5 y Mate para la tarea de clasificación; así mismo se implemento el algoritmo minería de uso HPG.

El desarrollo de la herramienta de código abierto bajo software libre y con metodología orientada a objetos hace posible la reutilización de código y permite acoplar funcionalidades con las que da la posibilidad de incluir nuevas características a la herramienta contribuyendo al mejoramiento de la misma.

La interfaz gráfica de la misma permite una interacción directa entre el usuario y la herramienta, facilitando la visualización de resultados a través de tablas, árboles de decisión, gramáticas y gráficos estadísticos.

Para la construcción de la herramienta se desarrollaron tres módulos de empleados, el módulo de utilidades para realizar la conexión de datos y que además contiene todas las clases y bibliotecas que son utilizadas en toda la aplicación, el módulo del kernel que incluye el preprocesamiento de los datos que

van a ser analizados y los algoritmos utilizados en el proceso KDD, y el módulo de la interfaz gráfica que permite la interacción amigable entre el usuario y la herramienta y facilita la visualización de los resultados y reglas obtenidos a través de tablas, gramáticas, reportes textuales, reportes estadísticos y gráficas.

A continuación puede apreciarse la descripción formal del desarrollo del proyecto Polaris.

6.1. ANÁLISIS UML

6.1.1. Diagramas de Paquetes. El diseño de los diagramas de paquetes de la herramienta Polaris se muestran en las figura 2.26, figura 2.27 y figura 2.28.

Figura 2.26. Paquete Principal

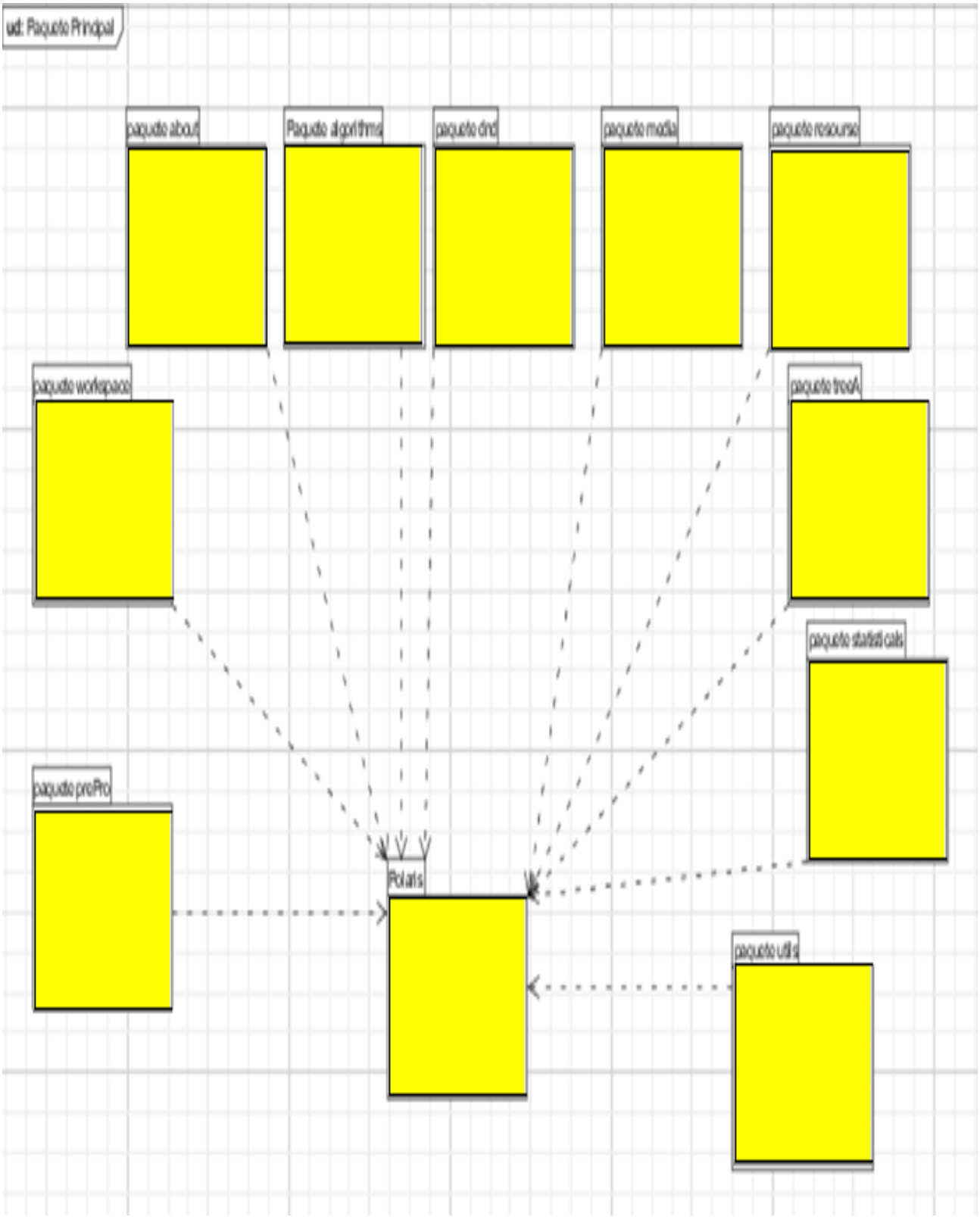

ud: Paquete Principal
paquete about
Paquete algorithms
paquete dnd
paquete media
paquete resources
paquete workspaces
paquete treeA
paquete statistics
paquete preProc
Polaris
paquete utils

Figura 2.27. Paquete Algorithms

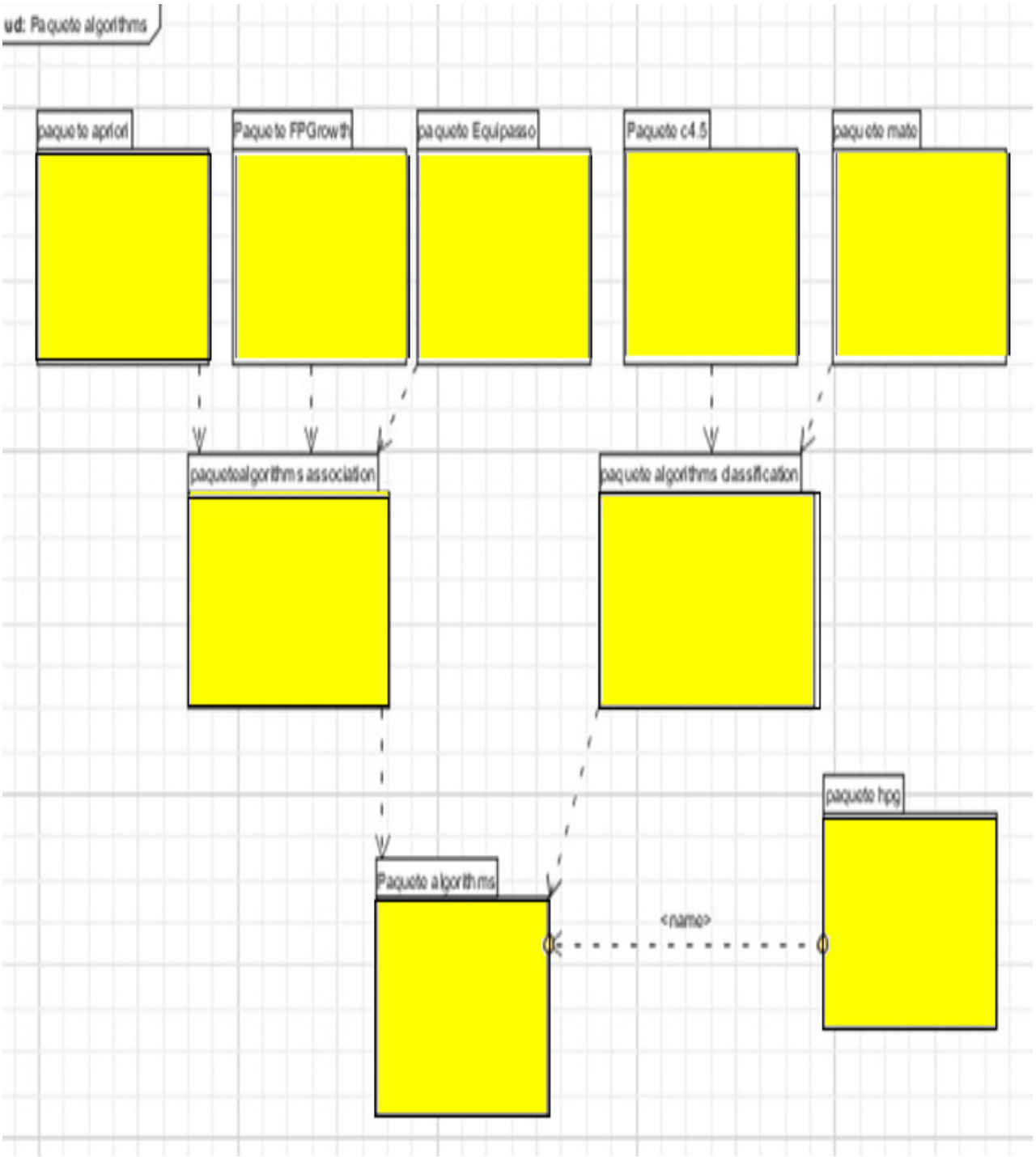
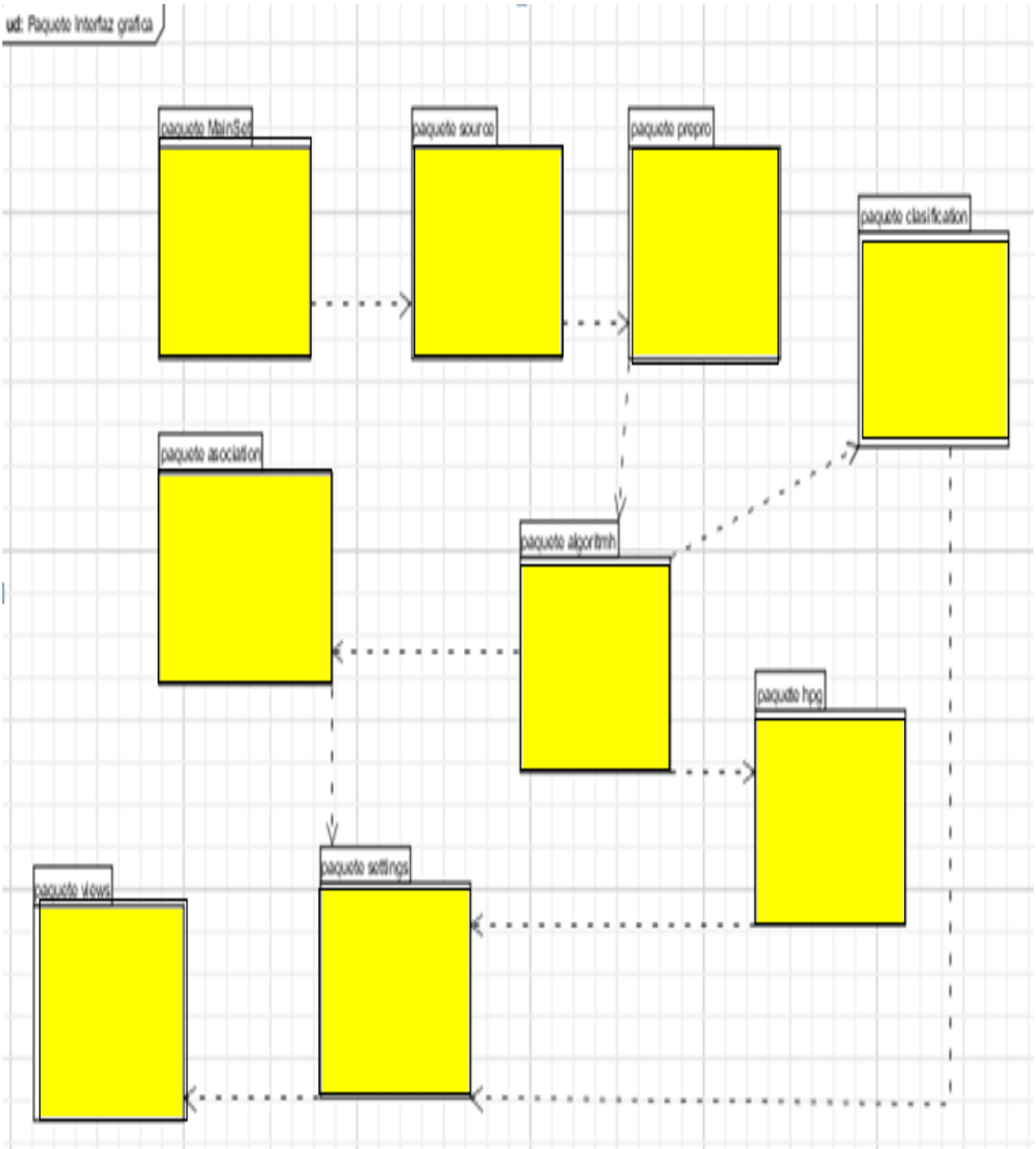


Figura 2.28. Paquete Interfaz Gráfica



6.1.2. Diagramas de Clases

El diseño de los diagramas de clases de la herramienta Polaris se muestran en la figura 2.29 hasta la figura 2.38.

Figura 2.29. Abrir Fuente

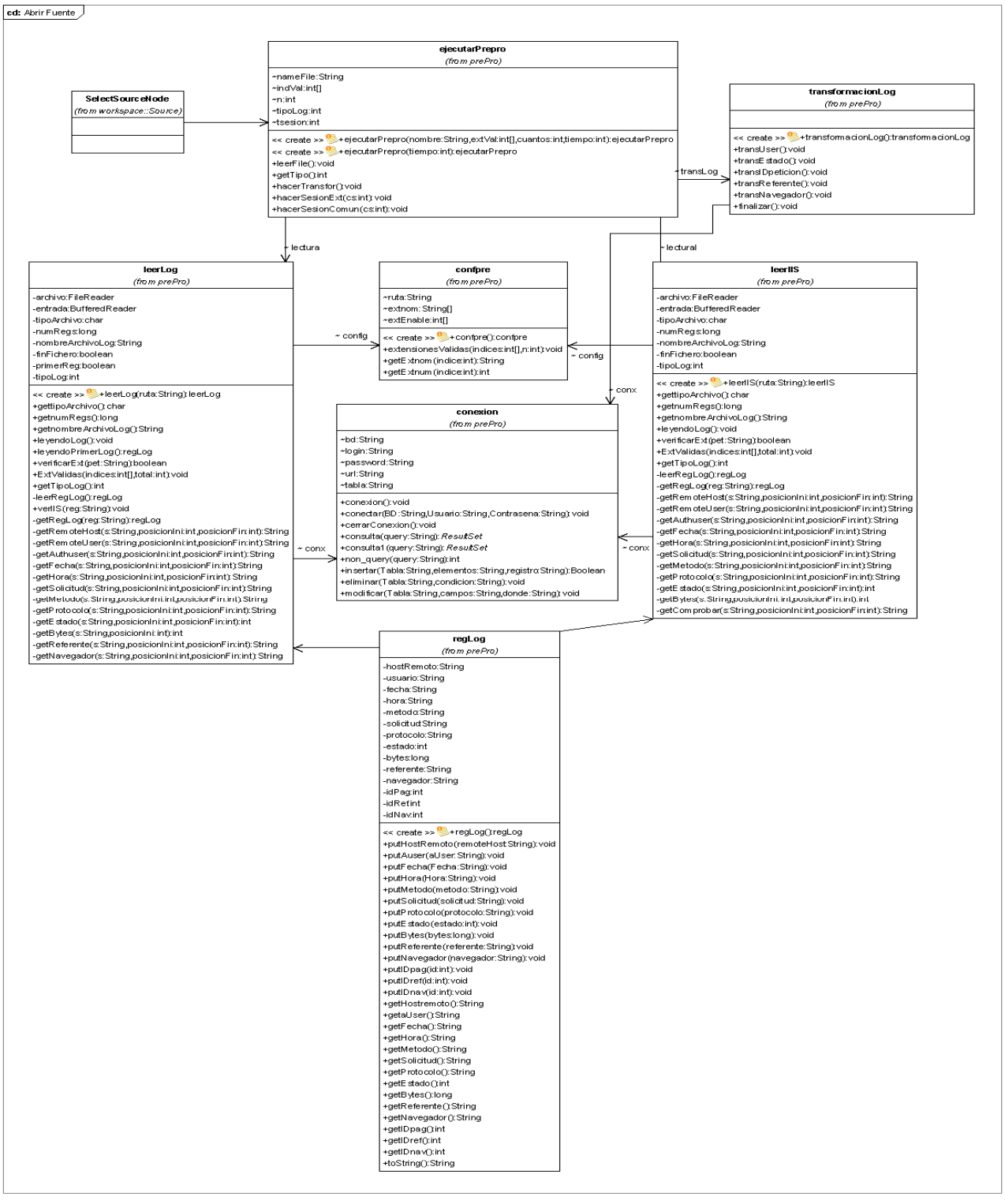

cd: Abrir Fuente
SelectSourceNode
(from workspace::Source)
ejecutarPrepro
(from prePro)
~nameFile:String
~indVal:int[]
~n:int
~tipoLog:int
~tsesion:int
<< create >> +ejecutarPrepro(nombre:String,extVal:int[],cuantos:int,tiempo:int):ejecutarPrepro
<< create >> +ejecutarPrepro(tiempo:int):ejecutarPrepro
+leerFile():void
+getTipo():int
+hacerTransfor():void
+hacerSesionExt(cs:int):void
+hacerSesionComun(cs:int):void
~transLog
transformacionLog
(from prePro)
<< create >> +transformacionLog():transformacionLog
+transUser():void
+transEstado():void
+transIDpeticion():void
+transReferente():void
+transNavegador():void
+finalizar():void
~lectura
~lectural
leerLog
(from prePro)
-archivo:FileReader
-entrada:BufferedReader
-tipoArchivo:char
-numRegs:long
-nombreArchivoLog:String
-finFichero:boolean
-primerReg:boolean
-tipoLog:int
<< create >> +leerLog(ruta:String):leerLog
+gettipoArchivo():char
+getnumRegs():long
+getnombreArchivoLog():String
+leyendoLog():void
+leyendoPrimerLog():regLog
+verificarExt(pet:String):boolean
+ExtValidas(indices:int[],total:int):void
+getTipoLog():int
-leerRegLog():regLog
+verIIS(reg:String):void
-getRegLog(reg:String):regLog
-getRemoteHost(s:String,posicionIni:int,posicionFin:int):String
-getRemoteUser(s:String,posicionIni:int,posicionFin:int):String
-getAuthuser(s:String,posicionIni:int,posicionFin:int):String
-getFecha(s:String,posicionIni:int,posicionFin:int):String
-getHora(s:String,posicionIni:int,posicionFin:int):String
-getSolicitud(s:String,posicionIni:int,posicionFin:int):String
-getMetodo(s:String,posicionIni:int,posicionFin:int):String
-getProtocolo(s:String,posicionIni:int,posicionFin:int):String
-getEstado(s:String,posicionIni:int,posicionFin:int):int
-getBytes(s:String,posicionIni:int):int
-getReferente(s:String,posicionIni:int,posicionFin:int):String
-getNavegador(s:String,posicionIni:int,posicionFin:int):String
~config
confpre
(from prePro)
~ruta:String
~extnom:String[]
~extEnable:int[]
<< create >> +confpre():confpre
+extensionesValidas(indices:int[],n:int):void
+getExtnom(indice:int):String
+getExtnum(indice:int):int
~config
~conx
conexion
(from prePro)
~bd:String
~login:String
~password:String
~url:String
~tabla:String
+conexion():void
+conectar(BD:String,Usuario:String,Contrasena:String):void
+cerrarConexion():void
+consulta(query:String):ResultSet
+consulta1(query:String):ResultSet
+non_query(query:String):int
+insertar(Tabla:String,elementos:String,registro:String):Boolean
+eliminar(Tabla:String,condicion:String):void
+modificar(Tabla:String,campos:String,donde:String):void
~conx
~conx
leerIIS
(from prePro)
-archivo:FileReader
-entrada:BufferedReader
-tipoArchivo:char
-numRegs:long
-nombreArchivoLog:String
-finFichero:boolean
-tipoLog:int
<< create >> +leerIIS(ruta:String):leerIIS
+gettipoArchivo():char
+getnumRegs():long
+getnombreArchivoLog():String
+leyendoLog():void
+verificarExt(pet:String):boolean
+ExtValidas(indices:int[],total:int):void
+getTipoLog():int
-leerRegLog():regLog
-getRegLog(reg:String):regLog
-getRemoteHost(s:String,posicionIni:int,posicionFin:int):String
-getRemoteUser(s:String,posicionIni:int,posicionFin:int):String
-getAuthuser(s:String,posicionIni:int,posicionFin:int):String
-getFecha(s:String,posicionIni:int,posicionFin:int):String
-getHora(s:String,posicionIni:int,posicionFin:int):String
-getSolicitud(s:String,posicionIni:int,posicionFin:int):String
-getMetodo(s:String,posicionIni:int,posicionFin:int):String
-getProtocolo(s:String,posicionIni:int,posicionFin:int):String
-getEstado(s:String,posicionIni:int,posicionFin:int):int
-getBytes(s:String,posicionIni:int,posicionFin:int):int
-getComprobar(s:String,posicionIni:int,posicionFin:int):String
regLog
(from prePro)
-hostRemoto:String
-usuario:String
-fecha:String
-hora:String
-metodo:String
-solicitud:String
-protocolo:String
-estado:int
-bytes:long
-referente:String
-navegador:String
-idPag:int
-idRef:int
-idNav:int
<< create >> +regLog():regLog
+putHostRemoto(remoteHost:String):void
+putAuser(aUser:String):void
+putFecha(Fecha:String):void
+putHora(Hora:String):void
+putMetodo(metodo:String):void
+putSolicitud(solicitud:String):void
+putProtocolo(protocolo:String):void
+putEstado(estado:int):void
+putBytes(bytes:long):void
+putReferente(referente:String):void
+putNavegador(navegador:String):void
+putIDpag(id:int):void
+putIDref(id:int):void
+putIDnav(id:int):void
+getHostremoto():String
+getaUser():String
+getFecha():String
+getHora():String
+getMetodo():String
+getSolicitud():String
+getProtocolo():String
+getEstado():int
+getBytes():long
+getReferente():String
+getNavegador():String
+getIDpag():int
+getIDref():int
+getIDnav():int
+toString():String

Figura2.30. Abrir Interfaz

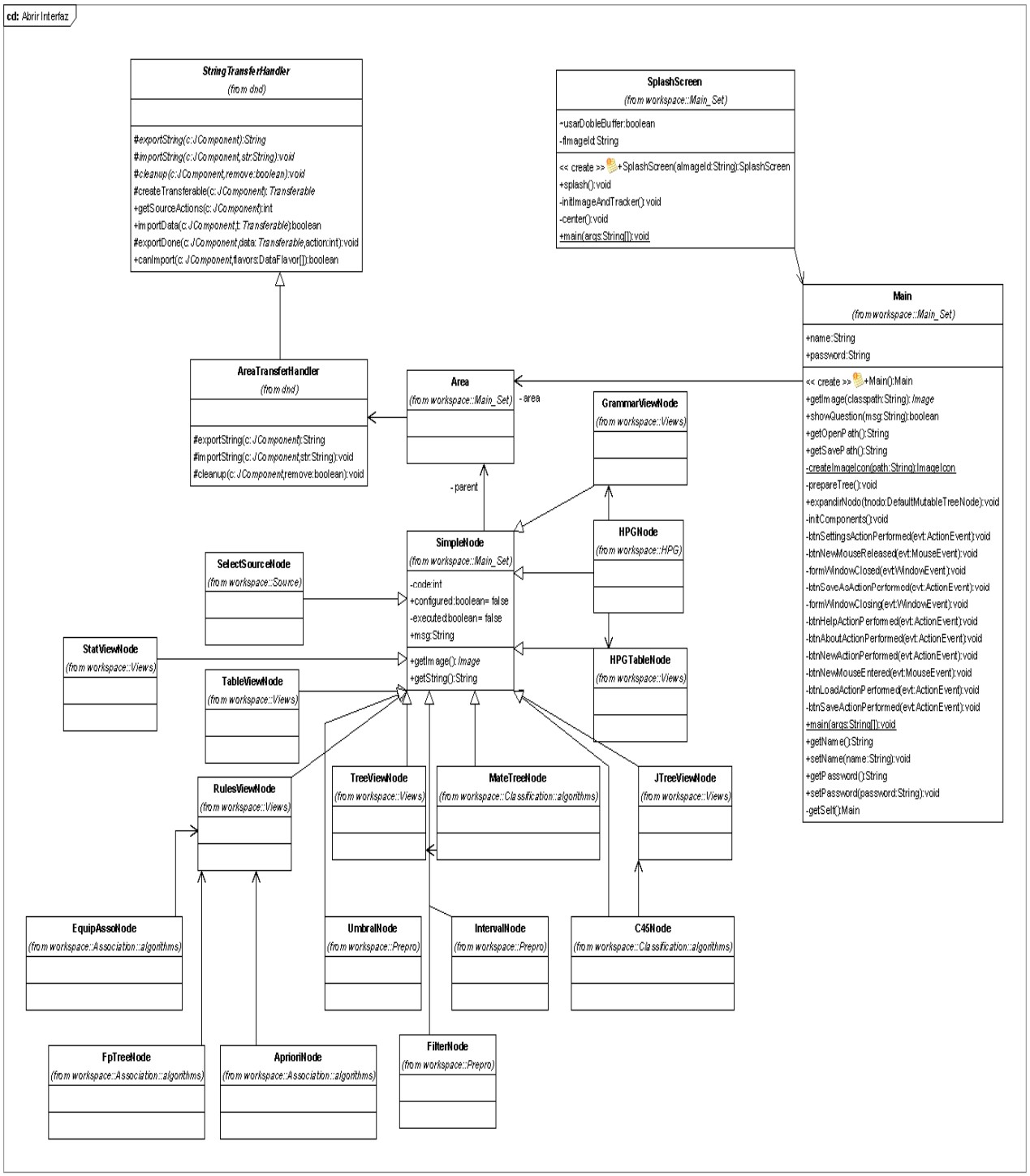


Figura 2.31. Hacer Asociación

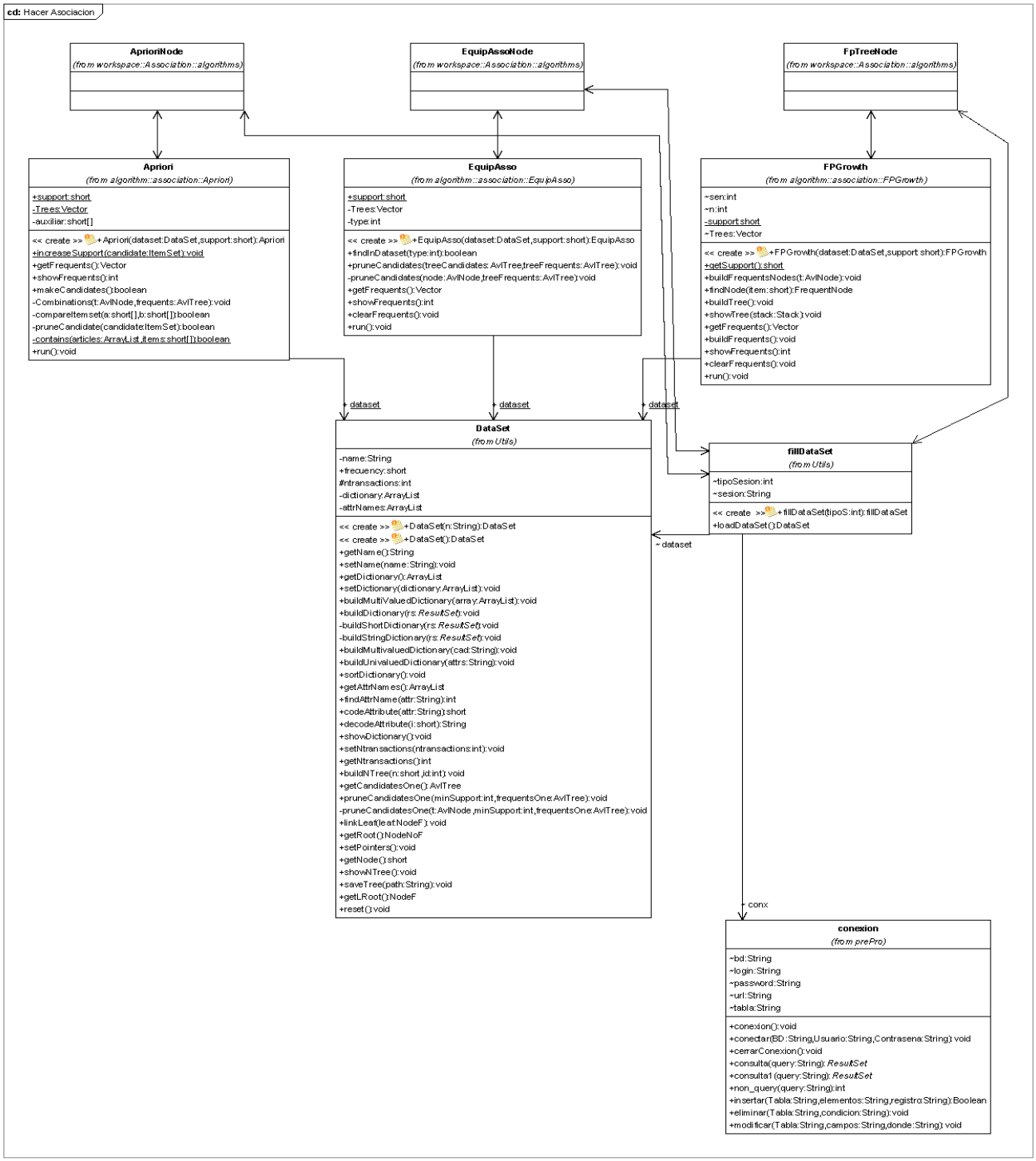


Figura 2.32. Hacer Clasificación

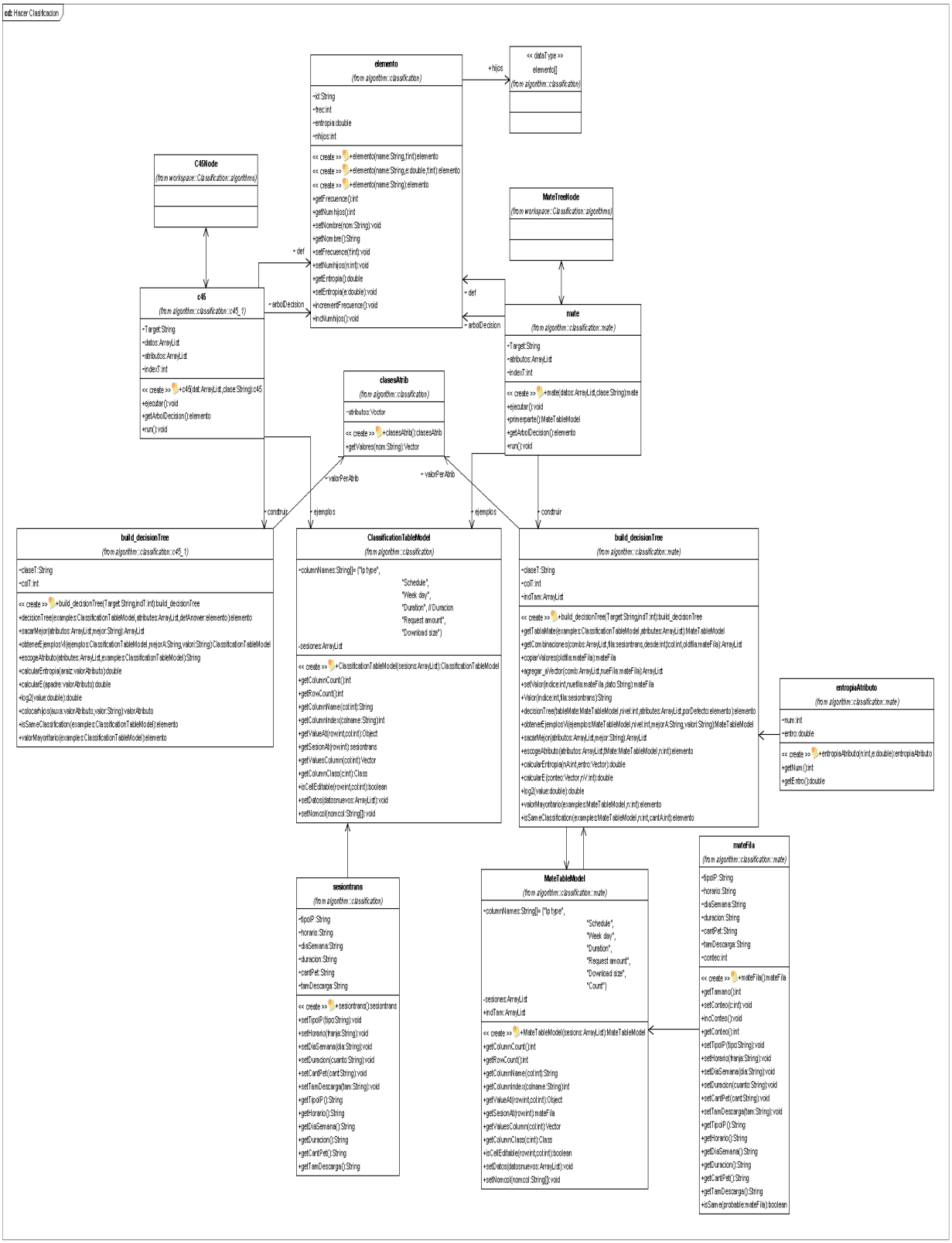


Figura 2.33. Hacer Discretizar

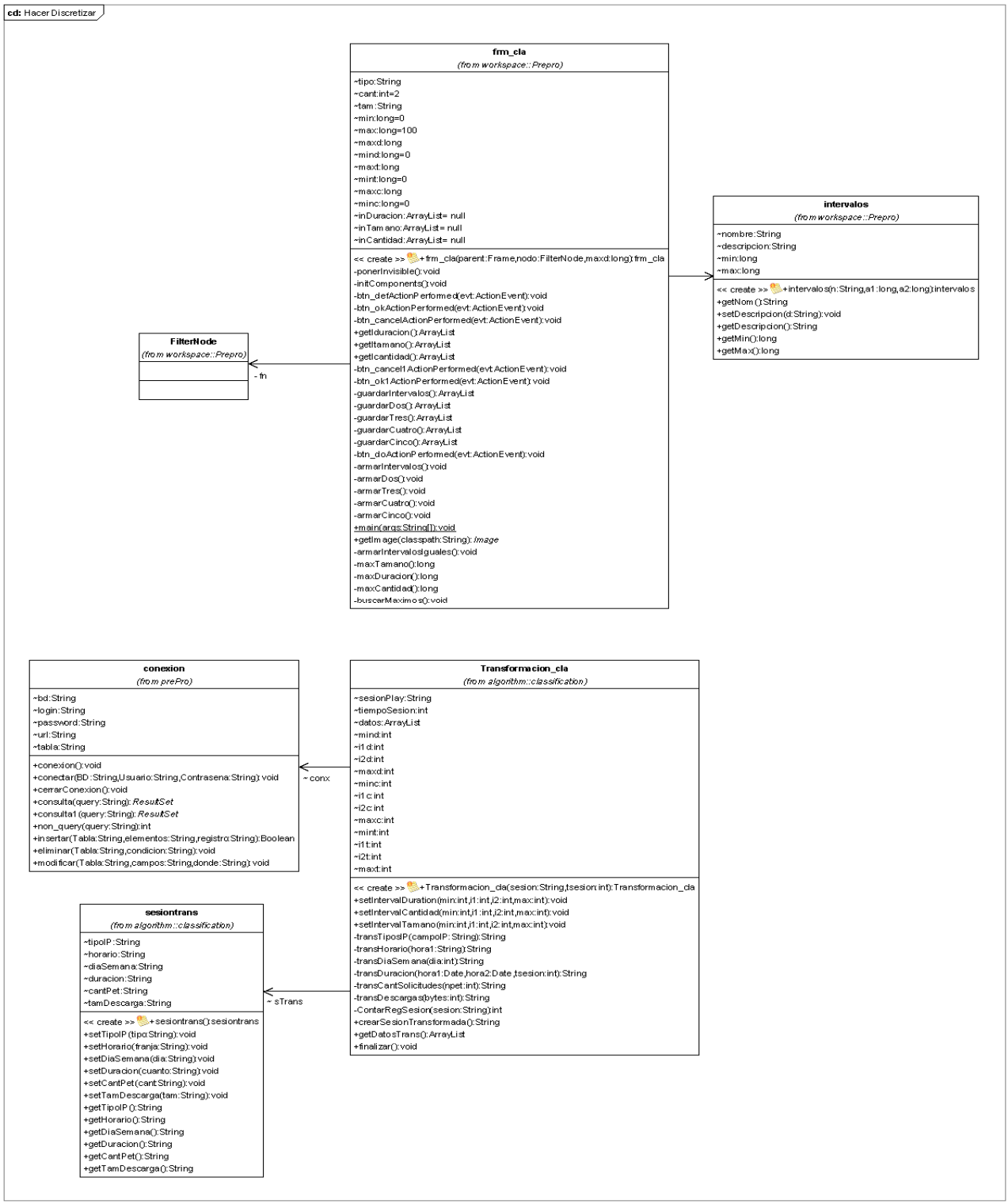


Figura 2.34. Hacer HPG

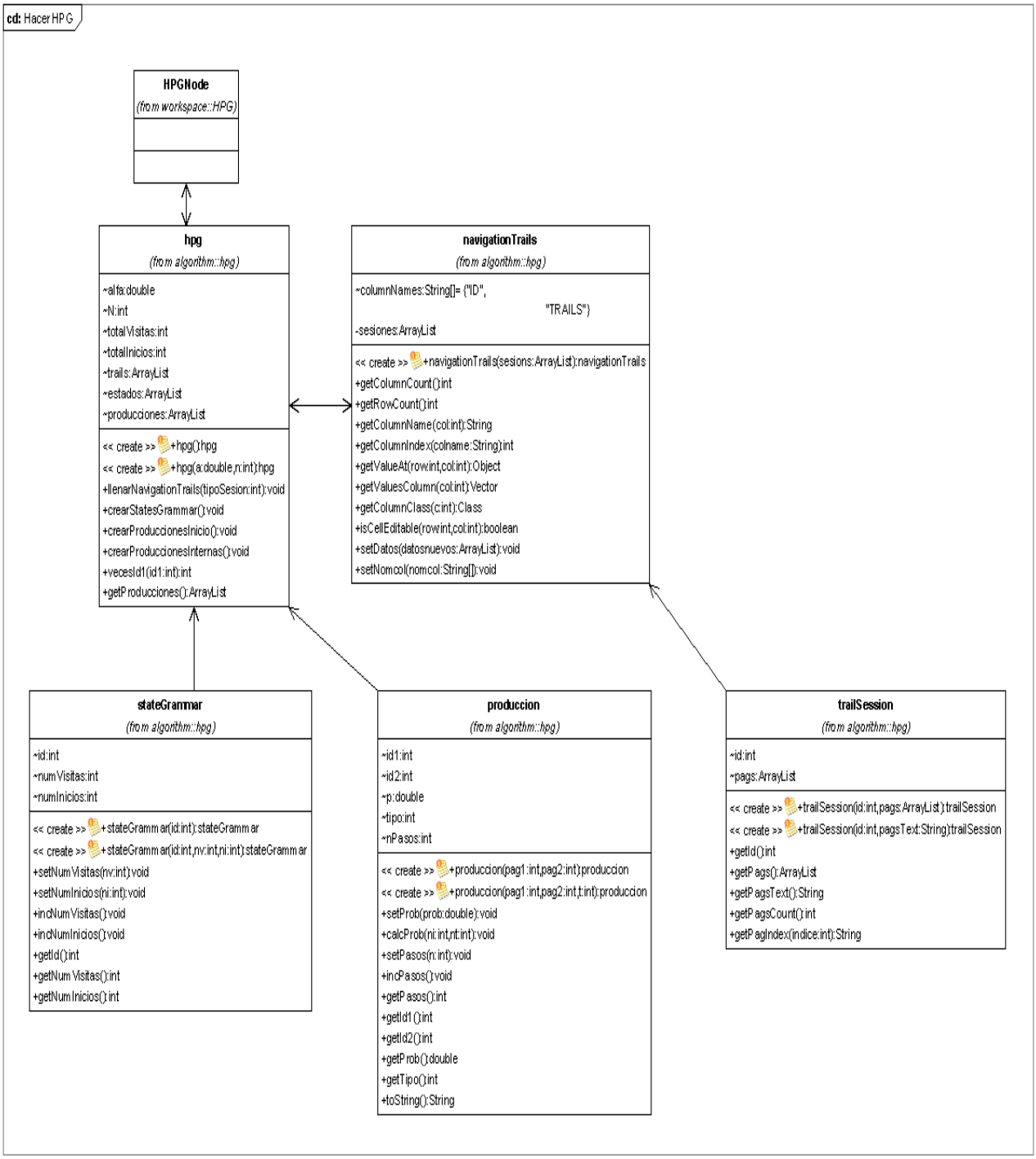


Figura 2.35. Mostrar Gramática

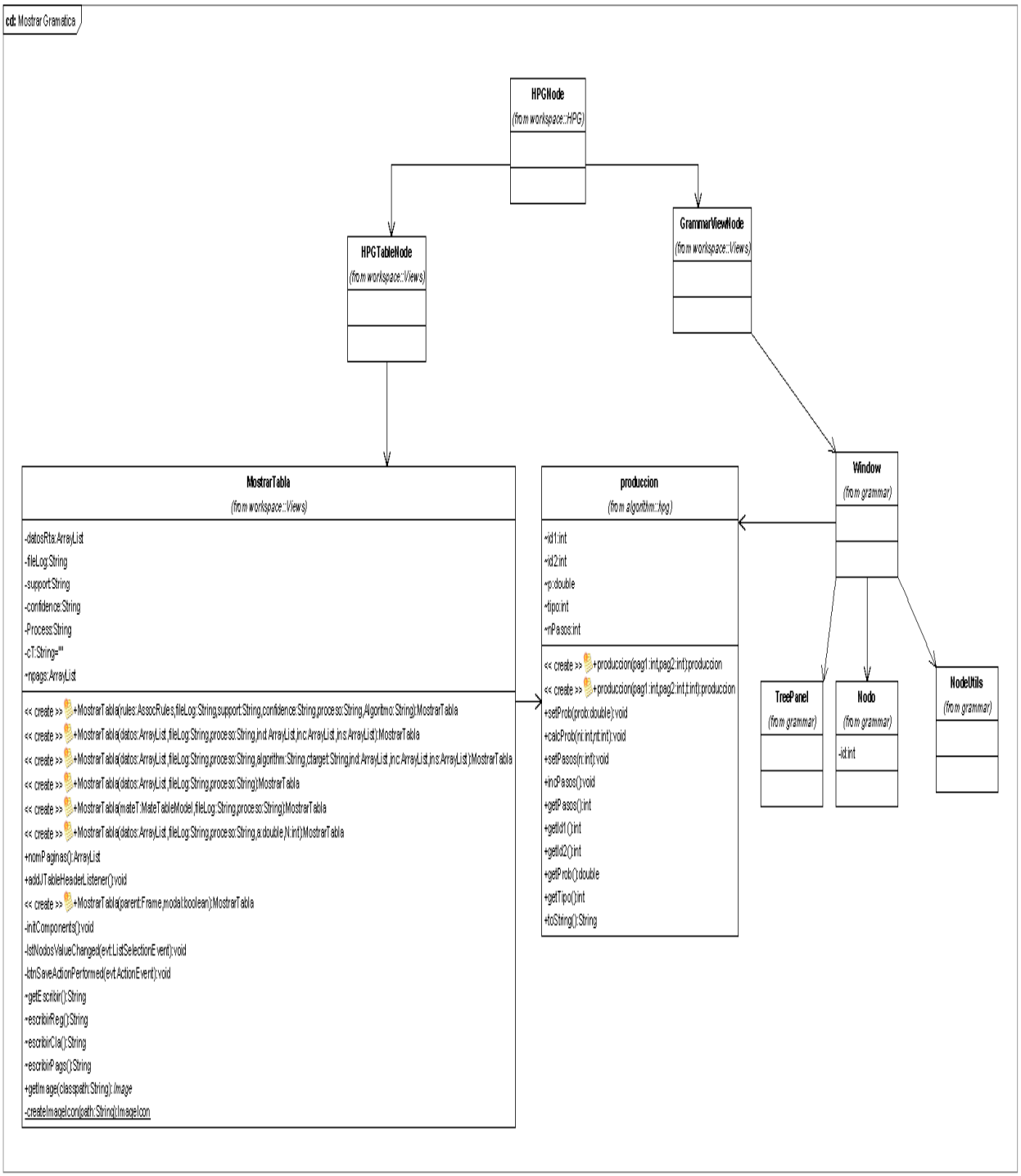


Figura 2.36. Ver Árbol de Clasificación

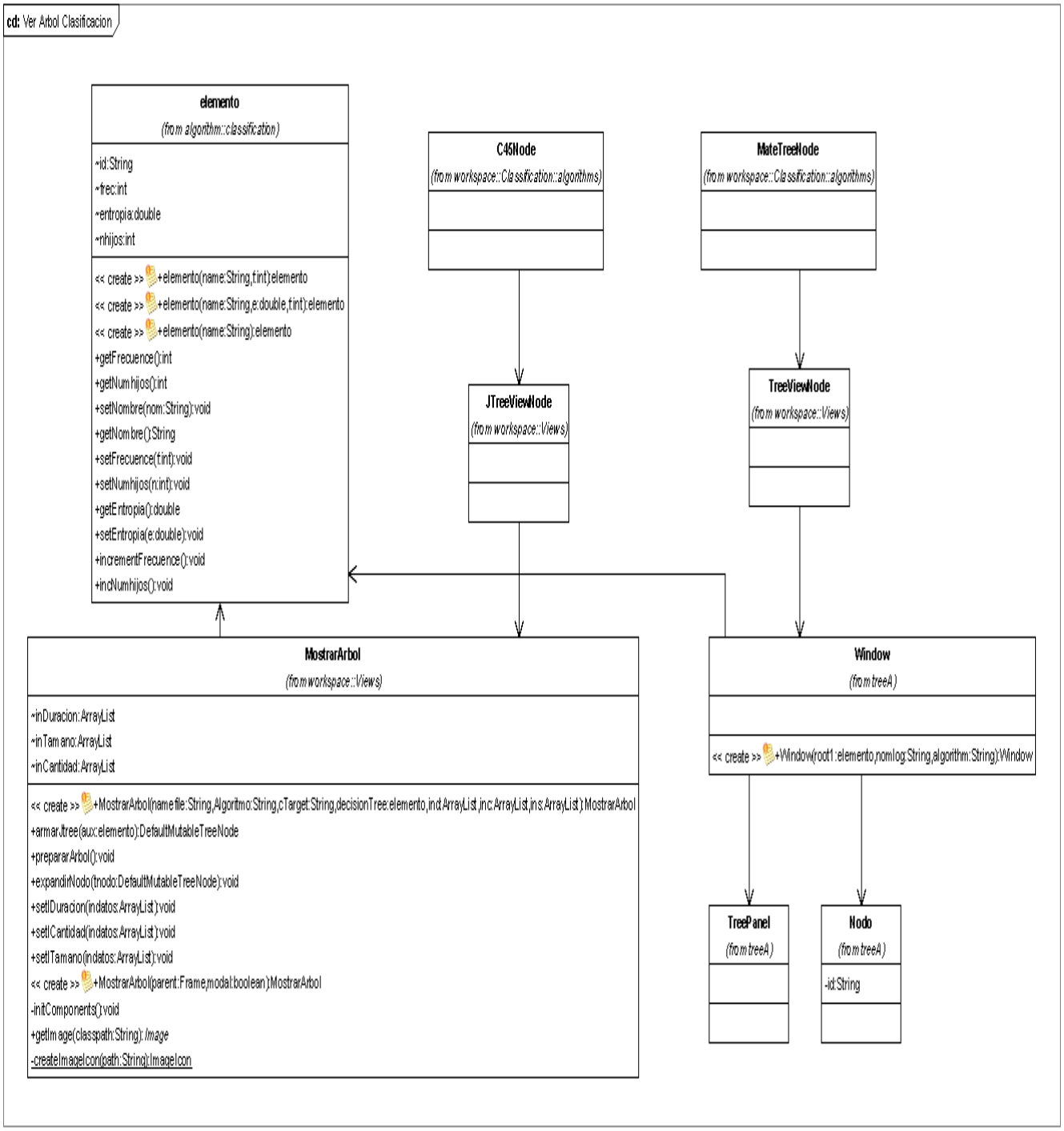


Figura 2.37. Ver Reglas de Asociación

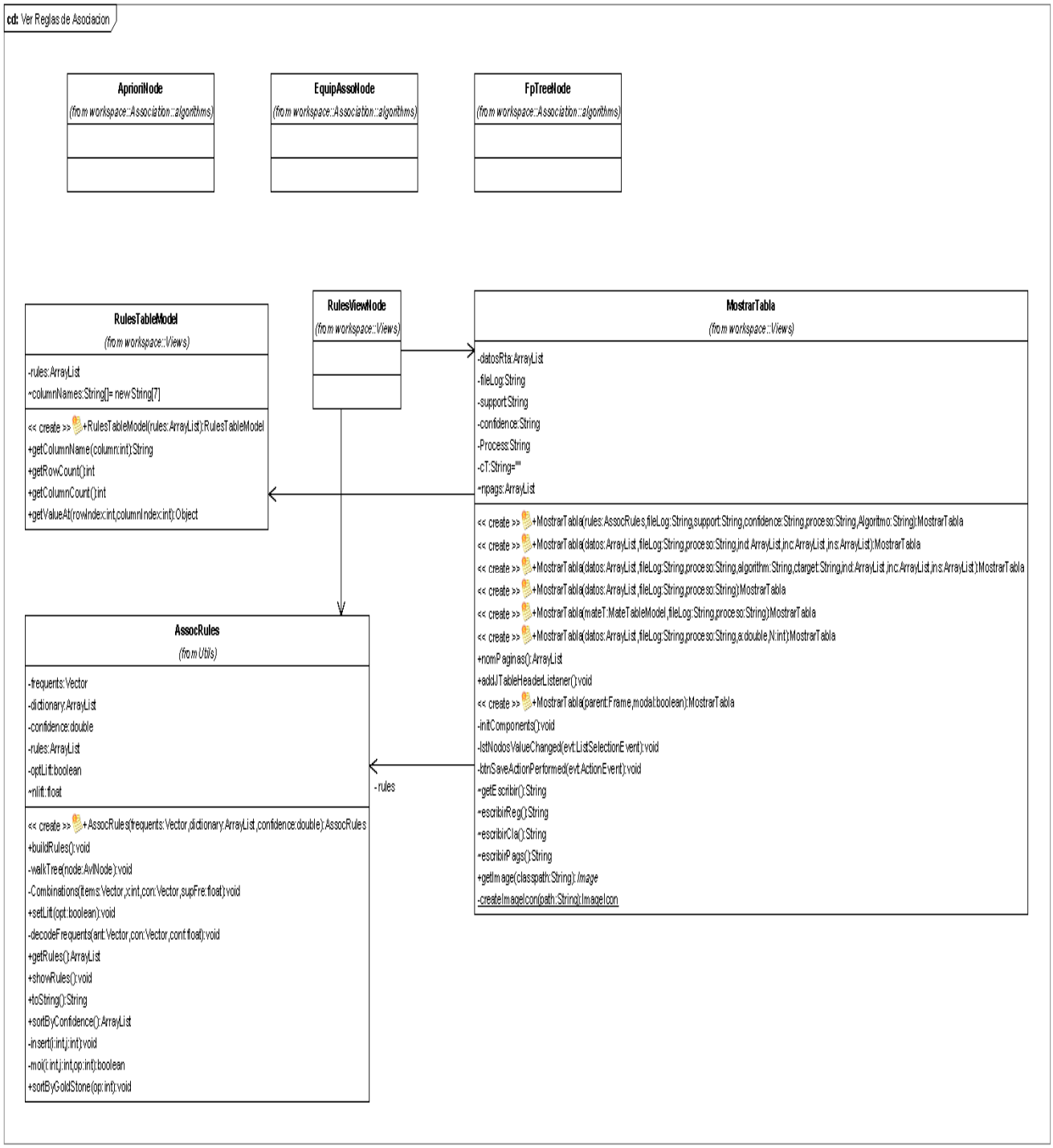
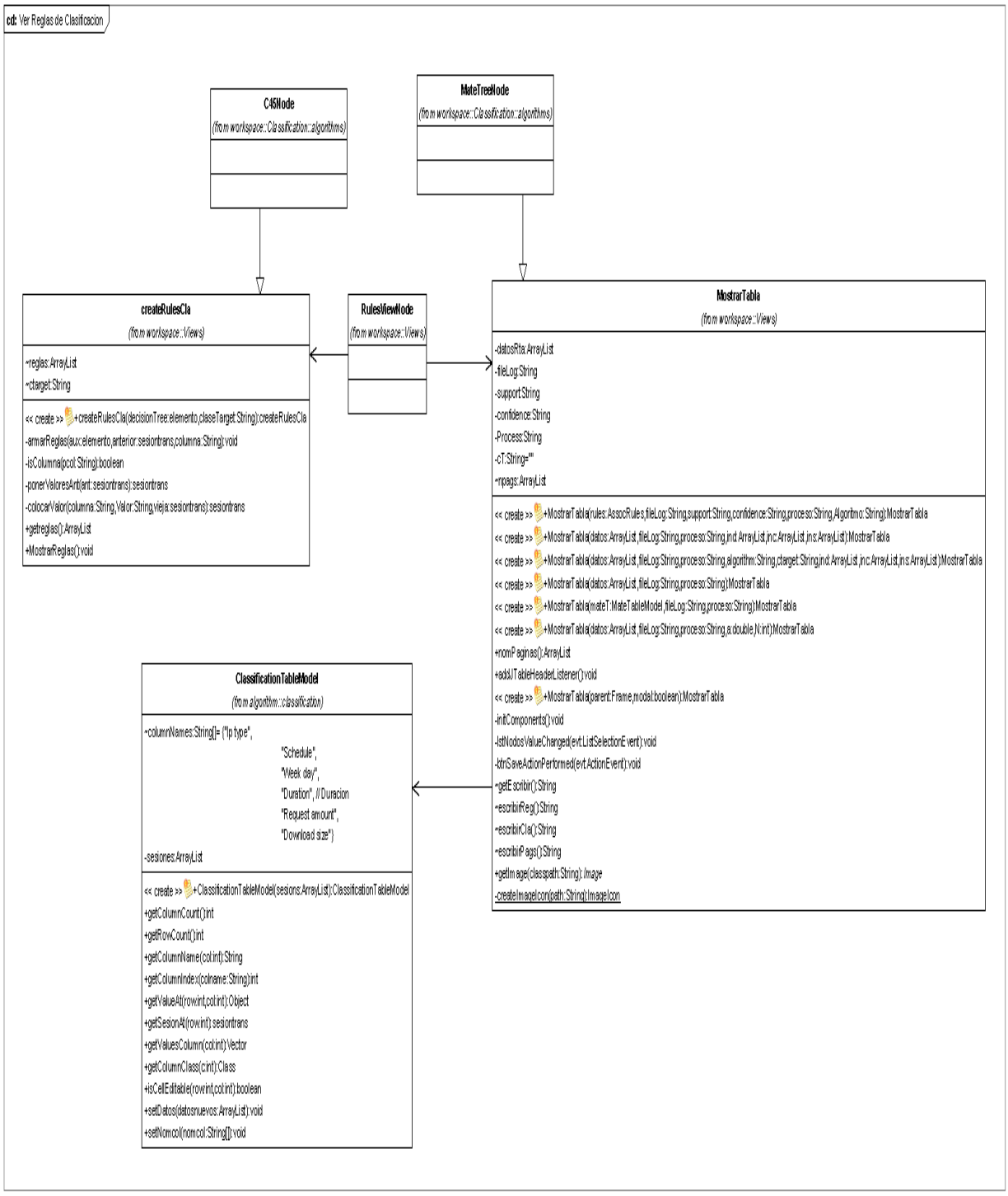


Figura 2.38. Ver Reglas de Clasificación



6.1.3. Clases Polaris

El diseño de las clases de la herramienta Polaris se muestran en la figura 2.39 hasta la figura 2.48.

Figura 2.39. Paquete about

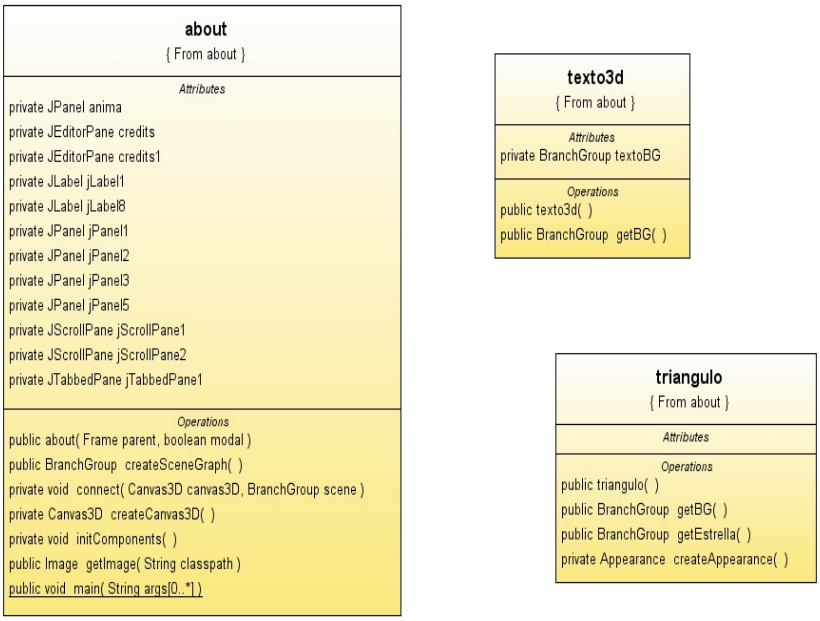
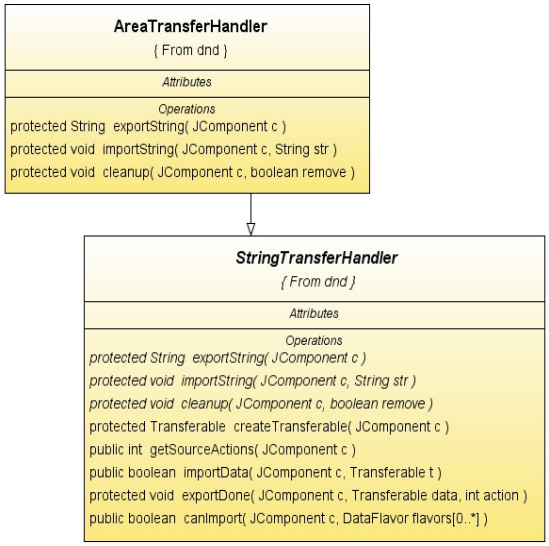
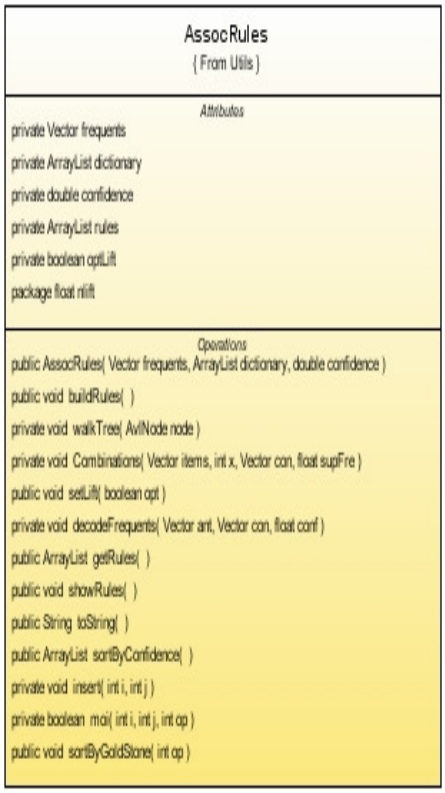


Figura 2.40. Paq




AssocRules
{ From Utils }
Attributes
private Vector frequents
private ArrayList dictionary
private double confidence
private ArrayList rules
private boolean optLift
package float nlift
Operations
public AssocRules(Vector frequents, ArrayList dictionary, double confidence)
public void buildRules( )
private void walkTree(AvlNode node)
private void Combinations(Vector items, int x, Vector con, float supFre)
public void setLift(boolean opt)
private void decodeFrequents(Vector ant, Vector con, float conf)
public ArrayList getRules( )
public void showRules( )
public String toString( )
public ArrayList sortByConfidence( )
private void insert(int i, int j)
private boolean mai(int i, int j, int op)
public void sortByGoldStone(int op)


DataSet
{ From Utils }
Attributes
private String name
public short frecuency
protected int ntransactions
private ArrayList dictionary
private ArrayList attrNames
Operations
public DataSet(String n)
public DataSet( )
public String getName( )
public void setName(String name)
public ArrayList getDictionary( )
public void setDictionary(ArrayList dictionary)
public void buildMultiValuedDictionary(ArrayList array)
public void buildDictionary(ResultSet rs)
private void buildShortDictionary(ResultSet rs)
private void buildStringDictionary(ResultSet rs)
public void buildMultivaluedDictionary(String cad)
public void buildUnivaluedDictionary(String attrs)
public void sortDictionary( )
public ArrayList getAttrNames( )
public int findAttrName(String attr)
public short codeAttribute(String attr)
public String decodeAttribute(short i)
public void showDictionary( )
public void setNtransactions(int ntransactions)
public int getNtransactions( )
public void buildNTree(short n, int id)
public AvlTree getCandidatesOne( )
public void pruneCandidatesOne(int minSuppart, AvlTree frequentsOne)
private void pruneCandidatesOne(AvlNode t, int minSuppart, AvlTree frequentsOne)
public void linkLeaf(NodeF leaf)
public NodeNoF getRoot( )
public void setPointers( )
public short getNode( )
public void showNTree( )
public void saveTree(String path)
public NodeF getLRoot( )
public void reset( )

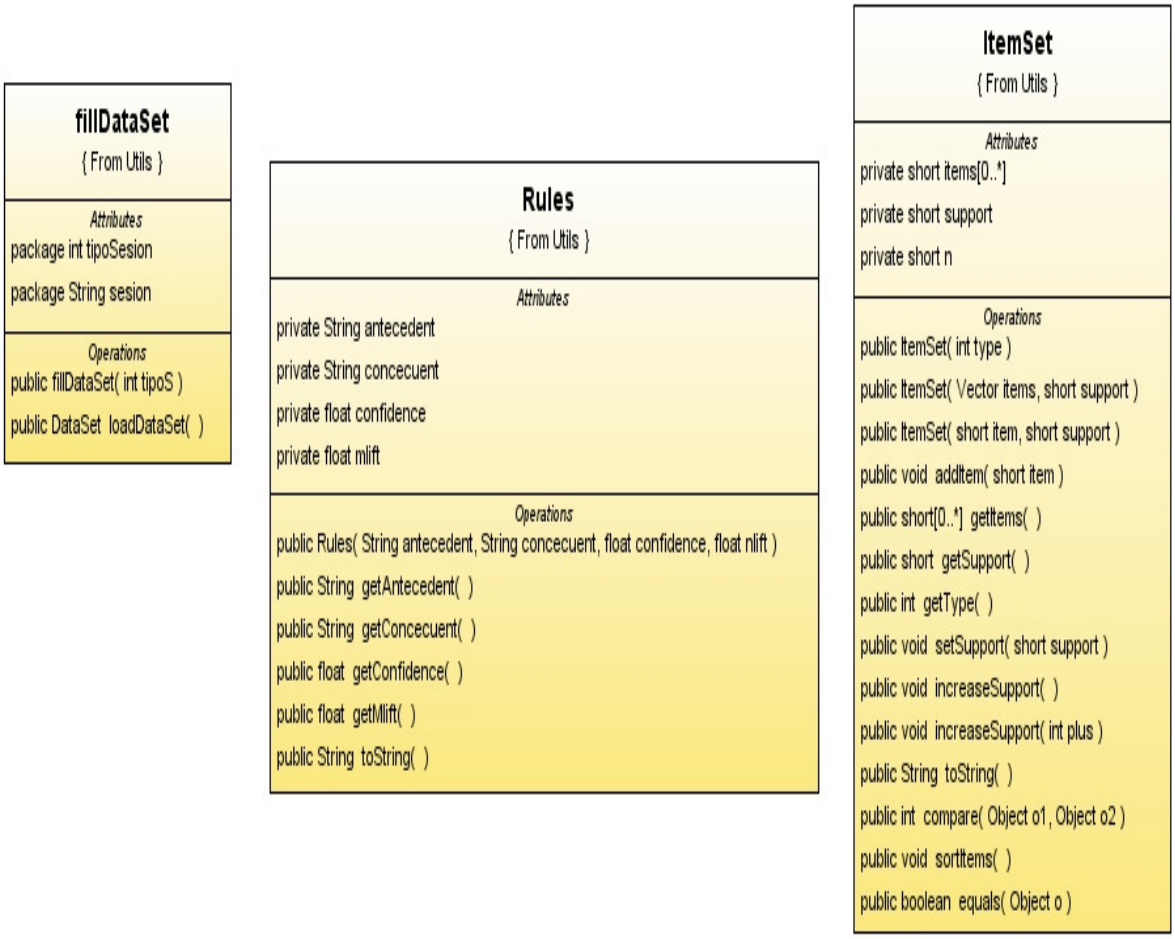

fillDataSet
{ From Utils }
Attributes
package int tipoSesion
package String sesion
Operations
public fillDataSet(int tipoS)
public DataSet loadDataSet( )
Rules
{ From Utils }
Attributes
private String antecedent
private String concecuent
private float confidence
private float mlift
Operations
public Rules(String antecedent, String concecuent, float confidence, float nlift)
public String getAntecedent( )
public String getConcecuent( )
public float getConfidence( )
public float getMlift( )
public String toString( )
ItemSet
{ From Utils }
Attributes
private short items[0..*]
private short support
private short n
Operations
public ItemSet(int type)
public ItemSet(Vector items, short support)
public ItemSet(short item, short support)
public void addItem(short item)
public short[0..*] getItems( )
public short getSupport( )
public int getType( )
public void setSupport(short support)
public void increaseSupport( )
public void increaseSupport(int plus)
public String toString( )
public int compare(Object o1, Object o2)
public void sortItems( )
public boolean equals(Object o)

Figura 2.41. Paquete algorithms

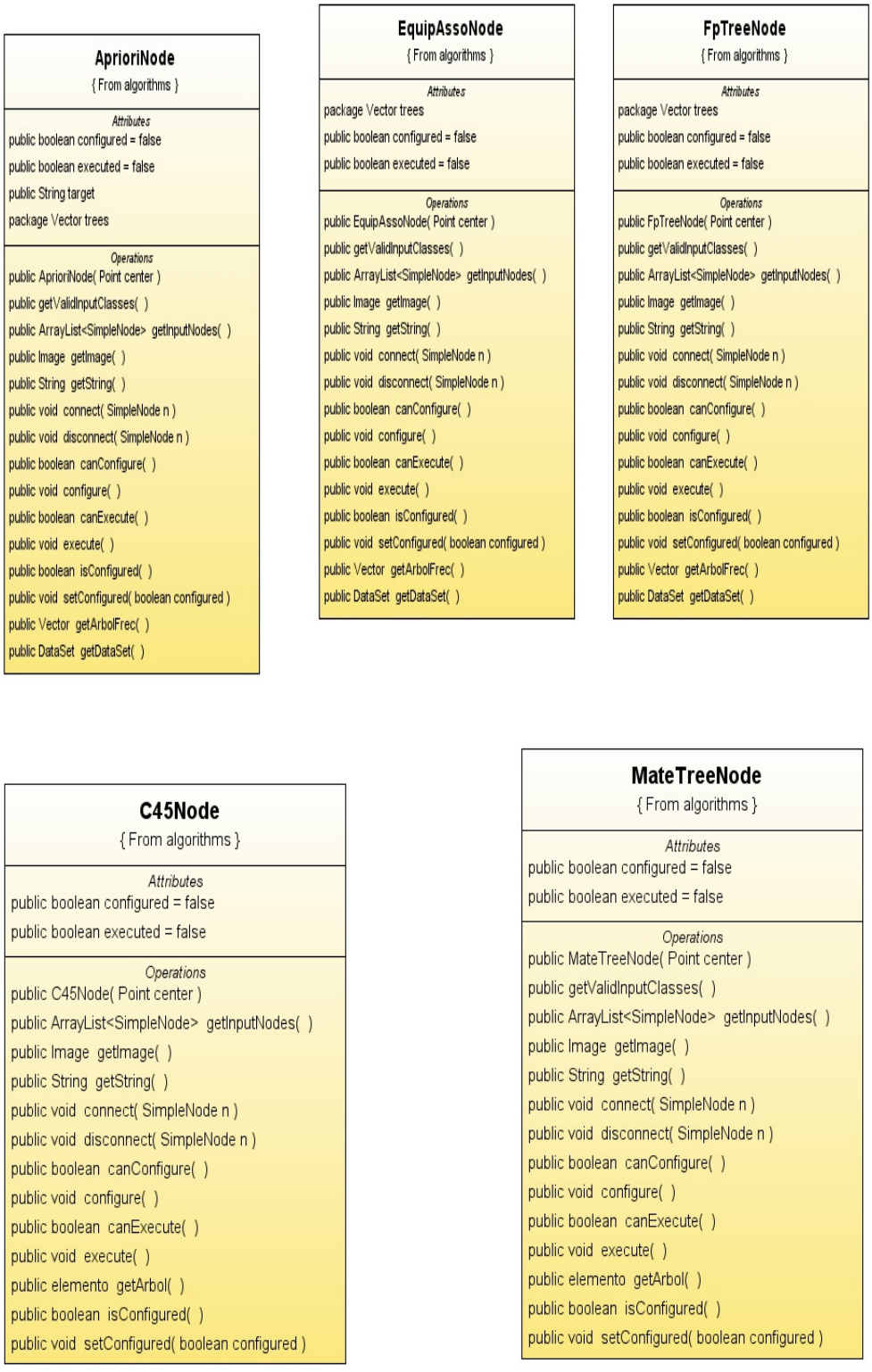
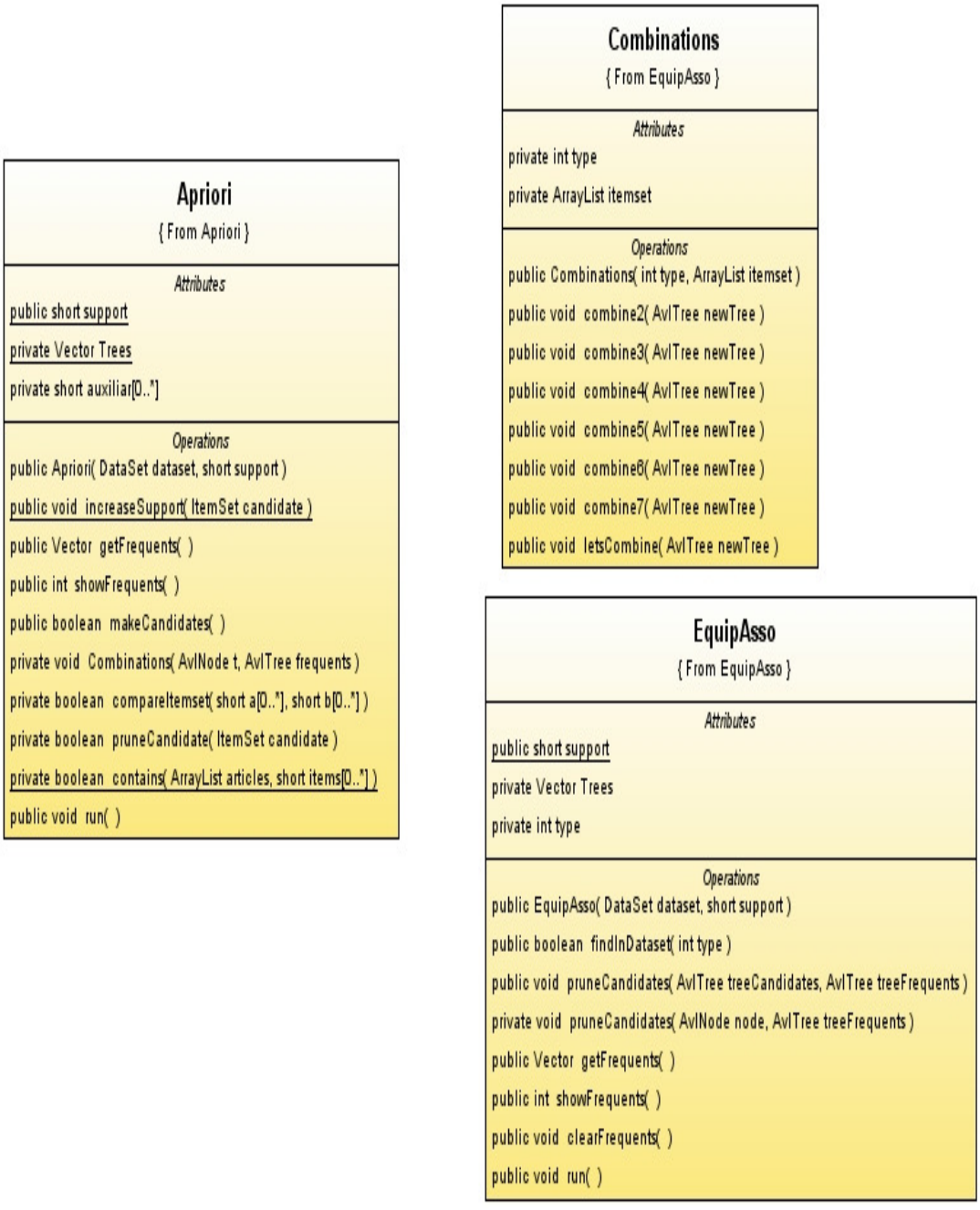
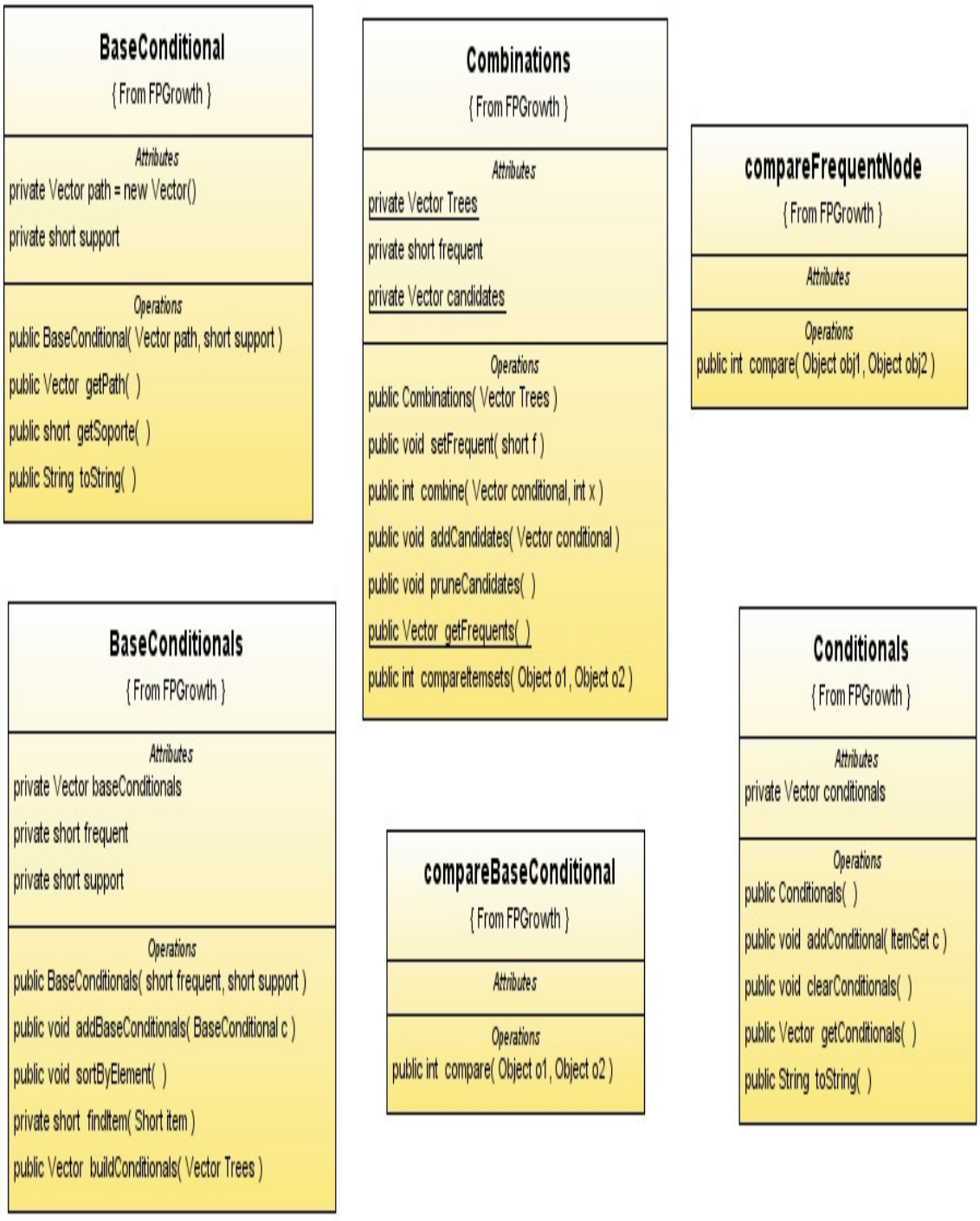


Figura 2.41.1. Paquete algorithm association




BaseConditional
{ From FPGrowth }
Attributes
private Vector path = new Vector()
private short support
Operations
public BaseConditional(Vector path, short support)
public Vector getPath( )
public short getSoporte( )
public String toString( )
Combinations
{ From FPGrowth }
Attributes
private Vector Trees
private short frequent
private Vector candidates
Operations
public Combinations(Vector Trees)
public void setFrequent(short f)
public int combine(Vector conditional, int x)
public void addCandidates(Vector conditional)
public void pruneCandidates( )
public Vector getFrequents( )
public int compareItemsets(Object o1, Object o2)
compareFrequentNode
{ From FPGrowth }
Attributes
Operations
public int compare(Object obj1, Object obj2)
BaseConditionals
{ From FPGrowth }
Attributes
private Vector baseConditionals
private short frequent
private short support
Operations
public BaseConditionals(short frequent, short support)
public void addBaseConditionals(BaseConditional c)
public void sortByElement( )
private short findItem(Short item)
public Vector buildConditionals(Vector Trees)
compareBaseConditional
{ From FPGrowth }
Attributes
Operations
public int compare(Object o1, Object o2)
Conditionals
{ From FPGrowth }
Attributes
private Vector conditionals
Operations
public Conditionals( )
public void addConditional(itemSet c)
public void clearConditionals( )
public Vector getConditionals( )
public String toString( )

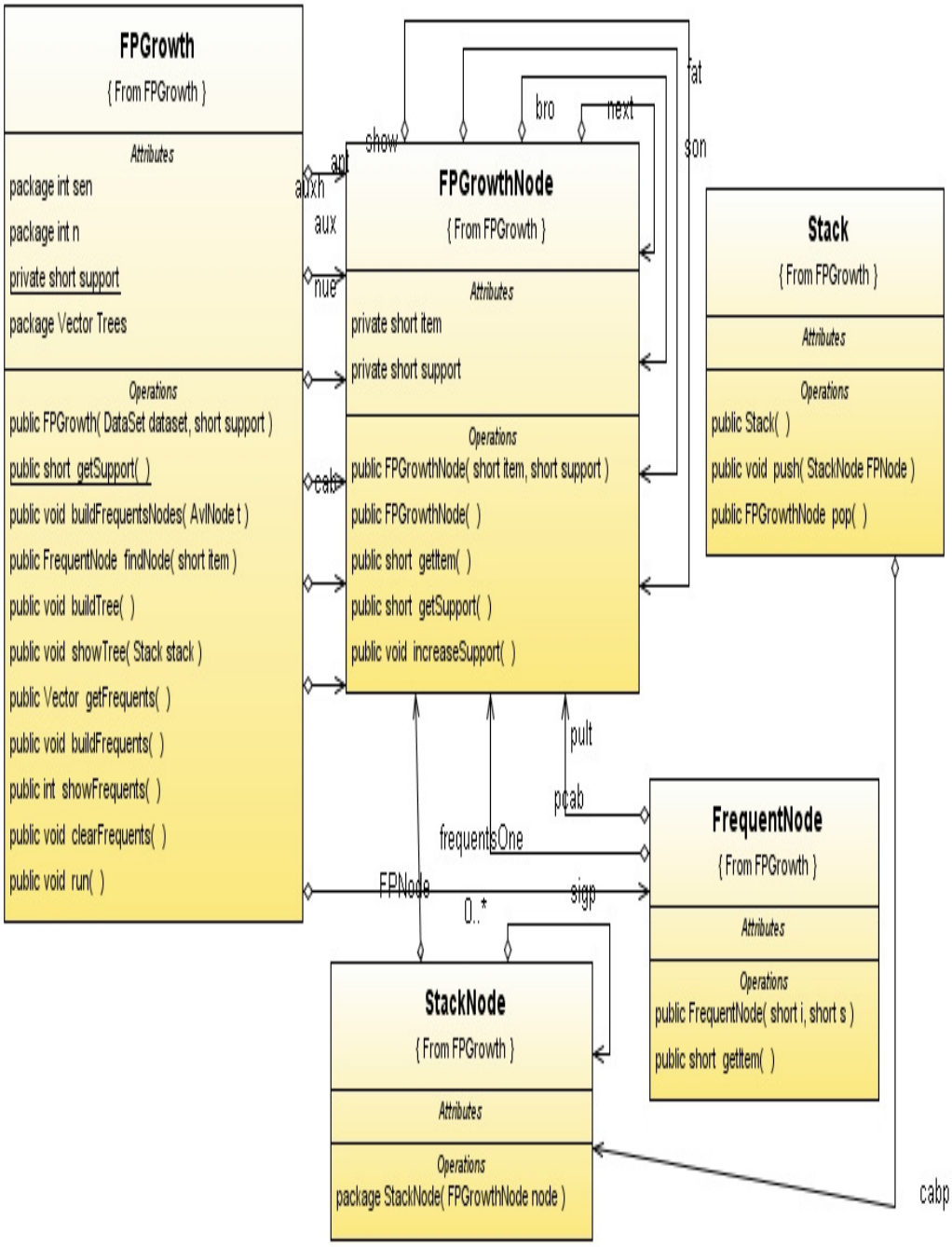
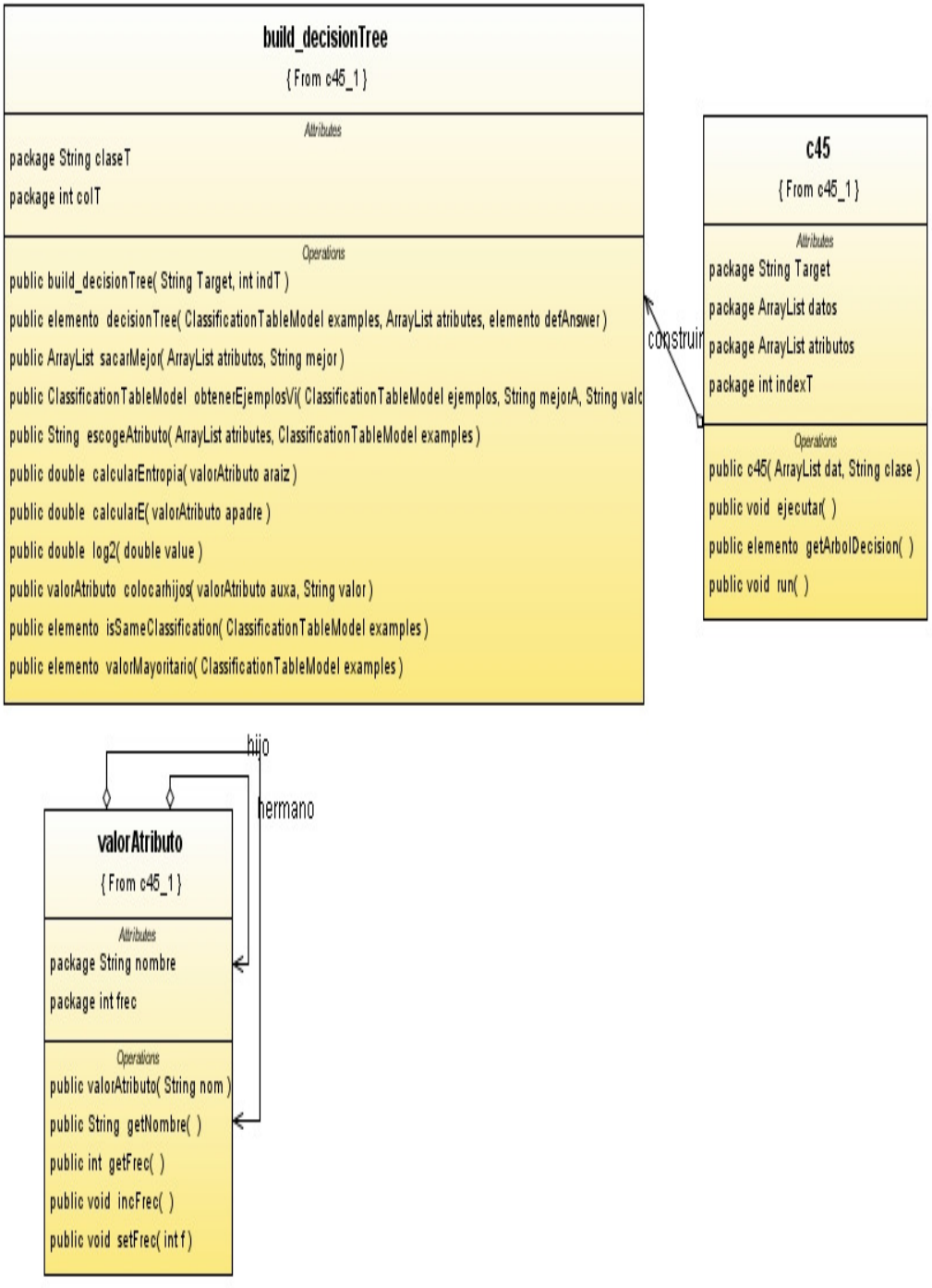
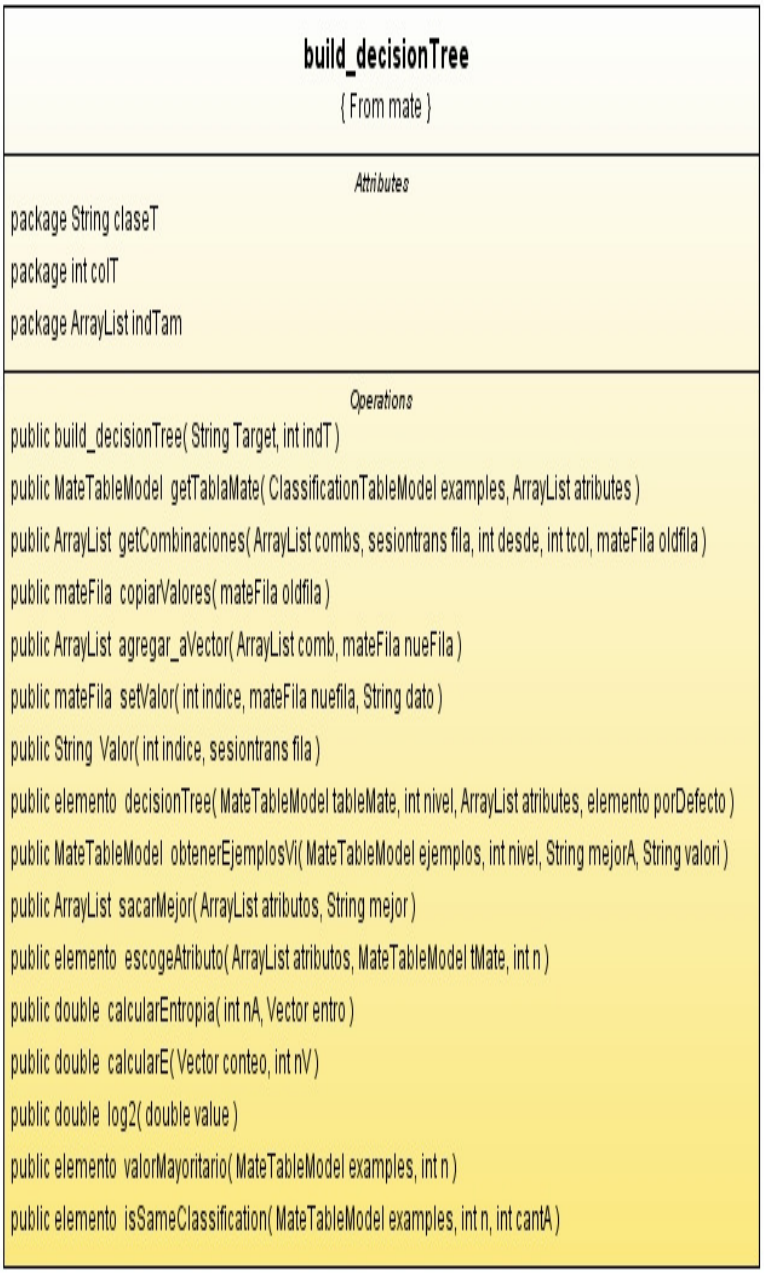
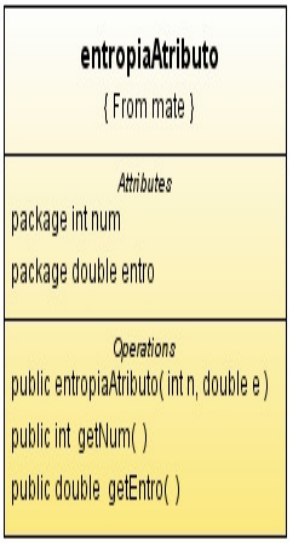
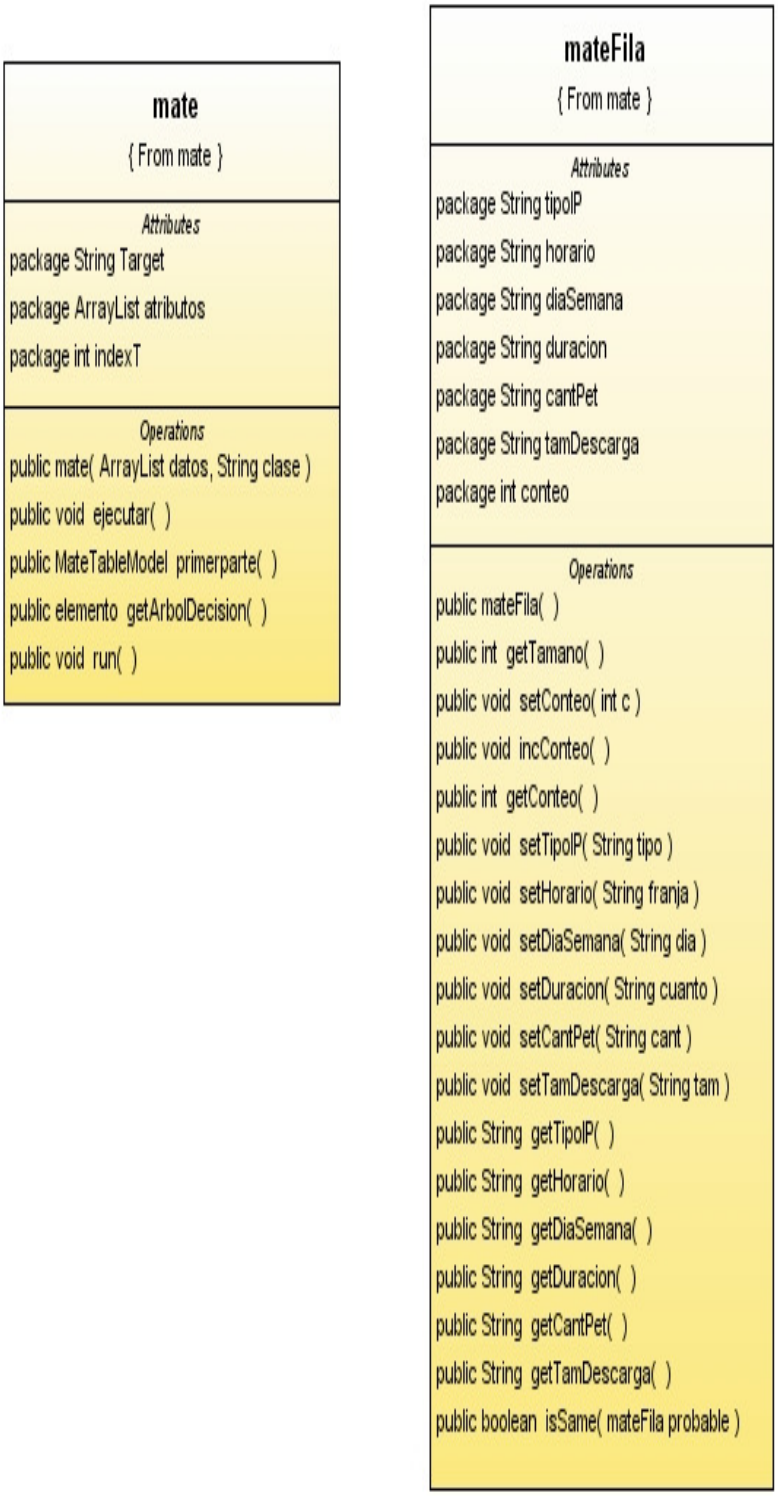

FPGrowth
{ From FPGrowth }
Attributes
package int sen
package int n
private short support
package Vector Trees
Operations
public FPGrowth(DataSet dataset, short support)
public short getSupport( )
public void buildFrequentsNodes(AvlNode t)
public FrequentNode findNode(short item)
public void buildTree( )
public void showTree(Stack stack)
public Vector getFrequents( )
public void buildFrequents( )
public int showFrequents( )
public void clearFrequents( )
public void run( )
FPGrowthNode
{ From FPGrowth }
Attributes
private short item
private short support
Operations
public FPGrowthNode(short item, short support)
public FPGrowthNode( )
public short getItem( )
public short getSupport( )
public void increaseSupport( )
Stack
{ From FPGrowth }
Attributes
Operations
public Stack( )
public void push(StackNode FPNode)
public FPGrowthNode pop( )
FrequentNode
{ From FPGrowth }
Attributes
Operations
public FrequentNode(short i, short s)
public short getItem( )
StackNode
{ From FPGrowth }
Attributes
Operations
package StackNode(FPGrowthNode node)
show
bro
next
fat
son
aux
auxh
nue
cab
pult
pcab
frequentsOne
FPNode
0..*
sigp
cabp

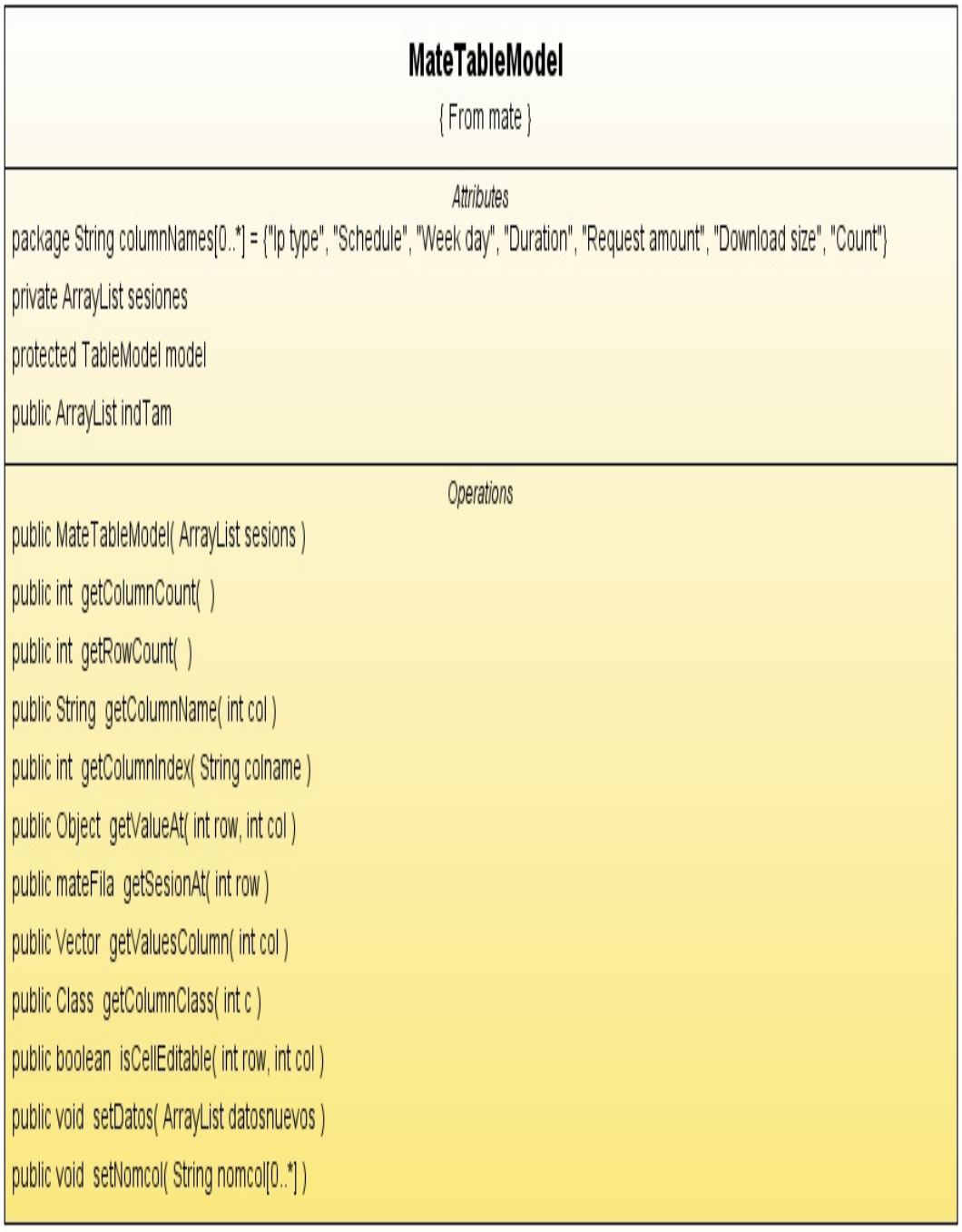
Figura 2.41.2. Paquete algorithm clasification

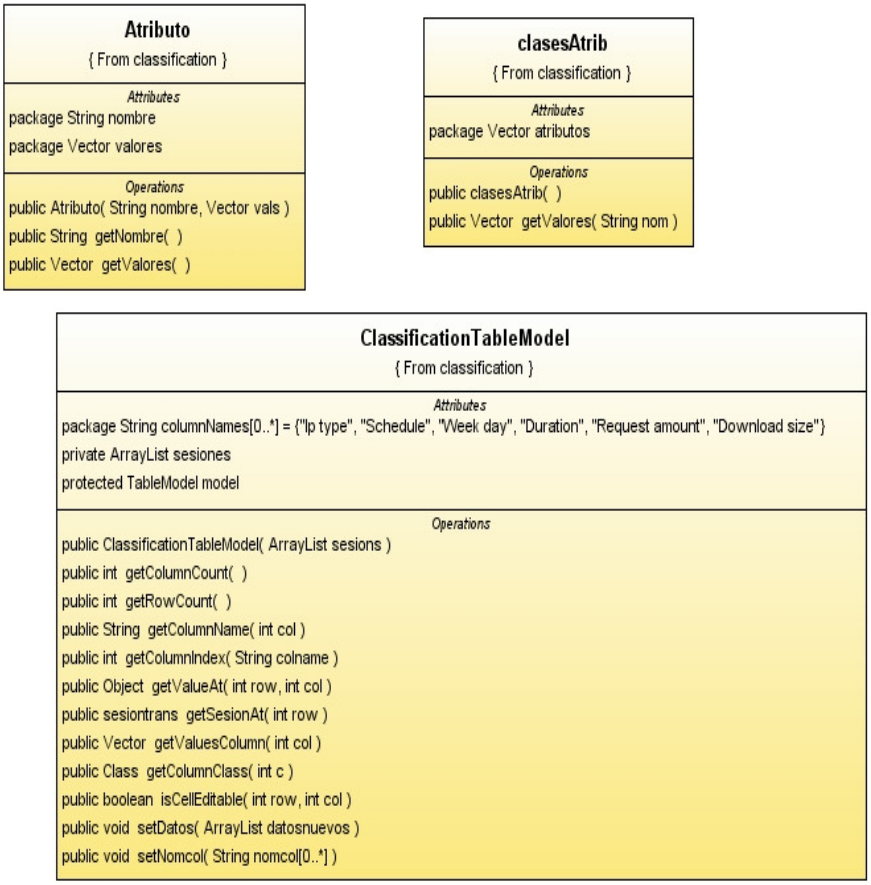



build_decisionTree
{ From mate }
Attributes
package String claseT
package int colT
package ArrayList indTam
Operations
public build_decisionTree(String Target, int indT)
public MateTableModel getTablaMate(ClassificationTableModel examples, ArrayList atributes)
public ArrayList getCombinaciones(ArrayList combs, sesiontrans fila, int desde, int tcol, mateFila oldfila)
public mateFila copiarValores(mateFila oldfila)
public ArrayList agregar_aVector(ArrayList comb, mateFila nueFila)
public mateFila setValor(int indice, mateFila nuefila, String dato)
public String Valor(int indice, sesiontrans fila)
public elemento decisionTree(MateTableModel tableMate, int nivel, ArrayList atributes, elemento porDefecto)
public MateTableModel obtenerEjemplosVi(MateTableModel ejemplos, int nivel, String mejorA, String valori)
public ArrayList sacarMejor(ArrayList atributos, String mejor)
public elemento escogeAtributo(ArrayList atributos, MateTableModel tMate, int n)
public double calcularEntropia(int nA, Vector entro)
public double calcularE(Vector conteo, int nV)
public double log2(double value)
public elemento valorMayoritario(MateTableModel examples, int n)
public elemento isSameClassification(MateTableModel examples, int n, int cantA)


entropiaAtributo
{ From mate }
Attributes
package int num
package double entro
Operations
public entropiaAtributo(int n, double e)
public int getNum( )
public double getEntro( )


mate
{ From mate }
Attributes
package String Target
package ArrayList atributos
package int indexT
Operations
public mate(ArrayList datos, String clase)
public void ejecutar( )
public MateTableModel primerparte( )
public elemento getArbolDecision( )
public void run( )
mateFila
{ From mate }
Attributes
package String tipoIP
package String horario
package String diaSemana
package String duracion
package String cantPet
package String tamDescarga
package int conteo
Operations
public mateFila( )
public int getTamano( )
public void setConteo(int c)
public void incConteo( )
public int getConteo( )
public void setTipoIP(String tipo)
public void setHorario(String franja)
public void setDiaSemana(String dia)
public void setDuracion(String cuanto)
public void setCantPet(String cant)
public void setTamDescarga(String tam)
public String getTipoIP( )
public String getHorario( )
public String getDiaSemana( )
public String getDuracion( )
public String getCantPet( )
public String getTamDescarga( )
public boolean isSame(mateFila probable)


MateTableModel
{ From mate }
Attributes
package String columnNames[0..*] = {"Ip type", "Schedule", "Week day", "Duration", "Request amount", "Download size", "Count"}
private ArrayList sesiones
protected TableModel model
public ArrayList indTam
Operations
public MateTableModel(ArrayList sesions)
public int getColumnCount( )
public int getRowCount( )
public String getColumnName(int col)
public int getColumnIndex(String colname)
public Object getValueAt(int row, int col)
public mateFila getSesionAt(int row)
public Vector getValuesColumn(int col)
public Class getColumnClass(int c)
public boolean isCellEditable(int row, int col)
public void setDatos(ArrayList datosnuevos)
public void setNomcol(String nomcol[0..*])


Atributo
{ From classification }
Attributes
package String nombre
package Vector valores
Operations
public Atributo(String nombre, Vector vals)
public String getNombre( )
public Vector getValores( )
clasesAtrib
{ From classification }
Attributes
package Vector atributos
Operations
public clasesAtrib( )
public Vector getValores(String nom)
ClassificationTableModel
{ From classification }
Attributes
package String columnNames[0..*] = {"Ip type", "Schedule", "Week day", "Duration", "Request amount", "Download size"}
private ArrayList sesiones
protected TableModel model
Operations
public ClassificationTableModel(ArrayList sesions)
public int getColumnCount( )
public int getRowCount( )
public String getColumnName(int col)
public int getColumnIndex(String colname)
public Object getValueAt(int row, int col)
public sesiontrans getSesionAt(int row)
public Vector getValuesColumn(int col)
public Class getColumnClass(int c)
public boolean isCellEditable(int row, int col)
public void setDatos(ArrayList datosnuevos)
public void setNomcol(String nomcol[0..*])

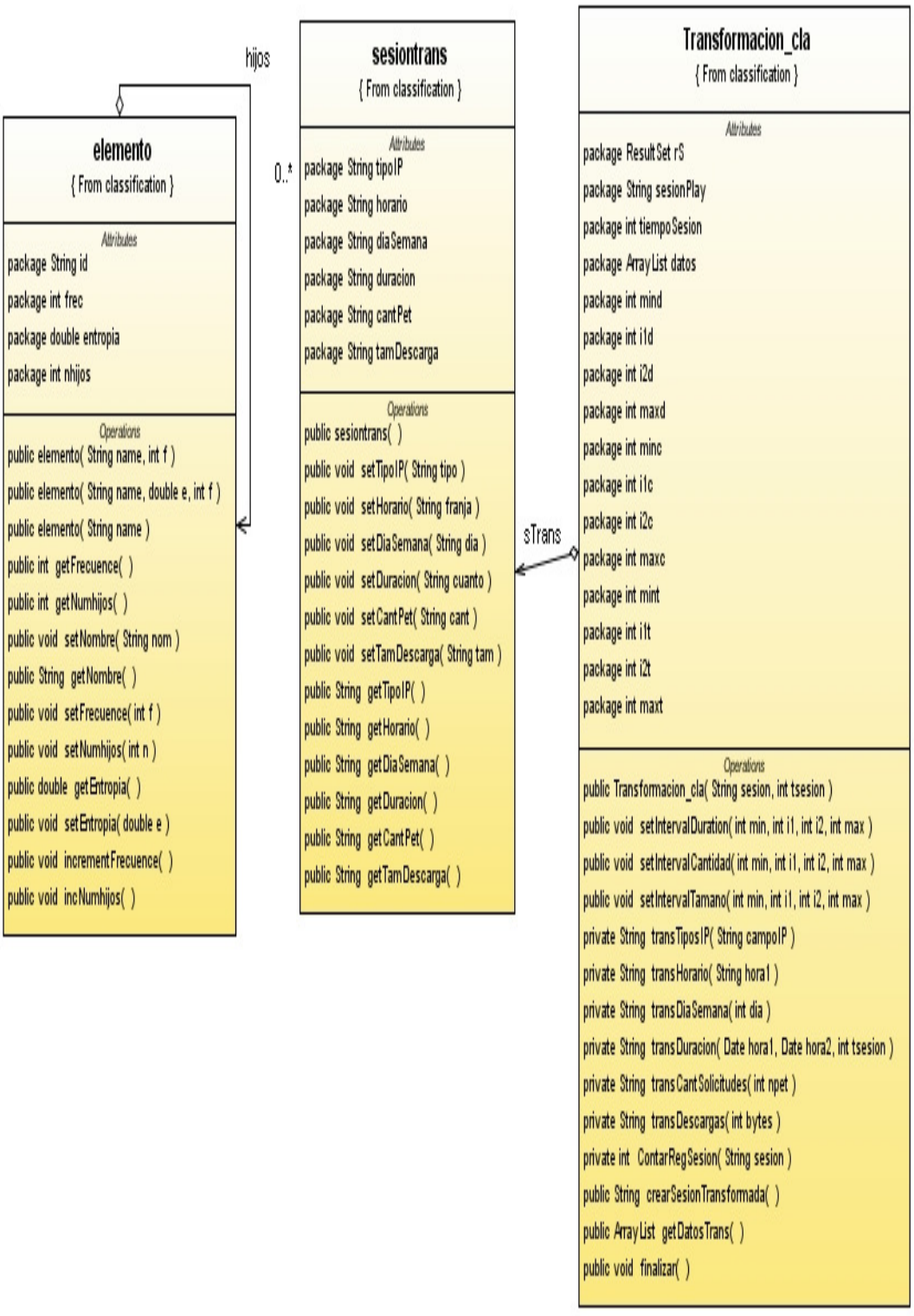

hijos
0..*
elemento
{ From classification }
Attributes
package String id
package int frec
package double entropia
package int nhijos
Operations
public elemento(String name, int f)
public elemento(String name, double e, int f)
public elemento(String name)
public int getFrecuence( )
public int getNumhijos( )
public void setNombre(String nom)
public String getNombre( )
public void setFrecuence(int f)
public void setNumhijos(int n)
public double getEntropia( )
public void setEntropia(double e)
public void incrementFrecuence( )
public void incNumhijos( )
sesiontrans
{ From classification }
Attributes
package String tipoIP
package String horario
package String diaSemana
package String duracion
package String cantPet
package String tamDescarga
Operations
public sesiontrans( )
public void setTipoIP(String tipo)
public void setHorario(String franja)
public void setDiaSemana(String dia)
public void setDuracion(String cuanto)
public void setCantPet(String cant)
public void setTamDescarga(String tam)
public String getTipoIP( )
public String getHorario( )
public String getDiaSemana( )
public String getDuracion( )
public String getCantPet( )
public String getTamDescarga( )
sTrans
Transformacion_cla
{ From classification }
Attributes
package ResultSet rS
package String sesionPlay
package int tiempoSesion
package ArrayList datos
package int mind
package int i1d
package int i2d
package int maxd
package int minc
package int i1c
package int i2c
package int maxc
package int mint
package int i1t
package int i2t
package int maxt
Operations
public Transformacion_cla(String sesion, int tsesion)
public void setIntervalDuration(int min, int i1, int i2, int max)
public void setIntervalCantidad(int min, int i1, int i2, int max)
public void setIntervalTamano(int min, int i1, int i2, int max)
private String transTiposIP(String campoIP)
private String transHorario(String hora1)
private String transDiaSemana(int dia)
private String transDuracion(Date hora1, Date hora2, int tsesion)
private String transCantSolicitudes(int npet)
private String transDescargas(int bytes)
private int ContarRegSesion(String sesion)
public String crearSesionTransformada( )
public ArrayList getDatosTrans( )
public void finalizar( )

Figura 2.41.3. Paquete HPG

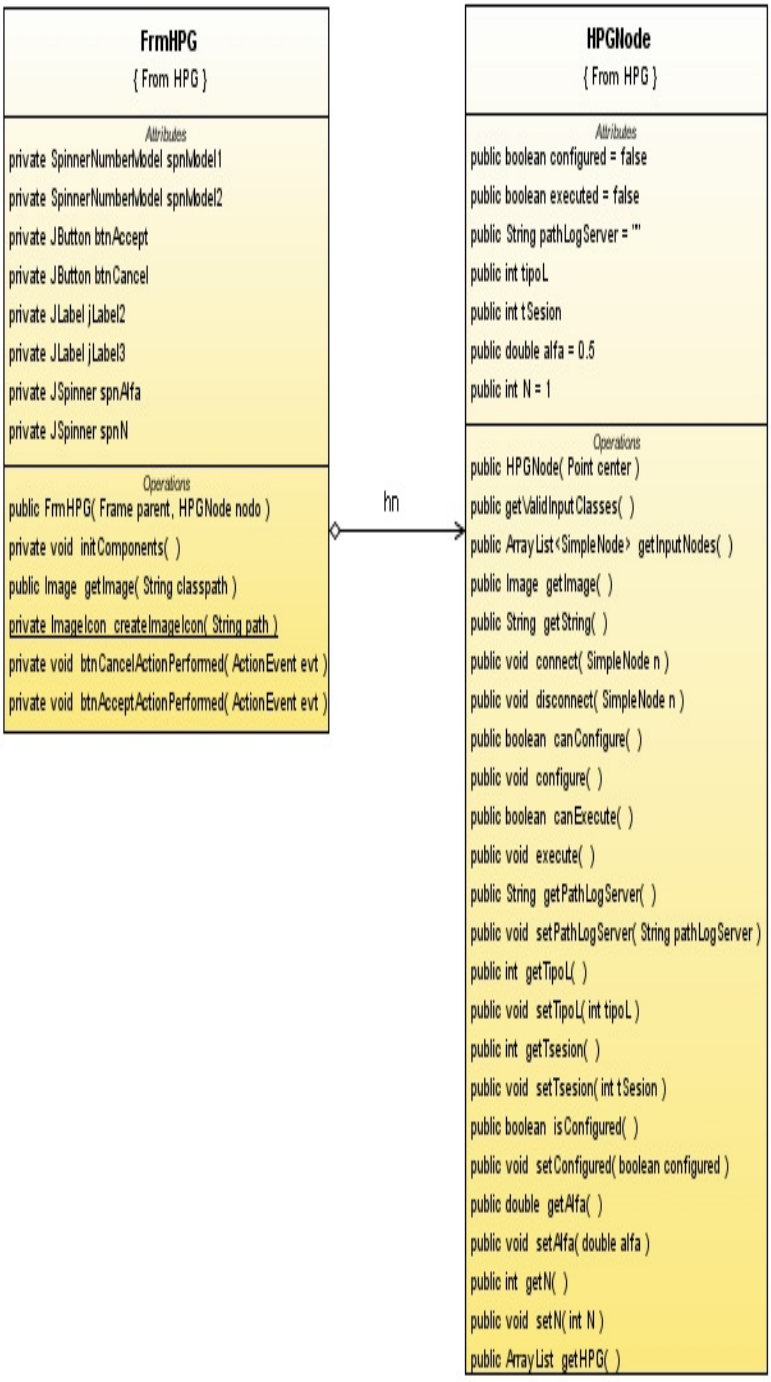
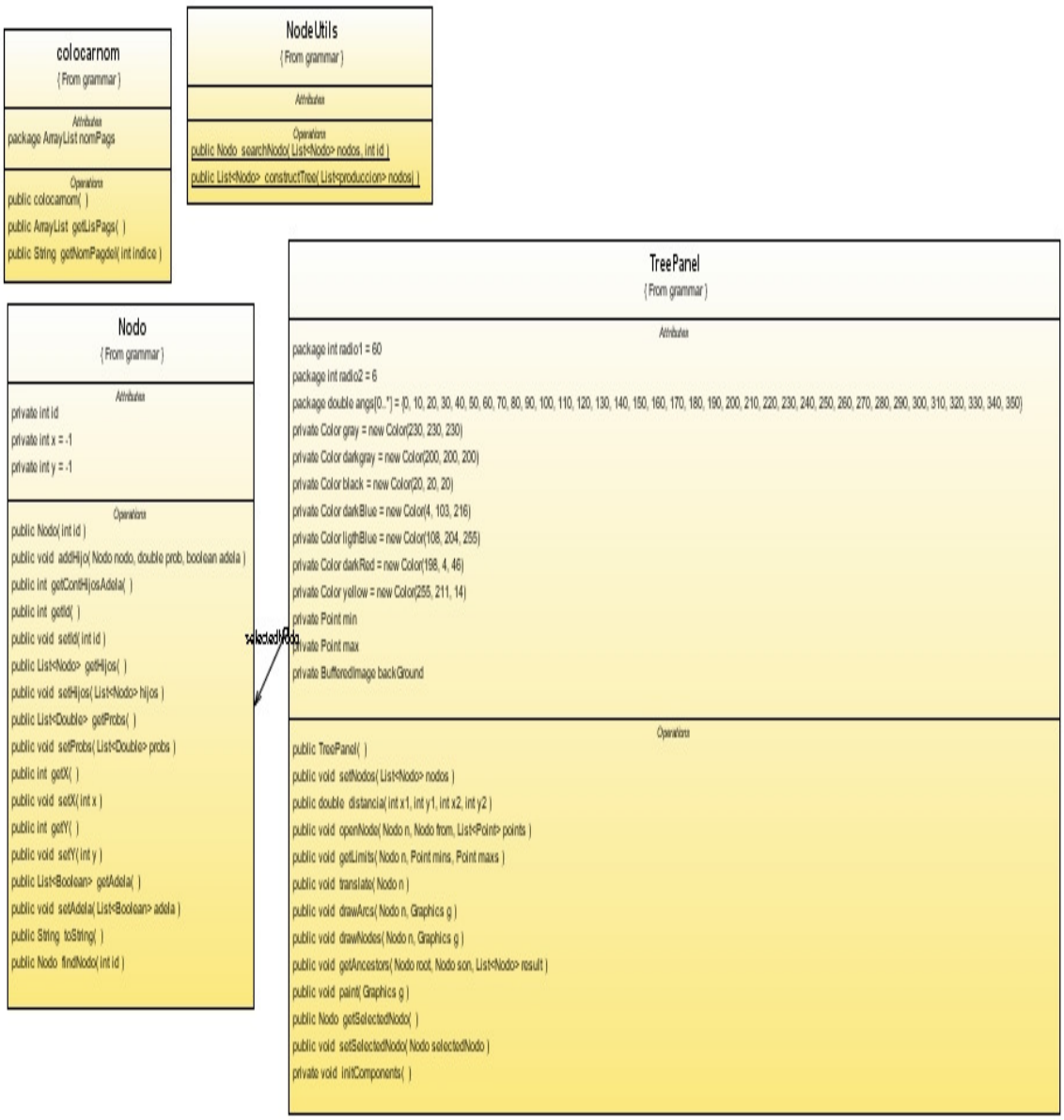


Figura 2.42. Paquete gramar



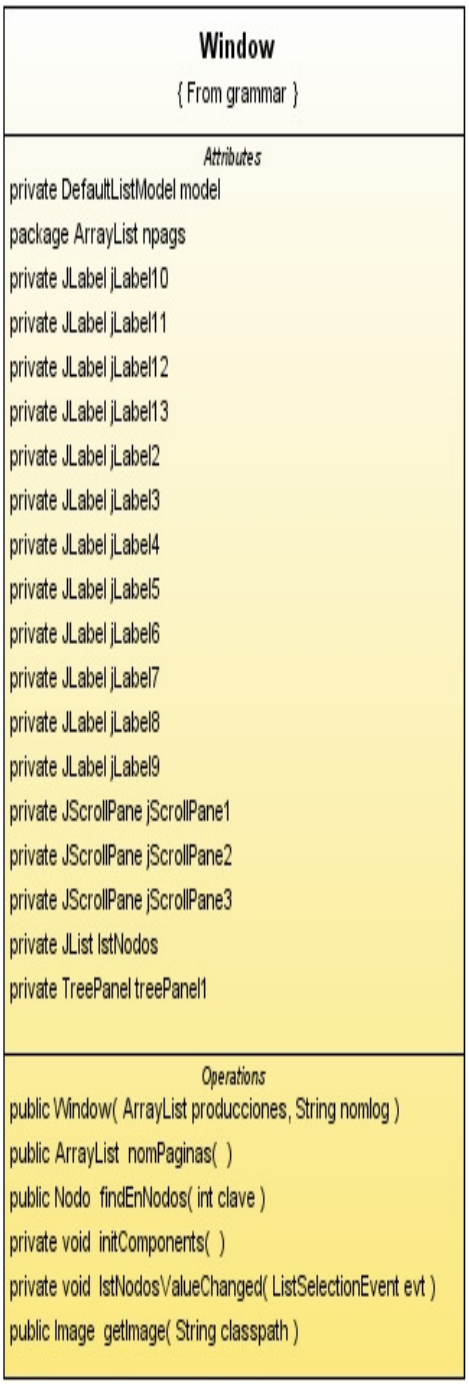
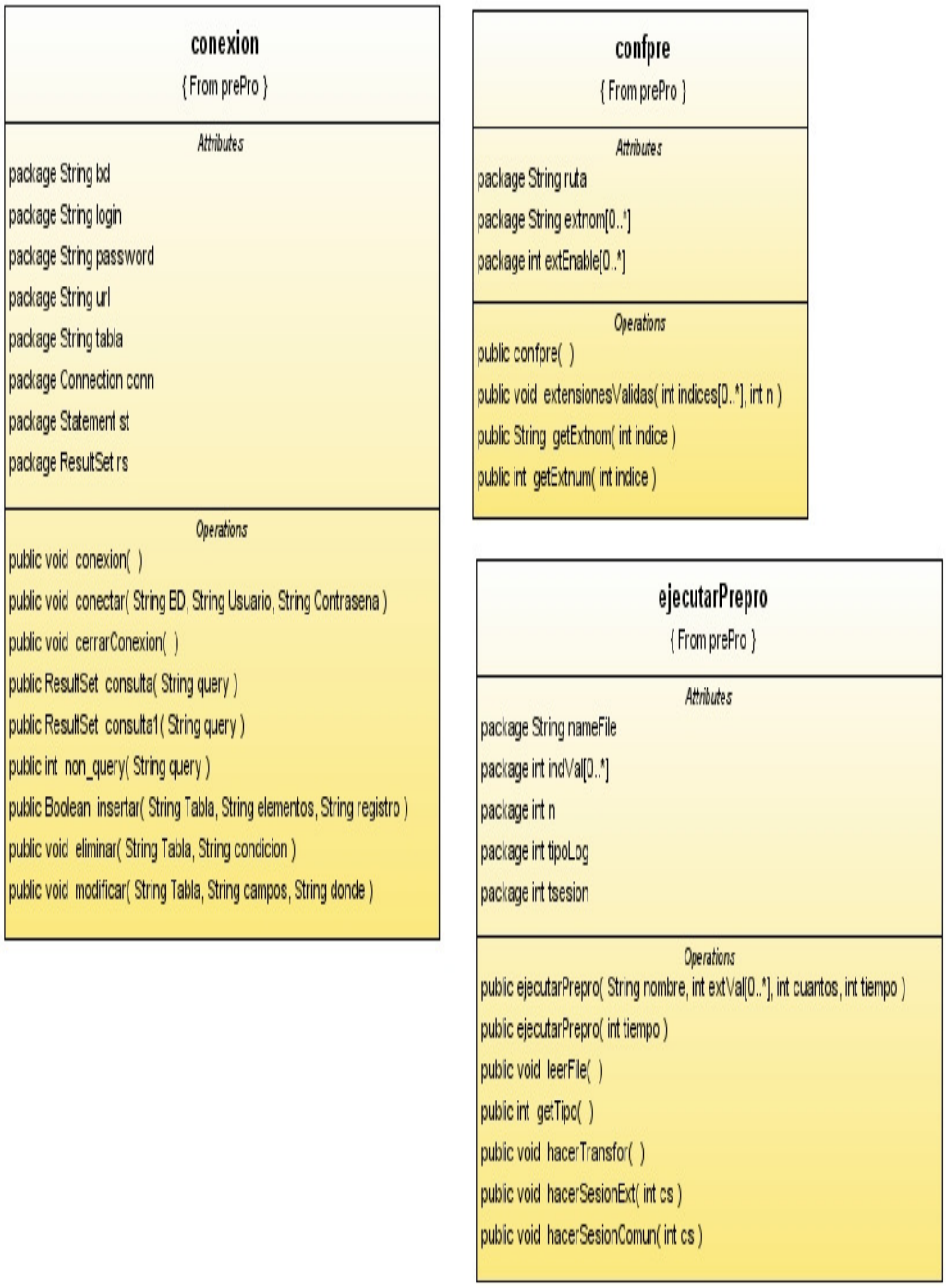
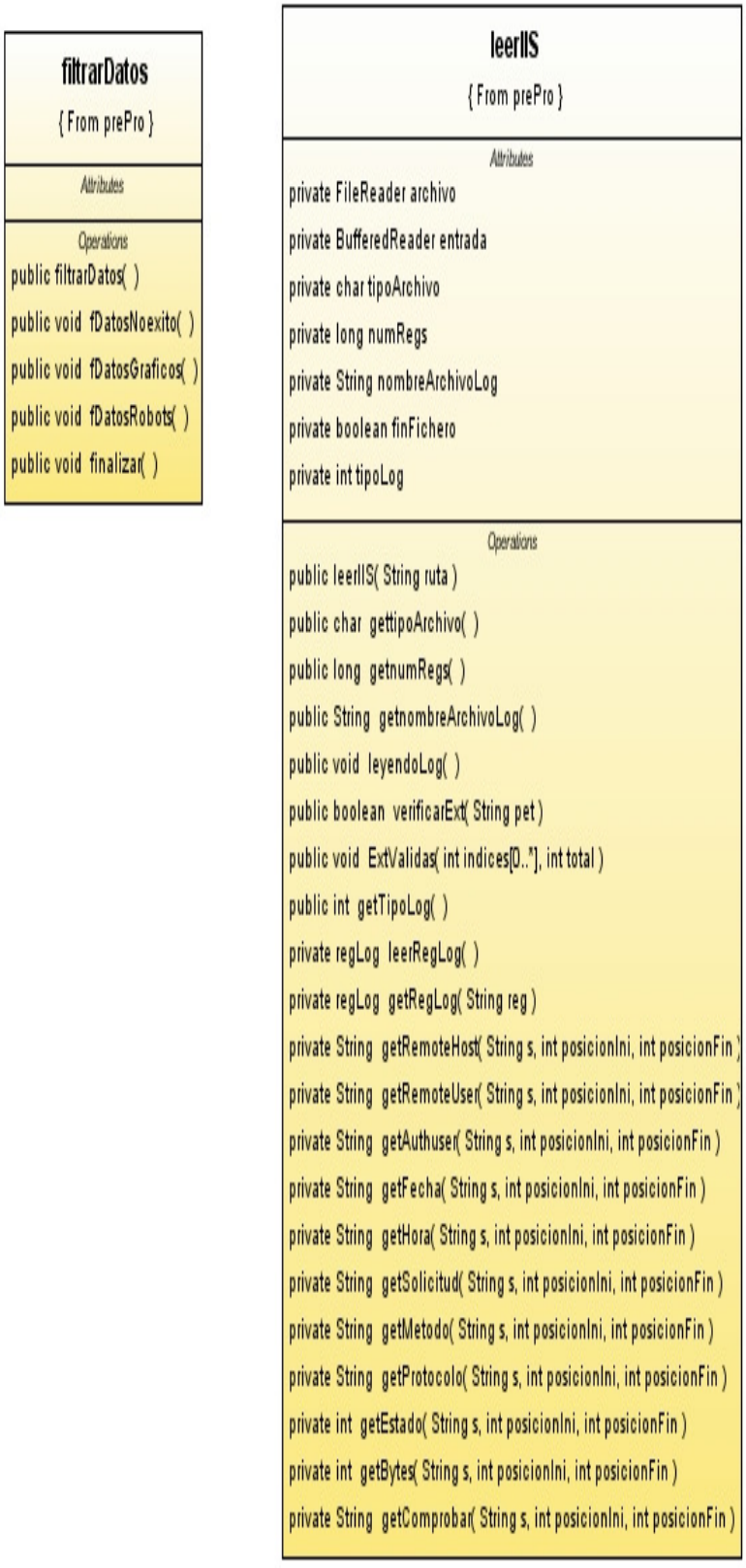

Window
{ From grammar }
Attributes
private DefaultListModel model
package ArrayList npags
private JLabel jLabel10
private JLabel jLabel11
private JLabel jLabel12
private JLabel jLabel13
private JLabel jLabel2
private JLabel jLabel3
private JLabel jLabel4
private JLabel jLabel5
private JLabel jLabel6
private JLabel jLabel7
private JLabel jLabel8
private JLabel jLabel9
private JScrollPane jScrollPane1
private JScrollPane jScrollPane2
private JScrollPane jScrollPane3
private JList lstNodos
private TreePanel treePanel1
Operations
public Window(ArrayList producciones, String nomlog)
public ArrayList nomPaginas( )
public Nodo findEnNodos(int clave)
private void initComponents( )
private void lstNodosValueChanged(ListSelectionEvent evt)
public Image getImage(String classpath)

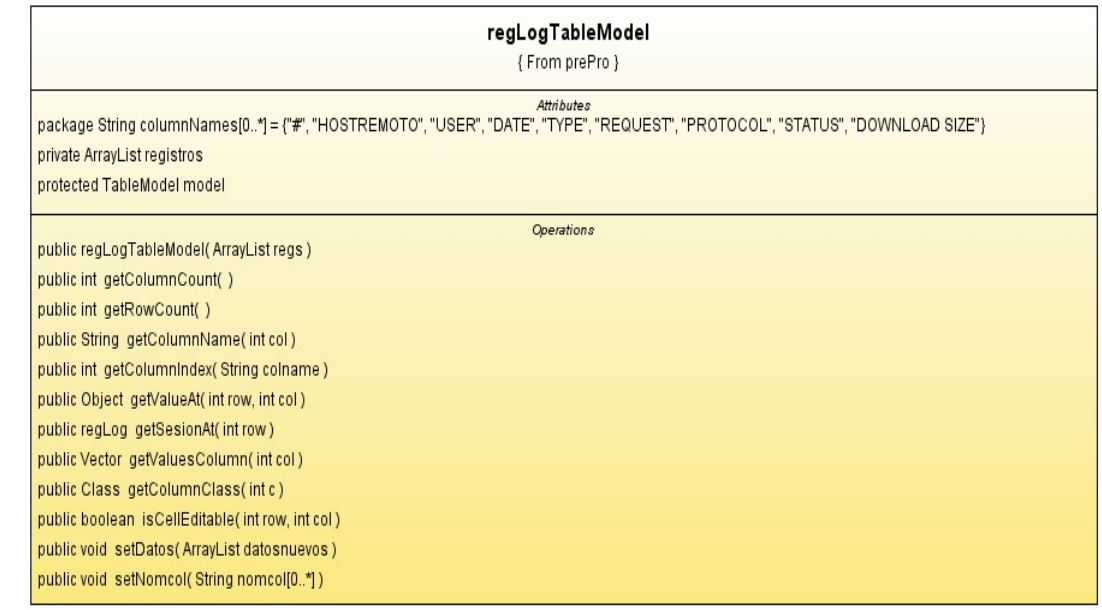
Figura 2.43. Paquete prePro

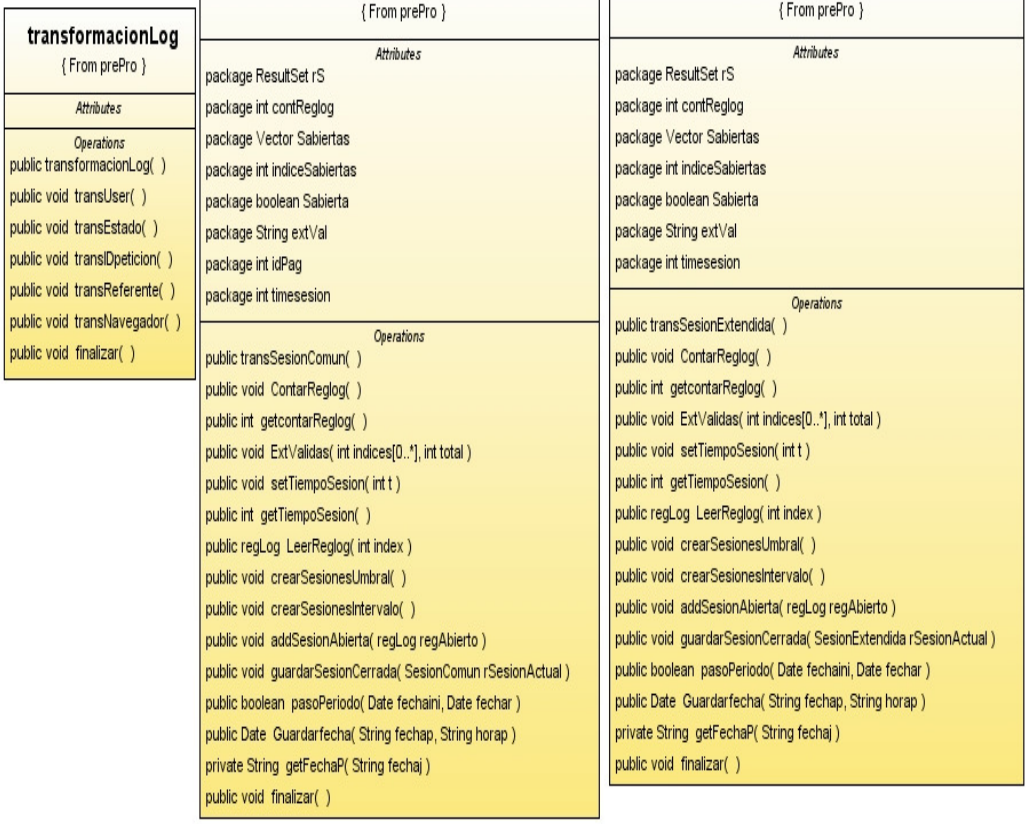


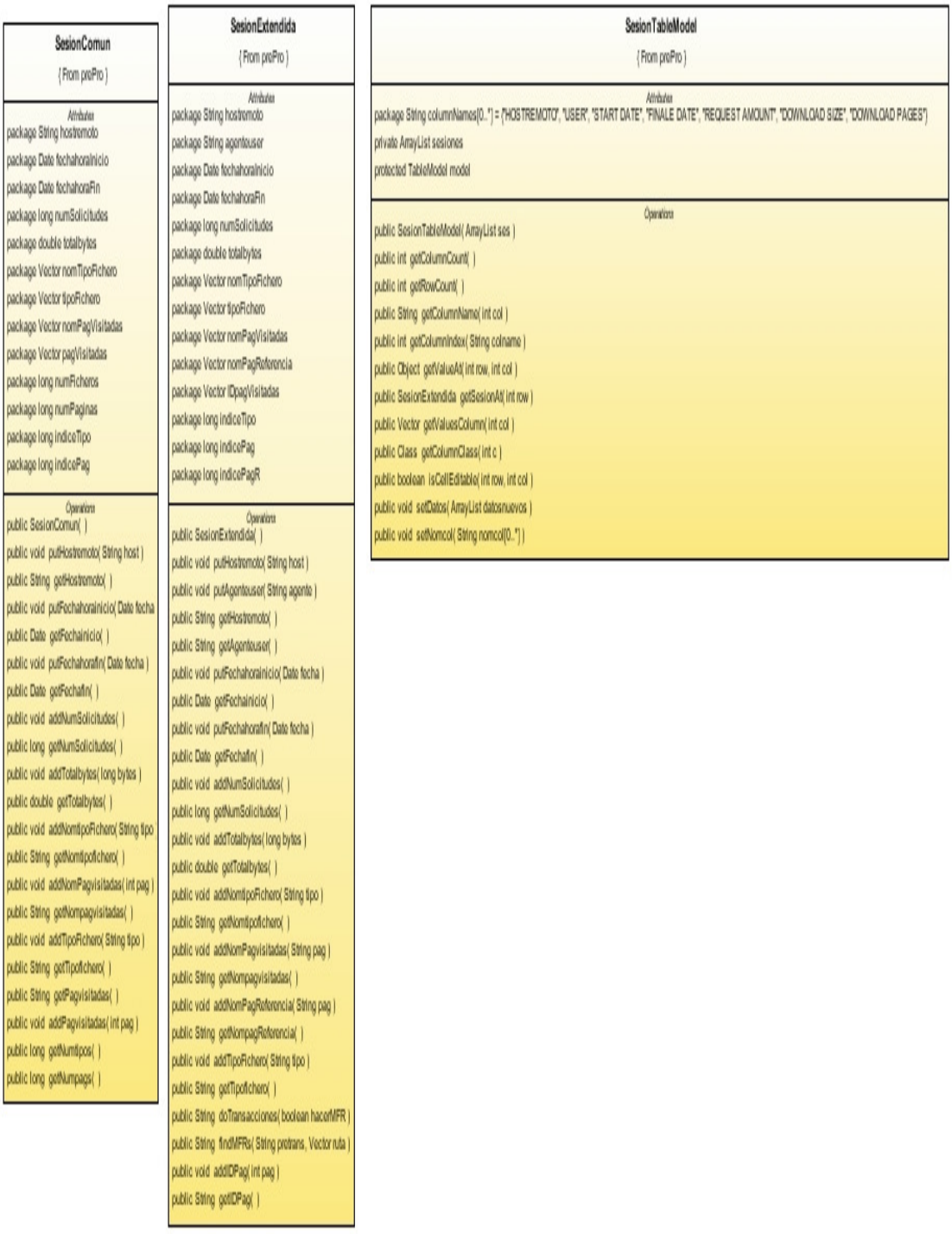

filtrarDatos
{ From prePro }
Attributes
Operations
public filtrarDatos( )
public void fDatosNoexito( )
public void fDatosGraficos( )
public void fDatosRobots( )
public void finalizar( )
leerIIS
{ From prePro }
Attributes
private FileReader archivo
private BufferedReader entrada
private char tipoArchivo
private long numRegs
private String nombreArchivoLog
private boolean finFichero
private int tipoLog
Operations
public leerIIS(String ruta)
public char gettipoArchivo( )
public long getnumRegs( )
public String getnombreArchivoLog( )
public void leyendoLog( )
public boolean verificarExt(String pet)
public void ExtValidas(int indices[0..*], int total)
public int getTipoLog( )
private regLog leerRegLog( )
private regLog getRegLog(String reg)
private String getRemoteHost(String s, int posicionIni, int posicionFin)
private String getRemoteUser(String s, int posicionIni, int posicionFin)
private String getAuthuser(String s, int posicionIni, int posicionFin)
private String getFecha(String s, int posicionIni, int posicionFin)
private String getHora(String s, int posicionIni, int posicionFin)
private String getSolicitud(String s, int posicionIni, int posicionFin)
private String getMetodo(String s, int posicionIni, int posicionFin)
private String getProtocolo(String s, int posicionIni, int posicionFin)
private int getEstado(String s, int posicionIni, int posicionFin)
private int getBytes(String s, int posicionIni, int posicionFin)
private String getComprobar(String s, int posicionIni, int posicionFin)

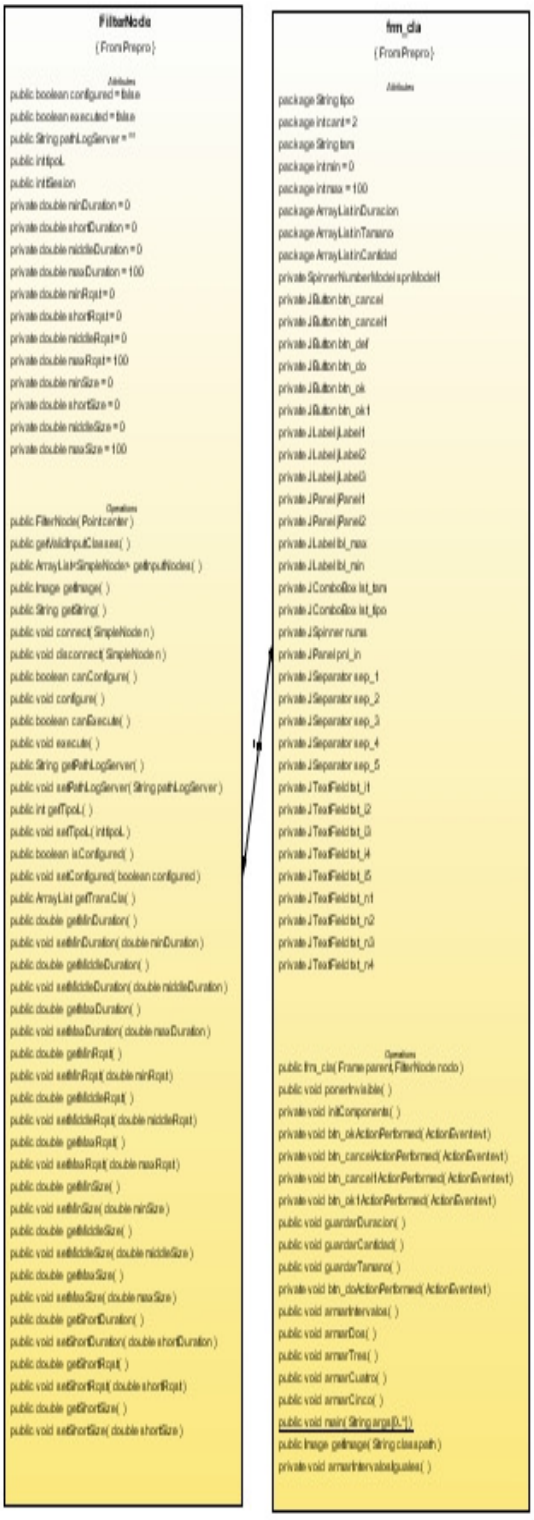

leerLog
{ From proPro }
Atributos
private FileReader archivo
private BufferedReader entrada
private char tipoArchivo
private long numRegs
private String nombreArchivoLog
private boolean finFichero
private boolean primerReg
private int tipoLog
Operations
public leerLog(String ruta)
public char gettipoArchivo( )
public long getnumRegs( )
public String getnombreArchivoLog( )
public void leyendoLog( )
public regLog leyendoPrimerLog( )
public boolean verificarExt(String pet)
public void ExtValidas(int indices[0..*], int total)
public int getTipoLog( )
private regLog leerRegLog( )
public void verIIS(String reg)
private regLog getRegLog(String reg)
private String getRemotoHost(String s, int posicionIni, int posicionFin
private String getRemotoUser(String s, int posicionIni, int posicionFin
private String getAuthuser(String s, int posicionIni, int posicionFin)
private String getFecha(String s, int posicionIni, int posicionFin)
private String getHora(String s, int posicionIni, int posicionFin)
private String getSolicitud(String s, int posicionIni, int posicionFin)
private String getMetodo(String s, int posicionIni, int posicionFin)
private String getProtocolo(String s, int posicionIni, int posicionFin)
private int getEstado(String s, int posicionIni, int posicionFin)
private int getBytes(String s, int posicionIni)
private String getReferente(String s, int posicionIni, int posicionFin)
private String getNavegador(String s, int posicionIni, int posicionFin)


regLog
{ From proPro }
Atributos
private String hostRemoto
private String usuario
private String fecha
private String hora
private String metodo
private String solicitud
private String protocolo
private int estado
private long bytes
private String referente
private String navegador
private int idPag
private int idRef
private int idNav
Operations
public regLog( )
public void putHostRemoto(String remotoHost
public void putAuser(String aUser)
public void putFecha(String Fecha)
public void putHora(String Hora)
public void putMetodo(String metodo)
public void putSolicitud(String solicitud)
public void putProtocolo(String protocolo)
public void putEstado(int estado)
public void putBytes(long bytes)
public void putReferente(String referente)
public void putNavegador(String navegador)
public void putIDpag(int id)
public void putIDref(int id)
public void putIDnav(int id)
public String getHostremoto( )
public String getaUser( )
public String getFecha( )
public String getHora( )
public String getMetodo( )
public String getSolicitud( )
public String getProtocolo( )
public int getEstado( )
public long getBytes( )
public String getReferente( )
public String getNavegador( )
public int getIDpag( )
public int getIDref( )
public int getIDnav( )
public String toString( )

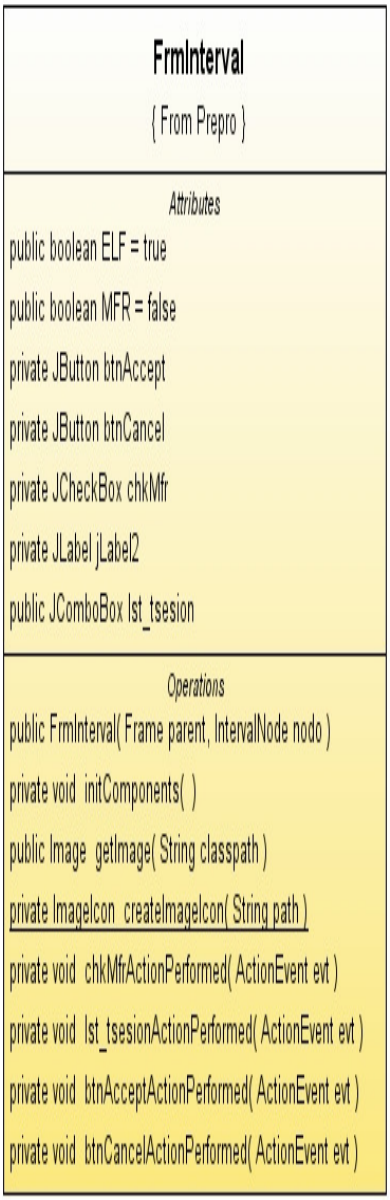

regLogTableModel
{ From prePro }
Attributes
package String columnNames[0..*] = {"#", "HOSTREMOTO", "USER", "DATE", "TYPE", "REQUEST", "PROTOCOL", "STATUS", "DOWNLOAD SIZE"}
private ArrayList registros
protected TableModel model
Operations
public regLogTableModel(ArrayList regs)
public int getColumnCount( )
public int getRowCount( )
public String getColumnName(int col)
public int getColumnIndex(String colname)
public Object getValueAt(int row, int col)
public regLog getSesionAt(int row)
public Vector getValuesColumn(int col)
public Class getColumnClass(int c)
public boolean isCellEditable(int row, int col)
public void setDatos(ArrayList datosnuevos)
public void setNomcol(String nomcol[0..*])


transformacionLog
{ From prePro }
Attributes
Operations
public transformacionLog( )
public void transUser( )
public void transEstado( )
public void transIDpeticion( )
public void transReferente( )
public void transNavegador( )
public void finalizar( )
transSesionComun
{ From prePro }
Attributes
package ResultSet rS
package int contReglog
package Vector Sabiertas
package int indiceSabiertas
package boolean Sabierta
package String extVal
package int idPag
package int timesesion
Operations
public transSesionComun( )
public void ContarReglog( )
public int getcontarReglog( )
public void ExtValidas(int indices[0..*], int total)
public void setTiempoSesion(int t)
public int getTiempoSesion( )
public regLog LeerReglog(int index)
public void crearSesionesUmbral( )
public void crearSesionesIntervalo( )
public void addSesionAbierta(regLog regAbierto)
public void guardarSesionCerrada(SesionComun rSesionActual)
public boolean pasoPeriodo(Date fechaini, Date fechar)
public Date Guardarfecha(String fechap, String horap)
private String getFechaP(String fechaj)
public void finalizar( )
transSesionExtendida
{ From prePro }
Attributes
package ResultSet rS
package int contReglog
package Vector Sabiertas
package int indiceSabiertas
package boolean Sabierta
package String extVal
package int timesesion
Operations
public transSesionExtendida( )
public void ContarReglog( )
public int getcontarReglog( )
public void ExtValidas(int indices[0..*], int total)
public void setTiempoSesion(int t)
public int getTiempoSesion( )
public regLog LeerReglog(int index)
public void crearSesionesUmbral( )
public void crearSesionesIntervalo( )
public void addSesionAbierta(regLog regAbierto)
public void guardarSesionCerrada(SesionExtendida rSesionActual)
public boolean pasoPeriodo(Date fechaini, Date fechar)
public Date Guardarfecha(String fechap, String horap)
private String getFechaP(String fechaj)
public void finalizar( )


SesionComun
{ From proPro }
Atributos
package String hostremoto
package Date fechahorainicio
package Date fechahoraFin
package long numSolicitudes
package double totalbytes
package Vector nomTipoFichero
package Vector tipoFichero
package Vector nomPagVisitadas
package Vector pagVisitadas
package long numFicheros
package long numPaginas
package long indiceTipo
package long indicePag
Operaciones
public SesionComun( )
public void putHostremoto(String host)
public String getHostremoto( )
public void putFechahorainicio(Date fecha
public Date getFechainicio( )
public void putFechahorafin(Date fecha)
public Date getFechafin( )
public void addNumSolicitudes( )
public long getNumSolicitudes( )
public void addTotalbytes(long bytes)
public double getTotalbytes( )
public void addNomtipoFichero(String tipo
public String getNomtipofichero( )
public void addNomPagvisitadas(int pag)
public String getNompagvisitadas( )
public void addTipoFichero(String tipo)
public String getTipofichero( )
public String getPagvisitadas( )
public void addPagvisitadas(int pag)
public long getNumtipos( )
public long getNumpags( )
SesionExtendida
{ From proPro }
Atributos
package String hostremoto
package String agenteuser
package Date fechahorainicio
package Date fechahoraFin
package long numSolicitudes
package double totalbytes
package Vector nomTipoFichero
package Vector tipoFichero
package Vector nomPagVisitadas
package Vector nomPagReferencia
package Vector IDpagVisitadas
package long indiceTipo
package long indicePag
package long indicePagR
Operaciones
public SesionExtendida( )
public void putHostremoto(String host)
public void putAgenteuser(String agente)
public String getHostremoto( )
public String getAgenteuser( )
public void putFechahorainicio(Date fecha)
public Date getFechainicio( )
public void putFechahorafin(Date fecha)
public Date getFechafin( )
public void addNumSolicitudes( )
public long getNumSolicitudes( )
public void addTotalbytes(long bytes)
public double getTotalbytes( )
public void addNomtipoFichero(String tipo)
public String getNomtipofichero( )
public void addNomPagvisitadas(String pag)
public String getNompagvisitadas( )
public void addNomPagReferencia(String pag)
public String getNompagReferencia( )
public void addTipoFichero(String tipo)
public String getTipofichero( )
public String doTransacciones(boolean hacerMFR)
public String findMFRs(String pretrans, Vector ruta)
public void addIDPag(int pag)
public String getIDPag( )
SesionTableModel
{ From proPro }
Atributos
package String columnNames[0..*] = {"HOSTREMOTO", "USER", "START DATE", "FINALE DATE", "REQUEST AMOUNT", "DOWNLOAD SIZE", "DOWNLOAD PAGES"}
privato ArrayList sesiones
protected TableModel model
Operaciones
public SesionTableModel(ArrayList ses)
public int getColumnCount( )
public int getRowCount( )
public String getColumnName(int col)
public int getColumnIndex(String colname)
public Object getValueAt(int row, int col)
public SesionExtendida getSesionAt(int row)
public Vector getValuesColumn(int col)
public Class getColumnClass(int c)
public boolean isCellEditable(int row, int col)
public void setDatos(ArrayList datosnuevos)
public void setNomcol(String nomcol[0..*])


FilterNode
{FromPrepro}
Attributes
public boolean configured = false
public boolean executed = false
public String pathLogServer = ""
public int tipoL
public int iSession
private double minDuration = 0
private double shortDuration = 0
private double middleDuration = 0
private double maxDuration = 100
private double minRqst = 0
private double shortRqst = 0
private double middleRqst = 0
private double maxRqst = 100
private double minSize = 0
private double shortSize = 0
private double middleSize = 0
private double maxSize = 100
Operations
public FilterNode(Point center)
public getValidInputClasses( )
public ArrayList<SimpleNode> getInputNodes( )
public Image getImage( )
public String getString( )
public void connect(SimpleNode n)
public void disconnect(SimpleNode n)
public boolean canConfigure( )
public void configure( )
public boolean canExecute( )
public void execute( )
public String getPathLogServer( )
public void setPathLogServer(String pathLogServer)
public int getTipoL( )
public void setTipoL(int tipoL)
public boolean isConfigured( )
public void setConfigured(boolean configured)
public ArrayList getTramaCla( )
public double getMinDuration( )
public void setMinDuration(double minDuration)
public double getMiddleDuration( )
public void setMiddleDuration(double middleDuration)
public double getMaxDuration( )
public void setMaxDuration(double maxDuration)
public double getMinRqst( )
public void setMinRqst(double minRqst)
public double getMiddleRqst( )
public void setMiddleRqst(double middleRqst)
public double getMaxRqst( )
public void setMaxRqst(double maxRqst)
public double getMinSize( )
public void setMinSize(double minSize)
public double getMiddleSize( )
public void setMiddleSize(double middleSize)
public double getMaxSize( )
public void setMaxSize(double maxSize)
public double getShortDuration( )
public void setShortDuration(double shortDuration)
public double getShortRqst( )
public void setShortRqst(double shortRqst)
public double getShortSize( )
public void setShortSize(double shortSize)
frm_cla
{FromPrepro}
Attributes
package String tipo
package int cant = 2
package String tam
package int min = 0
package int max = 100
package ArrayList tDuracion
package ArrayList tTamano
package ArrayList tCantidad
private SpinnerNumberModel spnModel1
private JButton btn_cancel
private JButton btn_cancel1
private JButton btn_def
private JButton btn_do
private JButton btn_ok
private JButton btn_ok1
private JLabel jLabel1
private JLabel jLabel2
private JLabel jLabel3
private JPanel jPanel1
private JPanel jPanel2
private JLabel lbl_max
private JLabel lbl_min
private JComboBox lst_tam
private JComboBox lst_tipo
private JSpinner nums
private JPanel pri_in
private JSeparator sep_1
private JSeparator sep_2
private JSeparator sep_3
private JSeparator sep_4
private JSeparator sep_5
private JTextField txt_t1
private JTextField txt_t2
private JTextField txt_t3
private JTextField txt_t4
private JTextField txt_t5
private JTextField txt_n1
private JTextField txt_n2
private JTextField txt_n3
private JTextField txt_n4
Operations
public frm_cla(Frame parent, FilterNode nodo)
public void ponerInvisible( )
private void initComponents( )
private void btn_okActionPerformed(ActionEvent evt)
private void btn_cancelActionPerformed(ActionEvent evt)
private void btn_cancel1ActionPerformed(ActionEvent evt)
private void btn_ok1ActionPerformed(ActionEvent evt)
public void guardarDuracion( )
public void guardarCantidad( )
public void guardarTamanno( )
private void btn_doActionPerformed(ActionEvent evt)
public void armarIntervalos( )
public void armarDos( )
public void armarTres( )
public void armarCuatro( )
public void armarCinco( )
public void main(String args[])
public Image getImage(String classpath)
private void armarIntervaloIguales( )


FrmGAP
{ From Prepro }
Attributes
public boolean ELF = true
public boolean MFR = false
private JButton btnAccept
private JButton btnCancel
private JCheckBox chkMfr
private JLabel jLabel2
public JComboBox lst_tsesion
Operations
public FrmGAP(Frame parent, UmbralNode nodo)
private void initComponents( )
public Image getImage(String classpath)
private ImageIcon createImageIcon(String path)
private void chkMfrActionPerformed(ActionEvent evt)
private void lst_tsesionActionPerformed(ActionEvent evt)
private void btnAcceptActionPerformed(ActionEvent evt)
private void btnCancelActionPerformed(ActionEvent evt)


FrmInterval
{ From Prepro }
Attributes
public boolean ELF = true
public boolean MFR = false
private JButton btnAccept
private JButton btnCancel
private JCheckBox chkMfr
private JLabel jLabel2
public JComboBox lst_tsesion
Operations
public FrmInterval(Frame parent, IntervalNode nodo)
private void initComponents( )
public Image getImage(String classpath)
private ImageIcon createImageIcon(String path)
private void chkMfrActionPerformed(ActionEvent evt)
private void lst_tsesionActionPerformed(ActionEvent evt)
private void btnAcceptActionPerformed(ActionEvent evt)
private void btnCancelActionPerformed(ActionEvent evt)

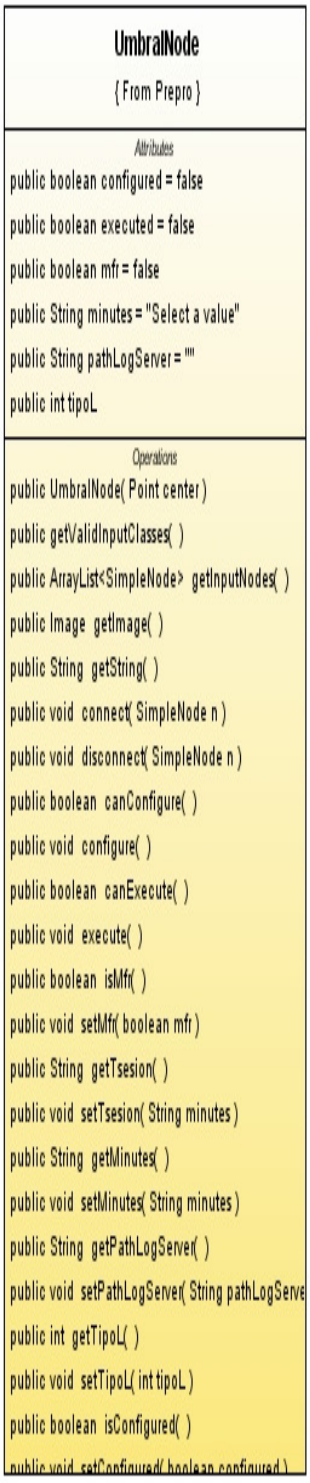

UmbralNode
{ From Prepro }
Attributes
public boolean configured = false
public boolean executed = false
public boolean mfr = false
public String minutes = "Select a value"
public String pathLogServer = ""
public int tipoL
Operations
public UmbralNode(Point center)
public getValidInputClasses( )
public ArrayList<SimpleNode> getInputNodes( )
public Image getImage( )
public String getString( )
public void connect(SimpleNode n)
public void disconnect(SimpleNode n)
public boolean canConfigure( )
public void configure( )
public boolean canExecute( )
public void execute( )
public boolean isMfr( )
public void setMfr(boolean mfr)
public String getTsesion( )
public void setTsesion(String minutes)
public String getMinutes( )
public void setMinutes(String minutes)
public String getPathLogServer( )
public void setPathLogServer(String pathLogServe
public int getTipoL( )
public void setTipoL(int tipoL)
public boolean isConfigured( )

Figura 2.44. Paquete statistical

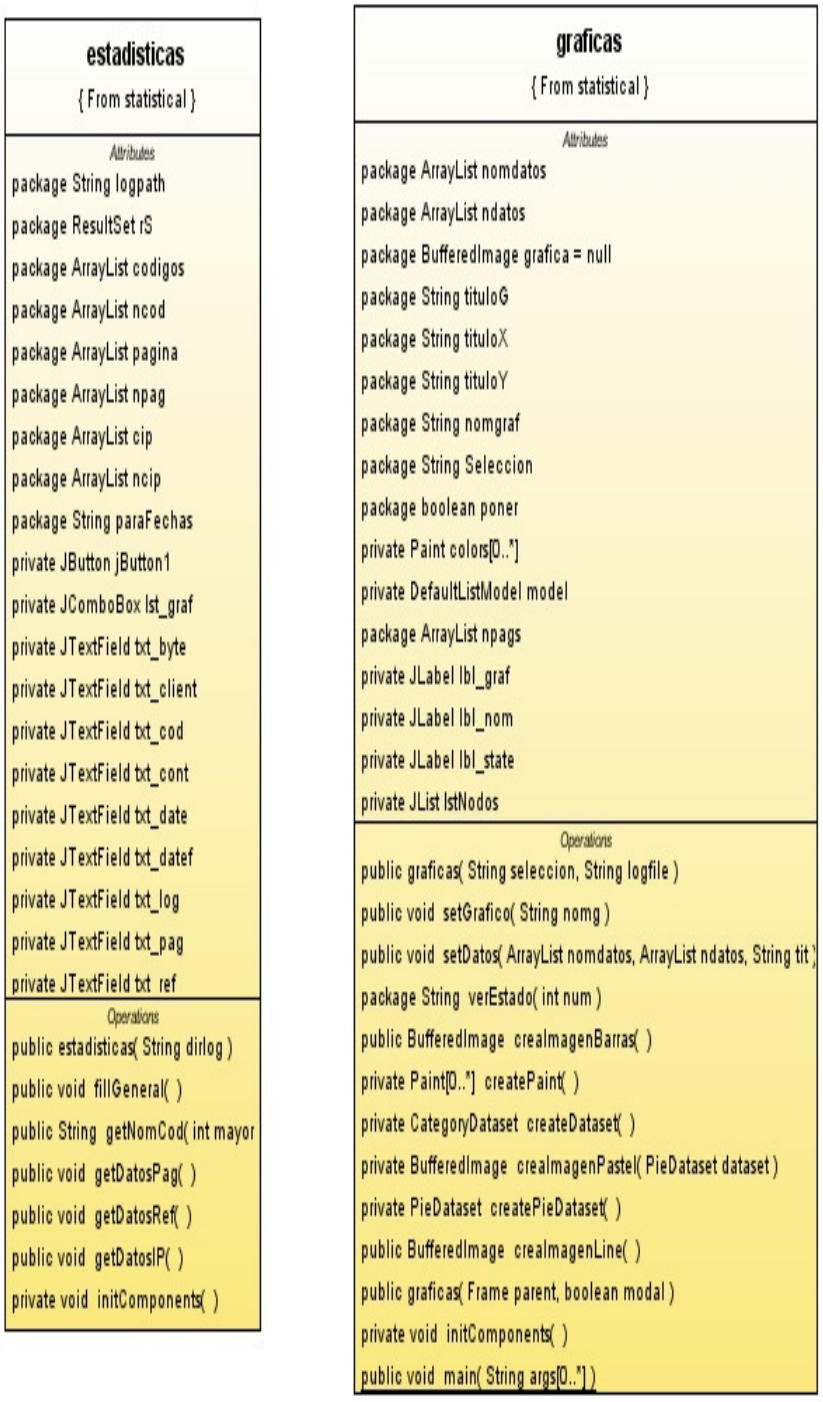


Figura 2.45. Paquete treeA

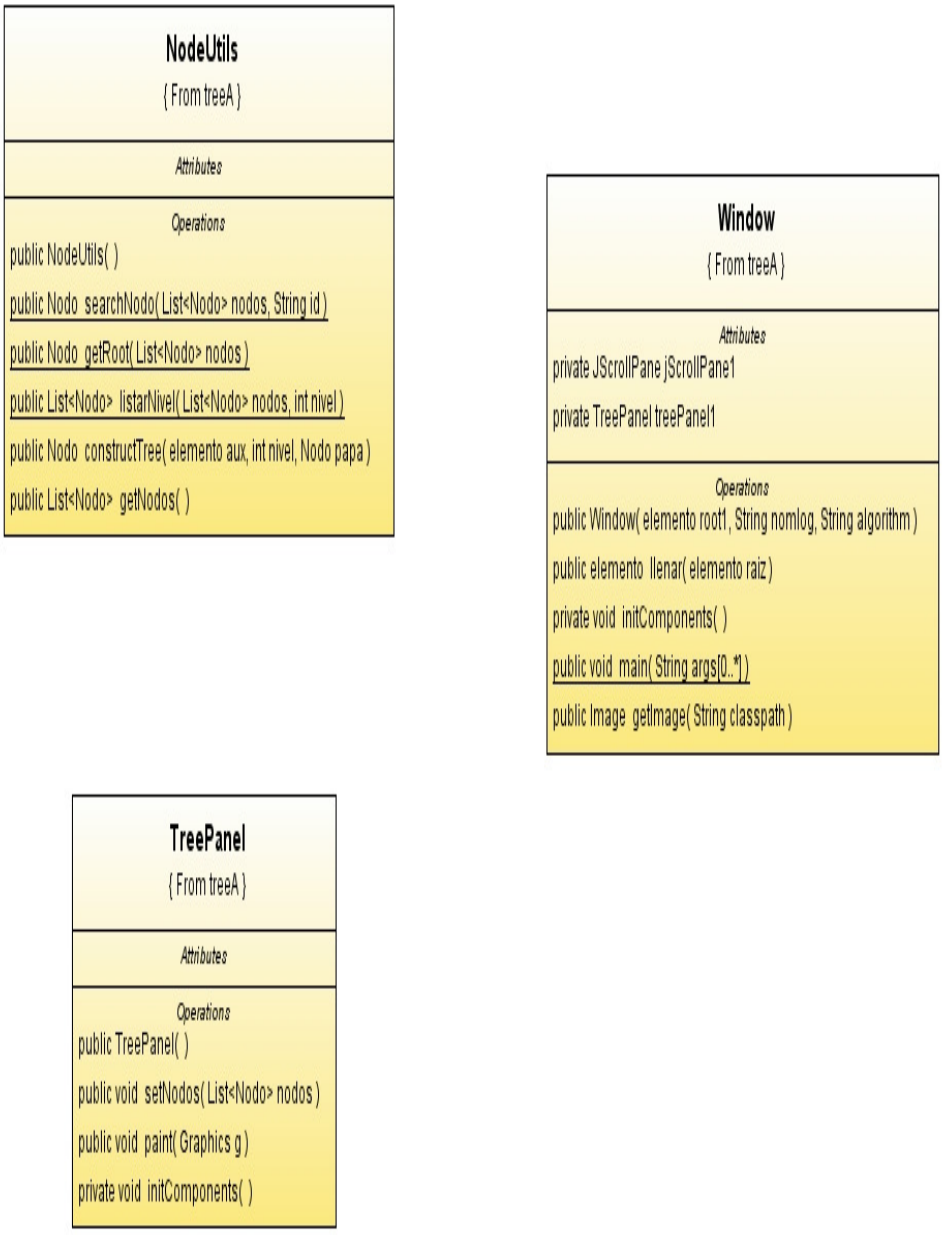
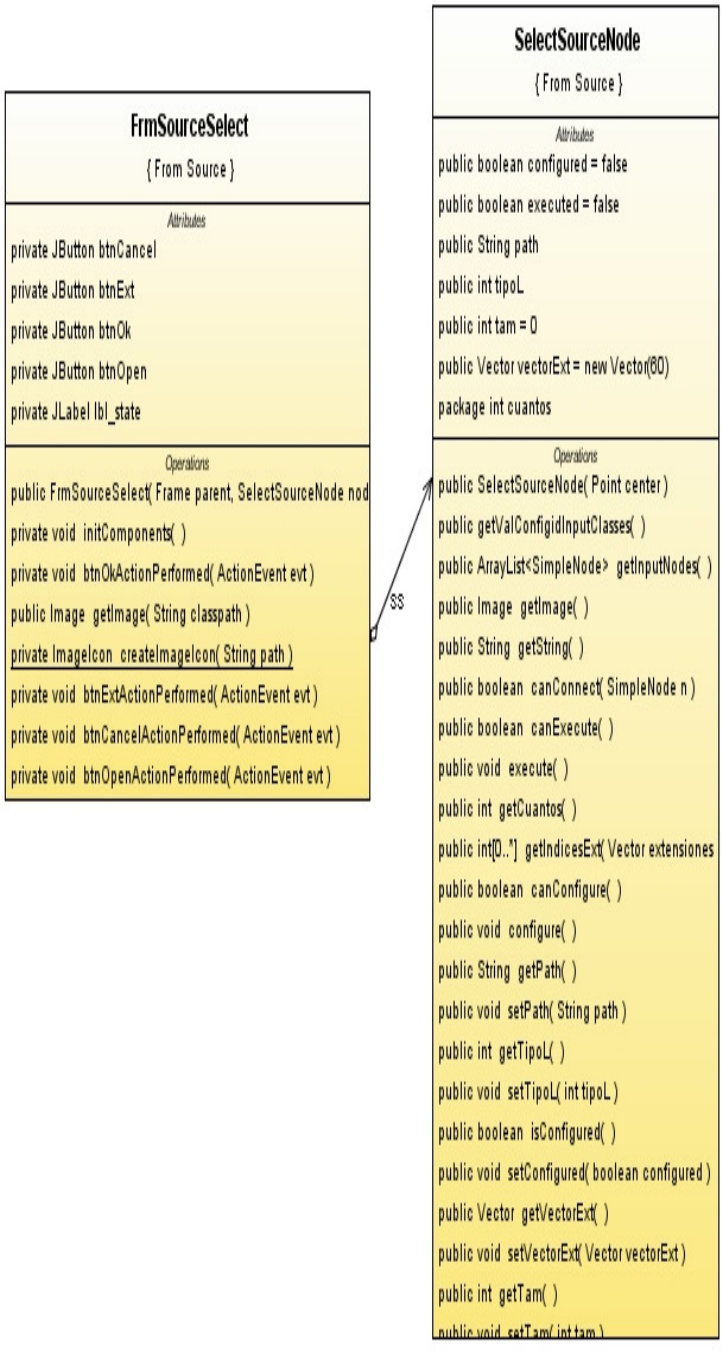
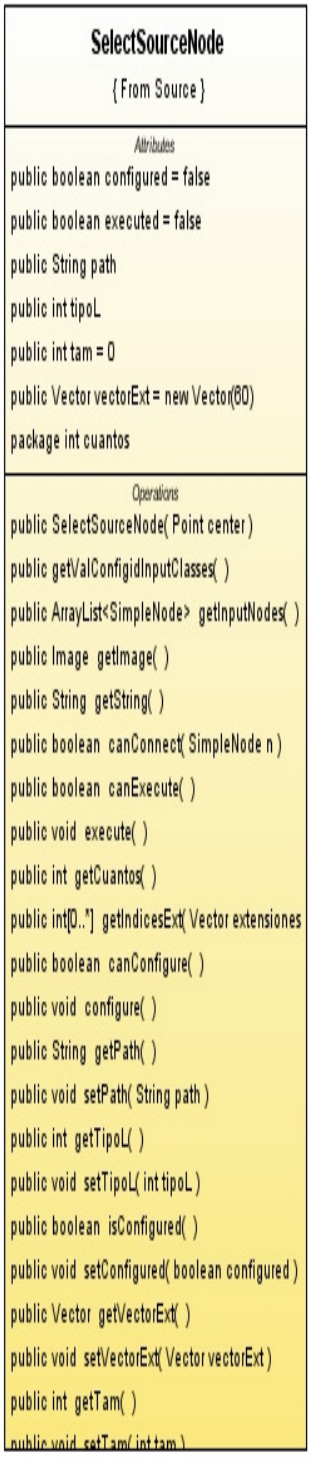


Figura 2.46. Paquete resource


FrmSourceSelect
{ From Source }
Attributes
private JButton btnCancel
private JButton btnExt
private JButton btnOk
private JButton btnOpen
private JLabel lbl_state
Operations
public FrmSourceSelect(Frame parent, SelectSourceNode nod
private void initComponents( )
private void btnOkActionPerformed(ActionEvent evt)
public Image getImage(String classpath)
private ImageIcon createImageIcon(String path)
private void btnExtActionPerformed(ActionEvent evt)
private void btnCancelActionPerformed(ActionEvent evt)
private void btnOpenActionPerformed(ActionEvent evt)
ss
SelectSourceNode
{ From Source }
Attributes
public boolean configured = false
public boolean executed = false
public String path
public int tipoL
public int tam = 0
public Vector vectorExt = new Vector(60)
package int cuantos
Operations
public SelectSourceNode(Point center)
public getValConfigidInputClasses( )
public ArrayList<SimpleNode> getInputNodes( )
public Image getImage( )
public String getString( )
public boolean canConnect(SimpleNode n)
public boolean canExecute( )
public void execute( )
public int getCuantos( )
public int[0..*] getIndicesExt(Vector extensiones
public boolean canConfigure( )
public void configure( )
public String getPath( )
public void setPath(String path)
public int getTipoL( )
public void setTipoL(int tipoL)
public boolean isConfigured( )
public void setConfigured(boolean configured)
public Vector getVectorExt( )
public void setVectorExt(Vector vectorExt)
public int getTam( )


SelectSourceNode
{ From Source }
Attributes
public boolean configured = false
public boolean executed = false
public String path
public int tipoL
public int tam = 0
public Vector vectorExt = new Vector(60)
package int cuantos
Operations
public SelectSourceNode(Point center)
public getValConfigidInputClasses( )
public ArrayList<SimpleNode> getInputNodes( )
public Image getImage( )
public String getString( )
public boolean canConnect(SimpleNode n)
public boolean canExecute( )
public void execute( )
public int getCuantos( )
public int[0..*] getIndicesExt(Vector extensiones
public boolean canConfigure( )
public void configure( )
public String getPath( )
public void setPath(String path)
public int getTipoL( )
public void setTipoL(int tipoL)
public boolean isConfigured( )
public void setConfigured(boolean configured)
public Vector getVectorExt( )
public void setVectorExt(Vector vectorExt)
public int getTam( )

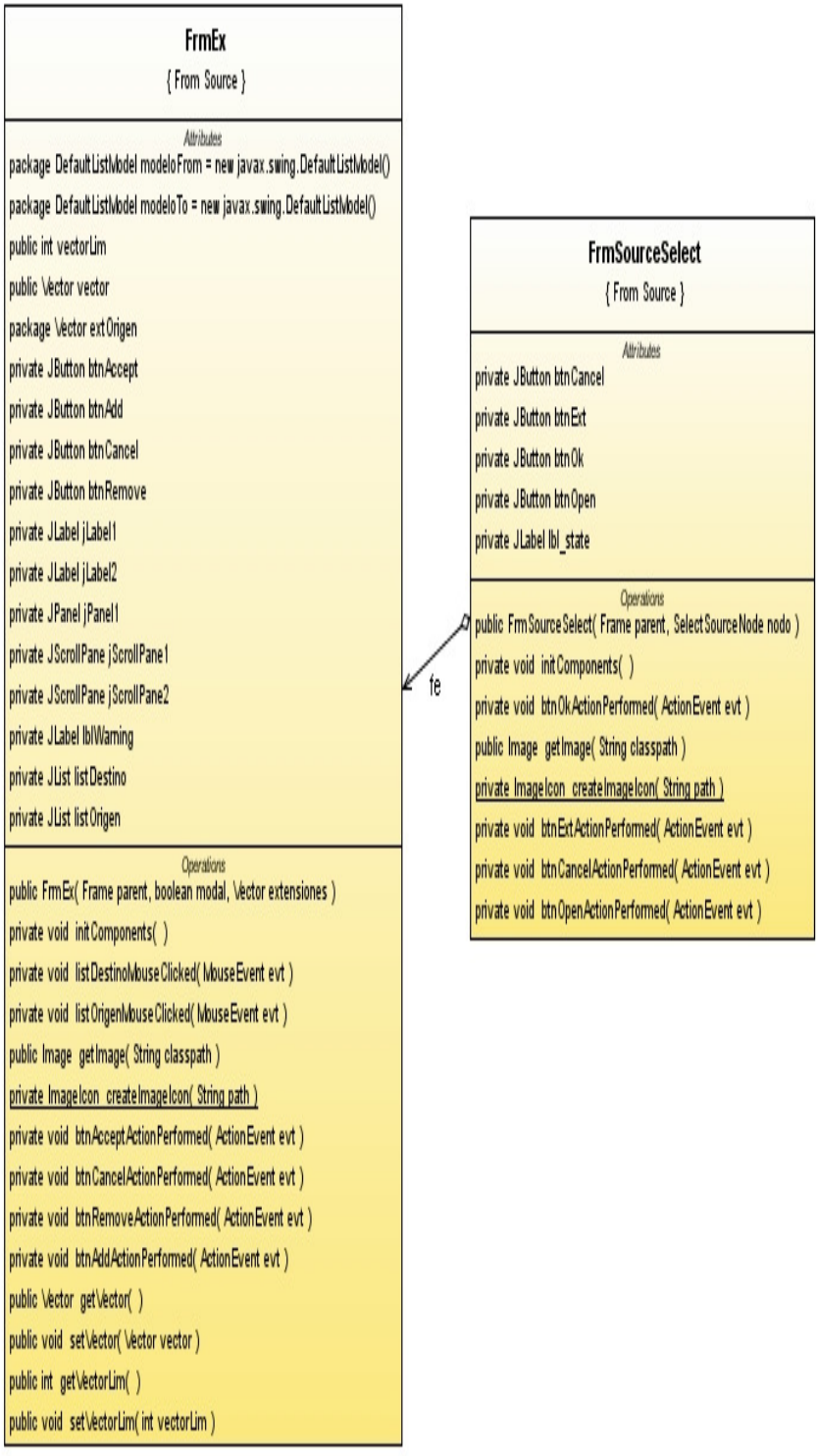
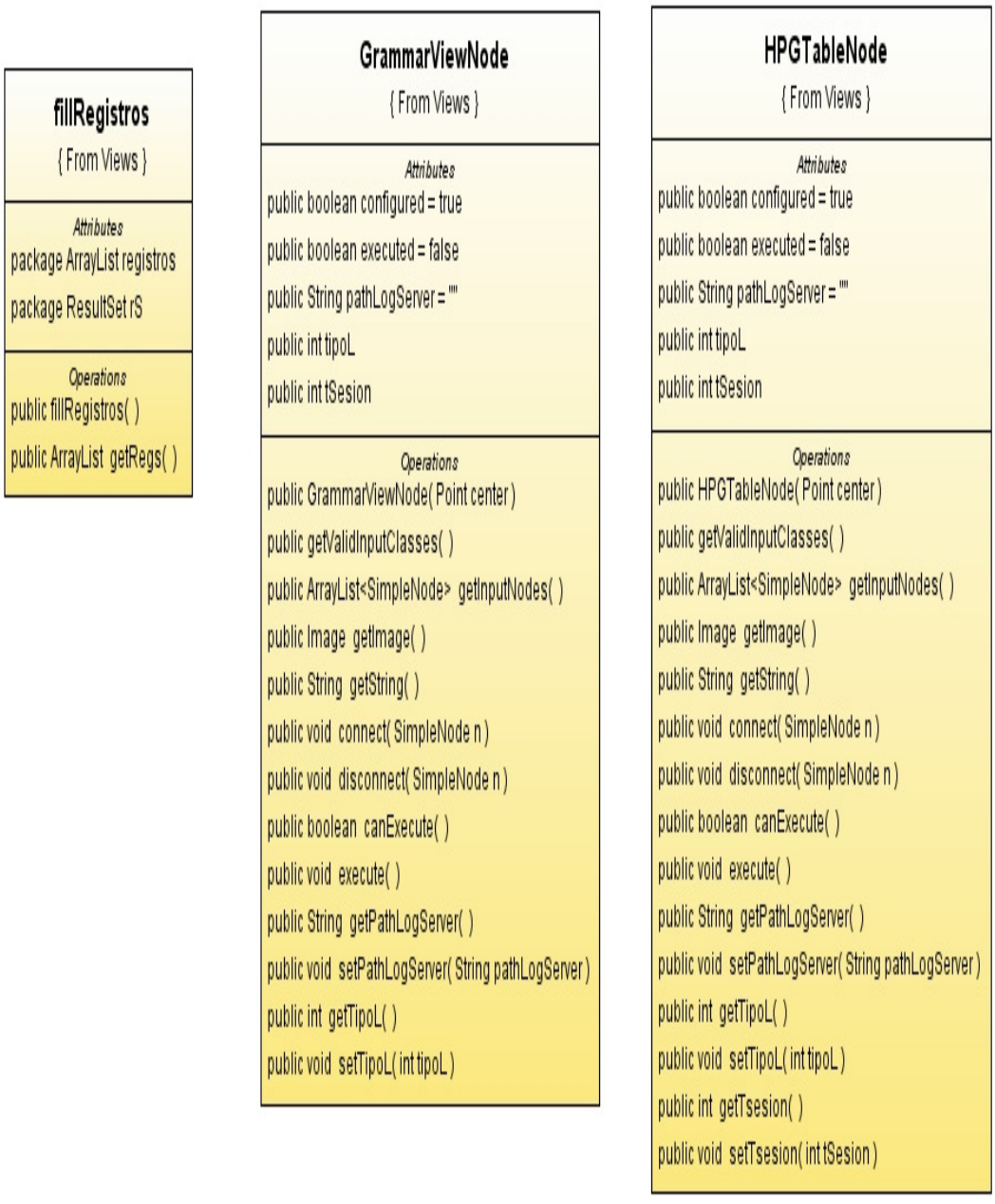
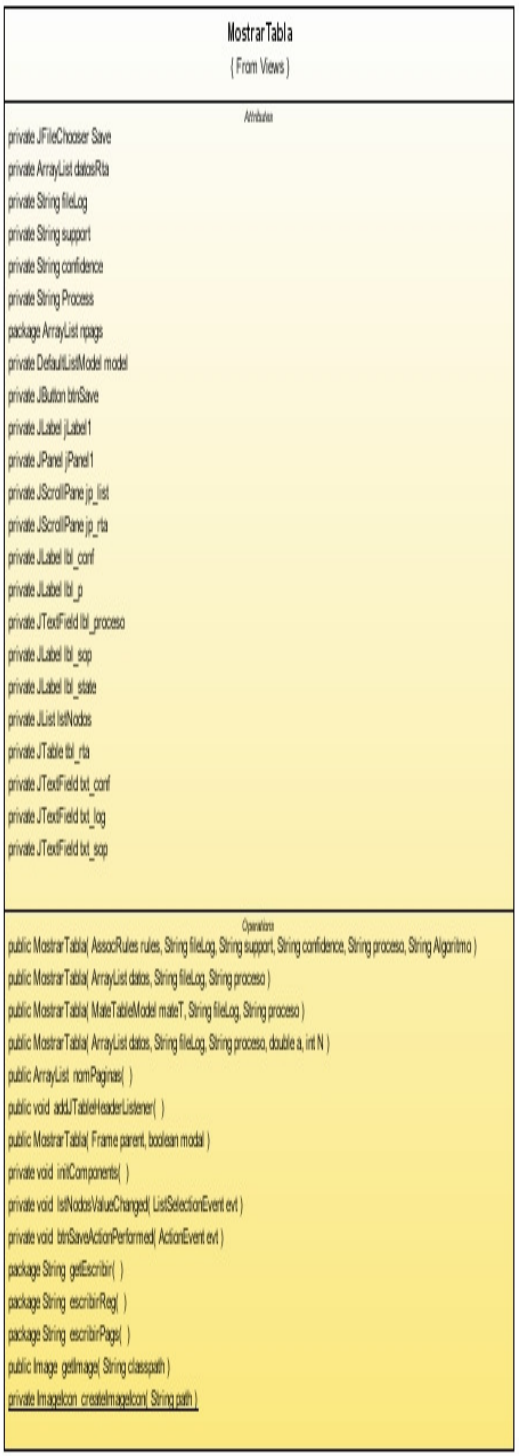
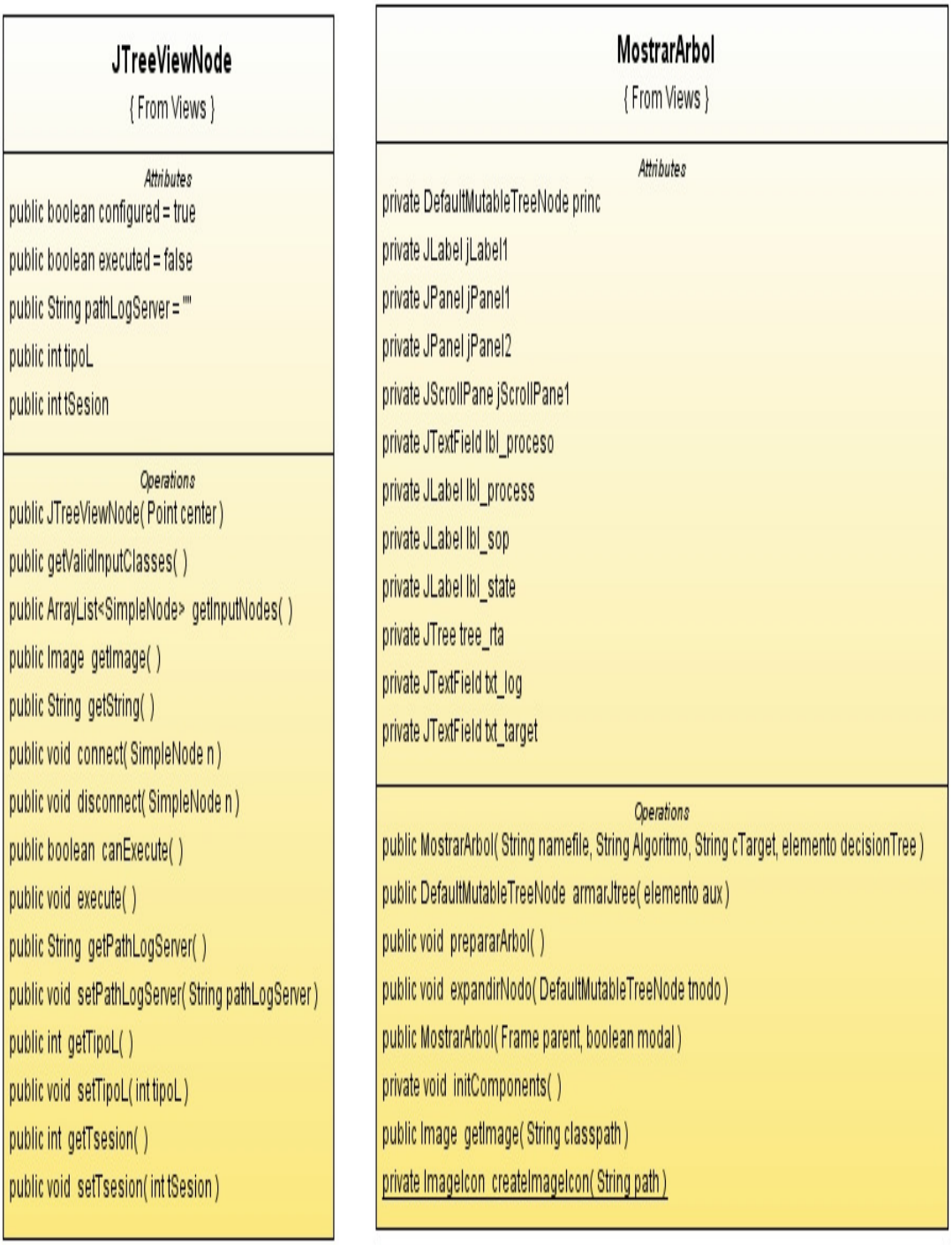
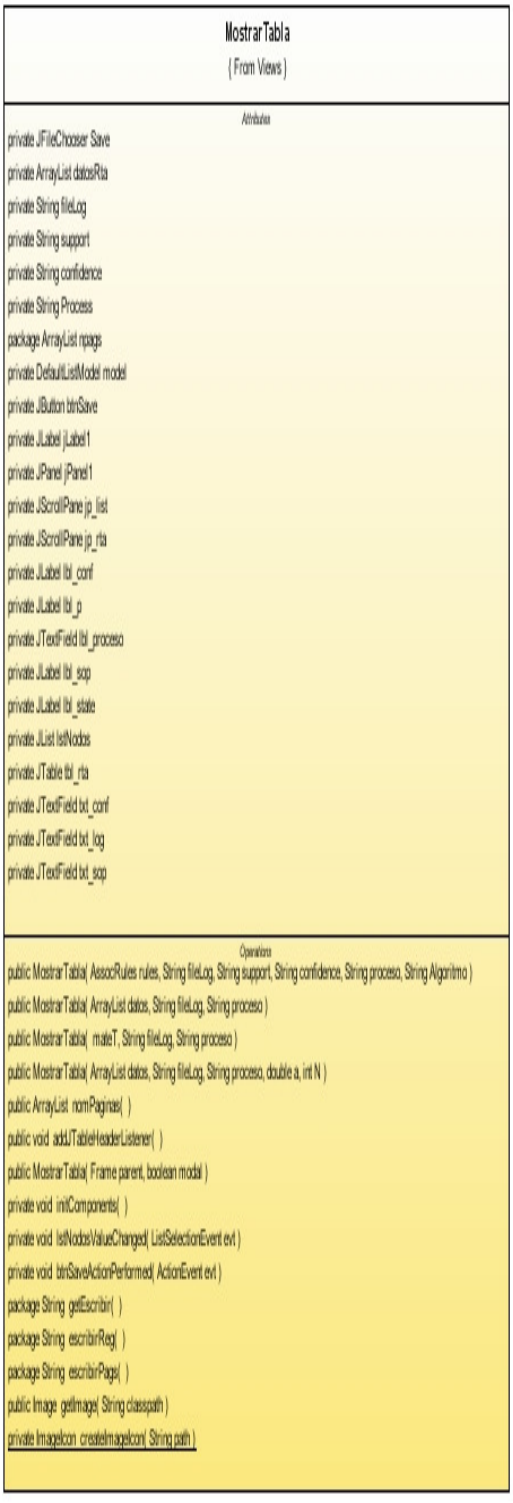

FrmEx
{ From Source }
Attributes
package DefaultListModel modeloFrom = new javax.swing.DefaultListModel()
package DefaultListModel modeloTo = new javax.swing.DefaultListModel()
public int vectorLim
public \Vector vector
package \Vector extOrigen
private JButton btnAccept
private JButton btnAdd
private JButton btnCancel
private JButton btnRemove
private JLabel jLabel1
private JLabel jLabel2
private JPanel jPanel1
private JScrollPane jScrollPane1
private JScrollPane jScrollPane2
private JLabel lblWarning
private JList listDestino
private JList listOrigen
Operations
public FrmEx(Frame parent, boolean modal, \Vector extensiones)
private void initComponents( )
private void listDestinoMouseClicked(MouseEvent evt)
private void listOrigenMouseClicked(MouseEvent evt)
public Image getImage(String classpath)
private ImageIcon createImageIcon(String path)
private void btnAcceptActionPerformed(ActionEvent evt)
private void btnCancelActionPerformed(ActionEvent evt)
private void btnRemoveActionPerformed(ActionEvent evt)
private void btnAddActionPerformed(ActionEvent evt)
public \Vector get\Vector( )
public void set\Vector(\Vector vector)
public int get\VectorLim( )
public void set\VectorLim(int vectorLim)
te
FrmSourceSelect
{ From Source }
Attributes
private JButton btnCancel
private JButton btnExt
private JButton btnOk
private JButton btnOpen
private JLabel lbl_state
Operations
public FrmSourceSelect(Frame parent, SelectSourceNode nodo)
private void initComponents( )
private void btnOkActionPerformed(ActionEvent evt)
public Image getImage(String classpath)
private ImageIcon createImageIcon(String path)
private void btnExtActionPerformed(ActionEvent evt)
private void btnCancelActionPerformed(ActionEvent evt)
private void btnOpenActionPerformed(ActionEvent evt)

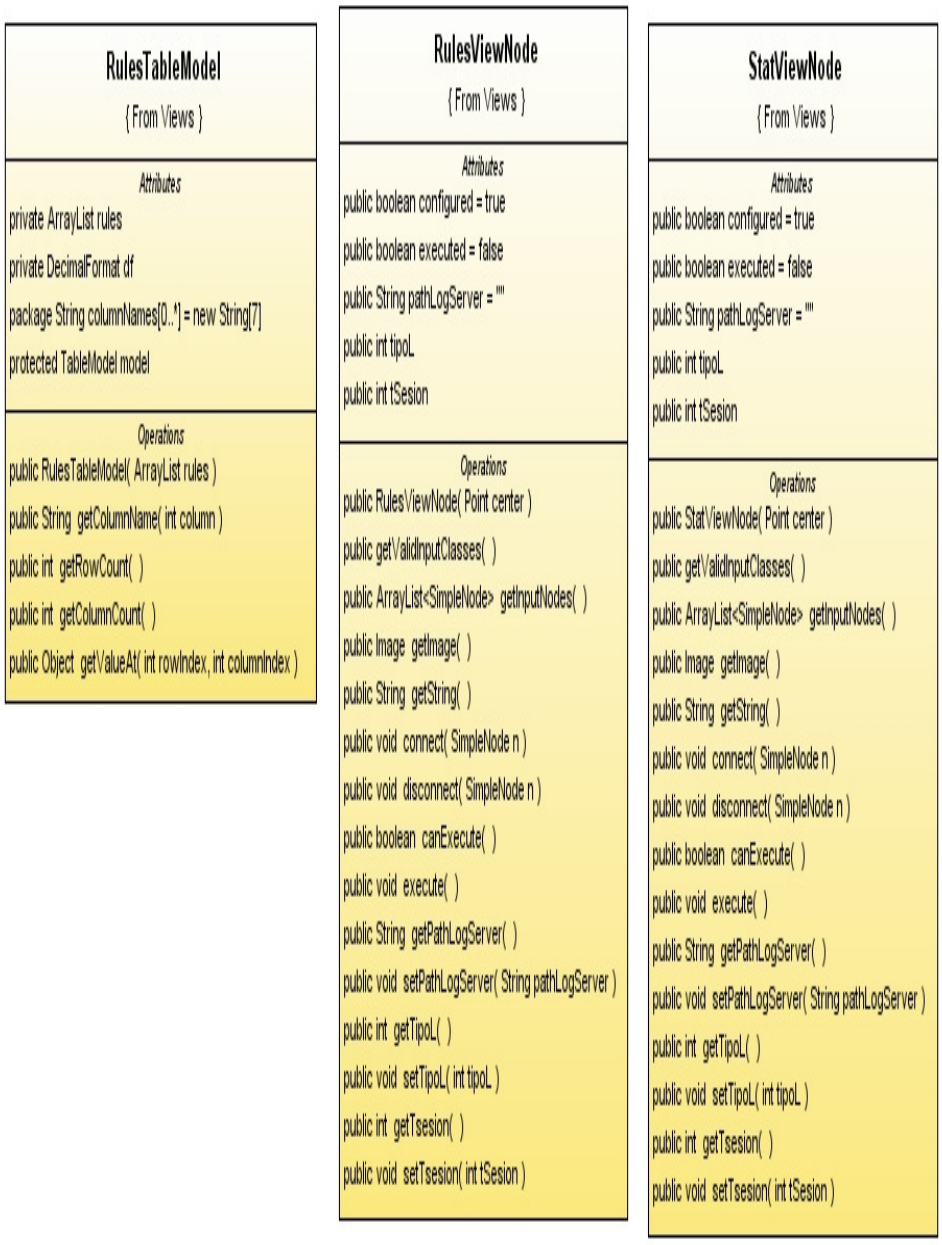
Figura 2.47. Paquete Workspace

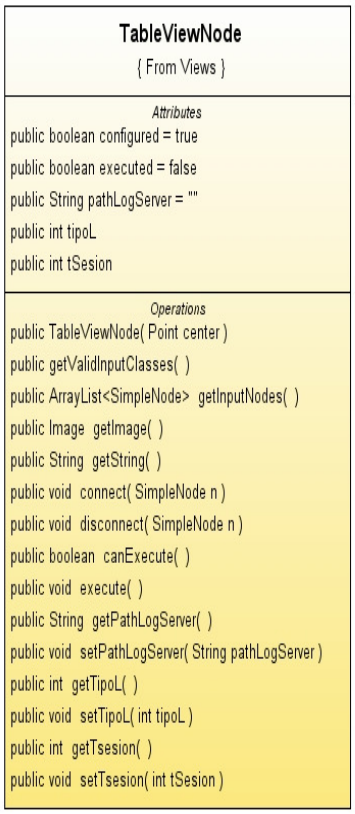


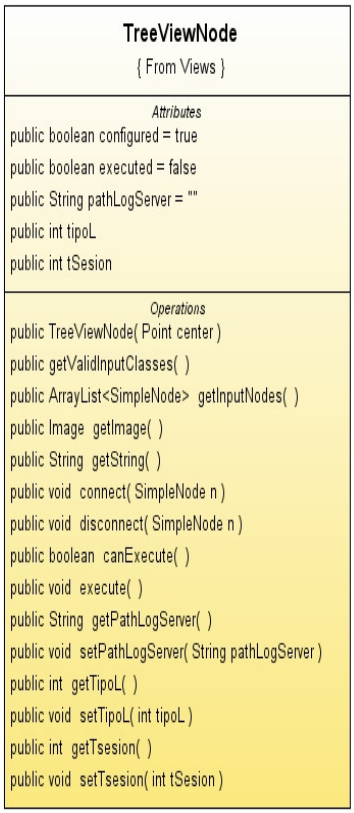

MostrarTabla
(From Views)
Attributes
private JFileChooser Save
private ArrayList datosRta
private String fileLog
private String support
private String confidence
private String Process
package ArrayList npags
private DefaultListModel model
private JButton btnSave
private JLabel jLabel1
private JPanel jPanel1
private JScrollPane jp_list
private JScrollPane jp_rta
private JLabel lbl_conf
private JLabel lbl_p
private JTextField lbl_proceso
private JLabel lbl_sop
private JLabel lbl_state
private JList lstNodos
private JTable tbl_rta
private JTextField txt_conf
private JTextField txt_log
private JTextField txt_sop
Operations
public MostrarTabla(AssocRules rules, String fileLog, String support, String confidence, String proceso, String Algoritmo)
public MostrarTabla(ArrayList datos, String fileLog, String proceso)
public MostrarTabla(MateTableModel mateT, String fileLog, String proceso)
public MostrarTabla(ArrayList datos, String fileLog, String proceso, double a, int N)
public ArrayList numPaginas( )
public void addJTableHeaderListener( )
public MostrarTabla(Frame parent, boolean modal)
private void initComponents( )
private void lstNodosValueChanged(ListSelectionEvent evt)
private void btnSaveActionPerformed(ActionEvent evt)
package String getEscribir( )
package String escribirReg( )
package String escribirPags( )
public Image getImage(String classpath)
private ImageIcon createImageIcon(String path)


JTreeViewNode
{ From Views }
Attributes
public boolean configured = true
public boolean executed = false
public String pathLogServer = ""
public int tipoL
public int tSesion
Operations
public JTreeViewNode(Point center)
public getValidInputClasses( )
public ArrayList<SimpleNode> getInputNodes( )
public Image getImage( )
public String getString( )
public void connect(SimpleNode n)
public void disconnect(SimpleNode n)
public boolean canExecute( )
public void execute( )
public String getPathLogServer( )
public void setPathLogServer(String pathLogServer)
public int getTipoL( )
public void setTipoL(int tipoL)
public int getTsesion( )
public void setTsesion(int tSesion)
MostrarArbol
{ From Views }
Attributes
private DefaultMutableTreeNode princ
private JLabel jLabel1
private JPanel jPanel1
private JPanel jPanel2
private JScrollPane jScrollPane1
private JTextField lbl_proceso
private JLabel lbl_process
private JLabel lbl_sop
private JLabel lbl_state
private JTree tree_rta
private JTextField txt_log
private JTextField txt_target
Operations
public MostrarArbol(String namefile, String Algoritmo, String cTarget, elemento decisionTree)
public DefaultMutableTreeNode armarJtree(elemento aux)
public void prepararArbol( )
public void expandirNodo(DefaultMutableTreeNode tnodo)
public MostrarArbol(Frame parent, boolean modal)
private void initComponents( )
public Image getImage(String classpath)
private ImageIcon createImageIcon(String path)

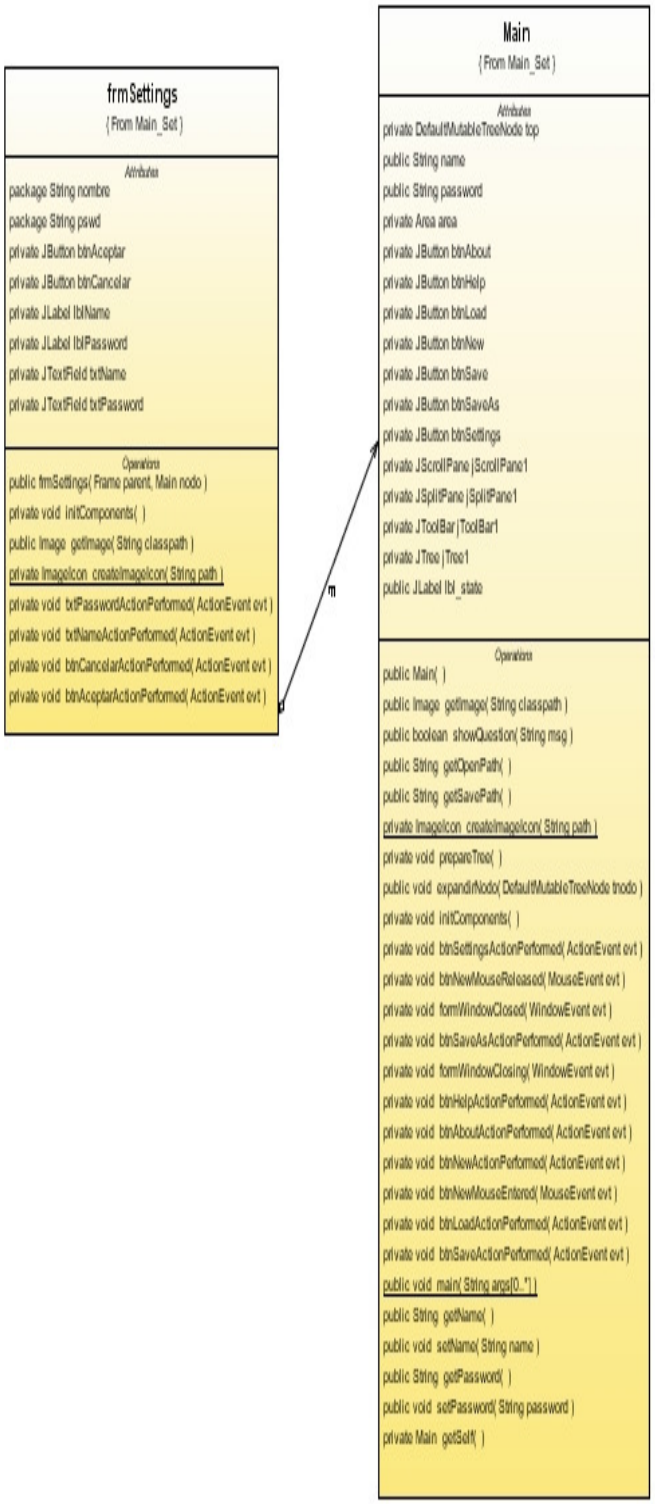

MostrarTabla
{ From Views }
Atributos
private JFileChooser Save
private ArrayList datosRta
private String fileLog
private String support
private String confidence
private String Process
package ArrayList npags
private DefaultListModel model
private JButton btnSave
private JLabel jLabel1
private JPanel jPanel1
private JScrollPane jp_list
private JScrollPane jp_rta
private JLabel lbl_conf
private JLabel lbl_p
private JTextField lbl_proceso
private JLabel lbl_scp
private JLabel lbl_state
private JList lstNodos
private JTable tbl_rta
private JTextField txt_conf
private JTextField txt_log
private JTextField txt_scp
Operations
public MostrarTabla(AssocRules rules, String fileLog, String support, String confidence, String proceso, String Algoritmo)
public MostrarTabla(ArrayList datos, String fileLog, String proceso)
public MostrarTabla(mateT, String fileLog, String proceso)
public MostrarTabla(ArrayList datos, String fileLog, String proceso, double a, int N)
public ArrayList nomPaginas( )
public void addJTableHeaderListener( )
public MostrarTabla(Frame parent, boolean modal)
private void initComponents( )
private void lstNodosValueChanged(ListSelectionEvent evt)
private void btnSaveActionPerformed(ActionEvent evt)
package String getEscribir( )
package String escribirReg( )
package String escribirPags( )
public Image getImage(String classpath)
private ImageIcon createImageIcon(String path)

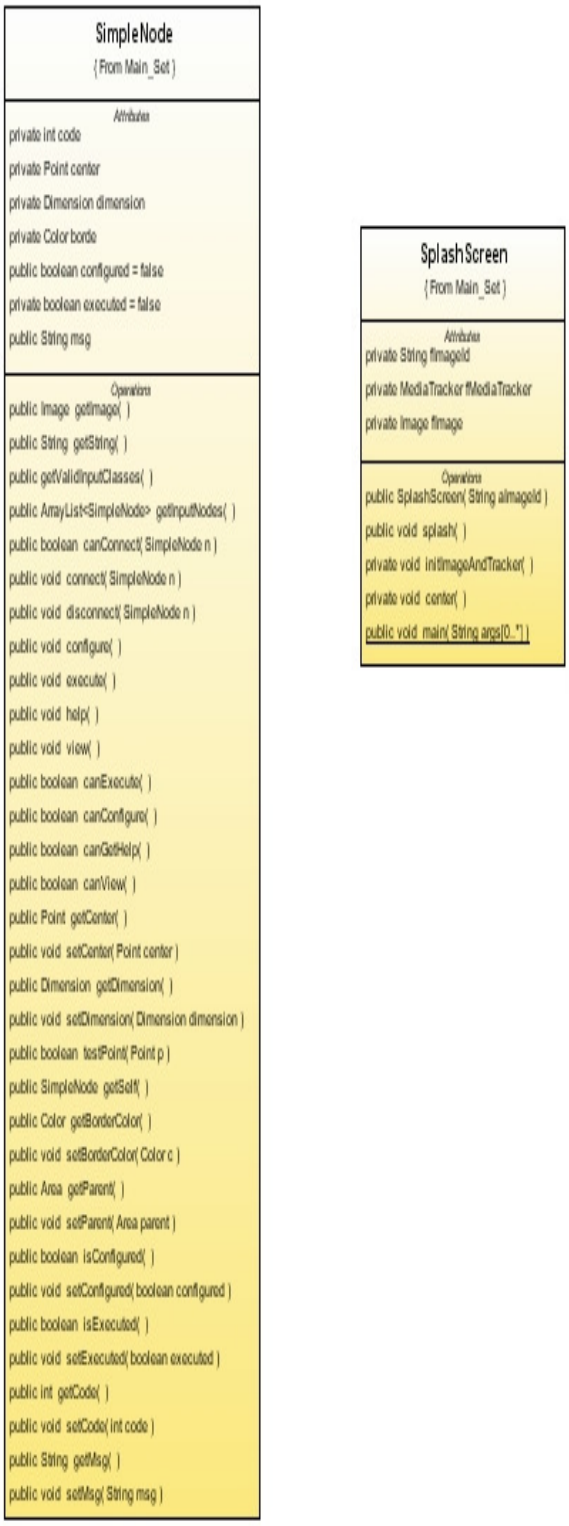

RulesTableModel
{ From Views }
Attributes
private ArrayList rules
private DecimalFormat df
package String columnNames[0..*] = new String[7]
protected TableModel model
Operations
public RulesTableModel(ArrayList rules)
public String getColumnName(int column)
public int getRowCount( )
public int getColumnCount( )
public Object getValueAt(int rowIndex, int columnIndex)
RulesViewNode
{ From Views }
Attributes
public boolean configured = true
public boolean executed = false
public String pathLogServer = ""
public int tipoL
public int tSesion
Operations
public RulesViewNode(Point center)
public getValidInputClasses( )
public ArrayList<SimpleNode> getInputNodes( )
public Image getImage( )
public String getString( )
public void connect(SimpleNode n)
public void disconnect(SimpleNode n)
public boolean canExecute( )
public void execute( )
public String getPathLogServer( )
public void setPathLogServer(String pathLogServer)
public int getTipoL( )
public void setTipoL(int tipoL)
public int getTsesion( )
public void setTsesion(int tSesion)
StatViewNode
{ From Views }
Attributes
public boolean configured = true
public boolean executed = false
public String pathLogServer = ""
public int tipoL
public int tSesion
Operations
public StatViewNode(Point center)
public getValidInputClasses( )
public ArrayList<SimpleNode> getInputNodes( )
public Image getImage( )
public String getString( )
public void connect(SimpleNode n)
public void disconnect(SimpleNode n)
public boolean canExecute( )
public void execute( )
public String getPathLogServer( )
public void setPathLogServer(String pathLogServer)
public int getTipoL( )
public void setTipoL(int tipoL)
public int getTsesion( )
public void setTsesion(int tSesion)


TableViewNode
{ From Views }
Attributes
public boolean configured = true
public boolean executed = false
public String pathLogServer = ""
public int tipoL
public int tSesion
Operations
public TableViewNode(Point center)
public getValidInputClasses( )
public ArrayList<SimpleNode> getInputNodes( )
public Image getImage( )
public String getString( )
public void connect(SimpleNode n)
public void disconnect(SimpleNode n)
public boolean canExecute( )
public void execute( )
public String getPathLogServer( )
public void setPathLogServer(String pathLogServer)
public int getTipoL( )
public void setTipoL(int tipoL)
public int getTsesion( )
public void setTsesion(int tSesion)


TreeViewNode
{ From Views }
Attributes
public boolean configured = true
public boolean executed = false
public String pathLogServer = ""
public int tipoL
public int tSesion
Operations
public TreeViewNode(Point center)
public getValidInputClasses( )
public ArrayList<SimpleNode> getInputNodes( )
public Image getImage( )
public String getString( )
public void connect(SimpleNode n)
public void disconnect(SimpleNode n)
public boolean canExecute( )
public void execute( )
public String getPathLogServer( )
public void setPathLogServer(String pathLogServer)
public int getTipoL( )
public void setTipoL(int tipoL)
public int getTsesion( )
public void setTsesion(int tSesion)


Area
{ From Main_Set }
Attributes
private int curMode = 1
private int SELECTNODE = 1
private int MOVENODE = 2
private int ADDSOURCENODE = 3
private String path
private boolean saved
public JLabel lbl_state
private boolean banArrastre = false
public String statusMsg
Operations
public void reset( )
public Area( )
public boolean showConfirm(String msg)
public void showError(String msg)
public void setCursor(int cursorType)
public void mouseClick( )
public SimpleNode testPoint( )
public void mostrarPopup( )
public void addNodo(SimpleNode n)
public void paint(Graphics g)
public List<SimpleNode> getNodos( )
public void setNodos(List<SimpleNode> nodos)
public String getPath( )
public void setPath(String path)
public boolean isSaved( )
public void setSaved(boolean saved)
public void loadFromFile( )
public void saveToFile( )
private SimpleNode getElementById(List<SimpleNode> nodos, int id
public boolean isBanArrastre( )
public void setBanArrastre(boolean banArrastre)
public String getStatusMsg( )
public void setBarra(JLabel lbl_stat)
public void setStatusMsg(String statusMsg)

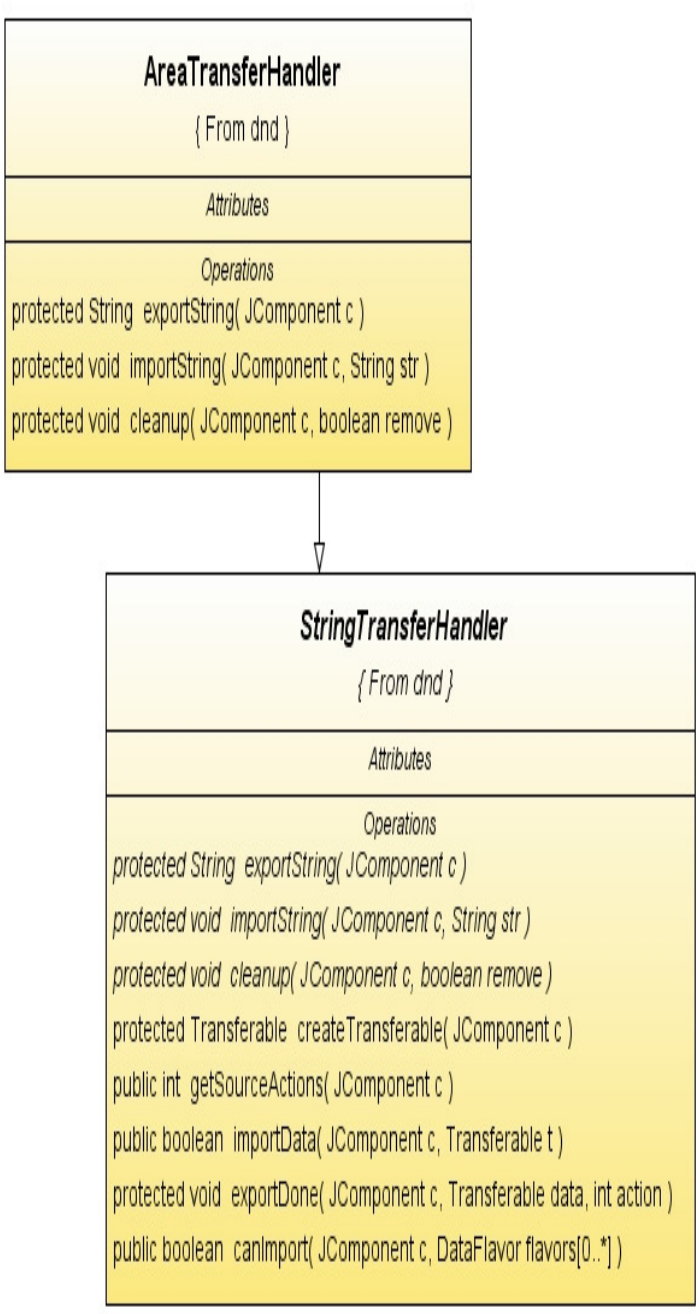

frmSettings
{ From Main_Set }
Attributes
package String nombre
package String pswd
private JButton btnAceptar
private JButton btnCancelar
private JLabel lblName
private JLabel lblPassword
private JTextField txtName
private JTextField txtPassword
Operations
public frmSettings(Frame parent, Main nodo)
private void initComponents( )
public Image getImage(String classpath)
private ImageIcon createImageIcon(String path)
private void txtPasswordActionPerformed(ActionEvent evt)
private void txtNameActionPerformed(ActionEvent evt)
private void btnCancelarActionPerformed(ActionEvent evt)
private void btnAceptarActionPerformed(ActionEvent evt)
m
Main
{ From Main_Set }
Attributes
private DefaultMutableTreeNode top
public String name
public String password
private Area area
private JButton btnAbout
private JButton btnHelp
private JButton btnLoad
private JButton btnNew
private JButton btnSave
private JButton btnSaveAs
private JButton btnSettings
private JScrollPane jScrollPane1
private JSplitPane jSplitPane1
private JToolBar jToolBar1
private JTree jTree1
public JLabel lbl_state
Operations
public Main( )
public Image getImage(String classpath)
public boolean showQuestion(String msg)
public String getOpenPath( )
public String getSavePath( )
private ImageIcon createImageIcon(String path)
private void prepareTree( )
public void expandirNodo(DefaultMutableTreeNode tnodo)
private void initComponents( )
private void btnSettingsActionPerformed(ActionEvent evt)
private void btnNewMouseReleased(MouseEvent evt)
private void formWindowClosed(WindowEvent evt)
private void btnSaveAsActionPerformed(ActionEvent evt)
private void formWindowClosing(WindowEvent evt)
private void btnHelpActionPerformed(ActionEvent evt)
private void btnAboutActionPerformed(ActionEvent evt)
private void btnNewActionPerformed(ActionEvent evt)
private void btnNewMouseEntered(MouseEvent evt)
private void btnLoadActionPerformed(ActionEvent evt)
private void btnSaveActionPerformed(ActionEvent evt)
public void main(String args[0..*])
public String getName( )
public void setName(String name)
public String getPassword( )
public void setPassword(String password)
private Main getSelf( )


SimpleNode
{ From Main_Sol }
Attributes
private int code
private Point center
private Dimension dimension
private Color borde
public boolean configured = false
private boolean executed = false
public String msg
Operations
public Image getImage( )
public String getString( )
public getValidInputClasses( )
public ArrayList<SimpleNode> getInputNodes( )
public boolean canConnect(SimpleNode n)
public void connect(SimpleNode n)
public void disconnect(SimpleNode n)
public void configure( )
public void execute( )
public void help( )
public void view( )
public boolean canExecute( )
public boolean canConfigure( )
public boolean canGetHelp( )
public boolean canView( )
public Point getCenter( )
public void setCenter(Point center)
public Dimension getDimension( )
public void setDimension(Dimension dimension)
public boolean testPoint(Point p)
public SimpleNode getSelf( )
public Color getBorderColor( )
public void setBorderColor(Color c)
public Area getParent( )
public void setParent(Area parent)
public boolean isConfigured( )
public void setConfigured(boolean configured)
public boolean isExecuted( )
public void setExecuted(boolean executed)
public int getCode( )
public void setCode(int code)
public String getMsg( )
public void setMsg(String msg)
SplashScreen
{ From Main_Sol }
Attributes
private String fImageId
private MediaTracker fMediaTracker
private Image fImage
Operations
public SplashScreen(String aImageId)
public void splash( )
private void initImageAndTracker( )
private void center( )
public void main(String args[0..*])


Area
{ From Main_Set }
Attributes
private int curMode = 1
private int SELECTNODE = 1
private int MOVEMODE = 2
private int ADDSOURCEMODE = 3
private String path
private boolean saved
public JLabel lbl_state
private boolean banArrastre = false
public String statusMsg
Operations
public void reset( )
public Area( )
public boolean showConfirm(String msg)
public void showError(String msg)
public void setCursor(int cursorType)
public void mouseClick( )
public SimpleNode testPoint( )
public void mostrarPopup( )
public void addNodo(SimpleNode n)
public void paint(Graphics g)
public List<SimpleNode> getNodos( )
public void setNodos(List<SimpleNode> nodos)
public String getPath( )
public void setPath(String path)
public boolean isSaved( )
public void setSaved(boolean saved)
public void loadFromFile( )
public void saveToFile( )
private SimpleNode getElementById(List<SimpleNode> nodos, int id
public boolean isBanArrastre( )
public void setBanArrastre(boolean banArrastre)
public String getStatusMsg( )
public void setBarra(JLabel lbl_stat)
public void setStatusMsg(String statusMsg)


frmSettings
{ From Main_Set }
Attributes
package String nombre
package String pswd
private JButton btnAceptar
private JButton btnCancelar
private JLabel lblName
private JLabel lblPassword
private JTextField txtName
private JTextField txtPassword
Operations
public frmSettings(Frame parent, Main nodo)
private void initComponents( )
public Image getImage(String classpath)
private ImageIcon createImageIcon(String path)
private void txtPasswordActionPerformed(ActionEvent ev
private void txtNameActionPerformed(ActionEvent evt)
private void btnCancelarActionPerformed(ActionEvent ev
private void btnAceptarActionPerformed(ActionEvent evt

Figura 2.48. Paquete dnd



6.1.4. Diagramas de Secuencia

El diseño de los diagramas de secuencia de la herramienta Polaris se muestran en las figura 2.49 hasta la figura 2.57.

Clase A priori

Figura 2.49: run

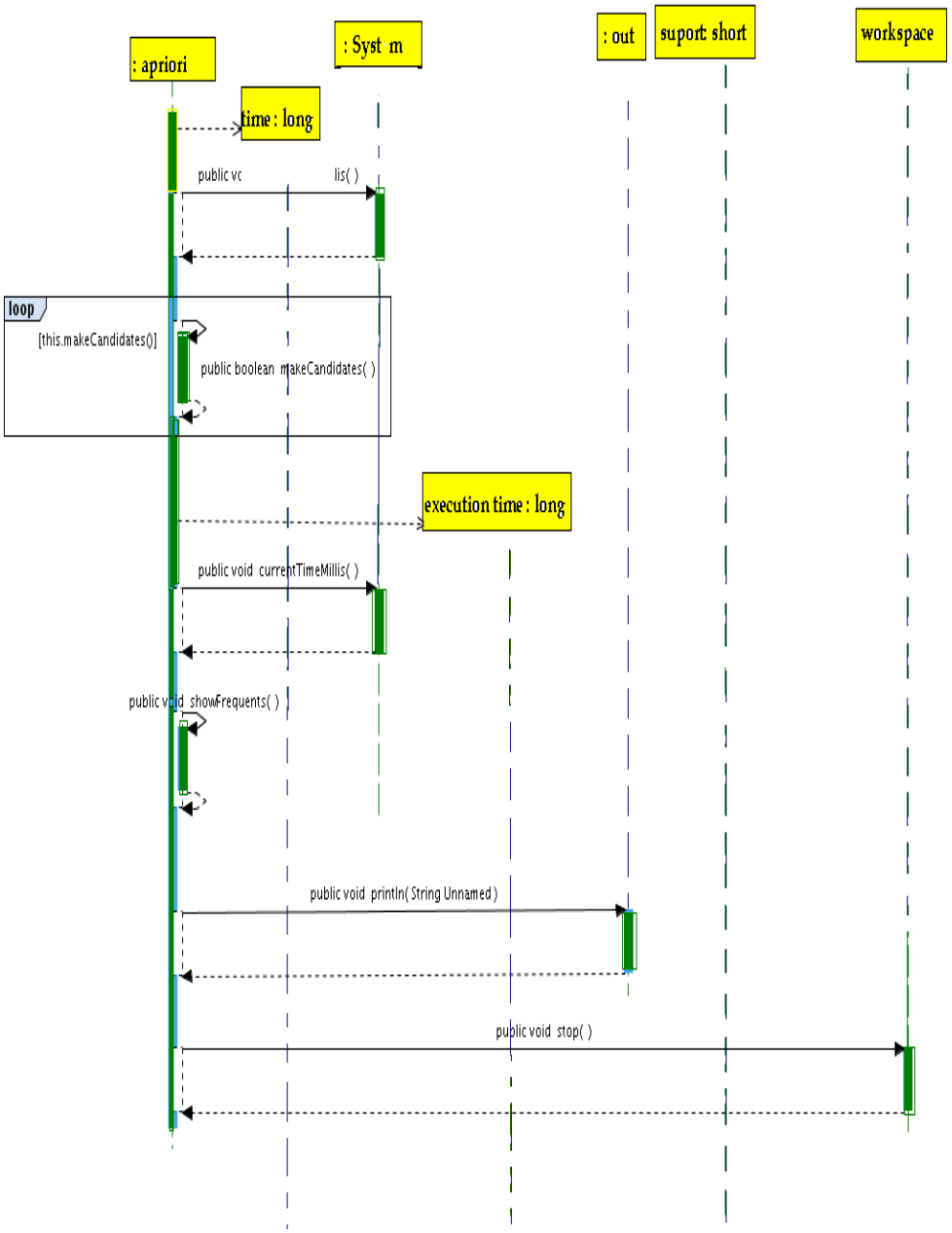


Figura 2.50: makeCandidates

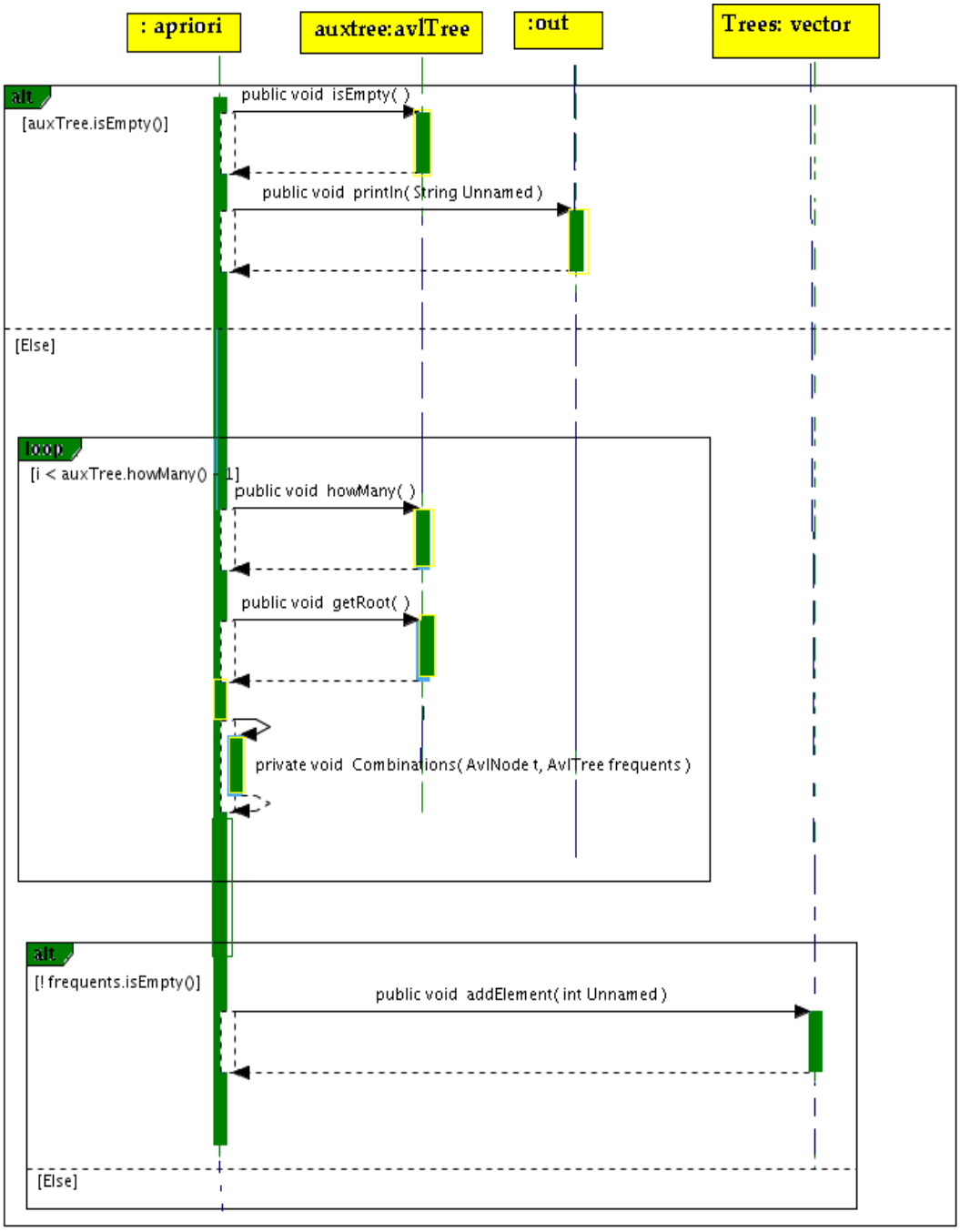
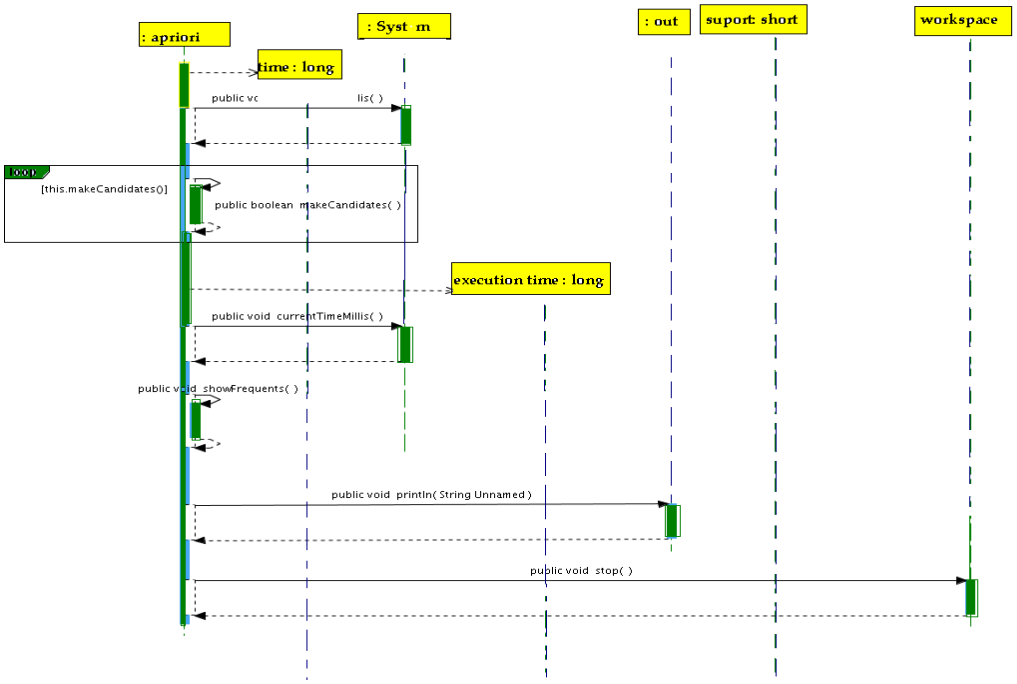
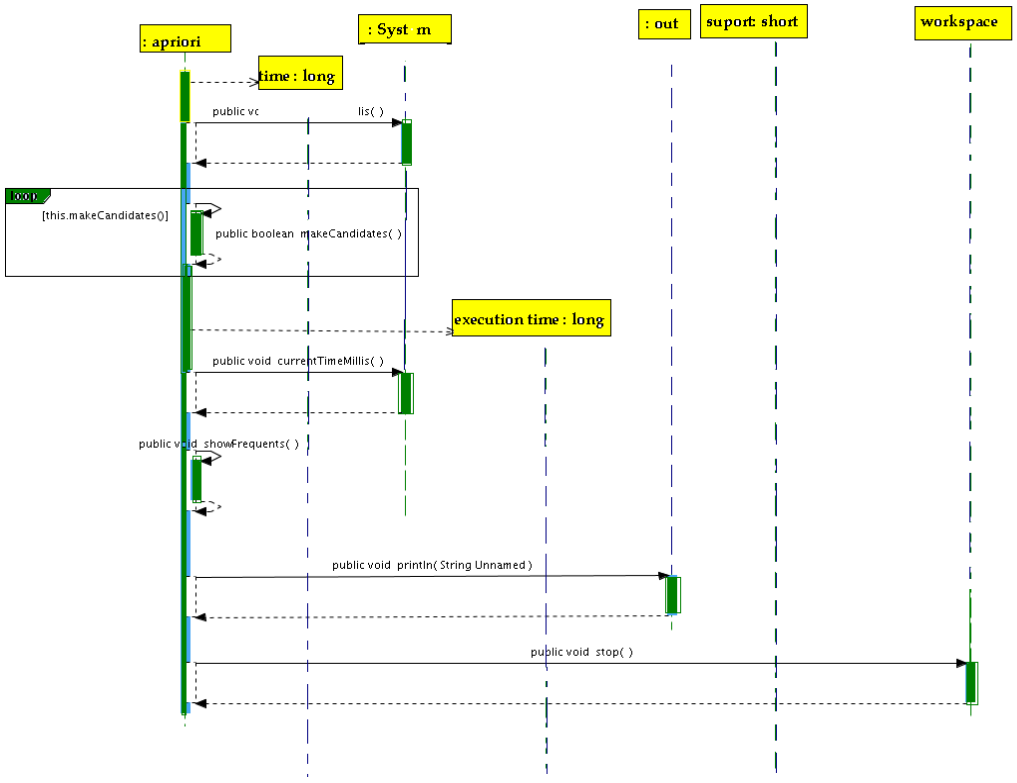


Figura 2.51: findInDataSet.



Clase DataSet

Figura 2.52. buildNTree



Clase FPGrowth

Figura 2.53: buildFrequentsNodes.

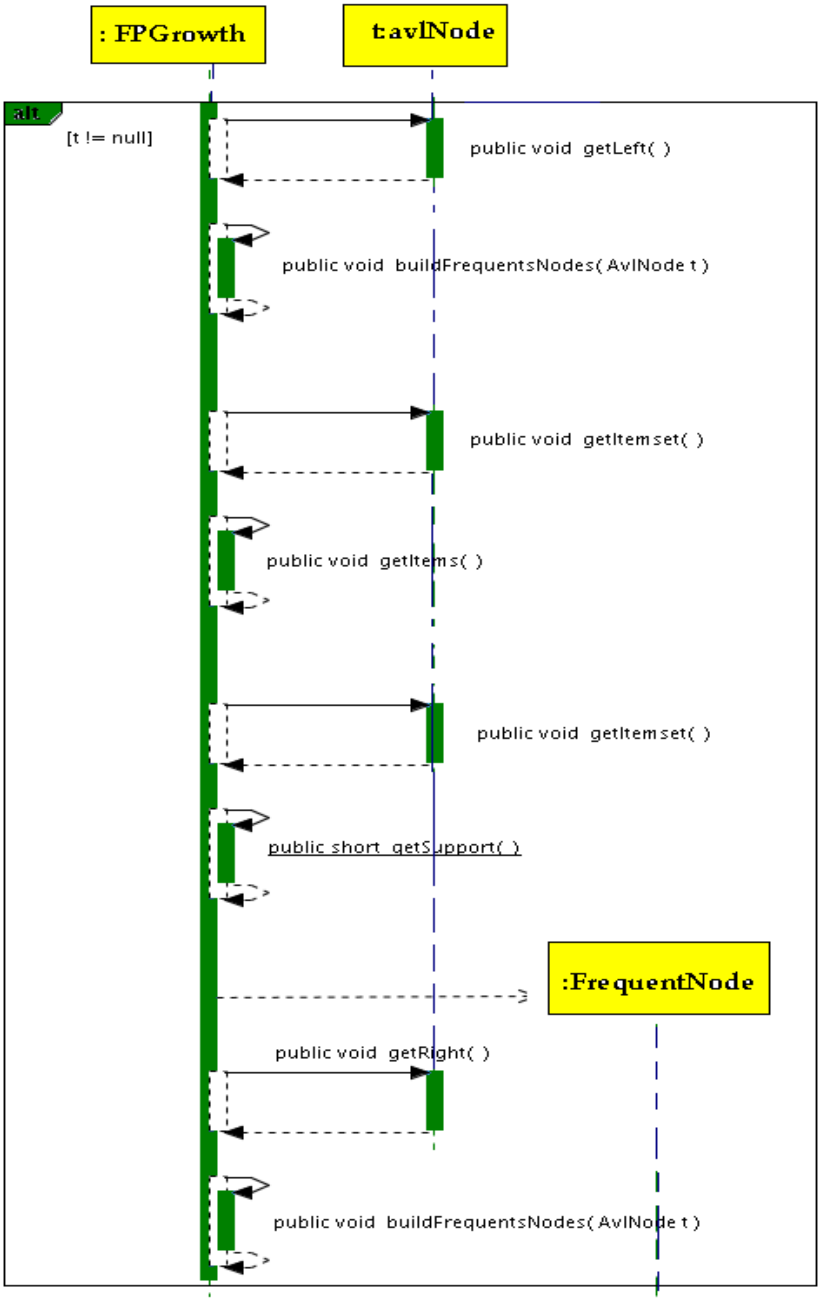
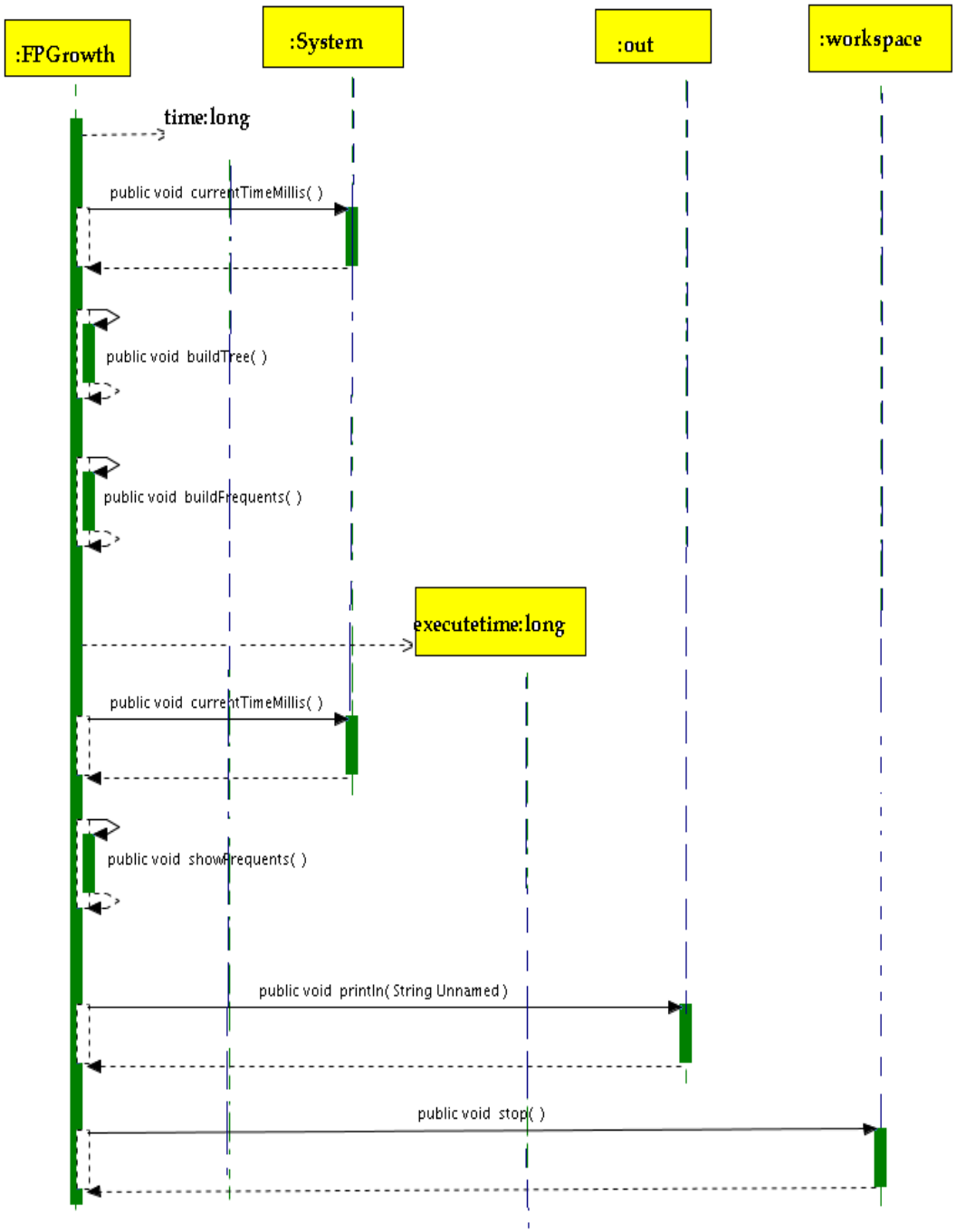
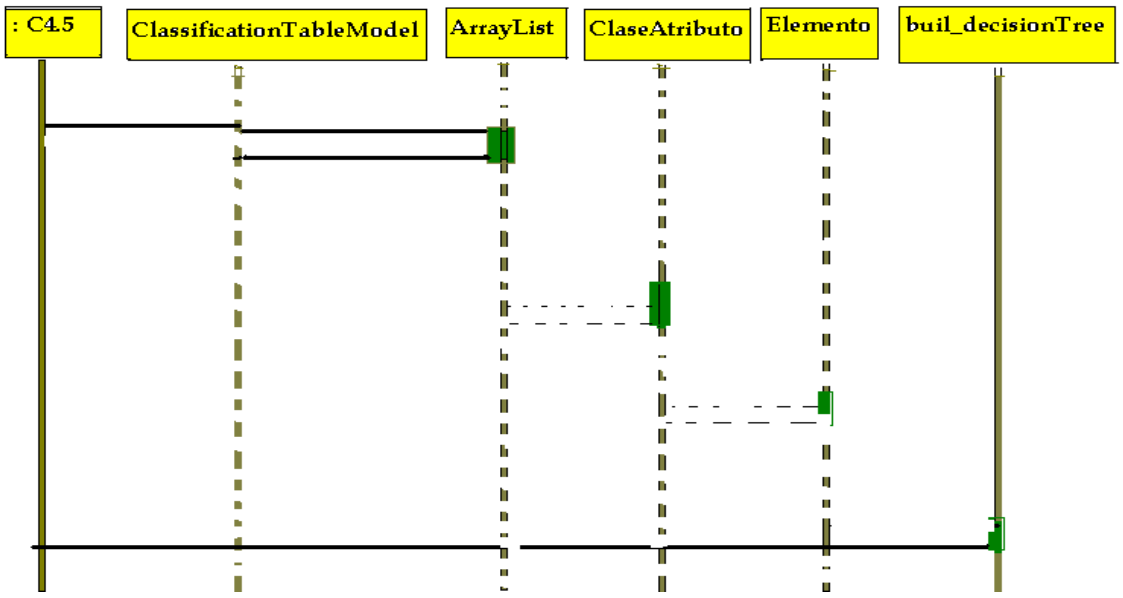


Figura 2.54: run



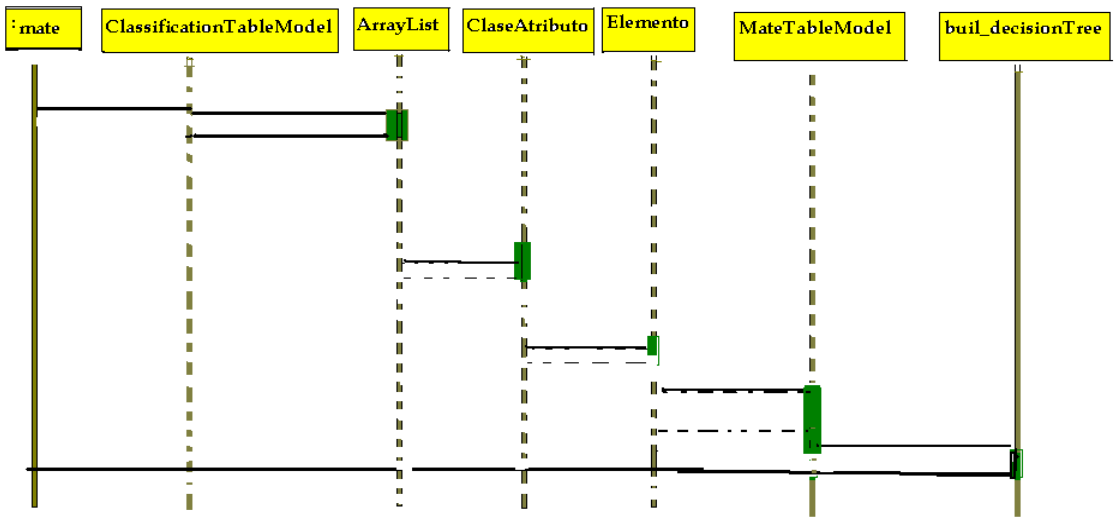
Clase C4.5

Figura 2.55: c4.5



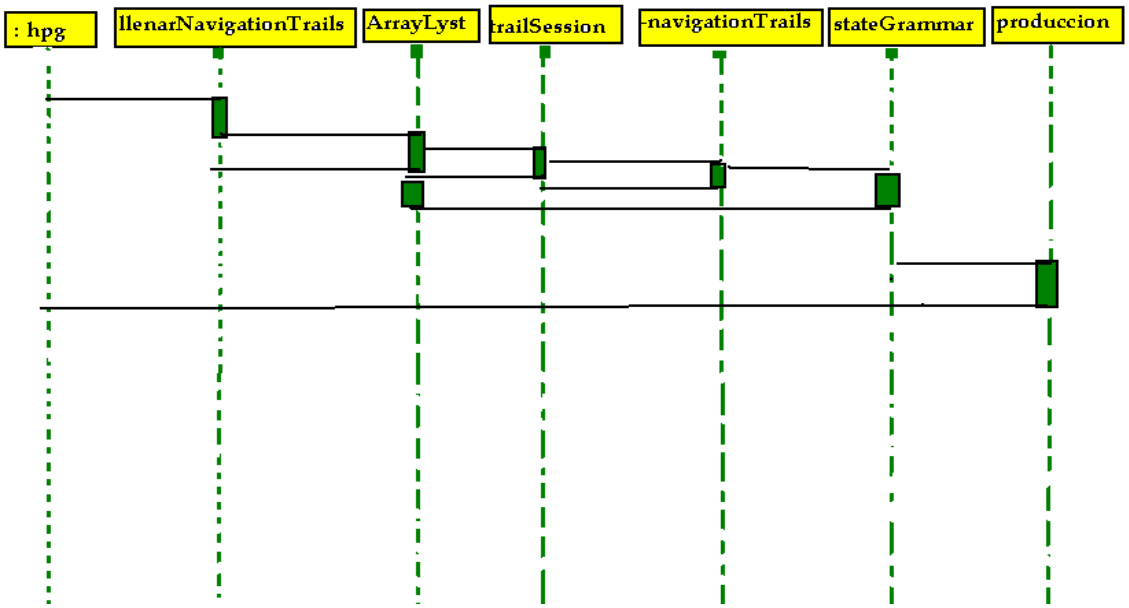
Clase Mate

Figura 2.56: mate



Clase HPG

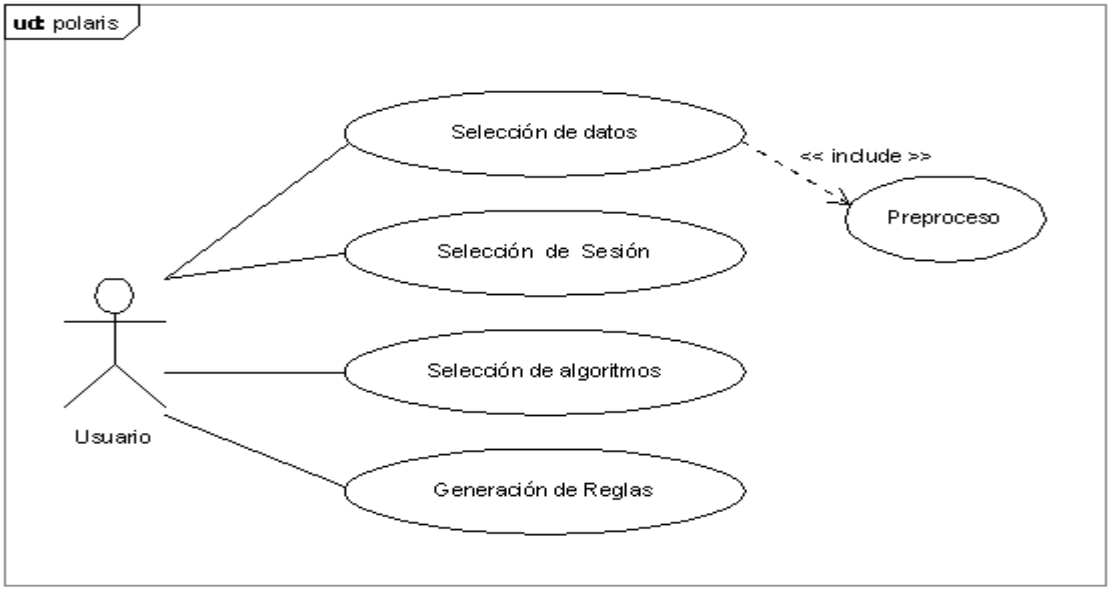
Figura 2.57: hpg



6.1.5. Diagramas de Casos de Uso

El diseño de los diagramas de casos de uso de la herramienta Polaris se muestran en las figura 2.56 hasta la figura 2.61.

Figura 2.56. Polaris


uc: polaris
Selección de datos
<< include >>
Preproceso
Selección de Sesión
Selección de algoritmos
Generación de Reglas
Usuario

Fi

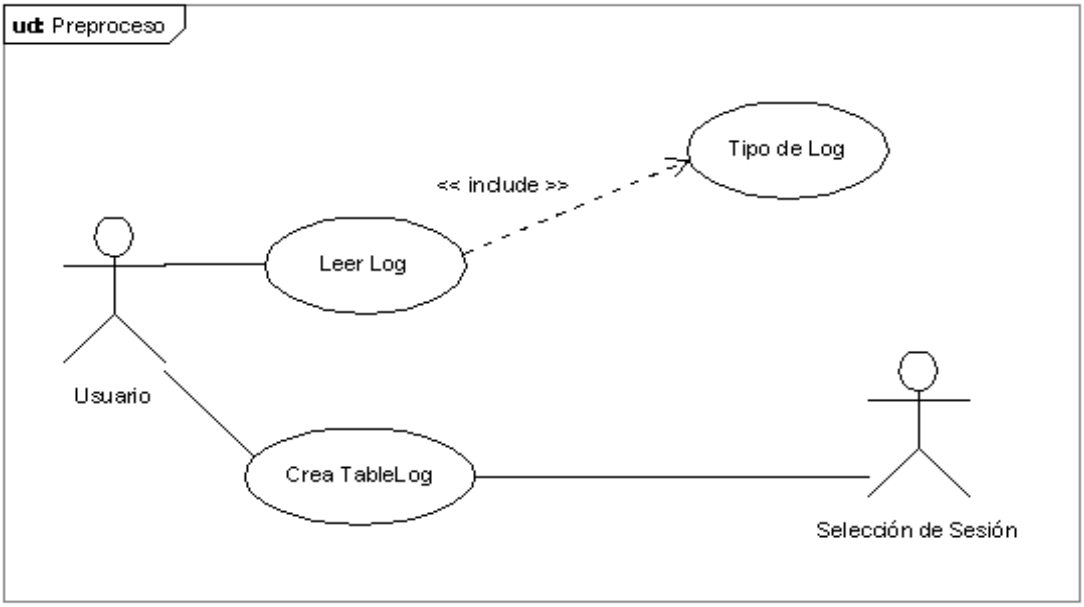

uc: Preproceso
Tipo de Log
<< include >>
Leer Log
Usuario
Crea TableLog
Selección de Sesión

Figura 2.58. Selección de Datos

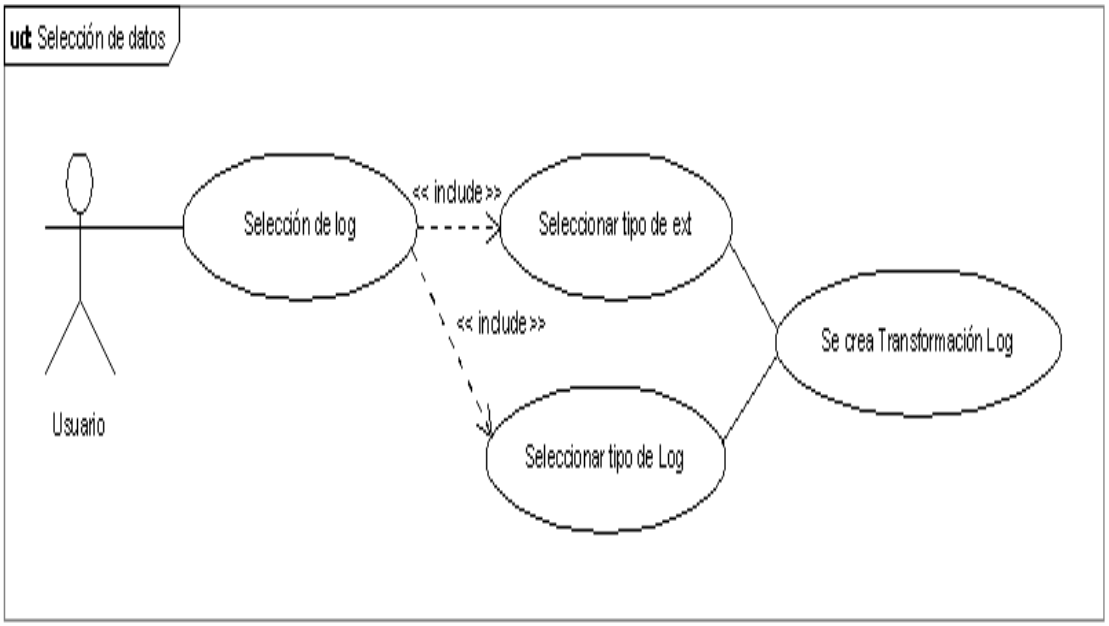


Figura 2.59. Selección de Sesión

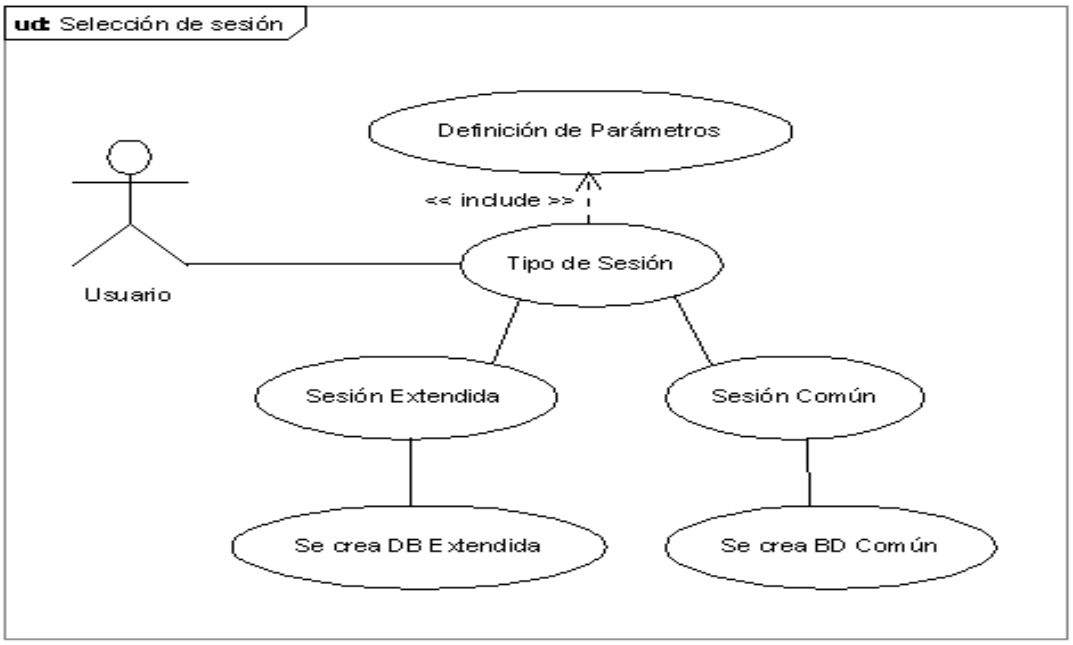


Figura 2.60. Selección de Algoritmos

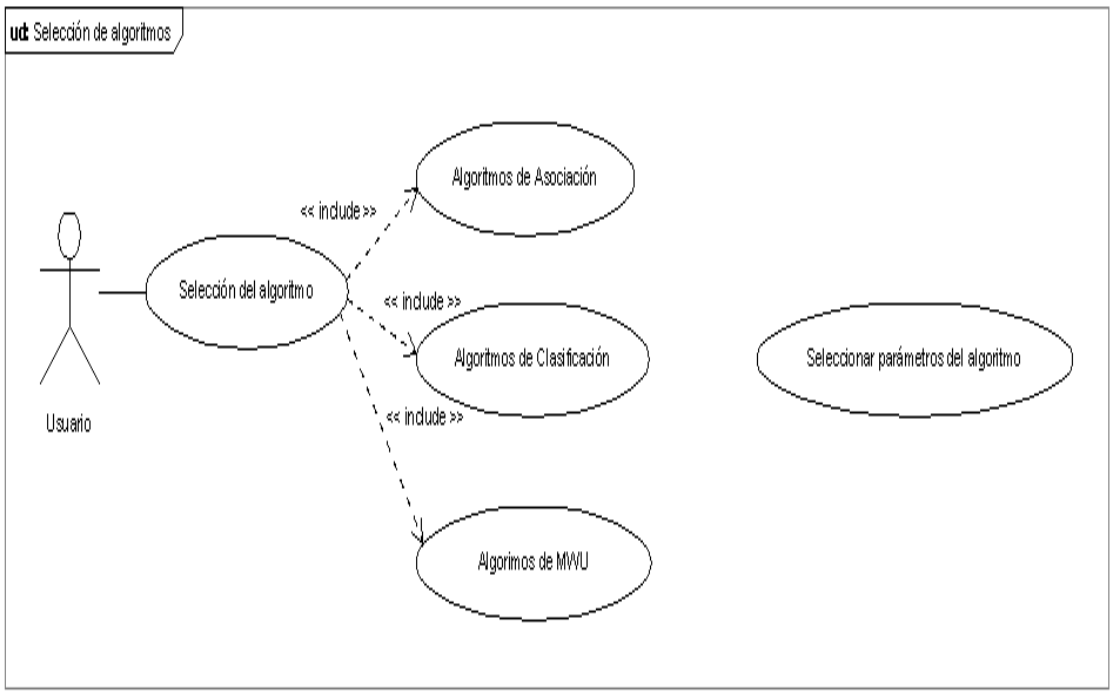
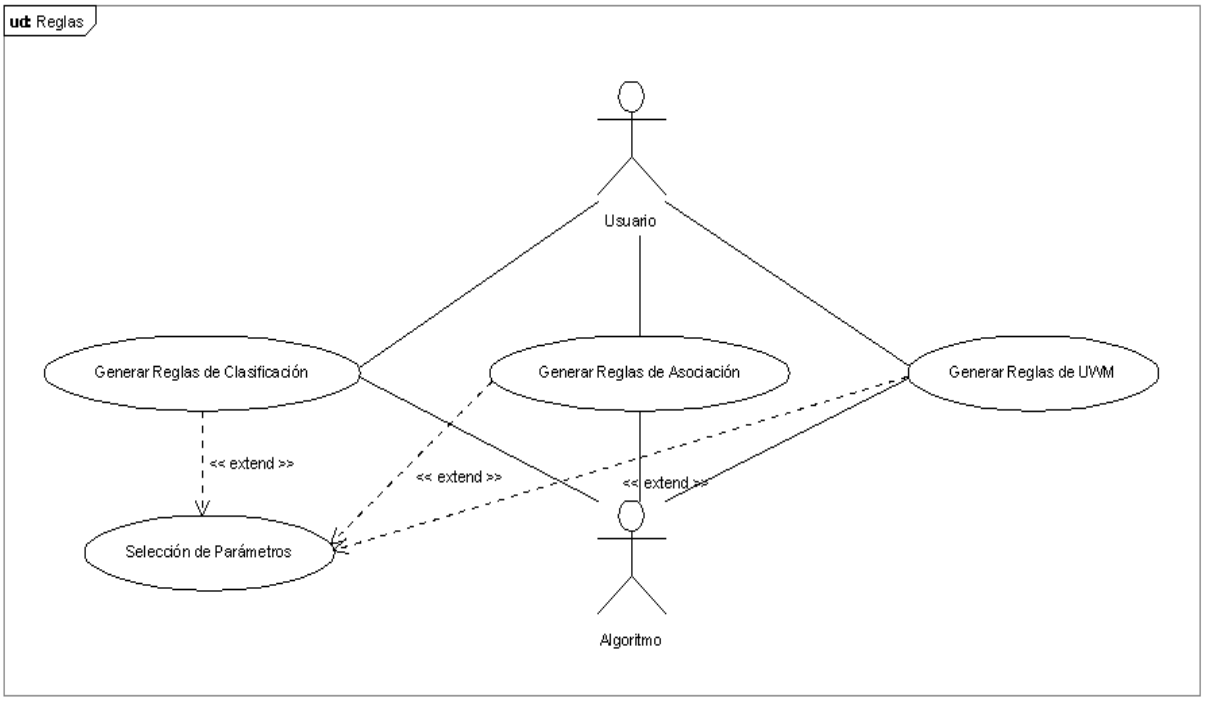


Figura 2.61. Reglas



7. IMPLEMENTACIÓN

7.1 INTRODUCCIÓN

La implementación de la herramienta software Polaris se realizó sobre:

- **Sistema Operativo**: *Fedora Core*™ en su versiones 6.0.
- **Lenguaje de programación:** Java™ 6.0.
- **Plataforma de Desarrollo**: NetBeans™ 5.5, como IDE de desarrollo.

7.2 ARQUITECTURA

Los módulos de software de la herramienta Polaris son:

Módulo de utilidades: Este módulo es el encargado de dos tareas principales:

- Realizar la conexión a la base de datos y poder acceder a la fuente de datos que repose en el disco duro.

- Contener la colección de clases principales y bibliotecas que son utilizadas por otras clases en la manipulación y visualización de datos, de esta manera puede hacerse la administración de las mismas y se hace factible la reutilización del código.

Módulo del kernel Polaris: En este módulo se encuentran los módulos de Prepocesamiento y Algoritmos, necesarios para poder realizar el proceso de descubrimiento de conocimiento paquetes primordiales para la ejecución de las tareas de descubrimiento de conocimiento. Dicha estructura se puede apreciar en la figura 2.62.

- El módulo de Prepocesamiento: Llamado Premining, contiene todas las rutinas necesarias para poder realizar la limpieza y transformación de los datos de entrada; es decir, en él se realiza el tratamiento preliminar de los datos para posteriormente aplicar las técnicas y los algoritmos de minería de datos respectivos.

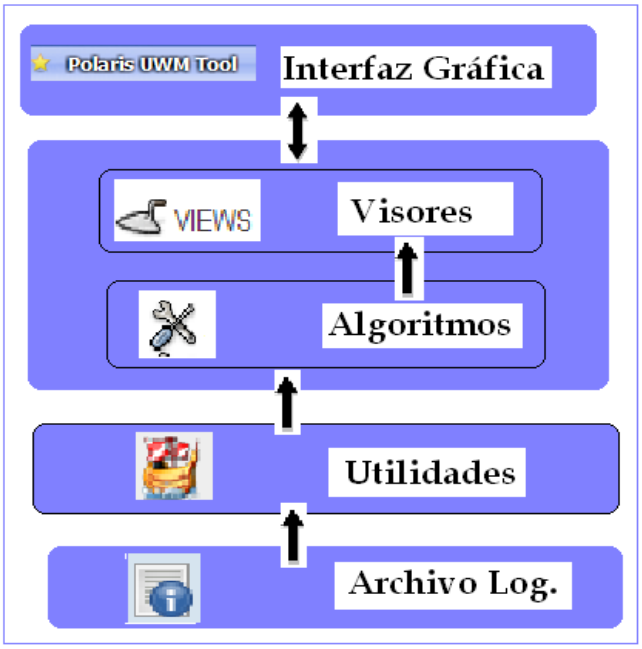
- El módulo de Algoritmos: Llamado Algorithm, contiene las clases necesarias para la aplicación de las técnicas y algoritmos de minería de datos respectivos para las tareas de asociación y clasificación.

Módulo de interfaz gráfica: Este módulo contiene los módulos de visores y el módulo da soporte visual a los demás módulos.

- El módulo de visores: Contiene las clases necesarias para construir y desplegar las estructuras para la visualización gráfica y dinámica de los resultados obtenidos después de la aplicación de las diferentes técnicas y algoritmos de minería de datos.

- El módulo de soporte visual: En el se encuentran propiamente las herramientas necesarios para que el usuario pueda interactuar de una forma fácil y agradable con los diferentes componentes de la misma.

Figura 2.62. Arquitectura Polaris



7.3. ESTRUCTURA DE PAQUETES DE APLICACIÓN

La estructura general de los paquetes de la herramienta software Polaris es la siguiente:

Paquete Prepro. Implementa todas las clases necesarias para realizar un preprocesamiento o tratamiento previo a los datos de entrada.

Paquete Utils. En él se implementan las clases necesarias para realizar la entrada de los datos para la aplicación de los respectivos algoritmos de Asociación. Además de lo anterior, en él también se encuentran todas las clases que son utilizadas por las demás clases del proyecto.

Paquete about. Contiene todas las clases necesarias para la interfaz gráfica en java™ 3d.

Paquete algorithm. Está compuesto por los paquetes y subpaquetes respectivos de la siguiente manera:

- Association
  - Apriori
  - Equipasso
  - FPGrowth
- Classification
  - C4.5
  - Mate.
- Hpg

Paquete grammar. Esta compuesto por todas las clases necesarias para la visualización de la gramática, resultado de la aplicación del algoritmo HPG.

Paquete Statistical. Esta compuesto por todas las clases necesarias para la generación y visualización de los reportes estadísticos.

Paquete resource. Este paquete contiene todos los archivos html utilizados en la ayuda o el "acerca de" de la herramienta.

Paquete dnd. Se compone de todas las clases necesarias para la aplicación de la metodología Drag´n Drop (arrastrar y soltar).

Paquete media. Contiene todas las imágenes utilizadas en la interfaz gráfica de la herramienta.

Paquete workspace. Se compone de todas las clases utilizadas en el área de trabajo de la interfaz gráfica de la herramienta.

Paquete treeA. Este paquete contiene las clases necesarias para la visualización del árbol de clasificación.

Cabe destacar en este aspecto que para la aplicación de los algoritmos de Asociación se reutilizó el código de los algoritmos realizados por el grupo de Investigación TARIYKDD [23].

Es por esto que aprovechando las ventajas de la programación orientada a objetos tales como la reusabilidad o la capacidad para que programas que traten las mismas estructuras de información reutilicen las definiciones de objetos y procedimientos empleados en otros programas y de las posibilidades que presenta un software libre; se decidió reutilizar e implementar el código sobre las técnicas y algoritmos para la generación de las reglas de asociación en la herramienta Polaris. Sin embargo, debido al tipo de información que es analizada en Polaris, se desarrollaron e implementaron clases propias de la herramienta.

A continuación se amplia y se detallan conceptos sobre los paquetes más importantes dentro del desarrollo de la herramienta y la manera como fueron implementadas las mismas.

7.3.1. Paquete Prepro. Esta clase se compone por las siguientes clases:

conexión: Se encarga de establecer, administrar y cerrar la conexión a una Base de Datos.

La clase conexión maneja los siguientes métodos:

- conectar(): Establece la conexión con la base de datos teniendo en cuenta el nombre de la misma, el usuario y la contraseña.
- cerrarConexion(): Cierra la conexión con la base de datos.
- consulta(): Para realizar un query o consulta a la base de datos.
- insertar(): Para realizar la inserción de elementos o registros a la base de datos.
- eliminar(): Para eliminar un registro de la base de datos.

- modificar(): Para modificar los registros de la base de datos.

confpre: Esta clase se encarga de realizar la configuración del proceso de preprocesamiento de los datos de entrada.

ejecutarPrepro: Esta clase se encarga de realizar un tratamiento previo o preprocesamiento de los datos de entrada. Contiene los métodos encargados de hacer el respectivo registro de sesión común o extendida, dependiendo del formato del archivo log que se encuentre leyendo.

filtrarDatos: Esta clase se encarga de filtrar o eliminar todos aquellos datos erróneos o aquellos datos que no contribuyen al análisis que se va a realizar.

Los métodos manejados por esta clase son:

- fDatosNoexito(): Elimina todas aquellas peticiones que no son exitosas, es decir aquellas que comiencen con los códigos de error de servidor 400 y 500.
- fDatosGraficos(): Elimina todos los archivos con extensiones .gif, .jpg, .jpeg, .png, .swf, .map.
- fDatosRobots(): Elimina peticiones realizadas por los robots, como Googlebot, InfoSeek Robot 1.0, Infoseek Sidewinder, PerlCrawler 1.0, Scooter, SpiderBot, Site Searcher, entre otros.

leerLog: Esta clase se encarga de la lectura del archivo log.

regLog: Esta clase se encarga de extraer los campos del archivo log para la obtención del registro log.

transformacionLog: Esta clase se encarga de la transformación del registro log de acuerdo a unos parámetros válidos establecidos, en cuanto a los códigos de estado del servidor, el navegador y el usuario entre otros.

transSesionComun: Esta clase se encarga de realizar la transformación de los datos de un archivo log con formato común.

transSesionExtendida: Esta clase se encarga de realizar la transformación de los datos de un archivo log con un formato extendido.

sesionComun: Esta clase se encarga de crear el registro de sesión para un archivo log con formato común.

Los métodos manejados por esta clase son:
- addNomPagvisitadas(): Vector que contiene el nombre de todas las páginas que componen el sitio web analizado.
- addPagvisitadas(): Vector que contiene el número de veces que cada una de las páginas del sitio Web analizado han sido accedidas.

sesionExtendida: Esta clase se encarga de crear el registro de sesión para un archivo log con formato extendido. Lo más importante de esta clase es que se encarga de hacer las transacciones de usuarios y de encontrar el Maximal Forward Reference – MFR -.

7.3.2. Paquete Premining. Este paquete se encarga de realizar un paso previo antes de aplicar los algoritmos de asociación, y que consiste en la carga de los datos de entrada.

fillDataSet: Esta clase es la encargada de cargar los datos de entrada de las sesiones para la tarea de Asociación con los algoritmos Apriori, Equipasso, FPGrowth. Los datos son obtenidos de la tabla de sesión (ya sea sesioncomun o sesionextendida) de la base de datos.

7.3.3. Paquete Algorithms

EL paquete Algorithms se compone a la vez de los paquetes ***association*** para la implementación de los algoritmos Apriori, Equipasso y FPGrowth, el paquete ***clasification*** con los algoritmos C4.5 y Mate y el paquete ***hpg.***

Paquete association

Esta conformado por los paquetes que implementan los algoritmos de asociación: Apriori, Equipasso y FPGrowth.

Paquete Apriori

La implementación del algoritmo Apriori se realizó utilizando una clase denominada *apriori.java.*

Los métodos que maneja esta clase son:

- getFrequents(): Retorna el vector de arboles AvlTree donde se almacenan los distintos itemsets frecuentes.
- makeCandidates(): Recorre cada nodo del árbol AvlTree de itemsets frecuentes tipo n para a partir de las combinaciones de sus elementos generar los itemsets candidatos tipo n+1.
- increaseSupport(): Determina el número de apariciones de un itemset candidato dentro del DataSet.
- showFrequents(): Representación tipo cadena de los itemsets frecuentes.

Al igual que las otras clases que implementan algoritmos de asociación, en el constructor de la clase *Apriori* se usan 2 parámetros: una instancia de la clase *DataSet*, donde vienen comprimidos el conjunto de datos y un entero corto, que es suministrado por el usuario, que será usado como el soporte del sistema durante la ejecución de este algoritmo.

De igual manera, en la construcción de la clase se instancia e inicializa un objeto de la clase *AvlTree* que almacenará los *itemsets* frecuentes tipo 1, el cual es alimentado haciendo uso del método *pruneCandidatesOne* de la clase *DataSet*, el cual recibe el soporte del sistema como parámetro. En este punto también se instancia e inicializa un arreglo que guardará los diferentes árboles balanceados (*AvlTree*), que contendrán los *itemsets* Frecuentes de cada tipo y que serán calculados por la herramienta.

Al disponer ya de los *itemsets* frecuentes tipo 1, estos son almacenados en la posición 0 de este arreglo.

Para disparar la ejecución del algoritmo usamos el método *run* el cual ejecuta un ciclo dentro del cual el método *makeCandidates* se encarga de generar todos los *itemset* candidatos para cada iteración. Para esto el método *makeCandidates* llama al objeto *auxTree*, instancia de la clase *AvlTree* que conserva el último árbol de *itemset* frecuentes y pregunta cuantos *itemsets* tiene en ese instante utilizando el método *howMany* de esta clase.

Para cada elemento encontrado dentro del árbol de *itemsets* frecuentes se recorre un ciclo con los demás miembros del árbol armando combinaciones que generan un nuevo *itemset* candidato haciendo uso del método *combinations* que recibe como

parametros el árbol *auxTree* de *itemset* Frecuentes y un árbol *AvlTree* donde se guardaran los *itemsets* Frecuentes tipo *k + 1* llamando *frequents*. Se aprovecha la ventaja de que los *itemsets* están ordenados en el árbol y se verifica que el *itemset* evaluado coincida en sus n − 1 primeros ítem, siendo n el tamaño de ese *itemset*, con el próximo *itemset* del árbol.

De ser así se instancia un nuevo *itemset* con los n − 1 ítem coincidentes y los 2 últimos ítem de cada *itemset* involucrado. De esta manera, se generan *itemsets* candidatos de tamaño *n + 1.*

Cada *itemset* candidato es pasado como parámetro al método *increaseSupport* donde es contado el número de ocurrencias de ese *itemset* candidato dentro del conjunto de datos. En este punto se hace una consideración importante y si los *itemsets* candidatos que se están construyendo son de tipo 3 o superior se evalúa el paso de poda, esto es, se descompone el *itemset* candidato construido en sus (n−2) posibles combinaciones del tipo anterior, siendo n el tamaño del *itemset* candidato que se esta generando.

No se evalúan las dos últimas combinaciones pues fueran estas las que generaron el candidato que se esta analizando y se tiene certeza de que existen en el árbol de *itemsets* Frecuentes.

Cada una de las combinaciones generadas son buscadas en *auxTree*, hay que recordar que este es el árbol que contiene todos los *itemsets* frecuentes del orden anterior al que se esta generando. Si una de las combinaciones del *itemset* candidato actual no es encontrada en el árbol *auxTree*, este *itemset* ya no es considerado y se procede a evaluar el próximo.

Si el *itemset* candidato actual es de tipo 2 o ha superado el paso de poda se procederá a hacer su conteo. Para ello, se hace una nueva instancia de la clase *Transaction* donde se carga una a una las transacciones del conjunto de datos y se compara con el contenido del *itemset*. Si coinciden los elementos del registro y el *itemset*, este último incrementa su soporte interno en uno.

Al final de este proceso se cuenta con un *itemset* candidato con su soporte establecido, este se compara con el soporte del sistema y de superarlo es incluido en el árbol *frequents*. Cuando ya se han evaluado todos los candidatos para un árbol *auxTree*, se pregunta si se han generado nuevos *itemsets* frecuentes en *frequents*.

Si el método *howMany* de *frequents* arroja existencias se almacena en el arreglo de árboles frecuentes y se asigna a *auxTree* el árbol *frequents* para iniciar de nuevo el proceso de generación de candidatos. Esto se hará hasta que el árbol *frequents* resulte vacío (no hayan nuevos *itemsets* frecuentes). En el arreglo de árboles frecuentes quedan almacenados todos los *itemsets* frecuentes organizados por tipo para que puedan ser visualizados como reglas.

Paquete FPGrowth

Contiene las clases necesarias para el funcionamiento del algoritmo el cuál genera itemsets candidatos a partir de un árbol Nario; siendo las más importantes *FPGrowthNode y FrequentNode* que son las encargadas de formar la estructura del árbol Nario, también se encuentran las clases *FPGrowth, Conditionals, BaseConditional, BaseConditionals y AvlTree*.

FPGrowthNode: Esta clase proporciona la estructura del *FPTree*, en donde cada nodo tiene un valor de *item*, un soporte y los punteros respectivos para realizar los enlaces entre nodos (Punteros al nodo hijo, al nodo padre, al nodo hermano y al siguiente nodo con el mismo nombre).

FrequentNode: Para la posterior creación de *itemsets* frecuentes se almacenan en un arreglo de objetos de tipo *FrequentNode* los *itemsets* frecuentes1 o de tamaño 1. Cada uno de los ítems frecuentes 1 de esta lista se encuentra enlazado a un nodo del árbol Nario, de ahora en adelante *FPtree*, que tenga el mismo nombre que el *itemset* frecuente1.

FPGrowth. Es la clase principal del paquete y en ella se obtienen los itemsets frecuentes.

Para la implementación del algoritmo se debe crear una lista con los itemset frecuentes tamaño 1 y almacenarlos en el arreglo *frequentsOne,* luego se construye el FPTree, para lo cual deben leerse las transacciones de Dataset y se almacenan únicamente los items que superen el mínino soporte.

Una vez se ha realizado lo anterior, se debe recorrer la estructura de los ítem frecuentes tamaño 1 y por cada uno de ellos se construye sus patrones condicionales base, los cuales se obtienen recorriendo *FPtree* desde cada nodo enlazado por los ítem frecuentes tamaño 1 a través de su puntero al nodo padre

hasta la raíz. Cada uno de los patrones condicionales base se almacenan en la clase *BaseConditional* y todo el conjunto de patrones condicionales base de un ítem frecuente tamaño 1 se almacenan en la clase *BaseConditionals.*

Cuando se han construido todos los patrones condicionales base de un ítem frecuente tamaño 1, el siguiente paso de la implementación es determinar cuales de sus ítem tienen soporte mayor o igual al mínimo. Para esto se debe sumar cada uno de los soportes que un ítem tenga en cada patrón condicional base, los ítem que cumplan con el soporte van a conformar los patrones condicionales, los cuales se almacenan en la clase *Conditionals*.

El último paso de la implementación es determinar el conjunto de *itemsets Frecuentes*. Para lo cual se hace uso de la clase *Combinations*, la cual toma los patrones condicionales y los combina con los ítems frecuentes tamaño 1.

Así como para *Apriori y EquipAsso* los *itemsets* frecuentes se almacenan en un vector de árboles AVL balanceados, en donde en cada posición del vector, se encuentran almacenados un tipo de *itemsets* frecuentes. Es decir en la posición 0 del vector se encuentran los *itemsets* frecuentes tipo 1, en la posición 1 los *itemsets* frecuentes tipo 2 y así mismo el resto de *itemsets* frecuentes.

Paquete Equipasso

Este paquete contiene las clases *Equipasso* y la clase *Combinations* orientadas a dar soporte al descubrimiento de reglas de asociación dentro del proceso KDD.

La clase *EquipAsso* se encargada de cargar únicamente los datos que pasan el umbral soporte dado (proceso *EquiKeep* dentro del algoritmo *EquipAsso*) y la clase *Combinations* se encarga de generar todas las combinaciones de un determinado tamaño para cada uno de los registros cargados del conjunto de datos y sus respectivo conteo (proceso *Associator* dentro del algoritmo *EquipAsso).*
La clase principal es *EquipAsso*, al hacer una instancia de esta clase se debe pasar como parámetros una instancia de la clase *DataSet*, llamada *dataset* y que contiene una versión comprimida del conjunto de datos, y un entero corto, que se llamado *support* y que cumple la tarea de soporte del sistema. Estas instancias son asignadas a atributos internos de esta clase. La clase *EquipAsso* cuenta con un atributo de tipo arreglo llamado *Trees* donde se almacenan los diferentes árboles de *itemsets* frecuentes organizados por tamaño.

El primer conjunto de *itemsets* son los de tamaño 1 y se puede obtenerlo al llamar al método *pruneCandidatesOne* de la clase *DataSet* que fue pasada como parámetro al constructor de la clase. Este método devuelve un árbol *AvlTree* que es insertado en la posición 0 del arreglo *Trees* y que contiene el conjunto de *itemsets* Frecuentes tamaño 1. Los demás conjuntos de *itemsets* Frecuentes (tamaño 2, tamaño 3,...) serán almacenados en las siguientes posiciones de ese arreglo organizados en árboles *AvlTree*.

Una vez instanciado un objeto *EquipAsso* se inicia el proceso del algoritmo llamando al método *run* de esta clase. Dentro de este método se inicia un ciclo controlado por el método *findInDataset* donde se recorre el conjunto de datos. Este método declara un objeto del tipo *Transaction* el cual, a través de su método *loadTransaction*, carga los registros del conjunto de datos.

En este punto se realiza un primer filtrado cargando únicamente aquellos ítem que sobrepasen el soporte del sistema y están por ende contenidos en el conjunto de *itemsets* frecuentes tamaño 1 y que ha sido almacenado en la primera posición del arreglo *Trees.*

Es por eso que son enviados como parámetros instancias del *dataset* actual y del *AvlTree* que contiene los *itemset* Frecuentes tamaño 1 para solo cargar del *dataset* los ítems que están en este conjunto. Posterior a este filtrado se pasa esta transacción como parámetro a una instancia de la clase *Combinations* para encontrar sus posibles combinaciones.

Puede darse el caso de que ninguno de los ítem provenientes del *dataset* supere el soporte, ni sea encontrado entre los *itemsets* Frecuentes uno, por lo que se cargará una transacción vacía esta es descartada y se continua con la siguiente transacción.

La clase *Combinations* recibe como parámetro un arreglo que contiene los ítems válidados de cada transacción provenientes del conjunto de datos. Este arreglo es cargado en el constructor de esta clase y posteriormente, con el llamado del método *letsCombine* de la clase *Combinations*, se genera un determinado grupo de combinaciones dependiendo del tamaño que se quiera generar, combinaciones tamaño 2, tamaño 3, etc. según sea el caso, con cada una de estas combinaciones crea objetos de la clase *ítemset*. El método *letsCombine* recibe como parámetro un árbol *AvlTree* llamado *treeCombinations* en el cual se van insertando los objetos *itemset* que han sido generados.

Se hace uso de un árbol balanceado *AvlTree*, para almacenar este tipo de conjuntos de *itemsets* para agilizar las búsquedas de un determinado *itemset* generado. Si un *itemset* generado a partir de un combinación ya existe en el árbol *treeCombinations* el soporte interno de este se incrementa en uno. De esta manera, la primera vez que se ejecute el método *letsCombine* se generarán las combinaciones tamaño 2 de cada transacción válida del conjunto de datos y quedarán almacenadas en *treeCombinations* el total de *itemsets* generados por el *dataset* y su correspondiente soporte.

Posterior a este paso se procede a barrer el árbol *treeCombinations* para seleccionar aquellos *itemsets* Frecuentes que hayan superado el soporte del sistema. Esta función se realiza haciendo uso del método *pruneCombinations*, de la clase *EquipAsso*, que recibe como parámetros el árbol *treeCombination* y un árbol *AvlTree* llamado *treeFrequents*, donde se guardaran únicamente los *Itemsets* Frecuentes. El método *pruneCombinations* es un método recursivo.

Al final de este método se liberan los recursos ocupados por *treeCombinations* y contaremos con un objeto *treeFrequents* donde están almacenados todos los *Itemsets* frecuentes de un determinado tamaño. Para concluir el proceso se pregunta si el árbol *treeFrequents* contiene elementos, se usa el método *howMany* de la clase *AvlTree* para este propósito.

De ser así este árbol es almacenado en el arreglo *Trees* de la clase *EquipAsso* en su respectiva posición de acuerdo al tamaño de los *itemsets* que contiene y el método *findInDataset* retorna una señal de true al ciclo principal del método *run* que controla la continuación del algoritmo.

El proceso se repite pero calculando las combinaciones e *Itemsets* Frecuentes tamaño 3. El algoritmo se detiene cuando el método *howMany* del árbol *treeFrequents* devuelva 0 lo que quiere decir que ya no se han generado *Itemsets* Frecuentes en esta iteración y el proceso de *EquipAsso* ha terminado. En este caso el método *findInDataset* retorna una señal de *false* y el ciclo principal de la clase *EquipAsso* concluye.

Paquete classification

Contiene el paquete C4.5 y el paquete Mate. Este paquete , adicionalmente tiene las clases *clasesAtrib, ClasificationTableModel, elemento, sesionTrans, Transformation_cla.*

ClaseAtrib: Contiene los atributos a analizar.

ClasificationTableModel: Contiene los datos discretizados.

elemento: Es uno de los nodos del árbol de decisión, que puede ser un atributo o un valor de un atributo.

SessionTrans: Es un registro de una sesión transformada.

Transformation_cla: Realiza la transformación de una sesión en una sesión discretizada.

Paquete C45

Este paquete implementa el algoritmo de clasificación C4.5, el cual permite descubrir conocimiento por medio de la clasificación de atributos a través de la ganancia de información de los mismos, aplicando fórmulas para obtener la entropía de cada atributo con respecto a otros.

Contiene las clases:

Clase c4.5. Inicialmente es creado un objeto de la clase c4.5. Se inicia cargando los datos que van a ser analizados en el algoritmo. Cada transacción se almacena en un *Arraylist.*

ClassificationTableModel. Es la estructura que será utilizada para crear el modelo de la tabla de datos.

Build_decisionTree. Encargada de la construcción del árbol de decisión.

En primera instancia se hace un conteo de los distintos valores en el atributo clase o target para encontrar la entropía inicial y hacer el cálculo de las entropías de cada atributo, para saber cual es la que brinda la mayor ganancia de información.

El flujo de entrada es un conjunto de datos presentados en un *ClassificationTabelModel*. En *atributo* se colocan los valores de los atributos. Posteriormente *build_dessiociontree* se encargada de la construcción del árbol de decisión.

Paquete mate

Contiene las clases implementadas en la programación del algoritmo *MateBy*.

Clase mate. Inicialmente es creado un objeto de la clase mate. Se inicia cargando los datos que van a ser analizados en el algoritmo.

MateTableModel. Es la estructura que será utilizada para crear el modelo de la tabla de datos.

Build_decisionTree. Encargada de la construcción del árbol de decisión.

mateFila. Devuelve un objeto de una especificada fila y columna.

El flujo de entrada es un conjunto de datos presentados en un *MateTabelModel*. En *atributo* se colocan los valores de los atributos. Posteriormente *build_dessiociontree* es la encargada de la construcción del árbol de decisión.

Paquete hpg

Las clases que conformar este paquete son:

Hpg. Esta clase se encarga de realizar el proceso en el cual se aplica el algoritmo HPG y la respectiva construcción de la gramática para que sea visualizada como un reporte.
Los métodos manejados en esta clase son:

- crearProduccionesInicio(): Como su nombre lo indica, crea las producciones iniciales.
- crearProduccionesInternas(): Crea las producciones internas.
- crearStatesGrammar(): Crea los estados de la gramática.
- getProducciones(): arraylist de producciones generadas.
- llenarNavigationTrails(): Llena los caminos de navegación.

7.3.4. Paquete Utils. Esta clase se compone por las siguientes clases:

AssocRules: Esta clase es la encargada de recorrer los arboles de ItemSets frecuentes, calcular sus combinaciones y generar las reglas para los algoritmos de asociación.

La clase AssocRules maneja los siguientes atributos:

- frequents: Es un vector de arboles AVL de ItemSets frecuentes.
- dictionary: Es un arreglo que contiene los nombres de los ItemSets frecuentes (Diccionario de datos).
- confidence: Es el parámetro de confianza con la que se va a evaluar las reglas de asociación.

La clase AssocRules maneja los siguientes métodos:

- buildRules(): Recorre el vector de arboles de ItemSets frecuentes para entrar a evaluar cada una de las combinaciones y obtener las reglas de asociación.
- getRules(): Retorna el vector que contiene las reglas de asociación.
- showRules(): Representación textual de las reglas ordenadas descendentemente de acuerdo al soporte.
- sortByGoldStone(int op): Método de ordenamiento binario. Ordena las reglas de mayor a menor de acuerdo al parámetro op. Si op es 0 las reglas serán ordenadas de acuerdo al tamaño de su consecuente. Y si op es 1 las reglas serán ordenadas de acuerdo a la confianza de reglas similares.

El curso normal de los eventos empieza con el llamado al método *buildRules* en el cual se inicia el recorrido del vector de de árboles AVL. La raíz de cada árbol es enviada como parámetro al método *walkTree* que es el encargado de recorrer el árbol y armar por cada rama recorrida un arreglo que es enviado al método *combinations* para que se generen todas las combinaciones posibles de esa rama y obtener las nuevas reglas.

Cada regla es evaluada para verificar que cumpla el nivel de confianza. La fórmula para la evaluación de confianza es: (soporte Frecuentes/soporte Antecedente) 100. Si una regla supera el nivel de confianza es enviada al método decodeFrecuents para su decodificación. Al final para efectos prácticos de representación existen varios métodos que permiten ordenar las reglas generadas de acuerdo a varios parámetros: soporte, confianza u orden alfabético.

AvlNode: Esta clase es la encargada de la construcción de la estructura interna de datos de un nodo perteneciente a un árbol ***AVL.***

La clase AvlNode maneja los siguientes métodos:

- getItemset(): Guarda los ítems –datos- con los que será cargado el nodo.
- getLeft(): Puntero hacia los hijos izquierdos.
- getRight(): Puntero hacia los hijos derechos.
- isCheck(): Variable de control de balance del árbol.
- setCheck(boolean check)
- toString(): Representación textual de un nodo AVL.

AvlTree: Esta clase es la encargada de crear un árbol AVL y de proveer los métodos necesarios para su manejo. Un árbol AVL es un tipo de árbol binario que es balanceado de tal manera que la profundidad de dos hojas cualquiera en el árbol difiera como máximo en uno.

La clase AvlTree maneja los siguientes métodos:

- getroot(): Encargado de encontrar la raíz del árbol AVL.
- addItem(): Adiciona un Item al Itemset.
- getItems(): Es un arreglo con los Items de este Itemset.
- getSupport(): Obtiene el valor del soporte del itemset.
- getType(): Obtiene el tamaño del Itemset.
- setSupport(): Establece el valor del soporte del Itemset.
- increaseSupport(): Incrementa el soporte de un Itemset en uno.
- n: variable de control de la profundidad o número de niveles del árbol.
- node: nodo para recorrer el árbol.
- stack: pila utilizada para almacenar la ruta del recorrido hecho en una inserción, búsqueda o eliminación.

Para lograr la construcción y manejo de un árbol AVL son necesarios algunos métodos que sirven para la construcción del árbol como métodos de inserción, búsqueda, profundidad del árbol, etc.

BaseDatos. Esta clase es utilizada siempre que se necesite efectuar conexiones a bases de datos por medio del objeto de conexión java™ para **Postgres**. Cabe aclarar que el sistema gestor de bases de datos (SGBD) inicial es Postgres pero el nombre del sistema gestor es totalmente parametrizable, permitiendo de esta manera la conexión a casi cualquier SGBD.

La clase BaseDatos maneja los siguientes métodos:

- iniciarBD():Inicializa la operación tomando el nombre de la base de datos, el usuario y el password de la misma.
- getNombre(): Obtiene el nombre de la base de datos.
- getTablas(): Retorna un vector con el nombre de las tablas que contenga la base de datos.
- insertarBD(): Permite insertar un registro nuevo en una tabla.
- cerrarBD(): Cierra la conexión a la base de datos.
- getDatos(): Permite ejecutar una consulta general a una tabla.
- db: Es el objeto encargado de establecer la conexión.
- stm: Es la variable encargada de ejecutar un *query* determinado.

Inicialmente se estable el nombre del controlador de la base de datos que se desea consultar. En este caso el controlador *Postgres*. Luego se llama al método *iniciarBD* que se encarga de establecer la conexión a la base de datos. Tiene como parámetros el nombre de la base de datos, el nombre del usuario y el password. Existen otros métodos implementados para la consulta del nombre de las tablas, inserción y consulta de los datos de una tabla.

El método *insertarBD* tiene como parámetros el nombre de la tabla, el identificador de un ítem y el nombre del ítem a insertar. *getDatos* solo requiere el nombre de la tabla a consultar.

DataSet. La clase DataSet es la estructura que los algoritmos de asociación utilizan para almacenar los datos de tal forma que ocupan menos espacio en memoria. La estructura que utiliza DataSet es un árbol N-Ario con los datos a minar codificados como datos tipo short, en esta estructura se aprovecha en una sola rama los ítems de distintas transacciones con datos repetidos controlando su número de apariciones o soporte.

La clase DataSet maneja los siguientes atributos:

- current: Es el último nodo insertado en el DataSet.
- lcurrent: Último nodo del DataSet referido en la lista.
- lnode: Un nuevo nodo en la lista.
- lroot: La raíz de una lista que referencia a todos los nodos del DataSet que tienen soporte.
- Ntransactions: Número de transacciones en el DataSet.
- root: La raíz del DataSet.

Los métodos utilizados por esta clase son:

- buildDictionary(): Construye el diccionario a partir de una conexión a base de datos.
- sortDictionary(): Ordena alfabéticamente el diccionario de datos.
- findAttrName(): Busca un atributo en la lista de nombres de los atributos.
- codeAttribute(): Retorna la codificación de un atributo, obtenido del diccionario de datos.
- decodeAttribute(): Retorna el nombre del atributo en la posición indicada.
- showDictionary(): Representación textual del diccionario de datos.
- setNtransactions(): Establece el número de transacciones del DataSet.
- getNtransactions(): Retorna el número de transacciones del DataSet.
- buildNTree(): Construye un nodo del DataSet.
- getCandidatesOne(): Retorna el árbol de itemsets frecuentes tipo 1.
- pruneCandidatesOne(): Para los Itemsets candidatos tipo 1, construye los itemsets frecuentes tipo 1 y los almacena en un árbol AVL.
- linkLeaf(): Enlaza un nodo con soporte a lista alterna. La lista alterna tiene referencias a los nodos del DataSet que tienen soporte.
- setPointers(): Establece una referencia a la raíz de la lista de nodos hoja o con soporte. Método útil para recorrer la lista alterna.
- getNode(): Retorna el valor de un item del DataSet.
- showNTree(): Representacion textual del DataSet.
- saveTree(): Guarda el DataSet en disco.
- getLRoot(): Recupera el DataSet de disco y lo vuelve a cargar en memoria.

FileManager. Esta clase gestiona todo lo relacionado con el flujo de información entre el archivo leído, el *DataSet* y el diccionario de datos.

Los métodos manejados por esta clase son:

- buildDataMatrix(): Retorna una matriz que representa un archivo de acceso aleatorio con los nombres de sus atributos y sus datos.
- buildMultivaluedDataset(): Retorna el DataSet que se construye a partir de un archivo de acceso aleatorio de tipo multivaluado o multicolumna.
- buildUnivaluedDataSet(): Retorna el DataSet que se construye a partir de un archivo de acceso aleatorio de tipo univaluado o binario.
- closeFile(): Cierra el flujo hacia el archivo de acceso aleatorio.
- dataAndAttributes(): A partir de un archivo de acceso aleatorio, almacena los nombres de los atributos en un arreglo, los datos en una matriz y el diccionario de datos en un ArrayList.
- deleteFile(): Borra físicamente el archivo de acceso aleatorio.

- getAttributes(): Retorna el arreglo que tiene los nombres los atributos del archivo de acceso aleatorio.
- getData(): Retorna la matriz que almacena los datos del archivo de acceso aleatorio.
- getDictionary(): Retorna el diccionario de datos construido a partir del archivo de acceso aleatorio.
- getFileName(): Retorna el nombre del archivo de acceso aleatorio.
- getFileSize(): Retorna el tamaño en bytes del archivo de acceso aleatorio.
- readCsv(): A partir de un Comma Separated Value File (csv), almacena los nombres de los atributos en un arreglo, los datos en una matriz y el diccionario de datos en un ArrayList.
- readFile(): Lee y muestra el contenido de los archivos sin hacer distinción del separador de transacciones, el cero.
- ReadTransaction(): Lee y muestra el contenido de un archivo de acceso aleatorio.
- setOutChannel(): Ubica el flujo en una posición determinada del archivo de acceso aleatorio.
- toString(): Retorna la representación textual del archivo plano relacionado con esta clase.
- writeItem(): Escribe un item de tipo short (2 bytes) en el archivo de acceso aleatorio.
- writeString(): Escribe una cadena en el archivo de acceso aleatorio.

ItemSet. Clase que almacena los valores de los itemsets.

La clase ItemSet maneja los siguientes atributos:

- ítem: Es un arreglo que almacena los ítems que forman el itemset.
- support: Es el soporte asociado a ese itemset.
- n: Es el tamaño del itemset.

Los métodos manejados por esta clase son:

- addItem(): Adiciona un item al itemset.
- getItems(): Devuelve el itemset asociado.
- getSupport(): Devuelve el soporte de este itemset.
- getType(): Devuelve el tamaño de un itemset.
- increaseSupport(): Incrementa el soporte de este itemset en uno.
- setSupport(): Establece el soporte de este itemset.

- toString(): Representacion textual del itemset.

NodeF. Esta clase se encarga de crear los nodos intermedios de un árbol de datos. También se implementan los métodos para enlazar un nodo a otro. Entre ellos se encuentran: *addSon, addBro y findBro* que se encargan de adicionar un hijo, adicionar un hermano y buscar si un nodo tiene hermanos respectivamente.

Clase NodeF. Esta clase extiende a la clase *NodeNoF*. Este tipo de nodos son las hojas de las ramas. Lo que las hace diferentes, son dos atributos adicionales, uno es el soporte, que se encarga de llevar a cabo el conteo de veces que una transacción repetida ha sido tenida en cuenta, y el otro es un puntero a otra hoja. En el momento de recorrer un árbol, el tipo de nodo nos permite saber que hemos llegado al final de una rama.

Transaction. Esta clase es la encargada de manejar los *itemsets* o conjuntos de ítem por cada transacción. Cada uno de los *itemsets* son almacenados en vectores. Esta clase se encarga de alimentar esos vectores, permite hacer consulta sobre ellos u organizarlos.

Los métodos manejados por la clase Transaction son:

- getArticles(): permite ver los artículos en una transacción.
- getSize(): devuelve el tamaño de una transacción.
- getItemset(): devuelve un ítem de la transacción.
- srtBySupport(): ordena las transacciones por soporte.
- srtByItem(): ordena las transacciones por el ítem.
- loaditemsets(): carga los artículos o ítem de una transacción.
- clearTransaction(): borra los ítem de una transacción.

7.3.5. Paquete media. En este paquete se encuentran todas las imágenes y el texto asociados a cada icono, utilizados en la interfaz gráfica de Polaris.

7.3.6. Paquete workspace. La interfaz gráfica de la aplicación esta desarrollada bajo la metodología Drag and Drog (arrastrar y soltar) la cual se refiere a la acción de arrastrar y soltar con el ratón objetos de una ventana a otra o entre partes de una misma ventana o programa. Los objetos arrastrados son habitualmente

archivos, pero también pueden ser arrastrados otros tipos de objetos en función del programa.

En el paquete workspace se encuentra contenido todo el código que integra la interfaz gráfica y cada uno de los algoritmos correspondientes a las etapas de minería de datos Web. La interfaz fue diseña da en NetBeans™ 5.5.

Workspace contiene los siguientes paquetes:

- Asociación
- Clasificación
- Clustering
- HPG
- Main_Set
- Prepro
- Source
- Views
- Help

Cada paquete se encuentra asociado a una etapa de la minería así:

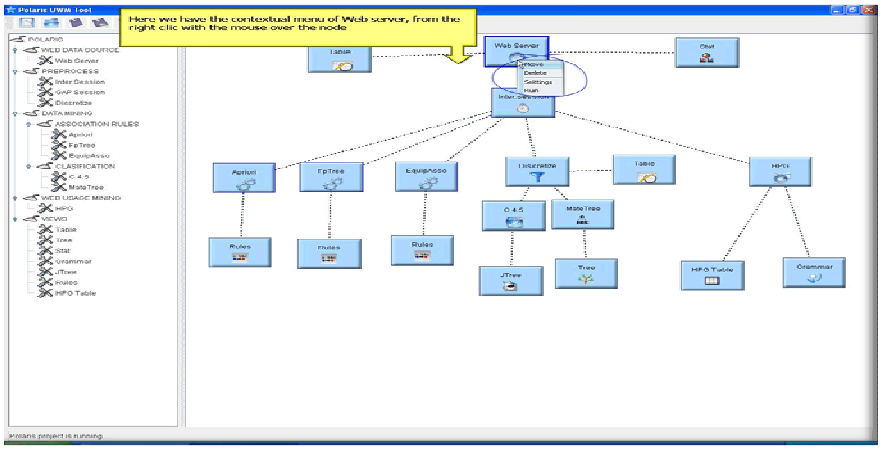
- Main_Set, contiene los archivos que se destacan como configuraciones principales.
- En Source está contenida la etapa de abrir un log y sus respectivas configuraciones.
- En Prepro, la etapa de preproceso.
- Clustering, se establece como una sección en desarrollo para futuros proyectos.
- HPG contiene todo lo referente al algoritmo que lleva su nombre.
- Views, se encarga de la etapa de visualización para cada algoritmo.
- Help muestra la ayuda.

Main_Set: El Main_Set contiene:

Area.java. Este archivo contiene todas las configuraciones respectivas a los componentes del panel izquierdo del área de trabajo y su presentación, por

ejemplo el menú contextual de cada nodo (entiéndase como nodo cada cuadro arrastrado al área de trabajo), tal y como se aprecia en la figura 2.63.

Figura 2.63. Área de trabajo



Es una clase que extiende de jPanel se encarga de dibujar los nodos y de la comunicación interna de los mismos. Es la encargada de los procesos de abrir y guardar el esquema de la minería, el formato para guardar es XML, estos procesos de escritura a medio de almacenamiento se llevan a cabo a través de la biblioteca JDOM, el esquema se guarda como *.pol

Figura 2.64. Código Paquete jdom

```
import org.jdom.Attribute;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;
```

Main.java. Muestra la ventana principal y realiza las conexiones de código requeridas para ejecutar la aplicación.

SimpleNode.java. Este archivo se caracteriza por ser un archivo de generalización de donde heredan todos los nodos. Los nodos heredan atributos tales como ícono, tamaño, nombre, color de borde etc y métodos cómo ejecución y configuración.

Figura 2.65. Código Configuraciones Principales de los Nodos.

```
public Color getBorderColor(){
    return borde;
}
public void setBorderColor(Color c){
    this.borde = c;
}

public Area getParent() {
    return parent;
}
public void setParent(Area parent) {
    this.parent = parent;
}

public boolean isConfigured() {
    return configured;
}
public void setConfigured(boolean configured) {
    this.configured = configured;
}

public boolean isExecuted() {
    return executed;
}
public void setExecuted(boolean executed) {
    this.executed = executed;
}
```

SplashScreen.java [illegible] nicial mientras se realiza la carga del programa.

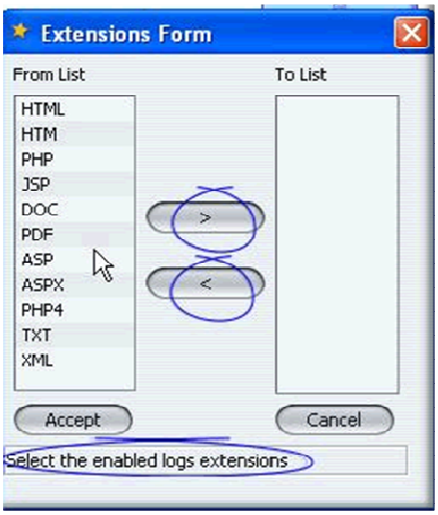
frmSettings: es el formulario en donde se establecen password y login para realizar la conexión a base de datos.

Figura 2.66. Formulario acceso BD.

Source. Contiene los siguientes formularios:

FrmExt. Es el formulario que muestra la selección de extensiones Web para minar.

Figura 2.67. Formulario extensiones.



FrmSorceSelect. Es el formulario de la configuración del nodo SelectSource.

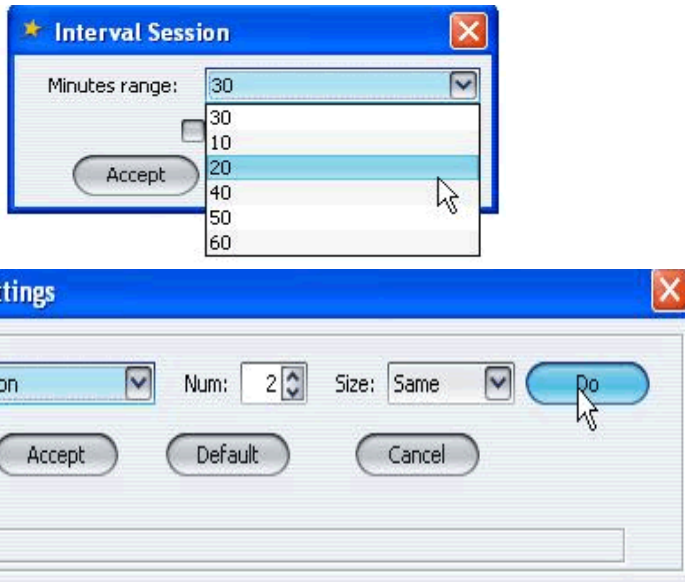
Figura 2.68. Formulario configuración SelectSource.

SelectSourceNode.java. Instanciación del **SimpleNode.java** para la selección del log.

Prepro. Contiene las siguientes instancias y formularios.
FilterNode, IntervalNode, UmbralNode: Es la instanciación del **SimpleNode.java**

FrmGAP, FrmInterval, frm_Cla: Son los formularios que muestran las configuraciones de FilterNode que hace la discretización y de las sesiones umbral e interval.

Figura 2.69. Formularios discretizacion y sesiones



hpg. Contiene la instanciación del **SimpleNode.java** y el formulario de configuración del HPG.

Classification y Association. En el paquete algorithms se muestra la instanciación del **SimpleNode.java** para cada algoritmo de asociación y de clasificación y en el paquete de settings se encuentran contenidos los formularios de configuración de cada algoritmo.

Figura 2.70. Formulario Configuración Asociación

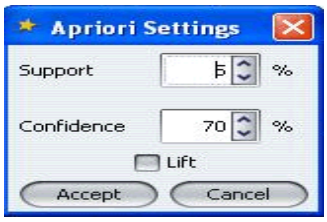
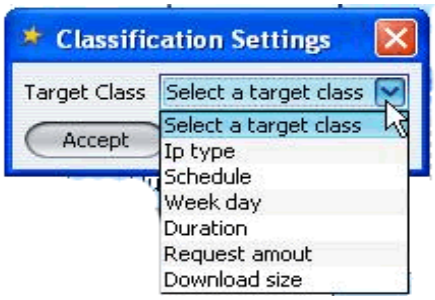


Figura 2.71. Formulario Configuración Clasificación



Views. Como su nombre lo indica en este paquete se encuentran los archivos que hacen posibles las visualizaciones: statistical, jtree, tree, grammar, table, rules, HPGtable. Para la visualización estadística se utiliza la librería jfreechart.

Figura 2.72. Visualización estadística.

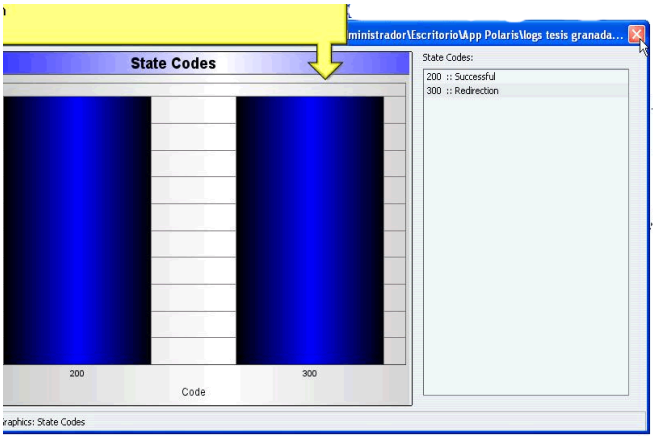
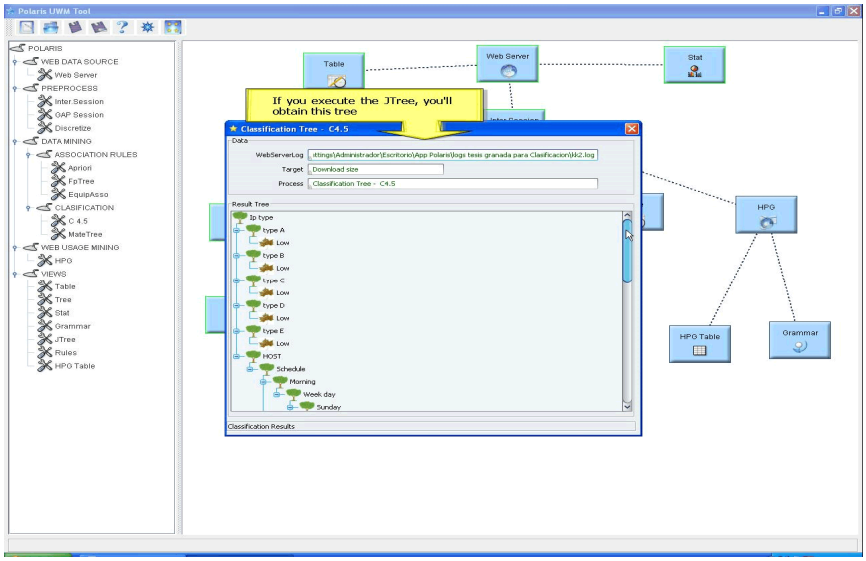


Figura 2.73. Visualización jTree.



Help. En este paquete están los archivos que cargan la visualización de la ayuda.

7.3.7. Paquete Statistical. Esta compuesto por todas las clases necesarias para la generación y visualización de los reportes estadísticos.

Las clases manejadas en este paquete son:

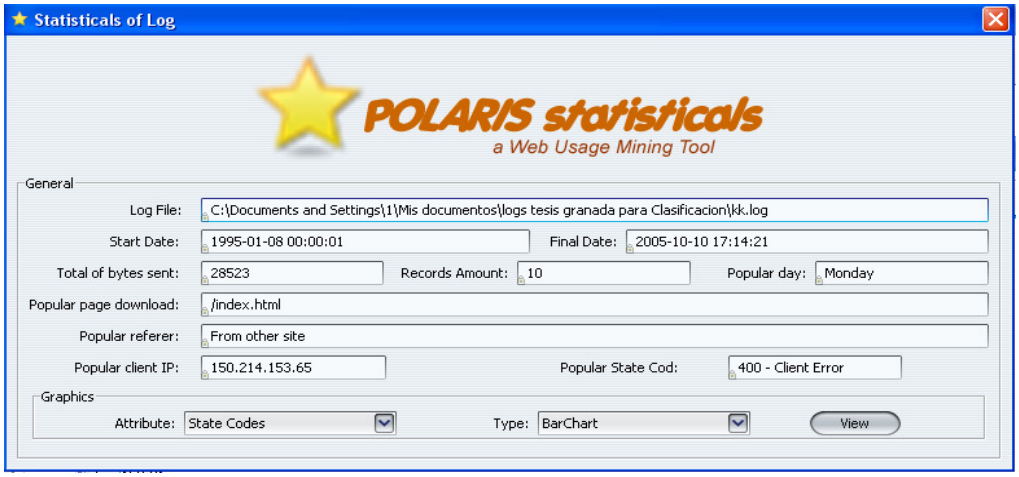
estadísticas. Esta clase es la utilizada para realizar formar las estadísticas generales obteniendo datos sobre los códigos de estado de los servidores, días de la semana, páginas descargadas y número de bytes descargados de las mismas.

gráficas. Esta clase es la utilizada para la realización de las gráficas estadísticas en forma de líneas, barras o pastel

En la visualización de los datos se puede apreciar una información general sobre el conjunto de datos analizados obteniendo las estadísticas sobre el total de bytes descargados, el día de la semana en la que se realizó el mayor número de descargas, la página web mas popular que ha sido descargada, la página de referencia mas popular que ha sido descargada, el código de estado de servidor que se puede apreciarse con mas frecuencia y la información sobre la IP del cliente que mas veces aparece en los registros.

Además de lo anterior, se cuenta también con la opción de visualización gráfica de los resultados obtenidos de acuerdo a unos ítems preestablecidos en cuanto a los códigos de estado, el día de la semana en el que se realizó la mayor descarga de páginas web, las páginas más descargadas y el cliente IP. Las gráficas pueden ser apreciadas en los tipos de líneas, barras y pies o pasteles.

Figura 2.74. Interfaz Paquete Statistical



8 PRUEBAS Y EVALUACIÓN DE RESULTADOS

Las pruebas de rendimiento de la herramienta se desarrollaron utilizando un computador Intel Pentium™ IV de 3.0. GHz, disco duro de 180 GB, memoria de 2 Gigas y tarjeta de video de 64 megas.

EL conjunto de datos de las pruebas esta conformado por diferentes archivos logs pertenecientes a repositorios de la Nasa y repositorios reales de la Universidad de Nariño.

Los archivos log utilizados para las pruebas de clasificación fueron obtenidos de las páginas.

http:// www.nasa.gov/centers/kennedy/home/index.html
http:// altair.ugr.es/portalcpp/web/

Son archivos log del centro espacial Kennedy NASA en Florida, con archivos log de acceso a la NASA de los meses de julio y agosto de 1995.

Estos archivos log fueron recolectados por Jim Dumoulin del centro espacial de Kennedy y Martin Arlitt y Carey Williamson de la universidad de Saskatchewan.

Los archivos log utilizados para las pruebas de asociación y HPG son repositorios del portal de la Universidad de Nariño.

El objetivo de las pruebas realizadas a continuación, es determinar el rendimiento de los algoritmos implementados en Polaris en cuanto al tiempo y los registros analizados en cada uno de las tareas de minería de datos y sus respectivos algoritmos.

8.1 PRUEBAS PARA LA TAREA DE ASOCIACIÓN

Se realizaron diferentes tipos de pruebas para la tarea de asociación la cual consistió en verificar el tiempo de ejecución de los algoritmos de asociación para cada conjunto de datos haciendo variación del soporte y medir el rendimiento en

cuanto a tiempo empleado. Los resultados se muestran en las tablas 2.11 hasta la tabla 2.19 y en la figura 2.75 hasta la figura 2.83.

Tabla 2.11. Conjunto de datos asociación y HPG

Conjunto de datos asociación y HPG		
Nombre del registro	Número de registros	Tamaño KB
access_log	176	457
ainfo_acces_log	188	1923
matriculas_access_log	476	5958
estudiantes_access_log	492	4467
akane_access_log	570	2037
personal_access_log	1013	768
ci_access_log	5443	10,068
www.access_log	19425	93759

Figura 2.75. Conjunto de datos asociación y HPG

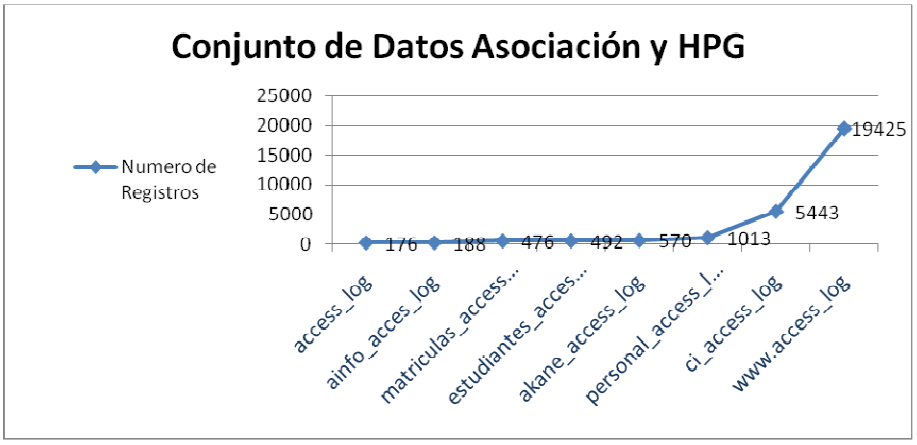


Tabla 2.12. Registro acces_log asociación

acces_log			
	tiempo milisegundos		
Soporte %	Apriori	FPGrowth	Equipasso
1	187	203	172
0,75	172	172	172
0,5	156	156	156
0,25	185	156	156
0,05	190	156	156

Figura 2.76. Gráfica soporte/Tiempo acces_log asociación

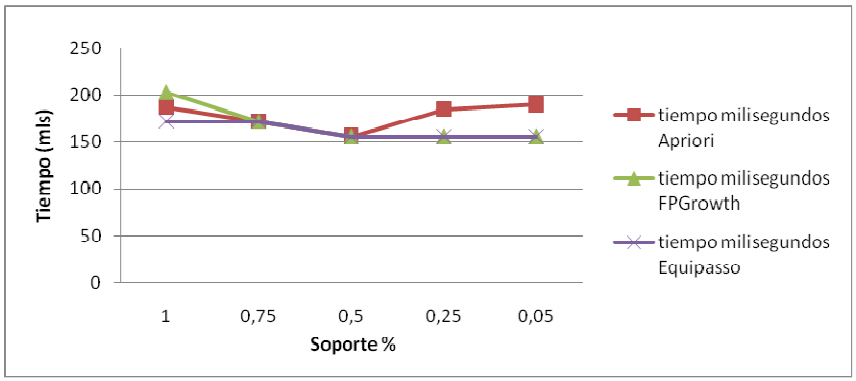


Tabla 2.13. Registro ainfo_access_log asociación

ainfo_acces_log			
	tiempo milisegundos		
Soporte %	Apriori	FPGrowth	Equipasso
1	219	203	219
0,75	328	343	227
0,5	297	187	234
0,25	325	172	170
0,05	302	150	147

Figura 2.77. Gráfica soporte/tiempo ainfo_acces_log asociación

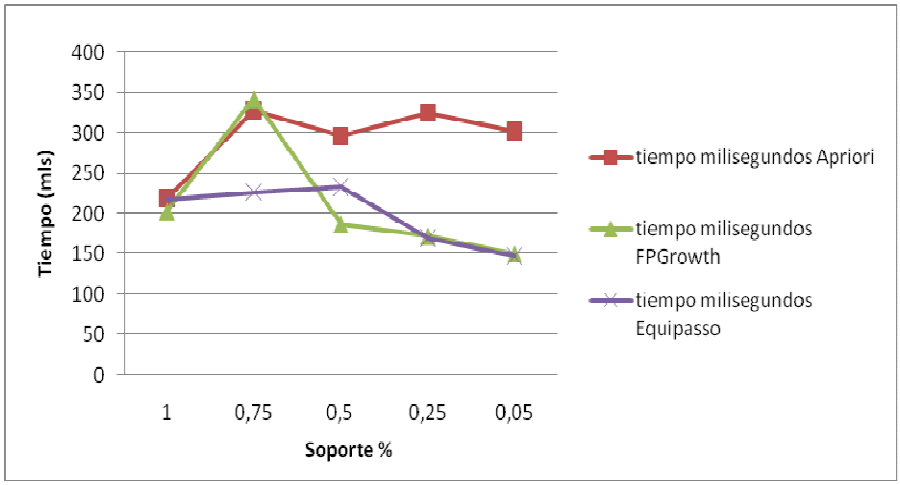


Tabla 2.14. Registro matriculas_acces_log

matriculas_access_log			
	tiempo milisegundos		
Soporte %	Apriori	FPGrowth	Equipasso
1	240	234	218
0,75	158	228	203
0,5	156	187	198
0,25	390	156	172
0,05	250	156	155

Figura 2.78. Gráfica soporte/tiempo matriculas_acces_log asociación

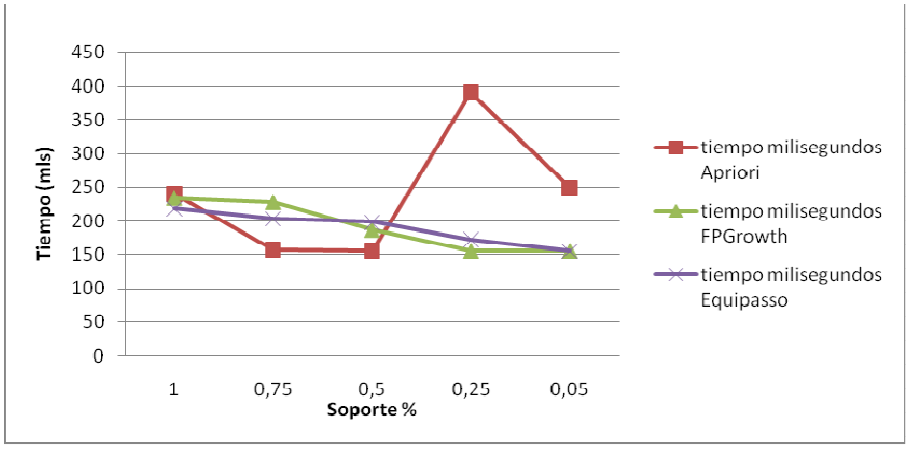
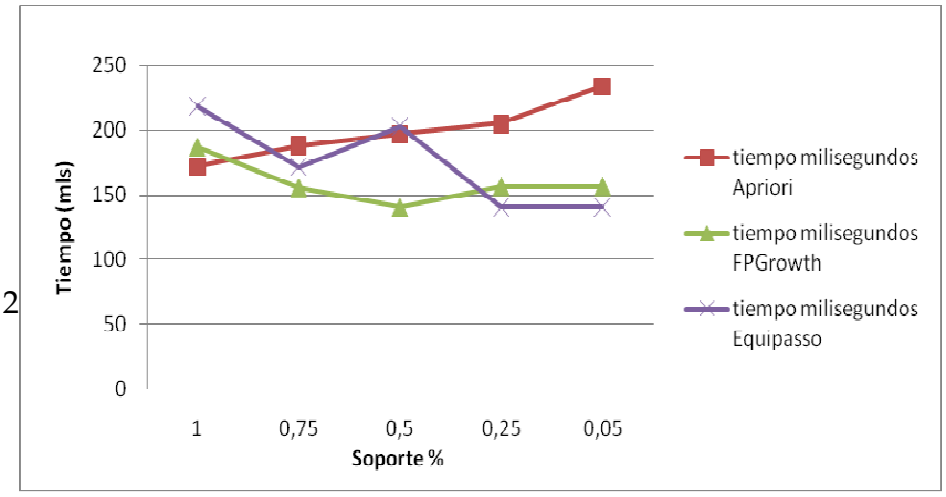


Tabla 2.15. Registro estudiantes_acces_log

estudiantes_access_log			
	tiempo milisegundos		
Soporte %	Apriori	FPGrowth	Equipasso
1	172	187	219
0,75	188	156	172
0,5	197	141	203
0,25	205	157	141
0,05	234	157	141

Figura 2.79. Gráfica soporte/tiempo estudiantes_acces_log asociación.

Tabla 2



akane_access_log			
	tiempo milisegundos		
Soporte %	Apriori	FPGrowth	Equipasso
1	171	282	266
0,75	172	203	234
0,5	250	203	225
0,25	266	215	215
0,05	255	210	207

Figura 2.80. Gráfica soporte/tiempo akane_acces_log asociación.

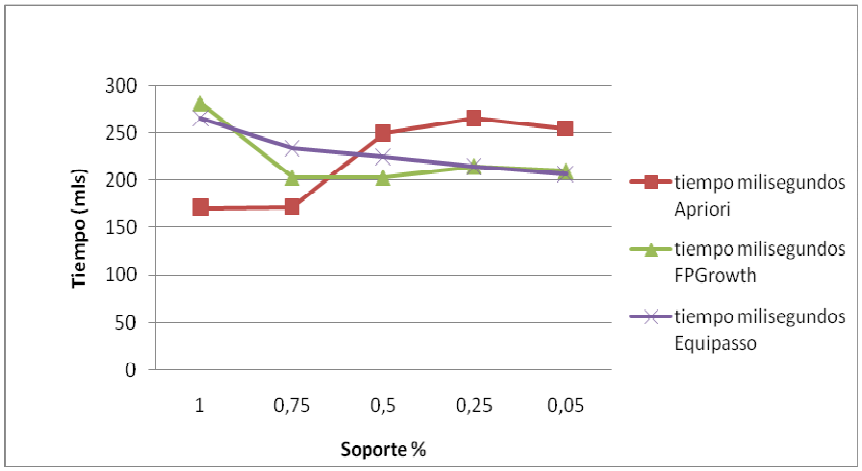


Tabla 2.17. Registro personal_acces_log

personal_access_log			
	tiempo milisegundos		
Soporte %	Apriori	FPGrowth	Equipasso
1	328	478	556
0,75	343	438	593
0,5	413	235	281
0,25	449	438	250
0,05	467	328	272

Figura 2.81. Gráfica soporte/tiempo personal_acces_log asociación.

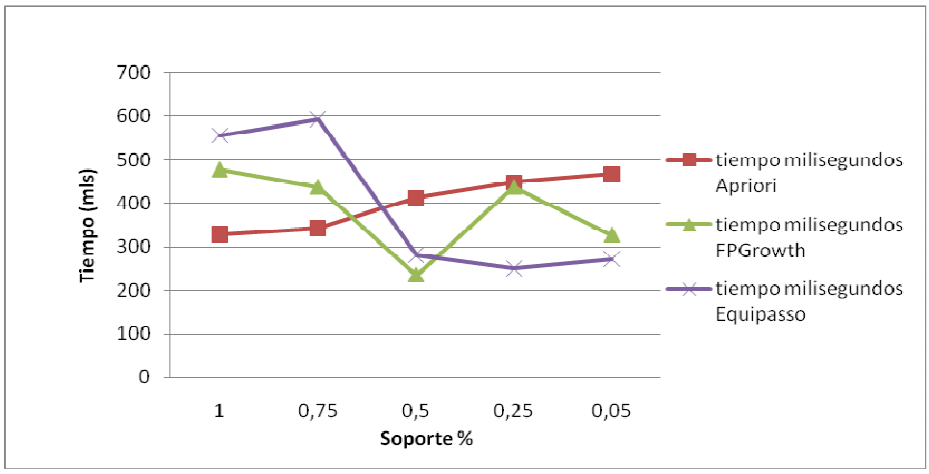


Tabla 2.18. Registro ci_acces_log

ci_access_log			
	tiempo milisegundos		
Soporte %	Apriori	FPGrowth	Equipasso
1	328	875	625
0,75	566	766	610
0,5	531	678	593
0,25	547	532	563
0,05	734	653	500

Figura 2.82. Gráfica soporte/tiempo ci_acces_log asociación.

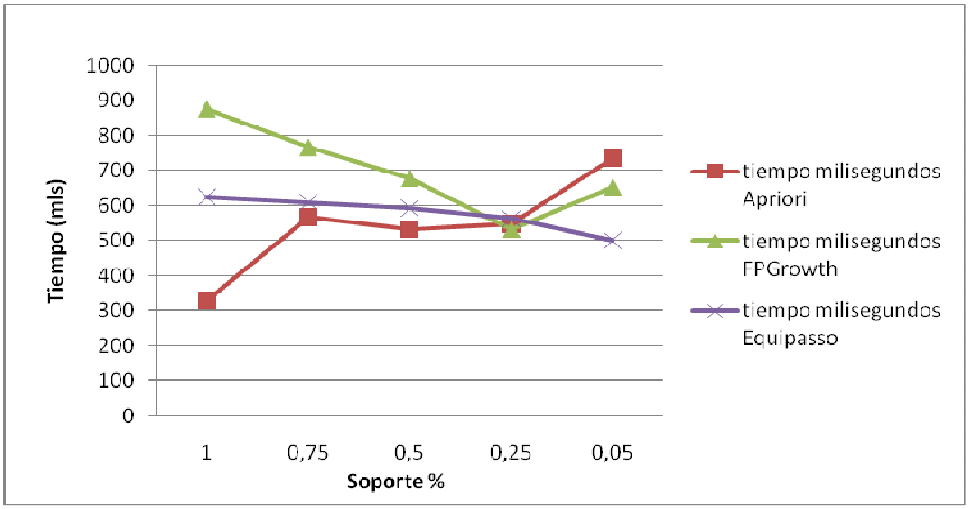
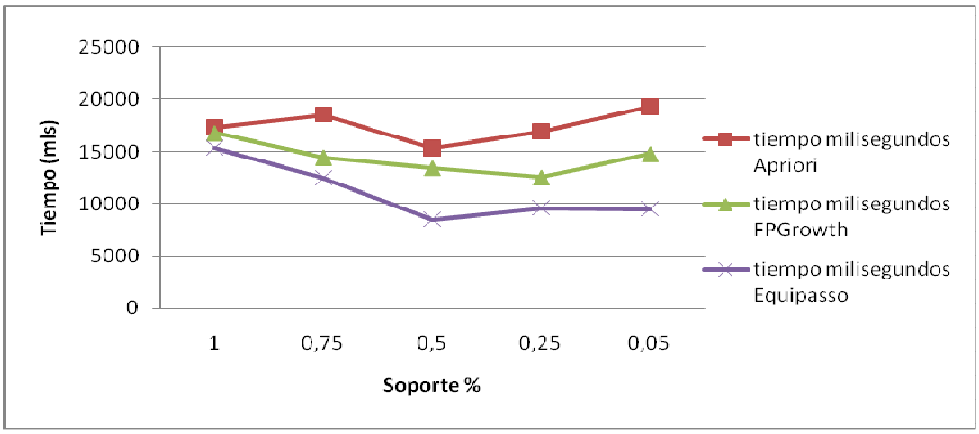


Tabla 2.19. Registro www.acces_log

www.acces_log			
	tiempo milisegundos		
Soporte %	Apriori	FPGrowth	Equipasso
1	17345	16799	15390
0,75	18585	14468	12532
0,5	15296	13437	8531
0,25	16906	12547	9625
0,05	19344	14781	9485

Figura 2.83. Gráfica soporte/tiempo www.acces_log asociación.



8.2 EVALUACIÓN DE LAS PRUEBAS ASOCIACIÓN

Las conclusiones a las que se llegaron después del análisis de los resultados obtenidos con los conjuntos de datos para la tarea de asociación fueron:

- El algoritmo Equipasso es el algoritmo que ofrece un mejor rendimiento en cuanto a tiempo en todas las pruebas realizadas.
- El algoritmo Apriori es sin duda el algoritmo que emplea un mayor tiempo de ejecución lo que se traduce en un mayor tiempo de ejecución y un menor rendimiento.
- En general se puede apreciar que el tiempo empleado por los algoritmos FPGrowth y Equipasso es directamente proporcional al valor del soporte tomado para el análisis, mientras que el tiemplo empleado por Apriori es inversamente proporcional al valor del soporte tomado. Es decir Apriori emplea más tiempo a medida que se disminuye el soporte, contrario a los algoritmos FPGrowth y Equipasso que emplean un mayor tiempo con un soporte mayor.

8.3 PRUEBAS PARA LA TAREA DE CLASIFICACIÓN

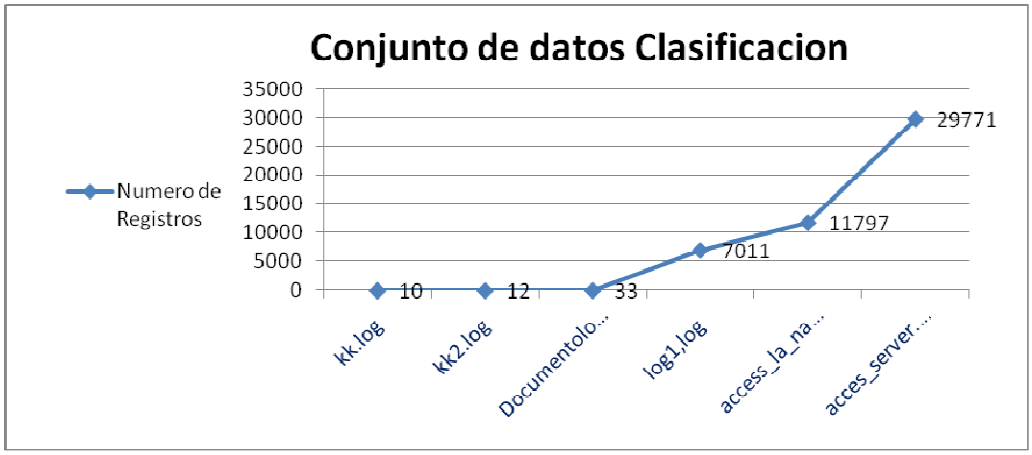
Se realizaron diferentes tipos de pruebas para la tarea de clasificación la cual consistió en verificar el tiempo de ejecución de los algoritmos de clasificación

utilizando diferente atributo clase o target para cada conjunto de datos y medir el rendimiento en cuanto a tiempo empleado. Los resultados se muestran en las tablas 2.20 hasta la tabla 2.28 y en la figura 2.84 hasta la figura 2.95

Tabla 2.20. Conjunto de datos clasificación

Conjunto de datos clasificación		
Nombre del registro	Número de registros	Tamaño KB
kk.log	10	2
kk2.log	12	1
Documentolog.log	33	13
log1,log	7011	3102
access_la_nave.log	11797	19387
acces_server.log	29771	40030

Figura 2.84. Conjunto de datos clasificación.



Para las pruebas realizadas fue seleccionada la opción de configuración por defecto del programa o default, que puede ser observada a continuación:

Tabla 2.21. Configuración default Polaris:

Atributo	**Intervalos**

Duration	Corta: Menos de 1/3 de sesión Media: Entre 1/3 y 2/3 de sesión Larga: Mayor a 2/3 de sesión
Request Amount	Pocas: Menos de 10 solicitudes Media: Entre 10 y 40 solicitudes Muchas: Mas de 40 solicitudes
Download Size	Baja: Menos de -2 Mb Media: Entre 2 y 10 Mb Alta: Mas de 10 Mb

Tabla 2.22. Registro kk2_log

kk2.log			
Discretización	tiempo milisegundos		
	target	C4,5	Mate
Default	IP type	0	0
	Schedule	0	0
	Week day	0	0
	Duration	0	0
	Request Amount	0	0
	Download size	16	47

Figura 2.85.
Gráfica target/tiempo kk2.log clasificación.

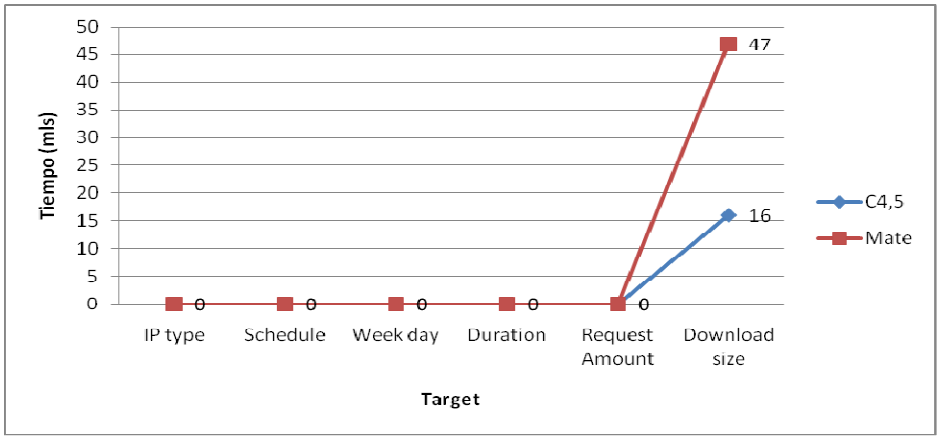


Tabla 2.23. Registro kk.log

kk.log

Discretización	tiempo milisegundos		
	target	C4,5	Mate
Default	IP type	0	15
	Schedule	16	31
	Week day	0	16
	Duration	0	15
	Request Amount	0	16
	Download size	0	16

Figura 2.86. Gráfica target/tiempo kk.log Clasificación.

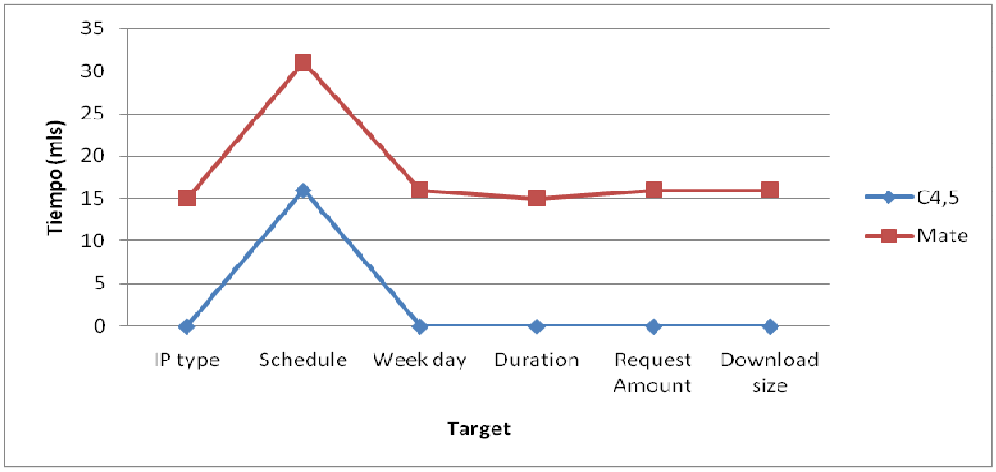


Tabla 2.24. Registro Documentolog.log

Documentolog.log			
Discretización	tiempo milisegundos		
	target	C4,5	Mate
Default	IP type	15	47
	Schedule	0	32
	Week day	0	31
	Duration	0	16
	Request amount	0	16
	Download size	47	109

Figura 2.87. Gráfica target/tiempo Documentolog.log Clasificación.

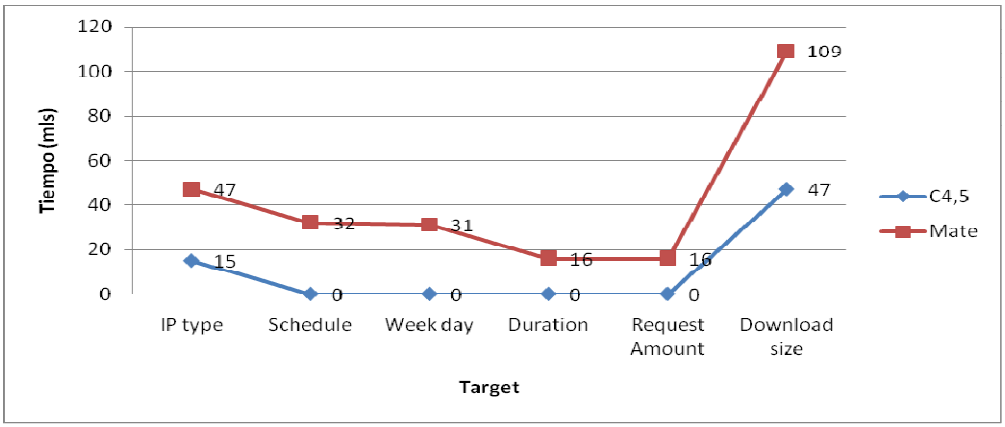


Tabla 2.25. Registro log1.log

log1.log			
Discretización	tiempo milisegundos		
	target	C4,5	Mate
Default	IP type	406	4750
	Schedule	422	4375
	Week day	0	3973
	Duration	219	4969
	Request amount	437	4500
	Download size	594	4500

Figura 2.88. Gráfica Target/Tiempo log1.log clasificación.

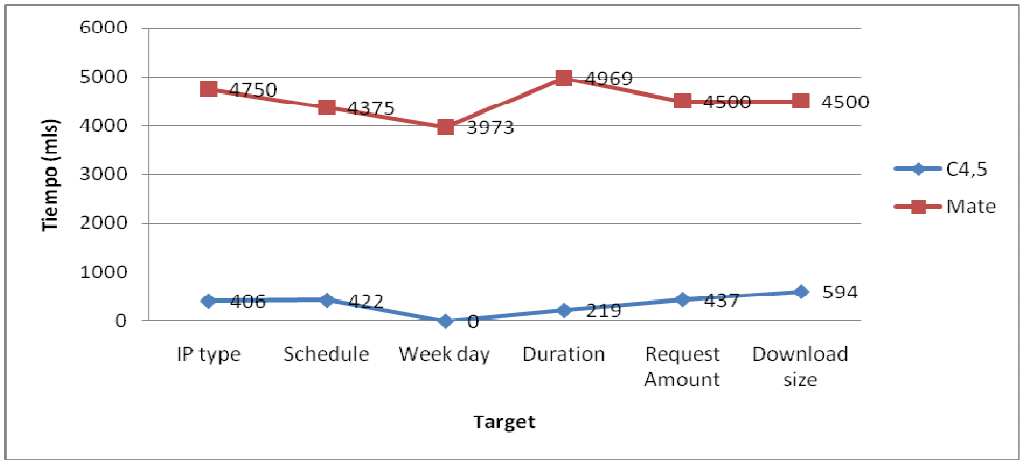


Tabla 2.26. Registro acces_la_nave.log.

acces_la_nave.log			
Discretización	tiempo milisegundos		
	target	C4,5	Mate
Default	IP type	62	1672
	Schedule	63	1595
	Week day	47	2125
	Duration	47	1531
	Request amount	47	1500
	Download size	63	1375

Figura 2.89. Gráfica Target/Tiempo acces_la_nave.log clasificación.

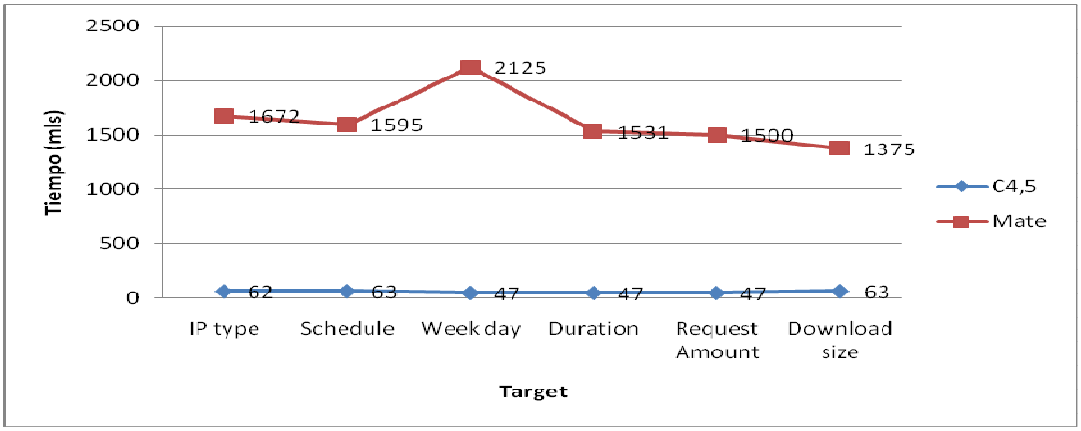


Tabla 2.27. Registro acces_server.log.

acces_server.log			
Discretización	tiempo milisegundos		
	target	C4,5	Mate
Default	IP type	79	1687
	Schedule	63	1437
	Week day	62	2282
	Duration	47	1656
	Request amount	63	1653
	Download size	63	1406

Figura 2.90. Gráfica Target/Tiempo acces_server.log Clasificación.

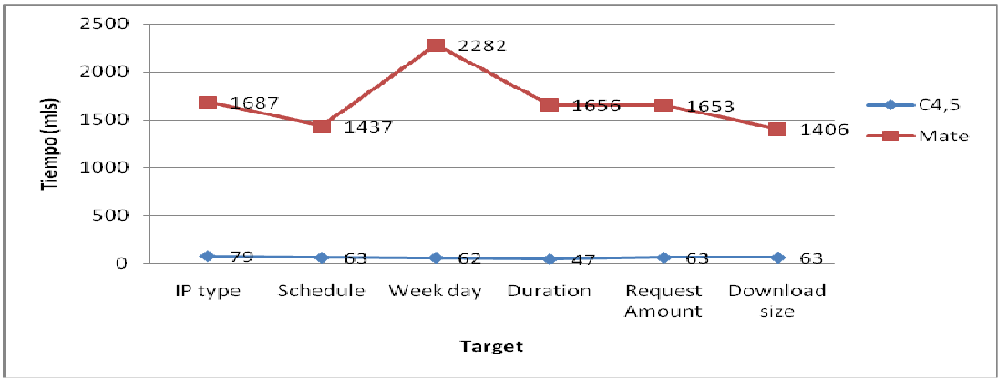
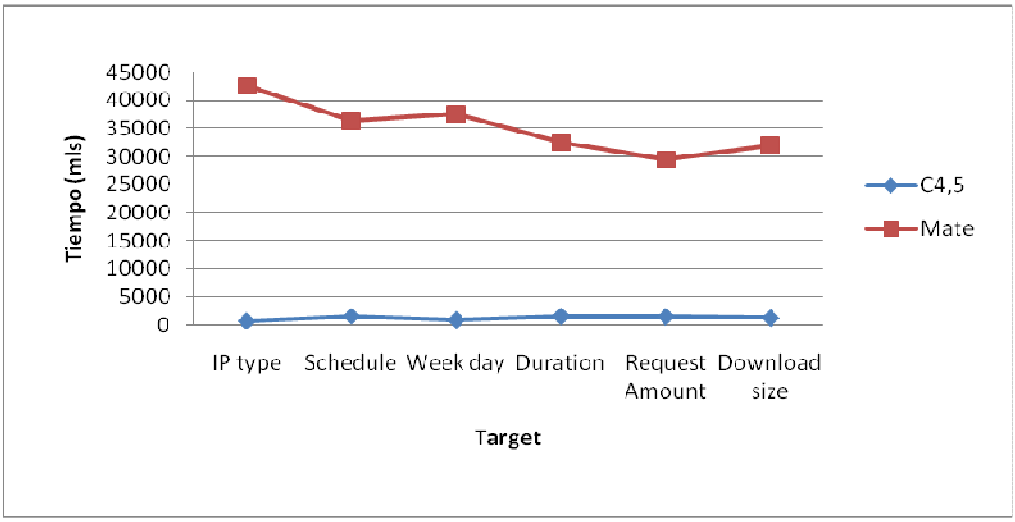


Tabla 2.28. Registro acces_server.log.

www.acces_log			
Discretización	tiempo milisegundos		
	target	C4,5	Mate
Default	IP type	734	42578
	Schedule	1578	36375
	Week day	937	37422
	Duration	1578	32457
	Request amount	1516	29406
	Download size	1282	31922

Figura 2.91. Gráfica Target/Tiempo www.acces_log clasificación.



A continuación y a manera de ejemplo, puede observarse de forma gráfica, la salidas obtenidas de la aplicación de clasificación al registro kk.log, uno de los registros que forma parte del conjunto de datos de prueba.

Figura 2.92. Registros kk.log

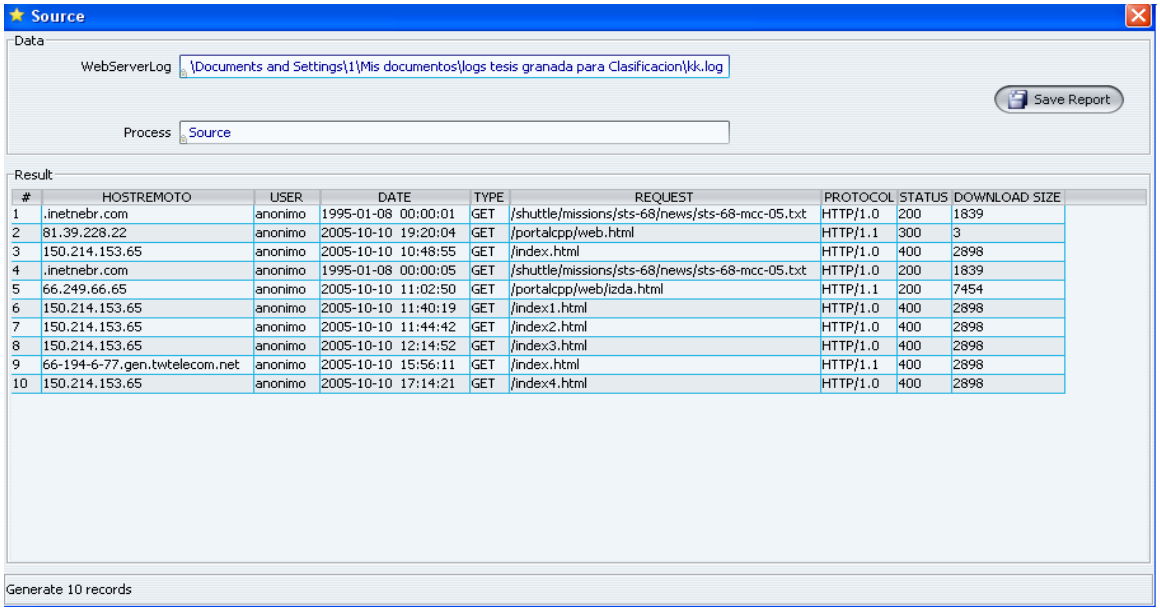


Figura 2.93. Discretización por default del registro kk.log.

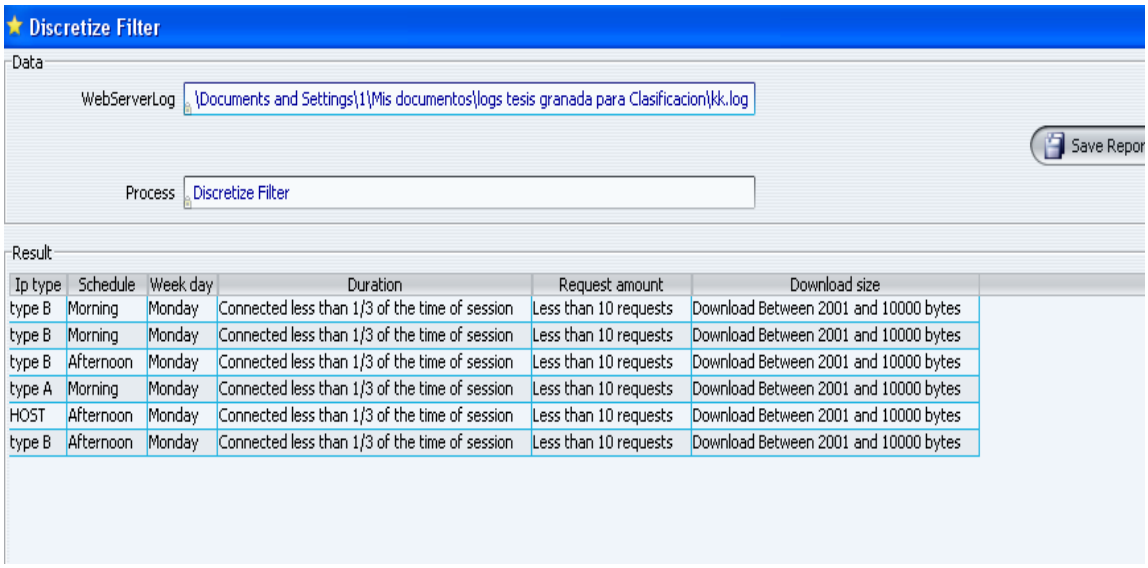


Figura 2.94. Clasification Tree registro kk.log

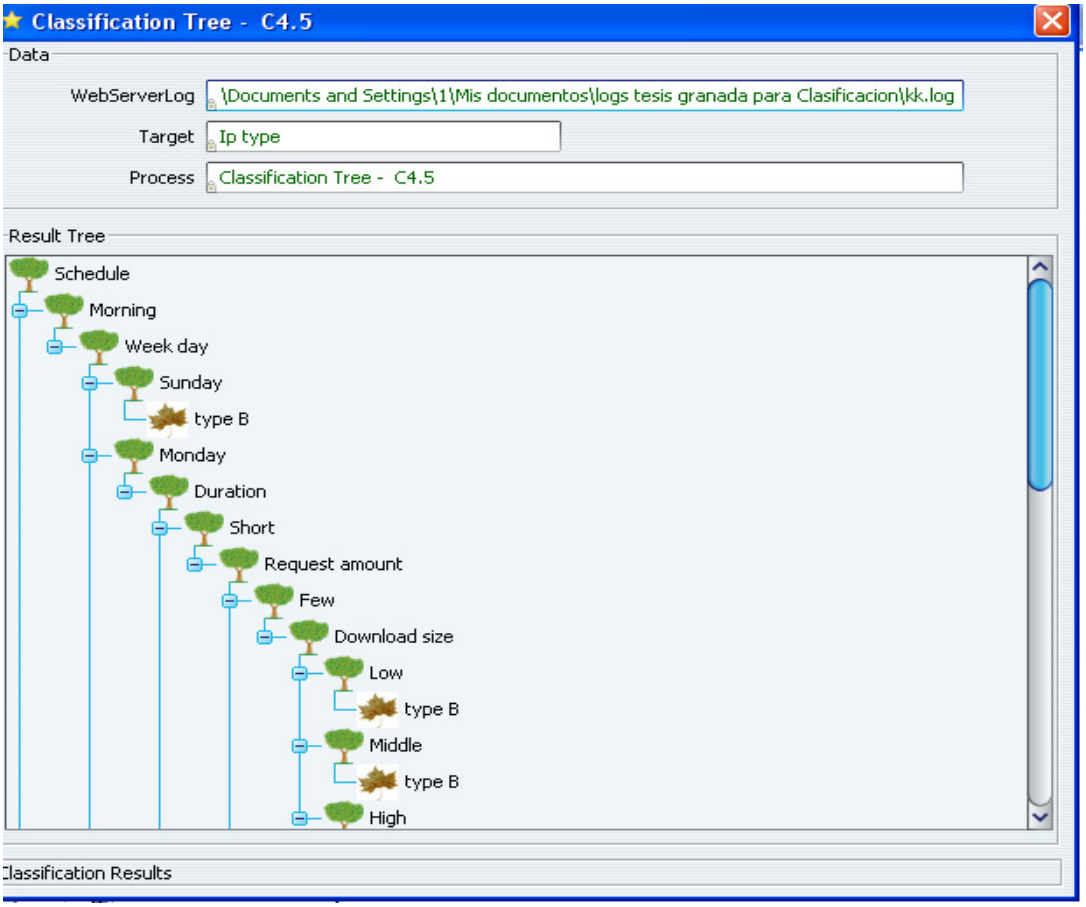
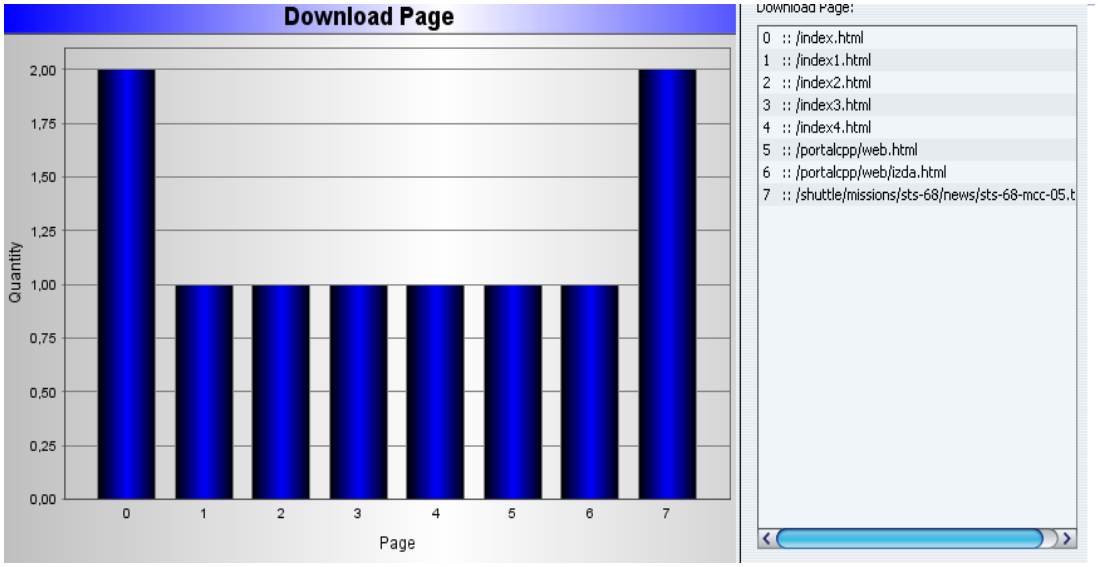

Classification Tree - C4.5
Data
WebServerLog
\Documents and Settings\1\Mis documentos\logs tesis granada para Clasificacion\kk.log
Target
Ip type
Process
Classification Tree - C4.5
Result Tree
Schedule
Morning
Week day
Sunday
type B
Monday
Duration
Short
Request amount
Few
Download size
Low
type B
Middle
type B
High
lassification Results

Figura 2.95. Vista estadística registro kk.log



8.4 EVALUACIÓN DE LAS PRUEBAS DE CLASIFICACIÓN

Las conclusiones a las que se llegaron después del análisis de los resultados obtenidos con los conjuntos de datos para la tarea de clasificación fueron:

- Se puede observar que la elección del atributo clase es determinante para que aumenten o disminuyan los tiempos de ejecución en ambos algoritmos dependiendo de la relevancia de dicho atributo dentro del conjunto.
- El rendimiento del algoritmo Mate se ve considerablemente afectado por el número de registros que se estén analizando. El tiempo empleado por el mismo es mayor cuanto mas grande sea la cantidad de registros a analizar disminuye si el número de registros también disminuye.
- El algoritmo C4.5. ofrece un mejor rendimiento en cuanto a tiempo que el algoritmo Mate.

8.5 PRUEBAS PARA HPG

Las pruebas con HPG fueron realizadas para determinar el rendimiento en cuanto a tiempo empleando por el algoritmo variando los valores de α en 1, 0.5 y 0. El conjunto de datos empleado es el mismo que se empleo para realizar las pruebas de asociación.

Los resultados se muestran en las tablas 2.29 hasta la tabla 2.36 y en la figura 2.84 hasta la figura 2.103.

Tabla 2.29. Registro acces_log HPG.

acces_log		
α	Tiempo	Producciones
1	250	89
0,5	203	89
0	188	79

Figura 2.96. Gráfica N/tiempo acces_log HPG.

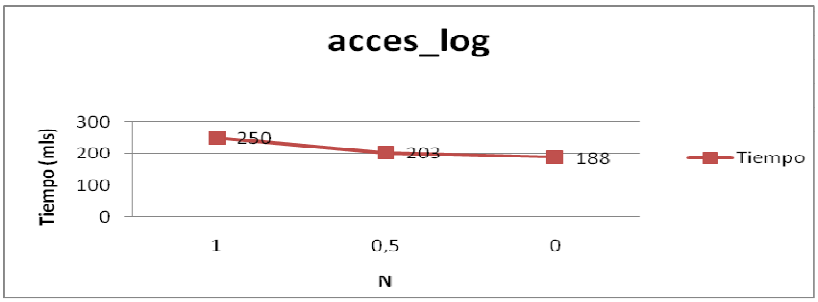


Tabla 2.30. Registro ainfo_acces_log HPG.

ainfo_acces_log		
α	Tiempo	Producciones
1	203	86
0,5	218	86
0	204	69

Figura 2.97. Gráfica N/tiempo ainfo_acces_log HPG.

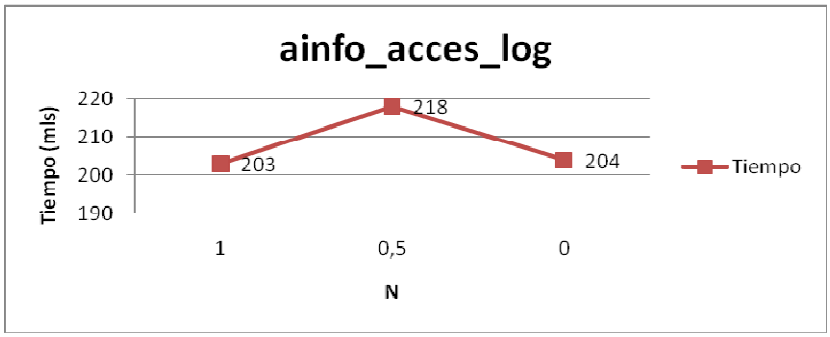


Tabla 2.31. Registro matriculas_acces_log HPG.

matriculas_acces_log		
α	Tiempo	Producciones
1	235	31
0,5	204	31
0	219	20

Figura 2.98. Gráfica N/tiempo matriculas_acces_log HPG.

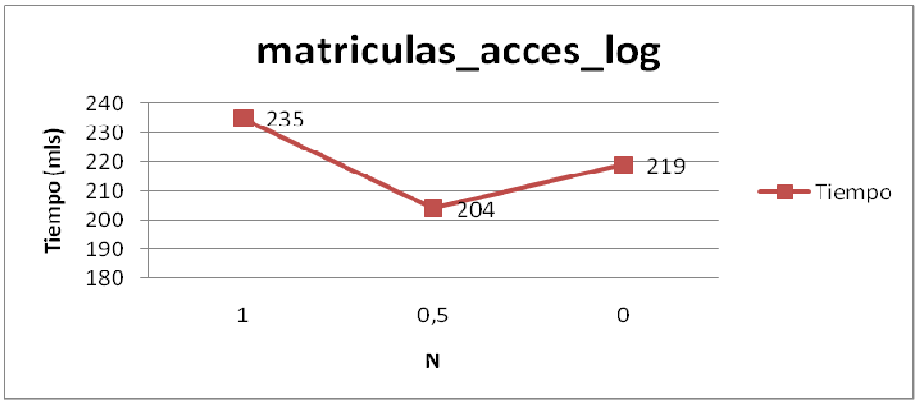


Tabla 2.32. Registro estudiantes_acces_log HPG.

estudiantes_acces_log		
α	Tiempo	Producciones
1	188	39
0,5	203	39
0	187	31

Figura 2.99. Gráfica N/tiempo estudiantes_acces_log HPG.

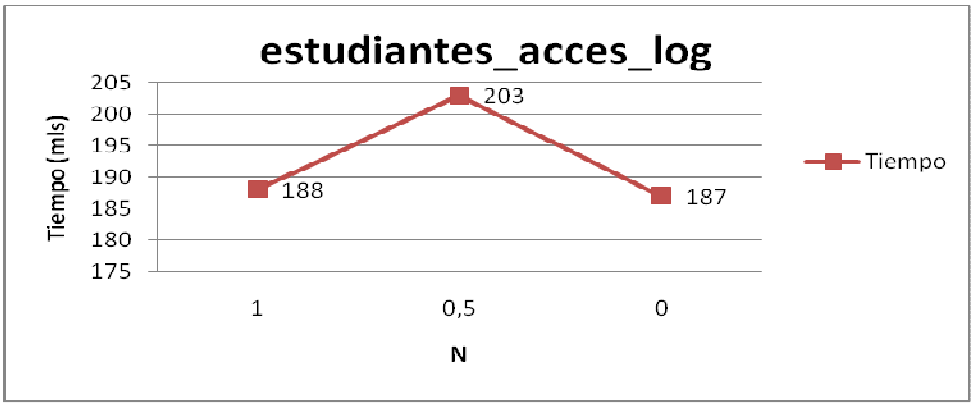


Tabla 2.33. Registro akane_acces_log HPG.

akane_acces_log		
α	Tiempo	Producciones
1	250	277
0,5	238	277
0	212	253

Figura 2.100. Gráfica N/Tiempo akane_acces_log HPG.

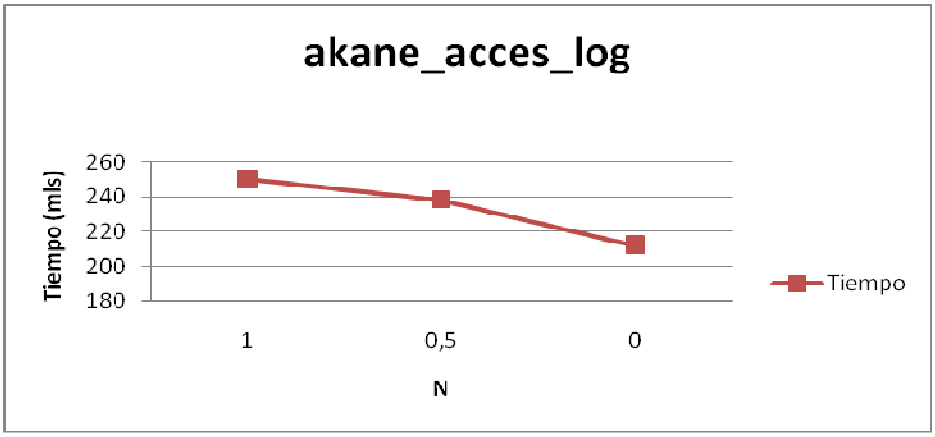


Tabla 2.34. Registro personal_acces_log HPG.

personal_acces_log		
α	Tiempo	Producciones
1	250	153
0,5	234	153
0	218	140

Figura 2.101. Gráfica N/Tiempo personal_acces_log HPG.

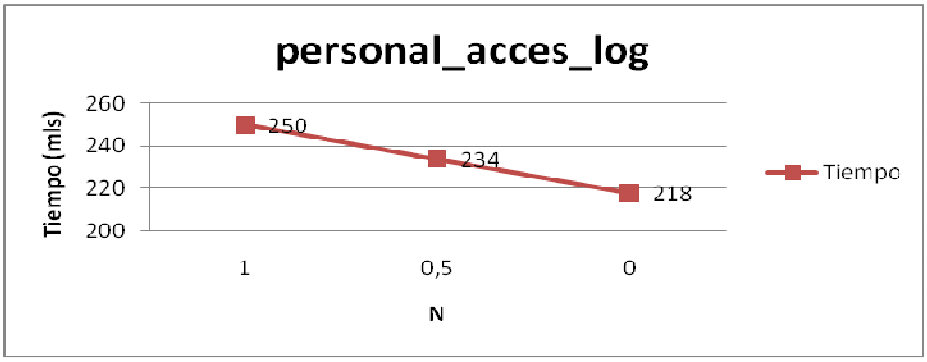


Tabla 2.35. Registro ci_acces_log HPG.

ci_acces_log		
α	Tiempo	Producciones
1	313	261
0,5	312	261
0	281	224

Figura 2.102. Gráfica N/tiempo ci_acces_log HPG.

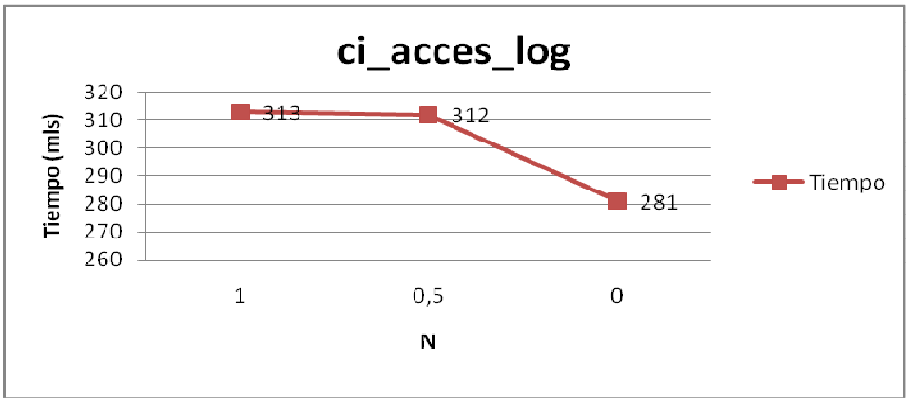
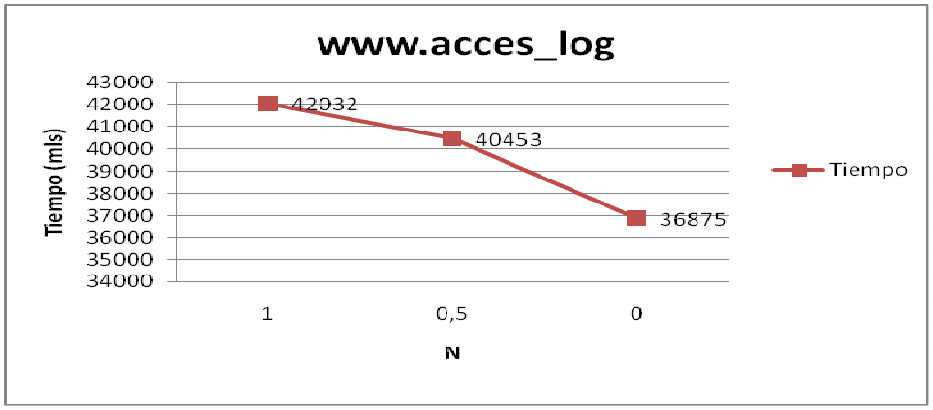


Tabla 2.36. Registro www.acces_log HPG.

www.acces_log		
α	Tiempo	Producciones
1	42032	5541
0,5	40453	5541
0	36875	5174

Figura 2.103. Gráfica N/tiempo www.acces_log HPG



8.6 EVALUACIÓN DE LAS PRUEBAS PARA HPG

Las conclusiones a las que se llegaron después del análisis de los resultados obtenidos con los conjuntos de datos para HPG fueron:

- El número de producciones es mayor cuando α toma el valor de 1 y menor cuando α toma el valor de 0.
- El tiempo de ejecución del algoritmo es directamente proporcional con el valor de α, es decir, en general se obtuvo un tiempo mayor para α=1 y menos para α =0.
- El HPG es una manera compacta de guardar la historia de navegación donde el tamaño de la gramática sólo depende del número de páginas visitado y no de la cantidad de datos del archivo log disponible.

9 CONCLUSIONES

En Polaris se diseñó e implementó una herramienta de minería de uso para la web que brinde apoyo en el análisis del tráfico de un sitio web y soporte la toma de decisiones de los usuarios.

La herramienta sigue los lineamientos de trabajo de investigación desarrollados por el grupo de investigación GRIAS-KDD en cuanto al análisis web mediante minería web y web semántica.

Para el desarrollo del proyecto se abarcaron las etapas del proceso de descubrimiento de conocimiento KDD que van desde la conexión hasta la visualización de los resultados, soportando todo el proceso de minería de datos y las tareas que en el se incluyen.

Se estudiaron varias herramientas para el análisis del tráfico web existentes en el mercado que fueron la base para el reconocimiento de debilidades y fortalezas de las mismas y para ayudarnos a desarrollar una herramienta en la que se integraran todas las fortalezas vistas y se eliminaran las dificultades encontradas.

Se realizó el estudio de técnicas y algoritmos de minería de datos, incluyendo, los nuevos algoritmos para las tareas de asociación y clasificación propuestos por Timarán y se implementaron los algoritmos Apriori, FPGrowth y Equipasso para la tarea de asociación, C4.5 y Mate para la tarea de clasificación; así mismo se implemento el algoritmo minería web de uso HPG.

Polaris es desarrollado bajo código abierto bajo software libre y con metodología orientada a objetos para hacer posible la reutilización de código y el acoplamiento de funcionalidades que contribuyan al mejoramiento de la misma.

La metodología Drag ´n Drog permite una interacción directa entre el usuario y la herramienta, facilitando la visualización de resultados a través de tablas, árboles de decisión, gramáticas y gráficos estadísticos.

Los módulos incluidos en Polaris fueron tres, el módulo de utilidades para realizar la conexión de datos y que además contiene las clases y bibliotecas utilizadas en toda la aplicación, el módulo del kernel que incluye el preprocesamiento de datos que y los algoritmos de minería de datos, y el módulo de la interfaz gráfica que permite la interacción amigable entre el usuario y la herramienta y facilita la visualización de los resultados y reglas obtenidos a través de tablas, gramáticas, reportes textuales, reportes estadísticos y gráficas.

En las pruebas con los algoritmos de minería de datos y de minería web de uso se concluyó:

- Asociación: Equipasso es el algoritmo que presenta un mejor rendimiento en cuanto al tiempo. Aunque FPGrowth maneja un buen tiempo de respuesta. El algoritmo Apriori es el algoritmo menos recomendado.
- Clasificación: El algoritmo que mejor se comporta es el C4.5, sin embargo puede aplicarse Mate si el conjunto de datos es pequeño.
- HPG: Se introduce una nueva técnica exclusiva de la minería de uso en la web para la construcción de los patrones de navegación de los usuarios.

10 TRABAJOS FUTUROS.

Actualmente, se cuenta con la primera versión de la herramienta. Sin embargo se hace necesario sugerir algunas recomendaciones que son el punto de partida para trabajos futuros y nuevas versiones de la herramienta:

- Realizar nuevas pruebas de rendimiento con repositorios de datos reales.
- Implementar la tarea de clustering con el algoritmo de k-medias, partiendo de la base teórica consignada en este documento.
- Adelantar estudios realizados sobre clustering para la web mediante lógica fuzzy.
- Complementar el algoritmo HPG con la implementación del árbol de exploración.
- Disponer de la herramienta como material de apoyo a las electivas de base de datos dentro del programa de Ingeniería de Sistemas para fomentar la utilización de la misma.
- Liberar, compartir y difundir una versión estable de Polaris como herramienta de apoyo para toma de decisiones.
- Publicar las actualizaciones y/o nuevas versiones de Polaris que se encontrarán disponibles en https://developer.berlios.de/projects/polarisweb/
- Asegurar la continuidad del proyecto.

ANEXOS

A.1. MARCAS REGISTRADAS DE SOFTWARE Y HARDWARE

- Apache HTTP Server™ es una marca registrada por Apache Software Foundation.
- ISS™ es una marca registrada por Microsoft Corp.
- Java™ es una marca registrada por Sun MicroSystems.
- Macintosh™ es una marca registrada por Apple Inc.
- NetBeans™ es una marca registrada por Sun MicroSystems.
- Netscape™ es una marca registrada por Netscape Communications Corp.
- Pentium™ es un marca registrada por Intel Corporation.
- PostgreSQL™ es una marca registrada por PostgreSQL Global Development Group.
- Sybase™ es una marca registrada por Sybase Inc.
- Tomcat™. es una marca registrada por Apache Software Foundation.
- Unix® es un marca registrada de The Open Group.
- Windows™ es un marca registrada por Microsoft Corp.
- Fedora Core ™ es una marce registrada por Red Hat Enterprise Linux.

- A.2. EXTENSIONES DE ARCHIVOS

Figura A.1. Extensiones de Archivos

Extensión	**Descripción**
AI	Generado por Adobe Illustrator.
ACV	Compresión Audio en OS/2
ANI	Cursores animados.
ARC	Comprimido, precisa del ARC.
ARJ	Formato de compresión.
ASC	Texto formato ASCII
ASM	Fuentes del lenguaje ensamblador.
ASP	Ejecutables de este lenguaje de Internet.
AVB	Utilizados por el chat de Microsoft
AVI	Vídeo de Windows™
BAS	Fuentes de Basic y Visual Basic
BAK	Copias de seguridad.
BAT	Ejecutable para procesos en lotes (batch)
BGB	Utilizados por el chat de Microsoft
BIN	Binario, generalmente un controlador.
BMP	Gráficos, mapa de bits.
C	Fuentes de versiones del lenguaje C, como C++ o C++Builder.
CDR	Gráficos de Corel Draw.
CHM	HTML compilado.
CFG	El contenido es del tipo de configuración de la aplicación
CMX	Presentaciones de Corel
CNT	Forma parte del sistema de ayudas HLP.
COM	Ejecutable estándar, en formato binario. Generalmente una orden o comando del sistema.
CPT	Imagen de Corel Photo Paint.
CUR	Fichero de cursores.
DAT	Muy genérico. Distintos programas lo utilizan como extensión de archivo de datos.
DB	Bases de datos Paradox.
DBF	Bases de datos tipo dBase.
DCU	Resultado de la compilación en Delphi.
DLL	Librería de enlace dinámico.
DOC	Documento Word (Microsoft).

DOT	Macros de Word (Microsoft)
DPR	Archivo principal de aplicaciones Delphi.
DRV	Un controlador (driver).
DVG	Gráficos de AutoCad
ESP	PostScript (se le denomina de tipo "encapsulado")
EXE	Ejecutable estándar, en formato binario.
FON	Fuentes de letra.
FPX	Gráficos, Kodak Flash Pix
FRM	Formularios de varios lenguajes como V. Basic o Clipper.
FT8	Gráfico de FreeHand 8 de Macromedia.
GIF	Gráfico que admite color de 8 bits.
H	Fichero de cabecera en lenguajes como C.
JAR	Almacenamiento de JS (ficheros de JavaScript).
JPG	Gráfico que admite color RGB de 24 bits.
JS	De JavaScript.
HLP	De ayuda (help)
HTM o HTML	Páginas web visibles con navegadores de Internet.
ICO	Gráfico de icono, generalmente de 32x32 y 16 colores
INI	Ficheros de inicio de Windows™.
JS	Ejecutables de JavaScript
LHA	Comprimido, utiliza el LHA.
MAC	Gráficos, mapa de bits de Corel Mac Paint
MDB	Base de Datos Access
MOV	Animaciones de QuickTime
MPG	Animaciones con compresión MPEG.
MP3	Audio/video comprimido, no requiere descompresor previo.
NET	Generalmente ficheros controladores de bloqueos de usuarios en red.
OBJ	Objetos, el resultado de la compilación.
OCX	Antiguamente ficheros llamados por ese nombre, hoy la extensión de ActiveX
OLD	Suele utilizarse al reemplazar un archivo, como copia de seguridad.
PAS	Fuentes de Delphi.
PCD	Gráficos, Kodak Photo-CD
PCT	Gráficos de Macintosh (PICT)
PCX	Gráficos de Paintbrush.
PDF	Estándar de Adobe, requiere el lector Acrobat Reader.
PFG	Fuentes de letra en Adobe
PIF	Accesos directos de Windows™ para MS-DOS.
PRG	Programas fuentes en entornos dBase, Fox, Clipper, etc.

PRT	Documento formateado para impresión.
PS	PostScript
PSD	Gráficos de Adobe PhotoShop
RAR	Comprimido, precisa de la herramienta Rar.
RAW	Gráficos de Corel Raster
REG	Entrada al registro de Windows™.
RES	Fichero de Recursos, se utiliza en los lenguajes visuales.
RTF	Texto con formato (suele denominarse "enriquecido")
SCT	Gráfico, mapa de bits de Scitex CT
SNM	Correos electrónicos con NetScape.
SQL	Fuentes en este lenguaje, utilizados por bases de datos tipo Oracle.
SYS	Ejecutable utilizado por el Sistema Operativo
TGA	Gráfico, mapa de bits de Targa.
TIF	Gráfico de alta resolución.
TTF	Fuentes de letra True Type
TMP	Temporales.
TXT	Texto sin formato.
USR	Suele utilizarse como fichero de usuarios.
VBP	Proyecto o archivo principal de una aplicación Visual Basic
VBX	Como los OCX son extensiones tipo OLE de Windows™.
WAV	De sonido de Windows™.
WIN	De Sistema de Windows™.
WCM	Macros de Corel WordPerfect
WMF	Metaarchivo de Windows™ (archivo con información sobre otros)
WIZ	Asistente creado por Word (Microsoft)
WPD	Documento de Corel Word Perfect.
WPG	Documento gráfico de Word Perfect.
WPT	Plantillas de Word Perfect
WRI	Texto de Write.
WS_	Documento de WordStart versión _ (depende del número)
XLS	Hoja de cálculo de Excel (Microsoft)
XML	Formato de página derivada de HTML
Z	Comprimido. Entornos Unix®.
ZIP	Comprimido. Precisa descompresión a través de WinZip para poderse utilizar.

A.3. MANUAL DE USUARIO

ÍNDICE

1. INTRODUCCIÓN

1.1. ¿QUE ES POLARIS?

Polaris es una herramienta de minería web de uso desarrollada por estudiantes pertenecientes al Grupo de Investigación GRIAS (Grupo de Investigación aplicado a Sistemas) dirigido por el Dr. Ricardo Timaran Pereira, Ph.D. del Programa de Ingeniería de Sistemas de la Universidad de Nariño de la ciudad de Pasto (Nariño – Colombia).

Polaris es un software que realiza minería web de uso a partir un log de acceso de servidores web: Apache HTTP Server ™ y IIS Server™.

Polaris tiene una completa variedad de algoritmos de asociación y clasificación, y un algoritmo de minería de uso llamado HPG (Gramática Probabilística de Hipertexto), también tiene siete formatos de visualización que ayuda a entender todo el proceso de minería.

Polaris esta continuamente en desarrollo y en constante optimización.
Para comentarios o sugerir ideas para mejorar, por favor escribirnos a polaris@gmail.com o entrar a la página del sitio http://polaris.berlios.de.

1.2. REQUERIMIENTOS

Para el correcto funcionamiento de la herramienta se necesita tener instalado el siguiente software:

- Postgres 8.2 o superior
- La máquina virtual de Java™ jre 6 o superior
- Componente de Java™ 3D 1.5.0 o superior
- Usuario del SGBD Postgres que tenga permiso de creación de base de datos
- Instalador de Polaris v1.0

2. INSTALACIÓN DE POSTGRESQL™ 8.2

2.1. BAJO WINDOWS™

Requisitos Previos: Antes de instalar PostgreSQL™, se debe comprobar si se tiene el siguiente software en tu ordenador:

- **Un descompresor de ficheros Zip:** Windows™ XP los maneja automáticamente; si no tienes uno disponible, instala IZArc, por ejemplo.

- **Sistema de Archivos NTFS**: Es necesario que la partición donde se instala postgreSQL™ se encuentre en este formato, por cuestiones de seguridad.

Instalación

- Descargar el instalador de postgresql™ 8.2 "*postgresql-8.2.3-1.zip*" desde *http://www.postgresql.org/download*
- Descomprimir el archivo .zip descargado
- Entrar a la carpeta extraída *postgresql-8.2.3-1*
- Ejecutar el instalador *postgresql-8.2.msi* y seguir los pasos del instalador

2.2. BAJO LINUX™

Información del paquete:

Web:	http://www.postgresql.org
Archivo a descargar:	postgresql-8.2.4.tar.gz

Se descomprime el archivo con el siguiente comando:

```
[root@localhost ~]# tar -zxf postgresql-8.2.4.tar.gz -C /opt/
```

En el directorio donde se descomprime el archivo:

```
[root@localhost ~]# cd /opt/postgresql-8.2.4/
```

Se crea un archivo de compilación llamado compile.sh para especificar las opciones de compilación y preservarlas en caso de error (se utiliza nano, pero no es de carácter obligatorio):

```
[root@localhost postgresql-8.2.4]# nano compile.sh
```

El contenido de este archivo será:

```
LDFLAGS=-lstdc++ ./configure \
--prefix=/usr/local/pgsql \
--with-perl \
--with-python \
--with-krb5 \
--with-openssl
```

Se guarda el archivo con CTRL+O y [ENTER], luego se sale del editor con CTRL+X.

Se cambia los atributos para hacerlo ejecutable:

```
[root@localhost postgresql-8.2.4]# chmod 755 compile.sh
```

Luego se ejecuta el script:

```
[root@localhost postgresql-8.2.4]# ./compile.sh
```

Se compila el paquete:

```
[root@localhost postgresql-8.2.4]# make
```

Se instala el paquete:

```
[root@localhost postgresql-8.2.4]# make install
```

Configuración Adicional para PostgreSQL™: Una vez todos los pasos anteriores han sido ejecutados sin ningún problema, se procede a realizar los últimos ajustes:

- Se debe crear un usuario para la administración de postgreSQL™:

```
[root@localhost postgresql-8.2.4]# adduser postgres
```

- Se asigna una contraseña al usuario que acabamos de crear:

```
[root@localhost postgresql-8.2.4]# passwd postgres
```

- Se crea el directorio "data" (donde postgreSQL™ almacenará las bases de datos creadas):

```
[root@localhost postgresql-8.2.4]# mkdir /usr/local/pgsql/data
```

- Al directorio, se le asigna su nuevo propietario:

```
[root@localhost postgresql-8.2.4]# chown postgres /usr/local/pgsql/data/
```

- Ya que se trabaja con postgreSQL™, el login debe realizarse con el usuario adecuado debe ser (postgreSQL™ no funciona desde una cuenta de super-usuario):

```
[root@localhost postgresql-8.2.4]# su – postgres
```

- Ahora se a inicializa el motor de base de datos:

  [postgres@localhost ~]# /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data/ &

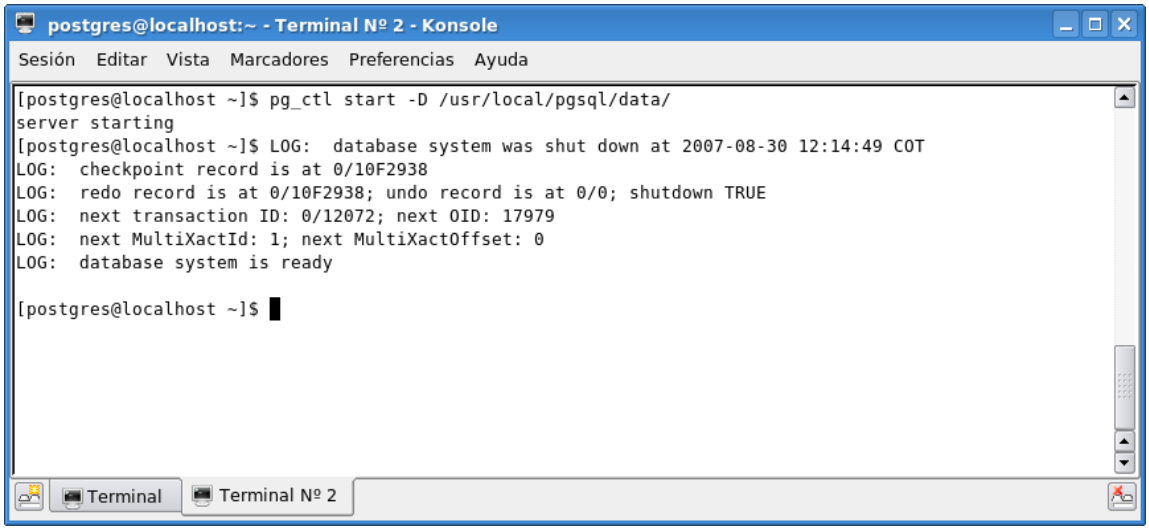
Finalizado el paso anterior, se observa un mensaje de que el servidor de bases de datos está listo para trabajar.

- Para probar que todo va bien, ya sería posible arrancar el servidor postgreSQL™:

  [postgres@localhost ~]# /usr/local/pgsql/bin/pg_ctl start -D /usr/local/pgsql/data/ &

Todo habrá salido correctamente si se aprecia la figura siguiente:

Figura 2.104 Ventana Database System ready



Significa que todo va bien hasta aquí y ya es posible crear las bases de datos.

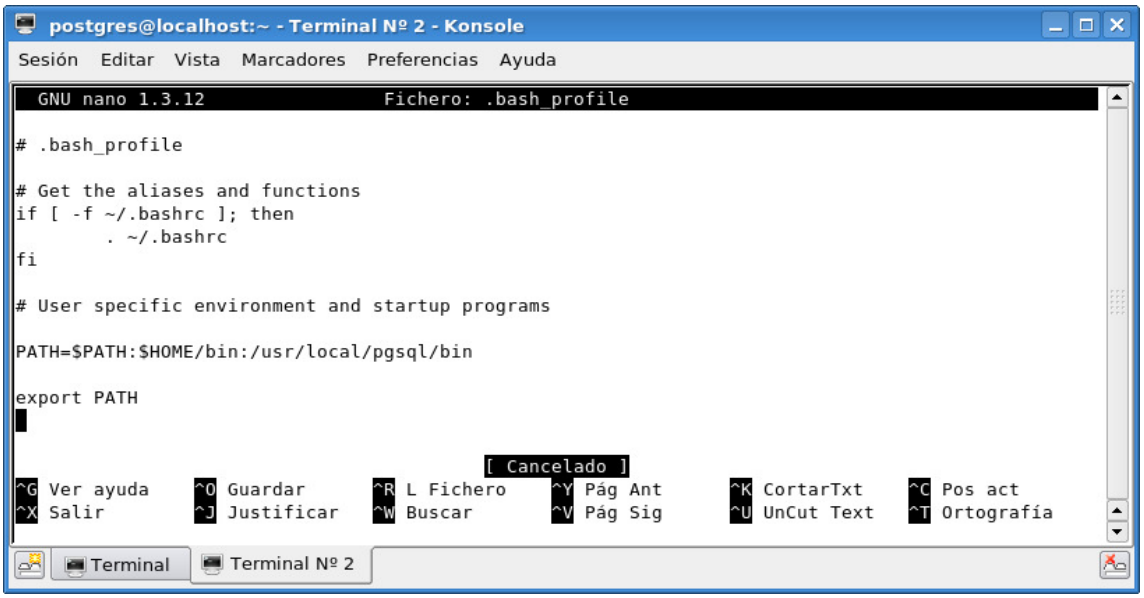
Los pasos para "arrancar" el servidor de bases de datos son un poco tediosos, así que se configura el sistema para que postgreSQL™ se inicie automáticamente:

- Primero se configuran las variables de entorno (archivo .bash_profile en el directorio home) en un perfil de usuario (usuario postgres), esto para no tener que utilizar las rutas completas en la ejecución de las tareas (ej. Creación de bases de datos):

[postgres@localhost ~]# nano .bash_profile

EL objetivo es apreciar la siguiente figura2.105:

Figura 2.105 Ventana configurar variables de entorno



Se observa que se añadió el directorio "/usr/local/pgsql/bin" donde están las utilidades de postgreSQL™, guardamos los cambios con CTRL+O, luego [ENTER] y salimos con CTRL+X.

- Para que los cambios surtan efecto, se debe abandonar el usuario postgres e ingresar nuevamente:

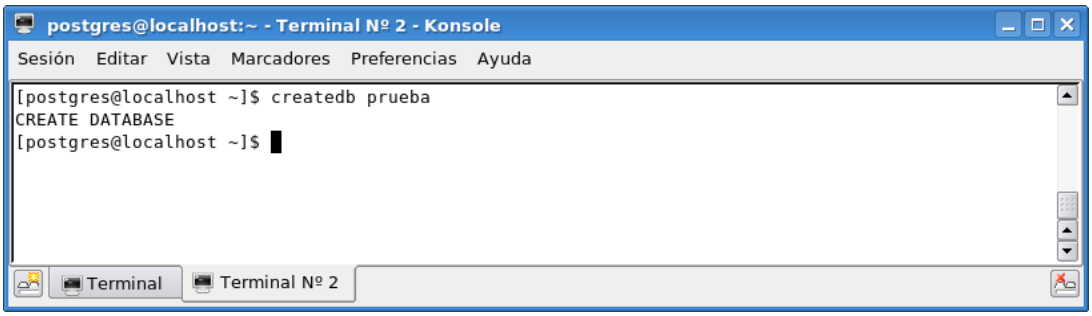
[postgres@localhost ~]# exit
[root@localhost postgresql-8.2.4]# su – postgres

- Se crea la primera base de datos:

[postgres@localhost ~]# createdb prueba

Si todo esta correcto aparece la imagen que se aprecia en la figura 2.106:

Figura 2.106 Creación exitosa base de datos



Ahora se configura el arranque automático de PostgreSQL™, para esto los fuentes del paquete incluyen un script encargado de esta tarea, así que debe ser copiado y su ruta debe ser especificada.

[root@localhost postgresql-8.2.4]# cp contrib/start-scripts/linux /etc/init.d/postgresql

Se cambian los atributos del script para hacerlo ejecutable:

[root@localhost postgresql-8.2.4]# chmod 700 /etc/init.d/postgresql

Se añade el servicio al sistema con:

[root@localhost postgresql-8.2.4]# chkconfig –add postgresql

Ahora, el servidor debe arrancar automáticamente en el próximo inicio del sistema

3. INSTALACIÓN

Para esto es necesario el archivo de instalación ***Setup Polaris.exe*** y seguir los siguientes pasos:

- Hacer doble clic en el archivo
- Seguir las instrucciones del instalador
- Aceptar la licencia GPL (General Public License)
- Hacer clic en install para comenzar la instalación, esperar a que instale todos los archivos necesarios y
- Ejecutar la aplicación (para esto es necesario los requerimientos no incluidos).

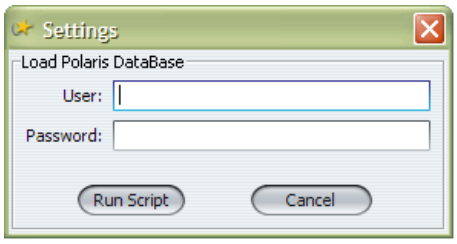
4. CONFIGURACIÓN

Se presiona el botón de la barra de herramientas que indica la configuración y aparece un mensaje de advertencia:

Figura 3.0. Settings

Una vez se lee el mensaje y se hace clic en el botón aceptar, aparece la ventana:

Figura 3.1. Ventana Settings.

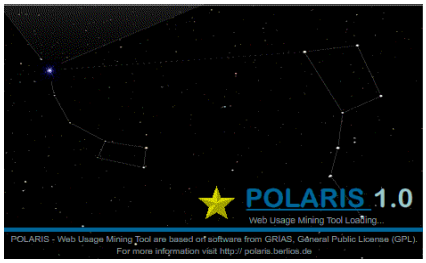


Se escribe el usuario con derechos de creación de bases de datos y usuario, y su contraseña, inmediatamente se hace clic en **Run Script**, si no hay ningún error aparece un mensaje de éxito, en caso contrario un mensaje de error. Si todo salió bien, se puede utilizar el software perfectamente.

5. EL ENTORNO DE LA APLICACIÓN

Al ejecutar el programa, lo primero que se encuentra es una imagen de carga, una vez la imagen desaparece se abre la aplicación.

Figura 3.2. Imagen de Carga.

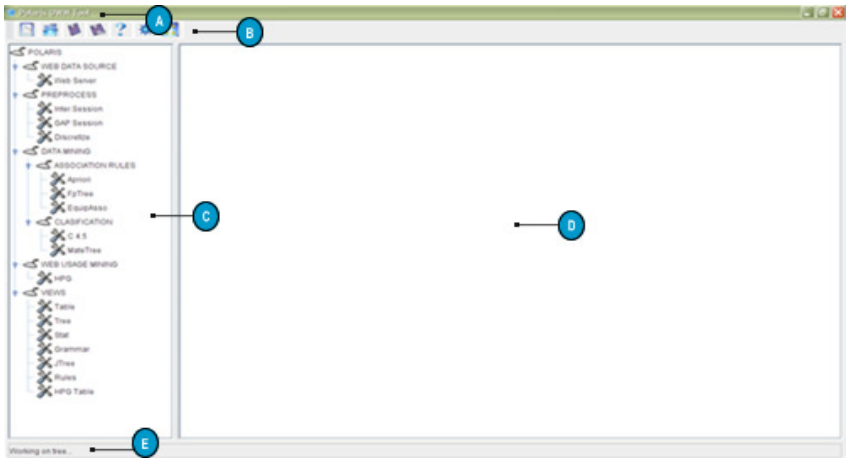


5.1. DESCRIPCIÓN PANTALLA PRINCIPAL

Se encuentra una pantalla con 5 áreas:

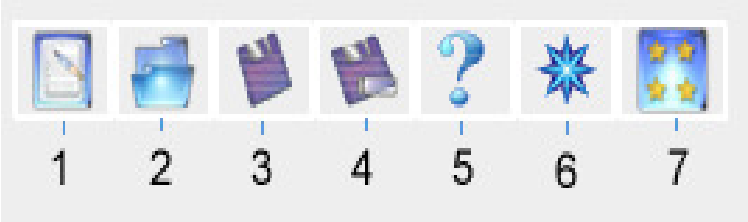
- **A -** La barra de título,
- **B -** La barra de opciones de programa,
- **C -** El área de las herramientas,
- **D -** Un área de trabajo en blanco
- **E -** La barra de estado.

Figura 3.3. Áreas pantalla principal.



5.2 BARRA DE OPCIONES DE PROGRAMA

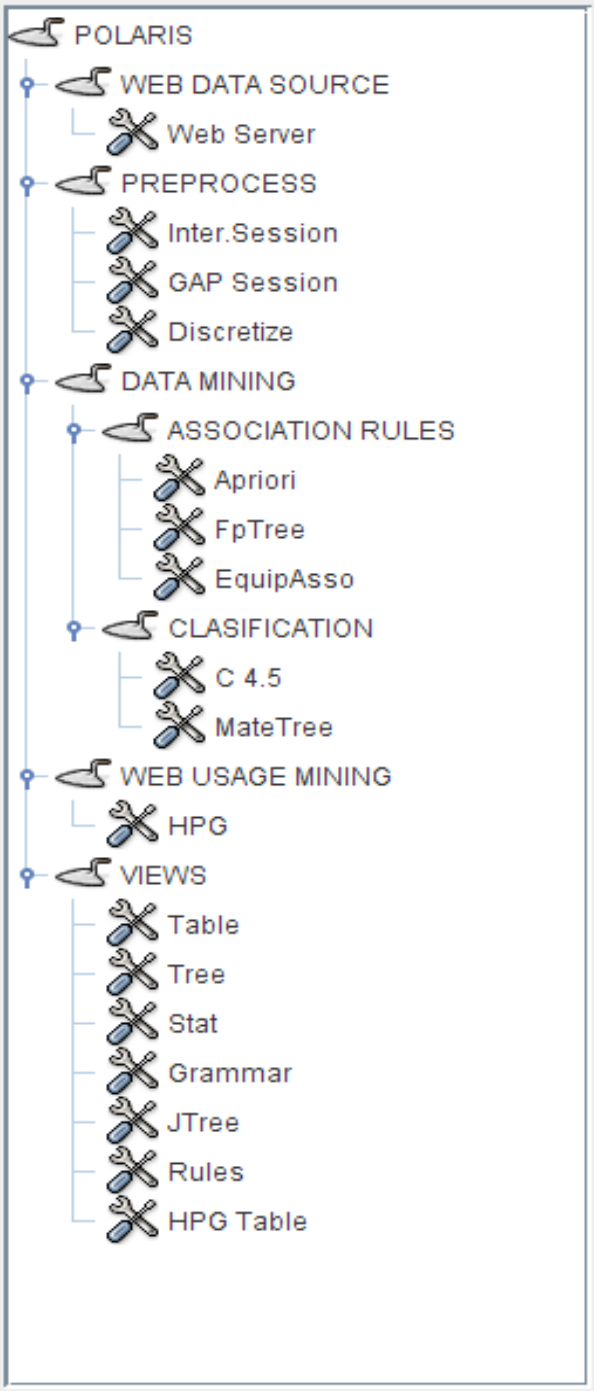
Figura 3.4. Barra de opciones.



En esta se encuentran todas las acciones que se pueden hacer con respecto a proyectos y a la aplicación en general. Las cuales son:

- **1 -** Nuevo Proyecto
- **2 -** Abrir Proyecto existente
- **3 -** Guardar Proyecto
- **4 -** Guardar Como
- **5 -** Ayuda
- **6 -** Acerca de
- **7 –** Configuración

Las acciones nuevo, abrir y guardar proyecto se encargan de manejar el archivo de la aplicación POLARIS con extensión pol (*.pol) pero además las acciones abrir y guardar como manejan archivos XML, así pues, el programa tiene dos opciones de archivo (pol, xml) para manejar proyectos.



La acción ayuda abre la página principal de la ayuda.

El Acerca de muestra quienes son los desarrolladores, el director del proyecto; quienes aportaron de una o muchas maneras para que este software fuera una realidad; y la licencia del software.

5.2. ÁREA DE HERRAMIENTAS

Figura 3.5. Área de Herramientas

El área de herramientas que se puede observar es la que se utiliza para lograr armar el algoritmo de minería de uso.

Los componentes son:

- Categoría Web Data Source

En esta categoría están los elementos que son fuentes de datos web.

Web Server: Indica que la fuente de datos es un log de un servidor web.

- Categoría Preprocess

Aquí se encuentran los elementos que se utilizan para hacer un previo proceso antes de realizar minería de datos.

Inter.Session: Indica que se realizara la transformación de la fuente a una sesión por intervalos.

GAP Session: La transformación de la fuente se hace a una sesión GAP, es decir, separación por tiempo muerto.

Discretize: Se hace una transformación de las sesiones en una nueva sesión necesaria para el módulo de clasificación.

- Categoría Data Mining

Los elementos de esta categoría son dos subcategorias que indican los modelos de minería de datos:

- Categoría Association Rules

Se encuentran la colección de algoritmos de minería de datos que generan reglas de asociación.

A priori: Indica el elemento que genera reglas de asociación a través del algoritmo a priori.

FP Tree: Indica el elemento que genera reglas de asociación a través del algoritmo fp tree.

EquipAsso: Indica el elemento que genera reglas de asociación a través del algoritmo EquipAsso.

- Categoría Classification

Colección de Algoritmos de Clasificación.

C4.5: Genera el árbol de decisión a través del algoritmo c4.5.

Mate Tree: Genera el árbol de decisión a través del algoritmo Mate Tree.

- Categoría Web Usage Mining

Técnicas de minería desarrolladas para encontrar patrones de navegación, y así, aportar a la minería web de uso.

HPG: Gramática Probabilística de Hipertexto, algoritmo que permita generar un grafo del sitio donde sus arcos manejen un arco de dirección con probabilidad.

- Categoría Views

Las posibles vistas de la aplicación, dependiendo del elemento al que se conecten.

Table: Indica que la vista es una tabla de datos.

Tree: La vista que se genera es un árbol grafico vertical.

Stat: Es un elemento donde se muestran las posibles estadísticas de un log.

Grammar: Indica que la vista es un grafo donde se observara la gramática.

JTree: La vista que se genera es un árbol horizontal.

Rules: Es un elemento donde se muestra una listado de las reglas resultantes por un algoritmo.

HPGTable: Indica que la vista es una tabla de producciones HPG.

5.3. ÁREA DE TRABAJO

El área de trabajo es donde se arma el algoritmo de minería de uso con los elementos del área de herramientas, es decir, se arrastra el componente que se va a utilizar, a la posición que desee dentro del área de trabajo e inmediatamente aparecerá un nodo que identifica al componente.

5.3.1. Nodo

Figura 3.6. Nodo.

Un nodo es un botón que identifica el elemento arrastrado de la barra de herramientas, este posee el nombre y un icono que lo identifica.

Cada nodo posee diferentes opciones al hacer clic derecho sobre este:

Move: Da la opción de ubicar el nodo en una nueva posición.

Delete: Se elimina el nodo del área de trabajo.

Add Connection: Adhiere una conexión con un nodo con el cual sea válido conectarse. (Web Server no tiene esta opción)

Remove Connection: Elimina la conexión que tenga este nodo.

Settings: Si el nodo requiere de configuración aparece esta opción.

Run: Se ejecuta la función del nodo.

5.3.2. Conexión

Figura 3.7. Conexión.

Una conexión se realiza con clic derecho en el nodo que desea, y escoger la opción Add Connection, y luego se escoge el nodo con el que se va a conectar.

La conexión se hace de un nodo que desea recibir la información de un nodo previo, siempre y cuando el nodo previo sea válido.

A continuación un listado de los nodos que pueden hacer conexión y sus posibles nodos previos:

Tabla 2.37. Conexiones entre nodos

NODOS	Se conecta con…
Inter.Session	Web Server
GAP.Session	Web Server
Discretize	Inter.Session GAP.Session
Apriori	Inter.Session GAP.Session
FpTree	Inter.Session GAP.Session
EquipAsso	Inter.Session GAP.Session
C4.5	Discretize
Mate Tree	Discretize
HPG	Inter.Session GAP.Session

Table	Web Server Inter.Session GAP.Session Discretize
Tree	C4.5 Mate Tree
Stat	Web Server
Grammar	HPG
JTree	C4.5 Mate Tree
Rules	Apriori Fp Tree EquipAsso C4.5 Mate Tree
HPGTable	HPG

5.3.3. Cambios de Estado de un Nodo. Un nodo tiene 3 Estados (los nodos de vista tienen dos) y durante la modificación del mismo hace el cambio de estado.

Este cambio se observa en el borde del nodo, así:

- Borde Negro:

Figura 3.8. Estados Nodo: Sin Modificaciones

No se ha realizado ninguna modificación al nodo.

- Borde Azul:

Figura 3.9. Estados Nodo: Modificaciones

La configuración del nodo se ha realizado.

- Borde Verde:

Figura 3.10. Estados Nodo: Ejecutado.

El nodo ha sido ejecutado.

6. EJEMPLO PASO A PASO

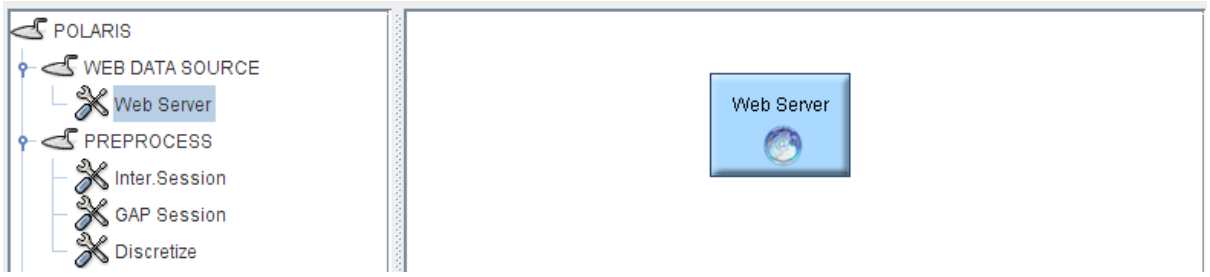
Una vez abierta la aplicación, un Nuevo proyecto es abierto automáticamente, por tanto es posible realizar un algoritmo de minería de uso.

Para armar el algoritmo se debe tener en cuenta que este siempre empieza por un WEB DATA SOURCE, por tanto, ninguna otra herramienta se habilita hasta que un elemento de este sea arrastrado al área de trabajo.

6.1. ARRASTRAR UN WEB DATA SOURCE

Se selecciona el elemento **Web Server** y lo se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo), la pantalla se verá de la siguiente manera:

Figura 3.11. Arrastrar un Web Data Source.



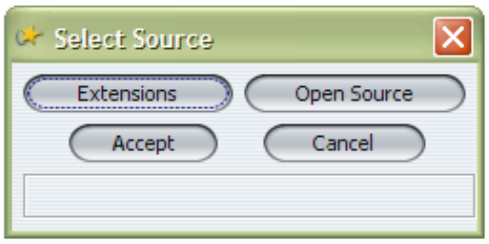
Una vez arrastrado el Web Server es posible arrastrar cualquier otra herramienta. Existen dos opciones para hacer en el momento:

- Se puede configurar y ejecutar el nodo
- Se puede armar el algoritmo de uso y luego hacer la ejecución de los nodos en su totalidad

6.2. CONFIGURAR UN WEB SERVER

Lo primero que se debe hacer es dar clic derecho en el nodo Web Server, y seleccionar la opción Settings, entonces aparecerá una pequeña ventana:

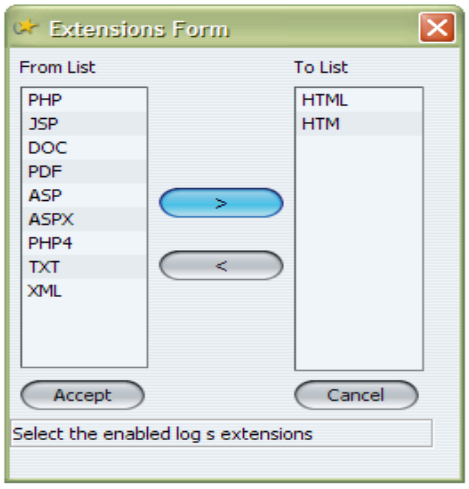
Figura 3.12. Select Source Form



Hay dos opciones de configuración:

- Extensions
  Se establece las extensiones válidas para realizar el análisis.

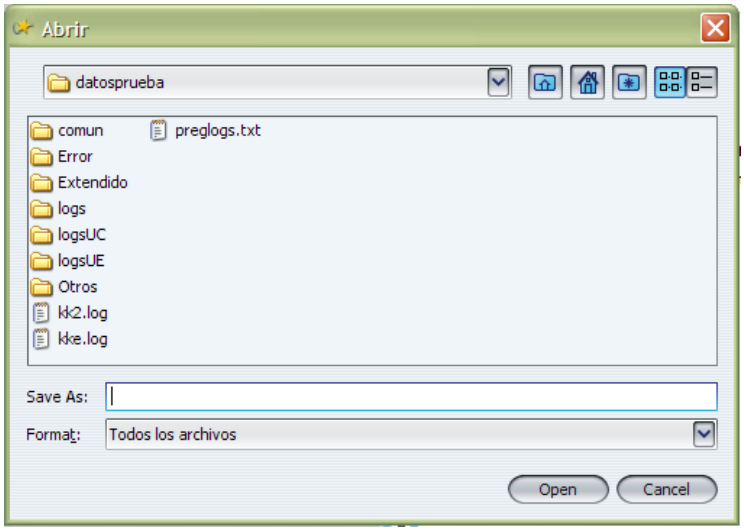
Figura 3.13. Extensions Form.



En la lista izquierda se encuentra el listado de posibles extensiones y en la lista derecha se encuentran las extensiones válidas para el análisis, esta última lista se arma se selecciona una extensión de la lista izquierda y presionando el botón (>) – en la imagen de color azul – o simplemente con doble clic, de la misma manera se puede retirar de la lista derecha se selecciona la extensión y presionando el botón (<) o con doble clic así vuelve la extensión a la lista izquierda. Para este ejemplo, se toma como extensiones válidas: HTML, HTM.

- Open Source
  Se escoge el archivo web server log que se va a analizar. Estos archivos tienen en su mayoría extensión .log, la ventana que aparece es:

Figura 3.14. Abrir Form.



Para este ejemplo se tomar el log de prueba: kk2.log.

El nodo ha cambiado de estado porque ya ha sido configurado.

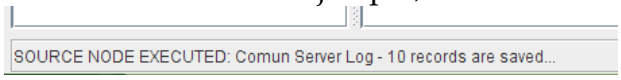
6.3. EJECUTAR WEB SERVER

Una vez configurado el nodo (porque si no se configura, el nodo no se podrá ejecutar) se hace clic derecho en el nodo y se selecciona la opción **Run**.

Al terminar la ejecución el nodo cambia de estado y aparece un mensaje en la barra de estado con el resultado de la ejecución:

- Si todo ha resultado con éxito el mensaje es:
  SOURCE NODE EXECUTED: ***(Tipo Log) – (Cantidad de registros)*** *records are saved…*

Para el ejemplo,



- Si ocurre algún error el mensaje es:

ERROR: ***(mensaje de error)***

6.4. ARRASTRAR DE PREPROCESS UNA SESIÓN Y CONECTAR A WEB SERVER

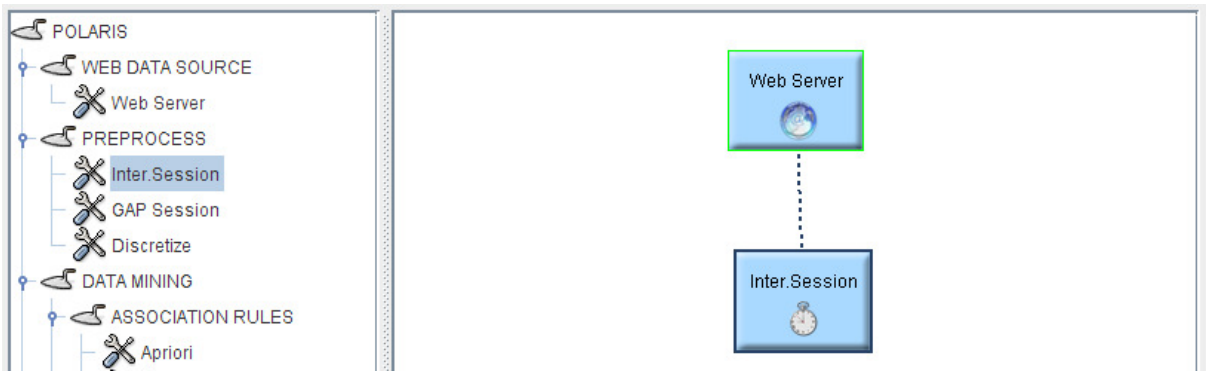
Se selecciona para el ejemplo, el elemento **Inter.Session** y se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo).

Una vez arrastrado el Inter.Session, existen dos opciones para hacer en el momento:

- Se puede configurar y ejecutar el nodo
- Se puede hacer la conexión con el Web Server

Para el ejemplo, se conecta Inter.Session con Web Server: un clic derecho en el nodo y se selecciona la opción Add Connection y luego se escoge el nodo Web Server.

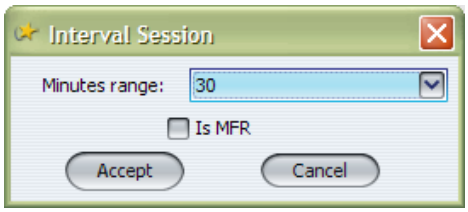
Figura 3.15. Conexion Inter.Session con Web Server.



6.5. CONFIGURAR UN NODO SESION

Lo primero que se debe hacer es un clic derecho en el nodo Inter.Session, y seleccionar la opción Settings, entonces aparecerá una pequeña ventana:

Figura 3.16. Interval Session Form.



Hay dos opciones de configuración:

- Minutes range, es decir, rango en minutos. Se escoge el intervalo en minutos de la sesión, este puede ser (10, 20, 30, 40, 50, 60 minutos) como predeterminado esta el rango de 30 minutos (esto porque es el periodo de intervalo estándar).

- Is MFR. Se habilita o deshabilita esta opción, que se encarga de decir si se realizan transacciones con Maximal Foware Reference, esto en caso de ser un log extendido.

Para este caso se toman los valores por defecto (30 minutos como rango y deshabilitado IsMFR) y Aceptar.

6.6. EJECUTAR UN NODO SESIÓN

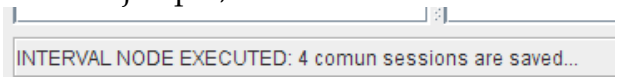
Una vez configurado el nodo (porque si no se configura, el nodo no se podrá ejecutar) se hace clic derecho en el nodo y se selecciona la opción **Run**.

Al terminar la ejecución el nodo cambia de estado y aparece un mensaje en la barra de estado con el resultado de la ejecución:

- Si todo ha resultado con éxito el mensaje es:

INTERVAL NODE EXECUTED: ***(Cantidad de sesiones) (Tipo Sesion)*** *sessions are saved…*

Para el ejemplo,



- Si ocurre algún error el mensaje es:

ERROR: ***(mensaje de error)***

6.7. ARRASTRAR UN NODO DE ASSOCIATION RULES Y CONECTAR A INTER.SESSION

Se selecciona para el ejemplo, el elemento **Apriori** y se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo).

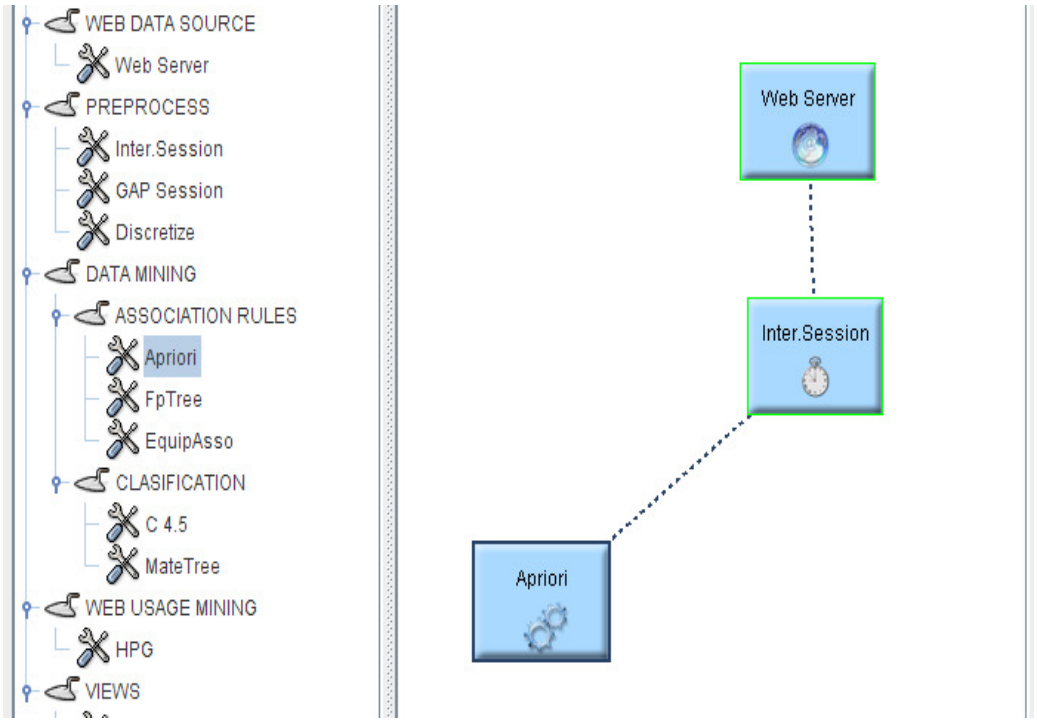
Una vez arrastrado el Apriori, existen dos opciones para hacer en el momento:

- Se puede configurar y ejecutar el nodo
- Se puede hacer la conexión con el Inter.Session

Para el ejemplo, se a conectar Apriori con Inter.Session: un clic derecho en el nodo y se selecciona la opción Add Connection y luego se escoge el nodo Inter.Session

La pantalla ahora se verá de la siguiente manera:

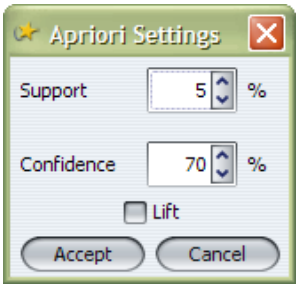
Figura 3.17. Conexión Apriori con Inter.Session.



6.8. CONFIGURAR UN NODO ASSOCIATION RULES

Lo primero que se debe hacer es dar clic derecho en el nodo Apriori, y seleccionar la opción Settings, entonces aparecerá una pequeña ventana:

Figura 3.18. A priori Settings Form.



Hay tres opciones de configuración:

- Support, Soporte. Es el porcentaje de soporte para la creación de los ítems frecuentes.
- Confidence, Confianza. Es el porcentaje de confianza para considerar una regla de asociación como válida.
- Lift, medida de sorpresa de una regla. Esta puede o no puede habilitarse y si se habilita se calculará y se observará su resultado.

Para este caso se toma los valores por defecto (Soporte 5%, Confianza 70% y Lift no aparece) y Aceptar.

6.9. EJECUTAR UN NODO ASSOCIATION RULES

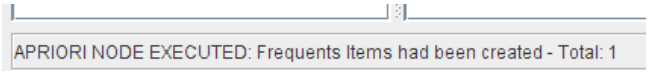
Una vez configurado el nodo (Porque si no se configura, el nodo no se podrá ejecutar) se hace clic derecho en el nodo y se selecciona la opción **Run**.

Al terminar la ejecución el nodo cambia de estado y aparece un mensaje en la barra de estado con el resultado de la ejecución:

- Si todo ha resultado con éxito el mensaje es:

APRIORI NODE EXECUTED: Frequents items had been created…

Para el ejemplo,



- Si ocurre algún error el mensaje es:

ERROR: ***(mensaje de error)***

6.10. ARRASTRAR DE PREPROCESS UN DISCRETIZE Y CONECTAR A INTER.SESSION

Se selecciona para el ejemplo, el elemento **Discretize** y se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo).

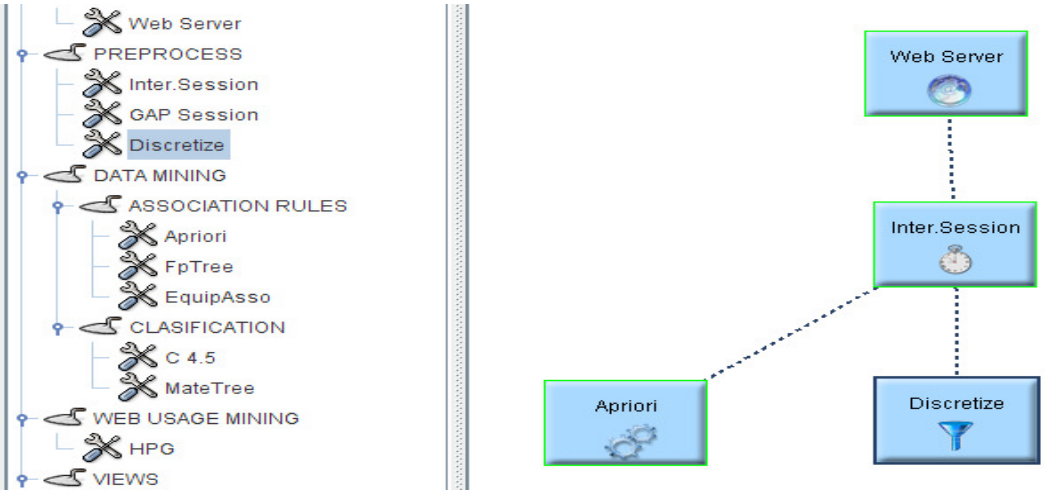
Una vez arrastrado el Discretize, existen dos opciones para hacer en el momento:

- Se puede configurar y ejecutar el nodo
- Se puede hacer la conexión con el Inter.Session

Para el ejemplo, se conecta Discretize con Inter.Session: clic derecho en el nodo y se selecciona la opción Add Connection y luego se escoge el nodo Inter.Session.

La pantalla ahora se verá de la siguiente manera (ver figura 3.19):

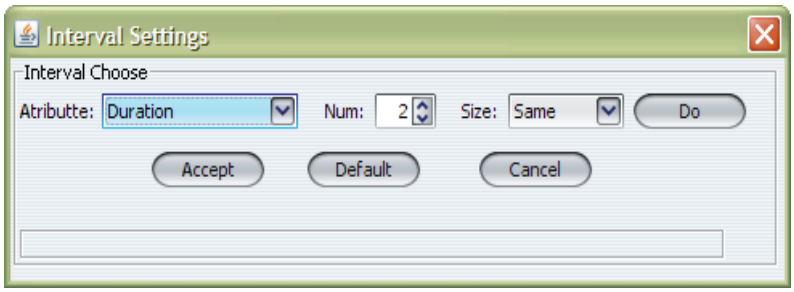
Figura 3.19. Conexión Discretize con Inter.Session



6.11. CONFIGURAR UN NODO DISCRETIZE

Lo primero que se debe hacer es: clic derecho en el nodo Discretize, y seleccionar la opción Settings entonces aparecerá una pequeña ventana:

Figura 3.20. Interval Settings Form.



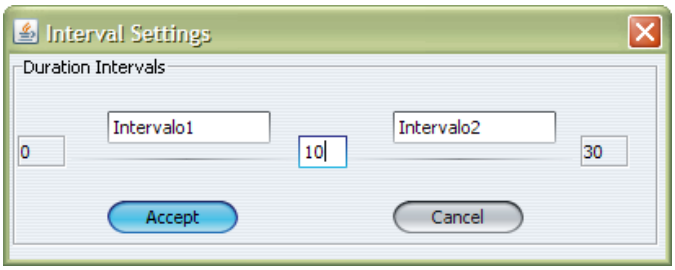
Hay tres opciones de configuración:

- Puede configurar los intervalos de 3 atributos de manera manual
- Puede configurar los 3 intervalos por defecto presionando el botón **Default**.
- En caso de no configurar el atributo, este tomara valores por defecto.

6.11.1. Configuración de atributo con Igual Tamaño de Intervalos. Se escoge el atributo de la lista **Atributte**, se escoge el número de intervalos de la lista **Num** – puede tener 2, 3, 4, 5 intervalos – y se escoge la opción **Same** de la lista **Size**. Entonces se presiona el botón **Do (Hacer)** e inmediatamente se construyen los intervalos del atributo seleccionado. Los nombres de los intervalos serán: *Intervalo **(# intervalo)***. Y aparece un mensaje en la barra de estado de la ventana así: ***(Atributo)*** *Interval is save…*

6.11.2. Configuración de atributo con diferente tamaño de intervalos. Se escoge el atributo de la lista **Atributte**, se escoge el número de intervalos de la lista **Num** – puede tener 2, 3, 4, 5 intervalos – y se escoge la opción **Diferent** de la lista **Size**. Entonces se presiona el botón **Do (Hacer)** e inmediatamente aparece la ventana de construcción de los intervalos así:

Figura 3.21. Interval Settings Duration Intervals Form.



El nombre del recuadro es: ***(Atributte)*** *Intervals*

A continuación se describe los elementos de la ventana:

- El recuadro donde hay un cero (0) es la caja de texto donde aparece el mínimo valor que puede tener el atributo.
- De ahí hay unas separaciones y según la cantidad de intervalos aparecen cajas de texto blancas, donde se colocan los limites de los intervalos, teniendo como primer intervalo el que empieza por el mínimo valor.
- El recuadro donde hay un treinta (30) es la caja de texto donde aparece el máximo valor que puede tener el atributo. El último atributo es el que tiene como limite superior el máximo valor.
- Aparecen además, sobre cada intervalo una caja de texto que contiene: *Intervalo****(# intervalo)***. Que viene siendo el nombre del intervalo, el cual puede ser cambiado, en caso que se quiera darle un nombre. Este es el que se mostrara en las visualizaciones tipo árbol.

Clic en Aceptar para construir los intervalos. Y aparece un mensaje en la barra de estado de la ventana así: ***(Atributo)*** *Interval is save…*

6.11.3. Configuración por defecto la configuración por defecto se activa así:

- Si no se entra a la opción Settings, este toma los valores por defecto.
- Si un atributo no es configurado, este toma los valores por defecto debido a que es necesario discretizarlo.
- Si se presiona el botón **Default** de la ventana Interval Settings.

Los intervalos por defecto son:

Tabla 2.38. Intervalos por Defecto

Atributo	Intervalos
Duration	Corta: Menos de 1/3 de sesión Media: Entre 1/3 y 2/3 de sesión Larga: Mayor a 2/3 de sesión
Request Amount	Pocas: Menos de 10 solicitudes Media: Entre 10 y 40 solicitudes Muchas: Mas de 40 solicitudes
Download Size	Baja: Menos de -2 Mb Media: Entre 2 y 10 Mb Alta: Mas de 10 Mb

Para este caso se toma los valores por defecto y Aceptar.

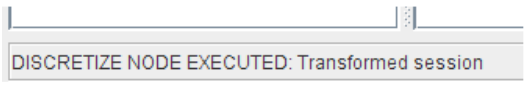
6.12. EJECUTAR UN NODO DISCRETIZE

Configurado o no (Porque si no se configura, el nodo tomará los valores por defecto) se hace clic derecho en el nodo y se selecciona la opción **Run.** Al terminar la ejecución el nodo cambia de estado y aparece un mensaje en la barra de estado con el resultado de la ejecución:

- Si todo ha resultado con éxito el mensaje es:

DISCRETIZE NODE EXECUTED: Transformed sessions

Para el ejemplo,



- Si ocurre algún error el mensaje es:

ERROR: ***(mensaje de error)***

6.13. ARRASTRAR UN NODO DE CLASIFICATION Y CONECTAR A DISCRETIZE

Se selecciona para el ejemplo, el elemento **C 4.5** y se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo).

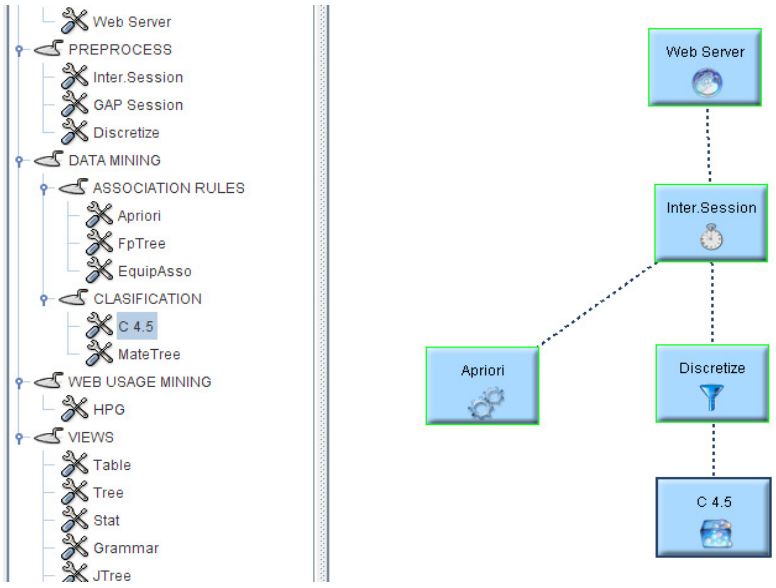
Una vez arrastrado el C 4.5, existen dos opciones para hacer en el momento:

- Se puede configurar y ejecutar el nodo
- Se puede hacer la conexión con el Discretize

Para el ejemplo, se conecta C4.5 con Discretize (esto porque para hacer clasificación es necesario discretizar los valores): clic derecho en el nodo y se selecciona la opción Add Connection y luego se escoge el nodo Discretize.

La pantalla ahora se verá de la siguiente manera (ver figura 3.22):

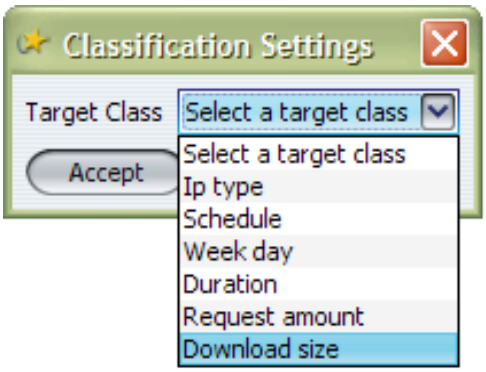
Figura 3.22. Conexión C4.5 con Discretize.



6.14. CONFIGURAR UN NODO CLASSIFICATION

Lo primero que se debe hacer es dar clic derecho en el nodo C 4.5, y seleccionar la opción Settings, entonces aparecerá una pequeña ventana:

Figura 3.23. Classification Settings Form.



Hay una opción de configuración:

- Target Class, Clase Etiqueta. Es el atributo del cual se hará el estudio de clasificación, de los 6 atributos de estudio, se escoge uno.

Para este caso se toma **Download size** y Aceptar.

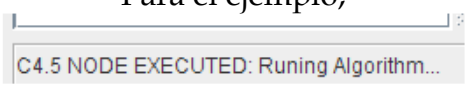
6.15. EJECUTAR UN NODO CLASSIFICATION

Una vez configurado el nodo (Porque si no se configura, el nodo no se podrá ejecutar) se hace clic derecho en el nodo y se selecciona la opción **Run**. Al terminar la ejecución el nodo cambia de estado y aparece un mensaje en la barra de estado con el resultado de la ejecución:

- Si todo ha resultado con éxito el mensaje es:

C 4.5 NODE EXECUTED: Running Algorithm…

Para el ejemplo,



- Si ocurre algún error el mensaje es:

ERROR: ***(mensaje de error)***

6.16. ARRASTRAR UN NODO DE WEB USAGE MINING Y CONECTAR A INTER.SESSION

Se selecciona para el ejemplo, el elemento **HPG** y se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo).

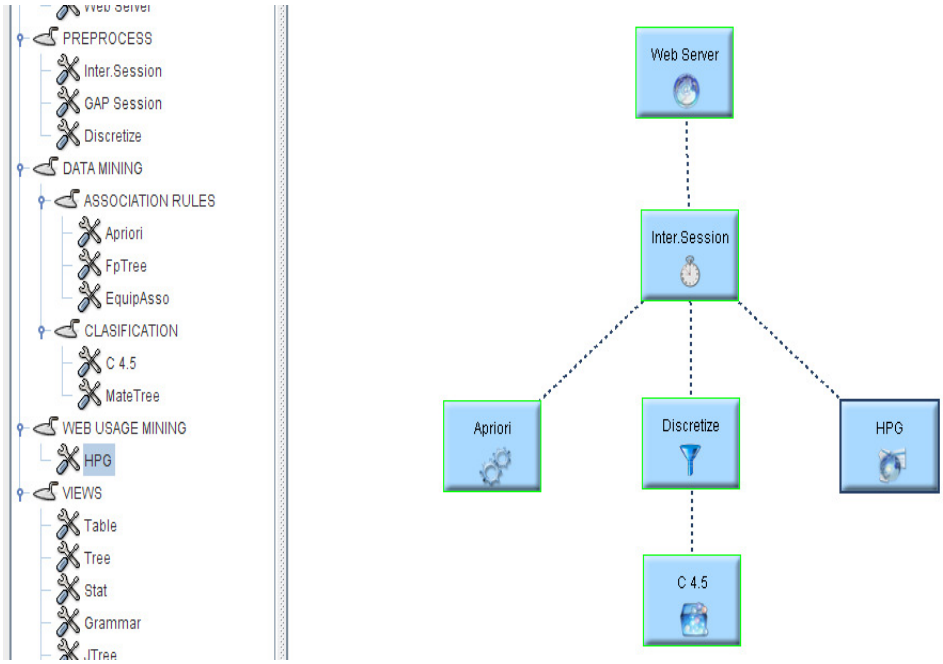
Una vez arrastrado el HPG, existen dos opciones para hacer en el momento:

- Se puede configurar y ejecutar el nodo
- Se puede hacer la conexión con el Inter.Session

Para el ejemplo, se conecta HPG con Inter.Session: clic derecho en el nodo y se selecciona la opción Add Connection y luego se escoge el nodo Inter.Session.

La pantalla ahora se verá de la siguiente manera:

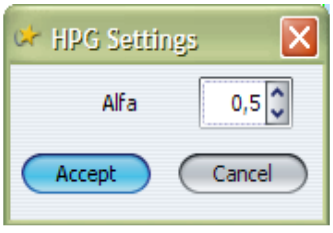
Figura 3.24. Conexión HPG con Inter.Session



6.17. CONFIGURAR UN NODO HPG

Lo primero que se debe hacer es dar clic derecho en el nodo HPG, y seleccionar la opción Settings, entonces aparecerá una pequeña ventana:

Figura 3.25. HPG Settings Form.



Hay una opción de configuración:

- Alfa. Este puede tomar 3 valores (1, 0, 0.5) e indica que páginas son tomadas como iniciales.

Para este caso **α = 0.5** y Aceptar.

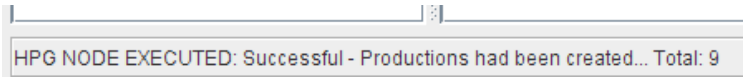
6.18. EJECUTAR UN NODO HPG

Una vez configurado el nodo (porque si no se configura, el nodo no se podrá ejecutar) se hace clic derecho en el nodo y se selecciona la opción **Run**. Al terminar la ejecución el nodo cambia de estado y aparece un mensaje en la barra de estado con el resultado de la ejecución:

- Si todo ha resultado con éxito el mensaje es:

HPG NODE EXECUTED: Successful – Productions had been created… Total: ***(# producciones)***

Para el ejemplo,



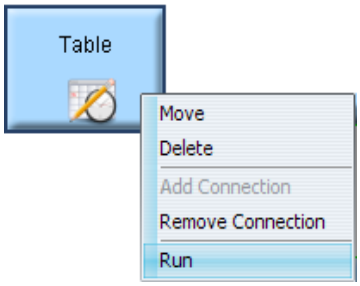
- Si ocurre algún error el mensaje es:

ERROR: ***(mensaje de error)***

6.19. ARRASTRAR UN NODO DE WEB VIEWS Y CONECTAR A UN NODO DE PROCESO (Cualquier nodo de categorías previas)

Un nodo **VIEW** presenta la visualización de un proceso del algoritmo y según sea el nodo al que se desee mirar los resultados será el nodo VIEW que se escogerá.

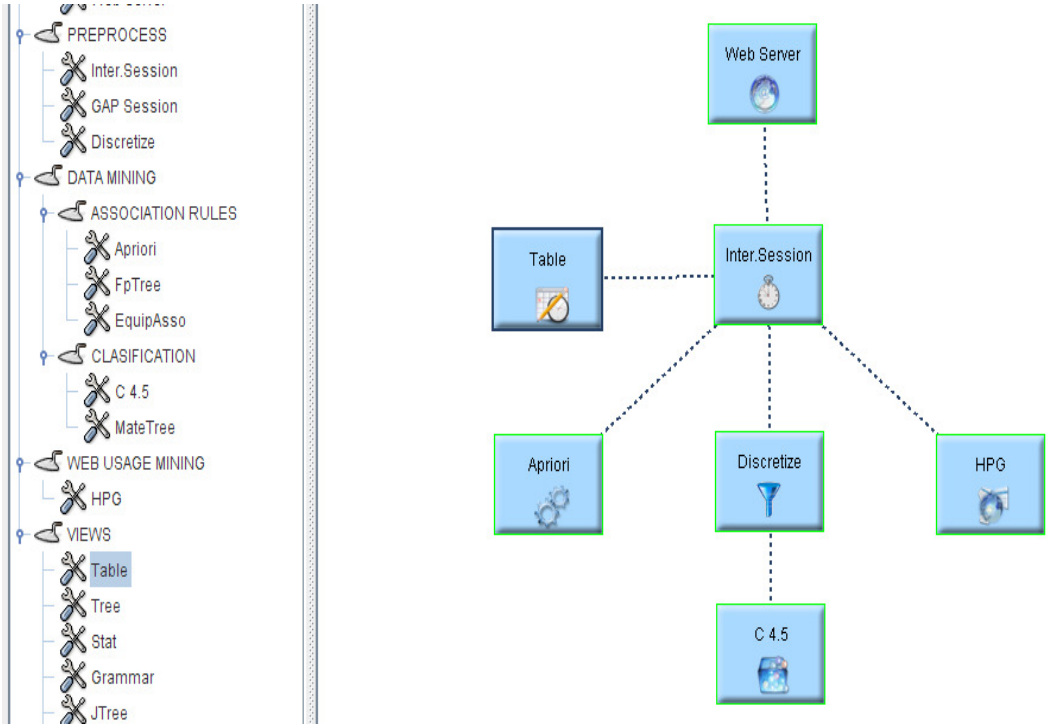
Figura 3.26. Nodo View.



Los nodos VIEW no tienen la opción **Settings**, así que solamente se ejecutan con la opción **Run**

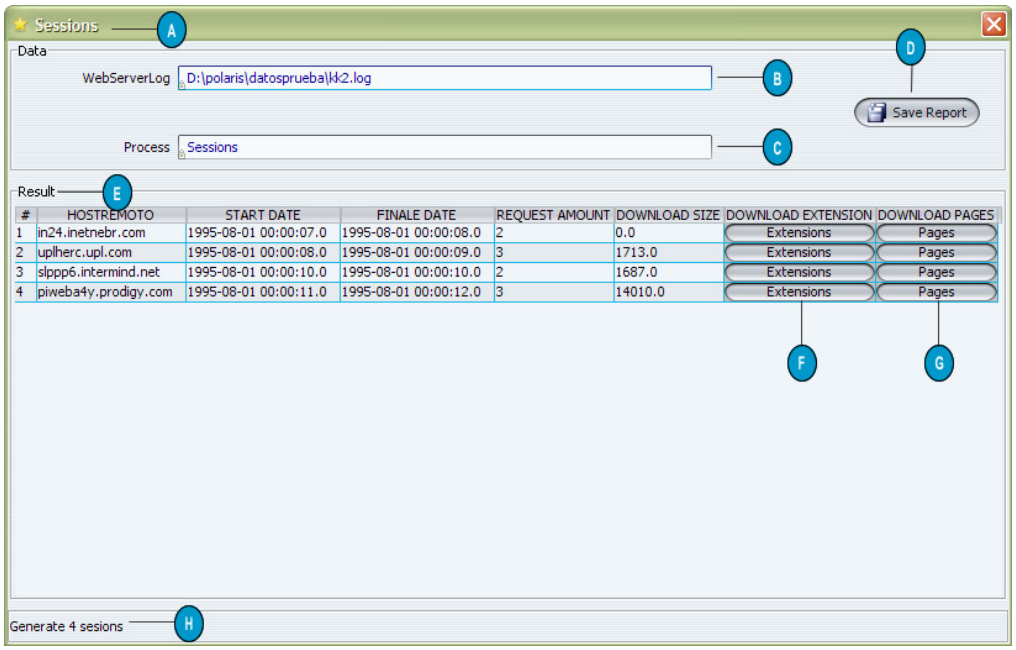
6.19.1. Arrastrar y conectar a un table. Se selecciona el nodo y lo se arrastra una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo). Una vez arrastrado el TABLE, se debe conectar a un nodo válido que desea visualizarse con este formato. Para el ejemplo, se conecta TABLE con Inter.Session: dando clic derecho en el nodo y se selecciona la opción Add Connection y luego se escoge el nodo Inter. La pantalla ahora se verá de la siguiente manera:

Figura 3.27. Conexión Table con Inter.Session



6.19.2. Ejecutar un nodo table. Se hace clic derecho en el nodo y se selecciona la opción **Run**. Al terminar la ejecución el nodo cambia de estado y aparece una ventana de visualización con una tabla de datos:

Figura 3.28. Visualización Tabla de Datos.



A – Barra de título.
B – Web Server Log, nombre del archivo analizado.
C – Process, nombre del proceso que se visualiza.
D – Save Report, opción para salvar reporte en formato html.
E – Result, muestra la tabla con los datos que resultan del proceso mencionado.
H – Barra de Estado, muestra la cantidad de registros resultantes.

En caso de que se conecten con un Nodo Sessions:

F – Botón Extensions, opción para ver las extensiones de la sesión.
G – Botón Pages, opción para ver las transacciones de la sesión.

- Si ocurre algún error el mensaje es:

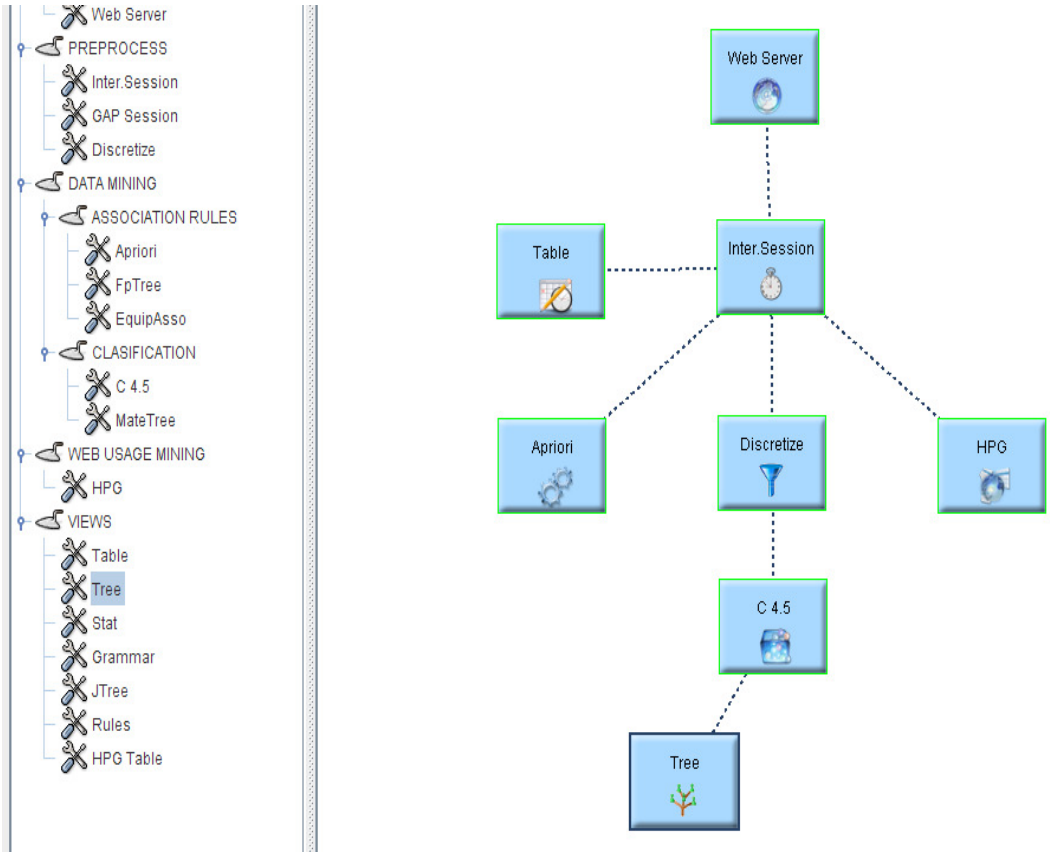
ERROR: ***(mensaje de error)***

6.19.3. Arrastrar y conectar a un tree. Se selecciona el nodo y lo se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo). Una vez arrastrado el TREE, se debe conectar a un nodo válido que desea visualizarse con este formato. Para el ejemplo, se a conectar TREE con C 4.5:

clic derecho en el nodo, se selecciona la opción Add Connection y luego se escoge el nodo C 4.5.

La pantalla ahora se verá de la siguiente manera:

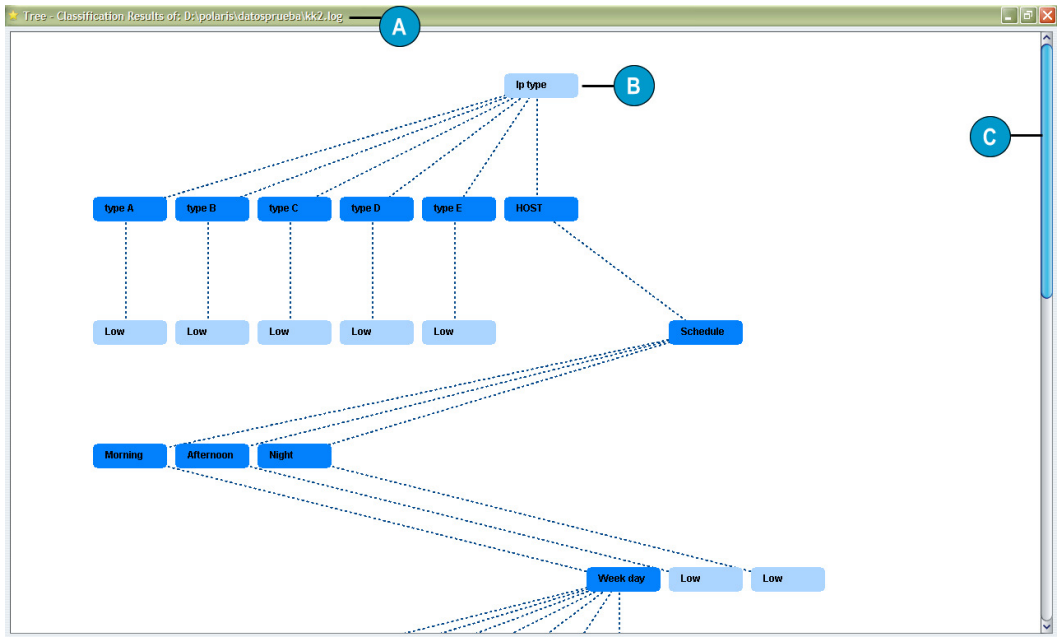
Figura 3.29. Conexión TREE con C4.5.



6.19.4. Ejecutar un nodo tree. Se hace clic derecho en el nodo y se selecciona la opción **Run.** Al terminar la ejecución el nodo cambia de estado y aparece:

- Una ventana de visualización con una tabla de datos:

Figura 3.30. Visualización Tree.



A – Barra de título: nombre de visualizador - proceso – nombre del archivo analizado
B – Árbol de toma de decisión resultante de los algoritmos de clasificación, nodos azul claro son raíz y hojas. Los nombres de los intervalos son los dados en la configuración del nodo Discretize.
C – Scroll, para bajar o subir por todo el tamaño del árbol resultante.

- Si ocurre algún error el mensaje es:

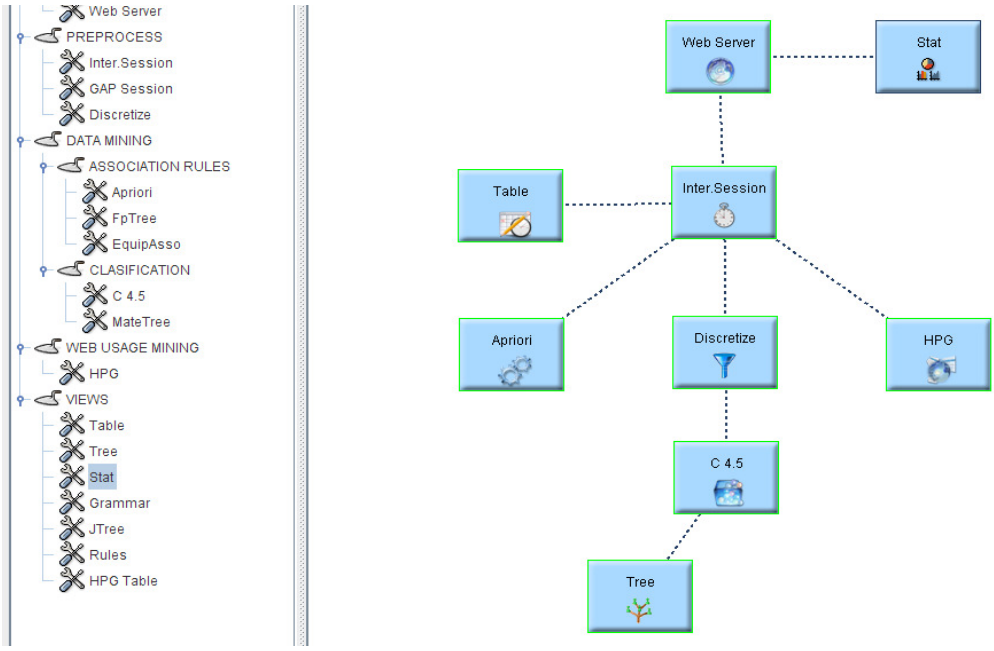
ERROR: ***(mensaje de error)***

6.19.5. Arrastrar y conectar a un stat (Statistical). Se selecciona el nodo y lo se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo). Una vez arrastrado el STAT, se debe conectar a un nodo válido que desea visualizarse con este formato. Para el ejemplo, se a

conectar STAT con Web Server: dando clic derecho en el nodo y se selecciona la opción Add Connection y luego se escoge el nodo Web Server .

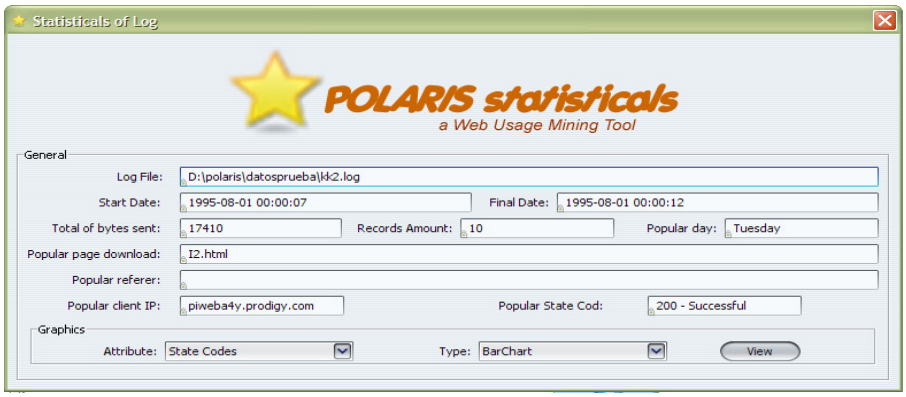
La pantalla ahora se verá de la siguiente manera:

Figura 3.31. Conexión STAT con Web Server.



6.19.6. Ejecutar un nodo stat. Se hace clic derecho en el nodo y se selecciona la opción **Run**. Al terminar la ejecución el nodo cambia de estado y aparece una ventana de visualización con una tabla de datos:

Figura 3.32. Visualización Statisticals.



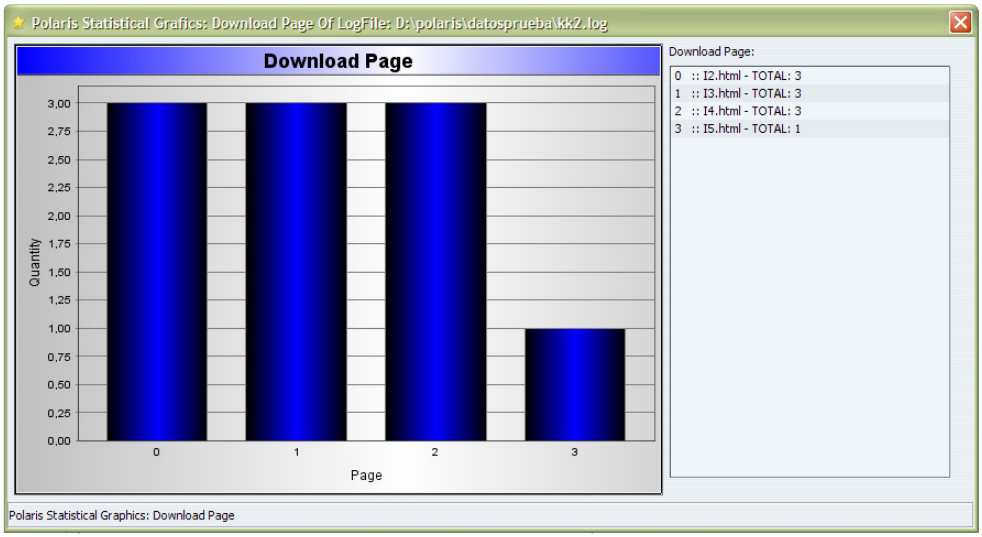
Ventana que muestra las estadísticas generales de un log, las que son:

- Nombre del archivo
- Fecha de inicio del log
- Fecha final del log
- Total de bytes enviados
- Cantidad de peticiones
- Día de mayor descarga
- Página más descargada
- Referencia más popular
- Cliente Ip más popular
- Código de estado más popular

Además aparece un recuadro **Graphics** que es para la visualización de gráficas estadísticas, así:

Se escoge de la lista **atributte** que estadística se va a realizar, y de la lista **type** selecciono la forma de visualización. Clic en el botón **View** y se observa:

Figura 3.33. Visualización Statisticals Barras.



En la barra de título la estadística y el nombre de archivo.
La gráfica de Barras que se escogió.
El listado de las páginas.

Pueden ser gráficas de barras, gráfica de pastel (si los elementos son menos de 100) y la gráfica lineal.

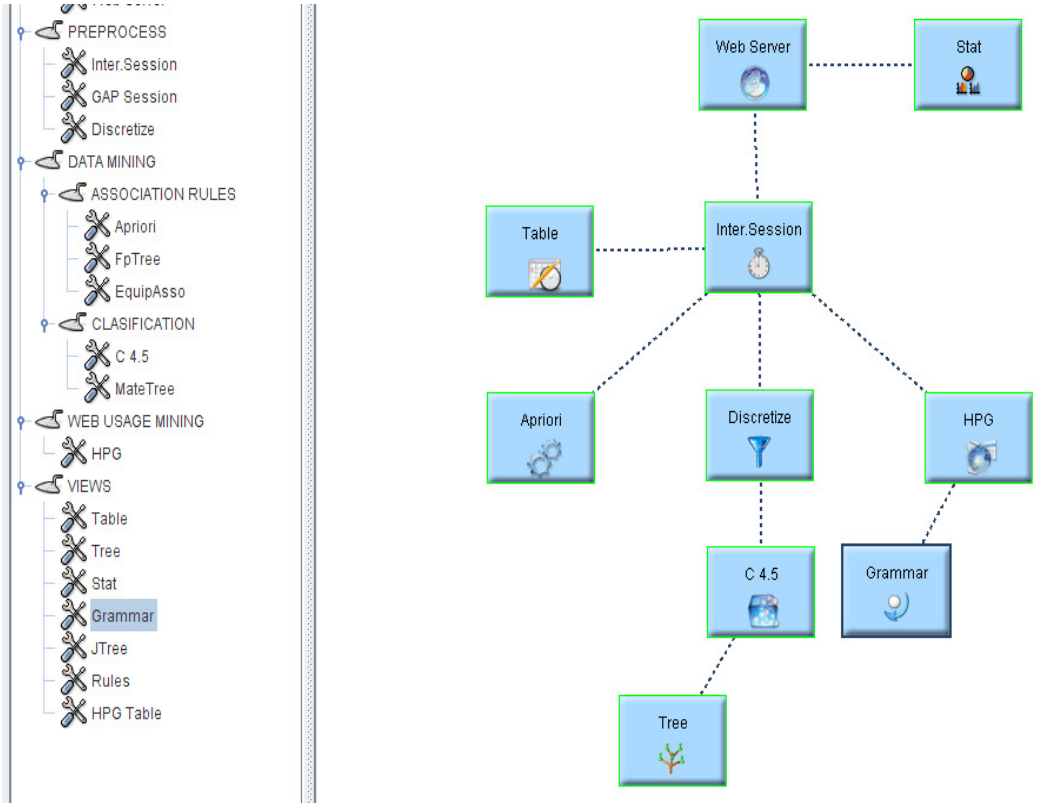
- Si ocurre algún error el mensaje es:

ERROR: ***(mensaje de error)***

6.19.7. Arrastrar y conectar a un gramar. Se selecciona el nodo y se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo). Una vez arrastrado el GRAMMAR, se debe conectar a un nodo válido que desea visualizarse con este formato. Para el ejemplo, se a conectar GRAMMAR con HPG: dando clic derecho en el nodo y se selecciona la opción Add Connection y luego se escoge el nodo HPG.

La pantalla ahora se verá de la siguiente manera:

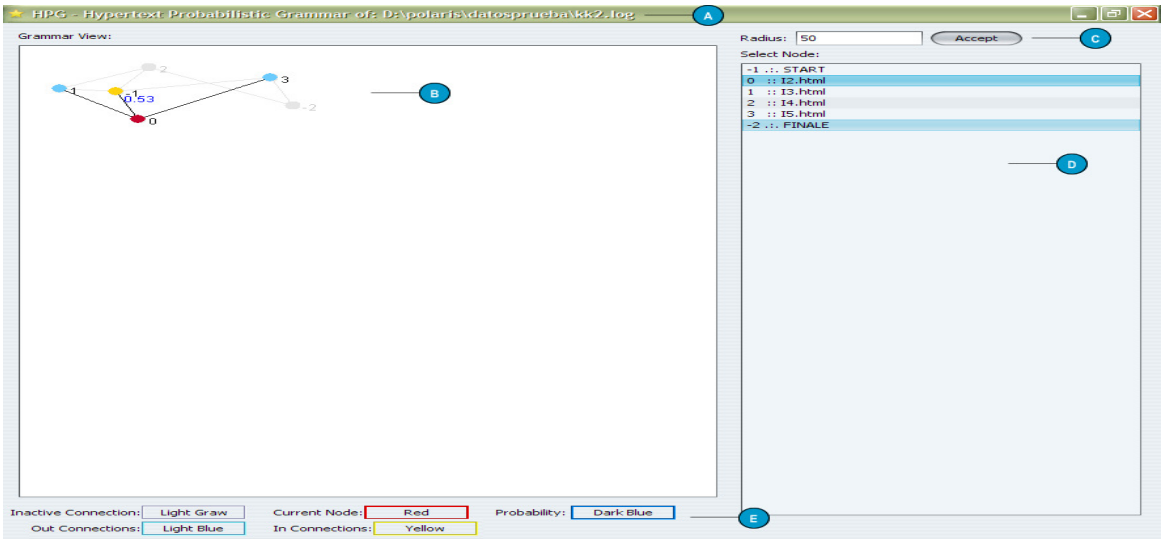
Figura 3.34. Conexión GRAMMAR con HPG.



6.19.8. Ejecutar un nodo gramar. Se hace clic derecho en el nodo y se selecciona la opción **Run**. Al terminar la ejecución el nodo cambia de estado y aparece:

- Una ventana de visualización con una tabla de datos:

Figura 3.35. Visualización Nodo Grammar.



A – Barra de título: nombre de Visualizador - proceso – nombre del archivo analizado
B – Grafo de la gramática resultante de los algoritmos de clasificación, grafo base en tono gris.
C – Radius: Manejo de tamaño del arco del grafo, predeterminado 50, en caso de cambio escribir en la caja de texto y aceptar.
D – Listado de las páginas que arman el nodo, cada fila contiene: Número con el que se identifica en el nodo y el nombre de la página.
E – Descripción de los colores que se manejan en el grafo.

Tabla 2.39. Descripción colores de los grafos.

Color		Descripción
Gris claro		Grafo base
Rojo		Nodo actual
Azul Oscuro		Probabilidad
Azul Claro		Nodos de donde sale para nodo actual
Amarillo		Nodos a donde parte desde nodo actual

Funcionamiento: Cuando se selecciona una página de la lista, en el grafo se visualiza el nodo actual, los nodos de referencia y los nodos hacia donde parte con la probabilidad.

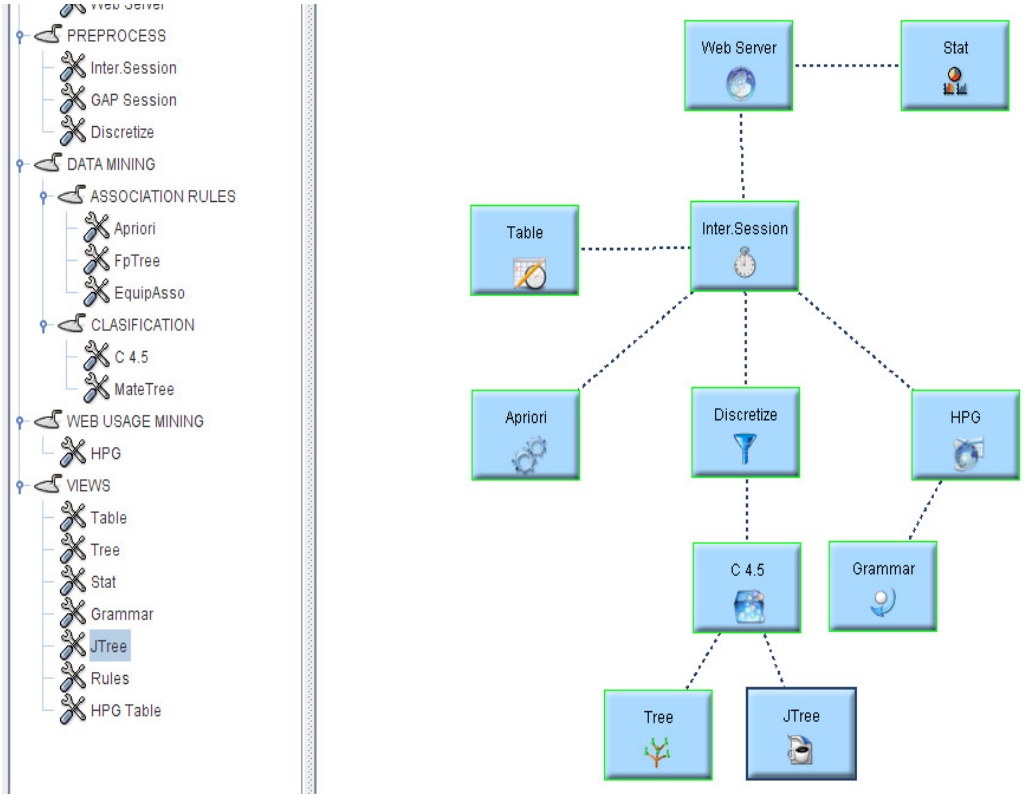
- Si ocurre algún error el mensaje es:

ERROR: ***(mensaje de error)***

6.19.9. Arrastrar y conectar a un jtree. Se selecciona el nodo y lo se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo). Una vez arrastrado el JTREE, se debe conectar a un nodo que desea visualizarse con este formato. Para el ejemplo, se a conectar JTREE con C 4.5: dando clic derecho en el nodo y se selecciona la opción Add Connection y luego se escoge el nodo C 4.5.

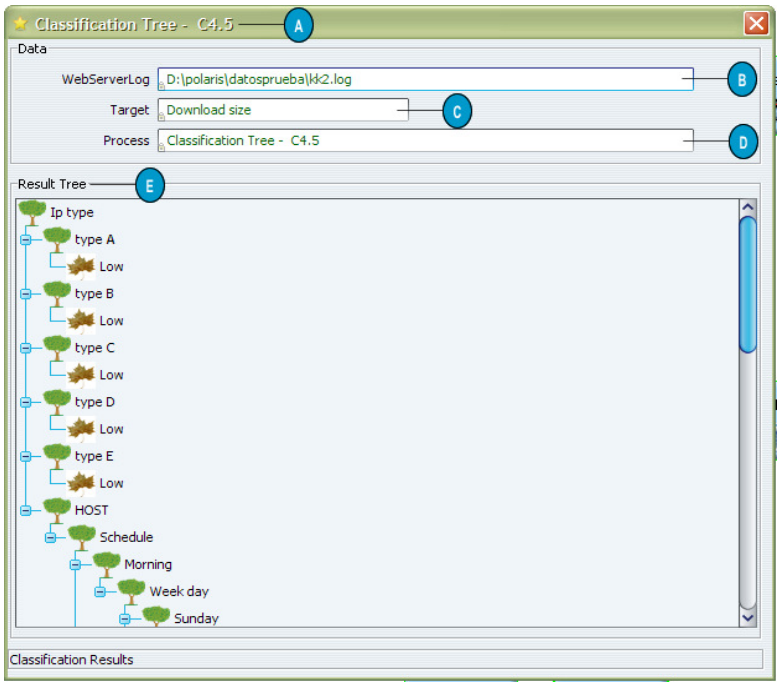
La pantalla ahora se verá de la siguiente manera:

Figura 3.36. Conexión JTree con C4.5



6.19.9.1. Ejecutar un nodo jtree. Se hace clic derecho en el nodo y se selecciona la opción **Run**. Al terminar la ejecución el nodo cambia de estado y aparece una ventana de visualización con una tabla de datos:

Figura 3.37. Classification Tree C4.5.



A – Barra de título: nombre de visualizador - proceso – algoritmo
B – Web Server Log, nombre del archivo analizado.
C – Target, clase que se analiza en el algoritmo de clasificación.
D – Process, nombre del proceso que se visualiza.
E – Árbol de toma de decisión resultante de los algoritmos de clasificación, Nodos hojas cargan imagen en forma de hoja. Los nombres de los intervalos son los dados en la configuración del nodo Discretize
C – Scroll, para bajar o subir por todo el tamaño del árbol resultante.

Si ocurre algún error el mensaje es:

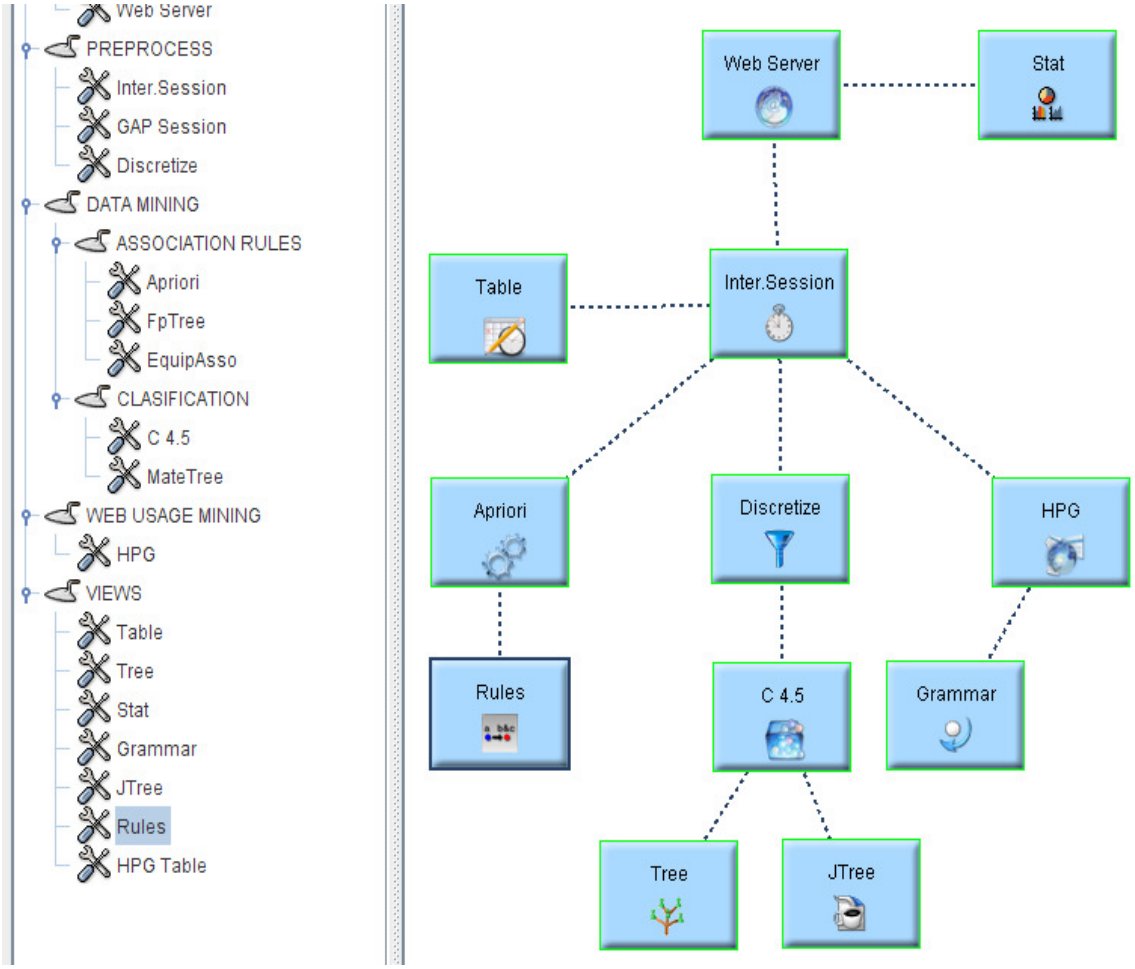
ERROR: ***(mensaje de error)***

6.19.10. Arrastrar y conectar un rules. Se selecciona el nodo y lo se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo). Una vez arrastrado el RULES, se debe conectar a un nodo válido que desea visualizarse con este formato.

Para el ejemplo, se a conectar RULES con Apriori: dando clic derecho en el nodo y se selecciona la opción Add Connection y luego se escoge el nodo A priori.

La pantalla ahora se verá de la siguiente manera:

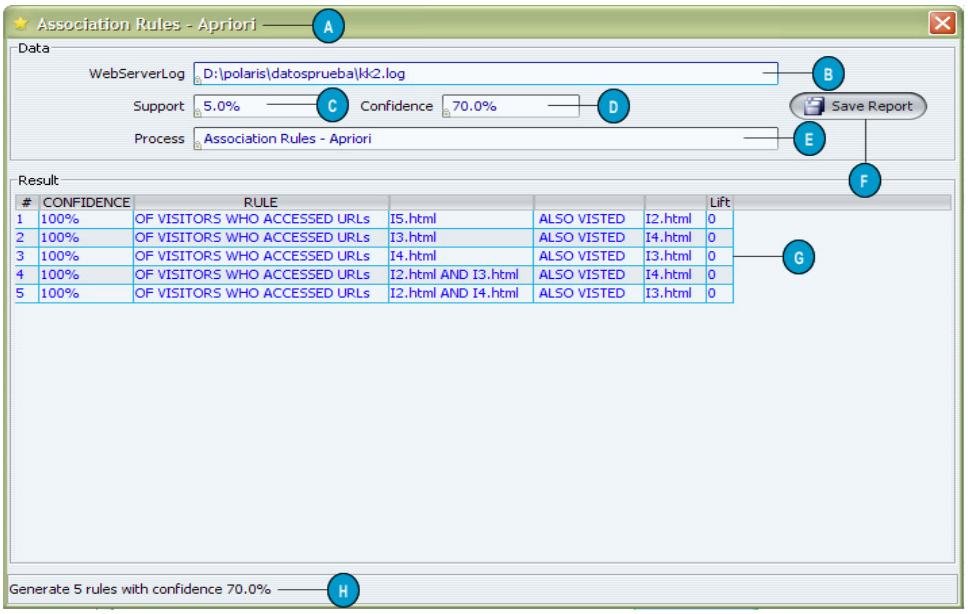
Figura 3.38. Conexión Rules con Apriori.



6.19.11. Ejecutar un nodo rules. Se hace clic derecho en el nodo y se selecciona la opción **Run**. Al terminar la ejecución el nodo cambia de estado y aparece:

- Una ventana de visualización con una tabla de datos:

Figura 3.39. Association Rules Apriori.



A – Barra de título. Proceso - algoritmo
B – Web Server Log, nombre del archivo analizado.
C – Soporte, porcentaje.
D – Confianza, porcentaje.
En caso de no ser proceso Association Rules, sino Classification en vez de los ítems C y D se mostrara la clase Target.
E – Process, nombre del proceso que se visualiza.
F – Save Report, opción para salvar reporte en formato html.
G – Result, muestra las reglas que resultan del proceso mencionado.
I – Barra de estado, muestra la cantidad de registros resultantes.

- Si ocurre algún error el mensaje es:

ERROR: ***(mensaje de error)***

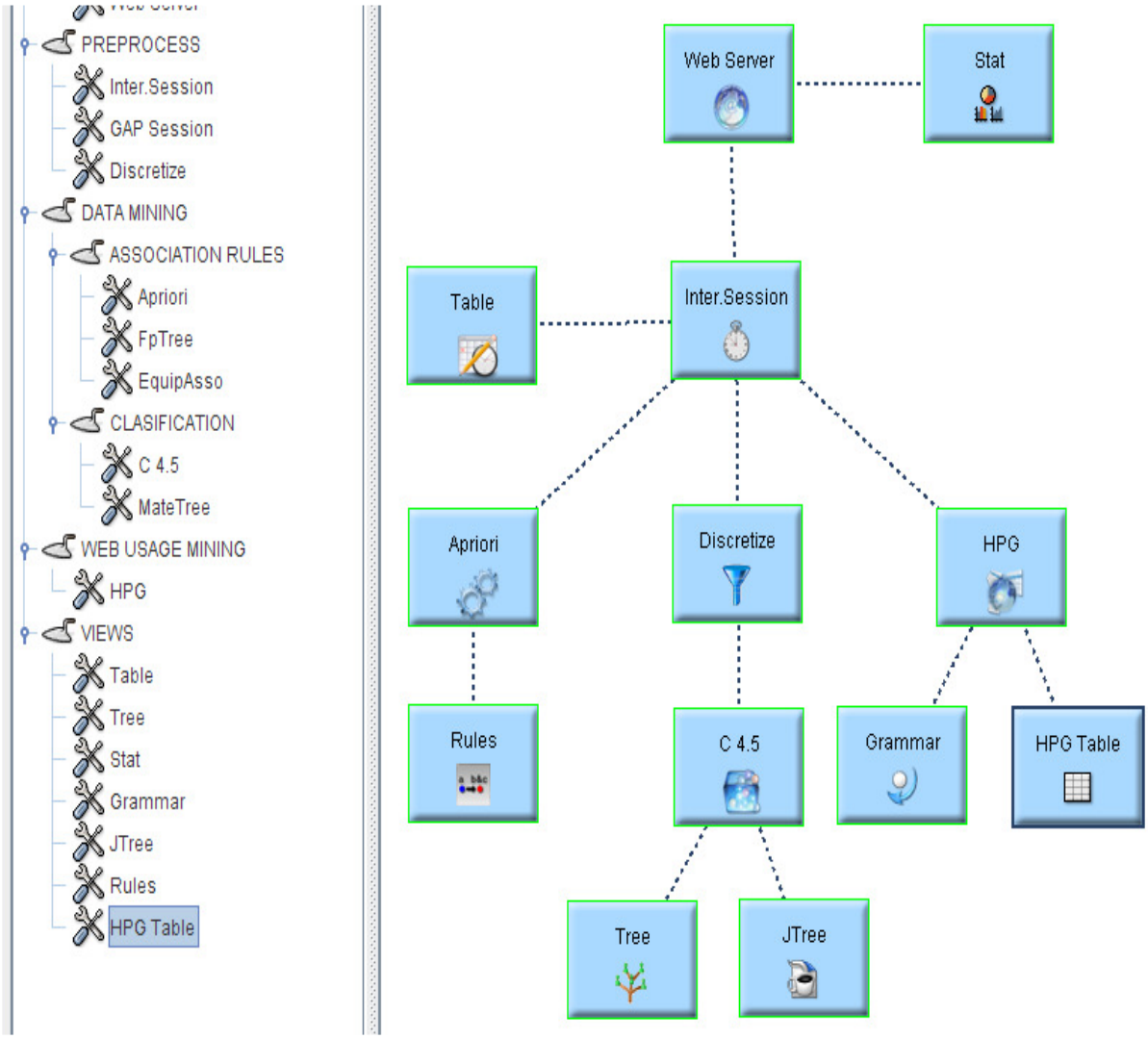
6.19.12. Arrastrar y conectar un HPGtable. Se selecciona el nodo y se arrastra a una posición cualquiera (preferiblemente una que permita una buena organización del algoritmo).

Una vez arrastrado el HPGTABLE, se debe conectar a un nodo válido que desea visualizarse con este formato.

Para el ejemplo, se a conectar HPGTABLE con HPG: dando clic derecho en el nodo y se selecciona la opción Add Connection y luego se escoge el nodo HPG.

La pantalla ahora se verá de la siguiente manera:

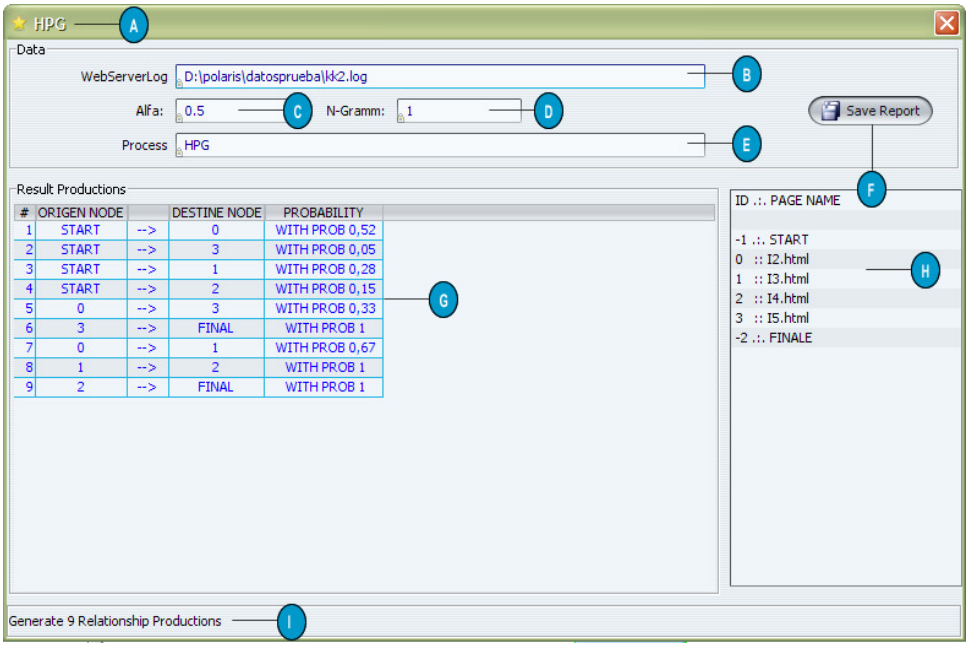
Figura 3.40. Conexión HPGTABLE con HPG.



6.19.13. Ejecutar un nodo HPGtable. Se hace clic derecho en el nodo y se selecciona la opción **Run**. Al terminar la ejecución el nodo cambia de estado y aparece:

- Una ventana de visualización con una tabla de datos:

Figura 3.41. Visualización HPGTABLE.



A – Barra de título. Proceso - algoritmo
B – Web Server Log, nombre del archivo analizado.
C – Índice alfa.
D – Modelo gramática.
E – Process, nombre del proceso que se visualiza.
F – Save Report, opción para salvar reporte en formato html.
G – Result, muestra las producciónes que resultan del proceso mencionado de la siguiente manera: nodo origen -> nodo destino con probabilidad.
H – Lista de páginas, muestra el identificador y la página correspondiente.
I – Barra de Estado, muestra la cantidad de producciones resultantes.

Si ocurre algún error el mensaje es:

ERROR: ***(mensaje de error)***

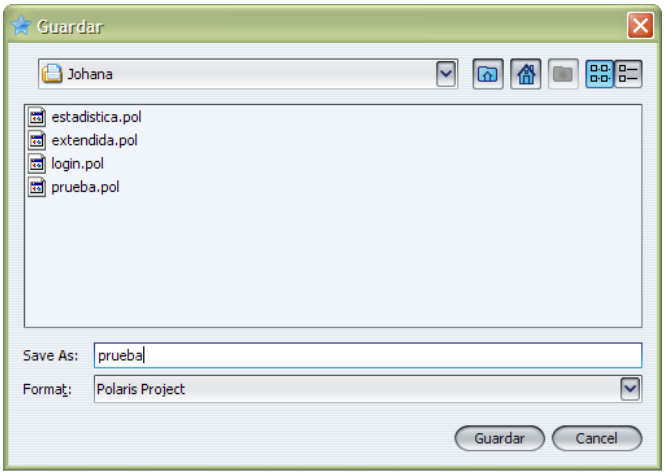
7. MANEJO DE PROYECTOS

Una vez se abre el software aparece cargado un nuevo proyecto, en el que se puede armar un algoritmo de minería de uso.

7.1. GUARDAR Y GUARDAR COMO

Pero también se puede guardar un proyecto presionando cualquiera de los dos botones: *Guardar y Guardar Como*.

Figura 3.42. Guardar Form.



Se abre una ventana para guardar el proyecto con extensión .pol (extensión de Polaris) y con extensión .xml (formato XML)

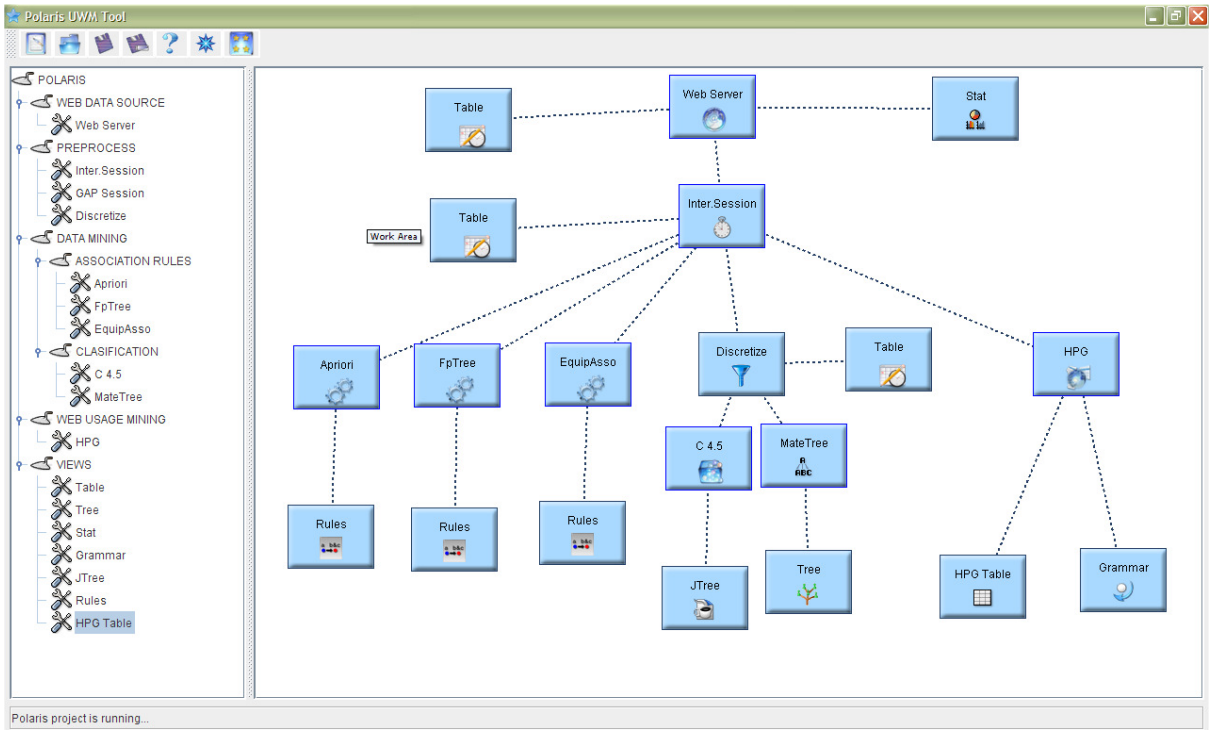
7.2. ABRIR

En caso de tener ya un proyecto guardado, este se abre presionando el botón *Abrir*.

Se abre una ventana que sirve para buscar en directorio proyectos con extensión .pol (extensión de Polaris) y con extensión .xml (formato XML). Se escoge un

archivo en este caso prueba.pol que terminará mostrando el algoritmo de minería de uso en el área de trabajo:

Figura 3.43. Proyecto Guardado.



8. OTRAS VENTANAS

8.1. AYUDA

Se presiona el botón de la barra de herramientas que indica la ayuda y se carga un navegador web la ayuda que lees en estos momentos.

8.2. ACERCA DE

Visualiza una ventana con las siguientes tres pestanas:

- Créditos

Figura 3.44. Créditos Form.

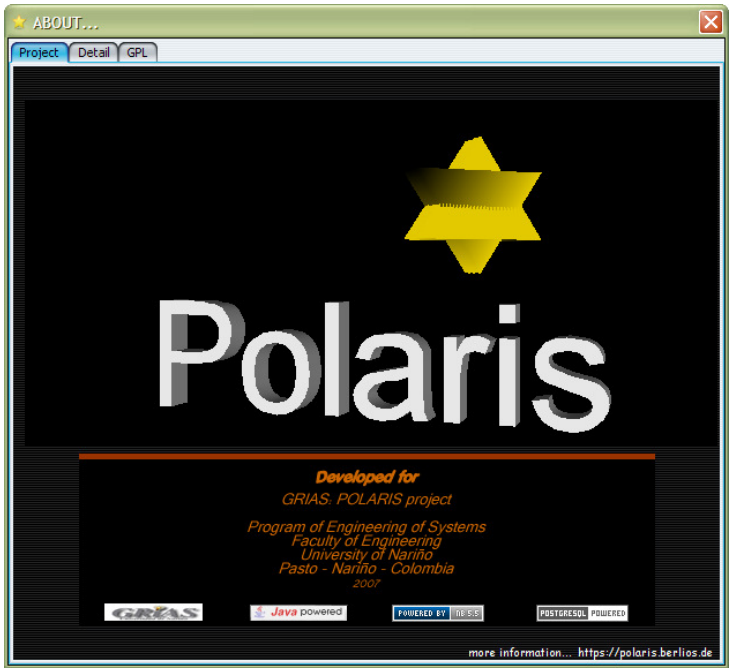
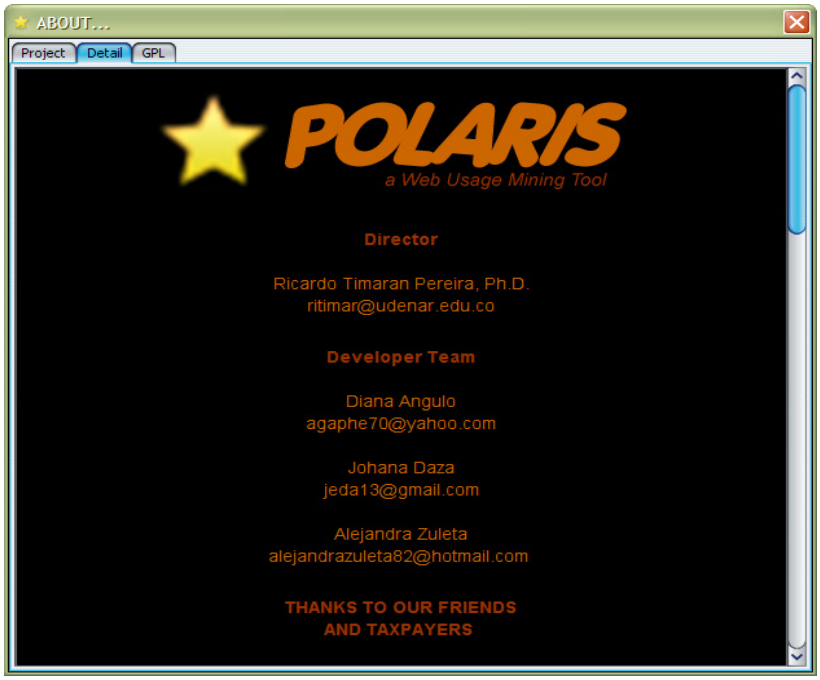
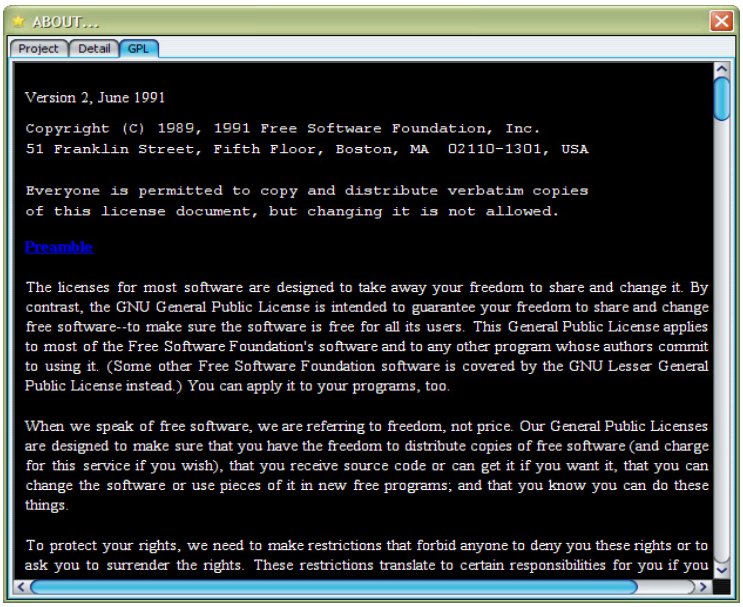


Figura 3.45. Detalles Créditos Form.



Licencia

Figura 3.46. Licencia Form.


ABOUT...
Project
Detail
GPL
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.
When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.
To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you

REFERENCIAS

[1] Agrawal 92

[2] Agrawal, R., Imielinski,T., Swami, A., Mining Association Rules between Sets of Items in Large Databases. ACM SIGMOD, 1993.

[3] Agrawal, R., Shim, K., Developing tightly coupled data mining applications on a relational database system. In The Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, 1996.

[4] Agrawal, R., Srikant, R., Fast algorithms for mining association rules. In VLDB Conference, Santiago, Chile, 1994.

[5] BILLSUS, Daniel., PAZZANI, Michael., Learning and Revising User Profiles: The Identification of Interesting Web Sites. Volume 27, Issue 3 (June 1997).

[6] Chaffee, Jason., Gauch, Susan., Personal ontologies for web navigation. McLean, Virginia, United States. Pags: 227 - 234. 2000. ISBN:1-58113-320-0.

[7] Chen, M.S., Park, J.S., Yu, P.S., Efficient data mining for path trasversal patterns. IEEE Transations on Knowldge and Data Engineering 10:2:209-221, 1998.

[8] Chen, Zhixiang., W, Ada., Fu., Tong, Frank C., Optimal algoritms for finding user acces session from very large Web Logs, Advances in Knowledge Discovery and Data Mining/PAKDD´02 Lecture Notes in Computer Science 2336 pages 290-296, 2002.

[9] Cooley, Robert., Tan, Pang-Ning., SRIVASTAVA, Jaideep., Discovery of interesting usage patterns from web data.

[10] Dr. Baeza – Yates, Ricardo., *Del HTML A La Fidelidad: Un modelo integral para el diseño web.*

[11] Dr. Baeza – Yates, Ricardo., *Excavando la Web.*

[12] Han, H., Pei, J., Mining frequent patterns by patterngrowth: Methodology and implications. In SIGKDD Explorations, volume 2:1420, 2000.

[13] Han, J., Kamber, M., Data Mining Concepts and Techniques. Morgan Kaufmann Publishers, 2001.

[14] Hartigan, J., *Clustering Algorithms*, John Wiley & Sons, New York, 1975.

[15] Hartigan, J., Wong, M., *Algorithm AS139: A k-means clustering algorithm*, Applied Statistics, VOL. 28, 100-108, 1979.

[16]

http://lieber.www.media.mit.edu/people/lieber/Lieberary/Letizia/Letizia.html.

[17] http://www.analog.cx .

[18] http://www.azuredesktop.com.

[19] http://www.boutell.com/wusage/download.html.

[20] http://www.DownloadAnalyzer.com.

[21] http://www.webmining.cl.

[22] http://www-2.cs.cmu.edu/afs/cs/project/theo-6/web-agent/www/project-

home.html.

[23] https://developer.berlios.de/projects/tariykdd/

[24] Imielnski, T., Mannila, H., A database perspective on knowledge discovery. In Communications of the ACM.

[25] Levene, M., Loizou G., A probabilistic approach to navigation in hypertext. Information Sciences: 114:165-186, 1999.

[26] Lic. KOBLINC, GUSTAVO D., *Web Mining: Estado Actual de Investigación.*

[27] Metha et. al 96.

[28] Mobasher, Bamshad., Srivastava, Jaideep., NAMIT, Jain., EUI-HONG, (Sam) Han. Web Mining: Pattern Discovery from World Wide Web Transactions.

[29] NASA National Aeronautics and Space Administration. Tropical cyclone windspeed indicator. http://pmesip. nsstc.nasa.gov/cyclone/.

[30] Object Oriented Design with Applications Benjamming Cummings 1991.

[31] Object Oriented Modeling and Design Prentice Hall 1991.

[32] Object-Oriented Software Enginnering: A Use Case Driven Approach Addison-Wesley 1992.

[33] PILIOPOULOU, FAULSTICH, M., UM, L. C., A Web Utilization Miner. Proceedings of the international workshop on the web and databases.

[34] Quinlan 93.

[35] Quinlan, J.R., Induction of decision trees. Machine Learning. 1986.

[36] Shafer et. al 96.

[37] Shiby et. al 98.

[38] Timarán R., Millán M., Machuca F., New AlgebraicOperators and SQL Primitives for Mining Association Rules, Proceedings of the IASTED International Conference Neural Networks and Computational Intelligence, Cancun, Mexico, 2003.

[39] Timarán, R., Descubrimiento de conocimiento en bases de datos: Una visión general. En Primer Congreso Nacional de Investigación y Tecnología en Ingeniería de Sistemas, Universidad del Quindío Armenia, Octubre 2002.

[40] Timarán, R., Mate-tree: un algoritmo para el descubrimiento de reglas de Clasificaciòn basado en operadores algebraicos. In 6Aa Conferencia

Iberoamericana en Sistemas, Cibernética e Informática CICI 2007, Orlando, Florida, EE.UU., Julio 2007.

[41] Timarán, R., Millán, M., Equipasso: An algorithm based on new relational algebraic operators for association rules discovery. In Fourth IASTED International Conference on Computational Intelligence, Calgary, Alberta, Canada, July 2005.

[42] Timarán, R., Nuevas Primitivas SQL para el Descubrimiento de Conocimiento en Arquitecturas Fuertemente Acopladas con un Sistema Gestor de Bases de Datos. PhD thesis, Universidad del Valle, 2005.

[43] Web Mining, Data Preparations.

[44] WebMining\instalacion\xtra\ayuda.htm.